Assignment 2

April 22, 2019

---

*You are currently looking at **version 1.2** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the Jupyter Notebook FAQ course resource.*

---

1 Assignment 2 - Pandas Introduction

All questions are weighted the same in this assignment. ## Part 1 The following code loads the olympics dataset (olympics.csv), which was derrived from the Wikipedia entry on All Time Olympic Games Medals, and does some basic data cleaning.

The columns are organized as # of Summer games, Summer medals, # of Winter games, Winter medals, total # number of games, total # of medals. Use this dataset to answer the questions below.

In [1]:
```
import pandas as pd

df = pd.read_csv('olympics.csv', index_col=0, skiprows=1)

for col in df.columns:
    if col[:2]=='01':
        df.rename(columns={col:'Gold'+col[4:]}, inplace=True)
    if col[:2]=='02':
        df.rename(columns={col:'Silver'+col[4:]}, inplace=True)
    if col[:2]=='03':
        df.rename(columns={col:'Bronze'+col[4:]}, inplace=True)
    if col[:1]=='':
        df.rename(columns={col:'#'+col[1:]}, inplace=True)

names_ids = df.index.str.split('\s\(') # split the index by '('

df.index = names_ids.str[0] # the [0] element is the country name (new index)
df['ID'] = names_ids.str[1].str[:3] # the [1] element is the abbreviation or ID (take fi

df = df.drop('Totals')
df
```

Out[1]:

	# Summer	Gold	Silver	Bronze	Total	\
Afghanistan	13	0	0	2	2	
Algeria	12	5	2	8	15	
Argentina	23	18	24	28	70	
Armenia	5	1	2	9	12	
Australasia	2	3	4	5	12	
Australia	25	139	152	177	468	
Austria	26	18	33	35	86	
Azerbaijan	5	6	5	15	26	
Bahamas	15	5	2	5	12	
Bahrain	8	0	0	1	1	
Barbados	11	0	0	1	1	
Belarus	5	12	24	39	75	
Belgium	25	37	52	53	142	
Bermuda	17	0	0	1	1	
Bohemia	3	0	1	3	4	
Botswana	9	0	1	0	1	
Brazil	21	23	30	55	108	
British West Indies	1	0	0	2	2	
Bulgaria	19	51	85	78	214	
Burundi	5	1	0	0	1	
Cameroon	13	3	1	1	5	
Canada	25	59	99	121	279	
Chile	22	2	7	4	13	
China	9	201	146	126	473	
Colombia	18	2	6	11	19	
Costa Rica	14	1	1	2	4	
Ivory Coast	12	0	1	0	1	
Croatia	6	6	7	10	23	
Cuba	19	72	67	70	209	
Cyprus	9	0	1	0	1	
...	...	...	...	...	...	
Spain	22	37	59	35	131	
Sri Lanka	16	0	2	0	2	
Sudan	11	0	1	0	1	
Suriname	11	1	0	1	2	
Sweden	26	143	164	176	483	
Switzerland	27	47	73	65	185	
Syria	12	1	1	1	3	
Chinese Taipei	13	2	7	12	21	
Tajikistan	5	0	1	2	3	
Tanzania	12	0	2	0	2	
Thailand	15	7	6	11	24	
Togo	9	0	0	1	1	
Tonga	8	0	1	0	1	
Trinidad and Tobago	16	2	5	11	18	
Tunisia	13	3	3	4	10	
Turkey	21	39	25	24	88	

Uganda	14	2	3	2	7
Ukraine	5	33	27	55	115
United Arab Emirates	8	1	0	0	1
United States	26	976	757	666	2399
Uruguay	20	2	2	6	10
Uzbekistan	5	5	5	10	20
Venezuela	17	2	2	8	12
Vietnam	14	0	2	0	2
Virgin Islands	11	0	1	0	1
Yugoslavia	16	26	29	28	83
Independent Olympic Participants	1	0	1	2	3
Zambia	12	0	1	1	2
Zimbabwe	12	3	4	1	8
Mixed team	3	8	5	4	17

