STUDENT ID :- 240840325025

DATA SET: Dataset: MovieLens dataset which is available at: <https://grouplens.org/datasets/movielens/latest/>

Write a Mapper and Reducer code to find the average rating for each movie from a dataset containing movie ratings.

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

public class RatingMovie {

public static class MapClass extends Mapper<LongWritable, Text, LongWritable, DoubleWritable> {

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

try {

String[] str = value.toString().split(",");

long movieId = Long.*parseLong*(str[1]);

double rating = Double.*parseDouble*(str[2]);

context.write(new LongWritable(movieId), new DoubleWritable(rating));

} catch (Exception e) {

System.***out***.println(e.getMessage());

}

}

}

public static class ReduceClass extends Reducer<LongWritable,DoubleWritable,LongWritable,DoubleWritable>

{

private DoubleWritable result = new DoubleWritable();

public void reduce(LongWritable key, Iterable<DoubleWritable> values,Context context) throws IOException, InterruptedException {

double sum = 0;

double cnt=0;

double avg=0;

for (DoubleWritable val : values)

{

sum += val.get();

cnt++;

}

avg = sum/cnt;

result.set(avg);

context.write(key, result);

//context.write(key, new LongWritable(sum));

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

//conf.set("name", "value")

//conf.set("mapreduce.input.fileinputformat.split.minsize", "134217728");

Job job = Job.*getInstance*(conf, "");

job.setJarByClass(RatingMovie.class);

job.setMapperClass(MapClass.class);

//job.setCombinerClass(ReduceClass.class);

job.setReducerClass(ReduceClass.class);

job.setNumReduceTasks(1);

job.setOutputKeyClass(LongWritable.class);

job.setOutputValueClass(DoubleWritable.class);

FileInputFormat.*addInputPath*(job, new Path(args[0]));

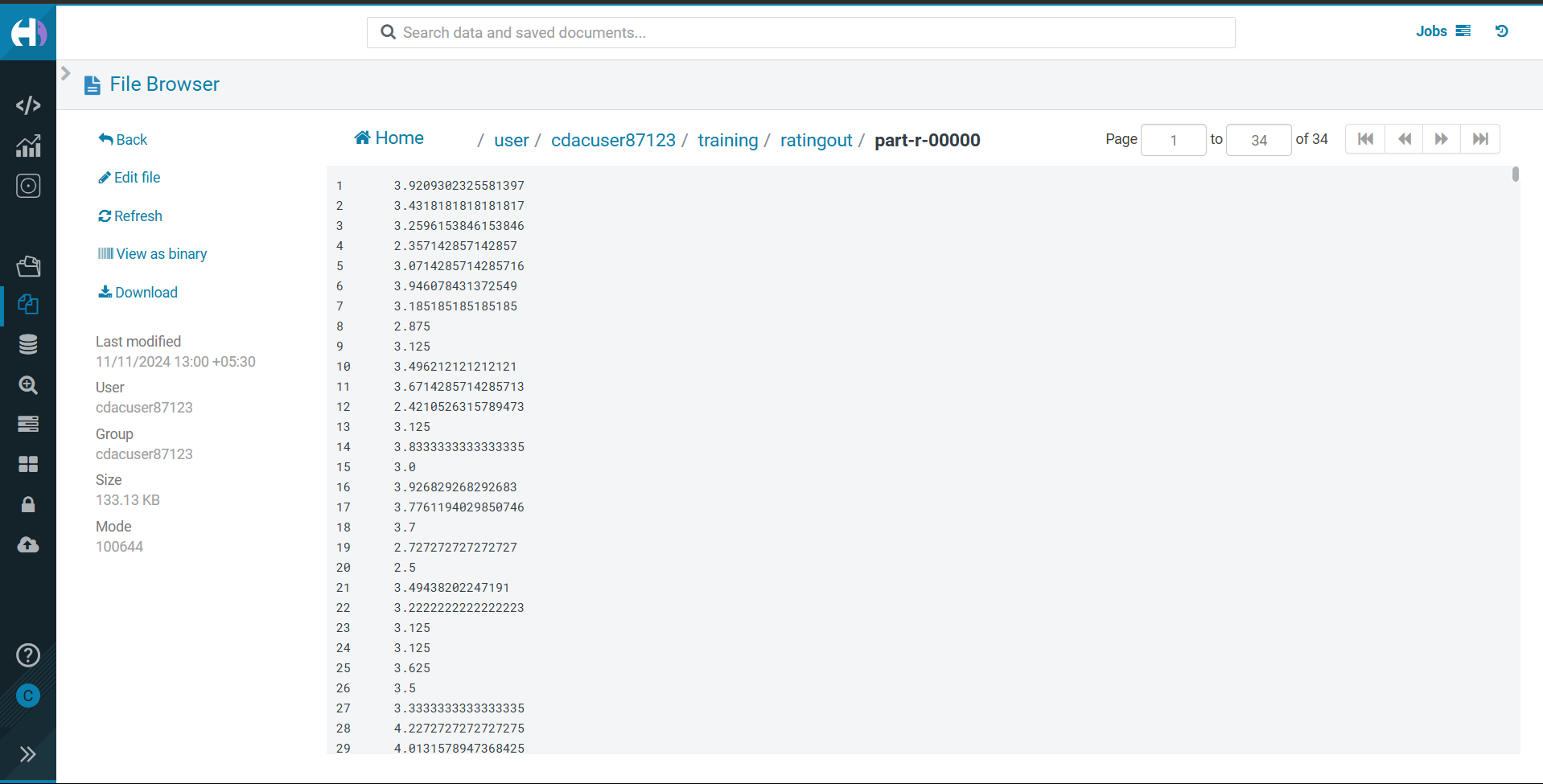
FileOutputFormat.*setOutputPath*(job, new Path(args[1]));

