DataFrame Manipulation

1. Merging DataFrames

Prepare dataset.

In [1]:

```
import pandas as pd

df = pd.DataFrame([{'Name': 'Chris', 'Item Purchased': 'Sponge', 'Cost': 22.50},
                   {'Name': 'Kevyn', 'Item Purchased': 'Kitty Litter', 'Cost': 2.50},
                   {'Name': 'Filip', 'Item Purchased': 'Spoon', 'Cost': 5.00}],
                  index=['Store 1', 'Store 1', 'Store 2'])
df
```

Out[1]:

	Cost	Item Purchased	Name
Store 1	22.5	Sponge	Chris
Store 1	2.5	Kitty Litter	Kevyn
Store 2	5.0	Spoon	Filip

1.1 Adding columns.

In [3]:

```
df['Date'] = ['December 1', 'January 1', 'Mid-May']
df
```

Out[3]:

	Cost	Item Purchased	Name	Date
Store 1	22.5	Sponge	Chris	December 1
Store 1	2.5	Kitty Litter	Kevyn	January 1
Store 2	5.0	Spoon	Filip	Mid-May

- Adding values to specific rows in a chosen column.

In [5]:

```
adf = df.reset_index()
adf['Date'] = pd.Series({0: 'December 1', 2: 'mid-May'})
adf
```

Out[5]:

	index	Cost	Item Purchased	Name	Date
0	Store 1	22.5	Sponge	Chris	December 1
1	Store 1	2.5	Kitty Litter	Kevyn	NaN
2	Store 2	5.0	Spoon	Filip	mid-May

Note: if no value is given to a row, *NaN* will be assigned.

1.2 Merging Function.

Applying *Merge* function to join tables.
Prepare two tables:

In [6]:

```
staff_df = pd.DataFrame([{'Name': 'Kelly', 'Role': 'Director of HR'},
                         {'Name': 'Sally', 'Role': 'Course liasion'},
                         {'Name': 'James', 'Role': 'Grader'}])
staff_df = staff_df.set_index('Name')
student_df = pd.DataFrame([{'Name': 'James', 'School': 'Business'},
                           {'Name': 'Mike', 'School': 'Law'},
                           {'Name': 'Sally', 'School': 'Engineering'}])
student_df = student_df.set_index('Name')
print(staff_df.head())
print()
print(student_df.head())
```

```
                 Role
Name
Kelly  Director of HR
Sally  Course liasion
James          Grader

            School
Name
James     Business
Mike           Law
Sally  Engineering
```

- Merging by **Index**

In [10]:

```
pd.merge(staff_df,       # Table 1
         student_df,     # Table 2
         how = 'outer',  # outer/inner/left/right
         left_index = True, right_index = True # whether join by Index.
        )
```

Out[10]:

	Role	School
Name		
James	Grader	Business
Kelly	Director of HR	NaN
Mike	NaN	Law
Sally	Course liasion	Engineering

- Merging on **specific (multiple) columns**

In [12]:

```
staff_df = pd.DataFrame([{'First Name': 'Kelly', 'Last Name': 'Desjardins', 'Role': 'Di
rector of HR'},
                         {'First Name': 'Sally', 'Last Name': 'Brooks', 'Role': 'Course
liasion'},
                         {'First Name': 'James', 'Last Name': 'Wilde', 'Role': 'Grader'
}])
student_df = pd.DataFrame([{'First Name': 'James', 'Last Name': 'Hammond', 'School': 'B
usiness'},
                           {'First Name': 'Mike', 'Last Name': 'Smith', 'School': 'Law'
},
                           {'First Name': 'Sally', 'Last Name': 'Brooks', 'School': 'En
gineering'}])
staff_df
student_df
pd.merge(staff_df,
         student_df,
         how='inner',
         left_on=['First Name','Last Name'],
         right_on=['First Name','Last Name']
        )
```

Out[12]:

	First Name	Last Name	Role	School
0	Sally	Brooks	Course liasion	Engineering

2. Idiomatic Pandas

Preparing dataset:

In [18]:

```
%cd D:\Data Science\GitHub\Python Learning\Python-for-Data-Science\Data Files\Python Learning

import pandas as pd
df = pd.read_csv('census.csv')
df
```

```
D:\Data Science\GitHub\Python Learning\Python-for-Data-Science\Data Files
\Python Learning
```

