**ETL Process**

**Sources**

1. prices.csv
2. sales.csv
3. walmart\_com-ecommerce\_product\_details.csv

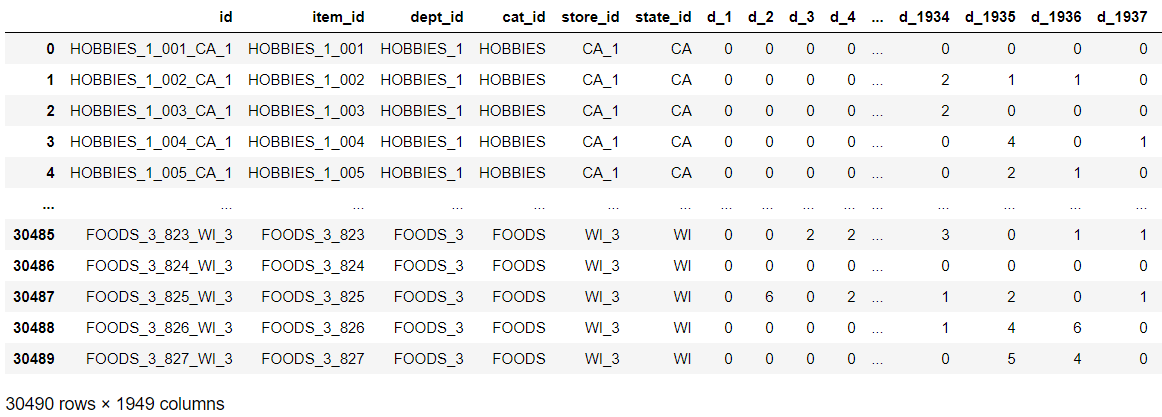
The prices and sales data were retrieved from here and the e-commerce data came from here. The prices/sales datasets did not include retrieval time information from the author and the tables were not normalized or ready to be loaded into a database.

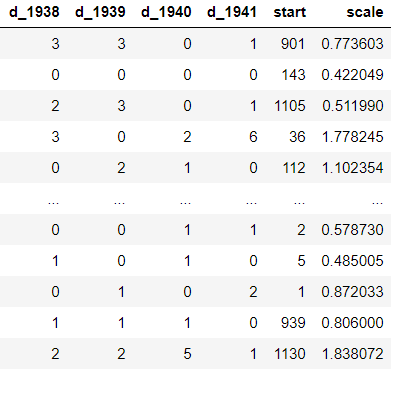
The sales/prices datasets included sales data from 9 stores across 3 states: California, Wisconsin and Texas. There were 3,048 items belonging to three categories: Hobbies, Food and Household. The items did not have any descriptions.

The e-commerce dataset is a sample of 30,000 products with their categorization and prices.

