



Projeto Final da Disciplina Infraestrutura Cassandra – Pós Graduação MIT em Engenharia de dados: Big Data

Rafael Diniz Ramos

Bancos de Dados SQL ou NoSQL?

A pergunta do enunciado vai muito mais ao encontro da necessidade, do tipo de negócio, do que de fato ter uma resposta correta.

SQL

Hoje os bancos de dados baseados em SQL ainda são amplamente mais utilizados, pois já estão no mercado há pelo menos 40 anos, provando assim uma de suas características, a consistência.

Os bancos de dados relacionais, a informação é guardada em tabelas que possuem schemas pré-definidos; além disso, essas tabelas se relacionam, através de um mecanismo de chave primária e chave estrangeira.

Como principais sistemas de gerenciamento de banco de dados (SGBD), temos o Oracle Database, o MySQL e o SQLServer, além de outros.

Algumas características dos bancos de dados SQL:

- São menos escaláveis devido a sua estrutura rígida de schemas.
- Utilizam tabelas que se relacionam para o armazenamento do dado.
- Ampla aceitação em todo mundo.
- Ótimo para armazenar dados altamente estruturados.
- Escalabilidade vertical

NoSQL

Os bancos de dados NoSQL, chegaram a partir da década de 2000 com o aumento relevante da produção de dados, oriundo principalmente da popularização da Internet.

Os sistemas do NoSQL foram projetados para serem distribuídos em várias máquinas (Nós), tornando assim sua escalabilidade flexível e para receberem um alto volume de dados. O

armazenamento da informação, não segue uma estrutura pré-definida como nos bancos de dados relacionais, sua estrutura é semiestruturada.

Os principais tipos de tecnologias NoSQL são bancos de dados chave-valor, de documentos e de grafos.

Redis → Chave-Valor

MongoDB → Documentos

Neo4j → Grafos

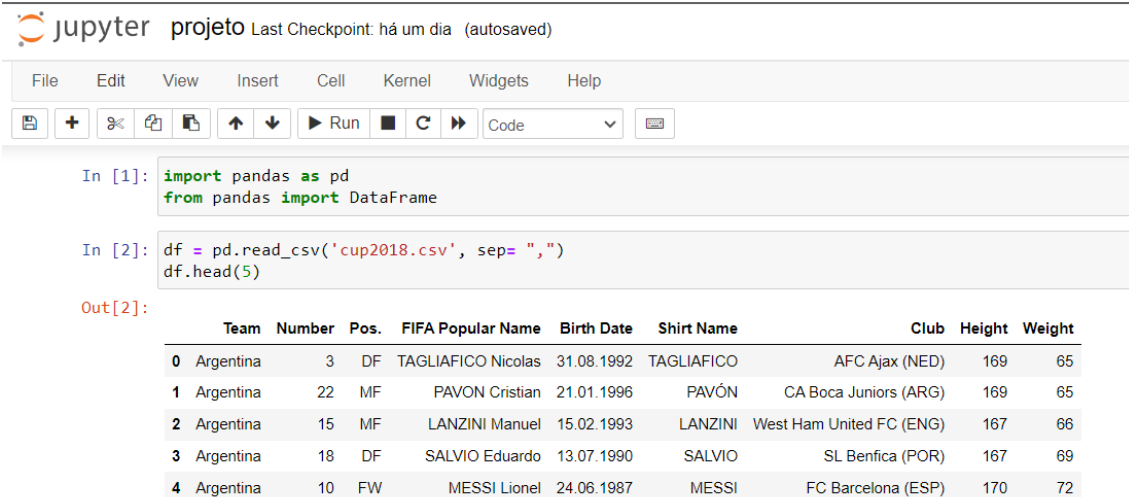
Em seguida vamos mostrar na prática como configurar um ambiente, quais ferramentas utilizar, fazer a carga dos dados e claro, fazer as queries de consultas no Cassandra.

Infraestrutura necessária para tratamento e consulta aos dados

Para tratamento do Data Frame e criação das tabelas para nossas queries de consultas no Cassandra, utilizaremos o Jupyter Notebook, Python, PySpark e a biblioteca do Pandas.

Nosso Data Frame será a respeito da Copa do Mundo de 2018, com dados sobre os jogadores.

Abaixo nosso Data Frame no Jupyter, para extrairmos as tabelas de acordo com nossas queries de consulta.



