The Series Data Structure

In [1]:
```
import pandas as pd
# pd.Series?
```

In [2]:
```
animals = ['Tiger', 'Bear', 'Moose']
pd.Series(animals)
```

Out[2]:
```
0    Tiger
1     Bear
2    Moose
dtype: object
```

In [3]:
```
numbers = [1, 2, 3]
pd.Series(numbers)
```

Out[3]:
```
0    1
1    2
2    3
dtype: int64
```

In [4]:
```
animals = ['Tiger', 'Bear', None]
pd.Series(animals)
```

Out[4]:
```
0    Tiger
1     Bear
2     None
dtype: object
```

In [5]:
```
numbers = [1, 2, None]
pd.Series(numbers)
```

Out[5]:
```
0    1.0
1    2.0
2    NaN
dtype: float64
```

```
In [6]: import numpy as np
        np.nan == None
```

```
Out[6]: False
```

```
In [7]: np.nan == np.nan
```

```
Out[7]: False
```

```
In [8]: np.isnan(np.nan)
```

```
Out[8]: True
```

```
In [9]: sports = {'Archery': 'Bhutan',
                  'Golf': 'Scotland',
                  'Sumo': 'Japan',
                  'Taekwondo': 'South Korea'}
        s = pd.Series(sports)
        s
```

```
Out[9]: Archery           Bhutan
        Golf            Scotland
        Sumo               Japan
        Taekwondo    South Korea
        dtype: object
```

```
In [10]: s.index
```

```
Out[10]: Index(['Archery', 'Golf', 'Sumo', 'Taekwondo'], dtype='object')
```

```
In [11]: s = pd.Series(['Tiger', 'Bear', 'Moose'], index=['India', 'America', 'Canad
         a'])
         s
```

```
Out[11]: India      Tiger
         America     Bear
         Canada     Moose
         dtype: object
```

```
In [12]: sports = {'Archery': 'Bhutan',
                   'Golf': 'Scotland',
                   'Sumo': 'Japan',
                   'Taekwondo': 'South Korea'}
         s = pd.Series(sports, index=['Golf', 'Sumo', 'Hockey'])
         s
```

```
Out[12]: Golf      Scotland
         Sumo         Japan
         Hockey         NaN
         dtype: object
```

Querying a Series

In [13]:
```
sports = {'Archery': 'Bhutan',
          'Golf': 'Scotland',
          'Sumo': 'Japan',
          'Taekwondo': 'South Korea'}
s = pd.Series(sports)
s
```

Out[13]:
```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo    South Korea
dtype: object
```

In [14]:
```
s.iloc[3]
```

Out[14]:
```
'South Korea'
```

In [15]:
```
s.loc['Golf']
```

Out[15]:
```
'Scotland'
```

In [16]:
```
s[3]
```

Out[16]:
```
'South Korea'
```

In [17]:
```
s['Golf']
```

Out[17]:
```
'Scotland'
```

In [18]:
```
sports = {99: 'Bhutan',
          100: 'Scotland',
          101: 'Japan',
          102: 'South Korea'}
s = pd.Series(sports)
```

In [19]:
```
s[0] #This won't call s.iloc[0] as one might expect, it generates an error
 instead
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-19-00fa51989f14> in <module>
----> 1 s[0] #This won't call s.iloc[0] as one might expect, it generates a
n error instead

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\series.py
in __getitem__(self, key)
   1062         key = com.apply_if_callable(key, self)
   1063         try:
-> 1064             result = self.index.get_value(self, key)
   1065 
   1066             if not is_scalar(result):

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\indexes\b
ase.py in get_value(self, series, key)
   4721         k = self._convert_scalar_indexer(k, kind="getitem")
   4722         try:
-> 4723             return self._engine.get_value(s, k, tz=getattr(series.d
type, "tz", None))
   4724         except KeyError as e1:
   4725             if len(self) > 0 and (self.holds_integer() or self.is_b
oolean()):

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_value()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_value()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.Int64Hash
Table.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.Int64Hash
Table.get_item()

KeyError: 0
```

