DataFrames

Throughout the course, most of our data exploration will be done with DataFrames. DataFrames are an extremely powerful tool and a natural extension of the Pandas Series. By definition all a DataFrame is:

A Pandas DataFrame consists of multiple Pandas Series that share index values.

Imports

In [28]:

```
import numpy as np
import pandas as pd
```

Creating a DataFrame from Python Objects

In [29]:

```
# help(pd.DataFrame)
```

In [30]:

```
# Make sure the seed is in the same cell as the random call
# https://stackoverflow.com/questions/21494489/what-does-numpy-random-seed0-do
np.random.seed(101)
mydata = np.random.randint(0,101,(4,3))
```

In [31]:

```
mydata
```

Out[31]:

```
array([[95, 11, 81],
       [70, 63, 87],
       [75,  9, 77],
       [40,  4, 63]])
```

In [32]:

```
myindex = ['CA','NY','AZ','TX']
```

In [33]:

```
mycolumns = ['Jan','Feb','Mar']
```

In [34]:

```
df = pd.DataFrame(data=mydata)
df
```

Out[34]:

	0	1	2

0	95	11	81
1	70	63	87
2	75	9	77
3	40	4	63

In [35]:

```
df = pd.DataFrame(data=mydata,index=myindex)
df
```

Out[35]:

	0	1	2
CA	95	11	81
NY	70	63	87
AZ	75	9	77
TX	40	4	63

In [36]:

```
df = pd.DataFrame(data=mydata,index=myindex,columns=mycolumns)
df
```

Out[36]:

	Jan	Feb	Mar
CA	95	11	81
NY	70	63	87
AZ	75	9	77
TX	40	4	63

In [37]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4 entries, CA to TX
Data columns (total 3 columns):
Jan    4 non-null int32
Feb    4 non-null int32
Mar    4 non-null int32
dtypes: int32(3)
memory usage: 80.0+ bytes
```

Reading a .csv file for a DataFrame

NOTE: We will go over all kinds of data inputs and outputs (.html, .csv, .xlxs , etc...) later on in the course! For now we just need to read in a simple .csv file.

CSV

Comma Separated Values files are text files that use commas as field delimeters.
Unless you're running the virtual environment included with the course, you may need to install `xlrd` and

openpyxl.

In your terminal/command prompt run:

```
conda install xlrd
conda install openpyxl
```

Then restart Jupyter Notebook. (or use pip install if you aren't using the Anaconda Distribution)

Understanding File Paths

You have two options when reading a file with pandas:

1. **If your .py file or .ipynb notebook is located in the exact same folder location as the .csv file you want to read, simply pass in the file name as a string, for example:**

```
df = pd.read_csv('some_file.csv')
```

2. **Pass in the entire file path if you are located in a different directory. The file path must be 100% correct in order for this to work. For example:**

```
df = pd.read_csv("C:\\Users\\myself\\files\\some_file.csv")
```

Print your current directory file path with pwd

In [38]:

```
pwd
```

Out[38]:

```
'C:\\Users\\Marcial\\Pierian-Data-Courses\\Machine-Learning-MasterClass\\03-Pandas'
```

List the files in your current directory with ls

In [39]:

```
ls
```

```
 Volume in drive C has no label.
 Volume Serial Number is 3652-BD2F

 Directory of C:\Users\Marcial\Pierian-Data-Courses\Machine-Learning-MasterClass\03-Panda
s

