Report

Assignment No.: 3

Problem -1:

Problem-1.1: News Article Data Extraction & Transformation:

- I visited the news API website https://newsapi.org/ and search for the news article data to extract it.
- After collecting the news article, I wrote the java program to perform Data Extraction and Data Filteration.
- I need to extract the data by searching the keywords (case insensitive) – “Canada”, “University”, “Dalhousie University”, “Halifax”, “Canada Education”, “Moncton”, “Toronto”, “oil”, “inflation”.
- Below is the algorithm for NewsExtraction and NewsFilteration.

1. Data Extraction Algorithm:

- I made NewsExtractor() method to perform data extraction.

```
Step-1: BEGIN
Step-2: Create a String array for the search keywords.
Step-3: FOR loop to search keyword in the String [] keywords.
 Step-3.1: SET News API URL to url with query parameter “q” SET to String [] keywords.
Step-3.2: Establishing the url connection by type casting it with HttpURLConnection.
Step-3.3: SET request method to “GET” and connect [1].
Step-3.4: IF httpresponsecode is 200 means the http response is succeeded, go to Step – 3.5.
Step-3.4.1: we use StringBuilder to read each line of the data and append to the string and scanner to go through URL stream and write all the JSON data into a string [1].
Step-3.4.2: Close the scanner.
Step-3.4.3: Use the JSON simple library to parse the string into a JSON object [1].
Step-3.4.4: Cast the object to JSONArray to find specific object we want from inside the array i.e “articles”.
Step-3.4.5: Create a new file to write the data of the searched keyword using FileWriter.
 Step-3.4.6: Close the FileWriter.
 Step-3.4.7: ELSE (http response failed or not equal to null)throw new exception
 Step-3.5: END IF
Step-4: END FOR
```

Step-5: END

2. Data Filteration Algorithm:

- I performed data filteration in the class NewsFilteration().
- I used 3 method to filter out the data which is given below.
- Below is the algorithm for their respective methods:

a) readFile() :

Step-1: BEGIN

Step-2: Create List of Strings to read lines present in the data.

Step-3: Declare lines by getting the filename inorder to go through each line and read all lines in StandardCharsets.UTF_8 .

Step-4: RETURN each line of the file.

Step-5: END

b) readAndFilterNewsData():

ELEMENTS TO BE FILTERED	REGEX USED
filterForURL	"(\"url\":(\"http.*?\"),)"
filterForAuthorURL	"(\"author\":(\"http.*?\"),)"
filterForURLToImage	"(\"urlToImage\":(\"http.*?\"|null),)"
filterWhenAuthorIsNull	"(\"author\":(\"|null),)"
filterWhenIdIsNull	"(,\"id\":(null))"
filterForSpecialCharacters	"[^A-Z0-9 a-z,.(')-_+:{}\"%$]|\\r\\n"
filterForTags	"(<[^>]*>)"

Note: On removing the special characters, the emoticons are removed too.

Step-1: BEGIN

Step-2: Create a String array for the search keywords.

Step-3: FOR loop to search keyword in the String [] keywords.

Step-3.1: Create an ArrayList of strings (filteredNewsList).

Step 3.2: Create an ArrayList of Strings (list)to read the file by providing the file path. This will read the existing file that was created during the data extraction process.

Step-3.2: Use REGEX to filter the data for URL, Special Characters, Emoticons, Null values.

(See the above REGEX table)

Step-3.3: FOR each of String list_element and list to traverse each line and filter data using REGEX.

Step-3.3.1: Replace the filtered data with “” by using replaceAll. Because we need to clean the data.

Step-3.3.2: Store the filter data in the list_element and call the same list and filter for the the next element. This will filter the data and append the filter data to the existing data.

Step-3.3.3: Replace the final filtered data having stringFilterSeparator with STRING_FILTER_SEPARATOR_REPLACEMENT. This will separate the data present in the array with the other list of Arrays.

Step-3.3.4: After filteration, ADD the filtered list_element to the ArrayList of String (filteresNewsList).

Step-3.4: END FOR

Step-3.5: Store filteredNewsList in String listDisplay.

Step-3.6: Create File to store the filtered data.

Step-3.7: Write the updated data using FileWriter

Step -3.8: Close FileWriter.

Step-3.9: Call the method **uploadNewsDataToMongo** and pass filteredNewsList to print all the filtered data in mongoDB.

Step-4: END FOR

Step-5: END

c) uploadNewsDataToMongoDB():

Step-1: BEGIN

Step-2: Create a method by passing the parameter ArrayList of string (filteredNewsList).

Step-3: Make a connection with MongoDB using MongoClient.

Step-4: Make a connection with the database created i.e. myMongoNews using MongoDatabase.

Step-5: Make a connection with the collection created in database i.e newsDataCollection.

Step-6: FOR each to call filteredNewsList and store data in each array.

Step-6.1: FOR each to parse the array in the mondoDB using the documents.

