BIG DATA HADOOP ASSIGNMENT-2

Session 2 : Assignment 1

TASK1:

We have a dataset of sales of different TV sets across different locations.

Dataset_Link

The fields are arranged like:

Company Name|Product Name|Size in inches|State|Pin Code|Price

Records look like:

Samsung|Optima|14|Madhya Pradesh|132401|14200

There are some invalid records which contain 'NA' in either Company Name or Product Name. 1:

1.Write a MapReduce program to filter out the invalid records. Map only job will fit for this

context.

Mapper:

```
package MapReduceAssignment;

import java.io.IOException;


import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.io.NullWritable;

public class TVDataSetMapper extends Mapper<LongWritable, Text,
NullWritable, Text > {
      public void map(LongWritable key, Text value, Context context)
                  throws IOException, InterruptedException {
            String[] lineArray = value.toString().split("\\|");



            if(!(lineArray[0].trim().equals("NA")
|| lineArray[1].trim().equals("NA")))
            {
            context.write(NullWritable.get(),  value);
            }
```

```
        }

}
```

Class:

```
package MapReduceAssignment;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import org.apache.hadoop.io.NullWritable;

public class TVDataSet {
        @SuppressWarnings("deprecation")
        public static void main(String[] args) throws Exception
                { Configuration conf = new Configuration();
                Job job = new Job(conf, "DemoTask1");
                job.setJarByClass(TVDataSet.class);

                job.setMapOutputKeyClass(NullWritable.class);
                job.setMapOutputValueClass(Text.class);

                job.setOutputKeyClass(NullWritable.class);
                job.setOutputValueClass(Text.class);
                job.setMapperClass(TVDataSetMapper.class);
                //job.setReducerClass(TVDataSetReducer.class);

                job.setInputFormatClass(TextInputFormat.class);
                job.setOutputFormatClass(TextOutputFormat.class);

                FileInputFormat.addInputPath(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job,new Path(args[1]));

                /*
                Path out=new Path(args[1]);
                out.getFileSystem(conf).delete(out);
                */

                job.waitForCompletion(true);
        }
}
```

BIG DATA HADOOP ASSIGNMENT-2

Output:

```
Samsung|Super|14|Maharashtra|619082|9200
Samsung|Super|14|Maharashtra|619082|9200
Samsung|Super|14|Maharashtra|619082|9200
Lava|Attention|20|Assam|454601|24200
Samsung|Decent|16|Kerala|922401|12200
Samsung|Optima|14|Madhya Pradesh|132401|14200
Zen|Super|14|Maharashtra|619082|9200
Lava|Attention|20|Assam|454601|24200
Onida|Decent|14|Uttar Pradesh|232401|16200
Onida|Lucid|18|Uttar Pradesh|232401|16200
Samsung|Optima|14|Madhya Pradesh|132401|14200
Zen|Super|14|Maharashtra|619082|9200
Lava|Attention|20|Assam|454601|24200
Akai|Decent|16|Kerala|922401|12200
Onida|Lucid|18|Uttar Pradesh|232401|16200
Samsung|Optima|14|Madhya Pradesh|132401|14200
```

2.Write a MapReduce program to calculate the total units sold for each Company.

Class:

```
package MapReduceAssignment;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;


public class TVDataSet {
	@SuppressWarnings("deprecation")
	public static void main(String[] args) throws Exception
		{ Configuration conf = new Configuration();
		Job job = new Job(conf, "DemoTask1");
		job.setJarByClass(TVDataSet.class);

		//job.setMapOutputKeyClass(String.class);
		//job.setMapOutputValueClass(IntWritable.class);

		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(IntWritable.class);
		job.setMapperClass(TVDataSetMapper.class);
		job.setReducerClass(TVDataSetReducer.class);

		job.setInputFormatClass(TextInputFormat.class);
```

```
            job.setOutputFormatClass(TextOutputFormat.class);

            FileInputFormat.addInputPath(job, new Path(args[0]));
            FileOutputFormat.setOutputPath(job,new Path(args[1]));


            /*
            Path out=new Path(args[1]);
            out.getFileSystem(conf).delete(out);
            */

            job.waitForCompletion(true);
      }
}
```

Mapper:

```
package MapReduceAssignment;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.*;


public class TVDataSetMapper extends Mapper<LongWritable, Text,
Text, IntWritable > {
      public void map(LongWritable key, Text value, Context context)
                  throws IOException, InterruptedException {
            String[] lineArray = value.toString().split("\\|");

            Text outputKey = new Text(lineArray[0].toUpperCase().trim());
                        context.write(outputKey, new IntWritable(1));

      }

}
```

Reducer:

```
package MapReduceAssignment;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
```

BIG DATA HADOOP ASSIGNMENT-2

```
public class TVDataSetReducer extends Reducer<Text, IntWritable,
Text, IntWritable>
{
	public void reduce(Text key, Iterable<IntWritable>
values,Context context) throws IOException, InterruptedException
	{

		int sum = 0;
		for (IntWritable value : values) {
			sum += value.get();
		}

		context.write(key, new IntWritable(sum));
	}

}
```

Output:

```
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost Hadoop]$ hadoop fs -cat MRTVTa
18/08/02 03:19:40 WARN util.NativeCodeLoader: Unab
ry for your platform... using builtin-java classes
AKAI    1
LAVA    3
ONIDA   3
SAMSUNG 7
ZEN     2
```

3. **Write a MapReduce program to calculate the total units sold in each state for Onida company.**

Class:

```
package MapReduceAssignment;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;


public class TVDataSet {
	@SuppressWarnings("deprecation")
	public static void main(String[] args) throws Exception
		{ Configuration conf = new Configuration();
```

```
            Job job = new Job(conf, "DemoTask1");
            job.setJarByClass(TVDataSet.class);

            //job.setMapOutputKeyClass(String.class);
            //job.setMapOutputValueClass(IntWritable.class);

            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(IntWritable.class);
            job.setMapperClass(TVDataSetMapper.class);
            job.setReducerClass(TVDataSetReducer.class);

            job.setInputFormatClass(TextInputFormat.class);
            job.setOutputFormatClass(TextOutputFormat.class);

            FileInputFormat.addInputPath(job, new Path(args[0]));
            FileOutputFormat.setOutputPath(job,new Path(args[1]));

            /*
            Path out=new Path(args[1]);
            out.getFileSystem(conf).delete(out);
            */

            job.waitForCompletion(true);
      }
}
```

Mapper:

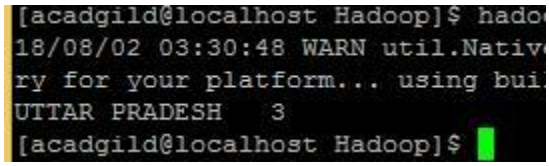
```
package MapReduceAssignment;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.*;


public class TVDataSetMapper extends Mapper<LongWritable, Text,
Text, IntWritable > {
      public void map(LongWritable key, Text value, Context context)
                  throws IOException, InterruptedException {
            String[] lineArray = value.toString().split("\\|");

            if(lineArray[0].trim().toUpperCase().equals("ONIDA"))
            {
            Text outputKey = new Text(lineArray[3].toUpperCase().trim());
                        context.write(outputKey, new IntWritable(1));
            }

      }

}
```

BIG DATA HADOOP ASSIGNMENT-2

Reducer:

```
package MapReduceAssignment;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class TVDataSetReducer extends Reducer<Text, IntWritable,
Text, IntWritable>
{
      public void reduce(Text key, Iterable<IntWritable>
values,Context context) throws IOException, InterruptedException
      {
            int sum = 0;
            for (IntWritable value : values) {
                  sum += value.get();
            }

            context.write(key, new IntWritable(sum));
      }

}
```

Output:



Task 2:

Dataset is sample data of songs heard by users on an online streaming platform.

The Description of data set attached in musicdata.txt is as follows: -

Col_1 - UserId

Col_2 - TrackId

Col_3 - Songs Share status (1 for shared, 0 for not shared)

Col_4 - Listening Platform (Radio or Web - 0 for radio, 1 for web)

BIG DATA HADOOP ASSIGNMENT-2

Col_5 - Song Listening Status (0 for skipped, 1 for fully heard)

1. Find the number of unique listeners in the data set.

Class:

```
package MapReduceAssignment;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;


public class MusicDataSet {
	@SuppressWarnings("deprecation")
	public static void main(String[] args) throws Exception
		{ Configuration conf = new Configuration();

		Job job = new Job(conf, "DemoTask1");
		job.setJarByClass(MusicDataSet.class);

		job.setMapOutputKeyClass(Text.class);
		job.setMapOutputValueClass(IntWritable.class);

		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(NullWritable.class);
		job.setMapperClass(MusicDataSetMapper.class);
		job.setReducerClass(MusicDataSetReducer.class);

		job.setInputFormatClass(TextInputFormat.class);
		job.setOutputFormatClass(TextOutputFormat.class);

		FileInputFormat.addInputPath(job, new Path(args[0]));
		FileOutputFormat.setOutputPath(job,new Path(args[1]));

		/*
		Path out=new Path(args[1]);
		out.getFileSystem(conf).delete(out);
		*/

		job.waitForCompletion(true);
	}
}
```

BIG DATA HADOOP ASSIGNMENT-2

Mapper:

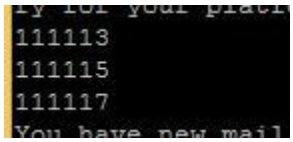
```
package MapReduceAssignment;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.*;

public class MusicDataSetMapper extends Mapper<LongWritable, Text,
Text, IntWritable > {
      public void map(LongWritable key, Text value, Context context)
                  throws IOException, InterruptedException {
            String[] lineArray = value.toString().split("\\|");


            Text outputKey = new Text(lineArray[0].toUpperCase().trim());
                        context.write(outputKey, new IntWritable(1));


      }

}
```

Reducer:

```
package MapReduceAssignment;
import java.io.IOException;
import      org.apache.hadoop.io.IntWritable;

import    org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;



public class MusicDataSetReducer extends Reducer<Text, IntWritable, Text, NullWritable>

{

        public void reduce(Text key, Iterable<IntWritable> values,Context context) throws IOException,
InterruptedException

        {
```

```
                context.write(key, NullWritable.get());

        }

}
```

Output:



2 What are the number of times a song was heard fully.

Class:

```
package MapReduceAssignment;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;


public class MusicDataSet {
      @SuppressWarnings("deprecation")
      public static void main(String[] args) throws Exception
            { Configuration conf = new Configuration();

            Job job = new Job(conf, "DemoTask1");
            job.setJarByClass(MusicDataSet.class);

            job.setMapOutputKeyClass(Text.class);
            job.setMapOutputValueClass(IntWritable.class);

            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(NullWritable.class);
            job.setMapperClass(MusicDataSetMapper.class);
            job.setReducerClass(MusicDataSetReducer.class);
```

```
            job.setInputFormatClass(TextInputFormat.class);
            job.setOutputFormatClass(TextOutputFormat.class);

            FileInputFormat.addInputPath(job, new Path(args[0]));
            FileOutputFormat.setOutputPath(job,new Path(args[1]));

            /*
            Path out=new Path(args[1]);
            out.getFileSystem(conf).delete(out);
            */

            job.waitForCompletion(true);
      }
}
```

Mapper:

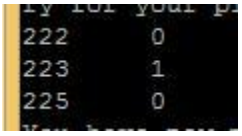
```
package MapReduceAssignment;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.*;


public class MusicDataSetMapper extends Mapper<LongWritable, Text,
Text, IntWritable > {
      public void map(LongWritable key, Text value, Context context)
                  throws IOException, InterruptedException {
            String[] lineArray = value.toString().split("\\|");


            Text outputKey = new Text(lineArray[1].toUpperCase().trim());
            if(lineArray[4].trim().equals("1"))
            {
                        context.write(outputKey, new IntWritable(1));
            }
            else
            {
                  context.write(outputKey, new IntWritable(0));
            }


      }

}
```

BIG DATA HADOOP ASSIGNMENT-2

Reducer:

```
package MapReduceAssignment;
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class MusicDataSetReducer extends Reducer<Text, IntWritable,
Text, IntWritable>
{
      public void reduce(Text key, Iterable<IntWritable>
values,Context context) throws IOException, InterruptedException
      {
            int sum = 0;
            for (IntWritable value : values) {
                  sum += value.get();
            }

            context.write(key, new IntWritable(sum));

      }

}
```

Output:



3. What are the number of times a song was shared

Class:

```
package MapReduceAssignment;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;


public class MusicDataSet {
```

BIG DATA HADOOP ASSIGNMENT-2

```
	@SuppressWarnings("deprecation")
	public static void main(String[] args) throws Exception
		{ Configuration conf = new Configuration();

		Job job = new Job(conf, "DemoTask1");
		job.setJarByClass(MusicDataSet.class);

		job.setMapOutputKeyClass(Text.class);
		job.setMapOutputValueClass(IntWritable.class);

		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(NullWritable.class);
		job.setMapperClass(MusicDataSetMapper.class);
		job.setReducerClass(MusicDataSetReducer.class);

		job.setInputFormatClass(TextInputFormat.class);
		job.setOutputFormatClass(TextOutputFormat.class);

		FileInputFormat.addInputPath(job, new Path(args[0]));
		FileOutputFormat.setOutputPath(job,new Path(args[1]));

		/*
		Path out=new Path(args[1]);
		out.getFileSystem(conf).delete(out);
		*/

		job.waitForCompletion(true);
	}
}
```

Mapper:

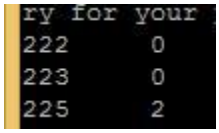
```
package MapReduceAssignment;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.*;


public class MusicDataSetMapper extends Mapper<LongWritable, Text,
Text, IntWritable > {
	public void map(LongWritable key, Text value, Context context)
			throws IOException, InterruptedException {
		String[] lineArray = value.toString().split("\\|");


		Text outputKey = new Text(lineArray[1].toUpperCase().trim());
		if(lineArray[2].trim().equals("1"))
		{
				context.write(outputKey, new IntWritable(1));
```

```
            }
            else
            {
                  context.write(outputKey, new IntWritable(0));
            }


      }

}
```

Reducer:

```
package MapReduceAssignment;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MusicDataSetReducer extends Reducer<Text, IntWritable,
Text, IntWritable>
{
      public void reduce(Text key, Iterable<IntWritable>
values,Context context) throws IOException, InterruptedException
      {
            int sum = 0;
            for (IntWritable value : values) {
                  sum += value.get();
            }

            context.write(key, new IntWritable(sum));

      }

}
```

Output:



ry for your	
222	0
223	0
225	2

BIG DATA HADOOP ASSIGNMENT-2

Task 3:

1. Use Sqoop tool to export data present in SQOOPOUT folder made while demo of Import table

```
mysql> show tables;
+--------------------+
| Tables_in_simplidb |
+--------------------+
| Person             |
| PersonImport       |
+--------------------+
2 rows in set (0.06 sec)

mysql> select * from PersonImport;
Empty set (0.00 sec)
```

```
[acadgild@localhost ~]$ sqoop export --connect jdbc:mysql://localhost/simplidb -
-table PersonImport --username root -P --export-dir /sqoopout
```

```
mysql> select * from PersonImport;
Empty set (0.00 sec)

mysql> select * from PersonImport;
+-----------+-------+--------+---------------+---------------+
| person_id | lname | fname  | area          | city          |
+-----------+-------+--------+---------------+---------------+
|         4 | Jhon  | Miller |  Los Angeles  | United States |
|         1 | Shyam | Ram    | Patna         | Bihar         |
|         2 | Tanya | Priya  | Whitefiled    | Bangalore     |
|         3 | James | Brown  | New York      | United States |
|       789 | a     | b      | c             | d             |
+-----------+-------+--------+---------------+---------------+
5 rows in set (0.00 sec)
```

2. Use Sqoop tool to import data from the same mysql table where the person_id = 3 into a new hdfs directory SQOOP_FILTER.

```
[acadgild@localhost ~]$ sqoop import --connect jdbc:mysql://localhost/simplidb
--username root -P --query 'select * from Person where $CONDITIONS AND person_id
 =3' --split-by person_id --target-dir /sqoop_filter
```

BIG DATA HADOOP ASSIGNMENT-2