CSCI 5408: Assignment 3

Problem 1
Task1

Steps followed to configure and initialize Apache Spark Cluster on GCP [9]:

1. Create an account in GCP.
2. Then go to Compute Engine and create a new VM instance.
   (I have created a new VM instance with the name **data-apache-spark**)
   2.1. Configure the newly created VM instance
   (The machine type I selected is - e2-medium (2 vCPUs, 4 GB memory) – Ubuntu 20.04 LTS as base OS/Boot Disk)
   (External IP: **34.69.27.203**)
   2.2. Select the zone as us-central1-a.
   2.3. Under the Firewall section, we need to select the following
   (I have allowed the HTTP & HTTPS traffic in the firewall, so basically it will allow all the traffic under Firewall and also added those under the Network tags)
   2.3.1. Allow HTTP traffic
   2.3.2. Allow HTTPS traffic
   2.4. After that the VM instance will be get created.
   2.5. The instance will automatically start running.
   2.6. All the required information of the instance will be visible on the instance's banner.
   2.7. Now, I connected via browser window using SSH, there is no need to download/install SSH or any other software for tunnelling.
   2.8. Now, the VM instance's terminal opens up, displaying a message "Transferring SSH keys to the VM" and "Establishing connection to SSH server".
   2.9. Now, we open the VM instance "**data-apache-spark**".
   (VPC Network needs to be updated)
   2.9.1. VM instance opens up.
   2.9.2. Go to "Network Interfaces" → Select the "nic0" option
   2.9.2.1. The VPC Network page opens up, where we need to go to the "Firewall" option.
   2.9.2.1.1. Under the Firewall option we look for "default-allow-http" and "default-allow-https" options.
   2.9.2.1.2. Select each option and under "Protocols and port" select "Allow all" for both firewall options of HTTP and HTTPS rules.
   2.10. Now, to install and run Apache Spark, we need Java, Scala and GIT.
   (So, we follow the below steps to install the Java, Scala and GIT on the VM)
   (First, we go the already open terminal in browser)
   2.10.1. Run the following command to update
   sudo apt-get update
   2.10.2. Run the following command to install jdk, scala and git
   sudo apt install default-jdk scala git -y

(The above command will install the default compatible version of Java, install Scala, and install GIT → the "-y" command installs everything in the working directory)

2.10.3. After successful installation, we run the below command to check for proper installation and also check the version.

java -version && scala -version && git –version

2.11. Now we need to download the Apache Spark

2.11.1. Creating a directory named Apache_Spark where we will download the apache spark.

mkdir Apache_Spark

2.11.2. Go to the created directory

cd Apache_Spark

2.11.3. Now, we download the Apache Spark from the following link

https://downloads.apache.org/spark/

(The above link has various versions of Apache Spark that we can download)

(Selecting the latest version **spark-3.1.2**)

2.11.4. Using the below command to download Apache Spark from the below link

https://downloads.apache.org/spark/spark-3.1.2/spark-3.1.2-bin-hadoop3.2.tgz

wget https://downloads.apache.org/spark/spark-3.1.2/spark-3.1.2-bin-hadoop3.2.tgz

2.11.5. The above command will download the **spark-3.1.2-bin-hadoop3.2.tgz** file in the Apache_Spark directory.

2.11.6. Once the download is complete, we extract the .tgz file using the below command

tar xvf spark-3.1.2-bin-hadoop3.2.tgz

2.11.7. We can run the below command to verify whether the file have been successfully extracted (under the Apache_Spark directory)

ls -all

(This command lists all the directories or files inside the current directory)

2.11.8. Now we move the extracted **spark-3.1.2-bin-hadoop3.2** to **/opt/spark** directory

sudo mv spark-3.1.2-bin-hadoop3.2/ /opt/spark

2.11.9. Now we go to the **/opt/spark** directory to check whether the file has been successfully moved

cd /
cd /opt/spark
ls -all

2.12. Now we need to setup the Spark Environment by configuring the paths.

2.12.1. We execute the following commands to configure Apache Spark's permanent path (a much-recommended practice)

Opening the .profile file
sudo nano ~/.profile

Adding the below lines in the .profile file
export SPARK_HOME=/opt/spark
export PATH="$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin"
export PYSPARK_PYTHON=/usr/bin/python3
Saving the file and closing it

To activate the changes made we run the below command

source ~/.profile

2.13. Now we have successfully installed Apache Spark and also set the permanent environment and configured the path.

2.14. Now we can verify whether Apache Spark has been installed correctly or not by starting the standalone master server by running the below command
start-master.sh
(we will get the following message)
starting org.apache.spark.deploy.master.Master
(the logging also starts and is done to the .out file)
/opt/spark/logs/spark-shahdhrumil1060-org.apache.spark.deploy.master.Master-1-data-apache-spark.out

2.15. If we want to check the logs, we can use the following commands
tail /opt/spark/logs/spark-shahdhrumil1060-org.apache.spark.deploy.master.Master-1-data-apache-spark.out

tail -f /opt/spark/logs/spark-shahdhrumil1060-org.apache.spark.deploy.master.Master-1-data-apache-spark.out

2.16. Now as we have started the master server, we can verify it by opening the following link in the browser
34.69.27.203:8080
(External IP:8080 → the port on which the server is running)

2.17. Now to start the Slave worker node we need to run the following command
start-worker.sh spark://data-apache-spark.us-central1-a.c.csci-5408-s21-317315.internal:7077
(appending the Master nodes internal URL, and using start-worker as start-slave is deprecated)

2.18. We can again check the logs by running below command
tail /opt/spark/logs/spark-shahdhrumil1060-org.apache.spark.deploy.master.Master-1-data-apache-spark.out
(We can verify the worker node)

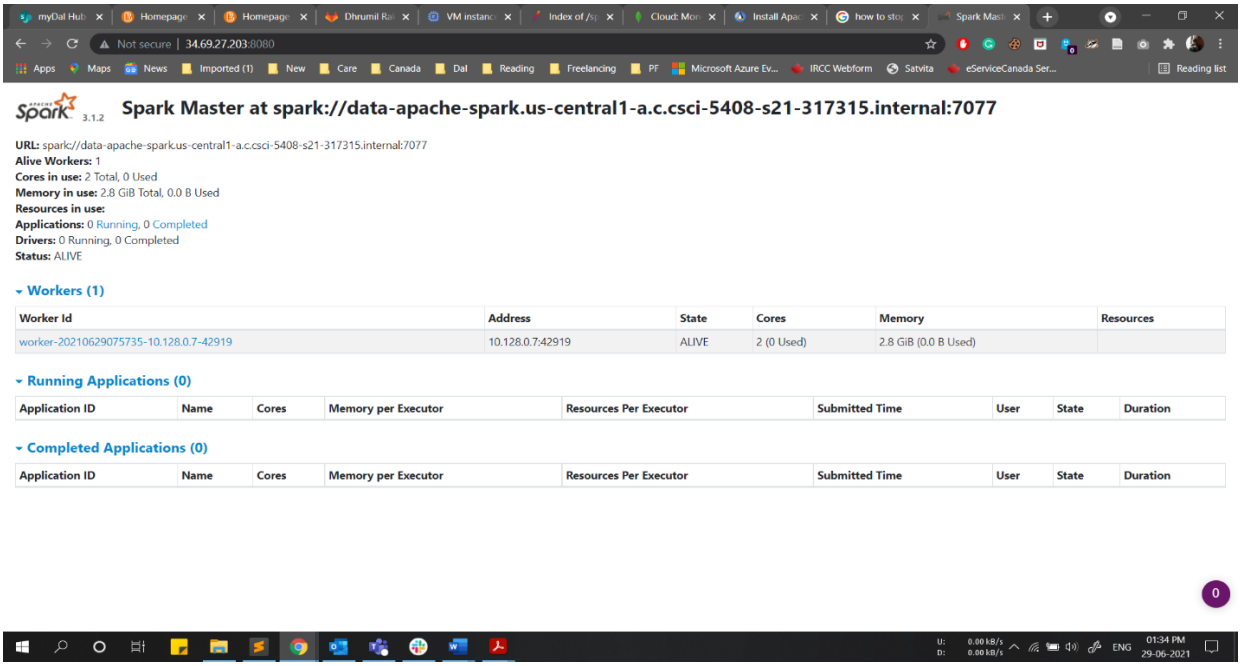

Figure 1. Screen Shot of the Master and Worker Nodes, created using GCP [9].

2.19. If we wish to stop master node or all nodes, we can use the following commands

stop-master.sh

stop-all.sh

2.20. Using speak shell

spark-shell

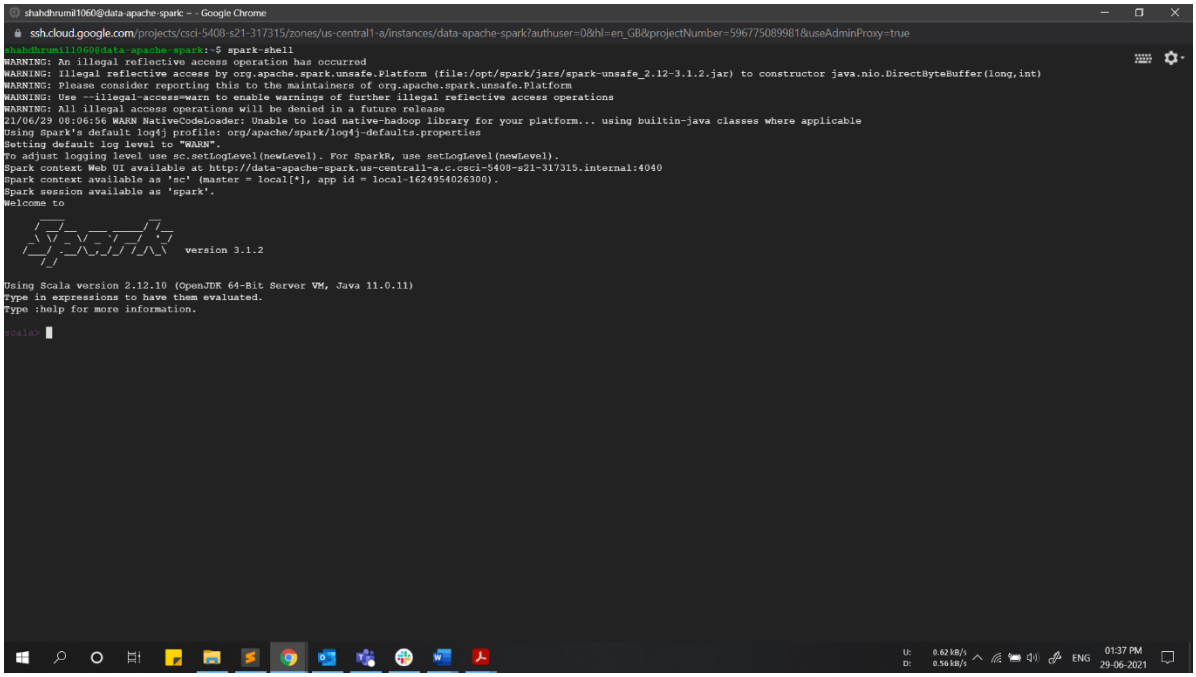

Figure 2. Starting Spark using spark-shell command, created using GCP [9].