Out[18]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010PO
0	40	3	6	1	0	Alabama	Alabama	477973
1	50	3	6	1	1	Alabama	Autauga County	5457
2	50	3	6	1	3	Alabama	Baldwin County	18226
3	50	3	6	1	5	Alabama	Barbour County	2745
4	50	3	6	1	7	Alabama	Bibb County	2291
5	50	3	6	1	9	Alabama	Blount County	5732
6	50	3	6	1	11	Alabama	Bullock County	1091
7	50	3	6	1	13	Alabama	Butler County	2094
8	50	3	6	1	15	Alabama	Calhoun County	11857
9	50	3	6	1	17	Alabama	Chambers County	3421
10	50	3	6	1	19	Alabama	Cherokee County	2598
11	50	3	6	1	21	Alabama	Chilton County	4364
12	50	3	6	1	23	Alabama	Choctaw County	1385
13	50	3	6	1	25	Alabama	Clarke County	2583
14	50	3	6	1	27	Alabama	Clay County	1393
15	50	3	6	1	29	Alabama	Cleburne County	1497
16	50	3	6	1	31	Alabama	Coffee County	4994
17	50	3	6	1	33	Alabama	Colbert County	5442
18	50	3	6	1	35	Alabama	Conecuh County	1322
19	50	3	6	1	37	Alabama	Coosa County	1153
20	50	3	6	1	39	Alabama	Covington County	3776
21	50	3	6	1	41	Alabama	Crenshaw County	1390
22	50	3	6	1	43	Alabama	Cullman County	8040
23	50	3	6	1	45	Alabama	Dale County	5025

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010PO
24	50	3	6	1	47	Alabama	Dallas County	4382
25	50	3	6	1	49	Alabama	DeKalb County	7110
26	50	3	6	1	51	Alabama	Elmore County	7930
27	50	3	6	1	53	Alabama	Escambia County	3831
28	50	3	6	1	55	Alabama	Etowah County	10443
29	50	3	6	1	57	Alabama	Fayette County	1724
...	...	...	...	...	...	...	...	
3163	50	2	3	55	131	Wisconsin	Washington County	13188
3164	50	2	3	55	133	Wisconsin	Waukesha County	38989
3165	50	2	3	55	135	Wisconsin	Waupaca County	5241
3166	50	2	3	55	137	Wisconsin	Waushara County	2449
3167	50	2	3	55	139	Wisconsin	Winnebago County	16699
3168	50	2	3	55	141	Wisconsin	Wood County	7474
3169	40	4	8	56	0	Wyoming	Wyoming	56362
3170	50	4	8	56	1	Wyoming	Albany County	3629
3171	50	4	8	56	3	Wyoming	Big Horn County	1166
3172	50	4	8	56	5	Wyoming	Campbell County	4613
3173	50	4	8	56	7	Wyoming	Carbon County	1588
3174	50	4	8	56	9	Wyoming	Converse County	1383
3175	50	4	8	56	11	Wyoming	Crook County	708
3176	50	4	8	56	13	Wyoming	Fremont County	4012
3177	50	4	8	56	15	Wyoming	Goshen County	1324
3178	50	4	8	56	17	Wyoming	Hot Springs County	481
3179	50	4	8	56	19	Wyoming	Johnson County	856
3180	50	4	8	56	21	Wyoming	Laramie County	9173
3181	50	4	8	56	23	Wyoming	Lincoln County	1810

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010PO
3182	50	4	8	56	25	Wyoming	Natrona County	7545
3183	50	4	8	56	27	Wyoming	Niobrara County	248
3184	50	4	8	56	29	Wyoming	Park County	2820
3185	50	4	8	56	31	Wyoming	Platte County	866
3186	50	4	8	56	33	Wyoming	Sheridan County	2911
3187	50	4	8	56	35	Wyoming	Sublette County	1024
3188	50	4	8	56	37	Wyoming	Sweetwater County	4380
3189	50	4	8	56	39	Wyoming	Teton County	2129
3190	50	4	8	56	41	Wyoming	Uinta County	2111
3191	50	4	8	56	43	Wyoming	Washakie County	853
3192	50	4	8	56	45	Wyoming	Weston County	720

3193 rows × 100 columns

2.1 Method Chaining

This can make your script more readable when applying multiple methods to the dataset. </br> **Example:**

In [24]:

```
(df.where(df['SUMLEV']==50)
    .dropna()
    .set_index(['STNAME','CTYNAME'])
    .rename(columns={'ESTIMATESBASE2010': 'Estimates Base 2010'}))
```

Out[24]:

		SUMLEV	REGION	DIVISION	STATE	COUNTY	CENSUS2010POP	E B 2
STNAME	CTYNAME							
Alabama	Autauga County	50.0	3.0	6.0	1.0	1.0	54571.0	
	Baldwin County	50.0	3.0	6.0	1.0	3.0	182265.0	
	Barbour County	50.0	3.0	6.0	1.0	5.0	27457.0	
	Bibb County	50.0	3.0	6.0	1.0	7.0	22915.0	
	Blount County	50.0	3.0	6.0	1.0	9.0	57322.0	
	Bullock County	50.0	3.0	6.0	1.0	11.0	10914.0	
	Butler County	50.0	3.0	6.0	1.0	13.0	20947.0	
	Calhoun County	50.0	3.0	6.0	1.0	15.0	118572.0	
	Chambers County	50.0	3.0	6.0	1.0	17.0	34215.0	
	Cherokee County	50.0	3.0	6.0	1.0	19.0	25989.0	
	Chilton County	50.0	3.0	6.0	1.0	21.0	43643.0	
	Choctaw County	50.0	3.0	6.0	1.0	23.0	13859.0	
	Clarke County	50.0	3.0	6.0	1.0	25.0	25833.0	
	Clay County	50.0	3.0	6.0	1.0	27.0	13932.0	
	Cleburne County	50.0	3.0	6.0	1.0	29.0	14972.0	
	Coffee County	50.0	3.0	6.0	1.0	31.0	49948.0	
	Colbert County	50.0	3.0	6.0	1.0	33.0	54428.0	
	Conecuh County	50.0	3.0	6.0	1.0	35.0	13228.0	
	Coosa County	50.0	3.0	6.0	1.0	37.0	11539.0	
	Covington County	50.0	3.0	6.0	1.0	39.0	37765.0	
	Crenshaw County	50.0	3.0	6.0	1.0	41.0	13906.0	
	Cullman County	50.0	3.0	6.0	1.0	43.0	80406.0	

