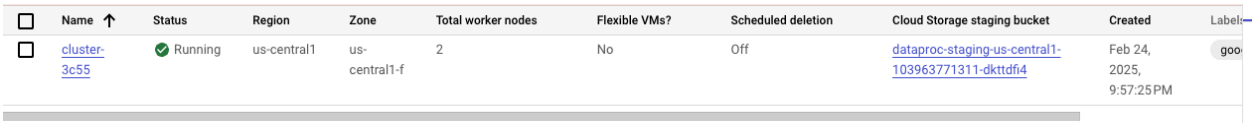
1. [2 points] Acquire the cluster.[1]



Name	Status	Region	Zone	Total worker nodes	Flexible VMs?	Scheduled deletion	Cloud Storage staging bucket	Created	Label
cluster-3c55	Running	us-central1	us-central1-f	2	No	Off	dataproc-staging-us-central1-103963771311-dkttdfi4	Feb 24, 2025, 9:57:25 PM	goo

2. [2 points] Load the five books into the master, move the data into HDFS.

Using “put” I moved the files from local to hadoop.

As you can see in the screenshot, the files are there.

```
hyunsu_lee@cluster-3c55-m:~$ hadoop fs -ls /user/lee/five-books
Found 5 items
-rw-r--r--   2 hyunsu_lee hadoop     179903 2025-02-25 03:13 /user/lee/five-books/a_tangled_tale.txt
-rw-r--r--   2 hyunsu_lee hadoop     173379 2025-02-25 03:13 /user/lee/five-books/alice_in_wonderland.txt
-rw-r--r--   2 hyunsu_lee hadoop     394246 2025-02-25 03:13 /user/lee/five-books/sylvie_and_bruno.txt
-rw-r--r--   2 hyunsu_lee hadoop     458755 2025-02-25 03:13 /user/lee/five-books/symbolic_logic.txt
-rw-r--r--   2 hyunsu_lee hadoop     135443 2025-02-25 03:13 /user/lee/five-books/the_game_of_logic.txt
```

[1] Be sure to enable the cloud-platform scope for the cluster (found under the “Manage security” menu when creating the Dataproc cluster on Compute Engine.

3. [2 points] *Without writing any code of your own*, verify that you have a good installation of hadoop by running wordcount on five-books.

I ran the code on github, and the results are partitioned. You can see the log successfully ran.

```
2025-02-25 03:17:58,758 INFO mapreduce.Job: Job job_1740452333082_0001 completed successfully
2025-02-25 03:17:58,843 INFO mapreduce.Job: Counters: 56
        File System Counters
                FILE: Number of bytes read=596708
                FILE: Number of bytes written=4642923
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1342361
                HDFS: Number of bytes written=313829
                HDFS: Number of read operations=50
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=21
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Killed map tasks=1
                Killed reduce tasks=1
                Launched map tasks=6
                Launched reduce tasks=7
                Data-local map tasks=6
                Total time spent by all maps in occupied slots (ms)=229821364
                Total time spent by all reduces in occupied slots (ms)=234822486
                Total time spent by all map tasks (ms)=67874
                Total time spent by all reduce tasks (ms)=69351
                Total vcore-milliseconds taken by all map tasks=67874
                Total vcore-milliseconds taken by all reduce tasks=69351
                Total megabyte-milliseconds taken by all map tasks=229821364
                Total megabyte-milliseconds taken by all reduce tasks=234822486
        Map-Reduce Framework
                Map input records=35119
                Map output records=219095
                Map output bytes=2091576
                Map output materialized bytes=596876
                Input split bytes=635
                Combine input records=219095
                Combine output records=42051
                Reduce input groups=29287
                Reduce shuffle bytes=596876
                Reduce input records=42051
                Reduce output records=29287
                Spilled Records=84102
                Shuffled Maps =35
                Failed Shuffles=0
                Merged Map outputs=35
                GC time elapsed (ms)=707
                CPU time spent (ms)=18320
                Physical memory (bytes) snapshot=6081519616
                Virtual memory (bytes) snapshot=57912176640
                Total committed heap usage (bytes)=5570035712
                Peak Map Physical memory (bytes)=628695040
                Peak Map Virtual memory (bytes)=4831883264
                Peak Reduce Physical memory (bytes)=464277504
                Peak Reduce Virtual memory (bytes)=4832751616
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
```

```
hyunsu_lee@cluster-3c55-m:~$ hadoop fs -get /books-count-0
hyunsu_lee@cluster-3c55-m:~$ ls -la books-count-0
total 328
drwxr-xr-x 2 hyunsu_lee hyunsu_lee  4096 Feb 25 03:19 .
drwxr-xr-x 5 hyunsu_lee hyunsu_lee  4096 Feb 25 03:19 ..
-rw-r--r-- 1 hyunsu_lee hyunsu_lee     0 Feb 25 03:19 _SUCCESS
-rw-r--r-- 1 hyunsu_lee hyunsu_lee 44296 Feb 25 03:19 part-r-00000
-rw-r--r-- 1 hyunsu_lee hyunsu_lee 45217 Feb 25 03:19 part-r-00001
-rw-r--r-- 1 hyunsu_lee hyunsu_lee 44738 Feb 25 03:19 part-r-00002
-rw-r--r-- 1 hyunsu_lee hyunsu_lee 45538 Feb 25 03:19 part-r-00003
-rw-r--r-- 1 hyunsu_lee hyunsu_lee 44404 Feb 25 03:19 part-r-00004
-rw-r--r-- 1 hyunsu_lee hyunsu_lee 45732 Feb 25 03:19 part-r-00005
-rw-r--r-- 1 hyunsu_lee hyunsu_lee 43904 Feb 25 03:19 part-r-00006
```

