In [103]:
```
import pandas as pd
```

1 - Read in the data into a pandas dataframe.

- The first 4 rows of our dataframe is empty, so we don not consider them.

In [104]:
```
df = pd.read_csv("API_EG.USE.PCAP.KG.OE_DS2_en_csv_v2_4028587.csv", sk
```

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

- All values relate to year's columns are in float type and reperesent the energy consumption for each country on that year.
- All series equal or greater than 2016 don not have any values.
- the 4 first column of data frame represent '***Country Name***', '***Country Code***', '***Indicator Name***', and '***Indicator Code***' and contain strings in them(object type).

In [105]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Data columns (total 67 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Country Name    266 non-null    object 
 1   Country Code    266 non-null    object 
 2   Indicator Name  266 non-null    object 
 3   Indicator Code  266 non-null    object 
 4   1960            31 non-null     float64
 5   1961            31 non-null     float64
 6   1962            31 non-null     float64
 7   1963            31 non-null     float64
 8   1964            31 non-null     float64
 9   1965            32 non-null     float64
 10  1966            32 non-null     float64
 11  1967            32 non-null     float64
 12  1968            32 non-null     float64
 13  1969            32 non-null     float64
 14  1970            32 non-null     float64
 15  1971            151 non-null    float64
 16  1972            151 non-null    float64
 17  1973            151 non-null    float64
 18  1974            151 non-null    float64
```

```
 19  1975            151 non-null    float64
 20  1976            151 non-null    float64
 21  1977            151 non-null    float64
 22  1978            151 non-null    float64
 23  1979            151 non-null    float64
 24  1980            151 non-null    float64
 25  1981            152 non-null    float64
 26  1982            152 non-null    float64
 27  1983            152 non-null    float64
 28  1984            152 non-null    float64
 29  1985            153 non-null    float64
 30  1986            153 non-null    float64
 31  1987            153 non-null    float64
 32  1988            153 non-null    float64
 33  1989            153 non-null    float64
 34  1990            207 non-null    float64
 35  1991            177 non-null    float64
 36  1992            177 non-null    float64
 37  1993            177 non-null    float64
 38  1994            177 non-null    float64
 39  1995            179 non-null    float64
 40  1996            179 non-null    float64
 41  1997            179 non-null    float64
 42  1998            179 non-null    float64
 43  1999            179 non-null    float64
 44  2000            184 non-null    float64
 45  2001            184 non-null    float64
 46  2002            184 non-null    float64
 47  2003            184 non-null    float64
 48  2004            217 non-null    float64
 49  2005            218 non-null    float64
 50  2006            218 non-null    float64
 51  2007            218 non-null    float64
 52  2008            185 non-null    float64
 53  2009            185 non-null    float64
 54  2010            185 non-null    float64
 55  2011            185 non-null    float64
 56  2012            186 non-null    float64
 57  2013            186 non-null    float64
 58  2014            179 non-null    float64
 59  2015            40 non-null     float64
 60  2016            0 non-null      float64
 61  2017            0 non-null      float64
 62  2018            0 non-null      float64
 63  2019            0 non-null      float64
 64  2020            0 non-null      float64
 65  2021            0 non-null      float64
 66  Unnamed: 66     0 non-null      float64
dtypes: float64(63), object(4)
memory usage: 139.4+ KB
```

3 - Display the dataframe.

- To show all columns of dataframe you can do as below.

In [106]:
```
pd.set_option("display.max_columns",67)
```

In [107]:
```
df
```

Out[107]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965
0	Aruba	ABW	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN	NaN
1	Africa Eastern and Southern	AFE	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN	NaN
2	Afghanistan	AFG	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN	NaN
3	Africa Western and Central	AFW	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN	NaN
4	Angola	AGO	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
261	Kosovo	XKX	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN	NaN

Energy

262	Yemen, Rep.	YEM	use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN	NaN
263	South Africa	ZAF	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN	NaN
264	Zambia	ZMB	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN	NaN
265	Zimbabwe	ZWE	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN	NaN

266 rows × 67 columns

4 - Find out what values occur under "Indicator Name".

- There is only one unique value under this column.

