INDEX

Map Reduce Assignment

1.Write a Program to find Average Word Length

Code:

```
import java.io.IOException;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
importorg.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;

public class word_avg extends Mapper <LongWritable,Text,Text,IntWritable> {


@Override
public void map(LongWritable key,Text value,Context context) throws
IOException,InterruptedException{
String line=value.toString();
//line.split("\\W+");
for(String word:line.split("\\W+")){
if(word.length()>0){
String firstChar = String.valueOf(word.charAt(0));
context.write(new Text(firstChar),new IntWritable(word.length()));

                    }
                }
            }

public static class sumReducer extends Reducer<Text,IntWritable,Text,FloatWritable>{
@Override
public void reduce(Text key,Iterable<IntWritable> values,Context context) throws
IOException,InterruptedException{
int count=0;
int sum = 0;
for(IntWritable value:values){
count+=1;
sum += value.get();
```

```
}
float ans = sum/count;
context.write(key, new FloatWritable(ans));
}
}


public static void main(String args[])throws Exception{
if(args.length!=2){
System.out.printf("Usage:WordCount<input dir> <output dir>\n");
System.exit(-1);
}
Job job=new Job();
/* specify the jar file that contains your driver , mapper,reducer.
                hadoop will transfer this jar file to nodes in your cluster running mapper and
                reducer tasks*/

job.setJarByClass(word_avg.class);

                /* Specify the paths to the input and output data based on the
                command-line arguments*/
job.setJobName("Word Count");
FileInputFormat.setInputPaths(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));

/* Specify the mapper and reducer classes */
job.setMapperClass(word_avg.class);
job.setReducerClass(sumReducer.class);

job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);

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



}

}
```

Output:

```
[training@localhost ~]$ hadoop jar /home/training/avg.jar /user/training/prashant/input /user/training/prashant/op2
24/08/01 15:44:37 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
24/08/01 15:44:37 INFO input.FileInputFormat: Total input paths to process : 1
24/08/01 15:44:37 WARN snappy.LoadSnappy: Snappy native library is available
24/08/01 15:44:37 INFO snappy.LoadSnappy: Snappy native library loaded
24/08/01 15:44:37 INFO mapred.JobClient: Running job: job_202407291425_0007
24/08/01 15:44:38 INFO mapred.JobClient:  map 0% reduce 0%
24/08/01 15:44:40 INFO mapred.JobClient:  map 100% reduce 0%
24/08/01 15:44:42 INFO mapred.JobClient: Job complete: job_202407291425_0007
24/08/01 15:44:42 INFO mapred.JobClient: Counters: 32
24/08/01 15:44:42 INFO mapred.JobClient:   File System Counters
24/08/01 15:44:42 INFO mapred.JobClient:     FILE: Number of bytes read=62
24/08/01 15:44:42 INFO mapred.JobClient:     FILE: Number of bytes written=361864
24/08/01 15:44:42 INFO mapred.JobClient:     FILE: Number of read operations=0
24/08/01 15:44:42 INFO mapred.JobClient:     FILE: Number of large read operations=0
24/08/01 15:44:42 INFO mapred.JobClient:     FILE: Number of write operations=0
24/08/01 15:44:42 INFO mapred.JobClient:     HDFS: Number of bytes read=153
24/08/01 15:44:42 INFO mapred.JobClient:     HDFS: Number of bytes written=30
24/08/01 15:44:42 INFO mapred.JobClient:     HDFS: Number of read operations=2
24/08/01 15:44:42 INFO mapred.JobClient:     HDFS: Number of large read operations=0
```

File Edit View History Bookmarks Tools Help

HDFS:/user/training/prashant... | 0 Hadoop Map/Reduce Admi...

localhost.localdomain:50075/browseBlock.jsp?blockId=-154796475

Most Visited | Hadoop NameNode | Hadoop JobTracker | Hadoop

File: /user/training/prashant/op2/part-r-00000

Goto : /user/training/prashant/op2 go

Go back to dir listing

Advanced view/download options

```
B       2.0
C       7.0
D       4.0
P       8.0
T       4.0
```

2.Write a Program to Fetch the Customer Data using Map Reduce

Name: Prashant Vithal Godhe
Rollno : 13
Date : 02-08-24

To find the Profession Count:

Code:

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
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;
import org.apache.hadoop.io.LongWritable;

import java.io.IOException;

public class cust_count {

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

    @Override
    public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {
      String line = value.toString();
      String[] fields = line.split(",");

      if (fields.length > 4) {
        String profession = fields[4].trim();
        context.write(new Text(profession), new IntWritable(1));
      }
    }
  }

  public static class SumReducer extends Reducer<Text, IntWritable,
    Text, IntWritable> { @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context
```

```
context) throws IOException, InterruptedException {
            int professionCount = 0;
            for (IntWritable value : values) {
                professionCount += value.get();
            }
            context.write(key, new IntWritable(professionCount));
        }
    }

    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.out.printf("Usage: ProfessionCount <input dir>
            <output dir>\n"); System.exit(-1);
        }
        Job job = Job.getInstance();
        job.setJarByClass(cust_count.class);
        job.setJobName("Profession Count");

        FileInputFormat.setInputPaths(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.setMapperClass(ProfessionMapper.class);
        job.setReducerClass(SumReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        boolean success = job.waitForCompletion(true);
        System.exit(success ? 0 : 1);
    }
}
```

Output:

```
[training@localhost ~]$  hadoop jar /home/training/pro_count.jar /user/training/
prashant/custs /user/training/prashant/cust_count
24/08/02 10:32:24 WARN mapred.JobClient: Use GenericOptionsParser for parsing th
e arguments. Applications should implement Tool for the same.
24/08/02 10:32:24 INFO input.FileInputFormat: Total input paths to process : 1
24/08/02 10:32:24 WARN snappy.LoadSnappy: Snappy native library is available
24/08/02 10:32:24 INFO snappy.LoadSnappy: Snappy native library loaded
24/08/02 10:32:24 INFO mapred.JobClient: Running job: job_202408020850_0003
24/08/02 10:32:25 INFO mapred.JobClient:  map 0% reduce 0%
24/08/02 10:32:28 INFO mapred.JobClient:  map 100% reduce 0%
24/08/02 10:32:30 INFO mapred.JobClient:  map 100% reduce 100%
24/08/02 10:32:30 INFO mapred.JobClient: Job complete: job_202408020850_0003
24/08/02 10:32:30 INFO mapred.JobClient: Counters: 32
24/08/02 10:32:30 INFO mapred.JobClient:   File System Counters
24/08/02 10:32:30 INFO mapred.JobClient:     FILE: Number of bytes read=193820
24/08/02 10:32:30 INFO mapred.JobClient:     FILE: Number of bytes written=74944
8
24/08/02 10:32:30 INFO mapred.JobClient:     FILE: Number of read operations=0
24/08/02 10:32:30 INFO mapred.JobClient:     FILE: Number of large read operatio
ns=0
24/08/02 10:32:30 INFO mapred.JobClient:     FILE: Number of write operations=0
24/08/02 10:32:30 INFO mapred.JobClient:     HDFS: Number of bytes read=391468
24/08/02 10:32:30 INFO mapred.JobClient:     HDFS: Number of bytes written=868
24/08/02 10:32:30 INFO mapred.JobClient:     HDFS: Number of read operations=2
24/08/02 10:32:30 INFO mapred.JobClient:     HDFS: Number of large read operatio
ns=0
24/08/02 10:32:30 INFO mapred.JobClient:     HDFS: Number of write operations=1
24/08/02 10:32:30 INFO mapred.JobClient:   Job Counters
24/08/02 10:32:30 INFO mapred.JobClient:     Launched map tasks=1
24/08/02 10:32:30 INFO mapred.JobClient:     Launched reduce tasks=1
```

File: /user/training/prashant/cust_count/part-r-00000

Goto : /user/training/prashant/cust go

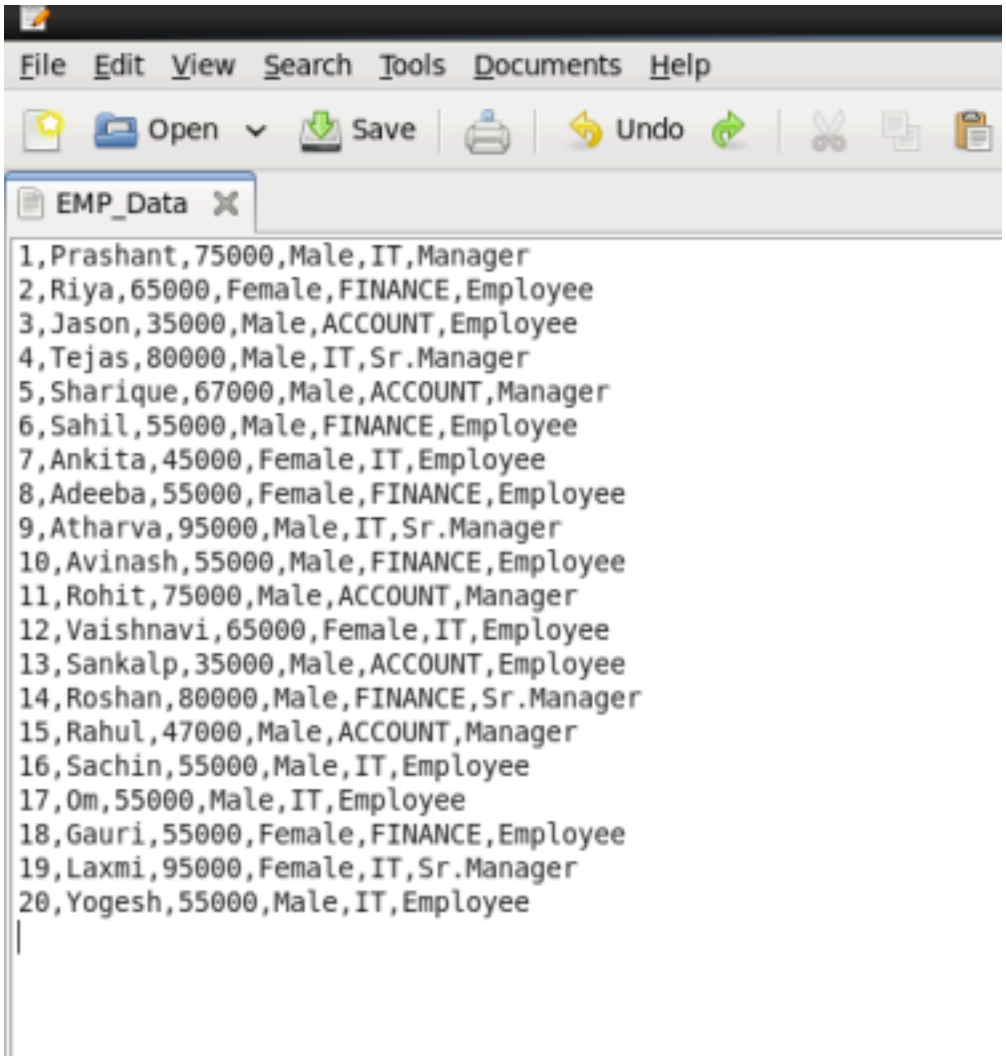
Go back to dir listing
Advanced view/download options

