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

```
for x in range(1,101):
    if x%3 == 0 and x%5 ==0:
        print("The number is",x,",\tit is fizzbuzz.")
    elif x%5 == 0:
        print("The number is",x,",\tit is buzz.")
    elif x%3 == 0:
        print("The number is",x,",\tit is fizz.")
    else:
        print("The number is",x,",\tit is indivisible by 3 or 5.")
        continue
```

```
The number is 1 ,       it is indivisible by 3 or 5.
The number is 2 ,       it is indivisible by 3 or 5.
The number is 3 ,       it is fizz.
The number is 4 ,       it is indivisible by 3 or 5.
The number is 5 ,       it is buzz.
The number is 6 ,       it is fizz.
The number is 7 ,       it is indivisible by 3 or 5.
The number is 8 ,       it is indivisible by 3 or 5.
The number is 9 ,       it is fizz.
The number is 10 ,      it is buzz.
The number is 11 ,      it is indivisible by 3 or 5.
The number is 12 ,      it is fizz.
The number is 13 ,      it is indivisible by 3 or 5.
The number is 14 ,      it is indivisible by 3 or 5.
The number is 15 ,      it is fizzbuzz.
The number is 16 ,      it is indivisible by 3 or 5.
The number is 17 ,      it is indivisible by 3 or 5.
The number is 18 ,      it is fizz.
The number is 19 ,      it is indivisible by 3 or 5.
The number is 20 ,      it is 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."

```
df_student = pd.read_csv(r"D:\桌面\bootcamp\week 6\student.csv")
print("The first 5 rows of the DataFrame is:\n",df_student.head(5))
print("\n----------------------------------------\n")
print("The information of the DataFrame is:")
df_student.info()
print("\n----------------------------------------\n")
#print("The summary statistics for the DataFrame is:\n",df_student.describe())
print("The summary statistics for the DataFrame is:\n",df_student['mark'].describe()) #Focused on the summary statistics for
```

```
The first 5 rows of the DataFrame is:
    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

----------------------------------------


The information of the DataFrame is:
<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

----------------------------------------

The summary statistics for the DataFrame is:
 count    35.000000
mean     74.657143
std      16.401117
min      18.000000
25%      62.500000
50%      79.000000
75%      88.000000
max      96.000000
Name: mark, dtype: float64
```

Exercise 2: Indexing and Slicing

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

```python
df_student = pd.read_csv(r"D:\桌面\bootcamp\week 6\student.csv")
print("The 'name' column in the DataFrame is:\n",df_student['name'])
print("\n----------------------------------------\n")
print("The 'name' and 'mark' columns in the DataFrame are:\n",df_student[['name','mark']])
print("\n----------------------------------------\n")
print("The first 3 rows in the DataFrame are:\n",df_student.loc[:2])
print("The first 3 rows in the DataFrame are:\n",df_student.iloc[:3]) #Alternatively, using df.iloc to achieve the task
print("\n----------------------------------------\n")
print("Rows where the 'class' is 'Four' in the DataFrame are:\n",df_student[df_student['class']=="Four"])
```

```
The 'name' column in the DataFrame is:
 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

----------------------------------------

The 'name' and 'mark' columns in the DataFrame are:
            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
```

```
----------------------------------------

The first 3 rows in the DataFrame are:
    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
The first 3 rows in the DataFrame are:
    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

----------------------------------------

Rows where the 'class' is 'Four' in the DataFrame are:
     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."

```
:
1  df_student = pd.read_csv(r"D:\桌面\bootcamp\week 6\student.csv")
2  df_student['passed'] = df_student["mark"] >= 60
3  print("The students who passed the exam are marked as followed:\n",df_student)
4  print("\n----------------------------------------\n")
5  df_renamed = df_student.copy().rename(columns={'mark':'score'})
6  print("Renaming the 'mark' column to 'score' in the DataFame:\n",df_renamed)
7  print("\n----------------------------------------\n")
8  df_drop = df_renamed.drop(columns=['passed'])
9  print("Dropping the column named 'passed' in the DataFrame:\n",df_drop)

The students who passed the exam are marked as followed:
     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

 ----------------------------------------

 Renaming the 'mark' column to 'score' in the DataFame:
     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

----------------------------------------

  Dropping the column named 'passed' in the DataFrame:
     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."
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."

```python
df_student = pd.read_csv(r"D:\桌面\bootcamp\week 6\student.csv")
print(df_student.groupby('class', as_index=False)['mark'].mean().sort_values('mark',ascending = False))
print("\n----------------------------------------\n")
print(df_student.groupby('class',as_index=False).size().sort_values('size',ascending = False))
print("\n----------------------------------------\n")
#print(df_student.groupby('class',as_index=False)['id'].count().sort_values('id',ascending = False))
#print("\n----------------------------------------\n")
print(df_student.groupby('gender',as_index=False)['mark'].mean().sort_values('mark',ascending = False))
```

```
   class       mark
6    Six  82.571429
2   Five  80.000000
0  Eight  79.000000
1  Fifth  78.000000
5  Seven  77.600000
7  Three  73.666667
3   Four  68.750000
4   Nine  41.500000

----------------------------------------

   class  size
5  Seven    10
3   Four     8
6    Six     7
7  Three     3
2   Five     2
4   Nine     2
0  Eight     1
1  Fifth     1

----------------------------------------

   gender       mark
0  female  77.312500
1    male  71.588235
```

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."

```
df_student = pd.read_csv(r"D:\桌面\bootcamp\week 6\student.csv")
pd.pivot_table(df_student, index='class', columns='gender', values='mark',aggfunc='mean')
```

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

```
df_student = pd.read_csv(r"D:\桌面\bootcamp\week 6\student.csv")
def assign_grade(mark):
    if mark >= 85:
        return 'A'
    elif mark >= 70:
        return 'B'
    elif mark >= 60:
        return 'C'
    else:
        return 'D'

df_student['grade'] = df_student['mark'].apply(assign_grade)
df_student.sort_values('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
24	25	Giff Tow	Seven	88	male	A
15	16	Gimmy	Four	88	male	A
14	15	Tade Row	NaN	88	male	A
13	14	Bigy	Seven	88	female	A
12	13	Kty	Seven	88	female	A
34	35	Rows Noump	Six	88	female	A

Exercise 6: Exporting Data

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

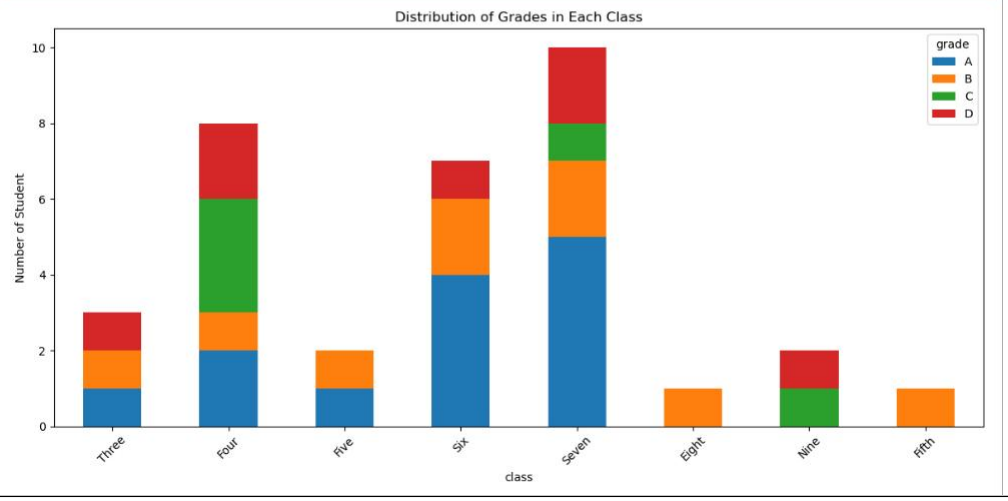
```
df_student.to_csv('students_with_grades.csv', index=False)
```

