Lab Manual

ADVANCED PYTHON PROGRAMMING

PANDIT DEENDAYAL ENERGY UNIVERSITY
GANDHINAGAR, GUJARAT, INDIA

SCHOOL OF TECHNOLOGY

Computer Science & Engineering

LAB File (2024-25)

Advanced Python Programming for Emerging Application's (23CP301P)

Student Name: ____

Enrollment No.:______________________ Semester:______________________

Division:__________

Group:_________

Instructor: Dr. Rajendra Choudhary

Table of Contents

LAB Assignment 1: Product Review from text file

Consider a scenario where you are working as a data scientist for a large e-commerce company. Your team is responsible for analyzing customer feedback data, which is stored in multiple text files. Each text file contains customer reviews for different product categories. Your task is to write a Python script that performs the following operations:

Read the contents of all the text files in a given directory.

For each review, extract the following information:

- Customer ID (a 6-digit alphanumeric code)
- Product ID (a 10-digit alphanumeric code)
- Review date (in the format "YYYY-MM-DD")
- Review rating (an integer between 1 and 5)
- Review text (the actual feedback provided by the customer)

Calculate the average review rating for each product and store it in a dictionary where the product ID is the key and the average rating is the value.

Determine the top 3 products with the highest average review ratings.

Create a new text file named "summary.txt" and write the following information into it:

- The total number of reviews processed.
- The total number of valid reviews (reviews with all required information extracted successfully).
- The total number of invalid reviews (reviews with missing or incorrect information).
- The product ID and average rating of the top 3 products with the highest average ratings.

Your Python script should be robust, handling any potential errors or exceptions during the file handling process. Additionally, you should implement efficient algorithms to handle large volumes of data without consuming excessive memory or processing time.

Write the Python script to achieve the above objectives and provide detailed comments explaining each step of your implementation.

Logs.txt

```
p1 c1 2023-07-03 5 <good>

p2 c2 2023-07-04 4 <better>

p3 c1 2023-07-07 3 <best>

p1 c3 2023-07-04 2 <worst>

p2 c4 2023-07-07 1 <awesome>
```

logs2.txt

```
p4 c1 2023-07-03 5 <good>
p2 c2 2023-07-04 3 <better>
p4 c1 2023-07-07 3 <best>
p1 c3 2023-07-04 5 <worst>
p1 c4 2023-07-07 4 <awesome>
```

summary.txt

```
p3 3.0
p2 2.6666666666666665
p1 4.0
p4 4.0
Valid: 10
Invalid: 0
Total: 10
```

LAB Assignment 2: Railway Ticket Reservation System

You are tasked with developing a railway ticket reservation system for a busy rail network. The system should handle ticket booking, seat availability, and generate reports for the railway administration. Your task is to implement a Python program that provides the following functionalities:

Load Train Data: The program should read the train data from a CSV file named "trains.csv." Each row in the CSV file represents a train with the following information:

- Train ID (a unique alphanumeric code)
- Train Name
- Source Station
- Destination Station
- Total Seats (total number of seats available on the train)

Load Passenger Data: The program should read the passenger data from a CSV file named "passengers.csv." Each row in the CSV file represents a passenger with the following information:

- Passenger Name
- Train ID (the ID of the train the passenger wants to book a ticket on)
- Number of Tickets (the number of tickets the passenger wants to book)

Check Seat Availability: Given the train ID and the number of tickets requested by a passenger, the program should check if there are enough seats available on the specified train for booking. If seats are available, the booking should be confirmed, and the total fare for the booking should be calculated as per the fare rules (you can define fare rules based on distance, class, etc.).

Update Seat Availability: After confirming the booking, the program should update the seat availability for the corresponding train.

Generate Reports:

Report 1: The program should generate a report showing the details of all the trains, including their names, source stations, destination stations, and the total number of seats available on each train.

Report 2: The program should generate a report showing the total revenue earned from each train based on the total number of confirmed bookings and their respective fares.

Handle Errors: The program should handle various types of errors gracefully, such as invalid train IDs, invalid passenger names, insufficient seats, etc., and provide appropriate error messages.

Note:

You can assume that the passenger data in "passengers.csv" will not exceed the available seats on any train.

You can design the fare rules based on your preference and mention them clearly in the program.

Write the Python program to implement the above functionalities for the railway ticket reservation system. Use comments to explain each step of your implementation and provide sample CSV files ("trains.csv" and "passengers.csv") for testing the program.

Sample data-

Trains.csv

Train ID,Train Name,Source Station,Destination Station,Total Seats, Available Seats, Total fare

T123,Express 123,City A,City B,200,198,500

T456,Superfast 456,City B,City C,150,147,1200

T789,Shatabdi 789,City C,City D,100,96,1400

T321,Intercity 321,City D,City E,180,179,200

T654,Rajdhani 654,City E,City F,250,250,1500

Passengers.csv

Passenger Name,Train ID,Number of Tickets

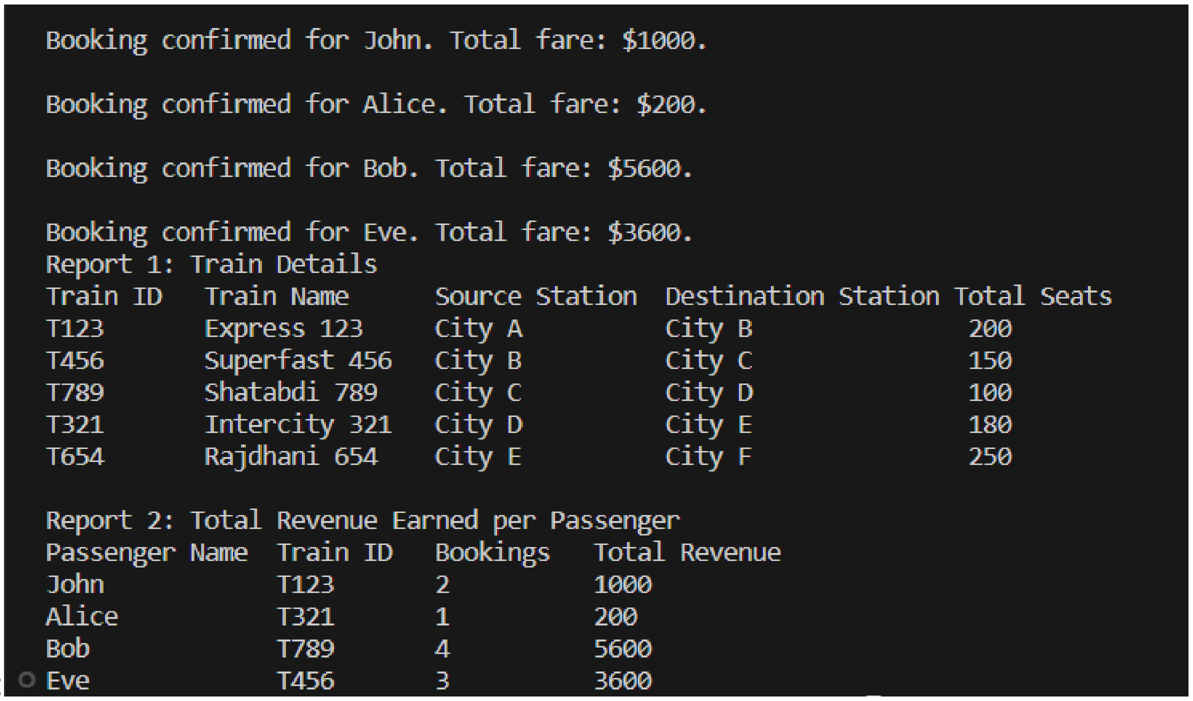
John,T123,2

Alice,T321,1

Bob,T789,4

Eve,T456,3

Output:



```
Booking confirmed for John. Total fare: $1000.

Booking confirmed for Alice. Total fare: $200.

Booking confirmed for Bob. Total fare: $5600.

Booking confirmed for Eve. Total fare: $3600.
Report 1: Train Details
Train ID   Train Name      Source Station  Destination Station Total Seats
T123       Express 123     City A          City B               200
T456       Superfast 456   City B          City C               150
T789       Shatabdi 789    City C          City D               100
T321       Intercity 321   City D          City E               180
T654       Rajdhani 654    City E          City F               250

Report 2: Total Revenue Earned per Passenger
Passenger Name  Train ID   Bookings   Total Revenue
John            T123       2          1000
Alice           T321       1          200
Bob             T789       4          5600
Eve             T456       3          3600
```

Lab Assignment 3 : CSV Manipulation

Your task is to write a Python program that reads this CSV file, calculates the average score for each student, and then creates a new CSV file named "student_average_grades.csv”

- Steps to Solve

1. Read the data from "student_grades.csv" using CSV file handling in Python.
2. For each student, calculate their average score across all subjects (Maths, Science, and English).
3. Create average functions to calculate the average for each student.
4. Store the student's name and their corresponding average score in a new dictionary.
5. Write the data from the dictionary into a new CSV file named "student_average_grades.csv" with two columns: "Name" and "Average."

Lab Assignment 4 :CSV In depth

You are working as a data engineer for a large retail company. Your team is responsible for processing and analyzing sales data from multiple stores across the country. The sales data is stored in CSV files, and each file represents sales data for a specific month and year. Each CSV file has the following columns:

- Date (in the format "YYYY-MM-DD")
- Store ID (a unique alphanumeric code)
- Product ID (a unique alphanumeric code)
- Quantity sold (an integer representing the number of products sold on that date)

The "product_names.csv" file has two columns: "Product ID" and "Product Name," and it contains the mapping for all products in the sales data.

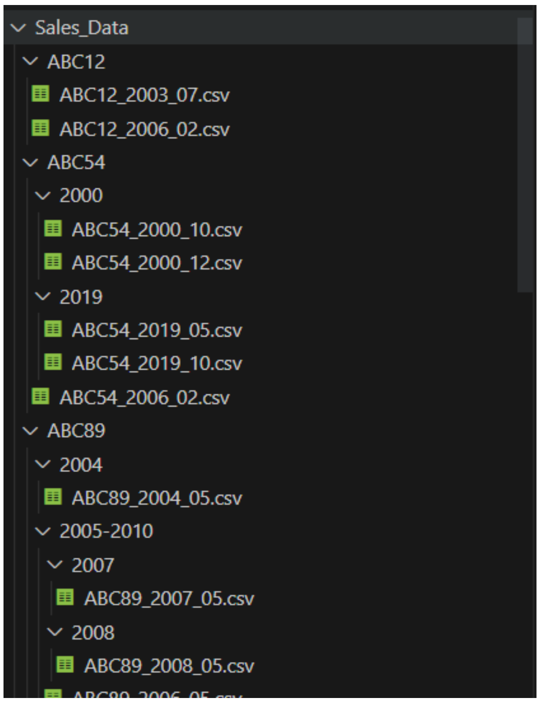
Your task is to write a Python program that performs the following operations:

- Read the sales data from all the CSV files in a given directory and its subdirectories.
- Calculate the total sales (quantity sold) for each product across all stores and all months.
- Determine the top 5 best-selling products in terms of the total quantity sold.

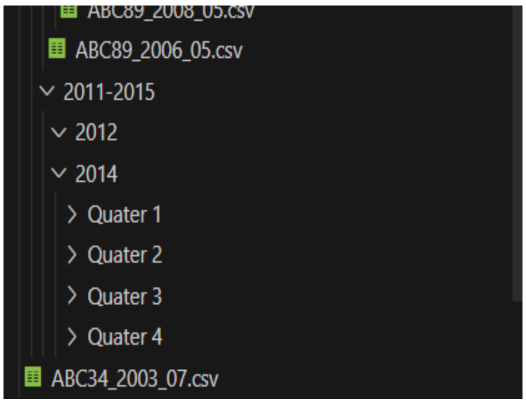
Create a new CSV file named "sales_summary.csv" and write the following information into it:

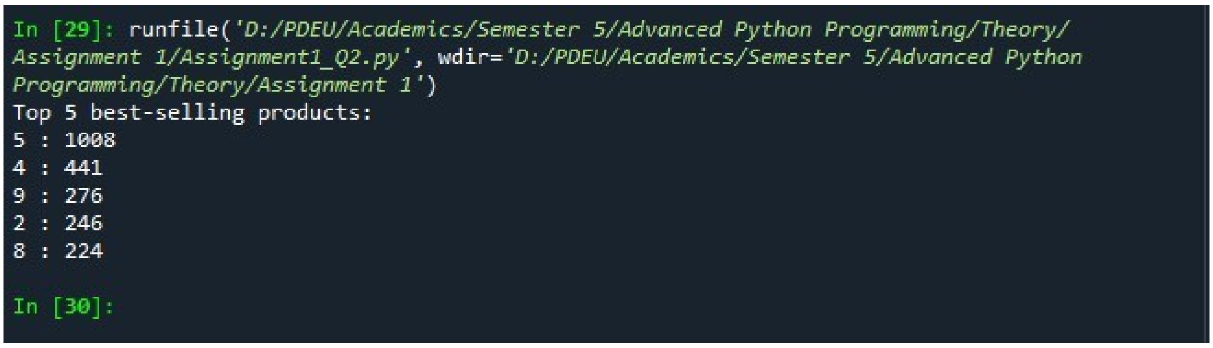
- Product ID
- Product Name
- Total Quantity Sold
- Average Quantity Sold per month (considering all months available in the data)

Output:

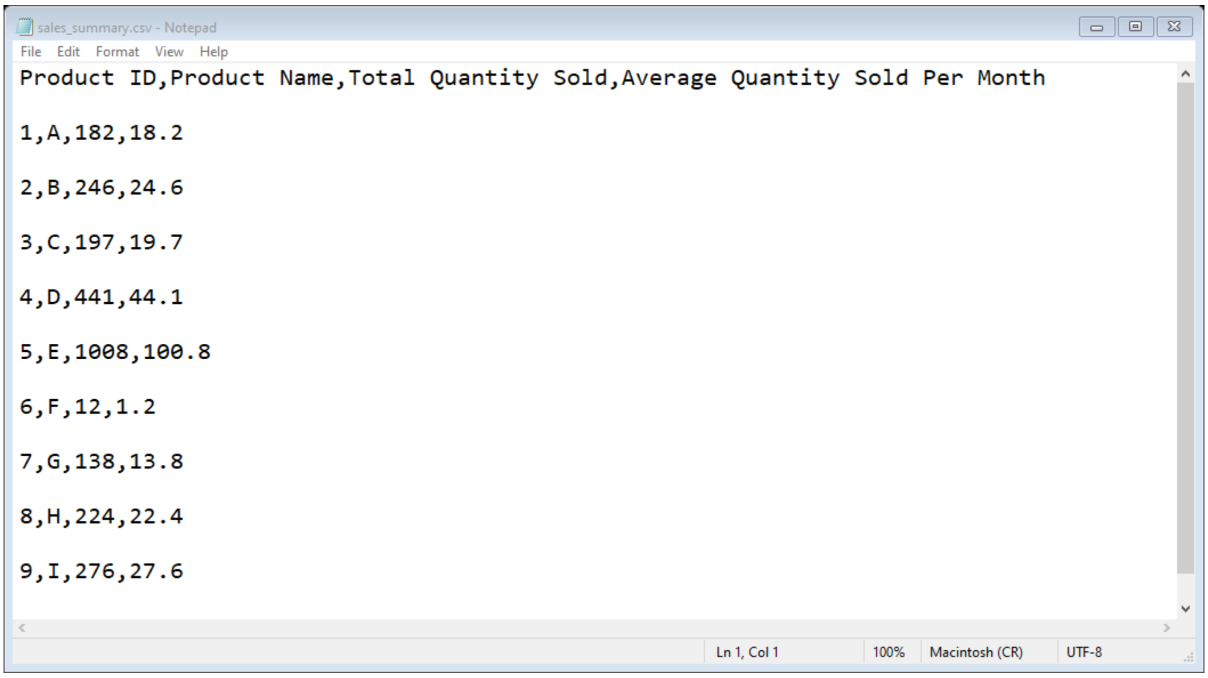


```
Sales_Data
  ABC12
    ABC12_2003_07.csv
    ABC12_2006_02.csv
  ABC54
    2000
      ABC54_2000_10.csv
      ABC54_2000_12.csv
    2019
      ABC54_2019_05.csv
      ABC54_2019_10.csv
    ABC54_2006_02.csv
  ABC89
    2004
      ABC89_2004_05.csv
    2005-2010
      2007
        ABC89_2007_05.csv
      2008
        ABC89_2008_05.csv
```





```
In [29]: runfile('D:/PDEU/Academics/Semester 5/Advanced Python Programming/Theory/
Assignment 1/Assignment1_Q2.py', wdir='D:/PDEU/Academics/Semester 5/Advanced Python
Programming/Theory/Assignment 1')
Top 5 best-selling products:
5 : 1008
4 : 441
9 : 276
2 : 246
8 : 224

In [30]:
```



```
Product ID,Product Name,Total Quantity Sold,Average Quantity Sold Per Month

1,A,182,18.2

2,B,246,24.6

3,C,197,19.7

4,D,441,44.1

5,E,1008,100.8

6,F,12,1.2

7,G,138,13.8

8,H,224,22.4

9,I,276,27.6
```

Lab Assignment 5 : JSON Handling

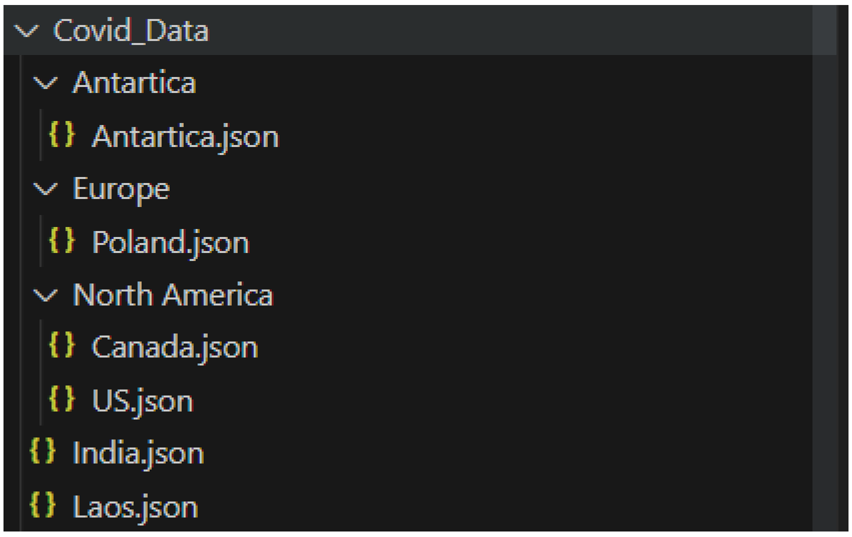
You are working as a data scientist for a healthcare organization, and your team has been tasked with analyzing COVID-19 data from multiple countries. The data is stored in JSON files, with each file representing the daily COVID-19 statistics for a specific country. Each JSON file has the following structure:

```
{  "country":  "Country   Name",

"date": "YYYY-MM-DD",

"confirmed_cases": { "total": 1000, "new": 50 },

"deaths": { "total": 20, "new": 2 },

"recovered": { "total": 800, "new": 30 }

}
```

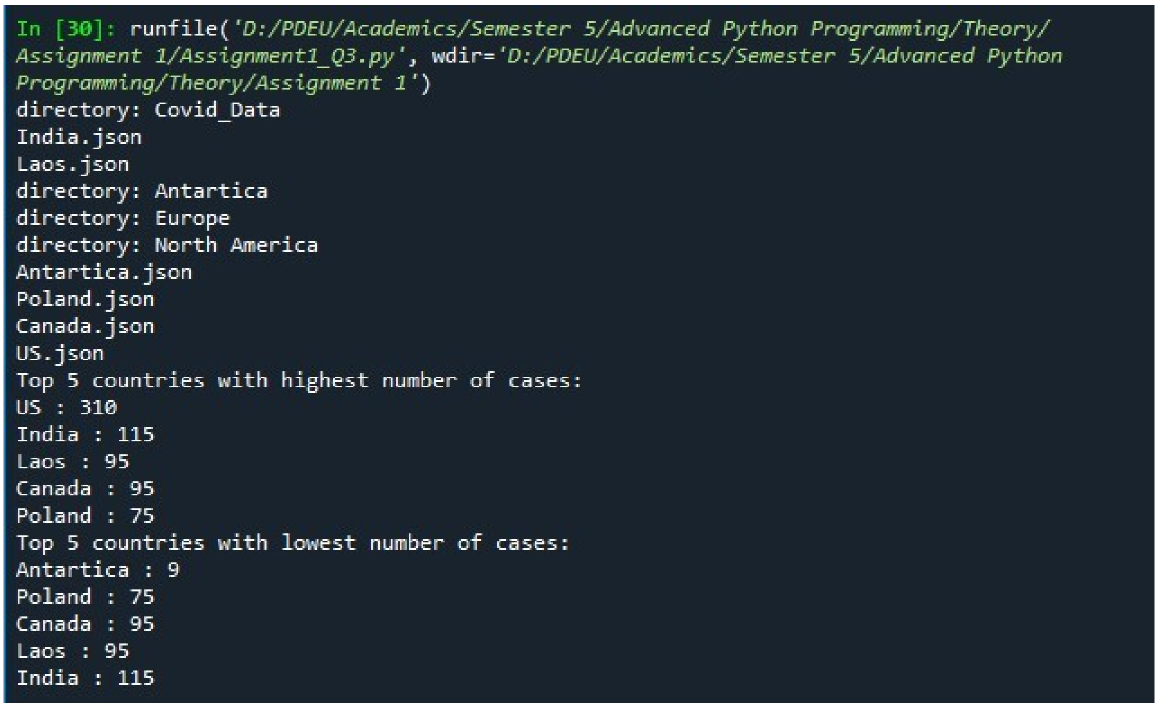
Your task is to write a Python program that performs the following operations:

1. Read COVID-19 data from all JSON files in a given directory and its subdirectories.
2. Calculate and display the following statistics for each country:
    1. Total confirmed cases.
    2. Total deaths.
    3. Total recovered cases.
    4. Total active cases (total confirmed cases minus total deaths and total recovered).
3. Determine the top 5 countries with the highest number of confirmed cases and the lowest number of confirmed cases.
4. Generate a summary report in JSON format that includes the statistics for all countries and save it to a file named "covid19_summary.json".

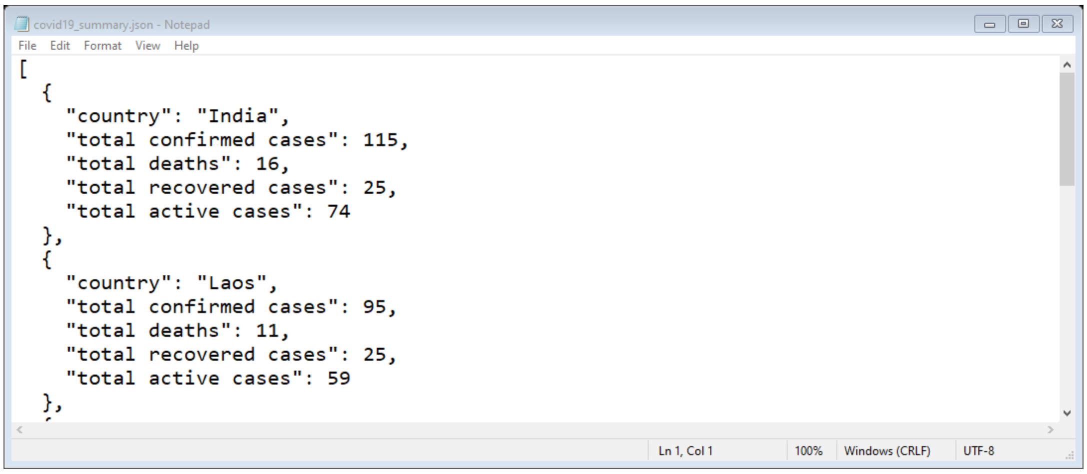
Output:



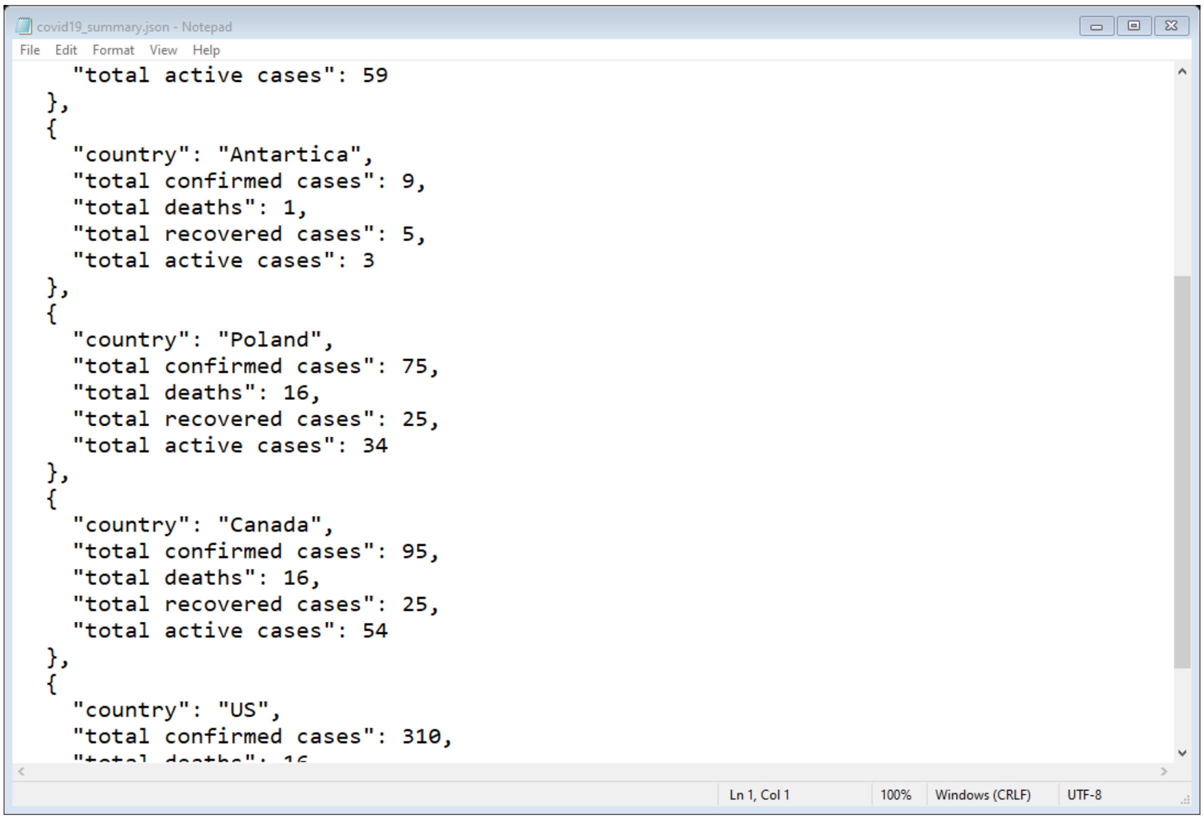
```
Covid_Data
  Antartica
    Antartica.json
  Europe
    Poland.json
  North America
    Canada.json
    US.json
  India.json
  Laos.json
```



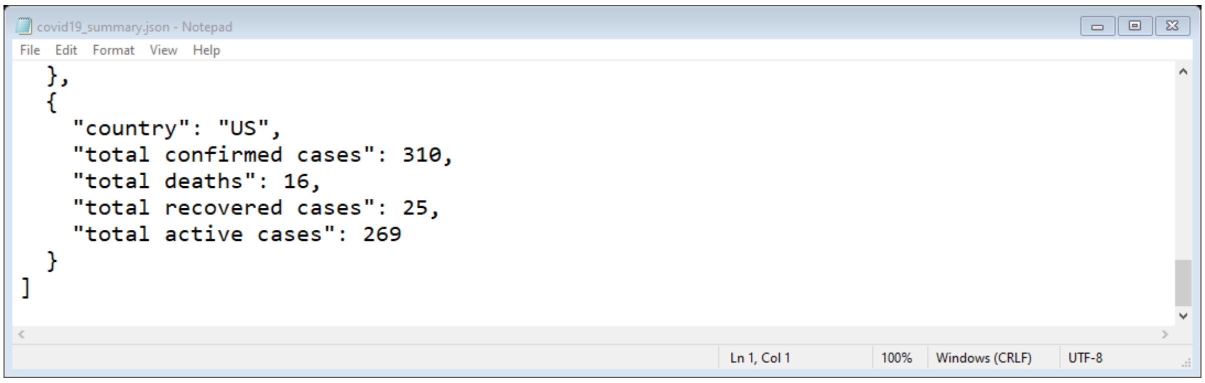
```
In [30]: runfile('D:/PDEU/Academics/Semester 5/Advanced Python Programming/Theory/
Assignment 1/Assignment1_Q3.py', wdir='D:/PDEU/Academics/Semester 5/Advanced Python
Programming/Theory/Assignment 1')
directory: Covid_Data
India.json
Laos.json
directory: Antartica
directory: Europe
directory: North America
Antartica.json
Poland.json
Canada.json
US.json
Top 5 countries with highest number of cases:
US : 310
India : 115
Laos : 95
Canada : 95
Poland : 75
Top 5 countries with lowest number of cases:
Antartica : 9
Poland : 75
Canada : 95
Laos : 95
India : 115
```



```
[
  {
    "country": "India",
    "total confirmed cases": 115,
    "total deaths": 16,
    "total recovered cases": 25,
    "total active cases": 74
  },
  {
    "country": "Laos",
    "total confirmed cases": 95,
    "total deaths": 11,
    "total recovered cases": 25,
    "total active cases": 59
  },
```



```
    "total active cases": 59
  },
  {
    "country": "Antartica",
    "total confirmed cases": 9,
    "total deaths": 1,
    "total recovered cases": 5,
    "total active cases": 3
  },
  {
    "country": "Poland",
    "total confirmed cases": 75,
    "total deaths": 16,
    "total recovered cases": 25,
    "total active cases": 34
  },
  {
    "country": "Canada",
    "total confirmed cases": 95,
    "total deaths": 16,
    "total recovered cases": 25,
    "total active cases": 54
  },
  {
    "country": "US",
    "total confirmed cases": 310,
    "total deaths": 16
```



```
  },
  {
    "country": "US",
    "total confirmed cases": 310,
    "total deaths": 16,
    "total recovered cases": 25,
    "total active cases": 269
  }
]
```

Lab Assignment 6: Error and Exception Handling

You are working on a project to build a custom text processing tool that reads input from various sources, processes the text data, and stores the results in an output file. As part of this project, you need to implement a robust exception handling mechanism to handle potential errors that may arise during the text processing.