4. [2 points] Run wordcount using the provided **`mapper_noll.py`** and the default reducer **`aggregate`**.

Changed code to name of my directory and mapper file to mapper_noll.py from the whole file directory (error occurred with full directory).
The code was implemented successfully as you can see in the log.

```
hyunsu_lee@cluster-3c55-m:~$ mapred streaming -files /home/hyunsu_lee/big-data-repo/hadoop/mapper_noll.py \
                -mapper mapper_noll.py \
                -input /user/lee/five-books \
                -reducer aggregate \
                -output /books-count-m-3
```

```
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1422276
                HDFS: Number of bytes written=103696
                HDFS: Number of read operations=107
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=21
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Killed map tasks=1
                Killed reduce tasks=1
                Launched map tasks=24
                Launched reduce tasks=7
                Data-local map tasks=24
                Total time spent by all maps in occupied slots (ms)=940095812
                Total time spent by all reduces in occupied slots (ms)=226611436
                Total time spent by all map tasks (ms)=277642
                Total time spent by all reduce tasks (ms)=66926
                Total vcore-milliseconds taken by all map tasks=277642
                Total vcore-milliseconds taken by all reduce tasks=66926
                Total megabyte-milliseconds taken by all map tasks=940095812
                Total megabyte-milliseconds taken by all reduce tasks=226611436
        Map-Reduce Framework
                Map input records=35119
                Map output records=207438
                Map output bytes=4167234
                Map output materialized bytes=870364
                Input split bytes=2726
                Combine input records=207438
                Combine output records=35772
                Reduce input groups=10201
                Reduce shuffle bytes=870364
                Reduce input records=35772
                Reduce output records=10201
                Spilled Records=71544
                Shuffled Maps =168
                Failed Shuffles=0
                Merged Map outputs=168
                GC time elapsed (ms)=1825
                CPU time spent (ms)=41480
                Physical memory (bytes) snapshot=17538666496
                Virtual memory (bytes) snapshot=149628276736
                Total committed heap usage (bytes)=17595105280
                Peak Map Physical memory (bytes)=626577408
                Peak Map Virtual memory (bytes)=4837695488
                Peak Reduce Physical memory (bytes)=429150208
                Peak Reduce Virtual memory (bytes)=4834439168
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=1419550
        File Output Format Counters
                Bytes Written=103696
2025-02-25 04:28:27,369 INFO streaming.StreamJob: Output directory: /books-count-m-3
```

5. [2 points] Run wordcount using the provided **`mapper_noll.py`** and the provided reducer **`reducer_noll.py`**.

Ran the mapper and reducer in the github repo.