	# Winter	Gold.1	Silver.1	Bronze.1	\
Afghanistan	0	0	0	0	
Algeria	3	0	0	0	
Argentina	18	0	0	0	
Armenia	6	0	0	0	
Australasia	0	0	0	0	
Australia	18	5	3	4	
Austria	22	59	78	81	
Azerbaijan	5	0	0	0	
Bahamas	0	0	0	0	
Bahrain	0	0	0	0	
Barbados	0	0	0	0	
Belarus	6	6	4	5	
Belgium	20	1	1	3	
Bermuda	7	0	0	0	
Bohemia	0	0	0	0	
Botswana	0	0	0	0	
Brazil	7	0	0	0	
British West Indies	0	0	0	0	
Bulgaria	19	1	2	3	
Burundi	0	0	0	0	
Cameroon	1	0	0	0	
Canada	22	62	56	52	
Chile	16	0	0	0	
China	10	12	22	19	
Colombia	1	0	0	0	
Costa Rica	6	0	0	0	
Ivory Coast	0	0	0	0	
Croatia	7	4	6	1	
Cuba	0	0	0	0	
Cyprus	10	0	0	0	
...	...	...	...	...	
Spain	19	1	0	1	

Sri Lanka	0	0	0	0
Sudan	0	0	0	0
Suriname	0	0	0	0
Sweden	22	50	40	54
Switzerland	22	50	40	48
Syria	0	0	0	0
Chinese Taipei	11	0	0	0
Tajikistan	4	0	0	0
Tanzania	0	0	0	0
Thailand	3	0	0	0
Togo	1	0	0	0
Tonga	1	0	0	0
Trinidad and Tobago	3	0	0	0
Tunisia	0	0	0	0
Turkey	16	0	0	0
Uganda	0	0	0	0
Ukraine	6	2	1	4
United Arab Emirates	0	0	0	0
United States	22	96	102	84
Uruguay	1	0	0	0
Uzbekistan	6	1	0	0
Venezuela	4	0	0	0
Vietnam	0	0	0	0
Virgin Islands	7	0	0	0
Yugoslavia	14	0	3	1
Independent Olympic Participants	0	0	0	0
Zambia	0	0	0	0
Zimbabwe	1	0	0	0
Mixed team	0	0	0	0

	Total.1	# Games	Gold.2	Silver.2	\
Afghanistan	0	13	0	0	
Algeria	0	15	5	2	
Argentina	0	41	18	24	
Armenia	0	11	1	2	
Australasia	0	2	3	4	
Australia	12	43	144	155	
Austria	218	48	77	111	
Azerbaijan	0	10	6	5	
Bahamas	0	15	5	2	
Bahrain	0	8	0	0	
Barbados	0	11	0	0	
Belarus	15	11	18	28	
Belgium	5	45	38	53	
Bermuda	0	24	0	0	
Bohemia	0	3	0	1	
Botswana	0	9	0	1	
Brazil	0	28	23	30	

British West Indies	0	1	0	0
Bulgaria	6	38	52	87
Burundi	0	5	1	0
Cameroon	0	14	3	1
Canada	170	47	121	155
Chile	0	38	2	7
China	53	19	213	168
Colombia	0	19	2	6
Costa Rica	0	20	1	1
Ivory Coast	0	12	0	1
Croatia	11	13	10	13
Cuba	0	19	72	67
Cyprus	0	19	0	1
...	...	...	...	...
Spain	2	41	38	59
Sri Lanka	0	16	0	2
Sudan	0	11	0	1
Suriname	0	11	1	0
Sweden	144	48	193	204
Switzerland	138	49	97	113
Syria	0	12	1	1
Chinese Taipei	0	24	2	7
Tajikistan	0	9	0	1
Tanzania	0	12	0	2
Thailand	0	18	7	6
Togo	0	10	0	0
Tonga	0	9	0	1
Trinidad and Tobago	0	19	2	5
Tunisia	0	13	3	3
Turkey	0	37	39	25
Uganda	0	14	2	3
Ukraine	7	11	35	28
United Arab Emirates	0	8	1	0
United States	282	48	1072	859
Uruguay	0	21	2	2
Uzbekistan	1	11	6	5
Venezuela	0	21	2	2
Vietnam	0	14	0	2
Virgin Islands	0	18	0	1
Yugoslavia	4	30	26	32
Independent Olympic Participants	0	1	0	1
Zambia	0	12	0	1
Zimbabwe	0	13	3	4
Mixed team	0	3	8	5

