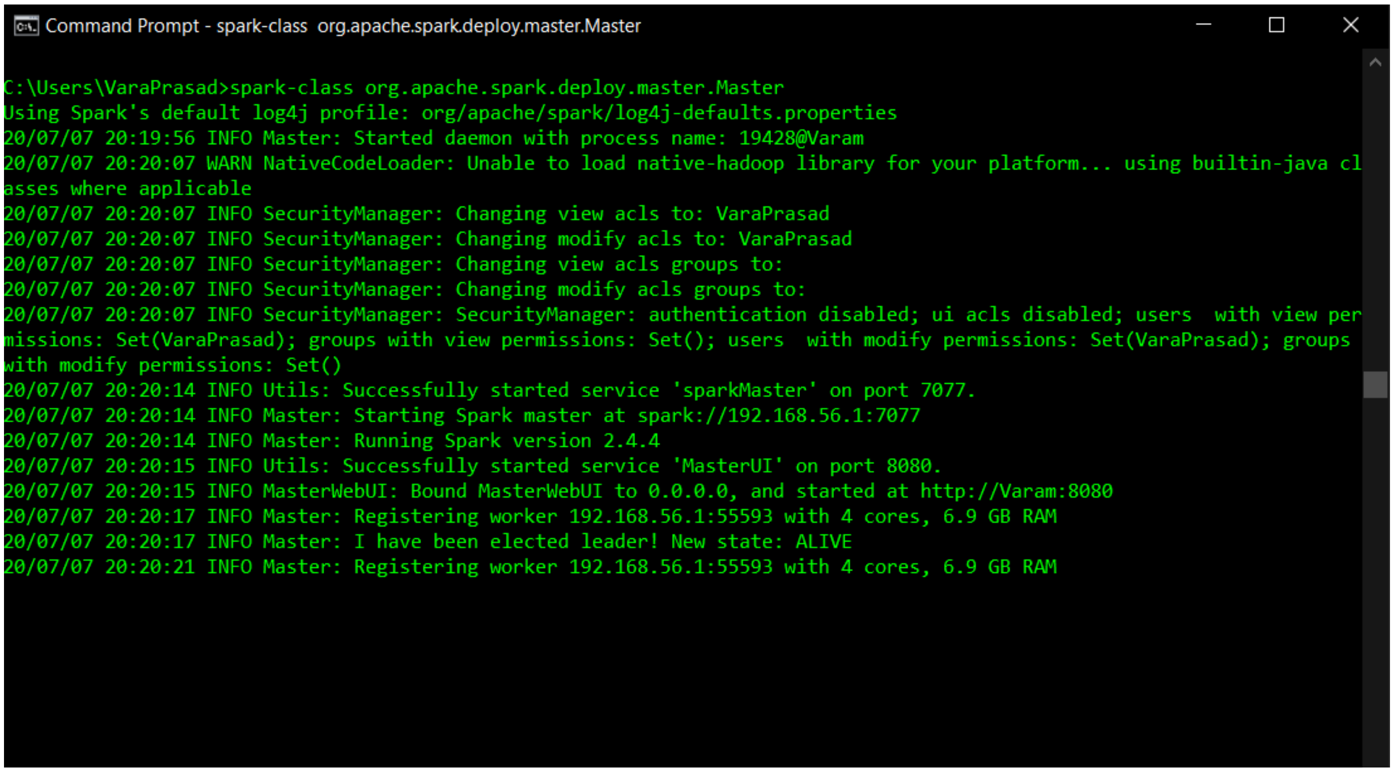
BigData Programming – Assignment 5

Varaprasad Kurra **002430487**

1. **SparkMinHash.java**

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.Function;
import org.apache.spark.api.java.function.Function2;
import org.apache.spark.api.java.function.PairFlatMapFunction;
import org.apache.spark.api.java.function.PairFunction;
import org.apache.spark.api.java.function.VoidFunction;
import org.apache.spark.broadcast.Broadcast;
import org.apache.spark.ml.feature.CountVectorizer;
import org.apache.spark.ml.feature.CountVectorizerModel;
import org.apache.spark.ml.linalg.Vector;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.RowFactory;
import org.apache.spark.sql.SparkSession;
import org.apache.spark.sql.types.DataTypes;
import org.apache.spark.sql.types.StructField;
import org.apache.spark.sql.types.StructType;
import scala.Tuple2;

public class SparkMinHashLSH
 {
      private static final String FILE_URI =
"file:///C:/Users/VaraPrasad/Desktop/Summer_Semester/LSH_Data_File";
      private static final double sizeAdj = 1.0;
      private static class JaccordSimilarity_class implements
PairFunction<Tuple2<Row,Row>,String,Double>
      {
                 public Tuple2<String, Double> call(Tuple2<Row,Row> arg0)
                 {

                  double similar_items=0.0;
                  double total_items=0.0;

