Just IT

B2W group

Apprenticeships | Training | Recruitment

Data Technician

Name: Sergios Vasileiou

Course Date: 2/2/25

Table of contents

Day 2: Task 1

It is a common software development interview question to create the below with a certain programming language. Create the below using Python syntax, test it and past the completed syntax and output below.

FizzBuzz:

Go through the integers from 1 to 100.
If a number is divisible by 3, print "fizz."
If a number is divisible by 5, print "buzz."
If a number is both divisible by 3 and by 5, print "fizzbuzz."
Otherwise, print just the number.

Paste your completed work to the right

```python
# FizzBuzz logic for the given range
for num in range(1, 101):
    if num % 3 == 0 and num % 5 == 0:
        print("FizzBuzz", end=" ")
    elif num % 3 == 0:
        print("Fizz", end=" ")
    elif num % 5 == 0:
        print("Buzz", end=" ")
    #Printing it horizontally
    else:
        print(num, end=" ")
```

```
1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz 16
```

Day 3: Task 1

Download the 'student.csv', complete the below exercises as a group and paste your input and output. Although this is a group activity, everyone should have the below answered so it supports your portfolio:

Exercise 1: Loading and Exploring the Data

1. Question: "Write the code to read a CSV file into a Pandas DataFrame."
2. Question: "Write the code to display the first 5 rows of the DataFrame."
3. Question: "Write the code to get the information about the DataFrame."
4. Question: "Write the code to get summary statistics for the DataFrame."

```
#First we import the pandas library
import pandas as pd
#Then we write the command to read our csv file
df = pd.read_csv('student.csv')  #Q1
#Bringing the first five rows of data
df.head(5) #Q2
#Acquiring the information on our file
df.info() #Q3
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35 entries, 0 to 34
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      35 non-null     int64
 1   name    34 non-null     object
 2   class   34 non-null     object
 3   mark    35 non-null     int64
 4   gender  33 non-null     object
dtypes: int64(2), object(3)
memory usage: 1.5+ KB
```

```
#First we import the pandas library
import pandas as pd
#Then we write the command to read our csv file
df = pd.read_csv('student.csv')  #Q1
#Bringing the first five rows of data
df.head(5) #Q2
#Acquiring the information on our file
#df.info() #Q3
#We acquire a summary of the statistical info of our data
df.describe() #Q4
```

	id	mark
count	35.000000	35.000000
mean	18.000000	74.657143
std	10.246951	16.401117
min	1.000000	18.000000
25%	9.500000	62.500000
50%	18.000000	79.000000
75%	26.500000	88.000000
max	35.000000	96.000000

Exercise 2: Indexing and Slicing

1. Question: "Write the code to select the 'name' column."
2. Question: "Write the code to select the 'name' and 'mark' columns."
3. Question: "Write the code to select the first 3 rows."
4. Question: "Write the code to select all rows where the 'class' is 'Four'."

```python
#First we import the pandas library
import pandas as pd
#Then we write the command to read our csv file
df = pd.read_csv('student.csv')
#Selecting the "Name" and "Mark" columns
df[['name','mark']] #Q2
```

	name	mark
0	John Deo	75
1	Max Ruin	85
2	Arnold	55
3	Krish Star	60
4	John Mike	60

```python
#First we import the pandas library
import pandas as pd

#Then we write the command to read our csv file
df = pd.read_csv('student.csv')

#Selecting the "Name" and "Mark" columns
#df[['name','mark']]  #Q2

#Selecting the first three roaw
df.head(3) #Q3
```

	id	name	class	mark	gender
0	1	John Deo	Four	75	female
1	2	Max Ruin	Three	85	male
2	3	Arnold	Three	55	male

```python
#First we import the pandas library
import pandas as pd

#Then we write the command to read our csv file
df = pd.read_csv('student.csv')

#Selecting the "Name" and "Mark" columns
#df[['name','mark']]  #Q2

#Selecting the first three roaw
#df.head(3) #Q3

#Finding the students in class 4
df[df['class'] == 'Four'] #Q4
```

	id	name	class	mark	gender
0	1	John Deo	Four	75	female
3	4	Krish Star	Four	60	female
4	5	John Mike	Four	60	female
5	6	Alex John	Four	55	male
9	10	Big John	Four	55	female
15	16	Gimmy	Four	88	male
20	21	Babby John	Four	69	female
30	31	Marry Toeey	Four	88	male