	Bronze.2	Combined total	ID
Afghanistan	2	2	AFG
Algeria	8	15	ALG

Argentina	28	70	ARG
Armenia	9	12	ARM
Australasia	5	12	ANZ
Australia	181	480	AUS
Austria	116	304	AUT
Azerbaijan	15	26	AZE
Bahamas	5	12	BAH
Bahrain	1	1	BRN
Barbados	1	1	BAR
Belarus	44	90	BLR
Belgium	56	147	BEL
Bermuda	1	1	BER
Bohemia	3	4	BOH
Botswana	0	1	BOT
Brazil	55	108	BRA
British West Indies	2	2	BWI
Bulgaria	81	220	BUL
Burundi	0	1	BDI
Cameroon	1	5	CMR
Canada	173	449	CAN
Chile	4	13	CHI
China	145	526	CHN
Colombia	11	19	COL
Costa Rica	2	4	CRC
Ivory Coast	0	1	CIV
Croatia	11	34	CRO
Cuba	70	209	CUB
Cyprus	0	1	CYP
...	...	...	...
Spain	36	133	ESP
Sri Lanka	0	2	SRI
Sudan	0	1	SUD
Suriname	1	2	SUR
Sweden	230	627	SWE
Switzerland	113	323	SUI
Syria	1	3	SYR
Chinese Taipei	12	21	TPE
Tajikistan	2	3	TJK
Tanzania	0	2	TAN
Thailand	11	24	THA
Togo	1	1	TOG
Tonga	0	1	TGA
Trinidad and Tobago	11	18	TRI
Tunisia	4	10	TUN
Turkey	24	88	TUR
Uganda	2	7	UGA
Ukraine	59	122	UKR
United Arab Emirates	0	1	UAE

```
United States                          750            2681  USA
Uruguay                                  6              10  URU
Uzbekistan                              10              21  UZB
Venezuela                                8              12  VEN
Vietnam                                  0               2  VIE
Virgin Islands                           0               1  ISV
Yugoslavia                              29              87  YUG
Independent Olympic Participants         2               3  IOP
Zambia                                   1               2  ZAM
Zimbabwe                                 1               8  ZIM
Mixed team                               4              17  ZZX

[146 rows x 16 columns]
```

1.0.1 Question 0 (Example)

What is the first country in df?

This function should return a Series.

```
In [2]: # You should write your whole answer within the function provided. The autograder will c
        # this function and compare the return value against the correct solution value
        def answer_zero():
            # This function returns the row for Afghanistan, which is a Series object. The assig
            # question description will tell you the general format the autograder is expecting
            return df.iloc[0]

        # You can examine what your function returns by calling it in the cell. If you have ques
        # about the assignment formats, check out the discussion forums for any FAQs
        answer_zero()
```

```
Out[2]: # Summer           13
        Gold                0
        Silver              0
        Bronze              2
        Total               2
        # Winter           0
        Gold.1              0
        Silver.1            0
        Bronze.1            0
        Total.1             0
        # Games            13
        Gold.2              0
        Silver.2            0
        Bronze.2            2
        Combined total      2
        ID                AFG
        Name: Afghanistan, dtype: object
```

1.0.2 Question 1

Which country has won the most gold medals in summer games?
This function should return a single string value.

```
In [3]: def answer_one():
            return df['Gold'].idxmax()

        answer_one()
```

```
Out[3]: 'United States'
```

1.0.3 Question 2

Which country had the biggest difference between their summer and winter gold medal counts?
This function should return a single string value.

```
In [4]: def answer_two():
            df['Gold_diff'] = df['Gold'] - df['Gold.1']
            df['Gold_diff'] = df['Gold_diff'].abs()
            return df['Gold_diff'].idxmax()

        answer_two()
```

```
Out[4]: 'United States'
```

1.0.4 Question 3

Which country has the biggest difference between their summer gold medal counts and winter gold medal counts relative to their total gold medal count?

$$\frac{\textit{Summer Gold} - \textit{Winter Gold}}{\textit{Total Gold}}$$

Only include countries that have won at least 1 gold in both summer and winter.
This function should return a single string value.

```
In [5]: def answer_three():

            df['SummerGold'] = df['Gold'] > 0
            df['WinterGold'] = df['Gold.1'] > 0
            df['TotalGold'] = df['Gold.2'] > 0
            df['AvgGold'] = 0

            for idx, row in df.iterrows():
                if(df.loc[idx, 'SummerGold'] and df.loc[idx, 'WinterGold'] and df.loc[idx, 'Tota
                    df.loc[idx, 'AvgGold'] = df.loc[idx, 'Gold_diff'] / df.loc[idx, 'Gold.2']

            return df['AvgGold'].idxmax()

        answer_three()
```

```
Out[5]: 'Bulgaria'
```

1.0.5 Question 4

Write a function that creates a Series called "Points" which is a weighted value where each gold medal (`Gold.2`) counts for 3 points, silver medals (`Silver.2`) for 2 points, and bronze medals (`Bronze.2`) for 1 point. The function should return only the column (a Series object) which you created, with the country names as indices.

