CSCI-5408

DATA MANAGEMENT, WAREHOUSING, & ANALYTICS

ASSIGNMENT – 2

Banner ID: B00977669

GitLab Assignment Link:
https://git.cs.dal.ca/saji/csci5408_w24_b00977669_christin_saji

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

Flowchart:

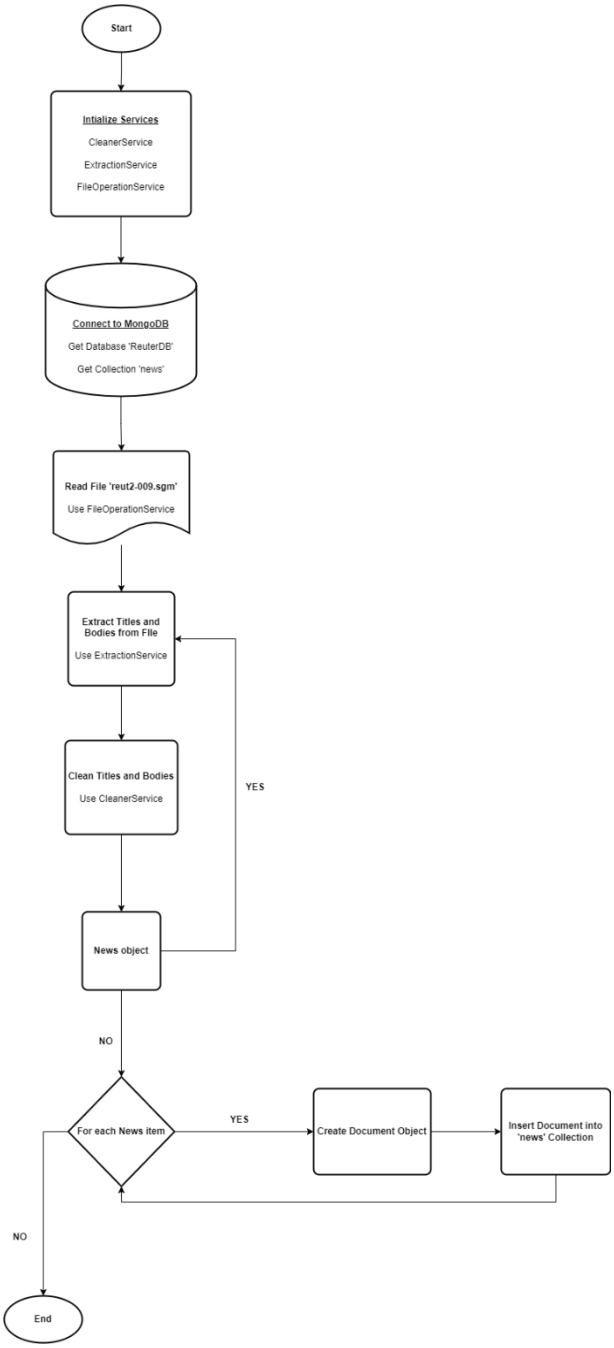


Figure 1 Flowchart of Reuters Data extracting, cleaning, and transformation.

Algorithm:

Step 1: Initialize ‘CleanerService’ for cleaning text data, ‘ExtractionService’ for use during data extraction, and ‘FileOperationService’ for handling file operations.

Step 2: Establish connection to MongoDB [1]. Access the ‘ReuterDb’ database within MongoDB and retrieve the ‘news’ collection from the database to store the news items.

Step 3: Read the data file using ‘FileOperationService’ to read “reut2-009.sgm” file’s content into memory.

Step 4: Extract, clean news items and create a ‘News’ object using ‘ExtractionService’, and ‘CleanerService’. Loop over till the end of the file.

Step 5: For each ‘News’ object, iterate over the list and create ‘Document’ objects for MongoDB with the title and body data.

Step 6: Insert ‘Documents’ object into the ‘news’ collection in the MongoDB database.

Functional Testing:

Created a ‘ReuterDb’ database and ‘news’ collection within the MongoDB. Currently, it has no documents inside the database.

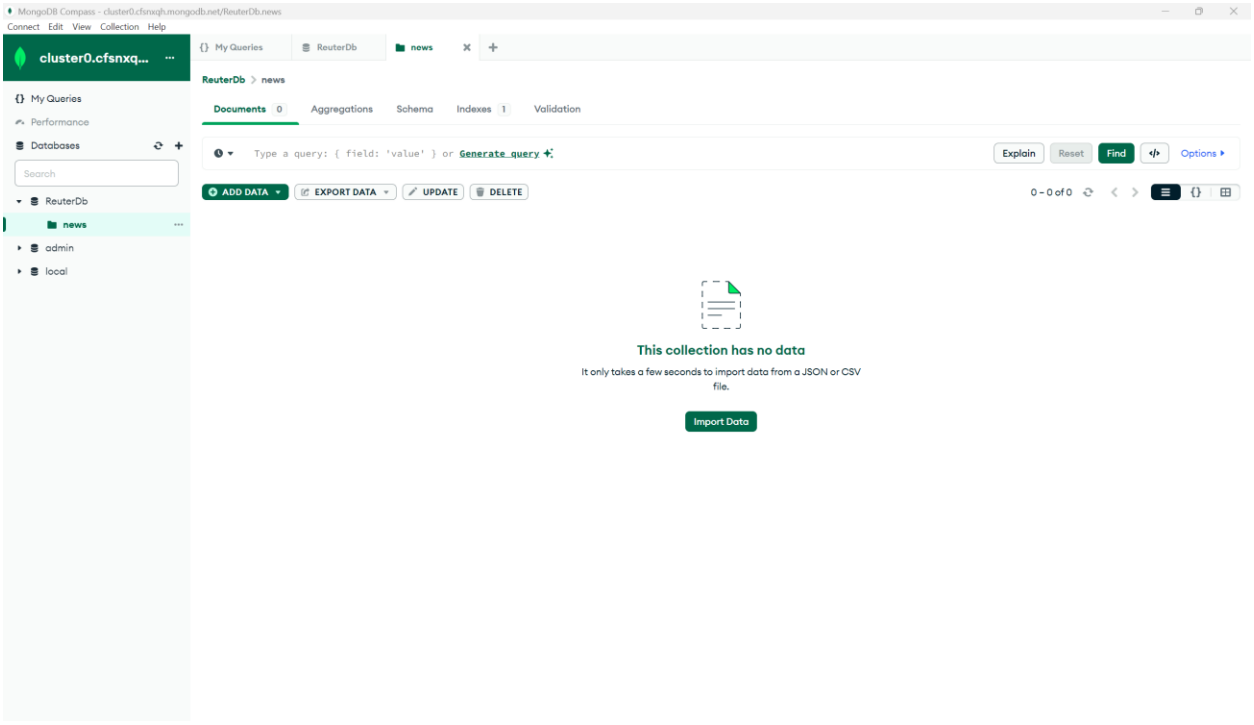


Figure 2 ReuterDb database on MongoDB

Extracted the raw data from the file to a ‘newsList’ variable.

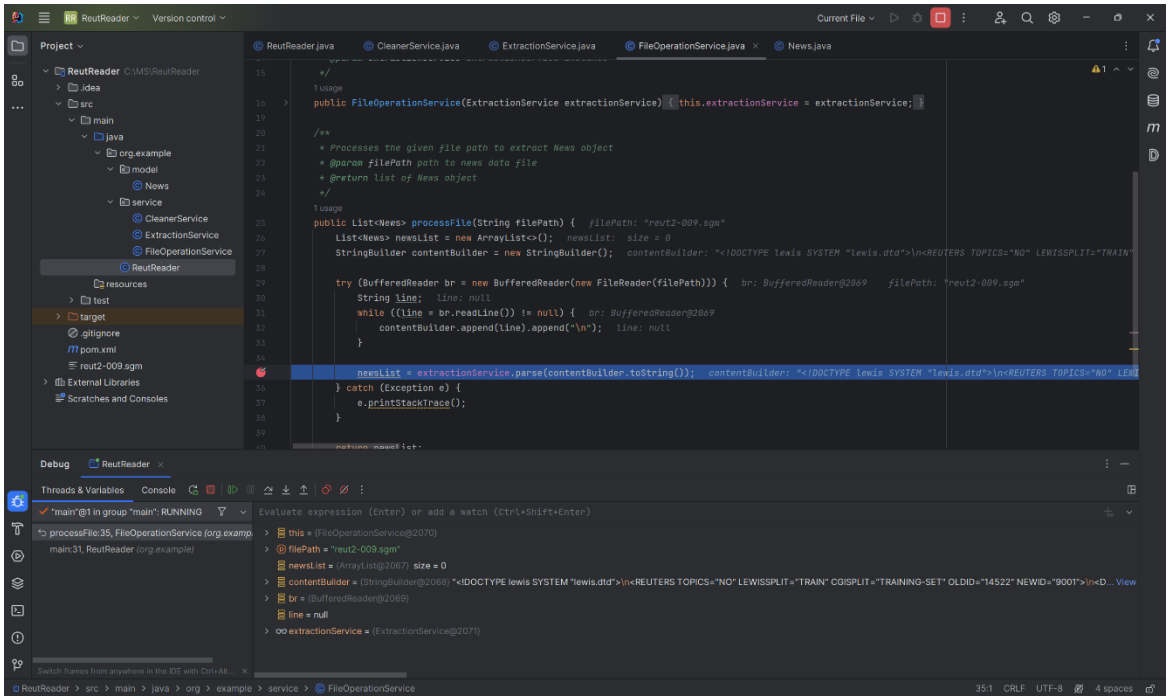


Figure 3 Extraction of raw data into variable

Title tag raw data extracted before the ‘CleanerService’ which currently has HTML entities and special characters.

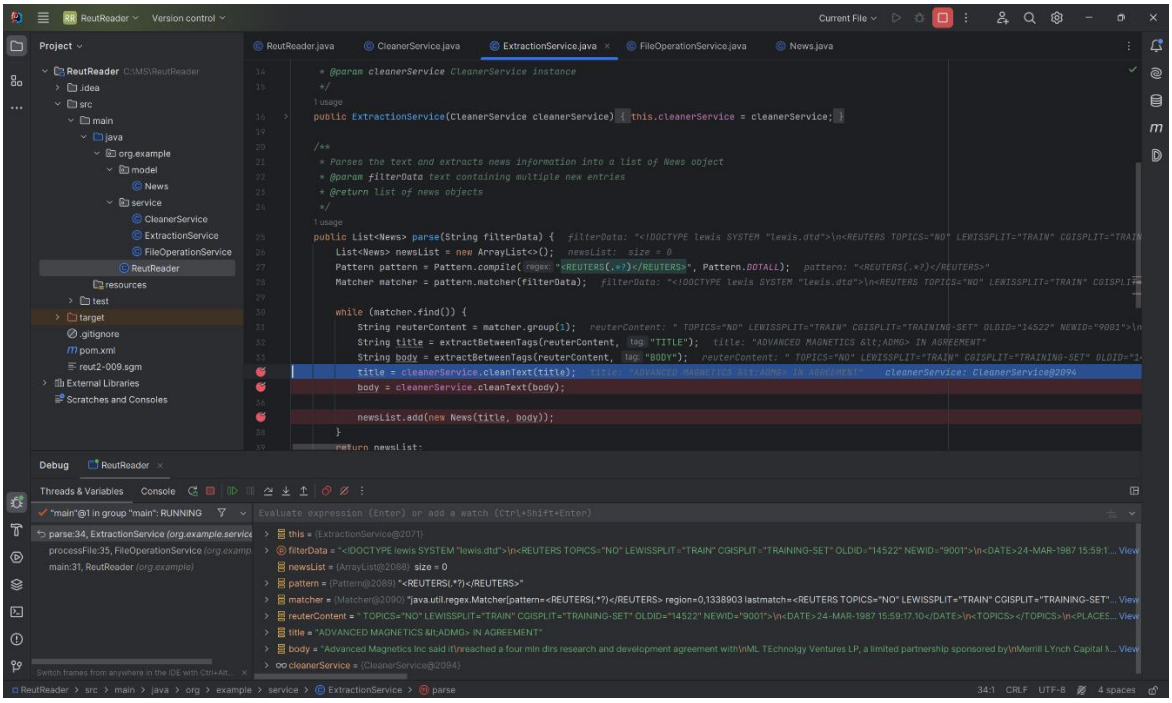


Figure 4 Title tag raw data before CleanerService

Title tag raw data after the ‘CleanerService’ by removing the HTML entities and special characters.

