Market Stock Analysis

Case Study Project

Prepared by:
Name: Suparna Dutta
Roll Number: PGDDSPJULY2020/05
Institute: BSE Institute, Makaut University
Email: suparnadutta@gmail.com

Acknowledgement

I would like to express thanks to Professor Ashok Gupta of BSE Institute, for guiding me in Big data analytics tools which helped me solve real life Big Data Problems and complete this Case Study.

I would like to appreciate Bombay Stock Exchange Institute for providing me with training in Big Data Tools and giving me an opportunity to learn Data Science.

Index

Objectives

Objectives of this project:

- Analyse and clean given data set by filtering out faulty records
- Formulate solution for 7 given questions in Spark, Pig Hive and Map-Reduce.
- Export output to local file system or to HDFS.

Functionality

1. Use the given csv file as input data and implement following transformations:

- **Filter Rows on specified criteria "Symbol equals GEOMETRIC":** Show records where symbol is geometric.
- **Select specific columns from those available: SYMBOL, OPEN, HIGH, LOW and CLOSE which meets above criteria:** Show symbol, open, high, low and close columns where symbol is geometric.
- **Generate count of the number of rows from above result:** Count number of records generated from above query.

2. Calculation of various statistical quantities and decision making:

- **Only lines with value "EQ" in the "series" column should be processed. As the first stage, filter out all the lines that do not fulfil this criteria:** Show records where series is EQ.
- **For every stock, for every year, calculate the following statistical parameters: Minimum, Maximum, Mean and Standard Deviation and store the generated information in properly designated tables:** From above records, find minimum, maximum, mean and standard deviation and store it in table.

3. Select any year for which data is available:

- **For the selected year, create a table that contains data only for those stocks that have an total traded quntity of 3 lakhs or more per day. Print out the first 25 entries of the table and submit:** Extract 25 records from a selected year where Total Trade Quantity >= 300000.
- **From among these, select any 10 stocks from IT ('HCLTECH', 'NIITTECH', 'TATAELXSI','TCS', 'INFY', 'WIPRO', 'DATAMATICS','TECHM','MINDTREE' and 'OFSS') and create a table combining their data:** From above records, select rows where SYMBOL is: HCLTECH or NIITTECH or TATAELXSI or TCS or INFY or WIPRO or DATAMATICS or TECHM or MINDTREE or OFSS and store it in a table.
- **Find out the Pearsons Correlation Coeffecient for every pair of stocks you have selected. Final output should be in decreasing order of the coefficient:** From above records, calculate Pearsons Correlation Coeffecient for every pair of symbol.

Hardware & Software Requirements

Software requirements for this Case Study:

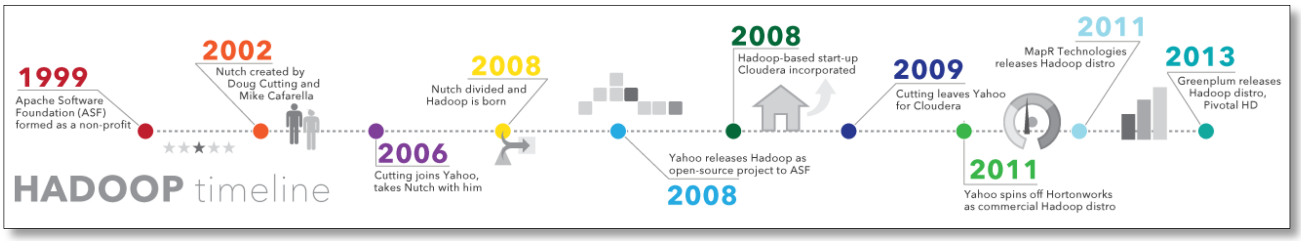
- Oracle Virtual Box
- 64 bit Ubuntu installed in Oracle Virtual Box
- Download and install Java, Hadoop, Hive, Pig and Spark

Minimum Hardware requirements for this Case Study:

- RAM: 8 GB
- Hard Drive Space: 50 GB
- Processor: i5 or equivalent

Why Hadoop

History of Hadoop:


1999
Apache Software Foundation (ASF) formed as a non-profit
2002
Nutch created by Doug Cutting and Mike Cafarella
2006
Cutting joins Yahoo, takes Nutch with him
2008
Nutch divided and Hadoop is born
2008
Yahoo releases Hadoop as open-source project to ASF
2008
Hadoop-based start-up Cloudera incorporated
2009
Cutting leaves Yahoo for Cloudera
2011
Yahoo spins off Hortonworks as commercial Hadoop distro
2011
MapR Technologies releases Hadoop distro
2013
Greenplum releases Hadoop distro, Pivotal HD
HADOOP timeline

Why is Hadoop important?

- **Ability to store and process huge amounts of any kind of data, quickly:** With data volumes and varieties constantly increasing, especially from social media and the Internet of Things (IoT), that's a key consideration.
- **Computing power:** Hadoop's distributed computing model processes big data fast. The more computing nodes you use, the more processing power you have.
- **Fault tolerance:** Data and application processing are protected against hardware failure. If a node goes down, jobs are automatically redirected to other nodes to make sure the distributed computing does not fail. Multiple copies of all data are stored automatically.
- **Flexibility:** Unlike traditional relational databases, you don't have to preprocess data before storing it. You can store as much data as you want and decide how to use it later. That includes unstructured data like text, images and videos.
- **Low cost:** The open-source framework is free and uses commodity hardware to store large quantities of data.
- **Scalability:** You can easily grow your system to handle more data simply by adding nodes. Little administration is required.

Big Data Tools

Pig vs Hive

Pig	Hive
Pig operates on the client side of a cluster.	Hive operates on the server side of a cluster.
Pig uses pig-latin language.	Hive uses HiveQL language.
Pig is a Procedural Data Flow Language.	Hive is a Declarative SQLish Language.
It is used to handle structured and semi-structured data.	It is mainly used to handle structured data.
It is used for programming.	It is used for creating reports.
Pig scripts end with .pig extension.	In HIve, all extensions are supported.
It does not support partitioning.	It supports partitioning.
It loads data quickly.	It loads data slowly.
It does not support JDBC.	It supports JDBC.
It does not support ODBC.	It supports ODBC.
Pig does not have a dedicated metadata database.	Hive makes use of the exact variation of dedicated SQL-DDL language by defining tables beforehand.
It supports Avro file format.	It does not support Avro file format.
Pig is suitable for complex and nested data structures.	Hive is suitable for batch-processing OLAP systems.
Pig does not support schema to store data.	Hive supports schema for data insertion in tables.

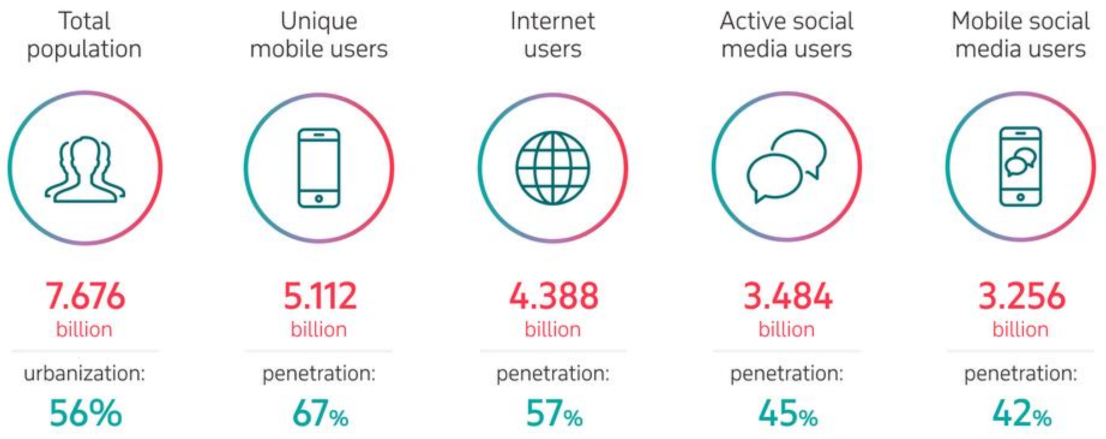
Map-Reduce vs Spark

Factors	Hadoop MapReduce	Apache Spark
Core Definition	MapReduce is a programming model that is implemented in processing huge amounts of data. MapReduce has been developed using Java MapReduce Programs work in two phases: • The Map Phase • The Reduce Phase The entire MapReduce process goes through the following 4 phases: • **Splitting**: The input	Apache Spark is an open-source, distributed processing system which is used for Big Data. Spark is an engine for large scale data processing. Spark has been developed using Scala. The main components of Apache Spark are as follows: • **Apache Spark Core**: It is the underlying general execution engine over which all other functionality is built. It provides in-memory computing and dataset references in external storage systems.

	is divided into a fixed size splits called input-splits. An input split is consumed by a single map. • **Mapping**: Here, data in each map is passed into a mapping function to produce output values. • **Shuffling**: This phase consumes the output of the mapping phase and the relevant records are consolidated. • **Reducing**: In this phase the relevant records are aggregated and a single output value is returned. This phase summarizes the complete dataset.	• **Spark SQL**: It is the module which provides information about the data structure and the computation being performed • **Spark Streaming**: Allows processing of real-time data. This data is then processed using complex algorithms and pushed out to file systems, databases and live systems. • **MLlib[Machine Learning]**: It is a library that contains a wide array of machine learning algorithms and tools for constructing, evaluating and tuning ML pipelines. • **GraphX**: It comes with a library to manipulate graph databases and perform computations. It unifies ETL process, exploratory process and iterative graph computation within a single system.
Processing Speed	MapReduce reads and writes data from the disk. Though it is faster than traditional systems, it is substantially slower than Spark.	It runs on RAM, stores intermediate data in-memory reducing the number of read/write cycles to the disk. Hence it is faster than the classical MapReduce.
Memory Usage	Does not support caching of Data.	Enhances the system performance by caching the data in-memory.
Coding	MapReduce requires handling low-level APIs due to which developers need to code each and every operation which makes it very difficult to work with.	Spark is easy to use and its Resilient Distributed Dataset helps to process data with its high-level operators. It provides rich APIs in Java, Scala, Python and R.
Latency Latency means Delay. It is the time the CPU has	MapReduce has a high-latency computing framework.	Spark provides a low latency computing.

to wait to get a response after it makes a request to the RAM.		
Recovery From Failure	MapReduce is highly faulted tolerant and is resilient to system faults and failures. Here there is no need to restart the application from scratch in case of failure.	Spark is also fault tolerant. Resilient Distributed Dataset [RDDs] allow for recovery of partitions on failed nodes. It also supports recovery by checkpointing to reduce the dependencies of an RDD. Hence, here too there is no need to restart the application from scratch in case of failure.
Scheduler	MapReduce is dependant on external job scheduler like Oozie to schedule its complex flows.	Due to in-memory computation Spark acts like its own flow scheduler.
Security	MapReduce is comparatively more secure because of Kerberos. It also supports Access Control Lists (ACLs) which are traditional file permission model.	Spark supports only one authentication which is the shared secret password authentication.
Cost	MapReduce is a cheaper option in terms of cost.	Spark is costlier due to its in-memory processing power and RAM requirement.
Function	MapReduce is a Data Processing Engine.	Spark is a Data Analytics Engine hence a choice for Data Scientist.
Framework	It is an open-source framework for writing data into HDFS and processing structured and unstructured data present in HDFS.	Spark is an independent real-time processing engine that can be installed in any Distributed File System.
Programming Language Supported	Java, C, C++, Ruby, Groovy, Perl, Python	Scala, Java, Python, R, SQL
Hardware Requirement	MapReduce can be run on commodity hardware.	Apache Spark requires mid to high-level hardware configuration to run efficiently.
	Hadoop requires a machine learning tool, one of which is Apache Mahout.	Spark has its own set of Machine Learning i.e. MLlib.
Redundancy Check	MapReduce does not support this feature.	Spark processes every record exactly once and hence eliminates duplication

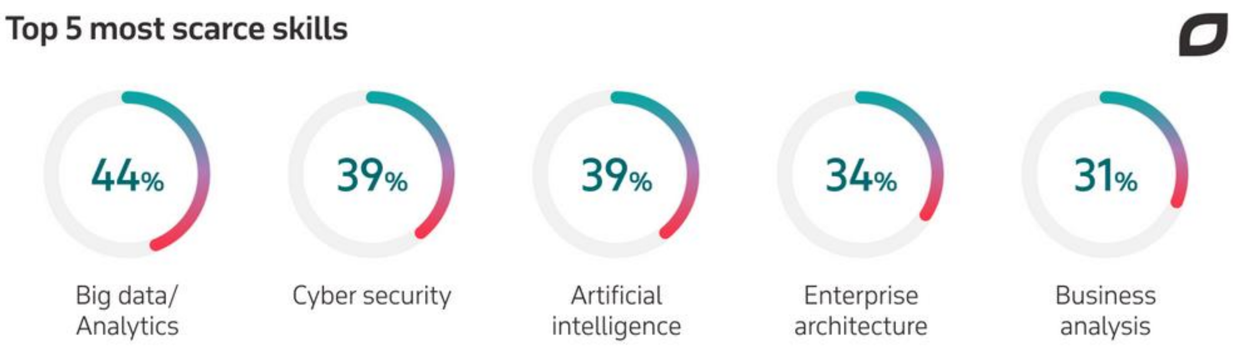
Future Scope of Big Data



Billions of connected devices and embedded systems that create, collect and share a wealth of IoT data analytics every day, all over the world.