The screenshot shows a Jupyter Notebook window titled 'projeto' with a status bar indicating 'Last Checkpoint: há um dia (autosaved)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for saving, adding cells, undo, redo, and running code. The code area shows two input cells:

```
In [1]: import pandas as pd
        from pandas import DataFrame

In [2]: df = pd.read_csv('cup2018.csv', sep= ",")
        df.head(5)
```

The output of the second cell is displayed as a table:

	Team	Number	Pos.	FIFA Popular Name	Birth Date	Shirt Name	Club	Height	Weight
0	Argentina	3	DF	TAGLIAFICO Nicolas	31.08.1992	TAGLIAFICO	AFC Ajax (NED)	169	65
1	Argentina	22	MF	PAVON Cristian	21.01.1996	PAVÓN	CA Boca Juniors (ARG)	169	65
2	Argentina	15	MF	LANZINI Manuel	15.02.1993	LANZINI	West Ham United FC (ENG)	167	66
3	Argentina	18	DF	SALVIO Eduardo	13.07.1990	SALVIO	SL Benfica (POR)	167	69
4	Argentina	10	FW	MESSI Lionel	24.06.1987	MESSI	FC Barcelona (ESP)	170	72

Docker

Vamos utilizar um container para rodar o Cassandra na versão 4.1.3.

Abaixo nós criamos o container de nome projeto, utilizando a porta do host 4000 e a do container 9042. Usamos a variável CASSANDRA_CLUSTER_NAME para alterar o nome do cluster em que o nó será incluído.

Além disso, espelhamos através do parâmetro -v o diretório local onde se encontra nosso arquivo csv para o diretório home do container.

```
root@LAPTOP-NV4PR600:/home/devrafa# docker run --name projeto --network cassandra-net -d -e CASSANDRA_CLUSTER_NAME=Infnet -v /home/devrafa:/home -p 4000:9042 cassandra:4.1.3
7ef5711c66503c8fd5987b6268e70a93986fd433c655bc9f440a7665c15028
root@LAPTOP-NV4PR600:/home/devrafa# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
7ef5711c6650   cassandra:4.1.3  "docker-entrypoint.s..."  22 seconds ago Up 21 seconds  7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:4000->9042/tcp, :::4000->9042/tcp  projeto
```

Através do comando abaixo nos conectamos ao container projeto e verificamos que o espelhamento foi bem sucedido.

```
root@7ef5711c6650:/home
root@LAPTOP-NV4PR600:/home/devrafa# docker exec -it projeto /bin/bash
root@7ef5711c6650:/# cd home
root@7ef5711c6650:/home# ls
compras.csv  compras_por_fornecedor.csv  jogador_por_clube.csv  wc2018-players.csv
root@7ef5711c6650:/home#
```

CQLSH

Abaixo, digitamos cqlsh para entrar na console do Cassandra e criarmos a keyspace analyses, onde nossas tabelas para consultas serão criadas.

Foi escolhida a estratégia simples com fator de replicação 1, ou seja, com uma cópia redundante de cada linha.

```
root@7ef5711c6650:/home
root@7ef5711c6650:/home# cqlsh
Connected to Infnet at 127.0.0.1:9042
[cqlsh 6.1.0 | Cassandra 4.1.3 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE analyses WITH replication = {'class': 'SimpleStrategy', 'replication_factor':'1'}
... ;
cqlsh> desc KEYSPACES;

analises  system_auth          system_schema  system_views
system    system_distributed    system_traces  system_virtual_schema

cqlsh>
```

Conectando ao keyspace analyses e criando nossa tabela jogador_por_clube. Após fizemos a cópia dos dados do arquivo para nossa tabela e chamamos um SELECT para ver se estava tudo ok.

```
cqlsh> use analyses ;
cqlsh:analises> CREATE TABLE jogador_por_clube (clube text, nome text, posicao text, PRIMARY KEY((clube), nome, posicao));
cqlsh:analises> desc tables;

jogador_por_clube
```

```

cqlsh:analises>
cqlsh:analises> CREATE TABLE jogador_por_clube (clube text, nome text, posicao text, PRIMARY KEY((clube), nome, posicao));
cqlsh:analises>
cqlsh:analises> COPY analises.jogador_por_clube FROM 'jogador_por_clube.csv' WITH DELIMITER = ',' AND HEADER = TRUE;
Using 3 child processes

Starting copy of analises.jogador_por_clube with columns [clube, nome, posicao].
Processed: 736 rows; Rate:      827 rows/s; Avg. rate:   1330 rows/s
736 rows imported from 1 files in 0.553 seconds (0 skipped).
cqlsh:analises>
cqlsh:analises>
cqlsh:analises>
cqlsh:analises>
cqlsh:analises> SELECT * FROM jogador_por_clube limit 30;

