INTERNATIONAL TECHNOLOGICAL UNIVERSITY

Name: Shahriyar Mammadov

Student ID: 95427

Graduate Program: Software Engineering

Python Course

Professor: Zhupei Shi

1 Introduction

Stock trading is one of profitable activities, in case can predict rightly most of the time. Different traders use various methods to predict it, including news to price data of companies or financial statements such as income statement, balance sheet and cash flow. Unfortunately, there is no data for investors' sentiments, attitudes, and expectations that ultimately affect stock prices. Even so, the prediction is not correct sometimes, it should provide good enough insight for trader to make right decision. In this project we will try based on financial data of companies make predictions for short and long term.

2. Dataset and Features

In order to download the raw data of individual stock from yahoo, in our case for Google, Yahoofinancials library for Python3 is used. This library is a powerful financial data module used for pulling both fundamental and technical data from Yahoo Finance. The advantage of this library is that, all methods returns data as a JSON format which is useful to share data with other systems over REST API interface or save locally in CSV files. To make more precise prediction for stock price of company in short and long term, we need to find most important metrics in Stock Fundamental Analysis. Below are key financial statements for the past years or quarters:

Income Statement – Annual/Quarterly. Show how profitable company.

Balance Sheet – Annual/Quarterly. Shows if enough cash or too much debt.

Cash Flow – Annual/Quarterly. Shows financial state, more valuable than income statement.
Stock Price Exchange – Daily. Data for past 10 years.

Below are key features retrieved from upper mentioned datasets impacting stock price which print output to console:

Stock Exchange Data Source: NasdaqGS
Currency: USD

---

1. Company Valuation Measures:

a) Market CAP(company's shares outstanding * current market price of one share)
b) PE(Price to Earnings Ratio)
c) Price to Sales Ratio (compares a company's stock price to its revenue)

2. Stock daily trading information:

a) Current Price
b) Current Change
c) Current percent change
d) Current volume
e) Daily Low
f) Daily High
g) Yearly high
h) Yearly Low
i) 50 day moving avg
j) 200day moving avg
k) Previous day close price
l) Open price
m) 10days Average daily volume
n) 30 days Average daily volume
o) Beta(3y monthly), stock volatility(>1, risky but profitable)

3. Financial Highlights based on Income statement/Cash Flow/Balance Sheet:

a) Interest Expense
b) Total Operating Expense
c) Income Tax Expense
d) Research and Development Cost
e) Operating Income
f) Total Revenue
g) Cost of Revenue

h) Income Before Tax
i) Gross Profit
j) Net Income
k) Net Income from Continuing Ops
l) Book Value(Net Asset=TotalAsset-IntangibleAsset(Patent,Goodwill))
m) EBITDA(Earnings before interest,tax depreciation and amartization)
n) Earnings per share

4. Dividents and splits. Currently, these values are missing in YahooFinancials dashboard as well:

a) Divindent Yield
b) Annual average Divident yield
c) 5y average divident yield
d) Dividend Rate
e) Annual average dividend rate
f) Payout Ratio
g) Ex Evidend Date

3. Project Architecture

Below are key technologies used and their brief description:

a) Kubernetes: To deploy application and Graylog2 server.

b) Istio: Observability, Security and Networking of application.

c) Github: To store Source Code of our application.

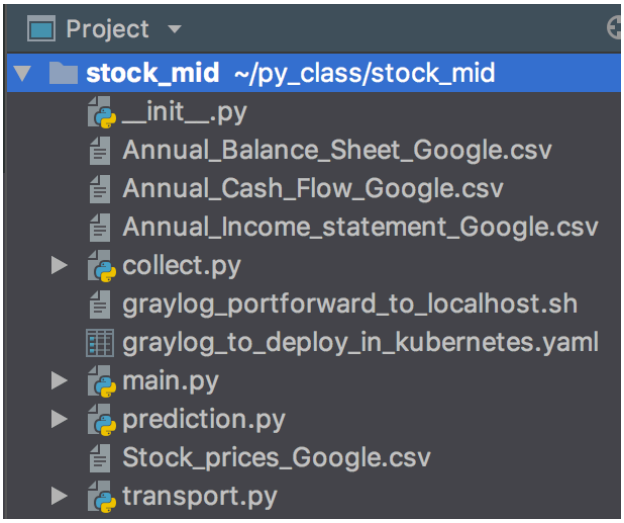
d) TravisCI: To run test_suite during development

e) DockerHub: To store artifactory, docker image of our application.

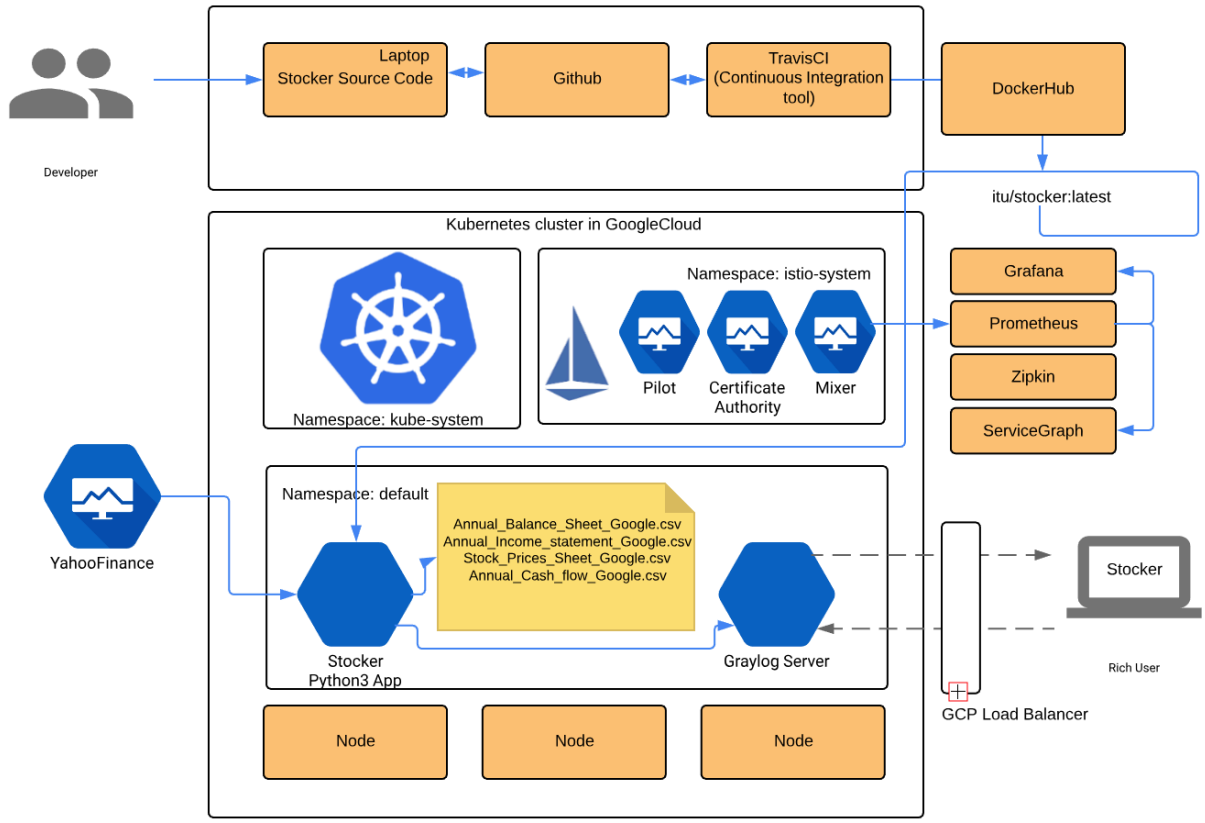
f) Graylog2: For monitoring,metrics, alert and notification of our application.