This function should return a Series named `Points` of length 146

```
In [6]: def answer_four():

            df['Points'] = (df['Gold.2']*3) + (df['Silver.2']*2) + df['Bronze.2']
            return df['Points']

        answer_four()
```

```
Out[6]: Afghanistan                   2
        Algeria                      27
        Argentina                   130
        Armenia                      16
        Australasia                  22
        Australia                   923
        Austria                     569
        Azerbaijan                   43
        Bahamas                      24
        Bahrain                       1
        Barbados                      1
        Belarus                     154
        Belgium                     276
        Bermuda                       1
        Bohemia                       5
        Botswana                      2
        Brazil                      184
        British West Indies           2
        Bulgaria                    411
        Burundi                       3
        Cameroon                     12
        Canada                      846
        Chile                        24
        China                      1120
        Colombia                     29
        Costa Rica                    7
        Ivory Coast                   2
        Croatia                      67
        Cuba                        420
        Cyprus                        2
                                    ...
        Spain                       268
        Sri Lanka                     4
        Sudan                         2
```

```
Suriname                              4
Sweden                             1217
Switzerland                         630
Syria                                 6
Chinese Taipei                       32
Tajikistan                            4
Tanzania                              4
Thailand                             44
Togo                                  1
Tonga                                 2
Trinidad and Tobago                  27
Tunisia                              19
Turkey                              191
Uganda                               14
Ukraine                             220
United Arab Emirates                  3
United States                      5684
Uruguay                              16
Uzbekistan                           38
Venezuela                            18
Vietnam                               4
Virgin Islands                        2
Yugoslavia                          171
Independent Olympic Participants      4
Zambia                                3
Zimbabwe                             18
Mixed team                           38
Name: Points, dtype: int64
```

1.1 Part 2

For the next set of questions, we will be using census data from the United States Census Bureau. Counties are political and geographic subdivisions of states in the United States. This dataset contains population data for counties and states in the US from 2010 to 2015. See this document for a description of the variable names.

The census dataset (census.csv) should be loaded as census_df. Answer questions using this as appropriate.

1.1.1 Question 5

Which state has the most counties in it? (hint: consider the sumlevel key carefully! You'll need this for future questions too...)

This function should return a single string value.

```
In [7]: census_df = pd.read_csv('census.csv')
        census_df
```

```
Out[7]:       SUMLEV  REGION  DIVISION  STATE  COUNTY     STNAME            CTYNAME  \
        0         40       3         6      1       0    Alabama            Alabama
```

