Assignment 1 – MapReduce

Due: Wednesday 02/19/2020 at 11:55PM

(NO LATE SUBMISSIONS WILL BE ACCEPTED)

Goals

In this assignment, you will explore taxi data with MapReduce. There are three main tasks, all of which are to be done using Hadoop Streaming (Hadoop version 2.x) and Python. For each of the tasks and subtasks, you are to use only **a *single* reducer**.

You will, first, run your programs on Dumbo using the sample dataset to test and debug your code, and then use the large dataset to generate the results for your submission.

Data

The data can be found on Dumbo's HDFS under the `/user/hc2660/hw1data` directory.

Use command `hfs -get` to grab the data to your local account.

The directory includes a subdirectory called `samples` which have three small datasets for you to test your code. The five csv files under the root directory is for you to run and generate results for submission.

Submission Instructions

One **.zip file** with the following structure:

1. One directory per task, named `taskX`, where X is the task's number.
2. If a task has a sub-task, use `taskX-Y`, where Y identifies the sub-task.

   E.g.: "task2-a" refers to sub-task "a" of Task 2. If you do an `ls` on your homework directory, it should contain the following sub-directories:

   ```
   task1
   task2-a
   task2-b
   task2-c
   task2-d
   task2-e
   task2-f
   task3
   ```

Inside each subdirectory:

1. Include the output directory generated by Hadoop (use the naming specified below),
2. Include your Python scripts used by your map-reduce job,
3. Name your map script as "map.py", and your reduce script as "reduce.py",
4. You may use combiners and partitioners, but they will not be tested, i.e., your program should work correctly *without* combiners or partitioners,
5. Your `map.py` codes are permitted to access the `mapreduce_map_input_file` environment variable in order to determine which input (.csv) file is being read. The CSV filenames will contain the substrings "trip", "fare", and "license". You may not use any other environment variables besides mapreduce_map_input_file.
6. DO NOT submit the input data

Tasks

Task 1: Write one map-reduce job that joins the 'trips' and 'fare' data (taxi data).

The 'fares' and 'trips' data share 4 attributes:

```
medallion, hack_license, vendor_id, pickup_datetime.
```

The join MUST BE a reduce-side inner join.

Output: A key-value pair per line. Use a "tab" to separate key and value, a comma in between

key: `medallion, hack_license, vendor_id, pickup_datetime`
value: the remaining attributes of 'trips' data in their original order
and
the remaining attributes of 'fare' data in their original order

You must respect this ordering requirement!
Refer to the .txt attachment for output samples of Task 1.

The *output directory* produced by Hadoop must be named `TripFareJoin`. The contents of `task1` subdirectory looks like:

```
ls -F task1/
TripFareJoin/    map.py*        reduce.py*
```

Task 2: Write map-reduce jobs for each of the following sub-tasks, using the output of Task 1 (joined data) as input for all these subtasks:

(Similar to Task 1, you must use a tab to separate the key and the value in the output tuples.)

a) Find the distribution of fare amounts (`fare_amount`) for each of the following ranges:

[0, 4],	[20.01, 24],	[40.01, 44],
[4.01, 8],	[24.01, 28],	[44.01, 48],
[8.01, 12],	[28.01, 32],	[48.01, infinite].
[12.01, 16],	[32.01, 36],	
[16.01, 20],	[36.01, 40],	

Thus, for each range, give the number of trips whose fare amount falls in that range.

Output: A key-value pair per line, where the key is the range, and the value is the number of trips. For example,

```
0,4   100
4.01,8   300
…
```

The output directory must be named `FareAmounts`.

b) Find the number of trips whose cost is *less than or equal* to $10 (total_amount).

Output: The number of trips.

The output directory must be named `TripAmount`.

c) Find the distribution of the number of passengers, i.e., for each number of passengers A, the number of trips that had A passengers.

Output: A key-value pair per line, where the key is the number of passengers, and the value is the number of trips.

The output directory must be named `NumberPassengers`.

d) Find the total revenue (for all taxis) and the total amount spent on tolls, **per day** (from `pickup_datetime`). Revenue should include the fare amount, tips, and surcharges.

Output: A key-value pair per line, where the key is the day `YYYY-MM-DD`, and the value contains the total revenue and the total tolls for that day, in this order.
Use two decimal digits, e.g., 3.02245 should be represented as 3.02. For example,

```
2016-01-01    100000.02,11000.00
2016-01-02    202000.00,1000.00
```

The output directory must be be named `TotalRevenue`.

e) For each taxi (`medallion`) find the **total** number of trips, and the **average** number of trips **per day**. For the average trips per day, use 2 decimal digits.

Your average should be over all days that the taxi drove, e.g., if the input data has entries for a given taxi on 6 different days, your average should be over 6 days.

Output: A key-value pair per line, where the key is the medallion, and the value contains the total number of trips and the average number of trips per day.

The output directory must be named `MedallionTrips`.

f) Find the number of different taxis (`medallion`) used by each driver (`license`).

Output: A key-value pair per line, where the key is the driver, and the value is the number of different taxis used by that driver.

The output directory must be named `UniqueTaxis`.

Task 3: Write a map-reduce job that joins the output from Task 1 with the vehicle data.

Note that they both share the `medallion` attribute.

The join MUST BE a reduce-side inner join.

Output: A key-value pair per line, where

- key: `medallion`
- value: remaining attributes of Task 1 output (with the remaining keys) in original order and
  remaining attributes of vehicle data in their original order

  You must follow this ordering requirement!

The output directory produced by Hadoop should be named `VehicleJoin`.

The End