Stock Price Analysis and Prediction

Kaustuv Majumder (50360369)
Ieshaan Sharmaa (50367364)
Niraj Chetry (50376622)

December 2021

1. Introduction

Yahoo finance API is a popular data source for financial data related to stock market. This data can be used to get the live data of stock, as well as historical stock performance.
Financial data is of great importance for investors and analysts as they tell us about the health of the companies as well as the market in general.

2. Objective

The aim of this project is to extract all stock related information from Yahoo finance and store it in relational data models. This project also aims to demonstrate the end-to-end data ecosystem using a realistic use case. The project has the following components:

- Data extraction from API (Yahoo Finance API)
- Data transformation and enrichment
- Data modelling
- Data quality check
- Predictive Analytic
- Sentiment Analysis

3. Target Users

The following user groups have been identified so far:

- Retail stock investors
- Stock market analysts

4. Database Details

The data model has been designed to accommodate stock information. This consists the following:

- Historical data of stocks (Day wise trend)
- Live status from the market
- Analytical Data related to stocks

4.1. Data Source

Data from Yahoo finance has been used to build this project. yfinance is a popular python wrapper to access this API for free. Thus, this library has been used in this project to access Yahoo Finance API.

4.2. Data Model

The relational model consists of the following tables.
Note - Some of these tables are used for storing actual financial data; while other relations are used to support analytic and predictions.

COMPANY

This table contains basic information of the company

Column Name	Data Type	Constraint	Description
SYMBOL	Varchar(5)	Primary Key	The stock symbol as per the market convention
NAME	Varchar(50)	Not Null	The company name

COMPANY_DETAIL

This table contains detail information of the company

Column Name	Data Type	Constraint 1	Constraint 2	Description
SYMBOL	Varchar(5)	Unique	Foreign Key (COMPANY.SYMBOL)	The stock symbol
CITY	Varchar(20)	Not Null		The city in which the company is registered
SECTOR	Varchar(50)	Not Null		Company's business (e.g. Finance, Technology)
SUMMARY	Text	Not Null		The company description as provided by Yahoo finance

STOCK_HISTORY

This table contains the historical information of the stocks

Column Name	Data Type	Constraint 1	Constraint 2	Description
SYMBOL	Varchar(5)	Primary Key	Foreign Key (COMPANY.SYMBOL)	The stock symbol
DATE	Date			The record date
OPEN	Float8	Not Null		Valuation at market open on that day
HIGH	Float8	Not Null		Highest intraday valuation on that day
LOW	Float8	Not Null		Lowest intraday valuation on that day
CLOSE	Float8	Not Null		Valuation at market closure on that day
VOLUME	Int8	Not Null		The volume of stocks in circulation
DIVIDEND	Int8			Dividend announced (if any)
STOCK_SPLIT	Int8			Stock Split announced (if any)

STOCK_MATRIX

This table contains the live stock data as fetched from the api.

Column Name	Data Type	Constraint 1	Constraint 2	Description
SYMBOL	Varchar(5)	Unique	Foreign Key (COMPANY.SYMBOL)	The stock symbol
DAY_HIGH	Float8	Not Null		Highest intraday valuation
DAY_LOW	Float8	Not Null		Lowest intraday valuation
PRICE	Float8	Not Null		Current price
RECOMMENDATION	Varchar(5)	Not Null		Buy/Sell recommendation

MEANS

Column Name	Data Type	Constraint	Description
SYMBOL	Varchar(5)	Foreign Key (COMPANY.SYMBOL)	The stock symbol
MEAN	Text	Not Null	
TARGET_MEAN	Float8	Not Null	