In [20]:
```
s.iloc[0]
```

Out[20]:
```
'Bhutan'
```

In [21]:
```
s = pd.Series([100.00, 120.00, 101.00, 3.00])
s
```

Out[21]:
```
0    100.0
1    120.0
2    101.0
3      3.0
dtype: float64
```

In [22]:
```python
total = 0
for item in s:
    total+=item
print(total)
```

```
324.0
```

In [23]:
```python
import numpy as np

total = np.sum(s)
print(total)
```

```
324.0
```

In [24]:
```python
#this creates a big series of random numbers
s = pd.Series(np.random.randint(0,1000,10000))
s.head()
```

Out[24]:
```
0    328
1    620
2    678
3    398
4    251
dtype: int32
```

In [25]:
```python
len(s)
```

Out[25]:
```
10000
```

In [26]:
```python
%%timeit -n 100
summary = 0
for item in s:
    summary+=item
```

```
1.47 ms ± 76.6 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [27]:
```python
%%timeit -n 100
summary = np.sum(s)
```

```
110 µs ± 52.2 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [28]:
```python
s+=2 #adds two to each item in s using broadcasting
s.head()
```

Out[28]:
```
0    330
1    622
2    680
3    400
4    253
dtype: int32
```

In [29]:
```
for label, value in s.iteritems():
    s.set_value(label, value+2)
s.head()
```

```
C:\Users\XZV838\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykern
el_launcher.py:2: FutureWarning: set_value is deprecated and will be remove
d in a future release. Please use .at[] or .iat[] accessors instead
```

Out[29]:
```
0    332
1    624
2    682
3    402
4    255
dtype: int32
```

In [30]:
```
%%timeit -n 10
s = pd.Series(np.random.randint(0,1000,10000))
for label, value in s.iteritems():
    s.loc[label]= value+2
```

```
4.87 s ± 325 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

In [31]:
```
%%timeit -n 10
s = pd.Series(np.random.randint(0,1000,10000))
s+=2
```

```
373 µs ± 57.8 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

In [32]:
```
s = pd.Series([1, 2, 3])
s.loc['Animal'] = 'Bears'
s
```

Out[32]:
```
0             1
1             2
2             3
Animal    Bears
dtype: object
```

In [33]:
```
original_sports = pd.Series({'Archery': 'Bhutan',
                             'Golf': 'Scotland',
                             'Sumo': 'Japan',
                             'Taekwondo': 'South Korea'})
cricket_loving_countries = pd.Series(['Australia',
                                      'Barbados',
                                      'Pakistan',
                                      'England'],
                                    index=['Cricket',
                                           'Cricket',
                                           'Cricket',
                                           'Cricket'])
all_countries = original_sports.append(cricket_loving_countries)
```

In [34]:
```
original_sports
```

Out[34]:
```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo    South Korea
dtype: object
```

In [35]:
```
cricket_loving_countries
```

Out[35]:
```
Cricket    Australia
Cricket     Barbados
Cricket     Pakistan
Cricket      England
dtype: object
```

In [36]:
```
all_countries
```

Out[36]:
```
Archery           Bhutan
Golf            Scotland
Sumo               Japan
Taekwondo    South Korea
Cricket        Australia
Cricket         Barbados
Cricket         Pakistan
Cricket          England
dtype: object
```

In [37]:
```
all_countries.loc['Cricket']
```

Out[37]:
```
Cricket    Australia
Cricket     Barbados
Cricket     Pakistan
Cricket      England
dtype: object
```

The DataFrame Data Structure

In [38]:
```
import pandas as pd
purchase_1 = pd.Series({'Name': 'Chris',
                        'Item Purchased': 'Dog Food',
                        'Cost': 22.50})
purchase_2 = pd.Series({'Name': 'Kevyn',
                        'Item Purchased': 'Kitty Litter',
                        'Cost': 2.50})
purchase_3 = pd.Series({'Name': 'Vinod',
                        'Item Purchased': 'Bird Seed',
                        'Cost': 5.00})
df = pd.DataFrame([purchase_1, purchase_2, purchase_3], index=['Store 1',
'Store 1', 'Store 2'])
df.head()
```