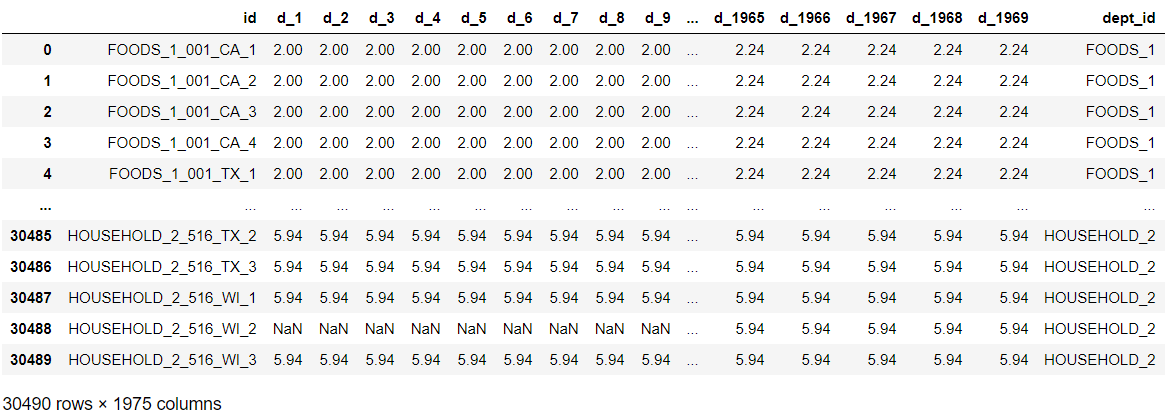
**Raw Datasets**

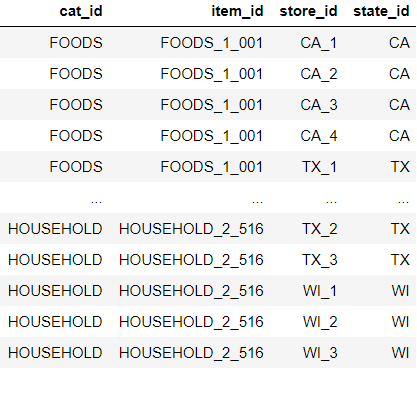
Sample of prices.csv data:



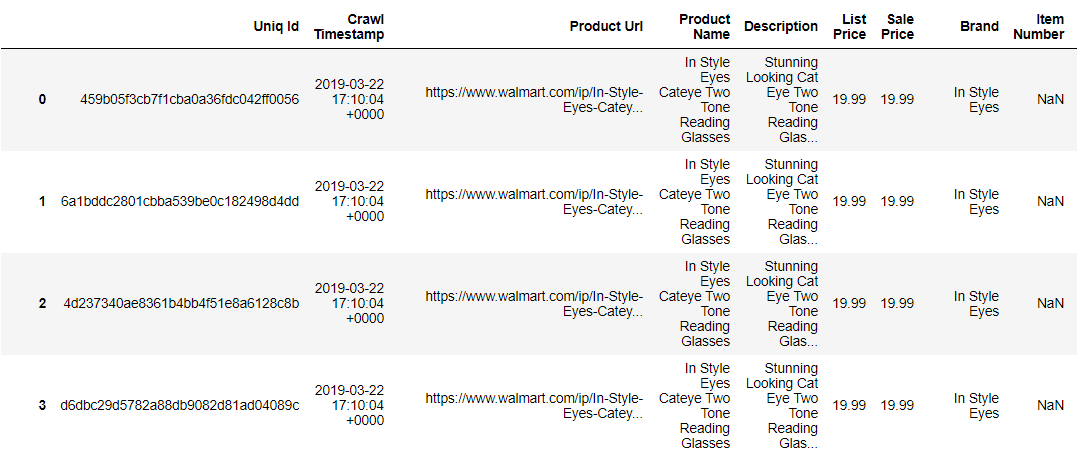


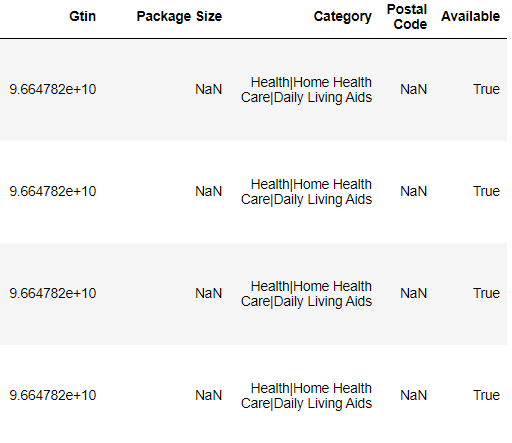
Sample of sales.csv data:





Sample of walmart\_comm-ecommerce\_product\_details.csv:





**Extraction**

The data were available in .csv format and read into Python dataframes for transformation and loading.

**Transformation**

1. sales.csv

**Normalization of category, department, item, store and state**

The primary key of the sales data was an alpha-numeric, composite key comprised of the item\_id + store\_id for most records, however, it was the store\_id + ‘\_X’ for subtotaled records.

The item\_id is also a composite ID using the dept\_id description followed by a serial number. The dept\_id is a composite id of the category (cat\_id), which is a category description followed by a serial number.

Finally, the store\_id is a composite ID comprised of the state\_id (two-letter state abbreviation) followed by a serial number. The state\_ids were set to match the state numbering of another datasets, namely TX=48, WI=55, and CA=6.