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 Sta	Four	60	female	C
5	John Mike	Four	60	female	C
6	Alex John	Four	55	male	D
7	My John R	Fifth	78	male	B
8	Asruid	Five	85	male	A
9	Tes Qry	Six	78		B
10	Big John	Four	55	female	D
11	Ronald	Six	89	female	A
12	Recky	Six	94	female	A
13	Kty	Seven	88	female	A
14	Bigy	Seven	88	female	A
15	Tade Row		88	male	A
16	Gimmy	Four	88	male	A
17	Tumyu	Six	54	male	D
18	Honny	Five	75	male	B
19	Tinny	Nine	18	male	D
20	Jackly	Nine	65	female	C
21	Babby Joh	Four	69	female	C
22	Reggid	Seven	55	female	D
23	Herod	Eight	79	male	B
24	Tiddy Now	Seven	78	male	B
25	Giff Tow	Seven	88	male	A
26	Crelea	Seven	79	male	B
27		Three	81		B

Exercise 7: If finished early try visualising the results

```
import matplotlib.pyplot as plt

df_student['class'] = pd.Categorical(df_student['class'],
                                     categories=['Three', 'Four', 'Five', 'Six', 'Seven', 'Eight', 'Nine','Fifth'],
                                     ordered=True)

pivot_table = df_student.pivot_table(values='id', index='class', columns='grade',aggfunc='count')

pivot_table.plot(kind='bar', stacked=True, figsize=(12, 6))
plt.title("Distribution of Grades in Each Class")
plt.ylabel("Number of Student")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
C:\Users\cxy\AppData\Local\Temp\ipykernel_17452\2745208492.py:7: FutureWarning: The default value of observed=False is deprecated and will change to observed=True in a future version of pandas. Specify observed=False to silence this warning and retain the current behavior
  pivot_table = df_student.pivot_table(values='id', index='class', columns='grade',aggfunc='count')
```



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

```python
df = pd.read_csv(r"D:\桌面\bootcamp\week 6\Day 4 Resources\GDP (nominal) per Capita.csv")
print("The fist 10 rows are:/n",df.head(10))
print("\n-----------------------------------\n")
print("The last 5 rows are:/n",df.tail(5))
print("\n-----------------------------------\n")
print("‘Country/Territory’ and ‘UN_Region’ columns shown as follow:\n",df[['Country/Territory','UN_Region']])
```

```
The fist 10 rows are:/n    Unnamed: 0 Country/Territory UN_Region  IMF_Estimate  IMF_Year  \
0           1            Monaco    Europe             0         0
1           2     Liechtenstein    Europe             0         0
2           3        Luxembourg    Europe        132372      2023
3           4           Ireland    Europe        114581      2023
4           5           Bermuda  Americas             0         0
5           6            Norway    Europe        101103      2023
6           7       Switzerland    Europe         98767      2023
7           8         Singapore      Asia         91100      2023
8           9       Isle of Man    Europe             0         0
9          10    Cayman Islands  Americas             0         0

   WorldBank_Estimate  WorldBank_Year  UN_Estimate UN_Year
0              234316            2021       234317    2021
1              157755            2020       169260    2021
2              133590            2021       133745    2021
3              100172            2021       101109    2021
4              114090            2021       112653    2021
5               89154            2021        89242    2021
6               91992            2021        93525    2021
7               72794            2021        66822    2021
8               87158            2019            0       0
9               86569            2021        85250    2021

-----------------------------------

The last 5 rows are:/n      Unnamed: 0 Country/Territory UN_Region  IMF_Estimate  IMF_Year  \
218         219            Malawi    Africa           496      2023
219         220       South Sudan    Africa           467      2023
220         221      Sierra Leone    Africa           415      2023
221         222       Afghanistan      Asia           611      2020
222         223           Burundi    Africa           249      2023

     WorldBank_Estimate  WorldBank_Year  UN_Estimate UN_Year
218                 635            2021          613    2021
219                1072            2015          400    2021
220                 480            2021          505    2021
221                 369            2021          373    2021
222                 222            2021          311    2021

-----------------------------------

‘Country/Territory’ and ‘UN_Region’ columns shown as follow:
     Country/Territory UN_Region
0              Monaco    Europe
1       Liechtenstein    Europe
2          Luxembourg    Europe
3             Ireland    Europe
4             Bermuda  Americas
..                ...       ...
218            Malawi    Africa
219       South Sudan    Africa
220      Sierra Leone    Africa
221       Afghanistan      Asia
222           Burundi    Africa

[223 rows 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.]

```
# number of countries per region

#df.groupby("UN_Region")["Country/Territory"].count()
#df.groupby("UN_Region")["Country/Territory"].size()
pd.pivot_table(df, index='UN_Region', values='Country/Territory',aggfunc='count')
```

UN_Region	Country/Territory
Africa	55
Americas	48
Asia	51
Europe	48
Oceania	20
World	1

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

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

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

```
# Countries in Europe below avarege

#Based on IMF Statistics
df_Europe_below_imfavg = df_Europe[df_Europe["IMF_Estimate"] < df_Europe["IMF_Estimate"].mean()]
df_Europe_below_imfavg["IMF_Estimate_Mean"] = df_Europe["IMF_Estimate"].mean()
print(df_Europe_below_imfavg[['Country/Territory','UN_Region','IMF_Estimate','IMF_Estimate_Mean']],"\n")

#Based on WorldBank Statistics
df_Europe_below_wbavg = df_Europe[df_Europe["WorldBank_Estimate"] < df_Europe["WorldBank_Estimate"].mean()]
df_Europe_below_wbavg["WorldBank_Estimate_Mean"] = df_Europe["WorldBank_Estimate"].mean()
print(df_Europe_below_wbavg[['Country/Territory','UN_Region','WorldBank_Estimate','WorldBank_Estimate_Mean']],"\n")

#Based Oon UN Statistics
df_Europe_below_unavg = df_Europe[df_Europe["UN_Estimate"] < df_Europe["UN_Estimate"].mean()]
df_Europe_below_unavg["UN_Estimate_Mean"] = df_Europe["UN_Estimate"].mean()
print(df_Europe_below_unavg[['Country/Territory','UN_Region','UN_Estimate','UN_Estimate_Mean']])

```

```
          Country/Territory UN_Region  IMF_Estimate  IMF_Estimate_Mean
1                    Monaco    Europe             0           34446.75
2             Liechtenstein    Europe             0           34446.75
9                Isle of Man    Europe             0           34446.75
14          Channel Islands    Europe             0           34446.75
15            Faroe Islands    Europe             0           34446.75
51                 Slovenia    Europe         32214           34446.75
52           Czech Republic    Europe         31368           34446.75
53                    Spain    Europe         31223           34446.75
54                  Estonia    Europe         31209           34446.75
57                Lithuania    Europe         28094           34446.75
59                 Portugal    Europe         26012           34446.75
60                   Latvia    Europe         25136           34446.75
62                 Slovakia    Europe         23457           34446.75
```

```python
## Which countries in Europe has higher GDP than UK?
df_uk = df_Europe[df_Europe["Country/Territory"] == 'United Kingdom']
#Based on IMF Statistics
uk_imf = df_uk['IMF_Estimate'].values[0]
df_country_aboveuk_imf = df_Europe[df_Europe["IMF_Estimate"]>uk_imf]
df_country_aboveuk_imf["UK_IMF_Estimate"] = uk_imf
print("Based on IMF Statistics,countries in Europe has higher GDP than UK are shown as follow:")
print(df_country_aboveuk_imf[['Country/Territory','UN_Region','IMF_Estimate','UK_IMF_Estimate']],"\n")
print("------------------------")

#Based on WorldBank Statistics
uk_wb = df_uk['WorldBank_Estimate'].values[0]
df_country_aboveuk_wb = df_Europe[df_Europe["WorldBank_Estimate"]>uk_wb]
df_country_aboveuk_wb["UK_WorldBank_Estimate"] = uk_wb
print("Based on WorldBank Statistics,countries in Europe has higher GDP than UK are shown as follow:")
print(df_country_aboveuk_wb[['Country/Territory','UN_Region','WorldBank_Estimate','UK_WorldBank_Estimate']],"\n")
print("------------------------")

#Based on UN Statistics
uk_un = df_uk['UN_Estimate'].values[0]
df_country_aboveuk_un = df_Europe[df_Europe["UN_Estimate"]>uk_un]
df_country_aboveuk_un["UK_UN_Estimate"] = uk_un
print("Based on UN Statistics,countries in Europe has higher GDP than UK are shown as follow:")
print(df_country_aboveuk_un[['Country/Territory','UN_Region','UN_Estimate','UK_UN_Estimate']],"\n")
print("------------------------")
```

```
Based on IMF Statistics,countries in Europe has higher GDP than UK are shown as follow:
   Country/Territory UN_Region  IMF_Estimate  UK_IMF_Estimate
3         Luxembourg    Europe        132372            46371
4            Ireland    Europe        114581            46371
6             Norway    Europe        101103            46371
7        Switzerland    Europe         98767            46371
```

Which countries below average by IMF world estimate?

```python
df_countries_below_imf = df[df['IMF_Estimate']<df['IMF_Estimate'].mean()]
df_countries_below_imf["IMF_world_avg_Estimate"] = df['IMF_Estimate'].mean()
df_countries_below_imf[['Country/Territory','IMF_Estimate','IMF_world_avg_Estimate']]
```

```
C:\Users\cxy\AppData\Local\Temp\ipykernel_35076\2372232952.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_countries_below_imf["IMF_world_avg_Estimate"] = df['IMF_Estimate'].mean()
```

	Country/Territory	IMF_Estimate	IMF_world_avg_Estimate
1	Monaco	0	15351.632287
2	Liechtenstein	0	15351.632287
5	Bermuda	0	15351.632287
9	Isle of Man	0	15351.632287
10	Cayman Islands	0	15351.632287
...	...	...	...
219	Malawi	496	15351.632287
220	South Sudan	467	15351.632287
221	Sierra Leone	415	15351.632287

IMF estimate 0 values

```python
df_imf_0 = df[(df['IMF_Estimate'] == 0) | (df['IMF_Year'] == 0)]
df_imf_0[['Country/Territory','UN_Region','IMF_Estimate','IMF_Year']]
```

5	Bermuda	Americas	0	0
9	Isle of Man	Europe	0	0
10	Cayman Islands	Americas	0	0
14	Channel Islands	Europe	0	0
15	Faroe Islands	Europe	0	0
19	Greenland	Americas	0	0
31	British Virgin Islands	Americas	0	0
37	US Virgin Islands	Americas	0	0
39	New Caledonia	Oceania	0	0
42	Guam	Oceania	0	0
58	Sint Maarten (Dutch part)	Americas	0	0
61	Northern Mariana Islands	Oceania	0	0
65	Saint Martin (French part)	Americas	0	0

Which country has highest UN Estimate?

```
highest_un = df.loc[df['UN_Estimate'].idxmax()]

# Display the country and the estimate
print("Country with highest UN Estimate:", highest_un['Country/Territory'])
print("UN Estimate value:", highest_un['UN_Estimate'])
```

```
Country with highest UN Estimate: Monaco
UN Estimate value: 234317
```

Which country has highest Worlbank Estimate?

```
highest_un = df.loc[df['WorldBank_Estimate'].idxmax()]

# Display the country and the estimate
print("Country with highest WorldBank Estimate:", highest_un['Country/Territory'])
print("WorldBank Estimate value:", highest_un['WorldBank_Estimate'])
```

```
Country with highest WorldBank Estimate: Monaco
WorldBank Estimate value: 234316
```

Which country has highest IMF Estimate?

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

# Display the country and the estimate
print("Country with highest IMF Estimate:", highest_un['Country/Territory'])
print("IMF Estimate value:", highest_un['IMF_Estimate'])
```

```
Country with highest IMF Estimate: Luxembourg
IMF Estimate value: 132372
```

```
# replace 0 with null values

import numpy as np

df.replace(0, np.nan, inplace=True)
df
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	UN_Year_Clean
1	Monaco	Europe	NaN	NaN	234316.0	2021.0	234317.0	2021	2021
2	Liechtenstein	Europe	NaN	NaN	157755.0	2020.0	169260.0	2021	2021
3	Luxembourg	Europe	132372.0	2023.0	133590.0	2021.0	133745.0	2021	2021
4	Ireland	Europe	114581.0	2023.0	100172.0	2021.0	101109.0	2021	2021
5	Bermuda	Americas	NaN	NaN	114090.0	2021.0	112653.0	2021	2021
...	...	...	...	...	...	...	...	...	...
219	Malawi	Africa	496.0	2023.0	635.0	2021.0	613.0	2021	2021
220	South Sudan	Africa	467.0	2023.0	1072.0	2015.0	400.0	2021	2021
221	Sierra Leone	Africa	415.0	2023.0	480.0	2021.0	505.0	2021	2021
222	Afghanistan	Asia	611.0	2020.0	369.0	2021.0	373.0	2021	2021
223	Burundi	Africa	249.0	2023.0	222.0	2021.0	311.0	2021	2021

223 rows × 9 columns

```
# Calculate the average of 'Worldbank_Estimate' and 'UN_Estimate' columns
avg_Worldbank_Estimate = df['WorldBank_Estimate'].mean()
print("The average of 'WorldBank_Estimate' column is:",avg_Worldbank_Estimate)
avg_UN_Estimate = df['UN_Estimate'].mean()
print("The average of 'UN_Estimate' column is:",avg_UN_Estimate)


df['avg_worldbank_un'] = df[['WorldBank_Estimate', 'UN_Estimate']].mean(axis=1)
df
```

