University of Pittsburgh
School of Computing and Information

INFSCI 2750: Cloud Computing

Spring 2020

Mini Project 3

Submitted by:

SL. No.	Name	Email address
1	Debdas Ghosh	deg107@pitt.edu
2	Piu Mallick	pim16@pitt.edu

Objective

The objective of this mini project is to learn and develop applications using **Apache Cassandra**. We will be using the previously assigned VMs.

VM Type	**IP Address**	**VM Name**
Master	134.209.160.211	CC-MON-25
Slave 1	64.225.17.118	CC-MON-26
Slave 2	138.197.96.66	CC-MON-27

Part 1: Setting up Cassandra: (50 points)

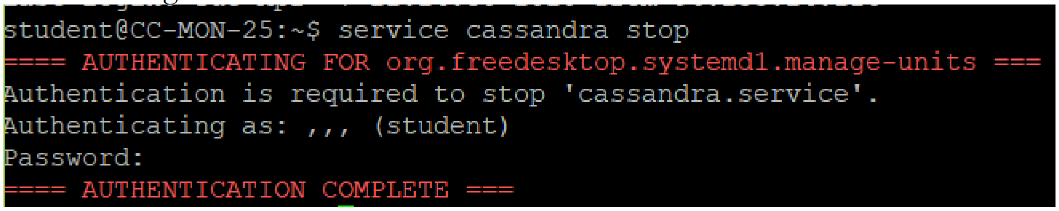
The first task is to configure a **Cassandra** distribution in our cluster of **VM**s. The entire **Cassandra** setup should be configured on top of a **two-node** or **three-node** cluster. As Cassandra has a "master-less" architecture, all of the Cassandra nodes can be configured in the same way.

Solution:

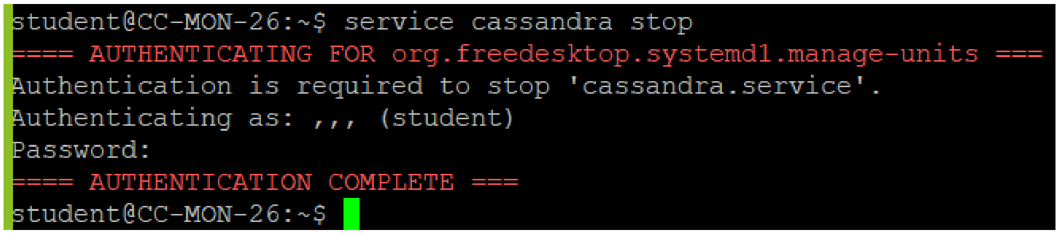
Setting up **Cassandra on a cluster with Debian package installation** is easy when compared to the manual set-up of **Hadoop** and **Spark**.
Here are the steps that we followed:

```
# ssh on to Master (134.209.160.211) and Slaves (64.225.17.118 and 138.197.96.66) and
# run the following commands:
echo "deb http://www.apache.org/dist/cassandra/debian 311x main" \ | sudo tee -a
/etc/apt/sources.list.d/cassandra.sources.list
curl https://www.apache.org/dist/cassandra/KEYS | sudo apt-key add -
sudo apt-get update
sudo apt-get install Cassandra
sudo service cassandra stop
```

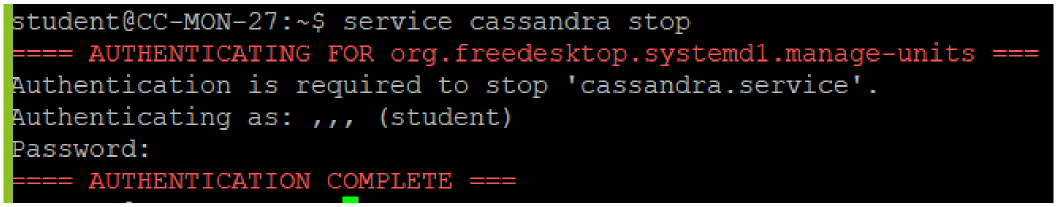
The following screenshots are below as evidence:



```
student@CC-MON-25:~$ service cassandra stop
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===
Authentication is required to stop 'cassandra.service'.
Authenticating as: ,,, (student)
Password:
==== AUTHENTICATION COMPLETE ===
```



```
student@CC-MON-26:~$ service cassandra stop
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===
Authentication is required to stop 'cassandra.service'.
Authenticating as: ,,, (student)
Password:
==== AUTHENTICATION COMPLETE ===
student@CC-MON-26:~$
```



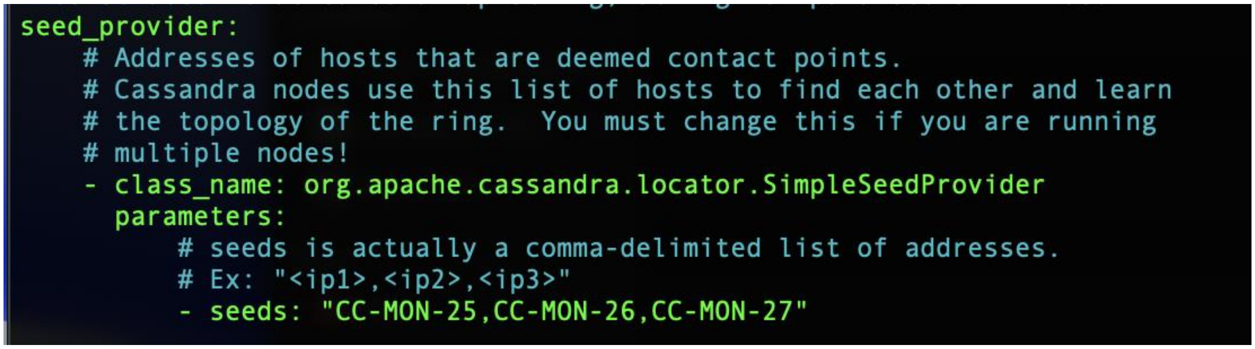
```
student@CC-MON-27:~$ service cassandra stop
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===
Authentication is required to stop 'cassandra.service'.
Authenticating as: ,,, (student)
Password:
==== AUTHENTICATION COMPLETE ===
```

Then, we modified the file ‘**cassandra.yaml’** on each node (**Master** node and 2 **Slave** nodes)

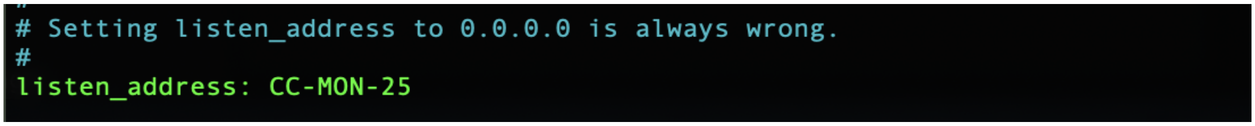
```
nano /etc/cassandra/cassandra.yaml
```

Master Node (CC-MON-25)
Following parameters were changed:

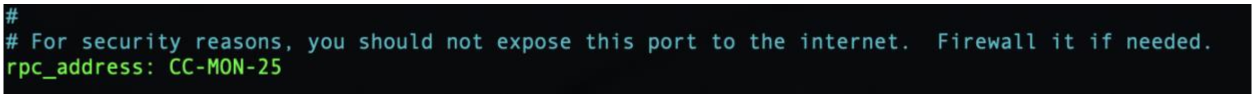
```
seeds: “CC-MON-25,CC-MON-26,CC-MON-27”
listen_address: CC-MON-25
rpc_address: CC-MON-25
```



```
seed_provider:
    # Addresses of hosts that are deemed contact points.
    # Cassandra nodes use this list of hosts to find each other and learn
    # the topology of the ring.  You must change this if you are running
    # multiple nodes!
    - class_name: org.apache.cassandra.locator.SimpleSeedProvider
      parameters:
          # seeds is actually a comma-delimited list of addresses.
          # Ex: "<ip1>,<ip2>,<ip3>"
          - seeds: "CC-MON-25,CC-MON-26,CC-MON-27"
```

