# CSEE5590 Big Data Programming

**In Class Programming –1 Report**
**(Jongkook Son)**

## Project Overview:

**Hadoop MapReduce and Hadoop Distributed File System (HDFS)**
**In this lesson, we are going to discuss about Hadoop MapReduce and Hadoop Distributed File System (HDFS)**

## Requirements/Task(s):

There are many ways to execute wordcount program:

1. Using any IDE like Intellij or Eclipse

2. Run on hadoop clusters

1. Counting the frequency of words in the given input with MapReduce algorithm

Use the following text file to count the frequency of words

2. Counting the frequency of words in given text file that starts with letter ‘a’

3. Determine the prime number in input and print number only once

## What I learned in ICP:

I could have learned the process of the MapReduce. A MapReduce job usually splits the input data-set into independent block which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. Thanks to this ICP2, I could have understand this process more deeply. In our code TokenizerMapper act as Mapper and IntSumReducer act as Reducer. By modifying Reducer, I could achieve other tasks given.

**ICP description what was the task you were performing and Screen shots that shows the successful execution of each required step of your code**

## TASK 1(Word Count)

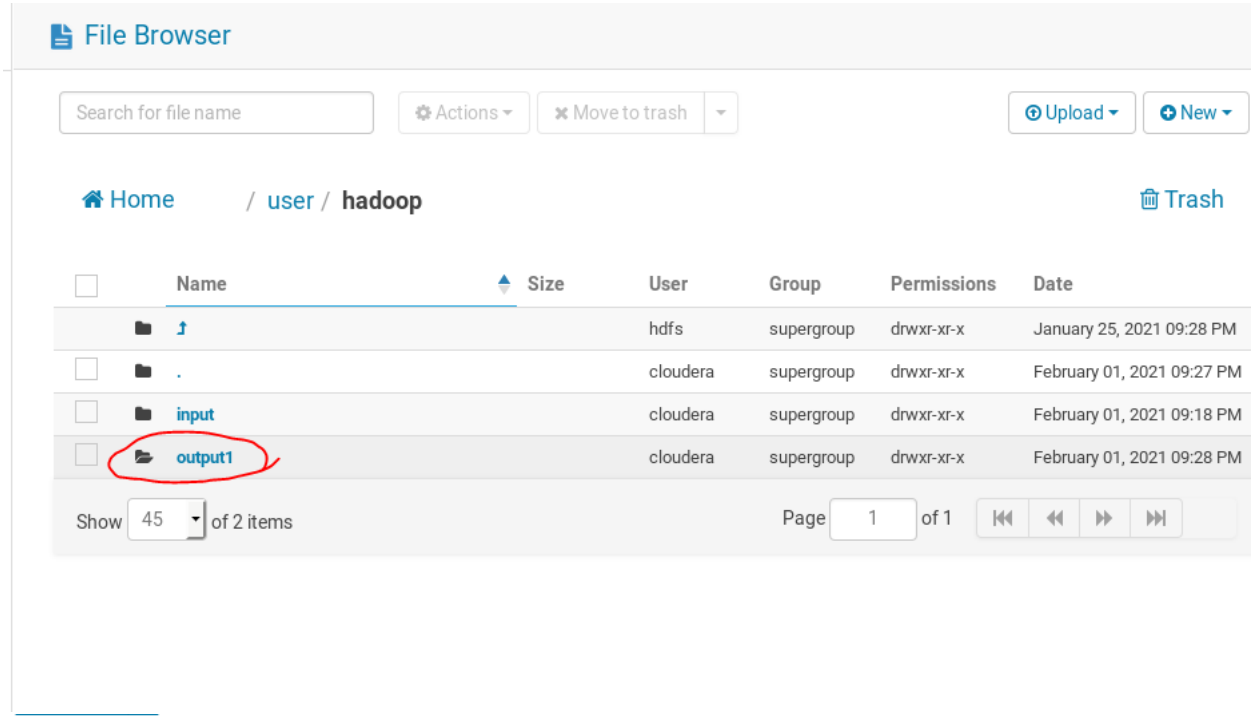
```
public static class TokenizerMapper//Mapper Class
     extends Mapper<Object, Text, Text, IntWritable>{
    //Input key, Input value, output key, output value

  private final static IntWritable one = new IntWritable(1);
  // Hadoop-only data type just like int
  private Text word = new Text();

  public void map(Object key, Text value, Context context
                  ) throws IOException, InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
    //Seperate String with Token using tokenizer(default seperator is space)
    while (itr.hasMoreTokens()) { //If there is more than one token implement function
      word.set(itr.nextToken()); // Send a parameter of key-value Reducer
      context.write(word, one); //Save as write file
    }
  }
}

public static class IntSumReducer //Reducer Class
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

  public void reduce(Text key, Iterable<IntWritable> values,
                             //Value with which the same key value can exist is passed to collection
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) { //Each object's value is fixed to 1
      sum += val.get(); // if it is same key then increase the sum
    }
    result.set(sum);
```

