Assignment 2 - Pandas

Tanvir, Ahmed, 20075186

The Story

Use Markdown cells to write a brief summary of the data analysis you are planning to undertake:

- What is the goal of this work?
- What kind of data is analyzed in this work?
- What summary statistics are obtained in this work?

This part is worth 3 marks. I recommend writing this part once you have completed all the remaining parts of this assignment.

Brief introduction and objective of the analysis

1. Constructed 2 dataframes using World Bank data.
2. The first DataFrame: 'dataframe_1' describes the Economy of the selected countries.
3. The second DataFrame: 'dataframe_2' describes the Energy consumption of the countries, access to electricity, how electricty is produced (fossil, renewable).
4. The goal was to identify which factors has the most correlation with renewable energy usage or shifts towards it.

Please note: that initially a few extra indicators were chosen to get a feel of the economy and the energy consumption but all of them weren't used in the analysis.

The data analysed was numeric, structured with proper labels.

Summary

No correlation was found between strength or size of an economy and the dependency on renewable sources fro electricity generation.

France and Canada are utilising green sources the most from my country list and Europeean countries have more inclination towards renewable energy.

Data Preparation

Countries

In [1]:
```
# Codes for the chosen countries

country_codes = ["CAN", "CHN", "DEU", "EGY", "FRA", "GBR", "IND", "JPN", "NGA", "USA", "ZAF"]
```

In [2]:
```
# Creating a dictionary in the country code:country name format:

# Step 1: Creating a list of country names in the same order as the country_codes list

country_proper_names = ['Canada', 'China', 'Egypt', 'France', 'Germany', 'India', 'Japan', 'Nigeria',
                        'South Africa', 'United Kingdom', 'United States']

country_names = {}
for i in range(0,len(country_codes)):
    country_names[country_codes[i]] = country_proper_names[i]

# The final dictionary
country_names
```

Out[2]:
```
{'CAN': 'Canada',
 'CHN': 'China',
 'DEU': 'Egypt',
 'EGY': 'France',
 'FRA': 'Germany',
 'GBR': 'India',
 'IND': 'Japan',
 'JPN': 'Nigeria',
 'NGA': 'South Africa',
 'USA': 'United Kingdom',
 'ZAF': 'United States'}
```

In [3]:
```python
# Grouping the countries in their respective continents in a dictionary

country_groups = {'EGY':'Africa', 'NGA':'Africa', 'ZAF':'Africa', 'CHN':'Asia', 'IND':'Asia', 'JPN':'Asia',
                  'FRA':'Europe', 'DEU':'Europe', 'GBR':'Europe', 'CAN':'North America', 'USA': 'North America'}

country_groups
```

Out[3]:
```
{'EGY': 'Africa',
 'NGA': 'Africa',
 'ZAF': 'Africa',
 'CHN': 'Asia',
 'IND': 'Asia',
 'JPN': 'Asia',
 'FRA': 'Europe',
 'DEU': 'Europe',
 'GBR': 'Europe',
 'CAN': 'North America',
 'USA': 'North America'}
```

Indicators

In [4]:
```python
import numpy as np
import pandas as pd
import wbgapi as wb
import matplotlib.pyplot as plt
%matplotlib inline
```

In [5]:
```python
# Creating a list of Indicator IDs for my first DataFrame

indicator_ids_1 = ['SP.POP.TOTL', 'SL.TLF.TOTL.IN', 'NY.GDP.MKTP.CD', 'NY.GDP.MKTP.KD.ZG', 'GC.DOD.TOTL.GD.ZS',
                   'FI.RES.TOTL.CD', 'BX.GSR.GNFS.CD', 'BM.GSR.GNFS.CD']

# Indicator IDs (indicator_ids_2) for my second DataFrame is done in a similar method
```

DataFrames

In [6]:
```python
# Creating a Pandas DataFrame from World Bank data

my_dataframe_1 = wb.data.DataFrame(indicator_ids_1, country_codes, time=range(2011, 2016))

#replacing most recent 5 years mrv=5 with time for chosen years

df = my_dataframe_1.unstack().stack(level=0) # using unstack and stack method to get the dataframe to my desired shape

# unstack() takes the indicators from being subcategories in the rows under country names to subcategories of year colum

# applying stack() again on level = 0 takes the year columns to a sublevel of rows

# How it looks
df.head(10)
```