1	50	3	6	1	1	Alabama	Autauga County
2	50	3	6	1	3	Alabama	Baldwin County
3	50	3	6	1	5	Alabama	Barbour County
4	50	3	6	1	7	Alabama	Bibb County
5	50	3	6	1	9	Alabama	Blount County
6	50	3	6	1	11	Alabama	Bullock County
7	50	3	6	1	13	Alabama	Butler County
8	50	3	6	1	15	Alabama	Calhoun County
9	50	3	6	1	17	Alabama	Chambers County
10	50	3	6	1	19	Alabama	Cherokee County
11	50	3	6	1	21	Alabama	Chilton County
12	50	3	6	1	23	Alabama	Choctaw County
13	50	3	6	1	25	Alabama	Clarke County
14	50	3	6	1	27	Alabama	Clay County
15	50	3	6	1	29	Alabama	Cleburne County
16	50	3	6	1	31	Alabama	Coffee County
17	50	3	6	1	33	Alabama	Colbert County
18	50	3	6	1	35	Alabama	Conecuh County
19	50	3	6	1	37	Alabama	Coosa County
20	50	3	6	1	39	Alabama	Covington County
21	50	3	6	1	41	Alabama	Crenshaw County
22	50	3	6	1	43	Alabama	Cullman County
23	50	3	6	1	45	Alabama	Dale County
24	50	3	6	1	47	Alabama	Dallas County
25	50	3	6	1	49	Alabama	DeKalb County
26	50	3	6	1	51	Alabama	Elmore County
27	50	3	6	1	53	Alabama	Escambia County
28	50	3	6	1	55	Alabama	Etowah County
29	50	3	6	1	57	Alabama	Fayette County
...	...	...	...	...	...	...	...
3163	50	2	3	55	131	Wisconsin	Washington County
3164	50	2	3	55	133	Wisconsin	Waukesha County
3165	50	2	3	55	135	Wisconsin	Waupaca County
3166	50	2	3	55	137	Wisconsin	Waushara County
3167	50	2	3	55	139	Wisconsin	Winnebago County
3168	50	2	3	55	141	Wisconsin	Wood County
3169	40	4	8	56	0	Wyoming	Wyoming
3170	50	4	8	56	1	Wyoming	Albany County
3171	50	4	8	56	3	Wyoming	Big Horn County
3172	50	4	8	56	5	Wyoming	Campbell County
3173	50	4	8	56	7	Wyoming	Carbon County
3174	50	4	8	56	9	Wyoming	Converse County
3175	50	4	8	56	11	Wyoming	Crook County
3176	50	4	8	56	13	Wyoming	Fremont County
3177	50	4	8	56	15	Wyoming	Goshen County
3178	50	4	8	56	17	Wyoming	Hot Springs County
3179	50	4	8	56	19	Wyoming	Johnson County
3180	50	4	8	56	21	Wyoming	Laramie County

```
3181      50       4         8     56      23    Wyoming      Lincoln County
3182      50       4         8     56      25    Wyoming      Natrona County
3183      50       4         8     56      27    Wyoming     Niobrara County
3184      50       4         8     56      29    Wyoming          Park County
3185      50       4         8     56      31    Wyoming        Platte County
3186      50       4         8     56      33    Wyoming     Sheridan County
3187      50       4         8     56      35    Wyoming     Sublette County
3188      50       4         8     56      37    Wyoming   Sweetwater County
3189      50       4         8     56      39    Wyoming         Teton County
3190      50       4         8     56      41    Wyoming         Uinta County
3191      50       4         8     56      43    Wyoming     Washakie County
3192      50       4         8     56      45    Wyoming        Weston County

      CENSUS2010POP  ESTIMATESBASE2010  POPESTIMATE2010     ...       \
0           4779736            4780127          4785161     ...
1             54571              54571            54660     ...
2            182265             182265           183193     ...
3             27457              27457            27341     ...
4             22915              22919            22861     ...
5             57322              57322            57373     ...
6             10914              10915            10887     ...
7             20947              20946            20944     ...
8            118572             118586           118437     ...
9             34215              34170            34098     ...
10            25989              25986            25976     ...
11            43643              43631            43665     ...
12            13859              13858            13841     ...
13            25833              25840            25767     ...
14            13932              13932            13880     ...
15            14972              14972            14973     ...
16            49948              49948            50177     ...
17            54428              54428            54514     ...
18            13228              13228            13208     ...
19            11539              11758            11758     ...
20            37765              37765            37796     ...
21            13906              13906            13853     ...
22            80406              80410            80473     ...
23            50251              50251            50358     ...
24            43820              43820            43803     ...
25            71109              71115            71142     ...
26            79303              79296            79465     ...
27            38319              38319            38309     ...
28           104430             104427           104442     ...
29            17241              17241            17231     ...
...             ...                ...              ...     ...
3163         131887             131885           131967     ...
3164         389891             389938           390076     ...
3165          52410              52410            52422     ...
```

3166	24496	24496	24506	...
3167	166994	166994	167059	...
3168	74749	74749	74807	...
3169	563626	563767	564516	...
3170	36299	36299	36428	...
3171	11668	11668	11672	...
3172	46133	46133	46244	...
3173	15885	15885	15837	...
3174	13833	13833	13826	...
3175	7083	7083	7114	...
3176	40123	40123	40222	...
3177	13249	13247	13408	...
3178	4812	4812	4813	...
3179	8569	8569	8581	...
3180	91738	91881	92271	...
3181	18106	18106	18091	...
3182	75450	75450	75472	...
3183	2484	2484	2492	...
3184	28205	28205	28259	...
3185	8667	8667	8678	...
3186	29116	29116	29146	...
3187	10247	10247	10244	...
3188	43806	43806	43593	...
3189	21294	21294	21297	...
3190	21118	21118	21102	...
3191	8533	8533	8545	...
3192	7208	7208	7181	...

