Analyzing the Behavior of Exchanges in United States

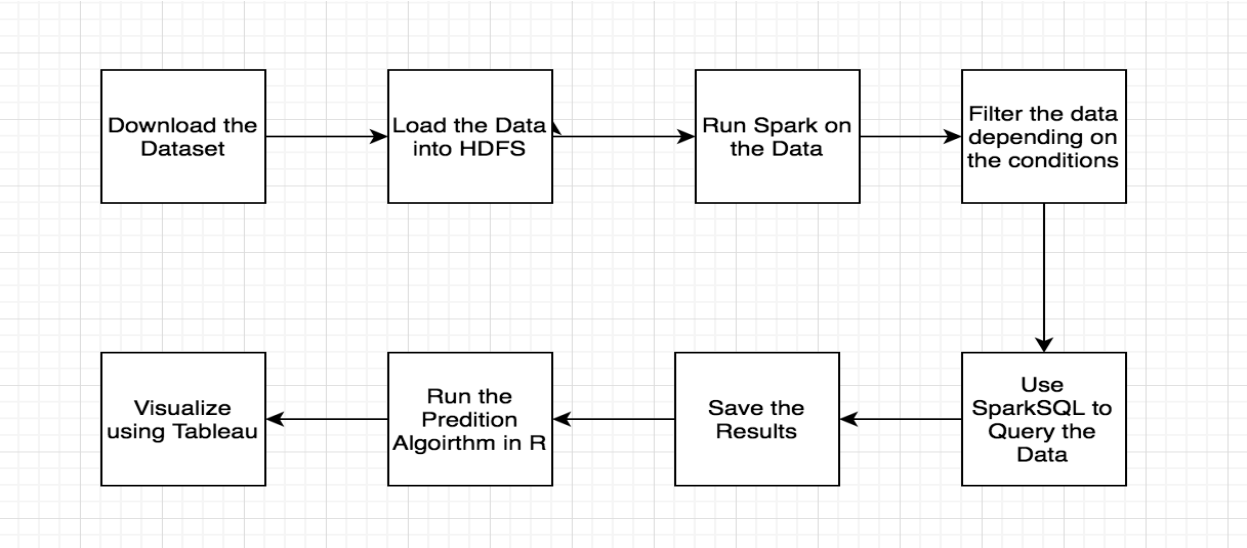
Big Data Project Report by

Saloni Bindra ***Monica Reddy Tirupari***

Overview

The main purpose is to find different trends in the working of exchanges in United States. A lot of work has been done in predicting the behavior of stocks from the history of stock process. However, here we have considered the data of all the stocks traded in an exchange, this data is nothing but the messages passed for various stages of ordering shares.

Architecture:



Technologies used:

- **Spark (PySpark)** - Loading and Filtering the Data
- **SparkSQL** – Query Processing
- **R** – Prediction Algorithm
- **Tableau** and **Python Plotly** - Visualizations

Dataset:

- Provided by US Securities and Exchange Commission
- It provides metrics for each individual security for each exchange
- Attributes: Ticker, Date, Security, McapRank, TurnRank, VolatilityRank, PriceRank, Cancels, Trades, LitTrades, OddLots, Hidden, TradesForHidden, OrderVol, TradeVol, LitVol, OddLotVol, HiddenVol, TradeVolForHidden

  The following is the URL for the dataset used:
  https://www.sec.gov/opa/data/market-structure/market-structure-data-security-and-exchange.html

Details about the dataset:

- Order-based exchanges: Arca, Bats-Y, Bats-Z, Boston, CHX, Edga-A, Edge-X, Nasdaq, NSX, PHLX
- Level-book exchanges: Amex, NYSE
- Exchanges denote trades against hidden orders: Arca, Bats-Y, Bats-Z, Boston, Edga-A, Edge-X, Nasdaq, PHLX
- Exchanges do not denote trades against hidden orders: Amex, CHX, NSX, NYSE
- Exchanges do not report trades against individual orders: Amex, NYSE
- Trades, Trade Volume, Hidden, Hidden Volume, Odd Lots and Odd Lot Volume are counted/summed from the direct feeds for: Arca, Bats-Y, Bats-Z, Boston, Edga-A, Edge-X, Nasdaq, PHLX
- Trades, Trade Volume, Odd Lots and Odd Lot Volume are counted/summed from the direct feeds for: CHX, NSX
- Trades, Trade Volume, Odd Lots and Odd Lot Volume are counted/summed from the SIP feeds for: Amex, NYSE
- Hidden and Hidden Volume are computed as the difference between Trades/Trade Volume reported on the SIP feed and Trades/Trade Volume reported on the direct feeds for: Amex, NYSE
- Hidden and Hidden Volume are not reported for: CHX, NSX

Terminologies:

For monthly (Date, Ticker, Exchange) data:
Trades: Count of all trade messages.
TradeVol('000): Sum of trade volume for all trade messages.
Cancels: Count of all cancel messages, either full or partial, for all exchanges.
Hidden: Count of trades against hidden orders from exchanges that report trades against hidden orders. Difference between SIP Trades and Direct Trades for NYSE and Amex.
LitTrades: Count of all trade messages for trades that are not against hidden orders. Computed as the difference between Trades and Hidden.
LitVol('000): Sum of trade volume for trades that are not against hidden orders. Computed as the difference between Trade Volume and Hidden Volume.
OrderVol('000): Sum of order volume for all add order messages.
TradesForHidden: Count of trades from exchanges that report trades against hidden orders.
HiddenVol('000): Sum of trade volume for trades against hidden orders from exchanges that report trades against hidden orders.
TradeVolForHidden('000): Sum of trade volume from exchanges that report trades against hidden orders.
OddLots: Count of odd lot trade messages for all exchanges.
OddLotVol('000): Sum of odd lot trade volume for all exchanges.

For quarterly (Date, Ticker) data:
TradesForOddLots = Trades: Count of trades from order-based exchanges.
TradeVolForOddLots('000) = TradeVol('000): Sum of trade volume from order-based exchanges.

The metrics:

Cancel-to-Trade = Cancels/LitTrades for Arca, Bats-Y, Bats-Z, Boston, CHX, Edga-A, Edge-X, Nasdaq, NSX, PHLX
Trade-to-Order-Volume = Sum of Total Amount of Trade/Sum of Total amount of Order for All Exchanges
Hidden Rate = 100*Hidden/TradesForHidden for Arca, Bats-Y, Bats-Z, Boston, Edga-A, Edge-X, Nasdaq, PHLX (& Amex and NYSE from 2014)
Hidden Volume = 100*HiddenVol/TradeVolForHidden for Arca, Bats-Y, Bats-Z, Boston, Edga-A, Edge-X, Nasdaq, PHLX (& Amex and NYSE from 2014)
Oddlot Rate=100*OddLots/TradesForOddLots for Arca, Bats-Y, Bats-Z, Boston, CHX, Edga-A, Edge-X, Nasdaq, NSX, PHLX
Oddlot Volume=100*OddLotVol/TradeVolForOddLots for Arca, Bats-Y, Bats-Z, Boston, CHX, Edga-A, Edge-X, Nasdaq, NSX, PHLX

List of Exchanges:

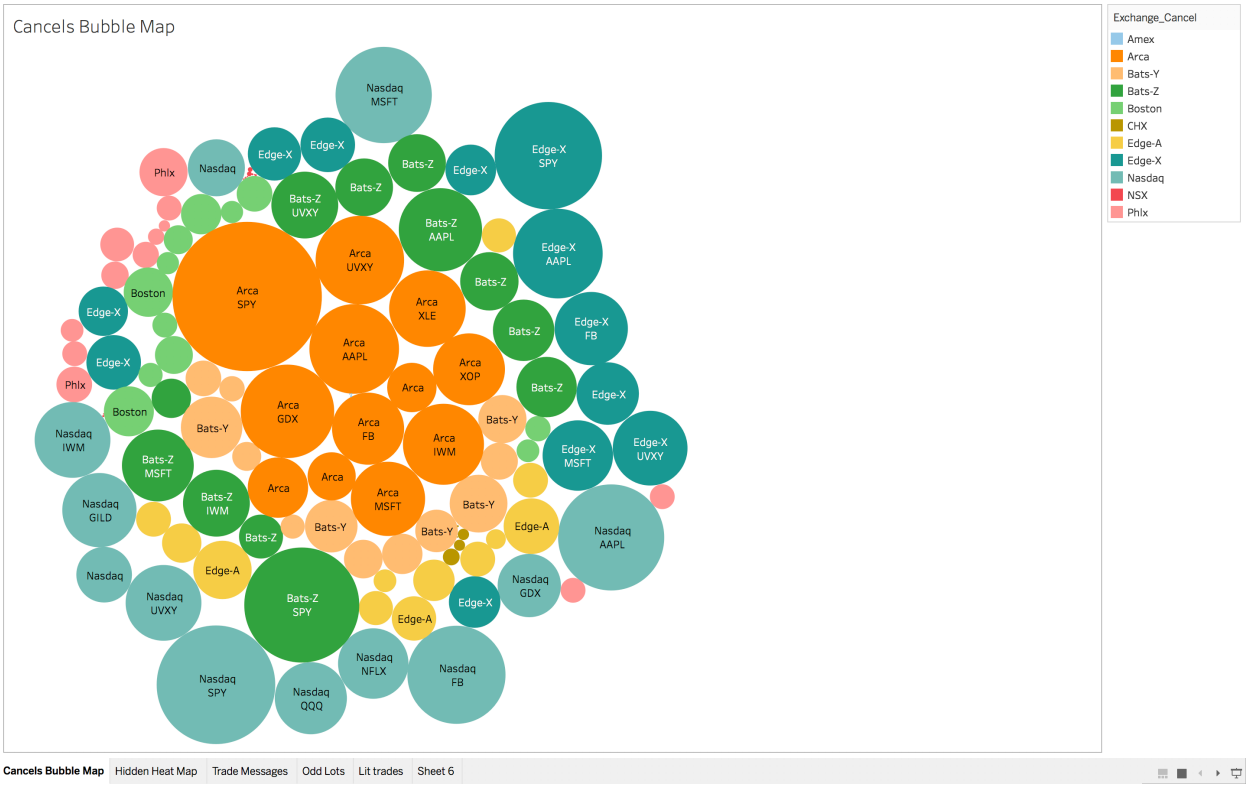
- ACRA
- PHIX
- NSX
- NASDAQ
- EDGE-X
- EDGE –A
- CHX
- BOSTON
- BATS – Z
- BATS –Y
- AMEX
- NYSE

