*You are currently looking at **version 1.0** 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 (https://www.coursera.org/learn/python-data-analysis/resources/0dhYG) course resource.*

Merging Dataframes

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

In [2]:
```
df['Date'] = ['December 1', 'January 1', 'mid-May']
df
```

Out[2]:

	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

In [3]:
```
df['Delivered'] = True
df
```

Out[3]:

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

In [4]:
```
df['Feedback'] = ['Positive', None, 'Negative']
df
```

Out[4]:

	Cost	Item Purchased	Name	Date	Delivered	Feedback
Store 1	22.5	Sponge	Chris	December 1	True	Positive
Store 1	2.5	Kitty Litter	Kevyn	January 1	True	None
Store 2	5.0	Spoon	Filip	mid-May	True	Negative

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	Delivered	Feedback
0	Store 1	22.5	Sponge	Chris	December 1	True	Positive
1	Store 1	2.5	Kitty Litter	Kevyn	NaN	True	None
2	Store 2	5.0	Spoon	Filip	mid-May	True	Negative

In [6]:
```
import pandas as pd

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)
print()
print(student_df)
```

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

            School
Name
James     Business
Mike           Law
Sally  Engineering
```

In [7]:
```
pd.merge(staff_df, student_df, how='outer', left_index=True, right_index=True)
```

Out[7]:

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

In [8]:
```
pd.merge(staff_df, student_df, how='inner', left_index=True, right_index=True)
```

Out[8]:

	Role	School
Name		
James	Grader	Business
Sally	Course liasion	Engineering

In [9]:
```
pd.merge(staff_df, student_df, how='left', left_index=True, right_index=True)
```

Out[9]:

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

In [10]:
```
pd.merge(staff_df, student_df, how='right', left_index=True, right_index=True)
```

Out[10]:

	Role	School
Name		
James	Grader	Business
Mike	NaN	Law
Sally	Course liasion	Engineering

In [11]:
```
staff_df = staff_df.reset_index()
student_df = student_df.reset_index()
pd.merge(staff_df, student_df, how='left', left_on='Name', right_on='Name')
```

Out[11]:

	Name	Role	School
0	Kelly	Director of HR	NaN
1	Sally	Course liasion	Engineering
2	James	Grader	Business

In [12]:
```
staff_df = pd.DataFrame([{'Name': 'Kelly', 'Role': 'Director of HR', 'Location':
'State Street'},
                         {'Name': 'Sally', 'Role': 'Course liasion', 'Location':
'Washington Avenue'},
                         {'Name': 'James', 'Role': 'Grader', 'Location': 'Washin
gton Avenue'}])
student_df = pd.DataFrame([{'Name': 'James', 'School': 'Business', 'Location': '
1024 Billiard Avenue'},
                           {'Name': 'Mike', 'School': 'Law', 'Location': 'Frater
nity House #22'},
                           {'Name': 'Sally', 'School': 'Engineering', 'Location
': '512 Wilson Crescent'}])
staff_df = staff_df.set_index('Name')
student_df = student_df.set_index('Name')
pd.merge(staff_df, student_df, how='outer', left_index=True, right_index=True)
```

Out[12]:

	Location_x	Role	Location_y	School
Name				
James	Washington Avenue	Grader	1024 Billiard Avenue	Business
Kelly	State Street	Director of HR	NaN	NaN
Mike	NaN	NaN	Fraternity House #22	Law
Sally	Washington Avenue	Course liasion	512 Wilson Crescent	Engineering

In [13]:
```
staff_df = pd.DataFrame([{'First Name': 'Kelly', 'Last Name': 'Desjardins', 'Rol
e': 'Director 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', 'Scho
ol': 'Business'},
                           {'First Name': 'Mike', 'Last Name': 'Smith', 'School
': 'Law'},
                           {'First Name': 'Sally', 'Last Name': 'Brooks', 'Schoo
l': 'Engineering'}])
staff_df
student_df
pd.merge(staff_df, student_df, how='outer', left_on=['First Name','Last Name'],
right_on=['First Name','Last Name'])
```

Out[13]:

	First Name	Last Name	Role	School
0	Kelly	Desjardins	Director of HR	NaN
1	Sally	Brooks	Course liasion	Engineering
2	James	Wilde	Grader	NaN
3	James	Hammond	NaN	Business
4	Mike	Smith	NaN	Law