Step-6.2: END FOR

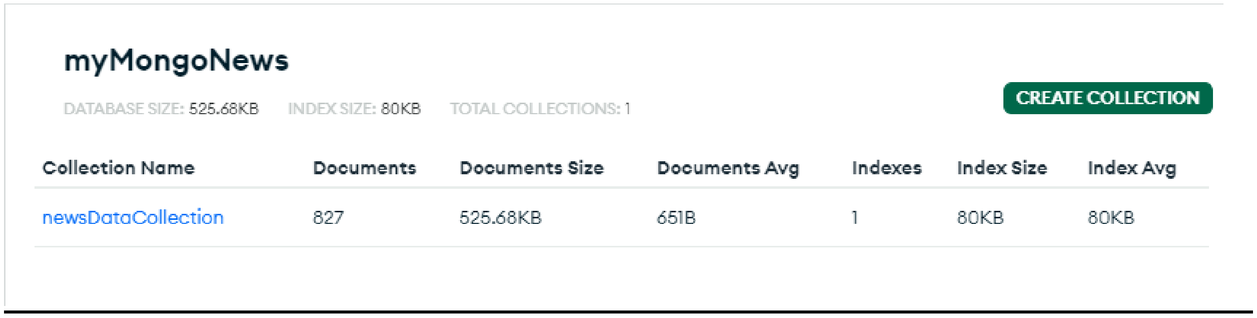
Step-7: ENDFOR

Step-8: Close mongoClient

Step-9: END

<u>Output of MongoDB:</u>

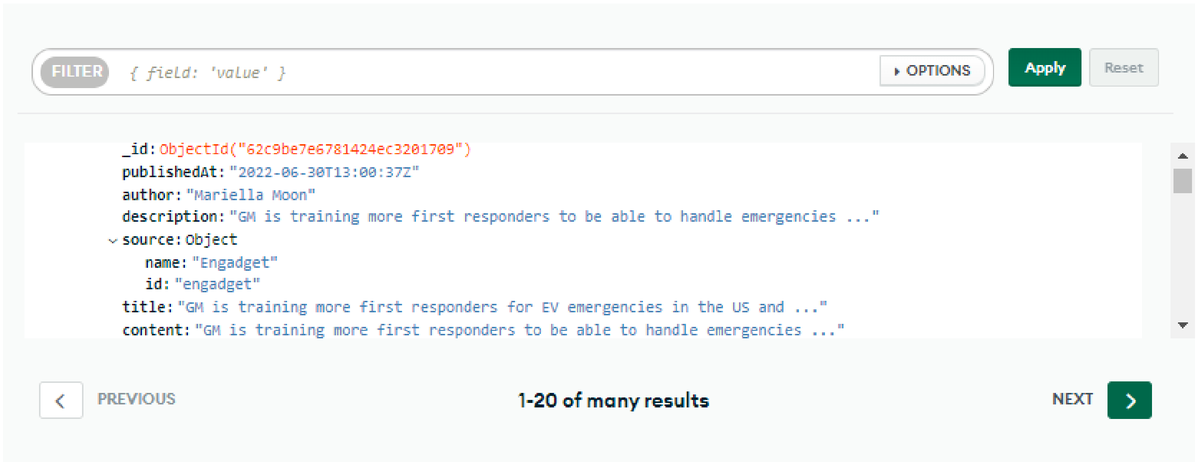
- From the below output images, we can see the URL, Special Characters, Emoticons, Tags, URL to Image is been filtered out.



myMongoNews

DATABASE SIZE: 525.68KB INDEX SIZE: 80KB TOTAL COLLECTIONS: 1

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
newsDataCollection	827	525.68KB	651B	1	80KB	80KB



```
_id: ObjectId("62c9be7e6781424ec3201709")
publishedAt: "2022-06-30T13:00:37Z"
author: "Mariella Moon"
description: "GM is training more first responders to be able to handle emergencies ..."
source: Object
  name: "Engadget"
  id: "engadget"
title: "GM is training more first responders for EV emergencies in the US and ..."
content: "GM is training more first responders to be able to handle emergencies ..."
```

<u>Problem-1.2: Movie Data Extraction & Transformation:</u>

- I visited the movies API website http://www.omdbapi.com/ and search for the news article data to extract it.
- After collecting the news article, I wrote the java program to perform Data Extraction and Data Filteration.
- I need to extract the data by searching the keywords (case insensitive) – "Canada", "University",", "Moncton", "Halifax", "Toronto", ""Vancouver", "Alberta", "Niagara".

1. <u>Data Extraction Algorithm:</u>

- I made MovieDataExtractor() method to perform data extraction.

Step-1: BEGIN

Step-2: Create a String array for the search keywords.

Step-3: FOR loop to search keyword in the String [] keywords.

Step-3.1: SET News API URL to url with query parameter "q" SET to String [] keywords.

Step-3.2: Establishing the url connection by type casting it with HttpURLConnection.

Step-3.3: SET request method to "GET" and connect [1].

Step-3.4: IF httpresponsecode is 200 which means if http response suceeds, go to Step-3.4.1.

Step-3.4.1: we use StringBuilder to read each line of the data and append to the string and scanner to go through URL stream and write all the JSON data into a string [1].

Step-3.4.2: Close the scanner.

Step-3.4.3: Create a new file to write the data of the searched keyword using FileWriter.

Step-3.4.4: Close the FileWriter.

Step-3.4.5: ELSE(means http response fails) throw new exception.

Step-3.5: END IF.

Step-4: END FOR.

Step-5: END

2. <u>Data Filteration Algorithm:</u>

- I performed data filteration in the class MovieDbFilteration()
- I used 3 method to filter out the data which is given below.
- Below is the algorithm for their respective methods:

a) readFile() :

Step-1: BEGIN

Step-2: Create List of Strings to read lines present in the data.

Step-3: Declare lines by getting the filename inorder to go through each line and read all lines in StandardCharsets.UTF_8 .

Step-4: RETURN each lines of the file .

Step-5: END

b) readAndFilterMovieData():

Elements to be filled	Regex used
filterForURL	(\"Poster\":(\"http.*?\"),)

Step-1: BEGIN

Step-2: Create a String array for the search keywords.

Step-3: FOR loop to search keyword in the String [] keywords.

Step-3.1: Create an ArrayList of strings (filteredMoviesList)

Step 3.2: Create an ArrayList of Strings (list)to read the file by providing the file path. This will read the existing file that was created during the data extraction process.

Step-3.3: Use REGEX to filter the data for URL, Special Characters, Emoticons, Null values.

(See the above REGEX table)

Step-3.4: FOR each of String list_element and list to traverse each line and filter data using REGEX.

Step-3.4.1: Replace the filtered data with “” by using replaceAll. This will clean the data.

Step-3.4.2: Store the filter data in the list_element and call the same list and filter for the the next element.

Step-3.4.3: Replace the final filtered data having stringFilterSeparator with STRING_FILTER_SEPARATOR_REPLACEMENT. This will separate the data present in the array with the other list of Arrays.

Step-3.4.4: After filteration, ADD the filtered list_element to the ArrayList of String (filteredMovieslist).

Step-3.5: ENDFOR

Step-3.6: Store **filteredMoviesList** in String listDisplay.

Step-3.7: Create File to store the filtered data.

Step-3.8: Write the updated data using FileWriter

Step -3.9: Close FileWriter.

Step-3.10: Call the method uploadMovieDataToMongo and pass filteredMoviesList to print all the filtered data in mongoDB.

Step-4: END FOR

Step-5: END

c) uploadMovieDataToMongoDB():

Step-1: BEGIN

Step-2: Create a method by passing the parameter ArrayList of string (filteredMovieList).

Step-3: Make a connection with MongoDB using MongoClient.

Step-4: Make a connection with the database created i.e. myMongoMovieDb using MongoDatabase.

Step-5: Make a connection with the collection created in database i.e movieDataCollection.

Step-6: FOR each to call filteredMovieList and store data in each array.

Step-6.1: FOR each to parse the array in the mondoDB using the documents.

Step-6.2: END FOR.

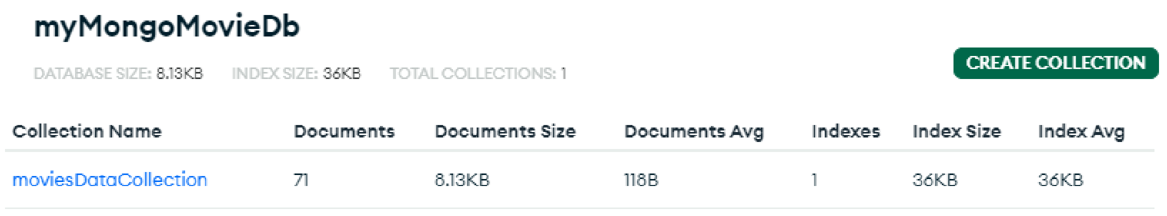
Step-7: END FOR.

Step-8: Close mongoClient.

Step-9: END

<u>Output of MongoDB:</u>

- From the below output images, we can see the URL, Special Characters, Emoticons, Tags, URL to Image is been filtered out.