```

clube	nome	posicao
Randers FC (DEN)	HALLDORSSON	GK
Asan Mugunghwa FC (KOR)	S J JU	MF
Real Betis (ESP)	A. GUARDADO	MF
Real Betis (ESP)	J. CAMPBELL	FW
Antalyaspor AS (TUR)	DIOUROU	DF
Malaga CF (ESP)	EN-NESYRI	FW
Kashima Antlers (JPN)	SHOJI	DF
Kashima Antlers (JPN)	UEDA	DF
AS Monaco (FRA)	FALCAO	FW
AS Monaco (FRA)	GLIK	DF
AS Monaco (FRA)	J. MOUTINHO	MF
AS Monaco (FRA)	KEÏTA BALDE	FW
AS Monaco (FRA)	LEMAR	FW
AS Monaco (FRA)	SIDIIBE	DF
AS Monaco (FRA)	SUBAŠIĆ	GK
AS Monaco (FRA)	TIELEMANS	MF
CD Olimpia (HON)	OVALLE	DF
FC Dynamo Kyiv (UKR)	PIVARIĆ	DF
AC Milan (ITA)	ANDRÉ SILVA	FW
AC Milan (ITA)	BIGLIA	MF
AC Milan (ITA)	C. ZAPATA	DF
AC Milan (ITA)	N. KALINIĆ	FW
AC Milan (ITA)	RODRIGUEZ	DF
Yeni Malatyaspor (TUR)	BOUTAIB	FW

Vamos dar início a algumas consultas e saber curiosidades sobre os jogadores que participaram da Copa.

Jogadores dos principais clubes do mundo que participaram da Copa.

Tratamento de nossa base para extrairmos as colunas que queremos para nossa query de consulta.

jupyter

projeto

Last Checkpoint: há um dia (autosaved)

File

Edit

View

Insert

Cell

Kernel

Widgets

Help

+

⌕

📄

📁

⬆

⬇

▶ Run

■

↺

▶

Code

▼

🗨

In [1]:

```
import pandas as pd
from pandas import DataFrame
```

In [2]:

```
df = pd.read_csv('cup2018.csv', sep= ",")
df.head(5)
```

Out[2]:

	Team	Number	Pos.	FIFA Popular Name	Birth Date	Shirt Name	Club	Height	Weight
0	Argentina	3	DF	TAGLIAFICO Nicolas	31.08.1992	TAGLIAFICO	AFC Ajax (NED)	169	65
1	Argentina	22	MF	PAVON Cristian	21.01.1996	PAVÓN	CA Boca Juniors (ARG)	169	65
2	Argentina	15	MF	LANZINI Manuel	15.02.1993	LANZINI	West Ham United FC (ENG)	167	66
3	Argentina	18	DF	SALVIO Eduardo	13.07.1990	SALVIO	SL Benfica (POR)	167	69
4	Argentina	10	FW	MESSI Lionel	24.06.1987	MESSI	FC Barcelona (ESP)	170	72

In [4]:

```
novo_cup = df.loc[:,['Shirt Name', 'Club', 'Pos.']]
novo_cup.rename(columns={'Shirt Name': 'Nome', 'Club': 'Clube', 'Pos.': 'Posicao'}, inplace = True)
novo_cup = novo_cup[['Clube', 'Nome', 'Posicao']]
novo_cup.to_csv('jogador_por_clube.csv', index= False)
novo_cup.head(5)
```

Out[4]:

	Clube	Nome	Posicao
0	AFC Ajax (NED)	TAGLIAFICO	DF
1	CA Boca Juniors (ARG)	PAVÓN	MF
2	West Ham United FC (ENG)	LANZINI	MF
3	SL Benfica (POR)	SALVIO	DF
4	FC Barcelona (ESP)	MESSI	FW

cqlsh:analises> select * from jogador_por_clube WHERE clube = 'Real Madrid CF (ESP)';

clube	nome	posicao
Real Madrid CF (ESP)	ASENSIO	MF
Real Madrid CF (ESP)	CARVAJAL	DF
Real Madrid CF (ESP)	CASEMIRO	MF
Real Madrid CF (ESP)	HAKIMI	DF
Real Madrid CF (ESP)	ISCO	MF
Real Madrid CF (ESP)	K. NAVAS	GK
Real Madrid CF (ESP)	KOVAČIĆ	MF
Real Madrid CF (ESP)	KROOS	MF
Real Madrid CF (ESP)	LUCAS V.	FW
Real Madrid CF (ESP)	MARCELO	DF
Real Madrid CF (ESP)	MODRIĆ	MF
Real Madrid CF (ESP)	NACHO	DF
Real Madrid CF (ESP)	RAMOS	DF
Real Madrid CF (ESP)	RONALDO	FW
Real Madrid CF (ESP)	VARANE	DF

(15 rows)

Vamos saber os jogadores mais novos das seleções que jogaram a Copa do Mundo 2018.

Tratamento de nossa base de dados.

```
In [12]: #jogadores mais novos
novo_cup = df.loc[:,['Team', 'Birth Date', 'Shirt Name']]
novo_cup2 = novo_cup[['Team', 'Birth Date', 'Shirt Name']]
novo_cup2['Birth Date'] = pd.to_datetime(df['Birth Date'], format='%d.%m.%Y')
novo_cup2.to_csv('jogadores_mais_novos.csv', index=False)
novo_cup2.head(5)
```

Out[12]:

	Team	Birth Date	Shirt Name
0	Argentina	1992-08-31	TAGLIAFICO
1	Argentina	1996-01-21	PAVÓN
2	Argentina	1993-02-15	LANZINI
3	Argentina	1990-07-13	SALVIO
4	Argentina	1987-06-24	MESSI

Criando a tabela no CQLSH

```
root@7ef5711c6650:/home
cqlsh:analises> CREATE TABLE jogadores_mais_novos (birth_date date, shirt_name text, team text, PRIMARY KEY((birth_date), shirt_name, team));
cqlsh:analises> desc tables;

jogador_por_clube  jogadores_mais_novos
cqlsh:analises>
```

Cópia dos dados para a tabela.

```
root@1f44cc26d915:/home
cqlsh:analises> COPY jogadores_mais_novos FROM 'jogadores_mais_novos.csv' WITH DELIMITER = ',' AND HEADER = TRUE;
Using 3 child processes

Starting copy of analises.jogadores_mais_novos with columns [team, birth_date, shirt_name].
Processed: 736 rows; Rate: 999 rows/s; Avg. rate: 1563 rows/s
736 rows imported from 1 files in 0.471 seconds (0 skipped).
cqlsh:analises>
```

```
(1 rows)
cqlsh:analises> SELECT * FROM jogadores_mais_novos WHERE team = 'France' AND birth_date > '1998-01-01';

team | birth_date | shirt_name
-----+-----+-----
France | 1998-12-20 | MBAPPE

(1 rows)
cqlsh:analises>
```

```
cqlsh:analises> SELECT * FROM jogadores_mais_novos WHERE team = 'Belgium' AND birth_date > '1997-01-01';

team | birth_date | shirt_name
-----+-----+-----
Belgium | 1997-05-07 | TIELEMANS

(1 rows)
```

```
cqlsh:analises> SELECT * FROM jogadores_mais_novos WHERE team = 'Brazil' AND birth_date > '1997-01-01';

team | birth_date | shirt_name
-----+-----+-----
Brazil | 1997-04-03 | G. JESUS

(1 rows)
cqlsh:analises>
```

```
(12 rows)
cqlsh:analises> SELECT * FROM jogadores_mais_novos WHERE team = 'Spain' AND birth_date > '1996-01-01';

team | birth_date | shirt_name
-----+-----+-----
Spain | 1996-01-21 | ASENSIO
```

Agora iremos descobrir os jogadores mais altos de algumas seleções.

