**ETL Project Final Report**

This extract transform load (ETL) project included the following team members: Jensen Binoji, Ted Stagner, and Cuong Huynh.

**Sources of Data**

The sources of data included the following sources:

1. The **Census**
   1. URL: <https://factfinder.census.gov/faces/nav/jsf/pages/searchresults.xhtml?refresh=t&keepList=t>
   2. Data extracted from census:
      1. Annual State Population Estimate
      2. Estimated Number without Health Insurance
      3. Rate of Uninsured= Number without Health/Population
      4. For the years 2009-2017.
2. **OASIS**--stands for Online Analytical Statistical Information System; Is a Georgia State Public Health Database.
   1. URL: <https://oasis.state.ga.us/oasis/webquery/qryER.aspx>
   2. Data extracted from OASIS:
      1. Emergency Room (ER) visitation rates for Georgia for the years 2009-2017.
3. **AHRQ**—stands for American Healthcare Research and Quality; Is a Healthcare Quality Organization,
   1. URL: <https://nhqr.net.ahrq.gov/inhqrdr/data/query>
   2. Data extracted from AHRQ:
      1. Pneumonia rates for Ohio, Oregon, and Georgia, for the years 2011 to 2015.

**Transformation**

In discussing our data transformations, I want to start by discussing the end goal of our data as this might explain the steps we took during the data transformations. We wanted data from both Medicaid expansion states and non-Medicaid expansion states. We got this via two Medicaid expansion states of Ohio and Oregon and we used Georgia as our one non-Medicaid expansion state. We wanted to compare these states Pre and Post Affordable care act implementation, hence we have the years of 2009 to 2017. More specifically, we wanted to compare health insurance rates. We derived the health insurance rate by dividing the number of uninsured by the population. We wanted to compare non-Medicaid expansion states to Medicaid expansion states.

We also wanted to compare health outcomes to health insurance rates. To look at health outcomes, we used rates of pneumonia from AHRQ and ER visitation rates from OASIS. The census data provided the population for a state for a given year. However, in regards to the uninsured we found counts for various age demographics for both male and female and needed to sum the various counts to get a total uninsured count. Then we divided the uninsured count totals by the population to get an uninsured rate. In our dateframes and database ‘rate’ refers to an uninsured rate. There were many other transformations done to aid in the review of the data like transposing the data (switching columns and rows) and or changing column headers.

We creatively utilized lists and dictionaries in our loops to loop through states, years, and data-frames. In one case, via a loop, we stored data-frames in a dictionary and were able to simply call the data-frames stored in the dictionary via the Dictionary[the Df] syntax.

In regards to cleaning data, we had several non-numeric values or missing values and something described as Data Statistically Unreliable or DSU data in the AHRQ data. Thinking that we did not want this type of data saved in the database we first tried to set all of these cells to null or empty string. However, when we attempted to save the data to a MySql database via the use df to SQL we received an error. To resolve this error we converted all DSU, empty string, and or NaNs to NaN, which the df to SQL method was then able to handle. The interesting thing about this is the code behind the df to SQL then converted the NaNs to nulls in the MySQL database.

In regards to joining data, we used a more surgical approach. We used loops and a **df.loc[row][col]** syntax to extract and load into a summary df. Via this method we were able join on an appropriate state and year data.

For filtering data, more specifically for finding the number of uninsured to sum, we used **df[col].str.contains** syntax to filter on the rows we needed to sum. Here is an example of this type of code:

**theUninsured=theDFs[theYear][(theDFs[theYear].index.str.contains('Estimate')) & \**

**(theDFs[theYear].index.str.contains('No health insurance coverage'))][f'{state}'].sum()**

**Loading**

Now comes the final part of our project , that is loading our stored data into a summary Dataframe and pushing this into a MySQL database. For this final part we used MySQL. We created a database called health insurance and created a table within the database called State summary. The table consists of 6 columns-- namely, stateYear, total, uninsured, rate, ‘ERvisit rate’ and ‘pneumonia rate’. Id was set as the primary key and given to auto-increment .

A screenshot of a cell phone

Description automatically generated

**Code for MySQL**

After that, we created a engine for converting the df to SQL command. For that we imported pymysql and also imported create engine from sqlalchemy. By inserting our specific password in the connection string we could access the MySQL database. After that minor errors were there and it needed some cleaning.

The main errors were, replacing the commas in the ‘ERvisitRate’ column and converting the ‘PneumoniaRate’ and ‘ERvisitRate’ columns to numeric. Converting the columns to numeric added back the NaN values to the Dataframe so the ‘to\_SQL’ command would not produce errors. Finally, we did a few plots using the matplotlib functions on the state summary Dataframe and since we don’t have values for all the years there are some irregularities in the plots.