Assignment3 (Score: 81.68 / 88.34)

1. Test cell (Score: 5.0 / 5.0)
2. Test cell (Score: 10.0 / 10.0)
3. Test cell (Score: 6.66 / 6.66)
4. Test cell (Score: 6.66 / 6.66)
5. Test cell (Score: 6.66 / 6.66)
6. Test cell (Score: 6.66 / 6.66)
7. Test cell (Score: 6.66 / 6.66)
8. Test cell (Score: 6.66 / 6.66)
9. Test cell (Score: 0.0 / 0.0)
10. Test cell (Score: 6.66 / 6.66)
11. Test cell (Score: 6.66 / 6.66)
12. Test cell (Score: 6.66 / 6.66)
13. Test cell (Score: 0.0 / 6.66)
14. Test cell (Score: 6.74 / 6.74)

Assignment 3¶

All questions are weighted the same in this assignment. This assignment requires more individual learning then the last did - you are encouraged to check out the pandas documentation (http://pandas.pydata.org/pandas-docs/stable/) to fi functions or methods you might not have used yet, or ask questions on Stack Overflow (http://stackoverflow.com/) and them as pandas and python related. All questions are worth the same number of points except question 1 which is wo 17% of the assignment grade.

Note: Questions 2-13 rely on your question 1 answer.

In [1]:

```
import pandas as pd
import numpy as np

# Filter all warnings. If you would like to see the warnings, please comment the two lines b
import warnings
warnings.filterwarnings('ignore')
```

Question 1¶

Load the energy data from the file `assets/Energy Indicators.xls`, which is a list of indicators of energy supply and renewable electricity production (assets/Energy%20Indicators.xls) from the United Nations (http://unstats.un.org/unsd/environment/excel_file_tables/2013/Energy%20Indicators.xls) for the year 2013, and should 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]`

Convert `Energy Supply` to gigajoules (**Note: there are 1,000,000 gigajoules in a petajoule**). For all countries which h 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
"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.
``'Bolivia (Plurinational State of)'` should be `'Bolivia'`. Additionally, there are several
untries with Numeric digits in their name. Make sure to remove these as well, e.g. `'Italy9'
should be `'Italy'`.

Next, load the GDP data from the file `assets/world_bank.csv`, which is a csv containing cou
ries' GDP from 1960 to 2015 from [World Bank](http://data.worldbank.org/indicator/NY.GDP.MKT
CD). 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 Te
nology](http://www.scimagojr.com/countryrank.php?category=2102) from the file `assets/scimag
r-3.xlsx`, which ranks countries based on their journal contributions in the aforementioned
ea. Call this DataFrame **ScimEn**.

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

The index of this DataFrame should be the name of the country, and the columns should be ['R
k', '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, and the rows of the
ataFrame should be sorted by "Rank".*
```
```

In [2]:

Student's answer

```
def answer_one():
    # YOUR CODE HERE
    # raise NotImplementedError()
    Energy = pd.read_excel('assets/Energy Indicators.xls',na_values=["..."],header = None,s
    Energy['Energy Supply'] = Energy['Energy Supply'].apply(lambda x: x*1000000)

    Energy['Country'] = Energy['Country'].str.replace(r" \(.*\)","")
    Energy['Country'] = Energy['Country'].str.replace(r"\d*","")
    Energy['Country'] = Energy['Country'].replace({'Republic of Korea' : 'South Korea',
                                                   'United States of America' : 'United States'
                                                   'United Kingdom of Great Britain and Norther
                                                   'China, Hong Kong Special Administrative Reg

    GDP = pd.read_csv('assets/world_bank.csv', skiprows = 4)
    GDP['Country Name'] = GDP['Country Name'].replace({'Korea, Rep.': 'South Korea',
                                                       'Iran, Islamic Rep.': 'Iran',
                                                       'Hong Kong SAR, China' : 'Hong Kong'

    ScimEn = pd.read_excel('assets/scimagojr-3.xlsx')

    merge1 = pd.merge(ScimEn,Energy,how="inner",left_on="Country",right_on="Country")
    merge1 = merge1[merge1["Rank"]<=15]

    GDP.rename(columns = {"Country Name":"Country"},inplace=True)
    GDP = GDP.loc[:,['2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014'
    merge2 = pd.merge(merge1,GDP,how="inner",left_on="Country",right_on="Country").set_inde

    return merge2
```