- The tool needs to perform the following steps:

1. Read the input data from a file specified by the user.
2. Process the text data by performing various operations, such as counting words, calculating character frequencies, and generating word clouds.
3. Store the processed results in an output file.

Your task is to design a Python program that incorporates appropriate exception handling to handle the following situations:

1. File Not Found Error: If the user provides an invalid file path or the input file is not found, your program should raise a custom exception FileNotFoundError with a suitable error message.
2. Invalid Input Data: During text processing, if any unexpected input data is encountered (e.g., non-string values or missing data), your program should raise a custom exception InvalidInputDataError with relevant details.
3. Disk Space Full: If the output file cannot be written due to insufficient disk space, your program should raise a custom exception DiskSpaceFullError

Additionally, the program should have the following features:

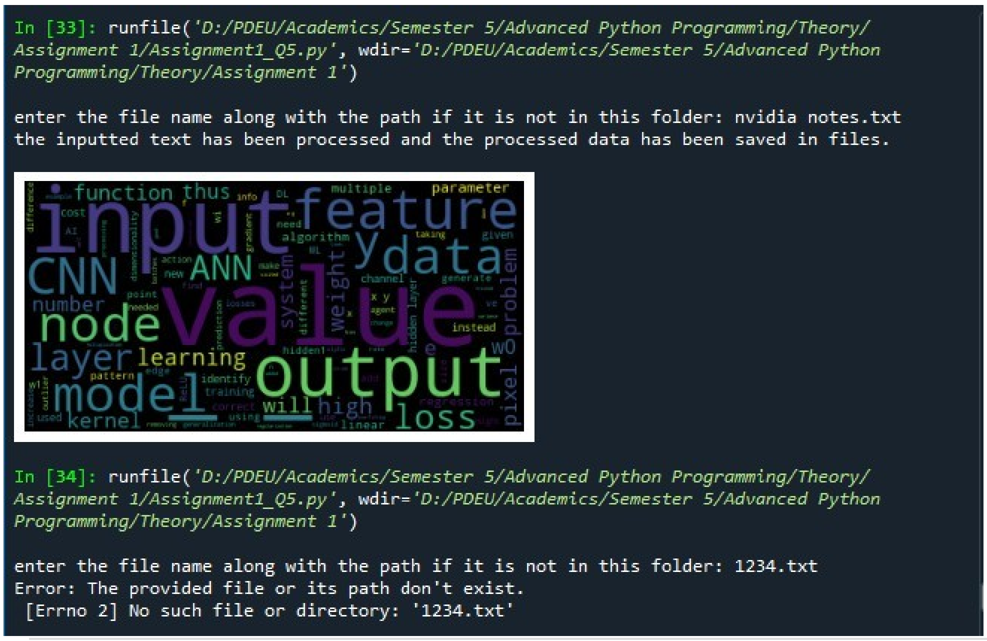
- The custom exception classes should inherit from the base Exception class and provide meaningful error messages.

- Proper logging should be implemented to capture details about the exceptions that occur during text processing.

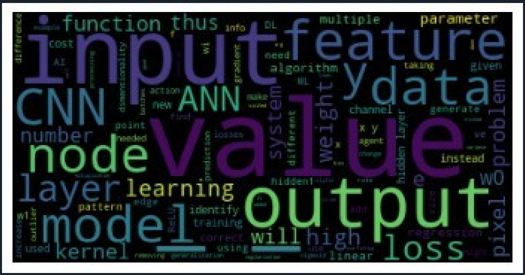
- The program should provide a user-friendly interface, allowing the user to enter the input file path and choose the desired text processing operations.

- The processed results should be stored in an output file with a suitable format (e.g., JSON, CSV, or plain text).

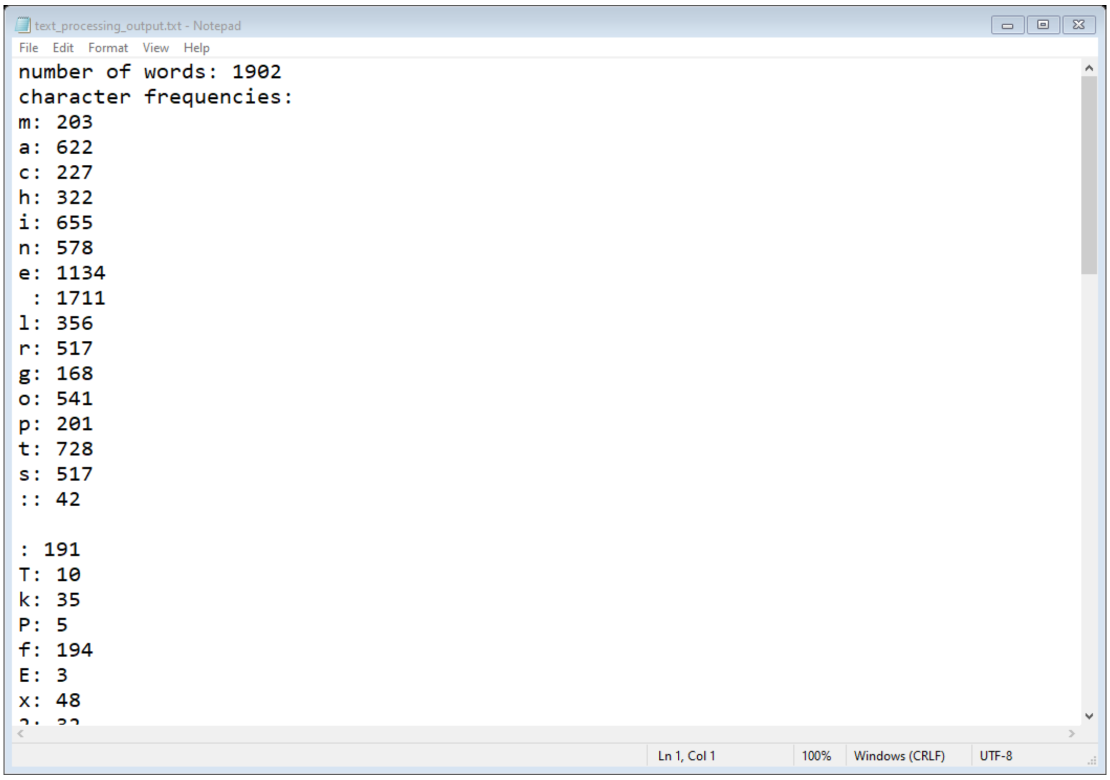
Output:



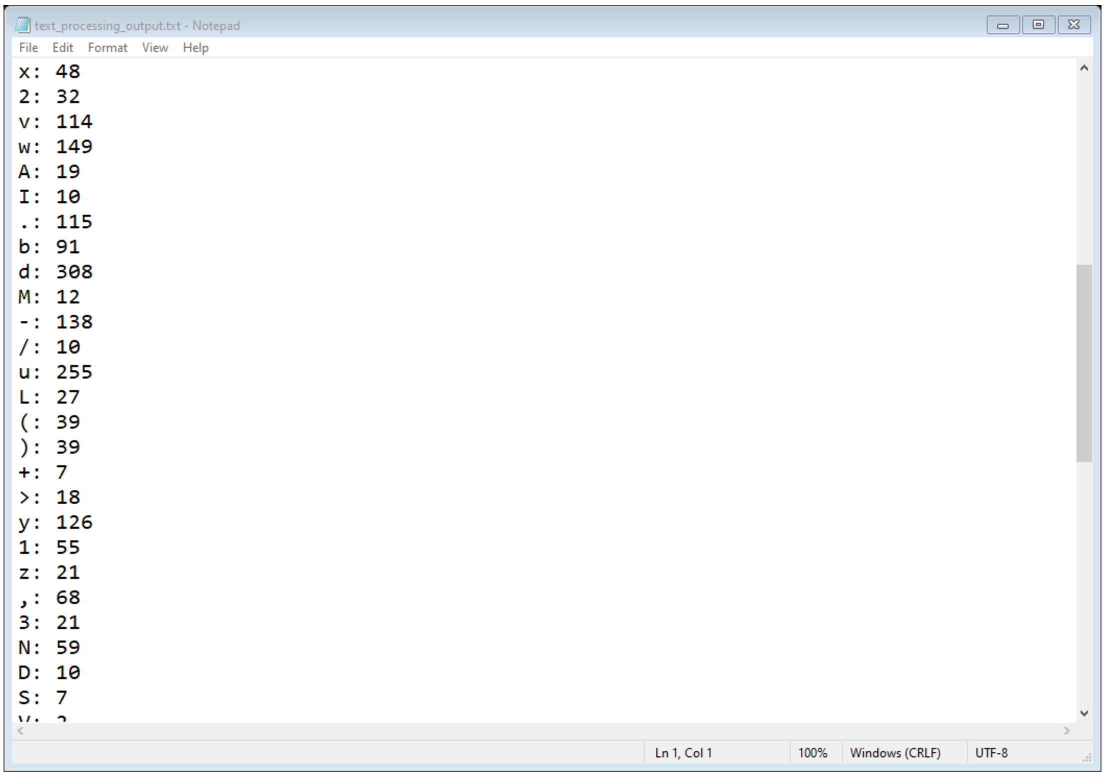
```
In [33]: runfile('D:/PDEU/Academics/Semester 5/Advanced Python Programming/Theory/
Assignment 1/Assignment1_Q5.py', wdir='D:/PDEU/Academics/Semester 5/Advanced Python
Programming/Theory/Assignment 1')

enter the file name along with the path if it is not in this folder: nvidia notes.txt
the inputted text has been processed and the processed data has been saved in files.
```



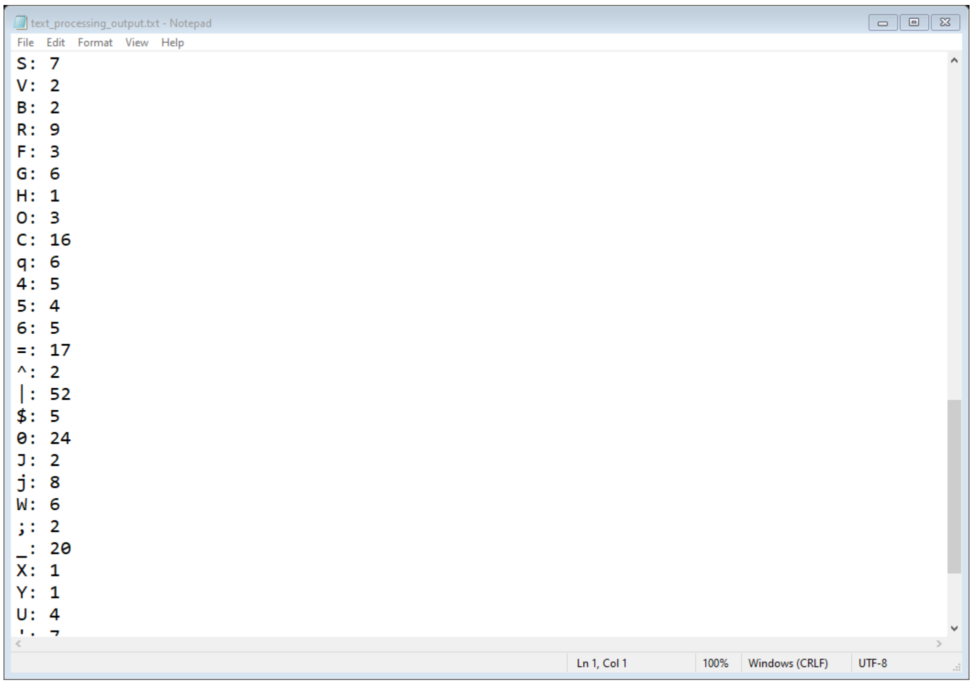
```
In [34]: runfile('D:/PDEU/Academics/Semester 5/Advanced Python Programming/Theory/
Assignment 1/Assignment1_Q5.py', wdir='D:/PDEU/Academics/Semester 5/Advanced Python
Programming/Theory/Assignment 1')

enter the file name along with the path if it is not in this folder: 1234.txt
Error: The provided file or its path don't exist.
 [Errno 2] No such file or directory: '1234.txt'
```


text_processing_output.txt - Notepad
File Edit Format View Help
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8

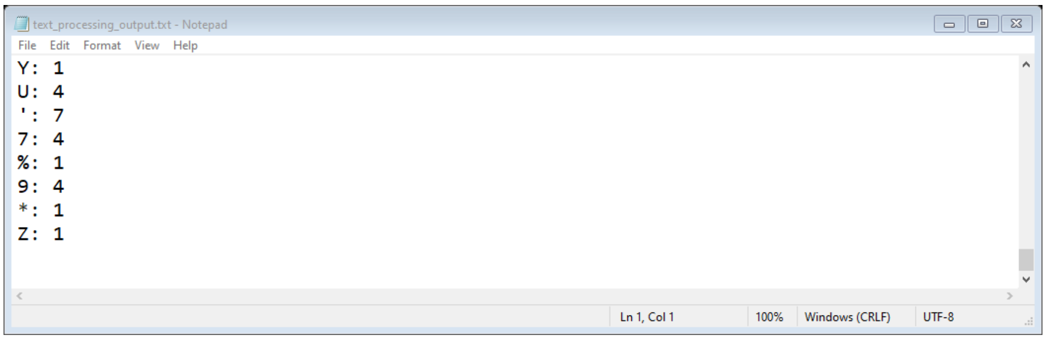
```
number of words: 1902
character frequencies:
m: 203
a: 622
c: 227
h: 322
i: 655
n: 578
e: 1134
 : 1711
l: 356
r: 517
g: 168
o: 541
p: 201
t: 728
s: 517
:: 42

: 191
T: 10
k: 35
P: 5
f: 194
E: 3
x: 48
```


text_processing_output.txt - Notepad
File Edit Format View Help
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8