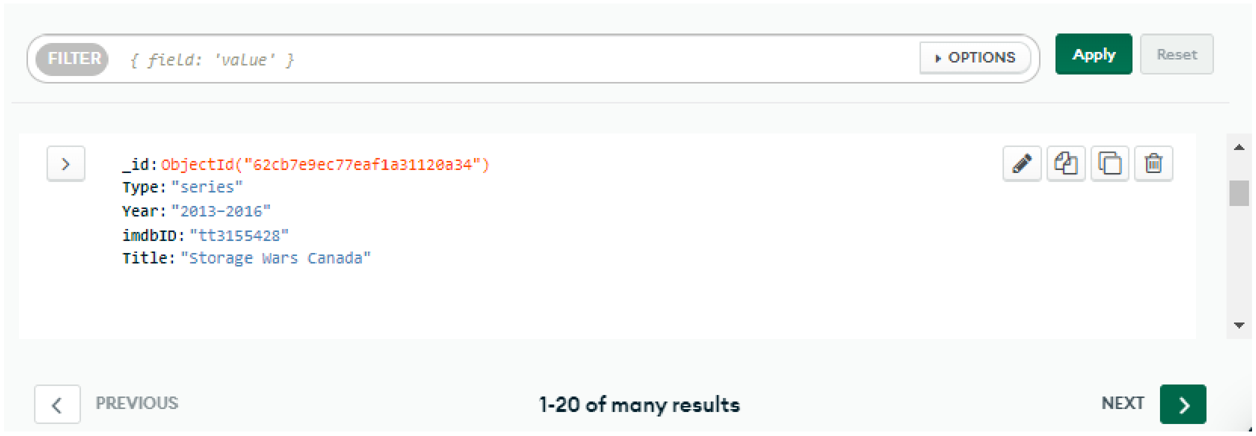


myMongoMovieDb

DATABASE SIZE: 8.13KB INDEX SIZE: 36KB TOTAL COLLECTIONS: 1

CREATE COLLECTION

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
moviesDataCollection	71	8.13KB	118B	1	36KB	36KB



FILTER { field: 'value' } OPTIONS Apply Reset

```
_id: ObjectId("62cb7e9ec77eaf1a31120a34")
Type: "series"
Year: "2013-2016"
imdbID: "tt3155428"
Title: "Storage Wars Canada"
```

PREVIOUS 1-20 of many results NEXT

<u>Problem-1.3: Reuter News Data Reading & Transformation:</u>

- I used the given news files(reut2-009.sgm and reut2-014.sgm)
- I wrote a java program to get the get the text present between the <REUTERS></REUTERS> tags, <TEXT></TEXT> and <TITLE></TITLE> tags in the above news files.
- I performed the operation in class ReuterRead().
- Below is the Algorithm for their respective methods:

a) extractFilename():

Step-1: BEGIN

Step-2: Create a string array of file names.

Step-3: FOR to read 2 files.

Step-4: END FOR

Step-5: END

b) readReuterFiles():

```
Step-1: BEGIN
Step-2: SET file path to read the files
Step-3: SET BufferedReader to read the files from given path.
Step-4: Declare reuterString using StringBuilder.
Step-5: WHILE the line in the given file initialized as “reuterLine” is not null, read each line of the file.
  Step-5.1: IF reuterLine becomes </REUTERS> which is equal to -1 go to step-5.1.1.
    Step-5.1.1: APPEND reuterString with that reuterLine.
    Step-5.1.2: APPEND reuterString with (‘ ‘).
  Step-5.2: ELSE  SET title to null, go to step-5.2.1
    Step-5.2.1: SET Pattern to check for <TITLE></TITLE>. This will just read the text between the title tags.
    Step-5.2.2: Use Matcher to match the tag in reuterString.
    Step-5.2.3: WHILE we find the match, go to step-5.2.3.1
      Step-5.2.3.1: IF matcher.group(1) is not null go to step-5.2.3.1.1
        Step-5.2.3.1.1: match <TITLE>
      Step-5.2.3.2: END IF
    Step-5.2.4: END WHILE
    Step-5.2.5: SET body to null
    Step-5.2.6: SET Pattern to check for <BODY></BODY>. This will just read the text between the Body tags.
    Step-5.2.7: Use Matcher to match the tag in reuterString.
      Step-5.2.7.1:  IF matcher.group(1) is not null, go to step-5.2.7.1.1
        Step-5.2.7.1.1: match <BODY>
      Step-5.2.7.2: END IF
    Step-5.2.5: END WHILE
    Step-5.2.6: Call the function uploadReuterDataToMongoDB to get the required data in mongoDB cloud.
  Step-5.2: END IF
Step-6: END WHILE
```

Step-7: END

c) uploadReutersToMongoDB():

Step-1 BEGIN

Step-2: Create a method by passing the parameter outputStringTitle and outputStringBody.

Step-3: Make a connection with MongoDB using MongoClient.

Step-4: Make a connection with the database created i.e. ReuterDB using MongoDatabase.

Step-5: Make a connection with the collection created in database i.e reuterDataCollection.

Step-6: CREATE Hashmap<String, String> mapTitleAndBody to map title and body.

Step-7: APPEND the outStringTitle to document.

Step-8: APPEND the outputStringBody i.e. TEXT to document.

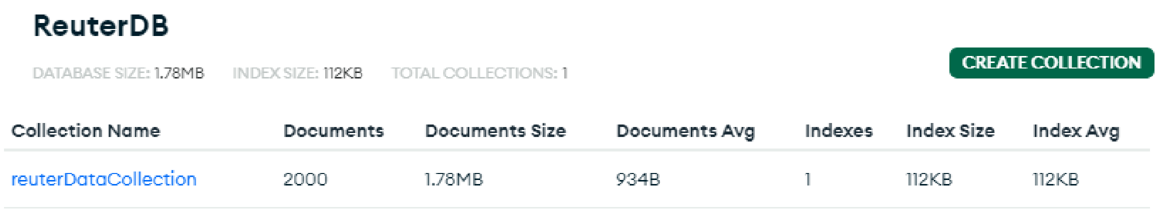
Step-9: INSERT the document in mongoCollection.

Step-10: Close mongoClient

Step-11: END.

<u>Output of MongoDB:</u>

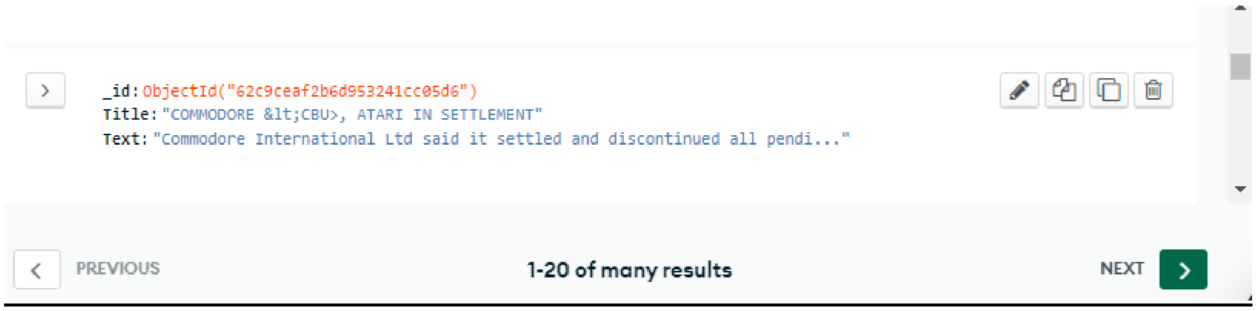
- From the below below output image, we can only see the texts between the tags <REUTES></REUTERS>, <Title></Title> and <Text></Text>.