In [3]:

Grade cell: `cell-780b5a4da845dbc3` Score: 5.0 / 5.0

```
assert type(answer_one()) == pd.DataFrame, "Q1: You should return a DataFrame!"

assert answer_one().shape == (15,20), "Q1: Your DataFrame should have 20 columns and 15 ent
```

In [4]:

```
answer_one()
```

Out[4]:

Country	Rank	Documents	Citable documents	Citations	Self-citations	Citations per document	H index	Energy Supply	Energy Supply per Capita	% Renewable	
China	1	127050	126767	597237	411683	4.70	138	1.271910e+11	93.0	19.754910	3.9923
United States	2	96661	94747	792274	265436	8.20	230	9.083800e+10	286.0	11.570980	1.4792
Japan	3	30504	30287	223024	61554	7.31	134	1.898400e+10	149.0	10.232820	5.4965
United Kingdom	4	20944	20357	206091	37874	9.84	139	7.920000e+09	124.0	10.600470	2.4196
Russian Federation	5	18534	18301	34266	12422	1.85	57	3.070900e+10	214.0	17.288680	1.3857
Canada	6	17899	17620	215003	40930	12.01	149	1.043100e+10	296.0	61.945430	1.5644
Germany	7	17027	16831	140566	27426	8.26	126	1.326100e+10	165.0	17.901530	3.3328
India	8	15005	14841	128763	37209	8.58	115	3.319500e+10	26.0	14.969080	1.2658
France	9	13153	12973	130632	28601	9.93	114	1.059700e+10	166.0	17.020280	2.6078
South Korea	10	11983	11923	114675	22595	9.57	104	1.100700e+10	221.0	2.279353	9.4101
Italy	11	10964	10794	111850	26661	10.20	106	6.530000e+09	109.0	33.667230	2.2021
Spain	12	9428	9330	123336	23964	13.08	115	4.923000e+09	106.0	37.968590	1.4148
Iran	13	8896	8819	57470	19125	6.46	72	9.172000e+09	119.0	5.707721	3.8955
Australia	14	8831	8725	90765	15606	10.28	107	5.386000e+09	231.0	11.810810	1.0219
Brazil	15	8668	8596	60702	14396	7.00	86	1.214900e+10	59.0	69.648030	1.8450

In [5]:

Grade cell: `cell-74b5f0b971379f64` Score: 10.0 / 10.0

```
# Cell for autograder.
```

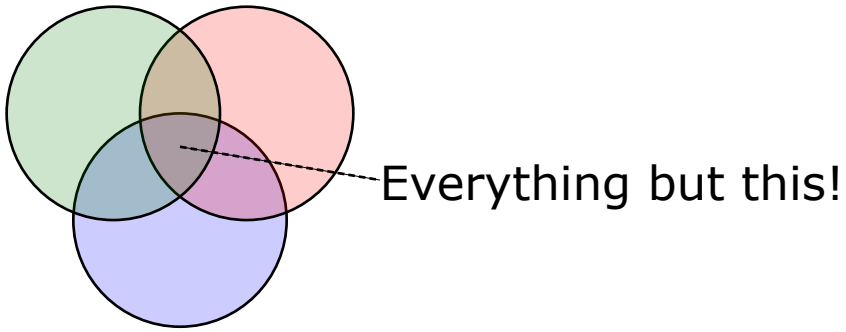
Question 2¶

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

This function should return a single number.