As enterprises gain the opportunity to store and analyze huge volumes of data, they will get to create and manage 60% of big data in the near future. However, individual consumers have a significant role to play in data growth, too. In the same report, IDC also estimates that 6 billion users, or 75% of the world's population, will be interacting with online data every day by 2025. In other terms, each connected user will be having at least one data interaction every 18 seconds.

Such large datasets are challenging to work with in terms of their storage and processing. Until recently, big data processing challenges were solved by open-source ecosystems, such as Hadoop and NoSQL. However, open-source technologies require manual configuration and troubleshooting, which can be rather complicated for most companies. In search for more elasticity, businesses started to migrate big data to the cloud.



No wonder data scientists are among the top fastest-growing jobs today, along with machine learning engineers and big data engineers. Big data is useless without analysis, and data scientists are those professionals who collect and analyze data with the help of analytics and reporting tools, turning it into actionable insights.

Fast data and actionable data will come to the forefront

Yet another prediction about the big data future is related to the rise of what is called 'fast data' and 'actionable data'.

Unlike big data, typically relying on Hadoop and NoSQL databases to analyze information in the batch mode, fast data allows for processing in real-time streams. Because of this stream processing, data can be analyzed promptly, within as little as just one millisecond. This brings more value to organizations that can make business decisions and take actions immediately when data arrives.

Fast data has also spoilt users, making them addicted to real-time interactions. As businesses are getting more digitized, which drives better customer experience, consumers expect to access data on the go. What's more, they want it personalized. In the research cited above, IDC predicts that nearly 30% of the global data will be real-time by 2025.

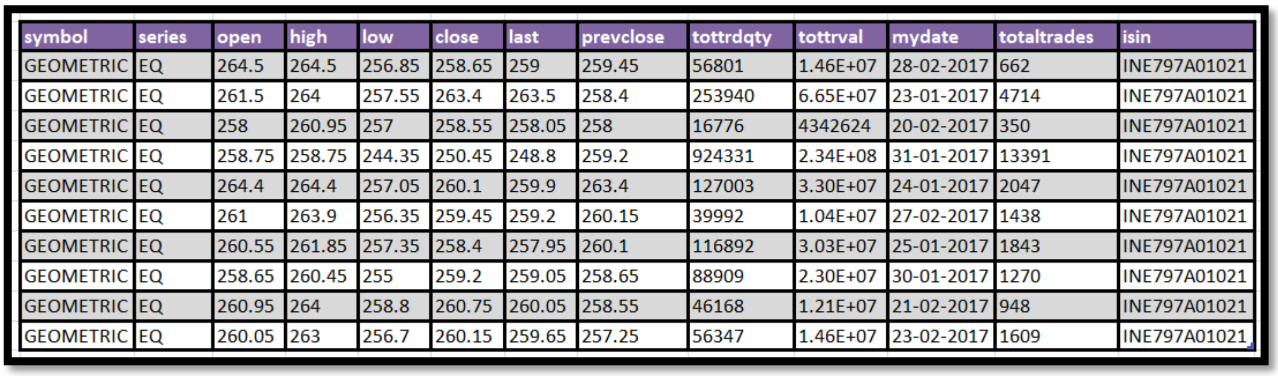
Actionable data is the missing link between big data and business value. As it was mentioned earlier, big data in itself is worthless without analysis since it is too complex, multi-structured, and voluminous. By processing data with the help of analytical platforms, organizations can make information accurate, standardized, and actionable. These insights help companies make more informed business decisions, improve their operations, and design more big data use cases.

SPARK

1. Use the given csv file as input data and implement following transformations:

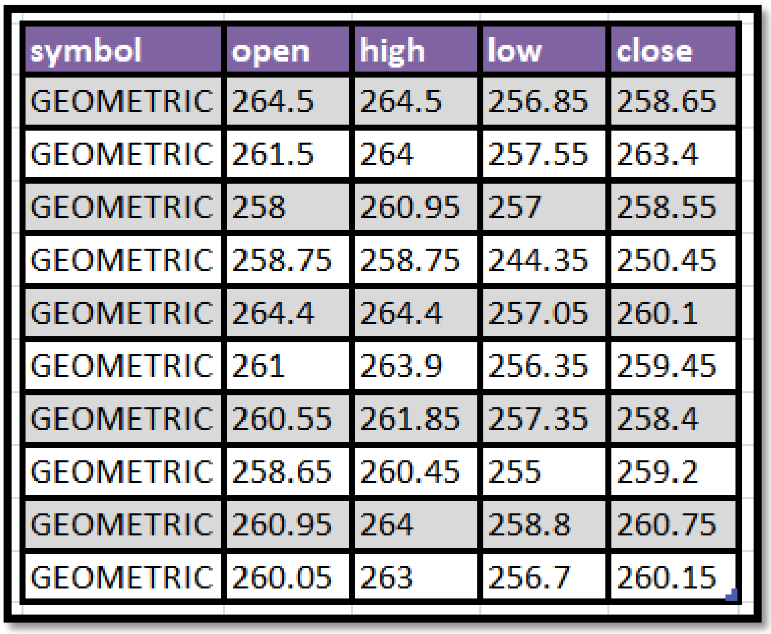
- **Filter Rows on specified criteria "Symbol equals GEOMETRIC":** Show records where symbol is geometric.
- **Select specific columns from those available: SYMBOL, OPEN, HIGH, LOW and CLOSE which meets above criteria:** Show symbol, open, high, low and close columns where symbol is geometric.
- **Generate count of the number of rows from above result:** Count number of records generated from above query.

```
val input = sc.textFile("/home/suparna/Desktop/stock_market_case_study/in
put/stock_market.csv").map(line => line.split(","))

val stock_filter = input.filter(x => x(0).toString == "GEOMETRIC")

val output_1_1 = stock_filter.map(x => {(x(0).toString, x(1).toString, x(
2).toDouble, x(3).toDouble, x(4).toDouble, x(5).toDouble, x(6).toDouble,
x(7).toDouble, x(8).toDouble, x(9).toDouble, x(10).toString, x(11).toLong
, x(12).toString)})

output_1_1.map(line => line.productIterator.mkString("\t")).repartition(1
).saveAsTextFile("/home/suparna/Desktop/stock_market_case_study/spark/out
put_1_1")
```



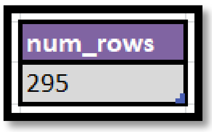
symbol	series	open	high	low	close	last	prevclose	tottrdqty	tottrval	mydate	totaltrades	isin
GEOMETRIC	EQ	264.5	264.5	256.85	258.65	259	259.45	56801	1.46E+07	28-02-2017	662	INE797A01021
GEOMETRIC	EQ	261.5	264	257.55	263.4	263.5	258.4	253940	6.65E+07	23-01-2017	4714	INE797A01021
GEOMETRIC	EQ	258	260.95	257	258.55	258.05	258	16776	4342624	20-02-2017	350	INE797A01021
GEOMETRIC	EQ	258.75	258.75	244.35	250.45	248.8	259.2	924331	2.34E+08	31-01-2017	13391	INE797A01021
GEOMETRIC	EQ	264.4	264.4	257.05	260.1	259.9	263.4	127003	3.30E+07	24-01-2017	2047	INE797A01021
GEOMETRIC	EQ	261	263.9	256.35	259.45	259.2	260.15	39992	1.04E+07	27-02-2017	1438	INE797A01021
GEOMETRIC	EQ	260.55	261.85	257.35	258.4	257.95	260.1	116892	3.03E+07	25-01-2017	1843	INE797A01021
GEOMETRIC	EQ	258.65	260.45	255	259.2	259.05	258.65	88909	2.30E+07	30-01-2017	1270	INE797A01021
GEOMETRIC	EQ	260.95	264	258.8	260.75	260.05	258.55	46168	1.21E+07	21-02-2017	948	INE797A01021
GEOMETRIC	EQ	260.05	263	256.7	260.15	259.65	257.25	56347	1.46E+07	23-02-2017	1609	INE797A01021

```
val stock_filter = input.filter(x => x(0).toString == "GEOMETRIC")

val output_1_2 = stock_filter.map(x => {(x(0).toString, x(2).toDouble, x(
3).toDouble, x(4).toDouble, x(5).toDouble)})

output_1_2.map(line => line.productIterator.mkString("\t")).repartition(1
).saveAsTextFile("/home/suparna/Desktop/stock_market_case_study/spark/out
put_1_2")
```



symbol	open	high	low	close
GEOMETRIC	264.5	264.5	256.85	258.65
GEOMETRIC	261.5	264	257.55	263.4
GEOMETRIC	258	260.95	257	258.55
GEOMETRIC	258.75	258.75	244.35	250.45
GEOMETRIC	264.4	264.4	257.05	260.1
GEOMETRIC	261	263.9	256.35	259.45
GEOMETRIC	260.55	261.85	257.35	258.4
GEOMETRIC	258.65	260.45	255	259.2
GEOMETRIC	260.95	264	258.8	260.75
GEOMETRIC	260.05	263	256.7	260.15

```
val output_1_3 = sc.parallelize(Seq(output_1_2.count(),""))

output_1_3.repartition(1).saveAsTextFile("/home/suparna/Desktop/stock_mar
ket_case_study/spark/output_1_3")
```

