Python Development – Tumelo Matobo

Task 2: Python Script for Data Processing

Sales Data Processing Script

This Python script is designed to perform basic data processing operations on a sales dataset. It calculates summary statistics, filters data based on specific criteria, generates data visualizations (histograms and bar charts), and saves the processed data to a new file. The script is designed to be run multiple times with different datasets, making it a versatile tool for data processing.

Dependencies

- **Python:** The script is written in Python and requires a Python interpreter to run. If you haven't installed Python on your system, download and install it from the official website. https://www.python.org/downloads/
- **Pandas:** An open-source data analysis and manipulation tool, used for organising data into a DataFrame and exporting it to a CSV file.
- **Matplotlib:** A comprehensive Python library for creating static, animated, and interactive visualizations.

To install these dependencies, open a terminal or command prompt and run the following commands:

```
pip install pandas matplotlib
```

How to Run the Script

1. Ensure that Python and the required libraries are installed in your system.
2. Choose a dataset that is in a CSV format on your computer. Ensure that the dataset is in a compatible format for processing.
3. Update the **file_path** variable in the script to specify the path for your dataset CSV file. Customise the script based on your dataset and processing requirements.
4. Open a terminal or command prompt.
5. Navigate to the directory where the script is saved.
6. Run the script in a Python environment by using the following command:

```
python data_processing_script.py
```

Script Breakdown

- The script supports CSV datasets containing sample data from various domains such as sales, customer data, or survey responses.
- **Data Loading:** The script imports the pandas and matplotlib libraries and reads the sales dataset from a CSV file into a DataFrame named **df**.

- **Data Processing:** The script filters the data based on specific criteria (product line, year, and customer name). It also calculates summary statistics for the filtered data (sum, mean, median, mode, maximum, minimum, range, standard deviation).
- **Data Visualization:** The script generates a histogram for the filtered data using matplotlib. A bar chart for the 2004 motorcycle sales by customer name was also generated. The histograms and bar chart are saved as PNG files.
- **Saving Processed Data:** The filtered data and summary statistics are then saved to new CSV files.

Code

data_processing_script.py

```python
import pandas as pd
import matplotlib.pyplot as plt

#Load the dataset
file_path = 'C:\\Users\\acer\\OneDrive\\Desktop\\Python Development\\Task 2\\sales_data_sample.csv'
#Read the dataset
df = pd.read_csv(file_path , encoding='ISO-8859-1')

numeric_columns = ['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER', 'SALES']
productline_col = ["Motorcycles", "Classic Cars", "Trucks and Buses", "Vintage Cars", "Planes", "Ships", "Trains"]
yearid_col = [2003, 2004, 2005]
customer_name_col = ["Alpha Cognac", "Amica Models & Co.", "Anna's Decorations, Ltd", "Atelier graphique", "Australian Collectables, Ltd",
                     "Australian Collectors, Co.", "Australian Gift Network, Co", "Auto Assoc. & Cie.", "Auto Canal Petit",
                     "Auto-Moto Classics Inc.", "AV Stores, Co.", "Baane Mini Imports", "Bavarian Collectables Imports, Co.", "Blauer See Auto, Co.",
                     "Boards & Toys Co.", "CAF Imports", "Cambridge Collectables Co.", "Canadian Gift Exchange Network", "Classic Gift Ideas, Inc",
                     "Classic Legends inc.", "Clover Collections, Co.", "Collectable Mini Designs Co.", "Collectables For Less inc.", "Corporate Gift Ideas Co.",
                     "Corrida Auto Replicas, Ltd", "Cruz & Sons Co.", "Daedalus Designs Imports", "Danish Wholesale Imports", "Diecast Classics Inc.", "Diecast Collectables",
                     "Double Decker Gift Stores, Ltd", "Dragon Souvenirs, Ltd.","Enaco Distributors", "Euro Shopping Channel", "FunGiftIdeas.com", "Gift Depot Inc.", "Gift Ideas Corp.",
                     "Gifts4AllAges.com", "giftsbymail.co.uk", "Handji Gifts& Co", "Heintze Collectables", "Herkku Gifts", "Iberia Gift Imports, Corp.", "La Corne D'abondance, Co", "La Rochelle Gifts",
                     "Land of Toys Inc.", "L'ordine Souveniers", "Lyon Souveniers", "Marseille Mini Autos", "Marta's Replicas Co.", "Men R US Retailers, Ltd.", "Microsale Inc.", "Mini Auto Werke", "Mini Caravy",
                     "Mini Classics", "Mini Creations Ltd.", "Mini Gifts Distributors Ltd.", "Mini Wheels Co.", "Motor Mint Distributors Inc.", "Muscle Machine Inc", "Norway Gifts By Mail, Co."]

#Filter data based on specific criteria (productline, yearid and customername)
filtered_data = df[(df['PRODUCTLINE'] == "Vintage Cars") & (df['YEAR_ID'] == 2003) & (df['CUSTOMERNAME'] == "Euro Shopping Channel")][numeric_columns]

