Data Technician

Name:

Course Date:

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

```
for g in range(1, 101):

    if g % 3 == 0:
        print("fizzbuzz")
    if g % 5 == 0:
        print("fizzbuzz")
    elif g % 3 == 0:
        print("fizz")
    elif g % 5 == 0:
        print("buzz")
    else:
        print(g)
```

```
1
2
fizzbuzz
fizz
4
fizzbuzz
fizzbuzz
fizz
7
```

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."

```
import pandas as pd

df_courses = pd.read_csv('student.csv')
df_courses.head()
df_courses.info()
df_courses.describe()
```

```
<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
```

	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'"

```
df_courses["name"]
df_courses[["name", "mark"]]
df_courses.iloc[:3]
df_courses[df_courses["class"] == "Four"]
```

	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."

```
df_courses["passed"] = df_courses["score"] >= 60
df_courses.rename(columns={"mark": "score"}, inplace=True)
df_courses.drop(columns=["passed"], inplace=True)
df_courses[["name", "score"]]
```

	name	score
0	John Deo	75
1	Max Ruin	85
2	Arnold	55
3	Krish Star	60
4	John Mike	60
5	Alex John	55
6	My John Rob	78
7	Asruid	85
8	Tes Qry	78
9	Big John	55

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."

1)

```
df_courses.groupby("class")["mark"].mean()
```

class	mark
Eight	79.000000
Fifth	78.000000
Five	80.000000
Four	68.750000
Nine	41.500000
Seven	77.600000
Six	82.571429
Three	73.666667

dtype: float64

2)

```
df_courses.groupby("class")["name"].count()
```

class	name
Eight	1
Fifth	1
Five	2
Four	8
Nine	2
Seven	10
Six	7
Three	2

dtype: int64

3)

```
df_courses.groupby("gender")["mark"].mean()
```

gender	mark
female	77.312500
male	71.588235

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."

1)

```
pd.pivot_table(df_courses, values="mark", index="class", columns="gender", aggfunc="mean")
```

1 to 8 of 8 entries Filter

class	female	male
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

Show 25 per page

2)

```
[30]
✓ 0s
def assign_grade(mark):
    if mark >= 85:
        return "A"
    elif mark >= 70:
        return "B"
    elif mark >= 60:
        return "C"
    else:
        return "D"

df_courses["grade"] = df_courses["mark"].apply(assign_grade)

df_courses[["name", "mark","grade"]]
```

	name	mark	grade
0	John Deo	75	B
1	Max Ruin	85	A
2	Arnold	55	D
3	Krish Star	60	C
4	John Mike	60	C
5	Alex John	55	D
6	My John Rob	78	B

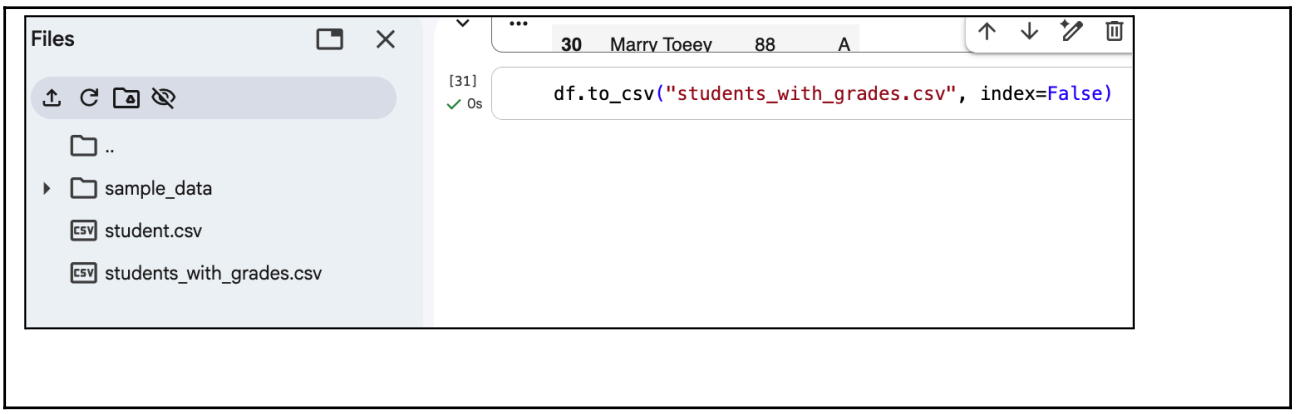
3)

```
df_courses.sort_values(by="mark", ascending=False)
```

