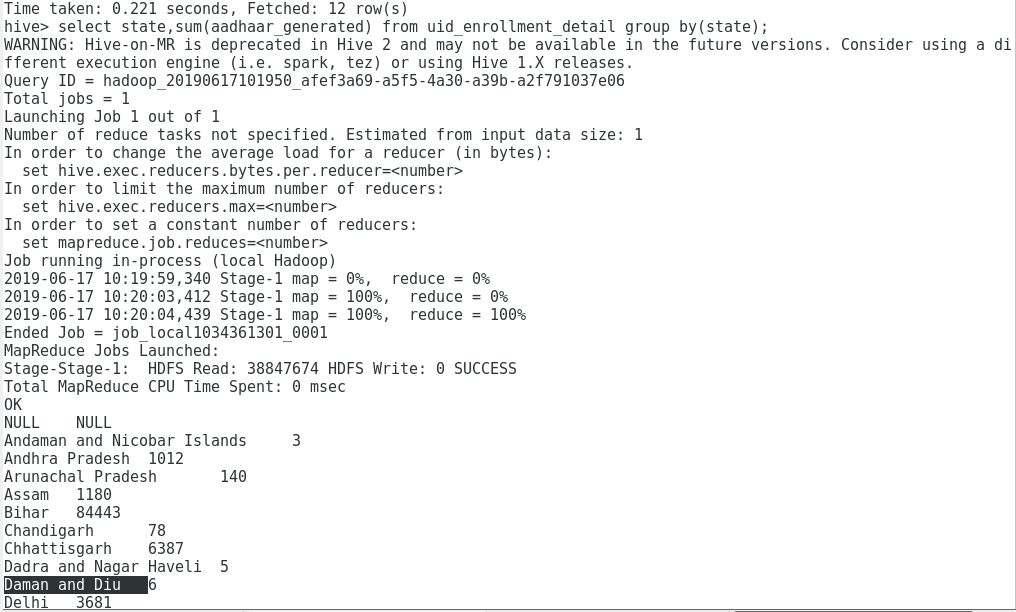
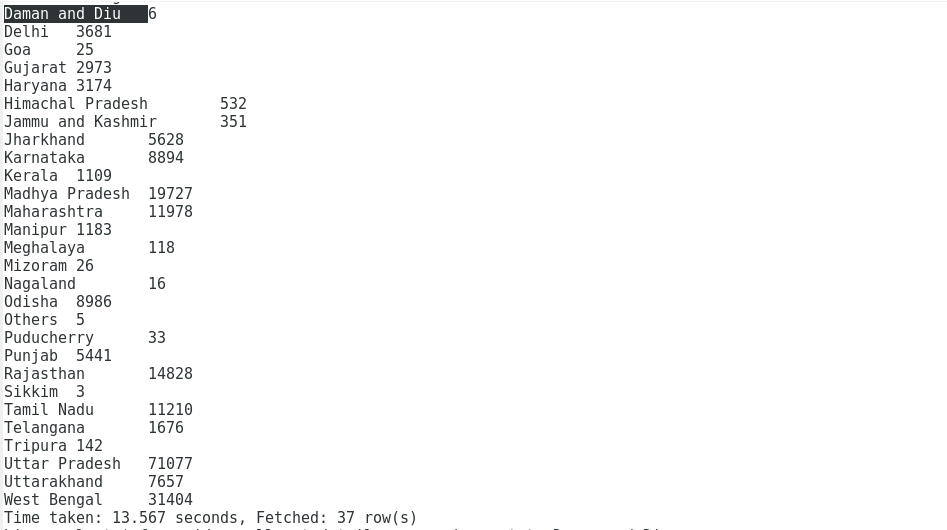
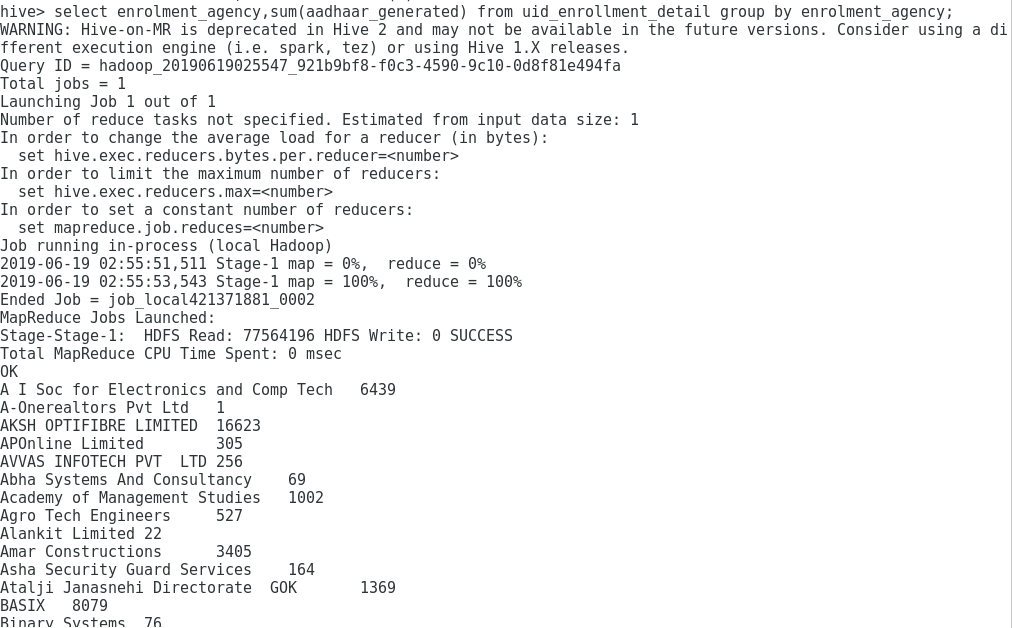
**HIVE QUERIES**

