

ANALYTICS SYSTEMS ENGINEERING (MSDS 436)

EXERCISE 4

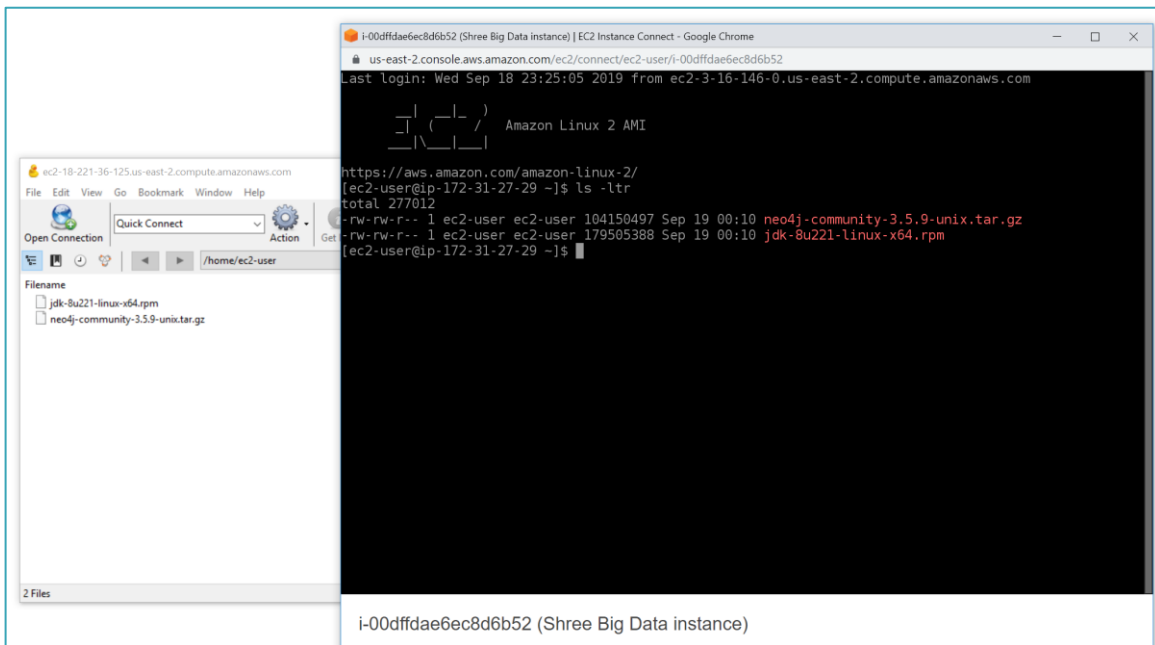
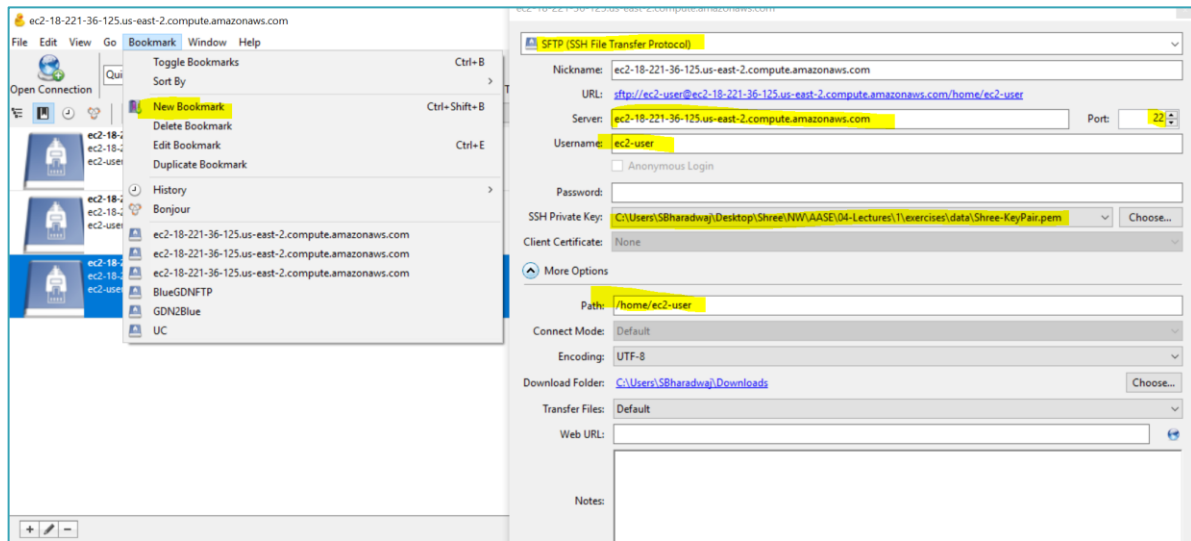
shreenidhi.bharadwaj@northwestern.edu | ChristopherFiore2015@u.northwestern.edu

CONTENTS

1. [Working with AWS](#)
 - a. [Amazon EC2 \(Week2\)](#)
 - b. [Object Storage \(Week2\)](#)
 - c. [PostgreSQL \(Week2\)](#)
 - d. [Tableau & DBeaver \(Week2\)](#)
 - e. [RedShift \(Week2\)](#)
 - f. [Elasticsearch \(Week3\)](#)
 - g. [Neo4j\(Week4\)](#)
 - h. [Jupyter Notebook \(Week5\)](#)
 - i. [Apache Spark \(Week5 & Week6\)](#)

INSTALL NEO4J ON AWS EC2

1. Download the latest stable version of neo4j. The “Community Edition” fits well for development purposes. Do not forget to select the Linux version of the server. This will download a tar.gz file. Alternatively, a version of neo4J (**neo4j-community-3.5.9-unix.tar.gz**) is already added to the exercise folder within canvas which you can use readily.
2. Download the JDK file “**jdk-8u221-linux-x64.rpm**” from canvas and store it locally.
3. Copy both the **neo4j-community-3.5.9-unix.tar.gz & jdk-8u221-linux-x64.rpm** files to EC2 using Cyberduck.