	id	name	class	mark	gender	grade
32	33	Kenn Rein	Six	96	female	A
11	12	Recky	Six	94	female	A
31	32	Binn Rott	Seven	90	female	A
10	11	Ronald	Six	89	female	A
30	31	Marry Toeey	Four	88	male	A
34	35	Rows Noump	Six	88	female	A
24	25	Giff Tow	Seven	88	male	A
14	15	Tade Row	NaN	88	male	A
15	16	Gimmy	Four	88	male	A

Exercise 6: Exporting Data

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



Exercise 7: If finished early try visualising the results

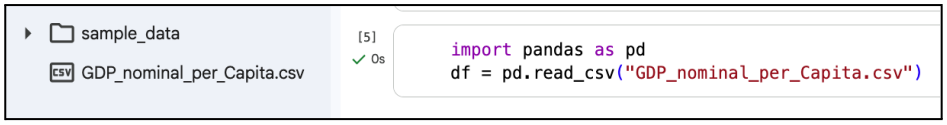
id	name	class	mark	gender	grade
1	John Deo	Four	75	female	B
2	Max Ruin	Three	85	male	A
3	Arnold	Three	55	male	D
4	Krish Star	Four	60	female	C
5	John Mike	Four	60	female	C
6	Alex John	Four	55	male	D
7	My John Rob	Fifth	78	male	B
8	Asruid	Five	85	male	A
9	Tes Qry	Six	78		B
10	Big John	Four	55	female	D

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.

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

1)



2)

```
print("First 10 rows:")
display(df.head(10))
```

```
First 10 rows:
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
1	Monaco	Europe	0	0	234316	2021	234317	2021
2	Liechtenstein	Europe	0	0	157755	2020	169260	2021
3	Luxembourg	Europe	132372	2023	133590	2021	133745	2021
4	Ireland	Europe	114581	2023	100172	2021	101109	2021
5	Bermuda	Americas	0	0	114090	2021	112653	2021
6	Norway	Europe	101103	2023	89154	2021	89242	2021
7	Switzerland	Europe	98767	2023	91992	2021	93525	2021
8	Singapore	Asia	91100	2023	72794	2021	66822	2021
9	Isle of Man	Europe	0	0	87158	2019	0	0
10	Cayman Islands	Americas	0	0	86569	2021	85250	2021

3)

```
print("\nLast 5 rows:")
display(df.tail(5))
```

```
Last 5 rows:
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
219	Malawi	Africa	496	2023	635	2021	613	2021
220	South Sudan	Africa	467	2023	1072	2015	400	2021
221	Sierra Leone	Africa	415	2023	480	2021	505	2021
222	Afghanistan	Asia	611	2020	369	2021	373	2021
223	Burundi	Africa	249	2023	222	2021	311	2021

4)

```
print("\nSelected columns:")
display(df[['Country/Territory', 'UN_Region']])
```

```
Selected columns:
```

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

223 rows × 2 columns

Day 4: Task 2

1) 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.

2) 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.

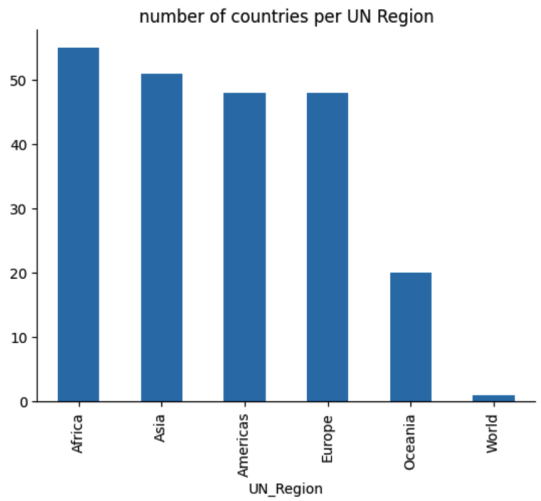
```
[15] ✓  # number of countries per region

[20] ✓ 0s  display(df['UN_Region'].value_counts())
```

