Presented By 4th Year MSSE Student - VIT Chennai Campus

DATASETS:

1. Recipe Items Latest- Json Format

TEAM DETAILS:

Coding

Prabakaran A 11MSE1108 [prabakaran.a2011@vit.ac.in](mailto:prabakaran.a2011@vit.ac.in) +91 8807574260

Visualisation & Documentation

Paranjothi M 11MSE1104 [paranjothi.m2011@vit.ac.in](mailto:paranjothi.m2011@vit.ac.in) +91 9789253459

Muthukumaran R 11MSE1090 muthukumaran.r2011@vit.ac.in +91 9791992669

**DATASET 1**

**NAME:** Recipe Items Latest- Json Format.

**TOOLS USED TO PROCESS DATA:**  Map Reduce using Java, Apache Hadoop 2.3 and Eclipse.

**SCHEMA:** ID, Name, Ingredients, URL, Image, Cook Time, Source, Recipe Yield, Date Published, Preparation Time, Description.

**Task 1: Preparation Time**

**Description:**

Process the Json recipe dataset to aggregate the total number of dishes with directly proportional to preparation time.

**Use cases:**

**public class prepTime { //DRIVER**

**public static class TokenizerMapper //MAPPER**

extends

Mapper<Object, Text, Text, IntWritable> {

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

private Text word = new Text();

Gson gson = new Gson();

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

throws IOException, InterruptedException {

Roo roo = gson.fromJson(value.toString(), Roo.class);

if (roo.prepTime != null) {

word.set(roo.prepTime);

} else {

word.set("none");

}

context.write(word, one);

}

}

**public static class IntSumReducer // REDUCER**

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();

String[] otherArgs = new GenericOptionsParser(conf, args)

.getRemainingArgs();

@SuppressWarnings("deprecation")

Job job = new Job(conf, "cookTime");

job.setJarByClass(prepTime.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path("hdfs://127.0.0.1:9000/in"));

FileOutputFormat.setOutputPath(job, new Path(

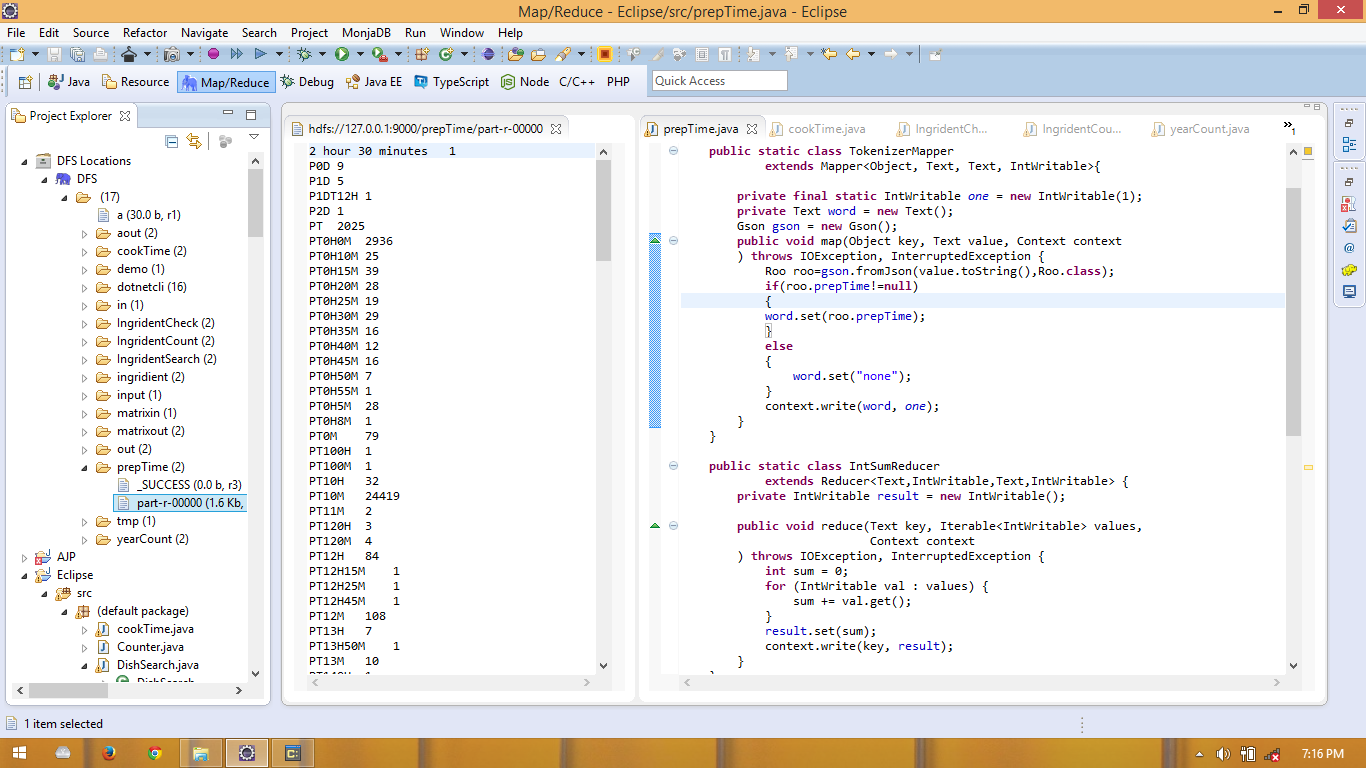
"hdfs://127.0.0.1:9000/prepTime"));