System.*exit*(job.waitForCompletion(true) ? 0 : 1);

}

}

Output



DATA SET: <https://de-mapreduce-gutenberg.s3.amazonaws.com/100-0.txt>

Write a Mapper and Reducer code to count the number of occurrences of each word in a text document

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount1 {

public static class TokenizerMapper

extends Mapper<LongWritable, Text, Text, IntWritable>{

private final static IntWritable ***one*** = new IntWritable(1);

private Text word = new Text();

public void map(LongWritable key, Text value, Context context

) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

String myword = itr.nextToken().toLowerCase();

word.set(myword);

context.write(word, ***one***);

//context.write(word, one);

}

}

}

public static class IntSumReducer

extends Reducer<Text,IntWritable,Text,IntWritable> {

private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,

Context context

) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.*getInstance*(conf, "word count");

job.setJarByClass(WordCount.class);

job.setMapperClass(TokenizerMapper.class);

job.setReducerClass(IntSumReducer.class);

job.setNumReduceTasks(1);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.*addInputPath*(job, new Path(args[0]));

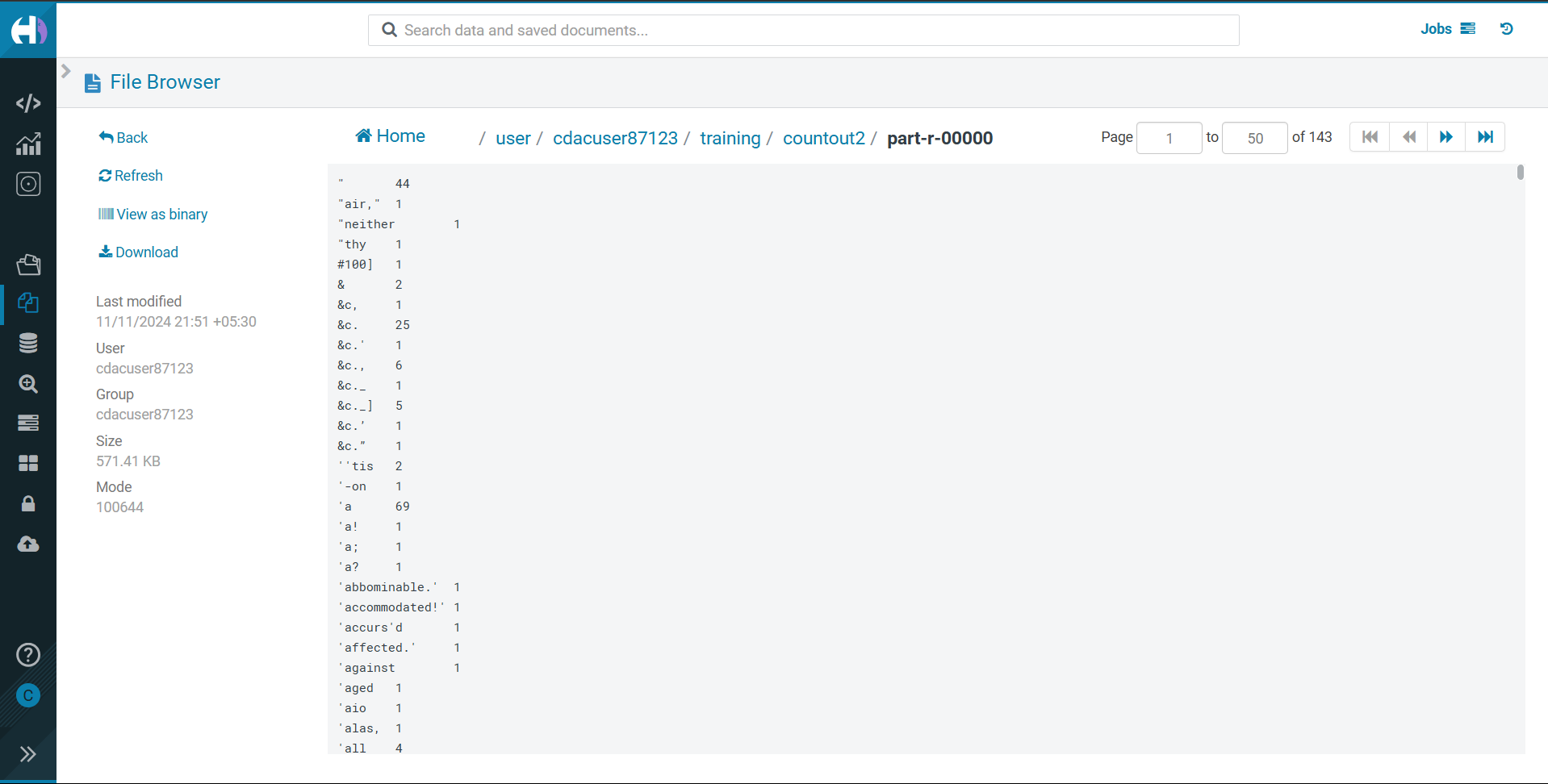
FileOutputFormat.*setOutputPath*(job, new Path(args[1]));

