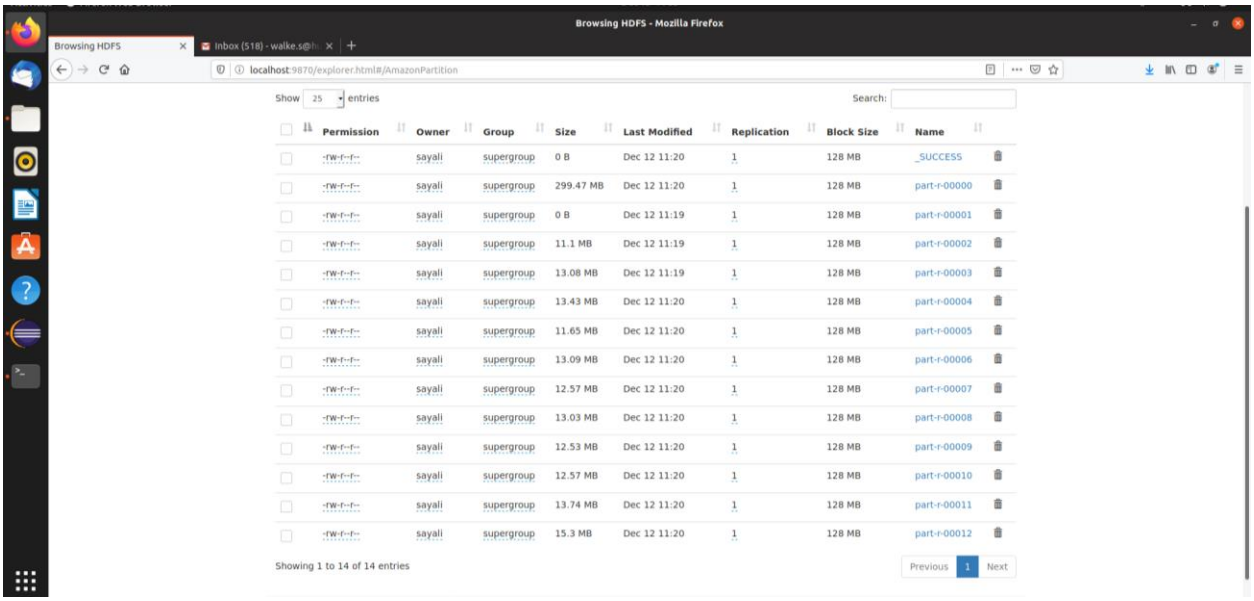
Amazon Partition by Day

(Organization Pattern)

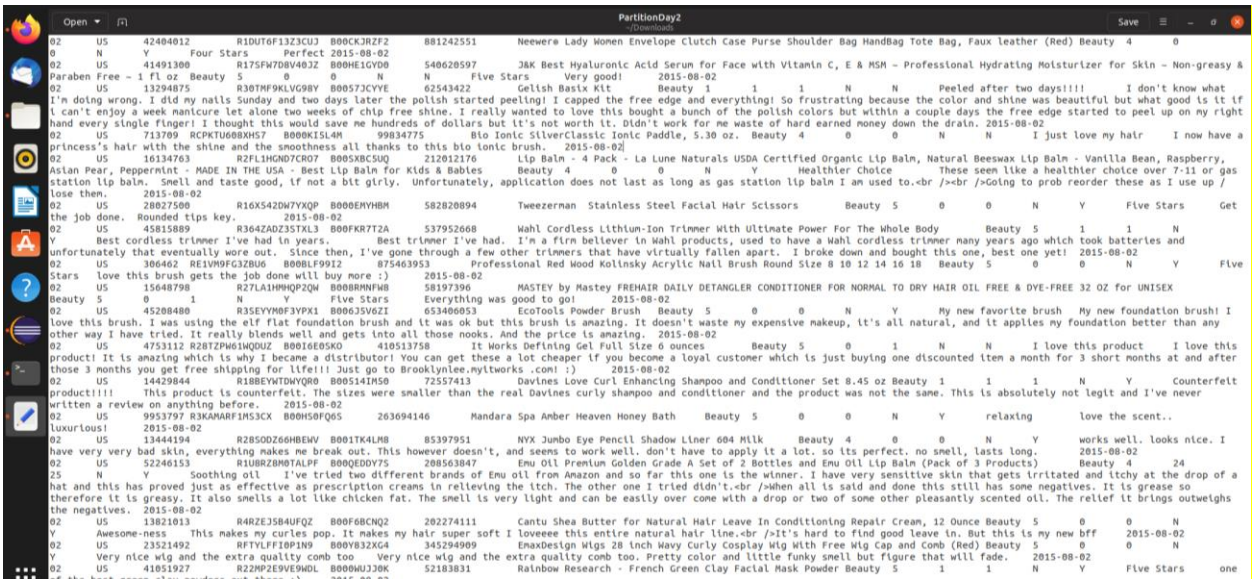
Partitioning is a pattern where data can be partitioned based on the number of partitioners mentioned in the Custom partitioner. It can be used to separate the dataset into different files based on the number of partitions. I have separated the Amazon Review Dataset according to the day from 2015-08-01 to 2015-08-12. This gives different 13 folders as shown in figure.