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

}

}

**Sample screen with output:**

****

**Visualization: Using Excel and Inkscape**

**Task 2: Cook Time**

**Description:**

Process the Json recipe dataset to aggregate the total number of dishes with directly proportional to COOK time.

**Use cases:**

**public class cookTime { // DRIVER**

**public static class TokenizerMapper // MAPPER**

extends

Mapper<Object, Text, Text, IntWritable> {

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

private Text word = new Text();

Gson gson = new Gson();

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

throws IOException, InterruptedException {

Roo roo = gson.fromJson(value.toString(), Roo.class);

if (roo.cookTime != null) {

word.set(roo.cookTime);

} else {

word.set("none");

}

context.write(word, one);

}

}

**public static class IntSumReducer //REDUCER**

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();

String[] otherArgs = new GenericOptionsParser(conf, args)

.getRemainingArgs();

@SuppressWarnings("deprecation")

Job job = new Job(conf, "Recipe");

job.setJarByClass(cookTime.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

// FileInputFormat.addInputPath(job, new Path(otherArgs[0]));

// FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

FileInputFormat.addInputPath(job, new Path("hdfs://127.0.0.1:9000/in"));

FileOutputFormat.setOutputPath(job, new Path(

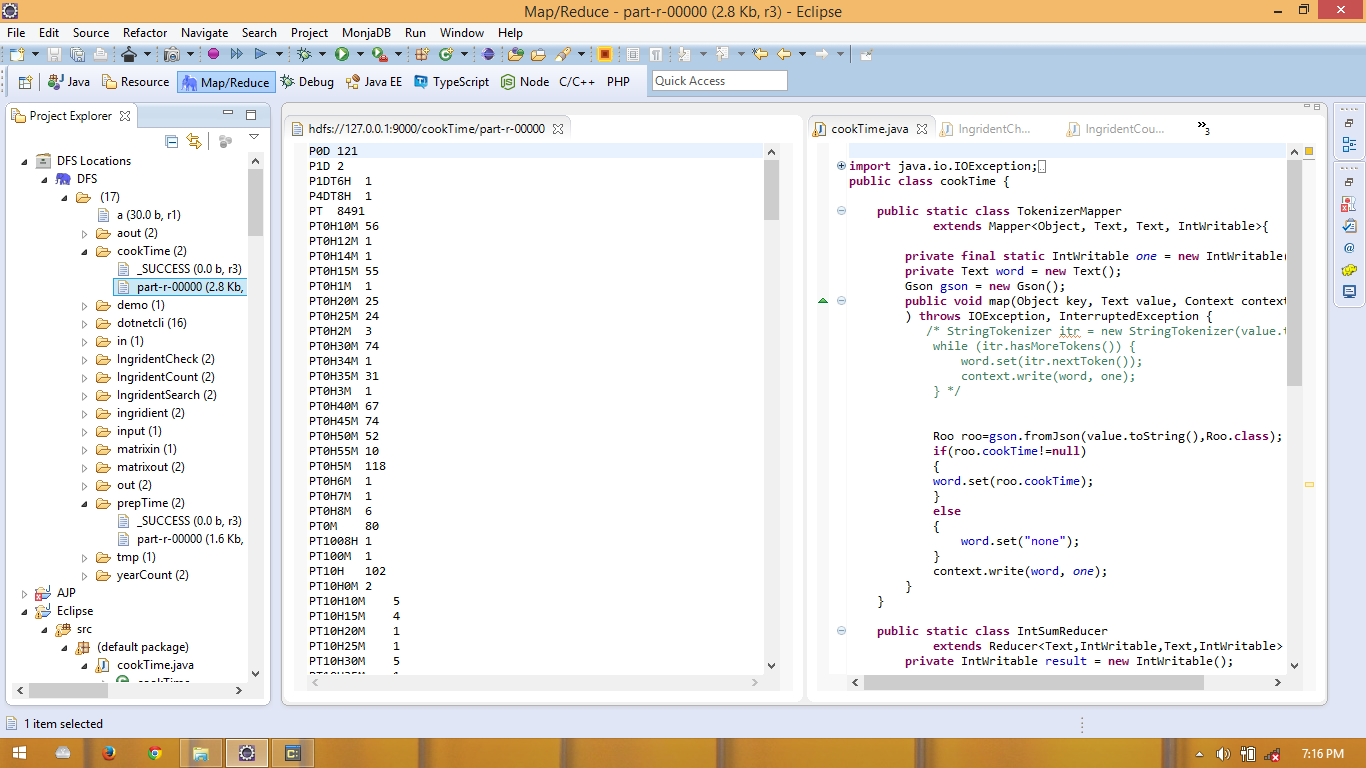
"hdfs://127.0.0.1:9000/cookTime"));