In [108]:
```
df["Indicator Name"]
```

Out[108]:
```
0      Energy use (kg of oil equivalent per capita)
1      Energy use (kg of oil equivalent per capita)
2      Energy use (kg of oil equivalent per capita)
3      Energy use (kg of oil equivalent per capita)
4      Energy use (kg of oil equivalent per capita)
                          ...
261    Energy use (kg of oil equivalent per capita)
262    Energy use (kg of oil equivalent per capita)
263    Energy use (kg of oil equivalent per capita)
264    Energy use (kg of oil equivalent per capita)
265    Energy use (kg of oil equivalent per capita)
Name: Indicator Name, Length: 266, dtype: object
```

- The value under the "Indicator Name" column is : "***Energy use (kg of oil equivalent per capita)***"

In [109]:
```
df["Indicator Name"].unique()
```

Out[109]:
```
array(['Energy use (kg of oil equivalent per capita)'], dtype=object)
```

5 - 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!)

- First we extract the unique value under the "Indicator Name" column.
- Then we set name of column to this value.

In [110]:
```
df.rename_axis(df["Indicator Name"].unique(), axis=1, inplace=True)
df
```

Out[110]:

	Energy use (kg of oil equivalent per capita)	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964
0	Aruba	ABW	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN
1	Africa Eastern and Southern	AFE	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN
2	Afghanistan	AFG	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN
3	Africa Western and Central	AFW	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN
4	Angola	AGO	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN

...	...	...	...	...	...	...	...	...	...
261	Kosovo	XKX	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN
262	Yemen, Rep.	YEM	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN
263	South Africa	ZAF	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN
264	Zambia	ZMB	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN
265	Zimbabwe	ZWE	Energy use (kg of oil equivalent per capita)	EG.USE.PCAP.KG.OE	NaN	NaN	NaN	NaN	NaN

266 rows × 67 columns

6 - 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 [111]:
```
dfcode=df[["Country Code" , "Country Name"]]
dfcode
```

Out[111]:

	Energy use (kg of oil equivalent per capita)	Country Code	Country Name
0		ABW	Aruba
1		AFE	Africa Eastern and Southern
2		AFG	Afghanistan
3		AFW	Africa Western and Central
4		AGO	Angola
...		...	...
261		XKX	Kosovo
262		YEM	Yemen, Rep.
263		ZAF	South Africa
264		ZMB	Zambia
265		ZWE	Zimbabwe

266 rows × 2 columns

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

- we created a dataframe named dfcode with two columns : "Country Code" and " Country Name" in step 6. Thus we can delete "Country Code" from our original dataframe, because we can access them through the common column of '**df**' and '**dfcode**', " Country Name"
- On the other hand, both the "Indicator Code" and "Indicator Name" columns have values that are unique for them.
- We set the name of columns to ***"Energy use (kg of oil equivalent per capita)"*** in step 5. Then it seems reasonable to the delete these two columns from our dataframe as well.

In [112]:
```
df.drop(columns =["Country Code", "Indicator Name", "Indicator Code"]
df
```

Out[112]:

Energy use (kg of oil equivalent per capita)	Country Name	1960	1961	1962	1963	1964	1965	1966	1967	1968	1969	1970	
0	Aruba	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	Africa Eastern and Southern	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	7
2	Afghanistan	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	Africa Western and Central	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5
4	Angola	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6
...	...	...	...	...	...	...	...	...	...	...	...	...	
261	Kosovo	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
262	Yemen, Rep.	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1
263	South Africa	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	20
264	Zambia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	8
265	Zimbabwe	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	9

266 rows × 64 columns

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

- Values under the column "Country Name" are unique, then we can set them as our indexes in the dataframe.

In [115]:
```
df.set_index("Country Name", inplace = True)
df
```

Out[115]:

Energy use (kg of oil equivalent per capita) Country Name	1960	1961	1962	1963	1964	1965	1966	1967	1968	1969	1970	1971
Aruba	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Africa Eastern and Southern	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	787.535043
Afghanistan	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Africa Western and Central	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	511.982019
Angola	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	637.410306
...	...	...	...	...	...	...	...	...	...	...	...	...
