CSCI 5408

DATA MANAGEMENT AND WAREHOUSING

Assignment-2

Banner ID: B00984408
GitLab Assignment Link:
https://git.cs.dal.ca/jems/csci5408_s24_b00984406_jems_patel

Contents

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

Algorithm:

1. Initialize MongoDB Connection
    - Create a `DatabaseConnection` object with the MongoDB URI and database name.

2. Initialize MongoDB Inserter
    - Create a `MongoInsert` object with the database connection.

3. Read Articles from File
    - Initialize an empty list of documents (`articles`).
    - Open the file specified in the `FILE_PATH` constant.
    - Read the entire file content into a string (`fileContent`).

4. Extract Articles Using Regex
    - Use the `REUTER_PATTERN` to find all articles in `fileContent`.
    - For each match:
    - Extract the article content.
    - Create a MongoDB document from the article content using `CreateArticle.createArticleDocument`.
    - If the document is not `null`, add it to the `articles` list.

5. Insert Articles into MongoDB
    - If `articles` is not empty, insert them into the MongoDB collection specified by `COLLECTION_NAME`.

6. Close MongoDB Connection
    - Close the database connection.

Flowchart:

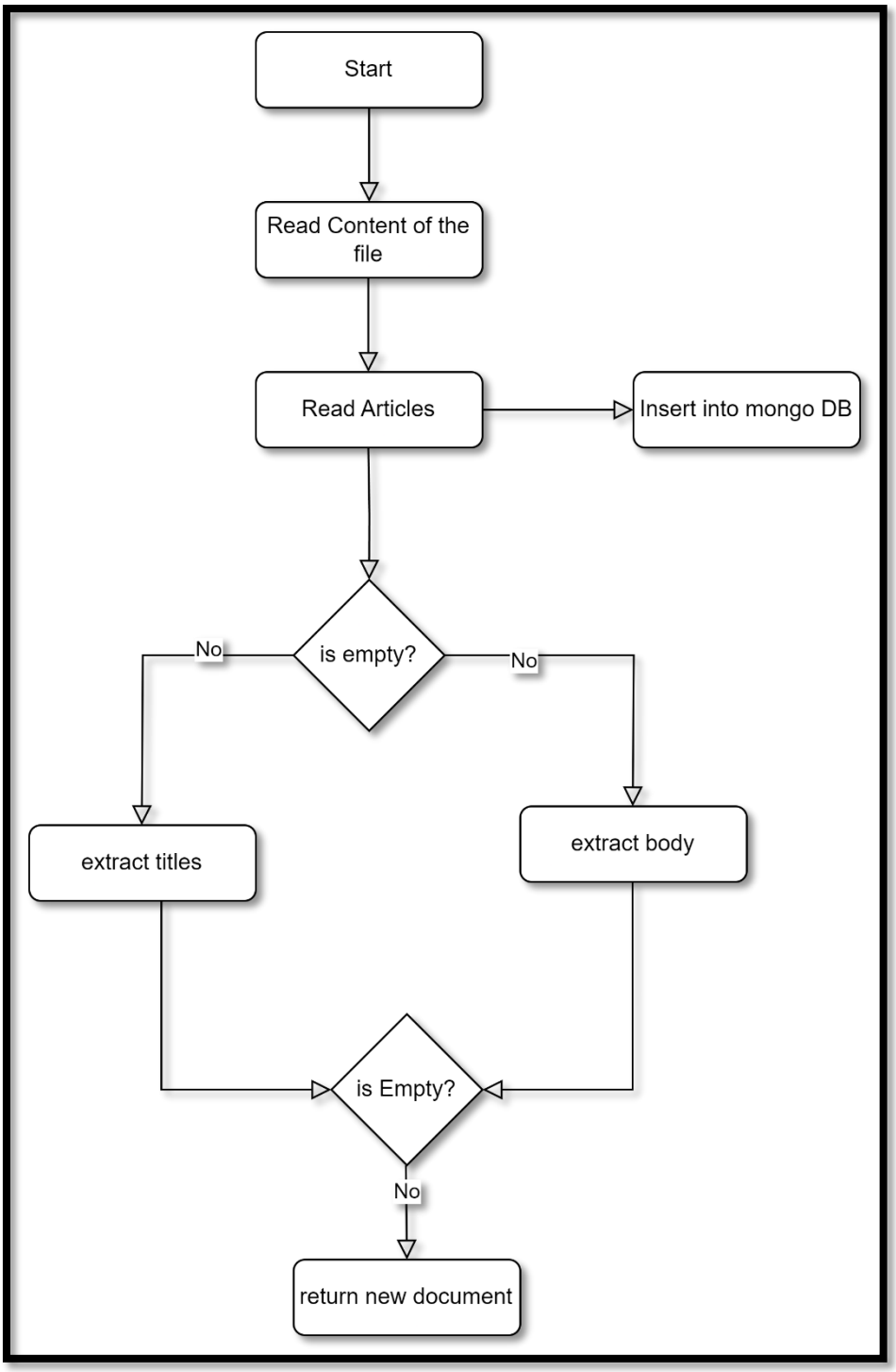

Figure 1: Flowchart of the Problem 1A

Execution:

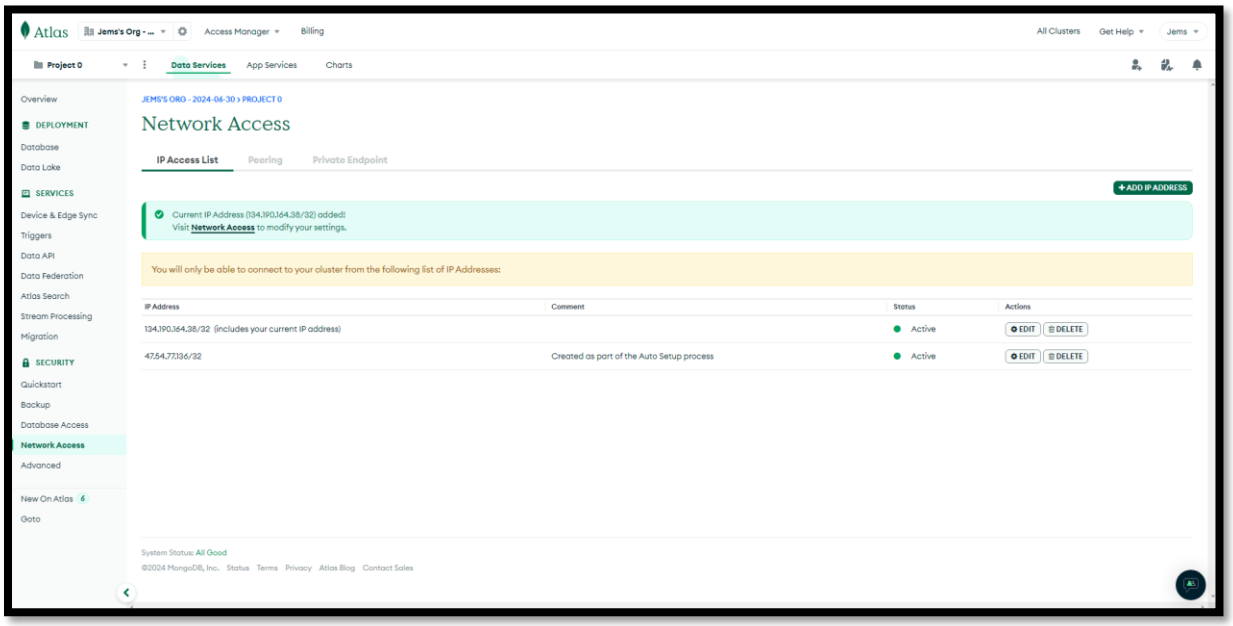

Figure 1.1.1: MongoDB Atlas

This is the dashboard for the MongoDB atlas that I have created for the lab 6 and I am using as they allow only one project to create.

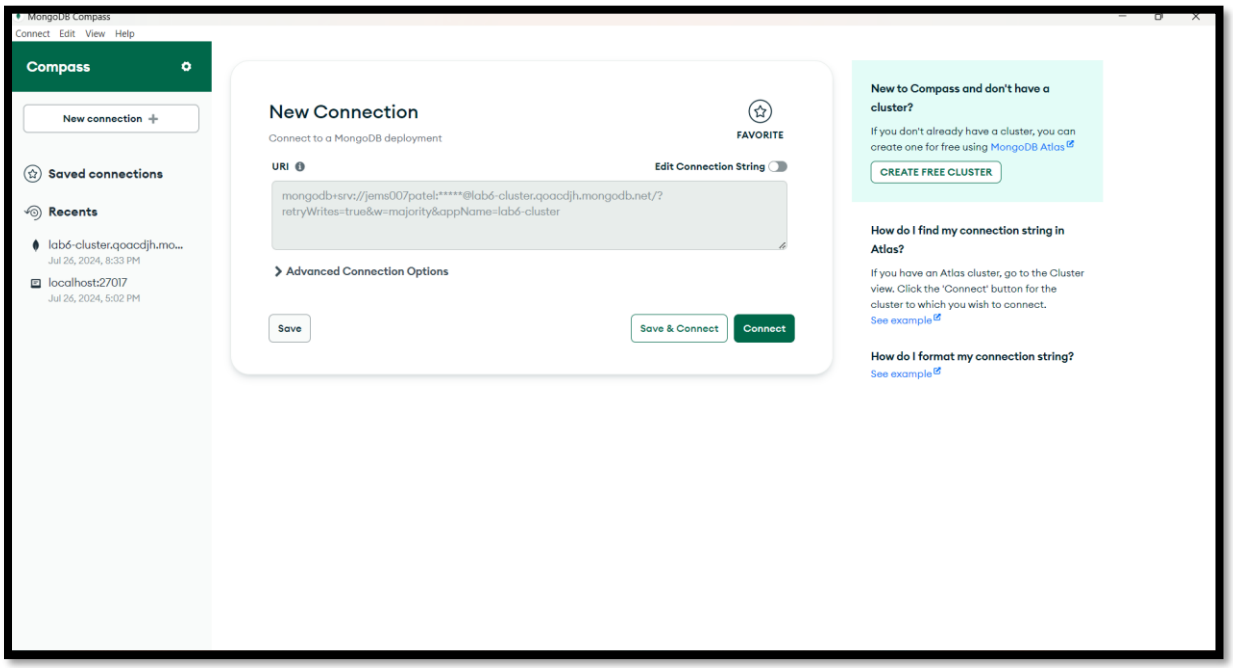

Figure 1.1.2: Connect it with the Mongo DB Compass

Now, I have initialized a connection in the MongoDB compass.

