Data Source

The data I am using to conduct my research is sales data from a big box electronics retailer for year 2019. I have the data stored locally in .csv format separated into multiple files, one for each month of sales records for the year. I had the task of aggregating this data into one workable file, and arranging the data in a Pandas data frame was the best way of doing this.

Pre-Processing

My first step was to read in all .csv files and combine them into one data set. Using **os.listdir**, I was able to get the file name for each file, then after creating an empty data frame to store each individual file, used the concatenate function of pandas to fuse them into one data frame.

I noticed some NaN values in my data set, so I had cleaning to do before getting started on analysis. Due to the size of my data set with over 300,000 rows, I decided it would be appropriate to drop rows with missing values.

To help with analyzing data by month, I decided to add a column showing only the month of each sales transaction, by extracting information from the order date column. I did this by converting the order date column to a string and used indexing to take the first two characters of each date, which was the numeric month. I ran into a problem when first attempting this, because an error occurred because there were some string values (which would not convert to integer) in the order date column. I created a new data frame made up of only values that did not contain strings in the order date column.

Since I knew I would be interested in looking at sales by city, I wanted to augment the data with a column showing only the city of the sale, extracted from the sales address column. Using the **‘.apply’** function and a lambda function I was able to split the order address column where the commas landed, and take the characters between the commas, which happened to be the city name, and place them into their own column.

One of my questions was related to sales at different times of day, so columns indicating which hour of the day as well as minute of the day that each sale took place was necessary to add. This was done by converting the data type of the Order Date column into datetime objects, which was a simpler approach than trying to parse the dates and times as strings.

Another necessary task was to add a column showing the value of each sales transaction, which multiplied the quantity of an item purchased by its unit price. An error occurred when attempting this, as the values in the price column that appeared to be doubles were strings.

Business Questions

# Question #1 – What was the best month for sales?

We can answer this question by grouping by value as displayed in the month column and summing the values. We can display results in tabular format first, then create a bar plot to accompany our findings. The results showed that December was the best month for sales.

# Question #2 – Which city recorded the most sales?

We can answer this question in similar fashion as we answered question #1. We will group by city and sum the sales totals to get the answer for which city did the most sales in 2019. After displaying the data in tabular fashion as well as with a bar plot, it is clear to see that the San Francisco location generated the most sales.

# Question # 3 – What time should advertisements be displayed?

In asking this question we are focusing on advertising strategy—what is the best time of day to display advertisements to maximize the chance of a customer buying a product? This question is best answered visually, so I plotted sales in a line graph to see which times of day produced the greatest number of sales. The result was quite logical, in that most sales occurred when the majority of consumers would be least busy-- near the lunchtime hour around noon, and the evening hour around 7pm. With this information, I reasoned that the best time to display advertisements were in the hour just before, during, and after these peak sales times.

# Question # 4 – Which products are most often sold together?

I can answer to this by paying attention to the order ID numbers in our data set. Products sold together will have the same order ID numbers, so I will look for duplicate ID numbers and see which products correspond to those order ID numbers. I can filter the data by using **[‘Order ID’].duplicated.** Using the **itertools** and **collections** libraries I was able to filter the results to view common combinations of items in a single order. Common pairs of items included Apple iPhone and Lightning Cable, Google Phone and USB-C charging cable, iPhone and Wired Headphones, and Google Phone with Wired Headphones. We could make use of this information by running promotions on items that tend to fall into the same sales basket.

# Question #5 – What product sold the most, and why?

I’m interested in which product sells the most, and also want to come up with a reason why. By taking a high level view of the sales data, there appears to be a pattern of cheaper items selling the most, and more expensive items selling the least. I aim to prove this hypothesis to be true by plotting the sales data on a chart with two y-axes—one for sales quantity, and another for price of each item. Looking at the plots, my hypothesis is mostly supported with a couple of exceptions. Batteries are an item in high demand, and they carry a price significantly lower than the other electronics items in our sales records, so as expected, those sold the most. The LG Washing Machine and Dryer were examples of items that were ordered far less frequently while carrying a high price, which was an expected result in my hypothesis, but the Macbook Pro being the most expensive item was not the least-ordered, as the hypothesis would suggest. We can conclude that demand plays a important role than price when it comes to total sales of an item.