COSC 2637/2633 Big Data Processing
Assignment 1 – Tax Trip Statistics

Assessment Type	– Individual assignment. – Submit online via Canvas → Assignment 1. – Marks awarded for meeting requirements as closely as possible. – Clarifications/updates may be made via announcements or relevant discussion forums.
Due Date	Due at 23:59, 8 Sep 2024
Marks	25

Overview
Write MapReduce programs that give you a chance to develop an understanding of principles when solving complex problems on the Hadoop execution platform.

Learning Outcomes
The key course learning outcomes are:
- CLO 1: model and implement efficient big data solutions for various application areas using appropriately selected algorithms and data structures.
- CLO 2: analyse methods and algorithms, to compare and evaluate them with respect to time and space requirements and make appropriate design choices when solving real-world problems.
- CLO 3: motivate and explain trade-offs in big data processing technique design and analysis in written and oral form.
- CLO 4: explain the Big Data Fundamentals, including the evolution of Big Data, the characteristics of Big Data and the challenges introduced.
- CLO 6: apply the novel architectures and platforms introduced for Big data, i.e., Hadoop, MapReduce and Spark.

Assessment Details
You have two datasets: *Trips.txt* which records trip information, and *Taxis.txt* which is about taxi information. Both Trips.txt and Taxis.txt are stored on HDFS. Complete the following MapReduce programming tasks with Python and the methods taught in this course only.

- Using any other language like Java will directly lead to a 0 mark on the assignment. Also, you are not allowed to use any Python MapReduce library such as mrjob.
- A reasonable big data processing program should not create a data structure to hold the complete data in memory. The data should be processed line by line, and the intermediate results are stored in memory only if it is the method taught in the course, like in-memory combining.

A sample of Taxis.txt	A sample of Trips.txt
Taxi#, company, model, year	**Trip#, Taxi#, fare, distance, pickup_x, pickup_y, dropoff_x, dropoff_y**
470,0,80,2018	0,354,232.64,127.23,46.069,85.566,10.355,4.83
332,11,88,2013	1,173,283.7,150.74,5.02,31.765,88.386,27.265
254,10,62,2018	2,8,83.84,43.17,63.269,33.156,92.953,60.647
460,4,90,2022	3,340,259.2,136.3,14.729,13.356,14.304,90.273
113,6,23,2015	4,32,270.07,152.65,27.965,13.37,77.925,62.82
275,16,13,2015	5,64,378.31,202.95,1.145,94.519,98.296,35.469
318,14,46,2014	6,480,235.98,121.23,66.982,66.912,5.02,31.765
	7,410,293.16,162.29,2.841,95.636,91.029,16.232

Task 1 (5 marks)
For each taxi, we consider three types of trips, *long trips* (>=200), *medium trips* (>=100 and <200), and *short trips* (<100). For each taxi and each type of trips, you are asked to count (i) the total number of trips, (ii) the maximum fare of trips, (iii) the minimum fare of trips, and (iv) the average fare per trip. The program should implement *in-mapper combining* with state preserved across lines.

The code must work for 3 reducers. You need to submit a shell script named *Task1-run.sh*. Running the shell script, the task is performed. Please make sure the shell script and code files are organized in the same folder (no subfolders).

Task 2 (10 marks)
You are asked to write a MapReduce program with Python to cluster trips in Trips.txt based on the dropoff locations. Your code should implement k-medoid clustering algorithm known as *Partitioning Around Medoids* (PAM) algorithm which is described below:

1. **Initialize:** randomly select k of the n data points as the medoids.
2. **Assignment**: Associate each data point to the closest medoid.
3. **Update**: For each medoid m and each data point o associated with m, swap m and o, and compute the total cost of the configuration (that is, the average dissimilarity[1] of o to all the data points associated to m). Select the medoid o with the lowest cost of the configuration.
4. **Iteration:** Repeatedly alternating steps 2 and 3 until there is no change in the assignments or after a given number v of iterations.

The code must work for 3 reducers, for different settings of k, and for different settings of v. The value of v and the initial k data points are input of the program using a separate file named as "initialization.txt". An example of the file for $k = 3$ and $v = 10$ looks like:

```
10
85.679    99.074
11.737    11.615
83.802     1.277
```

Your code should work for different settings of k and v. Also, you should write up a shell script named *Task2-run.sh*. Running the shell script, the task is performed where the shell script and code files are in the same folder (no subfolders).

Task 3 (10 marks)
You are required to use what you learned so far to solve a slightly more advanced task. The task is to write a MapReduce program with Python to count the number of trips for each taxi company and sort the companies in ascending order based on the total number of trips.

Both Taxis.txt and Trips.txt will be used and they are stored on HDFS. The code must work for 3 reducers. Also, you should write up a shell script named *Task3-run.sh*. Running the shell script, the task is performed where the shell script and code files are in the folder (no subfolders).

