PYTHON

2 Easy Ways to Get Tables From a Website with Pandas

An overview of pd.read_html and pd.read_clipboard

Byron Dolon
May 16 · 5 min read ★

Image courtesy of the girlfriend's art skills

The **pandas** library is well known for its easy-to-use data analysis capabilities. It's equipped with advanced indexing, DataFrame joining and data aggregation features. Pandas also has a **comprehensive I/O API** that you can use to input data from various sources and output data to various formats.

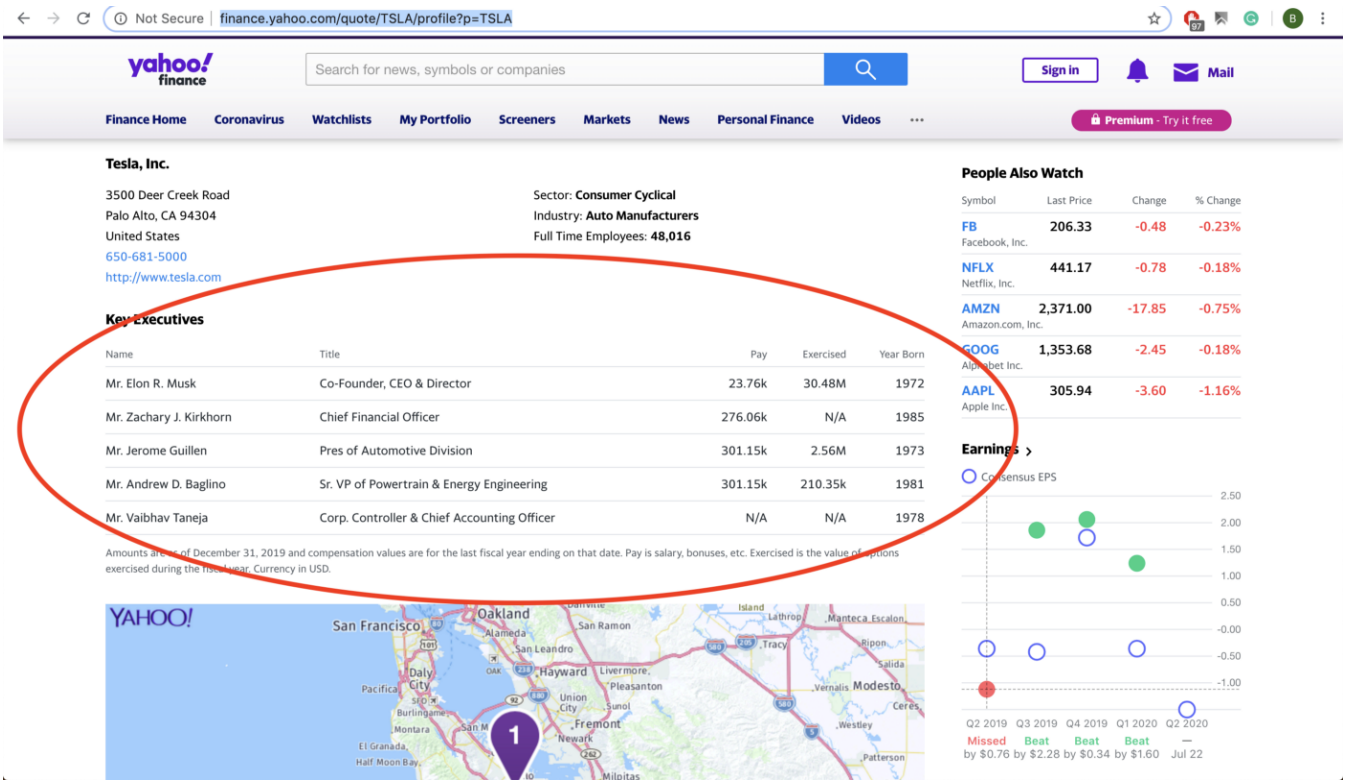
There are many occasions when you just need to get a table from a website to use in your analysis. Here's a look at how you can use the pandas read_html and read_clipboard to get tables from websites with just a couple lines of code.

