Marketing Sales Analyser

Version 1.0 (2020-2021)

Computer Science (083) Project

Developed By

Ananya Kumar

Delhi Public School, R.K.Puram, New Delhi

www.dpsrkp.net

Index

Certificate

This is to certify that the Marketing Sales Analyser Computer Science project is developed by **Ananya Kumar** under my supervision in the session 2020-2021.

The work done by her is original.

Ms. Sarika Kaushal
Computer Science Teacher
Date: 6th January, 2021

Acknowledgement

I would like to express my sincere gratitude to my computer teacher Ms. Sarika Kaushal for her vital support, guidance and encouragement without which this project would not come forth from my side. Who helped me complete the project by giving ideas,thoughts and made this project easy and accurate.

I wish to thank my parents for their undivided support and interest who inspired me and encouraged me to go my own way, without which I would be unable to complete my project.

Reference

1. Classnotes
2. CS Bhasha Youtube Channel
3. Python Documentation
4. StackExchange

Introduction

Keeping in view of growing requirements of effective and efficient solutions for various problems, with my knowledge of Python and SQL, in this project, I have tried to provide a simple solution for the Marketing Sector.

The project has complete sales analysis solutions to take care of all the day to day purchases in a multipurpose shop.

I have divided the program into small user defined packages to take care of all future upgrades and expansion. The main.py program borrows functions from the database.py and csvfiles.py packages.

The following is the list of all functions in each package along with their description.

Package Name : database.py

Sno	Function Name	Description
1	create_db()	To create database
2	check()	To check for existing database/table
3	create_t1()	To create table user_info with the following attributes : CID, Name, Age, Gender, ItemID, Quantity, Date, Location
4	create_t2()	To create table items with the following attributes : ItemID, Name, Category, Price
5	create_t3()	To create table locality with the following attributes : AreaID, Name
6	fill_t3()	To fill the table locality with accepted values
7	fill_t1()	To add a row(s) to the table user_info
8	add_purchase()	To add purchase to the table user_info
9	take_user_data()	To input purchase details with data validation
10	add_item_data()	To input item details with data validation
11	display_all()	To print all records in table user_info
12	item_sales()	EXTRA
13	printtable()	To print tables in a tabular format (with horizontal and vertical lines)

Package Name : csvfiles.py

Sno	Function Name	Description
1	open_csv()	To open csv file given by user with exception handling
2	fill_t2()	To fill the table items with pre installed item data
3	csv_to_db()	To transfer csv data to the database
4	db_to_pd()	EXTRA

Package Name : main.py

Sno	Function Name	Description
1	printmenu()	To print the menu
2	set_up()	To create all the tables and fill them with pre installed data

Package Name : graphs.py

Sno	Function Name	Description
1	graphmenu()	To print the menu for graphical operations
2	main()	To run the graphical operations in a loop
3	barplot0()	To plot bar graph for comparing product wise sales
4	barplot1()	To plot bar graph for comparing gender wise sales for each product
5	barplot2()	To plot bar graph for comparing area wise sales
6	piechart0()	To plot pie chart for comparing area wise sales
7	piechart1()	To plot pie chart for comparing product wise sales

Tables with structure used in the program

Table Name: user_info

Sno	Field Name	Data Type	Description
1	CID	decimal(10,0)	Primary Key for customer phone number
2	Name	varchar(20)	Name of the customer
3	Age	decimal(2,0)	Age of the customer
4	Gender	char(1)	Gender of the customer (M/F)
5	ItemID	varchar(6)	Foreign Key; ItemID of the purchase
6	Quantity	decimal(5,2)	Numeric value for purchase quantity
7	Date	date	Purchase date
8	AreaID	varchar(6)	Foreign Key; 3 letter Area code based on locality table

Table Name: items

Sno	FieldName/ Column Name	Data Type	Description
1	ItemID	varchar(6)	Primary Key; ItemID for each item
2	Name	varchar(20)	Name of the item
3	Category	varchar(15)	Item category from five decided categories
4	Price	decimal(6,2)	Numeric price of the purchase

Table Name: locality

Sno	FieldName/ Column Name	Data Type	Description
1	AreaID	varchar(3)	Primary Key for 3 letter area code
2	Name	varchar(20)	Name of the locality

External Packages/Modules used in the Program/Software

1. pymysql
2. datetime
3. csv
4. pyfiglet
5. numpy
6. pandas
7. matplotlib (pyplot)

Salient Features of the Project

- User friendly menu driven options
- Exceptions used for date and csv file name to avoid inconvenience to users
- Data Validation for all purchase details to avoid ambiguous/wrong/invalid data entry :
    - CID - Cross checks entered phone number with previous entries. If no match is detected, a new user is added. If a match is found, then it compares the customer names and proceeds with data entry only if the names match
    - Name - No numeric values are accepted
    - Age - Only numeric values below 100 are accepted
    - Gender - Either male(M) or female(F)
    - ItemID - Only products added to the table items are accepted
    - Quantity - Only positive numeric values are accepted
    - Date - All dates are checked using datetime module
    - AreaID - Only area codes added to the table locality are accepted
- Data Validation for all purchase details to avoid ambiguous/wrong/invalid data entry :
    - ItemID - Only unique item ids are accepted
    - Name - No numeric values are accepted
    - Price - Only positive numeric values are accepted
    - Category - Only five categories are accepted - Electronic, Grocery, Stationary, Medicines and Clothing
- Preloaded csv file named Users.csv with sample data
- Preloaded csv file named InstalledInfo.csv with common purchase items
- Tabular representation of all tables for quick navigation
- Check for existence of tables and database
- Automatic creation and data addition for all three tables used
- Attractive and easy to understand graphs for comparing products
- Sales comparison on the basis of area, gender and product
- Direct addition of sales from CSV file provided to database

