## 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

[Market Stock Analysis 1](#_Toc56960664)

[Acknowledgement 2](#_Toc56960665)

[Index 3](#_Toc56960666)

[Objectives 4](#_Toc56960667)

[Functionality 5](#_Toc56960668)

[Hardware & Software Requirements 6](#_Toc56960669)

[Why Hadoop 7](#_Toc56960670)

[Big Data Tools 8](#_Toc56960671)

[Future Scope of Big Data 11](#_Toc56960672)

[SPARK 13](#_Toc56960673)

[PIG 19](#_Toc56960674)

[HIVE 25](#_Toc56960675)

## 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:**

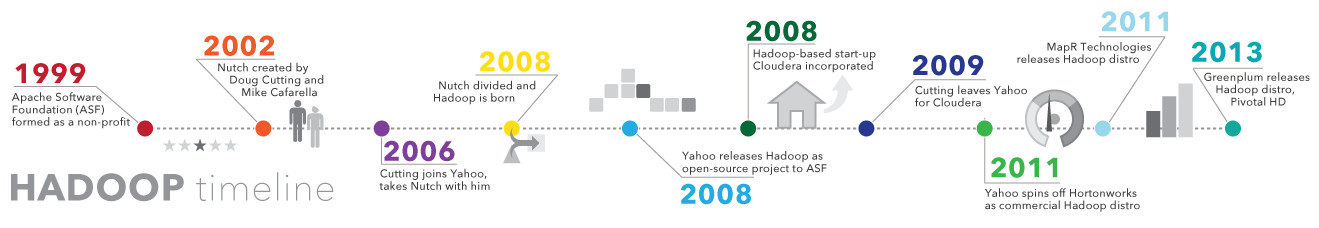
* Oracle Virtual Box
* 64 bit Ubuntu installed in Oracle Virtual Box
* Download and install Java, Hadoop, Hive, Pig and Spark

**Minimum Hardware requirements for this Case Study:**

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

## Why Hadoop

### History of Hadoop:



### 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

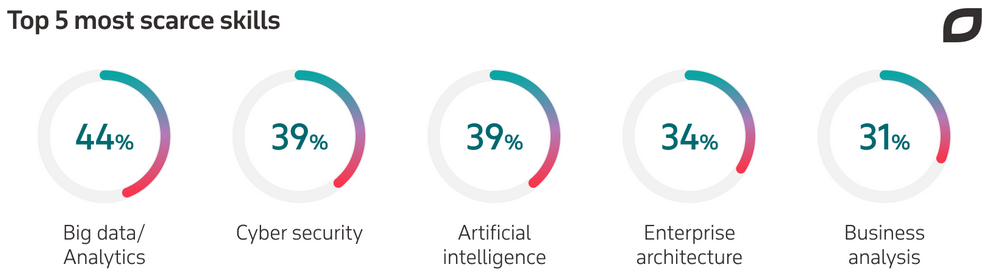
|  |  |  |
| --- | --- | --- |
| **Factors** | **Hadoop MapReduce** | **Apache Spark** |
| **Core Definition** | MapReduce is a programming model that is implemented in processing huge amounts of data.  MapReduce has been developed using Java  MapReduce Programs work in two phases:   * The Map Phase * The Reduce Phase   The entire MapReduce process goes through the following 4 phases:   * **Splitting**: The input 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. | 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. * **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 to wait to get a response after it makes a request to the RAM. | MapReduce has a high-latency computing framework. | Spark provides a low latency computing. |
| **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](https://www.itransition.com/blog/big-data-and-its-business-impacts).