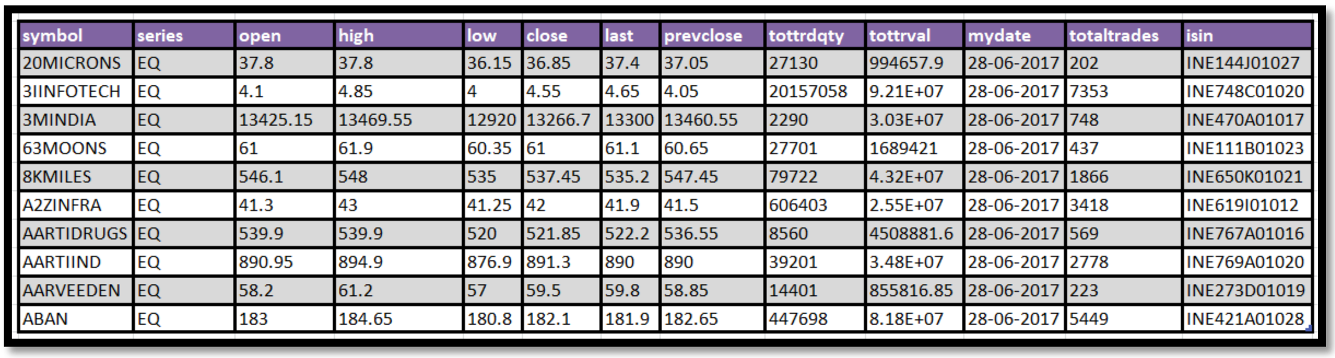


num_rows
295

2. Calculation of various statistical quantities and decision making:

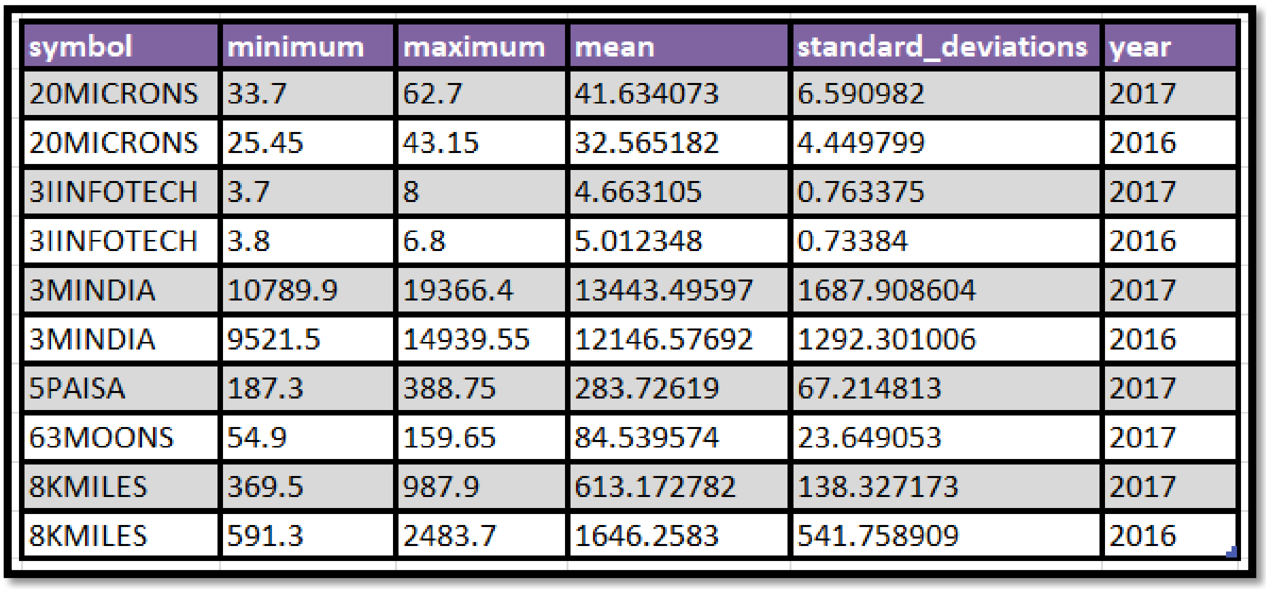
- **Only lines with value "EQ" in the "series" column should be processed. As the first stage, filter out all the lines that do not fulfil this criteria:** Show records where series is EQ.
- **For every stock, for every year, calculate the following statistical parameters: Minimum, Maximum, Mean and Standard Deviation and store the generated information in properly designated tables:** From above records, find minimum, maximum, mean and standard deviation and store it in table.

```
val input = sc.textFile("/home/suparna/Desktop/stock_market_case_study/in
put/stock_market.csv").map(line => line.split(","))

val stock_filter = input.filter(x => x(1).toString == "EQ")

val output_2_1 = stock_filter.map(x => {(x(0).toString, x(1).toString, x(
2).toDouble, x(3).toDouble, x(4).toDouble, x(5).toDouble, x(6).toDouble,
x(7).toDouble, x(8).toDouble, x(9).toDouble, x(10).toString, x(11).toLong
, x(12).toString)})
output_2_1.map(line => line.productIterator.mkString("\t")).repartition(1
).saveAsTextFile("/home/suparna/Desktop/stock_market_case_study/spark/out
put_2_1")
```



symbol	series	open	high	low	close	last	prevclose	tottrdqty	tottrval	mydate	totaltrades	isin
20MICRONS	EQ	37.8	37.8	36.15	36.85	37.4	37.05	27130	994657.9	28-06-2017	202	INE144J01027
3IINFOTECH	EQ	4.1	4.85	4	4.55	4.65	4.05	20157058	9.21E+07	28-06-2017	7353	INE748C01020
3MINDIA	EQ	13425.15	13469.55	12920	13266.7	13300	13460.55	2290	3.03E+07	28-06-2017	748	INE470A01017
63MOONS	EQ	61	61.9	60.35	61	61.1	60.65	27701	1689421	28-06-2017	437	INE111B01023
8KMILES	EQ	546.1	548	535	537.45	535.2	547.45	79722	4.32E+07	28-06-2017	1866	INE650K01021
A2ZINFRA	EQ	41.3	43	41.25	42	41.9	41.5	606403	2.55E+07	28-06-2017	3418	INE619I01012
AARTIDRUGS	EQ	539.9	539.9	520	521.85	522.2	536.55	8560	4508881.6	28-06-2017	569	INE767A01016
AARTIIND	EQ	890.95	894.9	876.9	891.3	890	890	39201	3.48E+07	28-06-2017	2778	INE769A01020
AARVEEDEN	EQ	58.2	61.2	57	59.5	59.8	58.85	14401	855816.85	28-06-2017	223	INE273D01019
ABAN	EQ	183	184.65	180.8	182.1	181.9	182.65	447698	8.18E+07	28-06-2017	5449	INE421A01028

```
val input_map = output_2_1.map(x => {(x._1.toString +"_"+ x._11.toString.
split("-")(0),x._6)})
val min_max_mean_std = input_map.groupByKey().mapValues(sq => (sq.min, sq
.max, sq.sum/sq.size, org.apache.spark.util.StatCounter(sq).stdev))
val output_2_2 = min_max_mean_std.map(x => {(x._1.split("_")(0), x._2._1,
 x._2._2, BigDecimal(x._2._3).setScale(6,BigDecimal.RoundingMode.HALF_UP)
, BigDecimal(x._2._4).setScale(6, BigDecimal.RoundingMode.HALF_UP), x._1.
split("_")(1))}).sortBy(_._6,false).sortBy(_._1)
output_2_2.map(line => line.productIterator.mkString("\t")).repartition(1
).saveAsTextFile("/home/suparna/Desktop/stock_market_case_study/spark/out
put_2_2")
```

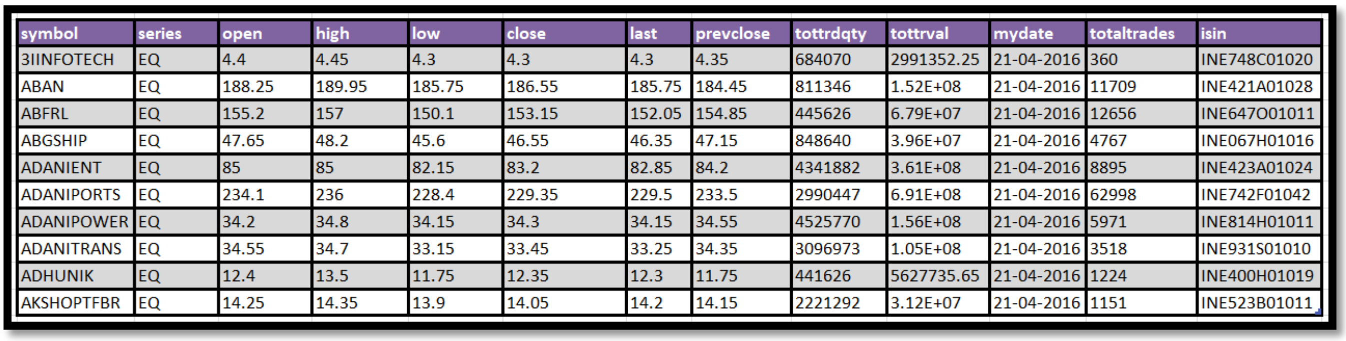


symbol	minimum	maximum	mean	standard_deviations	year
20MICRONS	33.7	62.7	41.634073	6.590982	2017
20MICRONS	25.45	43.15	32.565182	4.449799	2016
3IINFOTECH	3.7	8	4.663105	0.763375	2017
3IINFOTECH	3.8	6.8	5.012348	0.73384	2016
3MINDIA	10789.9	19366.4	13443.49597	1687.908604	2017
3MINDIA	9521.5	14939.55	12146.57692	1292.301006	2016
5PAISA	187.3	388.75	283.72619	67.214813	2017
63MOONS	54.9	159.65	84.539574	23.649053	2017
8KMILES	369.5	987.9	613.172782	138.327173	2017
8KMILES	591.3	2483.7	1646.2583	541.758909	2016

3. Select any year for which data is available:

- **For the selected year, create a table that contains data only for those stocks that have an total traded quntity of 3 lakhs or more per day. Print out the first 25 entries of the table and submit:** Extract 25 records from a selected year where Total Trade Quantity >= 300000.
- **From among these, select any 10 stocks from IT ('HCLTECH', 'NIITTECH', 'TATAELXSI','TCS', 'INFY', 'WIPRO', 'DATAMATICS','TECHM','MINDTREE' and 'OFSS') and create a table combining their data:** From above records, select rows where SYMBOL is: HCLTECH or NIITTECH or TATAELXSI or TCS or INFY or WIPRO or DATAMATICS or TECHM or MINDTREE or OFSS and store it in a table.
- **Find out the Pearsons Correlation Coeffecient for every pair of stocks you have selected. Final output should be in decreasing order of the coefficient:** From above records, calculate Pearsons Correlation Coeffecient for every pair of symbol.

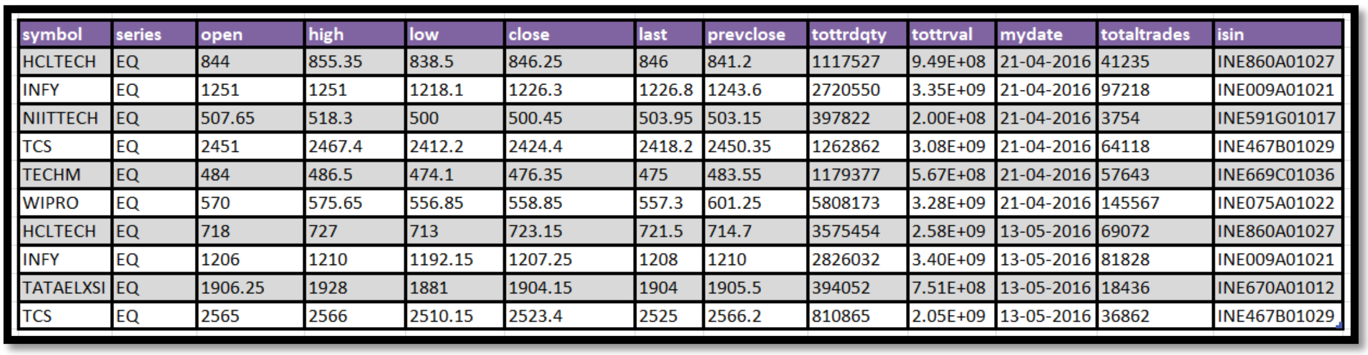
```
val stock_filter = input.filter(x => x(8).toLong >= 300000 && x(10).toStr
ing.split("-")(0) == "2016")

val stock_map = stock_filter.map(x => {(x(0).toString, x(1).toString, x(2
).toDouble, x(3).toDouble, x(4).toDouble, x(5).toDouble, x(6).toDouble, x
(7).toDouble, x(8).toDouble, x(9).toDouble, x(10).toString, x(11).toLong,
 x(12).toString)})

val output_3_1 = sc.parallelize(stock_map.take(25))
output_3_1.map(line => line.productIterator.mkString("\t")).repartition(1
).saveAsTextFile("/home/suparna/Desktop/stock_market_case_study/spark/out
put_3_1")
```



symbol	series	open	high	low	close	last	prevclose	tottrdqty	tottrval	mydate	totaltrades	isin
3IINFOTECH	EQ	4.4	4.45	4.3	4.3	4.3	4.35	684070	2991352.25	21-04-2016	360	INE748C01020
ABAN	EQ	188.25	189.95	185.75	186.55	185.75	184.45	811346	1.52E+08	21-04-2016	11709	INE421A01028
ABFRL	EQ	155.2	157	150.1	153.15	152.05	154.85	445626	6.79E+07	21-04-2016	12656	INE647O01011
ABGSHIP	EQ	47.65	48.2	45.6	46.55	46.35	47.15	848640	3.96E+07	21-04-2016	4767	INE067H01016
ADANIENT	EQ	85	85	82.15	83.2	82.85	84.2	4341882	3.61E+08	21-04-2016	8895	INE423A01024
ADANIPORTS	EQ	234.1	236	228.4	229.35	229.5	233.5	2990447	6.91E+08	21-04-2016	62998	INE742F01042
ADANIPOWER	EQ	34.2	34.8	34.15	34.3	34.15	34.55	4525770	1.56E+08	21-04-2016	5971	INE814H01011
ADANITRANS	EQ	34.55	34.7	33.15	33.45	33.25	34.35	3096973	1.05E+08	21-04-2016	3518	INE931S01010
ADHUNIK	EQ	12.4	13.5	11.75	12.35	12.3	11.75	441626	5627735.65	21-04-2016	1224	INE400H01019
AKSHOPTFBR	EQ	14.25	14.35	13.9	14.05	14.2	14.15	2221292	3.12E+07	21-04-2016	1151	INE523B01011