Idiomatic Pandas: Making Code Pandorable

In [14]:
```
import pandas as pd
df = pd.read_csv('census.csv')
df.head()
```

Out[14]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMATESE
0	40	3	6	1	0	Alabama	Alabama	4779736	4780127
1	50	3	6	1	1	Alabama	Autauga County	54571	54571
2	50	3	6	1	3	Alabama	Baldwin County	182265	182265
3	50	3	6	1	5	Alabama	Barbour County	27457	27457
4	50	3	6	1	7	Alabama	Bibb County	22915	22919

5 rows × 100 columns

In [15]:
```
(df.where(df['SUMLEV']==50)
    .dropna()
    .set_index(['STNAME','CTYNAME'])
    .rename(columns={'ESTIMATESBASE2010': 'Estimates Base 2010'}))
```

Out[15]:

		SUMLEV	REGION	DIVISION	STATE	COUNTY	CENSUS2010POP	Estimates Base 2010
STNAME	**CTYNAME**							
Alabama	**Autauga County**	50.0	3.0	6.0	1.0	1.0	54571.0	54571.0
	Baldwin County	50.0	3.0	6.0	1.0	3.0	182265.0	182265.0
	Barbour County	50.0	3.0	6.0	1.0	5.0	27457.0	27457.0
	Bibb County	50.0	3.0	6.0	1.0	7.0	22915.0	22919.0
	Blount County	50.0	3.0	6.0	1.0	9.0	57322.0	57322.0
	Bullock County	50.0	3.0	6.0	1.0	11.0	10914.0	10915.0
	Butler County	50.0	3.0	6.0	1.0	13.0	20947.0	20946.0
	Calhoun County	50.0	3.0	6.0	1.0	15.0	118572.0	118586.0
	Chambers County	50.0	3.0	6.0	1.0	17.0	34215.0	34170.0
	Cherokee County	50.0	3.0	6.0	1.0	19.0	25989.0	25986.0
	Chilton County	50.0	3.0	6.0	1.0	21.0	43643.0	43631.0
	Choctaw County	50.0	3.0	6.0	1.0	23.0	13859.0	13858.0
	Clarke County	50.0	3.0	6.0	1.0	25.0	25833.0	25840.0
	Clay County	50.0	3.0	6.0	1.0	27.0	13932.0	13932.0
	Cleburne County	50.0	3.0	6.0	1.0	29.0	14972.0	14972.0
	Coffee County	50.0	3.0	6.0	1.0	31.0	49948.0	49948.0
	Colbert County	50.0	3.0	6.0	1.0	33.0	54428.0	54428.0
	Conecuh County	50.0	3.0	6.0	1.0	35.0	13228.0	13228.0
	Coosa County	50.0	3.0	6.0	1.0	37.0	11539.0	11758.0
	Covington County	50.0	3.0	6.0	1.0	39.0	37765.0	37765.0
	Crenshaw County	50.0	3.0	6.0	1.0	41.0	13906.0	13906.0

In [16]:
```
df = df[df['SUMLEV']==50]
df.set_index(['STNAME','CTYNAME'], inplace=True)
df.rename(columns={'ESTIMATESBASE2010': 'Estimates Base 2010'})
```

Out[16]:

		SUMLEV	REGION	DIVISION	STATE	COUNTY	CENSUS2010POP	Estimates Base 2010
STNAME	**CTYNAME**							
Alabama	**Autauga County**	50	3	6	1	1	54571	54571
	Baldwin County	50	3	6	1	3	182265	182265
	Barbour County	50	3	6	1	5	27457	27457
	Bibb County	50	3	6	1	7	22915	22919
	Blount County	50	3	6	1	9	57322	57322
	Bullock County	50	3	6	1	11	10914	10915
	Butler County	50	3	6	1	13	20947	20946
	Calhoun County	50	3	6	1	15	118572	118586
	Chambers County	50	3	6	1	17	34215	34170
	Cherokee County	50	3	6	1	19	25989	25986
	Chilton County	50	3	6	1	21	43643	43631
	Choctaw County	50	3	6	1	23	13859	13858
	Clarke County	50	3	6	1	25	25833	25840
	Clay County	50	3	6	1	27	13932	13932
	Cleburne County	50	3	6	1	29	14972	14972
	Coffee County	50	3	6	1	31	49948	49948
	Colbert County	50	3	6	1	33	54428	54428
	Conecuh County	50	3	6	1	35	13228	13228
	Coosa County	50	3	6	1	37	11539	11758
	Covington County	50	3	6	1	39	37765	37765
	Crenshaw County	50	3	6	1	41	13906	13906

In [17]:
```
import numpy as np
def min_max(row):
    data = row[['POPESTIMATE2010',
                'POPESTIMATE2011',
                'POPESTIMATE2012',
                'POPESTIMATE2013',
                'POPESTIMATE2014',
                'POPESTIMATE2015']]
    #print(data)
    return pd.Series({'min': np.min(data), 'max': np.max(data)})
```

In [18]:
```
df.apply?
```

In [19]:
```
df.apply(min_max, axis=1)
```

Out[19]:

		max	min
STNAME	**CTYNAME**		
Alabama	**Autauga County**	55347.0	54660.0
	Baldwin County	203709.0	183193.0
	Barbour County	27341.0	26489.0
	Bibb County	22861.0	22512.0
	Blount County	57776.0	57373.0
	Bullock County	10887.0	10606.0
	Butler County	20944.0	20154.0
	Calhoun County	118437.0	115620.0
	Chambers County	34153.0	33993.0
	Cherokee County	26084.0	25859.0
	Chilton County	43943.0	43665.0
	Choctaw County	13841.0	13170.0
	Clarke County	25767.0	24675.0
	Clay County	13880.0	13456.0
	Cleburne County	15072.0	14921.0
	Coffee County	51211.0	50177.0
	Colbert County	54514.0	54354.0
	Conecuh County	13208.0	12662.0
	Coosa County	11758.0	10724.0