2.21. Or we can use pyspark as well

Pyspark

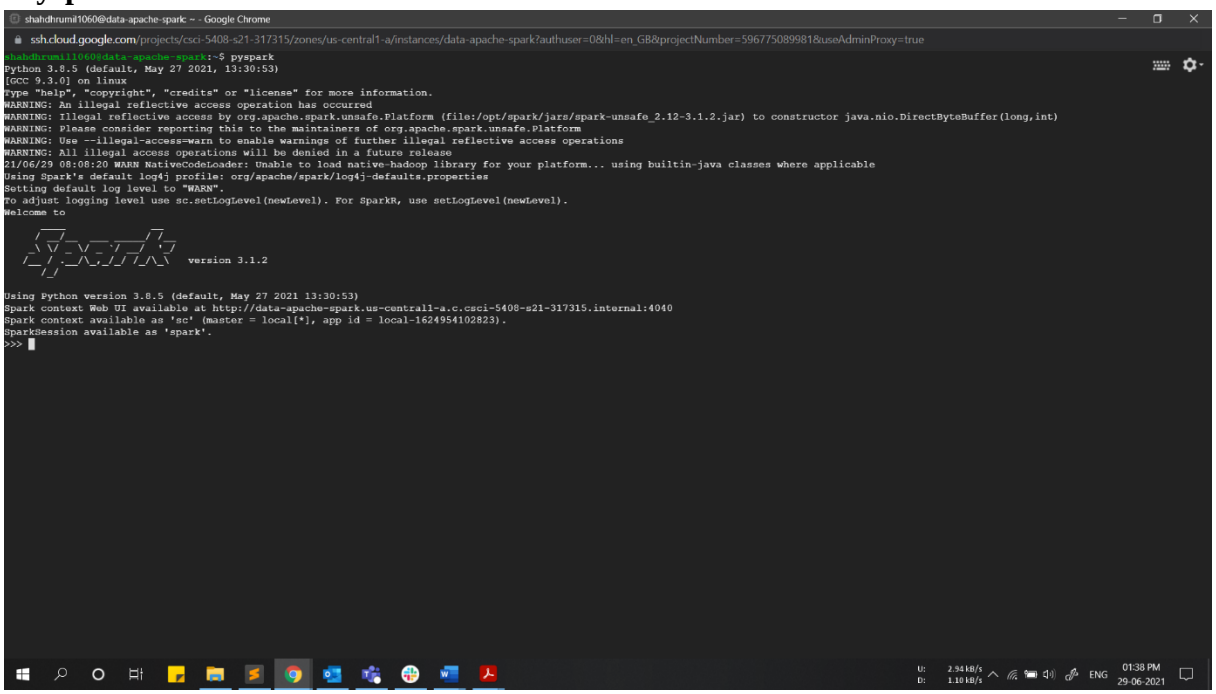

Figure 3. Starting Spark using pyspark command, created using GCP [9].

3. End.

Task2

NOTE:

To run the NewsProcessingEngine.java file following things should be done:

1. Change the path where the news articles files will be generated or create a **data** named folder in drive **E.**
2. A database named **myMongoNews** needs to be created on MongoDB along with a **drshah** named collection.

Algorithm/Pseudocode of the Extraction Engine used in NewsProcessingEngine.java file

1. Start
2. Instantiating and Initializing the NewsApiClient and passing my API Key as an argument
3. Creating an arraylist of searchKeywords
   (The keywords for which News articles are needed to be searched)
   3.1. Adding the keywords to the searchKeywords arraylist
4. Instantiating and Initializing a JSONObject and making it final
5. Instantiating and Initializing MongoCleint and passing host name and port as arguments
   (host: localhost, port: 27017)
6. Instantiating and Initializing the MongoCredential to null
   6.1. Creating a try catch block
      6.1.1. Passing the userName and database as arguments
7. Connecting to the MongoDB Database by using the MongoClient object and getDatabase() method
   (databaseName: myMongoNews)
8. Instantiating and Initializing the MongoCollection with type Document using the MongoDatabase object and getCollection() method
   8.1. Passing the cluster name as an argument to the getConnection() method
9. Looping through searchKeywords arraylist
   9.1. Storing the current loop iteration in an integer variable
   9.2. Using the newsApiClient's getEverything() method to get 100 news articles for each keyword
   9.3. Overriding the onSuccess() method
      9.3.1. Instantiating the FileWriter
      9.3.2. Maintaining a counter
      9.3.3. Looping through the articleResponse.getArticles() having size of 100
         (Iterating through the fetched articles for current keyword)
      9.3.4. Creating a try catch block
         9.3.4.1. Dividing the articles in groups of 5
         9.3.4.2. Creating a new file for every 6th news article fetched
            (Storing all the files in a separate folder created in **E** drive under folder named **data**)
            (The file names are <keyword><counter>)
            (20 files are generated per keyword containing 5 articles each)
         9.3.4.3. Incrementing the counter
      9.3.5. Using jsonObject to put the fetched title and description of news articles
      9.3.6. Creating a try catch block
         9.3.6.1. Writing the contents fetched to the current file created
         9.3.6.2. Flushing the file after the fetched news articles are written to the file

9.4. Coming out of the onSuccess() method
9.5. Overriding the onFailure() method with Throwable object as an argument
9.5.1. Printing the message on console
10. Coming out of the searchKeywords () loop
10.1. Creating a try catch block
10.1.1. Creating a new file and passing the path where the raw fetched news articles files are created
10.1.2. Creating an array of files to read and store all the generated files
10.1.3. Looping through the generated files list
10.1.3.1. Using Scanner to read the file
10.1.3.2. Using StringBuilder to read the contents of the file and storing it in a String
10.1.3.3. Splitting the String using RegEx
10.1.3.4. Creating a list of documents
10.1.3.5. Lopping through the read data from the current file
10.1.3.5.1. Calling removingSpecialCharacters(), removingHTML(), removingEOL(), removingURL(), removingUnicode() and getTtile() and getDescription() methods (These methods filter the raw data read from the file)
10.1.3.5.2. Adding the new doc to the list of documents
10.1.3.6. Coming out of the loop
10.1.4. Inserting the list of docs to the collection object
10.2. Coming out of the loop
11. End

Flowchart of the Filtration Engine used in the NewsProcessingEngine.java file

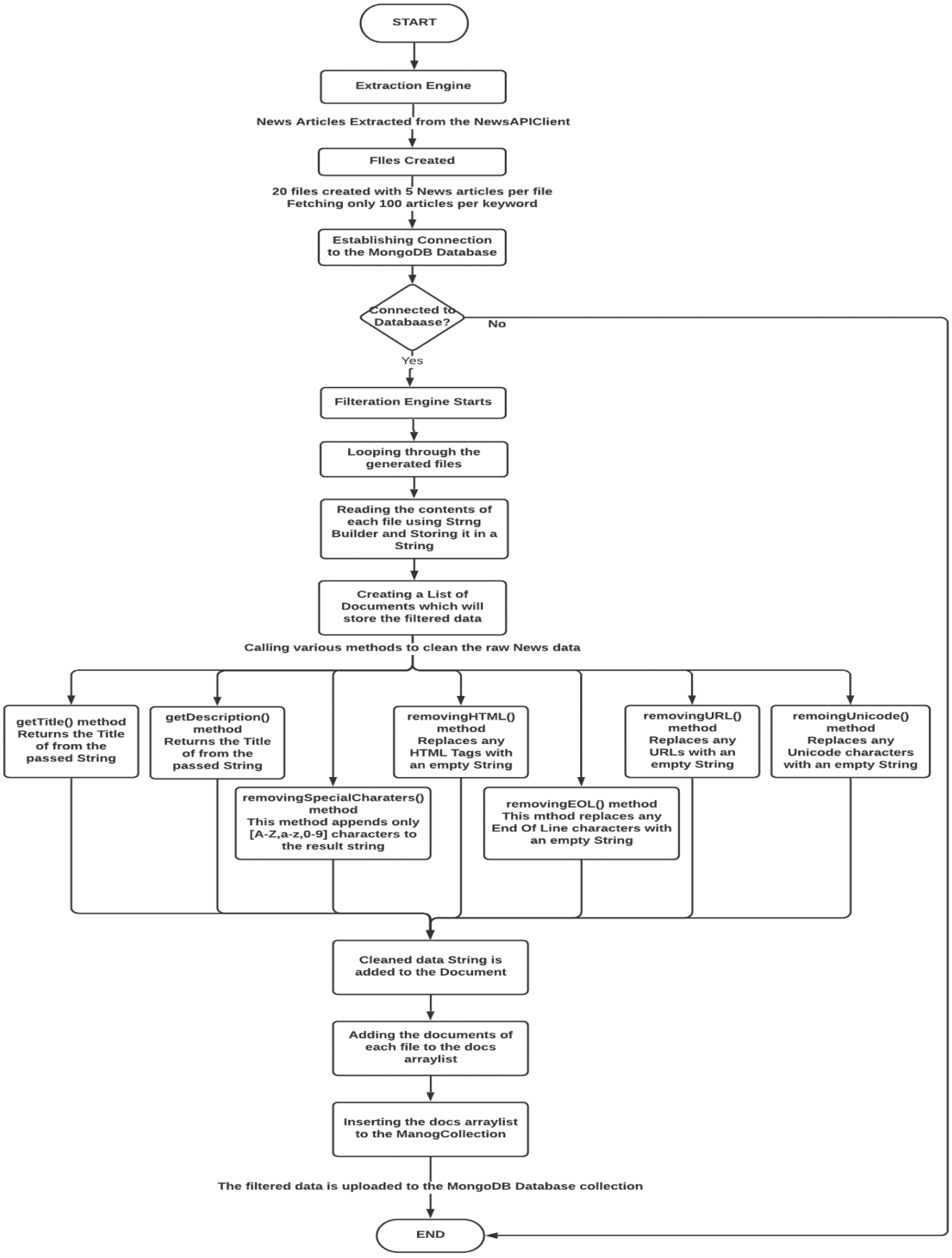

Figure 4. Flowchart of the Filtration Engine, created using LucidChart [8].

