Aufgabe 2

In [ ]:
```
import pandas as pd
```

1. Teilaufgabe:\ Load the countries.csv directly via URL import into your panda data frame!

In [ ]:
```
url = 'https://raw.githubusercontent.com/WHPAN0108/BHT-DataScience-S23/main/python-DS/country.csv'
df = pd.read_csv(url)
df
```

Out[ ]:

	Name	Population	Area	GDP	Currency
0	Germany	82521653	357385	3466.0	EUR
1	Japan	126045000	377835	4938.0	YEN
2	Canada	36503097	9984670	1529.0	CAD
3	Italy	60501718	301338	1850.0	EUR
4	Brazilia	208360000	8515770	1798.0	REAL
5	Taiwan	23938272	36197	744.0	NTD
6	Venezuela	28208977	912050	NaN	VED

Beschreibung zu 1.:

- Angabe der URL vom GitHub-Repository, URL muss Pfad zu RAW-Datei enthalten
- Einlesen der kommaseparierten CSV-Inhalte via "read_csv()"

2. Teilaufgabe:\ Display descriptive statistics for the numerical column (count, mean, std, min, 25%, 50%, 75%, max) HINT: describe

In [ ]:
```
# column 'Population'
print("Population (count): {}".format(df['Population'].count())) # count of rows // Zeilenanzahl
print("Population (sum): {}".format(df['Population'].sum()))    # sum of all values // Summe aller Werte
print("Population (mean): {}".format(df['Population'].mean()))    # mean of all values // Durschnitt aller Werte
print("Population (std): {}".format(df['Population'].std()))    # std of all values // Standardabweichung aller Werte
```

```
print("Population (min): {}".format(df['Population'].min()))   # min of all values // Minimalwert aller Werte
print("Population (quantile): {}".format(df['Population'].quantile([0.25, 0.50, 0.75])))   # quantile (25%, 50%, 75%)
print("Population (max): {}".format(df['Population'].max()))   # min of all values // Maximalwert aller Werte
print()

# column 'Area'
print("Population (count): {}".format(df['Area'].count())) # count of rows // Zeilenanzahl
print("Population (sum): {}".format(df['Area'].sum()))   # sum of all values // Summe aller Werte
print("Population (mean): {}".format(df['Area'].mean()))   # mean of all values // Durschnitt aller Werte
print("Population (std): {}".format(df['Area'].std()))   # std of all values // Standardabweichung aller Werte
print("Population (min): {}".format(df['Area'].min()))   # min of all values // Minimalwert aller Werte
print("Population (quantile): {}".format(df['Area'].quantile([0.25, 0.50, 0.75])))   # quantile (25%, 50%, 75%) // Tei
print("Population (max): {}".format(df['Area'].max()))   # min of all values // Maximalwert aller Werte
print()

# column 'GDP'
print("Population (count): {}".format(df['GDP'].count())) # count of rows // Zeilenanzahl
print("Population (sum): {}".format(df['GDP'].sum()))   # sum of all values // Summe aller Werte
print("Population (mean): {}".format(df['GDP'].mean()))   # mean of all values // Durschnitt aller Werte
print("Population (std): {}".format(df['GDP'].std()))   # std of all values // Standardabweichung aller Werte
print("Population (min): {}".format(df['GDP'].min()))   # min of all values // Minimalwert aller Werte
print("Population (quantile): {}".format(df['GDP'].quantile([0.25, 0.50, 0.75])))   # quantile (25%, 50%, 75%) // Teil
print("Population (max): {}".format(df['GDP'].max()))   # min of all values // Maximalwert aller Werte
```

```
Population (count): 7
Population (sum): 566078717
Population (mean): 80868388.14285715
Population (std): 66701952.313208535
Population (min): 23938272
Population (quantile): 0.25     32356037.0
0.50     60501718.0
0.75    104283326.5
Name: Population, dtype: float64
Population (max): 208360000

Population (count): 7
Population (sum): 20485245
Population (mean): 2926463.5714285714
Population (std): 4348507.096282464
Population (min): 36197
Population (quantile): 0.25     329361.5
0.50     377835.0
0.75    4713910.0
Name: Area, dtype: float64
Population (max): 9984670

Population (count): 6
Population (sum): 14325.0
Population (mean): 2387.5
Population (std): 1532.4975367027512
Population (min): 744.0
Population (quantile): 0.25    1596.25
0.50    1824.00
0.75    3062.00
Name: GDP, dtype: float64
Population (max): 4938.0
```