```
The average of 'WorldBank_Estimate' column is: 19540.805555555555
The average of 'UN_Estimate' column is: 18514.528037383177
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	UN_Year_Clean	avg_worldbank_un
1	Monaco	Europe	17377.736041	2022	234316.0	2021.0	234317.0	2021	2021	234316.5
2	Liechtenstein	Europe	17377.736041	2022	157755.0	2020.0	169260.0	2021	2021	163507.5
3	Luxembourg	Europe	132372.000000	2023	133590.0	2021.0	133745.0	2021	2021	133667.5
4	Ireland	Europe	114581.000000	2023	100172.0	2021.0	101109.0	2021	2021	100640.5
5	Bermuda	Americas	17377.736041	2022	114090.0	2021.0	112653.0	2021	2021	113371.5
...	...	...	...	...	...	...	...	...	...	...
219	Malawi	Africa	496.000000	2023	635.0	2021.0	613.0	2021	2021	624.0
220	South Sudan	Africa	467.000000	2023	1072.0	2015.0	400.0	2021	2021	736.0
221	Sierra Leone	Africa	415.000000	2023	480.0	2021.0	505.0	2021	2021	492.5

```
# Fill the null values in 'imf' column with the calculated average
df['IMF_Estimate'] = df['IMF_Estimate'].fillna(df['IMF_Estimate'].mean())
imf_mean_year = round(df['IMF_Year'].mean())
df['IMF_Year'] = df['IMF_Year'].fillna(imf_mean_year).astype(int).astype(str)
df
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	UN_Year_Clean
1	Monaco	Europe	17377.736041	2022	234316.0	2021.0	234317.0	2021	2021
2	Liechtenstein	Europe	17377.736041	2022	157755.0	2020.0	169260.0	2021	2021
3	Luxembourg	Europe	132372.000000	2023	133590.0	2021.0	133745.0	2021	2021
4	Ireland	Europe	114581.000000	2023	100172.0	2021.0	101109.0	2021	2021
5	Bermuda	Americas	17377.736041	2022	114090.0	2021.0	112653.0	2021	2021
...	...	...	...	...	...	...	...	...	...
219	Malawi	Africa	496.000000	2023	635.0	2021.0	613.0	2021	2021
220	South Sudan	Africa	467.000000	2023	1072.0	2015.0	400.0	2021	2021
221	Sierra Leone	Africa	415.000000	2023	480.0	2021.0	505.0	2021	2021
222	Afghanistan	Asia	611.000000	2020	369.0	2021.0	373.0	2021	2021
223	Burundi	Africa	249.000000	2023	222.0	2021.0	311.0	2021	2021

223 rows × 9 columns

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

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	UN_Year_Clean
1	Monaco	Europe	17377.736041	2022	234316.0	2021.0	234317.0	2021	2021
2	Liechtenstein	Europe	17377.736041	2022	157755.0	2020.0	169260.0	2021	2021
3	Luxembourg	Europe	132372.000000	2023	133590.0	2021.0	133745.0	2021	2021
4	Ireland	Europe	114581.000000	2023	100172.0	2021.0	101109.0	2021	2021
5	Bermuda	Americas	17377.736041	2022	114090.0	2021.0	112653.0	2021	2021
...	...	...	...	...	...	...	...	...	...
219	Malawi	Africa	496.000000	2023	635.0	2021.0	613.0	2021	2021
220	South Sudan	Africa	467.000000	2023	1072.0	2015.0	400.0	2021	2021
221	Sierra Leone	Africa	415.000000	2023	480.0	2021.0	505.0	2021	2021
222	Afghanistan	Asia	611.000000	2020	369.0	2021.0	373.0	2021	2021
223	Burundi	Africa	249.000000	2023	222.0	2021.0	311.0	2021	2021

223 rows × 9 columns

Checking Missing Values

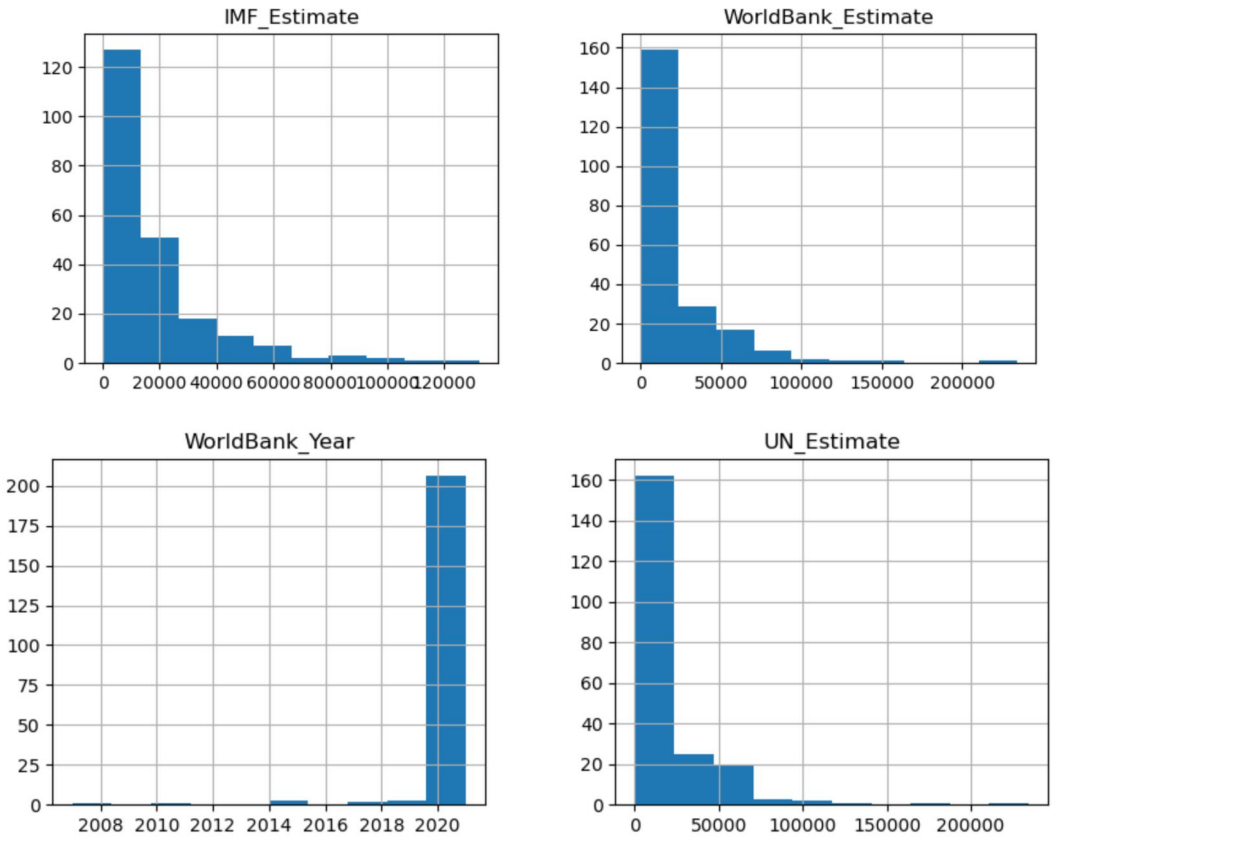
```
print(df.isnull())
print(df.isnull().sum())
```

```
     Country/Territory  UN_Region  IMF_Estimate  IMF_Year  WorldBank_Estimate  \
1                False      False         False     False               False
2                False      False         False     False               False
3                False      False         False     False               False
4                False      False         False     False               False
5                False      False         False     False               False
..                 ...        ...           ...       ...                 ...
219              False      False         False     False               False
220              False      False         False     False               False
221              False      False         False     False               False
222              False      False         False     False               False
223              False      False         False     False               False

     WorldBank_Year  UN_Estimate  UN_Year  UN_Year_Clean
1             False        False    False          False
2             False        False    False          False
3             False        False    False          False
4             False        False    False          False
5             False        False    False          False
..              ...          ...      ...            ...
219           False        False    False          False
220           False        False    False          False
221           False        False    False          False
222           False        False    False          False
223           False        False    False          False
```

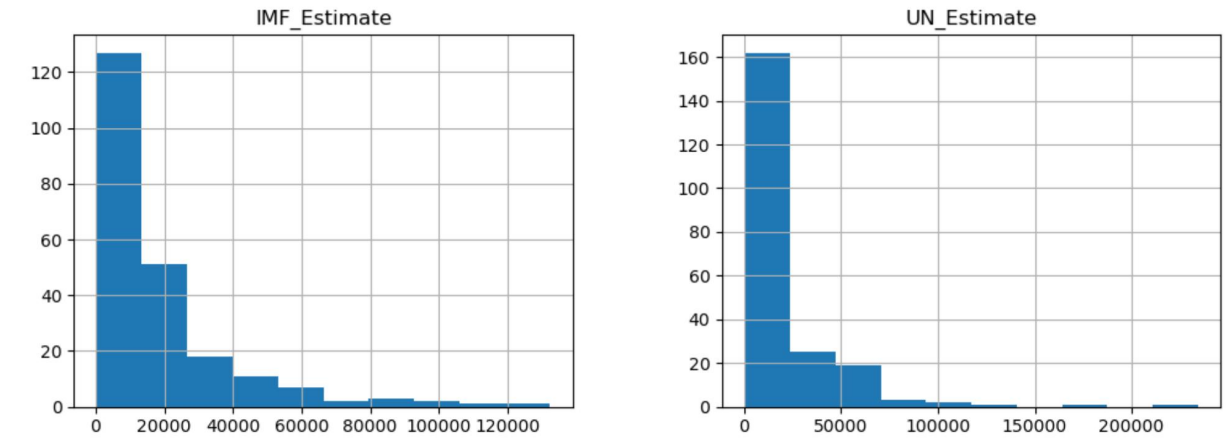
```
[223 rows x 9 columns]
Country/Territory     0
UN_Region             0
IMF_Estimate          0
IMF_Year              0
WorldBank_Estimate    7
WorldBank_Year        7
UN_Estimate           9
UN_Year               0
UN_Year_Clean         0
dtype: int64
```

Histogram

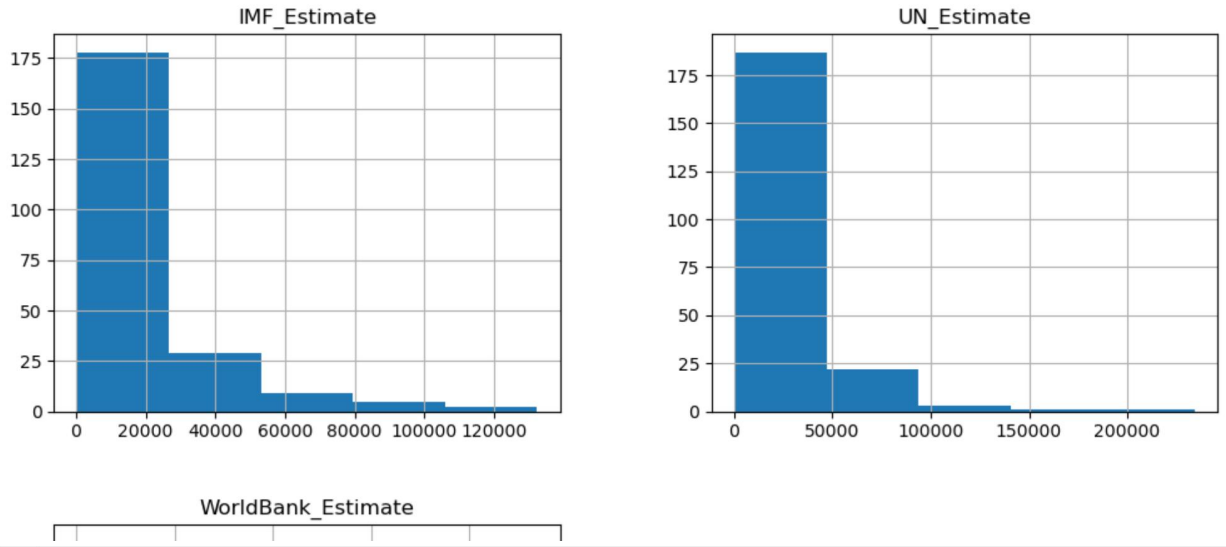
```
1 df.hist(figsize=(10,8))
2 plt.show()
```




