Just IT

B2W group

Apprenticeships | Training | Recruitment

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	`# Loop through numbers from 1 to 100`
	`for n in range(1, 101):`
	`  # Check if divisible by both 3 and 5 first`
	`  if n % 3 == 0 and n % 5 == 0:`
	`  print("Fizzbuzz")`
	`  # If divisible by only 3`
	`  elif n % 3 == 0:`
	`  print("Fizz")`
	`  # If divisible by only 5`
	`  elif n % 5 == 0:`
	`  print("Buzz")`
	`  # If not divisible by 3 or 5`
	`  else:`
	`  print(n)`

1

2

Fizz

4

Buzz

Fizz

7

8

Fizz

Buzz

11

Fizz

13

14

Fizzbuzz

16

17

Fizz

19

Buzz

Fizz

22

23

Fizz

Buzz

26

Fizz
28
29
Fizzbuzz
31
32
Fizz
34
Buzz
Fizz
37
38
Fizz
Buzz
41
Fizz
43
44
Fizzbuzz
46
47
Fizz
49
Buzz
Fizz
52
53

Fizz

Buzz

56

Fizz

58

59

Fizzbuzz

61

62

Fizz

64

Buzz

Fizz

67

68

Fizz

Buzz

71

Fizz

73

74

Fizzbuzz

76

77

Fizz

79

Buzz

Fizz

82

83

Fizz

Buzz

86

Fizz

88

89

Fizzbuzz

91

92

Fizz

94

Buzz

Fizz

97

98

Fizz

Buzz

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
# Read the CSV file into a DataFrame
df = pd.read_csv('student.csv')

# Display the first 5 rows to get a quick look at the data
print(df.head())

# Show info about the DataFrame: column names, data types, non-null counts, etc.
print(df.info())

# Get basic statistical summary (mean, std, min, max, etc.) for numerical columns
print(df.describe())
```

```python
import pandas as pd

# Read the CSV file into a DataFrame
df = pd.read_csv('student.csv')
```

```python
# Display the first 5 rows to get a quick look at the data
print(df.head())
```

```
   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
3   4  Krish Star   Four    60  female
4   5   John Mike   Four    60  female
```

```python
# Show info about the DataFrame: column names, data types, non-null counts, etc.
print(df.info())
```

```
<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
None
```

```python
# Get basic statistical summary (mean, std, min, max, etc.) for numerical columns
print(df.describe())
```

```
              id       mark
count  35.000000  35.000000
mean   18.000000  74.657143
std    10.246951  16.401117
min     1.000000  18.000000
```

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'."

```
# Select only the 'name' column
print(df['name'])
```

```python
# Select only the 'name' column
print(df['name'])
```

```
0         John Deo
1        Max Ruin
2          Arnold
3      Krish Star
4       John Mike
5       Alex John
6     My John Rob
7           Asruid
8         Tes Qry
9        Big John
10         Ronald
11          Recky
12            Kty
13           Bigy
14       Tade Row
15          Gimmy
16          Tumyu
17          Honny
18          Tinny
19         Jackly
20     Babby John
21         Reggid
22          Herod
23      Tiddy Now
24       Giff Tow
25         Crelea
26            NaN
27      Rojj Base
28    Tess Played
29      Reppy Red
30    Marry Toeey
31      Binn Rott
32      Kenn Rein
33       Gain Toe
34     Rows Noump
Name: name, dtype: object
```

```
# Select both 'name' and 'mark' columns
print(df[['name', 'mark']])
```

```python
# Select both 'name' and 'mark' columns
print(df[['name', 'mark']])
```

```
            name  mark
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
10        Ronald    89
11         Recky    94
12           Kty    88
13          Bigy    88
14      Tade Row    88
15         Gimmy    88
16         Tumyu    54
17         Honny    75
18         Tinny    18
19        Jackly    65
20    Babby John    69
21        Reggid    55
22         Herod    79
23     Tiddy Now    78
24      Giff Tow    88
25        Crelea    79
26           NaN    81
27     Rojj Base    86
28  Tess Played    55
29     Reppy Red    79
30  Marry Toeey    88
31     Binn Rott    90
32     Kenn Rein    96
33      Gain Toe    69
34    Rows Noump    88
```

# Use .head(3) to get the first 3 rows
print(df.head(3))

```
[9]  # Use .head(3) to get the first 3 rows
     print(df.head(3))
```

```
    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
```

# Select rows where 'class' is 'Four'
class_four_students = df[df["class"] == "Four"]
print(class_four_rows)

```
0s  # Select rows where the 'class' column is 'Four'
    class_four_rows = df[df['class'] == 'Four']
    print(class_four_rows)
```

```
    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
# Add 'passed' column (True if mark >= 60)
df['passed'] = df['mark'] >= 60

# Show the updated DataFrame
print(df)
```

```
    id          name  class  mark  gender  passed
0    1      John Deo   Four    75  female    True
1    2      Max Ruin  Three    85    male    True
2    3        Arnold  Three    55    male   False
3    4    Krish Star   Four    60  female    True
4    5     John Mike   Four    60  female    True
5    6     Alex John   Four    55    male   False
6    7  My John Rob  Fifth    78    male    True
7    8        Asruid   Five    85    male    True
8    9       Tes Qry    Six    78     NaN    True
9   10      Big John   Four    55  female   False
10  11        Ronald    Six    89  female    True
11  12         Recky    Six    94  female    True
12  13           Kty  Seven    88  female    True
13  14         Bigy  Seven    88  female    True
14  15      Tade Row    NaN    88    male    True
15  16         Gimmy   Four    88    male    True
16  17         Tumyu    Six    54    male   False
17  18         Honny   Five    75    male    True
18  19         Tinny   Nine    18    male   False
19  20        Jackly   Nine    65  female    True
20  21    Babby John   Four    69  female    True
21  22        Reggid  Seven    55  female   False
22  23         Herod  Eight    79    male    True
23  24     Tiddy Now  Seven    78    male    True
24  25      Giff Tow  Seven    88    male    True
25  26        Crelea  Seven    79    male    True
26  27           NaN  Three    81     NaN    True
27  28     Rojj Base  Seven    86  female    True
28  29  Tess Played  Seven    55    male   False
29  30     Reppy Red    Six    79  female    True
30  31  Marry Toeey   Four    88    male    True
31  32     Binn Rott  Seven    90  female    True
32  33     Kenn Rein    Six    96  female    True
33  34      Gain Toe  Seven    69    male    True
34  35    Rows Noump    Six    88  female    True
```

```python
# Rename 'mark' to 'score'
df = df.rename(columns={'mark': 'score'})

# Show the updated DataFrame
print(df)
```

```
    id          name  class  score  gender  passed
0    1      John Deo   Four     75  female    True
1    2      Max Ruin  Three     85    male    True
2    3        Arnold  Three     55    male   False
3    4   Krish Star    Four     60  female    True
4    5     John Mike   Four     60  female    True
5    6     Alex John   Four     55    male   False
6    7  My John Rob  Fifth     78    male    True
7    8       Asruid   Five     85    male    True
8    9      Tes Qry    Six     78     NaN    True
9   10     Big John   Four     55  female   False
10  11       Ronald    Six     89  female    True
11  12        Recky    Six     94  female    True
12  13          Kty  Seven     88  female    True
13  14        Bigy   Seven     88  female    True
14  15     Tade Row    NaN     88    male    True
15  16        Gimmy   Four     88    male    True
16  17        Tumyu    Six     54    male   False
17  18        Honny   Five     75    male    True
18  19        Tinny   Nine     18    male   False
19  20       Jackly   Nine     65  female    True
20  21   Babby John   Four     69  female    True
21  22       Reggid  Seven     55  female   False
22  23        Herod  Eight     79    male    True
23  24    Tiddy Now  Seven     78    male    True
24  25     Giff Tow  Seven     88    male    True
25  26       Crelea  Seven     79    male    True
26  27          NaN  Three     81     NaN    True
27  28    Rojj Base  Seven     86  female    True
28  29  Tess Played  Seven     55    male   False
29  30    Reppy Red    Six     79  female    True
30  31  Marry Toeey   Four     88    male    True
31  32    Binn Rott  Seven     90  female    True
32  33    Kenn Rein    Six     96  female    True
33  34     Gain Toe  Seven     69    male    True
34  35   Rows Noump    Six     88  female    True
```

```python
# Drop 'passed' column
df = df.drop('passed', axis=1)

# Show the updated DataFrame
print(df)
```

```
    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
5    6    Alex John   Four     55    male
6    7  My John Rob  Fifth     78    male
7    8       Asruid   Five     85    male
8    9      Tes Qry    Six     78     NaN
9   10      Big John   Four     55  female
10  11        Ronald    Six     89  female
11  12         Recky    Six     94  female
12  13           Kty  Seven     88  female
13  14         Bigy   Seven     88  female
14  15     Tade Row     NaN     88    male
15  16         Gimmy   Four     88    male
16  17         Tumyu    Six     54    male
17  18         Honny   Five     75    male
18  19         Tinny   Nine     18    male
19  20        Jackly   Nine     65  female
20  21   Babby John    Four     69  female
21  22        Reggid  Seven     55  female
22  23         Herod  Eight     79    male
23  24    Tiddy Now   Seven     78    male
24  25     Giff Tow   Seven     88    male
25  26        Crelea  Seven     79    male
26  27           NaN  Three     81     NaN
27  28    Rojj Base   Seven     86  female
28  29  Tess Played   Seven     55    male
29  30    Reppy Red     Six     79  female
30  31  Marry Toeey   Four     88    male
31  32    Binn Rott   Seven     90  female
32  33    Kenn Rein     Six     96  female
33  34     Gain Toe   Seven     69    male
34  35   Rows Noump     Six     88  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**."-mark as score**
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."

```
[88] # Group by 'class' and calculate the average mark for each class
     mean_mark_by_class = df.groupby('class')['score'].mean()
     print(mean_mark_by_class)
```

```
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
```

```
# Count how many students are in each class
student_count_by_class = df['class'].value_counts()
print(student_count_by_class)
```

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

```
# Group by 'gender' and calculate average score
average_score_by_gender = df.groupby('gender')['score'].mean()
print(average_score_by_gender)
```

```
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."

Create the pivot table

pivot_table = df.pivot_table(values="score", index="class", columns="gender", aggfunc="mean")

print(pivot_table)

```
[95] # Create a pivot table: rows = class, columns = gender, values = score
     pivot_table = df.pivot_table(values='score', index='class', columns='gender', aggfunc='mean')
     print(pivot_table)
```

```
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
```

```python
# Add a 'grade' column based on the score ranges
def get_grade(score):
    if score >= 85:
        return 'A'
    elif score >= 70:
        return 'B'
    elif score >= 60:
        return 'C'
    else:
        return 'D'

df['grade'] = df['score'].apply(get_grade)

# Show the updated DataFrame
print(df)
```

```
    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
5    6     Alex John   Four     55    male     D
6    7  My John Rob  Fifth     78    male     B
7    8        Asruid   Five     85    male     A
8    9       Tes Qry    Six     78     NaN     B
9   10      Big John   Four     55  female     D
10  11        Ronald    Six     89  female     A
11  12         Recky    Six     94  female     A
12  13           Kty  Seven     88  female     A
13  14         Bigy   Seven     88  female     A
14  15      Tade Row    NaN     88    male     A
15  16         Gimmy   Four     88    male     A
16  17         Tumyu    Six     54    male     D
17  18         Honny   Five     75    male     B
18  19         Tinny   Nine     18    male     D
19  20        Jackly   Nine     65  female     C
20  21    Babby John   Four     69  female     C
21  22        Reggid  Seven     55  female     D
22  23         Herod  Eight     79    male     B
23  24     Tiddy Now  Seven     78    male     B
24  25      Giff Tow  Seven     88    male     A
25  26        Crelea  Seven     79    male     B
26  27           NaN  Three     81     NaN     B
27  28     Rojj Base  Seven     86  female     A
28  29  Tess Played  Seven     55    male     D
29  30     Reppy Red    Six     79  female     B
```

```python
# Sort the DataFrame by 'score' in descending order
df_sorted = df.sort_values(by='score', ascending=False)
print(df_sorted)
```

```
    id          name  class  score  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
12  13          Kty  Seven     88  female     A
13  14         Bigy  Seven     88  female     A
27  28    Rojj Base  Seven     86  female     A
7    8       Asruid   Five     85    male     A
1    2     Max Ruin  Three     85    male     A
26  27          NaN  Three     81     NaN     B
29  30    Reppy Red    Six     79  female     B
25  26       Crelea  Seven     79    male     B
22  23        Herod  Eight     79    male     B
6    7  My John Rob  Fifth     78    male     B
23  24   Tiddy Now   Seven     78    male     B
8    9      Tes Qry    Six     78     NaN     B
17  18        Honny   Five     75    male     B
0    1     John Deo   Four     75  female     B
33  34     Gain Toe  Seven     69    male     C
20  21   Babby John   Four     69  female     C
19  20        Jackly   Nine     65  female     C
3    4   Krish Star   Four     60  female     C
4    5    John Mike   Four     60  female     C
2    3       Arnold  Three     55    male     D
5    6    Alex John   Four     55    male     D
9   10     Big John   Four     55  female     D
21  22        Reggid  Seven     55  female     D
28  29  Tess Played  Seven     55    male     D
16  17        Tumyu    Six     54    male     D
18  19        Tinny   Nine     18    male     D
```

Exercise 6: Exporting Data

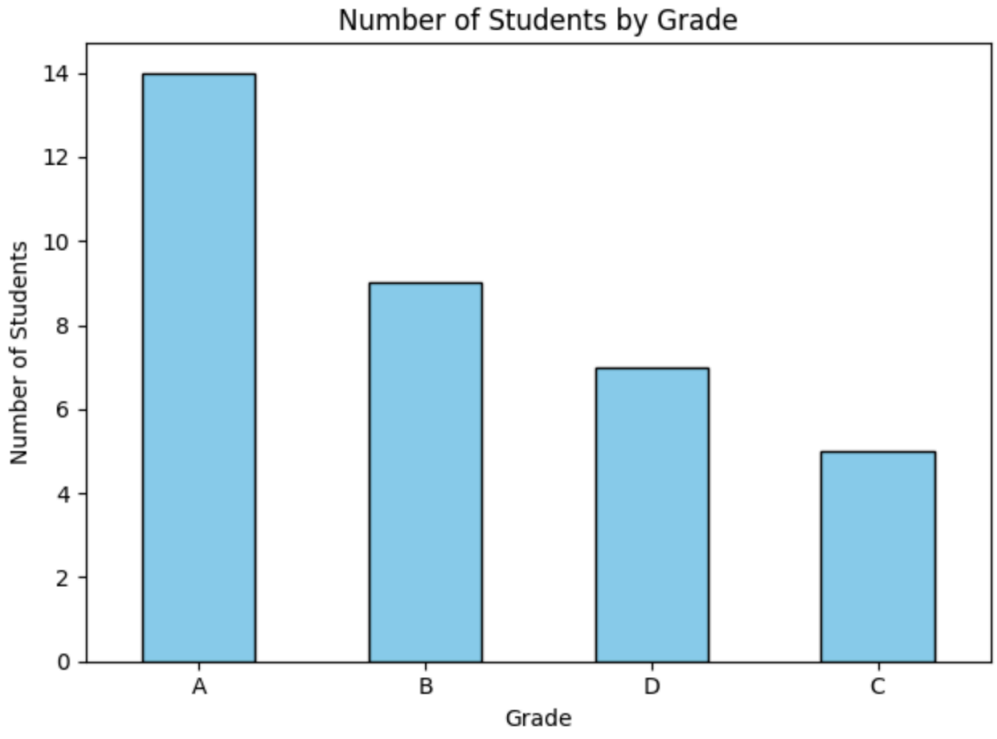
1. Question: "Write the code to save the DataFrame with the new 'grade' column to a new CSV file."

