Mini Assignment 7. Data Aggregartion & Visualization

Please add one code cell after each question and create a program to answer the question. Make sure your code runs without error. After you are finished, click on File -> Print -> Save as PDF to create a PDF output and upload it on Brightspace.

NOTE: Make sure your PDF does not have your name or any identifying information in the name or content of the file. Anonymity is essential for the peer-review process.

Data Aggregation: Black Friday

Here we have the dataset for Black Friday shoppers' purchase records acquired from Kaggle and available for download here. The dataset includes 12 columns and more than 500K rows. It contains the following variables:

- **User_ID**
- **Product_ID**
- **Gender**
- **Age**
- **Occupation**
- **City_Category**
- **Stay_In_Current_City_Years**
- **Marital_Status**
- **Product_Category_1**
- **Product_Category_2**
- **Product_Category_3**
- **Purchase**

1. Read the dataset into a dataframe, and check out the first few rows and column data types.

```
from google.colab import drive
drive.mount('/content/drive')

     Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
from numpy import nan
from matplotlib import pyplot as plt
```

```
data = pd.DataFrame(pd.read_csv('/content/drive/MyDrive/colabData/BlackFriday.csv', na_values
```

```
data.info()
data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 537577 entries, 0 to 537576
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   User_ID                     537577 non-null  int64  
 1   Product_ID                  537577 non-null  object 
 2   Gender                      537577 non-null  object 
 3   Age                         537577 non-null  object 
 4   Occupation                  537577 non-null  int64  
 5   City_Category               537577 non-null  object 
 6   Stay_In_Current_City_Years  537577 non-null  object 
 7   Marital_Status              537577 non-null  int64  
 8   Product_Category_1          537577 non-null  int64  
 9   Product_Category_2          370591 non-null  float64
 10  Product_Category_3          164278 non-null  float64
 11  Purchase                    537577 non-null  int64  
dtypes: float64(2), int64(5), object(5)
memory usage: 49.2+ MB
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Ye
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	

2. Display the min, average, and max of Purchase. It should be calculated and displayed in the same command.

```
data.Purchase.agg(["min", "mean", "max"])
```

```
min       185.000000
mean     9333.859853
max     23961.000000
Name: Purchase, dtype: float64
```

3. Display the number of products sold in each 'Product_Category_1'. Next, display the number of **UNIQUE** products sold in each 'Product_Category_1'.

Hint: for the second part of the question, you may use .nunique() to count the number of unique items in any group.

```
data["Product_Category_1"].count()
```

```
537577
```

```
data["Product_Category_1"].nunique()
```

```
18
```

4. Use groupby to display the average and standard deviation of the Purchase for each Gender and City_Category. Round the numbers to show only one decimal place.

```
data.groupby(["Gender", "City_Category"])[["Purchase"]].agg(["mean","std"]).round(1)
```

		Purchase	
		mean	std
Gender	City_Category		
F	A	8630.8	4642.5
	B	8590.5	4648.5
	C	9265.0	4854.4
M	A	9061.7	4931.4
	B	9400.8	4999.9
	C	10033.2	5175.9

5. Display the top three most profitable customers (User_ID & total purchase should be displayed).

   Hint: you should get the sum of Purchase for each customer and then use an aggregate function that displays the top largest customers in terms of their purchase. This should be all calculated and displayed in the same command.

