1 Overview

Welcome to SparkSQL Machine Practice. This MP has both java and python templates. Please use whichever you prefer.

2 General Requirements

Please note that our grader runs on a docker container and is **NOT connected to the internet**. Therefore, **no additional libraries are allowed** for this assignment. Also, you will **NOT be allowed to create any file or folder outside the current folder** (that is, you can only create files and folders in the folder that your solutions are in).

3 Set Up SparkSQL

Step 1: Start the "default" Docker machine that you created when following the "Tutorial: Docker installation" in week 4, run:

```
docker-machine env
# follow the instruction to configure your shell: eval $(...)
docker-machine start default
```

Step 2: Download the Dockerfile and related files for this MP, change the current folder, build, and run the docker image, run:

```
git clone https://github.com/UIUC-CS498-Cloud/Docker_MP3
cd Docker_MP3
docker build -t mp3 .
docker run -it mp3 bin/bash
```

Submission

1 Requirements

This assignment will be graded based on **JDK 8** & **Python 3.5**

2 Procedures

Step 1: Download the Java templates and change the current folder, run:

```
git clone https://github.com/UIUC-CS498-Cloud/MP3_SparkSQL_template.git

# Use Java Templates
cd MP3_SparkSQL_template/java

# Use Python Templates
cd MP3_SparkSQL_template/python

```

Step 2: Finish the exercises by editing the provided templates files. All you need to do is complete the parts marked with **TODO**. Please note that you are **NOT allowed to use any additional libraries.**

- Each exercise has a Java/Python code template. All you must do is edit this file.
- Each exercise should be implemented in one file only. Multiple file implementation is not allowed.
- The code should be compiled and run on the sample Docker image.
- You should run the parts in the order described below. Otherwise, you might not get the correct results. For example, you should create the RDDs/DataFrames first and then list them.
- Remember that the output is case sensitive.

Exercise A: **Setup**

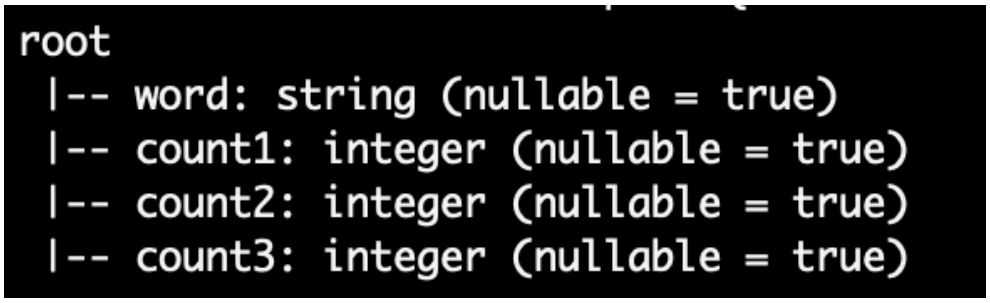
In this exercise, you will create one RDD and one DataFrame objects from Spark's API as shown below. To make the implementation easier, we have provided a boilerplate for this exercise in the following file: **MP3_PartA.java/py**.

1. Setup (10 points): Download the gbook file (http://storage.googleapis.com/books/ngrams/books/googlebooks-eng-all-1gram-20120701-a.gz), unzip it into our current directory (around 1.8g) and write a function to load it in an RDD & DataFrame. Print your DataFrame's schema.

Following are the commands we will use to run your file:

```
# Java
./run.sh MP3_PartA Output_PartA
# Python
spark-submit MP3_PartA.py
```

Your output should contain:



Exercise B: **Counting**

How many lines does the RDD contains? Answer this question via both RDD api & Spark SQL?

To make the implementation easier, we have provided a boilerplate for this exercise in the following file: **MP3_PartB.java** .

Following are the commands we will use to run your file:

```
# Java
./run.sh MP3_PartB Output_PartB
# Python
spark-submit MP3_PartB.py
```

Your output should contain:

```
86618505
+--------+
|count(1)|
+--------+
|86618505|
+--------+
```

Exercise C: **Filtering**

Count the number of appearances of word 'ATTRIBUTE'.

To make the implementation easier, we have provided a boilerplate for this exercise in the following file: **MP3_PartC.java** .

Following are the commands we will use to run your file:

```
# Java
./run.sh MP3_PartC Output_PartC
# Python
spark-submit MP3_PartC.py
```

Your output should contain:

```
201
+--------+
|count(1)|
+--------+
|     201|
+--------+
```

Exercise D: **MapReduce**

List the three most frequent 'word' with their count of appearances.

To make the implementation easier, we have provided a boilerplate for this exercise in the following file: **MP3_PartD.java** .

Following are the commands we will use to run your file:

```
# Java
./run.sh MP3_PartD Output_PartD
# Python
spark-submit MP3_PartD.py
```

Your output should contain:

```
+---------+--------+
|     word|count(1)|
+---------+--------+
|  all_DET|     425|
| are_VERB|     425|
|about_ADP|     425|
+---------+--------+
```

Exercise E: **Joining**

The following program construct a new DataFrame out of 'df' with a much smaller size.

```
df2 = df.select("word", "count1").distinct().limit(1000)
df2.createOrReplaceTempView('gbooks2') # Register table name for SQL
```

Now we are going to perform a JOIN operation on 'df2'. Do a self-join on 'df2' in lines with the same 'count1' values and see how many lines this JOIN could produce. Answer this question via DataFrame API and Spark SQL API

To make the implementation easier, we have provided a boilerplate for this exercise in the following file: **MP3_PartE.java** .

Following are the commands we will use to run your file:

```
# Java
./run.sh MP3_PartE Output_PartE
# Python
spark-submit MP3_PartE.py
```

The output should look similar to below:

```
9658
9658
```

Mark as completed