```python
# Save the DataFrame to a new CSV file
df.to_csv('updated_students.csv', index=False)

# Show the updated DataFrame
print(df)
```

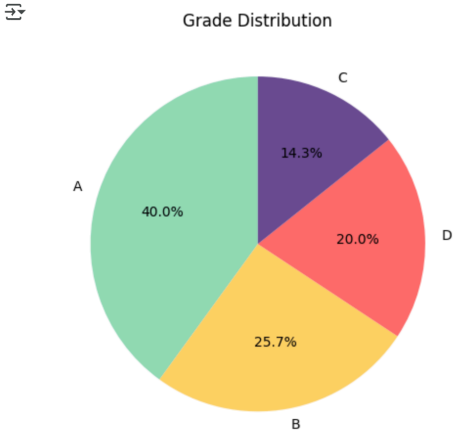
```
    id          name  class  score  gender grade
0    1      John Deo   Four     75  female     B
1    2      Max Ruin  Three     85    male     A
2    3        Arnold  Three     55    male     D
3    4    Krish Star   Four     60  female     C
4    5     John Mike   Four     60  female     C
5    6     Alex John   Four     55    male     D
6    7  My John Rob  Fifth     78    male     B
7    8        Asruid   Five     85    male     A
8    9       Tes Qry    Six     78     NaN     B
9   10      Big John   Four     55  female     D
10  11        Ronald    Six     89  female     A
11  12         Recky    Six     94  female     A
12  13           Kty  Seven     88  female     A
13  14         Bigy   Seven     88  female     A
14  15     Tade Row     NaN     88    male     A
15  16         Gimmy   Four     88    male     A
16  17         Tumyu    Six     54    male     D
17  18         Honny   Five     75    male     B
18  19         Tinny   Nine     18    male     D
19  20        Jackly   Nine     65  female     C
20  21    Babby John   Four     69  female     C
21  22        Reggid  Seven     55  female     D
22  23         Herod  Eight     79    male     B
23  24     Tiddy Now  Seven     78    male     B
24  25      Giff Tow  Seven     88    male     A
25  26        Crelea  Seven     79    male     B
26  27           NaN  Three     81     NaN     B
27  28     Rojj Base  Seven     86  female     A
28  29  Tess Played  Seven     55    male     D
29  30     Reppy Red    Six     79  female     B
30  31  Marry Toeey   Four     88    male     A
31  32     Binn Rott  Seven     90  female     A
32  33     Kenn Rein    Six     96  female     A
33  34      Gain Toe  Seven     69    male     C
34  35    Rows Noump    Six     88  female     A
```

Exercise 7: If finished early try visualising the results

```
import matplotlib.pyplot as plt

# Count students per grade
grade_counts = df['grade'].value_counts()

# Plot as bar chart
grade_counts.plot(kind='bar', color='skyblue', edgecolor='black')

plt.title('Number of Students by Grade')
plt.xlabel('Grade')
plt.ylabel('Number of Students')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



```
# Pie chart of grade distribution
df['grade'].value_counts().plot(kind='pie', autopct='%1.1f%%', startangle=90, colors=['#8fd9b6', '#ffd166', '#ff6b6b', '#6a4c93'])

plt.title('Grade Distribution')
plt.ylabel('')  # Hide y-label
plt.tight_layout()
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.

- 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

```
[9] import pandas as pd

    # Read the CSV file into a DataFrame named 'df'
    df = pd.read_csv('GDP (nominal) per Capita.csv', index_col=0)
```

```
# Print the first 10 rows of the DataFrame
print(df.head(10))
```

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

    WorldBank_Year  UN_Estimate UN_Year
1             2021       234317    2021
2             2020       169260    2021
3             2021       133745    2021
4             2021       101109    2021
5             2021       112653    2021
6             2021        89242    2021
7             2021        93525    2021
8             2021        66822    2021
9             2019            0       0
10            2021        85250    2021
```

```
# Print the last 5 rows of the DataFrame
print(df.tail(5))
```

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

     WorldBank_Year  UN_Estimate UN_Year
219            2021          613    2021
220            2015          400    2021
221            2021          505    2021
222            2021          373    2021
223            2021          311    2021
```

```
# Print only the 'Country/Territory' and 'UN_Region' columns
print(df[['Country/Territory', 'UN_Region']])
```

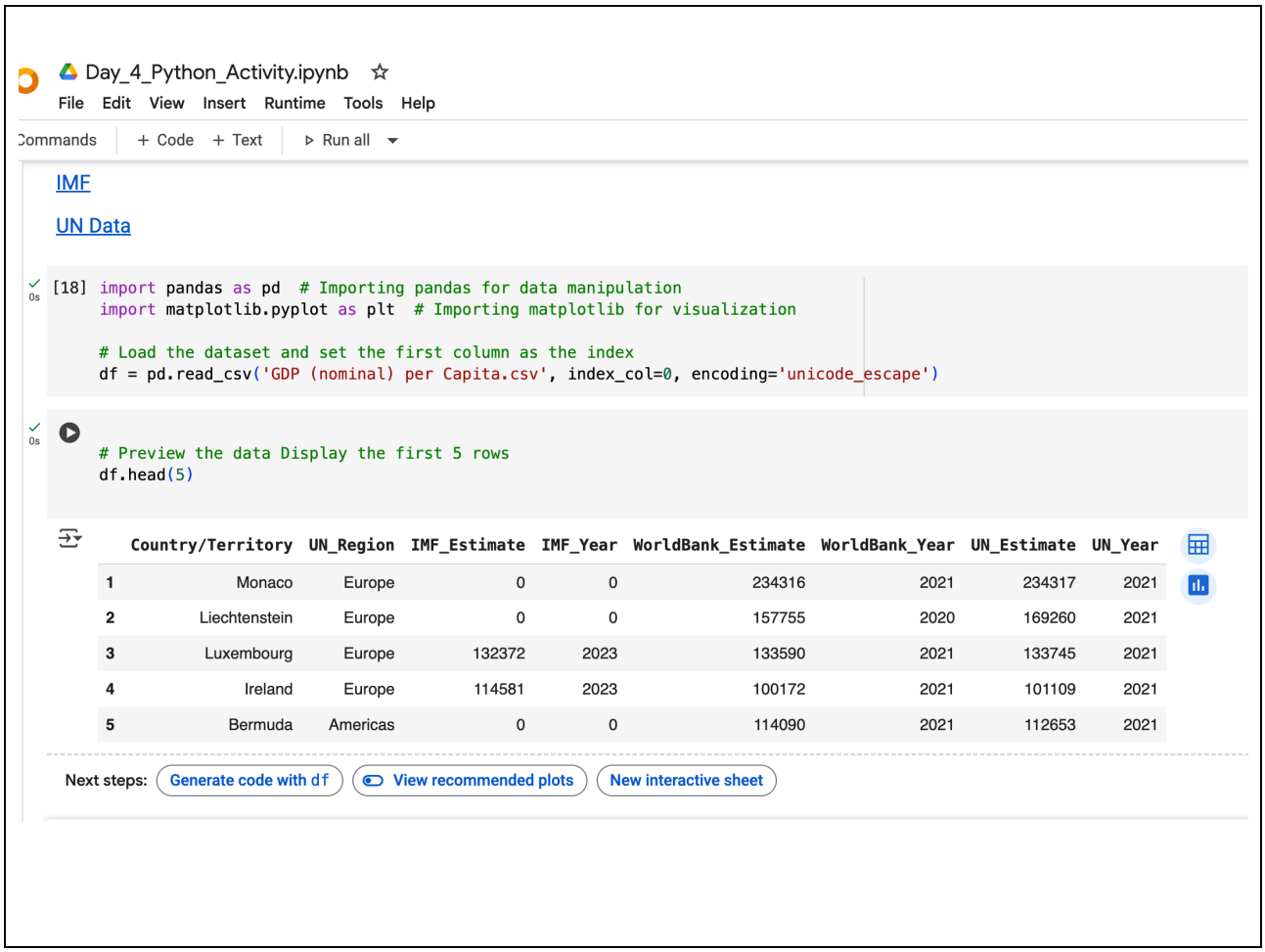
```
    Country/Territory UN_Region
1              Monaco    Europe
2       Liechtenstein    Europe
3          Luxembourg    Europe
4             Ireland    Europe
5             Bermuda  Americas
..                ...       ...
219            Malawi    Africa
220       South Sudan    Africa
221     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

[223 rows x 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.]



	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

Use this section to explore and inspect dataset.

```
[22] # Display the last 5 rows
     df.tail()
```

	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

```
# Display just the Country/Territory and UN_Region columns
df[['Country/Territory', 'UN_Region']]
```

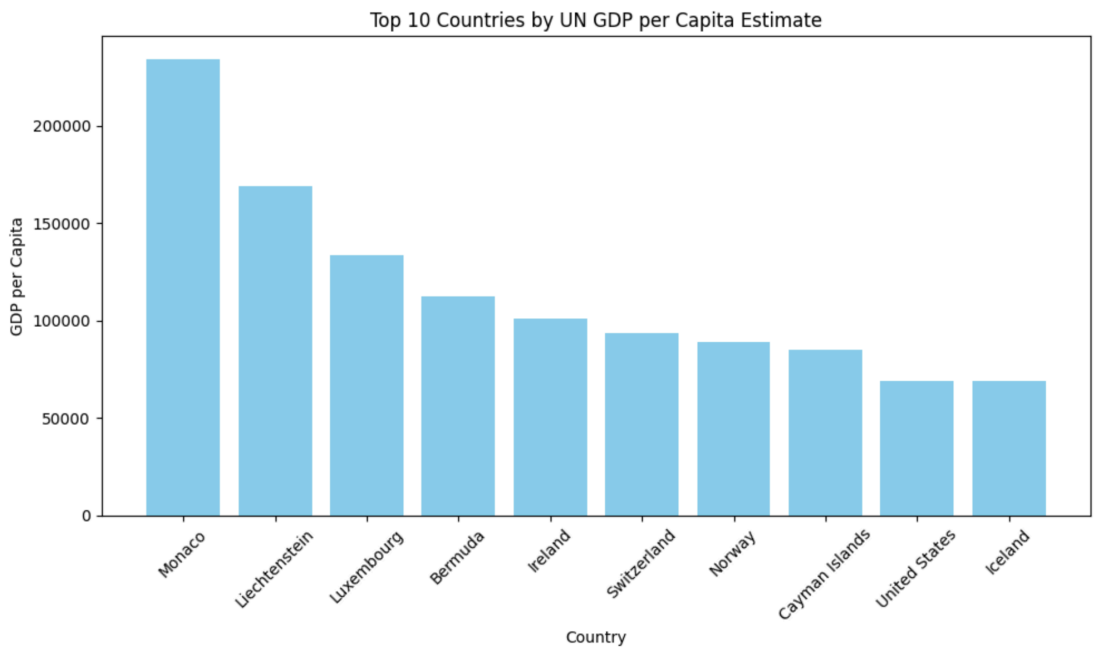
	Country/Territory	UN_Region
1	Monaco	Europe
2	Liechtenstein	Europe
3	Luxembourg	Europe
4	Ireland	Europe
5	Bermuda	Americas
...	...	...
219	Malawi	Africa
220	South Sudan	Africa
221	Sierra Leone	Africa
222	Afghanistan	Asia
223	Burundi	Africa

223 rows × 2 columns

```
# Find countries with IMF GDP per capita over $100,000
high_gdp = df[df['IMF_Estimate'] > 100000]
high_gdp[['Country/Territory', 'IMF_Estimate']]
```

	Country/Territory	IMF_Estimate
3	Luxembourg	132372
4	Ireland	114581
6	Norway	101103

```python
# Plot Top 10 countries by UN GDP estimate
top10 = df.sort_values(by='UN_Estimate', ascending=False).head(10)

plt.figure(figsize=(10,6))
plt.bar(top10['Country/Territory'], top10['UN_Estimate'], color='skyblue')
plt.title('Top 10 Countries by UN GDP per Capita Estimate')
plt.xlabel('Country')
plt.ylabel('GDP per Capita')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




```python
# number of countries per region
# Count the number of countries in each UN_Region
countries_per_region = df.groupby('UN_Region')['Country/Territory'].count()

# Print the result
print(countries_per_region)
```

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

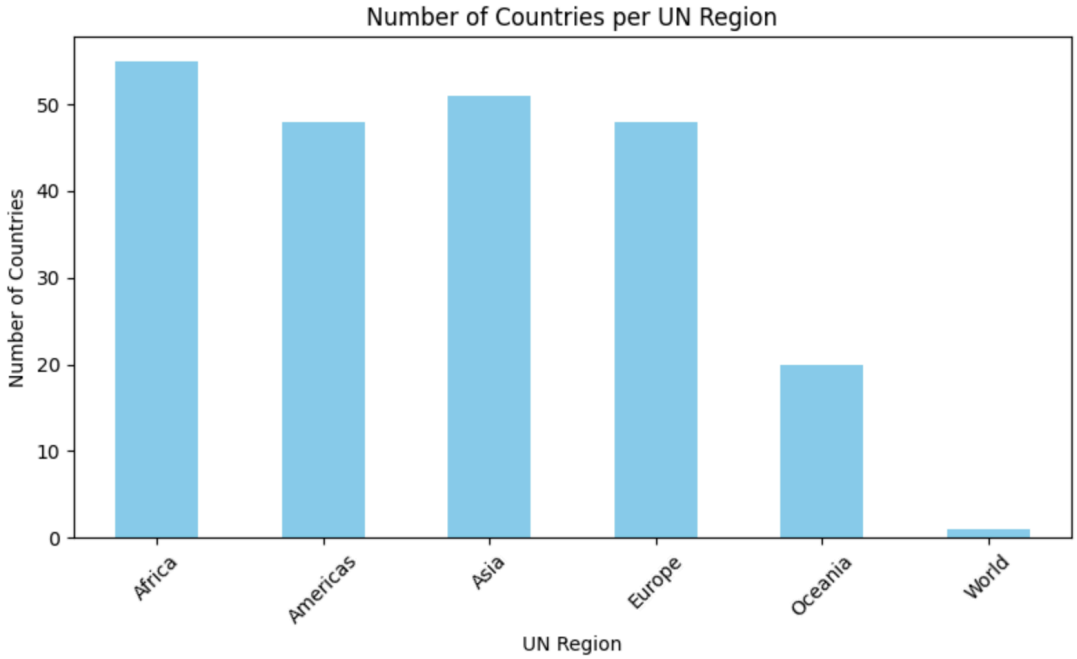
```
[13] Start coding or generate with AI.
```

```
[38] # Grouped by UN Region - Average GDP
     avg_un_region = df.groupby('UN_Region')['UN_Estimate'].mean().sort_values(ascending=False)
     print(avg_un_region)
```

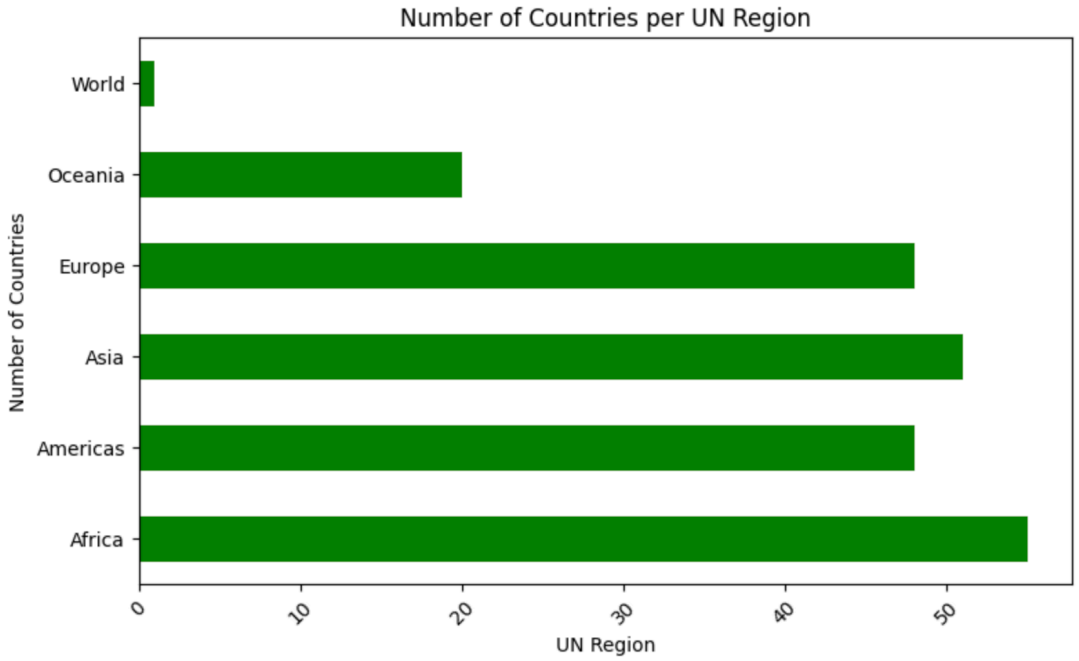
```
UN_Region
Europe      40610.791667
Americas    18703.750000
Asia        14069.019608
Oceania     12613.750000
World       12230.000000
Africa       2417.927273
Name: UN_Estimate, dtype: float64
```

```
[12] Start coding or generate with AI
```

```
# Count countries per region
countries_per_region = df.groupby('UN_Region')['Country/Territory'].count()

# Plotting the bar chart
plt.figure(figsize=(8,5))
countries_per_region.plot(kind='bar', color='skyblue')

plt.title('Number of Countries per UN Region')
plt.xlabel('UN Region')
plt.ylabel('Number of Countries')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



```python
# Count countries per region
countries_per_region = df.groupby('UN_Region')['Country/Territory'].count()

# Plotting the bar chart
plt.figure(figsize=(8,5))
countries_per_region.plot(kind='barh', color='green')

plt.title('Number of Countries per UN Region')
plt.xlabel('UN Region')
plt.ylabel('Number of Countries')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




```python
#What is European Union[n 1]?
european_union = df[df['Country/Territory'] == 'European Union[n 1]']
display(european_union)
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
36	European Union[n 1]	Europe	39940	2023	38411	2021	31875	2021

```python
# Count European Union occurence
eu_count = (df['Country/Territory'] == "European Union[n 1]").sum()

print("European Union occurs:", eu_count, "times")
```

```
European Union occurs: 1 times
```

```python
# Countries in Europe below avarege
# Step 1: Select only the rows where the region is Europe
europe = df[df['UN_Region'] == 'Europe']

# Step 2: Calculate the average GDP per capita for Europe
avg_gdp = europe['UN_Estimate'].mean()

# Step 3: Find countries in Europe with GDP below the average
below_avg = europe[europe['UN_Estimate'] < avg_gdp]

# Step 4: Print the average and the countries below it
print("Average GDP per capita in Europe:", round(avg_gdp, 2))
print("\nEuropean countries below average:\n")
print(below_avg[['Country/Territory', 'UN_Estimate']])
```

```
Average GDP per capita in Europe: 40610.79

European countries below average:

          Country/Territory  UN_Estimate
9                 Isle of Man            0
14          Channel Islands              0
15            Faroe Islands              0
36      European Union[n 1]          31875
40                    Malta          33642
41                    Italy          35579
51                 Slovenia          29135
52           Czech Republic          26809
53                    Spain          30058
54                  Estonia          27991
57                Lithuania          23844
59                 Portugal          24651
60                   Latvia          21267
62                 Slovakia          21390
63                   Greece          20571
70                  Croatia          16983
72                   Poland          17736
75                  Hungary          18728
78                  Romania          14698
87                 Bulgaria          12207
90                   Russia          12259
103              Montenegro           9252
106                  Serbia           8643
112  Bosnia and Herzegovina           7143
115                 Belarus           7121
118         North Macedonia           6600
120                 Albania           6296
```

```python
# Which countries in Europe has higher GDP than UK?

# Step 1: Filter only European countries
europe_df = df[df['UN_Region'] == 'Europe']

# Step 2: Get the UK's IMF GDP per capita
uk_gdp = df[df['Country/Territory'] == 'United Kingdom']['IMF_Estimate'].values[0]

# Step 3: Find European countries with higher GDP than the UK
higher_than_uk = europe_df[europe_df['IMF_Estimate'] > uk_gdp]

# Step 4: Print results
print("UK GDP per capita (IMF):", uk_gdp)
print("\nEuropean countries with higher GDP than the UK:\n")
print(higher_than_uk[['Country/Territory', 'IMF_Estimate']])
```

```
UK GDP per capita (IMF): 46371

European countries with higher GDP than the UK:

   Country/Territory  IMF_Estimate
3         Luxembourg        132372
4            Ireland        114581
6             Norway        101103
7        Switzerland         98767
13           Iceland         75180
16           Denmark         68827
18       Netherlands         61098
20           Austria         56802
22            Sweden         55395
23           Finland         54351
24           Belgium         53377
25        San Marino         52949
28           Germany         51383
```

```
# Grouped by UN Region - Average GDP
df.groupby('UN_Region')['IMF_Estimate'].mean()
```

