5)WordCount.py : OUTPUT

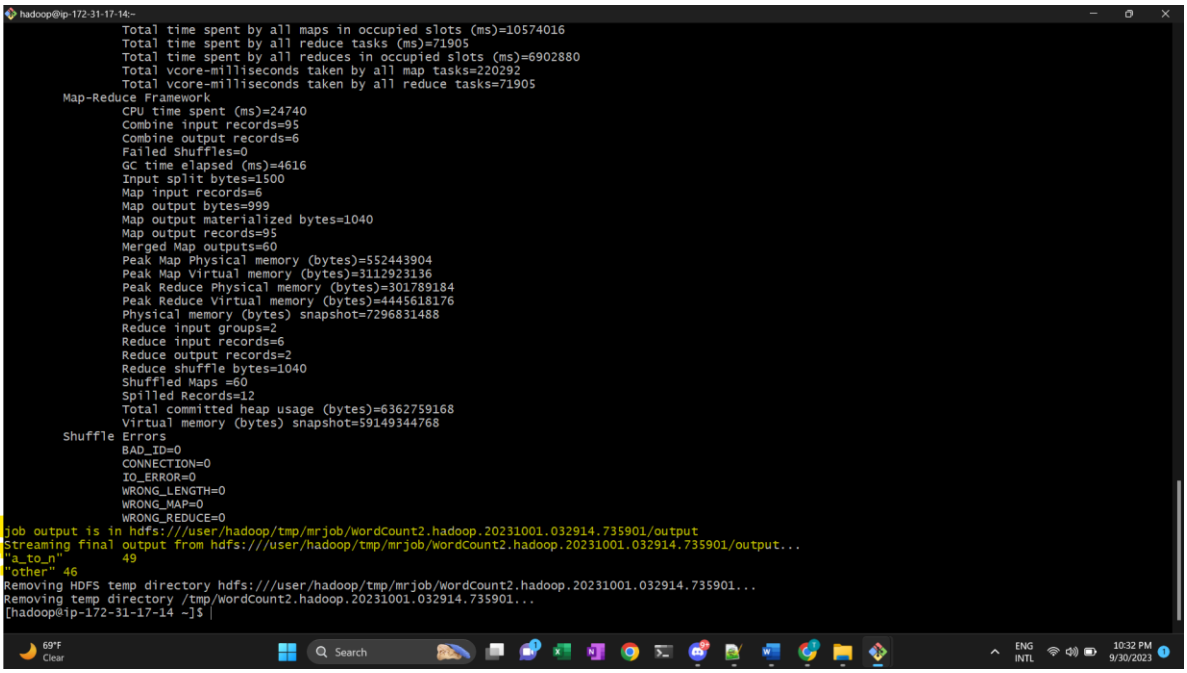
```
"hadoop"        1
"how"   2
"individual"    1
"mrjob" 1
"oriented"      1
"python"        1
"reduce"        1
"when"  1
"available"     1
"combine"       1
"following"     1
"in"    1
"is"    2
"job"   4
"more"  2
"or"    2
"reference"     1
"submitted"     1
"task"  2
"uploaded"      1
"will"  1
"within"        1
Removing HDFS temp directory hdfs:///user/hadoop/tmp/mrjob/WordCount.hadoop.20231002.031149.100199...
Removing temp directory /tmp/WordCount.hadoop.20231002.031149.100199...
[hadoop@ip-172-31-27-113 ~]$
```

7)WordCount2.py :

