Pandas, Matplotlib and Seaborn

```
import pandas as pd
```

Dataset Soruce

CONTENT

The figures presented here do not take into account differences in the cost of living in different countries, and the results vary greatly from one year to another based on fluctuations in the exchange rates of the country's currency. Such fluctuations change a country's ranking from one year to the next, even though they often make little or no difference to the standard of living of its population.

GDP per capita is often considered an indicator of a country's standard of living; however, this is inaccurate because GDP per capita is not a measure of personal income.

Comparisons of national income are also frequently made on the basis of purchasing power parity (PPP), to adjust for differences in the cost of living in different countries. (See List of countries by GDP (PPP) per capita.) PPP largely removes the exchange rate problem but not others; it does not reflect the value of economic output in international trade, and it also requires more estimation than GDP per capita. On the whole, PPP per capita figures are more narrowly spread than nominal GDP per capita figures.

Here are some resources to learn about GDP:

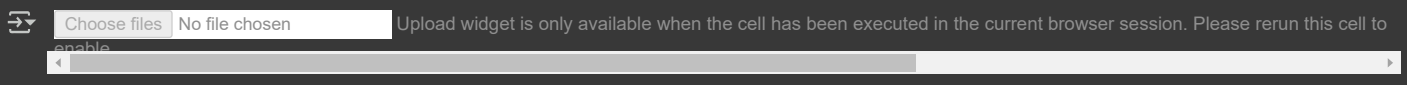
World Bank

Our World in Data

IMF

UN Data

```
from google.colab import files
uploaded = files.upload()
```




```
df = pd.read_csv("GDP (nominal) per Capita.csv",encoding= 'unicode_escape',  index_col=0)
```

EDA (Exploratory Data Analysis)

Use this section to explore and inspect dataset.

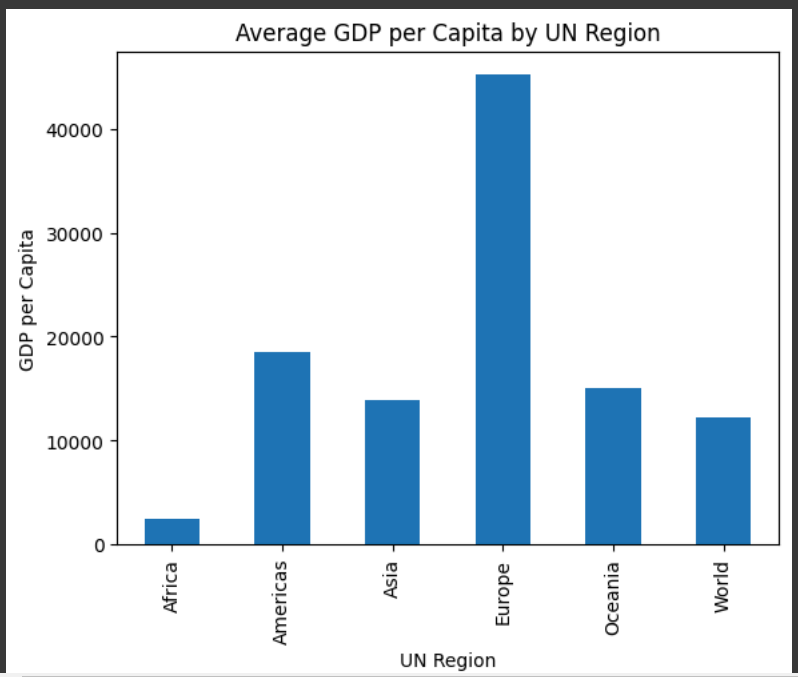
```
print (df)
```

```
     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
..                 ...       ...           ...       ...                 ...
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
1              2021       234317    2021
2              2020       169260    2021
3              2021       133745    2021
4              2021       101109    2021
5              2021       112653    2021
..              ...          ...     ...
219            2021          613    2021
220            2015          400    2021
221            2021          505    2021
222            2021          373    2021
223            2021          311    2021
```

```
[223 rows x 8 columns]
```

```
from __future__ import print_function

# Display unique UN regions
print(df['UN_Region'].unique())


# Group by UN region and get the mean GDP per capita
un_region_gdp = df.groupby('UN_Region')['WorldBank_Estimate'].mean()

# Display the result
print(un_region_gdp)
print()
# You can further process or visualize this data as needed
# Example visualization (requires matplotlib):
import matplotlib.pyplot as plt
un_region_gdp.plot(kind='bar')
plt.title('Average GDP per Capita by UN Region')
plt.xlabel('UN Region')
plt.ylabel('GDP per Capita')
plt.show()
```

```
['Europe' 'Americas' 'Asia' 'Oceania' 'Africa' 'World']
UN_Region
Africa       2470.836364
Americas    18565.125000
Asia        13921.313725
Europe      45193.687500
Oceania     15113.650000
World       12235.000000
Name: WorldBank_Estimate, dtype: float64
```




```
# Group data by 'UN_Region' and get the size of each group (number of countries)
region_counts = df.groupby('UN_Region').size()
# Print the results, showing the count of countries per region
print(region_counts)
```

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

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

```

# Search for "European Union[n 1]" in the 'Country/Territory' column
european_union_rows = df.loc[df['Country/Territory'] == "European Union[n 1]"]
```

```
# Display the rows where "European Union[n 1]" was found
print(european_union_rows)
```

```
       Country/Territory UN_Region  IMF_Estimate  IMF_Year  WorldBank_Estimate  \
36  European Union[n 1]    Europe         39940      2023               38411

    WorldBank_Year  UN_Estimate UN_Year
36            2021        31875    2021
```

```
# Countries in Europe below avarege
```

```
# Filter for European countries (excluding 'European Union[n 1]')
europe_df = df[(df['UN_Region'] == 'Europe') & (df['Country/Territory'] != 'European Union[n 1]')]

# Calculate average GDP for Europe (excluding 'European Union[n 1]')
average_gdp_europe = europe_df['WorldBank_Estimate'].mean()

# Filter for countries below average
below_average_countries = europe_df[europe_df['WorldBank_Estimate'] < average_gdp_europe]

# Display the result
print(below_average_countries[['Country/Territory', 'WorldBank_Estimate']])
```

```
           Country/Territory  WorldBank_Estimate
25               San Marino               45320
34                   France               43659
35                  Andorra               42137
40                    Malta               33487
41                    Italy               35658
51                 Slovenia               29291
52           Czech Republic               26821
53                    Spain               30104
54                  Estonia               27944
57                Lithuania               23723
59                 Portugal               24568
60                   Latvia               21148
62                 Slovakia               21392
63                   Greece               20193
70                  Croatia               17685
72                   Poland               18000
75                  Hungary               18728
78                  Romania               14858
87                 Bulgaria               12222
90                   Russia               12195
103              Montenegro                9466
106                  Serbia                9230
112  Bosnia and Herzegovina                7143
115                 Belarus                7302
118         North Macedonia                6695
120                 Albania                6493
127                 Moldova                5231
133                  Kosovo                5270
143                 Ukraine                4836
```

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

```
# 1. Filter for European countries
europe_df = df[df['UN_Region'] == 'Europe']

# 2. Get UK's GDP (using 'WorldBank_Estimate' as an example)
uk_gdp = europe_df[europe_df['Country/Territory'] == 'United Kingdom']['WorldBank_Estimate'].values[0]

# 3. Compare GDP values and filter for countries with higher GDP than the UK
higher_gdp_countries_df = europe_df[europe_df['WorldBank_Estimate'] > uk_gdp]

# 4. Display the result (Country/Territory and WorldBank_Estimate)
print(higher_gdp_countries_df[['Country/Territory', 'WorldBank_Estimate']])
```

```
    Country/Territory  WorldBank_Estimate
1              Monaco              234316
2      Liechtenstein              157755
3         Luxembourg              133590
4            Ireland              100172
6             Norway               89154
7        Switzerland               91992
9        Isle of Man               87158
13           Iceland               68728
14   Channel Islands               75153
15     Faroe Islands               69010
16           Denmark               68008
18       Netherlands               57768
20           Austria               53638
```

```
22            Sweden               61029
23           Finland               53655
24           Belgium               51247
28           Germany               51204
```

groupby()