                  String second_file_in_row = arg0._1().getString(0);
                  String first_file_in_row  = arg0._2().getString(0);

      double first_file_values []= ((Vector)arg0._1().get(1)).toArray();
      double second_file_values [] = ((Vector)arg0._2().get(1)).toArray();

                    Set<String> s1 = new LinkedHashSet<String>();
                for (int i = 0; i < first_file_values.length; i++)
                    {
                           s1.add(String.valueOf(first_file_values[i]));
                          }

                Set<String> s2 = new LinkedHashSet<String>();
                for (int i = 0; i < first_file_values.length; i++)
                    {
                           s1.add(String.valueOf(second_file_values[i]));
                          }

                Set<String> intersection = new LinkedHashSet<>(s1);
                intersection.retainAll(s2);
                Set<String> union = new LinkedHashSet<>(s1);
                union.addAll(s2);
                double similarity = (double)intersection.size() /
(double)union.size();
                    return new Tuple2<String, Double>(first_file_in_row + " , "
+ second_file_in_row, similarity);
                 }
                }

                        //System.out.println(first_file_in_row+"
"+second_file_in_row);
                        //System.out.println(first_file_values[i]+"
"+second_file_values[i]);

                    double similarity = (double)intersection/(double)Union;
                    return new Tuple2<String, Double>(first_file_in_row + " , "
+ second_file_in_row, similarity);
                }
              }


      private static class Cartesian_unique implements
Function<Tuple2<Row,Row>,Boolean>
      {
              public Boolean call(Tuple2<Row, Row> arg0) throws Exception
              {

                    boolean bool;
                    //Get First File name
                  String File1 = arg0._1().getString(0);
                  //Get Second File name
                  String File2 = arg0._2().getString(0);

                 // System.out.println("First File for Comparision is "+File1);
                 //System.out.println("Second File for comparision is "+File2);

                  //Compare the File name Lexically
                  if( bool = File1.compareToIgnoreCase(File2)>0?true: false);
                 return bool;
          }
      }
      public static void main(String[] args)
      {
              SparkSession spark =
SparkSession.builder().config("spark.master","local[*]").getOrCreate();
              JavaSparkContext sc = new JavaSparkContext(spark.sparkContext());
              sc.setLogLevel("WARN");

              JavaPairRDD<String,String> documents =
sc.wholeTextFiles(FILE_URI);
              System.out.println("Keys in Total Documents
"+documents.take((int)documents.count()).toString());

              class ShinglesCreator implements Function<String,String[]>
              {
                    public String[] call(String text) throws Exception
                    {
                          return ShingleUtils.getTextShingles(text);
                    }
              }

              JavaPairRDD<String,String[]> shinglesDocs =
documents.mapValues(new ShinglesCreator());

              shinglesDocs.values().foreach(new VoidFunction<String[]>()
                          {
                                public void call(String[] shingles) throws
Exception
                                    {
                                          for ( int i = 0; i <
shingles.length; i ++ )
                                      {
                                                System.out.print(shingles[i]
+ "|");
                                          }
                    System.out.println("\n");
                  }
              });

              // create characteristic matrix representation of each document
              StructType schema = new StructType(
                          new StructField[]
                                    {

      DataTypes.createStructField("file_path", DataTypes.StringType, false),

      DataTypes.createStructField("file_content",DataTypes.createArrayType(Da
taTypes.StringType, false),false)
                                 }
                    );

              Dataset<Row> df = spark.createDataFrame(shinglesDocs.map( new
Function<Tuple2<String, String[]>, Row>()
                          {
                                @Override
                                public Row call(Tuple2<String, String[]>
record)
                                {
                                    return
RowFactory.create(record._1().substring(record._1().lastIndexOf("/")+1),
record._2());
                                }
                          }), schema);
              df.show(true);

              CountVectorizer vectorizer = new
CountVectorizer().setInputCol("file_content").setOutputCol("feature_vector").
setBinary(true);
              CountVectorizerModel cvm = vectorizer.fit(df);
              final Broadcast<Integer> vocabSize =
sc.broadcast(cvm.vocabulary().length);

              System.out.println("vocab size = " + cvm.vocabulary().length);
              for (int i = 0; i < vocabSize.value(); i ++ )
              {
                    System.out.print(cvm.vocabulary()[i] + "(" + i + ") ");
              }
              System.out.println();

              Dataset<Row> characteristicMatrix = cvm.transform(df);
              System.out.println("Characteristic Matrix is");
              characteristicMatrix.show(true);