```
Accountant      197
Actor   196
Agricultural and food scientist 195
Architect       202
Artist  175
Athlete 196
Automotive mechanic     193
Carpenter       180
Chemist 206
Childcare worker        207
Civil engineer  193
Coach   199
Computer hardware engineer      204
Computer software engineer      216
Computer support specialist     222
Dancer  178
Designer        204
Doctor  189
Economist       189
Electrical engineer     192
Electrician     194
Engineering technician  204
Environmental scientist 176
Farmer  196
Financial analyst       198
Firefighter     217
```

3.Program to Understand MapperReduce Design Pattern.

Name: Prashant Vithal Godhe
Rollno : 13
Date : 05-08-24

To find the Min Salary based on Department:



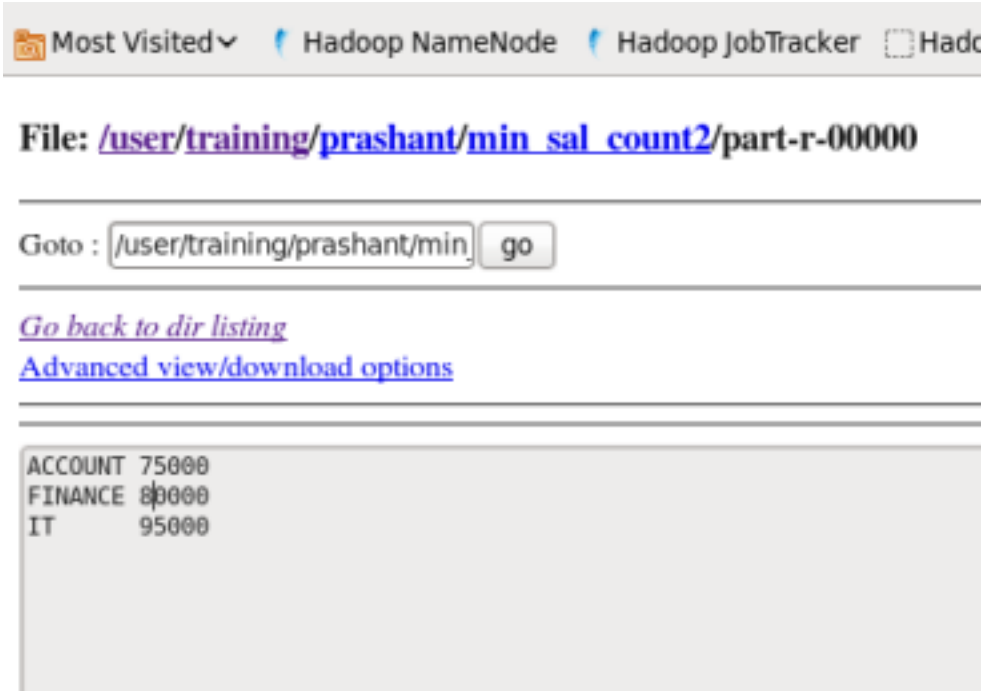
Code:

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
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;
import org.apache.hadoop.io.LongWritable;

import java.io.IOException;

public class Salary_c_dept {

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

        @Override
        public void map(LongWritable key, Text value, Context context)
                throws IOException, InterruptedException {
            String line = value.toString();
            String[] fields = line.split(",");
            if (fields.length > 5) {
                String Dept = fields[4].trim();
                int Salary = Integer.parseInt(fields[2].trim());
                context.write(new Text(Dept), new IntWritable(Salary));
            }
        }
    }

    public static class SumReducer extends Reducer<Text, IntWritable, Text, IntWritable> { @Override
        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {
            int minSalary = 0;
            for (IntWritable value : values) {
                    if(value.get() > minSalary){
                            minSalary = value.get();
                    }
                    else{
                            minSalary = minSalary;
                    }
            }
            context.write(key, new IntWritable(minSalary));
        }
    }

    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.out.printf("Usage: DepartmentMinSalary <input dir> <output dir>\n");
            System.exit(-1);
        }
        Job job = Job.getInstance();
```

```
        job.setJarByClass(Salary_c_dept.class);
        job.setJobName("Salary MIN");

        FileInputFormat.setInputPaths(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.setMapperClass(DeptMapper.class);
        job.setReducerClass(SumReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        boolean success = job.waitForCompletion(true);
        System.exit(success ? 0 : 1);
    }
}
```

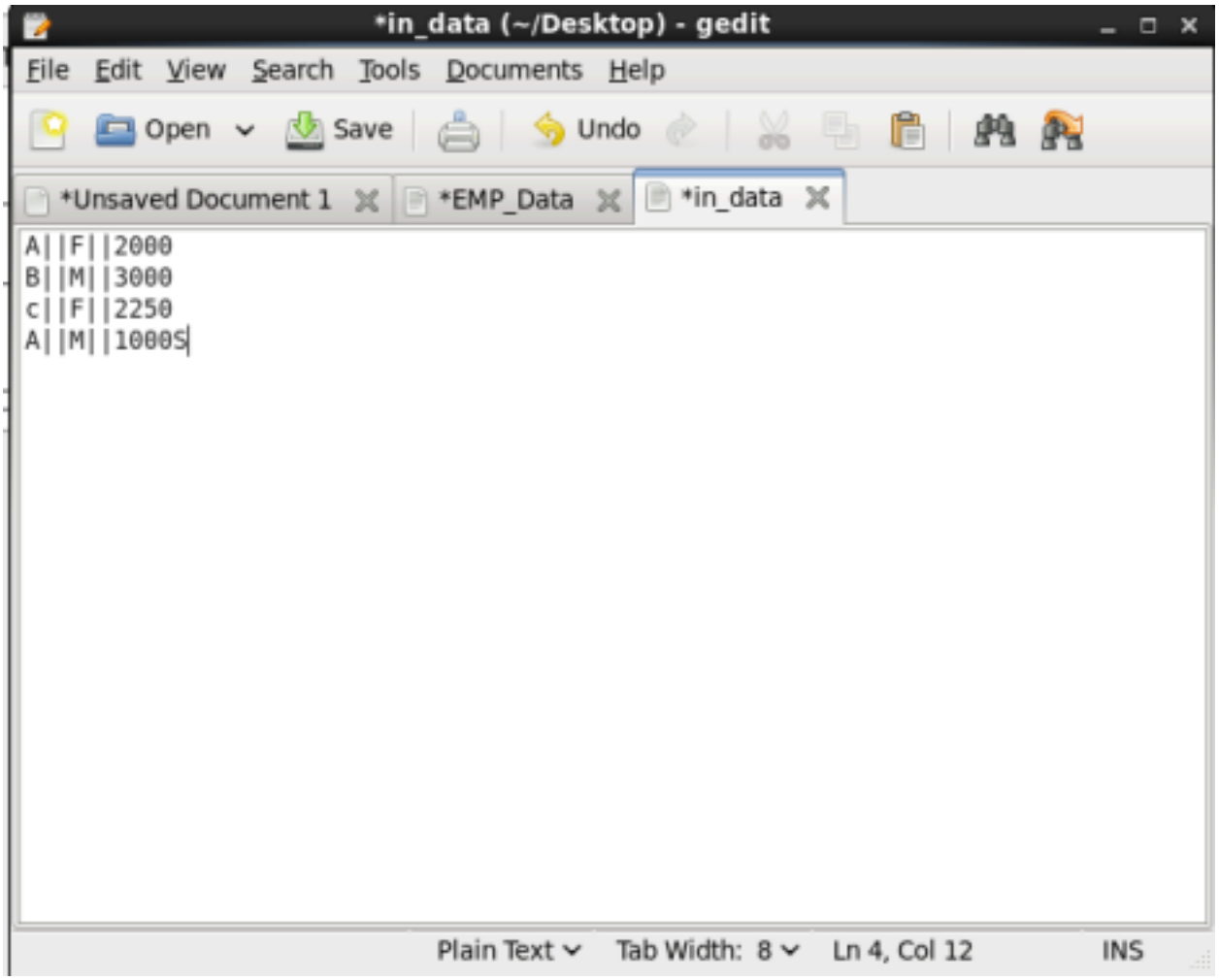
Output:



4.Program to Understand Only Mapper Design Pattern in Map Reduce

13 Prashant Godhe

Input file:



Code:

```
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class pattern {
```

```
    public static class patternMapper extends Mapper<Object, Text, Text, Text> {

        @Override

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

            String line = value.toString();
            String line1 = line.replace("||", ",");
            Text outputkey = new Text(line1);
            Text outputvalue = new Text("");
            context.write(outputkey, outputvalue);
        }
    }


    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: Separators <input path> <output path>");
            System.exit(-1);
        }

        Job job = new Job();
        job.setJarByClass(pattern.class);
        job.setJobName("Separators");

        FileInputFormat.setInputPaths(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.setMapperClass(patternMapper.class);
        job.setNumReduceTasks(0);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);

        boolean success = job.waitForCompletion(true);
        System.exit(success ? 0 : 1);
    }
}
```

Output:

training@localhost:~

File Edit View Search Terminal Help

```
[training@localhost ~]$ hdfs dfs -copyFromLocal /home/training/Desktop/in_data /user/training/prashant
[training@localhost ~]$  hadoop jar /home/training/pattern.jar /user/training/prashant/in_data /user/training/
prashant/opop
24/08/08 11:33:18 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications shou
ld implement Tool for the same.
24/08/08 11:33:18 INFO input.FileInputFormat: Total input paths to process : 1
24/08/08 11:33:18 WARN snappy.LoadSnappy: Snappy native library is available
24/08/08 11:33:18 INFO snappy.LoadSnappy: Snappy native library loaded
24/08/08 11:33:18 INFO mapred.JobClient: Running job: job_202408020850_0011
24/08/08 11:33:19 INFO mapred.JobClient:  map 0% reduce 0%
24/08/08 11:33:23 INFO mapred.JobClient:  map 100% reduce 0%
24/08/08 11:33:23 INFO mapred.JobClient: Job complete: job_202408020850_0011
24/08/08 11:33:23 INFO mapred.JobClient: Counters: 24
24/08/08 11:33:23 INFO mapred.JobClient:   File System Counters
24/08/08 11:33:23 INFO mapred.JobClient:     FILE: Number of bytes read=0
24/08/08 11:33:23 INFO mapred.JobClient:     FILE: Number of bytes written=180840
24/08/08 11:33:23 INFO mapred.JobClient:     FILE: Number of read operations=0
24/08/08 11:33:23 INFO mapred.JobClient:     FILE: Number of large read operations=0
24/08/08 11:33:23 INFO mapred.JobClient:     FILE: Number of write operations=0
24/08/08 11:33:23 INFO mapred.JobClient:     HDFS: Number of bytes read=159
24/08/08 11:33:23 INFO mapred.JobClient:     HDFS: Number of bytes written=40
24/08/08 11:33:23 INFO mapred.JobClient:     HDFS: Number of read operations=2
24/08/08 11:33:23 INFO mapred.JobClient:     HDFS: Number of large read operations=0
24/08/08 11:33:23 INFO mapred.JobClient:     HDFS: Number of write operations=1
```

File: /user/training/prashant/opop/part-m-00000

Goto : /user/training/prashant/opo go

Go back to dir listing

Advanced view/download options

```
A,F,2000
B,M,3000
c,F,2250
A,M,1000
```

Running jobs

none

Completed jobs

jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	job Scheduling Information	Diagnostic Info
job_202408020850_0011	NORMAL	training	Separators	100.00%	1	1	100.00%	0	0	NA	NA

5.Program to use Multiple Input files.

