Distributed architectures for big data processing and analytics

PoliStocks is an international company that collects and analyzes stock data. To identify interesting stocks, PoliStocks computes a set of statistics based on the following dataset file.

- Stocks_Prices.txt
    - Stocks_Prices.txt is a text file containing the historical information about the last 10 years of prices of thousands of stocks on several international financial markets.
    - The sampling rate is 1 minute (i.e., every minute the system collects the prices of the stocks under analyses and a new line for each stock is appended at the end of Stocks_Prices.txt)
    - Each line of the input file has the following format
        - stockId,date,hour:minute,price

          where *stockId* is a stock identifier (e.g., GOOG) and *price* is a floating point number whose value indicates the price of the stock *stockId* at time *date,hour:minute.*
        - For example, the line

          GOOG,2015/05/21,15:05,45.32

          means that the price of stock ***GOOG*** on ***May 21, 2015*** at ***15:05*** was ***45.32€***

Exercise 2 – Spark (19 points)

The managers of PoliStocks are interested in (A) performing some analyses about the daily variations of each stock during year 2017 and (B) identifying "stable trends in year 2017" for each stock.

The managers of PoliStocks asked you to develop **one single application** to address all the analyses they are interested in. The application has the following arguments: the path of the input file Stocks_Prices.txt and two output folders (associated with the outputs of the following points A and B, respectively).

Specifically, design a single application based on Spark RDDs or Spark DataFrames, and write the corresponding code, to address the following points:

A. *Analyses of daily variations in year 2017*. Considering only the prices in year 2017, the application must compute for each stock the number of dates associated with a daily price variation greater than 10 euro. Given a stock and a date, the daily price variation of that stock in that date is given by the difference between the highest price and the lowest price reached by that stock in that specific date (highest price–lowest price). Store in the first HDFS output folder the set of "*stockId*,*number_of_dates_dailyVariation>10*", one line for each *stockId*, reporting **only** the stocks associated with **at least one date** in year 2017 with a daily price variation greater than 10 euro.

B. *Stable trends in year 2017 for each stock*. Considering only the prices in year 2017, for each stock, the application must select all the sequences of two consecutive dates characterized by a "*stable trend*". Given a stock and two consecutive dates, the trend of that stock for those two dates is classified as a "*stable trend*" if and only if the absolute difference between the daily price variations of the two dates is at most 0.1 euro (i.e., abs(daily price variation of the first date - daily price variation of the second date) <= 0.1). The application stores in the second HDFS output folder the full set of all stable trends, one per line. Specifically, each line contains the following information about one stable trend: stockId, first date of the stable trend. Each output line has the following format:

    stockId,first_date

For instance, suppose that stock GOOG has a daily variation equal to 2.03 in April 2, 2017 and a daily variation equal to 2.05 in April 3, 2017. It means that "GOOG,2017/04/02,2017/04/03" is classified as a "stable trend" because abs(2.03-2.05)<=0.1 and the following line is inserted in the output:

    GOOG,2017/04/02

Note that you can have many pairs of consecutive dates of stable trends for each stock, hence having multiple lines in the output file for each stock.
Sorting the results is not required.

Suppose that someone has already implemented the following function

- *previousDate(String date)*
    - The parameter of this function is a string representing a date (in the format yyyy/mm/dd). The returned value is a string representing the previous date
    - For example, the invocation

        yesterday=previousDate("2016/06/20")

      returns and stores "2016/06/19" in the variable *yesterday.*