**21 Pandas Tips Every Data Scientist Should Know**

**Useful tips and tricks to become a master Pandas user**

In general, if you have an analytical task to accomplish and you don’t know how to do it? Probably, It already faced other data analysts before you. So before you start reinventing the wheel, a quick Google search can help you save time and write clean code.

In this article, we will cover 21 Pandas’ tips and tricks that will help you keep your code clean and don’t reinvent the wheel. I am sure there are other interesting tips and tricks, so if you know any additional tips feel free to leave them in a comment.

Most of the tips are from

[Khuyen Tran’s](https://medium.com/u/84a02493194a?source=post_page-----97c930638e9c--------------------------------)

LinkedIn posts.

**1: Print DataFrame in Markdown-friendly format**

Markdown is a lightweight markup language that is used to create formatted text using a plain-text editor. Sometimes, you might want to include a table in a markdown, such as GitHub README or Latex.

If you want to print a DataFrame in markdown format, use to\_markdown() function.

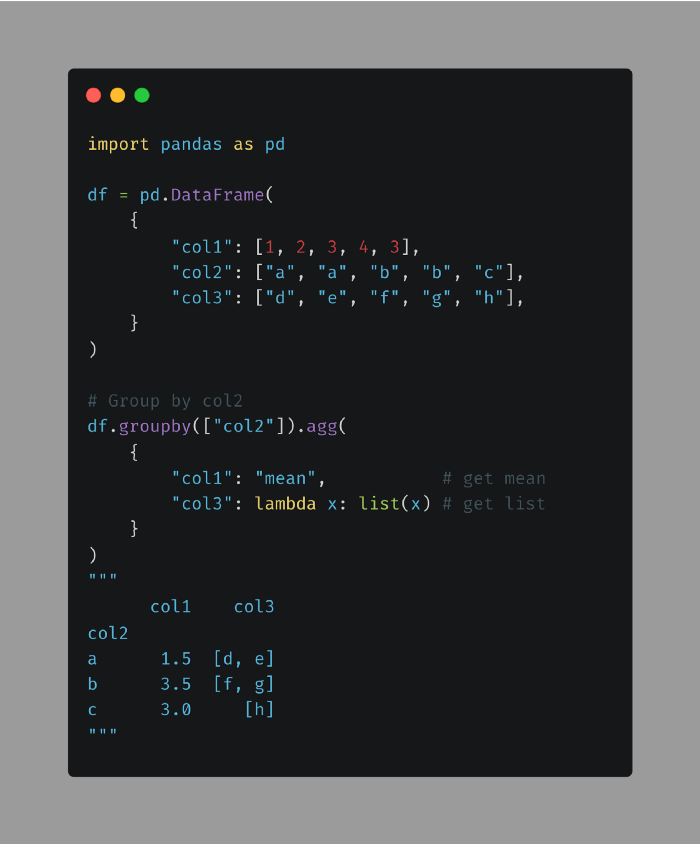


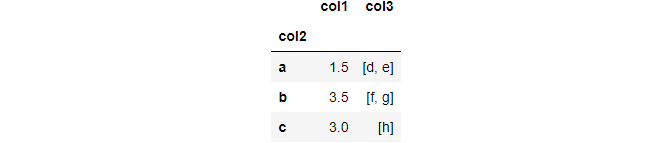
| | a | b |  
|---:|----:|----:|  
| 0 | 1 | 5 |  
| 1 | 2 | 6 |  
| 2 | 3 | 7 |  
| 3 | 4 | 8 |

+----+-----+-----+  
| | a | b |  
+====+=====+=====+  
| 0 | 1 | 5 |  
+----+-----+-----+  
| 1 | 2 | 6 |  
+----+-----+-----+  
| 2 | 3 | 7 |  
+----+-----+-----+  
| 3 | 4 | 8 |  
+----+-----+-----+