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

// job.submit();

}

**Sample screen with output:**



**Visualization:**

**Task 3: Year**

**Description:**

Process the Json recipe dataset to aggregate the total number of dishes with directly proportional to year of the dishes.

**Use cases:**

**public class yearCount { //DRIVER**

**public static class TokenizerMapper //MAPPER**

extends

Mapper<Object, Text, Text, Text> {

private final static Text one = new Text("1");

private Text word = new Text();

Gson gson = new Gson();

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

throws IOException, InterruptedException {

Roo roo = gson.fromJson(value.toString(), Roo.class);

try {

if (roo.datePublished != "") {

String date = roo.datePublished.substring(0, 4);

word.set(date + "");

context.write(word, one);

}

} catch (Exception e) {

// TODO: handle exception

}

**public static class IntSumReducer extends Reducer<Text, Text, Text, Text> { //REDUCER**

private Text result = new Text();

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

int sum = 0;

for (Text val : values) {

sum = sum + Integer.parseInt(val.toString());;

}

result.set(sum + "");

context.write(key, result);

}

}

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

Configuration conf = new Configuration();

String[] otherArgs = new GenericOptionsParser(conf, args)

.getRemainingArgs();

@SuppressWarnings("deprecation")

Job job = new Job(conf, "Recipe");

job.setJarByClass(yearCount.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

// FileInputFormat.addInputPath(job, new Path(otherArgs[0]));

// FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

FileInputFormat.addInputPath(job, new Path("hdfs://127.0.0.1:9000/in"));

FileOutputFormat.setOutputPath(job, new Path(

"hdfs://127.0.0.1:9000/yearCount"));

