CSCI 5408
DATA MANAGEMENT AND WAREHOUSING

ASSIGNMENT - 2

Banner ID: B00955671

GitLab Assignment Link:
https://git.cs.dal.ca/ssavaliya/csci5408_f23_b00955671_sumit_savaliya/-/tree/main/A2

Table of Contents

Problem 1A: Data Cleaning / Push to MongoDB Cluster

In this part, I have successfully cleaned the data and pushed it to MongoDB[1]. I have used Regex Expression to remove unwanted special symbols/characters.

Code Link (Package: problem1A)

FlowChart:

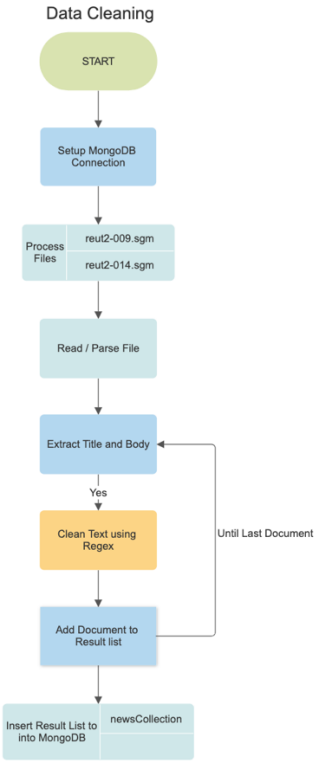

Figure 1: Flowchart of Data Cleaning Algorithm 1A.

Algorithm:

1. **Read and Parse XML File:**
   - The function starts by taking the **fileName** and **MongoCollection<Document>** as parameters.
   - It creates a **File** object using the provided file name and gets its absolute path.
   - It uses the Jsoup library to parse the XML file using the XML parser. The result is stored in a **org.jsoup.nodes.Document** object named **document**.
2. **Initialize List for MongoDB Documents:**
   - It initializes an empty **List<Document>** named **newsDocuments** to store the MongoDB documents that will be created from the XML file.
3. **Iterate Over "REUTERS" Elements:**
   - The function iterates over each "REUTERS" element in the parsed XML document.
4. **Extract Title and Body:**
   - For each "REUTERS" element, it extracts the text content of the "TEXT > TITLE" and "TEXT > BODY" elements.
   - The extracted text is passed through the **cleanText** method, which removes non-alphanumeric characters.

Steps Followed.