06/30/2020  05:21 PM    <DIR>          .
06/30/2020  05:21 PM    <DIR>          ..
01/27/2020  01:55 PM    <DIR>          .ipynb_checkpoints
06/30/2020  04:51 PM           565,390 00-Series.ipynb
06/30/2020  05:21 PM           207,278 01-DataFrames.ipynb
01/27/2020  06:24 PM           194,565 02-Conditional-Filtering.ipynb
06/30/2020  11:41 AM            82,092 03-Useful-Methods.ipynb
06/30/2020  11:41 AM            45,221 04-Missing-Data.ipynb
06/30/2020  11:42 AM             1,101 05-Groupby-Operations.ipynb
06/30/2020  11:42 AM             1,103 06-Combining-DataFrames.ipynb
06/30/2020  11:42 AM             1,095 07-Text-Methods.ipynb
06/30/2020  11:42 AM             1,095 08-Time-Methods.ipynb
06/30/2020  11:42 AM             1,101 09-Inputs-and-Outputs.ipynb
06/30/2020  11:42 AM             1,095 10-Simple-Plots.ipynb
06/30/2020  11:42 AM               951 11-Pandas-Project-Exercise.ipynb
06/30/2020  11:42 AM             1,118 12-Pandas-Project-Exercise-Solution.ipynb
02/07/2020  12:26 PM               177 movie_scores.csv
01/27/2020  02:28 PM            18,752 tips.csv
              15 File(s)      1,122,134 bytes
               3 Dir(s)  84,920,594,432 bytes free
```

In [40]:

```
df = pd.read_csv('tips.csv')
```

In [41]:

```
df
```

Out[41]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251
5	25.29	4.71	Male	No	Sun	Dinner	4	6.32	Erik Smith	213140353657882	Sun9679
6	8.77	2.00	Male	No	Sun	Dinner	2	4.38	Kristopher Johnson	2223727524230344	Sun5985
7	26.88	3.12	Male	No	Sun	Dinner	4	6.72	Robert Buck	3514785077705092	Sun8157
8	15.04	1.96	Male	No	Sun	Dinner	2	7.52	Joseph Mcdonald	3522866365840377	Sun6820
9	14.78	3.23	Male	No	Sun	Dinner	2	7.39	Jerome Abbott	3532124519049786	Sun3775
10	10.27	1.71	Male	No	Sun	Dinner	2	5.14	William Riley	566287581219	Sun2546
11	35.26	5.00	Female	No	Sun	Dinner	4	8.82	Diane Macias	4577817359320969	Sun6686
12	15.42	1.57	Male	No	Sun	Dinner	2	7.71	Chad Harrington	577040572932	Sun1300
13	18.43	3.00	Male	No	Sun	Dinner	4	4.61	Joshua Jones	6011163105616890	Sun2971
14	14.83	3.02	Female	No	Sun	Dinner	2	7.42	Vanessa Jones	30016702287574	Sun3848
15	21.58	3.92	Male	No	Sun	Dinner	2	10.79	Matthew Reilly	180073029785069	Sun1878
16	10.33	1.67	Female	No	Sun	Dinner	3	3.44	Elizabeth Foster	4240025044626033	Sun9715
17	16.29	3.71	Male	No	Sun	Dinner	3	5.43	John Pittman	6521340257218708	Sun2998
18	16.97	3.50	Female	No	Sun	Dinner	3	5.66	Laura Martinez	30422275171379	Sun2789
19	20.65	3.35	Male	No	Sat	Dinner	3	6.88	Timothy Oneal	6568069240986485	Sat9213
20	17.92	4.08	Male	No	Sat	Dinner	2	8.96	Thomas Rice	4403296224639756	Sat1709
21	20.29	2.75	Female	No	Sat	Dinner	2	10.14	Natalie Gardner	5448125351489749	Sat9618
22	15.77	2.23	Female	No	Sat	Dinner	2	7.88	Ashley Shelton	3524119516293213	Sat9786
23	39.42	7.58	Male	No	Sat	Dinner	4	9.86	Lance Peterson	3542584061609808	Sat239
24	19.82	3.18	Male	No	Sat	Dinner	2	9.91	Christopher Ross	36739148167928	Sat6236
25	17.81	2.34	Male	No	Sat	Dinner	4	4.45	Robert Perkins	30502930499388	Sat907
26	13.37	2.00	Male	No	Sat	Dinner	2	6.68	Kyle Avery	6531339539615499	Sat6651
27	12.69	2.00	Male	No	Sat	Dinner	2	6.34	Patrick Barber	30155551880343	Sat394
28	21.70	4.30	Male	No	Sat	Dinner	2	10.85	David Collier	5529694315416009	Sat3697
29	19.65	3.00	Female	No	Sat	Dinner	2	9.82	Melinda Murphy	5489272944576051	Sat2467
...	...	...	...	...	...	...	...	...	...	...	...
214	28.17	6.50	Female	Yes	Sat	Dinner	3	9.39	Marissa Jackson	4922302538691962	Sat3374
215	12.90	1.10	Female	Yes	Sat	Dinner	2	6.45	Jessica Owen	4726904879471	Sat6983
216	28.15	3.00	Male	Yes	Sat	Dinner	5	5.63	Shawn Barnett PhD	4590982568244	Sat7320
217	11.59	1.50	Male	Yes	Sat	Dinner	2	5.80	Gary Orr	30324521283406	Sat8489
218	7.74	1.44	Male	Yes	Sat	Dinner	2	3.87	Nicholas Archer	340517153733524	Sat4772

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
219	30.14	3.09	Female	Yes	Sat	Dinner	4	7.54	Shelby House	5020974032 52	Sat8863
220	12.16	2.20	Male	Yes	Fri	Lunch	2	6.08	Ricky Johnson	213109508670736	Fri4607
221	13.42	3.48	Female	Yes	Fri	Lunch	2	6.71	Leslie Kaufman	379437981958785	Fri7511
222	8.58	1.92	Male	Yes	Fri	Lunch	1	8.58	Jason Lawrence	3505302934650403	Fri6624
223	15.98	3.00	Female	No	Fri	Lunch	3	5.33	Mary Rivera	5343428579353069	Fri6014
224	13.42	1.58	Male	Yes	Fri	Lunch	2	6.71	Ronald Vaughn DVM	341503466406403	Fri5959
225	16.27	2.50	Female	Yes	Fri	Lunch	2	8.14	Whitney Arnold	3579111947217428	Fri6665
226	10.09	2.00	Female	Yes	Fri	Lunch	2	5.04	Ruth Weiss	5268689490381635	Fri6359
227	20.45	3.00	Male	No	Sat	Dinner	4	5.11	Robert Bradley	213141668145910	Sat4319
228	13.28	2.72	Male	No	Sat	Dinner	2	6.64	Glenn Jones	502061651712	Sat2937
229	22.12	2.88	Female	Yes	Sat	Dinner	2	11.06	Jennifer Russell	4793003293608	Sat3943
230	24.01	2.00	Male	Yes	Sat	Dinner	4	6.00	Michael Osborne	4258682154026	Sat7872
231	15.69	3.00	Male	Yes	Sat	Dinner	3	5.23	Jason Parks	4812333796161	Sat6334
232	11.61	3.39	Male	No	Sat	Dinner	2	5.80	James Taylor	6011482917327995	Sat2124
233	10.77	1.47	Male	No	Sat	Dinner	2	5.38	Paul Novak	6011698897610858	Sat1467
234	15.53	3.00	Male	Yes	Sat	Dinner	2	7.76	Tracy Douglas	4097938155941930	Sat7220
235	10.07	1.25	Male	No	Sat	Dinner	2	5.04	Sean Gonzalez	3534021246117605	Sat4615
236	12.60	1.00	Male	Yes	Sat	Dinner	2	6.30	Matthew Myers	3543676378973965	Sat5032
237	32.83	1.17	Male	Yes	Sat	Dinner	2	16.42	Thomas Brown	4284722681265508	Sat2929
238	35.83	4.67	Female	No	Sat	Dinner	3	11.94	Kimberly Crane	676184013727	Sat9777
239	29.03	5.92	Male	No	Sat	Dinner	3	9.68	Michael Avila	5296068606052842	Sat2657
240	27.18	2.00	Female	Yes	Sat	Dinner	2	13.59	Monica Sanders	3506806155565404	Sat1766
241	22.67	2.00	Male	Yes	Sat	Dinner	2	11.34	Keith Wong	6011891618747196	Sat3880
242	17.82	1.75	Male	No	Sat	Dinner	2	8.91	Dennis Dixon	4375220550950	Sat17
243	18.78	3.00	Female	No	Thur	Dinner	2	9.39	Michelle Hardin	3511451626698139	Thur672

244 rows × 11 columns

About this DataSet (in case you are interested)

- Description
  - One waiter recorded information about each tip he received over a period of a few months working in one restaurant. He collected several variables:
- Format
  - A data frame with 244 rows and 7 variables
- Details
  - tip in dollars,
  - bill in dollars,
  - sex of the bill payer,
  - whether there were smokers in the party,
  - day of the week,
  - time of day,
  - size of the party.

In all he recorded 244 tips. The data was reported in a collection of case studies for business statistics (Bryant & Smith 1995).

- References
  - Bryant, P. G. and Smith, M (1995) Practical Data Analysis: Case Studies in Business Statistics. Homewood, IL: Richard D. Irwin Publishing:
- Note: We created some additional columns with Fake data, including Name, CC Number, and Payment ID

- Note: We created some additional columns with Fake data, including Name, CC Number, and Payment ID.

DataFrames

Obtaining Basic Information About DataFrame

In [42]:

```
df.columns
```

Out[42]:

```
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size',
       'price_per_person', 'Payer Name', 'CC Number', 'Payment ID'],
      dtype='object')
```

In [43]:

```
df.index
```

Out[43]:

```
RangeIndex(start=0, stop=244, step=1)
```

In [44]:

```
df.head(3)
```

Out[44]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458

In [45]:

```
df.tail(3)
```

Out[45]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
241	22.67	2.00	Male	Yes	Sat	Dinner	2	11.34	Keith Wong	6011891618747196	Sat3880
242	17.82	1.75	Male	No	Sat	Dinner	2	8.91	Dennis Dixon	4375220550950	Sat17
243	18.78	3.00	Female	No	Thur	Dinner	2	9.39	Michelle Hardin	3511451626698139	Thur672

In [46]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 11 columns):
total_bill          244 non-null float64
tip                 244 non-null float64
sex                 244 non-null object
smoker              244 non-null object
day                 244 non-null object
time                244 non-null object
size                244 non-null int64
```

```
price_per_person    244 non-null float64
Payer Name          244 non-null object
CC Number           244 non-null int64
Payment ID          244 non-null object
dtypes: float64(3), int64(2), object(6)
memory usage: 21.0+ KB
```

In [47]:

```
len(df)
```

Out[47]:

```
244
```

In [48]:

```
df.describe()
```

Out[48]:

	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 [49]:

```
df.describe().transpose()
```

Out[49]:

	count	mean	std	min	25%	50%	75%	max
total_bill	244.0	1.978594e+01	8.902412e+00	3.070000e+00	1.334750e+01	1.779500e+01	2.412750e+01	5.081000e+01
tip	244.0	2.998279e+00	1.383638e+00	1.000000e+00	2.000000e+00	2.900000e+00	3.562500e+00	1.000000e+01
size	244.0	2.569672e+00	9.510998e-01	1.000000e+00	2.000000e+00	2.000000e+00	3.000000e+00	6.000000e+00
price_per_person	244.0	7.888197e+00	2.914234e+00	2.880000e+00	5.800000e+00	7.255000e+00	9.390000e+00	2.027000e+01
CC Number	244.0	2.563496e+15	2.369340e+15	6.040679e+10	3.040731e+13	3.525318e+15	4.553675e+15	6.596454e+15

Selection and Indexing

Let's learn how to retrieve information from a DataFrame.

COLUMNS

We will begin be learning how to extract information based on the columns

In [50]:

```
df.head()
```

Out[50]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251

Grab a Single Column

In [51]:

```
df['total_bill']
```

Out[51]:

```
0      16.99
1      10.34
2      21.01
3      23.68
4      24.59
5      25.29
6       8.77
7      26.88
8      15.04
9      14.78
10     10.27
11     35.26
12     15.42
13     18.43
14     14.83
15     21.58
16     10.33
17     16.29
18     16.97
19     20.65
20     17.92
21     20.29
22     15.77
23     39.42
24     19.82
25     17.81
26     13.37
27     12.69
28     21.70
29     19.65
       ...
214    28.17
215    12.90
216    28.15
217    11.59
218     7.74
219    30.14
220    12.16
221    13.42
222     8.58
223    15.98
224    13.42
225    16.27
226    10.09
227    20.45
228    13.28
229    22.12
230    24.01
231    15.69
232    11.61
233    10.77
```

```
234    15.53
235    10.07
236    12.60
237    32.83
238    35.83
239    29.03
240    27.18
241    22.67
242    17.82
243    18.78
Name: total_bill, Length: 244, dtype: float64
```