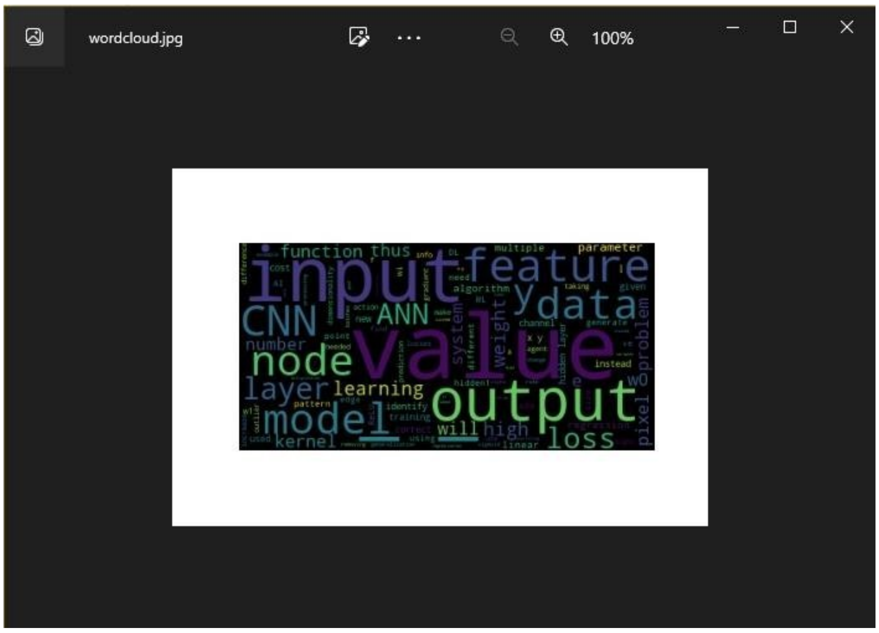
```
x: 48
2: 32
v: 114
w: 149
A: 19
I: 10
.: 115
b: 91
d: 308
M: 12
-: 138
/: 10
u: 255
L: 27
(: 39
): 39
+: 7
>: 18
y: 126
1: 55
z: 21
,: 68
3: 21
N: 59
D: 10
S: 7
```


text_processing_output.txt - Notepad
File Edit Format View Help
Ln 1, Col 1 100% Windows (CRLF) UTF-8

```
S: 7
V: 2
B: 2
R: 9
F: 3
G: 6
H: 1
O: 3
C: 16
q: 6
4: 5
5: 4
6: 5
=: 17
^: 2
|: 52
$: 5
0: 24
J: 2
j: 8
W: 6
;: 2
_: 20
X: 1
Y: 1
U: 4
```


text_processing_output.txt - Notepad
File Edit Format View Help
Ln 1, Col 1 100% Windows (CRLF) UTF-8

```
Y: 1
U: 4
': 7
7: 4
%: 1
9: 4
*: 1
Z: 1
```



Lab Assignment 7 : Pdf reading creation and editing

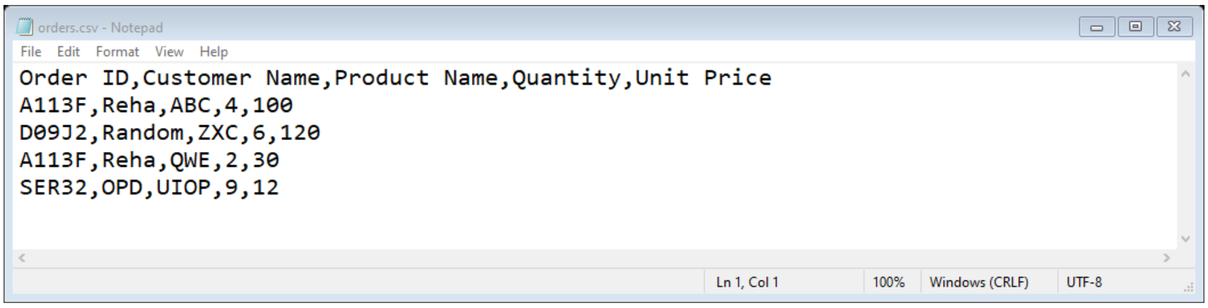
You are working for a company that sells products online. Your task is to develop a Python program that reads order data from a CSV file, generates individual PDF invoices for each order, and then merges all the PDF invoices into a single PDF file.

1. Load Order Data: The program should read order data from a CSV file named "orders.csv." Each row in the CSV file represents an order with the following information:
    1. Order ID (a unique alphanumeric code)
    2. Customer Name
    3. Product Name
    4. Quantity
    5. Unit Price
2. Calculate Total Amount: For each order, calculate the total amount by multiplying the quantity with the unit price.

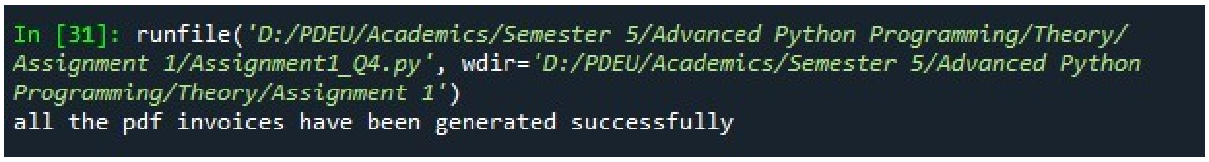
Generate PDF Invoices: Create individual PDF invoices for each order. Each invoice should contain the following details:

1. Invoice Number (same as the Order ID)
2. Date of Purchase (current date)
3. Customer Name
4. Product Name
5. Quantity
6. Unit Price
7. Total Amount

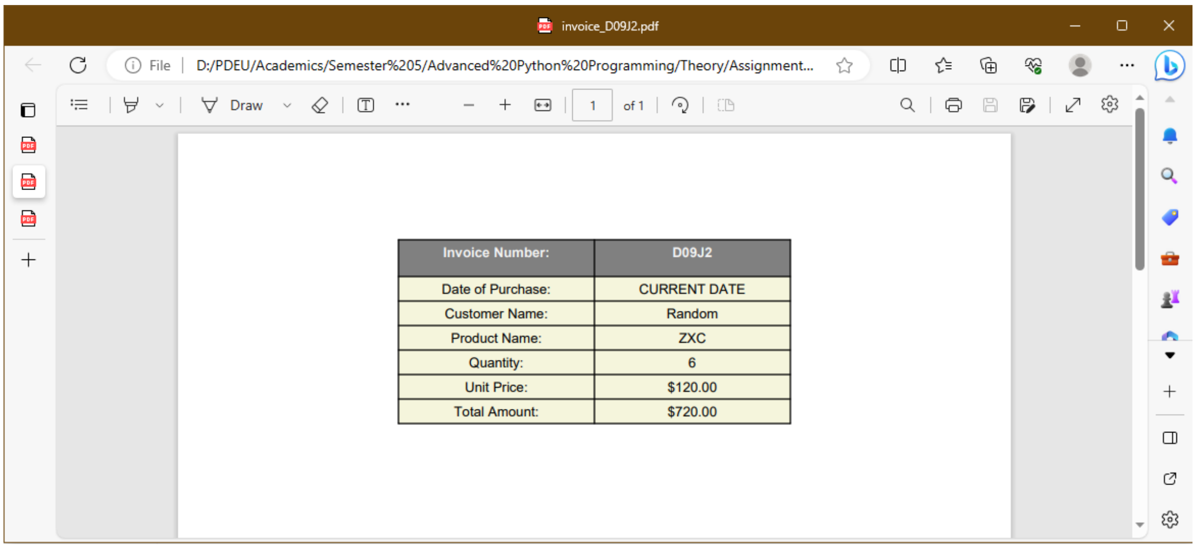
Output:



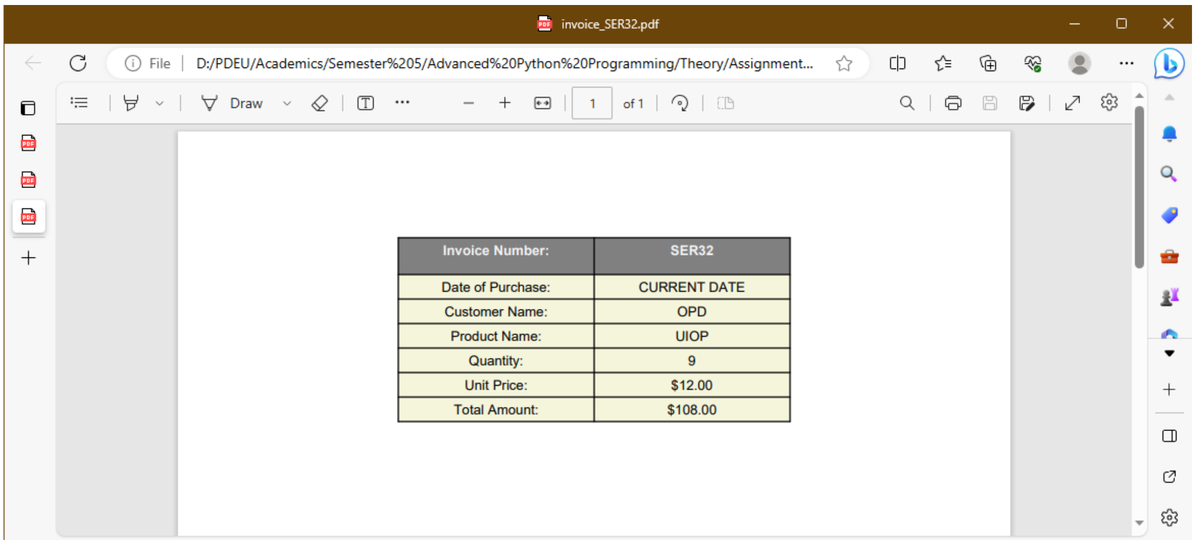
```
Order ID,Customer Name,Product Name,Quantity,Unit Price
A113F,Reha,ABC,4,100
D09J2,Random,ZXC,6,120
A113F,Reha,QWE,2,30
SER32,OPD,UIOP,9,12
```



```
In [31]: runfile('D:/PDEU/Academics/Semester 5/Advanced Python Programming/Theory/
Assignment 1/Assignment1_Q4.py', wdir='D:/PDEU/Academics/Semester 5/Advanced Python
Programming/Theory/Assignment 1')
all the pdf invoices have been generated successfully
```


invoice_D09J2.pdf
File | D:/PDEU/Academics/Semester%205/Advanced%20Python%20Programming/Theory/Assignment...
Draw
1 of 1

Invoice Number:	D09J2
Date of Purchase:	CURRENT DATE
Customer Name:	Random
Product Name:	ZXC
Quantity:	6
Unit Price:	$120.00
Total Amount:	$720.00


invoice_SER32.pdf
File | D:/PDEU/Academics/Semester%205/Advanced%20Python%20Programming/Theory/Assignment...
Draw
1 of 1

Invoice Number:	SER32
Date of Purchase:	CURRENT DATE
Customer Name:	OPD
Product Name:	UIOP
Quantity:	9
Unit Price:	$12.00
Total Amount:	$108.00

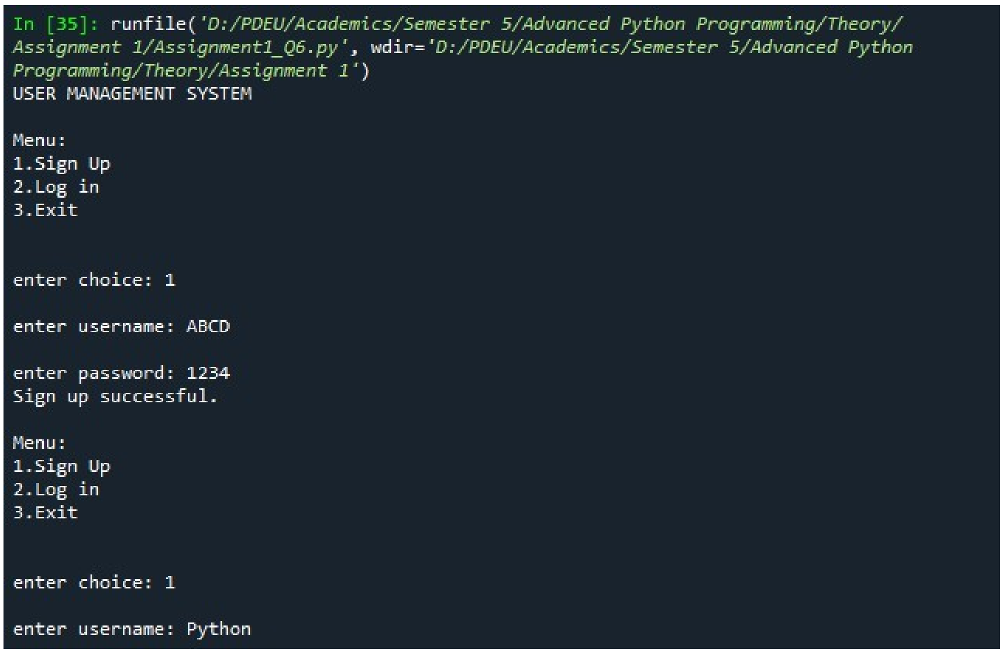
Lab Assignment 8 : User Management Automation

You are developing a command-line task management system for a small team of users.

