CSCI 5408

Assignment 3

Banner No - B00937694

Contents

GitHub Link Repo - https://git.cs.dal.ca/athaker/csci5408_s23_b00937694_abhisha_thaker/-/tree/main/A3

Problem 1A: Reuter News Data Reading & Transformation and storing in MongoDb.

Objective is lo#1 1.

1. From the two given news files (reut2-009.sgm,and reut2-014.sgm), create MongoDb Database - ReuterDb, where each Document contains a news article. The task must be done using a Java Program "ReutRead.java".

   a. To perform this operation, you need to write a Java code to scan the required texts between two<REUTERS></REUTERS> tags, <TEXT></TEXT> tags, and <TITLE></TITLE > tags.

   b. In the ReuterDb, you may consider each news as a document. You can also include nested or subdocument {

   title: "",

   text: ""

   }

Here's the file – ReutRead.java

FileContent

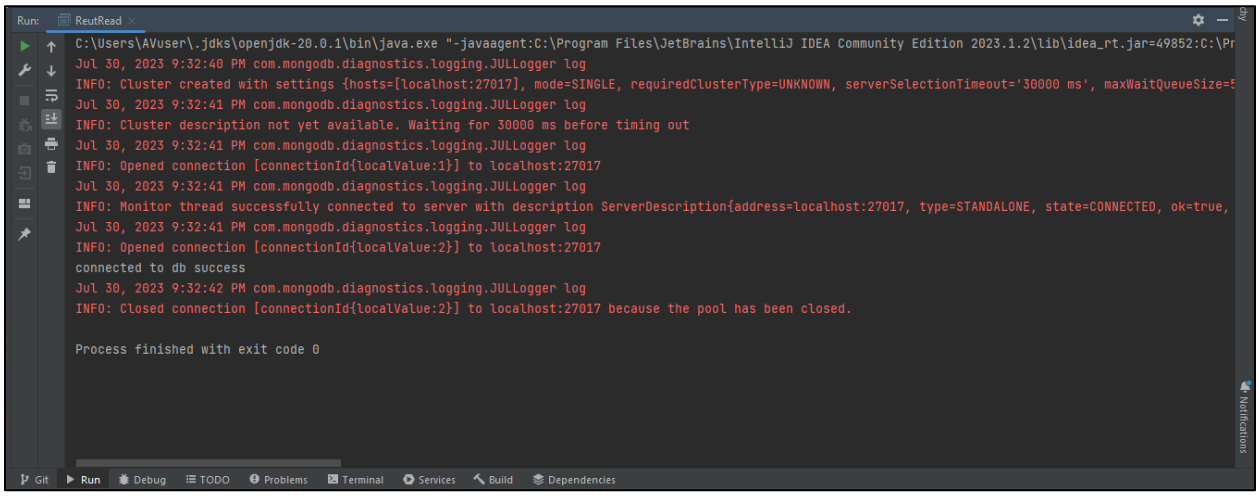
```
package org.example.problem1a;

import com.mongodb.MongoClient;
import com.mongodb.MongoClientURI;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class ReutRead {
    public static void main(String[] args) {
        String reut009 =
"C:/Users/AVuser/csci5408_s23_b00937694_abhisha_thaker/A3/src/main/resources/
reut2-009.sgm";
        String reut014 =
"C:/Users/AVuser/csci5408_s23_b00937694_abhisha_thaker/A3/src/main/resources/
reut2-014.sgm";

        String reut009Content = readFiles(reut009);
        String reut014Content = readFiles(reut014);
```

```
        String combinedContent = reut009Content + reut014Content;

        String regexText =
"<REUTERS(.*?)(.*?)>(.*?)<TEXT>(.*?)<TITLE>(.*?)<\\/TITLE>(.*?)<DATELINE>(.*?
)<\\/DATELINE>(.*?)<BODY>(.*?)<\\/BODY>(.*?)<\\/TEXT>";

        Pattern regex = Pattern.compile(regexText, Pattern.DOTALL);
        Matcher matcher = regex.matcher(combinedContent);

        String mongoConnect = "mongodb://localhost:27017/";
        MongoClientURI connectURL = new MongoClientURI(mongoConnect);

        try(MongoClient connectClient = new MongoClient(connectURL)){
            MongoDatabase db = connectClient.getDatabase("ReuterDb");
            MongoCollection<Document> collection =
db.getCollection("NewsArticles");
            Document textDocument = new Document();

            while (matcher.find()) {
                String titlecontent = matcher.group(5);
                String datecontent = matcher.group(7);
                String bodycontent = matcher.group(9);
                textDocument = new Document("title",
titlecontent).append("dateline", datecontent).append("body", bodycontent);
                Document document = new Document("title",
titlecontent).append("text",textDocument);
                collection.insertOne(document);
            }
            System.out.println("connected to db success");
            }catch(Exception e){
            e.printStackTrace();
        }
    }
    static String readFiles(String filePath)  {
        BufferedReader readFile = null;
        try {
            readFile = new BufferedReader(new FileReader(filePath));
            StringBuilder content = new StringBuilder();
            String line;

            while ((line = readFile.readLine()) != null) {
                content.append(line).append("\n");
            }

            String sgmContent = content.toString();
            return sgmContent;
        } catch(FileNotFoundException e){
            e.printStackTrace();
        } catch(IOException e){
            e.printStackTrace();
        } finally{
            try {
                readFile.close();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
```

```
        return null;
    }
}
```