Since the data has around 5717 stocks for each exchange, for visualization purpose we have considered the following stocks.
Please note: the code has been run on all the data and the results for each of these 5717 stocks have been taken into consideration. Only top 15 stocks have been chosen for visualization purpose

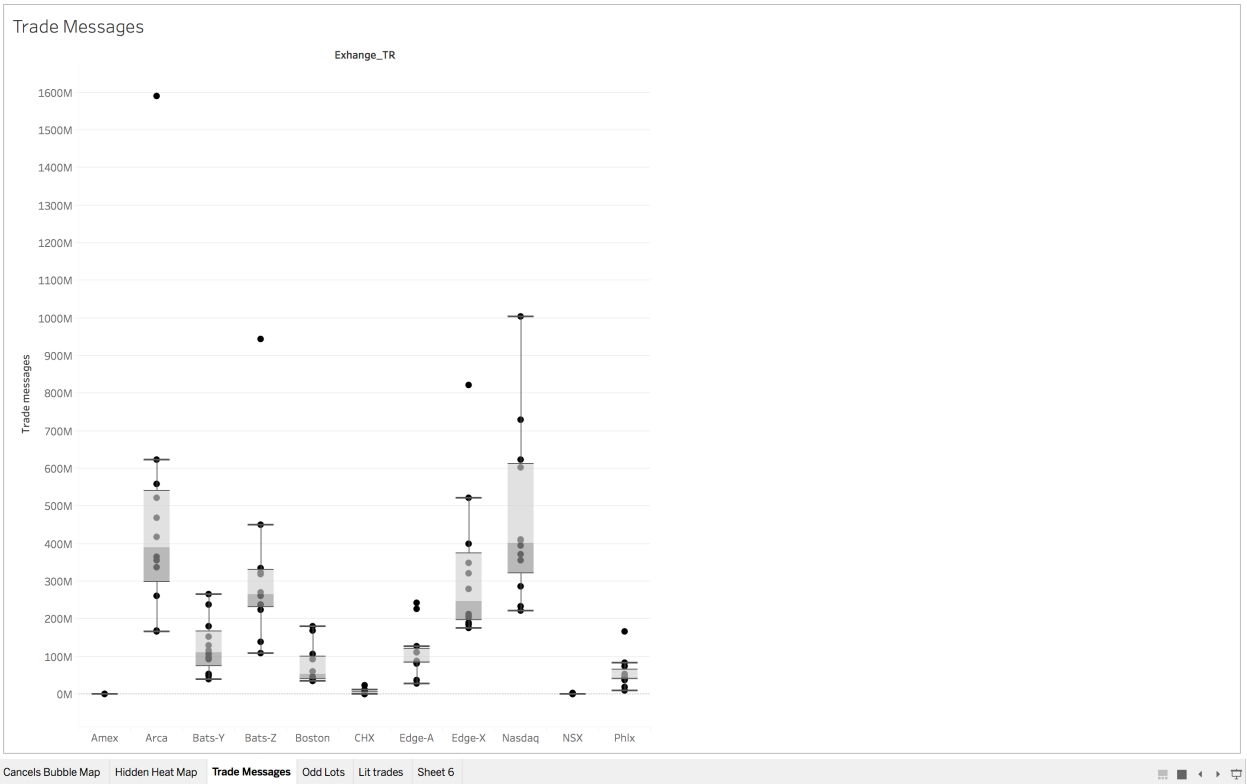
Sample Stocks

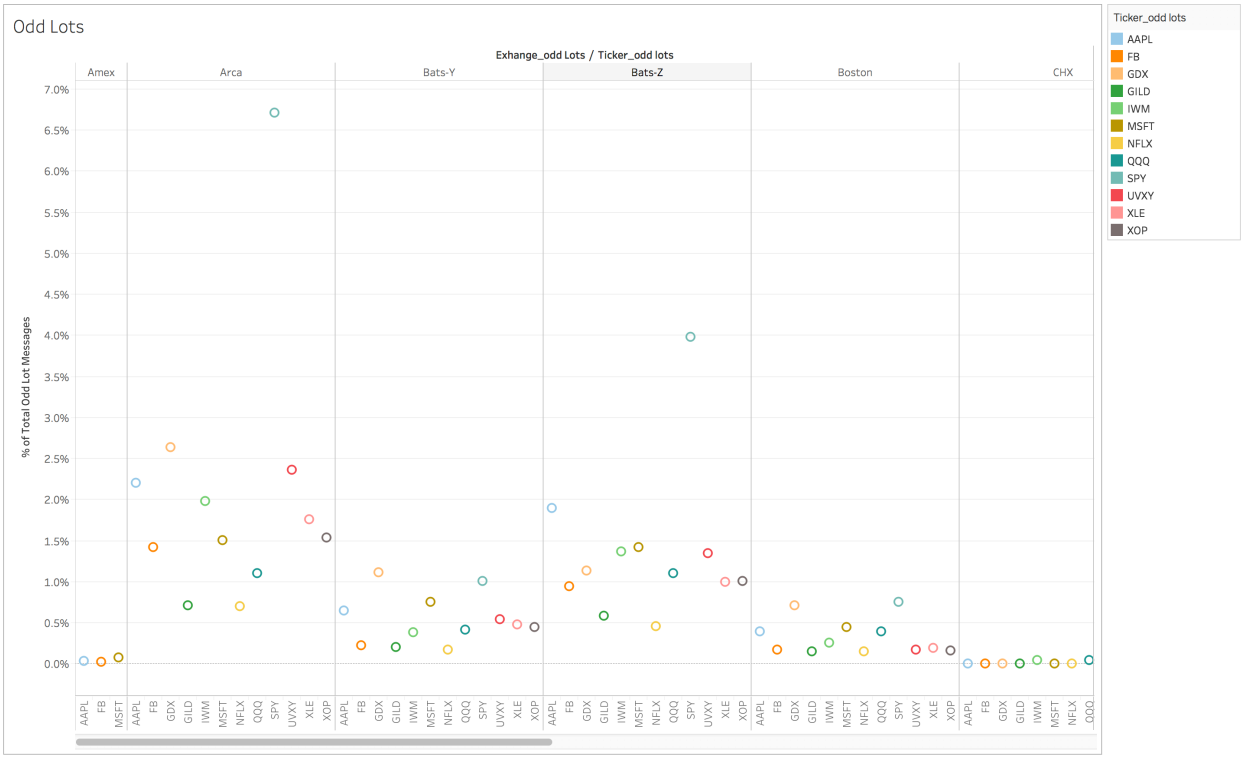
- SPY – SnP500
- AAPL – Apple
- FB - Facebook
- GDX – VanEck Investment
- MSFT - Microsoft
- UVXY – PowerShare ETF
- GOOGL – Alphabet Inc
- IWM – Index Fund
- XLE – Energy Sector
- GILD - Gilead Sciences
- QQQ- Powershare Series
- XOP – SNP Oil & Gas
- NFLX – Netflix

Visualizations:

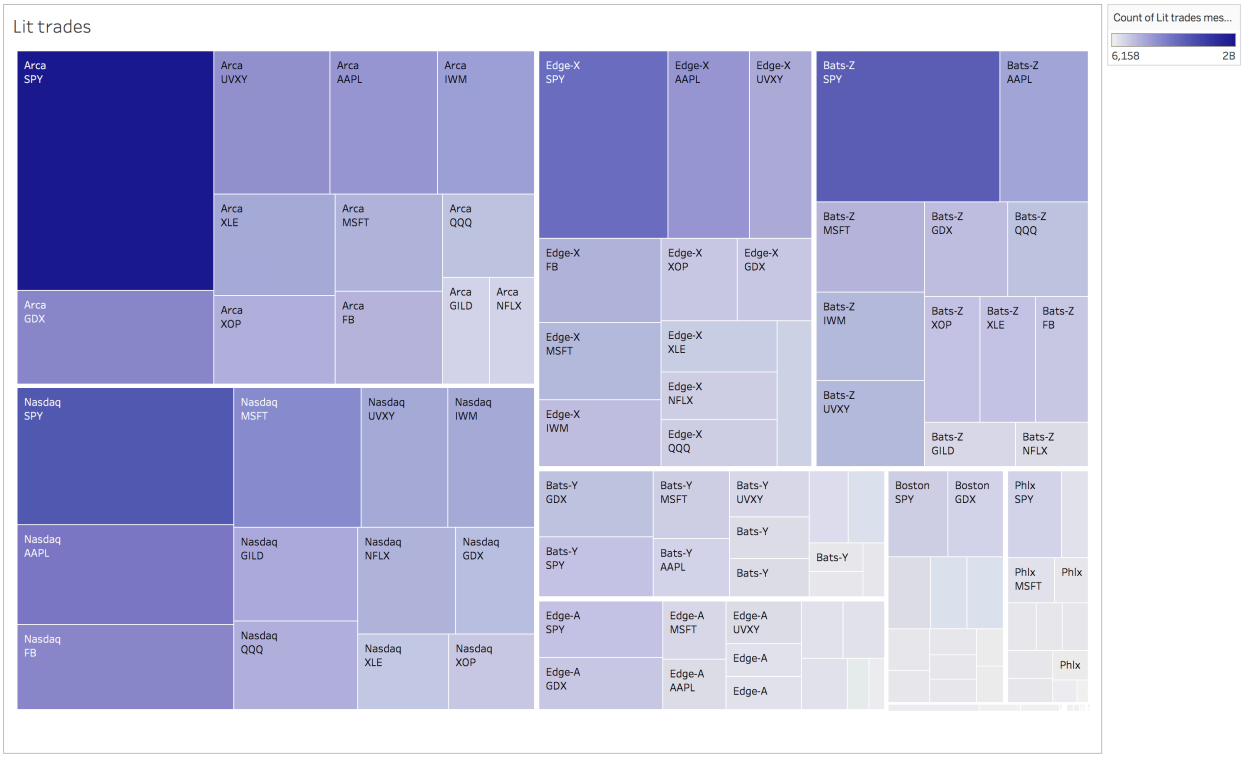



Hidden Heat Map
Exchange_Hidden
Ticker_Hid..
Amex
Arca
Bats-Y
Bats-Z
Boston
CHX
Edge-A
Edge-X
Nasdaq
NSX
Phlx
AAPL
FB
GDX
GILD
IWM
MSFT
NFLX
QQQ
SPY
UVXY
XLE
XOP
SUM(Hidden Messages)
6,158
2B
Cancels Bubble Map
Hidden Heat Map
Trade Messages
Odd Lots
Lit trades
Sheet 6


Trade Messages
Exhange_TR
Trade messages
1600M
1500M
1400M
1300M
1200M
1100M
1000M
900M
800M
700M
600M
500M
400M
300M
200M
100M
0M
Amex
Arca
Bats-Y
Bats-Z
Boston
CHX
Edge-A
Edge-X
Nasdaq
NSX
Phlx
Cancels Bubble Map
Hidden Heat Map
Trade Messages
Odd Lots
Lit trades
Sheet 6



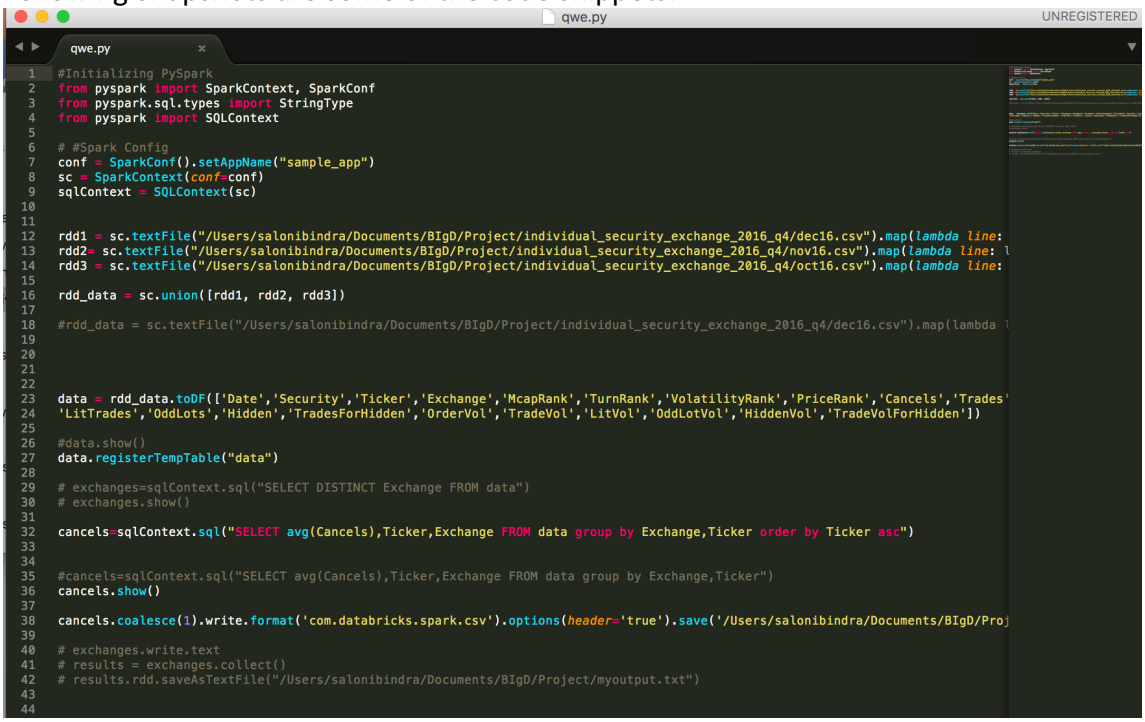
Cancels Bubble Map | Hidden Heat Map | Trade Messages | **Odd Lots** | Lit trades | Sheet 6

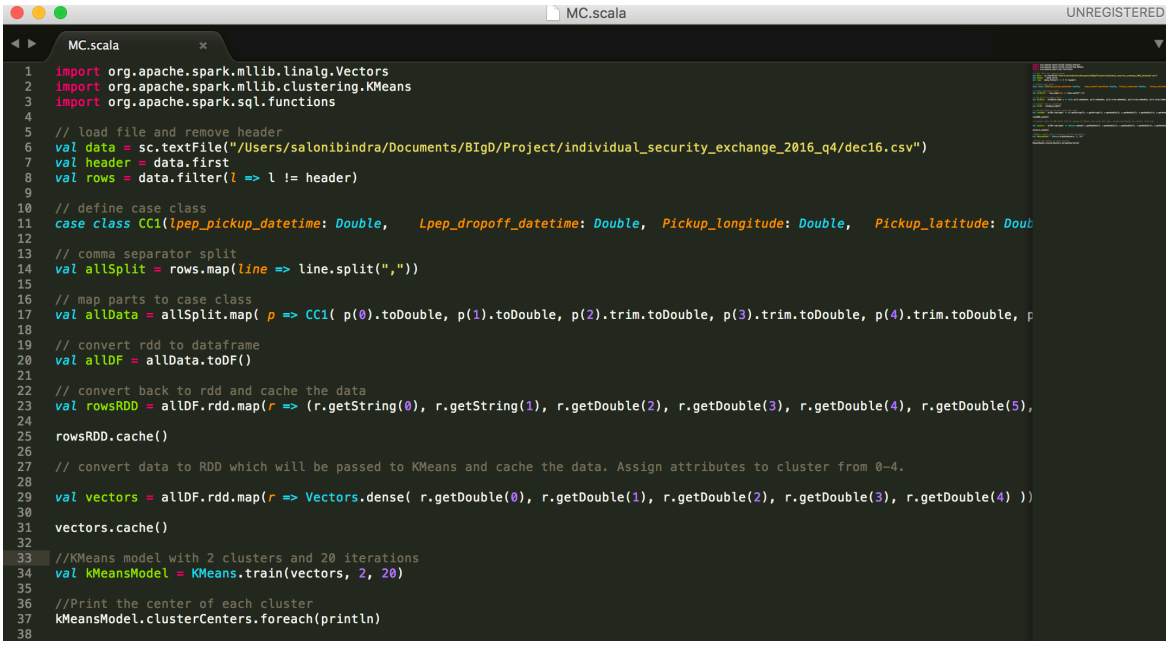


Cancels Bubble Map | Hidden Heat Map | Trade Messages | Odd Lots | **Lit trades** | Sheet 6