Task3

NOTE:

To run the WordCounterEngine.java file following things should be done:

1. Change the path where the news articles files will be read from (raw data) or create a **data** named folder in drive **E** and run the **NewsProcessingEngine.java** file which will generate the files containing raw data.

Algorithm of the WordCounterEngine.java file

1. Start
2. Creating an Array List of searchKeywords of type String
3. Creating a Hash Map of wordFrequencyCount with <key, Value> pair of type <String, Integer>
   (The Hash Map will store the data in the following manner:
   Key → searchKeyword → String
   Value → frequency → Integer)
4. Adding the following Keywords to the Array List of searchKeywords
   (Canada, NovaScotia, education, higher, learning, city, accommodation, price)
   (The keywords are case sensitive)
5. Creating a new file and passing the path where the files containing raw data are created
   (Stored all the files in a separate folder created in **E** drive under folder named **data**)
   (The file names are <keyword><counter>)
   (20 files are generated per keyword containing 5 articles each)
6. Creating an array of files to read and store all the generated files
7. Creating a Scanner object
8. Creating a String array named words, which will store all the words of the current file being read
9. Looping through the generated files list containing raw News data
   9.1. Using Scanner to read the file
   9.2. Using StringBuilder to read the contents of the file and storing it in a String
   9.3. Filtering the raw data read from the file using RegEx for each of the following
   (Special Characters, HTML Tags, URLs, Unicode Characters, End of Line Characters)
   (Replacing the above with an empty space)
   9.4. Now, splitting the contents of the file with space
   9.5. Storing the individual **words** in a words String array
   9.6. Looping through the words String array
   9.6.1. Looping through the searchKeywords Array List
   9.6.1.1. Checking if the current word is present in the searchKeywords Array List of keywords
   (That is whether the word is a keyword or not)
   9.6.1.1.1. If the word matches the keyword then fetching that words value (frequency) from the Hash Map
   9.6.1.1.2. Checking if the wordFrequencyCount contains the current keyword
   9.6.1.1.2.1. If No then,
   9.6.1.1.2.1.1. Adding the word to the wordFrequencyCount HashMap and setting the frequency to 1
   9.6.1.1.2.2. If Yes then,

9.6.1.1.2.2.1. Incrementing the current keyword's frequency by 1

9.6.2. Coming out of the loop

9.7. Coming out of the loop

10. Printing the wordFrequencyCount HashMap
11. End

Output:

{Canada=430, education=10, NovaScotia=108, city=26, price=4, accommodation=0, learning=5, higher=11}

Canada → 430 → Has the highest frequency
Accommodation → 0 → Has the lowest frequency

Problem 2
Task1

Nova Scotia Provincial Parks:

Table 1. Nova Scotia Provincial Parks List [1]

Regions	Parks
The Bay of Fundy Shore and Annapolis Valley	Annapolis Basin Look Off
	Anthony
	Bell
	Blomidon
	Blomidon Lookoff
	Caddell Rapids Lookoff
	Cape Chignecto
	Cape Split
	Central Grove
	Clairmont
	Coldbrook
	Cottage Cove
	Eatonville
	Falls Lake
	Five Islands
	Lake George
	Lake Midway
	Londonderry
	Lumsden Pond
	MacElmons Pond
	Mickey Hill
	Savary
	Scots Bay
	Smileys
	Valleyview
	Wentworth
Northumberland Shore	Amherst Shore
	Arisaig
	Balmoral Mills
	Bayfield Beach
	Beaver Mountain
	Blue Sea Beach
	Caribou-Munroes Island
	Fox Harbour
	Green Hill
	Gulf Shore
	Heather Beach
	Melmerby Beach
	Northport Beach
	Pomquet Beach

	Powells Point
	Rushtons Beach
	Salt Springs
	Shinimicas
	Tatamagouche
	Tidnish Dock
	Waterside Beach
Cape Breton Island	Barrachois
	Battery
	Ben Eoin
	Burnt Island
	Cabots Landing
	Cape Smokey
	Dalem Lake
	Dominion Beach
	Dundee
	Groves Point
	Irish Cove
	Lake O' Law
	Lennox Passage
	Mabou
	MacCormack
	Mira River
	North River
	Petersfield
	Point Michaud Beach
	Pondville Beach
	Port Hood Station
	Ross Ferry
	St Anns
	Trout Brook
	Uisge Bàn Falls
	West Mabou Beach
	Whycocomagh
Eastern Shore	Port Shoreham Beach
	Black Duck Cove
	Tor Bay
	Lochiel Lake
	Sherbrooke
	Marie Joseph
	Rainbow Haven Beach
	Lawrencetown Beach
	Porters Lake
	Martinique Beach
	Clam Harbour Beach
	Dollar Lake
	Elderbank
	Moose River Gold Mines

	Musquodoboit Valley
	Spry Bay
	Taylor Head
	Salsman
	Boylston
Halifax Region Metro	Oakfield
	Crystal Crescent Beach
	MacCormacks Beach
	McNabs and Lawlor Islands
	Jerry Lawrence
	Laurie
South Shore Region	Sand Hills Beach
	Cleveland Beach
	Queensland Beach
	Hubbards
	The Islands
	Sable River
	Thomas Raddall
	Summerville Beach
	Ten Mile Lake
	Camerons Brook
	Rissers Beach
	Fancy Lake
	Cookville
	Second Peninsula
	Graves Island
	Card Lake
	East River
	Bayswater
Yarmouth & Acadian Shore	Ellenwood Lake
	Glenwood
	Port Maitland
	Mavillette Beach
	Smugglers Cove

Cyphers:

1. First adding the uniqueness constraint
    a. Cypher Query
    CREATE CONSTRAINT ON (r:Region) ASSERT r.name IS UNIQUE

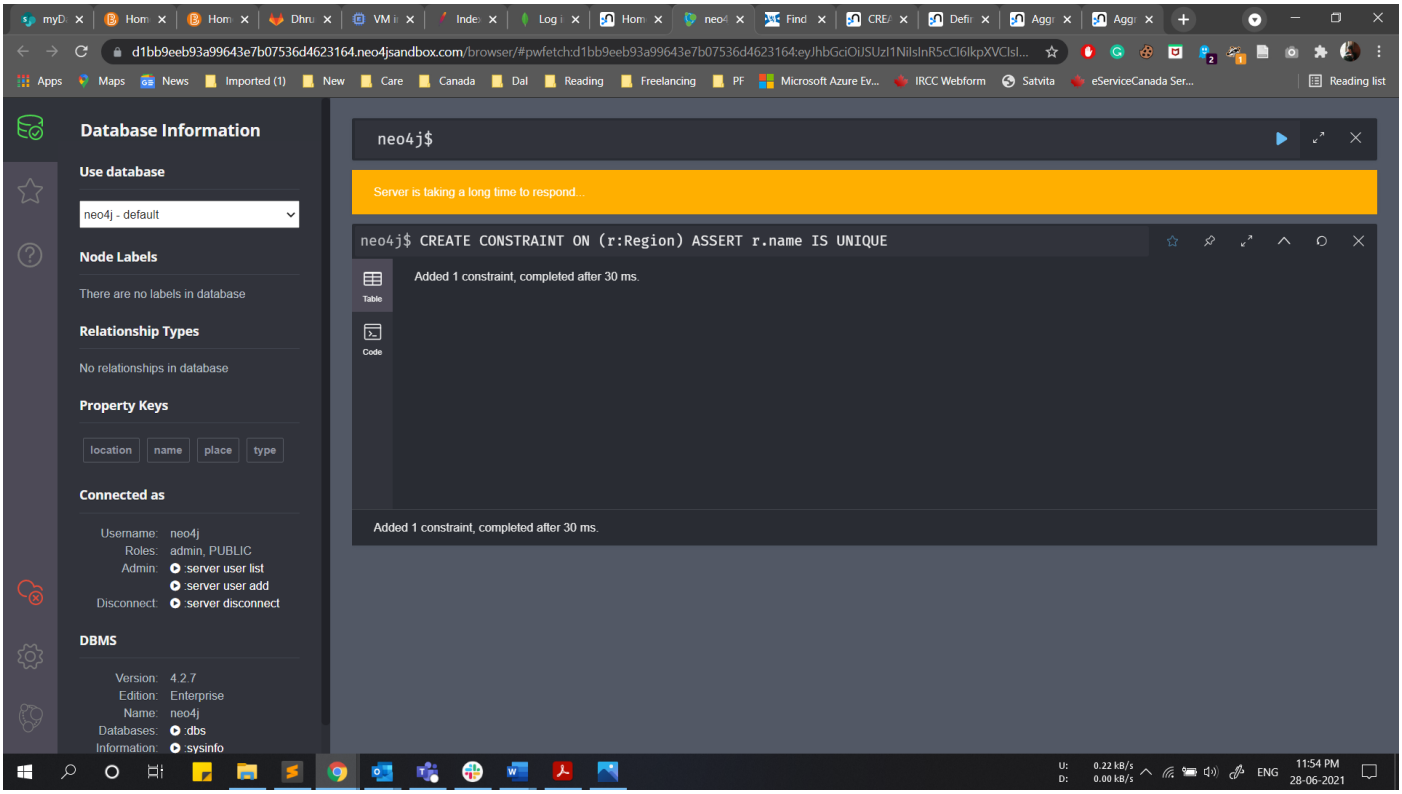

Figure 5. Constraint Cypher Query, created using Neo4j Sandbox [5].

