**CSE:587 Assignment 2: Big Data Processing with Hadoop**

**Ankita Das (50317491)**

Abstract

This project depicts the development and simulation of Big data processing with Hadoop. Here we have worked with basic text processing by calculating word count, N-grams, and Inverted index for 3 given text files. Also, we have performed join operation for two given dataset and implement KNN algorithm where train and test dataset were given. All these tasks have been performed in Hadoop environment and python has been used as programming language. The steps to perform these tasks and results have been explained in detail throughout this paper.

**PART1 - Setup and WordCount - 5 Points**

● Get familiar with the VM and the Hadoop framework

● In the folder gutenberg, located in your home directory, there are 3 documents. Use

necessary commands to transfer data to Hadoop distributed file system

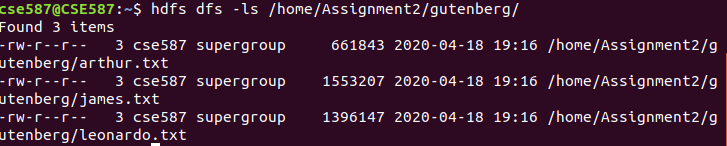
● Implement a MapReduce algorithm to produce count of every word in the document

**Solution:**

●We have been given 3 text files under the folder Gutenberg, located in our home directory. These 3 documents were transferred to Hadoop distributed file system using the below command:



We checked if the files have been transferred correctly to Hadoop by using following steps:



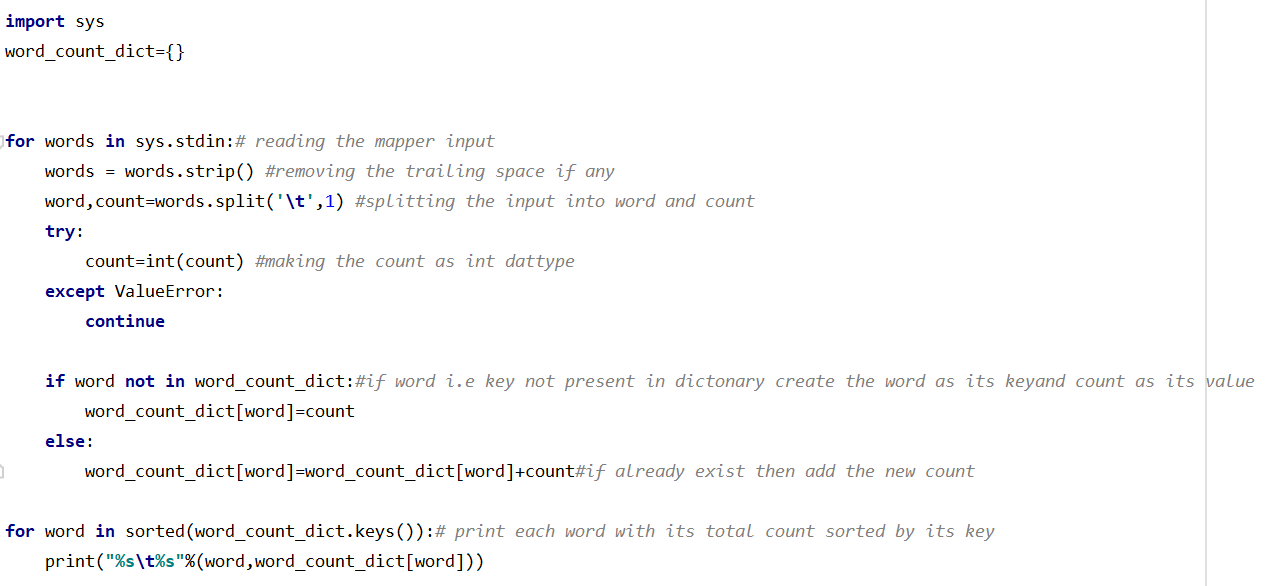
● Now, we have to implement MapReduce algorithm to produce count of every word in the document.

Mapper:



* We have to first read the input file from the Hadoop file system using sys library of Python
* Here sys.stdin will read the file line by line
* Finally, the line is stripped to remove trailing spaces and split word by word where each word has to be preprocessed to make it lowercase whether it is stopword or contains and punctuation as stated above in the code snippet
* We are removing all the stopword using stopword library of Python’s nltk.corpus and punctuation by using regular expression
* Finally, each word is printed with the delimiter as \t along with its count 1.

Reducer:



* Reducer takes the mapper printed input line by line using sys.stdin and separate it with delimiter ‘\t’ and finally assigning the word to variable and giving value 1 to count variable
* We have to check for count to be only int datatype so we do Typecasting
* Finally, we make a dictionary having key value pair where key is the word and value is its count and if the word comes again while iterating we just increment the value of that word (key) by 1
* We sort the dictionary by keys and print the key with “\t” as the delimiter along with its corresponding count.

Result: *Double- Click the output1.txt below*



**PART 2 - N-grams- 10 Points**

**●** Using the same gutenberg dataset, implement a MapReduce algorithm that will produce

modified tri-grams around the key words, after replacing the key word with ‘$’.

Example:

cat was sitting on a roof ---> if the key word was ‘sitting’ ---> the modified tri-grams would

be

cat\_was\_$, was\_$\_on,$\_on\_a,

● The key words to look for in the gutenberg dataset are ‘science’, ‘sea’ , ‘fire’.

● The algorithm after producing these modified tri-grams, should return the 10 most

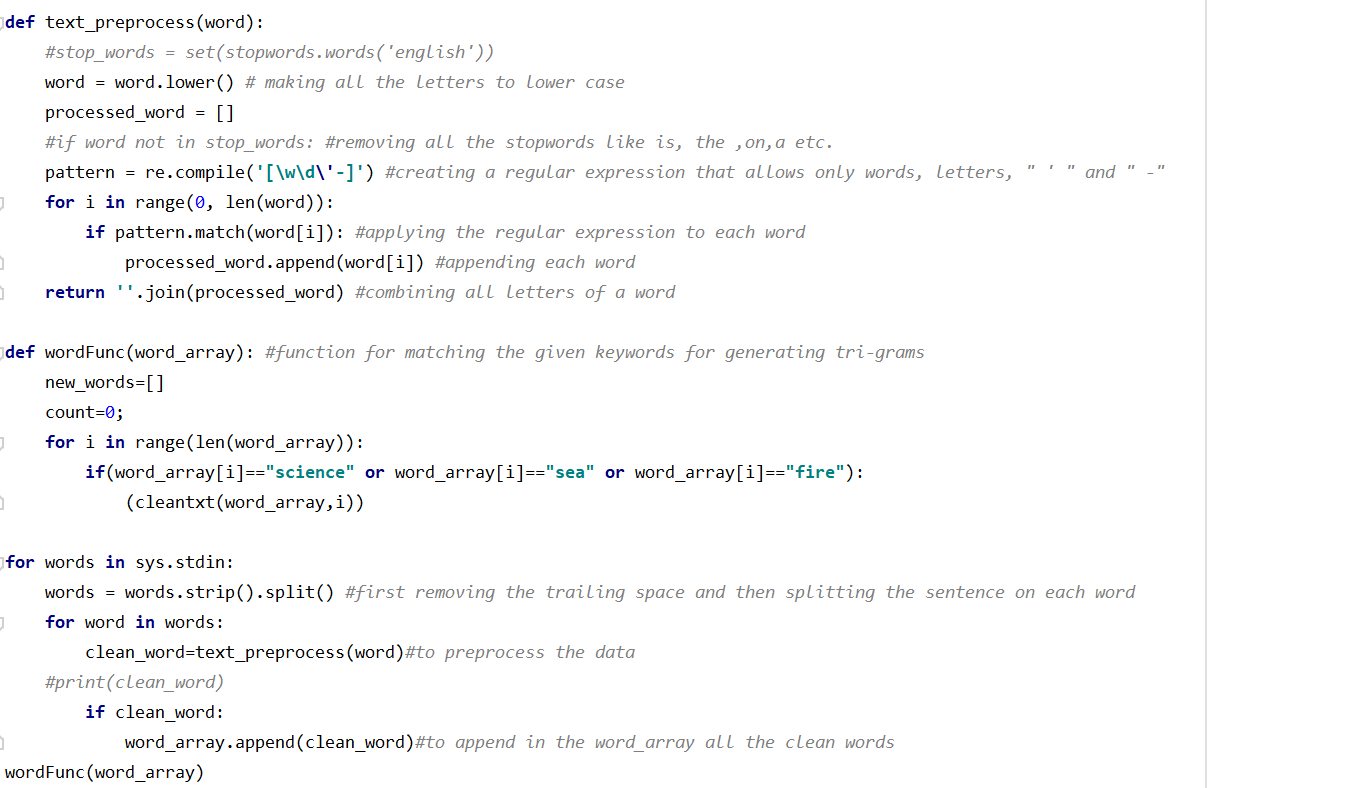
occurred modified tri-gram in the dataset.

**Solution:**

We need to implement a MapReduce algorithm that will produce modified tri-grams around the key words, after replacing the key word with ‘$’.

