E10-2

This Notebook is about using SPARK Dataframe functions to process nsedata.csv.

Problem

- Write SPARK code to solve the problem stated at the end this Notebook (**do not use the createTempView function!**)

Submission

Create and upload a PDF of this Notebook after completing your assignment. **BEFORE CONVERTING TO PDF and UPLOADING ENSURE THAT YOU REMOVE / TRIM LENGTHY DEBUG OUTPUTS** . Short debug outputs of up to 5 lines are acceptable.

In [1]:
```
import findspark
findspark.init()
```

In [2]:
```
import pyspark
from pyspark.sql.types import *
```

In [3]:
```
sc = pyspark.SparkContext(appName="E10-2")
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel
(newLevel).
23/10/31 16:33:32 WARN NativeCodeLoader: Unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
```

In [4]:
```
ss = pyspark.sql.SparkSession(sc)
```

In [5]:
```
dfr = ss.read
```

In [6]:
```
schemaStruct = StructType()
schemaStruct.add("SYMBOL", StringType(), True)
schemaStruct.add("SERIES", StringType(), True)
schemaStruct.add("OPEN", DoubleType(), True)
schemaStruct.add("HIGH", DoubleType(), True)
schemaStruct.add("LOW", DoubleType(), True)
schemaStruct.add("CLOSE", DoubleType(), True)
schemaStruct.add("LAST", DoubleType(), True)
schemaStruct.add("PREVCLOSE", DoubleType(), True)
schemaStruct.add("TOTTRDQTY", LongType(), True)
schemaStruct.add("TOTTRDVAL", DoubleType(), True)
schemaStruct.add("TIMESTAMP", StringType(), True)
schemaStruct.add("ADDNL", StringType(), True)
```

```
Out[6]: StructType([StructField('SYMBOL', StringType(), True), StructField('SERIES', St
        ringType(), True), StructField('OPEN', DoubleType(), True), StructField('HIGH',
        DoubleType(), True), StructField('LOW', DoubleType(), True), StructField('CLOS
        E', DoubleType(), True), StructField('LAST', DoubleType(), True), StructField
        ('PREVCLOSE', DoubleType(), True), StructField('TOTTRDQTY', LongType(), True),
        StructField('TOTTRDVAL', DoubleType(), True), StructField('TIMESTAMP', StringTy
        pe(), True), StructField('ADDNL', StringType(), True)])
```

```
In [7]: df = dfr.csv("/home/hduser/spark/nsedata.csv", schema=schemaStruct, header=True)
```

```
In [8]: df.printSchema()
```

```
root
 |-- SYMBOL: string (nullable = true)
 |-- SERIES: string (nullable = true)
 |-- OPEN: double (nullable = true)
 |-- HIGH: double (nullable = true)
 |-- LOW: double (nullable = true)
 |-- CLOSE: double (nullable = true)
 |-- LAST: double (nullable = true)
 |-- PREVCLOSE: double (nullable = true)
 |-- TOTTRDQTY: long (nullable = true)
 |-- TOTTRDVAL: double (nullable = true)
 |-- TIMESTAMP: string (nullable = true)
 |-- ADDNL: string (nullable = true)
```

```
In [9]: from pyspark.sql.functions import col, date_format, to_date

        df1 = df.withColumn("TIMESTAMP2", date_format(to_date(col("TIMESTAMP"), "dd-MMM-
```

```
In [10]: df1.printSchema()
```