Out[38]:

	Name	Item Purchased	Cost
Store 1	Chris	Dog Food	22.5
Store 1	Kevyn	Kitty Litter	2.5
Store 2	Vinod	Bird Seed	5.0

In [39]:
```
df.loc['Store 2']
```

Out[39]:
```
Name                      Vinod
Item Purchased        Bird Seed
Cost                          5
Name: Store 2, dtype: object
```

In [40]:
```
type(df.loc['Store 2'])
```

Out[40]:
```
pandas.core.series.Series
```

In [41]:
```
df.loc['Store 1']
```

Out[41]:

	Name	Item Purchased	Cost
Store 1	Chris	Dog Food	22.5
Store 1	Kevyn	Kitty Litter	2.5

In [42]:
```
df.loc['Store 1', 'Cost']
```

Out[42]:
```
Store 1    22.5
Store 1     2.5
Name: Cost, dtype: float64
```

```
In [43]: df.T
```

Out[43]:

	Store 1	Store 1	Store 2
Name	Chris	Kevyn	Vinod
Item Purchased	Dog Food	Kitty Litter	Bird Seed
Cost	22.5	2.5	5

```
In [44]: df.T.loc['Cost']
```

```
Out[44]: Store 1    22.5
         Store 1     2.5
         Store 2       5
         Name: Cost, dtype: object
```

```
In [45]: df['Cost']
```

```
Out[45]: Store 1    22.5
         Store 1     2.5
         Store 2     5.0
         Name: Cost, dtype: float64
```

```
In [46]: df.loc['Store 1']['Cost']
```

```
Out[46]: Store 1    22.5
         Store 1     2.5
         Name: Cost, dtype: float64
```

```
In [47]: df.loc[:,['Name', 'Cost']]
```

Out[47]:

	Name	Cost
Store 1	Chris	22.5
Store 1	Kevyn	2.5
Store 2	Vinod	5.0

```
In [48]: df.drop('Store 1')
```

Out[48]:

	Name	Item Purchased	Cost
Store 2	Vinod	Bird Seed	5.0

```
In [49]: df
```

Out[49]:

	Name	Item Purchased	Cost
Store 1	Chris	Dog Food	22.5
Store 1	Kevyn	Kitty Litter	2.5
Store 2	Vinod	Bird Seed	5.0

```
In [50]: copy_df = df.copy()
         copy_df = copy_df.drop('Store 1')
         copy_df
```

Out[50]:

	Name	Item Purchased	Cost
Store 2	Vinod	Bird Seed	5.0

```
In [51]: copy_df.drop?
```

```
In [52]: del copy_df['Name']
         copy_df
```

Out[52]:

	Item Purchased	Cost
Store 2	Bird Seed	5.0

```
In [53]: df['Location'] = None
         df
```

Out[53]:

	Name	Item Purchased	Cost	Location
Store 1	Chris	Dog Food	22.5	None
Store 1	Kevyn	Kitty Litter	2.5	None
Store 2	Vinod	Bird Seed	5.0	None

Dataframe Indexing and Loading

```
In [54]: costs = df['Cost']
         costs
```

```
Out[54]: Store 1    22.5
         Store 1     2.5
         Store 2     5.0
         Name: Cost, dtype: float64
```

In [55]:
```
costs+=2
costs
```
Out[55]:
```
Store 1    24.5
Store 1     4.5
Store 2     7.0
Name: Cost, dtype: float64
```
In [56]:
```
df
```
Out[56]:

	Name	Item Purchased	Cost	Location
Store 1	Chris	Dog Food	24.5	None
Store 1	Kevyn	Kitty Litter	4.5	None
Store 2	Vinod	Bird Seed	7.0	None