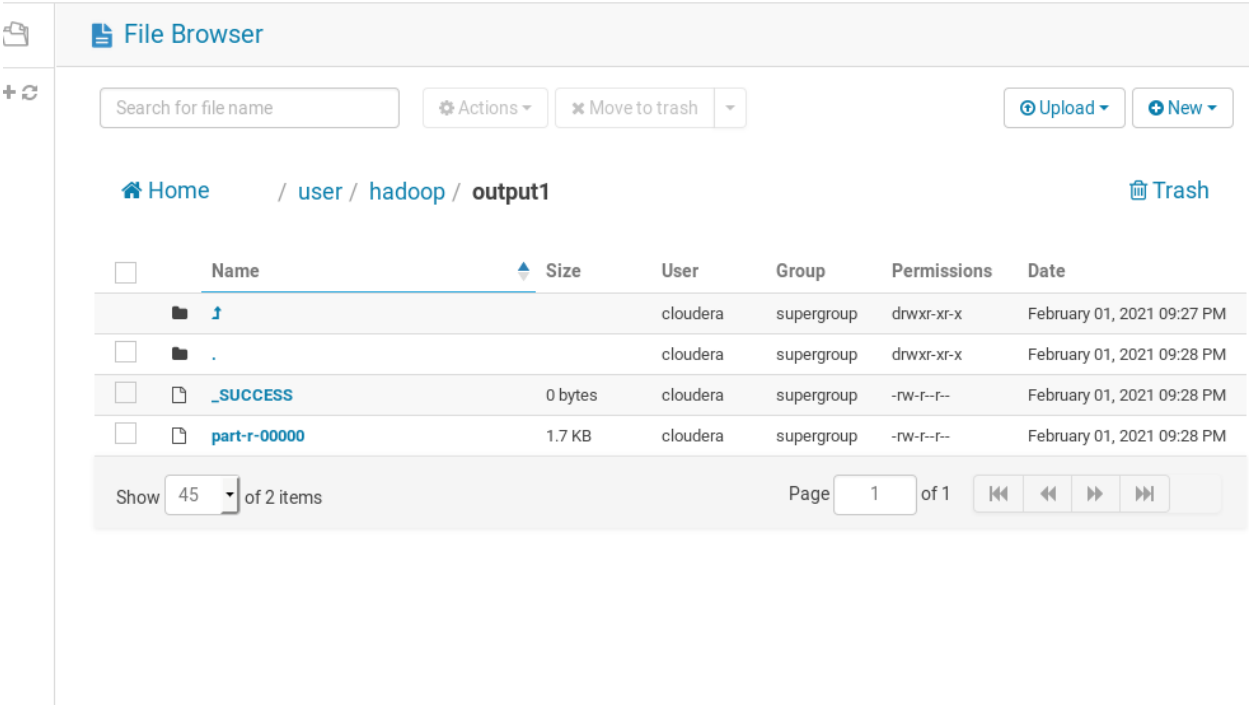
```
 public static void main(String[] args) throws Exception {
   Configuration conf = new Configuration();
   Job job = Job.getInstance(conf, "word count"); // make instance of Job
   job.setJarByClass(WordCount.class);
   job.setMapperClass(TokenizerMapper.class);
   job.setCombinerClass(IntSumReducer.class);
   job.setReducerClass(IntSumReducer.class); //Reduce: finish
   job.setOutputKeyClass(Text.class);
   job.setOutputValueClass(IntWritable.class);
   FileInputFormat.addInputPath(job, new Path(args[0])); //input dir
   FileOutputFormat.setOutputPath(job, new Path(args[1]));//output dir
   System.exit(job.waitForCompletion(true) ? 0 : 1);
 }
```

