

3 Easy Tricks to Get Started with Python (and Ditch Excel!)

An Introduction to Pandas for Excel Power Users



Nik Piepenbreier

May 25 · 7 min read ★

3 Easy Tricks to Get Started with Python (and Ditch Excel!)



Let's learn some Python and Pandas! Source: Nik Piepenbreier

So you've taken the mental leap and want to learn Python — that's awesome! **But where to start?**

Let me guide you through tasks you already know how to do in Excel and how to do them in Python!

You'll be using Pandas, the primary data analysis library in Python.

If you need to install Pandas, you can do this using pip or conda:

```
pip install pandas
#or
conda install pandas
```

Pandas loads data into *dataframes*, which you can think of as Excel sheets.

Let's load a dataset into a dataframe. You can accomplish this by using the following code. We'll also explore the first five rows by using the Pandas head method:

```
1 import pandas as pd
2
3 df = pd.read_excel('https://github.com/datagy/pivot_table_pandas/raw/master/sample_pivot.xlsx')
4 print(df.head())
5
6 # Returns:
7 #      Date Region      Type  Units  Sales
8 #0 2020-07-11   East  Children's Clothing   18.0   306
9 #1 2020-09-23  North  Children's Clothing   14.0   448
10 #2 2020-04-02  South   Women's Clothing   17.0   425
11 #3 2020-02-28   East  Children's Clothing   26.0   832
12 #4 2020-03-19   West   Women's Clothing    3.0    33
```

datagy-bye-excel01.py hosted with ❤ by GitHub

[view raw](#)

Generating our dataframe with Pandas. Source: Nik Piepenbreier

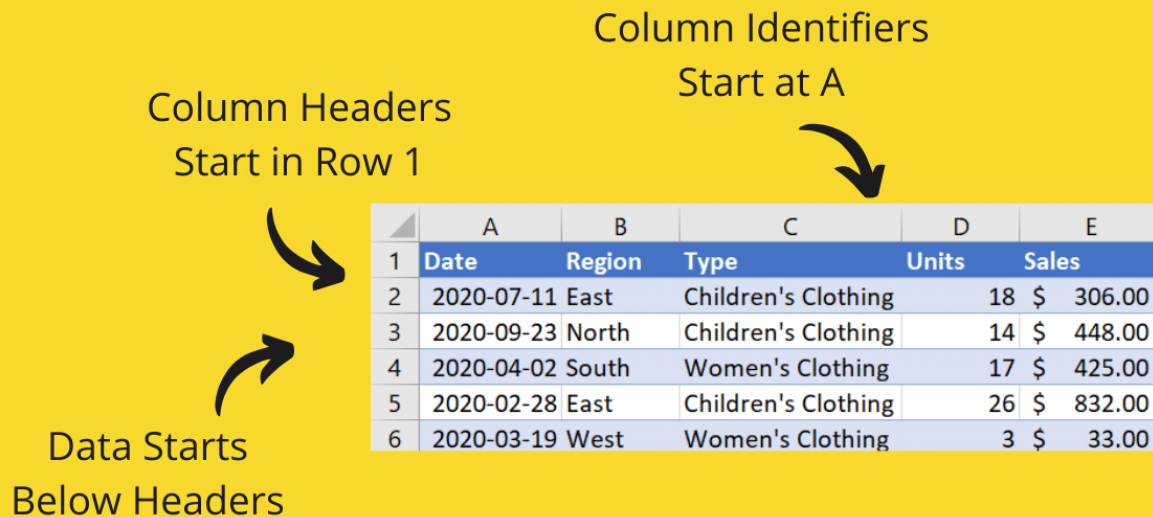
To follow along with the tutorial in Excel as well, the file [can be found here](#).

There are a number of differences between Excel sheets and Pandas dataframes. Let's take a quick look!