	Covington County	38060.0	37796.0
	Crenshaw County	13963.0	13853.0
	Cullman County	82005.0	80374.0
	Dale County	50358.0	49501.0
	Dallas County	43803.0	41131.0
	DeKalb County	71387.0	70869.0
	Elmore County	81468.0	79465.0
	Escambia County	38309.0	37784.0
	Etowah County	104442.0	103057.0
	Fayette County	17231.0	16759.0
	Franklin County	31734.0	31507.0
...	**...**	...	...
Wisconsin	**Washburn County**	15930.0	15552.0
	Washington County	133674.0	131967.0
	Waukesha County	396488.0	390076.0
	Waupaca County	52422.0	51945.0
	Waushara County	24581.0	24033.0

In [20]:
```
import numpy as np
def min_max(row):
    data = row[['POPESTIMATE2010',
                'POPESTIMATE2011',
                'POPESTIMATE2012',
                'POPESTIMATE2013',
                'POPESTIMATE2014',
                'POPESTIMATE2015']]
    row['max'] = np.max(data)
    row['min'] = np.min(data)
    return row
df.apply(min_max, axis=1)
```

Out[20]:

		SUMLEV	REGION	DIVISION	STATE	COUNTY	CENSUS2010POP	ESTIMATES
STNAME	**CTYNAME**							
Alabama	**Autauga County**	50.0	3.0	6.0	1.0	1.0	54571.0	54571.0
	Baldwin County	50.0	3.0	6.0	1.0	3.0	182265.0	182265.0
	Barbour County	50.0	3.0	6.0	1.0	5.0	27457.0	27457.0
	Bibb County	50.0	3.0	6.0	1.0	7.0	22915.0	22919.0
	Blount County	50.0	3.0	6.0	1.0	9.0	57322.0	57322.0
	Bullock County	50.0	3.0	6.0	1.0	11.0	10914.0	10915.0
	Butler County	50.0	3.0	6.0	1.0	13.0	20947.0	20946.0
	Calhoun County	50.0	3.0	6.0	1.0	15.0	118572.0	118586.0
	Chambers County	50.0	3.0	6.0	1.0	17.0	34215.0	34170.0
	Cherokee County	50.0	3.0	6.0	1.0	19.0	25989.0	25986.0
	Chilton County	50.0	3.0	6.0	1.0	21.0	43643.0	43631.0
	Choctaw County	50.0	3.0	6.0	1.0	23.0	13859.0	13858.0
	Clarke County	50.0	3.0	6.0	1.0	25.0	25833.0	25840.0
	Clay County	50.0	3.0	6.0	1.0	27.0	13932.0	13932.0
	Cleburne County	50.0	3.0	6.0	1.0	29.0	14972.0	14972.0
	Coffee County	50.0	3.0	6.0	1.0	31.0	49948.0	49948.0
	Colbert County	50.0	3.0	6.0	1.0	33.0	54428.0	54428.0
	Conecuh County	50.0	3.0	6.0	1.0	35.0	13228.0	13228.0
	Coosa County	50.0	3.0	6.0	1.0	37.0	11539.0	11758.0
	Covington County	50.0	3.0	6.0	1.0	39.0	37765.0	37765.0
	Crenshaw County	50.0	3.0	6.0	1.0	41.0	13906.0	13906.0
	Cullman	50.0	3.0	6.0	1.0	43.0	80406.0	80410.0

In [21]:
```
rows = ['POPESTIMATE2010',
        'POPESTIMATE2011',
        'POPESTIMATE2012',
        'POPESTIMATE2013',
        'POPESTIMATE2014',
        'POPESTIMATE2015']
df.apply(lambda x: np.max(x[rows]), axis=1)
```

Out[21]:

STNAME	CTYNAME	
Alabama	Autauga County	55347.0
	Baldwin County	203709.0
	Barbour County	27341.0
	Bibb County	22861.0
	Blount County	57776.0
	Bullock County	10887.0
	Butler County	20944.0
	Calhoun County	118437.0
	Chambers County	34153.0
	Cherokee County	26084.0
	Chilton County	43943.0
	Choctaw County	13841.0
	Clarke County	25767.0
	Clay County	13880.0
	Cleburne County	15072.0
	Coffee County	51211.0
	Colbert County	54514.0
	Conecuh County	13208.0
	Coosa County	11758.0
	Covington County	38060.0
	Crenshaw County	13963.0
	Cullman County	82005.0
	Dale County	50358.0
	Dallas County	43803.0
	DeKalb County	71387.0
	Elmore County	81468.0
	Escambia County	38309.0
	Etowah County	104442.0
	Fayette County	17231.0
	Franklin County	31734.0
		...
Wisconsin	Washburn County	15930.0
	Washington County	133674.0
	Waukesha County	396488.0
	Waupaca County	52422.0
	Waushara County	24581.0
	Winnebago County	169639.0
	Wood County	74807.0
Wyoming	Albany County	37956.0
	Big Horn County	12022.0
	Campbell County	49220.0
	Carbon County	15856.0
	Converse County	14343.0
	Crook County	7444.0
	Fremont County	41129.0
	Goshen County	13666.0
	Hot Springs County	4846.0
	Johnson County	8636.0
	Laramie County	97121.0
	Lincoln County	18722.0
	Natrona County	82178.0
	Niobrara County	2548.0
	Park County	29237.0
	Platte County	8812.0
	Sheridan County	30020.0
	Sublette County	10418.0
	Sweetwater County	45162.0
	Teton County	23125.0
	Uinta County	21102.0
	Washakie County	8545.0
	Weston County	7234.0

dtype: float64

Group by

In [22]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('census.csv')
df = df[df['SUMLEV']==50]
df
```