Exercise 3: Data Manipulation

1. Question: "Write the code to add a new column 'passed' that indicates whether the student passed (mark >= 60)."
2. Question: "Write the code to rename the 'mark' column to 'score'."
3. Question: "Write the code to drop the 'passed' column."

```python
#First we import the pandas library
import pandas as pd

#Then we write the command to read our csv file
df = pd.read_csv('student.csv')

#Creating a "passed" column,renaming 'mark' to 'score',dropping "passed" column
df['passed'] = df['mark'] >= 60 #Q1
df = df.rename(columns={'mark': 'score'}) #Q2
df = df.drop('passed', axis=1) #Q3

#Showcasing the first five rows with our changes included
df.head(5)
```

	id	name	class	score	gender
0	1	John Deo	Four	75	female
1	2	Max Ruin	Three	85	male
2	3	Arnold	Three	55	male
3	4	Krish Star	Four	60	female
4	5	John Mike	Four	60	female

Exercise 4: Aggregation and Grouping

1. Question: "Write the code to group the DataFrame by the 'class' column and calculate the mean 'mark' for each group."
2. Question: "Write the code to count the number of students in each class."
3. Question: "Write the code to calculate the average mark for each gender."

```
# Group by 'class' and calculate the mean of 'mark'
class_mean_score = df.groupby("class")["score"].mean()
print(class_mean_score) #Q1
```

```
class
Eight    79.000000
Fifth    78.000000
Five     80.000000
Four     68.750000
Nine     41.500000
Seven    77.600000
Six      82.571429
Three    73.666667
Name: score, dtype: float64
```

```
#Counting each student in every class
class_counts = df['class'].value_counts()
print(class_counts) #Q2
```

```
class
Seven    10
Four      8
Six       7
Three     3
Five      2
Nine      2
Fifth     1
Eight     1
Name: count, dtype: int64
```

```
#Calculating the avg score between the genders
gender_means = df.groupby('gender')['score'].mean()
print(gender_means) #Q3
```

```
gender
female    77.312500
male      71.588235
Name: score, dtype: float64
```

Exercise 5: Advanced Operations

1. Question: "Write the code to create a pivot table with 'class' as rows, 'gender' as columns, and 'mark' as values."
2. Question: "Write the code to create a new column 'grade' where marks >= 85 are 'A', 70-84 are 'B', 60-69 are 'C', and below 60 are 'D'."
3. Question: "Write the code to sort the DataFrame by 'mark' in descending order."

```python
#Creating the pivot table
pivot_table = df.pivot_table(values="score", index="class", columns="gender", aggfunc="mean")
print(pivot_table) #Q1

#Functions to assign grades based on marks
def assign_grade(score):
    if score >= 85:
        return "A"
    elif score >= 70:
        return "B"
    elif score >= 60:
        return "C"
    else:
        return "D"

#Appling the function to the 'score' column to create the 'grade' column
df['grade'] = df['score'].apply(assign_grade) #Q2

#Sorting in descending order
df_sorted = df.sort_values(by=['grade'], ascending=False) #Q3

print(df_sorted.head(15))
```

```
gender  female  male
class
Eight      NaN  79.0
Fifth      NaN  78.0
Five       NaN  80.0
Four      63.8  77.0
Nine      65.0  18.0
Seven     81.4  73.8
Six       89.2  54.0
Three      NaN  70.0
    id         name  class  score  gender grade
9   10     Big John   Four     55  female     D
28  29  Tess Played  Seven     55    male     D
2    3       Arnold  Three     55    male     D
16  17        Tumyu    Six     54    male     D
5    6    Alex John   Four     55    male     D
21  22       Reggid  Seven     55  female     D
18  19        Tinny   Nine     18    male     D
20  21   Babby John   Four     69  female     C
19  20       Jackly   Nine     65  female     C
33  34     Gain Toe  Seven     69    male     C
4    5    John Mike   Four     60  female     C
3    4   Krish Star   Four     60  female     C
29  30    Reppy Red    Six     79  female     B
26  27          NaN  Three     81     NaN     B
25  26       Crelea  Seven     79    male     B
```

