Analysis of NYSE dataset:

**Summary:**

This project was created to analyze various trends in the stock market by creating programs for average stock price per year, average per year – by date and symbol, overall average prices, top 10 stocks of all time, joined company and stock data for analysis, and binned and partitioned data.

It was also meant to learn Hadoop MapReduce, HBase, Hive and Pig

**Website Link of data:**

<https://www.kaggle.com/dgawlik/nyse>

prices.csv and securities.csv

**Analysis Performed:**

Top 10 – filtering pattern

Average – summarization patterns (by composite key, using combiner, using partitioner)

Reduce side join pattern – inner join

Data organization pattern – partitioning and binning pattern

**Conclusions:**

All the experiments were successful and all the analysis was successfully made using patterns.

Implemented Hadoop on local as well as aws. Implemented hbase, mapreduce, hive.

**Source Code**:

1. **Average:**
2. **Average.class**

/\*

Use the NYSE files in HDFS to find the average price of stock\_price\_high values

Remark - Completed and Running (NYSE\_Output\_11)

\*/

package Average;

import java.io.IOException;

import java.net.URI;

import java.net.URISyntaxException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FSDataInputStream;

import org.apache.hadoop.fs.FSDataOutputStream;

import org.apache.hadoop.fs.FileStatus;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.Text;

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.mapreduce.lib.output.TextOutputFormat;

/\*\*

\*

\* @author shruti

\*/

public class Average {

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

// TODO code application logic here

Configuration conf = new Configuration();

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

job.setJarByClass(Average.class);

job.setMapperClass(Avg\_Mapper.class);

job.setMapOutputKeyClass(YearPrice.class);

job.setMapOutputValueClass(DoubleWritable.class);

job.setReducerClass(Avg\_Reducer.class);

job.setOutputKeyClass(YearPrice.class);

job.setOutputValueClass(DoubleWritable.class);

Path in = new Path(args[0]);

Path out = new Path(args[1]);

FileInputFormat.setInputPaths(job, in);

FileOutputFormat.setOutputPath(job, out);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

job.setPartitionerClass(AvgPartitioner.class);

job.setNumReduceTasks(7);

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

}

}

1. **Avg\_Mapper.class**
2. package Average;
3. import java.io.IOException;
4. import org.apache.hadoop.io.DoubleWritable;
5. import org.apache.hadoop.io.Text;
6. import org.apache.hadoop.mapreduce.Mapper;
7. /\*\*
8. \*
9. \* @author shruti
10. \*/
11. public class Avg\_Mapper extends Mapper <Object, Text, YearPrice, DoubleWritable > {
13. public void map (Object key, Text value, Context context) throws IOException, InterruptedException
14. {
15. try{
17. String row[] = value.toString().split(",");
18. String symbol = row[1].trim();
19. double price = Double.parseDouble(row[3].trim());
20. String strDate = row[0];
21. String dateparts[] = strDate.split("/");
22. String yer = dateparts[2].substring(0,4).trim();
23. YearPrice yp = new YearPrice(yer, symbol);
24. // double stock\_price = Double.valueOf(price);
26. context.write(yp, new DoubleWritable(price));

29. }catch (NumberFormatException e){
31. }
32. }

35. }

**3 Avg\_Reducer:**

package Average;

import java.io.IOException;

import org.apache.hadoop.io.DoubleWritable;

//import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author shruti

\*/

public class Avg\_Reducer extends Reducer <YearPrice, DoubleWritable, YearPrice, DoubleWritable> {

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

{

float count = 0;

float sum =0;

for(DoubleWritable val: values)

{

sum += val.get();

count += 1;

}

context.write(key,new DoubleWritable(sum/count));

}

}

**4 AvgPartitioner**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Average;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.mapreduce.Partitioner;

/\*\*

\*

\* @author shruti

\*/

public class AvgPartitioner extends Partitioner<YearPrice, DoubleWritable>{

@Override

public int getPartition(YearPrice key, DoubleWritable value, int i) {

return (key.getYear().hashCode() % i);

}

}

**5 YearPrice**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Average;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import java.util.Objects;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableUtils;

/\*\*

\*

\* @author shruti

\*/

public class YearPrice implements Writable, WritableComparable<YearPrice>{

private String year;

private String symbol;

public YearPrice() {

}

public YearPrice(String year, String symbol) {

this.year = year;

this.symbol = symbol;