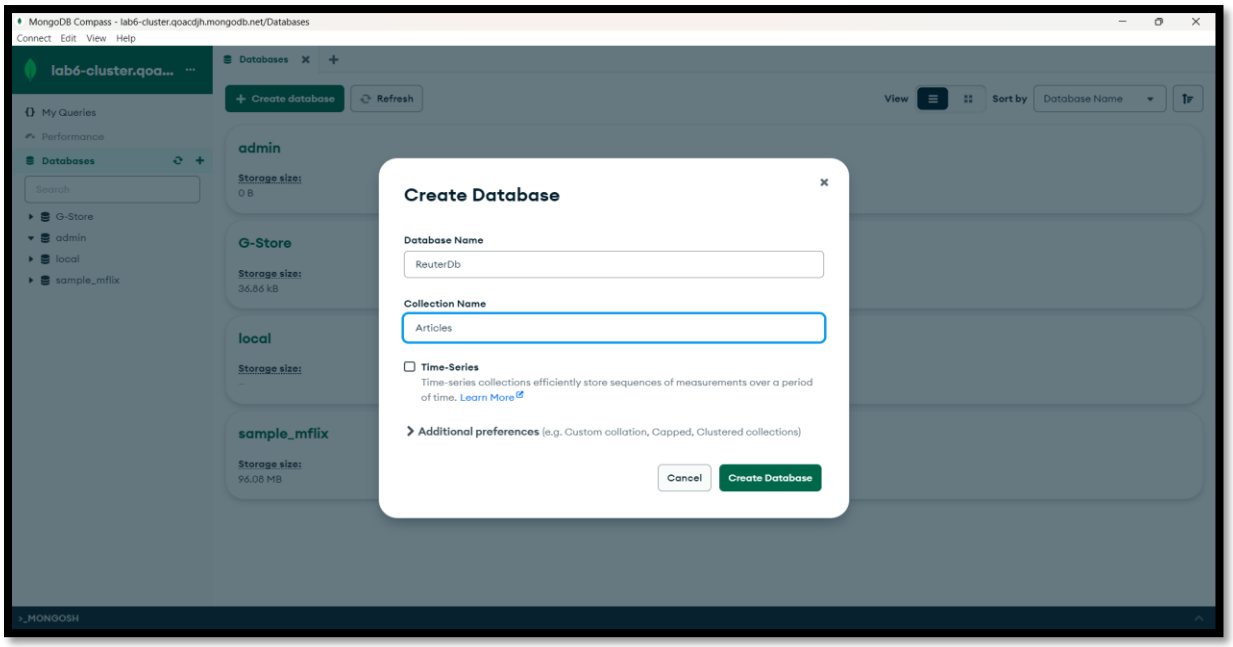

Figure 1.1.3: Create the database 'ReuterDb' and collection 'Articles'

After that I have created a database and collection as per the instruction in assignment.

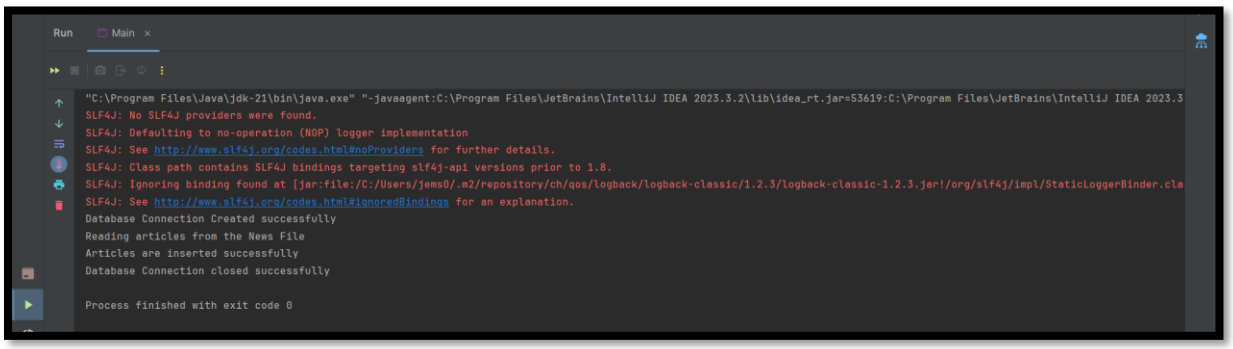

Figure 1.1.4: Program executed successfully

Now, I have executed the program which I have written in the java.

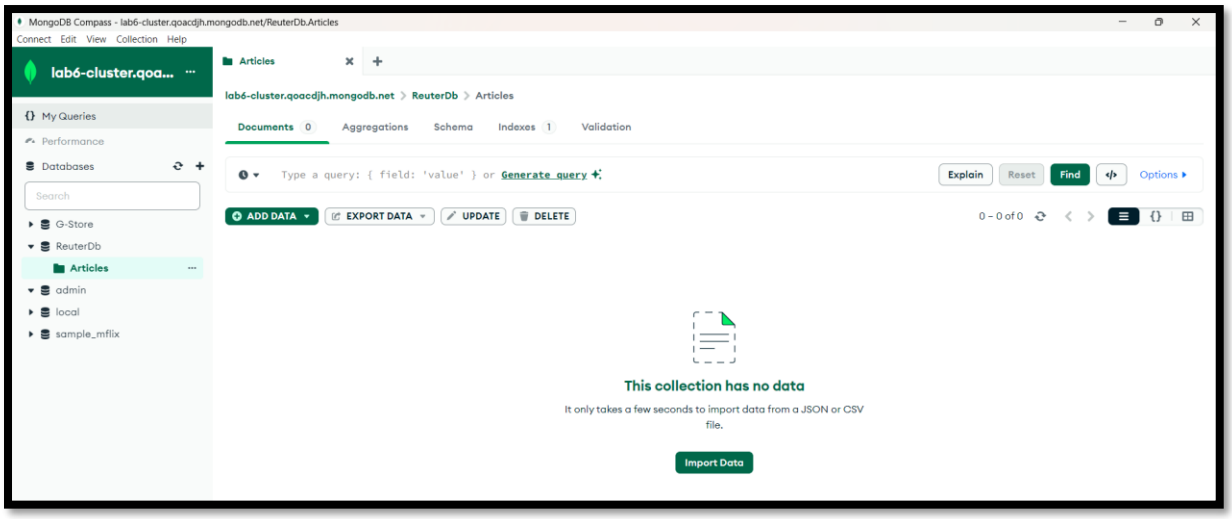

Figure 1.1.5: Articles collection before executing the program

Here, we can see that the articles had a nothing before the execution.

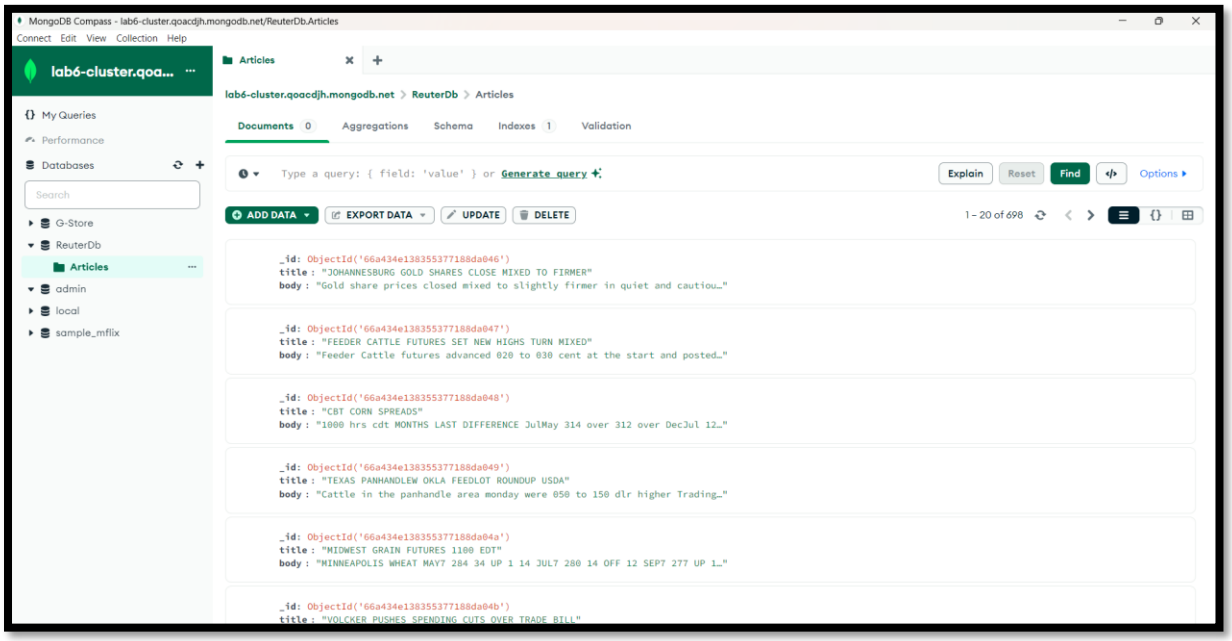

Figure 1.1.6: Articles collection after executing the program

After executing the code, the data is inserted successfully in the database.

Problem 1B: Reuter News Data Processing using Spark.

Algorithm:

1. Initialize Spark Context
    - Create a `SparkConf` object and configure it with the application name and master URL.
    - Initialize a `JavaSparkContext` with the `SparkConf` object to manage Spark operations and access resources.

2. Read and Process Input
    - Use `JavaSparkContext.textFile` to read the input file specified by `args[0]` into a `JavaRDD<String>`.
    - Clean and tokenize the text using `TextCleaner.cleanAndTokenize` to prepare it for further processing.
    - Filter out stop words from the tokenized words using `FilterStopWords.filterStopWords`.

3. Count Word Frequencies
    - Use `WordUtils.countWords` to count the occurrences of each word in the filtered `JavaRDD<String>`.
    - Convert the words into a `JavaPairRDD<String, Integer>`, where each pair represents a word and its count.

4. Analyze Word Frequencies
    - Retrieve the highest frequency words using `WordFrequencyGetter.getHighestFrequencyWords` and store them in a list.
    - Get the top N lowest frequency words and their total count using `WordFrequencyGetter.getLowestFrequencyWords`.

5. Display Results
    - Print the word(s) with the highest frequency from the list.
    - Print the words with the lowest frequency, limited to the top N, and the total count of such words.

7. Handle Exceptions and Close Context
    - Catch any exceptions during the processing and print an error message with the exception details.
    - Close the `JavaSparkContext` to release resources and clean up Spark-related operations.

Execution:

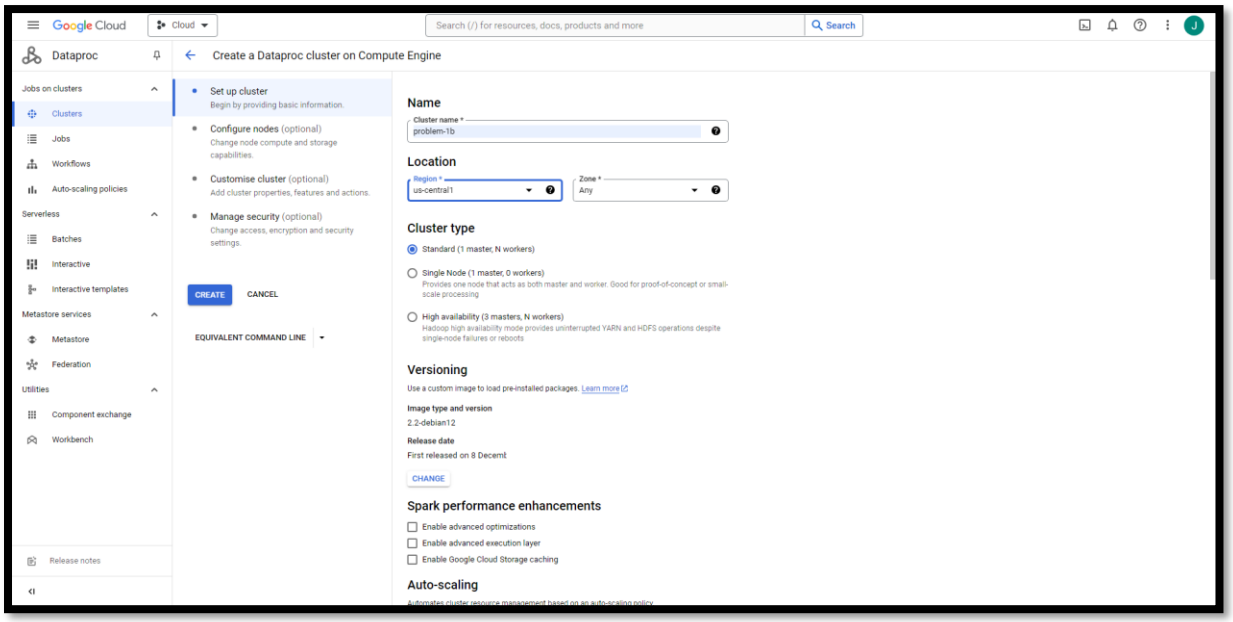

Figure 1.2.1: Setup the cluster

I have given the problem-1b as a cluster name in the dataproc.