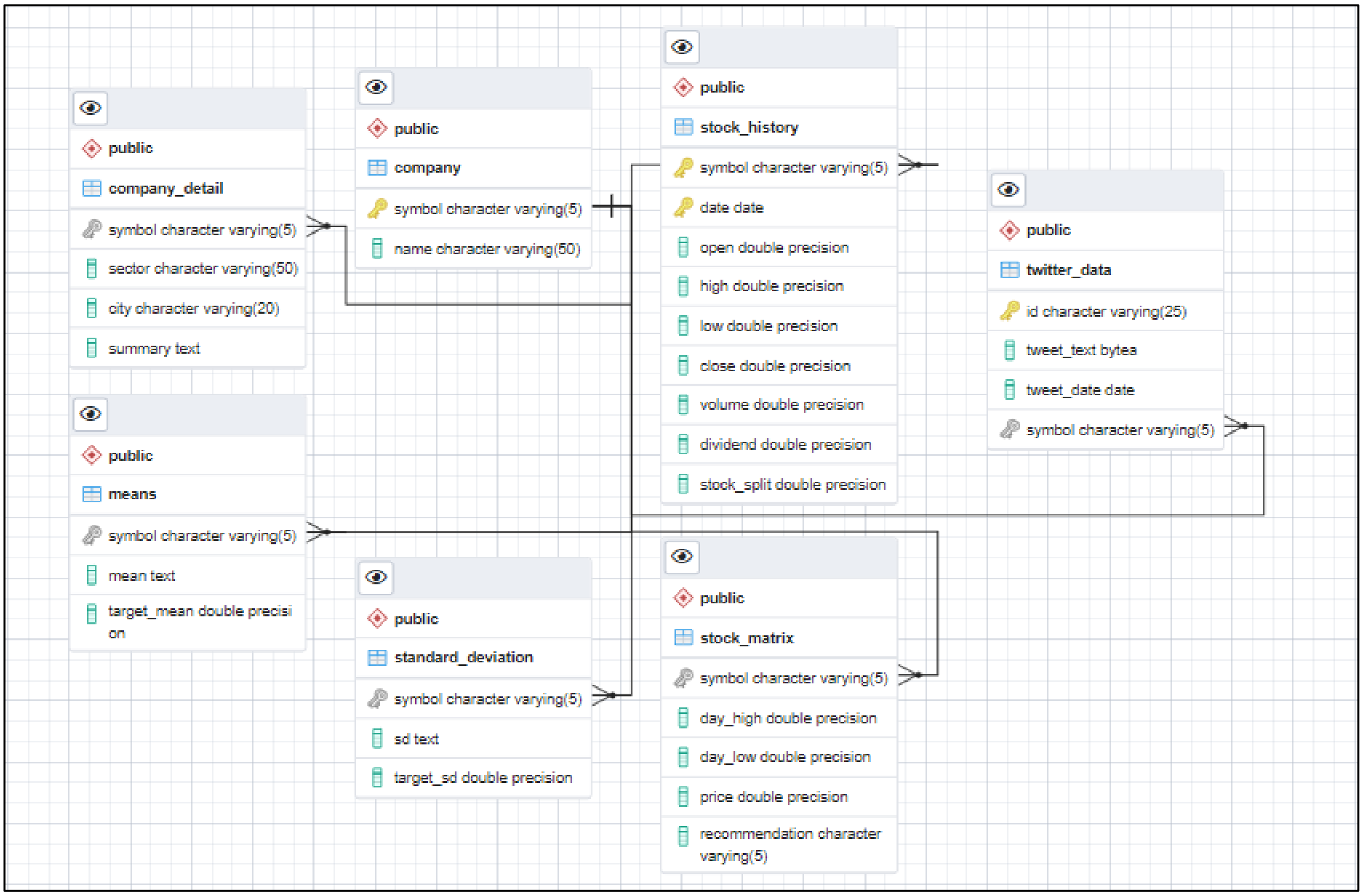
STANDARD_DEVIATION

Column Name	Data Type	Constraint	Description
SYMBOL	Varchar(5)	Foreign Key (COMPANY.SYMBOL)	The stock symbol
SD	Text	Not Null	
TARGET_SD	Float8	Not Null	

TWITTER_DATA

Column Name	Data Type	Constraint	Description
ID		Primary Key	
TWEET_TEXT		Not Null	
TWEET_DATE		Not Null	
SYMBOL			The stock symbol

4.3. Entity Relation Diagram



public
company_detail
symbol character varying(5)
sector character varying(50)
city character varying(20)
summary text
public
company
symbol character varying(5)
name character varying(50)
public
stock_history
symbol character varying(5)
date date
open double precision
high double precision
low double precision
close double precision
volume double precision
dividend double precision
stock_split double precision
public
twitter_data
id character varying(25)
tweet_text bytea
tweet_date date
symbol character varying(5)
public
means
symbol character varying(5)
mean text
target_mean double precisi on
public
standard_deviation
symbol character varying(5)
sd text
target_sd double precision
public
stock_matrix
symbol character varying(5)
day_high double precision
day_low double precision
price double precision
recommendation character varying(5)

4.4. Update Policy.

The tables are updated as per the following logic

- Company - Static Table, manually updated if required
- Company Detail - Static table, manually updated if required
- Stock History - Updated once every day
- Stock Matrix - Updated as per user request
- Mean – Static table. Updated only if the LSTM model is re-trained
- Standard Deviation - Static table. Updated only if the LSTM model is re-trained
- Twitter Data – Updated when new tweets are fetched from twitter api.

4.5. Functional Dependencies

Relation	Functional Dependencies
COMPANY	Symbol -> Name
COMPANY_DETAIL	Symbol -> Sector Symbol -> City Symbol -> Summary
STOCK_HISTORY	{Symbol, Date} -> Open {Symbol, Date} -> High {Symbol, Date} -> Low {Symbol, Date} -> Close
STOCK_MATRIX	Symbol -> Day High Symbol -> Day Low Symbol-> Price Symbol -> Recommendation
MEANS	Symbol -> Mean Symbol -> Target Mean
STANDARD_DEVIATION	Symbol -> Sd Symbol -> Target Sd
TWITTER_DATA	Id -> Tweet Text Id -> Tweet Date Id - > Symbol

5. Execution Steps

A. Execute drop.sql to drop the existing tables. This is just to make the project re-runnable.

B. Execute create.sql to create all the relations.

C. Execute load.sql to load data from csv files.

D. Execute Update.py to update the database with the latest market data (optional)

E. Execute Data_Analysis.ipnyb notebook to visualize the data analysis.

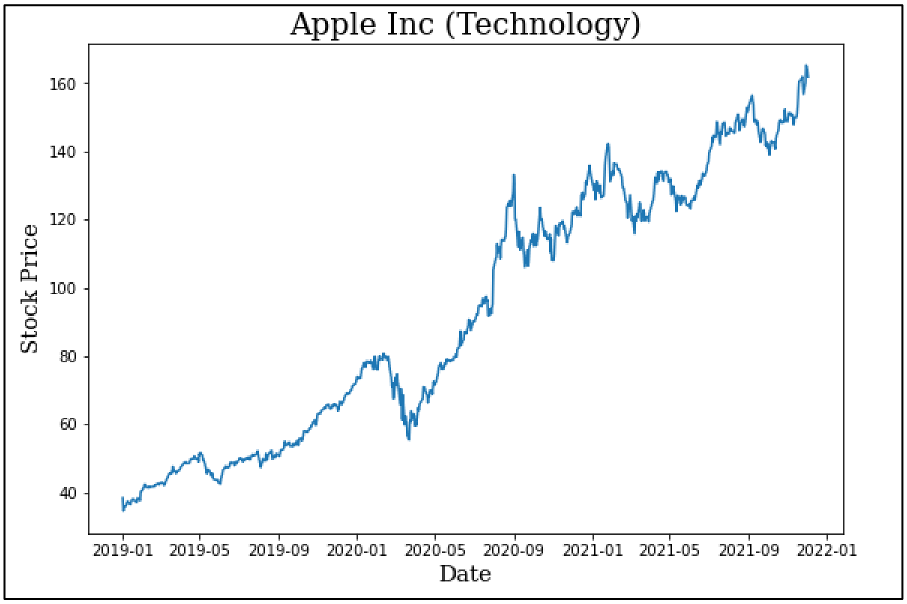
F. Execute Prediction.py for stock prediction