## SPARK

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

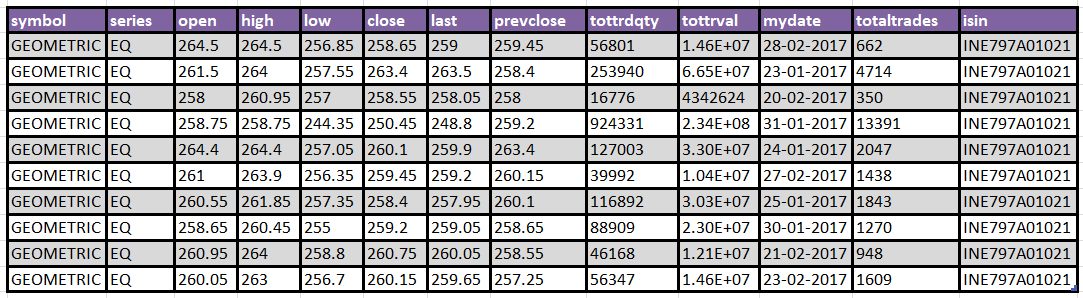
* **Filter Rows on specified criteria "Symbol equals GEOMETRIC":** Show records where symbol is geometric.
* **Select specific columns from those available: SYMBOL, OPEN, HIGH, LOW and CLOSE which meets above criteria:** Show symbol, open, high, low and close columns where symbol is geometric.
* **Generate 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/input/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)})**

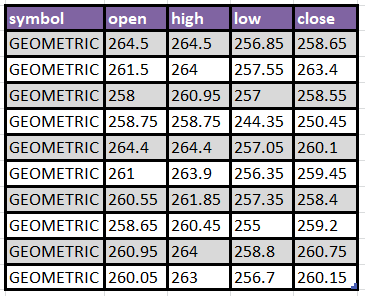
**output\_1\_1.map(line => line.productIterator.mkString("\t")).repartition(1).saveAsTextFile("/home/suparna/Desktop/stock\_market\_case\_study/spark/output\_1\_1")**



**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/output\_1\_2")**



**val output\_1\_3 = sc.parallelize(Seq(output\_1\_2.count(),""))**

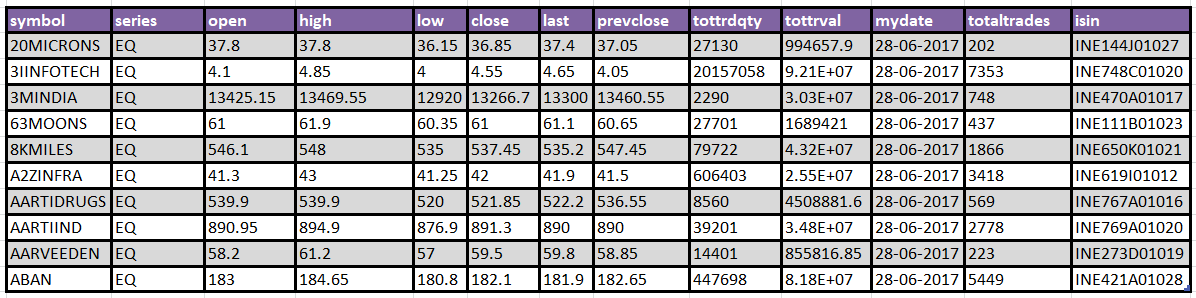
**output\_1\_3.repartition(1).saveAsTextFile("/home/suparna/Desktop/stock\_market\_case\_study/spark/output\_1\_3")**



### 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/input/stock\_market.csv").map(line => line.split(","))**

**val stock\_filter = input.filter(x => x(1).toString == "EQ")**

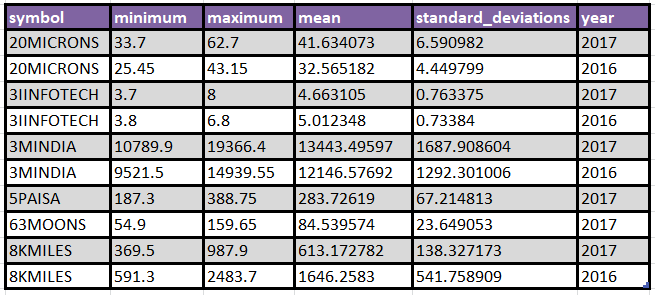
**val output\_2\_1 = stock\_filter.map(x => {(x(0).toString, x(1).toString, x(2).toDouble, x(3).toDouble, x(4).toDouble, x(5).toDouble, x(6).toDouble, x(7).toDouble, x(8).toDouble, x(9).toDouble, x(10).toString, x(11).toLong, x(12).toString)})**