2. Creating the Region Nodes and Relationships as per the Table 1.
   (Here I have also added meaningful label named location to the Region nodes)

   a. Cypher Query
   CREATE (:Region {name: "The Bay of Fundy Shore and Annapolis Valley", location:"NS"})-[:CONNECTED_TO_REGION]->(:Region {name: "Northumberland Shore", location:"NS"})-[:CONNECTED_TO_REGION]->(:Region {name: "Cape Breton Island", location:"NS"})-[:CONNECTED_TO_REGION]->(:Region {name: "Eastern Shore", location:"NS"})-[:CONNECTED_TO_REGION]->(:Region {name: "Halifax Region Metro", location:"NS"})-[:CONNECTED_TO_REGION]->(:Region {name: "South Shore Region", location:"NS"})-[:CONNECTED_TO_REGION]->(:Region {name: "Yarmouth & Acadian Shore", location:"NS"})

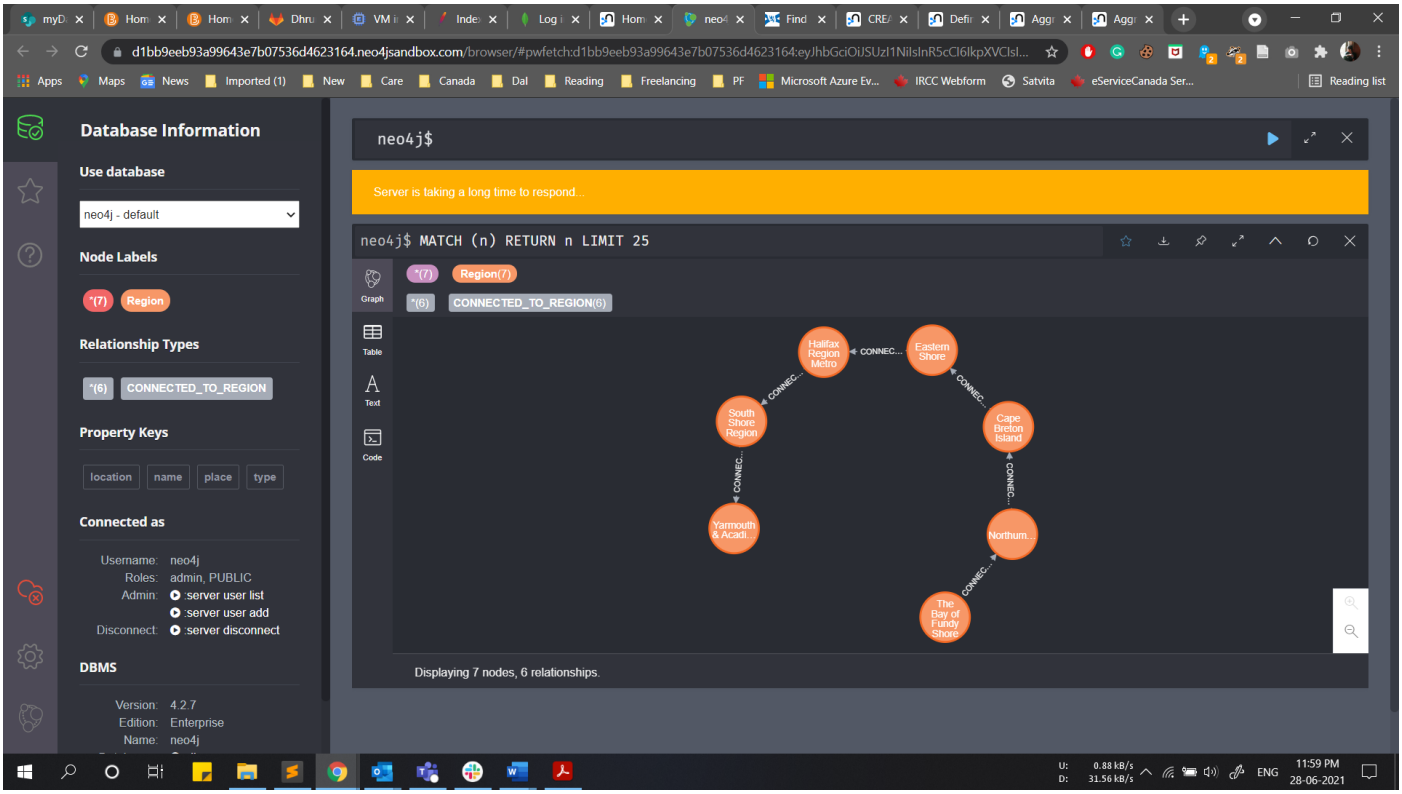

Figure 6. Region Nodes Create Cypher Query, created using Neo4j Sandbox [5].

3. Adding new relationships between the above created Region Nodes
   (Basically, creating relationships between all the nodes making the Regions fully connected)

   a. Cypher Query

```
--Yarmouth & Acadian Shore--
MATCH
(r1:Region {name: "Yarmouth & Acadian Shore"}),
(r2:Region {name: "The Bay of Fundy Shore and Annapolis Valley"}),
(r3:Region {name: "South Shore Region"}),
(r4:Region {name: "Northumberland Shore"}),
(r5:Region {name: "Cape Breton Island"}),
(r6:Region {name: "Eastern Shore"}),
(r7:Region {name: "Halifax Region Metro"})
CREATE
(r1)-[:CONNECTED_TO_REGION]->(r2),
(r1)-[:CONNECTED_TO_ REGION]->(r3),
(r1)-[:CONNECTED_TO_ REGION]->(r4),
(r1)-[:CONNECTED_TO_ REGION]->(r5),
(r1)-[:CONNECTED_TO_ REGION]->(r6),
(r1)-[:CONNECTED_TO_ REGION]->(r7)

--South Shore Region--
MATCH
(r1:Region {name: "South Shore Region"}),
(r2:Region {name: "The Bay of Fundy Shore and Annapolis Valley"}),
(r3:Region {name: "Northumberland Shore"}),
(r4:Region {name: "Cape Breton Island"}),
(r5:Region {name: "Eastern Shore"}),
```

```
(r6:Region {name: "Halifax Region Metro"})
CREATE
(r1)-[:CONNECTED_TO_ REGION]->(r2),
(r1)-[:CONNECTED_TO_ REGION]->(r3),
(r1)-[:CONNECTED_TO_ REGION]->(r4),
(r1)-[:CONNECTED_TO_ REGION]->(r5),
(r1)-[:CONNECTED_TO_ REGION]->(r6)

--Halifax Region Metro--
MATCH
(r1:Region {name: "Halifax Region Metro"}),
(r2:Region {name: "The Bay of Fundy Shore and Annapolis Valley"}),
(r3:Region {name: "Northumberland Shore"}),
(r4:Region {name: "Cape Breton Island"}),
(r5:Region {name: "Eastern Shore"}),
(r6:Region {name: "Yarmouth & Acadian Shore"})
CREATE
(r1)-[:CONNECTED_TO_REGION]->(r2),
(r1)-[:CONNECTED_TO_REGION]->(r3),
(r1)-[:CONNECTED_TO_REGION]->(r4),
(r1)-[:CONNECTED_TO_REGION]->(r5),
(r1)-[:CONNECTED_TO_REGION]->(r6)

--Eastern Shore--
MATCH
(r1:Region {name: "Eastern Shore"}),
(r2:Region {name: "The Bay of Fundy Shore and Annapolis Valley"}),
(r3:Region {name: "South Shore Region"}),
(r4:Region {name: "Northumberland Shore"}),
(r5:Region {name: "Cape Breton Island"}),
(r6:Region {name: "Yarmouth & Acadian Shore"})
CREATE
(r1)-[:CONNECTED_TO_REGION]->(r2),
(r1)-[:CONNECTED_TO_REGION]->(r3),
(r1)-[:CONNECTED_TO_REGION]->(r4),
(r1)-[:CONNECTED_TO_REGION]->(r5),
(r1)-[:CONNECTED_TO_REGION]->(r6)

--Cape Breton Island--
MATCH
(r1:Region {name: "Cape Breton Island"}),
(r2:Region {name: "The Bay of Fundy Shore and Annapolis Valley"}),
(r3:Region {name: "South Shore Region"}),
(r4:Region {name: "Northumberland Shore"}),
(r5:Region {name: "Yarmouth & Acadian Shore"}),
(r6:Region {name: "Halifax Region Metro"})
CREATE
(r1)-[:CONNECTED_TO_REGION]->(r2),
(r1)-[:CONNECTED_TO_REGION]->(r3),
(r1)-[:CONNECTED_TO_REGION]->(r4),
(r1)-[:CONNECTED_TO_REGION]->(r5),
```

```
(r1)-[:CONNECTED_TO_REGION]->(r6)

--Northumberland Shore--
MATCH
(r1:Region {name: "Northumberland Shore"}),
(r2:Region {name: "The Bay of Fundy Shore and Annapolis Valley"}),
(r3:Region {name: "South Shore Region"}),
(r4:Region {name: "Yarmouth & Acadian Shore"}),
(r5:Region {name: "Eastern Shore"}),
(r6:Region {name: "Halifax Region Metro"})
CREATE
(r1)-[:CONNECTED_TO_REGION]->(r2),
(r1)-[:CONNECTED_TO_REGION]->(r3),
(r1)-[:CONNECTED_TO_REGION]->(r4),
(r1)-[:CONNECTED_TO_REGION]->(r5),
(r1)-[:CONNECTED_TO_REGION]->(r6)

--The Bay of Fundy Shore and Annapolis Valley--
MATCH
(r1:Region {name: "The Bay of Fundy Shore and Annapolis Valley"}),
(r2:Region {name: "Yarmouth & Acadian Shore"}),
(r3:Region {name: "South Shore Region"}),
(r4:Region {name: "Cape Breton Island"}),
(r5:Region {name: "Eastern Shore"}),
(r6:Region {name: "Halifax Region Metro"})
CREATE
(r1)-[:CONNECTED_TO_REGION]->(r2),
(r1)-[:CONNECTED_TO_REGION]->(r3),
(r1)-[:CONNECTED_TO_REGION]->(r4),
(r1)-[:CONNECTED_TO_REGION]->(r5),
(r1)-[:CONNECTED_TO_REGION]->(r6)
```