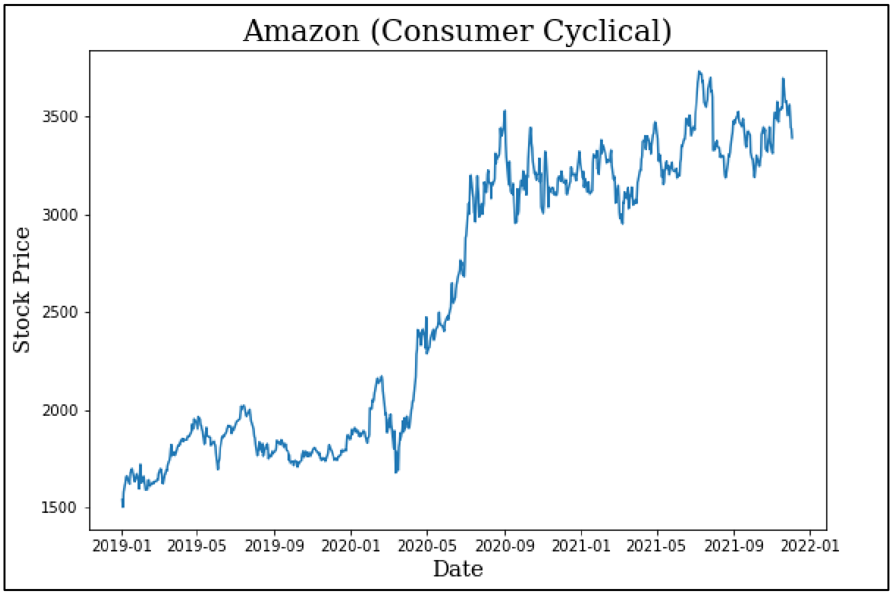
G. Execute Sentiment.py for sentiment analysis.

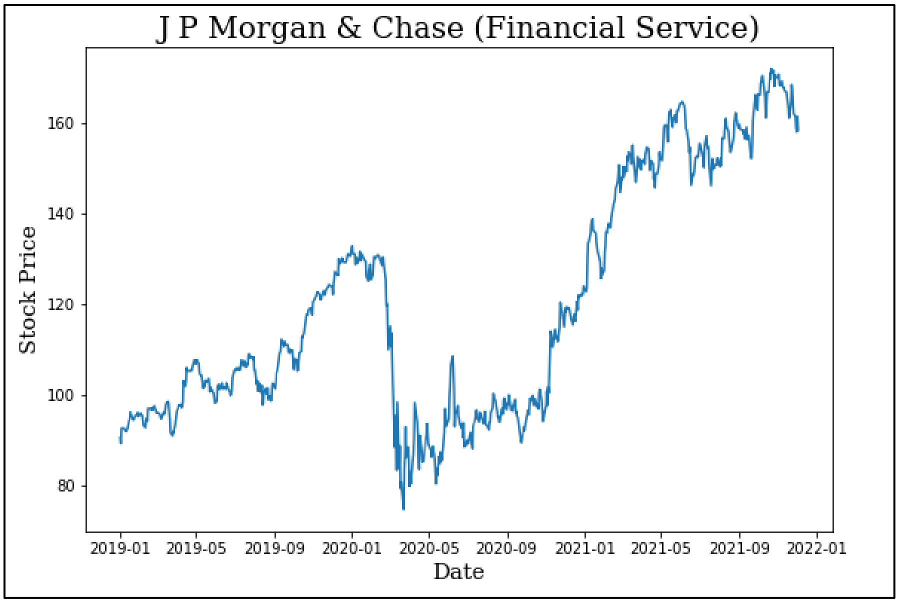
Note – User can use Stock_Insight.ipnyb notebook to get all the information aggregated at one place.