```

```
#Display the filtered dataframe
print(filtered_data.head)

#Calculate summary statistics for the filtered data
summary_stats = {}
for column in numeric_columns:
    column_data = filtered_data[column]
    summary_stats[column] = {
        'Sum': column_data.sum(),
        'Mean': column_data.mean(),
        'Mode': column_data.mode().iloc[0],
        'Median': column_data.median(),
        'Maximum': column_data.max(),
        'Minimum': column_data.min(),
        'Range': column_data.max() - column_data.min(),
        'Standard Deviation': column_data.std()
    }

#Print the summary statistics
for stat_type in ['Mean', 'Mode', 'Median', 'Maximum', 'Minimum', 'Range', 'Sum', 'Standard
Deviation']:
    print(f"\n{stat_type} Values:")
    for column, stats in summary_stats.items():
        print(f"{column}: {stats[stat_type]}")

#Select only numeric columns for the histogram
numeric_columns = ['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER', 'SALES']
filtered_numeric_data = filtered_data[numeric_columns]

#Create histogram for filtered data
for column in numeric_columns:
    plt.figure(figsize=(8,6))
    plt.hist(filtered_data[column], bins=20, color='skyblue', edgecolor = 'black')
    plt.title(f"Histogram of {column} for Vintage Cars sold in 2003 (Euro Shopping Channel)")
    plt.xlabel(column)
    plt.ylabel("Frequency")
    plt.savefig(f"{column}_histogram_filtered_data.png")
    plt.grid(True)
    plt.show()
    plt.close()
```

```
#Bar chart for 2004 Motorcycle sales
#Filter data for motorcycle sales in the year 2004
motorcycle_sales_2004 = df[(df['PRODUCTLINE'] == 'Motorcycles') & (df['YEAR_ID'] == 2004)]
#Calculate the total sales for each customer
customer_sales = motorcycle_sales_2004.groupby('CUSTOMERNAME')['SALES'].sum().reset_index()
#Plot the bar chart for motorcycle sales in 2004 by customer name
plt.figure(figsize=(10,6))
bar_width = 0.40
index = df.index
plt.bar(customer_sales['CUSTOMERNAME'], customer_sales['SALES'], color='purple')
plt.title("Bar Chart of Motorcycle Sales in 2004")
plt.xlabel("Customer Name")
plt.ylabel("Total Sales")
plt.xticks(rotation=90, ha='right')
plt.tight_layout()
plt.grid(True)
plt.show()
plt.savefig("motorcycle_sales_2004_bar_chart.png")

#Saving the filtered data to a new file
filtered_data.to_csv("filtered_data.csv", index=False)
#Saving the summary statistics to a new file
with open('summary_statistics.csv', 'w') as file:
    file.write("Column,Mean,Mode,Median,Maximum,Minimum,Range,Sum,Standard Deviation\n")
    for column, stats in summary_stats.items():
        file.write(f"{column},{stats['Mean']},{stats['Mode']},{stats['Median']},{stats['Maximum']},
        {stats['Minimum']},{stats['Range']},{stats['Sum']},{stats['Standard Deviation']}\n")
```

