Module 3: Hadoop MapReduce Framework

**Assignment** – Find the Hot and Cold Days

Code:

**package** in.edureka.mapreduce;

**import** java.io.IOException;

**import** java.util.Iterator;

**import** org.apache.hadoop.fs.Path;

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

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

**import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

**import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

**import** org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

**import** org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

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

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

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

**import** org.apache.hadoop.conf.Configuration;

**public** **class** WeatherData {

//Mapper

/\*\*

\*MaxTemperatureMapper class is static and extends Mapper abstract class

having four hadoop generics type LongWritable, Text, Text, Text.

\*/

**public** **static** **class** MaxTemperatureMapper **extends**

Mapper<LongWritable, Text, Text, Text> {

/\*\*

\* **@method** map

\* This method takes the input as text data type.

\* Now leaving the first five tokens,it takes 6th token is taken as temp\_max and

\* 7th token is taken as temp\_min. Now temp\_max > 35 and temp\_min < 10 are passed to the reducer.

\*/

@Override

**public** **void** map(LongWritable arg0, Text Value, Context context)

**throws** IOException, InterruptedException {

//Converting the record (single line) to String and storing it in a String variable line

String line = Value.toString();

//Checking if the line is not empty

**if** (!(line.length() == 0)) {

//date

String date = line.substring(6, 14);

//maximum temperature

**float** temp\_Max = Float

.*parseFloat*(line.substring(38, 45).trim());

//minimum temperature

**float** temp\_Min = Float

.*parseFloat*(line.substring(46, 53).trim());

//if maximum temperature is greater than 35 , its a hot day

**if** (temp\_Max > 40.0) {

// Hot day

context.write(**new** Text(date),

**new** Text("Hot Day"));

}

//if minimum temperature is less than 10 , its a cold day

**if** (temp\_Min < 10) {

// Cold day

context.write(**new** Text(date),

**new** Text("Cold Day"));

}

}

}

}

//Reducer

/\*\*

\*MaxTemperatureReducer class is static and extends Reducer abstract class

having four hadoop generics type Text, Text, Text, Text.

\*/

**public** **static** **class** MaxTemperatureReducer **extends**

Reducer<Text, Text, Text, Text> {

/\*\*

\* **@method** reduce

\* This method takes the input as key and list of values pair from mapper, it does aggregation

\* based on keys and produces the final context.

\*/

**public** **void** reduce(Text Key, Iterator<Text> Values, Context context)

**throws** IOException, InterruptedException {

//putting all the values in temperature variable of type String

String temperature = Values.next().toString();

context.write(Key, **new** Text(temperature));

}

}

/\*\*

\* **@method** main

\* This method is used for setting all the configuration properties.

\* It acts as a driver for map reduce code.

\*/

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

//reads the default configuration of cluster from the configuration xml files

Configuration conf = **new** Configuration();

//Initializing the job with the default configuration of the cluster

Job job = **new** ~~Job~~(conf, "weather example");

//Assigning the driver class name

job.setJarByClass(WeatherData.**class**);

//Key type coming out of mapper

job.setMapOutputKeyClass(Text.**class**);

//value type coming out of mapper

job.setMapOutputValueClass(Text.**class**);

//Defining the mapper class name

job.setMapperClass(MaxTemperatureMapper.**class**);

//Defining the reducer class name

job.setReducerClass(MaxTemperatureReducer.**class**);

//Defining input Format class which is responsible to parse the dataset into a key value pair

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

//Defining output Format class which is responsible to parse the dataset into a key value pair

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

//setting the second argument as a path in a path variable

Path OutputPath = **new** Path(args[1]);

//Configuring the input path from the filesystem into the job

FileInputFormat.*addInputPath*(job, **new** Path(args[0]));

//Configuring the output path from the filesystem into the job

FileOutputFormat.*setOutputPath*(job, **new** Path(args[1]));

//deleting the context path automatically from hdfs so that we don't have delete it explicitly

