Fusão de Dataframes

1 - Carregar Dataframe

In [1]:

```
import numpy as np
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

2 - Adicionar uma nova coluna (mesmo comprimento)

In [2]:

```
df['Date'] = ['December 1', 'January 1', 'mid-day']
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-day

3 - Flag de entraga do produto

In [3]:

```
df['Deliverd'] = True
df
```

Out[3]:

	Name	Item Purchased	Cost	Date	Deliverd
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-day	True

4 - Fornecer uma lista tão longa quanto o Dataframe

In [4]:

```
df['Feedback'] = ['Positive', None, 'Negative']
df
```

Out[4]:

	Name	Item Purchased	Cost	Date	Deliverd	Feedback
Store 1	Chris	Sponge	22.5	December 1	True	Positive
Store 1	Kevyn	Kitty Litter	2.5	January 1	True	None

	Name	Item Purchased	Cost	Date	Deliverd	Feedback
Store 2	Filip	Spoon	5.0	mid-day	True	Negative

5 - Reiniciando índices

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	Deliverd	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

6 - Criação de dois Dataframes com sobreposição

In [8]:

```
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
```

7 - Outer Join

In [9]:

```
pd.merge(staff_df, student_df, how='outer', left_index=True, right_index=True)
```

Out[9]:

	Role	School
Name		
James	Grader	Business
Kelly	Director of HR	NaN
Mike	NaN	Law

	Role	School
Name		
Sally	Course liasion	Engineering

8 - Inner Join

In [10]:

```
pd.merge(staff_df, student_df, how='inner', left_index=True, right_index=True)
```

Out[10]:

	Role	School
Name		
Sally	Course liasion	Engineering
James	Grader	Business

9 - Left Join

In [11]:

```
pd.merge(staff_df, student_df, how='left', left_index=True, right_index=True)
```

Out[11]:

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

10 - Right Join

In [12]:

```
pd.merge(staff_df, student_df, how='right', left_index=True, right_index=True)
```

Out[12]:

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

11 - Fusão por colunas

In [13]:

```
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[13]:

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

12 - Conflitos

In [14]:
```
staff_df = pd.DataFrame([{'Name': 'Kelly', 'Role': 'Director of HR', 'Location': 'State St
                         {'Name': 'Sally', 'Role': 'Course liasion', 'Location': 'Washingt
                         {'Name': 'James', 'Role': 'Grader', 'Location': 'Washington Avenu

student_df = pd.DataFrame([{'Name': 'James', 'School': 'Business', 'Location': '1024 Billi
                           {'Name': 'Mike', 'School': 'Law', 'Location': 'Fraternity House
                           {'Name': 'Sally', 'School': 'Engineering', 'Location': '512 Wil

pd.merge(staff_df, student_df, how='left', left_on='Name', right_on='Name')
```

Out[14]:

	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

13 - Multi-Indexação

In [17]:
```
staff_df = pd.DataFrame([{'First Name': 'Kelly', 'Last Name': 'Desjardins', 'Role': 'Direc
                         {'First Name': 'Sally', 'Last Name': 'Brooks', 'Role': 'Course li
                         {'First Name': 'James', 'Last Name': 'Wilde', 'Role': 'Grader'}])

student_df = pd.DataFrame([{'First Name': 'James', 'Last Name': 'Hammond', 'School': 'Busi
                           {'First Name': 'Mike', 'Last Name': 'Smith', 'School': 'Law'},
                           {'First Name': 'Sally', 'Last Name': 'Brooks', 'School': 'Engin

# staff_df
# student_df

pd.merge(staff_df, student_df, how='inner', left_on=['First Name','Last Name'], right_on=
```

Out[17]:

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

Python & Pandas Idiomáticos

14 - Importar CSV Censo USA

In [39]:
```
import pandas as pd