g) Python3.7

Project Source Code Structure:



Project Low Level Design(LLD) Chart:

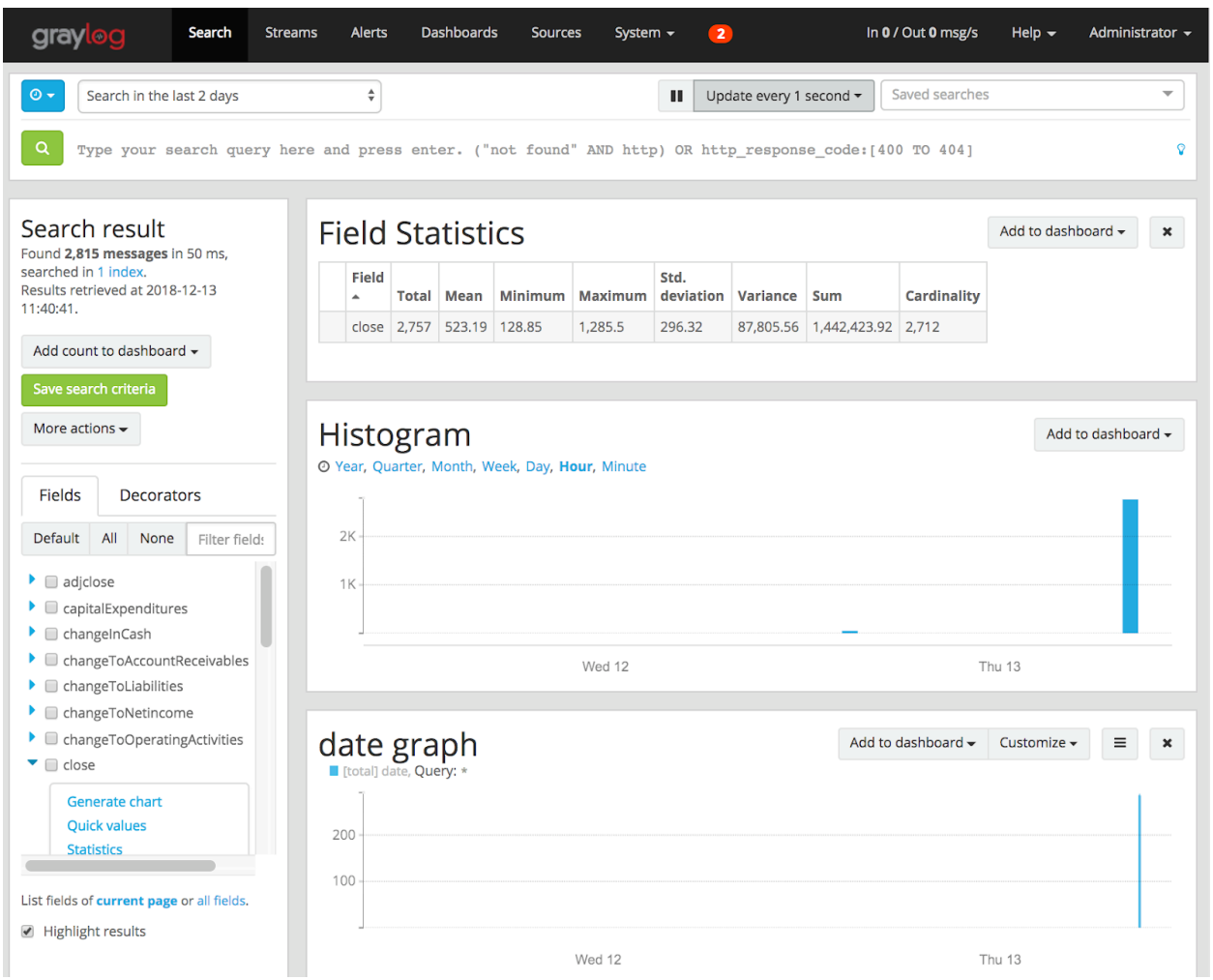


4. Realtime Monitoring

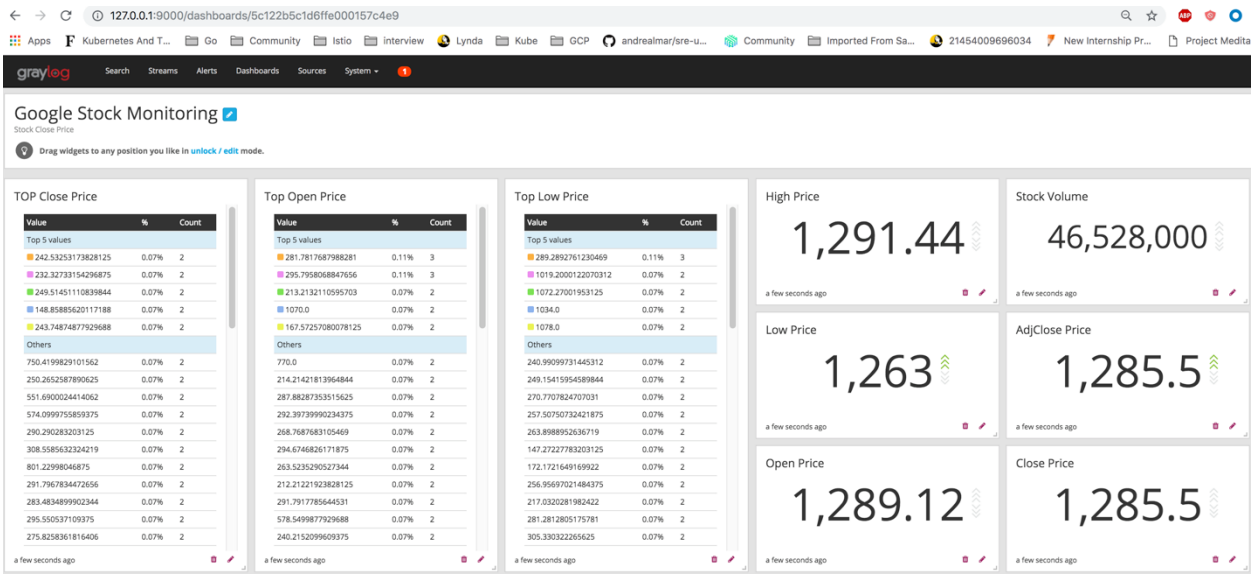
In this project industry leading open source log management platform Graylog is used as Stock monitoring. The main motivations are:

1. Analyze data without complete plan, prior to search.
2. Scalability in Google Cloud environment, enabling up to several terabytes of workload handling per day.
3. Search, aggregate, analyze, visualize and report high speed using Elasticsearch for storage.

a) Metrics:



b) Dashboard:



Realtime Statistics for Stock Price, Income Statement, Cash Flow and Balance Sheet:

Field Statistics

Field	Total	Mean	Minimum	Maximum	Std. deviation	Variance	Sum	Cardinality
capitalExpenditures	8	-11,090,000,000	-13,184,000,000	-9,950,000,000	1,270,934,302	1,615,274,000,000,008,192	-88,720,000,000	4
changeInCash	8	-2,045,750,000	-3,631,000,000	-551,000,000	1,099,250,056.86	1,208,350,687,500,000,256	-16,366,000,000	4
changeToAccountReceivables	8	-2,520,250,000	-3,768,000,000	-1,641,000,000	792,935,172.32	628,746,187,499,999,232	-20,162,000,000	4
changeToLiabilities	8	490,250,000	246,000,000	1,121,000,000	365,645,576.34	133,696,687,500,000,000	3,922,000,000	4
changeToNetincome	8	6,166,500,000	3,615,000,000	8,284,000,000	1,752,700,559.14	3,071,959,249,999,994,880	49,332,000,000	4
changeToOperatingActivities	8	2,295,250,000	1,461,000,000	3,682,000,000	879,385,971.86	773,319,687,500,000,256	18,362,000,000	4
depreciation	8	5,656,000,000	4,601,000,000	6,899,000,000	902,030,210.14	813,658,499,999,997,952	45,248,000,000	4
effectOfExchangeRate	8	-158,000,000	-434,000,000	405,000,000	342,386,477.54	117,228,500,000,000,000	-1,264,000,000	4