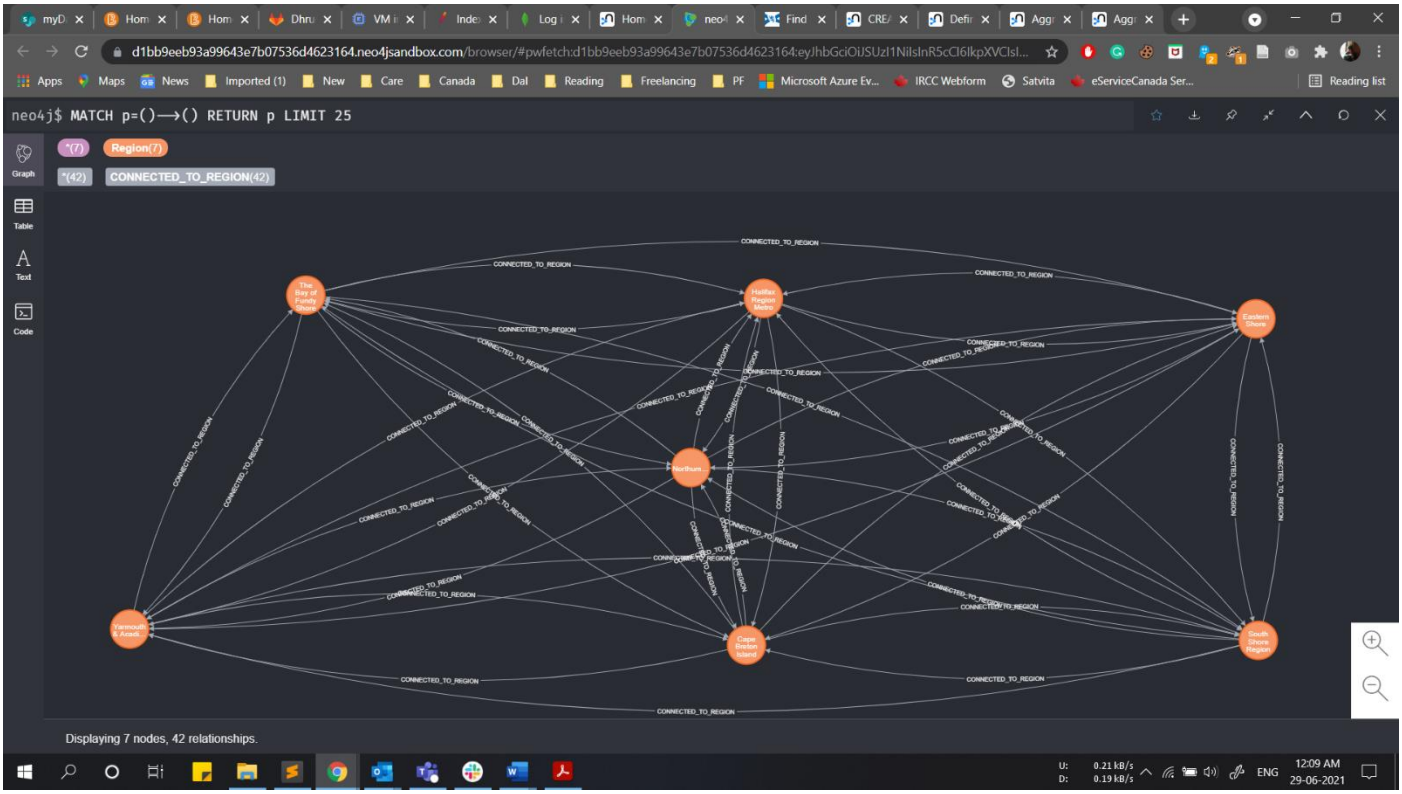

Figure 7. Relationship between Region Nodes Cypher Query, created using Neo4j Sandbox [5].

4. Creating Park nodes and adding their relationships with the already created Region nodes **(Here I have also added meaningful label named location and type to the Park nodes)**

    a. Cypher Query

--Yarmouth & Acadian Shore--
MATCH
(r:Region {name: "Yarmouth & Acadian Shore"})
CREATE
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Ellenwood Lake", location:"Yarmouth & Acadian Shore", type:"Camping Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Glenwood", location:"Yarmouth & Acadian Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Port Maitland", location:"Yarmouth & Acadian Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Mavillette Beach", location:"Yarmouth & Acadian Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Smugglers Cove", location:"Yarmouth & Acadian Shore", type:"Day Use Park"})

--South Shore Region--
MATCH
(r:Region {name: "South Shore Region"})
CREATE
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Sand Hills Beach", location:"South Shore Region", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Cleveland Beach", location:"South Shore Region", type:"Day Use Park"}),

```
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Queensland Beach", location:"South Shore Region", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Hubbards", location:"South Shore Region", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"The Islands", location:"South Shore Region", type:"Camping Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Sable River", location:"South Shore Region", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Thomas Raddall", location:"South Shore Region", type:"Camping Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Summerville Beach", location:"South Shore Region", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Ten Mile Lake", location:"South Shore Region", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Camerons Brook", location:"South Shore Region", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Rissers Beach", location:"South Shore Region", type:"Camping Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Fancy Lake", location:"South Shore Region", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Cookville", location:"South Shore Region", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Second Peninsula", location:"South Shore Region", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Graves Island", location:"South Shore Region", type:"Camping Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Card Lake", location:"South Shore Region", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"East River", location:"South Shore Region", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Bayswater", location:"South Shore Region", type:"Day Use Park"})

--Halifax Region Metro--
MATCH
(r:Region {name: "Halifax Region Metro"})
CREATE
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Oakfield", location:"Halifax Region Metro", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Crystal Crescent Beach", location:"Halifax Region Metro", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"MacCormacks Beach", location:"Halifax Region Metro", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"McNabs and Lawlor Islands", location:"Halifax Region Metro", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Jerry Lawrence", location:"Halifax Region Metro", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Laurie", location:"Halifax Region Metro", type:"Camping Park"})

--Eastern Shore--
```

```
MATCH
(r:Region {name: "Eastern Shore"})
CREATE
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Port Shoreham Beach",
location:"Eastern Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Black Duck Cove",
location:"Eastern Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Tor Bay", location:"Eastern Shore",
type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Lochiel Lake", location:"Eastern
Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Sherbrooke", location:"Eastern
Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Marie Joseph", location:"Eastern
Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Rainbow Haven Beach",
location:"Eastern Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Lawrencetown Beach",
location:"Eastern Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Porters Lake", location:"Eastern
Shore", type:"Camping Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Martinique Beach",
location:"Eastern Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Clam Harbour Beach",
location:"Eastern Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Dollar Lake", location:"Eastern
Shore", type:"Camping Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Elderbank", location:"Eastern
Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Moose River Gold Mines",
location:"Eastern Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Musquodoboit Valley",
location:"Eastern Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Spry Bay", location:"Eastern
Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Taylor Head", location:"Eastern
Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Salsman", location:"Eastern Shore",
type:"Camping Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Boylston", location:"Eastern Shore",
type:"Camping Park"})

--Cape Breton Island--
MATCH
(r:Region {name: "Cape Breton Island"})
CREATE
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Barrachois", location:"Cape Breton
Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Battery", location:"Cape Breton
Island", type:"Camping Park"}),
```

(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Ben Eoin", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Burnt Island", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Cabots Landing", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Cape Smokey", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Dalem Lake", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Dominion Beach", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Dundee", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Groves Point", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Irish Cove", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Lake O' Law", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Lennox Passage", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Mabou", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"MacCormack", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Mira River", location:"Cape Breton Island", type:"Camping Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"North River", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Petersfield", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Point Michaud Beach", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Pondville Beach", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Port Hood Station", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Ross Ferry", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"St Anns", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Trout Brook", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Uisge Bàn Falls", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"West Mabou Beach", location:"Cape Breton Island", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Whycocomagh", location:"Cape Breton Island", type:"Camping Park"})

```
--Northumberland Shore--
MATCH
(r:Region {name: "Northumberland Shore"})
CREATE
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Amherst Shore",
location:"Northumberland Shore", type:"Camping Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Arisaig", location:"Northumberland
Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Balmoral Mills",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Bayfield Beach",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Beaver Mountain",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Blue Sea Beach",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Caribou-Munroes Island",
location:"Northumberland Shore", type:"Camping Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Fox Harbour",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Green Hill",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Gulf Shore",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Heather Beach",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Melmerby Beach",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Northport Beach",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Pomquet Beach",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Powells Point",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Rushtons Beach",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Salt Springs",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Shinimicas",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Tatamagouche",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Tidnish Dock",
location:"Northumberland Shore", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Waterside Beach",
location:"Northumberland Shore", type:"Day Use Park"})

--The Bay of Fundy Shore and Annapolis Valley--
MATCH
(r:Region {name: "The Bay of Fundy Shore and Annapolis Valley"})
CREATE
```