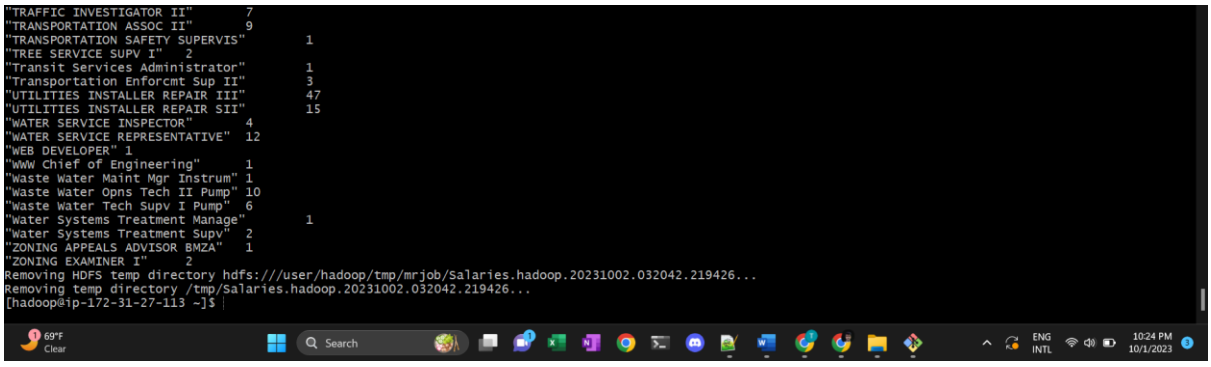
PROGRAM :

```
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



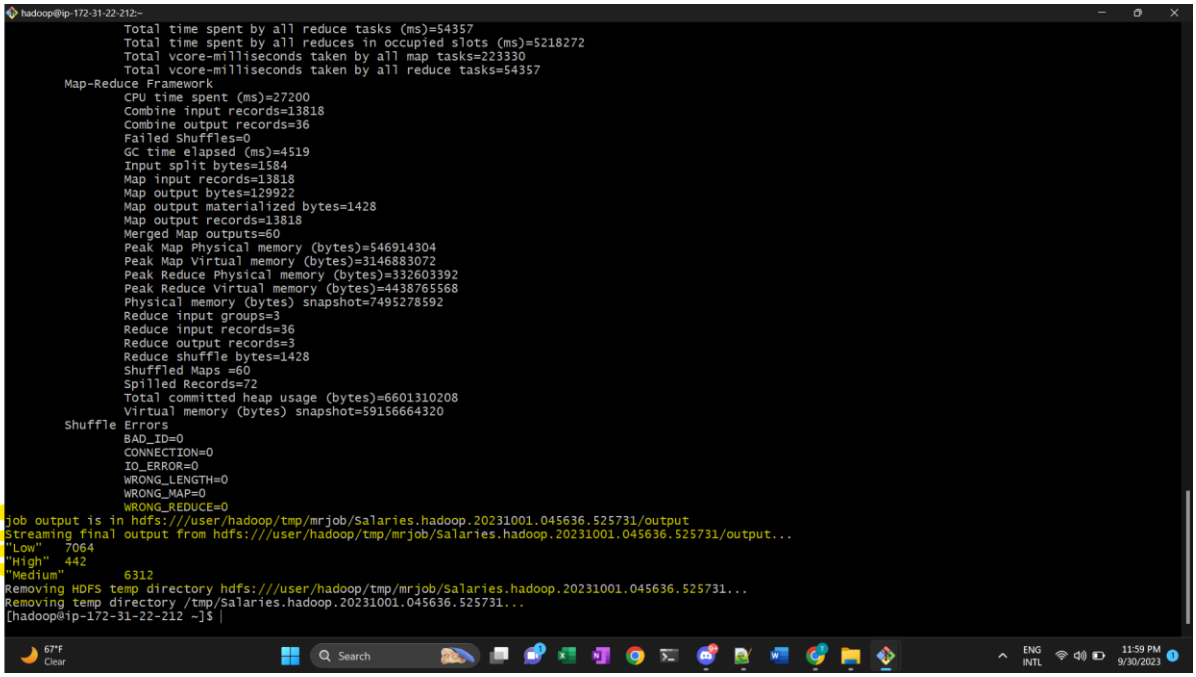
11)Salaries2.py :

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 :