		SUMLEV	REGION	DIVISION	STATE	COUNTY	CENSUS2010POP	E B 2
STNAME	**CTYNAME**							
	Dale County	50.0	3.0	6.0	1.0	45.0	50251.0	
	Dallas County	50.0	3.0	6.0	1.0	47.0	43820.0	
	DeKalb County	50.0	3.0	6.0	1.0	49.0	71109.0	
	Elmore County	50.0	3.0	6.0	1.0	51.0	79303.0	
	Escambia County	50.0	3.0	6.0	1.0	53.0	38319.0	
	Etowah County	50.0	3.0	6.0	1.0	55.0	104430.0	
	Fayette County	50.0	3.0	6.0	1.0	57.0	17241.0	
	Franklin County	50.0	3.0	6.0	1.0	59.0	31704.0	
...	**...**	...	...	...	...	...	...	
Wisconsin	**Washburn County**	50.0	2.0	3.0	55.0	129.0	15911.0	
	Washington County	50.0	2.0	3.0	55.0	131.0	131887.0	
	Waukesha County	50.0	2.0	3.0	55.0	133.0	389891.0	
	Waupaca County	50.0	2.0	3.0	55.0	135.0	52410.0	
	Waushara County	50.0	2.0	3.0	55.0	137.0	24496.0	
	Winnebago County	50.0	2.0	3.0	55.0	139.0	166994.0	
	Wood County	50.0	2.0	3.0	55.0	141.0	74749.0	
Wyoming	**Albany County**	50.0	4.0	8.0	56.0	1.0	36299.0	
	Big Horn County	50.0	4.0	8.0	56.0	3.0	11668.0	
	Campbell County	50.0	4.0	8.0	56.0	5.0	46133.0	
	Carbon County	50.0	4.0	8.0	56.0	7.0	15885.0	
	Converse County	50.0	4.0	8.0	56.0	9.0	13833.0	
	Crook County	50.0	4.0	8.0	56.0	11.0	7083.0	
	Fremont County	50.0	4.0	8.0	56.0	13.0	40123.0	
	Goshen County	50.0	4.0	8.0	56.0	15.0	13249.0	

		SUMLEV	REGION	DIVISION	STATE	COUNTY	CENSUS2010POP	E B 2
STNAME	**CTYNAME**							
	Hot Springs County	50.0	4.0	8.0	56.0	17.0	4812.0	
	Johnson County	50.0	4.0	8.0	56.0	19.0	8569.0	
	Laramie County	50.0	4.0	8.0	56.0	21.0	91738.0	
	Lincoln County	50.0	4.0	8.0	56.0	23.0	18106.0	
	Natrona County	50.0	4.0	8.0	56.0	25.0	75450.0	
	Niobrara County	50.0	4.0	8.0	56.0	27.0	2484.0	
	Park County	50.0	4.0	8.0	56.0	29.0	28205.0	
	Platte County	50.0	4.0	8.0	56.0	31.0	8667.0	
	Sheridan County	50.0	4.0	8.0	56.0	33.0	29116.0	
	Sublette County	50.0	4.0	8.0	56.0	35.0	10247.0	
	Sweetwater County	50.0	4.0	8.0	56.0	37.0	43806.0	
	Teton County	50.0	4.0	8.0	56.0	39.0	21294.0	
	Uinta County	50.0	4.0	8.0	56.0	41.0	21118.0	
	Washakie County	50.0	4.0	8.0	56.0	43.0	8533.0	
	Weston County	50.0	4.0	8.0	56.0	45.0	7208.0	

3142 rows × 98 columns

2.2 .apply() Method

This method will be very useful when you want to apply one function to every column or row of a table.

In [27]:

```
import numpy as np
rows = ['POPESTIMATE2010',
        'POPESTIMATE2011',
        'POPESTIMATE2012',
        'POPESTIMATE2013',
        'POPESTIMATE2014',
        'POPESTIMATE2015']
df.apply(lambda x: np.max(x[rows]), # function that will be applied
         axis=1                     # 0: apply to columns, 1: apply to rows
        )
```

Out[27]:

```
0       4858979
1         55347
2        203709
3         27341
4         22861
5         57776
6         10887
7         20944
8        118437
9         34153
10        26084
11        43943
12        13841
13        25767
14        13880
15        15072
16        51211
17        54514
18        13208
19        11758
20        38060
21        13963
22        82005
23        50358
24        43803
25        71387
26        81468
27        38309
28       104442
29        17231
         ...   
3163     133674
3164     396488
3165      52422
3166      24581
3167     169639
3168      74807
3169     586107
3170      37956
3171      12022
3172      49220
3173      15856
3174      14343
3175       7444
3176      41129
3177      13666
3178       4846
3179       8636
3180      97121
3181      18722
3182      82178
3183       2548
3184      29237
3185       8812
3186      30020
3187      10418
3188      45162
3189      23125
3190      21102
```

```
3191       8545
3192       7234
Length: 3193, dtype: int64
```

3. Group by

Prepare dataset.