(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Annapolis Basin Look Off", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Anthony", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Bell", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Blomidon", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Camping Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Blomidon Lookoff", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Caddell Rapids Lookoff", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Cape Chignecto", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Camping Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Cape Split", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Central Grove", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Clairmont", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Coldbrook", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Cottage Cove", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Eatonville", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Falls Lake", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Five Islands", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Camping Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Lake George", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Lake Midway", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Londonderry", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Lumsden Pond", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"MacElmons Pond", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Mickey Hill", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Savary", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Scots Bay", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Smileys", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Camping Park"}),
(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Valleyview", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Camping Park"}),

(r)-[:CONNECTED_TO_PARK]->(:Park {name:"Wentworth", location:"The Bay of Fundy Shore and Annapolis Valley", type:"Day Use Park"})

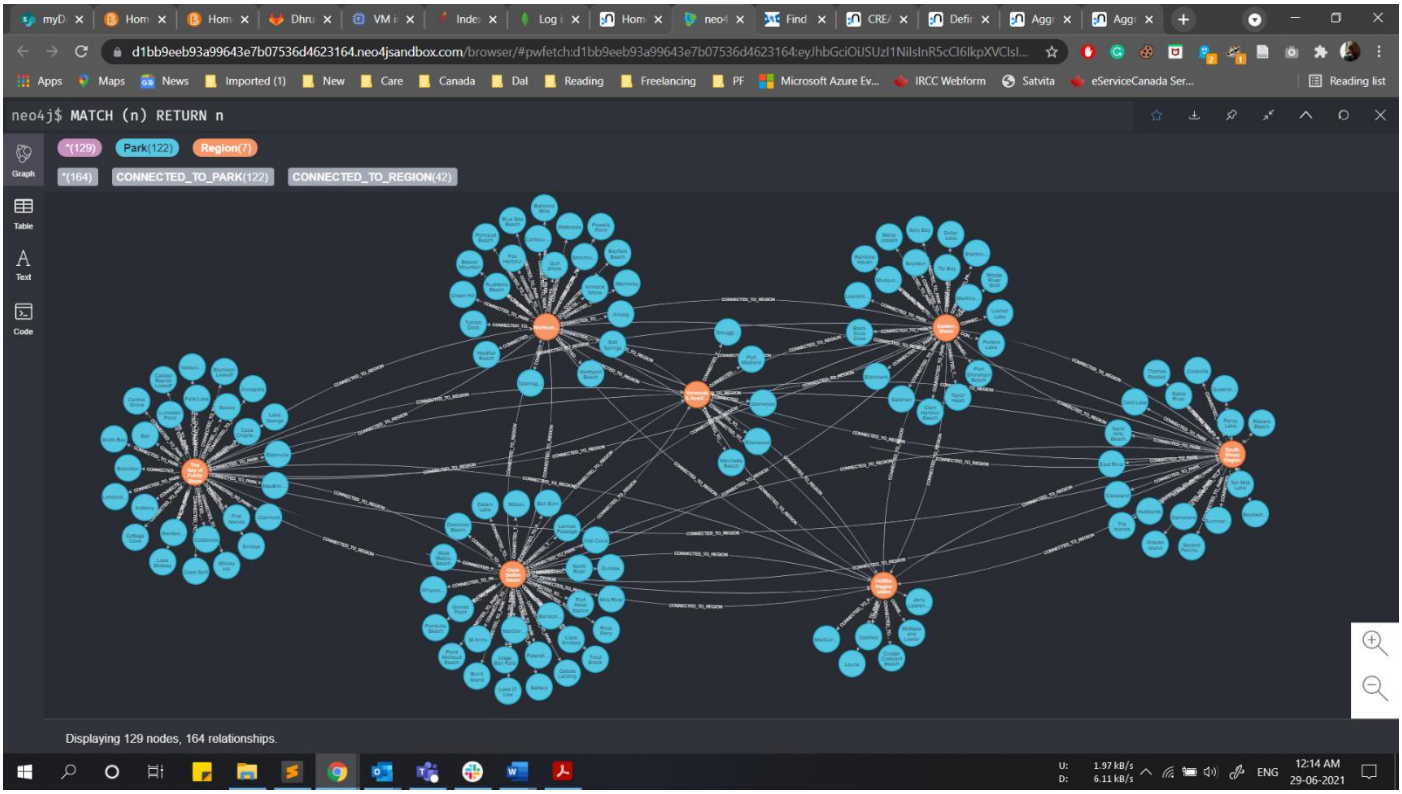

Figure 8. Creating Park Nodes and Adding Relationships Cypher Query, created using Neo4j Sandbox [5].

5. Writing the Cypher query that returns the count of each Region's Parks.
   (The below query puts a condition of DISTINCT regions being matched, COUNT the number of parks for each region and ORDER BY the number of parks in DESC order)
   a. Cypher Query

```
MATCH
(r:Region)-[:CONNECTED_TO_PARK]->(:Park)
WITH
DISTINCT r.name AS regionName, COUNT(*) AS numberOfParks
ORDER BY
numberOfParks DESC
RETURN
regionName, numberOfParks
```

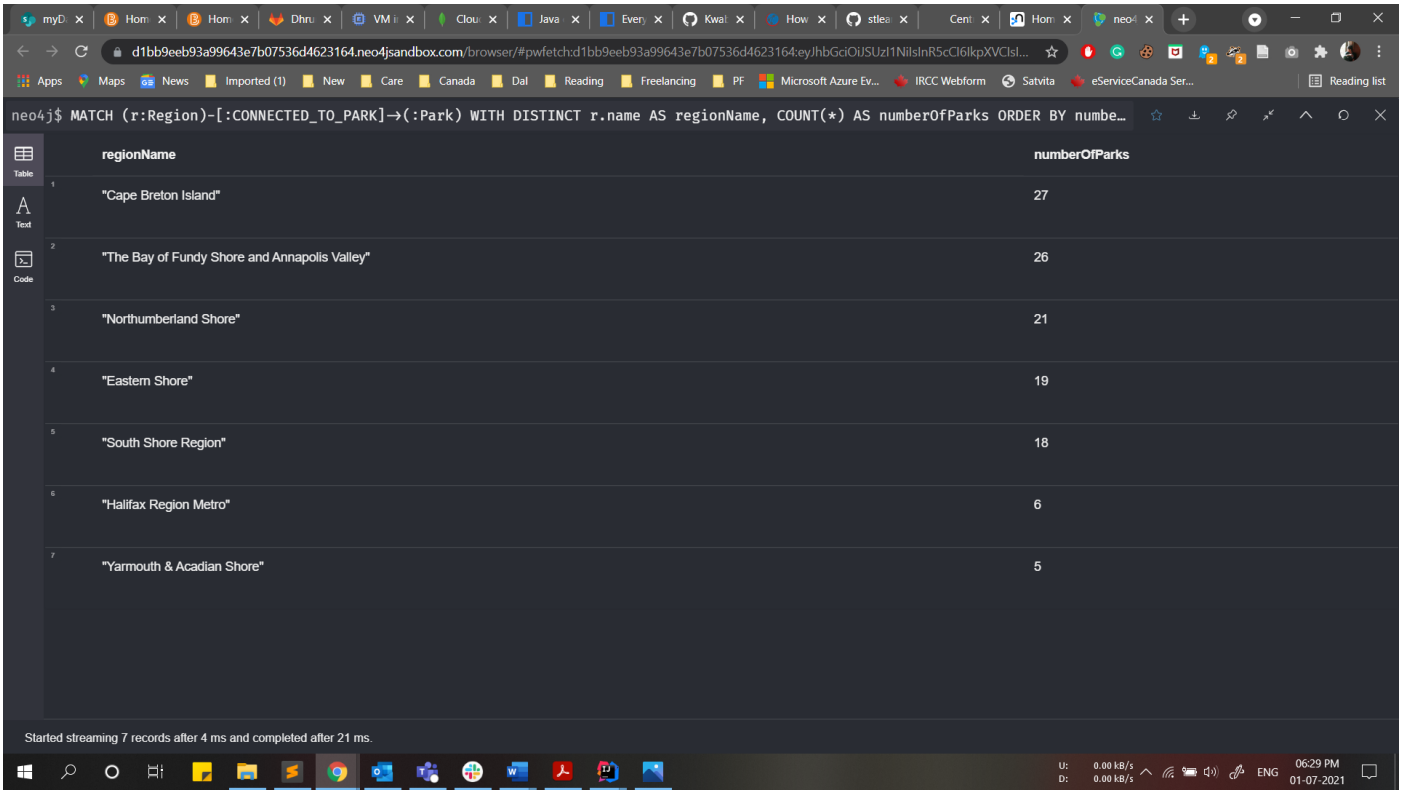

Figure 9. Counting each Region's Park Nodes Cypher Query, created using Neo4j Sandbox [5].