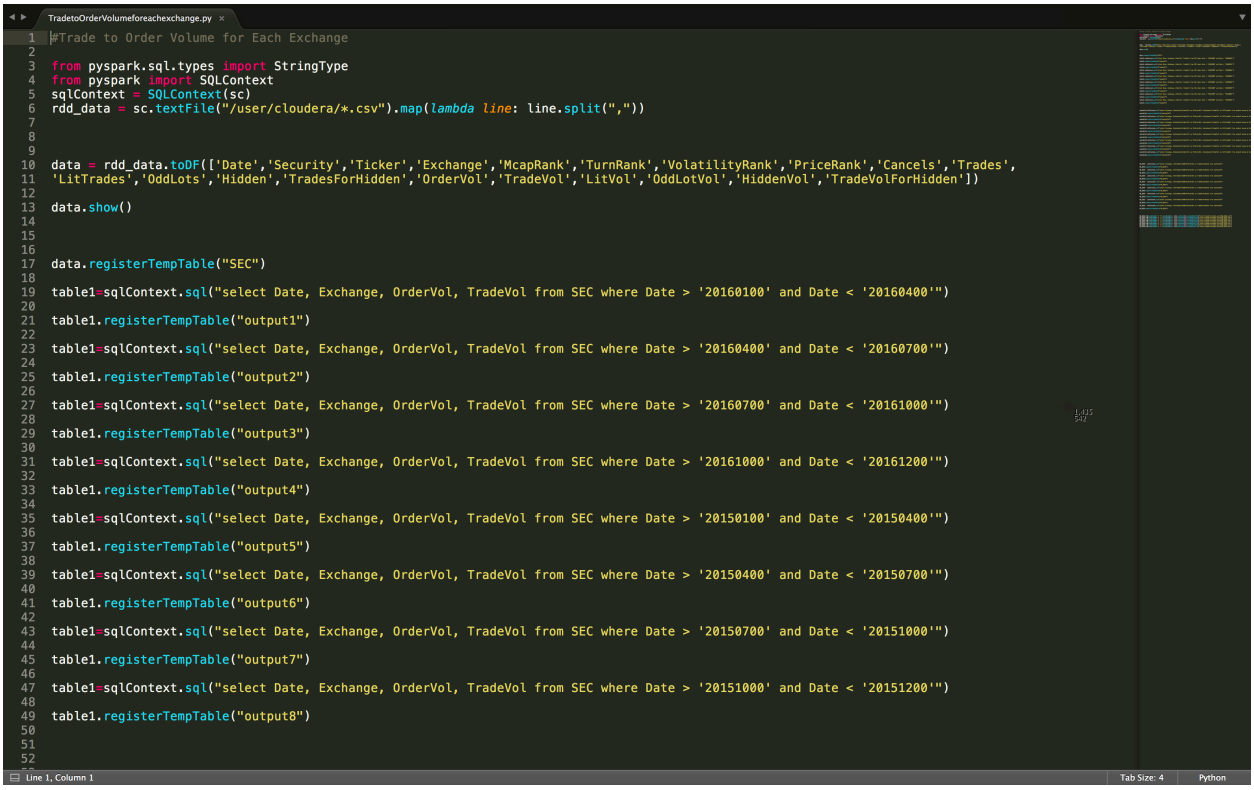
Code and output files for the above plotted figures can also be found at:
https://github.com/salonibindra2050/Big-Data-Analytics

Following Snapshots are some of the code snippets:

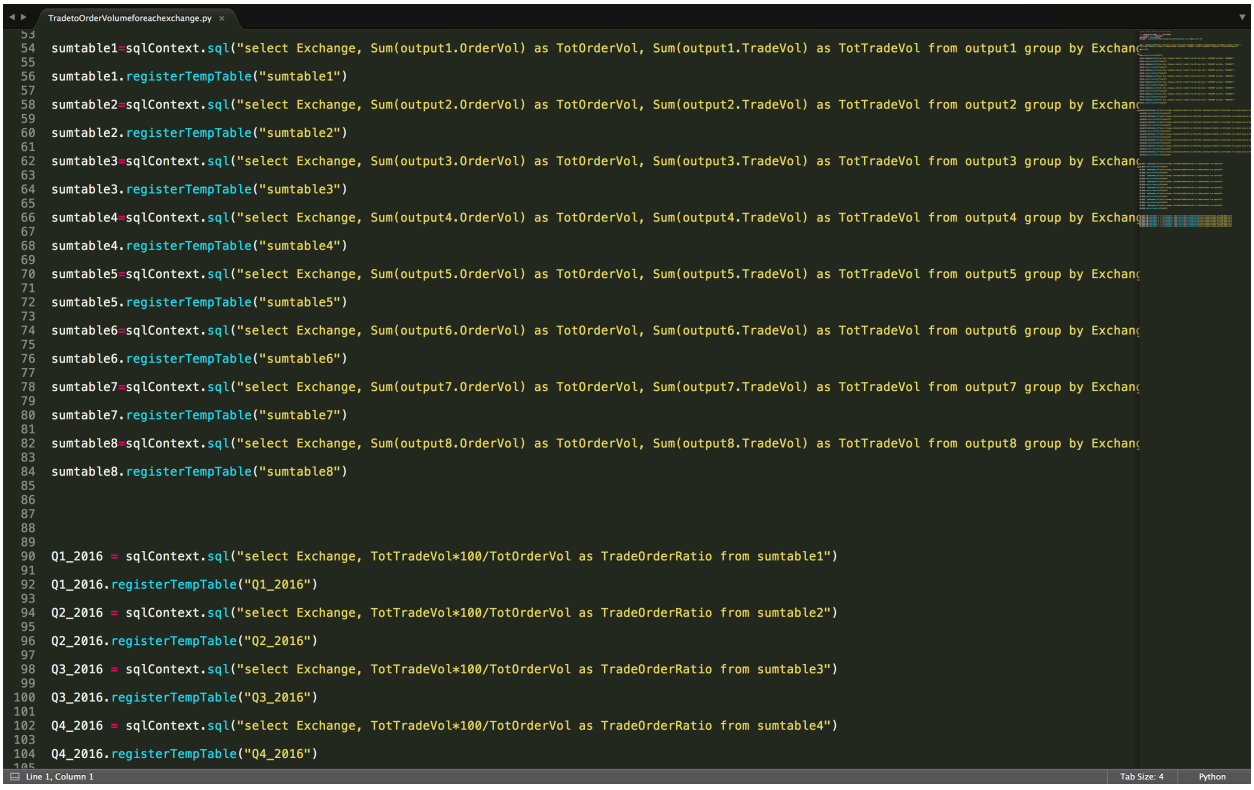




Trade to Order Volume for Each Exchange:



```
#Trade to Order Volume for Each Exchange

from pyspark.sql.types import StringType
from pyspark import SQLContext
sqlContext = SQLContext(sc)
rdd_data = sc.textFile("/user/cloudera/*.csv").map(lambda line: line.split(","))



data = rdd_data.toDF(['Date','Security','Ticker','Exchange','McapRank','TurnRank','VolatilityRank','PriceRank','Cancels','Trades',
'LitTrades','OddLots','Hidden','TradesForHidden','OrderVol','TradeVol','LitVol','OddLotVol','HiddenVol','TradeVolForHidden'])

data.show()



data.registerTempTable("SEC")

table1=sqlContext.sql("select Date, Exchange, OrderVol, TradeVol from SEC where Date > '20160100' and Date < '20160400'")

table1.registerTempTable("output1")

table1=sqlContext.sql("select Date, Exchange, OrderVol, TradeVol from SEC where Date > '20160400' and Date < '20160700'")

table1.registerTempTable("output2")

table1=sqlContext.sql("select Date, Exchange, OrderVol, TradeVol from SEC where Date > '20160700' and Date < '20161000'")

table1.registerTempTable("output3")

table1=sqlContext.sql("select Date, Exchange, OrderVol, TradeVol from SEC where Date > '20161000' and Date < '20161200'")

table1.registerTempTable("output4")

table1=sqlContext.sql("select Date, Exchange, OrderVol, TradeVol from SEC where Date > '20150100' and Date < '20150400'")

table1.registerTempTable("output5")

table1=sqlContext.sql("select Date, Exchange, OrderVol, TradeVol from SEC where Date > '20150400' and Date < '20150700'")

table1.registerTempTable("output6")

table1=sqlContext.sql("select Date, Exchange, OrderVol, TradeVol from SEC where Date > '20150700' and Date < '20151000'")

table1.registerTempTable("output7")

table1=sqlContext.sql("select Date, Exchange, OrderVol, TradeVol from SEC where Date > '20151000' and Date < '20151200'")

table1.registerTempTable("output8")
```