Tratamento de nossa base de dados.

```
In [5]: # jogadores mais altos
novo_cup = df.loc[:, ['Team', 'Height', 'Shirt Name']]
novo_cup3 = novo_cup[['Team', 'Height', 'Shirt Name']]
novo_cup3.to_csv('jogadores_mais_altos.csv', index=False)
novo_cup3.head(5)
```

Out[5]:

	Team	Height	Shirt Name
0	Argentina	169	TAGLIAFICO
1	Argentina	169	PAVÓN
2	Argentina	167	LANZINI
3	Argentina	167	SALVIO
4	Argentina	170	MESSI

Criando a tabela no CQLSH.

```
cqlsh> use analises ;
cqlsh:analises>
cqlsh:analises> CREATE TABLE jogadores_mais_altos (team text, height smallint, shirt_name text, PRIMARY KEY((team), height, shirt_name ));
cqlsh:analises>
cqlsh:analises> desc tables
jogador_por_clube jogadores_mais_altos
```

Fazendo a carga de dados do arquivo para a tabela.

```
(0 rows)
cqlsh:analises> COPY jogadores_mais_altos FROM 'jogadores_mais_altos.csv' WITH DELIMITER = ',' AND HEADER = TRUE;
Using 3 child processes
```

```
Starting copy of analises.jogadores_mais_altos with columns [team, height, shirt_name].
```

```
Processed: 736 rows; Rate: 990 rows/s; Avg. rate: 1552 rows/s
```

```
736 rows imported from 1 files in 0.475 seconds (0 skipped).
```

```
cqlsh:analises> select * from jogadores_mais_altos limit 10;
```

team	height	shirt_name
Peru	169	CUEVA
Peru	169	RUIDIAZ
Peru	169	TRAUCO
Peru	170	FLORES
Peru	172	CORZO
Peru	172	YOTUN
Peru	174	AQUINO
Peru	174	HURTADO
Peru	174	POLO
Peru	177	FARFAN

```
cqlsh:analises> select * FROM jogadores_mais_altos WHERE team = 'Brazil' AND height > 194;
```

team	height	shirt_name
Brazil	195	CASSIO

```
(1 rows)
```

```
cqlsh:analises> select * FROM jogadores_mais_altos WHERE team = 'Germany' AND height > 194;
```

team	height	shirt_name
Germany	195	SÜLE

```
(1 rows)
```

```
cqlsh:analises> select * FROM jogadores_mais_altos WHERE team = 'Serbia' AND height > 194;
```

team	height	shirt_name
Serbia	195	MILENKOVIĆ
Serbia	195	STOJKOVIĆ

```
(2 rows)
```

```
cqlsh:analises> select * FROM jogadores_mais_altos WHERE team = 'Denmark' AND height > 199;
```

team	height	shirt_name
Denmark	200	VESTERGAARD

```
(1 rows)
```

PySpark

Nota: Para essa parte do projeto, criei os containers de nome cassandra e pyspark, utilizando o mesmo comando docker run usado anteriormente e com o devido espelhamento da pasta local para a pasta home dos containers.


```
root@1f44cc26d915:/home
root@LAPTOP-NV4PR600:/home/devnafa# docker ps
CONTAINER ID   IMAGE      NAMES      COMMAND                  CREATED    STATUS      PORTS
1f44cc26d915   cassandra:4.1.3   cassandra   "docker-entrypoint.s..." 2 days ago Up About an hour   7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9042->9042/tcp, :::9042->9042/tcp
fb0f8c67d2bf   jupyter/pyspark-notebook:latest   "tini -g -- start-no..." 2 days ago Up About an hour (healthy)   0.0.0.0:4040->4040/tcp, :::4040->4040/tcp, 0.0.0.0:8888->8888/tcp, :::8888->8888/tcp
root@LAPTOP-NV4PR600:/home/devnafa# docker exec -it cassandra /bin/bash
root@1f44cc26d915:/# cd /home
root@1f44cc26d915:/home# ls
compras.csv  compras_por_fornecedor.csv  jogadores_mais_altos.csv  jogadores_mais_novos.csv  jogador_por_clube.csv  wc2018-players.csv
root@1f44cc26d915:/home#
```

Vamos agora fazer nossas consultas utilizando o PySpark com o Jupyter Notebook.

Primeiro através do comando `docker ps`, vamos verificar quais containers estão ativos.

Através do comando `docker exec -it pyspark /bin/bash`, acessamos o container `pyspark` onde temos a imagem do `Jupyter-pyspark`.