Note, before trying any of the code below, don't forget to import pandas.

```
import pandas as pd
```

1. pandas.read_html()

Let's try getting this table with key Tesla executives for this example:



Yahoo Finance table of Elon Musk and other Tesla executives information

The read_html function has this description:

> *Read HTML tables into a* `list` *of* `DataFrame` *objects.*

The function searches for HTML <table> related tags on the input (URL) you provide. It always returns a **list**, even if the site only has one table. To use the function, all you need to do is put the URL of the site you want as the first argument of the function. Running the function for the Yahoo Finance site looks like this:

```
pd.read_html('https://finance.yahoo.com/quote/TSLA/profile?p=TSLA')
```

```
Out[13]: [                    Name                                  Title  \
```

```
0          Mr. Elon R. Musk                    Co-Founder, CEO & Director
1   Mr. Zachary J. Kirkhorn                       Chief Financial Officer
2        Mr. Jerome Guillen                   Pres of Automotive Division
3     Mr. Andrew D. Baglino     Sr. VP of Powertrain & Energy Engineering
4        Mr. Vaibhav Taneja   Corp. Controller & Chief Accounting Officer

       Pay Exercised  Year Born
0   23.76k     30.48M       1972
1  276.06k        NaN       1985
2  301.15k      2.56M       1973
3  301.15k    210.35k       1981
4      NaN        NaN       1978  ]
```

Raw output of read_html

To get a DataFrame from this list, you only need to make one addition:

```
pd.read_html('https://finance.yahoo.com/quote/TSLA/profile?p=TSLA')
[0]
```

Adding the '[0]' selects the first element in the list. There is only one element in our list, and it is a DataFrame object. Running this code gives you this output:

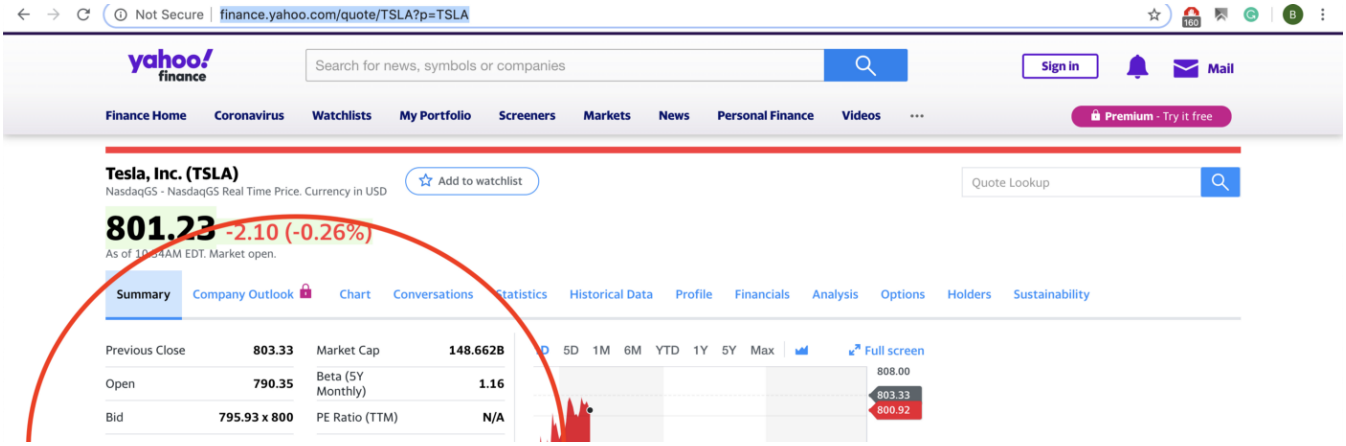
Out[14]:

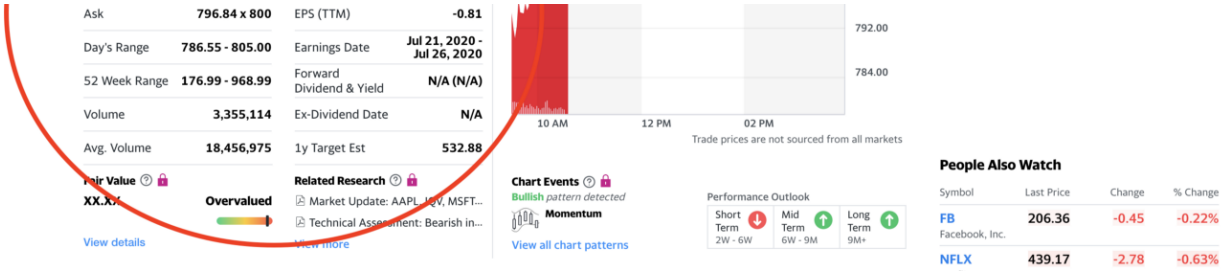
	Name	Title	Pay	Exercised	Year Born
0	Mr. Elon R. Musk	Co-Founder, CEO & Director	23.76k	30.48M	1972
1	Mr. Zachary J. Kirkhorn	Chief Financial Officer	276.06k	NaN	1985
2	Mr. Jerome Guillen	Pres of Automotive Division	301.15k	2.56M	1973
3	Mr. Andrew D. Baglino	Sr. VP of Powertrain & Energy Engineering	301.15k	210.35k	1981
4	Mr. Vaibhav Taneja	Corp. Controller & Chief Accounting Officer	NaN	NaN	1978

Output of read_html with list index selection

. . .