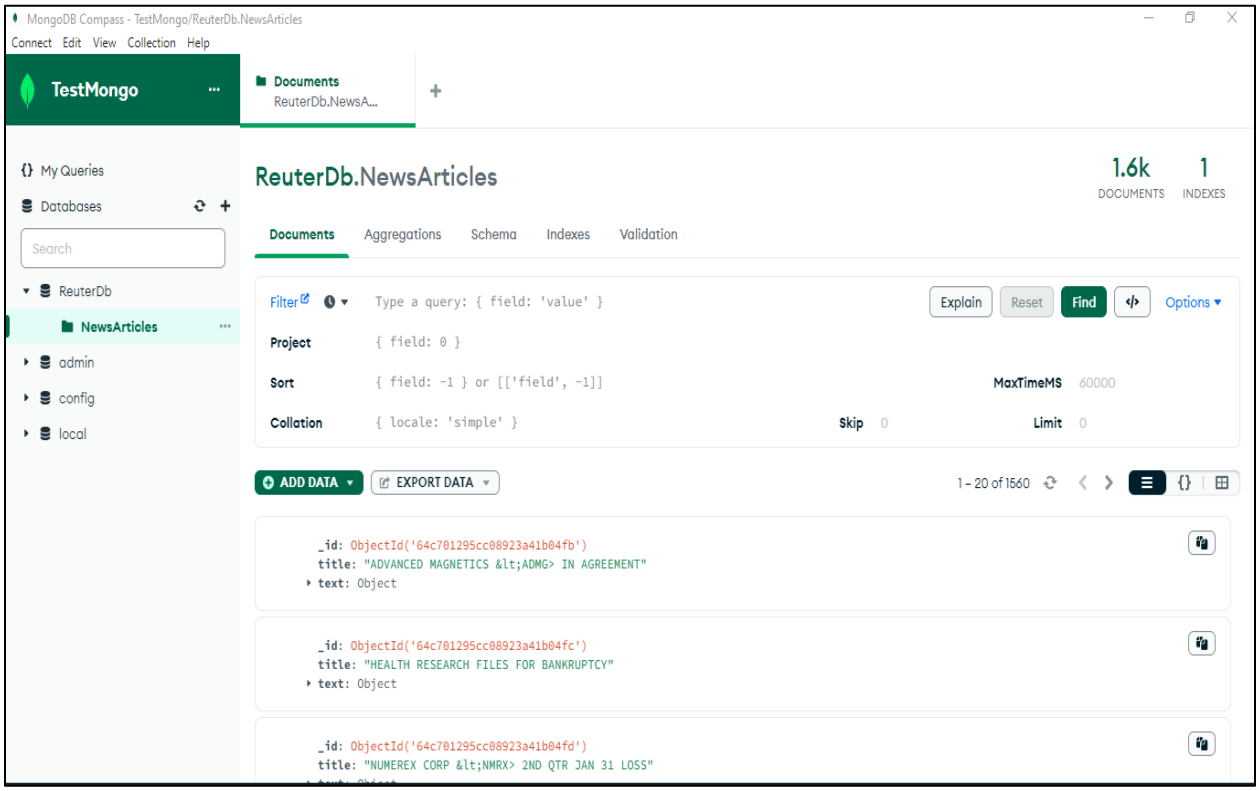
Code_Problem1a

Results In IntelliJ



Output_Problem1a_IntelliJ

In MongoDb Compass



Result_Problem1a

Have also exported json file from MongoDb compass and shared it in the zip file.

2. You need to include a flowchart and algorithm of your Reuters Data cleaning/transformation program on the PDF file.

Algorithm

1. Define the paths of two files: reut009 and reut014 and store in the string.

2. Read the content of both files by calling a separate method readFiles().

3. Combine the result of readFiles() method into a string - combinedContent. Because the format of both files is similar.

4. Define the pattern that matches with the content of the file

<REUTERS(.*?)(.*?)>(.*?)<TEXT>(.*?)<TITLE>(.*?)<\/TITLE>(.*?)<DATELINE>(.*?)<\/DATELINE>(.*?)<BODY>(.*?)<\/BODY>(.*?)<\/TEXT>

5. match the content between tags using capturing groups for title, dateline and body tags.

6. Create a MongoClientURI to connect to the MongoDB server.

7. Connect to the MongoDB server using the MongoClient.

8. create a database - "ReuterDb" using MongoDatabase.

9. create a collection - "NewsArticles" using MongoCollection.

10. For each Reuters article found in combinedContent, extract the title, date, and body using the regular expression.

11. Create a Document containing the title, date, and body information.

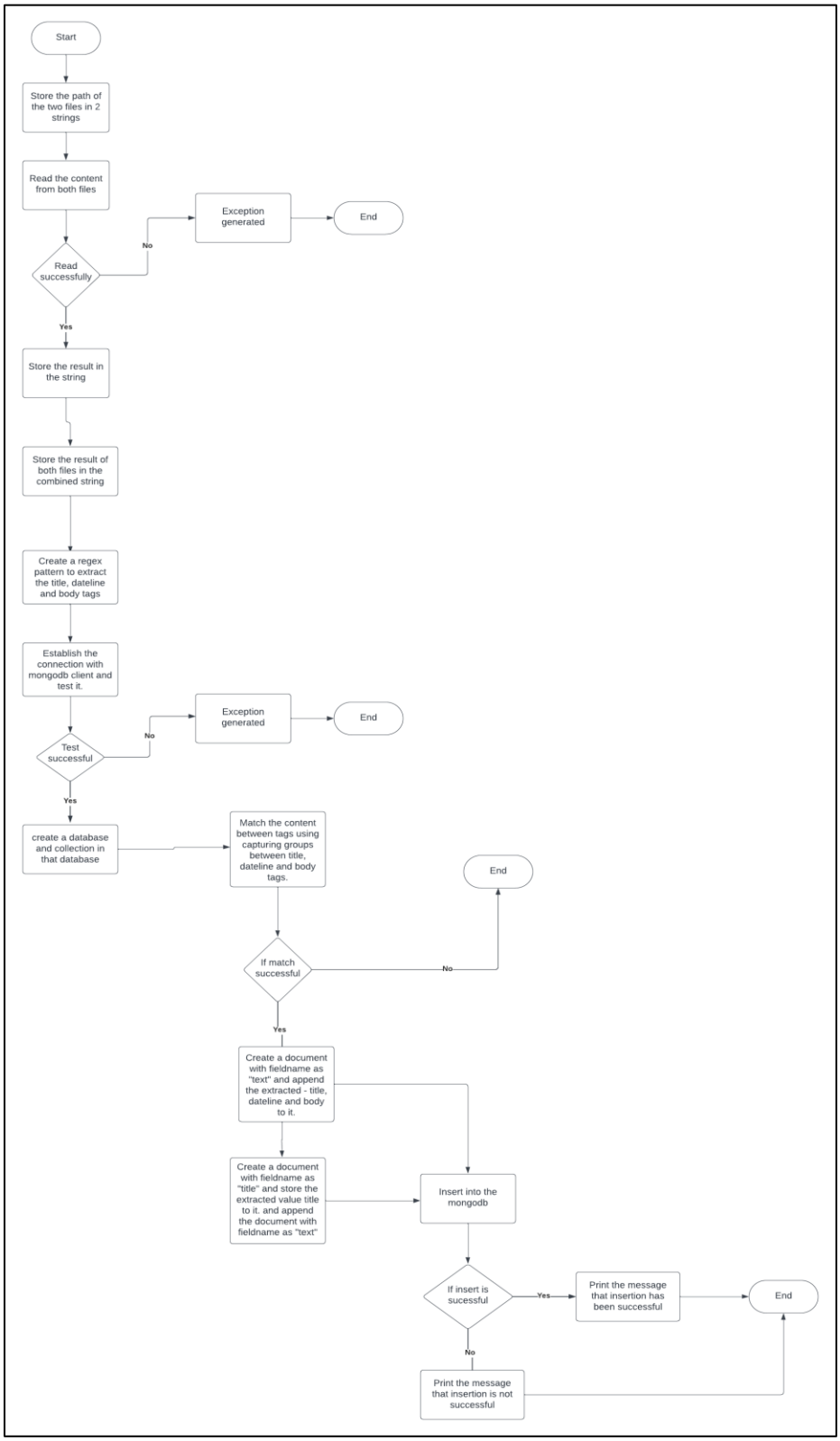
12. Create another Document named textDocument containing the title and the previous Document as the "text" field.