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

dtype: int64

```
from matplotlib import pyplot as plt
df['UN_Region'].value_counts().plot(kind='bar', title='number of countries per UN Region')
plt.gca().spines[['top', 'right',]].set_visible(False)
```




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

print("European Union[n 1]")
print(df.loc[df['Country/Territory'] == "European Union[n 1]"])

# Or display the first row as a DataFrame (keeps table format)
print(df.iloc[[0]])
```

```
European Union[n 1]
    Unnamed: 0    Country/Territory UN_Region  IMF_Estimate  IMF_Year  \
35          36  European Union[n 1]    Europe         39940      2023
```

```
# Countries in Europe below avarege

print("Countries in Europe with all estimates below their global averages:")
europe_df = europe_df.sort_values(by="WorldBank_Estimate", ascending=True)
display(european_countries_below_average)
```

Countries in Europe with all estimates below their global averages:

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
87	Bulgaria	Europe	14893	2023	12222	2021	12207	2021
90	Russia	Europe	14403	2023	12195	2021	12259	2021
103	Montenegro	Europe	11289	2023	9466	2021	9252	2021
106	Serbia	Europe	10849	2023	9230	2021	8643	2021
112	Bosnia and Herzegovina	Europe	8223	2023	7143	2021	7143	2021
115	Belarus	Europe	7944	2023	7302	2021	7121	2021
118	North Macedonia	Europe	7384	2023	6695	2021	6600	2021
120	Albania	Europe	7058	2023	6493	2021	6396	2021
127	Moldova	Europe	6342	2023	5231	2021	4468	2021
133	Kosovo	Europe	5641	2023	5270	2021	5663	2021
143	Ukraine	Europe	4654	2023	4836	2021	4596	2021

```
# Countries with GDP below average in Europe
# Sort the Countries in Europe by GDP
europe_df = df[df["UN_Region"] == "Europe"].copy()
europe_df = europe_df.sort_values(by="WorldBank_Estimate", ascending=True)
# Extract the mean GDP as a single number
mean_gdp_europe = europe_df["WorldBank_Estimate"].mean().round(2)
# Filter countries with GDP < average (for countries below average)
df_filtered_below = europe_df[europe_df["WorldBank_Estimate"] < mean_gdp_europe]
print(f"Average WorldBank Estimate for Europe: {mean_gdp_europe}")
print("Countries in Europe with WorldBank Estimate below average:")
display(df_filtered_below[["Country/Territory", "UN_Region", "WorldBank_Estimate"]])
```

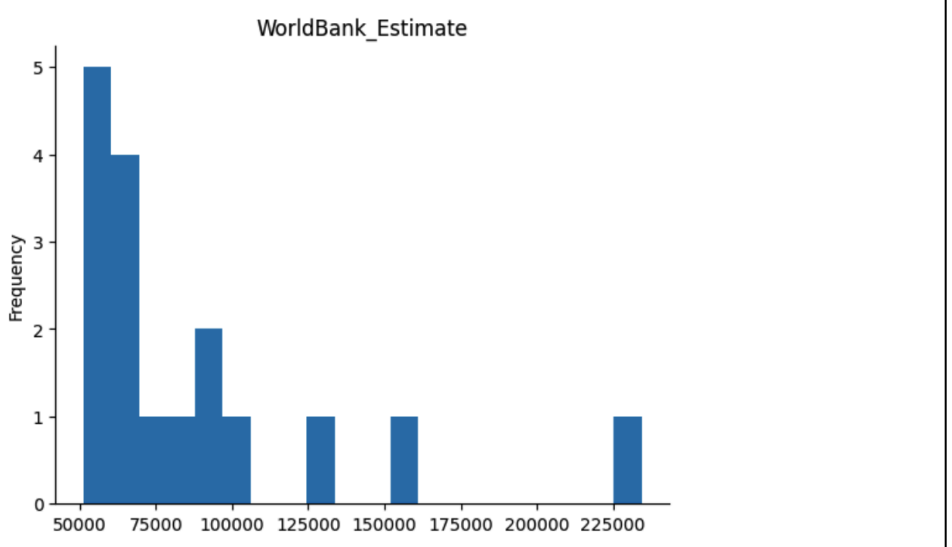
Average WorldBank Estimate for Europe: 45193.69
Countries in Europe with WorldBank Estimate below average:

	Country/Territory	UN_Region	WorldBank_Estimate
143	Ukraine	Europe	4836
127	Moldova	Europe	5231
133	Kosovo	Europe	5270
120	Albania	Europe	6493
118	North Macedonia	Europe	6695
112	Bosnia and Herzegovina	Europe	7143
115	Belarus	Europe	7302
106	Serbia	Europe	9230
103	Montenegro	Europe	9466
90	Russia	Europe	12195
87	Bulgaria	Europe	12222
78	Romania	Europe	14858