In [6]:

```
%%HTML
<svg width="800" height="300">
  <circle cx="150" cy="180" r="80" fill-opacity="0.2" stroke="black" stroke-width="2" fill="
  <circle cx="200" cy="100" r="80" fill-opacity="0.2" stroke="black" stroke-width="2" fill="
  <circle cx="100" cy="100" r="80" fill-opacity="0.2" stroke="black" stroke-width="2" fill="
  <line x1="150" y1="125" x2="300" y2="150" stroke="black" stroke-width="2" fill="black" str
  <text x="300" y="165" font-family="Verdana" font-size="35">Everything but this!</text>
</svg>
```



In [7]:

Student's answer

```
def answer_two():
    # YOUR CODE HERE
    # raise NotImplementedError()
    Energy = pd.read_excel('assets/Energy Indicators.xls',na_values=["..."],header = None,s
    Energy['Energy Supply'] = Energy['Energy Supply'].apply(lambda x: x*1000000)

    Energy['Country'] = Energy['Country'].str.replace(r" \(.*\)","")
    Energy['Country'] = Energy['Country'].str.replace(r"\d*","")
    Energy['Country'] = Energy['Country'].replace({'Republic of Korea' : 'South Korea',
                                                   'United States of America' : 'United States'
                                                   'United Kingdom of Great Britain and Norther
                                                   'China, Hong Kong Special Administrative Reg

    GDP = pd.read_csv('assets/world_bank.csv', skiprows = 4)
    GDP['Country Name'] = GDP['Country Name'].replace({'Korea, Rep.': 'South Korea',
                                                       'Iran, Islamic Rep.': 'Iran',
                                                       'Hong Kong SAR, China' : 'Hong Kong'

    ScimEn = pd.read_excel('assets/scimagojr-3.xlsx')

    inner1 = pd.merge(ScimEn,Energy,how="inner",left_on="Country",right_on="Country")

    GDP.rename(columns = {"Country Name":"Country"},inplace=True)
    GDP = GDP.loc[:,['2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014'
    inner2 = pd.merge(inner1,GDP,how="inner",left_on="Country",right_on="Country").set_inde

    outer1 = pd.merge(ScimEn,Energy,how="outer",left_on="Country",right_on="Country")
    outer2 = pd.merge(outer1,GDP,how="outer",left_on="Country",right_on="Country").set_inde

    return len(outer2)-len(inner2)
```

In [8]:

Grade cell: `cell-be24cfcaa87ab071` Score: 6.66 / 6.66

```
assert type(answer_two()) == int, "Q2: You should return an int number!"
```

In [9]:

```
answer_two()
```

Out[9]:

```
158
```

Question 3¶

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 orde

In [10]:

Student's answer

```
def answer_three():
    # YOUR CODE HERE
    # raise NotImplementedError()
    info=answer_one()
    return info[["2006","2007","2008","2009","2010","2011","2012","2013","2014","2015"]].ap
```

In [11]:

Grade cell: `cell-aaaa11ef7d26f4cf` Score: 6.66 / 6.66

```
assert type(answer_three()) == pd.Series, "Q3: You should return a Series!"
```

In [12]:

```
answer_three()
```

Out[12]:

```
Country
United States         1.536434e+13
China                 6.348609e+12
Japan                 5.542208e+12
Germany               3.493025e+12
France                2.681725e+12
United Kingdom        2.487907e+12
Brazil                2.189794e+12
Italy                 2.120175e+12
India                 1.769297e+12
Canada                1.660647e+12
Russian Federation    1.565459e+12
Spain                 1.418078e+12
Australia             1.164043e+12
South Korea           1.106715e+12
Iran                  4.441558e+11
dtype: float64
```

Question 4¶

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 [13]:

Student's answer

```
def answer_four():
    # YOUR CODE HERE
    # raise NotImplementedError()
    info=answer_one()
    info['avgGDP']=info[["2006","2007","2008","2009","2010","2011","2012","2013","2014","20
    info.sort_values(['avgGDP'],ascending = False,inplace=True)

    # g6=info.index[5]
    # info.loc[g6]["2015"]-info.loc[g6]["2006"]

    return info.iloc[5]['2015']-info.iloc[5]['2006']
```

In [14]:

Grade cell: `cell-564dd8e5e24b0f83` Score: 6.66 / 6.66

```
# Cell for autograder.
```

In [15]:

```
answer_four()
```

Out[15]:

```
246702696075.3999
```

Question 5¶

What is the mean energy supply per capita?