Successfully ran the map-reduce.

```
hyunsu_lee@cluster-3c55-m:~$ mapred streaming -files /home/hyunsu_lee/big-data-repo/hadoop/mapper_noll.py,/home/hyunsu_
lee/big-data-repo/hadoop/reducer_noll.py \
                -mapper  mapper_noll.py \
                -reducer reducer_noll.py \
                -input /user/lee/five-books \
                -output /books-counts-mr-0
```

```
                FILE: Number of bytes written=18190037
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1422276
                HDFS: Number of bytes written=236309
                HDFS: Number of read operations=107
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=21
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Killed map tasks=1
                Killed reduce tasks=1
                Launched map tasks=24
                Launched reduce tasks=7
                Data-local map tasks=24
                Total time spent by all maps in occupied slots (ms)=897533792
                Total time spent by all reduces in occupied slots (ms)=245837144
                Total time spent by all map tasks (ms)=265072
                Total time spent by all reduce tasks (ms)=72604
                Total vcore-milliseconds taken by all map tasks=265072
                Total vcore-milliseconds taken by all reduce tasks=72604
                Total megabyte-milliseconds taken by all map tasks=897533792
                Total megabyte-milliseconds taken by all reduce tasks=245837144
        Map-Reduce Framework
                Map input records=35119
                Map output records=207438
                Map output bytes=4167234
                Map output materialized bytes=4583118
                Input split bytes=2726
                Combine input records=0
                Combine output records=0
                Reduce input groups=10201
                Reduce shuffle bytes=4583118
                Reduce input records=207438
                Reduce output records=10201
                Spilled Records=414876
                Shuffled Maps =168
                Failed Shuffles=0
                Merged Map outputs=168
                GC time elapsed (ms)=2167
                CPU time spent (ms)=43570
                Physical memory (bytes) snapshot=17409654784
                Virtual memory (bytes) snapshot=149641244672
                Total committed heap usage (bytes)=17561550848
                Peak Map Physical memory (bytes)=673181696
                Peak Map Virtual memory (bytes)=4840419328
                Peak Reduce Physical memory (bytes)=437030912
                Peak Reduce Virtual memory (bytes)=4848148480
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=1419550
        File Output Format Counters
                Bytes Written=236309
```

6. [4 points] Modify the provided **`reducer_noll.py`** such that the resulting words don't all begin with "LongValueSum:"

Using "nano", I changed the code.

Added the lines to the original code:
if word.startwith('LongValueSum'):
    Word = word[13:]
You can see the changed code below.

```
hyunsu_lee@cluster-3c55-m:~$ cat /home/hyunsu_lee/big-data-repo/hadoop/reducer_noll.py
#!/usr/bin/env python
"""reducer.py"""

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)
    if word.startswith('LongValueSum'):
        word = word[13:]

    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue
    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # write result to STDOUT
            print ('%s\t%s' % (current_word, current_count))
        current_count = count
        current_word = word

# do not forget to output the last word if needed!
if current_word == word:
    print ('%s\t%s' % (current_word, current_count))
```

It's important to change the permission.

```
hyunsu_lee@cluster-3c55-m:~$ chmod +x /home/hyunsu_lee/big-data-repo/hadoop/reducer_noll.py
```

Ran the map-reduce process.

```
hyunsu_lee@cluster-3c55-m:~$ mapred streaming   -files /home/hyunsu_lee/big-data-repo/hadoop/mapper_noll.py,/home/hyuns
u_lee/big-data-repo/hadoop/reducer_noll.py   -mapper mapper_noll.py   -reducer reducer_noll.py   -input /user/lee/five-
books   -output /books-counts-clean-1
```

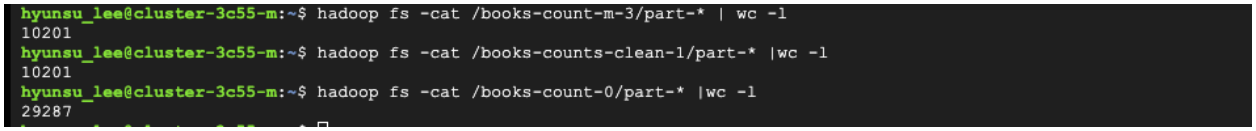
And successfully cleaned out the prefixes. Below are some of the examples.

```
hyunsu_lee@cluster-3c55-m:~$ hadoop fs -cat /books-counts-clean-1/part-* | head -10
a       4736
abate   1
abc     5
abcd    2
able    42
abounds 1
absolutely      9
absorbed        5
abstract        31
absurdity       2
```

7. [3 points] How many unique words were found by Q2.3, Q2.4 & Q2.6, respectively?

Based on the counting with wc-l.

- Q2.3 : 29287
- Q2.4 : 10201
- Q2.6 : 10201



8. [3 points] What accounts for the difference between {Q2.3 & Q2.4}, {Q2.4 & Q2.7} and {Q2.7 & Q2.3}, respectively.

{Q2.4 & Q2.7} have the same number, and it is expected since they use the same mapper which determines which words are identified.

The difference between Q2.3 and the others shows that mapper_noll.py processes text differently than Hadoop's built-in wordcount.

3. Analyzing Server Logs [55 points]

1. [6+9=15 points[2]] What is the percentage of each request type (GET, PUT, POST, etc.)?

```
1. Request Type Statistics:
-------------------------
Total Requests: 78244

Method  Count   Percentage
------  -----   ----------
HEAD    253     0.32%
POST    44584   56.98%
GET     33407   42.70%
```

2. [6+9=15 points] What percent of the responses fall into each of the following five types?
    a. Informational responses (100–199)
    b. Successful responses (200–299)
    c. Redirection messages (300–399)
    d. Client error responses (400–499)
    e. Server error responses (500–599)

```
2. Response Category Statistics:
------------------------------
Total Responses: 78244

Category        Count   Percentage
--------        -----   ----------
ClientError     4634    5.92%
Successful      70681   90.33%
Redirection     2929    3.74%
```

The rest of the types were not there and these three made up all of the errors.

3. [9+16=25 points] What 5 IP addresses generate the most client errors?

[2] The construct a+b=c is taken to mean *a* points for the mapper, *b* points for the reducer and *c* points for the total.

IP Address	Error Count	
173.255.176.5	2059	44.43%
212.9.160.24	126	2.72%
13.77.204.88	78	1.68%
51.210.243.185	58	1.25%
193.106.30.100	53	1.14%

Mapper code to parse out the information in the log.

```
hyunsu_lee@cluster-3dd0-m:~$ cat /home/hyunsu_lee/log_parser_mapper.py
#!/usr/bin/env python3
import re
import sys

# Regular expression to parse the Apache access log format
LOG_PATTERN = re.compile(r'([\d\.]+) - - \[(.*?)\] "([A-Z]+) (.*?) HTTP/[\d\.]+?" (\d+) (\d+|-) "(.*?)" "(.*?)" "(.*?)"')

# Input comes from STDIN (standard input)
for line in sys.stdin:
    line = line.strip()
    match = LOG_PATTERN.match(line)

    if match:
        ip = match.group(1)
        method = match.group(3)
        status_code = int(match.group(5))

        # Output for Question 1: Request Type Percentage
        print(f"REQUEST_TYPE\t{method}\t1")

        # Output for Question 2: Response Code Categories
        if 100 <= status_code < 200:
            category = "Informational"
        elif 200 <= status_code < 300:
            category = "Successful"
        elif 300 <= status_code < 400:
            category = "Redirection"
        elif 400 <= status_code < 500:
            category = "ClientError"
        elif 500 <= status_code < 600:
            category = "ServerError"
        else:
            category = "Unknown"

        print(f"RESPONSE_CATEGORY\t{category}\t1")

        # Output for Question 3: Top 5 IPs with client errors
        if 400 <= status_code < 500:
            print(f"CLIENT_ERROR_IP\t{ip}\t1")
```

Explanation for the Mapper code

1. Regular Expression Pattern

```
LOG_PATTERN = re.compile(r'([\d\.]+) - - \[(.*?)\] "([A-Z]+) (.*?) HTTP/[\d\.]+?" (\d+) (\d+|-) "(.*?)" "(.*?)" "(.*?)"')
```

This regex captures Apache log components:

- Group 1: ([\d\.]+) → IP address (e.g., "13.66.139.0")
- Group 2: (.*?) → Timestamp (e.g., "19/Dec/2020:13:57:26 +0100")
- Group 3: ([A-Z]+) → HTTP method (e.g., "GET", "POST")
- Group 4: (.*?) → Requested URL
- Group 5: (\d+) → Status code (e.g., 200, 404)

- Group 6: (\d+|-) → Response size in bytes or dash
- Group 7-9: Referrer, User-Agent, and extra field

2. Data Processing Flow

For each log line, the mapper:

1. Attempts to match the line against the regex pattern
2. Extracts key data: IP address, HTTP method, status code
3. Emits three types of key-value pairs:

For Question 1 (Request Type Percentages):

```
print(f"REQUEST_TYPE\t{method}\t1")
```

Emits: REQUEST_TYPE GET 1 (for each GET request)

For Question 2 (Response Categories):

```
if 100 <= status_code < 200:
    category = "Informational"
# ...other categories...
print(f"RESPONSE_CATEGORY\t{category}\t1")
```

Emits: `RESPONSE_CATEGORY Successful 1` (for each 200-level response)

For Question 3 (IPs with most client errors):

```
if 400 <= status_code < 500:
    print(f"CLIENT_ERROR_IP\t{ip}\t1")
```