```python
# Countries in Europe with higher GDP than UK
# Filter to Europe
df = df[df["UN_Region"] == "Europe"]
# Sort by GDP
df = df.sort_values(by="WorldBank_Estimate", ascending=False)
# Find the GDP of the UK
UK_GDP = df.loc[df["Country/Territory"] == "United Kingdom", "WorldBank_Estimate"].iloc[0]
# Filter countries with GDP >= UK
df_filtered = df[df["WorldBank_Estimate"] > UK_GDP]
# Show results
display(df_filtered[["Country/Territory", "UN_Region", "WorldBank_Estimate"]])
```

	Country/Territory	UN_Region	WorldBank_Estimate
1	Monaco	Europe	234316
2	Liechtenstein	Europe	157755
3	Luxembourg	Europe	133590
4	Ireland	Europe	100172
7	Switzerland	Europe	91992
6	Norway	Europe	89154
9	Isle of Man	Europe	87158
14	Channel Islands	Europe	75153
15	Faroe Islands	Europe	69010
13	Iceland	Europe	68728
16	Denmark	Europe	68008
22	Sweden	Europe	61029
18	Netherlands	Europe	57768

```python
from matplotlib import pyplot as plt
_df_24['WorldBank_Estimate'].plot(kind='hist', bins=20, title='WorldBank_Estimate')
plt.gca().spines[['top', 'right',]].set_visible(False)
```



```python
# Calculate global IMF average
global_imf_avg = df['IMF_Estimate'].mean()
df_sorted = df_filtered_nonzero.sort_values('IMF_Estimate', ascending=False)
print("Global IMF average:", global_imf_avg)

# Filter countries below global IMF average
below_imf_avg = df[df['IMF_Estimate'] < global_imf_avg]

# Display the result
print("Countries below global IMF estimate:")
display(below_imf_avg[['Country/Territory', 'IMF_Estimate']])
```

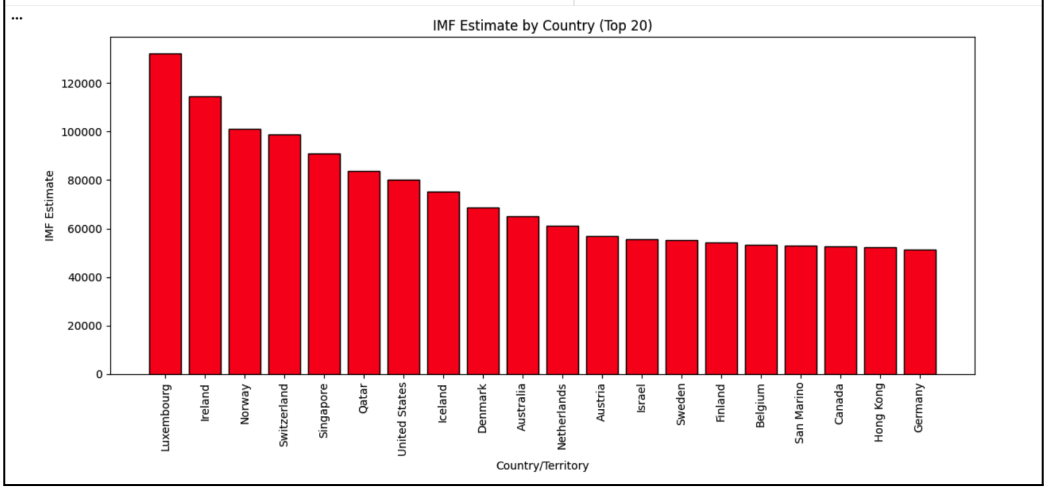
```
Global IMF average: 15351.632286995517
Countries below global IMF estimate:
```

	Country/Territory	IMF_Estimate
0	Monaco	0
1	Liechtenstein	0
4	Bermuda	0
8	Isle of Man	0
9	Cayman Islands	0
...	...	...
218	Malawi	496
219	South Sudan	467
220	Sierra Leone	415
221	Afghanistan	611
222	Burundi	249

159 rows × 2 columns

