

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 setL  
ogLevel(newLevel).  
23/10/31 12:31:30 WARN NativeCodeLoader: Unable to load native-hadoop libr  
ary for your platform... using builtin-java classes where applicable  
23/10/31 12:31:31 WARN Utils: Service 'SparkUI' could not bind on port 404  
0. Attempting port 4041.
```

```
In [4]: ss = pyspark.sql.Session(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('SERIE
S', StringType(), True), StructField('OPEN', DoubleType(), True), StructFi
eld('HIGH', DoubleType(), True), StructField('LOW', DoubleType(), True), S
tructField('CLOSE', DoubleType(), True), StructField('LAST', DoubleType(),
True), StructField('PREVCLOSE', DoubleType(), True), StructField('TOTTRDQT
Y', LongType(), True), StructField('TOTTRDVAL', DoubleType(), True), Struc
tField('TIMESTAMP', StringType(), True), StructField('ADDNL', StringType
(), True)])
```

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

```
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
```

```
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 [18]: from pyspark.sql import functions as F
```

```
result_df = df1.groupBy("SYMBOL", "TIMESTAMP2").agg(  
    F.min(col("CLOSE")).alias("min(CLOSE)"),  
    F.max(col("CLOSE")).alias("max(CLOSE)"),  
    F.avg(col("CLOSE")).alias("avg(CLOSE)"),  
    F.stddev(col("CLOSE")).alias("stddev(CLOSE)")  
)
```

```
In [19]: result_df = result_df.join(  
    df1.groupBy("SYMBOL", "TIMESTAMP2")  
        .agg(F.count("*").alias("tradedCount")),  
    on=["SYMBOL", "TIMESTAMP2"]  
)
```

```
In [20]: # Sort the result DataFrame in ascending order by company name and month-ye  
result_df = result_df.orderBy("SYMBOL", "TIMESTAMP2")
```

```
In [22]: # Save the result as a CSV file  
result_df.coalesce(1).write.csv("outputs.csv", header=True)
```

```
In [23]: ss.stop()  
sc.stop()
```

```
In [ ]:
```