}

public String getYear() {

return year;

}

public void setYear(String year) {

this.year = year;

}

public String getSymbol() {

return symbol;

}

public void setSymbol(String symbol) {

this.symbol = symbol;

}

@Override

public int hashCode() {

int hash = 5;

hash = 71 \* hash + Objects.hashCode(this.symbol);

return hash;

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

final YearPrice other = (YearPrice) obj;

if (!Objects.equals(this.symbol, other.symbol)) {

return false;

}

return true;

}

@Override

public void write(DataOutput d) throws IOException {

WritableUtils.writeString(d, year);

WritableUtils.writeString(d, symbol);

}

@Override

public void readFields(DataInput di) throws IOException {

year = WritableUtils.readString(di);

symbol = WritableUtils.readString(di);

}

@Override

public String toString() {

return symbol + " "+year;

}

@Override

public int compareTo(YearPrice o) {

int result = year.compareTo(o.year);

if(result ==0)

{

result = symbol.compareTo(o.symbol);

}

return result;

}

}

1. **AverageProj**

**1 AveragePrice**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package AverageProj;

/\*\*

\*

\* @author shruti

\*/

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.DoubleWritable;

//import org.apache.hadoop.io.AverageTuple;

//import org.apache.hadoop.io.YearPrice;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

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.input.TextInputFormat;

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

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

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

/\*\*

\*

\* @author shruti

\*/

public class AveragePrice extends Configured implements Tool {

/\*\*

\* @param args the command line arguments

\*/

public static class AvMapper

extends Mapper<LongWritable, Text, YearPrice,DoubleWritable> {

//private Text key = new Text();

//Text tup = new Text();

YearPrice yp = new YearPrice();

// private final static SimpleDateFormat frmt = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS");

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

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

double price = Double.parseDouble(words[3].trim());

// int year = Integer.parseInt(words[0].trim().substring(6,10));

String strDate = words[0];

String dateparts[] = strDate.split("/");

String yer = dateparts[2].substring(0,4).trim();

String symbol = words[1].trim();

//tup.setSymbol(symbol);

yp.setSymbol(symbol);

yp.setYear(yer);

//yp.setPrice(price);

//int count =

//ext.set(d);

//AverageTuple tup = new AverageTuple(symbol,year);

context.write(yp,new DoubleWritable(price));

}

}

public static class AvReducer extends Reducer<YearPrice, DoubleWritable, YearPrice, DoubleWritable> {

// public AverageTuple result = new AverageTuple();

//YearPrice yp=new YearPrice();

public void reduce(YearPrice key, Iterable<DoubleWritable> values,

Reducer.Context context)

throws IOException, InterruptedException {

double sum = 0;

int count = 0;

//int year = 0;

for(DoubleWritable v:values){

sum += v.get();

count += 1;

}

double av=sum/count;

for (DoubleWritable val: values )

{

context.write(key, av);

}

//double av = sum/count;

}

}

// public static class AvPartitioner extends Partitioner<AverageTuple, YearPrice> {

//

// @Override

// public int getPartition(AverageTuple key, YearPrice value, int i) {

//

// String year = String.valueOf(key.getYear());

// return (year.hashCode() & Integer.MAX\_VALUE) % i;

// }

//

//

// }

public int run(String[] args) throws Exception,ClassNotFoundException {

Configuration conf = getConf();

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

job.setJarByClass(AveragePrice.class);

//final File f = new File(AveragePrice.class.getProtectionDomain().getCodeSource().getLocation().getPath());

Path in = new Path(args[0]);

Path out = new Path(args[1]);

FileInputFormat.setInputPaths(job, in);

FileOutputFormat.setOutputPath(job, out);

job.setMapperClass(AvMapper.class);

job.setMapOutputKeyClass(YearPrice.class);

job.setMapOutputValueClass(DoubleWritable.class);

job.setGroupingComparatorClass(YearSymComparator.class);

// job.setCombinerClass(AvReducer.class);

job.setReducerClass(AvReducer.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

job.setOutputKeyClass(YearPrice.class);

job.setOutputValueClass(DoubleWritable.class);

job.setPartitionerClass(AvgPartitioner.class);

job.setNumReduceTasks(7);