investments	8	-14,383,500,000	-19,448,000,000	-6,222,000,000	5,186,698,492.3	26,901,841,249,999,945,728	-115,068,000,000	4
netBorrowings	8	-365,500,000	-1,335,000,000	-18,000,000	560,382,235.62	314,028,250,000,000,000	-2,924,000,000	4
netIncome	8	15,656,000,000	12,662,000,000	19,478,000,000	2,567,147,444.15	6,590,246,000,000,008,192	125,248,000,000	4
otherCashflowsFromFinancingActivities	8	-2,790,250,000	-3,366,000,000	-2,069,000,000	559,293,471.71	312,809,187,499,999,232	-22,322,000,000	4
otherCashflowsFromInvestingActivities	8	36,000,000	-1,978,000,000	1,419,000,000	1,257,067,022.88	1,580,217,500,000,000,000	288,000,000	4
repurchaseOfStock	8	-3,024,750,000	-4,846,000,000	-1,780,000,000	1,309,801,010.65	1,715,578,687,499,999,232	-24,198,000,000	3
totalCashflowsFromInvestingActivities	8	-26,833,000,000	-31,401,000,000	-21,055,000,000	4,548,764,007.95	20,691,253,999,999,909,888	-214,664,000,000	4
totalCashFromFinancingActivities	8	-5,735,500,000	-8,332,000,000	-2,087,000,000	2,688,000,232.51	7,225,345,249,999,994,880	-45,884,000,000	4
totalCashFromOperatingActivities	8	30,680,750,000	23,024,000,000	37,091,000,000	6,026,559,025.47	36,319,413,687,499,751,424	245,446,000,000	4

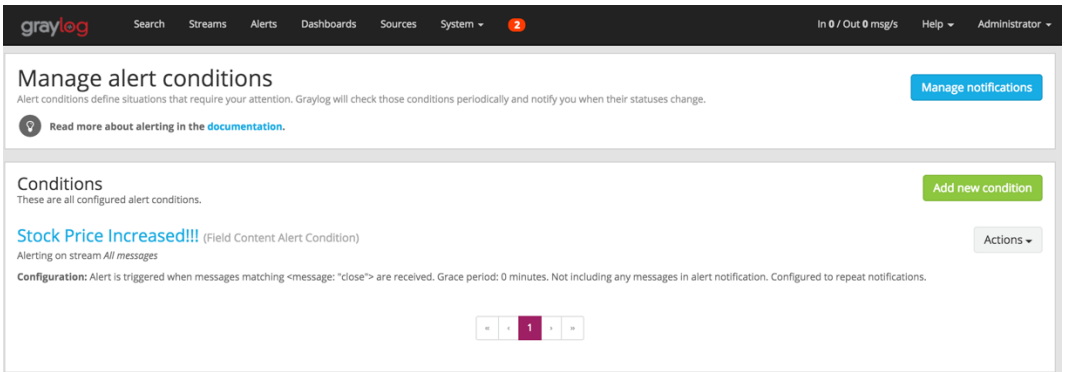
Field Statistics

Add to dashboard

Field	Total	Mean	Minimum	Maximum	Std. deviation	Variance	Sum	Cardinality
adjclose	2,757	523.19	128.85	1,285.5	296.32	87,805.56	1,442,423.92	2,712
close	2,757	523.19	128.85	1,285.5	296.32	87,805.56	1,442,423.92	2,712
high	2,757	527.92	134.82	1,291.44	298.63	89,179.63	1,455,481.52	2,670
low	2,757	518.3	123.77	1,263	293.84	86,341.8	1,428,958.17	2,696
open	2,757	523.36	131.39	1,289.12	296.36	87,829.63	1,442,903.76	2,685
volume	2,757	4,740,572	520,600	46,528,000	4,212,029.97	17,741,196,481,921.76	13,069,757,000	2,701