1. Setup a MongoDB Atlas Cluster.
    - Copy the connection URL / password to paste into the code.

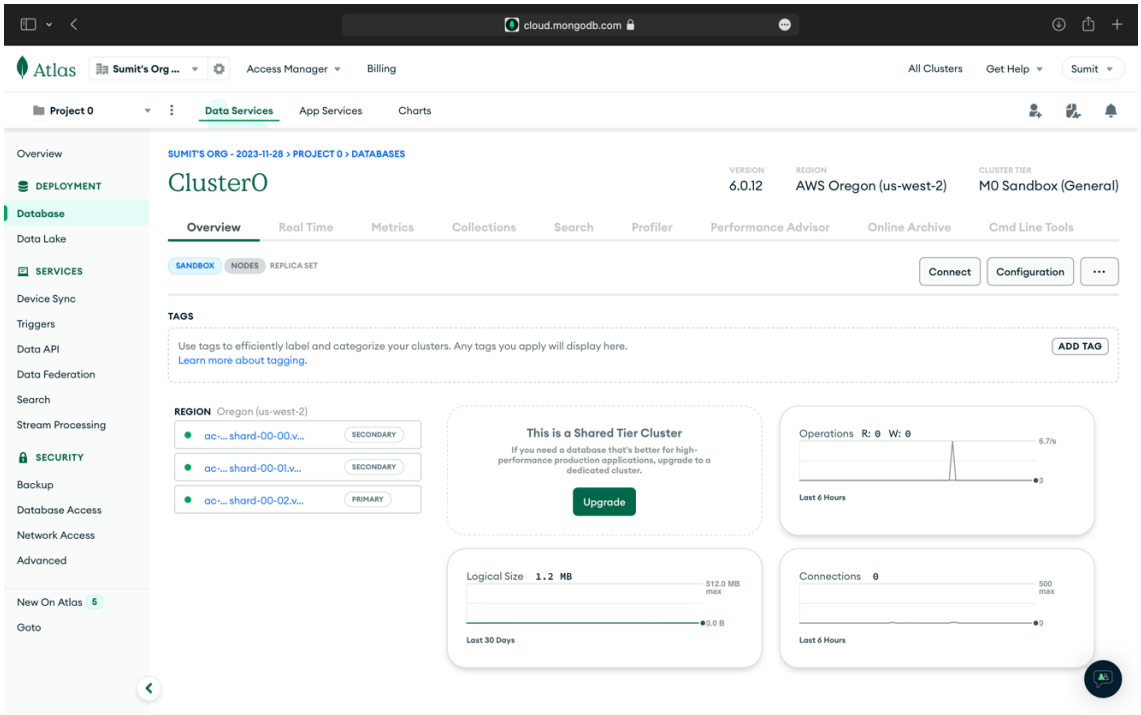

Figure 2: MongoDB Atlas Cluster

2. Initially the collections should be empty.

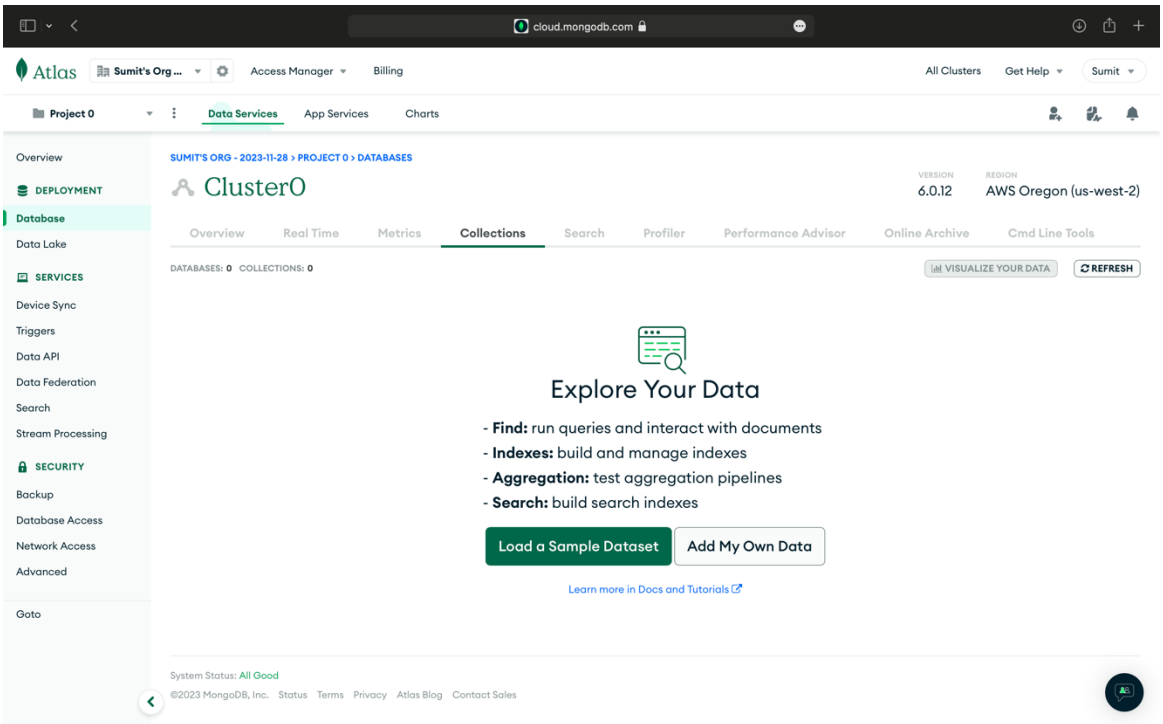

Figure 3: MongoDB Atlas Cluster - Collections

3. Now, Execute the code for Problem 1A and you should see the collection populate with the required documents as per the flowchart (see Fig. 1).
    - Make sure to replace the connectionString with your Atlas / Local MongoDB connection string.