Source Code

```
# Project Title  : Marketing Sales Analyser
# Version        : 1.0 2020-2021
# Developed By   : Ananya Kumar
# Guide          : Sarika Kaushal
# Last Updated On: 2021-03-22

# To run the entire project

# t1 - User table with the following attributes :
#      CID, Name, Age, Gender, ItemID, Quantity, Date, Location
# t2 - Item table with price and item category
# t3 - Locality table with locality name, pincode and ID

# Importing libraries
import pymysql
import datetime
import pandas
import database as dbop
import csvfiles as cvf
import graphs as gr
from pyfiglet import figlet_format

result = figlet_format("        Sales", font = 'bulbhead')
result2 = figlet_format("    Analysis", font = 'bulbhead')
print(result, result2, sep='\n')

def printmenu() :
    print()
    print("A. Add purchase detail")
    print("B. Display all purchases")
    print("C. Add a new item")
    print("D. Import purchase history from csv file")
    print("E. Graphical Analysis")
    print("Q. Quit")
    print()

def set_up() :
    if not dbop.check('tables', 'user_info',cr) :
        dbop.create_t1(cr)

    if not dbop.check('tables', 'items',cr) :
        dbop.create_t2(cr)
        cvf.fill_t2(cr,db)

    if not dbop.check('tables', 'locality',cr) :
        dbop.create_t3(cr)
        cvf.fill_t2(cr,db)
```

```
# Connecting to MySQL
print("Connecting to MySQL... ","", sep='\n')
db = pymysql.connect(host = "localhost", user = "root", passwd = "@Ananya5492")
print("Database connected... ","", sep='\n')
cr = db.cursor()
print("Cursor created... ","", sep='\n')

# Creating database
dbop.create_db(cr)
set_up()

while True :
    printmenu()

    ch = input("Enter choice : ").upper()

    if ch == 'A':
        dbop.fill_t1(cr,db)
    elif ch == 'B' :
        dbop.display_all(cr,db)
    elif ch == 'C' :
        dbop.add_item_data(cr,db)
    elif ch == 'D' :
        cvf.csv_to_db(cr,db)
    elif ch == 'E' :
        gr.main(cr,db)
    elif ch == 'Q' :
        print("Thank you ...")
        break
    else :
        print("Invalid input, please try again")

# To run the database part of program

# Importing libraries
import pymysql
import datetime

def create_db(cr) :
    # Creating database
    if check("databases",'sales_analysis',cr) :
        cr.execute("use sales_analysis")
    else :
        cr.execute("create database sales_analysis")
        cr.execute("use sales_analysis")

def check(category, name, cr) : # to check for a table/database
    sql = "show " + category
    cr.execute(sql)
    re = cr.fetchall()
```

```
    for i in re :
        if i[0].lower() == name :
            return True
    return False

def create_t1(cr) : # user_info -> [CID, Name, Age, Gender, ItemID, Quantity,
Date, Location]
    sql = "create table user_info(CID decimal(10) primary key, Name
varchar(20), Age decimal(2,0),\
    Gender char(1), ItemID varchar(6), Quantity decimal(5,2), Date date, AreaID
varchar(6))"
    cr.execute(sql)

def create_t2(cr) : # items -> [ItemID, Name, Category, Price]
    sql = "create table items(ItemID varchar(6) primary key, Name varchar(20),
Category varchar(15),\
    Price decimal(6,2))"
    cr.execute(sql)

def create_t3(cr) : # locality -> [AreaID, Name]
    sql = "create table locality(AreaID varchar(3) primary key, Name
varchar(20))"
    cr.execute(sql)

def fill_t3(cr,db) :
    # Accepted localities -
    loc = [ ['North West Delhi', 'NWS'],
            ['North Delhi','NTH'],
            ['North East Delhi','NES'],
            ['Central Delhi','CEN'],
            ['New Delhi','NEW'],
            ['East Delhi','EAS'],
            ['South Delhi','STH'],
            ['South West Delhi','SWD'],
            ['West Delhi','WES']        ]

    # Adding each locality
    for area in loc :
        sql = 'insert into locality(AreaID, Name) values("' + area[1] + '","' +
area[0] + '")'
        cr.execute(sql)
        db.commit()

def fill_t1(cr,db) :
    # Taking user input [CID, Name, Age, Gender, ItemID, Quantity, Date,
Location]
    inputs = take_user_data(cr)

    # Adding user data
    add_purchase(inputs,cr,db)
```

```
    # Asking if user wants to add again
    ch = input("Do you wish to add another purchase ? (Y/N) :")
    if ch.upper() == 'Y' :
        fill_t1(cr,db)


def add_purchase(details,cr,db) : # For adding data to user_info table
    sql = "insert into user_info values( %s, %s, %s, %s, %s, %s, %s, %s )"
    cr.execute(sql,details)
    db.commit()

def take_user_data(cr) :
    # Taking customer name
    while True :
        name = input("Enter customer name : ")
        for i in name :
            if i.isnumeric() :
                print("Invalid name")
                break
        else :
            break

    # Taking customer ID
    while True :
        cusID = input("Enter phone number (without extention): ")
        if cusID.isnumeric() and len(cusID) == 10 :
            cr.execute('select Name, CID from user_info')
            re = cr.fetchall()
            if re == () :
                break
            for j in re :
                if j[1] == cusID and name == j[0].lower() :
                    print('User identified')
                    break
                elif j[1] != cusID and name != j[0].lower() :
                    print('New user added')
                    k = 0
                    break
            if k == 0: break
            print("Invalid number")

    # Taking customer age
    while True :
        age = input("Enter customer age : ")
        if age.isnumeric() and len(age) < 3 : break
        print("Invalid age")

    # Taking customer gender
    while True :
        gen = input("Enter customer gender (M/F): ")
        if gen.upper() in ['M', 'F'] : break
```

```
    print("Invalid gender")

# Taking item ID
while True :

    # Printing accepted ItemID values
    print("Accepted ItemID values :")
    cr.execute('select * from items')
    re = cr.fetchall()
    printtable(re)

    # Asking is user wants to add item
    ch = input("Do you wish to add a new item ? (Y/N) :")
    if ch.upper() == 'Y' :
        add_item_data(cr,db)

    t = True

    # Inputing item ID
    itemID = input("Enter item ID : ")
    cr.execute('select ItemID from items')
    re = cr.fetchall()
    for j in re :
        if itemID == j[0] :
            print("Item matched")
            t = False
    if not t :
        break
    print("Invalid item ID")

# Taking purchase quantity
while True :
    quan = input("Enter purchase quantity : ")
    if quan.isnumeric() and int(quan) > 0 : break
    print("Invalid quantity")

# Taking purchase date
while True :
    yr = input("Enter purchase year (YYYY): ")
    mon = input("Enter purchase month (MM): ")
    dd = input("Enter purchase date (DD): ")

    try :
        newDate = datetime.datetime(int(yr),int(mon),int(dd))
        date = '%s/%s/%s' % (yr, mon, dd)
        break
    except ValueError :
        print("Invalid date")

# Taking locality
while True :
```

```
        # Printing accepted Area ID values
        print("Accepted area ID values :")
        cr.execute('select * from locality')
        re = cr.fetchall()
        biglist = []
        for tup in re :
            biglist.append(list(tup))
        printtable(biglist)

        # Inputing item ID
        t = True
        loc = input("Enter area ID : ")
        cr.execute('select AreaID from locality')
        re = cr.fetchall()
        for j in re :
            if loc == j[0].upper() :
                t = False
        if not t : break
        print("Invalid area ID")

    details = (cusID, name, age, gen, itemID, quan, date, loc)

    return details

def add_item_data(cr,db) :
    # Taking item name
    while True :
        name = input("Enter item name : ")
        if name.isalpha() : break
        print("Invalid item name")

    # Taking item price
    while True :
        price = input("Enter item price : ")
        if price.isnumeric() : break
        print("Invalid price")

    # Taking item category
    while True :

        # Printing accepted categories -
        acats = ['Electronic', 'Grocery', 'Stationary', 'Medicines',
'Clothing']
        print("Accepted categories are : ")
        printtable([acats])

        # Inputing category
        cat = input("Enter item category : ")
        if cat.isalpha() and cat in acats : break
        print("Invalid category")
```

```
    # Taking item category
    while True :
        itemID = input("Enter new item id : ")
        cr.execute('select ItemID from items')
        re = cr.fetchall()
        if itemID in re :
            print("Item ID already exists.")
            pass
        else :
            print("Adding new item ... ")
            break

    # Adding item to items table
    T = (itemID, name, cat, price)
    sql = "insert into items(ItemID, Name, Category, Price) values(%s,%s,%s,%s)"
    cr.execute(sql, T)
    db.commit()

    # Printing added item
    print("Item added to items table :")
    cr.execute('select * from items')
    re = cr.fetchall()
    printtable(re)

    # Asking if another item is to be added
    ch = input("Do you wish to add another item ? (Y/N) :")
    if ch.upper() == 'Y' :
        add_item_data(cr,db)

def display_all(cr,db) :
    sql = "select * from user_info"
    cr.execute(sql)
    table = cr.fetchall()
    printtable(table)

def printtable(biglist) : # to print tablular stuctures using nested lists/tuples
    col = len(biglist[0])
    row = len(biglist)
    larg = [0 for i in range(col)]

    for row in biglist :
        for i in range(col) : #column
            if len(str(row[i])) > larg[i] :
                larg[i] = len(str(row[i])) + 1
    print()
    ch = sum(larg)
    print('|', "-" * (ch+col-1), '|', sep="")
    for row in biglist :
        print('|', end='')
        for ind in range(len(row)) :
```

```
            remaining = larg[ind] - len(str(row[ind]))
            print(row[ind], end = ' '*remaining + '|')
        print()
        print("|","-" * (ch+col-1), '|', sep="")
    print()

# To handle csv and panda related operations

# Importing libraries
import database as dbop
import pandas as pd
import csv

def open_csv() :
    while True :
        fil = input("Enter csv file name : ")
        fil += '.csv'

        try :
            sample = open(fil)
            return fil
        except FileNotFoundError :
            print("File Not Found")

def fill_t2(cr,db) :
    with open("InstalledInfo.csv", "r") as fil :
        records = csv.reader(fil)

        for record in records :
            row = tuple(record)
            sql = "insert into items values( %s, %s, %s, %s )"
            cr.execute(sql, row)
            db.commit()

def csv_to_db(cr,db) :
    fname = open_csv()

    with open(fname, "r") as fil :
        records = csv.reader(fil)

        for row in records :
            row = tuple(row)
            try :
                dbop.add_purchase(row,cr,db)
            except :
                print("This value could not be added :")
                print(dbop.printtable([row]))
        print("All csv file records successfully added")
```

```
# To plot graphs

# Importing libraries
import matplotlib.pyplot as plt
import pymysql
import datetime
import pandas as pd
import database as dbop
import csvfiles as cvf
import numpy as np

# Graphical Menu
def graphmenu() :
    print()
    print("Select a graph : ")
    print("A. Product wise comparison using bar graph")
    print("B. Gender wise product comparison using bar graph")
    print("C. Area wise sales using bar graph")
    print("D. Area wise sales using pie chart")
    print("E. Product wise sales using pie chart")
    print("F. Go back to main menu")
    print()

# Running all graphical operations
def main(cr,db) :

    while True :

        # Printing menu and asking choice
        graphmenu()
        ch = input("Enter choice : ").upper()

        if ch == 'A' :
            barplot0(cr,db)
        elif ch == 'B' :
            barplot1(cr,db)
        elif ch == 'C' :
            barplot2(cr,db)
        elif ch == 'D' :
            piechart0(cr,db)
        elif ch == 'E' :
            piechart1(cr,db)
        elif ch == 'F' :
            break
        else :
            print("Invalid input")

# Creating bar graph to compare product wise sale
def barplot0(cr,db) :

    # Declaring variables
```

```
pronames = []
ref = []
PRICE = []
FPRO = []

# Extracting details from item table
sql = "select * from items"
cr.execute(sql)
re = cr.fetchall()
for row in re :
    pronames.append(row[1])
    ref.append([row[0], row[1], int(row[3])])

# Setting width of bar
barWidth = 0.25
fig = plt.subplots(figsize =(6, 4))

# Setting height of bar
for pro in ref :
    sql = "select Quantity from user_info where ItemID = '" + pro[0] + "'"
    cr.execute(sql)
    price = 0
    while True :
        res = cr.fetchone()
        if res == None :
            break
        else :
            price += int(res[0]) * pro[2]
    if price != 0 :
        PRICE.append(price)
        FPRO.append(pro[1])

# Adding colors
n = len(PRICE)
c = ['pink', 'red', 'orange', 'yellow', 'green', 'cyan', 'blue', 'purple']
COLOR = []
if n>len(c) :
    j = n - len(c)
    COLOR += c
    COLOR += c[:j+1]
else :
    COLOR += c[:n+1]

# Setting position of bar on X axis
br1 = np.arange(n)

# Make the plot
plt.bar(br1, PRICE, width = barWidth, edgecolor ='grey',
        color = COLOR)

# Adding Xticks
```

```
    plt.xlabel('Product', fontweight ='bold')
    plt.ylabel('Sales', fontweight ='bold')
    plt.xticks([r for r in range(n)], FPRO)
    plt.title("Product wise comparison using bar graph")

    plt.show()

# Creating bar graph to compare gender wise product sale
def barplot1(cr,db) :

    # Declaring variables
    pronames = []
    ref = []
    MALE = []
    FEMALE = []
    FPRO = []

    # Extracting details from item table
    sql = "select * from items"
    cr.execute(sql)
    re = cr.fetchall()
    for row in re :
        pronames.append(row[1])
        ref.append([row[0], row[1], int(row[3])])

    # Setting width of bar
    barWidth = 0.25
    fig = plt.subplots(figsize =(6, 4))

    # Setting height of bar
    for pro in ref :
        sql = "select Gender, Quantity from user_info where ItemID = '" +
pro[0] + "'"
        cr.execute(sql)
        mprice = 0
        fprice = 0
        while True :
            res = cr.fetchone()
            if res == None :
                break
            else :
                if res[0].upper() == 'M' :
                    mprice += int(res[1]) * pro[2]
                elif res[0].upper() == 'F' :
                    fprice += int(res[1]) * pro[2]
        if mprice != 0 or fprice !=0 :
            MALE.append(mprice)
            FEMALE.append(fprice)
            FPRO.append(pro[1])

    # Setting position of bar on X axis
```

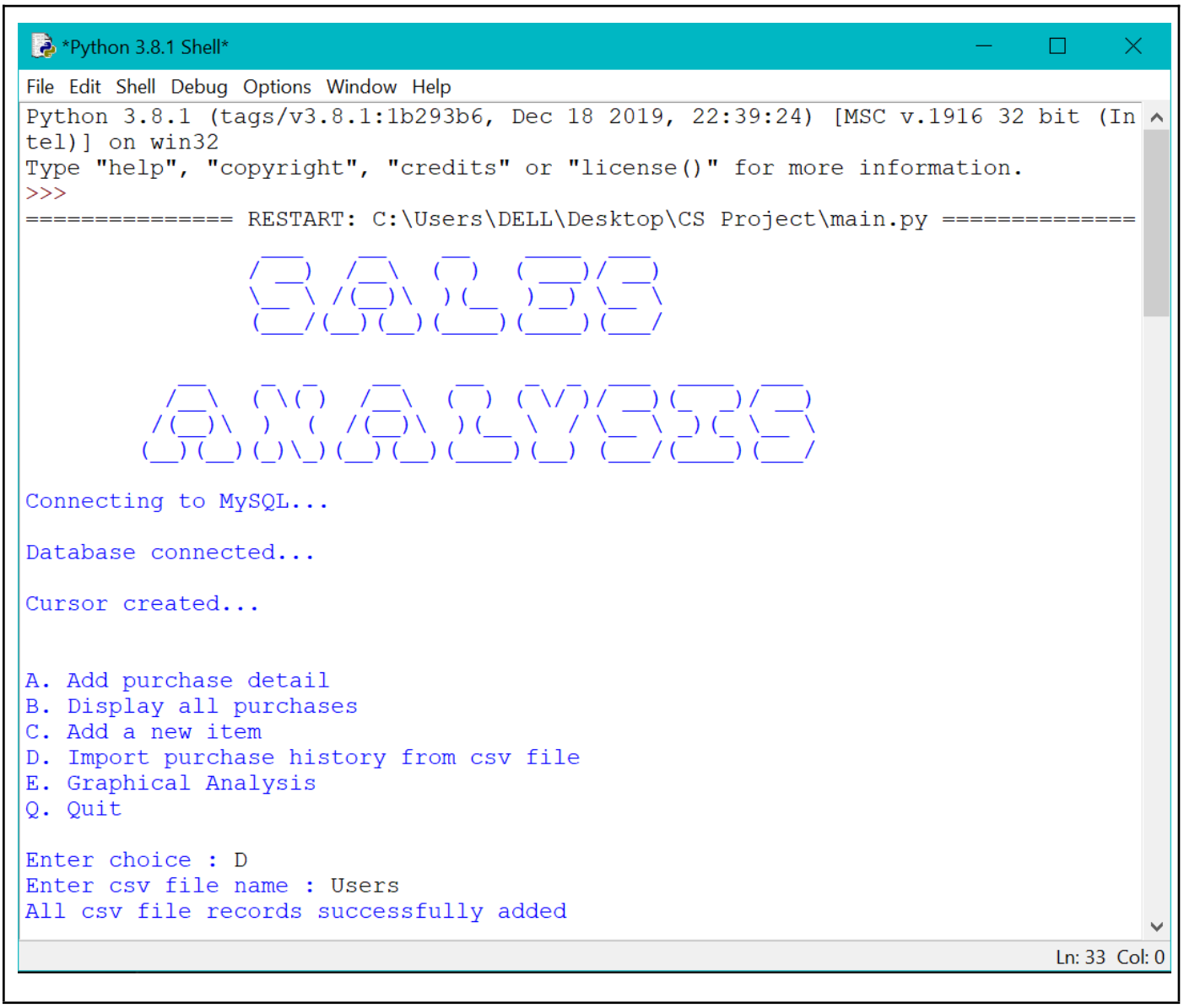
```
    br1 = np.arange(len(MALE))
    br2 = [x + barWidth for x in br1]

    # Make the plot
    plt.bar(br1, MALE, color ='cyan', width = barWidth, edgecolor ='grey',
label ='Male')
    plt.bar(br2, FEMALE, color ='pink', width = barWidth, edgecolor ='grey',
label ='Female')

    # Adding Xticks
    plt.xlabel('Product', fontweight ='bold')
    plt.ylabel('Sales', fontweight ='bold')
    plt.xticks([r + barWidth for r in range(len(MALE))], FPRO)
    plt.title("Gender wise product comparison using bar graph")

    plt.show()

# Creating bar graph to compare area wise product sales
def barplot2(cr,db) :

    # Declaring variables
    pronames = []
    SALE = {'NTH':0, 'NES':0, 'CEN':0, 'NEW':0, 'EAS':0, 'STH':0, 'SWD':0,
'WES':0}

    # Extracting details from item table
    sql = "select AreaID, Quantity, Price from user_info, items where
user_info.ItemID = items.ItemID"
    cr.execute(sql)
    re = cr.fetchall()
    for row in re :
        price = int(row[1]) * int(row[2])
        if price != 0 :
            pronames.append(row[0])
            SALE[row[0].strip()] += price

    # Setting width of bar
    barWidth = 0.5
    fig = plt.subplots(figsize =(6, 4))

    # Setting position of bar on X axis
    br1 = np.arange(len(SALE))

    # Make the plot
    plt.bar(br1, SALE.values(), width = barWidth, edgecolor ='grey',
            color=['pink', 'red', 'orange', 'yellow', 'green', 'cyan', 'blue',
'purple'])

    # Adding Xticks
    plt.xlabel('Area', fontweight ='bold')
    plt.ylabel('Sales', fontweight ='bold')
```