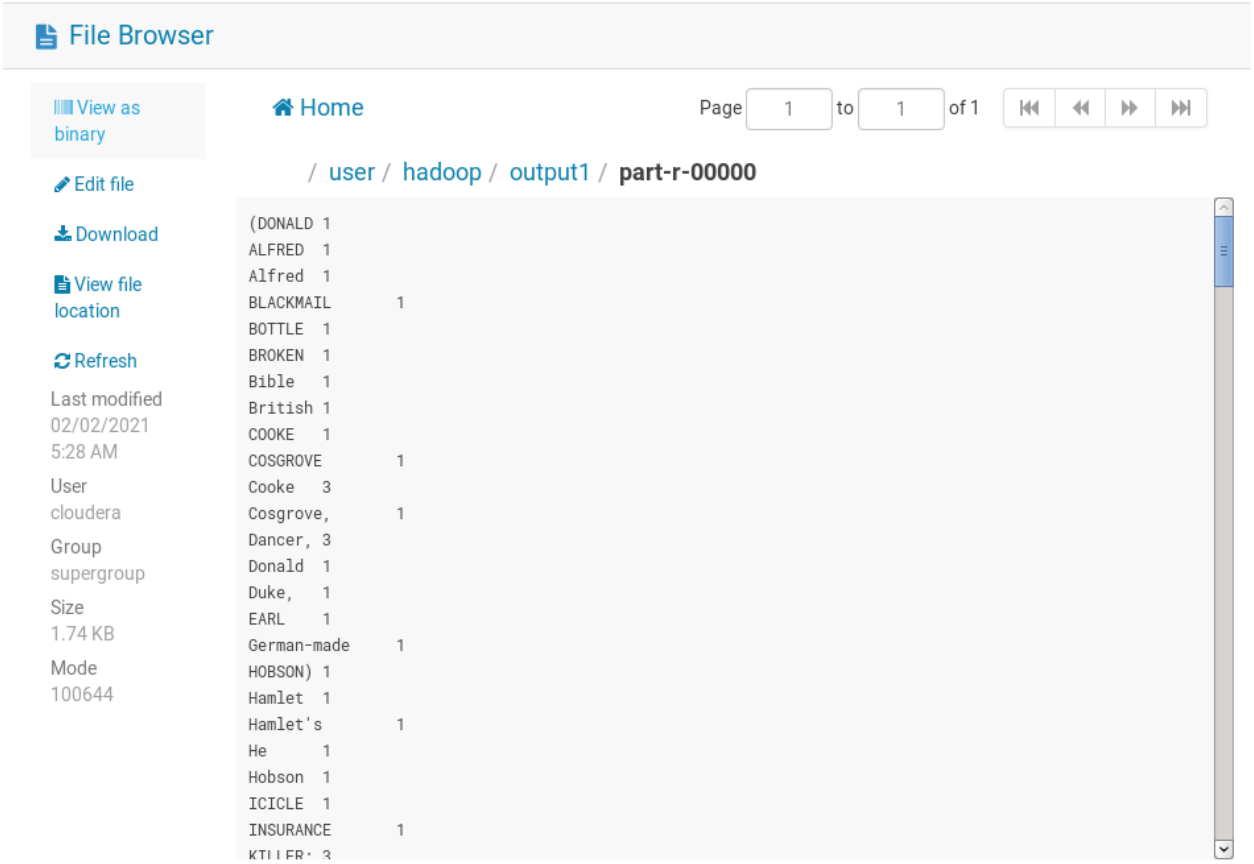
**<Code Implementation>**

⇨ **I used the given source code, It is divided into Mapper, Reducer, Drive code**

```
[cloudera@quickstart ~]$ hadoop jar Wordcount1.jar /user/hadoop/input/sample/txt /user/hadoop/output1
```




File Browser
View as binary
Edit file
Download
View file location
Refresh
Last modified
02/02/2021
5:28 AM
User
cloudera
Group
supergroup
Size
1.74 KB
Mode
100644
Home
Page 1 to 1 of 1
/ user / hadoop / output1 / part-r-00000
(DONALD 1
ALFRED 1
Alfred 1
BLACKMAIL 1
BOTTLE 1
BROKEN 1
Bible 1
British 1
COOKE 1
COSGROVE 1
Cooke 3
Cosgrove, 1
Dancer, 3
Donald 1
Duke, 1
EARL 1
German-made 1
HOBSON) 1
Hamlet 1
Hamlet's 1
He 1
Hobson 1
ICICLE 1
INSURANCE 1

