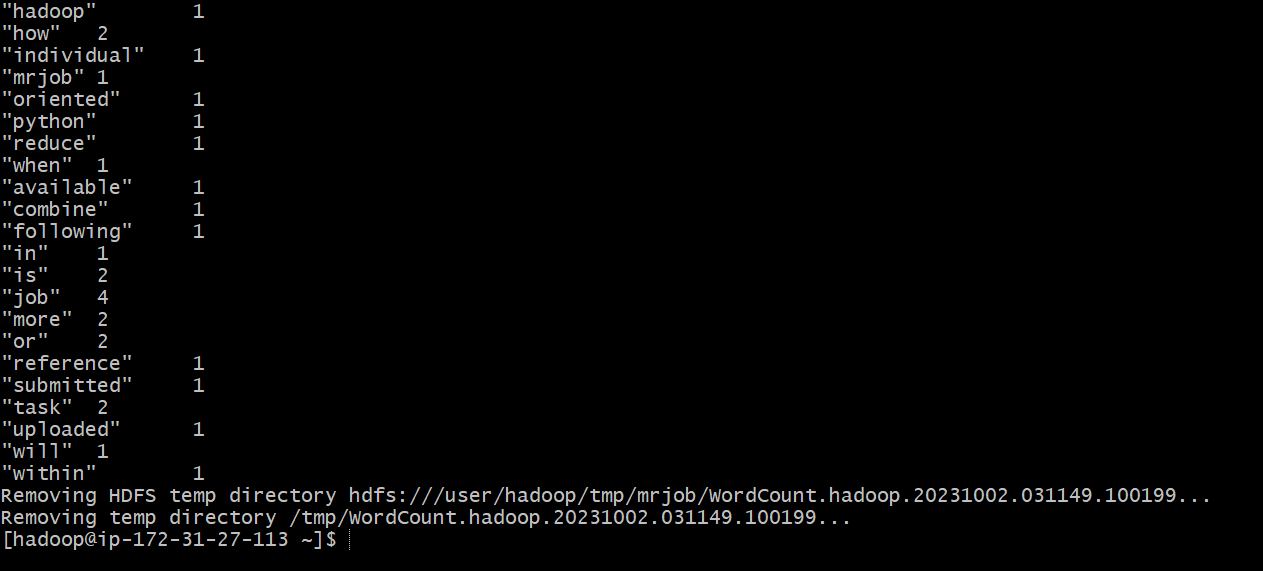
**BIG DATA TECHNOLOGES**

A20528281 **HOMEWORK\_3 Solutions** TRINADH RAYALA

5)WordCount.py : OUTPUT



7)WordCount2.py :

PROGRAM :

from mrjob.job import MRJob

import re

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

class WordGroupCounter(MRJob):

def mapper(self, \_, record):

for term in PATTERN.findall(record):

group\_key = "a\_to\_n" if 'a' <= term[0].lower() <= 'n' else "other"

yield group\_key, 1

def combiner(self, group\_key, totals):

yield group\_key, sum(totals)

