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

Step 1. Open a terminal shell. Open your local terminal shell and optionally go to your *big-data-3* directory.

```
PS C:\Users\                \Desktop\coursera\big-data-3>
```

Step 2. Start Docker. Make sure to start Docker by opening Docker Desktop.

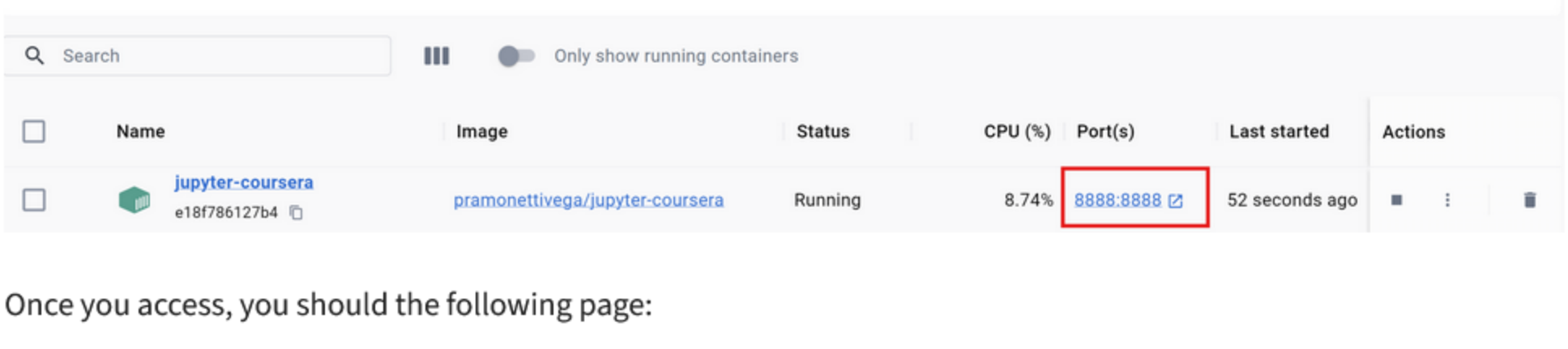
Once you have started Docker, go back to your terminal and run *docker pull pramonettivega/jupyter-coursera:latest* to pull a Docker image for this activity.

```
docker pull pramonettivega/jupyter-coursera:latest
```

