DALHOUSIE UNIVERSITY

# CSCI 5408 – Data Management, Warehousing, Analytics

## Assignment – 3

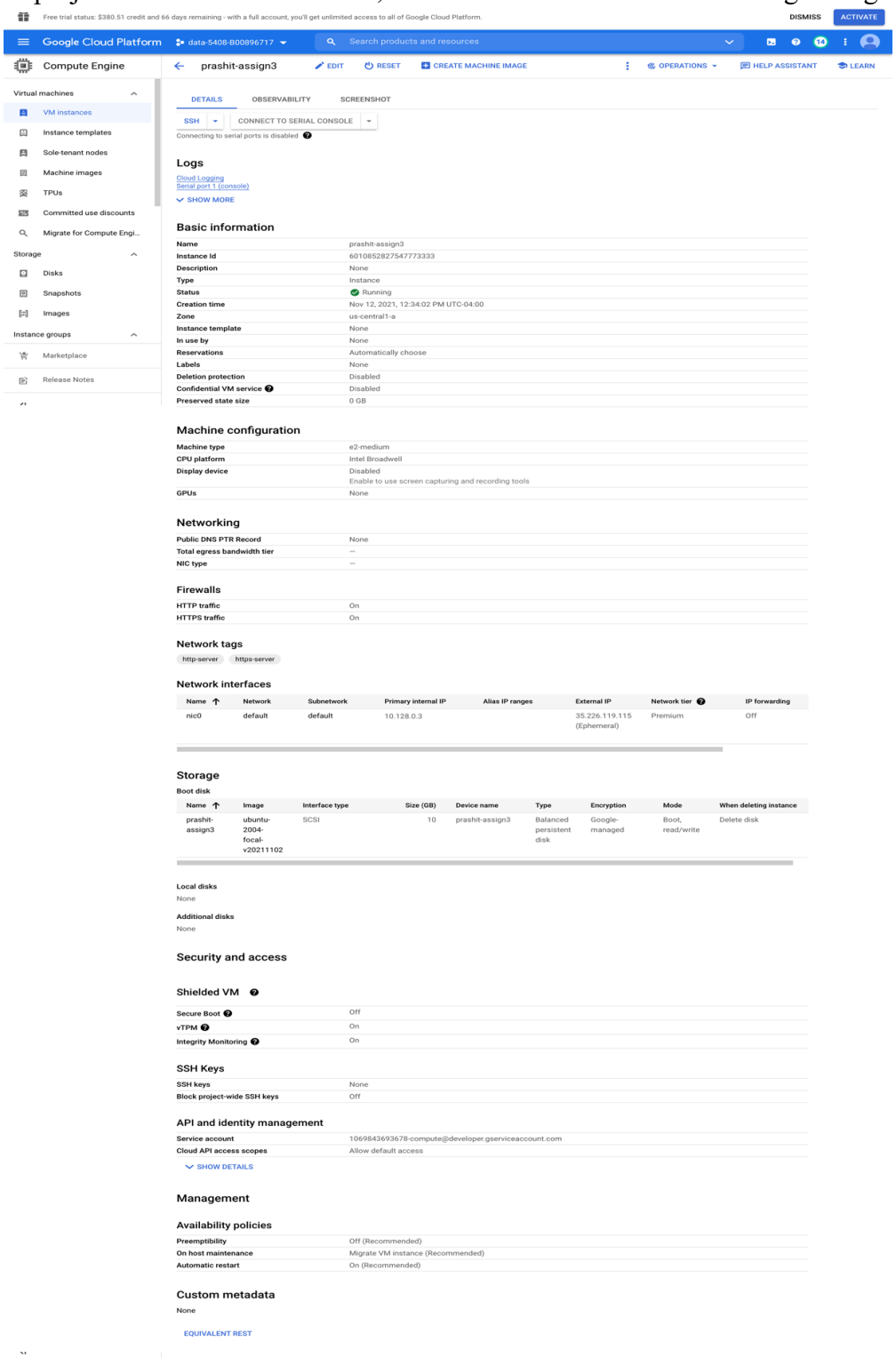
**Prashit Patel - B00896717**

**Gitlab Repository**
https://git.cs.dal.ca/pppatel/csci-5408-f2021-b00896717-prashit_patel.git

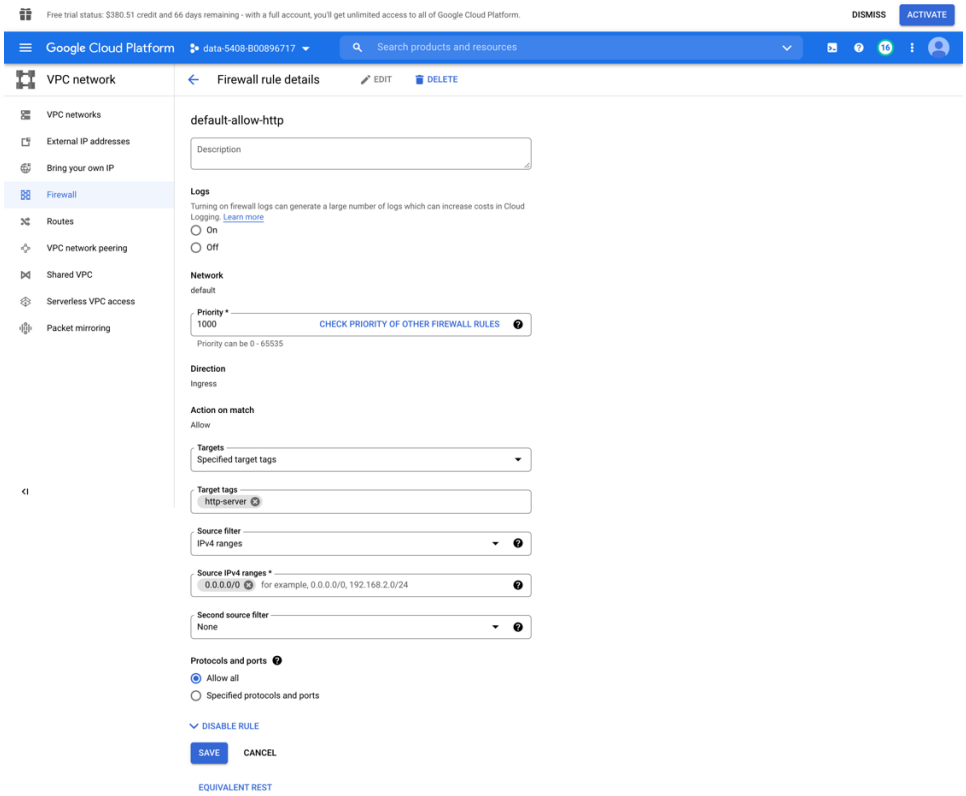
# Problem 1:

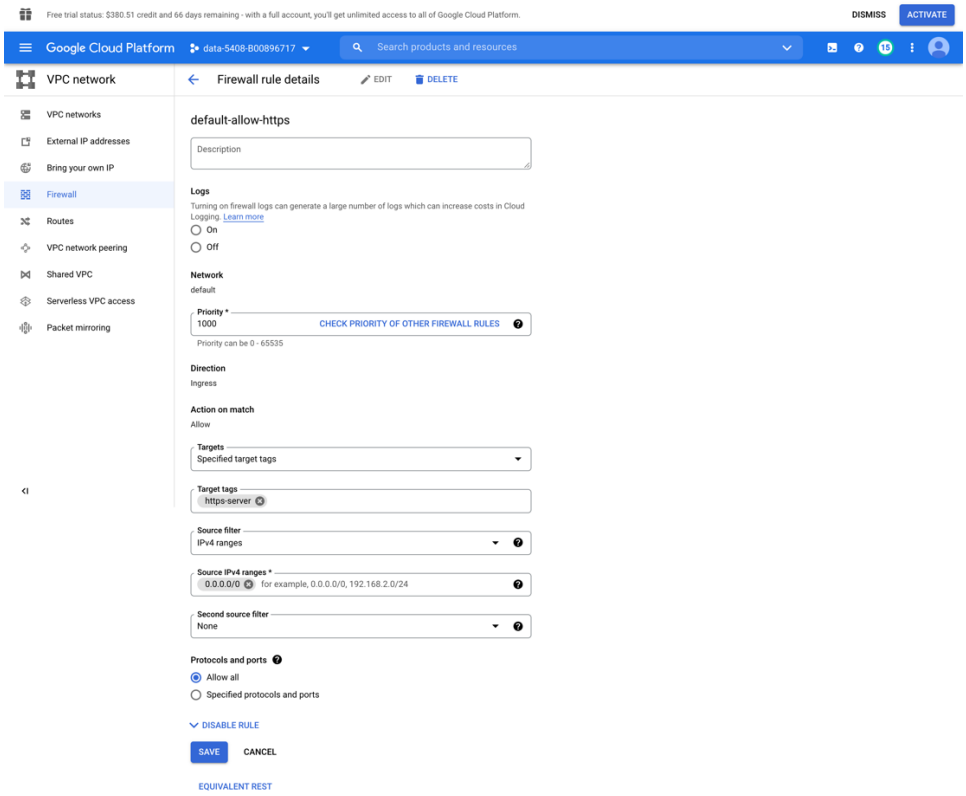
## Step – 1: Configuration and Initialization of Apache Spark cluster on GCP [1].

1. In project data-5408-B00896717, create a VM in GCP with following configuration.



2. Allow all traffic for http and https in firewall.



3. Open SSH command line to execute commands in the created VM.

```
Connected, host fingerprint: ssh-rsa 0 79:31:D9:3B:CD:D2:3B:B8:BA:F9:F0:EA:21:10
:99:5C:8C:18:C9:49:C5:DA:B6:BA:B8:B9:BF:69:E8:2A:A9:24
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-1021-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Nov 12 20:40:40 UTC 2021

  System load:  0.0               Processes:             112
  Usage of /:   32.6% of 9.52GB   Users logged in:       1
  Memory usage: 17%               IPv4 address for ens4: 10.128.0.3
  Swap usage:   0%


16 updates can be applied immediately.
8 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


Last login: Fri Nov 12 20:31:05 2021 from 35.235.245.128
prashitppatel@prashit-assign3:~$
```

4. Install jdk, scala and git on VM.

```
prashitppatel@prashit-assign3:~$ scala -version && java --version && git --version
Scala code runner version 2.11.12 -- Copyright 2002-2017, LAMP/EPFL
openjdk 11.0.11 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.20.04)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.20.04, mixed mode, sharing)
git version 2.25.1
prashitppatel@prashit-assign3:~$
```

5. Download spark setup with wget command.

```
prashitppatel@prashit-assign3:~/Apache_Spark$ wget https://downloads.apache.org/spark/spark-3.2.0/spark-3.2.0-bin-hadoop3.2.tgz
--2021-11-12 16:51:04--  https://downloads.apache.org/spark/spark-3.2.0/spark-3.2.0-bin-hadoop3.2.tgz
Resolving downloads.apache.org (downloads.apache.org)... 88.99.95.219, 135.181.214.104, 2a01:4f8:10a:201a::2, ...
Connecting to downloads.apache.org (downloads.apache.org)|88.99.95.219|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 300965906 (287M) [application/x-gzip]
Saving to: 'spark-3.2.0-bin-hadoop3.2.tgz'

spark-3.2.0-bin-hadoop3.2.tgz               100%[=========================================================================================>] 287.02M  24.7MB/s    in 12s

2021-11-12 16:51:17 (23.0 MB/s) - 'spark-3.2.0-bin-hadoop3.2.tgz' saved [300965906/300965906]

prashitppatel@prashit-assign3:~/Apache_Spark$
```

6. Install spark from the downloaded tar file.

```
prashitppatel@prashit-assign3:~/Apache_Spark$ tar xvf spark-3.2.0-bin-hadoop3.2.tgz
spark-3.2.0-bin-hadoop3.2/
spark-3.2.0-bin-hadoop3.2/NOTICE
spark-3.2.0-bin-hadoop3.2/kubernetes/
spark-3.2.0-bin-hadoop3.2/kubernetes/tests/
spark-3.2.0-bin-hadoop3.2/kubernetes/tests/python_executable_check.py
spark-3.2.0-bin-hadoop3.2/kubernetes/tests/autoscale.py
spark-3.2.0-bin-hadoop3.2/kubernetes/tests/worker_memory_check.py
spark-3.2.0-bin-hadoop3.2/kubernetes/tests/py_container_checks.py
spark-3.2.0-bin-hadoop3.2/kubernetes/tests/decommissioning.py
spark-3.2.0-bin-hadoop3.2/kubernetes/tests/pyfiles.py
spark-3.2.0-bin-hadoop3.2/kubernetes/tests/decommissioning_cleanup.py
spark-3.2.0-bin-hadoop3.2/kubernetes/dockerfiles/
spark-3.2.0-bin-hadoop3.2/kubernetes/dockerfiles/spark/
spark-3.2.0-bin-hadoop3.2/kubernetes/dockerfiles/spark/decom.sh
spark-3.2.0-bin-hadoop3.2/kubernetes/dockerfiles/spark/entrypoint.sh
spark-3.2.0-bin-hadoop3.2/kubernetes/dockerfiles/spark/bindings/
spark-3.2.0-bin-hadoop3.2/kubernetes/dockerfiles/spark/bindings/R/
spark-3.2.0-bin-hadoop3.2/kubernetes/dockerfiles/spark/bindings/R/Dockerfile
spark-3.2.0-bin-hadoop3.2/kubernetes/dockerfiles/spark/bindings/python/
spark-3.2.0-bin-hadoop3.2/kubernetes/dockerfiles/spark/bindings/python/Dockerfile
spark-3.2.0-bin-hadoop3.2/kubernetes/dockerfiles/spark/Dockerfile
spark-3.2.0-bin-hadoop3.2/jars/
spark-3.2.0-bin-hadoop3.2/jars/RoaringBitmap-0.9.0.jar
```