Kosovo	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Yemen, Rep.	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	116.893021
South Africa	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2004.341886
Zambia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	835.864574
Zimbabwe	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	993.124493

266 rows × 63 columns

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

- There are some columns which have no values, so we can ignore them.
- These columns are : columns from 2016 to the end.

In [116]:
```
df.dropna(axis=1, how="all", inplace = True)
df
```

Out[116]:

Energy use (kg of oil equivalent per capita) / Country Name	1960	1961	1962	1963	1964	1965	1966	1967	1968	1969	1970	1971
Aruba	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Africa Eastern and Southern	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	787.535043
Afghanistan	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Africa Western and Central	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	511.982019
Angola	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	637.410306
...	...	...	...	...	...	...	...	...	...	...	...	...
Kosovo	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Yemen, Rep.	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	116.893021
South Africa	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2004.341886
Zambia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	835.864574
Zimbabwe	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	993.124493

266 rows × 56 columns

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

In [118]:
```
df.describe()
```

Out[118]:

Energy use (kg of oil equivalent per capita)	1960	1961	1962	1963	1964	196
count	31.000000	31.000000	31.000000	31.000000	31.000000	32.00000
mean	2385.557701	2420.522343	2521.248064	2656.031388	2764.165888	2808.61751
std	1997.388654	1981.575639	1971.311855	1992.182413	2089.070745	2058.79865
min	289.057068	322.490578	350.101258	367.811430	410.251827	440.88293
25%	1401.298789	1455.073445	1568.387025	1725.539544	1791.042783	1740.68913
50%	1906.174930	1937.644745	2081.011621	2268.427890	2341.146313	2435.74681
75%	2787.029421	2824.192050	2943.750064	3104.157719	3224.790698	3324.41380
max	10523.406695	10534.018211	10414.540920	10465.813021	11150.050343	10926.39517

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

- The row "***count***" shows the number of cells with values in each year. Here, these values are energy use. For example, it is clear that between 1960 to 1964 we have only 31 values for enrgy consumption for countries.
- "***mean***" shows the average of the energy consumption of all countries in each year.
- "***std***" shows the standard deviation of the energy consumption of all countries in each year.
- The rows "***min***" and "***max***" indicate the minimum and maximum energy consumption of all countries in each year.
- The distances between
  - "***min***" and "***25%***",
  - "***25%***" and "***50%***",
  - "***50%***" and "***75%***",
  - "***75%***" and "***max***" indicate the first to fourth quartiles of data(energy usage), respectively.
- "***50%***" shows the median of the energy consumption of all countries in each year. By comparing the value of this row by the mean's row we can conclude that our data on that year is left or right skewed. For instance, in our dataframe all energy consumption of all countries in each year are right skewed.

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

- The function basic_statistics get a serie as an input and returns the basic statistics of it.
- By applying this function to the dataframe while setting the axis=1 we will have basic statistics for each country.

In [119]:
```
def basic_statistics(serie):
    return serie.describe()
basic_stat = df.apply(basic_statistics, axis=1)
basic_stat
```

Out[119]:

Country Name	count	mean	std	min	25%	50%	75
Aruba	0.0	NaN	NaN	NaN	NaN	NaN	N
Africa Eastern and Southern	44.0	774.382105	33.741279	725.668196	747.688295	771.573635	789.4547
Afghanistan	0.0	NaN	NaN	NaN	NaN	NaN	N
Africa Western and Central	44.0	569.449117	23.837452	511.982019	560.362458	569.460212	580.0802
Angola	44.0	507.655185	57.699718	433.572486	466.935088	491.890571	536.4730
...	...	...	...	...	...	...	