	RDOMESTICMIG2011	RDOMESTICMIG2012	RDOMESTICMIG2013	RDOMESTICMIG2014	\
0	0.002295	-0.193196	0.381066	0.582002	
1	7.242091	-2.915927	-3.012349	2.265971	
2	14.832960	17.647293	21.845705	19.243287	
3	-4.728132	-2.500690	-7.056824	-3.904217	
4	-5.527043	-5.068871	-6.201001	-0.177537	
5	1.807375	-1.177622	-1.748766	-2.062535	
6	-30.953709	-5.180127	-1.130263	14.354290	
7	-14.032727	-11.684234	-5.655413	1.085428	
8	-6.155670	-4.611706	-5.524649	-4.463211	
9	-2.731639	3.849092	2.872721	-2.287222	
10	6.339327	1.113180	5.488706	-0.076806	
11	-1.372935	-2.653369	0.480044	0.456017	
12	-15.455274	-0.737028	-8.766391	-1.274984	
13	-6.194363	-17.667705	-0.318345	-8.686428	
14	-10.744102	-13.345130	4.902871	5.702648	
15	-3.673524	-5.151880	7.345821	3.654485	
16	0.377640	7.675579	-13.146535	-3.602859	
17	-0.073423	1.065051	1.762390	1.835688	
18	-4.861559	-7.504690	-6.107224	-14.645416	

19	-33.930581	-10.291443	-4.313831	-22.958017
20	6.696899	-4.612668	0.740271	3.697932
21	1.729792	3.950156	-1.864936	3.084648
22	-1.404233	-1.019628	4.071247	5.087142
23	-10.749798	-5.277150	-15.236079	-11.979785
24	-15.635599	-11.308243	-16.745678	-9.344789
25	0.294677	-9.302391	-1.748807	0.267830
26	3.235576	0.822717	1.760531	-1.507057
27	-3.449988	-3.855889	-4.822706	-1.189831
28	-1.015919	2.062637	-1.931884	-1.726932
29	-5.015601	-0.646640	-3.725937	0.296745
...	...	...	...	...
3163	-0.794876	0.785279	-2.215465	1.601149
3164	-0.765799	2.128860	0.038132	0.760109
3165	3.111756	-2.241873	6.292687	-0.441031
3166	4.930022	-2.404973	-4.097017	-4.906711
3167	0.316712	2.889873	0.833819	-2.406192
3168	-4.081523	-5.019090	-6.901200	-5.596471
3169	-0.381530	9.636214	4.487115	-4.788275
3170	3.708956	2.637812	-3.544634	-3.334877
3171	4.868258	2.804930	16.815908	-8.026420
3172	-2.843479	15.601020	-5.895711	-8.550911
3173	-7.581980	-13.081441	3.178134	-2.970641
3174	-12.847499	15.493820	19.035533	-20.550587
3175	-1.544618	-4.202564	1.397819	6.378258
3176	2.747083	7.782673	-4.990688	-12.331633
3177	14.293649	3.961413	-8.079028	-7.017803
3178	3.322604	6.208609	3.095336	-6.017222
3179	4.995063	-4.058912	-0.812583	-10.715742
3180	-1.200428	15.547274	4.787847	-1.226133
3181	-9.802564	-11.566801	13.564556	6.125989
3182	7.189319	23.066162	24.322042	-0.958472
3183	-0.401849	0.806452	29.066295	-12.603387
3184	4.582951	8.057765	7.641997	-9.252437
3185	4.373094	5.392073	2.634593	6.055759
3186	0.958559	8.425487	4.546373	3.678069
3187	-23.741784	15.272374	-40.870074	-16.596273
3188	1.072643	16.243199	-5.339774	-14.252889
3189	-1.589565	0.972695	19.525929	14.143021
3190	-17.755986	-4.916350	-6.902954	-14.215862
3191	-11.637475	-0.827815	-2.013502	-17.781491
3192	-11.752361	-8.040059	12.372583	1.533635

	RDOMESTICMIG2015	RNETMIG2011	RNETMIG2012	RNETMIG2013	RNETMIG2014	\
0	-0.467369	1.030015	0.826644	1.383282	1.724718	
1	-2.530799	7.606016	-2.626146	-2.722002	2.592270	
2	17.197872	15.844176	18.559627	22.727626	20.317142	
3	-10.543299	-4.874741	-2.758113	-7.167664	-3.978583	

