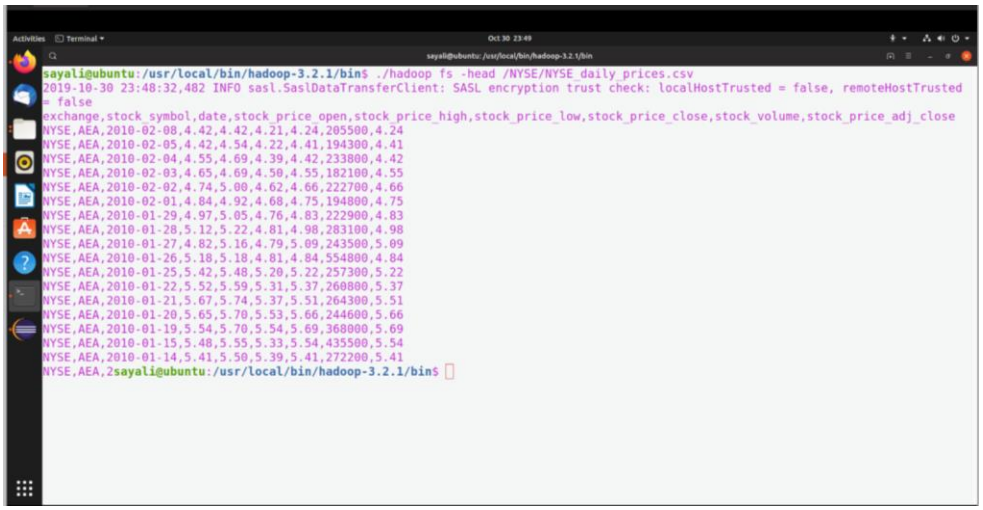
Advance map reduce on NYSE dataset

NYSE stock dataset looks like:



1] Create a Writable object that stores some fields from the the NYSE dataset to find

- the date of the max stock_volume

- the date of the min stock_volume

- the max stock_price_adj_close

Writable Class:

```
package sayali.Assignment4Part3;
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.WritableUtils;
public class MinMax implements Writable{
   String minStockVolDate;
   String maxStockVolDate;
   String maxStockPriceAdj;
   String minStock;
   String maxStock;
```

```
public String getMinStockVolDate() {
   return minStockVolDate;  }

public void setMinStockVolDate(String minStockVolDate) {
   this.minStockVolDate = minStockVolDate;  }

public String getMaxStockVolDate() {
   return maxStockVolDate;   }
 public void setMaxStockVolDate(String maxStockVolDate) {
   this.maxStockVolDate = maxStockVolDate;  }

public String getMaxStockPriceAdj() {
   return maxStockPriceAdj;  }

public void setMaxStockPriceAdj(String maxStockPriceAdj) {
   this.maxStockPriceAdj = maxStockPriceAdj;  }

public String getMinStock() {
   return minStock; }

public void setMinStock(String minStock) {
   this.minStock = minStock;  }

public String getMaxStock() {
   return maxStock; }

public void setMaxStock(String maxStock) {
   this.maxStock = maxStock; }
public void write(DataOutput d) throws IOException {

    WritableUtils.writeString(d, minStockVolDate);
    WritableUtils.writeString(d, maxStockVolDate);
    WritableUtils.writeString(d, maxStockPriceAdj);
    WritableUtils.writeString(d, minStock);
    WritableUtils.writeString(d, maxStock);  }
public void readFields(DataInput di) throws IOException {
   minStockVolDate = WritableUtils.readString(di);
   maxStockVolDate = WritableUtils.readString(di);
   maxStockPriceAdj = WritableUtils.readString(di);
   minStock = WritableUtils.readString(di);
   maxStock = WritableUtils.readString(di);  }

public String toString()
```

```
  { return (new
StringBuilder().append(minStockVolDate).append("\t").append(maxStockVolDate).toString());
  }  }
```

Mapper class:

```
package sayali.Assignment4Part3;
import java.io.IOException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class MinMaxMapper extends Mapper<Object, Text, Text, MinMax>{
  private Date minStockVolDate;
  private Date maxStockVolDate;
  private Float maxStockPriceAdj;
  private Integer minVol;
  private Integer maxVol;
  private Text stockSymbol = new Text();
  DateFormat formatter = new SimpleDateFormat("yyyy-mm-dd");
  public void map(Object key,Text value, Context context)throws IOException,
InterruptedException
  {

    String str[] = value.toString().split(",");
    try{
      minStockVolDate = formatter.parse(str[2]);
      maxStockVolDate = formatter.parse(str[2]);
      maxStockPriceAdj = Float.parseFloat(str[8]);
      minVol = Integer.parseInt(str[7]);
      maxVol = Integer.parseInt(str[7]);
      stockSymbol.set(str[1]);
      MinMax out1 = new MinMax();
      out1.setMinStockVolDate(minStockVolDate.toString());
      out1.setMaxStockVolDate(maxStockVolDate.toString());
      out1.setMaxStockPriceAdj(maxStockPriceAdj.toString());
      out1.setMaxStock(maxVol.toString());
      out1.setMinStock(minVol.toString());

      context.write(stockSymbol, out1);
    }
    catch(Exception e)
    {
    }
```

```
  } }
```