**<Output>**

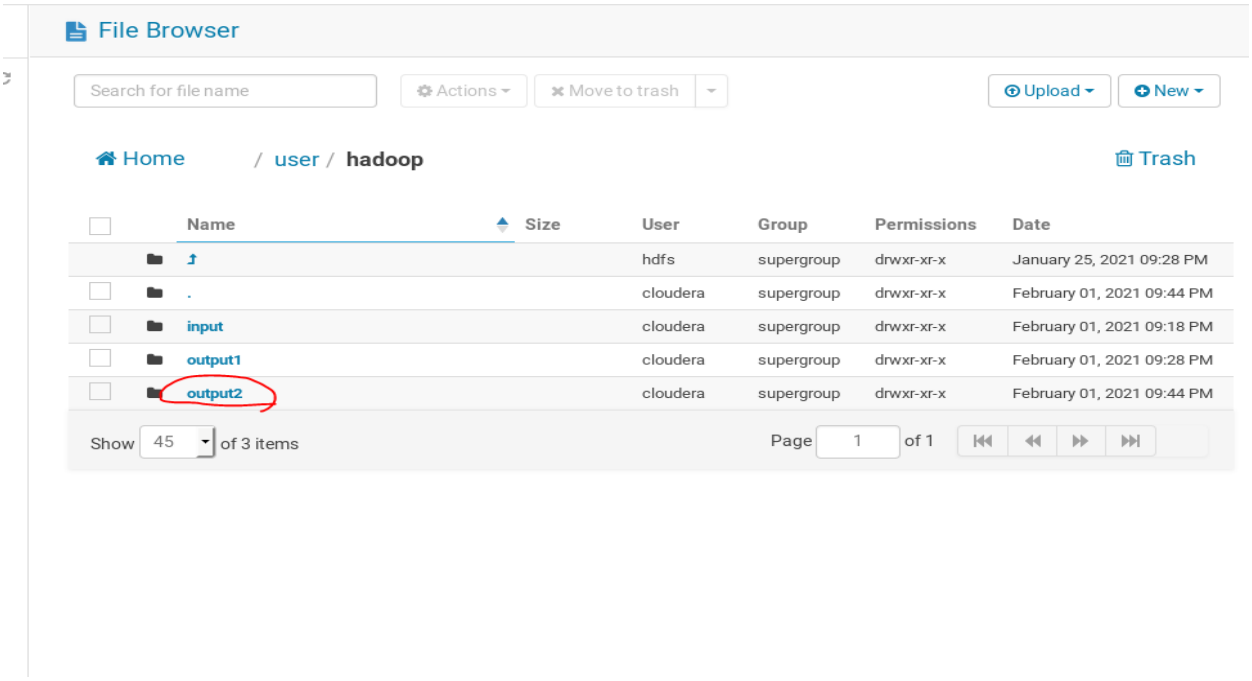
## RESULT OF TASK2(Count words Starts with “a”)

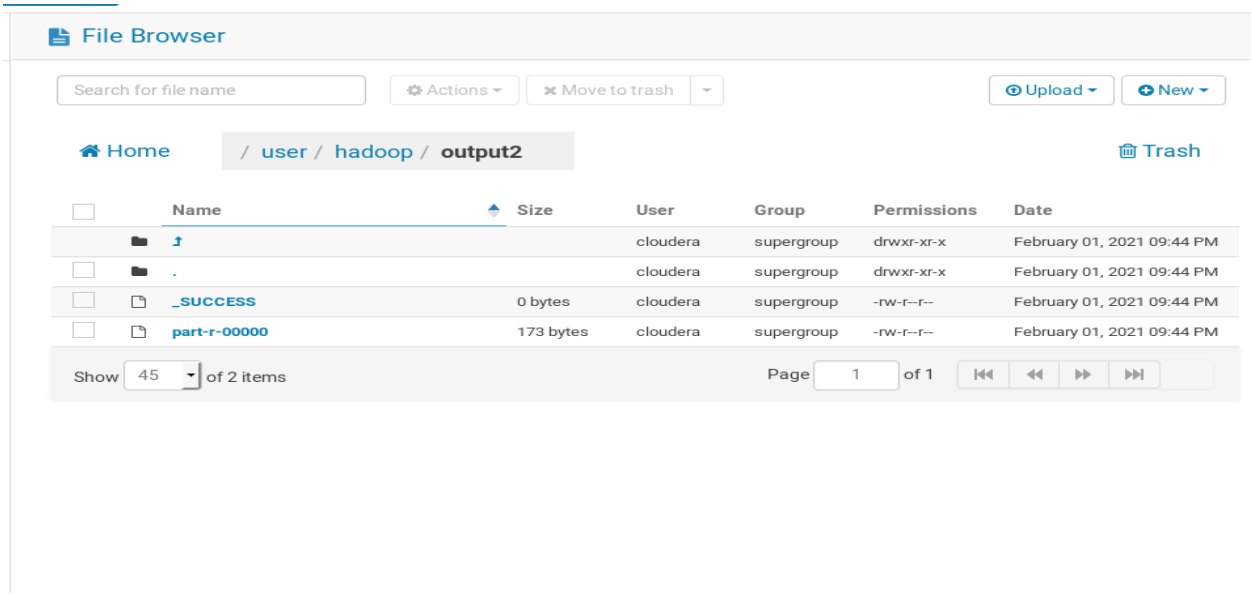
```
public static class IntSumReducer //Modified Reducer Class
        extends Reducer<Text, IntWritable, Text, IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
            //Value with which the same key value can exist is passed to collection
                    Context context
    ) throws IOException, InterruptedException {
        int sum = 0;
        if(key!=null && key.toString().toLowerCase().startsWith("a")){ //Only iterate if the first letter starts with a
        for (IntWritable val : values) { //Each object's value is fixed to 1
            sum += val.get(); // if it is same key then increase the sum
        }
        result.set(sum);
        context.write(key, result);
        }
    }
}
```