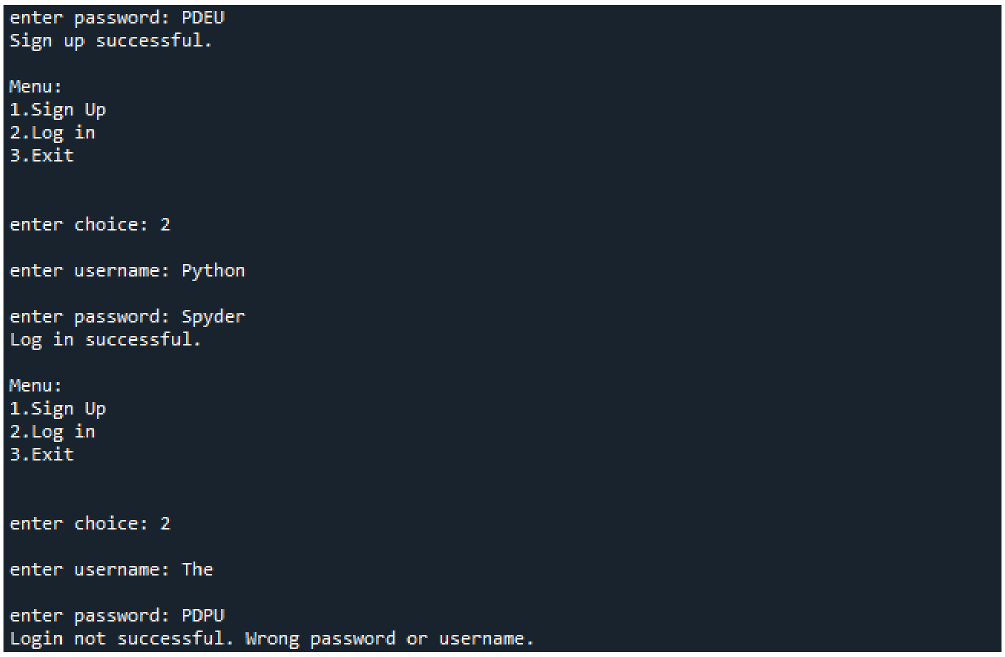
User Management:

- Implement a user registration system where users can sign up and log in. Store user data in a file, including usernames and hashed passwords.

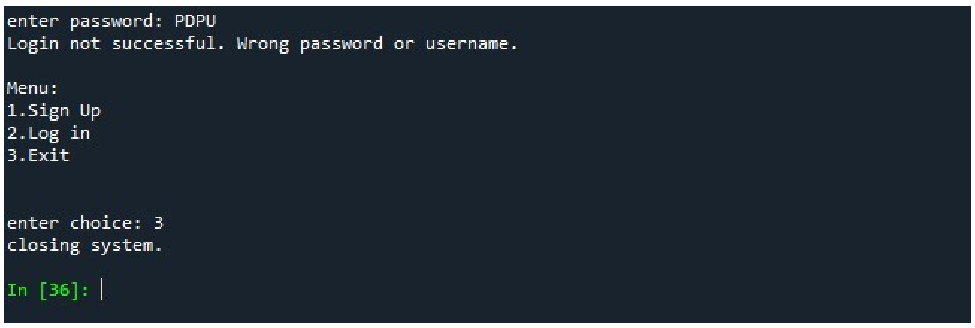
Output:



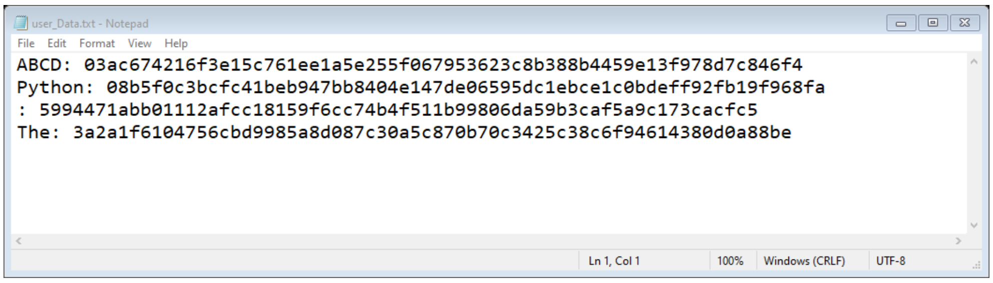
```
In [35]: runfile('D:/PDEU/Academics/Semester 5/Advanced Python Programming/Theory/
Assignment 1/Assignment1_Q6.py', wdir='D:/PDEU/Academics/Semester 5/Advanced Python
Programming/Theory/Assignment 1')
USER MANAGEMENT SYSTEM

Menu:
1.Sign Up
2.Log in
3.Exit


enter choice: 1

enter username: ABCD

enter password: 1234
Sign up successful.

Menu:
1.Sign Up
2.Log in
3.Exit


enter choice: 1

enter username: Python
```



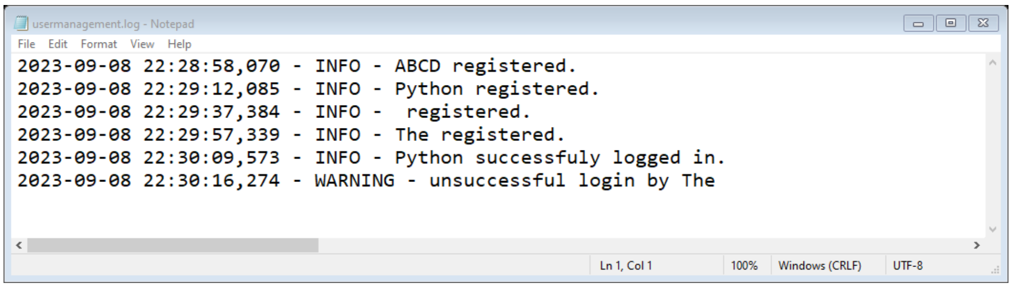
```
enter password: PDEU
Sign up successful.

Menu:
1.Sign Up
2.Log in
3.Exit


enter choice: 2

enter username: Python

enter password: Spyder
Log in successful.

Menu:
1.Sign Up
2.Log in
3.Exit


enter choice: 2

enter username: The

enter password: PDPU
Login not successful. Wrong password or username.
```



```
enter password: PDPU
Login not successful. Wrong password or username.

Menu:
1.Sign Up
2.Log in
3.Exit


enter choice: 3
closing system.

In [36]: 
```



```
ABCD: 03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4
Python: 08b5f0c3bcfc41beb947bb8404e147de06595dc1ebce1c0bdeff92fb19f968fa
: 5994471abb01112afcc18159f6cc74b4f511b99806da59b3caf5a9c173cacfc5
The: 3a2a1f6104756cbd9985a8d087c30a5c870b70c3425c38c6f94614380d0a88be
```



```
2023-09-08 22:28:58,070 - INFO - ABCD registered.
2023-09-08 22:29:12,085 - INFO - Python registered.
2023-09-08 22:29:37,384 - INFO -  registered.
2023-09-08 22:29:57,339 - INFO - The registered.
2023-09-08 22:30:09,573 - INFO - Python successfuly logged in.
2023-09-08 22:30:16,274 - WARNING - unsuccessful login by The
```

Lab Assignment 9: Image and Audio Data Processing

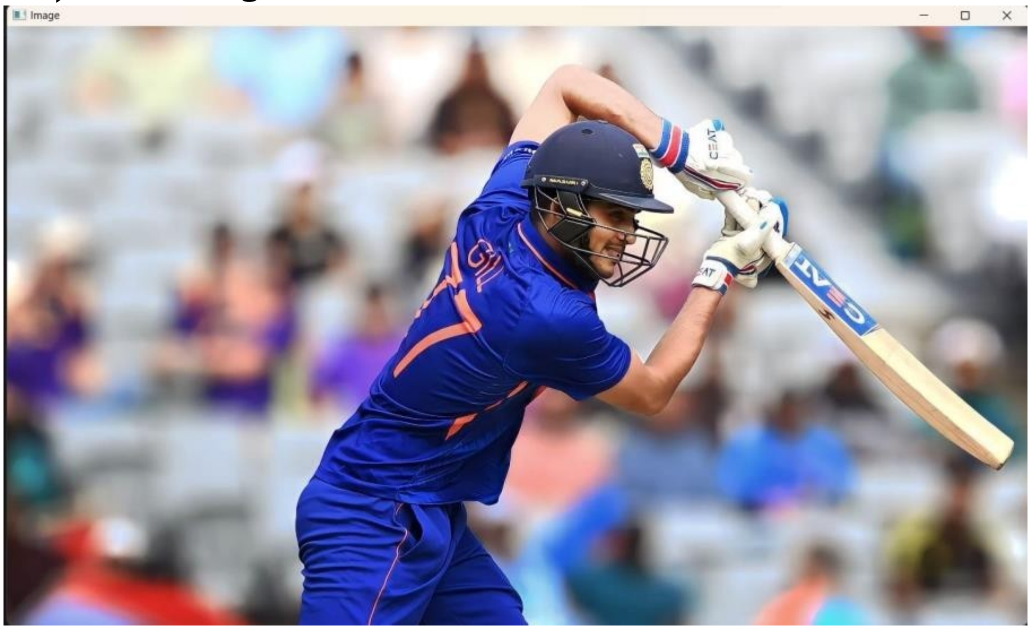
You are tasked with developing a comprehensive Python program that reads and manipulates both image and audio data. The goal is to create a tool that processes images and audio waveforms, allowing users to perform various operations on both types of data. This exercise aims to test your proficiency in handling different data formats and applying appropriate algorithms for manipulation.

Part 1: Image Data Processing

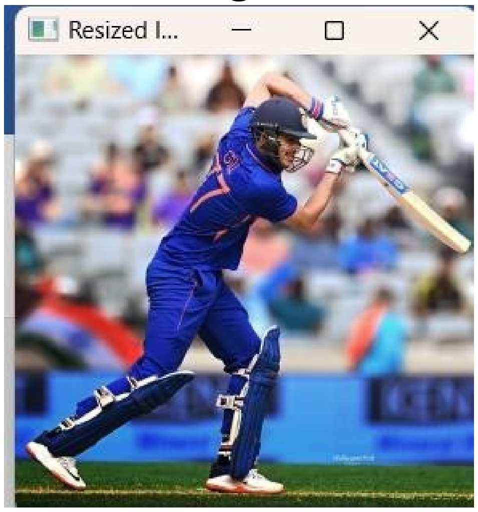
1. **Image Loading and Display:** Your program should allow users to load an image file and display it. Ensure you use an image processing library like Pillow (PIL) to handle image data.
2. **Image Manipulation:** Implement at least two image manipulation operations, such as:
    - Applying filters (e.g., Gaussian blur, edge detection).
    - Changing image dimensions or cropping.
    - Adjusting brightness, contrast, or saturation.
    - Converting to grayscale or other color spaces.
3. **Histogram Analysis:** Implement a feature that calculates and displays histograms for different color channels of the loaded image. Allow users to analyze and manipulate histogram data.

Output:

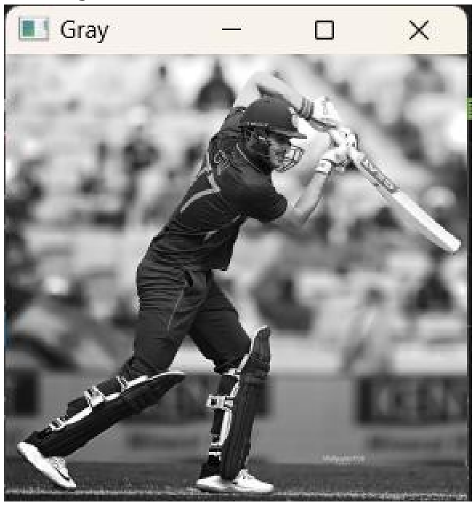
1) Main Image:



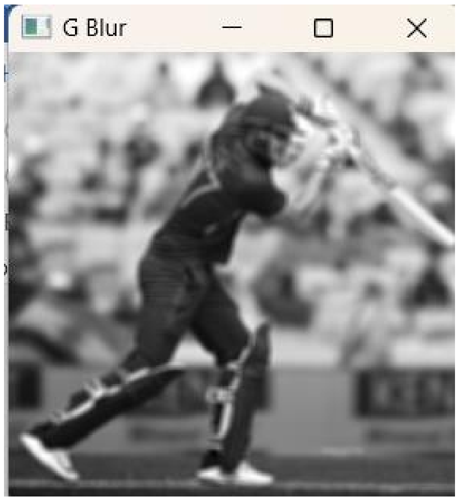
2) Resized Image:



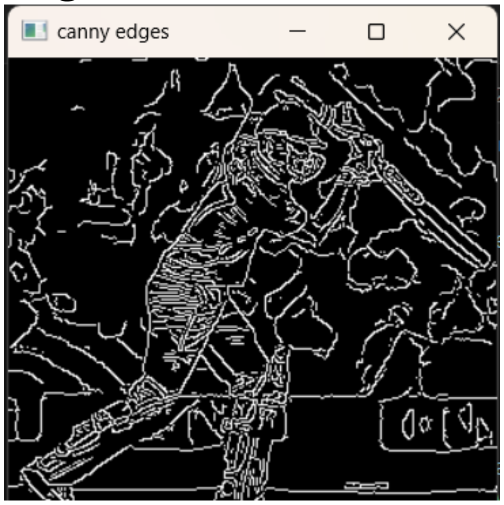
3) GrayScale:



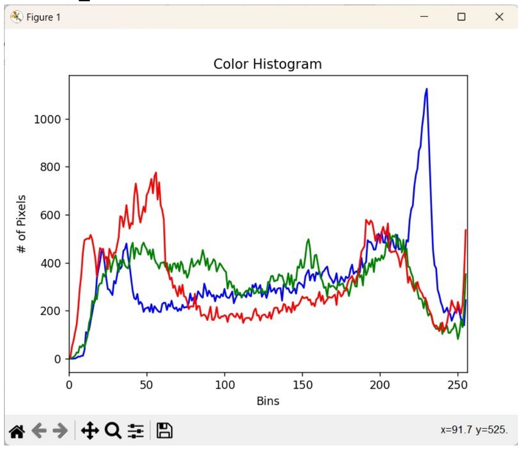
4) Gaussian Blur:



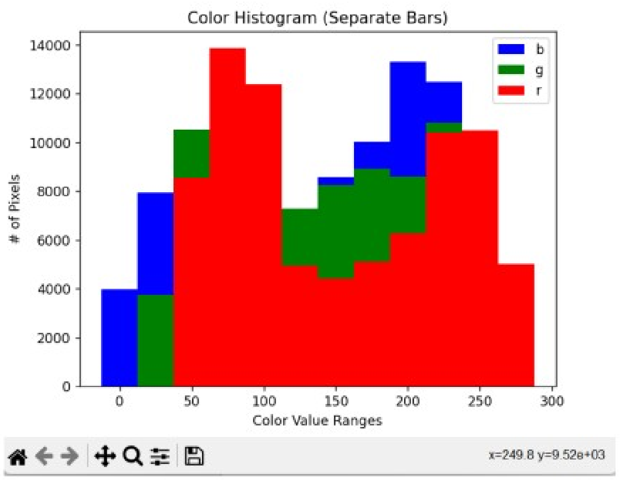
5) Edge Detection:



6) Graph:



7) Histogram:



Part 2: Audio Data Processing

1. **Audio Loading and Visualization:** Enable users to load an audio file and visualize its waveform using a suitable library like Matplotlib.
2. **Audio Manipulation:** Implement at least two audio manipulation operations, such as:
    - Changing the playback speed or pitch.
    - Applying audio effects (e.g., echo, reverb).
    - Clipping or trimming audio segments.
    - Adding silence or noise to the audio.
3. **Spectral Analysis:** Implement a feature that calculates and displays the spectrogram of the loaded audio. Allow users to analyze and manipulate spectral data.