Exercise 6: Exporting Data

1. Question: "Write the code to save the DataFrame with the new 'grade' column to a new CSV file."

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

#Defining the file path in Google Drive
save_path = "/content/drive/My Drive/student_with_grades.csv"

#Saving the DataFrame as a CSV file
df.to_csv(save_path, index=False)

#Printing confirmation
print(f"File saved successfully at: {save_path}")  #Q1

print(df.head())
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
File saved successfully at: /content/drive/My Drive/student_with_grades.csv
   id        name  class  score  gender grade
0   1    John Deo   Four     75  female     B
1   2    Max Ruin  Three     85    male     A
2   3      Arnold  Three     55    male     D
3   4  Krish Star   Four     60  female     C
4   5   John Mike   Four     60  female     C
```

Exercise 7: If finished early try visualising the results:

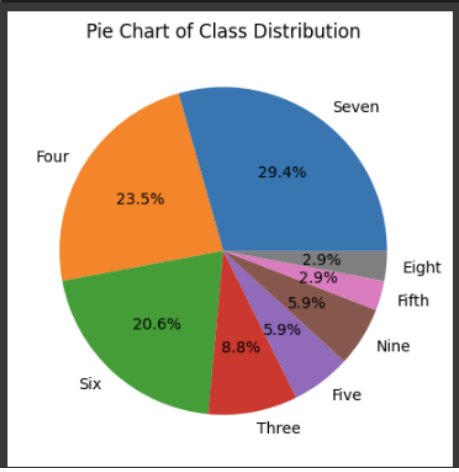
```python
#Importing the necessary library
import matplotlib.pyplot as plt

#Calculating the frequency of each class
class_counts = df['class'].value_counts()

#Creating the pie chart
plt.pie(class_counts,  # Data for the slices (class frequencies)
        labels=class_counts.index,  # Labels for each slice (class names)
        autopct='%1.1f%%')  # Display percentages with one decimal place

#Adding a title to the chart
plt.title("Pie Chart of Class Distribution")