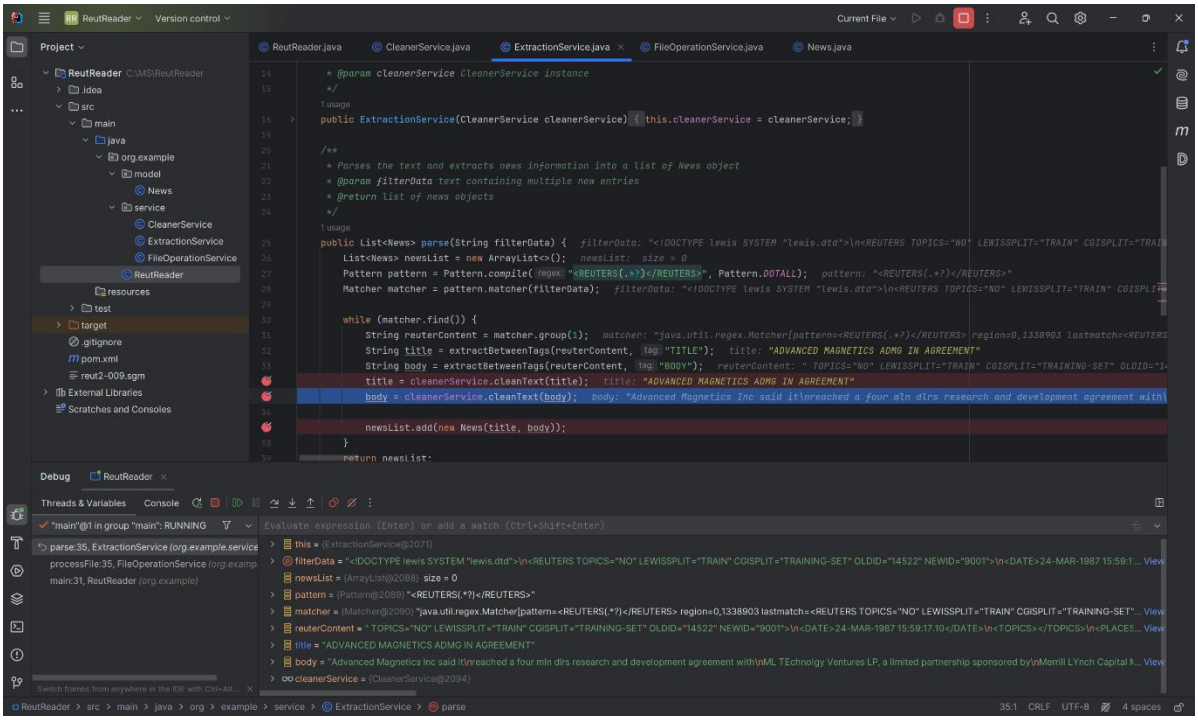


Figure 5 Title tag raw data after CleanerService

Similarly, cleaned body raw data by ‘CleanerService’.

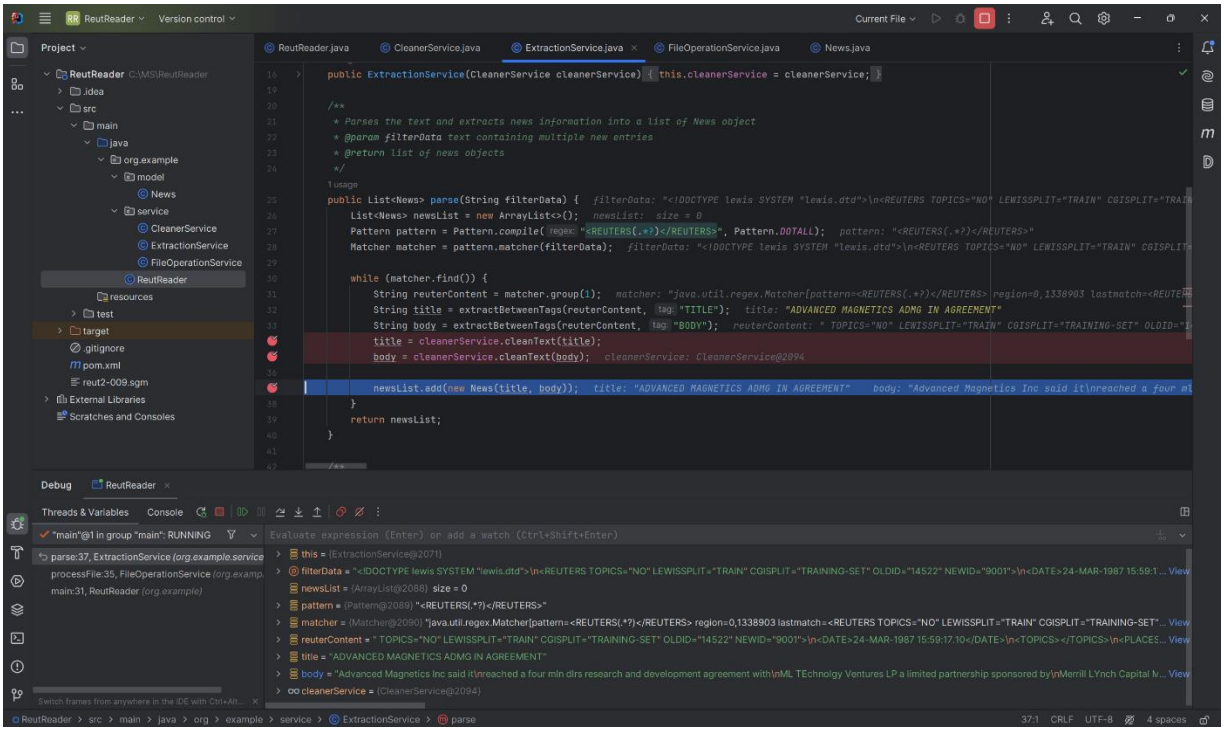


Figure 6 Body tag raw data after CleanerService

After the extraction and cleaning process there are total of 1000 ‘newsList’. Each news is then added to the MongoDB database.

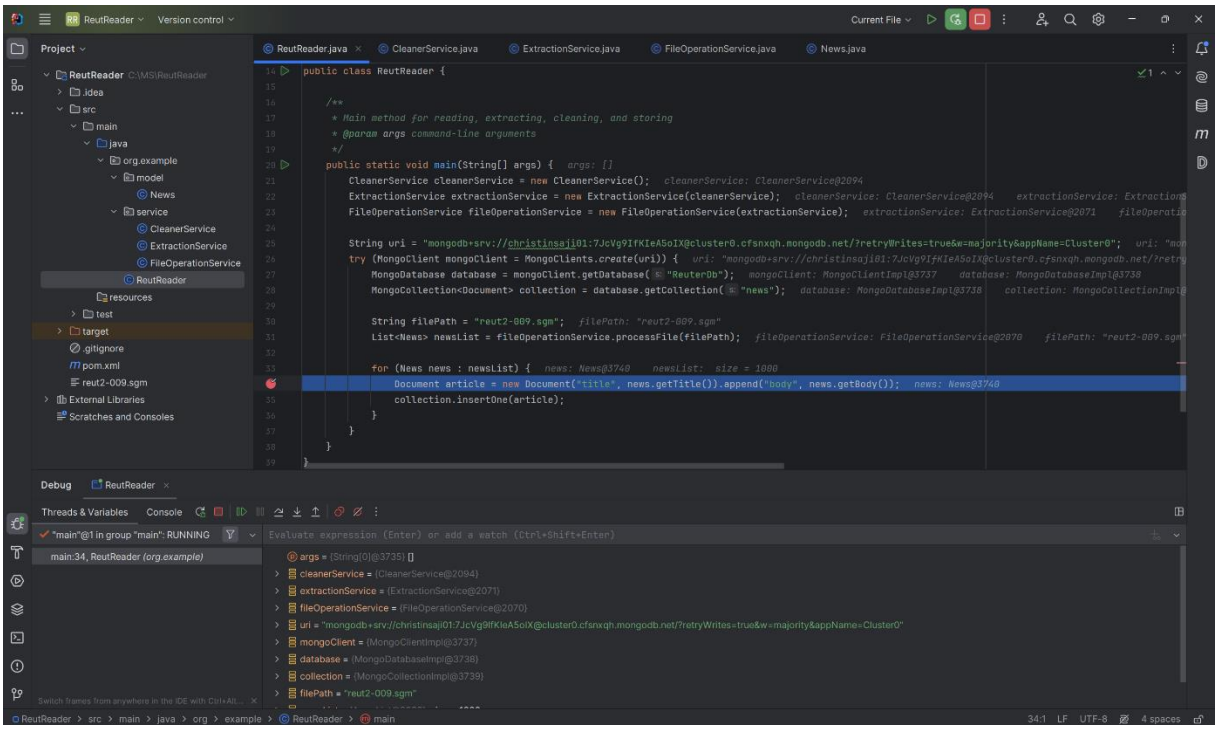


Figure 7 newsList containing 1000 documents

MongoDB containing all the 1000 documents.

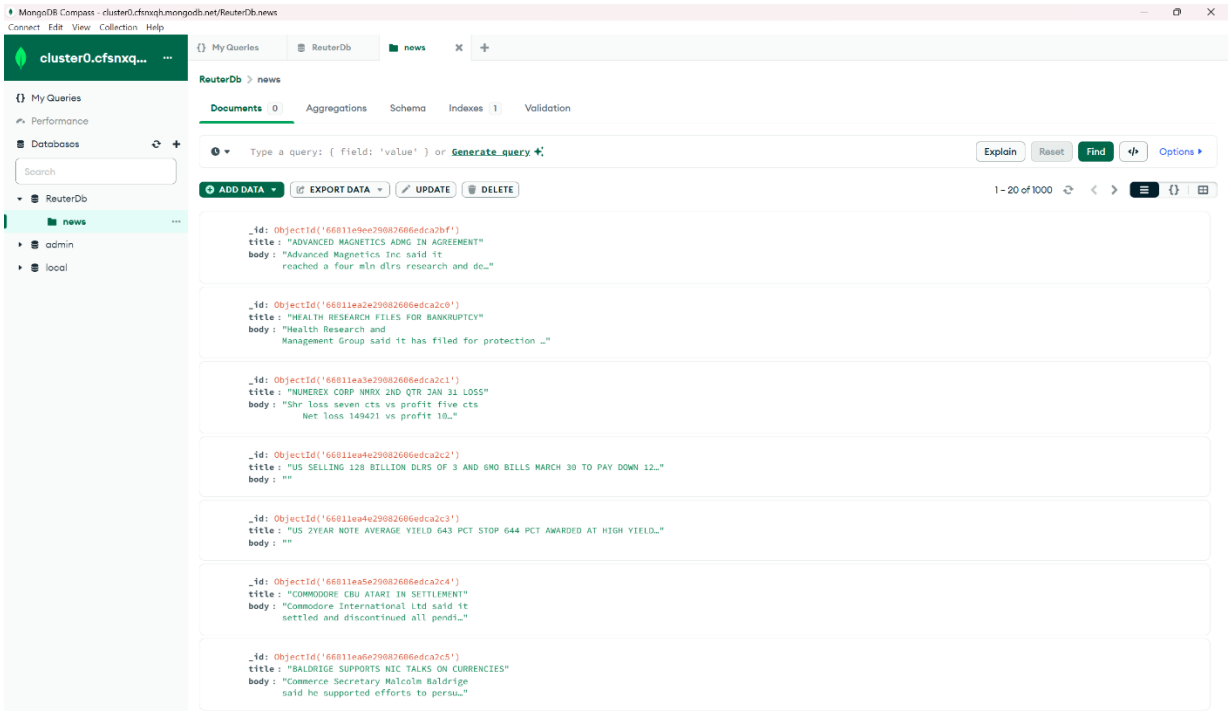


Figure 8 MongoDB database containing the processed data

Problem 1B: Reuter News Data Processing using **Spark**.

Explanation:

First, I prepared the program for deployment by packaging it into a JAR file named `SparkFreqCount-1.0-SNAPSHOT.jar`. This package contains all the necessary code and dependencies to run text analysis and count highest and lowest frequencies.

Next, I navigated to the Google Cloud Platform (GCP) and initiated a new project named "assignment-2." Within this project, I chose to use Dataproc, a service that lets you run Apache Spark projects easily on Google Cloud.

Creating a Dataproc cluster was my next task. This is essentially setting up a group of virtual machines on Google Cloud that work together to process data. I opted for a single node cluster, which means all tasks would be handled by a single virtual machine. This was sufficient for needs and helps to keep costs down.

For the virtual machine, I selected a configuration with 2 CPUs and 8 GB of memory, giving enough power to process our dataset. I also set the disk size to 250 GB to ensure there was plenty of space for data and any temporary files created during process.

After setting up the cluster, I accessed the virtual machine through an SSH browser window. This is like remote-controlling the virtual machine, allowing me to run commands directly on it.

With access to the machine, I uploaded three crucial files: the `SparkFreqCount` JAR package, our dataset file `reut2-009.sgm`, and a list of stop words in `stop-words.txt`. The stop words file is used to filter out common but unimportant words from our analysis.

Finally, I ran the `SparkFreqCount` program by issuing a command through the SSH window. This command instructed Dataproc to use Spark to process our dataset, clean it from any unnecessary content, exclude common stop words, and count how frequently each significant word appeared and list the 5 highest and lowest frequencies.

Algorithm:

Step 1: Initialize the ‘SparkSession’ object with the application name “SparkFreqCount” [2].

Step 2: Check for input arguments for file paths. If not provided, use default file paths for ‘dataFilePath’ and ‘stopWordsFilePath’.

Step 3: Use ‘StopWordsService’ to load stop words from the provided file path. Read all stop words from “stop-words.txt” into a ‘Set<String>’ converting each word to lowercase.

Step 4: Read the raw data text file from “reut02-009.sgm” into a ‘Dataset<Row>’.

Step 5: Clean the raw data by mapping each row to remove HTML tags, HTML entities, non-alphabetic characters, and convert text to lowercase.

Step 6: Split the cleaned data into individual words and flatten the result into a ‘Dataset<String>’ of words.

Step 7: Filter out stop words and words that do not meet length requirements.

Step 8: Group the filtered words by their value and count the occurrence of each word.

Step 9: Sort the word count data in descending order to find the highest frequency words and display the top five words with the highest frequencies. Similarly, for the lowest frequencies by ordering it into ascending.

Step 10: Terminate the Spark session.

