*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)](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]:

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

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

Out[2]:

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

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

Out[3]:

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

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

Out[4]:

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

In [5]:
```
adf = df.reset_index()
adf['Date'] = pd.Series({0: 'December 1', 2: 'mid-May'})
adf
```

Out[5]:

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

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
```

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

Out[7]:

Name	Role	School
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]:

Name	Role	School
Sally	Course liasion	Engineering
James	Grader	Business

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

Out[9]:

Name	Role	School
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]:

Name	Role	School
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': 'Washington Avenue'}])
student_df = pd.DataFrame([{'Name': 'James', 'School': 'Business', 'Location': '1024 Billiard Avenue'},
                           {'Name': 'Mike', 'School': 'Law', 'Location': 'Fraternity House #22'},
                           {'Name': 'Sally', 'School': 'Engineering', 'Location': '512 Wilson Crescent'}])
pd.merge(staff_df, student_df, how='left', left_on='Name', right_on='Name')
```

Out[12]:

	Name	Role	Location_x	School	Location_y
0	Kelly	Director of HR	State Street	NaN	NaN
1	Sally	Course liasion	Washington Avenue	Engineering	512 Wilson Crescent
2	James	Grader	Washington Avenue	Business	1024 Billiard Avenue

In [13]:
```
staff_df = pd.DataFrame([{'First Name': 'Kelly', 'Last Name': 'Desjardins', 'Role': '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', 'School': 'Business'},
                           {'First Name': 'Mike', 'Last Name': 'Smith', 'School': 'Law'},
                           {'First Name': 'Sally', 'Last Name': 'Brooks', 'School': 'Engineering'}])
```

In [14]:
```
staff_df
```

Out[14]:

	First Name	Last Name	Role
0	Kelly	Desjardins	Director of HR
1	Sally	Brooks	Course liasion
2	James	Wilde	Grader

In [15]:
```
student_df
```

Out[15]:

	First Name	Last Name	School
0	James	Hammond	Business
1	Mike	Smith	Law
2	Sally	Brooks	Engineering

In [16]:
```
pd.merge(staff_df, student_df, how='inner', left_on=['First Name','Last Nam
e'], right_on=['First Name','Last Name'])
```

Out[16]:

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

Idiomatic Pandas: Making Code Pandorable

In [17]:
```
import pandas as pd
df = pd.read_csv('census.csv')
df.head()
```

Out[17]:

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

5 rows × 100 columns

In [18]:
```
# 仅仅提取 SUMLEV = 50 的行
# drop NaN
# 将 index 设为 'STNAME', 'CTYNAME'
# 将 'ESTIMATESBASE2010'列重新命名为 'Estimates Base 2010'
(df.where(df['SUMLEV']==50)
    .dropna()
    .set_index(['STNAME','CTYNAME'])
    .rename(columns={'ESTIMATESBASE2010': 'Estimates Base 2010'})).head()
```

Out[18]:

		SUMLEV	REGION	DIVISION	STATE	COUNTY	CENSUS2010POP	Estimates Base 2010	P
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	

5 rows × 98 columns

In [19]:
```
# 用不同语法实现上面的结果
df = df[df['SUMLEV']==50]
df.set_index(['STNAME','CTYNAME'], inplace=True)
df.rename(columns={'ESTIMATESBASE2010': 'Estimates Base 2010'}).head()
```

Out[19]:

		SUMLEV	REGION	DIVISION	STATE	COUNTY	CENSUS2010POP	Estimates Base 2010	P
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	

5 rows × 98 columns

In [20]:
```
import numpy as np
def min_max(row):
    data = row[['POPESTIMATE2010',
                'POPESTIMATE2011',
                'POPESTIMATE2012',
                'POPESTIMATE2013',
                'POPESTIMATE2014',
                'POPESTIMATE2015']]
    # np.min(data) 对行进行最小值比较
    return pd.Series({'min': np.min(data), 'max': np.max(data)})
```

In [21]:
```
df.apply(min_max, axis=1).head()
```

Out[21]:

		min	max
STNAME	CTYNAME		
Alabama	Autauga County	54660.0	55347.0
	Baldwin County	183193.0	203709.0
	Barbour County	26489.0	27341.0
	Bibb County	22512.0	22861.0
	Blount County	57373.0	57776.0

In [22]:
```
import numpy as np
def min_max(row):
    data = row[['POPESTIMATE2010',
                'POPESTIMATE2011',
                'POPESTIMATE2012',
                'POPESTIMATE2013',
                'POPESTIMATE2014',
                'POPESTIMATE2015']]
    # 将上面列所产生的最大最小值，变为新添列，加入原 DataFrame 中
    row['max'] = np.max(data)
    row['min'] = np.min(data)
    return row
df.apply(min_max, axis=1).head()
```