OutputPath.getFileSystem(conf).~~delete~~(OutputPath);

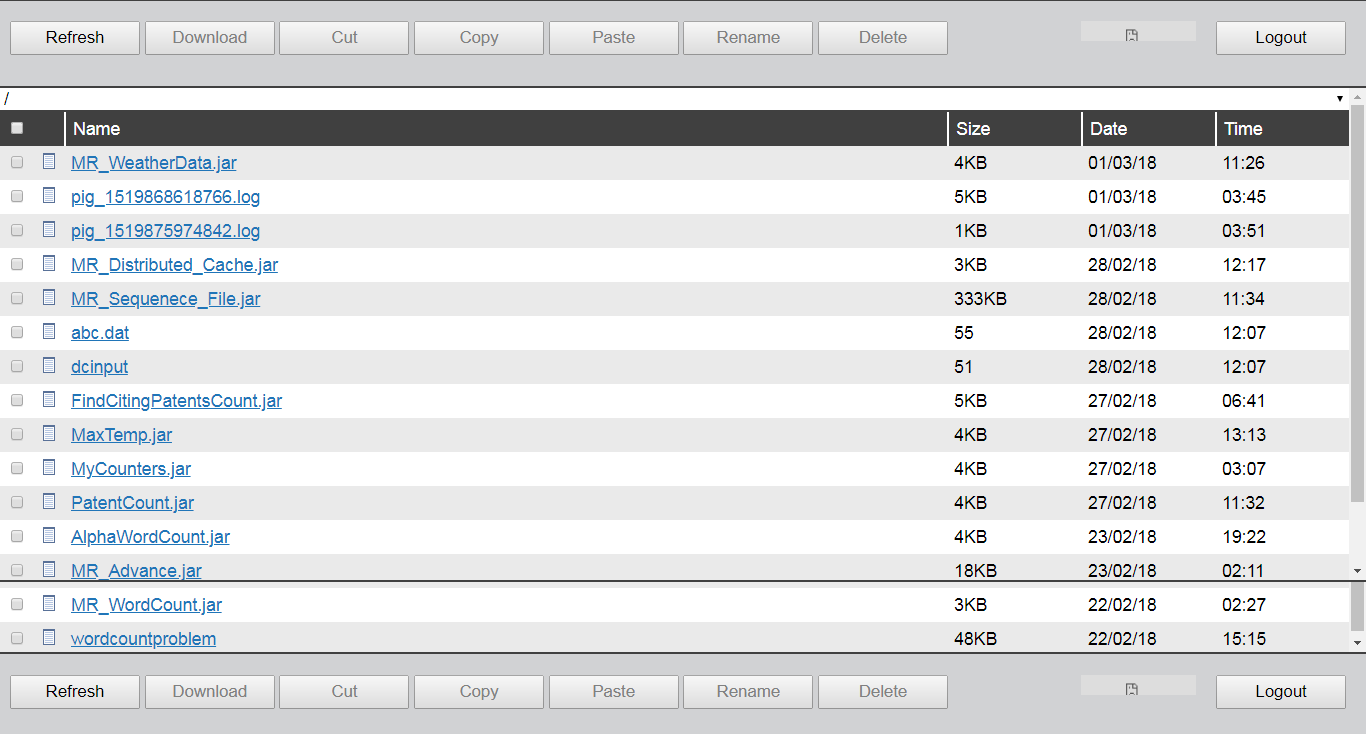
//exiting the job only if the flag value becomes false