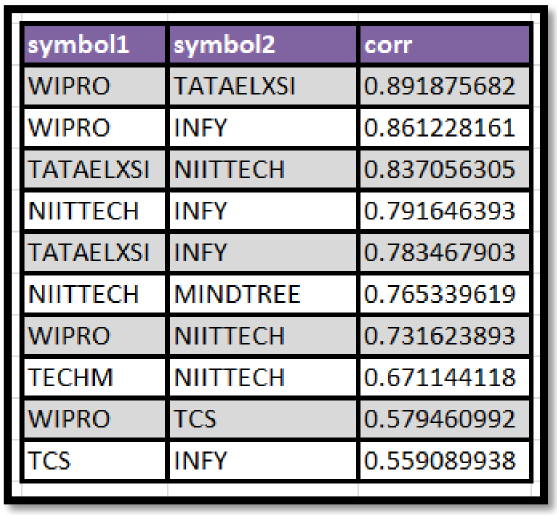
```
val output_3_2 = stock_map.filter(x => x._1 == "HCLTECH" || x._1 == "NIIT
TECH" || x._1 == "TATAELXSI" || x._1 == "TCS" || x._1 == "INFY" || x._1 =
= "WIPRO" || x._1 == "DATAMATICS" || x._1 == "TECHM" || x._1 == "MINDTREE
" || x._1 == "OFSS")
```

```
output_3_2.map(line => line.productIterator.mkString("\t")).repartition(1
).saveAsTextFile("/home/suparna/Desktop/stock_market_case_study/spark/out
put_3_2")
```



symbol	series	open	high	low	close	last	prevclose	tottrdqty	tottrval	mydate	totaltrades	isin
HCLTECH	EQ	844	855.35	838.5	846.25	846	841.2	1117527	9.49E+08	21-04-2016	41235	INE860A01027
INFY	EQ	1251	1251	1218.1	1226.3	1226.8	1243.6	2720550	3.35E+09	21-04-2016	97218	INE009A01021
NIITTECH	EQ	507.65	518.3	500	500.45	503.95	503.15	397822	2.00E+08	21-04-2016	3754	INE591G01017
TCS	EQ	2451	2467.4	2412.2	2424.4	2418.2	2450.35	1262862	3.08E+09	21-04-2016	64118	INE467B01029
TECHM	EQ	484	486.5	474.1	476.35	475	483.55	1179377	5.67E+08	21-04-2016	57643	INE669C01036
WIPRO	EQ	570	575.65	556.85	558.85	557.3	601.25	5808173	3.28E+09	21-04-2016	145567	INE075A01022
HCLTECH	EQ	718	727	713	723.15	721.5	714.7	3575454	2.58E+09	13-05-2016	69072	INE860A01027
INFY	EQ	1206	1210	1192.15	1207.25	1208	1210	2826032	3.40E+09	13-05-2016	81828	INE009A01021
TATAELXSI	EQ	1906.25	1928	1881	1904.15	1904	1905.5	394052	7.51E+08	13-05-2016	18436	INE670A01012
TCS	EQ	2565	2566	2510.15	2523.4	2525	2566.2	810865	2.05E+09	13-05-2016	36862	INE467B01029

```
implicit def calc_avg(x:Iterable[Double]) = new {
    def average:Double = x.sum / x.size
}
implicit def calc_std_dev(x:Iterable[Double]) = new {
    def standard_deviation:Double = org.apache.spark.util.StatCounter(x).
stdev
}

val stock_IT1 = output_3_2.map(x => {(x._11, x._1, x._6)}).map(x => (x._1
, {(x._2, x._3)}))

val stock_self_join=stock_IT1.join(stock_IT1)

val s1_ge_s2 = stock_self_join.filter(x => x._2._1._1 > x._2._2._1).sortB
y(_._2._1._1).sortBy(_._2._2._1).sortBy(_._1)

val map1 = s1_ge_s2.map(x => x._2)

val map2 = map1.map(x => ((x._1._1, x._2._1), (x._1._2, x._2._2, x._1._2
* x._2._2)))

val group_symb1_symb2 = map2.groupByKey()

val pearsoncoefficient = group_symb1_symb2.map(x => {((x._1._1, x._1._2),
 ((x._2.map(y=>y._3).average -
 x._2.map(y=>y._1).average * x._2.map(y=>y._2).average)/(x._2.map(y=>y._1
).standard_deviation * x._2.map(y=>y._2).standard_deviation)))})

val output_3_3 = pearsoncoefficient.map(x => (x._1._1, x._1._2, x._2.toDo
uble)).sortBy(_._3, false)

output_3_3.map(line => line.productIterator.mkString("\t")).repartition(1
).saveAsTextFile("/home/suparna/Desktop/stock_market_case_study/spark/out
put_3_3")
```



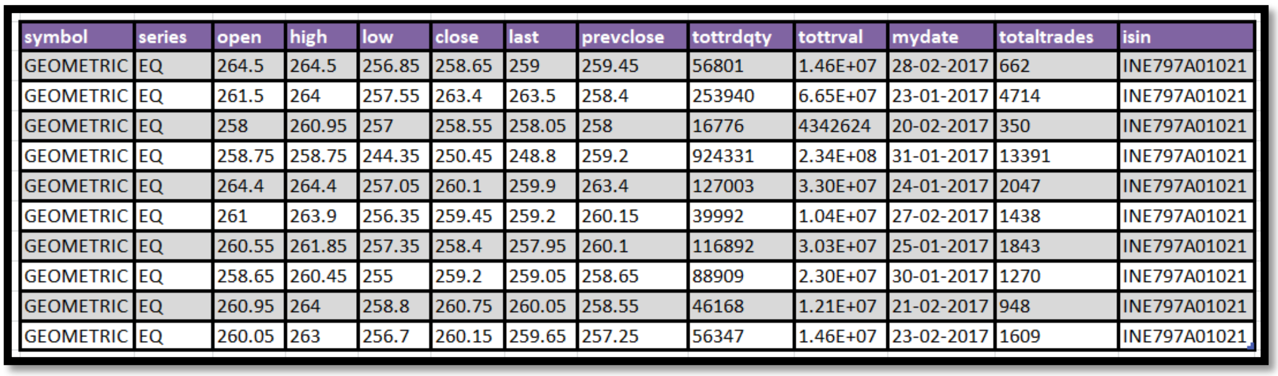
symbol1	symbol2	corr
WIPRO	TATAELXSI	0.891875682
WIPRO	INFY	0.861228161
TATAELXSI	NIITTECH	0.837056305
NIITTECH	INFY	0.791646393
TATAELXSI	INFY	0.783467903
NIITTECH	MINDTREE	0.765339619
WIPRO	NIITTECH	0.731623893
TECHM	NIITTECH	0.671144118
WIPRO	TCS	0.579460992
TCS	INFY	0.559089938

PIG

1. Use the given csv file as input data and implement following transformations:

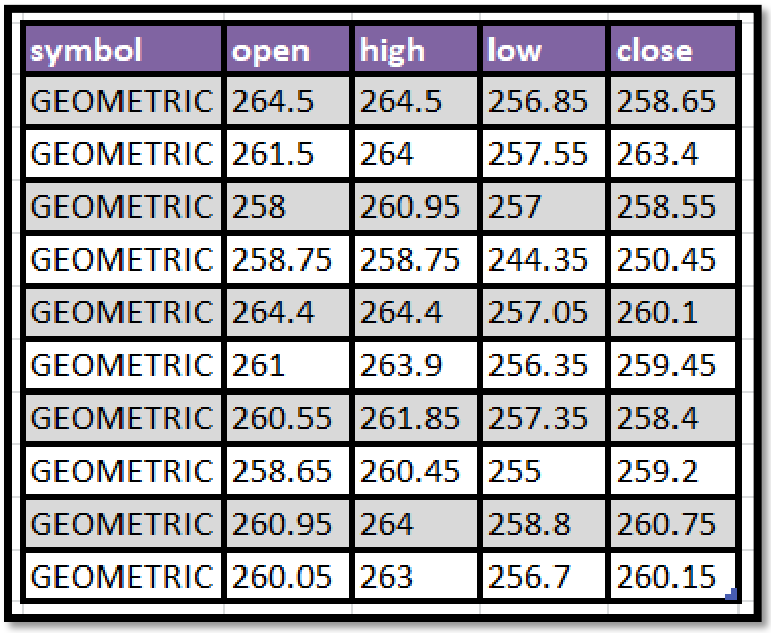
- **Filter Rows on specified criteria "Symbol equals GEOMETRIC":** Show records where symbol is geometric.
- **Select specific columns from those available: SYMBOL, OPEN, HIGH, LOW and CLOSE which meets above criteria:** Show symbol, open, high, low and close columns where symbol is geometric.
- **Generate count of the number of rows from above result:** Count number of records generated from above query.

```
stock_market_data = load '/home/suparna/Desktop/stock_market_case_study/i
nput/stock_market.csv' using org.apache.pig.piggybank.storage.CSVLoader()
 as (symbol:chararray, series:chararray, open:double, high:double, low:do
uble, close:double, last:double, prevclose:double, tottrdqty:double, tott
rdval:double, timestamp:chararray, totaltrades:int, isin:chararray);


stock_fltr = filter stock_market_data by LOWER(symbol) == 'geometric';

store stock_fltr into '/home/suparna/Desktop/stock_market_case_study/pig/
output/1_1' using org.apache.pig.piggybank.storage.CSVExcelStorage('\t',
'NO_MULTILINE', 'UNIX', 'WRITE_OUTPUT_HEADER') ;
```



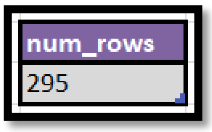
symbol	series	open	high	low	close	last	prevclose	tottrdqty	tottrval	mydate	totaltrades	isin
GEOMETRIC	EQ	264.5	264.5	256.85	258.65	259	259.45	56801	1.46E+07	28-02-2017	662	INE797A01021
GEOMETRIC	EQ	261.5	264	257.55	263.4	263.5	258.4	253940	6.65E+07	23-01-2017	4714	INE797A01021
GEOMETRIC	EQ	258	260.95	257	258.55	258.05	258	16776	4342624	20-02-2017	350	INE797A01021
GEOMETRIC	EQ	258.75	258.75	244.35	250.45	248.8	259.2	924331	2.34E+08	31-01-2017	13391	INE797A01021
GEOMETRIC	EQ	264.4	264.4	257.05	260.1	259.9	263.4	127003	3.30E+07	24-01-2017	2047	INE797A01021
GEOMETRIC	EQ	261	263.9	256.35	259.45	259.2	260.15	39992	1.04E+07	27-02-2017	1438	INE797A01021
GEOMETRIC	EQ	260.55	261.85	257.35	258.4	257.95	260.1	116892	3.03E+07	25-01-2017	1843	INE797A01021
GEOMETRIC	EQ	258.65	260.45	255	259.2	259.05	258.65	88909	2.30E+07	30-01-2017	1270	INE797A01021
GEOMETRIC	EQ	260.95	264	258.8	260.75	260.05	258.55	46168	1.21E+07	21-02-2017	948	INE797A01021
GEOMETRIC	EQ	260.05	263	256.7	260.15	259.65	257.25	56347	1.46E+07	23-02-2017	1609	INE797A01021

```
select_data = foreach stock_fltr generate symbol, open, high, low, close;

store select_data into '/home/suparna/Desktop/stock_market_case_study/pig
/output/1_2/' using org.apache.pig.piggybank.storage.CSVExcelStorage('\t'
, 'NO_MULTILINE', 'UNIX', 'WRITE_OUTPUT_HEADER');
```