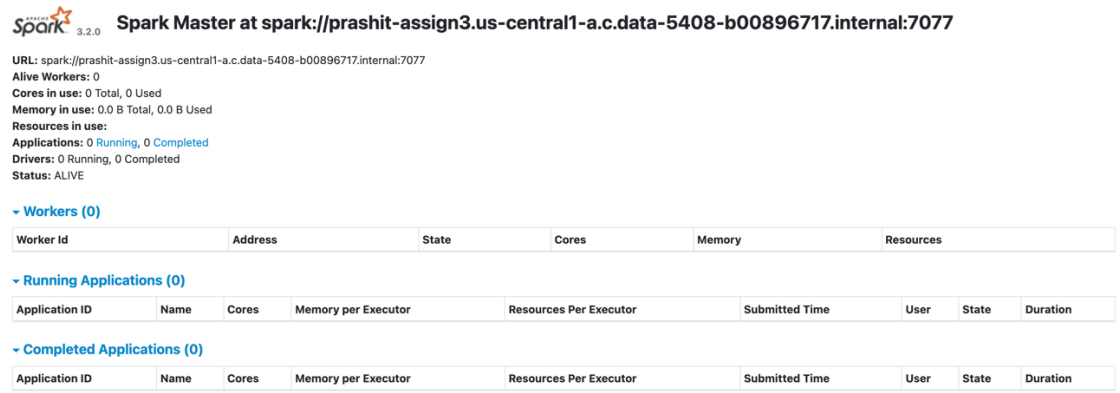
7. Successful installation will create following files in the directory.

```
prashitppatel@prashit-assign3:~/Apache_Spark/spark-3.2.0-bin-hadoop3.2/bin$ ls
beeline               find-spark-home      load-spark-env.sh  pyspark2.cmd     spark-class      spark-shell      spark-sql      spark-submit      sparkR
beeline.cmd           find-spark-home.cmd  pyspark            run-example      spark-class.cmd  spark-shell.cmd  spark-sql.cmd  spark-submit.cmd  sparkR.cmd
docker-image-tool.sh  load-spark-env.cmd   pyspark.cmd        run-example.cmd  spark-class2.cmd spark-shell2.cmd spark-sql2.cmd spark-submit2.cmd sparkR2.cmd
prashitppatel@prashit-assign3:~/Apache_Spark/spark-3.2.0-bin-hadoop3.2/bin$
```

8. After setting up path variables, execute start-master.sh to start the master node.

```
prashitppatel@prashit-assign3:~/Apache_Spark$ start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spark-prashitppatel-org.apache.spark.deploy.master.Master-1-prashit-assign3.out
prashitppatel@prashit-assign3:~/Apache_Spark$
```

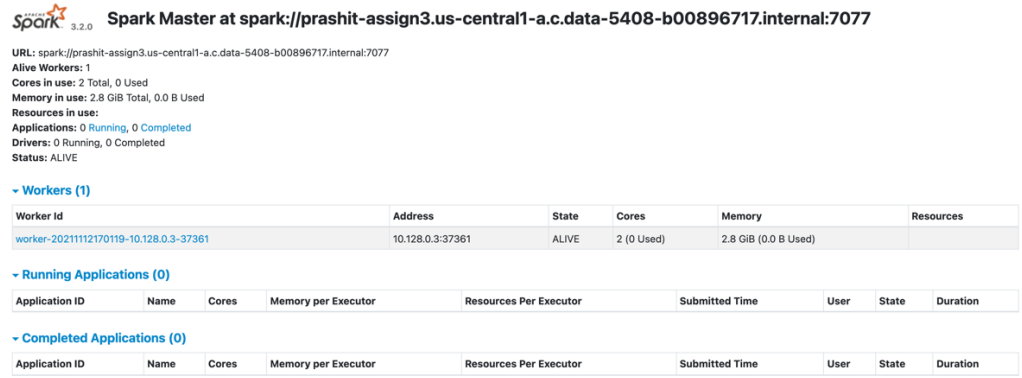
9. Open {external_ip}:8080 port in browser to verify successful creation of master node.




10. Start slave nodes using start-slave.sh {master_node_url}

```
prashitppatel@prashit-assign3:/opt/spark/bin$ start-slave.sh spark://prashit-assign3.us-central1-a.c.data-5408-b00896717.internal:7077
This script is deprecated, use start-worker.sh
starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark/logs/spark-prashitppatel-org.apache.spark.deploy.worker.Worker-1-prashit-assign3.out
prashitppatel@prashit-assign3:/opt/spark/bin$
```

11. Successful creation of slave node can be verified in the master node window under workers.



## Step – 9: Tweets extraction:

## Twitter search api [2] - https://api.twitter.com/1.1/search/tweets.json

- Parameters considered with above url are
  - q={keyword} – keyword is the search term,
  - tweet_mode=extended – getting whole tweet texts,
  - lang=en – tweet language,
  - count=100,
  - max_id={id} – to get next page of tweets
- One page gives 100 tweets.
- Bearer token from twitter developer account was used for authentication in headers.
- Repository for problem 1 - task1 contains a Main class, DbConnector class, TweetExtractor class, and TransformTweets class.
- Main class calls extractTweets() method of TweetExtractor and transformTweets() method of TransformTweets object for each keyword.
- addData() method of DbConnector calls insertOne() method of MongoDb and getData() method of DbConnector calls find() method of MongoDb [3].
- remove() method in TransformTweets class takes pattern and text and replaces the matching string with "" – empty string [4].
- cleanTweet() in TransformTweets class method calls remove() method to remove url, emoticons and special characters [5][6].
- Metadata stored after transformation includes – tweet id, tweet, user name, retweet count, date, time and place.

- Tweet Extraction Algorithm:
  1. BEGIN.
  2. SET Bearer Token in Authentication Header.
  3. CREATE DbConnector object for creating and storing keyword specific data in RawDb.
  4. INITIALIZE count variable = 0.
  5. WHILE count < 5
     a. CALL search api.
     b. Store response in body.
     c. Parse response to convert to JSON Array
     d. CREATE iterator for JSON Array.
     e. WHILE iterator has next data
        i. CALL addData method of DbConnector to store JSON object.
     f. UPDATE query to include max_id from next_results.
  6. PRINT tweets extracted for {keyword} message.
  7. END.