Learn more about groupby

```
# Group data by 'UN_Region' and calculate the mean of 'WorldBank_Estimate' for each region
mean_gdp_by_region = df.groupby('UN_Region')['WorldBank_Estimate'].mean()
print("Mean GDP by Region:\n", mean_gdp_by_region)

# Group data by 'UN_Region' and get the number of countries in each region
country_count_by_region = df.groupby('UN_Region').size()
print("\nNumber of Countries per Region:\n", country_count_by_region)

# Group data by 'UN_Region' and find the maximum 'WorldBank_Estimate' for each region
max_gdp_by_region = df.groupby('UN_Region')['WorldBank_Estimate'].max()
print("\nMaximum GDP by Region:\n", max_gdp_by_region)

# Group by 'UN_Region' and apply multiple aggregation functions
# Calculate the mean, median, and standard deviation of 'WorldBank_Estimate' for each region
agg_gdp_by_region = df.groupby('UN_Region')['WorldBank_Estimate'].agg(['mean', 'median', 'std'])
print("\nAggregated GDP statistics by Region:\n", agg_gdp_by_region)
```

```
Mean GDP by Region:
 UN_Region
Africa       2470.836364
Americas    18565.125000
Asia        13921.313725
Europe      45193.687500
Oceania     15113.650000
World       12235.000000
Name: WorldBank_Estimate, dtype: float64

Number of Countries per Region:
 UN_Region
Africa      55
Americas    48
Asia        51
Europe      48
Oceania     20
World        1
dtype: int64

Maximum GDP by Region:
 UN_Region
Africa       14653
Americas    114090
Asia         72794
Europe      234316
Oceania      60443
World        12235
Name: WorldBank_Estimate, dtype: int64

Aggregated GDP statistics by Region:
                    mean   median           std
UN_Region
Africa      2470.836364   1319.0   2772.447680
Americas   18565.125000  10022.5  22769.886210
Asia       13921.313725   4566.0  18403.393872
Europe     45193.687500  31795.5  43984.130016
Oceania    15113.650000   5902.0  17416.040076
World      12235.000000  12235.0           NaN
```

Which countries below average by IMF world estimate?

```
# prompt: Which countries below average by IMF world estimate?

# Calculate the average IMF estimate
average_imf_estimate = df['IMF_Estimate'].mean()

# Filter countries with IMF estimates below the average
below_average_countries = df[df['IMF_Estimate'] < average_imf_estimate]

# Display the countries below average by IMF estimate
```

```
print(below_average_countries[['Country/Territory', 'IMF_Estimate']])
```

```
     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]
```

IMF estimate 0 values

```
# prompt: IMF estimate 0 values

# Count the number of countries with an IMF estimate of 0
zero_imf_estimate_count = len(df[df['IMF_Estimate'] == 0])

# Print the result
print(f"Number of countries with an IMF estimate of 0: {zero_imf_estimate_count}")

# Display the countries with an IMF estimate of 0
zero_imf_estimate_countries = df[df['IMF_Estimate'] == 0]
print(zero_imf_estimate_countries[['Country/Territory', 'IMF_Estimate']])
```

```
Number of countries with an IMF estimate of 0: 26
                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?

```
# prompt: Which country has highest UN Estimate?

# Find the country with the highest UN estimate
highest_un_estimate_country = df.loc[df['UN_Estimate'].idxmax()]

# Display the country and its UN estimate
print(highest_un_estimate_country[['Country/Territory', 'UN_Estimate']])
```

```
Country/Territory    Monaco
UN_Estimate          234317
Name: 1, dtype: object
```

Which country has highest Worlbank Estimate?

```
# prompt: Which country has highest Worlbank Estimate?

# Find the country with the highest World Bank estimate
highest_worldbank_estimate_country = df.loc[df['WorldBank_Estimate'].idxmax()]

# Display the country and its World Bank estimate
print(highest_worldbank_estimate_country[['Country/Territory', 'WorldBank_Estimate']])
```

```
Country/Territory     Monaco
WorldBank_Estimate    234316
Name: 1, dtype: object
```

Which country has highest IMF Estimate?

```
# prompt: Which country has highest IMF Estimate?

# Find the country with the highest IMF estimate
highest_imf_estimate_country = df.loc[df['IMF_Estimate'].idxmax()]

# Display the country and its IMF estimate
print(highest_imf_estimate_country[['Country/Territory', 'IMF_Estimate']])
```

```
Country/Territory    Luxembourg
IMF_Estimate             132372
Name: 3, dtype: object
```

```
# replace 0 with null values
```

```
# prompt: replace 0 with null values

# Replace 0 values in 'IMF_Estimate' column with NaN
df['IMF_Estimate'] = df['IMF_Estimate'].replace(0, np.nan)

# You can verify the change:
print(df[df['IMF_Estimate'].isnull()])
```

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

     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            0       0
10                86569            2021        85250    2021
14                75153            2007            0       0
15                69010            2021            0       0
19                54571            2020        58185    2021
31                    0               0        49444    2021
37                39552            2020            0       0
39                37160            2021        34994    2021
42                35905            2021            0       0
58                28988            2018        26199    2021
61                23707            2019            0       0
65                21921            2014            0       0
```

```
68                20909            2021        20909    2021
71                19915            2021        19915    2021
76                    0               0        19264    2021
77                    0               0        19216    2021
82                17718            2021        14183    2021
85                    0               0        16199    2021
86                15743            2021            0       0
104                9500            2020        11255    2021
196                   0               0         1211    2021
204                 533            2020          925    2021
212                   0               0          654    2021
```

```
# Calculate the average of 'Worldbank_Estimate' and 'UN_Estimate' columns
```

```
# Calculate the average of 'Worldbank_Estimate' and 'UN_Estimate'
df['Average_Estimate'] = (df['WorldBank_Estimate'] + df['UN_Estimate']) / 2

# Display the DataFrame with the new 'Average_Estimate' column
df
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	Average_Estimat
1	Monaco	Europe	234316.5	0	234316	2021	234317	2021	234316.
2	Liechtenstein	Europe	163507.5	0	157755	2020	169260	2021	163507.
3	Luxembourg	Europe	132372.0	2023	133590	2021	133745	2021	133667.
4	Ireland	Europe	114581.0	2023	100172	2021	101109	2021	100640.
5	Bermuda	Americas	113371.5	0	114090	2021	112653	2021	113371.
...	...	...	...	...	...	...	...	...	.
219	Malawi	Africa	496.0	2023	635	2021	613	2021	624.
220	South Sudan	Africa	467.0	2023	1072	2015	400	2021	736.
221	Sierra Leone	Africa	415.0	2023	480	2021	505	2021	492.
222	Afghanistan	Asia	611.0	2020	369	2021	373	2021	371.
223	Burundi	Africa	249.0	2023	222	2021	311	2021	266.

223 rows × 9 columns

```
# Fill the null values in 'imf' column with the calculated average
```

```
# Calculate the average of 'Worldbank_Estimate' and 'UN_Estimate' where 'IMF_Estimate' is null
average_estimate_for_null_imf = df.loc[df['IMF_Estimate'].isnull(), ['WorldBank_Estimate', 'UN_Estimate']].mean(axis=1)

# Fill the null values in 'IMF_Estimate' column with the calculated average
df['IMF_Estimate'].fillna(average_estimate_for_null_imf, inplace=True)

# Display the updated DataFrame to verify the changes
df
```

```
<ipython-input-74-7f795d7d0c38>:5: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col

  df['IMF_Estimate'].fillna(average_estimate_for_null_imf, inplace=True)
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	Average_Estimat
1	Monaco	Europe	234316.5	0	234316	2021	234317	2021	234316.
2	Liechtenstein	Europe	163507.5	0	157755	2020	169260	2021	163507.
3	Luxembourg	Europe	132372.0	2023	133590	2021	133745	2021	133667.
4	Ireland	Europe	114581.0	2023	100172	2021	101109	2021	100640.
5	Bermuda	Americas	113371.5	0	114090	2021	112653	2021	113371.
...	...	...	...	...	...	...	...	...	.
219	Malawi	Africa	496.0	2023	635	2021	613	2021	624.
220	South Sudan	Africa	467.0	2023	1072	2015	400	2021	736.
221	Sierra Leone	Africa	415.0	2023	480	2021	505	2021	492.
222	Afghanistan	Asia	611.0	2020	369	2021	373	2021	371.
223	Burundi	Africa	249.0	2023	222	2021	311	2021	266.