13 Prashant Vithal Godhe

Big Data MultipleInput_File

file1

```
8/1/2024,user1,login,120,10
8/1/2024,user2,checkout,30,50
8/1/2024,user1,purchase,45,100
```

file2

```
8/2/2024,user3,login,20,60
8/2/2024,user2,purchase,150,120
8/2/2024,user1,logout,5,30
```

file3

```
3/2024,user2,15,login,10
8/3/2024,user1,90,purchase,200
8/3/2024,user3,20,logout,15
```

file4

```
user2,login,10,15,8/4/2024
user1,purchase,200,90,8/4/2024
user4,logout,15,20,8/4/2024
```

Commands:

```
hdfs dfs -copyFromLocal /home/training/Desktop/data1
/user/training/prashant

hdfs dfs -copyFromLocal /home/training/Desktop/data2
/user/training/prashant

hdfs dfs -copyFromLocal /home/training/Desktop/data3
/user/training/prashant

hdfs dfs -copyFromLocal /home/training/Desktop/data4
/user/training/prashant

hadoop jar /home/training/multimapper.jar /user/training/prashant/data1
```

/user/training/prashant/data2 /user/training/prashant/data3
/user/training/prashant/data4 /user/training/prashant/multimapperop

Code:

```
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.util.GenericOptionsParser;
import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;

public class multipleinputsfilesNew {

	public static class Mapper1 extends Mapper<LongWritable, Text, Text,
	Text> { @Override
	public void map(LongWritable key, Text value, Context context)
		throws IOException, InterruptedException {
	String line = value.toString();
	String[] words = line.split(",");
	if (words.length > 0) {
		String user_id = words[1];
		String duration = words[3];
		String amount = words[4];
		String values = duration + "|" + amount;
		context.write(new Text(user_id), new Text(values));
	}
	}
	}

	public static class Mapper2 extends Mapper<LongWritable, Text, Text,
	Text> { @Override
	public void map(LongWritable key, Text value, Context context)
		throws IOException, InterruptedException {
	String line = value.toString();
	String[] words = line.split(",");
	if (words.length > 0) {
		String user_id = words[1];
```

```
        String duration = words[3];
        String amount = words[4];
        String values = duration + "|" + amount;
        context.write(new Text(user_id), new Text(values));
}
}
}

public static class Mapper3 extends Mapper<LongWritable, Text, Text,
Text> { @Override
 public void map(LongWritable key, Text value, Context
   context) throws IOException, InterruptedException {
String line = value.toString();
String[] words = line.split(",");
if (words.length > 0) {
        String user_id = words[1];
        String duration = words[2];
        String amount = words[4];
        String values = duration + "|" + amount;
        context.write(new Text(user_id), new Text(values));
}
}
}

public static class Mapper4 extends Mapper<LongWritable, Text, Text,
Text> { @Override
 public void map(LongWritable key, Text value, Context
   context) throws IOException, InterruptedException {
String line = value.toString();
String[] words = line.split(",");
if (words.length > 0) {
        String user_id = words[0];
        String duration = words[2];
        String amount = words[3];
        String values = duration + "|" + amount;
        context.write(new Text(user_id), new Text(values));
}
}
}

public static class Red extends Reducer<Text, Text, Text, Text> {
@Override
public void reduce(Text key, Iterable<Text> values, Context context)
        throws IOException, InterruptedException {
int maxDuration = 0;
```

```
int minDuration = Integer.MAX_VALUE;
int maxAmount = 0;
int minAmount = Integer.MAX_VALUE;

for (Text value : values) {
        String[] parts = value.toString().split("\\|");
        if (parts.length == 2) {
        int duration = Integer.parseInt(parts[0]);
        int amount = Integer.parseInt(parts[1]);

        if (duration > maxDuration) maxDuration = duration;
        if (duration < minDuration) minDuration = duration;
        if (amount > maxAmount) maxAmount = amount;
        if (amount < minAmount) minAmount = amount;
        }
}

String result = String.format("%d\t%d\t%d\t%d", maxDuration, minDuration,
maxAmount, minAmount);
context.write(key, new Text(result));
}
}

public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
GenericOptionsParser parser = new GenericOptionsParser(conf, args);

String[] files = parser.getRemainingArgs();
if (files.length != 5) {
System.err.println("Usage: MultipleInputFiles <input1> <input2> <input3>
<input4> <output>");
System.exit(2);
}

Path p1 = new Path(files[0]);
Path p2 = new Path(files[1]);
Path p3 = new Path(files[2]);
Path p4 = new Path(files[3]);
Path p5 = new Path(files[4]);
Job job = Job.getInstance(conf, "Multiple Input Files");
job.setJarByClass(multipleinputsfilesNew.class);

MultipleInputs.addInputPath(job, p1, TextInputFormat.class,
Mapper1.class); MultipleInputs.addInputPath(job, p2,
TextInputFormat.class, Mapper2.class);
```

```
        MultipleInputs.addInputPath(job, p3, TextInputFormat.class,
        Mapper3.class); MultipleInputs.addInputPath(job, p4,
        TextInputFormat.class, Mapper4.class);

        job.setReducerClass(Red.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        FileOutputFormat.setOutputPath(job, p5);

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

Output:

Hue

Home / user / training / prashant / MultiOutputNewUpdated / **part-r-00000**

ACTIONS

View As Binary

Edit File

Download

View File Location

Refresh

INFO

Last Modified
Aug. 15, 2024 9:55 p.m.

User
training

Group
supergroup

Size
75 bytes

Mode
100644

First Block | Previous Block | Next Block | La

```
user1   200     5       200     10
user2   150     10      120     10
user3   20      20      60      15
user4   15      15      20      20
```

First Block | Previous Block | Next Block | La

localhost:50030/jobdetails.jsp?jobid=job_202408091304_0008&refresh=0

Most Visited | Hadoop NameNode | Hadoop JobTracker | Hadoop YARN | Hue | HBase Master

Hadoop job_202408091304_0008 on 0

User: training
Job Name: Multiple Input Files
Job File: hdfs://0.0.0.0:8020/var/lib/hadoop-hdfs/cache/mapred/mapred/staging/training/.staging/job_202408091304_0008/job.xml
Submit Host: localhost.localdomain
Submit Host Address: 127.0.0.1
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Fri Aug 16 10:25:22 IST 2024
Finished at: Fri Aug 16 10:25:31 IST 2024
Finished in: 9sec
Job Cleanup: Successful

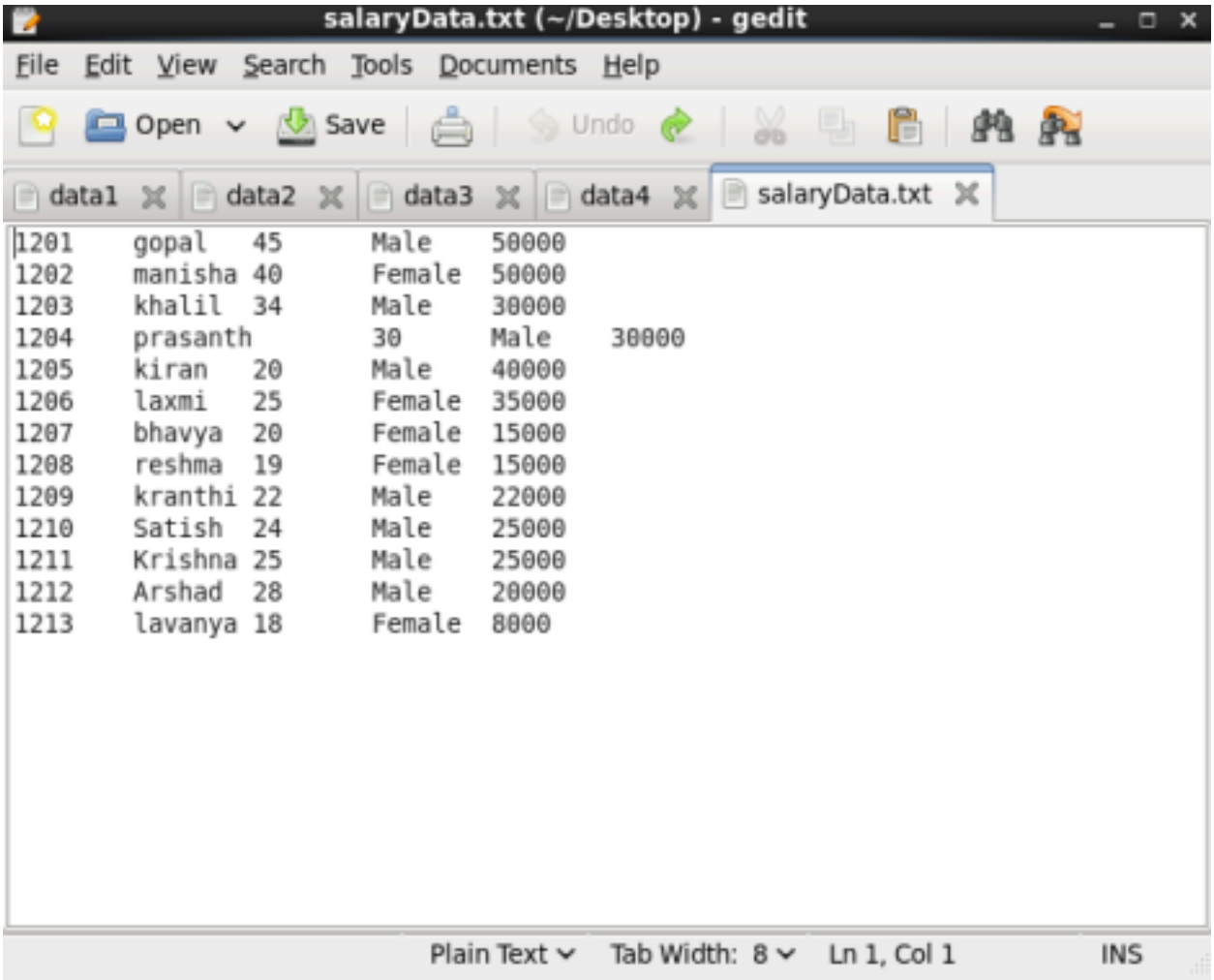
Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	4	0	0	4	0	0 / 0
reduce	100.00%	1	0	0	1	0	0 / 0

	Counter	Map	Reduce	Total
	FILE: Number of bytes read	0	0	179
	FILE: Number of bytes written	0	0	919,052
	FILE: Number of read operations	0	0	0

6.Program to find the male and female employee with salary in the range of age below or equal to 20, age greater than 20 and less than equal to 30, age greater than 30.

13 Prashant Vithal Godhe
Big Data Partitioner Data

Data:



Code:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
```