In [ ]:
```
!cat olympics.csv
```
In [58]:
```
df = pd.read_csv('olympics.csv')
df.head()
```
Out[58]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	NaN	№ Summer	01 !	02 !	03 !	Total	№ Winter	01 !	02 !	03 !	Total	№ Games	01 !	02 !	03 !
1	Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0	13	0	0	2
2	Algeria (ALG)	12	5	2	8	15	3	0	0	0	0	15	5	2	8
3	Argentina (ARG)	23	18	24	28	70	18	0	0	0	0	41	18	24	28
4	Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	1	2	9

In [59]:
```
df = pd.read_csv('olympics.csv', index_col = 0, skiprows=1)
df.head()
```
Out[59]:

	№ Summer	01 !	02 !	03 !	Total	№ Winter	01 !.1	02 !.1	03 !.1	Total.1	№ Games	01 !.2	02 !.2	03 !.2
Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0	13	0	0	2
Algeria (ALG)	12	5	2	8	15	3	0	0	0	0	15	5	2	8
Argentina (ARG)	23	18	24	28	70	18	0	0	0	0	41	18	24	28
Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	1	2	9
Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	0	0	2	3	4	5

```
In [60]: df.columns
```

```
Out[60]: Index(['№ Summer', '01 !', '02 !', '03 !', 'Total', '№ Winter', '01 !.1',
                '02 !.1', '03 !.1', 'Total.1', '№ Games', '01 !.2', '02 !.2', '03 !.
         2',
                'Combined total'],
               dtype='object')
```

```
In [61]: for col in df.columns:
             if col[:2]=='01':
                 df.rename(columns={col:'Gold' + col[4:]}, inplace=True)
             if col[:2]=='02':
                 df.rename(columns={col:'Silver' + col[4:]}, inplace=True)
             if col[:2]=='03':
                 df.rename(columns={col:'Bronze' + col[4:]}, inplace=True)
             if col[:1]=='№':
                 df.rename(columns={col:'#' + col[1:]}, inplace=True)

         df.head()
```

Out[61]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1
Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0
Algeria (ALG)	12	5	2	8	15	3	0	0	0	0
Argentina (ARG)	23	18	24	28	70	18	0	0	0	0
Armenia (ARM)	5	1	2	9	12	6	0	0	0	0
Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	0	0

Querying a DataFrame

```
In [62]: df['Gold'] > 0
```

```
Out[62]: Afghanistan (AFG)                               False
         Algeria (ALG)                                    True
         Argentina (ARG)                                  True
         Armenia (ARM)                                    True
         Australasia (ANZ) [ANZ]                          True
                                                         ...
         Independent Olympic Participants (IOP) [IOP]    False
         Zambia (ZAM) [ZAM]                              False
         Zimbabwe (ZIM) [ZIM]                             True
         Mixed team (ZZX) [ZZX]                           True
         Totals                                           True
         Name: Gold, Length: 147, dtype: bool
```

In [63]:
```python
only_gold = df.where(df['Gold'] > 0)
only_gold.head()
```

Out[63]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1
Afghanistan (AFG)	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Algeria (ALG)	12.0	5.0	2.0	8.0	15.0	3.0	0.0	0.0	0.0	0.0
Argentina (ARG)	23.0	18.0	24.0	28.0	70.0	18.0	0.0	0.0	0.0	0.0
Armenia (ARM)	5.0	1.0	2.0	9.0	12.0	6.0	0.0	0.0	0.0	0.0
Australasia (ANZ) [ANZ]	2.0	3.0	4.0	5.0	12.0	0.0	0.0	0.0	0.0	0.0