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

return 0;

}

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

int res = ToolRunner.run(new Configuration(), new AveragePrice(), args);

System.exit(res);

}

}

**2 AvgPartitioner**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package AverageProj;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.mapreduce.Partitioner;

/\*\*

\*

\* @author shruti

\*/

public class AvgPartitioner extends Partitioner<YearPrice, DoubleWritable>{

@Override

public int getPartition(YearPrice key, DoubleWritable value, int i) {

return (key.getYear().hashCode() % i);

}

}

**3 YearPrice**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package AverageProj;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import java.util.Objects;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableUtils;

/\*\*

\*

\* @author shruti

\*/

public class YearPrice implements Writable, WritableComparable<YearPrice>{

private String year;

private String symbol;

public YearPrice() {

}

public YearPrice(String year, String symbol) {

this.year = year;

this.symbol = symbol;

}

public String getYear() {

return year;

}

public void setYear(String year) {

this.year = year;

}

public String getSymbol() {

return symbol;

}

public void setSymbol(String symbol) {

this.symbol = symbol;

}

@Override

public int hashCode() {

int hash = 5;

hash = 71 \* hash + Objects.hashCode(this.symbol);

return hash;

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

final YearPrice other = (YearPrice) obj;

if (!Objects.equals(this.symbol, other.symbol)) {

return false;

}

return true;

}

@Override

public void write(DataOutput d) throws IOException {

WritableUtils.writeString(d, year);

WritableUtils.writeString(d, symbol);

}

@Override

public void readFields(DataInput di) throws IOException {

year = WritableUtils.readString(di);

symbol = WritableUtils.readString(di);

}

@Override

public String toString() {

return symbol + " "+year;

}

@Override

public int compareTo(YearPrice o) {

int result = year.compareTo(o.year);

if(result ==0)

{

result = symbol.compareTo(o.symbol);

}

return result;

}

}

**4 YearSymComparator**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package AverageProj;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableComparator;

/\*\*

\*

\* @author shruti

\*/

public class YearSymComparator extends WritableComparator{

protected YearSymComparator()

{

super(YearPrice.class, true);

}

@Override

public int compare(WritableComparable w1, WritableComparable w2)

{

YearPrice cw1 = (YearPrice) w1;

YearPrice cw2 = (YearPrice) w2;

return (cw1.getYear().compareTo(cw2.getYear()));

}

}

**3)**

**NYSEPartitioning**

**1 NYSEPart**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package PartitioningNYSE;

/\*\*

\*

\* @author shruti

\*/

import java.io.File;

import java.io.IOException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Arrays;

import java.util.Calendar;

import java.util.Date;

import java.util.Map;

import java.util.Set;

import java.util.TreeSet;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.apache.hadoop.conf.Configurable;

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.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Partitioner;

import org.apache.hadoop.mapreduce.Reducer;

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.util.Tool;

import org.apache.hadoop.util.ToolRunner;

/\*\*

\*

\* @author shruti

\*/

public class NYSEPart extends Configured implements Tool {

/\*\*

\* @param args the command line arguments

\*/

public static class IPcountMapper

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

private Text wordText = new Text();

// private final static SimpleDateFormat frmt = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS");

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

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

String strDate = words[0];

String dateparts[] = strDate.split("/");

String yer = dateparts[2];

// String symbol = words[1].trim();

// Date creationDate = frmt.parse(strDate);

// String year = String.valueOf(creationDate.getYear());

// String d = words[3].substring(4,7);

wordText.set(yer);

context.write(wordText, line);

}

}

public static class IPcountReducer extends Reducer<Text, Text, Text, NullWritable> {

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

Reducer.Context context)

throws IOException, InterruptedException {

for(Text t:values)

context.write(t, NullWritable.get());

}

}

public static class IPcountPartitioner extends Partitioner<Text, Text> {

@Override

public int getPartition(Text key, Text value, int i) {

return (key.hashCode() & Integer.MAX\_VALUE) % i;

}

}

public int run(String[] args) throws Exception,ClassNotFoundException {

Configuration conf = getConf();

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

job.setJarByClass(NYSEPart.class);

//final File f = new File(NYSEPart.class.getProtectionDomain().getCodeSource().getLocation().getPath());

Path in = new Path(args[0]);

