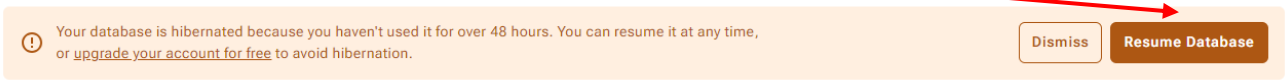
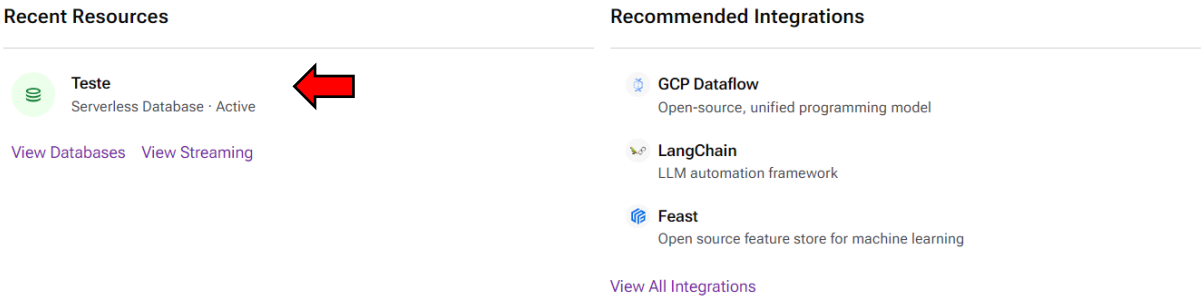
Tutorial Comandos Básicos Cassandra Usando CQL Console

1) Acesse o site da DataStax em: https://www.datastax.com/products/datastax-enterprise/dse-opscenter

2) No seu próximo acesso, provavelmente o banco estará em hibernação para poupar recursos do servidor. Clique no nome do banco e aguarde estar online novamente. Geralmente demora alguns minutos. Provavelmente a seguinte mensagem irá aparecer. **Clique** em **Resume Database**.

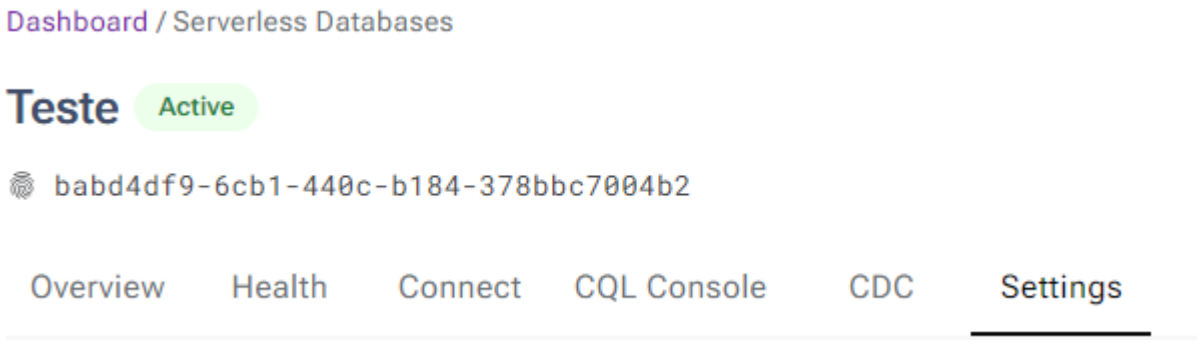


3) Na parte de baixo da tela clique no nome do banco criado anterior “Teste”:

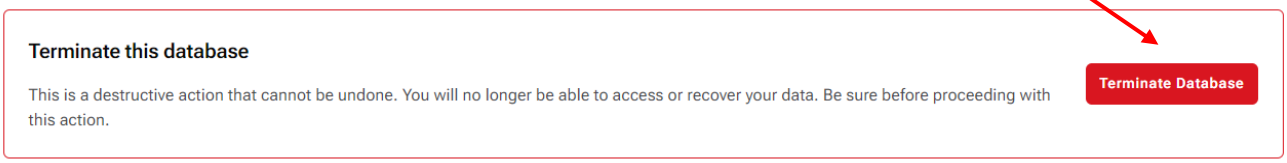


4) Excluiremos o banco criado para testes para podermos seguir com aquele desse tutorial.

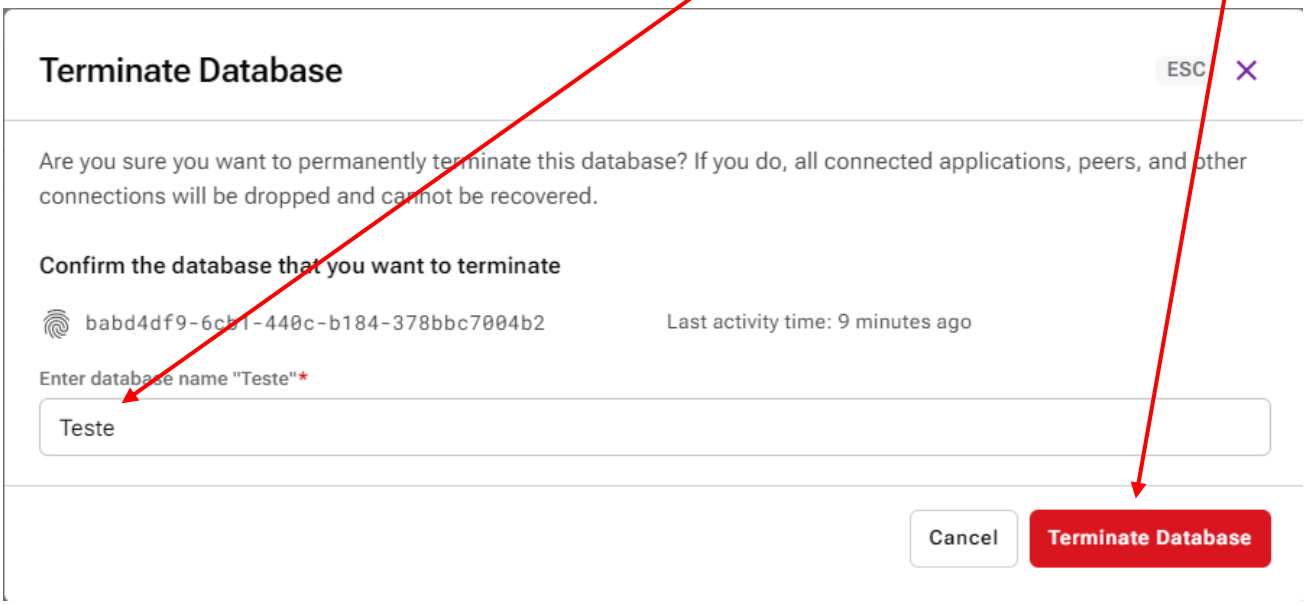
5) Com o banco Teste ativo e selecionado **clique** em **Settings**:



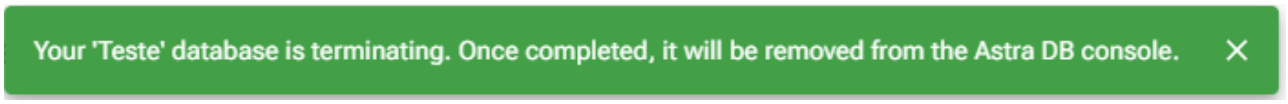
6) Role a tela até o final até aparecer o bloco a seguir. **Clique** em **Terminate Database**



7) Na janela que se abrirá informe o nome do banco "**Teste**" e **clique** novamente em **Terminate** Database.