To normalize the data, five separate dataframes were created with the unique text values of each column. The indexes of those dataframes became the primary keys of their respective tables. The keys were then inserted into the sales table as foreign keys.

**Dropping of the start and scale columns**

The start column denoted the period in which the first observed sales occurred. This column was dropped as this information could be queries from the future transposed sales table. The scale measure could not be reconciled and was also dropped.

**Transposition of the d\_1 to d\_1941 columns**

The sales dataset included daily sales summarized in daily columns, numbered d\_1 to d\_1941. This structure does not make analytics easy and was transposed to take advantage of the performance of an SQL database over normalized tables with large number of rows.

The day column was created and assigned the numeric value of the day. Furthermore, during the transposition, rows with 0 sales were dropped. Dropping 0 sales rows reduced the row count from over 83 million to under 19 million.

1. prices.csv

Prices were indexed using the same composite index as the sales data. This index was broken out into the 5 categories used in the sales table, matching foreign keys were assigned to the table and the primary key was the same as the sales data (based on the composite index).

The price data also required transposition as it also had daily prices as columns. Once this transformation was complete, there were 47 million rows of price data.

Price data was joined to the transposed sales data to minimize space in the final tables to be loaded in PostgreSQL.

The price data was further transformed into a table that showed date ranges the prices were effective at the state/store/category/department/item level. This table was joined to itself to get the price from the previous period, so that an indicator of whether the price increased or decreased could be calculated.

This price range/change table had a little over 80,000 rows, substantially smaller than the original data.

1. 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.

**Load**

The 5 sales category tables, main sales table, price range/change table, and ecommerce data were then loaded into a PostgreSQL database using psycopg2 and StringIO. The poor performance of sqlalchemy with very long datasets lead for a search for a more efficient way to load these tables. By using StringIO, the dataframes can be stored in csv in memory and read in chunks into PostgreSQL using COPY STDIN. Based on time estimates, this was at least 20 times faster for the very long tables (originally 80+ million rows).

The current structure of the data allows for indexes to be created in SQL and easier querying of date ;data.

**Determining the date of the sales/price data**

To determine the date of the data, the sum of sales by day were reviewed. There were 5 data points with near 0 sales. Upon further research, Walmart stores are closed only on Christmas Day. The days with almost no sales were separated by 365 days for 4 dates and 366 for one.

Using the timing of the leap year (366 days), we determined that the data set started on January 28th. Because the leap year happened in the second year of the dataset, the start years of the could be 2015, 2011, 2007, 2003, etc.

**Problems**

The sales/price datasets are limited in a number of ways:

1. the data set covers 3,048 unnamed items, from 7 unnamed departments belonging to 3 unnamed categories; sold and priced at 9 unknown stores across 3 states.
   1. there is no information about how these items were selected or if the 3,048 items are all of the items that belong to those 7 departments
2. the year of the dates remains unknown
   1. even though we figured out the dataset starts on January 28th, not knowing the exact year is a limitation (2015, 2011, 2007, 2003, etc.)
3. the format of the data required extensive manipulation
   1. creating columns by day for sales data made extracting the day information for different periods more difficult. After transposing this data, removing all repeated text indexes, and all days with 0 sales, the dataset was still over 500MB and over 18 million rows.

**Ecommerce**

1. The e-commerce dataset has 30,000 rows and appears to be a sample from a larger file. Without knowing the sampling method, it is hard to make generalizations about e-commerce products using this data.
2. The e-commerce data category (1st level) is the only data available at present to join the sample product information to the groups other datasets.

**Future Work**

The sales/prices data can be analyzed in many different ways:

1. Sales patterns across the items and departments can be analyzed by day/week/month of the year
2. Prices can be analyzed for patterns in sales/markdowns and increases and sales volumes can be analyzed to see if the change in price affected sales at the stores.
3. Departments can be reviewed to see which contributed the most to growth in sales and the relationship between price and sales can be reviewed more to see if lower cost or higher cost goods make up the most sales growth over the 5 years.