Enroll_No – 202300819010093

Name – Vishv Patel

Q- 1)

```
import java.io.IOException; 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;


public class WordCount {



    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 {
            String[] tokens = value.toString().split("\\s+");
for (String token : tokens) {
word.set(token);                context.write(word, one);
            }
        }
    }



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



    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
Job job = Job.getInstance(conf, "word count");
job.setJarByClass(WordCount.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);
    }
}
```

Q- 2)

```
import java.io.IOException;
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; public
class MinTemperature {



   public static class TempMapper extends Mapper<Object, Text, Text, IntWritable> {
private Text year = new Text();
       private IntWritable temperature = new IntWritable();


       public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
           String[] fields = value.toString().split("\\s+");
if (fields.length == 2) {                year.set(fields[0]);
               temperature.set(Integer.parseInt(fields[1]));
context.write(year, temperature);
           }
       }
   }



   public static class MinTempReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {
       public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException {            int minTemp = Integer.MAX_VALUE;
for (IntWritable val : values) {               minTemp = Math.min(minTemp, val.get());
           }
           context.write(key, new IntWritable(minTemp));
      }
```

```
    }


    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "minimum temperature");
job.setJarByClass(MinTemperature.class);
job.setMapperClass(TempMapper.class);
job.setReducerClass(MinTempReducer.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);
    }
}
```

Q- 3)

```
 import java.io.IOException; import
org.apache.hadoop.conf.Configuration; 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.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


public class AverageTokenCount {


    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 {
            String[] tokens = value.toString().split("\\s+");
for (String token : tokens) {
word.set(token);                context.write(word, one);
            }
            context.write(new Text("**LINE_COUNT**"), new IntWritable(tokens.length));
        }
    }
```

```
   public static class WordCountReducer 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();
          }
          if (!key.toString().equals("**LINE_COUNT**")) {
result.set(sum);                context.write(key, result);
          } else {
              context.write(new Text("TOTAL_TOKENS"), new IntWritable(sum));
          }
       }
   }


   public static class AverageReducer extends Reducer<Text, IntWritable, Text, Text> {
private int totalTokens = 0;         private int wordCount = 0;

       public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException {             if
(key.toString().equals("TOTAL_TOKENS")) {                for (IntWritable val : values) {
totalTokens = val.get();
```

```
            }
        } else {
            for (IntWritable val : values) {
             wordCount++;
            }
        }
    }

    @Override    protected void cleanup(Context context) throws IOException,
InterruptedException
{
        float averageCount = (float) totalTokens / wordCount;
        context.write(new Text("AverageCount"), new Text("=" + averageCount));
    }
  }


  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "average token count");
job.setJarByClass(AverageTokenCount.class);
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(WordCountReducer.class);
job.setReducerClass(AverageReducer.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);
    }
}
```

Q- 4)

```
import java.io.IOException; 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;

public class TokenCount {
```

```
   public static class TokenMapper extends Mapper<Object, Text, Text, IntWritable> {
private final static IntWritable one = new IntWritable(1);         private Text token =
new Text("TOKEN_COUNT");


       public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
           String[] tokens = value.toString().split("\\s+");
for (String word : tokens) {

              if (word.length() >= 4) {
               context.write(token, one);
              }
          }
       }
   }



   public static class TokenReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {
       public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException {
           int sum = 0;
           for (IntWritable val : values) {
sum += val.get();
           }
           context.write(new Text("Total count for token"), new IntWritable(sum));
       }
   }
```

```
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
Job job = Job.getInstance(conf, "token count");
job.setJarByClass(TokenCount.class);
job.setMapperClass(TokenMapper.class);
job.setReducerClass(TokenReducer.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);
    }
}
```

Q- 5)

```
import java.io.IOException; 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;

public class FemaleVoterSimple {


   public static class VoterMapper extends Mapper<Object, Text, Text, IntWritable> {

private final static IntWritable one = new IntWritable(1);     private Text femaleKey = new

Text("FemaleVoterCount");       public void map(Object key, Text value, Context context)

throws IOException,

InterruptedException {
          String[] fields = value.toString().split(",");            if (fields.length
== 4 && fields[2].equalsIgnoreCase("Female")) {
context.write(femaleKey, one);
          }
      }
   }
```

```
   public static class VoterReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {

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

int totalFemales = 0;            for

(IntWritable val : values) {

totalFemales += val.get();

         }

         context.write(new Text("No. of female voters are: "), new
IntWritable(totalFemales));

      }

   }



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

      Configuration conf = new Configuration();

      Job job = Job.getInstance(conf, "female voter count");

job.setJarByClass(FemaleVoterSimple.class);

job.setMapperClass(VoterMapper.class);

job.setReducerClass(VoterReducer.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);

   }
```

}

Q- 6)

```
import java.io.IOException;
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; public
class UserReviewCount {


    public static class ReviewMapper extends Mapper<Object, Text, Text, IntWritable> {
private final static IntWritable one = new IntWritable(1);         private Text userId = new
Text();


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