6. The below query returns the region with max number of parks of all the regions
    a. Cypher Query

```
MATCH
(r:Region)-[:CONNECTED_TO_PARK]->(:Park)
WITH
DISTINCT r.name AS regionName, COUNT(*) AS numberOfParks
ORDER BY
numberOfParks DESC LIMIT 1
RETURN
regionName, numberOfParks
```

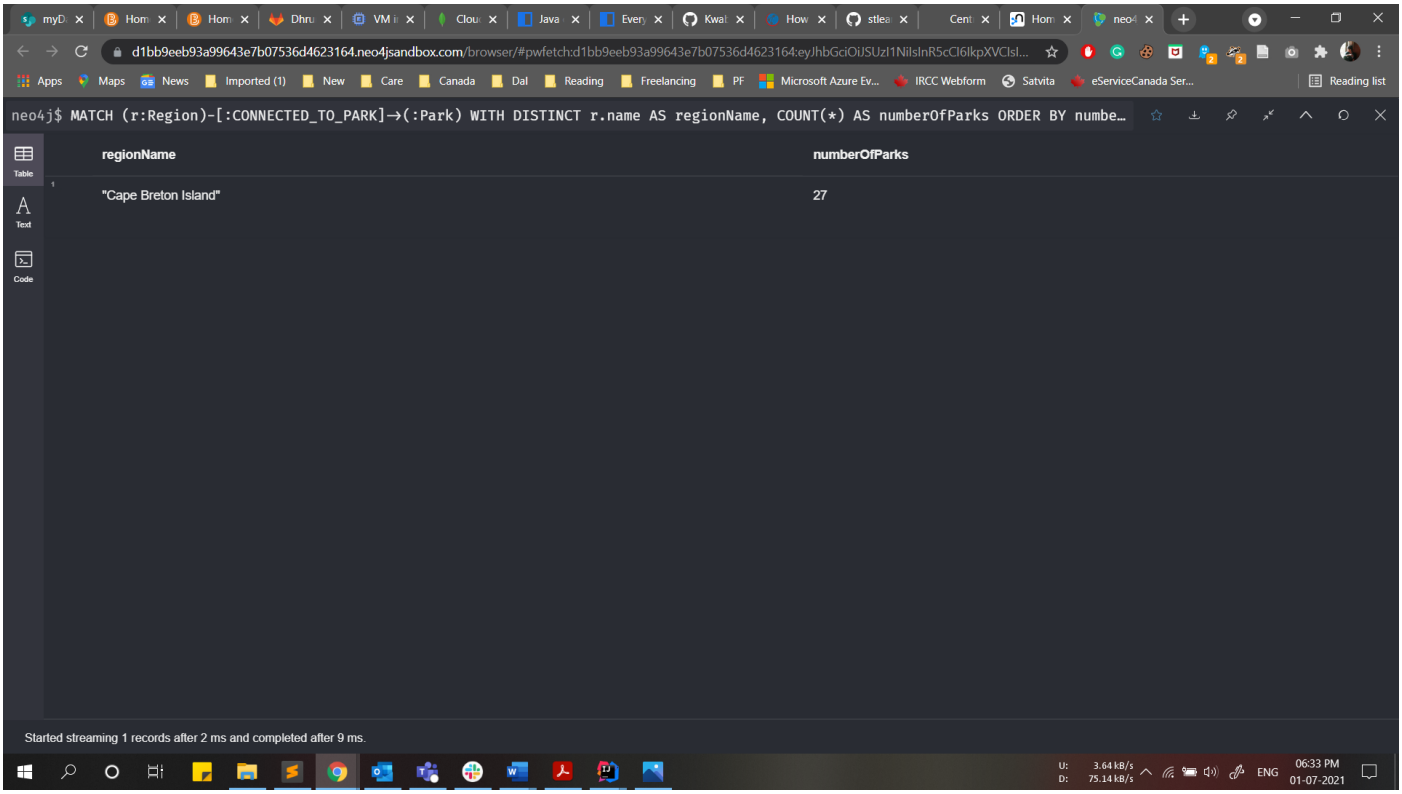

Figure 10. Returning the Region with highest Park Nodes Cypher Query, created using Neo4j Sandbox [5].

7. Additional Cypher Queries
    a. Dropping the uniqueness constraint
        i. Cypher Query

DROP CONSTRAINT ON (r:Region) ASSERT r.name IS UNIQUE

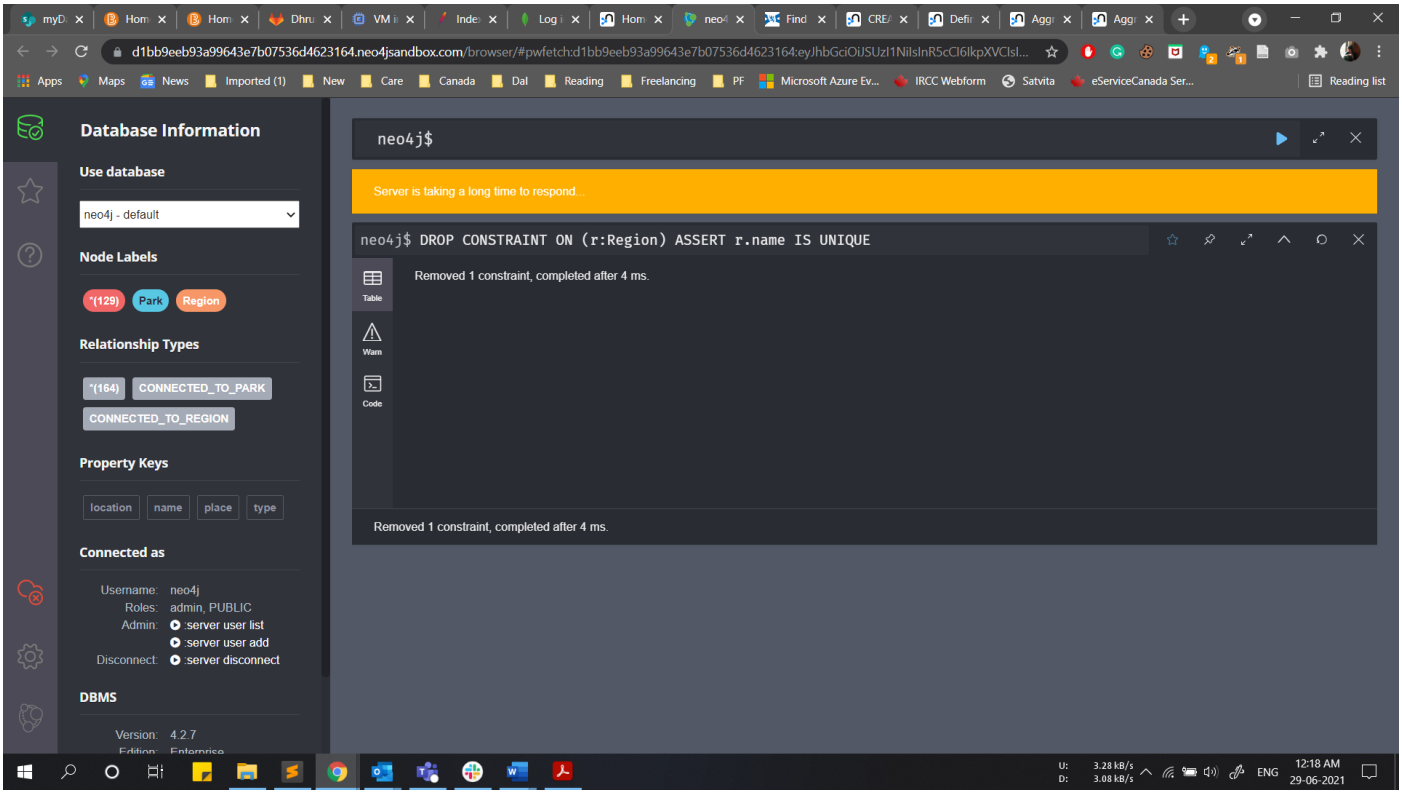

Figure 11. Drop Constraint Cypher Query, created using Neo4j Sandbox [5].

b. Deleting all the Region nodes by detaching all the Relationships of that node
   i. Cypher Query
   MATCH (r:Region) DETACH DELETE r

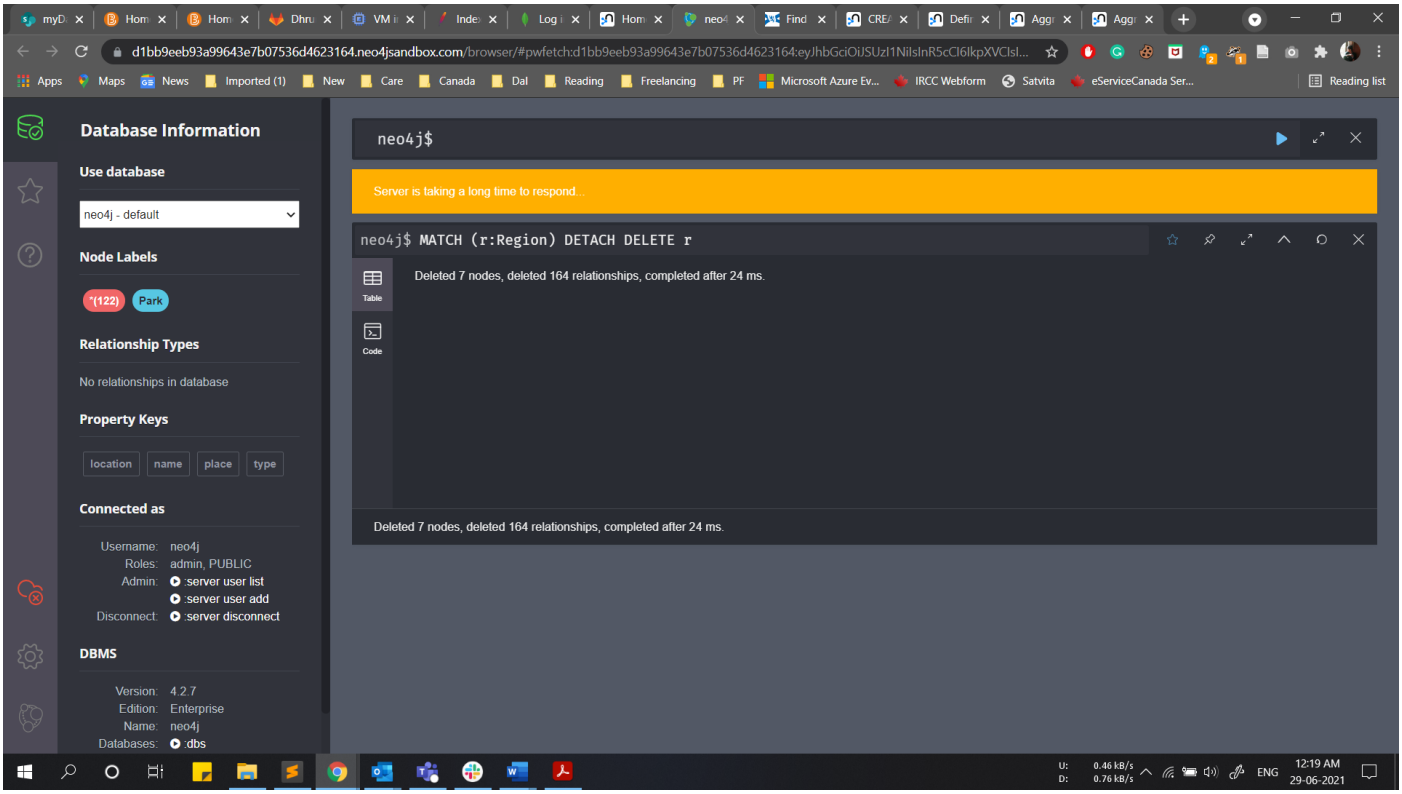

Figure 12. Delete and Detach Region Nodes Cypher Query, created using Neo4j Sandbox [5].

