In python, we call a pivot table a ‘Group By’.

You know you need one if you need to compare variables or aggregate values that are not readily available in your current data frame. For example, if you want to find out which cities have the greatest number of PPP loan recipients, you need to do a Group By. Think back to the Excel exam: If you need to know the total number of violations each restaurant has received on all inspections combined, you need a Group By.

The premise is the same as in Excel. First stop and think, what am I grouping by? In excel, you would put that column into rows. In Python, you just put it in parentheses.

Next, ask yourself, what am I measuring? In Excel, that would go into values. In python, that would go into brackets.

Then, rather than clicking on that little ‘i” in the values box, in Python you write the action out.

Below is an example -- here we are just counting the number of loans per state. (Which effectively the number of times each state name comes up.)

loansbystate = df.groupby('state')['state'].count().to\_frame(name = 'loans').reset\_index()

Now we know the raw number of loans per city, but what percentage of the total is that?

You’ve probably already done a .shape() so you know how many loans exist in your database. (766499)

Think of this like creating a new column next to your pivot table. In the example below, I’m calling my new column “percent,” and telling the computer to place it in my loansbycity dataframe. (our group by)

loansbystate[‘percent’] = loansbystate[‘loans’] / 766499

If you want to sum the amount of money received in each state, it would look like this:

loansbystate = df.groupby('state')['CurrentApprovalAmount'].sum().to\_frame(name = 'total dollars').reset\_index()

*You will notice that if you run df.head(), the computer has shortened your numbers and it’s annoying and hard to look at. You can run this code and it will cut that out.*

*​​pd.options.display.float\_format = '{:.2f}'.format*

*(Don’t ask me what the inside of those curly brackets mean. I haven’t a clue.)*

You can switch out .sum() with a wide range of commands. Some examples:

.max()

.min()

.mean()

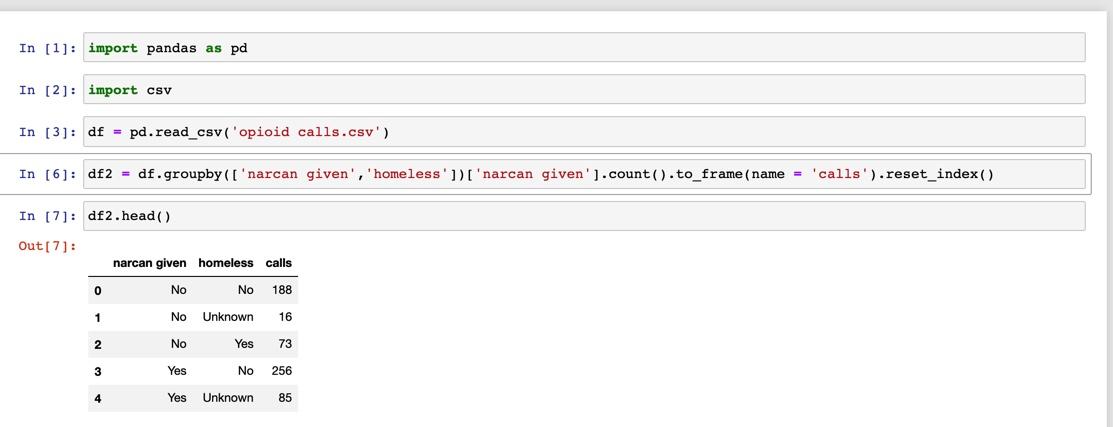
.median()

But what is all that stuff after the .sum() or .count() command, you ask?

To\_frame transforms your group by into its own data table, and you pick the column name for the thing you measured. In this instance we are summing money, so I just called it total dollars. You could call it anything you’d like. .reset\_index is necessary to make a new data table.

More often than not, you’ll be grouping on one column. But sometimes you’ll need to group by two. We needed to do this with the opioid data to determine if homeless people were less likely to be given narcan. So I’ll use that as an example here.

df2 = df.groupby(['narcan given','homeless'])['narcan given'].count().to\_frame(name = 'calls').reset\_index()



(To try this out, you’ll need to save your opioid data file as a csv)

… but we don’t have to get that complicated just yet.