Kosovo	15.0	1216.102185	133.739458	908.608824	1142.072713	1179.568641	1303.5480
Yemen, Rep.	43.0	223.657474	75.290318	90.216323	166.481024	221.497822	277.1749
South Africa	44.0	2486.783900	238.607050	1984.178909	2360.769433	2511.510478	2680.4802
Zambia	43.0	697.211345	92.585850	599.104868	615.968344	672.573813	771.0986
Zimbabwe	43.0	857.661556	66.749616	724.292388	822.144282	851.163079	893.3659

266 rows × 8 columns

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

In [120]:
```
basic_stat.describe()
```

Out[120]:

	count	mean	std	min	25%	50%	
count	266.000000	220.000000	219.000000	220.000000	220.000000	220.000000	220
mean	29.992481	1895.796955	456.997254	1105.604629	1617.384096	1894.713150	2200
std	20.445970	2368.170137	840.838073	1317.515592	1986.833783	2310.093780	2902
min	0.000000	12.400284	0.725879	9.548060	9.558898	11.760481	14
25%	5.000000	520.734943	72.313997	333.756270	465.168462	534.788916	562
50%	44.000000	999.663055	226.716363	538.671310	784.783249	985.088449	1132
75%	44.000000	2513.184092	515.951663	1442.352848	2257.972452	2602.494966	2892
max	56.000000	18278.542119	8751.700807	8775.992603	14525.570902	15664.302283	25096

14 - 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!)

- first we find the last year.
- then sort the data by the descending order
- then show only the first thirty rows.

In [121]:
```
year = df.columns[-1]
df.sort_values(year , ascending= False).head(30)
```

Out[121]:

Energy use (kg of oil equivalent per capita) Country Name	1960	1961	1962	1963	1964	
Iceland	3082.711563	2916.706232	3028.298369	3279.602269	3306.815957	3444.55
Canada	4251.435911	4307.820754	4451.560116	4694.120070	4903.607607	5153.93
North America	5516.355617	5494.086457	5654.544543	5869.044418	6024.094790	6201.77
United States	5641.740755	5612.079503	5774.586315	5986.783954	6136.938230	6307.89
Luxembourg	10523.406695	10534.018211	10414.540920	10465.813021	11150.050343	10926.39
Finland	2196.953067	2252.778690	2361.743876	2478.912009	2678.383166	2887.69
Norway	1906.174930	1937.644745	2052.054484	2182.993108	2320.013134	2539.59

Australia	3063.554271	3115.787084	3172.974865	3284.050959	3349.414167	3463.21
Korea, Rep.	NaN	NaN	NaN	NaN	NaN	
Sweden	2698.792303	2742.123469	2887.236252	3080.414075	3273.508965	3437.54
Post-demographic dividend	2812.288505	2848.671622	2969.271547	3127.901363	3237.046840	3322.74
Belgium	2519.497320	2570.815623	2810.061148	3043.306993	3021.647212	3116.03
High income	2761.770337	2799.712478	2918.228582	3069.538122	3175.606633	3260.61
New Zealand	1685.788431	1763.259908	1791.461322	1924.229007	2176.168871	2241.41
Netherlands	1825.934253	1879.150201	2081.011621	2268.427890	2349.083542	2493.15
Estonia	NaN	NaN	NaN	NaN	NaN	
OECD members	2666.979333	2702.066693	2815.604186	2959.832864	3060.624410	3142.69
Czech Republic	NaN	NaN	NaN	NaN	NaN	
Germany	1952.588632	1994.324633	2124.848539	2281.289928	2341.146313	2378.33
Austria	1546.261468	1554.034906	1675.873621	1823.995225	1855.085155	1851.84
France	1699.250872	1745.201546	1861.042066	1983.465817	2085.161125	2109.85
Japan	867.203098	971.998193	1013.257624	1147.292732	1272.480553	1374.52
Euro area	1403.942741	1455.391774	1567.862991	1686.600242	1762.650228	1816.95
European Union	1487.664770	1541.096121	1653.838779	1764.478845	1840.896534	1885.71
Slovenia	NaN	NaN	NaN	NaN	NaN	
Slovak Republic	NaN	NaN	NaN	NaN	NaN	
Switzerland	1398.654836	1454.755116	1568.911060	1869.587498	1819.435337	1943.60
Ireland	1318.812487	1396.466152	1412.729677	1453.962105	1522.194973	1511.90
Denmark	1922.973673	2023.308390	2296.289563	2502.927644	2600.155821	2822.73
Israel	NaN	NaN	NaN	NaN	NaN	

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

- We can conclude from the output that Iceland (17478.893037) had the highest consumption in 2015 Israel (2777.875324) had the lowest energy useage in that year between the 30 first countries.