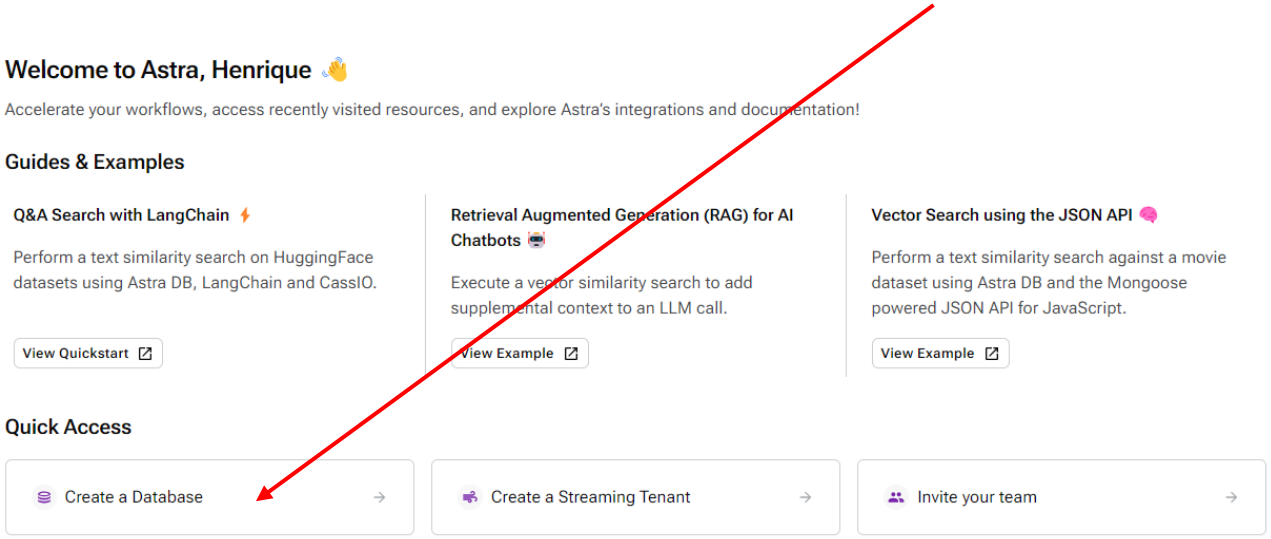


8) A mensagem a seguir será exibida informando que o banco está sendo excluído:



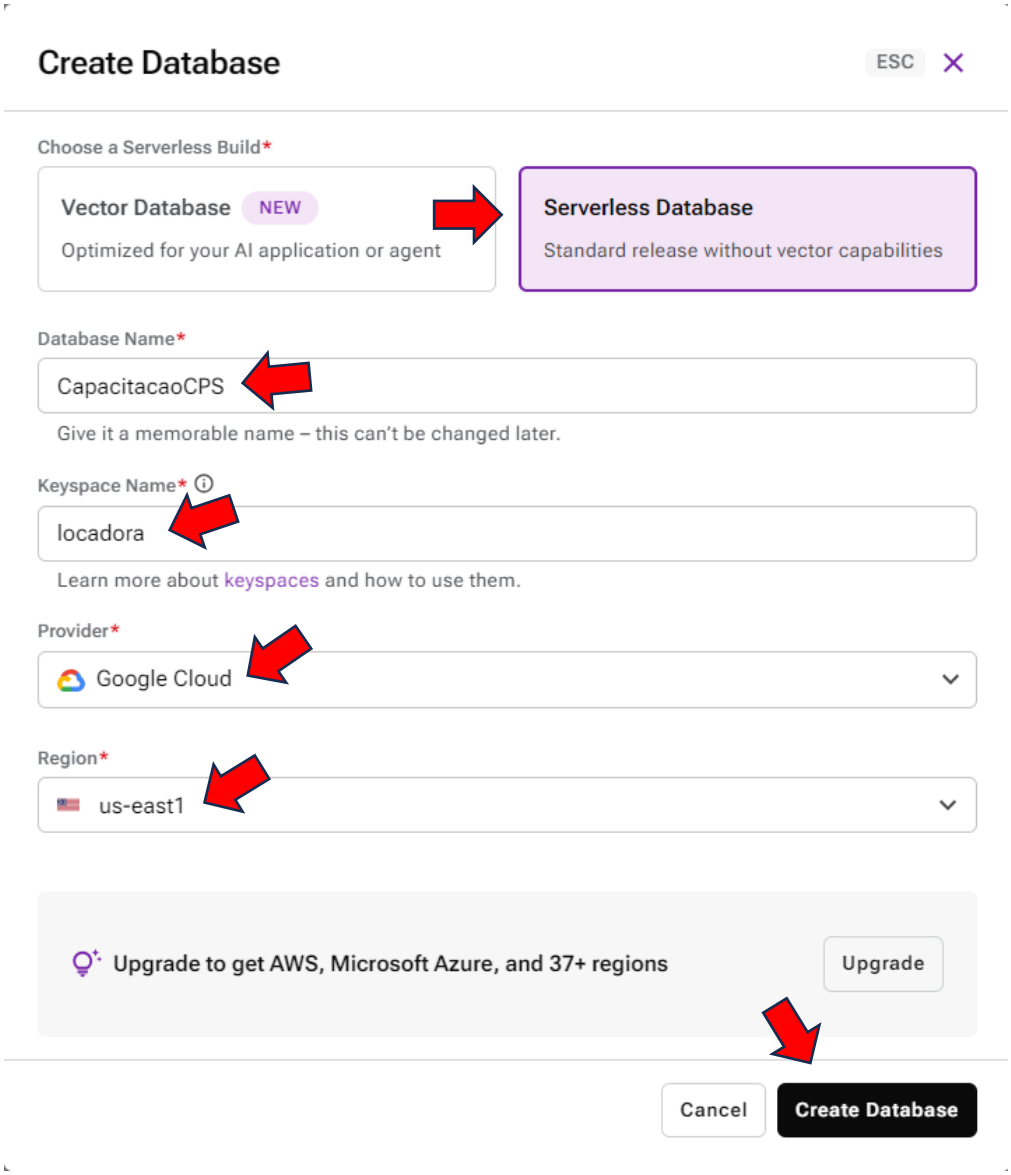
9) Aguarde o banco ser excluído para prosseguirmos com o tutorial. Geralmente demora um pouco essa ação.

10) Você deverá ver uma tela parecida com a seguir. **Clique** em **Create Database**:

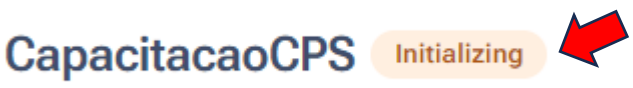


11) Preencha a tela que abriu com os seguintes dados e **clique** em **Create Database**:

Choose a Serverless Build: Serverless Database
Database Name: CapacitacaoCPS
Keyspace Name: locadora
Provider: Google Cloud
Region: us-east1



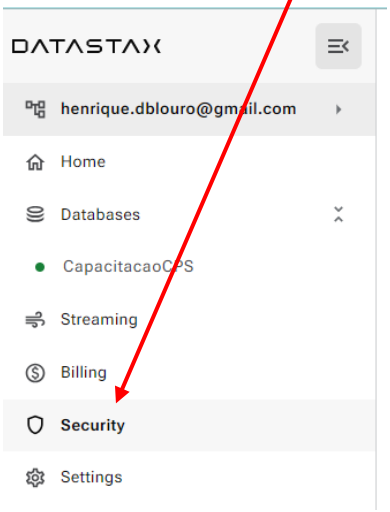
12) Aguarde o banco ser inicializado e liberado. Pode demorar um pouco.



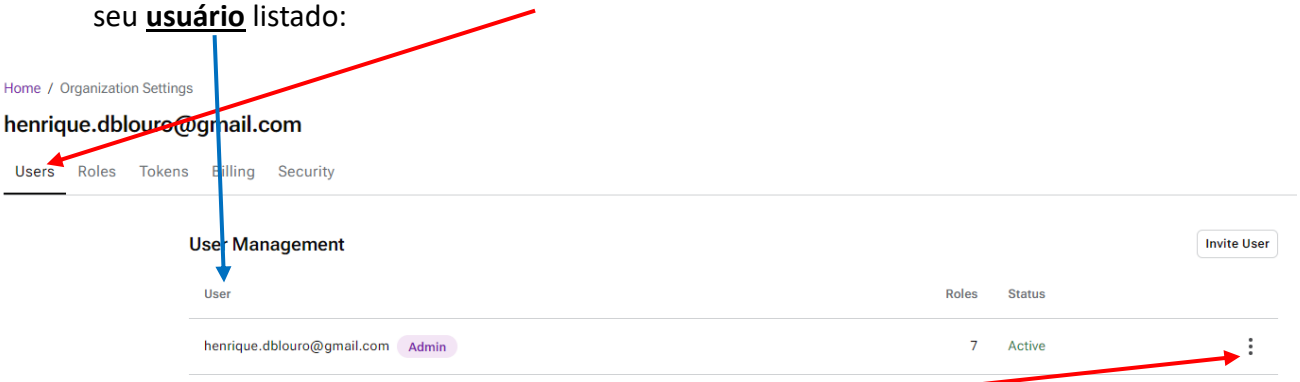
13) Para que possamos usar os comandos básicos desse tutorial, ainda será preciso fazer algumas configurações de segurança.

14) Agora iremos alterar suas credenciais de usuário para permitir vários comandos bloqueados por padrão.

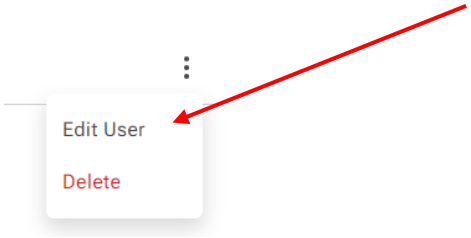
15) Na página principal no menu à esquerda **clique** em **Security**:



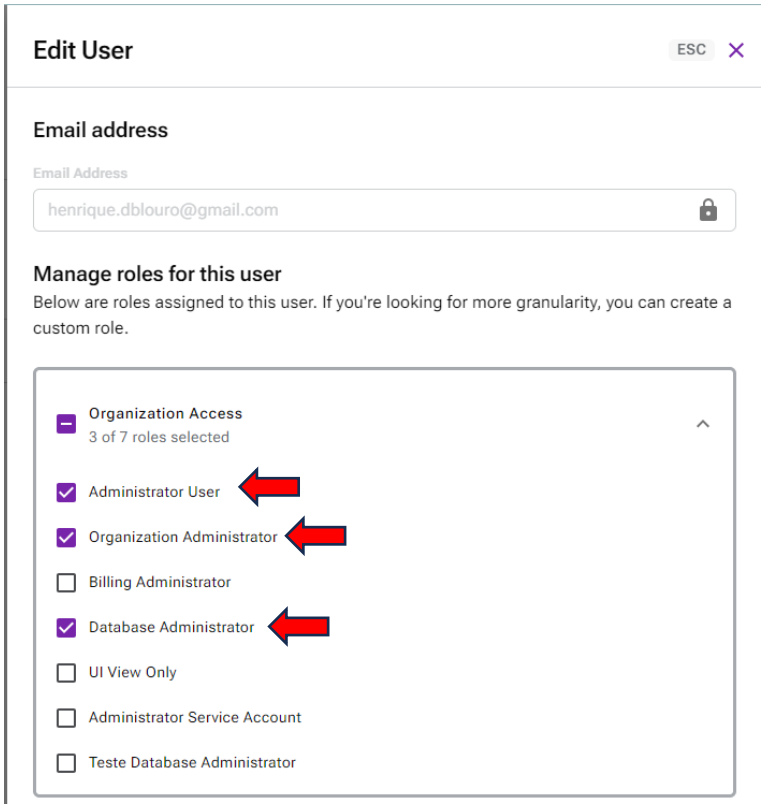
16) Na tela que se abriu **clique** em **Users** e você verá uma tela parecida com a seguir, só que com seu **usuário** listado:



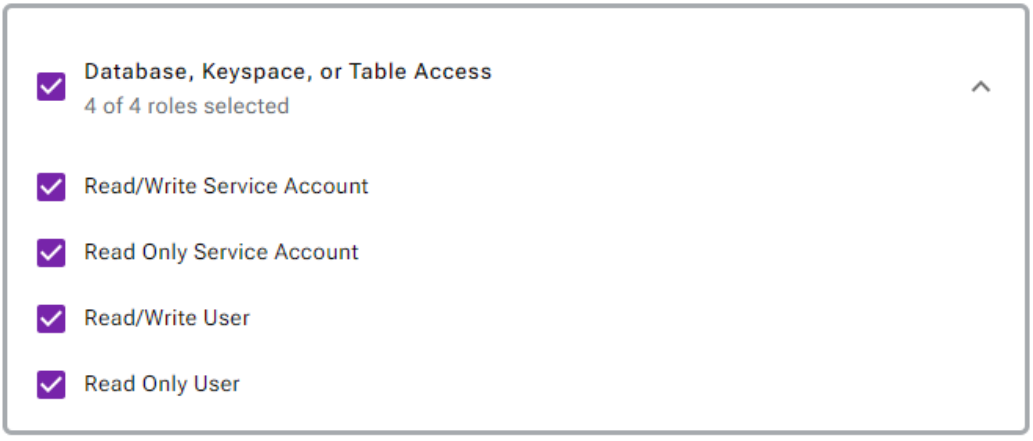
17) **Clique** nos **três pontinhos** à esquerda e no menu que se abrir em **Edit User**:



18) Na janela que se abriu, selecione no bloco **Organization Access** as opções conforme figura a seguir:

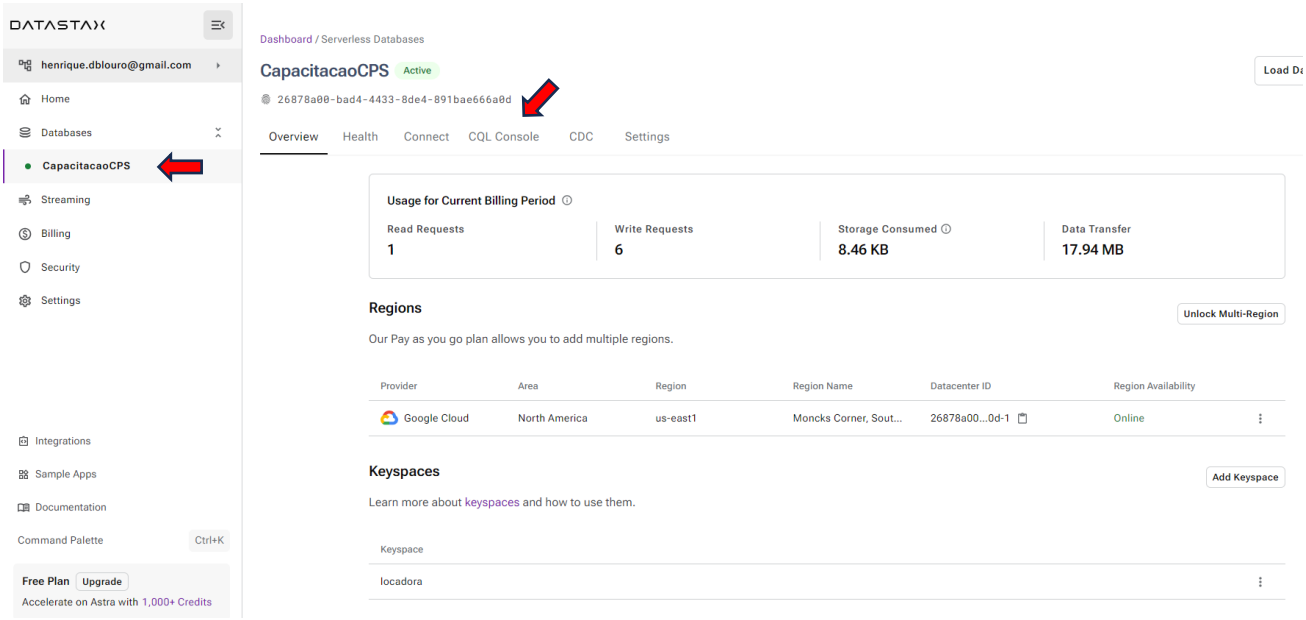


19) Logo abaixo, no bloco Database, Keyspace, or Table Access, selecione todas as opções conforme figura a seguir:

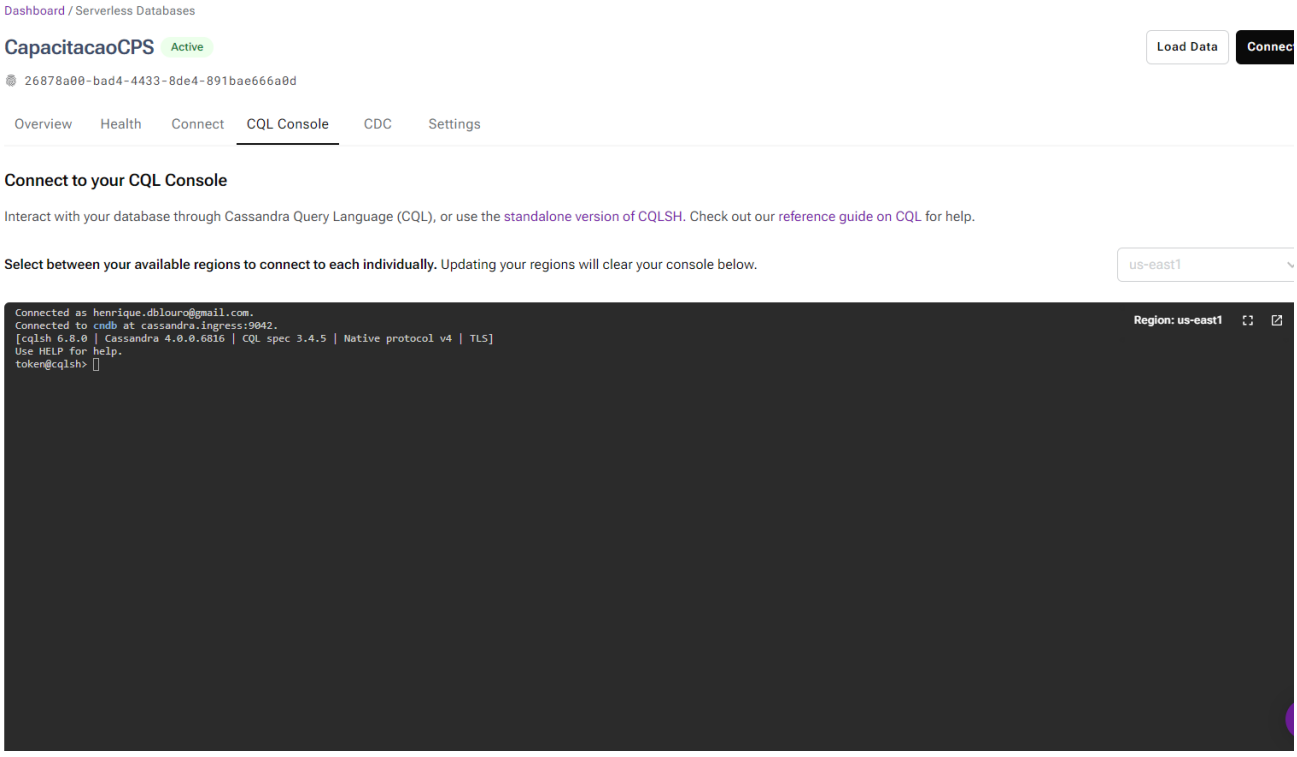


20) Feche a janela com **ESC** ou **clicando** no **X** no canto superior direito. Pronto as configurações de segurança com permissões de acesso e ações foram alteradas para seu usuário.

21) Com o banco ativo clique no **CQL Console** .



22) Assim que o CQL Console estiver disponível, você verá a seguinte tela:



23) Se a tela anterior foi apresentada podemos começar a testar os comandos básicos do Cassandra.

24) Vamos começar listando todas as keyspaces existentes:

Digite o seguinte comando e tecle ENTER:

DESCRIBE KEYSPACES;

Informações parecidas deverão aparecer. Note que a Keyspace “locadora” que criamos junto ao banco é listada.

```
Connected as henrique.dblouro@gmail.com.
Connected to cndb at cassandra.ingress:9042.
[cqlsh 6.8.0 | Cassandra 4.0.0.6816 | CQL spec 3.4.5 | Native protocol v4 | TLS]
Use HELP for help.
token@cqlsh> DESCRIBE KEYSPACES

system_auth    system    data_endpoint_auth  datastax_sla          system_views
system_schema  locadora  system_traces       system_virtual_schema

token@cqlsh>
```

25) Esse seria o comando para criar a keyspace “locadora”, mas a plataforma já fez isso por nós. Coloco aqui apenas para informação:

```
CREATE KEYSPACE locadora
WITH
REPLICATION = {
'class': 'SimpleStrategy',
'replication_factor' : 1
};
```

26) Vamos setar agora o keyspace a ser utilizado, ou seja, informar ao banco qual keyspace iremos usar. Digite o comando a seguir e tecle **ENTER**.

```
USE locadora;
```

27) Veja que o prompt mudou indicando que estamos trabalhando na keyspace **“locadora”**.

```
Connected as henrique.dblouro@gmail.com.
Connected to cndb at cassandra.ingress:9042.
[cqlsh 6.8.0 | Cassandra 4.0.0.6816 | CQL spec 3.4.5 | Native protocol v4 | TLS]
Use HELP for help.
token@cqlsh> DESCRIBE KEYSPACES

system_auth    system    data_endpoint_auth  datastax_sla          system_views
system_schema  locadora  system_traces       system_virtual_schema

token@cqlsh> use locadora;
token@cqlsh:locadora>
```