Reducer Class:

```
package sayali.Assignment4Part3;
import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MinMaxReducer extends Reducer<Text, MinMax, Text, Text> {
	public void reduce(Text key, Iterable<MinMax> values, Context context) throws
IOException, InterruptedException {

		String minStockVolDate = "";
		String maxStockVolDate = "";
		Float maxStockPrice = 0f;
		int minStock = Integer.MAX_VALUE;
		int maxStock = 0;
		for (MinMax m : values) {
			if (minStock > (Integer.parseInt(m.getMinStock()))) {
				minStockVolDate = m.getMinStockVolDate();
				minStock = Integer.parseInt(m.getMinStock());
			}
			if (maxStock < (Integer.parseInt(m.getMaxStock()))) {
				maxStockVolDate = m.getMaxStockVolDate();
				maxStock = Integer.parseInt(m.getMaxStock());
			}
			if (maxStockPrice < (Float.parseFloat(m.getMaxStockPriceAdj()))) {
				maxStockPrice = Float.parseFloat(m.getMaxStockPriceAdj());
			}
		}
		String temp = "\tMinStockVolumeDate\t"+minStockVolDate
+"\tMaxStockVolumeDate\t"+ maxStockVolDate +"\tMaxAdjClosePrice\t"+
String.valueOf(maxStockPrice);
		Text text = new Text(temp);
		context.write(key, text);
	}
}
```

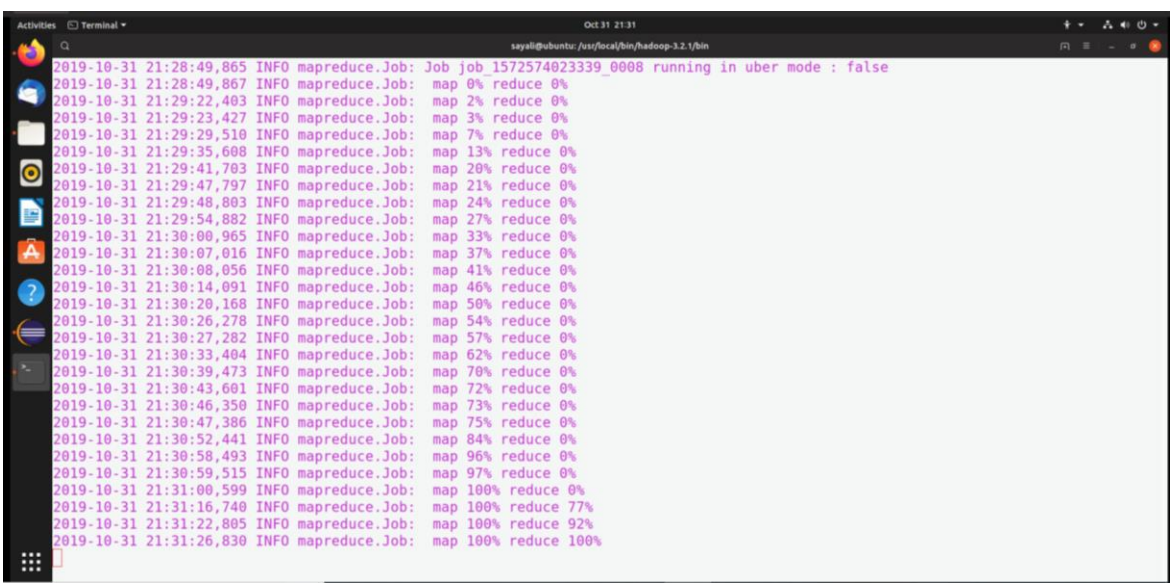
Main Class:

```
package sayali.Assignment4Part3;
import java.io.IOException;
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.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class HW4Part3 {
   public static void main(String[] args) throws IOException, InterruptedException,
ClassNotFoundException {
   Configuration conf = new Configuration();
   Job job = Job.getInstance(conf, "Stocks");
   job.setJarByClass(HW4Part3.class);
   job.setMapperClass(MinMaxMapper.class);
   job.setReducerClass(MinMaxReducer.class);
   job.setMapOutputKeyClass(Text.class);
   job.setMapOutputValueClass(MinMax.class);
   job.setOutputKeyClass(Text.class);
   job.setOutputValueClass(Text.class);
   FileInputFormat.addInputPath(job, new Path(args[0]));
   FileOutputFormat.setOutputPath(job, new Path(args[1]));
   System.exit(job.waitForCompletion(true) ? 0 : 1);
   }
}
```

OUTPUT:

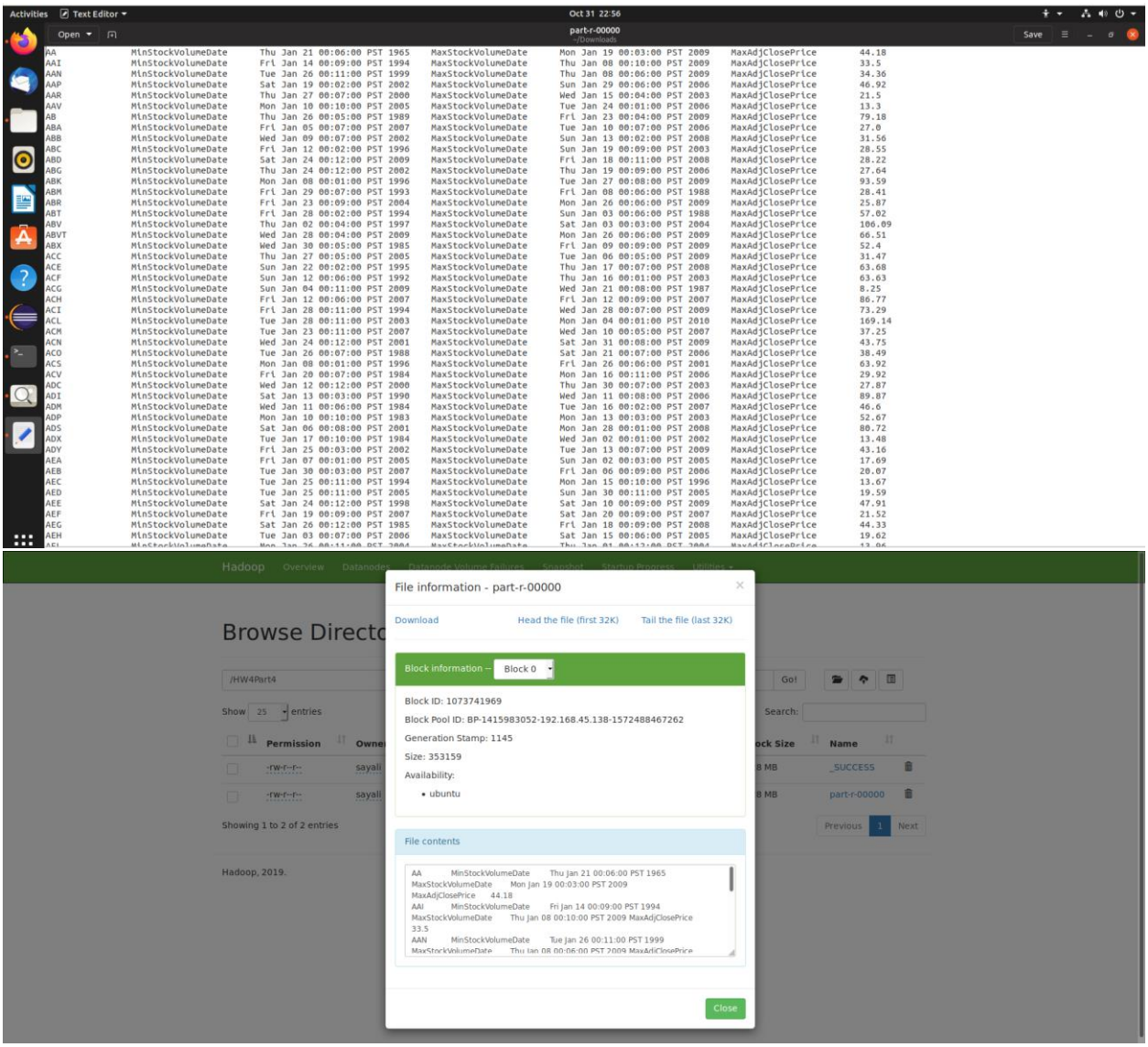
./hadoop jar /home/sayali/Desktop/HW4Part3.jar sayali.Assignment4Part3.HW4Part3 /NYSE/NYSE_daily_prices.csv /HW4Part3



```
2019-10-31 21:28:49,865 INFO mapreduce.Job: Job job_1572574023339_0008 running in uber mode : false
2019-10-31 21:28:49,867 INFO mapreduce.Job:  map 0% reduce 0%
2019-10-31 21:29:22,403 INFO mapreduce.Job:  map 2% reduce 0%
2019-10-31 21:29:23,427 INFO mapreduce.Job:  map 3% reduce 0%
2019-10-31 21:29:29,510 INFO mapreduce.Job:  map 7% reduce 0%
2019-10-31 21:29:35,608 INFO mapreduce.Job:  map 13% reduce 0%
2019-10-31 21:29:41,703 INFO mapreduce.Job:  map 20% reduce 0%
2019-10-31 21:29:47,797 INFO mapreduce.Job:  map 21% reduce 0%
2019-10-31 21:29:48,803 INFO mapreduce.Job:  map 24% reduce 0%
2019-10-31 21:29:54,882 INFO mapreduce.Job:  map 27% reduce 0%
2019-10-31 21:30:00,965 INFO mapreduce.Job:  map 33% reduce 0%
2019-10-31 21:30:07,016 INFO mapreduce.Job:  map 37% reduce 0%
2019-10-31 21:30:08,056 INFO mapreduce.Job:  map 41% reduce 0%
2019-10-31 21:30:14,091 INFO mapreduce.Job:  map 46% reduce 0%
2019-10-31 21:30:20,168 INFO mapreduce.Job:  map 50% reduce 0%
2019-10-31 21:30:26,278 INFO mapreduce.Job:  map 54% reduce 0%
2019-10-31 21:30:27,282 INFO mapreduce.Job:  map 57% reduce 0%
2019-10-31 21:30:33,404 INFO mapreduce.Job:  map 62% reduce 0%
2019-10-31 21:30:39,473 INFO mapreduce.Job:  map 70% reduce 0%
2019-10-31 21:30:43,601 INFO mapreduce.Job:  map 72% reduce 0%
2019-10-31 21:30:46,350 INFO mapreduce.Job:  map 73% reduce 0%
2019-10-31 21:30:47,386 INFO mapreduce.Job:  map 75% reduce 0%
2019-10-31 21:30:52,441 INFO mapreduce.Job:  map 84% reduce 0%
2019-10-31 21:30:58,493 INFO mapreduce.Job:  map 96% reduce 0%
2019-10-31 21:30:59,515 INFO mapreduce.Job:  map 97% reduce 0%
2019-10-31 21:31:00,599 INFO mapreduce.Job:  map 100% reduce 0%
2019-10-31 21:31:16,740 INFO mapreduce.Job:  map 100% reduce 77%
2019-10-31 21:31:22,805 INFO mapreduce.Job:  map 100% reduce 92%
2019-10-31 21:31:26,830 INFO mapreduce.Job:  map 100% reduce 100%
```

