Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

BIG DATA & ANALYTICS

ASSIGNMENT 3: OPEN BOOK EXAM

BACKGROUND.

It's the last day of your short-term internship in the Data Analytics Department of the start-up OptimiseYourJourney, which will enter the market next year with a clear goal in mind: "*leverage Big Data technologies for improving the user experience in transportation*". Your contribution in assignments 1 and 2 has proven the potential OptimiseYourJourney can obtain by applying MapReduce and Spark SQL to analyse large-scale public transportation datasets as the one in the New York City Bike Sharing System: https://www.citibikenyc.com/

OptimiseYourJourney

Today, you are asked to complete a new exercise (over the very same NYC dataset you have used in assignments 1 and 2) and by applying the very same techniques (sequential approach, MapReduce simulator and Spark SQL) you have used in assignments 1 and 2.

- Exercise 1: Complete the exercise using the sequential approach.
- Exercises 2 & 3: Complete the exercise using the MapReduce simulator (Exercise 2 - map stage & Exercise 3 – reduce stage).
- Exercise 4: Complete the exercise using Spark SQL.

DATASET:

This dataset occupies ~80MB and contains 73 files. Each file contains all the trips registered the CitiBike system for a concrete day:

- 2019_05_01.csv => All trips registered on the 1st of May of 2019.
- 2019_05_02.csv => All trips registered on the 2nd of May of 2019.
- ...
- 2019_07_12.csv => All trips registered on the 12th of July of 2019.

Altogether, the files contain 444,110 rows. Each row contains the following fields: ***start_time , stop_time , trip_duration , start_station_id , start_station_name , start_station_latitude , start_station_longitude , stop_station_id , stop_station_name , stop_station_latitude , stop_station_longitude , bike_id , user_type , birth_year , gender , trip_id***

- **(00)** ***start_time***
  - A String representing the time the trip started at. <%Y/%m/%d %H:%M:%S>
  - Example: "2019/05/02 10:05:00"
- **(01)** ***stop_time***
  - A String representing the time the trip finished at. <%Y/%m/%d %H:%M:%S>
  - Example: "2019/05/02 10:10:00"
- **(02)** ***trip_duration***
  - An Integer representing the duration of the trip.
  - Example: 300
- **(03)** ***start_station_id***
  - An Integer representing the ID of the CityBike station the trip started from.
  - Example: 150
- **(04)** ***start_station_name***
  - A String representing the name of the CitiBike station the trip started from.
  - Example: "E 2 St &; Avenue C".
- **(05)** ***start_station_latitude***
  - A Float representing the latitude of the CitiBike station the trip started from.
  - Example: 40.7208736
- **(06)** ***start_station_longitude***
  - A Float representing the longitude of the CitiBike station the trip started from.
  - Example: -73.98085795
- **(07)** ***stop_station_id***
  - An Integer representing the ID of the CityBike station the trip stopped at.
  - Example: 150
- **(08)** ***stop_station_name***
  - A String representing the name of the CitiBike station the trip stopped at.
  - Example: "E 2 St &; Avenue C".

- **(09)** ***stop_station_latitude***
  - A Float representing the latitude of the CitiBike station the trip stopped at.
  - Example: 40.7208736
- **(10)** ***stop_station_longitude***
  - A Float representing the longitude of the CitiBike station the trip stopped at.
  - Example: -73.98085795
- **(11)** ***bike_id***
  - An Integer representing the id of the bike used in the trip.
  - Example: 33882.
- **(12)** ***user_type***
  - A String representing the type of user using the bike (it can be either "Subscriber" or "Customer").
  - Example: "Subscriber".
- **(13)** ***birth_year***
  - An Integer representing the birth year of the user using the bike.
  - Example: 1990.
- **(14)** ***gender***
  - An Integer representing the gender of the user using the bike (it can be either 0 => Unknown; 1 => male; 2 => female).
  - Example: 2.
- **(15)** ***trip_id***
  - An Integer representing the id of the trip.
  - Example: 190.

TASKS / EXERCISES.

The tasks / exercises to be completed as part of the assignment are described in the next pages of this PDF document.