```
data.groupby("User_ID")[["Purchase"]].sum().sort_values("Purchase", ascending = False).head()
```

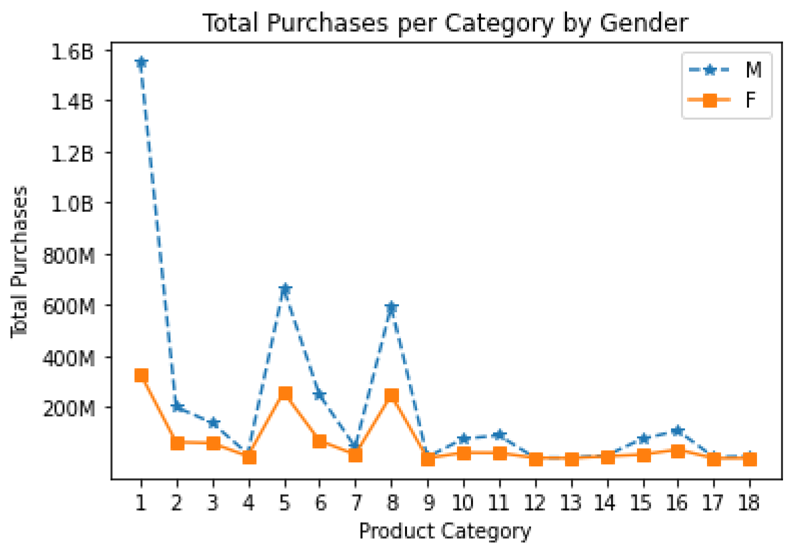
	Purchase
User_ID	
1004277	10536783
1001680	8699232
1002909	7577505
1001941	6817493

6. Use pivot table to display the total Purchase for each Product_Category_1 (rows) and Gender (columns). Next, use the pivot table to create a line plot with two lines: one for male purchase across categor_1, and another for female purchase across category_1. Set proper line style, marker, color, legend, plot title, and axes labels.

   Which category_1 values are more popular for each of the male and female customers?

```
pivot = pd.pivot_table(data, values = 'Purchase', index = 'Product_Category_1', columns = 'Ge
```

```
plt.plot(pivot.M, marker = '*', linestyle = "--", label = "M")
plt.plot(pivot.F, marker = 's', linestyle = "-", label = "F")
plt.title("Total Purchases per Category by Gender")
plt.xlabel("Product Category")
plt.ylabel("Total Purchases")
plt.xticks([x+1 for x in range(18)])
plt.yticks([200000000, 400000000, 600000000, 800000000, 1000000000, 1200000000, 1400000000,
plt.legend()
plt.show()
```



Data Visualization

7. Read the population dataset and financial dataset separately as dataframes. No need to change the index names.

```
fin_csv = pd.read_csv('/content/drive/MyDrive/colabData/wdi_data.csv')
pop_csv = pd.read_csv('/content/drive/MyDrive/colabData/wdi_population.csv')
```

```
fin_df = pd.DataFrame(fin_csv)
pop_df = pd.DataFrame(pop_csv)
```

8. Create a slice of financial dataset for 2017 data, and create another slice for 2000 data; store them in separate variables called fin2017 and fin2000.

   Use the above two variables to creae a line plot with two lines; one displays the GDP of the countries for 2017 and the other one displays the GDP of the countries for 2000. Make sure to:

- set the x axis as country names and y axis as GDP
- set proper linestyle and marker for each line
- set proper color for line and markers
- set proper legend for the plot with labels for each line
- set proper plot title and axes labels