Out[22]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIM
1	50	3	6	1	1	Alabama	Autauga County	54571	54571
2	50	3	6	1	3	Alabama	Baldwin County	182265	182265
3	50	3	6	1	5	Alabama	Barbour County	27457	27457
4	50	3	6	1	7	Alabama	Bibb County	22915	22919
5	50	3	6	1	9	Alabama	Blount County	57322	57322
6	50	3	6	1	11	Alabama	Bullock County	10914	10915
7	50	3	6	1	13	Alabama	Butler County	20947	20946
8	50	3	6	1	15	Alabama	Calhoun County	118572	118586
9	50	3	6	1	17	Alabama	Chambers County	34215	34170
10	50	3	6	1	19	Alabama	Cherokee County	25989	25986
11	50	3	6	1	21	Alabama	Chilton County	43643	43631
12	50	3	6	1	23	Alabama	Choctaw County	13859	13858
13	50	3	6	1	25	Alabama	Clarke County	25833	25840
14	50	3	6	1	27	Alabama	Clay County	13932	13932
15	50	3	6	1	29	Alabama	Cleburne County	14972	14972
16	50	3	6	1	31	Alabama	Coffee County	49948	49948
17	50	3	6	1	33	Alabama	Colbert County	54428	54428
18	50	3	6	1	35	Alabama	Conecuh County	13228	13228
19	50	3	6	1	37	Alabama	Coosa County	11539	11758
20	50	3	6	1	39	Alabama	Covington County	37765	37765
21	50	3	6	1	41	Alabama	Crenshaw County	13906	13906
22	50	3	6	1	43	Alabama	Cullman County	80406	80410