**2: Group rows into a list**

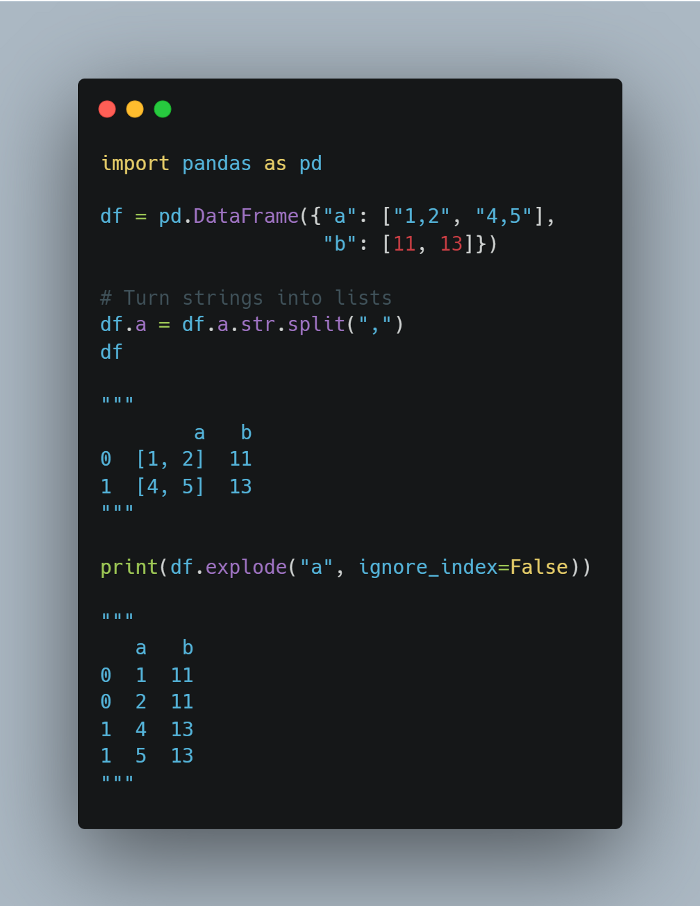
It is common to use groupby to get the statistics of rows in the same group such as count, mean, median, etc. If you want to group rows into a list instead, use “lambda x: list(x)”.

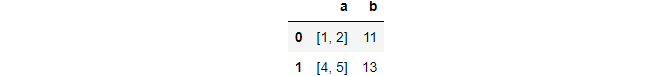


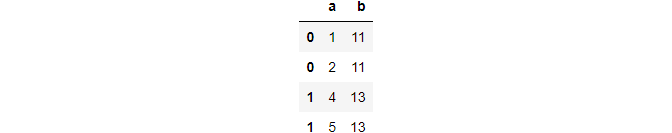


**3: DataFrame.explode()**

When working with a DataFrame, if you want to turn a string into a list and then split elements in a list into multiple rows, use the combination of str.split() and explode().