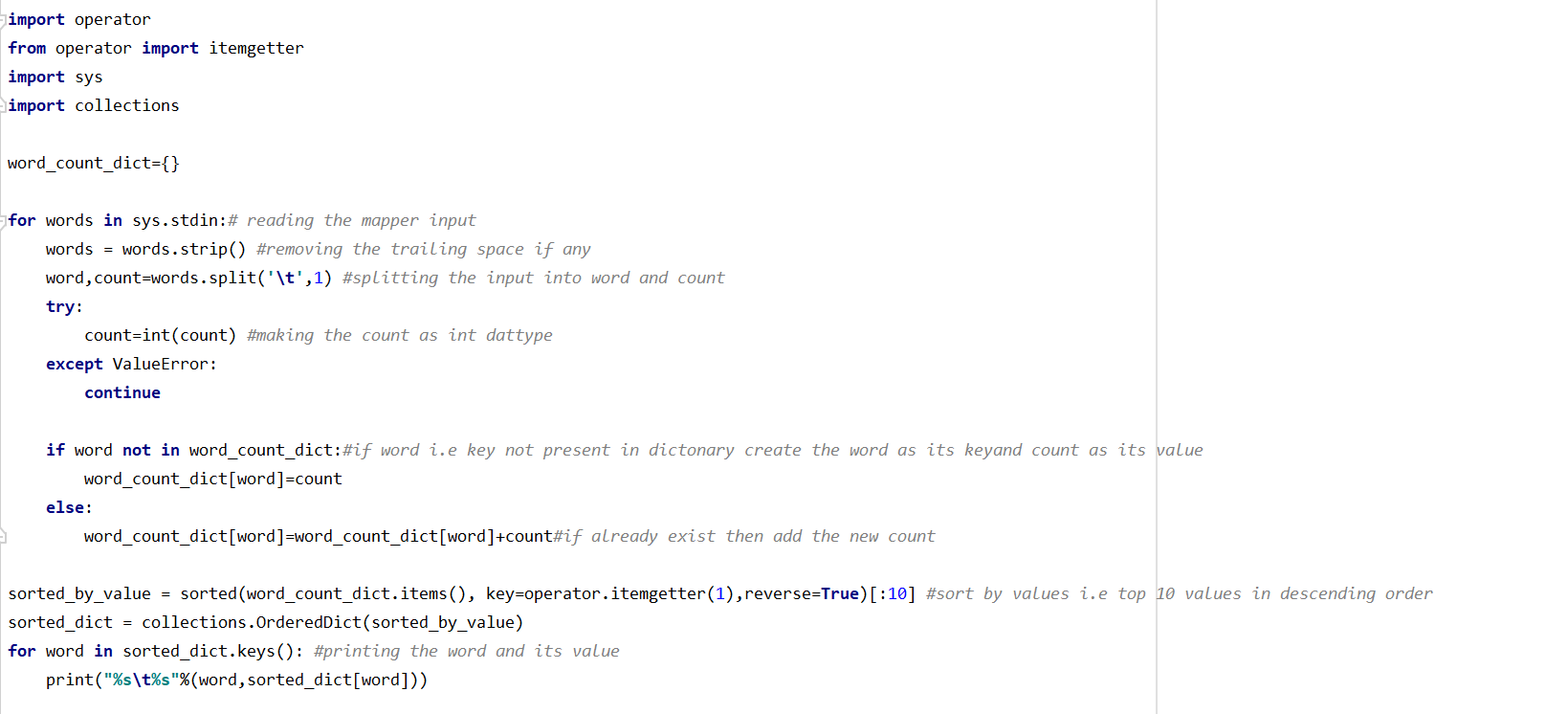
Mapper:





* Here we are reading each line of each file using sys.stdin stripping and splitting it for each word is processed for lower case punctuation just like the part1 of this assignment
* Now this each word is passed to wordFunc() function where it will be checked with the 3 keywords “science” “sea” and “fire”
* And if it matches its index along with the entire word\_array will be given to our clean\_text() function where the trigrams will be made surrounding the index of the keyword and replacing it with the $
* Finally, the new trigram made and its count as 1 is printed which will be input to reducer
* **Note**: we are not removing stopwords as to generate more meaningful output.

Reducer:



* The reducer file will take the mapper output as its input and will use the dictonary to map all the trigram with its corresponding counts.
* Only thing here we will be sorting the key value pair by value and will return the top 10 trigram.

Result: *Double- Click the output2.txt below*



**PART 3 - Inverted Index - 5 Points**

● Using the gutenberg dataset, implement a MapReduce algorithm to produce inverted

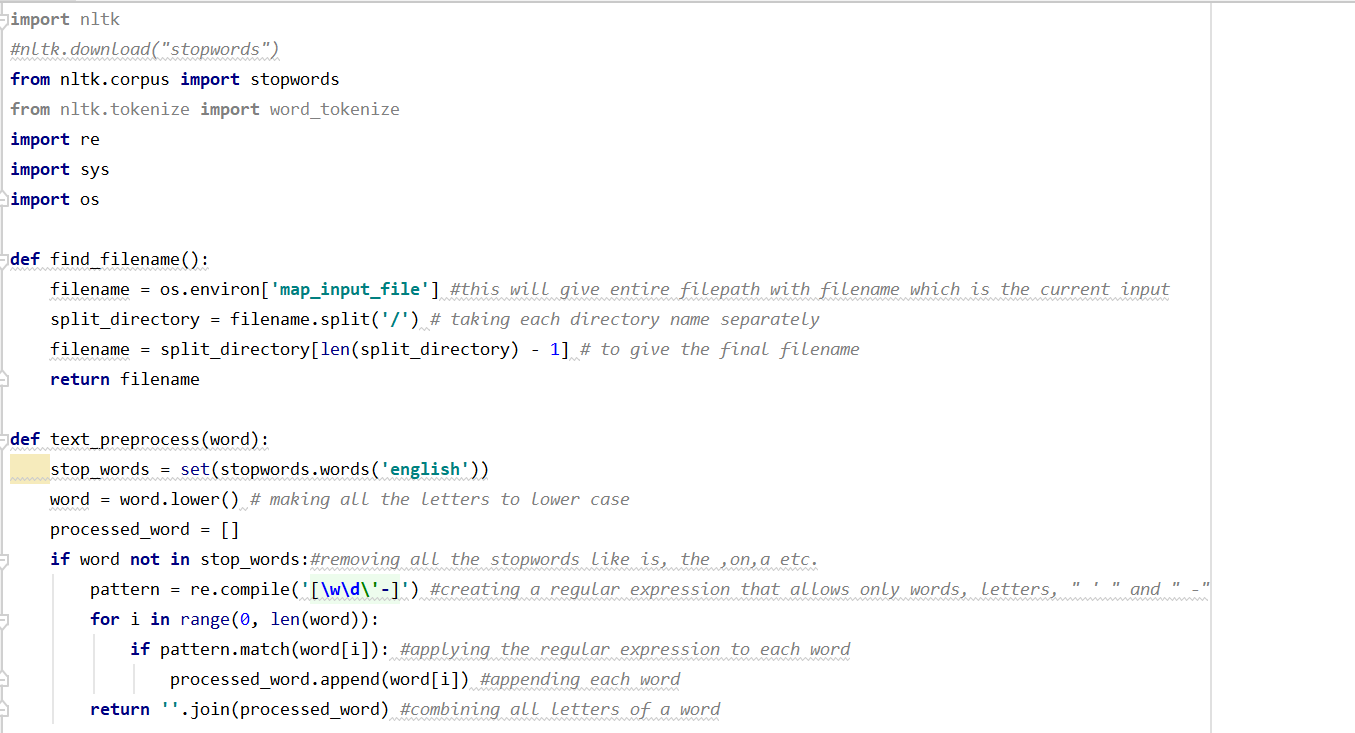
index for the whole dataset.

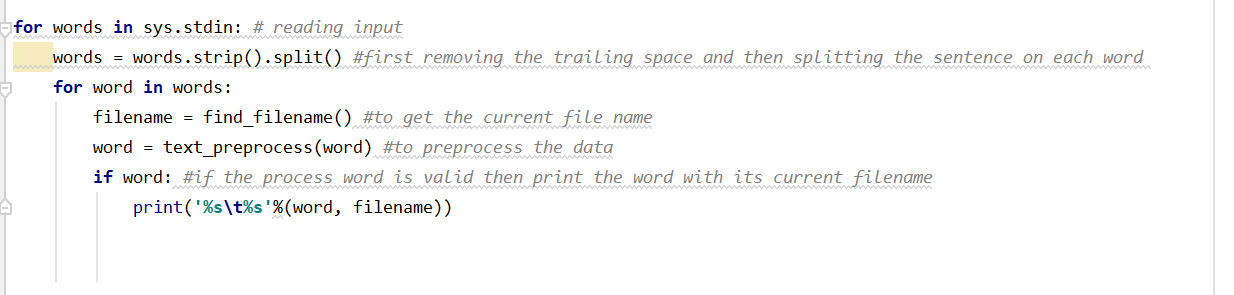
● A small explanation of what inverted index is can be found in the link Inverted index

**Solution:**

We need to implement a MapReduce algorithm to produce inverted index for the whole dataset.