Functional Testing:

After writing the Spark program I build the jar file named SparkFreqCount-1.0-SNAPSHOT.jar

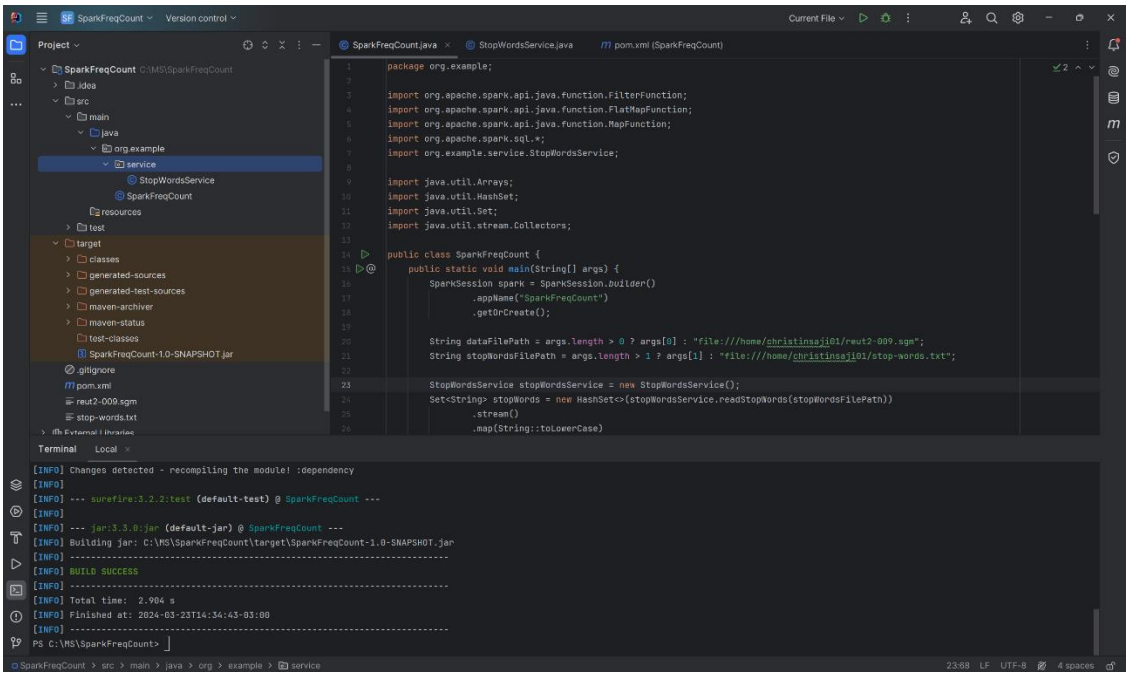


Figure 9 Successful build of SparkFreqCount jar file

In the GCP cloud, I created a project named assignment-2 then selected Dataproc.

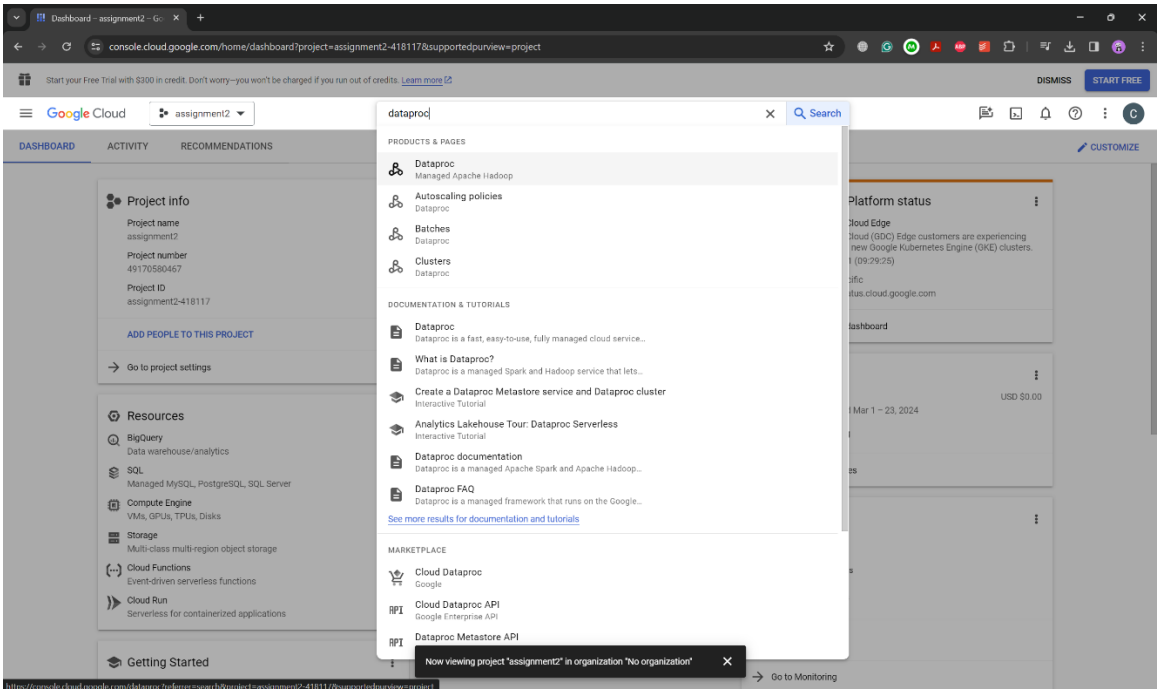


Figure 10 Under assignment-2 selected Dataproc

Created a Cloud Dataproc Cluster.

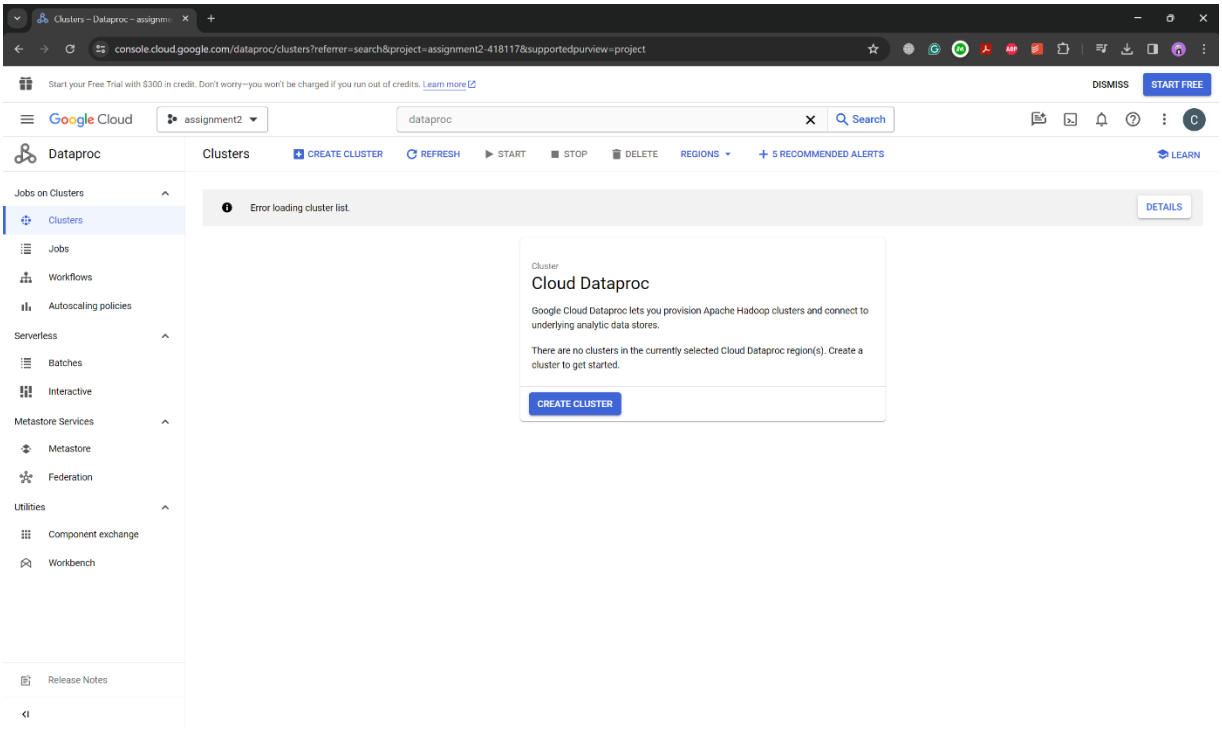


Figure 11 Created a Dataproc Cluster

Created cluster on Compute Engine.

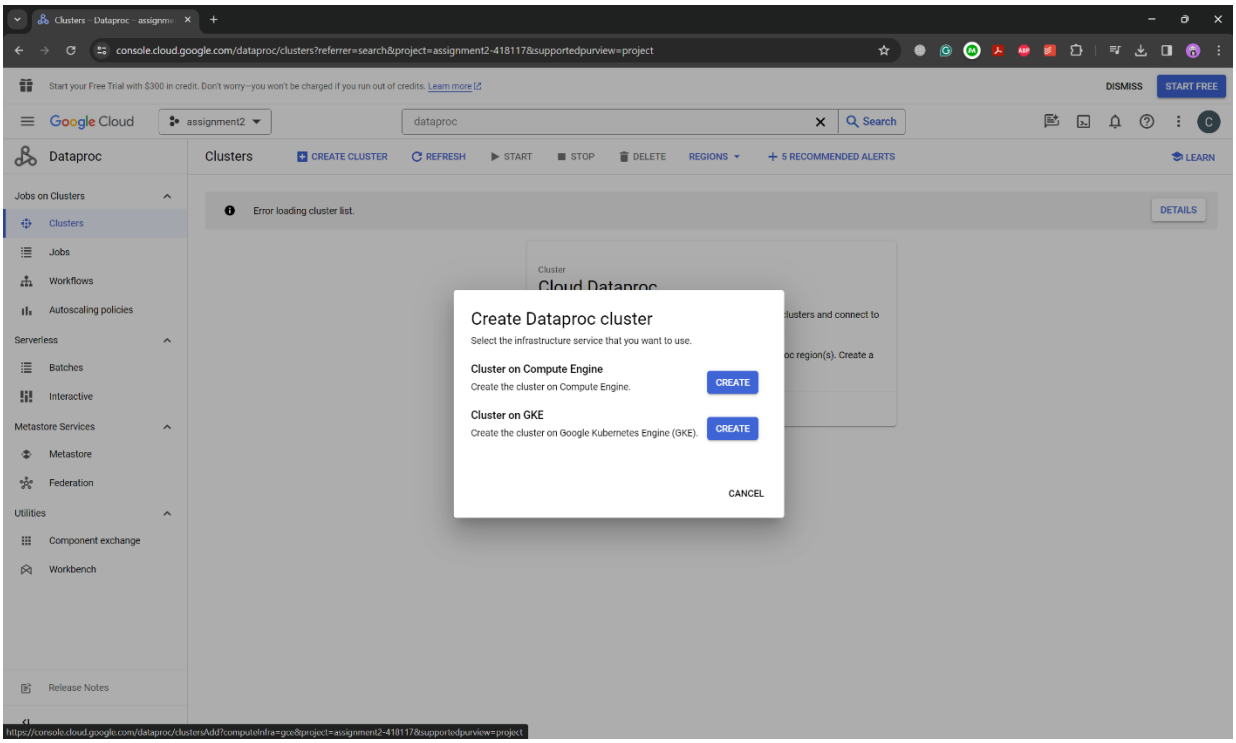


Figure 12 Created cluster on Compute Engine

Renamed the cluster name to sparkfreqcount-cluster and selected single node option for cluster type.

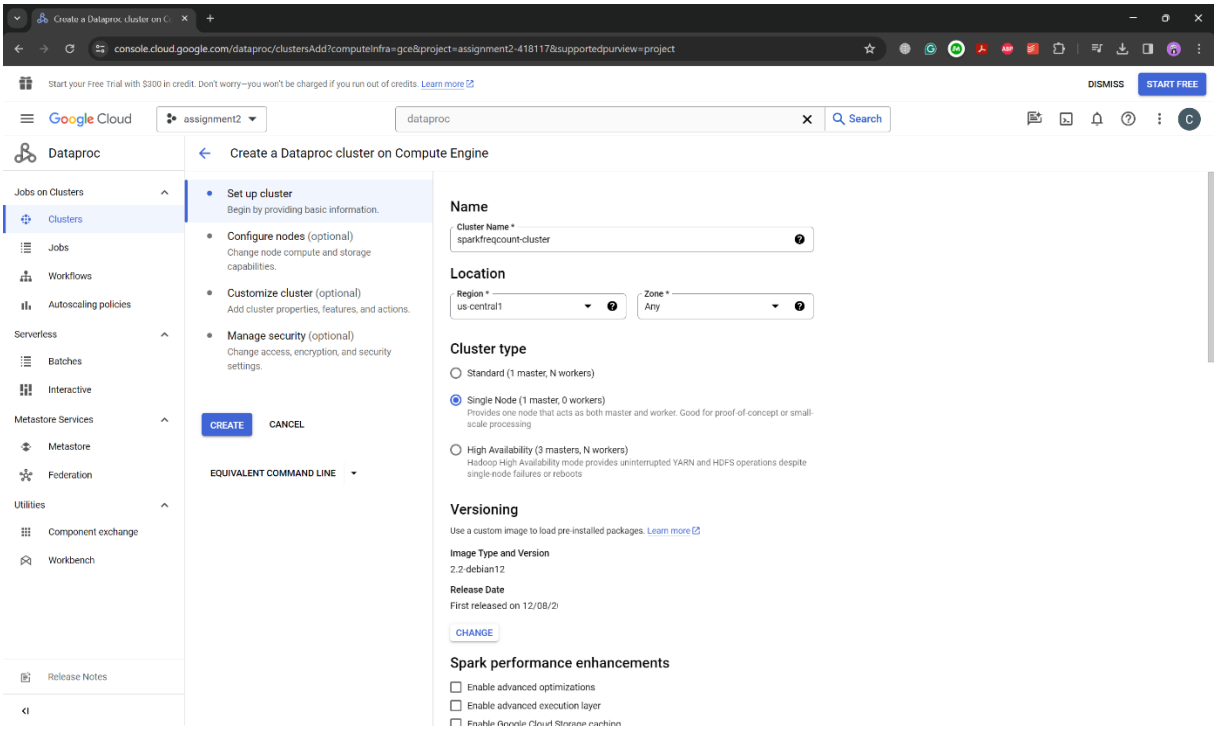


Figure 13 Settings for set up cluster

In configure nodes, I changed machine type to 2vCpu with 8 GB memory and primary disk size to 250 GB.