223 rows × 9 columns

```
# Drop the temporary 'Average_Estimate' column if not needed
```

```
# prompt: Drop the temporary  'Average_Estimate' column with column

df = df.drop(columns=['Average_Estimate'])
df
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
1	Monaco	Europe	234316.5	0	234316	2021	234317	2021
2	Liechtenstein	Europe	163507.5	0	157755	2020	169260	2021
3	Luxembourg	Europe	132372.0	2023	133590	2021	133745	2021
4	Ireland	Europe	114581.0	2023	100172	2021	101109	2021
5	Bermuda	Americas	113371.5	0	114090	2021	112653	2021
...	...	...	...	...	...	...	...	...
219	Malawi	Africa	496.0	2023	635	2021	613	2021
220	South Sudan	Africa	467.0	2023	1072	2015	400	2021
221	Sierra Leone	Africa	415.0	2023	480	2021	505	2021
222	Afghanistan	Asia	611.0	2020	369	2021	373	2021
223	Burundi	Africa	249.0	2023	222	2021	311	2021

223 rows × 8 columns

Visit this link to learn more about ffill

Visit this link to learn more about bfill

Checking Missing Values

```
# prompt: Checking Missing Values

# Check for missing values in the entire DataFrame
print(df.isnull().sum())
```

```
Country/Territory     0
UN_Region             0
IMF_Estimate          0
IMF_Year              0
WorldBank_Estimate    0
```

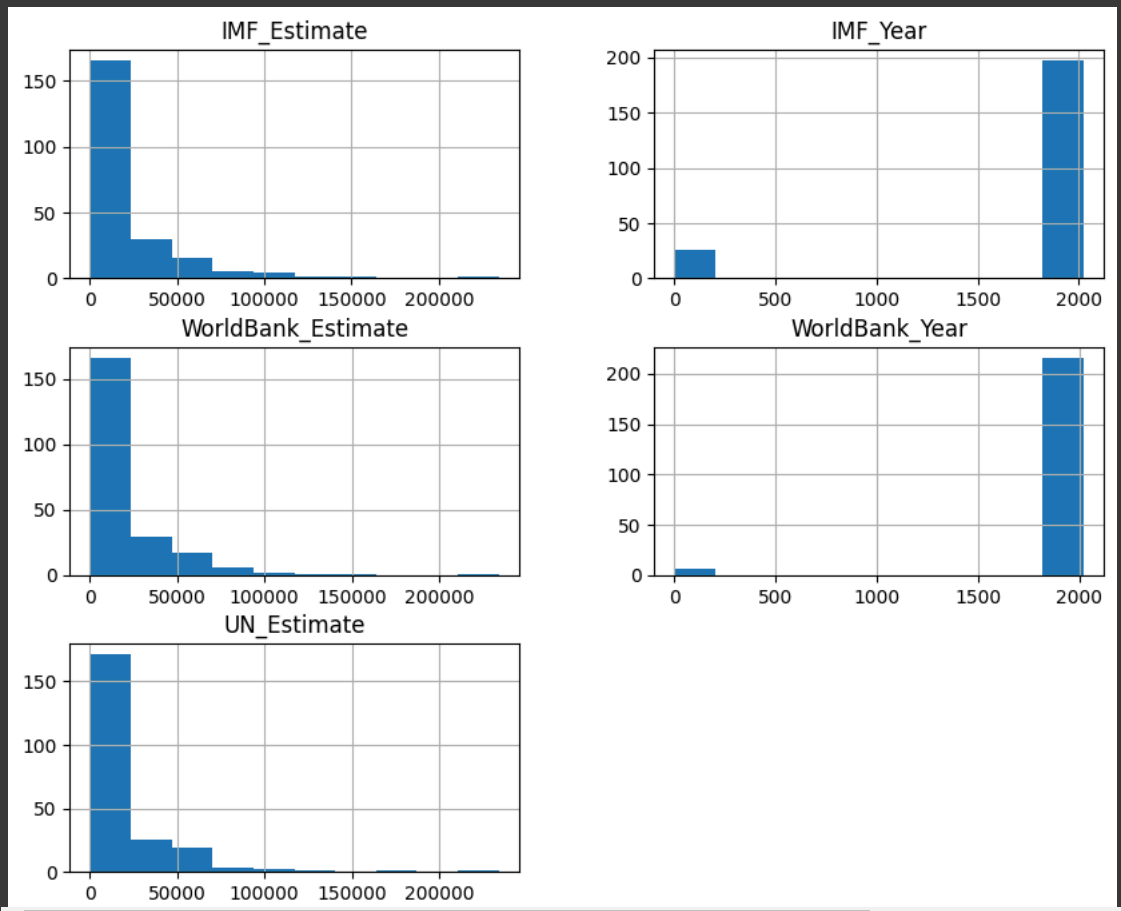
```
WorldBank_Year        0
UN_Estimate           0
UN_Year               0
dtype: int64
```

Visualization

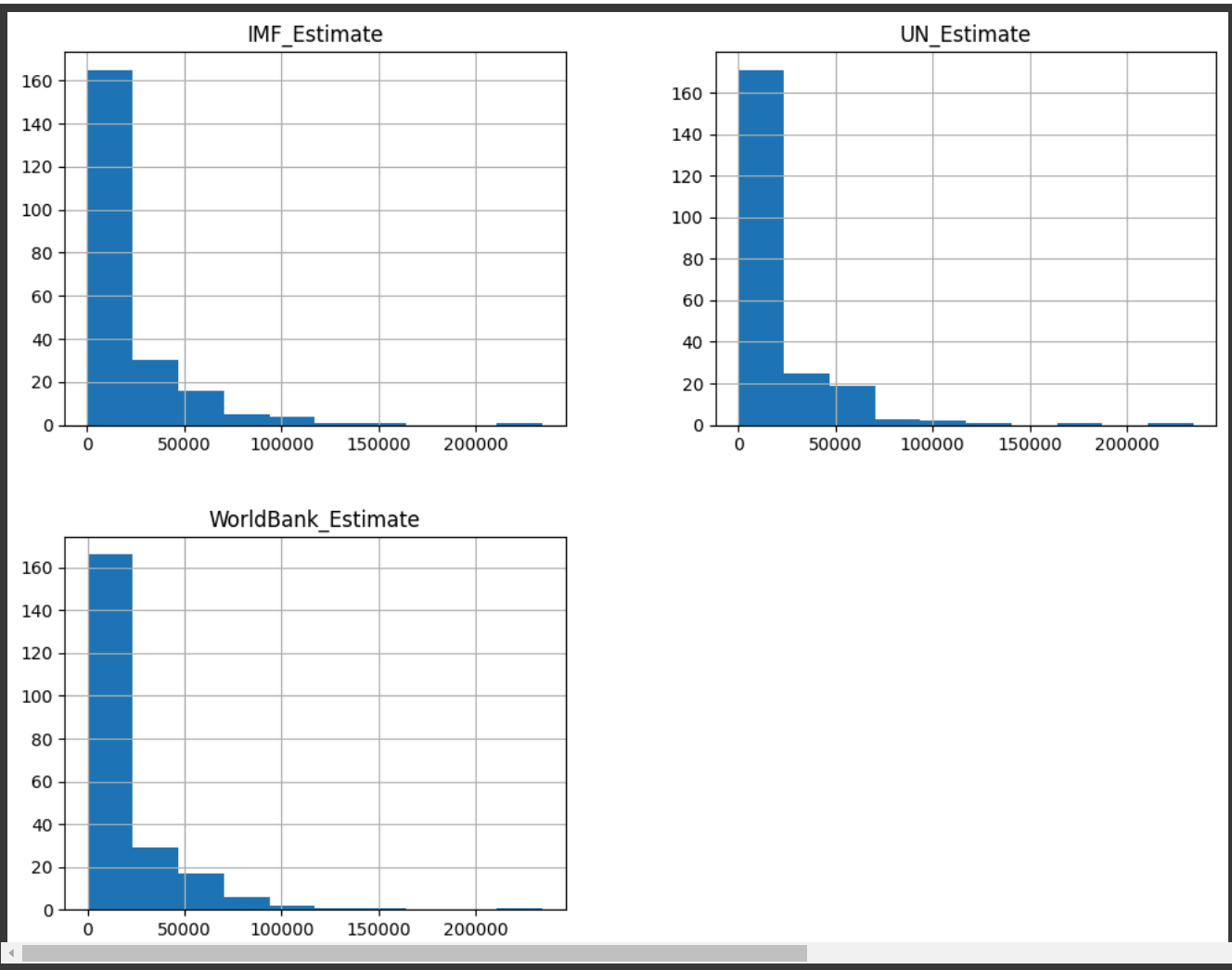
```
import matplotlib.pyplot as plt
import seaborn as sns
```

Histogram

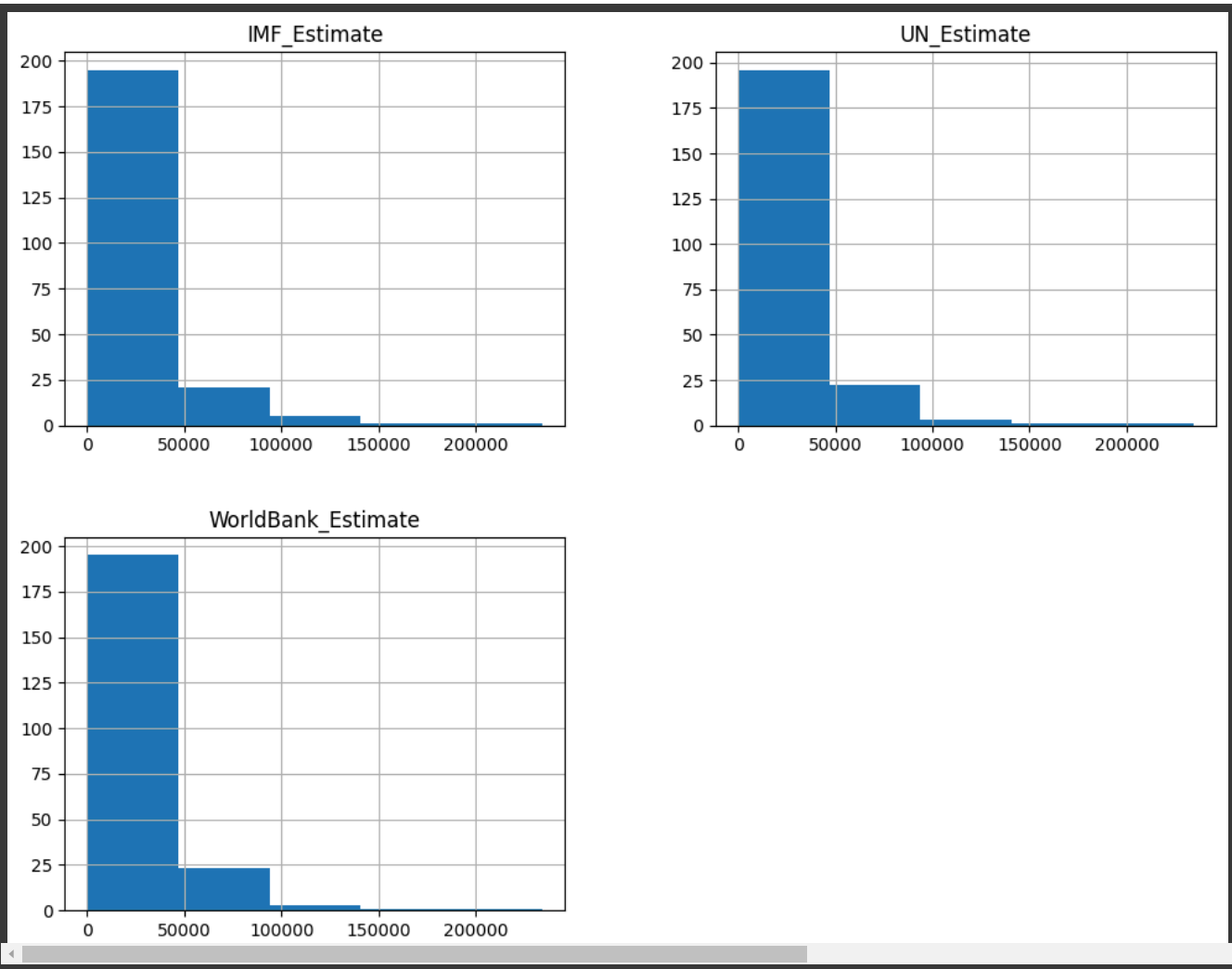
```
df.hist(figsize=(10,8))
plt.show()
```