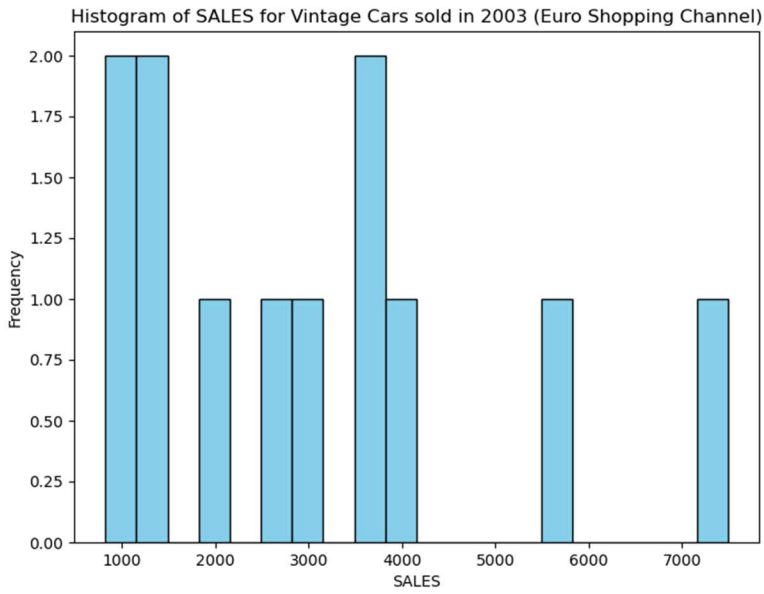
Output

1. **Summary Statistics:** The calculated summary statistics for the numeric columns are calculated.

```
Mean Values:
ORDERNUMBER: 10167.416666666666
QUANTITYORDERED: 34.416666666666664
PRICEEACH: 71.8925
ORDERLINENUMBER: 3.75
SALES: 3028.418333333333

Mode Values:
ORDERNUMBER: 10205
QUANTITYORDERED: 20
PRICEEACH: 100.0
ORDERLINENUMBER: 1
SALES: 820.4

Median Values:
ORDERNUMBER: 10154.5
QUANTITYORDERED: 34.0
PRICEEACH: 73.495
ORDERLINENUMBER: 3.5
SALES: 2925.59

Maximum Values:
ORDERNUMBER: 10205
QUANTITYORDERED: 49
PRICEEACH: 100.0
ORDERLINENUMBER: 7
SALES: 7492.4

Minimum Values:
ORDERNUMBER: 10128
QUANTITYORDERED: 20
PRICEEACH: 37.17
ORDERLINENUMBER: 1
SALES: 820.4

Range Values:
ORDERNUMBER: 77
QUANTITYORDERED: 29
PRICEEACH: 62.83
ORDERLINENUMBER: 6
SALES: 6672.0
```

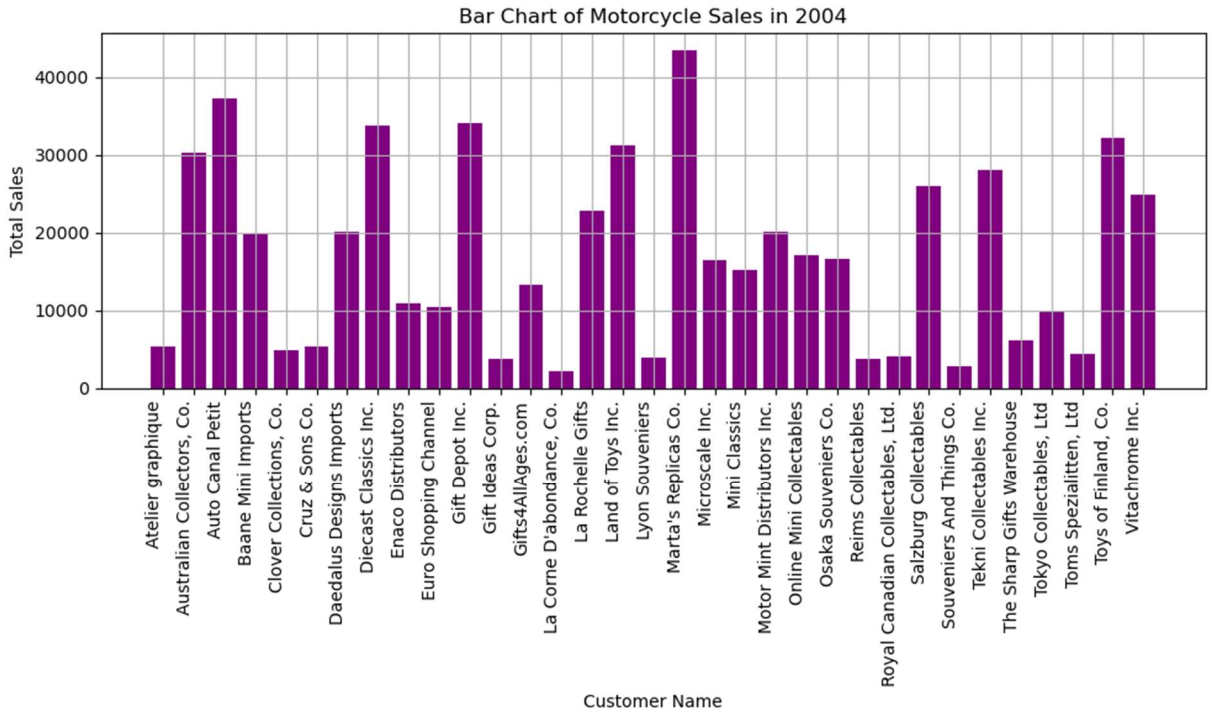
```
Sum Values:
ORDERNUMBER: 122009
QUANTITYORDERED: 413
PRICEEACH: 862.71
ORDERLINENUMBER: 45
SALES: 36341.02

Standard Deviation Values:
ORDERNUMBER: 34.48967785856921
QUANTITYORDERED: 9.95862653302364
PRICEEACH: 26.56608436778675
ORDERLINENUMBER: 2.378884383296277
SALES: 2022.8815790838994
```