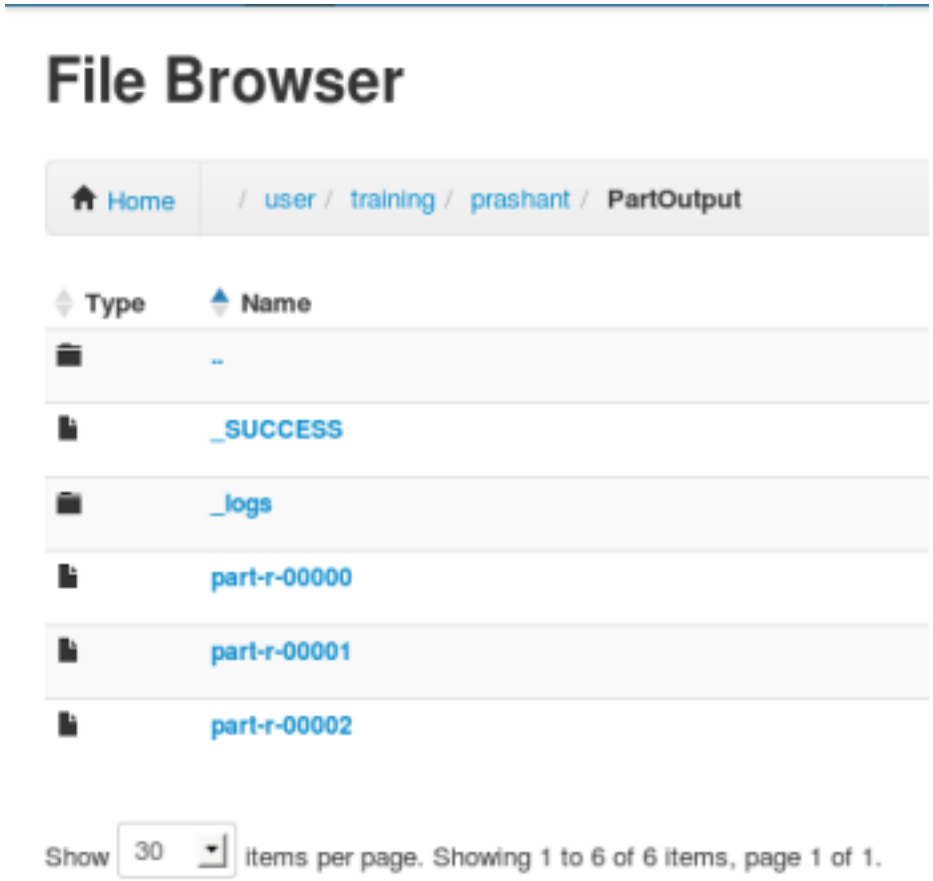
```
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Partitioner;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.util.*;
import java.io.*;

public class DataPartitionEmployee extends Configured implements Tool

                        { public static class MapClass extends

  Mapper<LongWritable,Text,Text,Text>{ public void map(LongWritable

                            key,Text value,Context context){

          try {
              String[] str= value.toString().split("\t",-3);
              String gender=str[3];
              context.write(new Text(gender), new Text(value));
          }
          catch(Exception e){
              System.out.println(e.getMessage());
          }

      }
  }

  //Reducer class

  public static class ReduceClass extends
      Reducer<Text,Text,Text,IntWritable>{ public int max=-1;

      public void reduce(Text key,Iterable<Text>values,Context context)throws
IOException ,InterruptedException {
          max=-1;
          for(Text val:values){
              String[] str= val.toString().split("\t",-3);
              if(Integer.parseInt(str[4])>max)
                  max=Integer.parseInt(str[4]);
          }
```

```
        context.write(new Text(key), new IntWritable(max));
    }
}

//partitioner class
public static class CaderPartitioner extends Partitioner<Text,Text>{

    public int getPartition(Text key,Text value,int
        numReduceTasks){ String[]
        str=value.toString().split("\t");
        int age=Integer.parseInt(str[2]);

        if(numReduceTasks==0){
            return 0;
        }
        if(age<=20){
            return 0;
        }
        else if(age>20 && age<=30){
            return 1 % numReduceTasks;
        }
        else{
            return 2 % numReduceTasks;
        }
    }
}

public int run(String[]arg) throws Exception {
    Configuration conf= getConf();
    Job job= new Job(conf,"");
    job.setJarByClass( DataPartitionEmployee.class);
    FileInputFormat.setInputPaths(job,new
    Path(arg[0]));
    FileOutputFormat.setOutputPath(job, new
    Path(arg[1]));
    job.setMapperClass(MapClass.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);
    //set partitioner statement
    job.setPartitionerClass(CaderPartitioner.class);
    job.setReducerClass(ReduceClass.class);
    job.setNumReduceTasks(3);
    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.
    class); job.setOutputKeyClass(Text.class);
```

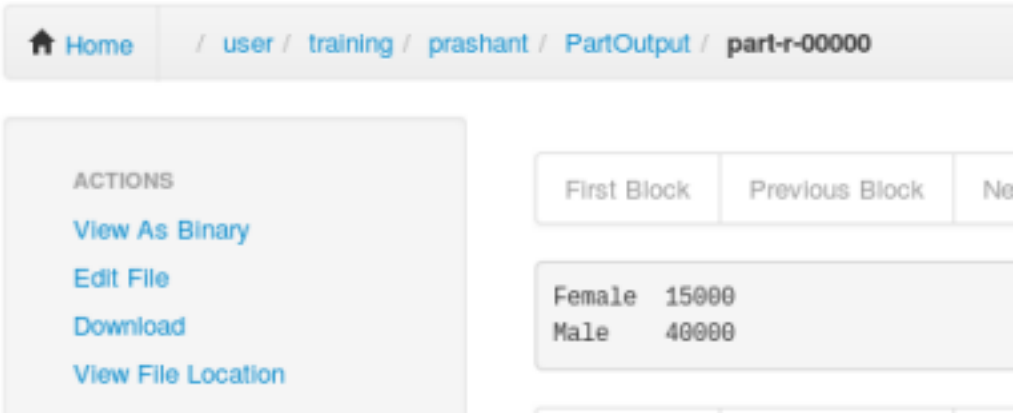
```
        job.setOutputValueClass(IntWritable.class);
        System.exit(job.waitForCompletion(true)?0:1);
        return 0;
    }

    public static void main(String[]args) throws Exception{
        int res =ToolRunner.run(new Configuration(),new
        DataPartitionEmployee(),args); System.exit(0);


    }

}
```

Output:

```
[training@localhost ~]$ hadoop jar /home/training/part.jar /user/training/prashant/salaryData.txt /user/training/prashant/PartOutput
24/08/16 11:47:56 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
24/08/16 11:47:56 INFO input.FileInputFormat: Total input paths to process : 1
24/08/16 11:47:56 WARN snappy.LoadSnappy: Snappy native library is available
24/08/16 11:47:56 INFO snappy.LoadSnappy: Snappy native library loaded
24/08/16 11:47:56 INFO mapred.JobClient: Running job: job_202408091304_0010
24/08/16 11:47:57 INFO mapred.JobClient:  map 0% reduce 0%
24/08/16 11:48:00 INFO mapred.JobClient:  map 100% reduce 0%
24/08/16 11:48:02 INFO mapred.JobClient:  map 100% reduce 33%
24/08/16 11:48:03 INFO mapred.JobClient:  map 100% reduce 66%
24/08/16 11:48:04 INFO mapred.JobClient:  map 100% reduce 100%
24/08/16 11:48:04 INFO mapred.JobClient: Job complete: job_202408091304_0010
24/08/16 11:48:04 INFO mapred.JobClient: Counters: 32
24/08/16 11:48:04 INFO mapred.JobClient:   File System Counters
24/08/16 11:48:04 INFO mapred.JobClient:     FILE: Number of bytes read=469
24/08/16 11:48:04 INFO mapred.JobClient:     FILE: Number of bytes written=737150
24/08/16 11:48:04 INFO mapred.JobClient:     FILE: Number of read operations=0
24/08/16 11:48:04 INFO mapred.JobClient:     FILE: Number of large read operations=0
24/08/16 11:48:04 INFO mapred.JobClient:     FILE: Number of write operations=0
24/08/16 11:48:04 INFO mapred.JobClient:     HDFS: Number of bytes read=483
24/08/16 11:48:04 INFO mapred.JobClient:     HDFS: Number of bytes written=72
24/08/16 11:48:04 INFO mapred.JobClient:     HDFS: Number of read operations=2
24/08/16 11:48:04 INFO mapred.JobClient:     HDFS: Number of large read operations=0
24/08/16 11:48:04 INFO mapred.JobClient:     HDFS: Number of write operations=3
24/08/16 11:48:04 INFO mapred.JobClient:   Job Counters
24/08/16 11:48:04 INFO mapred.JobClient:     Launched map tasks=1
24/08/16 11:48:04 INFO mapred.JobClient:     Launched reduce tasks=3
24/08/16 11:48:04 INFO mapred.JobClient:     Data-local map tasks=1
24/08/16 11:48:04 INFO mapred.JobClient:     Total time spent by all maps in occupied slots (ms)=2210
24/08/16 11:48:04 INFO mapred.JobClient:     Total time spent by all reduces in occupied slots (ms)=6834
24/08/16 11:48:04 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots (ms)=0
24/08/16 11:48:04 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0
24/08/16 11:48:04 INFO mapred.JobClient:   Map-Reduce Framework
24/08/16 11:48:04 INFO mapred.JobClient:     Map input records=13
24/08/16 11:48:04 INFO mapred.JobClient:     Map output records=13
24/08/16 11:48:04 INFO mapred.JobClient:     Map output bytes=425
24/08/16 11:48:04 INFO mapred.JobClient:     Input split bytes=122
```