```python
df_filtered_nonzero = df[df['IMF_Estimate'] > 0].sort_values(by='IMF_Estimate', ascending=False).head(20)
# Sort by IMF_Estimate for better visualization
df_sorted = df_filtered_nonzero.sort_values('IMF_Estimate', ascending=False)
# Create a bar plot with countries on x-axis
plt.figure(figsize=(12, 6))
plt.bar(df_sorted['Country/Territory'], df_sorted['IMF_Estimate'], color='red', edgecolor='black')
plt.title('IMF Estimate by Country (Top 20)')
plt.xlabel('Country/Territory')
plt.ylabel('IMF Estimate')
plt.xticks(rotation=90)  # rotate country names for readability

# Remove top and right spines
plt.tight_layout()
plt.show()
```



IMF estimate 0 values

```
df[df['IMF_Estimate'] == 0]
df.loc[df['IMF_Estimate'] == 0, ['Country/Territory', 'IMF_Estimate']]
print("Countries with IMF estimate 0:")
display(df[df['IMF_Estimate'] == 0][['Country/Territory', 'IMF_Estimate']])
```

```
Countries with IMF estimate 0:
```

	Country/Territory	IMF_Estimate
0	Monaco	0
1	Liechtenstein	0
4	Bermuda	0
8	Isle of Man	0
9	Cayman Islands	0
13	Channel Islands	0
14	Faroe Islands	0
18	Greenland	0
30	British Virgin Islands	0
36	US Virgin Islands	0
38	New Caledonia	0
41	Guam	0
57	Sint Maarten (Dutch part)	0

Which country has highest UN Estimate?

```
df = df.sort_values("UN_Estimate", ascending=False)
print("The Country with the highest GDP based on UN estimate:")
print(df.head(1)[["Country/Territory", "UN_Estimate"]])
```

```
The Country with the highest GDP based on UN estimate:
  Country/Territory  UN_Estimate
1            Monaco       234317
```

Which country has highest Worlbank Estimate?

```
df = df.sort_values(("WorldBank_Estimate"), ascending= False)
print("The Country with the highest GDP based on WorldBank estimate:")
print(df.head(1)[["Country/Territory", "WorldBank_Estimate"]])
```

```
The Country with the highest GDP based on WorldBank estimate:
  Country/Territory  WorldBank_Estimate
1            Monaco              234316
```

Which country has highest IMF Estimate?

```
highest_imf_row = df.loc[df['IMF_Estimate'].idxmax()]

# Display result
print("Country with highest IMF Estimate:")
display(highest_imf_row[['Country/Territory', 'IMF_Estimate']])
```

```
Country with highest IMF Estimate:
```

	3
Country/Territory	Luxembourg
IMF_Estimate	132372

```python
# 1. Calculate global averages
global_avgs = df[['IMF_Estimate', 'UN_Estimate', 'WorldBank_Estimate']].mean()
europe_df = df[df['UN_Region'] == 'Europe']

below_global_avg = europe_df[
    (europe_df['IMF_Estimate'] < global_avgs['IMF_Estimate']) &
    (europe_df['UN_Estimate'] < global_avgs['UN_Estimate']) &
    (europe_df['WorldBank_Estimate'] < global_avgs['WorldBank_Estimate'])]

print("Countries below global averages:")
display(below_global_avg[['Country/Territory', 'IMF_Estimate', 'UN_Estimate', 'WorldBank_Estimate']])
```

```
Countries below global averages:
```

	Country/Territory	IMF_Estimate	UN_Estimate	WorldBank_Estimate
86	Bulgaria	14893	12207	12222
89	Russia	14403	12259	12195
102	Montenegro	11289	9252	9466
105	Serbia	10849	8643	9230
111	Bosnia and Herzegovina	8223	7143	7143
114	Belarus	7944	7121	7302
117	North Macedonia	7384	6600	6695
119	Albania	7058	6396	6493

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.