ReuterDB

DATABASE SIZE: 1.78MB INDEX SIZE: 112KB TOTAL COLLECTIONS: 1

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
reuterDataCollection	2000	1.78MB	934B	1	112KB	112KB



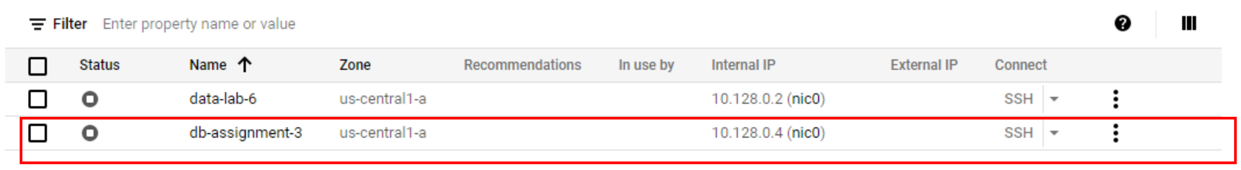
```
_id: ObjectId("62c9ceaf2b6d953241cc05d6")
Title: "COMMODORE &lt;CBU>, ATARI IN SETTLEMENT"
Text: "Commodore International Ltd said it settled and discontinued all pendi..."
```

Problem -2:

Problem-2.1: Data Processing using Spark – MapReduce to perform count:

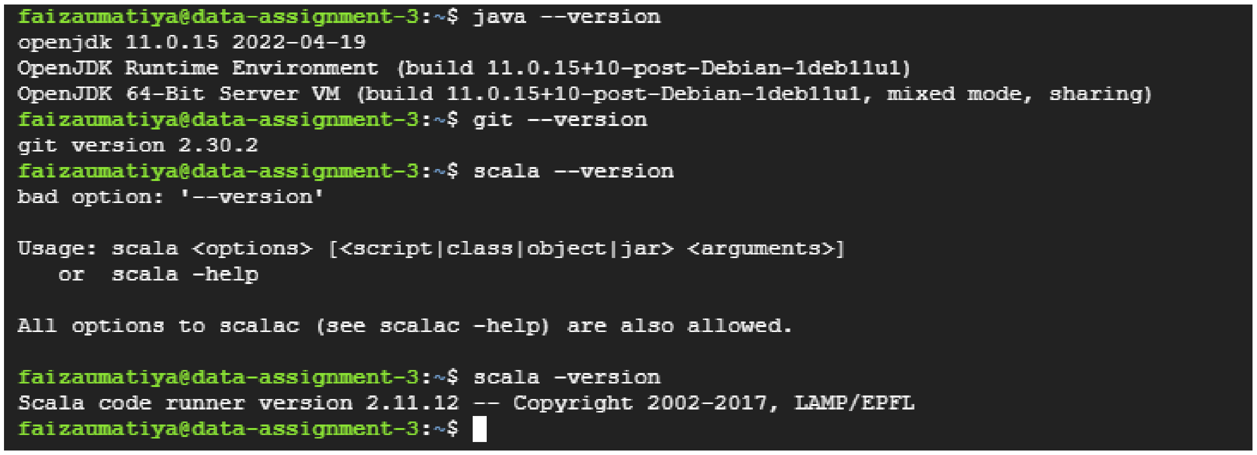
Configuration and Initialization of Apache Spark Cluster:

1. Installing Apache Spark on GCP:
- Created new Compute Engine instance on GCP (VM) named – db-assignment-3
2. Configure VM:
- Region – us-central1 (Iowa)
- Zone – us-central1-a
- Series – E2
- Machine type- e2 medium (2 VCPU , 4 GB memory)
- Boot disk- Ubuntu 20.04 LTS
- Firewall- Allow HTTP traffic and Allow HTTPS traffic.
3. Connect to VM:



	Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
☐	●	data-lab-6	us-central1-a			10.128.0.2 (nic0)		SSH
☐	●	db-assignment-3	us-central1-a			10.128.0.4 (nic0)		SSH

4. Open db-assignment-3 in SSH browser.
5. Check the versions after opening bash from SSH (VM instances) as shown below:



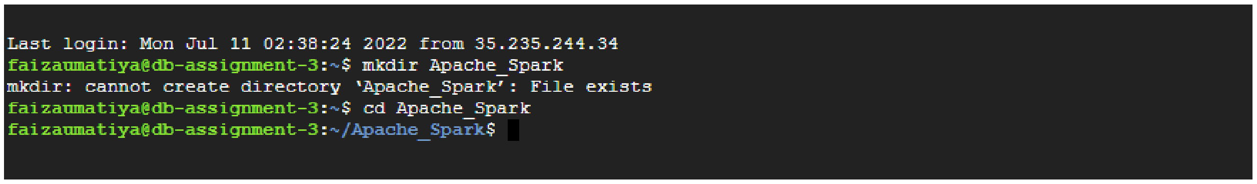
```
faizaumatiya@data-assignment-3:~$ java --version
openjdk 11.0.15 2022-04-19
OpenJDK Runtime Environment (build 11.0.15+10-post-Debian-1deb11u1)
OpenJDK 64-Bit Server VM (build 11.0.15+10-post-Debian-1deb11u1, mixed mode, sharing)
faizaumatiya@data-assignment-3:~$ git --version
git version 2.30.2
faizaumatiya@data-assignment-3:~$ scala --version
bad option: '--version'

Usage: scala <options> [<script|class|object|jar> <arguments>]
   or  scala -help

All options to scalac (see scalac -help) are also allowed.

faizaumatiya@data-assignment-3:~$ scala -version
Scala code runner version 2.11.12 -- Copyright 2002-2017, LAMP/EPFL
faizaumatiya@data-assignment-3:~$
```

6. Installed Apache Spark Binary:
- mkdir Apache_Spark
- cd Apache_Spark



```
Last login: Mon Jul 11 02:38:24 2022 from 35.235.244.34
faizaumatiya@db-assignment-3:~$ mkdir Apache_Spark
mkdir: cannot create directory 'Apache_Spark': File exists
faizaumatiya@db-assignment-3:~$ cd Apache_Spark
faizaumatiya@db-assignment-3:~/Apache_Spark$
```

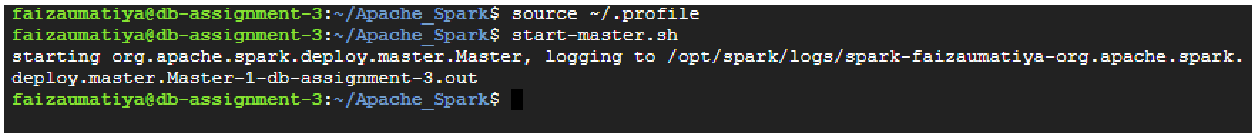
7. Configure Path:

- echo “export SPARK_HOME=/opt/spark” >> ~/.profile
- echo “export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin” >> ~/.profile
- echo “export PYSPARK_PYTHON=/usr/bin/python3” >> ~/.profile
- source ~/.profile

