Lean Hog Commodity Data

**Extraction**

The Lean Hog Commodity data is accessible via the Investing.com website (URL: <https://www.investing.com/commodities/lean-hogs-historical-data>). The available data consists of a historical price and volume table that extends back one month by default. We use splinter’s Browser and bs4’s BeautifulSoup to scrape the html data from this website. We then loop through the available tables in the html object and select the table that contains the phrase “Open” as our desired table uniquely contains that phrase. We have extracted our desired table.

**Transformation**

First, we do some initial cleaning on the extracted table. The table is currently in a string format that allows us to easily replace data. We format the full string prior to splitting the string into a list. We clean the list and remove any of the blank items from the list; each item in the list is going to become a prospective row. We then loop through the list and split each item, now we have a list of lists. We clean the list of lists to remove additional blank lists that may occur and we strip and leading or trailing blank-space as well.

We then create a dataframe from the list of lists; the first list becomes the headers for the dataframe and the rest of the lists become rows in the dataframe. Next we clean the dataframe by removing unnecessary text (K, PCT), cleaning up the ‘Date’ column, converting the ‘Date’ column into DateTime format, and converting the non-DateTime columns to float64 format.

**Load**

We establish the connection to the Postgres database set up for this project: bacon\_db. We then attempt to pull the “lean\_hog\_commodity” table in case the data already exists in the database. If the table already exists in the database we append the new data from the most recent extraction and transformation. We then clean up the table by removing rows that contain duplicate Dates with the a bias to keep the duplicate Date that was extracted most recently (ex: If Row A contains the Date “2020-06-05” and Row B contains the same Date, we would keep the Row that was pulled more recently). Now that the dataframe contains the new and old data as well as unique ‘Date’ rows we can add the dataframe to the Postgres database. If the table already exists in the database we replace the table with the new dataframe, otherwise we create the table fresh with our dataframe.

**Data Security**

Along with the scripts that we have provided as part of our work product we have included a config.py file that will contain the API key as well as the access information for the Postgres database. It is important to ensure protection of this config.py file.  
While the data is all accessible via the websites that the Beautiful Bacon utilized, it is still best practices to protect the ETL data in the database.