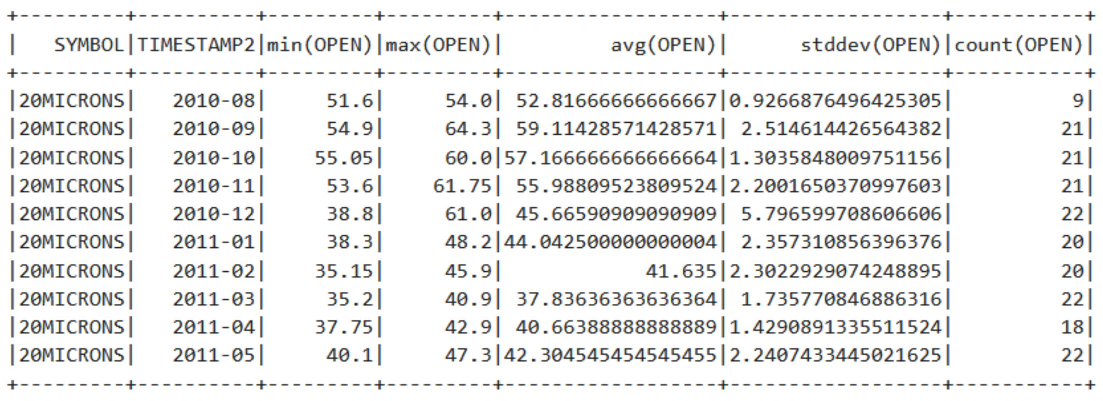
```
root
 |-- SYMBOL: string (nullable = true)
 |-- SERIES: string (nullable = true)
 |-- OPEN: double (nullable = true)
 |-- HIGH: double (nullable = true)
 |-- LOW: double (nullable = true)
 |-- CLOSE: double (nullable = true)
 |-- LAST: double (nullable = true)
 |-- PREVCLOSE: double (nullable = true)
 |-- TOTTRDQTY: long (nullable = true)
 |-- TOTTRDVAL: double (nullable = true)
 |-- TIMESTAMP: string (nullable = true)
 |-- ADDNL: string (nullable = true)
 |-- TIMESTAMP2: string (nullable = true)
```

Problem Statement

Using SPARK Dataframe functions write code to create the data shown below for all the traded companies. Save this data in an output file in **ascending order** of the company names, year and month.

SYMBOL | Month-Year | min(CLOSE) | max(CLOSE) | avg(CLOSE) | stddev(CLOSE) | tradedCount |

The output should appear as follows



```
+---------+----------+---------+---------+------------------+------------------+-----------+
|   SYMBOL|TIMESTAMP2|min(OPEN)|max(OPEN)|         avg(OPEN)|      stddev(OPEN)|count(OPEN)|
+---------+----------+---------+---------+------------------+------------------+-----------+
|20MICRONS|   2010-08|     51.6|     54.0| 52.81666666666667|0.9266876496425305|          9|
|20MICRONS|   2010-09|     54.9|     64.3| 59.11428571428571| 2.514614426564382|         21|
|20MICRONS|   2010-10|    55.05|     60.0|57.166666666666664|1.3035848009751156|         21|
|20MICRONS|   2010-11|     53.6|    61.75| 55.98809523809524|2.2001650370997603|         21|
|20MICRONS|   2010-12|     38.8|     61.0| 45.66590909090909| 5.796599708606606|         22|
|20MICRONS|   2011-01|     38.3|     48.2|44.042500000000004| 2.357310856396376|         20|
|20MICRONS|   2011-02|    35.15|     45.9|            41.635|2.3022929074248895|         20|
|20MICRONS|   2011-03|     35.2|     40.9| 37.83636363636364| 1.735770846886316|         22|
|20MICRONS|   2011-04|    37.75|     42.9| 40.66388888888889|1.4290891335511524|         18|
|20MICRONS|   2011-05|     40.1|     47.3|42.304545454545455|2.2407433445021625|         22|
+---------+----------+---------+---------+------------------+------------------+-----------+
```

tradedCount = number of times the company shares have been traded in that month

Notes and Hints:

- use the functions **groupBy** (based on SYMBOL and TIMESTAMP2) and **agg** to create the individual statistics like min, max, avg, etc.
- use **join** (based on SYMBOL and TIMESTAMP2) to combine the individual dataframes into a single table

