Module 6: Assignment – 1

Loading the data

```
scala> val stocks = (spark
     | .read
     | .format("csv")
     | .option("inferSchema","true")
     | .option("header","true")
     | .load("file:///home/bitnami/sparkdata/NSE_RELIANCE_5_1.csv"))
stocks: org.apache.spark.sql.DataFrame = [time: timestamp, open: double ... 14 more fields]
```

```
scala> val stockDF = stocks.select("time","open","high","low","close")
stockDF: org.apache.spark.sql.DataFrame = [time: timestamp, open: double ... 3 more fields]
```

```
scala> stockDF.show(5)
+-------------------+----------+----------+----------+----------+
|               time|      open|      high|       low|     close|
+-------------------+----------+----------+----------+----------+
|2020-04-03 07:20:00|1053.771557|1056.297519|1051.790411|1053.424857|
|2020-04-03 07:25:00|1053.573443|1055.455532|  1053.3258|1054.266844|
|2020-04-03 07:30:00|1053.969672|1057.535735|1053.078156|1056.941391|
|2020-04-03 07:35:00|1056.842334|1057.931964|1055.009774|1056.941391|
|2020-04-03 07:40:00|1056.941391|1061.250384|1056.644219|1058.328193|
+-------------------+----------+----------+----------+----------+
only showing top 5 rows
```

Creating a View

```
scala> stockDF.createOrReplaceTempView("tblStock")

scala> spark.sql("select * from tblStock").show(5)
+-------------------+----------+----------+----------+----------+
|               time|      open|      high|       low|     close|
+-------------------+----------+----------+----------+----------+
|2020-04-03 07:20:00|1053.771557|1056.297519|1051.790411|1053.424857|
|2020-04-03 07:25:00|1053.573443|1055.455532|  1053.3258|1054.266844|
|2020-04-03 07:30:00|1053.969672|1057.535735|1053.078156|1056.941391|
|2020-04-03 07:35:00|1056.842334|1057.931964|1055.009774|1056.941391|
|2020-04-03 07:40:00|1056.941391|1061.250384|1056.644219|1058.328193|
+-------------------+----------+----------+----------+----------+
only showing top 5 rows
```

1. Find out the average ‘close’ price of Reliance throughout the duration of the dataset

Solution:

```
scala> spark.sql("""
     | select round(avg(close),2) avg_close_price
     | from tblStocks""").show
+---------------+
|avg_close_price|
+---------------+
|        1414.25|
+---------------+
```

2. If a Reliance stock was bought at the beginning of the trading day, '2020-04-07' (YYYY-MM-DD), at the close price of the first 5-minute window, scan the dataset to find out the point to sell the stock to maximize profits. You are required to print the specific timestamp

Solution:

```
scala> spark.sql("""
     | select time, max(high) as max_high
     | from tblStocks
     | group by time
     | order by max_high desc limit 1""").show
+-------------------+--------+
|               time|max_high|
+-------------------+--------+
|2020-06-05 03:45:00|  1618.0|
+-------------------+--------+


scala> spark.sql("""
     | select time, max(close) as max_close
     | from tblStocks
     | group by time
     | order by max_close desc limit 1""").show
+-------------------+---------+
|               time|max_close|
+-------------------+---------+
|2020-06-05 03:55:00|   1610.0|
+-------------------+---------+
```

If we go by highest price within a 5 minute interval to calculate maximum profit that can be made then:

Maximum price for within a 5 minute interval was reached on 2020-06-05 at 03:45:00, and thus, the maximum profit would be:

1618-1095 = Rs. 523

If we go by maximum closing price to calculate maximum profit that can be made then:

Maximum overall closing price was reached on 2020-06-05 at 03:55:00, and thus, the maximum profit would be:
1610-1095 = Rs.515

3. Find out the net profit or net loss to be accumulated if one stock of Reliance is bought at the opening of every 5-minute slot and sold at the lowest possible point in that 5- minute slot