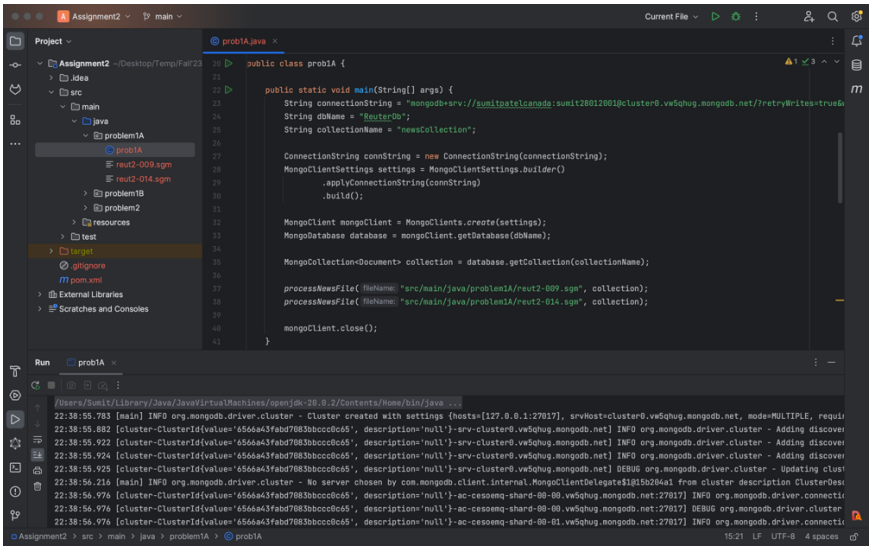

Figure 4: Code Execution for Problem 1A

4. Result: MongoDB Atlas now has a collection names “newsCollection” which contains all the cleaned titles and body.

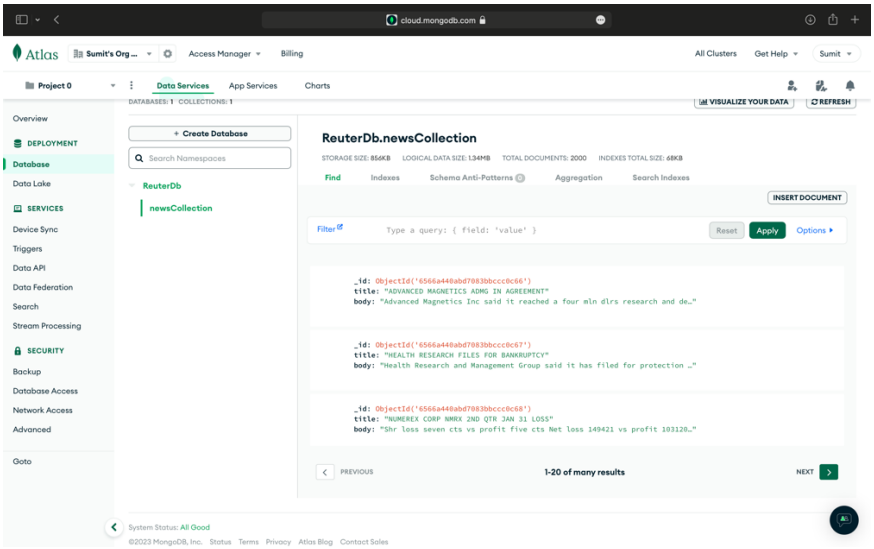

Figure 5: Cleaned News Data inserted in MongoDB

5. This completes Problem 1A and we have clean Titles and Body of every article from the reut-009.sgm and reut-014.sgm files.

I will be using this cleaned up data while solving Problem 2 as it requires titles of all the news articles. So I have decided to fetch them from this newsCollection itself and avoid redundant tasks.

Problem 1B: News Data Processing using Spark

Steps followed: UniqueWordCount.py file can be found in the repository inside folder Problem1B.

Setup Up the Environment:

- We began by setting up the environment on Google Cloud Platform[2] Apache Spark[3] using DataProc on GCP and Also setup a cloud Storage bucket named **assignment2-prob-1b**.