In [28]:

```
import pandas as pd
import numpy as np
df = pd.read_csv('census.csv')
df = df[df['SUMLEV']==50]
df
```

Out[28]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010PO
1	50	3	6	1	1	Alabama	Autauga County	5457
2	50	3	6	1	3	Alabama	Baldwin County	18226
3	50	3	6	1	5	Alabama	Barbour County	2745
4	50	3	6	1	7	Alabama	Bibb County	2291
5	50	3	6	1	9	Alabama	Blount County	5732
6	50	3	6	1	11	Alabama	Bullock County	1091
7	50	3	6	1	13	Alabama	Butler County	2094
8	50	3	6	1	15	Alabama	Calhoun County	11857
9	50	3	6	1	17	Alabama	Chambers County	3421
10	50	3	6	1	19	Alabama	Cherokee County	2598
11	50	3	6	1	21	Alabama	Chilton County	4364
12	50	3	6	1	23	Alabama	Choctaw County	1385
13	50	3	6	1	25	Alabama	Clarke County	2583
14	50	3	6	1	27	Alabama	Clay County	1393
15	50	3	6	1	29	Alabama	Cleburne County	1497
16	50	3	6	1	31	Alabama	Coffee County	4994
17	50	3	6	1	33	Alabama	Colbert County	5442
18	50	3	6	1	35	Alabama	Conecuh County	1322
19	50	3	6	1	37	Alabama	Coosa County	1153
20	50	3	6	1	39	Alabama	Covington County	3776
21	50	3	6	1	41	Alabama	Crenshaw County	1390
22	50	3	6	1	43	Alabama	Cullman County	8040
23	50	3	6	1	45	Alabama	Dale County	5025
24	50	3	6	1	47	Alabama	Dallas County	4382

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010PO
25	50	3	6	1	49	Alabama	DeKalb County	7110
26	50	3	6	1	51	Alabama	Elmore County	7930
27	50	3	6	1	53	Alabama	Escambia County	3831
28	50	3	6	1	55	Alabama	Etowah County	10443
29	50	3	6	1	57	Alabama	Fayette County	1724
30	50	3	6	1	59	Alabama	Franklin County	3170
...	...	...	...	...	...	...	...	
3162	50	2	3	55	129	Wisconsin	Washburn County	1591
3163	50	2	3	55	131	Wisconsin	Washington County	13188
3164	50	2	3	55	133	Wisconsin	Waukesha County	38989
3165	50	2	3	55	135	Wisconsin	Waupaca County	5241
3166	50	2	3	55	137	Wisconsin	Waushara County	2449
3167	50	2	3	55	139	Wisconsin	Winnebago County	16699
3168	50	2	3	55	141	Wisconsin	Wood County	7474
3170	50	4	8	56	1	Wyoming	Albany County	3629
3171	50	4	8	56	3	Wyoming	Big Horn County	1166
3172	50	4	8	56	5	Wyoming	Campbell County	4613
3173	50	4	8	56	7	Wyoming	Carbon County	1588
3174	50	4	8	56	9	Wyoming	Converse County	1383
3175	50	4	8	56	11	Wyoming	Crook County	708
3176	50	4	8	56	13	Wyoming	Fremont County	4012
3177	50	4	8	56	15	Wyoming	Goshen County	1324
3178	50	4	8	56	17	Wyoming	Hot Springs County	481
3179	50	4	8	56	19	Wyoming	Johnson County	856
3180	50	4	8	56	21	Wyoming	Laramie County	9173

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010PO
3181	50	4	8	56	23	Wyoming	Lincoln County	1810
3182	50	4	8	56	25	Wyoming	Natrona County	7545
3183	50	4	8	56	27	Wyoming	Niobrara County	248
3184	50	4	8	56	29	Wyoming	Park County	2820
3185	50	4	8	56	31	Wyoming	Platte County	866
3186	50	4	8	56	33	Wyoming	Sheridan County	2911
3187	50	4	8	56	35	Wyoming	Sublette County	1024
3188	50	4	8	56	37	Wyoming	Sweetwater County	4380
3189	50	4	8	56	39	Wyoming	Teton County	2129
3190	50	4	8	56	41	Wyoming	Uinta County	2111
3191	50	4	8	56	43	Wyoming	Washakie County	853
3192	50	4	8	56	45	Wyoming	Weston County	720

3142 rows × 100 columns

Groupby function will create *Groupby* object referencing to each part of the split dataframe segmented by the selected column.