This is just one method of solving the problem! You can discover of any other method, using any other combination of Dataframe functions-

```
In [11]: from pyspark.sql.functions import col, date_format, min, max, avg, stddev, count
```

```
In [12]: result_df = df1.groupBy("SYMBOL", "TIMESTAMP2").agg(
             date_format("TIMESTAMP2", "yyyy-MM").alias("Month-Year"),
             min("CLOSE").alias("min(CLOSE)"),
             max("CLOSE").alias("max(CLOSE)"),
             avg("CLOSE").alias("avg(CLOSE)"),
             stddev("CLOSE").alias("stddev(CLOSE)"),
             count("CLOSE").alias("tradedCount")
         )
```

```
In [13]: result_df = result_df.orderBy("SYMBOL", "TIMESTAMP2")
```

```
In [14]: result_df.coalesce(1).write.csv("/home/hduser/spark/company_stats", header=True,
```

```
In [15]: result_df.show(25)
```

```
+---------+---------+----------+---------+---------+------------------+-------
-----------+-----------+
|   SYMBOL|TIMESTAMP2|Month-Year|min(CLOSE)|max(CLOSE)|        avg(CLOSE)|     st
ddev(CLOSE)|tradedCount|
+---------+---------+----------+---------+---------+------------------+-------
-----------+-----------+
|20MICRONS|   2010-08|   2010-08|     51.55|      54.3|             52.75|1.06477
69719523452|          9|
|20MICRONS|   2010-09|   2010-09|      54.9|      60.9|  58.4547619047619|1.72691
23285436907|         21|
|20MICRONS|   2010-10|   2010-10|     54.35|     58.55| 56.37857142857143|0.89492
61741299467|         21|
|20MICRONS|   2010-11|   2010-11|     53.35|      60.3| 55.69047619047619|1.82801
93549043067|         21|
|20MICRONS|   2010-12|   2010-12|     36.85|      56.0|44.845454545454544| 4.8190
57136859004|         22|
|20MICRONS|   2011-01|   2011-01|      41.3|     47.75|43.917500000000004|1.98926
56010646148|         20|
|20MICRONS|   2011-02|   2011-02|      37.4|     44.95| 41.49249999999999|2.00317
34691155494|         20|
|20MICRONS|   2011-03|   2011-03|     35.85|      40.1| 37.70227272727272|1.35246
48813966484|         22|
|20MICRONS|   2011-04|   2011-04|     37.45|     41.65|            40.425|1.05819
63011486593|         18|
|20MICRONS|   2011-05|   2011-05|     40.15|      51.1| 43.17045454545455|2.69460
03807839873|         22|
|20MICRONS|   2011-06|   2011-06|      46.0|      50.5| 47.73636363636364|1.08914
35930279205|         22|
|20MICRONS|   2011-07|   2011-07|      46.8|     53.75| 49.30476190476191|1.83056
71772639778|         21|
|20MICRONS|   2011-08|   2011-08|      46.6|      54.7| 51.05952380952381|2.28667
98555776837|         21|
|20MICRONS|   2011-09|   2011-09|     51.85|     59.35|56.145238095238106|1.77566
21836588724|         21|
|20MICRONS|   2011-10|   2011-10|      52.0|     69.95| 58.51052631578948| 5.5576
66850281101|         19|
|20MICRONS|   2011-11|   2011-11|     56.15|     66.95|           61.3625| 4.3740
22447178467|         20|
|20MICRONS|   2011-12|   2011-12|     56.55|      63.6| 59.39523809523809|1.91232
21983954974|         21|
|20MICRONS|   2012-01|   2012-01|      61.7|      66.0| 63.04523809523808|1.22340
76141974053|         21|
|20MICRONS|   2012-02|   2012-02|      63.7|     77.55|             75.03| 3.8406
20837970422|         20|
|20MICRONS|   2012-03|   2012-03|     79.05|      85.1| 81.84761904761905| 1.5780
11105036673|         21|
|20MICRONS|   2012-04|   2012-04|     80.35|      91.5| 84.16578947368421| 3.6019
69084618818|         19|
|20MICRONS|   2012-05|   2012-05|     84.35|      93.1| 88.00227272727273|2.73576
30289645605|         22|
|20MICRONS|   2012-06|   2012-06|      84.7|     89.95| 87.38095238095237|1.74645
90025589037|         21|
|20MICRONS|   2012-07|   2012-07|     84.45|      97.7| 92.38181818181819|4.42677
31302067475|         22|
|20MICRONS|   2012-08|   2012-08|     96.85|    120.85|114.29999999999998| 7.2407
00932920781|         21|
+---------+---------+----------+---------+---------+------------------+-------
-----------+-----------+
only showing top 25 rows
```

In [ ]:
```
ss.stop()
sc.stop()
```

In [ ]:
```
```