Out[22]:

		SUMLEV	REGION	DIVISION	STATE	COUNTY	CENSUS2010POP	ESTIMATESBA
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	1
	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	

5 rows × 100 columns

In [23]:
```
rows = ['POPESTIMATE2010',
        'POPESTIMATE2011',
        'POPESTIMATE2012',
        'POPESTIMATE2013',
        'POPESTIMATE2014',
        'POPESTIMATE2015']
df.apply(lambda x: np.max(x[rows]), axis=1)
```

Out[23]:
```
STNAME    CTYNAME
Alabama   Autauga County        55347.0
          Baldwin County       203709.0
          Barbour County        27341.0
          Bibb County           22861.0
          Blount County         57776.0
                                  ...
Wyoming   Sweetwater County      45162.0
          Teton County          23125.0
          Uinta County          21102.0
          Washakie County        8545.0
          Weston County          7234.0
Length: 3142, dtype: float64
```

Group by

In [24]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('census.csv')
df = df[df['SUMLEV']==50]
df.head()
```

Out[24]:

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

5 rows × 100 columns

In [25]:
```
#%%timeit -n 10
#for state in df['STNAME'].unique():
#    avg = np.average(df.where(df['STNAME']==state).dropna()['CENSUS2010PO
P'])
    # print('Counties in state ' + state + ' have an average population of
 ' + str(avg))
```

In [26]:
```
#%%timeit -n 10
#for group, frame in df.groupby('STNAME'):
#    avg = np.average(frame['CENSUS2010POP'])
#    # print('Counties in state ' + group + ' have an average population of
' + str(avg))
```

In [27]:
```
df = df.set_index('STNAME')
df.head()
```

Out[27]:

STNAME	SUMLEV	REGION	DIVISION	STATE	COUNTY	CTYNAME	CENSUS2010POP	ESTIMATESBA
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 [28]:
```
# if the 1st letter of the index is less than a capital M we will return a
 0
# if it's less than a capital Q, then return 1
# Otherwise, return 2
def fun(item):
    if item[0]<'M':
        return 0
    if item[0]<'Q':
        return 1
    return 2
```

In [29]:
```
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.
```

Split, Apply, Combine Pattern

In [30]:
```
df = pd.read_csv('census.csv')
df = df[df['SUMLEV']==50]
```

`groupby` object has a method called `agg`.

With `agg` function, we simply pass a column name that we are interested in, and the function we want to apply.

或者

将需要操作的列放在 `agg` 前也一样, 但是返回的列名不同。

```
In [31]: df.groupby('STNAME').agg({'CENSUS2010POP': np.average}).head()
```

Out[31]:

STNAME	CENSUS2010POP
Alabama	71339.343284
Alaska	24490.724138
Arizona	426134.466667
Arkansas	38878.906667
California	642309.586207

```
In [32]: df.groupby('STNAME')['CENSUS2010POP'].agg({np.average}).head()
```

Out[32]:

STNAME	average
Alabama	71339.343284
Alaska	24490.724138
Arizona	426134.466667
Arkansas	38878.906667
California	642309.586207

level: If the axis is a MultiIndex (hierarchical), group by a particular level or levels.
所以 level = 0 就是按照 index 来 `groupby`

```
In [33]: print(type(df.groupby(level=0)['POPESTIMATE2010','POPESTIMATE2011']))
         print(type(df.groupby(level=0)['POPESTIMATE2010']))
```

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

下两个表达方式的结果相同：使用 level = 0 和 index

In [34]:
```
(df.set_index('STNAME').groupby(level=0)['CENSUS2010POP']
    .agg({'avg': np.average, 'sum': np.sum})).head()
```

```
C:\Users\XZV838\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykern
el_launcher.py:2: FutureWarning: using a dict on a Series for aggregation
is deprecated and will be removed in a future version. Use
named aggregation instead.

    >>> grouper.agg(name_1=func_1, name_2=func_2)
```

Out[34]:

STNAME	avg	sum
Alabama	71339.343284	4779736
Alaska	24490.724138	710231
Arizona	426134.466667	6392017
Arkansas	38878.906667	2915918
California	642309.586207	37253956

In [35]:
```
(df.groupby('STNAME')['CENSUS2010POP']
    .agg({'avg': np.average, 'sum': np.sum})).head()
```

```
C:\Users\XZV838\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykern
el_launcher.py:2: FutureWarning: using a dict on a Series for aggregation
is deprecated and will be removed in a future version. Use
named aggregation instead.