13. Insert the textDocument into the "NewsArticles" collection.

14. Repeat steps 10 to 13 until all articles are processed.

15. Close the connection to the MongoDB server.

Flow-Chart



Flowchart

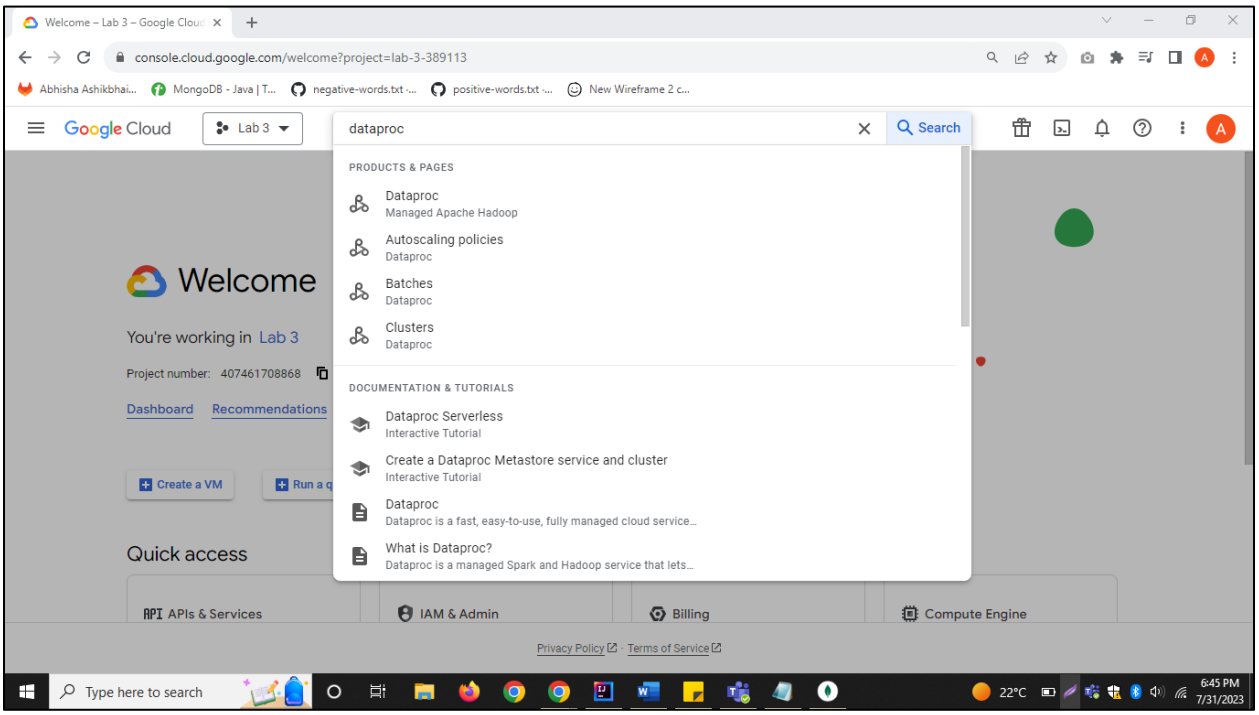
Problem 1 b

1. Using your GCP cloud account, configure and initialize Apache Spark cluster. (Follow the tutorials provided in Lab session).

2. Create a flowchart or write ½ page explanation on how you completed the task, include this part in your PDF file. Note: If for some reason, you fail to work on GCP cloud account (valid reasons required), you need to create local standalone Hadoop/Spark cluster to perform the next set of operations.

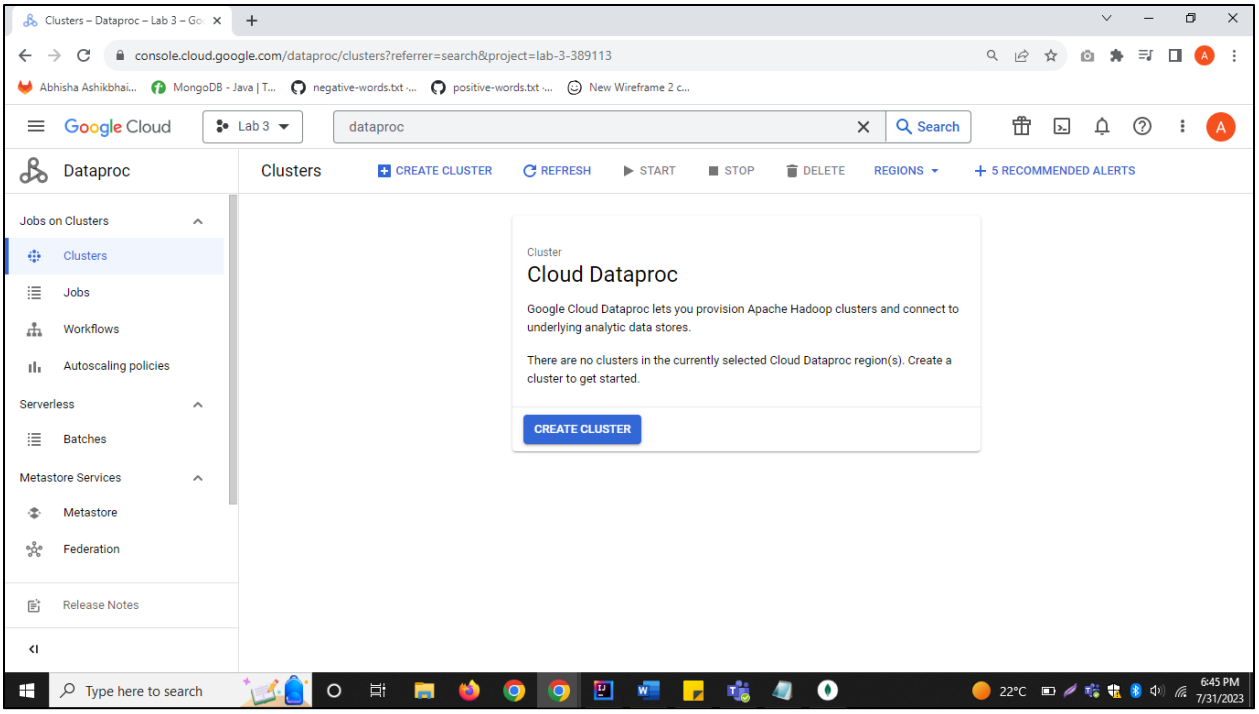
   Screenshots of configuration and initialize Apache Spark cluster

