expected voters:  
  
public class Educational {  
  
    public static class MyMapper extends Mapper<LongWritable,Text,IntWritable,IntWritable>   
    {  
        public void map(LongWritable inkey,Text inval,Context context) throws IOException, InterruptedException  
        {  
            try  
           {  
            String year=context.getConfiguration().get("aaa");  
            int y=Integer.parseInt(year);  
              
              
            String line=inval.toString();  
            String arr[]=line.split(",");  
            int age1=0;  
             
              
            for (String str : arr)  
            {  
                String age=arr[0];  
                age1=Integer.parseInt(age);  
                  
            }  
             
            context.write(new IntWritable(y),new IntWritable(age1));  
            }  
            catch(Exception e)  
            {  
                e.printStackTrace();  
            }  
        }  
  
    }  
  
    public static class MyReduce extends Reducer<IntWritable,IntWritable,Text,IntWritable>{  
  
          
        protected void reduce(IntWritable key, Iterable<IntWritable> values,Context context)  
                throws IOException, InterruptedException {  
            int counter=18;  
            int agekey=Integer.parseInt(String.valueOf(key));  
            int reqage=counter-agekey;  
            int count=0;  
              
            for(IntWritable value:values)  
            {  
                int inval=Integer.parseInt(String.valueOf(value));  
                if(inval==reqage)  
                {  
                    count++;  
                }  
                  
            }  
            context.write(new Text("Expected voters are : "), new IntWritable(count));  
              
        }  
    }  
  
      
    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException   
    {  
        Configuration cfg=new Configuration();  
        System.out.println("Enter how many years expected for new Voter :");  
        Scanner s=new Scanner(System.in);  
        int str=s.nextInt();  
          
    //    System.out.println(str);  
          
        cfg.setInt("aaa", str);  
          
    //    System.out.println("correct");  
          
        Job job=new Job(cfg,"Educational");  
          
        job.setJarByClass(Educational.class);  
          
        job.setMapperClass(MyMapper.class);  
        job.setReducerClass(MyReduce.class);  
        job.setNumReduceTasks(1);  
        job.setMapOutputKeyClass(IntWritable.class);  
        job.setMapOutputValueClass(IntWritable.class);  
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        FileOutputFormat.setOutputPath(job, new Path(args[1]));  
        System.exit(job.waitForCompletion(true) ? 0 : 1);  
      
    }  
}

expected citizens:  
  
public class SeniorCiti {  
  
    public static class MyMapper extends Mapper<LongWritable,Text,IntWritable,IntWritable>{  
  
        @Override  
        protected void map(LongWritable key, Text value,Context context)  
                throws IOException, InterruptedException {  
            String year=context.getConfiguration().get("year");  
            int year1=Integer.parseInt(year);  
              
            String records=value.toString();  
            String recordparts[]=records.split(",");  
            String age=recordparts[0];  
            int age1=Integer.parseInt(age);  
              
            context.write(new IntWritable(year1),new IntWritable(age1));  
              
        }  
  
    }  
      
    public static class MyReducer extends Reducer<IntWritable,IntWritable,Text,IntWritable>{  
  
        @Override  
        protected void reduce(IntWritable key, Iterable<IntWritable> values,Context context)  
                throws IOException, InterruptedException {  
            int counter=60;  
            int agekey=Integer.parseInt(String.valueOf(key));  
            int reqage=counter-agekey;  
              
              
           int count=0;  
              
            for(IntWritable value:values)  
            {  
                int inval=Integer.parseInt(String.valueOf(value));  
                if(inval==reqage)  
                {  
                    count++;  
                }  
                  
            }  
              
              
            context.write(new Text("Expected Sr.Citizen : "), new IntWritable(count));  
              
        }  
  
    }  
      
    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {  
        Configuration conf = new Configuration();  
        System.out.println("Enter the No. of Year: ");  
        Scanner sc=new Scanner(System.in);  
        int n=sc.nextInt();  
          
        conf.setInt("year",n);      
          
        Job job = new Job(conf, "Map Reduce Search Txn by Arg");  
        job.setJarByClass(SeniorCiti.class);  
        job.setMapperClass(MyMapper.class);  
        job.setReducerClass(MyReducer.class);  
        job.setMapOutputKeyClass(IntWritable.class);  
        job.setMapOutputValueClass(IntWritable.class);  
        job.setNumReduceTasks(1);  
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        FileOutputFormat.setOutputPath(job, new Path(args[1]));  
        System.exit(job.waitForCompletion(true) ? 0 : 1);       
  