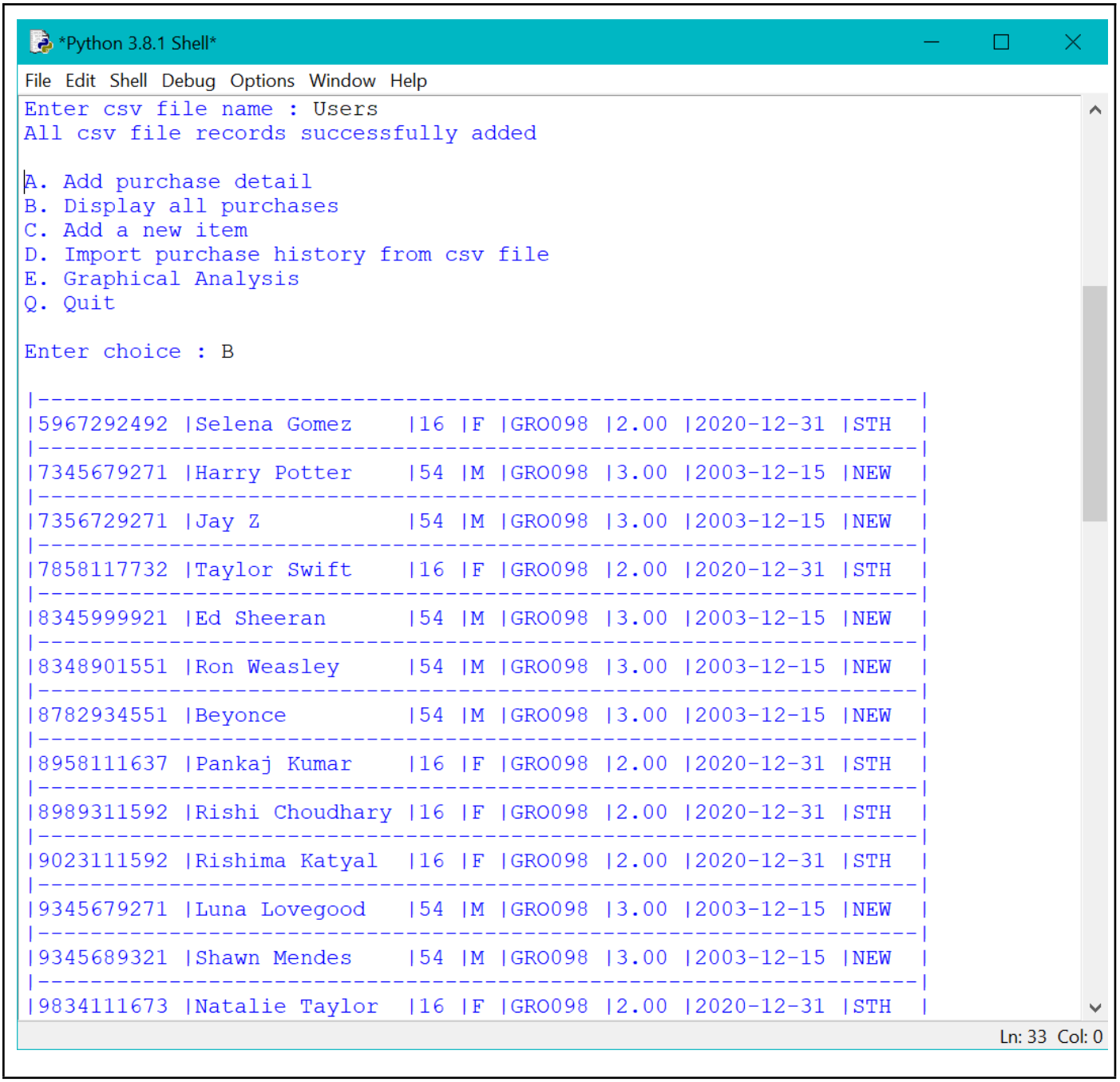
```
    plt.xticks([r for r in range(len(SALE.values()))], pronames)
    plt.title("Area wise sales using bar graph")

    plt.show()

# Creating pie chart to compare area wise sale
def piechart0(cr,db) :

    # Declaring variables
    SALE = {'NTH':0, 'NES':0, 'CEN':0, 'NEW':0, 'EAS':0, 'STH':0, 'SWD':0,
'WES':0}

    # Extracting details from item table
    sql = "select AreaID, Quantity, Price from user_info, items where
user_info.ItemID = items.ItemID"
    cr.execute(sql)
    re = cr.fetchall()
    for row in re :
        SALE[row[0].strip()] += int(row[1]) * int(row[2])

    # Removing values with no sales
    L = []
    for i in range(len(SALE.keys())) :
        if SALE[list(SALE.keys())[i]] == 0 :
            L.append(list(SALE.keys())[i])
    for i in L :
        del SALE[i]

    # Creating plot
    fig = plt.figure(figsize =(10, 7))
    plt.pie(SALE.values(), labels = SALE.keys())
    plt.title("Area wise sales using pie chart")

    # Show plot
    plt.show()

# Creating pie chart to compare product wise sale
def piechart1(cr,db) :

    # Declaring variables
    pronames = []
    ref = []
    PRICE = []
    FPRO = []

    # Extracting details from item table
    sql = "select * from items"
    cr.execute(sql)
    re = cr.fetchall()
    for row in re :
        pronames.append(row[1])
```

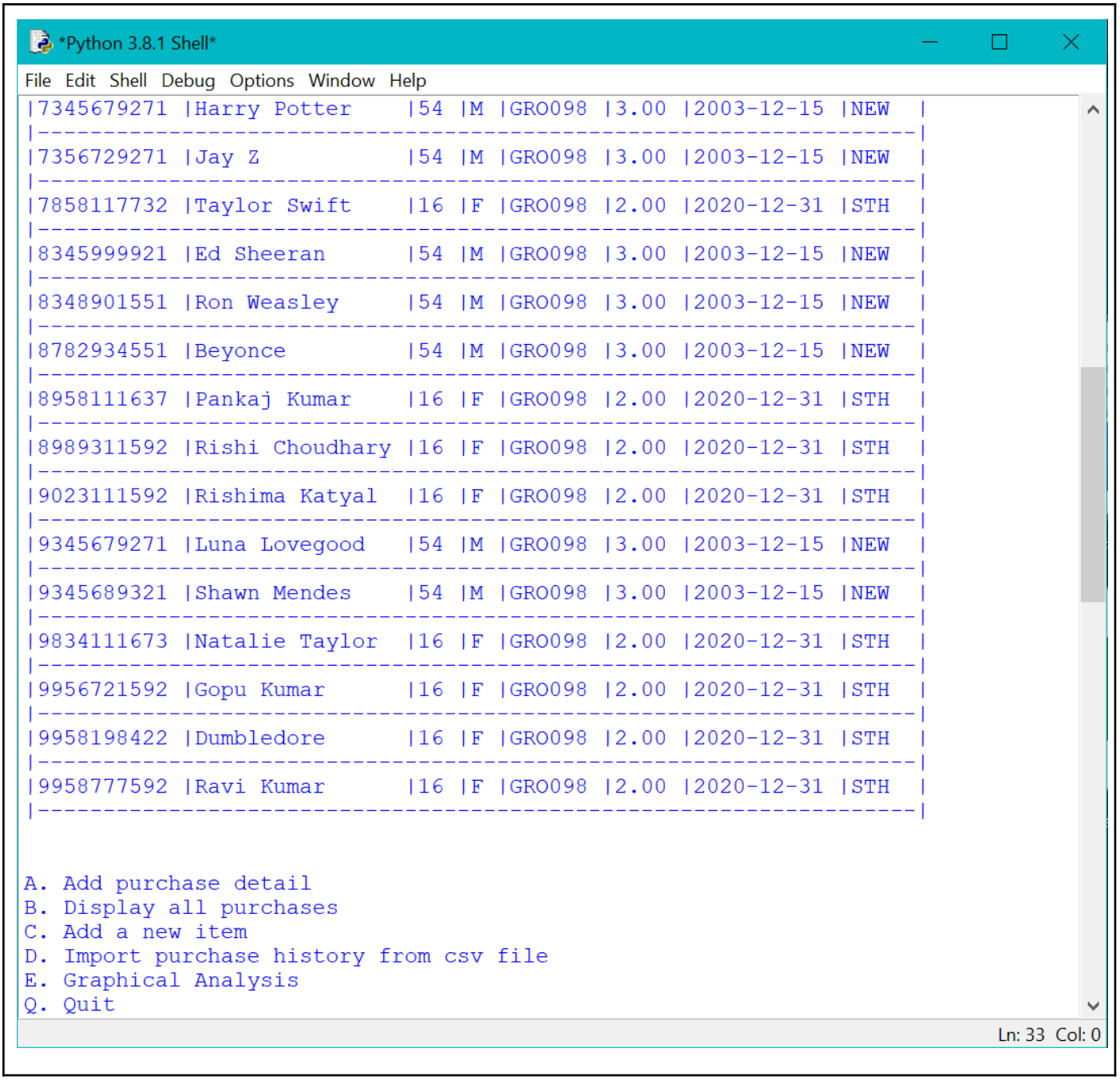
```
    ref.append([row[0], row[1], int(row[3])])

# Setting height of bar
for pro in ref :
    sql = "select Quantity from user_info where ItemID = '" + pro[0] + "'"
    cr.execute(sql)
    price = 0
    while True :
        res = cr.fetchone()
        if res == None :
            break
        else :
            price += int(res[0]) * pro[2]
    if price != 0 :
        PRICE.append(price)
        FPRO.append(pro[1])

# Creating plot
fig = plt.figure(figsize =(10, 7))
plt.pie(PRICE, labels = FPRO)
plt.title("Product wise sales using pie chart")

# Show plot
plt.show()
```