   - First search the word "dataproc" in the searchbar. Then, from the list, click on 'Dataproc'



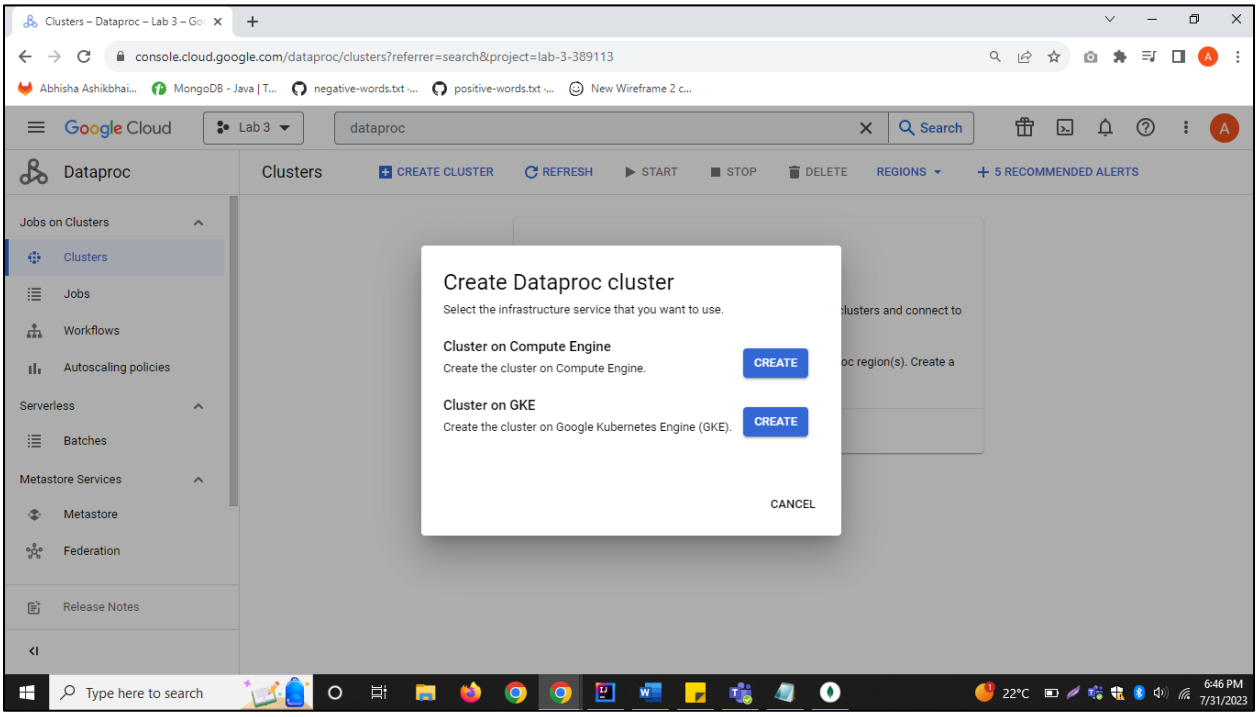
Dataproc_1

   - Then, click on **'Create Cluster'**, and you will see the following screen



Dataproc_2

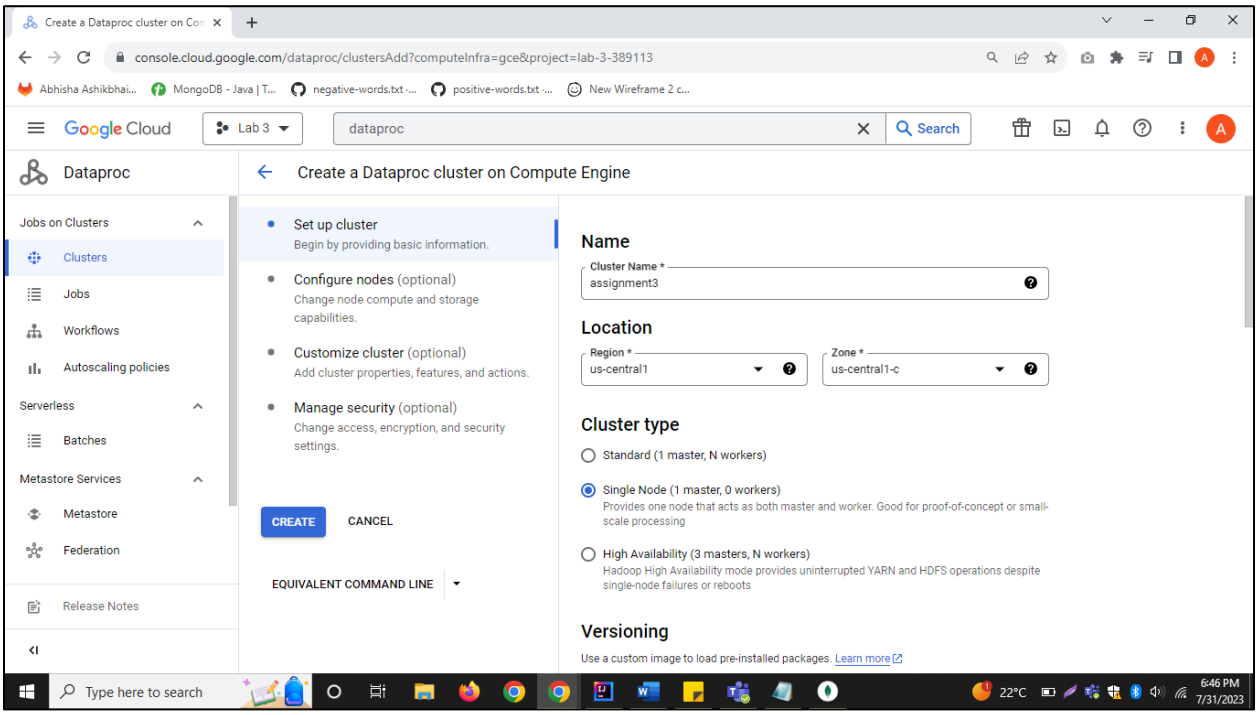
- Then, click on **'Create'** button given besides **Cluster on Compute Engine'.**



