Prachi Kotadia(A20549927)

CSP554—Big Data Technologies

## Assignment #3 (Modules 03a & 03b, 15 points)

**6. (3 points) Submit (1) a copy of this modified program and (2) a screenshot of the results of the program’s execution as the output of your assignment.**

**Code:**

# WordCount2.py

from mrjob.job import MRJob

import re

WORD\_RE = re.compile(r"[\w']+")

class MRWordCount(MRJob):

    def mapper(self, \_, line):

        for word in WORD\_RE.findall(line):

            if word[0 ] >= 'a' and word[0] <= 'n':

                yield 'a\_to\_n', 1

            else:

                yield 'other', 1

    def combiner(self, word, counts):

        yield word, sum(counts)

    def reducer(self, word, counts):

        yield word, sum(counts)

if \_\_name\_\_ == '\_\_main\_\_':

    MRWordCount.run()

**Query:**

****

**Output:**

A computer screen with white text

Description automatically generated

**8. (4 points) When you have accomplished this, please submit the following, (1) a copy of your MRJob code and (2) a copy of the output of the execution of that code.**

**Code:**

#WordCount3.py

from mrjob.job import MRJob

import re

WORD\_RE = re.compile(r"[\w']+")

class MRWordCount(MRJob):

    def mapper(self, \_, line):

        for word in WORD\_RE.findall(line):

            yield len(word), 1

    def combiner(self, word, counts):

        yield word, sum(counts)

    def reducer(self, word, counts):

        yield word, sum(counts)

if \_\_name\_\_ == '\_\_main\_\_':

    MRWordCount.run()

**Query**:



**Output:**

A black screen with white text

Description automatically generated

**10. (5 points) When you have accomplished this, please submit the following, (1) a copy of your MRJob code and (2) a copy of the output of the execution of that code for at least the first 10 bigram key value pairs.**

**Code:**

#WordCount4.py

from mrjob.job import MRJob

import re

WORD\_RE = re.compile(r"[\w']+")

class MRWordCount(MRJob):

    def mapper(self, \_, line):

        line = line.lower()

        words = WORD\_RE.findall(line)

        for i in range(len(words) - 1):

            yield words[i] + ' ' + words[i + 1], 1

    def combiner(self, word, counts):

        # Sum up the counts for each word pair

        yield word, sum(counts)

    def reducer(self, word, counts):

        # Sum up the counts for each word pair

        yield word, sum(counts)

if \_\_name\_\_ == '\_\_main\_\_':

    MRWordCount.run()

**Query:**



**Output:** A screen shot of a computer code

Description automatically generated

**14. (3 points) Submit (1) a copy of this modified program and (2) a screenshot of the results of the program’s execution as the output of your assignment.**

**Code:**

# Salaries2.py

from mrjob.job import MRJob

class MRSalaries2(MRJob):

    def mapper(self, \_, line):

        # Split the input line into fields

        fields = line.split('\t')

        # Extract the annual salary from the fields

        annual\_salary = float(fields[5])

        # Determine the salary group based on the annual salary

        if annual\_salary >= 100000.00:

            salary\_group = 'High'

        elif 50000.00 <= annual\_salary <= 99999.99:

            salary\_group = 'Medium'

        else:

            salary\_group = 'Low'

        # Emit the salary group as the key and a count of 1 as the value

        yield salary\_group, 1

    def combiner(self, salary\_group, counts):

        # Sum up the counts for each salary group

        yield salary\_group, sum(counts)

    def reducer(self, salary\_group, counts):

        # Sum up the counts for each salary group and yield the result

        yield salary\_group, sum(counts)

if \_\_name\_\_ == '\_\_main\_\_':

    MRSalaries2.run()

**Query:**

****

**Output:**

A computer screen with white text

Description automatically generated

**15. Remember to terminate your EMR cluster and remove your S3 bucket!**

A screenshot of a computer

Description automatically generated