OUTPUT:

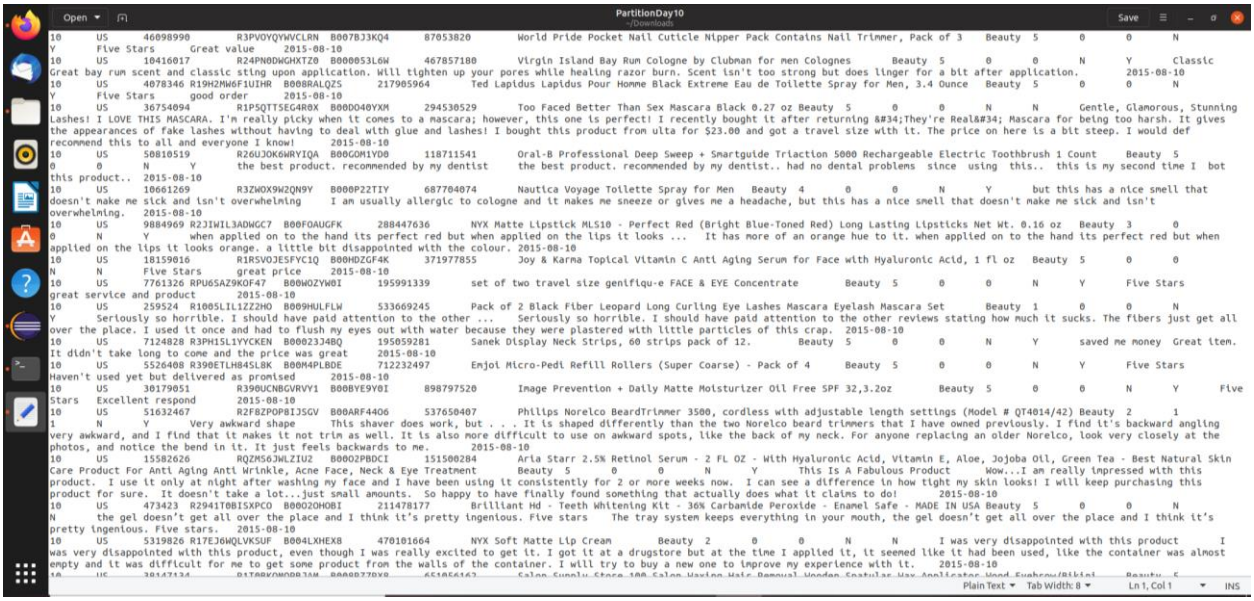
./hadoop jar /home/sayali/Desktop/PartitionByDay.jar sayali.AmazonReviews6.AmazonPartitionByDay /AmazonReviews/AmazonReviews.tsv /AmazonPartition



File that contains reviews from 2015-08-02



File that contains reviews from 2015-08-10



Map Reduce Code

Mapper:

```
public class DayPartitionMapper extends Mapper<LongWritable, Text, Text, Text> {
        private Text inputRec = new Text();
        private Text day = new Text();
protected void map(LongWritable key, Text value, Mapper.Context context) throws IOException,
InterruptedException {
                if (key.get() == 0) {
                        return;
                }
                String[] line = value.toString().split("\\t");
                String[] dayPart = line[14].split("-");
                String dayVal = dayPart[2].trim();
                day.set(dayVal);
                inputRec.set(value);
                context.write(day, inputRec);
        }
}
```

Reducer:

```
public class DayPartitionReducer extends Reducer<Text, Text, Text, NullWritable> {
   protected void reduce(Text key, Iterable<Text> values, Reducer.Context context)
                throws IOException, InterruptedException{
     for(Text t: values){
        context.write(t, NullWritable.get());
     }
  }
}
```

Partitioner:

```
public class DayPartitionPartitioner extends Partitioner<Text, Text> {
Text yrval = new Text();
        @Override
        public int getPartition(Text key, Text value, int numPartitions) {
                int n = 1;
                if (numPartitions == 0) {
                        return 0;
                } else if (key.equals(("01"))) {
                        return n % numPartitions;
                } else if (key.equals(new Text("02"))) {
                        return 2 % numPartitions;
                } else if (key.equals(new Text("03"))) {
                        return 3 % numPartitions;
```

```
		} else if (key.equals(new Text("04"))) {
			return 4 % numPartitions;
		} else if (key.equals(new Text("05"))) {
			return 5 % numPartitions;
		} else if (key.equals(new Text("06"))) {
			return 6 % numPartitions;
		} else if (key.equals(new Text("07"))) {
			return 7 % numPartitions;
		} else if (key.equals(new Text("08"))) {
			return 8 % numPartitions;
		} else if (key.equals(new Text("09"))) {
			return 9 % numPartitions;
		} else if (key.equals(new Text("10"))) {
			return 10 % numPartitions;
		} else if (key.equals(new Text("11"))) {
			return 11 % numPartitions;
		} else if (key.equals(new Text("12"))) {
			return 12 % numPartitions;
		}
		else {
			return 13 % numPartitions;
		}
	}
}
```

Driver:

```
public class AmazonPartitionByDay {
public static void main(String[] args) throws IOException, InterruptedException,
ClassNotFoundException {
		Configuration conf = new Configuration();
		Job job = Job.getInstance(conf, "Partitioning");
		job.setJarByClass(AmazonPartitionByDay.class);
		job.setMapperClass(DayPartitionMapper.class);
		job.setMapOutputKeyClass(Text.class);
		job.setMapOutputValueClass(Text.class);
		// Custom Partitioner::
		job.setPartitionerClass(DayPartitionPartitioner.class);
		job.setReducerClass(DayPartitionReducer.class);
		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(NullWritable.class);
		job.setNumReduceTasks(13);
		FileInputFormat.addInputPath(job, new Path(args[0]));
		FileOutputFormat.setOutputPath(job, new Path(args[1]));
		System.exit(job.waitForCompletion(true) ? 0 : 1);
	}
}
```