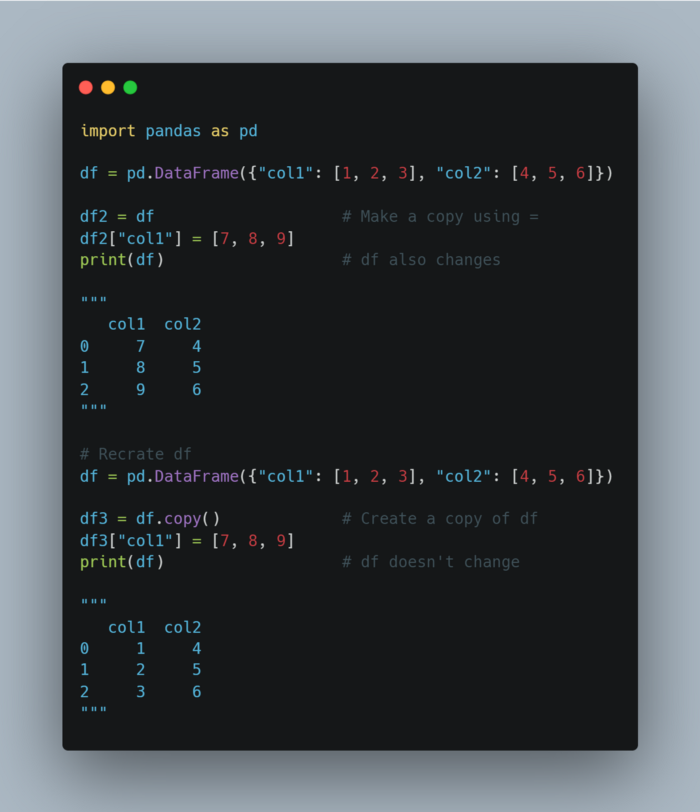


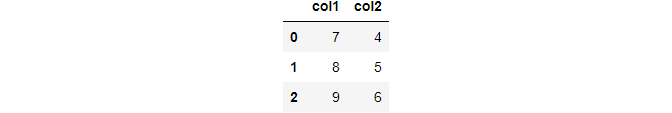


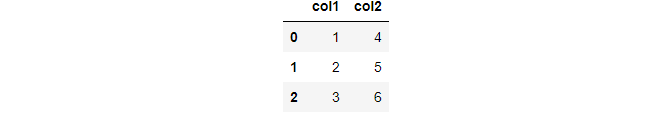
**4: DataFrame.copy()**

Have you ever tried to make a copy of a DataFrame user “=”? You will not get a copy but a reference to the original DataFrame. Thus, changing the new DataFrame will also change the original DataFrame.

A better way to make a copy is to use df.copy(). Now, changing the copy will not affect the original DataFrame.



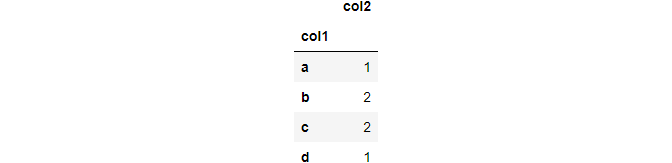




**5: Groupby().count vs Groupby( ).size**

* If you want to get the count of elements in one column of a Pandas DataFrame, use groupby and count.
* If you want to get the size of groups composed of 2 or more columns, use groupby and size instead.

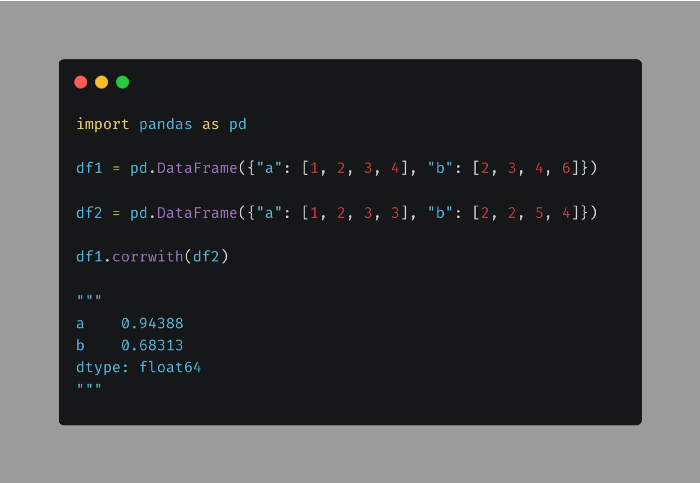




col1 col2  
a S 1  
b M 1  
 S 1  
c L 2  
d L 1  
dtype: int64

**6: Correlation**

If you want to compute the correlation between rows or columns of two DataFrame, use .corrwith().

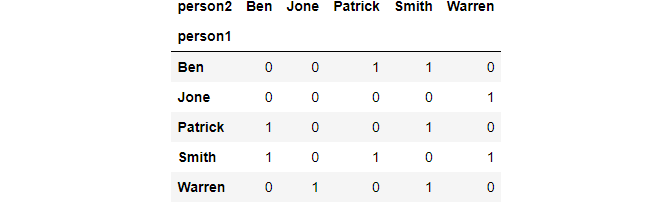


a 0.94388  
b 0.68313  
dtype: float64

**7: Cross-Tabulation**

Cross-tabulation allows you to analyze the relationship between multiple variables. To turn a Pandas DataFrame into a cross-tabulation, use pandas.crosstab().

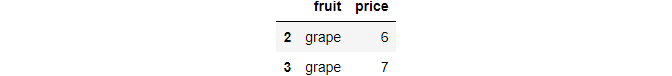


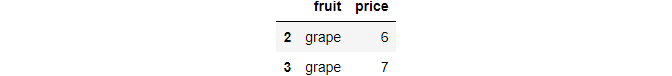


**8: DataFrame.query()**

It can be lengthy to filter columns of pandas DataFrame using brackets. To shorten the filtering statements, use df.query() instead.



