Assignment 3 - More Pandas

Question 1 (20%)

Load the energy data from the file `Energy Indicators.xls`, which is a list of indicators of energy supply and renewable electricity production from the United Nations for the year 2013, and should be put into a DataFrame with the variable name of **energy**.

Keep in mind that this is an Excel file, and not a comma separated values file. Also, make sure to exclude the footer and header information from the datafile. The first two columns are unneccessary, so you should get rid of them, and you should change the column labels so that the columns are:

`['Country', 'Energy Supply', 'Energy Supply per Capita', '% Renewable's]`

Convert `Energy Supply` to gigajoules (there are 1,000,000 gigajoules in a petajoule). For all countries which have missing data (e.g. data with "...") make sure this is reflected as `np.NaN` values.

Rename the following list of countries (for use in later questions):

```
"Republic of Korea": "South Korea",
   "United States of America": "United States",
   "United Kingdom of Great Britain and Northern Ireland": "United Kingdom",
   "China, Hong Kong Special Administrative Region": "Hong Kong"
```

There are also several countries with parenthesis in their name. Be sure to remove these, e.g. `'Bolivia (Plurinational State of)'` should be `'Bolivia'`.

Next, load the GDP data from the file `worldbank.csv`, which is a csv containing countries' GDP from 1960 to 2015 from World Bank. Call this DataFrame **GDP**.

Make sure to skip the header, and rename the following list of countries:

```
"Korea, Rep.": "South Korea",
"Iran, Islamic Rep.": "Iran",
"Hong Kong SAR, China": "Hong Kong"
```

Finally, load the Sciamgo Journal and Country Rank data for Energy Engineering and Power Technology from the file `scimagojr-3.xlsx`, which ranks countries based on their journal contributions in the aforementioned area. Call this DataFrame **ScimEn**.

Join the three datasets: GDP, Energy, and ScimEn into a new dataset (using the intersection of country names). Use only the last 10 years (2006-2015) of GDP data and only the top 15 countries by Scimagojr 'Rank' (Rank 1 through 15).

The index of this DataFrame should be the name of the country, and the columns should be ['Rank', 'Documents', 'Citable documents', 'Citations', 'Self-citations', 'Citations per document', 'H index', 'Energy Supply', 'Energy Supply per Capita', '% Renewable', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015'].

This function should return a DataFrame with 20 columns and 15 entries.

