Project Description

Please complete the following tasks.

1- Spark-RDDs: Scalable Close-Contacts Public Health Applications (25 points)

Overview: This problem studies different distance computations related to COVID-19 social distancing. Assume three data files containing points stored on HDFS (or locally), where each file corresponds to a set of two-dimensional data records (x,y) with both the X and Y dimensions ranging from 1 to 10,000 units. Each point also refers to a unique identifier and other attributes such as name, their age, etc. These points indicate the location of people seated at an outdoor concert as in Figure 1.

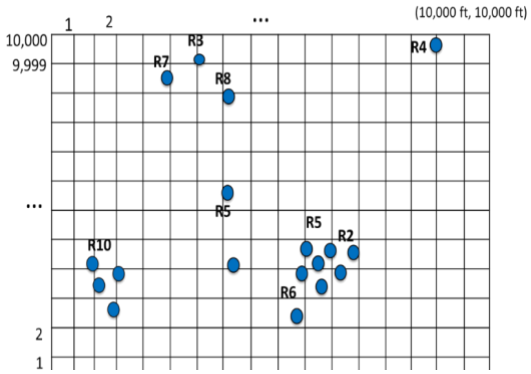

Figure 1: PEOPLE at a concert seated close to each other (2D Points).

Step 1 – Data Creation: (5 points)
Create three datasets called **PEOPLE-large**, **INFECTED-small** and **PEOPLE-SOME-INFECTED-large**, each composed of 2D points (x,y) and the aforementioned information, namely an id and a set of additional attributes. Specifically, each line in the file should contain a point in the format (id, x, y, attributes), where id denotes a unique identifier of that point, x denotes its value in the X axis, and y denotes its value in the Y axis, and attributes are a set of additional properties (you have freedom over those additional properties). **PEOPLE-SOME-INFECTED-large** contains one additional required attribute INFECTED, which could either have a value of "yes" or "no" to indicate infection status of that person .

You can choose a random function to create the points, and scale this dataset to be fairly large, say 100MB, but only to a size that your laptop/PC can handle. Otherwise, reduce the size of your data.

PEOPLE-large should be a large data file which includes everyone at the concert. However, you don't know who in this file (which entries) are infected with COVID or not.

INFECTED-small should have the exact same data schema as **PEOPLE-large**. However, everyone in this file is infected with COVID. It should be a (small) subset of **PEOPLE-large**.

PEOPLE-SOME-INFECTED-large is the same dataset as **PEOPLE-large**, with the only difference being that this dataset contains an additional column "INFECTED" which indicates whether a person is infected with COVID or not ("yes" or "no") based on the records in INFECTED-small.

Step 2 – Queries:
For each of the queries below, please design a solution on the Spark infrastructure. We encourage you to work with Scala, as that is the more popular scalable solution for using Spark natively. However, alternatively, you and your team could also opt to use instead PySpark, the Python API over Spark. In either case, your code should be scalable and utilize **RDDs. (Do not use SparkSQL on this problem.) Provide a brief description and discussion of your solution.**

Query 1: Given a huge file **PEOPLE-large** and a small file **INFECTED-small**; with the later one containing the people known to be infected with COVID-19. Everyone attended the same concert today. Then for each infected person infect-i in the **INFECTED-small** file, find all people p-j in the **PEOPLE-large** file that were seated within 6 units range of infect-i and that, thus, are now at risk of having become infected due to close contact.
Return the list of join pairs composed of people pi from the **PEOPLE-large** files that are a close contact with the person infect-i that infected them (pi, infect-i). (6 points)

Query 2: Given the same setup in Query 1, however, return the pi.id of all people pi from **PEOPLE-Large** that were a close contact to at least one infected person at the above concert. If a person pi was close contact of two or more different infected people, please return that person pi only once (i.e., remove duplicates). Return the result (pi.id). (6 points)