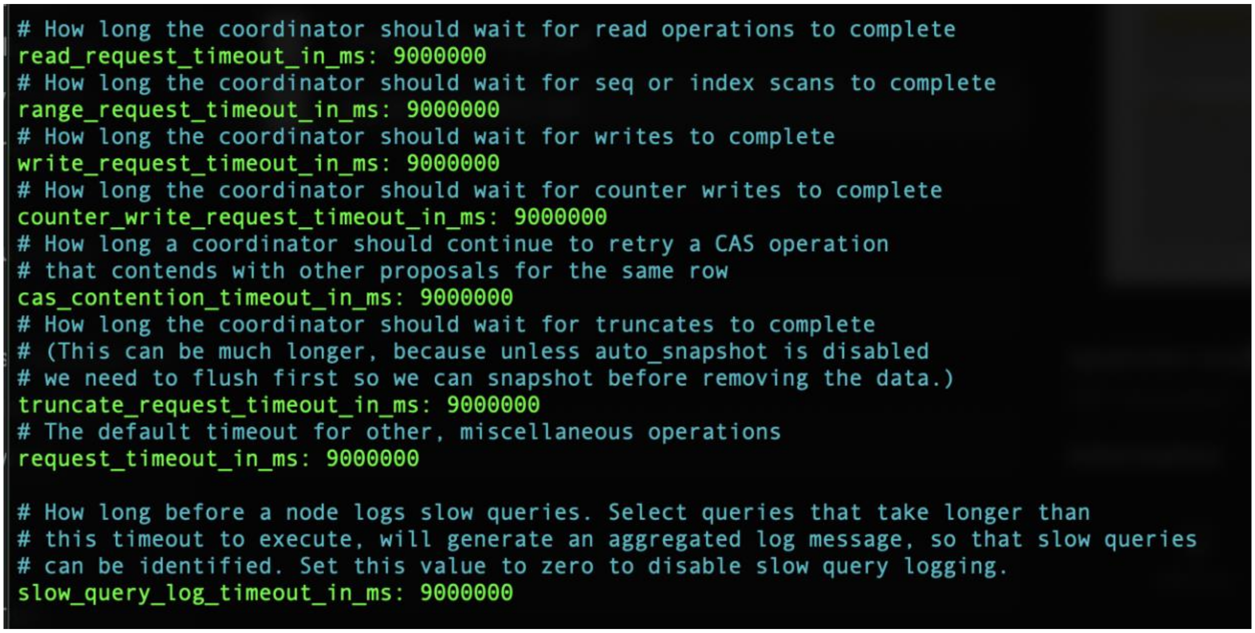


```
# Setting listen_address to 0.0.0.0 is always wrong.
#
listen_address: CC-MON-25
```



```
#
# For security reasons, you should not expose this port to the internet.  Firewall it if needed.
rpc_address: CC-MON-25
```

The following parameters were changed to **9000000** as we were getting '**TimeOut Error**' while executing the programs for Part 2 and Part 3.

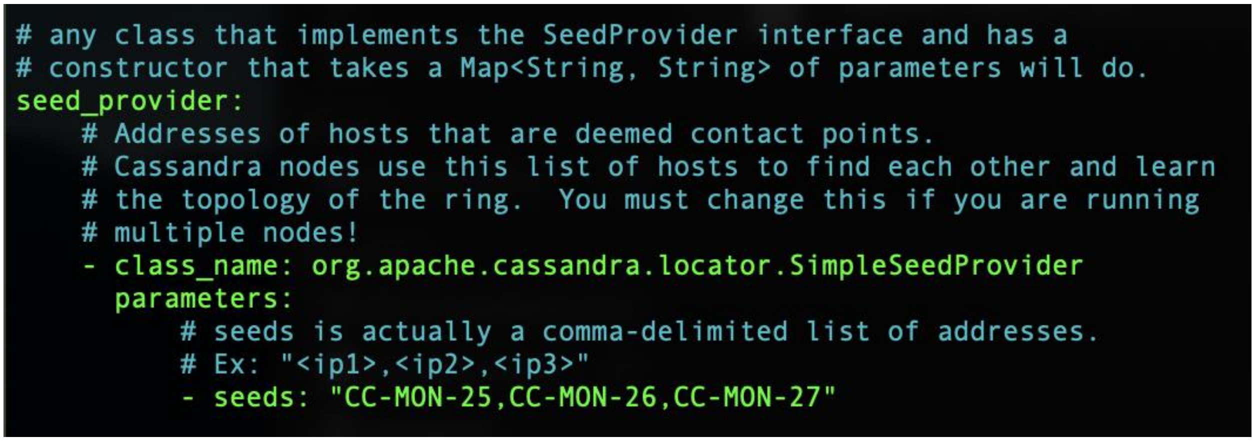


```
# How long the coordinator should wait for read operations to complete
read_request_timeout_in_ms: 9000000
# How long the coordinator should wait for seq or index scans to complete
range_request_timeout_in_ms: 9000000
# How long the coordinator should wait for writes to complete
write_request_timeout_in_ms: 9000000
# How long the coordinator should wait for counter writes to complete
counter_write_request_timeout_in_ms: 9000000
# How long a coordinator should continue to retry a CAS operation
# that contends with other proposals for the same row
cas_contention_timeout_in_ms: 9000000
# How long the coordinator should wait for truncates to complete
# (This can be much longer, because unless auto_snapshot is disabled
# we need to flush first so we can snapshot before removing the data.)
truncate_request_timeout_in_ms: 9000000
# The default timeout for other, miscellaneous operations
request_timeout_in_ms: 9000000

# How long before a node logs slow queries. Select queries that take longer than
# this timeout to execute, will generate an aggregated log message, so that slow queries
# can be identified. Set this value to zero to disable slow query logging.
slow_query_log_timeout_in_ms: 9000000
```

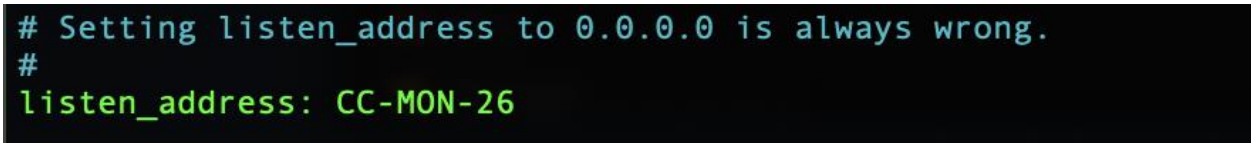
Slave 1 Node (CC-MON-26)

Following parameters were changed:

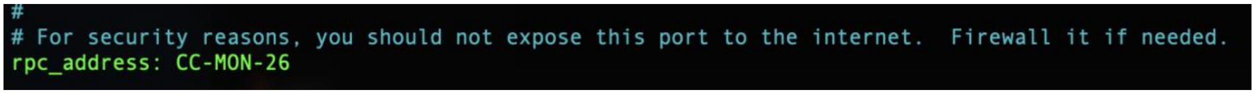
```
seeds: "CC-MON-25,CC-MON-26,CC-MON-27"
listen_address: CC-MON-26
rpc_address: CC-MON-26
```



```
# any class that implements the SeedProvider interface and has a
# constructor that takes a Map<String, String> of parameters will do.
seed_provider:
    # Addresses of hosts that are deemed contact points.
    # Cassandra nodes use this list of hosts to find each other and learn
    # the topology of the ring.  You must change this if you are running
    # multiple nodes!
    - class_name: org.apache.cassandra.locator.SimpleSeedProvider
      parameters:
          # seeds is actually a comma-delimited list of addresses.
          # Ex: "<ip1>,<ip2>,<ip3>"
          - seeds: "CC-MON-25,CC-MON-26,CC-MON-27"
```

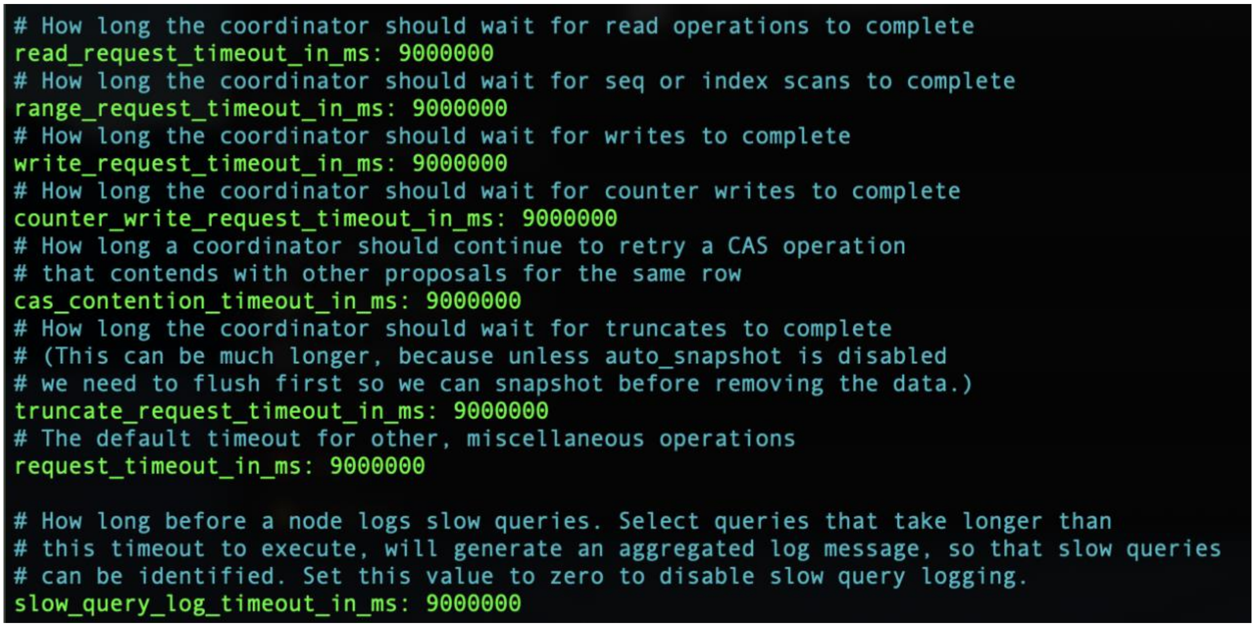


```
# Setting listen_address to 0.0.0.0 is always wrong.
#
listen_address: CC-MON-26
```



```
#
# For security reasons, you should not expose this port to the internet.  Firewall it if needed.
rpc_address: CC-MON-26
```

The following parameters were changed to **9000000** as we were getting '**TimeOut Error**' while executing the programs for Part 2 and Part 3.