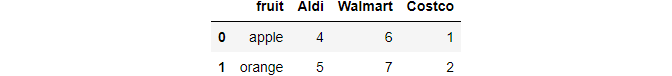


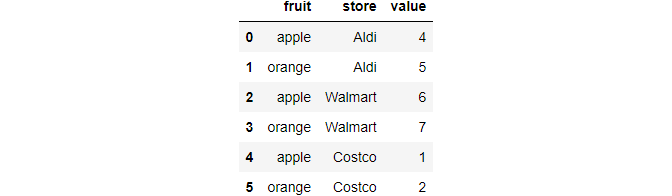


**9: Unpivot DataFrame**

* If you want to unpivot a DataFrame from wide to long format, use pandas.melt().
* For example, you can use pandas.melt() to turn multiple columns (“Aldi”, “Walmart”, “Costco”) into values of one column (“store”).



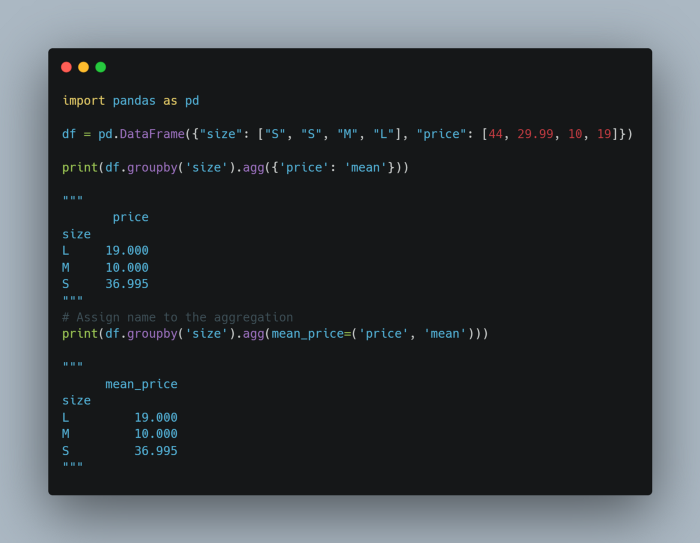


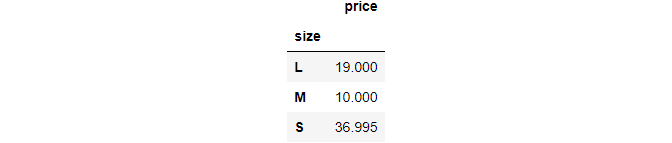


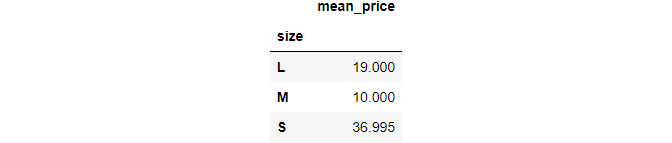
**10: Rename aggregated column**

By default, aggregating a column returns the name of that column.

If you want to assign a new name to the aggregation, use name = (column, agg\_method).







**11: Normalized Value Counts**

* If you want to get the count of a value in a column, use value\_counts.
* However, if you want to get the percentage of a value in a column, add normalize=True to value\_counts.