df = pd.read_csv('./Data/census.csv')
df.head()
```

Out[39]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMATESBASE2(
0	40	3	6	1	0	Alabama	Alabama	4779736	4780
1	50	3	6	1	1	Alabama	Autauga County	54571	545
2	50	3	6	1	3	Alabama	Baldwin County	182265	1822
3	50	3	6	1	5	Alabama	Barbour County	27457	274

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMATESBASE20
4	50	3	6	1	7	Alabama	Bibb County	22915	229

5 rows × 100 columns

15 - Encadeamento de métodos (Idiomático)

In [21]:

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

Out[21]:

STNAME	CTYNAME	SUMLEV	REGION	DIVISION	STATE	COUNTY	CENSUS2010POP	Estimates Base 2010	POPESTI
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	
...	**...**	...	...	...	...	...	...	...	
Wyoming	**Sweetwater County**	50.0	4.0	8.0	56.0	37.0	43806.0	43806.0	
	Teton County	50.0	4.0	8.0	56.0	39.0	21294.0	21294.0	
	Uinta County	50.0	4.0	8.0	56.0	41.0	21118.0	21118.0	
	Washakie County	50.0	4.0	8.0	56.0	43.0	8533.0	8533.0	
	Weston County	50.0	4.0	8.0	56.0	45.0	7208.0	7208.0	

3142 rows × 98 columns

16 - Forma Tradicional do código anterior

In [22]:

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

Out[22]:

STNAME	CTYNAME	SUMLEV	REGION	DIVISION	STATE	COUNTY	CENSUS2010POP	Estimates Base 2010	POPESTI

STNAME	CTYNAME	SUMLEV	REGION	DIVISION	STATE	COUNTY	CENSUS2010POP	Estimates Base 2010	POPESTI
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	
...	...	...	...	...	...	...	...	...	
Wyoming	Sweetwater County	50	4	8	56	37	43806	43806	
	Teton County	50	4	8	56	39	21294	21294	
	Uinta County	50	4	8	56	41	21118	21118	
	Washakie County	50	4	8	56	43	8533	8533	
	Weston County	50	4	8	56	45	7208	7208	

3142 rows × 98 columns

17 - Apply and Lambdas

In [23]:

```
import numpy as np

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
```

Função Groupby()

18 - Exclusão da totalização dos estados (==40)

In [25]:
```
df = df[df['SUMLEV']==50]
df.head()
```

Out[25]:

		SUMLEV	REGION	DIVISION	STATE	COUNTY	CENSUS2010POP	ESTIMATESBASE2010
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

19 - Iterar todos os estados e gerar uma lista dos números com a média da população

In [91]:
```
# %%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))
```

20 - Função Groupby()

In [92]:
```
# %%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))
```

21 - Método agg() - construir um data frame sumário, com a população média por Estado

In [34]:
```
df.groupby('STNAME').agg({'CENSUS2010POP': np.average})
```

Out[34]:

	CENSUS2010POP
STNAME	
Alabama	1.405805e+05
Alaska	4.734873e+04
Arizona	7.990021e+05
Arkansas	7.673468e+04
California	1.262846e+06

STNAME	CENSUS2010POP
Colorado	1.547445e+05
Connecticut	7.942438e+05
Delaware	4.489670e+05
District of Columbia	6.017230e+05
Florida	5.529797e+05
Georgia	1.210957e+05
Hawaii	4.534337e+05
Idaho	6.967031e+04
Illinois	2.491385e+05
Indiana	1.394366e+05
Iowa	6.092710e+04
Kansas	5.383242e+04
Kentucky	7.172507e+04
Louisiana	1.394884e+05
Maine	1.562778e+05
Maryland	4.618842e+05
Massachusetts	8.730172e+05
Michigan	2.353248e+05
Minnesota	1.205438e+05
Mississippi	7.150113e+04
Missouri	1.032574e+05
Montana	3.471632e+04
Nebraska	3.885832e+04
Nevada	3.000612e+05
New Hampshire	2.393582e+05
New Jersey	7.992631e+05
New Mexico	1.211282e+05
New York	6.151778e+05
North Carolina	1.888214e+05
North Dakota	2.491078e+04
Ohio	2.592473e+05
Oklahoma	9.618849e+04
Oregon	2.070851e+05
Pennsylvania	3.735994e+05
Rhode Island	3.508557e+05
South Carolina	1.968240e+05
South Dakota	2.430388e+04

	CENSUS2010POP
STNAME	
Tennessee	1.322105e+05
Texas	1.972201e+05
Utah	1.842590e+05
Vermont	8.343213e+04
Virginia	1.193718e+05
Washington	3.362270e+05
West Virginia	6.617836e+04
Wisconsin	1.558078e+05
Wyoming	4.696883e+04

22 - Função Groupby (Série vs. Dataframe)

In [40]:

```
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'>
```

```
/var/folders/01/_r7b02r11p15j0s54gb9x0040000gn/T/ipykernel_3089/817127102.py:1: FutureWarn
ing: Indexing with multiple keys (implicitly converted to a tuple of keys) will be depreca
ted, use a list instead.
  print(type(df.groupby(level=0)['POPESTIMATE2010','POPESTIMATE2011']))
```

23 - Na Série

In [46]:

```
(df.set_index('STNAME').groupby(level=0)['CENSUS2010POP'].agg(avg='mean', soma='sum'))
```

Out[46]:

	avg	soma
STNAME		
Alabama	1.405805e+05	9559472
Alaska	4.734873e+04	1420462
Arizona	7.990021e+05	12784034
Arkansas	7.673468e+04	5831836
California	1.262846e+06	74507912
Colorado	1.547445e+05	10058392
Connecticut	7.942438e+05	7148194
Delaware	4.489670e+05	1795868
District of Columbia	6.017230e+05	1203446
Florida	5.529797e+05	37602620
Georgia	1.210957e+05	19375306
Hawaii	4.534337e+05	2720602
Idaho	6.967031e+04	3135164
Illinois	2.491385e+05	25661264

	avg	soma
STNAME		
Indiana	1.394366e+05	12967604
Iowa	6.092710e+04	6092710
Kansas	5.383242e+04	5706236
Kentucky	7.172507e+04	8678734
Louisiana	1.394884e+05	9066744
Maine	1.562778e+05	2656722
Maryland	4.618842e+05	11547104
Massachusetts	8.730172e+05	13095258
Michigan	2.353248e+05	19767280
Minnesota	1.205438e+05	10607850
Mississippi	7.150113e+04	5934594
Missouri	1.032574e+05	11977854
Montana	3.471632e+04	1978830
Nebraska	3.885832e+04	3652682
Nevada	3.000612e+05	5401102
New Hampshire	2.393582e+05	2632940
New Jersey	7.992631e+05	17583788
New Mexico	1.211282e+05	4118358
New York	6.151778e+05	38756204
North Carolina	1.888214e+05	19070966
North Dakota	2.491078e+04	1345182
Ohio	2.592473e+05	23073008
Oklahoma	9.618849e+04	7502702
Oregon	2.070851e+05	7662148
Pennsylvania	3.735994e+05	25404758
Rhode Island	3.508557e+05	2105134
South Carolina	1.968240e+05	9250728
South Dakota	2.430388e+04	1628360
Tennessee	1.322105e+05	12692210
Texas	1.972201e+05	50291122
Utah	1.842590e+05	5527770
Vermont	8.343213e+04	1251482
Virginia	1.193718e+05	15995826
Washington	3.362270e+05	13449080
West Virginia	6.617836e+04	3705988
Wisconsin	1.558078e+05	11373972
Wyoming	4.696883e+04	1127252

24 - No Dataframe

In [55]:
```
(df.set_index('STNAME').groupby(level=0)['POPESTIMATE2010','POPESTIMATE2011'].agg(['mean',

# df.set_index('STNAME').groupby(level=0)[['POPESTIMATE2010','POPESTIMATE2011']]
#     .agg({"POPESTIMATE2010": [np.mean, np.sum], "POPESTIMATE2011": [np.mean, np.sum]})
```