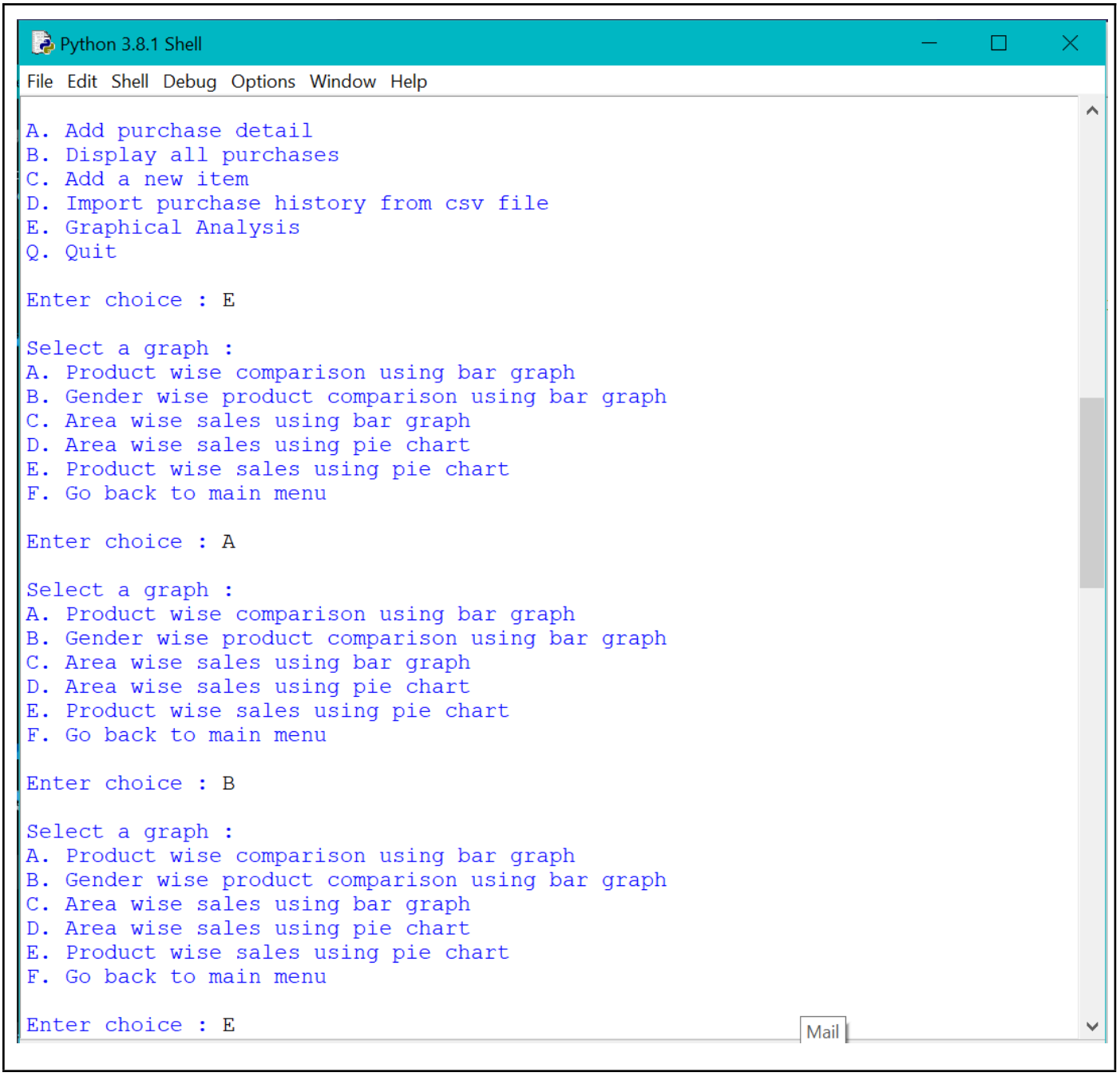
Output Screen

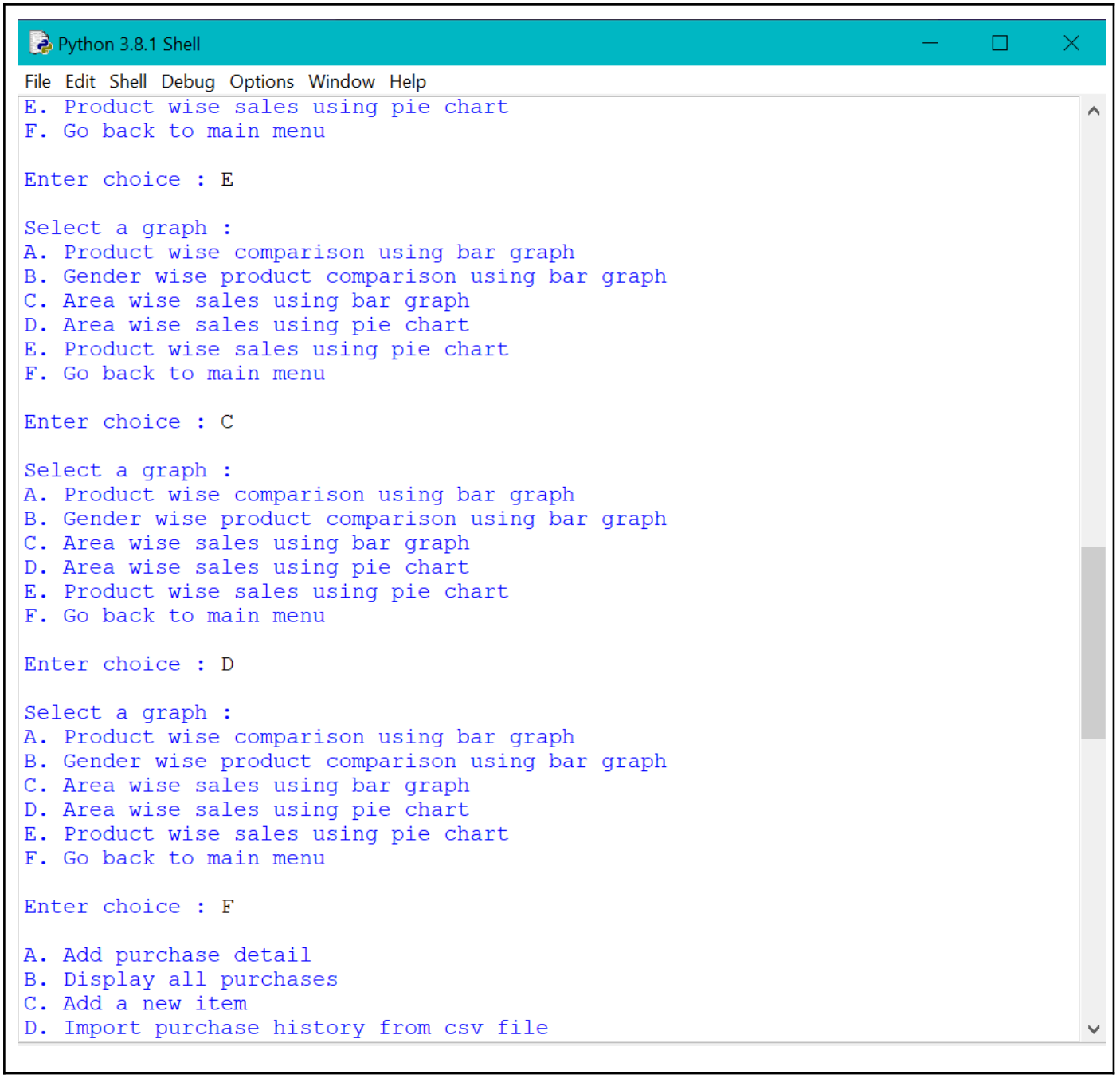
(All Operations)




Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Enter csv file name : Users
All csv file records successfully added
A. Add purchase detail
B. Display all purchases
C. Add a new item
D. Import purchase history from csv file
E. Graphical Analysis
Q. Quit
Enter choice : B
5967292492	Selena Gomez	16	F	GRO098	2.00	2020-12-31	STH
7345679271	Harry Potter	54	M	GRO098	3.00	2003-12-15	NEW
7356729271	Jay Z	54	M	GRO098	3.00	2003-12-15	NEW
7858117732	Taylor Swift	16	F	GRO098	2.00	2020-12-31	STH
8345999921	Ed Sheeran	54	M	GRO098	3.00	2003-12-15	NEW
8348901551	Ron Weasley	54	M	GRO098	3.00	2003-12-15	NEW
8782934551	Beyonce	54	M	GRO098	3.00	2003-12-15	NEW
8958111637	Pankaj Kumar	16	F	GRO098	2.00	2020-12-31	STH
8989311592	Rishi Choudhary	16	F	GRO098	2.00	2020-12-31	STH
9023111592	Rishima Katyal	16	F	GRO098	2.00	2020-12-31	STH
9345679271	Luna Lovegood	54	M	GRO098	3.00	2003-12-15	NEW
9345689321	Shawn Mendes	54	M	GRO098	3.00	2003-12-15	NEW
9834111673	Natalie Taylor	16	F	GRO098	2.00	2020-12-31	STH
Ln: 33 Col: 0


Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
7345679271	Harry Potter	54	M	GRO098	3.00	2003-12-15	NEW
7356729271	Jay Z	54	M	GRO098	3.00	2003-12-15	NEW
7858117732	Taylor Swift	16	F	GRO098	2.00	2020-12-31	STH
8345999921	Ed Sheeran	54	M	GRO098	3.00	2003-12-15	NEW
8348901551	Ron Weasley	54	M	GRO098	3.00	2003-12-15	NEW
8782934551	Beyonce	54	M	GRO098	3.00	2003-12-15	NEW
8958111637	Pankaj Kumar	16	F	GRO098	2.00	2020-12-31	STH
8989311592	Rishi Choudhary	16	F	GRO098	2.00	2020-12-31	STH
9023111592	Rishima Katyal	16	F	GRO098	2.00	2020-12-31	STH
9345679271	Luna Lovegood	54	M	GRO098	3.00	2003-12-15	NEW
9345689321	Shawn Mendes	54	M	GRO098	3.00	2003-12-15	NEW
9834111673	Natalie Taylor	16	F	GRO098	2.00	2020-12-31	STH
9956721592	Gopu Kumar	16	F	GRO098	2.00	2020-12-31	STH
9958198422	Dumbledore	16	F	GRO098	2.00	2020-12-31	STH
9958777592	Ravi Kumar	16	F	GRO098	2.00	2020-12-31	STH
A. Add purchase detail
B. Display all purchases
C. Add a new item
D. Import purchase history from csv file
E. Graphical Analysis
Q. Quit
Ln: 33 Col: 0


Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
A. Add purchase detail
B. Display all purchases
C. Add a new item
D. Import purchase history from csv file
E. Graphical Analysis
Q. Quit
Enter choice : E
Select a graph :
A. Product wise comparison using bar graph
B. Gender wise product comparison using bar graph
C. Area wise sales using bar graph
D. Area wise sales using pie chart
E. Product wise sales using pie chart
F. Go back to main menu
Enter choice : A
Select a graph :
A. Product wise comparison using bar graph
B. Gender wise product comparison using bar graph
C. Area wise sales using bar graph
D. Area wise sales using pie chart
E. Product wise sales using pie chart
F. Go back to main menu
Enter choice : B
Select a graph :
A. Product wise comparison using bar graph
B. Gender wise product comparison using bar graph
C. Area wise sales using bar graph
D. Area wise sales using pie chart
E. Product wise sales using pie chart
F. Go back to main menu
Enter choice : E
Mail


Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
E. Product wise sales using pie chart
F. Go back to main menu
Enter choice : E
Select a graph :
A. Product wise comparison using bar graph
B. Gender wise product comparison using bar graph
C. Area wise sales using bar graph
D. Area wise sales using pie chart
E. Product wise sales using pie chart
F. Go back to main menu
Enter choice : C
Select a graph :
A. Product wise comparison using bar graph
B. Gender wise product comparison using bar graph
C. Area wise sales using bar graph
D. Area wise sales using pie chart
E. Product wise sales using pie chart
F. Go back to main menu
Enter choice : D
Select a graph :
A. Product wise comparison using bar graph
B. Gender wise product comparison using bar graph
C. Area wise sales using bar graph
D. Area wise sales using pie chart
E. Product wise sales using pie chart
F. Go back to main menu
Enter choice : F
A. Add purchase detail
B. Display all purchases
C. Add a new item
D. Import purchase history from csv file

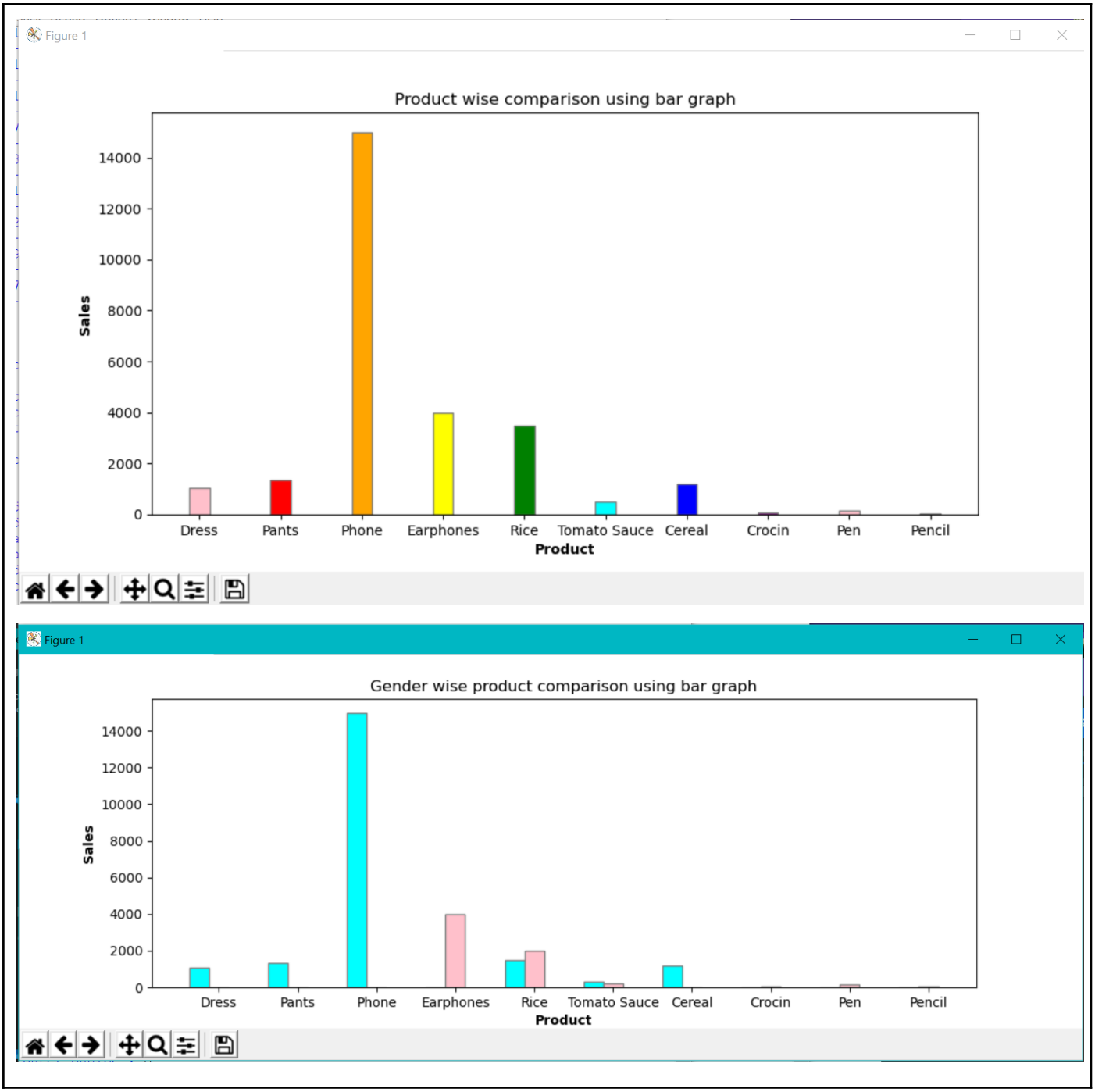


Figure 1
Product wise comparison using bar graph
Sales
14000
12000
10000
8000
6000
4000
2000
0
Dress
Pants
Phone
Earphones
Rice
Tomato Sauce
Cereal
Crocin
Pen
Pencil
Product
Figure 1
Gender wise product comparison using bar graph
Sales
14000
12000
10000
8000
6000
4000
2000
0
Dress
Pants
Phone
Earphones
Rice
Tomato Sauce
Cereal
Crocin
Pen
Pencil
Product

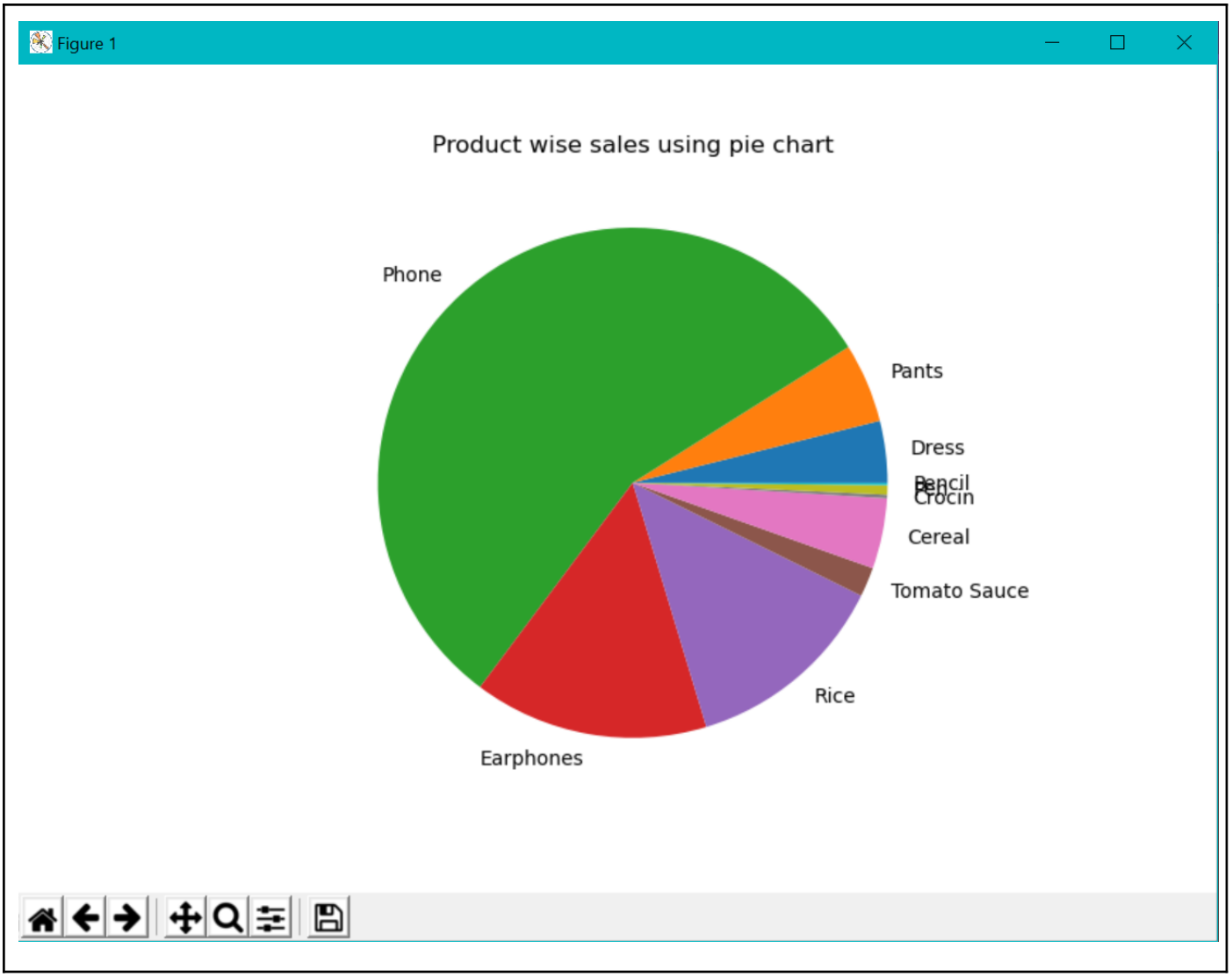


Figure 1
Product wise sales using pie chart
Phone
Pants
Dress
Pencil
Crocin
Cereal
Tomato Sauce
Rice
Earphones