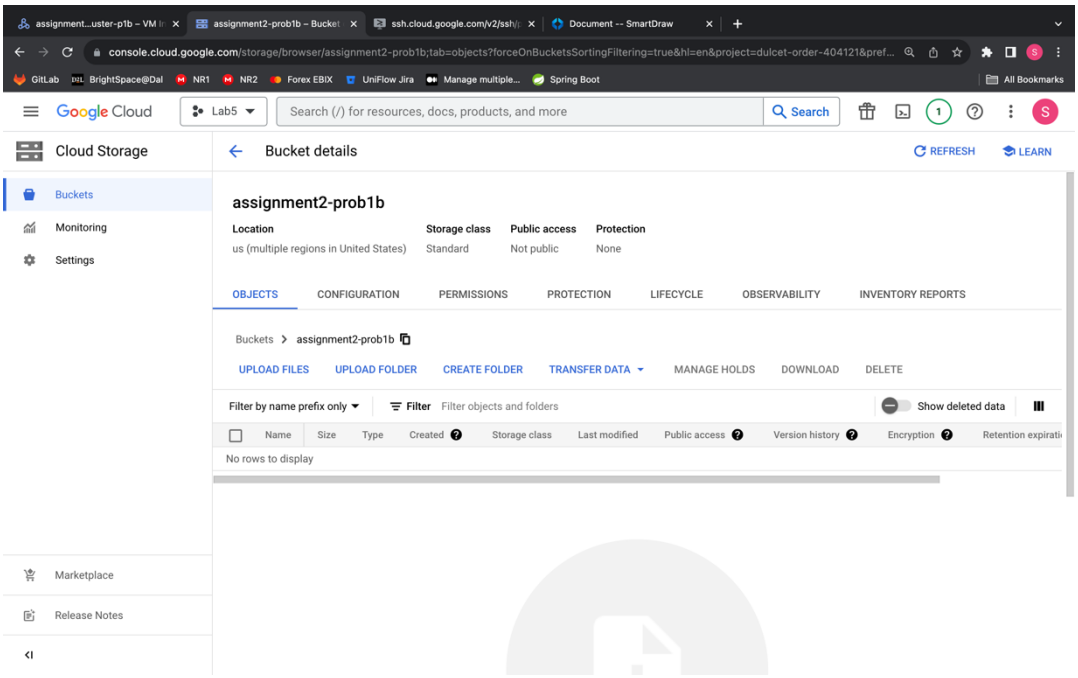

Figure 6: Setup GCS - Google Cloud Storage Bucket

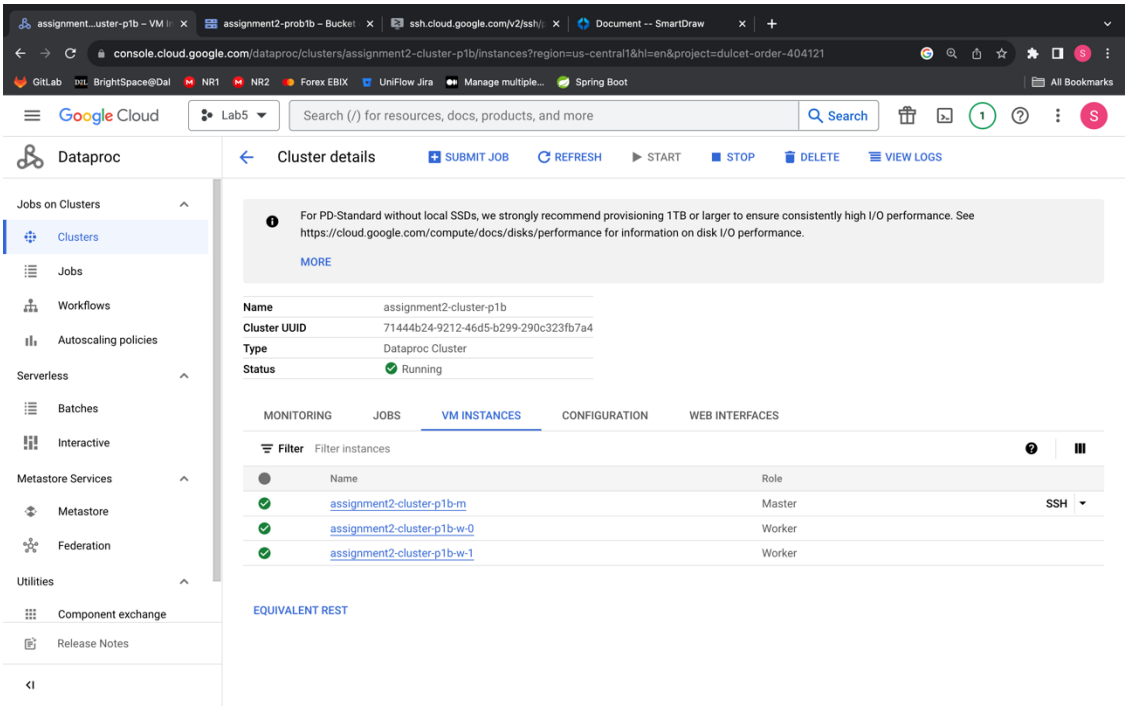

Figure 7: Setup Dataproc Apache Spark Cluster

Initializing Spark Session:

- The Spark session was initialized using PySpark's **SparkSession** API, providing a name for the application.

```
spark = SparkSession.builder.appName("WordCount").getOrCreate()
```

Reading Input Data:

- We read the input data from the "reut2-009.sgm" file using Spark's **read.text** method. The resulting RDD (Resilient Distributed Dataset) was processed to extract lines of text.

```
lines = spark.read.text(input_file).rdd.map(lambda r: r[0])
```

Tokenization and FlatMap:

- We tokenized the lines of text and used **flatMap** to create a flat list of words.

```
words = lines.flatMap(lambda line: line.split(" "))
```

Mapping to Key-Value Pairs:

- Each word was then mapped to a key-value pair, with the word as the key and an initial count of 1.

```
word_key_values = words.map(lambda word: (word, 1))
```

Reducing by Key for Count:

- We applied the **reduceByKey** operation to group the key-value pairs by the word and sum up the counts, effectively obtaining the frequency of each unique word.

```
word_counts = word_key_values.reduceByKey(lambda a, b: a + b)
```

Collecting and Outputting Result:

- The result was collected into a local variable for further processing and output.

```
result = word_counts.collect()
```

Saving Local Result to File:

- The local result was saved to a text file using the **saveAsTextFile** method. The specified local file path was used to store the result.

```
result.saveAsTextFile("file://" + output_file)
```

Uploading Result to GCS:

- To share and store the result centrally, we used the **write.text** method to upload the result to a specified GCS bucket and path.

```
• result.write.text("gs://{}/{}".format(gcs_bucket, gcs_output_path))
```

Stopping Spark Session:

- Finally, we stopped the Spark session to release resources.

```
spark.stop()
```

Transfer “reut2-014.sgm” file to HDFS storage using

hdfs dfs -put ./reut2-009-sgm /user/sumitmsavaliya

Connected to cluster through SSH and Submitted the spark job.