Now, let's try getting this table with summary statistics for the Tesla stock:





Yahoo Finance summary table for Tesla stock

We'll try the same code as before:

```
pd.read_html('https://finance.yahoo.com/quote/TSLA?p=TSLA')
```

```
Out[20]: [                0                1
          0  Previous Close           803.33
          1            Open           790.35
          2             Bid   799.72 x 800
          3             Ask   801.22 x 800
          4    Day's Range  786.55 - 805.00
          5  52 Week Range  176.99 - 968.99
          6          Volume          3414278
          7    Avg. Volume         18456975,
                           0                1
          0      Market Cap          148.095B
          1    Beta (5Y Monthly)          1.16
          2    PE Ratio (TTM)           NaN
          3         EPS (TTM)         -0.81
          4    Earnings Date  Jul 22, 2020 - Jul 27, 2020
          5  Forward Dividend & Yield  N/A (N/A)
          6    Ex-Dividend Date           NaN
          7       1y Target Est        532.88]
```

Table #1

Table #2

Raw output of read_html #2

It looks like we got all the data we need, but there are two elements in the list now. This is because the table we see in the screenshot above is separated into two different tables in the HTML source code. We could do the same index trick as before, but if you want to combine both tables into one, all you need to do is **concatenate** the two list elements like this:

```
separate = pd.read_html('https://finance.yahoo.com/quote/TSLA?
p=TSLA')
pd.concat([separate[0],separate[1]])
```

```
Out[19]:
```

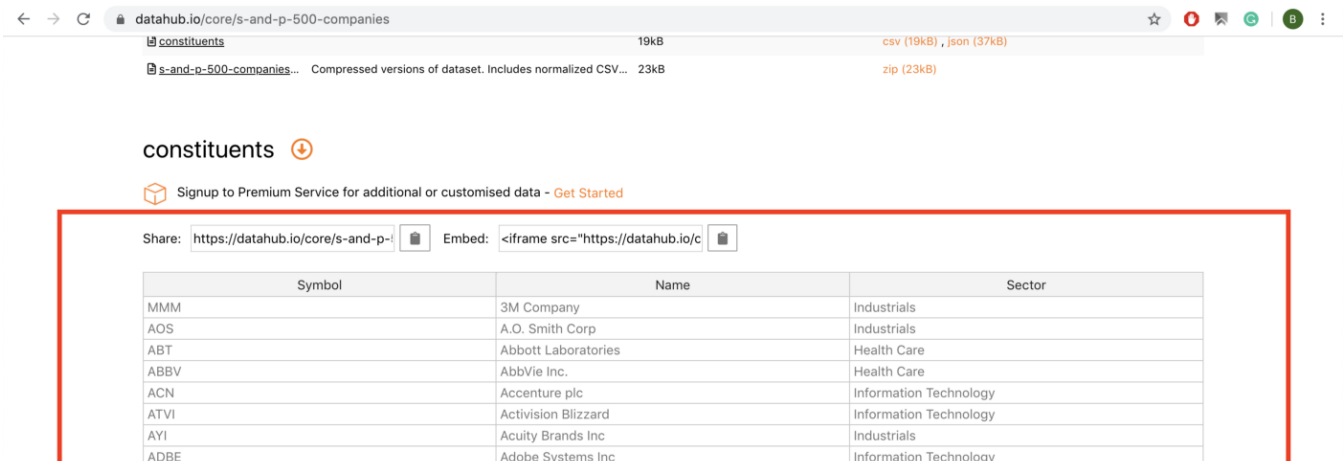
0	Previous Close	803.33
1	Open	790.35
2	Bid	801.59 x 800
3	Ask	801.18 x 800
4	Day's Range	786.55 - 805.00
5	52 Week Range	176.99 - 968.99
6	Volume	3310216
7	Avg. Volume	18456975
0	Market Cap	148.497B
1	Beta (5Y Monthly)	1.16
2	PE Ratio (TTM)	NaN
3	EPS (TTM)	-0.81
4	Earnings Date	Jul 22, 2020 - Jul 27, 2020
5	Forward Dividend & Yield	N/A (N/A)
6	Ex-Dividend Date	NaN
7	1y Target Est	532.88

Output of pd.concat of two list elements from read_html

There's plenty more you could do to process this data for analysis- just renaming the column headers would be a great start. But getting this far took about 12 seconds, which is great if you just need test data from a static site.

