# DSCI 551 – HW5

# (Hadoop and Spark)

# (Spring 2024)

100 points, Due 4/15, Monday, 11:59pm

In this homework assignment, we consider a film dataset which consists of the following files: film.csv, film_actor.csv, and actor.csv.

1. [40 points] Write a Hadoop MapReduce program named SQL2MR.java that finds answer to the following SQL query on the **film.csv** data.

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

   You can modify SQL2MR.java provided to you in this handout.

   Before running the program, please remove the header from the film.csv file and save it under a HDFS directory called input.

   You are reminded of the following steps to compile and run the java program:

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

   **Submission**: SQL2MR.java, sql2mr.jar, and part-r-00000 file under output.

2. [Spark DataFrame, 30 points] For each of the following SQL queries on the dataset, write a Spark DataFrame script to find the answer to the query. Show the output of your script. You can assume that the following has already been executed before your script.

   **[Please refer to the given templates]**

   ```
   import pyspark.sql.functions as fc
   film = spark.read.csv('film.csv', header=True, inferSchema=True)
   actor = spark.read.csv('actor.csv', header=True, inferSchema=True)
   film_actor = spark.read.csv('film_actor.csv', header=True, inferSchema=True)
   ```

   a. SELECT title, description
      FROM film
      WHERE rating = "PG"

LIMIT 5

b. SELECT rating, avg(replacement_cost)
FROM film
WHERE length >= 60
GROUP BY rating
HAVING count(*) >= 160

c. SELECT actor_id FROM film_actor WHERE film_id = 1)
intersect
(SELECT actor_id FROM film_actor where film_id = 23)

d. SELECT DISTINCT first_name, last_name
FROM actor JOIN film_actor ON actor.actor_id = film_actor.actor_id
WHERE film_id in (1, 2, 3)
ORDER BY first_name
LIMIT 5

e. SELECT rental_duration, rating, min(length), max(length), avg(length), count(length)
FROMfilm
GROUP BY rental_duration, rating
ORDER BY rental_duration desc
LIMIT 10

3. [Spark RDD, 30 points] For each of the SQL queries in Question 2 (repeated here), write a Spark RDD script to find the answer to the query. Show the output of your script.
**[Please refer to the given templates]**

a. SELECT title, description
FROM film
WHERE rating = "PG"
LIMIT 5

b. SELECT rating, avg(replacement_cost)
FROM film
WHERE length >= 60
GROUP BY rating
HAVING count(*) >= 160

c. SELECT actor_id FROM film_actor WHERE film_id = 1)
intersect
(SELECT actor_id FROM film_actor where film_id = 23)

d. SELECT DISTINCT first_name, last_name

```
FROM actor JOIN film_actor ON actor.actor_id = film_actor.actor_id
WHERE film_id in (1, 2, 3)
ORDER BY first_name
LIMIT 5
```

e.
```
SELECT rental_duration, rating, min(length), max(length), avg(length), count(length)
FROM film
GROUP BY rental_duration, rating
ORDER BY rental_duration desc
LIMIT 10
```

**Submission details:**

- **Q1:** Please Submit the following files (file names should be as mentioned below)
  - SQL2MR.java
  - sql2mr.jar
  - part-r-00000
- **Q2 and Q3:** Please submit following files (file names should be as mentioned below)
  - Q2.py and Q3.py
  - after each query (2a, 2b, 2c, etc….), include the output of the query as comments in the respective python files
  - when Q2.py and Q3.py are ran, they should also print the outputs to the console [Please see instructions in attached template]
- Please **do not** zip your files
- 0 points if your code does not run
- Only modules given in templates. **Do not** import any other extra modules.
- LATE submissions are not accepted.