28) Criaremos agora nossa primeira tabela na keyspace **"locadora"**. Digite o seguinte comando e tecle **ENTER**.

```
CREATE TABLE cliente (
cli_id int PRIMARY KEY,
cli_cpf varchar,
cli_nome varchar,
cli_cidade varchar,
);
```

Você deverá ter a seguinte tela:

```
Connected as henrique.dblouro@gmail.com.
Connected to cndb at cassandra.ingress:9042.
[cqlsh 6.8.0 | Cassandra 4.0.0.6816 | CQL spec 3.4.5 | Native protocol v4 | TLS]
Use HELP for help.
token@cqlsh> DESCRIBE KEYSPACES

system_auth    system    data_endpoint_auth  datastax_sla          system_views
system_schema  locadora  system_traces       system_virtual_schema

token@cqlsh> use locadora;
token@cqlsh:locadora> CREATE TABLE cliente (
                  ... cli_id int PRIMARY KEY,
                  ... cli_cpf varchar,
                  ... cli_nome varchar,
                  ... cli_cidade varchar,
                  ... );
token@cqlsh:locadora>
```

29) Vamos verificar se a estrutura da tabela foi criada corretamente. Digite o seguinte comando:

```
DESCRIBE KEYSPACE locadora;
```

Você deverá ver a seguinte tela:

```
token@cqlsh:locadora> DESCRIBE KEYSPACE locadora;

CREATE KEYSPACE locadora WITH replication = {'class': 'NetworkTopologyStrategy', 'us-east1': '3'}  AND durable_writes = true;

CREATE TABLE locadora.cliente (
    cli_id int PRIMARY KEY,
    cli_cidade text,
    cli_cpf text,
    cli_nome text
) WITH additional_write_policy = '99PERCENTILE'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.UnifiedCompactionStrategy'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    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 = 'BLOCKING'
    AND speculative_retry = '99PERCENTILE';

token@cqlsh:locadora>
```

Perceba que o gerenciador seta uma série de parâmetros automaticamente. Infelizmente essa é uma capacitação de fundamentos e não entraremos em detalhes sobe eles. Além disso o comando DESCRIBE KEYSPACE locadora mostra a linha de comando da criação da keyspace.

Até aqui você deve estar percebendo uma certa semelhança entre a linguagem SQL e a CQL do Cassandra. Acredito que isso seja proposital. Para que inventar a roda novamente? Se ainda tem dúvidas, veja as diferenças entre os bancos relacionais e o NoSQL Cassandra no material da segunda semana.

30) Caso queira, pode listar apenas a estrutura da tabela **cliente**. Digite o seguinte comando:

DESCRIBE TABLE cliente;

```
token@cqlsh:locadora> describe table cliente;

CREATE TABLE locadora.cliente (
    cli_id int PRIMARY KEY,
    cli_cidade text,
    cli_cpf text,
    cli_nome text
) WITH additional_write_policy = '99PERCENTILE'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.UnifiedCompactionStrategy'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    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 = 'BLOCKING'
    AND speculative_retry = '99PERCENTILE';

token@cqlsh:locadora>
```

31) Vamos inserir algumas rows nessa tabela. Digite (ou copie e cole) os seguintes comandos:

Para colar no Console use o botão direito do mouse

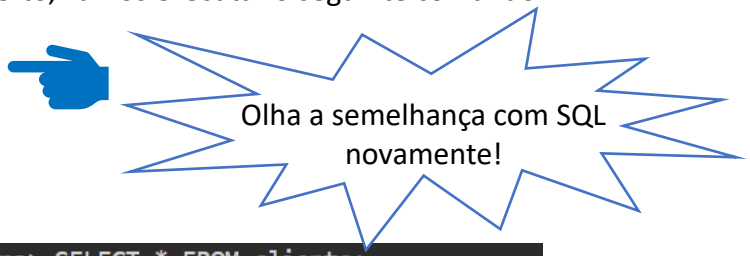
INSERT INTO cliente (cli_id, cli_cpf, cli_nome, cli_cidade) VALUES (1,'11111111111','Bil Clinton','New York');
INSERT INTO cliente (cli_id, cli_cpf, cli_nome, cli_cidade) VALUES (2,'22222222222','Luisa Sonsa','Sao Paulo');
INSERT INTO cliente (cli_id, cli_cpf, cli_nome, cli_cidade) VALUES (3,'33333333333','Maria Bonita','Rio de Janeiro');
INSERT INTO cliente (cli_id, cli_cpf, cli_nome, cli_cidade) VALUES (4,'44444444444','Jesse James','Salvador');
INSERT INTO cliente (cli_id, cli_cpf, cli_nome, cli_cidade) VALUES (5,'55555555555','Hubert Iro','Recife');
INSERT INTO cliente (cli_id, cli_cpf, cli_nome, cli_cidade) VALUES (6,'66666666666','Carlos Tiago','Campinas');

```
token@cqlsh:locadora> INSERT INTO cliente (cli_id, cli_cpf, cli_nome, cli_cidade) VALUES (1,'11111111111','Bil Clinton','New York');
token@cqlsh:locadora> INSERT INTO cliente (cli_id, cli_cpf, cli_nome, cli_cidade) VALUES (2,'22222222222','Luisa Sonsa','Sao Paulo');
token@cqlsh:locadora> INSERT INTO cliente (cli_id, cli_cpf, cli_nome, cli_cidade) VALUES (3,'33333333333','Maria Bonita','Rio de Janeiro');
token@cqlsh:locadora> INSERT INTO cliente (cli_id, cli_cpf, cli_nome, cli_cidade) VALUES (4,'44444444444','Jesse James','Salvador');
token@cqlsh:locadora> INSERT INTO cliente (cli_id, cli_cpf, cli_nome, cli_cidade) VALUES (5,'55555555555','Hubert Iro','Recife');
token@cqlsh:locadora> INSERT INTO cliente (cli_id, cli_cpf, cli_nome, cli_cidade) VALUES (6,'66666666666','Carlos Tiago','Campinas');
token@cqlsh:locadora>
```

Region: us-east1

32) Para verificarmos se deu tudo certo, vamos executar o seguinte comando:

SELECT * FROM cliente;



A saída deverá ser:

```
token@cqlsh:locadora> SELECT * FROM cliente;

 cli_id | cli_cidade     | cli_cpf     | cli_nome
--------+----------------+-------------+--------------
      5 |         Recife | 55555555555 |   Hubert Iro
      1 |       New York | 11111111111 |  Bil Clinton
      2 |      Sao Paulo | 22222222222 |  Luisa Sonsa
      4 |       Salvador | 44444444444 |  Jesse James
      6 |       Campinas | 66666666666 | Carlos Tiago
      3 | Rio de Janeiro | 33333333333 | Maria Bonita

(6 rows)
token@cqlsh:locadora>
```

33) Crie agora as duas tabelas a seguir:

CREATE TABLE filme (
flm_id int PRIMARY KEY,
flm_titulo varchar,
flm_midia varchar,
flm_preco float,
flm_id_gen int
);

```
CREATE TABLE genero (
gen_id int PRIMARY KEY,
gen_descricao varchar,
);
```