Time taken by this job reduce is:

31.26-28.49= 2 mins 77 seconds



2] Redo part 1] of this document, but cram multiple values (max stock_volume, min stock_volume, max stock_price_adj_close) into a Text object with some delimiter. Use a Combiner.

Mapper class:

```
package sayali.Assignment4Part4;
import java.io.IOException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
```

```
import java.util.Date;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MinMaxMapper extends Mapper<Object, Text, Text, Text> {
	private Date minStockVolDate;
	private Date maxStockVolDate;
	private Float maxStockPriceAdj;
	private Integer minVol;
	private Integer maxVol;
	Text textKey = new Text();
	Text textValue = new Text();
	DateFormat formatter = new SimpleDateFormat("yyyy-mm-dd");

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

		String str[] = value.toString().split(",");
		try {
			minStockVolDate = formatter.parse(str[2]);
			maxStockVolDate = formatter.parse(str[2]);
			maxStockPriceAdj = Float.parseFloat(str[8]);
			minVol = Integer.parseInt(str[7]);
			maxVol = Integer.parseInt(str[7]);
			textKey.set(str[1]);
			String v = String.valueOf(minVol) + "," + minStockVolDate + "," +
String.valueOf(maxVol) + ","
						+ maxStockVolDate + "," +
String.valueOf(maxStockPriceAdj);
			textValue.set(v);
			context.write(textKey, textValue);
		} catch (Exception e) {
		}
	}
}
```

Reducer class:

```
package sayali.Assignment4Part4;
import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class MinMaxReducer extends Reducer<Text, Text, Text, Text> {
```

