Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

# BIG DATA & ANALYTICS

ASSIGNMENT 1: MAP REDUCE

## **BACKGROUND.**

From the window you can see the sun shining in a lovely spring morning. Its Monday, 10am, and you are in the open plan office of a new start-up, OptimiseYourJourney, which will enter the market next year with a clear goal in mind: "*leverage Big Data-based technologies for improving the user experience in transportation*".

The start-up is at a very early stage, and has no clear product in mind yet. However, they have offered a short-term internship in their Data Analytics Department to help them exploring the datasets, technologies and techniques that can be applied in their future products. They do not pay very well (0€ per hour), but you see this as a good opportunity to complement your knowledge in the module Big Data & Analytics you are studying at the moment, so you have decided to give it a go.

OptimiseYourJourney

In the department meeting that has just finished your boss was particularly happy. During the weekend he was exploring public datasets over the internet and he found a dataset regarding the New York City Bike Sharing System: https://www.citibikenyc.com/
The official website (https://www.citibikenyc.com/system-data) provides publicly available datasets in a monthly basis. Each of these datasets amalgamate all the bike trips of the month. For example, the files "201905-citibike-tripdata.csv.zip", "201906-citibike-tripdata.csv.zip" and "201907-citibike-tripdata.csv.zip" contain information for all the trips that took place in May, June and July of 2019, resp.

Your boss thinks this dataset provides a great opportunity to explore the potential of MapReduce in analysing large datasets. He has already cleaned the dataset for you to perform some data analysis on it.

## 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. **A01_Part1**/A01_Part1.py
    2. **A01_Part2**/A01_Part2.py
    3. **A01_Part3**/A01_Part3.py
    4. **A01_Part4**/my_mapper.py
    5. **A01_Part4**/my_reducer.py
    6. **A01_Part5**/my_mapper.py
    7. **A01_Part5**/my_reducer.py
    8. **A01_Part6**/my_mapper.py
    9. **A01_Part6**/my_reducer.py

  **Marks are as follows:**
    - **A01_Part1**/A01_Part1.py => 16 marks
    - **A01_Part2**/A01_Part2.py => 16 marks
    - **A01_Part3**/A01_Part3.py => 18 marks
    - **A01_Part4**/my_mapper.py => 8 marks
    - **A01_Part4**/my_reducer.py => 8 marks
    - **A01_Part5**/my_mapper.py => 8 marks
    - **A01_Part5**/my_reducer.py => 8 marks
    - **A01_Part6**/my_mapper.py => 9 marks
    - **A01_Part6**/my_reducer.py => 9 marks

  **Tasks:**
    - **A01_Part1**/A01_Part1.py
    - **A01_Part2**/A01_Part2.py
    - **A01_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.

    - **A01_Part4**/my_mapper.py
    - **A01_Part5**/my_mapper.py
    - **A01_Part6**/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.

    - **A01_Part4**/my_reducer.py
    - **A01_Part5**/my_reducer.py
    - **A01_Part6**/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 1-9.**

- 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.
  - **A01_Part1**/result.txt
  - **A01_Part2**/result.txt
  - **A01_Part3**/result.txt
  - **A01_Part4/1_my_map_simulation** => 73 files for the output of each map process.
  - **A01_Part4/2_my_sort_simulation**/sort_1.txt => input for first reduce process.
  - **A01_Part4/2_my_sort_simulation**/sort_2.txt => input for second reduce process.
  - **A01_Part4/3_my_reduce_simulation**/reduce_sort_1.txt => output of first reduce.
  - **A01_Part4/2_my_reduce_simulation**/reduce_sort_2.txt => output of second reduce.
  - **A01_Part5/1_my_map_simulation** => 73 files for the output of each map process.
  - **A01_Part5/2_my_sort_simulation**/sort_1.txt => input for first reduce process.
  - **A01_Part5/3_my_reduce_simulation**/reduce_sort_1.txt => output of first reduce.
  - **A01_Part6/1_my_map_simulation** => 73 files for the output of each map process.
  - **A01_Part6/2_my_sort_simulation**/sort_1.txt => input for first reduce process.
  - **A01_Part6/3_my_reduce_simulation**/reduce_sort_1.txt => output of first reduce.

- 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., A01_Part4), copy the folder **my_results/A01_Part4** into the folder **my_results/check_results/Student_Attempts/A01_Part4**.
  - Open the file **test_checker.py** and edit the line 112 with the value of the part you are attempting (e.g., part = 4).
  - 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 A01_Part1 and A01_Part4 is correct or not.

- python3.13 test_checker.py 1

```
--------------------------------------------------------
Checking :
./Assignment_Solutions/A01_Part1/result.txt
./Student_Attempts/A01_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.

- python3.13 test_checker.py 4

```
--------------------------------------------------------
Checking :
./Assignment_Solutions/A01_Part4/2_my_sort_simulation/sort_1.txt
./Student_Attempts/A01_Part4/2_my_sort_simulation/sort_1.txt

Test did not pass.
--------------------------------------------------------
Checking :
./Assignment_Solutions/A01_Part4/2_my_sort_simulation/sort_2.txt
./Student_Attempts/A01_Part4/2_my_sort_simulation/sort_2.txt

Test passed!
--------------------------------------------------------
Checking :
./Assignment_Solutions/A01_Part4/3_my_reduce_simulation/reduce_sort_1.txt
./Student_Attempts/A01_Part4/3_my_reduce_simulation/reduce_sort_1.txt

Test did not pass.
--------------------------------------------------------
Checking :
./Assignment_Solutions/A01_Part4/3_my_reduce_simulation/reduce_sort_2.txt
./Student_Attempts/A01_Part4/3_my_reduce_simulation/reduce_sort_2.txt

Test passed!
--------------------------------------------------------
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.