symbol	open	high	low	close
GEOMETRIC	264.5	264.5	256.85	258.65
GEOMETRIC	261.5	264	257.55	263.4
GEOMETRIC	258	260.95	257	258.55
GEOMETRIC	258.75	258.75	244.35	250.45
GEOMETRIC	264.4	264.4	257.05	260.1
GEOMETRIC	261	263.9	256.35	259.45
GEOMETRIC	260.55	261.85	257.35	258.4
GEOMETRIC	258.65	260.45	255	259.2
GEOMETRIC	260.95	264	258.8	260.75
GEOMETRIC	260.05	263	256.7	260.15

```
grp_cols = group select_data all;

stock_count= foreach grp_cols generate COUNT(select_data.symbol) as (coun
t_geo:long);

store stock_count into '/home/suparna/Desktop/stock_market_case_study/pig
/output/1_3/' using org.apache.pig.piggybank.storage.CSVExcelStorage('\t'
, 'NO_MULTILINE', 'UNIX', 'WRITE_OUTPUT_HEADER');
```



num_rows
295

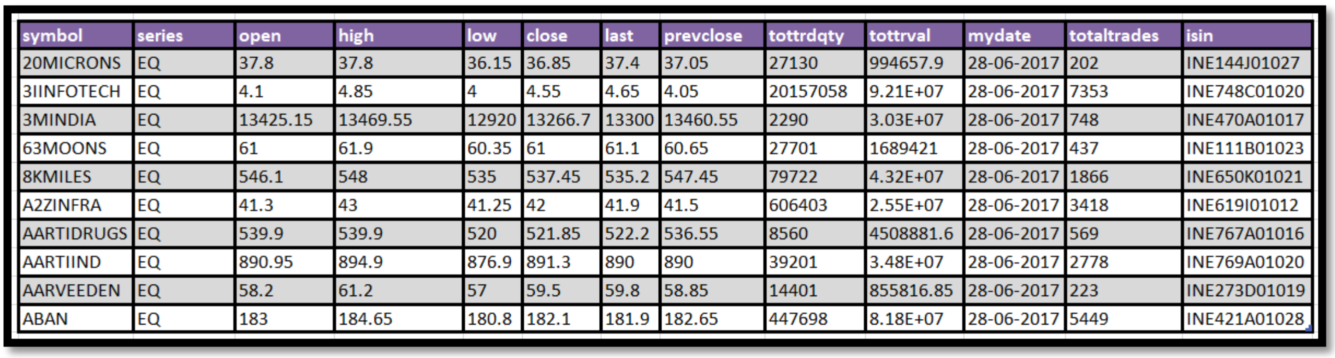
2. Calculation of various statistical quantities and decision making:

- **Only lines with value "EQ" in the "series" column should be processed. As the first stage, filter out all the lines that do not fulfil this criteria:** Show records where series is EQ.

- **For every stock, for every year, calculate the following statistical parameters: Minimum, Maximum, Mean and Standard Deviation and store the generated information in properly designated tables:** From above records, find minimum, maximum, mean and standard deviation and store it in table.

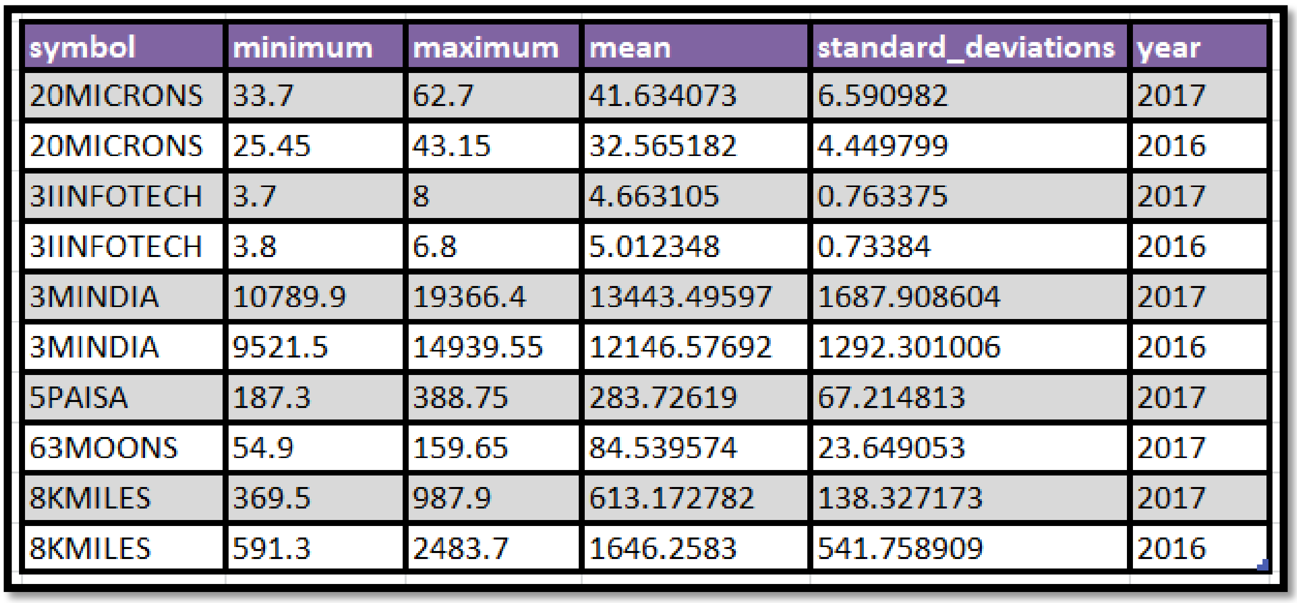
```
stock_market_data = load '/home/suparna/Desktop/stock_market_case_study/i
nput/stock_market.csv' using org.apache.pig.piggybank.storage.CSVLoader()
 as ( symbol:chararray, series:chararray, open:double, high:double, low:d
ouble, close:double, last:double, prevclose:double, tottrdqty:double, tot
trdval:double, timestamp:chararray, totaltrades:int, isin:chararray);
```

```
stock_fltr = filter stock_market_data by LOWER(series) == 'eq';

store stock_fltr into '/home/suparna/Desktop/stock_market_case_study/pig/
output/2_1/' using org.apache.pig.piggybank.storage.CSVExcelStorage('\t',
 'NO_MULTILINE', 'UNIX', 'WRITE_OUTPUT_HEADER');
```



symbol	series	open	high	low	close	last	prevclose	tottrdqty	tottrval	mydate	totaltrades	isin
20MICRONS	EQ	37.8	37.8	36.15	36.85	37.4	37.05	27130	994657.9	28-06-2017	202	INE144J01027
3IINFOTECH	EQ	4.1	4.85	4	4.55	4.65	4.05	20157058	9.21E+07	28-06-2017	7353	INE748C01020
3MINDIA	EQ	13425.15	13469.55	12920	13266.7	13300	13460.55	2290	3.03E+07	28-06-2017	748	INE470A01017
63MOONS	EQ	61	61.9	60.35	61	61.1	60.65	27701	1689421	28-06-2017	437	INE111B01023
8KMILES	EQ	546.1	548	535	537.45	535.2	547.45	79722	4.32E+07	28-06-2017	1866	INE650K01021
A2ZINFRA	EQ	41.3	43	41.25	42	41.9	41.5	606403	2.55E+07	28-06-2017	3418	INE619I01012
AARTIDRUGS	EQ	539.9	539.9	520	521.85	522.2	536.55	8560	4508881.6	28-06-2017	569	INE767A01016
AARTIIND	EQ	890.95	894.9	876.9	891.3	890	890	39201	3.48E+07	28-06-2017	2778	INE769A01020
AARVEEDEN	EQ	58.2	61.2	57	59.5	59.8	58.85	14401	855816.85	28-06-2017	223	INE273D01019
ABAN	EQ	183	184.65	180.8	182.1	181.9	182.65	447698	8.18E+07	28-06-2017	5449	INE421A01028

```
grp = group stock_fltr by (symbol, SUBSTRING(timestamp,0,4));

statistics = foreach grp {
    minimum = MIN(stock_fltr.close); maximum = MAX(stock_fltr.close); mea
n = AVG(stock_fltr.close); row_count = COUNT(stock_fltr.close);

     close_square = foreach stock_fltr generate close * close as (close_
sq:double);

     generate group.$0 as (symbol:chararray), group.$1 as (year:int), mi
nimum as (MIN:double), maximum as (MAX:double), ROUND_TO(mean,6) as (mean
:double), row_count * 1.0 as (row_count:double), ROUND_TO(SUM(close_squar
e.close_sq),4) as (sum_close_sq:double);
};

all_statistics = foreach statistics generate $0, $1, $2, $3, $4, ROUND_TO
(SQRT(($6 / $5) - ($4 * $4)),6) as (stddev:double);

sort_statistics = order all_statistics by symbol, year desc;

store sort_statistics into '/home/suparna/Desktop/stock_market_case_study
/pig/output/2_2/' using org.apache.pig.piggybank.storage.CSVExcelStorage(
'\t', 'NO_MULTILINE', 'UNIX', 'WRITE_OUTPUT_HEADER');
```

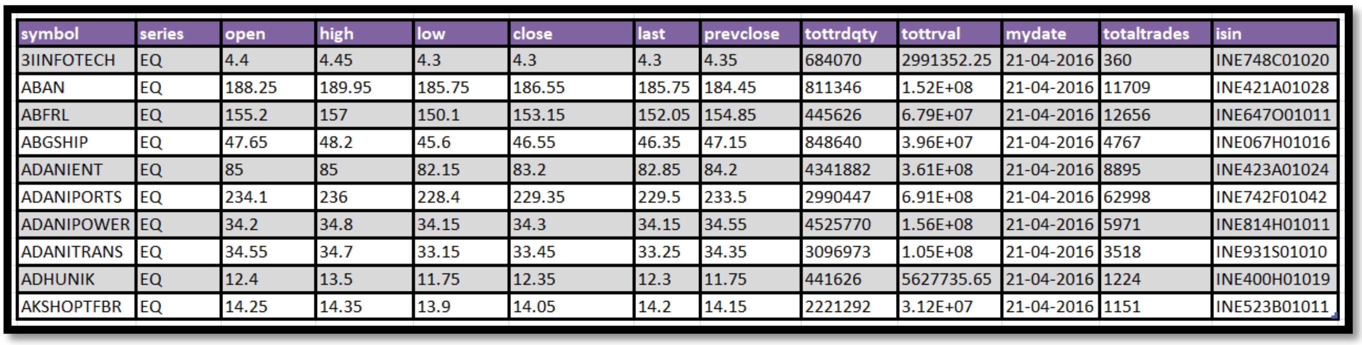


symbol	minimum	maximum	mean	standard_deviations	year
20MICRONS	33.7	62.7	41.634073	6.590982	2017
20MICRONS	25.45	43.15	32.565182	4.449799	2016
3IINFOTECH	3.7	8	4.663105	0.763375	2017
3IINFOTECH	3.8	6.8	5.012348	0.73384	2016
3MINDIA	10789.9	19366.4	13443.49597	1687.908604	2017
3MINDIA	9521.5	14939.55	12146.57692	1292.301006	2016
5PAISA	187.3	388.75	283.72619	67.214813	2017
63MOONS	54.9	159.65	84.539574	23.649053	2017
8KMILES	369.5	987.9	613.172782	138.327173	2017
8KMILES	591.3	2483.7	1646.2583	541.758909	2016