4. Install java on the EC2 Instance and confirm it is installed successfully
 - a. **sudo yum install -y jdk-8u221-linux-x64.rpm**
 - b. **java -version**

```
i-00dffdae6ec8d6b52 (Shree Big Data instance) | EC2 Instance Connect - Google Chrome
us-east-2.console.aws.amazon.com/ec2/connect/ec2-user/i-00dffdae6ec8d6b52

Installing:
jdk1.8          x86_64          2000:1.8.0_221-fcs          /jdk-8u221-linux-x64          298 M

Transaction Summary
=====
Install 1 Package

Total size: 298 M
Installed size: 298 M
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : 2000:jdk1.8-1.8.0_221-fcs.x86_64          1/1
Unpacking JAR files...
  tools.jar...
  plugin.jar...
  javaws.jar...
  deploy.jar...
  rt.jar...
  jsse.jar...
  charsets.jar...
  localedata.jar...
  Verifying : 2000:jdk1.8-1.8.0_221-fcs.x86_64          1/1

Installed:
  jdk1.8.x86_64 2000:1.8.0_221-fcs

Complete!
[ec2-user@ip-172-31-27-29 ~]$ java -version
java version "1.8.0_221"
Java(TM) SE Runtime Environment (build 1.8.0_221-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.221-b11, mixed mode)
[ec2-user@ip-172-31-27-29 ~]$
```

5. Install Neo4j on the EC2 Instance and move it to /usr/local and rename the folder to neo4j.

- a. **tar xvfz neo4j-community-3.5.9-unix.tar.gz**
- b. **sudo mv neo4j-community-3.5.9 /usr/local/neo4j**

```
total 277012
drwxr-xr-x 10 ec2-user ec2-user      198 Aug 22 11:27 neo4j-community-3.5.9
-rw-rw-r--  1 ec2-user ec2-user 104150497 Sep 19 00:10 neo4j-community-3.5.9-unix.tar.gz
-rw-rw-r--  1 ec2-user ec2-user 179505388 Sep 19 00:10 jdk-8u221-linux-x64.rpm
[ec2-user@ip-172-31-27-29 ~]$ sudo mv neo4j-community-3.5.9 /usr/local/neo4j
[ec2-user@ip-172-31-27-29 ~]$
```

6. Open “neo4j.conf” using vi and locate “Network connector configuration” section to Uncomment (delete “#”, use x by placing cursor on #) in front of “dbms.connectors.default_listen_address=0.0.0.0” on line 54 and save the file by typing (ESC, :wq! + Enter)

- a. **vi /usr/local/neo4j/conf/neo4j.conf**

Note: This will let you connect to Neo4j from your local machine

```
i-00dffdae6ec8d6b52 (Shree Big Data Instance) | EC2 Instance Connect - Google Chrome
us-east-2.console.aws.amazon.com/ec2/connect/ec2-user/i-00dffdae6ec8d6b52

# The amount of memory to use for mapping the store files, in bytes (or
# kilobytes with the 'k' suffix, megabytes with 'm' and gigabytes with 'g').
# If Neo4j is running on a dedicated server, then it is generally recommended
# to leave about 2-4 gigabytes for the operating system, give the JVM enough
# heap to hold all your transaction state and query context, and then leave the
# rest for the page cache.
# The default page cache memory assumes the machine is dedicated to running
# Neo4j, and is heuristically set to 50% of RAM minus the max Java heap size.
#dbms.memory.pagecache.size=10g

# Network connector configuration
#dbms.connector.default_listen_address=0.0.0.0

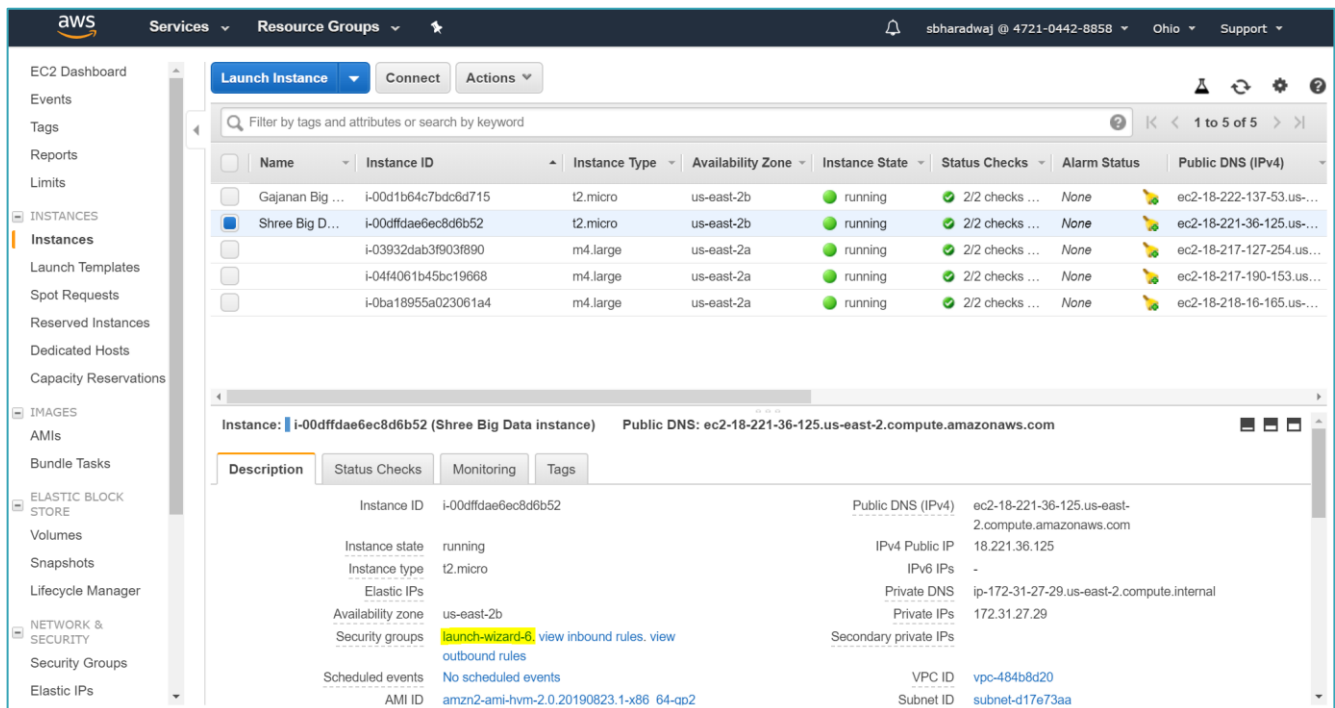
# With default configuration Neo4j only accepts local connections.
# To accept non-local connections, uncomment this line:
#dbms.connector.default_listen_address=0.0.0.0

# You can also choose a specific network interface, and configure a non-default
# port for each connector, by setting their individual listen_address.

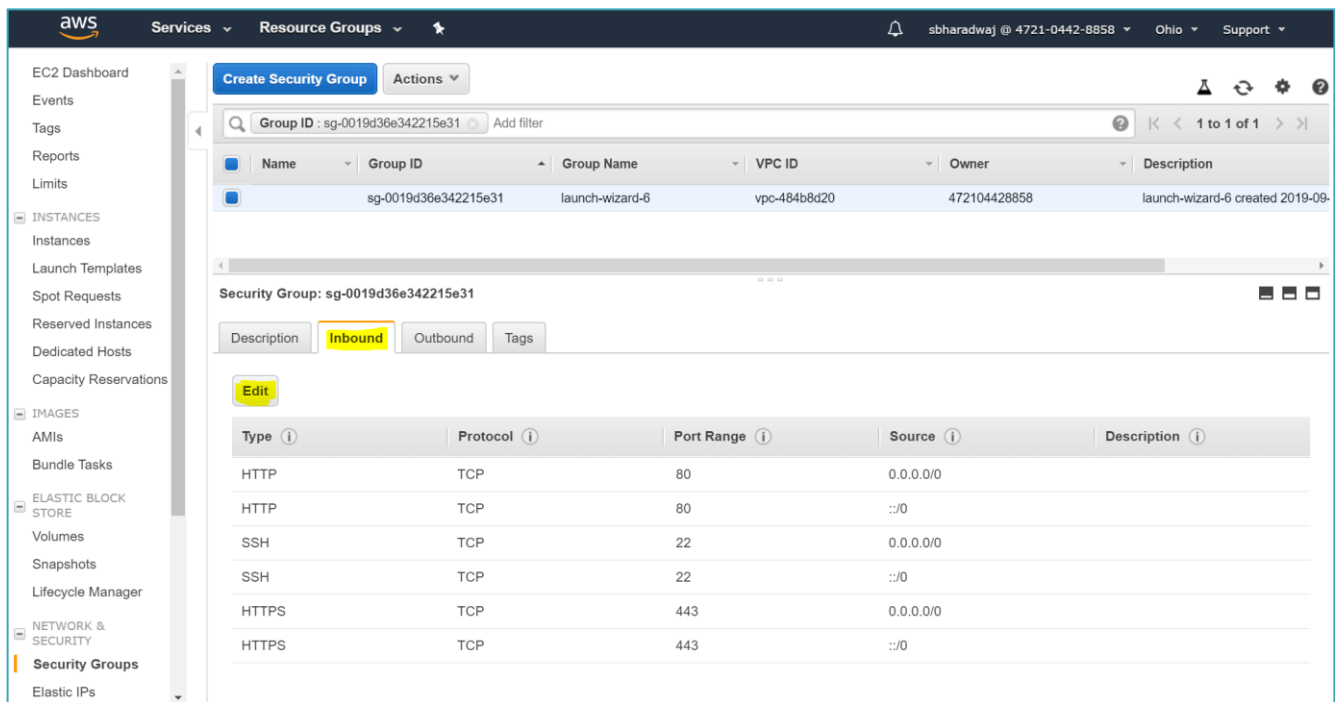
# The address at which this server can be reached by its clients. This may be the server's IP address or DNS name, or
# it may be the address of a reverse proxy which sits in front of the server. This setting may be overridden for
# individual connectors below.
#dbms.connector.default_advertised_address=localhost

# You can also choose a specific advertised hostname or IP address, and
# configure an advertised port for each connector, by setting their
# individual advertised_address.
```