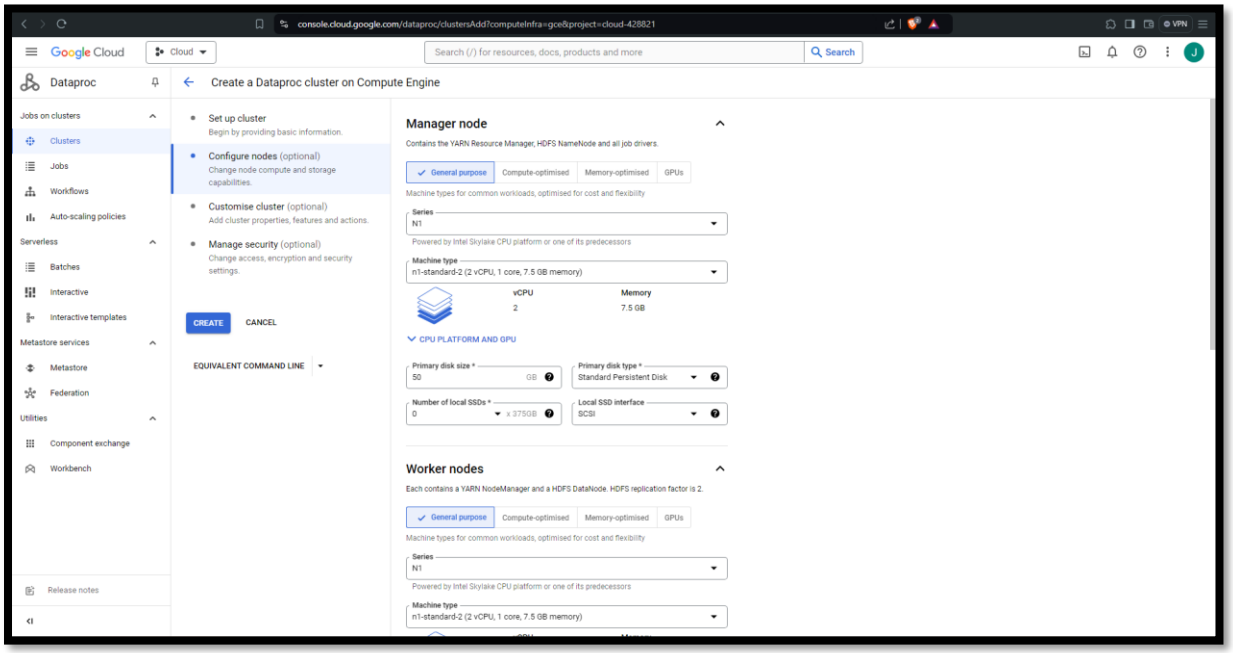

Figure 1.2.2: Configure the manager node

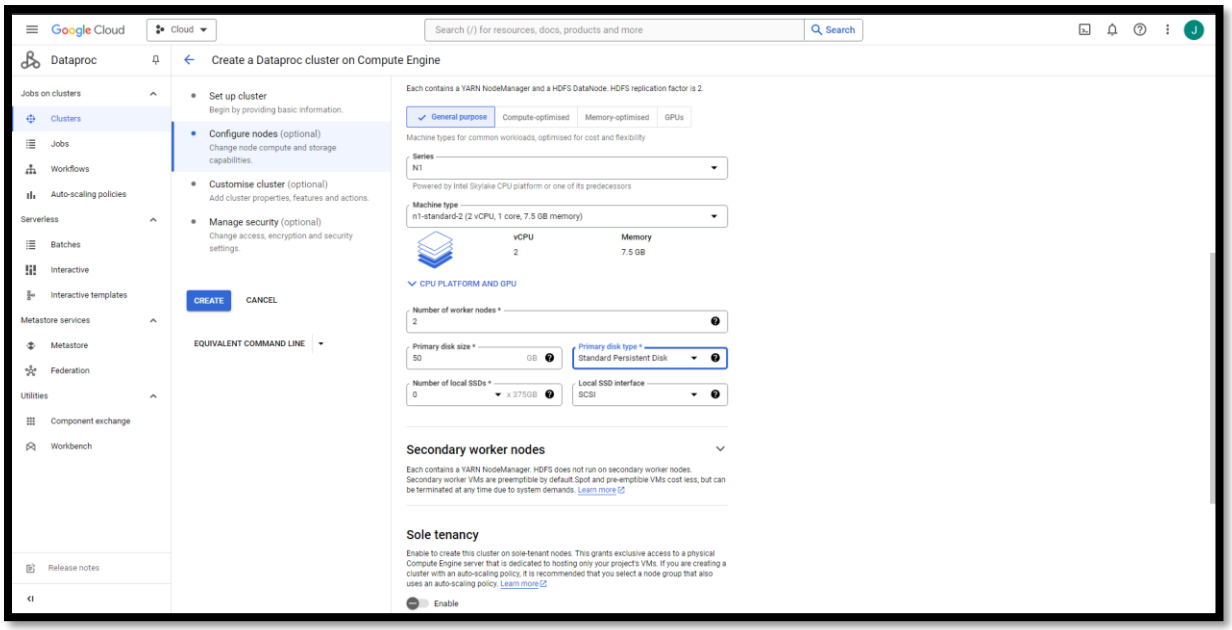

Figure 1.2.3: Configure the worker node

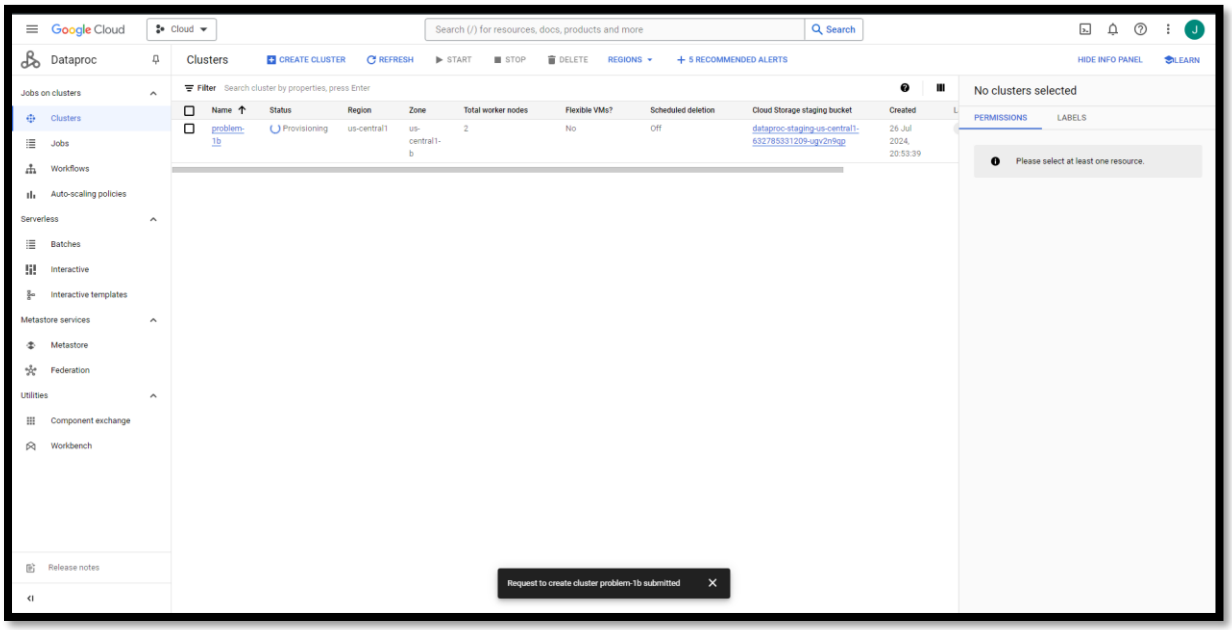

Figure 1.2.4: Creating the cluster

After creating the cluster, I connected it with the SSH.