              // create minhashSignature for each document
              final Broadcast<Double> sSize = sc.broadcast(sizeAdj);
              JavaPairRDD<String,List<Integer>> minhashSignature =
characteristicMatrix.toJavaRDD().mapToPair(new PairFunction<Row, String,
List<Integer>>()
                          {
                        public Tuple2<String, List<Integer>> call(Row row)
throws Exception
                        {
                        int signatureMatrixRowNum =
(int)(vocabSize.value()*sSize.value());
                        //System.out.println("Signature Matrix Row Number
"+signatureMatrixRowNum);
                        MinHashHelper mh = new
MinHashHelper(vocabSize.value()); // vocabSize.value(): number of rows the
signature matrix
                        double[] characteristicMatrixRow =
((Vector)row.getAs("feature_vector")).toArray();

                        List<Integer> signatureVector = new
ArrayList<Integer>();

                        for ( int i = 0; i < signatureMatrixRowNum; i ++ )
                            {
                                int[] p = mh.getPermutation();
                                int flag = Integer.MAX_VALUE;
                          for ( int j = 0; j < characteristicMatrixRow.length;
j ++ )
                          {
                                if ( characteristicMatrixRow[j] == 1.0 && flag
> p[j] )
                                {
                                    flag = p[j];
                                }
                          }
                          signatureVector.add(flag);
                    }

                      return new Tuple2<String, List<Integer>>((String)
row.getAs("file_path"), signatureVector);
                }
              });

              //System.out.println("Minhash signatures:");

        //System.out.println(minhashSignature.take((int)minhashSignature.count(
)).toString());

              // LSH implementation using hashCode
              class LocalSensitiveHashingImpl implements PairFlatMapFunction
<Tuple2<String, List<Integer>>, String, String>
              {
                    private final int ROWS  = 16;  // r factor

                public Iterator<Tuple2<String, String>> call(Tuple2<String,
List<Integer>> signatureColumn) throws Exception

                {
                    String documentName = signatureColumn._1;
                    int BANDS = signatureColumn._2.size()/ROWS;  // b factor
                    String signatureVector =
signatureColumn._2.toString().replaceAll("[^\\d.]", "");  // only the
signature left here

                    List<Tuple2<String, String>> lsh = new ArrayList<>();

                    for ( int i = 0; i < BANDS; i++ )
                    {
                          String singleBand = signatureVector.substring(i*ROWS,
(i+1)*ROWS);
                          lsh.add(new Tuple2<>("BAND-" + i + "-[" +
singleBand.hashCode() + "]", documentName));
                    }
                      return lsh.iterator();
                }
              }
              JavaPairRDD<String,String> minhashResult =
minhashSignature.flatMapToPair(new LocalSensitiveHashingImpl());
              //System.out.println("Minhash results:");

        //System.out.println(minhashResult.take((int)minhashResult.count()).toS
tring());


              // finally
              JavaPairRDD<String,String> similarDocuments =
minhashResult.reduceByKey(new Function2<String,String,String>()
                          {
                    public String call(final String document1,final String
document2)
                          {
                          return document1 + "," + document2;
                    }
              }).filter(new Function<Tuple2<String, String>,Boolean>()
                          {
                    public Boolean call(Tuple2<String, String> bucketDocument)
                    {
                          if ( bucketDocument._2.contains(","))
                                return true;
                          else
                                return false;
                    }
              });

              System.out.println("===> FINAL:"+
similarDocuments.take((int)similarDocuments.count()).toString());

              JavaRDD<Row> path_and_vector =
characteristicMatrix.select("file_path", "feature_vector").toJavaRDD();
        //System.out.println("Fields required for calculating Carteesian
Product");

        //System.out.println(path_and_vector.take((int)path_and_vector.count())
.toString());


              JavaPairRDD<Row,Row> CartesionProd =
path_and_vector.cartesian(path_and_vector);
        //System.out.println("Rows after Catesian Product are");

        //System.out.println(CartesionProd.take((int)CartesionProd.count()).toS
tring());

              JavaPairRDD<Row,Row> CartesionProd_Unique = CartesionProd.filter(
new Cartesian_unique());

              System.out.println("Rows after Catesian Product are");

        System.out.println(CartesionProd_Unique.take((int)CartesionProd_Unique.
count()).toString());
              JavaPairRDD<String,Double> JaccordSimilarity =
CartesionProd_Unique.mapToPair(new JaccordSimilarity_class());
              System.out.println("");
              System.out.println("Jaccord Similarity");

        System.out.println(JaccordSimilarity.take((int)JaccordSimilarity.count(
)).toString());


              vocabSize.unpersist();
              vocabSize.destroy();
              sSize.unpersist();
              sSize.destroy();
              sc.close();