In [1]:

```
import pandas as pd
import numpy as np
```

In [10]:

```

##FIRST DATA FRAME
##open file and put into variable. pd.read_excel('Energy Indicators.xls', sheetname='Energy') work
s too
energy = pd.read_excel(open('Energy Indicators.xls', 'rb'), sheetname='Energy')
##drop first two columns. axis=0 for row, 1 for column
energy=energy.drop(energy.columns[[0,1]], axis=1)
```

```
energy=energy.drop(energy.columns[[0,1]], axis=1)
##drop header and footer by just taking the relevant rows with values
energy=energy.iloc[16:242]
##rename columns, Also works: energy.columns=['Country', 'Energy Supply', 'Energy Supply per Capit
a', '% Renewable']
for col in energy.columns:
    if col=='Environmental Indicators: Energy':
        energy.rename(columns={col:'Country'}, inplace=True)
    if col=='Unnamed: 3':
        energy.rename(columns={col: 'Energy Supply'}, inplace=True)
    if col=='Unnamed: 4':
        energy.rename(columns={col: 'Energy Supply per Capita'}, inplace=True)
    if col=='Unnamed: 5':
        energy.rename(columns={col: '% Renewable'}, inplace=True)
##reset index so that it starts at 0 instead of 16 and delete the 'index' column that results and
has the old humbers
energy=energy.reset_index()
del energy['index']
##where there is a '...' change it to NaN
energy['Energy Supply per Capita']=energy['Energy Supply per Capita'].replace('...', value=np.NaN)
energy['Energy Supply']=energy['Energy Supply'].replace('...', value=np.NaN)
energy['% Renewable']=energy['% Renewable'].astype(np.float64)
##convert petajoules to gigajoules
energy['Energy Supply']*=1000000
## set [condition: where Country value starts with x, Look at: Country column for that record] = t
o a value
energy.loc[energy['Country'].str.startswith('United States of America'), ['Country']]='United
States'
energy.loc[energy['Country'].str.startswith('Republic of Korea'), ['Country']]='South Korea'
energy.loc[energy['Country'].str.startswith('United Kingdom of Great Britain and Northern Ireland'
), ['Country']]='United Kingdom'
energy.loc[energy['Country'].str.startswith('China, Hong Kong Special Administrative Region'),
['Country']]='Hong Kong'
##any value in country column that ends with a ")" replace anything in parentheses with nothing (d
elete it)
##white space, escape special charachters for an actual (, any charachter of any number,
##escape special charachters for an actual ) and replace with nothing
energy.loc[energy['Country'].str.endswith(')'), ['Country']]=energy['Country'].str.replace(" \(.*\)
","")
##have to take the numbers off of country names otherwise merge won't work properly 'China' != 'Ch
ina2'
##replace any number of digits with nothing
energy['Country'] = energy['Country'].str.replace('\d+', '')


##SECOND DATA FRAME

##load file and skip header. 'world_bank.csv' is not name of file as saved on local. It's file nam
e that works in Jupyter
GDP=pd.read_csv('worldbank.csv', encoding = 'ISO-8859-1', skiprows=4)
##replace specific values in Country Name column
GDP['Country Name']=GDP['Country Name'].replace(['Korea, Rep.', 'Iran, Islamic Rep.', 'Hong Kong SA
R, China'], value=['South Korea', 'Iran', 'Hong Kong'])
##take only last 10 years of GDP data, along with 'Country Name'
GDP.rename(columns={'Country Name':'Country'}, inplace=True)
GDP=GDP[['Country', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015']
]

##THIRD DATA FRAME
ScimEn=pd.read_excel('scimagojr.xlsx')


##JOIN THE DATA SETS

##merge in proper order.
merged=pd.merge(GDP, energy, how='inner', on='Country')
merged=pd.merge(merged, ScimEn, on='Country')
##take top 15 by rank
top15=merged[merged['Rank']<=15]
##set 'Country' as index
top15=top15.set_index('Country')
#rearrange column names
top15=top15[['Rank', 'Documents', 'Citable documents', 'Citations', 'Self-citations', 'Citations pe
r document', 'H index', 'Energy Supply', 'Energy Supply per Capita', '% Renewable', '2006', '2007',
'2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015']]

top15
```

Out[10]:

	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Energy Supply per Capita	% Renewable	2006
Country											
Australia	14	10616	10496	129788	22759	12.23	123	5.386000e+09	231.0	11.810810	7.4757
Brazil	15	10599	10521	84010	20271	7.93	97	1.214900e+10	59.0	69.648030	1.1076
Canada	8	20689	20353	285554	53955	13.80	165	1.043100e+10	296.0	61.945430	1.3154
China	1	147887	147512	856806	583858	5.79	162	1.271910e+11	93.0	19.754910	2.7521
Germany	7	20898	20640	193676	39615	9.27	140	1.326100e+10	165.0	17.901530	3.0024
Spain	12	11002	10886	167492	31489	15.22	130	4.923000e+09	106.0	37.968590	1.2645
France	9	15584	15387	173959	37411	11.16	129	1.059700e+10	166.0	17.020280	2.3250
United Kingdom	4	24328	23671	278694	52119	11.46	159	7.920000e+09	124.0	10.600470	2.6782
India	5	21450	21183	179494	54929	8.37	132	3.319500e+10	26.0	14.969080	9.2031
Iran	13	10969	10872	94111	31251	8.58	85	9.172000e+09	119.0	5.707721	2.5864
Italy	11	13662	13457	154242	37030	11.29	121	6.530000e+09	109.0	33.667230	1.9426
Japan	3	34294	34054	275980	73491	8.05	145	1.898400e+10	149.0	10.232820	4.5303
South Korea	10	14037	13952	151281	29670	10.78	116	1.100700e+10	221.0	2.279353	1.0118
Russian Federation	6	21259	20915	45629	17368	2.15	65	3.070900e+10	214.0	17.288680	9.8993
United States	2	113579	111426	1085684	370574	9.56	259	9.083800e+10	286.0	11.570980	1.3855

Question 2 (6.6%)

The previous question joined three datasets then reduced this to just the top 15 entries. When you joined the datasets, but before you reduced this to the top 15 items, how many entries did you lose?