8. Configure Permanent Path:

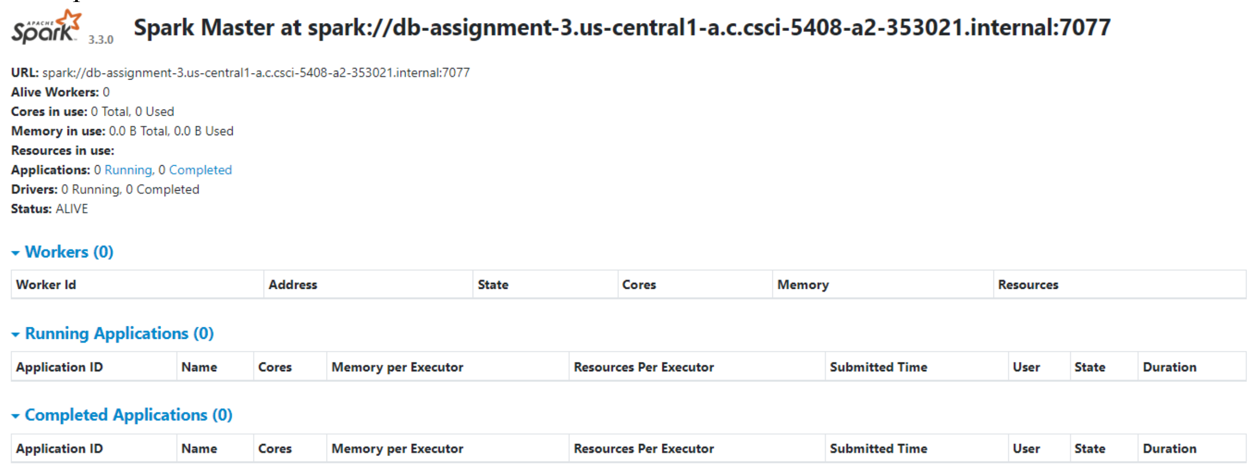
- sudo nano ~/.profile
- export SPARK_HOME=/opt/spark
- export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
- export PYSPARK_PYTHON=/usr/bin/python3
- Save the file and exit
- source ~/.profile

Start master node:



```
faizaumatiya@db-assignment-3:~/Apache_Spark$ source ~/.profile
faizaumatiya@db-assignment-3:~/Apache_Spark$ start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spark-faizaumatiya-org.apache.spark.
deploy.master.Master-1-db-assignment-3.out
faizaumatiya@db-assignment-3:~/Apache_Spark$
```

VM’s public IP address:8080:



3.3.0 **Spark Master at spark://db-assignment-3.us-central1-a.c.csci-5408-a2-353021.internal:7077**

URL: spark://db-assignment-3.us-central1-a.c.csci-5408-a2-353021.internal:7077
Alive Workers: 0
Cores in use: 0 Total, 0 Used
Memory in use: 0.0 B Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (0)

Worker Id	Address	State	Cores	Memory	Resources

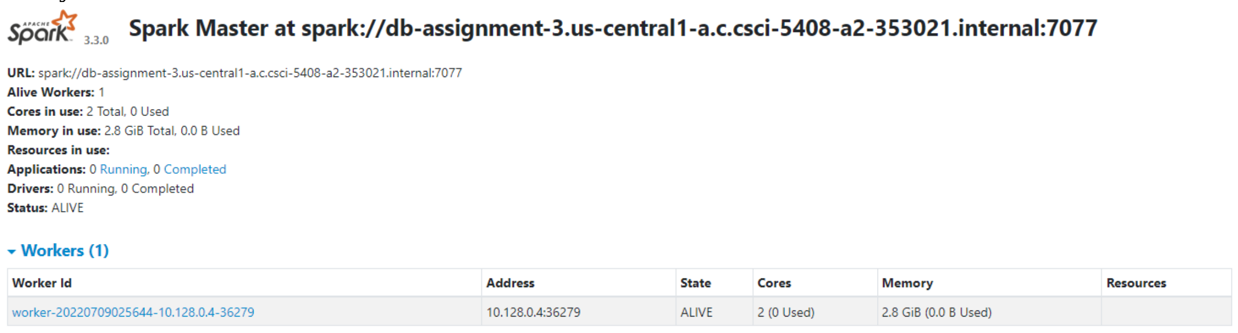
Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Start slave node:
Verify the worker node:

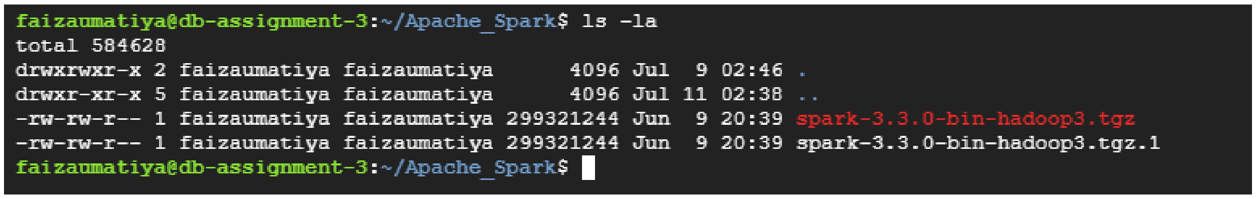


3.3.0 **Spark Master at spark://db-assignment-3.us-central1-a.c.csci-5408-a2-353021.internal:7077**

URL: spark://db-assignment-3.us-central1-a.c.csci-5408-a2-353021.internal:7077
Alive Workers: 1
Cores in use: 2 Total, 0 Used
Memory in use: 2.8 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20220709025644-10.128.0.4-36279	10.128.0.4:36279	ALIVE	2 (0 Used)	2.8 GiB (0.0 B Used)	



```
faizaumatiya@db-assignment-3:~/Apache_Spark$ ls -la
total 584628
drwxrwxr-x 2 faizaumatiya faizaumatiya      4096 Jul  9 02:46 .
drwxr-xr-x 5 faizaumatiya faizaumatiya      4096 Jul 11 02:38 ..
-rw-rw-r-- 1 faizaumatiya faizaumatiya 299321244 Jun  9 20:39 spark-3.3.0-bin-hadoop3.tgz
-rw-rw-r-- 1 faizaumatiya faizaumatiya 299321244 Jun  9 20:39 spark-3.3.0-bin-hadoop3.tgz.1
faizaumatiya@db-assignment-3:~/Apache_Spark$
```

Word Counter Engine:

- I wrote a MapReduce program in java to count the frequency of words of the keywords: “Canada”, “University”, “Dalhousie University”, “Halifax”, “Canada Education”, “Moncton”, “Toronto”, “oil”, “inflation”.
- I performed the operation in WordCounterEngine Class.
- I used 2 methods to perform the operation whose algorithm is given below:

a) ReadNewsFile():

```
Step-1: BEGIN
Step-2: Create List of Strings to read lines present in the data.
Step-3: Declare lines by getting the filename inorder to go through each line and read all lines in StandardCharsets.UTF_8 .
Step-4: RETURN each lines of the file.
Step-5: END
```

b) readAndWordCount():

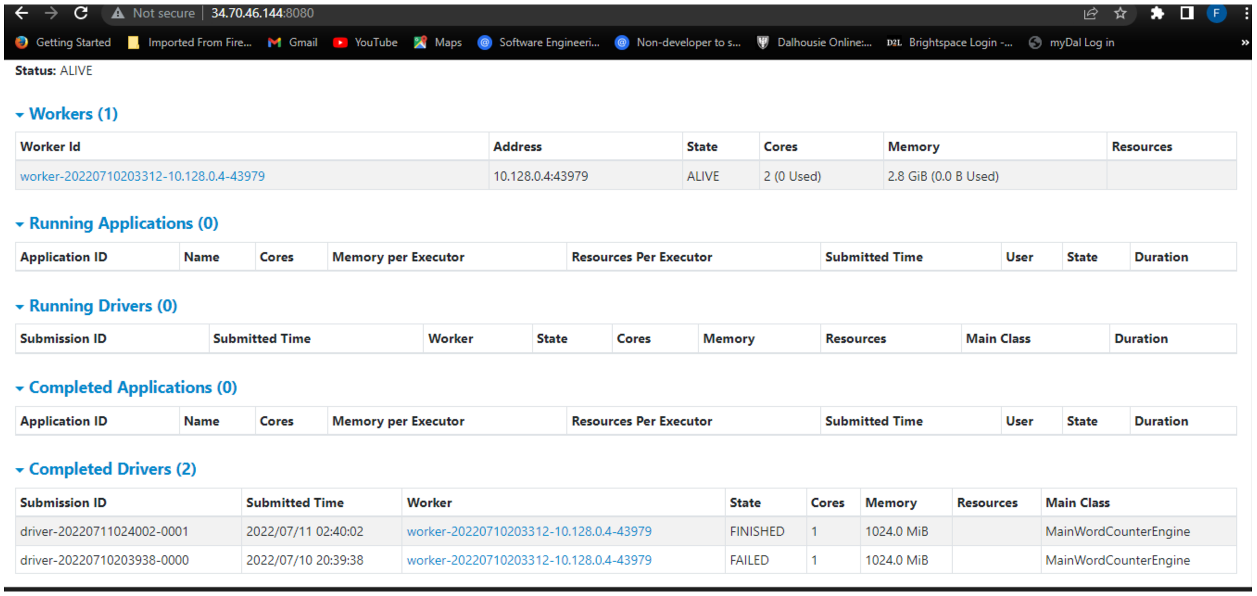
```
Step-1: BEGIN
Step-2: Create a String array for the search keywords.
Step-3: Create ArrayList of Hashmap<String, integer> eachline to look for the keyword in each line of each file.
Step-4: FOR loop to search keyword in the String [] keywords.
Step-4.1: SET file path to list of String i.e. listOfFiles to read the existing file.
Step-4.2:  Create a string to store the data of listOfFiles in string format.
Step-4.3: Create Hashmap called wordCountMap to map the keyword and its occurences in each file.
Step-4.4: FOR to traverse each line of each file using the keywords.length.
Step-4.4.1: SET pattern to search for specific keywords[k] it can be “Canada”, “oil”, “Toronto” etc from the given string [] keywords.
Step-4.4.2: SET Matcher to match the keyword in each existing file.
Step-4.4.3: WHILE we find the match.
Step-4.4.3.1: IF wordCountMap contains the keyword, go to step 4.4.3.1.1.
```

```
Step-4.4.3.1.1: Store the keyword with its occurrence in hashmap wordCountMap and increment the ocuurence by 1.
Step-4.4.3.2: ELSE Store the keyword and its 1 occurrence in hashmap wordCountMap.
Step-4.4.3.3: END IF
Step-4.4.4: END WHILE
Step-4.5: END FOR
Step-4.6: Create a HashMap to “map” the key pairs by CALLING reduce function.
Step-4.6.1: FOR traverse each file using the given keyword by its length of Array i.e. keywords.length.
Step-4.6.1.1: IF the keywords present in the file is null, go to 4.6.1.1.1
Step-4.6.1.1.1: PRINT 0
Step-4.6.1.2: ELSE PRINT the frequency or the no. of  occurences of the keywords.
Step-4.6.3: END IF
Step-5: END FOR
Step-6: STOP
```

c) reduce():

```
Step-1: START
Step-2: Create a HashMap<String, Integer> finalMap
Step-3: FOR each to iterate maps with the listOfMap to process the data that comes from map.
Step-4: Merge the map using the merge function by merging old value and new value of the map.
Step-5: END FOR
Step-5: RETURN finalMap.
Step-6: END
```

- After the MapReduce program and VM instance Configuration, I created JAR file and uploaded to the SSH browser.
- After uploading the jar file, I ran the below command:
- `spark-submit --master spark://34.70.46.144:7077 --deploy-mode cluster --class MainWordCounterEngine Problem2_1-1.0-SNAPSHOT.jar`
- On running the above command, I get the below output:



Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20220710203312-10.128.0.4-43979	10.128.0.4:43979	ALIVE	2 (0 Used)	2.8 GiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Running Drivers (0)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class	Duration

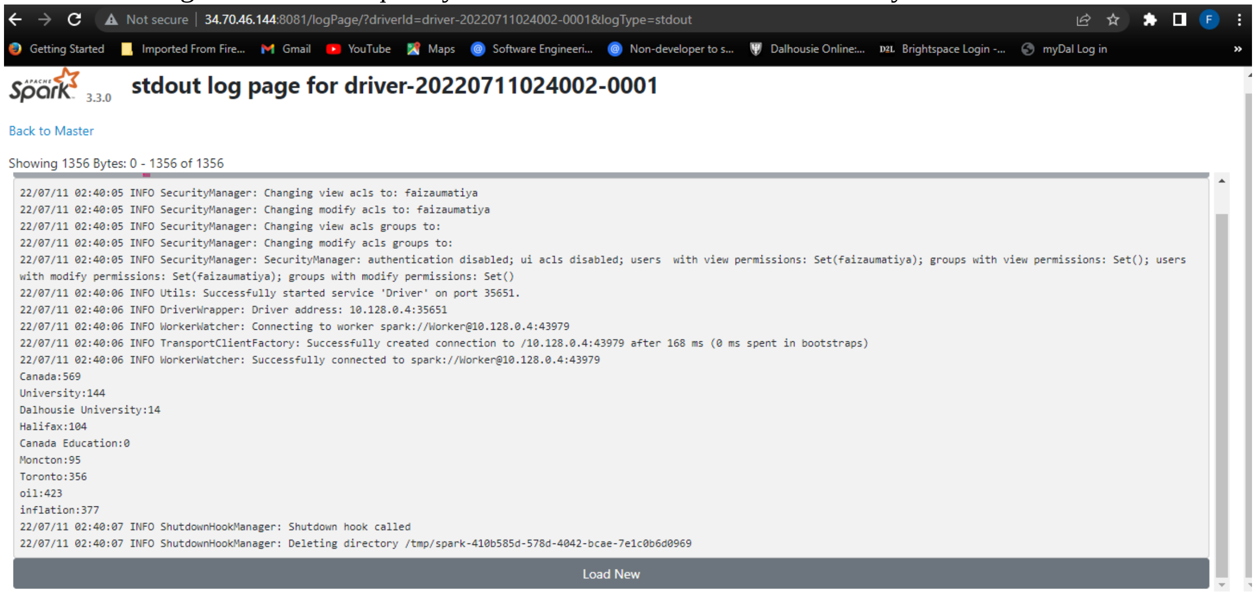
Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Drivers (2)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Resources	Main Class
driver-20220711024002-0001	2022/07/11 02:40:02	worker-20220710203312-10.128.0.4-43979	FINISHED	1	1024.0 MiB		MainWordCounterEngine
driver-20220710203938-0000	2022/07/10 20:39:38	worker-20220710203312-10.128.0.4-43979	FAILED	1	1024.0 MiB		MainWordCounterEngine