## <u>Main Message</u>

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.**

Submit to Canvas by the 21st of March, 17:00h.

- Submissions up to 1 week late will have 10 marks deducted.
- Submissions up to 2 weeks late will have 20 marks deducted.

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 1: MapReduce' 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 1. (16 marks)

**Technology:**

Python (without using the MapReduce simulator).

**Your task is to:**

- Compute the amount of trips starting from and finishing at each station_name.

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

- Do not modify the name of the function nor the parameters it receives.
- In particular, the function must read the dataset provided in input_folder and must open and write the results to output_file.
- 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.

Results:

Output one text line per station_name. Lines must follow alphabetic order in the name of the station. Each line must have the following format:

```
station_name \t (total_trips_starting_from_the_station, total_trips_finishing_at_the_station) \n
```

## EXERCISE 2. (16 marks)

**Technology:**

Python (without using the MapReduce simulator).

**Your task is to:**

- Compute the top_n_bikes with highest total duration time for their trips.

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

- Do not modify the name of the function nor the parameters it receives.
- In particular, the function must read the dataset provided in input_folder and must open and write the results to output_file. The number of bikes to output is specified by the parameter top_n_bikes. For example, if top_n_bikes = 10, then you must output the 10 bikes with highest total duration time for their trips.
- 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.

Results:

Output one text line per bike_id. Lines must follow decreasing order in highest total duration time for their trips. Each line must have the following format:

```
bike_id \t (total_duration_time_for_their_trips, total_number_of_trips) \n
```

## EXERCISE 3. (18 marks)

**Technology:**

Python (without using the MapReduce simulator).

**Your task is to:**

- Sometimes bikes are re-organised (moved) from station A to station B to balance the amount of bikes available in both stations. A truck operates this bike re-balancing service, and the trips done by-truck are not logged into the dataset. Compute all the times a given bike_id was moved by the truck re-balancing system.

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

- Do not modify the name of the function nor the parameters it receives.
- In particular, the function must read the dataset provided in input_folder and must open and write the results to output_file. The concrete bike to be tracked is specified by the parameter bike_id. For example, if bike_id = 35143, then you must track all the times the bike with id 35143 was moved by the truck re-balancing system.
- 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.

Results:

Output one text line per moving trip. Lines must follow temporal order. Each line must have the following format:

```
By_Truck \t (time_it_was_logged_at_station2, station2_id, time_it_was_logged_at_station3, station3_id) \n
```

For example, if the dataset **contains** the following 2 trips:

- **Trip1:** A user used bike_id to start a trip from Station1 on 2019/05/10 09:00:00 and finished the trip at Station2 on 2019/05/10 10:00:00
- **Trip2:** A user used bike_id to start a trip from Station3 on 2019/05/10 11:00:00 and finished the trip at Station4 on 2019/05/10 12:00:00

And the dataset **does not contain** any extra trip:

- **Trip3:** A user used bike_id to start a trip from Station2 and finish at Station3 anytime between 2019/05/10 10:00:00 and 2019/05/10 11:00:00

Then it is clear that the bike was moved from Station2 to Station3 by truck, and we output:

```
By_Truck \t (2019/05/10 10:00:00, Station2, 2019/05/10 11:00:00, Station3) \n
```

## EXERCISE 4 AND EXERCISE 7 (16 marks)

**Technology:**

Python - Use the MapReduce simulator.

**Your task is to:**

- Compute the amount of trips starting from and finishing at each station_name.

**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:

Output one text line per station_name. Lines must follow alphabetic order in the name of the station. Each line must have the following format:

```
station_name \t (total_trips_starting_from_the_station, total_trips_finishing_at_the_station) \n
```

## EXERCISE 5 AND EXERCISE 8 (16 marks)

**Technology:**

Python - Use the MapReduce simulator.

**Your task is to:**

- Compute the top_n_bikes with highest total duration time for their trips.

**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. The extra parameter top_n_bikes is provided via my_reducer_input_parameters.
- You can also program any other auxiliary functions you might need.

Results:

Output one text line per bike_id. Lines must follow decreasing order in highest total duration time for their trips. Each line must have the following format:

```
bike_id \t (total_duration_time_for_their_trips, total_number_of_trips) \n
```

## EXERCISE 6 AND EXERCISE 9 (18 marks)

**Technology:**

Python - Use the MapReduce simulator.

**Your task is to:**

- Sometimes bikes are re-organised (moved) from station A to station B to balance the amount of bikes available in both stations. A truck operates this bike re-balancing service, and the trips done by-truck are not logged into the dataset. Compute all the times a given bike_id was moved by the truck re-balancing system.

**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. The extra parameter bike_id is 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:

Output one text line per moving trip. Lines must follow temporal order. Each line must have the following format:

```
By_Truck \t (time_it_was_logged_at_station2, station2_id, time_it_was_logged_at_station3, station3_id) \n
```

For example, if the dataset **contains** the following 2 trips:

- **Trip1:** A user used bike_id to start a trip from Station1 on 2019/05/10 09:00:00 and finished the trip at Station2 on 2019/05/10 10:00:00
- **Trip2:** A user used bike_id to start a trip from Station3 on 2019/05/10 11:00:00 and finished the trip at Station4 on 2019/05/10 12:00:00

And the dataset **<u>does not contain</u>** any extra trip:

- **Trip3:** A user used bike_id to start a trip from Station2 and finish at Station3
  anytime between 2019/05/10 10:00:00 and 2019/05/10 11:00:00

<u>Then it is clear that the bike was moved from Station2 to Station3 by truck, and we output:</u>

```
By_Truck \t (2019/05/10 10:00:00, Station2, 2019/05/10 11:00:00, Station3) \n
```