UN_Region	IMF_Estimate
Africa	2802.345455
Americas	11871.041667
Asia	16665.254902
Europe	34446.750000
Oceania	9133.150000
World	13440.000000

dtype: float64

Which countries below average by IMF world estimate?

```
# Step 1: Calculate the average IMF GDP estimate across all countries
world_avg_imf = df['IMF_Estimate'].mean()

# Step 2: Filter countries with IMF GDP estimate below the world average
below_world_avg = df[df['IMF_Estimate'] < world_avg_imf]

# Step 3: Print the average and countries below it
print("World average IMF GDP per capita:", round(world_avg_imf, 2))
print("\nCountries with IMF GDP per capita below world average:\n")
print(below_world_avg[['Country/Territory', 'IMF_Estimate']])
```

```
World average IMF GDP per capita: 15351.63

Countries with IMF GDP per capita below world average:

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

[159 rows x 2 columns]
```

```
Start coding or generate with AI.
```

IMF estimate 0 **values**

```
# Filter countries where IMF_Estimate is exactly 0
imf_zero = df[df['IMF_Estimate'] == 0]

# Print the countries with IMF GDP estimate = 0
print("Countries with IMF GDP estimate equal to 0:\n")
print(imf_zero[['Country/Territory', 'IMF_Estimate']])
```

```
Countries with IMF GDP estimate equal to 0:

              Country/Territory  IMF_Estimate
1                        Monaco             0
2                 Liechtenstein             0
5                       Bermuda             0
9                   Isle of Man             0
10              Cayman Islands             0
14             Channel Islands             0
15               Faroe Islands             0
19                   Greenland             0
31      British Virgin Islands             0
37           US Virgin Islands             0
39               New Caledonia             0
42                        Guam             0
58    Sint Maarten (Dutch part)             0
61     Northern Mariana Islands             0
65   Saint Martin (French part)             0
68     Turks and Caicos Islands             0
71             French Polynesia             0
76                 Cook Islands             0
77                     Anguilla             0
82                     CuraÃ§ao             0
85                   Montserrat             0
86               American Samoa             0
104                        Cuba             0
196                     Zanzibar             0
204                       Syria             0
212                North Korea              0
```

Which country has highest UN Estimate?

```python
# Find the index of the max UN_Estimate
max_idx = df['UN_Estimate'].idxmax()

# Get the row with the highest UN_Estimate
highest_un = df.loc[max_idx, ['Country/Territory', 'UN_Estimate']]

print("Country with the highest UN Estimate GDP per capita:")
print(highest_un)
```

```
Country with the highest UN Estimate GDP per capita:
Country/Territory    Monaco
UN_Estimate          234317
Name: 1, dtype: object
```

Which country has highest Worlbank Estimate?

```python
# Find the index of the max WorldBank_Estimate
max_idx = df['WorldBank_Estimate'].idxmax()

# Get the row with the highest WorldBank_Estimate
highest_wb = df.loc[max_idx, ['Country/Territory', 'WorldBank_Estimate']]

print("Country with the highest World Bank Estimate GDP per capita:")
print(highest_wb)
```

```
Country with the highest World Bank Estimate GDP per capita:
Country/Territory     Monaco
WorldBank_Estimate    234316
Name: 1, dtype: object
```

Which country has highest IMF Estimate?

```
# Find the index of the max IMF_Estimate
max_idx = df['IMF_Estimate'].idxmax()

# Get the row with the highest IMF_Estimate
highest_imf = df.loc[max_idx, ['Country/Territory', 'IMF_Estimate']]

print("Country with the highest IMF Estimate GDP per capita:")
print(highest_imf)
```

```
Country with the highest IMF Estimate GDP per capita:
Country/Territory    Luxembourg
IMF_Estimate             132372
Name: 3, dtype: object
```

Filling 0 Values by average

```
[64] import numpy as np
```

```
[81] # replace 0 with null values
     df.replace(0, pd.NA, inplace=True)
```

```
# Calculate the average of 'Worldbank_Estimate' and 'UN_Estimate' columns
df['avg_worldbank_un'] = df[['WorldBank_Estimate', 'UN_Estimate']].mean(axis=1)
df.head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	avg_worldbank_un
1	Monaco	Europe	<NA>	<NA>	234316	2021	234317	2021	234316.5
2	Liechtenstein	Europe	<NA>	<NA>	157755	2020	169260	2021	163507.5
3	Luxembourg	Europe	132372	2023	133590	2021	133745	2021	133667.5
4	Ireland	Europe	114581	2023	100172	2021	101109	2021	100640.5
5	Bermuda	Americas	<NA>	<NA>	114090	2021	112653	2021	113371.5

```
[101] # Fill the null values in 'imf' column with the calculated average
      # Step 1: Calculate the average of IMF_Estimate (excluding nulls)
      imf_avg = df['IMF_Estimate'].mean()

      # Step 2: Fill null values in the column with the average, and ensure the type is correct
      df['IMF_Estimate'] = df['IMF_Estimate'].fillna(imf_avg).astype(float)

      # Step 3: Print confirmation
      print("Filled missing IMF_Estimate values with average:", round(imf_avg, 2))
```

```
Filled missing IMF_Estimate values with average: 17377.74
```

```
# Drop the temporary 'avg_worldbank_un' column if not needed
df.drop(columns=['avg_worldbank_un'], inplace=True)
```

Checking Missing Values

Visit this link to learn more about bfill

```
[105] # Show how many missing (NaN) values are in each column
      print(df.isna().sum())
```

```
Country/Territory      0
UN_Region              0
IMF_Estimate           0
IMF_Year              26
WorldBank_Estimate     7
WorldBank_Year         7
UN_Estimate            9
UN_Year                0
dtype: int64
```

```
[106] # Returns True if any missing values exist in the DataFrame
      print(df.isna().any().any())
```

```
True
```

```
# Filter and show rows where at least one column is missing (NaN)
missing_rows = df[df.isna().any(axis=1)]
print(missing_rows)
```

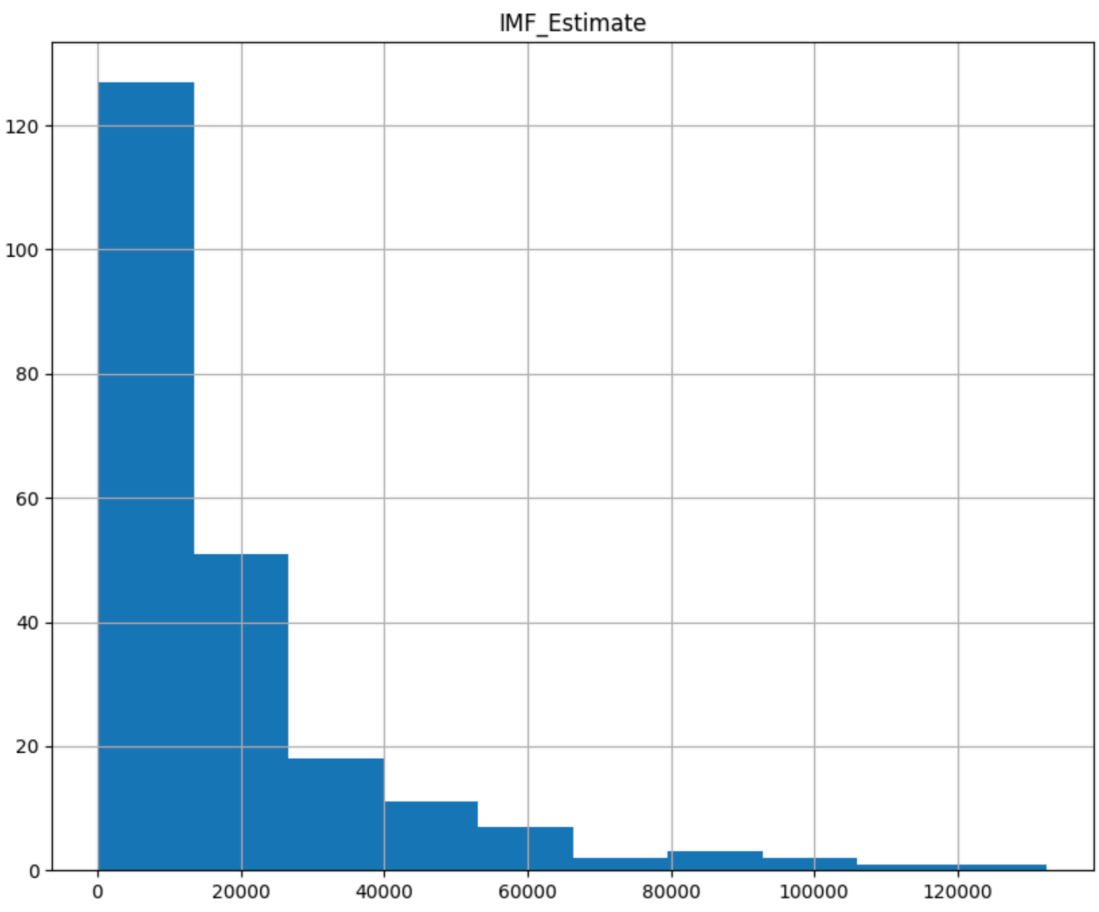
```
              Country/Territory UN_Region  IMF_Estimate IMF_Year  \
1                        Monaco    Europe  17377.736041     <NA>
2                 Liechtenstein    Europe  17377.736041     <NA>
5                       Bermuda  Americas  17377.736041     <NA>
9                   Isle of Man    Europe  17377.736041     <NA>
10                Cayman Islands  Americas  17377.736041     <NA>
14               Channel Islands    Europe  17377.736041     <NA>
15                 Faroe Islands    Europe  17377.736041     <NA>
19                     Greenland  Americas  17377.736041     <NA>
31       British Virgin Islands  Americas  17377.736041     <NA>
37            US Virgin Islands  Americas  17377.736041     <NA>
39                New Caledonia   Oceania  17377.736041     <NA>
42                         Guam   Oceania  17377.736041     <NA>
46                       Taiwan      Asia  33907.000000     2023
58    Sint Maarten (Dutch part)  Americas  17377.736041     <NA>
61     Northern Mariana Islands   Oceania  17377.736041     <NA>
65   Saint Martin (French part)  Americas  17377.736041     <NA>
68     Turks and Caicos Islands  Americas  17377.736041     <NA>
71              French Polynesia   Oceania  17377.736041     <NA>
76                  Cook Islands   Oceania  17377.736041     <NA>
77                      Anguilla  Americas  17377.736041     <NA>
82                    CuraÃ§ao  Americas  17377.736041     <NA>
85                    Montserrat  Americas  17377.736041     <NA>
86                American Samoa   Oceania  17377.736041     <NA>
104                         Cuba  Americas  17377.736041     <NA>
196                     Zanzibar    Africa  17377.736041     <NA>
204                        Syria      Asia  17377.736041     <NA>
212                  North Korea      Asia  17377.736041     <NA>

    WorldBank_Estimate WorldBank_Year UN_Estimate UN_Year
1               234316           2021      234317    2021
2               157755           2020      169260    2021
5               114090           2021      112653    2021
9                87158           2019        <NA>       0
10               86569           2021       85250    2021
14               75153           2007        <NA>       0
15               69010           2021        <NA>       0
19               54571           2020       58185    2021
31                <NA>           <NA>       49444    2021
37               39552           2020        <NA>       0
39               37160           2021       34994    2021
42               35905           2021        <NA>       0
46                <NA>           <NA>        <NA>       0
58               28988           2018       26199    2021
```

Visualization

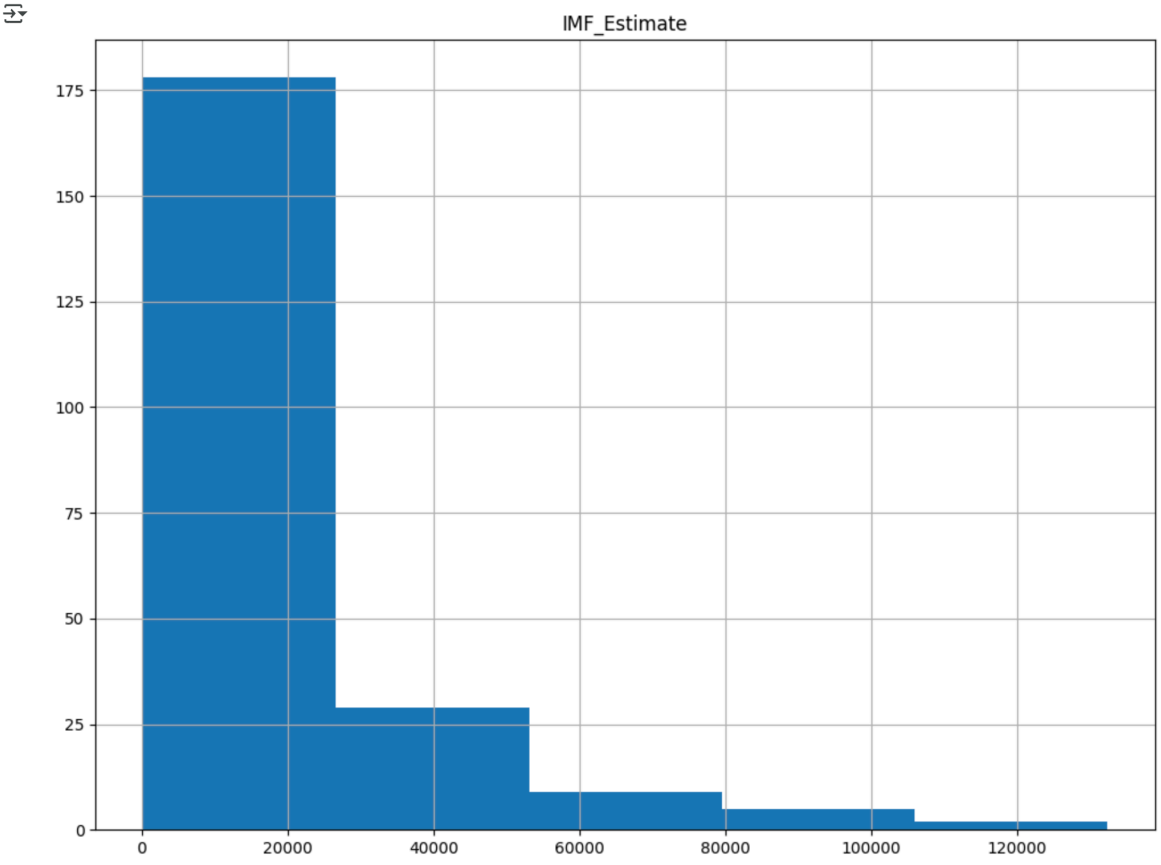
```
[110] import matplotlib.pyplot as plt
      import seaborn as sns
```

Histogram

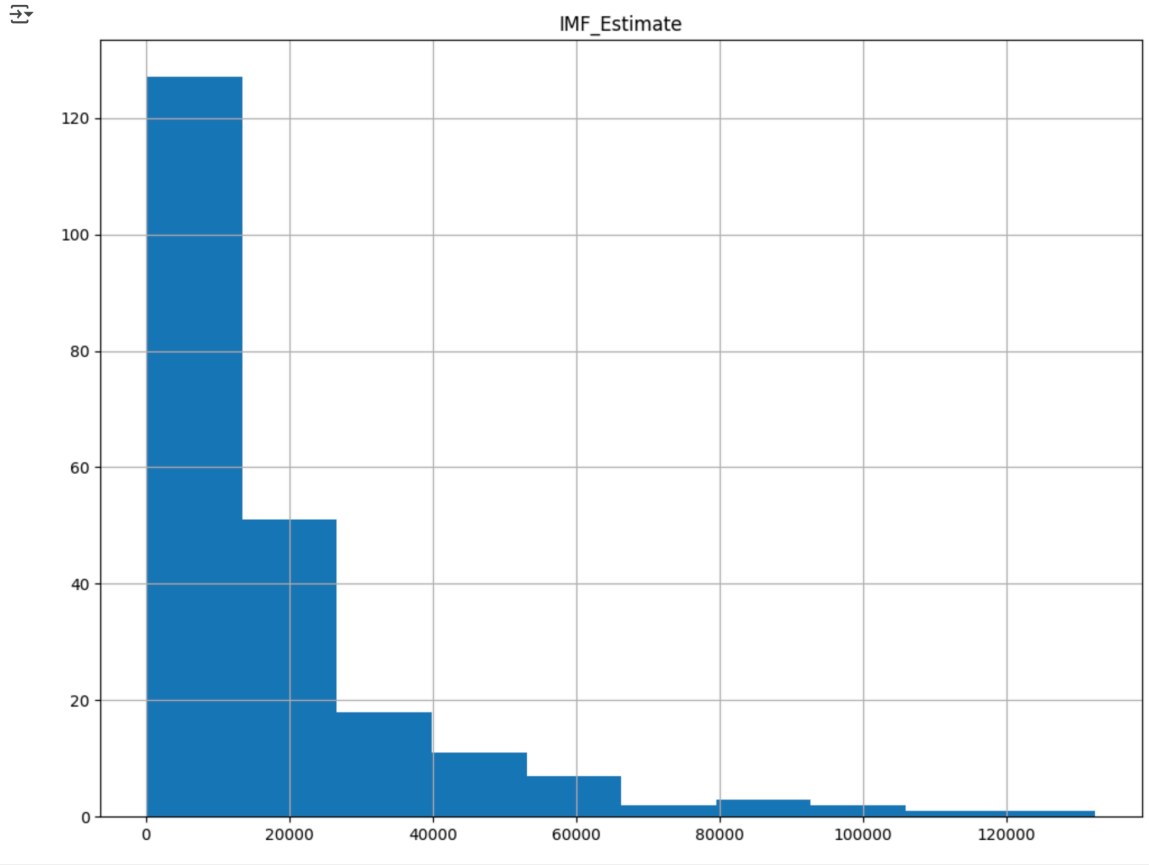
```
df.hist(figsize=(10,8))
plt.show()
```