In [52]:

```
type(df['total_bill'])
```

Out[52]:

```
pandas.core.series.Series
```

Grab Multiple Columns

In [53]:

```
# Note how its a python list of column names! Thus the double brackets.
df[['total_bill','tip']]
```

Out[53]:

	total_bill	tip
0	16.99	1.01
1	10.34	1.66
2	21.01	3.50
3	23.68	3.31
4	24.59	3.61
5	25.29	4.71
6	8.77	2.00
7	26.88	3.12
8	15.04	1.96
9	14.78	3.23
10	10.27	1.71
11	35.26	5.00
12	15.42	1.57
13	18.43	3.00
14	14.83	3.02
15	21.58	3.92
16	10.33	1.67
17	16.29	3.71
18	16.97	3.50
19	20.65	3.35
20	17.92	4.08
21	20.29	2.75
22	15.77	2.23
23	39.42	7.58
24	19.82	3.18

	total_bill	tip
25	17.81	2.34
26	13.37	2.00
27	12.69	2.00
28	21.70	4.30
29	19.65	3.00
...	...	...
214	28.17	6.50
215	12.90	1.10
216	28.15	3.00
217	11.59	1.50
218	7.74	1.44
219	30.14	3.09
220	12.16	2.20
221	13.42	3.48
222	8.58	1.92
223	15.98	3.00
224	13.42	1.58
225	16.27	2.50
226	10.09	2.00
227	20.45	3.00
228	13.28	2.72
229	22.12	2.88
230	24.01	2.00
231	15.69	3.00
232	11.61	3.39
233	10.77	1.47
234	15.53	3.00
235	10.07	1.25
236	12.60	1.00
237	32.83	1.17
238	35.83	4.67
239	29.03	5.92
240	27.18	2.00
241	22.67	2.00
242	17.82	1.75
243	18.78	3.00

244 rows × 2 columns

Create New Columns

In [54]:

```
df['tip_percentage'] = 100* df['tip'] / df['total_bill']
```

In [55]:

```
df.head()
```

Out[55]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID	tip_percer
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959	5.94
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608	16.05
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458	16.65
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260	13.97
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251	14.68

In [56]:

```
df['price_per_person'] = df['total_bill'] / df['size']
```

In [57]:

```
df.head()
```

Out[57]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID	tip_percer
0	16.99	1.01	Female	No	Sun	Dinner	2	8.495000	Christy Cunningham	3560325168603410	Sun2959	5.94
1	10.34	1.66	Male	No	Sun	Dinner	3	3.446667	Douglas Tucker	4478071379779230	Sun4608	16.05
2	21.01	3.50	Male	No	Sun	Dinner	3	7.003333	Travis Walters	6011812112971322	Sun4458	16.65
3	23.68	3.31	Male	No	Sun	Dinner	2	11.840000	Nathaniel Harris	4676137647685994	Sun5260	13.97
4	24.59	3.61	Female	No	Sun	Dinner	4	6.147500	Tonya Carter	4832732618637221	Sun2251	14.68

In [58]:

```
help(np.round)
```

```
Help on function round_ in module numpy:

round_(a, decimals=0, out=None)
    Round an array to the given number of decimals.

    See Also
    --------
    around : equivalent function; see for details.
```

Adjust Existing Columns

In [59]:

```
# Because pandas is based on numpy, we get awesome capabilities with numpy's universal fu
nctions!
df['price_per_person'] = np.round(df['price_per_person'],2)
```

In [60]:

```
df.head()
```

Out[60]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID	tip_percer
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959	5.94
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608	16.05
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458	16.65
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260	13.97
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251	14.68

Remove Columns

In [61]:

```
# df.drop('tip_percentage',axis=1)
```

In [62]:

```
df = df.drop("tip_percentage",axis=1)
```

In [63]:

```
df.head()
```

Out[63]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251

Index Basics

Before going over the same retrieval tasks for rows, let's build some basic understanding of the pandas DataFrame Index.

