Experiment 5

Name: Asad Shaikh
Branch: MTech CE
Registration Id: 242050023

Aim: Apply MapReduce Algorithms to Perform Analytics on a Single Node Cluster
a) Phrase Frequency Analysis
b) Search Records with Matching Criteria
c) Aggregation of Inputs and Search Records Based on Aggregated Output

Theory:

Step 2: Task A - Phrase Frequency Analysis

Objective: Analyze the frequency of phrases in the given dataset.

Mapper:

- The **Mapper** will read the text input (could be a large dataset like logs or articles), split it into words, and output each word with a count of 1.

```
# Mapper for Phrase Frequency Analysis
class Mapper:
    def map(self, _, line):
        # Tokenize the line into words
        words = line.split()
        for word in words:
            # Emit each word and a count of 1
            print(f"{word}\t1")
```

Reducer:

- The **Reducer** will aggregate the word counts for each word and output the total count of each phrase (or word).

```
# Reducer for Phrase Frequency Analysis
class Reducer:
    def reduce(self, word, counts):
        # Sum up all counts for the word
        total_count = sum(counts)
        # Emit the word and its total frequency
        print(f"{word}\t{total_count}")
```

Execution:

1. Run the MapReduce job using the Hadoop command line:

```
hadoop jar /path/to/hadoop-streaming.jar \
-input /input/dataset.txt \
-output /output/frequency_result \
-mapper Mapper.py \
-reducer Reducer.py
```

Step 3: Task B - Search Records with Matching Criteria

Objective: Search for records in the dataset that meet specific criteria (e.g., records containing a particular keyword).

Mapper:

- The **Mapper** will check if a record matches the given criteria (e.g., contains a keyword).

```
# Mapper for Search Records
class Mapper:
    def map(self, _, record):
        # Check if the record matches the search keyword
        if "keyword" in record:
            print(f"{record}\t1")  # Emit matched record
```

Reducer:

- The **Reducer** will receive all the records that matched the criteria and output the results.

```
# Reducer for Search Records
class Reducer:
    def reduce(self, record, values):
        # Simply output the record if it matched the keyword
        print(f"{record}")
```

Execution:

1. Run the MapReduce job to search for matching records:

```
hadoop jar /path/to/hadoop-streaming.jar \
-input /input/dataset.txt \
-output /output/search_result \
-mapper Mapper.py \
-reducer Reducer.py
```

Step 4: Task C - Aggregation of Inputs and Search Records Based on Aggregation Output

Objective: Aggregate numeric data (e.g., sum) based on certain keys and filter the results based on the aggregated values.

Mapper:

- The **Mapper** will emit key-value pairs based on the input data for aggregation. For example, for each category, it will emit the value for aggregation.

```
# Mapper for Aggregation and Search Based on Output
class Mapper:
    def map(self, _, record):
        # Example: record might be in format "category,value"
        category, value = record.split(',')
        print(f"{category}\t{value}")
```

Reducer:

- The **Reducer** will aggregate the values for each category (e.g., sum the values) and then filter records based on the result (e.g., sum greater than a threshold).

```
# Reducer for Aggregation and Filter
class Reducer:
    def reduce(self, category, values):
        # Aggregate values (e.g., sum)
        total_sum = sum([int(value) for value in values])

        # Filter based on aggregation (e.g., sum > threshold)
        if total_sum > 100:
            print(f"{category}\t{total_sum}")
```

