Continue working the PPP Loan data you used in the first tutorial. You can work off the same jupyter notebook as well. But remember, you probably changed your headers so the exact code below may not work verbatim. Be sure to use the exact names of your columns where they are relevant. For example, the original data had a column called ‘BorrowerCity’. You might have changed that column to ‘city’. So in that first line of code below, you’d switch BorrowerCity to city.

Python has a lot of very powerful filtering options. We will not cover them all, but we will get to the ones you will use most commonly.

**Text Filters**

Before you start filtering text, it is important that everything is either all lower case or all upper case. Python is case sensitive, so if you’re looking for “PHOENIX” you will miss values that say “phoenix”.

Get in the habit of using code like this before you filter:

df["BorrowerCity"]= df["BorrowerCity"].str.lower()

This, like when we changed data types, overwrites the original column BorrowerCity with all lower-case values.

It is also good to strip potential spaces at the beginning or ends of words.

To do that, use the above code, but replace .str.lower() with .str.rstrip() to strip space on the right and .str.lstrip() to strip spaces on the left.

If you want a perfect match, you will use ==

If you want to find any value that contains a word you’re looking for, you’ll use a str.contains command.

Let’s see what that looks like.

*#create a new dataframe where the city is exactly ‘phoenix’*

phx = df[df['BorrowerCity'] == 'phoenix']

*#create a new dataframe where the city contains the word phoenix*

*#na=False is just telling the computer to ignore the null values*

phx2 = df[df['BorrowerCity'].str.contains('phoenix', na=False)]

With either one of these lines of code, the equivalent in Excel would be to filter your data, copy and paste it into a new sheet and name the sheet. Much more efficient this way, right?

Now, we have a whole new dataframe called phx, or phx2, whichever filter you chose.

But what if we want more than one city? We separate the two commands with what is called a pipe. This thing: |

df2 = df[(df['BorrowerCity'] == 'phoenix') | (df['BorrowerCity'] == 'scottsdale')]

It gets a little tricker when you need multiple values based on a contains filter.

First, we start by making a list, and giving it a name.

cities = ['phoenix', 'scottsdale', 'mesa']

Then we make the filter.

citiesdf = df[df['BorrowerCity'].str.contains('|'.join(cities)) == True]

If you change the end to ==False, it will EXCLUDE those cities. That will come in handy, too.

You can also write a filter based on multiple columns. Let’s say we want to know rows where the owner is a woman AND a veteran.

df4 = df[(df['Gender'] == 'Female Owned') & (df['Veteran'] == 'Veteran')]

Or, what if we want rows where the person is EITHER Female OR a Veteran? You use a pipe instead of an &

df4 = df[(df['Gender'] == 'Female Owned') | (df['Veteran'] == 'Veteran')]

**DATE FILTERS**

Date filters work very similarly to number filters, but make sure you’ve converted your dates to actual dates before you go running filters on them. If you forget, the filter may not work or it may give you inaccurate results.

*#Between two dates*

daterange = df[df.DateApproved.between('04/10/2020', '04/12/2020')]

*#after, before or exact date*

AfterApril11 = df[df['DateApproved'] > '04/11/2020']

BeforeApril11 = df[df['DateApproved'] < '04/11/2020']

April11 = df[df['DateApproved'] == '04/11/2020']

If you want to exclude a date, != means DOES NOT EQUAL

If you want to include the date you’re specifying, much like in Excel, you’d just write >= or =<

Numbers

With numbers, the biggest difference is that we don’t use quotes.

Jobs = phx2[phx2['JobsReported'] == 100] #this gives you anything that equals 100

Jobs2 = phx2[phx2['JobsReported'] >= 100] #this is equal or greater than 100

Jobs3 = phx2[phx2['JobsReported'] <= 100] #less than or equal to 100

Note this code works off our phoenix filter. If you change phx2 to df, it will work off the original data.

For practice, make your own filters based on what you’re interested in.