2. pandas.read_clipboard()

Here's a table with S&P 500 company information we can try to get:



AAP	Advance Auto Parts	Consumer Discretionary
AMD	Advanced Micro Devices Inc	Information Technology
AES	AES Corp	Utilities
AET	Aetna Inc	Health Care
AMG	Affiliated Managers Group Inc	Financials
AFL	AFLAC Inc	Financials
A	Agilent Technologies Inc	Health Care
APD	Air Products & Chemicals Inc	Materials
ALK	Alaska Air Group Inc	Industrials
AKAM	Akamai Technologies Inc	Information Technology

This is a preview version. There might be more data in the original version.

S&P500 information from datahub.io

The data is distributed under an ODC license, which means it's free to share, create, and adapt the data on the site. I was initially going to use this site for my read_html example, but after I ran the function for the third time, I was greeted with an error.

```
pd.read_html('https://datahub.io/core/s-and-p-500-companies')
```

```
---------------------------------------------------------------------------
HTTPError                                 Traceback (most recent call last)
<ipython-input-21-0c097067f0f9> in <module>
      2 import pandas as pd
      3
----> 4 pd.read_html('https://datahub.io/core/s-and-p-500-companies')

~/opt/anaconda3/lib/python3.7/site-packages/pandas/io/html.py in read_html(io, match, flavor, header, index_col, skip
rows, attrs, parse_dates, thousands, encoding, decimal, converters, na_values, keep_default_na, displayed_only)
   1103         na_values=na_values,
   1104         keep_default_na=keep_default_na,
-> 1105         displayed_only=displayed_only,
   1106     )

~/opt/anaconda3/lib/python3.7/site-packages/pandas/io/html.py in _parse(flavor, io, match, attrs, encoding, displayed
_only, **kwargs)
    910             break
    911     else:
--> 912         raise_with_traceback(retained)
    913
    914     ret = []

~/opt/anaconda3/lib/python3.7/site-packages/pandas/compat/__init__.py in raise_with_traceback(exc, traceback)
     44     if traceback == Ellipsis:
     45         _, _, traceback = sys.exc_info()
---> 46     raise exc.with_traceback(traceback)
     47
     48

HTTPError: HTTP Error 403: Forbidden
```

HTTP 403 error from trying to read_html datahub.io

The HTTP 403 error happens when you try to access a webpage and the site successfully understands your request, but will not authorize it. This can occur when you try to access a site that you don't have access to.

In this case, you can access the site from your browser, but the site won't let you access it from a script. Many sites have rules about scraping on their "robots.txt" file, which you can find by appending "/robots.txt" after the top-level domain of the site's URL. For example, Facebook's would be "https://facebook.com/robots.txt".

To avoid an error like this, you might be tempted to copy the data onto an Excel sheet, then load that file with the pd.read_excel function.

Instead, pandas offers a feature that allows you to copy data directly from your clipboard! The read_clipboard function has this description:

> *Read text from clipboard and pass to* ***read_csv***

If you've used pandas before, you've probably used pd.read_csv to get a local file for use in data analysis. The read_clipboard function just takes the text you have copied and treats it as if it were a csv. It will return a **DataFrame** based on the text you copied.

To get the S&P 500 table from datahub.io, select and copy the table from your browser, then enter the code below.

```
pd.read_clipboard()
```

Out[23]:

	A	Agilent Technologies Inc	Health Care
0	AAL	American Airlines Group	Industrials
1	AAP	Advance Auto Parts	Consumer Discretionary
2	AAPL	Apple Inc.	Information Technology
3	ABBV	AbbVie Inc.	Health Care
4	ABC	AmerisourceBergen Corp	Health Care
...	...	...	...
499	XYL	Xylem Inc.	Industrials
500	YUM	Yum! Brands Inc	Consumer Discretionary
501	ZBH	Zimmer Biomet Holdings	Health Care
502	ZION	Zions Bancorp	Financials
503	ZTS	Zoetis	Health Care

Output of pd.read_clipboard

Perfect! We've got a ready to use DataFrame, exactly as seen from the website!

. . .

You can check out the read_html and read_clipboard documentation for more information. There, you'll find that there's a lot more you can do with these functions to customize exactly how you want to input data from websites.

Good luck with your I/O!

Python Pandas Programming Data Analysis Data Science