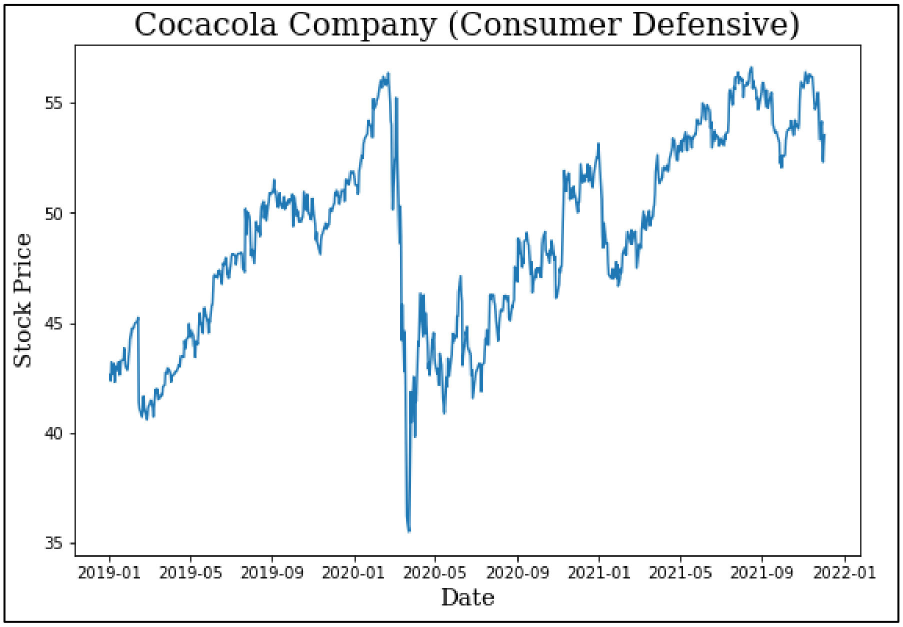
6. Data Analysis

A. Best Performing stock in each sector (Last three years)

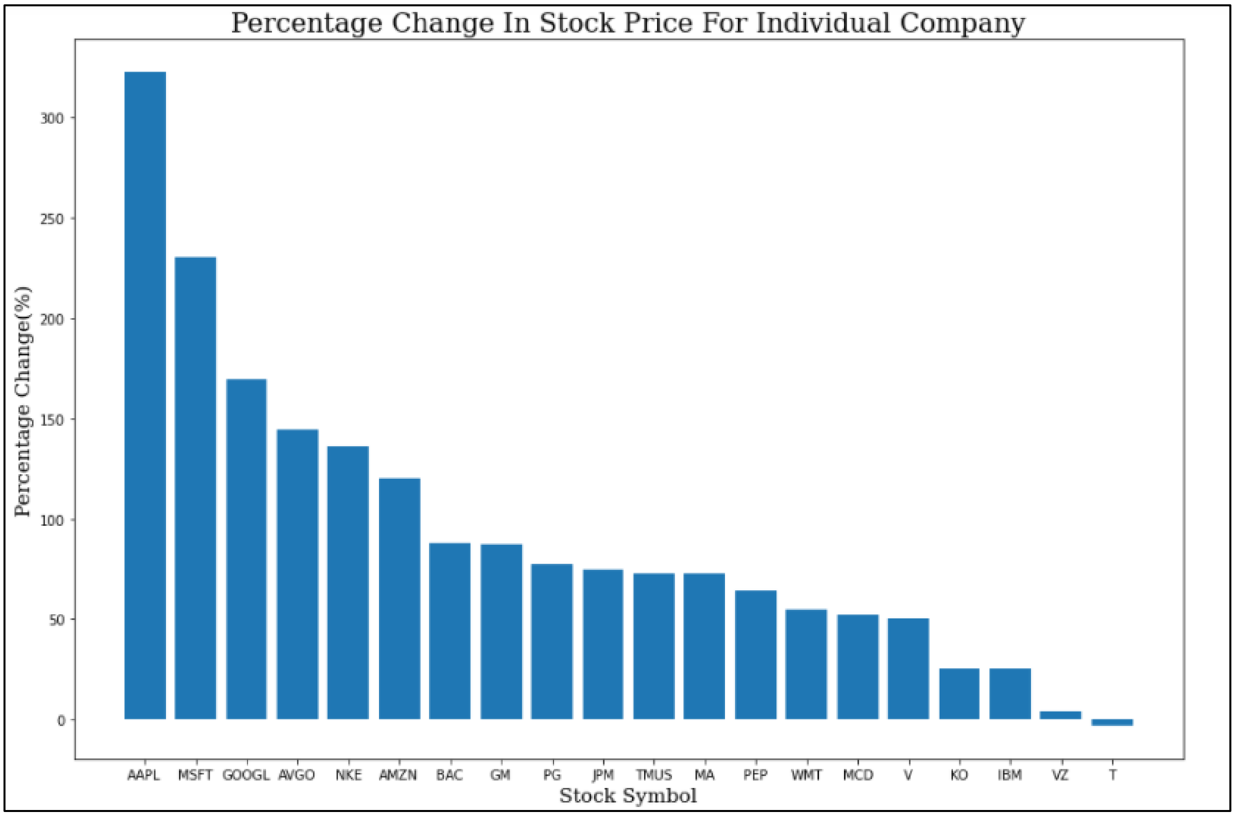



Amazon (Consumer Cyclical)

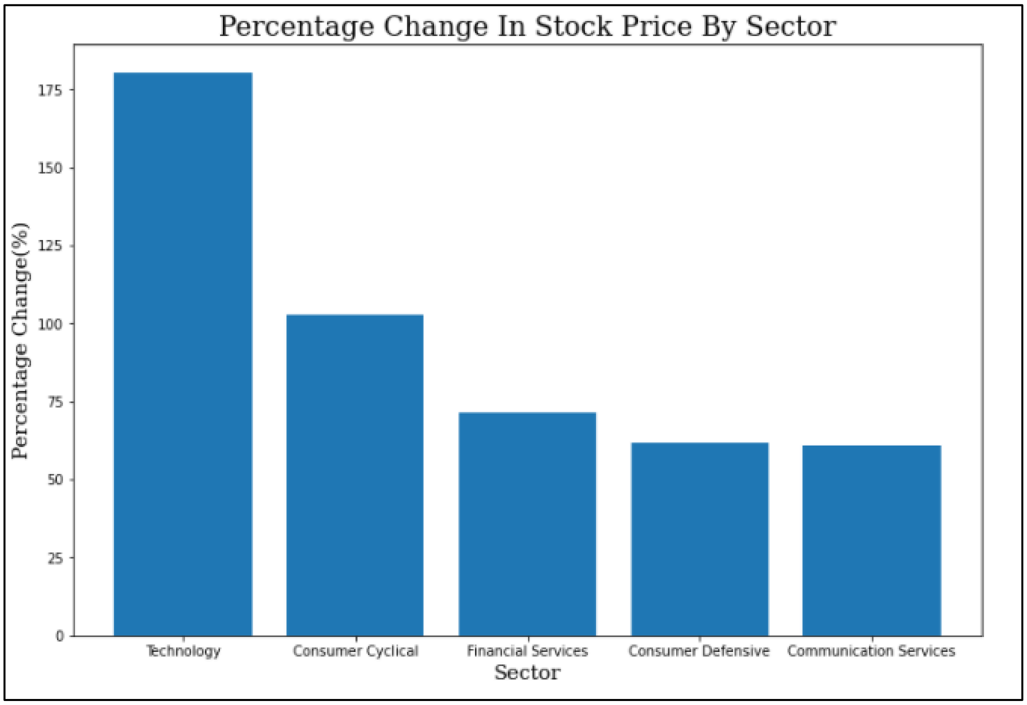

J P Morgan & Chase (Financial Service)


Cocacola Company (Consumer Defensive)