Out[6]:

	series	BM.GSR.GNFS.CD	BX.GSR.GNFS.CD	FI.RES.TOTL.CD	GC.DOD.TOTL.GD.ZS	NY.GDP.MKTP.CD	NY.GDP.MKTP.KD.ZG	SL.TLF.TOTL.I
economy								
CAN	YR2011	5.684596e+11	5.467770e+11	6.581899e+10	NaN	1.793327e+12	3.146881	19147395
	YR2012	5.894798e+11	5.549615e+11	6.854634e+10	NaN	1.828366e+12	1.762223	19322866
	YR2013	5.890646e+11	5.600825e+11	7.193709e+10	NaN	1.846597e+12	2.329123	19546552
	YR2014	5.896265e+11	5.733055e+11	7.469996e+10	NaN	1.805750e+12	2.870036	19629145
	YR2015	5.347210e+11	4.961373e+11	7.975352e+10	NaN	1.556509e+12	0.659177	19747709
CHN	YR2011	1.826949e+12	2.008852e+12	3.254674e+12	NaN	7.551500e+12	9.550832	778977720
	YR2012	1.943247e+12	2.175092e+12	3.387513e+12	NaN	8.532230e+12	7.863736	782865417
	YR2013	2.120215e+12	2.355595e+12	3.880368e+12	NaN	9.570406e+12	7.766150	786673270
	YR2014	2.241603e+12	2.462902e+12	3.900039e+12	NaN	1.047568e+13	7.425764	791323527
	YR2015	2.002282e+12	2.360152e+12	3.405253e+12	NaN	1.106155e+13	7.041329	795251107

In [7]:
```python
# Multiindexing the rows

dataframe_1 = df.iloc[:, ::-1]

index = pd.MultiIndex.from_product([country_codes, [2011, 2012, 2013, 2014, 2015]],
                                   names=['Country', 'Year'])

dataframe_1.index = index
```

In [8]:
```python
# Multiindexing the columns
dataframe_1.columns = pd.MultiIndex.from_tuples([('Population', 'Total'), ('Population', 'Total labor force'),
                                                 ('GDP','Growth (annual %)'), ('GDP', 'Gross (USD)'),
                                                 ('Economic strength', 'Central government debt (% of GDP)'),
                                                 ('Economic strength', 'Total reserves (USD)'),
                                                 ('Commerce', 'Exports (USD)'), ('Commerce', 'Imports (USD)')])
```

In [9]:
```python
# Re-arranging the columns using a variable called 't'
t = list(dataframe_1.columns)     # creates a list of column names

# I want to swap positions of column 3 and 4
t[2], t[3] = t[3], t[2]

dataframe_1 = dataframe_1[t]

"""
This can also be manually done as below.

dataframe_1 = dataframe_1[[('Population', 'Total'), ('Population', 'Total labor force'),
                           ('GDP', 'Gross (USD)'), ('GDP','Growth (annual %)'),
                           ('Economic strength', 'Central government debt (% of GDP)'),
                           ('Economic strength', 'Total reserves (USD)'),
                           ('Commerce', 'Exports (USD)'), ('Commerce', 'Imports (USD)')]]
"""
```

Out[9]:
```
"\nThis can also be manually done as below.\n\ndataframe_1 = dataframe_1[[('Population', 'Total'), ('Population', 'Tota
l labor force'),\n                           ('GDP', 'Gross (USD)'), ('GDP','Growth (annual %)'), \n
('Economic strength', 'Central government debt (% of GDP)'), \n                           ('Economic strength', 'Total
reserves (USD)'), \n                           ('Commerce', 'Exports (USD)'), ('Commerce', 'Imports (USD)')]]\n"
```

In [10]:
```python
# My first DataFrame

dataframe_1.head(10)
```

Out[10]:

		Population		GDP		Economic strength		Commerce	
		Total	Total labor force	Gross (USD)	Growth (annual %)	Central government debt (% of GDP)	Total reserves (USD)	Exports (USD)	Imports (USD)
Country	**Year**								
CAN	**2011**	3.433933e+07	19147395.0	1.793327e+12	3.146881	NaN	6.581899e+10	5.467770e+11	5.684596e+11
	2012	3.471422e+07	19322866.0	1.828366e+12	1.762223	NaN	6.854634e+10	5.549615e+11	5.894798e+11
	2013	3.508295e+07	19546552.0	1.846597e+12	2.329123	NaN	7.193709e+10	5.600825e+11	5.890646e+11
	2014	3.543744e+07	19629145.0	1.805750e+12	2.870036	NaN	7.469996e+10	5.733055e+11	5.896265e+11
	2015	3.570291e+07	19747709.0	1.556509e+12	0.659177	NaN	7.975352e+10	4.961373e+11	5.347210e+11
CHN	**2011**	1.345035e+09	778977720.0	7.551500e+12	9.550832	NaN	3.254674e+12	2.008852e+12	1.826949e+12
	2012	1.354190e+09	782865417.0	8.532230e+12	7.863736	NaN	3.387513e+12	2.175092e+12	1.943247e+12
	2013	1.363240e+09	786673270.0	9.570406e+12	7.766150	NaN	3.880368e+12	2.355595e+12	2.120215e+12
	2014	1.371860e+09	791323527.0	1.047568e+13	7.425764	NaN	3.900039e+12	2.462902e+12	2.241603e+12
	2015	1.379860e+09	795251107.0	1.106155e+13	7.041329	NaN	3.405253e+12	2.360152e+12	2.002282e+12

In [11]:
```python
# The dataframe has a column with NaN values (there are a few inputs in this column though, let's see if it'll be useful,

# Please note:
# some of the columns will be excluded in this analysis, they're just presented for informational purposes
# and possible exploratory data analysis

# I'll keep this dataframe as is for now and drop the columns not necessary as we as the NaN column when needed.
```

In [12]:

```
# Creating the second DataFrame following the same steps as above

indicator_ids_2 = ['SP.POP.TOTL', 'EG.ELC.ACCS.ZS', 'EG.USE.ELEC.KH.PC', 'EG.ELC.LOSS.ZS',
                   'EG.ELC.FOSL.ZS', 'EG.ELC.RNWX.ZS', 'EG.ELC.HYRO.ZS', 'EG.ELC.NUCL.ZS']

my_dataframe_2 = wb.data.DataFrame(indicator_ids_2, country_codes, time=range(2011, 2016))

df2 = my_dataframe_2.unstack().stack(level=0) # using unstack and stack method to get the dataframe to my desired shape

dataframe_2 = df2.iloc[:, ::-1]

index_2 = pd.MultiIndex.from_product([country_codes, [2011, 2012, 2013, 2014, 2015]],
                                   names=['Country', 'Year'])

dataframe_2.index = index_2

# How the dataframe looks like
dataframe_2.head(10)
```

Out[12]:

	series	SP.POP.TOTL	EG.USE.ELEC.KH.PC	EG.ELC.RNWX.ZS	EG.ELC.NUCL.ZS	EG.ELC.LOSS.ZS	EG.ELC.HYRO.ZS	EG.ELC.FOSL.ZS	EG.E
Country	**Year**								
CAN	**2011**	3.433933e+07	15644.540278	3.298016	14.707805	8.800576	59.040234	22.543932	
	2012	3.471422e+07	15336.624857	3.507259	14.900157	8.438532	59.723302	21.424611	
	2013	3.508295e+07	15750.811633	4.409954	15.549008	8.466956	58.888079	20.769341	
	2014	3.543744e+07	15588.487146	5.570376	16.119075	8.711767	57.254617	20.763125	
	2015	3.570291e+07	NaN	6.267257	15.546561	NaN	56.744193	21.067180	
CHN	**2011**	1.345035e+09	3295.784868	2.137640	1.835336	5.740233	14.624130	81.174003	
	2012	1.354190e+09	3466.019539	2.657515	1.953846	5.810062	17.308734	77.859893	
	2013	1.363240e+09	3757.185088	3.564878	2.053005	5.777010	16.731349	77.424467	
	2014	1.371860e+09	3905.317598	4.056660	2.339286	5.471266	18.552494	74.822887	
	2015	1.379860e+09	NaN	4.857004	NaN	NaN	19.069813	72.962076	