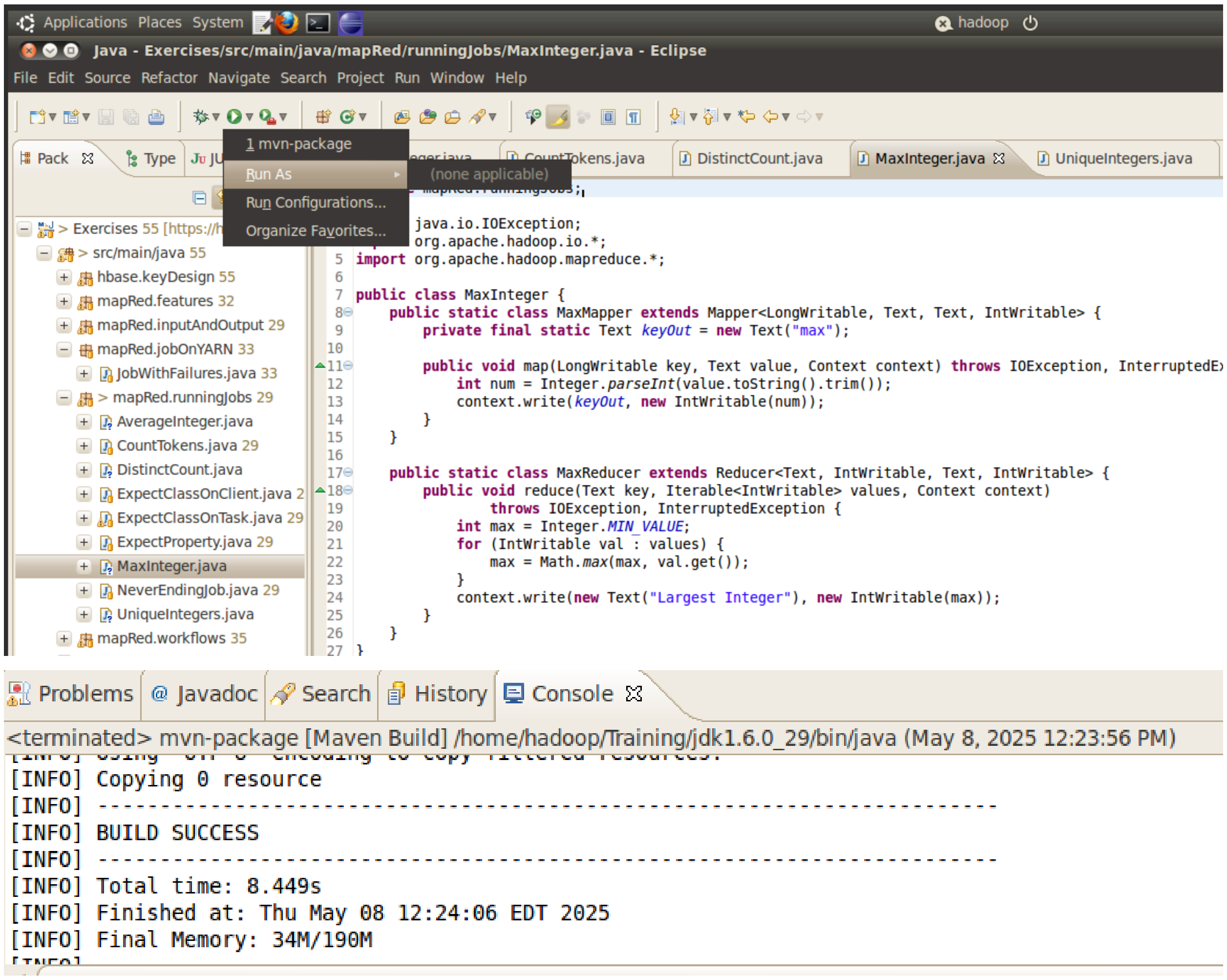
Execution:

1. Run the MapReduce job to aggregate and filter the records based on the sum:

```
hadoop jar /path/to/hadoop-streaming.jar \
-input /input/dataset.txt \
-output /output/aggregation_result \
-mapper Mapper.py \
-reducer Reducer.py
```

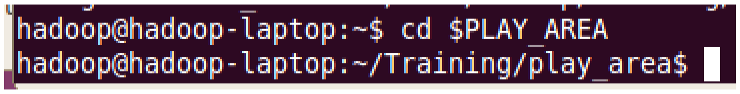
Code / Output:

Run as mvn-package:



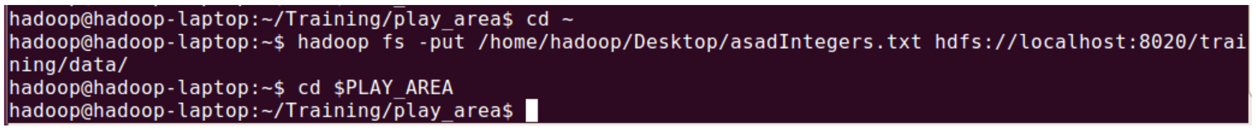
```
                    java.io.IOException;
                    org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;

public class MaxInteger {
    public static class MaxMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
        private final static Text keyOut = new Text("max");

        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedE
            int num = Integer.parseInt(value.toString().trim());
            context.write(keyOut, new IntWritable(num));
        }
    }

    public static class MaxReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
        public void reduce(Text key, Iterable<IntWritable> values, Context context)
                throws IOException, InterruptedException {
            int max = Integer.MIN_VALUE;
            for (IntWritable val : values) {
                max = Math.max(max, val.get());
            }
            context.write(new Text("Largest Integer"), new IntWritable(max));
        }
    }
}
```

```
<terminated> mvn-package [Maven Build] /home/hadoop/Training/jdk1.6.0_29/bin/java (May 8, 2025 12:23:56 PM)
[INFO] Copying 0 resource
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 8.449s
[INFO] Finished at: Thu May 08 12:24:06 EDT 2025
[INFO] Final Memory: 34M/190M
```



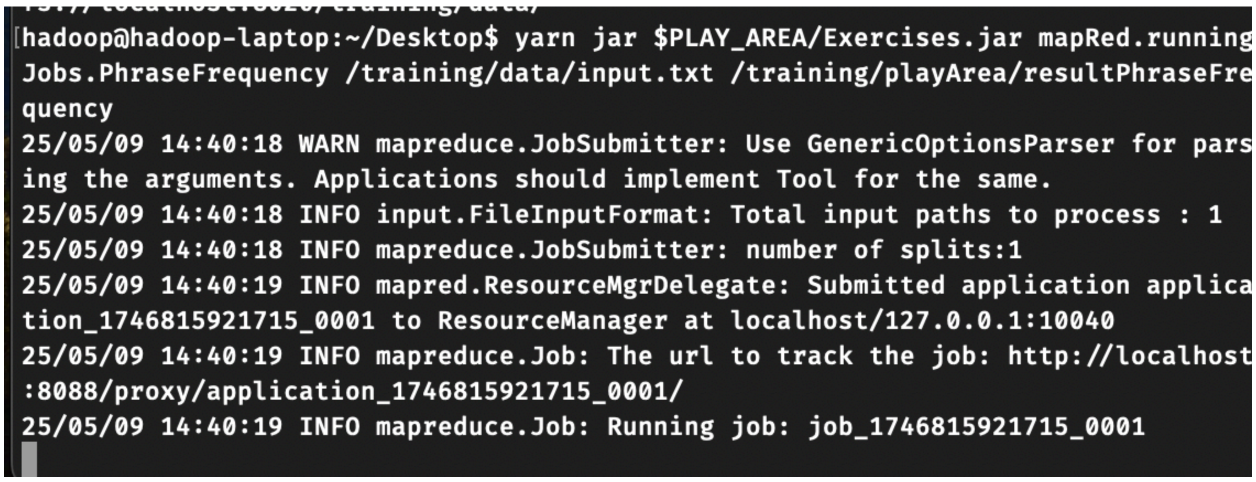
```
hadoop@hadoop-laptop:~$ cd $PLAY_AREA
hadoop@hadoop-laptop:~/Training/play_area$
```

Putting Input file from Desktop to Hdfs:



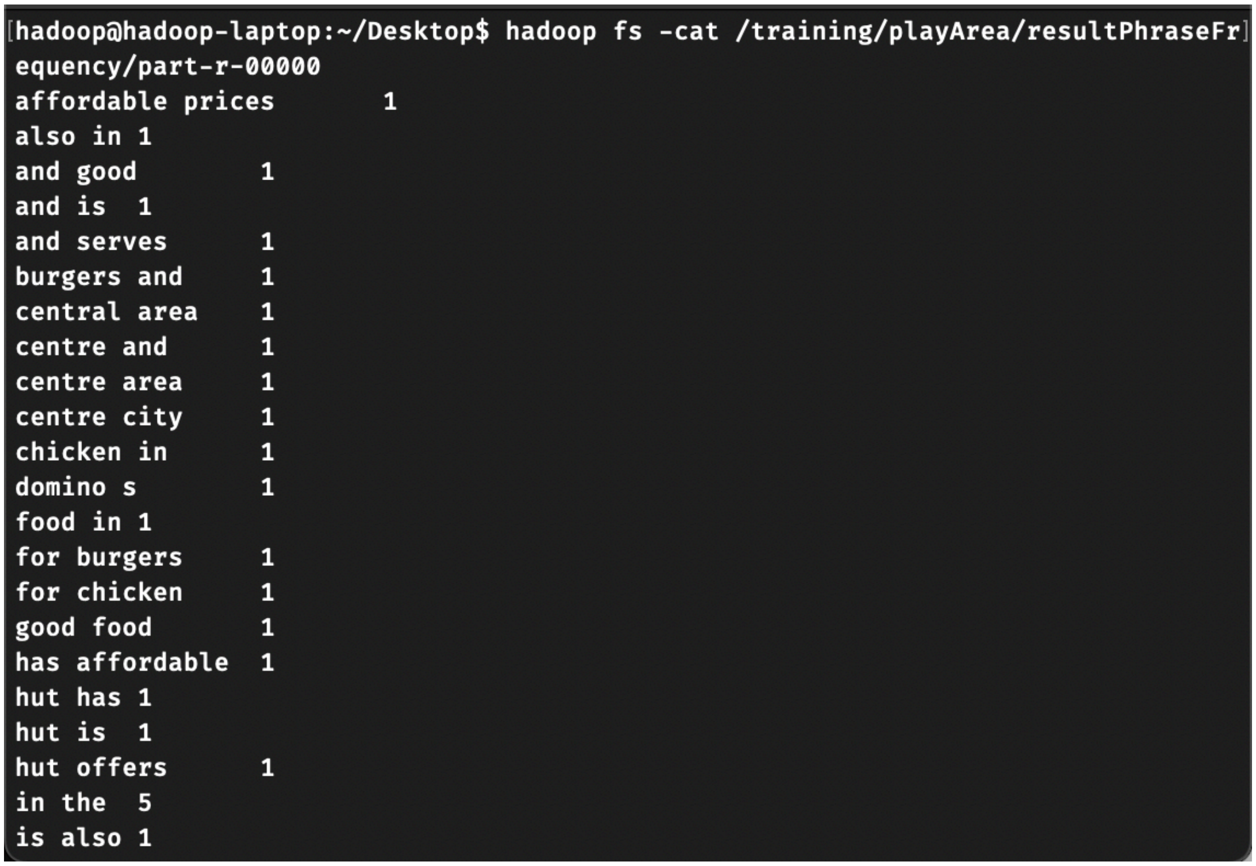
```
hadoop@hadoop-laptop:~/Training/play_area$ cd ~
hadoop@hadoop-laptop:~$ hadoop fs -put /home/hadoop/Desktop/asadIntegers.txt hdfs://localhost:8020/trai
ning/data/
hadoop@hadoop-laptop:~$ cd $PLAY_AREA
hadoop@hadoop-laptop:~/Training/play_area$
```

Executing PhraseFrequency.java:



```
[hadoop@hadoop-laptop:~/Desktop$ yarn jar $PLAY_AREA/Exercises.jar mapRed.running
Jobs.PhraseFrequency /training/data/input.txt /training/playArea/resultPhraseFre
quency
25/05/09 14:40:18 WARN mapreduce.JobSubmitter: Use GenericOptionsParser for pars
ing the arguments. Applications should implement Tool for the same.
25/05/09 14:40:18 INFO input.FileInputFormat: Total input paths to process : 1
25/05/09 14:40:18 INFO mapreduce.JobSubmitter: number of splits:1
25/05/09 14:40:19 INFO mapred.ResourceMgrDelegate: Submitted application applica
tion_1746815921715_0001 to ResourceManager at localhost/127.0.0.1:10040
25/05/09 14:40:19 INFO mapreduce.Job: The url to track the job: http://localhost
:8088/proxy/application_1746815921715_0001/
25/05/09 14:40:19 INFO mapreduce.Job: Running job: job_1746815921715_0001
```

