

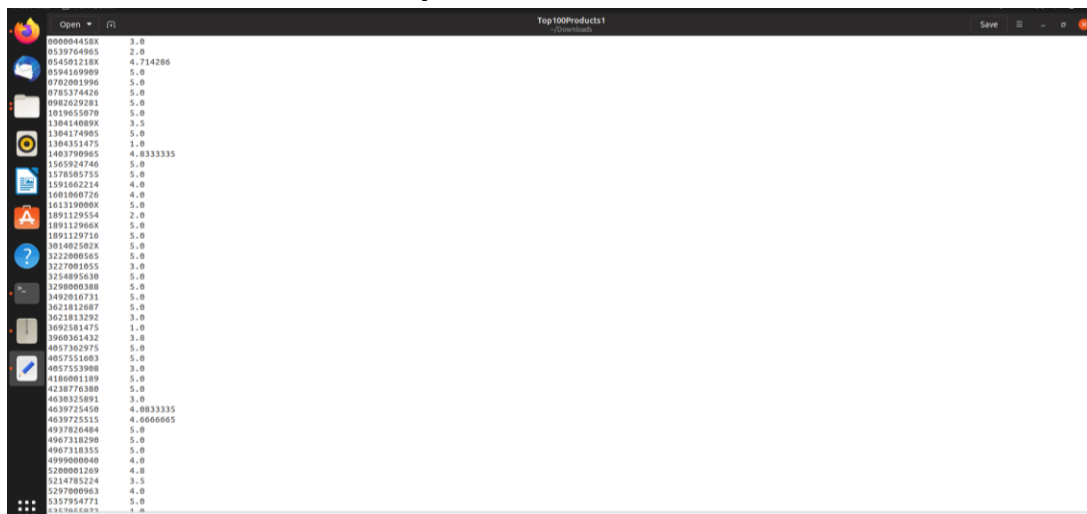
# Top 100 Products (Filtering Pattern)

This filtering pattern outputs the top products and the output file shows the ProductId and their corresponding rating. The output of first map reduce is average of ratings for each product. The output of second map reduce sort the product based on ratings and gives Top 100 Products.

## OUTPUT:

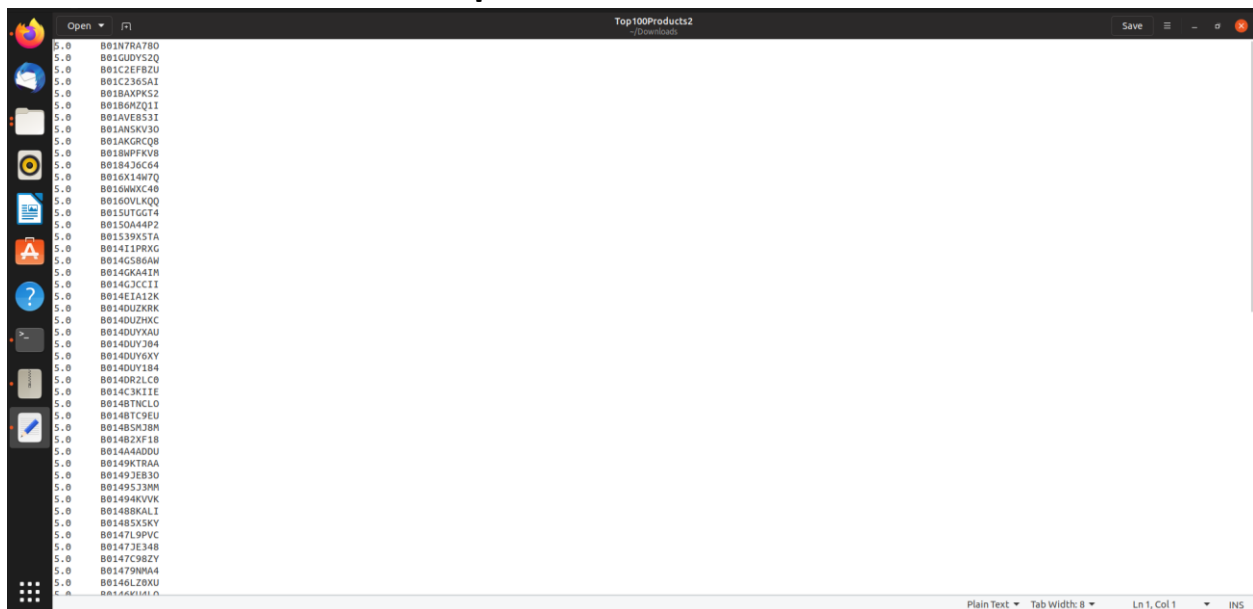
```
./hadoop jar /home/sayali/Desktop/Top100Products.jar sayali.AmazonReviews.Top100Products
/AmazonReviews/AmazonReviews.tsv /Top100Products1/Top100Products2
```