```
df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(bins=5, figsize=(12,9))

plt.show()
```



```
df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(figsize=(12,9))

plt.show()
```



```
df["WorldBank_Estimate"].agg(["min","max"])
```

	WorldBank_Estimate
min	222
max	234316

dtype: int64

```
[116] 234316/5
      #1 bin size if bins=5
```

```
46863.2
```

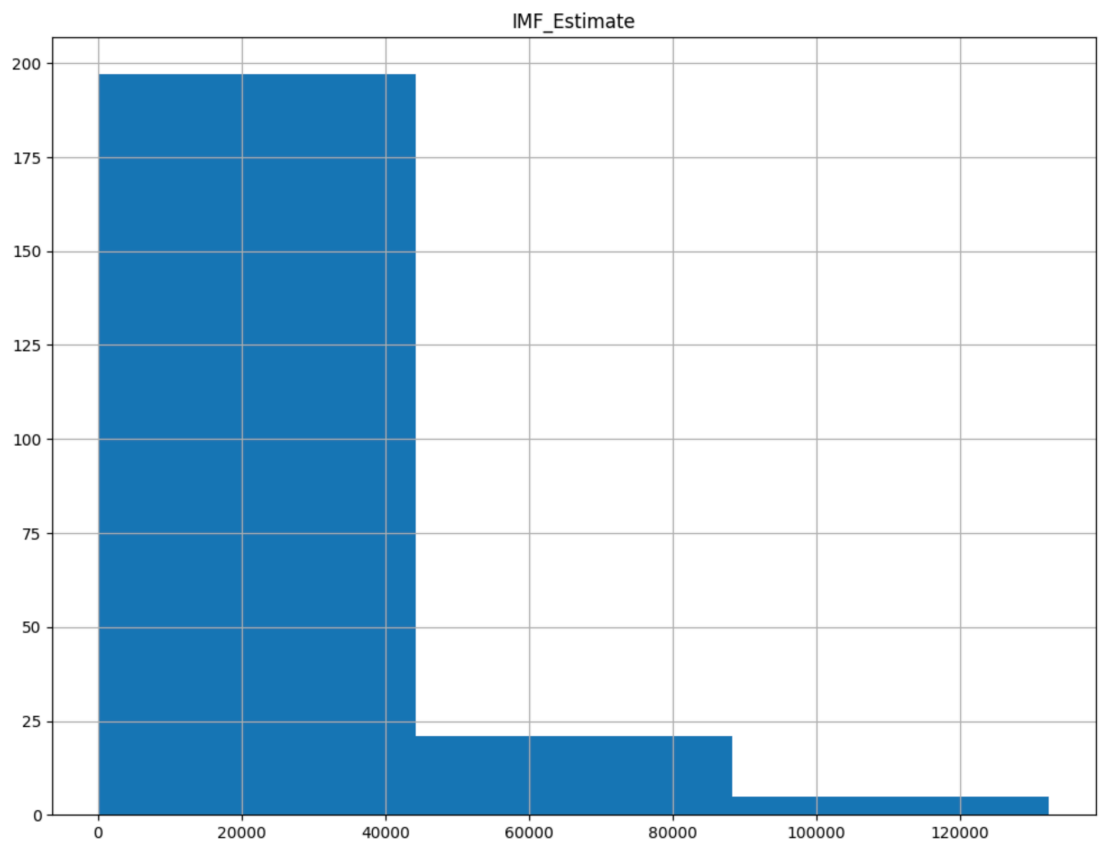
```
[117] df[df["WorldBank_Estimate"]<=46863.2]["WorldBank_Estimate"].count()
```

```
np.int64(188)
```

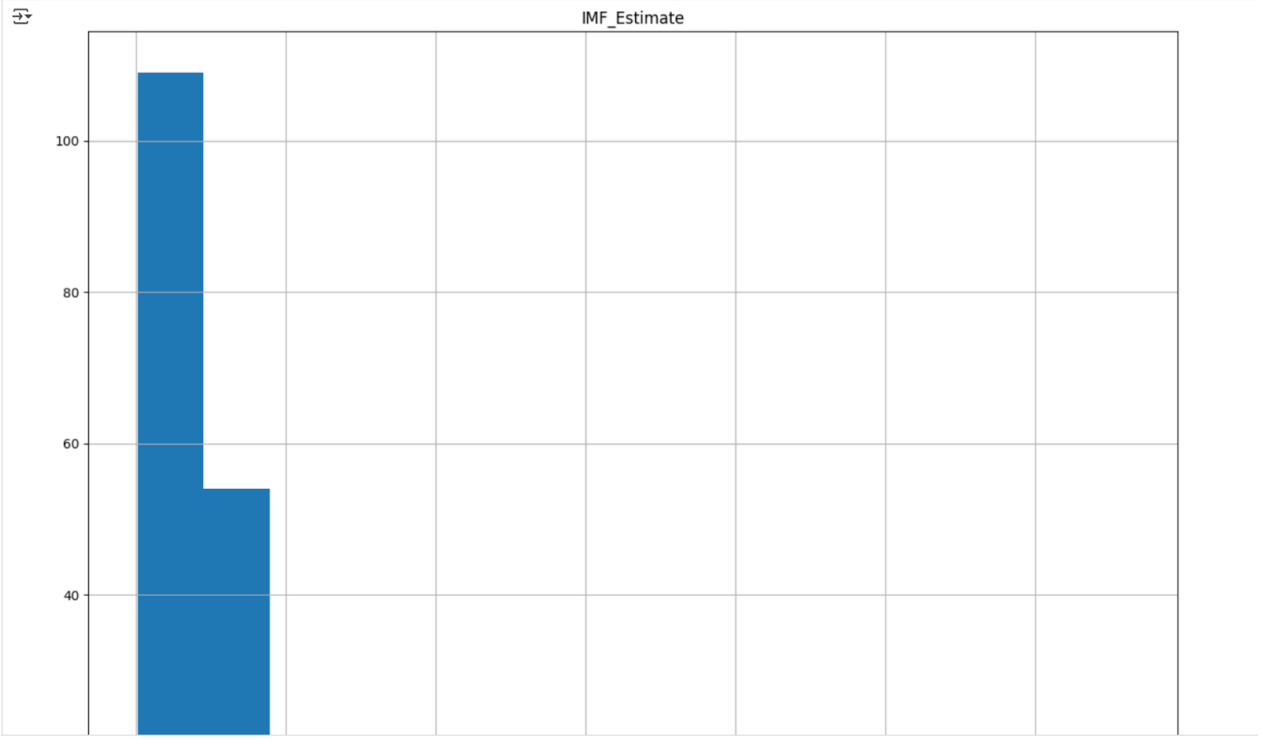
```
[118] 234316/10
      #1 bin size if bins not given any number
```

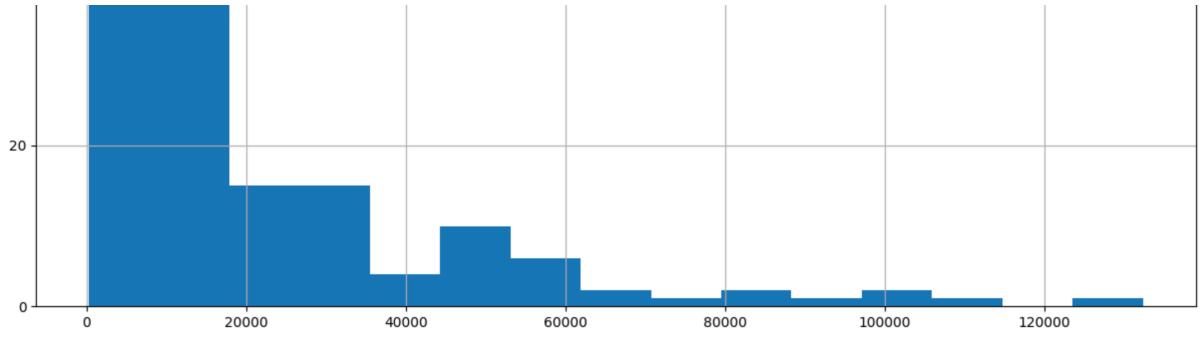
```
23431.6
```

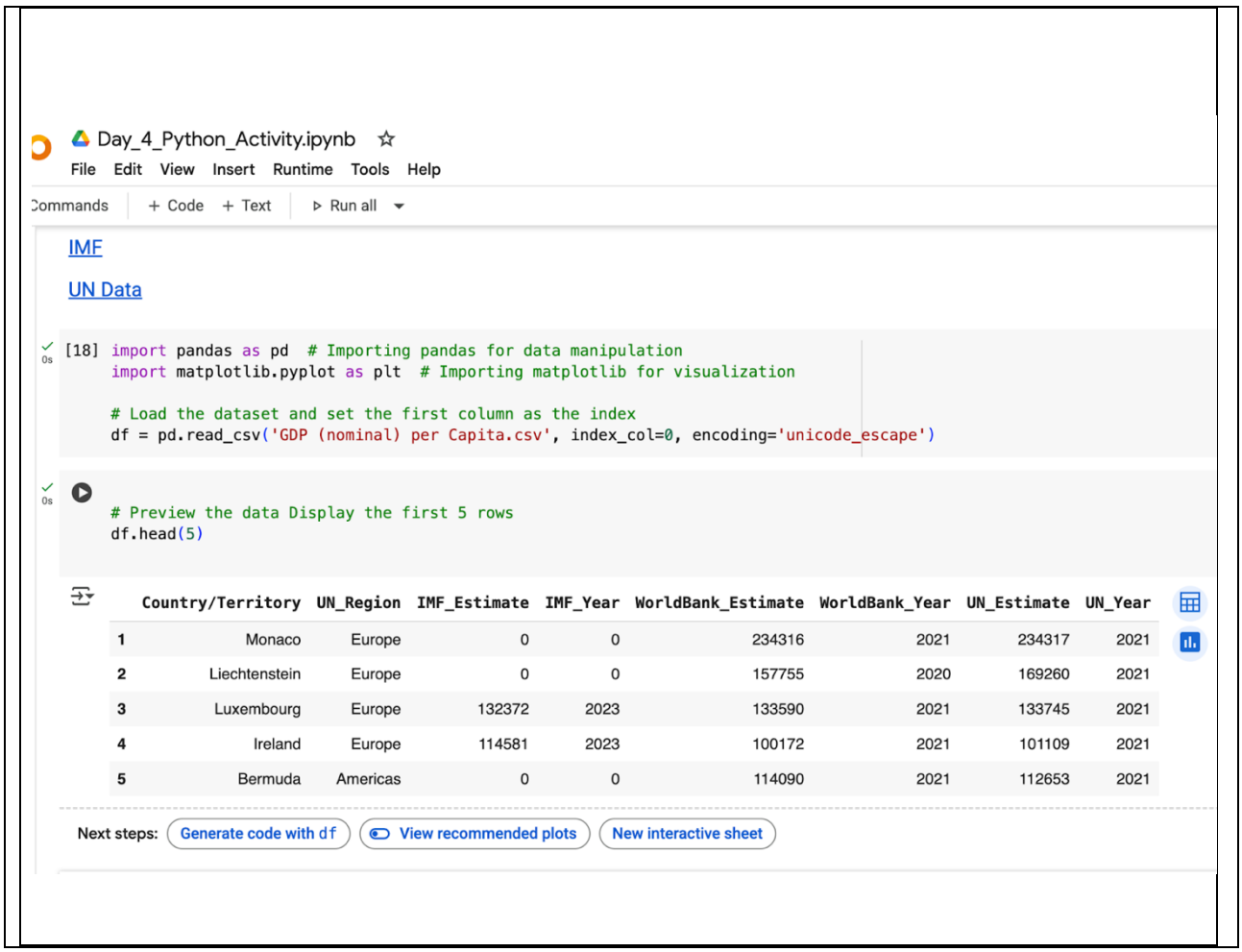
```
df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(bins=3, figsize=(12,9))

plt.show()
```



```
df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(bins=15, figsize=(15,12))

#23400/15 = 15300
plt.show()
```







	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

Use this section to explore and inspect dataset.

```
[22] # Display the last 5 rows
     df.tail()
```

	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

```
# Display just the Country/Territory and UN_Region columns
df[['Country/Territory', 'UN_Region']]
```

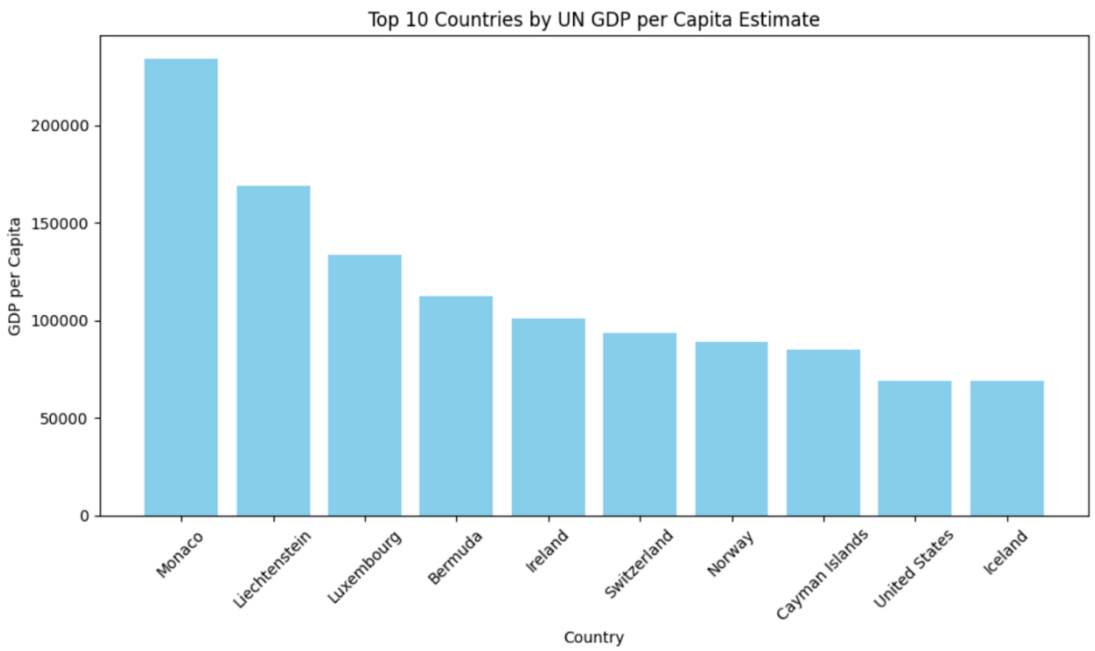
	Country/Territory	UN_Region
1	Monaco	Europe
2	Liechtenstein	Europe
3	Luxembourg	Europe
4	Ireland	Europe
5	Bermuda	Americas
...	...	...
219	Malawi	Africa
220	South Sudan	Africa
221	Sierra Leone	Africa
222	Afghanistan	Asia
223	Burundi	Africa

223 rows × 2 columns

```
# Find countries with IMF GDP per capita over $100,000
high_gdp = df[df['IMF_Estimate'] > 100000]
high_gdp[['Country/Territory', 'IMF_Estimate']]
```

	Country/Territory	IMF_Estimate
3	Luxembourg	132372
4	Ireland	114581
6	Norway	101103

```python
# Plot Top 10 countries by UN GDP estimate
top10 = df.sort_values(by='UN_Estimate', ascending=False).head(10)

plt.figure(figsize=(10,6))
plt.bar(top10['Country/Territory'], top10['UN_Estimate'], color='skyblue')
plt.title('Top 10 Countries by UN GDP per Capita Estimate')
plt.xlabel('Country')
plt.ylabel('GDP per Capita')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




```python
# number of countries per region
# Count the number of countries in each UN_Region
countries_per_region = df.groupby('UN_Region')['Country/Territory'].count()

# Print the result
print(countries_per_region)
```

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

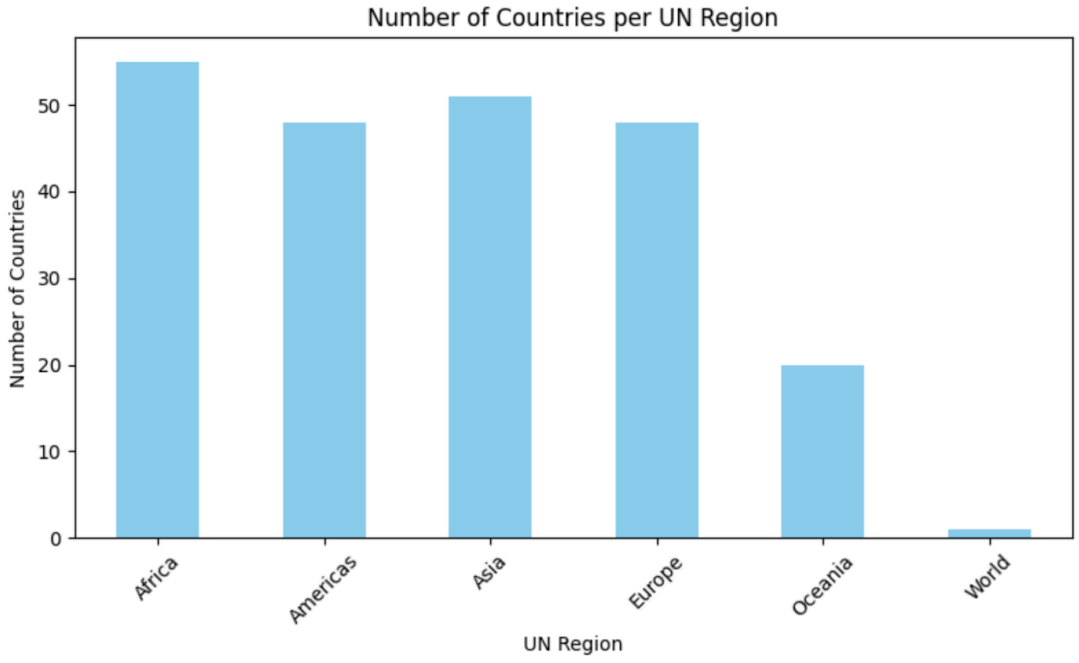
```
[13] Start coding or generate with AI.
```

```
[38] # Grouped by UN Region - Average GDP
     avg_un_region = df.groupby('UN_Region')['UN_Estimate'].mean().sort_values(ascending=False)
     print(avg_un_region)
```