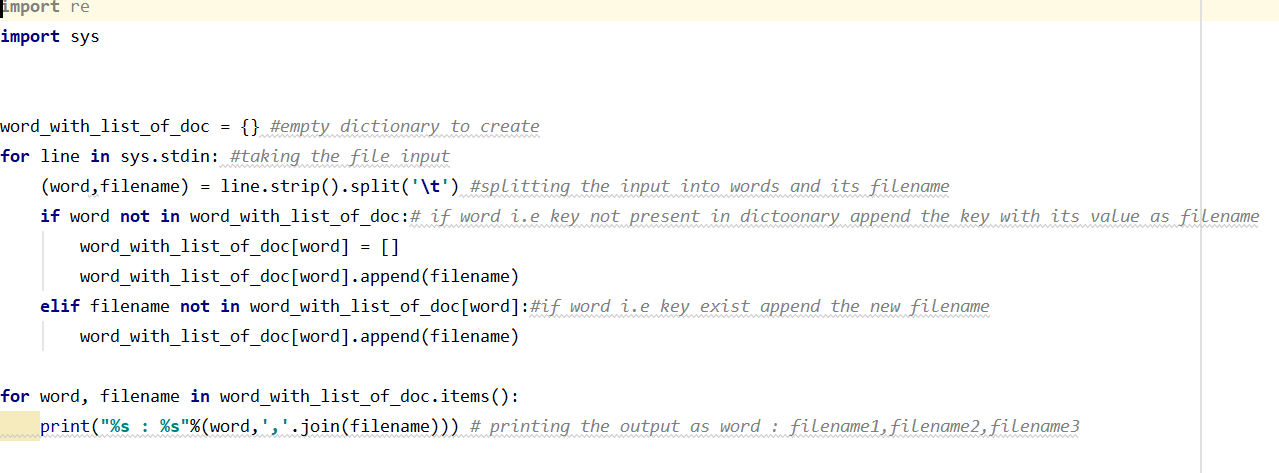
Mapper:





* This mapper too uses the sys.stdin file to read each line of all the files.
* However here we are using os.environ() which will return the path of the current word and where will take the last index value from the given path which is ultimately the file in which the word was present.
* Finally, we will be printing the word and its corresponding file name which will be input to the reducer

Reducer:



* Here the input will be spitted for the words and its corresponding filename
* The dictonary word\_with\_list\_of\_doc is used to maintain the key as word and value as list of its corresponding filename in which it is present.
* We will be using .join() to join each value of the filename list by “,”
* Finally, we print the word separated by “ : ” with its corresponding file name.

Result: *Double- Click the output3.txt below*



**PART 4 - Relational Join - 5 Points**

● Using the Dataset provided along with the assignment, Implement a MapReduce

algorithm to join two datasets using a primary key

● The assumed primary key is the ‘Employee ID’

**Solution:**

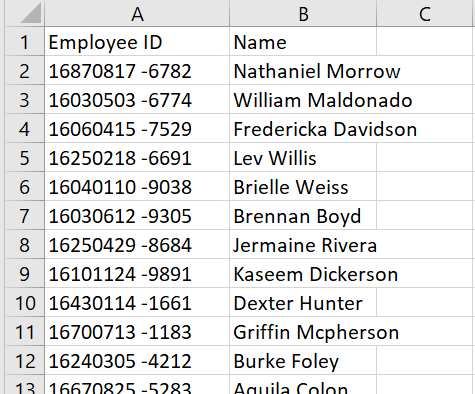
Here we need to perform join operation between two given datasets. Dataset 1 (join1) is having details about Employee ID and Employee Name, whereas Dataset 2 (join2) is having details about Employee Name, Salary, Country and Passcode. We need to perform the join by assuming Employee ID as the key.

The mapper will read join1 and join2 dataset from command line and on the basis on the column numbers it can differentiate between two files. Initially all of the fields will be initialized to “-1”.

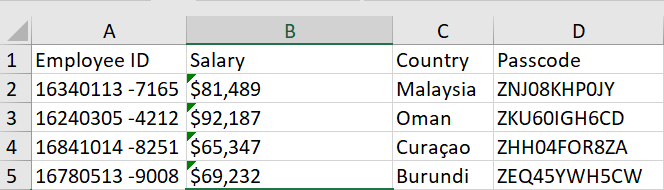
When mapper will not get any value for a particular column, for example “Name” column for join2 dataset, it will set it to 1. The output of the mapper will be the collection of the data from join1 and join2, and will be fed to the reducer.

Now on the basis of the “-1”, it reducer can identify which data is coming for join 1 and which is from join2. As, for “Name” field it will be always “-1” for join2 dataset. Now we have taken two dictionary to store the data by making “Employee ID” as key. Then on the basis of the key we joined the data and got the desired output.

Dataset 1 (join1):

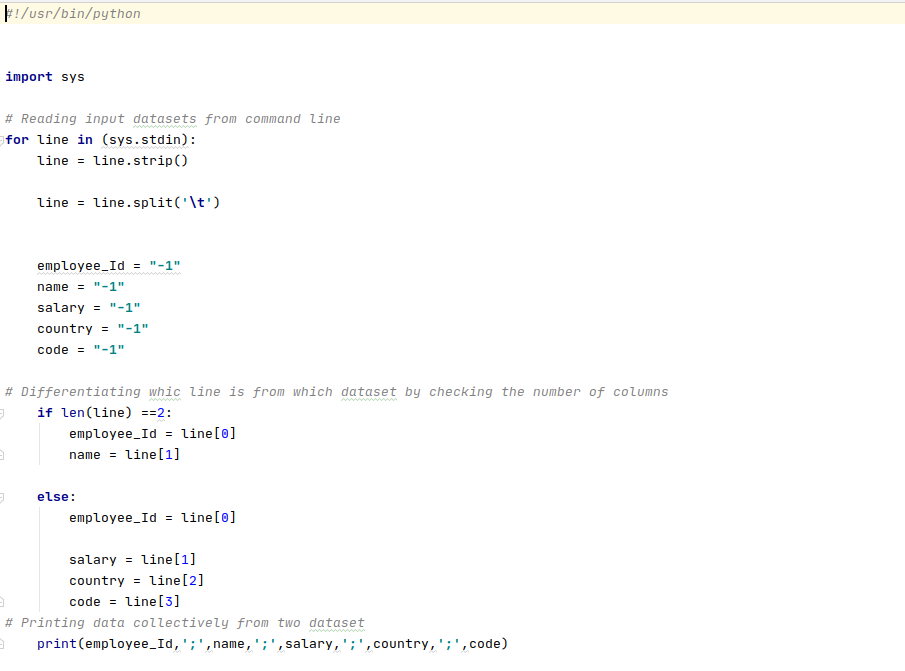


Dataset 2 (join2)



To perform the join, we have written mapper and reducer in python. Below is the snippet of the code:

Mapper.py:



Reducer:



Now we are all setup with the code and need to run on the Hadoop. To do this we have performed below steps

Step 1:

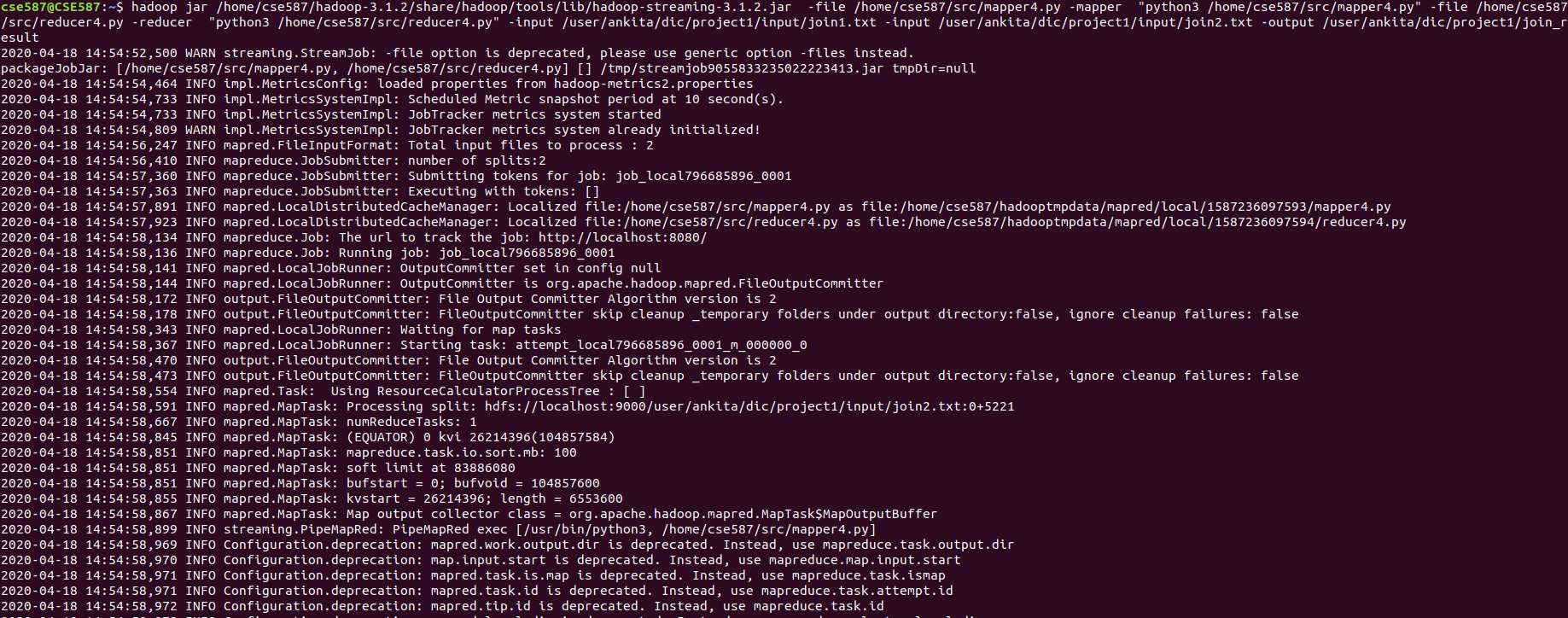
We have copied the input files join1 and join2 into Hadoop using the below command:

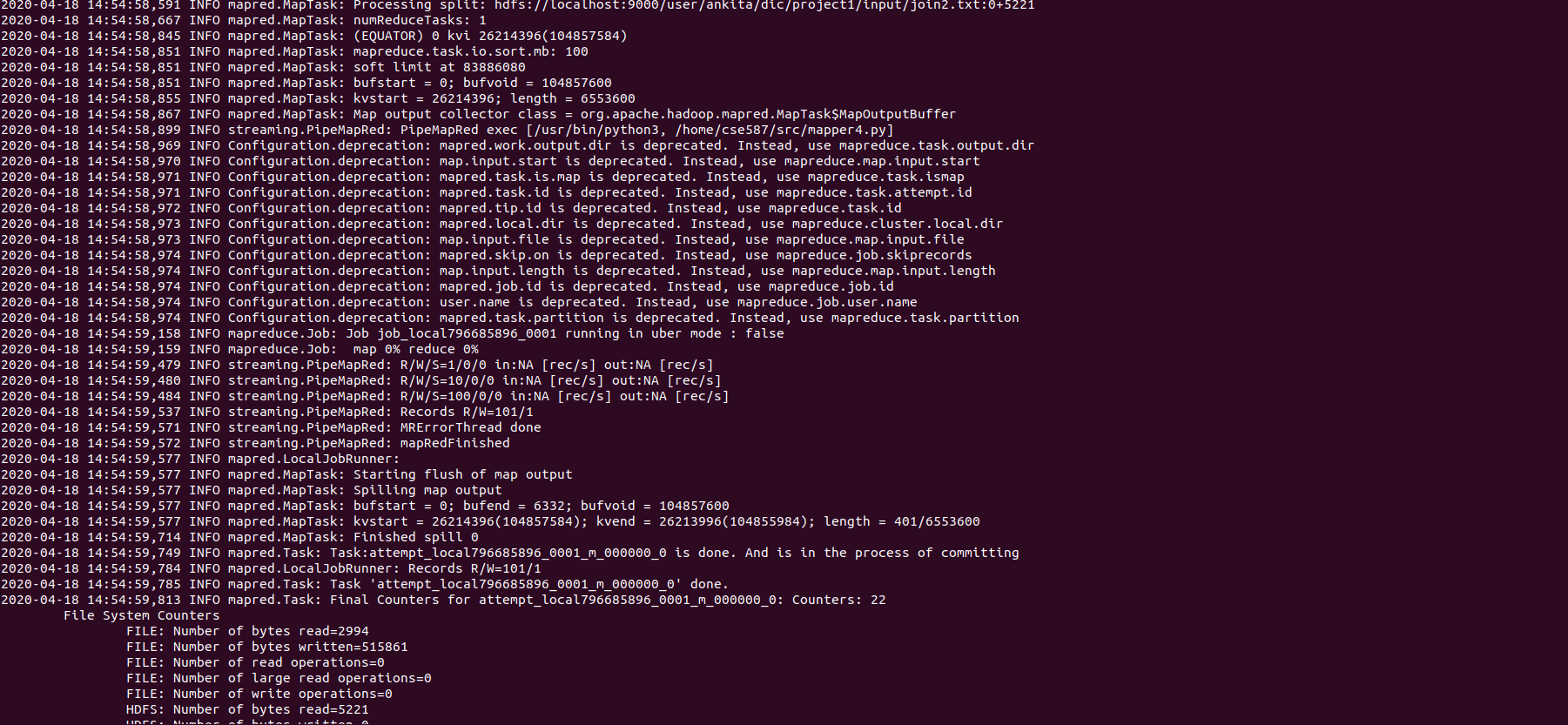


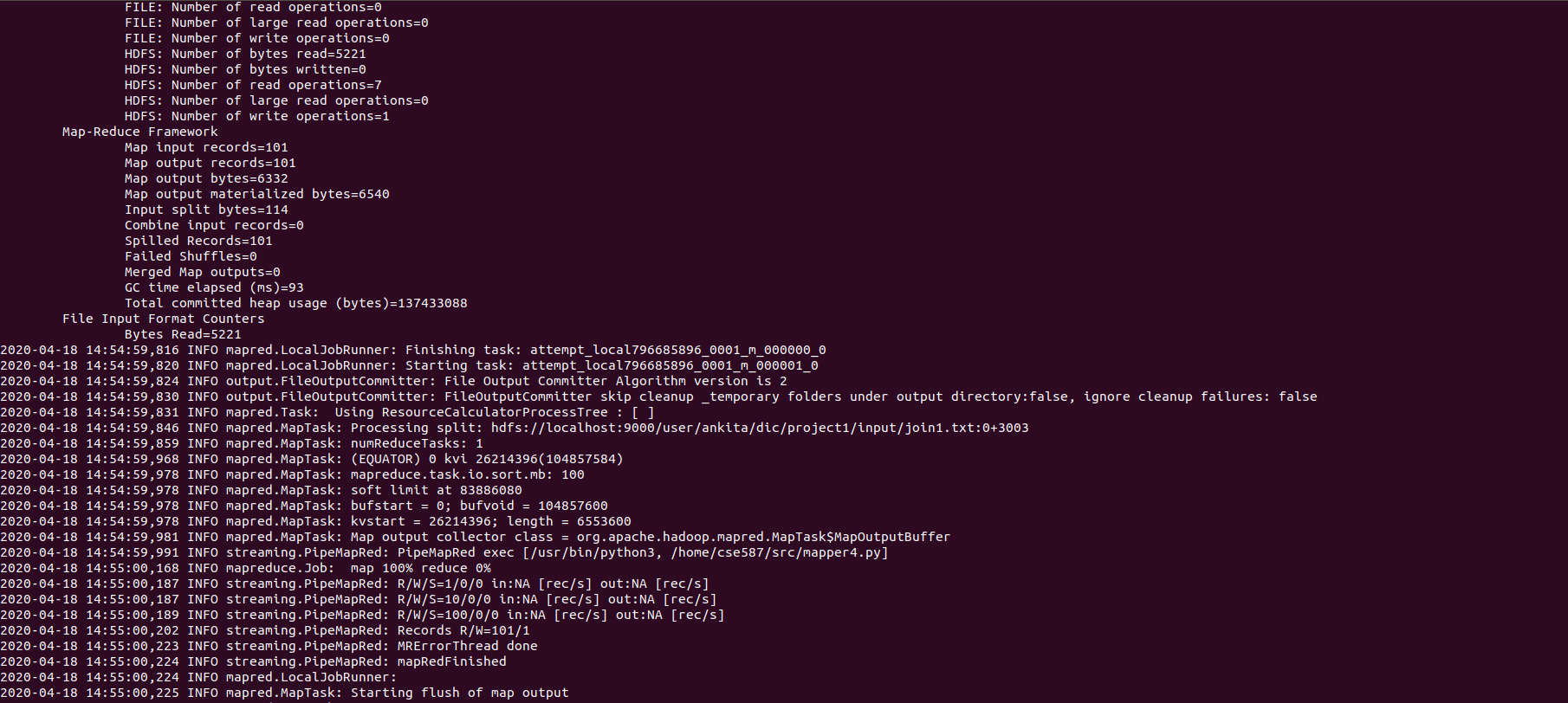
Step 2: We have already coded the mapper and reducer program in python. To run these two with given input we on Hadoop below command has been provided.



