DSCI 551 – HW5

(Hadoop and Spark)

(Fall 2021)

100 points, Due 11/28, Sunday

SET UP:

Download solution.json, hw5_grade.py, run.sh from blackboard and put it under your homework folder and submit them along with your homework.

Run "chmod 707 run.sh"

This give you a chance to test your answer with TA's grading script.

1. [50 points] Write a Hadoop MapReduce SQL2MR.java to implement the following SQL query in Hadoop MapReduce on the film table (from Sakila) stored in multiple CSV files: **film1.csv and film2.csv**. To learn how to compile and run your java file in Hadoop, please refer to the WordCount tutorial. **Your program does not need to use a combiner. However, you should think about how to implement one (e.g., being able to write pseudocode describing the input, output, and logic of combiner and related reducer).**

```
SELECT rating, avg(rental_rate)
FROM film
where length >= 60
group by rating
having count(*) >= 160
```

   Note that each line in the csv file has four values: film_id, rental_rate, length, and rating. For example:

```
1    0.99    86    PG
2    4.99    48    G
…
```

   Execution format:

```
hadoop jar hw5.jar SQL2MR input output
```

   where input is a directory storing all the CSV files and output is the directory for storing output from your mapreduce program. Two sample CSV files are provided but note that your program may be tested using additional files with the same format.

Submission:

- SQL2MR.java
- hw5.jar
- q1_screenshot.png: A screenshot showing the output of your program on EC2. For example (from WordCount), **name it "q1.png"**:

```
[ec2-user@ip-172-31-26-81 class]$ cd output
[ec2-user@ip-172-31-26-81 output]$ ls -l
total 4
-rw-r--r-- 1 ec2-user ec2-user 23 Nov  4 23:32 part-r-00000
-rw-r--r-- 1 ec2-user ec2-user  0 Nov  4 23:32 _SUCCESS
[ec2-user@ip-172-31-26-81 output]$ cat part-r-00000
hello   8
this    2
world   6
[ec2-user@ip-172-31-26-81 output]$
```

For question 2 and 3 below, you will be using three csv files: **film.csv, actor.csv, and film_actor.csv** as input. Note that each of these files contains the complete information for the corresponding table (i.e. data are NOT split into multiple files like in question 1). Note also that the files have headers in the first line.

2. [20 points, Spark DataFrame] Write a Spark script using Spark DataFrame functions for each of the following questions. 5 points each question.

   Execution format: python3 q2_x.py

   Name all your files q2_x.py

   Any typo in column name and order will result in loss of ALL POINT

   a) Find out how many films are rated as either 'PG' or 'PG-13', use '**.show()'** to output your result, your columns' name and order should look **EXACTLY** like:
   |count|

   b) Find first and last name of actors who have played in the film 'ANONYMOUS HUMAN', use '**.show()'** to output your result, your columns' name and order should look **EXACTLY** like:
   |first_name|last_name|

   c) Find name of actor who has played in the most number of films, use '**.show()'** to output your result, your columns' name and order should look **EXACTLY** like:
   |first_name|last_name|

   d) Find answer to the SQL query in Question 1:

   ```
   SELECT rating, avg(rental_rate)
   ```

FROM film

where length >= 60

group by rating

having count(*) >= 160

use '**.show()'** to output your result, and round **'avg_rate' to 2 decimal with pyspark.sql.functions.round (aka. fc.round)**

your columns' name and order should look **EXACTLY** like:
|rating|avg_rate|

3. [30 points, Spark RDD]

For each of the subquestion in Question 2 above, write a Spark script using Spark RDD API functions to answer the question. 7.5 points each question.

Execution format: python3 q3_x.py

Name all your files q3_x.py

a) output your result with the following template:

```
for count in res:
        print(count)
```

b) output your result with the following template:

```
for first, last in res:
        print(first, last)
```

c) output your result with the following template:

```
for first, last in res:
        print(first, last)
```

d) output your result with the following template (round average to 2 decimal with the template):

```
for rating, avg in res:
        print(rating, "{:.2f}".format(avg))
```

Submission:

1. Put all files in the same directory and compress it into a zip file. Zip file name format:

LASTNAME_FIRSTNAME_HW5.zip

Make sure when the file is unzipped, the folder name is LASTNAME_FIRSTNAME_HW5

2. Your submission folder should contain 19 files and look **EXACTLY** like, each extra file (-2 pts)

```
dexuanluo@MacBook-Pro src % ls
SQL2MR.java     film1.csv       hw5.jar         q2_a.py         q2_d.py         q3_c.py         solution.json
actor.csv       film2.csv       hw5_grade.py    q2_b.py         q3_a.py         q3_d.py
film.csv        film_actor.csv  q1.png          q2_c.py         q3_b.py         run.sh
```

Please understand how TA will run your python scripts for q2 and q3. The TAs will simply run:

./run.sh

Once done, your folder will look EXACTLY like:

```
dexuanluo@MacBook-Pro src % ls
SQL2MR.java     film1.csv       hw5.jar         q2_a.py         q2_b.py.res     q2_d.py         q3_a.py.res     q3_c.py         q3_d.py.res
actor.csv       film2.csv       hw5_grade.py    q2_a.py.res     q2_c.py         q2_d.py.res     q3_b.py         q3_c.py.res     run.sh
film.csv        film_actor.csv  q1.png          q2_b.py         q2_c.py.res     q3_a.py         q3_b.py.res     q3_d.py         solution.json
```

After all ".res" files are generated, we will run

python3 hw5_grade.py

Your folder will look EXACTLY like:

```
dexuanluo@MacBook-Pro src % python3 hw5_grade.py
dexuanluo@MacBook-Pro src % ls
SQL2MR.java     film2.csv       q1.png          q2_b.py.res     q2_d.py.res     q3_b.py.res     q3_d.py.res
actor.csv       film_actor.csv  q2_a.py         q2_c.py         q3_a.py         q3_c.py         run.sh
film.csv        hw5.jar         q2_a.py.res     q2_c.py.res     q3_a.py.res     q3_c.py.res     score.res
film1.csv       hw5_grade.py    q2_b.py         q2_d.py         q3_b.py         q3_d.py         solution.json
```

After running this command, a score.res file will be generated.

If your column name or order are incorrect, points will be deducted as mentioned before.

You can test your files with the given grading script before you submit them, but please make sure to delete all the .res files before compressing the folder and making your submission.

If you change a single byte in hw5_grade.py, run.sh or solution.json, 50 pts will be deducted.