- It is clear that Iceland as the first country in the list had consumption more than twice of the second country in the list Canada. But from Canada to the end of list, the consumption is almost one third in 2015.

16 - 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?

- While in subtraction the Nan is considered as zero, we can do the following.

In [122]:
```
diffrence = df["2010"] - df["2000"]
diffrence.sort_values(ascending=False).head(30)
```

Out[122]:
```
Country Name
Trinidad and Tobago                                 7343.881646
Iceland                                             5931.617954
Oman                                                2817.166090
Saudi Arabia                                        2028.716721
Gabon                                               1931.836600
Kazakhstan                                          1837.648729
Caribbean small states                              1773.999293
Kuwait                                              1569.504601
Brunei Darussalam                                   1180.641360
Turkmenistan                                        1164.474277
Norway                                              1108.512430
China                                               1055.735243
Korea, Rep.                                         1042.816465
Gibraltar                                           1037.278934
Iran, Islamic Rep.                                   894.791886
Estonia                                              848.028800
Late-demographic dividend                            791.572455
East Asia & Pacific (IDA & IBRD countries)           789.415631
East Asia & Pacific (excluding high income)          778.339797
Upper middle income                                  721.507825
East Asia & Pacific                                  707.439099
Luxembourg                                           652.641236
Thailand                                             605.452916
Middle East & North Africa                           597.518172
Russian Federation                                   594.763584
Bosnia and Herzegovina                               590.489037
Finland                                              566.591674
Latvia                                               530.193540
Kosovo                                               496.432062
Malaysia                                             493.635326
dtype: float64
```

Or we can do the above calculation as below as well:

In [123]:
```
diffrence = df["2010"].fillna(0) - df["2000"].fillna(0)
diffrence.sort_values(ascending=False).head(30)
```

Out[123]:
```
Country Name
Trinidad and Tobago                                 7343.881646
Iceland                                             5931.617954
Oman                                                2817.166090
Saudi Arabia                                        2028.716721
Gabon                                               1931.836600
Montenegro                                          1898.288421
Kazakhstan                                          1837.648729
Caribbean small states                              1773.999293
Kuwait                                              1569.504601
Brunei Darussalam                                   1180.641360
Turkmenistan                                        1164.474277
Norway                                              1108.512430
China                                               1055.735243
Korea, Rep.                                         1042.816465
Gibraltar                                           1037.278934
Iran, Islamic Rep.                                   894.791886
Estonia                                              848.028800
Late-demographic dividend                            791.572455
East Asia & Pacific (IDA & IBRD countries)           789.415631
East Asia & Pacific (excluding high income)          778.339797
Upper middle income                                  721.507825
East Asia & Pacific                                  707.439099
Luxembourg                                           652.641236
Thailand                                             605.452916
Middle East & North Africa                           597.518172
Russian Federation                                   594.763584
Bosnia and Herzegovina                               590.489037
Finland                                              566.591674
Latvia                                               530.193540
Kosovo                                               496.432062
dtype: float64
```

Conclution :

- As we can see Trinidad and Tobago had the highest growth rates between 2000 to 2010 in terms of energy consumption and the 30th country in this regard is Kosovo.
- All first 30 countries in the list had positive consumption growth rates from 2000 to 2010.
- However, I have to mention that in calculations of the difference between the two years, we considered the missing values as zero. Considering that the data for energy consumption is somehow a time series, it might better to execute a more proper way to filling missing values instead of fill them by zero.

17 - 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).

- The "mean_as_percent" function calculates the *mean* values by year as a percentage of the *maximal* values.
- By applying the function to the dataframe this function executes on all columns of the dataframe.