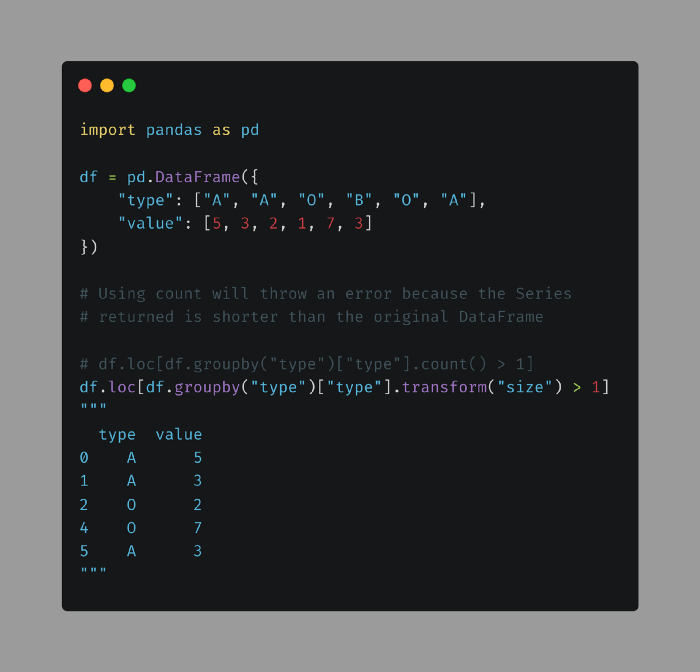


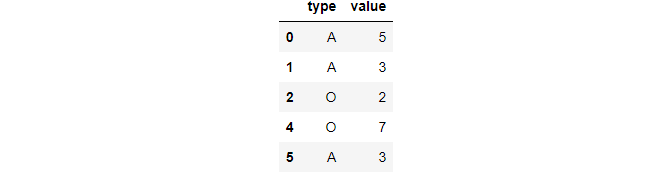
S 4  
M 2  
L 1  
XL 1  
dtype: int64

S 0.500  
M 0.250  
L 0.125  
XL 0.125  
dtype: float64

**12: df.transform() instead of df.count()**

* To filter pandas DataFrame based on the occurrences of categories, you might attempt to use df.groupby and df.count. However, since the Series returned by the count method is shorter than the original DataFrame, you will get an error when filtering.
* Instead of using count, use transform. This method will return the Series with the same length as the original DataFrame. Now you can filter without encountering any errors.

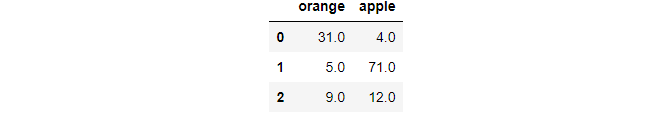




**13: Fill in Null Values**

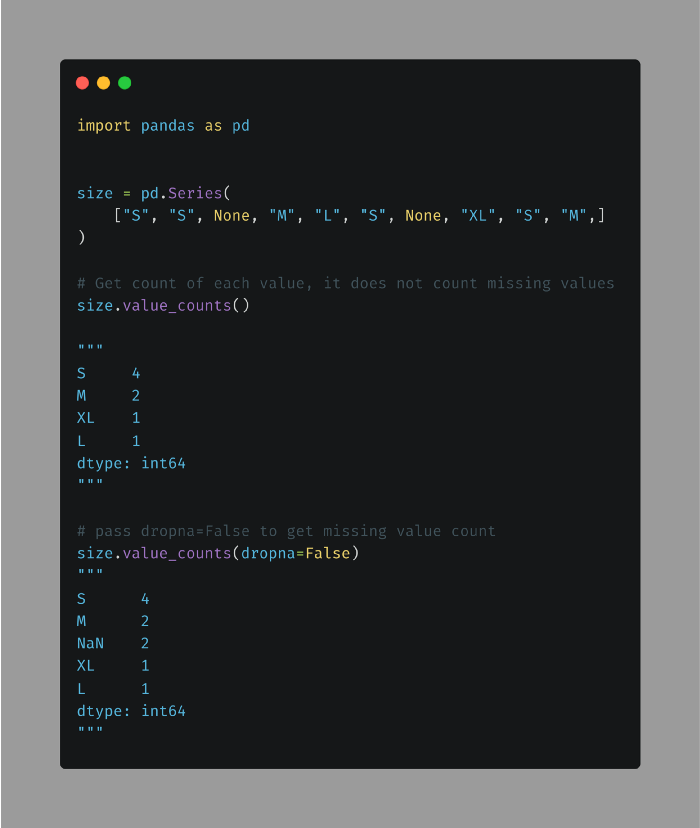
* If you want to fill null values in one DataFrame with non-null values at the same locations from another DataFrame, use pandas.DataFrame.combine\_first.
* In the code below, the values at the first row of store1 are updated with the values at the first row of store2.





**14: Value Counts Missing Values**

By default, pandas.value\_counts() ignore missing values. Pass dropna=False to make it count missing values.



**15: Filter Columns in DataFrame**

If you want to filter columns of a pandas DataFrame based on characters in their names, use DataFrame.filter. This can be handy if you create dummy variables and you want to select columns based on the prefix.



