**Instructions:**

This project involves performing distributed analytics on a large-scale dataset using Python on a Hadoop cluster. The dataset is a subset of Yelp's businesses, reviews, and user data.

There are four questions in this project. You will write four sets of Python scripts that implement four MapReduce algorithms. The first set of scripts constructs an inverted index of the business categories; the second set computes frequency distributions about users; the third set finds top-K records; the fourth set analyzes the relationships between check-ins and businesses. The dataset can be found and downloaded [here](https://www.kaggle.com/yelp-dataset/yelp-dataset/version/3) (registration to Kaggle is required).

* All scripts are to be written using Python >= 3.7.0.
* You should use LF line terminators in your scripts.
* **You may NOT change the shebang line(#!/usr/local/bin/python3.9) of the Python scripts in your submission**.
* **You may NOT use any third-party packages. Only built-in Python libraries are allowed**.
* **You should round your answer to at most 2 decimal places (only if necessary) using the built-in round() function.**

Important notes:

* You should strictly follow the implementation instructions, the input/output format, and the filename. Implementations that do not follow the correct format will be marked as 0.
* You may NOT change the filename of the Python scripts and the output files (if any).
* **The time limit for each set of scripts (MapReduce algorithm) is 360 seconds. If your script goes beyond the time limit, it will be terminated and considered as nothing.**
* **your code must be well-commented.**

**Hint: It is highly recommended to test your implementations in the Docker-based Hadoop environment. We  will use the same environment to evaluate your submission. You need to zip your repository and submit as one zip file with the name of *project2.zip*.** **by the due date. The directory structure in *project2.zip* should look like this:**

EECS4415\_Project\_2/

├─ q1mapper.py

├─ q1reducer.py

├─ q2mapper.py

├─ q2reducer.py

├─ q3mapper.py

├─ q3reducer.py

├─ q4mapper.py

├─ q4reducer.py

├─ info.txt

├─ README.md

├─ Q1.png

├─ .gitignore

**Hint: Please note that you should not include any output files in your submission. We will execute your codes to generate the outputs and then compare them with the correct answers.**

Your scripts will be executed in **a Hadoop cluster (Apache Hadoop 3.3.1)**. An automated judge will programmatically evaluate your solution. Please refer to each question for the detailed breakdown of project marks

**Task 1)**

In information retrieval, an inverted index is an index data structure that stores a mapping from words to a collection of documents that they appear in. Your task is to build an inverted index that maps categories (of businesses) to businesses. In other words, given a collection of businesses, an inverted index is a dictionary where each category is associated with a list of the business ids (comma-separated) that belong to that category

Diagram

Description automatically generated

in this question, your task is to write two Python scripts (***q1mapper.py*** and ***q1reducer.py***) that implement a MapReduce algorithm for constructing the inverted index. For a given collection of Yelp business data (*[yelp\_academic\_dataset\_business.json](https://www.kaggle.com/yelp-dataset/yelp-dataset/version/3?select=yelp_academic_dataset_business.json)*), the algorithm computes the inverted index of the categories to businesses that are **open on the weekend** (Saturdays, Sundays, or both). The output of the MapReduce job should consist of a number of lines in the following format:

Chart

Description automatically generated with low confidence

Namely, each line should contain a category and a list of indexes, separated by **a tab character** (\t). The order does not matter.

For example:

Hotels & Travel ['01OmT3PXceLBRsYyWdummy', '023Lc6yStQnKuqymEdummy', '03PyMX1yFb\_VYgPw4dummy']

Canadian (New) ['04L7iPGEirkbMLjCfdummy', '05IsY\_MFfDBY7JUXgdummy', '06kN4XJq0TtubaPyqdummy', '07sJIkOlJsO1e1PHtdummy']

Your script should be run as follows:

$ yarn jar /path-to-jar/hadoop-streaming-3.3.1.jar -files q1mapper.py,q1reducer.py -mapper /path-to-mapper/q1mapper.py -reducer /path-to-reducer/q1reducer.py -input /path-to-json-in-hdfs/yelp\_academic\_dataset\_business.json -output /path-to-directory-in-hdfs/output

For example:

$ yarn jar /opt/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.1.jar -files q1mapper.py,q1reducer.py -mapper ./q1mapper.py -reducer ./q1reducer.py -input /yelp\_academic\_dataset\_business.json -output /output

**Task 2)**

In this question, your task is to write two Python scripts (*q2mapper.py* and *q2reducer.py*) that implement a MapReduce algorithm for computing frequencies. For a given collection of Yelp user data (*[yelp\_academic\_dataset\_user.json](https://www.kaggle.com/yelp-dataset/yelp-dataset/version/3?select=yelp_academic_dataset_user.json)*), the algorithm computes the percent proportion of Yelp accounts created in each month (irrespective of the year). For example, the input dataset contains four users who joined Yelp on 2010-01-01, 2011-02-01, 2018-03-01, and 2018-04-01, respectively. In this case, the percent proportions should be 25% for January, 25% for February, 25% for March, 25% for April, and 0% for the other months. The output of the MapReduce job should be one line per pair of values separated by a tab character (\t) as follows:

month(integer) proportion%

For example:

Table

Description automatically generated with medium confidence

Due to rounding, the percentages may not add up to 100%.