3.1.1 Groupby column

In [29]:

```python
for group, frame in df.groupby('STNAME'):
    avg = np.average(frame['CENSUS2010POP'])
    print('Counties in state ' + group + ' have an average population of ' + str(avg))
```

```
Counties in state Alabama have an average population of 71339.34328358209
Counties in state Alaska have an average population of 24490.724137931036
Counties in state Arizona have an average population of 426134.4666666667
Counties in state Arkansas have an average population of 38878.90666666667
Counties in state California have an average population of 642309.58620689
66
Counties in state Colorado have an average population of 78581.1875
Counties in state Connecticut have an average population of 446762.125
Counties in state Delaware have an average population of 299311.3333333333
Counties in state District of Columbia have an average population of 60172
3.0
Counties in state Florida have an average population of 280616.5671641791
Counties in state Georgia have an average population of 60928.63522012578
Counties in state Hawaii have an average population of 272060.2
Counties in state Idaho have an average population of 35626.86363636364
Counties in state Illinois have an average population of 125790.5098039215
7
Counties in state Indiana have an average population of 70476.10869565218
Counties in state Iowa have an average population of 30771.262626262625
Counties in state Kansas have an average population of 27172.55238095238
Counties in state Kentucky have an average population of 36161.39166666667
Counties in state Louisiana have an average population of 70833.9375
Counties in state Maine have an average population of 83022.5625
Counties in state Maryland have an average population of 240564.6666666666
6
Counties in state Massachusetts have an average population of 467687.78571
428574
Counties in state Michigan have an average population of 119080.0
Counties in state Minnesota have an average population of 60964.6551724137
9
Counties in state Mississippi have an average population of 36186.54878048
781
Counties in state Missouri have an average population of 52077.62608695652
Counties in state Montana have an average population of 17668.125
Counties in state Nebraska have an average population of 19638.07526881720
3
Counties in state Nevada have an average population of 158855.9411764706
Counties in state New Hampshire have an average population of 131647.0
Counties in state New Jersey have an average population of 418661.61904761
905
Counties in state New Mexico have an average population of 62399.363636363
64
Counties in state New York have an average population of 312550.0322580645
4
Counties in state North Carolina have an average population of 95354.83
Counties in state North Dakota have an average population of 12690.3962264
15094
Counties in state Ohio have an average population of 131096.63636363635
Counties in state Oklahoma have an average population of 48718.84415584415
5
Counties in state Oregon have an average population of 106418.72222222222
Counties in state Pennsylvania have an average population of 189587.746268
65672
Counties in state Rhode Island have an average population of 210513.4
Counties in state South Carolina have an average population of 100551.3913
0434782
Counties in state South Dakota have an average population of 12336.0606060
60606
Counties in state Tennessee have an average population of 66801.1052631579
Counties in state Texas have an average population of 98998.27165354331
Counties in state Utah have an average population of 95306.37931034483
Counties in state Vermont have an average population of 44695.78571428572
Counties in state Virginia have an average population of 60111.29323308271
Counties in state Washington have an average population of 172424.10256410
256
Counties in state West Virginia have an average population of 33690.8
Counties in state Wisconsin have an average population of 78985.9166666666
7
Counties in state Wyoming have an average population of 24505.478260869564
```

3.1.2 Groupby a function

DataFrame can also be grouped by a function.
Note: the column being used to group the dataset has to be set as the index.

In [30]:

```
df = df.set_index('STNAME')

def fun(item):
    if item[0]<'M':
        return 0
    if item[0]<'Q':
        return 1
    return 2

for group, frame in df.groupby(fun):
    print('There are ' +
          str(len(frame)) +
          ' records in group ' +
          str(group) +
          ' for processing.')
```

```
There are 1177 records in group 0 for processing.
There are 1134 records in group 1 for processing.
There are 831 records in group 2 for processing.
```

3.2 Groupby & Aggregate

Create a summary statistics of a column with data grouped by another column.

In [32]:

```
(df.groupby('STNAME')
  .agg({'CENSUS2010POP':    # Column being aggregated
        np.average}))       # Function being applied
```

Out[32]:

	CENSUS2010POP
STNAME	
Alabama	71339.343284
Alaska	24490.724138
Arizona	426134.466667
Arkansas	38878.906667
California	642309.586207
Colorado	78581.187500
Connecticut	446762.125000
Delaware	299311.333333
District of Columbia	601723.000000
Florida	280616.567164
Georgia	60928.635220
Hawaii	272060.200000
Idaho	35626.863636
Illinois	125790.509804
Indiana	70476.108696
Iowa	30771.262626
Kansas	27172.552381
Kentucky	36161.391667
Louisiana	70833.937500
Maine	83022.562500
Maryland	240564.666667
Massachusetts	467687.785714
Michigan	119080.000000
Minnesota	60964.655172
Mississippi	36186.548780
Missouri	52077.626087
Montana	17668.125000
Nebraska	19638.075269
Nevada	158855.941176
New Hampshire	131647.000000
New Jersey	418661.619048
New Mexico	62399.363636
New York	312550.032258
North Carolina	95354.830000
North Dakota	12690.396226
Ohio	131096.636364