```
1 df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(figsize=(12,9))
2
3 plt.show()
```



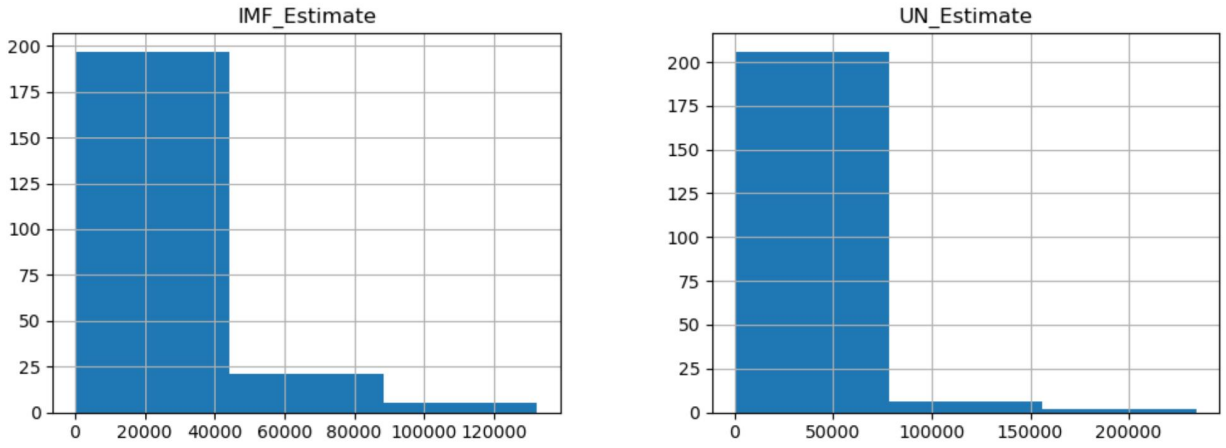
```
df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(bins=5, figsize=(12,9))

plt.show()
```



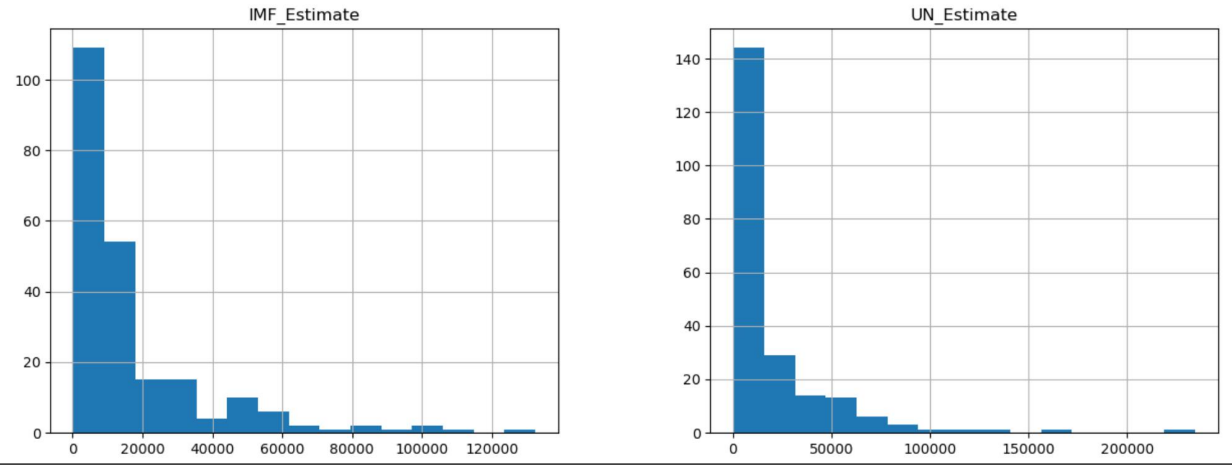

```
df["WorldBank_Estimate"].agg(["min","max"])
```

```
min       222.0
max    234316.0
Name: WorldBank_Estimate, dtype: float64
```

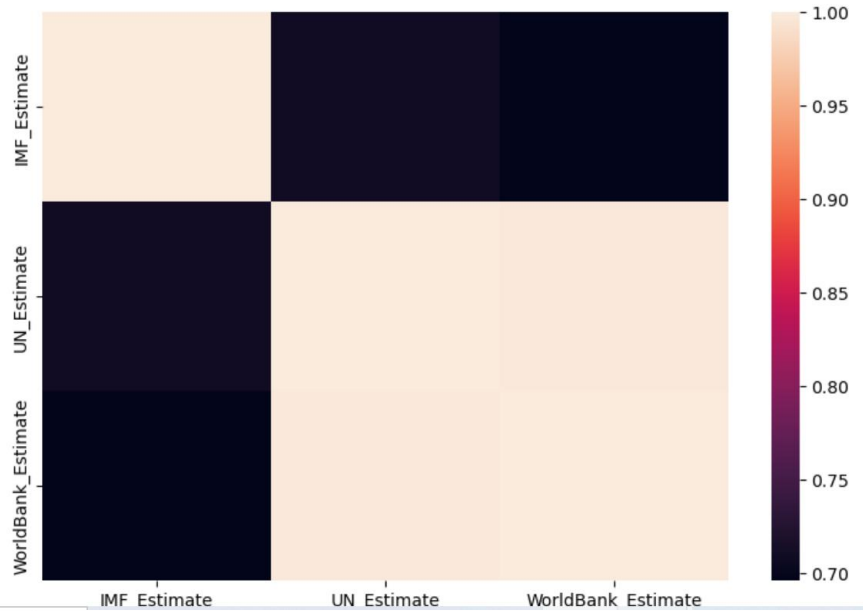
```
df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(bins=3, figsize=(12,9))

plt.show()
```



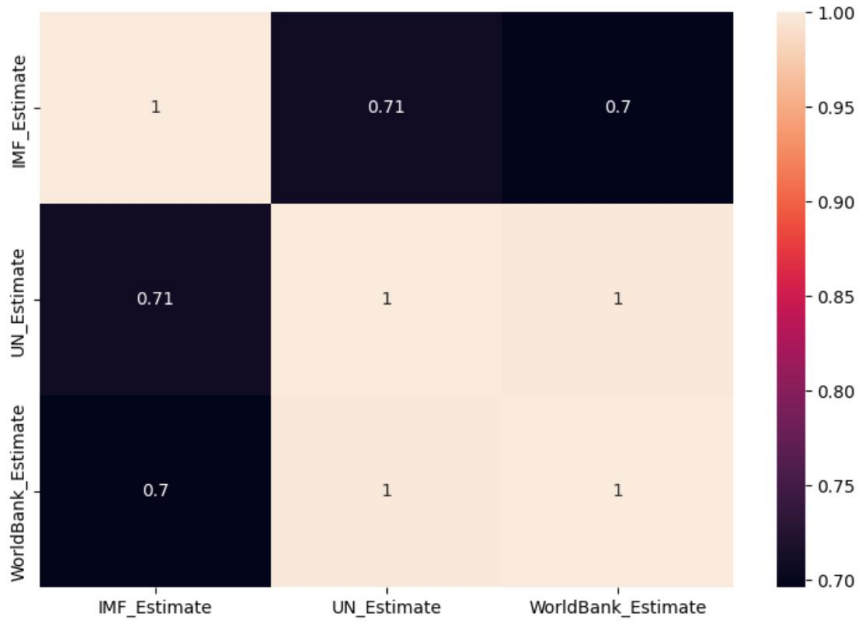

```
df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(bins=15, figsize=(15,12))

