### Starting Letter Count [12 points]

AKSHTA (UIN: 652062814)

LAAVANYA GANESH (UIN: 654324917)

PUJITHA PRASHANTH HEGDE (UIN: 659660537)

SAVITHA TAMRAPARNI-VITTAL (UIN: 662405004)

USING HADOOP/MAPREDUCE FOR DATA PROCESSING AND DATA MANAGEMENT

IDS 561: ANALYTICS FOR BIG DATA PROJECT 1

**QUESTION 1**

In your implementation, **transform all words to lowercase** and **ignore all non-alphabetic characters (except whitespace).** So you should treat \*Bees as bees in the following example.

Output must be in the following format:

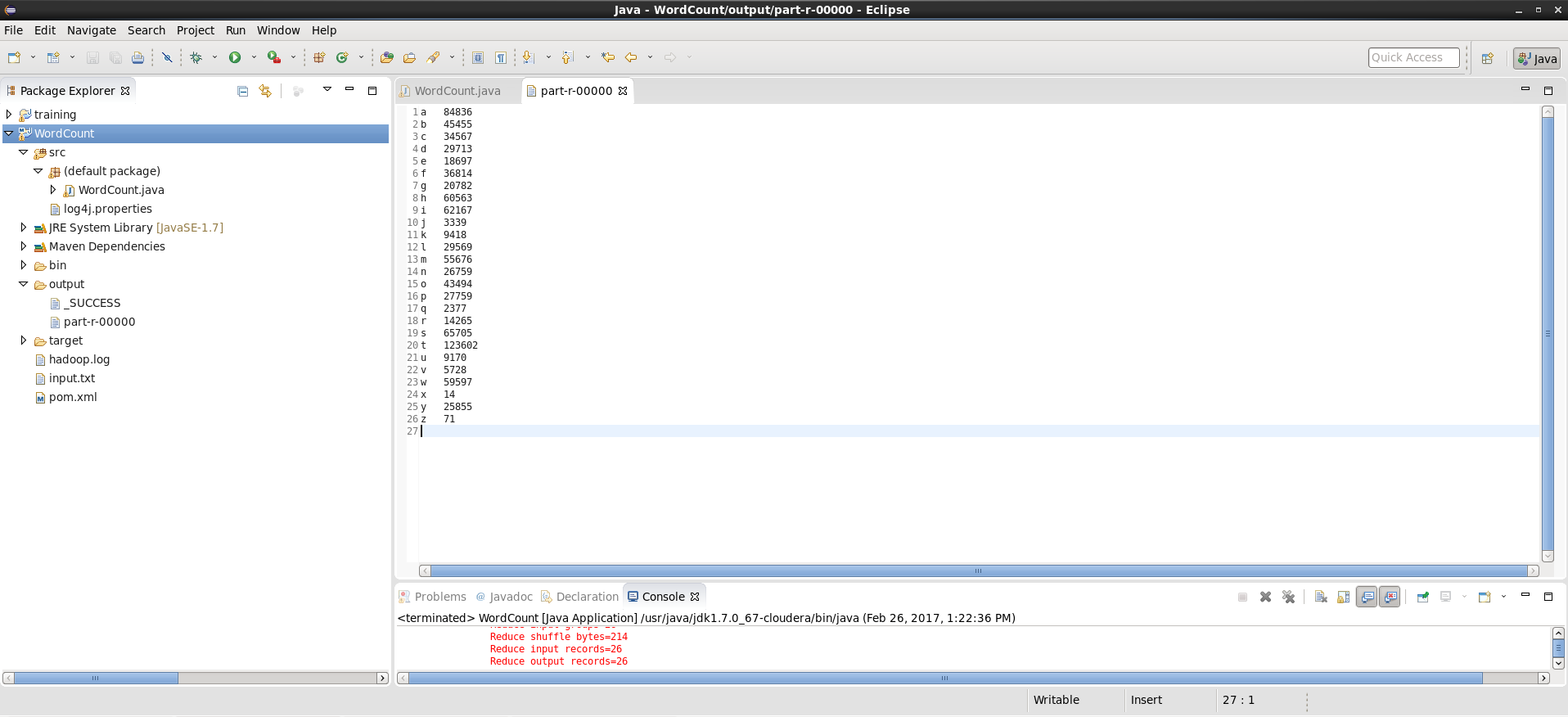
a<\tab>count of letters starting with a, e.g., apple  
b<\tab>count of letters starting with b, e.g., \*Bees  
…

**SOLUTION**

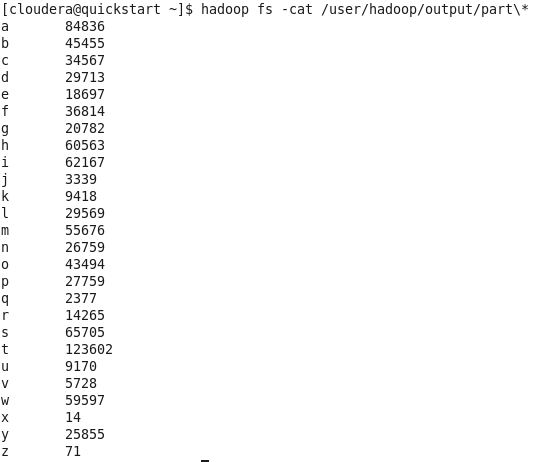
**PROGRAM CODE**

import java.io.IOException;  
import java.util.StringTokenizer;  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.Text;  
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;  
  
public class WordCount   
{  
    public static class TokenizerMapper  
    extends Mapper<Object, Text, Text, IntWritable>  
    {  
        private final static IntWritable one = new IntWritable(1);  
        private Text word = new Text();  
  
        public void map(Object key, Text value, Context context) throws IOException, InterruptedException   
        {  
            for (String token: value.toString().split("\\s+"))   
            {  
                for(char ch = 'a'; ch <= 'z'; ch ++ )  
                {  
                    String firstLetter = new StringBuilder().append(ch).toString();  
                    if(token.toLowerCase().startsWith(firstLetter))  
                    {  
                        context.write(new Text(firstLetter), one);  
                    }  
                }  
            }  
        }  
    }  
  
    public static class IntSumReducer  
    extends Reducer<Text,IntWritable,Text,IntWritable>   
    {  
        private IntWritable result = new IntWritable();  
        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException   
        {  
            int sum = 0;  
            for (IntWritable val : values)   
            {  
                sum += val.get();  
            }  
            result.set(sum);  
            context.write(key, result);  
        }  
    }  
  
    public static void main(String[] args) throws Exception   
    {  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf, "word count");  
        job.setJarByClass(WordCount.class);  
        job.setMapperClass(TokenizerMapper.class);  
        job.setCombinerClass(IntSumReducer.class);  
        job.setReducerClass(IntSumReducer.class);  
        job.setOutputKeyClass(Text.class);  
        job.setOutputValueClass(IntWritable.class);  
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        FileOutputFormat.setOutputPath(job, new Path(args[1]));  
        System.exit(job.waitForCompletion(true) ? 0 : 1);  
    }  
}

**SCREENSHOT OF JAVA OUTPUT**

****

**SCREENSHOT OF TERMINAL EXCEUTION**

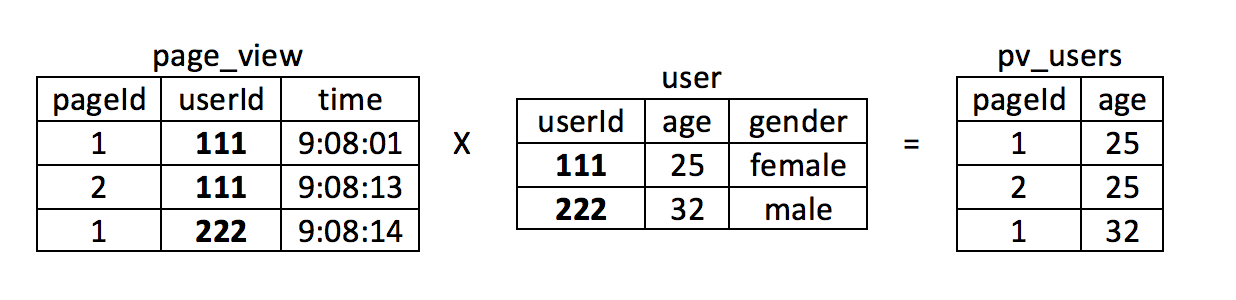
****

### Joining Tables [20 points]

**Description**

Suppose we have two datasets from a website:

1. page\_view: anonymous visitor records with *which* page is visited by *who* at *when*.
2. user: user database containing userId, age, and gender



**QUESTION 2**

1. HiveSQL **[8 pts]**

* Create a table called page\_view and a table called user, and populate them with input data files using HiveSQL
* Create a HiveSQL script to join these table. Submitthe **script** and **screenshot of the result** (you can use a SELECT query to display the content).

1. Write a MapReduce program for this join query (check the workflow in the slides). **[12 pts]**

* Skip and ignore header.
* Upload your **java file(s)** (Mapper, Reducer, Driver) and **output results** to blackboard.

**SOLUTION 2(A)**

**SCRIPT**

hadoop fs -mkdir /user/hadoop  
hadoop fs -ls /user/hadoop  
  
hadoop fs -put /home/cloudera/Documents/BigData/page\_view.txt /user/hadoop  
hadoop fs -put /home/cloudera/Documents/BigData/user.txt /user/hadoop

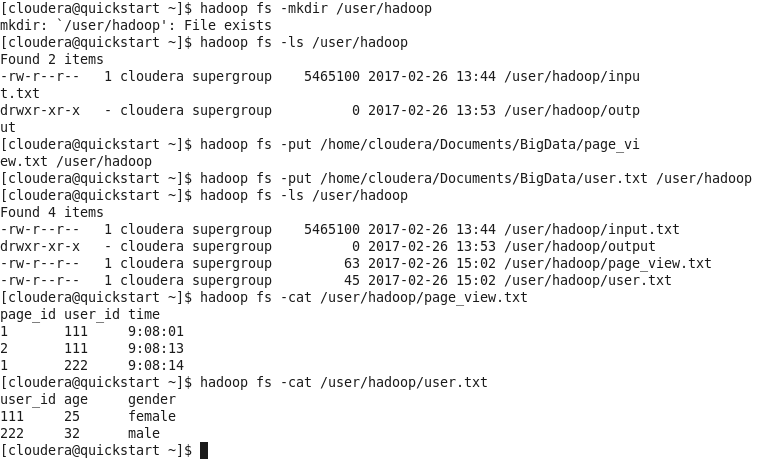
hadoop fs -ls /user/hadoop

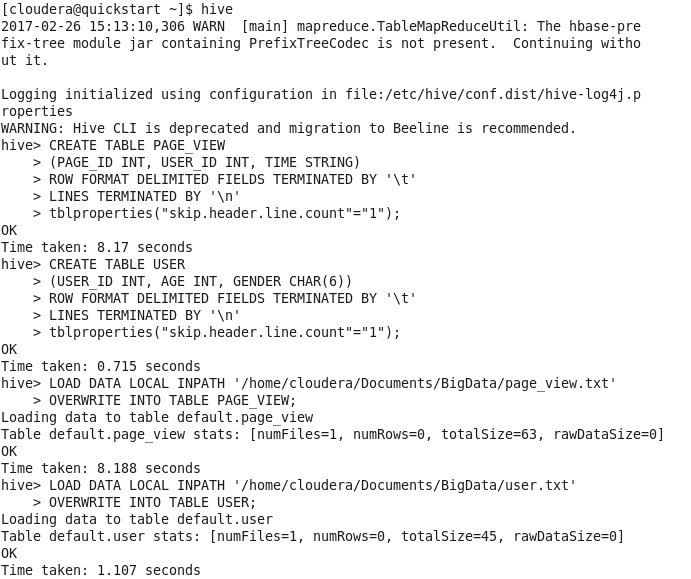
hadoop fs -cat /user/hadoop/page\_view.txt   
hadoop fs -cat /user/hadoop/user.txt

hive

hive> CREATE TABLE PAGE\_VIEW   
    > (PAGE\_ID INT, USER\_ID INT, TIME STRING)   
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'   
    > LINES TERMINATED BY '\n'  
    > tblproperties("skip.header.line.count" = "1");  
hive> LOAD DATA LOCAL INPATH '/home/cloudera/Documents/BigData/page\_view.txt'  
    > OVERWRITE INTO TABLE PAGE\_VIEW;  
  
  
hive> CREATE TABLE USER   
    > (USER\_ID INT, AGE INT, GENDER CHAR(6))   
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'   
    > LINES TERMINATED BY '\n'   
    > tblproperties("skip.header.line.count" = "1");  
  
hive> LOAD DATA LOCAL INPATH '/home/cloudera/Documents/BigData/user.txt'   
    > OVERWRITE INTO TABLE USER;  
  
  
hive> SET hive.cli.print.header=true;  
hive> select p.page\_id as page\_id, u.age as age from page\_view p join user u on (p.user\_id = u.user\_id);

**SCREENSHOTS**

****

****

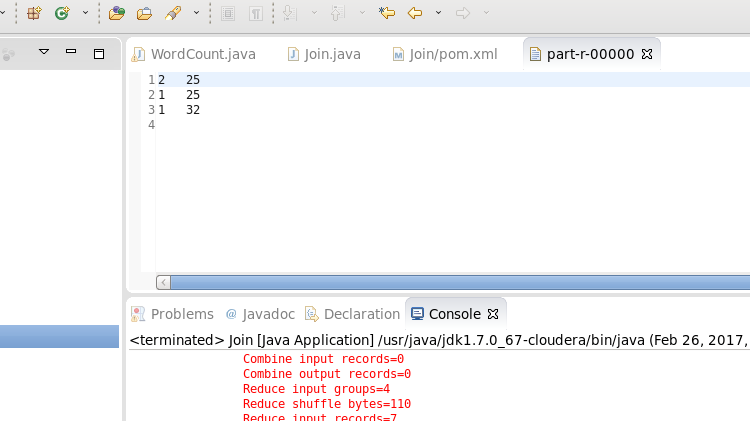
****

**SOLUTION 2(B)**

**PROGRAM CODE**

import java.io.IOException;  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.Text;  
import java.util.ArrayList;  
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.MultipleInputs;  
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
  
public class Join  
{  
  
    public static class PageViewMapper extends Mapper<LongWritable, Text, Text, Text>  
    {     
        private Text userID = new Text();  
        private Text table\_PageID = new Text();  
         
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException  
        {     
            String[] token = value.toString().split("\t");  
            if (Character.isLetter(token[0].charAt(0)))  
            {  
                for (int i = 1; i < token.length; i++)  
                {  
                       token[i] += token[i];;  
                   }  
               }  
               userID.set(token[1]);  
               table\_PageID.set("1" + "\t" + token[0]);  
               context.write(userID, table\_PageID);  
           }  
    }  
     
    public static class UserMapper extends Mapper<LongWritable, Text, Text, Text>  
    {     
        private Text userID = new Text();  
        private Text table\_Age = new Text();  
         
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException  
        {     
            String[] token = value.toString().split("\t");  
               if (Character.isLetter(token[0].charAt(0)))  
               {  
                   for (int i = 1; i < token.length; i++)  
                   {  
                       token[i] += token[i];  
                   }  
               }  
               userID.set(token[0]);  
               table\_Age.set("2" + "\t" + token[1]);  
               context.write(userID, table\_Age);  
           }  
    }  
     
    public static class JoinReducer extends Reducer<Text,Text,Text,Text>  
    {     
        private Text Key = new Text();  
        private Text Value = new Text();     
     
        public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException  
        {     
            ArrayList<Text> PageID\_list = new ArrayList<Text>();  
            ArrayList<Text> Age\_list = new ArrayList<Text>();  
            for (Text val : values)  
            {  
                   String[] recordCheck = val.toString().split("\t");  
                  if (recordCheck[0].equals("1"))  
                {  
                       PageID\_list.add(new Text(recordCheck[1]));     
                   }  
                 
                else if (recordCheck[0].equals("2"))  
                {  
                    Age\_list.add(new Text(recordCheck[1]));  
                }                     
            }  
  
            for (Text PageID : PageID\_list)  
            {  
                   for (Text Age : Age\_list)  
                   {  
                       Key.set(PageID.toString());  
                       Value.set(Age.toString());  
                    context.write(Key, Value);  
                 }  
             }  
         }  
    }  
     
    public static void main(String[] args) throws Exception  
     
    {     
         Configuration conf = new Configuration();  
         Job job = Job.getInstance(conf, "join 2 tables");  
         job.setJarByClass(Join.class);  
         MultipleInputs.addInputPath(job, new Path(args[0]),TextInputFormat.class,PageViewMapper.class);  
         MultipleInputs.addInputPath(job, new Path(args[1]),TextInputFormat.class,UserMapper.class);  
         job.setReducerClass(JoinReducer.class);  
         job.setOutputKeyClass(Text.class);  
         job.setOutputValueClass(Text.class);  
         FileOutputFormat.setOutputPath(job, new Path(args[2]));  
         System.exit(job.waitForCompletion(true) ? 0 : 1);  
    }  
}

**SCREENSHOT OF JAVA OUTPUT**

****

**SCREENSHOT OF TERMINAL EXECUTION**

****

**QUESTION 3**

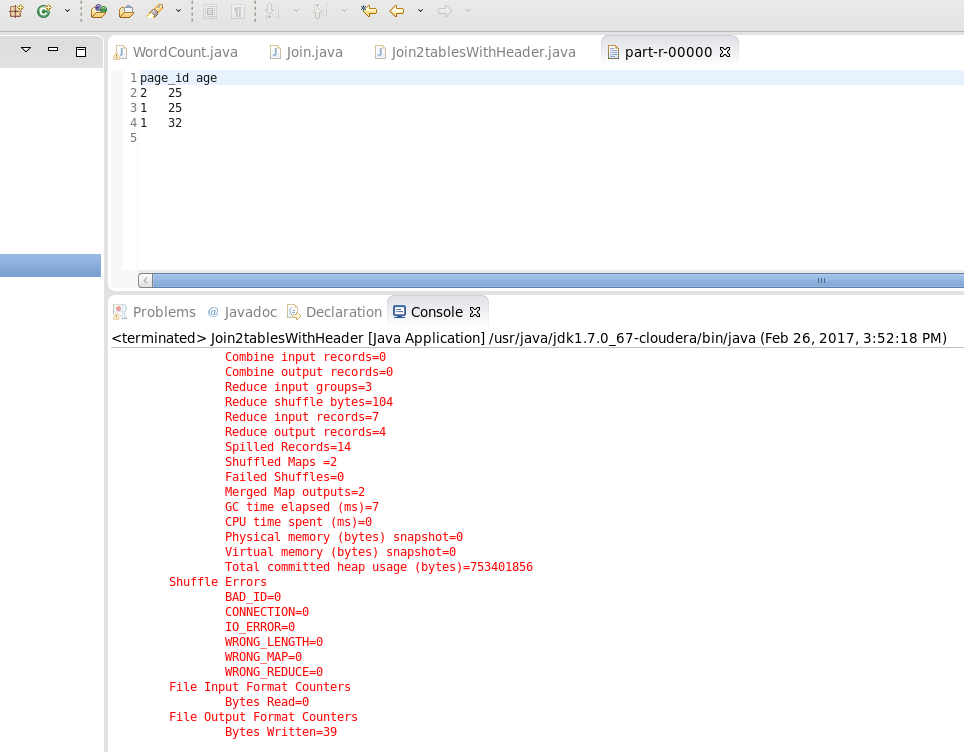
(Bonus) You will receive bonus points if your program can identify the header from input files and output it for the joined table (must be the first row). **[3pt]**

**SOLUTION**

**PROGRAM CODE**

import java.io.IOException;  
import java.util.ArrayList;  
  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.Text;  
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.MultipleInputs;  
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
  
public class Join2TablesWithHeader  
{  
  
    public static class PageViewMapper extends Mapper<LongWritable, Text, Text, Text>   
    {      
        private Text userID = new Text();  
        private Text table\_PageID = new Text();  
          
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException   
        {      
            String[] token = value.toString().split("\t");  
            for (int i = 0; i < token.length; i++)  
            {  
                if (Character.isDigit(token[0].charAt(0)))   
                {  
                       token[i] = token[i];  
                   }  
                else  
                {  
                    token[i] = "$" + token[i];  
                }  
               }  
               userID.set(token[1]);  
               table\_PageID.set("1" + "\t" + token[0]);  
               context.write(userID, table\_PageID);  
           }  
    }  
      
    public static class UserMapper extends Mapper<LongWritable, Text, Text, Text>   
    {      
        private Text userID = new Text();  
        private Text table\_Age = new Text();  
          
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException   
        {      
            String[] token = value.toString().split("\t");  
            for (int i = 0; i < token.length; i++)  
            {  
                if (Character.isDigit(token[0].charAt(0)))   
                {  
                       token[i] = token[i];  
                   }  
                else  
                {  
                    token[i] = "$" + token[i];  
                }  
               }  
               userID.set(token[0]);  
               table\_Age.set("2" + "\t" + token[1]);  
               context.write(userID, table\_Age);  
           }  
    }  
      
    public static class JoinReducer extends Reducer<Text,Text,Text,Text>   
    {      
        private Text Key = new Text();  
        private Text Value = new Text();      
      
        public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException   
        {      
             ArrayList<Text> PageID\_list = new ArrayList<Text>();  
             ArrayList<Text> Age\_list = new ArrayList<Text>();  
               
            for (Text val : values)   
            {  
                   String[] recordCheck = val.toString().split("\t");  
                  if (recordCheck[0].equals("1"))   
                {  
                       if (Character.isDigit(recordCheck[1].charAt(0)))  
                       {  
                           PageID\_list.add(new Text(recordCheck[1]));  
                       }  
                       else   
                       {  
                           PageID\_list.add(new Text(recordCheck[1].substring(1)));   
                       }  
                   }  
                    
                else if (recordCheck[0].equals("2"))   
                {  
                       if (Character.isDigit(recordCheck[1].charAt(0)))  
                       {  
                           Age\_list.add(new Text(recordCheck[1]));  
                       }  
                       else   
                       {  
                           Age\_list.add(new Text(recordCheck[1].substring(1)));  
                       }  
                   }  
            }  
  
            for (Text PageID : PageID\_list)   
            {  
                   for (Text Age : Age\_list)   
                   {  
                       Key.set(PageID.toString());  
                       Value.set(Age.toString());  
                    context.write(Key, Value);  
                 }  
             }  
         }  
    }  
      
    public static void main(String[] args) throws Exception   
      
    {      
         Configuration conf = new Configuration();  
         Job job = Job.getInstance(conf, "join 2 tables");  
         job.setJarByClass(Join2TablesWithHeader.class);  
         MultipleInputs.addInputPath(job, new Path(args[0]),TextInputFormat.class,PageViewMapper.class);  
         MultipleInputs.addInputPath(job, new Path(args[1]),TextInputFormat.class,UserMapper.class);  
         job.setReducerClass(JoinReducer.class);  
         job.setOutputKeyClass(Text.class);  
         job.setOutputValueClass(Text.class);  
         FileOutputFormat.setOutputPath(job, new Path(args[2]));  
         System.exit(job.waitForCompletion(true) ? 0 : 1);  
    }  
}

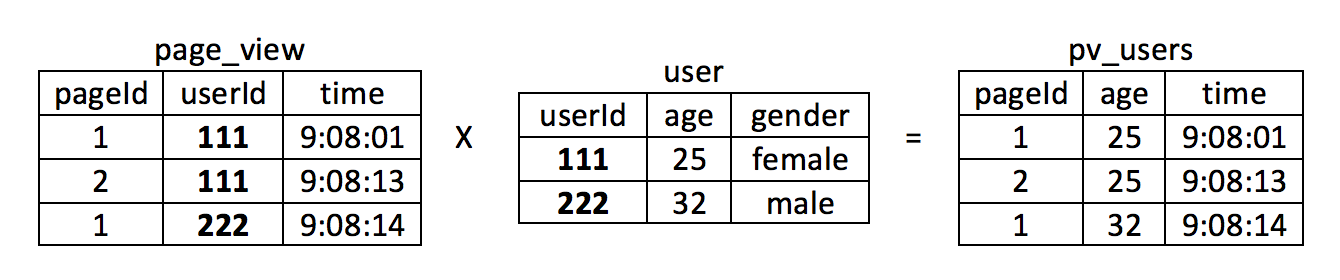
**SCREENSHOT OF JAVA OUTPUT**

****

**SCREENSHOT OF TERMINAL EXCEUTION**



**QUESTION 4**

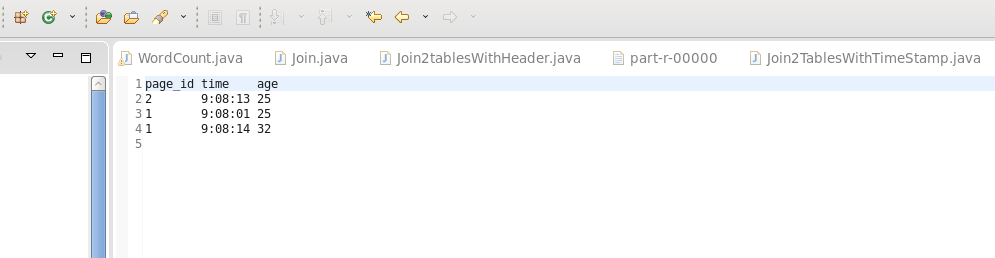
(Bonus) Write a MapReduce program to output the timestamp as shown below. **[2pt]**  


**SOLUTION**

**PROGRAM CODE**

import java.io.IOException;  
  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.Text;  
  
import java.util.ArrayList;  
  
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.MultipleInputs;  
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
  
public class Join2TablesWithTimestamp  
{  
  
    public static class PageViewMapper extends Mapper<LongWritable, Text, Text, Text>   
    {      
        private Text userID = new Text();  
        private Text table\_PageID\_ts = new Text();  
          
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException   
        {      
            String[] token = value.toString().split("\t");  
            for (int i = 0; i < token.length; i++)  
            {  
                if (Character.isDigit(token[0].charAt(0)))   
                {  
                       token[i] = token[i];  
                   }  
                else  
                {  
                    token[i] = "$" + token[i];  
                }  
               }  
               userID.set(token[1]);  
               table\_PageID\_ts.set("1" + "\t" + token[0] + "\t" + token[2]);  
               context.write(userID, table\_PageID\_ts);  
           }  
       }  
      
    public static class UserMapper extends Mapper<LongWritable, Text, Text, Text>   
    {      
        private Text userID = new Text();  
        private Text table\_Age = new Text();  
          
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException   
        {      
            String[] token = value.toString().split("\t");  
            for (int i = 0; i < token.length; i++)  
            {  
                if (Character.isDigit(token[0].charAt(0)))   
                {  
                       token[i] = token[i];  
                   }  
                else  
                {  
                    token[i] = "$" + token[i];  
                }  
               }  
               userID.set(token[0]);  
               table\_Age.set("2" + "\t" + token[1]);  
               context.write(userID, table\_Age);  
        }  
    }  
      
    public static class JoinReducer extends Reducer<Text,Text,Text,Text>   
    {      
        private Text Key = new Text();  
        private Text Value = new Text();      
      
        public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException   
        {      
            ArrayList<Text> PageID\_list = new ArrayList<Text>();  
            ArrayList<Text> Age\_list = new ArrayList<Text>();   
            for (Text val : values)   
            {  
                   String[] recordCheck = val.toString().split("\t");  
                  if (recordCheck[0].equals("1"))   
                {  
                       if (Character.isDigit(recordCheck[1].charAt(0)))  
                       {  
                           PageID\_list.add(new Text(recordCheck[1] + "\t" + "\t" + recordCheck[2]));  
                       }  
                       else   
                       {  
                           PageID\_list.add(new Text(recordCheck[1].substring(1) + "\t" + recordCheck[2].substring(1)));   
                       }  
                   }  
                  
                   else if (recordCheck[0].equals("2"))   
                   {  
                       if (Character.isDigit(recordCheck[1].charAt(0)))  
                       {  
                           Age\_list.add(new Text(recordCheck[1]));  
                       }  
                       else   
                       {  
                           Age\_list.add(new Text(recordCheck[1].substring(1)));  
                       }                    
                   }  
            }  
  
            for (Text PageID : PageID\_list)   
            {  
                for (Text Age : Age\_list)   
                {  
                    Key.set(PageID.toString());  
                     Value.set(Age.toString());  
                       context.write(Key, Value);  
                  }  
             }  
        }  
    }  
      
    public static void main(String[] args) throws Exception      
    {      
         Configuration conf = new Configuration();  
         Job job = Job.getInstance(conf, "join 2 tables");  
         job.setJarByClass(Join2TablesWithTimestamp.class);  
         MultipleInputs.addInputPath(job, new Path(args[0]),TextInputFormat.class,PageViewMapper.class);  
         MultipleInputs.addInputPath(job, new Path(args[1]),TextInputFormat.class,UserMapper.class);  
         job.setReducerClass(JoinReducer.class);  
         job.setOutputKeyClass(Text.class);  
         job.setOutputValueClass(Text.class);  
         FileOutputFormat.setOutputPath(job, new Path(args[2]));  
         System.exit(job.waitForCompletion(true) ? 0 : 1);  
    }  
}

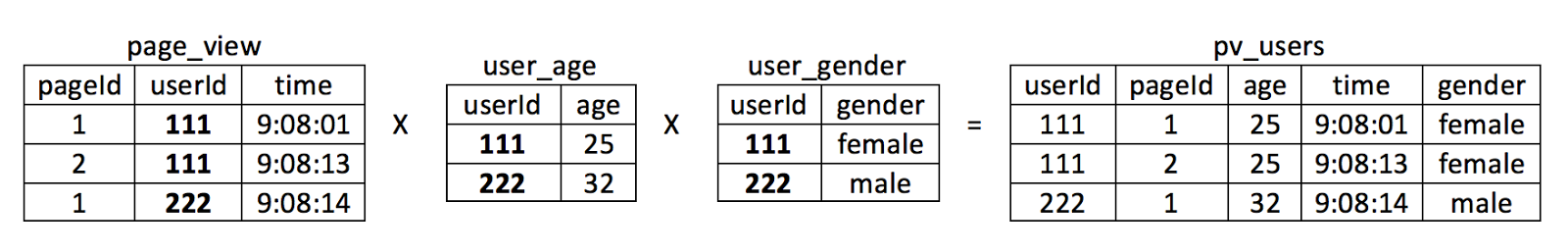
**SCREENSHOT OF JAVA OUTPUT**

****

**SCREENSHOT OF TERMINAL EXECUTION**



**QUESTION 5**

(Bonus) Generate similar input files for the 3 tables below and write a MapReduce program to perform the join task. **[3pt]**  


**SOLUTION**

**PROGRAM CODE**

import java.io.IOException;  
  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.Text;  
  
import java.util.ArrayList;  
  
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.MultipleInputs;  
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
  
public class Join3Tables   
{  
  
    public static class PageViewMapper extends Mapper<LongWritable, Text, Text, Text>   
    {      
        private Text userID = new Text();  
        private Text table\_PageID\_ts = new Text();    
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException   
        {      
            String[] token = value.toString().split("\t");  
            for (int i = 0; i < token.length; i++)  
            {  
                if (Character.isDigit(token[0].charAt(0)))   
                {  
                       token[i] = token[i];  
                   }  
                else  
                {  
                    token[i] = "$" + token[i];  
                }  
               }  
               userID.set(token[1]);  
               table\_PageID\_ts.set("1" + "\t" + token[1] + "\t" + token[0] + "\t" + token[2]);  
              context.write(userID, table\_PageID\_ts);  
        }  
    }  
      
    public static class UserAgeMapper extends Mapper<LongWritable, Text, Text, Text>   
    {      
        private Text userID = new Text();  
        private Text table\_Age = new Text();  
          
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException   
        {      
            String[] token = value.toString().split("\t");  
            for (int i = 0; i < token.length; i++)  
            {  
                if (Character.isDigit(token[0].charAt(0)))   
                {  
                       token[i] = token[i];  
                   }  
                else  
                {  
                    token[i] = "$" + token[i];  
                }  
               }  
               userID.set(token[0]);  
               table\_Age.set("2" + "\t" + token[0] + "\t" + token[1]);  
               context.write(userID, table\_Age);  
        }  
    }  
  
    public static class UserGenderMapper extends Mapper<LongWritable, Text, Text, Text>   
    {      
        private Text userID = new Text();  
        private Text table\_Gender = new Text();  
          
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException   
        {      
               String[] token = value.toString().split("\t");  
            for (int i = 0; i < token.length; i++)  
            {  
                if (Character.isDigit(token[0].charAt(0)))   
                {  
                       token[i] = token[i];  
                   }  
                else  
                {  
                    token[i] = "$" + token[i];  
                }  
               }  
               userID.set(token[0]);  
               table\_Gender.set("3" + "\t" + token[0] + "\t" + token[1]);  
               context.write(userID, table\_Gender);  
        }  
    }  
      
    public static class JoinReducer extends Reducer<Text,Text,Text,Text>   
    {      
        private Text Key = new Text();  
        private Text Value = new Text();      
      
        public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException   
        {      
            ArrayList<Text> PageID\_TS\_list = new ArrayList<Text>();  
             ArrayList<Text> Age\_list = new ArrayList<Text>();  
             ArrayList<Text> Gender\_list = new ArrayList<Text>();  
               
               for (Text val : values)   
               {  
                   String[] recordCheck = val.toString().split("\t");  
                  if (recordCheck[0].equals("1"))   
                {  
                       if (Character.isDigit(recordCheck[1].charAt(0)))  
                       {  
                           PageID\_TS\_list.add(new Text(recordCheck[1] + "\t" + "\t" + recordCheck[2] + "\t" + "\t" + recordCheck[3]));  
                       }  
                       else   
                       {  
                           PageID\_TS\_list.add(new Text(recordCheck[1].substring(1) + "\t" + recordCheck[2].substring(1) + "\t" + recordCheck[3].substring(1)));   
                       }  
                   }  
                  
                   else if (recordCheck[0].equals("2"))   
                   {  
                       if (Character.isDigit(recordCheck[1].charAt(0)))  
                       {  
                           Age\_list.add(new Text(recordCheck[2]));  
                       }  
                       else   
                       {  
                           Age\_list.add(new Text(recordCheck[2].substring(1)));  
                       }  
                }      
                    
                   else if (recordCheck[0].equals("3"))   
                {  
                       if (Character.isDigit(recordCheck[1].charAt(0)))  
                       {  
                           Gender\_list.add(new Text(recordCheck[2]));  
                       }  
                       else   
                       {  
                           Gender\_list.add(new Text(recordCheck[2].substring(1)));  
                       }  
                  }                               
             }  
  
               for (Text PageID : PageID\_TS\_list)   
               {  
                   for (Text Age : Age\_list)  
                   {  
                       for (Text Gender : Gender\_list)   
                    {  
                           Key.set(PageID.toString());  
                            Value.set(Age.toString() + "\t" + Gender.toString());  
                           context.write(Key, Value);  
                       }  
                   }               
               }  
        }  
    }  
     
    public static void main(String[] args) throws Exception   
    {      
         Configuration conf = new Configuration();  
         Job job = Job.getInstance(conf, "join 3 tables");  
         job.setJarByClass(Join3Tables.class);  
         MultipleInputs.addInputPath(job, new Path(args[0]),TextInputFormat.class,PageViewMapper.class);  
         MultipleInputs.addInputPath(job, new Path(args[1]),TextInputFormat.class,UserAgeMapper.class);  
         MultipleInputs.addInputPath(job, new Path(args[2]),TextInputFormat.class,UserGenderMapper.class);  
         job.setReducerClass(JoinReducer.class);  
         job.setOutputKeyClass(Text.class);  
         job.setOutputValueClass(Text.class);  
         FileOutputFormat.setOutputPath(job, new Path(args[3]));  
         System.exit(job.waitForCompletion(true) ? 0 : 1);  
    }  
}

**SCREENSHOT OF JAVA OUTPUT**

****

**SCREENSHOT OF TERMINAL EXECUTION**

****