The below image shows the frequency of the word count of the searched keywords in all the files.



stdout log page for driver-20220711024002-0001

Showing 1356 Bytes: 0 - 1356 of 1356

```
22/07/11 02:40:05 INFO SecurityManager: Changing view acls to: faizaumatiya
22/07/11 02:40:05 INFO SecurityManager: Changing modify acls to: faizaumatiya
22/07/11 02:40:05 INFO SecurityManager: Changing view acls groups to:
22/07/11 02:40:05 INFO SecurityManager: Changing modify acls groups to:
22/07/11 02:40:05 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: Set(faizaumatiya); groups with view permissions: Set(); users
with modify permissions: Set(faizaumatiya); groups with modify permissions: Set()
22/07/11 02:40:06 INFO Utils: Successfully started service 'Driver' on port 35651.
22/07/11 02:40:06 INFO DriverWrapper: Driver address: 10.128.0.4:35651
22/07/11 02:40:06 INFO WorkerWatcher: Connecting to worker spark://Worker@10.128.0.4:43979
22/07/11 02:40:06 INFO TransportClientFactory: Successfully created connection to /10.128.0.4:43979 after 168 ms (0 ms spent in bootstraps)
22/07/11 02:40:06 INFO WorkerWatcher: Successfully connected to spark://Worker@10.128.0.4:43979
Canada:569
University:144
Dalhousie University:14
Halifax:104
Canada Education:0
Moncton:95
Toronto:356
oil:423
inflation:377
22/07/11 02:40:07 INFO ShutdownHookManager: Shutdown hook called
22/07/11 02:40:07 INFO ShutdownHookManager: Deleting directory /tmp/spark-410b585d-578d-4042-bcae-7e1c0b6d0969
```

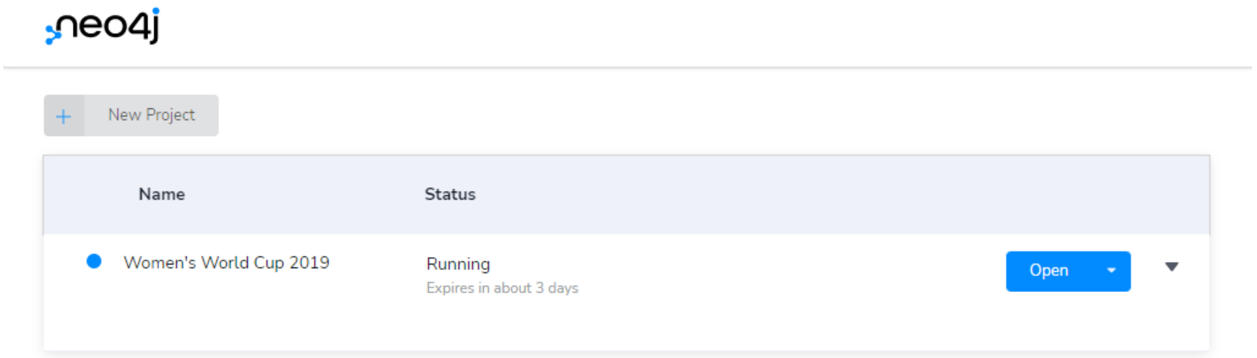
The above output shows the highest frequency and lowest frequency which is given in the following table:

Table shows word with its highest and lowest frequency:

Word	Frequency
Canada	569 (Highest Frequency)
Canada Education	0 (Lowest Frequency)

Problem-2.2: Neo4J Database:

- After practicing the neo4js during labs and assignments I understood that Neo4J lets us retrieve the data from the graph.
- From the below examples, i.e. Fig: 2 I can can build the graph based on the relationship between tournament and team. But if my focus is to retrieve just the data of the team named "Equitorial Guinea", graph let's you get that specific data.
- We can use various CRUD operations on Neo4js: Create, Read, Update, Delete.
- The nodes are represented as circular/oval shape and relationship is represented as arrow directed.
- The dataset I chose is Women's World Cup 2019.



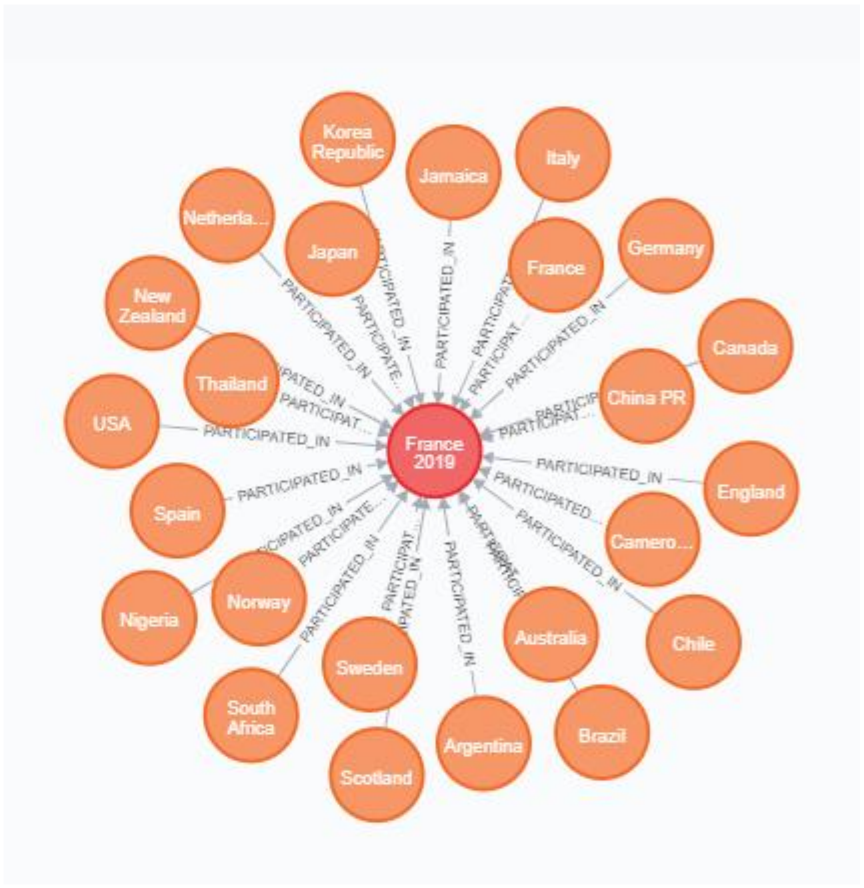
Name	Status
Women's World Cup 2019	Running Expires in about 3 days

- Below are the queries that I practiced:

Question: Who's playing in World Cup Tournament 2019?