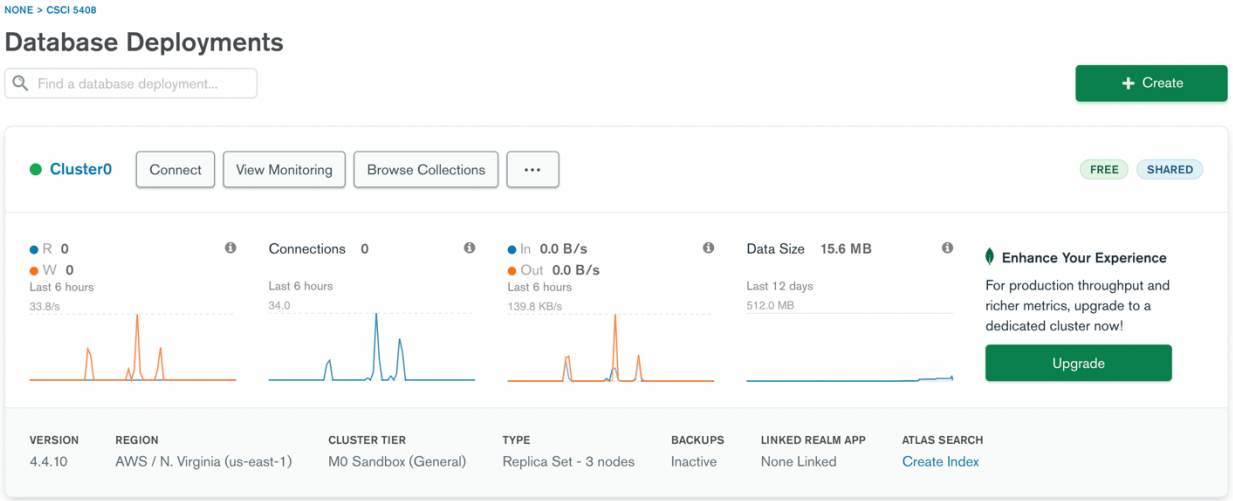
- Tweet Transformation Algorithm:
  1. BEGIN.
  2. CREATE DbConnector object with keyword and call getData for getting data stored in RawDb.
  3. CREATE DbConnector object for creating and storing transformed keyword specific data in ProcessedDb.
  4. CREATE iterator for received data.
  5. WHILE the iterator has next data.
     a. CREATE new JSON object [7].
     b. Extract and store id from tweet_data.
     c. Extract full_text from tweet_data.
     d. IF first 2 characters of full_text contains 'RT' AND retweeted_status is NOT NULL
        i. Extract full_text from retweeted
     e. CALL cleanTweet method for full_text and store in JSON object.
     f. Extract created_at.
     g. Split created_at to split time and date information and store in JSON object.
     h. Extract name from user.
     i. Extract retweet_count from tweet_data.
     j. Extract name from place.
     k. IF name is null store NA, ELSE name.
     l. CALL addData method of DbConnector to store JSON object to ProcessedDb.
  6. PRINT tweets processed for {keyword} message.
  7. END.

- Patterns considered:
  a. (https?|ftp|file)://[-a-zA-Z0-9+&@#/%?=~_|!:,.;]*[-a-zA-Z0-9+&@#/%=~_|] – to remove url [5].
  b. [^\p{L}\p{N}\p{P}\p{Z}] – to remove emoticons [6].
  c. [!"#$%&'()*+,-./:;<=>?@\[\]^_`{|}~] – to remove special characters.

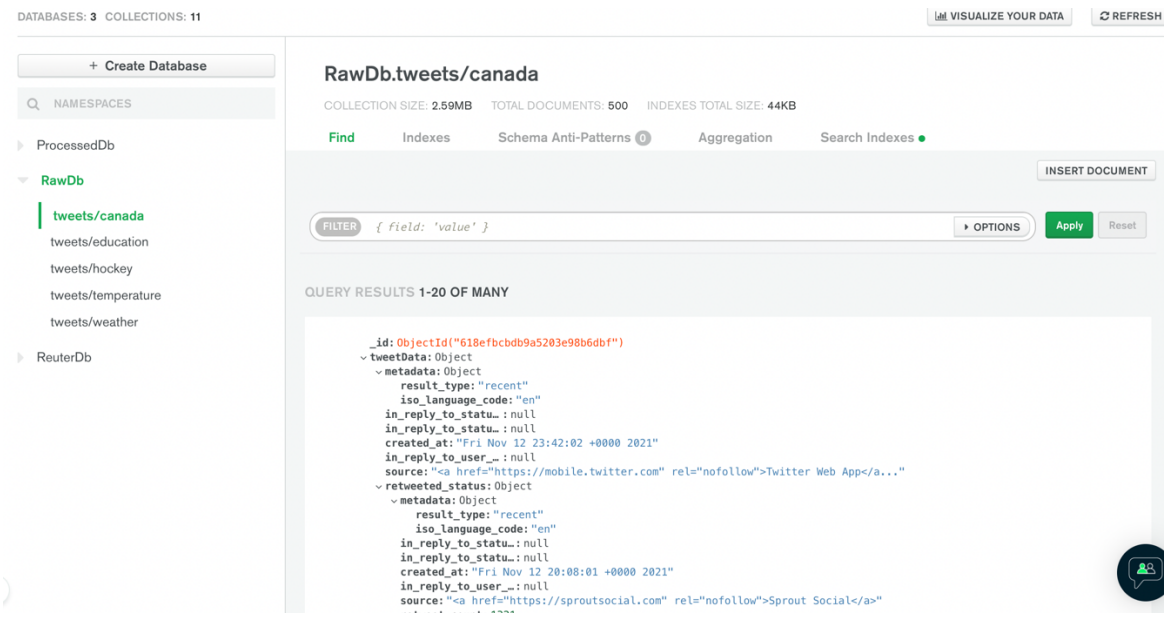
- Output of above program after executing it on GCP.

```
prashitppatel@prashit-assign3:~$ java -cp task1-1.0-SNAPSHOT-jar-with-dependencies.jar Main
Tweets extracted for weather
Tweets extracted for hockey
Tweets extracted for canada
Tweets extracted for temperature
Tweets extracted for education
Tweets processed for weather
Tweets processed for hockey
Tweets processed for canada
Tweets processed for temperature
Tweets processed for education
prashitppatel@prashit-assign3:~$
```

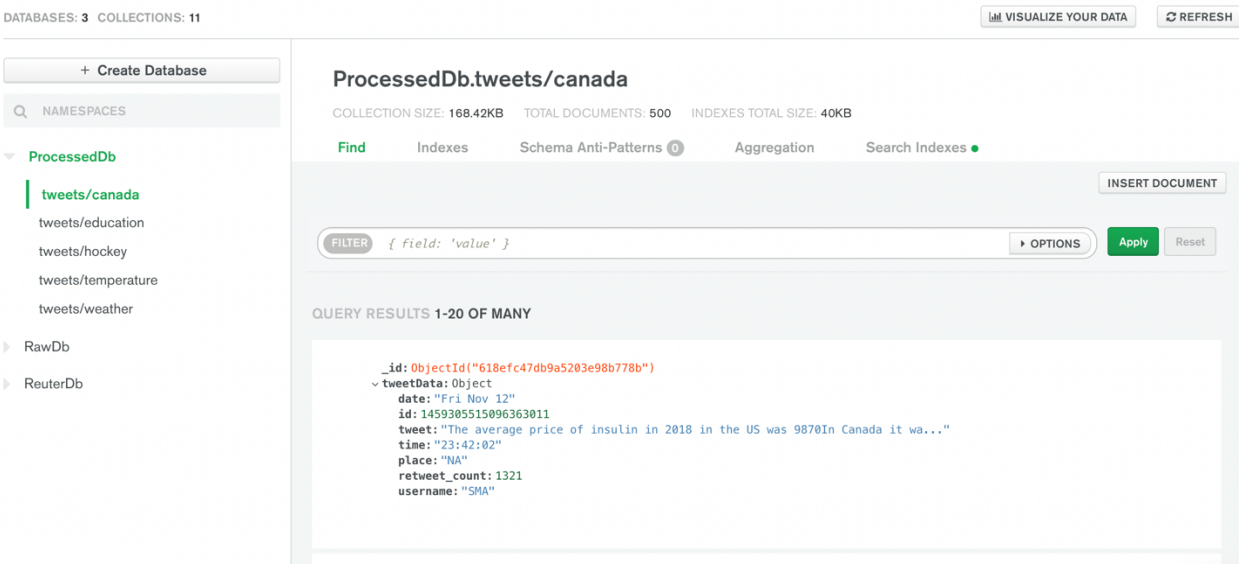
- MongoDb collection is generated on Atlas cluster4 and populated as follows.