Age Below and Equal to 20



Age Greater than 20 below Equal to 30

Home / user / training / prashant / PartOutput / part-r-00001

ACTIONS

View As Binary

Edit File

Download

View File Location

First Block | Previous Block | Next B

```
Female  35000
Male    30000
```

Age Greater than 30

Home / user / training / prashant / PartOutput / part-r-00002

ACTIONS

View As Binary

Edit File

Download

View File Location

First Block | Previous Block | Nex

```
Female  50000
Male    50000
```

Hadoop job_202408091304_0010 on 0

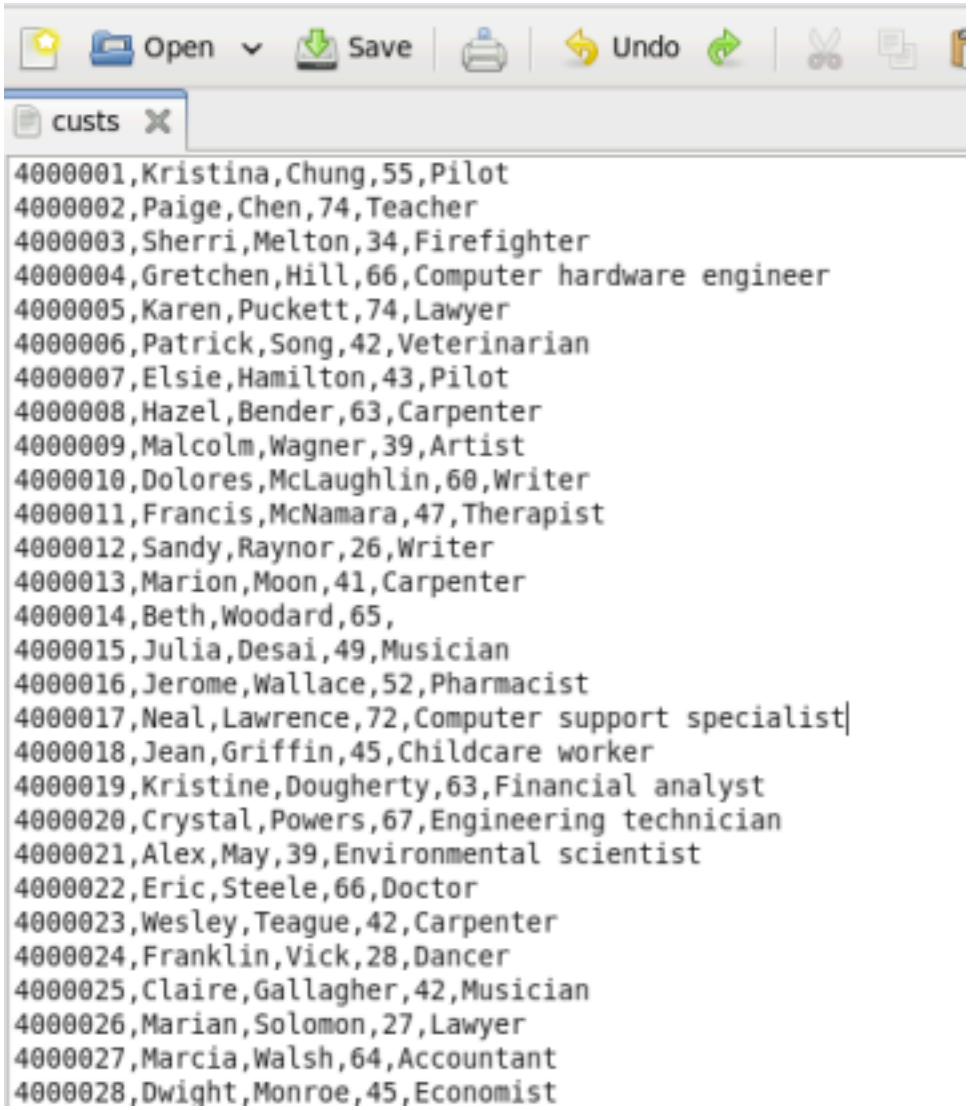
User: training
Job Name: part.jar
Job File: hdfs://0.0.0.0:8020/var/lib/hadoop-hdfs/cache/mapred/mapred/staging/training/.staging/job_202408091304_0010/job.xml
Submit Host: localhost.localdomain
Submit Host Address: 127.0.0.1
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Fri Aug 16 11:47:56 IST 2024
Finished at: Fri Aug 16 11:48:03 IST 2024
Finished in: 7sec
Job Cleanup: Successful

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	1	0	0	1	0	0 / 0
reduce	100.00%	3	0	0	3	0	0 / 0

7.Program to count the number of customers' first names starting with letter f,g,h,i,j.

Name : Prashant Vithal Godhe
Roll No : 13
Partitioner assignment - sorting or counting customer name

Customer Data




```
4000001,Kristina,Chung,55,Pilot
4000002,Paige,Chen,74,Teacher
4000003,Sherri,Melton,34,Firefighter
4000004,Gretchen,Hill,66,Computer hardware engineer
4000005,Karen,Puckett,74,Lawyer
4000006,Patrick,Song,42,Veterinarian
4000007,Elsie,Hamilton,43,Pilot
4000008,Hazel,Bender,63,Carpenter
4000009,Malcolm,Wagner,39,Artist
4000010,Dolores,McLaughlin,60,Writer
4000011,Francis,McNamara,47,Therapist
4000012,Sandy,Raynor,26,Writer
4000013,Marion,Moon,41,Carpenter
4000014,Beth,Woodard,65,
4000015,Julia,Desai,49,Musician
4000016,Jerome,Wallace,52,Pharmacist
4000017,Neal,Lawrence,72,Computer support specialist
4000018,Jean,Griffin,45,Childcare worker
4000019,Kristine,Dougherty,63,Financial analyst
4000020,Crystal,Powers,67,Engineering technician
4000021,Alex,May,39,Environmental scientist
4000022,Eric,Steele,66,Doctor
4000023,Wesley,Teague,42,Carpenter
4000024,Franklin,Vick,28,Dancer
4000025,Claire,Gallagher,42,Musician
4000026,Marian,Solomon,27,Lawyer
4000027,Marcia,Walsh,64,Accountant
4000028,Dwight,Monroe,45,Economist
```

code:

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
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.Mapper;
import org.apache.hadoop.mapreduce.Partitioner;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class lastCount {
  public static class PartitionerEx extends Configured implements Tool {
    public static class MyMap extends Mapper<LongWritable, Text, Text,
    Text> { @Override
      public void map(LongWritable key, Text value, Context
            context) throws IOException,
            InterruptedException {
        try {
            String[] str = value.toString().split(",");
            String name = str[2];
            String firstChar =
            String.valueOf(name.charAt(0)).toUpperCase();
            context.write(new Text(firstChar), new Text(name));
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
      }
    }

    public static class MyReducer extends Reducer<Text, Text, Text,
      IntWritable> { @Override
public void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {
        int count = 0;
        for (Text value : values) {
            count += value.getLength();
        }
        context.write(new Text(""), new
      IntWritable(count)); }
    }

    public static class MyPart extends
```

```
    Partitioner<Text, Text> { @Override
    public int getPartition(Text key, Text value, int
        numReduceTasks) { if (numReduceTasks == 0) {
            return 0;
        }
        char firstChar = key.toString().charAt(0);
        switch (firstChar) {
            case 'F':
                return 0;
            case 'G':
                return 1;
            case 'H':
                return 2;
            case 'I':
                return 3;
            case 'J':
                return 4;
            default:
                return 5;
        }
    }
}

@Override
public int run(String[] args) throws Exception {
    if (args.length != 2) {
        System.out.println("<input path> <output path>");
        return -1;
    }

    Configuration conf = getConf();
    Job job = Job.getInstance(conf, "Part");
    job.setJarByClass(PartitionerEx.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.setMapperClass(MyMap.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);

    job.setPartitionerClass(MyPart.class);
    job.setReducerClass(MyReducer.class);
    job.setNumReduceTasks(5);
```

```
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        return job.waitForCompletion(true) ? 0 : 1;
    }

    public static void main(String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new
        PartitionerEx(), args); System.exit(res);
    }
  }
}
```

Output:

```
File  Edit  View  Search  Terminal  Help
[training@localhost ~]$ hadoop jar /home/training/lastcount.jar /user/training/p
rashant/custs /user/training/prashant/lastName
24/08/22 19:48:19 WARN mapred.JobClient: Use GenericOptionsParser for parsing th
e arguments. Applications should implement Tool for the same.
24/08/22 19:48:19 INFO input.FileInputFormat: Total input paths to process : 1
24/08/22 19:48:19 WARN snappy.LoadSnappy: Snappy native library is available
24/08/22 19:48:19 INFO snappy.LoadSnappy: Snappy native library loaded
24/08/22 19:48:20 INFO mapred.JobClient: Running job: job_202408221915_0001
24/08/22 19:48:21 INFO mapred.JobClient:  map 0% reduce 0%
24/08/22 19:48:27 INFO mapred.JobClient:  map 100% reduce 0%
24/08/22 19:48:31 INFO mapred.JobClient:  map 100% reduce 20%
24/08/22 19:48:32 INFO mapred.JobClient:  map 100% reduce 40%
24/08/22 19:48:35 INFO mapred.JobClient:  map 100% reduce 60%
24/08/22 19:48:36 INFO mapred.JobClient:  map 100% reduce 80%
24/08/22 19:48:38 INFO mapred.JobClient:  map 100% reduce 100%
24/08/22 19:48:39 INFO mapred.JobClient: Job complete: job_202408221915_0001
24/08/22 19:48:39 INFO mapred.JobClient: Counters: 32
24/08/22 19:48:39 INFO mapred.JobClient:   File System Counters
24/08/22 19:48:39 INFO mapred.JobClient:     FILE: Number of bytes read=94798
24/08/22 19:48:39 INFO mapred.JobClient:     FILE: Number of bytes written=12869
00
24/08/22 19:48:39 INFO mapred.JobClient:     FILE: Number of read operations=0
24/08/22 19:48:39 INFO mapred.JobClient:     FILE: Number of large read operatio
ns=0
24/08/22 19:48:39 INFO mapred.JobClient:     FILE: Number of write operations=0
24/08/22 19:48:39 INFO mapred.JobClient:     HDFS: Number of bytes read=381471
24/08/22 19:48:39 INFO mapred.JobClient:     HDFS: Number of bytes written=28
24/08/22 19:48:39 INFO mapred.JobClient:     HDFS: Number of read operations=2
24/08/22 19:48:39 INFO mapred.JobClient:     HDFS: Number of large read operatio
ns=0
24/08/22 19:48:39 INFO mapred.JobClient:     HDFS: Number of write operations=5
24/08/22 19:48:39 INFO mapred.JobClient:   Job Counters
24/08/22 19:48:39 INFO mapred.JobClient:     Launched map tasks=1
24/08/22 19:48:39 INFO mapred.JobClient:     Launched reduce tasks=5
24/08/22 19:48:39 INFO mapred.JobClient:     Data-local map tasks=1
24/08/22 19:48:39 INFO mapred.JobClient:     Total time spent by all maps in occ
upied slots (ms)=5813
```

Contents of directory /user/training/prashant/lastName

Goto : /user/training/prashant/lastI go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modificatio
_SUCCESS	file	0 KB	1	64 MB	2024-08-22
_logs	dir				2024-08-22
part-r-00000	file	0.01 KB	1	64 MB	2024-08-22
part-r-00001	file	0.01 KB	1	64 MB	2024-08-22
part-r-00002	file	0.01 KB	1	64 MB	2024-08-22
part-r-00003	file	0 KB	1	64 MB	2024-08-22
part-r-00004	file	0.01 KB	1	64 MB	2024-08-22

Go back to DFS home

F:

File: /user/training/prashant/lastName/part-r-00000

Goto : /user/training/prashant/lastI go

Go back to dir listing
Advanced view/download options