## OUTPUT of the first Map reduce:



000004420X	3.0
0139704965	2.0
054501218X	4.714286
0594109900	5.0
0702001996	5.0
0705374426	5.0
0910629281	5.0
1019655070	5.0
110414089X	3.5
1104174095	5.0
1104351475	1.0
1403790965	4.833333
1505924746	5.0
1578505755	5.0
1591602214	4.0
1601060726	4.0
161319000X	5.0
1891129554	2.0
189112966X	5.0
1891129716	5.0
191402502X	5.0
3220000565	5.0
3227001055	3.0
3254895030	5.0
3290000388	5.0
3492010731	5.0
3621812687	5.0
3621813292	3.0
3692001475	1.0
3900361432	3.0
4057362975	5.0
4057551603	5.0
4057553908	3.0
4100001189	5.0
4238776380	5.0
4630325091	3.0
4639725450	4.083333
4639725515	4.666665
4937826484	5.0
4967310290	5.0
4967318355	5.0
4999000040	4.0
5200001200	4.0
5214785224	3.5
5297000963	4.0
5179544771	5.0
0137066075	1.0

## OUTPUT of the second Map reduce:



5.0	001N7RA780
5.0	001G0U5Y2Q
5.0	001C2EFBZU
5.0	001C236SAI
5.0	001BAXPK52
5.0	001B6KZQ11
5.0	001AVEB53I
5.0	001ANSKV30
5.0	001AKGRQ8
5.0	001BWPFX8
5.0	0018436C64
5.0	0016X14W7Q
5.0	0010AMXC40
5.0	00160VKQ0
5.0	001SUTGGT4
5.0	00150A44P2
5.0	001539X5TA
5.0	0014I1PRXG
5.0	0014G5B6AW
5.0	0014GKA4IM
5.0	0014G2CC1I
5.0	0014E1A12K
5.0	0014DUZKRK
5.0	0014DUZHXC
5.0	0014DUYXAU
5.0	0014DUY304
5.0	0014DUY6XY
5.0	0014DUY184
5.0	00140R2LC0
5.0	0014C8K1IE
5.0	001487NCL0
5.0	001487C9EU
5.0	001485K3M
5.0	001402XF18
5.0	0014A4AD0U
5.0	00149KTRAA
5.0	001493EB30
5.0	00149533M
5.0	001494KVVK
5.0	001488KALI
5.0	001485K5XY
5.0	00147L9PVC
5.0	001473E348
5.0	00147C98ZY
5.0	001479M044
5.0	00146LZ0XU
5.0	00146K01810

# Map reduce Code

```
package sayali.AmazonReviews;
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;
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;
```

## Mapper 1:

```
public class Top100ProductsMapper1 extends Mapper<LongWritable, Text, Text, FloatWritable>
{
    private Text text = new Text();
    private FloatWritable score = new FloatWritable();
    protected void map(LongWritable key, Text value, Context context) throws IOException,
    InterruptedException {
        if(key.get() == 0){
            return;
        }
        else{
            String[] line = value.toString().split("\\t");
            String productId = line[3].trim();
            float ratingValue = Float.valueOf(line[7].trim());
            text.set(productId);
            score.set(ratingValue);
            context.write(text, score);
        }
    }
}
```