3. Select any year for which data is available:

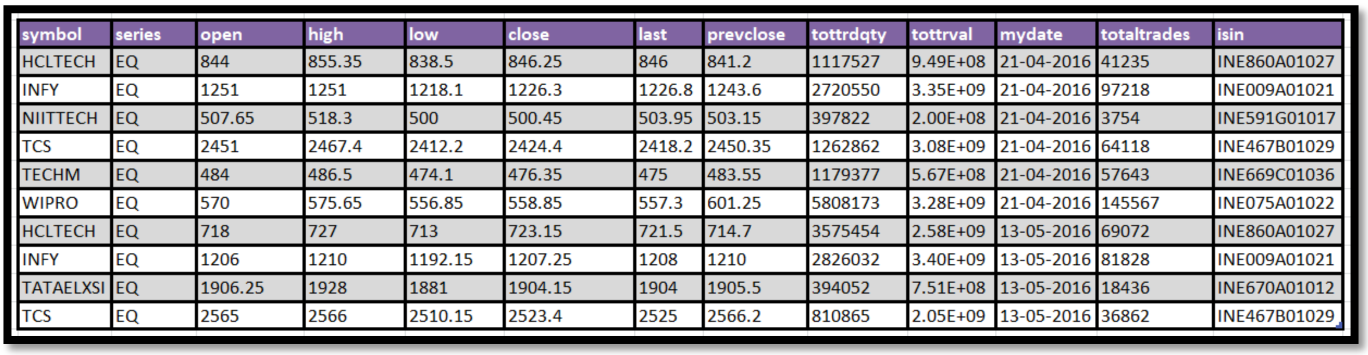
- **For the selected year, create a table that contains data only for those stocks that have an total traded quntity of 3 lakhs or more per day. Print out the first 25 entries of the table and submit:** Extract 25 records from a selected year where Total Trade Quantity >= 300000.
- **From among these, select any 10 stocks from IT ('HCLTECH', 'NIITTECH', 'TATAELXSI','TCS', 'INFY', 'WIPRO', 'DATAMATICS','TECHM','MINDTREE' and 'OFSS') and create a table combining their data:** From above records, select rows where SYMBOL is: HCLTECH or NIITTECH or TATAELXSI or TCS or INFY or WIPRO or DATAMATICS or TECHM or MINDTREE or OFSS and store it in a table.
- **Find out the Pearsons Correlation Coeffecient for every pair of stocks you have selected. Final output should be in decreasing order of the coefficient:** From above records, calculate Pearsons Correlation Coeffecient for every pair of symbol.

```
stock_fltr = filter stock_market_data by tottrdqty >= 300000 and SUBSTRIN
G(timestamp,0,4) == '2016';

limit_data = limit stock_fltr 25;

store limit_data into '/home/suparna/Desktop/stock_market_case_study/pig/
output/3_1/' using org.apache.pig.piggybank.storage.CSVExcelStorage('\t',
 'NO_MULTILINE', 'UNIX', 'WRITE_OUTPUT_HEADER');
```



symbol	series	open	high	low	close	last	prevclose	tottrdqty	tottrval	mydate	totaltrades	isin
3IINFOTECH	EQ	4.4	4.45	4.3	4.3	4.3	4.35	684070	2991352.25	21-04-2016	360	INE748C01020
ABAN	EQ	188.25	189.95	185.75	186.55	185.75	184.45	811346	1.52E+08	21-04-2016	11709	INE421A01028
ABFRL	EQ	155.2	157	150.1	153.15	152.05	154.85	445626	6.79E+07	21-04-2016	12656	INE647O01011
ABGSHIP	EQ	47.65	48.2	45.6	46.55	46.35	47.15	848640	3.96E+07	21-04-2016	4767	INE067H01016
ADANIENT	EQ	85	85	82.15	83.2	82.85	84.2	4341882	3.61E+08	21-04-2016	8895	INE423A01024
ADANIPORTS	EQ	234.1	236	228.4	229.35	229.5	233.5	2990447	6.91E+08	21-04-2016	62998	INE742F01042
ADANIPOWER	EQ	34.2	34.8	34.15	34.3	34.15	34.55	4525770	1.56E+08	21-04-2016	5971	INE814H01011
ADANITRANS	EQ	34.55	34.7	33.15	33.45	33.25	34.35	3096973	1.05E+08	21-04-2016	3518	INE931S01010
ADHUNIK	EQ	12.4	13.5	11.75	12.35	12.3	11.75	441626	5627735.65	21-04-2016	1224	INE400H01019
AKSHOPTFBR	EQ	14.25	14.35	13.9	14.05	14.2	14.15	2221292	3.12E+07	21-04-2016	1151	INE523B01011

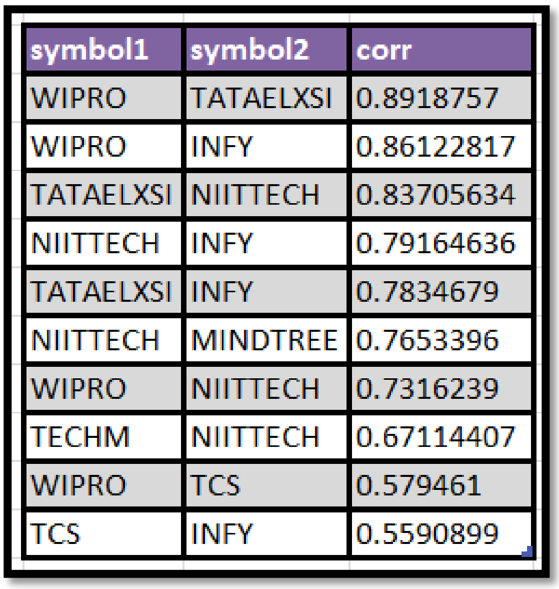
```
itstock = filter stock_fltr by LOWER(symbol) in ('hcltech','niittech','ta
taelxsi','tcs','infy','wipro','datamatics','techm','mindtree','ofss');

store itstock into '/home/suparna/Desktop/stock_market_case_study/pig/out
put/3_2/' using org.apache.pig.piggybank.storage.CSVExcelStorage('\t', 'N
O_MULTILINE', 'UNIX', 'WRITE_OUTPUT_HEADER');
```



symbol	series	open	high	low	close	last	prevclose	tottrdqty	tottrval	mydate	totaltrades	isin
HCLTECH	EQ	844	855.35	838.5	846.25	846	841.2	1117527	9.49E+08	21-04-2016	41235	INE860A01027
INFY	EQ	1251	1251	1218.1	1226.3	1226.8	1243.6	2720550	3.35E+09	21-04-2016	97218	INE009A01021
NIITTECH	EQ	507.65	518.3	500	500.45	503.95	503.15	397822	2.00E+08	21-04-2016	3754	INE591G01017
TCS	EQ	2451	2467.4	2412.2	2424.4	2418.2	2450.35	1262862	3.08E+09	21-04-2016	64118	INE467B01029
TECHM	EQ	484	486.5	474.1	476.35	475	483.55	1179377	5.67E+08	21-04-2016	57643	INE669C01036
WIPRO	EQ	570	575.65	556.85	558.85	557.3	601.25	5808173	3.28E+09	21-04-2016	145567	INE075A01022
HCLTECH	EQ	718	727	713	723.15	721.5	714.7	3575454	2.58E+09	13-05-2016	69072	INE860A01027
INFY	EQ	1206	1210	1192.15	1207.25	1208	1210	2826032	3.40E+09	13-05-2016	81828	INE009A01021
TATAELXSI	EQ	1906.25	1928	1881	1904.15	1904	1905.5	394052	7.51E+08	13-05-2016	18436	INE670A01012
TCS	EQ	2565	2566	2510.15	2523.4	2525	2566.2	810865	2.05E+09	13-05-2016	36862	INE467B01029

```
stock_market_data = load '/home/suparna/Desktop/stock_market_case_study/i
nput/stock_market.csv' using org.apache.pig.piggybank.storage.CSVLoader()
 as ( symbol:chararray, series:chararray, open:double, high:double, low:d
ouble, close:double, last:double, prevclose:double, tottrdqty:double, tot
trdval:double, timestamp:chararray, totaltrades:int, isin:chararray);

stock_fltr = filter stock_market_data by tottrdqty >= 300000 and SUBSTRIN
G(timestamp,0,4) == '2016';

stock_it_1 = filter stock_fltr by LOWER(symbol) in ('hcltech', 'niittech'
, 'tataelxsi','tcs', 'infy', 'wipro', 'datamatics','techm','mindtree', 'o
fss');

stock_it_2 = filter stock_fltr by LOWER(symbol) in ('hcltech', 'niittech'
, 'tataelxsi','tcs', 'infy', 'wipro', 'datamatics','techm','mindtree', 'o
fss');

stock_it_1_2 = join stock_it_1 by timestamp, stock_it_2 by timestamp;

fltr = filter stock_it_1_2 by stock_it_1::symbol > stock_it_2::symbol;

group_symbol_1_2 = group fltr by (stock_it_1::symbol,stock_it_2::symbol);
```

```
correlation = foreach group_symbol_1_2 {
    row_count = COUNT(fltr.stock_it_1::symbol);

      stats = foreach fltr generate stock_it_1::close as (close1:double),
 stock_it_2::close as (close2:double), stock_it_1::close * stock_it_1::cl
ose as (c1c1:double), stock_it_2::close * stock_it_2::close as (c2c2:doub
le), stock_it_1::close * stock_it_2::close as (c1c2:double);

      generate group.$0 as (symbol1:chararray), group.$1 as (symbol2:char
array), SUM(stats.close1) as (sum_c1:double), SUM(stats.close2)  as (sum_
c2:double), SUM(stats.c1c1) as (sum_c1c1:double), SUM(stats.c2c2) as (sum
_c2c2:double), SUM(stats.c1c2) as (sum_c1c2:double), row_count * 1.0 as (
num:double), fltr.$10 as timestamp;
};

correlation_stock_it = order correlation by symbol1, symbol2, timestamp;

pearsons_corr_coeff_it = foreach correlation generate symbol1, symbol2, R
OUND_TO((sum_c1c2 - (sum_c1 * sum_c2 / num)) / SQRT((sum_c1c1 -
 (sum_c1 * sum_c1) / num) * (sum_c2c2 -
 (sum_c2 * sum_c2) / num)),8) as (pearsoncoefficient:double);

pearsons_corr_coeff_sort = order pearsons_corr_coeff_it by pearsoncoeffic
ient desc;

store pearsons_corr_coeff_sort into '/home/suparna/Desktop/stock_market_c
ase_study/pig/output/3_3/' using org.apache.pig.piggybank.storage.CSVExce
lStorage('\t', 'NO_MULTILINE', 'UNIX', 'WRITE_OUTPUT_HEADER');
```



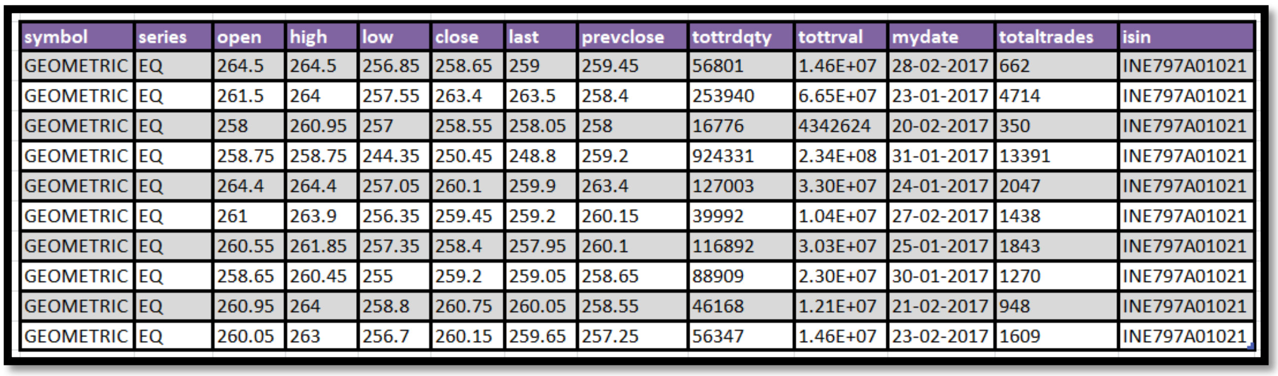
symbol1	symbol2	corr
WIPRO	TATAELXSI	0.8918757
WIPRO	INFY	0.86122817
TATAELXSI	NIITTECH	0.83705634
NIITTECH	INFY	0.79164636
TATAELXSI	INFY	0.7834679
NIITTECH	MINDTREE	0.7653396
WIPRO	NIITTECH	0.7316239
TECHM	NIITTECH	0.67114407
WIPRO	TCS	0.579461
TCS	INFY	0.5590899