```
2753
```

G:

File: /user/training/prashant/lastName/part-r-00001

Goto : /user/training/prashant/lastN go

Go back to dir listing
Advanced view/download options

3342

H:

File: /user/training/prashant/lastName/part-r-00002

Goto : /user/training/prashant/lastN go

Go back to dir listing
Advanced view/download options

6154

I:

File: /user/training/prashant/lastName/part-r-00003

Goto : /user/training/prashant/lastN go

Go back to dir listing
Advanced view/download options

82

File: /user/training/prashant/lastName/part-r-00004

Goto : /user/training/prashant/lastN go

Go back to dir listing

Advanced view/download options

J:

```
1113
```

8.Map Reduce Patent Assignment.

Name : Prashant Vithal Godhe Roll No : 13

MAP REDUCE ASSIGNMENT

Data:

code:

```
import java.io.IOException;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
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 subpatentcount {

	public static class PatentMapper extends Mapper<LongWritable, Text,
IntWritable, IntWritable> {
	@Override
	public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
	String line = value.toString();
	String[] parts = line.split(" ");

	if (parts.length > 1) {
		int patentId = Integer.parseInt(parts[0]);

			context.write(new IntWritable(patentId), new IntWritable(1));
	}
	}
	}

	public static class PatentReducer extends Reducer<IntWritable, IntWritable,
IntWritable, IntWritable> {
	@Override
	public void reduce(IntWritable key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException {
	int Subpatents = 0;

	for (IntWritable value : values) {
		Subpatents += value.get();
	}

	context.write(key, new IntWritable(Subpatents));
	}
	}

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

	Job job = Job.getInstance();
	job.setJarByClass(subpatentcount.class);
	job.setJobName("Patent Subpatent Count");

	FileInputFormat.setInputPaths(job, new Path(args[0]));
	FileOutputFormat.setOutputPath(job, new Path(args[1]));

	job.setMapperClass(PatentMapper.class);
	job.setCombinerClass(PatentReducer.class);
```

```
        job.setReducerClass(PatentReducer.class);

        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(IntWritable.class);

        boolean success = job.waitForCompletion(true);
        System.exit(success ? 0 : 1);
        }
    }
```

output:

```
[training@localhost ~]$ hadoop jar /home/training/Desktop/subpatent.jar /user/training/prashant/patent /user/training/prashant/subpatentOP
24/09/05 15:15:40 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
24/09/05 15:15:40 INFO input.FileInputFormat: Total input paths to process : 1
24/09/05 15:15:40 WARN snappy.LoadSnappy: Snappy native library is available
24/09/05 15:15:40 INFO snappy.LoadSnappy: Snappy native library loaded
24/09/05 15:15:40 INFO mapred.JobClient: Running job: job_202409021417_0035
24/09/05 15:15:41 INFO mapred.JobClient:  map 0% reduce 0%
24/09/05 15:15:44 INFO mapred.JobClient:  map 100% reduce 0%
24/09/05 15:15:46 INFO mapred.JobClient:  map 100% reduce 100%
24/09/05 15:15:46 INFO mapred.JobClient: Job complete: job_202409021417_0035
24/09/05 15:15:46 INFO mapred.JobClient: Counters: 32
24/09/05 15:15:46 INFO mapred.JobClient:   File System Counters
24/09/05 15:15:46 INFO mapred.JobClient:     FILE: Number of bytes read=9916
24/09/05 15:15:46 INFO mapred.JobClient:     FILE: Number of bytes written=382736
24/09/05 15:15:46 INFO mapred.JobClient:     FILE: Number of read operations=0
24/09/05 15:15:46 INFO mapred.JobClient:     FILE: Number of large read operations=0
24/09/05 15:15:46 INFO mapred.JobClient:     FILE: Number of write operations=0
24/09/05 15:15:46 INFO mapred.JobClient:     HDFS: Number of bytes read=216900
24/09/05 15:15:46 INFO mapred.JobClient:     HDFS: Number of bytes written=6603
24/09/05 15:15:46 INFO mapred.JobClient:     HDFS: Number of read operations=2
24/09/05 15:15:46 INFO mapred.JobClient:     HDFS: Number of large read operations=0
24/09/05 15:15:46 INFO mapred.JobClient:     HDFS: Number of write operations=1
24/09/05 15:15:46 INFO mapred.JobClient:   Job Counters
```

File: /user/training/prashant/subpatentOP/part-r-00000

Goto : /user/training/prashant/subp go

Go back to dir listing

Advanced view/download options

1	19
2	10
3	15
4	9
5	11
6	21
7	25
8	7
9	15
10	12

Hadoop job_202409021417_0007 on 0

User: training
Job Name: job2550331755564808760.jar
Job File: hdfs://0.0.0.0:8020/var/lib/hadoop-hdfs/cache/mapred/mapred/staging/training/.staging/job_202409021417_0007/job.xml
Submit Host: localhost.localdomain
Submit Host Address: 127.0.0.1
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Thu Sep 05 11:21:33 IST 2024
Finished at: Thu Sep 05 11:21:39 IST 2024
Finished in: 6sec
Job Cleanup: Successful

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	1	0	0	1	0	0 / 0
reduce	100.00%	1	0	0	1	0	0 / 0

	Counter	Map	Reduce	Total
	FILE: Number of bytes read	0	0	269,979
	FILE: Number of bytes written	0	0	1,533,934

Pig Assignment

9.Find Top 10 word from any document using Pig

Name : Prashant Vithal Godhe

Rollno : 13

Date : 06-09-24

Top 10 word count

Data:

code:

A = LOAD '/user/training/shakespere/poems' AS (line:chararray); B = foreach A generate

TOKENIZE(line) as tokens; C = foreach B generate flatten(tokens) as words; D = group C by words;
E = foreach D generate group, COUNT(C);
F = order E by $1 DESC;
G = LIMIT F 10;
DUMP G;

output:

```
2024-09-06 10:11:32,027 [main
2024-09-06 10:11:32,031 [main
2024-09-06 10:11:32,031 [main
(the,1223)
(to,919)
(and,823)
(of,807)
(in,736)
(my,585)
(And,585)
(I,581)
(that,569)
(his,539)
[training@localhost ~]$
```

10.Perform String operations of transaction dataset using PIG.

Name : Prashant Vithal Godhe
Roll no : 13

Tranzdata String functions pig assignment

1.Extract the domain names from the email addresses and normalize them to uppercase.

code:
upperdomains = FOREACH tranz GENERATE UPPER(REGEX_EXTRACT(email, '.*@([^.]+).*', 1)) AS upperdomain;

dump upperdomains;
output:

```
Counters:
Total records written : 28
Total bytes written : 438
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_202409021417_0096


2024-09-19 11:10:59,838 [main] INFO  org.apache.pig.ba
2024-09-19 11:10:59,843 [main] INFO  org.apache.hadoop
2024-09-19 11:10:59,843 [main] INFO  org.apache.pig.ba
(DOMAIN)
(EDUCATION)
(COMPANY)
(DOMAIN)
(STARTUP)
(SERVICES)
(UNIVERSITY)
(DOMAIN)
(EDUCATION)
```

2.Extract and split the Product_Code field into Category, SubCategory, and ProductID.

code:
productCSI = FOREACH
tranz GENERATE REGEX_EXTRACT(product_code, '^(\\w+)\\-(\\w+)\\-(\\d+)$', 1)
AS Category, REGEX_EXTRACT(product_code, '^(\\w+)\\-(\\w+)\\-(\\d+)$', 2) AS

SubCategory, REGEX_EXTRACT(product_code, '^(\\w+)\\-(\\w+)\\-(\\d+)$', 3) AS ProductID;

output:

```
Input(s):
Successfully read 28 records (5591 bytes) from: "/user/training/prashant/Tranzdata.txt"

Output(s):
Successfully stored 28 records (976 bytes) in: "hdfs://0.0.0.0:8020/tmp/temp-1378809411/tmp168091392"

Counters:
Total records written : 28
Total bytes written : 976
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_202409021417_0097


2024-09-19 11:14:36,982 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2024-09-19 11:14:36,983 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2024-09-19 11:14:36,983 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(Electronics,Mobile,001)
(Clothing,WomensWear,102)
(Grocery,Dairy,201)
(Electronics,Television,105)
(HomeFurniture,LivingRoom,301)
(Clothing,MensWear,305)
(Electronics,Mobile,002)
(Grocery,Vegetables,202)
(Clothing,WomensWear,104)
(Electronics,Laptop,203)
```

3.Detect and replace invalid characters (#, $, %) in the Description field and trim leading/trailing spaces.

code:

new_desc = FOREACH tranz GENERATE REPLACE(TRIM(description), '#|\\$|%', '') AS new_desc;

output:

```
2024-09-19 11:19:17,556 [main] INFO  org.apache.pig.backend.hadc
2024-09-19 11:19:17,557 [main] INFO  org.apache.hadoop.mapreduce
2024-09-19 11:19:17,557 [main] INFO  org.apache.pig.backend.hadc
("Latest mobile phone with discount and free delivery")
("Cotton sarees with festival offer")
("Fresh dairy products at low prices")
("Smart TV with exchange offer and 20 off")
("Luxury sofa set with discount")
("Men's formal wear flat 50 off")
("Mobile phone with free accessories")
("Fresh vegetables with 10 off")
("Summer dresses discount")
("Laptop with 15 cashback offer")
("Luxury bed with festival discount")
("Packaged snacks flat 10 off")
("Microwave oven with cashback")
("Designer kurtis with special offers")
("Mobile phone with cashback and free delivery")
("Sectional sofa with free installation")
("Fresh milk with offer on bulk purchase")
("Men's blazer flat 20 off")
("Smartphone with exchange offer")
("King-size bed with off")
("Fresh vegetables with special deal")
("Designer sarees with festival discount")
("Smart TV with cashback")
("Chips with festival offer")
("Luxury recliner with festival discount")
("Cotton sarees with special offer")
("Smartphone with discount and accessories")
("King-size bed with free mattress")
grunt>
```

4.Split the Full_Name field into FirstName and LastName, and normalize them to uppercase. code:

splited_name = FOREACH tranz GENERATE UPPER(REGEX_EXTRACT(TRIM(full_name), '^(\\w+)\\s+(.*)$', 1)) AS fname,UPPER(REGEX_EXTRACT(TRIM(full_name), '^(\\w+)\\s+(.*)$', 2)) AS lname;

output:

```
Input(s):
Successfully read 28 records (5591 bytes) from: "/user/training/prashant/Tranzdata.txt"

Output(s):
Successfully stored 28 records (600 bytes) in: "hdfs://0.0.0.0:8020/tmp/temp1028482582/tmp-1279544029"

Counters:
Total records written : 28
Total bytes written : 600
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_202409021417_0099