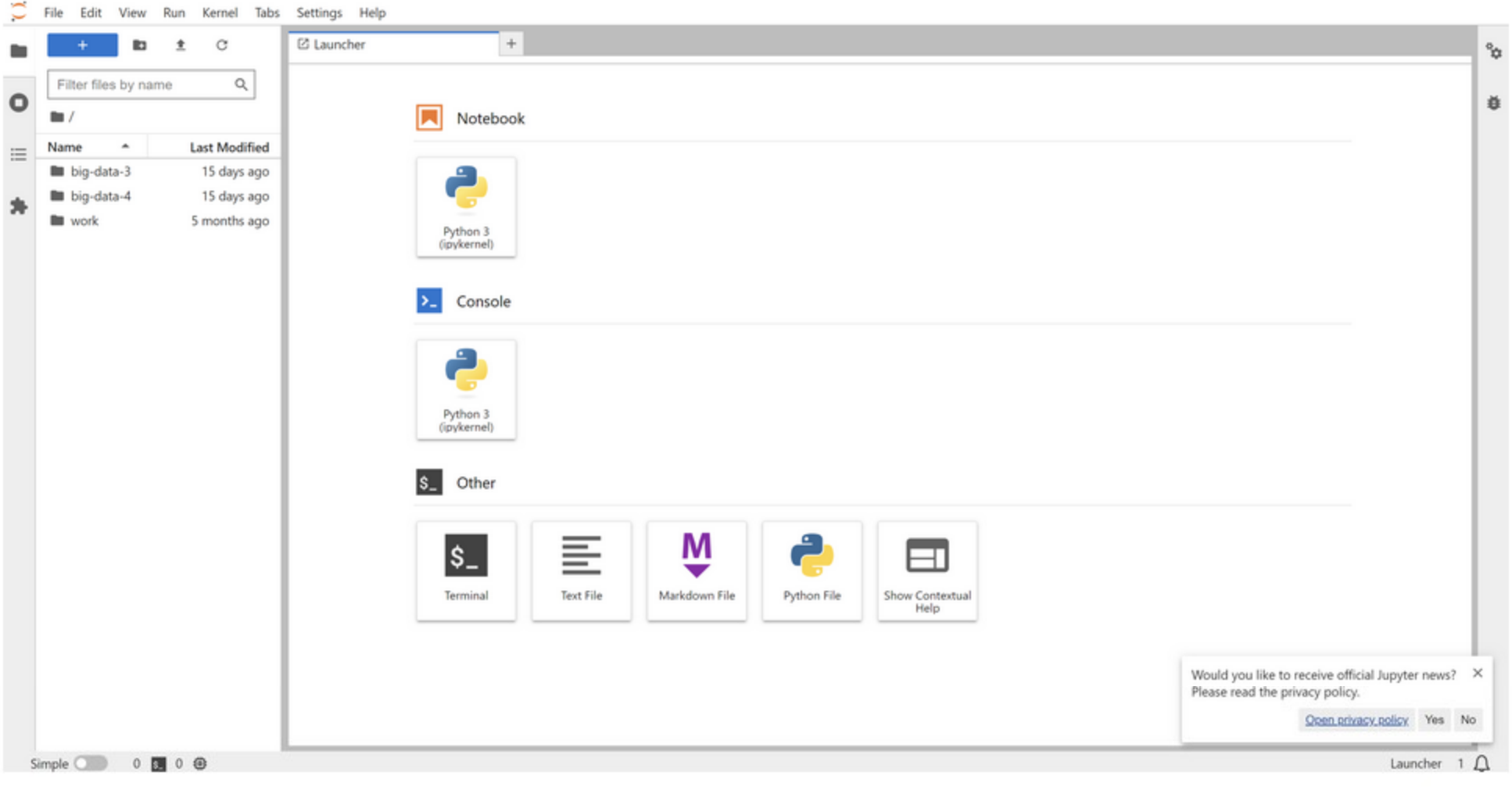
Step 3. Run the container and access Jupyter. Run *docker run -p 8888:8888 pramonettivega/jupyter-coursera* to start the container.

```
docker run --name jupyter-coursera -p 8888:8888 pramonettivega/jupyter-coursera
```