Através do comando `jupyter notebook --ip 0.0.0.0 --port 8888 --allow-root` chamamos o Jupyter.

```
root@LAPTOP-NV4PR600:/home/devnafa# docker ps
CONTAINER ID   IMAGE      NAMES      COMMAND                  CREATED    STATUS      PORTS
1f44cc26d915   cassandra:4.1.3   cassandra   "docker-entrypoint.s..." 2 days ago Up 31 minutes   7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9042->9042/tcp, :::9042->9042/tcp
fb0f8c67d2bf   jupyter/pyspark-notebook:latest   "tini -g -- start-no..." 2 days ago Up 31 minutes (unhealthy)   0.0.0.0:4040->4040/tcp, :::4040->4040/tcp, 0.0.0.0:8888->8888/tcp, :::8888->8888/tcp
root@LAPTOP-NV4PR600:/home/devnafa# docker exec -it pyspark /bin/bash
(base) jovyan@fb0f8c67d2bf:~$ jupyter notebook --ip 0.0.0.0 --port 8888 --allow-root
[I 2023-12-17 22:36:59.966 ServerApp] Package notebook took 0.0000s to import
[I 2023-12-17 22:36:59.985 ServerApp] Package jupyter_lsp took 0.0174s to import
[W 2023-12-17 22:36:59.985 ServerApp] A `_jupyter_server_extension_points` function was not found in jupyter_lsp. Instead, a `_jupyter_server_extension_paths` function was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.
[I 2023-12-17 22:36:59.987 ServerApp] Package jupyter_server_mathjax took 0.0017s to import
[I 2023-12-17 22:36:59.996 ServerApp] Package jupyter_server_terminals took 0.0081s to import
[I 2023-12-17 22:36:59.930 ServerApp] Package jupyterlab took 0.0000s to import
[I 2023-12-17 22:36:59.930 ServerApp] Package jupyterlab_git took 0.0395s to import
[I 2023-12-17 22:36:59.934 ServerApp] Package nbclassic took 0.0032s to import
[W 2023-12-17 22:36:59.937 ServerApp] A `_jupyter_server_extension_points` function was not found in nbclassic. Instead, a `_jupyter_server_extension_paths` function was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.
[I 2023-12-17 22:36:59.938 ServerApp] Package nbdtm took 0.0000s to import
[I 2023-12-17 22:36:59.938 ServerApp] Package notebook_shim took 0.0000s to import
[W 2023-12-17 22:36:59.938 ServerApp] A `_jupyter_server_extension_points` function was not found in notebook_shim. Instead, a `_jupyter_server_extension_paths` function was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.
[I 2023-12-17 22:36:59.940 ServerApp] jupyter_lsp | extension was successfully linked.
[I 2023-12-17 22:36:59.940 ServerApp] jupyter_server_mathjax | extension was successfully linked.
[I 2023-12-17 22:36:59.954 ServerApp] jupyter_server_terminals | extension was successfully linked.
[I 2023-12-17 22:36:59.964 ServerApp] jupyterlab | extension was successfully linked.
[I 2023-12-17 22:36:59.964 ServerApp] jupyterlab_git | extension was successfully linked.
[I 2023-12-17 22:36:59.971 ServerApp] nbclassic | extension was successfully linked.
[I 2023-12-17 22:36:59.971 ServerApp] nbdtm | extension was successfully linked.
[I 2023-12-17 22:36:59.980 ServerApp] notebook | extension was successfully linked.
[I 2023-12-17 22:37:00.347 ServerApp] notebook_shim | extension was successfully linked.
[I 2023-12-17 22:37:00.428 ServerApp] notebook_shim | extension was successfully loaded.
[I 2023-12-17 22:37:00.433 ServerApp] jupyter_lsp | extension was successfully loaded.
[I 2023-12-17 22:37:00.434 ServerApp] jupyter_server_mathjax | extension was successfully loaded.
[I 2023-12-17 22:37:00.435 ServerApp] jupyter_server_terminals | extension was successfully loaded.
[I 2023-12-17 22:37:00.440 LabApp] JupyterLab extension loaded from /opt/conda/lib/python3.11/site-packages/jupyterlab
[I 2023-12-17 22:37:00.440 LabApp] JupyterLab application directory is /opt/conda/share/jupyter/lab
[I 2023-12-17 22:37:00.442 LabApp] Extension Manager is 'pypi'.
[I 2023-12-17 22:37:00.449 ServerApp] jupyterlab | extension was successfully loaded.
[I 2023-12-17 22:37:00.460 ServerApp] jupyterlab_git | extension was successfully loaded.
```

Com o Jupyter aberto em nosso browser, vamos começar nossas consultas as tabelas que criamos anteriormente, mas agora utilizando o PySpark.