This function should return a single number.

NOTE TO SELF: This worked in Jupyter but not the Autograder

In [14]:

```
merged=pd.merge(energy, GDP, how='inner', left_on='Country', right_on='Country')
merged=pd.merge(ScimEn, merged, on='Country')
outer=pd.merge(energy, GDP, how='outer', left_on='Country', right_on='Country')
#should merged be outer?
outer=pd.merge(ScimEn, merged, how='outer', on='Country')
len(outer)-len(merged)
```

Out[14]:

```
31
```

In [15]:

```
get_ipython().run_cell_magic('HTML', '', '<svg width="800" height="300">\n  <circle cx="150" cy="180" r="80" fill-opacity="0.2" stroke="black" stroke-width="2" fill="blue" />\n  <circle cx="200" cy="100" r="80" fill-opacity="0.2" stroke="black" stroke-width="2" fill="red" />\n  <circle cx="100" cy="100" r="80" fill-opacity="0.2" stroke="black" stroke-width="2" fill="green" />\n  <line x1="150" y1="125" x2="300" y2="150" stroke="black" stroke-width="2" fill="black" stroke-dasharray="5,3"/>\n  <text  x="300" y="165" font-family="Verdana" font-size="35">Everything but this!</text>\n</svg>')
```

Question 3 (6.6%)

What are the top 15 countries for average GDP over the last 10 years?

This function should return a Series named `avgGDP` *with 15 countries and their average GDP sorted in descending order.*

In [18]:

```
#average 2006 through 2015 across for each row
top15['avg']=top15.iloc[:, 10:21].mean(axis=1)
sorted_GDP=top15.sort_values(by='avg', ascending=False)
avgGDP=sorted_GDP['avg']
avgGDP
```

Out[18]:

```
Country
United States         1.562296e+13
China                 6.940051e+12
Japan                 5.176323e+12
Germany               3.532694e+12
United Kingdom        2.724781e+12
France                2.688774e+12
Italy                 2.130286e+12
Brazil                1.989027e+12
Russian Federation    1.656011e+12
Canada                1.611540e+12
India                 1.594349e+12
Spain                 1.406411e+12
Australia             1.202763e+12
South Korea           1.165810e+12
Iran                  4.369708e+11
Name: avg, dtype: float64
```

Question 4 (6.6%)

By how much had the GDP changed over the 10 year span for the country with the 6th largest average GDP?

This function should return a single number.

In [19]:

```
top15['avg']=top15.iloc[:, 11:21].mean(axis=1)
sorted_GDP=top15.sort_values(by='avg', ascending=False)
sorted_GDP.iloc[5]['2015']-sorted_GDP.iloc[5]['2006']
```

Out[19]:

```
108550000000.0
```

Question 5 (6.6%)

What is the mean energy supply per capita?

This function should return a single number.

In [20]:

```
top15['Energy Supply per Capita'].mean()
```

Out[20]:

```
157.6
```

Question 6 (6.6%)

What country has the maximum % Renewable and what is the percentage?

This function should return a tuple with the name of the country and the percentage.

In [21]:

```
#get the index value and actual value of row containing max value for '% Renewable'
top15['% Renewable'].idxmax(), top15['% Renewable'].max()
```

Out[21]:

```
('Brazil', 69.64803)
```

Question 7 (6.6%)

Create a new column that is the ratio of Self-Citations to Total Citations. What is the maximum value for this new column, and what country has the highest ratio?

This function should return a tuple with the name of the country and the ratio.

In [22]:

```
top15['% citations self']=top15['Self-citations']/top15['Citations']
#get the index value and actual value of row containing max value for '% citations self'
top15['% citations self'].idxmax(), top15['% citations self'].max()
```

Out[22]:

```
('China', 0.6814354708066936)
```

Question 8 (6.6%)

Create a column that estimates the population using Energy Supply and Energy Supply per capita. What is the third most populous country according to this estimate?

This function should return a single string value.