This function should return a single number.

In [16]:

Student's answer

```
info = answer_one()
# info
type(info['Energy Supply per Capita'].mean())
float(info['Energy Supply per Capita'].mean())
def answer_five():
    # YOUR CODE HERE
    # raise NotImplementedError()
    info = answer_one()
    return info['Energy Supply per Capita'].mean()
#     return float(info['Energy Supply per Capita'].mean())
answer_five()
```

Out[16]:

```
157.6
```

In [17]:

Grade cell: `cell-30cc66180851638c` Score: 6.66 / 6.66

```
# Cell for autograder.
```

Question 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 [18]:

Student's answer

```
info = answer_one()
# info['% Renewable']
result=info.sort_values(by='% Renewable', ascending=False).iloc[0]
(result.name,result['% Renewable'])
def answer_six():
    # YOUR CODE HERE
    # raise NotImplementedError()
    info = answer_one()
    result=info.sort_values(by='% Renewable', ascending=False).iloc[0]
    return (result.name,result['% Renewable'])
```

In [19]:

Grade cell: `cell-2bd201c5c7bdd80f` Score: 6.66 / 6.66

```
assert type(answer_six()) == tuple, "Q6: You should return a tuple!"

assert type(answer_six()[0]) == str, "Q6: The first element in your result should be the na
```

In [20]:

```
answer_six()
```

Out[20]:

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

Question 7¶

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

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

In [21]:

Student's answer

```
def answer_seven():
    # YOUR CODE HERE
    # raise NotImplementedError()
    info = answer_one()
    info['Citation ratio']=info['Self-citations']/info['Citations']
    result=info.sort_values(by='Citation ratio', ascending=False).iloc[0]
    return (result.name,result['Citation ratio'])
```

In [22]:

Grade cell: `cell-b7a163e9231b88c9` Score: 6.66 / 6.66

```
assert type(answer_seven()) == tuple, "Q7: You should return a tuple!"

assert type(answer_seven()[0]) == str, "Q7: The first element in your result should be the
```

In [23]:

```
answer_seven()
```

Out[23]:

```
('China', 0.6893126179389422)
```

Question 8¶

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

This function should return the name of the country

In [24]:

Student's answer

```
def answer_eight():
    # YOUR CODE HERE
    # raise NotImplementedError()
    info = answer_one()
    return (info['Energy Supply']/info['Energy Supply per Capita']).sort_values(ascending=F
```

In [25]:

Grade cell: `cell-3f3620c88df08b20` Score: 0.0 / 0.0

```
assert type(answer_eight()) == str, "Q8: You should return the name of the country!"
```

In [26]:

```
answer_eight()
```

Out[26]:

```
'United States'
```

Question 9¶

Create a column that estimates the number of citable documents per person. What is the correlation between the num 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.

(Optional: Use the built-in function `plot9()` to visualize the relationship between Energy Supply per Capita vs. Citable per Capita)

In [27]:

Student's answer

```
def answer_nine():
    # YOUR CODE HERE
    # raise NotImplementedError()
    Top15 = answer_one()
    Top15['PopEst'] = Top15['Energy Supply'] / Top15['Energy Supply per Capita']
    Top15['Citable docs per Capita'] = Top15['Citable documents'] / Top15['PopEst']
    return Top15['Citable docs per Capita'].corr(Top15['Energy Supply per Capita'])
```