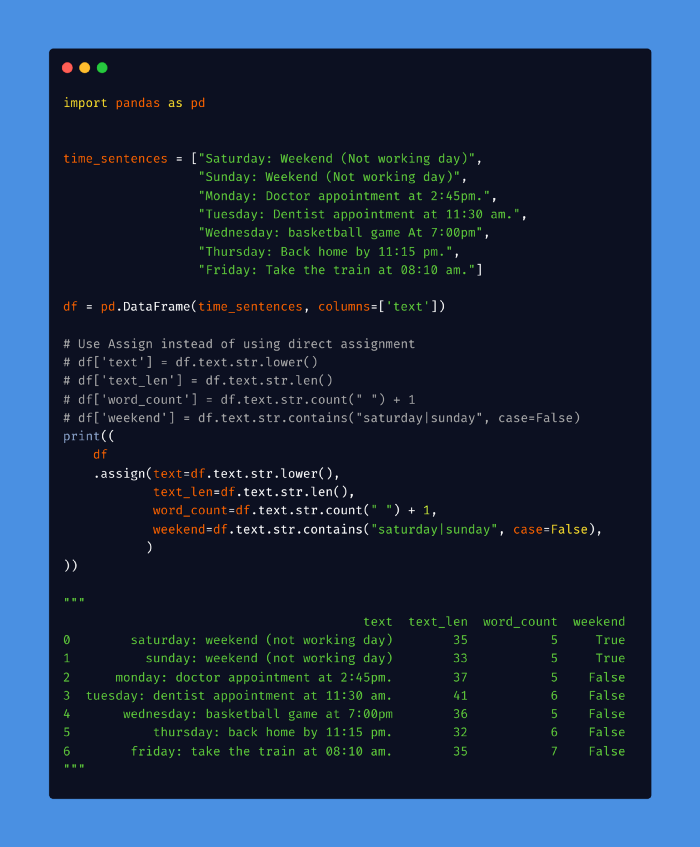
**16: Convert Data Types Automatically**

If you don’t know the dtypes of columns in your DataFrame, you can use convert\_dtypes() to quickly convert columns to the best possible types.



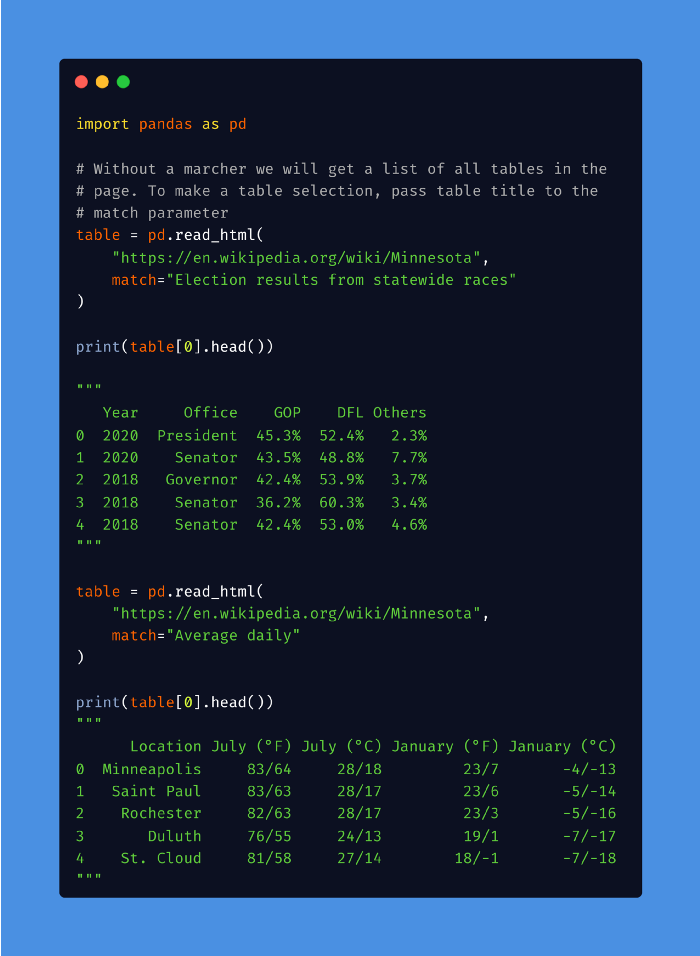
**17: Assign new columns to a DataFrame**

Use Dataframe.assign() method to assign new columns to your DataFrame, Dataframe.assign() returns a new object (a copy) with the new columns added to the original ones. Existing columns that are re-assigned will be overwritten.



**18: Read HTML Tables**

.read\_html() can be useful for quickly incorporating tables from various websites without figuring out how to scrape the site’s HTML.