In [23]:
```
'''%%timeit -n 10
for state in df['STNAME'].unique():
    avg = np.average(df.where(df['STNAME']==state).dropna()['CENSUS2010POP'])
    print('Counties in state ' + state + ' have an average population of ' + str
(avg))'''
```

Out[23]:
```
"%%timeit -n 10\nfor state in df['STNAME'].unique():\n    avg = np.average(df.
where(df['STNAME']==state).dropna()['CENSUS2010POP'])\n    print('Counties in
state ' + state + ' have an average population of ' + str(avg))"
```

In [24]:
```
#faster
'''
%%timeit -n 10
for group, frame in df.groupby('STNAME'):
    #print(group)
    avg = np.average(frame['CENSUS2010POP'])
    print('Counties in state ' + group + ' have an average population of ' + str
(avg))'''
```

Out[24]:
```
"\n%%timeit -n 10\nfor group, frame in df.groupby('STNAME'):\n    #print(grou
p)\n    avg = np.average(frame['CENSUS2010POP'])\n    print('Counties in state
' + group + ' have an average population of ' + str(avg))"
```

In [25]:
```
df.head()
```

Out[25]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMATESE
1	50	3	6	1	1	Alabama	Autauga County	54571	54571
2	50	3	6	1	3	Alabama	Baldwin County	182265	182265
3	50	3	6	1	5	Alabama	Barbour County	27457	27457
4	50	3	6	1	7	Alabama	Bibb County	22915	22919
5	50	3	6	1	9	Alabama	Blount County	57322	57322

5 rows × 100 columns

In [26]:
```
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.
```

In [27]:
```
df = pd.read_csv('census.csv')
df = df[df['SUMLEV']==50]
```

In [28]:
```
df.groupby('STNAME').agg({'CENSUS2010POP': np.average})
```

Out[28]:

	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

In [29]:
```
print(type(df.groupby(level=0)['POPESTIMATE2010','POPESTIMATE2011']))
print(type(df.groupby(level=0)['POPESTIMATE2010']))
```

```
<class 'pandas.core.groupby.DataFrameGroupBy'>
<class 'pandas.core.groupby.SeriesGroupBy'>
```

In [30]:
```
#df.groupby?
df.groupby(level=0)['POPESTIMATE2010','POPESTIMATE2011']
```

Out[30]:
```
<pandas.core.groupby.DataFrameGroupBy object at 0x7f52f23e34e0>
```

In [31]:
```
(df.set_index('STNAME').groupby(level=0)['CENSUS2010POP'].agg({'avg': np.averag
e, 'sum': np.sum}))
```

Out[31]:

	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

In [32]:
```
(df.set_index('STNAME').groupby(level=0)['POPESTIMATE2010','POPESTIMATE2011']
    .agg({'avg': np.average, 'sum': np.sum}))
```

Out[32]:

	avg		sum	
	POPESTIMATE2010	POPESTIMATE2011	POPESTIMATE2010	POPESTIMATE2011
STNAME				
Alabama	71420.313433	71658.328358	4785161	4801108
Alaska	24621.413793	24921.379310	714021	722720
Arizona	427213.866667	431248.800000	6408208	6468732
Arkansas	38965.253333	39180.506667	2922394	2938538
California	643691.017241	650000.586207	37334079	37700034
Colorado	78878.968750	79991.875000	5048254	5119480
Connecticut	447464.625000	448719.875000	3579717	3589759
Delaware	299930.333333	302638.666667	899791	907916
District of Columbia	605126.000000	620472.000000	605126	620472
Florida	281341.641791	285157.208955	18849890	19105533
Georgia	61090.905660	61712.452830	9713454	9812280
Hawaii	272796.000000	275645.400000	1363980	1378227
Idaho	35704.227273	36003.045455	1570986	1584134
Illinois	125894.598039	126096.882353	12841249	12861882
Indiana	70549.891304	70835.271739	6490590	6516845
Iowa	30815.090909	30963.525253	3050694	3065389
Kansas	27226.895238	27332.542857	2858824	2869917
Kentucky	36232.808333	36399.016667	4347937	4367882
Louisiana	71014.859375	71490.328125	4544951	4575381
Maine	82980.937500	83016.062500	1327695	1328257
Maryland	241183.708333	243507.125000	5788409	5844171
Massachusetts	468931.142857	472271.214286	6565036	6611797
Michigan	119004.445783	118995.048193	9877369	9876589
Minnesota	61044.862069	61472.632184	5310903	5348119
Mississippi	36223.365854	36317.060976	2970316	2977999
Missouri	52139.582609	52265.973913	5996052	6010587
Montana	17690.053571	17816.892857	990643	997746
Nebraska	19677.688172	19810.569892	1830025	1842383
Nevada	159025.882353	159930.529412	2703440	2718819
New Hampshire	131670.800000	131834.400000	1316708	1318344
New Jersey	419232.428571	421092.095238	8803881	8842934
New Mexico	62567.909091	62976.545455	2064741	2078226
New York	312950.322581	314890.354839	19402920	19523202
North Carolina	95589.790000	96510.250000	9558979	9651025