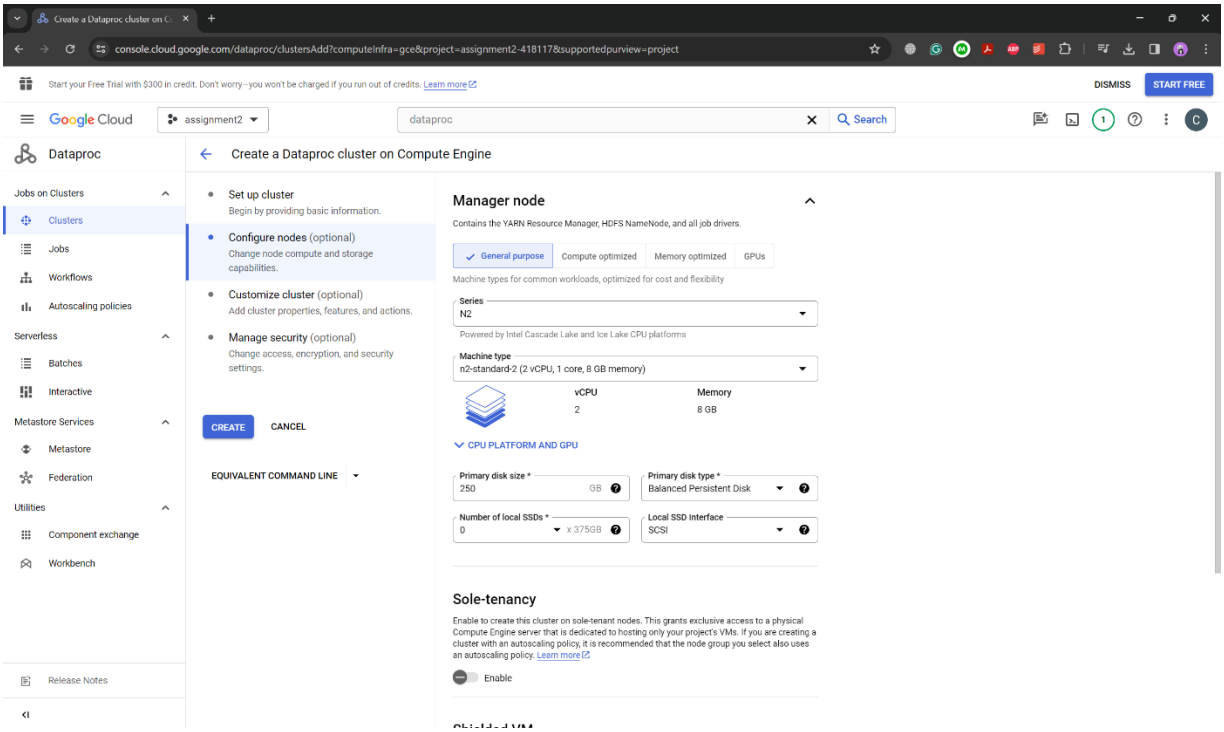


Figure 14 Settings for configure nodes

Selected the sparkfreqcount-cluster.

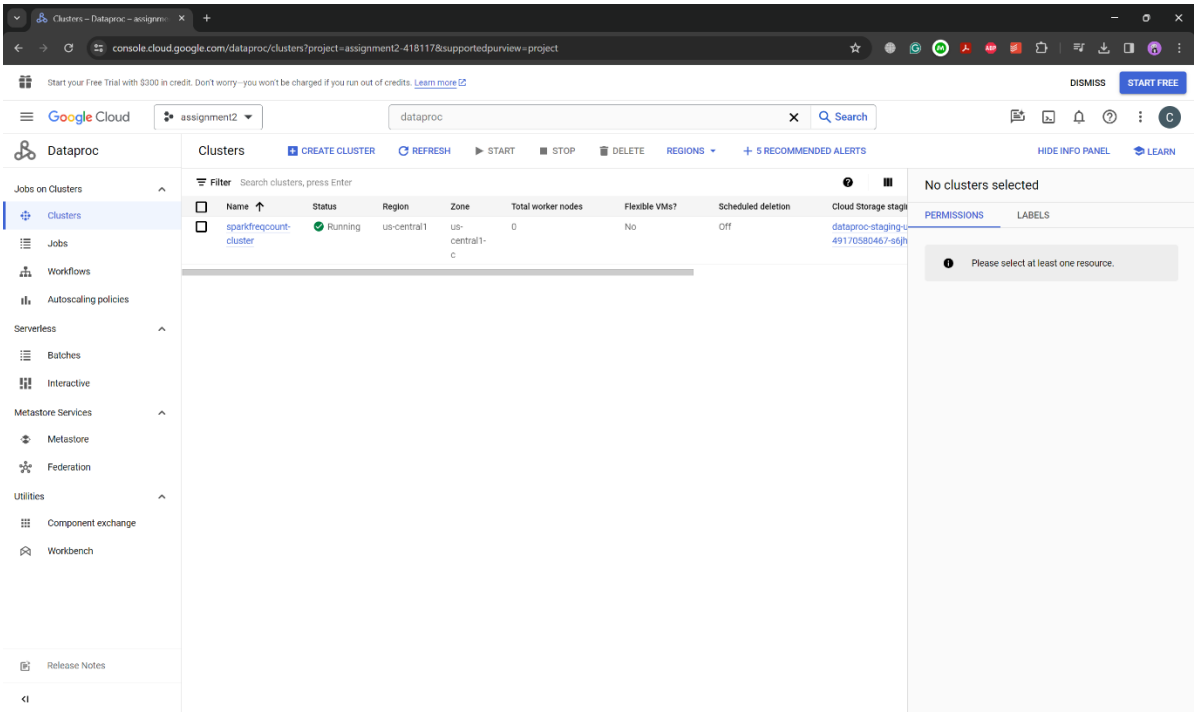


Figure 15 Clusters menu

Then, selected the VM instance of the sparkfreqcount cluster.

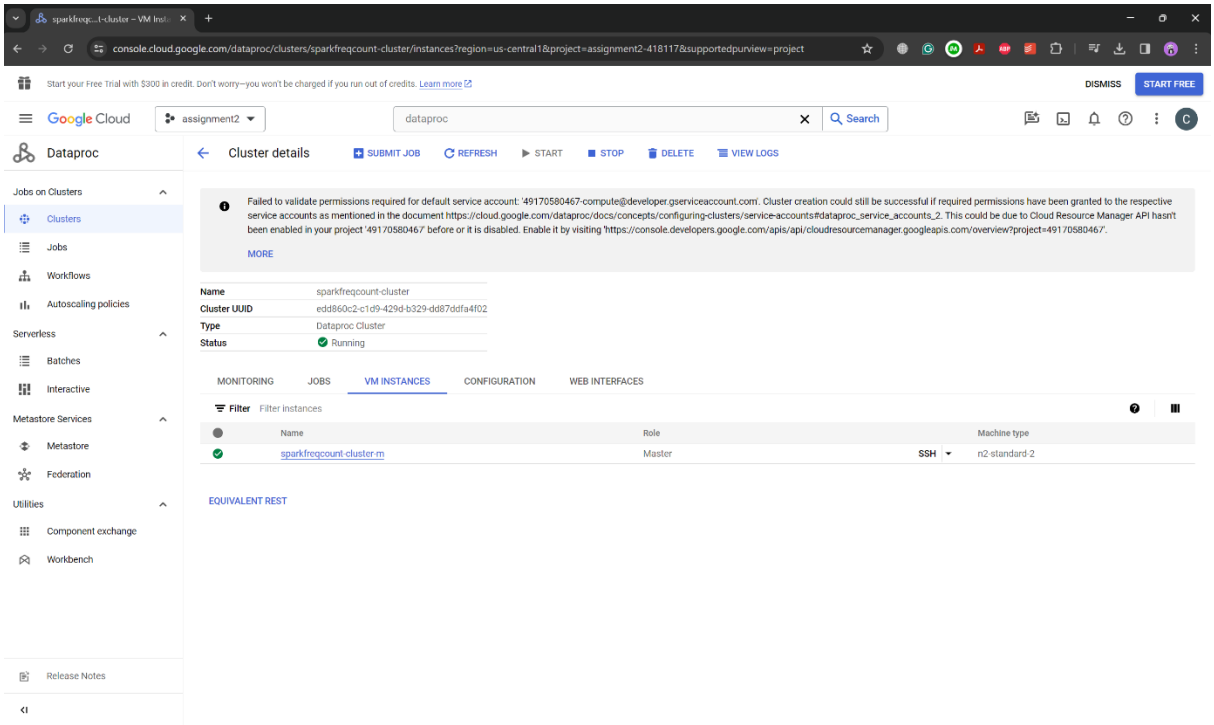


Figure 16 VM instances for sparkfreqcount-cluster

Opened SSH browser window for the sparkfreqcount-cluster-m.

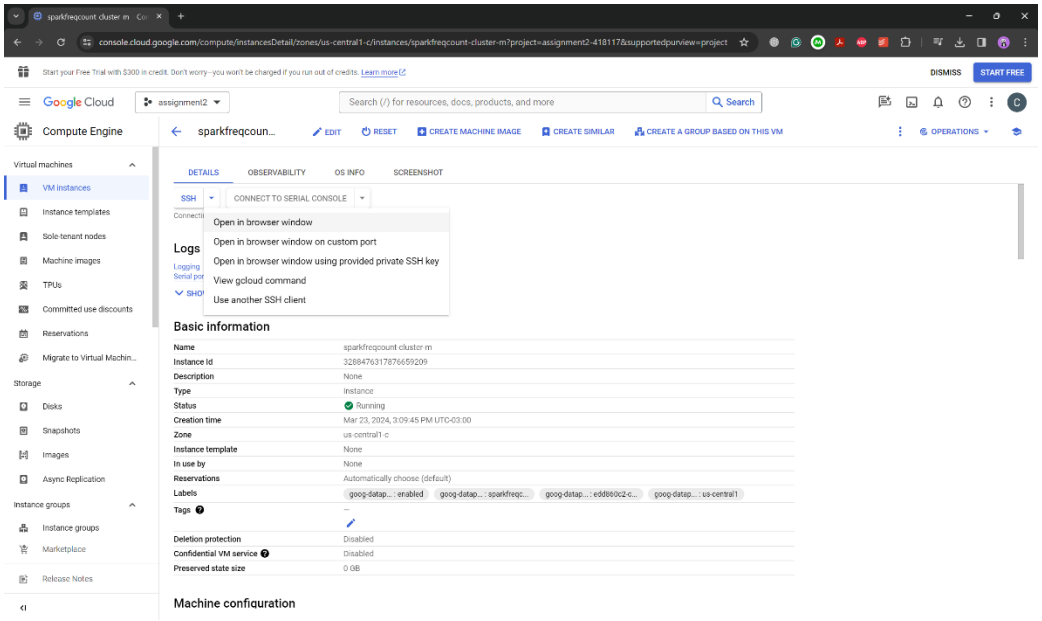


Figure 17 SSH in browser window

I uploaded SparkFreqCount-1.0-SNAPSHOT.jar, reut2-009.sgm, and stop-words.txt.