**output\_2\_1.map(line => line.productIterator.mkString("\t")).repartition(1).saveAsTextFile("/home/suparna/Desktop/stock\_market\_case\_study/spark/output\_2\_1")**

**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/output\_2\_2")**

### 3. Select any year for which data is available:

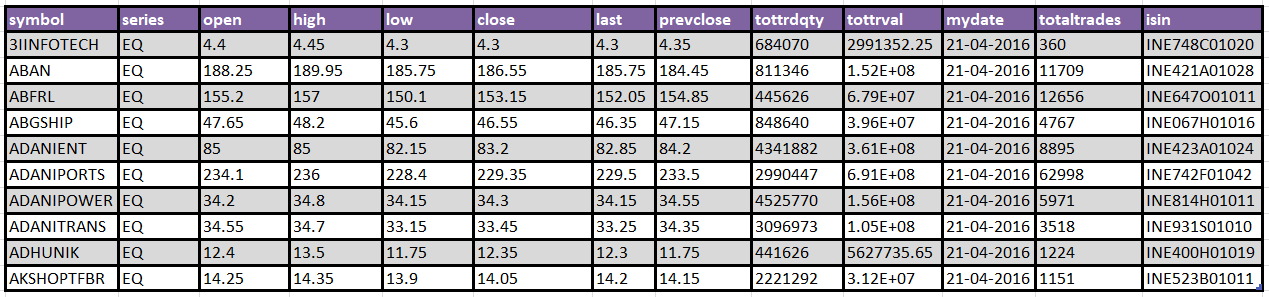
* **For the selected year, create a table that contains data only for those stocks that have an total traded quntity of 3 lakhs or more per day. Print out the first 25 entries of the table and submit:** Extract 25 records from a selected year where Total Trade Quantity >= 300000.
* **From among these, 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).toString.split("-")(0) == "2016")**

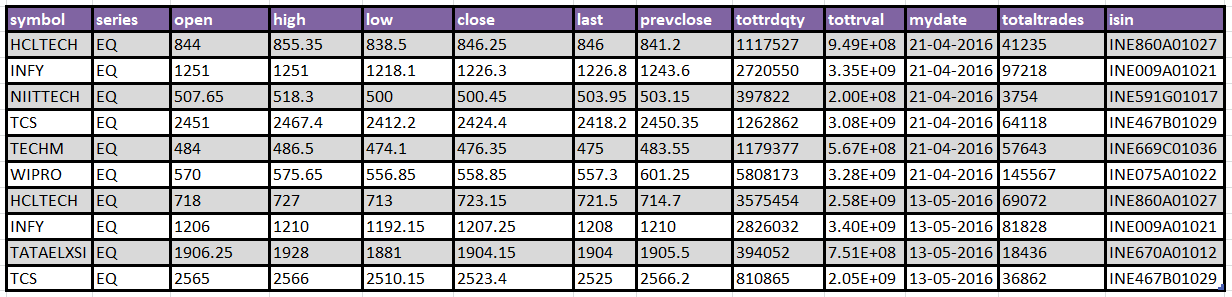
**val stock\_map = stock\_filter.map(x => {(x(0).toString, x(1).toString, x(2).toDouble, x(3).toDouble, x(4).toDouble, x(5).toDouble, x(6).toDouble, x(7).toDouble, x(8).toDouble, x(9).toDouble, x(10).toString, x(11).toLong, x(12).toString)})**

**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/output\_3\_1")**



**val output\_3\_2 = stock\_map.filter(x => x.\_1 == "HCLTECH" || x.\_1 == "NIITTECH" || 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/output\_3\_2")**

