**Group Name: SNAP (Slack Channel: etl\_group)**

SNAP is the Supplemental Nutrition Assistance Program, commonly referred to as “Food Stamps.” SNAP is also called FNS (Food & Nutrition Services.) This program is funded by the federal government through a grant program and the federal government also approves and maintains the list of authorized venders (IE where you can spend your food stamps.) The states are responsible for checking the eligibility and authorizing the individuals and the amount to be credited to their food stamp card each month.

**The project:**

We downloaded individual files, loaded time into Pandas data frames, preformed cleanup and then loaded the data frames into a SQL database with a table for each file. We named the database “snap\_db” and the tables are food access tables (Total 7 tables), “fs\_recipt”, “snap\_participants”, and “store\_locator”. More information about each table is below in the section about that table. Each member of the group worked on one table, and we worked as a team to determine what tables to download and how they relate to each other.

**Getting started:**

The group researched many data sources available from the three sites outlined in the project guidelines and after talking about the choices we chose to work with SNAP data. We then looked at many sites of available data about the SNAP program and talked about how we could relate each of the sites data to the main project. We settled on the four data sets outlined below because they would all build on each other to form a larger picture of SNAP then just the individual tables would show on their own. We will talk about the relations at the end of this report.

**Making the SQL password more secure:**

We wanted to make sure that none of use accidentally uploaded our SQL password to GIT. To achieve this, we each built a folder on or local hard drives in the same location (outside of the Git pull zone) and used the Jupyter code…

sys.path.append("C:/SQL\_PW/")

import config

This allows Jupyter to go to that path on or local computers to go to a config.py file and get the password.

The config.py file text is one line and is simply…

password = "*your\_password*”

* Replace the text *your\_password* with the actual password.

**E(xtract):**

**store\_locator:**

Retailer data for the entire US, including the following fields. (schema)

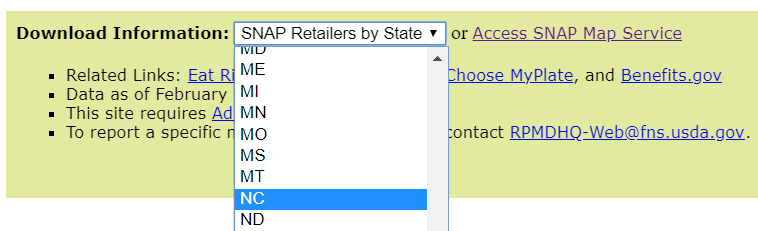
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Type** | **Start** | **Length** | **Decimals** | **Field explanation** |
| Store\_Name | ASCII | 1 | 56 |  |  |
| Longitude | ASCII | 57 | 10 |  |  |
| Latitude | ASCII | 67 | 9 |  |  |
| Address | ASCII | 76 | 43 |  |  |
| Address\_Line\_\_2 | ASCII | 119 | 40 |  |  |
| City | ASCII | 159 | 26 |  |  |
| State | ASCII | 185 | 2 |  |  |
| Zip5 | ASCII | 187 | 5 |  |  |
| Zip4 | ASCII | 192 | 4 |  |  |
| County | ASCII | 196 | 21 |  |  |

This data shows the number of authorized retailers for each state, including the address and the geotag (geo coordinates) of each retailer. With this data we can breakdown the retailer data by state\*, county, zip code or even look at the distances between retailers using the geocodes.

\*When we downloaded the whole country’s data set, it had 250,000 records and took well over an hour to load into a Jupyter data frame, so we decided to work with the much smaller North Carolina file that has only 9,241 records.

The data is located at <https://www.fns.usda.gov/snap/retailerlocator> and was downloaded as a CSV file.

At the bottom of the page, there is a dropdown that allows you to download the data as a CSV file



**food\_access:**

In this data, several indicators are available to measure food access along these dimensions. For example, users can choose alternative distance markers to measure **low access** in a neighborhood, such as the number and share of people more than half a mile to a supermarket or 1 mile to a supermarket. Users can also view other census-tract-level characteristics that provide context on food access in neighborhoods, such as whether the tract has a high percentage of households far from supermarkets and **without vehicles**, individuals with **low income**, or people residing in group quarters.

The data is located at <https://data.world/usda/grocery-stores> and was downloaded as an XLSX file.

This data can be used to look for areas that are food desert, and when combined with the retailer from Dan’s data we can look for areas that are food desert, but have plenty of grocery stores. We can also use it to see the population of different areas and when combined with the individual snap participants from Zhen’s data we can see the ratio of participants that joined food stamp service.

Documentation on the table (schema) can be found at <https://www.ers.usda.gov/webdocs/DataFiles/80591/documentation.pdf?v=0>

**snap\_participants:**

**[Data Source**]

Supplemental Nutrition Assistance Programe (SNAP) Participation by County

<https://public.opendatasoft.com/explore/dataset/supplemental-nutrition-assistance-program-snap-data-system-participation-by-coun/table/>

**[Goal]**

Calculate total population, count of people in poverty, count of SNAP participants, and total benefits for each state.

**fs\_receipt:**

Data source: found using <https://opendata.socrata.com/> Government US-Food-Stamps-By-State data in csv form.

Data wrangling: cleaning data and possible join with different years

Data schema: State, Resident pop, Number of People in January 2008, % Total pop Jan 2008, Preliminary December 2008, % Total pop 2008, initial Jan 2009, % Tot Jan 2009, % Change Dec 08-Jan 09, % Change Jan 08- Jan 09

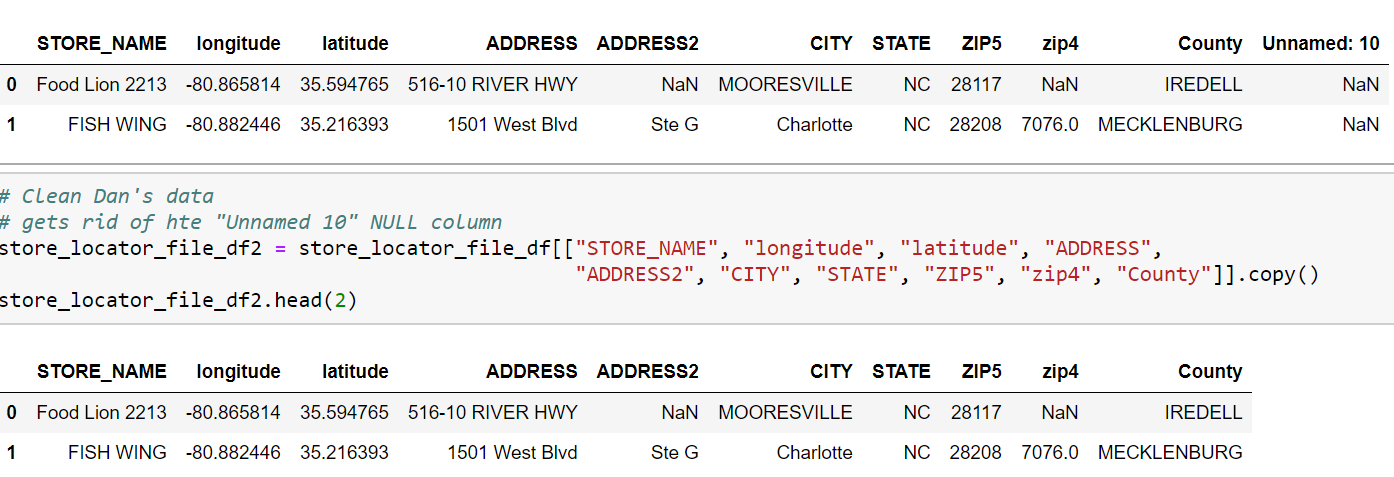
Working on data food stamp receipt. In May 2008 a Farm Bill (The Food, Conservation, and Energy Act of 2008) was vetoed by the president. This bill increased government commitment to improving food assistance program by implementing structures to improve access to nutritional health, simplifying administration and maintaining state flexibility and reducing the stigma by changing the name of Food Stamp Act of 1977 to Food and Nutritional Act of 2008 and modernizing it by using EBT as the standard issuance.

Would like to see the immediate effect of the bill on food stamp receipt across states as it relates with how many people are accepted into food stamp and the number of people that left the program with the bill reform that occurred in 2008.

**T(ransform):**

**store\_locator:**

The store location data was already very clean data. It had an “Unnamed: 10” field at the end of the dataset that was only null (NaN) values. We simply created a second data frame that excluded this field.



**food\_access:**

Since this data is pretty clean and all the columns can be used in future analysis. We focused on transformed it to several tables which should be easier for us to do the future analysis.

-Get separate Urban and Rural data  
 [table 1&2] 'foodaccess\_rural' 'foodaccess\_urban'

-Get areas with low access (do not have grocery store nearby)

[table 3&4] 'data\_rural\_la' 'data\_urban\_la'

-Get the percent of low vehicle access (households do not have vehicle)

[table 5&6] 'data\_rural\_vehicle' 'data\_rural\_vehicle',

-Get the population data for different areas

[table 7] ‘population'

**snap\_participants:**

1: The dataset has about 6300 columns. Loading into pandas will probably crash it. I had to cut a tremendous number of columns in the original .csv file. Luckily the columns of interest are all in the first 100 columns. I created another .csv using only the columns that are relevant and used this .csv as the dataset to clean.

2. The dataset is supposed to have data from year 1969 to 2011. But a lot of years are missing from the original dataset. For this project I chose 2 years to analyze, 2007 and 2010.

3. Dataset has NaN cells. In the timeframe of this project, I dropped these cells. If time permitted, it’d make more sense to use a median value to replace the NaN values.

4. Dataset has negative numbers in cells. According to the website information, cells with negative numbers mean data is not available. Similar as above, in the timeframe of this project, I dropped these cells. If time permitted, it’d make more sense to use a median value to replace the negative values.

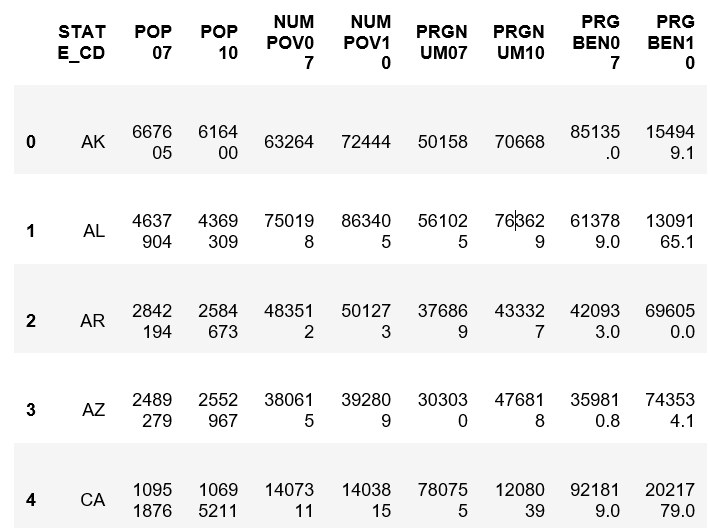
5. Population and number of SNAP participants columns have decimal point. After examining the dataset closely, I found that the decimal points are actually thousand place marks. But I cannot simply replace the points with empty strings, because some numbers have 1 digit after the point, some numbers have 2 digits after the point. I had to use a conditional function to check whether a number has a decimal point. If it does, I will multiply that number by 1000.

Note: in the timeframe of this project, I wasn’t able to figure out how to make the above step work. First of all, when pandas reads .csv into the dataframe, it automatically converts all numbers into floats. I did some research about this pandas behavior and found that it’s because some columns have NaN values. So first I had to force pandas to read in columns as object type to make the data look as is. In the conditional function, I convert the columns into string then check whether ‘.’ is in the value. Unfortunately this function does not work. I did not have time do research more and find out how to make it work.

Dan helped me converted those numbers in ACL as a workaround

6. Some state names have “ in it. Time is not permitted to research more on the dataset to decide how to clean those rows. I dropped those rows.

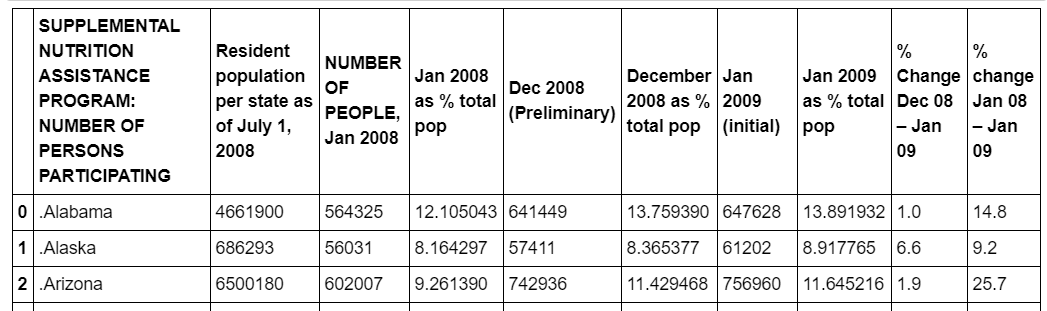
7.Calculate sum of population, number of poverty, number of SNAP participants, benefits group by state.



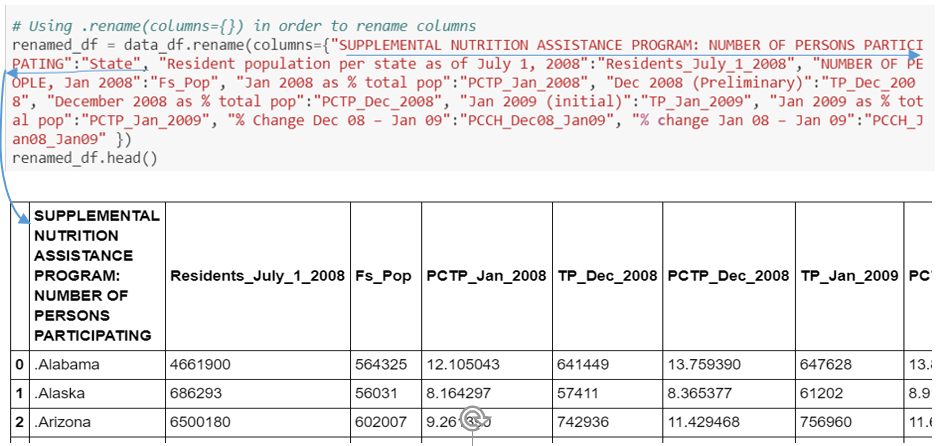
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

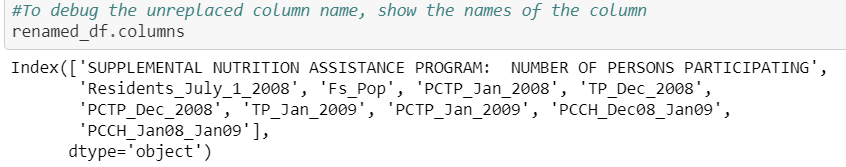
**fs\_receipt:**

Data was already cleaned and aggregated. Need to simplify the field/column name.

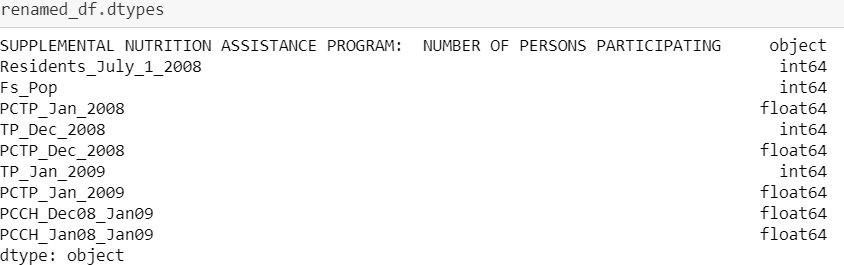


Renamed columns with one failed attempt.

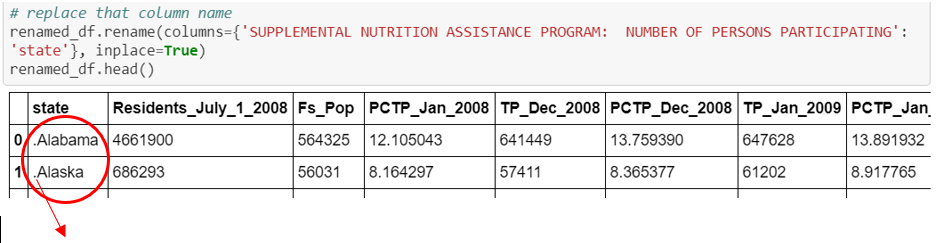




Also checked the type of data.

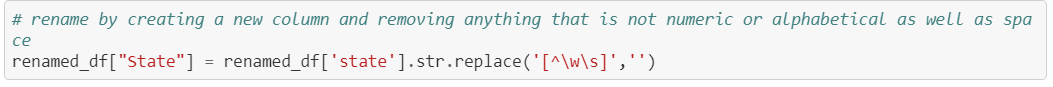


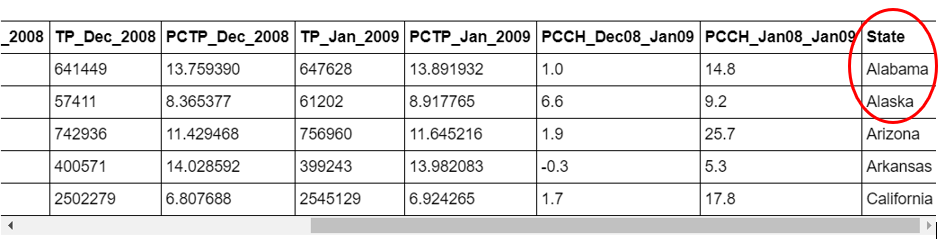
Successfully replaced the column name by using the name revealed by the code “rename\_df.columns.



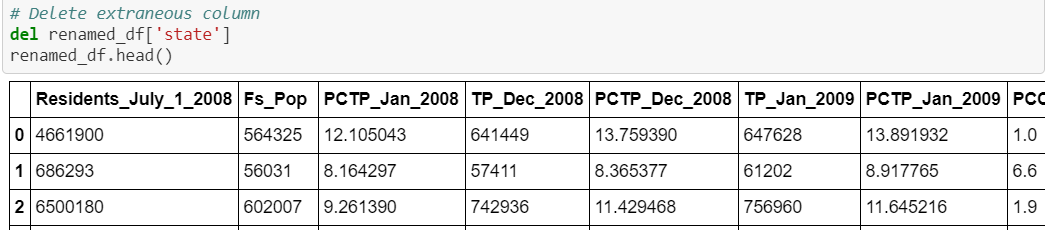
Worked on removing “.” from the column.

Used the code below to create another column with the same data. Only collecting the alphabet or numeric and excludes anything else for example ‘.’ and strings.

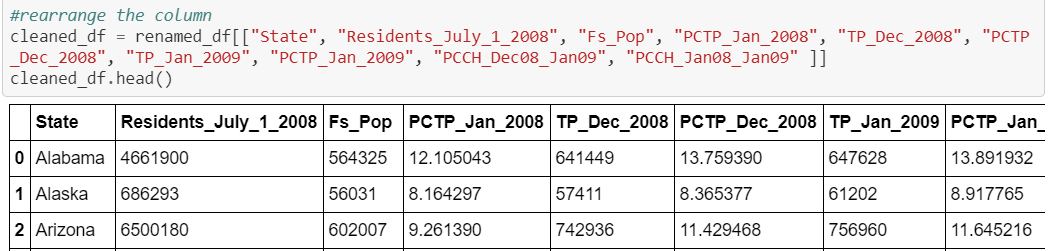




The State created (with capitalized S) became the last column in the data. Deleted the first column “state”.



Rearranged the column to move the new column from last position to the first by creating a new data frame.



**L(oad):**

**store\_locator:, food\_access, snap\_participants, fs\_recipt: (all tables)**

With the raw data cleaned up so that the null field is removed, we picked a table name that would be unique and represent that the data set is. We then preformed a load to the SQL database that we had setup that checks to see if the table already exists, and if it does, removes it and then loads the new data into the database.

Please note that once the table was live, future imports would be imported into a view, and then the view would be run against the table to identify what records were new. Then, only the new records would be appended to the live table. We would also want to have a unique identifier for each record as well as a data of insertion filed.