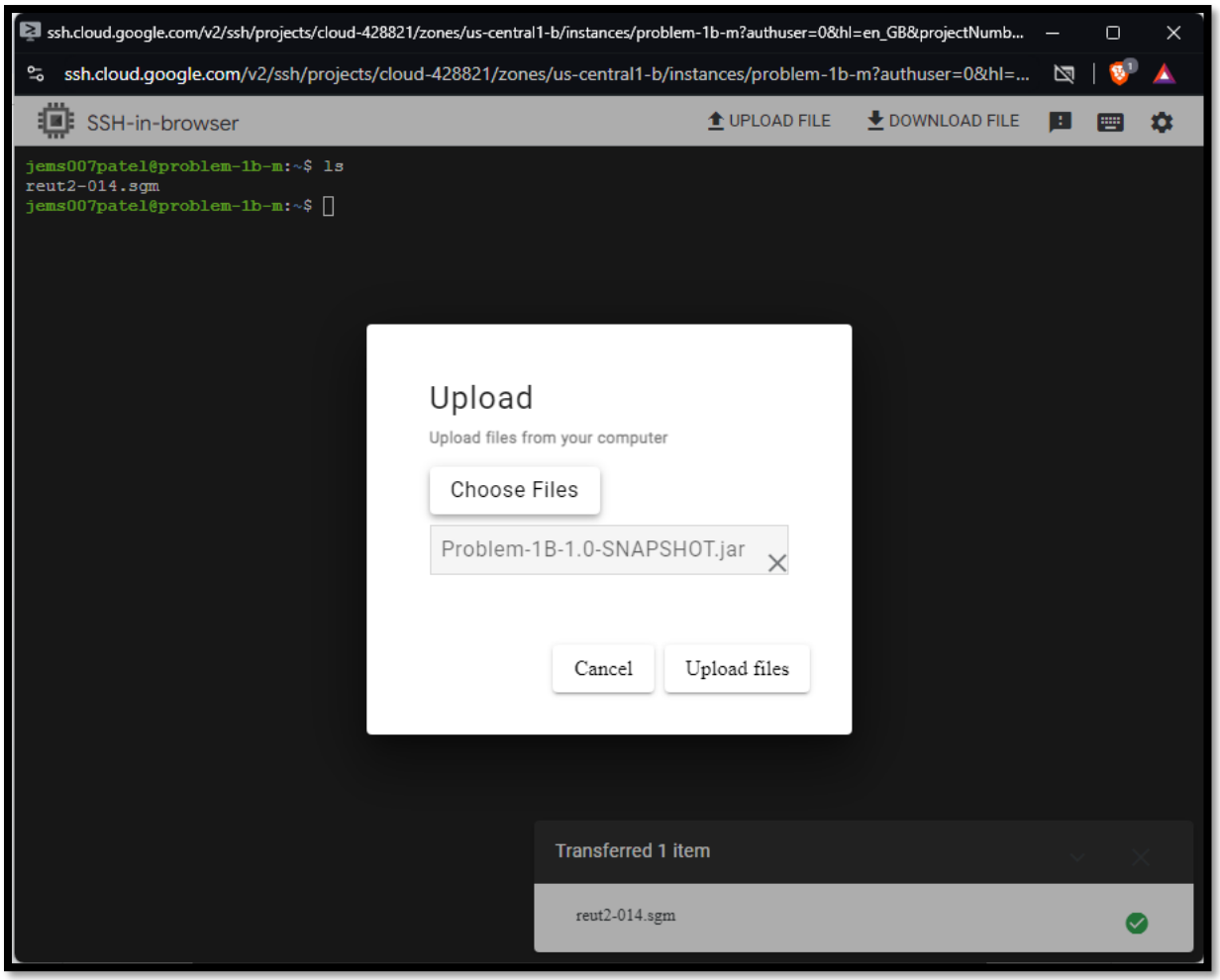

Figure 1.2.5: Upload the jar file of the java code

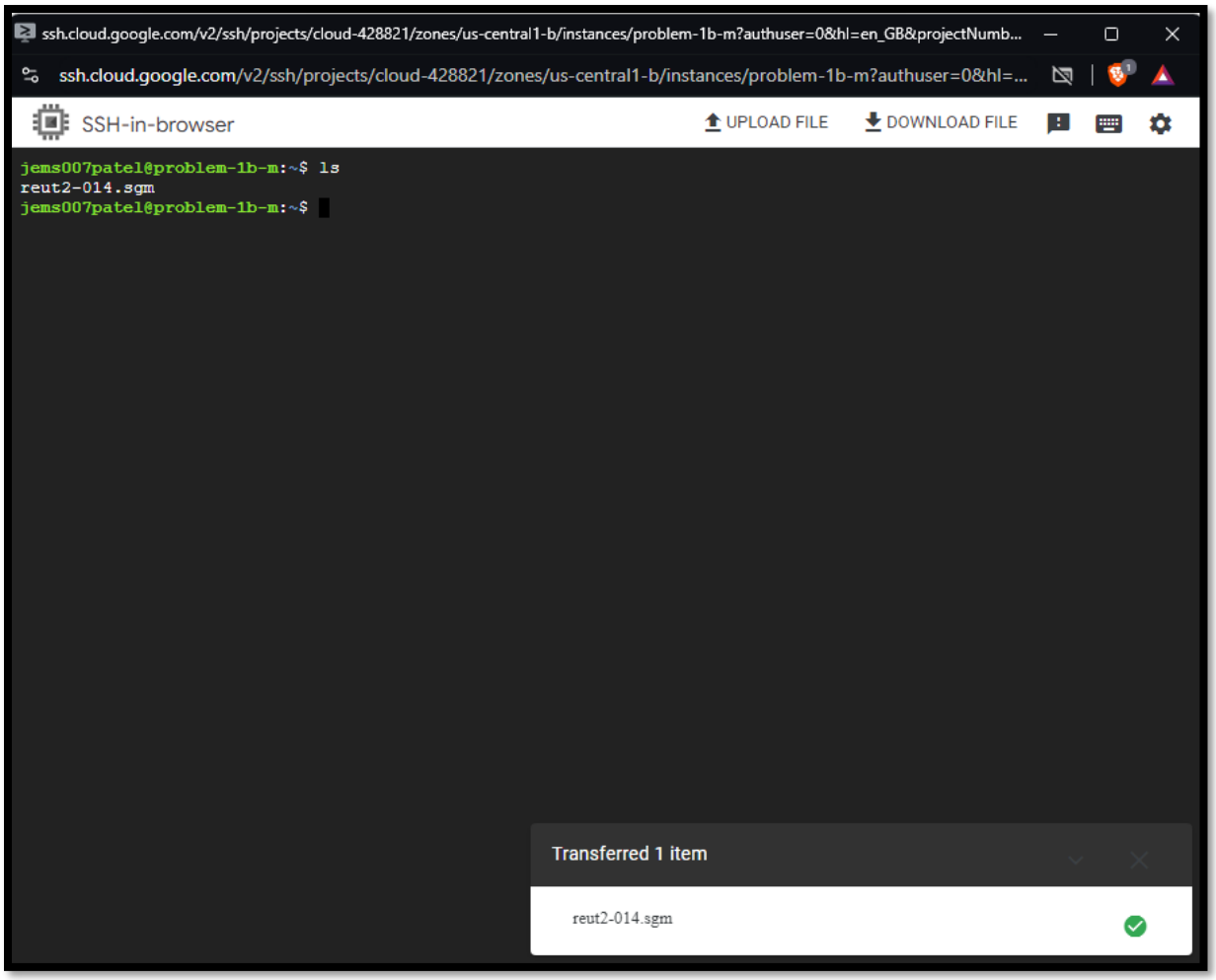

Figure 1.2.6: List files after uploading it

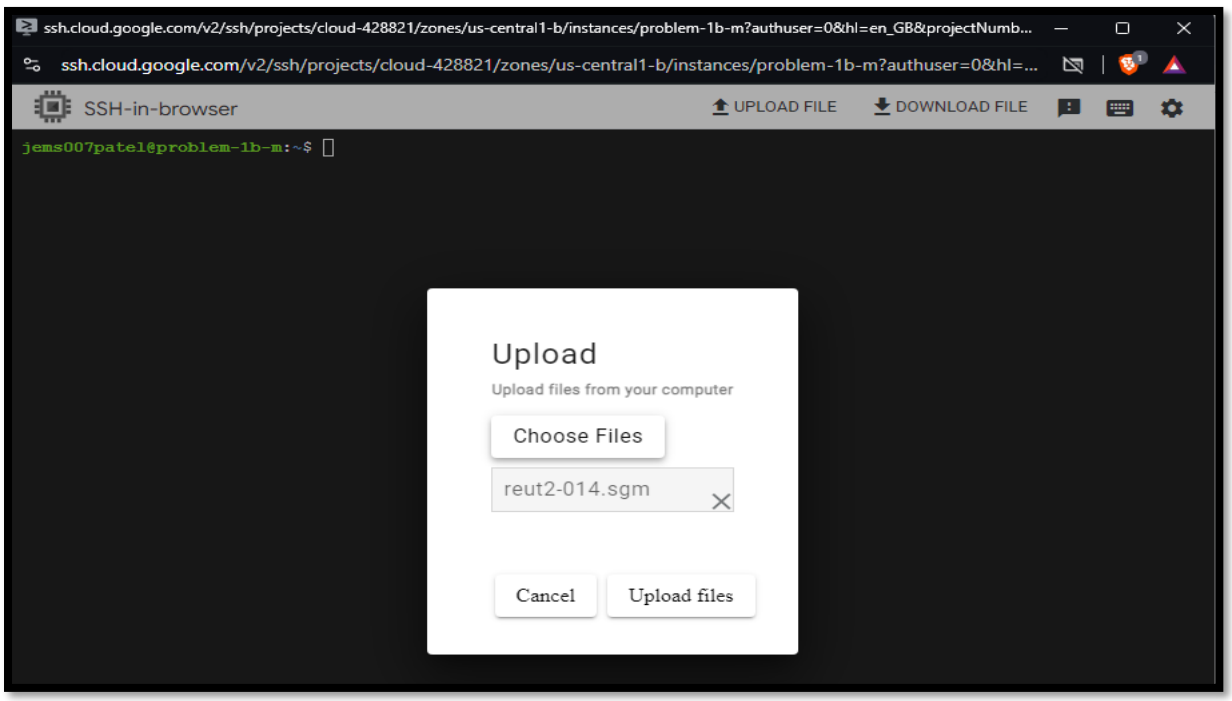

Figure 1.2.7: Upload the news file that is given in the assignment

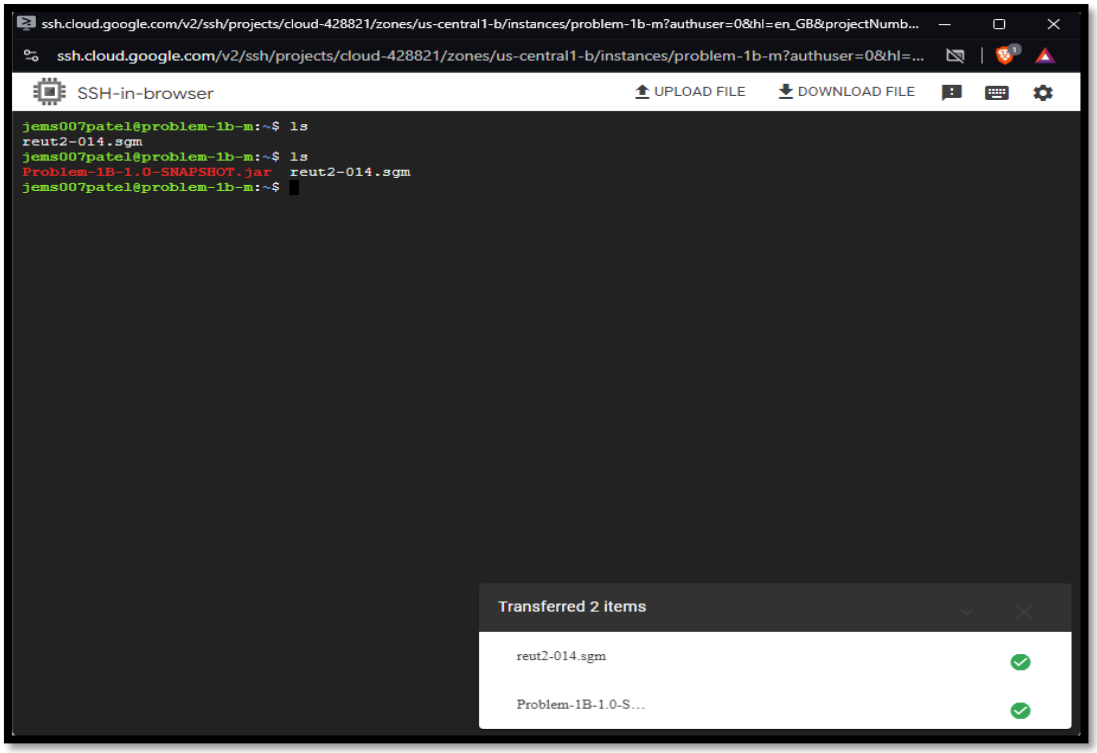

Figure 1.2.8: List all the files

After connecting it with the SSH, I uploaded two files one is jar file and another is news file.

After that, run the java code and get the highest and lowest frequency count words.

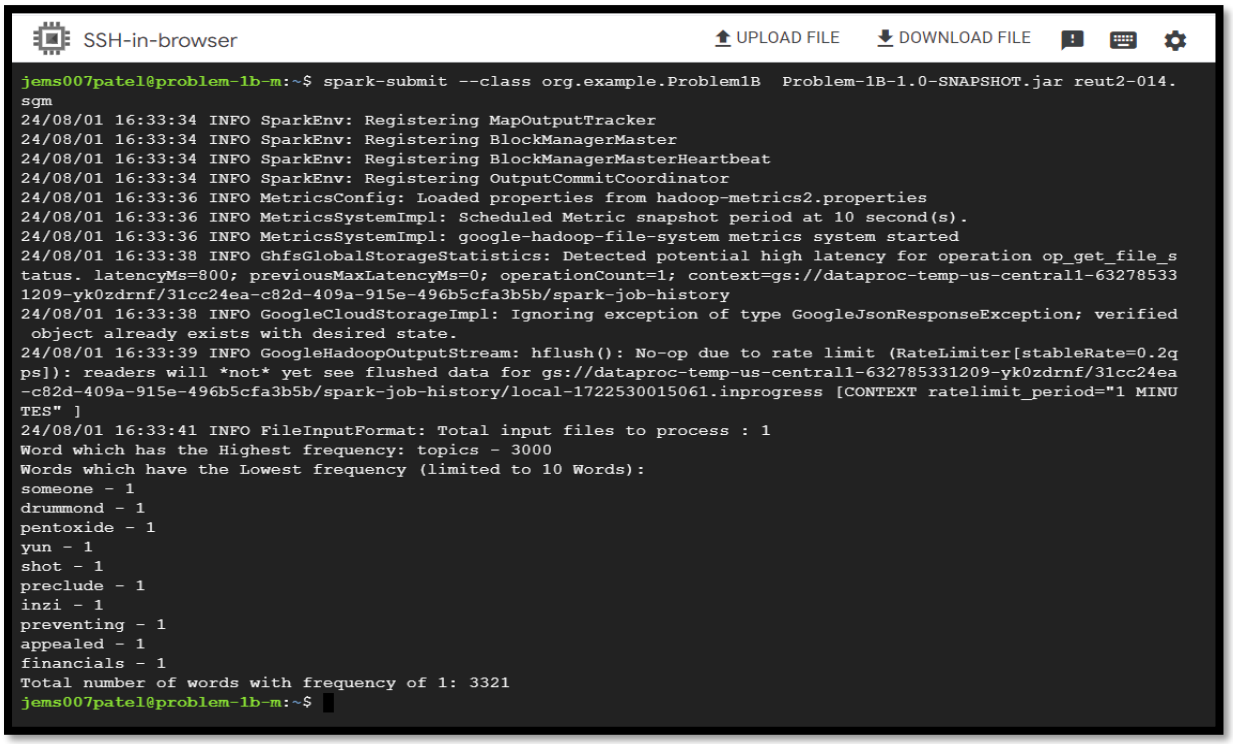

Figure 1.2.9: Frequency of some unique words along with the highest and lowest frequency

Problem 2: Building Neo4J Graph Database for relation visualization.