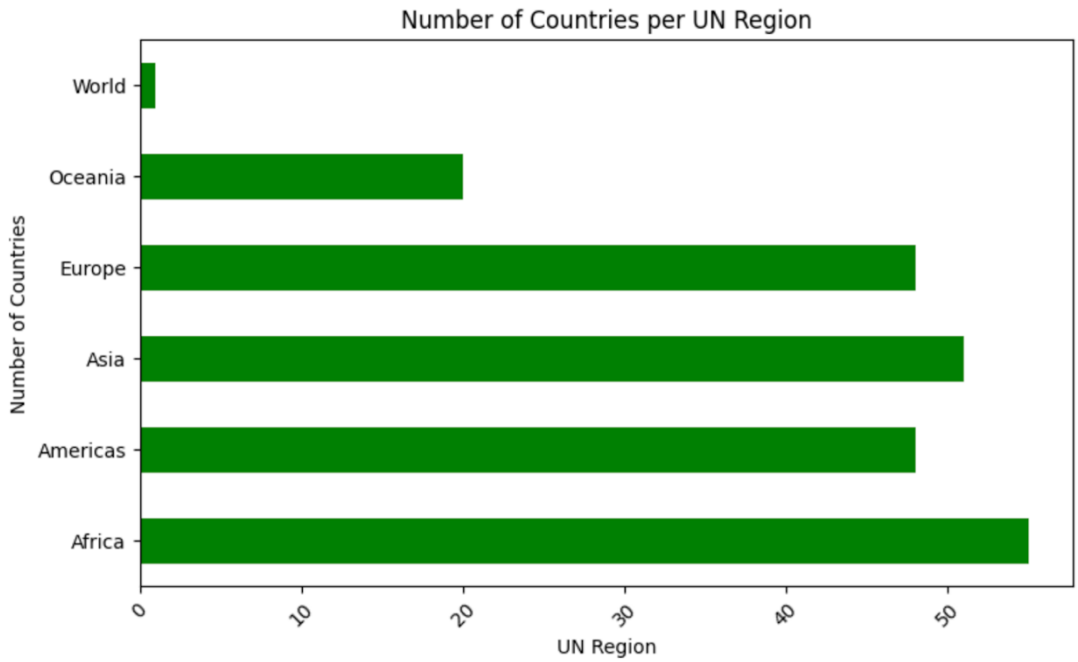
```
UN_Region
Europe      40610.791667
Americas    18703.750000
Asia        14069.019608
Oceania     12613.750000
World       12230.000000
Africa       2417.927273
Name: UN_Estimate, dtype: float64
```

```
[12] Start coding or generate with AI
```

```
# Count countries per region
countries_per_region = df.groupby('UN_Region')['Country/Territory'].count()

# Plotting the bar chart
plt.figure(figsize=(8,5))
countries_per_region.plot(kind='bar', color='skyblue')

plt.title('Number of Countries per UN Region')
plt.xlabel('UN Region')
plt.ylabel('Number of Countries')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



```python
# Count countries per region
countries_per_region = df.groupby('UN_Region')['Country/Territory'].count()

# Plotting the bar chart
plt.figure(figsize=(8,5))
countries_per_region.plot(kind='barh', color='green')

plt.title('Number of Countries per UN Region')
plt.xlabel('UN Region')
plt.ylabel('Number of Countries')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




```python
#What is European Union[n 1]?
european_union = df[df['Country/Territory'] == 'European Union[n 1]']
display(european_union)
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
36	European Union[n 1]	Europe	39940	2023	38411	2021	31875	2021

```python
# Count European Union occurence
eu_count = (df['Country/Territory'] == "European Union[n 1]").sum()

print("European Union occurs:", eu_count, "times")
```

```
European Union occurs: 1 times
```

```python
# Countries in Europe below avarege
# Step 1: Select only the rows where the region is Europe
europe = df[df['UN_Region'] == 'Europe']

# Step 2: Calculate the average GDP per capita for Europe
avg_gdp = europe['UN_Estimate'].mean()

# Step 3: Find countries in Europe with GDP below the average
below_avg = europe[europe['UN_Estimate'] < avg_gdp]

# Step 4: Print the average and the countries below it
print("Average GDP per capita in Europe:", round(avg_gdp, 2))
print("\nEuropean countries below average:\n")
print(below_avg[['Country/Territory', 'UN_Estimate']])
```

```
Average GDP per capita in Europe: 40610.79

European countries below average:

          Country/Territory  UN_Estimate
9                 Isle of Man            0
14          Channel Islands              0
15            Faroe Islands              0
36      European Union[n 1]          31875
40                    Malta          33642
41                    Italy          35579
51                 Slovenia          29135
52           Czech Republic          26809
53                    Spain          30058
54                  Estonia          27991
57                Lithuania          23844
59                 Portugal          24651
60                   Latvia          21267
62                 Slovakia          21390
63                   Greece          20571
70                  Croatia          16983
72                   Poland          17736
75                  Hungary          18728
78                  Romania          14698
87                 Bulgaria          12207
90                   Russia          12259
103              Montenegro           9252
106                  Serbia           8643
112  Bosnia and Herzegovina           7143
115                 Belarus           7121
118         North Macedonia           6600
120                 Albania           6396
```

```python
# Which countries in Europe has higher GDP than UK?

# Step 1: Filter only European countries
europe_df = df[df['UN_Region'] == 'Europe']

# Step 2: Get the UK's IMF GDP per capita
uk_gdp = df[df['Country/Territory'] == 'United Kingdom']['IMF_Estimate'].values[0]

# Step 3: Find European countries with higher GDP than the UK
higher_than_uk = europe_df[europe_df['IMF_Estimate'] > uk_gdp]

# Step 4: Print results
print("UK GDP per capita (IMF):", uk_gdp)
print("\nEuropean countries with higher GDP than the UK:\n")
print(higher_than_uk[['Country/Territory', 'IMF_Estimate']])
```

```
UK GDP per capita (IMF): 46371

European countries with higher GDP than the UK:

   Country/Territory  IMF_Estimate
3         Luxembourg        132372
4            Ireland        114581
6             Norway        101103
7        Switzerland         98767
13           Iceland         75180
16           Denmark         68827
18       Netherlands         61098
20           Austria         56802
22            Sweden         55395
23           Finland         54351
24           Belgium         53377
25        San Marino         52949
28           Germany         51383
```

```
# Grouped by UN Region - Average GDP
df.groupby('UN_Region')['IMF_Estimate'].mean()
```

UN_Region	IMF_Estimate
Africa	2802.345455
Americas	11871.041667
Asia	16665.254902
Europe	34446.750000
Oceania	9133.150000
World	13440.000000

dtype: float64

Which countries below average by IMF world estimate?

```
# Step 1: Calculate the average IMF GDP estimate across all countries
world_avg_imf = df['IMF_Estimate'].mean()

# Step 2: Filter countries with IMF GDP estimate below the world average
below_world_avg = df[df['IMF_Estimate'] < world_avg_imf]

# Step 3: Print the average and countries below it
print("World average IMF GDP per capita:", round(world_avg_imf, 2))
print("\nCountries with IMF GDP per capita below world average:\n")
print(below_world_avg[['Country/Territory', 'IMF_Estimate']])
```

```
World average IMF GDP per capita: 15351.63

Countries with IMF GDP per capita below world average:

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

[159 rows x 2 columns]
```

```
[ ] Start coding or generate with AI.
```

IMF estimate 0 **values**

```
# Filter countries where IMF_Estimate is exactly 0
imf_zero = df[df['IMF_Estimate'] == 0]

# Print the countries with IMF GDP estimate = 0
print("Countries with IMF GDP estimate equal to 0:\n")
print(imf_zero[['Country/Territory', 'IMF_Estimate']])
```

```
Countries with IMF GDP estimate equal to 0:

              Country/Territory  IMF_Estimate
1                        Monaco             0
2                 Liechtenstein             0
5                       Bermuda             0
9                   Isle of Man             0
10                Cayman Islands             0
14               Channel Islands             0
15                 Faroe Islands             0
19                     Greenland             0
31       British Virgin Islands             0
37            US Virgin Islands             0
39                New Caledonia             0
42                         Guam             0
58    Sint Maarten (Dutch part)             0
61     Northern Mariana Islands             0
65   Saint Martin (French part)             0
68     Turks and Caicos Islands             0
71              French Polynesia             0
76                  Cook Islands             0
77                      Anguilla             0
82                      CuraÃ§ao             0
85                    Montserrat             0
86                American Samoa             0
104                         Cuba             0
196                      Zanzibar             0
204                         Syria             0
212                  North Korea             0
```

Which country has highest UN Estimate?

```
# Find the index of the max UN_Estimate
max_idx = df['UN_Estimate'].idxmax()

# Get the row with the highest UN_Estimate
highest_un = df.loc[max_idx, ['Country/Territory', 'UN_Estimate']]

print("Country with the highest UN Estimate GDP per capita:")
print(highest_un)
```

```
Country with the highest UN Estimate GDP per capita:
Country/Territory    Monaco
UN_Estimate          234317
Name: 1, dtype: object
```

Which country has highest Worlbank Estimate?

```
# Find the index of the max WorldBank_Estimate
max_idx = df['WorldBank_Estimate'].idxmax()

# Get the row with the highest WorldBank_Estimate
highest_wb = df.loc[max_idx, ['Country/Territory', 'WorldBank_Estimate']]

print("Country with the highest World Bank Estimate GDP per capita:")
print(highest_wb)
```

```
Country with the highest World Bank Estimate GDP per capita:
Country/Territory     Monaco
WorldBank_Estimate    234316
Name: 1, dtype: object
```

Which country has highest IMF Estimate?

```
# Find the index of the max IMF_Estimate
max_idx = df['IMF_Estimate'].idxmax()

# Get the row with the highest IMF_Estimate
highest_imf = df.loc[max_idx, ['Country/Territory', 'IMF_Estimate']]

print("Country with the highest IMF Estimate GDP per capita:")
print(highest_imf)
```

```
Country with the highest IMF Estimate GDP per capita:
Country/Territory    Luxembourg
IMF_Estimate             132372
Name: 3, dtype: object
```

Filling 0 Values by average

```
[64] import numpy as np
```

```
[81] # replace 0 with null values
     df.replace(0, pd.NA, inplace=True)
```

```
# Calculate the average of 'Worldbank_Estimate' and 'UN_Estimate' columns
df['avg_worldbank_un'] = df[['WorldBank_Estimate', 'UN_Estimate']].mean(axis=1)
df.head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	avg_worldbank_un
1	Monaco	Europe	<NA>	<NA>	234316	2021	234317	2021	234316.5
2	Liechtenstein	Europe	<NA>	<NA>	157755	2020	169260	2021	163507.5
3	Luxembourg	Europe	132372	2023	133590	2021	133745	2021	133667.5
4	Ireland	Europe	114581	2023	100172	2021	101109	2021	100640.5
5	Bermuda	Americas	<NA>	<NA>	114090	2021	112653	2021	113371.5

```
[101] # Fill the null values in 'imf' column with the calculated average
      # Step 1: Calculate the average of IMF_Estimate (excluding nulls)
      imf_avg = df['IMF_Estimate'].mean()

      # Step 2: Fill null values in the column with the average, and ensure the type is correct
      df['IMF_Estimate'] = df['IMF_Estimate'].fillna(imf_avg).astype(float)

      # Step 3: Print confirmation
      print("Filled missing IMF_Estimate values with average:", round(imf_avg, 2))
```

```
Filled missing IMF_Estimate values with average: 17377.74
```

```
# Drop the temporary 'avg_worldbank_un' column if not needed
df.drop(columns=['avg_worldbank_un'], inplace=True)
```

Checking Missing Values

Visit this link to learn more about bfill

```
[105] # Show how many missing (NaN) values are in each column
      print(df.isna().sum())
```

```
Country/Territory      0
UN_Region              0
IMF_Estimate           0
IMF_Year              26
WorldBank_Estimate     7
WorldBank_Year         7
UN_Estimate            9
UN_Year                0
dtype: int64
```

```
[106] # Returns True if any missing values exist in the DataFrame
      print(df.isna().any().any())
```

```
True
```

```
# Filter and show rows where at least one column is missing (NaN)
missing_rows = df[df.isna().any(axis=1)]
print(missing_rows)
```

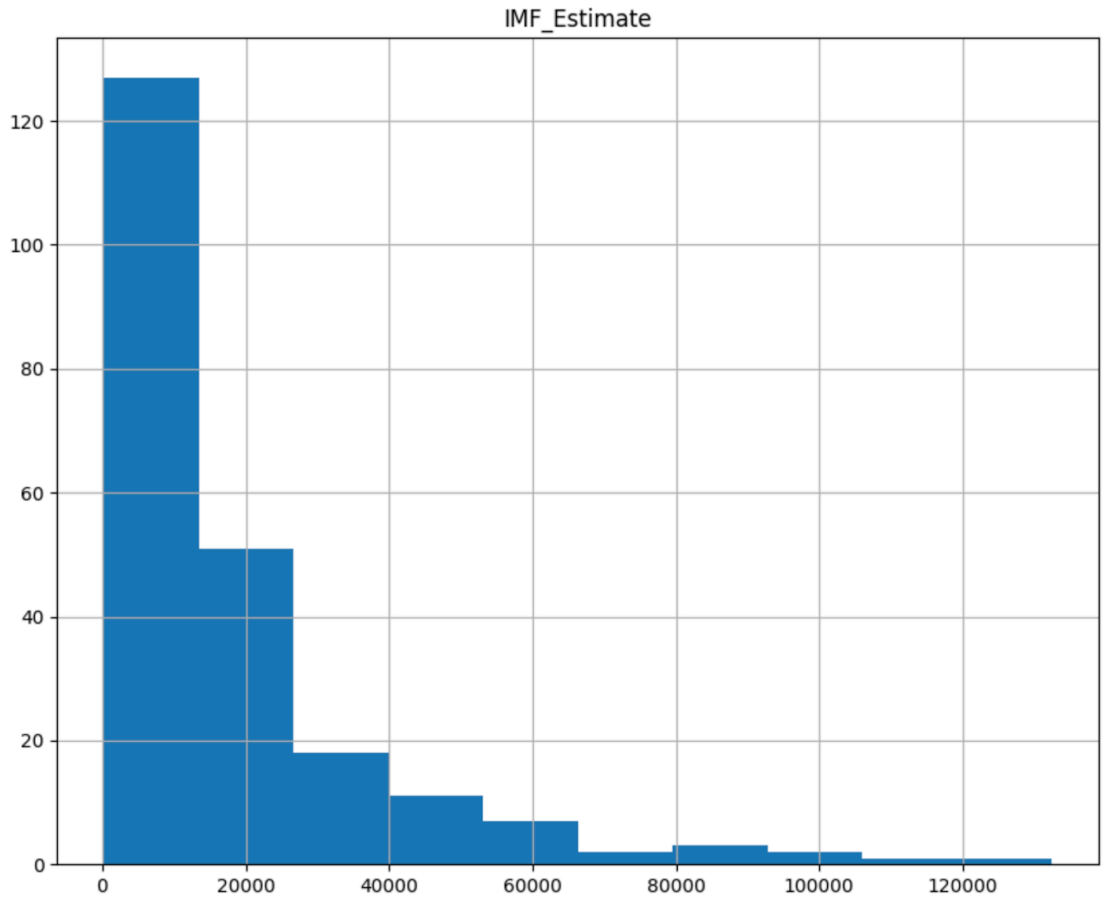
```
              Country/Territory UN_Region  IMF_Estimate IMF_Year  \
1                        Monaco    Europe  17377.736041     <NA>
2                 Liechtenstein    Europe  17377.736041     <NA>
5                       Bermuda  Americas  17377.736041     <NA>
9                   Isle of Man    Europe  17377.736041     <NA>
10                Cayman Islands  Americas  17377.736041     <NA>
14               Channel Islands    Europe  17377.736041     <NA>
15                 Faroe Islands    Europe  17377.736041     <NA>
19                     Greenland  Americas  17377.736041     <NA>
31       British Virgin Islands  Americas  17377.736041     <NA>
37            US Virgin Islands  Americas  17377.736041     <NA>
39                New Caledonia   Oceania  17377.736041     <NA>
42                         Guam   Oceania  17377.736041     <NA>
46                       Taiwan      Asia  33907.000000     2023
58    Sint Maarten (Dutch part)  Americas  17377.736041     <NA>
61     Northern Mariana Islands   Oceania  17377.736041     <NA>
65   Saint Martin (French part)  Americas  17377.736041     <NA>
68     Turks and Caicos Islands  Americas  17377.736041     <NA>
71              French Polynesia   Oceania  17377.736041     <NA>
76                  Cook Islands   Oceania  17377.736041     <NA>
77                      Anguilla  Americas  17377.736041     <NA>
82                      CuraÃ§ao  Americas  17377.736041     <NA>
85                    Montserrat  Americas  17377.736041     <NA>
86                American Samoa   Oceania  17377.736041     <NA>
104                         Cuba  Americas  17377.736041     <NA>
196                     Zanzibar    Africa  17377.736041     <NA>
204                        Syria      Asia  17377.736041     <NA>
212                  North Korea      Asia  17377.736041     <NA>

    WorldBank_Estimate WorldBank_Year UN_Estimate UN_Year
1               234316           2021      234317    2021
2               157755           2020      169260    2021
5               114090           2021      112653    2021
9                87158           2019        <NA>       0
10               86569           2021       85250    2021
14               75153           2007        <NA>       0
15               69010           2021        <NA>       0
19               54571           2020       58185    2021
31                <NA>           <NA>       49444    2021
37               39552           2020        <NA>       0
39               37160           2021       34994    2021
42               35905           2021        <NA>       0
46                <NA>           <NA>        <NA>       0
58               28988           2018       26199    2021
```

