[(http://www.pieriandata.com)](http://www.pieriandata.com)

Useful Methods

Let's cover some useful methods and functions built in to pandas. This is actually just a small sampling of the functions and methods available in Pandas, but they are some of the most commonly used. The documentation (https://pandas.pydata.org/pandas-docs/stable/reference/index.html) is a great resource to continue exploring more methods and functions (we will introduce more further along in the course). Here is a list of functions and methods we'll cover here (click on one to jump to that section in this notebook.):

Make sure to view the video lessons to get the full explanation!

The .apply() method

Here we will learn about a very useful method known as **apply** on a DataFrame. This allows us to apply and broadcast custom functions on a DataFrame column

In [1]:
```python
import pandas as pd
import numpy as np
```

In [2]:
```python
df = pd.read_csv('tips.csv')
```

In [3]:
```python
df.head()
```

Out[3]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	356032
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	447807
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	601181
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	467613
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	483273

apply with a function

In [4]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   total_bill        244 non-null    float64
 1   tip               244 non-null    float64
 2   sex               244 non-null    object 
 3   smoker            244 non-null    object 
 4   day               244 non-null    object 
 5   time              244 non-null    object 
 6   size              244 non-null    int64  
 7   price_per_person  244 non-null    float64
 8   Payer Name        244 non-null    object 
 9   CC Number         244 non-null    int64  
 10  Payment ID        244 non-null    object 
dtypes: float64(3), int64(2), object(6)
memory usage: 21.1+ KB
```

In [5]:
```python
def last_four(num):
    return str(num)[-4:]
```

In [6]:
```python
df['CC Number'][0]
```

Out[6]:
```
3560325168603410
```

In [7]:
```
last_four(3560325168603410)
```

Out[7]:
```
'3410'
```

In [8]:
```
df['last_four'] = df['CC Number'].apply(last_four)
```

In [9]:
```
df.head()
```

Out[9]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	356032
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	447807
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	601181
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	467613
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	483273

Using .apply() with more complex functions

In [10]:
```
df['total_bill'].mean()
```

Out[10]:
```
19.78594262295082
```

In [11]:
```
def yelp(price):
    if price < 10:
        return '$'
    elif price >= 10 and price < 30:
        return '$$'
    else:
        return '$$$'
```

In [12]:
```
df['Expensive'] = df['total_bill'].apply(yelp)
```

In [13]:
```
# df
```

apply with lambda

In [14]:
```
def simple(num):
    return num*2
```

In [15]:
```
lambda num: num*2
```

Out[15]:
```
<function __main__.<lambda>(num)>
```

In [16]:
```
df['total_bill'].apply(lambda bill:bill*0.18)
```

Out[16]:
```
0      3.0582
1      1.8612
2      3.7818
3      4.2624
4      4.4262
        ...
239    5.2254
240    4.8924
241    4.0806
242    3.2076
243    3.3804
Name: total_bill, Length: 244, dtype: float64
```

apply that uses multiple columns

Note, there are several ways to do this:

https://stackoverflow.com/questions/19914937/applying-function-with-multiple-arguments-to-create-a-new-pandas-column (https://stackoverflow.com/questions/19914937/applying-function-with-multiple-arguments-to-create-a-new-pandas-column)

In [17]:
```
df.head()
```

Out[17]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	356032
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	447807
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	601181
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	467613
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	483273

In [18]:
```
def quality(total_bill,tip):
    if tip/total_bill  > 0.25:
        return "Generous"
    else:
        return "Other"
```

In [19]:
```
df['Tip Quality'] = df[['total_bill','tip']].apply(lambda df: quality(df['t
```

In [20]:
```
df.head()
```

Out[20]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	356032
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	447807
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	601181
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	467613
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	483273

In [21]:
```
import numpy as np
```

In [22]:
```
df['Tip Quality'] = np.vectorize(quality)(df['total_bill'], df['tip'])
```

In [23]:
```
df.head()
```

Out[23]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	356032
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	447807
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	601181
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	467613
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	483273

So, which one is faster?

In [24]:
```
import timeit

# code snippet to be executed only once
setup = '''
import numpy as np
import pandas as pd
df = pd.read_csv('tips.csv')
def quality(total_bill,tip):
    if tip/total_bill  > 0.25:
        return "Generous"
    else:
        return "Other"
'''