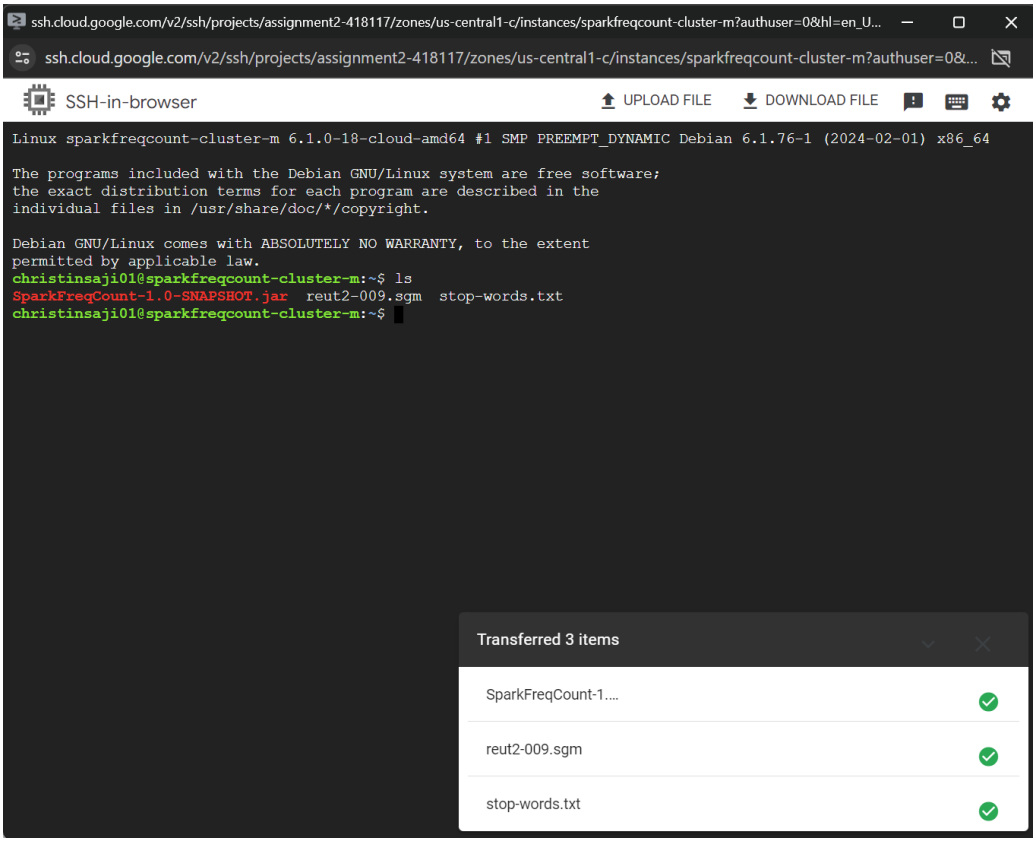


Figure 18 Uploaded files in the SSH cloud

Executed the SparkFreqCount program using the command “spark-submit” –class org.example.SparkFreqCount-1.0-SNAPSHOT.jar file:///home/christinsaji01/reut2-009.sgm stop-words.txt

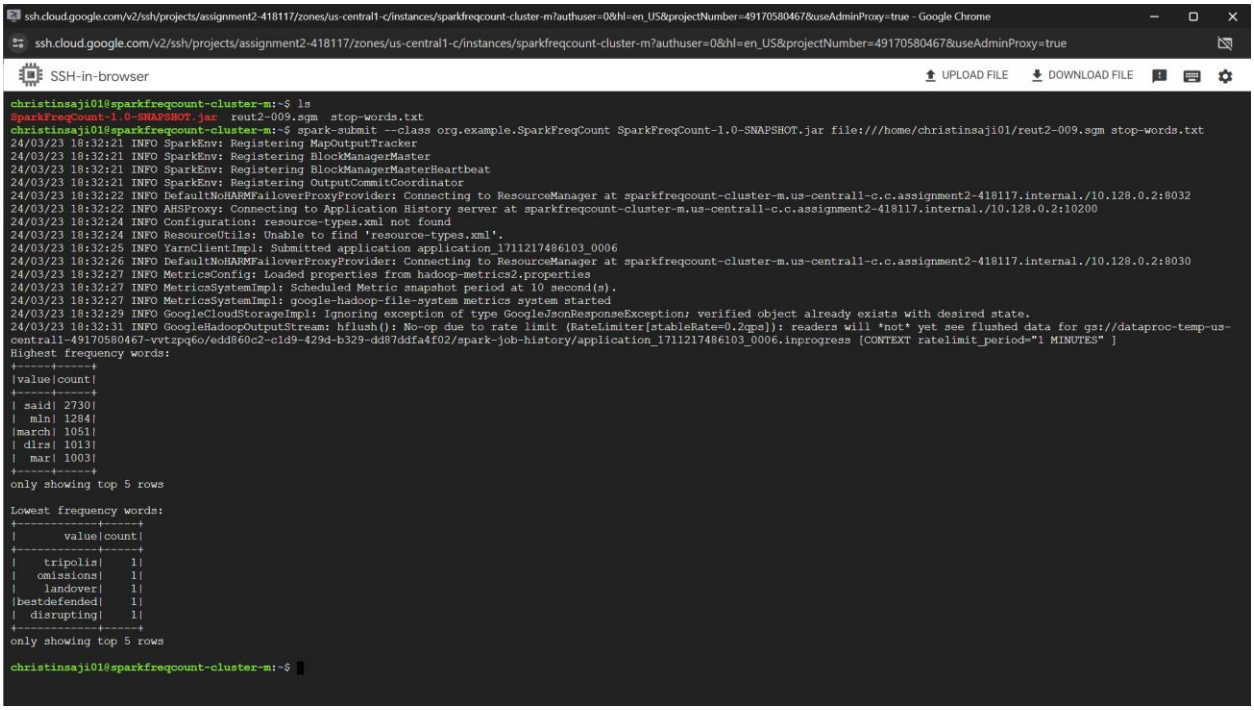


Figure 19 Output of the SparkFreqCount program

Problem 2: Building **Neo4J** Graph Database for relation visualization.

From the Assignment 1, final ERD I took, Student-Enrollment-Program-Course relationship.

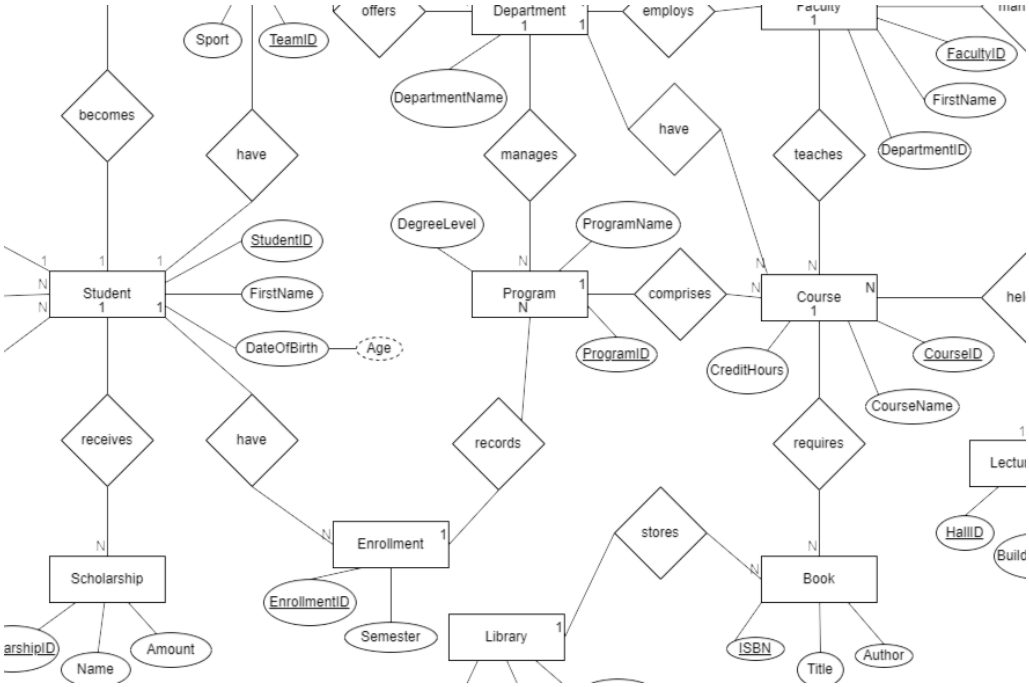


Figure 20 Part of final ERD from Assignment 1 Student-Enrollment-Program-Course relation

Firstly, I created a Assignment 2 project in the Neo4J desktop application.

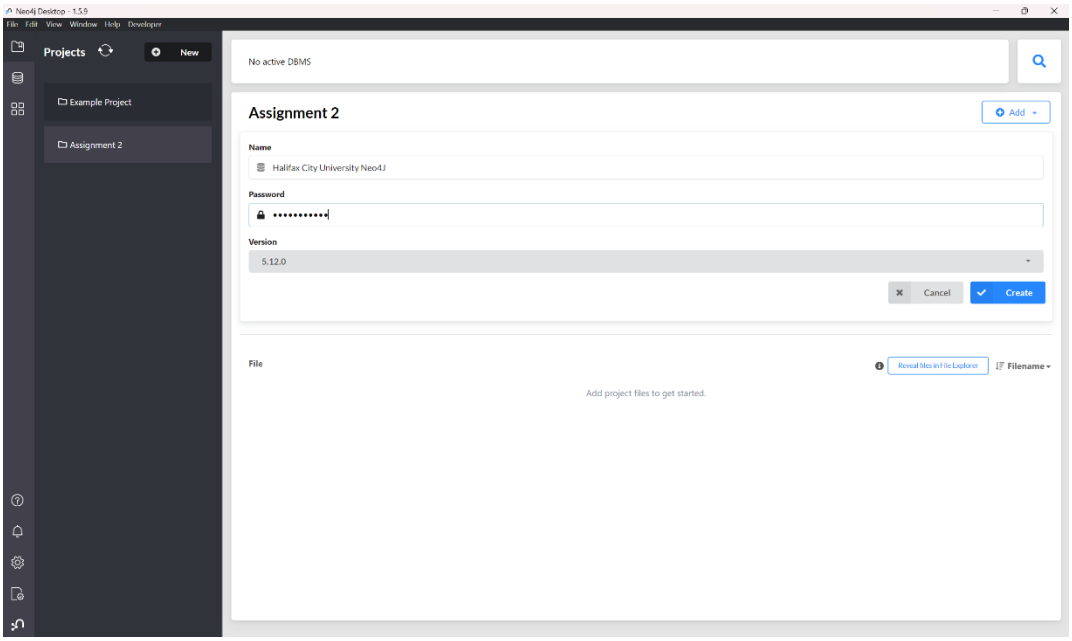


Figure 21 Assignment 2 project in Neo4J application

Inside the Assignment 2 project, I created database named Halifax City University Neo4J and activated it.

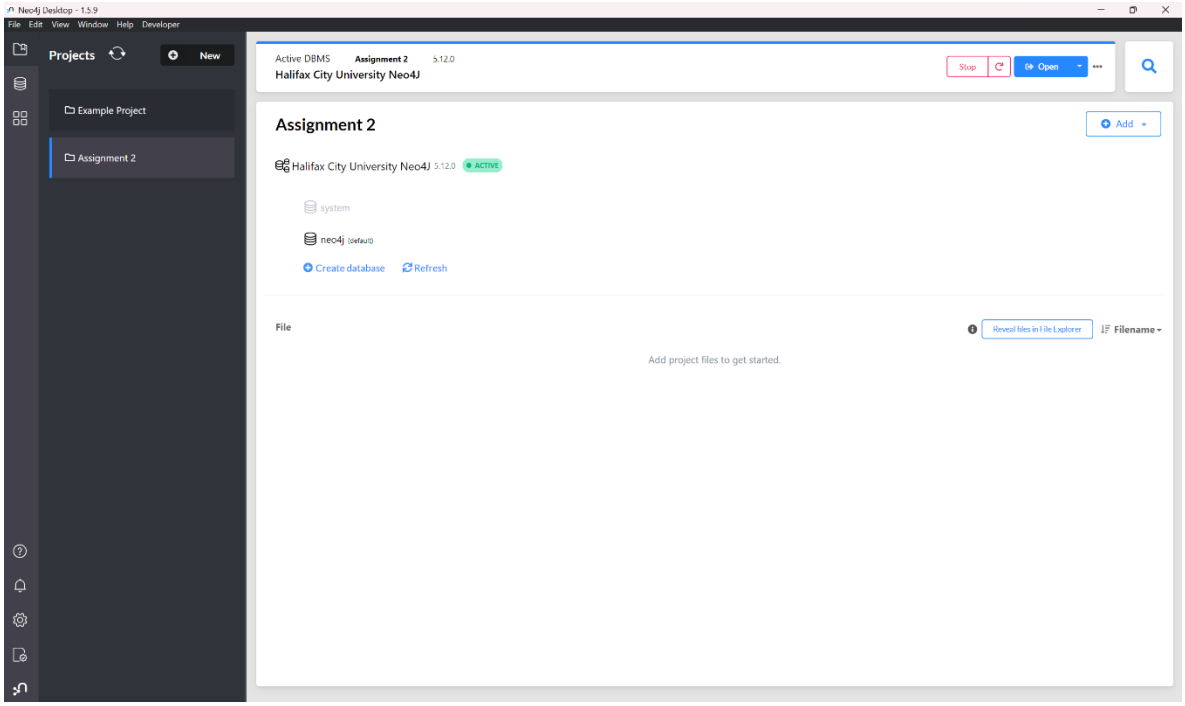


Figure 22 Halifax City University Neo4J database inside Assignment 2

Then I opened the Neo4J Browser from the apps section in the Neo4J Desktop application to run the cypher query language.

I began by establishing the primary entities as nodes within graph. Each `Student` node captures essential attributes like ‘StudentID’, ‘FirstName’ and ‘DateOfBirth’. `Enrollment` nodes is tied to a specific semester. `Program` nodes encapsulate distinct academic pathways such as 'MACS' and 'Data Science,' detailing the degree level. `Course` nodes represent individual subjects, complete with unique identifiers and descriptive names.

Students are linked to their respective enrollments through `:ENROLLED` relationships, suggesting a enrollment in the institution [3]. The `:RECORDS` relationships from enrollments to programs establish which courses are undertaken during a particular enrollment period. Finally, programs are decomposed into the courses they comprise through `:COMPRISES` relationships, allowing us to trace a student's academic trajectory down to the specific subjects studied.

Using Cypher's `CREATE` statements, I created the nodes and relationships. Each `CREATE` action sequentially built graph, starting with students to the courses they study.