In [13]:
```python
# Multiindexing the columns again

dataframe_2.columns = pd.MultiIndex.from_tuples([('Population', 'Total'),
                                                  ('Electricity T&D', 'Electricity consumption (kWh/capita)'),
                                                  ('Electricity production source (% of total)','Solar & Wind'),
                                                  ('Electricity production source (% of total)','Nuclear'),
                                                  ('Electricity T&D', 'Trans & Dist loss (% of output)'),
                                                  ('Electricity production source (% of total)','Hydro'),
                                                  ('Electricity production source (% of total)','Fossil fuels'),
                                                  ('Population', 'Access to electricity (% of population)')])

dataframe_2 = dataframe_2[[('Population', 'Total'),
                           ('Population', 'Access to electricity (% of population)'),
                            ('Electricity T&D', 'Electricity consumption (kWh/capita)'),
                            ('Electricity T&D', 'Trans & Dist loss (% of output)'),
                            ('Electricity production source (% of total)','Solar & Wind'),
                            ('Electricity production source (% of total)','Nuclear'),
                            ('Electricity production source (% of total)','Hydro'),
                            ('Electricity production source (% of total)','Fossil fuels')]]

dataframe_2.head(10)
```

Out[13]:

		Population		**Electricity T&D**		**Electricity production source (% of total)**			
		Total	**Access to electricity (% of population)**	**Electricity consumption (kWh/capita)**	**Trans & Dist loss (% of output)**	**Solar & Wind**	**Nuclear**	**Hydro**	**Fossil fuels**
Country	**Year**								
CAN	**2011**	3.433933e+07	100.000000	15644.540278	8.800576	3.298016	14.707805	59.040234	22.543932
	2012	3.471422e+07	100.000000	15336.624857	8.438532	3.507259	14.900157	59.723302	21.424611
	2013	3.508295e+07	100.000000	15750.811633	8.466956	4.409954	15.549008	58.888079	20.769341
	2014	3.543744e+07	100.000000	15588.487146	8.711767	5.570376	16.119075	57.254617	20.763125
	2015	3.570291e+07	100.000000	NaN	NaN	6.267257	15.546561	56.744193	21.067180
CHN	**2011**	1.345035e+09	99.848724	3295.784868	5.740233	2.137640	1.835336	14.624130	81.174003
	2012	1.354190e+09	99.961929	3466.019539	5.810062	2.657515	1.953846	17.308734	77.859893
	2013	1.363240e+09	99.996445	3757.185088	5.777010	3.564878	2.053005	16.731349	77.424467

		Population		Electricity T&D		Electricity production source (% of total)			
		Total	Access to electricity (% of population)	Electricity consumption (kWh/capita)	Trans & Dist loss (% of output)	Solar & Wind	Nuclear	Hydro	Fossil fuels
Country	Year								
	2014	1.371860e+09	100.000000	3905.317598	5.471266	4.056660	2.339286	18.552494	74.822887
	2015	1.379860e+09	100.000000	NaN	NaN	4.857004	NaN	19.069813	72.962076

Data Analysis

Use Pandas `groupby()` and `pivot_table()` methods to construct 8 different summary statistics. They must include the following Pandas techniques:

- `groupby()` combined with `aggregate()` , `filter()` , `transform()` , and `apply()` methods.
- `groupby()` using an external key, the dictionary `country_groups` you have constructed above.
- at least one summary statistics must use the `pivot_table()` method.
- at least two summary statistics must use data from both DataFrames.

The necessary Pandas techniques are explained in Notebooks 2.8 and 2.9.

Important: Make sure your summary statistics make sense and tell a story. This story must be summarized in the first part of this assignment, "The Story".

This part is worth 10 marks: 1 mark for Python code for each summary statistic and 2 marks for comments explaining the Python code and the summary statistics.

In [14]:
```
# Application of groupby

gdp_max = dataframe_1.groupby(level='Country')[[('GDP', 'Gross (USD)')]].mean()
gdp_max.sort_values(by=[('GDP', 'Gross (USD)')], ascending=False)
```

Out[14]:

	GDP
	Gross (USD)
Country	
USA	1.685798e+13
CHN	9.438274e+12
JPN	5.411953e+12
DEU	3.651388e+12
GBR	2.848216e+12
FRA	2.731172e+12
IND	1.930025e+12
CAN	1.766110e+12
NGA	4.805336e+11
ZAF	4.042793e+11
EGY	2.877005e+11

USA, China and Japan had higher avearge GDP than the rest of the countries between 2011 and 2015

In [15]:
```
# GDP growth of the countries using groupby.filter()

# Filter by average GDP growth more than 5%

#dataframe_1.groupby('Country').filter(lambda x: x[('GDP','Growth (annual %)')].mean() > 3)

growth = dataframe_1.groupby('Country').filter(lambda x: x[('GDP','Growth (annual %)')].mean() > 5)

growth.groupby('Country')[[('GDP','Growth (annual %)')]].mean().sort_values([('GDP','Growth (annual %)')], ascending=Fals
```