	CENSUS2010POP
STNAME	
Oklahoma	48718.844156
Oregon	106418.722222
Pennsylvania	189587.746269
Rhode Island	210513.400000
South Carolina	100551.391304
South Dakota	12336.060606
Tennessee	66801.105263
Texas	98998.271654
Utah	95306.379310
Vermont	44695.785714
Virginia	60111.293233
Washington	172424.102564
West Virginia	33690.800000
Wisconsin	78985.916667
Wyoming	24505.478261

In [39]:

```
df.head()
```

Out[39]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	CTYNAME	CENSUS2010POP	ESTI
STNAME								
Alabama	50	3	6	1	1	Autauga County	54571	
Alabama	50	3	6	1	3	Baldwin County	182265	
Alabama	50	3	6	1	5	Barbour County	27457	
Alabama	50	3	6	1	7	Bibb County	22915	
Alabama	50	3	6	1	9	Blount County	57322	

5 rows × 99 columns

In [40]:

```
(df.groupby(level=0)['CENSUS2010POP'].agg({'avg': np.average, 'sum': np.sum}))
```

```
D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: FutureWarning: usi
ng a dict on a Series for aggregation
is deprecated and will be removed in a future version
  """Entry point for launching an IPython kernel.
```

Out[40]:

	avg	sum
STNAME		
Alabama	71339.343284	4779736
Alaska	24490.724138	710231
Arizona	426134.466667	6392017
Arkansas	38878.906667	2915918
California	642309.586207	37253956
Colorado	78581.187500	5029196
Connecticut	446762.125000	3574097
Delaware	299311.333333	897934
District of Columbia	601723.000000	601723
Florida	280616.567164	18801310
Georgia	60928.635220	9687653
Hawaii	272060.200000	1360301
Idaho	35626.863636	1567582
Illinois	125790.509804	12830632
Indiana	70476.108696	6483802
Iowa	30771.262626	3046355
Kansas	27172.552381	2853118
Kentucky	36161.391667	4339367
Louisiana	70833.937500	4533372
Maine	83022.562500	1328361
Maryland	240564.666667	5773552
Massachusetts	467687.785714	6547629
Michigan	119080.000000	9883640
Minnesota	60964.655172	5303925
Mississippi	36186.548780	2967297
Missouri	52077.626087	5988927
Montana	17668.125000	989415
Nebraska	19638.075269	1826341
Nevada	158855.941176	2700551
New Hampshire	131647.000000	1316470
New Jersey	418661.619048	8791894
New Mexico	62399.363636	2059179
New York	312550.032258	19378102
North Carolina	95354.830000	9535483
North Dakota	12690.396226	672591
Ohio	131096.636364	11536504

	avg	sum
STNAME		
Oklahoma	48718.844156	3751351
Oregon	106418.722222	3831074
Pennsylvania	189587.746269	12702379
Rhode Island	210513.400000	1052567
South Carolina	100551.391304	4625364
South Dakota	12336.060606	814180
Tennessee	66801.105263	6346105
Texas	98998.271654	25145561
Utah	95306.379310	2763885
Vermont	44695.785714	625741
Virginia	60111.293233	7994802
Washington	172424.102564	6724540
West Virginia	33690.800000	1852994
Wisconsin	78985.916667	5686986
Wyoming	24505.478261	563626

4. Scales

In [41]:

```
import pandas as pd
df = pd.DataFrame(['A+', 'A', 'A-', 'B+', 'B', 'B-', 'C+', 'C', 'C-', 'D+', 'D'],
                  index=['excellent', 'excellent', 'excellent', 'good', 'good', 'good',
'ok', 'ok', 'ok', 'poor', 'poor'])
df.rename(columns={0: 'Grades'}, inplace=True)
```

4.1 Creating Cateogrical Variable

In [42]:

```
df['Grades'].astype('category').head()
```

Out[42]:

```
excellent    A+
excellent     A
excellent    A-
good         B+
good          B
Name: Grades, dtype: category
Categories (11, object): [A, A+, A-, B, ..., C+, C-, D, D+]
```

We can set order for categorical variable by using *ordered* parameter.

In [43]:

```
grades = df['Grades'].astype('category',
                             categories=['D', 'D+', 'C-', 'C', 'C+', 'B-', 'B', 'B+',
'A-', 'A', 'A+'],
                             ordered=True)
grades.head()
```

```
D:\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3296: Futu
reWarning: specifying 'categories' or 'ordered' in .astype() is deprecate
d; pass a CategoricalDtype instead
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Out[43]:

```
excellent    A+
excellent     A
excellent    A-
good         B+
good          B
Name: Grades, dtype: category
Categories (11, object): [D < D+ < C- < C ... B+ < A- < A < A+]
```

4.2 Tranferring Numerical Variable to Categorical Variable

Using *cut* function to split into equally-spaced.

In [44]:

```
import numpy as np
df = pd.read_csv('census.csv')
df = df[df['SUMLEV']==50]
df = df.set_index('STNAME').groupby(level=0)['CENSUS2010POP'].agg({'avg': np.average})
pd.cut(df['avg'],10)
```