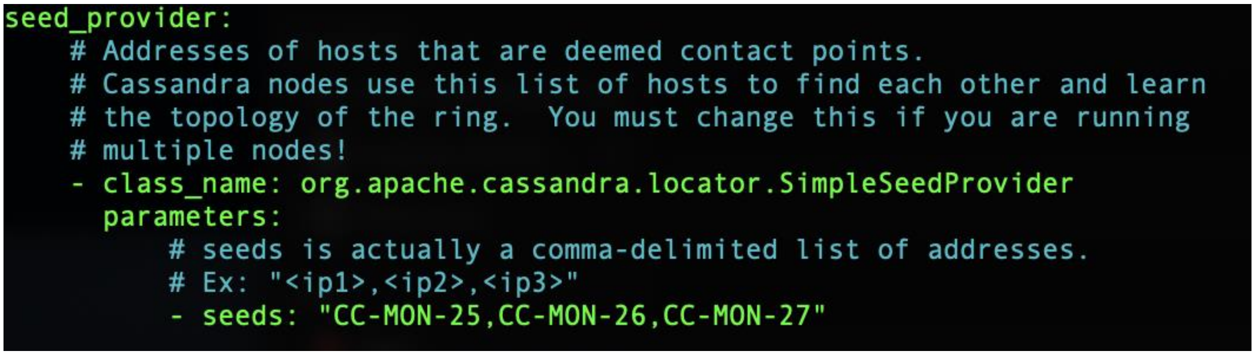


```
# How long the coordinator should wait for read operations to complete
read_request_timeout_in_ms: 9000000
# How long the coordinator should wait for seq or index scans to complete
range_request_timeout_in_ms: 9000000
# How long the coordinator should wait for writes to complete
write_request_timeout_in_ms: 9000000
# How long the coordinator should wait for counter writes to complete
counter_write_request_timeout_in_ms: 9000000
# How long a coordinator should continue to retry a CAS operation
# that contends with other proposals for the same row
cas_contention_timeout_in_ms: 9000000
# How long the coordinator should wait for truncates to complete
# (This can be much longer, because unless auto_snapshot is disabled
# we need to flush first so we can snapshot before removing the data.)
truncate_request_timeout_in_ms: 9000000
# The default timeout for other, miscellaneous operations
request_timeout_in_ms: 9000000

# How long before a node logs slow queries. Select queries that take longer than
# this timeout to execute, will generate an aggregated log message, so that slow queries
# can be identified. Set this value to zero to disable slow query logging.
slow_query_log_timeout_in_ms: 9000000
```

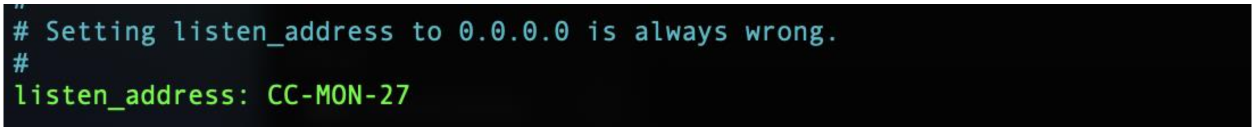
Slave 2 Node (CC-MON-27)

Following parameters were changed:

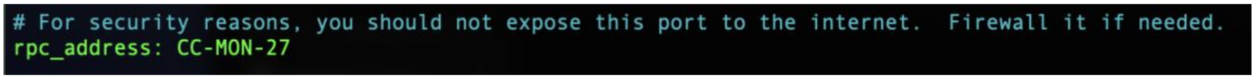
```
seeds: "CC-MON-25,CC-MON-26,CC-MON-27"
listen_address: CC-MON-27
rpc_address: CC-MON-27
```



```
seed_provider:
    # Addresses of hosts that are deemed contact points.
    # Cassandra nodes use this list of hosts to find each other and learn
    # the topology of the ring.  You must change this if you are running
    # multiple nodes!
    - class_name: org.apache.cassandra.locator.SimpleSeedProvider
      parameters:
          # seeds is actually a comma-delimited list of addresses.
          # Ex: "<ip1>,<ip2>,<ip3>"
          - seeds: "CC-MON-25,CC-MON-26,CC-MON-27"
```

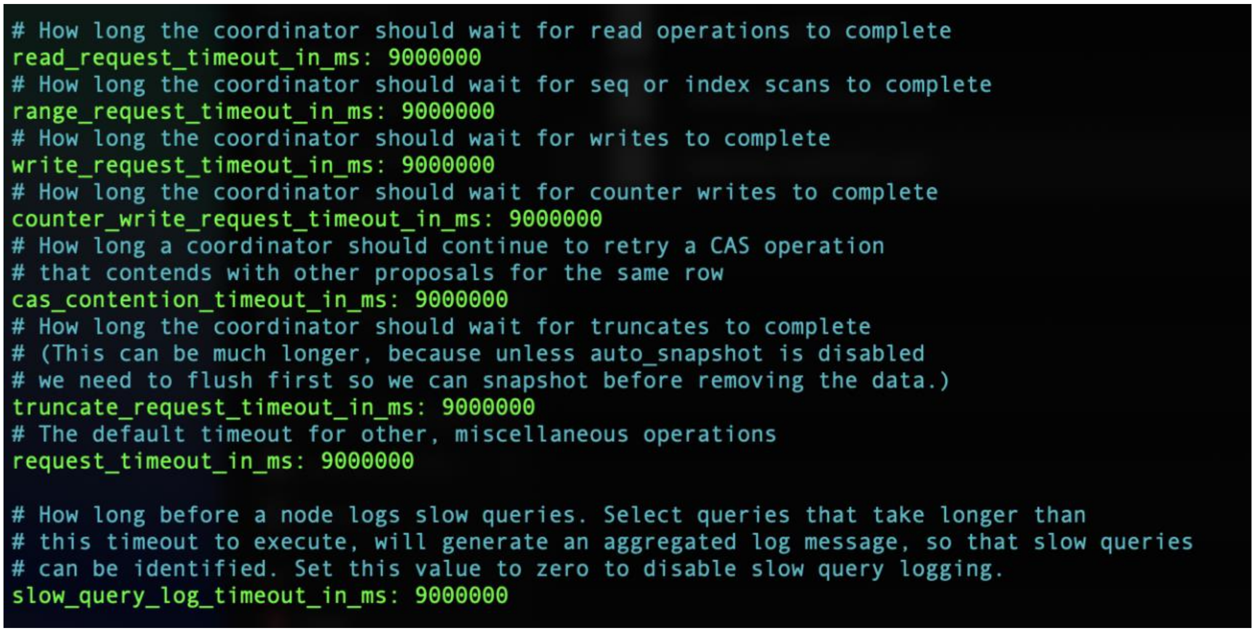


```
# Setting listen_address to 0.0.0.0 is always wrong.
#
listen_address: CC-MON-27
```



```
# For security reasons, you should not expose this port to the internet.  Firewall it if needed.
rpc_address: CC-MON-27
```

The following parameters were changed to **9000000** as we were getting '**TimeOut Error**' while executing the programs for Part 2 and Part 3.



```
# How long the coordinator should wait for read operations to complete
read_request_timeout_in_ms: 9000000
# How long the coordinator should wait for seq or index scans to complete
range_request_timeout_in_ms: 9000000
# How long the coordinator should wait for writes to complete
write_request_timeout_in_ms: 9000000
# How long the coordinator should wait for counter writes to complete
counter_write_request_timeout_in_ms: 9000000
# How long a coordinator should continue to retry a CAS operation
# that contends with other proposals for the same row
cas_contention_timeout_in_ms: 9000000
# How long the coordinator should wait for truncates to complete
# (This can be much longer, because unless auto_snapshot is disabled
# we need to flush first so we can snapshot before removing the data.)
truncate_request_timeout_in_ms: 9000000
# The default timeout for other, miscellaneous operations
request_timeout_in_ms: 9000000

# How long before a node logs slow queries. Select queries that take longer than
# this timeout to execute, will generate an aggregated log message, so that slow queries
# can be identified. Set this value to zero to disable slow query logging.
slow_query_log_timeout_in_ms: 9000000
```

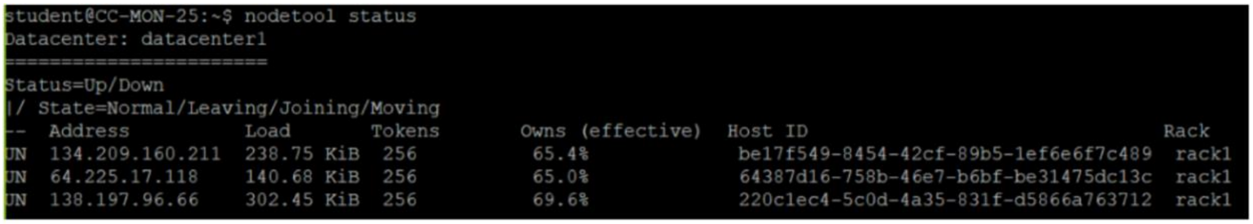
With all the prep-work done (as stated above), we can start the Cassandra cluster by executing the following command on each node:

```
sudo cassandra -Rf
```

On a new ssh session, the following command is executed to check the status of the **Cassandra** cluster.

```
nodetool status
```

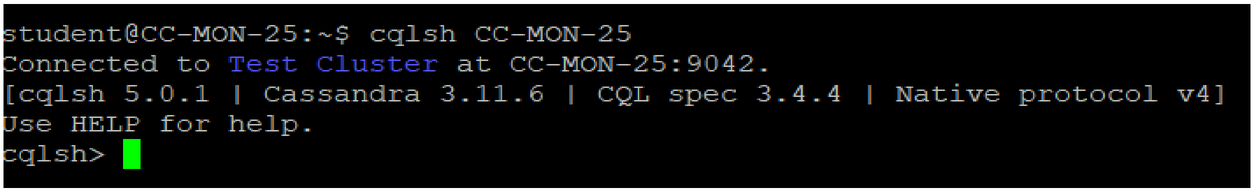
Screenshot below:



```
student@CC-MON-25:~$ nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address          Load        Tokens       Owns (effective)  Host ID                               Rack
UN  134.209.160.211  238.75 KiB  256          65.4%             be17f549-8454-42cf-89b5-1ef6e6f7c489  rack1
UN  64.225.17.118    140.68 KiB  256          65.0%             64387d16-758b-46e7-b6bf-be31475dc13c  rack1
UN  138.197.96.66    302.45 KiB  256          69.6%             220c1ec4-5c0d-4a35-831f-d5866a763712  rack1
```

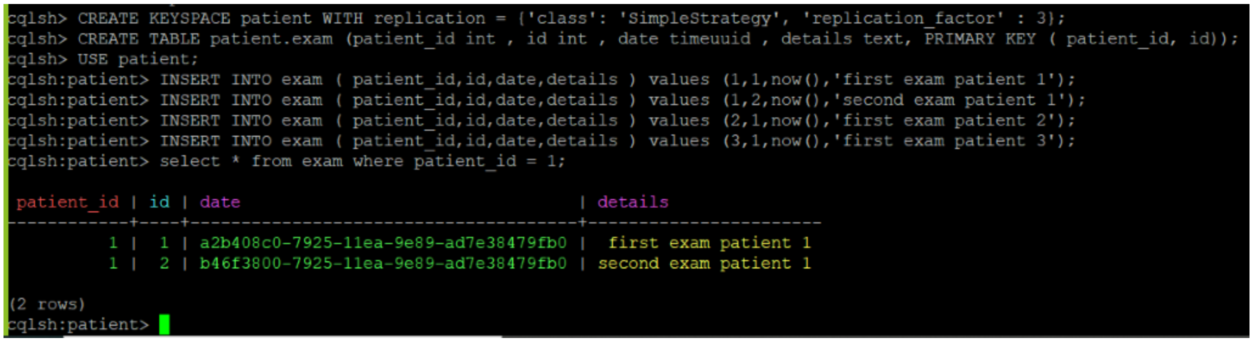
To start a **Cassandra CQL** shell on the cluster, simply run the following command.

```
cqlsh CC-MON-25
```



```
student@CC-MON-25:~$ cqlsh CC-MON-25
Connected to Test Cluster at CC-MON-25:9042.
[cqlsh 5.0.1 | Cassandra 3.11.6 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh>
```

<u>Sample example of Cassandra execution</u> from the tutorial (creating a keyspace **patient**, followed by creating a table **exam** and then inserting 4 **patient records**):



```
cqlsh> CREATE KEYSPACE patient WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 3};
cqlsh> CREATE TABLE patient.exam (patient_id int , id int , date timeuuid , details text, PRIMARY KEY ( patient_id, id));
cqlsh> USE patient;
cqlsh:patient> INSERT INTO exam ( patient_id,id,date,details ) values (1,1,now(),'first exam patient 1');
cqlsh:patient> INSERT INTO exam ( patient_id,id,date,details ) values (1,2,now(),'second exam patient 1');
cqlsh:patient> INSERT INTO exam ( patient_id,id,date,details ) values (2,1,now(),'first exam patient 2');
cqlsh:patient> INSERT INTO exam ( patient_id,id,date,details ) values (3,1,now(),'first exam patient 3');
cqlsh:patient> select * from exam where patient_id = 1;

 patient_id | id | date                                 | details
------------+----+--------------------------------------+-----------------------
          1 |  1 | a2b408c0-7925-11ea-9e89-ad7e38479fb0 |  first exam patient 1
          1 |  2 | b46f3800-7925-11ea-9e89-ad7e38479fb0 | second exam patient 1

(2 rows)
cqlsh:patient>
```

<u>Please note</u>: It was made sure that previous **Hadoop** and **Spark** services are all shut down to empty the memory use before starting the **Cassandra** nodes on all **VM**s.

-R parameter was used when **Cassandra** was run with the **root user**.

The project's source code was written in **JAVA** and used **Maven** as **<u>build management tool</u>**.
The **JAVA code** and **pom.xml** file for the project is in the **mini-project-03** folder.

The **mini-project-03- 1.0.0-jar-with-dependencies.jar** file was built locally with maven to include all the source code provided and was uploaded to the VM server for running.

Note here we need to build the jar file with dependencies in order to successfully run it.

```
mvn package
scp target/mini-project-03-1.0.0-jar-with-dependencies.jar root@134.209.160.211:~/
```

Other than the JAVA code and jar file, we also provided all the CQL commands used in this project in **mini-project-03/src/cql.txt** .

Part 2: Import Data into Cassandra: (25 points)

As a part of the project, we will be working with the **log data set** which has been provided in '**access_log.zip**' in the **Mini Project 1**.

We need to use **CQL** (**Cassandra Query Language**) or **JAVA driver** of **Cassandra** to import the access logs into Cassandra.

We need to create one **keyspace** and one **table** at least in **Cassandra** to store all the logs.

Solution:

The **ImportData.java** file is the source code for importing the **access_log** file into **Cassandra**. The import process can be launched by:

```
java -cp mini-project-03-1.0.0-jar-with-dependencies.jar log.ImportData
```

The above program is executed to setup **CQL** for **project_03** keyspace synchronously.

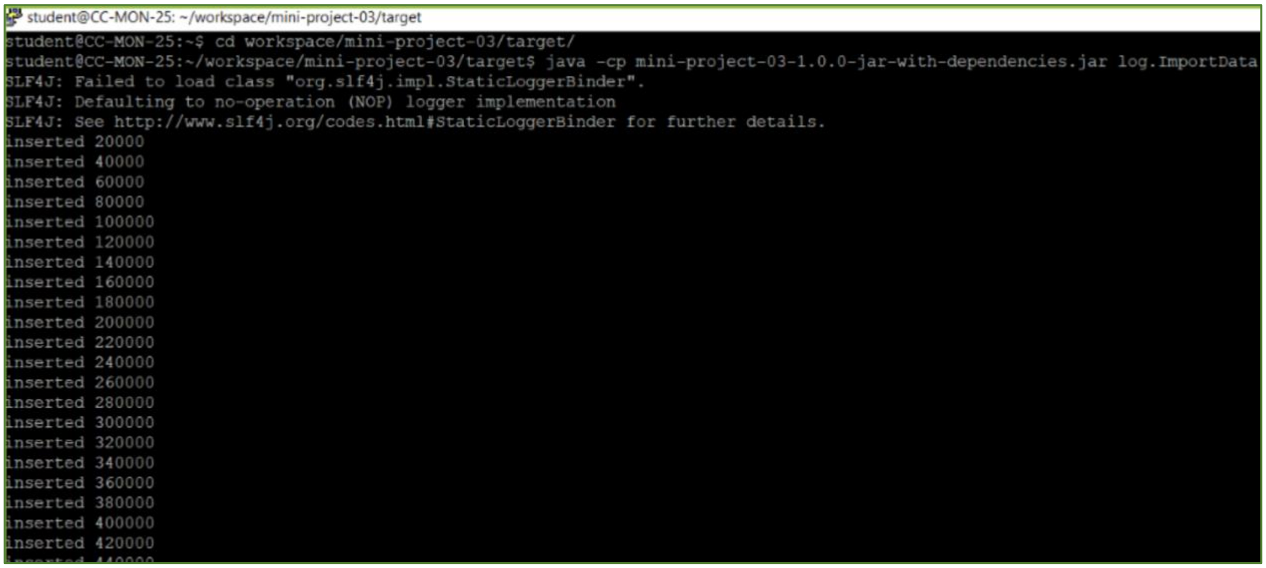
Various steps included in the process:

1. Drop existing table.
2. Creating Keyspace.
3. Creating **log** table.
4. Creating **index** on columns '**ip**' and '**path**'.
5. Create counter table for '**ip**' and '**path**'.
6. Create **UDF** for creating max group count on '**ip**' and '**path**' columns directly from log table.

After the setup is done, we read the log file row by row and insert it into the database. Here, for each row of the file, we insert the raw data pre-processed by regular expression into the 'log' table.

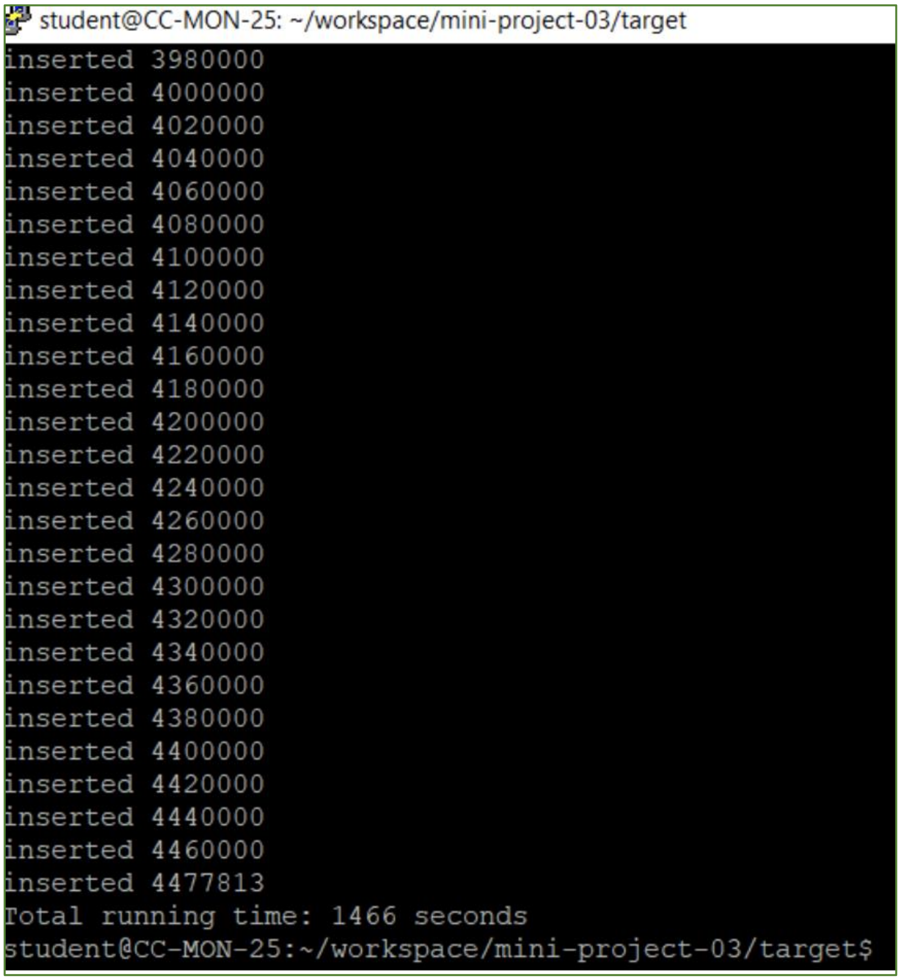
For the accessing IP address and the resource path being accessed, we use the **UPDATE CQL** to increment the count column by 1 in the corresponding tables, namely **ip** and **path**. This is feasible by using the **UPDATE CQL** alone since the **count** columns in the tables have the data type of counter. Upon invocation of the **UPDATE CQL** on a non-existing key, **Cassandra** automatically creates a row with the key and set the counter to **0**. By setting `count = count + 1`, the value of the count column will be 1 for the key's appearance.

We used asynchronous execution on inserts. The **JAVA** class Semaphore was used to limit the number of existing asynchronous requests. No more than 256 requests can exist in the request pool, matching the configuration of the **Cassandra** database.



student@CC-MON-25: ~/workspace/mini-project-03/target

```
student@CC-MON-25:~$ cd workspace/mini-project-03/target/
student@CC-MON-25:~/workspace/mini-project-03/target$ java -cp mini-project-03-1.0.0-jar-with-dependencies.jar log.ImportData
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
inserted 20000
inserted 40000
inserted 60000
inserted 80000
inserted 100000
inserted 120000
inserted 140000
inserted 160000
inserted 180000
inserted 200000
inserted 220000
inserted 240000
inserted 260000
inserted 280000
inserted 300000
inserted 320000
inserted 340000
inserted 360000
inserted 380000
inserted 400000
inserted 420000
```

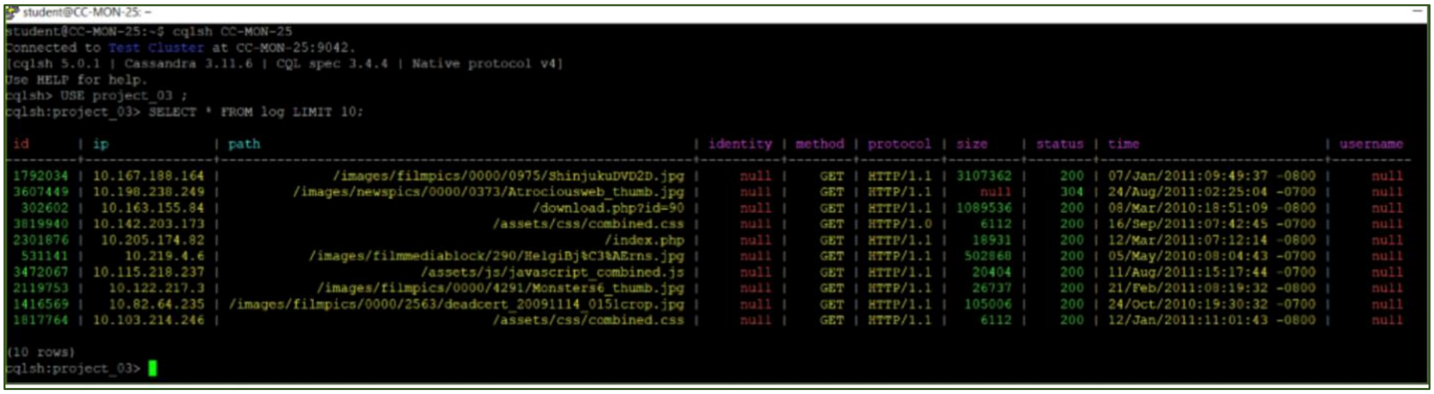


student@CC-MON-25: ~/workspace/mini-project-03/target

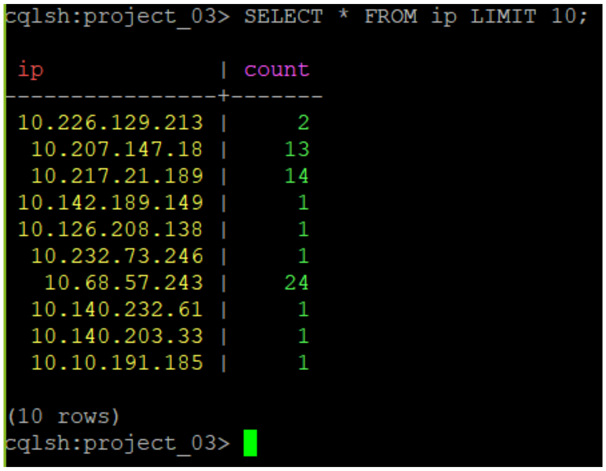
```
inserted 3980000
inserted 4000000
inserted 4020000
inserted 4040000
inserted 4060000
inserted 4080000
inserted 4100000
inserted 4120000
inserted 4140000
inserted 4160000
inserted 4180000
inserted 4200000
inserted 4220000
inserted 4240000
inserted 4260000
inserted 4280000
inserted 4300000
inserted 4320000
inserted 4340000
inserted 4360000
inserted 4380000
inserted 4400000
inserted 4420000
inserted 4440000
inserted 4460000
inserted 4477813
Total running time: 1466 seconds
student@CC-MON-25:~/workspace/mini-project-03/target$
```

The whole operation was performed in **1466** seconds, as shown in the figure above.
We can view the resulting tables in **CQLSH**.

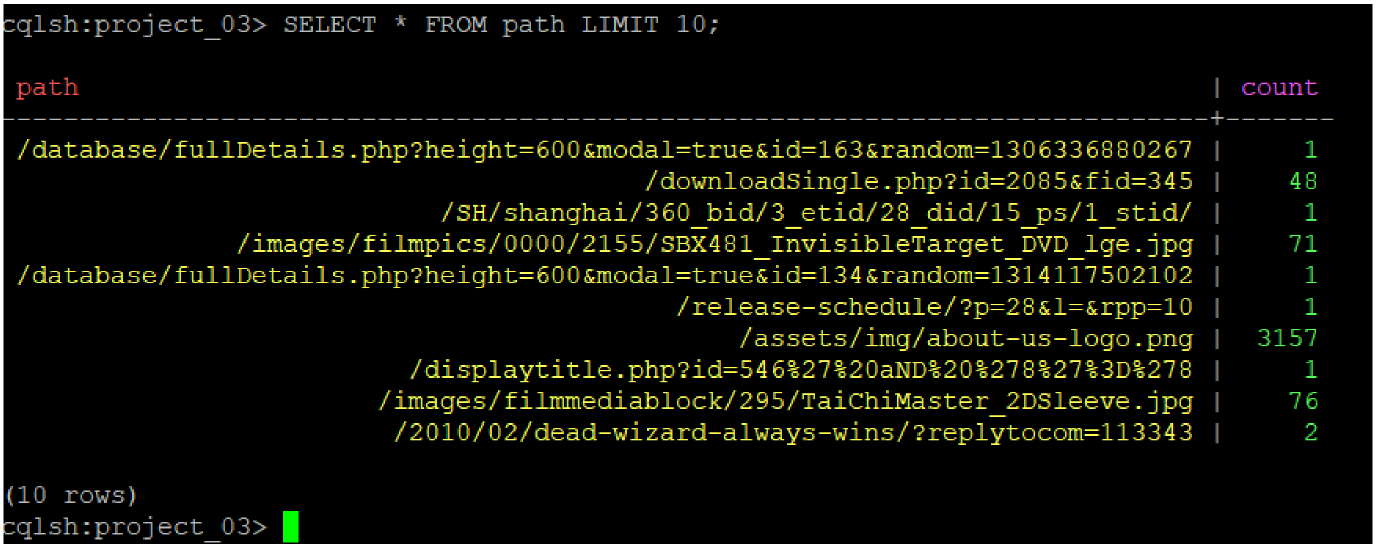
The screenshots are displayed below:



```
student@CC-MON-25: ~
student@CC-MON-25:~$ cqlsh CC-MON-25
Connected to Test Cluster at CC-MON-25:9042.
[cqlsh 5.0.1 | Cassandra 3.11.6 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> USE project_03 ;
cqlsh:project_03> SELECT * FROM log LIMIT 10;

 id      | ip             | path                                                      | identity | method | protocol | size    | status | time                       | username
---------+----------------+-----------------------------------------------------------+----------+--------+----------+---------+--------+----------------------------+----------
 1792034 | 10.167.188.164 |              /images/filmpics/0000/0975/ShinjukuDVD2D.jpg |     null |    GET | HTTP/1.1 | 3107362 |    200 | 07/Jan/2011:09:49:37 -0800 |     null
 3607449 | 10.198.238.249 |         /images/newspics/0000/0373/Atrociousweb_thumb.jpg |     null |    GET | HTTP/1.1 |    null |    304 | 24/Aug/2011:02:25:04 -0700 |     null
  302602 |  10.163.155.84 |                                       /download.php?id=90 |     null |    GET | HTTP/1.1 | 1089536 |    200 | 08/Mar/2010:18:51:09 -0800 |     null
 3819940 | 10.142.203.173 |                                  /assets/css/combined.css |     null |    GET | HTTP/1.0 |    6112 |    200 | 16/Sep/2011:07:42:45 -0700 |     null
 2301876 |  10.205.174.82 |                                                /index.php |     null |    GET | HTTP/1.1 |   18931 |    200 | 12/Mar/2011:07:12:14 -0800 |     null
  531141 |     10.219.4.6 |           /images/filmmediablock/290/HelgiBj%C3%AErns.jpg |     null |    GET | HTTP/1.1 |  502868 |    200 | 05/May/2010:08:04:43 -0700 |     null
 3472067 | 10.115.218.237 |                         /assets/js/javascript_combined.js |     null |    GET | HTTP/1.1 |   20404 |    200 | 11/Aug/2011:15:17:44 -0700 |     null
 2119753 |   10.122.217.3 |            /images/filmpics/0000/4291/Monsters6_thumb.jpg |     null |    GET | HTTP/1.1 |   26737 |    200 | 21/Feb/2011:08:19:32 -0800 |     null
 1416569 |   10.82.64.235 | /images/filmpics/0000/2563/deadcert_20091114_0151crop.jpg |     null |    GET | HTTP/1.1 |  105006 |    200 | 24/Oct/2010:19:30:32 -0700 |     null
 1817764 | 10.103.214.246 |                                  /assets/css/combined.css |     null |    GET | HTTP/1.1 |    6112 |    200 | 12/Jan/2011:11:01:43 -0800 |     null

(10 rows)
cqlsh:project_03>
```



```
cqlsh:project_03> SELECT * FROM ip LIMIT 10;

 ip              | count
-----------------+-------
 10.226.129.213 |     2
  10.207.147.18 |    13
  10.217.21.189 |    14
 10.142.189.149 |     1
 10.126.208.138 |     1
  10.232.73.246 |     1
   10.68.57.243 |    24
  10.140.232.61 |     1
  10.140.203.33 |     1
  10.10.191.185 |     1

(10 rows)
cqlsh:project_03>
```



```
cqlsh:project_03> SELECT * FROM path LIMIT 10;

 path                                                                   | count
------------------------------------------------------------------------+-------
 /database/fullDetails.php?height=600&modal=true&id=163&random=1306336880267 |     1
                                   /downloadSingle.php?id=2085&fid=345 |    48
                              /SH/shanghai/360_bid/3_etid/28_did/15_ps/1_stid/ |     1
                         /images/filmpics/0000/2155/SBX481_InvisibleTarget_DVD_lge.jpg |    71
 /database/fullDetails.php?height=600&modal=true&id=134&random=1314117502102 |     1
                                       /release-schedule/?p=28&l=&rpp=10 |     1
                                           /assets/img/about-us-logo.png |  3157
                           /displaytitle.php?id=546%27%20aND%20%278%27%3D%278 |     1
                          /images/filmmediablock/295/TaiChiMaster_2DSleeve.jpg |    76
                           /2010/02/dead-wizard-always-wins/?replytocom=113343 |     2

(10 rows)
cqlsh:project_03>
```

Part 3: Operate Data in Cassandra: (25 points)

As a part of the project, we will be working with the '**log data set**' which has been sorted in Cassandra. We need to use **CQL** (**Cassandra Query Language**) or **JAVA** driver of **Cassandra** to operate the access logs in **Cassandra**.

We need to get the results for the questions below:

Problems:

1. How many hits were made to the website item "**/assets/img/release-schedulelogo.png**"?
2. How many hits were made from the IP: **10.207.188.188**?
3. Which path in the website has been hit most? How many hits were made to the path?
4. Which IP accesses the website most? How many accesses were made by it?

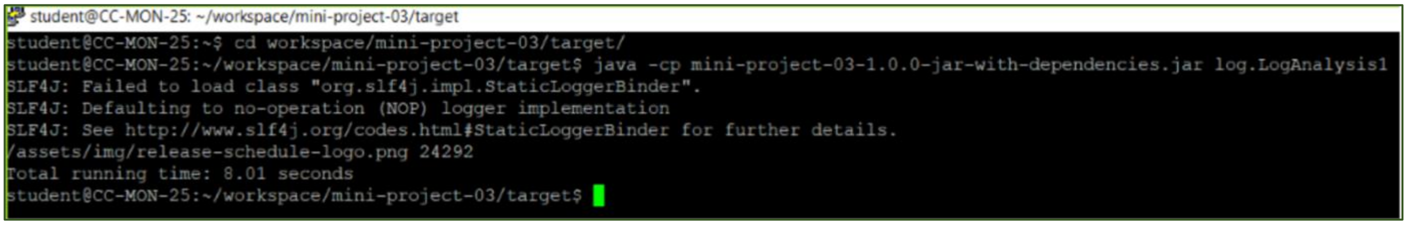
The first 2 questions can be answered by **SELECT** in **CQL**.
The last 2 questions need one extra step: We can either use **JAVA** driver to insert the counts of the items into a new table and use another **CQL** to get the answer or just use one user-defined function to get the answer of the **group-max query**.

<u>Solution:</u>

<u>Problem 1:</u>

The **LogAnalysis1.java** file is the source code for **Problem 1**.
The program is launched by the following command anywhere, either on the cluster or a local machine.

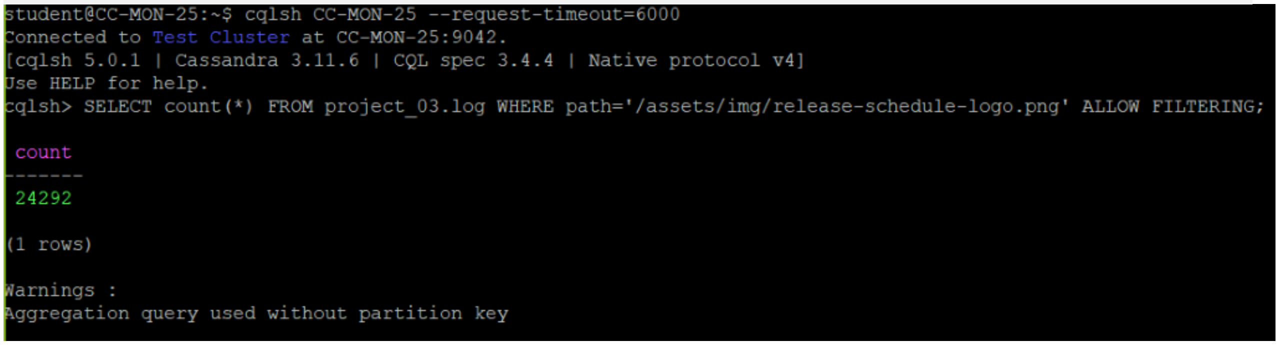
```
java -cp mini-project-03-1.0.0-jar-with-dependencies.jar log.LogAnalysis1
```



```
student@CC-MON-25: ~/workspace/mini-project-03/target
student@CC-MON-25:~$ cd workspace/mini-project-03/target/
student@CC-MON-25:~/workspace/mini-project-03/target$ java -cp mini-project-03-1.0.0-jar-with-dependencies.jar log.LogAnalysis1
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
/assets/img/release-schedule-logo.png 24292
Total running time: 8.01 seconds
student@CC-MON-25:~/workspace/mini-project-03/target$
```

The above problem can also be solved using plain **CQL**.
The **CQL** query is shown below:

```
SELECT COUNT(*)
FROM project_03.log
WHERE path='/assets/img/release-schedule-logo.png'
ALLOW FILTERING;
```



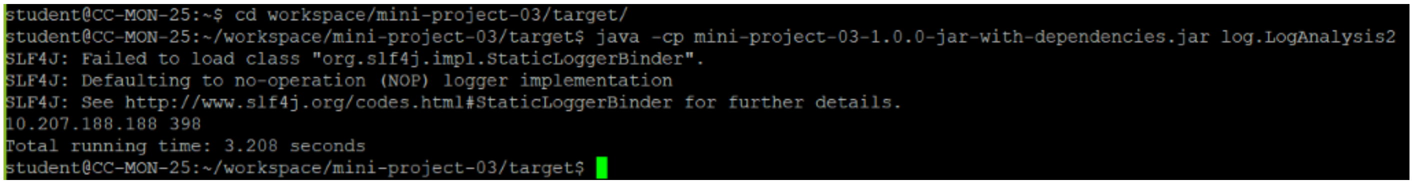
```
student@CC-MON-25:~$ cqlsh CC-MON-25 --request-timeout=6000
Connected to Test Cluster at CC-MON-25:9042.
[cqlsh 5.0.1 | Cassandra 3.11.6 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> SELECT count(*) FROM project_03.log WHERE path='/assets/img/release-schedule-logo.png' ALLOW FILTERING;

 count
-------
 24292

(1 rows)

Warnings :
Aggregation query used without partition key
```

Hence, as per the screenshots above, the path '**/assets/img/release-schedule-logo.png**' was accessed **24292** times.

Problem 2

The **LogAnalysis2.java** file is the source code for **Problem 2**.
The program can be launched by the following command anywhere, either on the cluster or a local machine.

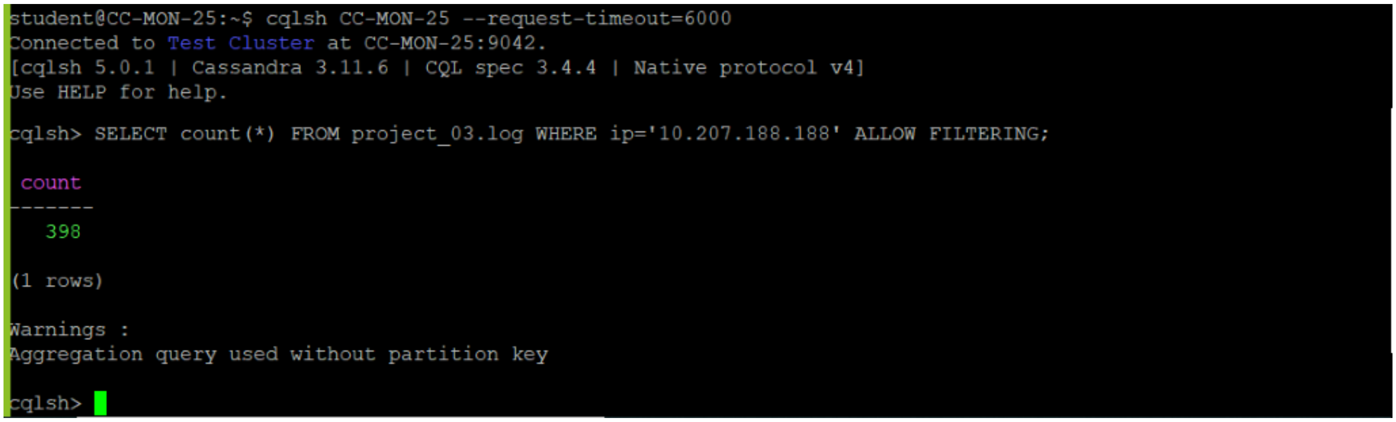
```
java -cp mini-project-03-1.0.0-jar-with-dependencies.jar log.LogAnalysis2
```



```
student@CC-MON-25:~$ cd workspace/mini-project-03/target/
student@CC-MON-25:~/workspace/mini-project-03/target$ java -cp mini-project-03-1.0.0-jar-with-dependencies.jar log.LogAnalysis2
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
10.207.188.188 398
Total running time: 3.208 seconds
student@CC-MON-25:~/workspace/mini-project-03/target$
```

The above problem can also be solved using plain **CQL**.
The **CQL** query is shown below:

```
SELECT COUNT(*)
FROM project_03.log
WHERE ip = '10.207.188.188'
ALLOW FILTERING;
```



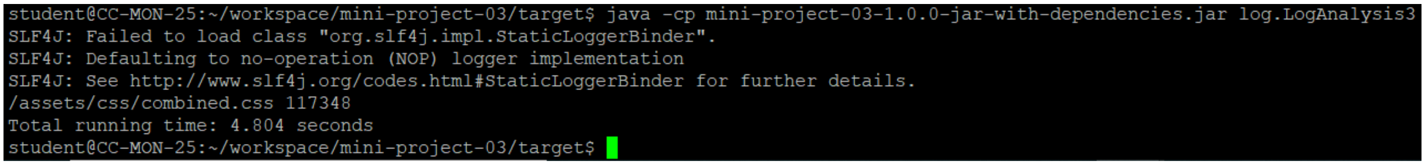
```
student@CC-MON-25:~$ cqlsh CC-MON-25 --request-timeout=6000
Connected to Test Cluster at CC-MON-25:9042.
[cqlsh 5.0.1 | Cassandra 3.11.6 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> SELECT count(*) FROM project_03.log WHERE ip='10.207.188.188' ALLOW FILTERING;

 count
-------
   398

(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh>
```

Hence, according to the screenshots, the IP Address **10.207.188.188** accessed the website **398** times.

Problem 3

The **LogAnalysis3.java** file is the source code for **Problem 3**.
The program can be launched by the following command anywhere, either on the cluster or a local machine.

```
java -cp mini-project-03-1.0.0-jar-with-dependencies.jar log.LogAnalysis3
```



```
student@CC-MON-25:~/workspace/mini-project-03/target$ java -cp mini-project-03-1.0.0-jar-with-dependencies.jar log.LogAnalysis3
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
/assets/css/combined.css 117348
Total running time: 4.804 seconds
student@CC-MON-25:~/workspace/mini-project-03/target$
```

The **JAVA** program retrieves the whole path table and find the max count row internally.
We can also get the same result from running the two following **CQL** in **CQLSH**.

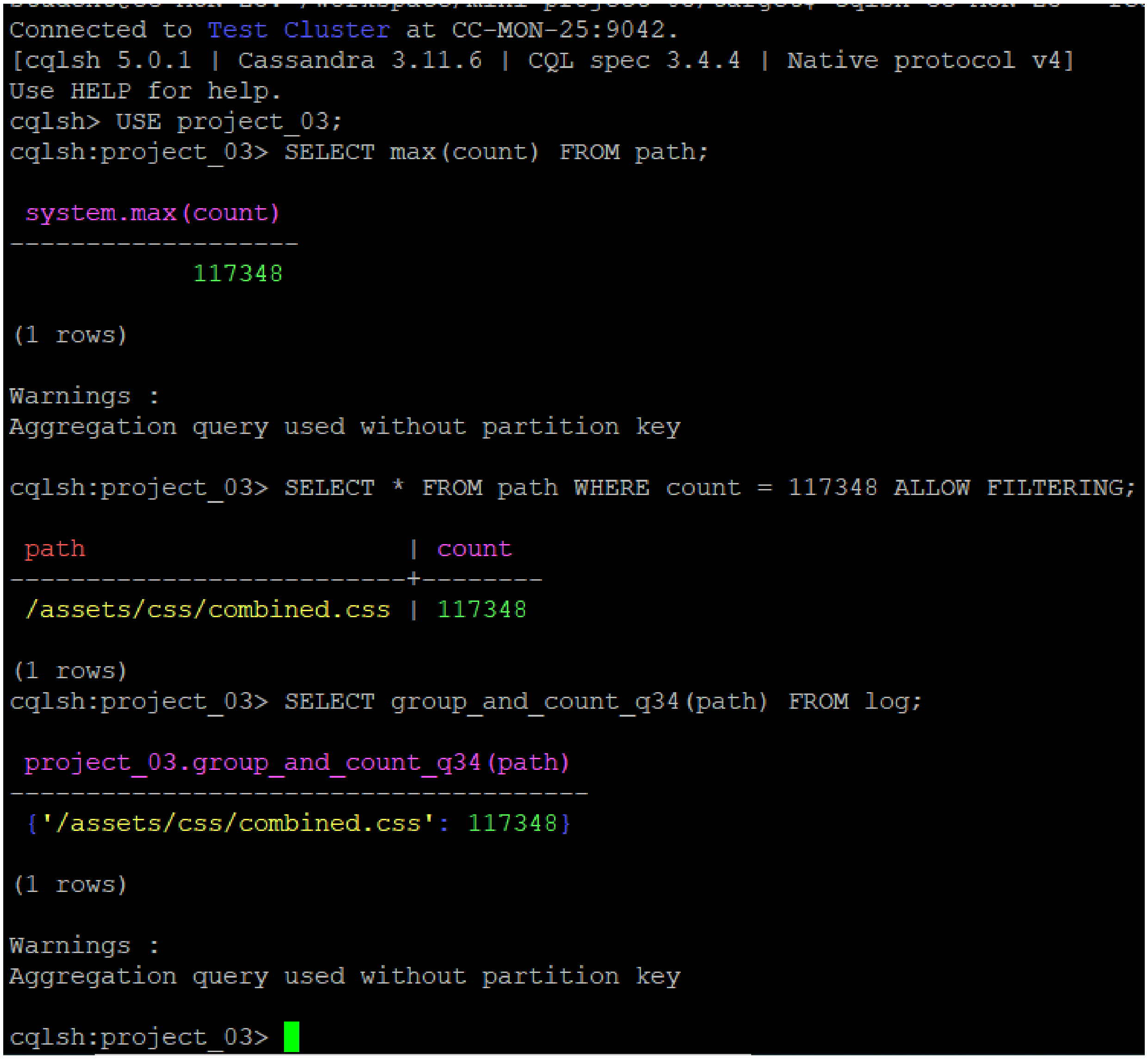
Cassandra doesn't support subqueries in the latest version. So, we had to do it with two separate queries instead of a single nested query.

```
SELECT MAX(count)
FROM path;

SELECT *
FROM path
WHERE count = 117348
ALLOW FILTERING;
```

It’s also possible to get the result from the log table with the **UDF** (**User-Defined Function:** ***group_and_count_q34()***) we defined earlier.

```
SELECT group_and_count_q34(path)
FROM log;
```

The following screenshot is shown below as evidence:



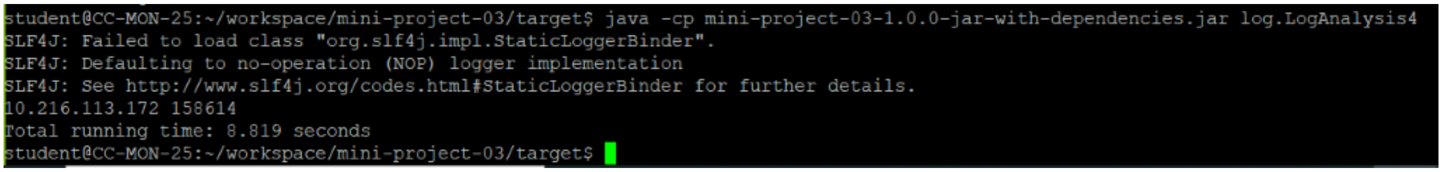
```
Connected to Test Cluster at CC-MON-25:9042.
[cqlsh 5.0.1 | Cassandra 3.11.6 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> USE project_03;
cqlsh:project_03> SELECT max(count) FROM path;

 system.max(count)
-------------------
            117348

(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh:project_03> SELECT * FROM path WHERE count = 117348 ALLOW FILTERING;

 path                       | count
----------------------------+--------
 /assets/css/combined.css | 117348

(1 rows)
cqlsh:project_03> SELECT group_and_count_q34(path) FROM log;

 project_03.group_and_count_q34(path)
--------------------------------------
 {'/assets/css/combined.css': 117348}

(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh:project_03>
```

As per the screenshots, ‘**/assets/css/combined.css**’ was the most accessed website, with **117348** hits.

Problem 4

The **LogAnalysis4.java** file is the source code for **Problem 4**.
The program can be launched by the following command anywhere, either on the cluster or a local machine.

```
java -cp mini-project-03-1.0.0-jar-with-dependencies.jar log.LogAnalysis4
```



```
student@CC-MON-25:~/workspace/mini-project-03/target$ java -cp mini-project-03-1.0.0-jar-with-dependencies.jar log.LogAnalysis4
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
10.216.113.172 158614
Total running time: 8.819 seconds
student@CC-MON-25:~/workspace/mini-project-03/target$
```

The **JAVA** program actually retrieves the whole **ip table** and find the max count row internally.
We can also get the same result from running the two following **CQL** in **CQLSH**.

Cassandra doesn't support subqueries in the latest version. So, we had to do it with two separate queries instead of a single nested query.

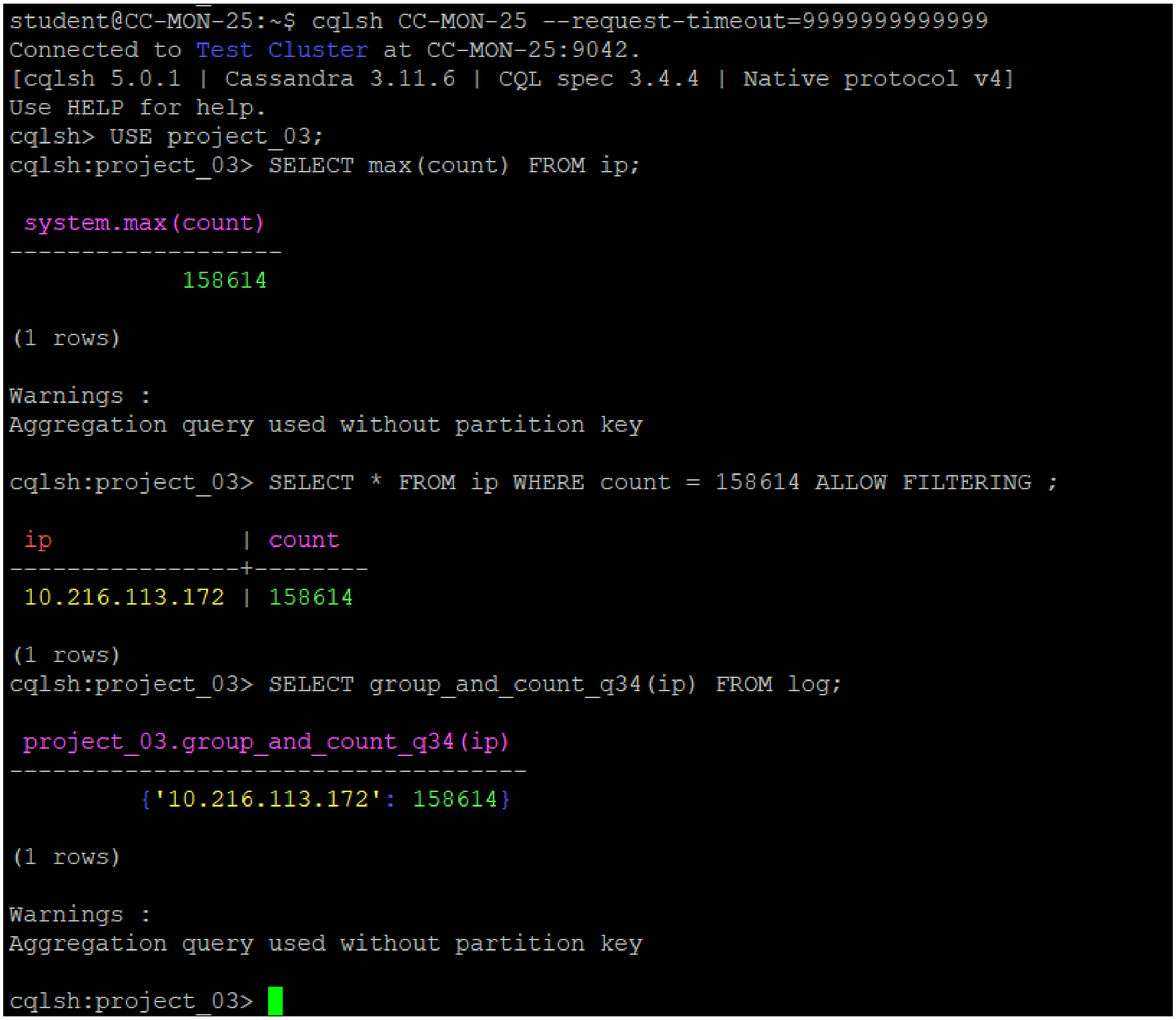
```
SELECT MAX(count)
FROM ip;

SELECT *
FROM ip
WHERE count = 158614
ALLOW FILTERING;
```

It's also possible to get the result from the log table with the **UDF** (**User-Defined Function:** ***group_and_count_q34()***) we defined earlier.

```
SELECT group_and_count_q34(ip)
FROM log;
```

Due to the fact the system has to query and aggregate over the big table of about 4.5 - 5 million rows, this query took a few minutes to run.

The following screenshot is shown below as evidence:



```
student@CC-MON-25:~$ cqlsh CC-MON-25 --request-timeout=9999999999999
Connected to Test Cluster at CC-MON-25:9042.
[cqlsh 5.0.1 | Cassandra 3.11.6 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> USE project_03;
cqlsh:project_03> SELECT max(count) FROM ip;

 system.max(count)
-------------------
            158614

(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh:project_03> SELECT * FROM ip WHERE count = 158614 ALLOW FILTERING ;

 ip             | count
----------------+--------
 10.216.113.172 | 158614

(1 rows)
cqlsh:project_03> SELECT group_and_count_q34(ip) FROM log;

 project_03.group_and_count_q34(ip)
------------------------------------
         {'10.216.113.172': 158614}

(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh:project_03> 
```

As per the screenshots, the I.P. **10.216.113.172** accessed the website the most, which is **158614** times.