plt.show()
```

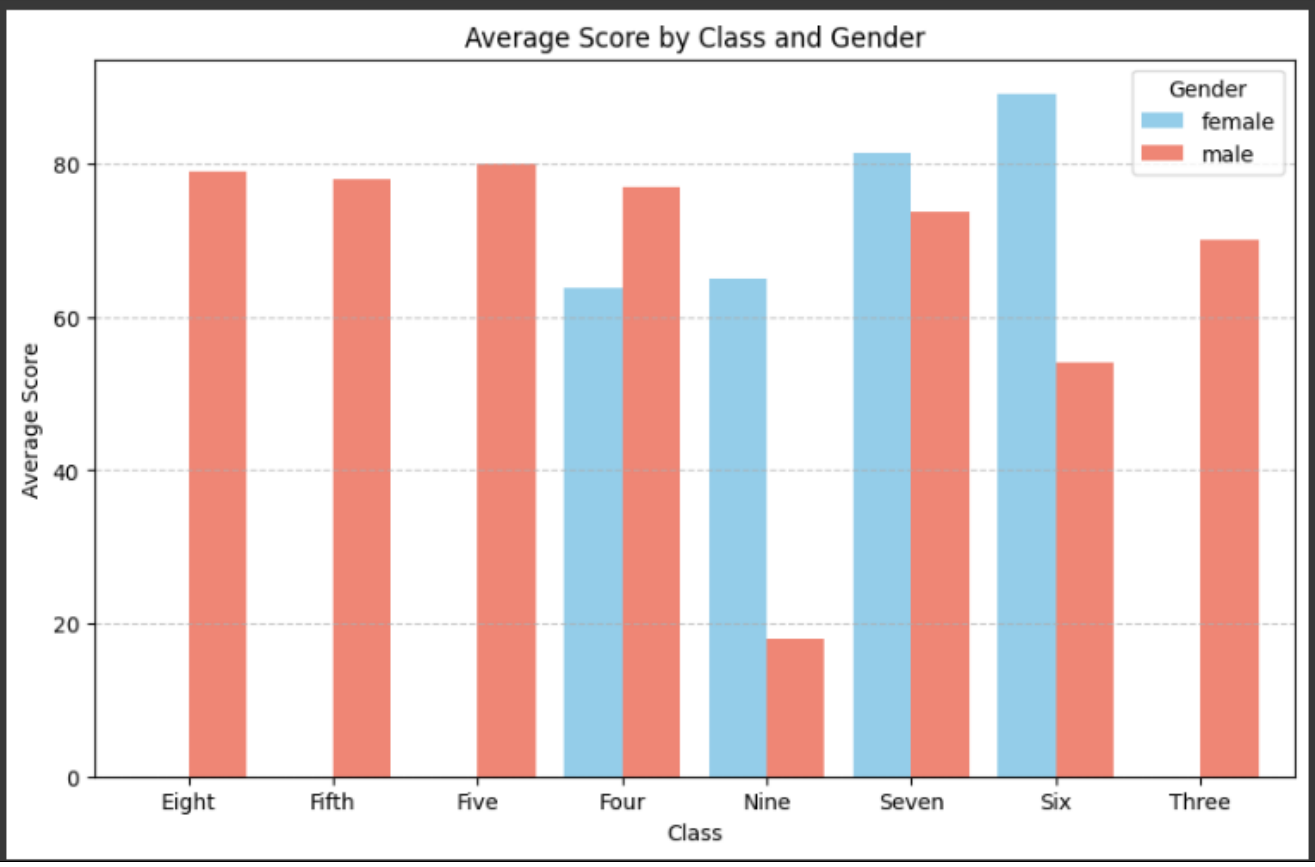


```python
import matplotlib.pyplot as plt

#Plotting the pivot table as a column chart (bar chart)
pivot_table.plot(kind="bar", figsize=(10, 6), width=0.8, color=['skyblue', 'salmon'])

#Customizing the chart
plt.title("Average Score by Class and Gender")
plt.xlabel("Class")
plt.ylabel("Average Score")
plt.xticks(rotation=0)  # Keep class labels readable
plt.legend(title="Gender")
plt.grid(axis="y", linestyle="--", alpha=0.7)


plt.show()
```



Day 4: Task 1

Using the 'GDP (nominal) per Capita.csv' which can be downloaded from the shared Folder, complete the below exercises and paste your input and output. Work individually, but we will work and support each other in the room.

1. Read and save the 'GDP (nominal) per Capita' data to a data frame called "df" in Jyputer notebook
2. Print the first 10 rows
3. Print the last 5 rows
4. Print 'Country/Territory' and 'UN_Region' columns

```
#Importanting the panda library as pd
import pandas as pd
#Reading our data file after uploading it
df = pd.read_csv('GDP (nominal) per Capita.csv')
#Showing the first ten columns
df.head(10)
```

	Unnamed: 0	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	Wor
0	1	Monaco	Europe	0	0	234316	
1	2	Liechtenstein	Europe	0	0	157755	
2	3	Luxembourg	Europe	132372	2023	133590	
3	4	Ireland	Europe	114581	2023	100172	
4	5	Bermuda	Americas	0	0	114090	
5	6	Norway	Europe	101103	2023	89154	
6	7	Switzerland	Europe	98767	2023	91992	
7	8	Singapore	Asia	91100	2023	72794	
8	9	Isle of Man	Europe	0	0	87158	
9	10	Cayman Islands	Americas	0	0	86569	