Out[15]:

	GDP
	Growth (annual %)
Country	
CHN	7.929562
IND	6.498058
NGA	5.034347

China's GDP was the fastest growing between 2011-2015

In [16]:

```python
# The percentage of the labor force in the countries apply() method

df_3 = dataframe_1[[('Population', 'Total')]].droplevel(level=0, axis=1)
df_3 = df_3.reset_index()

df_4 = dataframe_1[[('Population', 'Total labor force')]].droplevel(level=0, axis=1)
df_4 = df_4.reset_index()

def ratio(x):
    x['Total labor force'] /= df_3['Total']/100
    return x

labor_force_percentage = df_4.groupby('Country').apply(ratio)

labor_force_percentage.columns = ['Country', 'Year', 'labor force percetage']

labor_force_percentage.groupby('Country')['labor force percetage'].mean()
```

Out[16]:

```
Country
CAN    55.567898
CHN    57.749414
DEU    52.180651
EGY    32.302633
FRA    45.881623
GBR    51.592087
IND    36.040963
JPN    51.508466
NGA    31.700477
USA    50.113420
ZAF    37.488959
Name: labor force percetage, dtype: float64
```

China, Canada, Germany, United Kingdom, Japan and the US have more than 50 percent of their population into the workforce

In [17]:
```python
# Application of groupby.aggregate method

export_by_countries = dataframe_1.groupby(level='Country')[[('Commerce', 'Exports (USD)')]].aggregate(['min',
                                                                                                      np.mean,
                                                                                                      max])
export_by_countries.sort_values(by=[('Commerce', 'Exports (USD)', 'max')], ascending=False)
```

Out[17]:

	Commerce		
	Exports (USD)		
	min	mean	max
Country			
CHN	2.008852e+12	2.272519e+12	2.462902e+12
USA	2.143556e+12	2.275359e+12	2.392613e+12
DEU	1.575247e+12	1.672834e+12	1.773618e+12
JPN	7.847108e+11	8.644595e+11	9.306604e+11
GBR	8.037030e+11	8.284525e+11	8.679430e+11
FRA	7.775447e+11	8.173200e+11	8.535030e+11
CAN	4.961373e+11	5.462527e+11	5.733055e+11
IND	4.286309e+11	4.545416e+11	4.855830e+11
ZAF	9.634652e+10	1.130611e+11	1.269350e+11
NGA	4.904777e+10	8.680304e+10	1.024375e+11
EGY	3.756940e+10	4.503704e+10	4.860130e+10

In [18]:
```
import_by_countries = dataframe_1.groupby(level='Country')[[('Commerce', 'Imports (USD)')]].aggregate(['min',
                                                                                                        np.mean,
                                                                                                        max])

import_by_countries.sort_values(by=[('Commerce', 'Imports (USD)', 'max')], ascending=False)
```

Out[18]:

	Commerce		
	Imports (USD)		
	min	mean	max
Country			
USA	2.698073e+12	2.775889e+12	2.876564e+12
CHN	1.826949e+12	2.026859e+12	2.241603e+12
DEU	1.320209e+12	1.446217e+12	1.515877e+12
JPN	8.079867e+11	9.480099e+11	1.014813e+12
GBR	8.475310e+11	8.679509e+11	9.221976e+11
FRA	7.873782e+11	8.515478e+11	8.897318e+11
CAN	5.347210e+11	5.742703e+11	5.896265e+11
IND	4.918801e+11	5.477624e+11	5.799086e+11
ZAF	1.008028e+11	1.172173e+11	1.235595e+11
NGA	7.194744e+10	8.134336e+10	9.079363e+10
EGY	6.138110e+10	6.748850e+10	7.399600e+10

In [19]:
```
profit = export_by_countries[('Commerce', 'Exports (USD)', 'mean')] - import_by_countries[('Commerce', 'Imports (USD)',
profit.sort_values(ascending=False)
```

Out[19]:
```
Country
CHN    2.456596e+11
DEU    2.266173e+11
NGA    5.459673e+09
ZAF   -4.156178e+09
EGY   -2.245146e+10
CAN   -2.801757e+10
FRA   -3.422773e+10
GBR   -3.949841e+10
JPN   -8.355042e+10
IND   -9.322084e+10
USA   -5.005302e+11
dtype: float64
```

China, USA and Germany are the top 3 exporters and importers of good and services among the countries.