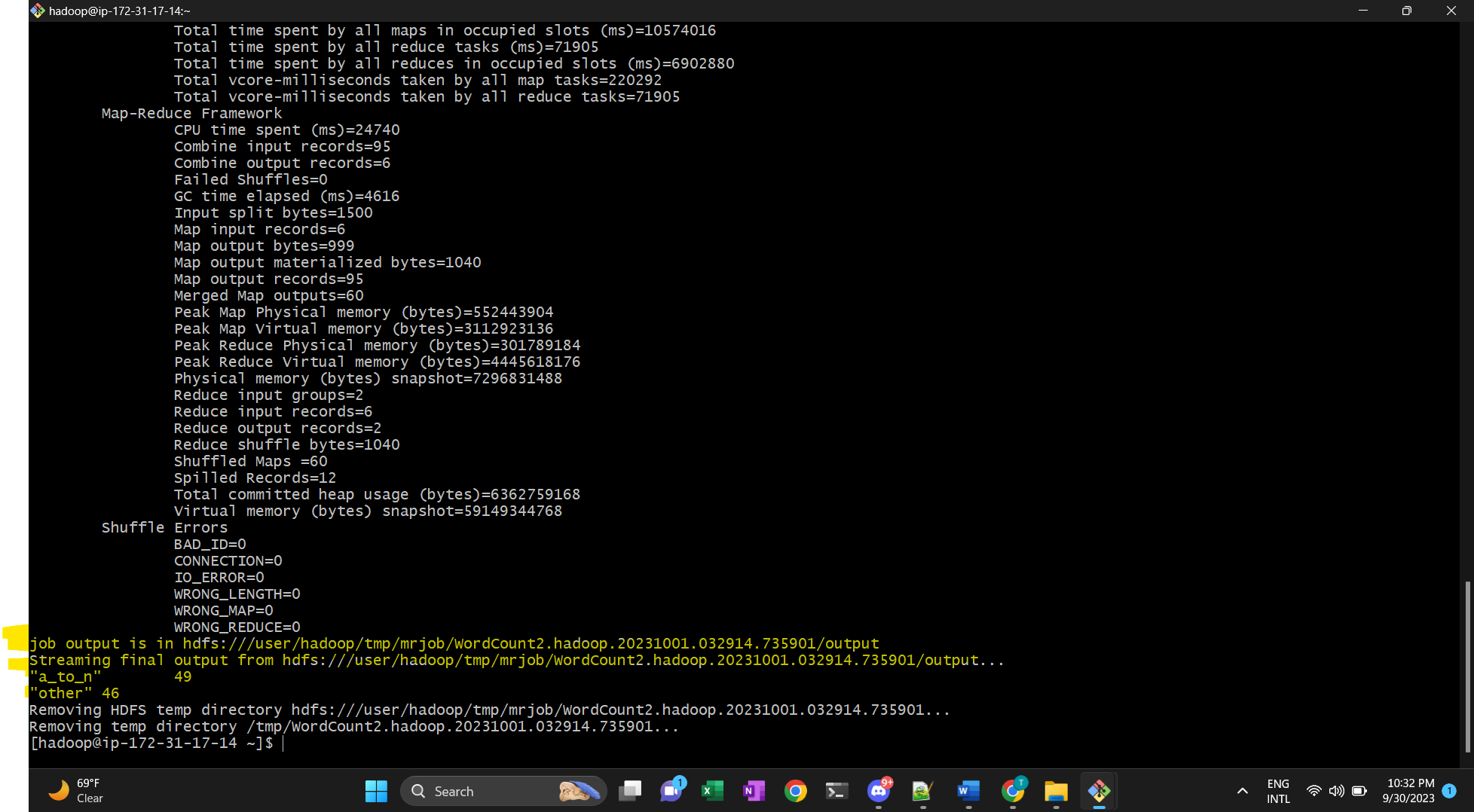
def reducer(self, group\_key, totals):

yield group\_key, sum(totals)

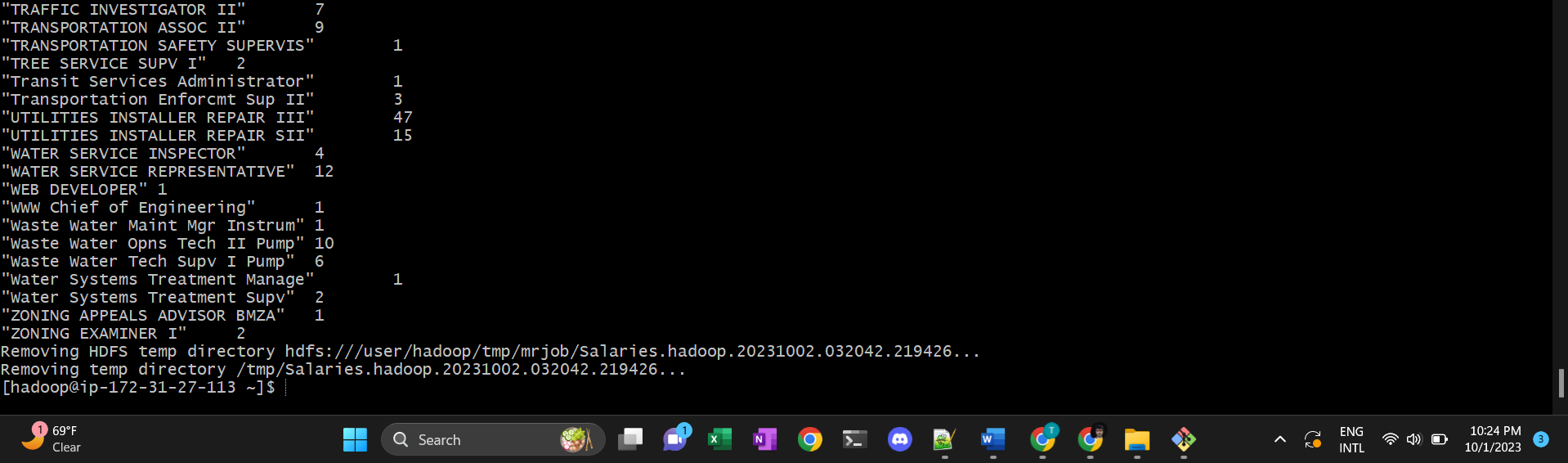
if \_\_name\_\_ == '\_\_main\_\_':

WordGroupCounter.run()

OUTPUT



9)Salaries.py : OUTPUT



**11)Salaries2.py :**

PROGRAM :

from mrjob.job import MRJob

class MRSalaryClassification(MRJob):

def mapper(self, \_, record\_line):

try:

(employee\_name, position\_title, org\_id, organization, join\_date, yearly\_pay, total\_pay) = record\_line.split('\t')

processed\_salary = float(yearly\_pay.replace(",", "").replace("$", ""))

if processed\_salary >= 100000:

yield "High", 1

elif 50000 <= processed\_salary < 100000:

yield "Medium", 1

else:

yield "Low", 1

except Exception as processing\_error:

print(f"Error processing record: {record\_line}. Error: {processing\_error}")

def combiner(self, salary\_category, category\_counts):

yield salary\_category, sum(category\_counts)

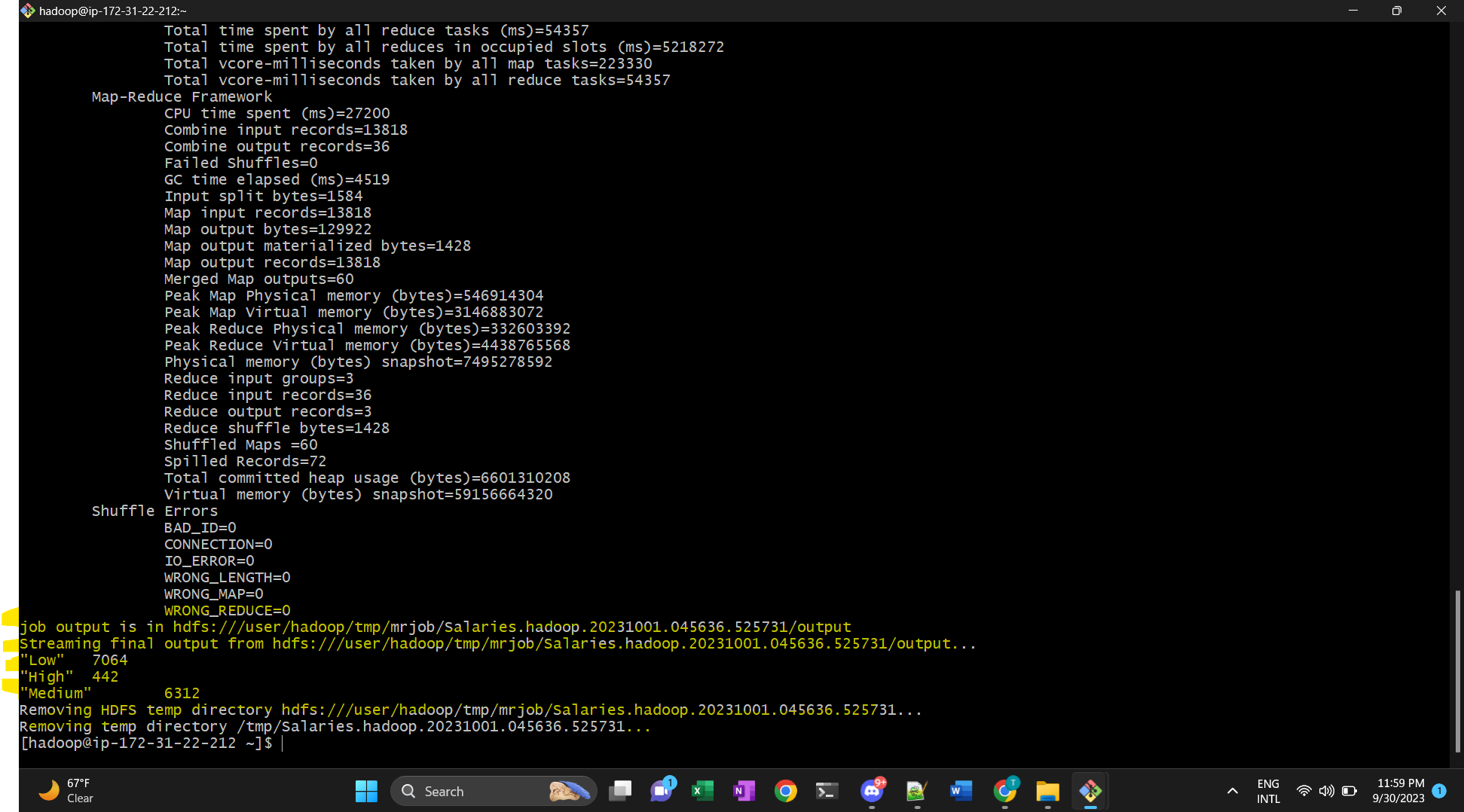
def reducer(self, salary\_category, category\_counts):

yield salary\_category, sum(category\_counts)

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

MRSalaryClassification.run()

OUTPUT :



13) program to perform the task of outputting a count of the number of movies each user (identified via their user id) reviewed.

PROGRAM :

from mrjob.job import MRJob

from mrjob.step import MRStep

class MovieReview(MRJob):

def steps(self):

return [MRStep(mapper=self.distribute\_data, reducer=self.collate\_data)]

def distribute\_data(self, \_, entry):

user\_id, film\_id, \_, \_ = entry.split(',')

yield user\_id, 1

def collate\_data(self, user\_id, occurrences):

yield user\_id, sum(occurrences)

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

MovieReview(args=[data\_source]).run()

OUTPUT :

