CSE 4510/5310 – Mgmt & Proc Big Data

Fall 2022

Hands-on Activity 4

Apache Spark DataFrames

Total Points: 40

Date Assigned: Friday, Oct 28, 2022 **Due Date:** Saturday, Nov 5, 2022

Submission Instructions: This is an individual activity. Please submit your work on Canvas as a Jupyter Notebook `ipynb` file named `cse4510_yourname_activity4.ipynb`. **Your code must be written in pyspark.**

Key Big Data Processing Methods Demonstrated

- To use Spark DataFrames to analyze a real-world dataset
  - To preprocess a dataset using PySpark
  - To answer a business question by augmenting and aggregating data

For this assignment, you will be working with a more complete version of the *San Francisco Fire Call*s Dataset. This dataset is approximately 2.3GB and contains fire calls from 2000 - 2022, 6M rows and 34 columns. Locate the dataset here: https://data.sfgov.org/Public-Safety/Fire-Department-Calls-for-Service/nuek-vuh3.

1. **Data Cleaning** (20 points)
   (a) (1 point) Create a parallelized Spark Context (local[*]) and use it to complete the assignment.
   (b) (1 point) Read the dataset using 0.1% sampling ratio to infer the schema.
   (c) (2 points) Create a new column named *Delay*, which is the difference in minutes (rounded to 2DP) between *Response DtTm* and *Received DtTm*. Display the *Response DtTm*, *Received DtTm*, and *Delay* for the record with CallNumber 203350320.
   (d) (2 points) Drop the following columns from the dataset and arrange the data such that the same schema from the tutorial given in class can be used to read the dataset: *Received DtTm, Entry DtTm, Dispatch DtTm, Response DtTm, On Scene DtTm, Transport DtTm, Hospital DtTm*
   (e) (2 points) Save the modified dataset with only the columns featured in the in-class tutorial to a single CSV file
   (f) (1 point) Read the updated dataset using the schema provided in the in-class tutorial
   (g) (1 point) Return a list of 10 Distinct *Call Types*
   (h) (1 point) Return a count of the distinct *Call Types*
   (i) (1 point) Rename the *Delay* column to *ResponseDelayedinMins* and return the top 5 records (sorted by ResponseDelayedinMins in descending order) where ResponseDelayedinMins > 5 (Show only the ResponseDelayedinMins column in your output)
   (j) (2 points) Return a list of the top 10 delayed Fire Responses sorted in descending order and showing the *Call Type*, *Address* (Full address in title-case in one column including City and Zip Code), *Battalion*, *ResponseDelayedinMins*, and *Unit Type*
   (k) (2 points) Create a horizontal bar chart of the top 10 delayed Fire Responses (*ResponseDelayedinMins* vs *Call Number*)
   (l) (2 points) Convert the following fields to date: *CallDate*, *WatchDate*, *AvailableDtTm*
   (m) (2 points) Show a complete list of the distinct *years* represented in the dataset based on the *IncidentDate*
2. **EDA** (20 points)
   (a) (2 points) Create a *MonthYear* Column in the dataset of the format YYYYMM (eg: 202209) based on the *IncidentDate*
   (b) (10 points) Create an in-memory DataFrame that stores the *Temperature* and the *MonthYear* from Global Summary of the Month (GSOM) data dynamically harvested from the NOAA API from the weather station located Downtown, San Francisco.
   (c) (5 points) Augment the dataset with the temperature data and group the data by *MonthYear*
   (d) (3 points) Answer the following question: Is there correlation between the average monthly temperature and the number of fire calls per month?

3. **Bonus** (5 points)

   (a) Redo Activity 3 using JavaSpark DataFrames to produce the following output. Include your Java code as Markdown code-blocks in your notebook as well a screenshot showing your Java output.

```
+------------+-----+--------------------------------+-------------+
|countries   |count|bar                             |summary_count|
+------------+-----+--------------------------------+-------------+
|USA         |34325|████████████████████████████████|(34.33k)     |
|France      |8311 |████████                        |(8.31k)      |
|UK          |7490 |███████                         |(7.49k)      |
|India       |6373 |██████                          |(6.37k)      |
|Italy       |5056 |█████                           |(5.06k)      |
|Germany     |3721 |███                             |(3.72k)      |
|Japan       |3701 |███                             |(3.7k)       |
|Canada      |3621 |███                             |(3.62k)      |
|Spain       |2731 |██                              |(2.73k)      |
|Hong Kong   |1884 |█                               |(1.88k)      |
|Turkey      |1552 |█                               |(1.55k)      |
|Belgium     |1354 |█                               |(1.35k)      |
|South Korea |1299 |█                               |(1.3k)       |
|Sweden      |1229 |█                               |(1.23k)      |
|Australia   |1181 |█                               |(1.18k)      |
|Mexico      |1173 |█                               |(1.17k)      |
|China       |1166 |█                               |(1.17k)      |
|West Germany|1114 |█                               |(1.11k)      |
|Russia      |1083 |█                               |(1.08k)      |
|Netherlands |1031 |█                               |(1.03k)      |
+------------+-----+--------------------------------+-------------+
only showing top 20 rows
```