Query 3: Given a huge point file **PEOPLE-SOME-INFECTED-large** that contains people with some of them but not all infected as indicated by the attribute (INFECTED = "yes" or "no"). Then for each infected person infect-i in the **PEOPLE-SOME-INFECTED-large** file, count how many people p-j in the same file are within 6 feet range of infect-i (but don't count infect-i itself in its own count). Return the result (infect-i, count-of-close-contacts-of-infect-i). (8 points)

2- SparkSQL and MLlib: Processing Purchase Transactions (20 points + 5 bonus)

Overview: This problem studies Purchase Transactions from users. After generating the data, you analyze the data using SparkSQL. After which you will utilize Spark MLlib to predict the predict Purchase Prices and analyze the results.

Step 1 – Data Creation:
Write a program that creates two datasets (files): Customers and Purchases. Each line in Customers file represents one customer, and each line in Purchases file represents one purchase transaction. The attributes within each line are comma separated.
Customers (C) dataset should have the following attributes for each customer:

- ID: unique sequential number (integer) from 1 to 50,000 (50,000 lines)
- Name: random sequence of characters of length between 10 and 20 (no commas!)
- Age: random number (integer) between 18 to 100
- CountryCode: random number (integer) between 1 and 500
- Salary: random number (float) between 100 and 10,000,000

Purchases (P) dataset have the following attributes for each purchase transaction:

- TransID: unique sequential number (integer) from 1 to 5,000,000 (5M purchases)
- CustID: References one of the customer IDs, i.e., on Avg. a customer has 100 transactions.
- TransTotal: Purchase amount as random number (float) between 10 and 2000
- TransNumItems: Number of items as random number (integer) between 1 and 15
- TransDesc: Text of characters of length between 20 and 50 (careful: no commas)

Task 2.0) Generate the above data and load your data into your storage. **(5 points)**

Options. Your team should choose one of the two options below depending on if you want to practice your scalable SQL skills or your scalable Data Mining skills. You are not expected to work on both options.
If your team submits one option, you can gain the full 15 points. If your team turns in both options, you can gain up to 5 bonus points for the 2nd option. You could then achieve a total of 25 points for this problem 2.

Option 2.A: Use SparkSQL to implement following workflows queries (15 points)

Task 2.A.1) Filter out (drop) the Purchases from P with a total purchase amount above $600. Store the result as T1. (5 points)

Task 2.A.2) Group the Purchases in T1 by the Number of Items purchased. For each group calculate the median, min and max of total amount spent for purchases in that group. Report the result back to the client side. (5 points)

Task 2.A.3) Group the Purchases in T1 by customer ID only for young customers between 18 and 25 years of age. For each group report the customer ID, their age, and total number of items that this person has purchased, and total amount spent by the customer. Store the result as T3. (5 points)

Option 2.B: Use MLlib to implement the following prediction tasks (15 points).

Task 2.B.1) Data Preparation: Generate a dataset composed of customer ID, TransID, Age, Salary, TransNumItems and TransTotal. Store it as *Dataset*.
Then (randomly) split your *Dataset* into two subsets, namely, *Trainset* and *Testset*, such that *Trainset* contains 80% of *Dataset* and *Testset* the remaining 20%. (3 points)

Task 2.B.2) Predict the price: Identify and train 2 machine learning algorithms to predict **TransTotal.** You could treat this problem as a regression model (see) in which the "features" (X) for this task are *Age, Salary, TransNumItems* and the "outcome" (Y) is *TransTotal*. The algorithms should be trained over *Trainset* and applied (inference) over *Testset* (Task 2.B.1). (8 points)

Task 2.B.3) Analyze and discuss your results: Identify and use at least 1 metric to evaluate the above two algorithms from Task 2.B.2 using Regression Evaluation (https://spark.apache.org/docs/1.6.1/mllib-evaluation-metrics.html#regression-model-evaluation). Report your results determining which model was more effective and provide a conclusion. (4 points)

* * *