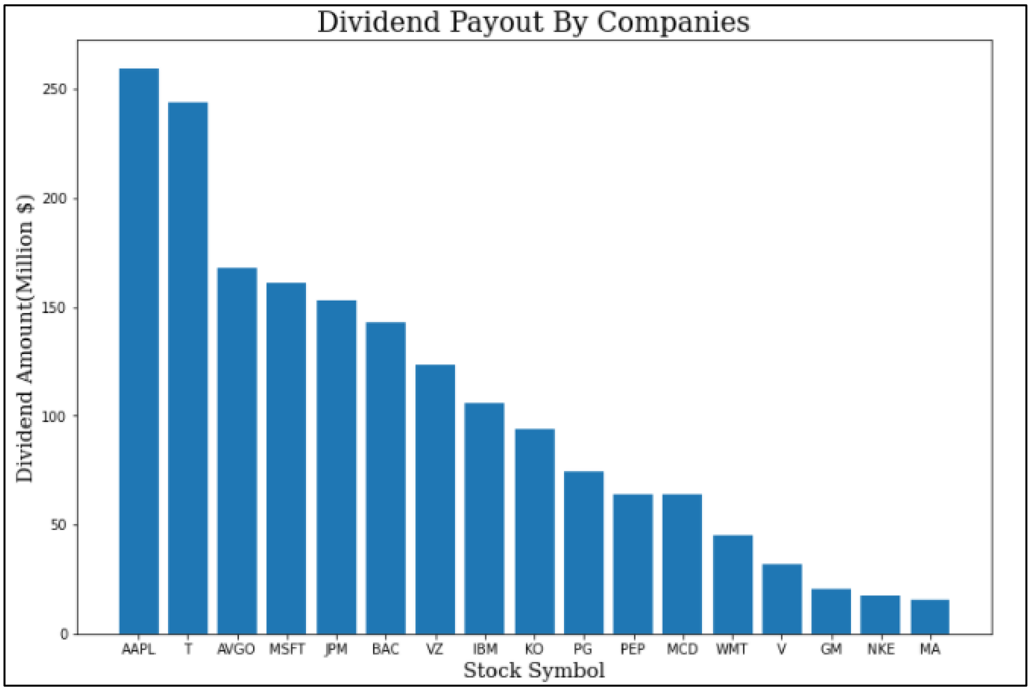
B. Percent change of stock price for each company in the last three years



C. Percent change of stock price for each sector in the last three years



D. Dividend Payout by Companies in the last three years

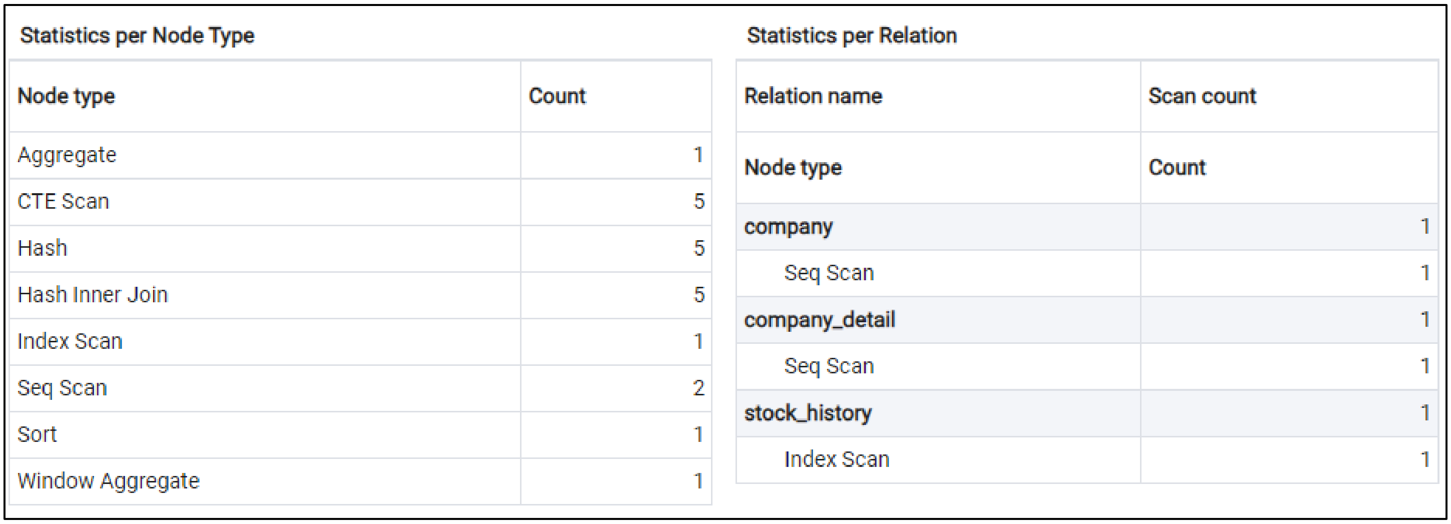


7. Query Performance Analysis

The performance analysis of some of the complex queries have been performed. Below are the results:

- **Percent change of stock price for each company**

```
    with a as (
  select ROW_NUMBER() OVER (order by symbol) as row_, symbol, date, close
    from stock_history),

  b as (
  select symbol, min(row_) as min_row, max(row_) as max_row
    from a
    where date> to_date('01012019', 'mmddyyyy')
    group by symbol),

  c as (
  select a.symbol, a.close as oldest_close
    from a, b
    where a.symbol = b.symbol and a.row_ = b.min_row),

  d as (
  select a.symbol, a.close as latest_close
    from a, b
    where a.symbol = b.symbol and a.row_ = b.max_row)

select *, ((latest_close-oldest_close)/oldest_close)*100 as pcpt_change
  from c natural join d natural join company natural join company_detail
  order by pcpt_change desc
```