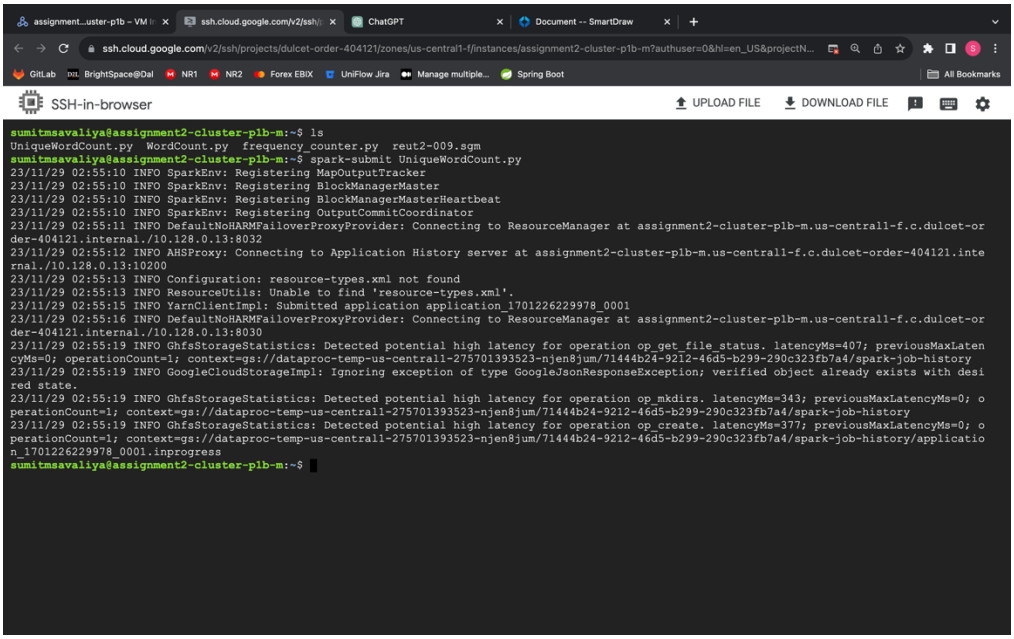

Figure 8: Spark Job Completed and Saved Result to GCS

Once the job is complete, We get the results.txt file in our GCS Bucket.

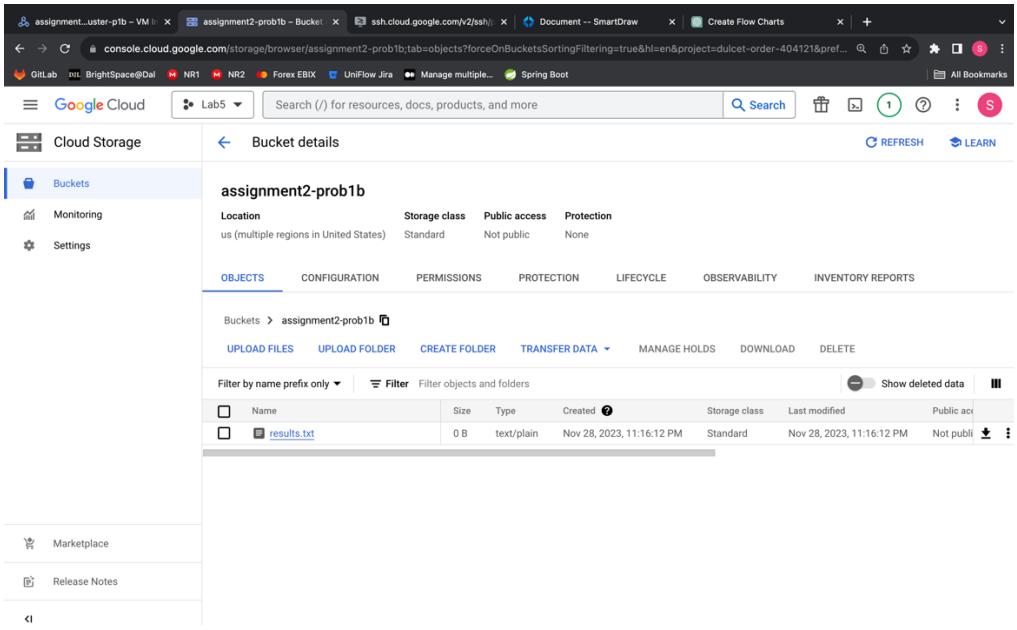

Figure 9: Results.txt file in GCS Bucket

This Completes Problem 1B with Counting of frequency of Unique words in the Reut2-009.sgm file.

Problem 2: Sentiment Analysis using Bow Model

Successfully implemented the task using files of Positive and Negative words from the same author.

This algorithm performs sentiment analysis on news titles stored in a MongoDB collection. The sentiment analysis involves scoring each news title based on the presence of positive and negative words. The results are then inserted into a MySQL table for further analysis. Below is a detailed breakdown of the algorithm:

1. **Connect to MongoDB:**
    - Established a connection to MongoDB using the provided connection string, database name, and collection name.
    - Fetched the desired collection (**newsCollection**) from the MongoDB database.