34) Você pode listar todas as tabelas criadas dentro da keyspace “locadora” se quiser, usando o seguinte comando:

```
DESCRIBE KEYSPACE LOCADORA;
```

35) Em seguida insira os seguintes dados:

```
INSERT INTO filme (flm_id,flm_titulo,flm_midia,flm_preco,flm_id_gen) VALUES (1,'O Fantasma da Opera','DVD', 7, 1);
INSERT INTO filme (flm_id,flm_titulo,flm_midia,flm_preco,flm_id_gen) VALUES (2,'Sobre Meninos e Lobos','DVD', 7, 3);
INSERT INTO filme (flm_id,flm_titulo,flm_midia,flm_preco,flm_id_gen) VALUES (3,'Notting Hill','DVD',6,2);
INSERT INTO filme (flm_id,flm_titulo,flm_midia,flm_preco,flm_id_gen) VALUES (4,'O Exorcista','VHS',4,4);
INSERT INTO filme (flm_id,flm_titulo,flm_midia,flm_preco,flm_id_gen) VALUES (5,'O Aviador','DVD',7,1);
INSERT INTO filme (flm_id,flm_titulo,flm_midia,flm_preco,flm_id_gen) VALUES (6,'Guerra nas Estrelas','DVD',7,5);
INSERT INTO filme (flm_id,flm_titulo,flm_midia,flm_preco,flm_id_gen) VALUES (7,'A Tempestade do Seculo','VHS',4,1);
INSERT INTO filme (flm_id,flm_titulo,flm_midia,flm_preco,flm_id_gen) VALUES (8,'O Navio Fantasma','DVD',6,3);
INSERT INTO filme (flm_id,flm_titulo,flm_midia,flm_preco,flm_id_gen) VALUES (9,'A Taca do Mundo e Nossa','DVD',7,2);
INSERT INTO filme (flm_id,flm_titulo,flm_midia,flm_preco,flm_id_gen) VALUES (10,'Efeito Borboleta','DVD',6,3);

INSERT INTO genero (gen_id,gen_descricao) VALUES (1,'Drama');
INSERT INTO genero (gen_id,gen_descricao) VALUES (2,'Comedia');
INSERT INTO genero (gen_id,gen_descricao) VALUES (3,'Suspense');
INSERT INTO genero (gen_id,gen_descricao) VALUES (4,'Terror');
INSERT INTO genero (gen_id,gen_descricao) VALUES (5,'Sci Fi');
```

36) Para verificar se os dados foram incluídos com sucesso digite os seguintes comandos:

```
SELECT * FROM filme;

SELECT * FROM genero;
```

Você deverá ter algo semelhante à figura abaixo:

```
token@cqlsh:locadora> SELECT * FROM filme;

 flm_id | flm_id_gen | flm_midia | flm_preco | flm_titulo
--------+------------+-----------+-----------+-------------------------
      5 |          1 |       DVD |         7 |               O Aviador
     10 |          3 |       DVD |         6 |        Efeito Borboleta
      1 |          1 |       DVD |         7 |     O Fantasma da Opera
      8 |          3 |       DVD |         6 |        O Navio Fantasma
      2 |          3 |       DVD |         7 |   Sobre Meninos e Lobos
      4 |          4 |       VHS |         4 |             O Exorcista
      7 |          1 |       VHS |         4 |  A Tempestade do Seculo
      6 |          5 |       DVD |         7 |     Guerra nas Estrelas
      9 |          2 |       DVD |         7 | A Taca do Mundo e Nossa
      3 |          2 |       DVD |         6 |             Notting Hill

(10 rows)
token@cqlsh:locadora> SELECT * FROM genero;

 gen_id | gen_descricao
--------+---------------
      5 |        Sci Fi
      1 |         Drama
      2 |       Comedia
      4 |        Terror
      3 |      Suspense

(5 rows)
token@cqlsh:locadora>
```

Perceba que inserimos primeiro as linhas dos filmes que continham o ID do gênero, sem ter importado ainda as linhas dos gêneros. Isso no modelo relacional teria dado erro de quebra de relacionamento (chave estrangeira) e não seria possível inserir os registros dos filmes, sem primeiro ter inserido os registros dos gêneros. Essa é uma das grandes diferenças do NoSQL para os relacionais. Não há relacionamento. Tudo tem que ser feito na "unha". Os relacionamentos nos bancos relacionais pagam um alto preço em performance, algo muito importante nos bancos NoSQL, pois precisam trabalhar com grande volume de dados de maneira muito rápida. Se você precisa de dados normalizados então precisa de um banco relacional e não um NoSQL.

37) Passaremos agora para as consultas. Utilizaremos o comando SELECT para tanto. Faremos algumas consultas nas tabelas criadas. Qualquer semelhança com o SQL não é mera coincidência. Você pode continuar digitando ou copiando e colando os comandos a seguir.

A qualquer momento que você sair da plataforma, ao acessar o Console CQL novamente, não esqueça de selecionar primeiro a keyspace "locadora" com o comando "use locadora;".

38) Selecionar o id, nome e cidade de um cliente:

```
SELECT cli_id, cli_nome, cli_cidade FROM cliente;
```

```
token@cqlsh> SELECT cli_id, cli_nome, cli_cidade FROM cliente;
InvalidRequest: Error from server: code=2200 [Invalid query] message="No keyspace has been specified. USE a keyspace, or explicitly specify keyspace.tablename"
token@cqlsh> use locadora;
token@cqlsh:locadora> SELECT cli_id, cli_nome, cli_cidade FROM cliente;

 cli_id | cli_nome     | cli_cidade
--------+--------------+---------------
      5 |   Hubert Iro |         Recife
      1 |  Bil Clinton |       New York
      2 |  Luisa Sonsa |      Sao Paulo
      4 |  Jesse James |       Salvador
      6 | Carlos Tiago |       Campinas
      3 | Maria Bonita | Rio de Janeiro

(6 rows)
token@cqlsh:locadora>
```

39) Selecionar os clientes moradores na cidade de Campinas:

```
SELECT * FROM cliente WHERE cli_cidade = 'Campinas';
```

Na consulta anterior, você irá receber uma mensagem de erro, indicando que se for executada uma consulta desse tipo, a performance do banco é imprevisível. Isso acontece pois se essa tabela tivesse bilhões de linhas, possivelmente vários nós do cluster precisariam ser consultados para conseguir trazer o resultado desejado. Seria basicamente o equivalente a um *full table scan* em um banco relacional, mas em uma escala muito maior. Podemos forçar a execução da consulta adicionando a instrução ALLOW FILTERING no final do comando como explicado na mensagem, mas isso não é recomendado a não ser que a tabela seja muito pequena.

```
token@cqlsh:locadora> SELECT * FROM cliente WHERE cli_cidade = 'Campinas';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable perfo
rmance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"
token@cqlsh:locadora>
```