```
fin2017 = fin_df.query('year == 2017')
fin2000 = fin_df.query('year == 2000')
```

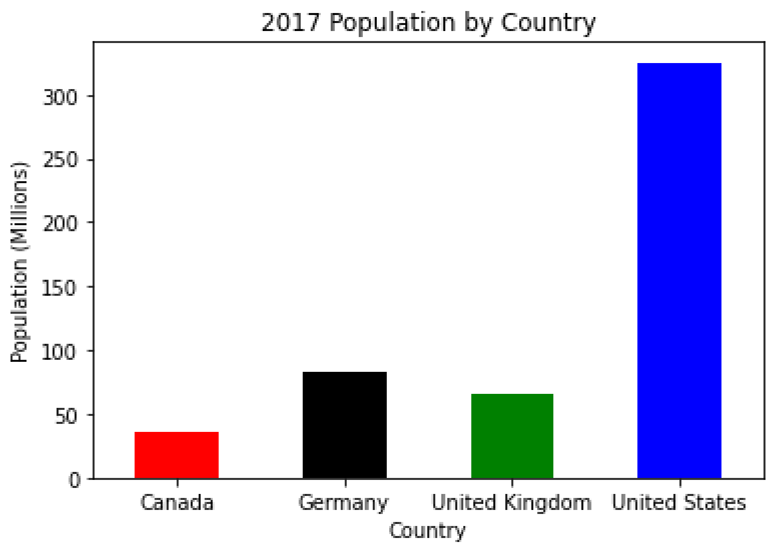
```
plt.plot(fin2017.country, fin2017.GDP, label = "2017", linestyle = "--", marker = 'D', c = "B
plt.plot(fin2000.country, fin2000.GDP, label = "2000", linestyle = ":", marker = 's', c = "Gr
plt.title("GDP 2000 vs. 2017")
plt.xlabel("Country")
plt.ylabel("GDP")
plt.legend()
plt.show()
```

GDP 2000 vs. 2017
2017
2000
18
16
14
12

9. Slice the population dataset for the year 2017 records, and store the slice in 'pop2017' variable. Next, create a bar plot to compare the population of the countries in 2017. Make sure to set proper bar color, plot title, and axes labels.

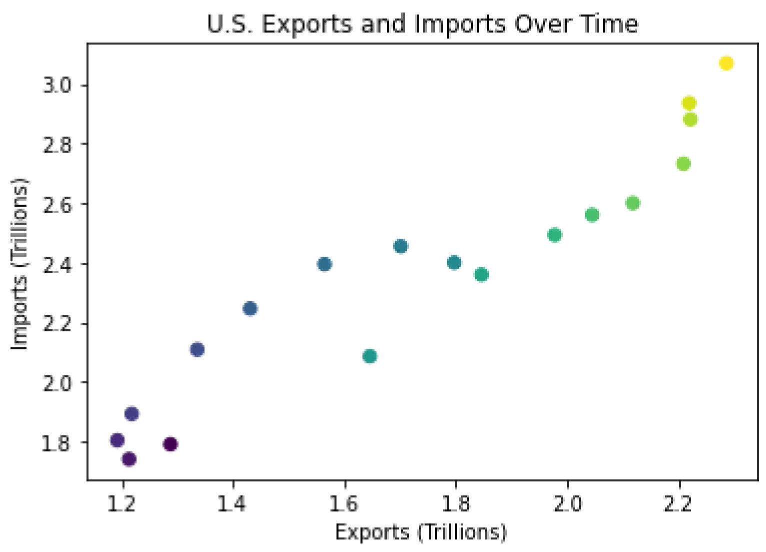
```
pop2017 = pop_df.query('year == 2017')
```

```
colors = ["Red", "Black", "Green", "Blue"]
pop2017.plot(kind = "bar", x = "country", y = "Population", rot = 0, title = "2017 Population
plt.legend().remove()
```



10. Slice the financial dataset for the rows for the United States. Then, use that to create a scatter plot to display the 'Exports' against the 'Imports' of the United States, and use the 'year' as color for the markers; it means the markers (data points) of later year will be shown with brighter colors on the scatterplot.

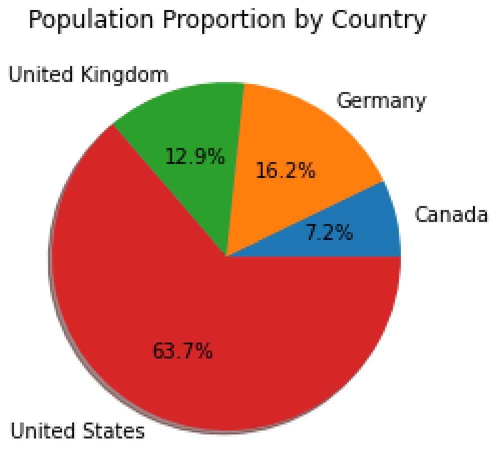
```
fin_df_us = fin_df.query("country == 'United States'")


plt.scatter(x = fin_df_us.Exports, y = fin_df_us.Imports, c = fin_df_us.year, s = 40)
plt.title("U.S. Exports and Imports Over Time")
plt.xlabel("Exports (Trillions)")
plt.ylabel("Imports (Trillions)")
plt.show()
```



11. Use pop2017 dataframe to create a pie chart that displays the countries' population as a portion of the total population. Make sure to properly label the portions of the pie chart with country names, and the percentage of population they hold (the percentage should be flot with one decimal with a % sign). Explode the 'Canada' portion of the pie chart and properly title the plot.

```
plt.pie(pop2017.Population, labels = pop2017.country, shadow = True, autopct = '%1.1f%%')
plt.title("Population Proportion by Country")
```

```
    Text(0.5, 1.0, 'Population Proportion by Country')
```


Population Proportion by Country
United Kingdom
Germany
Canada
United States