2. **Retrieve News Titles:**
    - Query the MongoDB collection to retrieve the news titles.
    - Stored the news titles in a list (**newsTitles**).

3. **Read Positive and Negative Words:**
    - Read positive and negative words from files (**positive-words.txt** and **negative-words.txt**).
    - Stored positive and negative words in separate lists (**positiveWords** and **negativeWords**).

4. **Sentiment Analysis Loop:**
    - Iterated over each news title in the **newsTitles** list.

5. **Build Bag of Words (BoW):**
    - built a Bag of Words (BoW) for each news title.
    - Used a map (**bow**) to store word frequencies in the news title.

6. **Score Calculation:**
    - Compared each word in the BoW to positive and negative words.
    - Calculated a sentiment score based on the frequency of positive and negative words.

7. **Determine Polarity:**
    - Determined the polarity of the news title based on the calculated score.
    - If the score is negative, the polarity is set to "Negative".
    - If the score is positive, the polarity is set to "Positive".
    - If the score is zero, the polarity is set to "Neutral".

8. **Store Results:**
    - Stored the news title, matched words, sentiment score, and polarity in a map (**result**).
    - Added each result map to a list (**results**).

9. **Insert into MySQL Table:**
    - Established a connection to a MySQL database using the JDBC URL, username, and password.
    - Inserted each result into a table named **NewsAnalysis** with columns: **NewsTitle**, **MatchedWords**, **Score**, and **Polarity**.
    - Used a prepared statement to efficiently insert multiple records.

10. **Close Connections:**
    - Closed the MongoDB connection after processing all news titles.
    - Closede the MySQL connection after inserting all results into the table.

Functional Testing

1. Please ensure to create the MySQL [4] Database and Table for storing the database using the following commands.(CODE LINK Package problem2)

   CREATE DATABASE NewsDb;

   USE NewsDb;

   CREATE TABLE NewsAnalysis (
   Id INT PRIMARY KEY AUTO_INCREMENT,
   NewsTitle VARCHAR(255),
   MatchedWords VARCHAR(1000),
   Score INT,
   Polarity VARCHAR(10)
   );

2. Ensure that ConnectionString of Mongodb is properly provided and it contains the filtered documents that were created in Problem 1A (Execute code for Problem 1A if data isn't there as it deals with cleaning up the titles for this problem)