Dataproc_3

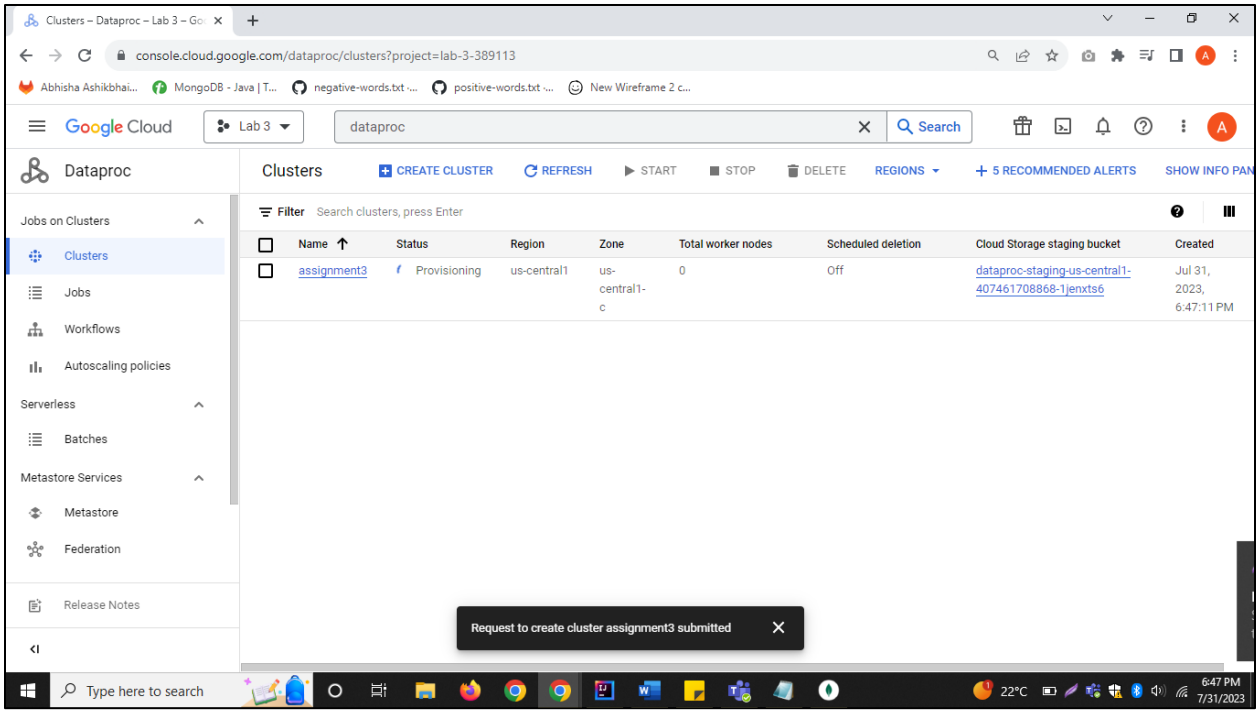
- Enter the name in the clustername as assignment3, and choose cluster type as **'Single Node'.**

  Note: Here, standard is not chosen as the cluster type as the credits are over.



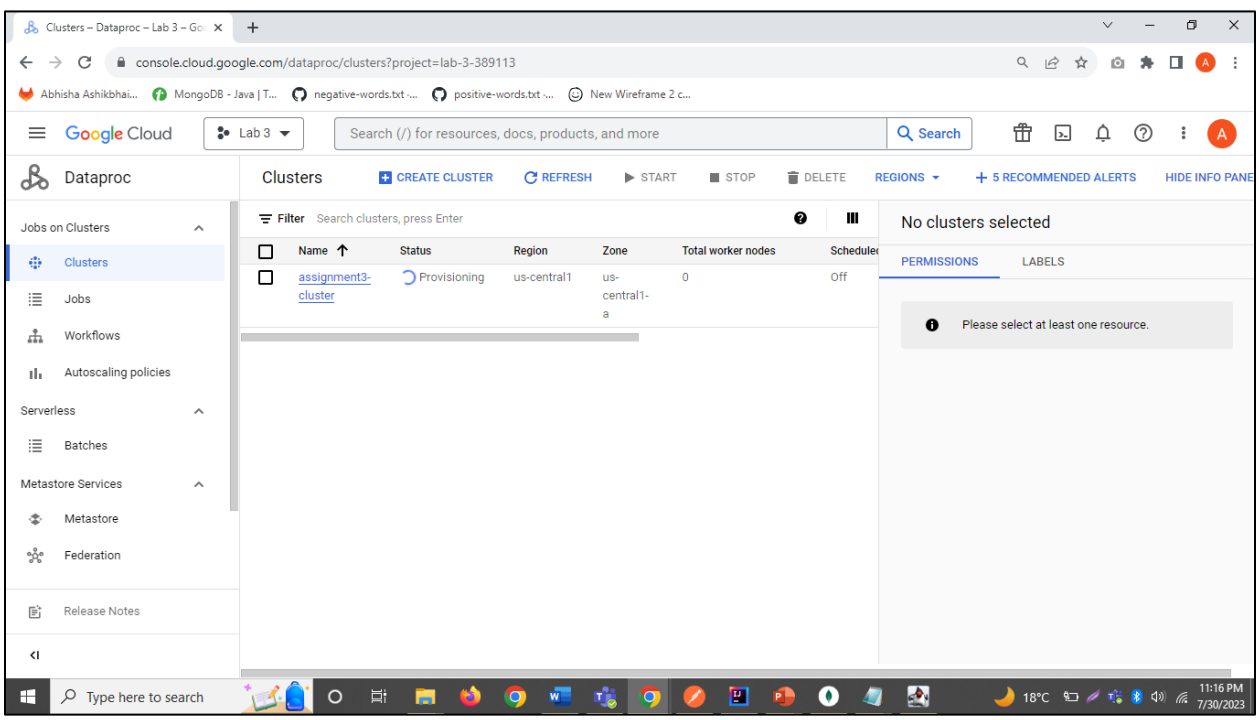
Dataproc_4

- Process of cluster creation will begin. It will take some time to initialize the cluster and the status will be updated from 'Provisioning' to 'Running'