System.*exit*(job.waitForCompletion(true) ? 0 : 1);

}

}

OUTPUT



DATA SET: [SalesJan2009.csv](https://www.cs.ucy.ac.cy/courses/DSC511/data/SalesJan2009.csv)S

Write a Mapper and Reducer code to calculate the average sale amount per customer from a dataset containing customer transaction records.

import java.io.\*;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.fs.\*;

import org.apache.hadoop.mapreduce.lib.input.\*;

import org.apache.hadoop.mapreduce.lib.output.\*;

public class AvgSales {

public static class MapClass extends Mapper<LongWritable,Text,Text,DoubleWritable>

{

public void map(LongWritable key, Text value, Context context)

{

try{

String[] str = value.toString().split(",");

Double sales = Double.*parseDouble*(str[11]);

context.write(new Text(str[4]),new DoubleWritable(sales));

}

catch(Exception e)

{

System.***out***.println(e.getMessage());

}

}

}

public static class ReduceClass extends Reducer<Text,DoubleWritable,Text,DoubleWritable>

{

private DoubleWritable result = new DoubleWritable();

public void reduce(Text key, Iterable<DoubleWritable> values,Context context) throws IOException, InterruptedException {

double sum = 0;

double cnt=0;

double avg=0;

for (DoubleWritable val : values)

{

sum += val.get();

cnt++;

}

avg = sum/cnt;

avg = Math.*abs*(avg);

result.set(avg);

context.write(key, result);

//context.write(key, new LongWritable(sum));

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

//conf.set("name", "value")

//conf.set("mapreduce.input.fileinputformat.split.minsize", "134217728");

Job job = Job.*getInstance*(conf, "All time high price is");

job.setJarByClass(AvgSales.class);

job.setMapperClass(MapClass.class);

//job.setCombinerClass(ReduceClass.class);

job.setReducerClass(ReduceClass.class);

job.setNumReduceTasks(1);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(DoubleWritable.class);

FileInputFormat.*addInputPath*(job, new Path(args[0]));