LAB Assignment 10: Logging engineer

Analyze your project to identify potential places for logging involves understanding the application's structure, components, and potential points of interest where capturing information would be beneficial. Here's a systematic approach to help you identify these places:

Understand the Application's Purpose and Flow: Familiarize yourself with the application's functionality and objectives. Understand the user interactions, data processing steps, and overall flow of the program.
Identify Critical Components and Functions: Identify key components, functions, or methods that play a central role in the application's operation. These might include functions responsible for user input processing, data transformation, database interactions, or external API calls.
Identify Decision Points: Look for decision points in the application where different paths or outcomes are possible. These decision points often involve conditionals (if statements, switches, etc.) that determine the application's behaviour.
Identify External Interactions: Identify any interactions with external services, APIs, databases, or files. These interactions can provide insights into the data exchange between your application and external entities.
Identify Exception Handling: Pay attention to exception handling mechanisms in the application. Whenever an exception is caught, it's often helpful to log information about the exception, its context, and potential reasons for its occurrence.
Identify Loops and Iterations: Examine loops and iterations in your application. These might involve processing multiple items or steps in a repetitive manner. Logging within loops can help track progress and the values being processed.
Identify Inputs and Outputs: Look for points where the application interacts with user inputs, configuration settings, or external data sources. Logging inputs and outputs can help track data transformations and ensure that inputs are correctly processed.
Identify Troubleshooting Points: Consider where troubleshooting or debugging might be necessary in the future. These might be areas prone to errors or complex logic that might require detailed inspection.
Identify User Actions: If the application involves user interactions, consider logging user actions or events that help you understand how users are interacting with the software.
Consider Performance Monitoring: If performance is a concern, consider logging timing information to analyze the execution time of different components and identify potential bottlenecks.
Consult Documentation and Comments: Review any existing documentation, comments, or architectural diagrams that provide insights into the application's structure and behavior.
Brainstorm with Stakeholders: Discuss potential logging points with other developers, stakeholders, or users of the application. They might provide valuable insights into where logging would be most beneficial.
Think Like a Debugger: Put yourself in the shoes of someone who needs to debug the application. Where would you look for information to understand why something went wrong or to verify that everything is working as expected?

Once you've identified potential places for logging, you can strategically insert logging statements at these points. Remember to vary the logging levels (e.g., DEBUG, INFO, WARNING, ERROR, CRITICAL) based on the importance of the information being logged. Regularly reviewing and adjusting your logging strategy as the application evolves is crucial for maintaining effective and relevant logs.

Task- Find the potential places of logging write modules potential places in each module where you need of logging. Create a dummy code for your project. Every member in the project will select minimum two function and implement dummy code of that function with logging implementation.

LAB Assignment 11 : Data Analysis

You are working for a large e-commerce platform, and your task is to perform customer segmentation based on their shopping behavior. You have access to a dataset containing information about customer transactions. The dataset is in a CSV format and contains the following columns:

1. **CustomerID**: Unique identifier for each customer.
2. **TotalAmountSpent**: The total amount spent by each customer on the platform.
3. **TotalItemsPurchased**: The total number of items purchased by each customer.
4. **LastPurchaseDate**: The date of the customer's most recent purchase.
5. **AveragePurchaseValue**: The average value of each customer's purchases. Using NumPy and Pandas, your

goal is to perform the following tasks:

1. **Data Loading**: Load the dataset into a Pandas DataFrame for analysis.
2. **Data Cleaning**: Check for missing values, duplicates, or any inconsistencies in the data. If found, clean the data appropriately.
3. **Descriptive Statistics**: Calculate basic statistics such as mean, median, and standard deviation of **TotalAmountSpent** and **TotalItemsPurchased**.
4. **Customer Segmentation**: Divide the customers into segments based on their shopping behavior. You can use techniques like K-means clustering or any other method you prefer. For example, you might create segments like "High Spenders," "Frequent Shoppers," and "Inactive Customers."
5. **Visualization**: Create visualizations (e.g., scatter plots, bar charts) to represent the different customer segments you've identified.
6. **Customer Insights**: Provide insights into each customer segment. What distinguishes one segment from another? How can the e-commerce platform tailor its marketing strategies for each segment?
7. **Customer Engagement Recommendations**: Based on your analysis, provide recommendations for the e-commerce platform on how to engage with each customer segment more effectively. For example, should they offer discounts, provide personalized product recommendations, or run targeted marketing campaigns?

This problem requires you to use Pandas for data manipulation, NumPy for numerical operations, and potentially machine learning libraries for customer segmentation. It showcases the power of data analysis and segmentation for making data-driven decisions in e-commerce.

DataSet for the above analysis- https://www.kaggle.com/datasets/puneetbhaya/online-retail/

Dataset contains following information –

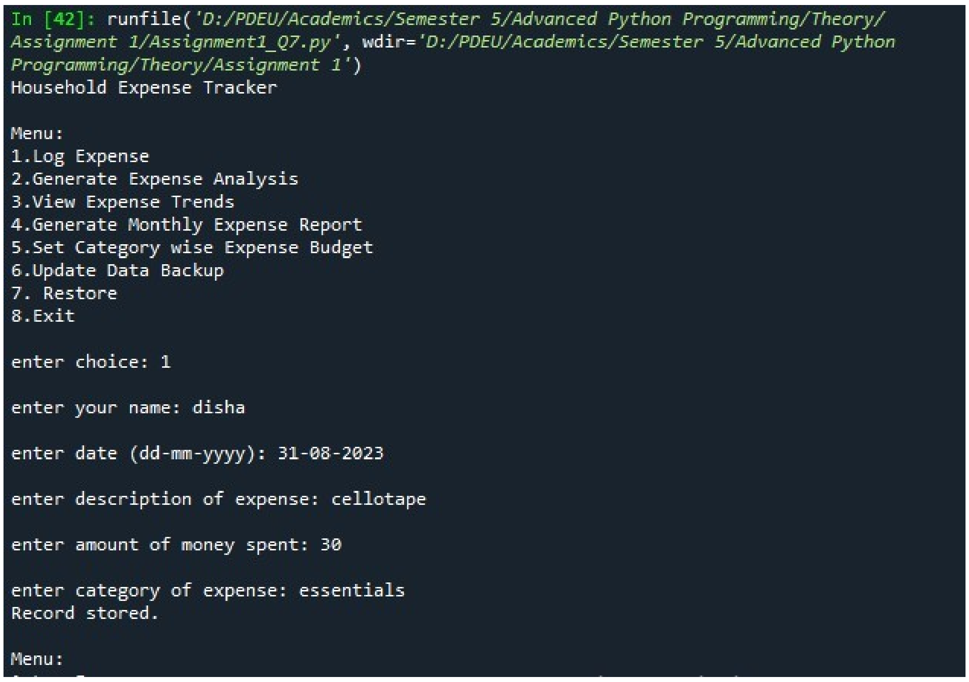
Description Quantity InvoiceDate UnitPrice CustomerID Country

Lab Assignment 12 : Household Expenses Tracker

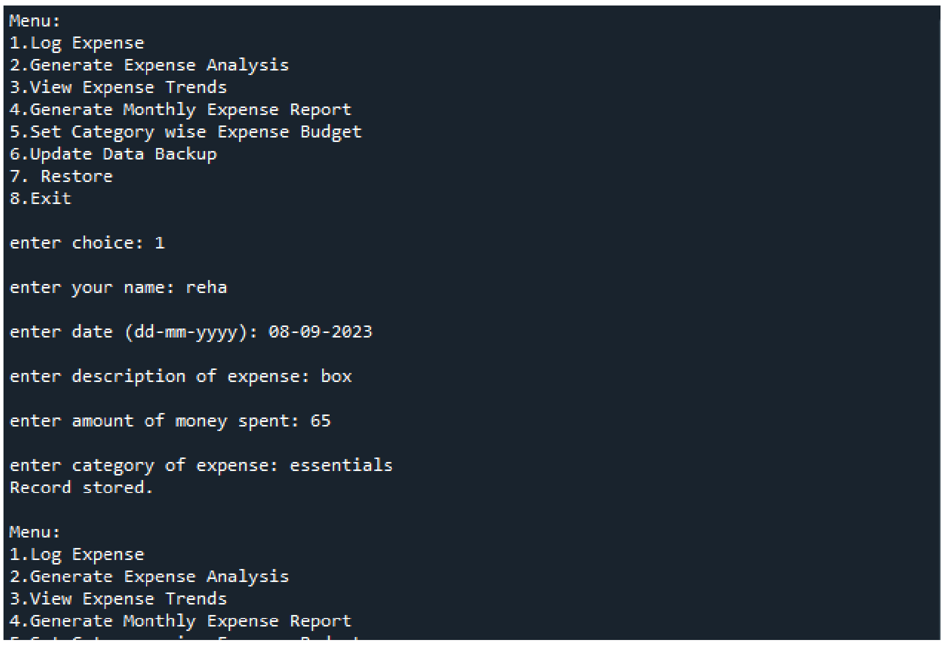
You have been tasked with creating a Python program to help manage household expenses. The program should allow family members to input their daily expenses, store them in a CSV file, and provide functionalities for analysis and reporting.

1. Expense Logging: Create a Python program that allows users to input their daily expenses. The program should prompt the user for their name, date of the expense, description, and amount spent. The data should be stored in a CSV file named expenses.csv with columns 'Name', 'Date', 'Description', and 'Amount'.
2. Expense Analysis: Develop a function that reads the expenses.csv file and calculates the total expenses for each family member. Display the total expenses for each member along with the average daily expense for the household.
3. Expense Trends: Implement a feature that generates a line chart using a plotting library (e.g., Matplotlib) to visualize the expense trends over the last month. The x-axis should represent the dates, and the y-axis should show the cumulative expenses for each day.
4. Expense Categorization: Enhance the program to allow users to categorize their expenses. Prompt the user to assign a category (e.g., groceries, utilities, entertainment) to each expense entry. Update the CSV file to include a 'Category' column.
5. Expense Reporting: Create a monthly expense report by reading the data from expenses.csv and generating a report that includes the following:
    - Total expenses for each family member for the month.
    - A breakdown of expenses by category.
    - A comparison of monthly expenses over different months using bar charts.
6. Expense Budgeting: Add an option for users to set a monthly budget for each category. After entering expenses, the program should calculate the remaining budget for each category and provide a warning if the budget is exceeded.
7. Data Backup and Restore: Implement a backup and restore feature that allows users to save a copy of the expenses.csv file to a backup location and restore it if needed. Handle cases where the file might be missing or corrupted.

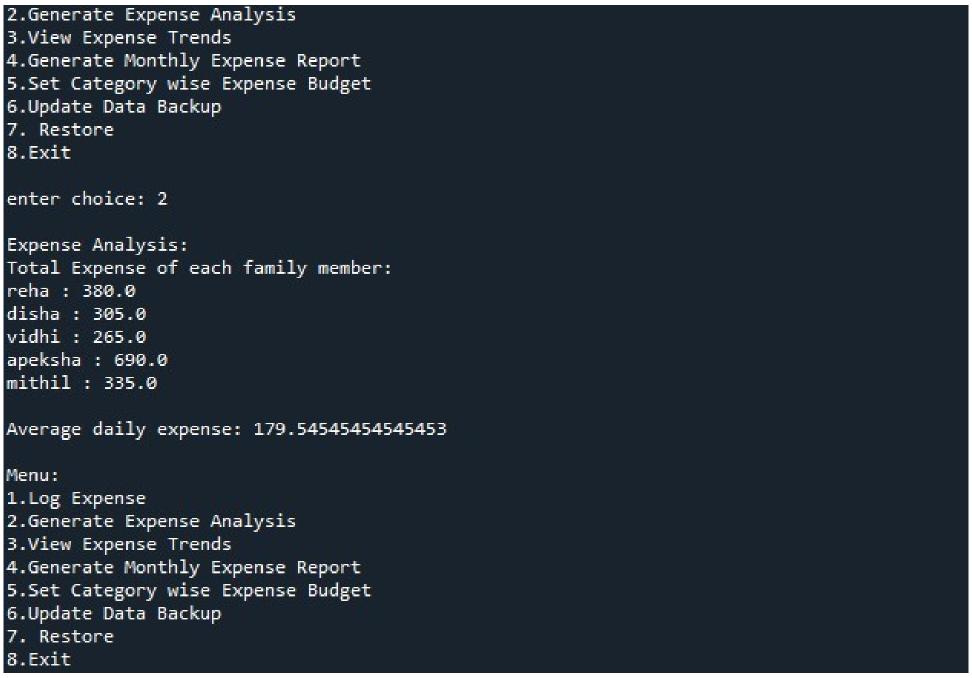
Output:



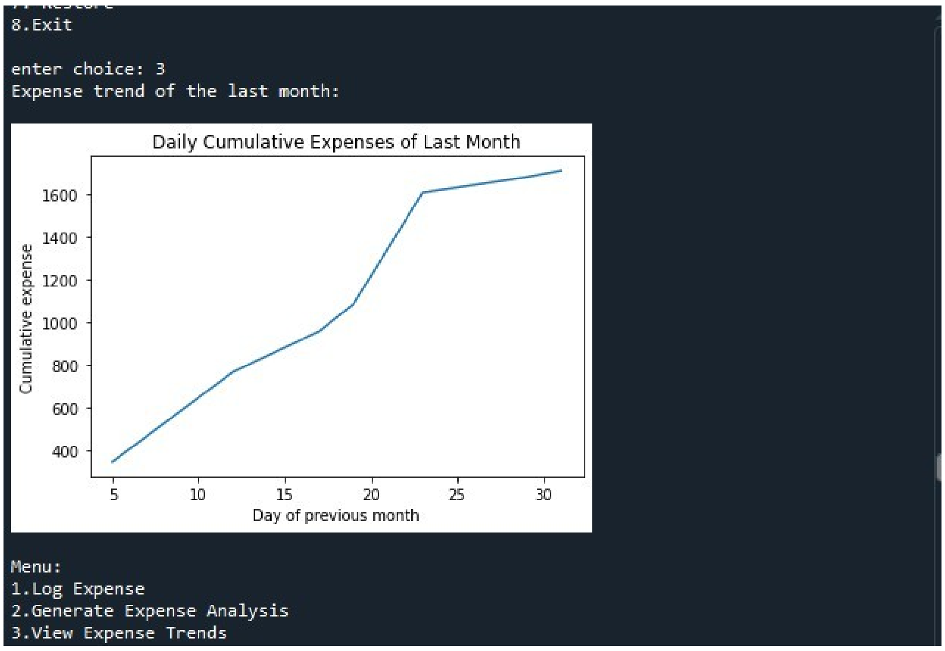
```
In [42]: runfile('D:/PDEU/Academics/Semester 5/Advanced Python Programming/Theory/
Assignment 1/Assignment1_Q7.py', wdir='D:/PDEU/Academics/Semester 5/Advanced Python
Programming/Theory/Assignment 1')
Household Expense Tracker

Menu:
1.Log Expense
2.Generate Expense Analysis
3.View Expense Trends
4.Generate Monthly Expense Report
5.Set Category wise Expense Budget
6.Update Data Backup
7. Restore
8.Exit

enter choice: 1

enter your name: disha

enter date (dd-mm-yyyy): 31-08-2023

enter description of expense: cellotape

enter amount of money spent: 30

enter category of expense: essentials
Record stored.

Menu:
```



```
Menu:
1.Log Expense
2.Generate Expense Analysis
3.View Expense Trends
4.Generate Monthly Expense Report
5.Set Category wise Expense Budget
6.Update Data Backup
7. Restore
8.Exit

enter choice: 1

enter your name: reha

enter date (dd-mm-yyyy): 08-09-2023

enter description of expense: box

enter amount of money spent: 65

enter category of expense: essentials
Record stored.

Menu:
1.Log Expense
2.Generate Expense Analysis
3.View Expense Trends
4.Generate Monthly Expense Report
```

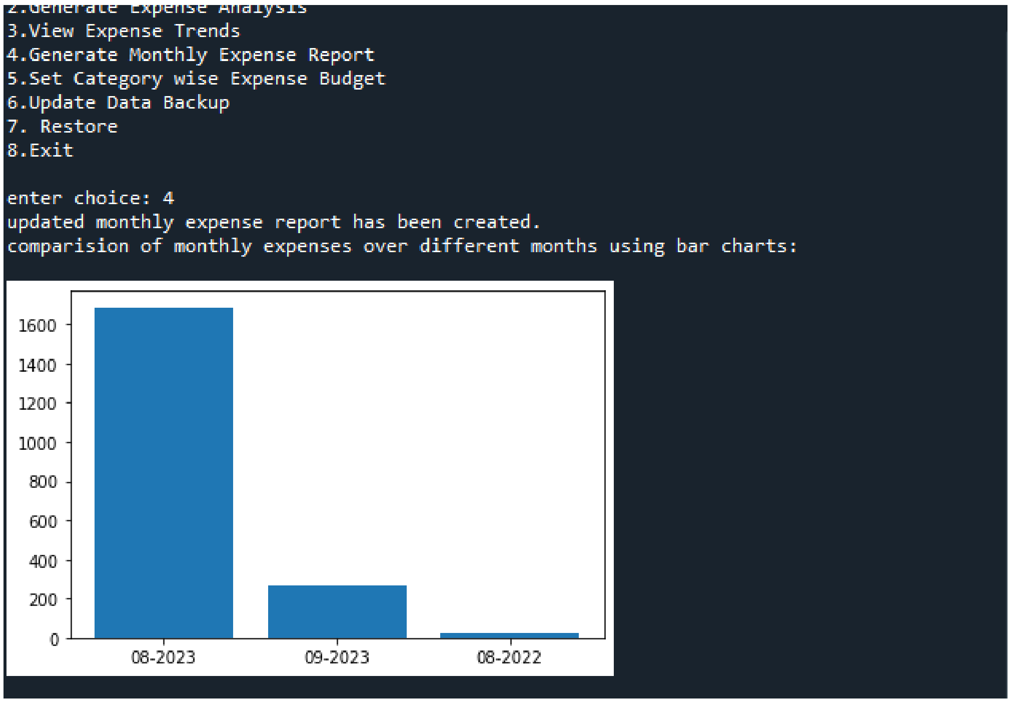


```
2.Generate Expense Analysis
3.View Expense Trends
4.Generate Monthly Expense Report
5.Set Category wise Expense Budget
6.Update Data Backup
7. Restore
8.Exit

enter choice: 2

Expense Analysis:
Total Expense of each family member:
reha : 380.0
disha : 305.0
vidhi : 265.0
apeksha : 690.0
mithil : 335.0

Average daily expense: 179.54545454545453

Menu:
1.Log Expense
2.Generate Expense Analysis
3.View Expense Trends
4.Generate Monthly Expense Report
5.Set Category wise Expense Budget
6.Update Data Backup
7. Restore
8.Exit
```

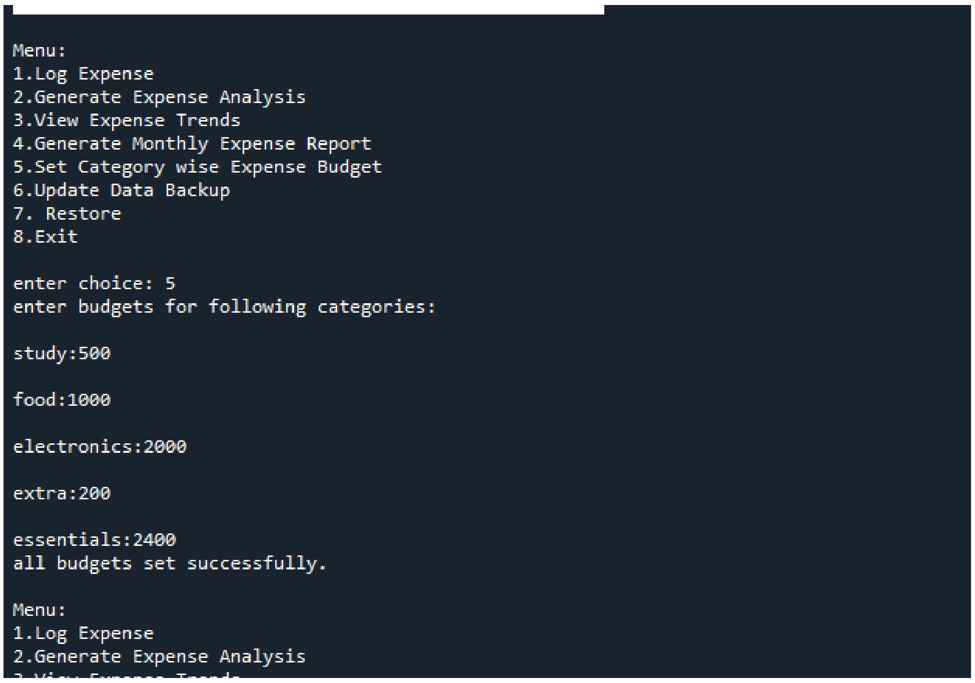


```
7. Restore
8.Exit

enter choice: 3
Expense trend of the last month:
```

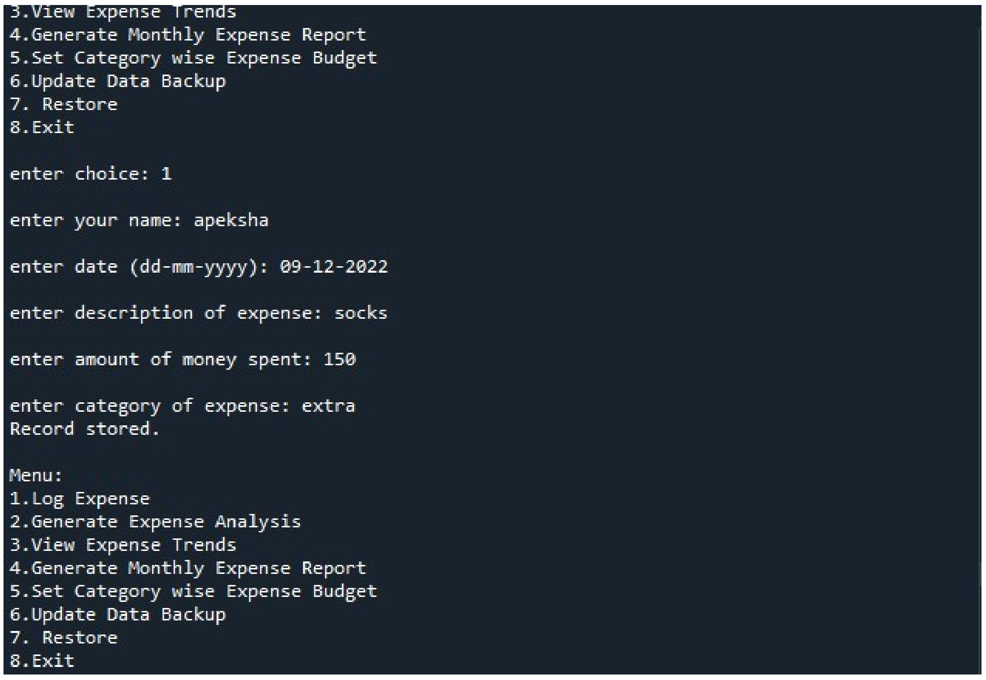
```
Menu:
1.Log Expense
2.Generate Expense Analysis
3.View Expense Trends
```



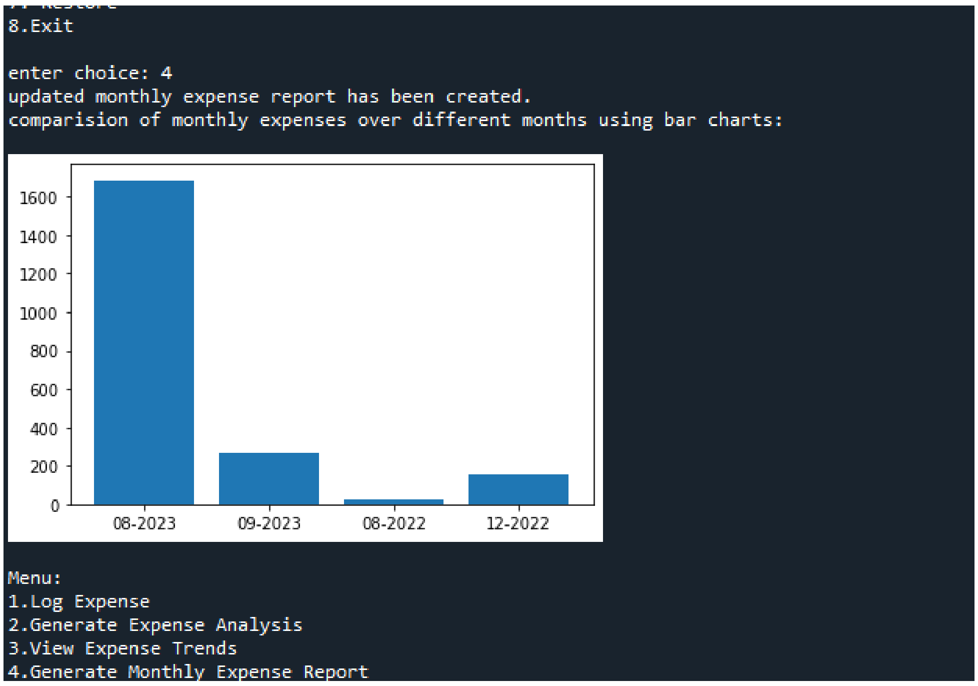
```
2.Generate Expense Analysis
3.View Expense Trends
4.Generate Monthly Expense Report
5.Set Category wise Expense Budget
6.Update Data Backup
7. Restore
8.Exit

enter choice: 4
updated monthly expense report has been created.
comparision of monthly expenses over different months using bar charts:
```



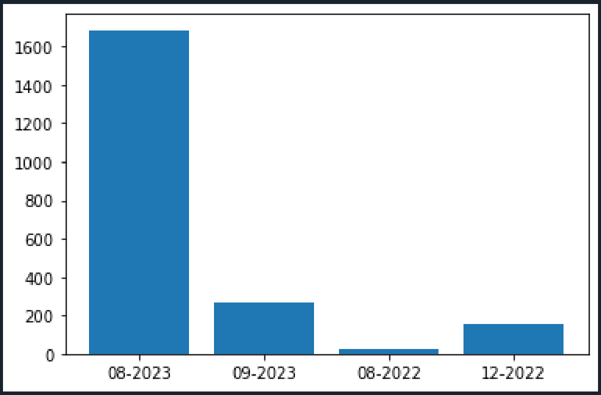
```
Menu:
1.Log Expense
2.Generate Expense Analysis
3.View Expense Trends
4.Generate Monthly Expense Report
5.Set Category wise Expense Budget
6.Update Data Backup
7. Restore
8.Exit

enter choice: 5
enter budgets for following categories:

study:500

food:1000

electronics:2000

extra:200

essentials:2400
all budgets set successfully.

Menu:
1.Log Expense
2.Generate Expense Analysis
```



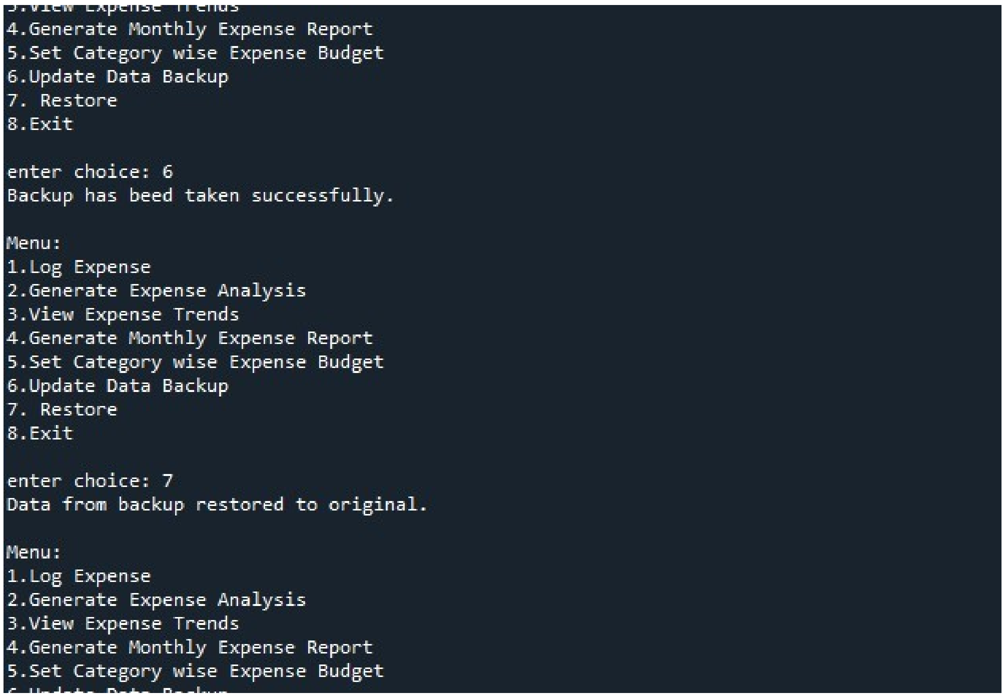
```
3.View Expense Trends
4.Generate Monthly Expense Report
5.Set Category wise Expense Budget
6.Update Data Backup
7. Restore
8.Exit

enter choice: 1

enter your name: apeksha

enter date (dd-mm-yyyy): 09-12-2022

enter description of expense: socks

enter amount of money spent: 150

enter category of expense: extra
Record stored.

Menu:
1.Log Expense
2.Generate Expense Analysis
3.View Expense Trends
4.Generate Monthly Expense Report
5.Set Category wise Expense Budget
6.Update Data Backup
7. Restore
8.Exit
```



