

Big Data 230 : EMERGING TECHNOLOGIES IN BIG DATA PROJECT

STOCK ANALYSIS QUERIES

//Moving Files to VM

```
scp *.csv BigData230@52.183.35.71:/tmp
```

```
scp -P 2222 *.csv root@localhost:/tmp
```

//Creating Data Frames from files

```
val
```

```
s1=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tmp/daily_AAPL.csv").withColumn("Stock",lit("AAPL"))
```

```
val
```

```
s2=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tmp/daily_ACN.csv").withColumn("Stock",lit("ACN"))
```

```
val
```

```
s3=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tmp/daily_ADBE.csv").withColumn("Stock",lit("ADBE"))
```

```
val
```

```
s4=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tmp/daily_AMZN.csv").withColumn("Stock",lit("AMZN"))
```

```
val
```

```
s5=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tmp/daily_AVGO.csv").withColumn("Stock",lit("AVGO"))
```

```
val
```

```
s6=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tmp/daily_BABA.csv").withColumn("Stock",lit("BABA"))
```

```
val
```

```
s7=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tmp/daily_CRM.csv").withColumn("Stock",lit("CRM"))
```

```
val
```

```
s8=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tmp/daily_CSCO.csv").withColumn("Stock",lit("CSCO"))
```

```
val  
s9=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tmp/  
daily_EXPE.csv").withColumn("Stock",lit("EXPE"))
```

```
val  
s10=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tm  
p/daily_FB.csv").withColumn("Stock",lit("FB"))
```

```
val  
s11=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tm  
p/daily_GOOGL.csv").withColumn("Stock",lit("GOOGL"))
```

```
val  
s12=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tm  
p/daily_IBM.csv").withColumn("Stock",lit("IBM"))
```

```
val  
s13=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tm  
p/daily_INTC.csv").withColumn("Stock",lit("INTC"))
```

```
val  
s14=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tm  
p/daily_MSFT.csv").withColumn("Stock",lit("MSFT"))
```

```
val  
s15=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tm  
p/daily_MU.csv").withColumn("Stock",lit("MU"))
```

```
val  
s16=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tm  
p/daily_NFLX.csv").withColumn("Stock",lit("NFLX"))
```

```
val  
s17=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tm  
p/daily_NVDA.csv").withColumn("Stock",lit("NVDA"))
```

```
val  
s18=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tm  
p/daily_ORCL.csv").withColumn("Stock",lit("ORCL"))
```

```
val
s19=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tmp/daily_sbux.csv").withColumn("Stock",lit("SBUX"))
```

```
val
s20=spark.read.format("csv").option("header","true").option("inferSchema","true").load("file:///tmp/daily_TXN.csv").withColumn("Stock",lit("TXN"))
```

```
val StockData =
s1.unionAll(s2).unionAll(s3).unionAll(s4).unionAll(s5).unionAll(s6).unionAll(s7).unionAll(s8).unionAll(s9).unionAll(s10).unionAll(s11).unionAll(s12).unionAll(s13).unionAll(s14).unionAll(s15).unionAll(s16).unionAll(s17).unionAll(s18).unionAll(s19).unionAll(s20)
```

//Writing to parquet file

```
StockData.write.parquet("hdfs:///tmp/AllStockData.parquet")
```

```
val StockDataDF= spark.read.parquet("hdfs:///tmp/AllStockData.parquet")
```

```
StockDataDF.show(5)
```

```
+-----+-----+-----+-----+-----+-----+
| timestamp| open| high| low| close| volume|Stock|
+-----+-----+-----+-----+-----+-----+
|2019-08-16 00:00:00| 133.04| 134.59|130.25|133.76|2896600| IBM|
|2019-08-15 00:00:00| 131.22| 132.37|130.25|131.92|3713800| IBM|
|2019-08-14 00:00:00| 133.91|134.154|130.98|131.25|5114200| IBM|
|2019-08-13 00:00:00|133.7909| 136.57|132.81|135.79|4522100| IBM|
|2019-08-12 00:00:00| 135.66| 135.87|133.18|134.12|3895600| IBM|
+-----+-----+-----+-----+-----+-----+
```

only showing top 5 rows

Q1 : . Tech stock with max movement in 20 years

```
StockDataDF.createOrReplaceTempView("StockData")
```

```
spark.sql("select * from StockData where Stock = 'SBUX']").show(3)
```

```
val StockReturn =
```

```
StockDataDF.groupBy($"Stock").agg(max($"high").as("max_stock"),min($"low").as("min_stock")).withColumn("sold",coalesce($"max_stock"-$"min_stock",lit(0.0))).withColumn("Return",coalesce($"sold"*100/$"min_stock",lit(0.0)))
```

```
StockReturn.orderBy($"Return"desc).show
```