```
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	0
max	234316

dtype: int64

```
234316/5
#1 bin size if bins=5
```

```
46863.2
```

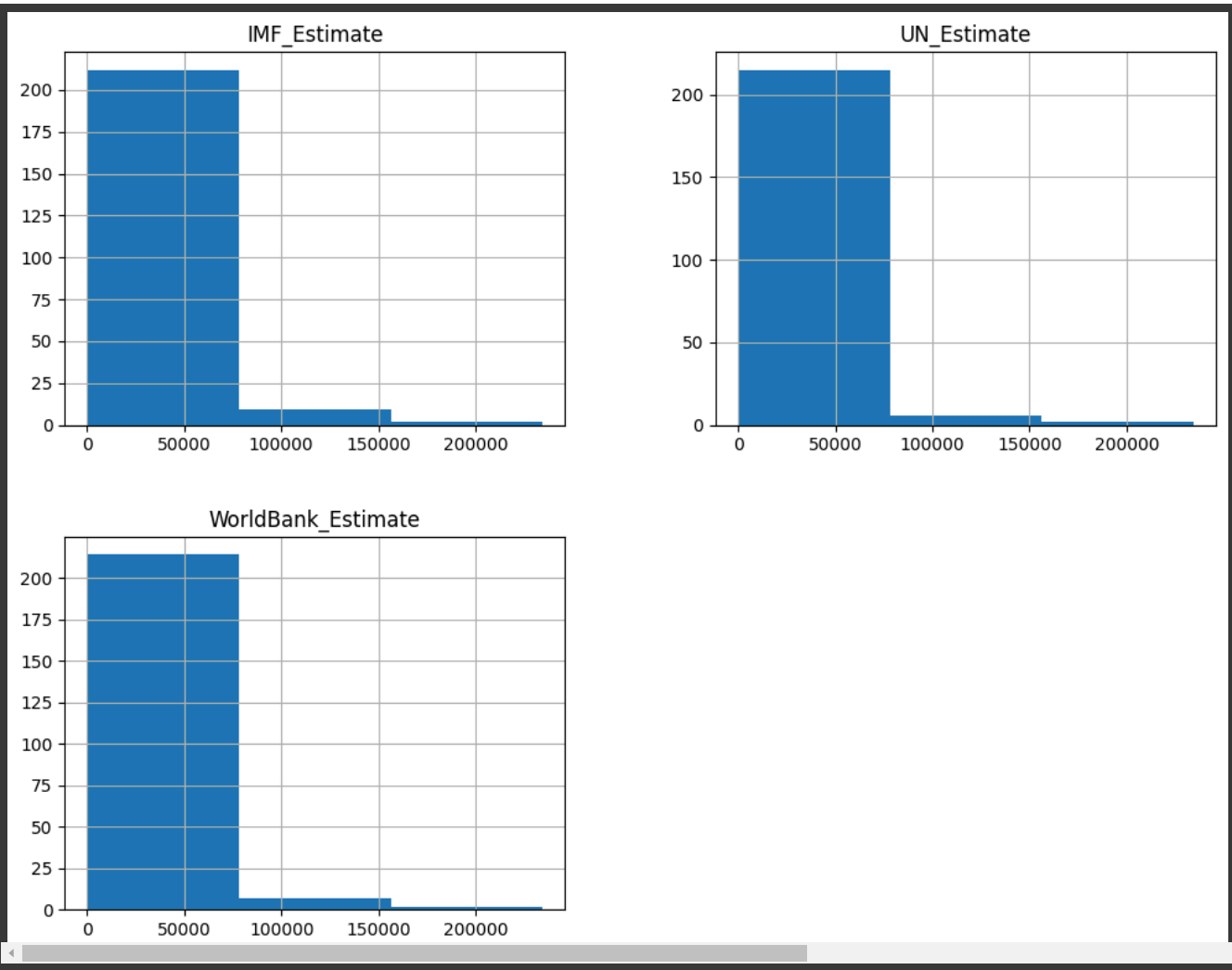
```
df[df["WorldBank_Estimate"]<=46863.2]["WorldBank_Estimate"].count()
```

```
195
```