# code snippet whose execution time is to be measured
stmt_one = '''
df['Tip Quality'] = df[['total_bill','tip']].apply(lambda df: quality(df['t
'''

stmt_two = '''
df['Tip Quality'] = np.vectorize(quality)(df['total_bill'], df['tip'])
'''
```

In [25]:
```
timeit.timeit(setup = setup,
                    stmt = stmt_one,
                    number = 1000)
```

Out[25]:
```
5.0198852999999986
```

In [26]:
```
timeit.timeit(setup = setup,
                    stmt = stmt_two,
                    number = 1000)
```

Out[26]:
```
0.21840849999999534
```

Wow! Vectorization is much faster! Keep **np.vectorize()** in mind for the future.

Full Details: https://docs.scipy.org/doc/numpy/reference/generated/numpy.vectorize.html (https://docs.scipy.org/doc/numpy/reference/generated/numpy.vectorize.html)

df.describe for statistical summaries

In [27]:
```
df.describe()
```

Out[27]:

	total_bill	tip	size	price_per_person	CC Number
count	244.000000	244.000000	244.000000	244.000000	2.440000e+02
mean	19.785943	2.998279	2.569672	7.888197	2.563496e+15
std	8.902412	1.383638	0.951100	2.914234	2.369340e+15
min	3.070000	1.000000	1.000000	2.880000	6.040679e+10
25%	13.347500	2.000000	2.000000	5.800000	3.040731e+13
50%	17.795000	2.900000	2.000000	7.255000	3.525318e+15
75%	24.127500	3.562500	3.000000	9.390000	4.553675e+15
max	50.810000	10.000000	6.000000	20.270000	6.596454e+15

In [28]:
```
df.describe().transpose()
```

Out[28]:

	count	mean	std	min	25%	5
total_bill	244.0	1.978594e+01	8.902412e+00	3.070000e+00	1.334750e+01	1.779500e
tip	244.0	2.998279e+00	1.383638e+00	1.000000e+00	2.000000e+00	2.900000e
size	244.0	2.569672e+00	9.510998e-01	1.000000e+00	2.000000e+00	2.000000e
price_per_person	244.0	7.888197e+00	2.914234e+00	2.880000e+00	5.800000e+00	7.255000e
CC Number	244.0	2.563496e+15	2.369340e+15	6.040679e+10	3.040731e+13	3.525318e

sort_values()

In [29]:
```
df.sort_values('tip')
```

Out[29]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
67	3.07	1.00	Female	Yes	Sat	Dinner	1	3.07	Tiffany Brock	435
236	12.60	1.00	Male	Yes	Sat	Dinner	2	6.30	Matthew Myers	354
92	5.75	1.00	Female	Yes	Fri	Dinner	2	2.88	Leah Ramirez	350
111	7.25	1.00	Female	No	Sat	Dinner	1	7.25	Terri Jones	355
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	356
...	...	...	...	...	...	...	...	...	...	
141	34.30	6.70	Male	No	Thur	Lunch	6	5.72	Steven Carlson	352
59	48.27	6.73	Male	No	Sat	Dinner	4	12.07	Brian Ortiz	659
23	39.42	7.58	Male	No	Sat	Dinner	4	9.86	Lance Peterson	354
212	48.33	9.00	Male	No	Sat	Dinner	4	12.08	Alex Williamson	
170	50.81	10.00	Male	Yes	Sat	Dinner	3	16.94	Gregory Clark	547

244 rows × 14 columns

In [31]:
```
# Helpful if you want to reorder after a sort
# https://stackoverflow.com/questions/13148429/how-to-change-the-order-of-d
df.sort_values(['tip','size'])
```

Out[31]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
67	3.07	1.00	Female	Yes	Sat	Dinner	1	3.07	Tiffany Brock	435
111	7.25	1.00	Female	No	Sat	Dinner	1	7.25	Terri Jones	355
92	5.75	1.00	Female	Yes	Fri	Dinner	2	2.88	Leah Ramirez	350
236	12.60	1.00	Male	Yes	Sat	Dinner	2	6.30	Matthew Myers	354
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	356
...	...	...	...	...	...	...	...	...	...	
141	34.30	6.70	Male	No	Thur	Lunch	6	5.72	Steven Carlson	352
59	48.27	6.73	Male	No	Sat	Dinner	4	12.07	Brian Ortiz	659
23	39.42	7.58	Male	No	Sat	Dinner	4	9.86	Lance Peterson	354
212	48.33	9.00	Male	No	Sat	Dinner	4	12.08	Alex Williamson	
170	50.81	10.00	Male	Yes	Sat	Dinner	3	16.94	Gregory Clark	547

244 rows × 14 columns

df.corr() for correlation checks

Wikipedia on Correlation (https://en.wikipedia.org/wiki/Correlation_and_dependence)

In [29]:
```
df.corr()
```

Out[29]:

	total_bill	tip	size	price_per_person	CC Number
total_bill	1.000000	0.675734	0.598315	0.647554	0.104576
tip	0.675734	1.000000	0.489299	0.347405	0.110857
size	0.598315	0.489299	1.000000	-0.175359	-0.030239
price_per_person	0.647554	0.347405	-0.175359	1.000000	0.135240
CC Number	0.104576	0.110857	-0.030239	0.135240	1.000000

In [30]:
```
df[['total_bill','tip']].corr()
```

Out[30]:

	total_bill	tip
total_bill	1.000000	0.675734
tip	0.675734	1.000000

idxmin and idxmax

In [31]:
```
df.head()
```

Out[31]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	356032
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	447807
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	601181
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	467613
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	483273

In [32]:
```
df['total_bill'].max()
```

Out[32]:
```
50.81
```

In [33]:
```
df['total_bill'].idxmax()
```

Out[33]:
```
170
```

In [34]:
```
df['total_bill'].idxmin()
```

Out[34]:
```
67
```

In [35]:
```
df.iloc[67]
```

Out[35]:
```
total_bill                      3.07
tip                                1
sex                           Female
smoker                           Yes
day                              Sat
time                          Dinner
size                               1
price_per_person                3.07
Payer Name             Tiffany Brock
CC Number           4359488526995267
Payment ID                   Sat3455
last_four                       5267
Expensive                          $
Tip Quality                 Generous
Name: 67, dtype: object
```

In [36]:
```
df.iloc[170]
```

Out[36]:
```
total_bill                     50.81
tip                               10
sex                             Male
smoker                           Yes
day                              Sat
time                          Dinner
size                               3
price_per_person               16.94
Payer Name             Gregory Clark
CC Number           5473850968388236
Payment ID                   Sat1954
last_four                       8236
Expensive                        $$$
Tip Quality                    Other
Name: 170, dtype: object
```

value_counts

Nice method to quickly get a count per category. Only makes sense on categorical columns.

In [37]:
```
df.head()
```

Out[37]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	356032
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	447807
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	601181
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	467613
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	483273

In [38]:
```
df['sex'].value_counts()
```

Out[38]:
```
Male      157
Female     87
Name: sex, dtype: int64
```

replace

Quickly replace values with another one.

In [39]:
```
df.head()
```

Out[39]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	356032
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	447807
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	601181
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	467613
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	483273

In [40]:
```
df['Tip Quality'].replace(to_replace='Other',value='Ok')
```

Out[40]:
```
0            Ok
1            Ok
2            Ok
3            Ok
4            Ok
5            Ok
6            Ok
7            Ok
8            Ok
9            Ok
10           Ok
11           Ok
12           Ok
13           Ok
14           Ok
15           Ok
16           Ok
17           Ok
18           Ok
19           Ok
20           Ok
21           Ok
22           Ok
23           Ok
24           Ok
25           Ok
26           Ok
27           Ok
28           Ok
29           Ok
          ...
214          Ok
215          Ok
216          Ok
217          Ok
218          Ok
219          Ok
220          Ok
221    Generous
222          Ok
223          Ok
224          Ok
225          Ok
226          Ok
227          Ok
228          Ok
229          Ok
230          Ok
231          Ok
232    Generous
233          Ok
234          Ok
235          Ok
236          Ok
237          Ok
238          Ok
239          Ok
240          Ok
241          Ok
242          Ok
```