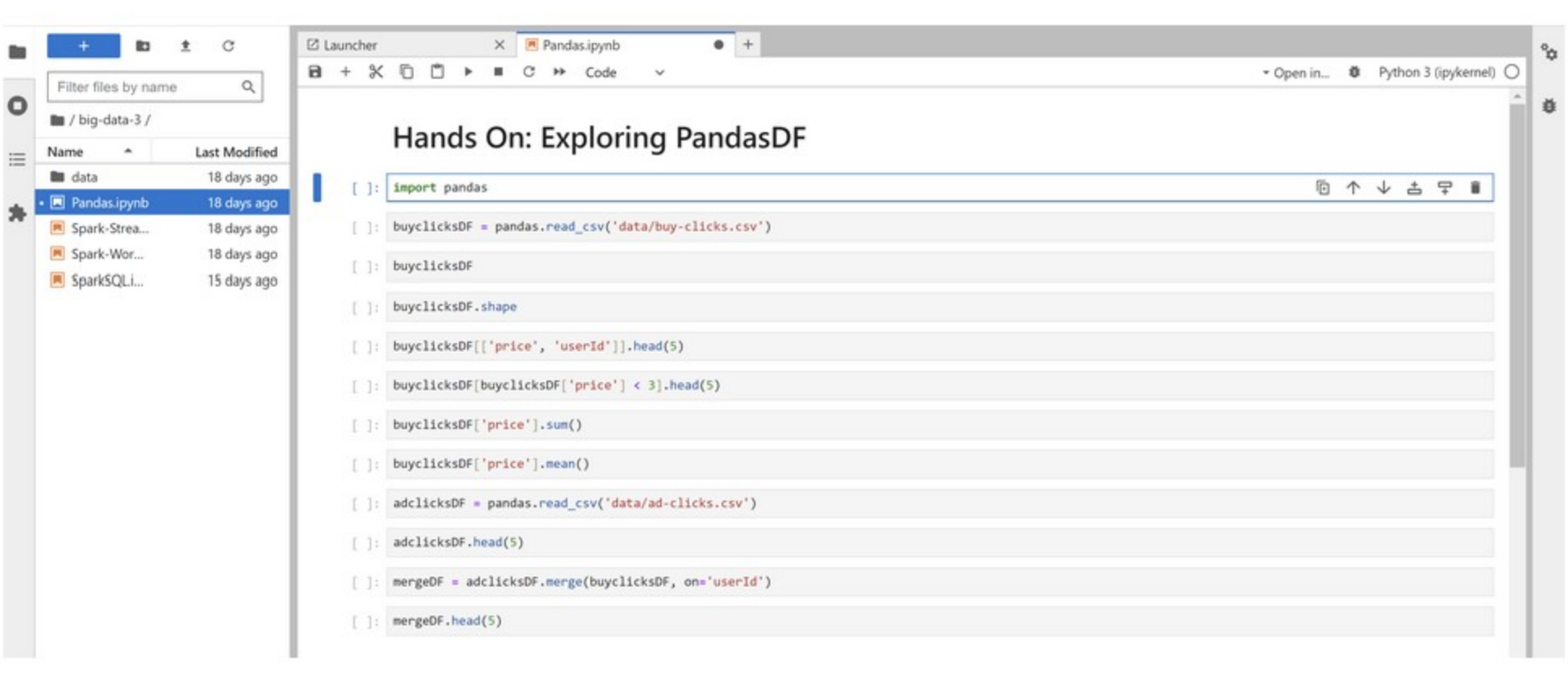
When Jupyter starts running, click on the port to access JupyterLab in your browser:



Once you access, you should the following page:



Step 4. Open your notebook. Double click on the *big-data-3* folder and then open the *Pandas.ipynb* notebook.



Step 5. Load Pandas and Read a CSV file into a DataFrame. We first load the Pandas library:

```
[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:

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

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

Step 6. View the contents and shape of a DataFrame. We can view the contents of the DataFrame by executing the variable:

```
[3]: buyclicksDF
```

[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
4	2016-05-26 17:06:54	6093	5709	11	2222	5	20.0
...	...	...	...	...	...	...	...

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

4	2016-05-26 17:06:54	6093	5709	11	2222	5	20.0
...	...	...	...	...	...	...	...
2942	2016-06-16 10:36:54	39838	34373	35	305	0	1.0
2943	2016-06-16 10:36:54	39839	37360	168	2016	1	2.0

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

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

[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:

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

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

Step 7. Filter rows and columns of a DataFrame. We can view only the *price* and *userId* columns of the DataFrame:

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

[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:

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

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

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

We can also calculate the average price:

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

A complete list of operations for Pandas DataFrames is at https://pandas.pydata.org/docs/reference/frame.html

Step 9. Combine two DataFrames. We can combine two DataFrames on a single column. First, we will load *ad-clicks.csv* into a new DataFrame:

```
[11]: adclicksDF = pandas.read_csv('data/ad-clicks.csv')
```

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

```
[12]: adclicksDF.head(5)
```

[12]:

	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:

```
[13]: 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:*

```
[14]: mergeDF.head(5)
```

[14]:

	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

Step 10. Exiting the container. To exit JupyterLab, simply close the tab in your browser. To stop the container, go to Docker Desktop and click on the *stop* button. We recommend not to delete the container, as this container will be used for multiple activities across this specialization.