**Extract Transform Load (ETL) Project**

Dec 18, 2019

*Team 2:*

Charles Zhao (Extract),

Dan Botari (Transform),

Sushaine Goyal (Load),

Jenny Bersinic (PM)

**Proposal:** Through this process we are looking to set up for the final project and will therefore look to leverage stock prices and news articles to create a unified dataframe for the ETL project. This will allow for an eventual measurement of the affect sentiment (news, etc.) has on stock prices. This sentiment analysis will be correlated with stock price information to determine whether there exists a correlation between a sentiment expressed within news articles and the stock price movement.

For the ETL project, we are focusing on one company, Tesla, to conduct this initial analysis.

**Data Formats**

*Yahoo Finance* stock API will be leveraged to retrieve daily stock prices including ‘date’, ‘high’, ‘low’, ‘open’, ‘close’, ‘volume’, ‘adjusted close’. *News API ‘*articles’, ‘title’ will be leveraged to create a string which will be used to conduct sentiment analysis, using ‘bag of words’ methodology.

**Data Extraction**

We used Pandas within Jupyter notebook to validate data we were looking to extract. The Twitter API & BING News were considered but both were unable to pass the ‘published date’, therefore we used the [News API](https://newsapi.org/) to pull articles based on Tesla coverage, for the last 30 days.

The response format leveraged for the news API was JSON. Title & description were looped through separately, to create two lists:

* List 1 produced top 10 titles per day (300 total)
* List 2 produced top 10 associated descriptions (300 total)

We then chunked the data to associate 10 titles & descriptions per day in order to match published dates.

A screenshot of a cell phone

Description automatically generatedA screenshot of a social media post

Description automatically generated

The output was a data frame which included ‘date’, ‘title’, ‘desc’:

A close up of text on a white background

Description automatically generated

We used pandas\_datareader in Jupyter notebook to collect Tesla (TSLA) stock data from Yahoo Finance, for the last 30 days.

A screenshot of a social media post

Description automatically generated

Once the dataframes were created, they were both exported into CSVs: stock.csv, new.csv.

**Data Transformation & Cleaning**

Using pandas within Jupyter notebook, we imported the CSV files created in the previous extraction into pandas data frames.

**News Transformation:**

We examined the dataframe and cleansed, combining title & description into a new column called ‘content’.

A screenshot of a social media post

Description automatically generated

**Stock Transformation:**

We examined the dataframe and no cleansing was required.

In order to merge the two data sources, we needed to create a date dimension table that would allow us to map the two data sets onto a common date framework.

A screenshot of a cell phone

Description automatically generated

Once this table was created, we needed to create additional logic that would map Saturday & Sunday date information to the next Monday in order to align to business days vs calendar days.

We merged the date dimension dataframe with the news dataframe to create a new dataframe as step 1 of the overall data merge process. A screenshot of a social media post

Description automatically generated

From this newly created dataframe, we then extracted the columns we were interested in: ‘date column’ (common column, primary key), ‘contents column’ (article contents of the news stories). We then grouped by the ‘date’ field, aggregating the ‘content’ fields of rows with similar dates. The output of this step gave us a single row for each date (unique date, unique content).

We took this new dataframe and merged it with our news data frame to create the new master dataframe containing content from the date dimension table, the news dataframe and the stock data frame.

We cleaned the master dataframe, resetting the index to ‘date’ in preparation for the upload to postgres.

We grouped the news data by business day we left joined the news data to the stock data, by date. In order to format the news data into a ‘bag of words’ we merged ‘title’ & ‘desc’ into one cell, per day.

A close up of text on a white background

Description automatically generated

Challenges:

1. News dataframe contained information for every calendar date in the selected range (30 days), whereas the stock dataframe only contained information for the weekdays in the same period. This created a mismatch of the field targeted for the primary to join the two tables. The use of the data dimension table as the common date reference, allowed us to combine these two dataframes along a common index.
2. We had weekend news stories which needed to be mapped to the Monday following the weekend, in order for us to use this info later in our sentiment analysis of the impact of news articles upon stock price. This involved writing logic to re-code weekend news content and map it to the following Monday date reference. We then re-grouped the data in the dataframe, such that the weekend news content was concatenated with the following Monday news content to create a dataframe containing weekday only dates, and article content along with stock price information.
3. In the above step, the one issue we did not correct for was the presence of a holiday during our reference timeframe which created a ‘missing values’ for the stock related information on that date. To correct this would require specialized targeted code which will be addressed in subsequent development efforts.

**Data Loading & Storage**

In preparation to load the output of the Jupyter dataframe, we created a database in postgres (relational database) with a single table called ‘stock information’ building the schema, with a primary key of ‘date’, to receive the dataframe from Jupyter. Once we created the schema, within Jupyter we created the database connection to the postgres database and confirmed the table name. Upon confirmation that the table existed, we pushed the Jupyter dataframe to the stock info table in postgres. We confirmed the existence of the content of the dataframe in the postgres table by running a ‘select all’ query against the table.

**A screenshot of a computer

Description automatically generated**