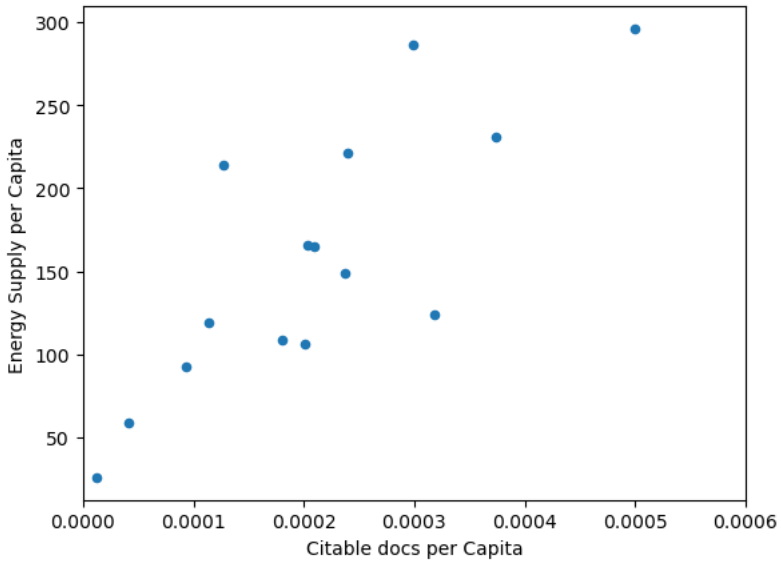
In [28]:

```
def plot9():
    import matplotlib as plt
    %matplotlib inline

    Top15 = answer_one()
    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', xl
```

In [29]:

```
plot9()
```



In [30]:

Grade cell: `cell-3cb5c699065a4a20` Score: 6.66 / 6.66

```
assert answer_nine() >= -1. and answer_nine() <= 1., "Q9: A valid correlation should betwee
```

In [31]:

```
answer_nine()
```

Out[31]:

```
0.7940010435442946
```

Question 10¶

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 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 r

In [32]:

Student's answer

```
def answer_ten():
    # YOUR CODE HERE
    # raise NotImplementedError()
    Top15 = answer_one()
    Rmedian=Top15["% Renewable"].median()
    Top15["HighRenew"]= Top15["% Renewable"].apply(lambda x:0 if x<Rmedian else 1 )
    return Top15["HighRenew"]
```

In [33]:

Grade cell: `cell-b29a631fd9a7730f` Score: 6.66 / 6.66

```
assert type(answer_ten()) == pd.Series, "Q10: You should return a Series!"
```

In [34]:

```
answer_ten()
```

Out[34]:

```
Country
China                1
United States        0
Japan                0
United Kingdom       0
Russian Federation   1
Canada               1
Germany              1
India                0
France               1
South Korea          0
Italy                1
Spain                1
Iran                 0
Australia            0
Brazil               1
Name: HighRenew, dtype: int64
```

Question 11¶

Use the following dictionary to group the Countries by Continent, then create a DataFrame that displays the sample siz
(the number of countries in each continent bin), and the sum, mean, and std deviation for the estimated population of e
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'}
```

This function should return a DataFrame with index named Continent `['Asia', 'Australia', 'Europe', 'North America', 'South America']` and columns `['size', 'sum', 'mean', 'std']`

In [35]:

Student's answer

```
def answer_eleven():
    # YOUR CODE HERE
    # raise NotImplementedError()
    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'}

    Top15 = answer_one()
    Top15['PopEst'] = Top15['Energy Supply'] / Top15['Energy Supply per Capita']
    Top15['Continent'] = pd.Series(ContinentDict)

    return Top15.groupby('Continent')['PopEst'].agg([np.size,np.sum, np.mean, np.std])
```

In [36]:

Grade cell: `cell-18d1a07971b25743` Score: 6.66 / 6.66

```
assert type(answer_eleven()) == pd.DataFrame, "Q11: You should return a DataFrame!"