Importamos o `pyspark`, o `SparkSession` para criarmos nossa sessão, o pacote de tipos de dados do SQL e todas as funções SQL.

Em seguida atribuímos ao objeto `spark` a sessão criada, informando os pacotes do spark para baixar e a versão do conector para conectar o Spark com o Cassandra.

Em conf, passamos o nome do host e a porta por onde iremos comunicar. Junto a isso, passamos as informações do keyspace e o nome da tabela, para que ela seja encontrada, carregada e seja feita a criação de sua imagem.

No próximo comando, através do `SELECT`, chamamos nossa tabela `jogador_por_clube`.

```
File Edit View Run Kernel Settings Help
+ ✂ 📄 📌 ▶ ■ 🔁 ⏏ Code ▼

[16]: import pyspark
      from pyspark.sql import SparkSession
      from pyspark.sql.types import StructType, StructField, StringType, IntegerType
      from pyspark.sql.functions import *

[2]: spark = (SparkSession.builder.appName("Cassandra")
             .config("spark.jars.packages", "com.datastax.spark:spark-cassandra-connector_2.12:3.4.1")
             .getOrCreate())

[3]: conf = { "spark.cassandra.connection.host": "cassandra",
             "spark.cassandra.connection.port": 9042,
             }
      (
        spark.read.format("org.apache.spark.sql.cassandra")
        .options(**conf)
        .options(keyspace="analises", table="jogador_por_clube")
        .load()
        .createOrReplaceTempView("jogador_por_clube")
      )

[21]: spark.sql('select * from jogador_por_clube').show(2, truncate= False)

+-----+-----+-----+
|clube          |nome      |posicao|
+-----+-----+-----+
|Hibernian FC (SCO)      |MACLAREN|FW     |
|Club Universidad de Chile (CHI)|COOPER  |MF     |
+-----+-----+-----+
only showing top 2 rows
```

Quais jogadores do Barcelona jogaram a Copa?

```
[12]: df.filter(df.clube.like("%Barcelona%")).show(truncate = False)
```

```
+-----+-----+-----+
|clube          |nome      |posicao|
+-----+-----+-----+
|FC Barcelona (ESP)|A. INIESTA|MF     |
|FC Barcelona (ESP)|DEMBELE   |FW     |
|FC Barcelona (ESP)|JORDI ALBA|DF     |
|FC Barcelona (ESP)|MESSI     |FW     |
|FC Barcelona (ESP)|P. COUTINHO|MF     |
|FC Barcelona (ESP)|PAULINHO  |MF     |
|FC Barcelona (ESP)|PIQUÉ     |DF     |
|FC Barcelona (ESP)|RAKITIĆ   |MF     |
|FC Barcelona (ESP)|SERGIO    |MF     |
|FC Barcelona (ESP)|SUAREZ    |FW     |
|FC Barcelona (ESP)|TER STEGEN|GK     |
|FC Barcelona (ESP)|UMTITI    |DF     |
|FC Barcelona (ESP)|VERMAELEN|DF     |
|FC Barcelona (ESP)|Y. MINA   |DF     |
+-----+-----+-----+
```

Aqui nós buscamos apenas os atacantes do Barcelona que jogaram a Copa.

```
[19]: # Atacantes do Barcelona que jogaram a Copa.  
df.select("clube", "nome", "posicao").filter((col("posicao") == "FW" & (col("clube") == "FC Barcelona (ESP)")).show(truncate = False)
```

clube	nome	posicao
FC Barcelona (ESP)	DEMBELE	FW
FC Barcelona (ESP)	MESSI	FW
FC Barcelona (ESP)	SUAREZ	FW

Agora vamos trabalhar com a tabela jogadores_mais_altos.