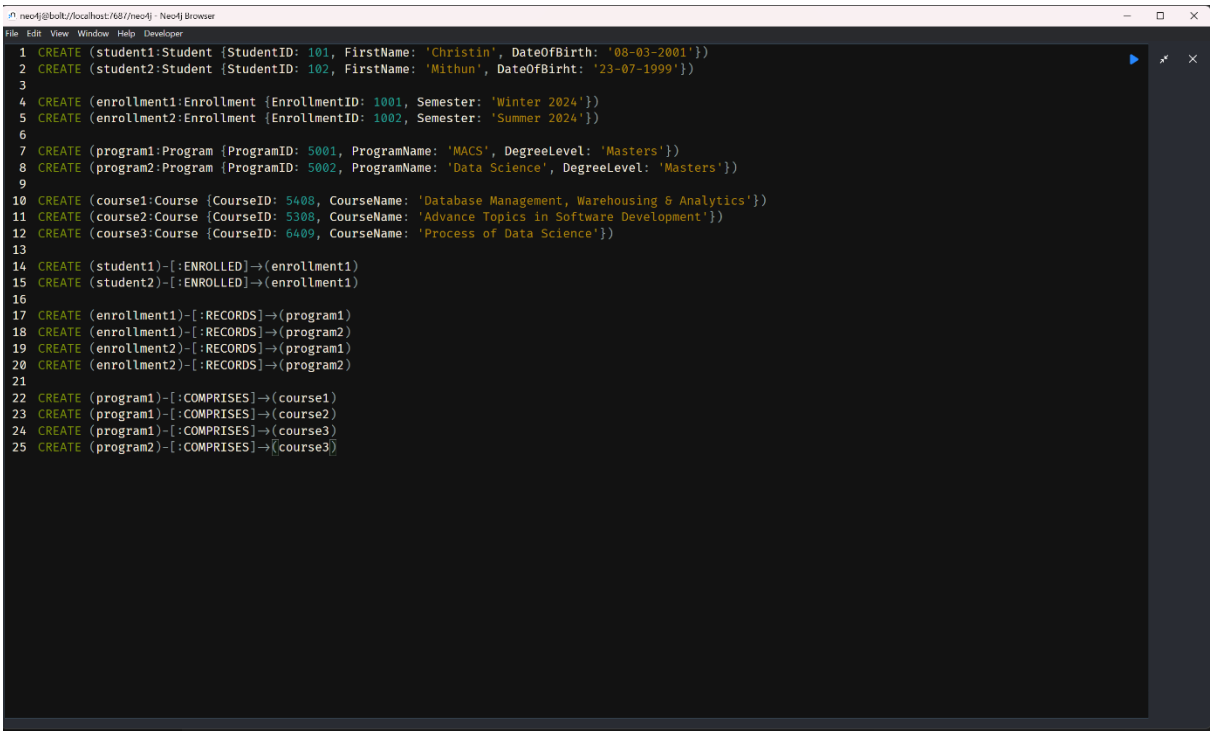


Figure 23 Cypher query to create node and make relationship

Finally, I used cypher query 'MATCH (n) RETURN n' to select and return all nodes in the graph database.

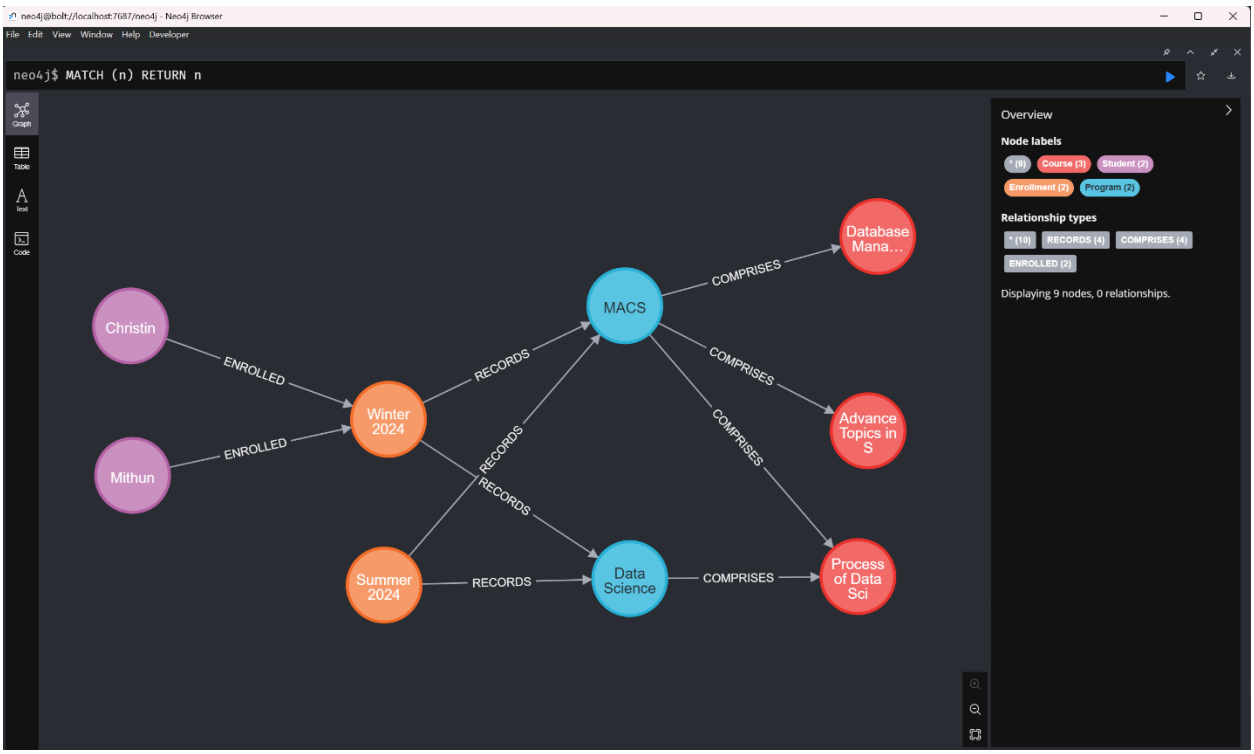


Figure 24 Displaying the nodes of the graph database

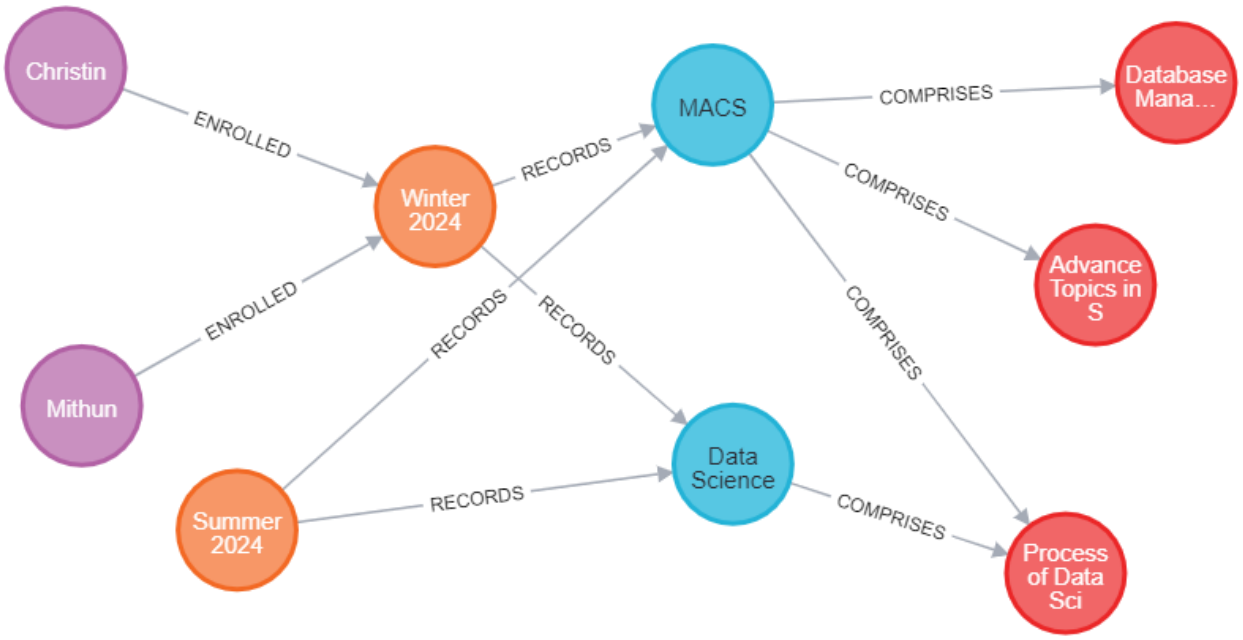


Figure 25 Final Neo4J Graph

Problem 3: Sentiment Analysis using BOW model on title of Reuters News Articles.

First, I loaded the positive-words.txt and negative-words.txt. It had 2006 positive words and 4783 negative words.

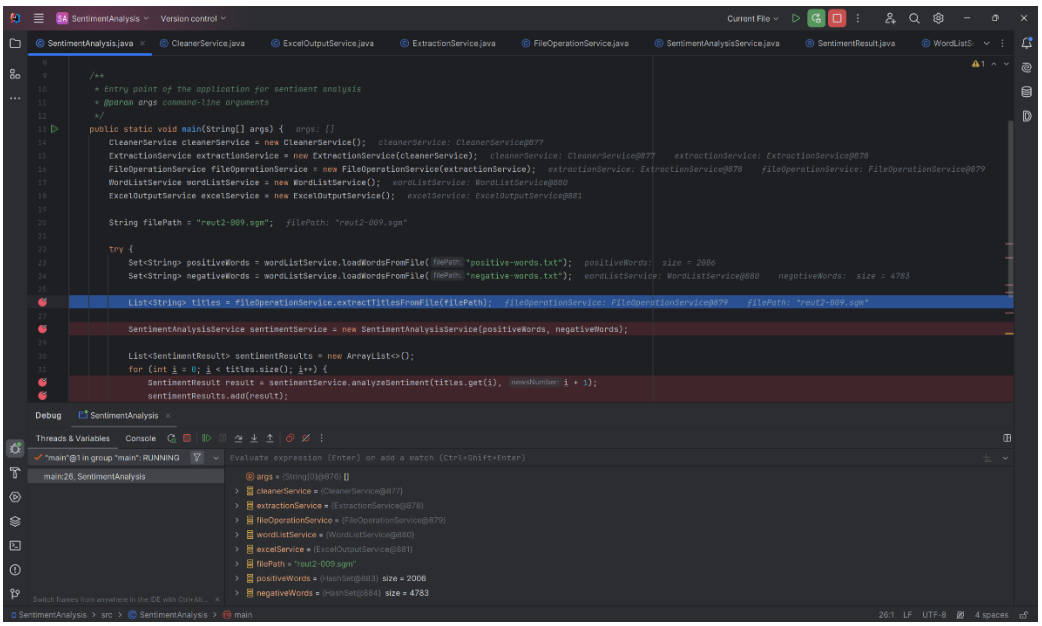


Figure 26 positive-words.txt and negative-words.txt loaded on the program

Extracted the whole raw data from the 'reut2-009.sgm' file on the 'contentBuilder' variable.

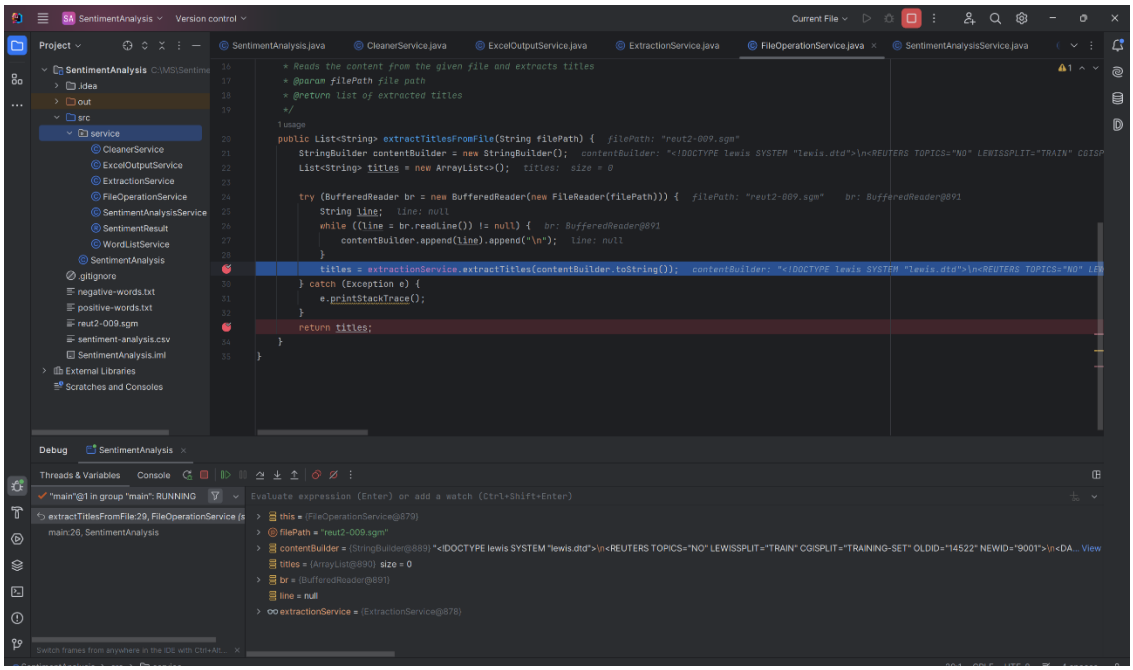


Figure 27 contentBuilder variable having whole raw data

Then, extracted the ‘TITLE’ tag from inside the ‘REUTES’ tag, which is currently having html entities and special characters inside it.