Path out = new Path(args[1]);

FileInputFormat.setInputPaths(job, in);

FileOutputFormat.setOutputPath(job, out);

job.setMapperClass(IPcountMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setCombinerClass(IPcountReducer.class);

job.setReducerClass(IPcountReducer.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(NullWritable.class);

job.setPartitionerClass(IPcountPartitioner.class);

job.setNumReduceTasks(7);

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

return 0;

}

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

int res = ToolRunner.run(new Configuration(), new NYSEPart(), args);

System.exit(res);

}

}

**4)**

**PriceSecuritiesJoin**

**PriceSecuritiesJoin**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package PriceSecuritiesJoin;

import java.io.IOException;

import java.util.ArrayList;

import org.apache.hadoop.conf.Configuration;

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.Reducer;

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

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

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

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

/\*\*

\*

\* @author shruti

\*/

public class PriceSecuritiesJoin {

public static class JoinMapper1 extends Mapper<Object, Text, Text, Text> {

private Text outkey = new Text();

private Text outvalue = new Text();

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

// Parse the input string into a nice map

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

String symbol = separatedInput[0].substring(1).trim();

// if (symbol == null) {

// return;

//}

// The foreign join key is the user ID

outkey.set(symbol);

// Flag this record for the reducer and then output

outvalue.set("S" + value.toString());

context.write(outkey, outvalue);

}

}

public static class JoinMapper2 extends Mapper<Object, Text, Text, Text> {

private Text outkey = new Text();

private Text outvalue = new Text();

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

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

String symbol = separatedInput[1].trim();

// if (symbol == null) {

// return;

//}

// The foreign join key is the user ID

outkey.set(symbol);

// Flag this record for the reducer and then output

outvalue.set("P" + value.toString());

context.write(outkey, outvalue);

}

}

public static class JoinReducer extends Reducer<Text, Text, Text, Text> {

private static final Text EMPTY\_TEXT = new Text("");

private Text tmp = new Text();

private ArrayList<Text> listA = new ArrayList<Text>();

private ArrayList<Text> listB = new ArrayList<Text>();

private String joinType = null;

public void setup(Context context) {

// Get the type of join from our configuration

joinType = context.getConfiguration().get("join.type");

}

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

// Clear our lists

listA.clear();

listB.clear();

// iterate through all our values, binning each record based on what

// it was tagged with. Make sure to remove the tag!

while (values.iterator().hasNext()) {

tmp = values.iterator().next();

System.out.println(Character.toString((char) tmp.charAt(0)));

// if (Character.toString((char) tmp.charAt(0)).equals("S")) {

if(tmp.charAt(0)=='S')

listA.add(new Text(tmp.toString().substring(1)));

// }

// if (Character.toString((char) tmp.charAt(0)).equals("P")) {

else if(tmp.charAt(0)=='P')

listB.add(new Text(tmp.toString().substring(1)));

//}

// System.out.println(tmp);

}

// Execute our join logic now that the lists are filled

System.out.println(listB.size());

executeJoinLogic(context);

}

private void executeJoinLogic(Context context) throws IOException, InterruptedException {

// if (joinType.equalsIgnoreCase("inner")) {

// If both lists are not empty, join A with B

//System.out.println("here3");

if (!listA.isEmpty() && !listB.isEmpty()) {

for (Text A : listA) {

//System.out.println("here1");

for (Text B : listB) {

//System.out.println("here2");

context.write(A, B);

}

}

}

// }

}

}

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

Configuration conf = new Configuration();

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

job.setJarByClass(PriceSecuritiesJoin.class);

// Use MultipleInputs to set which input uses what mapper

// This will keep parsing of each data set separate from a logical

// standpoint

// The first two elements of the args array are the two inputs

MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class, JoinMapper1.class);

MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class, JoinMapper2.class);

job.getConfiguration().set("join.type", "inner");

//job.setNumReduceTasks(0);

job.setReducerClass(JoinReducer.class);

job.setOutputFormatClass(TextOutputFormat.class);

TextOutputFormat.setOutputPath(job, new Path(args[2]));

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

job.waitForCompletion(true);

}

}

**5)**

**Projcheck**

**Projcheck**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package projcheck;

import java.io.File;

import java.io.IOException;

import java.util.Arrays;

import java.util.Map;

import java.util.Set;

import java.util.TreeSet;