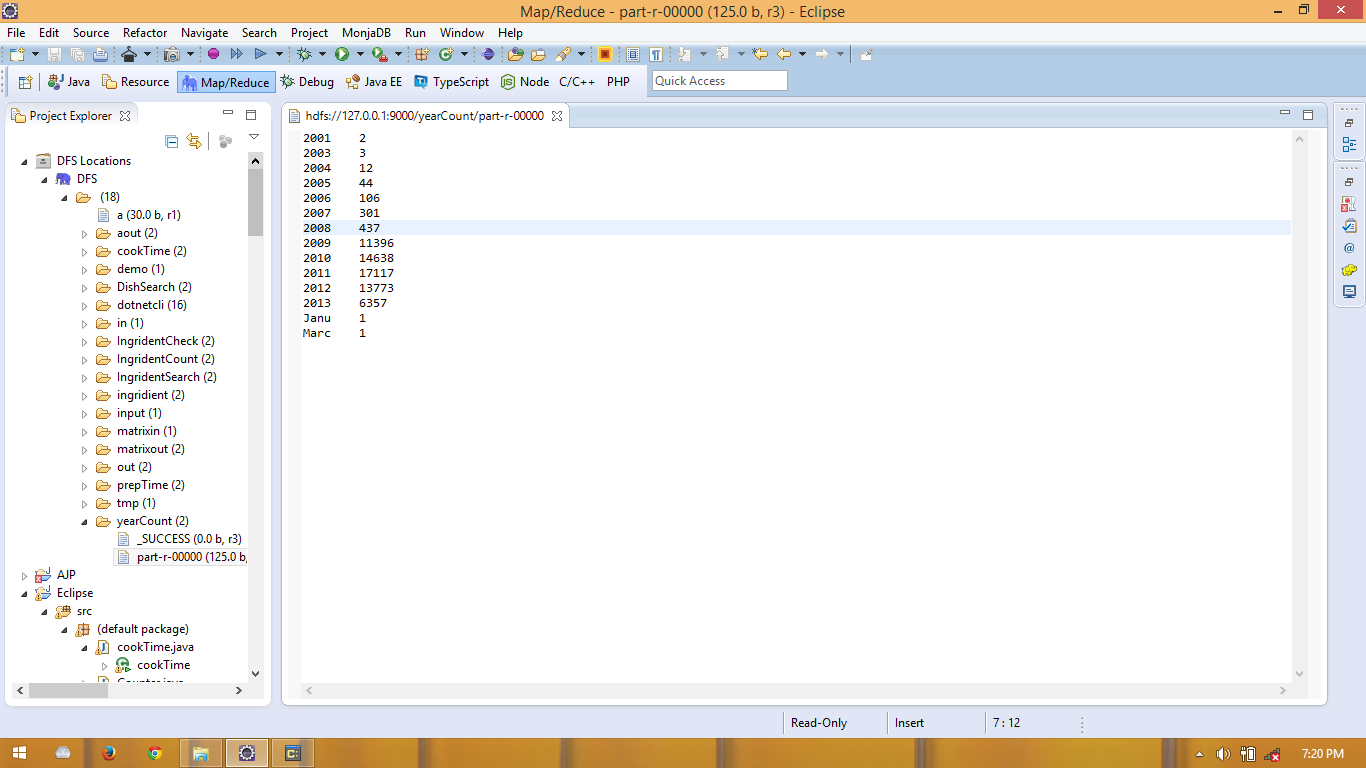
System.exit(job.waitForCompletion(true) ? 0 : 1);

// job.submit();

}

}

**Sample screen with output:**

****

**Visualization:**

**Task 4: Ingredients Count**

**Description:**

In this Dataset we counted the number of dishes with respect to the Preparation Time

**Use cases:**

**public class IngridentCount { //MAPPER**

**public static class TokenizerMapper //MAPPER**

extends

Mapper<Object, Text, Text, Text> {

private final static Text one = new Text(“1”);

private Text word = new Text();

Gson gson = new Gson();

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

throws IOException, InterruptedException {

Roo roo = gson.fromJson(value.toString(), Roo.class);

String[] ia = roo.ingredients.split(“\n”);

if (ia.length != 0) {

word.set(ia.length + “”);

} else {

word.set(“0”);

}

context.write(word, one);

}

}

**Public static class IntSumReducer extends Reducer<Text, Text, Text, Text> {//REDUCER**

private Text result = new Text();

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

int sum = 0;

for (Text val : values) {

sum = sum + Integer.parseInt(val.toString());;

}

result.set(sum + "");

context.write(key, result);

}

}

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

Configuration conf = new Configuration();

String[] otherArgs = new GenericOptionsParser(conf, args)

.getRemainingArgs();

@SuppressWarnings("deprecation")

Job job = new Job(conf, "Recipe");

job.setJarByClass(IngridentCount.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

// FileInputFormat.addInputPath(job, new Path(otherArgs[0]));

// FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

FileInputFormat.addInputPath(job, new Path("hdfs://127.0.0.1:9000/in"));

FileOutputFormat.setOutputPath(job, new Path(

"hdfs://127.0.0.1:9000/IngridentCount"));