**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).sortBy(\_.\_2.\_1.\_1).sortBy(\_.\_2.\_2.\_1).sortBy(\_.\_1)**

**val map1 = s1\_ge\_s2.map(x => x.\_2)**

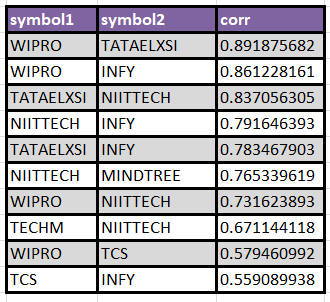
**val map2 = map1.map(x => ((x.\_1.\_1, x.\_2.\_1), (x.\_1.\_2, x.\_2.\_2, x.\_1.\_2 \* x.\_2.\_2)))**

**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.toDouble)).sortBy(\_.\_3, false)**

**output\_3\_3.map(line => line.productIterator.mkString("\t")).repartition(1).saveAsTextFile("/home/suparna/Desktop/stock\_market\_case\_study/spark/output\_3\_3")**



## PIG

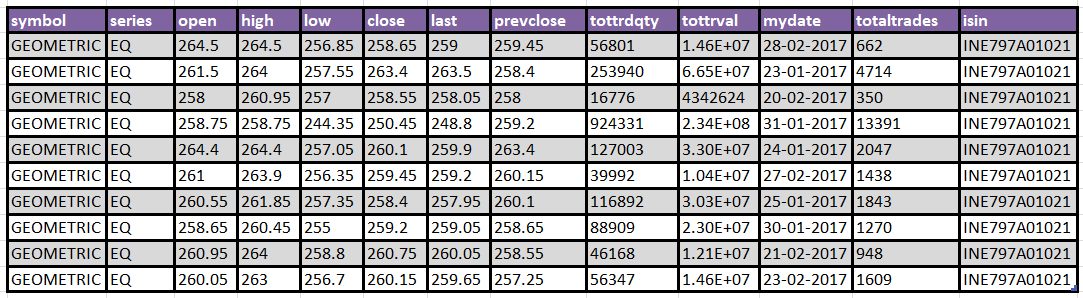
### 1. Use the given csv file as input data and implement following transformations:

* **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/input/stock\_market.csv' using org.apache.pig.piggybank.storage.CSVLoader() as (symbol:chararray, series:chararray, open:double, high:double, low:double, close:double, last:double, prevclose:double, tottrdqty:double, tottrdval:double, timestamp:chararray, totaltrades:int, isin:chararray);**

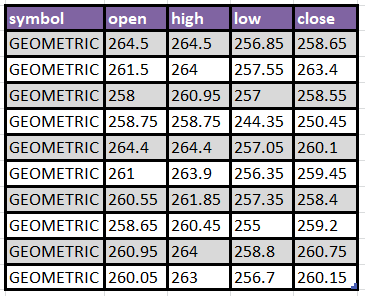
**stock\_fltr = filter stock\_market\_data by LOWER(symbol) == 'geometric';**

**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') ;**



**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');**



**grp\_cols = group select\_data all;**

**stock\_count= foreach grp\_cols generate COUNT(select\_data.symbol) as (count\_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');**



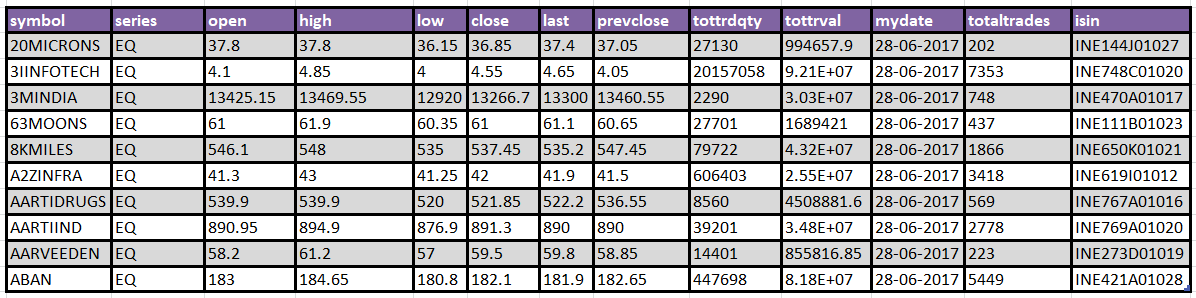
### 2. Calculation of various statistical quantities and decision making:

* **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/input/stock\_market.csv' using org.apache.pig.piggybank.storage.CSVLoader() as ( symbol:chararray, series:chararray, open:double, high:double, low:double, close:double, last:double, prevclose:double, tottrdqty:double, tottrdval: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');**



**grp = group stock\_fltr by (symbol, SUBSTRING(timestamp,0,4));**

**statistics = foreach grp {**

**minimum = MIN(stock\_fltr.close); maximum = MAX(stock\_fltr.close); mean = AVG(stock\_fltr.close); row\_count = COUNT(stock\_fltr.close);**

**close\_square = foreach stock\_fltr generate close \* close as (close\_sq:double);**

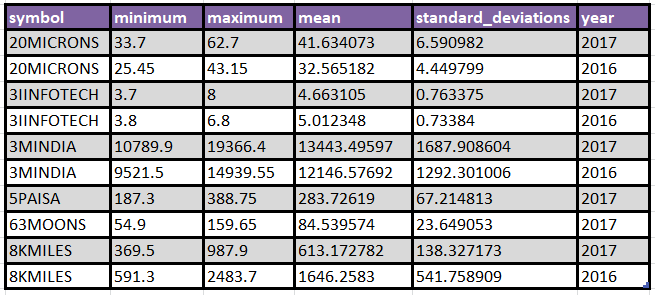
**generate group.$0 as (symbol:chararray), group.$1 as (year:int), minimum as (MIN:double), maximum as (MAX:double), ROUND\_TO(mean,6) as (mean:double), row\_count \* 1.0 as (row\_count:double), ROUND\_TO(SUM(close\_square.close\_sq),4) as (sum\_close\_sq:double);**

**};**

**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');**

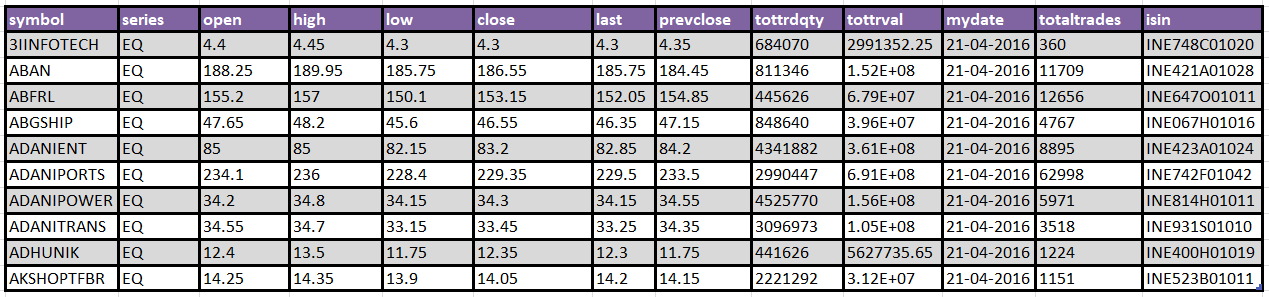


### 3. Select any year for which data is available:

* **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.

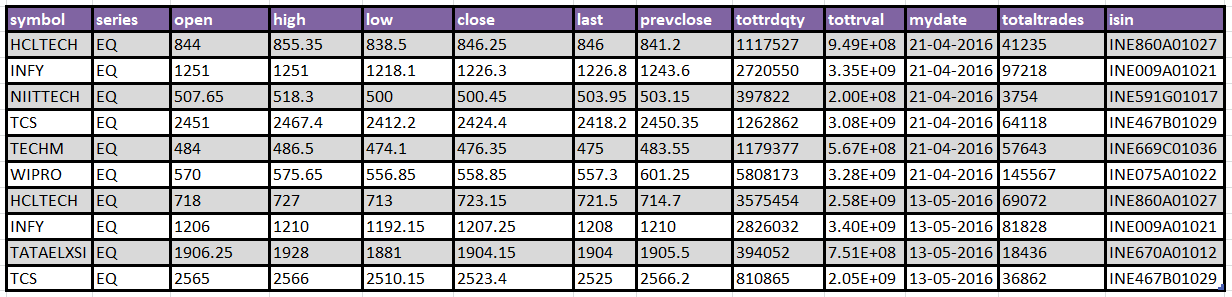
**stock\_fltr = filter stock\_market\_data by tottrdqty >= 300000 and SUBSTRING(timestamp,0,4) == '2016';**