Note that task 3 should have three MapReduce subtasks where the first is a join operation, the second is a counting operation, and the third is sorting. The execution of the three subtasks should be specified in Task3-run.sh. It is illegal to copy Trips.txt and/or Taxis.txt to the local machine and process them.

Submission
Your assignment should follow the requirement below and submit via Canvas > Assignment 1. Assessment declaration: when you submit work electronically, you agree to the assessment declaration:

[1] Using Euclidean distance between two points as the dissimilarity measure.

Format Requirements

Failure to follow the requirements incurs up to 10 marks penalty.

1. If your student ID is s1234567, then please create a zip file named s1234567_BDP_A1.zip.
   - You need to include a "README" file in the zip file. In the README, specify sufficient information on how to run your codes for each task in AWS EMR.
   - The code files and shell scripts for all three tasks are in the same folder (i.e., no subfolders), and then zip the folder.
   - Do not include hadoop-streaming-3.1.4.jar in the zip file.
2. On HDFS, the input files must be in /Input/ and the output must be in /Output/, as follows:
   /Input/Trips.txt
   /Input/Taxis.txt
   /Output/Task1
   /Output/Task2
   /Output/Task3
   Note that the filenames and directory names are case-sensitive. You are asked to follow the name convention strictly, e.g., using "Task1" not "task1".
3. Besides the zip file, organize the codes and the shell scripts of all three tasks in a separate PDF file (copy & paste into a text editor and then save it as a PDF file). Submit the PDF file (so, there are two submissions, one is the zip file, and the other is the PDF file). The PDF file is for Turnitin plagiarism check. Your submission will not be graded if no such a PDF file is submitted or if the PDF does not pass the Turnitin plagiarism check.

Functional Requirements

Failure to follow the requirements incurs up to 5 marks penalty.

- The code must be well written using a good coding style.
- The codes and scripts must come with concise and clear comments to explain the logical flow of the program.

Academic integrity and plagiarism (standard warning)

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge, and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks, and/or ideas of others you have quoted (i.e., directly copied), summarized, paraphrased, discussed, or mentioned in your assessment through the appropriate referencing methods.
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

Submitting material generated by an AI tool as your own work constitutes plagiarism and academic dishonesty. DO NOT simply copy other people's work, it is not difficult for us to detect copied work and we will pursue such cases.

RMIT University treats plagiarism as a very serious offense constituting misconduct. Plagiarism covers a variety of inappropriate behaviours, including:

- Failure to properly document a source.
- Copyright material from the internet or databases
- Collusion between students

For further information on our policies and procedures, please refer to
https://www.rmit.edu.au/students/student-essentials/rights-and-responsibilities/academic-integrity

RMIT UNIVERSITY

Marking Guide

- Late submission results in penalty of 10% marks for (up to) every 24 hours being late.
- If unexpected circumstances affect your ability to complete the assignment, you can apply for special consideration.
  - Requests for special consideration within 7*24 hours, please email the course coordinator directly with supporting evidence.
  - Request for special consideration of more than 7*24 hours must be via the University Special consideration: https://www.rmit.edu.au/students/student-essentials/assessment-and-exams/assessment/special-consideration.

Task 1	0 marks - cannot run on AWS EMR or - no/unreasonable output or - >1 major logic error in the code	1 mark output incorrect due to 1 major logic error in the code or shell script	2-3 marks output incorrect due to >1 minor logic error in the code or shell script	4 marks output incorrect due to 1 minor logic error in the code or shell script	5 marks output correct and no code/script error, concise and clear comments
Task 2	0 marks - cannot run on AWS EMR or - no/unreasonable output or - >1 major logic error in the code	1-3 marks output incorrect due to 1 major logic error in the code or shell script	4-6 marks output incorrect due to >1 minor logic error in the code or shell script	7-8 marks output incorrect due to 1 minor logic error in the code or shell script	9-10 marks output correct and no code/script error, concise and clear comments
Task 3	0 marks - cannot run on AWS EMR or - no/unreasonable output or - >1 major logic error in the code	1-3 marks output incorrect due to 1 major logic error in the code or shell script	4-6 marks output incorrect due to >1 minor logic error in the code or shell script	7-8 marks output incorrect due to 1 minor logic error in the code or shell script	9-10 marks output correct and no code/script error, concise and clear comments
Functional requirement	Failure penalty on functional requirements detailed in the specification				
Format requirement	Failure penalty on format requirements detailed in the specification				

Marking in Practice

- Please thoroughly test your program prior to submission, as markers are not expected to modify any code (.py or .sh) you submit. For example, if your Bash script contains improper comments that prevent it from running, you will lose marks.
- The data files provided in this assessment do not represent big data; they are intended solely to verify the accuracy of your program's output. Nonetheless, always design your program with the potential to handle real big data scenarios in mind. The marker may test your code with a larger data set.