- Collections in MongoDb after program execution are as follows.
  - RawDb



- ProcessedDb



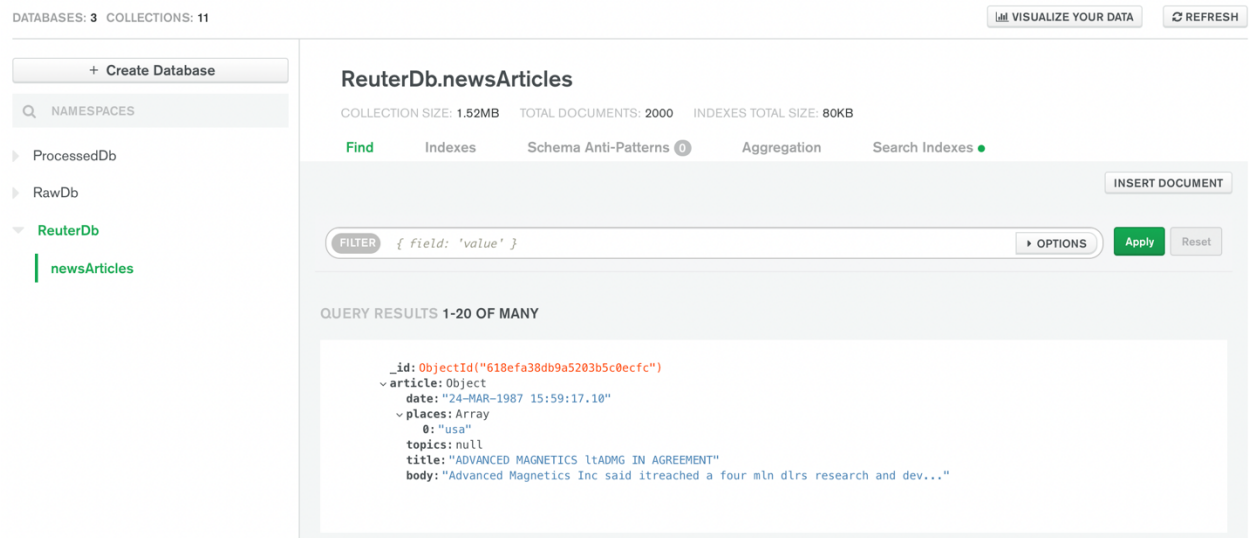
## Problem 2:
## Step – 10: News articles extraction [8]:

- Repository for problem 2 – task2 contains a Main class, DbConnector class, and ArticleExtractor class.
- Main class calls extractArticles() method of ArticleExtractor object of the provided files.
- addData() method of DbConnector calls insertOne() method of MongoDb and getData() method of DbConnector calls find() method of MongoDb.
- remove() method in ArticleExtractor class takes pattern and text and replaces the matching string with “” – empty string.
- cleanTweet() in ArticleExtractor class method calls remove() method to remove url, emoticons and special characters.
- Metadata stored for each article after transformation includes – title, body, topics, date, time and places.

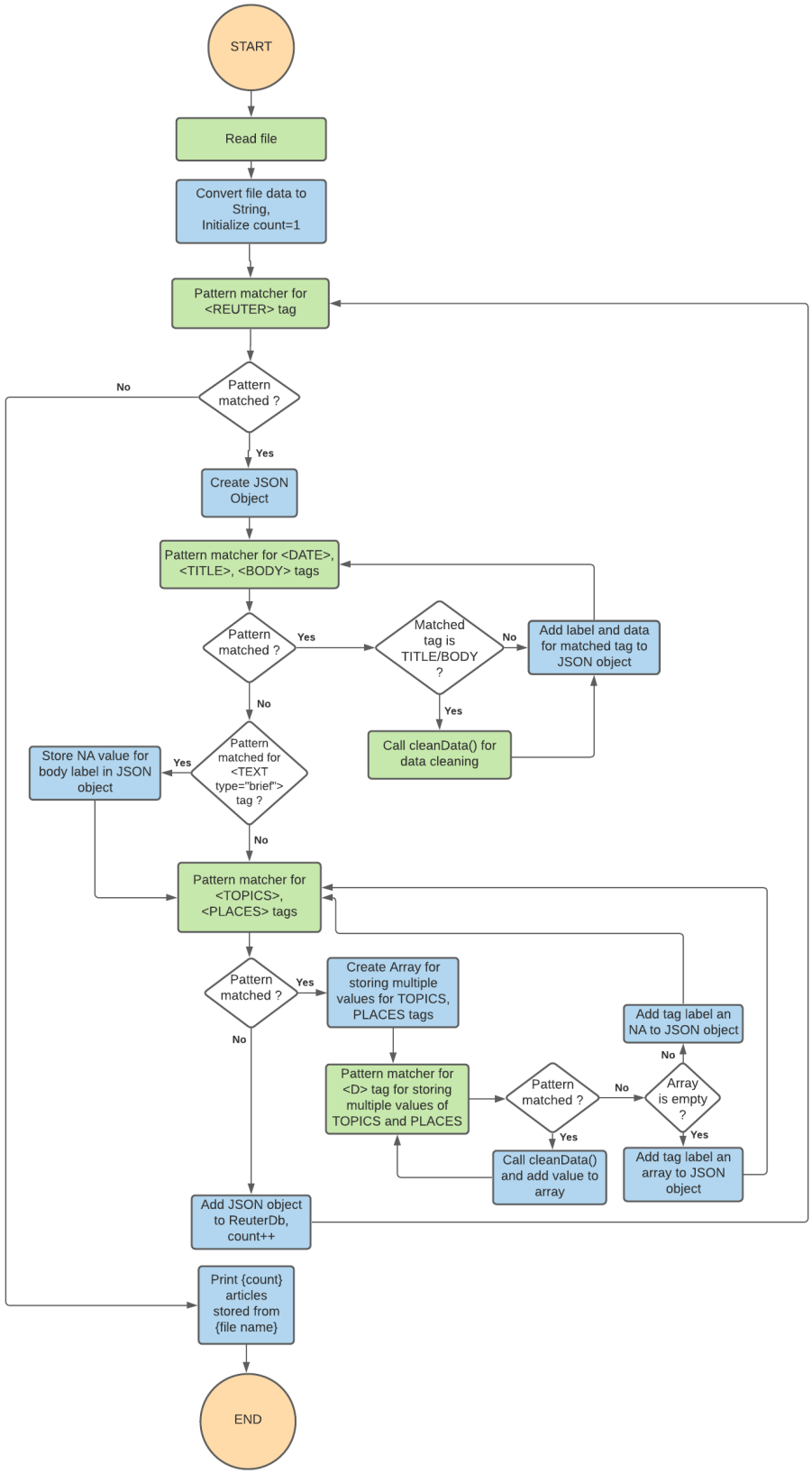
- Output of above program after executing it on GCP.

```
prashitppatel@prashit-assign3:~$ java -cp task2-1.0-SNAPSHOT-jar-with-dependencies.jar Main
1001 articles stored from reut2-009.sgm
1001 articles stored from reut2-014.sgm
prashitppatel@prashit-assign3:~$
```

- Collection in MongoDb after program execution is as follows.



- Flow Chart for news extraction program [9]:

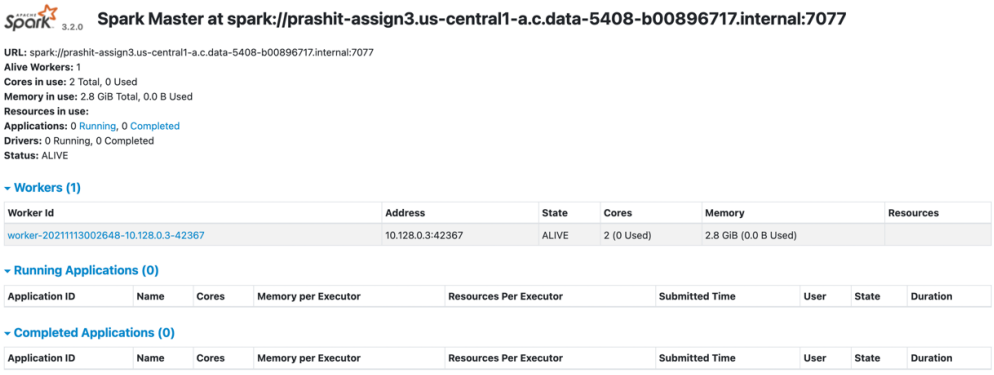


## Problem 3:
## Step – 12: Map Reduce in Apache Spark:

- Repository for problem 3 – task3 contains a Main class, MapReduce class, Input class and Output class [10].
- Input class has prepareInput() method to convert json input files to single text file [11].
- Output class contains store() method to store the output of map reduce to output file [12].
- Main class calls prepareInput() method of Input class at the start of the program, provides the generated text file to mapReduce class and stores output of map reduce by calling store() method of Output class.
- MapReduce class contains calculate() method that will perform map reduce operations [13][14][15].

- MapReduce Algorithm
  1. BEGIN
  2. INITIALIZE spark config object by with appName="mapReduce" and master with Apache Spark master node url.
  3. Store keywords to be matched in tokens array.
  4. READ input text file with textFile() method of spark config object.
  5. CALL flatMap() method on returned RDD strings from Step 4. Perform following tasks in flatMap() method:
     i. Split lines with " " and store generated words in Array List.
     ii. Convert words to lower case and filter words by tokens array.
     iii. Return iterator of filtered Array List.
  6. CALL mapToPair() function on iterator from Step 5 which will return pair - (word in lower case, 1).
  7. CALL reduceByKey() function on word pairs to reduce and add words based on same key.
  8. END.

- Start master and slave nodes to execute program in gcp using start-master.sh and start-slave.sh commands. 2 cores are used for slave nodes as seen under workers [1]:



- Upload jar file with dependencies and store it in bin folder under spark directory. In my case path was /opt/spark/bin. Also store all input json files in a data-files (user specified) directory under same path.

```
prashitppatel@prashit-assign3:/opt/spark/bin$ cp -i ~/task3-1.0-SNAPSHOT-jar-with-dependencies.jar /opt/spark/bin/
```

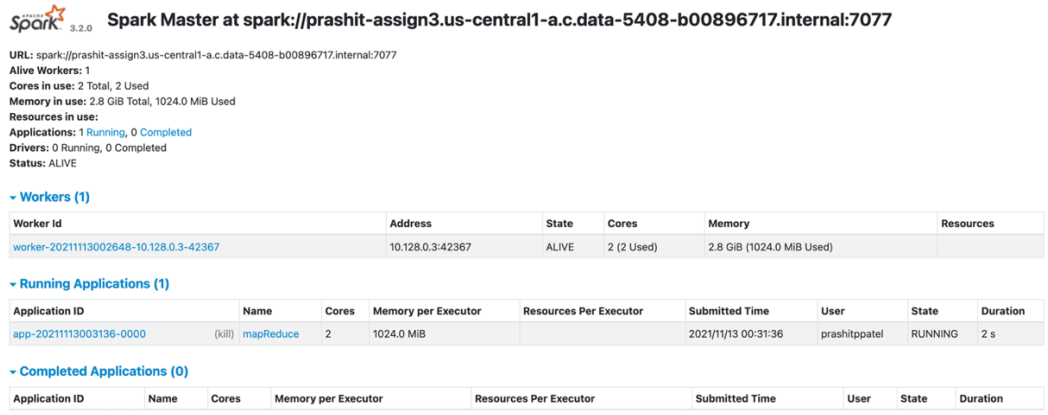
```
prashitppatel@prashit-assign3:~$ mkdir data-files
prashitppatel@prashit-assign3:~$ mv newsArticles.json data-files/
prashitppatel@prashit-assign3:~$ mv canada.json data-files/
prashitppatel@prashit-assign3:~$ mv education.json data-files/
prashitppatel@prashit-assign3:~$ mv hockey.json data-files/
prashitppatel@prashit-assign3:~$ mv temperature.json data-files/
prashitppatel@prashit-assign3:~$ mv weather.json data-files/
prashitppatel@prashit-assign3:~$ cp -r data-files/ /opt/spark/bin/
```

```
prashitppatel@prashit-assign3:/opt/spark/bin$ ls
beeline                find-spark-home      pyspark          run-example.cmd   spark-shell       spark-sql.cmd     spark-submit2.cmd  task3-1.0-SNAPSHOT-jar-with-dependencies.jar
beeline.cmd            find-spark-home.cmd  pyspark.cmd      spark-class       spark-shell.cmd   spark-sql2.cmd    sparkR
data-files             load-spark-env.cmd   pyspark2.cmd     spark-class.cmd   spark-shell2.cmd  spark-submit      sparkR.cmd
docker-image-tool.sh   load-spark-env.sh    run-example      spark-class2.cmd  spark-sql         spark-submit.cmd  sparkR2.cmd
prashitppatel@prashit-assign3:/opt/spark/bin$
```

- Run the program using command [16]:
  Spark-submit --class {Main Class name} {jar file name}

```
prashitppatel@prashit-assign3:/opt/spark/bin$ spark-submit --class Main task3-1.0-SNAPSHOT-jar-with-dependencies.jar
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/opt/spark/jars/spark-unsafe_2.12-3.2.0.jar) to constructor java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
21/11/13 00:25:23 INFO SparkContext: Running Spark version 3.2.0
21/11/13 00:25:23 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

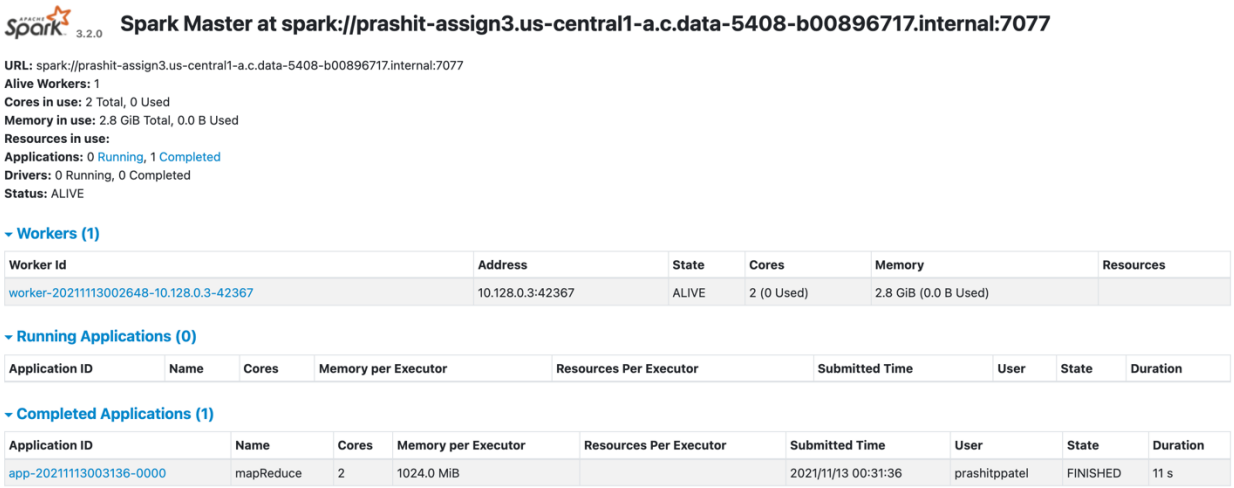
- On starting the program, application is started under Apache Spark and is using 2 cores and is in running state as seen below:




- After successful execution of program, output.txt is generated containing required output.

```
prashitppatel@prashit-assign3:/opt/spark/bin$ cat output.txt
Final Output [(canada,1532), (education,1656), (rain,172), (cold,220), (indoor,20), (ice,64), (hot,76), (flu,8), (snow,92)]
```

- Successful completion status and related details can be seen on Apache Spark dashboard which mentions total execution time (11s) and state finished as follows:



Spark 3.2.0 **Spark Master at spark://prashit-assign3.us-central1-a.c.data-5408-b00896717.internal:7077**

**URL:** spark://prashit-assign3.us-central1-a.c.data-5408-b00896717.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, 1 Completed
**Drivers:** 0 Running, 0 Completed
**Status:** ALIVE

**Workers (1)**

| Worker Id | Address | State | Cores | Memory | Resources |
|---|---|---|---|---|---|
| worker-20211113002648-10.128.0.3-42367 | 10.128.0.3:42367 | 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 |
|---|---|---|---|---|---|---|---|---|

**Completed Applications (1)**

| Application ID | Name | Cores | Memory per Executor | Resources Per Executor | Submitted Time | User | State | Duration |
|---|---|---|---|---|---|---|---|---|
| app-20211113003136-0000 | mapReduce | 2 | 1024.0 MiB | | 2021/11/13 00:31:36 | prashitppatel | FINISHED | 11 s |

- Output of map reduce

| **Keyword** | **Frequency** |
|---|---|
| canada | 1532 |
| education | 1656 |
| rain | 172 |
| cold | 220 |
| indoor | 20 |
| ice | 64 |
| hot | 76 |
| flu | 8 |
| snow | 92 |

Highest Frequency – education – 1656
Lowest Frequency – flu – 8

## Problem 4:
## Step – 14: Neo4j graph database cypher and graph [17]:

- **Cypher to create nodes [18]:**

```
CREATE (n:Province {name: 'Newfoundland and Labrador', capital: 'St. John's'})
CREATE (n:Province {name: 'Prince Edward Island', capital: 'Charlottetown'})
CREATE (n:Province {name: 'Nova Scotia', capital: 'Halifax'})
CREATE (n:Province {name: 'New Brunswick', capital: 'Fredericton'})
CREATE (n:Province {name: 'Quebec', capital: 'Quebec City'})
CREATE (n:Province {name: 'Ontario', capital: 'Toronto'})
CREATE (n:Province {name: 'Manitoba', capital: 'Winnipeg'})
CREATE (n:Province {name: 'Saskatchewan', capital: 'Regina'})
CREATE (n:Province {name: 'Alberta', capital: 'Edmonton'})
CREATE (n:Province {name: 'British Columbia', capital: 'Victoria'})
CREATE (n:Province {name: 'Nunavut', capital: 'Iqaluit'})
CREATE (n:Province {name: 'Northwest Territories', capital: 'Yellowknife'})
CREATE (n:Province {name: 'Yukon', capital: 'Whitehorse'})
```

Here, all the provinces are considered as nodes with Province as node label. Node has fields name and capital where name includes the name of the province and capital includes the capital city of that province.

- **Cypher to create relationships [18]:**

```
MATCH (a:Province), (b:Province)
WHERE a.name = 'Nova Scotia' AND b.name = 'New Brunswick'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'New Brunswick' AND b.name = 'Nova Scotia'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Prince Edward Island' AND b.name = 'New Brunswick'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'New Brunswick' AND b.name = 'Prince Edward Island'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Prince Edward Island' AND b.name = 'Nova Scotia'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name
```

```
MATCH (a:Province), (b:Province)
WHERE a.name = 'Nova Scotia' AND b.name = 'Prince Edward Island'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Prince Edward Island' AND b.name = 'Quebec'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Quebec' AND b.name = 'Prince Edward Island'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Quebec' AND b.name = 'New Brunswick'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'New Brunswick' AND b.name = 'Quebec'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Quebec' AND b.name = 'Newfoundland and Labrador'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Newfoundland and Labrador' AND b.name = 'Quebec'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Quebec' AND b.name = 'Ontario'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Ontario' AND b.name = 'Quebec'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Quebec' AND b.name = 'Nunavut'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name
```

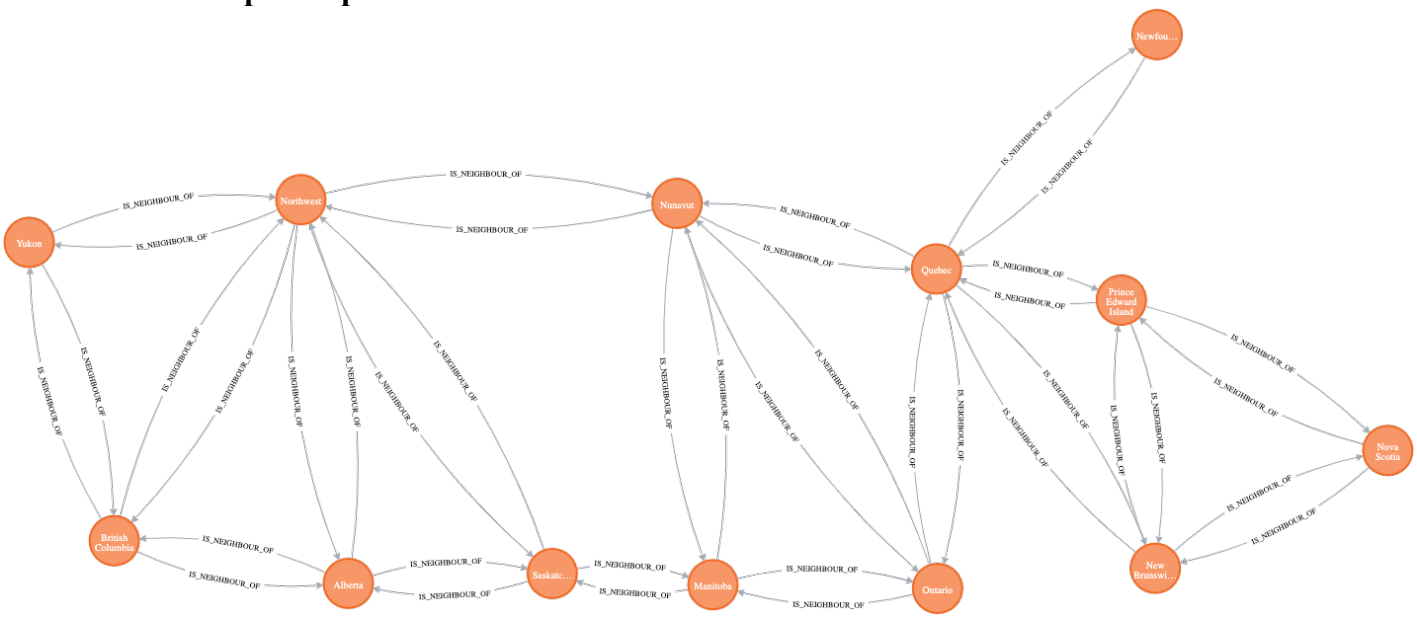
```
MATCH (a:Province), (b:Province)
WHERE a.name = 'Nunavut' AND b.name = 'Quebec'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Nunavut' AND b.name = 'Ontario'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Ontario' AND b.name = 'Nunavut'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Nunavut' AND b.name = ‘Manitoba’
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Manitoba’' AND b.name = 'Nunavut'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = ‘Ontario’ AND b.name = ‘Manitoba’
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Manitoba' AND b.name = 'Ontario'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Northwest Territories' AND b.name = 'Nunavut'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Nunavut' AND b.name = 'Northwest Territories'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Northwest Territories' AND b.name = 'Saskatchewan'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name
```

```
MATCH (a:Province), (b:Province)
WHERE a.name = 'Saskatchewan' AND b.name = 'Northwest Territories'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Manitoba' AND b.name = 'Saskatchewan'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Saskatchewan' AND b.name = ‘Manitoba'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Northwest Territories' AND b.name = 'Alberta'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = ‘Alberta' AND b.name = 'Northwest Territories'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Northwest Territories' AND b.name = 'Yukon'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = ‘Yukon’ AND b.name = 'Northwest Territories'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Northwest Territories' AND b.name = 'British Columbia'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = ‘British Columbia’ AND b.name = 'Northwest Territories'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Yukon' AND b.name = 'British Columbia'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name
```

```
MATCH (a:Province), (b:Province)
WHERE a.name = ‘British Columbia’ AND b.name = ‘Yukon’
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Alberta' AND b.name = 'British Columbia'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = ‘British Columbia’ AND b.name = ‘Alberta’
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = 'Alberta' AND b.name = 'Saskatchewan'
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name

MATCH (a:Province), (b:Province)
WHERE a.name = ‘Saskatchewan’ AND b.name = ‘Alberta’
CREATE (a)-[r:IS_NEIGHBOUR_OF {name: a.name + '<->' + b.name}]->(b)
RETURN type(r), r.name
```

Here, edges are considered as relationship between 2 provinces. a and b are province nodes who are neighbours of each other. Relationship is created between them where relationship name is IS_NEIGHBOUR_OF. It is a 2-way relation. For example, New Brunswick is neighbour of Nova Scotia and Nova Scotia is neighbour of New Brunswick. So, 2 relationships will be created between these 2 provinces.

- **Graph Output**



- **References**

[1] “Lab 6 Notes” [Online-Brightspace]. [Accessed on 10th November 2021]

[2] “Tweeter Developer” [Online].
Available: https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/api-reference/get-search-tweets [Accessed on 8th November 2021]

[3] “MongoDb Documentation” [Online].
Available: https://mongodb.github.io/mongo-java-driver/3.4/driver/getting-started/quick-start/ [Accessed on 7th November 2021]

[4] “w3 schools” [Online].
Available: https://www.w3schools.com/java/java_regex.asp [Accessed on 8th November 2021]

[5] “Stack Overflow” [Online].
Available: https://stackoverflow.com/questions/163360/regular-expression-to-match-urls-in-java [Accessed on 9th November 2021]

[6] “Baeldung” [Online].
Available: https://www.baeldung.com/java-string-remove-emojis [Accessed on 9th November 2021]

[7] “Mkyong” [Online].
Available: https://mkyong.com/java/json-simple-example-read-and-write-json/ [Accessed on 9th November 2021]

[8] “Stack Overflow” [Online].
Available: https://stackoverflow.com/questions/6560672/java-regex-to-extract-text-between-tags [Accessed on 10th November 2021]

[9] “Lucid Chart” [Online].
Available: https://www.lucidchart.com/pages/examples/flowchart-maker [Accessed on 12th November 2021]

[10] “Apache Spark” [Online].
Available: https://spark.apache.org/docs/latest/rdd-programming-guide.html [Accessed on 10th November 2021]

[11] “Stack Overflow” [Online].
Available: https://stackoverflow.com/questions/28947250/create-a-directory-if-it-does-not-exist-and-then-create-the-files-in-that-direct [Accessed on 11th November 2021]

[12] “Stack Overflow” [Online].
Available: https://stackoverflow.com/questions/22347124/how-to-open-all-files-in-a-directory-in-java-read-them-creat-new-files-write [Accessed on 11th November 2021]

[13] “Spark by examples” [Online].
Available: https://sparkbyexamples.com/spark/spark-flatmap-usage-with-example/ [Accessed on 11th November 2021]

[14] “Tabnine” [Online].
Available:
https://www.tabnine.com/code/java/methods/org.apache.spark.api.java.JavaRDD/mapToPair
[Accessed on 11th November 2021]

[15] “Stack Overflow” [Online].
Available: https://stackoverflow.com/questions/25091091/apache-spark-reducebykey-java
[Accessed on 11th November 2021]

[16] “YouTube” [Online].
Available: https://www.youtube.com/watch?v=5uHG0aqir5s [Accessed on 12th November 2021]

[17] “World Atlas” [Online].
Available: https://www.worldatlas.com/geography/capital-cities-of-canada-s-provinces-territories.html [Accessed on 12th November 2021]

[18] “Neo4j Docs” [Online].
Available: https://neo4j.com/docs/cypher-manual/current/clauses/create/#create-create-multiple-nodes [Accessed on 12th November 2021]