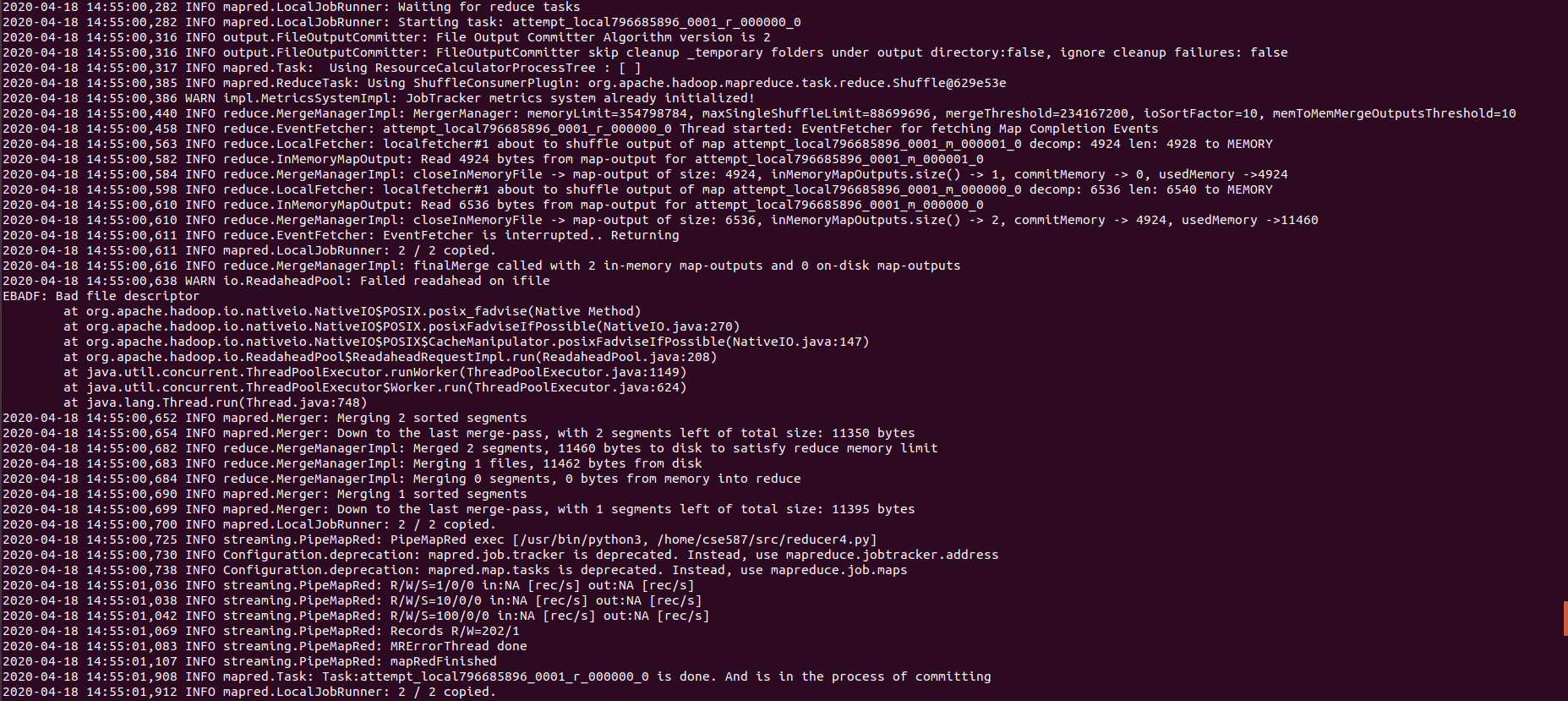
Running screen shots:



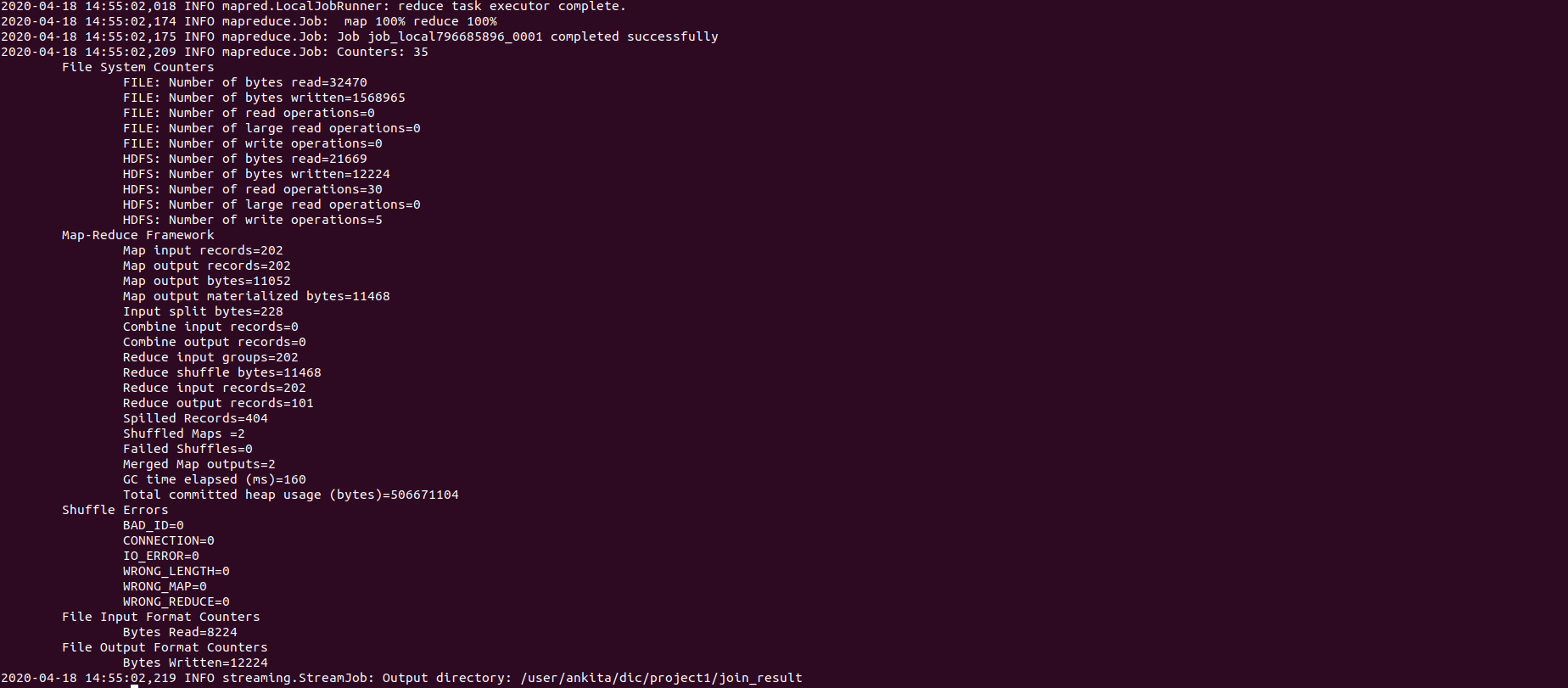












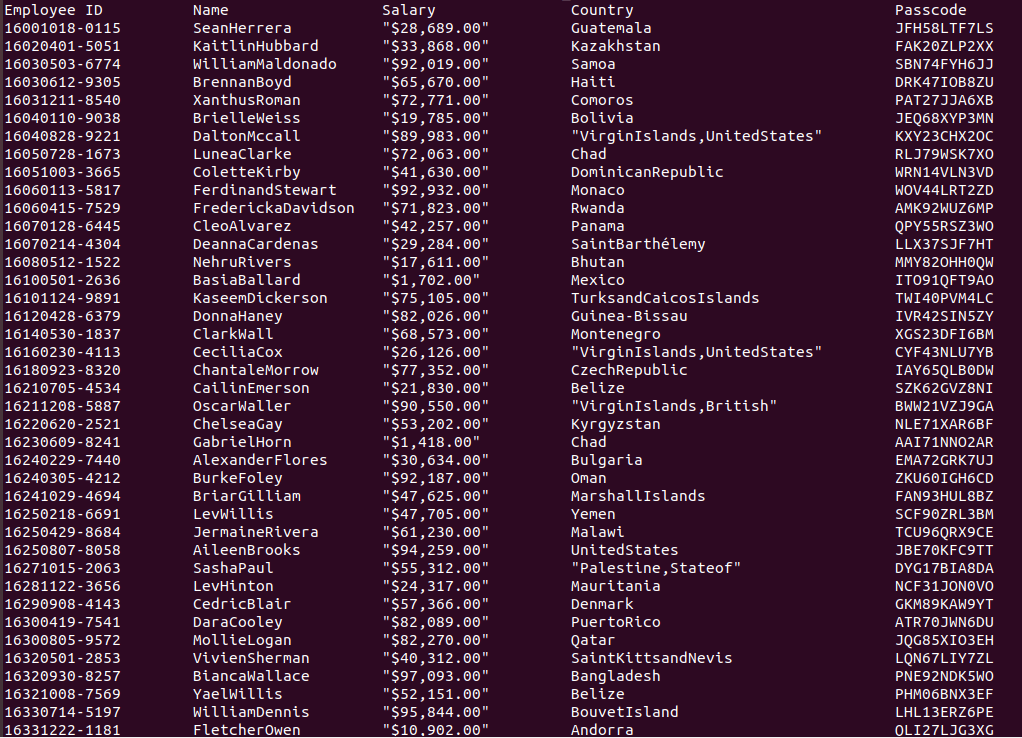
The command ran successfully.