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



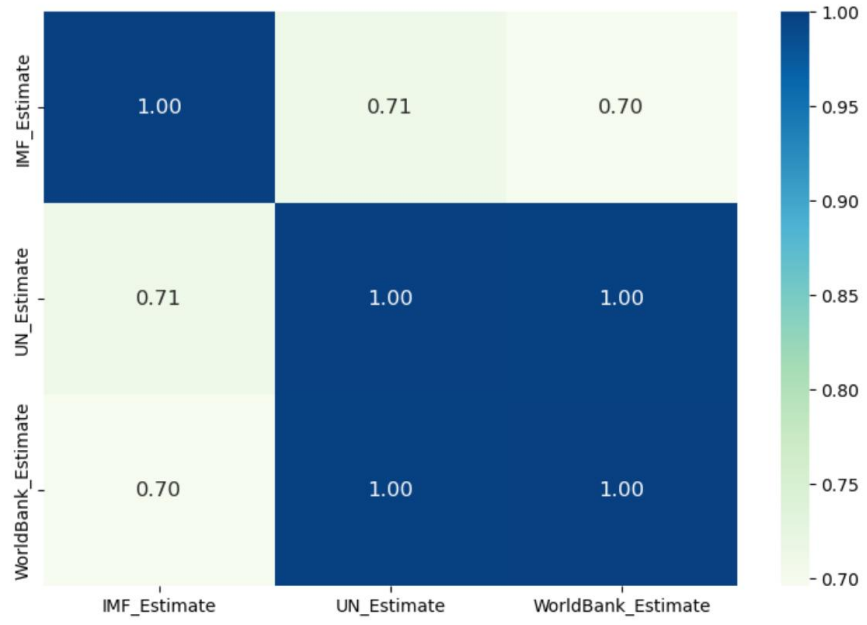
```
corr = df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()

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

plt.show()
```



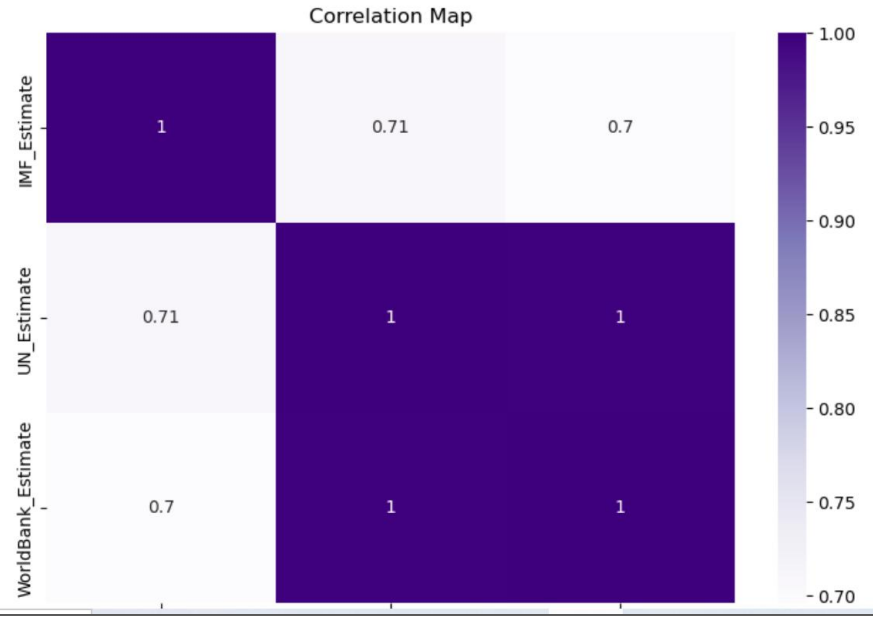

```
corr = df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()

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

sns.heatmap(corr, annot=True)

plt.show()
```



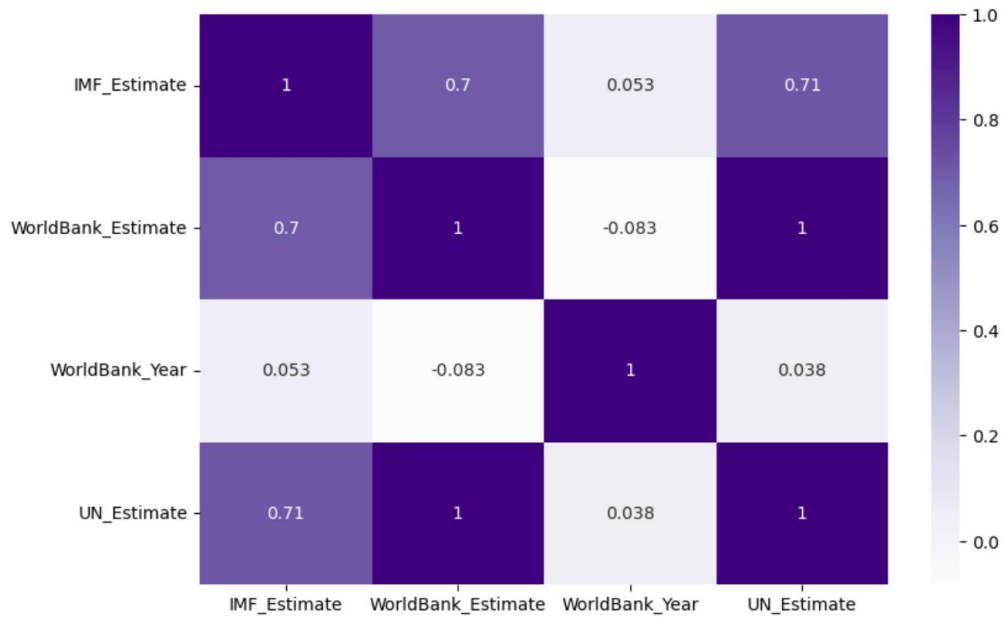
```python
corr = df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()

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

sns.heatmap(corr, annot=True, fmt=".2f", cmap = 'GnBu', annot_kws={"size": 12})

plt.show()
```




```python
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()
```



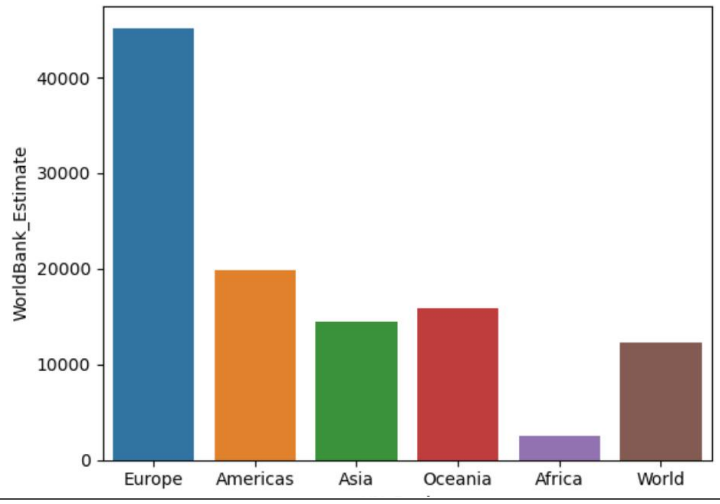
```
corr = df.select_dtypes(include=[int, float]).corr()

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

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

plt.show()
```