In [64]:

```
df.head()
```

Out[64]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251

In [65]:

```
df.index
```

Out[65]:

```
RangeIndex(start=0, stop=244, step=1)
```

In [66]:

```
df.set_index('Payment ID')
```

Out[66]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994
Sun2251	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221
Sun9679	25.29	4.71	Male	No	Sun	Dinner	4	6.32	Erik Smith	213140353657882
Sun5985	8.77	2.00	Male	No	Sun	Dinner	2	4.38	Kristopher Johnson	2223727524230344
Sun8157	26.88	3.12	Male	No	Sun	Dinner	4	6.72	Robert Buck	3514785077705092
Sun6820	15.04	1.96	Male	No	Sun	Dinner	2	7.52	Joseph Mcdonald	3522866365840377
Sun3775	14.78	3.23	Male	No	Sun	Dinner	2	7.39	Jerome Abbott	3532124519049786
Sun2546	10.27	1.71	Male	No	Sun	Dinner	2	5.14	William Riley	566287581219
Sun6686	35.26	5.00	Female	No	Sun	Dinner	4	8.82	Diane Macias	4577817359320969
Sun1300	15.42	1.57	Male	No	Sun	Dinner	2	7.71	Chad Harrington	577040572932
Sun2971	18.43	3.00	Male	No	Sun	Dinner	4	4.61	Joshua Jones	6011163105616890
Sun3848	14.83	3.02	Female	No	Sun	Dinner	2	7.42	Vanessa Jones	30016702287574
Sun1878	21.58	3.92	Male	No	Sun	Dinner	2	10.79	Matthew Reilly	180073029785069
Sun9715	10.33	1.67	Female	No	Sun	Dinner	3	3.44	Elizabeth Foster	4240025044626033
Sun2998	16.29	3.71	Male	No	Sun	Dinner	3	5.43	John Pittman	6521340257218708
Sun2789	16.97	3.50	Female	No	Sun	Dinner	3	5.66	Laura Martinez	30422275171379
Sat9213	20.65	3.35	Male	No	Sat	Dinner	3	6.88	Timothy Oneal	6568069240986485
Sat1709	17.92	4.08	Male	No	Sat	Dinner	2	8.96	Thomas Rice	4403296224639756
Sat9618	20.29	2.75	Female	No	Sat	Dinner	2	10.14	Natalie Gardner	5448125351489749
Sat9786	15.77	2.23	Female	No	Sat	Dinner	2	7.88	Ashley Shelton	3524119516293213
Sat239	39.42	7.58	Male	No	Sat	Dinner	4	9.86	Lance Peterson	3542584061609808
Sat6236	19.82	3.18	Male	No	Sat	Dinner	2	9.91	Christopher Ross	36739148167928
Sat907	17.81	2.34	Male	No	Sat	Dinner	4	4.45	Robert Perkins	30502930499388
Sat6651	13.37	2.00	Male	No	Sat	Dinner	2	6.68	Kyle Avery	6531339539615499
Sat394	12.69	2.00	Male	No	Sat	Dinner	2	6.34	Patrick Barber	30155551880343
Sat3697	21.70	4.30	Male	No	Sat	Dinner	2	10.85	David Collier	5529694315416009
Sat2467	19.65	3.00	Female	No	Sat	Dinner	2	9.82	Melinda Murphy	5489272944576051