Step 3:

We need to check the output file using below command:



A portion of the out put is as below:



Step 4:

To have this result set in local, below command has been provided:



The file join\_result.txt has been saved.

Please find the result dataset join\_result.txt attached herewith as below: *Double- Click the output4.txt below*



**BONUS: K-Nearest Neighbour - 5 Point** s

● Using the train and test set provided along with the assignment, Implement KNN

algorithm using MapReduce.

● You can assume the test set is small.

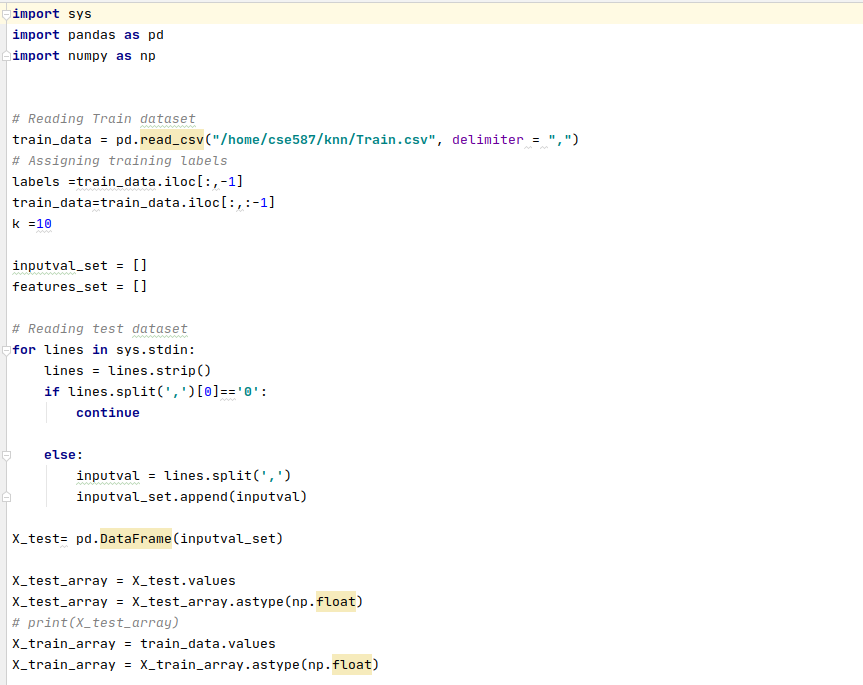
● The algorithm should return the corresponding predicted label for each test instance

**Solution:**

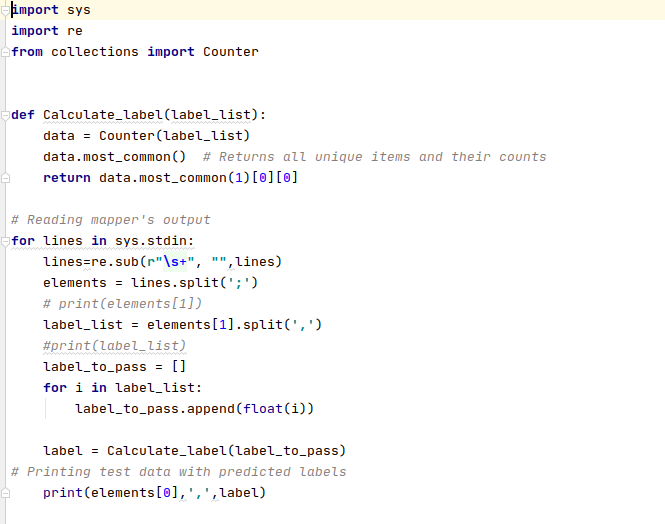
Here, we need to implement K-nearest-neighbor algorithm. We have been given a train dataset ehich is having 48 feature columns and 1 label column. A test dataset is also given, having 48 features. We need to predict the label for each test data.

To implement this algorithm, we have written corresponding mapper and reducer program in python. The mapper will read the datasets from command line and for every test data it will calculate the distance from all of the train data using Euclidean distance. This set of distances will be used to determine the nearest label for each test data and the most frequent one has been chosen in the reducer to be the predicted label.

Mapper.py



Reducer.py



Now we are all setup with the code and need to run on the Hadoop. To do this we have performed below steps

Step 1:

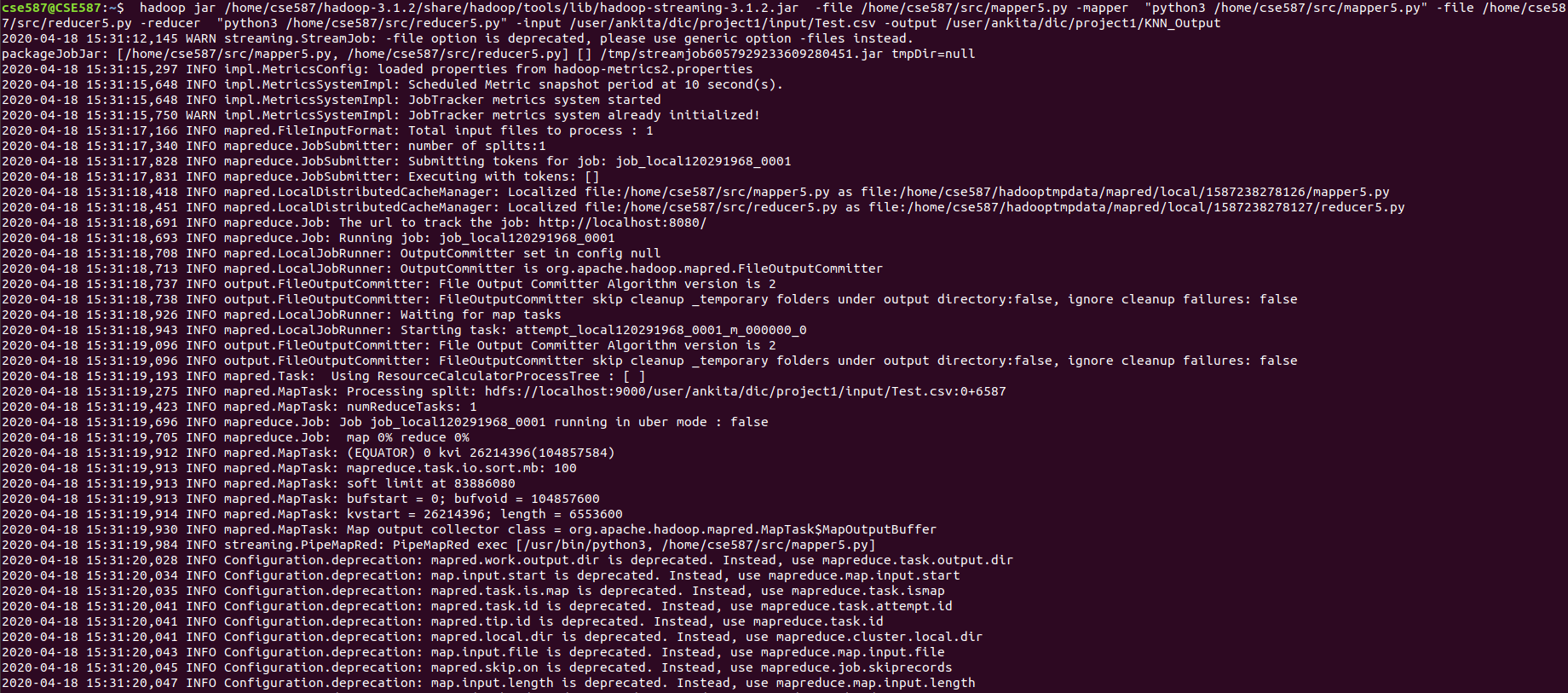
We have copied the input files into Hadoop using the below command:

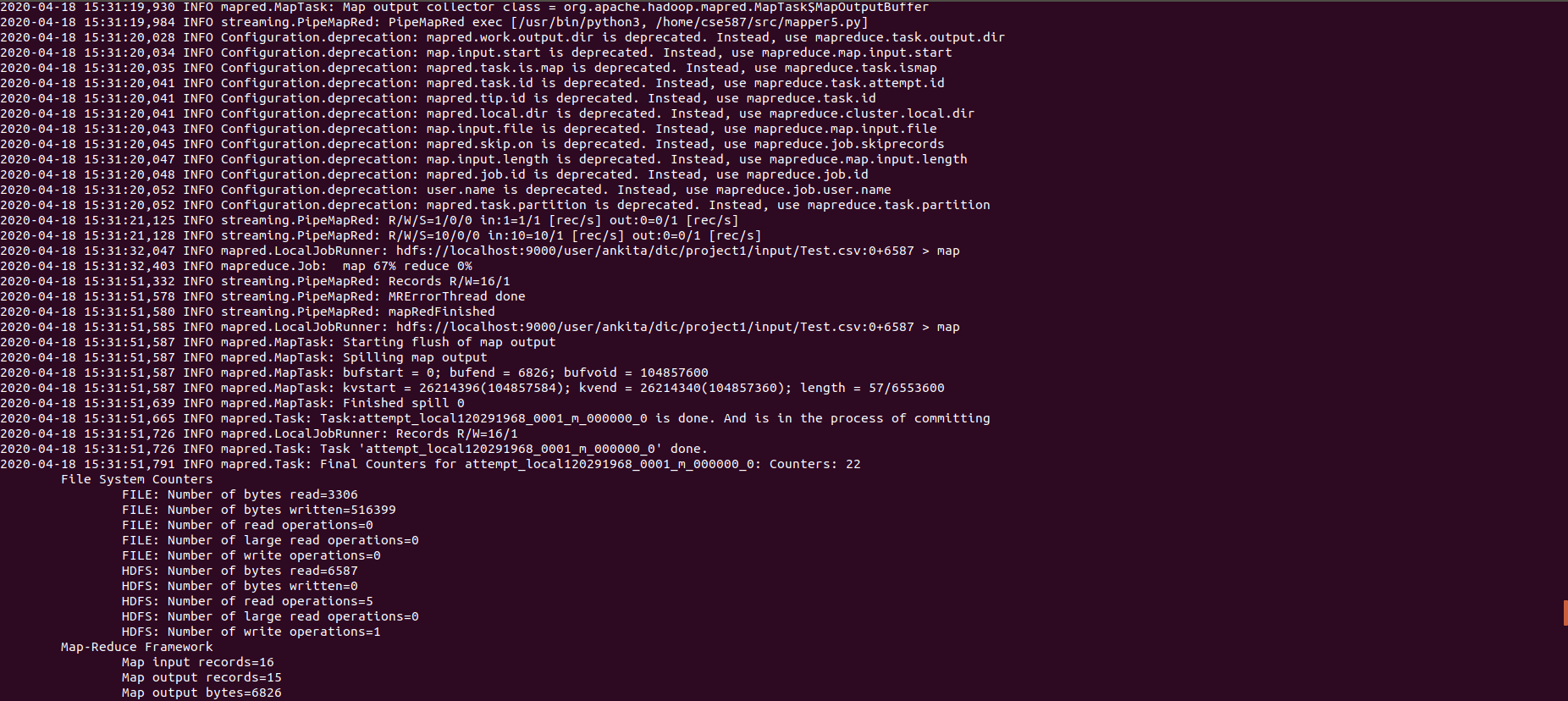


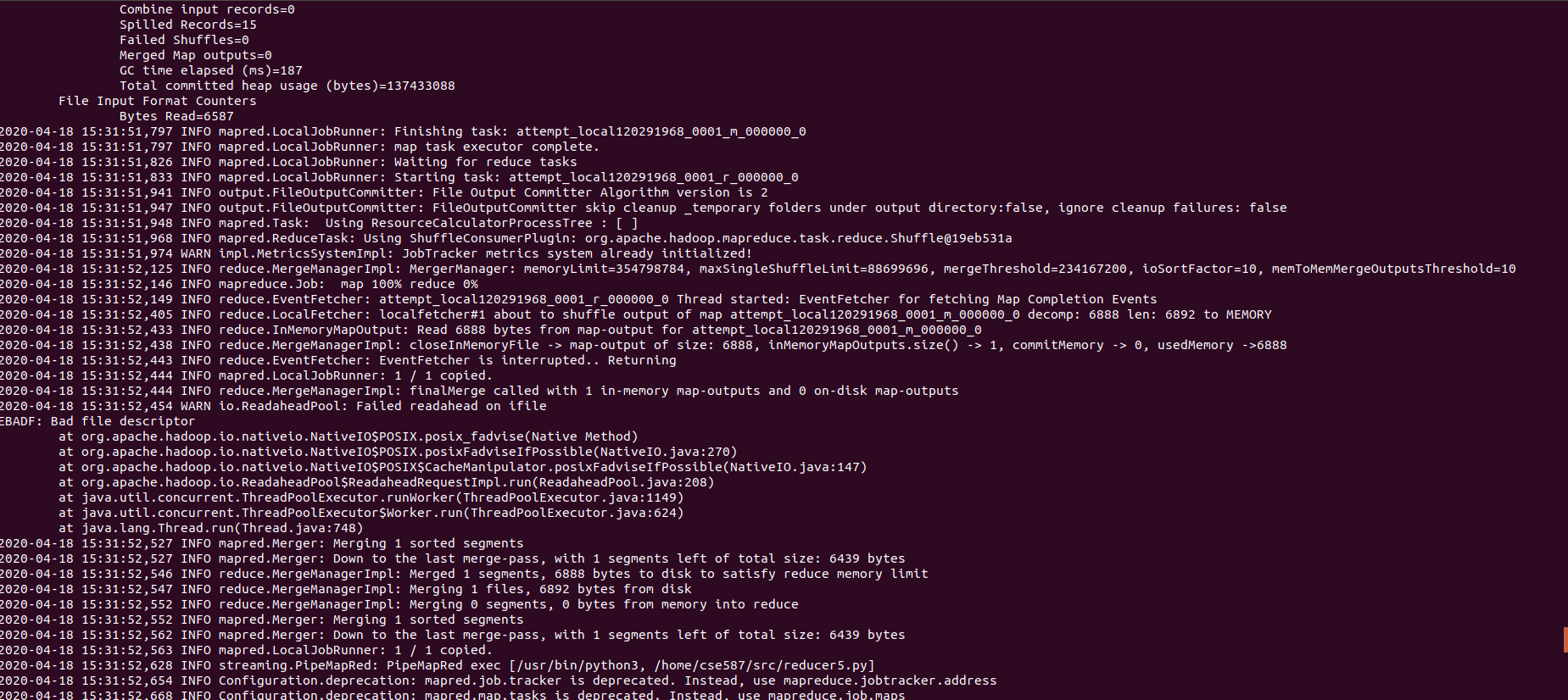
Step 2: We have already coded the mapper and reducer program in python. To run these two with given input we on Hadoop below command has been provided.

