**‘FUNDING SECURED @ 420’**



*An extraction, transformation and loading team project focused on Tesla, Spacex and the Twitter account of Elon Musk*

Tesla Evaluation

# Authors: John Michals, Jeremy Randolph, Kenneth Reed, Khrystyne Vaughan

# Purpose

After narrowly escaping a potentially devastating short position, the bank’s equity trading desk approached our business intelligence group with a request for data surrounding their largest equity position, Tesla. The traders, nearly burned by the now infamous, “Funding Secured” tweet by founder and CEO Elon Musk, requested all available information on Musk’s twitter activity, any available information on Spacex, Musk’s aerospace transportation services company and direct influencer of Tesla stock price and finally, the Tesla stock price history since the company turned public in the summer of 2010.

The traders had several questions they were attempting to answer with our provided data. Those questions were:

* *Does price volatility increase, decrease or remain flat in the lead up to or the wake of a Spacex launch?*
* *Does the outcome of a Spacex launch have a direct effect to the price of the stock?*
* *Does the stock price react to the private twitter account of the founder and CEO?*
* *Does the popularity of a given tweet have a measurable effect on the stock price?*
* *Is the volume of Tesla stock traded subject to change when Musk tweets?*

Through several extraction, transformation and loading techniques, our team was able to provide the equity desk with the requested data in a tabular format stored on a MySQL database. The table provided all the relevant data that was reliably retrievable to provide the framework for which to answer the traders’ inquiries.

# Extraction

Stock data sets are some of the more readily available and reliable data sets published. Utilizing the data available on yahoo finance, a media property providing financial news and data for the Yahoo network, our team was able to export the Tesla stock data, up to the requested date of October 29th, 2018, to a csv file. The file included Tesla’s daily stock performance since June 29th, 2010 and included several columns that included, date, open, high, low, close, adjusted close, and volume. Below is a link to where the data was originally obtained:

Tesla Stock Price: <https://finance.yahoo.com/quote/TSLA/history/>

The SpaceX data was slightly more difficult to obtain but thankfully, our team is frequent visitors to Kaggle. Kaggle is the world’s largest community of data scientists and contains a plethora of available data sets. On Kaggle, our team was able to find a loaded csv file that contained several columns of data on Spacex launches. The columns were, time, booster version, launch site, payload, payload mass (kg), orbit, customer, mission outcome and landing outcome. Below is a link to where the data was obtained:

SpaceX Launch Data: <https://www.kaggle.com/scoleman/spacex-launch-data>

The Twitter data presented the most interesting and unique challenge for our group to extract. After all, there aren’t exactly user Twitter tables floating around on Kaggle or Yahoo. However, by utilizing several Python libraries, namely Tweepy and pandas, as well as the Twitter API, our group was able to successfully extract the most recent Elon Musk tweets. Additionally, a loop was used to extract the tweets from Twitter, and then append each tweet into an array. Only selected fields were extracted, specifically, id, date, text, favorite and retweet counts. The below code highlights the Twitter API utilized, as well as the loop executed, to extract the tweets:

auth = tweepy.OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_token\_secret)

api = tweepy.API(auth, parser=tweepy.parsers.JSONParser())

*# Target User*

target\_user = "elonmusk"

*# Tweet Texts*

tweet\_texts = []

*# Create a loop to iteratively run API requests*

for x in range(1, 200): #20 per page

*# Get all tweets from home feed (for each page specified)*

public\_tweets = api.user\_timeline(target\_user, page=x)

*# Loop through all tweets*

for tweet in public\_tweets:

*# Print Tweet*

#print(tweet["text"],tweet["created\_at"])

*# Store Tweet in Array*

tweet\_texts.append([tweet["id"],tweet["created\_at"],tweet["text"],tweet["favorite\_count"],tweet["retweet\_count"]])

Through our loop, our group was able to capture 3,229 total Tweets. These tweets were loaded into a dataframe and columns were assigned to represent the various captured data. This data was exported to a csv, however, the data was ultimately abandoned given its lack of relevance to the traders’ initial inqueries. However, the tweet data frame was further transformed in an effort to create a table that would better serve the traders’ intentions.