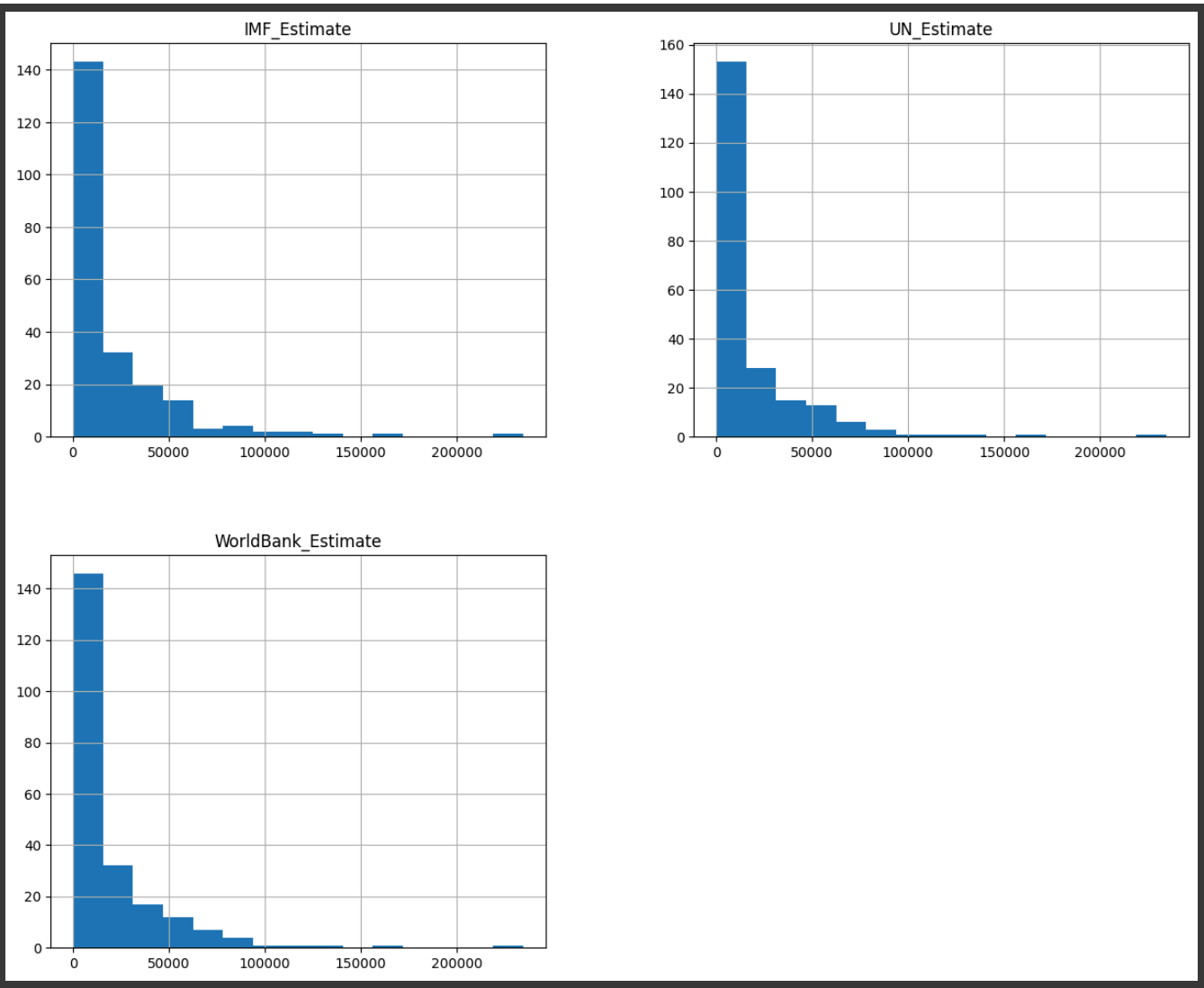
```
234316/10
#1 bin size if bins not given any number
```

```
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()
```

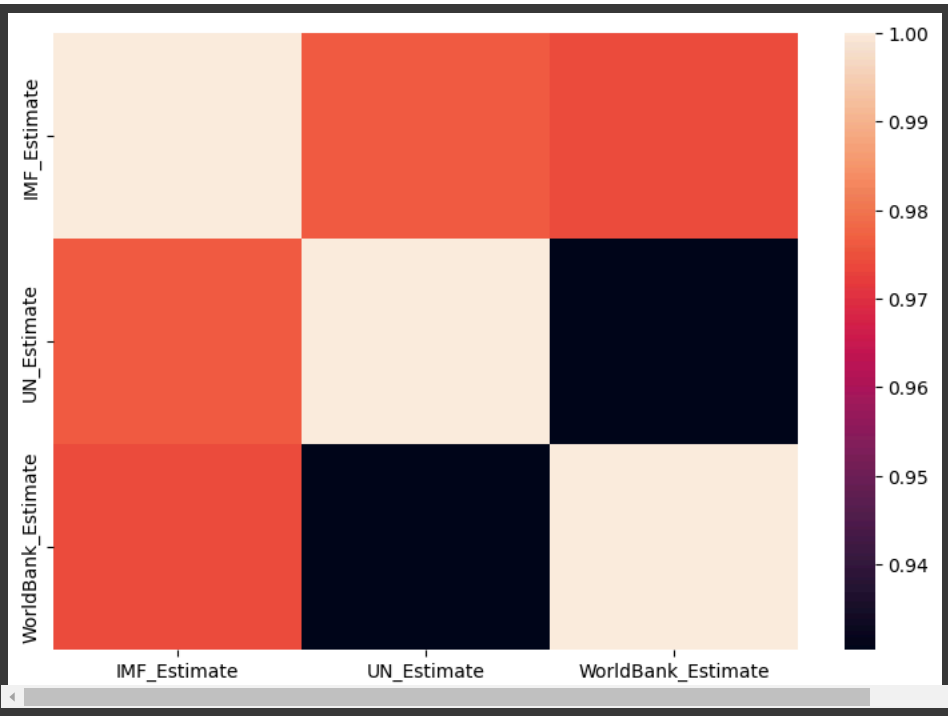


Correlation Heatmap

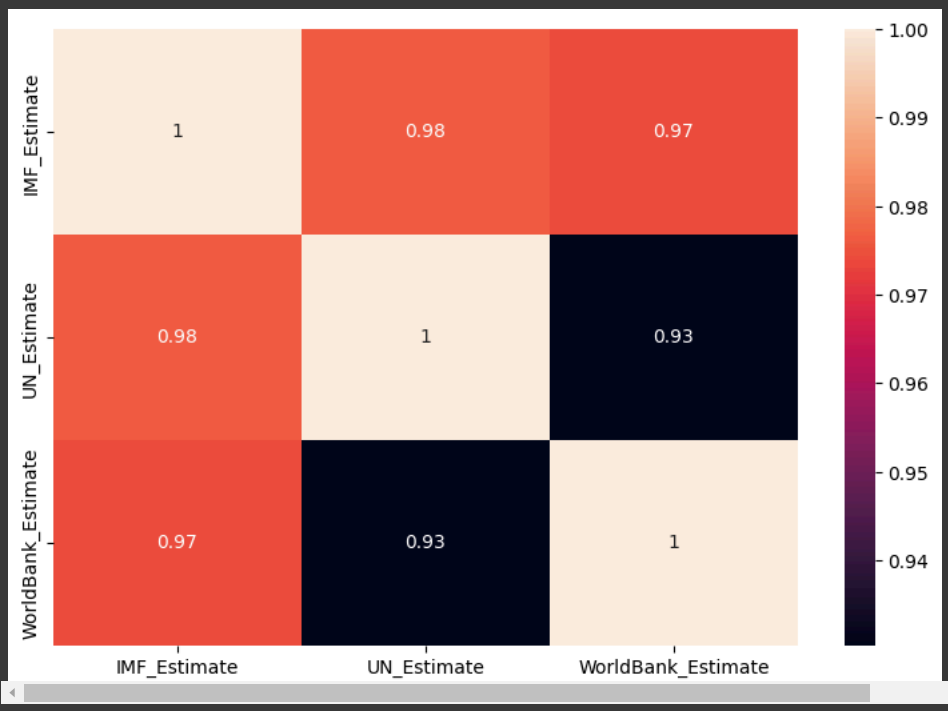
```
df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()
```

