In [ ]:
```
import pandas as pd
import numpy as np
import json
```

Load the modules.

In [ ]:
```
df = pd.read_csv('API_EG.USE.PCAP.KG.OE_DS2_en_csv_v2_4028587.csv',ski
```

Read in the data into a pandas dataframe.

```
You need to skip 4 lines from the first of csv file.
```

In [ ]:
```
df.info(verbose=False)
```

Display the usual basic information (datatypes, names, non-null values for columns) about the dataframe.

```
Every row represent a county and column is based on year from
1960 to 2022.
```

In [ ]:
```
pd.set_option("display.max_columns",100)
pd.set_option("display.max_rows",50)
df
```

Display the dataframe.

```
Because the terminal has limitation to show the result i restr
icted it.
```

```
In [ ]: df["Indicator Name"].unique() , len(df["Indicator Name"].unique())
```

Find out what values occur under "Indicator Name".

```
Get the unique value in column "Indicator Name".
There is just one unique value for the column "Indicator Name"
.
```

```
In [ ]: df.rename(columns={"Indicator Name" : df["Indicator Name"][0]},inplace
```

Set the name of the columns to the only value you found under "Indicator Name" (the columns and their labels remain the same!). (Apply the operation to your dataframe!)

```
In [ ]: dfcode = pd.DataFrame(df,columns=["Country Name","Country Code"])
        dfcode
```

Create a new dataframe dfcode which contains the "Country Code" and "Country Name" columns of your original dataframe. You will continue working with your original dataframe, in what follows, though.

In [ ]:
```
df = df.drop(columns=["Country Code","Energy use (kg of oil equivalent
```

> Remove the "Country Code", "Indicator Name", "Indicator Code" columns. (Apply the operation to your dataframe!)

```
Get rid of unnecessary columns so we can concentrate on main d
ata.
```

In [ ]:
```
df.set_index("Country Name",inplace=True)
```

> Make the column "Country Name" the index of your dataframe. (Apply the operation to your dataframe!)

```
Since the value of the "Country Name" column is a unique strin
g we can use it as index.
```

In [ ]:
```
df = df.dropna(axis = "columns", how ='all')
df
```

> Get rid of columns which contain no values. (Apply the operation to your dataframe!)

```
 There is no numerical values for some of years, so it is bett
er to find and remove these years.
```

In [ ]:
```
df_min_years = df.loc[:,"1960":"2015"].min(axis="rows")
df_max_years = df.loc[:,"1960":"2015"].max(axis="rows")
df_mean_years = df.loc[:,"1960":"2015"].mean(axis="rows")
df_median_years = df.loc[:,"1960":"2015"].median(axis="rows")
df_std_years = df.loc[:,"1960":"2015"].std(axis="rows")
df_median_years.sort_values(ascending=False).head(5)
```

Display the basic statistics by year, incl. minimal, maximal, mean, median, standard deviation values. (Make sure you display these values for all the years!)

What can you conclude from the yearly descriptives? Answer in a markdown cell.

```
2006 has the lower value and 1973 has the maximum value.The me
an value is in the range of 675.1 to 3746.1 .
```

In [ ]:
```
df_min_country = df.min(axis="columns")
df_max_country = df.max(axis="columns")
df_mean_country = df.mean(axis="columns")
df_median_country = df.median(axis="columns")
df_std_country = df.std(axis="columns")
```

Display the basic statistics of your previous by country-statistics (i.e., the mean of country means, etc.).

```
If you define axis as rows it calculate statistic based on yea
r, and if you define axis as columns the statistic value is ba
sed on countries.
```

In [ ]:
```
df_min = df_min_country.min()
df_max = df_max_country.max()
df_mean = df_mean_country.mean()
df_median = df_median_country.mean()
df_std = df_std_country.std()
```

Display the basic statistics by country, incl. minimal, maximal, mean, median, standard deviation values. (Make sure you display these values for all the countries!)

```
In [ ]: df.sort_values(by='2015', ascending = False,inplace = False).head(30)
```

Display the first thirty rows of your original dataframe after sorting by the values for the **last year** in **descending** order. (Do not apply sorting to your original dataframe, just display!)

What are some of the conclusions you can draw from the data you displayed (e.g., about the "countries" in the data)?

```
Some of counries has small data specially between 1960 and 197
0. most of cell in these years have no value.
```

```
In [ ]: df["difference"] = df["2010"].fillna(0) - df["2000"].fillna(0)
        df.sort_values(by=['difference'], ascending=False,inplace = False).hea
```

Calculate the difference between the (0-filled) values for the **years 2010 and 2000** by country, sort it in a **descending** order, and display the **first 30 rows**. What are some of the conclusions you could draw from what you see?

```
With fillna(0) command we replace none value with zero then we
subtract the two value for year 2000 and 2010.
```

In [ ]:

```
mean_year = df.loc[:,"1960":"2015"].mean(axis="rows")
max_year = df.loc[:,"1960":"2015"].max(axis="rows")
percentage = (mean_year/max_year)*100
percentage
```

Display the **mean** values by year as a percentage of the **maximal** values for that year (e.g., if the mean value for 1965 were 200, and the maximal for 1965 were 400, then for 1965, you should be displaying 50).

In [ ]:

```
df[df.loc[:,"1960":"2015"].isna()].index
```

Display the countries (not the subdataframe, just the countries!) where the values for *all* years are missing.

In [ ]:

```
df.loc[df["2015"]>mean_year["2015"],:]
```

Display the subdataframe with the countries whose final year's values are bigger than the **final** year values' mean.

In [ ]:

```
with open("capital.json", "r") as file:
    capitaldf = json.load(file)
capitaldf = pd.DataFrame(capitaldf["data"],columns=["name","capital","
capitaldf
```

Load the "capital.json" file, and arrive at a dataframe capitaldf containing the **name, capital, iso2, iso3** information (these three should end up as the columns) for every country in it.

In [ ]:
```
dfnocode = dfcode.loc[dfcode["Country Code"].isin(capitaldf["iso3"]),:
dfnocode
```

Find the subdataframe of the dfcode dataframe you created earlier on where the "Country Code" value is **not** in the "iso3" column of capitaldf. Hint: look up and use the .isin pandas Series method. Store this subdataframe in the variable dfnocode.

Display dfnocode and explain in a markdown cell what you see and what this could be used for.

```
There are the name of each country and iso 3 code, so you can
connect every data frame that uses iso3 code instead of name o
f country.
```

In [ ]:
```
df.loc[df.index.isin(dfnocode["Country Name"]) == False,:]
```

Now display the subdataframe of your main dataframe with the yearly data where the index (the Country Name) is not in the "Country Name" column of dfnocode.