In [33]:
```
(df.set_index('STNAME').groupby(level=0)['POPESTIMATE2010','POPESTIMATE2011']
    .agg({'POPESTIMATE2010': np.average, 'POPESTIMATE2011': np.sum}))
```

Out[33]:

	POPESTIMATE2010	POPESTIMATE2011
STNAME		
Alabama	71420.313433	4801108
Alaska	24621.413793	722720
Arizona	427213.866667	6468732
Arkansas	38965.253333	2938538
California	643691.017241	37700034
Colorado	78878.968750	5119480
Connecticut	447464.625000	3589759
Delaware	299930.333333	907916
District of Columbia	605126.000000	620472
Florida	281341.641791	19105533
Georgia	61090.905660	9812280
Hawaii	272796.000000	1378227
Idaho	35704.227273	1584134
Illinois	125894.598039	12861882
Indiana	70549.891304	6516845
Iowa	30815.090909	3065389
Kansas	27226.895238	2869917
Kentucky	36232.808333	4367882
Louisiana	71014.859375	4575381
Maine	82980.937500	1328257
Maryland	241183.708333	5844171
Massachusetts	468931.142857	6611797
Michigan	119004.445783	9876589
Minnesota	61044.862069	5348119
Mississippi	36223.365854	2977999
Missouri	52139.582609	6010587
Montana	17690.053571	997746
Nebraska	19677.688172	1842383
Nevada	159025.882353	2718819
New Hampshire	131670.800000	1318344
New Jersey	419232.428571	8842934
New Mexico	62567.909091	2078226
New York	312950.322581	19523202
North Carolina	95589.790000	9651025
North Dakota	12726.981132	685326
Ohio	131145.068182	11545442

Scales

In [34]:
```
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)
df
```

Out[34]:

	Grades
excellent	A+
excellent	A
excellent	A-
good	B+
good	B
good	B-
ok	C+
ok	C
ok	C-
poor	D+
poor	D

In [35]:
```
df['Grades'].astype('category')
```

Out[35]:
```
excellent    A+
excellent     A
excellent    A-
good         B+
good          B
good         B-
ok           C+
ok            C
ok           C-
poor         D+
poor          D
Name: Grades, dtype: category
Categories (11, object): [A, A+, A-, B, ..., C+, C-, D, D+]
```

In [36]:
```
grades = df['Grades'].astype('category',
                             categories=['D', 'D+', 'C-', 'C', 'C+', 'B-', 'B',
'B+', 'A-', 'A', 'A+'],
                             ordered=True)
grades
```

Out[36]:
```
excellent    A+
excellent     A
excellent    A-
good         B+
good          B
good         B-
ok           C+
ok            C
ok           C-
poor         D+
poor          D
Name: Grades, dtype: category
Categories (11, object): [D < D+ < C- < C ... B+ < A- < A < A+]
```

In [37]:
```
grades > 'C'
```

Out[37]:
```
excellent     True
excellent     True
excellent     True
good          True
good          True
good          True
ok            True
ok           False
ok           False
poor         False
poor         False
Name: Grades, dtype: bool
```

In [38]:
```
import pandas as pd
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.ave
rage})
pd.cut(df['avg'],10)
```

```
Out[38]: STNAME
         Alabama                  (11706.0871, 75333.413]
         Alaska                   (11706.0871, 75333.413]
         Arizona                 (390320.176, 453317.529]
         Arkansas                 (11706.0871, 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                  (11706.0871, 75333.413]
         Hawaii                  (264325.471, 327322.823]
         Idaho                    (11706.0871, 75333.413]
         Illinois                 (75333.413, 138330.766]
         Indiana                  (11706.0871, 75333.413]
         Iowa                     (11706.0871, 75333.413]
         Kansas                   (11706.0871, 75333.413]
         Kentucky                 (11706.0871, 75333.413]
         Louisiana                (11706.0871, 75333.413]
         Maine                    (75333.413, 138330.766]
         Maryland                (201328.118, 264325.471]
         Massachusetts           (453317.529, 516314.881]
         Michigan                 (75333.413, 138330.766]
         Minnesota                (11706.0871, 75333.413]
         Mississippi              (11706.0871, 75333.413]
         Missouri                 (11706.0871, 75333.413]
         Montana                  (11706.0871, 75333.413]
         Nebraska                 (11706.0871, 75333.413]
         Nevada                  (138330.766, 201328.118]
         New Hampshire            (75333.413, 138330.766]
         New Jersey              (390320.176, 453317.529]
         New Mexico               (11706.0871, 75333.413]
         New York                (264325.471, 327322.823]
         North Carolina           (75333.413, 138330.766]
         North Dakota             (11706.0871, 75333.413]
         Ohio                     (75333.413, 138330.766]
         Oklahoma                 (11706.0871, 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             (11706.0871, 75333.413]
         Tennessee                (11706.0871, 75333.413]
         Texas                    (75333.413, 138330.766]
         Utah                     (75333.413, 138330.766]
         Vermont                  (11706.0871, 75333.413]
         Virginia                 (11706.0871, 75333.413]
         Washington              (138330.766, 201328.118]
         West Virginia            (11706.0871, 75333.413]
         Wisconsin                (75333.413, 138330.766]
         Wyoming                  (11706.0871, 75333.413]
         Name: avg, dtype: category
         Categories (10, object): [(11706.0871, 75333.413] < (75333.413, 138330.766] <
         (138330.766, 201328.118] < (201328.118, 264325.471] ... (390320.176, 453317.52
         9] < (453317.529, 516314.881] < (516314.881, 579312.234] < (579312.234, 64230
         9.586]]
```