	IMF_Estimate	UN_Estimate	WorldBank_Estimate
IMF_Estimate	1.000000	0.976263	0.974294
UN_Estimate	0.976263	1.000000	0.930331
WorldBank_Estimate	0.974294	0.930331	1.000000

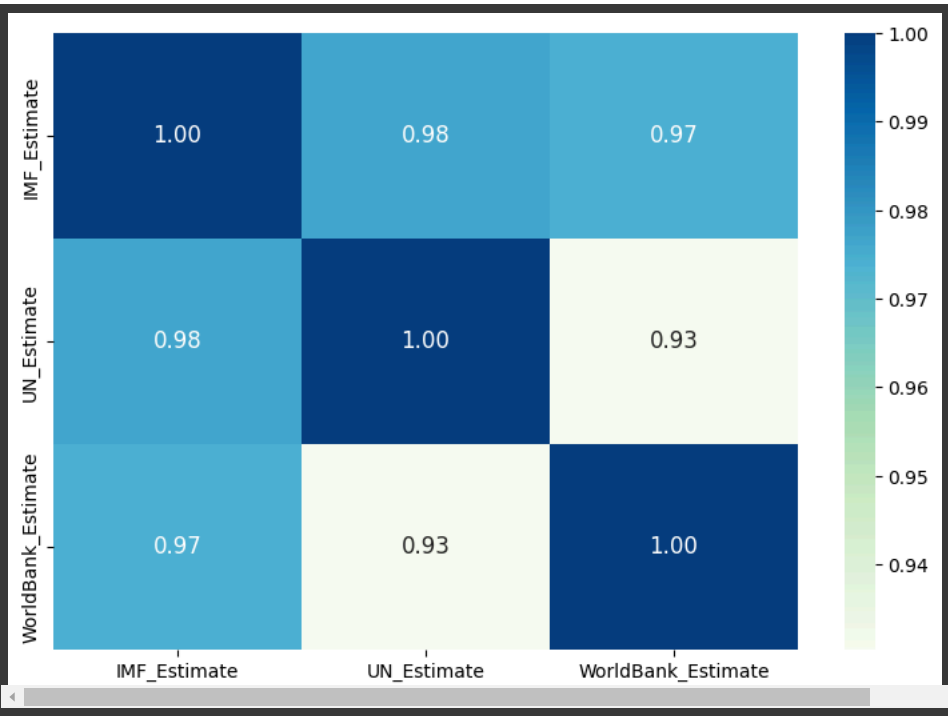
```
corr = df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()

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

plt.show()
```



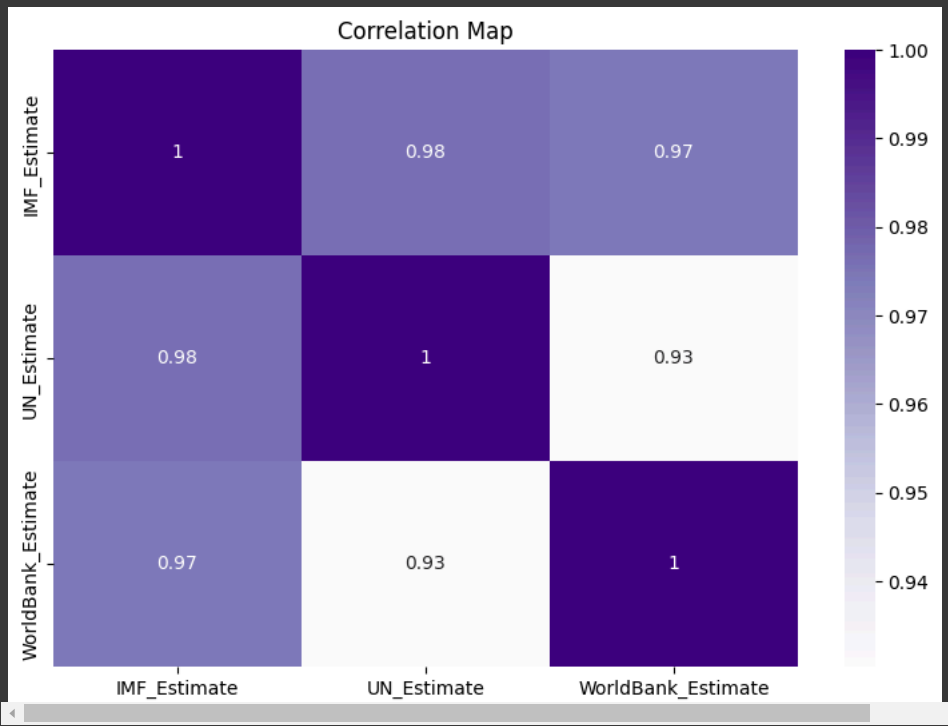

```
corr = df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()

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

sns.heatmap(corr, annot=True)

plt.show()
```



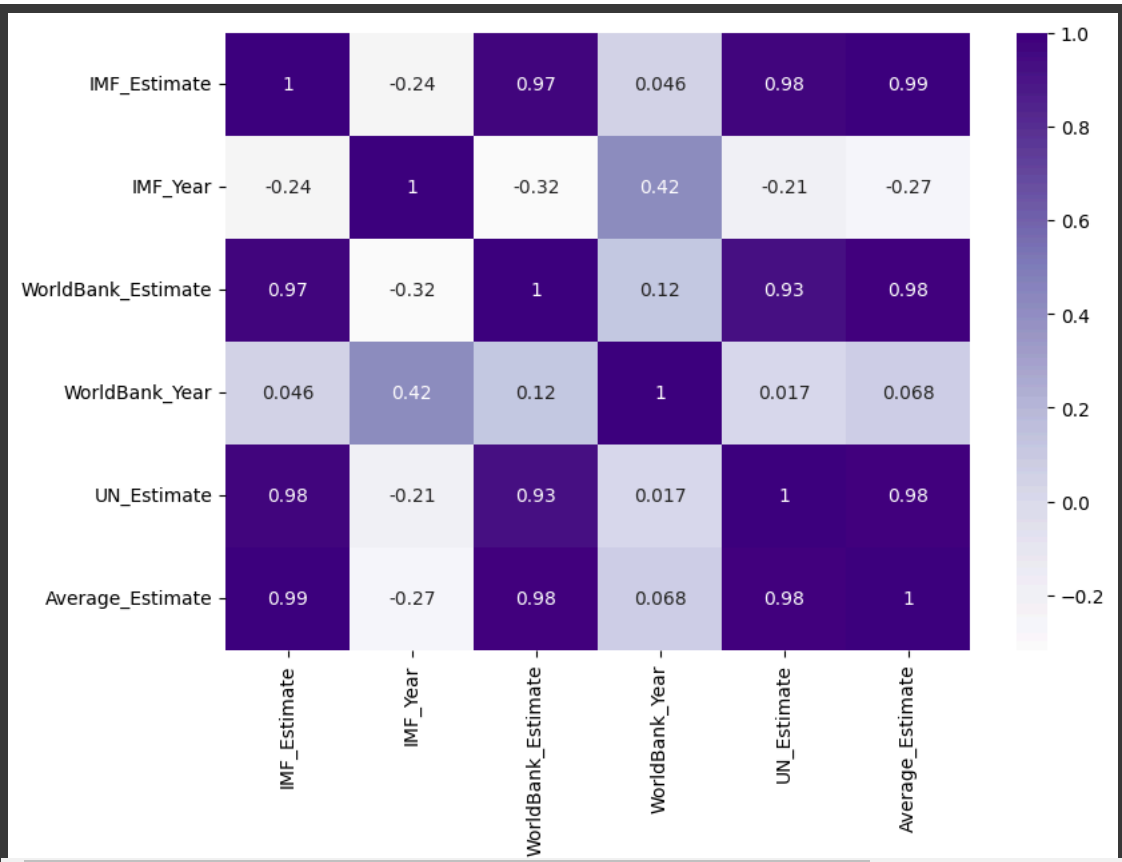

```
corr = df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()

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

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

plt.show()
```




```
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()
```