In [64]:
```python
only_gold['Gold'].count()
```

Out[64]:
```
100
```

In [65]:
```python
df['Gold'].count()
```

Out[65]:
```
147
```

In [66]:
```python
only_gold = only_gold.dropna()
only_gold.head()
```

Out[66]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1
Algeria (ALG)	12.0	5.0	2.0	8.0	15.0	3.0	0.0	0.0	0.0	0.0
Argentina (ARG)	23.0	18.0	24.0	28.0	70.0	18.0	0.0	0.0	0.0	0.0
Armenia (ARM)	5.0	1.0	2.0	9.0	12.0	6.0	0.0	0.0	0.0	0.0
Australasia (ANZ) [ANZ]	2.0	3.0	4.0	5.0	12.0	0.0	0.0	0.0	0.0	0.0
Australia (AUS) [AUS] [Z]	25.0	139.0	152.0	177.0	468.0	18.0	5.0	3.0	4.0	12.0

```
In [67]: only_gold = df[df['Gold'] > 0]
         only_gold.head()
```

Out[67]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1
Algeria (ALG)	12	5	2	8	15	3	0	0	0	0
Argentina (ARG)	23	18	24	28	70	18	0	0	0	0
Armenia (ARM)	5	1	2	9	12	6	0	0	0	0
Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	0	0
Australia (AUS) [AUS] [Z]	25	139	152	177	468	18	5	3	4	12

```
In [68]: len(df[(df['Gold'] > 0) | (df['Gold.1'] > 0)])

Out[68]: 101
```

```
In [69]: df[(df['Gold.1'] > 0) & (df['Gold'] == 0)]
```

Out[69]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1
Liechtenstein (LIE)	16	0	0	0	0	18	2	2	5	9

Indexing Dataframes

```
In [70]: df.head()
```

Out[70]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1
Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0
Algeria (ALG)	12	5	2	8	15	3	0	0	0	0
Argentina (ARG)	23	18	24	28	70	18	0	0	0	0
Armenia (ARM)	5	1	2	9	12	6	0	0	0	0
Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	0	0

In [71]:
```
df['country'] = df.index
df.head()
```

Out[71]:

	# Summer	Gold	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1
Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0
Algeria (ALG)	12	5	2	8	15	3	0	0	0	0
Argentina (ARG)	23	18	24	28	70	18	0	0	0	0
Armenia (ARM)	5	1	2	9	12	6	0	0	0	0
Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	0	0

In [72]:
```
df = df.set_index('Gold')
df.head()
```

Out[72]:

	# Summer	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Gold.2	Si
Gold												
0	13	0	2	2	0	0	0	0	0	13	0	
5	12	2	8	15	3	0	0	0	0	15	5	
18	23	24	28	70	18	0	0	0	0	41	18	
1	5	2	9	12	6	0	0	0	0	11	1	
3	2	4	5	12	0	0	0	0	0	2	3	

In [73]:
```
df = df.reset_index()
df.head()
```

Out[73]:

	Gold	# Summer	Silver	Bronze	Total	# Winter	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Gold.2
0	0	13	0	2	2	0	0	0	0	0	13	0
1	5	12	2	8	15	3	0	0	0	0	15	5
2	18	23	24	28	70	18	0	0	0	0	41	18
3	1	5	2	9	12	6	0	0	0	0	11	1
4	3	2	4	5	12	0	0	0	0	0	2	3

In [74]:
```
df = pd.read_csv('census.csv')
df.head()
```

Out[74]:

	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 [75]:
```
df['SUMLEV'].unique()
```

Out[75]:
```
array([40, 50], dtype=int64)
```

In [76]:
```
df=df[df['SUMLEV'] == 50]
df.head()
```

Out[76]:

	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 [77]:
```python
columns_to_keep = ['STNAME',
                   'CTYNAME',
                   'BIRTHS2010',
                   'BIRTHS2011',
                   'BIRTHS2012',
                   'BIRTHS2013',
                   'BIRTHS2014',
                   'BIRTHS2015',
                   'POPESTIMATE2010',
                   'POPESTIMATE2011',
                   'POPESTIMATE2012',
                   'POPESTIMATE2013',
                   'POPESTIMATE2014',
                   'POPESTIMATE2015']
df = df[columns_to_keep]
df.head()
```

Out[77]:

	STNAME	CTYNAME	BIRTHS2010	BIRTHS2011	BIRTHS2012	BIRTHS2013	BIRTHS2014	BIRTHS2
1	Alabama	Autauga County	151	636	615	574	623	
2	Alabama	Baldwin County	517	2187	2092	2160	2186	2
3	Alabama	Barbour County	70	335	300	283	260	
4	Alabama	Bibb County	44	266	245	259	247	
5	Alabama	Blount County	183	744	710	646	618	

In [78]:
```python
df = df.set_index(['STNAME', 'CTYNAME'])
df.head()
```

Out[78]:

		BIRTHS2010	BIRTHS2011	BIRTHS2012	BIRTHS2013	BIRTHS2014	BIRTHS201
STNAME	**CTYNAME**						
Alabama	**Autauga County**	151	636	615	574	623	60
	Baldwin County	517	2187	2092	2160	2186	224
	Barbour County	70	335	300	283	260	26
	Bibb County	44	266	245	259	247	25
	Blount County	183	744	710	646	618	60

In [79]:
```
df.loc['Michigan', 'Washtenaw County']
```

Out[79]:
```
BIRTHS2010             977
BIRTHS2011            3826
BIRTHS2012            3780
BIRTHS2013            3662
BIRTHS2014            3683
BIRTHS2015            3709
POPESTIMATE2010     345563
POPESTIMATE2011     349048
POPESTIMATE2012     351213
POPESTIMATE2013     354289
POPESTIMATE2014     357029
POPESTIMATE2015     358880
Name: (Michigan, Washtenaw County), dtype: int64
```

In [80]:
```
df.loc[ [('Michigan', 'Washtenaw County'),
         ('Michigan', 'Wayne County')] ]
```

Out[80]:

		BIRTHS2010	**BIRTHS2011**	**BIRTHS2012**	**BIRTHS2013**	**BIRTHS2014**	**BIRTHS2**
STNAME	**CTYNAME**						
Michigan	**Washtenaw County**	977	3826	3780	3662	3683	3
	Wayne County	5918	23819	23270	23377	23607	23

Missing values

In [81]:
```
df = pd.read_csv('log.csv')
df.head()
```

Out[81]:

	time	**user**	**video**	**playback position**	**paused**	**volume**
0	1469974424	cheryl	intro.html	5	False	10.0
1	1469974454	cheryl	intro.html	6	NaN	NaN
2	1469974544	cheryl	intro.html	9	NaN	NaN
3	1469974574	cheryl	intro.html	10	NaN	NaN
4	1469977514	bob	intro.html	1	NaN	NaN

In [82]:
```
df.fillna?
```

In [83]:
```
df = df.set_index('time')
df = df.sort_index()
df.head()
```

Out[83]:

	user	video	playback position	paused	volume
time					
1469974424	cheryl	intro.html	5	False	10.0
1469974424	sue	advanced.html	23	False	10.0
1469974454	cheryl	intro.html	6	NaN	NaN
1469974454	sue	advanced.html	24	NaN	NaN
1469974484	cheryl	intro.html	7	NaN	NaN

In [84]:
```
df = df.reset_index()
df = df.set_index(['time', 'user'])
df.head()
```

Out[84]:

		video	playback position	paused	volume
time	**user**				
1469974424	**cheryl**	intro.html	5	False	10.0
	sue	advanced.html	23	False	10.0
1469974454	**cheryl**	intro.html	6	NaN	NaN
	sue	advanced.html	24	NaN	NaN
1469974484	**cheryl**	intro.html	7	NaN	NaN

In [85]:
```
df = df.fillna(method='ffill')
df.head()
```

Out[85]:

		video	playback position	paused	volume
time	**user**				
1469974424	**cheryl**	intro.html	5	False	10.0
	sue	advanced.html	23	False	10.0
1469974454	**cheryl**	intro.html	6	False	10.0
	sue	advanced.html	24	False	10.0
1469974484	**cheryl**	intro.html	7	False	10.0