Repositórios de Dados e NoSQL Programa da Disciplina eEDB-016: Hands-on Cassandra

Case: Sensor Data Modeling

Grupo 1:
Marcelo Barbugli
Gisele Siqueira
Diego Moura
Roberto Eyama
Ricardo Geroto
Matheus Higa

Atividade feita utilizando as plataformas DataStax e AWS Cloud9:
Evidências de:

- Criar as tabelas
- Popular dados
- Realizar consultas

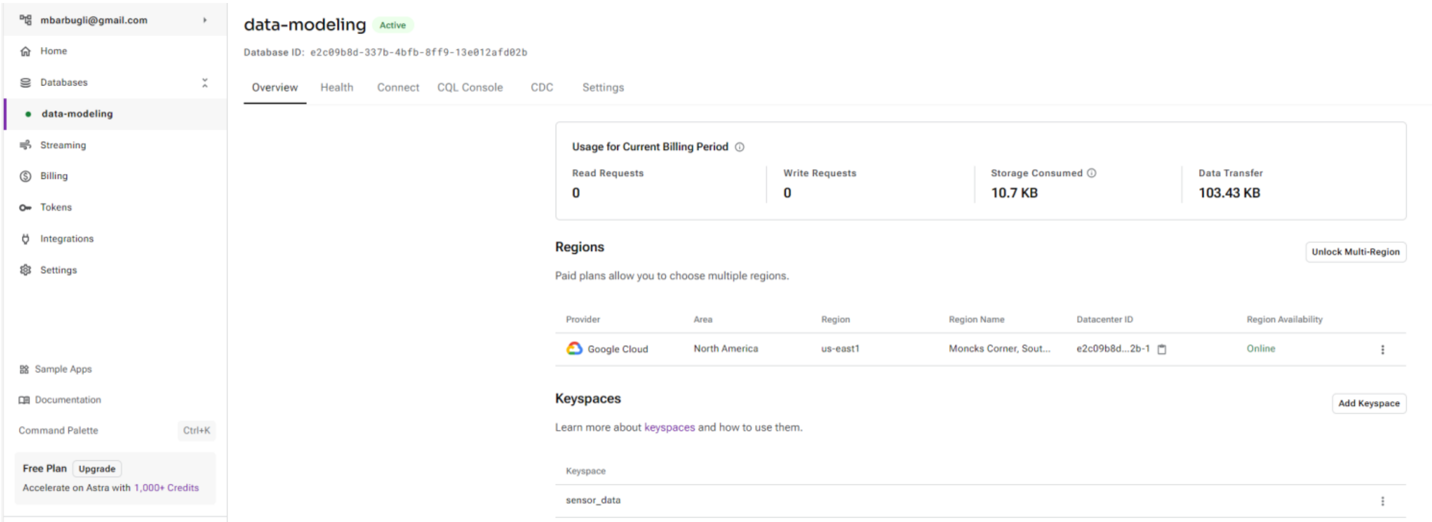
(extra) Step1: Criando um banco de dados no Astra (datastax)

```
get CLoud provider
[INFO]  Database 'data-modeling' and keyspace 'sensor_data' are being created.
[INFO]  Database 'data-modeling' has status 'PENDING' waiting to be 'ACTIVE' ...
[INFO]  Database 'data-modeling' has status 'ACTIVE' (took 111861 millis)
[OK]    Database 'data-modeling' is ready.
gitpod /workspace/data-modeling-sensor-data (main) $ astra db list
+---------------------+--------------------------------------+-----------+-------+---+-----------+
| Name                | id                                   | Regions   | Cloud | V | Status    |
+---------------------+--------------------------------------+-----------+-------+---+-----------+
| --wait              | f5efef7c-30bb-494d-95ba-f6573f558ca3 | us-east1  | gcp   |   | ACTIVE    |
| data-modeling       | e2c09b8d-337b-4bfb-8ff9-13e012afd02b | us-east1  | gcp   |   | ACTIVE    |
+---------------------+--------------------------------------+-----------+-------+---+-----------+

gitpod /workspace/data-modeling-sensor-data (main) $ astra db get data-modeling
+------------------+-----------------------------------------+
| Attribute        | Value                                   |
+------------------+-----------------------------------------+
| Name             | data-modeling                           |
| id               | e2c09b8d-337b-4bfb-8ff9-13e012afd02b    |
| Cloud            | GCP                                     |
| Regions          | us-east1                                |
| Status           | ACTIVE                                  |
| Vector           | Disabled                                |
| Default Keyspace | sensor_data                             |
| Creation Time    | 2024-06-06T00:38:31Z                    |
|                  |                                         |
| Keyspaces        | [0] sensor_data                         |
|                  |                                         |
|                  |                                         |
| Regions          | [0] us-east1                            |
|                  |                                         |
+------------------+-----------------------------------------+

gitpod /workspace/data-modeling-sensor-data (main) $ astra db cqlsh data-modeling -k sensor_data
[INFO]  Cqlsh is starting, please wait for connection establishment...
Connected to cndb at 127.0.0.1:9042.
[cqlsh 6.8.0 | Cassandra 4.0.0.6816 | CQL spec 3.4.5 | Native protocol v4]
Use HELP for help.
token@cqlsh:sensor_data> 7W
```

Astra Dashboard – Serverless Database Service (SaaS)

Db: <u>data-modeling</u>

Keyspace: <u>sensor_data</u>



Step1: Criando banco de dados Cassandra via CQL shell, criando a KEYSPACE 'sensor_data' e passando a chamada para iniciar a criação das tabelas dentro deste *keyspace*:

DataStax:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS 4   EXPOSED PORTS   AZURE   COMMENTS

./cassandragitpod /workspace/data-modeling-sensor-data (main) $ ./cassandra
Starting a Cassandra cluster ... DONE!
Cassandra successfully started.
gitpod /workspace/data-modeling-sensor-data (main) $ cqlsh
WARNING: cqlsh was built against 4.1.4, but this server is 4.0.13.  All features may not work!
Connected to Cassandra Cluster at 127.0.0.1:9042
[cqlsh 6.1.0 | Cassandra 4.0.13 | CQL spec 3.4.5 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE sensor_data
 'class': 'NetworkTopologyStrategy',
  'DC-Housto   ... WITH replication = {
   ...   'class': 'NetworkTopologyStrategy',
   ...   'DC-Houston': 1 };
cqlsh> USE sensor_data;
cqlsh:sensor_data>
```

AWS:

```
cqlsh> DROP KEYSPACE IF EXISTS killrvideo;


cqlsh> // CREATE KEYSPACE killrvideo WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 3 };
cqlsh> CREATE KEYSPACE sensor_data WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };
cqlsh>
cqlsh>
cqlsh> USE sensor_data;
```

Step 2: Criando as tabelas networks, sensors_by_network, temperatures_by_network, e temperatures_by_sensor e executando DESCRIBE para confirmar a criação. A PRIMARY KEY é composta pela Partition Key + Clustering key. Exemplo da tabela networks é Partition Key = bucket; Clustering key = name e assim formando a PRIMARY KEY.

DataStax:

```
Use HELP for help.
cqlsh> CREATE KEYSPACE sensor_data
 'class': 'NetworkTopologyStrategy',
  'DC-Housto   ... WITH replication = {
   ...   'class': 'NetworkTopologyStrategy',
   ...   'DC-Houston': 1 };
cqlsh> USE sensor_data;
cqlsh:sensor_data> CREATE TABLE IF NOT EXISTS networks (
               ...   bucket TEXT,
               ...   name TEXT,
               ...   description TEXT,
               ...   region TEXT,
               ...   num_sensors INT,
               ...   PRIMARY KEY ((bucket),name)
               ... );
cqlsh:sensor_data> CREATE TABLE IF NOT EXISTS temperatures_by_network (
               ...   network TEXT,
               ...   week DATE,
               ...   date_hour TIMESTAMP,
               ...   sensor TEXT,
               ...   avg_temperature FLOAT,
               ...   latitude DECIMAL,
               ...   longitude DECIMAL,
               ...   PRIMARY KEY ((network,week),date_hour,sensor)
               ... ) WITH CLUSTERING ORDER BY (date_hour DESC, sensor ASC);
cqlsh:sensor_data> CREATE TABLE IF NOT EXISTS sensors_by_network (
               ...   network TEXT,
               ...   sensor TEXT,
               ...   latitude DECIMAL,
               ...   longitude DECIMAL,
               ...   characteristics MAP<TEXT,TEXT>,
               ...   PRIMARY KEY ((network),sensor)
               ... );
cqlsh:sensor_data> CREATE TABLE IF NOT EXISTS temperatures_by_sensor (
               ...   sensor TEXT,
               ...   date DATE,
               ...   timestamp TIMESTAMP,
               ...   value FLOAT,
               ...   PRIMARY KEY ((sensor,date),timestamp)
               ... ) WITH CLUSTERING ORDER BY (timestamp DESC);
cqlsh:sensor_data> DESCRIBE TABLES;

networks  sensors_by_network  temperatures_by_network  temperatures_by_sensor
```

AWS:

```
cqlsh:sensor_data> CREATE TABLE IF NOT EXISTS networks (
               ...   bucket TEXT,
               ...   name TEXT,
               ...   description TEXT,
               ...   region TEXT,
               ...   num_sensors INT,
               ...   PRIMARY KEY ((bucket),name)
               ... );
cqlsh:sensor_data> CREATE TABLE IF NOT EXISTS temperatures_by_network (
               ...   network TEXT,
               ...   week DATE,
               ...   date_hour TIMESTAMP,
               ...   sensor TEXT,
               ...   avg_temperature FLOAT,
               ...   latitude DECIMAL,
               ...   longitude DECIMAL,
               ...   PRIMARY KEY ((network,week),date_hour,sensor)
               ... ) WITH CLUSTERING ORDER BY (date_hour DESC, sensor ASC);
cqlsh:sensor_data> CREATE TABLE IF NOT EXISTS sensors_by_network (
               ...   network TEXT,
               ...   sensor TEXT,
               ...   latitude DECIMAL,
               ...   longitude DECIMAL,
               ...   characteristics MAP<TEXT,TEXT>,
               ...   PRIMARY KEY ((network),sensor)
               ... );
cqlsh:sensor_data> CREATE TABLE IF NOT EXISTS temperatures_by_sensor (
               ...   sensor TEXT,
               ...   date DATE,
               ...   timestamp TIMESTAMP,
               ...   value FLOAT,
               ...   PRIMARY KEY ((sensor,date),timestamp)
               ... ) WITH CLUSTERING ORDER BY (timestamp DESC);CREATE TABLE IF NOT EXISTS temperatures_by_sensor (
               ...   sensor TEXT,
               ...   date DATE,
               ...   timestamp TIMESTAMP,
               ...   value FLOAT,
               ...   PRIMARY KEY ((sensor,date),timestamp)
               ... ) WITH CLUSTERING ORDER BY (timestamp DESC);CREATE TABLE IF NOT EXISTS temperatures_by_sensor (
               ...   sensor TEXT,
               ...   date DATE,
               ...   timestamp TIMESTAMP,
               ...   value FLOAT,
               ...   PRIMARY KEY ((sensor,date),timestamp)
               ... ) WITH CLUSTERING ORDER BY (timestamp DESC);
cqlsh:sensor_data>
cqlsh:sensor_data>  DESCRIBE TABLES;

networks  sensors_by_network  temperatures_by_network  temperatures_by_sensor
```

STEP3: Execução de script para realizar INSERT nas 4 tabelas.

DataStax:

```
cqlsh:sensor_data> source '/workspace/data-modeling-sensor-data/assets/sensor_data.cql'
WARNING: cqlsh was built against 4.1.4, but this server is 4.0.13.  All features may not work!
```

AWS:

```
cqlsh:sensor_data> -- Populate table networks:
cqlsh:sensor_data> ---------------------------
cqlsh:sensor_data> INSERT INTO networks
               ... (bucket,name,description,region,num_sensors)
               ... VALUES ('all','forest-net',
               ...         'forest fire detection network',
               ...         'south',3);
cqlsh:sensor_data> INSERT INTO networks
               ... (bucket,name,description,region,num_sensors)
               ... VALUES ('all','volcano-net',
               ...         'volcano monitoring network',
               ...         'north',2);
INSERT INTO tcqlsh:sensor_data>
cqlsh:sensor_data>
cqlsh:sensor_data> -- Populate table sensors_by_network:
cqlsh:sensor_data> -------------------------------------
cqlsh:sensor_data> INSERT INTO sensors_by_network
itude)
VALUES ('forest-ne               ... (network,sensor,latitude,longitude,characteristics)
               ... VALUES ('forest-net','s1001',30.526503,-95.582815,
               ...        {'accuracy':'medium','sensitivity':'high'});
cqlsh:sensor_data> INSERT INTO sensors_by_network
               ... (network,sensor,latitude,longitude,characteristics)
               ... VALUES ('forest-net','s1002',30.518650,-95.583585,
               ...        {'accuracy':'medium','sensitivity':'high'});
cqlsh:sensor_data> INSERT INTO sensors_by_network
tude)
VAL               ... (network,sensor,latitude,longitude,characteristics)
               ... VALUES ('forest-net','s1003',30.515056,-95.556225,
               ...        {'accuracy':'medium','sensitivity':'high'});

---------------------------cqlsh:sensor_data> INSERT INTO sensors_by_network
               ... (network,sensor,latitude,longitude,characteristics)
               ... VALUES ('volcano-net','s2001',44.460321,-110.828151,
               ...        {'accuracy':'high','sensitivity':'medium'});
cqlsh:sensor_data> INSERT INTO sensors_by_network
```

Realizando SELECT * na tabela networks: Partition Key = bucket; Clustering key = name; outros atributos = description, num_sensors, region.

DataStax:

```
cqlsh:sensor_data> SELECT * FROM networks;

 bucket | name        | description                   | num_sensors | region
--------+-------------+-------------------------------+-------------+--------
    all |  forest-net | forest fire detection network |           3 |  south
    all | volcano-net |    volcano monitoring network |           2 |  north
```

AWS:

```
cqlsh:sensor_data> SELECT * FROM networks;

 bucket | name        | description                | num_sensors | region
--------+-------------+----------------------------+-------------+--------
    all |  forest-net | forest fire detection network |          3 |  south
    all | volcano-net |    volcano monitoring network |          2 |  north

(2 rows)
```

Realizando SELECT, passando duas colunas que são as Partition Key (network e week), duas colunas que são as Clustering Key (date_hour e sensor), onde juntas formam a PRIMARY KEY. Outro atributo é a coluna avg_temperature.

DataStax:

```
cqlsh:sensor_data> SELECT network, week, date_hour,
               ...        sensor, avg_temperature
               ... FROM temperatures_by_network;

 network    | week       | date_hour                       | sensor | avg_temperature
------------+------------+---------------------------------+--------+-----------------
 forest-net | 2020-06-28 | 2020-07-04 12:00:00.000000+0000 |  s1001 |            97.5
 forest-net | 2020-06-28 | 2020-07-04 12:00:00.000000+0000 |  s1002 |             100
 forest-net | 2020-06-28 | 2020-07-04 12:00:00.000000+0000 |  s1003 |            98.5
 forest-net | 2020-06-28 | 2020-07-04 00:00:00.000000+0000 |  s1001 |            79.5
 forest-net | 2020-06-28 | 2020-07-04 00:00:00.000000+0000 |  s1002 |              81
 forest-net | 2020-06-28 | 2020-07-04 00:00:00.000000+0000 |  s1003 |            80.5
 forest-net | 2020-07-05 | 2020-07-06 12:00:00.000000+0000 |  s1001 |           106.5
 forest-net | 2020-07-05 | 2020-07-06 12:00:00.000000+0000 |  s1002 |             109
 forest-net | 2020-07-05 | 2020-07-06 12:00:00.000000+0000 |  s1003 |            1372
 forest-net | 2020-07-05 | 2020-07-06 00:00:00.000000+0000 |  s1001 |            90.5
 forest-net | 2020-07-05 | 2020-07-06 00:00:00.000000+0000 |  s1002 |              90
 forest-net | 2020-07-05 | 2020-07-06 00:00:00.000000+0000 |  s1003 |            90.5
 forest-net | 2020-07-05 | 2020-07-05 12:00:00.000000+0000 |  s1001 |            98.5
 forest-net | 2020-07-05 | 2020-07-05 12:00:00.000000+0000 |  s1002 |            99.5
 forest-net | 2020-07-05 | 2020-07-05 12:00:00.000000+0000 |  s1003 |           101.5
 forest-net | 2020-07-05 | 2020-07-05 00:00:00.000000+0000 |  s1001 |            80.5
 forest-net | 2020-07-05 | 2020-07-05 00:00:00.000000+0000 |  s1002 |              82
 forest-net | 2020-07-05 | 2020-07-05 00:00:00.000000+0000 |  s1003 |            82.5