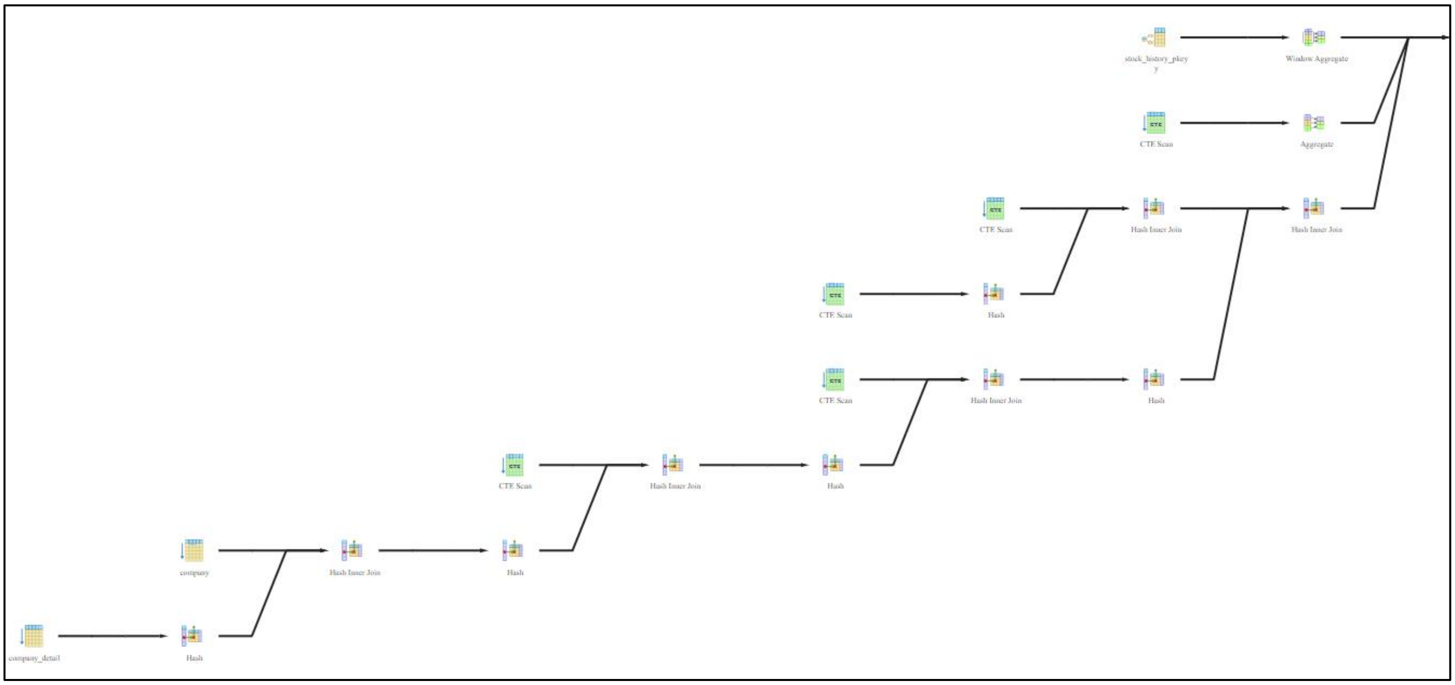


Statistics per Node Type

Node type	Count
Aggregate	1
CTE Scan	5
Hash	5
Hash Inner Join	5
Index Scan	1
Seq Scan	2
Sort	1
Window Aggregate	1

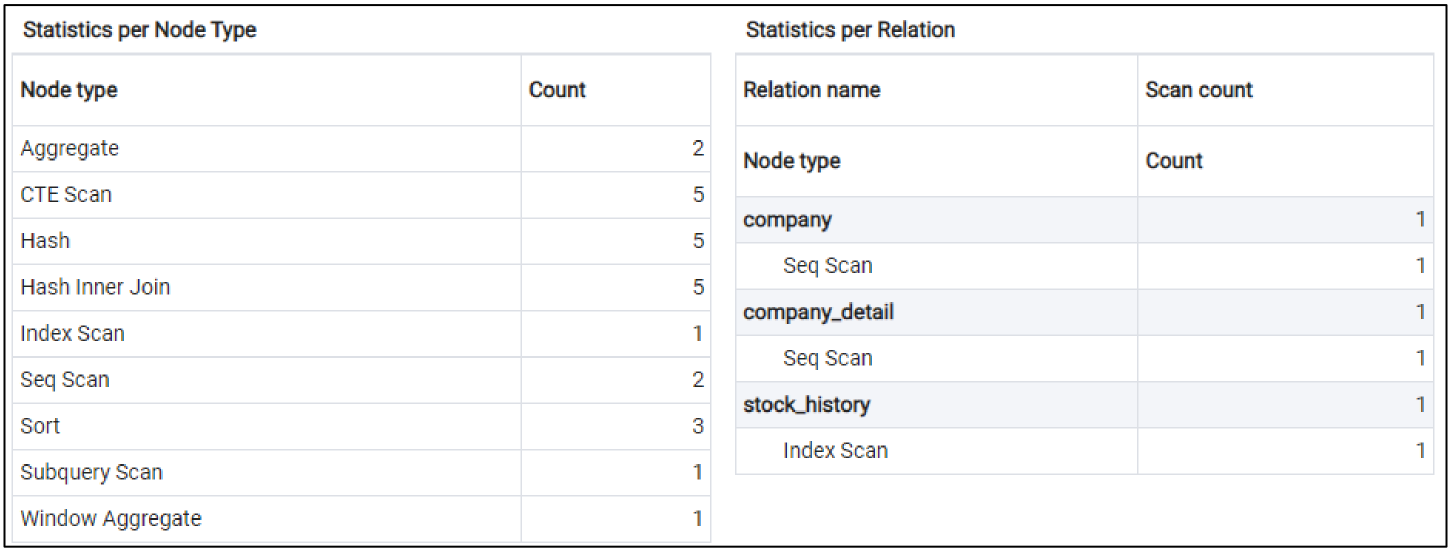
Statistics per Relation

Relation name	Scan count
Node type	**Count**
company	1
Seq Scan	1
company_detail	1
Seq Scan	1
stock_history	1
Index Scan	1