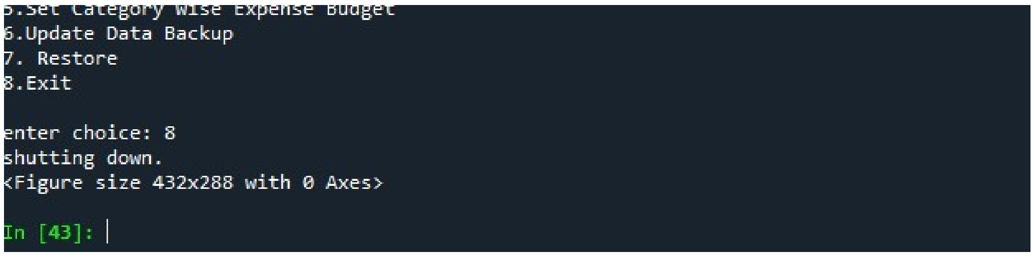
```
8.Exit

enter choice: 4
updated monthly expense report has been created.
comparision of monthly expenses over different months using bar charts:
```



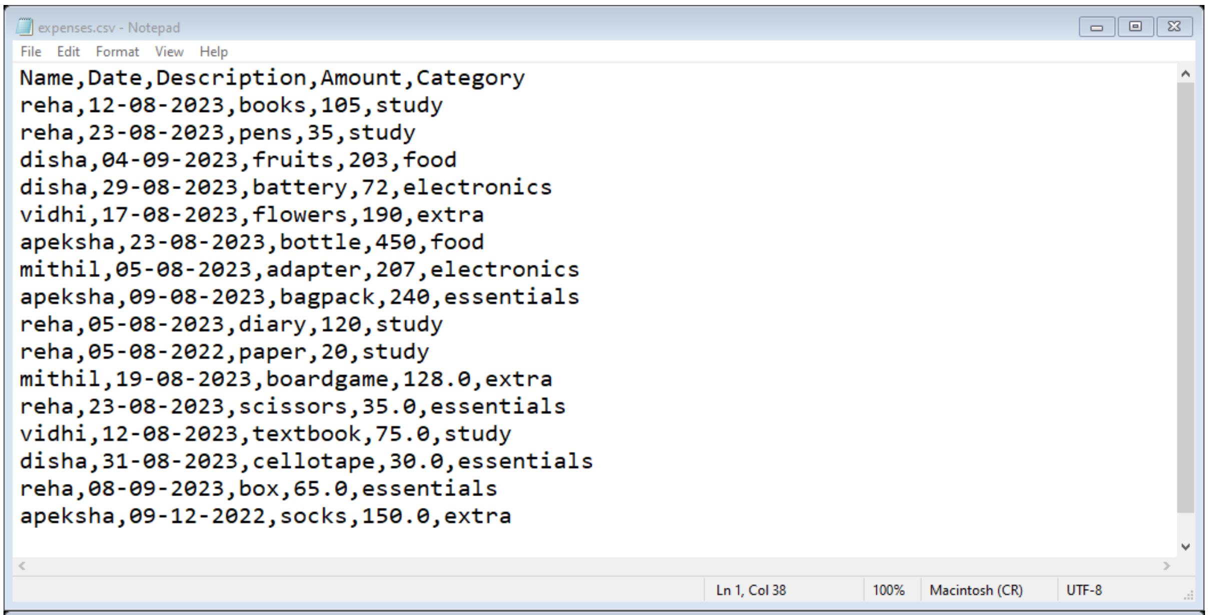
```
Menu:
1.Log Expense
2.Generate Expense Analysis
3.View Expense Trends
4.Generate Monthly Expense Report
```



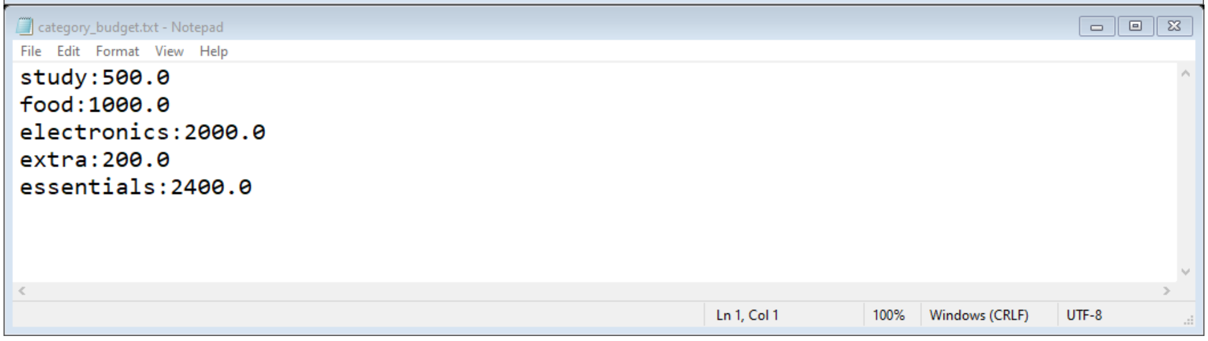
```
4.Generate Monthly Expense Report
5.Set Category wise Expense Budget
6.Update Data Backup
7. Restore
8.Exit

enter choice: 6
Backup has beed taken successfully.

Menu:
1.Log Expense
2.Generate Expense Analysis
3.View Expense Trends
4.Generate Monthly Expense Report
5.Set Category wise Expense Budget
6.Update Data Backup
7. Restore
8.Exit

enter choice: 7
Data from backup restored to original.

Menu:
1.Log Expense
2.Generate Expense Analysis
3.View Expense Trends
4.Generate Monthly Expense Report
5.Set Category wise Expense Budget
```



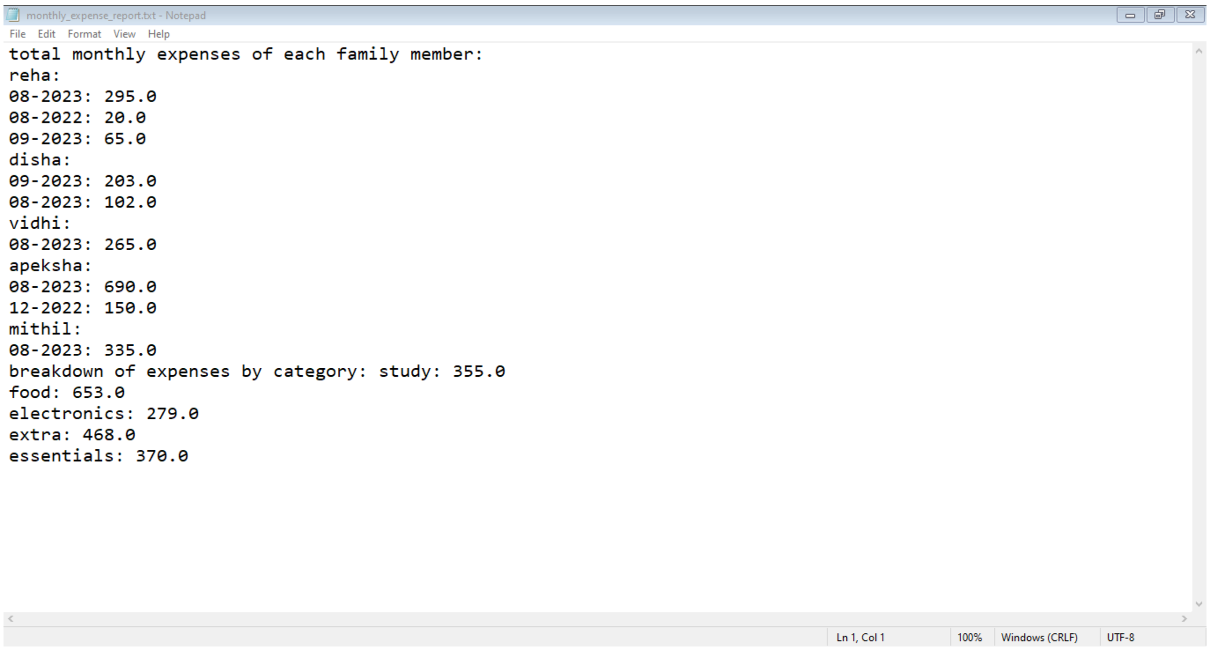
```
6.Update Data Backup
7. Restore
8.Exit

enter choice: 8
shutting down.
<Figure size 432x288 with 0 Axes>

In [43]:
```


expenses.csv - Notepad
File Edit Format View Help
Ln 1, Col 38 100% Macintosh (CR) UTF-8

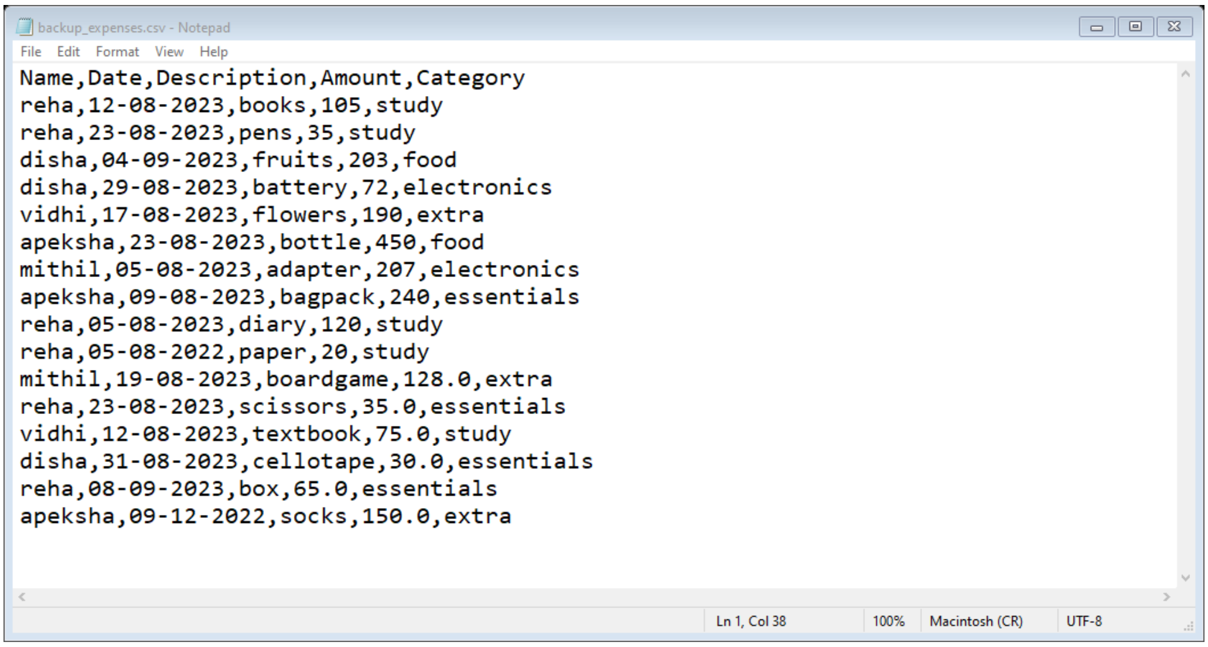
```
Name,Date,Description,Amount,Category
reha,12-08-2023,books,105,study
reha,23-08-2023,pens,35,study
disha,04-09-2023,fruits,203,food
disha,29-08-2023,battery,72,electronics
vidhi,17-08-2023,flowers,190,extra
apeksha,23-08-2023,bottle,450,food
mithil,05-08-2023,adapter,207,electronics
apeksha,09-08-2023,bagpack,240,essentials
reha,05-08-2023,diary,120,study
reha,05-08-2022,paper,20,study
mithil,19-08-2023,boardgame,128.0,extra
reha,23-08-2023,scissors,35.0,essentials
vidhi,12-08-2023,textbook,75.0,study
disha,31-08-2023,cellotape,30.0,essentials
reha,08-09-2023,box,65.0,essentials
apeksha,09-12-2022,socks,150.0,extra
```


category_budget.txt - Notepad
File Edit Format View Help
Ln 1, Col 1 100% Windows (CRLF) UTF-8

```
study:500.0
food:1000.0
electronics:2000.0
extra:200.0
essentials:2400.0
```


monthly_expense_report.txt - Notepad
File Edit Format View Help
Ln 1, Col 1 100% Windows (CRLF) UTF-8

```
total monthly expenses of each family member:
reha:
08-2023: 295.0
08-2022: 20.0
09-2023: 65.0
disha:
09-2023: 203.0
08-2023: 102.0
vidhi:
08-2023: 265.0
apeksha:
08-2023: 690.0
12-2022: 150.0
mithil:
08-2023: 335.0
breakdown of expenses by category: study: 355.0
food: 653.0
electronics: 279.0
extra: 468.0
essentials: 370.0
```


backup_expenses.csv - Notepad

File Edit Format View Help

```
Name,Date,Description,Amount,Category
reha,12-08-2023,books,105,study
reha,23-08-2023,pens,35,study
disha,04-09-2023,fruits,203,food
disha,29-08-2023,battery,72,electronics
vidhi,17-08-2023,flowers,190,extra
apeksha,23-08-2023,bottle,450,food
mithil,05-08-2023,adapter,207,electronics
apeksha,09-08-2023,bagpack,240,essentials
reha,05-08-2023,diary,120,study
reha,05-08-2022,paper,20,study
mithil,19-08-2023,boardgame,128.0,extra
reha,23-08-2023,scissors,35.0,essentials
vidhi,12-08-2023,textbook,75.0,study
disha,31-08-2023,cellotape,30.0,essentials
reha,08-09-2023,box,65.0,essentials
apeksha,09-12-2022,socks,150.0,extra
```

Ln 1, Col 38 | 100% | Macintosh (CR) | UTF-8