Emits: `CLIENT_ERROR_IP 173.255.176.5 1` (for each 400-level error from this IP)

```
hyunsu_lee@cluster-3dd0-m:~$ cat /home/hyunsu_lee/log_parser_reducer.py
#!/usr/bin/env python3
import sys

current_full_key = None
current_count = 0

# Input comes from STDIN (output of the mapper)
for line in sys.stdin:
    line = line.strip()
    try:
        key_type, key, count = line.split('\t')
        count = int(count)
    except (ValueError, IndexError):
        continue

    full_key = f"{key_type}\t{key}"

    if current_full_key == full_key:
        current_count += count
    else:
        if current_full_key:
            print(f"{current_full_key}\t{current_count}")
        current_full_key = full_key
        current_count = count

# Output the last key
if current_full_key:
    print(f"{current_full_key}\t{current_count}")
```

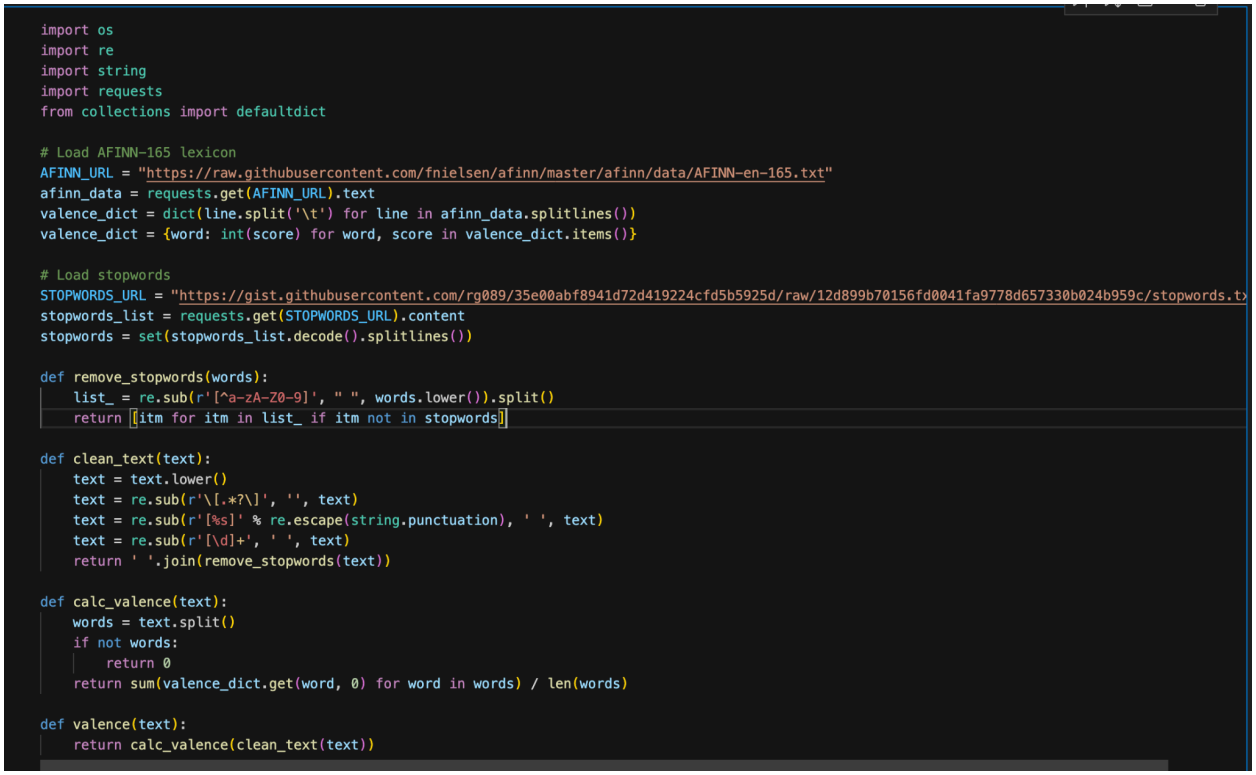
```
hyunsu_lee@cluster-3dd0-m:~$ ./analyze_results.sh

1. Request Type Statistics:
-------------------------
Total Requests: 78244

Method  Count   Percentage
------  -----   ----------
HEAD    253     0.32%
POST    44584   56.98%
GET     33407   42.70%

2. Response Category Statistics:
------------------------------
Total Responses: 78244

Category        Count   Percentage
--------        -----   ----------
ClientError     4634    5.92%
Successful      70681   90.33%
Redirection     2929    3.74%

3. Top 5 IPs Generating Client Errors:
-----------------------------------
Total Client Errors: 4634

IP Address      Error Count     Percentage of All Client Errors
----------      -----------     ---------------------------
173.255.176.5   2059    44.43%
212.9.160.24    126     2.72%
13.77.204.88    78      1.68%
51.210.243.185  58      1.25%
193.106.30.100  53      1.14%
```

```
hyunsu_lee@cluster-3dd0-m:~$ cat analyze_results.sh
#!/bin/bash

# Create temp directory
mkdir -p temp_analysis

# Get MapReduce output
hadoop fs -cat /output/user/lee/log_analysis_results/part-* > temp_analysis/all_results.txt

# Calculate request type counts and percentages (Question 1)
echo -e "\n1. Request Type Statistics:"
echo "--------------------------"
grep "REQUEST_TYPE" temp_analysis/all_results.txt | awk '{print $2, $3}' | \
  awk '{ count[$1] += $2 }
       END {
         for (type in count) { total += count[type] }
         print "Total Requests:", total
         print ""
         print "Method\tCount\tPercentage"
         print "------\t-----\t----------"
         for (type in count) {
           printf "%s\t%d\t%.2f%%\n", type, count[type], (count[type]/total)*100