        }
}
```

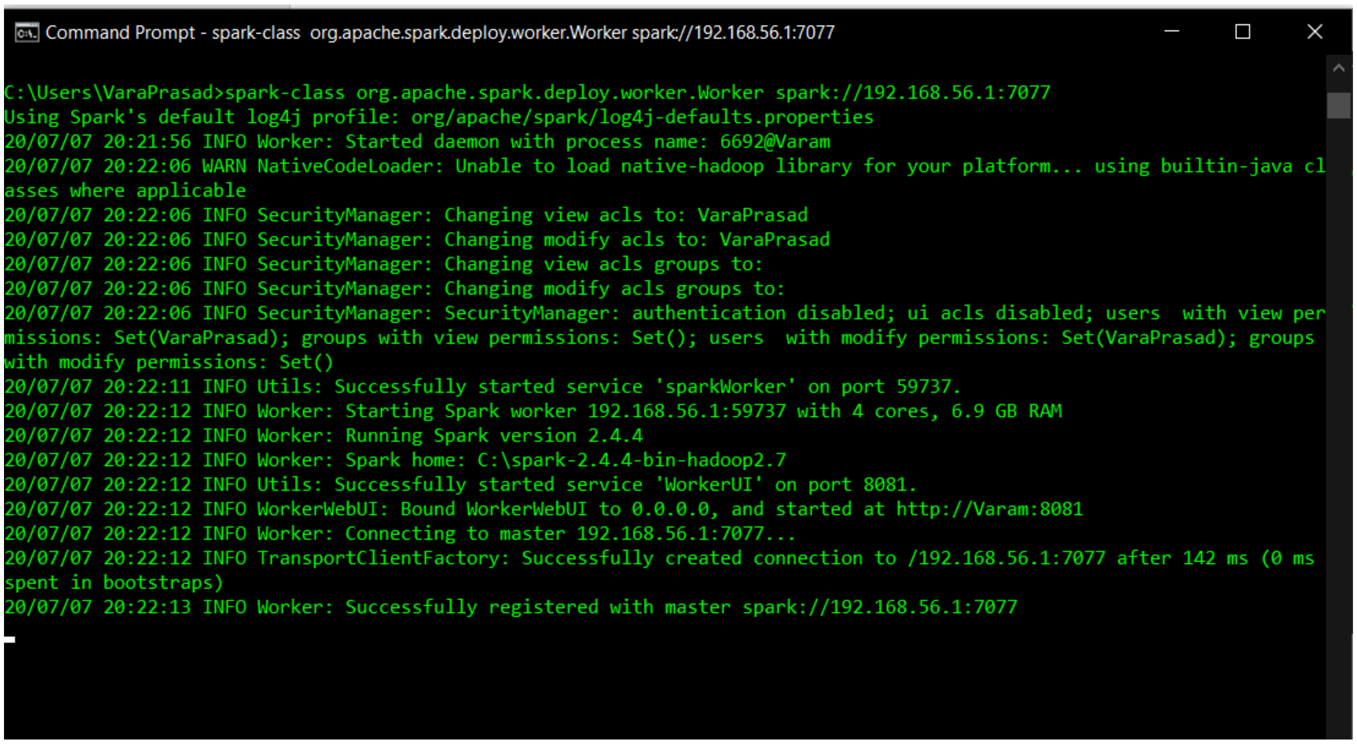
2. **Screen Shot showing the initiation of Master Node.**



Command Prompt - spark-class org.apache.spark.deploy.master.Master

```
C:\Users\VaraPrasad>spark-class org.apache.spark.deploy.master.Master
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
20/07/07 20:19:56 INFO Master: Started daemon with process name: 19428@Varam
20/07/07 20:20:07 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
20/07/07 20:20:07 INFO SecurityManager: Changing view acls to: VaraPrasad
20/07/07 20:20:07 INFO SecurityManager: Changing modify acls to: VaraPrasad
20/07/07 20:20:07 INFO SecurityManager: Changing view acls groups to:
20/07/07 20:20:07 INFO SecurityManager: Changing modify acls groups to:
20/07/07 20:20:07 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view per
missions: Set(VaraPrasad); groups with view permissions: Set(); users  with modify permissions: Set(VaraPrasad); groups
with modify permissions: Set()
20/07/07 20:20:14 INFO Utils: Successfully started service 'sparkMaster' on port 7077.
20/07/07 20:20:14 INFO Master: Starting Spark master at spark://192.168.56.1:7077
20/07/07 20:20:14 INFO Master: Running Spark version 2.4.4
20/07/07 20:20:15 INFO Utils: Successfully started service 'MasterUI' on port 8080.
20/07/07 20:20:15 INFO MasterWebUI: Bound MasterWebUI to 0.0.0.0, and started at http://Varam:8080
20/07/07 20:20:17 INFO Master: Registering worker 192.168.56.1:55593 with 4 cores, 6.9 GB RAM
20/07/07 20:20:17 INFO Master: I have been elected leader! New state: ALIVE
20/07/07 20:20:21 INFO Master: Registering worker 192.168.56.1:55593 with 4 cores, 6.9 GB RAM
```

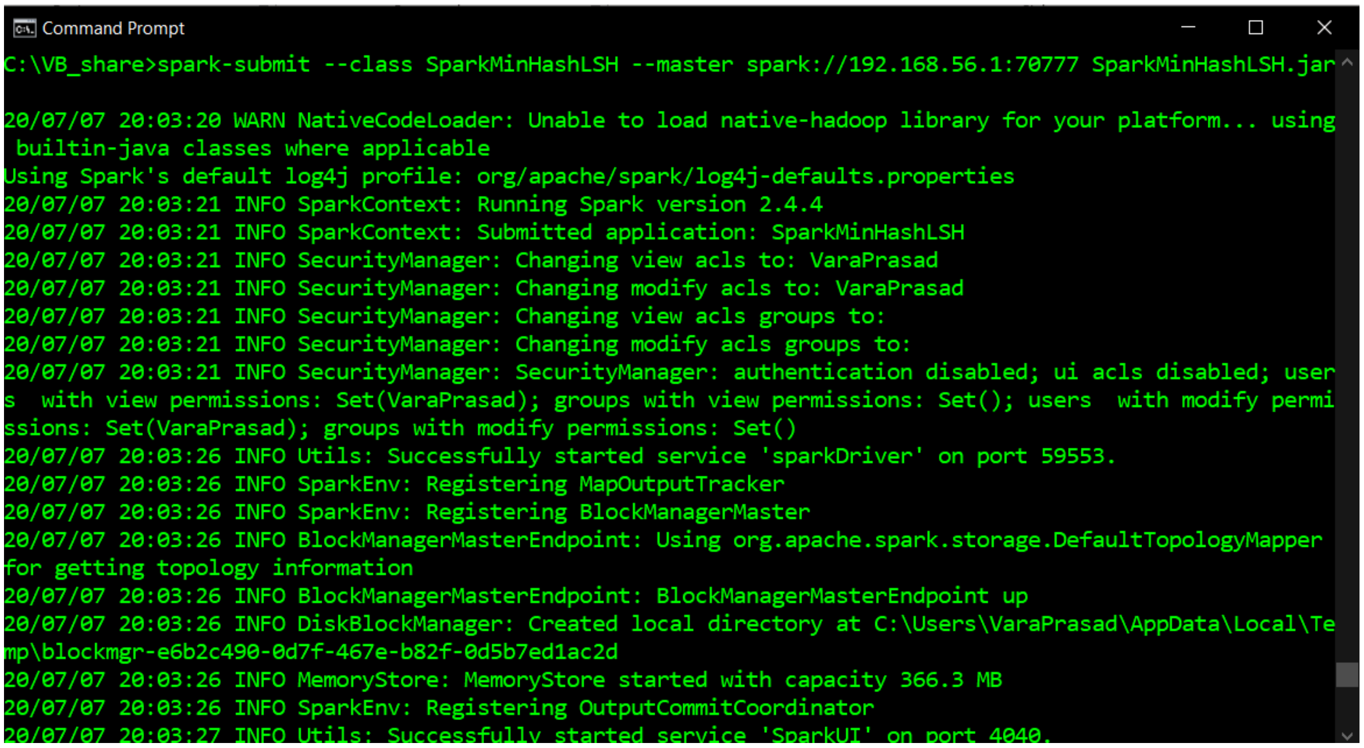
3. **Screen Shot Showing the Worker node initiation.**



Command Prompt - spark-class org.apache.spark.deploy.worker.Worker spark://192.168.56.1:7077