Stock	max_stock	min_stock	sold	Return
AMZN	2050.5	5.51	2044.99	37114.15607985481
NFLX	716.16	4.85	711.31	14666.18556701031
MU	145.313	1.59	143.72299999999998	9039.182389937107
AAPL	705.07	12.72	692.35	5443.003144654088
NVDA	292.76	5.75	287.01	4991.478260869565
EXPE	161.0	6.0	155.0	2583.3333333333335
AVGO	323.2	14.33	308.87	2155.4082344731332
CRM	187.94	9.0	178.94	1988.2222222222222
ADBE	313.11	15.7	297.41	1894.331210191083
CSCO	146.75	8.12	138.63	1707.266009852217
ACN	197.47	11.3	186.17	1647.5221238938052
SBUX	99.72	7.06	92.66	1312.4645892351275
GOOGL	1296.975	95.96	1201.0149999999999	1251.5787828261775
FB	218.62	17.55	201.07	1145.6980056980055
TXN	130.37	13.1	117.27000000000001	895.1908396946567
MSFT	141.68	14.87	126.81	852.7908540685945
ORCL	60.5	7.25	53.25	734.4827586206897
INTC	75.828	12.05	63.778000000000006	529.2780082987551
IBM	215.9	54.0119	161.8881	299.7267268879636
BABA	211.7	57.2	154.5	270.1048951048951

Q2: Which year did each of the individual stocks have the most change (+ or -) in their price

```
scala> val StockDataDF= spark.read.parquet("hdfs:///tmp/AllStockData.parquet")
```

```
StockDataDF: org.apache.spark.sql.DataFrame = [timestamp: timestamp, open: double ... 5 more fields]
```

```
scala> StockDataDF.createOrReplaceTempView("StockData")
```

```
scala> val YearlyDataDF = spark.sql("select max(high) as maxhigh, min(low) as minlow, Stock, year(timestamp) as yearstock from StockData group by Stock, yearstock order by yearstock")
```

```
YearlyDataDF: org.apache.spark.sql.DataFrame = [maxhigh: double, minlow: double ... 2 more fields]
```

```
scala> YearlyDataDF.createOrReplaceTempView("YearData")
```

```
scala> val maxYearlyFluctuationDF = spark.sql("select max((maxhigh - minlow)/minlow) as  
range, Stock from YearData group by Stock")
```

```
maxYearlyFluctuationDF: org.apache.spark.sql.DataFrame = [range: double, Stock: string]
```

```
scala> maxYearlyFluctuationDF.createOrReplaceTempView("maxYearlyFluctuation")
```

```
scala> val maxYearChanges = spark.sql("select y.*, range from YearData as y join  
maxYearlyFluctuation as m on y.Stock = m.Stock where (y.maxhigh - y.minlow)/y.minlow =  
m.range").show(50)
```

maxhigh	minlow	Stock	yearstock	range
91.5	14.875	AMZN	2000	5.151260504201681
170.188	53.438	ADBE	2000	2.1847748792993746
99.781	35.0	TXN	2000	1.8508857142857145
146.75	35.156	CSCO	2000	3.17425190579133
150.375	13.625	AAPL	2000	10.036697247706423
118.625	40.25	MSFT	2000	1.9472049689440993
35.0	10.16	ORCL	2001	2.4448818897637796
51.313	13.46	SBUX	2001	2.8122585438335808
72.66	7.2	NVDA	2002	9.091666666666665
126.39	54.0119	IBM	2002	1.3400398801004965
36.78	12.95	INTC	2002	1.8401544401544403
79.54	9.25	NFLX	2004	7.59891891891892
31.88	6.0	EXPE	2008	4.3133333333333335
697.3485	257.44	GOOGL	2008	1.7087806867619637
8.97	1.59	MU	2008	4.6415094339622645
51.43	17.74	ACN	2010	1.899098083427283
58.58	22.67	FB	2013	1.5840317600352887
187.94	36.09	CRM	2013	4.207536713771127
105.0	51.89	AVGO	2014	1.0235112738485257
191.75	88.0849	BABA	2017	1.1768770810888132