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

}

}

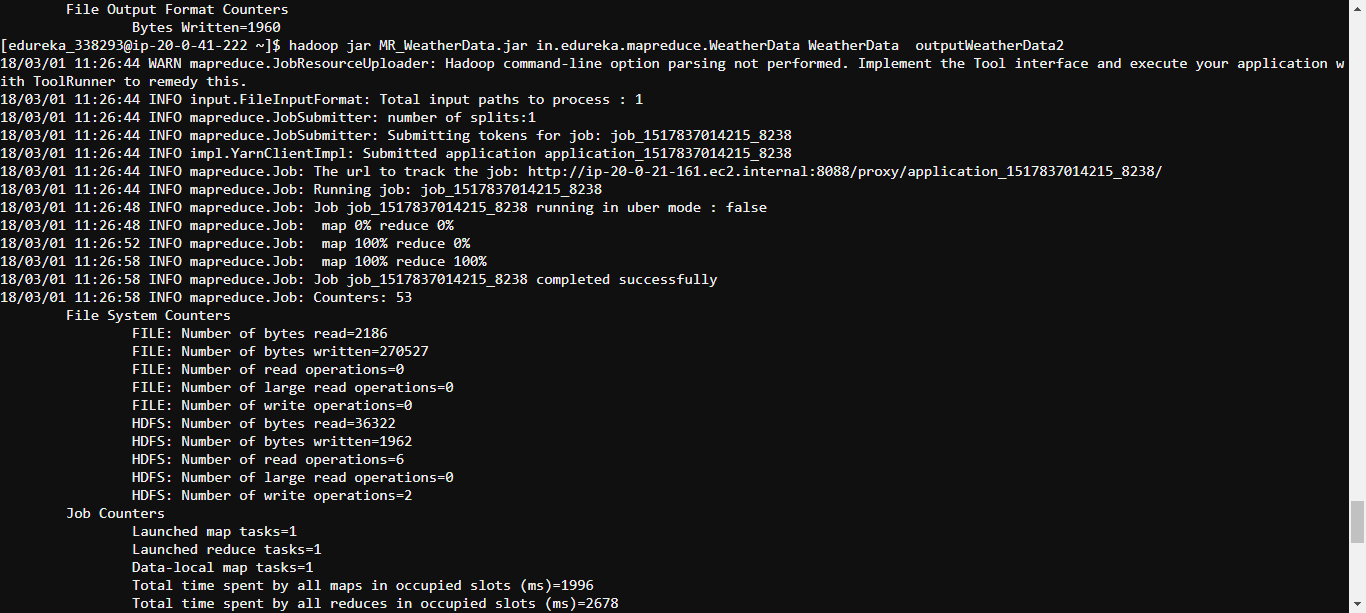
FTP:



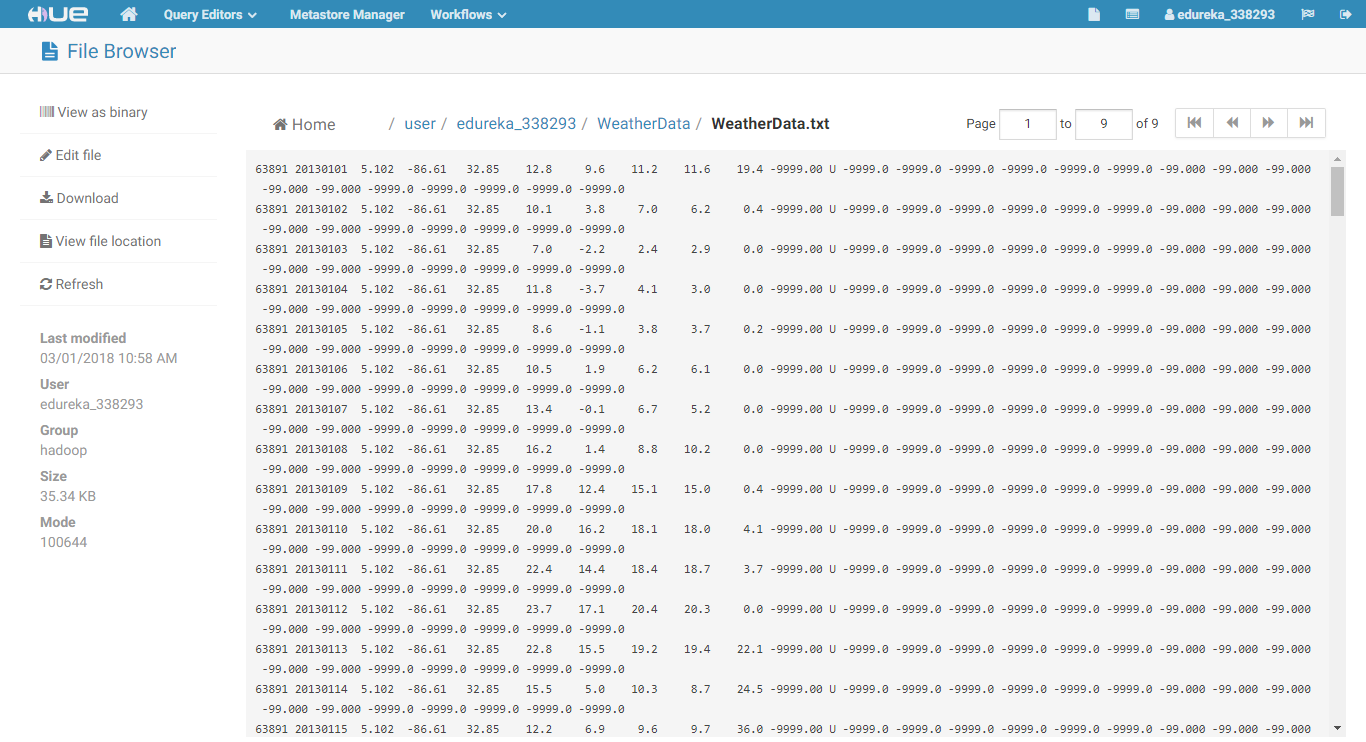
Command:

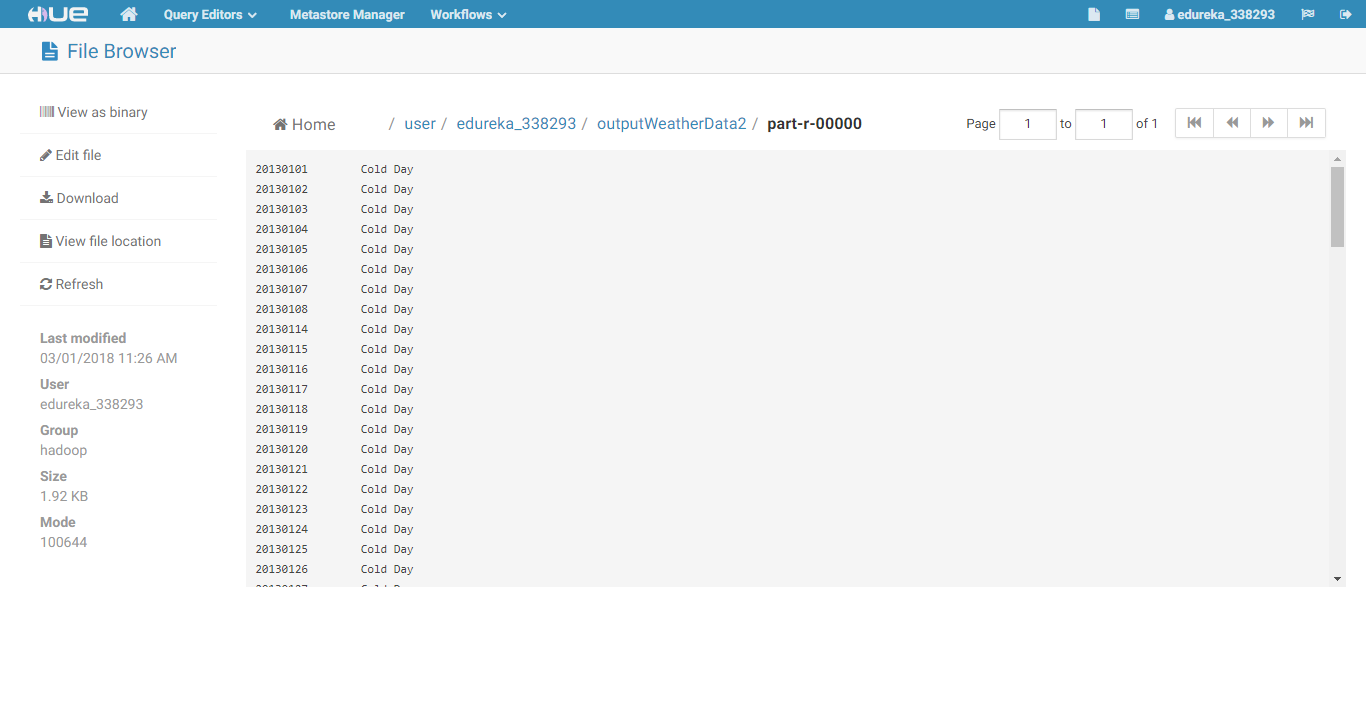
hadoop jar MR\_WeatherData.jar in.edureka.mapreduce.WeatherData WeatherData outputWeatherData