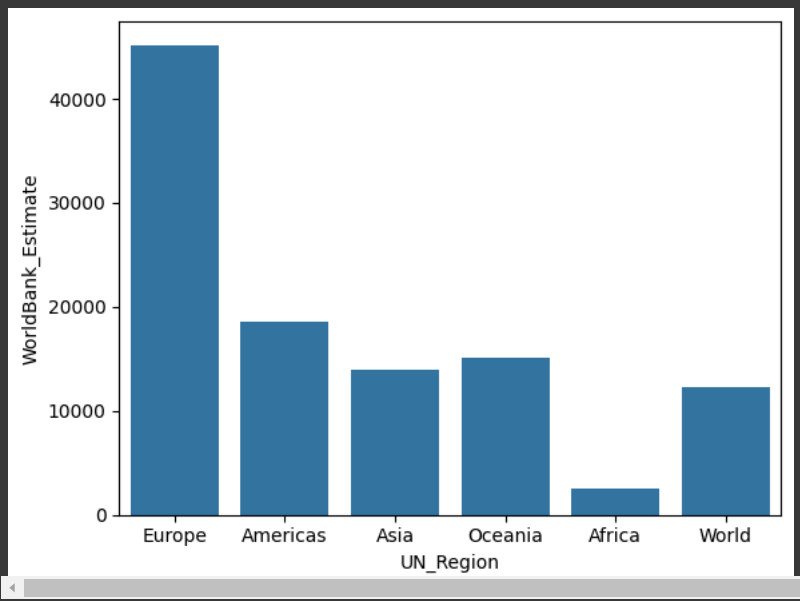


Bar plot

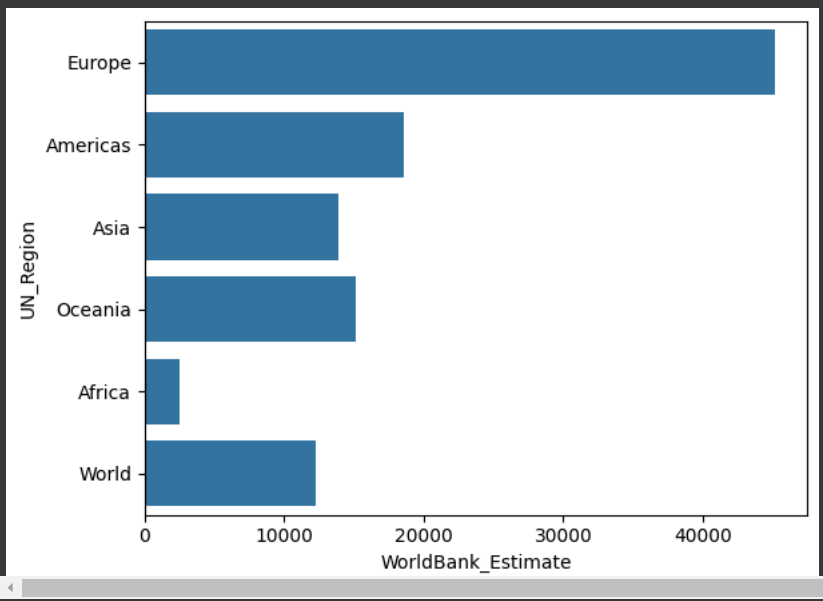
```
df.head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	Average_Estimate
1	Monaco	Europe	234316.5	0	234316	2021	234317	2021	234316.5
2	Liechtenstein	Europe	163507.5	0	157755	2020	169260	2021	163507.5
3	Luxembourg	Europe	132372.0	2023	133590	2021	133745	2021	133667.5
4	Ireland	Europe	114581.0	2023	100172	2021	101109	2021	100640.5
5	Bermuda	Americas	113371.5	0	114090	2021	112653	2021	113371.5

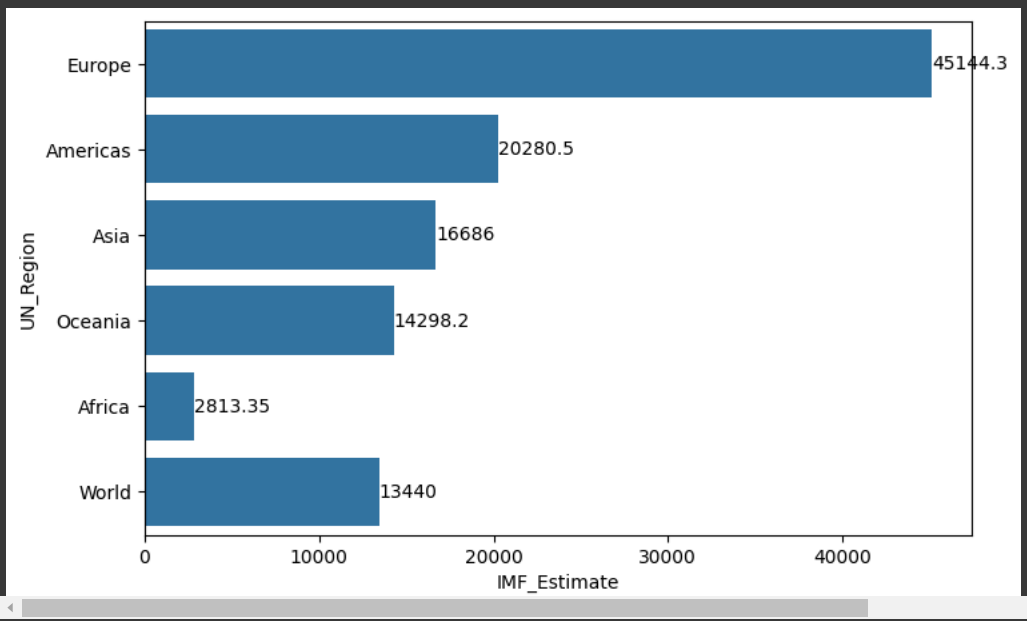
```
sns.barplot(x="UN_Region", y="WorldBank_Estimate", data=df, errorbar=None)

plt.show()
```



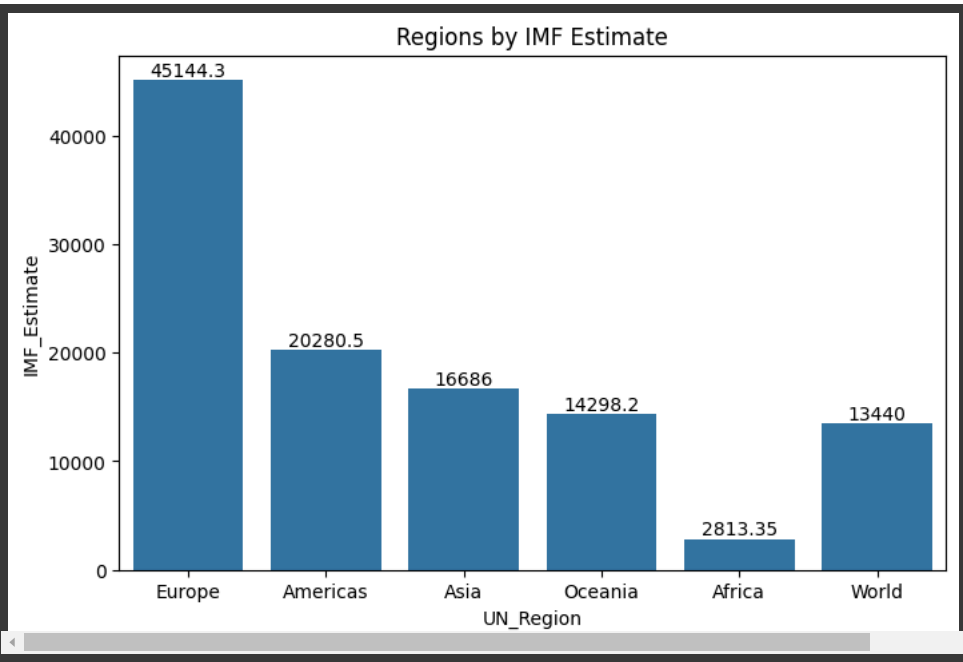
```
sns.barplot(x="WorldBank_Estimate", y="UN_Region", data=df, errorbar=None)