```
#Importanting the panda library as pd
import pandas as pd
#Reading our data file after uploading it
df = pd.read_csv('GDP (nominal) per Capita.csv')
#Showing the first ten columns
#df.head(10)

#Showing the last 5 columns(No number required as, 5, is the default)
df.tail()
```

	Unnamed: 0	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	W
218	219	Malawi	Africa	496	2023	635	
219	220	South Sudan	Africa	467	2023	1072	
220	221	Sierra Leone	Africa	415	2023	480	
221	222	Afghanistan	Asia	611	2020	369	
222	223	Burundi	Africa	249	2023	222	

```
#Importanting the panda library as pd
import pandas as pd
#Reading our data file after uploading it
df = pd.read_csv('GDP (nominal) per Capita.csv')
#Showing the first ten columns
#df.head(10)

#Showing the last 5 columns(No number required as, 5, is the default)
#df.tail()

#Bringing up only the two required columns
df[['Country/Territory','UN_Region']]
```

	Country/Territory	UN_Region
0	Monaco	Europe
1	Liechtenstein	Europe
2	Luxembourg	Europe
3	Ireland	Europe
4	Bermuda	Americas
...	...	...
218	Malawi	Africa
219	South Sudan	Africa
220	Sierra Leone	Africa
221	Afghanistan	Asia
222	Burundi	Africa

223 rows × 2 columns

Day 4: Task 2

Back with 'GDP (nominal) per Capita'. As a group, import and work your way through the Day_4_Python_Activity.ipynb notebook which can be found on the shared Folder. There are questions to answer, but also opportunities to have fun with the data – paste your input and output below.

Once complete, and again as a group, work with some more data and have some fun – there is no set agenda for this section, other than to embed the skills developed this week. Paste your input and output below and upon return we'll discuss progress made.

[Additional data found here.]

I am attaching some screenshots of the work I have done. I tried to explore the data more and do more than what was asked. I will also attach the file of my notebook since the data is too much to post everything on here.

https://colab.research.google.com/drive/1ic0AMZNIKhZXCOtuFw6B1xznn8VU9_9A?usp=sharing

```
[5] #Dataframe information
    df.info()
```

Show hidden output

```
[158] #Display the first few rows of the DataFrame
      df.head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year
1	Monaco	Europe	234316.5	NaN	234316.0	2021.0
2	Liechtenstein	Europe	163507.5	NaN	157755.0	2020.0
3	Luxembourg	Europe	132372.0	2023.0	133590.0	2021.0
4	Ireland	Europe	114581.0	2023.0	100172.0	2021.0
5	Bermuda	Americas	113371.5	NaN	114090.0	2021.0

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
[161] #Get statistics for the DataFrame
      df.describe()
```

```
#Counting the occurrences of each unique value in the 'UN_Region' column
df['UN_Region'].value_counts()
```

UN_Region	count
Africa	55
Asia	51
Europe	48
Americas	48
Oceania	20
World	1

```
#Filter the DataFrame to select rows where the 'UN_Region' is 'Europe'
df[df['UN_Region'] == 'Europe']
```

Show hidden output

```
[102] #Filter the DataFrame to select rows where the 'UN_Region' is 'Americas'
      df[df['UN_Region'] == 'Americas']
```

Show hidden output

```
[103] #Filter the DataFrame to select rows where the 'UN_Region' is 'Africa'
      df[df['UN_Region'] == 'Africa']
```

Show hidden output

```
[104] #Filter the DataFrame to select rows where the 'UN_Region' is 'Asia'
      df[df['UN_Region'] == 'Asia']
```

Show hidden output