**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');**

**itstock = filter stock\_fltr by LOWER(symbol) in ('hcltech','niittech','tataelxsi','tcs','infy','wipro','datamatics','techm','mindtree','ofss');**

**store itstock into '/home/suparna/Desktop/stock\_market\_case\_study/pig/output/3\_2/' using org.apache.pig.piggybank.storage.CSVExcelStorage('\t', 'NO\_MULTILINE', 'UNIX', 'WRITE\_OUTPUT\_HEADER');**



**stock\_market\_data = load '/home/suparna/Desktop/stock\_market\_case\_study/input/stock\_market.csv' using org.apache.pig.piggybank.storage.CSVLoader() as ( symbol:chararray, series:chararray, open:double, high:double, low:double, close:double, last:double, prevclose:double, tottrdqty:double, tottrdval:double, timestamp:chararray, totaltrades:int, isin:chararray);**

**stock\_fltr = filter stock\_market\_data by tottrdqty >= 300000 and SUBSTRING(timestamp,0,4) == '2016';**

**stock\_it\_1 = filter stock\_fltr by LOWER(symbol) in ('hcltech', 'niittech', 'tataelxsi','tcs', 'infy', 'wipro', 'datamatics','techm','mindtree', 'ofss');**

**stock\_it\_2 = filter stock\_fltr by LOWER(symbol) in ('hcltech', 'niittech', 'tataelxsi','tcs', 'infy', 'wipro', 'datamatics','techm','mindtree', 'ofss');**

**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::close as (c1c1:double), stock\_it\_2::close \* stock\_it\_2::close as (c2c2:double), stock\_it\_1::close \* stock\_it\_2::close as (c1c2:double);**

**generate group.$0 as (symbol1:chararray), group.$1 as (symbol2:chararray), 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;**

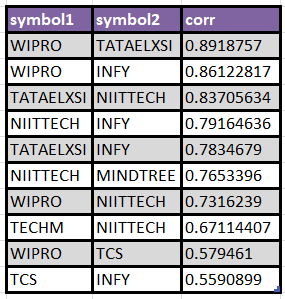
**};**

**correlation\_stock\_it = order correlation by symbol1, symbol2, timestamp;**

**pearsons\_corr\_coeff\_it = foreach correlation generate symbol1, symbol2, ROUND\_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 pearsoncoefficient desc;**

**store pearsons\_corr\_coeff\_sort into '/home/suparna/Desktop/stock\_market\_case\_study/pig/output/3\_3/' using org.apache.pig.piggybank.storage.CSVExcelStorage('\t', 'NO\_MULTILINE', 'UNIX', 'WRITE\_OUTPUT\_HEADER');**



## 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/**

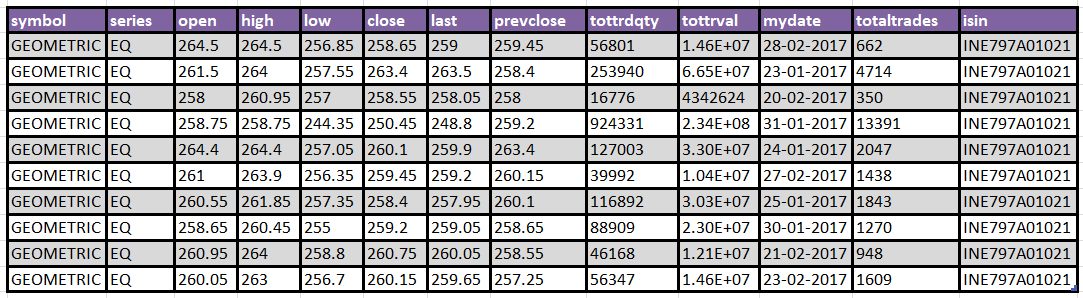
**hive -e “create database stock\_db”**

**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, totaltrades int, isin string) row format delimited fields terminated by ',' stored as textfile location '/stock\_market\_case\_study/input'”**

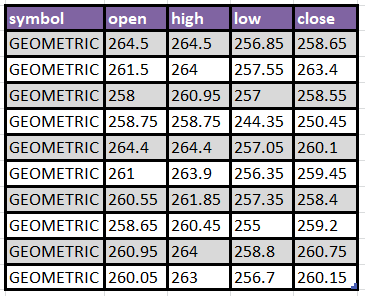
**hive -e**

**“select \* from  stock\_db.stock\_market\_data where lower(symbol) = 'geometric'” > /home/suparna/Desktop/stock\_market\_case\_study/hive/output1\_1.tsv**



**hive -e**

**“select symbol, open, high, low, close from stock\_db.stock\_market\_data where lower(symbol) = 'geometric'” > /home/suparna/Desktop/stock\_market\_case\_study/hive/output1\_2.tsv**



**hive -e**

**“select count(\*) as num\_rows from stock\_db.stock\_market\_data where lower(symbol) = 'geometric'” > /home/suparna/Desktop/stock\_market\_case\_study/hive/output1\_3.tsv**

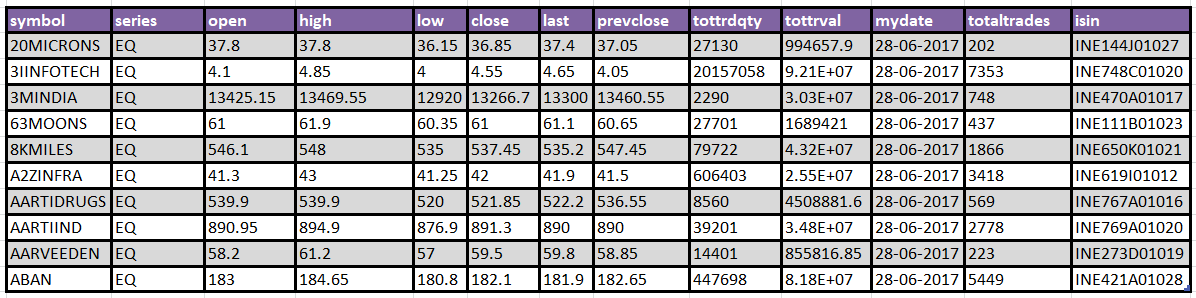


### 2. Calculation of various statistical quantities and decision making:

* **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**

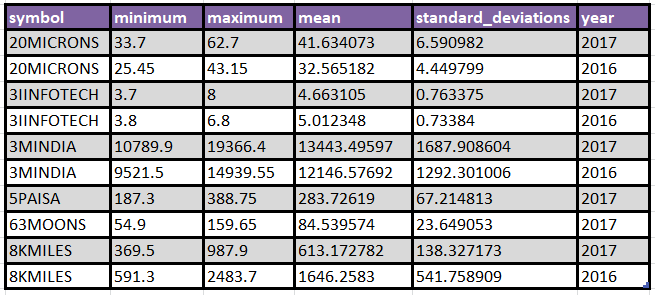


**hive -e**

**“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\_market\_data where lower(series) = 'eq' group by symbol, substr(mydate,1,4) order 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 delimited fields terminated by ',' stored as textfile location '/home/suparna/Desktop/stock\_market\_case\_study/hive/output2\_2.tsv'”**



