Homework 5: Hadoop and Spark

DSCI 551 – Spring 2025

Due: 11:59pm, 4/18, 2025, **Friday**

Points: 100

In this homework, you are provided with 3 csv files on film, actor, and their relationship. The data are mostly based on the MySQL Sakila sample database: https://dev.mysql.com/doc/sakila/en/.

Remember to use the provided templates for your homework.

1. [Hadoop MapReduce, 40 points] Modify the provided SQL2MR.java, fill in the missing codes, to answer the following SQL question using film.csv.

   ```
   select rating, avg(length)
   from film
   where special_features like '%Trailer%' and rental_rate >= 2
   group by rating
   having count(*) > 60
   order by rating
   ```

   Note: you should remove the header row before processing. Recall steps of compiling and executing the code:

   - **hadoop com.sun.tools.javac.Main SQL2MR.java**
   - **jar cf sql2mr.jar SQL2MR*.class**
   - **hadoop jar sql2mr.jar SQL2MR input output**

   where the input directory stores the film.csv file (with header row removed).

2. [Spark DataFrame, 30 points] For each of the following questions, write a Spark DataFrame script to answer the question.

   (1) Select title
   From film
   Where description like '%Amazing%' and rental_rate < 1
   Order by title desc
   Limit 5

   (you can assume that the “like” operator in SQL-like syntax is case sensitive in this question).

   (2) Select rating, avg(rental_rate)
   From film

```
Where rental_duration >= 3
Group by rating
Having count(*) > 200
```

(3)
```
with t as
        (select actor_id, count(*) cnt
         from film_actor
         group by actor_id)
select actor_id from t
where cnt = (select max(cnt) from t)
```

(Note there may be multiple such actors, i.e., whose cnt is maximum).

(4)
```
select title from film_actor natural join actor natural join film
where first_name = "TOM" and last_name = "MCKELLEN"
order by title
limit 5
```

(5)
```
select first_name, last_name
from actor
where   actor_id in (select actor_id from film_actor where film_id = 23)
        and actor_id not in (select actor_id from film_actor where film_id = 1)

order by first_name
```

3. [RDD, 30 points] Write an RDD script for each of SQL queries in Question 2. Note that your script should parallelize the computations as much as possible. **For example, it should not call collect() to collect data from intermediate RDDs for further processing**.

Submissions Details:

1. Please submit ONLY 5 files:

   [Q1 - SQL2MR.java, sql2mr.jar, part-r-00000 || Q2 - q2_spark_dataframe.py || Q3 - q3_spark_rdd.py]

2. Folder Structure:

   Q1 files: SQL2MR.java, sql2mr.jar, part-r-00000 → must be in the same folder
   (e.g., Q1_SQL2MR/SQL2MR.java, Q1_SQL2MR/sql2mr.jar, Q1_SQL2MR/part-r-00000)

   Q2 file: q2_spark_dataframe.py

   Q3 file: q3_spark_rdd.py

3. Make sure to submit the updated files with your solutions.

4. Do NOT modify any other contents in the provided templates. Just fill your code where instructed by comments.
5. Your code should work for different test cases (not hardcoded outputs).
6. For Q2 and Q3 Add the output for each query as a comment right after the corresponding result variable.
7. Please open all folders to access the datasets and templates provided in hw5.zip
8. IMPORTANT: Even if template has:
   resultX = None
   print(resultX)
   - It is NOT necessary to write everything in 1 line. You can write 2-3 lines of clean and correct code
   - This applies to both: Q2 → q2_spark_dataframe.py and Q3 → q3_spark_rdd.py
   - Focus on correctness and clarity of code, not on making it 1 line