China, Germany and Nigeria are making profits.

In [20]:
```python
# Total reserves by continents using dataframe_1

# To groupby() using the dictionary country_groups (an external key) I need to reset the index of the multi-indexed data

reset_df1 = dataframe_1.reset_index()

#print(reset_df)

# Now setting the index to the newly created column 'Country' assigning the value to a new dataframe
country_idx_df_1 = reset_df1.set_index(['Country'])

#reset_df_groupby = reset_df.groupby(country_groups)[[('Economic strength', 'Total reserves (USD)')]].sum()
#reset_df_groupby

#calling groupby on the new dataframe q to use country_groups

reserves_by_continent = country_idx_df_1.groupby(country_groups)[[('Economic strength', 'Total reserves (USD)')]].sum()

reserves_by_continent
```

Out[20]:

	Economic strength
	Total reserves (USD)
Africa	5.076284e+11
Asia	2.572806e+13
Europe	2.447299e+12
North America	2.738945e+12

In [21]:
```
reserves_list = dataframe_1.groupby('Country')[[('Economic strength', 'Total reserves (USD)')]].max()

reserves_list.sort_values(by=[('Economic strength', 'Total reserves (USD)')], ascending=False)
```

Out[21]:

	Economic strength
	Total reserves (USD)
Country	
CHN	3.900039e+12
JPN	1.295839e+12
USA	5.742681e+11
IND	3.533191e+11
DEU	2.488565e+11
FRA	1.845218e+11
GBR	1.481093e+11
CAN	7.975352e+10
ZAF	5.068808e+10
NGA	4.383064e+10
EGY	1.863754e+10

The selected Asian countries have the highest reserves than the rest due to China and Japan having the most amount of reserves at the top 2 position on the table.

In [22]:
```python
# Which continents were utilising the most solar and wind [dataframe_2]?

country_idx_df_2 = dataframe_2.reset_index()

country_idx_df_2 = country_idx_df_2.set_index(['Country'])

# Groupby below using country_groups

solar_wind = country_idx_df_2.groupby(country_groups)[[('Electricity production source (% of total)','Solar & Wind')]].m

solar_wind
```

Out[22]:

	Electricity production source (% of total)
	Solar & Wind
Africa	0.536075
Asia	4.535155
Europe	13.502376
North America	5.393571

Europe has larger proportion of its electricty production by Solar and Wind energies.

In [23]:
```
# Which countries used more renewable sources to generate electricity than fossil fuels?

renewable = dataframe_2[[('Electricity production source (% of total)','Solar & Wind'),
                                 ('Electricity production source (% of total)','Nuclear'),
                                 ('Electricity production source (% of total)','Hydro')]]


renewable[('Electricity production source (% of total)','Fossil')] = dataframe_2[[('Electricity production source (% of 
                                                                                    'Fossil fuels')]]
#removing multi-index from the columns

renewable = renewable.droplevel(level=0, axis=1)


renewable = renewable.reset_index()

renewable['Sum of renewable (% of total electricity prod.)'] = renewable[['Solar & Wind', 'Nuclear', 'Hydro']].sum(axis=
```

```
<ipython-input-23-07ac4d89d187>:8: 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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py)
  renewable[('Electricity production source (% of total)','Fossil')] = dataframe_2[[('Electricity production source (%
 of total)',
```

In [24]:
```python
renewable.set_index('Country')

renewable.groupby('Country')[['Sum of renewable (% of total electricity prod.)']].mean().sort_values(by=
                              ['Sum of renewable (% of total electricity prod.)'], ascending=False)
```

Out[24]:

Country	Sum of renewable (% of total electricity prod.)
FRA	92.360655
CAN	78.305179
DEU	40.370703
GBR	35.661474
USA	31.704813
CHN	22.348338
NGA	19.140297
IND	18.718404
JPN	15.535208
EGY	8.843665

France and Canada generate most of their electricity from renewable sources

In [25]:
```
# Using pivot_table() method

fossil_fuel_use = dataframe_2.droplevel(level=0 ,axis=1).pivot_table('Fossil fuels', index='Country', aggfunc='mean')

fossil_fuel_use.sort_values(by='Fossil fuels', ascending=False)
```

Out[25]:

Country	Fossil fuels
ZAF	93.674061
EGY	91.156335
NGA	80.859703
IND	80.685692
JPN	79.925226
CHN	76.848665
USA	67.929200
GBR	63.655315
DEU	58.223146
CAN	21.313638
FRA	7.177624

Most countries rely very heavily on fossil fuels for electricity production with the only exception of Canada and France as seen in the previous summary.