    >>> grouper.agg(name_1=func_1, name_2=func_2)
```

Out[35]:

STNAME	avg	sum
Alabama	71339.343284	4779736
Alaska	24490.724138	710231
Arizona	426134.466667	6392017
Arkansas	38878.906667	2915918
California	642309.586207	37253956

In [36]:
```
(df.set_index('STNAME').groupby(level=0)['POPESTIMATE2010','POPESTIMATE2011']
    .agg({'avg': np.average, 'sum': np.sum})).head()
```

```
C:\Users\XZV838\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas
\core\groupby\generic.py:1455: FutureWarning: using a dict with renaming is
deprecated and will be removed
in a future version.

For column-specific groupby renaming, use named aggregation

    >>> df.groupby(...).agg(name=('column', aggfunc))

  return super().aggregate(arg, *args, **kwargs)
```

Out[36]:

	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

In [37]:
```
(df.groupby('STNAME')['POPESTIMATE2010','POPESTIMATE2011']
    .agg({'avg': np.average, 'sum': np.sum})).head()
```

Out[37]:

	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

In [38]:
```
(df.set_index('STNAME').groupby(level=0)['POPESTIMATE2010','POPESTIMATE2011']
    .agg({'POPESTIMATE2010': np.average, 'POPESTIMATE2011': np.sum})).head()
```

Out[38]:

STNAME	POPESTIMATE2010	POPESTIMATE2011
Alabama	71420.313433	4801108
Alaska	24621.413793	722720
Arizona	427213.866667	6468732
Arkansas	38965.253333	2938538
California	643691.017241	37700034

Scales

In [39]:
```
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)
df
```

Out[39]:

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

In [40]:
```
grades = df['Grades'].astype('category')
grades
```

Out[40]:
```
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 [41]:
```
# 让 categories 有大小之分
from pandas.api.types import CategoricalDtype
grades = df['Grades'].astype(CategoricalDtype(
                                categories=['D', 'D+', 'C-', 'C', 'C+', 'B-',
'B', 'B+', 'A-', 'A', 'A+'],ordered=True))
grades
```

Out[41]:
```
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 [42]:
```
grades > 'C'
```

Out[42]:
```
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 [43]:
```
df = pd.read_csv('census.csv')
df = df[df['SUMLEV']==50]
df = df.set_index('STNAME').groupby(level=0)['CENSUS2010POP'].agg({'avg': n
p.average})
pd.cut(df['avg'],10).head()
```

```
C:\Users\XZV838\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykern
el_launcher.py:3: FutureWarning: using a dict on a Series for aggregation
is deprecated and will be removed in a future version. Use
named aggregation instead.

    >>> grouper.agg(name_1=func_1, name_2=func_2)

  This is separate from the ipykernel package so we can avoid doing imports
until
```

Out[43]:
```
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]
Name: avg, dtype: category
Categories (10, interval[float64]): [(11706.087, 75333.413] < (75333.413, 1
38330.766] < (138330.766, 201328.118] < (201328.118, 264325.471] ... (39032
0.176, 453317.529] < (453317.529, 516314.881] < (516314.881, 579312.234] <
(579312.234, 642309.586]]
```

Pivot Tables

`pivot_table` 类似 `groupby`，两者可以完全等价。

- pd.pivot_table(df, index=[字段1], values=[字段2], aggfunc=[函数], fill_value=0)
- df.groupby([字段1])[字段2].agg(函数).fillna(0)

但是 `pivot_table` 加入了 columns 与 margin 功能的 `groupby`，比 `groupby` 更加灵活。

`pivot_table` 有四个最重要的参数 index, values, columns, aggfunc

- index: 可以拥有多层 index。df.pivot_table(index = ['a', 'b'])
- values: 对需要的计算数据进行筛选, 哪些列的数据需要参与计算
- columns: 表格生成后的列名 (`groupby` 无法做到)
- aggfunc: 设置我们对数据聚合时进行的函数操作。

In [44]:
```
#http://open.canada.ca/data/en/dataset/98f1a129-f628-4ce4-b24d-6f16bf24dd64
df = pd.read_csv('cars.csv')
df.head()
```

Out[44]:

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

In [45]:
```
df.pivot_table(values='(kW)', index='YEAR', columns='Make', aggfunc=np.mean
)
```

Out[45]:

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 [46]:
```
df.pivot_table(values='(kW)', index='YEAR', columns='Make', aggfunc=[np.mea
n,np.min], margins=True)
```

Out[46]:

	mean									an
Make	**BMW**	**CHEVROLET**	**FORD**	**KIA**	**MITSUBISHI**	**NISSAN**	**SMART**	**TESLA**	**All**	**BM**
YEAR										
2012	NaN	NaN	NaN	NaN	49.0	80.0	NaN	NaN	64.500000	N
2013	NaN	NaN	107.0	NaN	49.0	80.0	35.0	280.000000	158.444444	N
2014	NaN	104.0	107.0	NaN	49.0	80.0	35.0	268.333333	135.000000	N
2015	125.0	104.0	107.0	81.0	49.0	80.0	35.0	320.666667	181.428571	12
2016	125.0	104.0	107.0	81.0	49.0	80.0	35.0	409.700000	252.263158	12
All	125.0	104.0	107.0	81.0	49.0	80.0	35.0	345.478261	190.622642	12