For creating the graph database using neo4j, I have created one free instance named “A2-Problem2” in the neo4jaura website.

Execution:

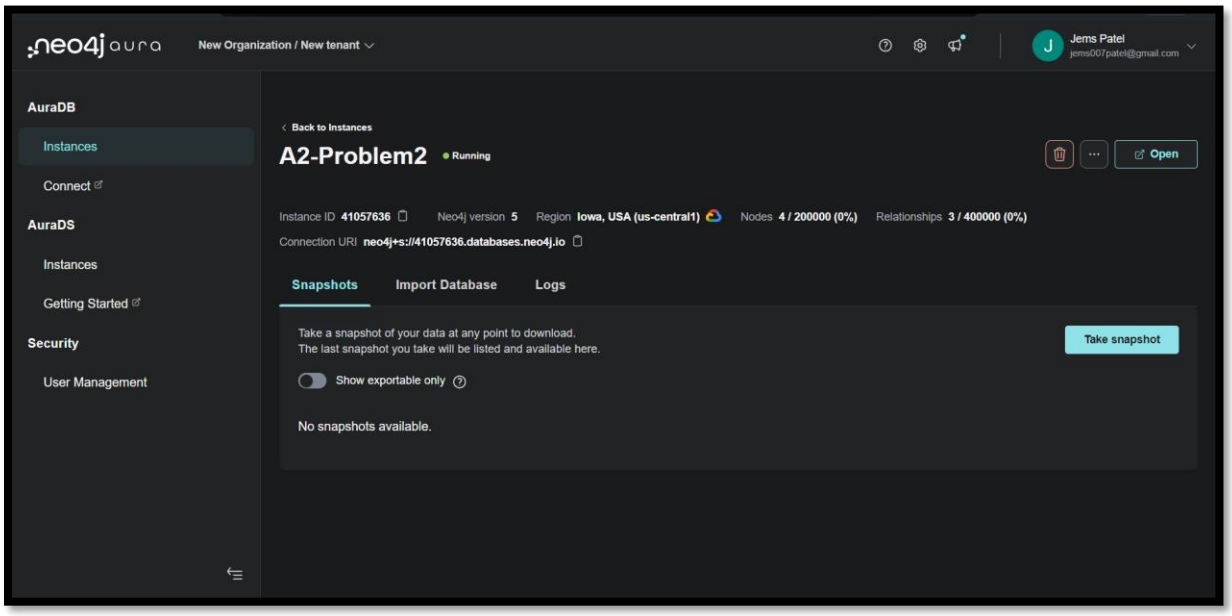

Figure 2.1: Create the instance named “A2-Problem2”

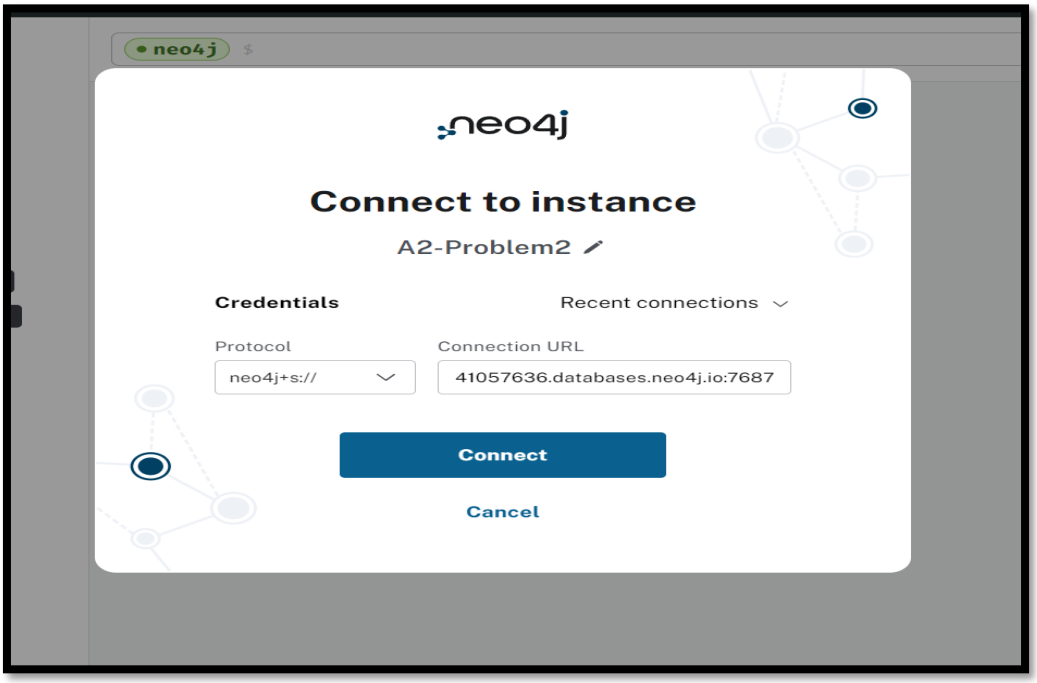

Figure 2.2: Create connection with that instance

After that, I have created a connection with that instance and opened a neo4j desktop.

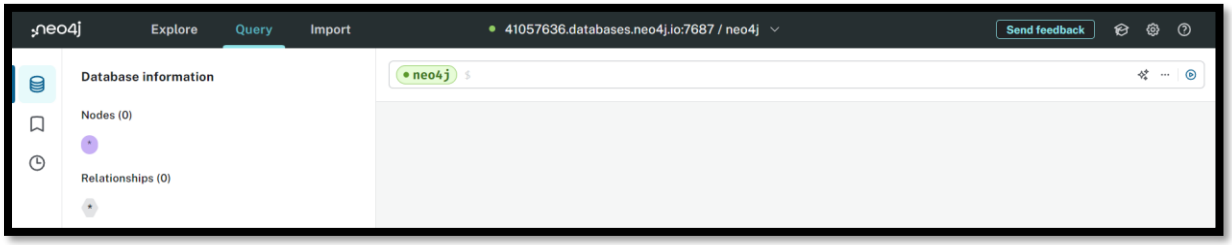

Figure 2.3: Neo4j Desktop

After, to construct the 4 nodes and relationship using the cypher query, I have taken 4 entities from mine Assignment 1.

1. Park
    - Park_ID
    - Park_Name
    - Park_Area
    - Size
    - Description
    - Park_Location
    - Opening_Hours
2. Site
    - Site_ID
    - Site_Number
    - Site_Name
    - Site_Location
    - Park_ID
    - Capacity
    - Rate
    - Allowed_Equipments
3. Reservation
    - Reservation_ID
    - User_ID
    - Site_ID
    - Duration
    - Status
    - Status (Pending or Confirmed)
4. User
    - User_ID
    - Name
    - Email
    - Password
    - Mobile_Number
    - Address
    - Reservation
    - Courses

After that, I have created 4 nodes using the above entities and provided a node detail of each node.

Figure 2.4: Cypher query to create the new node "Park"

Figure 2.5: New node "Park" created.

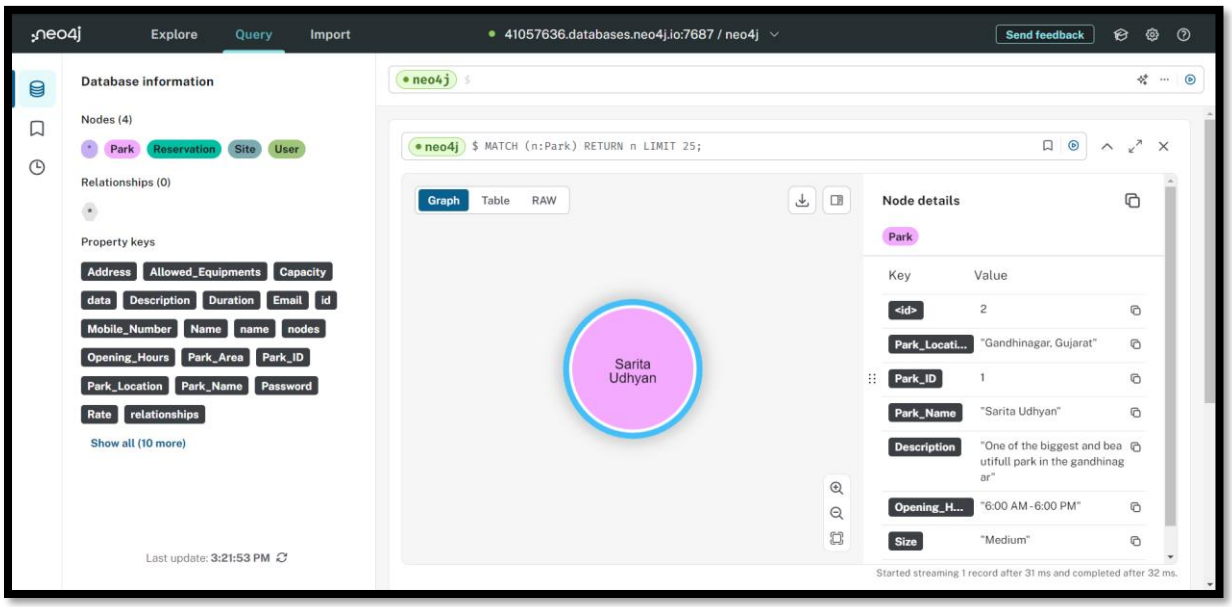

Figure 2.6: Details of the node "Park"

Figure 2.7: Cypher query to create the new node "Site"

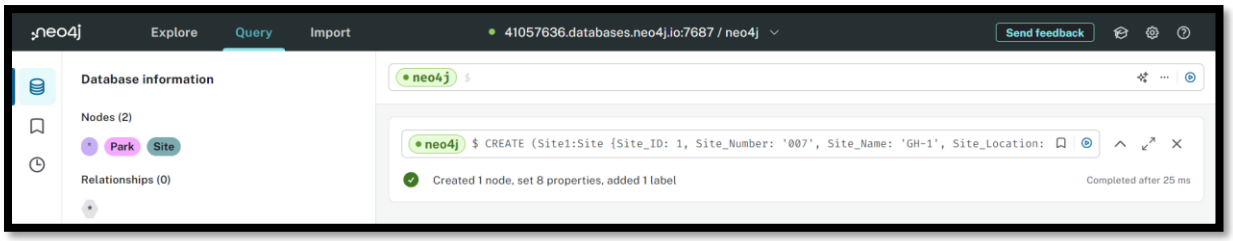

Figure 2.8: New node "Site" Created

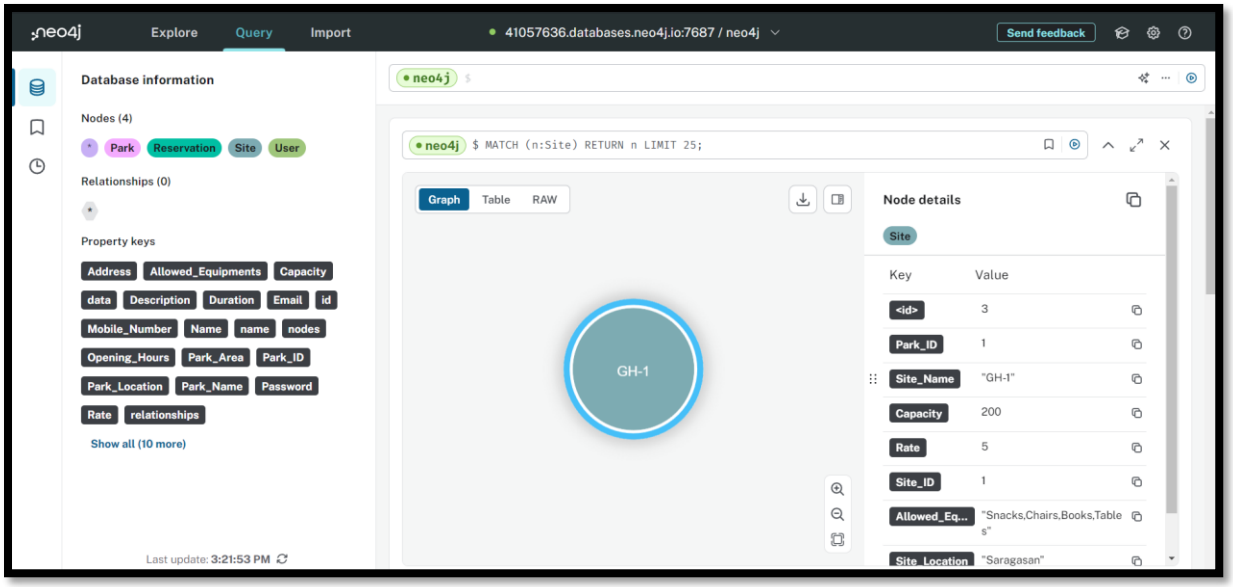

Figure 2.9: Details of the node "Site"

