**MapReduceDriver**

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.Text;

import org.apache.hadoop.mapreduce.Job;

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

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

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

public class MapReduceDriver {

public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {

Path input\_dir1=new Path("hdfs://localhost:9000/user/local/employee.txt");

Path input\_dir2=new Path("hdfs://localhost:9000/user/local/employee\_reference.txt");

Path output\_dir=new Path("hdfs://localhost:9000/output\_data/");

Configuration conf = new Configuration();

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

job.setJarByClass(MapReduceDriver.class);

job.setMapperClass(MapReduceMapper.class);

job.setMapperClass(MapReduceMapperTwo.class);

job.setReducerClass(MapReduceReducer.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.class);

job.setNumReduceTasks(1);

MultipleInputs.addInputPath(job,input\_dir1, TextInputFormat.class, MapReduceMapper.class);

MultipleInputs.addInputPath(job,input\_dir2, TextInputFormat.class, MapReduceMapperTwo.class);

FileOutputFormat.setOutputPath(job,output\_dir );

output\_dir.getFileSystem(job.getConfiguration()).delete(output\_dir,true);

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

}

}

**MapReduceMapper**

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

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

public void map(LongWritable offset, Text value, Context context) throws IOException,

InterruptedException {

String line = value.toString();

String[] words = line.split(",");

if(words[4].equals("developer"))

{

String key = words[0]+words[1];

int salary= Integer.parseInt(words[2]+words[3]);

System.out.println(key+"::"+salary);

context.write(new Text(key), new IntWritable(salary));

}

}

}

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

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

public void map(LongWritable offset, Text value, Context con) throws IOException,

InterruptedException {

String line = value.toString();

String[] words = line.split(",");

String key = words[0]+words[1];

con.write(new Text(key), new IntWritable(0));

}

}

**MapReduceReducer**

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class MapReduceReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

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

Context context) throws IOException, InterruptedException {

System.out.println("MyReducer.reducer()"+key);

System.out.println("hello from reducer");

int total=0;

while(salaries.iterator().hasNext()) {

total += salaries.iterator().next().get();

}

System.out.println(total);

if(total>0){

context.write(key, new IntWritable(total));

}

}

}