Visualization

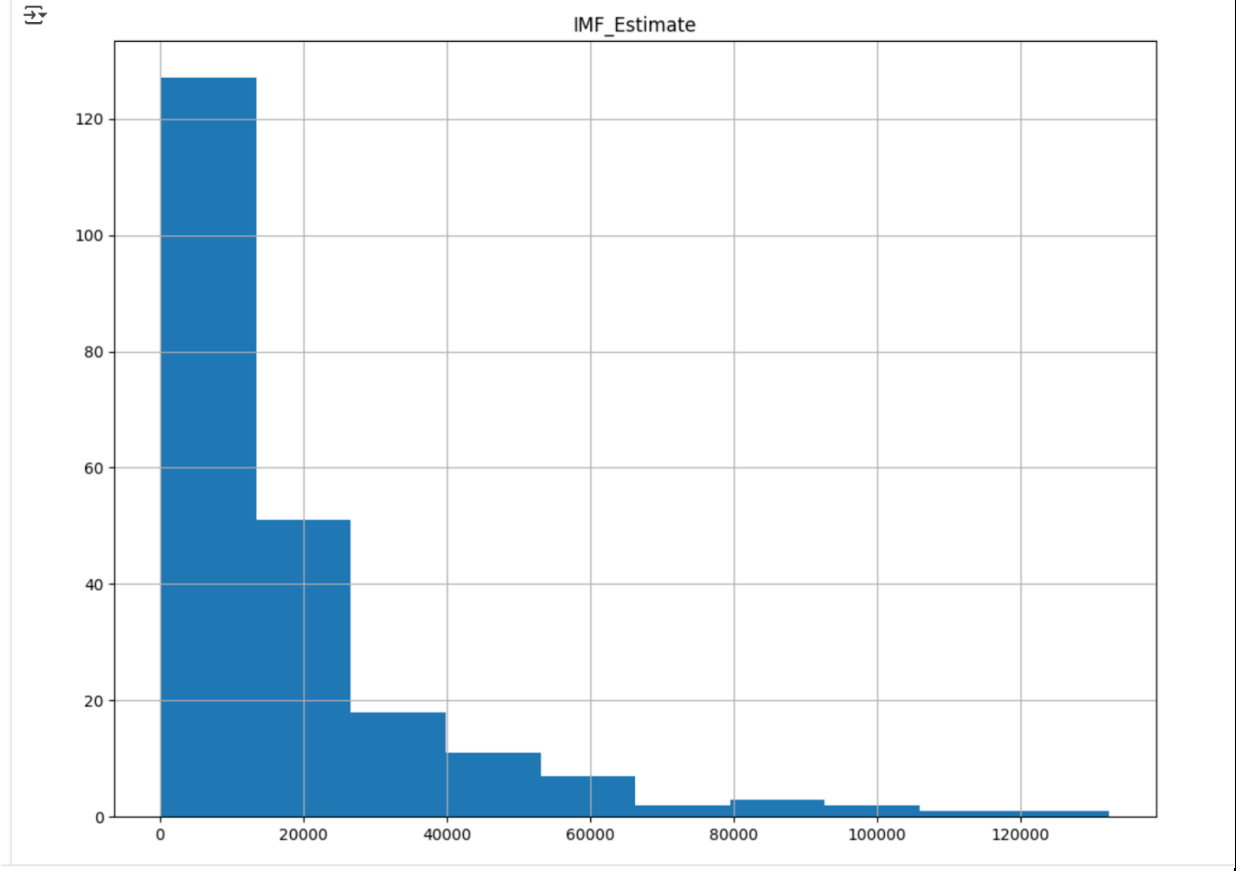
```
[110] import matplotlib.pyplot as plt
      import seaborn as sns
```

Histogram

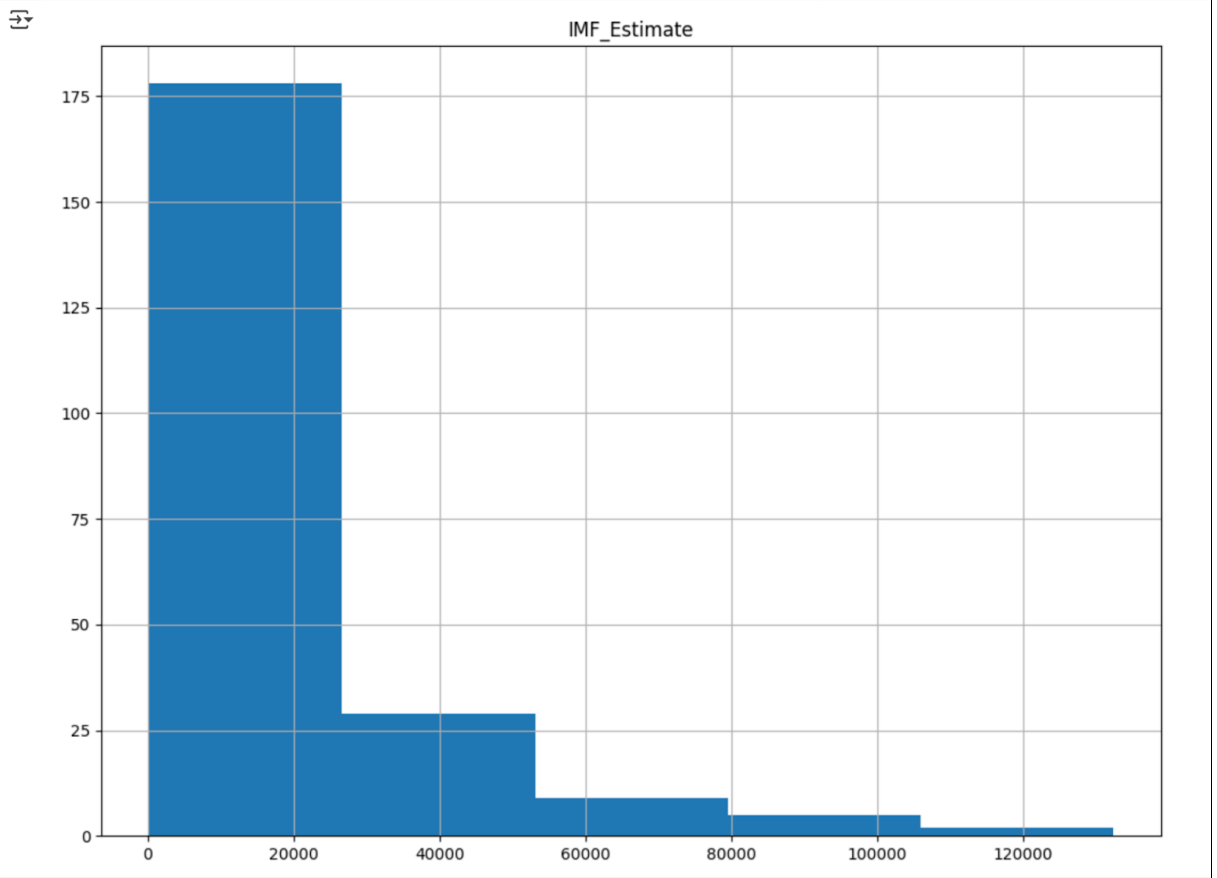
```
df.hist(figsize=(10,8))
plt.show()
```



```
df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(figsize=(12,9))

plt.show()
```



```
df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(bins=5, figsize=(12,9))

plt.show()
```



```
df["WorldBank_Estimate"].agg(["min","max"])
```

	WorldBank_Estimate
min	222
max	234316

dtype: int64

```
[116] 234316/5
      #1 bin size if bins=5
```

```
46863.2
```

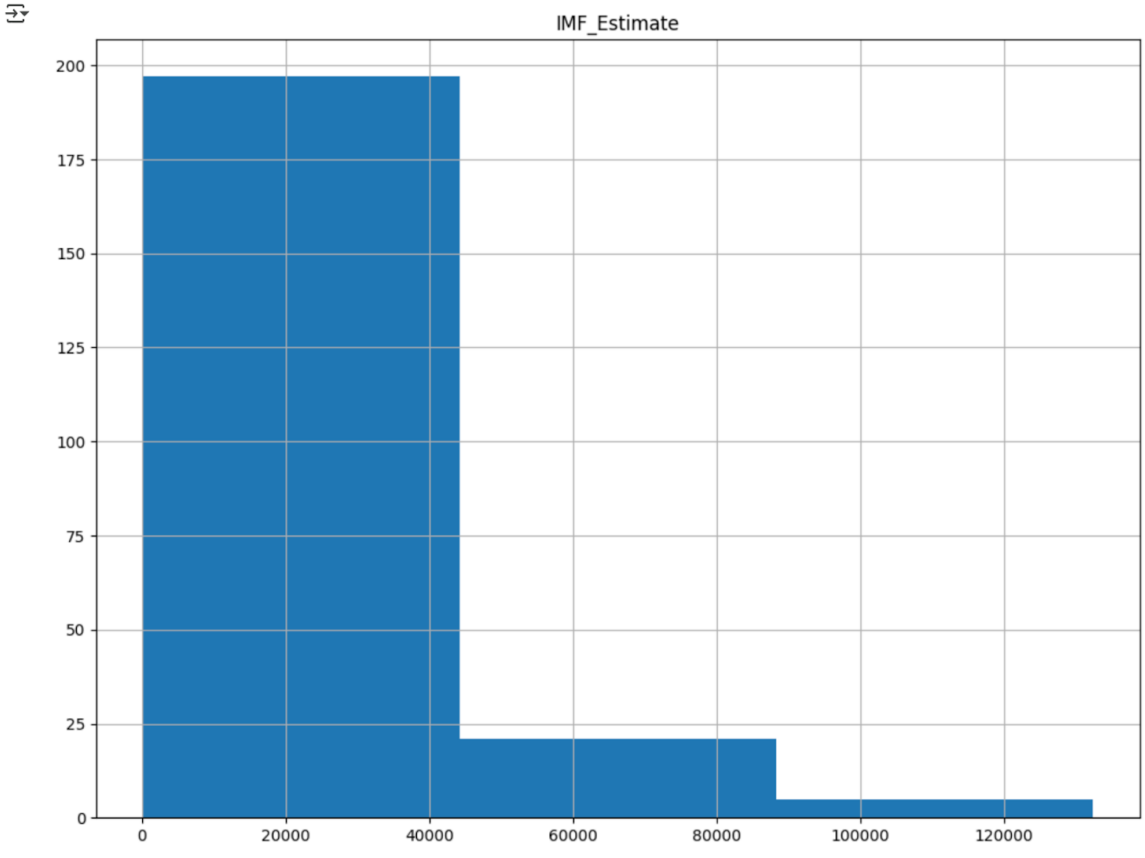
```
[117] df[df["WorldBank_Estimate"]<=46863.2]["WorldBank_Estimate"].count()
```

```
np.int64(188)
```

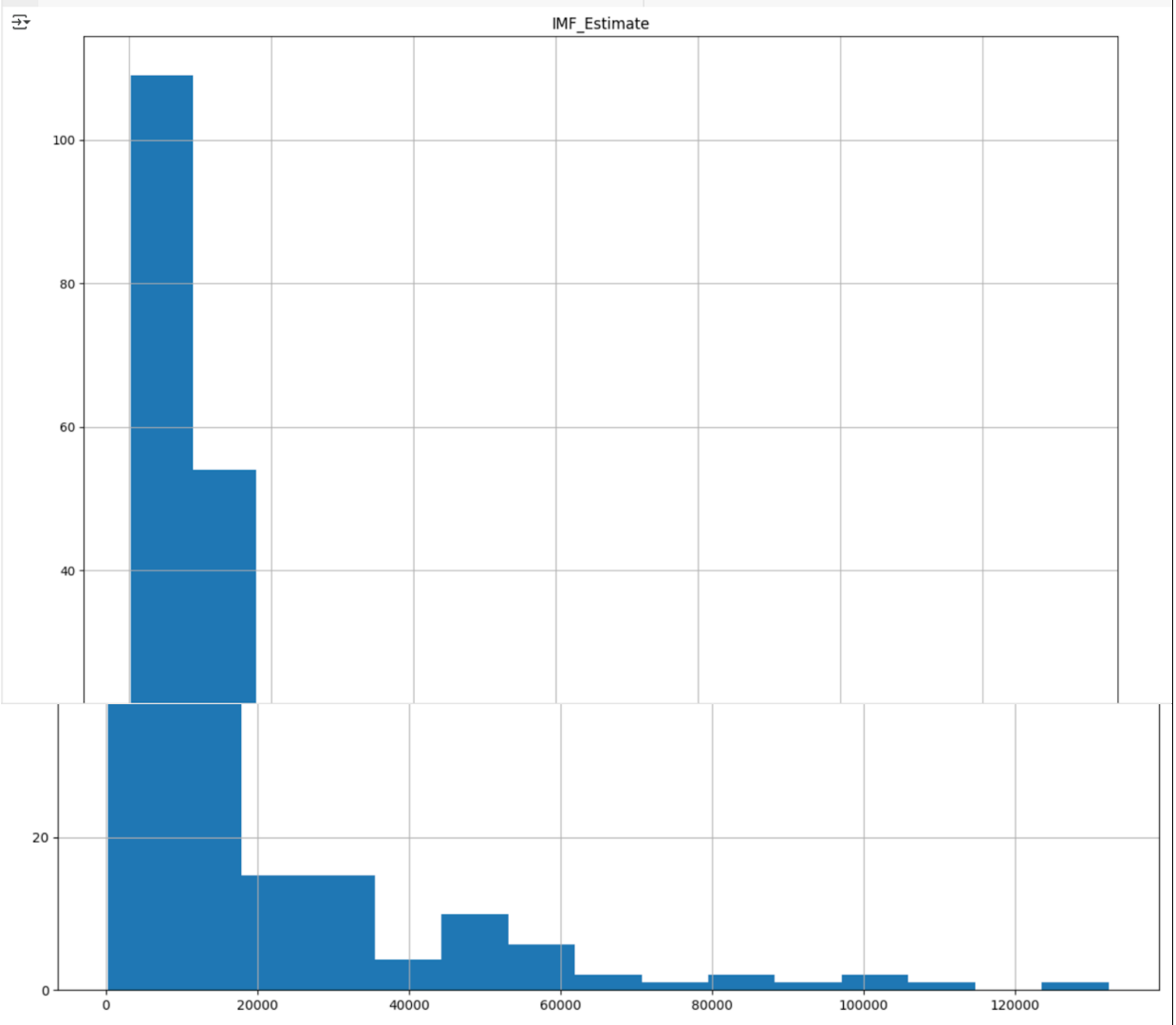
```
[118] 234316/10
      #1 bin size if bins not given any number
```

```
23431.6
```

```
df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(bins=3, figsize=(12,9))

plt.show()
```



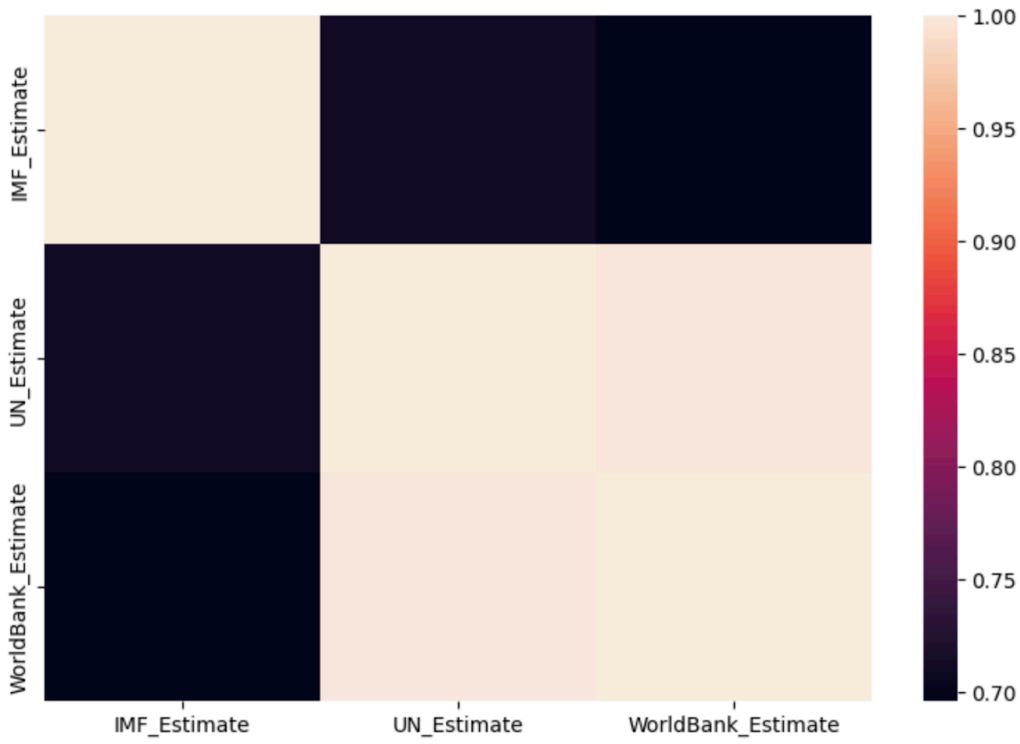
```
df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(bins=15, figsize=(15,12))

#23400/15 = 15300
plt.show()
```