2024-09-19 11:25:39,312 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapRe
2024-09-19 11:25:39,317 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input
2024-09-19 11:25:39,317 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - To
(RAHUL,SINGH)
(PRIYA,SHARMA)
(ARJUN,MENON)
(RAHUL,SINGH)
(SNEHA,VERMA)
(AMIT,PATEL)
(KAVITA,AGARWAL)
(RAVI,KUMAR)
(PRIYA,SHARMA)
(SURESH,MALHOTRA)
(PALAK,MEHTA)
(ARUN,SEN)
```

5.Extract protocol, domain, and page name from the URL field and combine protocol and domain into a new field.

code:

pdpg = FOREACH tranz GENERATE REGEX_EXTRACT(url, '^(https?://)([^/]+)(/.*)?$', 1) AS protocol,REGEX_EXTRACT(url, '^(https?://)([^/]+)(/.*)?$', 2) AS domain,REGEX_EXTRACT(url, '^(https?://)([^/]+)(/.*)?$', 3) AS pagename,CONCAT(REGEX_EXTRACT(url, '^(https?://)([^/]+)(/.*)?$', 1), REGEX_EXTRACT(url, '^(https?://)([^/]+)(/.*)?$', 2)) AS pdpg; **output:**

```
Input(s):
Successfully read 28 records (5591 bytes) from: "/user/training/prashant/Tranzdata.txt"

Output(s):
Successfully stored 28 records (2073 bytes) in: "hdfs://0.0.0.0:8020/tmp/temp530310654/tmp182462026"

Counters:
Total records written : 28
Total bytes written : 2073
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_202409021417_0100


2024-09-19 11:39:56,546 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.Map
2024-09-19 11:39:56,552 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total in
2024-09-19 11:39:56,552 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil -
(http://,example.in,/home/index.html,http://example.in)
(https://,shoponline.com,/products/clothing,https://shoponline.com)
(http://,services.org,/contact/support.html,http://services.org)
(http://,example.in,/home/tv.html,http://example.in)
(https://,startup.biz,/about/products,https://startup.biz)
(http://,services.org,/men/formal.html,http://services.org)
(https://,university.ac.in,/electronics/mobile.html,https://university.ac.in)
(http://,grocery.org,/shop/vegetables.html,http://grocery.org)
(https://,shoponline.com,/products/dresses,https://shoponline.com)
(http://,example.in,/laptop.html,http://example.in)
(https://,startup.biz,/furniture/bedroom.html,https://startup.biz)
(http://,grocery.org,/snacks/offers.html,http://grocery.org)
(http://,services.org,/appliances/microwave.html,http://services.org)
(http://,example.in,/womenswear/kurtis.html,http://example.in)
(http://,example.in,/mobile/offer.html,http://example.in)
```

6.Validate phone numbers to ensure they contain only 10 digits, then format them into (XXX) XXX-XXXX format.

code:

```
A = LOAD '/user/training/sharique/tranz' USING PigStorage(',') AS
(txnid:int,custid:int,txndate:chararray,procode:chararray,amount:float,paymeth:chararray,email:c
hararray,description:chararray,name:chararray,url:chararray,phone:chararray);
B = FILTER A BY SIZE(phone) < 11;
C = FOREACH B GENERATE SUBSTRING(phone,0, 3) AS one, SUBSTRING(phone,3, 6) AS
two, SUBSTRING(phone,6, 10) AS three;
D = FOREACH C GENERATE CONCAT(CONCAT(CONCAT(CONCAT(CONCAT('(', one),
')'),two),'-'),three);
```

output:

```
2024-09-19 12:08:02,112 [main] WARN  org.apache.pig.backend.hadoop.executionengine.mapR
educeLayer.MapReduceLauncher - Encountered Warning ACCESSING_NON_EXISTENT_FIELD 1 time(
s).
2024-09-19 12:08:02,112 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapR
educeLayer.MapReduceLauncher - Success!
2024-09-19 12:08:02,113 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFor
mat - Total input paths to process : 1
2024-09-19 12:08:02,113 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util
.MapRedUtil - Total input paths to process : 1
((987)654-3210)
((998)877-6655)
((876)543-2109)
((987)654-3210)
((789)456-1230)
((912)345-6789)
((987)654-3201)
((987)654-1234)
((998)877-6655)
((876)543-2100)
((765)432-1890)
((812)345-6789)
((987)123-4560)
((879)654-3210)
((998)877-6622)
((987)654-5678)
((876)543-2109)
((912)345-6700)
((987)654-3212)
((765)432-1009)
((987)654-1230)
((987)123-4567)
((987)654-3214)
((987)654-1235)
((789)654-3210)
((987)123-4569)
((912)345-6788)
grunt>
```

Hive Assignment

11.Using Hive perform operations on the movie database.

Name : Prashant Vithal Godhe
Roll no : 13

Date 30-09-2024

The movie data was already loaded in the previous lecture so we created the movie table in hive.

code:

```
hive> CREATE EXTERNAL TABLE movie
    > (id INT, name STRING, year INT)
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
    > LOCATION '/user/training/movie';
```

output:

```
hive> CREATE EXTERNAL TABLE movie
    > (id INT, name STRING, year INT)
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
    > LOCATION '/user/training/movie';
OK
Time taken: 1.277 seconds
hive> [training@localhost ~]$ S
```

Added the sql movierating table in hadoop

code:

```
sqoop import --connect jdbc:mysql://localhost/movielens --table movierating
--fields-terminated-by '\t' --username training --password training;
```

output:

```
[3]+  Stopped                 hive
[training@localhost ~]$ hdfs dfs -rmdir 'user/training/movierating';
rmdir: `user/training/movierating': No such file or directory
[training@localhost ~]$ hdfs dfs -rmdir '/user/training/movierating';
[training@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/movielens --table movierating --fields-terminated-by '\t' --username training --password training;
24/09/30 13:56:08 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
24/09/30 13:56:08 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
24/09/30 13:56:08 INFO tool.CodeGenTool: Beginning code generation
24/09/30 13:56:08 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `movierating` AS t LIMIT 1
24/09/30 13:56:08 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `movierating` AS t LIMIT 1
24/09/30 13:56:08 INFO orm.CompilationManager: HADOOP HOME is /usr/lib/hadoop
Note: /tmp/sqoop-training/compile/7747e5524bcfae5a0ee2e51db55c059f/movierating.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
24/09/30 13:56:08 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-training/compile/7747e5524bcfae5a0ee2e51db55c059f/movierating.jar
24/09/30 13:56:08 WARN manager.MySQLManager: It looks like you are importing from mysql.
24/09/30 13:56:08 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
24/09/30 13:56:08 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
24/09/30 13:56:08 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
24/09/30 13:56:08 WARN manager.CatalogQueryManager: The table movierating contains a multi-column primary key. Sqoop will default to the column userid only for this job.
24/09/30 13:56:08 WARN manager.CatalogQueryManager: The table movierating contains a multi-column primary key. Sqoop will default to the column userid only for this job.
24/09/30 13:56:08 INFO mapreduce.ImportJobBase: Beginning import of movierating
24/09/30 13:56:09 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
24/09/30 13:56:09 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(`userid`), MAX(`userid`) FROM `movierating`
24/09/30 13:56:09 INFO mapred.JobClient: Running job: job_202409021417_0111
24/09/30 13:56:10 INFO mapred.JobClient:  map 0% reduce 0%
24/09/30 13:56:15 INFO mapred.JobClient:  map 50% reduce 0%
24/09/30 13:56:18 INFO mapred.JobClient:  map 75% reduce 0%
24/09/30 13:56:18 INFO mapred.JobClient: Job complete: job_202409021417_0111
24/09/30 13:56:18 INFO mapred.JobClient: Counters: 23
24/09/30 13:56:18 INFO mapred.JobClient:   File System Counters
```

After adding the movierating table in hadoop created the external movierating table in hive.

code:

create EXTERNAL table movierating(userid INT,movieid INT,rating int) row format delimited fields terminated by '\t' LOCATION '/user/training/movierating';

output:

```
hive> create EXTERNAL table movierating(userid INT,movieid INT,rating int) row format delimited fields terminated by '\t' LOCATION '/user/training/movierating';
OK
Time taken: 0.879 seconds
hive>
```

HDFS:/user/training/movie - Mozilla Firefox

```
hive> create EXTERNAL table movierating(userid INT,movieid INT,rating int) row format delimited fields terminated by '\t' LOCATION '/user/training/movierating';
OK
Time taken: 1.3 seconds
hive> select * from movierating limit 10;
OK
1       1193    5
1       661     3
```

training - File Browser

1.What is the oldest known movie in the database? (Note that movies with unknown years have a value of 0 in the year field --- these do not belong in your answer.)

In the first query I ordered the movie table by year in ascending order and then selected the first value in the table.

code:

select * from movie where year<>0 ORDER by year limit 1;

output:

```
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 0.96 sec   HDFS Read: 0 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 960 msec
OK
3132    Daddy Long Legs 1919
Time taken: 5.273 seconds
hive>
```

2.List the name and year of all unrated movies (movies where the movie data has no related movierating data).

In the second query i joined the movie and movierating using left outer join to find t the name and year of all unrated movies.

code:

SELECT m.name, m.year FROM movie m LEFT OUTER JOIN movierating r ON m.id = r.movieid WHERE r.movieid IS NULL;

output:

```
Guardian Angel  1994
Headless Body in Topless Bar    1995
Happiness Is in the Field       1995
Gospa   1995
New York Cop    1996
Beyond Bedlam   1993
Desert Winds    1995
Girl in the Cadillac    1995
Homage  1995
Two Crimes      1995
Criminals       1996
Scream of Stone 0
Asfour Stah     1990
```

3.Produce an updated copy of the movie data with two new fields:

numratings
--- the number of ratings for the movie

avgrating
--- the average rating for the movie (Unrated movies are not needed in this copy.)

In the third query I created an updated copy by joining the movie and movierating. code:

Create table update_movie_data as select m.id,m.name, count(r.rating) as numratings, avg(r.rating) as avgrating from movie m join movierating r on m.id=r.movieid group by m.id,m.name having count(r.rating)>0

select * from update_movie_data limit 10;

output:

```
hive> Create table update_movie_data as select m.id,m.name, count(r.rating) as numratings, avg(r.rating) as avgrating from movie
 count(r.rating)>0
    > ;
Total MapReduce jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_202409021417_0127, Tracking URL = http://0.0.0.0:50030/jobdetails.jsp?jobid=job_202409021417_0127
Kill Command = /usr/lib/hadoop/bin/hadoop job  -Dmapred.job.tracker=0.0.0.0:8021 -kill job_202409021417_0127
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
```