Figure 2.10: Cypher query to create the new node "User"

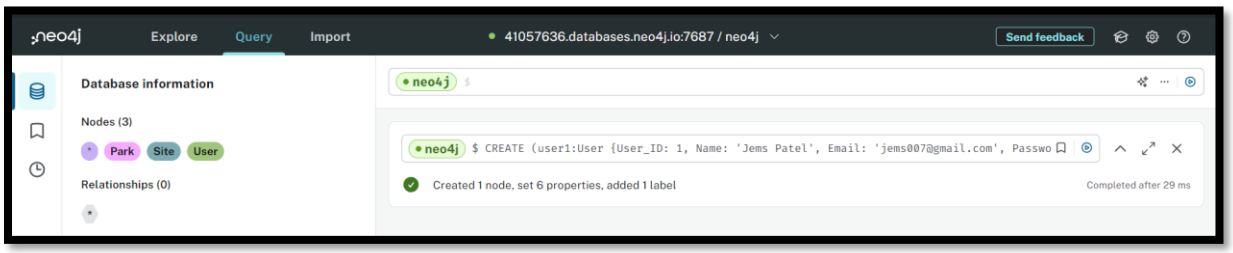

Figure 2.11: New node "User" Created

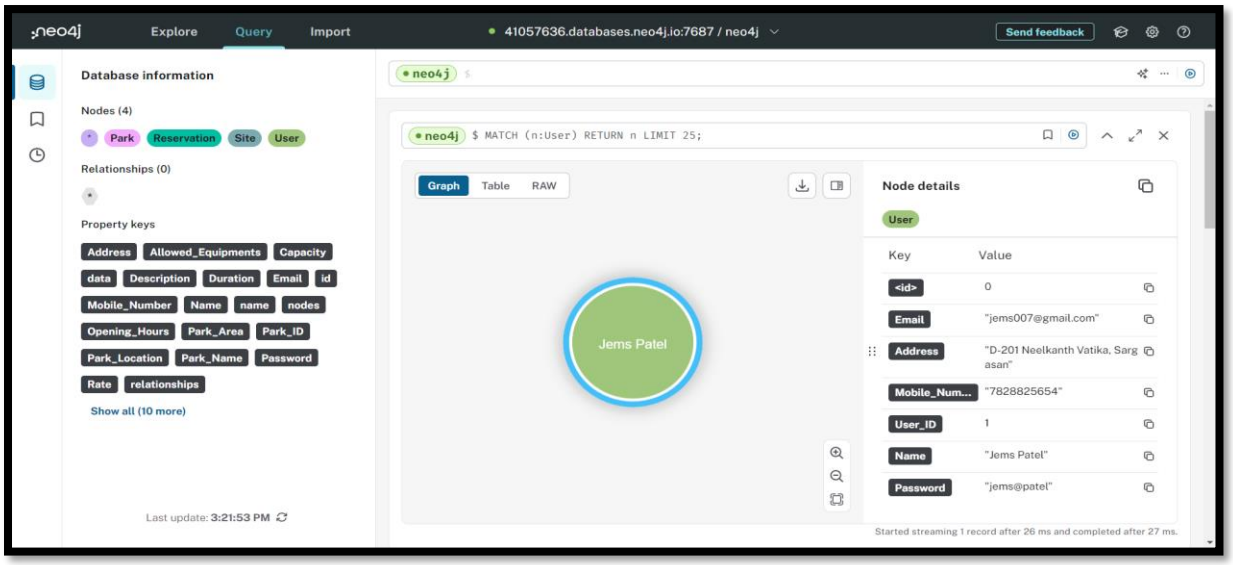

Figure 2.12: Details of the node “User”

Figure 2.13: Cypher query to create the new node “Reservation”

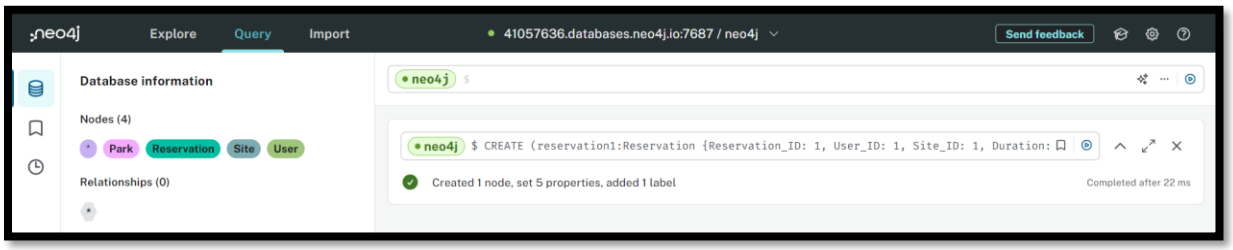

Figure 2.14: New node “Reservation” Created

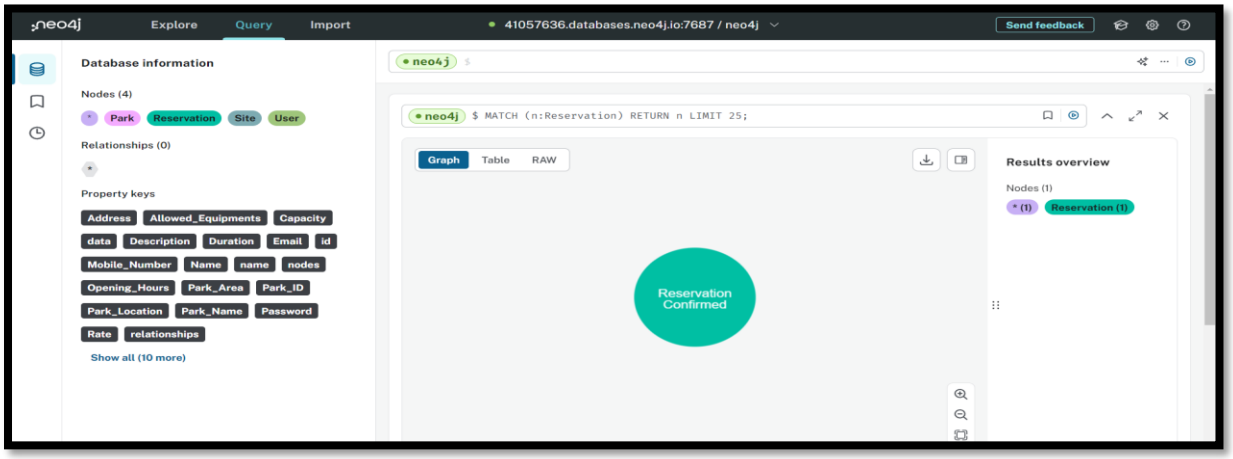

Figure 2.15: Details of the node “Reservation”

After creating the nodes, I have created edges between all the nodes using the cipher query that I created above.

Figure 2.16: Cypher query to create edge between "Park" and "Site"

Figure 2.17: edge created between "Park" and "Site"

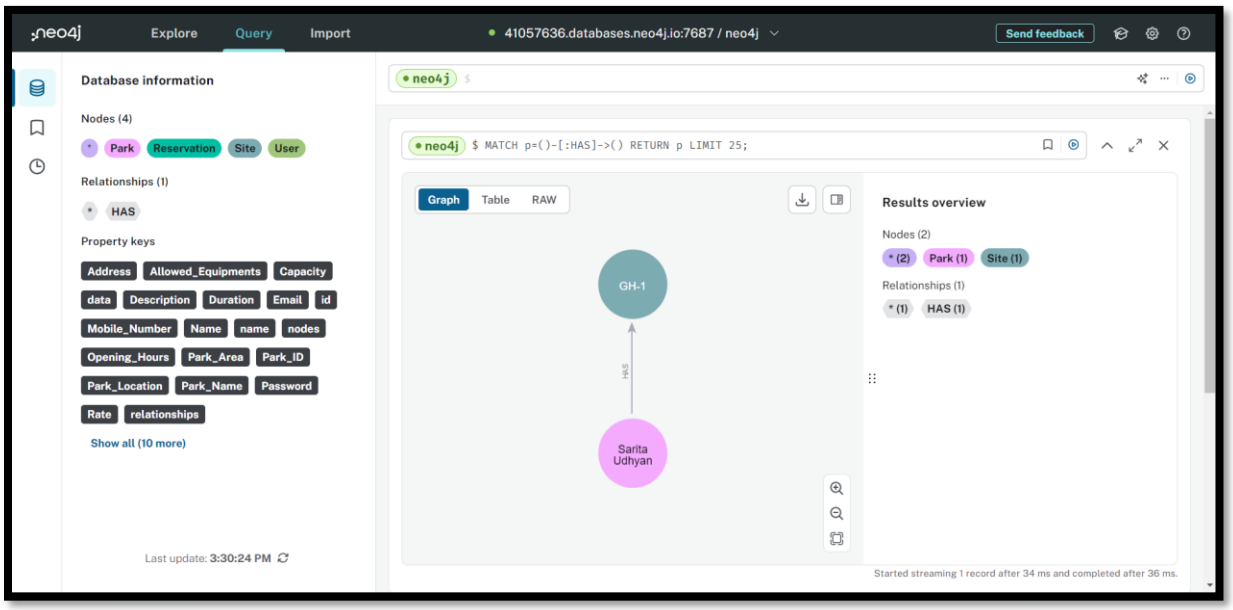

Figure 2.18: Details of the edge between the "Park" and "Site"

Figure 2.19: Cypher query to create edge between "Site" and "Reservation"

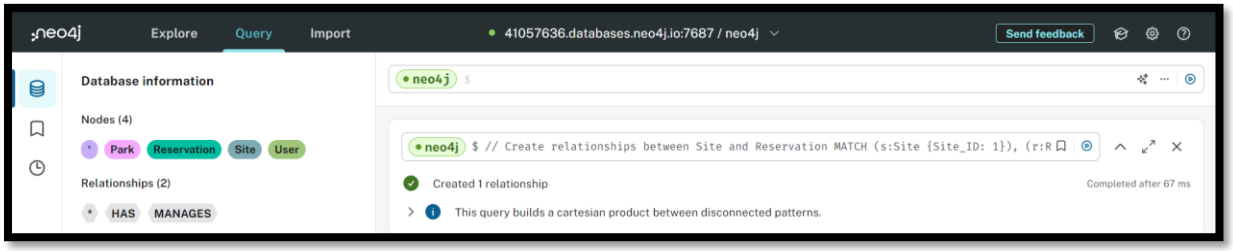

Figure 2.20: edge created between "Site" and "Reservation"

Figure 2.21: Details of the edge between the "Site" and "Reservation"

