**TRANSFORMATION**

1. SALES ZIP FILE (5 YEARS):

For both sales and price, the day columns were transposed so the sales and prices changes can be viewed by row. The dataset did not specify when the data was collected so it was impossible to determine the exact start or end date of the data. After doing some research it was discovered that Walmart is closed only on Christmas Day. Using this information, it was determined that there were days with 0 sales for all items and they were 365 days apart, one range being 366 days apart likely due to a leap year. This helped determine what day and month the sales were during. The only thing that could not be determined was the year but knowing the day and month can help in identifying trends and how sales is impacted from price changes. After determining the date, day and month were included into the dataset. Columns such as state\_id and cat\_id (category) were also included in replacement of their strings to help reduce storage space. Since the product ID was a string, to reduce storage a product ID table was created with integer id values. Various columns were also deleted as there was not much information regarding it and their values. After cleaning it resulted in the below table:

* SALES TABLE
* UNIQUE PRODUCT ID TABLE
* STATE ID TABLE
* CAT ID TABLE

For the price file, a similar process was done like the sales dataset. Item, category, and state were replaced with their integer ID. The day and month were also included in the dataset. The dataset for the price changes is very large and in order to gain insightful information it was decided that instead of showing price changes in every day a “Markup” and “Markdown” column was created. These columns would count the numbers of markdown and markup every 6 months over the 5 years. It was decided to work initially with 6 months to simplify the data but next time every month may be considered instead to have a more detailed look at price changes. This was calculated by sort the dataset by the product name and day. If the next row price increase it would be counted as a Markup, if the next row was lower then it would be counted as a Markdown, and if there was no change it would continue onto the next row without counting anything. The function would sum these counts and then reset the sum once it reached 6 months and this process would iterate. From this function it resulted in the following table:

PRICE MAKRUP MARKDOWN TABLE (6 MONTH/5 YEARS)

2. ONLINE PRODUCT DATABASE:

Various columns were deleted such as URL link, UPC and timestamp of data collection. The columns were deleted due to not being able to gain insightful information from them. The main columns cleaned was the categories column as it was formatted to include all its sub-categories. It was decided to focus on categories and not the sub-categories as the other datasets did not have the column so the online product database alone would not be enough to get important findings from sub-categories. After splitting the first category from the column, the categories column was overwritten with only the first category. Once that was completed the string was then replaced with a category integer ID. The cleaned table is the following:

ONLINE PRODUCT TABLE

3. WALMART SALES 2010-2012:

The 3 datasets including features, stores, and training were inner merge on stores to eliminate NA’s. After columns were deleted such as the price markdown columns due to majority of the values being NA’s and it was not determined what the values meant for stores. The data was also broken down from store and category level but due to the category being a category integer ID and being unable to determine the actual category, it was decided to group the weekly sales by store to remove category.

4. API HOLIDAY:

The holiday table required no transformation or cleaning as the data frame was created during the extraction process of getting the holiday dates from the holiday API. This table was used to merge on the Walmart sales 2010-2012 dataset to identify when holidays occurs and how it impacted the weekly sales.

5. WALMART DAILY STOCK DATA:

The data was filtered on the beginning and end data of the 2010-2012 Walmart dataset. Since the stock data is daily, a week column was added to identify the day that follow under that week. This was done using the datetime and timedelta module and function.

6. walmart\_com-ecommerce\_product\_details.csv:

Initially, the Walmart e-commerce category data (which was a composite field of up to 6 levels of category descriptions) was separated and normalized in category level tables. A table that ‘mapped’ the categories was created.

To further reduce the size of this data, a single e-commerce table comprised only of the first category level and sale price data were retained. This category level is sufficient for mapping to the existing data.

7. WALMART JSON DATA:

A function was created to see the data types and total NAN values and their percentage for each column. The function showed that the Walmart.json has around 15% NAN values for openDate and less than a percent null values for timezone and since these two columns will not be included in the analysis, they left intact to use other information of those rows. Then it filtered by storeType to include just the Walmart Supercenter and Walmart and relevant columns had selected from it with the name of store database. The column STATE and ID of state\_id data were renamed to state and state\_id and then it was merged with store on state.

8. Walmart Market Share:

First the data types were determined. The '%' sign were removed from the values in the 'MARKET\_SHARE' column, then datatype was converted from object to integer.

The values (AP style state abbreviations) in ‘STATE’ column were replaced with the FIPS state codes, using the 'id\_code' dictionary created prior.

The ‘STATE’ column renamed to ‘STATE\_ID’ and set it as index for the final dataframe.

9. Walmart, Amazon, Target and Costco financial database:

The first step was to transpose the table, switching index with columns. Since first column was set to index and the dates were positioned as the column names on the first row, when applying df.T to each dataset the dates then became the indexes and the categories became the columns. So, the index was then reset using df.reset\_index(inplace=True) on each dataframe. This resulted in a new column with the dates, however, they were under a column named “index” so those column names needed to be changed to “Date”. Each dataset also included some values for TTM (Trailing Twelve Months) which muddied the desired clarity so they were dropped using df[:-1]. The Target dataset also had figures for 2020, so those were dropped with df[:-2]. To take a look at which categories had the most complete data, a df.dropna() was performed on each. Taking a look at the resulting dataframes, the categories that remained in all 4 datasets were considered for further transformation and analysis. The categories decided on were:

* Revenue in Millions of USD
* Net Income in Millions of USD
* Earnings Per Share in USD

The next issue was that the dates were presented as a year-month pair. Some of the datasets had different months, but the figures remained consistent. In order to have consistent data for a date on which to merge, the months were removed using a dt.strftime(%Y) function. Once the dates were consistent, the tables were ready for a merge along the Date column. Merging two tables at a time, columns were renamed to distinguish Walmart, Amazon, Target, and Costco's category columns respectively. All four tables were merged into one. The Target dataframe unfortunately did not have any values for 2010, so an outer join had to be performed to merge that into the main merged table. The last step was separating the merged table into the three desired tables. This was accomplished by creating new dataframes for each category and selecting the relevant columns to group together with each including Date as the first column. From there, the five dataframes had their indexes set as “Date” to provide a key for the rest of the data so they can be related to each other and any other data linked by a year value for date.