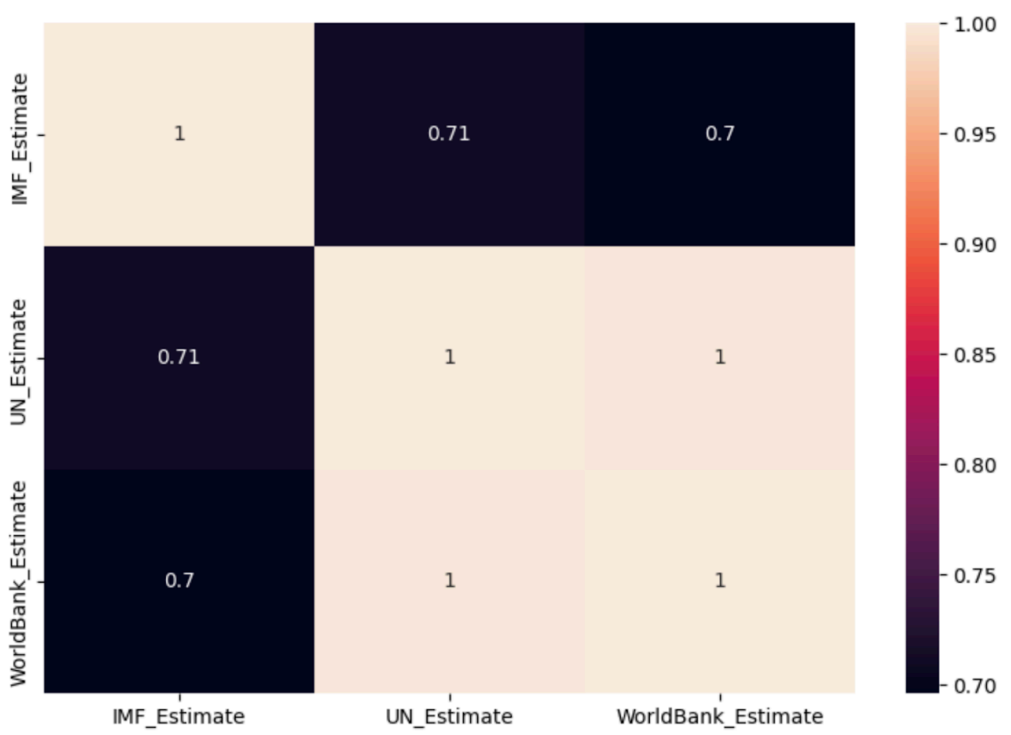
```
df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()
```

	IMF_Estimate	UN_Estimate	WorldBank_Estimate
IMF_Estimate	1.000000	0.709518	0.695935
UN_Estimate	0.709518	1.000000	0.998438
WorldBank_Estimate	0.695935	0.998438	1.000000

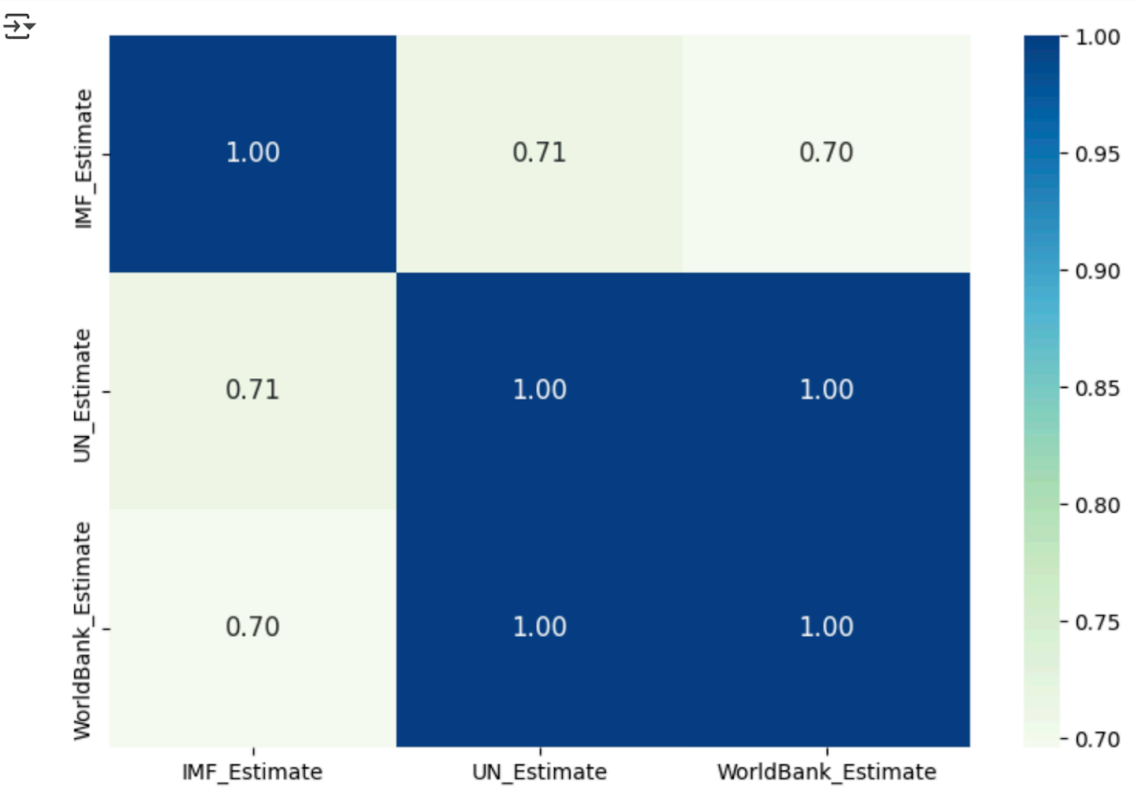
```
corr = df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()

plt.figure(figsize=(9,6))
sns.heatmap(corr)

plt.show()
```



```
corr = df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()

plt.figure(figsize=(9,6))

sns.heatmap(corr, annot=True)

plt.show()
```

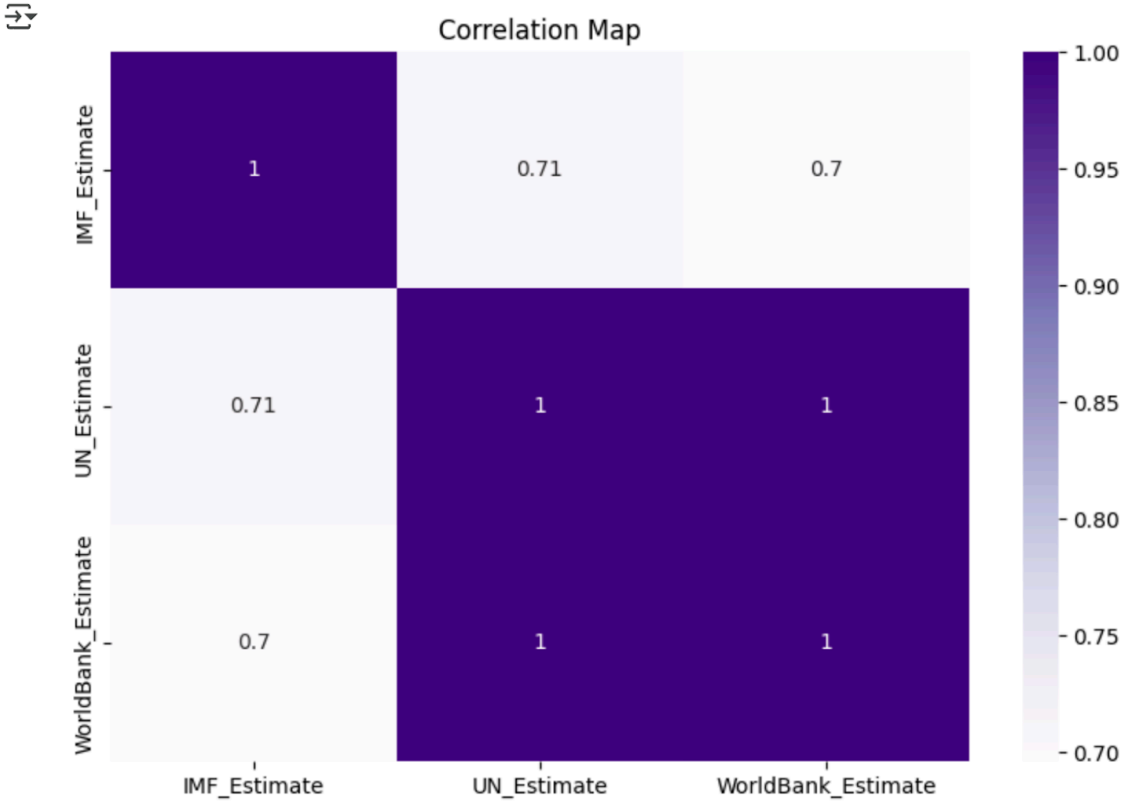


```
[158] corr = df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()
      plt.figure(figsize=(9,6))
      sns.heatmap(corr, annot=True, fmt=".2f", cmap = 'GnBu', annot_kws={"size": 12})
      plt.show()
```

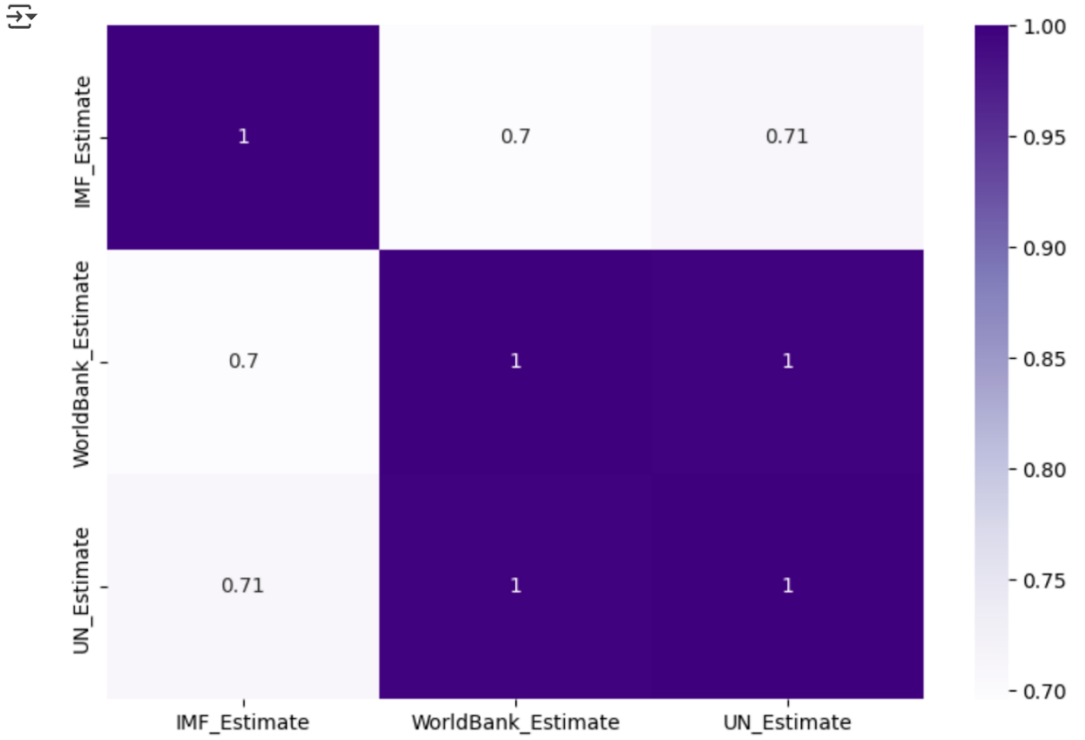



```
[159] corr = df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()
```

```
corr = df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()

plt.figure(figsize=(9,6))

sns.heatmap(corr, annot=True, cmap = 'Purples')

plt.title("Correlation Map")


plt.show()
```



```
corr = df.select_dtypes(include=[int, float]).corr()

plt.figure(figsize=(9,6))

sns.heatmap(corr, annot=True, cmap = 'Purples')

plt.show()
```

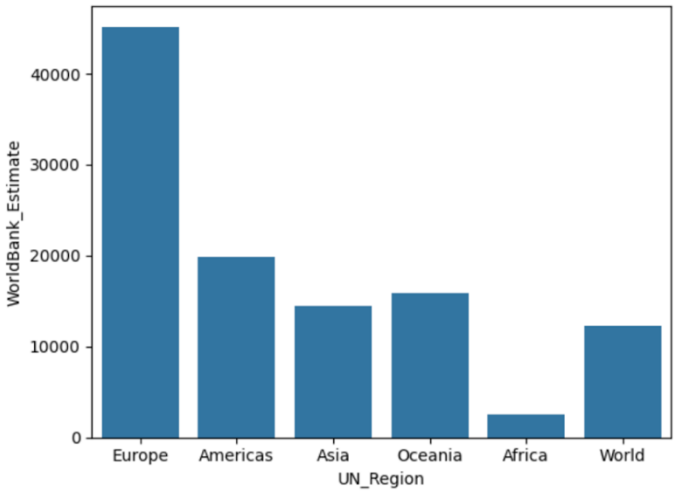


Bar plot

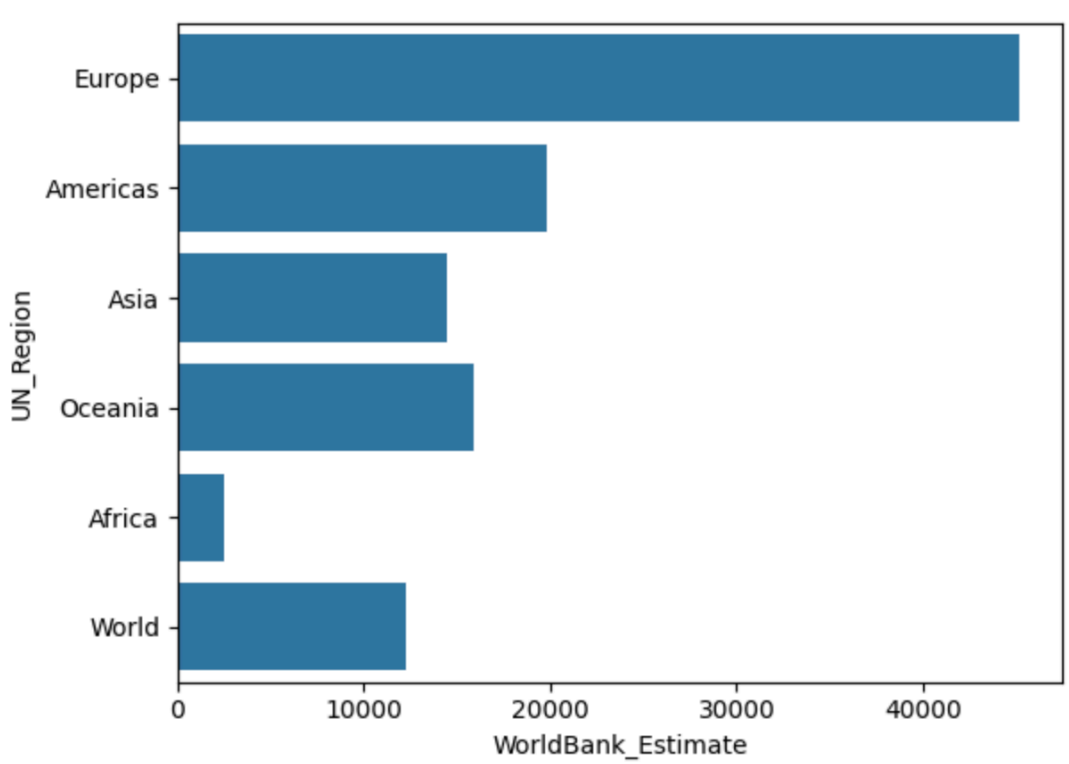
```
[161] df.head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
1	Monaco	Europe	17377.736041	<NA>	234316.0	2021	234317.0	2021
2	Liechtenstein	Europe	17377.736041	<NA>	157755.0	2020	169260.0	2021
3	Luxembourg	Europe	132372.000000	2023	133590.0	2021	133745.0	2021
4	Ireland	Europe	114581.000000	2023	100172.0	2021	101109.0	2021
5	Bermuda	Americas	17377.736041	<NA>	114090.0	2021	112653.0	2021

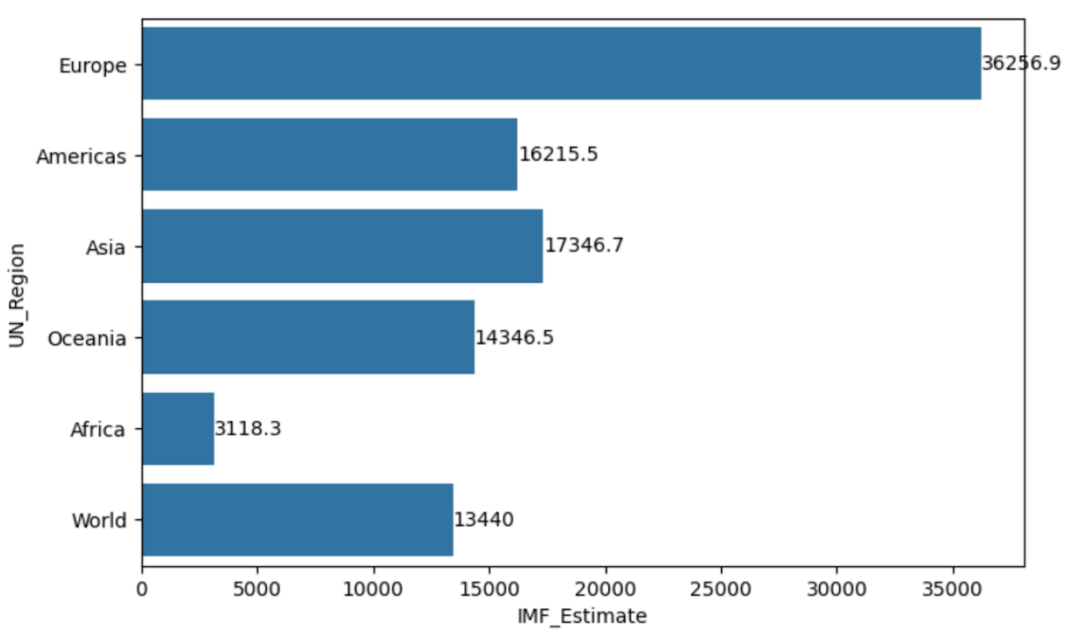
```
sns.barplot(x="UN_Region", y="WorldBank_Estimate", data=df, errorbar=None)