Beschreibung zu 2.:

- Funktion count() zum ermitteln der Zeilenanzahl
- Funktion sum() zum ermitteln der Gesamtsumme
- Funktion mean() zum ermitteln der Durchschnittsmenge
- Funktion std() zum ermitteln der Standardabweichung
- Funktion min() / max() zum ermitteln des Minimal-/Maximalwertes
- Funktion quantile() zum ermitteln der Teile einer Datenmenge

3. Teilaufgabe:\ Show the last 4 rows of the data frame.

In [ ]:
```
df.tail(4)
```

Out[ ]:

	Name	Population	Area	GDP	Currency
3	Italy	60501718	301338	1850.0	EUR
4	Brazilia	208360000	8515770	1798.0	REAL
5	Taiwan	23938272	36197	744.0	NTD
6	Venezuela	28208977	912050	NaN	VED

Beschreibung zu 3.:\ Mit Hilfe der Funktion "tail()" können die letzten n-Zeilen eines Datenrahmens abgefragt werden.

4. Teilaufgabe:\ Show all the rows of countries that have the EURO

In [ ]:
```
df.iloc[[0,3], :]
```

Out[ ]:

	Name	Population	Area	GDP	Currency
0	Germany	82521653	357385	3466.0	EUR
3	Italy	60501718	301338	1850.0	EUR

Beschreibung zu 4.:\ Mit der Funktion iloc(Zeile(n), Spalte(n)) können gezielt Zeilen und/oder Spalten eines Datenrahmens selektiert werden.

5. Teilaufgabe:\ Show only name and Currency in a new data frame

In [ ]:
```
df2 = df.iloc[:, [0,4]]
df2
```

Out[ ]:

	Name	Currency
0	Germany	EUR
1	Japan	YEN
2	Canada	CAD
3	Italy	EUR
4	Brazilia	REAL
5	Taiwan	NTD
6	Venezuela	VED

Beschreibung zu 5.:\ Mit der Funktion iloc(Zeile(n), Spalte(n)) kann selektiert hier die Spalten "Name" und "Currency" in ein neuen Datenrahmen abgefragt werden. Als erstes wird mit einem Doppelpunkt ":" angegeben dass alle Zeilen selektiert werden. In der zweiten Angabe wird in einen Array angegeben dass die Spalten am Index 0=Name und 4=Currency selektiert werden.

6. Teilaufgabe:\ Show only the rows/countries that have more than 2000 GDP (it is in Milliarden USD Bruttoinlandsprodukt)

In [ ]:
```
df.query('GDP > 2000.0')[['Name', 'GDP']]
```

Out[ ]:

	Name	GDP
0	Germany	3466.0
1	Japan	4938.0

Beschreibung zu 6.:

- Im ersten Schritt werden mit der Funktion "query" alle Zeilen aus der Spalte "GDP" selektiert die einen Wert über 2000.0 enthalten.
- Im zweiten Schritt werden die zwei Spalten "Name" (= Ländernamen) und "GDP" über einen Multi Selection ausgewählt und angezeigt.

7. Teilaufgabe:\ Select all countries where with inhabitants between 50 and 150 Mio

In [ ]:
```
df.query('Population > (50*1_000_000) and Population < (150*1_000_000)')[['Name', 'Population']]
```

Out[ ]:

	Name	Population
0	Germany	82521653
1	Japan	126045000
3	Italy	60501718

Beschreibung zu 7.:

- Ähnlich wie die Teilaufgabe 7. wird über die Query-Funktion die Spalte "Population" nach Werten gefiltert die eine Bevölkerung zwischen 50 und 150 Mio. Menschen haben.
- Im letzten Schritt werden zur Ausgabe des Datenrahmens wieder die Spalten selektiert, hier "Name" (= Ländernamen) und "Population" (= Bevölkerungsanzahl).

8. Teilaufgabe:\ Calculate the GDP average (ignore the missing value)

In [ ]:
```
df['GDP'].mean()
```

Out[ ]:
```
2387.5
```

Beschreibung zu 8.:\ Es wird wieder über die eckigen Klammern die Spalte "GDP" ausgewählt und mit Hilfe der Funktion "mean()" die Durchschnittsmenge aller Werte berechnet.