In [26]:
```python
# Population vs Energy Consumption

df_comparison_1 = dataframe_1.loc[:, ('Population', 'Total')]
comparison_1 = pd.DataFrame(df_comparison_1).join(pd.DataFrame(dataframe_2.loc[:, ('Electricity T&D',
                                                                       'Electricity consumption (kWh/capita)')]))

comp = comparison_1.groupby('Country')[[('Population', 'Total'),
                                        ('Electricity T&D', 'Electricity consumption (kWh/capita)')]].max()

comp = comp.sort_values(('Electricity T&D', 'Electricity consumption (kWh/capita)'), ascending=False)

comp
```

Out[26]:

	Population	Electricity T&D
	Total	**Electricity consumption (kWh/capita)**
Country		
CAN	3.570291e+07	15750.811633
USA	3.207390e+08	13245.881928
JPN	1.278330e+08	8099.598695
FRA	6.654827e+07	7367.843768
DEU	8.168661e+07	7281.272174
GBR	6.511622e+07	5471.933475
ZAF	5.538637e+07	4566.323754
CHN	1.379860e+09	3905.317598
EGY	9.244255e+07	1685.818794
IND	1.310152e+09	804.516349
NGA	1.811375e+08	156.797152

Per capita electricity usage is very high in Canada and the US. The table shows that population doesn't have any impact on the energy consumption.

In [27]:
```
# GDP vs Sustainable energy: using 2 dataframes

df_comparison_2 = dataframe_1.loc[:, ('GDP', 'Gross (USD)')]

comparison_2 = pd.DataFrame(df_comparison_1).join(pd.DataFrame(dataframe_2.loc[:,
                                                        [('Electricity production source (% of total)','Solar & Wind
                                                        ('Electricity production source (% of total)','Nuclear'),
                                                        ('Electricity production source (% of total)','Hydro')]]))

comparison_2
```

Out[27]:

		Population	Electricity production source (% of total)		
		Total	Solar & Wind	Nuclear	Hydro
Country	Year				
CAN	2011	3.433933e+07	3.298016	14.707805	59.040234
	2012	3.471422e+07	3.507259	14.900157	59.723302
	2013	3.508295e+07	4.409954	15.549008	58.888079
	2014	3.543744e+07	5.570376	16.119075	57.254617
	2015	3.570291e+07	6.267257	15.546561	56.744193
CHN	2011	1.345035e+09	2.137640	1.835336	14.624130
	2012	1.354190e+09	2.657515	1.953846	17.308734
	2013	1.363240e+09	3.564878	2.053005	16.731349
	2014	1.371860e+09	4.056660	2.339286	18.552494

In [28]:

```
#Using transform() method

#normalised the column by dvinding the max value for each category

comparison_2.groupby('Country').transform(lambda x: x/x.max())
```

Out[28]:

		Population	Electricity production source (% of total)		
		Total	Solar & Wind	Nuclear	Hydro
Country	**Year**				
CAN	**2011**	0.961808	0.526230	0.912447	0.988563
	2012	0.972308	0.559616	0.924380	1.000000
	2013	0.982636	0.703650	0.964634	0.986015
	2014	0.992564	0.888806	1.000000	0.958665
	2015	1.000000	1.000000	0.964482	0.950118
CHN	**2011**	0.974762	0.440115	0.784571	0.766873
	2012	0.981397	0.547151	0.835232	0.907651
	2013	0.987955	0.733966	0.877620	0.877374
	2014	0.994202	0.835219	1.000000	0.972872

The normalised dataframe above shows that use of Solar and Wind energy had been gradually increasing in all countries except Egypt. No data on Nigeria is available for this category.

In [29]:
```
comparison_2.groupby('Country')[[('Electricity production source (% of total)', 'Solar & Wind')]].mean().round(3)
```

Out[29]:

	Electricity production source (% of total)
	Solar & Wind
Country	
CAN	4.611
CHN	3.455
DEU	21.335
EGY	0.960
FRA	4.767
GBR	14.405
IND	4.819
JPN	5.332
NGA	0.000
USA	6.177
ZAF	0.648

Germany is leading in harnessing solar and wind energy follwed by the Unied Kingdom.