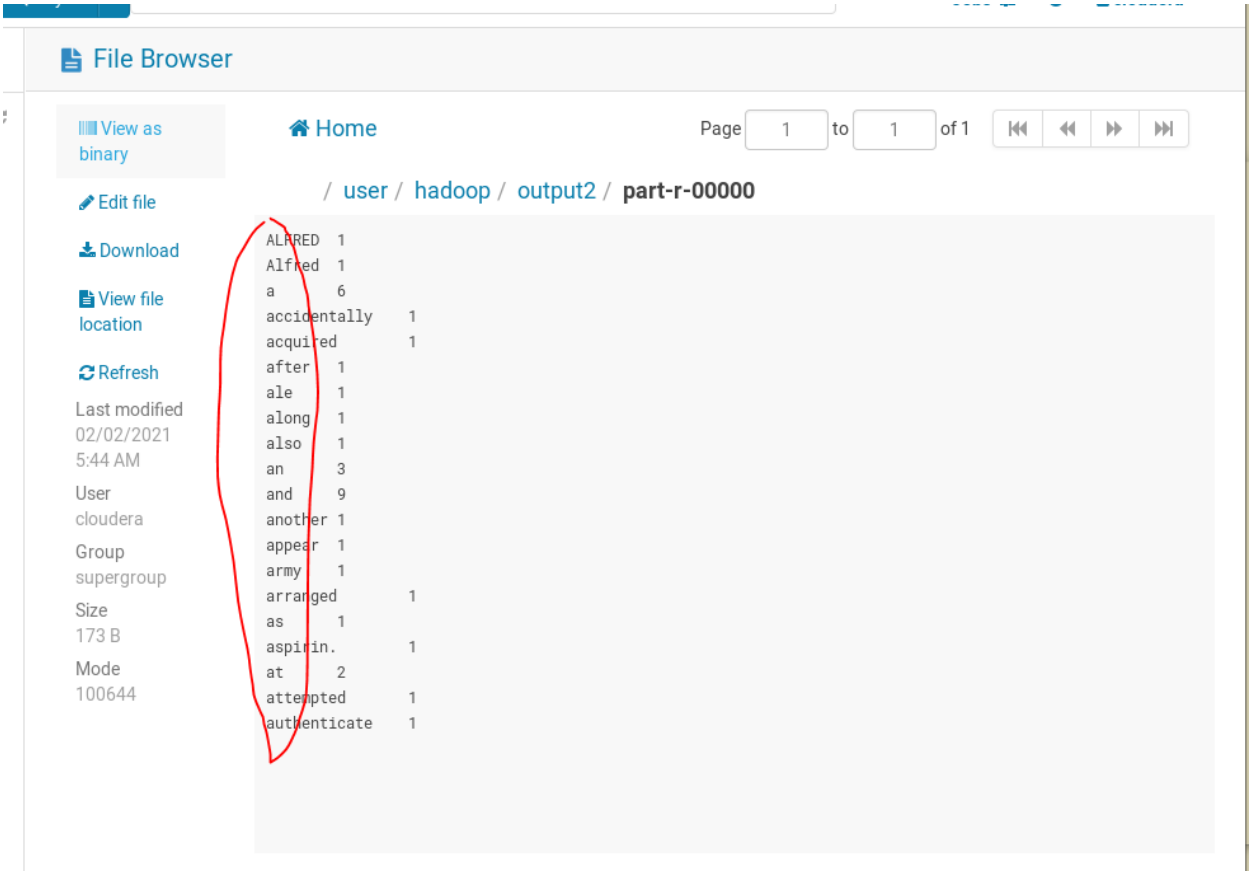
**<Code Implementation>**

⇨ **To fulfill this task, We can either change Mapper or Reducer. In my case, I changed Reducer. Every code is the same with word count source code except that In Reducer only process when key value, which is the word, is equated with “a”**

```
[cloudera@quickstart ~]$ hadoop jar Wordcount2.jar /user/hadoop/input/sample/txt /user/hadoop/output2
```