         }
       }'

# Calculate response category counts and percentages (Question 2)
echo -e "\n2. Response Category Statistics:"
echo "--------------------------------"
grep "RESPONSE_CATEGORY" temp_analysis/all_results.txt | awk '{print $2, $3}' | \
  awk '{ count[$1] += $2 }
       END {
         for (cat in count) { total += count[cat] }
         print "Total Responses:", total
         print ""
         print "Category\tCount\tPercentage"
         print "--------\t-----\t----------"
         for (cat in count) {
           printf "%s\t%d\t%.2f%%\n", cat, count[cat], (count[cat]/total)*100
         }
       }'

# Find top 5 IPs with client errors (Question 3)
echo -e "\n3. Top 5 IPs Generating Client Errors:"
echo "-------------------------------------"
grep "CLIENT_ERROR_IP" temp_analysis/all_results.txt | awk '{print $2, $3}' | \
  awk '{ count[$1] += $2; total += $2 }
       END {
         print "Total Client Errors:", total
         print ""
         print "IP Address\tError Count\tPercentage of All Client Errors"
         print "----------\t----------\t----------------------------"
         n=0
         for (ip in count) {
           arr[n] = ip
           n++
         }
         # Sort by count (descending)
         for (i=0; i<n-1; i++) {
           for (j=i+1; j<n; j++) {
             if (count[arr[i]] < count[arr[j]]) {
               temp = arr[i]
               arr[i] = arr[j]
```

```
# Find top 5 IPs with client errors (Question 3)
echo -e "\n3. Top 5 IPs Generating Client Errors:"
echo "-------------------------------------"
grep "CLIENT_ERROR_IP" temp_analysis/all_results.txt | awk '{print $2, $3}' | \
  awk '{ count[$1] += $2; total += $2 }
       END {
         print "Total Client Errors:", total
         print ""
         print "IP Address\tError Count\tPercentage of All Client Errors"
         print "----------\t----------\t----------------------------"
         n=0
         for (ip in count) {
           arr[n] = ip
           n++
         }
         # Sort by count (descending)
         for (i=0; i<n-1; i++) {
           for (j=i+1; j<n; j++) {
             if (count[arr[i]] < count[arr[j]]) {
               temp = arr[i]
               arr[i] = arr[j]
               arr[j] = temp
             }
           }
         }
         # Print top 5
         for (i=0; i<5 && i<n; i++) {
           printf "%s\t%d\t%.2f%%\n", arr[i], count[arr[i]], (count[arr[i]]/total)*100
         }
       }'

# Clean up
rm -rf temp_analysis
```