2. **Data Visualizations:**
    a. **Histogram for SALES for Vintage Cars sold in 2003 (Euro Shopping Channel).**
    In this histogram the x-axis represents the range of sales values in 2003 and the y-axis represents the frequency of occurrence. Each bar in the histogram corresponds to a specific range of sales values, with the height of the bar indicating the number of sales falling within that range. Areas with the higher bars indicate ranges of sales where a greater number of transactions occurred.



b. Bar Chart for the 2004 Motorcycle sales by customer name

This bar chart illustrates the total sales of motorcycles for each customer who made purchases in the year 2004. The x-axis represents the customer names, while the y-axis indicates the total sales amount for each customer. Each bar in the chart corresponds to a customer who purchased motorcycles in 2004, with the height of the bar representing the total sales amount. Customers with taller bars made higher-value purchases, indicating their significant contribution to motorcycle sales in 2004.



3. **Filtered Data CSV File**

   The script saves the filtered data to a new CSV file containing only the numeric columns. The data is filtered based on specific criteria: productline, yearid, and customername. In this case, the filtered data includes sales transactions for vintage cars sold in 2003 to the Euro Shopping Channel. The CSV file contains only the numeric columns from the original dataset, which are: ORDERNUMBER, QUANTITYORDERED, PRICEEACH, ORDERLINENUMBER, and SALES.

	A	B	C	D	E	F
1	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	
2	10205	36	100	2	3735.72	
3	10205	48	63.61	1	3053.28	
4	10205	40	100	3	7492.4	
5	10128	41	100	2	5544.02	
6	10153	31	100	7	3641.57	
7	10128	43	92.16	1	3962.88	
8	10153	22	83.38	6	1834.36	
9	10205	32	37.17	5	1189.44	
10	10205	24	38.08	4	913.92	
11	10133	49	57.1	6	2797.9	
12	10133	27	50.19	7	1355.13	
13	10156	20	41.02	1	820.4	
14						

4. **Summary Statistics CSV File**

   The script calculates summary statistics for the filtered data and saves them to a new CSV file. The summary statistics include measures such as mean, mode, median, maximum, minimum, range, and standard deviation for each numeric column in the filtered dataset.

A	B	C	D	E	F	G	H	I
Column	Mean	Mode	Median	Maximum	Minimum	Range	Sum	Standard Deviation
ORDERNUMBER	10167.416666666666	10205	10154.5	10205	10128	77	122009	34.48967785856921
QUANTITYORDERED	34.416666666666664	20	34	49	20	29	413	9.958626533
PRICEEACH	71.8925	100	73.495	100	37.17	62.83	862.71	26.56608436778675
ORDERLINENUMBER	3.75	1	3.5	7	1	6	45	2.378884383296277
SALES	3028.418333333333	820.4	2925.59	7492.4	820.4	6672	36341.02	2022.8815790838994

The provided Python script offers a robust and versatile tool for processing sales data. By leveraging the power of pandas and matplotlib libraries, the script facilitates various data processing tasks, including data summarization, filtering based on specific criteria, and visualization of key insights. Through the calculated summary statistics and the generated histogram, users gain valuable insights into the distribution of sales amounts for a subset of the data. The ability to filter data based on criteria such as product line, year, and customer name enables targeted analysis and exploration of specific segments of the dataset. Furthermore, the script's flexibility allows for seamless integration with different datasets, making it a valuable asset for repetitive data processing tasks across various sales datasets. The saved filtered data in CSV format provides a structured output that can be further utilized for downstream analysis, reporting, or integration with other systems. Overall, this Python script serves as an efficient and adaptable solution for sales data processing, empowering users to derive meaningful insights and make informed decisions based on their sales datasets.