File Browser
Search for file name
Actions
Move to trash
Upload
New
Home
/ user / hadoop / output2
Trash
Name
Size
User
Group
Permissions
Date
cloudera supergroup drwxr-xr-x February 01, 2021 09:44 PM
cloudera supergroup drwxr-xr-x February 01, 2021 09:44 PM
_SUCCESS 0 bytes cloudera supergroup -rw-r--r-- February 01, 2021 09:44 PM
part-r-00000 173 bytes cloudera supergroup -rw-r--r-- February 01, 2021 09:44 PM
Show 45 of 2 items
Page 1 of 1


File Browser
View as binary
Edit file
Download
View file location
Refresh
Last modified 02/02/2021 5:44 AM
User cloudera
Group supergroup
Size 173 B
Mode 100644
Home
Page 1 to 1 of 1
/ user / hadoop / output2 / part-r-00000
ALFRED 1
Alfred 1
a 6
accidentally 1
acquired 1
after 1
ale 1
along 1
also 1
an 3
and 9
another 1
appear 1
army 1
arranged 1
as 1
aspirin. 1
at 2
attempted 1
authenticate 1

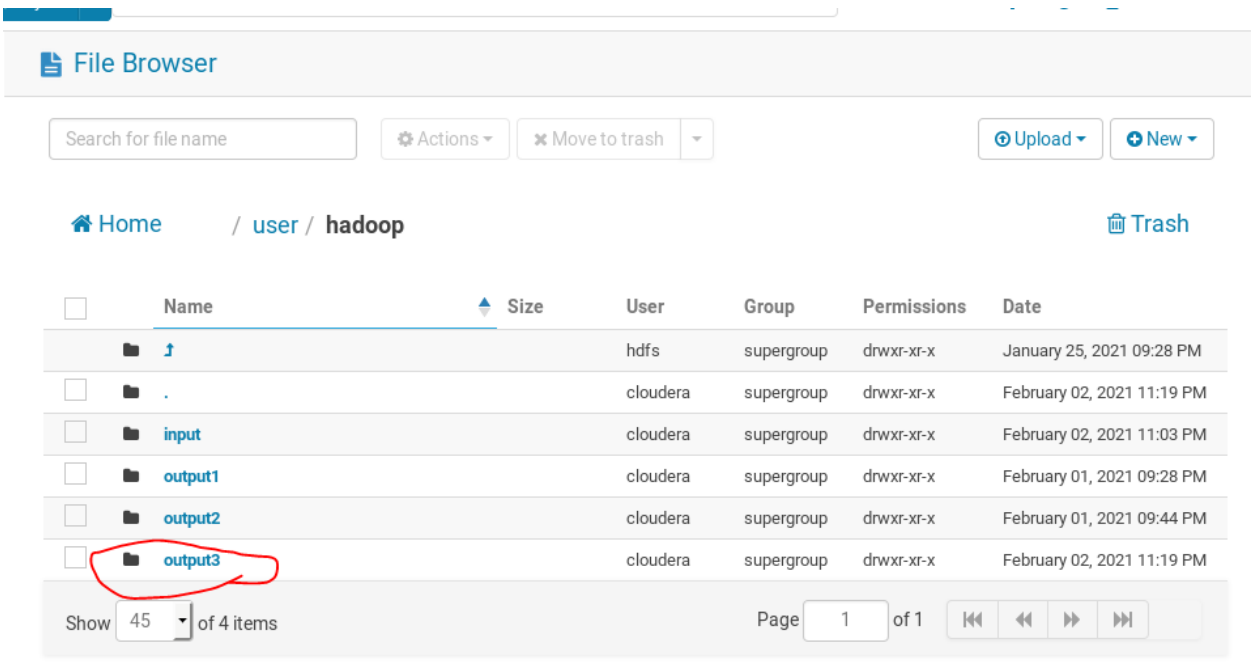
**<Output>**

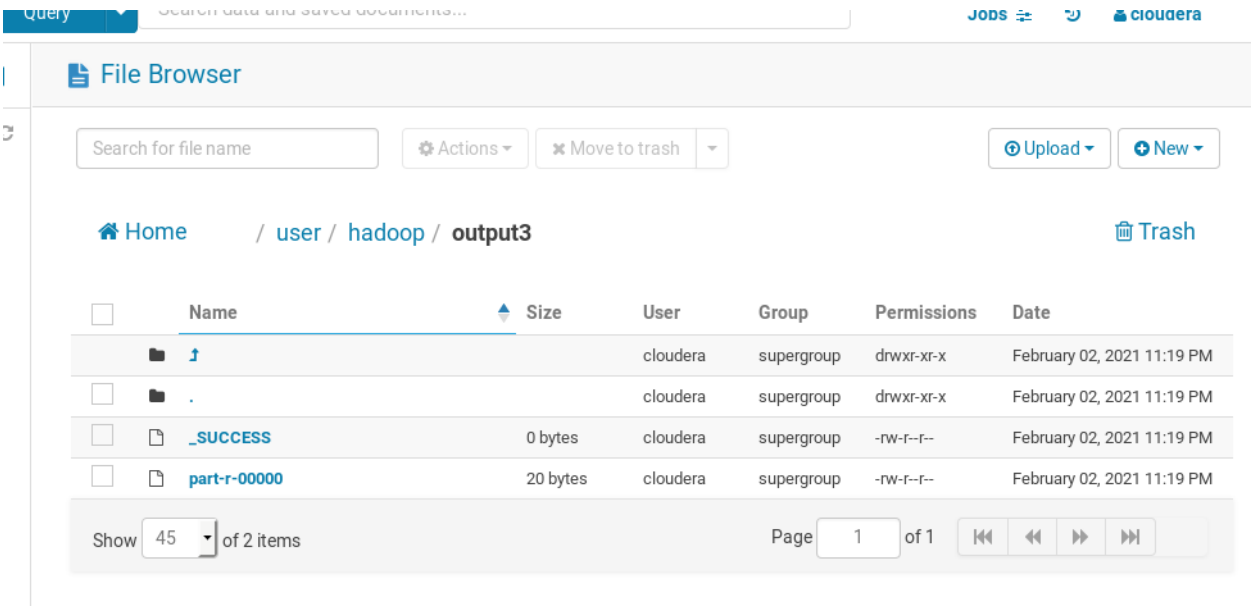