Pivot Tables

In [39]:
```
#http://open.canada.ca/data/en/dataset/98f1a129-f628-4ce4-b24d-6f16bf24dd64
df = pd.read_csv('cars.csv')
```

In [40]:
```
df.head()
```

Out[40]:

	YEAR	Make	Model	Size	(kW)	Unnamed: 5	TYPE	CITY (kWh/100 km)	HWY (kWh/100 km)	COMB (kWh/1 km)
0	2012	MITSUBISHI	i-MiEV	SUBCOMPACT	49	A1	B	16.9	21.4	18.7
1	2012	NISSAN	LEAF	MID-SIZE	80	A1	B	19.3	23.0	21.1
2	2013	FORD	FOCUS ELECTRIC	COMPACT	107	A1	B	19.0	21.1	20.0
3	2013	MITSUBISHI	i-MiEV	SUBCOMPACT	49	A1	B	16.9	21.4	18.7
4	2013	NISSAN	LEAF	MID-SIZE	80	A1	B	19.3	23.0	21.1

In [41]:
```
df.pivot_table(values='(kW)', index='YEAR', columns='Make', aggfunc=np.mean)
```

Out[41]:

Make	BMW	CHEVROLET	FORD	KIA	MITSUBISHI	NISSAN	SMART	TESLA
YEAR								
2012	NaN	NaN	NaN	NaN	49.0	80.0	NaN	NaN
2013	NaN	NaN	107.0	NaN	49.0	80.0	35.0	280.000000
2014	NaN	104.0	107.0	NaN	49.0	80.0	35.0	268.333333
2015	125.0	104.0	107.0	81.0	49.0	80.0	35.0	320.666667
2016	125.0	104.0	107.0	81.0	49.0	80.0	35.0	409.700000

In [42]:
```
df.pivot_table(values='(kW)', index='YEAR', columns='Make', aggfunc=[np.mean,np.min], margins=True)
```

Out[42]:

	mean									amin
Make	**BMW**	**CHEVROLET**	**FORD**	**KIA**	**MITSUBISHI**	**NISSAN**	**SMART**	**TESLA**	**All**	**BMW**
YEAR										
2012	NaN	NaN	NaN	NaN	49.0	80.0	NaN	NaN	64.500000	NaN
2013	NaN	NaN	107.0	NaN	49.0	80.0	35.0	280.000000	158.444444	NaN
2014	NaN	104.0	107.0	NaN	49.0	80.0	35.0	268.333333	135.000000	NaN
2015	125.0	104.0	107.0	81.0	49.0	80.0	35.0	320.666667	181.428571	125.0
2016	125.0	104.0	107.0	81.0	49.0	80.0	35.0	409.700000	252.263158	125.0
All	125.0	104.0	107.0	81.0	49.0	80.0	35.0	345.478261	190.622642	125.0

Date Functionality in Pandas

In [43]:
```
import pandas as pd
import numpy as np
```

Timestamp

In [44]:
```
pd.Timestamp('9/1/2016 10:05AM')
```

Out[44]:
```
Timestamp('2016-09-01 10:05:00')
```

Period

In [45]:
```
pd.Period('1/2016')
```

Out[45]:
```
Period('2016-01', 'M')
```

In [46]:
```
pd.Period('3/5/2016')
```

Out[46]:
```
Period('2016-03-05', 'D')
```

DatetimeIndex

In [47]:
```
t1 = pd.Series(list('abc'), [pd.Timestamp('2016-09-01'), pd.Timestamp('2016-09-02'), pd.Timestamp('2016-09-03')])
t1
```