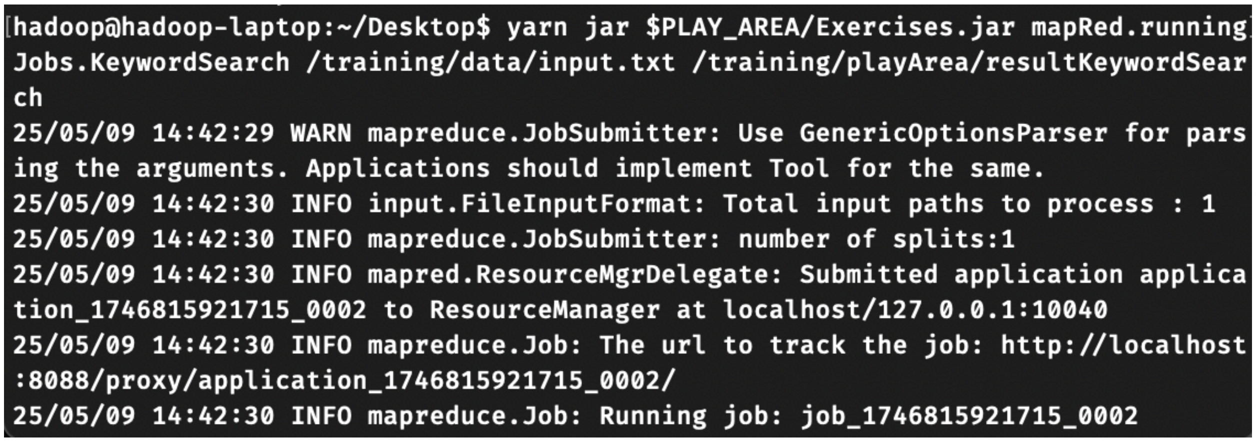
Reading the output:



```
[hadoop@hadoop-laptop:~/Desktop$ hadoop fs -cat /training/playArea/resultPhraseFr]
equency/part-r-00000
affordable prices       1
also in 1
and good        1
and is  1
and serves      1
burgers and     1
central area    1
centre and      1
centre area     1
centre city     1
chicken in      1
domino s        1
food in 1
for burgers     1
for chicken     1
good food       1
has affordable  1
hut has 1
hut is  1
hut offers      1
in the  5
is also 1
```

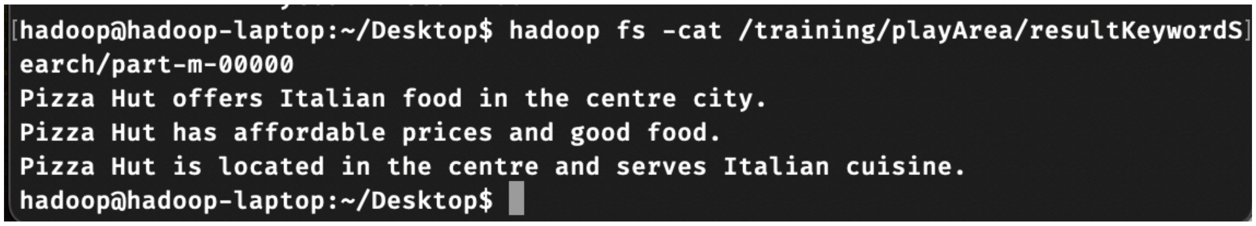
Output: Gives us the count of bigram phrases present in the input.txt file

Executing KeywordSearch.java:



```
hadoop@hadoop-laptop:~/Desktop$ yarn jar $PLAY_AREA/Exercises.jar mapRed.running
Jobs.KeywordSearch /training/data/input.txt /training/playArea/resultKeywordSear
ch
25/05/09 14:42:29 WARN mapreduce.JobSubmitter: Use GenericOptionsParser for pars
ing the arguments. Applications should implement Tool for the same.
25/05/09 14:42:30 INFO input.FileInputFormat: Total input paths to process : 1
25/05/09 14:42:30 INFO mapreduce.JobSubmitter: number of splits:1
25/05/09 14:42:30 INFO mapred.ResourceMgrDelegate: Submitted application applica
tion_1746815921715_0002 to ResourceManager at localhost/127.0.0.1:10040
25/05/09 14:42:30 INFO mapreduce.Job: The url to track the job: http://localhost
:8088/proxy/application_1746815921715_0002/
25/05/09 14:42:30 INFO mapreduce.Job: Running job: job_1746815921715_0002
```

