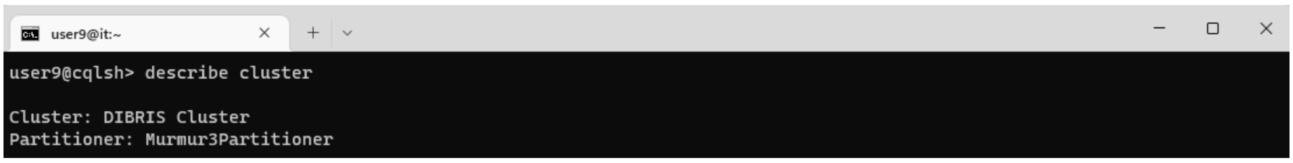
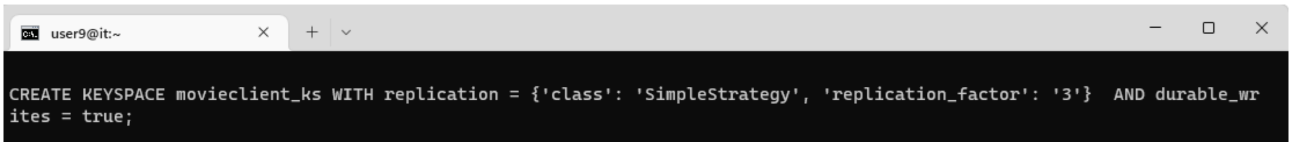
ASSIGNMENT III: CASSANDRA

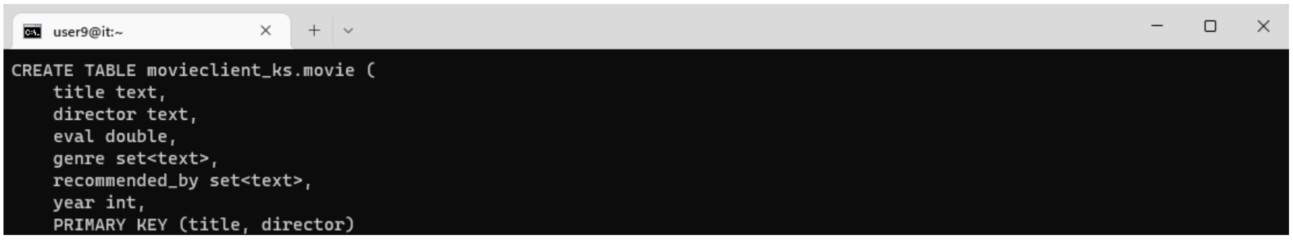
EXERCISE 1
Using the command *describe cluster*, the name of cluster and of the partitioner are:

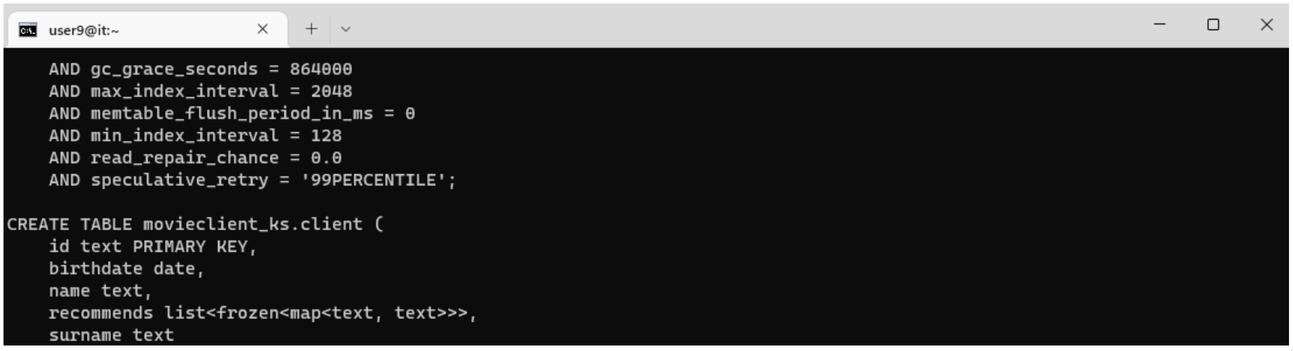

```
user9@cqlsh> describe cluster

Cluster: DIBRIS Cluster
Partitioner: Murmur3Partitioner
```

Using the command *describe movieclient_ks*, we can see that the replication factor is equal to 3.