(18 rows)
```

AWS:

```
cqlsh:sensor_data> SELECT network, week, date_hour,
               ...        sensor, avg_temperature
               ... FROM temperatures_by_network;

 network     | week       | date_hour                       | sensor | avg_temperature
-------------+------------+---------------------------------+--------+-----------------
 forest-net | 2020-06-28 | 2020-07-04 12:00:00.000000+0000 |  s1001 |            97.5
 forest-net | 2020-06-28 | 2020-07-04 12:00:00.000000+0000 |  s1002 |             100
 forest-net | 2020-06-28 | 2020-07-04 12:00:00.000000+0000 |  s1003 |            98.5
 forest-net | 2020-06-28 | 2020-07-04 00:00:00.000000+0000 |  s1001 |            79.5
 forest-net | 2020-06-28 | 2020-07-04 00:00:00.000000+0000 |  s1002 |              81
 forest-net | 2020-06-28 | 2020-07-04 00:00:00.000000+0000 |  s1003 |            80.5
 forest-net | 2020-07-05 | 2020-07-06 12:00:00.000000+0000 |  s1001 |           106.5
 forest-net | 2020-07-05 | 2020-07-06 12:00:00.000000+0000 |  s1002 |             109
 forest-net | 2020-07-05 | 2020-07-06 12:00:00.000000+0000 |  s1003 |            1372
 forest-net | 2020-07-05 | 2020-07-06 00:00:00.000000+0000 |  s1001 |            90.5
 forest-net | 2020-07-05 | 2020-07-06 00:00:00.000000+0000 |  s1002 |              90
 forest-net | 2020-07-05 | 2020-07-06 00:00:00.000000+0000 |  s1003 |            90.5
 forest-net | 2020-07-05 | 2020-07-05 12:00:00.000000+0000 |  s1001 |            98.5
 forest-net | 2020-07-05 | 2020-07-05 12:00:00.000000+0000 |  s1002 |            99.5
 forest-net | 2020-07-05 | 2020-07-05 12:00:00.000000+0000 |  s1003 |           101.5
 forest-net | 2020-07-05 | 2020-07-05 00:00:00.000000+0000 |  s1001 |            80.5
 forest-net | 2020-07-05 | 2020-07-05 00:00:00.000000+0000 |  s1002 |              82
 forest-net | 2020-07-05 | 2020-07-05 00:00:00.000000+0000 |  s1003 |            82.5

(18 rows)
```

Realizando SELECT * da tabela sensors_by_network. Partition Key = network; Clustering key = sensor; outros atributos = characteristics, latitude, longitude.

DataStax:

```
cqlsh:sensor_data> SELECT * FROM sensors_by_network;

 network     | sensor | characteristics                              | latitude  | longitude
-------------+--------+----------------------------------------------+-----------+------------
  forest-net |  s1001 | {'accuracy': 'medium', 'sensitivity': 'high'} | 30.526503 |  -95.582815
  forest-net |  s1002 | {'accuracy': 'medium', 'sensitivity': 'high'} | 30.518650 |  -95.583585
  forest-net |  s1003 | {'accuracy': 'medium', 'sensitivity': 'high'} | 30.515056 |  -95.556225
 volcano-net |  s2001 | {'accuracy': 'high', 'sensitivity': 'medium'} | 44.460321 | -110.828151
 volcano-net |  s2002 | {'accuracy': 'high', 'sensitivity': 'medium'} | 44.463195 | -110.830124

(5 rows)
```

AWS:

```
cqlsh:sensor_data> SELECT * FROM sensors_by_network;

 network     | sensor | characteristics                               | latitude  | longitude
-------------+--------+-----------------------------------------------+-----------+------------
  forest-net |  s1001 | {'accuracy': 'medium', 'sensitivity': 'high'} | 30.526503 |  -95.582815
  forest-net |  s1002 | {'accuracy': 'medium', 'sensitivity': 'high'} | 30.518650 |  -95.583585
  forest-net |  s1003 | {'accuracy': 'medium', 'sensitivity': 'high'} | 30.515056 |  -95.556225
 volcano-net |  s2001 | {'accuracy': 'high', 'sensitivity': 'medium'} | 44.460321 | -110.828151
 volcano-net |  s2002 | {'accuracy': 'high', 'sensitivity': 'medium'} | 44.463195 | -110.830124

(5 rows)
```

Realizando SELECT * da tabela temperature_by_sensor. Partition Key = sensor e date; Clustering key = timestamp; outros atributos = value.

DataStax:

```
cqlsh:sensor_data> SELECT * FROM temperatures_by_sensor;

 sensor | date       | timestamp                       | value
--------+------------+---------------------------------+-------
  s1001 | 2020-07-04 | 2020-07-04 12:59:59.000000+0000 |    98
  s1001 | 2020-07-04 | 2020-07-04 12:00:01.000000+0000 |    97
  s1001 | 2020-07-04 | 2020-07-04 00:59:59.000000+0000 |    79
  s1001 | 2020-07-04 | 2020-07-04 00:00:01.000000+0000 |    80
  s1001 | 2020-07-05 | 2020-07-05 12:59:59.000000+0000 |    99
  s1001 | 2020-07-05 | 2020-07-05 12:00:01.000000+0000 |    98
  s1001 | 2020-07-05 | 2020-07-05 00:59:59.000000+0000 |    80
  s1001 | 2020-07-05 | 2020-07-05 00:00:01.000000+0000 |    81
  s1002 | 2020-07-06 | 2020-07-06 12:59:59.000000+0000 |   110
  s1002 | 2020-07-06 | 2020-07-06 12:00:01.000000+0000 |   108
  s1002 | 2020-07-06 | 2020-07-06 00:59:59.000000+0000 |    90
  s1002 | 2020-07-06 | 2020-07-06 00:00:01.000000+0000 |    90
  s1003 | 2020-07-04 | 2020-07-04 12:59:59.000000+0000 |    98
  s1003 | 2020-07-04 | 2020-07-04 12:00:01.000000+0000 |    99
  s1003 | 2020-07-04 | 2020-07-04 00:59:59.000000+0000 |    80
  s1003 | 2020-07-04 | 2020-07-04 00:00:01.000000+0000 |    81
  s1003 | 2020-07-06 | 2020-07-06 12:59:59.000000+0000 |  1429
  s1003 | 2020-07-06 | 2020-07-06 12:00:01.000000+0000 |  1315
  s1003 | 2020-07-06 | 2020-07-06 00:59:59.000000+0000 |    90
  s1003 | 2020-07-06 | 2020-07-06 00:00:01.000000+0000 |    90
  s1003 | 2020-07-05 | 2020-07-05 12:59:59.000000+0000 |   102
  s1003 | 2020-07-05 | 2020-07-05 12:00:01.000000+0000 |   101
  s1003 | 2020-07-05 | 2020-07-05 00:59:59.000000+0000 |    82
```

AWS:

```
cqlsh:sensor_data> SELECT * FROM temperatures_by_sensor;

 sensor | date       | timestamp                       | value
--------+------------+---------------------------------+-------
  s1001 | 2020-07-04 | 2020-07-04 12:59:59.000000+0000 |    98
  s1001 | 2020-07-04 | 2020-07-04 12:00:01.000000+0000 |    97
  s1001 | 2020-07-04 | 2020-07-04 00:59:59.000000+0000 |    79
  s1001 | 2020-07-04 | 2020-07-04 00:00:01.000000+0000 |    80
  s1001 | 2020-07-05 | 2020-07-05 12:59:59.000000+0000 |    99
  s1001 | 2020-07-05 | 2020-07-05 12:00:01.000000+0000 |    98
  s1001 | 2020-07-05 | 2020-07-05 00:59:59.000000+0000 |    80
  s1001 | 2020-07-05 | 2020-07-05 00:00:01.000000+0000 |    81
  s1002 | 2020-07-06 | 2020-07-06 12:59:59.000000+0000 |   110
  s1002 | 2020-07-06 | 2020-07-06 12:00:01.000000+0000 |   108
  s1002 | 2020-07-06 | 2020-07-06 00:59:59.000000+0000 |    90
  s1002 | 2020-07-06 | 2020-07-06 00:00:01.000000+0000 |    90
  s1003 | 2020-07-04 | 2020-07-04 12:59:59.000000+0000 |    98
  s1003 | 2020-07-04 | 2020-07-04 12:00:01.000000+0000 |    99
  s1003 | 2020-07-04 | 2020-07-04 00:59:59.000000+0000 |    80
  s1003 | 2020-07-04 | 2020-07-04 00:00:01.000000+0000 |    81
  s1003 | 2020-07-06 | 2020-07-06 12:59:59.000000+0000 |  1429
  s1003 | 2020-07-06 | 2020-07-06 12:00:01.000000+0000 |  1315
  s1003 | 2020-07-06 | 2020-07-06 00:59:59.000000+0000 |    90
  s1003 | 2020-07-06 | 2020-07-06 00:00:01.000000+0000 |    90
  s1003 | 2020-07-05 | 2020-07-05 12:59:59.000000+0000 |   102
  s1003 | 2020-07-05 | 2020-07-05 12:00:01.000000+0000 |   101
  s1003 | 2020-07-05 | 2020-07-05 00:59:59.000000+0000 |    82
  s1003 | 2020-07-05 | 2020-07-05 00:00:01.000000+0000 |    83
  s1002 | 2020-07-05 | 2020-07-05 12:59:59.000000+0000 |    99
  s1002 | 2020-07-05 | 2020-07-05 12:00:01.000000+0000 |   100
  s1002 | 2020-07-05 | 2020-07-05 00:59:59.000000+0000 |    82
  s1002 | 2020-07-05 | 2020-07-05 00:00:01.000000+0000 |    82
  s1002 | 2020-07-04 | 2020-07-04 12:59:59.000000+0000 |   100
  s1002 | 2020-07-04 | 2020-07-04 12:00:01.000000+0000 |   100
  s1002 | 2020-07-04 | 2020-07-04 00:59:59.000000+0000 |    80
  s1002 | 2020-07-04 | 2020-07-04 00:00:01.000000+0000 |    82
  s1001 | 2020-07-06 | 2020-07-06 12:59:59.000000+0000 |   107
  s1001 | 2020-07-06 | 2020-07-06 12:00:01.000000+0000 |   106
  s1001 | 2020-07-06 | 2020-07-06 00:59:59.000000+0000 |    90
  s1001 | 2020-07-06 | 2020-07-06 00:00:01.000000+0000 |    90

