Amber van harskamp and rachel zhang

ELT REPORT

Does tweeting about a company have an effect on its stock value?

DATA ANALYTICS AND VISUALISATION BOOTCAMP

MONASH UNIVERSITY

TABLE OF CONTENTS

1. INTRODUCTION

Does tweeting about a company have an effect on its stock value? In order to answer this question, a database containing tweets and trading information relating to particular companies on the NASDAQ was created. The companies examined were Apple, Google, Amazon, Tesla and Microsoft since they are the most well-known and hence expected to be tweeted about frequently. Moreover, in an effort to ensure enough data was available to be able to draw sufficient conclusions, a five-year period from 2015 to 2019 was collected.

This report outlines the process for the creation of the database, including: the extraction of the data, the transformation and the loading of said data into a suitable data platform for query.

2. METHODOLOGY

A. EXTRACTION

i. Tweet data

The data, which was embedded in three separate CSV files, was extracted from Kaggle.com[1] on 28 July 2021.

[1]https://www.kaggle.com/omermetinn/tweets-about-the-top-companies-from-2015-to-2020/tasks?taskId=2825

The first CSV is titled *Company* and contains two columns, 'ticker_symbol' and 'company_name'. The 'ticker_symbol' relates to the symbol the company is referred to on the NASDAQ, which is made up of 4-5 uppercase letters. While 'company_name' is the name by which the company is colloquially known.

The second CSV file is called *Company_Tweet* and contains two columns, 'tweet_id' and 'ticker_symbol'. The 'tweet_id' is an 18-digit number assigned to the tweet when it is posted on the platform. The 'ticker_symbol' column again relates to the symbol the company is referred to on the NASDAQ.

The final CSV is named *Tweet* and contains seven columns, 'tweet_id', 'writer', 'post_date', 'body', 'comment_num' , 'retweet_num', 'like_num'. The 'tweet_id' is the same as described in the previous paragraph. The 'writer' column contains the twitter username of the writer who posted the tweet. The 'post-date' refers to the date which the tweet was posted on the platform and is stored in a UNIX-format. The 'body' column consists of the message of the tweet in UTF-8 characters. The 'comment_num' column encompasses the number of comments made after a tweet was posted. The 'retweet_num' column contains the number of times a tweet was reposted by another user of the platform. Finally, the 'like_num' column refers to the number of times other users have 'liked' a post.

As the data was taken from Kaggle and not from Twitter API itself, there are limitations to the dataset. Firstly, the method used to extract the original data was not described on the site, nor how it was initially cleaned. Secondly, the date which the data was taken from Twitter is also not known. Thus, it is unable to be confirmed at which date this data was accurate. Perhaps, some of these tweets have been commented on, retweeted or liked since.

II. Stock Market Data

In order to get historical stock data, the yfinance[2] library was installed and utilized. The finance library was queried using the company ticker symbol and the data from the same period as the Twitter data, that is from the 1 January 2015 to 31 December 2019 was extracted using the download function. This was done for all five companies being investigated. Each company had its own dataframe, containing stock trading information, which was stored in six columns: 'Open', 'High', 'Low', 'Close', 'Adj Close' and 'Volume'. The first five columns relate to price of the stock while the last column relates to trading volume.

B. TRANSFORMATION

Before the data was able to be loaded into Postgres, the data needed to be cleaned. Jupyter Notebook was used as the IDE along with Python version 3.6.10 to do this.

[2] https://pypi.org/project/yfinance/

I. Procedure

To begin with, several libraries were imported into Jupyter Notebook, including; pandas, sqlalchemy and datetime. Pandas was then used to read in the CSV files.

i. Cleaning Procedure for *Company* CSV

Once the CSV was read in as a dataframe, the index was reset to optimise later importation into Postgres. The previous 'index' column was renamed 'id'. Other than these steps, no further cleaning was necessary for this CSV.

ii. Cleaning Procedure for *Company_Tweet* CSV

As was done with the previous CSV, the index was reset and the previous 'index' column was renamed 'id'. Once again, no further cleaning was required.

iii. Cleaning Procedure for *Tweet* CSV

After being read in, the 'writer' column was removed as no value was seen in having the twitter usernames without further demographic data. No other columns were eliminated.

The datetime library, specifically the time delta function[3], was utilised to convert the UNIX-format of date/time to the more readable and easy-to-query datetime format. The converted dates were then added to the dataframe in a new column named 'datetime' and the 'post_date' column was removed. A split[4] was then performed, putting the date into its own column named 'tweet_date'. The 'datetime' column was then eliminated. This process now puts the date in a format mirroring that of the date format for the stock data which will allow for easier plotting.

iv. Cleaning Procedure for the Stock Data

For each company's dataframe, a new column was added which contained the company's ticker symbol. The index was then reset.

The dataframes were then combined into one using the concat function. The new dataframe was then sorted by ticker symbol and date and the index reset. Furthermore, the columns were renamed to better reflect their content, 'index' to 'id', 'Date' to 'stock_date', 'Open' to 'open_price', 'Close' to 'close_price', 'Volume' to 'trading_volume'. Finally, the columns were reordered logically with 'High', 'Low' and 'Adj_Close' removed.

[3] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_datetime.html
[4] https://stackoverflow.com/questions/35595710/splitting-timestamp-column-into-separate-date-and-time-columns

C. LOADING

Once clean dataframes were achieved, tables were created in Postgres through PgAdmin. For the Twitter data from the CSVs, each dataframe had a corresponding table made - being three in total. A fourth table was then made to hold the stock data taken from the yfinance library. The table structures were as followed:

I. COMPANY_INDEX

For the Company_Index table, three columns were created, 'id' set as an integer and the primary key, 'ticker_symbol' set as VARCHAR with seven characters and 'company_name' set as VARCHAR with twenty-five characters - all with no null values.

II. TWEET_DF_NEW

For the Tweet_df_new table, six columns were created, 'tweet_id' set as a big integer and the primary key, 'body' set as VARCHAR with no character limit, 'comment_num', 'retweet_num' and 'like_num' all set as integer and 'tweet_date' set as date - all with no null values.

III. TWEET_INDEX

For the Tweet_Index table, three columns were created, 'id' set as an integer and the primary key, 'tweet_id' set as a big integer and 'ticker_symbol' set as VARCHAR with no character limitation - all with no null values.

IV. STOCK_DATA

For the stock_data table, six columns were created, 'id' set as integer and the primary key, 'ticker_symbol' set as VARCHAR with no character limit, 'stock_date' set as date, 'open_price', 'close_price' and 'trading_volume' all set as integer - all with no null values.

An engine was set up to connect between Jupyter Notebook and PgAdmin. Then the to_sql function was used to load the data into the tables created in PgAdmin.