```
243          Ok
Name: Tip Quality, Length: 244, dtype: object
```

In [41]:
```
df['Tip Quality'] = df['Tip Quality'].replace(to_replace='Other',value='Ok'
```

In [42]:
```
df.head()
```

Out[42]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	356032
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	447807
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	601181
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	467613
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	483273

unique

In [59]:
```
df['size'].unique()
```

Out[59]:
```
array([2, 3, 4, 1, 6, 5], dtype=int64)
```

In [60]:
```
df['size'].nunique()
```

Out[60]:
```
6
```

In [57]:
```
df['time'].unique()
```

Out[57]:
```
array(['Dinner', 'Lunch'], dtype=object)
```

map

In [45]:
```
my_map = {'Dinner':'D','Lunch':'L'}
```

In [46]:
```
df['time'].map(my_map)
```

Out[46]:
```
0      D
1      D
2      D
3      D
4      D
5      D
6      D
7      D
8      D
9      D
10     D
11     D
12     D
13     D
14     D
15     D
16     D
17     D
18     D
19     D
20     D
21     D
22     D
23     D
24     D
25     D
26     D
27     D
28     D
29     D
      ..
214    D
215    D
216    D
217    D
218    D
219    D
220    L
221    L
222    L
223    L
224    L
225    L
226    L
227    D
228    D
229    D
230    D
231    D
232    D
233    D
234    D
235    D
236    D
237    D
238    D
239    D
240    D
241    D
242    D
```