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.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

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.input.TextInputFormat;

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

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

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

/\*\*

\*

\* @author shruti

\*/

public class Projcheck extends Configured implements Tool{

/\*\*

\* @param args the command line arguments

\*/

public static class IPcountMapper

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

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

private Text wordText = new Text();

private Text word = new Text();

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

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

// for (String word : words) {

word.set(words[1].trim()+" "+words[3].trim());

wordText.set(words[0].trim());

context.write(wordText, word);

// }

}

}

public static class IPcountReducer extends Reducer<Text, Text, Text, Text> {

//private IntWritable result = new IntWritable();

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

Reducer.Context context)

throws IOException, InterruptedException {

for(Text t:values)

context.write(key, t);

}

}

public int run(String[] args) throws Exception,ClassNotFoundException {

Configuration conf = getConf();

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

job.setJarByClass(Projcheck.class);

final File f = new File(Projcheck.class.getProtectionDomain().getCodeSource().getLocation().getPath());

//String inFiles = f.getAbsolutePath().replace("/build/classes", "") + "/src/inFiles/access.log";

// String outFiles = f.getAbsolutePath().replace("/build/classes", "") + "/src/outFiles/IPcount";

//use the arguments instead if provided.

// if (args.length > 1) {

// inFiles = args[1];

// outFiles = args[2];

//}

//System.out.println(inFiles);

//System.out.println(outFiles);

Path in = new Path(args[1]);

Path out = new Path(args[2]);

FileInputFormat.setInputPaths(job, in);

FileOutputFormat.setOutputPath(job, out);

job.setMapperClass(IPcountMapper.class);

job.setCombinerClass(IPcountReducer.class);

job.setReducerClass(IPcountReducer.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.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 Projcheck(), args);

System.exit(res);

}

}

**6)**

**Top10Stocks**

**1 Top10**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Top10;

import java.io.IOException;

import java.net.URISyntaxException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

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.mapreduce.lib.output.TextOutputFormat;

/\*\*

\*

\* @author shruti

\*/

public class Top10 {

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

// TODO code application logic here

Configuration conf = new Configuration();

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

job.setJarByClass(Top10.class);

job.setMapperClass(Top10Mapper.class);

job.setMapOutputKeyClass(NullWritable.class);

job.setMapOutputValueClass(Text.class);

job.setReducerClass(Top10Reducer.class);

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(Text.class);

Path in = new Path(args[0]);

Path out = new Path(args[1]);

FileInputFormat.setInputPaths(job, in);

FileOutputFormat.setOutputPath(job, out);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

//job.setPartitionerClass(AvgPartitioner.class);

job.setNumReduceTasks(1);

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

}

}

**2 Top10Mapper**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Top10;

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author shruti

\*/

public class Top10Mapper extends Mapper <Object, Text, NullWritable, Text > {

private TreeMap<Double,Text> maprec = new TreeMap<Double, Text>();

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

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

String symbol = row[1].trim();

double price = Double.parseDouble(row[3].trim());

maprec.put(price,new Text(value));

if(maprec.size()>10){

maprec.remove(maprec.firstKey());

}

//context.write(yp, new DoubleWritable(price));

}

protected void cleanup(Context context) throws IOException, InterruptedException{

for(Text t: maprec.values()){

context.write(NullWritable.get(), t);

}

}

}

**3 Top10Reducer**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Top10;

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

/\*\*

\*

\* @author shruti

\*/

public class Top10Reducer extends Reducer <NullWritable, Text, NullWritable, Text> {

private TreeMap<Double,Text> maprec = new TreeMap<Double,Text>();

public void reduce(NullWritable key, Iterable <Text> values, Reducer.Context context) throws IOException, InterruptedException

{

for(Text val: values)

{

String row[] = values.toString().split(",");

maprec.put(Double.parseDouble(row[3].trim()),new Text(val));

if(maprec.size()>10)

maprec.remove(maprec.firstKey());

}

for(Text t:maprec.descendingMap().values())

context.write(NullWritable.get(),t);

}

}

**Hive queries:**

Create table average(symbol varchar(25), year int, price double(50,50));

Load data local inpath ‘/home/shruti/Downloads/average.txt’ into table average;

Select top 10 \* from average price desc;