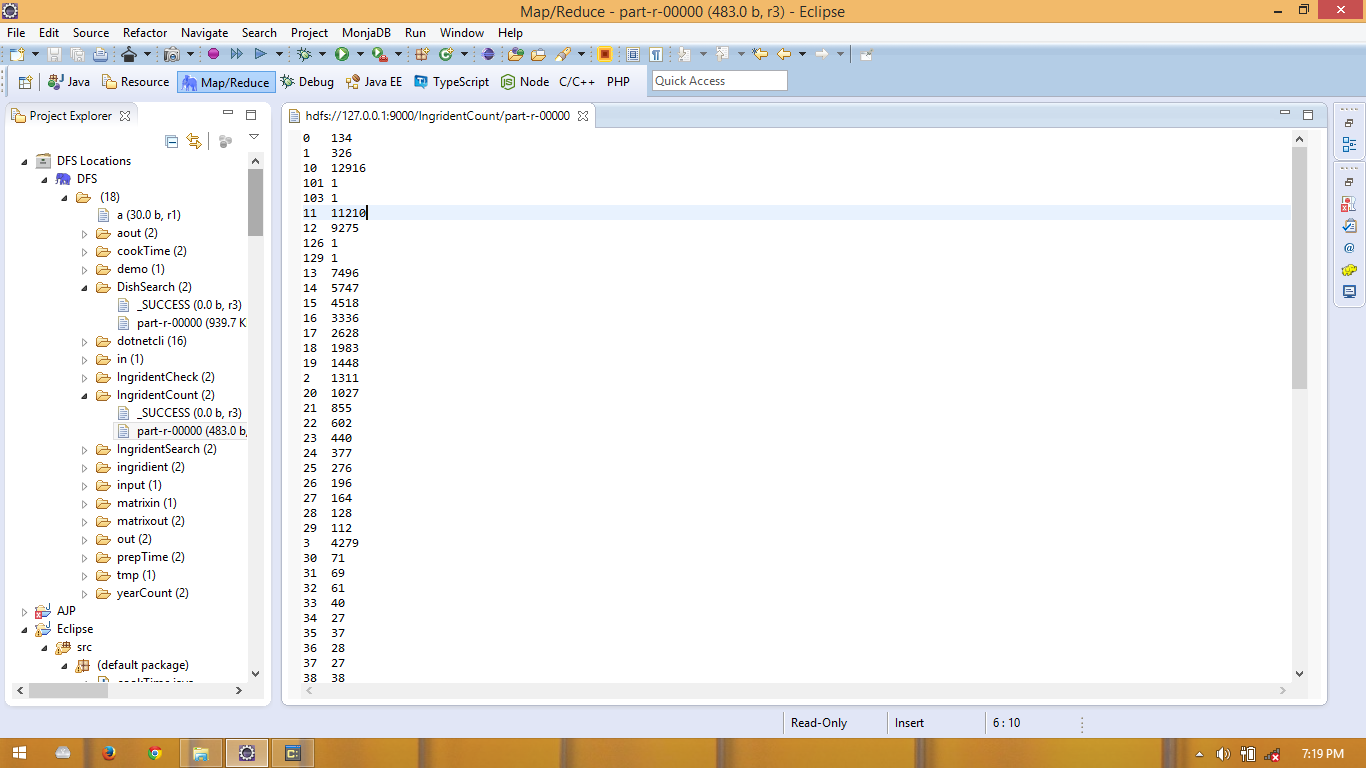
System.exit(job.waitForCompletion(true) ? 0 : 1);

// job.submit();

}

}

**Sample screen with output:**

****

**Visualization:**

**Task 5: Ingredient Check**

**Description:**

Can you please tell me the maximum number of ingredients to make a dish?

**Use cases:**

**public class IngridentCheck { //DRIVER**

**public static class TokenizerMapper //MAPPER**

extends

Mapper<Object, Text, Text, Text> {

private final static Text one = new Text("1");

private Text word = new Text("none");

Gson gson = new Gson();

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

throws IOException, InterruptedException {

Roo roo = gson.fromJson(value.toString(), Roo.class);

String p = roo.ingredients.replace("\n", ";");

String[] ia = roo.ingredients.split("\n");

if (ia.length > 100) {

one.set(p + "");

word.set(roo.name + " " + ia.length + " ");

context.write(word, one);

}

}

}

**public static class IntSumReducer extends Reducer<Text, Text, Text, Text> { //REDUCER**

private Text result = new Text();

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

StringBuilder string = new StringBuilder();

for (Text text : values) {

string.append(text);

}

context.write(key, new Text(string.toString()));

}

}

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

Configuration conf = new Configuration();

String[] otherArgs = new GenericOptionsParser(conf, args)

.getRemainingArgs();

@SuppressWarnings("deprecation")

Job job = new Job(conf, "Recipe");

job.setJarByClass(IngridentCheck.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

// FileInputFormat.addInputPath(job, new Path(otherArgs[0]));

// FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

FileInputFormat.addInputPath(job, new Path("hdfs://127.0.0.1:9000/in"));

FileOutputFormat.setOutputPath(job, new Path(

"hdfs://127.0.0.1:9000/IngridentCheck"));