4	0.177258	-5.088389	-4.363636	-5.403729	0.754533
5	-1.369970	1.859511	-0.848580	-1.402476	-1.577232
6	-16.167247	-29.001673	-2.825524	1.507017	17.243790
7	-6.529805	-13.936612	-11.586865	-5.557058	1.184103
8	-3.376322	-5.791579	-4.092677	-5.062836	-3.912834
9	1.349468	-1.821092	4.701181	3.781439	-1.290228
10	-3.239866	6.416167	1.420264	5.757384	0.230419
11	-2.253483	-0.823761	-2.447504	0.868651	0.957636
12	-5.291205	-15.528177	-0.737028	-8.766391	-1.274984
13	-5.613667	-6.077488	-17.509958	-0.159172	-8.486280
14	3.912450	-10.816697	-13.345130	4.977157	5.776708
15	-3.123961	-3.673524	-5.151880	7.345821	3.654485
16	2.214774	2.166460	11.513368	-10.438741	-0.767822
17	-0.110260	0.513964	1.469035	2.276420	2.533249
18	2.684140	-4.861559	-7.504690	-6.107224	-14.645416
19	-5.387581	-34.017138	-10.380162	-4.403703	-23.049483
20	-0.316945	6.881460	-4.559952	0.793147	3.750759
21	3.439504	2.666763	5.099293	-0.502098	4.734577
22	7.915406	-1.031427	-0.634159	4.542916	5.593387
23	-5.107706	-9.575283	-0.776637	-12.640155	-9.503292
24	-14.687232	-15.727573	-11.378047	-16.792849	-9.368689
25	0.028141	1.375159	-8.656001	-1.029539	1.198187
26	2.067820	3.674511	1.558176	2.306047	-0.951175
27	1.190902	-3.397716	-3.803428	-4.769999	-1.136950
28	-2.082234	-0.632554	2.446383	-1.518596	-1.234901
29	-2.797536	-5.132243	-0.705426	-3.785079	0.237396
...	...	...	...	...	...
3163	-0.434498	-0.431504	1.162817	-1.763330	2.104796
3164	-0.719858	0.102448	3.180527	1.189727	2.077633
3165	-0.480617	3.359933	-2.011937	6.561277	-0.134227
3166	-4.397793	5.174486	-2.160399	-3.810226	-4.535615
3167	-4.557985	0.842573	3.502335	1.531624	-1.545153
3168	-3.958322	-3.733590	-4.562809	-6.442917	-5.040889
3169	-3.221091	0.289680	10.694870	5.440390	-3.727831
3170	-9.911169	6.736119	6.433032	0.719587	1.429233
3171	5.095861	4.868258	3.144921	17.236306	-7.608378
3172	10.916963	-2.649606	15.558684	-5.916543	-8.509402
3173	-23.300971	-7.392431	-12.636926	3.623073	-2.338590
3174	-0.070403	-12.774915	16.502720	20.093063	-19.358233
3175	18.629317	-0.982939	-3.642222	2.096729	7.071547
3176	-13.673610	3.093562	8.027411	-4.747240	-12.013555
3177	-11.899450	14.886132	4.841727	-6.903896	-5.761986
3178	-5.454164	5.191569	6.001656	2.888981	-6.224712
3179	0.933652	5.227392	-4.058912	-0.812583	-10.715742
3180	0.278940	-0.973320	17.914554	6.003143	-0.207819
3181	1.555544	-9.691801	-11.566801	13.619696	6.234414
3182	-0.061057	7.689674	23.749508	25.085233	-0.110593
3183	7.492114	-0.401849	0.806452	29.066295	-12.603387

3184	-2.878980	6.486639	11.127389	10.877797	-5.585731
3185	4.662270	4.373094	4.933173	2.176403	5.598720
3186	-3.298406	2.122524	9.342778	5.523001	4.781489
3187	-22.870900	-21.092907	16.828794	-39.211861	-14.409938
3188	-14.248864	1.255221	16.243199	-5.295460	-14.075283
3189	-0.564849	0.654527	2.408578	21.160658	16.308671
3190	-12.127022	-18.136812	-5.536861	-7.521840	-14.740608
3191	1.682288	-11.990126	-1.182592	-2.250385	-18.020168
3192	6.935294	-12.032179	-8.040059	12.372583	1.533635