plt.show()
```



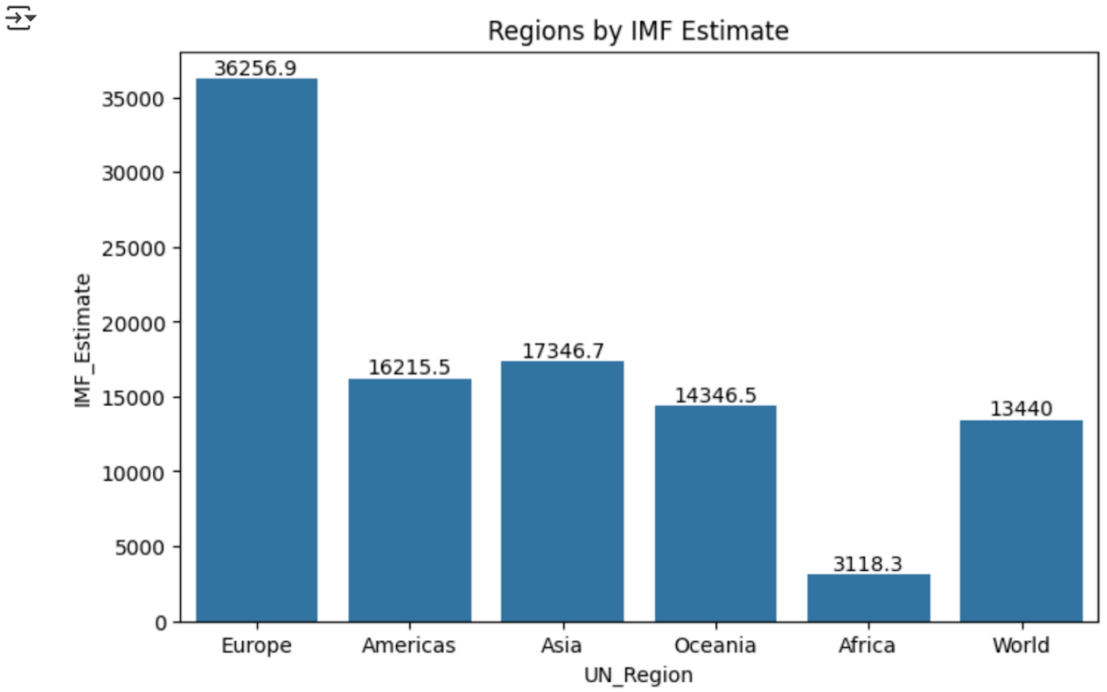
```
sns.barplot(x="WorldBank_Estimate", y="UN_Region", data=df, errorbar=None)
plt.show()
```



```
fig = plt.figure(figsize = (8,5))

ax = sns.barplot(x = "IMF_Estimate",  y = "UN_Region",
data = df, errorbar = None)

ax.bar_label(ax.containers[0])

plt.show()
```

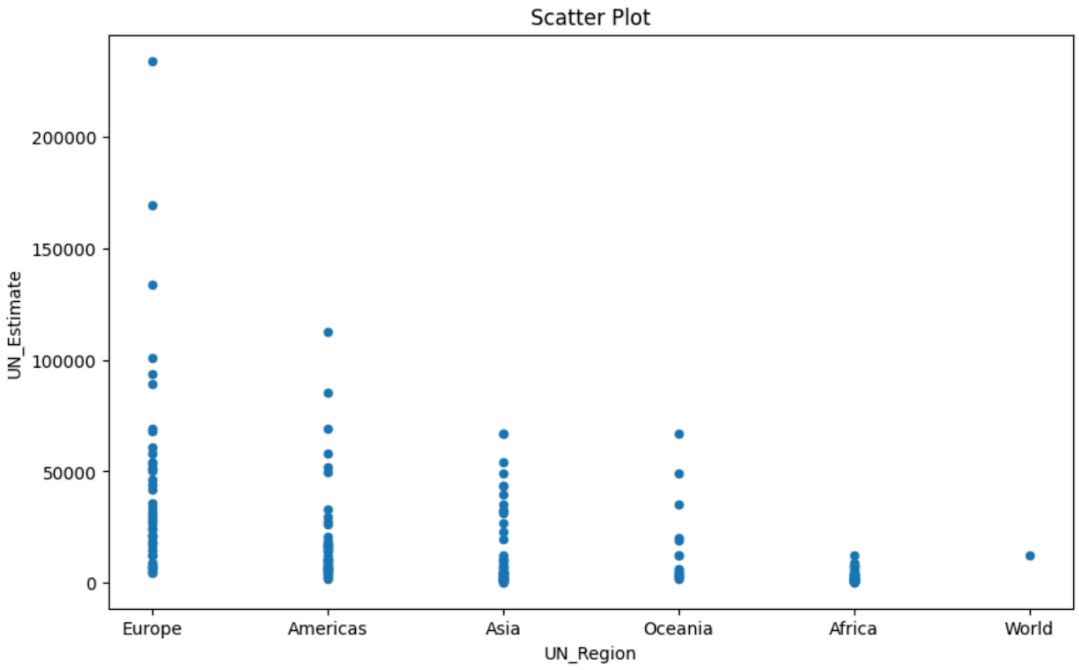


```
[165] fig = plt.figure(figsize = (8,5))
      ax = sns.barplot(x = "UN_Region",  y = "IMF_Estimate",
                       data = df, errorbar = None)

      ax.bar_label(ax.containers[0])


      ax.set_title("Regions by IMF Estimate")
      plt.show()
```



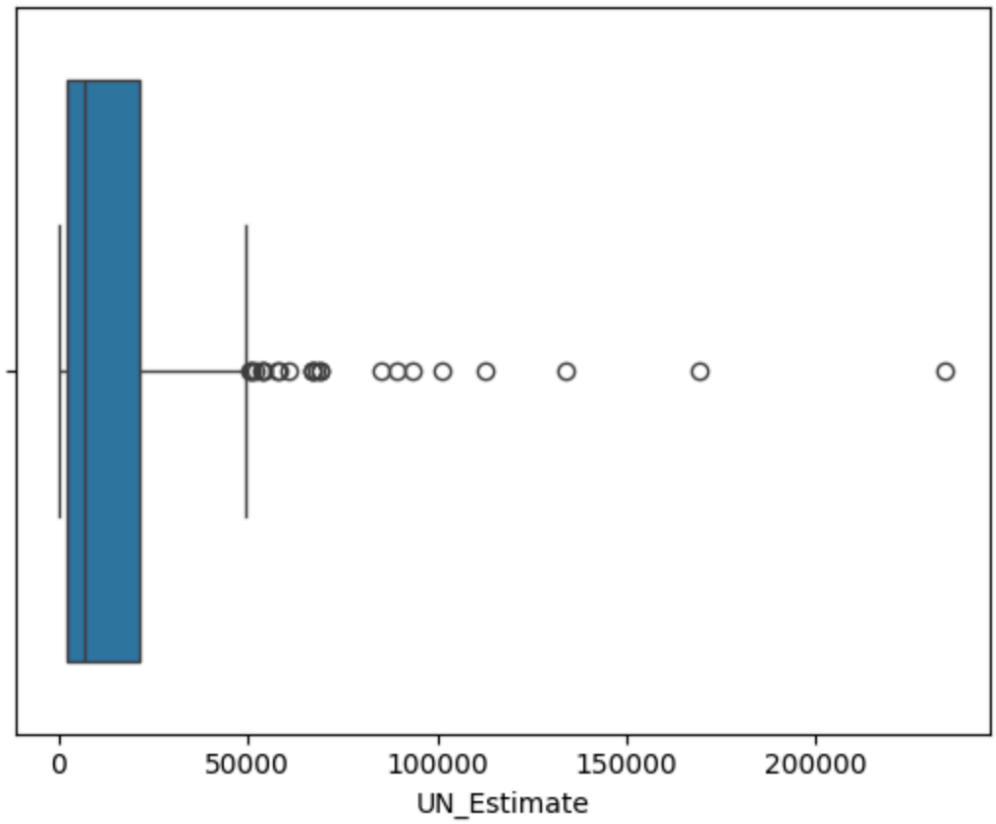
Scatter Plot

```
df.plot(x='UN_Region', y='UN_Estimate', kind='scatter',
        figsize=(10,6),
        title="Scatter Plot")

plt.show()
```



Boxplot and Outliers

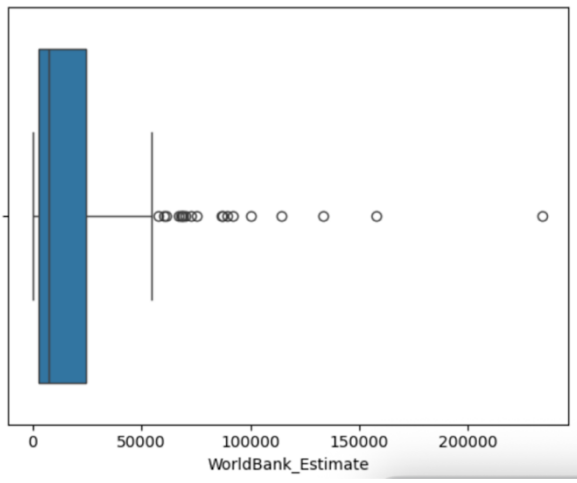
```
sns.boxplot(x=df["UN_Estimate"])

plt.show()
```



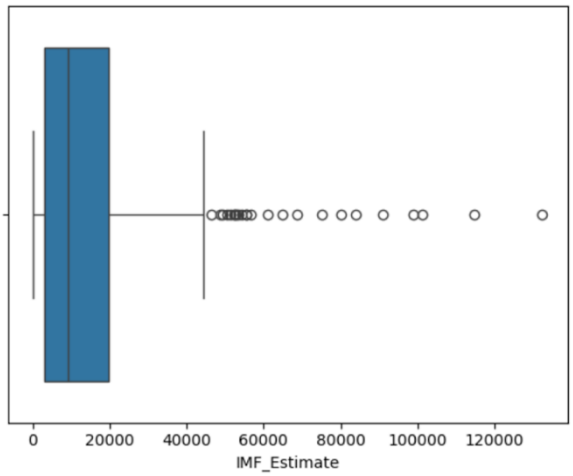
```
[168] df[df["UN_Estimate"]>50000].head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
1	Monaco	Europe	17377.736041	<NA>	234316.0	2021	234317.0	2021
2	Liechtenstein	Europe	17377.736041	<NA>	157755.0	2020	169260.0	2021
3	Luxembourg	Europe	132372.000000	2023	133590.0	2021	133745.0	2021
4	Ireland	Europe	114581.000000	2023	100172.0	2021	101109.0	2021
5	Bermuda	Americas	17377.736041	<NA>	114090.0	2021	112653.0	2021

```
sns.boxplot(x=df["WorldBank_Estimate"])

plt.show()
```




```
[170] sns.boxplot(x=df["IMF_Estimate"])

    plt.show()
```




```
[171] df[df["UN_Estimate"]>100000]
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
1	Monaco	Europe	17377.736041	<NA>	234316.0	2021	234317.0	2021
2	Liechtenstein	Europe	17377.736041	<NA>	157755.0	2020	169260.0	2021
3	Luxembourg	Europe	132372.000000	2023	133590.0	2021	133745.0	2021
4	Ireland	Europe	114581.000000	2023	100172.0	2021	101109.0	2021
5	Bermuda	Americas	17377.736041	<NA>	114090.0	2021	112653.0	2021

```
[171] df[df["UN_Estimate"]>100000]
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
1	Monaco	Europe	17377.736041	<NA>	234316.0	2021	234317.0	2021
2	Liechtenstein	Europe	17377.736041	<NA>	157755.0	2020	169260.0	2021
3	Luxembourg	Europe	132372.000000	2023	133590.0	2021	133745.0	2021
4	Ireland	Europe	114581.000000	2023	100172.0	2021	101109.0	2021
5	Bermuda	Americas	17377.736041	<NA>	114090.0	2021	112653.0	2021

```
[172] df.UN_Estimate.mean()
```

```
np.float64(18514.528037383177)
```

```
[173] df.shape
```

```
(223, 8)
```

```
[ ] Start coding or generate with AI.
```

Create another dataframe called data excluding 5 countries with highest UN estimate

```
[174] data = df[~(df["UN_Estimate"]>100000)]
```

```
data.head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
6	Norway	Europe	101103.000000	2023	89154.0	2021	89242.0	2021
7	Switzerland	Europe	98767.000000	2023	91992.0	2021	93525.0	2021
8	Singapore	Asia	91100.000000	2023	72794.0	2021	66822.0	2021
9	Isle of Man	Europe	17377.736041	<NA>	87158.0	2019	NaN	0
10	Cayman Islands	Americas	17377.736041	<NA>	86569.0	2021	85250.0	2021

```
[176] data.shape
```

```
(218, 8)
```

```
[177] data.UN_Estimate.mean()
```

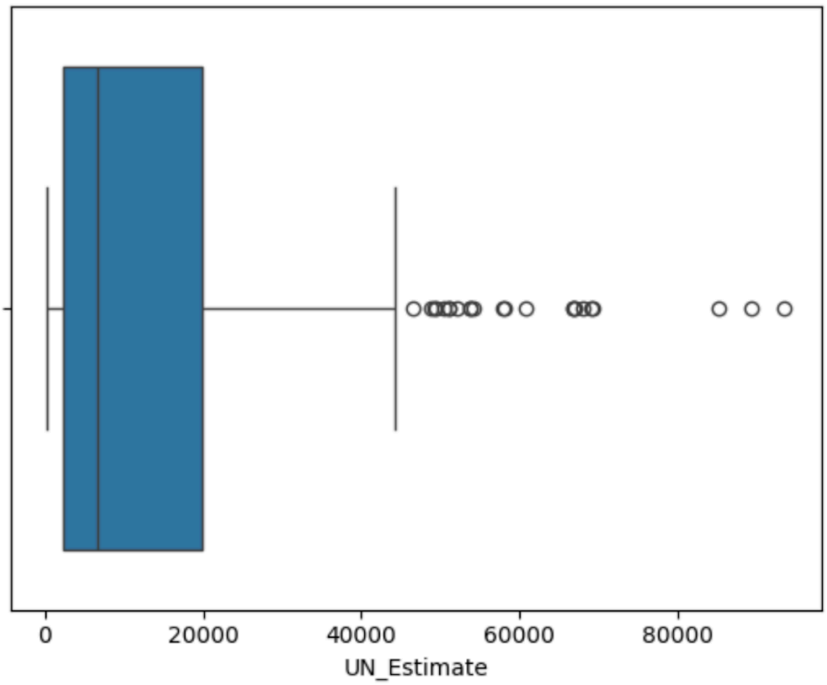
```
np.float64(15363.755980861244)
```

```
[178] df.UN_Estimate.mean()
```

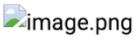
```
np.float64(18514.528037383177)
```

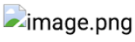
image.png

```
sns.boxplot(x=data["UN_Estimate"])
plt.show()
```



Removing outliers





```
[180] lower_q = df["UN_Estimate"].quantile(0.25)
      lower_q
```

```
np.float64(2331.75)
```

```
[181] higher_q = df["UN_Estimate"].quantile(0.75)
      higher_q
```

```
np.float64(21359.25)
```

```
[182] iqr = higher_q - lower_q
      iqr
```

```
np.float64(19027.5)
```

```
[183] upper_boundary = higher_q + 1.5 * iqr
      upper_boundary
```

```
np.float64(49900.5)
```

```
[184] lower_boundary = lower_q - 1.5 * iqr
      lower_boundary
```

```
np.float64(-26209.5)
```

```
[185] df_filtered = df[(df["UN_Estimate"] < upper_boundary) & (df["UN_Estimate"] > lower_boundary)]
```


```
[185] df_filtered = df[(df["UN_Estimate"] < upper_boundary) & (df["UN_Estimate"] > lower_boundary)]
```


```
[186] df_filtered.head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
27	Hong Kong	Asia	52429.000000	2023	49801.0	2021	49259.0	2021
29	Macau	Asia	50571.000000	2023	43874.0	2021	43555.0	2021
30	United Arab Emirates	Asia	49451.000000	2023	44316.0	2021	43295.0	2021
31	British Virgin Islands	Americas	17377.736041	<NA>	NaN	<NA>	49444.0	2021
32	New Zealand	Oceania	48826.000000	2023	48781.0	2021	48824.0	2021

```
[187] df_filtered.shape
      # there were 223 rows - 196 = 27 outliers dropped
```

```
(190, 8)
```

```
df_filtered.UN_Estimate.mean()
```

```
np.float64(10488.947368421053)
```

```
[189] df.UN_Estimate.mean()
```

```
np.float64(18514.528037383177)
```

```
[190] #how can we create a table with following
      df_filtered.WorldBank_Estimate.mean()
```

```
np.float64(10355.304347826086)
```

```
df.WorldBank_Estimate.mean()
```

```
np.float64(19540.805555555555)
```

```
[192] df_filtered.IMF_Estimate.mean()
```

```
np.float64(11636.771413304836)
```

```
[193] df.IMF_Estimate.mean()
```

```
np.float64(17377.736040609136)
```

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.