## RESULT OF TASK3(Determine the prime number in input)

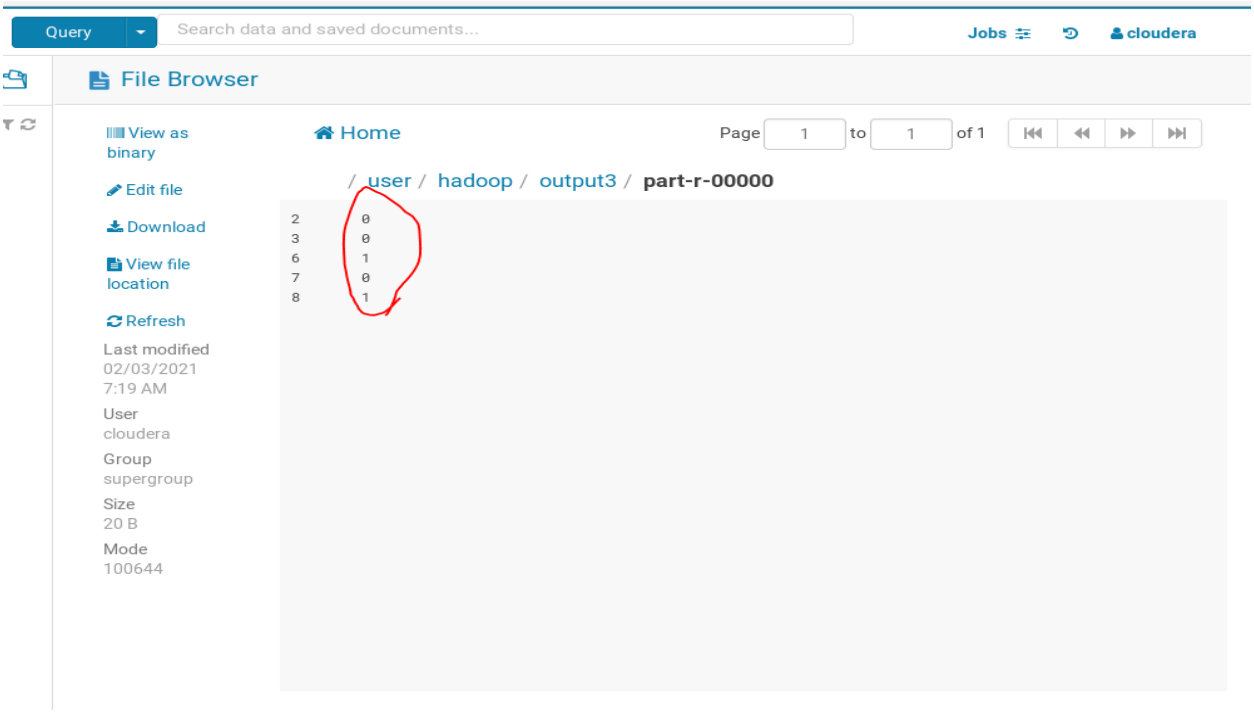
```
public static class IntSumReducer //Modified Reducer Class for prime number
        extends Reducer<Text, IntWritable, Text, IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
         //Value with which the same key value can exist is passed to collection
                       Context context
    ) throws IOException, InterruptedException {
        int sum = 0;
        int i_key=Integer.parseInt(key.toString());
        for (int i=2; i<i_key/2; i++) {
            if(i_key%i==0){
         //If certain number is divided by number except 1 and itself then it is not prime number
                sum=1;
                break;
            }
        }
        //If sum is 0, then prime number and If 1, then not prime number
        result.set(sum);
        context.write(key, result);
        }
 }
```