Out[47]:
```
2016-09-01    a
2016-09-02    b
2016-09-03    c
dtype: object
```

In [48]:
```
type(t1.index)
```

Out[48]:
```
pandas.tseries.index.DatetimeIndex
```

PeriodIndex

In [49]:
```
t2 = pd.Series(list('def'), [pd.Period('2016-09'), pd.Period('2016-10'), pd.Period('2016-11')])
t2
```

Out[49]:
```
2016-09    d
2016-10    e
2016-11    f
Freq: M, dtype: object
```

In [50]:
```
type(t2.index)
```

Out[50]:
```
pandas.tseries.period.PeriodIndex
```

Converting to Datetime

In [51]:
```
d1 = ['2 June 2013', 'Aug 29, 2014', '2015-06-26', '7/12/16']
ts3 = pd.DataFrame(np.random.randint(10, 100, (4,2)), index=d1, columns=list('ab
'))
ts3
```

Out[51]:

	a	b
2 June 2013	52	28
Aug 29, 2014	56	16
2015-06-26	26	12
7/12/16	69	89

In [52]:
```
ts3.index = pd.to_datetime(ts3.index)
ts3
```

Out[52]:

	a	b
2013-06-02	52	28
2014-08-29	56	16
2015-06-26	26	12
2016-07-12	69	89

In [53]:
```
pd.to_datetime('4.7.12', dayfirst=True)
```

Out[53]:
```
Timestamp('2012-07-04 00:00:00')
```

Timedeltas

In [54]:
```
pd.Timestamp('9/3/2016')-pd.Timestamp('9/1/2016')
```

Out[54]:
```
Timedelta('2 days 00:00:00')
```

In [55]:
```
pd.Timestamp('9/2/2016 8:10AM') + pd.Timedelta('12D 3H')
```

Out[55]:
```
Timestamp('2016-09-14 11:10:00')
```

Working with Dates in a Dataframe

In [56]:
```
dates = pd.date_range('10-01-2016', periods=9, freq='2W-SUN')
dates
```

Out[56]:
```
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')
```

In [57]:
```
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[57]:

	Count 1	Count 2
2016-10-02	108	129
2016-10-16	108	121
2016-10-30	111	115
2016-11-13	112	117
2016-11-27	109	122
2016-12-11	110	125
2016-12-25	108	122
2017-01-08	117	124
2017-01-22	123	123

In [58]:
```
df.index.weekday_name
```

Out[58]:
```
array(['Sunday', 'Sunday', 'Sunday', 'Sunday', 'Sunday', 'Sunday',
       'Sunday', 'Sunday', 'Sunday'], dtype=object)
```

In [59]:
```
df.diff()
```

Out[59]:

	Count 1	Count 2
2016-10-02	NaN	NaN
2016-10-16	0.0	-8.0
2016-10-30	3.0	-6.0
2016-11-13	1.0	2.0
2016-11-27	-3.0	5.0
2016-12-11	1.0	3.0
2016-12-25	-2.0	-3.0
2017-01-08	9.0	2.0
2017-01-22	6.0	-1.0

In [60]:
```
df.resample('M').mean()
```

Out[60]:

	Count 1	Count 2
2016-10-31	109.0	121.666667
2016-11-30	110.5	119.500000
2016-12-31	109.0	123.500000
2017-01-31	120.0	123.500000

In [61]:
```
df['2017']
```

Out[61]:

	Count 1	Count 2
2017-01-08	117	124
2017-01-22	123	123

In [62]:
```
df['2016-12']
```

Out[62]:

	Count 1	Count 2
2016-12-11	110	125
2016-12-25	108	122

In [63]:
```
df['2016-12':]
```

Out[63]:

	Count 1	Count 2
2016-12-11	110	125
2016-12-25	108	122
2017-01-08	117	124
2017-01-22	123	123

In [64]:
```
df.asfreq('W', method='ffill')
```

Out[64]:

	Count 1	Count 2
2016-10-02	108	129
2016-10-09	108	129
2016-10-16	108	121
2016-10-23	108	121
2016-10-30	111	115
2016-11-06	111	115
2016-11-13	112	117
2016-11-20	112	117
2016-11-27	109	122
2016-12-04	109	122
2016-12-11	110	125
2016-12-18	110	125
2016-12-25	108	122
2017-01-01	108	122
2017-01-08	117	124
2017-01-15	117	124
2017-01-22	123	123

In [65]:
```
import matplotlib.pyplot as plt
%matplotlib inline

df.plot()
```

Out[65]: <matplotlib.axes._subplots.AxesSubplot at 0x7f52f03ac240>