```
243    D
Name: time, Length: 244, dtype: object
```

In [48]:
```
df.head()
```

Out[48]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	356032
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	447807
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	601181
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	467613
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	483273

Duplicates

.duplicated() and .drop_duplicates()

In [50]:
```
# Returns True for the 1st instance of a duplicated row
df.duplicated()
```

Out[50]:
```
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
10     False
11     False
12     False
13     False
14     False
15     False
16     False
17     False
18     False
19     False
20     False
21     False
22     False
23     False
24     False
25     False
26     False
27     False
28     False
29     False
       ...
214    False
215    False
216    False
217    False
218    False
219    False
220    False
221    False
222    False
223    False
224    False
225    False
226    False
227    False
228    False
229    False
230    False
231    False
232    False
233    False
234    False
235    False
236    False
237    False
238    False
239    False
240    False
241    False
242    False
```

```
243    False
Length: 244, dtype: bool
```

In [51]:
```
simple_df = pd.DataFrame([1,2,2],['a','b','c'])
```

In [52]:
```
simple_df
```

Out[52]:

	0
a	1
b	2
c	2

In [53]:
```
simple_df.duplicated()
```

Out[53]:
```
a    False
b    False
c     True
dtype: bool
```

In [54]:
```
simple_df.drop_duplicates()
```

Out[54]:

	0
a	1
b	2

between

left: A scalar value that defines the left boundary right: A scalar value that defines the right boundary inclusive: A Boolean value which is True by default. If False, it excludes the two passed arguments while checking.

In [64]:
```
df['total_bill'].between(10,20,inclusive=True)
```

Out[64]:
```
0       True
1       True
2      False
3      False
4      False
5      False
6      False
7      False
8       True
9       True
10      True
11     False
12      True
13      True
14      True
15     False
16      True
17      True
18      True
19     False
20      True
21     False
22      True
23     False
24      True
25      True
26      True
27      True
28     False
29      True
       ...  
214    False
215     True
216    False
217     True
218    False
219    False
220     True
221     True
222    False
223     True
224     True
225     True
226     True
227    False
228     True
229    False
230    False
231     True
232     True
233     True
234     True
235     True
236     True
237    False
238    False
239    False
240    False
241    False
242     True
```