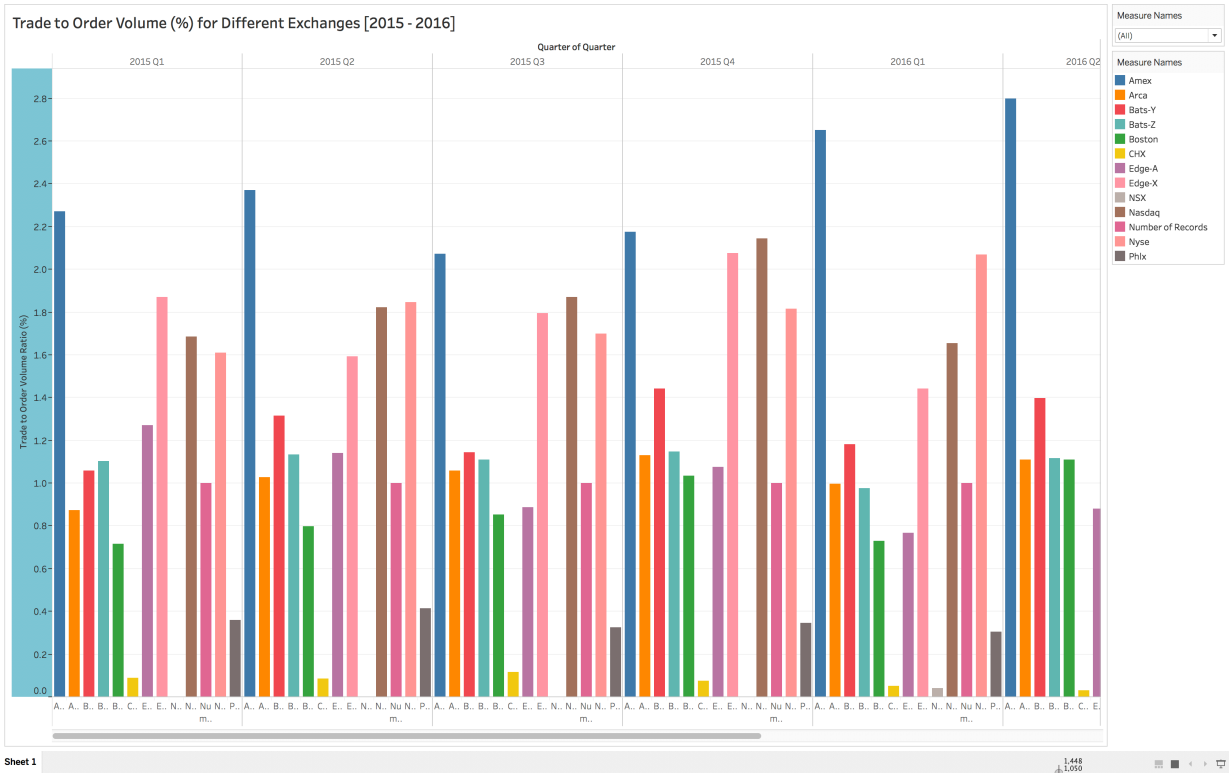
```
sumtable1=sqlContext.sql("select Exchange, Sum(output1.OrderVol) as TotOrderVol, Sum(output1.TradeVol) as TotTradeVol from output1 group by Exchan

sumtable1.registerTempTable("sumtable1")

sumtable2=sqlContext.sql("select Exchange, Sum(output2.OrderVol) as TotOrderVol, Sum(output2.TradeVol) as TotTradeVol from output2 group by Exchan

sumtable2.registerTempTable("sumtable2")

sumtable3=sqlContext.sql("select Exchange, Sum(output3.OrderVol) as TotOrderVol, Sum(output3.TradeVol) as TotTradeVol from output3 group by Exchan

sumtable3.registerTempTable("sumtable3")

sumtable4=sqlContext.sql("select Exchange, Sum(output4.OrderVol) as TotOrderVol, Sum(output4.TradeVol) as TotTradeVol from output4 group by Exchan

sumtable4.registerTempTable("sumtable4")

sumtable5=sqlContext.sql("select Exchange, Sum(output5.OrderVol) as TotOrderVol, Sum(output5.TradeVol) as TotTradeVol from output5 group by Exchan

sumtable5.registerTempTable("sumtable5")

sumtable6=sqlContext.sql("select Exchange, Sum(output6.OrderVol) as TotOrderVol, Sum(output6.TradeVol) as TotTradeVol from output6 group by Exchan

sumtable6.registerTempTable("sumtable6")

sumtable7=sqlContext.sql("select Exchange, Sum(output7.OrderVol) as TotOrderVol, Sum(output7.TradeVol) as TotTradeVol from output7 group by Exchan

sumtable7.registerTempTable("sumtable7")

sumtable8=sqlContext.sql("select Exchange, Sum(output8.OrderVol) as TotOrderVol, Sum(output8.TradeVol) as TotTradeVol from output8 group by Exchan

sumtable8.registerTempTable("sumtable8")



Q1_2016 = sqlContext.sql("select Exchange, TotTradeVol*100/TotOrderVol as TradeOrderRatio from sumtable1")

Q1_2016.registerTempTable("Q1_2016")

Q2_2016 = sqlContext.sql("select Exchange, TotTradeVol*100/TotOrderVol as TradeOrderRatio from sumtable2")

Q2_2016.registerTempTable("Q2_2016")

Q3_2016 = sqlContext.sql("select Exchange, TotTradeVol*100/TotOrderVol as TradeOrderRatio from sumtable3")

Q3_2016.registerTempTable("Q3_2016")

Q4_2016 = sqlContext.sql("select Exchange, TotTradeVol*100/TotOrderVol as TradeOrderRatio from sumtable4")

Q4_2016.registerTempTable("Q4_2016")
```

```
TradetoOrderVolumeforeachexchange.py
sumtable8=sqlContext.sql("select Exchange, Sum(output8.OrderVol) as TotOrderVol, Sum(output8.TradeVol) as TotTradeVol from output8 group by Exchan

sumtable8.registerTempTable("sumtable8")




Q1_2016 = sqlContext.sql("select Exchange, TotTradeVol*100/TotOrderVol as TradeOrderRatio from sumtable1")

Q1_2016.registerTempTable("Q1_2016")

Q2_2016 = sqlContext.sql("select Exchange, TotTradeVol*100/TotOrderVol as TradeOrderRatio from sumtable2")

Q2_2016.registerTempTable("Q2_2016")

Q3_2016 = sqlContext.sql("select Exchange, TotTradeVol*100/TotOrderVol as TradeOrderRatio from sumtable3")

Q3_2016.registerTempTable("Q3_2016")

Q4_2016 = sqlContext.sql("select Exchange, TotTradeVol*100/TotOrderVol as TradeOrderRatio from sumtable4")

Q4_2016.registerTempTable("Q4_2016")

Q1_2015 = sqlContext.sql("select Exchange, TotTradeVol*100/TotOrderVol as TradeOrderRatio from sumtable5")

Q1_2015.registerTempTable("Q1_2015")

Q2_2015 = sqlContext.sql("select Exchange, TotTradeVol*100/TotOrderVol as TradeOrderRatio from sumtable6")

Q2_2015.registerTempTable("Q2_2015")

Q3_2015 = sqlContext.sql("select Exchange, TotTradeVol*100/TotOrderVol as TradeOrderRatio from sumtable7")

Q3_2015.registerTempTable("Q3_2015")

Q4_2015 = sqlContext.sql("select Exchange, TotTradeVol*100/TotOrderVol as TradeOrderRatio from sumtable8")

Q4_2015.registerTempTable("Q4_2015")




Q1_2015.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/Q1_2015.csv")
Q2_2015.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/Q2_2015.csv")
Q3_2015.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/Q3_2015.csv")
Q4_2015.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/Q4_2015.csv")
Q1_2016.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/Q1_2016.csv")
Q2_2016.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/Q2_2016.csv")
Q3_2016.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/Q3_2016.csv")
Q4_2016.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/Q4_2016.csv")

Line 1, Column 1                                                    Tab Size: 4    Python
```

Output result for the above Spark code:

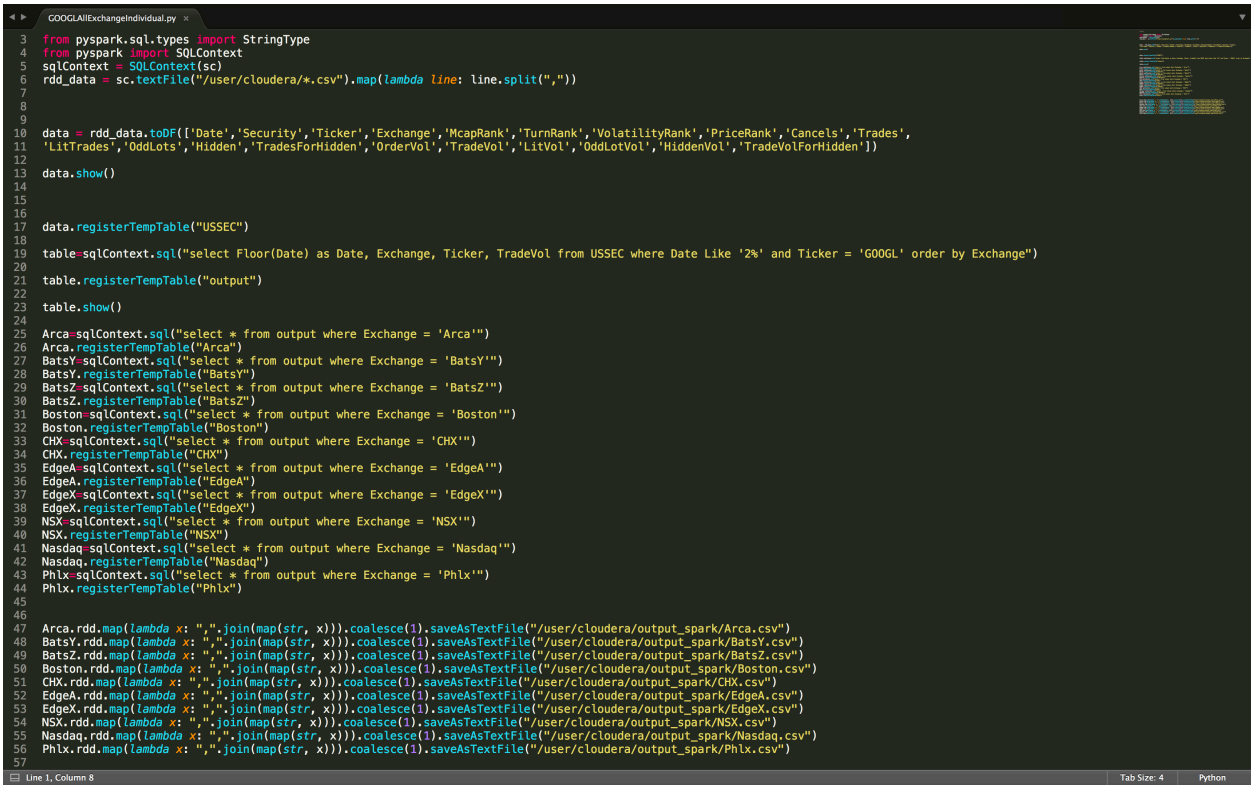
	A	B	C	D	E	F	G	H	I	J	K	L	M
1		Bats-Y	Bats-Z	Arca	Nasdaq	CHX	Edge-A	Edge-X	Amex	Boston	NYSE	Phlx	NSX
2	2015_Q1	1.0595281	1.1031163	0.8751496	1.6844781	0.0881179	1.2712419	1.8716597	2.2717808	0.7164282	1.6090636	0.3582875	0
3	2015_Q2	1.3169208	1.1336805	1.0275861	1.8213652	0.0851081	1.1410816	1.593089	2.3694046	0.7983563	1.8465138	0.4132811	0
4	2015_Q3	1.1449829	1.1089373	1.0583792	1.8715084	0.1150571	0.8863159	1.7968326	2.0731285	0.8539787	1.6981781	0.3244577	0
5	2015_Q4	1.4434592	1.1472597	1.1304819	2.1458822	0.0751042	1.0757764	2.0747723	2.1747208	1.0329694	1.81602	0.3465112	0
6	2016_Q1	1.1832446	0.977673	0.9963385	1.6538641	0.0522932	0.7670111	1.4407295	2.6508201	0.7287646	2.0676642	0.3053261	0.0405044
7	2016_Q2	1.3967295	1.1177446	1.1086052	1.7989071	0.0315688	0.8821608	1.6655781	2.7994547	1.1101132	2.4766579	0.4152328	0.0941428
8	2016_Q3	1.7175449	1.15566	0.9492183	1.7663844	0.025855	1.1524199	1.8819658	2.3651817	1.2813397	2.5287991	0.3947411	0.2020633
9	2016_Q4	1.9476376	1.210121	0.9707156	1.956603	0.0490621	1.2513488	2.0501184	2.2648484	1.4519217	2.6535069	0.4161472	0.5248188
10													

Output is plotted in Tableau:



From the above Trade to Order Volume Ratio (TOR) visualization, we can infer that TOR for Amex has the maximum ratio, followed by NYSE and NASDAQ.

Google Trend Analysis for all Exchanges:



```
from pyspark.sql.types import StringType
from pyspark import SQLContext
sqlContext = SQLContext(sc)
rdd_data = sc.textFile("/user/cloudera/*.csv").map(lambda line: line.split(","))



data = rdd_data.toDF(['Date','Security','Ticker','Exchange','McapRank','TurnRank','VolatilityRank','PriceRank','Cancels','Trades',
'LitTrades','OddLots','Hidden','TradesForHidden','OrderVol','TradeVol','LitVol','OddLotVol','HiddenVol','TradeVolForHidden'])

data.show()



data.registerTempTable("USSEC")

table=sqlContext.sql("select Floor(Date) as Date, Exchange, Ticker, TradeVol from USSEC where Date Like '2%' and Ticker = 'GOOGL' order by Exchange")

table.registerTempTable("output")

table.show()

Arca=sqlContext.sql("select * from output where Exchange = 'Arca'")
Arca.registerTempTable("Arca")
BatsY=sqlContext.sql("select * from output where Exchange = 'BatsY'")
BatsY.registerTempTable("BatsY")
BatsZ=sqlContext.sql("select * from output where Exchange = 'BatsZ'")
BatsZ.registerTempTable("BatsZ")
Boston=sqlContext.sql("select * from output where Exchange = 'Boston'")
Boston.registerTempTable("Boston")
CHX=sqlContext.sql("select * from output where Exchange = 'CHX'")
CHX.registerTempTable("CHX")
EdgeA=sqlContext.sql("select * from output where Exchange = 'EdgeA'")
EdgeA.registerTempTable("EdgeA")
EdgeX=sqlContext.sql("select * from output where Exchange = 'EdgeX'")
EdgeX.registerTempTable("EdgeX")
NSX=sqlContext.sql("select * from output where Exchange = 'NSX'")
NSX.registerTempTable("NSX")
Nasdaq=sqlContext.sql("select * from output where Exchange = 'Nasdaq'")
Nasdaq.registerTempTable("Nasdaq")
Phlx=sqlContext.sql("select * from output where Exchange = 'Phlx'")
Phlx.registerTempTable("Phlx")


Arca.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/Arca.csv")
BatsY.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/BatsY.csv")
BatsZ.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/BatsZ.csv")
Boston.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/Boston.csv")
CHX.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/CHX.csv")
EdgeA.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/EdgeA.csv")
EdgeX.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/EdgeX.csv")
NSX.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/NSX.csv")
Nasdaq.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/Nasdaq.csv")
Phlx.rdd.map(lambda x: ",".join(map(str, x))).coalesce(1).saveAsTextFile("/user/cloudera/output_spark/Phlx.csv")

