**Room 8 ETL Report**

Brian, Ryan, Luke, Aaron

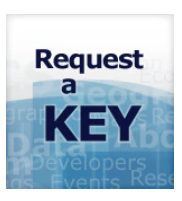
10/07/2021

**Introduction:**

This ETL guide will serve as directions and a step-by-step walkthrough of how we managed data to create meaningful visualizations using Python to explore relationships between several sets of data from the 2019 Annual Business Survey (ABS) APIs. The ABS contains information related to economic and demographic statistics for businesses (Bureau 2021). The datasets available encompass Company Summary, Characteristics of Businesses, Characteristics of Business Owners, and Technology Characteristics of Businesses, though this group primarily concerned ourselves with the first set of data: Company Summary. We were looking to create diagrams relating to race, gender, number of employees, annual payroll, veteran status, and sales receipts. The requisite data will need to be called, transformed, and merged to create a convenient, cumulative file from which visualizations can be created. A companion Jupyter Notebook file can also be found in the Git Repository for this project.

**Data Sources:**

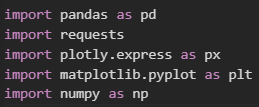
Bureau, U. S. C. (2021, March 21). *Annual Business Survey (ABS) APIs*. The United States Census Bureau. Retrieved October 7, 2021, from https://www.census.gov/data/developers/data-sets/abs.2019.html.

**Extraction:**

Using the data source above, we were able to generate an API key using the generously sized button on the left margin of the page (see image, right). This API key will be necessary later as the resource allows a limited number of free API-calls per IP address, but a much larger number of calls to a registered API key. This API key, along with an up-to-date version of Python, are required.

Python will need a few libraries installed to correctly process the steps for this report. These libraries include: *pandas*, *requests*, *mathplotlib*, and *plotly*. If these libraries are not already available to you, you will need to install them on your machine using command line prompts. An example showing the syntax for installing *pandas* is shown here: . If you’ve had to install the packages, a restart of the program you are running Python in will likely need to be closed and reopened for changes to take hold.

Once the appropriate packages have been installed, a new Python program can be started. Start by importing these libraries using the following code:



This should initialize the packages and allow you to use them in code.

The next thing to consider is how to get the API to respond correctly to our code. The source website itself gives a few excellent examples, but it is important to note that the format follows a few rules. The first part of the URL we used is shown below:



The URL itself is specific to each dataset, differentiated by the series of letters following the “…*data/2018/*” portion. For our project, we exclusively used “api.census.gov/data/2018/abscs”, where “abscs” is short for “Annual Business Survey Company Summary. Knowing the root of our URL is a good start, but we need to specify which information we’d like to return from the API. Variables for data to be returned can be found under each section of the main website by clicking on the link next to *Variables* listed below the **Company Summary** section. 

This page lists the variables that can be returned using a *get* call from the API. For our data, we are using the following variables:

* NAME
* EMP
* PAYANN
* SEX
* EMPSZFI
* ETH\_GROUP
* RACE\_GROUP
* VET\_GROUP

Adding these variable names into the URL (immediately following the snippet above) specifies which data sets we are returning. Our URL is now:



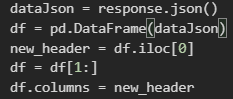
The URL from here can filter multiple ways and dictates that you include the API key registered earlier. We were filtering two predetermined states, our URL is now:

https://api.census.gov/data/2018/abscs?get=NAME,EMP,PAYANN,SEX,EMPSZFI,ETH\_GROUP,RACE\_GROUP,VET\_GROUP&for=state:50,56&key=55e66c014b79233a93d62e4de48c6b4cbae8d3fc

As you can see, the state numbers we used were 50 and 56, numbers defined in the API, and the API key is added to the end of the URL. Using this now-complete URL, we can return multiple variables from the two designated states using our registered API key.

 Next, we use the URL in combination with proper syntax to make a *get* request. We saved that request as a new variable, as seen below:

Knowing the capabilities of *pandas*, we were able to use built-in methods to transform he response to a *.json* format, and then transform the *.json* to a dataframe, our preferred method for dealing with information in Python. The dataframe needs slight formatting, due to the inconvenient headers. Below, you can see the steps we took for the previous *.json* steps and also the reformatting of the dataframe.



We now have a decent starting point for our data, and can begin to use it.

**Transformation**

Our data, as extracted, contained many columns filled with codes that were referenced in a different location. Our approach to solving this included manually writing our own dataframes including these codes and their definitions. The excerpt below shows one such example:

sexCodes =[["Sex Category", "SEX"], ["Total", "001"], ["Female", "002"], ["Male", "003"], ["Equally male-/female", "004"], ["Classifiable", "096"], ["Unclassifiable", "098"]]

dfSex = pd.DataFrame(sexCodes)

new\_headerSex = dfSex.iloc[0]

dfSex = dfSex[1:]

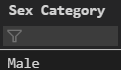
dfSex.columns = new\_headerSex

This code builds a dataframe with the lists, and then transforms the first row into the headers for those columns. Once this dataframe has been built and transformed, we can merge this to the main dataframe created previously which should let us connect ambiguous codes to their written-out definitions.

Merging the dataframes utilizes the *pandas* library of functions and the convenient structure of our data. An example of the code used to merge the sexCodes dataframe to our main dataframe is shown below (NOTE: the portion following “on =” must be the same as the second entry in the dataframe you built):

dfMerged = df.merge(dfSex, how = "inner", on = "SEX")

This new dataframe, ‘dfMerged’, will contain all the information we will need and make it easier for interpreting the data. Where before, we were looking at a meaningless ‘003’, now we can see that that value is equal to ‘Male’.



We repeated this step to build dataframes for each variable in our main URL and merge them to our main dataframe.

Because we will not be using all the data columns (and they might clutter our results), we will drop them in the code. We performed this step right before making our visualizations to ensure the integrity of our data. In the example below, we are going to create a new, unique dataframe specific to our question (exploring the distribution of the sex of employees) and drop unused categories.

dfMergedSex = dfMerged[dfMerged["Sex Category"] != "Classifiable"]

dfMergedSex = dfMergedSex[dfMergedSex["Sex Category"] != "Unclassifiable"]

dfMergedSex = dfMergedSex[dfMergedSex["Sex Category"] != "Total"]

dfMergedSex = dfMergedSex[dfMergedSex["Sex Category"] != "Equally male-/female"]

As you can see, we are not interested in the labels noted as *Classifiable, Unclassifiable, Total, Equally male/female* and so have excluded them from the new dataframe “dfMergedSex”.

These new merged dataframes can be then narrowed down more in order to fulfill our goal of comparing Vermont statistics to Wyoming statistics. This can be achieved by the following:

dfMergedSexVM = dfMergedSex[dfMergedSex["NAME"] != "Wyoming"]

dfMergedSexWY = dfMergedSex[dfMergedSex["NAME"] != "Vermont"]

We now have individual dataframes for the sex distributions in Vermont and Wyoming that we can conveniently use to build visualizations. This process was repeated for the other variables.

**Conclusions**

We have Extracted and Transformed data – completing the ETL process for this project. After finding the data, making an appropriate API call, and finally transforming the data, we can use it seamlessly to address our original questions with visualizations. Following the steps written out above, an identical report can be built to match what we have procured.