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

		String minStockVolDate = "";
		String maxStockVolDate = "";
		Float maxStockPrice = 0f;
		int minStock = Integer.MAX_VALUE;
		int maxStock = 0;
		for (Text value : values) {
			String line = value.toString();
	String[] field = line.split(",");
			if (minStock > (Integer.parseInt(field[0]))) {
				minStockVolDate =field[1];
				minStock = Integer.parseInt(field[0]);
			}
			if (maxStock < (Integer.parseInt(field[2]))) {
				maxStockVolDate = field[3];
				maxStock = Integer.parseInt(field[3]);
			}
			if (maxStockPrice < (Float.parseFloat(field[4]))) {
				maxStockPrice = Float.parseFloat(field[4]);
			}
		}
		String temp = "\tMinStockVolumeDate\t"+minStockVolDate
+"\tMaxStockVolumeDate\t"+ maxStockVolDate +"\tMaxAdjClosePrice\t"+
String.valueOf(maxStockPrice);
		Text text = new Text(temp);
		context.write(key, text);
	}
}
```

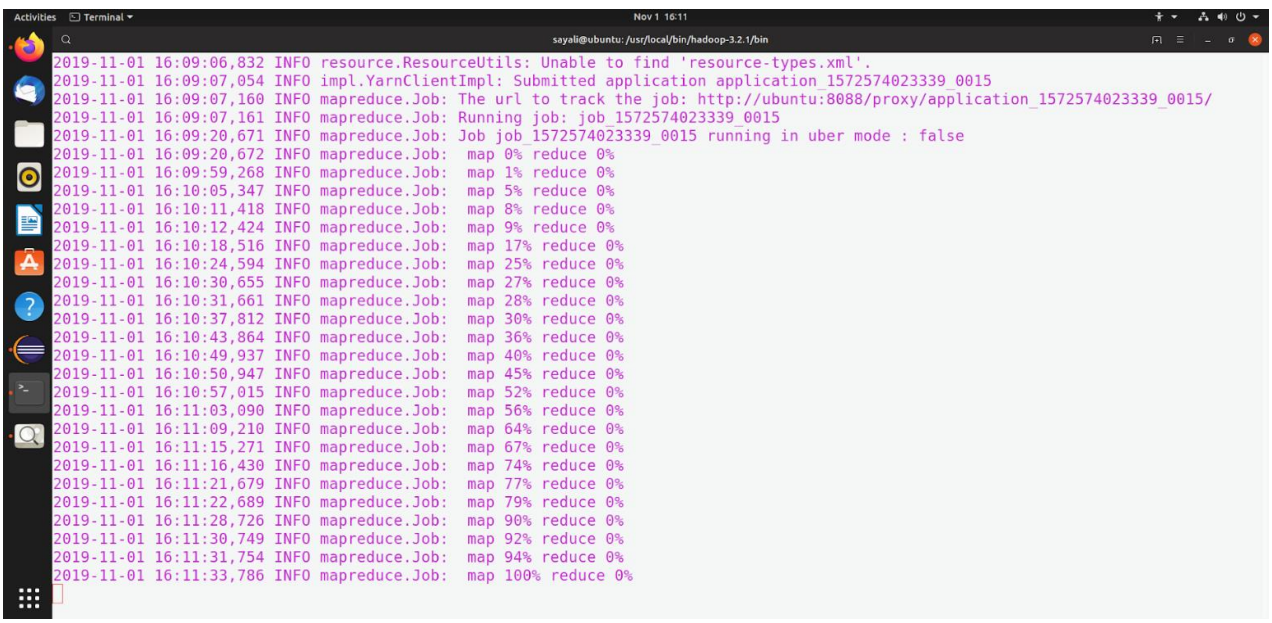
Main class:

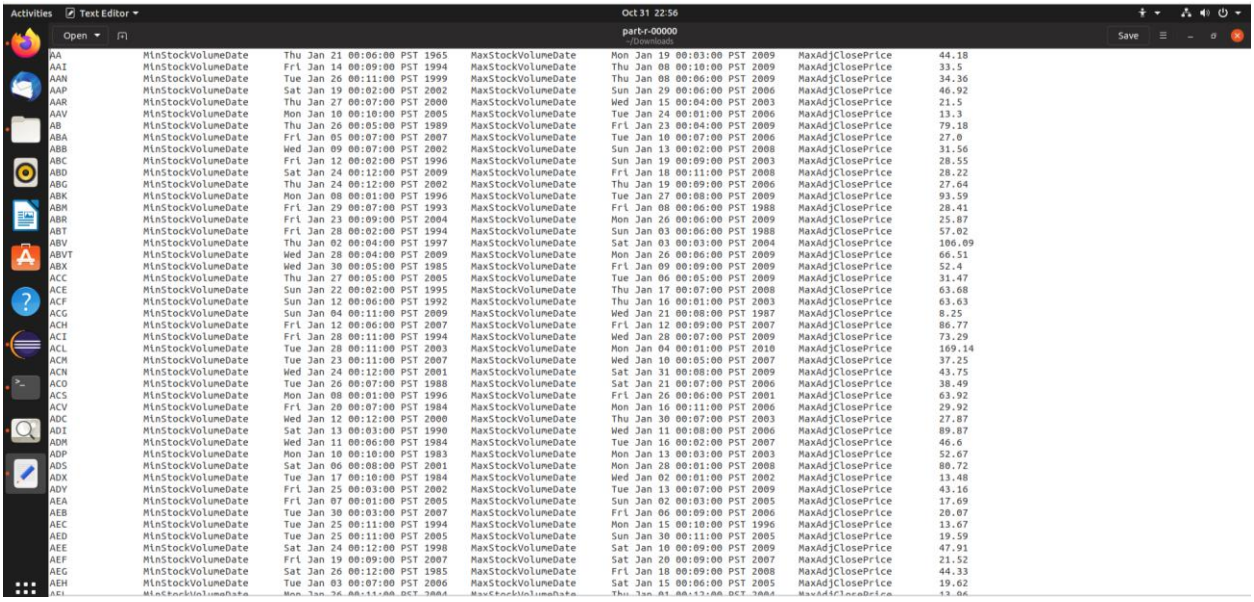
```
package sayali.Assignment4Part4;
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class HW4Part4 {
   public static void main(String[] args) throws IOException, InterruptedException,
ClassNotFoundException {
```

```
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "Stocks");
    job.setJarByClass(HW4Part4.class);
    job.setMapperClass(MinMaxMapper.class);
    job.setReducerClass(MinMaxReducer.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

OUTPUT:

./hadoop jar /home/sayali/Desktop/HW4Part4.jar sayali.Assignment4Part4.HW4Part4 /NYSE/NYSE_daily_prices.csv /HW4Part4





```
AA      MinStockVolumeDate  Thu Jan 21 00:06:00 PST 1965  MaxStockVolumeDate  Mon Jan 19 00:03:00 PST 2009  MaxAdjClosePrice  44.18
AAI     MinStockVolumeDate  Fri Jan 14 00:09:00 PST 1994  MaxStockVolumeDate  Thu Jan 08 00:10:00 PST 2009  MaxAdjClosePrice  33.5
AAN     MinStockVolumeDate  Tue Jan 26 00:11:00 PST 1999  MaxStockVolumeDate  Thu Jan 08 00:06:00 PST 2009  MaxAdjClosePrice  34.36
AAP     MinStockVolumeDate  Sat Jan 19 00:02:00 PST 2002  MaxStockVolumeDate  Sun Jan 29 00:06:00 PST 2006  MaxAdjClosePrice  46.92
AAR     MinStockVolumeDate  Thu Jan 27 00:07:00 PST 2000  MaxStockVolumeDate  Wed Jan 15 00:04:00 PST 2003  MaxAdjClosePrice  21.5
AAV     MinStockVolumeDate  Mon Jan 10 00:10:00 PST 2005  MaxStockVolumeDate  Tue Jan 24 00:01:00 PST 2006  MaxAdjClosePrice  13.3
AB      MinStockVolumeDate  Thu Jan 26 00:05:00 PST 1989  MaxStockVolumeDate  Fri Jan 23 00:04:00 PST 2009  MaxAdjClosePrice  79.18
ABA     MinStockVolumeDate  Fri Jan 05 00:07:00 PST 2007  MaxStockVolumeDate  Tue Jan 10 00:07:00 PST 2006  MaxAdjClosePrice  27.0
ABB     MinStockVolumeDate  Wed Jan 09 00:07:00 PST 2002  MaxStockVolumeDate  Sun Jan 13 00:02:00 PST 2008  MaxAdjClosePrice  31.56
ABC     MinStockVolumeDate  Fri Jan 12 00:02:00 PST 1996  MaxStockVolumeDate  Sun Jan 19 00:09:00 PST 2003  MaxAdjClosePrice  28.55
ABD     MinStockVolumeDate  Sat Jan 24 00:12:00 PST 2009  MaxStockVolumeDate  Fri Jan 18 00:11:00 PST 2008  MaxAdjClosePrice  28.22
ABG     MinStockVolumeDate  Thu Jan 24 00:12:00 PST 2002  MaxStockVolumeDate  Thu Jan 19 00:09:00 PST 2006  MaxAdjClosePrice  27.64
ABK     MinStockVolumeDate  Mon Jan 08 00:01:00 PST 1996  MaxStockVolumeDate  Tue Jan 27 00:08:00 PST 2009  MaxAdjClosePrice  93.59
ABM     MinStockVolumeDate  Fri Jan 29 00:07:00 PST 1993  MaxStockVolumeDate  Fri Jan 08 00:06:00 PST 1988  MaxAdjClosePrice  28.41
ABR     MinStockVolumeDate  Fri Jan 23 00:09:00 PST 2004  MaxStockVolumeDate  Mon Jan 26 00:06:00 PST 2009  MaxAdjClosePrice  25.87
ABT     MinStockVolumeDate  Fri Jan 28 00:02:00 PST 1994  MaxStockVolumeDate  Sun Jan 03 00:06:00 PST 1988  MaxAdjClosePrice  57.02
ABV     MinStockVolumeDate  Thu Jan 02 00:04:00 PST 1997  MaxStockVolumeDate  Sat Jan 03 00:03:00 PST 2004  MaxAdjClosePrice  106.09
ABVT    MinStockVolumeDate  Wed Jan 28 00:04:00 PST 2009  MaxStockVolumeDate  Mon Jan 26 00:06:00 PST 2009  MaxAdjClosePrice  66.51
ABX     MinStockVolumeDate  Wed Jan 30 00:05:00 PST 1985  MaxStockVolumeDate  Fri Jan 09 00:09:00 PST 2009  MaxAdjClosePrice  52.4
ACC     MinStockVolumeDate  Thu Jan 27 00:05:00 PST 2005  MaxStockVolumeDate  Tue Jan 06 00:05:00 PST 2009  MaxAdjClosePrice  31.47
ACE     MinStockVolumeDate  Sun Jan 22 00:02:00 PST 1995  MaxStockVolumeDate  Thu Jan 17 00:07:00 PST 2008  MaxAdjClosePrice  63.68
ACF     MinStockVolumeDate  Sun Jan 12 00:06:00 PST 1992  MaxStockVolumeDate  Thu Jan 16 00:01:00 PST 2003  MaxAdjClosePrice  63.63
ACG     MinStockVolumeDate  Sun Jan 04 00:11:00 PST 2009  MaxStockVolumeDate  Wed Jan 21 00:08:00 PST 1987  MaxAdjClosePrice  8.25