```
            String[] fields = value.toString().split(",");
if (fields.length > 0) {
userId.set(fields[0].trim());
context.write(userId, one);
            }
        }
    }

    public static class ReviewReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {

        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException {
int totalReviews = 0;              for
(IntWritable val : values) {
totalReviews += val.get();
            }
            context.write(key, new IntWritable(totalReviews));
        }
    }


    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();         Job
job = Job.getInstance(conf, "user review count");
job.setJarByClass(UserReviewCount.class);
job.setMapperClass(ReviewMapper.class);
```

```
job.setReducerClass(ReviewReducer.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);
    }
}
```

Q- 7)

7.1)

```
import java.io.IOException; import
org.apache.hadoop.conf.Configuration; import
org.apache.hadoop.fs.Path; 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;
```

```
public class ComedyMovies {


   public static class ComedyMapper extends Mapper<Object, Text, Text, Text> {


      public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
         String[] fields = value.toString().split(",");          if
(fields.length > 2 && fields[2].contains("Comedy")) {
context.write(new Text(fields[1]), value);
         }
      }
   }


   public static class IdentityReducer extends Reducer<Text, Text, Text, Text> {
      public void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
         for (Text val : values) {
context.write(key, val);
         }
      }
   }
```

```
        Job job = Job.getInstance(conf, "comedy movies");
        job.setJarByClass(ComedyMovies.class);
        job.setMapperClass(ComedyMapper.class);
job.setReducerClass(IdentityReducer.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);
    }
}
```

7.2)

```
import java.io.IOException; 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;
```

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



public class DocumentaryMovies1995 {


    public static class DocumentaryMapper extends Mapper<Object, Text, Text,
IntWritable> {         private final static IntWritable one = new IntWritable(1);
private Text word = new Text("Documentary_1995");


        public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
            String[] fields = value.toString().split(",");
            if (fields.length > 2 && fields[2].contains("Documentary") &&
fields[1].contains("(1995)")) {                 context.write(word, one);
            }
        }
    }


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

```
    }
        Job job = Job.getInstance(conf, "documentary count 1995");
        job.setJarByClass(DocumentaryMovies1995.class);
        job.setMapperClass(DocumentaryMapper.class);
job.setReducerClass(SumReducer.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);
    }
}
```

7.3)

```
import java.io.IOException; 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.FileInp
```

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



utFormat; import
org.apache.hadoop.mapreduce.lib.output.FileO
utputFormat;


public class MissingGenresCount {


    public static class MissingGenresMapper extends Mapper<Object, Text, Text,
IntWritable> {
        private final static IntWritable one = new IntWritable(1);
private Text word = new Text("Missing_Genres");


        public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
            String[] fields = value.toString().split(",");
if (fields.length < 3 || fields[2].trim().isEmpty()) {
context.write(word, one);
            }
        }
    }


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

```
            int sum = 0;
            for (IntWritable val : values) {
sum += val.get();
            }
            context.write(key, new IntWritable(sum));
        }
    }
        Job job = Job.getInstance(conf, "missing genres count");
        job.setJarByClass(MissingGenresCount.class);
        job.setMapperClass(MissingGenresMapper.class);
job.setReducerClass(SumReducer.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);
    }
}
```

7.4)

```
import java.io.IOException; import
org.apache.hadoop.conf.Configuration; import
org.apache.hadoop.fs.Path; import org.apache.hadoop.io.Text;
```

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



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 GoldMovies {


    public static class GoldMapper extends Mapper<Object, Text, Text, Text> {
        public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
            String[] fields = value.toString().split(",");             if
(fields.length > 1 && fields[1].contains("Gold")) {
context.write(new Text(fields[1]), value);
            }
        }
    }


    public static class IdentityReducer extends Reducer<Text, Text, Text, Text> {
        public void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
            for (Text val : values) {
context.write(key, val);
```

```
        }
      }
   }

   public static void main(String[] args) throws Exception {
      Configuration conf = new Configuration();
Job job = Job.getInstance(conf, "gold movies");
job.setJarByClass(GoldMovies.class);
job.setMapperClass(GoldMapper.class);
job.setReducerClass(IdentityReducer.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);
    }
}
```

7.5)

```
import java.io.IOException; 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;

public class DramaRomanceMovies {
```

```
    public static class DramaRomanceMapper extends Mapper<Object, Text, Text,
IntWritable> {
        private final static IntWritable one = new IntWritable(1);
private Text word = new Text("Drama_Romance");

        public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
            String[] fields = value.toString().split(",");
            if (fields.length > 2 && fields[2].contains("Drama") &&
fields[2].contains("Romance")) {
context.write(word, one);
            }
        }
    }

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

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

```
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "drama romance movies count");
job.setJarByClass(DramaRomanceMovies.class);
job.setMapperClass(DramaRomanceMapper.class);
job.setReducerClass(SumReducer.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);
    }
}
```