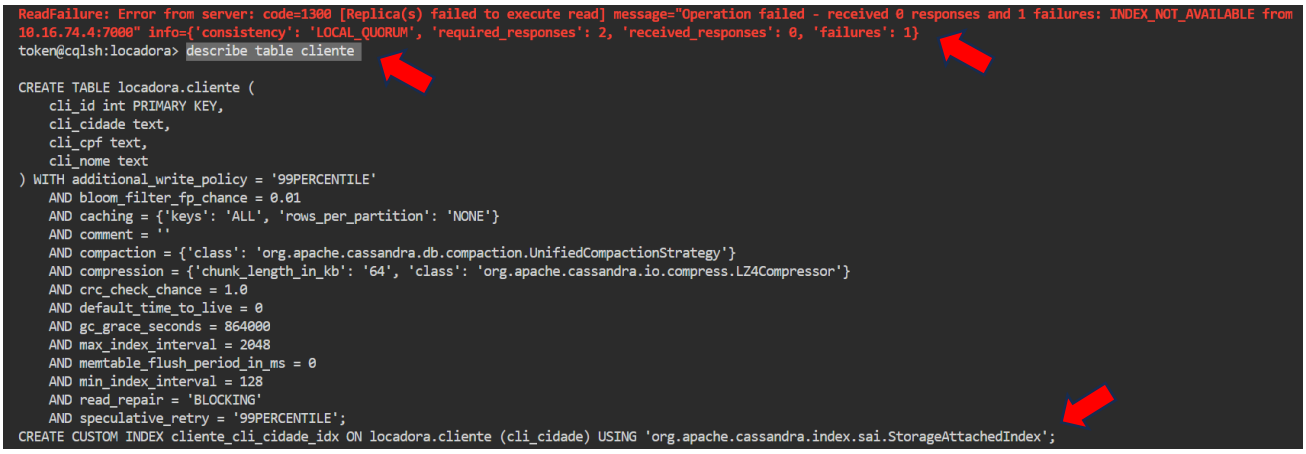
Contudo, há uma forma melhor de resolver isso, criando um índice secundário. Assim como em um banco relacional, no Cassandra é possível criar índices para colunas que não fazem parte da chave primária. Para isso, é utilizado o comando **CREATE INDEX**. Digite o comando a seguir no CQL Console e de ENTER:

```
CREATE INDEX ON locadora.cliente (cli_cidade);
```

Se você continuar a receber a mesma mensagem de erro, aguarde alguns minutos e tente novamente. A criação de um índice geralmente demora um pouco dependendo de como foi feita a criação do banco nos servidores e clusters.

Caso queira conferir se o índice foi criado, use o comando “describe table cliente” para ver a estrutura da tabela cliente.

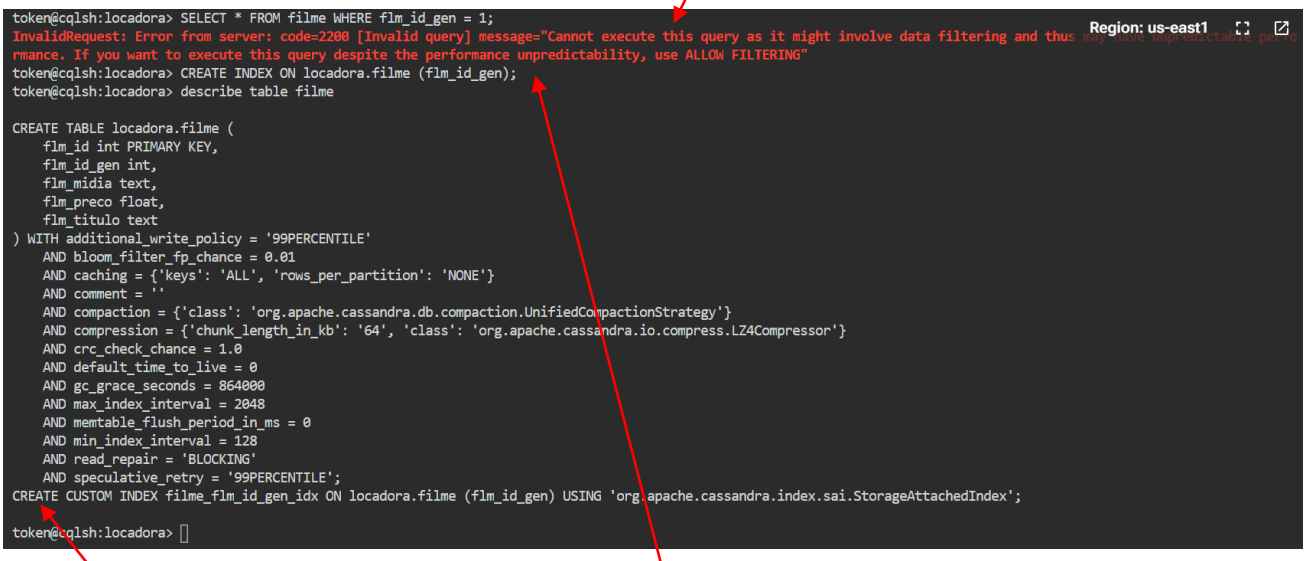
Agora poderemos ver o resultado, conforme mostrado abaixo:



40) Selecionar os filmes do gênero Drama:

SELECT * FROM filme WHERE flm_id_gen = 1;

Tivemos o mesmo problema da consulta anterior:



Iremos corrigir isso criando outro **<u>índice secundário</u>**:

CREATE INDEX ON locadora.filme (flm_id_gen);

41) Depois do índice criado e dado o tempo de espera que fique disponível, o comando select anterior irá funcionar normalmente, conforme figura a seguir:

```
token@cqlsh:locadora> SELECT * FROM filme WHERE flm_id_gen = 1;

 flm_id | flm_id_gen | flm_midia | flm_preco | flm_titulo
--------+------------+-----------+-----------+------------------------
      5 |          1 |       DVD |         7 |              O Aviador
      1 |          1 |       DVD |         7 |   O Fantasma da Opera
      7 |          1 |       VHS |         4 | A Tempestade do Seculo

(3 rows)
token@cqlsh:locadora>
```

42) Selecionar o gênero Suspense na tabela gênero. Nesse caso obteríamos o mesmo erro das consultas anteriores. Dessa vez não criaremos um índice secundário. Utilizaremos a cláusula ALLOW FILTERING no final do comando.

SELECT * FROM genero WHERE gen_descricao = 'Suspense' ALLOW FILTERING;

```
token@cqlsh:locadora> SELECT * FROM genero WHERE gen_descricao = 'Suspense' ALLOW FILTERING;

 gen_id | gen_descricao
--------+---------------
      3 |      Suspense

(1 rows)
token@cqlsh:locadora>
```

43) Vamos contar quantos registros ou linhas (rows) temos na tabela filme:

SELECT count(*) FROM filme;

```
token@cqlsh:locadora> SELECT count(*) FROM filme;

 count
-------
    10

(1 rows)

Warnings :
Aggregation query used without partition key
```

A mensagem informa que estamos efetuando uma query que pode envolver todo o banco e causar lentidão, uma vez que não foi identificada uma partição existente. Por ser uma capacitação de fundamentos, não abordaremos a questão de partições e você pode ignorar esse aviso.

44) Passaremos agora para as alterações nos dados, utilizando o comando UPDATE.