## Reducer 1:

```
public class Top100ProductsReducer1 extends Reducer<Text, FloatWritable, Text, FloatWritable>{
    private FloatWritable result = new FloatWritable();

    @Override
    protected void reduce(Text key, Iterable<FloatWritable> values, Context context) throws IOException,
    InterruptedException{
        float sum = 0;
```

```

    int count = 0;
    for(FloatWritable val:values){
        sum+=val.get();
        count = count+1;
    }
    float average = sum/count;
    result.set(average);
    context.write(key,result);
}
}

```

## Mapper 2:

```

public class Top100ProductsMapper2 extends Mapper<LongWritable, Text, FloatWritable, Text> {
    public void map(LongWritable key, Text value, Context context) {

        String[] row = value.toString().split("\\t");
        String productId = row[0].trim();
        float prodRating = Float.parseFloat(row[1].trim());

        try {
            Text id = new Text(productId);
            FloatWritable prodRate = new FloatWritable(prodRating);
            context.write(prodRate, id); }
        catch (Exception e) {
        } } }

```

## Reducer 2:

```

public class Top100ProductsReducer2 extends Reducer<FloatWritable, Text, FloatWritable, Text> {
    int count = 0;
    public void reduce(FloatWritable key, Iterable<Text> value, Context context) throws
IOException, InterruptedException {
        for (Text val : value) {

            if (count < 100) {
                context.write(key, val);
            }
            count++;
        } }
    }
}

```

## Comparator:

```

public class GroupComparator extends WritableComparator {

    protected GroupComparator() {
        super(FloatWritable.class, true);
    }
}

```

```

    }

    public int compare(WritableComparable w1, WritableComparable w2) {
        FloatWritable cw1 = (FloatWritable) w1;
        FloatWritable cw2 = (FloatWritable) w2;
        int result = cw1.get() < cw2.get() ? 1 : cw1.get() == cw2.get() ? 0 : -1;
        return result;
    }
}

```

## MAIN CLASS:

```

public class Top100Products {
    public static void main(String[] args) throws IOException, InterruptedException,
        ClassNotFoundException {

        Configuration conf1 = new Configuration();
        Job job1 = Job.getInstance(conf1, "Top 100 Products");
        job1.setJarByClass(Top100Products.class);
        job1.setMapperClass(Top100ProductsMapper1.class);
        job1.setMapOutputKeyClass(Text.class);
        job1.setMapOutputValueClass(FloatWritable.class);
        job1.setReducerClass(Top100ProductsReducer1.class);
        job1.setOutputKeyClass(Text.class);
        job1.setOutputValueClass(FloatWritable.class);
        FileInputFormat.addInputPath(job1, new Path(args[0]));
        FileOutputFormat.setOutputPath(job1, new Path(args[1]));

        boolean complete = job1.waitForCompletion(true);
        Configuration conf2 = new Configuration();
        if (complete) {
            Job job2 = Job.getInstance(conf2, "Top 100 Products");
            job2.setJarByClass(Top100Products.class);
            job2.setMapperClass(Top100ProductsMapper2.class);
            job2.setMapOutputKeyClass(FloatWritable.class);
            job2.setMapOutputValueClass(Text.class);
            job2.setSortComparatorClass(GroupComparator.class);
            job2.setReducerClass(Top100ProductsReducer2.class);
            job2.setOutputKeyClass(FloatWritable.class);
            job2.setOutputValueClass(Text.class);
            FileInputFormat.addInputPath(job2, new Path(args[1]));
            FileOutputFormat.setOutputPath(job2, new Path(args[2]));

            System.exit(job2.waitForCompletion(true) ? 0 : 1);
        }
    }
}

```