```
import java.io.IOException;
import java.util.ArrayList;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.lib.ChainMapper;
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.KeyValueTextInputFormat;
import org.apache.hadoop.mapreduce.lib.jobcontrol.ControlledJob;
import org.apache.hadoop.mapreduce.lib.jobcontrol.JobControl;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Frequent_Itemsets
{
   public static ArrayList<String> FrequentItem= new ArrayList<String>();
   public static class ItemCountMapper1 extends Mapper < LongWritable, Text, Text, IntWritable >
   {
      private final static IntWritable count = new IntWritable( 1);
      private Text Item = new Text();
      @Override
      public void map( LongWritable key, Text value, Context context) throws IOException,
InterruptedException
      {
         String Temp[] = value.toString().split("\\t");
         Item.set( Temp[1]);
         context.write( Item, count);
      }
   }
   public static class Reduce
   extends
   Reducer < Text, IntWritable, Text, IntWritable >
   {
      private IntWritable result = new IntWritable();
      @Override
      public void reduce( Text key, Iterable < IntWritable > values, Context context) throws IOException,
InterruptedException
      {
         int sum = 0;
```

```
        for (IntWritable val : values)
        {
           sum += val.get();
        }
        if(sum>=70)
        {
           FrequentItem.add(String.valueOf(key));
           result.set( sum);
           context.write( key, result);
           System.out.println("FrequentItem: "+FrequentItem);
        }
     }
  }
  public static class ItemCountMapper2 extends Mapper <  LongWritable, Text, Text, IntWritable >
  {

     private final static IntWritable count = new IntWritable( 1);
     private Text Item = new Text();
     public static Integer transactionNumber = 1;
     public static String transactionSet = " ";
     @Override
     public void map( LongWritable key, Text value, Context context) throws IOException,
InterruptedException
     {
        String T = FrequentItem.toString().replace("[", "");
        T = T.replace("]", "");
        System.out.println("T: "+T);
        String Temp[] = value.toString().split("\\t");
        if(Temp[0].equalsIgnoreCase(String.valueOf(transactionNumber)))
        {
           transactionSet = transactionSet +","+ Temp[1];
        }
        else
        {
           transactionNumber++;
           System.out.println("transactionSet: "+transactionSet);
           String itemsInTrx[] = transactionSet.split(",");
           String FrequentItemsSetInTrx = " ";
           for(int i = 1; i<=itemsInTrx.length-1; i++)
           {
              String FreqItemset[] = T.split(",");
              for(int j = 1; j<=FreqItemset.length-1; j++)
              {
                 if(FreqItemset[j].trim().equalsIgnoreCase(itemsInTrx[i]))
                    FrequentItemsSetInTrx = FrequentItemsSetInTrx + "," +itemsInTrx[i];
              }
           }
           System.out.println("FrequentItemsSetInTrx: "+FrequentItemsSetInTrx);
```

```
            String F[] = FrequentItemsSetInTrx.split(",");
            for(int k = 1; k<= (F.length-1); k++)
            {
               for(int l = k+1;l<=(F.length-1);l++)
               {
                  if((F[k].equalsIgnoreCase(F[l])==false))
                  {
                     Item.set(F[k]+","+F[l]);
                     System.out.println("Item: "+Item);
                     context.write( Item, count);
                  }
               }
            }
            transactionSet = "";
         }
      }
   }

   public static void main( String[] args) throws Exception
   {
      Configuration conf = new Configuration();
      Job job = Job.getInstance( conf, "FISH");

      job.setJarByClass(Frequent_Itemsets.class);

      FileInputFormat.addInputPath( job, new Path("input"));
      FileOutputFormat.setOutputPath( job, new Path("Singletons"));
      job.setMapperClass( ItemCountMapper1.class);
      job.setCombinerClass( Reduce.class);
      job.setReducerClass( Reduce.class);

      job.setOutputKeyClass( Text.class);
      job.setOutputValueClass( IntWritable.class);

      job.waitForCompletion(true);

      Configuration conf2 = new Configuration();
      Job job2 = Job.getInstance( conf2, "FISH");

      job2.setJarByClass(Frequent_Itemsets.class);

      FileInputFormat.addInputPath( job2, new Path("input"));
      FileOutputFormat.setOutputPath( job2, new Path("output"));
      job2.setMapperClass( ItemCountMapper2.class);
      job2.setCombinerClass( Reduce.class);
      job2.setReducerClass( Reduce.class);

      job2.setOutputKeyClass( Text.class);
```

```
        job2.setOutputValueClass( IntWritable.class);

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

    }
}
```