Your script should be run as follows:

$ yarn jar /path-to-jar/hadoop-streaming-3.3.1.jar -files q2mapper.py,q2reducer.py -mapper /path-to-mapper/q2mapper.py -reducer /path-to-reducer/q2reducer.py -input /path-to-json-in-hdfs/yelp\_academic\_dataset\_user.json -output /path-to-directory-in-hdfs/output

For example:

$ yarn jar /opt/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.1.jar -files q2mapper.py,q2reducer.py -mapper ./q2mapper.py -reducer ./q2reducer.py -input /yelp\_academic\_dataset\_user.json -output /output

**Task 3)**

Yelp users can vote a review as useful, funny, or cool (UFC). The total number of UFC votes of a review is the sum of useful, funny, and cool votes it has received. For example, a review has received 10 useful votes, 20 funny votes, and 30 cool votes. In this case, the total number of UFC votes is 60 for this review. For a given collection of Yelp review data (*[yelp\_academic\_dataset\_review.json](https://www.kaggle.com/yelp-dataset/yelp-dataset/version/3?select=yelp_academic_dataset_review.json)*), your task is to write two Python scripts (*q3mapper.py* and *q3reducer.py*) that implement a MapReduce algorithm to find the top 4415 reviews with the most UFC votes in descending order (from the most UFC votes to the least). If there are multiple reviews with the same number of UFC votes, they should be sorted in descending order according to the date created (from the most recently created to the least). The output of the MapReduce job should be one line per pair of values separated by a tab character (\t) as follows:

review\_id #UFC\_votes

For example:

Table

Description automatically generated with medium confidence

In the example above, the reviews HhG2x0157t1aKqSxEdummy and IiG2x0157t1aKqSxEdummy have the same number of UFC votes. The review HhG2x0157t1aKqSxEdummy comes before IiG2x0157t1aKqSxEdummy because the review HhG2x0157t1aKqSxEdummy was created later than IiG2x0157t1aKqSxEdummy.

Your script should be run as follows:

$ yarn jar /path-to-jar/hadoop-streaming-3.3.1.jar -files q3mapper.py,q3reducer.py -mapper /path-to-mapper/q3mapper.py -reducer /path-to-reducer/q3reducer.py -input /path-to-json-in-hdfs/yelp\_academic\_dataset\_review.json -output /path-to-directory-in-hdfs/output

For example:

$ yarn jar /opt/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.1.jar -files q3mapper.py,q3reducer.py -mapper ./q3mapper.py -reducer ./q3reducer.py -input /yelp\_academic\_dataset\_review.json -output /output

The original file provided by Kaggle (*[yelp\_academic\_dataset\_review.json](https://www.kaggle.com/yelp-dataset/yelp-dataset/version/3?select=yelp_academic_dataset_review.json)*) will be used for evaluation

**Task 4)**

The Yelp check-in data (*[yelp\_academic\_dataset\_checkin.json](https://www.kaggle.com/yelp-dataset/yelp-dataset/version/3?select=yelp_academic_dataset_checkin.json)*) contains the information about check-ins on businesses, in which each line consists of a business ID and a comma-separated list of timestamps for each check-in. In this question, your task is to write two Python scripts (*q4mapper.py* and *q4reducer.py*) that implement a MapReduce algorithm to generate check-in logs in a distributed manner. Each line of the check-in logs should contain a unique identifier (UID), the timestamp of the check-in, and the name of the business. The order of the log does not matter. The UID should follow the same format as the Yelp dataset, which is a random-generated unique string with 22 characters. The valid characters to construct UID include: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), underscore (\_), and dash (-). The output of the MapReduce job should be one line per triplet (separated by a tab character) as follows:

uid timestamp business\_name

For example:

cXAC0I4CEeykaetxJdummy 2022-02-21 17:02:08 York University

cXAD8o4CEeyPW-txJdummy 2022-02-22 14:07:04 York University

cXAH6Y4CEey\_COtxJdummy 2022-05-15 06:28:29 Lassonde School of Engineering

cXAGJo4CEeyuZutxJdummy 2022-03-04 00:17:43 Lassonde School of Engineering

TvpK-Y4CEeyr1DUGsdummy 2022-01-25 03:19:54 YYZ

TvpLQ44CEeyRhDUGsdummy 2022-01-25 04:34:30 YYZ

TvpLjo4CEeyigDUGsdummy 2022-01-26 03:53:17 YYZ

...

Two input files ([*yelp\_academic\_dataset\_checkin.json*](https://www.kaggle.com/yelp-dataset/yelp-dataset/version/3?select=yelp_academic_dataset_checkin.json) and [*yelp\_academic\_dataset\_business.json*](https://www.kaggle.com/yelp-dataset/yelp-dataset/version/3?select=yelp_academic_dataset_business.json)) will be passed to the MapReduce job. Your script should be run as follows:

$ yarn jar /path-to-jar/hadoop-streaming-3.3.1.jar -files q4mapper.py,q4reducer.py -mapper /path-to-mapper/q4mapper.py -reducer /path-to-reducer/q4reducer.py -input /path-to-json-in-hdfs/yelp\_academic\_dataset\_business.json -input /path-to-json-in-hdfs/yelp\_academic\_dataset\_checkin.json -output /path-to-directory-in-hdfs/output

For example:

$ yarn jar /opt/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.1.jar -files q4mapper.py,q4reducer.py -mapper ./q4mapper.py -reducer ./q4reducer.py -input /yelp\_academic\_dataset\_business.json -input /yelp\_academic\_dataset\_checkin.json -output /output

The original files provided by Kaggle ([*yelp\_academic\_dataset\_checkin.json*](https://www.kaggle.com/yelp-dataset/yelp-dataset/version/3?select=yelp_academic_dataset_checkin.json) and [*yelp\_academic\_dataset\_business.json*](https://www.kaggle.com/yelp-dataset/yelp-dataset/version/3?select=yelp_academic_dataset_business.json)) will be used for evaluation.