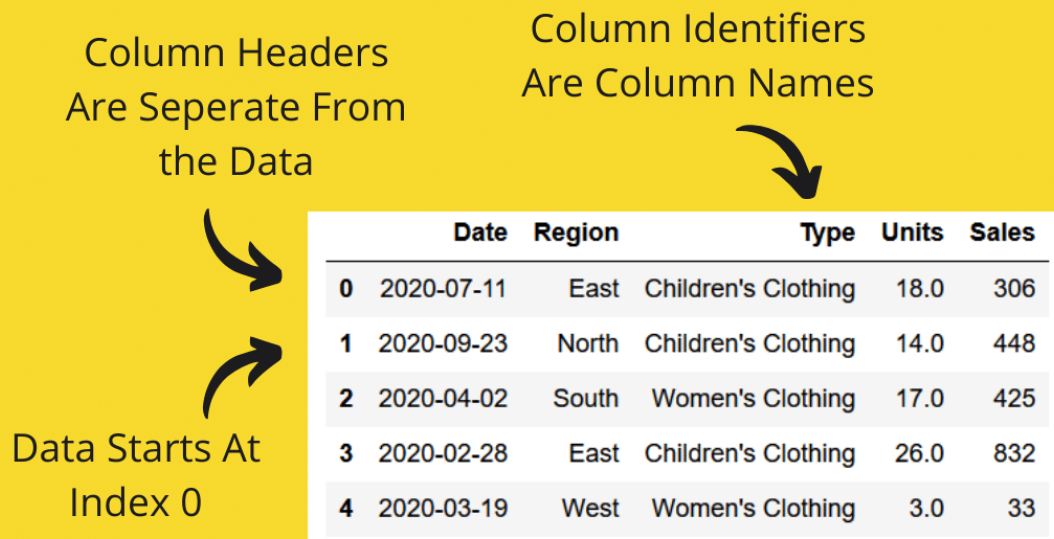
**Excel and Pandas
Compared**
datagy.io



Excel Worksheet



Pandas Dataframe



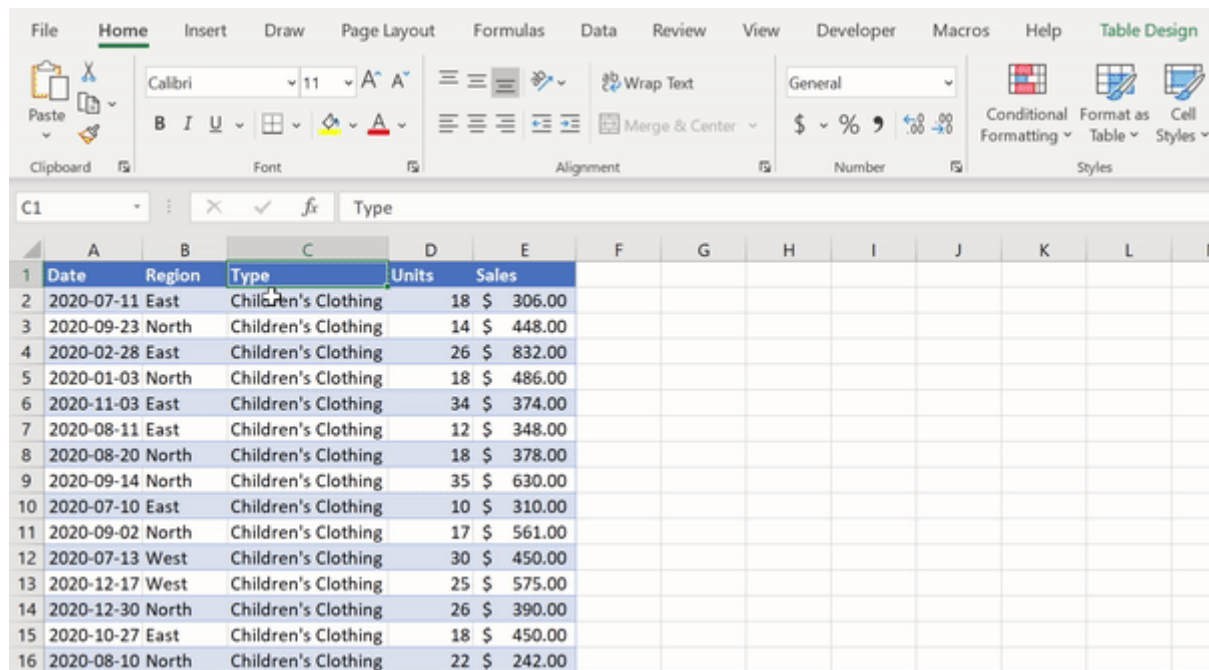
Comparing Excel worksheets and Pandas dataframes. Source: Nik Piepenbreier

Ok, now let's get started!

Filtering, Sort, & Reorder Columns

Excel is a much more visual tool, which makes it easy to click a button that abstracts the function behind what you want to accomplish.

Sort by a Single Column



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Date	Region	Type	Units	Sales								
2	2020-07-11	East	Children's Clothing	18	\$ 306.00								
3	2020-09-23	North	Children's Clothing	14	\$ 448.00								
4	2020-02-28	East	Children's Clothing	26	\$ 832.00								
5	2020-01-03	North	Children's Clothing	18	\$ 486.00								
6	2020-11-03	East	Children's Clothing	34	\$ 374.00								
7	2020-08-11	East	Children's Clothing	12	\$ 348.00								
8	2020-08-20	North	Children's Clothing	18	\$ 378.00								
9	2020-09-14	North	Children's Clothing	35	\$ 630.00								
10	2020-07-10	East	Children's Clothing	10	\$ 310.00								
11	2020-09-02	North	Children's Clothing	17	\$ 561.00								
12	2020-07-13	West	Children's Clothing	30	\$ 450.00								
13	2020-12-17	West	Children's Clothing	25	\$ 575.00								
14	2020-12-30	North	Children's Clothing	26	\$ 390.00								
15	2020-10-27	East	Children's Clothing	18	\$ 450.00								
16	2020-08-10	North	Children's Clothing	22	\$ 242.00								

Sorting a single columns in Excel. Source: Nik Piepenbreier

For example, if you wanted to sort a column in Excel, you could simply:

- Select the Data tab,
- Highlight the column you want to sort, and
- Click Sort A to Z or Sort Z to A.

To do this in Pandas, you would write:

```
1 df = df.sort_values(by = 'Type', ascending=False)
2
3 # Alternatively, you could write ascending=True if you wanted
4 # to sort the values in ascending order
```

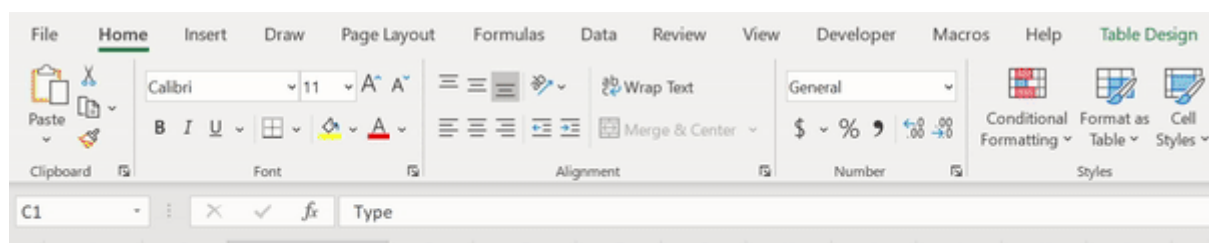
datagy-bye-excel02.py hosted with ❤ by GitHub

[view raw](#)

Sorting by a single column in Pandas. Source: Nik Piepenbreier

Sort by Multiple Columns

Sometimes you might want to sort by multiple columns.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Date	Region	Type	Units	Sales								
2	2020-07-11	East	Children's Clothing	18	\$ 306.00								
3	2020-09-23	North	Children's Clothing	14	\$ 448.00								
4	2020-02-28	East	Children's Clothing	26	\$ 832.00								
5	2020-01-03	North	Children's Clothing	18	\$ 486.00								
6	2020-11-03	East	Children's Clothing	34	\$ 374.00								
7	2020-08-11	East	Children's Clothing	12	\$ 348.00								
8	2020-08-20	North	Children's Clothing	18	\$ 378.00								
9	2020-09-14	North	Children's Clothing	35	\$ 630.00								
10	2020-07-10	East	Children's Clothing	10	\$ 310.00								
11	2020-09-02	North	Children's Clothing	17	\$ 561.00								
12	2020-07-13	West	Children's Clothing	30	\$ 450.00								
13	2020-12-17	West	Children's Clothing	25	\$ 575.00								
14	2020-12-30	North	Children's Clothing	26	\$ 390.00								
15	2020-10-27	East	Children's Clothing	18	\$ 450.00								
16	2020-08-10	North	Children's Clothing	22	\$ 242.00								

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Date	Region	Type	Units	Sales								
2	2020-04-02	South	Women's Clothing	17	\$ 425.00								
3	2020-03-19	West	Women's Clothing	3	\$ 33.00								
4	2020-02-05	North	Women's Clothing	33	\$ 627.00								
5	2020-01-24	South	Women's Clothing	12	\$ 396.00								
6	2020-03-25	East	Women's Clothing	29	\$ 609.00								
7	2020-04-16	South	Women's Clothing	16	\$ 352.00								
8	2020-06-12	East	Women's Clothing	35	\$ 1,050.00								
9	2020-06-16	North	Women's Clothing	34	\$ 884.00								
10	2020-06-23	West	Women's Clothing	18	\$ 288.00								
11	2020-01-04	North	Women's Clothing	19	\$ 361.00								
12	2020-06-29	East	Women's Clothing	24	\$ 504.00								
13	2020-12-20	East	Women's Clothing	30	\$ 930.00								
14	2020-01-22	North	Women's Clothing	23	\$ 276.00								
15	2020-06-06	East	Women's Clothing	28	\$ 560.00								
16	2020-09-06	East	Women's Clothing	11	\$ 363.00								

Sorting by multiple columns in Excel. Source: Nik Piepenbreier

Again, Excel makes this easy:

Click on the Data tab

- Click on Sort
- Enter the columns you want to sort by

To do this with Pandas, simply add a list to the “by” argument:

```
1 df = df.sort_values(by = ['Type', 'Region'], ascending=False)
```

datagy-bye-excel03.py hosted with ❤ by GitHub

[view raw](#)

Sorting by multiple columns in Pandas. Source: Nik Piepenbreier

Filtering Columns

Filtering columns is an easy task in Excel! Simply click on the Data tab, then Filter. This creates arrows on all the column headers. When you click on these, simply fill in your selection:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Date	Region	Type	Units	Sales								
2	2020-03-25	East	Women's Clothing	29	\$ 609.00								
3	2020-06-12	East	Women's Clothing	35	\$ 1,050.00								
4	2020-06-29	East	Women's Clothing	24	\$ 504.00								
5	2020-12-20	East	Women's Clothing	30	\$ 930.00								
6	2020-06-06	East	Women's Clothing	28	\$ 560.00								
7	2020-09-06	East	Women's Clothing	11	\$ 363.00								
8	2020-04-29	East	Women's Clothing	14	\$ 462.00								
9	2020-07-30	East	Women's Clothing	21	\$ 399.00								

10	2020-11-19 East	Women's Clothing	8	\$	256.00														
11	2020-05-12 East	Women's Clothing	29	\$	812.00														
12	2020-11-26 East	Women's Clothing	22	\$	440.00														
13	2020-03-04 East	Women's Clothing	4	\$	116.00														
14	2020-04-19 East	Women's Clothing	24	\$	672.00														
15	2020-05-07 East	Women's Clothing	5	\$	100.00														
16	2020-08-10 East	Women's Clothing	11	\$	143.00														

Filtering columns in Excel. Source: Nik Piepenbreier

In Pandas, this is just as easy:

```
1 df = df[df['Region'] == 'East']
```

datagy-bye-excel04.py hosted with ❤ by GitHub

[view raw](#)

Filtering a column in Pandas. Source: Nik Piepenbreier

What's great about this, you can also use comparison operators to select more than 10 units, or select based on multiple conditions:

- **Comparison:** You can use > (greater than), < (less than), == (equal to), >= (greater than or equal to), and <= (less than or equal to),
- **'And' Conditions:** wrap each condition in brackets and separate the brackets with an ampersand (&)
- **'Or' Conditions:** wrap each condition in brackets and separate the brackets with a pipe (|)

```
1 # Select More Than 10 Units sold:
2 df = df[df['Units'] > 10]
3
4 # Select with an AND condition:
5 # Return all units sold greater than 10 in East region
6 df = df[(df['Units'] > 10) & (df['Region'] == 'East')]
7
8 # Select with an OR condition:
9 # Return all units sold greater than 10 OR sold in East region
10 df = df[(df['Units'] > 10) | (df['Region'] == 'East')]
```

datagy-bye-excel05.py hosted with ❤ by GitHub

[view raw](#)

Different types of filters in Pandas. Source: Nik Piepenbreier

Reordering Columns

Reordering columns is more of a visual cue for yourself.

To drag a reorder a column in Excel, you'd select the column by click its header, hover on the side until the cursor changes to a four-pointed arrow, then hold the SHIFT key and drag the column to a new position:

	A	B	C	D	E
1	Date	Region	Type	Units	Sales
2	2020-03-25	East	Women's Clothing	29	\$ 609.00
3	2020-06-12	East	Women's Clothing	35	\$ 1,050.00
4	2020-06-29	East	Women's Clothing	24	\$ 504.00
5	2020-12-20	East	Women's Clothing	30	\$ 930.00
6	2020-06-06	East	Women's Clothing	28	\$ 560.00
7	2020-09-06	East	Women's Clothing	11	\$ 363.00
8	2020-04-29	East	Women's Clothing	14	\$ 462.00
9	2020-07-30	East	Women's Clothing	21	\$ 399.00
10	2020-11-19	East	Women's Clothing	8	\$ 256.00
11	2020-05-12	East	Women's Clothing	29	\$ 812.00
12	2020-11-26	East	Women's Clothing	22	\$ 440.00
13	2020-03-04	East	Women's Clothing	4	\$ 116.00

Moving a column in Excel. Source: Nik Piepenbreier

To accomplish the same thing in Pandas, you simply write in the order of columns you want to have into a double set of square brackets:

```
1 df = df[['Date', 'Sales', 'Region', 'Type', 'Units']]
```

datagy-bye-excel06.py hosted with ❤ by GitHub

[view raw](#)

Re-ordering columns in Pandas. Source: Nik Piepenbreier

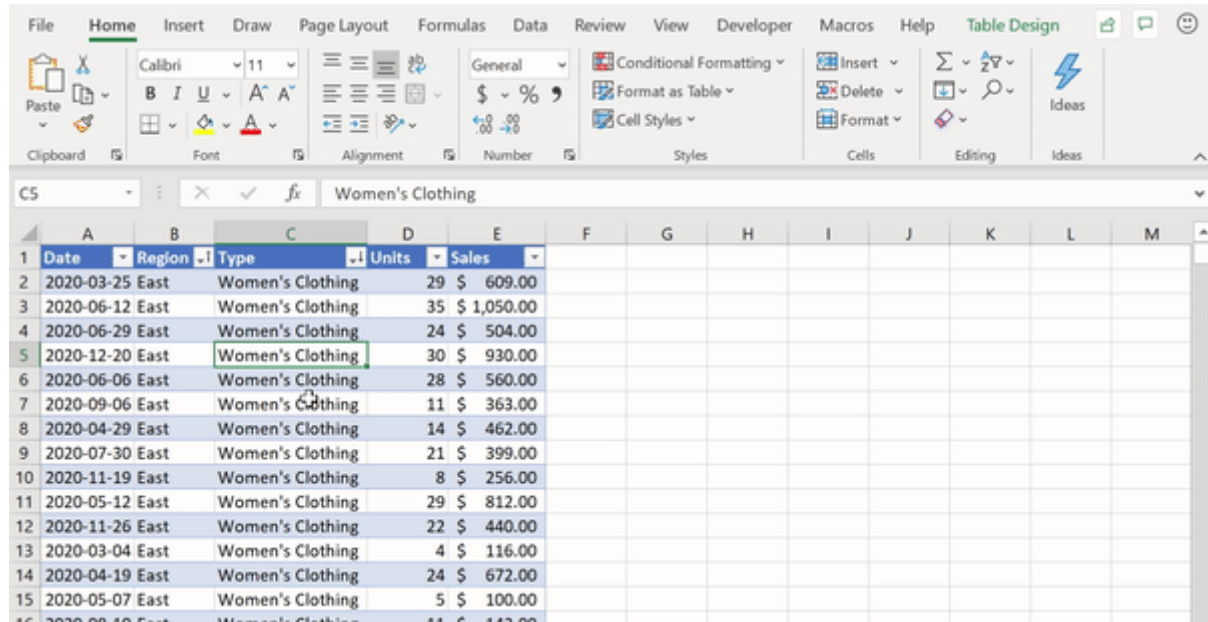
Pivot Tables (with Percentages)

Pivot tables are one of those things that take you to the next level in Excel.

They allow you to easily summarize data quickly, without needing to rely on complex formulas.

Say you wanted to know the total value of sales in each region, you would:

1. Select your data and click PivotTable on the Insert Tab and click OK to create your table.
2. Drag Region into the Rows box, and Sales into the values tab. Excel automatically assumes we want to add up the values.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Date	Region	Type	Units	Sales								
2	2020-03-25	East	Women's Clothing	29	\$ 609.00								
3	2020-06-12	East	Women's Clothing	35	\$ 1,050.00								
4	2020-06-29	East	Women's Clothing	24	\$ 504.00								
5	2020-12-20	East	Women's Clothing	30	\$ 930.00								
6	2020-06-06	East	Women's Clothing	28	\$ 560.00								
7	2020-09-06	East	Women's Clothing	11	\$ 363.00								
8	2020-04-29	East	Women's Clothing	14	\$ 462.00								
9	2020-07-30	East	Women's Clothing	21	\$ 399.00								
10	2020-11-19	East	Women's Clothing	8	\$ 256.00								
11	2020-05-12	East	Women's Clothing	29	\$ 812.00								
12	2020-11-26	East	Women's Clothing	22	\$ 440.00								
13	2020-03-04	East	Women's Clothing	4	\$ 116.00								
14	2020-04-19	East	Women's Clothing	24	\$ 672.00								
15	2020-05-07	East	Women's Clothing	5	\$ 100.00								

Creating a pivot table in Excel. Source: Nik Piepenbreier

To accomplish the same thing in Pandas, you could simply use the `pivot_table` function:

```
1 pivot = pd.pivot_table(df, index='Region', values='Sales', aggfunc='sum')
2 print(pivot)
3
4 # This returns the same table as Excel:
5 #
6 #Region
7 #East    167763
8 #North   138700
9 #South    59315
10 #West    61476
```

datagy-bye-excel07.py hosted with ❤ by GitHub

[view raw](#)

Creating a pivot table in Pandas. Source: Nik Piepenbreier

Let's break this down a little bit:

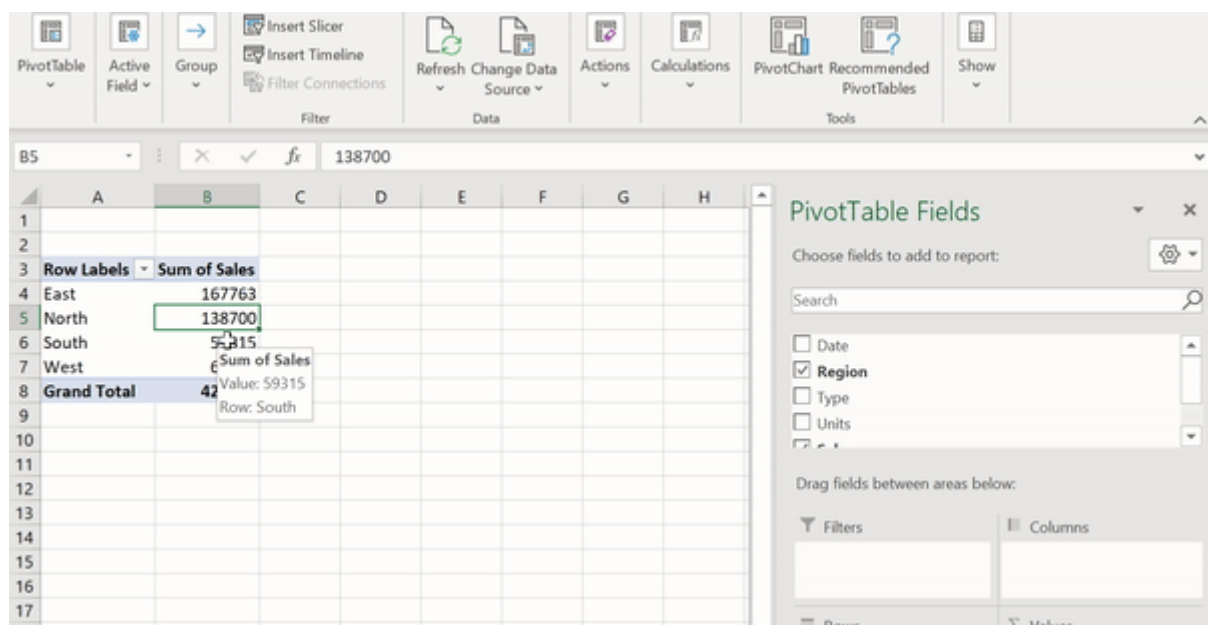
- We create a new variable called `pivot`.
- We use the `pandas pivot_table` function. Our first argument is the dataframe `df`.

- The index argument is 'region', which tells Pandas to create rows based on the 'Region' column.
- We assign the argument values the field 'Sales', to let Pandas know we want to calculate the Sales column.
- Finally, we use the aggfunc ('aggregation function') argument to tell Pandas to sum up the values. The default value is 'mean'.

For a deep dive into the Pandas Pivot Table function, check out my other post on [Pivot Tables in Pandas](#).

Show Pivot Table Values as Percentages

You may want to show your values as percentages of the column total. Again, Excel makes this very easy:



Calculating percentages in pivot tables in Excel. Source: Nik Piepenbreier

- Simply right-click on a value,
- Select Show Values as → % of column total

It's just as easy to do this in Pandas. The easiest way is to create a new column for this:

```
1 pivot = pd.pivot_table(df, index='Region', values='Sales', aggfunc='sum')
2 pivot['% of column total'] = pivot['Sales']/pivot['Sales'].sum()*100
3 print(pivot)
4
5 # This returns:
```

6	#	Sales	% of column total
7	#Region		
8	#East	167763	39.265402
9	#North	138700	32.463125
10	#South	59315	13.882843
11	#West	61476	14.388631

datagy-by-excel08.py hosted with ❤ by GitHub

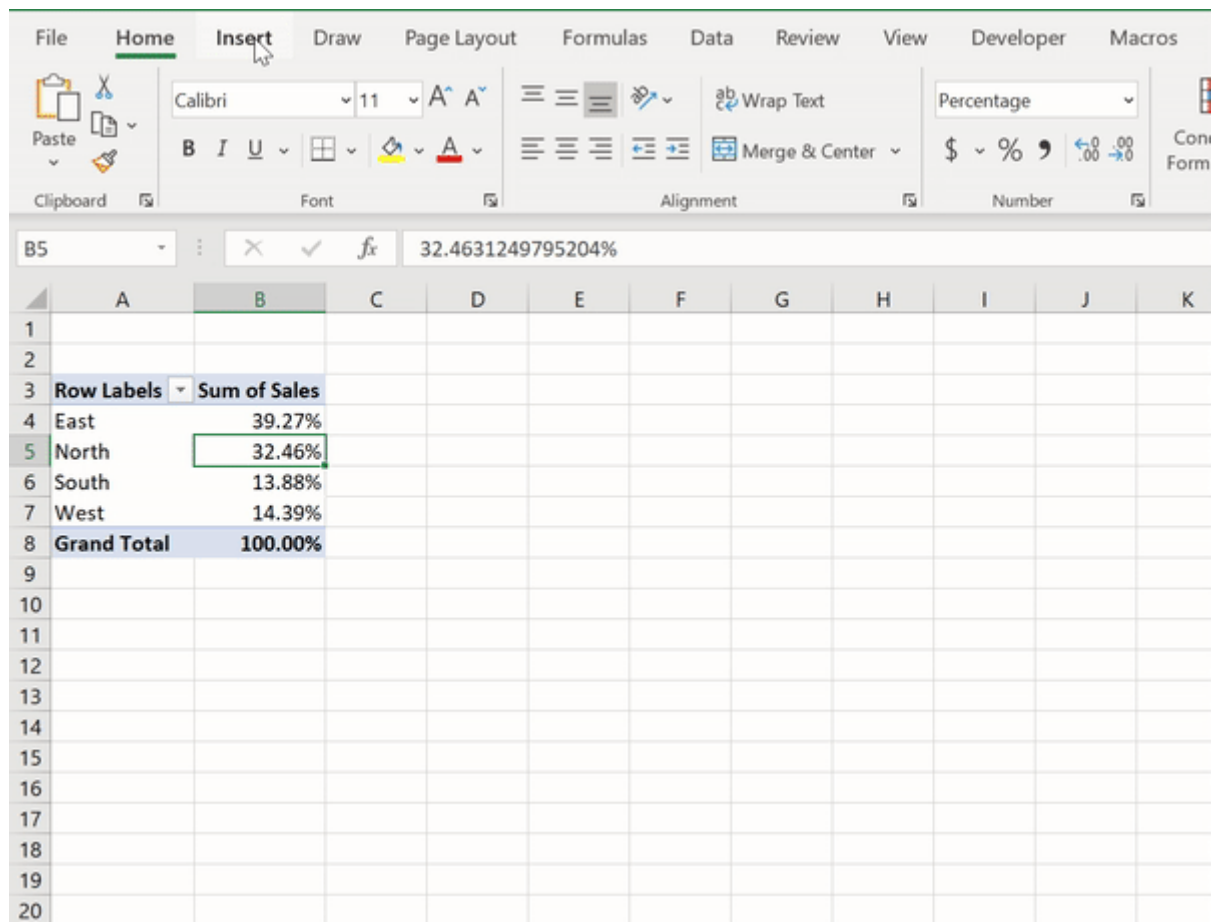
[view raw](#)

Calculating pivot table percentages in Pandas. Source: Nik Piepenbreier

Let's take a look at what's happening:

- We declare a new column by using `pivot['% of column total']` — this assigns the name '% of column total' to the column
- We then divide each value in the row (`pivot['sales']`) by the sum of the entire column (`pivot['sales'].sum()`) and multiply it by 100

Creating Charts



Creating a chart in Excel. Source: Nik Piepenbreier

Now, what if you wanted to create some charts, this is incredibly easy in both Excel and in Python.

Let's look at Excel first. If you want to plot this pivot table out as a column graph:

- Place your pointer on one of the cells in the table,
- Go to Insert → 2-D Column Chart

In Pandas, this is even easier. Pandas comes with one of Python's top data visualization libraries functionality built-in. It's as easy as adding `.plot(kind = 'bar')` to the end of your code:

```
1 import matplotlib.pyplot as plt
2 pivot = pd.pivot_table(df, index='Region', values='Sales', aggfunc='sum')
3 pivot['% of column total'] = pivot['Sales']/pivot['Sales'].sum()*100
4 pivot['Sales'].plot(kind='bar')
5 plt.savefig(PATH_TO_WHERE_YOU_WANT_TO_SAVE)
```

datagy-bye-excel09.py hosted with ❤ by GitHub

[view raw](#)

Creating a chart in Pandas and Matplotlib. Source: Nik Piepenbreier

This might look a little daunting. Let's break it down:

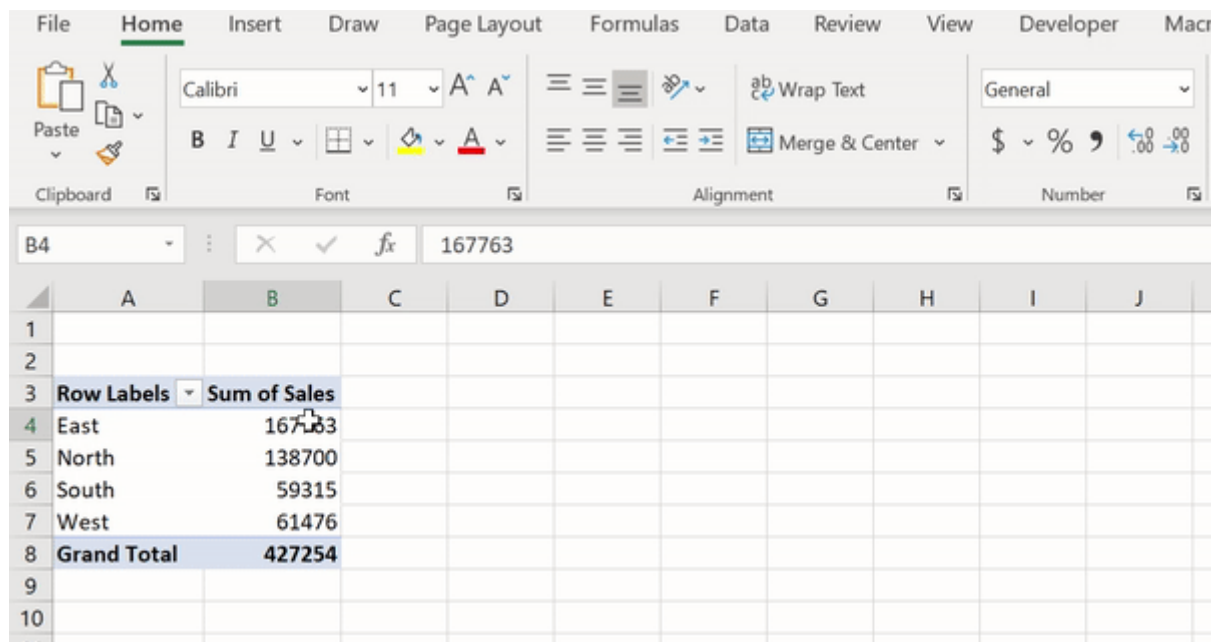
- Import pyplot from matplotlib as plt
- Re-write your earlier code (lines 2–3)
- Now plot out the 'Sales' column and assign a kind = 'bar' as an argument
- Finally, save the file by using the savefig method.

Note: if you're using Jupyter notebooks, you can display the chart inline by writing this code after importing your library:

```
%matplotlib inline
```

Bonus Tip: Format Values Properly

While you're working with data, it may be helpful to format your values properly.



Formatting your data in Excel. Source: Nik Piepenbreier

For example, formatting currencies as dollars (etc.) or percentages as percentages.

To do this in Excel, you would:

- Select the values you want to format,
- Under the Home tab in the Number section, select your desired type

Pandas hides this away a little bit, which is something that can stump newcomers quite a bit.

An easy way to accomplish this would be using the `apply()` function. What this function does is take a series and apply another function to it. The function being applied would be the one formatting the values.

If you wanted to format the Sales column in the pivot dataframe, you could write:

```
1 import pandas as pd
2 df = pd.read_excel('https://github.com/datagy/pivot_table_pandas/raw/master/sample_pivot.xlsx')
3
4 pivot = pd.pivot_table(df, index='Region', values='Sales', aggfunc='sum')
5 pivot['% of column total'] = pivot['Sales']/pivot['Sales'].sum()*100
6
7 def format(x):
8     return "${:,.2f}".format(x)
9
10 pivot['Sales'] = pivot['Sales'].apply(format)
```

```

11
12 print(pivot)
13
14 # This returns:
15 #           Sales % of column total
16 #Region
17 #East    $167,763.00      39.265402
18 #North    $138,700.00      32.463125
19 #South     $59,315.00      13.882843
20 #West     $61,476.00      14.388631

```

datagy-bye-excel10.py hosted with ❤ by GitHub

[view raw](#)

Formatting values in Pandas. Source: Nik Piepenbreier

This might seem a little round-up (and it is), but it does give you a ton of flexibility in terms of how you style your data. Let's take a closer look:

- We define a function called `format()` that takes one argument (`x`)
- The function only serves to return a formatted value using string formats in a particular format.
- The `${:,.2f}` part represents the actual formatting. The colon (`:`) is used to signify the beginning of the formatting, the comma (`,`) is used to signal comma separators for thousands, and the `.2` signals two decimal places.
- This notation can be a little tricky to get used to and I tend to google the style I want and copy and paste it.

Similarly, if you wanted to stay percentages, you could write:

```

1  import pandas as pd
2
3  df = pd.read_excel('https://github.com/datagy/pivot_table_pandas/raw/master/sample_pivot.xlsx')
4
5  pivot = pd.pivot_table(df, index='Region', values='Sales', aggfunc='sum')
6  pivot['% of column total'] = pivot['Sales']/pivot['Sales'].sum()
7
8  def format_currency(x):
9      return "${:,.2f}".format(x)
10
11  def format_percent(y):
12      return "{:.2%}".format(y)
13
14  pivot['Sales'] = pivot['Sales'].apply(format_currency)

```



```
15 pivot['% of column total'] = pivot['% of column total'].apply(format_percent)
16
17 print(pivot)
18
19 # This returns:
20 #           Sales % of column total
21 #Region
22 #East    $167,763.00      39.27%
23 #North   $138,700.00      32.46%
24 #South    $59,315.00      13.88%
25 #West    $61,476.00      14.39%
```

datagy-bye-excel11.py hosted with ❤ by GitHub

[view raw](#)

Formatting percentages in Pandas. Source: Nik Piepenbreier

In this code, we created another function called `format_percent()` and went through similar steps as above.

Note: The ‘% of column total’ column has been amended to not multiply the value by 100. This is because the formatter does this automatically.

Where You Can Learn More

Thank you so much for reading this article! I hope you found it useful!

I have written a number of articles that explain how to take on common and advanced Excel tasks in Python, if you want to check them out!

If you’re ready to take the plunge into Python and Pandas, I have also written an eBook that provides a complete introduction to Python, Pandas, and Matplotlib and will have you up and running in no time!

You can find it by [clicking this link](#).

Have a great day!

Python

Data Science

Software Development

Coding

Data

Medium[About](#) [Help](#) [Legal](#)

Get the Medium app

