Laboratory 3 - Spark SQL

In this lab, we will analyze historical data about the usage of a bike sharing system of Barcelona. We will consider the occupancy of the stations where users can pick up or drop off bikes. Your task consists in identifying the most "critical" timeslots *(day of the week, hour)* for each station. You are requested to exploit different BigData approaches, by using the RDD approach seen in LAB 1 and 2 and by using Spark SQL and DataFrame data structure.

1. Input Data

The analysis is based on 2 files available in the HFDS shared folder of the BigData@Polito cluster:

1. `/data/students/bigdata_internet/lab3/register.csv`
2. `/data/students/bigdata_internet/lab3/stations.csv`

register.csv

It contains the historical information about the number of used and free slots for ~3000 stations from May 2008 to September 2008. Each line of register.csv corresponds to one reading about the situation of one station at a specific timestamp. Each line has the following format (`\t` stands for the tab character) :

```
station\ttimestamp\tused_slots\tfree_slots
```

For example, the line

```
23  2008-05-15 19:01:00 5   13
```

means that there were 5 used slots and 13 free slots at station 23 on May 15, 2008 at 19:01:00.

Hint: while reading a csv file as a Spark DataFrame you can specify in the options of `spark.read.load` the separator by using the parameter `sep` ("tab" corresponds to `\t`).

The first line of `register.csv` contains the header of the file. Pay attention that some of the lines of `register.csv` contain wrong data due to temporary problems of the monitoring system. Specifically, some lines are characterized by `used_slots = 0` and `free_slots = 0` . These lines must be filtered out before performing the analysis.

Questions: How many rows of data we obtain before and after the data cleaning above?

stations.csv

It contains the description of the stations. Each line of registers.csv has the following format:

```
id\tlongitude\tlatitude\tname
```

For example, the line

```
1   2.180019    41.397978   Gran Via Corts Catalanes
```

contains the information about station `1`. The coordinates of station `1` are `2.180019,41.397978` and its name is `Gran Via Corts Catalanes`.

2. Exercise

Write a single Spark application that selects the pairs `(station, timeslot)` that are characterized by a high "criticality" value. You are requested to **solve the problem by using two different sets of Spark APIs**.

- Implement **a first version** of the application by using RDDs.
- Implement **a second version** of the application based on the use of Spark DataFrames.

In this application, each pair `(day of the week, hour)` is a timeslot T_j and it is associated with all the readings associated with that pair, independently of the specific date. For instance, the timeslot `(Wednesday, 15)` corresponds to all the readings made on Wednesdays from 15:00:00 to 15:59:59.

Hint To handle columns containing timestamps, consider using either the `hour(timestamp)` and `date_format(timestamp,'EEEE')` of SQL or the `datetime` package of Python (e.g., functions `strptime()` and `strftime()`).

A station S_i from the set of stations S is in the critical state if the number of free slots is equal to 0 (i.e., the station if full). The "criticality" $C(S_i, T_j)$ of a station S_i in the timeslot T_j is defined as the fraction of readings with crical state with respect to the total:

$$C(S_i, T_j) = \frac{\text{Number of readings of } S_i \text{ in } T_j \text{ with free\_slots equal to } 0}{\text{Number of readings of } S_i \text{ in } T_j}$$

Write two versions (based on RDDs and DataFrames, respectively) of an application that:

- Computes the **criticality value** $C(S_i, T_j)$ for each pair (S_i, T_j).
- Selects only the **critical pairs** (S_i, T_j) having a criticality value $C(S_i, T_j)$ greater than a minimum threshold. The **minimum criticality threshold** is a float between 0 and 1 passed as an argument of the application.
- Order the results **by increasing criticality**. If there are two or more records characterized by the same criticality value, consider the station id value (in ascending order). If also the station is the same, consider the day of the week (ascending from Monday to Sunday) and finally the hour (ascending from 0 to 23).
- Store the sorted **critical pairs** $\hat{C}(S_i, T_j)$ in the output folder, by using a **csv files (with header)**, where columns are separated by "tab". Store exactly the following attributes separated by a "tab":
  - station
  - station longitude
  - station latitude
  - day of week
  - hour
  - criticality value

For example, a line of the output could be:

```
1   2.180019    41.397978   Tuesday 20  0.678
```

The `output_folder` must be an argument of the application.

Run the two versions of the application, with minimum criticality threshold equal to 0.6. **Question** How many critical pairs do you obtain? Report also the complete output result of the applications.

3. BONUS TASK

Compute the distance $d(S_i, center)$ between each station S_i and the city center. The city center has coordinates:

- latitude = 41.386904
- longitude = 2.169989

To compute the distance implement the Haversine function as a UDF (use the formula in https://en.wikipedia.org/wiki/Haversine_formula).
Then, compute the average number of used slot per station as:

$$U(S_i) = \text{Average among readings of used slots of } S_i$$

Finally, do the average of $U(S_i)$ among:

1. the stations S_i that are closer than 1.5 km from the center, i.e., $U1 = avg(U(S_i))$ where S_i such that $d(S_i, center) < 1.5km$
2. the stations S_i that are farther than 1.5 km from the center, i.e., $U2 = avg(U(S_i))$ where S_i such that $d(S_i, center) >= 1.5km$

Questions: Are more used the stations closer to center or the ones farther? Report the values obtained of U1 and U2.

How to write and submit the report

In your report, you must answer to all questions, report the code and scripts the that you have written, and show the output. . You are are required to comment each instruction (or group of instructions) - i.e., what it the goal of the piece of code. You must follow the order in which questions and exercises are posed.

Your report must be a PDF. It can be directly generated from a Jupyter notebook. Go on `File-> Export Notebook As ... -> Export Notebook As PDF` . It must be submitted through the Teaching Portal course page (didattica.polito.it).

Naming convention: The file must be named with the following schema:

```
s<id>_lab<n>.pdf
```

For example, if you student ID is 123456 and you are submitting lab 3, the file must be: `s123456_lab3.pdf` Reports with a wrong file name will **NOT** be considered.

Report lenght: the report length must not exceed 10 pages, otherwise it will **NOT** be considered valid.