```
[105] #Filter the DataFrame to select rows where the 'UN_Region' is 'Oceania'
      df[df['UN_Region'] == 'Oceania']
```

Show hidden output

```
[106] # number of countries per region
```

```
[107] region_counts = df['UN_Region'].value_counts().sort_values(ascending=False)
      print(region_counts)
```

Show hidden output

```
[106] # number of countries per region

region_counts = df['UN_Region'].value_counts().sort_values(ascending=False)
print(region_counts)
```

```
UN_Region
Africa      55
Asia        51
Europe      48
Americas    48
Oceania     20
World        1
Name: count, dtype: int64
```

```
[108] #What is European Union[n 1]?
```

```
[164] df[df['Country/Territory'] == 'European Union[n 1]']
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UI
36	European Union[n 1]	Europe	39940.0	2023.0	38411.0	2021.0	

```
[110] # This  represents the aggregate GDP data for the European Union as a whole.
eu_data = df[df['Country/Territory'] == 'European Union[n 1]']
print(eu_data)
```

```
      Country/Territory UN_Region  IMF_Estimate  IMF_Year  WorldBank_Estimate  \
36  European Union[n 1]    Europe       39940.0    2023.0             38411.0

    WorldBank_Year  UN_Estimate UN_Year
36          2021.0      31875.0    2021
```

```
[111] # Countries in Europe below avarege

# Calculating the average GDP for Europe
europe_avg_gdp = df[df['UN_Region'] == 'Europe']['WorldBank_Estimate'].mean()
print('The average european GDP is', europe_avg_gdp,'!')
```

```
The average european GDP is 45193.6875 !
```

```
[167] # Filtering countries in Europe with GDP below average
eu_countries_below_avg = df[(df['UN_Region'] == 'Europe') & (df['WorldBank_Estimate'] < europe_avg_g
# Removing non-relevant columns
eu_countries_below_avg = eu_countries_below_avg.drop(columns=['IMF_Year', 'UN_Region', 'WorldBank_Ye
print(eu_countries_below_avg)
```

```
          Country/Territory  IMF_Estimate  WorldBank_Estimate  UN_Estimate
34                   France       44408.0             43659.0      44229.0
35                  Andorra       44387.0             42137.0      42066.0
36      European Union[n 1]       39940.0             38411.0      31875.0
40                    Malta       36989.0             33487.0      33642.0
41                    Italy       36812.0             35658.0      35579.0
51                 Slovenia       32214.0             29291.0      29135.0
52           Czech Republic       31368.0             26821.0      26809.0
53                    Spain       31223.0             30104.0      30058.0
54                  Estonia       31209.0             27944.0      27991.0
57                Lithuania       28094.0             23723.0      23844.0
59                 Portugal       26012.0             24568.0      24651.0
60                   Latvia       25136.0             21148.0      21267.0
62                 Slovakia       23457.0             21392.0      21390.0
63                   Greece       22595.0             20193.0      20571.0
70                  Croatia       20537.0             17685.0      16983.0
72                   Poland       19912.0             18000.0      17736.0
75                  Hungary       19385.0             18728.0      18728.0
78                  Romania       18530.0             14858.0      14698.0
87                 Bulgaria       14893.0             12222.0      12207.0
90                   Russia       14403.0             12195.0      12259.0
103               Montenegro       11289.0              9466.0       9252.0
106                   Serbia       10849.0              9230.0       8643.0
112  Bosnia and Herzegovina        8223.0              7143.0       7143.0
115                  Belarus        7944.0              7302.0       7121.0
118          North Macedonia        7384.0              6695.0       6600.0
120                  Albania        7058.0              6493.0       6396.0
127                  Moldova        6342.0              5231.0       4468.0
133                   Kosovo        5641.0              5270.0       5663.0
143                  Ukraine        4654.0              4836.0       4596.0
```

```
#Calculating the difference between the average GDP for Europe and the GDP of each country below ave
eu_countries_below_avg['GDP_Difference'] = europe_avg_gdp - eu_countries_below_avg['WorldBank_Estima