assert answer_eleven().shape[0] == 5, "Q11: Wrong row numbers!"

assert answer_eleven().shape[1] == 4, "Q11: Wrong column numbers!"
```

In [37]:

```
answer_eleven()
```

Out[37]:

	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¶

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 gr with no countries.

In [38]:

```
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'}

Top15 = answer_one()
Top15['Continent'] = pd.Series(ContinentDict)
Top15['% Renewable']=pd.cut(Top15['% Renewable'],5)

Top15.groupby(['Continent','% Renewable']).size()
# Top15.groupby(['Continent','% Renewable'])['Continent'].agg(np.size).dropna()
```

Out[38]:

```
Continent      % Renewable
Asia           (2.212, 15.753]     4
               (15.753, 29.227]    1
               (29.227, 42.701]    0
               (42.701, 56.174]    0
               (56.174, 69.648]    0
Australia      (2.212, 15.753]     1
               (15.753, 29.227]    0
               (29.227, 42.701]    0
               (42.701, 56.174]    0
               (56.174, 69.648]    0
Europe         (2.212, 15.753]     1
               (15.753, 29.227]    3
               (29.227, 42.701]    2
               (42.701, 56.174]    0
               (56.174, 69.648]    0
North America  (2.212, 15.753]     1
               (15.753, 29.227]    0
               (29.227, 42.701]    0
               (42.701, 56.174]    0
               (56.174, 69.648]    1
South America  (2.212, 15.753]     0
               (15.753, 29.227]    0
               (29.227, 42.701]    0
               (42.701, 56.174]    0
               (56.174, 69.648]    1
dtype: int64
```

In [39]:

Student's answer

```
def answer_twelve():
    # YOUR CODE HERE
    # raise NotImplementedError()
    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'}

    Top15 = answer_one()
    Top15['Continent'] = pd.Series(ContinentDict)
    Top15['% Renewable']=pd.cut(Top15['% Renewable'],5)

    return Top15.groupby(['Continent','% Renewable'])['Continent'].agg(np.size).dropna()
```

In [40]:

```
assert type(answer_twelve()) == pd.Series, "Q12: You should return a Series!"

assert len(answer_twelve()) == 9, "Q12: Wrong result numbers!"
```

In [41]:

Grade cell: `cell-6c665602d6babab9` Score: 0.0 / 6.66

```
assert type(answer_twelve()) == pd.Series, "Q12: You should return a Series!"

assert len(answer_twelve()) == 25, "Q12: Wrong result numbers!"
```

```
You have failed this test due to an error. The traceback has been removed because it may cor

AssertionError: Q12: Wrong result numbers!
```

In [42]:

```
assert type(answer_twelve()) == pd.Series, "Q12: You should return a Series!"

assert len(answer_twelve()) == 9, "Q12: Wrong result numbers!"
```

In [43]:

```
answer_twelve()
```

Out[43]:

```
Continent      % Renewable
Asia           (2.212, 15.753]     4.0
               (15.753, 29.227]    1.0
Australia      (2.212, 15.753]     1.0
Europe         (2.212, 15.753]     1.0
               (15.753, 29.227]    3.0
               (29.227, 42.701]    2.0
North America  (2.212, 15.753]     1.0
               (56.174, 69.648]    1.0
South America  (56.174, 69.648]    1.0
Name: Continent, dtype: float64
```

Question 13¶

Convert the Population Estimate series to a string with thousands separator (using commas). Use all significant digits (
not round the results).

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 esti
string*

In [44]:

Student's answer

```
def answer_thirteen():
    # YOUR CODE HERE
    # raise NotImplementedError()
    Top15 = answer_one()
    Top15['PopEst'] = Top15['Energy Supply'] / Top15['Energy Supply per Capita']
    return Top15['PopEst'].apply('{:,}'.format)
```

In [45]:

Grade cell: `cell-10fee7228cf973f6` Score: 6.74 / 6.74