	RNETMIG2015
0	0.712594
1	-2.187333
2	18.293499
3	-10.543299
4	1.107861
5	-0.884411
6	-13.193961
7	-6.430868
8	-2.806406
9	2.346901
10	-2.931307
11	-1.752709
12	-5.291205
13	-5.411736
14	3.986270
15	-3.123961
16	5.350738
17	0.588052
18	2.684140
19	-5.387581
20	-0.264121
21	5.087600
22	8.417777
23	-1.998668
24	-14.711389
25	0.956790
26	2.757093
27	1.243830
28	-1.588308
29	-2.857058
...	...
3163	0.059931
3164	0.593567
3165	-0.173022
3166	-4.024395
3167	-3.685304
3168	-3.414223

```
3169    -2.091573
3170    -5.166460
3171     5.513554
3172    10.978525
3173   -22.600668
3174     1.126443
3175    19.309219
3176   -13.352750
3177   -10.635133
3178    -5.663940
3179     0.933652
3180     1.673640
3181     1.662823
3182     0.793743
3183     7.492114
3184     0.856839
3185     4.207414
3186    -2.198937
3187   -20.664059
3188   -14.070195
3189     1.520747
3190   -12.606351
3191     1.441961
3192     6.935294

[3193 rows x 100 columns]
```

```
In [8]: def answer_five():
            a = census_df.loc[census_df['COUNTY']!=0, ['STATE', 'COUNTY', 'STNAME']]
            b = a.groupby(['STNAME']).agg({'COUNTY': 'count'})
            return b['COUNTY'].idxmax()

        answer_five()
```

```
Out[8]: 'Texas'
```

1.1.2 Question 6

Only looking at the three most populous counties for each state, what are the three most populous states (in order of highest population to lowest population)? Use `CENSUS2010POP`.

This function should return a list of string values.

```
In [9]: def answer_six():
            a = census_df.loc[census_df['COUNTY']!=0, ['STATE', 'COUNTY', 'CTYNAME', 'STNAME', '
            b = a.groupby(['STNAME']).apply(pd.DataFrame.sort_values, 'CENSUS2010POP', ascending
            c = b.groupby(['STNAME']).head(3)
            d = c.groupby(['STNAME']).agg({'CENSUS2010POP': 'sum'})
            d.sort_values(by=['CENSUS2010POP'], inplace=True, ascending=False)
            e = d.head(3)
```

```
            return e['CENSUS2010POP'].index.values.tolist()

        answer_six()
```

```
Out[9]: ['California', 'Texas', 'Illinois']
```

1.1.3 Question 7

Which county has had the largest absolute change in population within the period 2010-2015? (Hint: population values are stored in columns POPESTIMATE2010 through POPESTIMATE2015, you need to consider all six columns.)

e.g. If County Population in the 5 year period is 100, 120, 80, 105, 100, 130, then its largest change in the period would be |130-80| = 50.

This function should return a single string value.

```
In [18]: import numpy as np
         def answer_seven():
             a = census_df.loc[census_df['COUNTY']!=0, ['CTYNAME', 'POPESTIMATE2010', 'POPESTIMA
             a = a.set_index('CTYNAME')
             b = pd.DataFrame((np.max(a.values, axis=1) - np.min(a.values, axis=1)), index=a.ind
             return b['LARGEST_CHANGE'].idxmax()

         answer_seven()
```

```
Out[18]: 'Harris County'
```

1.1.4 Question 8

In this datafile, the United States is broken up into four regions using the "REGION" column.

Create a query that finds the counties that belong to regions 1 or 2, whose name starts with 'Washington', and whose POPESTIMATE2015 was greater than their POPESTIMATE 2014.

This function should return a 5x2 DataFrame with the columns = ['STNAME', 'CTYNAME'] and the same index ID as the census_df (sorted ascending by index).

```
In [27]: def answer_eight():
             a = census_df.loc[census_df['COUNTY']!=0, ['STNAME', 'CTYNAME', 'REGION', 'POPESTIM
             region1 = a['REGION']==1
             region2 = a['REGION']==2
             region_condition = region1 | region2
             starts_with_washington = a['CTYNAME'].str.startswith('Washington')
             greater_popestimate = a['POPESTIMATE2015'] > a['POPESTIMATE2014']
             b = a[region_condition & starts_with_washington & greater_popestimate]
             return b[['STNAME', 'CTYNAME']]

         answer_eight()
```

```
Out[27]:          STNAME            CTYNAME
         896        Iowa  Washington County
```

```
1419     Minnesota  Washington County
2345  Pennsylvania  Washington County
2355  Rhode Island  Washington County
3163     Wisconsin  Washington County
```

In [ ]: