Exploring Pandas DataFrames

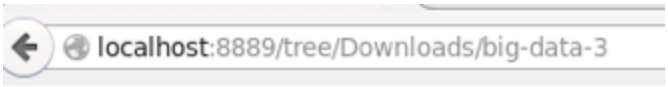
By the end of this activity, you will be able to:

1. Read a CSV file into a Pandas DataFrame
2. View the contents and shape of a DataFrame
3. Filter rows and columns of a DataFrame
4. Calculate the average and sum of a column in a DataFrame
5. Combine two DataFrames by joining on a single column

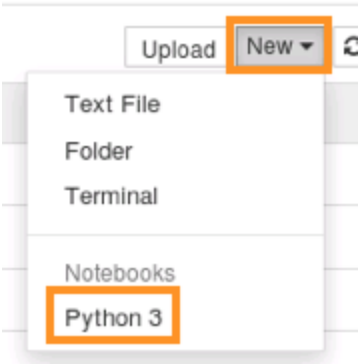
This activity consists of programming in a Jupyter Python Notebook. If you have not already started the Jupyter server, follow the instructions in the Reading entitled *Starting Jupyter for Python Notebooks.*

Step 1. **Open a web browser and create a new Jupyter Python Notebook.** Open a web browser by clicking on the web browser icon at the top of the toolbar:

Navigate to *localhost:8889/tree/Downloads/big-data-3*:



Create a new Python Notebook by clicking on *New,* and then click on *Python 3:*



Step 2. **Load Pandas and Read a CSV file into a DataFrame.** We first load the Pandas library:

```
In [1]: import pandas
```

Note that to execute commands in Jupyter Notebooks, hold the <*shift*> key and press <*enter*>.

We can load the file *buy-clicks.csv* into a Pandas DataFrame:

```
In [2]: buyclicksDF = pandas.read_csv('buy-clicks.csv')
```

This command assigns the DataFrame to a new variable named *buyclicksDF,* and reads the CSV using *pandas.read_csv().*

Step 3. **View the contents and shape of a DataFrame.** We can view the contents of the DataFrame by executing the variable:

```
In [3]: buyclicksDF
```

Out[3]:

	timestamp	txId	userSessionId	team	userId	buyId	price
0	2016-05-26 15:36:54	6004	5820	9	1300	2	3.0
1	2016-05-26 15:36:54	6005	5775	35	868	4	10.0
2	2016-05-26 15:36:54	6006	5679	97	819	5	20.0
3	2016-05-26 16:36:54	6067	5665	18	121	2	3.0

Note that the Notebook does not display all the rows and displays the missing ones as *....* :

28	2016-05-27 02:06:54	6567	5860	57	2221	2	3.0
29	2016-05-27 03:36:54	6651	5955	64	2009	3	5.0
...	...	...	...	...	...	...	...
2917	2016-06-16 08:06:54	39557	34632	72	1294	0	1.0
2918	2016-06-16 08:06:54	39558	34498	59	2029	3	5.0

We can view the first five rows by using the *head(5)* command:

```
In [4]: buyclicksDF.head(5)
```

Out[4]:

	timestamp	txId	userSessionId	team	userId	buyId	price
0	2016-05-26 15:36:54	6004	5820	9	1300	2	3.0
1	2016-05-26 15:36:54	6005	5775	35	868	4	10.0
2	2016-05-26 15:36:54	6006	5679	97	819	5	20.0
3	2016-05-26 16:36:54	6067	5665	18	121	2	3.0
4	2016-05-26 17:06:54	6093	5709	11	2222	5	20.0

We can see how many rows and columns are in the DataFrame by looking at its shape:

```
In [5]: buyclicksDF.shape
Out[5]: (2947, 7)
```

The result says that there are 2947 rows and 7 columns.

Step 4. **Filter rows and columns of a DataFrame.** We can view only the *price* and *userId* columns of the DataFrame:

```
In [6]: buyclicksDF[['price', 'userId']].head(5)
```

Out[6]:

	price	userId
0	3.0	1300
1	10.0	868
2	20.0	819
3	3.0	121
4	20.0	2222

The *[[ ]]* creates a copy of the DataFrame with only the specified columns.

We can also filter rows based on a criteria. The following selects rows with a price less than 3:

```
In [7]: buyclicksDF[buyclicksDF['price'] < 3].head(5)
```

Out[7]:

	timestamp	txId	userSessionId	team	userId	buyId	price
9	2016-05-26 18:36:54	6184	5697	35	2199	1	2.0
14	2016-05-26 20:06:54	6271	5706	9	1652	0	1.0
15	2016-05-26 20:36:54	6292	5921	2	518	0	1.0
18	2016-05-26 22:06:54	6395	5880	35	2146	1	2.0
19	2016-05-26 22:36:54	6411	6230	77	1457	0	1.0

Step 5. **Calculate sum and average of a column.** Pandas DataFrames provide many aggregation operations. We can calculate the total price:

```
In [8]: buyclicksDF['price'].sum()
Out[8]: 21407.0
```

We can also calculate the average price:

```
In [9]: buyclicksDF['price'].mean()
Out[9]: 7.263997285374957
```

A complete list of statistical aggregation operations for Pandas DataFrames is at http://pandas.pydata.org/pandas-docs/stable/api.html#computations-descriptive-stats

Step 6. **Combine two DataFrames.** We can combine two DataFrames on a single column. First, we will load *ad-clicks.csv* into a new DataFrame:

```
In [10]: adclicksDF = pandas.read_csv('ad-clicks.csv')
```

If we look at the contents, we see that *adclicksDF* also has a column named *userId:*

```
In [11]: adclicksDF.head(5)
```

Out[11]:

	timestamp	txId	userSessionId	teamId	userId	adId	adCategory
0	2016-05-26 15:13:22	5974	5809	27	611	2	electronics
1	2016-05-26 15:17:24	5976	5705	18	1874	21	movies
2	2016-05-26 15:22:52	5978	5791	53	2139	25	computers
3	2016-05-26 15:22:57	5973	5756	63	212	10	fashion
4	2016-05-26 15:22:58	5980	5920	9	1027	20	clothing

We can create a combine *buyclicksDF* and *adclicksDF* on the *userId* column with the following command:

```
In [12]: mergeDF = adclicksDF.merge(buyclicksDF, on='userId')
```

The combined DataFrame is assigned to a new variable named *mergeDF.* The command *adclicks.merge()* combines *adclicksDF* with the first argument *buyclicksDF*, and *on='userId'* denotes which column to join on.

We can see that the combined DataFrame contains the columns from both *adclicksDF* and *buyclicksDF:*

```
In [13]: mergeDF.head(5)
```

Out[13]:

	timestamp_x	txId_x	userSessionId_x	teamId	userId	adId	adCategory	timestamp_y	txId_y	userSessionId_y	team	buyId	price
0	2016-05-26 15:13:22	5974	5809	27	611	2	electronics	2016-05-30 13:06:54	11058	9769	27	1	2.0
1	2016-05-26 15:13:22	5974	5809	27	611	2	electronics	2016-06-03 18:36:54	17005	15910	27	4	10.0
2	2016-05-26 15:13:22	5974	5809	27	611	2	electronics	2016-06-07 12:06:54	22930	20644	27	5	20.0
3	2016-05-26 15:13:22	5974	5809	27	611	2	electronics	2016-06-11 02:06:54	29101	26524	27	4	10.0
4	2016-05-26 15:13:22	5974	5809	27	611	2	electronics	2016-06-13 02:36:54	32796	26524	27	4	10.0