45) Iremos alterar o preço do filme “Efeito Borboleta”, cujo id é 10 e o preço era de R$ 6,00 e passará para R$ 7,50.

```
UPDATE filme SET flm_preco = 7.5 WHERE flm_id = 10;
```

46) Vamos verificar se o preço foi alterado:

```
SELECT * FROM filme;
```

```
token@cqlsh:locadora> UPDATE filme SET flm_preco = 7.5 WHERE flm_id = 10;
token@cqlsh:locadora> SELECT * FROM filme
                  ... ;

 flm_id | flm_id_gen | flm_midia | flm_preco | flm_titulo
--------+------------+-----------+-----------+------------------------
      5 |          1 |       DVD |         7 |              O Aviador
     10 |          3 |       DVD |       7.5 |      Efeito Borboleta
      1 |          1 |       DVD |         7 |     O Fantasma da Opera
      8 |          3 |       DVD |         6 |        O Navio Fantasma
      2 |          3 |       DVD |         7 |   Sobre Meninos e Lobos
      4 |          4 |       VHS |         4 |             O Exorcista
      7 |          1 |       VHS |         4 |  A Tempestade do Seculo
      6 |          5 |       DVD |         7 |     Guerra nas Estrelas
      9 |          2 |       DVD |         7 | A Taca do Mundo e Nossa
      3 |          2 |       DVD |         6 |             Notting Hill

(10 rows)
token@cqlsh:locadora>
```

47) Alteremos agora a cidade da cliente “Maria Bonita”. Ela se mudou do Rio de Janeiro para São Paulo. Seu **id** de cliente é 3:

```
UPDATE cliente SET cli_cidade = 'Sao Paulo' WHERE cli_id = 3;
SELECT * FROM cliente;
```

```
token@cqlsh:locadora> UPDATE cliente SET cli_cidade = 'Sao Paulo' WHERE cli_id = 3
                  ... ;
token@cqlsh:locadora> SELECT * FROM cliente;

 cli_id | cli_cidade | cli_cpf     | cli_nome
--------+------------+-------------+-------------
      5 |     Recife | 55555555555 |   Hubert Iro
      1 |   New York | 11111111111 |  Bil Clinton
      2 |  Sao Paulo | 22222222222 |  Luisa Sonsa
      4 |   Salvador | 44444444444 |  Jesse James
      6 |   Campinas | 66666666666 | Carlos Tiago
      3 |  Sao Paulo | 33333333333 | Maria Bonita

(6 rows)
```

48) E para encerar, iremos apagar alguns registros das tabelas. Utilizaremos o comando **DELETE** para tal.

49) Vamos apagar o registro do cliente “Bil Clinton”, cujo id é 1:

```
DELETE FROM cliente WHERE cli_id=1;
```

Vamos ver se a exclusão deu certo:

```
SELECT * FROM cliente;
```

```
token@cqlsh:locadora> SELECT * FROM cliente;

 cli_id | cli_cidade | cli_cpf     | cli_nome
--------+------------+-------------+--------------
      5 |     Recife | 55555555555 |   Hubert Iro
      1 |   New York | 11111111111 |  Bil Clinton
      2 |  Sao Paulo | 22222222222 |  Luisa Sonsa
      4 |   Salvador | 44444444444 |  Jesse James
      6 |   Campinas | 66666666666 | Carlos Tiago
      3 |  Sao Paulo | 33333333333 | Maria Bonita

(6 rows)
```

Note que na listagem não aparece mais o cliente excluído:

```
token@cqlsh:locadora> DELETE FROM cliente WHERE cli_id=1;
token@cqlsh:locador> SELECT * FROM cliente;

 cli_id | cli_cidade | cli_cpf     | cli_nome
--------+------------+-------------+--------------
      5 |     Recife | 55555555555 |   Hubert Iro
      2 |  Sao Paulo | 22222222222 |  Luisa Sonsa
      4 |   Salvador | 44444444444 |  Jesse James
      6 |   Campinas | 66666666666 | Carlos Tiago
      3 |  Sao Paulo | 33333333333 | Maria Bonita

(5 rows)
```

50) Apagaremos agora o filme “O Exorcista” cujo **id** é 4. A mídia era uma fita VHS e estragou com o tempo:

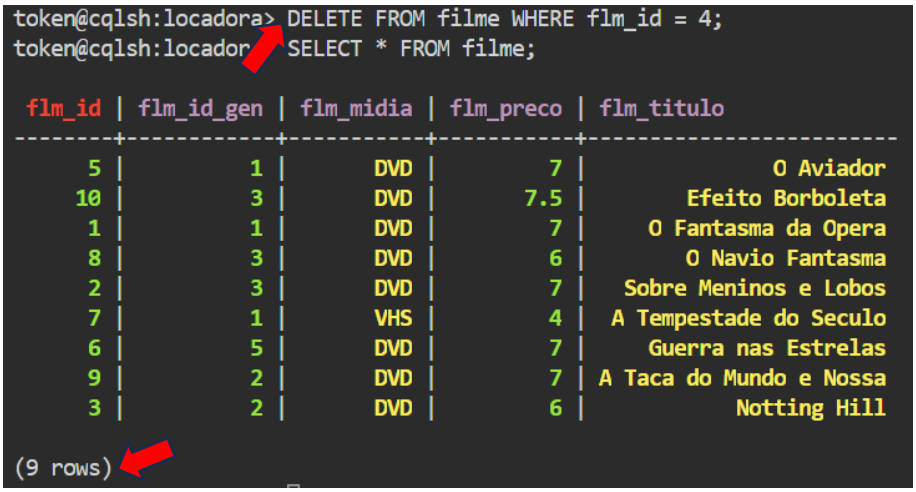
```
DELETE FROM filme WHERE flm_id = 4;
```

```
token@cqlsh:locadora> SELECT * FROM filme;

 flm_id | flm_id_gen | flm_midia | flm_preco | flm_titulo
--------+------------+-----------+-----------+------------------------
      5 |          1 |       DVD |         7 |              O Aviador
     10 |          3 |       DVD |       7.5 |       Efeito Borboleta
      1 |          1 |       DVD |         7 |     O Fantasma da Opera
      8 |          3 |       DVD |         6 |       O Navio Fantasma
      2 |          3 |       DVD |         7 |   Sobre Meninos e Lobos
      4 |          4 |       VHS |         4 |            O Exorcista
      7 |          1 |       VHS |         4 |  A Tempestade do Seculo
      6 |          5 |       DVD |         7 |     Guerra nas Estrelas
      9 |          2 |       DVD |         7 | A Taca do Mundo e Nossa
      3 |          2 |       DVD |         6 |            Notting Hill

(10 rows)
```

A imagem a seguir mostra que o filme não está mais na tabela:



```
token@cqlsh:locadora> DELETE FROM filme WHERE flm_id = 4;
token@cqlsh:locador> SELECT * FROM filme;

 flm_id | flm_id_gen | flm_midia | flm_preco | flm_titulo
--------+------------+-----------+-----------+------------------------
      5 |          1 |       DVD |         7 |               O Aviador
     10 |          3 |       DVD |       7.5 |        Efeito Borboleta
      1 |          1 |       DVD |         7 |     O Fantasma da Opera
      8 |          3 |       DVD |         6 |        O Navio Fantasma
      2 |          3 |       DVD |         7 |   Sobre Meninos e Lobos
      7 |          1 |       VHS |         4 |  A Tempestade do Seculo
      6 |          5 |       DVD |         7 |     Guerra nas Estrelas
      9 |          2 |       DVD |         7 | A Taca do Mundo e Nossa
      3 |          2 |       DVD |         6 |            Notting Hill

(9 rows)
```

51) Assim, chegamos ao final desse tutorial. Fique à vontade para explorar os comandos mostrados aqui ou outros que queira testar.

52) Nesse endereço https://cassandra.apache.org/doc/latest/cassandra/cql/index.html você encontra um tutorial completo dos comandos CQL (Cassandra Query Language).