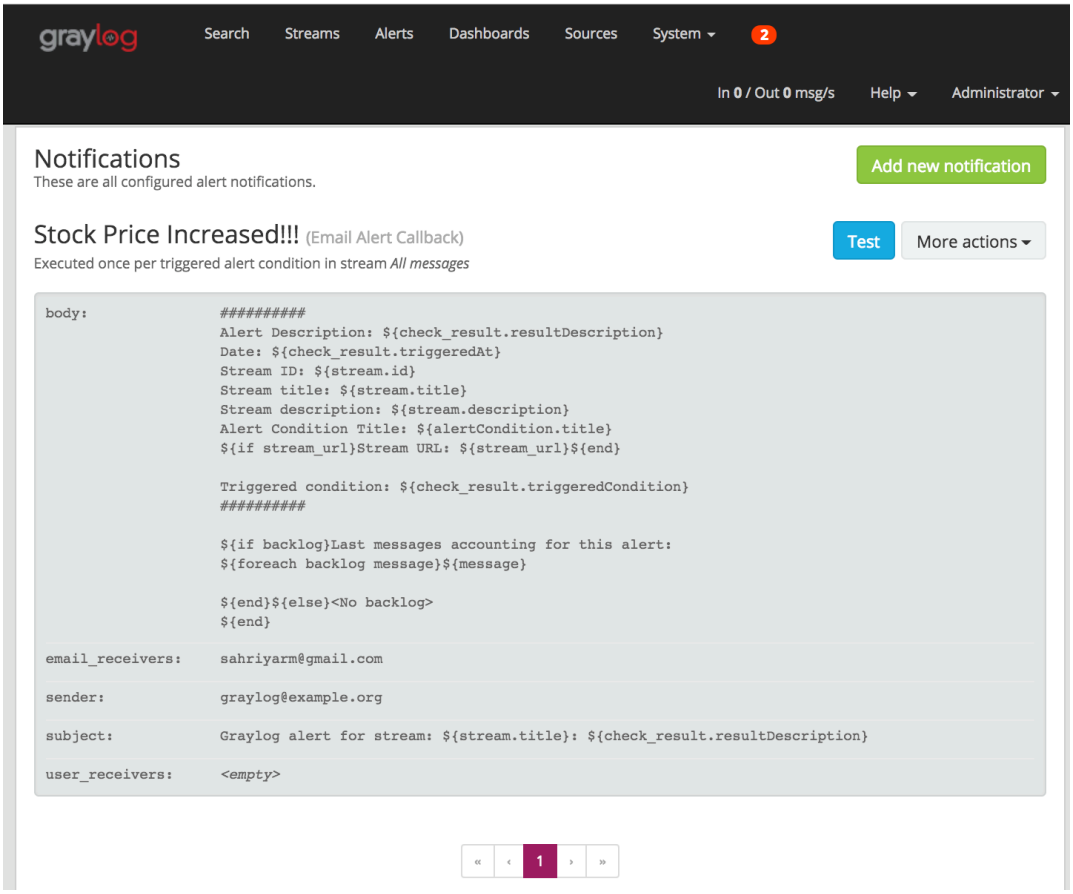
c) Alerts:

Create new alarms based on conditions that matters:



d) Notifications:

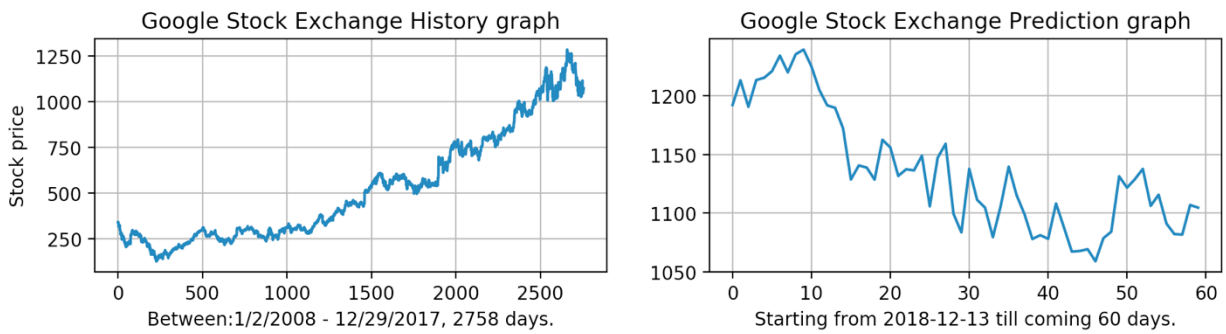
Get notification via email based on set alarms:



5. Prediction

In this step, we will do prediction based on past 10 years daily data collected for Adjclose feature and predict upcoming 60 days, assuming it is long term investment. Meantime, we generated some key features of Company which can help to make right decision. In Prediction phase steps are: Based on saved CSV file, Stock Adjacent Close Price for past 10 years Train Model, Test to measure confidence level and Predict for upcoming 2 month daily.

