Advanced Data Analysis, Second Assignment

Deadline: 11 November, 2020, 23h59
Late delivery policy: no late delivery allowed (excluding medical or exceptional situations). You are granted two grace days that you can use in the following three assignments, at your discretion.

Data files

data.txt: The file data.txt contains lines of text with five columns of data separated by a space. The first three elements are the year, month and day for the date of each data point. The fourth column is the closing time value of the NASDAQ Composite index (in US dollars) and the fifth column is the price of a barrel crude oil (in US dollars) for the Crude Oil Prices, West Texas Intermediate.

data_new.txt: The same as file data.txt, with an extra first line, with the names of columns.

eurusd.txt: The file eurusd.txt contains two columns of data separated by a tab ('\t'). The first column is a date, in the format dd-MM-yyyy (e.g. 30-01-2020), and the second column is the EUR USD exchange rate.

NOTES:

The data files only includes business days, and have no values for holidays or weekends. For these exercises we will pretend that only business days exist, since on holidays and weekends trading tends to be suspended. This means you can ignore the gaps and process the data as if all days were consecutive.

The data files have different initial and final dates. You should not remove entries from any of the files.

Your answers to questions 1 to 4 should be submitted as a notebook; the answer to question 5 can be included as text in the notebook, or as a pdf file.

Question 1 [5 points out of 20]

Write code, using Spark RDD interface, to print out the answers to the following questions:

1.a) How many days with data do you have for each year, for the years 2016 through 2019?

1.b) What is the minimum price of crude oil (fifth column) for each year from 2015 through 2020?

1.c) What was the date (year, month, day) when the crude oil price fell to the lowest value?

1.d) What is the average number of days with data for the years 2016 through 2019?

Question 2 [5 points out of 20]

Write code, using Spark SQL interface, to print out the answers to the following questions:

2.a) How many days with data do you have for each year, for the years 2016 through 2019?

2.b) What is the minimum price of crude oil (fifth column) for each year from 2015 through 2020?

2.c) What was the date (year, month, day) when the crude oil price fell to the lowest value?

2.d) What is the average number of days with data for the years 2016 through 2019?

Question 3 [4 points out of 20]

Plot the data for the oil price from 2016 through 2019, along with a moving average starting from mid 2016 to the end of 2019, with the moving average being the average of the prior 250 business days.

Add a comment to your code or a text cell in your notebook discussing whether this solution is equivalent to having a moving average over the prior year period.

Question 4 [4 points out of 20]

The data in “data.txt” file is either in USD (oil price) or indexed to prices in USD (NASDAQ index). For an investor in Europe, it would be interesting to understand how these values evolve, if they were priced in Euros, taking into account the EUR/USD exchange rate.

4.a) Plot all the data available for the NASDAQ index, both the original value and the value adjusted to Euros.

4.b) For all data available for the oil price, plot the valorization (in %) from the first day, for both the price in USD and the price adjusted to Euros.

Question 5 [2 points out of 20]

For each Spark computation, it is possible to access the logical, optimized and physical execution plans. For example, the following code will output the execution plan for one of the queries presented in lecture 9.

spark.sql("SELECT sex, avg(height) FROM heights GROUP BY sex""").explain(mode="formatted")

The result is the following:

```
== Physical Plan ==
* HashAggregate (5)
+- Exchange (4)
   +- * HashAggregate (3)
      +- * Project (2)
         +- Scan csv  (1)


(1) Scan csv
Output [2]: [_c0#909, _c1#910]
Batched: false
Location: InMemoryFileIndex [file:/home/jovyan/work/heights.txt]
ReadSchema: struct<_c0:double,_c1:double>
```

```
(2) Project [codegen id : 1]
Output [2]: [_c0#909 AS sex#913, _c1#910 AS height#916]
Input [2]: [_c0#909, _c1#910]

(3) HashAggregate [codegen id : 1]
Input [2]: [sex#913, height#916]
Keys [1]: [knownfloatingpointnormalized(normalizenanandzero(sex#913)) A
S sex#913]
Functions [1]: [partial_avg(height#916)]
Aggregate Attributes [2]: [sum#1096, count#1097L]
Results [3]: [sex#913, sum#1098, count#1099L]

(4) Exchange
Input [3]: [sex#913, sum#1098, count#1099L]
Arguments: hashpartitioning(sex#913, 200), true, [id=#530]

(5) HashAggregate [codegen id : 2]
Input [3]: [sex#913, sum#1098, count#1099L]
Keys [1]: [sex#913]
Functions [1]: [avg(height#916)]
Aggregate Attributes [1]: [avg(height#916)#1092]
Results [2]: [sex#913, avg(height#916)#1092 AS avg(height)#1093]
```

Given this information, and your knowledge that Spark SQL computations are transformed into Spark RDD computation, explain how the given SQL code executes in a distributed fashion (something like what is presented in slide 15, 17, 19, 21 of lecture 8).