```

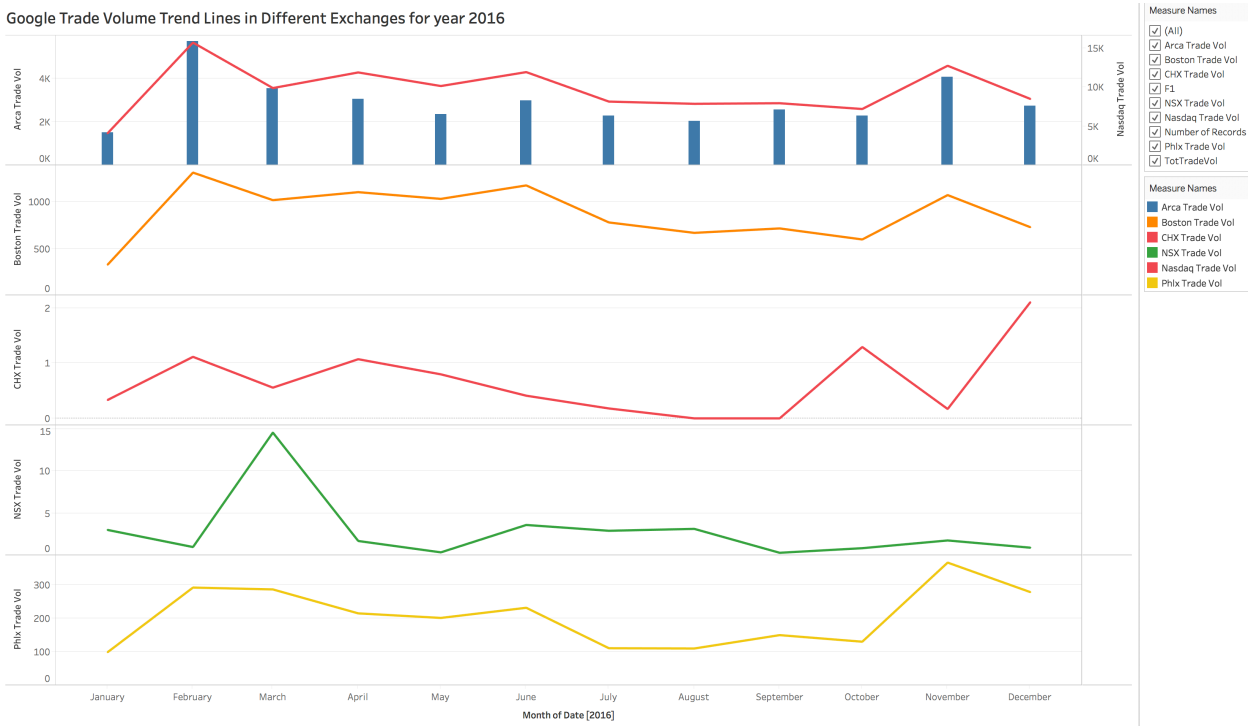
We have used R program to join the tables that have been outputted from the spark using HDFS.

```
Arca <- read.csv("/Users/monicareddytirupari/Desktop/USSEC\ Big\ Data\ Project/GOOGL\ All\ Exchange/Arca.csv")
Boston <- read.csv("/Users/monicareddytirupari/Desktop/USSEC\ Big\ Data\ Project/GOOGL\ All\ Exchange/Boston.csv")
CHX <- read.csv("/Users/monicareddytirupari/Desktop/USSEC\ Big\ Data\ Project/GOOGL\ All\ Exchange/CHX.csv")
Nasdaq <- read.csv("/Users/monicareddytirupari/Desktop/USSEC\ Big\ Data\ Project/GOOGL\ All\ Exchange/Nasdaq.csv")
NSX <- read.csv("/Users/monicareddytirupari/Desktop/USSEC\ Big\ Data\ Project/GOOGL\ All\ Exchange/NSX.csv")
Phlx <- read.csv("/Users/monicareddytirupari/Desktop/USSEC\ Big\ Data\ Project/GOOGL\ All\ Exchange/Phlx.csv")

names(Arca) <- c('Date','Exchange','Ticker','TradeVol')
names(Boston) <- c('Date','Exchange','Ticker','TradeVol')
names(CHX) <- c('Date','Exchange','Ticker','TradeVol')
names(Nasdaq) <- c('Date','Exchange','Ticker','TradeVol')
names(NSX) <- c('Date','Exchange','Ticker','TradeVol')
names(Phlx) <- c('Date','Exchange','Ticker','TradeVol')


df1 <- sqldf("SELECT Date, Arca.TradeVol as ArcaTradeVol, Boston.TradeVol as BostonTradeVol FROM Arca JOIN Boston USING(Date)")
df2 <- sqldf("SELECT Date, ArcaTradeVol, BostonTradeVol, CHX.TradeVol as CHXTradeVol FROM df1 JOIN CHX USING(Date)")
df3 <- sqldf("SELECT Date, ArcaTradeVol, BostonTradeVol, CHXTradeVol, Nasdaq.TradeVol as NasdaqTradeVol
             FROM df2 JOIN Nasdaq USING(Date)")
df4 <- sqldf("SELECT Date, ArcaTradeVol, BostonTradeVol, CHXTradeVol, NasdaqTradeVol,
             NSX.TradeVol as NSXTradeVol FROM df3 JOIN NSX USING(Date)")
GOOGLTrendAnalysis <- sqldf("SELECT Date, ArcaTradeVol, BostonTradeVol, CHXTradeVol, NasdaqTradeVol,
             NSXTradeVol, Phlx.TradeVol as PhlxTradeVol FROM df4 JOIN Phlx USING(Date)")

View(GOOGLTrendAnalysis)

write.csv(GOOGLTrendAnalysis, "/Users/monicareddytirupari/Desktop/GOOGLTrendAnalysis.csv")

```

This is a Snippet of the Output data with the required attributes of Google Trade Volume for different Exchanges:

	A	B	C	D	E	F	G	H	I
1		Date	ArcaTradeVol	BostonTradeVol	CHXTradeVol	NasdaqTradeVol	NSXTradeVol	PhlxTradeVol	TotTradeVol
2	1	20160404	79.919	29.931	0	305.159	0	12.274	427.283
3	2	20160405	77.253	30.61	0	310.742	0	7.431	426.036
4	3	20160406	78.838	28.447	0.3	322.869	0	6.939	437.393
5	4	20160407	76.049	30.148	0	375.1	0	6.848	488.145
6	5	20160408	96.035	27.636	0	316.821	0	6.386	446.878
7	6	20160411	98.327	42.587	0	368.864	0	8.408	518.186
8	7	20160412	104.983	35.838	0	368.812	0	3.811	513.444
9	8	20160413	119.468	33.735	0	407.386	0	12.116	572.705
10	9	20160414	87.23	30.625	0	377.144	0	3.545	498.544
11	10	20160415	88.681	38.1	0	375.422	0	6.141	508.344
12	11	20160418	121.405	50.619	0	443.311	1.2	10.647	627.182
13	12	20160419	151.303	61.881	0	585.875	0	7.109	806.168
14	13	20160420	106.326	55.966	0.232	428.635	0	7.937	599.096
15	14	20160421	159.823	63.253	0	846.448	0	5.897	1075.421
16	15	20160422	551.998	183.536	0.156	1853.404	0.1	29.397	2618.591
17	16	20160425	185.152	59.342	0.387	714.21	0	13.63	972.721
18	17	20160426	208.245	53.98	0	806.936	0	8.42	1077.581

The following Trend Lines have been plotted for different exchanges using Tableau:



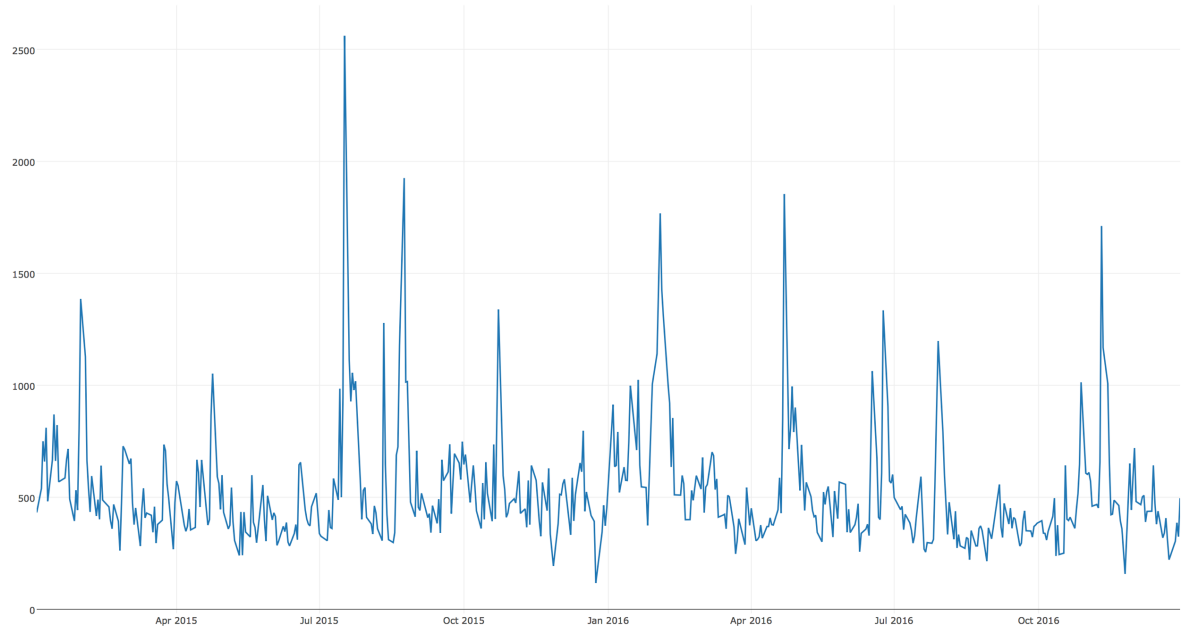
Using Python Plotly function, we have made an Interactive plot for Google Trade Volume for Nasdaq Exchange:

GOOGLNasdaqIplot.py

```
1  import plotly.plotly as py
2  import plotly.graph_objs as go
3
4  import pandas as pd
5
6  df = pd.read_csv("~/GOOGLNasdaq.csv")
7
8  data = [go.Scatter(
9            x=df.Date,
10           y=df['TradeVol'])]
11
12 py.iplot(data)
```

The interactive plot can be visualized using the following URL:

https://plot.ly/~MonicaReddyTirupari/24/

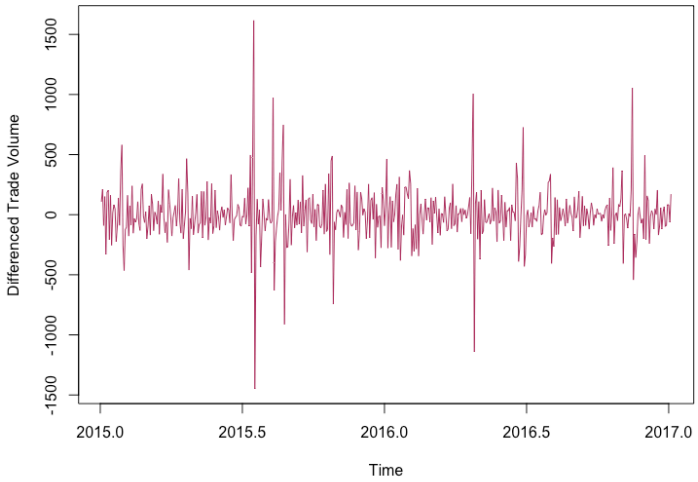


Prediction of Google Trade Volume for January 2017:

We have Used R Programming for the prediction using ARIMA model. We took two years of Google Trade Volume data of Nasdaq Exchange from 2015 – 2016 for the prediction and building the model.