    }  
  
}

education category wise employed   
  
public class EduEmp {  
  
    public static class MapClass extends Mapper<LongWritable,Text,Text,LongWritable>  
    {  
        public void map(LongWritable key, Text value, Context context)  
        {  
           try{  
                String[] str= value.toString().split(",");  
                String edu = str[1].toString();  
                long ww = Long.parseLong(str[9]);  
                context.write(new Text(edu), new LongWritable(ww));  
            }  
            catch(Exception e)  
            {  
                System.out.println(e.getMessage());  
            }  
        }  
          
    }  
      
      
    public static class ReduceClass extends Reducer<Text, LongWritable, Text, Text>  
    {  
        private Text result = new Text();  
          
        public void reduce(Text key, Iterable<LongWritable> values,Context context) throws IOException, InterruptedException {  
                     long count = 0,count1 = 0;  
               for (LongWritable val : values)   
               {  
                if(val.get() > 0)  
                {  
                    count++;  
                }  
                else  
                {  
                    count1++;  
                }  
               }  
               String str = String.format("%d\t%d", count,count1);  
               result.set(str);  
               context.write(key, result);  
           }  
    }  
      
    public static void main(String args[]) throws Exception  
    {  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf, "edu emp");  
        job.setJarByClass(EduEmp.class);  
        job.setMapperClass(MapClass.class);  
        job.setReducerClass(ReduceClass.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);  
    }      
}

no of widow candidates  
  
public class SW1 {  
      
    public static class MapClass extends Mapper<LongWritable,Text,Text,Text>  
    {  
        protected void map(LongWritable key, Text value, Context context) throws java.io.IOException, InterruptedException  
        {  
              
                String[] str= value.toString().split(",");  
                String ms = str[2];  
                String gen = str[3].toString();  
                  
                context.write(new Text(gen), new Text(ms));  
  
        }  
          
    }  
      
    public static class ReduceClass extends Reducer<Text, Text, Text, Text>  
    {  
        private Text result = new Text();  
          
        public void reduce(Text key, Iterable<Text> values,Context context) throws IOException, InterruptedException   
        {  
                 
               long count = 0;  
                 
               for (Text val : values)   
               {  
                   if(key.equals("Female") && val.equals("Widowed"))  
                   {  
                        count++;   
                   }  
               }  
                 
               String str = String.valueOf(count);  
               result.set(str);  
               context.write(key, result);  
          }  
    }  
      
    public static void main(String args[]) throws Exception  
    {  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf, "num of female widows");  
        job.setJarByClass(SW1.class);  
        job.setMapperClass(MapClass.class);  
        job.setReducerClass(ReduceClass.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);  
    }      
  
}

parents gender wise count  
  
public class SW2 {  
  
    public static class MapClass extends Mapper<LongWritable,Text,Text,Text>  
    {  
        protected void map(LongWritable key, Text value, Context context) throws java.io.IOException, InterruptedException  
        {  
              
                String[] str= value.toString().split(",");  
                String parents = str[6].toString();  
                String gen = str[3];  
                String edu = str[1];  
                String op = gen + "," + edu;  
                  
                context.write(new Text(parents), new Text(op));  
  
        }  
          
    }  
      
    public static class ReduceClass extends Reducer<Text, Text, Text, Text>  
    {  
        private Text result = new Text();  
          
        public void reduce(Text key, Iterable<Text> values,Context context) throws IOException, InterruptedException   
        {  
                 
               long count = 0,count1 =0;  
                 
               for (Text val : values)   
               {  
                   String[] st = val.toString().split(",");  
                   String gen = st[0];  
                   String edu = st[1];  
                     
                   if(gen.equals("Female") && edu.equals("Children"))  
                   {  
                        count++;   
                   }  
                   else if(gen.equals("Male") && edu.equals("Children"))  
                   {  
                       count1++;  
                   }  
               }  
                 
               String str = String.format("%d\t%d", count,count1);  
                 
               result.set(str);  
               context.write(key, result);  
          }  
    }  
      
    public static void main(String args[]) throws Exception  
    {  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf, "parents gender count");  
        job.setJarByClass(SW2.class);  
        job.setMapperClass(MapClass.class);  
        job.setReducerClass(ReduceClass.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);  
    }         
      
}