	...	...	...	...	...	...	...	...	...	...	...
	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
Sat3374	28.17	6.50	Female	Yes	Sat	Dinner	3	9.39	Marissa Jackson	4922302538691962	
~~Sat6983~~	~~12.90~~	~~1.10~~	~~Female~~	~~Yes~~	~~Sat~~	~~Dinner~~	~~2~~	~~6.45~~	~~Jessica Owen~~	~~4726904879471~~	
Sat7320	28.15	3.00	Male	Yes	Sat	Dinner	5	5.63	Shawn Barnett PhD	4590982568244	
Sat8489	11.59	1.50	Male	Yes	Sat	Dinner	2	5.80	Gary Orr	30324521283406	
Sat4772	7.74	1.44	Male	Yes	Sat	Dinner	2	3.87	Nicholas Archer	340517153733524	
Sat8863	30.14	3.09	Female	Yes	Sat	Dinner	4	7.54	Shelby House	502097403252	
Fri4607	12.16	2.20	Male	Yes	Fri	Lunch	2	6.08	Ricky Johnson	213109508670736	
Fri7511	13.42	3.48	Female	Yes	Fri	Lunch	2	6.71	Leslie Kaufman	379437981958785	
Fri6624	8.58	1.92	Male	Yes	Fri	Lunch	1	8.58	Jason Lawrence	3505302934650403	
Fri6014	15.98	3.00	Female	No	Fri	Lunch	3	5.33	Mary Rivera	5343428579353069	
Fri5959	13.42	1.58	Male	Yes	Fri	Lunch	2	6.71	Ronald Vaughn DVM	341503466406403	
Fri6665	16.27	2.50	Female	Yes	Fri	Lunch	2	8.14	Whitney Arnold	3579111947217428	
Fri6359	10.09	2.00	Female	Yes	Fri	Lunch	2	5.04	Ruth Weiss	5268689490381635	
Sat4319	20.45	3.00	Male	No	Sat	Dinner	4	5.11	Robert Bradley	213141668145910	
Sat2937	13.28	2.72	Male	No	Sat	Dinner	2	6.64	Glenn Jones	502061651712	
Sat3943	22.12	2.88	Female	Yes	Sat	Dinner	2	11.06	Jennifer Russell	4793003293608	
Sat7872	24.01	2.00	Male	Yes	Sat	Dinner	4	6.00	Michael Osborne	4258682154026	
Sat6334	15.69	3.00	Male	Yes	Sat	Dinner	3	5.23	Jason Parks	4812333796161	
Sat2124	11.61	3.39	Male	No	Sat	Dinner	2	5.80	James Taylor	6011482917327995	
Sat1467	10.77	1.47	Male	No	Sat	Dinner	2	5.38	Paul Novak	6011698897610858	
Sat7220	15.53	3.00	Male	Yes	Sat	Dinner	2	7.76	Tracy Douglas	4097938155941930	
Sat4615	10.07	1.25	Male	No	Sat	Dinner	2	5.04	Sean Gonzalez	3534021246117605	
Sat5032	12.60	1.00	Male	Yes	Sat	Dinner	2	6.30	Matthew Myers	3543676378973965	
Sat2929	32.83	1.17	Male	Yes	Sat	Dinner	2	16.42	Thomas Brown	4284722681265508	
Sat9777	35.83	4.67	Female	No	Sat	Dinner	3	11.94	Kimberly Crane	676184013727	
Sat2657	29.03	5.92	Male	No	Sat	Dinner	3	9.68	Michael Avila	5296068606052842	
Sat1766	27.18	2.00	Female	Yes	Sat	Dinner	2	13.59	Monica Sanders	3506806155565404	
Sat3880	22.67	2.00	Male	Yes	Sat	Dinner	2	11.34	Keith Wong	6011891618747196	
Sat17	17.82	1.75	Male	No	Sat	Dinner	2	8.91	Dennis Dixon	4375220550950	
Thur672	18.78	3.00	Female	No	Thur	Dinner	2	9.39	Michelle Hardin	3511451626698139	

244 rows × 10 columns

In [67]:

```
df.head()
```

Out[67]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251

In [68]:

```
df = df.set_index('Payment ID')
```

In [69]:

```
df.head()
```

Out[69]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994
Sun2251	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221

In [70]:

```
df = df.reset_index()
```

In [71]:

```
df.head()
```

Out[71]:

	Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
0	Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
1	Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
2	Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
3	Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994
4	Sun2251	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221

ROWS

Let's now explore these same concepts but with Rows.