3. Make sure that MySQL Connection URL and username password are provided correctly.

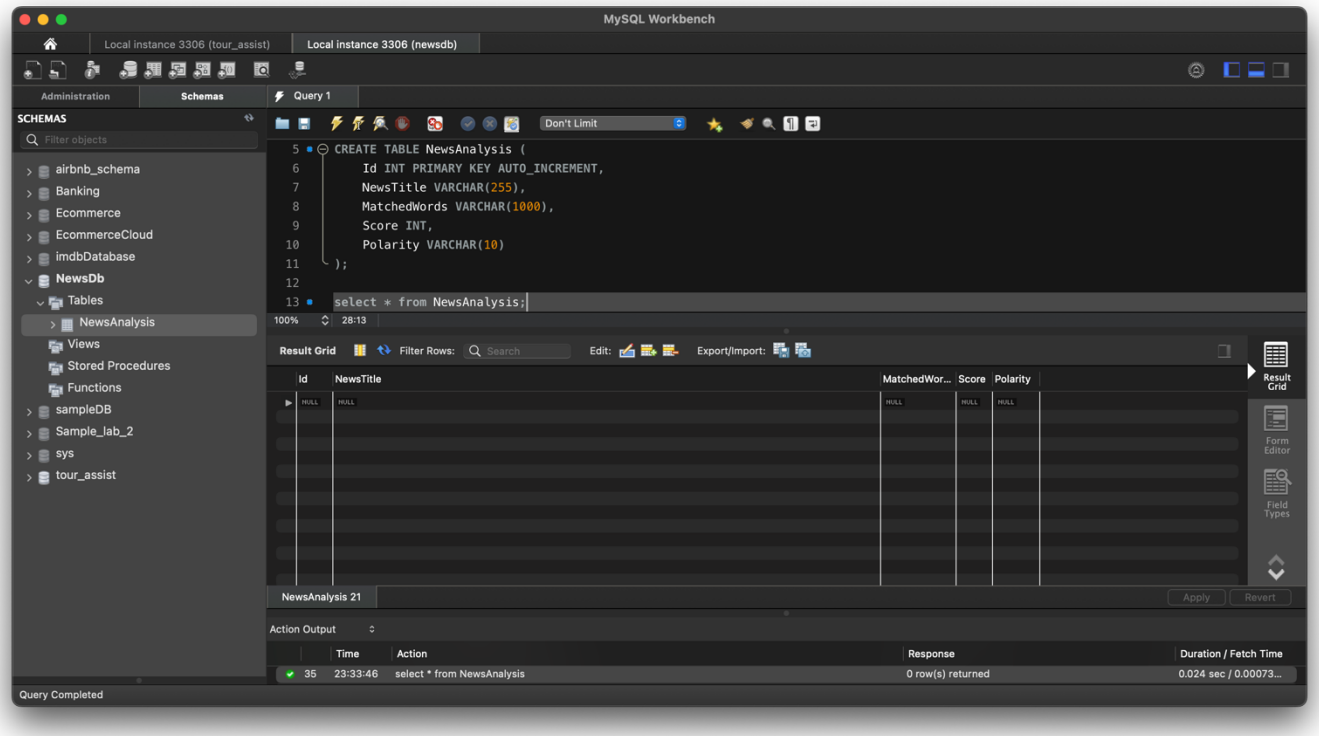

Figure 10: NewsAnaylsis Table before code execution

4. Execute the Code for Problem 2

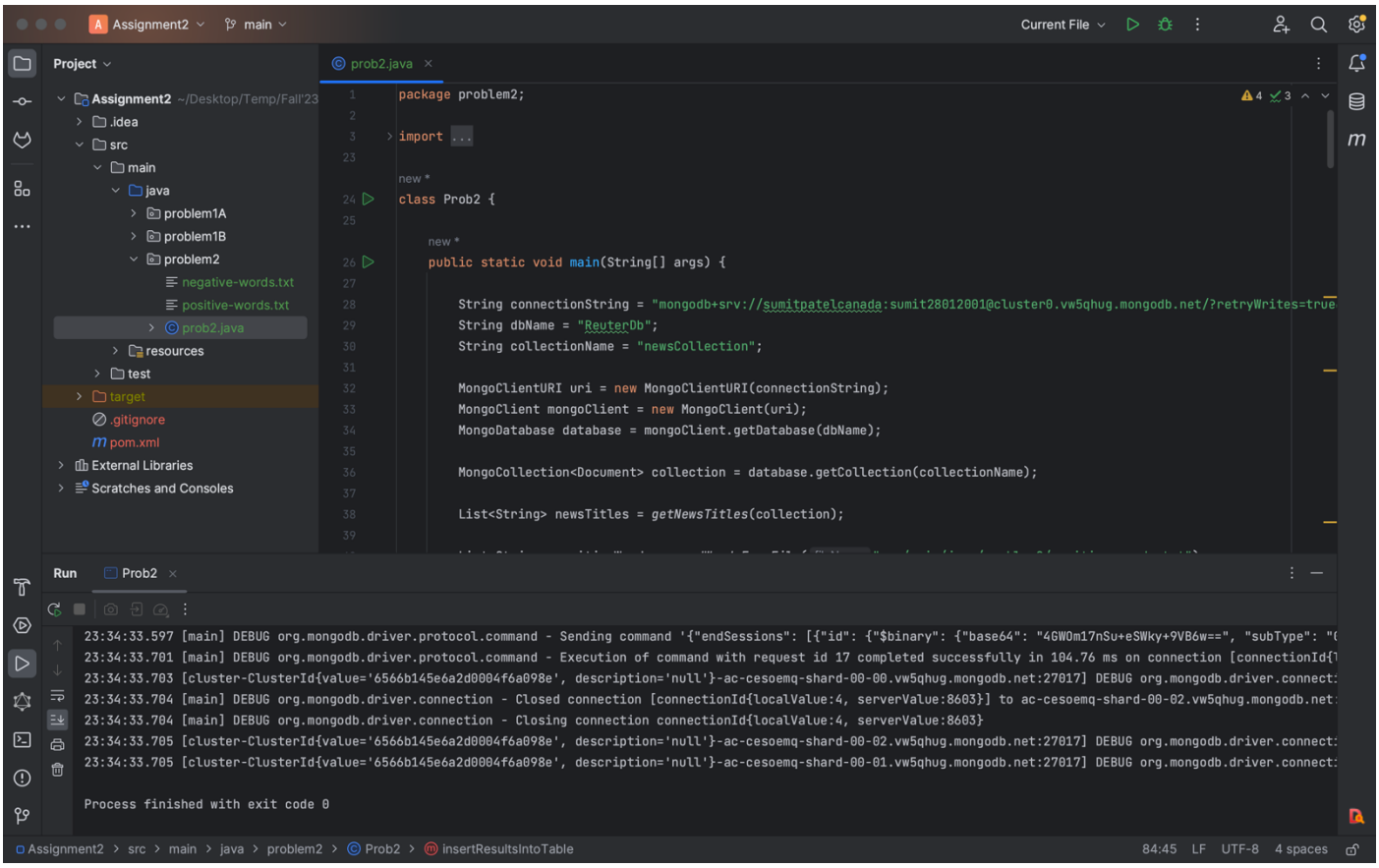

Figure 11: Executing Problem2

5. Check and verify the score and polarity back in MySQL NewsAnalysis Table. The required data should be populated now.

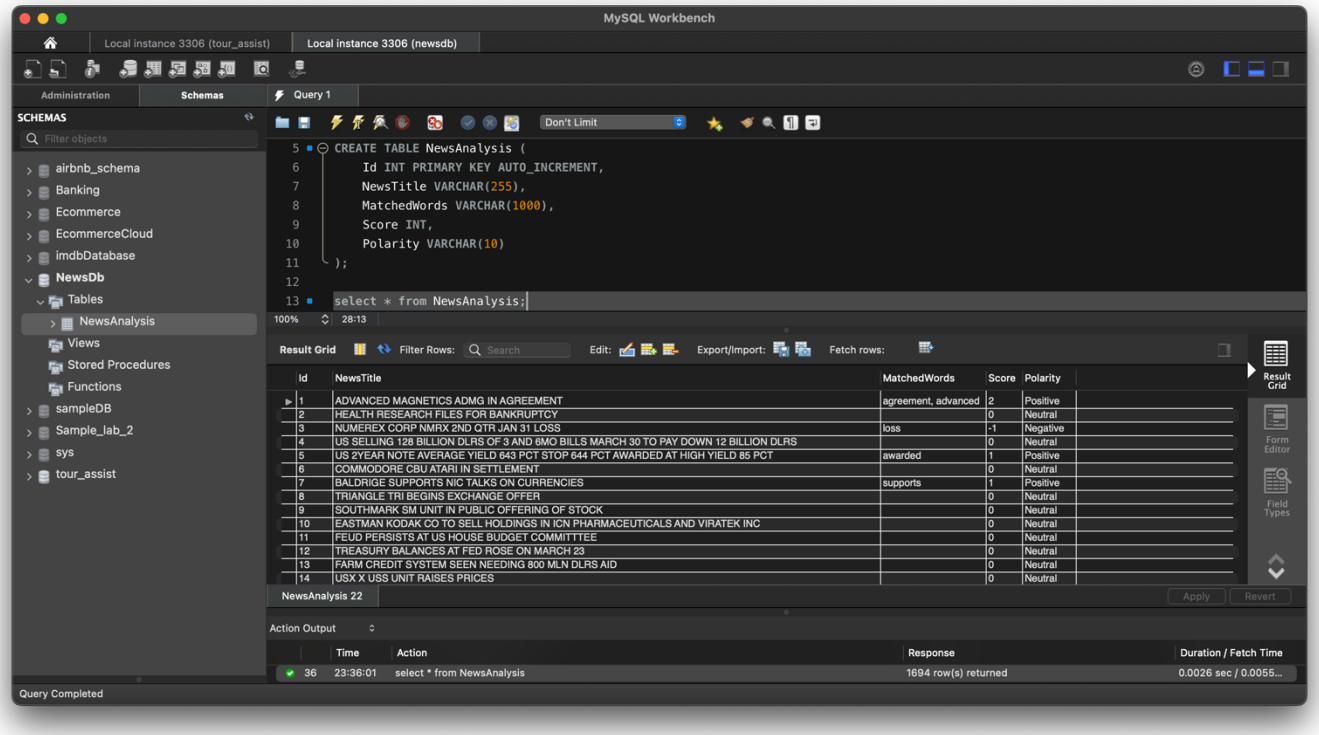

Figure 12: News Analysis Data with score and polarity

References:

[1] MongoDB Atlas, "MongoDB Atlas. Fully managed MongoDB in the cloud.". Available at: https://www.mongodb.com/cloud/atlas/lp/try4?utm_content=controlhterms&utm_source=google&utm_campaign=search_gs_pl_evergreen_atlas_core-high-int_prosp-brand_gic-null_amers-ca_ps-all_desktop_eng_lead&utm_term=mongodb&utm_medium=cpc_paid_search&utm_ad=e&utm_ad_campaign_id=19616985274&adgroup=146373896140&cq_cmp=19616985274&gad_source=1&gclid=CjwKCAiAvJarBhA1EiwAGgZl0GdOJFBA4-u9A8UJ2Lr-ds431cymrh_NeRWq2EIWrICJxdQ6XbL9hhoCOeMQAvD_BwE (Accessed: 27 November 2023).

[2] Google cloud platform. Available at: https://console.cloud.google.com/sql/instances?hl=en&project=sample-lab-data (Accessed: 27 November 2023).

[3] Spark, "PySpark Overview", Available at: https://spark.apache.org/docs/latest/api/python/index.html (Accessed: 27 November 2023).

[4] "*MySQL Workbench download now", MySQL*. Available at: https://www.mysql.com/products/workbench/ (Accessed: 28 November 2023).

[5] *IntelliJ IDEA – the leading Java and Kotlin Ide* (no date) *JetBrains*. Available at: https://www.mysql.com/products/workbench/ (Accessed: 27 November 2023).