Dataproc_5

- Status has been updated to 'Running'.

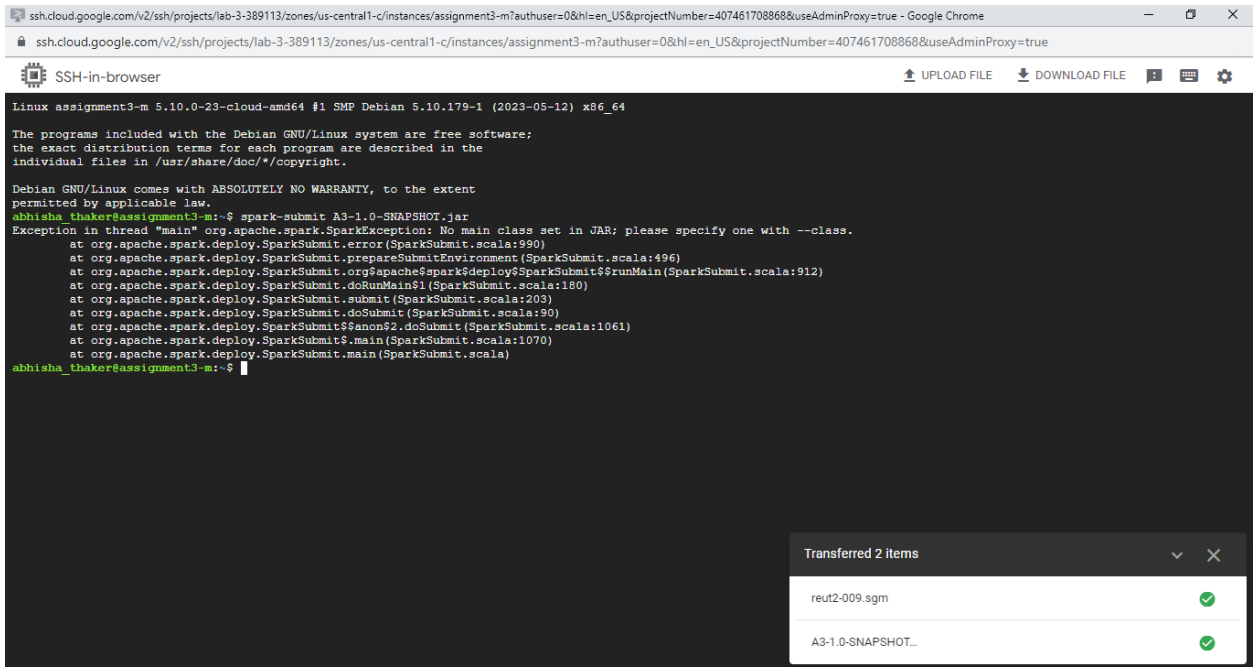


Dataproc_6

3. Write a MapReduce program using Java (WordCounter.java Engine) to count (frequency count) the unique words found in “reut2-009.sgm”.

   Jar file attached in the zip file.

   Output after running in gcp



4. You need to include a flowchart/algorithm of your MapReduce program on the PDF file.

   Algorithm of the MapReduce Program

   1. First the content in the file is separated in words using string tokenizer in mapper method
   2. Then count the words which have occurred once and store it in hashmap as key value pair.
   3. Then, pass it as the input to the reducer method. It will sum up the values for each word and then it displays the final count for each word.

5. In your PDF file, report the words that have highest and lowest frequencies.

   Word with the highest frequencies – 7304 times repeated – [the]
   Word with the lowest frequencies – only 1 time occurrence.