In [72]:

```
df.head()
```

Out[72]:

	Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
0	Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
1	Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
2	Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
3	Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994
4	Sun2251	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221

In [73]:

```
df = df.set_index('Payment ID')
```

In [74]:

```
df.head()
```

Out[74]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994
Sun2251	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221

Grab a Single Row

In [75]:

```
# Integer Based
df.iloc[0]
```

Out[75]:

```
total_bill                          16.99
tip                                  1.01
sex                                Female
smoker                                 No
day                                   Sun
time                               Dinner
size                                    2
price_per_person                     8.49
Payer Name           Christy Cunningham
CC Number              3560325168603410
Name: Sun2959, dtype: object
```

In [76]:

```
# Name Based
df.loc['Sun2959']
```

Out[76]:

```
total_bill                          16.99
tip                                  1.01
sex                                Female
smoker                                 No
day                                   Sun
time                               Dinner
size                                    2
price_per_person                     8.49
Payer Name           Christy Cunningham
CC Number              3560325168603410
Name: Sun2959, dtype: object
```

Grab Multiple Rows

In [77]:

```
df.iloc[0:4]
```

Out[77]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994

In [78]:

```
df.loc[['Sun2959','Sun5260']]
```

Out[78]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994

Remove Row

Typically are datasets will be large enough that we won't remove rows like this since we won't know thier row location for some specific condition, instead, we drop rows based on conditions such as missing data or column values. The next lecture will cover this in a lot more detail.

In [79]:

```
df.head()
```

Out[79]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994
Sun2251	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221

In [80]:

```
df.drop('Sun2959',axis=0).head()
```

Out[80]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994
Sun2251	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221
Sun9679	25.29	4.71	Male	No	Sun	Dinner	4	6.32	Erik Smith	213140353657882

In [81]:

```
# Error if you have a named index!
# df.drop(0,axis=0).head()
```

Insert a New Row

Pretty rare to add a single row like this. Usually you use pd.concat() to add many rows at once. You could use the .append() method with a list of pd.Series() objects, but you won't see us do this with realistic real-world data.

In [82]:

```
one_row = df.iloc[0]
```

In [83]:

```
one_row
```

Out[83]:

```
total_bill                          16.99
tip                                  1.01
sex                                Female
smoker                                 No
day                                   Sun
time                               Dinner
size                                    2
price_per_person                     8.49
Payer Name           Christy Cunningham
CC Number              3560325168603410
Name: Sun2959, dtype: object
```

In [84]:

```
type(one_row)
```

Out[84]:

```
pandas.core.series.Series
```

In [85]:

```
df.tail()
```

Out[85]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sat2657	29.03	5.92	Male	No	Sat	Dinner	3	9.68	Michael Avila	5296068606052842
Sat1766	27.18	2.00	Female	Yes	Sat	Dinner	2	13.59	Monica Sanders	3506806155565404
Sat3880	22.67	2.00	Male	Yes	Sat	Dinner	2	11.34	Keith Wong	6011891618747196
Sat17	17.82	1.75	Male	No	Sat	Dinner	2	8.91	Dennis Dixon	4375220550950
Thur672	18.78	3.00	Female	No	Thur	Dinner	2	9.39	Michelle Hardin	3511451626698139

In [87]:

```
df.append(one_row).tail()
```

Out[87]:

Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sat1766	27.18	2.00	Female	Yes	Sat	Dinner	2	13.59	Monica Sanders	3506806155565404
Sat3880	22.67	2.00	Male	Yes	Sat	Dinner	2	11.34	Keith Wong	6011891618747196
Sat17	17.82	1.75	Male	No	Sat	Dinner	2	8.91	Dennis Dixon	4375220550950
Thur672	18.78	3.00	Female	No	Thur	Dinner	2	9.39	Michelle Hardin	3511451626698139
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410