HIVE

1. Use the given csv file as input data and implement following transformations:

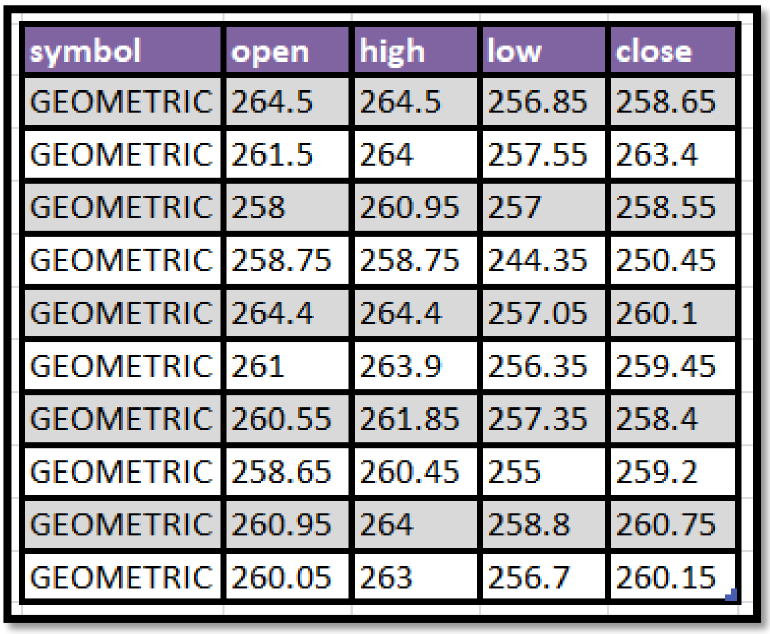
- **Filter Rows on specified criteria "Symbol equals GEOMETRIC":** Show records where symbol is geometric.
- **Select specific columns from those available: SYMBOL, OPEN, HIGH, LOW and CLOSE which meets above criteria:** Show symbol, open, high, low and close columns where symbol is geometric.
- **Generate count of the number of rows from above result:** Count number of records generated from above query.

```
hdfs dfs -mkdir /stock_market_case_study
hdfs dfs -mkdir /stock_market_case_study/input
hdfs dfs -
put /home/suparna/Desktop/stock_market_case_study/input/stock_market.csv
/stock_market_case_study/input/

hive -e “create database stock_db”

hive -e
“create external table stock_db.stock_market_data (symbol string, series
string, open double, high double, low double, close double, last double,
prevclose double, tottrdqty int, tottrval double, mydate string, totaltra
des int, isin string) row format delimited fields terminated by ',' store
d as textfile location '/stock_market_case_study/input'”

hive -e
“select * from  stock_db.stock_market_data where lower(symbol) = 'geometr
ic'” > /home/suparna/Desktop/stock_market_case_study/hive/output1_1.tsv
```



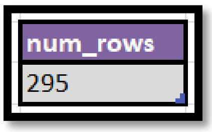
symbol	series	open	high	low	close	last	prevclose	tottrdqty	tottrval	mydate	totaltrades	isin
GEOMETRIC	EQ	264.5	264.5	256.85	258.65	259	259.45	56801	1.46E+07	28-02-2017	662	INE797A01021
GEOMETRIC	EQ	261.5	264	257.55	263.4	263.5	258.4	253940	6.65E+07	23-01-2017	4714	INE797A01021
GEOMETRIC	EQ	258	260.95	257	258.55	258.05	258	16776	4342624	20-02-2017	350	INE797A01021
GEOMETRIC	EQ	258.75	258.75	244.35	250.45	248.8	259.2	924331	2.34E+08	31-01-2017	13391	INE797A01021
GEOMETRIC	EQ	264.4	264.4	257.05	260.1	259.9	263.4	127003	3.30E+07	24-01-2017	2047	INE797A01021
GEOMETRIC	EQ	261	263.9	256.35	259.45	259.2	260.15	39992	1.04E+07	27-02-2017	1438	INE797A01021
GEOMETRIC	EQ	260.55	261.85	257.35	258.4	257.95	260.1	116892	3.03E+07	25-01-2017	1843	INE797A01021
GEOMETRIC	EQ	258.65	260.45	255	259.2	259.05	258.65	88909	2.30E+07	30-01-2017	1270	INE797A01021
GEOMETRIC	EQ	260.95	264	258.8	260.75	260.05	258.55	46168	1.21E+07	21-02-2017	948	INE797A01021
GEOMETRIC	EQ	260.05	263	256.7	260.15	259.65	257.25	56347	1.46E+07	23-02-2017	1609	INE797A01021

```
hive -e
“select symbol, open, high, low, close from stock_db.stock_market_data wh
ere lower(symbol) = 'geometric'” > /home/suparna/Desktop/stock_market_cas
e_study/hive/output1_2.tsv
```



symbol	open	high	low	close
GEOMETRIC	264.5	264.5	256.85	258.65
GEOMETRIC	261.5	264	257.55	263.4
GEOMETRIC	258	260.95	257	258.55
GEOMETRIC	258.75	258.75	244.35	250.45
GEOMETRIC	264.4	264.4	257.05	260.1
GEOMETRIC	261	263.9	256.35	259.45
GEOMETRIC	260.55	261.85	257.35	258.4
GEOMETRIC	258.65	260.45	255	259.2
GEOMETRIC	260.95	264	258.8	260.75
GEOMETRIC	260.05	263	256.7	260.15

```
hive -e
“select count(*) as num_rows from stock_db.stock_market_data where lower(
symbol) = 'geometric'” > /home/suparna/Desktop/stock_market_case_study/hi
ve/output1_3.tsv
```

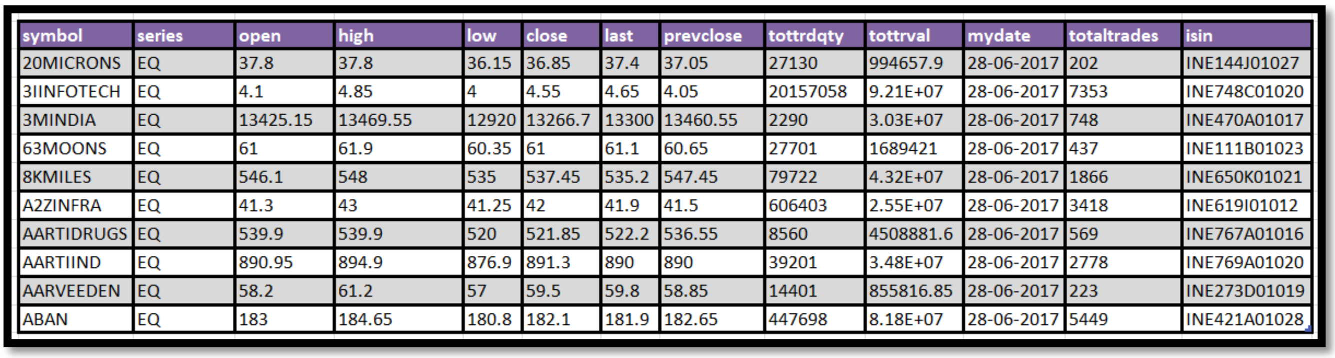


num_rows
295

2. Calculation of various statistical quantities and decision making:

- **Only lines with value "EQ" in the "series" column should be processed. As the first stage, filter out all the lines that do not fulfil this criteria:** Show records where series is EQ.
- **For every stock, for every year, calculate the following statistical parameters: Minimum, Maximum, Mean and Standard Deviation and store the generated information in properly designated tables:** From above records, find minimum, maximum, mean and standard deviation and store it in table.

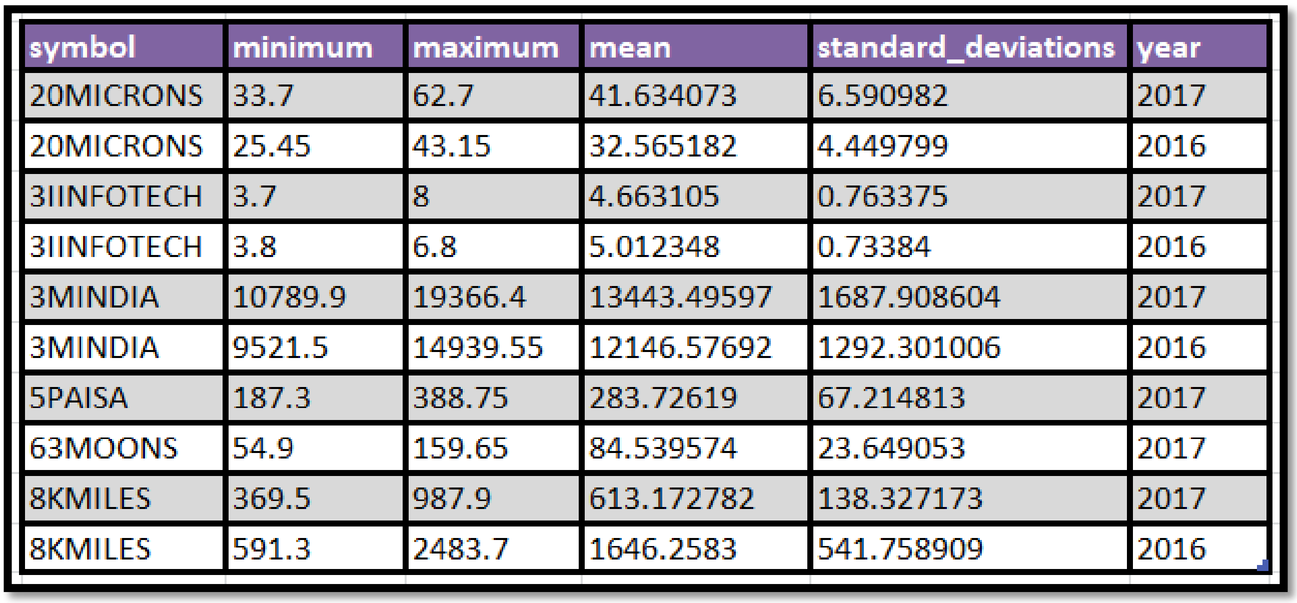
```
hive -e
“select * from  stock_db.stock_market_data where lower(series) = 'eq'” >
/home/suparna/Desktop/stock_market_case_study/hive/output2_1.tsv
```



symbol	series	open	high	low	close	last	prevclose	tottrdqty	tottrval	mydate	totaltrades	isin
20MICRONS	EQ	37.8	37.8	36.15	36.85	37.4	37.05	27130	994657.9	28-06-2017	202	INE144J01027
3IINFOTECH	EQ	4.1	4.85	4	4.55	4.65	4.05	20157058	9.21E+07	28-06-2017	7353	INE748C01020
3MINDIA	EQ	13425.15	13469.55	12920	13266.7	13300	13460.55	2290	3.03E+07	28-06-2017	748	INE470A01017
63MOONS	EQ	61	61.9	60.35	61	61.1	60.65	27701	1689421	28-06-2017	437	INE111B01023
8KMILES	EQ	546.1	548	535	537.45	535.2	547.45	79722	4.32E+07	28-06-2017	1866	INE650K01021
A2ZINFRA	EQ	41.3	43	41.25	42	41.9	41.5	606403	2.55E+07	28-06-2017	3418	INE619I01012
AARTIDRUGS	EQ	539.9	539.9	520	521.85	522.2	536.55	8560	4508881.6	28-06-2017	569	INE767A01016
AARTIIND	EQ	890.95	894.9	876.9	891.3	890	890	39201	3.48E+07	28-06-2017	2778	INE769A01020
AARVEEDEN	EQ	58.2	61.2	57	59.5	59.8	58.85	14401	855816.85	28-06-2017	223	INE273D01019
ABAN	EQ	183	184.65	180.8	182.1	181.9	182.65	447698	8.18E+07	28-06-2017	5449	INE421A01028

```
hive -e
“select symbol, min(close) as minimum, max(close) as
maximum, round(avg(close),6) as mean, round(stddev_pop(close),6) as
standard_deviation, substr(mydate,1,4) as year from  stock_db.stock_marke
t_data where lower(series) = 'eq' group by symbol, substr(mydate,1,4) ord
er by symbol, year desc” > /home/suparna/Desktop/stock_market_case_study/
hive/output2_2.tsv
```

```
hive -e
“create external table  stock_db.stock_statistical_param (symbol string,
min float, max float, mean float, std float, year string) row format deli
mited fields terminated by ',' stored as textfile location '/home/suparna
/Desktop/stock_market_case_study/hive/output2_2.tsv'”
```



symbol	minimum	maximum	mean	standard_deviations	year
20MICRONS	33.7	62.7	41.634073	6.590982	2017
20MICRONS	25.45	43.15	32.565182	4.449799	2016
3IINFOTECH	3.7	8	4.663105	0.763375	2017
3IINFOTECH	3.8	6.8	5.012348	0.73384	2016
3MINDIA	10789.9	19366.4	13443.49597	1687.908604	2017
3MINDIA	9521.5	14939.55	12146.57692	1292.301006	2016
5PAISA	187.3	388.75	283.72619	67.214813	2017
63MOONS	54.9	159.65	84.539574	23.649053	2017
8KMILES	369.5	987.9	613.172782	138.327173	2017
8KMILES	591.3	2483.7	1646.2583	541.758909	2016