Figure 2.22: Cypher query to create edge between "User" and "Registration"

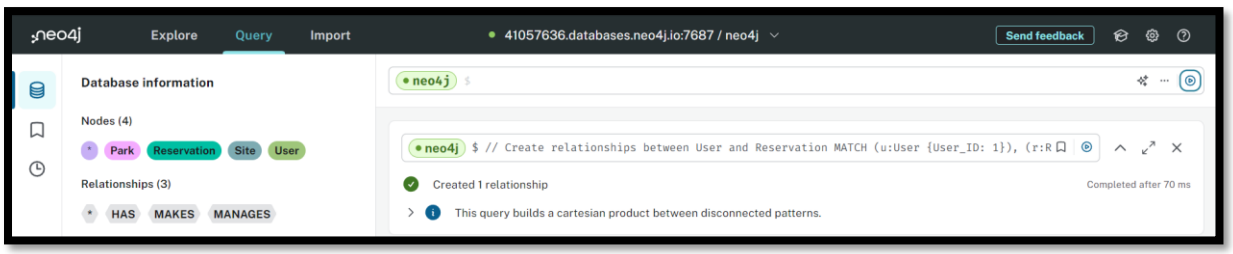

Figure 2.23: edge created between "User" and "Reservation"

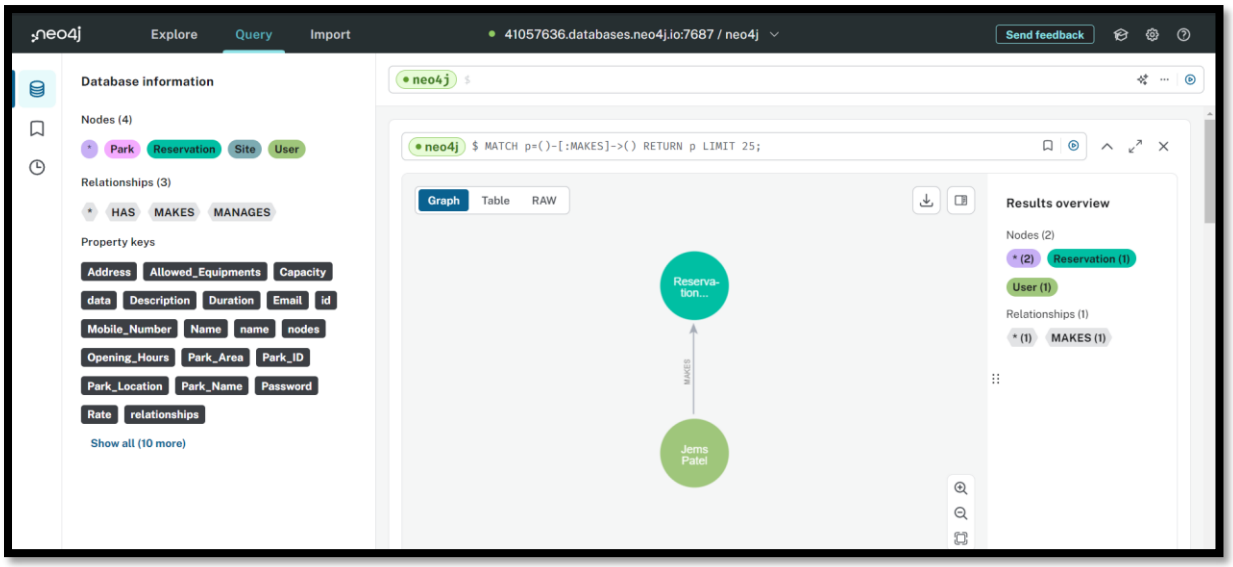

Figure 2.24: Details of the edge between the "User" and "Reservation"

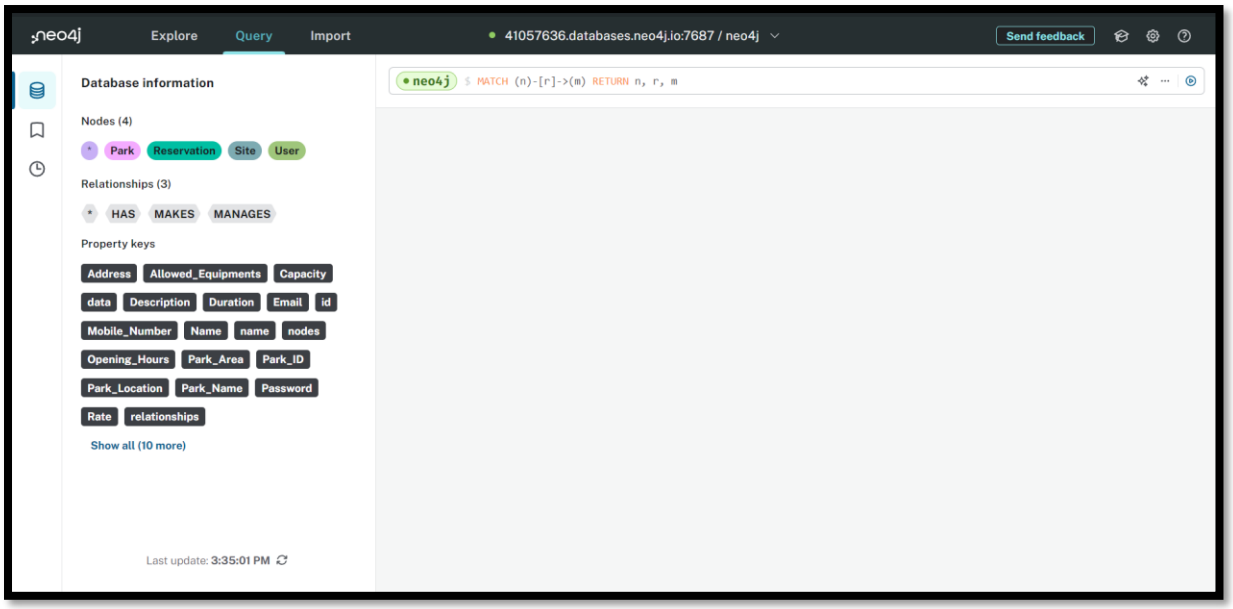

Figure 2.25: Cypher query to see all the nodes with the relationship

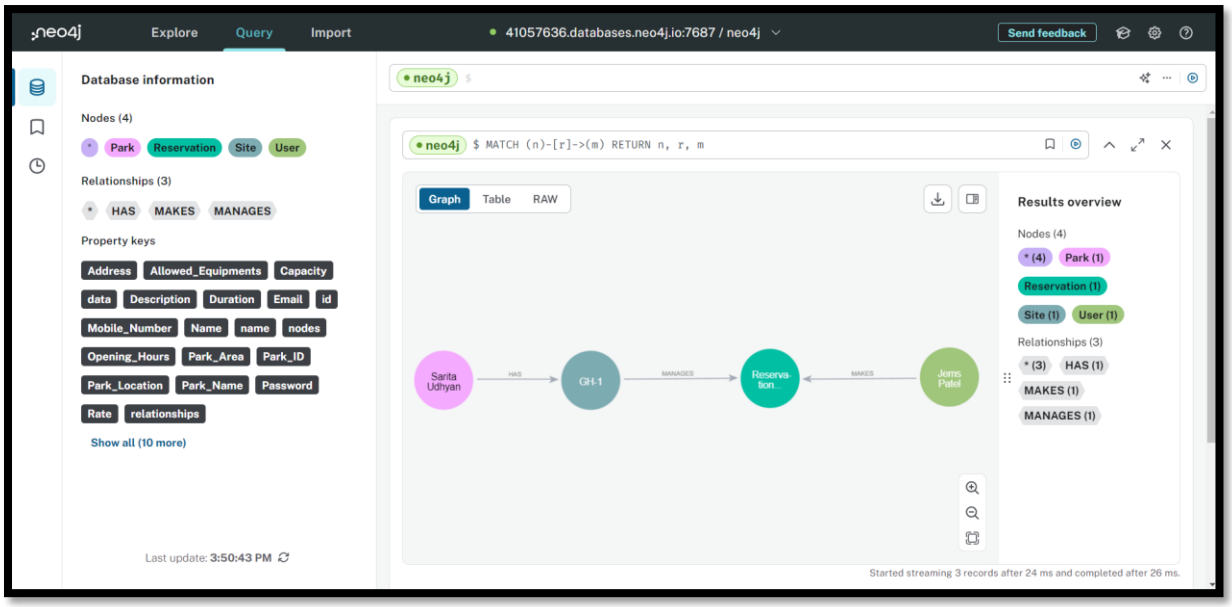

Figure 2.26: Graph created for the given nodes

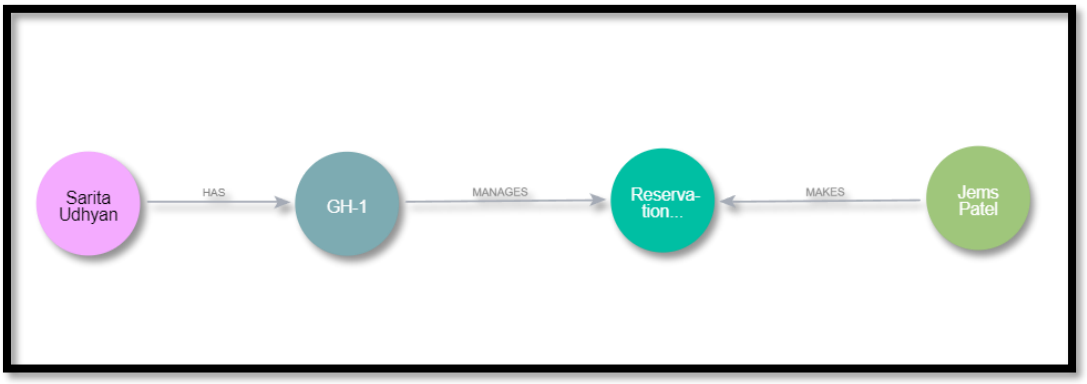

Figure 2.27: Final Graph for the nodes with the relationship

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

Algorithm:

1. Initialize Lists and Read Articles
    - Load Articles: Use `readArticles()` from the `NewsReader` class to read news articles from the specified file.
    - Load Sentiment Words: Load positive and negative words from files using `loadWordsFromFile()` from `WordUtils`.

2. Process Each Article
    - Loop Through Articles: Iterate over each article retrieved from the `readArticles()` method.
    - Extract Title: Get the title of the article.
    - Create Bag of Words: Generate a bag of words for the article title using `createBagOfWords()` from `WordUtils`.
    - Analyze Sentiment: Perform sentiment analysis on the article using `analyzeSentiment()` from `SentimentAnalysis`.

3. Store Results
    - Create SentimentResult: For each article, create a `SentimentResult` object containing the news number, title, matched words, score, and polarity.
    - Collect Results: Add each `SentimentResult` object to a list of results.

4. Write Results to CSV
    - Save to CSV: Use `writeResultsToCsv()` from `ImportCSV` to write the collected sentiment analysis results to a CSV file.

5. Extract Title and Body
    - Title Extraction: Use `extractTitle()` from `ExtractContents` to get the title from the raw text.
    - Body Extraction: Use `extractBody()` from `ExtractContents` to get the body of the article.