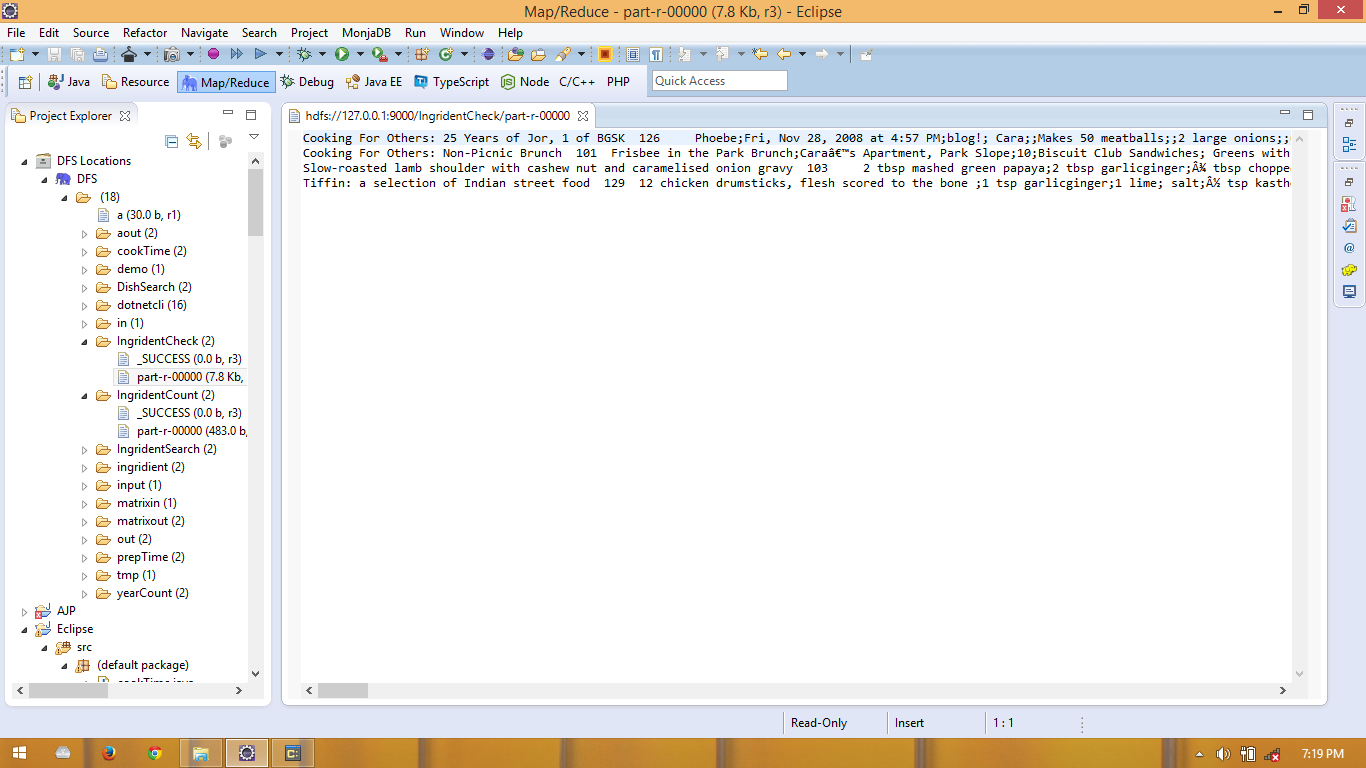
System.exit(job.waitForCompletion(true) ? 0 : 1);

// job.submit();

}

}

**Sample screen with output:**

****

**Task 6: Ingredient Search**

**Description:**

With basic ingredients like tomatto and onion and also say number of ingredients say 5 to make a dish.

**Use Case:**

**public class IngridientSearch { //DRIVER**

**public static class TokenizerMapper //MAPPER**

extends

Mapper<Object, Text, Text, Text> {

private final static Text one = new Text("1");

private Text word = new Text("none");

Gson gson = new Gson();

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

throws IOException, InterruptedException {

Roo roo = gson.fromJson(value.toString(), Roo.class);

String p = roo.ingredients.replace("\n", ";");

String[] ia = roo.ingredients.split("\n");

if (ia.length == 5) {

if (roo.ingredients.contains("mango")) {

one.set(p);

word.set(roo.name);

context.write(word, one);

}

}

}

}

**public static class IntSumReducer extends Reducer<Text, Text, Text, Text> {//REDUCER**

private Text result = new Text();

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

StringBuilder s = new StringBuilder();

for (Text text : values) {

s.append(text.toString());

}

context.write(key, new Text(s.toString()));

}

}

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

Configuration conf = new Configuration();

String[] otherArgs = new GenericOptionsParser(conf, args)

.getRemainingArgs();

@SuppressWarnings("deprecation")

Job job = new Job(conf, "IngridientSearch");

job.setJarByClass(IngridientSearch.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

