**Big Data Analytics**

**BAD601**

**Experiment 4:**

Develop a MapReduce program to find the tags associated with each movie by analyzing movie lens data.

**Open Notepad and write**

import java.io.IOException;

import java.util.StringJoiner;

import java.util.HashMap;

import java.util.Map;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

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 MovieTags {

// Mapper Class

public static class TagMapper extends Mapper<LongWritable, Text, Text, Text> {

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

String line = value.toString();

String[] fields = line.split(",");

if (fields.length >= 4 && !fields[0].equals("userId")) { // Skip header

String movieTitle = fields[2]; // Extract movie title

String tag = fields[3]; // Extract tag

context.write(new Text(movieTitle), new Text(tag)); // Emit movieTitle -> tag

}

}

}

// Reducer Class

public static class TagReducer extends Reducer<Text, Text, Text, Text> {

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

StringJoiner tags = new StringJoiner(", ");

for (Text val : values) {

tags.add(val.toString());

}

context.write(key, new Text(tags.toString())); // Emit movieTitle -> tags

}

}

// Driver Code

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Movie Tags");

job.setJarByClass(MovieTags.class);

job.setMapperClass(TagMapper.class);

job.setReducerClass(TagReducer.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

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

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

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

}

}

**Save the Program:**

Click of Save and save the program with the name MovieTags.java

**Make a CSV file and save as tags.csv:**

userId,movieId,movieTitle,tag,timestamp

1,296,Terminator 2: Judgment Day (1991),funny,1139045764

2,306,Pulp Fiction (1994),dark humor,1255630284

3,307,The Matrix (1999),sci-fi,1343294286

4,307,The Matrix (1999),futuristic,1343294287

5,308,Se7en (1995),thriller,1455639852

6,308,Se7en (1995),suspense,1467890234

7,309,Inception (2010),mind-bending,1501234567

8,309,Inception (2010),dreams,1501237890

9,310,The Dark Knight (2008),gritty,1522345678

10,310,The Dark Knight (2008),masterpiece,1522348999

**Change the directory from cmd:**

Go to the directory where the code is saved using cd

**Get the Hadoop class path:**

hadoop classpath

**Compile the Java File**

Now, compile your Java program using the copied classpath:

javac -classpath "PASTE\_HADOOP\_CLASSPATH\_HERE" -d . MovieTags.java

**Create a JAR File**

Once compilation is successful, create the JAR file:

jar cf movietags.jar MovieTags\*.class

**Upload Input Files to HDFS**

start-dfs.cmd

start-yarn.cmd

hdfs dfs -mkdir -p /user/hadoop/movie

hdfs dfs -put tags.csv /user/hadoop/movie/

**Run the Hadoop Job**

hadoop jar movietags.jar MovieTags /user/hadoop/movie/tags.csv /output/movietags

**View the Output**

hdfs dfs -cat /output/movietags/part-r-00000