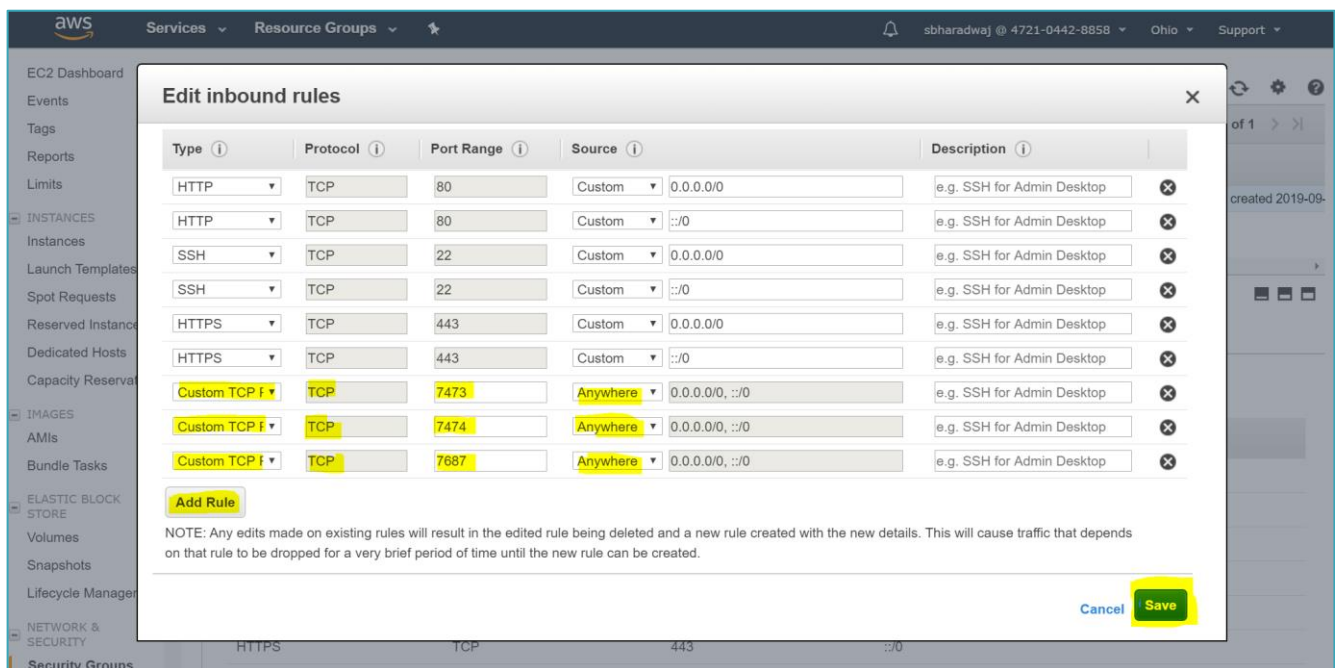
7. To Access to Neo4j from your laptop we will need to open specific ports in AWS EC2 instance. Locate the security group of your EC2 instance & click on it