Problem 2: Sentiment Analysis using BOW model on title of Reuters News Articles

List of Positive Words - https://gist.github.com/mkulakowski2/4289437

List of Negative Words - https://gist.github.com/mkulakowski2/4289441

Java Code

```
package problem2;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class problem2 {
    public static void main(String[] args) {
        // Create 4 strings that stores the file location
        String positivePath =
"C:/Users/AVuser/csci5408_s23_b00937694_abhisha_thaker/A3/src/main/resources/
positive-words.txt";
        String negativePath =
"C:/Users/AVuser/csci5408_s23_b00937694_abhisha_thaker/A3/src/main/resources/
negative-words.txt";

        String reut009Title =
"C:/Users/AVuser/csci5408_s23_b00937694_abhisha_thaker/A3/src/main/resources/
reut2-009.sgm";
        String reut014Title =
"C:/Users/AVuser/csci5408_s23_b00937694_abhisha_thaker/A3/src/main/resources/
reut2-014.sgm";

        // Read those 4 files
        String positiveList = readFiles(positivePath);
        String negativeList = readFiles(negativePath);

        String reut009Content = readFiles(reut009Title);
        String reut014Content = readFiles(reut014Title);
        //concatenate the two files
        String combinedContent = reut009Content + reut014Content;

        // extract the title tags
        String regexScript = "<TITLE>(.*?)<\\/TITLE>";

        Pattern regex = Pattern.compile(regexScript, Pattern.DOTALL);
        Matcher matcher = regex.matcher(combinedContent);
```

```
        String titlecontent;
        List<String> titleList = new ArrayList<>();
        while (matcher.find()) {
             titlecontent = matcher.group(1);
             titleList.add(titlecontent);
        }

        String[] wordSplit = titleList.toArray(new String[0]);

        for (String word : wordSplit) {
            System.out.println("wordsplit" + word);
        }

        // splits the files whenever a new line is encountered and stores it
in the string array
        String[] positive = positiveList.split("\n");
        String[] negative = negativeList.split("\n");

        int counterPos = 0;
        int counterNeg = 0;

        String[] wordsBag;

        String[] words;
        int[] frequencies;

        // Initialize the model for the table
        DefaultTableModel model = new DefaultTableModel(
                new Object[]{"News", "Title", "Match", "Polarity"},
                0
        );

        for(String line: wordSplit){
            words = line.toLowerCase().split("\\s+");
            wordsBag = line.toLowerCase().split("\\s+");

            frequencies = new int[words.length];

            counterPos = 0;
            counterNeg = 0;

            for (String word : wordsBag) {
                for (String pos : positive) {
                    if (word.equals(pos)) {
                        counterPos++;
                        // System.out.println(word);
                    }
                }
                for (String neg : negative) {
                    if (word.equals(neg)) {
                        counterNeg++;
                        // System.out.println(word);
                    }
                }
            }
            for (int i = 0; i < words.length; i++) {
```

```
                    int counter = 1;
                    for (int j = i + 1; j < words.length; j++) {
                        if (words[i].equalsIgnoreCase(words[j])) {
                            counter++;
                            words[j] = "";
                        }
                    }
                    frequencies[i] = counter;
                    //  System.out.println("match
found"+wordSplit[i]+counter);
            }
            List<String> matches = new ArrayList<>();
            List<Integer> frequency = new ArrayList<>();
            Map<String, Integer> wordFrequencyMap = new HashMap<>();
            Map<String,Integer> finalFrequencyMap = new HashMap<>();
            for (int i = 0; i < words.length; i++) {
                if (!words[i].isEmpty() && frequencies[i] > 0) {
                    matches.add(words[i]);
                    frequency.add(frequencies[i]);
                    wordFrequencyMap.put(words[i],frequencies[i]);
                    if(frequencies[i] > 1){
                        finalFrequencyMap.put(words[i],frequencies[i]);
                    }
                }
            }
            matchFrequency(line,wordFrequencyMap);
            String polarity = displayTag(line,counterPos,counterNeg);
            System.out.println("Polarity is"+polarity);
            model.addRow(new Object[]{
                    model.getRowCount() + 1, // News
                    line, // Title
                    finalFrequencyMap.toString(), // Match
                    polarity // Polarity
            });
        }
        JTable table = new JTable(model);

        // Show the JTable in a JScrollPane
        JScrollPane scrollPane = new JScrollPane(table);

        // Create a JFrame to hold the JScrollPane
        JFrame frame = new JFrame("Table");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.add(scrollPane);
        frame.pack();
        frame.setVisible(true);
    }

    private static void matchFrequency(String line, Map<String, Integer>
wordFrequencyMap) {
        System.out.println("For line 1"+line);
        System.out.print("bow1 = {");

        for (Map.Entry<String, Integer> entry : wordFrequencyMap.entrySet())
{
            String word = entry.getKey();
            int freq = entry.getValue();
```

```
            System.out.print("\"" + word+ ""+ "\":" + ""+ freq +"");
            System.out.print(" ");
        }
        System.out.println("}");
    }
    static String readFiles(String filePath)  {
        BufferedReader readFile = null;
        try {
            readFile = new BufferedReader(new FileReader(filePath));
            StringBuilder content = new StringBuilder();
            String line;

            while ((line = readFile.readLine()) != null) {
                content.append(line).append("\n");
            }

            String sgmContent = content.toString();
            return sgmContent;
        } catch(FileNotFoundException e){
            e.printStackTrace();
        } catch(IOException e){
            e.printStackTrace();
        } finally{
            try {
                readFile.close();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
        return null;
    }

    static String displayTag(String line, int npos, int nneg){
     //   System.out.println("Line is"+line);
      //   System.out.println("No of positive words from the line is"+npos);
        // System.out.println("No of negative words from the line is"+nneg);
        int overallScore = npos - nneg;
      //  System.out.println(overallScore);
        String polarity = null;
        if(overallScore > 0){
            polarity = "positive";
        } else if(overallScore < 0) {
            polarity = "negative";
        }else if(overallScore == 0) {
            polarity = "neutral";
        }
        return polarity;
    }
}
```

Summary

1. First 2 file paths stores positive and negative word lists, and next two file path stores two news content files named reut2-009.sgm and reut2-014.sgm.

2. Then a method readFiles() is created to read the contents of the specified files and this method returns the content as a string.

3. Then it reads the content of reut2-009.sgm and reut2-014.sgm, extracts the text between <TITLE> and </TITLE> tags, and stores the titles in a list - titleList.

4. Then each title's case is changed to lowercase and split() function is called extract the words by space.

It counts the matching words and displays the frequency of each word.

5. For each title, the code compares the words with the positive and negative word lists and counts the number of positive and negative words.