```
C:\Users\VaraPrasad>spark-class org.apache.spark.deploy.worker.Worker spark://192.168.56.1:7077
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
20/07/07 20:21:56 INFO Worker: Started daemon with process name: 6692@Varam
20/07/07 20:22:06 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
20/07/07 20:22:06 INFO SecurityManager: Changing view acls to: VaraPrasad
20/07/07 20:22:06 INFO SecurityManager: Changing modify acls to: VaraPrasad
20/07/07 20:22:06 INFO SecurityManager: Changing view acls groups to:
20/07/07 20:22:06 INFO SecurityManager: Changing modify acls groups to:
20/07/07 20:22:06 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view per
missions: Set(VaraPrasad); groups with view permissions: Set(); users  with modify permissions: Set(VaraPrasad); groups
with modify permissions: Set()
20/07/07 20:22:11 INFO Utils: Successfully started service 'sparkWorker' on port 59737.
20/07/07 20:22:12 INFO Worker: Starting Spark worker 192.168.56.1:59737 with 4 cores, 6.9 GB RAM
20/07/07 20:22:12 INFO Worker: Running Spark version 2.4.4
20/07/07 20:22:12 INFO Worker: Spark home: C:\spark-2.4.4-bin-hadoop2.7
20/07/07 20:22:12 INFO Utils: Successfully started service 'WorkerUI' on port 8081.
20/07/07 20:22:12 INFO WorkerWebUI: Bound WorkerWebUI to 0.0.0.0, and started at http://Varam:8081
20/07/07 20:22:12 INFO Worker: Connecting to master 192.168.56.1:7077...
20/07/07 20:22:12 INFO TransportClientFactory: Successfully created connection to /192.168.56.1:7077 after 142 ms (0 ms
spent in bootstraps)
20/07/07 20:22:13 INFO Worker: Successfully registered with master spark://192.168.56.1:7077
```

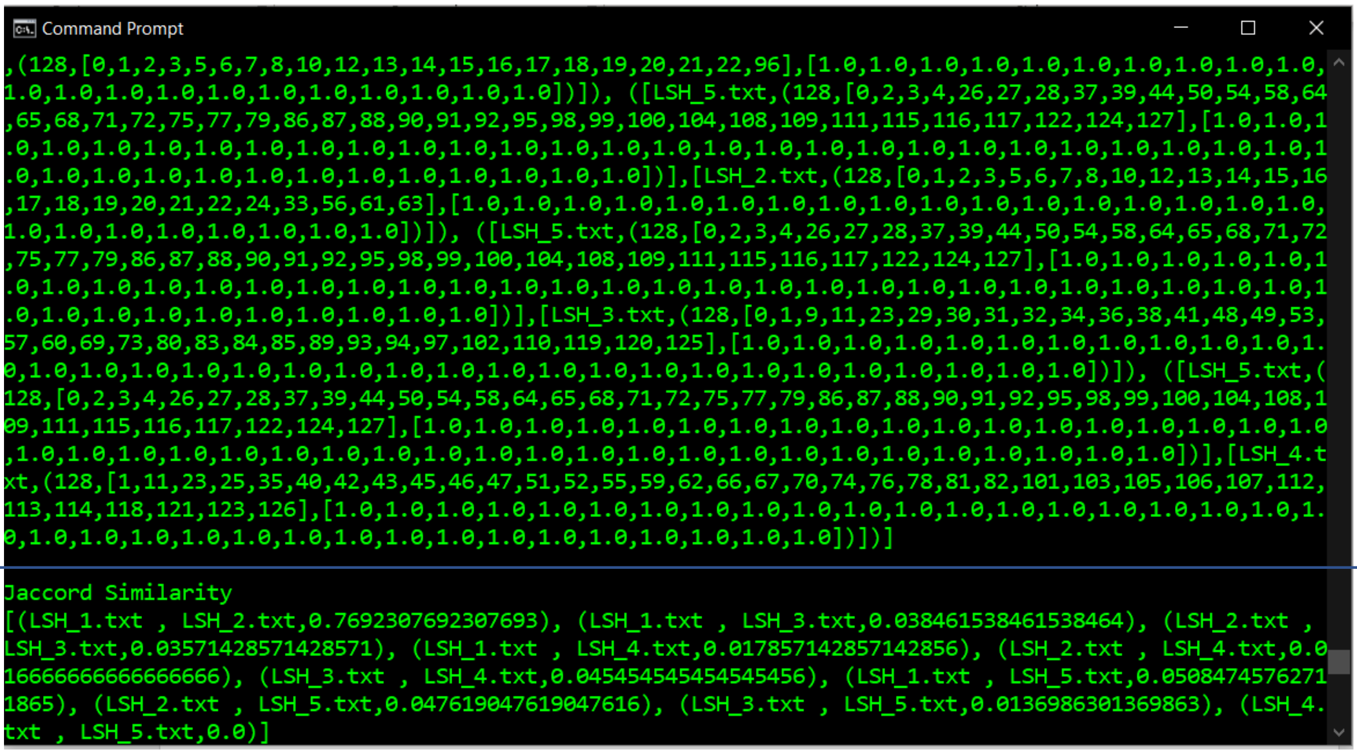
4. **Screen Shot showing the submission of Job.**



Command Prompt

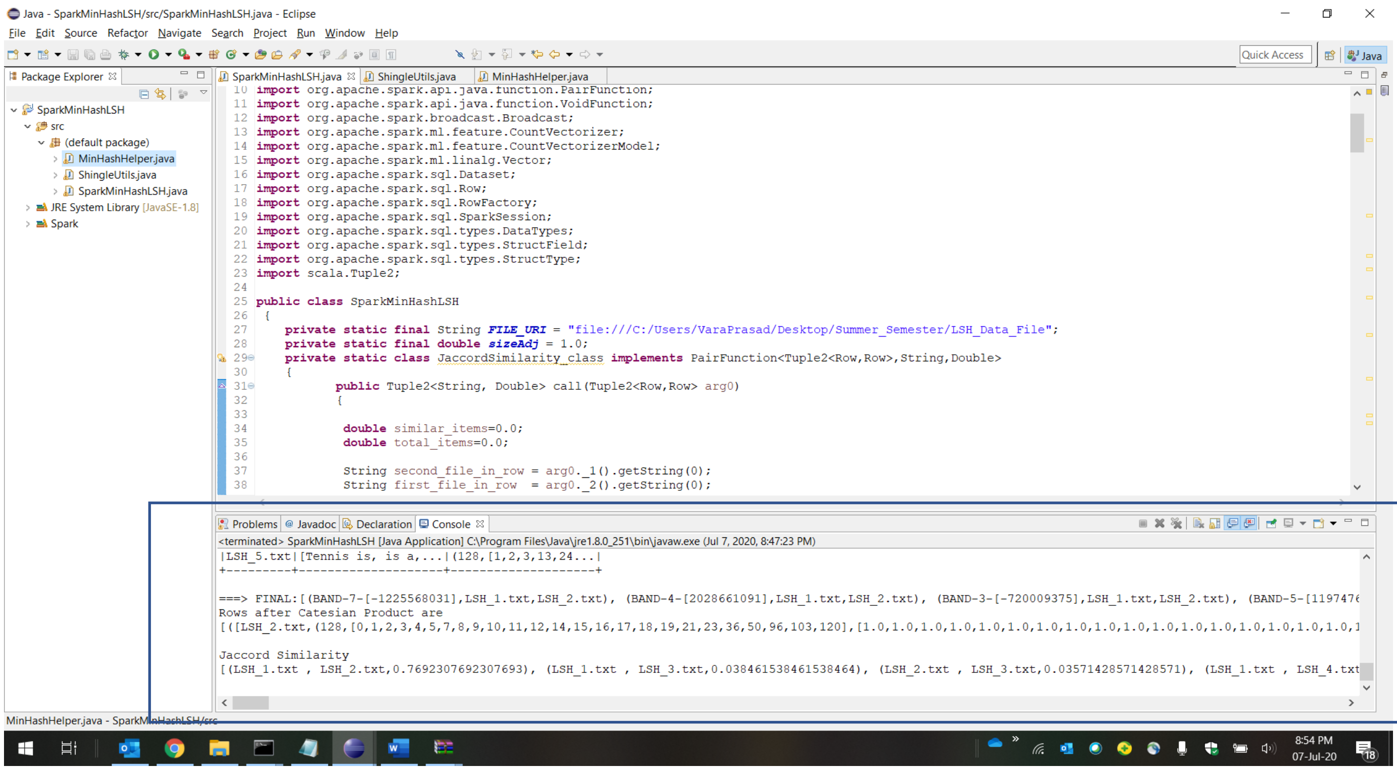
```
C:\VB_share>spark-submit --class SparkMinHashLSH --master spark://192.168.56.1:70777 SparkMinHashLSH.jar