```
/var/folders/01/_r7b02r11p15j0s54gb9x0040000gn/T/ipykernel_3089/596025003.py:1: FutureWarn
ing: Indexing with multiple keys (implicitly converted to a tuple of keys) will be depreca
ted, use a list instead.
  (df.set_index('STNAME').groupby(level=0)['POPESTIMATE2010','POPESTIMATE2011'].agg(['mea
n', 'sum']))
```

Out[55]:

	POPESTIMATE2010		POPESTIMATE2011	
	mean	sum	mean	sum
STNAME				
Alabama	1.407400e+05	9570322	1.412091e+05	9602216
Alaska	4.760140e+04	1428042	4.818133e+04	1445440
Arizona	8.010260e+05	12816416	8.085915e+05	12937464
Arkansas	7.690511e+04	5844788	7.732995e+04	5877076
California	1.265562e+06	74668158	1.277967e+06	75400068
Colorado	1.553309e+05	10096508	1.575225e+05	10238960
Connecticut	7.954927e+05	7159434	7.977242e+05	7179518
Delaware	4.498955e+05	1799582	4.539580e+05	1815832
District of Columbia	6.051260e+05	1210252	6.204720e+05	1240944
Florida	5.544085e+05	37699780	5.619274e+05	38211066
Georgia	1.214182e+05	19426908	1.226535e+05	19624560
Hawaii	4.546600e+05	2727960	4.594090e+05	2756454
Idaho	6.982160e+04	3141972	7.040596e+04	3168268
Illinois	2.493446e+05	25682498	2.497453e+05	25723764
Indiana	1.395826e+05	12981180	1.401472e+05	13033690
Iowa	6.101388e+04	6101388	6.130778e+04	6130778
Kansas	5.394008e+04	5717648	5.414938e+04	5739834
Kentucky	7.186673e+04	8695874	7.219640e+04	8735764
Louisiana	1.398446e+05	9089902	1.407810e+05	9150762
Maine	1.561994e+05	2655390	1.562655e+05	2656514
Maryland	4.630727e+05	11576818	4.675337e+05	11688342
Massachusetts	8.753381e+05	13130072	8.815729e+05	13223594
Michigan	2.351755e+05	19754738	2.351569e+05	19753178
Minnesota	1.207023e+05	10621806	1.215482e+05	10696238
Mississippi	7.157388e+04	5940632	7.175901e+04	5955998
Missouri	1.033802e+05	11992104	1.036308e+05	12021174
Montana	3.475940e+04	1981286	3.500863e+04	1995492

	POPESTIMATE2010		POPESTIMATE2011	
	mean	sum	mean	sum
STNAME				
Nebraska	3.893670e+04	3660050	3.919964e+04	3684766
Nevada	3.003822e+05	5406880	3.020910e+05	5437638
New Hampshire	2.394015e+05	2633416	2.396989e+05	2636688
New Jersey	8.003528e+05	17607762	8.039031e+05	17685868
New Mexico	1.214554e+05	4129482	1.222486e+05	4156452
New York	6.159657e+05	38805840	6.197842e+05	39046404
North Carolina	1.892867e+05	19117958	1.911094e+05	19302050
North Dakota	2.498259e+04	1349060	2.538244e+04	1370652
Ohio	2.593431e+05	23081532	2.594481e+05	23090884
Oklahoma	9.639990e+04	7519192	9.709297e+04	7573252
Oregon	2.074579e+05	7675944	2.091086e+05	7737018
Pennsylvania	3.738828e+05	25424028	3.748589e+05	25490404
Rhode Island	3.510730e+05	2106438	3.506187e+05	2103712
South Carolina	1.972721e+05	9271788	1.988397e+05	9345466
South Dakota	2.436713e+04	1632598	2.460564e+04	1648578
Tennessee	1.324289e+05	12713170	1.333002e+05	12796816
Texas	1.979950e+05	50488726	2.012115e+05	51308928
Utah	1.850284e+05	5550852	1.877627e+05	5632880
Vermont	8.346453e+04	1251968	8.355827e+04	1253374
Virginia	1.197879e+05	16051574	1.210565e+05	16221566
Washington	3.371530e+05	13486120	3.411615e+05	13646458
West Virginia	6.622232e+04	3708450	6.624814e+04	3709896
Wisconsin	1.558960e+05	11380408	1.564307e+05	11419440
Wyoming	4.704300e+04	1129032	4.731400e+04	1135536

25 - Comportamento não esperado dada a mudança de rotulagem

In [37]:

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