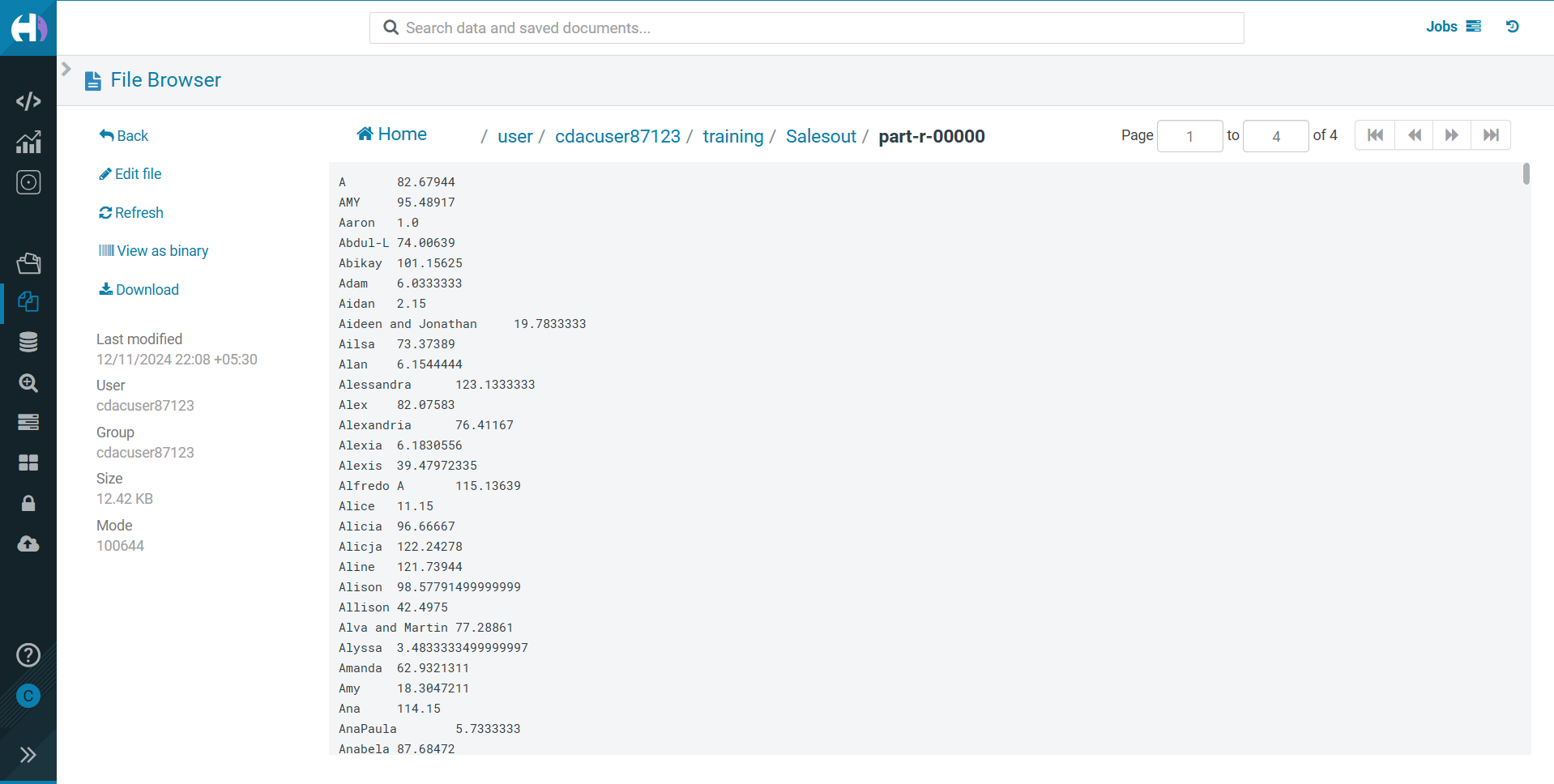
FileOutputFormat.*setOutputPath*(job, new Path(args[1]));

System.*exit*(job.waitForCompletion(true) ? 0 : 1);

}

}

OUTPUT



DATA SET: <https://de-mapreduce-gutenberg.s3.amazonaws.com/100-0.txt>

Write a Mapper and Reducer code to find the total number of occurrences of each alphabet in a text document (case-insensitive).

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordOccurence {

public static class AlphabetMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

private final static IntWritable ***one*** = new IntWritable(1);

private Text letter = new Text();

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

String line = value.toString().toLowerCase();

for (char c : line.toCharArray()) {

if (Character.*isLetter*(c)) {

letter.set(Character.*toString*(c));

context.write(letter, ***one***);

}

}

}

}

public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.*getInstance*(conf, "alphabet count");

job.setJarByClass(WordOccurence.class);

job.setMapperClass(AlphabetMapper.class);

job.setReducerClass(IntSumReducer.class);

job.setNumReduceTasks(1);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.*addInputPath*(job, new Path(args[0]));

FileOutputFormat.*setOutputPath*(job, new Path(args[1]));

System.*exit*(job.waitForCompletion(true) ? 0 : 1);

}

}

OUTPUT