20/07/07 20:03:20 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using
 builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
20/07/07 20:03:21 INFO SparkContext: Running Spark version 2.4.4
20/07/07 20:03:21 INFO SparkContext: Submitted application: SparkMinHashLSH
20/07/07 20:03:21 INFO SecurityManager: Changing view acls to: VaraPrasad
20/07/07 20:03:21 INFO SecurityManager: Changing modify acls to: VaraPrasad
20/07/07 20:03:21 INFO SecurityManager: Changing view acls groups to:
20/07/07 20:03:21 INFO SecurityManager: Changing modify acls groups to:
20/07/07 20:03:21 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; user
s  with view permissions: Set(VaraPrasad); groups with view permissions: Set(); users  with modify permi
ssions: Set(VaraPrasad); groups with modify permissions: Set()
20/07/07 20:03:26 INFO Utils: Successfully started service 'sparkDriver' on port 59553.
20/07/07 20:03:26 INFO SparkEnv: Registering MapOutputTracker
20/07/07 20:03:26 INFO SparkEnv: Registering BlockManagerMaster
20/07/07 20:03:26 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper
for getting topology information
20/07/07 20:03:26 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
20/07/07 20:03:26 INFO DiskBlockManager: Created local directory at C:\Users\VaraPrasad\AppData\Local\Te
mp\blockmgr-e6b2c490-0d7f-467e-b82f-0d5b7ed1ac2d
20/07/07 20:03:26 INFO MemoryStore: MemoryStore started with capacity 366.3 MB
20/07/07 20:03:26 INFO SparkEnv: Registering OutputCommitCoordinator
20/07/07 20:03:27 INFO Utils: Successfully started service 'SparkUI' on port 4040.
```

5. Screen Shot showing the result.