```
D:\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: FutureWarning: usi
ng a dict on a Series for aggregation
is deprecated and will be removed in a future version
  after removing the cwd from sys.path.
```

```
Out[44]:

STNAME
Alabama                   (11706.087, 75333.413]
Alaska                    (11706.087, 75333.413]
Arizona                 (390320.176, 453317.529]
Arkansas                  (11706.087, 75333.413]
California              (579312.234, 642309.586]
Colorado                 (75333.413, 138330.766]
Connecticut             (390320.176, 453317.529]
Delaware                (264325.471, 327322.823]
District of Columbia    (579312.234, 642309.586]
Florida                 (264325.471, 327322.823]
Georgia                   (11706.087, 75333.413]
Hawaii                  (264325.471, 327322.823]
Idaho                     (11706.087, 75333.413]
Illinois                 (75333.413, 138330.766]
Indiana                   (11706.087, 75333.413]
Iowa                      (11706.087, 75333.413]
Kansas                    (11706.087, 75333.413]
Kentucky                  (11706.087, 75333.413]
Louisiana                 (11706.087, 75333.413]
Maine                    (75333.413, 138330.766]
Maryland                (201328.118, 264325.471]
Massachusetts           (453317.529, 516314.881]
Michigan                 (75333.413, 138330.766]
Minnesota                 (11706.087, 75333.413]
Mississippi               (11706.087, 75333.413]
Missouri                  (11706.087, 75333.413]
Montana                   (11706.087, 75333.413]
Nebraska                  (11706.087, 75333.413]
Nevada                  (138330.766, 201328.118]
New Hampshire            (75333.413, 138330.766]
New Jersey              (390320.176, 453317.529]
New Mexico                (11706.087, 75333.413]
New York                (264325.471, 327322.823]
North Carolina           (75333.413, 138330.766]
North Dakota              (11706.087, 75333.413]
Ohio                     (75333.413, 138330.766]
Oklahoma                  (11706.087, 75333.413]
Oregon                   (75333.413, 138330.766]
Pennsylvania            (138330.766, 201328.118]
Rhode Island            (201328.118, 264325.471]
South Carolina           (75333.413, 138330.766]
South Dakota              (11706.087, 75333.413]
Tennessee                 (11706.087, 75333.413]
Texas                    (75333.413, 138330.766]
Utah                     (75333.413, 138330.766]
Vermont                   (11706.087, 75333.413]
Virginia                  (11706.087, 75333.413]
Washington              (138330.766, 201328.118]
West Virginia             (11706.087, 75333.413]
Wisconsin                (75333.413, 138330.766]
Wyoming                   (11706.087, 75333.413]
Name: avg, dtype: category
Categories (10, interval[float64]): [(11706.087, 75333.413] < (75333.413,
138330.766] < (138330.766, 201328.118] < (201328.118, 264325.471] ... (390
320.176, 453317.529] < (453317.529, 516314.881] < (516314.881, 579312.234]
< (579312.234, 642309.586]]
```

5. Pivot Tables

Prepare datasets:

In [45]:

```
df = pd.read_csv('cars.csv')
df.head()
```

Out[45]:

	YEAR	Make	Model	Size	(kW)	Unnamed: 5	TYPE	CITY (kWh/100 km)	HW (kWh/1(kn
0	2012	MITSUBISHI	i-MiEV	SUBCOMPACT	49	A1	B	16.9	21
1	2012	NISSAN	LEAF	MID-SIZE	80	A1	B	19.3	23
2	2013	FORD	FOCUS ELECTRIC	COMPACT	107	A1	B	19.0	21
3	2013	MITSUBISHI	i-MiEV	SUBCOMPACT	49	A1	B	16.9	21
4	2013	NISSAN	LEAF	MID-SIZE	80	A1	B	19.3	23

Creating pivot table:

In [46]:

```
df.pivot_table(values='(kW)',
               index='YEAR',
               columns='Make',
               aggfunc=[np.mean,np.min],
               margins=True)
```

Out[46]:

	mean								
Make	**BMW**	**CHEVROLET**	**FORD**	**KIA**	**MITSUBISHI**	**NISSAN**	**SMART**	**TESLA**	**All**
YEAR									
2012	NaN	NaN	NaN	NaN	49.0	80.0	NaN	NaN	64.500
2013	NaN	NaN	107.0	NaN	49.0	80.0	35.0	280.000000	158.444
2014	NaN	104.0	107.0	NaN	49.0	80.0	35.0	268.333333	135.000
2015	125.0	104.0	107.0	81.0	49.0	80.0	35.0	320.666667	181.428
2016	125.0	104.0	107.0	81.0	49.0	80.0	35.0	409.700000	252.263
All	125.0	104.0	107.0	81.0	49.0	80.0	35.0	345.478261	190.622

6. Date Functionality in Pandas

6.1 Timestamp

Convert string to timestamp.

In [47]:

```
pd.Timestamp('9/1/2016 10:05AM')
```

Out[47]:

```
Timestamp('2016-09-01 10:05:00')
```

6.2 Perid

In [48]:

```
pd.Period('1/2016')
```

Out[48]:

```
Period('2016-01', 'M')
```

In [49]:

```
pd.Period('3/5/2016')
```

Out[49]:

```
Period('2016-03-05', 'D')
```

6.3 Date Index

- Using datetime as **index** of a series

In [51]:

```
t1 = pd.Series(list('abc'),
               [pd.Timestamp('2016-09-01'),     # set index for series
                pd.Timestamp('2016-09-02'),
                pd.Timestamp('2016-09-03')])
t1
```

Out[51]:

```
2016-09-01    a
2016-09-02    b
2016-09-03    c
dtype: object
```

- Using Period as **index** of a series

In [53]:

```
t2 = pd.Series(list('def'),
               [pd.Period('2016-09'),      # set index for series
                pd.Period('2016-10'),
                pd.Period('2016-11')])
t2
```

Out[53]:

```
2016-09    d
2016-10    e
2016-11    f
Freq: M, dtype: object
```

6.4 Converting to Datetime

In [56]:

```
d1 = ['2 June 2013', 'Aug 29, 2014', '2015-06-26', '7/12/16']
ts3 = pd.DataFrame(np.random.randint(10, 100, (4,3)),
                   index=d1,
                   columns=list('abc'))
# Converting the strings to datetime format.
ts3.index = pd.to_datetime(ts3.index)
ts3
```

Out[56]:

	a	b	c
2013-06-02	72	11	46
2014-08-29	20	92	51
2015-06-26	66	67	10
2016-07-12	23	51	12

In [57]:

```
pd.to_datetime('4.7.12', dayfirst = True)
```

Out[57]:

```
Timestamp('2012-07-04 00:00:00')
```

6.5 Timedeltas

- Date difference.

In [58]:

```
pd.Timestamp('9/3/2016') - pd.Timestamp('9/1/2016')
```

Out[58]:

```
Timedelta('2 days 00:00:00')
```

In [59]:

```
pd.Timestamp('9/2/2016 8:10AM') + pd.Timedelta('12D 3H')
```

Out[59]:

```
Timestamp('2016-09-14 11:10:00')
```

6.6 Working with Dates in a Dataframe

- **Generate list of Date**

In [60]:

```
dates = pd.date_range('10-01-2016',    # Start date
                      periods=9,       # Number of periods to generate
                      freq='2W-SUN'    # Length of period
                     )
dates
```

Out[60]:

```
DatetimeIndex(['2016-10-02', '2016-10-16', '2016-10-30', '2016-11-13',
               '2016-11-27', '2016-12-11', '2016-12-25', '2017-01-08',
               '2017-01-22'],
              dtype='datetime64[ns]', freq='2W-SUN')
```

Using this list of date as the index of a DataFrame.

In [61]:

```
df = pd.DataFrame({'Count 1': 100 + np.random.randint(-5, 10, 9).cumsum(),
                  'Count 2': 120 + np.random.randint(-5, 10, 9)}, index=dates)
df
```

Out[61]:

	Count 1	Count 2
2016-10-02	95	117
2016-10-16	101	123
2016-10-30	103	122
2016-11-13	112	129
2016-11-27	111	115
2016-12-11	113	127
2016-12-25	109	127
2017-01-08	117	122
2017-01-22	116	119

Extract the weekday name of those dates.

In [62]:

```
df.index.weekday_name
```

Out[62]:

```
Index(['Sunday', 'Sunday', 'Sunday', 'Sunday', 'Sunday', 'Sunday', 'Sunda
y',
       'Sunday', 'Sunday'],
      dtype='object')
```

- **Using Resample Function**

This function can help us chnage the time interval of a DataFrame to something else, i.e. aggregate monthly data into yearly data.

In [63]:

```
df.resample('M').mean()
```

Out[63]:

	Count 1	Count 2
2016-10-31	99.666667	120.666667
2016-11-30	111.500000	122.000000
2016-12-31	111.000000	127.000000
2017-01-31	116.500000	120.500000

- **Slicing DataFrame by Date index**

Selecting by year:

In [64]:

```
df['2017']
```

Out[64]:

	Count 1	Count 2
2017-01-08	117	122
2017-01-22	116	119

Selecting by year-month:

In [65]:

```
df['2016-12']
```

Out[65]:

	Count 1	Count 2
2016-12-11	113	127
2016-12-25	109	127

In [66]:

```
df['2016-12':]
```

Out[66]:

	Count 1	Count 2
2016-12-11	113	127
2016-12-25	109	127
2017-01-08	117	122
2017-01-22	116	119

- **Forward Filling**

In [67]:

```
df.asfreq('W', method = 'ffill')
```

Out[67]:

	Count 1	Count 2
2016-10-02	95	117
2016-10-09	95	117
2016-10-16	101	123
2016-10-23	101	123
2016-10-30	103	122
2016-11-06	103	122
2016-11-13	112	129
2016-11-20	112	129
2016-11-27	111	115
2016-12-04	111	115
2016-12-11	113	127
2016-12-18	113	127
2016-12-25	109	127
2017-01-01	109	127
2017-01-08	117	122
2017-01-15	117	122
2017-01-22	116	119