- The following exercises are placed in the folder **my_code:**
  1. **A03_Part1**/A03_Part1.py
  2. **A03_Part2**/my_mapper.py
  3. **A03_Part2**/my_reducer.py
  4. **A03_Part3**/A03_Part3.py

  Marks are as follows:
  - **A03_Part1**/A03_Part1.py => 33 marks
  - **A03_Part2**/my_mapper.py => 17 marks
  - **A03_Part2**/my_reducer.py => 17 marks
  - **A03_Part3**/A03_Part3.py => 33 marks

  Tasks:
  - **A03_Part1**/A01_Part1.py
  - **A03_Part3**/A01_Part3.py
    Complete the function **my_main** of the Python program.
    Do not modify the name of the function nor the parameters it receives.

  - **A03_Part2**/my_mapper.py
    Complete the function **my_map** of the Python program.
    Do not modify the name of the function nor the parameters it receives.

  - **A03_Part2**/my_reducer.py
    Complete the function **my_reduce** of the Python program.
    Do not modify the name of the function nor the parameters it receives.

RUBRIC.

Exercises 1-4.

- 20% of the marks => Complete attempt of the exercise (even if it does not lead to the right solution or right format due to small differences).
- 40% of the marks => Right solution and format (following the aforementioned rules) for the provided dataset.
- 40% of the marks => Right solution and format (following the aforementioned rules) for any "Additional Dataset" test case we will generate. The marks will be allocated in a per test basis (i.e., if 4 extra test are tried, each of them will represent 10% of the marks).

TEST YOUR SOLUTIONS.

- The folder **my_results** contains the expected results for each exercise.
  - **A03_Part1**/result.txt
  - **A03_Part2/1_my_map_simulation** => 73 files for the output of each map process.
  - **A03_Part2/2_my_sort_simulation**/sort_1.txt => input for first reduce process.
  - **A03_Part2/3_my_reduce_simulation**/reduce_sort_1.txt => output of first reduce.
  - **A03_Part3**/result.txt

- Moreover, the subfolder **my_results/check_results** allows you to see if your code is producing the expected output or not.
  - The file **test_checker.py** needs two folders and compares if their files are equal or not.
  - When you have completed one part (e.g., A03_Part2), copy the folder **my_results/A03_Part2** into the folder **my_results/check_results/Student_Attempts/A03_Part2**.
  - Open the file **test_checker.py** and edit the line 107 with the value of the part you are attempting (e.g., part = 2).
  - Run the program **test_checker.py**. It will tell you whether your output is correct or not.

For example, as an example let's run the Python program **test_checker.py** to see if the solution attempt done by the student for A03_Part1 and A03_Part2 is correct or not.

- python3.9 test_checker.py 1

```
--------------------------------------------------------
Checking :
./Assignment_Solutions/A03_Part1/result.txt
./Student_Attempts/A03_Part1/result.txt

Test passed!
--------------------------------------------------------
Congratulations, the code passed all the tests!
--------------------------------------------------------
```

  As we can see, the code of the student is correct, and thus it gets the marks.

- python3.9 test_checker.py 2

```
--------------------------------------------------------
Checking :
./Assignment_Solutions/A03_Part2/2_my_sort_simulation/sort_1.txt
./Student_Attempts/A03_Part2/2_my_sort_simulation/sort_1.txt

Test did not pass.
--------------------------------------------------------
Checking :
./Assignment_Solutions/A03_Part2/3_my_reduce_simulation/reduce_sort_1.txt
./Student_Attempts/A03_Part2/3_my_reduce_simulation/reduce_sort_1.txt
```

```
Test did not pass.
-----------------------------------------------------------
Sorry, the output of some files is incorrect!
-----------------------------------------------------------
```

As we can see, the code of the student is not correct, and thus it does not get the marks. The problem was that some output lines in some files were wrong.

Main Message

Use the program **test_checker.py** to ensure that all your exercises produce the expected output (and in the right format!).

SUBMISSION DETAILS / SUBMISSION CODE OF CONDUCT.

This open book exam is taking place on Friday 21st of May, 3:30pm – 6:30pm.
Submit to Canvas by the 21st of May, 6:29pm.

- Late submissions will not be accepted.

On submitting the assignment you adhere to the following declaration of authorship. If you have any doubt regarding the plagiarism policy discussed at the beginning of the semester do not hesitate in contacting me.

Declaration of Authorship

I, ___YOUR NAME___, declare that the work presented in this assignment titled 'Assignment 3: Open Book Exam' is my own. I confirm that:

- This work was done wholly by me as part of my BSc. in Software Development or BSc. in Web Development.
- Where I have consulted the published work and source code of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this assignment source code and report is entirely my own work.