The first, additional transformation was creating a column “Popular Sum” that would be comprised of the tweet column “Favorite Count” and “Retweet Count”. The purpose was to generate a singular measure of the popularity of a given Musk tweet. Following this transformation, our group added a date column utilizing the datetime library. However, given that Musk tweeted several times a day in certain instances, we had a table that included a number of tweets for the same day. Given that our traders’ were only interested in the most potentially impactful tweets, and not the lesser popular ones, we performed a sort on the dataframe by Date and Popular Sum, from most recent, and then dropped any duplicate dates. The code for these essential transformations is below:

tweet\_ETL\_df["Popular\_Sum"] = tweet\_ETL\_df["Favorite\_Count"] + tweet\_ETL\_df["Retweet\_Count"]

tweet\_ETL\_dt\_df['Date']=tweet\_ETL\_df.Tweet\_Date=pd.to\_datetime(tweet\_ETL\_df.Tweet\_Date).dt.strftime('%Y-%m-%d')

sorted\_df = tweet\_ETL\_dt\_df.sort\_values(['Date','Popular\_Sum'], ascending=False)

sorted\_distinct = sorted\_df.drop\_duplicates(['Date'])

# Transform

Utilizing the Python library Pandas, inside of Jupyter Notebook, our first step was to import the newly extracted csvs for the stock prices and the Spacex launch data. Below are the codes executed to perform these actions on the Spacex and stock data:

spacex\_file = "Resources/spacex\_launch\_data.csv"

teslatwo\_file="TSLA \_to\_29th.csv"

spacex\_df = pd.read\_csv(spacex\_file)

teslatwo\_df=pd.read\_csv(teslatwo\_file)

Now that our data was readable into two independent Pandas Dataframes, our team then began a fairly straightforward transformation process. For Spacex, the transformation included limiting the specific columns of data that were relevant for answering our equity traders’ initial questions. We then renamed the columns to correspond properly to the database that we eventually exported our data into (more on this topic in the next section). Our next step was to then perform a drop on any duplicate data that existed and lastly to set the index of the table to the id that was created. Once the transformation was finished, our team pushed the new dataframe to a csv file. Below is the code executed to perform these transformation steps and obtain our Spacex final table:

spacex\_cols = ["Flight Number", "Date","Launch Site", "Payload", "Customer", "Mission Outcome", "Landing Outcome"]

spacex\_transform = spacex\_df[spacex\_cols].copy()

spacex\_transform = spacex\_transform.rename(columns={"Flight Number":"id",

"Date":"real\_date",

"Launch Site": "launch\_site",

"Payload":"payload",

"Customer":"customer",

"Mission Outcome":"mission\_outcome",

"Landing Outcome":"landing\_outcome"})

spacex\_transform.drop\_duplicates("id", inplace=True)

spacex\_transform.set\_index("id", inplace=True)

spacex\_transform.head()

The cleanup and transformation on the Tesla stock data was slightly more involved. Similar steps performed with the Spacex dataframe were copied over for the Tesla stock data. Two additional columns were created and added to the dataframe from the remaining data to further answer the equity traders’ questions. The first column was the calculated range for the given day. This range, the low price for the given day subtracted from the high price for that same given day, was added to better illustrate to the equity team the intraday movement of the underlying stock price. The thought process behind building this column was that the traders would need a quick reference column to run analysis on that would measure the intraday activity of Tesla stock. Additionally, to better approximate how the intraday movement related to the actual underlying price of Tesla stock, we calculated a “Range percent”, which was the range divided by the closing price. That value was then multiplied by one hundred to get a value that would reside between zero and one hundred. The logic behind the calculated value is the greater the value in the range percent column, the more volatile that particular trading day was. This would allow traders to attempt to answer if a given trading day was more or less volatile given a particular tweet by Elon Musk or a Spacex related event. The code for these two additional columns is below:

tesla\_transform["range"] = tesla\_transform["high\_price"].subtract(tesla\_transform["low\_price"], fill\_value=0)

tesla\_transform["rng\_perct\_close"] =(tesla\_transform["range"]/tesla\_transform["closing\_price"])\*100

Following the creation of these two data columns, the dataframe was ready for export and was pushed to a new csv file for ultimate loading into our database.

[TAKE KEN’s PORTION AND SPLIT HERE]

Tweet Data:

When it comes to Elon Musk’s tweet, we decided to use the most popular tweets of the day. If the tweet had the most retweets and saved as a favorite the most, we stored those in their own file.

# Load

1. What can be done with multiple tables
2. What we can use from the table
3. Why you chose the fields you did