// FileInputFormat.addInputPath(job, new Path(otherArgs[0]));

// FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

FileInputFormat.addInputPath(job, new Path("hdfs://127.0.0.1:9000/in"));

FileOutputFormat.setOutputPath(job, new Path(

"hdfs://127.0.0.1:9000/IngridentSearch"));

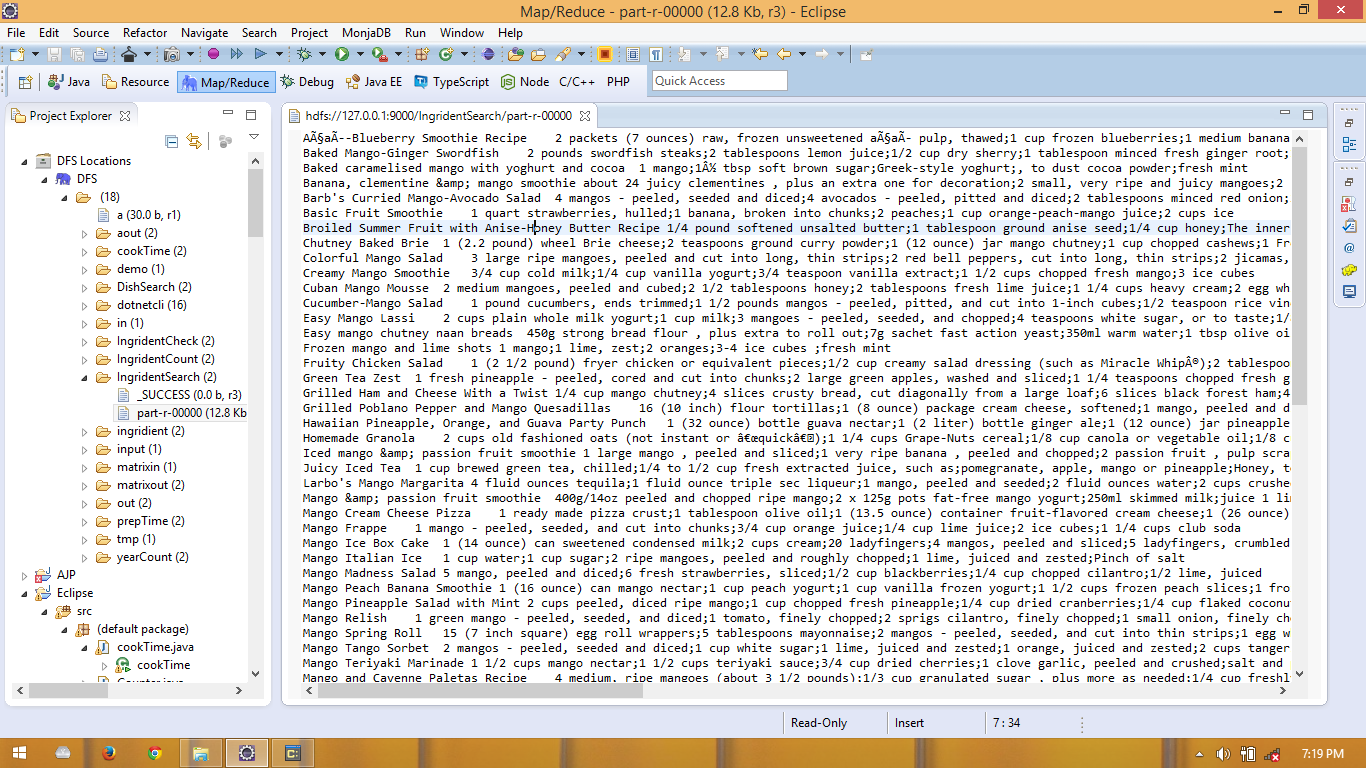
System.exit(job.waitForCompletion(true) ? 0 : 1);

// job.submit();

}

}

**Sample Screenshot with output:**

****

**Task 7: Dish Search**

**Description:**

With your ingredients how many dishes you can make?