Date Functionality in Pandas

In [47]:
```
import pandas as pd
import numpy as np
```

Timestamp

In [48]:
```
pd.Timestamp('9/1/2016 10:05AM')
```

Out[48]:
```
Timestamp('2016-09-01 10:05:00')
```

Period

In [49]:
```
pd.Period('1/2016')
```

Out[49]:
```
Period('2016-01', 'M')
```

In [50]:
```
pd.Period('3/5/2016')
```

Out[50]:
```
Period('2016-03-05', 'D')
```

DatetimeIndex

In [51]:
```
t1 = pd.Series(list('abc'), [pd.Timestamp('2016-09-01'), 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
```

In [52]:
```
type(t1.index)
```

Out[52]:
```
pandas.core.indexes.datetimes.DatetimeIndex
```

PeriodIndex

In [53]:
```
t2 = pd.Series(list('def'), [pd.Period('2016-09'), pd.Period('2016-10'), pd
.Period('2016-11')])
t2
```

Out[53]:
```
2016-09    d
2016-10    e
2016-11    f
Freq: M, dtype: object
```

In [54]:
```
type(t2.index)
```

Out[54]:
```
pandas.core.indexes.period.PeriodIndex
```

Converting to Datetime

In [55]:
```
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=lis
t('ab'))
ts3
```

Out[55]:

	a	b
2 June 2013	61	99
Aug 29, 2014	73	44
2015-06-26	52	47
7/12/16	59	31

In [56]:
```
ts3.index = pd.to_datetime(ts3.index)
ts3
```

Out[56]:

	a	b
2013-06-02	61	99
2014-08-29	73	44
2015-06-26	52	47
2016-07-12	59	31

In [57]:
```
pd.to_datetime('4.7.12', dayfirst=True)
```

Out[57]:
```
Timestamp('2012-07-04 00:00:00')
```

Timedeltas

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')
```

Working with Dates in a Dataframe

In [60]:
```
dates = pd.date_range('10-01-2016', periods=9, freq='2W-SUN')
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')
```

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=dat
es)
df
```

Out[61]:

	Count 1	Count 2
2016-10-02	107	118
2016-10-16	116	121
2016-10-30	123	125
2016-11-13	125	125
2016-11-27	128	118
2016-12-11	137	119
2016-12-25	143	121
2017-01-08	143	125
2017-01-22	142	123

In [62]:
```
df.index.weekday_name
```

Out[62]:
```
Index(['Sunday', 'Sunday', 'Sunday', 'Sunday', 'Sunday', 'Sunday', 'Sunda
y',
       'Sunday', 'Sunday'],
      dtype='object')
```

In [63]:
```
df.diff()
```

Out[63]:

	Count 1	Count 2
2016-10-02	NaN	NaN
2016-10-16	9.0	3.0
2016-10-30	7.0	4.0
2016-11-13	2.0	0.0
2016-11-27	3.0	-7.0
2016-12-11	9.0	1.0
2016-12-25	6.0	2.0
2017-01-08	0.0	4.0
2017-01-22	-1.0	-2.0

In [64]:
```
df.resample('M').mean()
```

Out[64]:

	Count 1	Count 2
2016-10-31	115.333333	121.333333
2016-11-30	126.500000	121.500000
2016-12-31	140.000000	120.000000
2017-01-31	142.500000	124.000000

In [65]:
```
df['2017']
```

Out[65]:

	Count 1	Count 2
2017-01-08	143	125
2017-01-22	142	123

In [66]:
```
df['2016-12']
```

Out[66]:

	Count 1	Count 2
2016-12-11	137	119
2016-12-25	143	121

In [67]:
```
df['2016-12':]
```

Out[67]:

	Count 1	Count 2
2016-12-11	137	119
2016-12-25	143	121
2017-01-08	143	125
2017-01-22	142	123

In [68]:
```
df.asfreq('W', method='ffill')
```

Out[68]:

	Count 1	Count 2
2016-10-02	107	118
2016-10-09	107	118
2016-10-16	116	121
2016-10-23	116	121
2016-10-30	123	125
2016-11-06	123	125
2016-11-13	125	125
2016-11-20	125	125
2016-11-27	128	118
2016-12-04	128	118
2016-12-11	137	119
2016-12-18	137	119
2016-12-25	143	121
2017-01-01	143	121
2017-01-08	143	125
2017-01-15	143	125
2017-01-22	142	123

In [69]:
```
import matplotlib.pyplot as plt
%matplotlib inline

df.plot()
```

Out[69]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xa9f1940>
```