In [23]:

```
top15['Pop. Est.']=top15['Energy Supply']/top15['Energy Supply per Capita']
sort_pop=top15.sort_values(by='Pop. Est.', ascending=False)
sort_pop.iloc[2].name
```

Out[23]:

```
'United States'
```

Question 9

Create a column that estimates the number of citable documents per person. What is the correlation between the number of citable documents per capita and the energy supply per capita? Use the `.corr()` method, (Pearson's correlation).

This function should return a single number.

In [24]:

```
top15['Pop. Est.']=top15['Energy Supply']/top15['Energy Supply per Capita']
top15['Citable documents per person']=top15['Citable documents']/top15['Pop. Est.']
top15['Citable documents per person'].corr(top15['Energy Supply per Capita'])
```

Out[24]:

```
0.7919971698278406
```

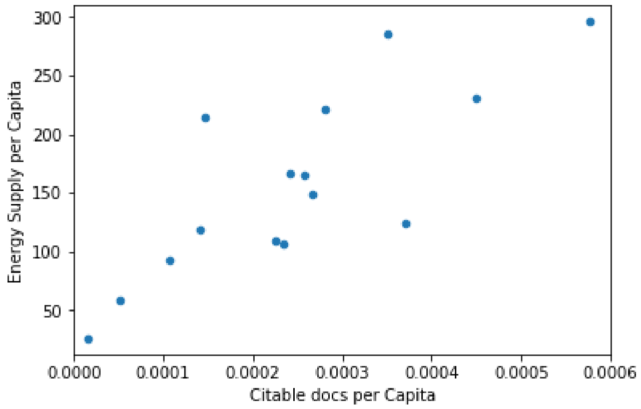
In [26]:

```
import matplotlib as plt
get_ipython().magic('matplotlib inline')


top15['PopEst'] = top15['Energy Supply'] / top15['Energy Supply per Capita']
top15['Citable docs per Capita'] = top15['Citable documents'] / top15['PopEst']
top15.plot(x='Citable docs per Capita', y='Energy Supply per Capita', kind='scatter', xlim=[0,
0.0006])
```

Out[26]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fb4ee6be80>
```



Question 10 (6.6%)

Create a new column with a 1 if the country's % Renewable value is at or above the median for all countries in the top 15, and a 0 if the country's % Renewable value is below the median.

This function should return a series named `HighRenew` whose index is the country name sorted in ascending order of rank.

In [27]:

```
top15.loc[top15['% Renewable'] >= top15['% Renewable'].median(), 'HighRenew'] = 1
top15.loc[top15['% Renewable'] < top15['% Renewable'].median(), 'HighRenew'] = 0
HighRenew=top15['HighRenew']
HighRenew
```

Out[27]:

```
Country
Australia             0.0
Brazil                1.0
Canada                1.0
China                 1.0
Germany               1.0
Spain                 1.0
France                1.0
United Kingdom        0.0
India                 0.0
Iran                  0.0
Italy                 1.0
Japan                 0.0
South Korea           0.0
Russian Federation    1.0
United States         0.0
Name: HighRenew, dtype: float64
```

Question 11 (6.6%)

Use the following dictionary to group the Countries by Continent, then create a dateframe that displays the sample size (the number of countries in each continent bin), and the sum, mean, and std deviation for the estimated population of each country.

```
ContinentDict  = {'China':'Asia',
                  'United States':'North America',
                  'Japan':'Asia',
                  'United Kingdom':'Europe',
                  'Russian Federation':'Europe',
                  'Canada':'North America',
                  'Germany':'Europe',
                  'India':'Asia',
                  'France':'Europe',
                  'South Korea':'Asia',
                  'Italy':'Europe',
                  'Spain':'Europe',
                  'Iran':'Asia',
                  'Australia':'Australia',
                  'Brazil':'South America'}
```

In [28]:

```
top15['Pop. Est.']=top15['Energy Supply']/top15['Energy Supply per Capita']
ContinentDict  = {'China':'Asia', 'United States':'North America', 'Japan':'Asia', 'United Kingdom'
:'Europe', 'Russian Federation':'Europe', 'Canada':'North America',
                      'Germany':'Europe', 'India':'Asia', 'France':'Europe', 'South Korea':'Asia',
Italy':'Europe', 'Spain':'Europe',
                      'Iran':'Asia', 'Australia':'Australia', 'Brazil':'South America'}
#create a dataframe of the 'Pop. Est.', groupby keys in dictionary, count how many in each group
continents=pd.DataFrame(top15['Pop. Est.'].groupby(by=ContinentDict).count())
#rename column to 'size'
continents.rename(columns={'Pop. Est.':'size'}, inplace=True)
#rename index
continents.index.name='Continent'
#get sum by group, etc.
continents['sum']=top15['Pop. Est.'].groupby(by=ContinentDict).sum()
continents['mean']=top15['Pop. Est.'].groupby(by=ContinentDict).mean()
continents['std']=top15['Pop. Est.'].groupby(by=ContinentDict).std()
continents
```

Out[28]:

	size	sum	mean	std
Continent				
Asia	5	2.898666e+09	5.797333e+08	6.790979e+08
Australia	1	2.331602e+07	2.331602e+07	NaN
Europe	6	4.579297e+08	7.632161e+07	3.464767e+07
North America	2	3.528552e+08	1.764276e+08	1.996696e+08
South America	1	2.059153e+08	2.059153e+08	NaN

Question 12 (6.6%)

Cut % Renewable into 5 bins. Group Top15 by the Continent, as well as these new % Renewable bins. How many countries are in each of these groups?

This function should return a Series with a MultiIndex of `Continent`*, then the bins for* `% Renewable`*. Do not include groups with no countries.*

In [29]:

```
ContinentDict  = {'China':'Asia', 'United States':'North America', 'Japan':'Asia', 'United Kingdom'
:'Europe', 'Russian Federation':'Europe', 'Canada':'North America',
                      'Germany':'Europe', 'India':'Asia', 'France':'Europe', 'South Korea':'Asia',
Italy':'Europe', 'Spain':'Europe',
                      'Iran':'Asia', 'Australia':'Australia', 'Brazil':'South America'}
#create empty data frame
continents=pd.DataFrame()
#create column in empty data frame. Map ContinentDict from index values of top15. Need to_series()
because mapping from index not column
continents['Continent']=top15.index.to_series().map(ContinentDict)
#create another column that takes % Renewables from top15, cuts into bin and assigns to correspond
ing index value (Country Name)
```

```
continents['% Renewable Bin']=pd.cut(top15['% Renewable'], 5)
#groupby the two columns. Size returns number of countries/elements per Continent, bin combo. coun
t() doesn't return what you want
continents.groupby(['Continent', '% Renewable Bin']).size()
```

Out[29]:

```
Continent      % Renewable Bin
Asia           (2.212, 15.753]     4
               (15.753, 29.227]    1
Australia      (2.212, 15.753]     1
Europe         (2.212, 15.753]     1
               (15.753, 29.227]    3
               (29.227, 42.701]    2
North America  (2.212, 15.753]     1
               (56.174, 69.648]    1
South America  (56.174, 69.648]    1
dtype: int64
```

Question 13 (6.6%)

Convert the Population Estimate series to a string with thousands separator (using commas)

e.g. 12345678.90 -> 12,345,678.90

This function should return a Series `PopEst` *whose index is the country name and whose values are the population estimate string.*

In [30]:

```
top15['Pop. Est.']=top15['Energy Supply']/top15['Energy Supply per Capita']
#format to a string with thousands separator using comma
PopEst=top15.apply(lambda x: "{:,}".format(x['Pop. Est.']), axis=1)
PopEst
```

Out[30]:

```
Country
Australia                23,316,017.316017315
Brazil                   205,915,254.23728815
Canada                    35,239,864.86486486
China                   1,367,645,161.2903225
Germany                   80,369,696.96969697
Spain                      46,443,396.2264151
France                    63,837,349.39759036
United Kingdom           63,870,967.741935484
India                   1,276,730,769.2307692
Iran                      77,075,630.25210084
Italy                    59,908,256.880733944
Japan                    127,409,395.97315437
South Korea              49,805,429.864253394
Russian Federation              143,500,000.0
United States            317,615,384.61538464
dtype: object
```