```
/var/folders/01/_r7b02r11p15j0s54gb9x0040000gn/T/ipykernel_3089/1219683218.py:1: FutureWar
ning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprec
ated, use a list instead.
  (df.set_index('STNAME').groupby(level=0)['POPESTIMATE2010','POPESTIMATE2011']
```

Out[37]:

	POPESTIMATE2010	POPESTIMATE2011
STNAME		
Alabama	1.407400e+05	9602216
Alaska	4.760140e+04	1445440
Arizona	8.010260e+05	12937464

STNAME	POPESTIMATE2010	POPESTIMATE2011
Arkansas	7.690511e+04	5877076
California	1.265562e+06	75400068
Colorado	1.553309e+05	10238960
Connecticut	7.954927e+05	7179518
Delaware	4.498955e+05	1815832
District of Columbia	6.051260e+05	1240944
Florida	5.544085e+05	38211066
Georgia	1.214182e+05	19624560
Hawaii	4.546600e+05	2756454
Idaho	6.982160e+04	3168268
Illinois	2.493446e+05	25723764
Indiana	1.395826e+05	13033690
Iowa	6.101388e+04	6130778
Kansas	5.394008e+04	5739834
Kentucky	7.186673e+04	8735764
Louisiana	1.398446e+05	9150762
Maine	1.561994e+05	2656514
Maryland	4.630727e+05	11688342
Massachusetts	8.753381e+05	13223594
Michigan	2.351755e+05	19753178
Minnesota	1.207023e+05	10696238
Mississippi	7.157388e+04	5955998
Missouri	1.033802e+05	12021174
Montana	3.475940e+04	1995492
Nebraska	3.893670e+04	3684766
Nevada	3.003822e+05	5437638
New Hampshire	2.394015e+05	2636688
New Jersey	8.003528e+05	17685868
New Mexico	1.214554e+05	4156452
New York	6.159657e+05	39046404
North Carolina	1.892867e+05	19302050
North Dakota	2.498259e+04	1370652
Ohio	2.593431e+05	23090884
Oklahoma	9.639990e+04	7573252
Oregon	2.074579e+05	7737018
Pennsylvania	3.738828e+05	25490404
Rhode Island	3.510730e+05	2103712

STNAME	POPESTIMATE2010	POPESTIMATE2011
South Carolina	1.972721e+05	9345466
South Dakota	2.436713e+04	1648578
Tennessee	1.324289e+05	12796816
Texas	1.979950e+05	51308928
Utah	1.850284e+05	5632880
Vermont	8.346453e+04	1253374
Virginia	1.197879e+05	16221566
Washington	3.371530e+05	13646458
West Virginia	6.622232e+04	3709896
Wisconsin	1.558960e+05	11419440
Wyoming	4.704300e+04	1135536

Tabelas Pivot

26 - Carregar conjunto de dados "car.csv"

In [56]:
```
df = pd.read_csv('./Data/cars.csv')
```

In [57]:
```
df.head()
```

Out[57]:

	YEAR	Make	Model	Size	(kW)	Unnamed: 5	TYPE	CITY (kWh/100 km)	HWY (kWh/100 km)	COMB (kWh/100 km)	(Le
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	

27 - Comparação da marca de veículos elétricos contra os anos e fazer essa comparação em termos de capacidade da bateria

In [58]:
```
df.pivot_table(values='(kW)', index='YEAR', columns='Make', aggfunc=np.mean)
```

Out[58]:

Make / YEAR	BMW	CHEVROLET	FORD	KIA	MITSUBISHI	NISSAN	SMART	TESLA
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

Make	BMW	CHEVROLET	FORD	KIA	MITSUBISHI	NISSAN	SMART	TESLA
YEAR								
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

28 - Lista de funções diferentes para aplicar

In [59]:

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

Out[59]:

	mean										
Make	BMW	CHEVROLET	FORD	KIA	MITSUBISHI	NISSAN	SMART	TESLA	All	BMW	CHEV
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	

Datas no Pandas

29 - Timestamp

In [60]:

```
pd.Timestamp('9/1/2017 10:05AM')
```

Out[60]:

```
Timestamp('2017-09-01 10:05:00')
```

30 - Period

In [61]:

```
pd.Period('1/2017')
```

Out[61]:

```
Period('2017-01', 'M')
```

In [62]:

```
pd.Period('3/1/2017')
```

Out[62]:

```
Period('2017-03-01', 'D')
```

31 - DatetimeIndex

In [63]:

```
t1 = pd.Series(list('abc'), [pd.Timestamp('2016-09-01'), pd.Timestamp('2016-09-02'), pd.Ti
t1
```

Out[63]:

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

32 - Tipo da série

In [64]:
```
type(t1.index)
```

Out[64]:
```
pandas.core.indexes.datetimes.DatetimeIndex
```

33 - PeriodIndex

In [65]:
```
t2 = pd.Series(list('def'), [pd.Period('2016-09'), pd.Period('2016-10'), pd.Period('2016-1
t2
```

Out[65]:
```
2016-09    d
2016-10    e
2016-11    f
Freq: M, dtype: object
```

In [66]:
```
type(t2.index)
```

Out[66]:
```
pandas.core.indexes.period.PeriodIndex
```

34 - Novo DataFrame para conversão to_datetime()

In [69]:
```
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[69]:

	a	b
2 June 2013	27	51
Aug 29, 2014	93	27
2015-06-26	47	34
7/12/16	15	24

35 - Usando to_datetime()

In [70]:
```
ts3.index = pd.to_datetime(ts3.index)
ts3
```

Out[70]:

	a	b
2013-06-02	27	51
2014-08-29	93	27
2015-06-26	47	34
2016-07-12	15	24

36 - Data no formato Europeu

In [73]:
```
pd.to_datetime('4.7.12', dayfirst=True)
```

Out[73]:
```
Timestamp('2012-07-04 00:00:00')
```

37 - Timedeltas

In [75]:
```
pd.Timestamp('9/3/2016') - pd.Timestamp('9/1/2016')
```

Out[75]:
```
Timedelta('2 days 00:00:00')
```

38 - Encontrar datas e horas com Timedeltas

In [76]:
```
pd.Timestamp('9/2/2016 8:10AM') + pd.Timedelta('12D 3H')
```

Out[76]:
```
Timestamp('2016-09-14 11:10:00')
```

39 - Método date_range()

