Big Data Analytics

Prof. Dr. Martin Theobald, Alessandro Temperoni & Jingjing Xu
Summer Semester 2022 – University of Luxembourg

Exercise Sheet #4 – Due on June 2, 2022 – 14:00 (before the lecture starts)

Please upload the source code of your solutions on or before the due date to the provided Moodle assignment as a single zip file using your group id as the file name. Provide some brief instructions on how to run your solution to each problem in a file called `Problem_X.txt`*, and report also the most important results (such as number of results, runtimes, etc.) of your solutions to that problem within this file. Remember that all solutions may be submitted in groups of up to 3 students.*

***Note:* Since this exercise sheet imposes intensive computational workloads, it is highly recommended to run the following experiments on the IRIS HPC cluster by using either the interactive or batching modes of the Spark launcher scripts (see once more the "Getting Started" guide provided on Moodle for detailed instructions).**

Geo-Spatial & Temporal Data Analysis — 12 Points

Problem 1. Consider the NYC taxi trips and NYC boroughs GeoJson data sets which are available from Moodle. Also consider the shell script `RunTaxiTrips-shell.scala` from Moodle as a basis to perform the following analytical tasks.

Note: to aggregate multiple taxi trips (e.g., by their duration) please use instances of Spark's `StatCounter` class and `merge` them according to the grouping conditions you wish to apply to the trips (similarly to the provided Moodle script).

(a) Compute the count, sum and average duration of all taxi trips which *started and ended in the same NYC borough* over the entire period of time recorded in the data set. **2 Points**

(b) Compute the count, sum and average duration of all taxi trips which *started and ended in a different NYC borough* over the entire period of time recorded in the data set. **2 Points**

(c) Repeat steps (a) and (b), but this time return one such statistic for each borough individually (*Brooklyn*, *Manhattan*, ...) over the entire period of time recorded in the data set. Which is thus the busiest borough in NYC? **2 Points**

(d) Repeat steps (a) and (b), but this time return one such statistic for each day-of-the-week (*Monday*, *Tuesday*, ..., *Sunday*) over the entire period of time recorded in the data set. Which is thus the busiest day-of-the-week in NYC? **2 Points**

(e) Modify the provided script such that it indeed computes the *average duration between two subsequent trips* conducted by the same taxi driver *per NYC borough* and *per hour-of-the-day* (at which the first trip starts). **2 Points**

Note that the current script computes these statistics only per borough. That is, please add the hour-of-the-day (at which the first trip starts) as an additional grouping condition to the provided Moodle script.

(f) Let us assume we wish to detect typical *rush hours* in the NYC dataset. To do so, normalize all trip durations by the direct geo-spatial distance between their start and end points, and then compute the average normalized trip duration for each hour-of-the-day and across all taxi trips. **2 Points**

That is, for each taxi trip, compute its duration (e.g., in seconds) and divide this duration by the direct distance (e.g., in miles or kilometres) between the start and end point of this trip. Then again group the taxi trips according to the hour-of-the-day at which they started, compute their averages, and sort these averages (one for each hour) in descending order.

Hint: See the Esri geometry API (`http://esri.github.io/geometry-api-java/javadoc/`*) for a reference of the necessary distance operators.*

Please make sure that you properly make use of the Spark infrastructure by loading the taxi trips and their various transformations into RDDs and by computing the requested tasks as much as possible via parallel RDD transformations. Summarize the results of this experiment in your `Problem_1.txt` file.

Analyzing Traffic Safety Data **12 Points**

Problem 2. For this exercise, we will consider the "Motor Vehicle Collisions" open data collection (available from `https://data.cityofnewyork.us/` and via Moodle) from the New York Police Department, a CSV file which consists of records with information about collisions that occurred in New York City in 2013. Each line of this data set contains (amongst others) the following fields:

```
DATE, TIME, BOROUGH, LATITUDE, LONGITUDE, LOCATION, ON STREET NAME, CROSS
STREET NAME, OFF STREET NAME, NUMBER OF PERSONS INJURED, NUMBER OF PERSONS KILLED,
CONTRIBUTING FACTOR VEHICLE 1, CONTRIBUTING FACTOR VEHICLE 2,
VEHICLE TYPE CODE 1, VEHICLE TYPE CODE 2
```

(a) Define a new Scala class `Collision` with appropriate fields to store the above attributes (and only those). Consider the Scala function `parseCollisions` provided on Moodle, and parse the lines of the CSV file into an RDD whose elements are of type `Collision`. Also filter out useless or meaningless lines from your data set, i.e., remove all records where *any* of the above fields is empty. **2 Points**

(b) Find the most dangerous street crossings according to the number of people that were injured or even killed in collisions which are recorded within the data set. Return tuples of `(BOROUGH, ON STREET NAME, CROSS STREET NAME)` together with the number of people involved (either injured or killed) and a list of the 5 most common contributing factors (of either one of the two vehicles involved in a collision) for each such crossing. Sort all crossings in descending order of the total number of people involved in these accidents and report the top-25 most dangerous crossings. **2 Points**

(c) Repeat the procedure of (b) but instead of aggregating collisions by the street crossings, this time aggregate the collisions by the *day-of-the-week* (Mo-Su) and the *hour-of-the-day* (0-23) (i.e., 7x24 combinations overall) among all collisions reported for this period. Sort the aggregated collisions by the number of people involved to see what the most dangerous days/hours of the week/day are and report the top-25 most dangerous such times. **2 Points**

(d) Finally, let us investigate if any type of vehicle is particularly likely involved in accidents. Aggregate all accidents by their `VEHICLE TYPE CODE` (either 1 or 2) and sort the various vehicle types in descending order of the number of accidents they were involved in. Report the top-5 types of vehicles together with their numbers of accidents. **2 Points**

(e) Repeat the above steps (b)–(d), but this time compute the results for the *number of persons killed* and the *number of persons injured* individually. For each step, report the top-5 results which have the highest differences between the *number of persons killed* and the *number of persons injured*, respectively. **4 Points**

Please make sure that you properly make use of the Spark infrastructure by loading the collisions and their various transformations into RDDs and by computing the requested tasks as much as possible via parallel RDD transformations. Summarize the results of this experiment in your `Problem_2.txt` file.

Final Big Data Use-Case **12 Points**

Problem 3. As for the very last exercise of this term, you may implement a Big Data use-case of your

choice in either Spark or PySpark. You may select any openly available dataset (including the ones already discussed in the lecture) and apply any of the previously discussed analytical techniques (using MLlib, Spark-SQL, DataFrames, GraphX, etc.) to solve your use-case. Please clearly refer to the dataset you used and formulate your goals and objectives in your `Problem_3.txt` file.

Recommended resources for new datasets include `https://www.kaggle.com`, `http://snap.stanford.edu`, `https://www.kdnuggets.com/datasets/`, `https://github.com/CSSEGISandData/COVID-19`, and any others you may wish to explore. The effort for this should be comparable to a regular exercise. (Please however avoid downloading code, we will do some basic plagiarism checks on your solutions.)

Also here, grading will be based on the appropriateness and creativity of your solution. Please make sure that you properly make use of the Spark infrastructure by using RDDs and parallel RDD transformations as much as possible. Summarize also the results of this experiment in your `Problem_3.txt` file.