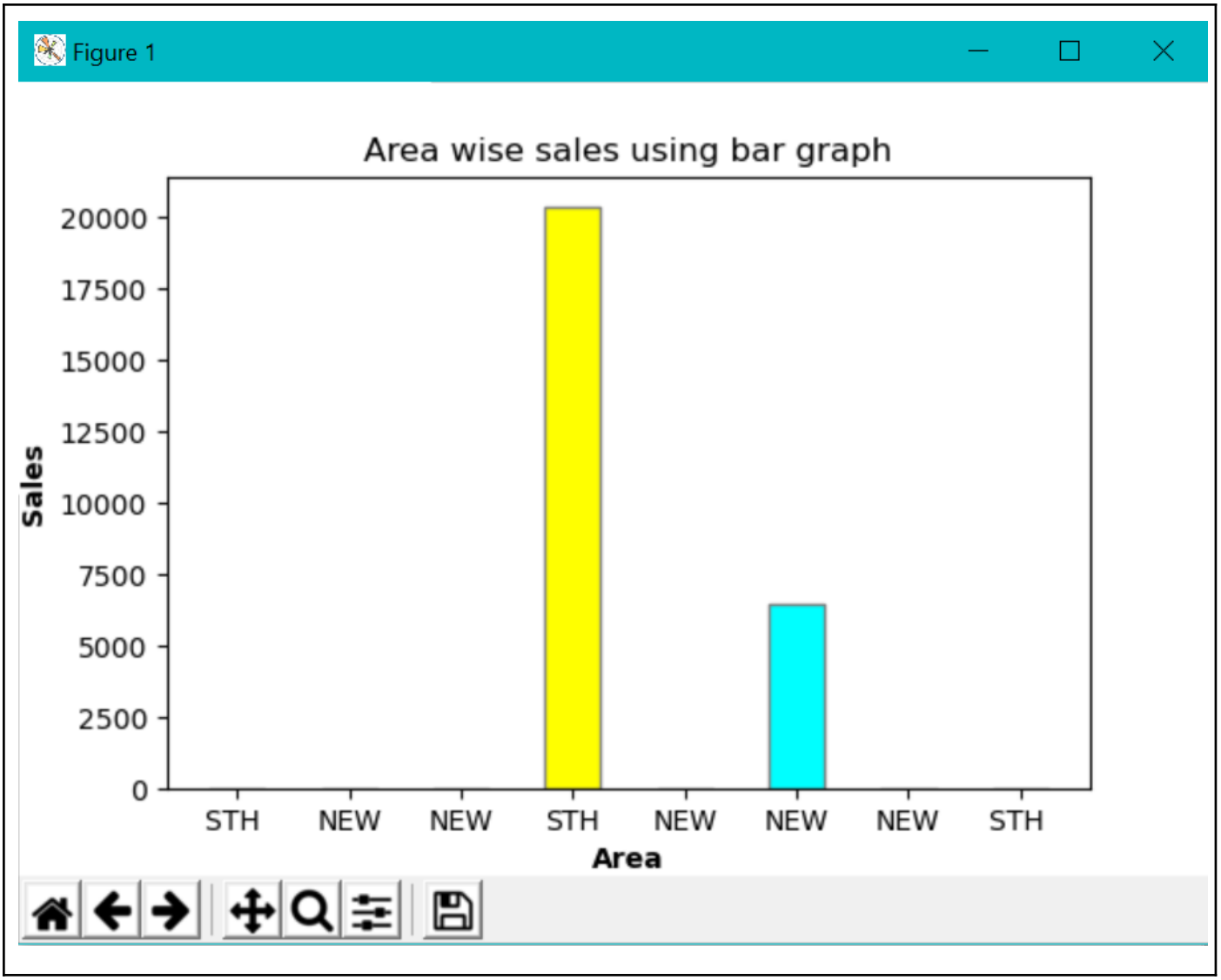


Figure 1
Area wise sales using bar graph
20000
17500
15000
12500
10000
7500
5000
2500
0
Sales
STH
NEW
NEW
STH
NEW
NEW
NEW
STH
Area

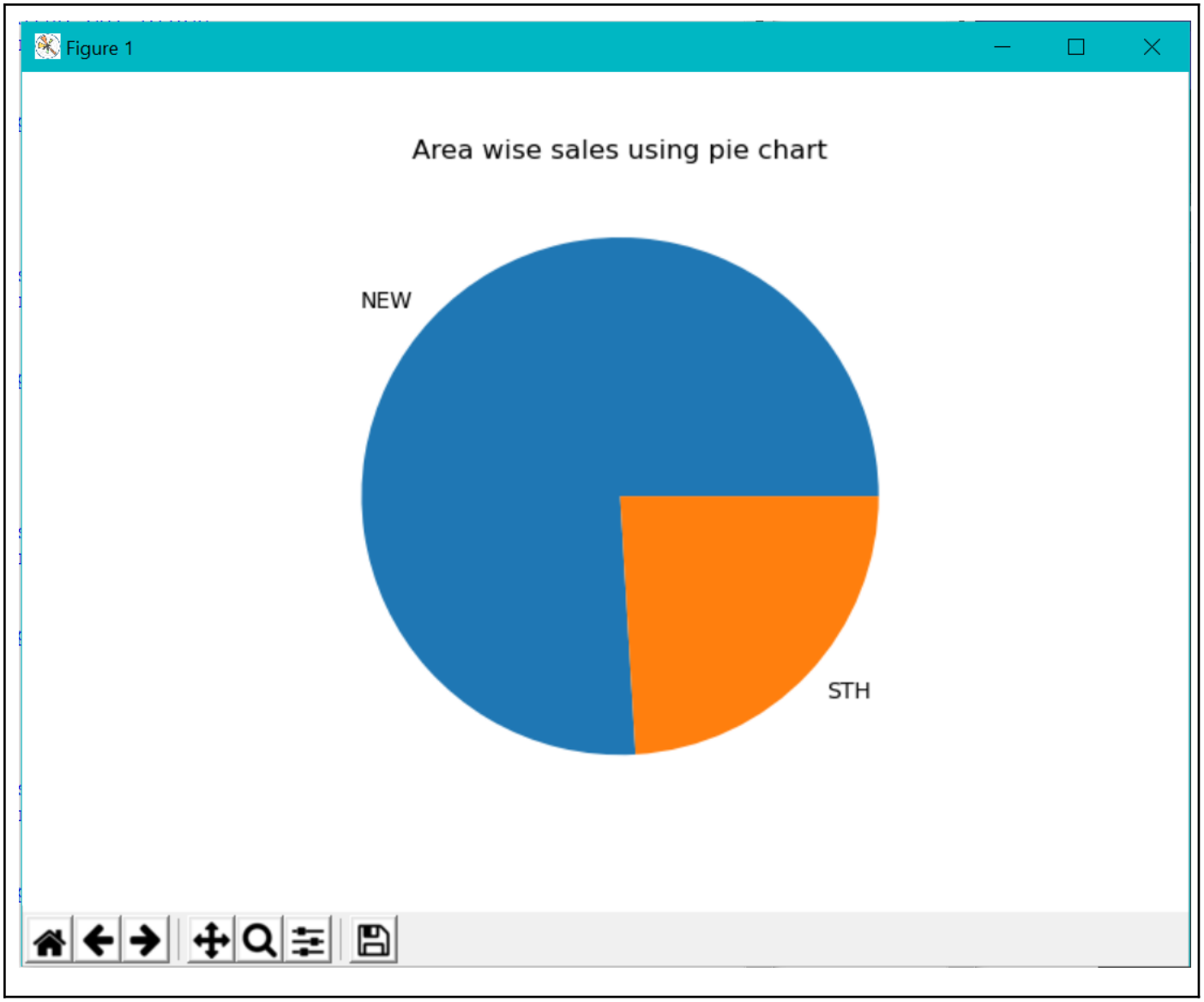
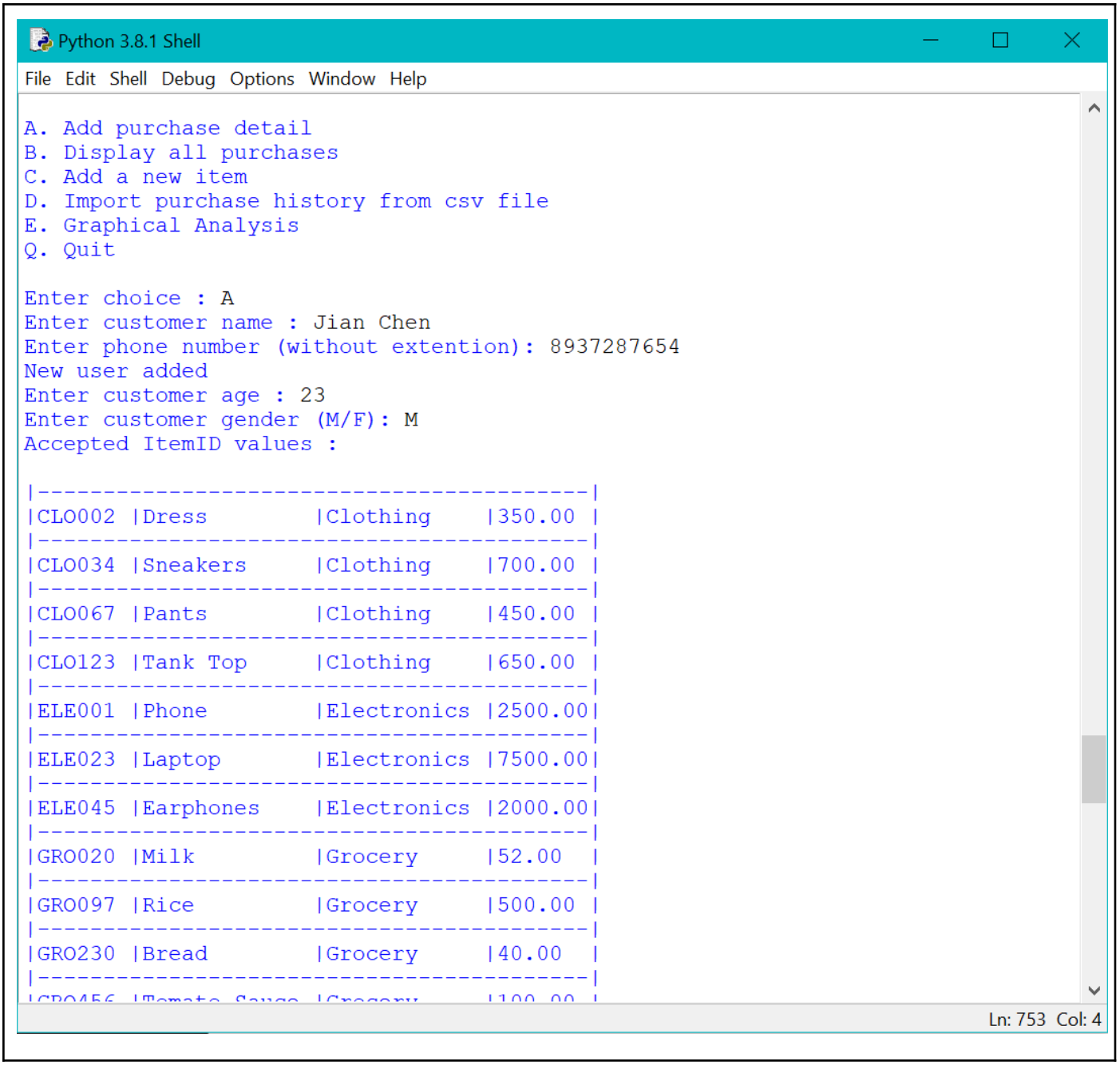
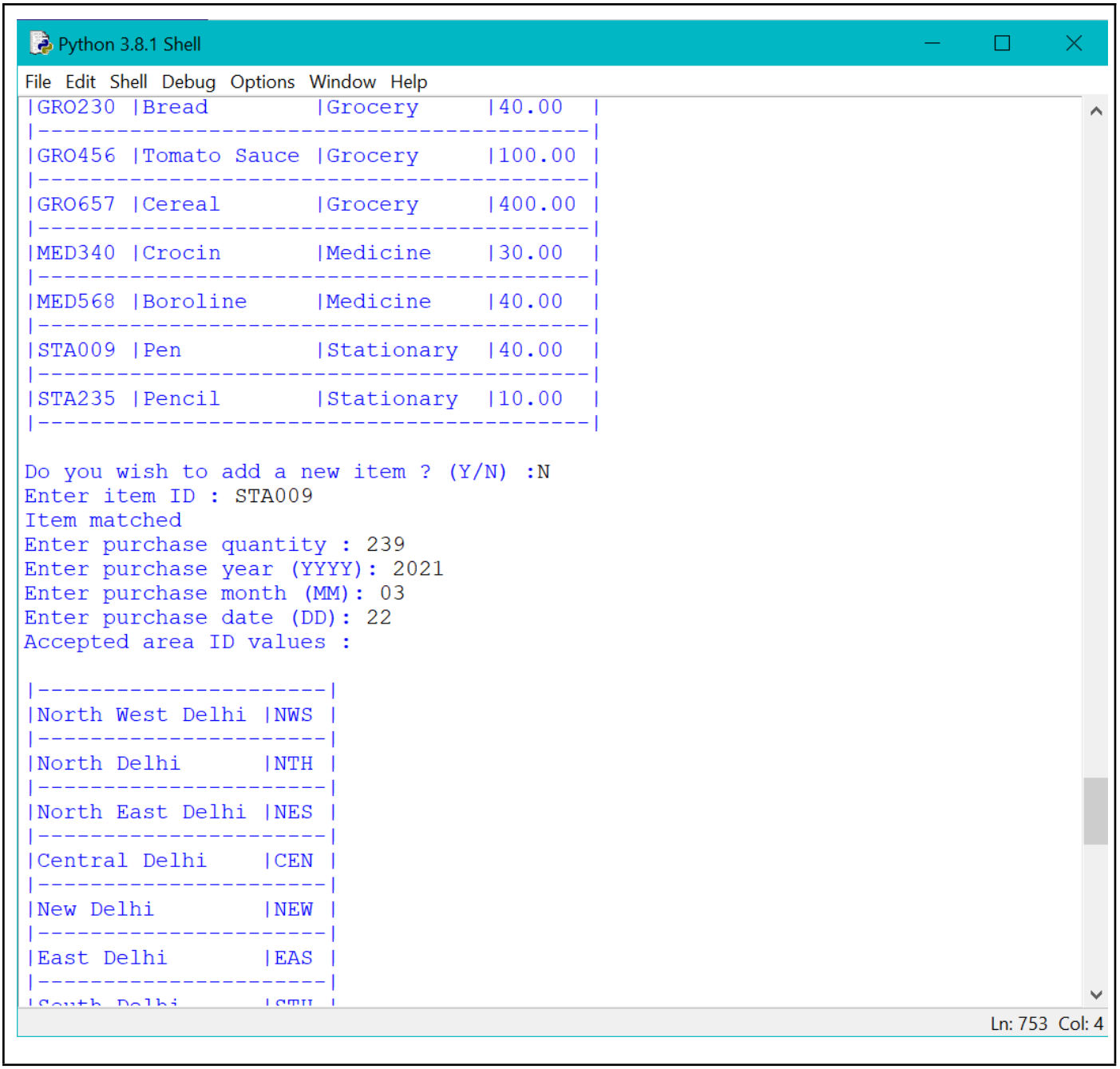
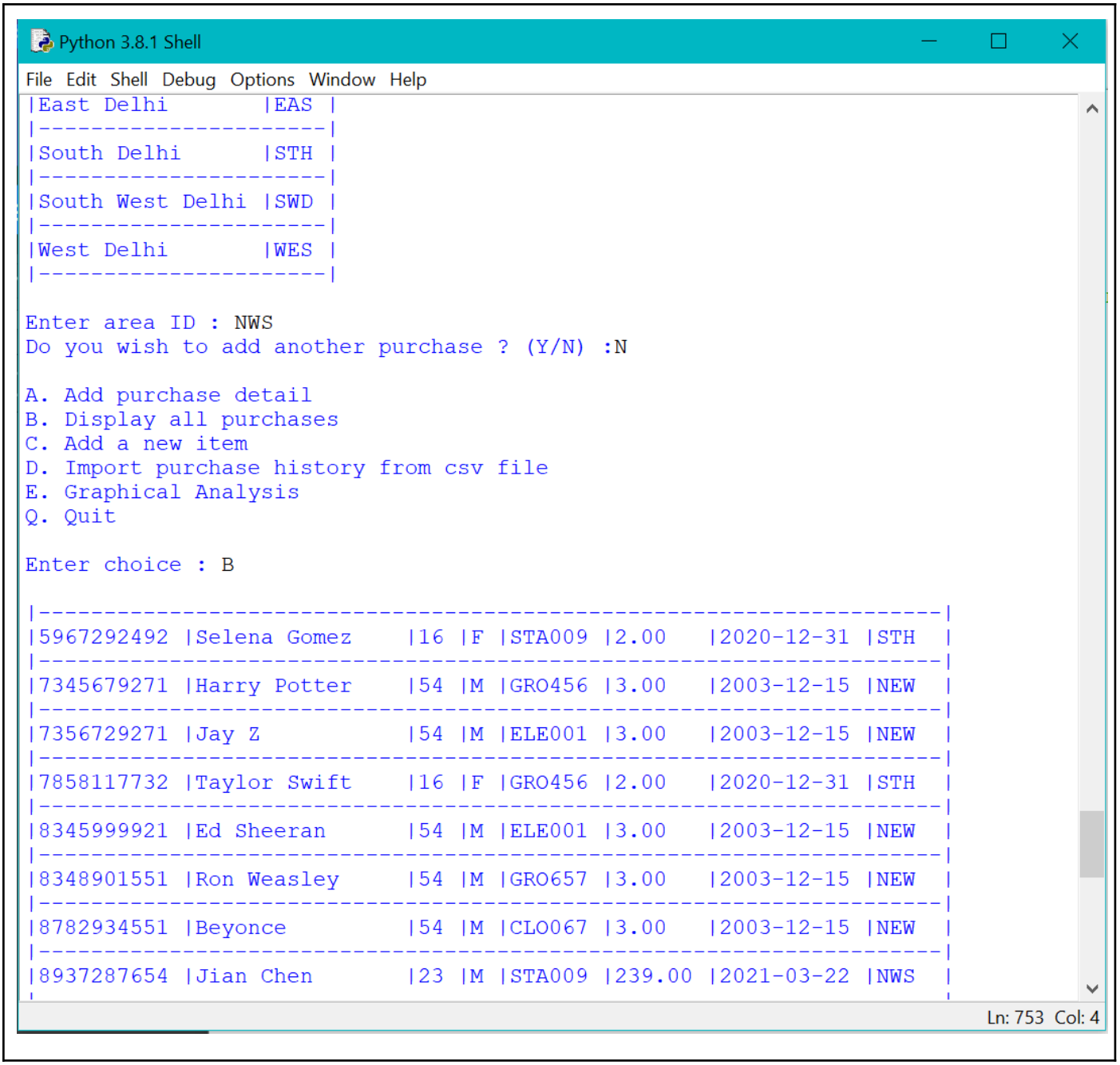
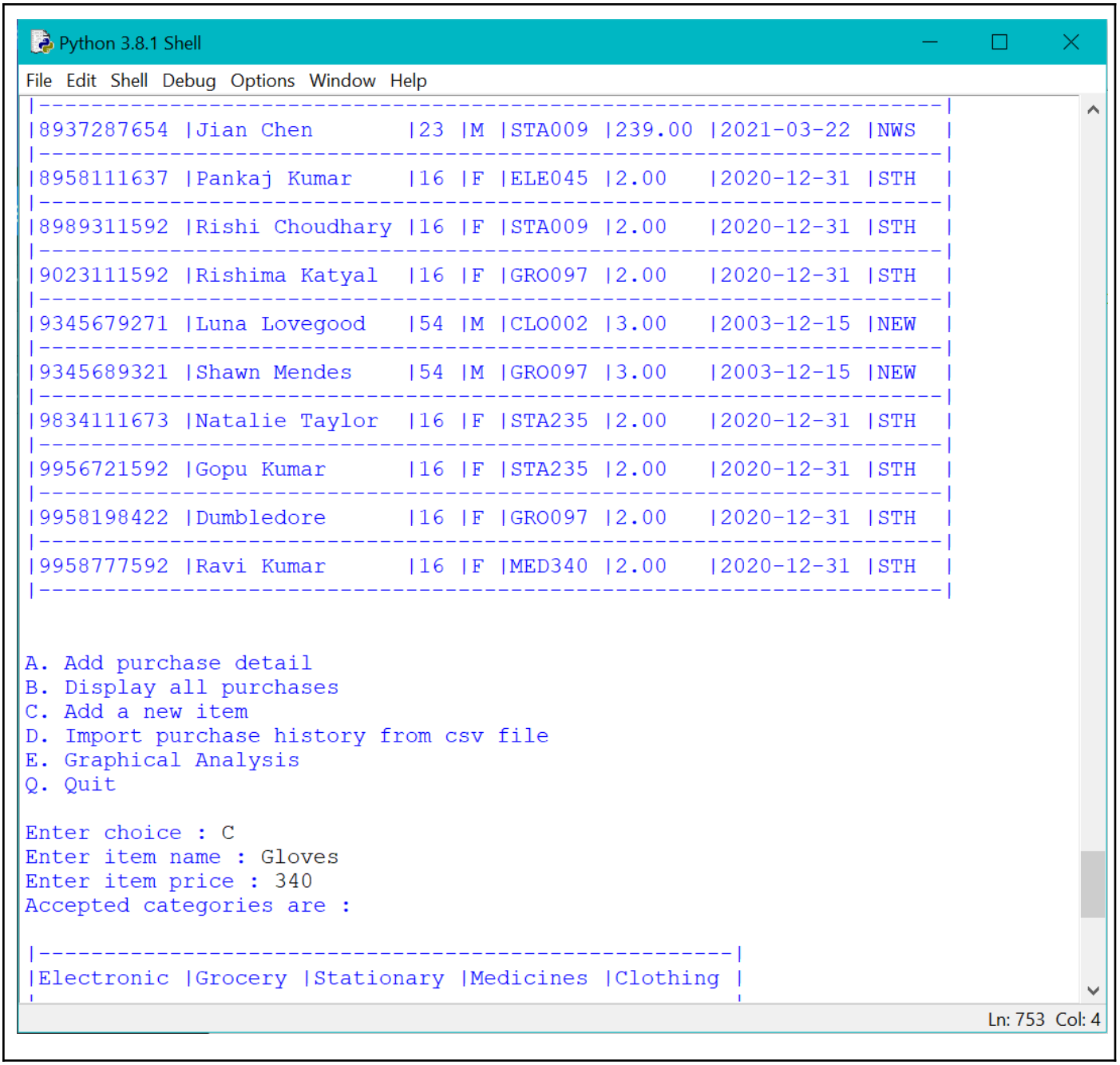