```
243     True
Name: total_bill, Length: 244, dtype: bool
```

```
In [65]: df[df['total_bill'].between(10,20,inclusive=True)]
```

Out[65]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478
8	15.04	1.96	Male	No	Sun	Dinner	2	7.52	Joseph Mcdonald	3522
9	14.78	3.23	Male	No	Sun	Dinner	2	7.39	Jerome Abbott	3532
10	10.27	1.71	Male	No	Sun	Dinner	2	5.14	William Riley	
12	15.42	1.57	Male	No	Sun	Dinner	2	7.71	Chad Harrington	
13	18.43	3.00	Male	No	Sun	Dinner	4	4.61	Joshua Jones	6011
14	14.83	3.02	Female	No	Sun	Dinner	2	7.42	Vanessa Jones	30
16	10.33	1.67	Female	No	Sun	Dinner	3	3.44	Elizabeth Foster	4240
17	16.29	3.71	Male	No	Sun	Dinner	3	5.43	John Pittman	6521
18	16.97	3.50	Female	No	Sun	Dinner	3	5.66	Laura Martinez	30
20	17.92	4.08	Male	No	Sat	Dinner	2	8.96	Thomas Rice	4403
22	15.77	2.23	Female	No	Sat	Dinner	2	7.88	Ashley Shelton	3524
24	19.82	3.18	Male	No	Sat	Dinner	2	9.91	Christopher Ross	36
25	17.81	2.34	Male	No	Sat	Dinner	4	4.45	Robert Perkins	30
26	13.37	2.00	Male	No	Sat	Dinner	2	6.68	Kyle Avery	6531
27	12.69	2.00	Male	No	Sat	Dinner	2	6.34	Patrick Barber	30
29	19.65	3.00	Female	No	Sat	Dinner	2	9.82	Melinda Murphy	5489
31	18.35	2.50	Male	No	Sat	Dinner	4	4.59	Danny Santiago	
32	15.06	3.00	Female	No	Sat	Dinner	2	7.53	Amanda Wilson	213
34	17.78	3.27	Male	No	Sat	Dinner	2	8.89	Jacob Castillo	3551
36	16.31	2.00	Male	No	Sat	Dinner	3	5.44	William Ford	3527
37	16.93	3.07	Female	No	Sat	Dinner	3	5.64	Erin Lewis	5161
38	18.69	2.31	Male	No	Sat	Dinner	3	6.23	Brandon Bradley	4427
40	16.04	2.24	Male	No	Sat	Dinner	3	5.35	Adam Edwards	3544
41	17.46	2.54	Male	No	Sun	Dinner	2	8.73	David Boyer	3536

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
42	13.94	3.06	Male	No	Sun	Dinner	2	6.97	Bryan Brown	36
45	18.29	3.00	Male	No	Sun	Dinner	2	9.14	Richard Fitzgerald	375
49	18.04	3.00	Male	No	Sun	Dinner	2	9.02	William Roth	6573
50	12.54	2.50	Male	No	Sun	Dinner	2	6.27	Jeremiah Neal	2225
...	...	...	...	...	...	...	...	...	...	
191	19.81	4.19	Female	Yes	Thur	Lunch	2	9.90	Kristy Boyd	4317
193	15.48	2.02	Male	Yes	Thur	Lunch	2	7.74	Raymond Sullivan	180
194	16.58	4.00	Male	Yes	Thur	Lunch	2	8.29	Benjamin Weber	
196	10.34	2.00	Male	Yes	Thur	Lunch	2	5.17	Eric Martin	30
198	13.00	2.00	Female	Yes	Thur	Lunch	2	6.50	Katherine Bond	4
199	13.51	2.00	Male	Yes	Thur	Lunch	2	6.76	Joseph Murphy MD	6547
200	18.71	4.00	Male	Yes	Thur	Lunch	3	6.24	Jason Conrad	4
201	12.74	2.01	Female	Yes	Thur	Lunch	2	6.37	Abigail Parks	3586
202	13.00	2.00	Female	Yes	Thur	Lunch	2	6.50	Ashley Shaw	180
203	16.40	2.50	Female	Yes	Thur	Lunch	2	8.20	Toni Brooks	3582
205	16.47	3.23	Female	Yes	Thur	Lunch	3	5.49	Carly Reyes	4
209	12.76	2.23	Female	Yes	Sat	Dinner	2	6.38	Sarah Cunningham	341
213	13.27	2.50	Female	Yes	Sat	Dinner	2	6.64	Robin Andersen	
215	12.90	1.10	Female	Yes	Sat	Dinner	2	6.45	Jessica Owen	4
217	11.59	1.50	Male	Yes	Sat	Dinner	2	5.80	Gary Orr	30
220	12.16	2.20	Male	Yes	Fri	Lunch	2	6.08	Ricky Johnson	213
221	13.42	3.48	Female	Yes	Fri	Lunch	2	6.71	Leslie Kaufman	379
223	15.98	3.00	Female	No	Fri	Lunch	3	5.33	Mary Rivera	5343
224	13.42	1.58	Male	Yes	Fri	Lunch	2	6.71	Ronald Vaughn DVM	341
225	16.27	2.50	Female	Yes	Fri	Lunch	2	8.14	Whitney Arnold	3579
226	10.09	2.00	Female	Yes	Fri	Lunch	2	5.04	Ruth Weiss	5268
228	13.28	2.72	Male	No	Sat	Dinner	2	6.64	Glenn Jones	