```
,(128,[0,1,2,3,5,6,7,8,10,12,13,14,15,16,17,18,19,20,21,22,96],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,
1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])]), ([LSH_5.txt,(128,[0,2,3,4,26,27,28,37,39,44,50,54,58,64
,65,68,71,72,75,77,79,86,87,88,90,91,92,95,98,99,100,104,108,109,111,115,116,117,122,124,127],[1.0,1.0,1
.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1
.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])],[LSH_2.txt,(128,[0,1,2,3,5,6,7,8,10,12,13,14,15,16
,17,18,19,20,21,22,24,33,56,61,63],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,
1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])]), ([LSH_5.txt,(128,[0,2,3,4,26,27,28,37,39,44,50,54,58,64,65,68,71,72
,75,77,79,86,87,88,90,91,92,95,98,99,100,104,108,109,111,115,116,117,122,124,127],[1.0,1.0,1.0,1.0,1.0,1
.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1
.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])],[LSH_3.txt,(128,[0,1,9,11,23,29,30,31,32,34,36,38,41,48,49,53,
57,60,69,73,80,83,84,85,89,93,94,97,102,110,119,120,125],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.
0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])]), ([LSH_5.txt,(
128,[0,2,3,4,26,27,28,37,39,44,50,54,58,64,65,68,71,72,75,77,79,86,87,88,90,91,92,95,98,99,100,104,108,1
09,111,115,116,117,122,124,127],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0
,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])],[LSH_4.t
xt,(128,[1,11,23,25,35,40,42,43,45,46,47,51,52,55,59,62,66,67,70,74,76,78,81,82,101,103,105,106,107,112,
113,114,118,121,123,126],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.
0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0])])]

Jaccord Similarity
[(LSH_1.txt , LSH_2.txt,0.7692307692307693), (LSH_1.txt , LSH_3.txt,0.038461538461538464), (LSH_2.txt ,
LSH_3.txt,0.03571428571428571), (LSH_1.txt , LSH_4.txt,0.017857142857142856), (LSH_2.txt , LSH_4.txt,0.0
16666666666666666), (LSH_3.txt , LSH_4.txt,0.045454545454545456), (LSH_1.txt , LSH_5.txt,0.0508474576271
1865), (LSH_2.txt , LSH_5.txt,0.047619047619047616), (LSH_3.txt , LSH_5.txt,0.0136986301369863), (LSH_4.
txt , LSH_5.txt,0.0)]
```

Screen shot of the result Eclipse:



```java
import org.apache.spark.api.java.function.PairFunction;
import org.apache.spark.api.java.function.VoidFunction;
import org.apache.spark.broadcast.Broadcast;
import org.apache.spark.ml.feature.CountVectorizer;
import org.apache.spark.ml.feature.CountVectorizerModel;
import org.apache.spark.ml.linalg.Vector;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.RowFactory;
import org.apache.spark.sql.SparkSession;
import org.apache.spark.sql.types.DataTypes;
import org.apache.spark.sql.types.StructField;
import org.apache.spark.sql.types.StructType;
import scala.Tuple2;

public class SparkMinHashLSH
 {
    private static final String FILE_URI = "file:///C:/Users/VaraPrasad/Desktop/Summer_Semester/LSH_Data_File";
    private static final double sizeAdj = 1.0;
    private static class JaccordSimilarity_class implements PairFunction<Tuple2<Row,Row>,String,Double>
    {
            public Tuple2<String, Double> call(Tuple2<Row,Row> arg0)
            {

             double similar_items=0.0;
             double total_items=0.0;

             String second_file_in_row = arg0._1().getString(0);
             String first_file_in_row  = arg0._2().getString(0);
```

```
<terminated> SparkMinHashLSH [Java Application] C:\Program Files\Java\jre1.8.0_251\bin\javaw.exe (Jul 7, 2020, 8:47:23 PM)
|LSH_5.txt|[Tennis is, is a,...|(128,[1,2,3,13,24...|
+---------+--------------------+--------------------+

===> FINAL:[(BAND-7-[-1225568031],LSH_1.txt,LSH_2.txt), (BAND-4-[2028661091],LSH_1.txt,LSH_2.txt), (BAND-3-[-720009375],LSH_1.txt,LSH_2.txt), (BAND-5-[119747
Rows after Catesian Product are
[([LSH_2.txt,(128,[0,1,2,3,4,5,7,8,9,10,11,12,14,15,16,17,18,19,21,23,36,50,96,103,120],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1

Jaccord Similarity
[(LSH_1.txt , LSH_2.txt,0.7692307692307693), (LSH_1.txt , LSH_3.txt,0.038461538461538464), (LSH_2.txt , LSH_3.txt,0.03571428571428571), (LSH_1.txt , LSH_4.txt
```