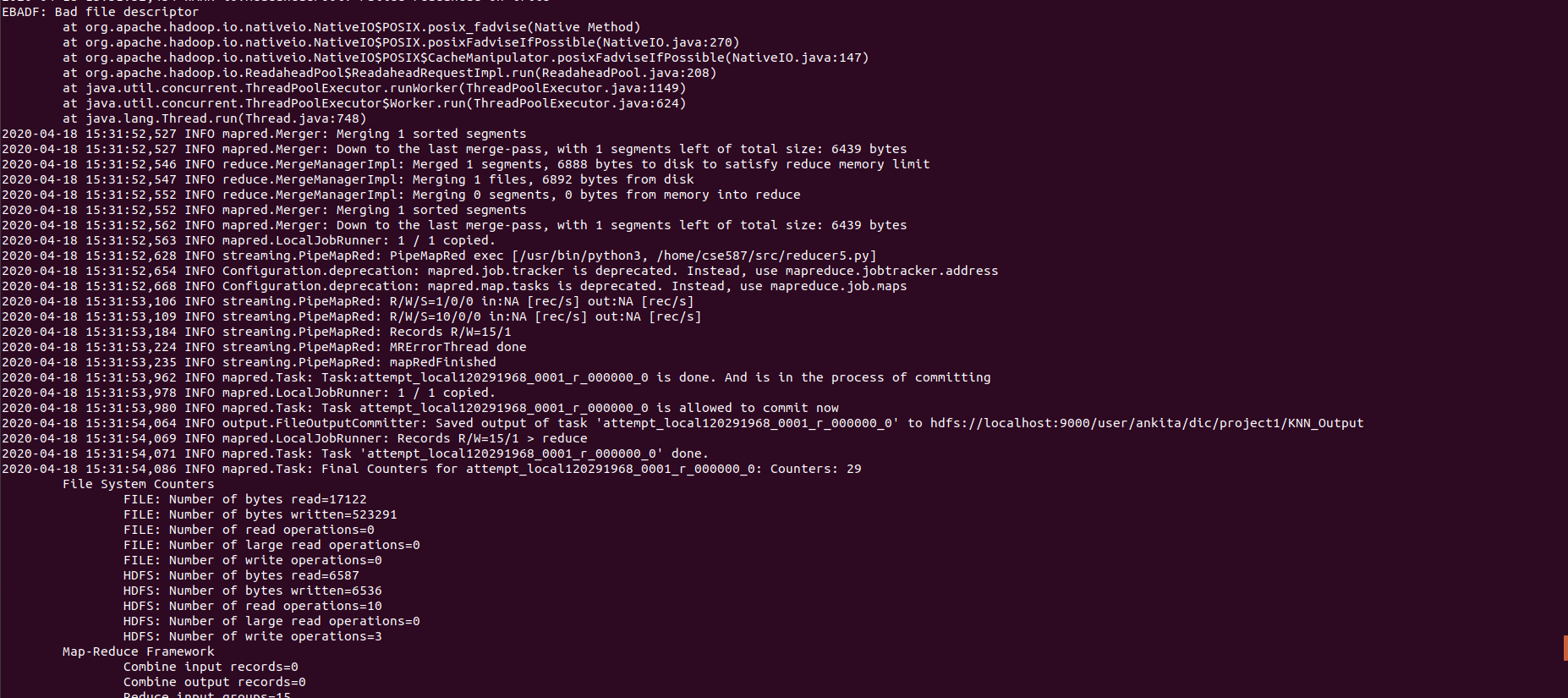


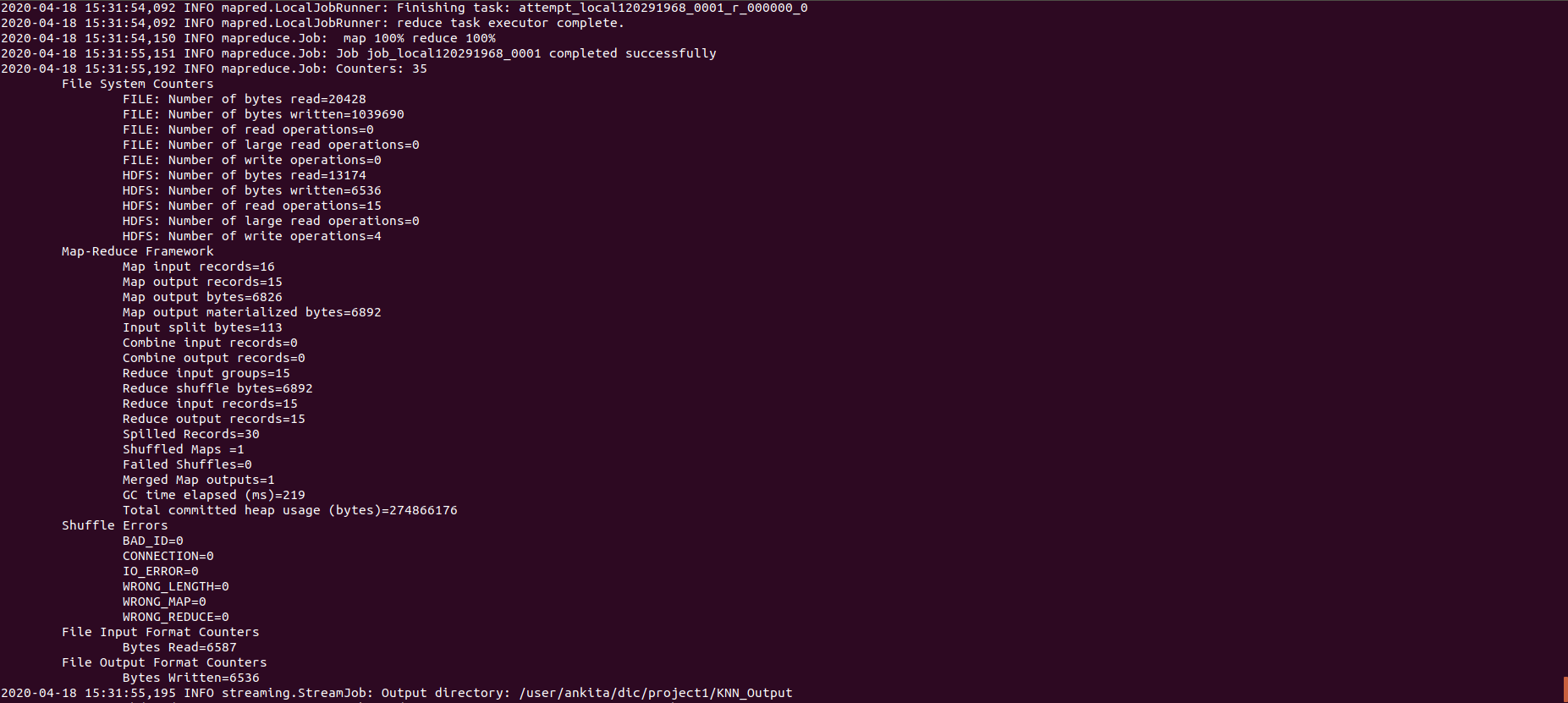
Running screen shots:











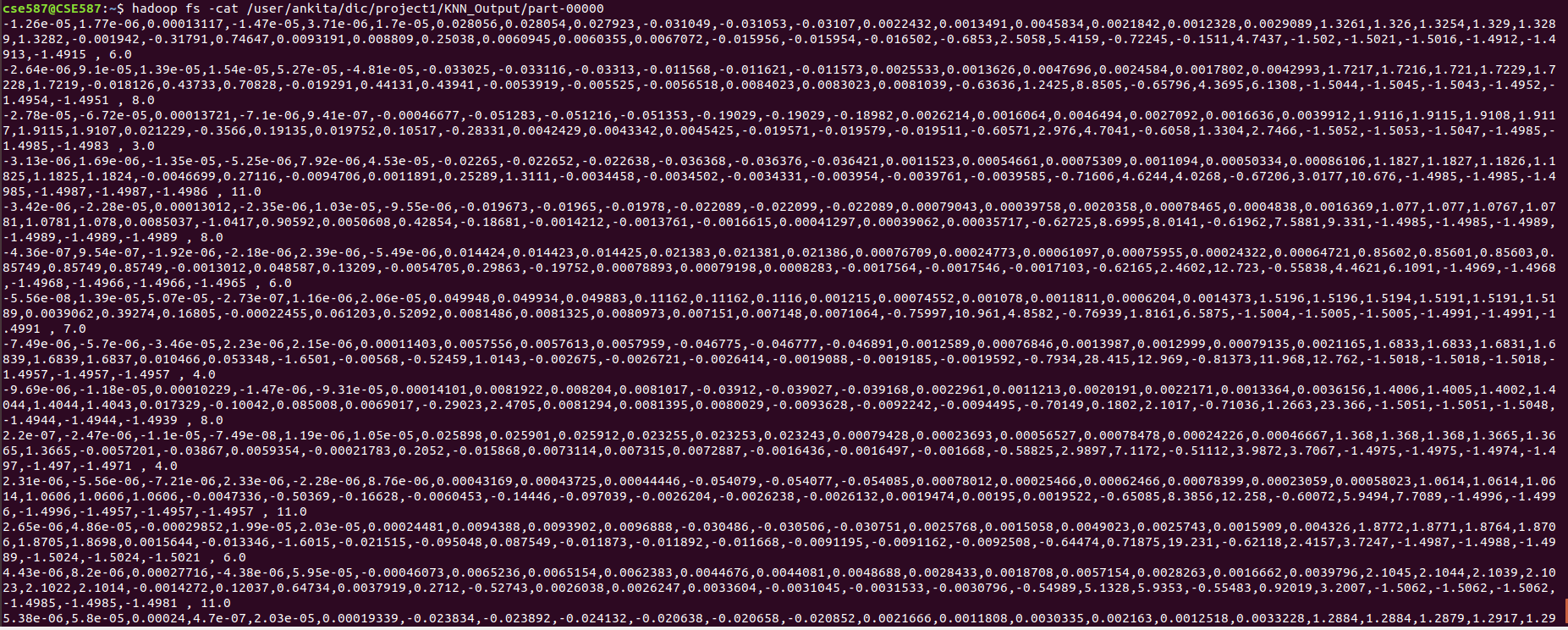
The command ran successfully.

Step 3:

We need to check the output file using below command:



A portion of the output is as below:



Step 4:

To have this result set in local, below command has been provided:



The file KNN\_Output.txt has been saved.

Please find the result dataset join\_result.txt attached herewith as below: *Double- Click the output5.txt below*



**Conclusion:**

This project depicts the development and simulation of Big data processing with Hadoop. We have performed all of the given tasks and the results for all were satisfactory.

**Acknowledgments:**

We are extremely grateful to Professor Deen Dayal Mohan and for teaching all the

necessary concepts related to Bigdata and Hadoop and helping in this project throughout. Also, we would like to express our sincere thanks and appreciation to Kyung Won Lee, Michael Long, Lawzeem, Chunwei Ma, Rui Li, Vinooth Rao Kulkarni to support us at every step where we find any difficulties.

**References:**

<https://stackoverflow.com/questions/30676448/extracting-n-grams-from-huge-text>

<https://pmatigakis.wordpress.com/2011/09/14/using-python-and-hadoop-streaming-to-build-an-inverted-index/>

<https://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/>

<https://blog.matthewrathbone.com/2016/02/09/python-tutorial.html>

<http://rare-chiller-615.appspot.com/mr2.html>

<http://www.cs.utah.edu/~lifeifei/knnj/>

<https://www.hadoop360.datasciencecentral.com/blog/k-nearest-neighbor-algorithm-using-python>