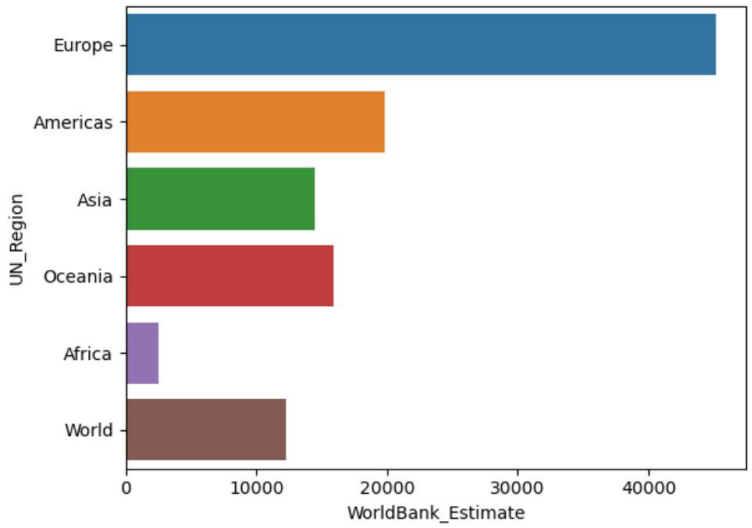

```
df.head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	UN_Year_Clean
1	Monaco	Europe	17377.736041	2022	234316.0	2021.0	234317.0	2021	2021
2	Liechtenstein	Europe	17377.736041	2022	157755.0	2020.0	169260.0	2021	2021
3	Luxembourg	Europe	132372.000000	2023	133590.0	2021.0	133745.0	2021	2021
4	Ireland	Europe	114581.000000	2023	100172.0	2021.0	101109.0	2021	2021
5	Bermuda	Americas	17377.736041	2022	114090.0	2021.0	112653.0	2021	2021

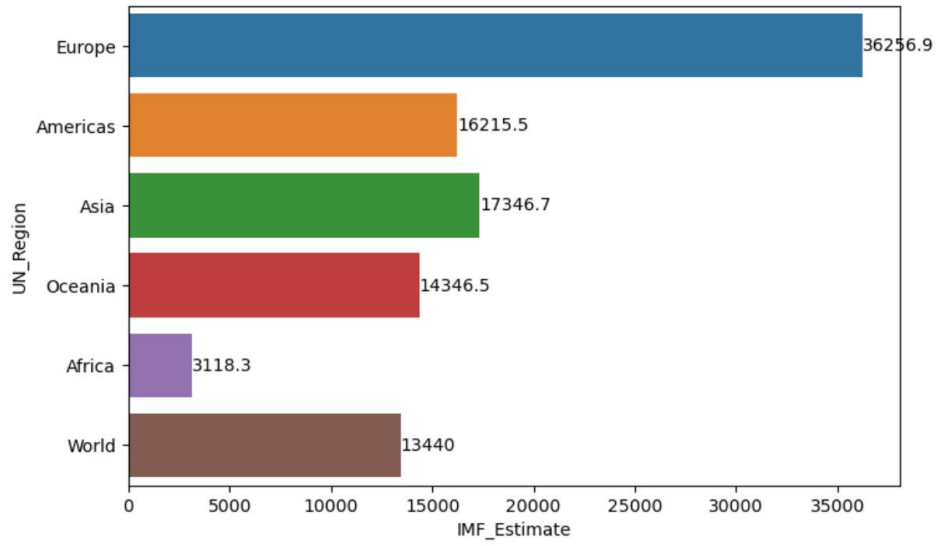
```
sns.barplot(x="UN_Region", y="WorldBank_Estimate", data=df, errorbar=None)

plt.show()
```



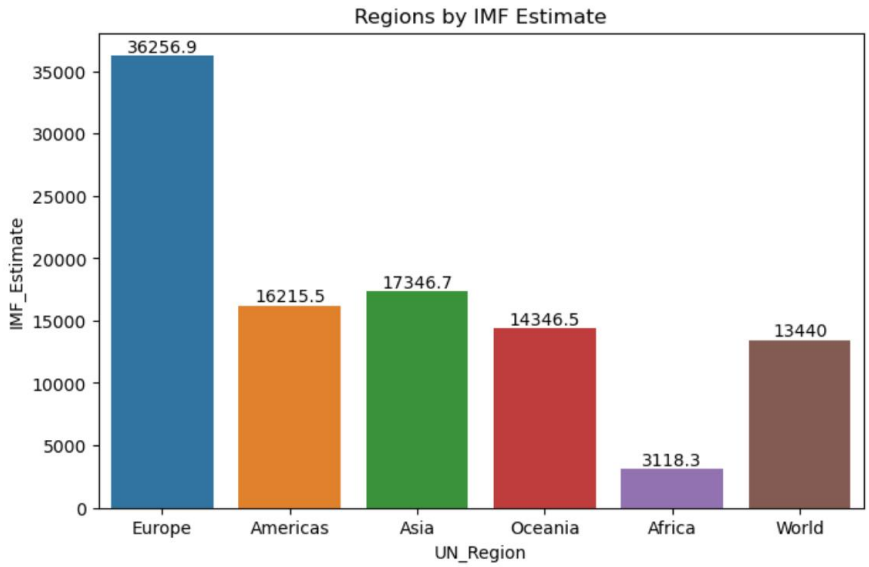
```
sns.barplot(x="WorldBank_Estimate", y="UN_Region", data=df, errorbar=None)

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()
```

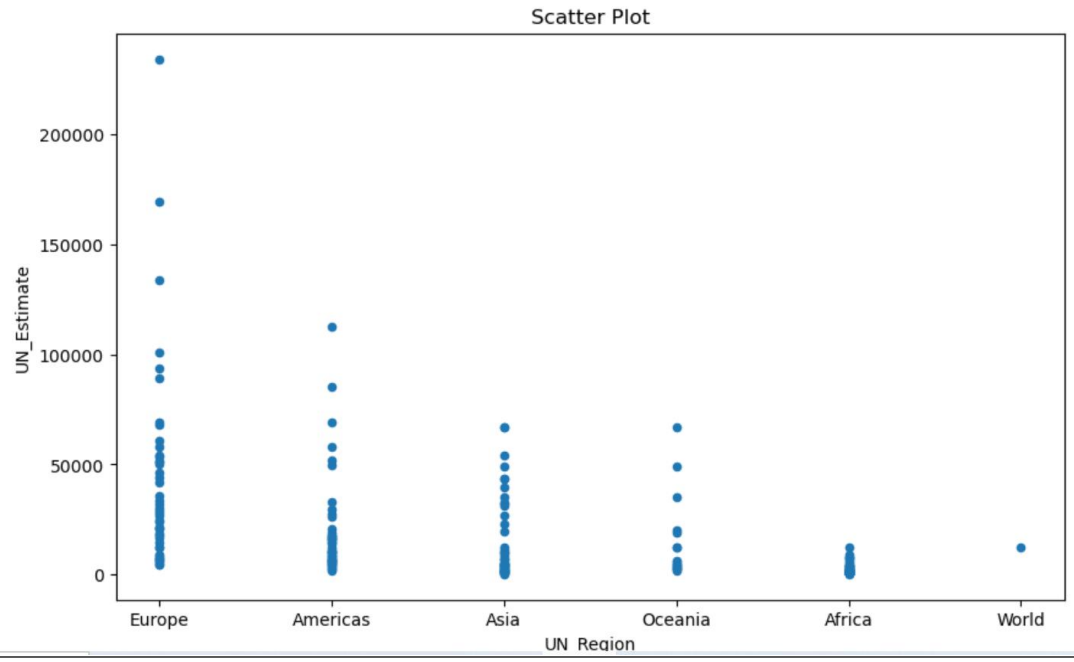