8. Click on “inbound” tab & then click on “Edit” to add new rules



9. Add “Custom TCP rule” for ports 7473, 7474, 7687 and click “Save”



10. Start the Neo4J server

- `sudo /usr/local/neo4j/bin/neo4j start`
- `tail -f /usr/local/neo4j/logs/neo4j.log` (if you want to check the logs)

```
[ec2-user@ip-172-31-27-29 ~]$ sudo /usr/local/neo4j/bin/neo4j start
Active database: graph.db
Directories in use:
  home:      /usr/local/neo4j
  config:    /usr/local/neo4j/conf
  logs:      /usr/local/neo4j/logs
  plugins:   /usr/local/neo4j/plugins
  import:    /usr/local/neo4j/import
  data:      /usr/local/neo4j/data
  certificates: /usr/local/neo4j/certificates
  run:       /usr/local/neo4j/run
Starting Neo4j.
WARNING: Max 1024 open files allowed, minimum of 40000 recommended. See the Neo4j manual.
Started neo4j (pid 1113). It is available at http://0.0.0.0:7474/
There may be a short delay until the server is ready.
See /usr/local/neo4j/logs/neo4j.log for current status.
[ec2-user@ip-172-31-27-29 ~]$ tail -f /usr/local/neo4j/logs/neo4j.log
nohup: ignoring input
2019-09-19 00:58:35.198+0000 INFO  ===== Neo4j 3.5.9 =====
2019-09-19 00:58:35.227+0000 INFO  Starting...
2019-09-19 00:58:40.073+0000 INFO  Bolt enabled on 0.0.0.0:7687.
2019-09-19 00:58:43.245+0000 INFO  Started.
2019-09-19 00:58:45.465+0000 INFO  Remote interface available at http://localhost:7474/
```

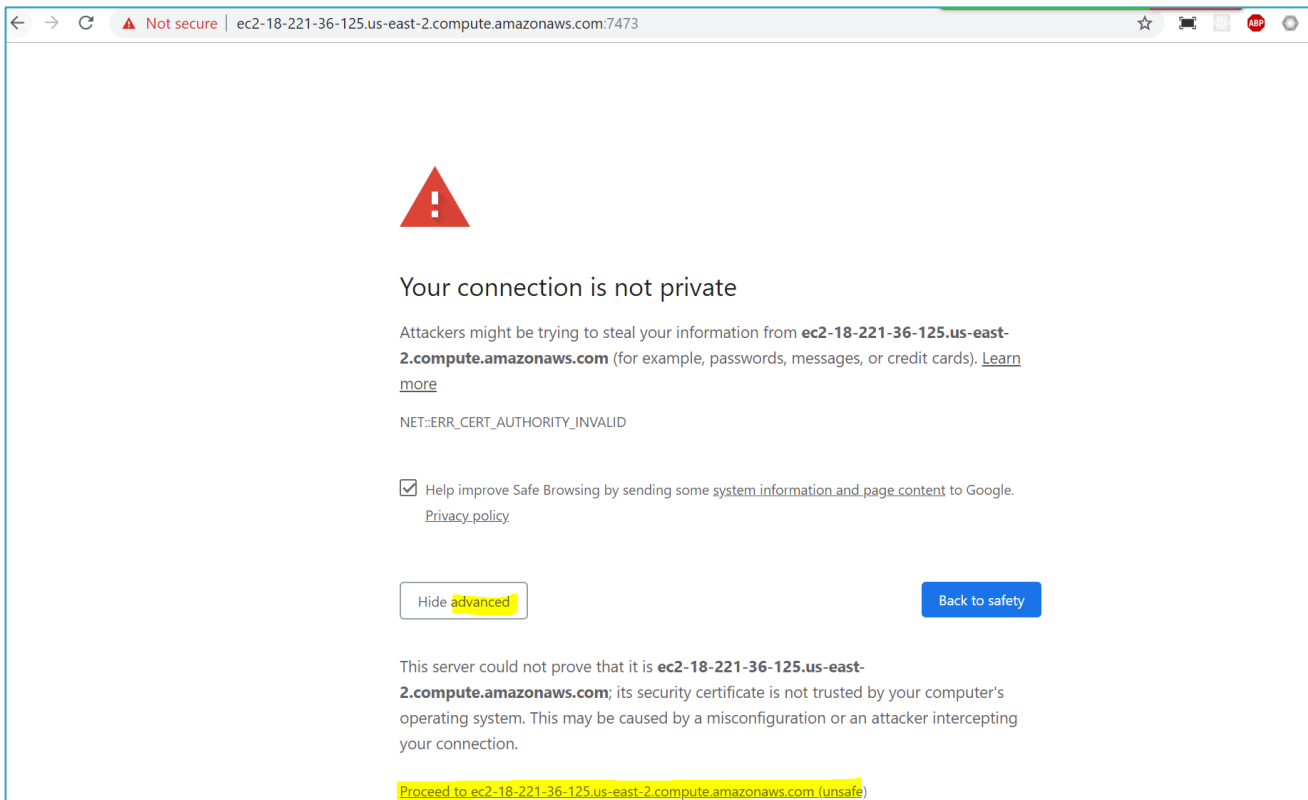
11. Neo4J Server setup is now complete. Connect to Neo4j server running in AWS EC2 instance by typing below URL on your laptop chrome browser

- https://<YOUR_PUBLIC_DNS>:7473/
E.g. <https://ec2-18-221-36-125.us-east-2.compute.amazonaws.com:7473/>

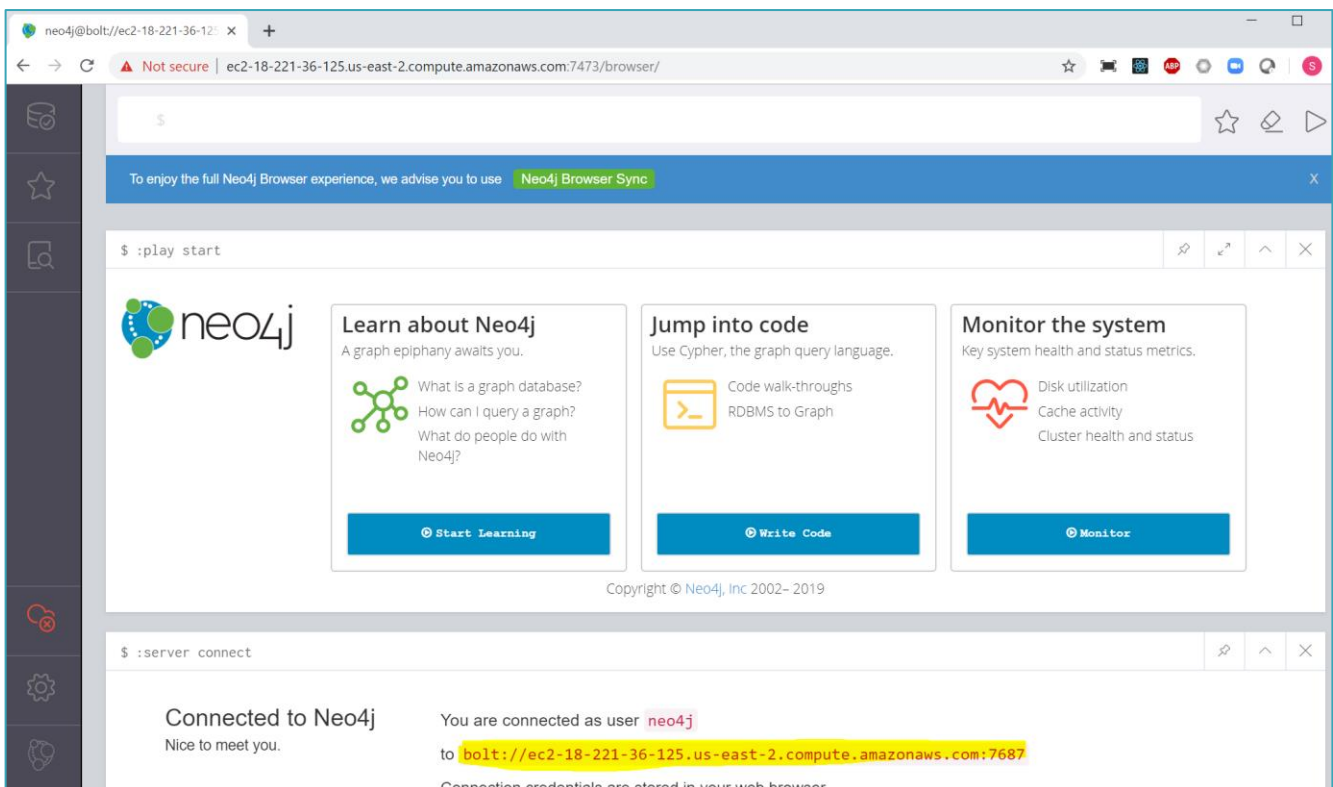
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
Gajanan Big ...	i-00d1b64c7bdc6d715	t2.micro	us-east-2b	running	2/2 checks ...	None	ec2-18-222-137-53.us-...
Shree Big D...	i-00dfdae6ec8d6b52	t2.micro	us-east-2b	running	2/2 checks ...	None	ec2-18-221-36-125.us-...
i-03932ab3f903f890		m4.large	us-east-2a	running	2/2 checks ...	None	ec2-18-217-127-254.us...
i-04f4061b45bc19668		m4.large	us-east-2a	running	2/2 checks ...	None	ec2-18-217-190-153.us...
i-0ba18955a023061a4		m4.large	us-east-2a	running	2/2 checks ...	None	ec2-18-218-165.us-...

Instance: i-00dfdae6ec8d6b52 (Shree Big Data instance)		Public DNS: ec2-18-221-36-125.us-east-2.compute.amazonaws.com
Description		
Instance ID	i-00dfdae6ec8d6b52	Public DNS (IPv4)
Instance state	running	ec2-18-221-36-125.us-east-2.compute.amazonaws.com
Instance type	t2.micro	IPv4 Public IP
Elastic IPs		18.221.36.125
Availability zone	us-east-2b	IPv6 IPs
Security groups	launch-wizard-6, view inbound rules, view outbound rules	Private DNS
Scheduled events	No scheduled events	ip-172-31-27-29.us-east-2.compute.internal
		Private IPs
		172.31.27.29
		Secondary private IPs
		VPC ID
		vpc-484b8d20

12. Once loaded connect to Neo4j using 'neo4j' as userid & password. On your first login, the neo4j app asks you to update the password, update the password to "root" & keep it handy for later use.



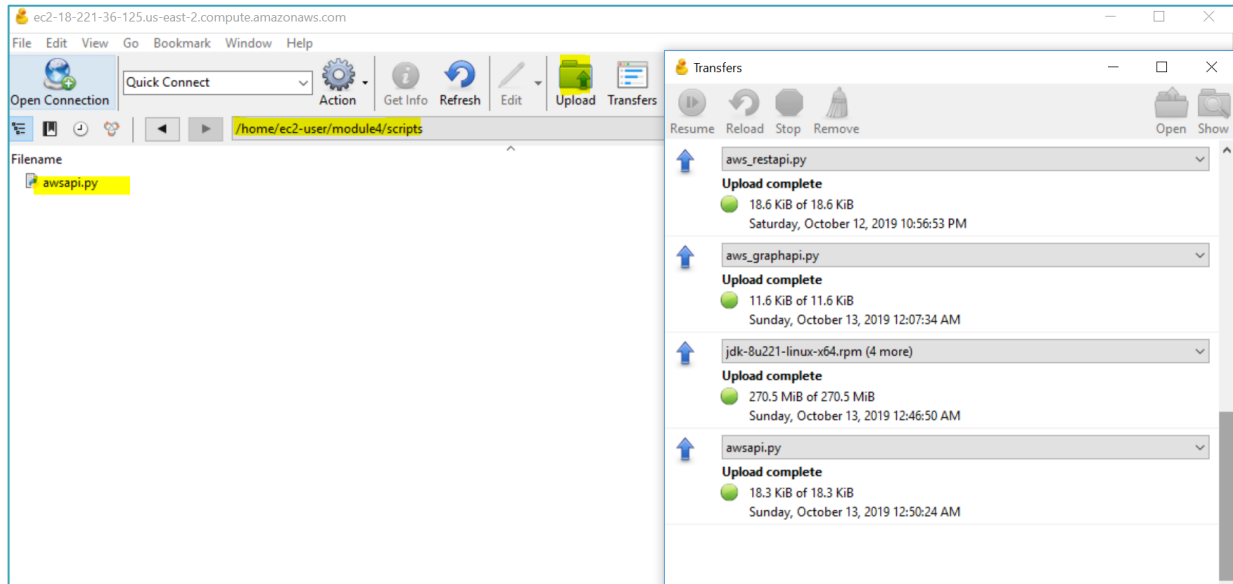
After successful login you should see a screen like below:



Note: Neo4j installation is now complete.

IMPORTING DATA INTO Neo4J

1. Using Cyberduck please upload the scripts into the module4/scripts folder within EC2.
Note: If the scripts folder does not exist then create it.

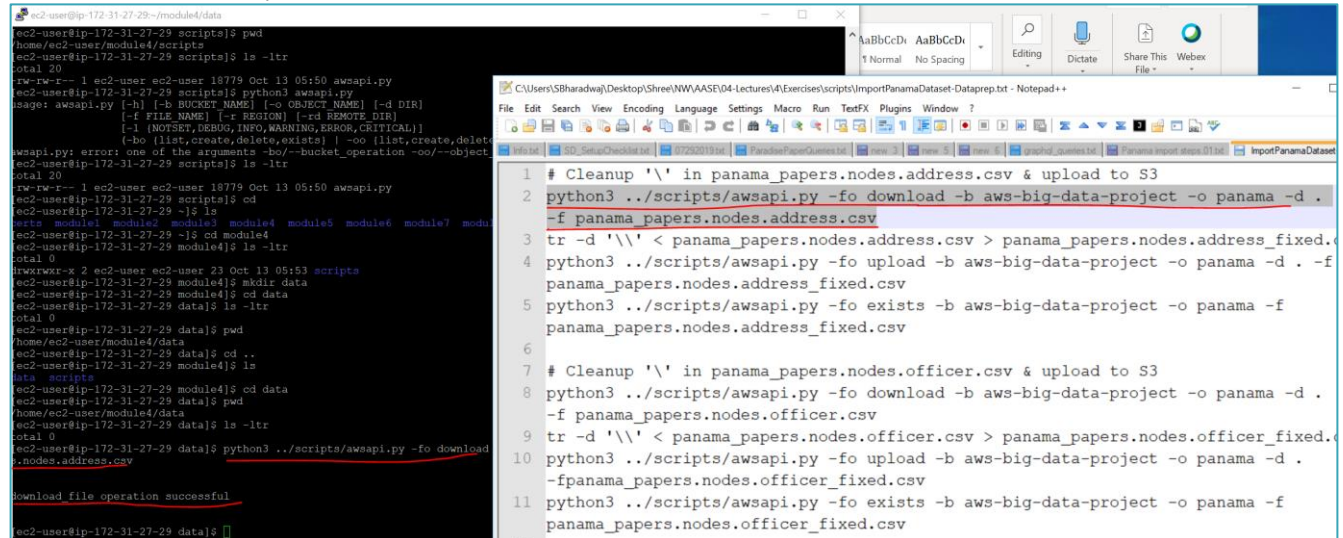


2. Login to EC2 instance and create the data folder within the module4 folder.
mkdir data
cd data

```
ec2-user@ip-172-31-27-29:~/module4/data
[ec2-user@ip-172-31-27-29 scripts]$ pwd
/home/ec2-user/module4/scripts
[ec2-user@ip-172-31-27-29 scripts]$ ls -ltr
total 20
-rw-rw-r-- 1 ec2-user ec2-user 18779 Oct 13 05:50 awsapi.py
[ec2-user@ip-172-31-27-29 scripts]$ python3 awsapi.py
usage: awsapi.py [-h] [-b BUCKET_NAME] [-o OBJECT_NAME] [-d DIR]
                [-f FILE_NAME] [-r REGION] [-rd REMOTE_DIR]
                [-l {NOTSET,DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                (-bo {list,create,delete,exists} | -oo {list,create,delete,exists} | -fo {upload,download,delete,exists})
awsapi.py: error: one of the arguments -bo/--bucket_operation -oo/--object_operation -fo/--file_operation is required
[ec2-user@ip-172-31-27-29 scripts]$ ls -ltr
total 20
-rw-rw-r-- 1 ec2-user ec2-user 18779 Oct 13 05:50 awsapi.py
[ec2-user@ip-172-31-27-29 scripts]$ cd
[ec2-user@ip-172-31-27-29 ~]$ ls
certs  module1  module2  module3  module4  module5  module6  module7  module8  module9  notebooks  Untitled.ipynb
[ec2-user@ip-172-31-27-29 ~]$ cd module4
[ec2-user@ip-172-31-27-29 module4]$ ls -ltr
total 0
drwxrwxr-x 2 ec2-user ec2-user 23 Oct 13 05:53 scripts
[ec2-user@ip-172-31-27-29 module4]$ mkdir data
[ec2-user@ip-172-31-27-29 module4]$ cd data
[ec2-user@ip-172-31-27-29 data]$ ls -ltr
total 0
[ec2-user@ip-172-31-27-29 data]$ pwd
/home/ec2-user/module4/data
[ec2-user@ip-172-31-27-29 data]$ cd ..
[ec2-user@ip-172-31-27-29 module4]$ ls
data  scripts
[ec2-user@ip-172-31-27-29 module4]$ cd data
[ec2-user@ip-172-31-27-29 data]$ pwd
/home/ec2-user/module4/data
[ec2-user@ip-172-31-27-29 data]$ ls -ltr
total 0
[ec2-user@ip-172-31-27-29 data]$
```


3. Within the data folder, use the commands given in the ImportPanamaDataset-Dataprep.txt to prep the data for importing it into Neo4J.

Note: Please verify the counts as indicated in the text file.



The screenshot shows a terminal window on the left and a Notepad++ editor on the right. The terminal window displays a series of commands and their outputs, including directory listings, file downloads, and directory creations. The Notepad++ editor shows a text file named 'ImportPanamaDataset-Dataprep.txt' containing a series of commands for data preparation, including cleanup, download, and upload operations.

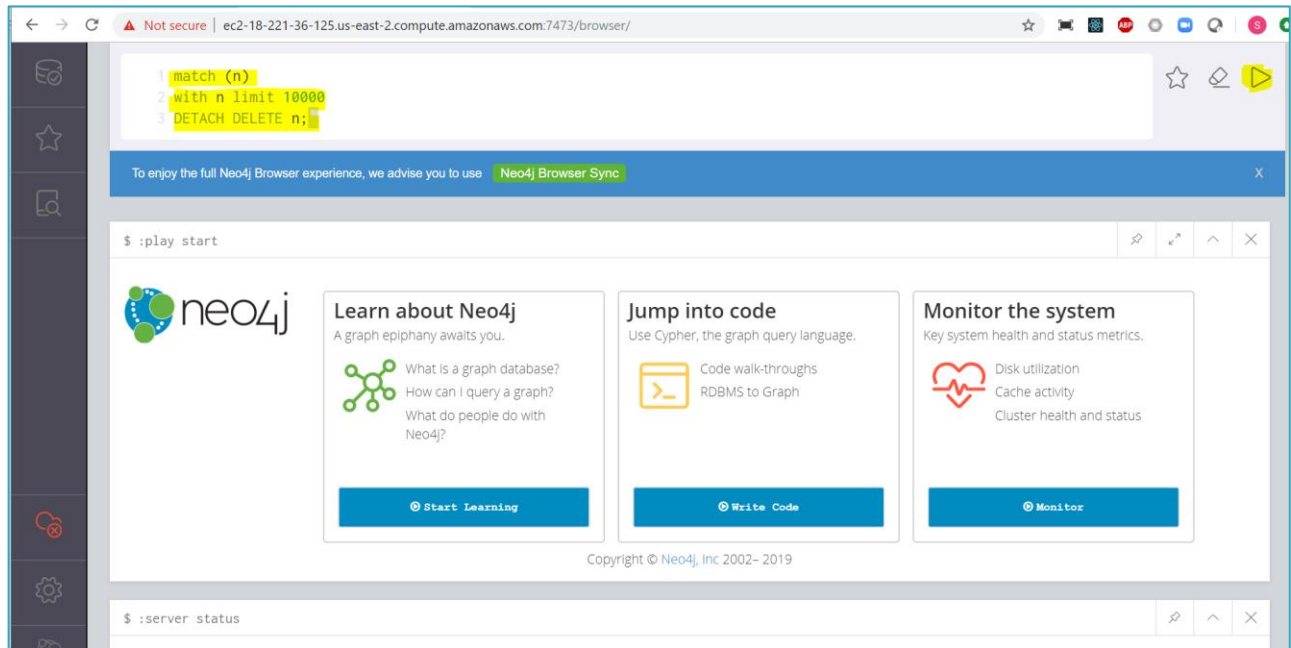
```
ec2-user@ip-172-31-27-29:~/module4/data$ pwd
/home/ec2-user/module4/scripts
ec2-user@ip-172-31-27-29:~/module4/scripts$ ls -ltr
total 20
-rw-rw-r-- 1 ec2-user ec2-user 18779 Oct 13 05:50 awsapi.py
ec2-user@ip-172-31-27-29:~/module4/scripts$ python3 awsapi.py
usage: awsapi.py [-h] [-b BUCKET_NAME] [-o OBJECT_NAME] [-d DIR]
                 [-f FILE_NAME] [-r REGION] [-rd REMOTE_DIR]
                 [-l (NOTSET,DEBUG,INFO,WARNING,ERROR,CRITICAL)]
                 [-bo (list,create,delete,exists)] [-oo (list,create,delete)]
awsapi.py: error: one of the arguments -bo/-bucket_operation -oo/--object_operation is required
ec2-user@ip-172-31-27-29:~/module4/scripts$ ls -ltr
total 20
-rw-rw-r-- 1 ec2-user ec2-user 18779 Oct 13 05:50 awsapi.py
ec2-user@ip-172-31-27-29:~/module4/scripts$ cd
ec2-user@ip-172-31-27-29:~$ ls
data scripts
ec2-user@ip-172-31-27-29:~$ cd module4
ec2-user@ip-172-31-27-29:~/module4$ ls -ltr
total 0
lrwxrwxr-x 2 ec2-user ec2-user 23 Oct 13 05:53 scripts
ec2-user@ip-172-31-27-29:~/module4$ mkdir data
ec2-user@ip-172-31-27-29:~/module4$ cd data
ec2-user@ip-172-31-27-29:~/module4/data$ ls -ltr
total 0
ec2-user@ip-172-31-27-29:~/module4/data$ python3 ../scripts/awsapi.py -fo download
panama_papers.nodes.address.csv
Download file operation successful
ec2-user@ip-172-31-27-29:~/module4/data$
```

```
1 # Cleanup '\\' in panama_papers.nodes.address.csv & upload to S3
2 python3 ../scripts/awsapi.py -fo download -b aws-big-data-project -o panama -d .
-f panama_papers.nodes.address.csv
3 tr -d '\\' < panama_papers.nodes.address.csv > panama_papers.nodes.address_fixed.csv
4 python3 ../scripts/awsapi.py -fo upload -b aws-big-data-project -o panama -d . -f
panama_papers.nodes.address_fixed.csv
5 python3 ../scripts/awsapi.py -fo exists -b aws-big-data-project -o panama -f
panama_papers.nodes.address_fixed.csv
6
7 # Cleanup '\\' in panama_papers.nodes.officer.csv & upload to S3
8 python3 ../scripts/awsapi.py -fo download -b aws-big-data-project -o panama -d .
-f panama_papers.nodes.officer.csv
9 tr -d '\\' < panama_papers.nodes.officer.csv > panama_papers.nodes.officer_fixed.csv
10 python3 ../scripts/awsapi.py -fo upload -b aws-big-data-project -o panama -d .
-f panama_papers.nodes.officer_fixed.csv
11 python3 ../scripts/awsapi.py -fo exists -b aws-big-data-project -o panama -f
panama_papers.nodes.officer_fixed.csv
```

VISUALIZING CONNECTED DATA IN NEO4J

1. Login to Neo4J database
https://<EC2_PUBLIC_DNS>:7473/browser/ with credentials (user : neo4j)
Ex: <https://ec2-18-221-36-125.us-east-2.compute.amazonaws.com:7473/>
2. Delete all existing data using the cypher query below
match (n)
with n limit 10000

DETACH DELETE n;



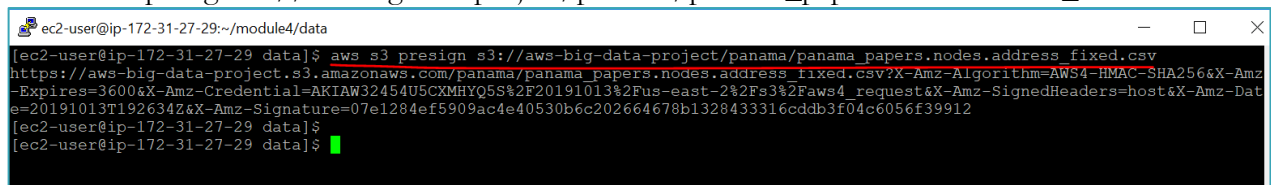
3. Load the data/files from S3 into Neo4J. To do so, we will need

- a) Create a pre authenticated URL
- b) Use the pre authenticated URL to import data into Neo4J

a) Create a pre authenticated URL for panama_papers.nodes.address_fixed.csv

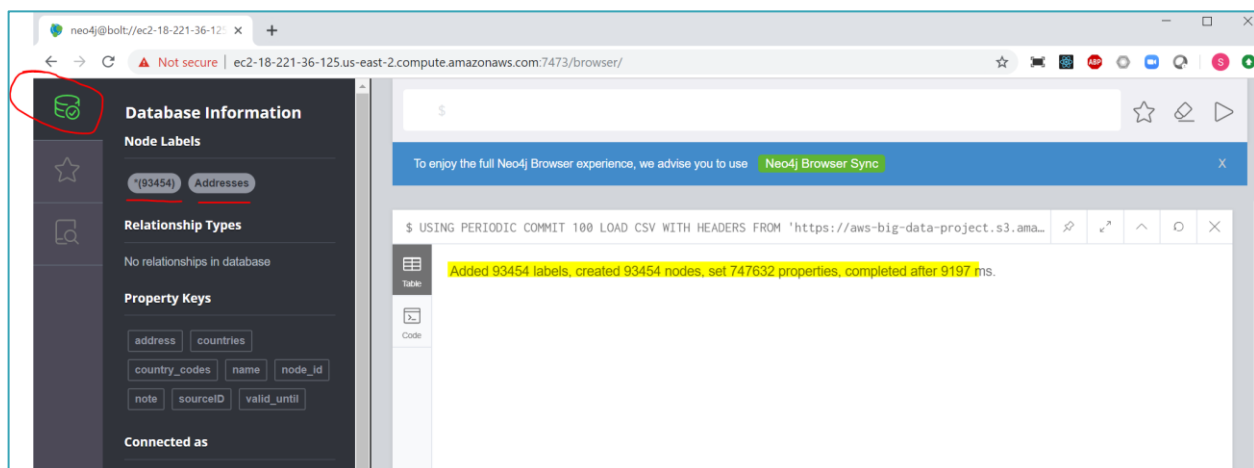
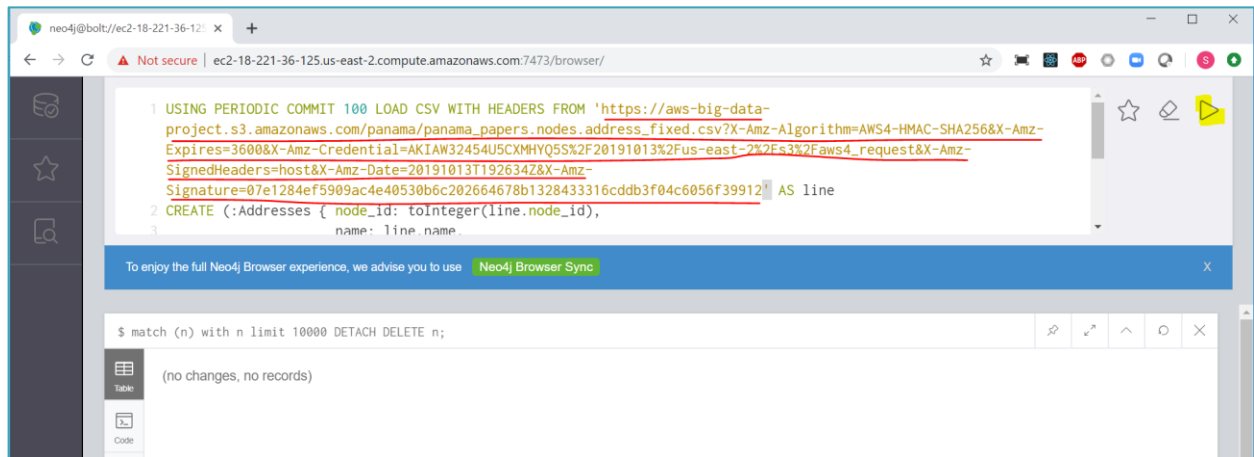
aws s3 presign s3://<s3 bucket> /panama/panama_papers.nodes.address_fixed.csv

Ex: aws s3 presign s3://aws-big-data-project/panama/panama_papers.nodes.address_fixed.csv



b) Use the pre authenticated URL (output from the previous query) to import panama_papers.nodes.address_fixed.csv data into Neo4J.

Note: Refer to ImportPanamaDataset-UsingCypher.txt



- c) Repeat the same process to import rest of the data from S3 into Neo4J. Please refer to the ImportPanamaDataset-UsingCypher.txt script to do the same.
- d) Run Queries to visualize connected dataset to derive insights such as basic statistics
MATCH (n)
RETURN labels(n), count(*)
ORDER BY count(*) DESC

The screenshot shows the Neo4j Browser interface. On the left, the 'Database Information' sidebar is visible, showing 'Node Labels' (Addresses, Entities, Intermediaries, Officers) and 'Relationship Types' (intermediary_of, officer_of, registered_address). The main area displays a Cypher query:

```
1 MATCH (n)
2 RETURN labels(n), count(*)
3 ORDER BY count(*) DESC
```

The results table shows the following data:

labels(n)	count(*)
["Officers"]	238402
["Entities"]	213634
["Addresses"]	93454
["Intermediaries"]	14110

Another insight : The distribution of degrees for our entities shows a typical power law: there are some addresses which have 127 companies registered and some people who have almost 90 shell companies registered to their name.

MATCH (n)

WITH labels(n) AS type, size((n)--()) AS degree

RETURN type,

max(degree) AS max, round(avg(degree)) AS avg, round(stdev(degree)) AS stdev

The screenshot shows the Neo4j Browser interface with a more complex Cypher query:

```
1 MATCH (n)
2 WITH labels(n) AS type, size((n)--()) AS degree
3 RETURN type,
4 max(degree) AS max, round(avg(degree)) AS avg, round(stdev(degree)) AS stdev
```

The results table shows the following data:

type	max	avg	stdev
["Addresses"]	1007	2.0	7.0
["Intermediaries"]	7016	15.0	121.0
["Entities"]	1007	2.0	4.0
["Officers"]	3883	2.0	9.0

Exercise 4 is now complete