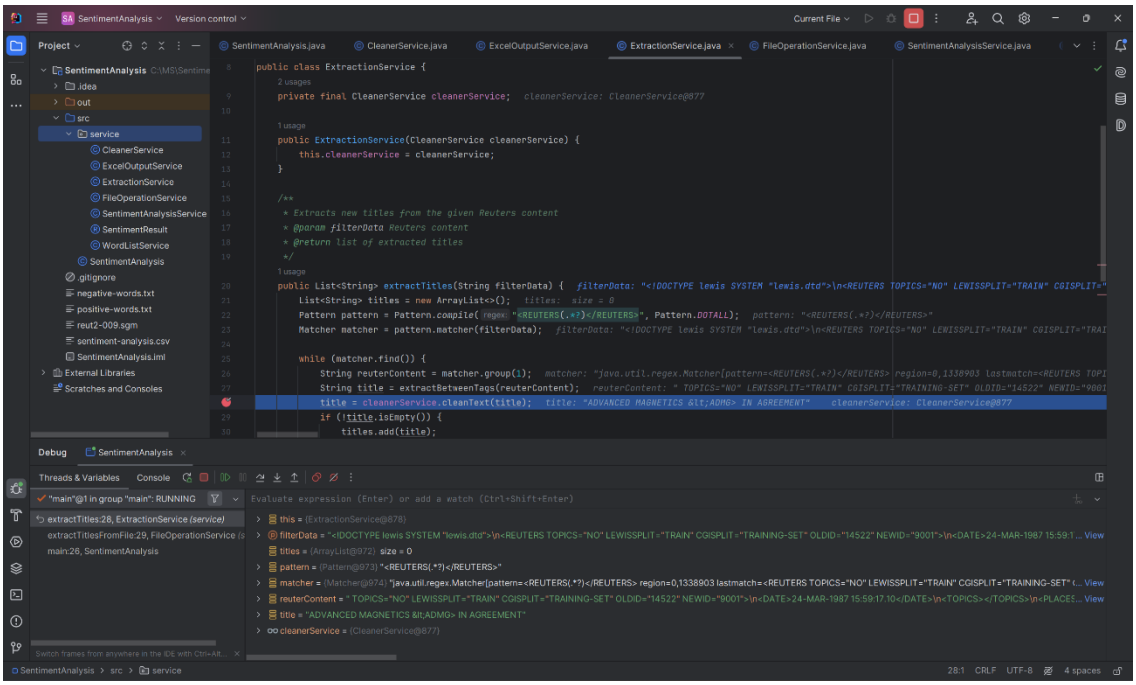


Figure 28 Title tag before the CleanerService

Cleaned out TITLE tags data using the ‘CleanerService’ which removes the html entities and special characters.

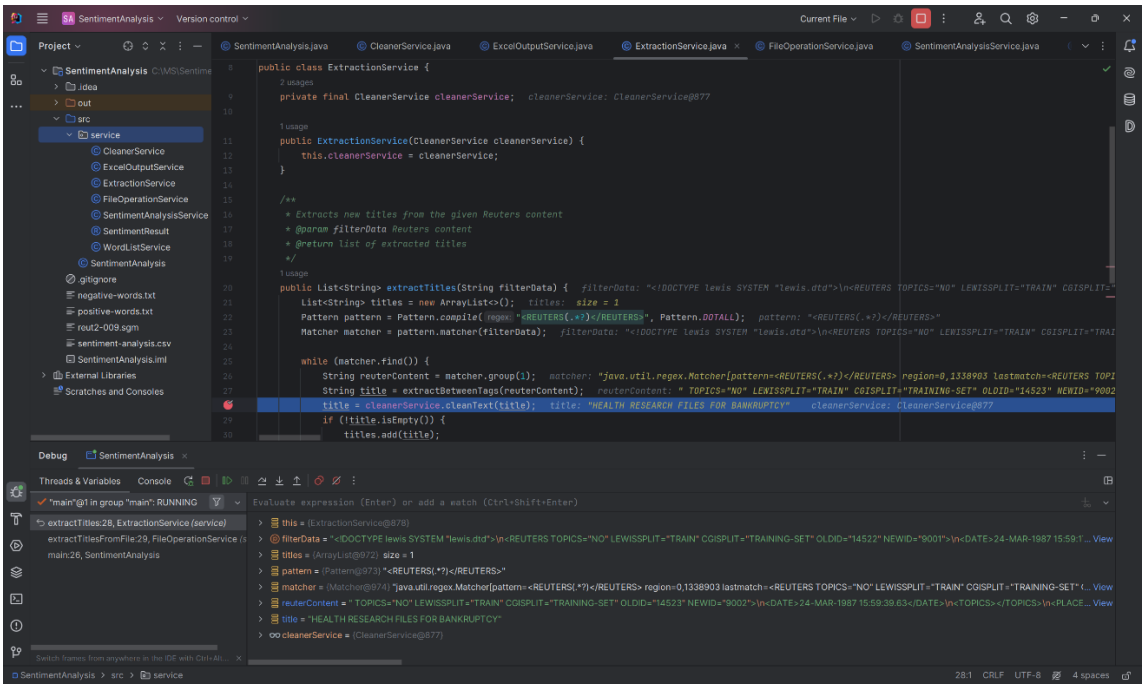


Figure 29 Cleaned data after the CleanerService

After extracting and cleaning the raw data there are about total 996 titles for which Sentimental analysis is to be done.

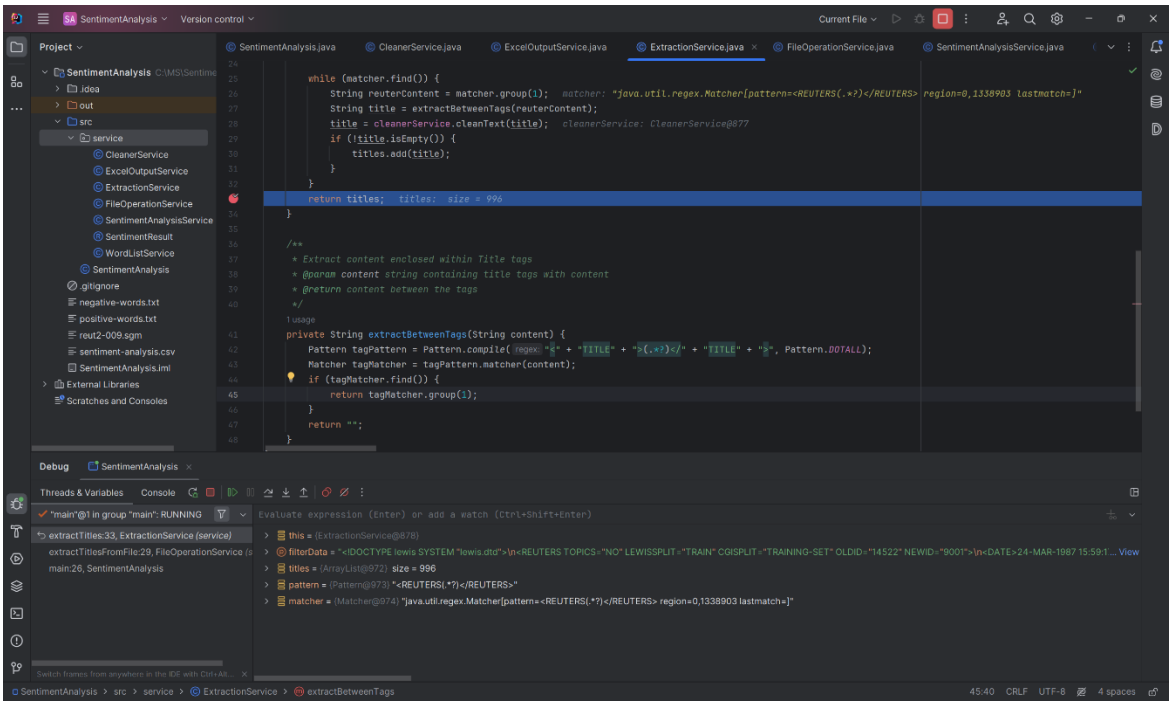


Figure 30 Number of titles in the file

Each title is converted to a List<String> carrying a bag of words for the process of sentimental analysis.

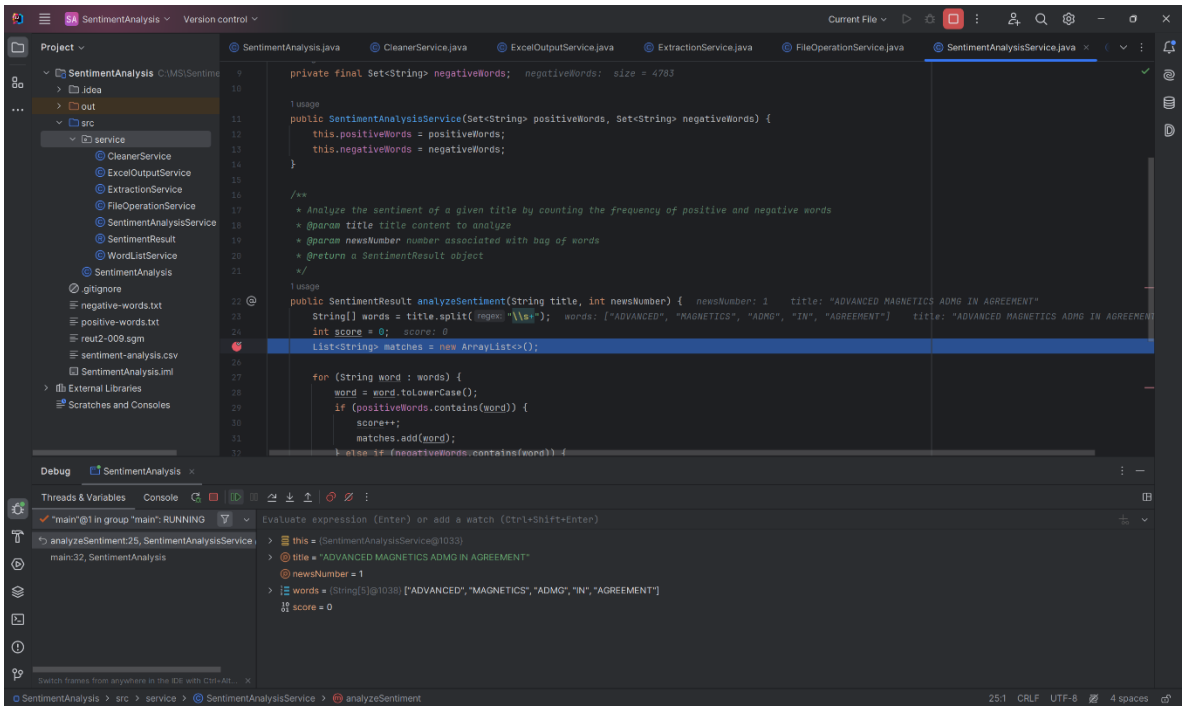


Figure 31 Bag of words of a title

Each bag of words is then compared to the positive words and negative words. If a word is matched to one of the positive words, then score is increased by 1 or decreased by 1 if it matches to a negative word. The matched words are scored in a ArrayList so that they can be displayed comma separated under one column while displaying in the csv file. If the total score is greater than 0 then the polarity is set to 'Positive' or if it's less than 0 then it is set to 'Neutral', and finally is score is 0 then polarity is set to 'Neutral'.

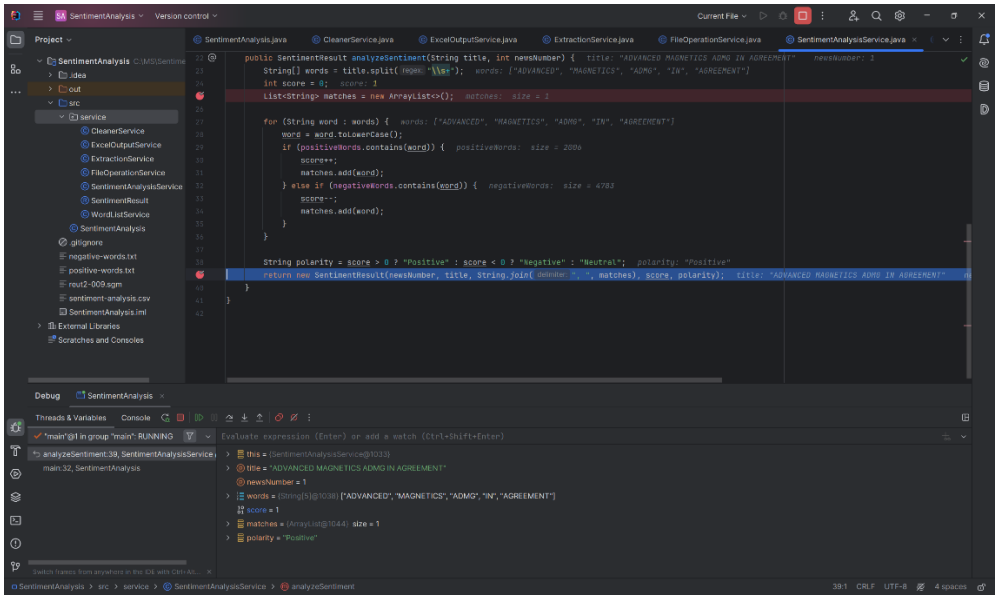


Figure 32 Sentimental Analysis of a title

Then after the sentimental analysis the object values are store in the csv file named 'sentimental-analysis.csv'.