Console Graph Output:



Console Table Output:

Statistic Analysis:
==========
Stock Exchange Data Source: NasdaqGS
Currency: USD

Valuation Measures
---
main.market CAP: 742544048128
PE(Price to Earnings Ratio): 40.287464
Price to Sales: 5.717771

Stock Measures

Current Price: 1073.54
Current Change: -0.1899414
Current percent change -0.00017689867
Current volume: 1246498
Daily Low: 1064.994

Daily High: 1088.07
Yearly high: 1291.44
Yearly Low: 984
50 day moving avg: 1069.5326
200day moving avg: 1153.7058
Previous day close price: 1073.73
Open price: 1075.67
10days Average daily volume: 1607487
30 days Average daily volume: 2064629
Beta(3y monthly), stock volatility(>1, risky but profitable): 1.330345

Financial Highlights: Income/Cash/Balance Sheet

Interest Expense: -109000000
Operating Income: 28882000000
Total Operating Expense: 81973000000
Total Revenue: 110855000000
Cost of Revenue: 45583000000
Income Before Tax: 27193000000
Income Tax Expense: 14531000000
Gross Profit 65272000000
Net Income: 12662000000
Net Income from Continuing Ops: 12662000000
Research and Development Cost: 16625000000
Book Value(Net Asset=TotalAsset-IntangibleAsset(Patent,Goodwill)): 169840000000
EBITDA(Earnings before interest,tax depreciation and amartization): 28882000000
Earnings per share: 26.646998679291404

Dividents and splits, Currently, missing in YahooFinancials as well
---
Divindent Yield None
Annual average Divident yield: None
5y average divident yield: None
Dividend Rate: None
Annual average dividend rate None
Payout Ratio: 0
Ex Evidend Date: -

Operation completed in: 0.03518414497375488 seconds.

confidence: 96 %
Forecast for coming 60 days:

[1192.18867397 1213.24643734 1190.61279088 1213.41802436 1215.37487671
 1221.00951923 1234.27039938 1220.03326493 1235.41203456 1239.38231534
 1224.9236855 1204.93698029 1191.84397379 1189.73832726 1172.33287172
 1128.66790824 1140.57801907 1138.80492954 1128.54843852 1162.48487194
 1155.89293886 1131.63078396 1137.34122512 1136.41134063 1148.87765111

1105.80464819 1147.16911438 1159.06566674 1099.49466619 1083.52153004
1137.67128692 1111.27866315 1104.64382294 1079.39233095 1106.52027687
1139.57020619 1115.31608767 1099.0491367 1077.8844649 1081.12480761
1078.02885659 1108.11807042 1087.81858325 1067.14854688 1067.63351168
1069.21279141 1058.94016578 1078.53574806 1084.09986338 1131.2968647
1121.66924644 1128.94163631 1137.63634299 1106.21565892 1115.6448688
1091.03209359 1082.0418146 1081.64192113 1106.90283798 1104.62481983]

6. Conclusion

In this project I summarized most of our learnings from this course. It was nice experience, to learn about Python libraries and functions used, at the same time was great to know more about insights of financial data, which is key indicator for successful prediction. Hopefully, by keeping advancing project, could be rich someday ☺

7. References

https://finance.yahoo.com/quote/GOOGL/key-statistics?p=GOOGL

https://github.com/JECSand/yahoofinancials

Appendix A Python3.7 Application Source Code

Main.py

```
#!/usr/bin/python
from __future__ import print_function
import sys
import time
from yahoofinancials import YahooFinancials as YF
import collect
import prediction

DEFAULT_ARGS = ('GOOGL')
MODULE_ARGS = ('yf', 'yahoofinancial', 'yahoofinancials')
HELP_ARGS = ('-h', '--help')
global mark
global company
mark= '-' * 64
if len(sys.argv) > 1:
     company=sys.argv[1]
else:
    company='Google'

def timeit(f, *args):
    print(mark)
    st = time.time()
    f(*args)
    et = time.time()
    print(mark)
    print('Operation completed in: ',et - st, 'seconds.')

if __name__ == '__main__':
    api = set(s for s in dir(YF) if s.startswith('get_'))
    api.update(MODULE_ARGS)
    api.update(HELP_ARGS)
    ts = sys.argv[1:]
    queries = [q for q in ts if q in api]
    ts = [t for t in ts if not t in queries] or DEFAULT_ARGS
    if [h for h in HELP_ARGS if h in queries]:
        helpapi(queries)
    elif queries:
        customapi(queries, ts)
    else:
        a=collect.Collect(ts[0] if 1 == len(ts) else ts)
        timeit(a.data_collect)
        timeit(a.print_console)
        timeit(a.save_csv)
        b=prediction.Prediction()
        b.training_data()
        b.testing_data()
        b.predict_price()
```