3. MapReduce Pattern

1. [7 points] In the mapper (which is given a sequence of lines of speeches as input):
    a. Read each line in each president's folder, and clean & calculate valence using the function *valence.py*. See the code below
2. [6 points] In the reducer (which is given all (president, word valence) key-value pairs with the same key, i.e.president):
    a. Calculates the average at the end.



```python
import os
import re
import string
import requests
from collections import defaultdict

# Load AFINN-165 lexicon
AFINN_URL = "https://raw.githubusercontent.com/fnielsen/afinn/master/afinn/data/AFINN-en-165.txt"
afinn_data = requests.get(AFINN_URL).text
valence_dict = dict(line.split('\t') for line in afinn_data.splitlines())
valence_dict = {word: int(score) for word, score in valence_dict.items()}

# Load stopwords
STOPWORDS_URL = "https://gist.githubusercontent.com/rg089/35e00abf8941d72d419224cfd5b5925d/raw/12d899b70156fd0041fa9778d657330b024b959c/stopwords.t
stopwords_list = requests.get(STOPWORDS_URL).content
stopwords = set(stopwords_list.decode().splitlines())

def remove_stopwords(words):
    list_ = re.sub(r'[^a-zA-Z0-9]', " ", words.lower()).split()
    return [itm for itm in list_ if itm not in stopwords]

def clean_text(text):
    text = text.lower()
    text = re.sub(r'\[.*?\]', '', text)
    text = re.sub(r'[%s]' % re.escape(string.punctuation), ' ', text)
    text = re.sub(r'[\d]+', ' ', text)
    return ' '.join(remove_stopwords(text))

def calc_valence(text):
    words = text.split()
    if not words:
        return 0
    return sum(valence_dict.get(word, 0) for word in words) / len(words)

def valence(text):
    return calc_valence(clean_text(text))
```

```python
def mapper(directory):
    mapped_data = []
    for president_folder in os.listdir(directory):
        president_path = os.path.join(directory, president_folder)
        if os.path.isdir(president_path):
            for filename in os.listdir(president_path):
                if filename.endswith(".txt"):
                    file_path = os.path.join(president_path, filename)
                    with open(file_path, "r", encoding="utf-8") as file:
                        for line in file:
                            line = line.strip()
                            if line:
                                mapped_data.append((president_folder, valence(line)))
    return mapped_data
```

✓ 0.0s Python

```python
def reducer(mapped_data):
    president_scores = defaultdict(list)
    for president, score in mapped_data:
        president_scores[president].append(score)

    president_avg = {pres: sum(scores)/len(scores) for pres, scores in president_scores.items() if scores}
    return president_avg
```

	President	Average Sentiment
0	coolidge	0.112565
1	tyler	0.081399
2	wilson	0.081781
3	ford	0.128761
4	pierce	0.069799
5	lincoln	0.049569
6	washington	0.137754
7	reagan	0.090472
8	hoover	0.040317
9	jefferson	0.038722
10	bharrison	0.070424
11	monroe	0.122910
12	carter	0.049715
13	taft	0.070318
14	madison	0.024648
15	roosevelt	0.033681
16	eisenhower	0.155894
17	buchanan	0.016166
18	lbjohnson	0.068642
19	adams	0.097433
20	arthur	0.052209
21	fillmore	0.091505
22	kennedy	0.071535
23	fdroosevelt	0.043886
24	hayes	0.073810
25	obama	0.125876
26	bush	0.077597
27	johnson	0.018668
28	cleveland	0.038579
29	nixon	0.081092
30	harrison	0.095126
31	taylor	0.117563
32	clinton	0.089824
33	truman	0.125647
34	gwbush	0.081587
35	garfield	0.083695
36	harding	0.096041
37	mckinley	0.093523
38	vanburen	0.070218
39	polk	0.065838
40	grant	0.077659
41	jqadams	0.057082
42	jackson	0.048892

4. Hadoop Errors [15 points]

[7 points] Where (what server & location) did the divide-by-zero error messages show up and how many did you find?

Using “yarn logs”, application id to identify the run, grep, and wc for counting, i found the number of errors and the locations, and server.