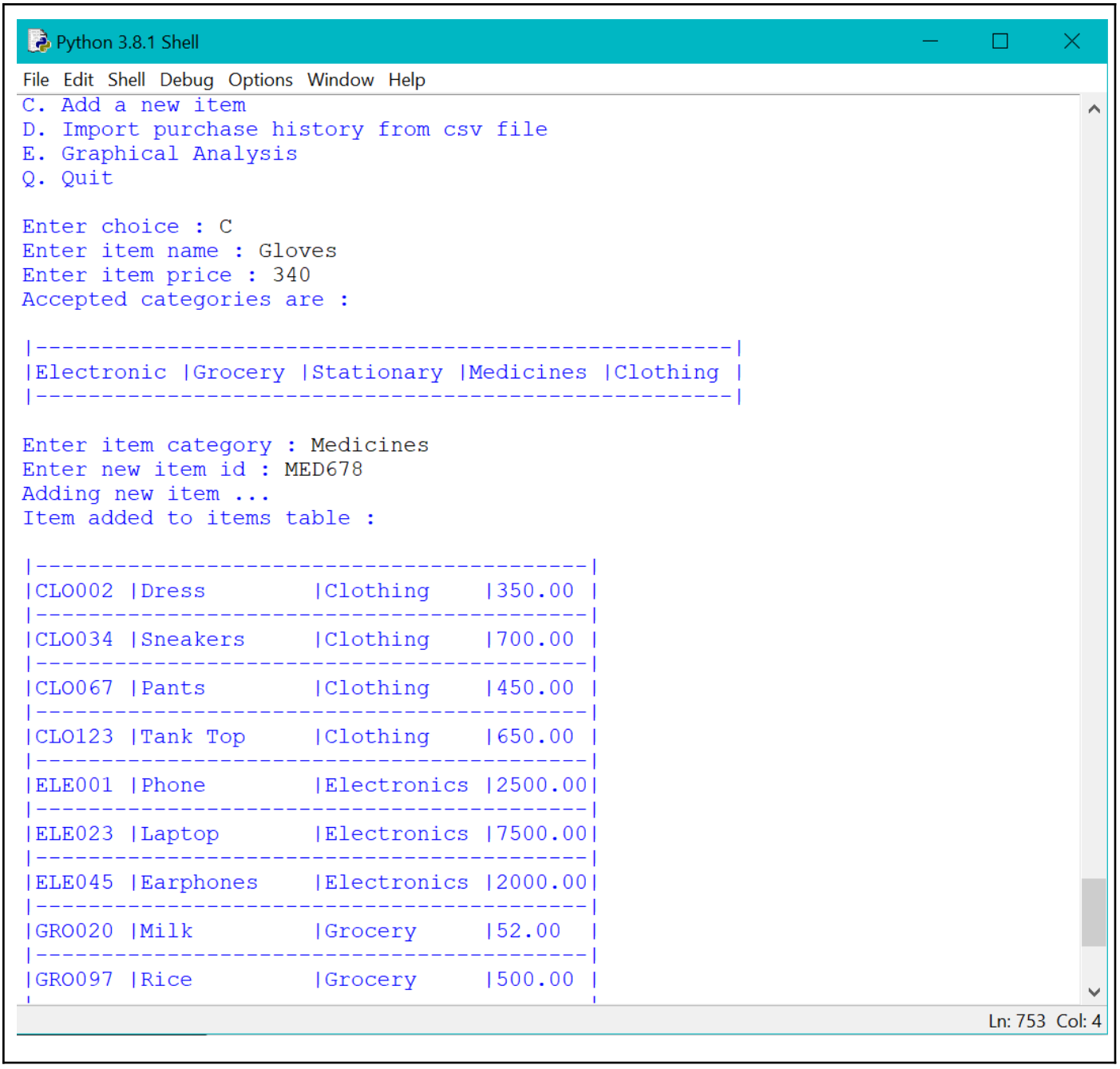
Figure 1
Area wise sales using pie chart
NEW
STH

