7(b) : MapReduce application for Job Scheduling on Hadoop cluster.

Step 1: Start Cloudera and open Eclipse. Create a new Java Project. Enter the project name and Check the default output folder and Click Next

Step 2: Add external jars to compile the code. **Click on Libraries Tab** and "Add External Jars". Right Select all the jars present in folder /usr/lib/hadoop/client , /usr/lib/Hadoop, /usr/lib/hadoop/lib.

Step 3: Select File system and click on usr/lib/Hadoop/lib. Select all Jar files and click on Ok and Finish.

Step 4: Project is created with name Word Count. Right Click on the project new Class.

Step 5 : Create a jar file of the program. Right click on project select "Export" and then click on "Jar File" under Java folder. Give the location path where you want to store your .jar file.

Step 6: Write the java program using Map, Reduce and Driver methods and save it.

Code:

```
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;

import java.io.IOException;
import java.util.StringTokenizer;

public class WordCountJobScheduler {

   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 {
         StringTokenizer tokenizer = new StringTokenizer(value.toString());
         while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, 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(WordCountJobScheduler.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);
  }
}
```

OUTPUT:

```
Hello world hello
World is round
Hello again

Hello   3
World   2
is      1
```

round 1
again 1

RESULT:
Thus the map reduce application on Hadoop cluster was executed on apache beans software successfully.