⇨ **Like TASK 2, To fulfill Task 3, We need to change the reducer code from the source code. First thing need to add is parse key value to int then insert a code that determines key is prime number or not**

```
[cloudera@quickstart ~]$ hadoop jar Primenumber.jar /user/hadoop/input/numbers.txt /user/hadoop/output3
21/02/02 23:19:26 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
21/02/02 23:19:26 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement
he Tool interface and execute your application with ToolRunner to remedy this.
21/02/02 23:19:27 INFO input.FileInputFormat: Total input paths to process : 1
21/02/02 23:19:27 INFO mapreduce.JobSubmitter: number of splits:1
21/02/02 23:19:27 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1612331613408_0002
21/02/02 23:19:28 INFO impl.YarnClientImpl: Submitted application application_1612331613408_0002
21/02/02 23:19:28 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application
612331613408_0002/
21/02/02 23:19:28 INFO mapreduce.Job: Running job: job_1612331613408_0002
21/02/02 23:19:39 INFO mapreduce.Job: Job job_1612331613408_0002 running in uber mode : false
21/02/02 23:19:39 INFO mapreduce.Job:  map 0% reduce 0%
21/02/02 23:19:48 INFO mapreduce.Job:  map 100% reduce 0%
21/02/02 23:19:55 INFO mapreduce.Job:  map 100% reduce 100%
21/02/02 23:19:55 INFO mapreduce.Job: Job job_1612331613408_0002 completed successfully
21/02/02 23:19:55 INFO mapreduce.Job: Counters: 49
```


File Browser
Search for file name
Actions
Move to trash
Upload
New
Home / user / hadoop
Trash
Name
Size
User
Group
Permissions
Date
hdfs supergroup drwxr-xr-x January 25, 2021 09:28 PM
. cloudera supergroup drwxr-xr-x February 02, 2021 11:19 PM
input cloudera supergroup drwxr-xr-x February 02, 2021 11:03 PM
output1 cloudera supergroup drwxr-xr-x February 01, 2021 09:28 PM
output2 cloudera supergroup drwxr-xr-x February 01, 2021 09:44 PM
output3 cloudera supergroup drwxr-xr-x February 02, 2021 11:19 PM
Show 45 of 4 items
Page 1 of 1


Query
Jobs
cloudera
File Browser
Search for file name
Actions
Move to trash
Upload
New
Home / user / hadoop / output3
Trash
Name
Size
User
Group
Permissions
Date
cloudera supergroup drwxr-xr-x February 02, 2021 11:19 PM
. cloudera supergroup drwxr-xr-x February 02, 2021 11:19 PM
_SUCCESS 0 bytes cloudera supergroup -rw-r--r-- February 02, 2021 11:19 PM
part-r-00000 20 bytes cloudera supergroup -rw-r--r-- February 02, 2021 11:19 PM
Show 45 of 2 items
Page 1 of 1


Query
Search data and saved documents...
Jobs
cloudera
File Browser
View as binary
Edit file
Download
View file location
Refresh
Last modified
02/03/2021 7:19 AM
User
cloudera
Group
supergroup
Size
20 B
Mode
100644
Home
Page 1 to 1 of 1
/ user / hadoop / output3 / part-r-00000
2 0
3 0
6 1
7 0
8 1

**<Output>**