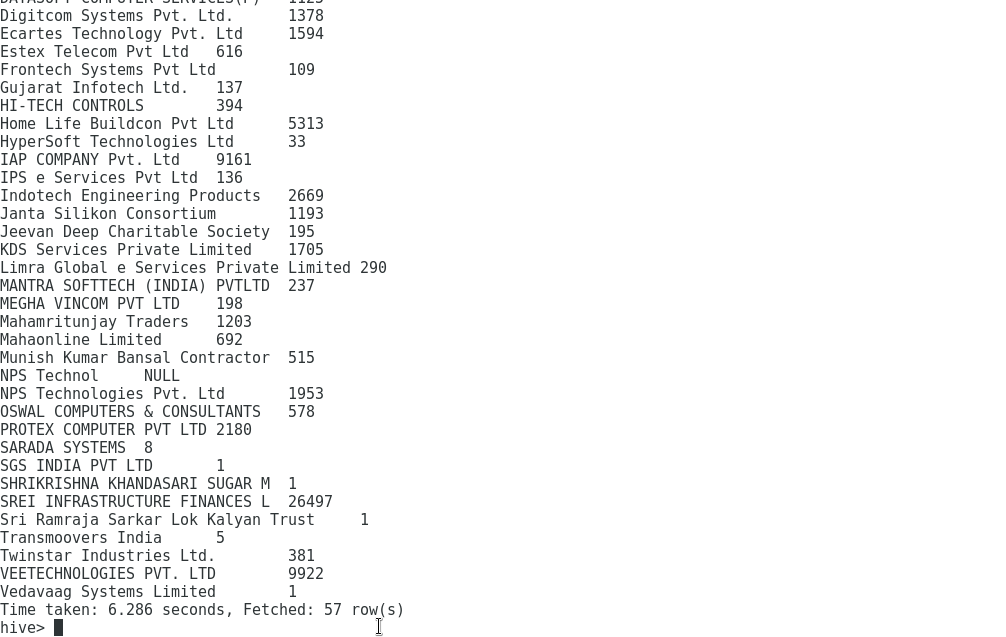
**1.Number of Aadhaar Identities which are generated in each state.**

****

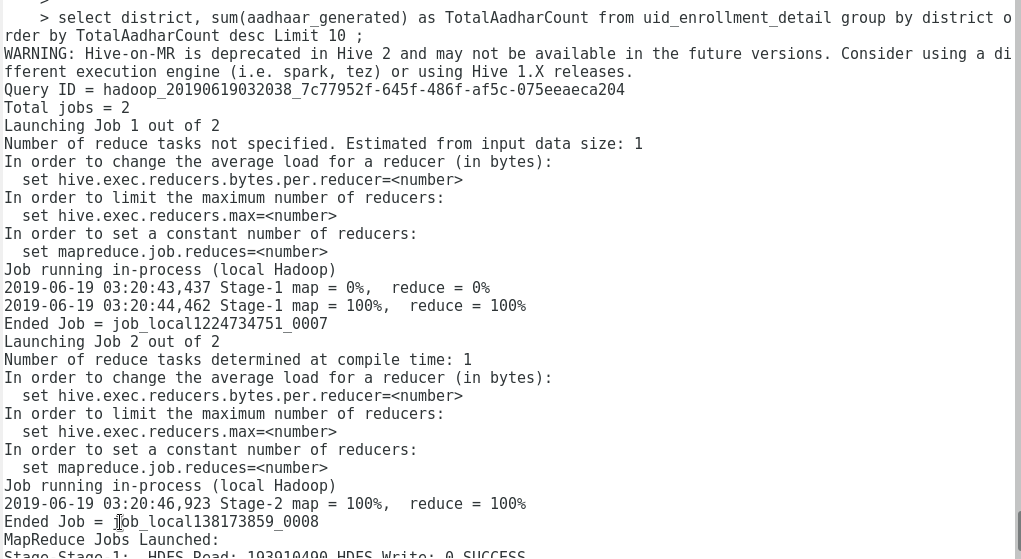
****

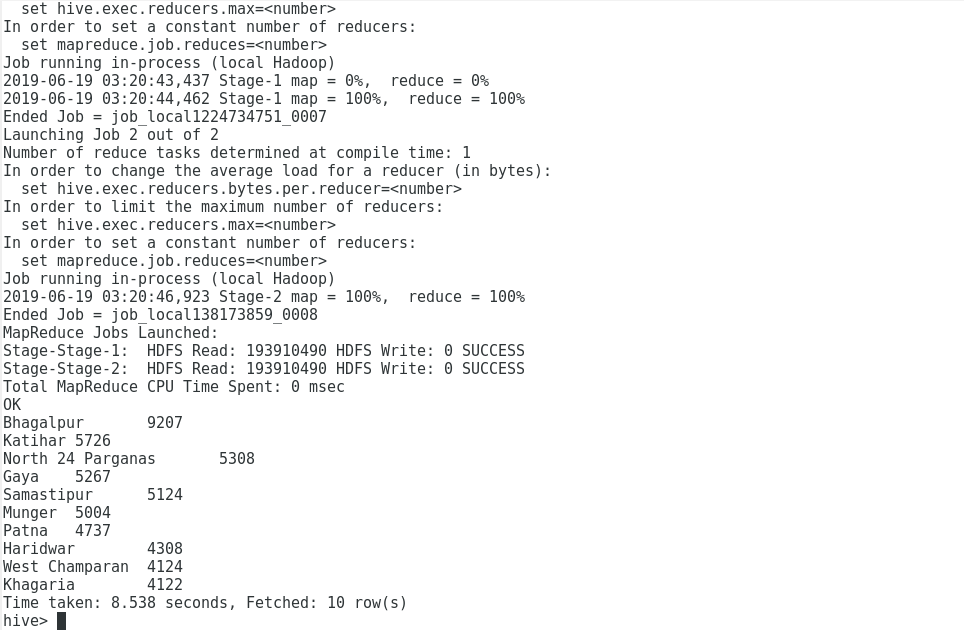
**2.Number of Aadhaar Identities which are generated by each Enrollment Agency**

****

****

**3.Top 10 districts with maximum Aadhaar identities generated for both Male and Female.**

****

****

**CORRESPONDING MAPREDUCE CODE FOR EACH OF THE HIVE QUERIES**

**COMMON DRIVER CLASS**

* **DriverClass.java**

import java.io.IOException;

import org.apache.hadoop.conf.Configured;

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.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.Mapper.Context;

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.util.Tool;

import org.apache.hadoop.util.ToolRunner;

import org.apache.log4j.Logger;

public class Driver extends configured implements tool{

public int run(String[] args) throws Exception {

if (args.length != 3) {

System.out.println("Usage: [input] [output1] [output2]");

System.exit(-1);

}

Job stateWiseCount = new Job();

stateWiseCount.setJobName("Aadhaar Data Analysis");

stateWiseCount.setJarByClass(Driver.class);