```
hive> select * from update_movie_data limit 10;
OK
1       Toy Story       2077    4.146846413095811
2       Jumanji 701     3.20114122681883
3       Grumpier Old Men        478     3.01673640167364
4       Waiting to Exhale       170     2.7294117647058824
5       Father of the Bride Part II     296     3.0067567567567566
6       Heat    940     3.8787234042553194
7       Sabrina 458     3.410480349344978
8       Tom and Huck    68      3.014705882352941
9       Sudden Death    102     2.656862745098039
10      GoldenEye       888     3.5405405405405403
Time taken: 0.041 seconds
hive>
```

4. What are the 10 highest---rated movies?

code:

SELECT m.name, m.year, AVG(mr.rating) AS avgRating FROM movie m
JOIN movierating mr ON m.id = mr.movieid
GROUP BY m.id, m.name, m.year
ORDER BY avgRating DESC
LIMIT 10;

output:

```
One Little Indian       1973    5.0
Bittersweet Motel       2000    5.0
Ulysses 0       5.0
Smashing Time   1967    5.0
Baby, The       1973    5.0
Gate of Heavenly Peace, The     1995    5.0
Song of Freedom 1936    5.0
Schlafes Bruder 0       5.0
Lured   1947    5.0
I Am Cuba       0       4.8
Time taken: 21.647 seconds
hive>
```

Extra Assignment

Selected the user and occupation and joined them using left outer join and find the id and gender of occupation as farmers.

code:

select u.id,u.gender from user u left outer join occupation o on u.occupationid = o.id where o.name = 'farmer';

output:

```
mysql> select u.id,u.gender from user u left outer join occupation o on u.occupationid = o.id where o.name = 'farmer';
+------+--------+
| id   | gender |
+------+--------+
|   42 | M      |
|  550 | M      |
|  704 | F      |
| 1011 | M      |
| 1559 | M      |
| 1736 | M      |
| 1787 | M      |
| 1818 | M      |
| 2349 | M      |
| 2405 | M      |
| 2758 | M      |
| 3040 | M      |
| 3404 | M      |
| 3836 | F      |
| 4486 | M      |
| 4987 | F      |
| 5287 | M      |
+------+--------+
17 rows in set (0.02 sec)
```

Selected the average age of artists from user by joining with the occupation table. code:

select AVG(u.age) from user u left outer join occupation o on u.occupationid = o.id where o.name = 'artist';

output:

```
mysql> select AVG(u.age) fr
+------------+
| AVG(u.age) |
+------------+
|    30.6629 |
+------------+
1 row in set (0.01 sec)

mysql> Ctrl-C -- exit!
Aborted
[training@localhost ~]$
```

H Base Assignment

12.HBase Assignment.

Name: Prashant Vithal Godhe
Roll No : 13
HBASE Assignment
Date : 07-10-2024

1. create a test table called “test” with a column family name ‘data’

code:
create 'test', 'data'
output:

```
hbase(main):004:0> create 'test', 'data'
0 row(s) in 1.0560 seconds

hbase(main):005:0>
```

2. check whether the table is created or not

code:
list
output:

```
hbase(main):005:0> list
TABLE
employee
t1
tabnew
test
tnew1
5 row(s) in 0.0110 seconds

hbase(main):006:0>
```

3. put some data in the table. Here, we will create three different rows with different columns under the

same column family:

$ put ‘test’, ‘row1’, ‘data:name’, ‘alex’

$ put ‘test’, ‘row2’, ‘data:age’, ‘37’

$ put ‘test’, ‘row3’, ‘data:city’, ‘vancouver’

code:
put 'test', 'row1', 'data:name', 'alex'
put 'test', 'row2', 'data:age', '37'
put 'test', 'row3', 'data:city', 'Vancuover'

output:

```
hbase(main):006:0> put 'test', 'row1', 'data:name', 'alex'
0 row(s) in 0.0170 seconds
```

```
hbase(main):010:0> scan 'test'
ROW                                                    COLUMN+CELL
 row1                                                  column=data:name, timestamp=1728290239920, value=alex
 row2                                                  column=data:age, timestamp=1728290425492, value=23
2 row(s) in 0.0030 seconds
```

```
hbase(main):015:0> put 'test', 'row2', 'data:age', '37'
0 row(s) in 0.0020 seconds
```

```
hbase(main):016:0> put 'test', 'row3', 'data:city', 'Vancouver'
0 row(s) in 0.0030 seconds
```

```
hbase(main):018:0> scan 'test'
ROW                                                    COLUMN+CELL
 row1                                                  column=data:name, timestamp=1728290239920, value=alex
 row2                                                  column=data:age, timestamp=1728290684327, value=37
 row3                                                  column=data:city, timestamp=1728290710341, value=Vancouver
3 row(s) in 0.0030 seconds
```

4.list all entries in your table

code:
scan ‘test’
output:

```
hbase(main):018:0> scan 'test'
ROW                                                    COLUMN+CELL
 row1                                                  column=data:name, timestamp=1728290239920, value=alex
 row2                                                  column=data:age, timestamp=1728290684327, value=37
 row3                                                  column=data:city, timestamp=1728290710341, value=Vancouver
3 row(s) in 0.0030 seconds
```

5. count the number of rows in the table

code:
count 'test'
output:

```
hbase(main):014:0> count 'test'
3 row(s) in 0.0060 seconds

hbase(main):015:0>
```

6. Write down a command that will add age to ‘alex’.

code:
put 'test', 'row1', 'data:age', '30'
output:

```
hbase(main):020:0> scan 'test'
ROW                                                    COLUMN+CELL
 row1                                                  column=data:age, timestamp=1728290801423, value=30
 row1                                                  column=data:name, timestamp=1728290239920, value=alex
 row2                                                  column=data:age, timestamp=1728290684327, value=37
 row3                                                  column=data:city, timestamp=1728290710341, value=Vancouver
3 row(s) in 0.0060 seconds
```

7. update the age in row2 to 32.

code:
put 'test', 'row2', 'data:age', '32'
output:

```
hbase(main):021:0> put 'test', 'row2', 'data:age', '32'
0 row(s) in 0.0060 seconds

hbase(main):022:0> scan 'test'
ROW                                                    COLUMN+CELL
 row1                                                  column=data:age, timestamp=1728290801423, value=30
 row1                                                  column=data:name, timestamp=1728290239920, value=alex
 row2                                                  column=data:age, timestamp=1728290861056, value=32
 row3                                                  column=data:city, timestamp=1728290710341, value=Vancouver
3 row(s) in 0.0070 seconds
```

8. use ’scan’ to show all rows with a value under the column ’data:age’

code:
scan 'test', FILTER => "ColumnPrefixFilter('age')"
output:

```
hbase(main):023:0> scan 'test', FILTER => "ColumnPrefixFilter('age')"
ROW                                                    COLUMN+CELL
 row1                                                  column=data:age, timestamp=1728290801423, value=30
 row2                                                  column=data:age, timestamp=1728290861056, value=32
2 row(s) in 0.0120 seconds
```

9. add a new city in ‘data:city’ for row3 to ’New York’

code:
put 'test', 'row3', 'data:city', 'New York'

output:

```
hbase(main):024:0> put 'test', 'row3', 'data:city', 'New York'
0 row(s) in 0.0030 seconds

hbase(main):025:0> scan 'test'
ROW                                                    COLUMN+CELL
 row1                                                  column=data:age, timestamp=1728290801423, value=30
 row1                                                  column=data:name, timestamp=1728290239920, value=alex
 row2                                                  column=data:age, timestamp=1728290861056, value=32
 row3                                                  column=data:city, timestamp=1728290951429, value=New York
3 row(s) in 0.0050 seconds
```

10 add gender information to row1 and row3

code:

put 'test', 'row3', 'data:gender', 'male'

put 'test', 'row1', 'data:gender', 'female'

output:

```
hbase(main):002:0> put 'test', 'row3', 'data:gender', 'male'
0 row(s) in 0.0070 seconds

hbase(main):003:0> put 'test', 'row1', 'data:gender', 'female'
0 row(s) in 0.0050 seconds

hbase(main):004:0> scan 'test'
ROW                                                    COLUMN+CELL
 row1                                                  column=data:age, timestamp=1728290801423, value=30
 row1                                                  column=data:gender, timestamp=1728291112836, value=female
 row1                                                  column=data:name, timestamp=1728290239920, value=alex
 row2                                                  column=data:age, timestamp=1728290861056, value=32
 row3                                                  column=data:city, timestamp=1728290951429, value=New York
 row3                                                  column=data:gender, timestamp=1728291101382, value=male
3 row(s) in 0.0050 seconds
```

11. use 'scan' to show all versions of all cells

code:

scan 'test', {VERSIONS => 3}

output:

```
hbase(main):005:0> scan 'test', {VERSIONS => 3}
ROW                                                    COLUMN+CELL
 row1                                                  column=data:age, timestamp=1728290801423, value=30
 row1                                                  column=data:gender, timestamp=1728291112836, value=female
 row1                                                  column=data:name, timestamp=1728290239920, value=alex
 row2                                                  column=data:age, timestamp=1728290861056, value=32
 row2                                                  column=data:age, timestamp=1728290684327, value=37
 row2                                                  column=data:age, timestamp=1728290425492, value=23
 row3                                                  column=data:city, timestamp=1728290951429, value=New York
 row3                                                  column=data:city, timestamp=1728290718341, value=Vancouver
 row3                                                  column=data:city, timestamp=1728290505402, value=Thane
 row3                                                  column=data:gender, timestamp=1728291101382, value=male
3 row(s) in 0.0290 seconds
```

12. display all the row created or modified in last 10 min.

code:

scan 'test', {TIMERANGE => [172890801423, 1728291101382]}

output:

```
hbase(main):010:0> scan 'test', {TIMERANGE => [1500850658864, 1510854270767]}
ROW                    COLUMN+CELL
0 row(s) in 0.0180 seconds
```

13. display age column values created or modified in last 15 min.

code:

scan 'test', {COLUMN => 'data:age', TIMERANGE => [172890801423, 1728291101382]}

output:

```
hbase(main):012:0> scan 'test', {COLUMN => 'data:age', TIMERANGE => [1537850658864, 153
7854270767]}
ROW                    COLUMN+CELL
0 row(s) in 0.0060 seconds

hbase(main):013:0>
```

14. display row3 of city column values created or modified in last 15 min.

code:

scan 'test', {COLUMN => 'data:city', TIMERANGE => [172829006462,1728290525663]}

output:

```
hbase(main):014:0> get 'test', 'row', {COLUMN => 'data:city', TIMERANGE => [15378506588
64, 1537854270767]}
COLUMN                 CELL
0 row(s) in 0.0070 seconds

hbase(main):015:0>
```

15. delete table data only

code:

truncate 'test'

output:

```
hbase(main):015:0> truncate 'test'
Truncating 'test' table (it may take a while):
 - Disabling table...
 - Dropping table...
 - Creating table...
0 row(s) in 3.2500 seconds

hbase(main):016:0>
```

16. delete table 'test'

code:

disable 'test'

drop 'test'

output:

```
hbase(main):016:0> disable 'test'
0 row(s) in 2.0310 seconds

hbase(main):017:0> drop 'test'
0 row(s) in 1.1100 seconds

hbase(main):018:0> 
```