9. Teilaufgabe:\ Calculate the GDP average (missing value treated as 0)

In [ ]:
```
df2 = df['GDP']
df2.fillna(0.0).mean()
```

Out[ ]:
```
2046.4285714285713
```

Beschreibung zu 9.:

- Es wird im ersten Schritt die Spalte "GDP" ausgewählt und zum besseren Arbeiten in ein neuen Datenrahmen kopiert.
- Im nächsten Schritt werden alle Werte die NaN (Not a Number) oder Null sind mit einen gegebenen Wert gefüllt, hier mit 0.0.
- Zuletzt wird wieder die Durschnittsmenge berechnet.

10. Teilaufgabe:\ Calculate the population density (population/area) of all countries and add as new column

In [ ]:
```
df2 = pd.DataFrame({'Population density': df['Population'] / df['Area']})
df3 = pd.concat([df, df2], axis=1)
df3
```

Out[ ]:

	Name	Population	Area	GDP	Currency	Population density
0	Germany	82521653	357385	3466.0	EUR	230.904075
1	Japan	126045000	377835	4938.0	YEN	333.597999
2	Canada	36503097	9984670	1529.0	CAD	3.655914
3	Italy	60501718	301338	1850.0	EUR	200.776928
4	Brazilia	208360000	8515770	1798.0	REAL	24.467547
5	Taiwan	23938272	36197	744.0	NTD	661.333039
6	Venezuela	28208977	912050	NaN	VED	30.929200

Beschreibung zu 10.:

- In der 1. Zeile wird ein neuer Datenrahmen mit nur einer Spalte "Population density" erzeugt. Die Spaltenwerte ergeben sich aus der Division der Spalten "Population" und "Area".
- In der 2. Zeile wird über die Konkatenation des Datenrahmens "df" und "df2" werden alle Spalten zusammengefügt. Wichtig hierbei ist die Angabe der Achsenbeziehung (axis=1). Zur besseren Übersicht wird ein neuer Datenrahmen "df3" erzeugt und ausgegeben.

11. Teilaufgabe:\ Sort by country name alphabetically

In [ ]:
```
df.sort_values('Name', ascending=True)
```

Out[ ]:

	Name	Population	Area	GDP	Currency
4	Brazilia	208360000	8515770	1798.0	REAL
2	Canada	36503097	9984670	1529.0	CAD
0	Germany	82521653	357385	3466.0	EUR
3	Italy	60501718	301338	1850.0	EUR
1	Japan	126045000	377835	4938.0	YEN
5	Taiwan	23938272	36197	744.0	NTD
6	Venezuela	28208977	912050	NaN	VED

Beschreibung zu 11.:\ Mit Hilfe der Funktion "sort_values(...)" werden alle Zeilen des Datenrahmens ausgehend von der Spalte "Name" (= Ländernamen) alphabetisch neugeordnet.

12. Teilaufgabe:\ Create a new data frame from the original where the area is changed: all countries with > 1000000 get BIG and <= 1000000 get SMALL in the cell replaced!

In [ ]:

```
df2 = df.copy() # deep copy

for index, row in df.iterrows(): # iterate over first data frame
    if row['Area'] > 1_000_000: # area is bigger than 1_000_000
        u = str(row['Name']).upper() # string of name to uppercase
        df2.loc[int(index), ('Name')] = u # replace element by location
    else: # area is bigger than 1_000_000
        l = str(row['Name']).lower() # string of name to lowercase
        df2.loc[int(index), ('Name')] = l # replace element by location

df2
```

Out[ ]:

	Name	Population	Area	GDP	Currency
0	germany	82521653	357385	3466.0	EUR
1	japan	126045000	377835	4938.0	YEN
2	CANADA	36503097	9984670	1529.0	CAD
3	italy	60501718	301338	1850.0	EUR
4	BRAZILIA	208360000	8515770	1798.0	REAL
5	taiwan	23938272	36197	744.0	NTD
6	venezuela	28208977	912050	NaN	VED

Beschreibung zu 12.:

- Zuerst eine Kopie vom originalen Datenrahmen erstellen ("df" mit Kopie zu "df2").
- Dann über alle Zeilen vom Datenrahmen "df" iterieren.
- Prüfe, ob Area über 1.000.000 liegt oder darunter. Entsprechen Ländernamen in groß/klein schreibweise umformatieren.
- Formatierten String an den entsprechenden Index mit der "loc()"-Funktion in die Kopie "df2" schreiben.