In [124]:
```
def mean_as_percent(column):
    value = (column.mean()/column.max())*100
    return value
df.apply(mean_as_percent)
```

Out[124]:
```
Energy use (kg of oil equivalent per capita)
1960    22.669063
1961    22.978148
1962    24.208922
1963    25.378166
1964    24.790614
1965    25.704887
1966    27.558484
1967    28.676396
1968    28.423581
1969    27.937783
1970    29.908848
1971     4.648137
1972     4.854237
1973     4.650674
1974     4.936191
1975     6.891470
1976     6.512473
1977     6.714076
1978     7.621300
1979     7.911123
1980     7.606008
1981     7.969176
1982     7.358451
1983     6.937944
1984    10.107683
1985    13.011385
1986    13.287579
1987    13.817290
1988    14.030779
1989    14.496496
1990    14.650733
1991    14.829843
1992    13.987773
```

```
1993    13.492196
1994    13.526670
1995    13.459529
1996    13.153490
1997    11.766205
1998    11.653551
1999    11.285838
2000    12.071856
2001    11.602816
2002    10.760776
2003    11.293554
2004     9.663841
2005    11.080832
2006    11.429581
2007    12.166881
2008    15.145194
2009    14.119852
2010    14.430814
2011    13.533453
2012    13.845241
2013    13.285970
2014    13.778170
2015    24.189451
dtype: float64
```

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

- It is wise to have an overview of countries that have no values for all years.
- This can be done as below.

In [125]:
```
Nan_values_allyears = df.isna().all(axis=1)
df.loc[Nan_values_allyears, :].index
```

Out[125]:
```
Index(['Aruba', 'Afghanistan', 'Andorra', 'American Samoa', 'Burundi'
,
       'Burkina Faso', 'Bermuda', 'Central African Republic',
       'Channel Islands', 'Cayman Islands', 'Faroe Islands',
       'Micronesia, Fed. Sts.', 'Guinea', 'Greenland', 'Guam', 'Isle
of Man',
       'Not classified', 'Lao PDR', 'Liberia', 'Low income', 'Liechte
nstein',
       'Macao SAR, China', 'St. Martin (French part)', 'Monaco', 'Mad
agascar',
       'Mali', 'Northern Mariana Islands', 'Mauritania', 'Malawi',
       'New Caledonia', 'Nauru', 'Papua New Guinea', 'Puerto Rico',
       'West Bank and Gaza', 'French Polynesia', 'Rwanda', 'Sierra Le
one',
       'San Marino', 'Somalia', 'Sint Maarten (Dutch part)',
       'Turks and Caicos Islands', 'Chad', 'Tuvalu', 'Uganda',
       'British Virgin Islands', 'Virgin Islands (U.S.)'],
      dtype='object', name='Country Name')
```

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

In [126]:
```
mask = df[df.columns[len(df.columns)-1]] > df[df.columns[len(df.columr
df.loc[mask]
```

Out[126]:

Energy use (kg of oil equivalent per capita) Country Name	1960	1961	1962	1963	1964	
Australia	3063.554271	3115.787084	3172.974865	3284.050959	3349.414167	3463.21
Belgium	2519.497320	2570.815623	2810.061148	3043.306993	3021.647212	3116.03
Canada	4251.435911	4307.820754	4451.560116	4694.120070	4903.607607	5153.93
Finland	2196.953067	2252.778690	2361.743876	2478.912009	2678.383166	2887.69
High income	2761.770337	2799.712478	2918.228582	3069.538122	3175.606633	3260.61
Iceland	3082.711563	2916.706232	3028.298369	3279.602269	3306.815957	3444.55
Korea, Rep.	NaN	NaN	NaN	NaN	NaN	
Luxembourg	10523.406695	10534.018211	10414.540920	10465.813021	11150.050343	10926.39
North America	5516.355617	5494.086457	5654.544543	5869.044418	6024.094790	6201.77
Netherlands	1825.934253	1879.150201	2081.011621	2268.427890	2349.083542	2493.15
Norway	1906.174930	1937.644745	2052.054484	2182.993108	2320.013134	2539.59
New Zealand	1685.788431	1763.259908	1791.461322	1924.229007	2176.168871	2241.41
Post-demographic dividend	2812.288505	2848.671622	2969.271547	3127.901363	3237.046840	3322.74
Sweden	2698.792303	2742.123469	2887.236252	3080.414075	3273.508965	3437.54
United States	5641.740755	5612.079503	5774.586315	5986.783954	6136.938230	6307.89

20 - 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 [127]:
```
capital = pd.read_json("capital.json")
capitaldf = pd.json_normalize(capital["data"])
capitaldf
```

Out[127]:

	name	capital	iso2	iso3
0	Afghanistan	Kabul	AF	AFG
1	Aland Islands	Mariehamn	AX	ALA
2	Albania	Tirana	AL	ALB
3	Algeria	Algiers	DZ	DZA
4	American Samoa	Pago Pago	AS	ASM
...	...	...	...	...