EXERCISE DESCRIPTION

Consider the following dataset containing just 6 rows. The trip with longest trip_duration is the one highlighted with ▭ . Such trip has trip_id = 3 and trip_duration = 4000.

…	**trip_duration**	…	**trip_id**
…	1000	…	0
…	2000	…	1
…	3000	…	2
…	**4000**	**…**	**3**
…	2000	…	4
…	3000	…	5

EXERCISE:

- Compute the trip with longest trip_duration. Output both the trip_id and trip_duration of such trip.
    - Note: you can assume the dataset contains just one trip with such longest trip_duration. In other words, you do not have to worry about how to break ties among multiple trips having the very same longest trip_duration.

Example 1:

Given the dataset above, the exercise must report:

- trip_id → 3
- trip_duration → 4000

EXERCISE 1. (33 marks)

Technology:

Python (sequential approach, do not use the MapReduce simulator).

Your task is to:

- Compute the trip with longest trip_duration. Output both the trip_id and trip_duration of such trip.
    - o Note: you can assume the dataset contains just one trip with such longest trip_duration. In other words, you do not have to worry about how to break ties among multiple trips having the very same longest trip_duration.

Complete the function **my_main** of the Python program.

- o Do not modify the name of the function nor the parameters it receives.
- o In particular, the function must read the dataset provided in input_folder and must open and write the results to output_file.
- o You can use the auxiliary function process_line which, given one line from a dataset file, returns a tuple with its content.
- o You can also program any other auxiliary functions you might need.

Results:

Output just one text line with the result. The line must have the following format:

```
trip_id \t (trip_duration) \n
```

Note:

To represent the information, I recommend you not to use a dictionary. Instead, use just a couple of integer variables representing the best trip_duration *&* trip_id *found so far.*

If you want to use any other representation, it is perfectly fine.

EXERCISE 2 AND EXERCISE 3. (34 marks)

Technology:

Python - Use the MapReduce simulator.

Your task is to:

- Compute the trip with longest trip_duration. Output both the trip_id and trip_duration of such trip.
  - Note: you can assume the dataset contains just one trip with such longest trip_duration. In other words, you do not have to worry about how to break ties among multiple trips having the very same longest trip_duration.

my_mapper.py => Complete the function **my_map** of the Python program.

- Do not modify the name of the function nor the parameters it receives.
- In particular, the function must read the content of a file provided by my_input_stream and must write the results to the file provided by my_output_stream. There are no extra parameters provided via my_mapper_input_parameters.
- You can use the auxiliary function process_line which, given one line from a dataset file, returns a tuple with its content.
- You can also program any other auxiliary functions you might need.

my_reducer.py => Complete the function **my_reduce** of the Python program.

- Do not modify the name of the function nor the parameters it receives.
- In particular, the function must read the content of a file provided by my_input_stream and must write the results to the file provided by my_output_stream. There are no extra parameters provided via my_reducer_input_parameters.
- You can also program any other auxiliary functions you might need.

Results:

In my_mapper.py, use a universal key to ensure that just one Reduce process is used. Each Map process outputs just one text line, which must have the following format:

```
universal \t (trip_id, trip_duration) \n
```

In my_reducer.py, output just one text line, which must have the following format:

```
trip_id \t (trip_duration) \n
```

Note:

To represent the information, I recommend you not to use a dictionary. Instead, use just a couple of integer variables representing the best trip_duration *&* trip_id *found so far.*

If you want to use any other representation, it is perfectly fine.

EXERCISE 4. (33 marks)

Technology:

Spark SQL.

Your task is to:

- Compute the trip with longest trip_duration. Output both the trip_id and trip_duration of such trip.
  - Note: you can assume the dataset contains just one trip with such longest trip_duration. In other words, you do not have to worry about how to break ties among multiple trips having the very same longest trip_duration.

Complete the function **my_main** of the Python program.

- Do not modify the name of the function nor the parameters it receives.
- The entire work must be done within Spark SQL:
  - The function my_main must start with the creation operation 'read' above loading the dataset to Spark SQL.
  - The function my_main must finish with an action operation 'collect', gathering and printing by the screen the result of the Spark SQL job.
  - The function my_main must not contain any other action operation 'collect' other than the one appearing at the very end of the function.
  - The resVAL iterator returned by 'collect' must be printed straight away, you cannot edit it to alter its format for printing.

Results:

Output just one Row, which must have the following fields (and this particular order for the fields as well!):

Row(trip_id, trip_duration)