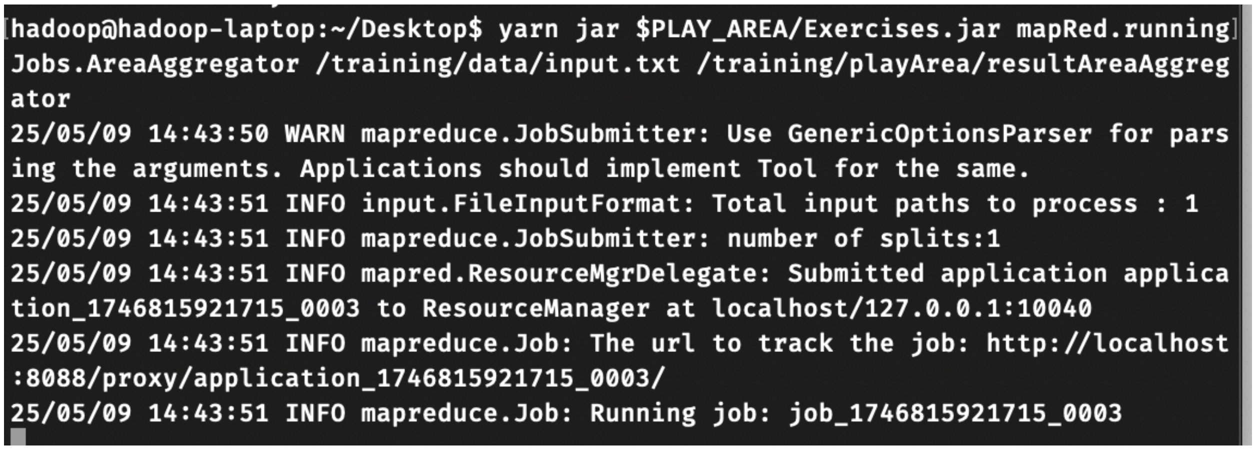
Reading the output:



```
hadoop@hadoop-laptop:~/Desktop$ hadoop fs -cat /training/playArea/resultKeywordS
earch/part-m-00000
Pizza Hut offers Italian food in the centre city.
Pizza Hut has affordable prices and good food.
Pizza Hut is located in the centre and serves Italian cuisine.
hadoop@hadoop-laptop:~/Desktop$
```

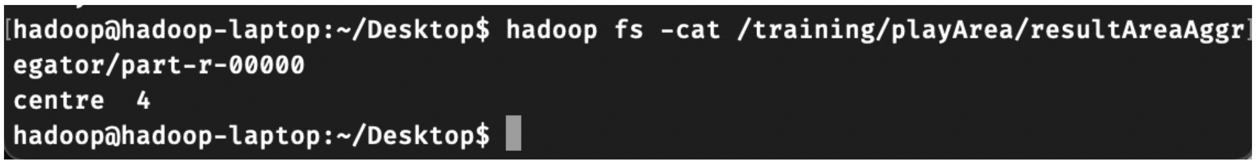
Output: 69.2 → Lists all the lines containing the keyword "Pizza Hut".

Executing AreaAggregator.java:



```
hadoop@hadoop-laptop:~/Desktop$ yarn jar $PLAY_AREA/Exercises.jar mapRed.running
Jobs.AreaAggregator /training/data/input.txt /training/playArea/resultAreaAggreg
ator
25/05/09 14:43:50 WARN mapreduce.JobSubmitter: Use GenericOptionsParser for pars
ing the arguments. Applications should implement Tool for the same.
25/05/09 14:43:51 INFO input.FileInputFormat: Total input paths to process : 1
25/05/09 14:43:51 INFO mapreduce.JobSubmitter: number of splits:1
25/05/09 14:43:51 INFO mapred.ResourceMgrDelegate: Submitted application applica
tion_1746815921715_0003 to ResourceManager at localhost/127.0.0.1:10040
25/05/09 14:43:51 INFO mapreduce.Job: The url to track the job: http://localhost
:8088/proxy/application_1746815921715_0003/
25/05/09 14:43:51 INFO mapreduce.Job: Running job: job_1746815921715_0003
```

Reading the output:



```
hadoop@hadoop-laptop:~/Desktop$ hadoop fs -cat /training/playArea/resultAreaAggr
egator/part-r-00000
centre  4
hadoop@hadoop-laptop:~/Desktop$
```

Output: 4 → There are 4 'centre' areas in the Input.txt file.

Conclusion:
In this experiment, we learned about Hadoop and how it is used to analyze big data. We also learned about the MapReduce programming model which is used to process large amounts of data in parallel. We also implemented four programs, ‘PhraseFrequency’, ‘KeywordSearch’ and ‘AreaAggregator’ in MapReduce.