(36 rows)
```

STEP4: Buscando todas as informações da tabela networks, ordenando pela Clustering key = name.
Realizando SELECT nas colunas name, description, region, num_sensors da tabela networks:
A importância do WHERE é porque bucket representa a Partition Key da tabela networks, sendo necessário passar na query a condição de igualdade (ou desigualdade) com valor atribuído, ordenando e garantindo unicidade quando a tabela foi criada.

DataStax:

```
cqlsh:sensor_data> SELECT name, description,
               ...        region, num_sensors
               ... FROM networks
               ... WHERE bucket = 'all';

 name        | description                   | region | num_sensors
-------------+-------------------------------+--------+-------------
  forest-net | forest fire detection network |  south |           3
 volcano-net |    volcano monitoring network |  north |           2
```

AWS:

```
cqlsh:sensor_data> SELECT name, description,
               ...        region, num_sensors
               ... FROM networks
               ... WHERE bucket = 'all';

 name        | description                   | region | num_sensors
-------------+-------------------------------+--------+-------------
  forest-net | forest fire detection network |  south |           3
 volcano-net |    volcano monitoring network |  north |           2

(2 rows)
```

STEP5: Encontre as temperaturas médias horárias para cada <u>sensor</u> na <u>network</u> = **forest-net** e intervalo de datas [2020-07-05, 2020-07-06] dentro da semana de 2020-07-05; order by <u>date</u> (desc) e <u>hour</u> (desc).

DataStax:

```
cqlsh:sensor_data> SELECT date_hour, avg_temperature,
               ...        latitude, longitude, sensor
               ... FROM temperatures_by_network
               ... WHERE network    = 'forest-net'
               ...   AND week       = '2020-07-05'
               ...   AND date_hour >= '2020-07-05'
               ...   AND date_hour  < '2020-07-07';

 date_hour                       | avg_temperature | latitude  | longitude  | sensor
---------------------------------+-----------------+-----------+------------+--------
 2020-07-06 12:00:00.000000+0000 |           106.5 | 30.526503 | -95.582815 |  s1001
 2020-07-06 12:00:00.000000+0000 |             109 | 30.518650 | -95.583585 |  s1002
 2020-07-06 12:00:00.000000+0000 |            1372 | 30.515056 | -95.556225 |  s1003
 2020-07-06 00:00:00.000000+0000 |            90.5 | 30.526503 | -95.582815 |  s1001
 2020-07-06 00:00:00.000000+0000 |              90 | 30.518650 | -95.583585 |  s1002
 2020-07-06 00:00:00.000000+0000 |            90.5 | 30.515056 | -95.556225 |  s1003
 2020-07-05 12:00:00.000000+0000 |            98.5 | 30.526503 | -95.582815 |  s1001
 2020-07-05 12:00:00.000000+0000 |            99.5 | 30.518650 | -95.583585 |  s1002
 2020-07-05 12:00:00.000000+0000 |           101.5 | 30.515056 | -95.556225 |  s1003
 2020-07-05 00:00:00.000000+0000 |            80.5 | 30.526503 | -95.582815 |  s1001
 2020-07-05 00:00:00.000000+0000 |              82 | 30.518650 | -95.583585 |  s1002
 2020-07-05 00:00:00.000000+0000 |            82.5 | 30.515056 | -95.556225 |  s1003

(12 rows)
```

AWS:

```
cqlsh:sensor_data> SELECT date_hour, avg_temperature,
               ...        latitude, longitude, sensor
               ... FROM temperatures_by_network
               ... WHERE network    = 'forest-net'
               ...   AND week       = '2020-07-05'
               ...   AND date_hour >= '2020-07-05'
               ...   AND date_hour  < '2020-07-07';

 date_hour                       | avg_temperature | latitude  | longitude  | sensor
---------------------------------+-----------------+-----------+------------+--------
 2020-07-06 12:00:00.000000+0000 |           106.5 | 30.526503 | -95.582815 |  s1001
 2020-07-06 12:00:00.000000+0000 |             109 | 30.518650 | -95.583585 |  s1002
 2020-07-06 12:00:00.000000+0000 |            1372 | 30.515056 | -95.556225 |  s1003
 2020-07-06 00:00:00.000000+0000 |            90.5 | 30.526503 | -95.582815 |  s1001
 2020-07-06 00:00:00.000000+0000 |              90 | 30.518650 | -95.583585 |  s1002
 2020-07-06 00:00:00.000000+0000 |            90.5 | 30.515056 | -95.556225 |  s1003
 2020-07-05 12:00:00.000000+0000 |            98.5 | 30.526503 | -95.582815 |  s1001
 2020-07-05 12:00:00.000000+0000 |            99.5 | 30.518650 | -95.583585 |  s1002
 2020-07-05 12:00:00.000000+0000 |           101.5 | 30.515056 | -95.556225 |  s1003
 2020-07-05 00:00:00.000000+0000 |            80.5 | 30.526503 | -95.582815 |  s1001
 2020-07-05 00:00:00.000000+0000 |              82 | 30.518650 | -95.583585 |  s1002
 2020-07-05 00:00:00.000000+0000 |            82.5 | 30.515056 | -95.556225 |  s1003

(12 rows)
```

Encontre as temperaturas médias horárias para cada sensor na network = **forest-net** e intervalo de datas [2020-07-04,2020-07-06] entre as semanas de 2020-06-28 e 2020-07-05; order by date (desc) e hour (desc).

Solution1:

DataStax:

```
cqlsh:sensor_data> SELECT date_hour, avg_temperature,
               ...        latitude, longitude, sensor
               ... FROM temperatures_by_network
               ... WHERE network    = 'forest-net'
               ...   AND week       = '2020-06-28'
               ...   AND date_hour >= '2020-07-04'
               ...   AND date_hour  < '2020-07-07';

 date_hour                       | avg_temperature | latitude  | longitude  | sensor
---------------------------------+-----------------+-----------+------------+--------
 2020-07-04 12:00:00.000000+0000 |            97.5 | 30.526503 | -95.582815 |  s1001
 2020-07-04 12:00:00.000000+0000 |             100 | 30.518650 | -95.583585 |  s1002
 2020-07-04 12:00:00.000000+0000 |            98.5 | 30.515056 | -95.556225 |  s1003
 2020-07-04 00:00:00.000000+0000 |            79.5 | 30.526503 | -95.582815 |  s1001
 2020-07-04 00:00:00.000000+0000 |              81 | 30.518650 | -95.583585 |  s1002
 2020-07-04 00:00:00.000000+0000 |            80.5 | 30.515056 | -95.556225 |  s1003

(6 rows)
```

AWS:

```
cqlsh:sensor_data>
cqlsh:sensor_data> SELECT date_hour, avg_temperature,
               ...        latitude, longitude, sensor
               ... FROM temperatures_by_network
               ... WHERE network     = 'forest-net'
               ...   AND week        = '2020-06-28'
               ...   AND date_hour >= '2020-07-04'
               ...   AND date_hour  < '2020-07-07';

 date_hour                       | avg_temperature | latitude  | longitude  | sensor
---------------------------------+-----------------+-----------+------------+--------
 2020-07-04 12:00:00.000000+0000 |            97.5 | 30.526503 | -95.582815 |  s1001
 2020-07-04 12:00:00.000000+0000 |             100 | 30.518650 | -95.583585 |  s1002
 2020-07-04 12:00:00.000000+0000 |            98.5 | 30.515056 | -95.556225 |  s1003
 2020-07-04 00:00:00.000000+0000 |            79.5 | 30.526503 | -95.582815 |  s1001
 2020-07-04 00:00:00.000000+0000 |              81 | 30.518650 | -95.583585 |  s1002
 2020-07-04 00:00:00.000000+0000 |            80.5 | 30.515056 | -95.556225 |  s1003

(6 rows)
```

Solution2:

DataStax:

```
cqlsh:sensor_data> SELECT date_hour, avg_temperature,
               ...        latitude, longitude, sensor
               ... FROM temperatures_by_network
               ... WHERE network     = 'forest-net'
               ...   AND week       IN ('2020-07-05','2020-06-28')
               ...   AND date_hour >= '2020-07-04'
               ...   AND date_hour  < '2020-07-07';

 date_hour                       | avg_temperature | latitude  | longitude  | sensor
---------------------------------+-----------------+-----------+------------+--------
 2020-07-04 12:00:00.000000+0000 |            97.5 | 30.526503 | -95.582815 |  s1001
 2020-07-04 12:00:00.000000+0000 |             100 | 30.518650 | -95.583585 |  s1002
 2020-07-04 12:00:00.000000+0000 |            98.5 | 30.515056 | -95.556225 |  s1003
 2020-07-04 00:00:00.000000+0000 |            79.5 | 30.526503 | -95.582815 |  s1001
 2020-07-04 00:00:00.000000+0000 |              81 | 30.518650 | -95.583585 |  s1002
 2020-07-04 00:00:00.000000+0000 |            80.5 | 30.515056 | -95.556225 |  s1003
 2020-07-06 12:00:00.000000+0000 |           106.5 | 30.526503 | -95.582815 |  s1001
 2020-07-06 12:00:00.000000+0000 |             109 | 30.518650 | -95.583585 |  s1002
 2020-07-06 12:00:00.000000+0000 |            1372 | 30.515056 | -95.556225 |  s1003
 2020-07-06 00:00:00.000000+0000 |            90.5 | 30.526503 | -95.582815 |  s1001
 2020-07-06 00:00:00.000000+0000 |              90 | 30.518650 | -95.583585 |  s1002
 2020-07-06 00:00:00.000000+0000 |            90.5 | 30.515056 | -95.556225 |  s1003
 2020-07-05 12:00:00.000000+0000 |            98.5 | 30.526503 | -95.582815 |  s1001
 2020-07-05 12:00:00.000000+0000 |            99.5 | 30.518650 | -95.583585 |  s1002
 2020-07-05 12:00:00.000000+0000 |           101.5 | 30.515056 | -95.556225 |  s1003
 2020-07-05 00:00:00.000000+0000 |            80.5 | 30.526503 | -95.582815 |  s1001
 2020-07-05 00:00:00.000000+0000 |              82 | 30.518650 | -95.583585 |  s1002
 2020-07-05 00:00:00.000000+0000 |            82.5 | 30.515056 | -95.556225 |  s1003

(18 rows)
```

AWS:

```
cqlsh:sensor_data> SELECT date_hour, avg_temperature,
               ...        latitude, longitude, sensor
               ... FROM temperatures_by_network
               ... WHERE network    = 'forest-net'
               ...   AND week      IN ('2020-07-05','2020-06-28')
               ...   AND date_hour >= '2020-07-04'
               ...   AND date_hour  < '2020-07-07';

 date_hour                       | avg_temperature | latitude  | longitude  | sensor
---------------------------------+-----------------+-----------+------------+--------
 2020-07-04 12:00:00.000000+0000 |            97.5 | 30.526503 | -95.582815 |  s1001
 2020-07-04 12:00:00.000000+0000 |             100 | 30.518650 | -95.583585 |  s1002
 2020-07-04 12:00:00.000000+0000 |            98.5 | 30.515056 | -95.556225 |  s1003
 2020-07-04 00:00:00.000000+0000 |            79.5 | 30.526503 | -95.582815 |  s1001
 2020-07-04 00:00:00.000000+0000 |              81 | 30.518650 | -95.583585 |  s1002
 2020-07-04 00:00:00.000000+0000 |            80.5 | 30.515056 | -95.556225 |  s1003
 2020-07-06 12:00:00.000000+0000 |           106.5 | 30.526503 | -95.582815 |  s1001
 2020-07-06 12:00:00.000000+0000 |             109 | 30.518650 | -95.583585 |  s1002
 2020-07-06 12:00:00.000000+0000 |            1372 | 30.515056 | -95.556225 |  s1003
 2020-07-06 00:00:00.000000+0000 |            90.5 | 30.526503 | -95.582815 |  s1001
 2020-07-06 00:00:00.000000+0000 |              90 | 30.518650 | -95.583585 |  s1002
 2020-07-06 00:00:00.000000+0000 |            90.5 | 30.515056 | -95.556225 |  s1003
 2020-07-05 12:00:00.000000+0000 |            98.5 | 30.526503 | -95.582815 |  s1001
 2020-07-05 12:00:00.000000+0000 |            99.5 | 30.518650 | -95.583585 |  s1002
 2020-07-05 12:00:00.000000+0000 |           101.5 | 30.515056 | -95.556225 |  s1003
 2020-07-05 00:00:00.000000+0000 |            80.5 | 30.526503 | -95.582815 |  s1001
 2020-07-05 00:00:00.000000+0000 |              82 | 30.518650 | -95.583585 |  s1002
 2020-07-05 00:00:00.000000+0000 |            82.5 | 30.515056 | -95.556225 |  s1003

(18 rows)
```

STEP6: Buscando informações de todos os sensors (Clustering Key) em network (Partition Key) onde o valor seja = **'forest-net'**.

DataStax:

```
cqlsh:sensor_data> SELECT *
               ... FROM sensors_by_network
               ... WHERE network = 'forest-net';

 network    | sensor | characteristics                                  | latitude  | longitude
------------+--------+--------------------------------------------------+-----------+------------
 forest-net |  s1001 | {'accuracy': 'medium', 'sensitivity': 'high'}    | 30.526503 | -95.582815
 forest-net |  s1002 | {'accuracy': 'medium', 'sensitivity': 'high'}    | 30.518650 | -95.583585
 forest-net |  s1003 | {'accuracy': 'medium', 'sensitivity': 'high'}    | 30.515056 | -95.556225

(3 rows)
```

AWS:

```
cqlsh:sensor_data> SELECT *
               ... FROM sensors_by_network
               ... WHERE network = 'forest-net';

 network    | sensor | characteristics                               | latitude  | longitude
------------+--------+-----------------------------------------------+-----------+-----------
 forest-net |  s1001 | {'accuracy': 'medium', 'sensitivity': 'high'} | 30.526503 | -95.582815
 forest-net |  s1002 | {'accuracy': 'medium', 'sensitivity': 'high'} | 30.518650 | -95.583585
 forest-net |  s1003 | {'accuracy': 'medium', 'sensitivity': 'high'} | 30.515056 | -95.556225

(3 rows)
```

STEP7: Encontre medições brutas para o sensor **s1003** em 2020-07-06; order by **timestamp** (desc).

DataStax:

```
cqlsh:sensor_data> SELECT timestamp, value
               ... FROM temperatures_by_sensor
               ... WHERE sensor = 's1003'
               ...   AND date   = '2020-07-06';

 timestamp                       | value
---------------------------------+-------
 2020-07-06 12:59:59.000000+0000 |  1429
 2020-07-06 12:00:01.000000+0000 |  1315
 2020-07-06 00:59:59.000000+0000 |    90
 2020-07-06 00:00:01.000000+0000 |    90

(4 rows)
```

AWS:

```
cqlsh:sensor_data> SELECT timestamp, value
               ... FROM temperatures_by_sensor
               ... WHERE sensor = 's1003'
               ...   AND date   = '2020-07-06';

 timestamp                       | value
---------------------------------+-------
 2020-07-06 12:59:59.000000+0000 |  1429
 2020-07-06 12:00:01.000000+0000 |  1315
 2020-07-06 00:59:59.000000+0000 |    90
 2020-07-06 00:00:01.000000+0000 |    90

(4 rows)
```

Consideração final entre DataStax e AWS Cloud9:

1. A diferença que encontramos foi ao criar imagem do Cassandra no Cloud9, onde é necessário. Já no dataStax precisamos apenas iniciar pois a imagem já está montada.
2. Para criar o KEYSPACE, a sintaxe entre DataStax e AWS Cloud9 sofre alteração, pois no DataStax permite a criação da classe da réplica = 'NetworkTopologyStrategy', enquanto nossa conta na AWS Academy só permite o

‘SimpleStrategy’. Outro fator é que pelo DataStax (GCP) passamos a região que nossa replicação estará alocada e pelo AWS Cloud9 podemos escolher o ‘replication_factor’ = 1 (ou 2, 3) região(ões). E.g.:

a. DataStax: CREATE KEYSPACE sensor_data WITH REPLICATION = { 'class': 'NetworkTopologyStrategy', 'DC-Houston': 1 };
b. AWS Cloud9: CREATE KEYSPACE sensor_data WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };