**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 four tables are “food\_access”, “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)

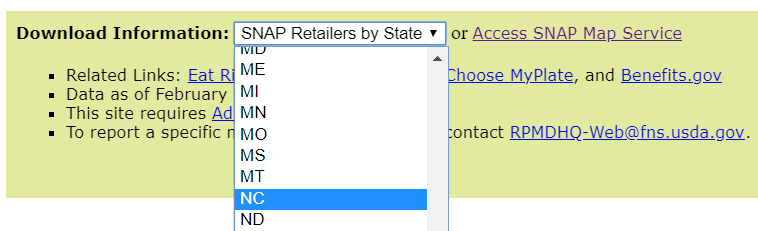
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Type** | **Start** | **Length** | **Decimals** | **Field explanation** |
| Store\_Name | ASCII | 1 | 56 |  |  |
| Longitude | ASCII | 57 | 10 |  |  |
| Latitude | ASCII | 67 | 9 |  |  |
| Address | ASCII | 76 | 43 |  |  |
| Address\_Line\_\_2 | 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:**

Data: Food Access Research --several indicators to measure food access.

\*Low access to the grocery store, distance, vehicle availability, age, income etc.

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 stamp food desert, but have plenty of grocery stores.

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: US Department of Agriculture - Food and Nutrition Service

National Level Data of participation and cost from 2016 to 2019

<https://www.fns.usda.gov/pd/supplemental-nutrition-assistance-program-snap>

Type of data wrangling: data cleaning, aggregation

Schema: Persons, Households, Benefits, and Average Monthly Benefit per Person & Household (edited)

**fs\_recipt:**

Data source: using <https://opendata.socrata.com/> Government US-Food-Stamps-By-State.

Data wrangling: cleaning data and possible join with different years

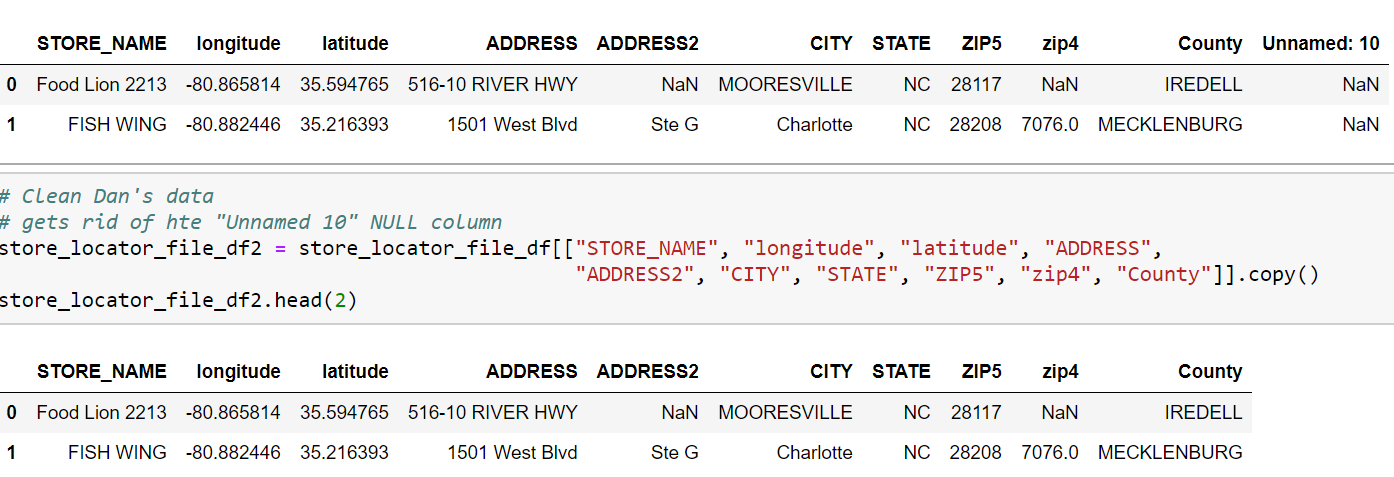
Data schema: State, Resident pop, Number of People in January 2008, % Total pop Jan 2008, Preliminary December 2008, % Total pop 2008, initial Jan 2009, % Tot Jan 2009, % Change Dec 08-Jan 09, % Change Jan 08- Jan 09

Working on data food stamp receipt. Would like to see the progression as it relates with how many people are accepted into food stamp and the number of people that leave the program.

**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:**

**snap\_participants:**

**fs\_recipt:**

**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.

**Data Relations:**