Metropolitan State University
ICS 432 - 01: Distributed and Cloud Computing
Fall 2021

Assignment 3: MapReduce
Total points: 40
Out: Saturday, November 6, 2021
Due: 11:59 PM on Friday, November 19, 2021

Objective

The objective of this homework is to practice solving problems using the map reduce programming paradigm. You are only required to implement the problems on this homework on one machine (either on Cloud9, GCP editor, or locally on your computer if you have a Python IDE).

What to submit?

In this homework you will solve several problems using the map reduce programming paradigm. Include the following in your homework submission for each exercise:

1- Plan of your map reduce solution that include the output key and value from the mapper and the function of the reducer. Explain your plan with an example of 5 rows at least.
2- A screenshot of your `mapper.py`.
3- A screenshot of `reducer.py`.
4- A screenshot of the output of the map-reduce job.

Exercise 1: Analyzing Stock Data

In this exercise, you are given a file called `NYSE.csv` that include data about **N**ew **Y**ork **S**tock **E**xchange. The following figure is a screenshot of a part of the file.

```
1  NYSE,GRT,4/27/2006,16.87,17,16.87,17,17500,3.75
2  NYSE,GCH,1/31/2008,10.59,10.76,10.08,10.19,190000,4.73
3  NYSE,GWW,4/27/2012,54.7,57.09,54.6,56.07,453200,49.1
4  NYSE,GGG,11/29/2001,22.75,22.75,22.62,22.62,29600,1.17
5  NYSE,GWW,5/22/1998,53.43,55.12,53,54.91,361600,9.7
6  NYSE,GLT,4/9/2007,14.08,14.19,13.75,14.19,16500,8.53
```

Examine the file contents and familiarize yourself with the format of each row. Basically, each row in the file consists of the following nine comma-separated fields:

1. exchange
2. stock_symbol
3. date
4. stock_price_open
5. stock_proce_high
6. stock_price_low
7. stock_price_close
8. stock_volume
9. stock_proce_adj_close

Task 1.1

Implement a map reduce job to find how many lines are there for each stock_symbol. Note that this is very similar to the word count example. To read the individual attributes from a row, `split` the row on ',' and the output of the split is an array and you can access an attribute using its index. For example, stock_symbol attribute is at index 1.

Name the files in this job as `mapper1.py` and `reducer1.py`.

Task 1.2

Implement a map reduce job to find the maximum trading price for each stock_symbol. Name the files for this task as `mapper2.py` and `reducer2.py`.

Task 1.3

Implement a map reduce job to find the highest price for each stock but consider only records that have a volume greater than 250,000. Name the files for this task as `mapper3.py` and `reducer3.py`.

Task 1.4

Implement a map reduce job to find the highest price for each stock for each year (Hint: the mapper output key is a combination of stock_symbol and year). Name the files for this task as `mapper4.py` and `reducer4.py`.

To extract the year part from the date field, use Python's `datetime` library.

Task 1.5

The goal of this task is to find how distinct stock_symbols are there in the data set. One way to implement this task is to use two consecutive map reduce jobs as follows:

1- Job 1: the first map reduce job is similar to Task 1 where the output of this job includes only one row for each stock symbol.
2- Job 2: Implement another map reduce job (in files `mapper5.py` and `reducer5.py`) where the mapper outputs one row for each input row and the both the key and values of the mapper outputs are set to 1. The reducer then will receive all inputs with key 1 and the reducer just adds the values.

Assume for example, the following is the output of Job 1:

```
GRT   5
GCH   3
GGG   2
```

`mapper5.py` reads these three lines as input and emits the following:

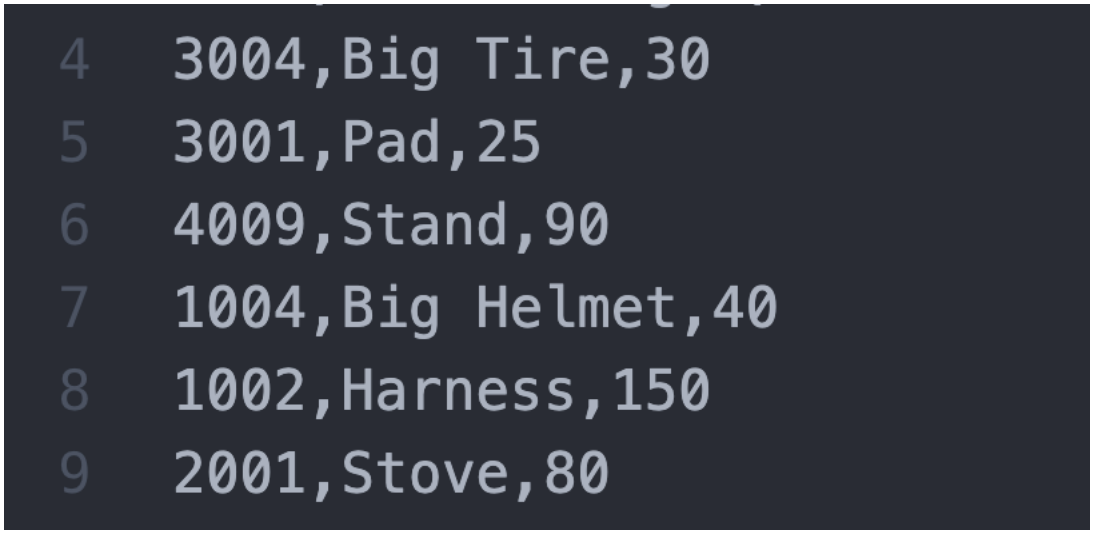
```
1     1
1     1
1     1
```

`Reducer5.py` then receives all inputs with the same key and adds the values. The output of `reducer5.py` is then as follows

```
1     3
```

Exercise 2: Implementing Relational Union and Intersection using map-reduce

In this exercise, you are given one input folder called `stores` that includes two files, named `store1.txt` and `store2.txt`, where the files include product information from two stores. The following is a screenshot of a part of one file.



Each row in the file consists of the three attributes, namely `product ID`, `product description`, and `price`, and product ID is a primary key which means that there are no two lines in the same file with the same product ID.

Task 2.1

Implement a map reduce job to find the union of the two files. Note that union produces in the output a record if the record exists in at least one of the input files. Note also that a record is produced in the output once even if it appears in both input files. The output should include product id as key and product price as value.

For example:

[A,B,C,D] union [A,B,F,E] →[A,B,C,D,F,E]

Task 2.2

Implement a map reduce job to find the intersection of the two files. Note that intersection produces in the output a record if the record exists in both input files. Note also that a record is produced in the output once. The output should include product id as key and product price as value.

For example:

[A,B,C,D] intersection [A,B,F,E] →[A,B]