**Use Case:**

**public class DishSearch { //DRIER**

public String item;

**public static class TokenizerMapper //MAPPER**

extends

Mapper<Object, Text, Text, Text> {

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

private Text word = new Text();

Gson gson = new Gson();

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

throws IOException, InterruptedException {

Roo roo = gson.fromJson(value.toString(), Roo.class);

try {

if (roo.ingredients.contains("tomato")

&& roo.ingredients.contains("onion")

&& roo.ingredients.contains("salt")) {

word.set(roo.name + " " + roo.description);

} else {

word.set("none");

}

} catch (Exception e) {

word.set("none");

}

context.write(new Text("1"), word);

}

}

**public static class IntSumReducer extends Reducer<Text, Text, Text, Text> {//REDUCER**

private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

int sum = 0;

for (Text val : values) {

if (!val.toString().matches("none")) {

context.write(new Text(), val);

}

}

}

}

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

Configuration conf = new Configuration();

String[] otherArgs = new GenericOptionsParser(conf, args)

.getRemainingArgs();

@SuppressWarnings("deprecation")

Job job = new Job(conf, "cookTime");

job.setJarByClass(DishSearch.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

// FileInputFormat.addInputPath(job, new Path(otherArgs[0]));

// FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

FileInputFormat.addInputPath(job, new Path("hdfs://127.0.0.1:9000/in"));

FileOutputFormat.setOutputPath(job, new Path(

"hdfs://127.0.0.1:9000/DishSearch"));

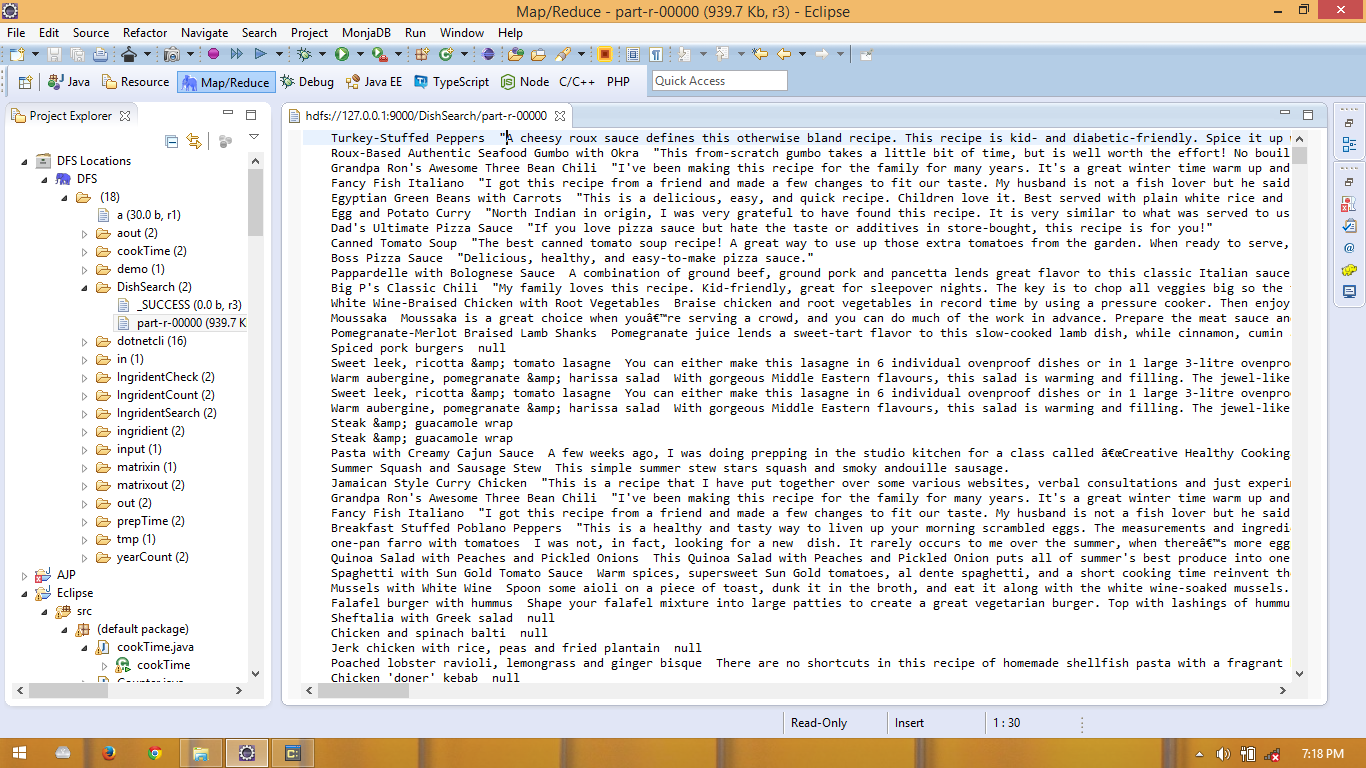
System.exit(job.waitForCompletion(true) ? 0 : 1);

// job.submit();

}

}

**Sample Screenshot with output:**

****