Collect.py

```
import csv
import sys
import transport as tp
from yahoofinancials import YahooFinancials as YF
import main


class Collect:
    def __init__(self,ticker):
        self.tick = YF(ticker)
        self.d_stock = self.tick.get_historical_price_data('2008-01-01', '2019-01-01',
'daily')
        self.a_income = self.tick.get_financial_stmts('annual', 'income')
        self.a_balance = self.tick.get_financial_stmts('annual', 'balance')
        self.a_cash = self.tick.get_financial_stmts('annual', 'cash')

    def data_collect(self):
        try:
            r = self.tick._cache.keys()
        except AttributeError:
            pass
        else:
            print(main.mark)
            print(r)

    def print_console(self):
        print(main.mark)
        print('Statistic Analysis:')
        print('==============================================')
        print('Stock Exchange Data Source: ', self.tick.get_stock_exchange())
        print('Currency: ', self.tick.get_currency())
        print(main.mark)
        print('Valuation Measures')
        print(main.mark)
        print('main.market CAP: ',self.tick.get_market_cap())
        print('PE(Price to Earnings Ratio): ', self.tick.get_pe_ratio())
        print('Price to Sales: ', self.tick.get_price_to_sales())
        print(main.mark)
        print('Stock Measures')
        print(main.mark)
        print('Current Price: ',self.tick.get_current_price())
        print('Current Change: ', self.tick.get_current_change())
        print('Current percent change',self.tick.get_current_percent_change())
        print('Current volume: ',self.tick.get_current_volume())
        print('Daily Low: ', self.tick.get_daily_low())
        print('Daily High:', self.tick.get_daily_high())
        print('Yearly high: ', self.tick.get_yearly_high())
        print('Yearly Low: ', self.tick.get_yearly_low())
        print('50 day moving avg: ',self.tick.get_50day_moving_avg())
        print('200day moving avg: ',self.tick.get_200day_moving_avg())
        print('Previous day close price: ',self.tick.get_prev_close_price())
        print('Open price: ',self.tick.get_open_price())
        print('10days Average daily volume:
',self.tick.get_ten_day_avg_daily_volume())
        print('30 days Average daily volume:
',self.tick.get_three_month_avg_daily_volume())
        print('Beta(3y monthly), stock volatility(>1, risky but profitable):
',self.tick.get_beta())
        print(main.mark)
```

```
        print('Financial Highlights: Income/Cash/Balance Sheet')
        print(main.mark)
        print('Interest Expense: ',self.tick.get_interest_expense())
        print('Operating Income: ', self.tick.get_operating_income())
        print('Total Operating Expense: ',self.tick.get_total_operating_expense())
        print('Total Revenue: ',self.tick.get_total_revenue())
        print('Cost of Revenue: ',self.tick.get_cost_of_revenue())
        print('Income Before Tax: ',self.tick.get_income_before_tax())
        print('Income Tax Expense: ',self.tick.get_income_tax_expense())
        print('Gross Profit', self.tick.get_gross_profit())
        print('Net Income: ', self.tick.get_net_income())
        print('Net Income from Continuing Ops:
',self.tick.get_net_income_from_continuing_ops())
        print('Research and Development Cost:
',self.tick.get_research_and_development())
        print('Book Value(Net Asset=TotalAsset-IntangibleAsset(Patent,Goodwill)):
',self.tick.get_book_value())
        print('EBITDA(Earnings before interest,tax depreciation and amartization):
',self.tick.get_ebit())
        print('Earnings per share: ',self.tick.get_earnings_per_share())
        print(main.mark)
        print('Dividents and splits, Currently, missing in YahooFinancials as well')
        print(main.mark)
        print('Divindent Yield',self.tick.get_dividend_yield())
        print('Annual average Divident yield: ',self.tick.get_annual_avg_div_yield())
        print('5y average divident yield: ',self.tick.get_five_yr_avg_div_yield())
        print('Dividend Rate: ',self.tick.get_dividend_rate())
        print('Annual average dividend rate',self.tick.get_annual_avg_div_rate())
        print('Payout Ratio: ',self.tick.get_payout_ratio())
        print('Ex Evidend Date: ',self.tick.get_exdividend_date())

    def save_csv(self):

        # Save Stock Prices in CSV
        with open('Stock_prices_'+ main.company +'.csv', mode='w') as result_w:
            result = csv.writer(result_w, delimiter=',', quotechar='"',
quoting=csv.QUOTE_MINIMAL)
            result.writerow(['Date', 'high', 'low', 'adjclose', 'close', 'open',
'volume'])
            for k,v in self.d_stock['GOOGL'].items():
                if k=='prices':
                    for i in v:
                      #    transport(v)

result.writerow([i['formatted_date'],i['high'],i['low'],i['adjclose'],i['close'],i['op
en'],i['volume']])

        # Save Annual Income Statement in CSV
        with open('Annual_Income_statement_' + main.company + '.csv', mode='w') as
result_w:
            result = csv.writer(result_w, delimiter=',', quotechar='"',
quoting=csv.QUOTE_MINIMAL)
            result.writerow(['Year', 'researchDevelopment', 'incomeBeforeTax',
'minorityInterest', 'netIncome',
                             'effectOfAccountingCharges',
'sellingGeneralAdministrative', 'grossProfit', 'ebit',
                             'operatingIncome', 'otherOperatingExpenses',
'interestExpense', 'extraordinaryItems',
                             'nonRecurring', 'otherItems', 'incomeTaxExpense',
'totalRevenue', 'totalOperatingExpenses',
                             'costOfRevenue', 'totalOtherIncomeExpenseNet',
```

```
'discontinuedOperations',
                                'netIncomeFromContinuingOps',
'netIncomeApplicableToCommonShares'])
            for i in self.a_income['incomeStatementHistory']['GOOGL']:
                for k, v in i.items():
                    #  tp.transport(v)
                    result.writerow(
                                    [k, v['researchDevelopment'], v['incomeBeforeTax'],
v['minorityInterest'], v['netIncome'],
                                    v['effectOfAccountingCharges'],
v['sellingGeneralAdministrative'], v['grossProfit'],
                                    v['ebit'], v['operatingIncome'],
v['otherOperatingExpenses'], v['interestExpense'],
                                    v['extraordinaryItems'], v['nonRecurring'],
v['otherItems'], v['incomeTaxExpense'],
                                    v['totalRevenue'], v['totalOperatingExpenses'],
v['costOfRevenue'],
                                    v['totalOtherIncomeExpenseNet'],
v['discontinuedOperations'],
                                    v['netIncomeFromContinuingOps'],
v['netIncomeApplicableToCommonShares']])

        # Save Annual Balance Sheet in CSV
        with open('Annual_Balance_Sheet_' + main.company + '.csv', mode='w') as
result_w:
            result = csv.writer(result_w, delimiter=',', quotechar='"',
quoting=csv.QUOTE_MINIMAL)
            result.writerow(['Year', 'intangibleAssets', 'totalLiab',
'totalStockholderEquity', 'otherCurrentLiab', 'totalAssets',
                                'commonStock', 'otherCurrentAssets', 'retainedEarnings',
'otherLiab', 'goodWill', 'treasuryStock',
                                'otherAssets', 'cash', 'totalCurrentLiabilities',
'deferredLongTermAssetCharges', 'otherStockholderEquity',
                                'propertyPlantEquipment', 'totalCurrentAssets',
'longTermInvestments', 'netTangibleAssets',
                                'shortTermInvestments', 'netReceivables', 'longTermDebt',
'accountsPayable'])
            for i in self.a_balance['balanceSheetHistory']['GOOGL']:
                for k, v in i.items():
                    #  tp.transport(v)
                    result.writerow(
                                    [k, v['intangibleAssets'], v['totalLiab'],
v['totalStockholderEquity'], v['otherCurrentLiab'],
                                    v['totalAssets'], v['commonStock'],
v['otherCurrentAssets'], v['retainedEarnings'], v['otherLiab'],
                                    v['goodWill'], v['treasuryStock'], v['otherAssets'],
v['cash'], v['totalCurrentLiabilities'],
                                    v['deferredLongTermAssetCharges'],
v['otherStockholderEquity'],
                                    v['propertyPlantEquipment'], v['totalCurrentAssets'],
v['longTermInvestments'],
                                    v['netTangibleAssets'], v['shortTermInvestments'],
v['netReceivables'], v['longTermDebt'],
                                    v['accountsPayable']])

        # Save Annual Cash Flow in CSV
        with open('Annual_Cash_Flow_' + main.company + '.csv', mode='w') as result_w:
            result = csv.writer(result_w, delimiter=',', quotechar='"',
quoting=csv.QUOTE_MINIMAL)
            result.writerow(['Year', 'investments', 'changeToLiabilities',
'totalCashflowsFromInvestingActivities', 'netBorrowings',
```

```
                                'totalCashFromFinancingActivities',
'changeToOperatingActivities', 'netIncome', 'changeInCash',
                                'repurchaseOfStock', 'effectOfExchangeRate',
'totalCashFromOperatingActivities', 'depreciation',
                                'otherCashflowsFromInvestingActivities',
'changeToAccountReceivables', 'otherCashflowsFromFinancingActivities',
                                'changeToNetincome', 'capitalExpenditures'])
            for i in self.a_cash['cashflowStatementHistory']['GOOGL']:
                for k, v in i.items():
                  #  tp.transport(v)
                    result.writerow(
                        [k, v['investments'], v['changeToLiabilities'],
v['totalCashflowsFromInvestingActivities'], v['netBorrowings'],
                         v['totalCashFromFinancingActivities'],
v['changeToOperatingActivities'], v['netIncome'], v['changeInCash'],
                         v['repurchaseOfStock'], v['effectOfExchangeRate'],
v['totalCashFromOperatingActivities'], v['depreciation'],
                         v['otherCashflowsFromInvestingActivities'],
v['changeToAccountReceivables'], v['otherCashflowsFromFinancingActivities'],
                         v['changeToNetincome'], v['capitalExpenditures']])
```