231	15.69	3.00	Male	Yes	Sat	Dinner	3	5.23	Jason Parks	4

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
232	11.61	3.39	Male	No	Sat	Dinner	2	5.80	James Taylor	6011
233	10.77	1.47	Male	No	Sat	Dinner	2	5.38	Paul Novak	6011
234	15.53	3.00	Male	Yes	Sat	Dinner	2	7.76	Tracy Douglas	4097
235	10.07	1.25	Male	No	Sat	Dinner	2	5.04	Sean Gonzalez	3534
236	12.60	1.00	Male	Yes	Sat	Dinner	2	6.30	Matthew Myers	3543
242	17.82	1.75	Male	No	Sat	Dinner	2	8.91	Dennis Dixon	4
243	18.78	3.00	Female	No	Thur	Dinner	2	9.39	Michelle Hardin	3511

130 rows × 14 columns

sample

In [68]:
```
df.sample(5)
```

Out[68]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
216	28.15	3.00	Male	Yes	Sat	Dinner	5	5.63	Shawn Barnett PhD	45
136	10.33	2.00	Female	No	Thur	Lunch	2	5.16	Donna Kelly	1800
13	18.43	3.00	Male	No	Sun	Dinner	4	4.61	Joshua Jones	60111
146	18.64	1.36	Female	No	Thur	Lunch	3	6.21	Kelly Estrada	
56	38.01	3.00	Male	Yes	Sat	Dinner	4	9.50	James Christensen DDS	3497

In [69]:
```
df.sample(frac=0.1)
```

Out[69]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
73	25.28	5.00	Female	Yes	Sat	Dinner	2	12.64	Julie Holmes	5418
141	34.30	6.70	Male	No	Thur	Lunch	6	5.72	Steven Carlson	3526
239	29.03	5.92	Male	No	Sat	Dinner	3	9.68	Michael Avila	5296
237	32.83	1.17	Male	Yes	Sat	Dinner	2	16.42	Thomas Brown	4284
69	15.01	2.09	Male	Yes	Sat	Dinner	2	7.50	Adam Hall	4700
108	18.24	3.76	Male	No	Sat	Dinner	2	9.12	Steven Grant	4112
85	34.83	5.17	Female	No	Thur	Lunch	4	8.71	Shawna Cook	6011
156	48.17	5.00	Male	No	Sun	Dinner	6	8.03	Ryan Gonzales	35231
196	10.34	2.00	Male	Yes	Thur	Lunch	2	5.17	Eric Martin	30
41	17.46	2.54	Male	No	Sun	Dinner	2	8.73	David Boyer	3536
236	12.60	1.00	Male	Yes	Sat	Dinner	2	6.30	Matthew Myers	3543
225	16.27	2.50	Female	Yes	Fri	Lunch	2	8.14	Whitney Arnold	3579
61	13.81	2.00	Male	Yes	Sat	Dinner	2	6.90	Ryan Hernandez	47
203	16.40	2.50	Female	Yes	Thur	Lunch	2	8.20	Toni Brooks	3582
5	25.29	4.71	Male	No	Sun	Dinner	4	6.32	Erik Smith	2131
220	12.16	2.20	Male	Yes	Fri	Lunch	2	6.08	Ricky Johnson	2131
119	24.08	2.92	Female	No	Thur	Lunch	4	6.02	Melanie Jordan	6
96	27.28	4.00	Male	Yes	Fri	Dinner	2	13.64	Eric Carter	4563
159	16.49	2.00	Male	No	Sun	Dinner	4	4.12	Christopher Soto	30
26	13.37	2.00	Male	No	Sat	Dinner	2	6.68	Kyle Avery	6531
129	22.82	2.18	Male	No	Thur	Lunch	3	7.61	Raymond Torres	4
21	20.29	2.75	Female	No	Sat	Dinner	2	10.14	Natalie Gardner	54481
94	22.75	3.25	Female	No	Fri	Dinner	2	11.38	Jamie Garza	6
39	31.27	5.00	Male	No	Sat	Dinner	3	10.42	Mr. Brandon Berry	6011

nlargest and nsmallest

In [71]:
```
df.nlargest(10,'tip')
```

Out[71]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	
170	50.81	10.00	Male	Yes	Sat	Dinner	3	16.94	Gregory Clark	5473
212	48.33	9.00	Male	No	Sat	Dinner	4	12.08	Alex Williamson	
23	39.42	7.58	Male	No	Sat	Dinner	4	9.86	Lance Peterson	3542
59	48.27	6.73	Male	No	Sat	Dinner	4	12.07	Brian Ortiz	6596
141	34.30	6.70	Male	No	Thur	Lunch	6	5.72	Steven Carlson	3526
183	23.17	6.50	Male	Yes	Sun	Dinner	4	5.79	Dr. Michael James	4
214	28.17	6.50	Female	Yes	Sat	Dinner	3	9.39	Marissa Jackson	4922
47	32.40	6.00	Male	No	Sun	Dinner	4	8.10	James Barnes	3552
239	29.03	5.92	Male	No	Sat	Dinner	3	9.68	Michael Avila	5296
88	24.71	5.85	Male	No	Thur	Lunch	2	12.36	Roger Taylor	4