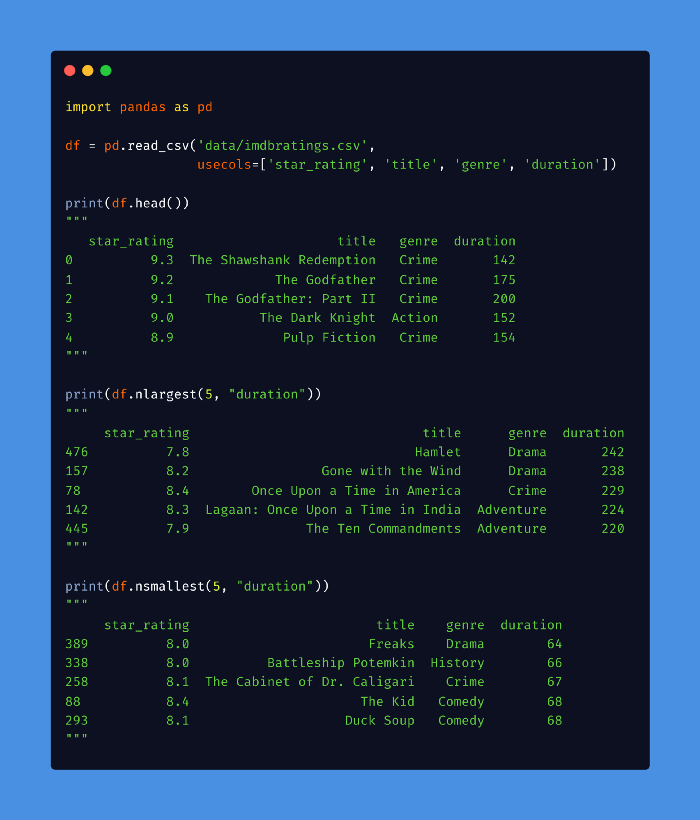


Year Office GOP DFL Others  
0 2020 President 45.3% 52.4% 2.3%  
1 2020 Senator 43.5% 48.8% 7.7%  
2 2018 Governor 42.4% 53.9% 3.7%  
3 2018 Senator 36.2% 60.3% 3.4%  
4 2018 Senator 42.4% 53.0% 4.6%Location July (°F) July (°C) January (°F) January (°C)  
0 Minneapolis 83/64 28/18 23/7 −4/−13  
1 Saint Paul 83/63 28/17 23/6 −5/−14  
2 Rochester 82/63 28/17 23/3 −5/−16  
3 Duluth 76/55 24/13 19/1 −7/−17  
4 St. Cloud 81/58 27/14 18/−1 −7/−18

**19: ‘nlargest’ and ‘nsmallest ‘**

Use .nlargest() and .nsmallest() to sort columns inDataFrame based on a specific column instead of using .sort\_values()

Data Link: [IMDB Rating](https://raw.githubusercontent.com/fares-ds/data_analysis_in_python_with_pandas/master/pandas_basics/data/imdbratings.csv)



**20: Create a Rank Column**

Pandas DataFrame.rank() method returns a rank of every respective index of a series passed. The rank is returned based on position after sorting.

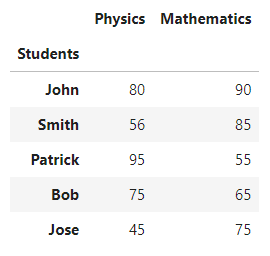
In the following example, a new Rank column is created which ranks the Student by their Marks.

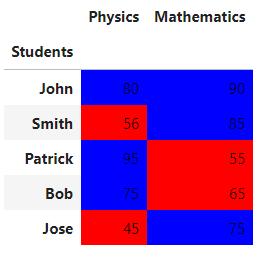


Students Marks  
0 John 80  
1 Smith 56  
2 Patrick 95  
3 Bob 75  
4 Jose 45Students Marks Rank  
0 John 80 2.0  
1 Smith 56 4.0  
2 Patrick 95 1.0  
3 Bob 75 3.0  
4 Jose 45 5.0

**21: Color Values in DataFrame**

Color styling adds more readability to the end user. Pandas have the style property that follows us to apply different styles to DataFrames.





**Conclusion:**

Pandas is a fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool, built on top of the Python programming language. Mastering Pandas will take your analysis skills to the next level and knowing best practices will save you a lot of time and energy.