c. Deleting all the Park nodes by detaching all the Relationships of that node
   i. Cypher Query
   MATCH (p:Park) DETACH DELETE p

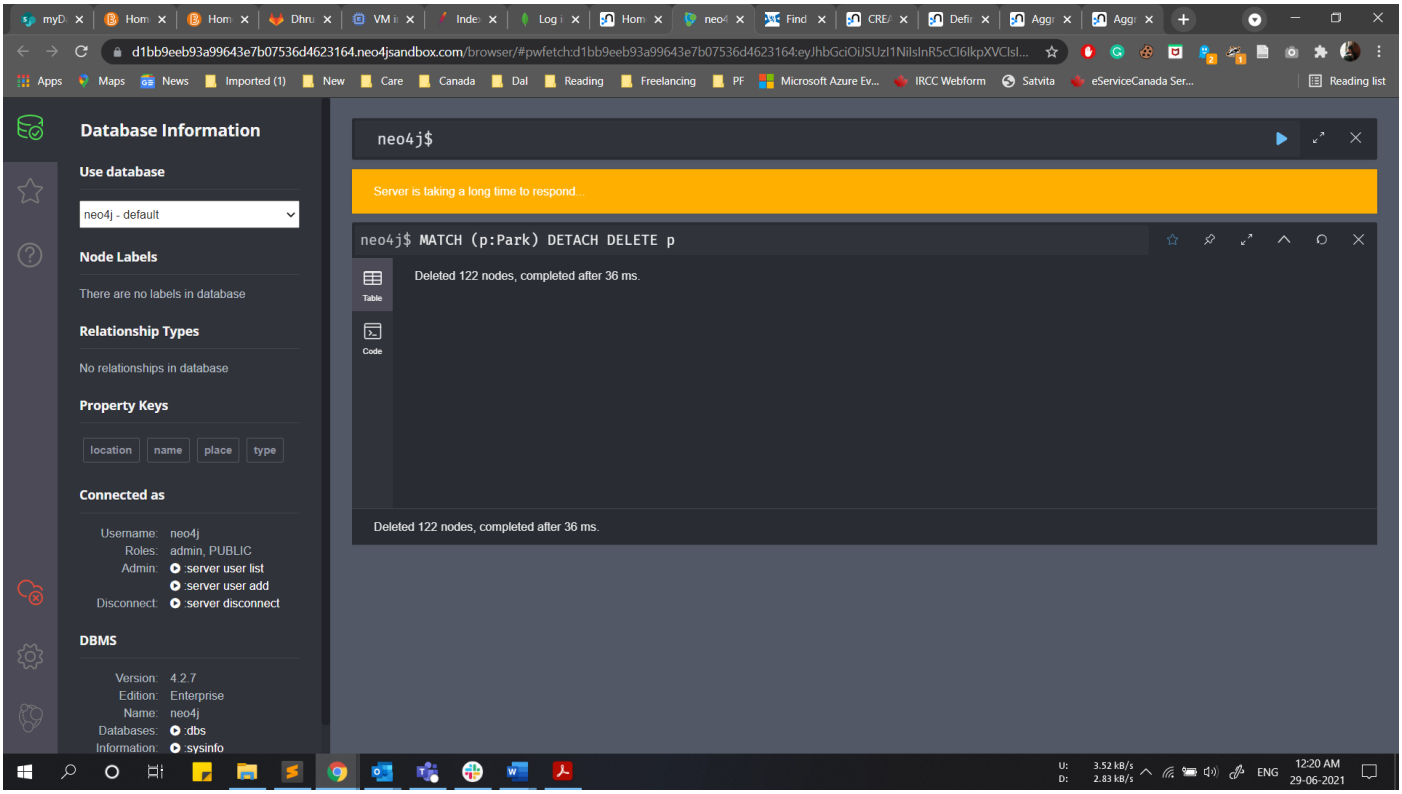

Figure 13. Delete and Detach Park Nodes Cypher Query, created using Neo4j Sandbox [5].

d. Query to retrieve all the nodes
   i. Cypher Queries

MATCH (n) RETURN n

MATCH p=()-->() RETURN p

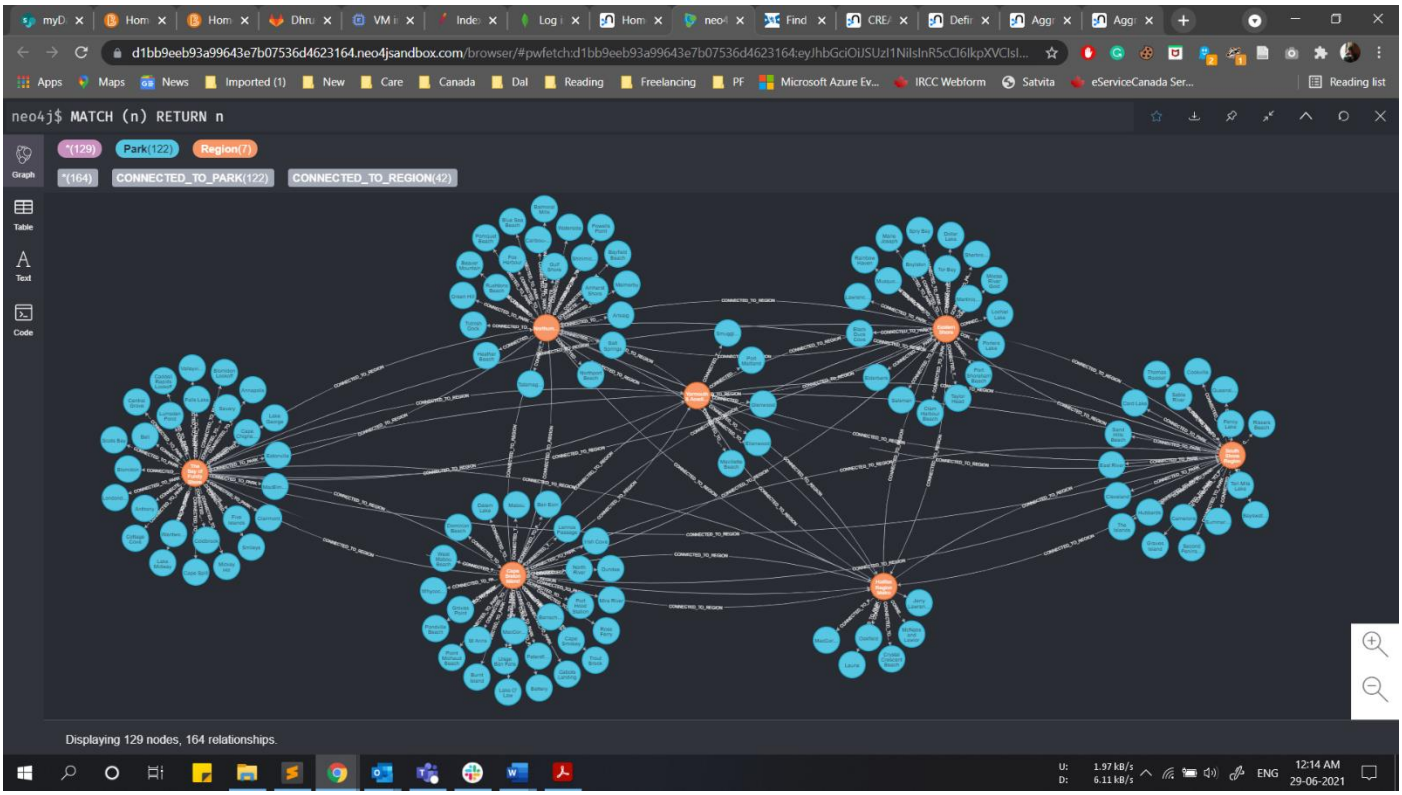

Figure 14. View the entire graph Cypher Query, created using Neo4j Sandbox [5].

NOTE: I have added the Cypher.txt file containing all the Cypher Queries used above along with the screenshots in a Neo4j folder inside the main zip folder.

References:

[1] P. o. N. Scotia, "Nova Scotia Provincial Parks," [Online]. Available: https://parks.novascotia.ca/. [Accessed 2021].

[2] "NewsAPI - Search News and Blog Articles on Web," News API, [Online]. Available: https://newsapi.org/. [Accessed 2021].

[3] ComputingforGeeks, "Install Apache Spark on Ubuntu 20.04/18.04 & Debian 10/9," Computing for Geeks, 2014. [Online]. Available: https://computingforgeeks.com/how-to-install-apache-spark-on-ubuntu-debian/. [Accessed 2021].

[4] "Spark," [Online]. Available: https://downloads.apache.org/spark/. [Accessed 2021].

[5] Neo4j, "Neo4j Sandbox," Neo4j, [Online]. Available: https://sandbox.neo4j.com/. [Accessed 2021].

[6] C. -. C. Y. Passion, "Java connecting to MongoDB database examples," CodeJava, 2012. [Online]. Available: https://www.codejava.net/java-se/jdbc/java-connecting-to-mongodb-database-examples. [Accessed 2021].

[7] L. Vogel, "Regular expressions in Java - Tutorial," Vogella, 2007. [Online]. Available: https://www.vogella.com/tutorials/JavaRegularExpressions/article.html. [Accessed 2021].

[8] Lucidchart, "Intelligent Diagramming | Lucidchart," Lucidchart, [Online]. Available: https://www.lucidchart.com/pages/. [Accessed 2021].

[9] Google, "Google Cloud Platform," Google, [Online]. Available: https://console.cloud.google.com/home/dashboard?project=csci-5408-s21-317315&pli=1. [Accessed 2021].