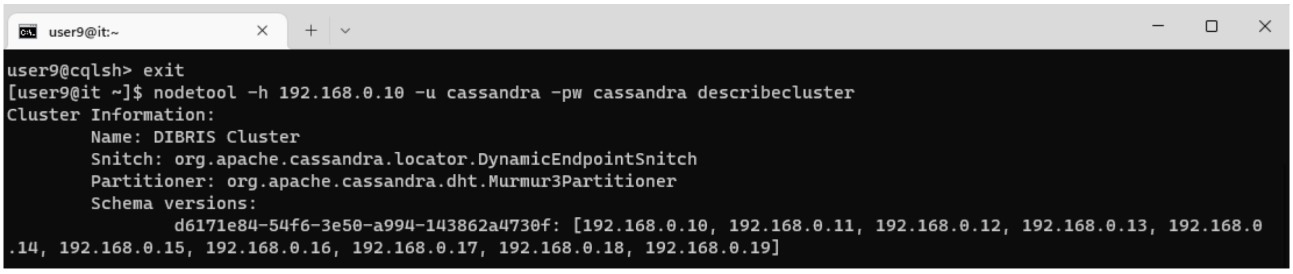

```
CREATE KEYSPACE movieclient_ks WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '3'}  AND durable_wr
ites = true;
```

Using the previous command we can also see that there are two tables named *movie* and *client*. The schemas of the tables are:

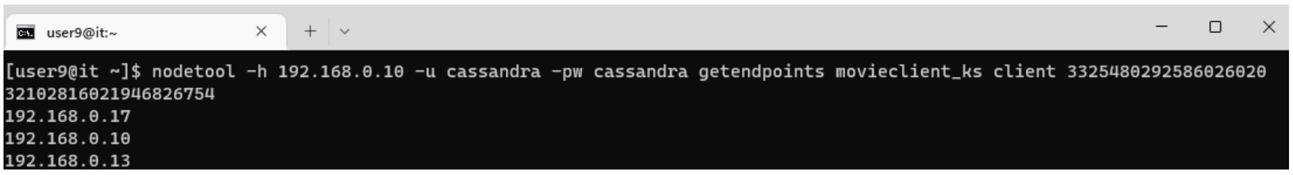

```
CREATE TABLE movieclient_ks.movie (
    title text,
    director text,
    eval double,
    genre set<text>,
    recommended_by set<text>,
    year int,
    PRIMARY KEY (title, director)
```


```
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

CREATE TABLE movieclient_ks.client (
    id text PRIMARY KEY,
    birthdate date,
    name text,
    recommends list<frozen<map<text, text>>>,
    surname text
```

EXERCISE 2
- Data are uniformly distributed accross the cluster through *MurmurHash* hash values. *Murmur3Partitioner* is the default partitioner.

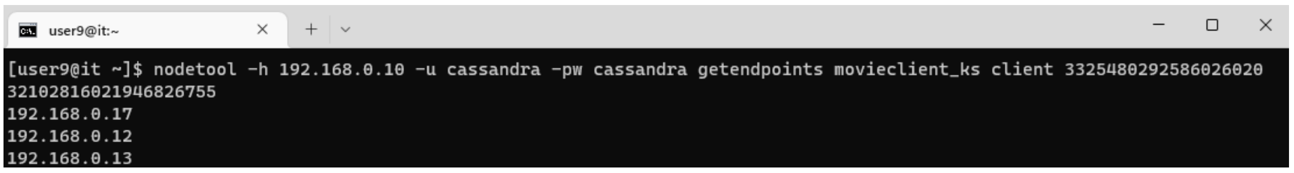

```
user9@cqlsh> exit
[user9@it ~]$ nodetool -h 192.168.0.10 -u cassandra -pw cassandra describecluster
Cluster Information:
        Name: DIBRIS Cluster
        Snitch: org.apache.cassandra.locator.DynamicEndpointSnitch
        Partitioner: org.apache.cassandra.dht.Murmur3Partitioner
        Schema versions:
                d6171e84-54f6-3e50-a994-143862a4730f: [192.168.0.10, 192.168.0.11, 192.168.0.12, 192.168.0.13, 192.168.0
.14, 192.168.0.15, 192.168.0.16, 192.168.0.17, 192.168.0.18, 192.168.0.19]
```

- Data are replicated in three machines with the following IP addresses: 192.168.0.17, 192.168.0.10, 192.168.0.13. Three replicas because the replication factor is equal to 3.



