Lab 6

In this lab, we continue our work on the Amazon dataset, using Apache Spark. Your task is similar to that of Lab 3: given the original Amazon food dataset (that you can find at /data/students/bigdata-01QYD/Lab3/Reviews.csv), find all the pairs of items frequently reviewed together.
In Ex1 you find the steps that you are required to perform on the dataset, that are similar to the one you already implemented in Hadoop.

Ex. 1

Write a single Spark application that:

- Transposes the original Amazon food dataset, obtaining a PairRDD of the type:
  <user_id> → <list of the product_ids reviewed by user_id>
- Counts the frequencies of all the pairs of products reviewed together;
- Writes on the output folder all the pairs of products that appear more than once and their frequencies. The pairs of products must be sorted by frequency.

The input Amazon food dataset (available in the HFDS shared folder of the BigData@Polito cluster: /data/students/bigdata-01QYD/Lab3/Reviews.csv) lists all the reviews per-row (one review per line), and is comma-separated. In each line, two of the columns represent the user id and product id. The schema of Reviews.csv is the following:
Id,ProductId,UserId,ProfileName,HelpfulnessNumerator,HelpfulnessDenominator,Score,Time,Summary,Text

Inspect the output of your application to search for interesting (or funny!) facts, and analyze the job execution as usual (performances, number of executors, etc…).

Bonus task

Extend the implemented application in order to write on the standard output the top 10, most frequent, pairs and their frequencies.

Note that Spark 1.6, or above, provides the following actions that can be applied on an RDD of type JavaRDD<T>:
1) List<T> top(int n, java.util.Comparator<T> comp)
2) List<T> takeOrdered (int n, java.util.Comparator<T> comp)

top returns the **n largest elements** of the RDD based on the specified Comparator

takeOrdered returns **the n smallest elements** of the RDD based on the specified Comparator