246	Wallis and Futuna	Mata Utu	WF	WLF
247	Western Sahara	El-Aaiun	EH	ESH
248	Yemen	Sanaa	YE	YEM
249	Zambia	Lusaka	ZM	ZMB
250	Zimbabwe	Harare	ZW	ZWE

251 rows × 4 columns

21 - 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.

- ***'dfnocode' containing 50 countries from 'dfcode' that their codes are not in 'iso3' column of 'capitaldf'.***
- ***So if we want to add a new country to 'capitaldf' dataframe from our original dataframe, we can check it from 'dfnocode' whethere that country is already in the 'capitaldf' or not.***

In [128]:
```
filt = dfcode["Country Code"].isin(capitaldf["iso3"])
dfnocode = dfcode.loc[~filt]
dfnocode
```

Out[128]:

	Energy use (kg of oil equivalent per capita)	Country Code	Country Name

1	AFE	Africa Eastern and Southern
3	AFW	Africa Western and Central
7	ARB	Arab World
36	CEB	Central Europe and the Baltics
38	CHI	Channel Islands
49	CSS	Caribbean small states
61	EAP	East Asia & Pacific (excluding high income)
62	EAR	Early-demographic dividend
63	EAS	East Asia & Pacific
64	ECA	Europe & Central Asia (excluding high income)
65	ECS	Europe & Central Asia
68	EMU	Euro area
73	EUU	European Union
74	FCS	Fragile and conflict affected situations
95	HIC	High income
98	HPC	Heavily indebted poor countries (HIPC)
102	IBD	IBRD only
103	IBT	IDA & IBRD total
104	IDA	IDA total
105	IDB	IDA blend
107	IDX	IDA only
110	INX	Not classified
128	LAC	Latin America & Caribbean (excluding high income)
134	LCN	Latin America & Caribbean
135	LDC	Least developed countries: UN classification
136	LIC	Low income
139	LMC	Lower middle income
140	LMY	Low & middle income
142	LTE	Late-demographic dividend
153	MEA	Middle East & North Africa
156	MIC	Middle income

161	MNA	Middle East & North Africa (excluding high inc...
170	NAC	North America
181	OED	OECD members
183	OSS	Other small states
191	PRE	Pre-demographic dividend
197	PSS	Pacific island small states
198	PST	Post-demographic dividend
204	SAS	South Asia
215	SSA	Sub-Saharan Africa (excluding high income)
217	SSF	Sub-Saharan Africa
218	SST	Small states
230	TEA	East Asia & Pacific (IDA & IBRD countries)
231	TEC	Europe & Central Asia (IDA & IBRD countries)
236	TLA	Latin America & the Caribbean (IDA & IBRD coun...
238	TMN	Middle East & North Africa (IDA & IBRD countries)
240	TSA	South Asia (IDA & IBRD)
241	TSS	Sub-Saharan Africa (IDA & IBRD countries)
249	UMC	Upper middle income
259	WLD	World

In [129]:
```
len(dfnocode)
```

Out[129]: 50

22 - 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.

In [130]:
```
condition = df.index.isin(dfnocode["Country Name"])
df.loc[~condition]
```

Out[130]:

Energy use (kg of oil equivalent per capita) Country Name	1960	1961	1962	1963	1964	1965	1966	1967	1968	1969	1970	1971
Aruba	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Afghanistan	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Angola	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	637.410306
Albania	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	785.161526
Andorra	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
Kosovo	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Yemen, Rep.	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	116.893021
South Africa	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2004.341886
Zambia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	835.864574
Zimbabwe	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	993.124493

216 rows × 56 columns

In [ ]:
```

```