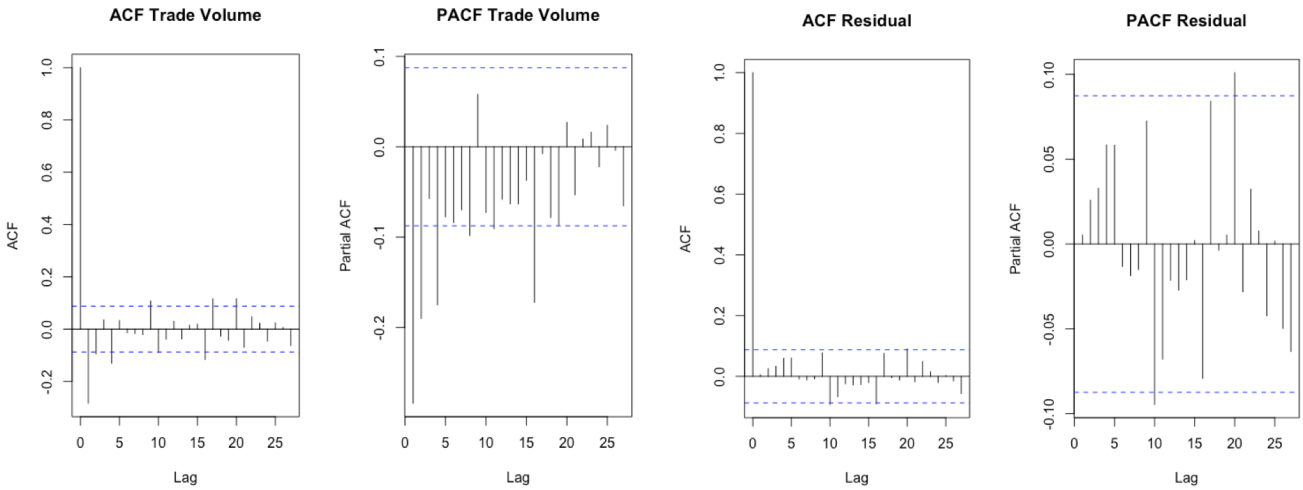
```
install.packages('plotly')
install.packages('forecast')
library(plotly)
library(forecast)
data = read.csv('/Users/monicareddytirupari/GOOGLNasdaq2.csv')

data = ts(data[,2],start = c(2015,1),frequency = 250)
plot(data, xlab='Years', ylab = 'Trade Volume', col = 'darkviolet')

plot(diff(data),ylab='Differenced Trade Volume', col = 'maroon')

plot(log10(data),ylab='Log (Trade Volume)', col = 'darkblue')

plot(diff(log10(data)),ylab='Differenced Log (Trade Volume)', col = 'darkorange')

par(mfrow = c(1,2))
acf(ts(diff(log10(data))),main='ACF Trade Volume')
pacf(ts(diff(log10(data))),main='PACF Trade Volume')

require(forecast)
ARIMAfit = auto.arima(log10(data), approximation=FALSE,trace=FALSE)
summary(ARIMAfit)

par(mfrow = c(1,1))
pred = predict(ARIMAfit, n.ahead = 36)
View(pred)
plot(data,type='l',xlim=c(2017,2017.3),ylim=c(1,1600),xlab = 'Year',ylab = 'Trade Volume', pch = 0.5, col = 'darkgreen')
lines(10^(pred$pred),col='blue')
lines(10^(pred$pred+2*pred$se),col='orange')
lines(10^(pred$pred-2*pred$se),col='orange')

par(mfrow=c(1,2))
acf(ts(ARIMAfit$residuals),main='ACF Residual')
pacf(ts(ARIMAfit$residuals),main='PACF Residual')

```

The differentiated plot obtained is a stationary time series which is essential for prediction.

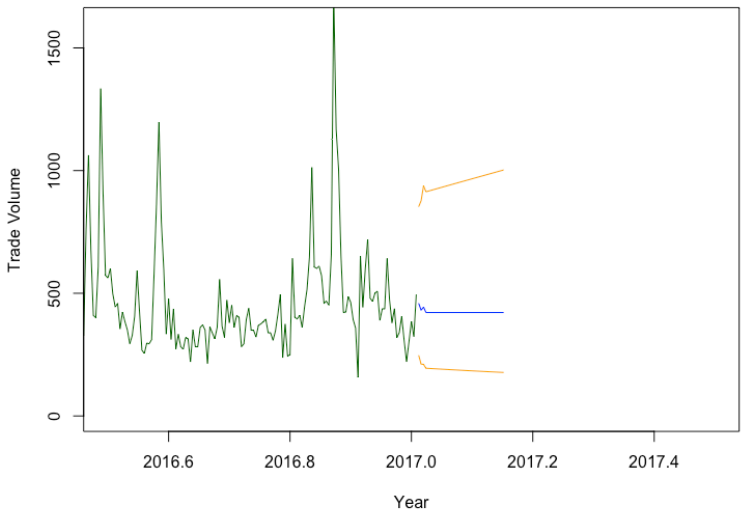


ACF and PACF Charts for Trade Volume:

Auto-correlated function and Partial Auto-correlated function shows how much the data is correlated and fitting the ARIMA model by training the data repeatedly. This gives us the residual charts which are perfectly correlated and gives an accurate prediction output.



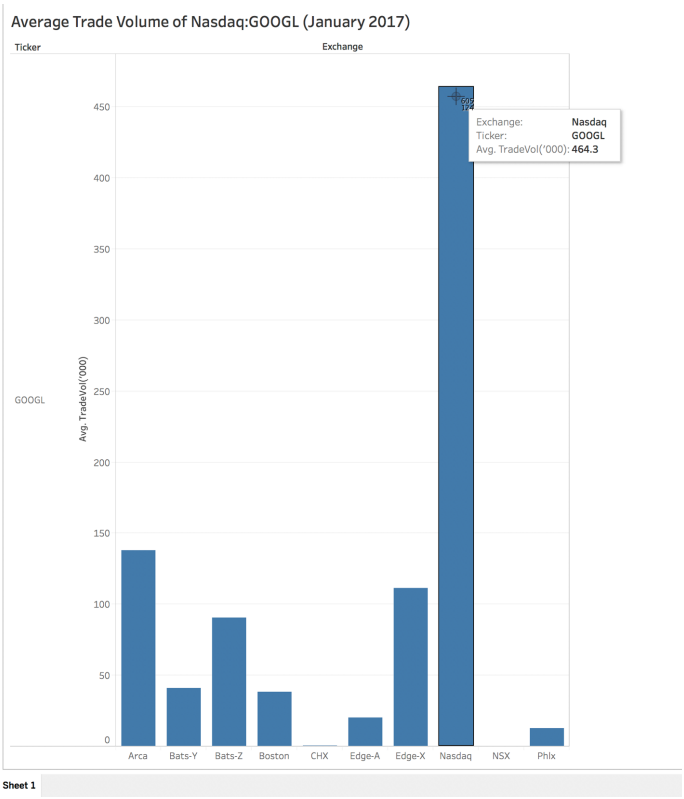
The following graph shows the prediction of Trade Volume for the month of January 2015, where it can be observed that the average trade volume is close to 470,000 Trade Volume



*The above graph is plotted w.r.t Trade Volume is in an order of thousands.

Evaluating the Predicted Output:

With the Actual data, recently available in the USSEC website. We have plotted Trade Volume of Google for January 2017 and it can be observed that the average trade volume for Nasdaq:GOOGL is 464,000 which is very close to our predicted value.



Innovation:

- In Stock Market, the different stock exchanges analyze and predict stock price and market capitalization but here we have analyzed various attributes of a stock and have predicted the future *trade volume* of a stock, which is not generally predicted in the stock market
- We have chosen Nasdaq Exchange for prediction because we want to hunt for high-volume stocks and it has stricter requirements than any other exchange.
- Although Calculating volume is easy. Understanding and analyzing what volume means is more important.

Applicability:

- Calculating volume is simply the total amount of shares traded for the day, which includes both buy and sell orders.
- While volume is only one tool of many, it adds value to your investing decision.
- If the stock that's appreciating on high volume, it's more likely to be a sustainable move.
- If you see a stock that's appreciating on low volume, it could be a dead cat bounce.
- Logically, when more money is moving a stock price, it means there is more demand for that stock.
- If a small amount of money is moving the stock price, the odds of that move being sustainable are lower.
- Low-volume stocks are also called illiquid stocks, buying of it could end up trapped in a pump and dump scheme.
- We have done our prediction using monthly trading volume instead of daily, to have a better idea of which stock offers more liquidity

Reference Link: http://www.investopedia.com/

Code and output files can also be found at: https://github.com/salonibindra2050/Big-Data-Analytics