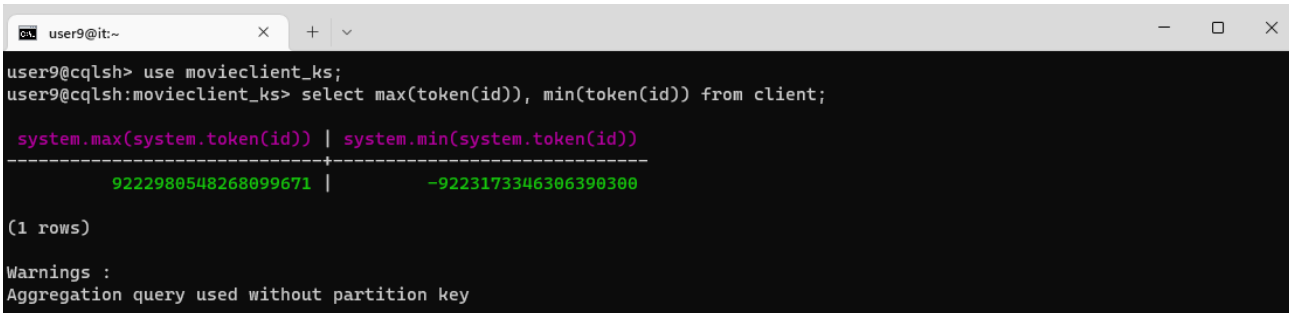
```
[user9@it ~]$ nodetool -h 192.168.0.10 -u cassandra -pw cassandra getendpoints movieclient_ks client 3325480292586026020
32102816021946826754
192.168.0.17
192.168.0.10
192.168.0.13
```

- Data are replicated in three machines with the following IP addressess: 192.168.0.17, 192.168.0.12, 192,168.0.13. We can say that the machines with IP addresses 192.168.0.17, 192.168.0.13 have rows with different partition keys. Replication factor is equal to 3, so we have three replicas.



```
[user9@it ~]$ nodetool -h 192.168.0.10 -u cassandra -pw cassandra getendpoints movieclient_ks client 3325480292586026020
32102816021946826755
192.168.0.17
192.168.0.12
192.168.0.13
```

- The range of hash values associated to the partition keys is:



```
user9@cqlsh> use movieclient_ks;
user9@cqlsh:movieclient_ks> select max(token(id)), min(token(id)) from client;

 system.max(system.token(id)) | system.min(system.token(id))
------------------------------+------------------------------
          9222980548268099671 |         -9223173346306390300

(1 rows)

Warnings :
Aggregation query used without partition key
```

EXERCISE 3

- **Q1:** *SELECT * FROM movie WHERE title='Dracula';*
- **Q2:** *SELECT * FROM movie WHERE title IN ('Dracula', 'Gang');*
- **Q3 (invalid query unless you use ALLOW FILTERING):** *SELECT title FROM movie WHERE director='Robert Altman';*
- **Q4:** *SELECT genre FROM movie WHERE director='Robert Altman' and title='Gang';*
- **Q5:** *SELECT year FROM movie WHERE director='Robert Altman' AND title IN ('Gang', 'Aria');*
- **Q6 (invalid query unless you use ALLOW FILTERING):** *SELECT title FROM movie WHERE director='Robert Altman' and year>1990;*
- **Q7 (invalid query unless you use ALLOW FILTERING):** *SELECT title, genre FROM movie WHERE director='Ken Loach' and genre CONTAINS 'Comedy';*
- **Q8 (invalid query: ORDER BY is only supported when the partition key is restricted by an EQ or an IN):** *SELECT title, year FROM movie WHERE director='Ken Loach' ORDER BY title;*
- **Q9 (invalid query: GROUP BY currently only support groups of columns following their declared order in the PRIMARY KEY):** *SELECT director, COUNT(*) FROM movie GROUP BY director;*
- **Q10 (invalid query unless you use ALLOW FILTERING):** *SELECT id FROM client WHERE birthdate>'2000-01-01';*
- **Q11a (invalid query unless you use ALLOW FILTERING):** *SELECT COUNT(*) FROM movie WHERE title='Spirits' AND director='Todd Sheets' AND recommended_by CONTAINS '332548029258602602032102816021946826754';*

- **Q11b (invalid query unless you use ALLOW FILTERING):** *SELECT COUNT(*) FROM client WHERE id='332548029258602602032102816021946826754' AND recommends CONTAINS {'title':'Spirits', 'director':'Todd Sheets'};*
- **Q12a (invalid query unless you use ALLOW FILTERING):** *SELECT COUNT(*) FROM movie WHERE director='Todd Sheets' AND recommended_by CONTAINS '332548029258602602032102816021946826754';*
- **Q12b (Invalid query: index on 'director' or a condition over partition key 'title' is needed):** *SELECT COUNT(*) FROM client WHERE id='332548029258602602032102816021946826754' AND recommends CONTAINS {director='Todd Sheets'};*
  The difference among Q11 and Q12 lies in the absence of a condition over 'title', this still makes possible to execute the query over the movie table but not over the client table (unless an index on 'director' is created)

- **Q13 (invalid query unless you use ALLOW FILTERING):** *SELECT id, name, surname, birthdate FROM client WHERE recommends CONTAINS {'title':'Gang', 'director':'Robert Altman'};*