**From:** Piruz Alemi & Zhongping Yang

**Subject:** ETL Project (Extract, Transform, Load)

**Date:** Jan 29th, 2020

**Team Effort**

This project was conducted by the joint efforts of [**Piruz**](https://app.slack.com/team/UNS8GPH6E) **Alemi** & [**Zhongping Yang**](https://app.slack.com/team/UNTA6KQE6)

**Project Proposal**

Before we started writing any code, we reached the conclusion to research the *Derivatives Market Option pricing*

**Finding Data**

The novelty of our project was that we used not only CSV data sources, but we were also able to **Scrape Option Prices of SPY** directly from the screens. We used the following sites to use as sources of data for our research including quantopian with its embedded Jupyter Notebook:

* [data.world](https://data.world/)
* <https://finance.yahoo.com/>
* [Kaggle](https://www.kaggle.com/)
* <https://finance.yahoo.com/quote/SPY/options?p=SPY>
* <https://www.quantopian.com/research>

For SPY we accessed the CSV Stock Prices + the SPY Options on the Screen as in:

A screenshot of a cell phone

Description automatically generated

We scraped the data direct from the web screen!

**Data Cleanup & Analysis**

Once we have identified your datasets, we performed ETL on the data:

* The type of transformation needed for this data (cleaning, joining, filtering, aggregating, etc) were:
  + The history of Option prices is not available. We captured this data daily
  + Key variables of interest were the Contract Name, Strike price, Stock Price, Implied Volatility
  + Cleaning:
    - We cleaned IV, as the Implied Volatility included the data as a string in %
    - We stripped DateTime from its Time
    - We transformed strings in variables expecting floats to NaN, using Numpy Library as np
  + We joined the CSV file of historical prices of Stocks with our daily Screen Scrapings
  + We automated the latter task through a task manager
* We created a series of charts showing the relation of implied Volatility, Volume, Strike Price and Stock Price as it relates through time (Date) as a sample like the following. For brevity we have not included the full report here.

A picture containing screenshot

Description automatically generated

* We loaded our Joined data set into two production databases as both (relational AND non-relational).
* The final tables (SQL) and collections (MongoDB) may be used in the production database. For MongoDB as in:

A screenshot of a cell phone

Description automatically generated

**A screenshot of a computer

Description automatically generated**

**We finalized by:**

* **L**oading: the final database, tables/collections in both SQL & MongoDB. This was chosen, as Historical Option Prices is not available. We also needed the flexibility to add other types of data.
* **T**wo sets of data bases were used, to compare the “efficiency” of each database + coding. The Flexibility of MongoDB was preferential to coding vs. pre-defining each table & variable in SQL. However in the final analysis we understood the power of directly loading a Data frame into SQL – with a single command. So was also the power of a single line command for scraping Option prices. Clearly in terms of coding we prefer Pandas SQL interfacing.
* MongoDB offered us additional flexibility as we were dealing with a variable “Collection”, including future correspondences and additions on Derivatives markets that could potentially be scraped from other Web pages. Furthermore, parts of our data was changing every 3 seconds, and parts on a daily basis and parts never changing.
* We also used <https://cronitor.io/docs/using-cronitor-cli> to schedule the task of scraping Options data daily and appending to a headless CSV file, except for the first day.

Please see our report on Github or follow our link on Bootcampspot!

Respectfully,

Team: Piruz Alemi & Zhongping Yang