- 24 ZeroDivisionError messages
- Server: cluster-134a-w-1.c.cs119-451916.internal
- Location: /var/log/hadoop-yarn/userlogs/application_1740584010684_0008/container_*/stderr

```
2025-02-26 19:47:37,457 INFO mapreduce.Job:  map 100% reduce 100%
2025-02-26 19:47:38,469 INFO mapreduce.Job: Job job_1740584010684_0008 failed with state FAIL
ED due to: Task failed task_1740584010684_0008_m_000004
Job failed as tasks failed. failedMaps:1 failedReduces:0 killedMaps:0 killedReduces: 0

2025-02-26 19:47:38,546 INFO mapreduce.Job: Counters: 14
        Job Counters
                Failed map tasks=23
                Killed map tasks=23
                Killed reduce tasks=7
                Launched map tasks=26
                Other local map tasks=17
                Data-local map tasks=9
                Total time spent by all maps in occupied slots (ms)=870781006
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=257171
                Total vcore-milliseconds taken by all map tasks=257171
                Total megabyte-milliseconds taken by all map tasks=870781006
        Map-Reduce Framework
                CPU time spent (ms)=0
                Physical memory (bytes) snapshot=0
                Virtual memory (bytes) snapshot=0
2025-02-26 19:47:38,546 ERROR streaming.StreamJob: Job not successful!
Streaming Command Failed!
```

```
hyunsu_lee@cluster-134a-m:~$ yarn logs -applicationId application_1740584010684_0008 | grep -
i "ZeroDivisionError" | wc -l
2025-02-26 19:47:50,911 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to Resourc
eManager at cluster-134a-m.local./10.128.0.14:8032
2025-02-26 19:47:51,791 INFO client.AHSProxy: Connecting to Application History server at clu
ster-134a-m.local./10.128.0.14:10200
2025-02-26 19:47:52,687 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.prope
rties
2025-02-26 19:47:52,778 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 s
econd(s).
2025-02-26 19:47:52,779 INFO impl.MetricsSystemImpl: google-hadoop-file-system metrics system
 started
Feb 26, 2025 7:47:54 PM com.google.cloud.hadoop.fs.gcs.GhfsGlobalStorageStatistics trackDurat
ion
INFO: periodic connector metrics: {exception_count=2, gcs_api_client_non_found_response_count
=1, gcs_api_client_side_error_count=1, gcs_api_time=355, gcs_api_total_request_count=2, gcs_c
onnector_time=451, gcs_list_dir_request=1, gcs_list_dir_request_duration=178, gcs_list_dir_re
quest_max=178, gcs_list_dir_request_mean=178, gcs_list_dir_request_min=178, gcs_metadata_requ
est=1, gcs_metadata_request_duration=177, gcs_metadata_request_max=177, gcs_metadata_request_
mean=177, gcs_metadata_request_min=177, gs_filesystem_create=3, gs_filesystem_initialize=2, o
p_list_status=1, op_list_status_duration=451, op_list_status_max=451, op_list_status_mean=451
, op_list_status_min=451, uptimeSeconds=1} [CONTEXT ratelimit_period="5 MINUTES" ]
2025-02-26 19:47:54,554 INFO Configuration.deprecation: io.bytes.per.checksum is deprecated.
Instead, use dfs.bytes-per-checksum
24
```

```
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000002/./mapper_noll.py", line 14, in main
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000002/./mapper_noll.py", line 20, in <module>
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000003/./mapper_noll.py", line 14, in main
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000003/./mapper_noll.py", line 20, in <module>
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000004/./mapper_noll.py", line 14, in main
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000004/./mapper_noll.py", line 20, in <module>
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000005/./mapper_noll.py", line 14, in main
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000005/./mapper_noll.py", line 20, in <module>
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000006/./mapper_noll.py", line 14, in main
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000006/./mapper_noll.py", line 20, in <module>
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000007/./mapper_noll.py", line 14, in main
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000007/./mapper_noll.py", line 20, in <module>
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000008/./mapper_noll.py", line 14, in main
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000008/./mapper_noll.py", line 20, in <module>
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000009/./mapper_noll.py", line 14, in main
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000009/./mapper_noll.py", line 20, in <module>
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000010/./mapper_noll.py", line 14, in main
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000010/./mapper_noll.py", line 20, in <module>
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000011/./mapper_noll.py", line 14, in main
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000011/./mapper_noll.py", line 20, in <module>
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000012/./mapper_noll.py", line 14, in main
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000012/./mapper_noll.py", line 20, in <module>
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000013/./mapper_noll.py", line 14, in main
  File "/hadoop/yarn/nm-local-dir/usercache/hyunsu_lee/appcache/application_1740584010684_000
8/container_1740584010684_0008_01_000013/./mapper_noll.py", line 20, in <module>
```

```
Container: container_1740584010684_0008_01_000007 on cluster-134a-w-1.c.cs119-451916.internal_8026
Container: container_1740584010684_0008_01_000018 on cluster-134a-w-1.c.cs119-451916.internal_8026
Container: container_1740584010684_0008_01_000006 on cluster-134a-w-1.c.cs119-451916.internal_8026
Container: container_1740584010684_0008_01_000017 on cluster-134a-w-1.c.cs119-451916.internal_8026
```

[8 points] How many such messages did you find? Is the count you found consistent with what you might expect from random.randint(0,99)?

Yes, it's consistent. random.randint(0,99) has a 1/100 (1%) chance of generating 0, which would cause a divide-by-zero error.