/\* Field separator for reducer output\*/

stateWiseCount.getConfiguration().set("mapreduce.output.textoutputformat.separator", " | ");

stateWiseCount.setMapperClass(MapperClass.class);

stateWiseCount.setReducerClass(ReducerClass.class);

stateWiseCount.setInputFormatClass(TextInputFormat.class);

stateWiseCount.setMapOutputKeyClass(Text.class);

stateWiseCount.setMapOutputValueClass(IntWritable.class);

stateWiseCount.setOutputKeyClass(Text.class);

stateWiseCount.setOutputValueClass(IntWritable.class);

Path inputFilePath = new Path(args[0]);

Path outputFilePath = new Path(args[1]);

FileInputFormat.addInputPath(stateWiseCount, inputFilePath);

FileOutputFormat.setOutputPath(stateWiseCount, outputFilePath);

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

}

public static void main(String[] args) throws Exception {

ToolRunner.run(new Driver(), args);

}

}

1. **QUERY1**

* **MAPPER.JAVA**

public class MapperClass extends Mapper<LongWritable, Text, Text, IntWritable> {

Text state = new Text();

IntWritable count = new IntWritable();

private static final Logger LOG = Logger.getLogger(MapperClass.class);

public void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

String[] list = value.toString().split(",");

if (key.get() != 0) {

state.set(list[2]);

count.set(Integer.parseInt(list[8]));

LOG.info("Reading line: " + state + "Aadhaars generated: "

+ list[8]);

context.write(state, count);

}

LOG.info("Skipped line with key:" + key.get());

}

**}**

* **Reducer.java**

public class ReducerClass extends

Reducer<Text, IntWritable, Text, IntWritable> {

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

throws IOException, InterruptedException {

int sum = 0;

for (IntWritable count : values) {

sum += count.get();

}

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

}

}

**QUERY 2**

Mapper class

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.Mapper;

import org.apache.log4j.Logger;

public class MapperClass extends Mapper<LongWritable, Text, Text, IntWritable> {

Text agencies = new Text();

IntWritable count = new IntWritable();

private static final Logger LOG = Logger.getLogger(MapperClass.class);

public void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

String[] list = value.toString().split(",");

if (key.get() != 0) {

agencies.set(list[1]);

count.set(Integer.parseInt(list[8]));

LOG.info("Reading line: " + agencies + "Aadhaars generated: "

+ list[8]);

context.write(agencies, count);

}

LOG.info("Skipped line with key:" + key.get());

}

}

**Reducer Class**

**import java.io.IOException;**

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

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

**import org.apache.hadoop.mapreduce.Reducer;**

**public class ReducerClass extends**

**Reducer<Text, IntWritable, Text, IntWritable> {**

**public void reduce(Text key, Iterable<IntWritable> values, Context context)**

**throws IOException, InterruptedException {**

**int sum = 0;**

**for (IntWritable count : values) {**

**sum += count.get();**

**}**

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

**}**

**}**

**Query 3**

**Mapper Class**

**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.Mapper;**

**import org.apache.log4j.Logger;**

**public class MapperClass extends Mapper<LongWritable, Text, Text, IntWritable> {**

**Text district = new Text();**

**IntWritable count = new IntWritable();**

**private static final Logger LOG = Logger.getLogger(MapperClass.class);**

**public void map(LongWritable key, Text value, Context context)**

**throws IOException, InterruptedException {**

**String[] list = value.toString().split(",");**

**if (key.get() != 0) {**

**district.set(list[3]);**

**count.set(Integer.parseInt(list[8]));**

**LOG.info("Reading line: " + district + "Aadhaars generated: "**

**+ list[8]);**

**context.write(district, count);**

**}**

**LOG.info("Skipped line with key:" + key.get());**

**}**

**}**

**Reducer Class**

**import java.io.IOException;**

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

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

**import org.apache.hadoop.mapreduce.Reducer;**

**public class ReducerClass extends**

**Reducer<Text, IntWritable, Text, IntWritable> {**

**public void reduce(Text key, Iterable<IntWritable> values, Context context)**

**throws IOException, InterruptedException {**

**int sum = 0,k=0;**

**for (IntWritable count : values) {**

**sum += count.get();**

**k++;**

**}**

**foR(int i=k;i<=k-10;i--)**

**{**

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

**}**

**}**

**}**