In [77]:
```
dates = pd.date_range('10-01-2016', periods=9, freq='2W-SUN')
dates
```

Out[77]:
```
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')
```

40 - Introduzindo datas aleatórias

In [78]:
```
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[78]:

	Count 1	Count 2
2016-10-02	105	119
2016-10-16	106	115
2016-10-30	108	122
2016-11-13	114	117
2016-11-27	112	120
2016-12-11	121	115
2016-12-25	122	123
2017-01-08	119	127
2017-01-22	115	127

41 - Podemos verificar qual o dia da semana

In [80]:
```
df.index.day_name
```

Out[80]:
```
<bound method inherit_from_data.<locals>.method of DatetimeIndex(['2016-10-02', '2016-10-1
6', '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')>
```

42 - Diferença entre datas

In [81]:
```
df.diff()
```

Out[81]:

	Count 1	Count 2
2016-10-02	NaN	NaN
2016-10-16	1.0	-4.0
2016-10-30	2.0	7.0
2016-11-13	6.0	-5.0
2016-11-27	-2.0	3.0
2016-12-11	9.0	-5.0
2016-12-25	1.0	8.0
2017-01-08	-3.0	4.0
2017-01-22	-4.0	0.0

43 - Função resample()

In [82]:

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

Out[82]:

	Count 1	Count 2
2016-10-31	106.333333	118.666667
2016-11-30	113.000000	118.500000
2016-12-31	121.500000	119.000000
2017-01-31	117.000000	127.000000

44 - Indexação parcial em datas

In [84]:

```
df.loc['2017']
```

Out[84]:

	Count 1	Count 2
2017-01-08	119	127
2017-01-22	115	127

In [85]:

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

Out[85]:

	Count 1	Count 2
2016-12-11	121	115
2016-12-25	122	123

In [86]:

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

Out[86]:

	Count 1	Count 2
2016-12-11	121	115
2016-12-25	122	123
2017-01-08	119	127

	Count 1	Count 2
2017-01-22	115	127

45 - Preenchendo valores ausentes

In [87]:

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

Out[87]:

	Count 1	Count 2
2016-10-02	105	119
2016-10-09	105	119
2016-10-16	106	115
2016-10-23	106	115
2016-10-30	108	122
2016-11-06	108	122
2016-11-13	114	117
2016-11-20	114	117
2016-11-27	112	120
2016-12-04	112	120
2016-12-11	121	115
2016-12-18	121	115
2016-12-25	122	123
2017-01-01	122	123
2017-01-08	119	127
2017-01-15	119	127
2017-01-22	115	127

46 - Plotando

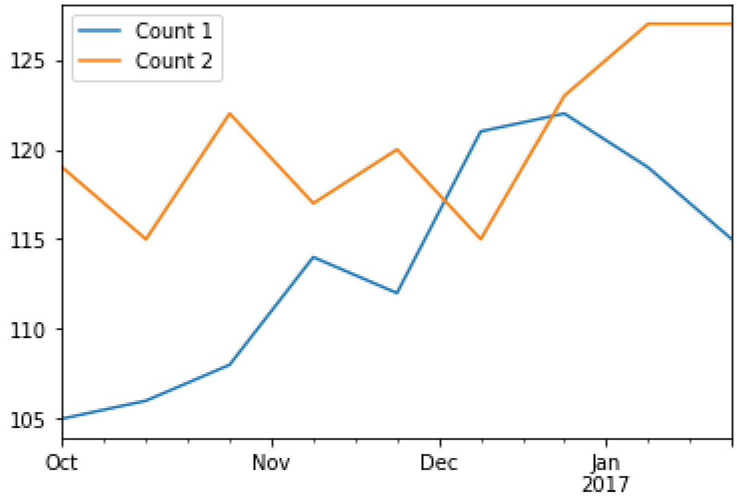
In [88]:

```
import matplotlib.pyplot as plt
%matplotlib inline

df.plot()
```

Out[88]:

```
<AxesSubplot:>
```


Count 1
Count 2
125
120
115
110
105
Oct
Nov
Dec
Jan
2017