Transport.py

```
import requests

# Dont forget to port-forward Graylog Server to localhost if server is not running
locally
graylog_url = "http://127.0.0.1:12201/gelf"

def  transport(data):
    header = {"Content-type": "application/x-www-form-urlencoded", "Accept":
"text/plain"}
    payload = "{'host':'yahoofinancials_itu.org', 'short_message':'sako'," +
str(data).replace('{', '').replace('}', '') + "}".replace('\'', '"')
    response_code=requests.post(graylog_url, data=payload, headers=header)
    if response_code == '202':
        print('Data was successfully sent to Graylog Server')
    else:
        print('Failed to sent data to Graylog Server')
```

Prediction.py

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import datetime
import main

# Predict 60 days stock price based on close price
class Prediction:
    def __init__(self):
        self.data = pd.read_csv('Stock_prices_'+main.company+'.csv', parse_dates=True)
        predict_out = int(60)
        self.data['Prediction'] = self.data[['adjclose']].shift(-predict_out)
        X = np.array(self.data.drop(['Date','Prediction'], 1))
        X = preprocessing.scale(X)
        self.X_forecast = X[-predict_out:]
```

```
        X = X[:-predict_out]
        y = np.array(self.data['Prediction'])
        y = y[:-predict_out]
        self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(X, y,
test_size=0.2)

    # Training
    def training_data(self):
        self.clf = LinearRegression()
        self.clf.fit(self.X_train, self.y_train)

    # Testing
    def testing_data(self):
        confidence = self.clf.score(self.X_test, self.y_test)
        self.forecast_prediction = self.clf.predict(self.X_forecast)
        print("confidence: ", int(confidence*100),'%')
        print("Forecast for coming 60 days: ")
        print(main.mark)
        print(self.forecast_prediction)

    # Predicting
    def predict_price(self):
        plt.figure(1)
        plt.subplot(221)
        plt.plot(self.data['adjclose'])
        plt.title('Google Stock Exchange History graph')
        plt.grid(True)
        plt.ylabel('Stock price')
        plt.xlabel('Between:1/2/2008 - 12/29/2017, 2758 days.')
        plt.subplot(222)
        plt.title('Google Stock Exchange Prediction graph')
        plt.grid(True)
        plt.plot(self.forecast_prediction)
        current_date=datetime.datetime.today().strftime('%Y-%m-%d')
        plt.xlabel('Starting from '+str(current_date)+' till coming 60 days.')
        plt.show()
```

Graylog Deployment File to run in Kubernetes Cluster

```
## Namespace

apiVersion: v1
kind: Namespace
metadata:
  name: graylog
---
### Graylog

apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  creationTimestamp: null
  name: graylog
  namespace: graylog
spec:
  replicas: 1
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
```

```
      labels:
        service: graylog
    spec:
      containers:
      - env:
        - name: GRAYLOG_WEB_ENDPOINT_URI
          value: http://127.0.0.1:9000/api
        - name: GRAYLOG_ELASTICSEARCH_HOSTS
          value: "http://elasticsearch:9200/"
        - name: GRAYLOG_PASSWORD_SECRET
          value: somesaltpassword
        - name: GRAYLOG_ROOT_PASSWORD_SHA2
          value: 5D5E792708BFA15F0AB42E817B4E69379777D2722E0529DFB031C0B847DB137D
        image: graylog2/server:2.4.3-1
        name: graylog
        ports:
        - containerPort: 9000
        - containerPort: 12201
        resources: {}
      restartPolicy: Always
status: {}
---
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    service: graylog
  name: graylog
  namespace: graylog
spec:
  ports:
  - name: "9000"
    port: 9000
    targetPort: 9000
  - name: "12201"
    port: 12201
    targetPort: 12201
  selector:
    service: graylog
  type: LoadBalancer
status:
  loadBalancer: {}
---
### ElasticSearch

apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  creationTimestamp: null
  name: elasticsearch
  namespace: graylog
spec:
  replicas: 1
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        service: elasticsearch
    spec:
```

```yaml
      containers:
      - args:
        - elasticsearch
        - -Des.cluster.name=graylog
        image: elasticsearch:2
        name: elasticsearch
        resources: {}
      restartPolicy: Always
status: {}
---
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  namespace: graylog
  labels:
    service: elasticsearch
  name: elasticsearch
spec:
  clusterIP: None
  ports:
  - name: headless
    port: 55555
    targetPort: 0
  selector:
    service: elasticsearch
status:
  loadBalancer: {}
---

### MongoDB

apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  creationTimestamp: null
  name: mongo
  namespace: graylog
spec:
  replicas: 1
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        service: mongo
    spec:
      containers:
      - image: mongo:3
        name: mongo
        resources: {}
      restartPolicy: Always
status: {}
---
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    service: mongo
  name: mongo
```