```
fig = plt.figure(figsize = (8,5))
ax = sns.barplot(x = "UN_Region",  y = "IMF_Estimate",
                 data = df, errorbar = None)

ax.bar_label(ax.containers[0])


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

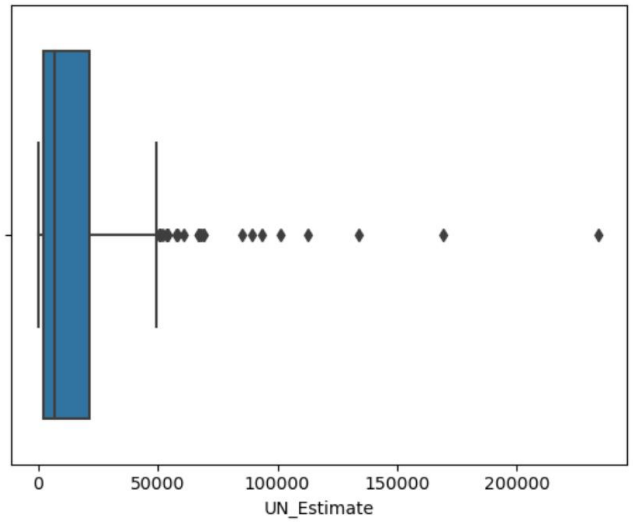


Scatter Plot

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

plt.show()
```



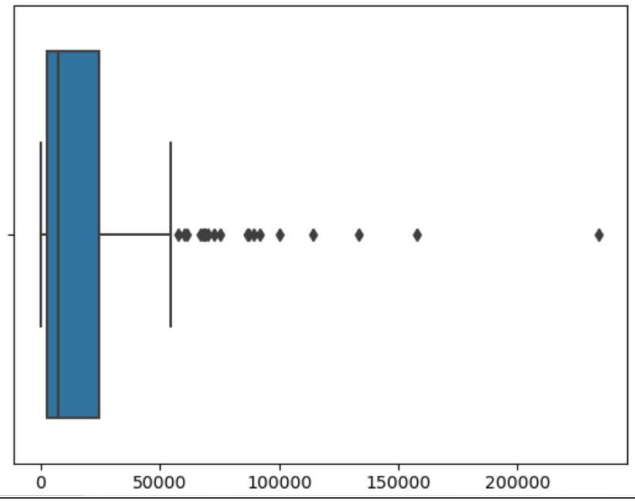
```
sns.boxplot(x=df["UN_Estimate"])

plt.show()
```



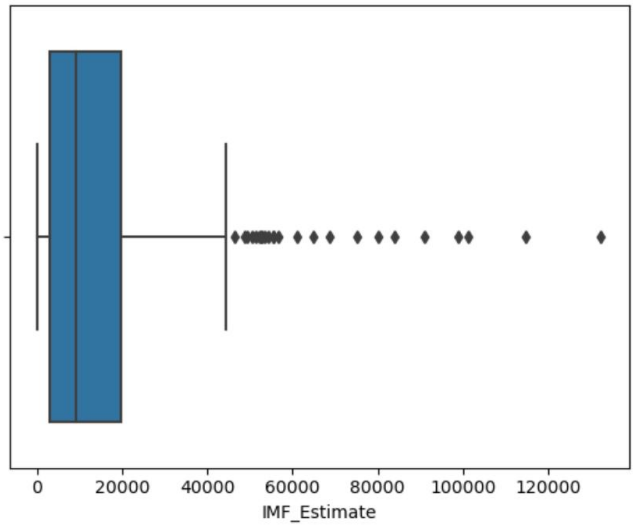

```
df[df["UN_Estimate"]>50000].head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	UN_Year_Clean
1	Monaco	Europe	17377.736041	2022	234316.0	2021.0	234317.0	2021	2021
2	Liechtenstein	Europe	17377.736041	2022	157755.0	2020.0	169260.0	2021	2021
3	Luxembourg	Europe	132372.000000	2023	133590.0	2021.0	133745.0	2021	2021
4	Ireland	Europe	114581.000000	2023	100172.0	2021.0	101109.0	2021	2021
5	Bermuda	Americas	17377.736041	2022	114090.0	2021.0	112653.0	2021	2021

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

plt.show()
```



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

plt.show()
```




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

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	UN_Year_Clean
1	Monaco	Europe	17377.736041	2022	234316.0	2021.0	234317.0	2021	2021
2	Liechtenstein	Europe	17377.736041	2022	157755.0	2020.0	169260.0	2021	2021
3	Luxembourg	Europe	132372.000000	2023	133590.0	2021.0	133745.0	2021	2021
4	Ireland	Europe	114581.000000	2023	100172.0	2021.0	101109.0	2021	2021
5	Bermuda	Americas	17377.736041	2022	114090.0	2021.0	112653.0	2021	2021

```
df.UN_Estimate.mean()
```

```
18514.528037383177
```

```
df.shape
```

```
(223, 9)
```

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

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

```
data.head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	UN_Year_Clean
6	Norway	Europe	101103.000000	2023	89154.0	2021.0	89242.0	2021	2021
7	Switzerland	Europe	98767.000000	2023	91992.0	2021.0	93525.0	2021	2021
8	Singapore	Asia	91100.000000	2023	72794.0	2021.0	66822.0	2021	2021
9	Isle of Man	Europe	17377.736041	2022	87158.0	2019.0	NaN	0	0
10	Cayman Islands	Americas	17377.736041	2022	86569.0	2021.0	85250.0	2021	2021

```
data.shape
```

```
(218, 9)
```

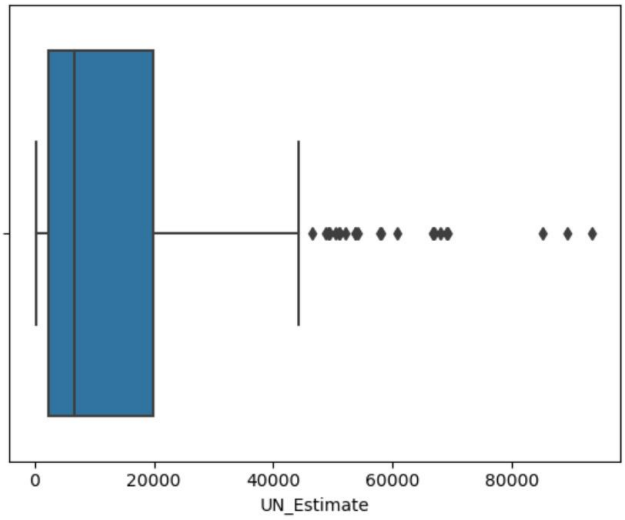
```
data.UN_Estimate.mean()
```

```
15363.755980861244
```

```
df.UN_Estimate.mean()
```

```
18514.528037383177
```

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




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

```
2331.75
```

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

```
21359.25
```

```
iqr = higher_q - lower_q
iqr
```

```
19027.5
```

```
upper_boundary = higher_q + 1.5 * iqr
upper_boundary
```

```
49900.5
```

```
lower_boundary = lower_q - 1.5 * iqr
lower_boundary
```

```
-26209.5
```

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

```
df_filtered.head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	UN_Year_Clean
27	Hong Kong	Asia	52429.000000	2023	49801.0	2021.0	49259.0	2021	2021
29	Macau	Asia	50571.000000	2023	43874.0	2021.0	43555.0	2021	2021
30	United Arab Emirates	Asia	49451.000000	2023	44316.0	2021.0	43295.0	2021	2021
31	British Virgin Islands	Americas	17377.736041	2022	NaN	NaN	49444.0	2021	2021
32	New Zealand	Oceania	48826.000000	2023	48781.0	2021.0	48824.0	2021	2021

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

```
(190, 9)
```

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

```
10488.947368421053
```

```
df.UN_Estimate.mean()
```

```
18514.528037383177
```

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

```
10355.304347826086
```

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

```
19540.805555555555
```

```
1 df_filtered.IMF_Estimate.mean()
```

```
11636.771413304836
```

```
1 df.IMF_Estimate.mean()
```

```
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.