ACH     MinStockVolumeDate  Fri Jan 12 00:06:00 PST 2007  MaxStockVolumeDate  Fri Jan 12 00:09:00 PST 2007  MaxAdjClosePrice  86.77
ACI     MinStockVolumeDate  Fri Jan 28 00:11:00 PST 1994  MaxStockVolumeDate  Wed Jan 28 00:07:00 PST 2009  MaxAdjClosePrice  73.29
ACL     MinStockVolumeDate  Tue Jan 28 00:11:00 PST 2003  MaxStockVolumeDate  Mon Jan 04 00:01:00 PST 2010  MaxAdjClosePrice  169.14
ACM     MinStockVolumeDate  Tue Jan 23 00:11:00 PST 2007  MaxStockVolumeDate  Wed Jan 10 00:05:00 PST 2007  MaxAdjClosePrice  37.25
ACN     MinStockVolumeDate  Wed Jan 24 00:12:00 PST 2001  MaxStockVolumeDate  Sat Jan 31 00:08:00 PST 2009  MaxAdjClosePrice  43.75
ACO     MinStockVolumeDate  Tue Jan 26 00:07:00 PST 1988  MaxStockVolumeDate  Sat Jan 21 00:07:00 PST 2006  MaxAdjClosePrice  38.49
ACS     MinStockVolumeDate  Mon Jan 08 00:01:00 PST 1996  MaxStockVolumeDate  Fri Jan 26 00:06:00 PST 2001  MaxAdjClosePrice  63.92
ACV     MinStockVolumeDate  Fri Jan 20 00:07:00 PST 1984  MaxStockVolumeDate  Mon Jan 16 00:11:00 PST 2006  MaxAdjClosePrice  29.92
ADC     MinStockVolumeDate  Wed Jan 12 00:12:00 PST 2000  MaxStockVolumeDate  Thu Jan 30 00:07:00 PST 2003  MaxAdjClosePrice  27.87
ADI     MinStockVolumeDate  Sat Jan 13 00:03:00 PST 1990  MaxStockVolumeDate  Wed Jan 11 00:08:00 PST 2006  MaxAdjClosePrice  89.87
ADM     MinStockVolumeDate  Wed Jan 11 00:06:00 PST 1984  MaxStockVolumeDate  Tue Jan 16 00:02:00 PST 2007  MaxAdjClosePrice  46.6
ADP     MinStockVolumeDate  Mon Jan 10 00:10:00 PST 1983  MaxStockVolumeDate  Mon Jan 13 00:03:00 PST 2003  MaxAdjClosePrice  52.67
ADS     MinStockVolumeDate  Sat Jan 06 00:08:00 PST 2001  MaxStockVolumeDate  Mon Jan 28 00:01:00 PST 2008  MaxAdjClosePrice  80.72
ADX     MinStockVolumeDate  Tue Jan 17 00:10:00 PST 1984  MaxStockVolumeDate  Wed Jan 02 00:01:00 PST 2002  MaxAdjClosePrice  13.48
ADY     MinStockVolumeDate  Fri Jan 25 00:03:00 PST 2002  MaxStockVolumeDate  Tue Jan 13 00:07:00 PST 2009  MaxAdjClosePrice  43.16
AEA     MinStockVolumeDate  Fri Jan 07 00:01:00 PST 2005  MaxStockVolumeDate  Sun Jan 02 00:03:00 PST 2005  MaxAdjClosePrice  17.69
AEB     MinStockVolumeDate  Tue Jan 30 00:03:00 PST 2007  MaxStockVolumeDate  Fri Jan 06 00:09:00 PST 2006  MaxAdjClosePrice  20.07
AEC     MinStockVolumeDate  Tue Jan 25 00:11:00 PST 1994  MaxStockVolumeDate  Mon Jan 15 00:10:00 PST 1996  MaxAdjClosePrice  13.67
AED     MinStockVolumeDate  Tue Jan 25 00:11:00 PST 2005  MaxStockVolumeDate  Sun Jan 30 00:11:00 PST 2005  MaxAdjClosePrice  19.59
AEE     MinStockVolumeDate  Sat Jan 24 00:12:00 PST 1998  MaxStockVolumeDate  Sat Jan 10 00:09:00 PST 2009  MaxAdjClosePrice  47.91
AEF     MinStockVolumeDate  Fri Jan 19 00:09:00 PST 2007  MaxStockVolumeDate  Sat Jan 20 00:09:00 PST 2007  MaxAdjClosePrice  21.52
AEG     MinStockVolumeDate  Sat Jan 26 00:12:00 PST 1985  MaxStockVolumeDate  Fri Jan 18 00:09:00 PST 2008  MaxAdjClosePrice  44.33
AEH     MinStockVolumeDate  Tue Jan 03 00:07:00 PST 2006  MaxStockVolumeDate  Sat Jan 15 00:06:00 PST 2005  MaxAdjClosePrice  19.62
```