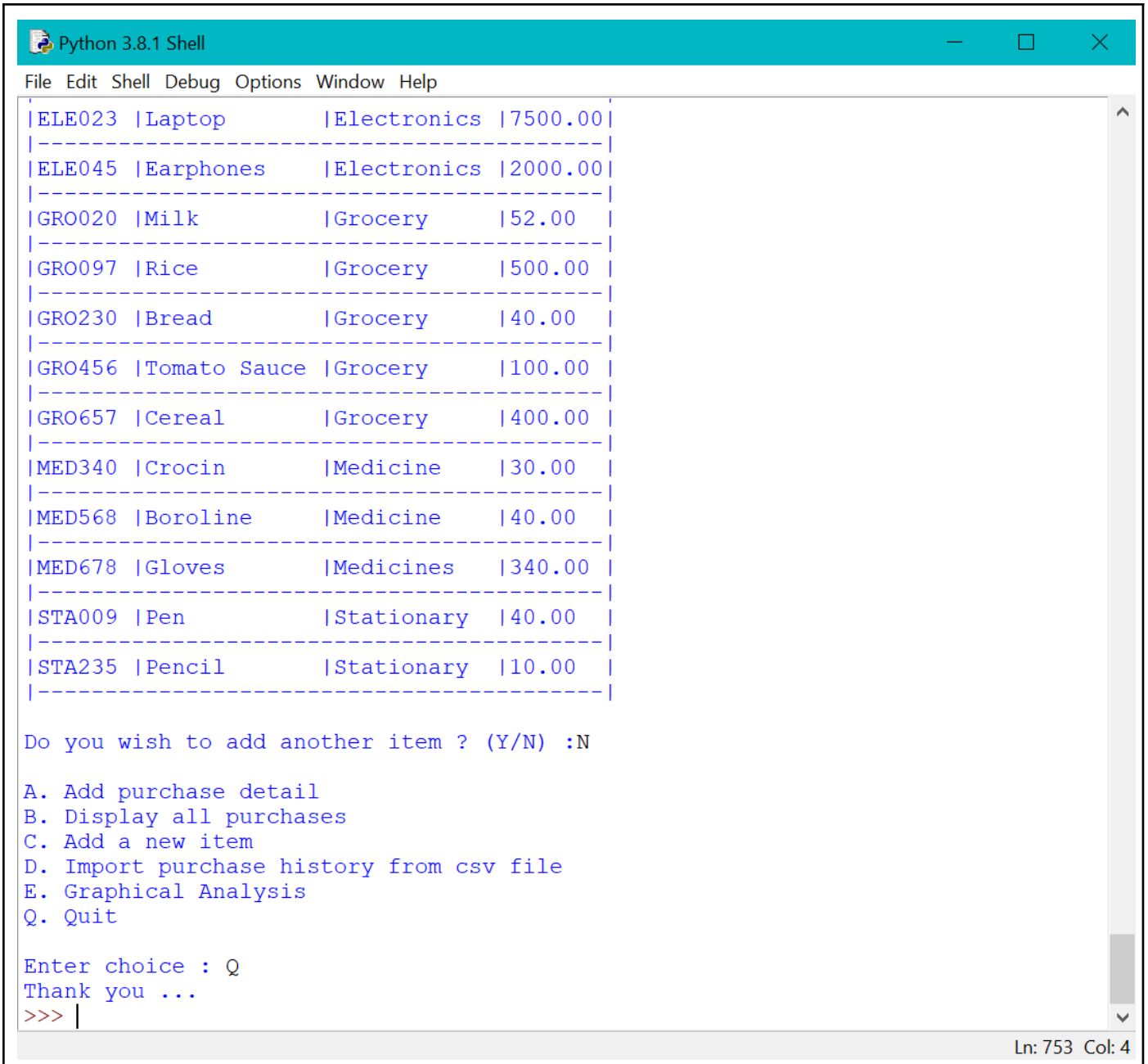

Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
A. Add purchase detail
B. Display all purchases
C. Add a new item
D. Import purchase history from csv file
E. Graphical Analysis
Q. Quit
Enter choice : A
Enter customer name : Jian Chen
Enter phone number (without extention): 8937287654
New user added
Enter customer age : 23
Enter customer gender (M/F): M
Accepted ItemID values :
CLO002	Dress	Clothing	350.00
CLO034	Sneakers	Clothing	700.00
CLO067	Pants	Clothing	450.00
CLO123	Tank Top	Clothing	650.00
ELE001	Phone	Electronics	2500.00
ELE023	Laptop	Electronics	7500.00
ELE045	Earphones	Electronics	2000.00
GRO020	Milk	Grocery	52.00
GRO097	Rice	Grocery	500.00
GRO230	Bread	Grocery	40.00
Ln: 753 Col: 4


Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
GRO230	Bread	Grocery	40.00
GRO456	Tomato Sauce	Grocery	100.00
GRO657	Cereal	Grocery	400.00
MED340	Crocin	Medicine	30.00
MED568	Boroline	Medicine	40.00
STA009	Pen	Stationary	40.00
STA235	Pencil	Stationary	10.00
Do you wish to add a new item ? (Y/N) :N
Enter item ID : STA009
Item matched
Enter purchase quantity : 239
Enter purchase year (YYYY): 2021
Enter purchase month (MM): 03
Enter purchase date (DD): 22
Accepted area ID values :
North West Delhi	NWS
North Delhi	NTH
North East Delhi	NES
Central Delhi	CEN
New Delhi	NEW
East Delhi	EAS
Ln: 753 Col: 4


Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
East Delhi	EAS
South Delhi	STH
South West Delhi	SWD
West Delhi	WES
Enter area ID : NWS
Do you wish to add another purchase ? (Y/N) :N
A. Add purchase detail
B. Display all purchases
C. Add a new item
D. Import purchase history from csv file
E. Graphical Analysis
Q. Quit
Enter choice : B
5967292492 |Selena Gomez |16 |F |STA009 |2.00 |2020-12-31 |STH
7345679271 |Harry Potter |54 |M |GRO456 |3.00 |2003-12-15 |NEW
7356729271 |Jay Z |54 |M |ELE001 |3.00 |2003-12-15 |NEW
7858117732 |Taylor Swift |16 |F |GRO456 |2.00 |2020-12-31 |STH
8345999921 |Ed Sheeran |54 |M |ELE001 |3.00 |2003-12-15 |NEW
8348901551 |Ron Weasley |54 |M |GRO657 |3.00 |2003-12-15 |NEW
8782934551 |Beyonce |54 |M |CLO067 |3.00 |2003-12-15 |NEW
8937287654 |Jian Chen |23 |M |STA009 |239.00 |2021-03-22 |NWS
Ln: 753 Col: 4


Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
8937287654	Jian Chen	23	M	STA009	239.00	2021-03-22	NWS
8958111637	Pankaj Kumar	16	F	ELE045	2.00	2020-12-31	STH
8989311592	Rishi Choudhary	16	F	STA009	2.00	2020-12-31	STH
9023111592	Rishima Katyal	16	F	GRO097	2.00	2020-12-31	STH
9345679271	Luna Lovegood	54	M	CLO002	3.00	2003-12-15	NEW
9345689321	Shawn Mendes	54	M	GRO097	3.00	2003-12-15	NEW
9834111673	Natalie Taylor	16	F	STA235	2.00	2020-12-31	STH
9956721592	Gopu Kumar	16	F	STA235	2.00	2020-12-31	STH
9958198422	Dumbledore	16	F	GRO097	2.00	2020-12-31	STH
9958777592	Ravi Kumar	16	F	MED340	2.00	2020-12-31	STH
A. Add purchase detail
B. Display all purchases
C. Add a new item
D. Import purchase history from csv file
E. Graphical Analysis
Q. Quit
Enter choice : C
Enter item name : Gloves
Enter item price : 340
Accepted categories are :
Electronic	Grocery	Stationary	Medicines	Clothing
Ln: 753 Col: 4


Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
C. Add a new item
D. Import purchase history from csv file
E. Graphical Analysis
Q. Quit
Enter choice : C
Enter item name : Gloves
Enter item price : 340
Accepted categories are :
|Electronic |Grocery |Stationary |Medicines |Clothing |
Enter item category : Medicines
Enter new item id : MED678
Adding new item ...
Item added to items table :
CLO002	Dress	Clothing	350.00
CLO034	Sneakers	Clothing	700.00
CLO067	Pants	Clothing	450.00
CLO123	Tank Top	Clothing	650.00
ELE001	Phone	Electronics	2500.00
ELE023	Laptop	Electronics	7500.00
ELE045	Earphones	Electronics	2000.00
GRO020	Milk	Grocery	52.00
GRO097	Rice	Grocery	500.00
Ln: 753 Col: 4


Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
ELE023	Laptop	Electronics	7500.00
ELE045	Earphones	Electronics	2000.00
GRO020	Milk	Grocery	52.00
GRO097	Rice	Grocery	500.00
GRO230	Bread	Grocery	40.00
GRO456	Tomato Sauce	Grocery	100.00
GRO657	Cereal	Grocery	400.00
MED340	Crocin	Medicine	30.00
MED568	Boroline	Medicine	40.00
MED678	Gloves	Medicines	340.00
STA009	Pen	Stationary	40.00
STA235	Pencil	Stationary	10.00
Do you wish to add another item ? (Y/N) :N
A. Add purchase detail
B. Display all purchases
C. Add a new item
D. Import purchase history from csv file
E. Graphical Analysis
Q. Quit
Enter choice : Q
Thank you ...
>>>
Ln: 753 Col: 4

Hardware Requirement

PC/Laptop/MacBook with
Intel core i7 or any equivalent
With at least 2 GB RAM
10 MB free space on Hard Disk
LCD/LED

Operating System & Compiler

MS Windows/Ubuntu/MacOS

Python IDLE 3.8.1
OR
colab.research.google.com (gmail account)

and
MySQL 8.x