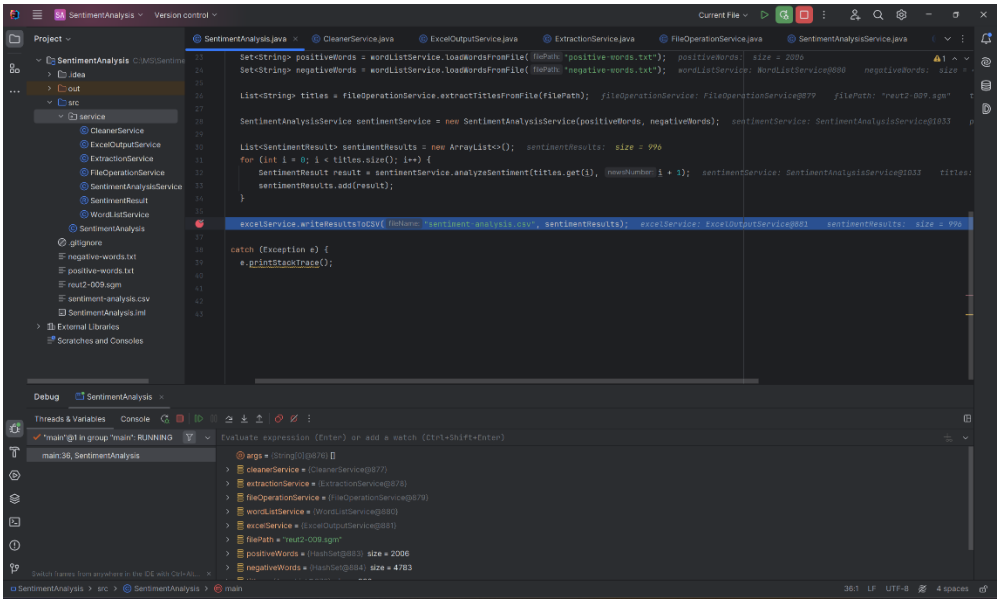


Figure 33 sentimental-analysis file creation

Finally, while printing in the csv file it is formatted as mentioned in the task.

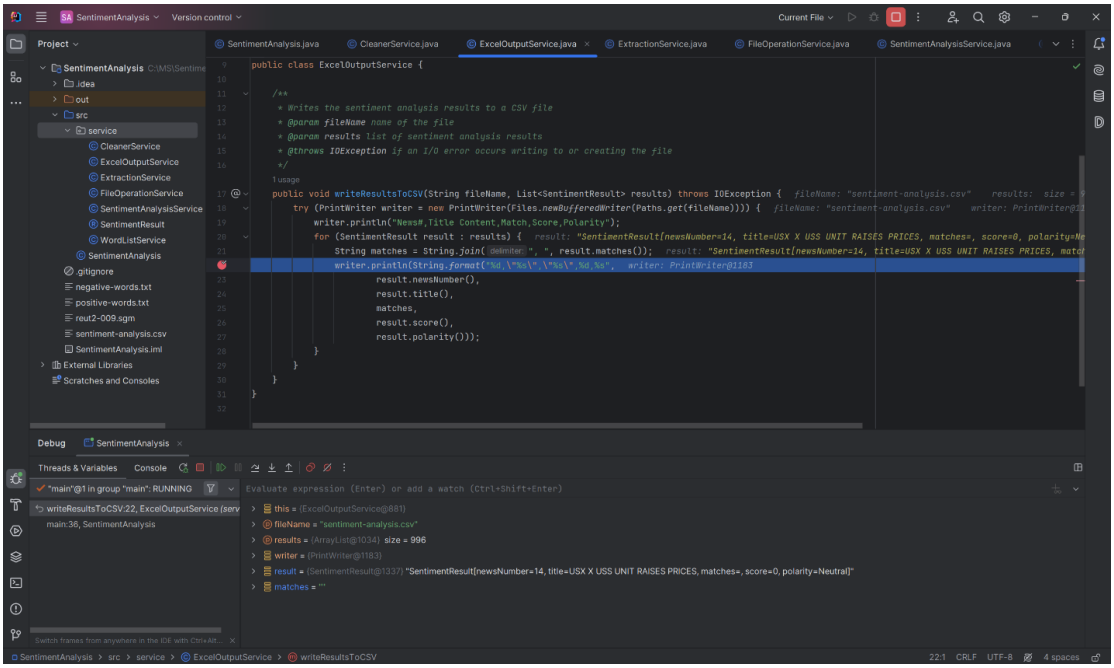


Figure 34 Formatting of the content

Output:

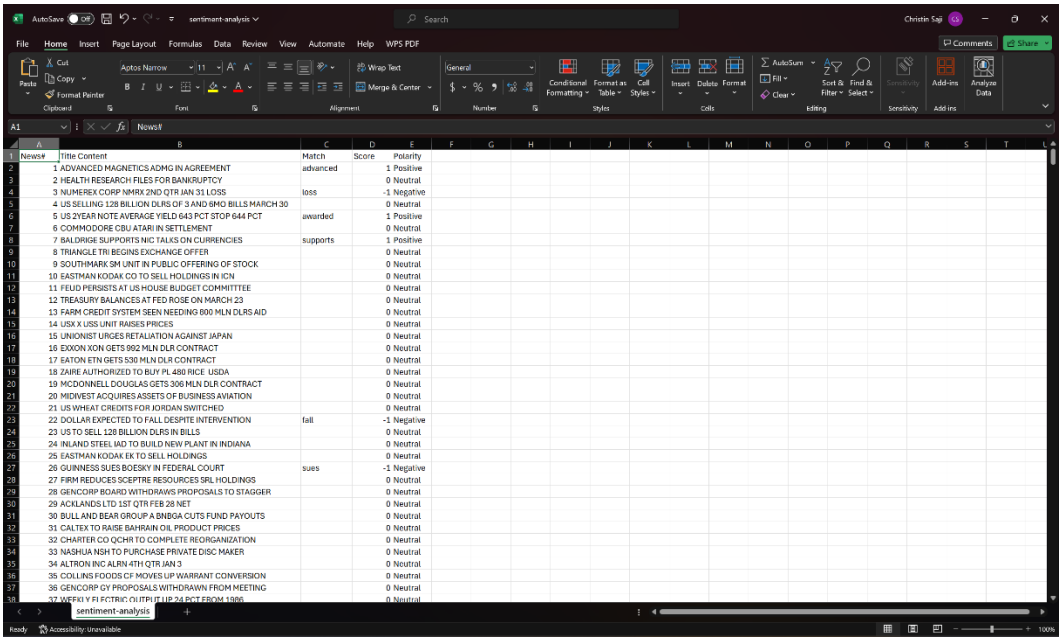
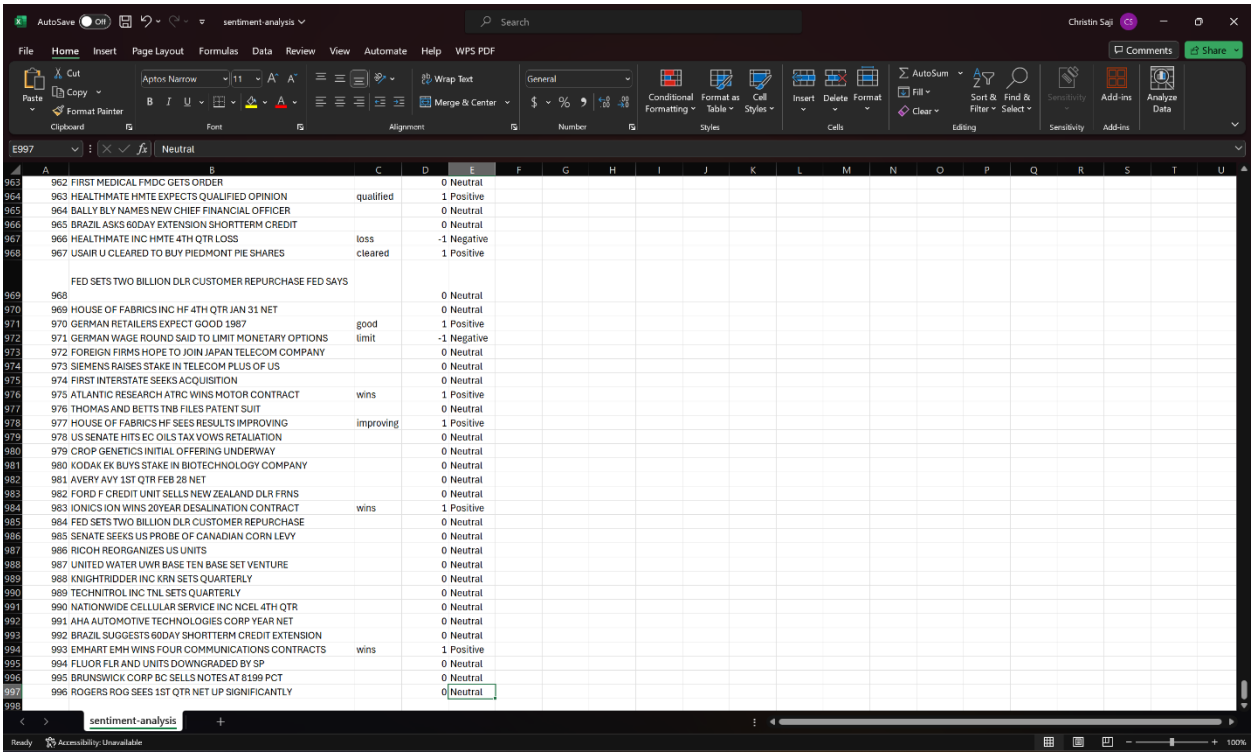


Figure 35 Starting sentimental analysis row from csv file



963	962	FIRST MEDICAL FMDC GETS ORDER		0	Neutral
964	963	HEALTHMATE HMTE EXPECTS QUALIFIED OPINION	qualified	1	Positive
965	964	BALLY BLY NAMES NEW CHIEF FINANCIAL OFFICER		0	Neutral
966	965	BRAZIL ASKS 60DAY EXTENSION SHORTTERM CREDIT		0	Neutral
967	966	HEALTHMATE INC HMTE 4TH QTR LOSS	loss	-1	Negative
968	967	USAIR U CLEARED TO BUY PIEDMONT PIE SHARES	cleared	1	Positive
		FED SETS TWO BILLION DLR CUSTOMER REPURCHASE FED SAYS			
969	968			0	Neutral
970	969	HOUSE OF FABRICS INC HF 4TH QTR JAN 31 NET		0	Neutral
971	970	GERMAN RETAILERS EXPECT GOOD 1987	good	1	Positive
972	971	GERMAN WAGE ROUND SAID TO LIMIT MONETARY OPTIONS	limit	-1	Negative
973	972	FOREIGN FIRMS HOPE TO JOIN JAPAN TELECOM COMPANY		0	Neutral
974	973	SIEMENS RAISES STAKE IN TELECOM PLUS OF US		0	Neutral
975	974	FIRST INTERSTATE SEEKS ACQUISITION		0	Neutral
976	975	ATLANTIC RESEARCH ATRC WINS MOTOR CONTRACT	wins	1	Positive
977	976	THOMAS AND BETTS TNB FILES PATENT SUIT		0	Neutral
978	977	HOUSE OF FABRICS HF SEES RESULTS IMPROVING	improving	1	Positive
979	978	US SENATE HITS EC OILS TAX VOWS RETALIATION		0	Neutral
980	979	CROP GENETICS INITIAL OFFERING UNDERWAY		0	Neutral
981	980	KODAK EK BUYS STAKE IN BIOTECHNOLOGY COMPANY		0	Neutral
982	981	AVERY AVY 1ST QTR FEB 28 NET		0	Neutral
983	982	FORD F CREDIT UNIT SELLS NEW ZEALAND DLR FRNS		0	Neutral
984	983	IONICS ION WINS 20YEAR DESALINATION CONTRACT	wins	1	Positive
985	984	FED SETS TWO BILLION DLR CUSTOMER REPURCHASE		0	Neutral
986	985	SENATE SEEKS US PROBE OF CANADIAN CORN LEVY		0	Neutral
987	986	RICOH REORGANIZES US UNITS		0	Neutral
988	987	UNITED WATER UWR BASE TEN BASE SET VENTURE		0	Neutral
989	988	KNIGHTRIDDER INC KRN SETS QUARTERLY		0	Neutral
990	989	TECHNITROL INC TNL SETS QUARTERLY		0	Neutral
991	990	NATIONWIDE CELLULAR SERVICE INC NCEL 4TH QTR		0	Neutral
992	991	AHA AUTOMOTIVE TECHNOLOGIES CORP YEAR NET		0	Neutral
993	992	BRAZIL SUGGESTS 60DAY SHORTTERM CREDIT EXTENSION		0	Neutral
994	993	EMHART EMH WINS FOUR COMMUNICATIONS CONTRACTS	wins	1	Positive
995	994	FLUOR FLR AND UNITS DOWNGRADED BY SP		0	Neutral
996	995	BRUNSWICK CORP BC SELLS NOTES AT 8199 PCT		0	Neutral
997	996	ROGERS ROG SEES 1ST QTR NET UP SIGNIFICANTLY		0	Neutral

Figure 36 Ending sentimental analysis row from csv file

References

[1] "Quick Start - Java Sync Driver," MongoDB, [Online]. Available: https://www.mongodb.com/docs/drivers/java/sync/current/quick-start/ [Accessed: March 21, 2024].

[2] Naveen (NNK), "Spark Read Text File: RDD: DataFrame," *Spark By {Examples}*, [Online], Feb 8, 2023. Available: https://sparkbyexamples.com/spark/spark-read-text-file-rdd-dataframe/ [Accessed: March 21, 2024].

[3] "Query a neo4j database using Cypher - Getting Started," Neo4j Graph Data Platform, [Online]. Available: https://neo4j.com/docs/getting-started/cypher-intro/ [Accessed: March 22, 2024].