### 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**

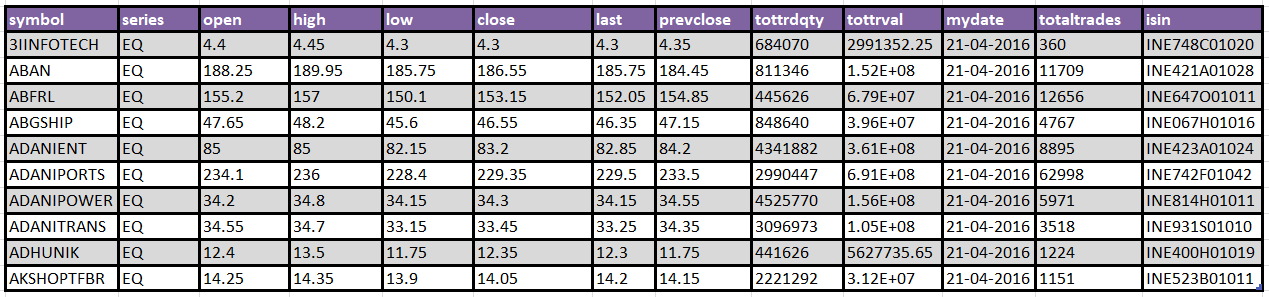
**“create table stock\_db.stock\_market\_data\_2016(symbol string, series string, open double, high double, low double, close double, last double, prevclose double, tottrdqty int, tottrval double, mydate string, totaltrades int, isin string)”**

**hive -e**

**“insert overwrite table stock\_db.stock\_market\_data\_2016 select \* from  stock\_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/Desktop/stock\_market\_case\_study/hive/output3\_1.tsv**



**hive -e**

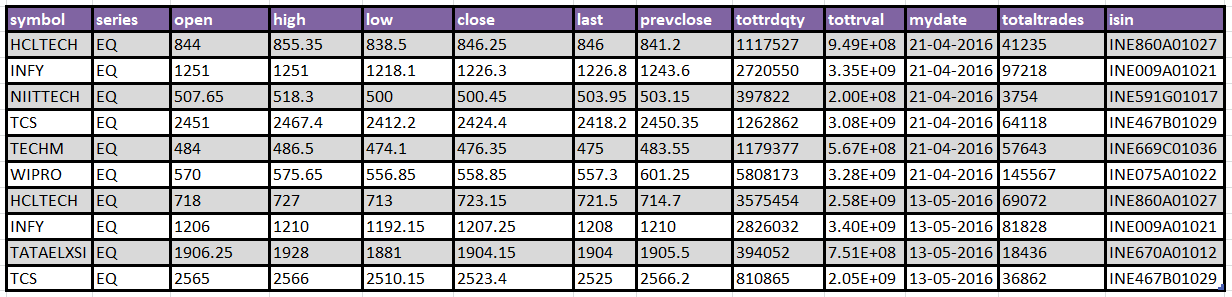
**“create table stock\_db.it\_companies\_stock (symbol string, series string, open double, high double, low double, close double, last double, prevclose double, tottrdqty int, tottrval double, mydate string, totaltrades int, isin string)”**

**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', 'ofss')”**

**hive -e**

**“select \* from stock\_db.it\_companies\_stock” > /home/suparna/Desktop/stock\_market\_case\_study/hive/output3\_2.tsv**



**hive -e**

**“create table stock\_db.it\_companies\_stock\_close (symbol1 string, close1 float, symbol2 string, close2 float,mydate string)”**

**hive -e**

**“insert overwrite table stock\_db.it\_companies\_stock\_close select t1.symbol,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\_companies\_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 string, corr float)”**

**hive -e**

**“insert overwrite table stock\_db.it\_pearsons\_corr\_coeff select symbol1, symbol2, (avg(close1\*close2) - (avg(close1) \*avg(close2)))/(stddev\_pop(close1) \* stddev\_pop(close2)) as pearsoncoefficient from stock\_db.it\_companies\_stock\_close group by symbol1, symbol2 order by pearsoncoefficient desc”**

**hive -e**

**“select \* from stock\_db.it\_pearsons\_corr\_coeff” > /home/suparna/Desktop/stock\_market\_case\_study/hive/output3\_3.tsv**