```
  namespace: graylog
spec:
  clusterIP: None
  ports:
  - name: headless
    port: 55555
    targetPort: 0
  selector:
    service: mongo
status:
  loadBalancer: {}
---
```

Graylog port-forward to localhost shell script

```
#!/bin/bash
# port-forward graylog
NAMESPACE=${1:-graylog}
export POD=$(kubectl -n $NAMESPACE get pod -l service=graylog -o
jsonpath='{.items[0].metadata.name}')
kubectl port-forward --namespace $NAMESPACE $POD 9000:9000 &
kubectl port-forward --namespace $NAMESPACE $POD 12201:12201
```

Appendix B Google Daily Stock Prices

```
Date,high,low,adjclose,close,open,volume
2018-12-
03,1135.0,1111.72998046875,1116.3599853515625,1116.3599853515625,1132.1600341796875,28
76600
2018-12-
04,1114.239990234375,1060.780029296875,1062.469970703125,1062.469970703125,1112.989990
234375,2281500
2018-12-
06,1079.4200439453125,1042.47998046875,1078.0799560546875,1078.0799560546875,1045.0,23
71800
2018-12-
07,1085.030029296875,1039.3699951171875,1046.5799560546875,1046.5799560546875,1072.229
98046875,2134100
2018-12-
10,1059.5999755859375,1033.0,1053.1800537109375,1053.1800537109375,1042.93994140625,16
82600
2018-12-
11,1070.4000244140625,1050.0999755859375,1061.6500244140625,1061.6500244140625,1066.93
994140625,1692600
2018-12-
12,1091.72998046875,1071.969970703125,1073.72998046875,1073.72998046875,1077.079956054
6875,1447400
```

Appendix C Google Annual Income Statement

```
Year    researchDevelopment    incomeBeforeTax    minorityInterest   netIncome
effectOfAccountingCharges  sellingGeneralAdministrative   grossProfit    ebit
operatingIncome    otherOperatingExpenses interestExpense    extraordinaryItems
nonRecurring   otherItems incomeTaxExpense   totalRevenue   totalOperatingExpenses
costOfRevenue  totalOtherIncomeExpenseNet discontinuedOperations
netIncomeFromContinuingOps netIncomeApplicableToCommonShares
12/31/17   16625000000    27193000000       12662000000       19765000000
65272000000    28882000000    28882000000       -109000000          14531000000
110855000000 81973000000    45583000000    -1689000000       12662000000
12662000000
12/31/16   13948000000    24150000000       19478000000       17470000000
55134000000    23716000000    23716000000       -124000000          4672000000
90272000000    66556000000    35138000000    434000000     19478000000    19478000000
12/31/15   12282000000    19651000000       16348000000       15183000000
46825000000    19360000000    19360000000       -104000000          3303000000
74989000000    55629000000    28164000000    291000000     16348000000    16348000000
12/31/14   9832000000 17259000000       14136000000       13982000000    40688000000
16874000000    16874000000       -101000000          3639000000
```

Appendix D Google Annual Balance Sheet

```
Year,intangibleAssets,totalLiab,totalStockholderEquity,otherCurrentLiab,totalAssets,co
mmonStock,otherCurrentAssets,retainedEarnings,otherLiab,goodWill,treasuryStock,otherAs
sets,cash,totalCurrentLiabilities,deferredLongTermAssetCharges,otherStockholderEquity,
propertyPlantEquipment,totalCurrentAssets,longTermInvestments,netTangibleAssets,shortT
ermInvestments,netReceivables,longTermDebt,accountsPayable
2017-12-
31,2692000000,44793000000,152502000000,10651000000,197295000000,40247000000,2983000000
,113247000000,16641000000,16747000000,-
992000000,3352000000,10715000000,24183000000,680000000,-
992000000,42383000000,124308000000,7813000000,133063000000,91156000000,18705000000,394
3000000,3137000000
2016-12-
31,3307000000,28461000000,139036000000,5851000000,167497000000,36307000000,3175000000,
105131000000,7770000000,16468000000,-
2402000000,2202000000,12918000000,16756000000,383000000,-
2402000000,34234000000,105408000000,5878000000,119261000000,73415000000,15632000000,39
35000000,2041000000
2015-12-
31,3847000000,27130000000,120331000000,4327000000,147461000000,32982000000,1590000000,
89223000000,5825000000,15869000000,-
1874000000,3432000000,15409000000,19310000000,251000000,-
1874000000,29016000000,90114000000,5183000000,100615000000,56517000000,13459000000,199
5000000,1931000000
2014-12-
31,4607000000,25327000000,103860000000,2803000000,129187000000,28767000000,2637000000,
75066000000,5320000000,15599000000,27000000,3363000000,16585000000,16779000000,1760000
00,27000000,23883000000,78656000000,3079000000,83654000000,46048000000,9974000000,2992
000000,1715000000
```

Appendix E Google Annual Cash Flow

```
Year,investments,changeToLiabilities,totalCashflowsFromInvestingActivities,netBorrowin
gs,totalCashFromFinancingActivities,changeToOperatingActivities,netIncome,changeInCash
,repurchaseOfStock,effectOfExchangeRate,totalCashFromOperatingActivities,depreciation,
otherCashflowsFromInvestingActivities,changeToAccountReceivables,otherCashflowsFromFin
ancingActivities,changeToNetincome,capitalExpenditures
2017-12-31,-19448000000,1121000000,-31401000000,-86000000,-
8298000000,3682000000,12662000000,-2203000000,-
4846000000,405000000,37091000000,6899000000,1419000000,-3768000000,-
3366000000,8284000000,-13184000000
2016-12-31,-18229000000,333000000,-31165000000,-1335000000,-
8332000000,2420000000,19478000000,-3631000000,-3693000000,-
170000000,36036000000,6100000000,-1978000000,-2578000000,-3304000000,7158000000,-
10212000000
2015-12-31,-13635000000,246000000,-23711000000,-23000000,-
4225000000,1618000000,16348000000,-1798000000,-1780000000,-
434000000,26572000000,5024000000,75000000,-2094000000,-2422000000,5609000000,-
9950000000
2014-12-31,-6222000000,261000000,-21055000000,-18000000,-
2087000000,1461000000,14136000000,-551000000,-1780000000,-
433000000,23024000000,4601000000,628000000,-1641000000,-2069000000,3615000000,-
11014000000
```