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()
```

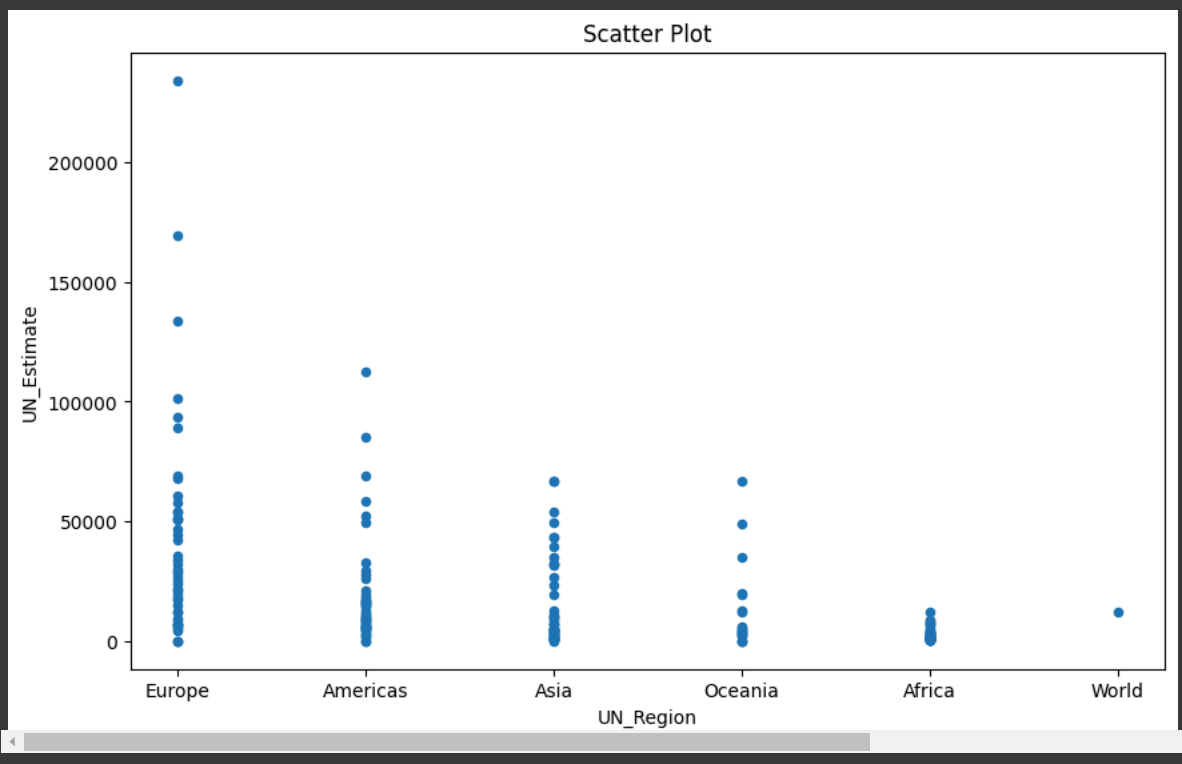



```
fig = plt.figure(figsize = (8,5))
ax = sns.barplot(x = "UN_Region",  y = "IMF_Estimate",
                 data = df, errorbar = None)

ax.bar_label(ax.containers[0])


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



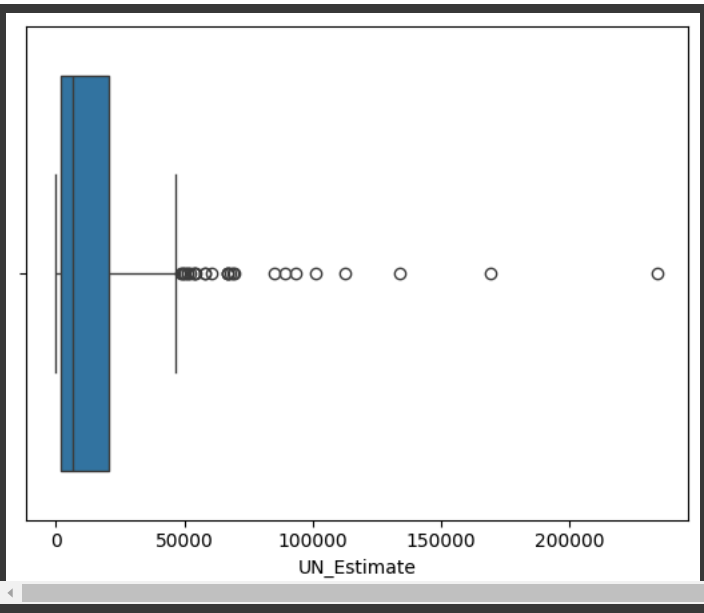
Scatter Plot

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

plt.show()
```



Boxplot and Outliers

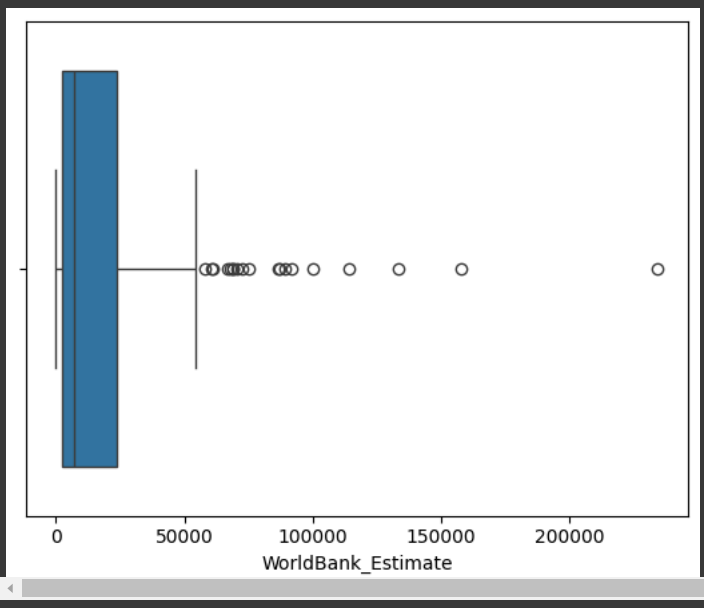
```
sns.boxplot(x=df["UN_Estimate"])

plt.show()
```



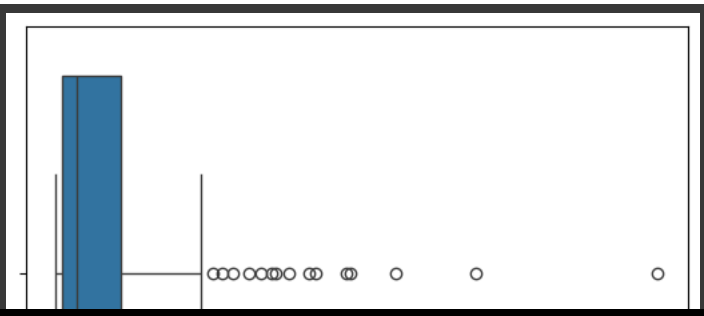

```
df[df["UN_Estimate"]>50000].head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
1	Monaco	Europe	234316.5	0	234316	2021	234317	2021
2	Liechtenstein	Europe	163507.5	0	157755	2020	169260	2021
3	Luxembourg	Europe	132372.0	2023	133590	2021	133745	2021
4	Ireland	Europe	114581.0	2023	100172	2021	101109	2021
5	Bermuda	Americas	113371.5	0	114090	2021	112653	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
1	Monaco	Europe	234316.5	0	234316	2021	234317	2021
2	Liechtenstein	Europe	163507.5	0	157755	2020	169260	2021
3	Luxembourg	Europe	132372.0	2023	133590	2021	133745	2021
4	Ireland	Europe	114581.0	2023	100172	2021	101109	2021