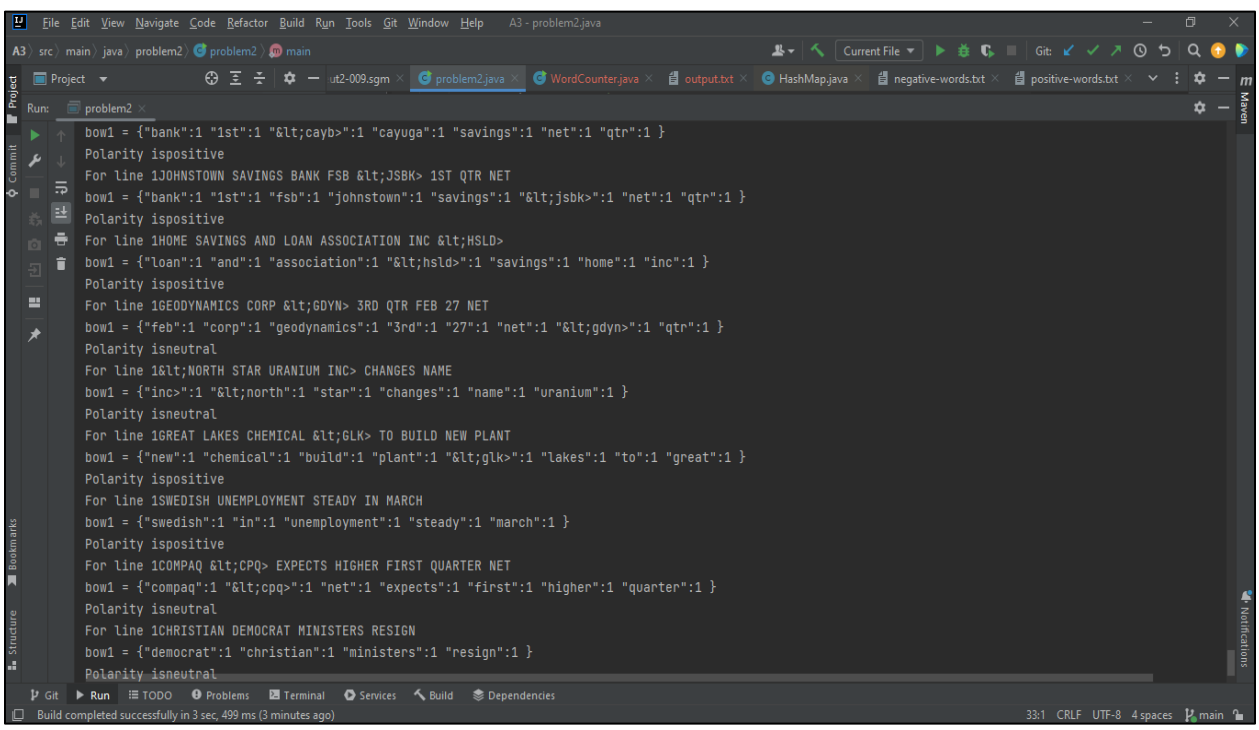
Based on the counts, it determines the polarity of the sentiment (positive, negative, or neutral).

(overall score = positive - negative)- if overall score > 0 - positive, overall score < 0 - negative and if overall score = 0, neutral.

6. Then a table is created to display the results. Each row in the table represents a title along with its polarity and a list of matching words with their frequencies.

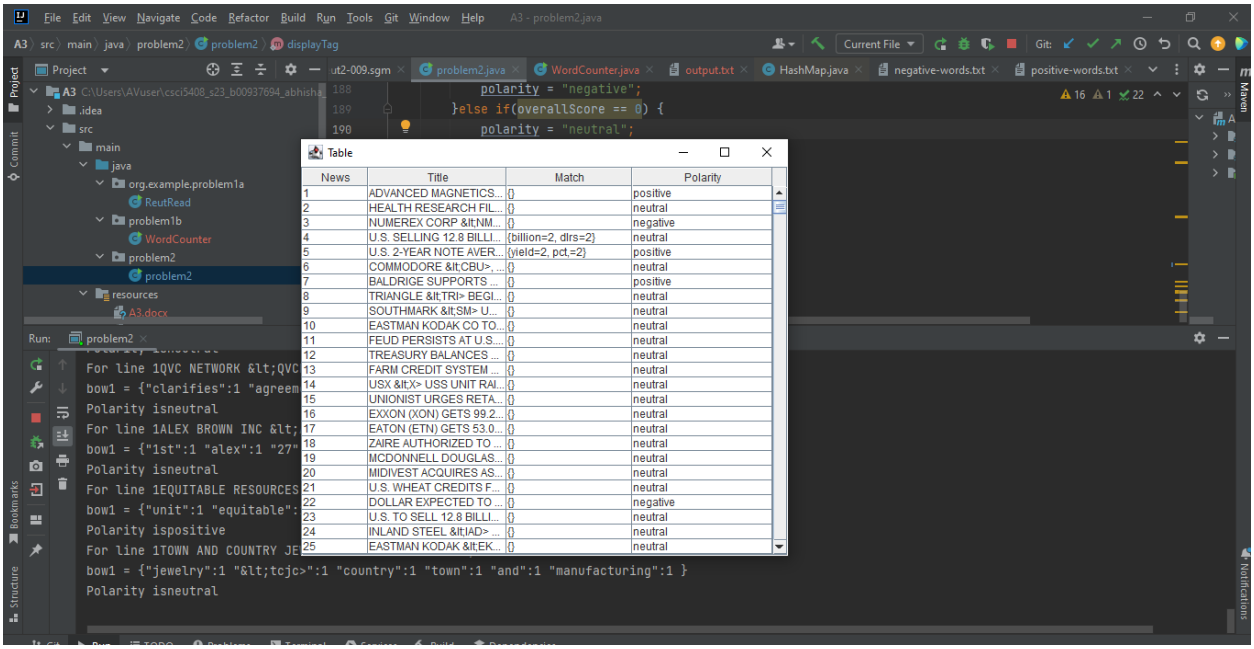
7. The results are displayed using a Swing-based graphical user interface (JTable) in a JFrame with a JScrollPane.

Output



problem2_output

Output Problem 2



Problem2_with_Jframe

All Problems References

[1] F. Dib, "Regex101: Build, test, and debug regex," regex101. [Online]. Available: https://regex101.com/. [Accessed: 12-Jul-2023].

[2] Shaw, D. (2017, July 15). How to find distinct word using MapReduce. BIG DATA PROGRAMMERS. https://bigdataprogrammers.com/get-distinct-words-file-using-map-reduce/

[3] MongoDB Compass download (GUI). (n.d.). MongoDB. Retrieved July 31, 2023, from https://www.mongodb.com/try/download/compass

[4] How to use tables. (n.d.). Oracle.com. Retrieved July 31, 2023, from https://docs.oracle.com/javase/tutorial/uiswing/components/table.html

[5] HashMap (java platform SE 8). (2023, June 14). Oracle.com. https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html

[6] Beugnet, M. (2022, February 1). Getting started with MongoDB and Java - CRUD operations tutorial. Mongodb.com. https://www.mongodb.com/developer/languages/java/java-setup-crud-operations/

[7] “BufferedWriter (java platform SE 8),” Oracle.com, 05-Apr-2023. [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/io/BufferedWriter.html. [Accessed: 13-Jul2023].

[8] HDFS Architecture. (n.d.). Apache.org. Retrieved July 31, 2023, from https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

[9] Data, U. B. [@UnboxingBigData]. (2018, November 29). Understanding word count program with map reduce (with demonstration). Youtube. https://www.youtube.com/watch?v=p-rzyWW7zjw

[10] Reducer (Apache Hadoop main 2.7.0 API). (2015, April 10). Apache.org. https://hadoop.apache.org/docs/r2.7.0/api/org/apache/hadoop/mapreduce/Reducer.html