Webconsole:



HUE:





Output:

20130101 Cold Day

20130102 Cold Day

20130103 Cold Day

20130104 Cold Day

20130105 Cold Day

20130106 Cold Day

20130107 Cold Day

20130108 Cold Day

20130114 Cold Day

20130115 Cold Day

20130116 Cold Day

20130117 Cold Day

20130118 Cold Day

20130119 Cold Day

20130120 Cold Day

20130121 Cold Day

20130122 Cold Day

20130123 Cold Day

20130124 Cold Day

20130125 Cold Day

20130126 Cold Day

20130127 Cold Day

20130128 Cold Day

20130129 Cold Day

20130130 Cold Day

20130131 Cold Day

20130201 Cold Day

20130202 Cold Day

20130203 Cold Day

20130204 Cold Day

20130205 Cold Day

20130206 Cold Day

20130207 Cold Day

20130208 Cold Day

20130209 Cold Day

20130210 Cold Day

20130211 Cold Day

20130212 Cold Day

20130213 Cold Day

20130214 Cold Day

20130215 Cold Day

20130216 Cold Day

20130217 Cold Day

20130218 Cold Day

20130219 Cold Day

20130220 Cold Day

20130221 Cold Day

20130222 Cold Day

20130223 Cold Day

20130224 Cold Day

20130225 Cold Day

20130226 Cold Day

20130227 Cold Day

20130228 Cold Day

20130301 Cold Day

20130302 Cold Day

20130303 Cold Day

20130304 Cold Day

20130305 Cold Day

20130306 Cold Day

20130307 Cold Day

20130308 Cold Day

20130309 Cold Day

20130311 Cold Day

20130312 Cold Day

20130313 Cold Day

20130314 Cold Day

20130315 Cold Day

20130316 Cold Day

20130317 Cold Day

20130319 Cold Day

20130320 Cold Day

20130321 Cold Day

20130322 Cold Day

20130323 Cold Day

20130324 Cold Day

20130325 Cold Day

20130326 Cold Day

20130327 Cold Day

20130328 Cold Day

20130329 Cold Day

20130330 Cold Day

20130401 Cold Day

20130402 Cold Day

20130403 Cold Day

20130404 Cold Day

20130405 Cold Day

20130406 Cold Day

20130407 Cold Day

20130412 Cold Day

20130413 Cold Day

20130414 Cold Day

20130419 Cold Day

20130420 Cold Day

20130421 Cold Day

20130422 Cold Day

20130425 Cold Day

20130426 Cold Day

20130430 Cold Day

20130503 Cold Day

20130504 Cold Day

20130505 Cold Day

20130506 Cold Day

20130508 Cold Day

20130512 Cold Day

20130513 Cold Day

20130514 Cold Day

20130522 Cold Day

20130525 Cold Day