**Team Members:** Jason Fung, Tosveena Thomas, Bradley Weintraub

**Goal:** To use extract, transform, and load (ETL) functions to aggregate information on cleaning supplies products from multiple vendors and, if possible, compare prices across each.

**Extract**

The team identified Target, Walgreens, and Walmart as retail behemoths of cleaning supplies. Using web scraping methods, each team member will pull product data from one retailer.

Original Data Sources

*Target:* <https://www.target.com/c/cleaning-supplies-household-essentials/-/N-5xsyy?Nao=>

*Walgreens:* <https://www.walgreens.com/store/store/category/productlist.jsp?Erp=72&N=362122&Eon=362122>

*Walmart:* <https://www.walmart.com/browse/household-essentials/cleaning-supplies/1115193_1071966>

Data Formatting

*Target:* Target’s cleaning products were stored under the “Cleaning Supplies” category on their webpage. Each page contained 24 results and there were 38 pages. To perform the web scraping to pull the column information needed for our table, some information was stored within the html of the “Cleaning Supplies” page while other information required looking at the html in the detail section of the item once you clicked on the item. To traverse through each item’s detail page, it required pulling in the link from the html for each page to enter the item detail page.  A code was also created to scroll down to the bottom of the webpages to ensure that the page was fully loaded so that the scraping code could pull all html information.

*Walgreens:* Data was in HTML across multiple pages. Using splinter’s Browser and BeautifulSoup, the data was traversable. From there, key elements like name, item code, and price were available.

*Walmart:* Cleaning products can be found on Walmart’s product website within the ‘Cleaning Supplies’ department. Each result page contains 40 products. Step one was to scrape the first 40 result pages for individual product hyperlinks. 1,600 product hyperlinks were returned; however, many were duplicates (i.e., same product displayed on multiple pages). Only unique hyperlinks -- 1,014 -- were stored into a list. Step two was to scrape each product’s page for product details (e.g., name, universal product code, price, and customer rating). Data was in HTML format.

**Transform**

Data Cleaning and Transformation

*Target*: After scraping the data, storing results into a list of dictionaries, a dataframe was created. There were several items that needed to be cleaned up and reformatted. When the reviews were pulled in,they were pulled in as “4.2 out of 5, 2332 reviews2332”. In order to get the rating score, The first 3 integers needed to be sliced. In order to get the total number of reviews, the string needed to return the number after the word “reviews”. For price clean-up, there was a long section of blank space to the left of all the prices. This blank space needed to be removed. For reformatting the links, the “[www.target](http://www.target)” needed to be added at the beginning of each link.

*Walgreens:* By using web scraping the data gathered from the website.  The attributes that were used to find the data were wag-product and by using these tags the data for product name, price, rating, number of reviews and upc code were found.  After running through the pages for this data it was then translated into a sql data frame.

*Walmart:* Every product detail, or attribute, was scraped and first placed into a list. If a value was not returned, *‘’* was used as a placeholder. Next, all lists were placed into a single dictionary. Then, the dictionary was converted into a DataFrame. The product price was originally stored across 3 attributes -- dollars, decimal, and cents -- so a new price column was created using their concatenation. The aforementioned 3 columns became redundant and were dropped. All rows without a product name were also dropped, and the columns were reorganized. Finally, the DataFrame was written to a csv file.

**Load**

Database

Relational database created through pgAdmin 4. We chose to maintain separate tables for data from each source because the available product characteristics varied. However, the products can be referenced across tables through ‘upc’ where available. This makes it relational.

Tables

Create a table for each Target, Walgreens, and Walmart product data. Columns match the headers within their respective csv file. Import csv data using COPY and FROM statement.

**Analysis**

The ‘upc’ codes within the Target and Walmart datasets were comparable. Walgreens were not comparable, perhaps because they were likely retailer-unique. The Target and Walmart data were joined in SQL and a ‘cheaper\_retailer’ was determined by comparing prices. Of 76 items with matching upc codes, Walmart had the cheaper price for 52 (68.4%) items. Target was cheaper for 15 (19.7%) items. It was a tie in price for 9 (11.8%) items.

**Directory**

* **notebook\_target.ipynb:** Jupyter notebook pulling Target data, writing to csv
* **notebook\_walgreens.ipynb:** Jupyter notebook pulling Walgreens data, writing to csv)
* **notebook\_walmart.pynb:** Jupyter notebook pulling Walmart data, writing to csv
* **data\_target.csv:** csv file with exported Target data
* **data\_walgreens.csv:** csv file with exported Walgreens data
* **data\_walmart.csv:** csv file with exported Walmart data
* **create\_load\_database.sql:** Query used to create tables and import data from csv files; also compares price data between Target and Walmart
* **price\_comparison:** csv file with exported price comparison data