Execution:

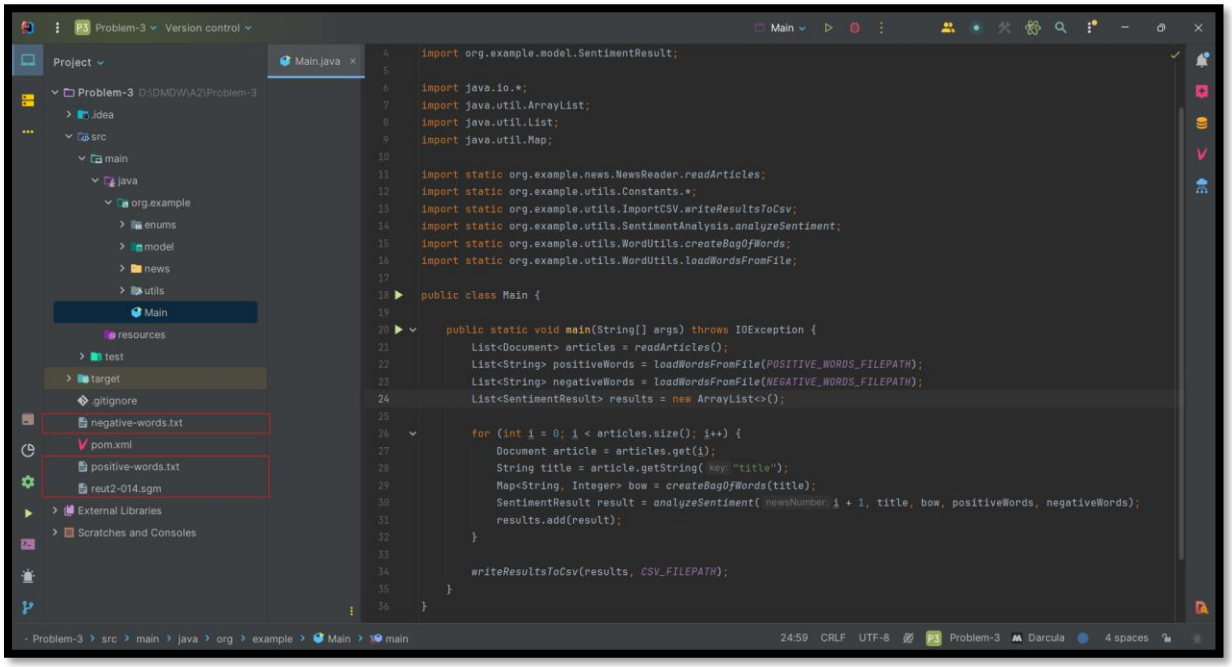

Figure 3.1: Java code before executing it

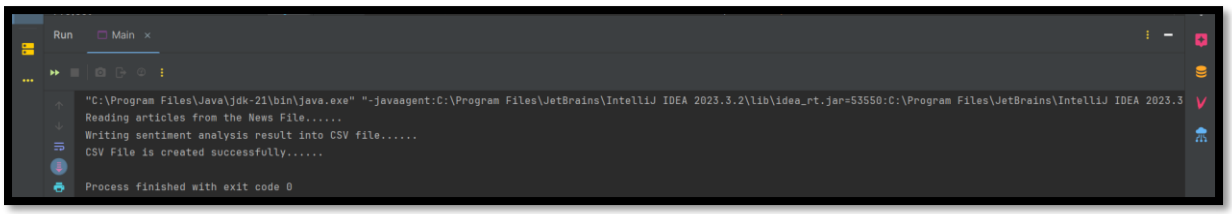

Figure 3.2: Output after executing the code

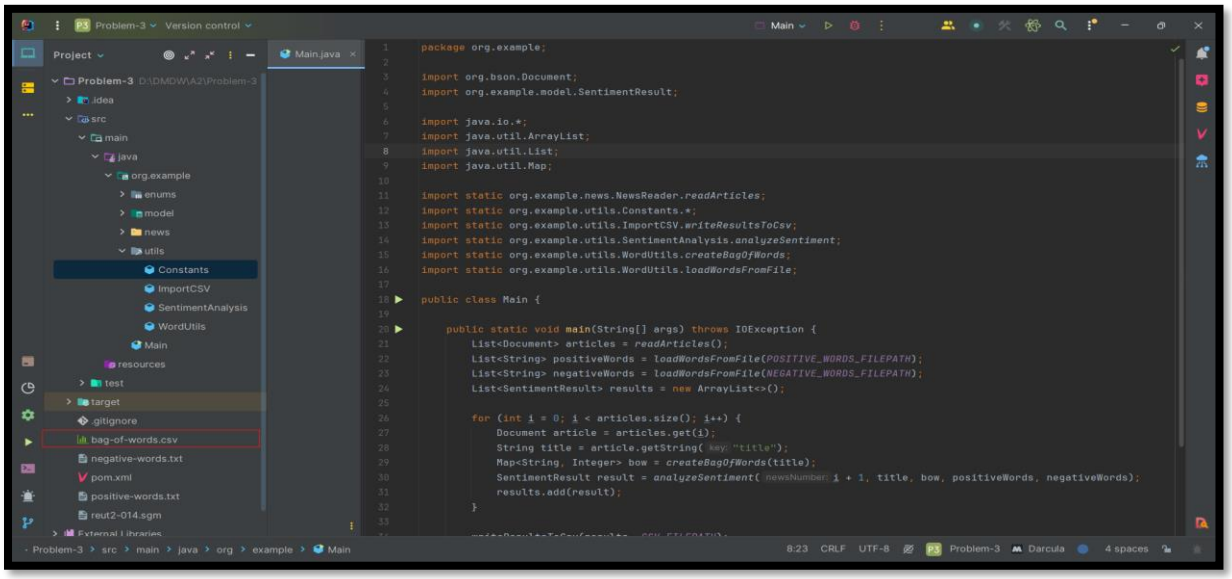

Figure 3.3: 'bag-of-words.csv' file generated after the execution

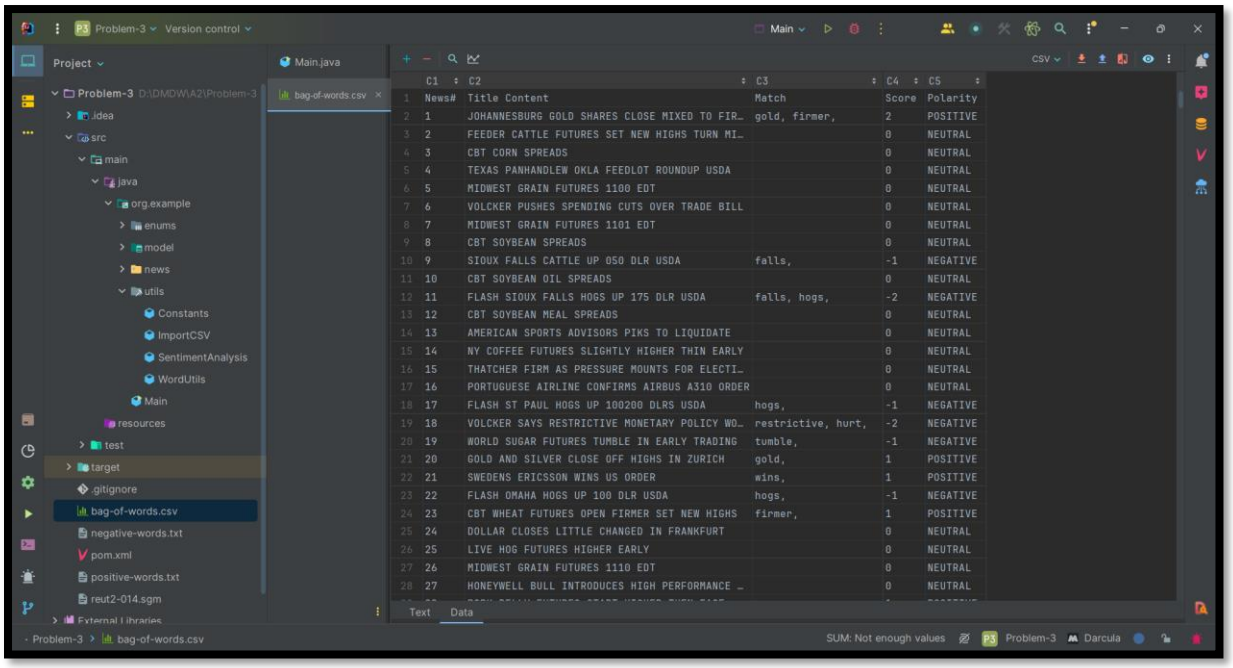

Figure 3.4: Some starting columns of the csv file

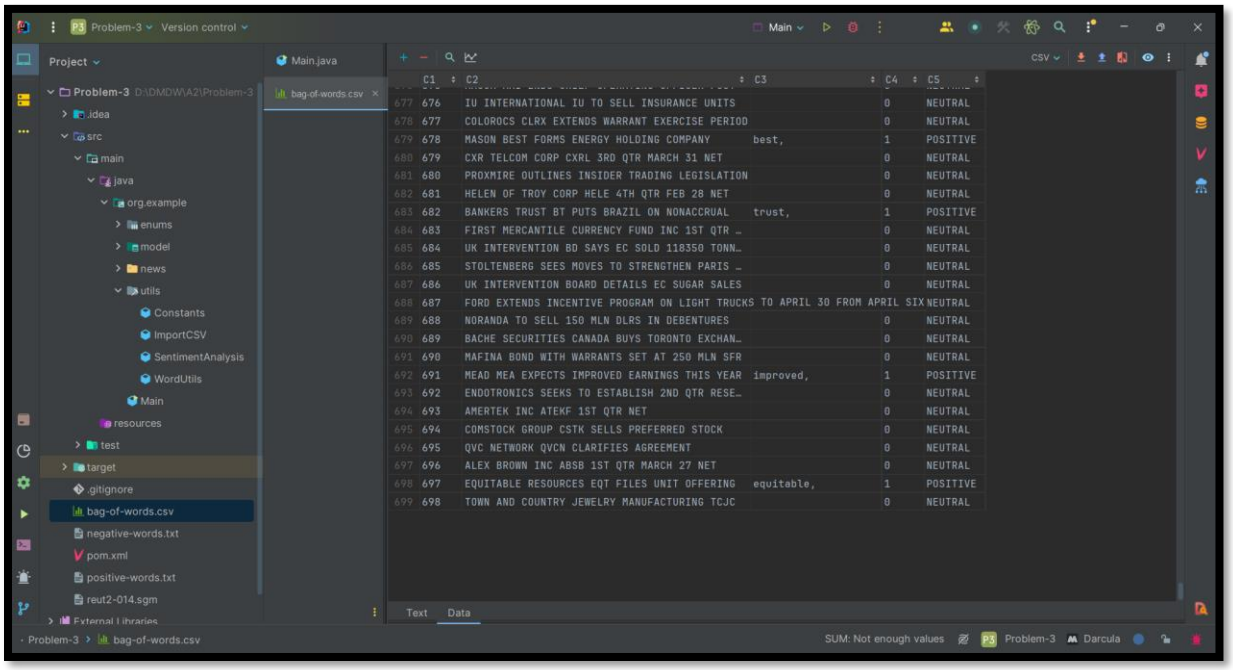

Figure 3.5: Few last columns of the csv files

References:

[1] Algs4, "Stopwords List," Princeton University, [Online]. Available: https://algs4.cs.princeton.edu/35applications/stopwords.txt. [Accessed: 25-Jul-2024].

[2] GeeksforGeeks, "What is Document Object in Java DOM?," [Online]. Available: https://www.geeksforgeeks.org/what-is-document-object-in-java-dom/. [Accessed: 26-Jul-2024].

[3] W3Schools, "Java Regular Expressions," [Online]. Available: https://www.w3schools.com/java/java_regex.asp. [Accessed: 26-Jul-2024].

[4] Neo4j, "Aura Graph Database," [Online]. Available: https://neo4j.com/cloud/platform/aura-graph-database/. [Accessed: 26-Jul-2024].

[5] Apache Spark. "JavaRDD (Apache Spark 3.4.0 API Documentation)." Available: https://spark.apache.org/docs/latest/api/java/org/apache/spark/api/java/JavaRDD.html. [Accessed: 29-Jul-2024].

[6] Apache Spark. "Java Programming Guide." Available: https://spark.apache.org/docs/0.9.0/java-programming-guide.html. [Accessed: 29-Jul-2024].

[7] Javatpoint. "Java Tuple." Available: https://www.javatpoint.com/java-tuple. [Accessed: 29-Jul-2024].