- **Percent change of stock price for each sector**

```
with a as (
        select ROW_NUMBER() OVER (order by symbol) as row_, symbol, date, close
          from stock_history),

        b as (
        select symbol, min(row_) as min_row, max(row_) as max_row
          from a
          where date> to_date('01012019', 'mmddyyyy')
          group by symbol),

        c as (
        select a.symbol, a.close as oldest_close
          from a, b
          where a.symbol = b.symbol and a.row_ = b.min_row),

        d as (
        select a.symbol, a.close as latest_close
          from a, b
          where a.symbol = b.symbol and a.row_ = b.max_row),

        e as (
        select *, ((latest_close-oldest_close)/oldest_close)*100 as pcpt_change
          from c natural join d natural join company natural join company_detail
          order by pcpt_change desc
        )

        select sector, sum(pcpt_change)/count(*) as pcpt_change_by_sector
          from e
          group by sector
          order by pcpt_change_by_sector desc
```

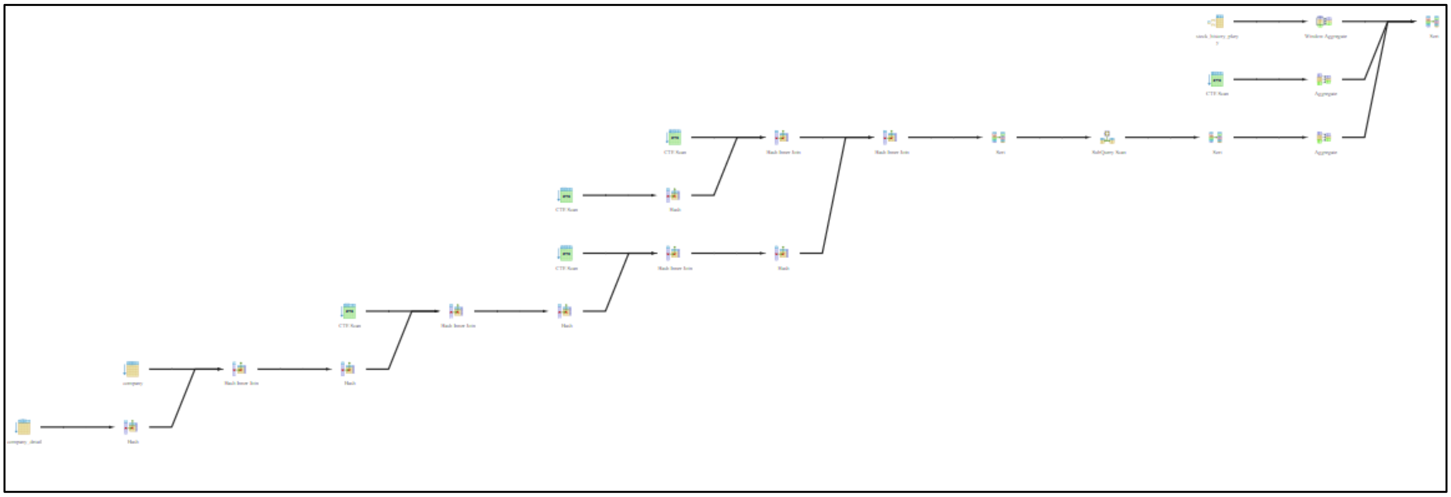


Statistics per Node Type

Node type	Count
Aggregate	2
CTE Scan	5
Hash	5
Hash Inner Join	5
Index Scan	1
Seq Scan	2
Sort	3
Subquery Scan	1
Window Aggregate	1

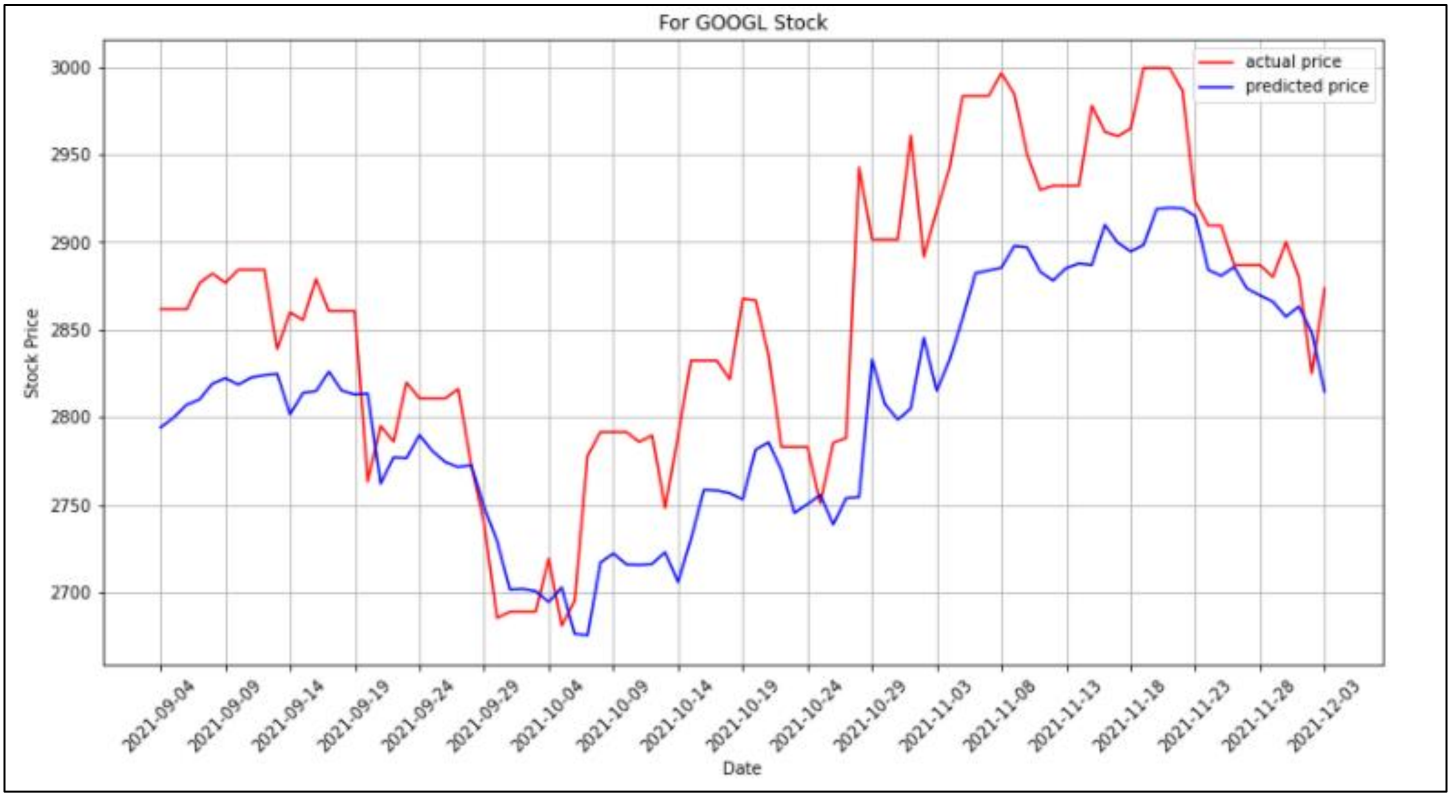
Statistics per Relation

Relation name	Scan count
Node type	**Count**
company	1
Seq Scan	1
company_detail	1
Seq Scan	1
stock_history	1
Index Scan	1



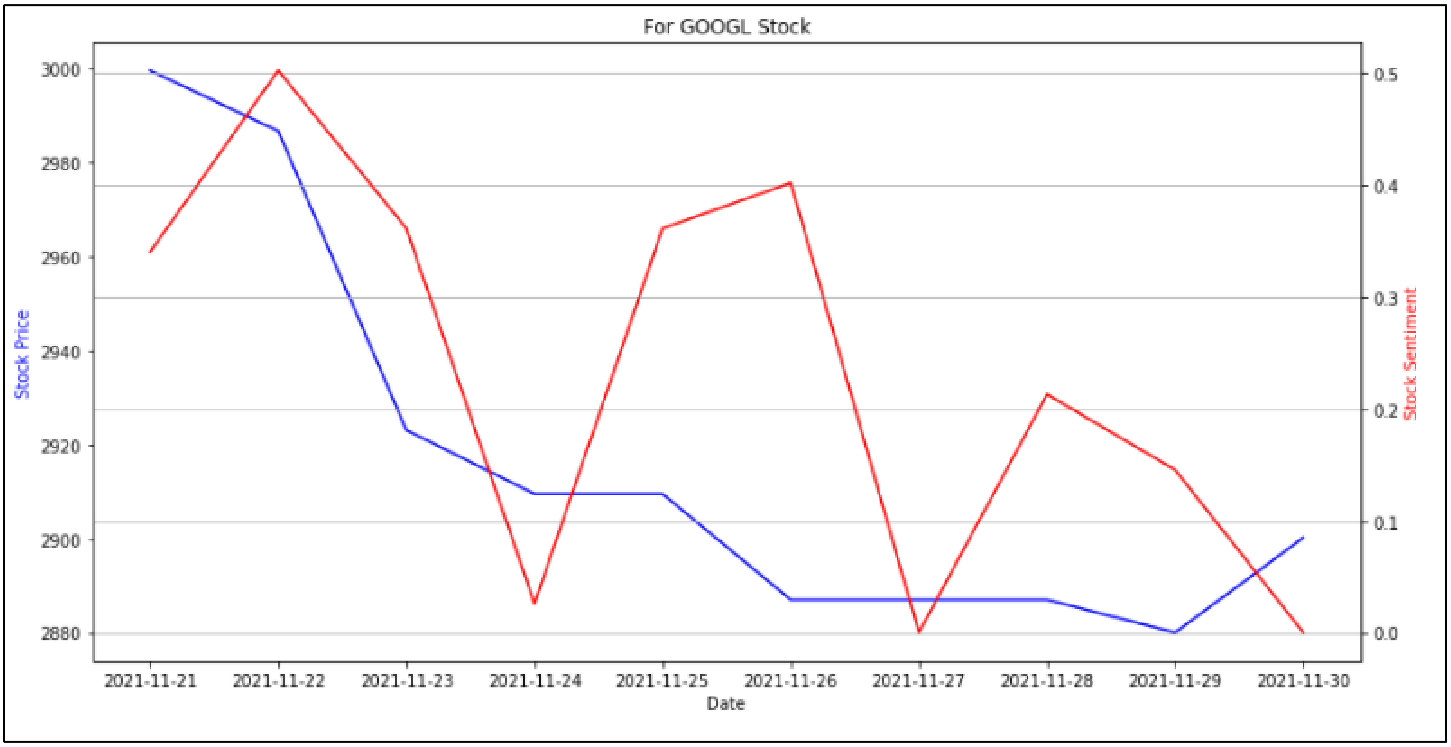
8. Predictive Modelling

The stock-market data is analyzed and modelled to design a live prediction for stock price. The historical data obtained from Yahoo Finance API is non-stationary time-series data and this information is utilized to train a Deep Learning model to predict the future stock price. *Long-Short Term Memory* network or shortly *LSTM*, which is an enhanced version of *Recurrent Neural Network* (*RNN*) is applied for this purpose. *LSTM* models are very appropriate for complex time-series data, as it can store and memorize very old information. The historical data is divided into Train and Test followed by applying the LSTM model on the Train data to determine the performance on unseen Test data. Attached below is a snap for the actual stock price vs predicted stock price for GOOGLE obtained from the model.



9. Sentiment Analysis

Tweets related to stocks are fetched for the last 10 days of November and the corresponding sentiment values are calculated to get the attitude of investors towards the stock. The sentiment value is measured by a whole number ranging between -1 to +1 which is called polarity. The more polarity we have, the more positive/better is the sentiment and vice-versa. The stock price and the market sentiment are having similar kinds of trends as presumed.



10.Conclusion

The user will thus have input from three sources to make decision regarding any stock:

- Recommendation from Yahoo finance
- Predicted future price of the stock
- Current Sentiment about the company

These three data points should be sufficient for an investor to arrive to a decision whether to buy, sell, or hold a stock.