Solution:

```
scala> spark.sql("""
     | select time, open as buying_price, low as lowest_selling_price,
     | round(((low)-(open)),2) as profit_or_loss
     | from tblStocks
     | group by time, buying_price, lowest_selling_price
     | order by time limit 20""").show
+-------------------+------------+--------------------+--------------+
|               time|buying_price|lowest_selling_price|profit_or_loss|
+-------------------+------------+--------------------+--------------+
|2020-04-03 07:20:00| 1053.771557|         1051.790411|         -1.98|
|2020-04-03 07:25:00| 1053.573443|           1053.3258|         -0.25|
|2020-04-03 07:30:00| 1053.969672|         1053.078156|         -0.89|
|2020-04-03 07:35:00| 1056.842334|         1055.009774|         -1.83|
|2020-04-03 07:40:00| 1056.941391|         1056.644219|          -0.3|
|2020-04-03 07:45:00| 1058.328193|         1057.238563|         -1.09|
|2020-04-03 07:50:00| 1057.931964|         1056.495633|         -1.44|
|2020-04-03 07:55:00| 1057.089977|          1055.15836|         -1.93|
|2020-04-03 08:00:00| 1057.931964|         1057.139506|         -0.79|
|2020-04-03 08:05:00| 1058.031021|         1057.981493|         -0.05|
|2020-04-03 08:10:00| 1067.490993|         1064.519274|         -2.97|
|2020-04-03 08:15:00|  1066.50042|         1064.073517|         -2.43|
|2020-04-03 08:20:00| 1065.311733|         1058.922537|         -6.39|
|2020-04-03 08:25:00| 1058.922537|         1055.554589|         -3.37|
|2020-04-03 08:30:00| 1059.714995|         1057.832907|         -1.88|
|2020-04-03 08:35:00| 1060.061696|         1057.238563|         -2.82|
|2020-04-03 08:40:00| 1057.931964|         1051.988526|         -5.94|
|2020-04-03 08:45:00| 1057.436678|         1053.424857|         -4.01|
|2020-04-03 08:50:00| 1057.189034|         1055.653646|         -1.54|
|2020-04-03 08:55:00| 1057.931964|         1057.634792|          -0.3|
+-------------------+------------+--------------------+--------------+
```

4. Find out the net profit or net loss to be accumulated if one stock of Reliance is bought at the opening of every 5-minute slot and sold at the highest possible point in that 5- minute slot

Solution:

```
scala> spark.sql("""
     | select time, open as buying_price, high as highest_selling_price,
     | round(((high)-(open)),2) as profit_or_loss
     | from tblStocks
     | group by time, buying_price, highest_selling_price
     | order by time limit 20""").show
+-------------------+------------+---------------------+--------------+
|               time|buying_price|highest_selling_price|profit_or_loss|
+-------------------+------------+---------------------+--------------+
|2020-04-03 07:20:00| 1053.771557|          1056.297519|          2.53|
|2020-04-03 07:25:00| 1053.573443|          1055.455532|          1.88|
|2020-04-03 07:30:00| 1053.969672|          1057.535735|          3.57|
|2020-04-03 07:35:00| 1056.842334|          1057.931964|          1.09|
|2020-04-03 07:40:00| 1056.941391|          1061.250384|          4.31|
|2020-04-03 07:45:00| 1058.328193|          1060.507454|          2.18|
|2020-04-03 07:50:00| 1057.931964|          1059.764524|          1.83|
|2020-04-03 07:55:00| 1057.089977|          1058.625365|          1.54|
|2020-04-03 08:00:00| 1057.931964|          1059.714995|          1.78|
|2020-04-03 08:05:00| 1058.031021|           1069.81884|         11.79|
|2020-04-03 08:10:00| 1067.490993|          1074.771705|          7.28|
|2020-04-03 08:15:00|  1066.50042|          1069.323554|          2.82|
|2020-04-03 08:20:00| 1065.311733|          1067.441465|          2.13|
|2020-04-03 08:25:00| 1058.922537|          1062.736243|          3.81|
|2020-04-03 08:30:00| 1059.714995|          1061.795199|          2.08|
|2020-04-03 08:35:00| 1060.061696|          1061.696141|          1.63|
|2020-04-03 08:40:00| 1057.931964|           1058.82348|          0.89|
|2020-04-03 08:45:00| 1057.436678|          1059.566409|          2.13|
|2020-04-03 08:50:00| 1057.189034|          1059.516881|          2.33|
|2020-04-03 08:55:00| 1057.931964|          1061.696141|          3.76|
+-------------------+------------+---------------------+--------------+
```