The time taken for this map reduce job can be calculated from the screenshot as:

12.01-9.20= 2 minutes 81 seconds

3] Determine the average stock_price_adj_close value by the year. Use a Writable object to pass count and local average in which a Reducer could be used as a Combiner.

Writable Object:

```
package sayali.Assignment4Part5;
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import java.util.Date;
import org.apache.hadoop.io.Writable;

public class Stock implements Writable {

	Integer count;
	Float stockValAvg;
	public Integer getCount() {
		return count;
	}
	public void setCount(Integer count) {
		this.count = count;
	}
	public Float getStockValAvg() {
```

```
            return stockValAvg;
        }
        public void setStockValAvg(Float stockValAvg) {
            this.stockValAvg = stockValAvg;
        }
        public void write(DataOutput d) throws IOException {
            d.writeInt(count);
            d.writeFloat(stockValAvg);
        }
        public void readFields(DataInput di) throws IOException {
            count = di.readInt();
            stockValAvg = di.readFloat();
        }
        @Override
        public String toString() {
            return stockValAvg + "\t";
        }
}
```

Mapper Class:

```
package sayali.Assignment4Part5;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class StockMapper extends Mapper<Object,Text,IntWritable,Stock>{

   IntWritable year = new IntWritable();
   Stock outStock = new Stock();
   @Override
   protected void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
      try{
      String str[] = value.toString().split(",");
      if(str[0].equals("exchange")) {
         return;
      }
      Float stockVal = Float.parseFloat(str[8]);
      year.set(Integer.parseInt(str[2].substring(0, 4)));
         outStock.setStockValAvg(stockVal);
         outStock.setCount(1);
         context.write(year,outStock);
      }
```

```
        catch(Exception e)
        {
            System.out.println("Non numeric value");
        }
    } }
```

Reducer Class:

```
package sayali.Assignment4Part5;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Reducer;
public class StockReducer extends Reducer<IntWritable,Stock,IntWritable,Stock>{

    Stock result = new Stock();
    @Override
    protected void reduce(IntWritable key, Iterable<Stock> values, Context context)
                    throws IOException, InterruptedException {

        int sum = 0;
        int count = 0;
        for(Stock val:values)
        {
            sum += val.getCount() * val.getStockValAvg();
            count += val.getCount();
        }
        result.setStockValAvg((float)sum/count);
        result.setCount(count);
        context.write(key, result);
       }
}
```

Main class:

```
package sayali.Assignment4Part5;
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.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class ClosePriceByYear {
```

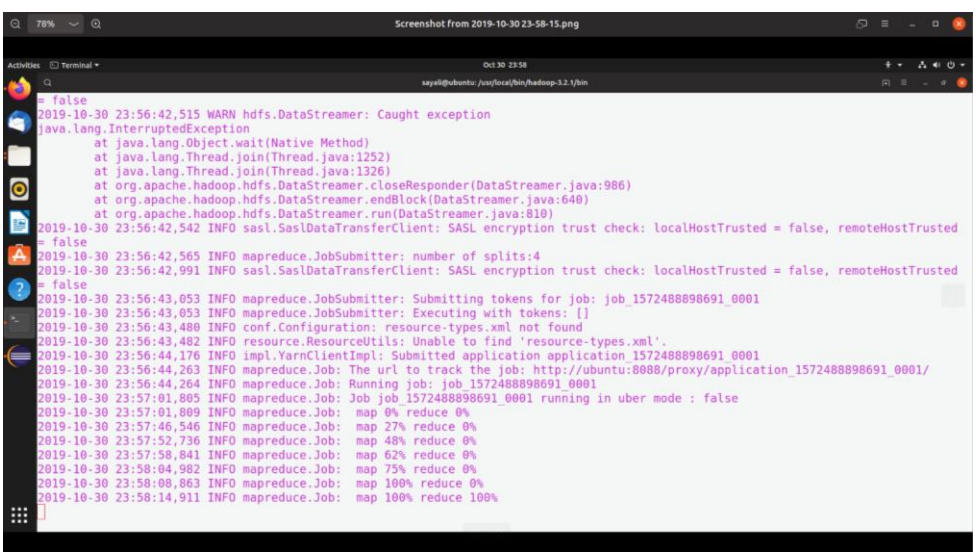
```
    public static void main(String[] args) throws IOException, InterruptedException,
ClassNotFoundException {

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "stockAvg");
        job.setJarByClass(ClosePriceByYear.class);
        job.setMapperClass(StockMapper.class);
        job.setMapOutputKeyClass(IntWritable.class);
        job.setMapOutputValueClass(Stock.class);
        job.setCombinerClass(StockReducer.class);
        job.setReducerClass(StockReducer.class);
        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(Stock.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job,new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

OUTPUT:

./hadoop jar /home/sayali/Desktop/closePriceByYear.jar sayali.Assignment4Part5.ClosePriceByYear /NYSE/NYSE_daily_prices.csv /closePriceByYear

Map reduce job:



Ubuntu - VMware Workstation 15 Player (Non-commercial use only)

Activities Text Editor Oct 31 12:16

Open ClosePriceByYear ~/Downloads Save

```
1962    0.50793654
1963    0.5707453
1964    0.7320158
1965    0.8630952
1966    1.0206431
1967    1.3099601
1968    1.3995575
1969    1.4408
1970    4.426419
1971    5.7530327
1972    7.574881
1973    7.590344
1974    6.2770753
1975    7.6837945
1976    9.96917
1977    11.32917
1978    11.552331
1979    12.784585
1980    9.1217375
1981    5.7117195
1982    3.155245
1983    4.682796
1984    4.3459597
1985    4.4323845
1986    5.764898
1987    6.719144
1988    5.522681
1989    6.1024566
1990    5.8779564
1991    6.6672087
1992    7.1358576
1993    8.165755
1994    8.977041
1995    9.883766
1996    11.941832
1997    15.030122
1998    16.770529
1999    17.00822
2000    18.49629
2001    18.749416
2002    18.220537
2003    18.874907
2004    23.50795
2005    26.197186
2006    29.039282
2007    32.928192
```

Plain Text Tab Width: 8 Ln 1, Col 1 INS

Type here to search 12:16 PM 10/31/2019