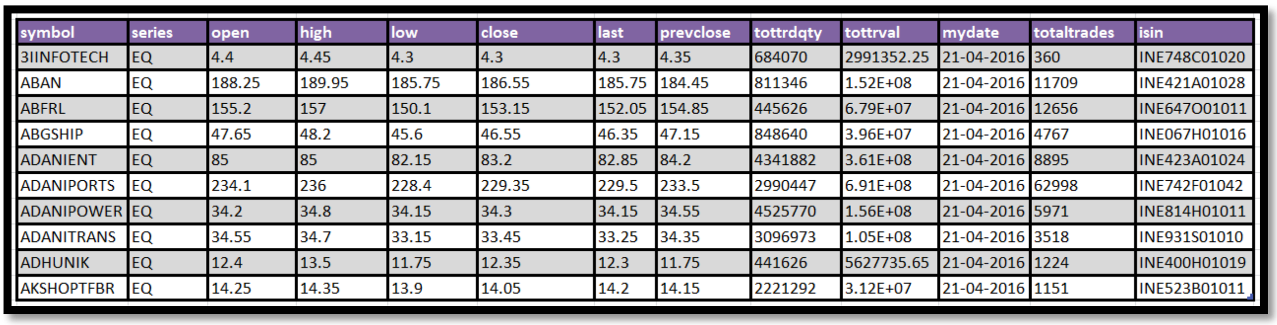
3. Select any year for which data is available:

- **For the selected year, create a table that contains data only for those stocks that have an total traded quntity of 3 lakhs or more per day. Print out the first 25 entries of the table and submit:** Extract 25 records from a selected year where Total Trade Quantity >= 300000.
- **From among these, select any 10 stocks from IT ('HCLTECH', 'NIITTECH', 'TATAELXSI','TCS', 'INFY', 'WIPRO', 'DATAMATICS','TECHM','MINDTREE' and 'OFSS') and create a table combining their data:** From above records, select rows where SYMBOL is: HCLTECH or NIITTECH or TATAELXSI or TCS or INFY or WIPRO or DATAMATICS or TECHM or MINDTREE or OFSS and store it in a table.
- **Find out the Pearsons Correlation Coeffecient for every pair of stocks you have selected. Final output should be in decreasing order of the coefficient:** From above records, calculate Pearsons Correlation Coeffecient for every pair of symbol.

```
hive -e
“create table stock_db.stock_market_data_2016(symbol string, series strin
g, open double, high double, low double, close double, last double, prevc
lose double, tottrdqty int, tottrval double, mydate string, totaltrades i
nt, isin string)”
```

```
hive -e
“insert overwrite table stock_db.stock_market_data_2016 select * from  st
ock_db.stock_market_data where tottrdqty >= 300000 and substr(mydate,1,4)
 = '2016'  “
```

```
hive -e
“select * from  stock_db.stock_market_data_2016 limit 25” > /home/suparna
```

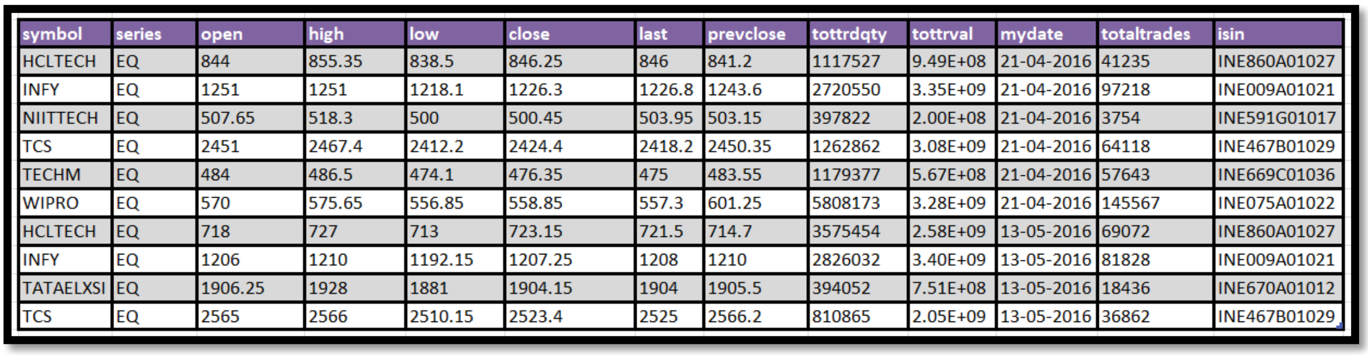


symbol	series	open	high	low	close	last	prevclose	tottrdqty	tottrval	mydate	totaltrades	isin
3IINFOTECH	EQ	4.4	4.45	4.3	4.3	4.3	4.35	684070	2991352.25	21-04-2016	360	INE748C01020
ABAN	EQ	188.25	189.95	185.75	186.55	185.75	184.45	811346	1.52E+08	21-04-2016	11709	INE421A01028
ABFRL	EQ	155.2	157	150.1	153.15	152.05	154.85	445626	6.79E+07	21-04-2016	12656	INE647O01011
ABGSHIP	EQ	47.65	48.2	45.6	46.55	46.35	47.15	848640	3.96E+07	21-04-2016	4767	INE067H01016
ADANIENT	EQ	85	85	82.15	83.2	82.85	84.2	4341882	3.61E+08	21-04-2016	8895	INE423A01024
ADANIPORTS	EQ	234.1	236	228.4	229.35	229.5	233.5	2990447	6.91E+08	21-04-2016	62998	INE742F01042
ADANIPOWER	EQ	34.2	34.8	34.15	34.3	34.15	34.55	4525770	1.56E+08	21-04-2016	5971	INE814H01011
ADANITRANS	EQ	34.55	34.7	33.15	33.45	33.25	34.35	3096973	1.05E+08	21-04-2016	3518	INE931S01010
ADHUNIK	EQ	12.4	13.5	11.75	12.35	12.3	11.75	441626	5627735.65	21-04-2016	1224	INE400H01019
AKSHOPTFBR	EQ	14.25	14.35	13.9	14.05	14.2	14.15	2221292	3.12E+07	21-04-2016	1151	INE523B01011

```
/Desktop/stock_market_case_study/hive/output3_1.tsv
```

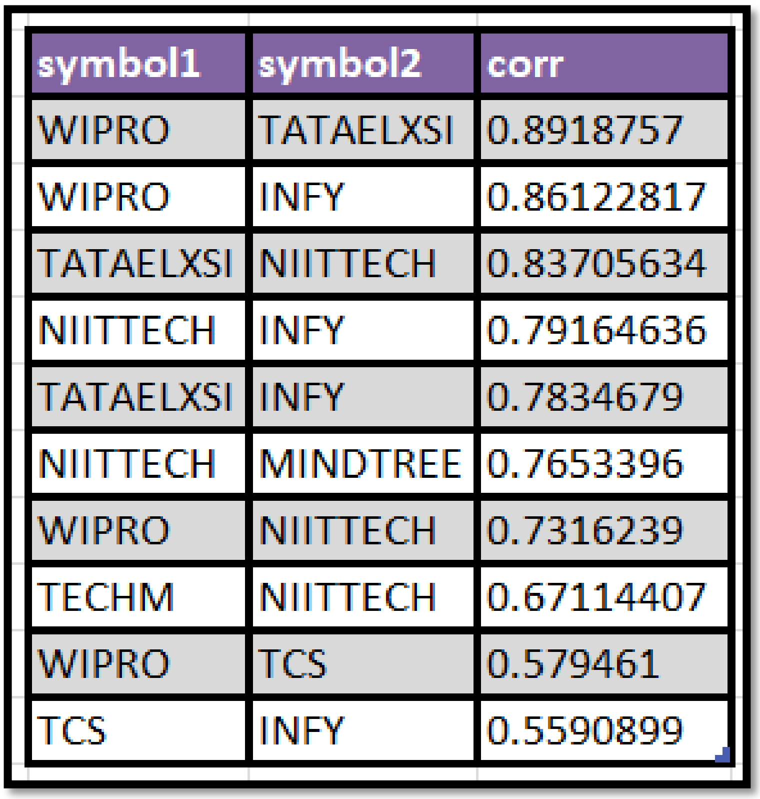
```
hive -e
“create table stock_db.it_companies_stock (symbol string, series string,
open double, high double, low double, close double, last double, prevclos
e double, tottrdqty int, tottrval double, mydate string, totaltrades int,
 isin string)”
```

```
hive -e
“insert overwrite table stock_db.it_companies_stock select * from  stock_
db.stock_market_data_2016 where lower(symbol) in ('hcltech', 'niittech',
'tataelxsi','tcs', 'infy', 'wipro', 'datamatics','techm','mindtree', 'ofs
s')”

hive -e
“select * from stock_db.it_companies_stock” > /home/suparna/Desktop/stock
_market_case_study/hive/output3_2.tsv
```



symbol	series	open	high	low	close	last	prevclose	tottrdqty	tottrval	mydate	totaltrades	isin
HCLTECH	EQ	844	855.35	838.5	846.25	846	841.2	1117527	9.49E+08	21-04-2016	41235	INE860A01027
INFY	EQ	1251	1251	1218.1	1226.3	1226.8	1243.6	2720550	3.35E+09	21-04-2016	97218	INE009A01021
NIITTECH	EQ	507.65	518.3	500	500.45	503.95	503.15	397822	2.00E+08	21-04-2016	3754	INE591G01017
TCS	EQ	2451	2467.4	2412.2	2424.4	2418.2	2450.35	1262862	3.08E+09	21-04-2016	64118	INE467B01029
TECHM	EQ	484	486.5	474.1	476.35	475	483.55	1179377	5.67E+08	21-04-2016	57643	INE669C01036
WIPRO	EQ	570	575.65	556.85	558.85	557.3	601.25	5808173	3.28E+09	21-04-2016	145567	INE075A01022
HCLTECH	EQ	718	727	713	723.15	721.5	714.7	3575454	2.58E+09	13-05-2016	69072	INE860A01027
INFY	EQ	1206	1210	1192.15	1207.25	1208	1210	2826032	3.40E+09	13-05-2016	81828	INE009A01021
TATAELXSI	EQ	1906.25	1928	1881	1904.15	1904	1905.5	394052	7.51E+08	13-05-2016	18436	INE670A01012
TCS	EQ	2565	2566	2510.15	2523.4	2525	2566.2	810865	2.05E+09	13-05-2016	36862	INE467B01029

```
hive -e
“create table stock_db.it_companies_stock_close (symbol1 string, close1 f
loat, symbol2 string, close2 float,mydate string)”

hive -e
“insert overwrite table stock_db.it_companies_stock_close select t1.symbo
l,t1.close, t2.symbol, t2.close, from_unixtime(unix_timestamp(t1.mydate,
'yyyy-mm-dd'), 'yyyy-mmm-
dd') as md from stock_db.it_companies_stock t1 cross join stock_db.it_com
panies_stock t2 where t1.symbol > t2.symbol and t1.mydate=t2.mydate order
 by t1.symbol asc,t2.symbol asc, from_unixtime(unix_timestamp(md, 'yyyy-
mm-dd'), 'yyyy-mmm-dd') asc”

hive -e
“create table stock_db.it_pearsons_corr_coeff(symbol1 string, symbol2 str
ing, corr float)”

hive -e
“insert overwrite table stock_db.it_pearsons_corr_coeff select symbol1, s
ymbol2, (avg(close1*close2) -
 (avg(close1) *avg(close2)))/(stddev_pop(close1) * stddev_pop(close2)) as
 pearsoncoefficient from stock_db.it_companies_stock_close group by symbo
l1, symbol2 order by pearsoncoefficient desc”

hive -e
“select * from stock_db.it_pearsons_corr_coeff” > /home/suparna/Desktop/s
tock_market_case_study/hive/output3_3.tsv
```



symbol1	symbol2	corr
WIPRO	TATAELXSI	0.8918757
WIPRO	INFY	0.86122817
TATAELXSI	NIITTECH	0.83705634
NIITTECH	INFY	0.79164636
TATAELXSI	INFY	0.7834679
NIITTECH	MINDTREE	0.7653396
WIPRO	NIITTECH	0.7316239
TECHM	NIITTECH	0.67114407
WIPRO	TCS	0.579461
TCS	INFY	0.5590899