```
assert type(answer_thirteen()) == pd.Series, "Q13: You should return a Series!"

assert len(answer_thirteen()) == 15, "Q13: Wrong result numbers!"
```

In [46]:

```
answer_thirteen()
```

Out[46]:

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

Optional¶

Use the built in function `plot_optional()` to see an example visualization.

In [47]:

```
def plot_optional():
    import matplotlib as plt
    %matplotlib inline
    Top15 = answer_one()
    ax = Top15.plot(x='Rank', y='% Renewable', kind='scatter',
                    c=['#e41a1c','#377eb8','#e41a1c','#4daf4a','#4daf4a','#377eb8','#4daf4a'
                       '#4daf4a','#e41a1c','#4daf4a','#4daf4a','#e41a1c','#dede00','#ff7f00'
                    xticks=range(1,16), s=6*Top15['2014']/10**10, alpha=.75, figsize=[16,6])

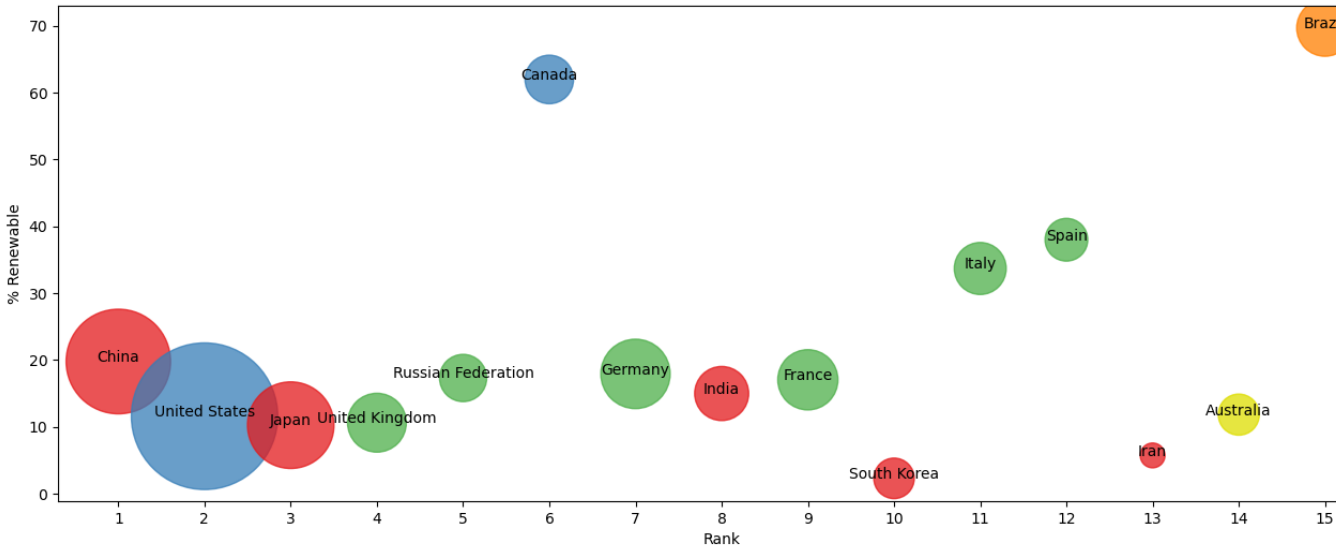
    for i, txt in enumerate(Top15.index):
        ax.annotate(txt, [Top15['Rank'][i], Top15['% Renewable'][i]], ha='center')

    print("This is an example of a visualization that can be created to help understand the
This is a bubble chart showing % Renewable vs. Rank. The size of the bubble corresponds to t
2014 GDP, and the color corresponds to the continent.")
```

In [48]:

```
plot_optional()
```

```
This is an example of a visualization that can be created to help understand the data. This
```



In [ ]:

This assignment was graded by mooc_adswpy:e5e20d3b91dd, v1.46.070623