# Background

The purpose of this paper is to describe an extract, transform, and load (ETL) project. ETL is the process of sourcing data from one location, extracting it from that location, transforming it from its original structure, and loading into a new destination (see figure 1). The use case developed in this project focuses on stock trading data.

Figure 1: ETL Process

A picture containing text, room, scene, gambling house

Description automatically generated

According to Companies Market Cap (retrieved on 12/7/2021), the two largest companies by market capitalization are Google and Apple. The goal of the project was to perform a comparison of these two companies stock performance over a 15-years period. The focus was on the ETL portion to support an analysis at a later point in time. In other words, the actual analysis (e.g., plotting, etc.) is not in scope of this project.

# Datasets

Two datasets were identified and located from Kaggle.com that pertained to Apple and Microsoft trading data (<https://www.kaggle.com/nikhilkohli/us-stock-market-data-60-extracted-features>). The timeframe for both company stocks that were analyzed were between October 2005 to August 2020. The original format of the data was in a .csv format.

# Coding

## Dependencies

Dependencies included tools from pandas, datetime, sqlalchemy, and psycopg2 (see figure 2)

Figure 2: Dependencies

Graphical user interface, text, application

Description automatically generated

## Evaluating the data

The Microsoft data (MSFT.csv) and the Apple data (AAPL.csv) was imported and the data was previewed using Pandas and there were ~64 columns of data (see figure 3). The first thing that was done is that the data was dropping all columns with the exception of six columns which was the trading date, the opening price, high and low prices for each session, the closing price, and the volume that was exchanged for each session (see figure 4). These were still treated as separate data frames for each of the companies.

Figure 3: Preview of data as originally sourced.

Table

Description automatically generated

Figure 4: Filtering the data

Table

Description automatically generated

## Data Cleansing

Following this, the data was evaluating to identify any null values which, had there been, we would have dropped those rows. In this case, there were no null values (see figure 5). Once this was known, a pd.merge script was run to join the two data sets using the "Date" field which only brought in those columns mentioned above as the others were dropped. Because the join results in two columns with a "x" and "y" for each one. The columns were renamed consistent with how we planned to import the data into Postgres (see figure 6).

Figure 5: Checking for null values.

Graphical user interface, text, application, email

Description automatically generated

Figure 6: Joins and Renaming Columns.

Table

Description automatically generated

The next step was evaluating the data types. With the exception of volume, every column reflected as a float. The volume was an integer, and the "date" was an object. Therefore, the only column that required a change in the data type was the "date". The datetime code was used to change the data type for that particular column (see figure 7).

Figure 7: Viewing and changing datatypes.

Table

Description automatically generated

## Exporting to Postgres

For the export process, sql was used for this as it was clear that any future use of the datasets seemed relational in nature (e.g., the use of primary keys, etc.). Sqlalchemy's "declaritive\_base" and "create\_engine" features were used for creating a Postgres path and exporting the DataFrame into sql file that can be imported into Postgres. Furthermore, a sql query was generated that would streamline someone's ability to create a table with this new data (see figure 8). A .gitignore was used for the "secret" file that included the password. If someone were to reuse this code, this file would need to be generated as well as a predefined database named "tech\_stocks". Lastly, the final data frame was also exported as a csv ("finalTechStocks.csv").

Figure 8: Import into Postgres (i.e., Load)

**Text

Description automatically generated**

# Conclusion

As mentioned, this exercise was intended to serve as an illustration for ETL that could then support a follow-on analysis, such as the historical performance comparison between these two companies. Should this happen at a later time, the data is prepped for the end user to conduct an analysis.