print(eu_countries_below_avg[['Country/Territory', 'WorldBank_Estimate', 'GDP_Difference']])
```

```
          Country/Territory  WorldBank_Estimate  GDP_Difference
34                   France             43659.0       1534.6875
35                  Andorra             42137.0       3056.6875
36      European Union[n 1]             38411.0       6782.6875
40                    Malta             33487.0      11706.6875
41                    Italy             35658.0       9535.6875
51                 Slovenia             29291.0      15902.6875
52           Czech Republic             26821.0      18372.6875
53                    Spain             30104.0      15089.6875
54                  Estonia             27944.0      17249.6875
57                Lithuania             23723.0      21470.6875
59                 Portugal             24568.0      20625.6875
60                   Latvia             21148.0      24045.6875
62                 Slovakia             21392.0      23801.6875
63                   Greece             20193.0      25000.6875
70                  Croatia             17685.0      27508.6875
72                   Poland             18000.0      27193.6875
75                  Hungary             18728.0      26465.6875
78                  Romania             14858.0      30335.6875
87                 Bulgaria             12222.0      32971.6875
90                   Russia             12195.0      32998.6875
103               Montenegro              9466.0      35727.6875
106                   Serbia              9230.0      35963.6875
112  Bosnia and Herzegovina               7143.0      38050.6875
115                  Belarus              7302.0      37891.6875
118          North Macedonia              6695.0      38498.6875
120                  Albania              6493.0      38700.6875
127                  Moldova              5231.0      39962.6875
133                   Kosovo              5270.0      39923.6875
143                  Ukraine              4836.0      40357.6875
```

```
[119] ## Which countries in Europe has higher GDP than UK?
```

```
# Get the UK's GDP
uk_gdp = df[df['Country/Territory'] == 'United Kingdom']['WorldBank_Estimate'].values[0]
print('The GDP of the UK is', int(uk_gdp),'.')
```

```
The GDP of the UK is 46510 .
```

```
[170] # Filter countries in Europe with higher GDP than UK
higher_gdp_countries = df[(df['UN_Region'] == 'Europe') & (df['WorldBank_Estimate'] > uk_gdp)]
print(higher_gdp_countries[['Country/Territory', 'WorldBank_Estimate']])
```

```
   Country/Territory  WorldBank_Estimate
1             Monaco            234316.0
2      Liechtenstein            157755.0
3         Luxembourg            133590.0
4            Ireland            100172.0
6             Norway             89154.0
7        Switzerland             91992.0
9        Isle of Man             87158.0
13           Iceland             68728.0
14   Channel Islands             75153.0
15     Faroe Islands             69010.0
16           Denmark             68008.0
18       Netherlands             57768.0
20           Austria             53638.0
22            Sweden             61029.0
23           Finland             53655.0
24           Belgium             51247.0
28           Germany             51204.0
```

```
[122] #Get GDP values for EU, UK, and global average
eu_gdp = df[df['Country/Territory'] == 'European Union[n 1]']['WorldBank_Estimate'].values[0]
uk_gdp = df[df['Country/Territory'] == 'United Kingdom']['WorldBank_Estimate'].values[0]
global_avg_gdp = df['WorldBank_Estimate'].mean()

print("European Union GDP:", eu_gdp)
print("United Kingdom GDP:", uk_gdp)
print("Global Average GDP:", global_avg_gdp)
```

```
European Union GDP: 38411.0
United Kingdom GDP: 46510.0
Global Average GDP: 19540.805555555555
```

Course Notes

It is recommended to take notes from the course, use the space below to do so, or use the revision guide shared with the class:

We have included a range of additional links to further resources and information that you may find useful, these can be found within your revision guide.

END OF WORKBOOK

Please check through your work thoroughly before submitting and update the table of contents if required.

Please send your completed work booklet to your trainer.