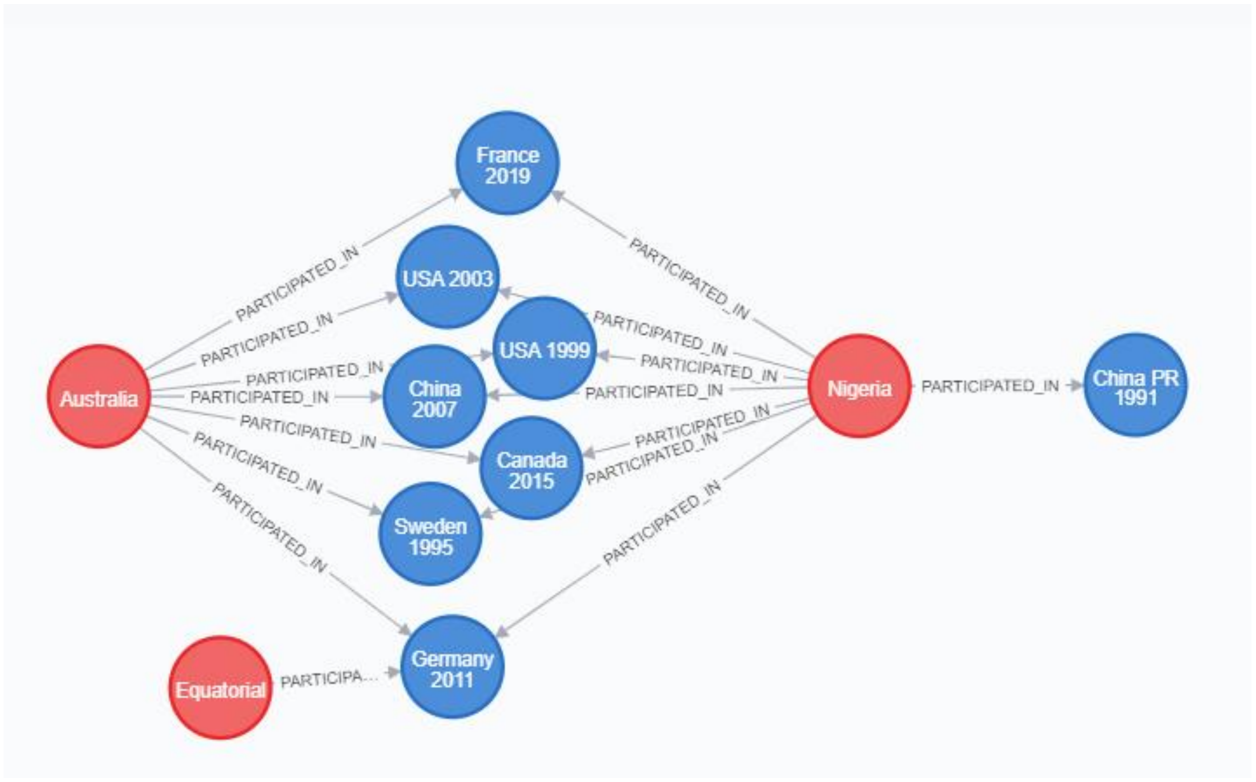
Query 1:

MATCH path = (t:Tournament {year: 2019})<-[:PARTICIPATED_IN]-(team) RETURN path



Query 2:

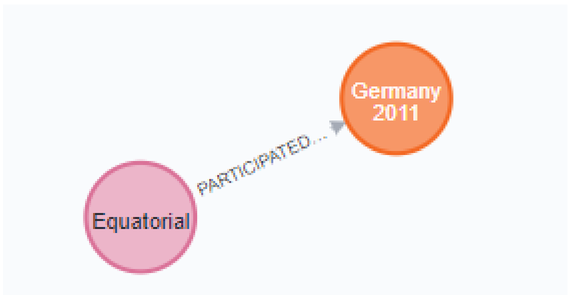
```
MATCH p=()-[r:PARTICIPATED_IN]->() RETURN p LIMIT 10
```



If I just want to get the get the name of a particular team, for instance, I want just “Equatorial Guinea”, I can write the query as,

<u>Query 3:</u>

MATCH p=(a:Team)-[r:PARTICIPATED_IN]->(b:Tournament) Where a.name="Equatorial Guinea" RETURN p LIMIT 20



References:

[1] G. Kuruppu, “Getting JSON data from a RESTful API using JAVA,” *The Startup*, 02-Aug-2020. [Online]. Available: https://medium.com/swlh/getting-json-data-from-a-restful-api-using-java-b327aafb3751. [Accessed: 7-Jul-2022].

[2] J. Paul, “How to read a text file into ArrayList in Java? Examples,” *Java67.com*. [Online]. Available: https://www.java67.com/2016/07/how-to-read-text-file-into-arraylist-in-java.html. [Accessed: 8-Jul-2022].

[3] “Java - FileWriter Class,” *Tutorialspoint.com*. [Online]. Available: https://www.tutorialspoint.com/java/java_filewriter_class.htm. [Accessed: 8-Jul-2022].

[4] “How to Remove Special Characters from string in Java,” *www.javatpoint.com*. [Online]. Available: https://www.javatpoint.com/how-to-remove-special-characters-from-string-in-java. [Accessed: 9-Jul-2022].

[5] “Cloud: MongoDB cloud,” Mongodb.com. [Online]. Available: https://cloud.mongodb.com/v2/62c0d998b2d2eb715e565b55#clusters/connect?clusterId=Cluster0. [Accessed: 11-Jul-2022].

[6] “Cypher query language,” *Neo4j Graph Data Platform*. [Online]. Available: https://neo4j.com/developer/cypher/. [Accessed: 12-Jul-2022].

[7] “Home - Neo4j sandbox,” *Neo4j.com*. [Online]. Available: https://sandbox.neo4j.com/. [Accessed: 12-Jul-2022].

[8] Brightspace, “Lab_7_Neo4j,” *Brightspace*. [Online]. Available: https://dal.brightspace.com/d2l/le/content/221749/viewContent/3051777/View. [Accessed: 7- Jul - 2022].

[9] Brightspace, “Lab6,” *Brightspace*. [Online]. Available: https://dal.brightspace.com/d2l/le/content/221749/viewContent/3049202/View. [Accessed: 6-Jul - 2022].

[10] GitHub, “read-open-source-code,” *github.com*. [Online]. Available: https://github.com/yintaoxue/read-open-source-code/blob/master/lucene-4.7.2/src/org/apache/lucene/benchmark/utils/ExtractReuters.java. [Accessed: 11- Jul – 2022].

[11] Google, “Google Cloud Platform,” *Google*. [Online]. Available: https://console.cloud.google.com/welcome?project=csci-5408-a2-353021. [Accessed 5- Jul - 2022].

GitLab Link:

https://git.cs.dal.ca/umatiya/csci5408_s22_a3_faiza_umatiya_b00899642