Abaixo nós filtramos todos os jogadores das seleções acima de 1,95m.

```
[6]: df = spark.sql('select * from jogadores_mais_altos')
```

```
[27]: df.select("team", "height").filter(df.height >= 195).show(100)
```

team	height
Germany	195
Argentina	199
Nigeria	196
Nigeria	197
Panama	197
Serbia	195
Serbia	195
Korea Republic	197
Denmark	195
Denmark	195
Denmark	200
England	196
Belgium	197
Belgium	199
Iceland	198
Sweden	198
Costa Rica	196
Russia	196
Senegal	195
Senegal	196
Senegal	196
Poland	195
Croatia	201
Brazil	195

Aqui geramos uma coluna de classificação de acordo com a altura.

```
[37]: df2 = df.withColumn("classification", expr("CASE WHEN height <= '170' THEN 'BAIXINHO' " +  
"WHEN height <= '180' THEN 'NORMAL' " +  
"WHEN height <= '190' THEN 'ALTO'" +  
"WHEN height >= '191' THEN 'GIRAFa' " +  
"ELSE height END"))
```

```
[38]: df2.show(50,truncate=False)
```

	team	height	shirt_name	classification
	Mexico	166	J. AQUINO	BAIXINHO
	Mexico	167	A. GUARDADO	BAIXINHO
	Mexico	171	M. FABIÁN	NORMAL
	Mexico	173	J. DOS SANTOS	NORMAL
	Mexico	174	JESÚS C.	NORMAL
	Mexico	175	J. HERNÁNDEZ	NORMAL
	Mexico	176	H. LOZANO	NORMAL
	Mexico	177	J. GALLARDO	NORMAL
	Mexico	178	CARLOS V	NORMAL
	Mexico	178	G. DOS SANTOS	NORMAL
	Mexico	178	O. PERALTA	NORMAL
	Mexico	180	H. HERRERA	NORMAL
	Mexico	180	M. LAYÚN	NORMAL
	Mexico	182	H. MORENO	ALTO
	Mexico	182	J. CORONA	ALTO
	Mexico	184	R. MÁRQUEZ	ALTO
	Mexico	185	G. OCHOA	ALTO
	Mexico	186	E. ÁLVAREZ	ALTO
	Mexico	188	A. TALAVERA	ALTO
	Mexico	188	H. AYALA	ALTO
	Mexico	188	RAÚL	ALTO
	Mexico	189	C. SALCEDO	ALTO

Mexico	188	RAÚL	ALTO
Mexico	189	C. SALCEDO	ALTO
Mexico	190	D. REYES	ALTO
Germany	176	KIMMICH	NORMAL
Germany	179	RUDY	NORMAL
Germany	180	GÜNDÖGAN	NORMAL
Germany	180	REUS	NORMAL
Germany	180	ÖZIL	NORMAL
Germany	181	PLATTENHARDT	ALTO
Germany	181	WERNER	ALTO
Germany	182	KROOS	ALTO
Germany	183	BRANDT	ALTO
Germany	185	DRAXLER	ALTO
Germany	185	HECTOR	ALTO
Germany	186	MÜLLER	ALTO
Germany	187	TER STEGEN	ALTO
Germany	189	GINTER	ALTO
Germany	189	GOMEZ	ALTO
Germany	189	GORETZKA	ALTO
Germany	189	KHEDIRA	ALTO
Germany	189	TRAPP	ALTO
Germany	191	RÜDIGER	GIRAFÁ
Germany	192	BOATENG	GIRAFÁ
Germany	192	HUMMELS	GIRAFÁ
Germany	193	NEUER	GIRAFÁ
Germany	195	SÜLE	GIRAFÁ
England	170	STERLING	BAIXINHO
England	173	ROSE	NORMAL
England	174	DELPH	NORMAL
England	175	ALEXANDER-ARNOLD	NORMAL

+-----+-----+-----+-----+

only showing top 50 rows

Vamos agora saber a faixa etária dos jogadores que mais se destaca na Copa e posteriormente demonstrar esse resultado em um gráfico.

```
[56]: spark.sql('select * from jogadores_mais_novos')

[56]: DataFrame[team: string, birth_date: date, shirt_name: string]

[5]: df = spark.sql('select * from jogadores_mais_novos')

[53]: entre_18_25_anos = df.select('team', 'birth_date', 'shirt_name').filter((col('birth_date') >= '1993-01-01') & (col('birth_date') <= '2000-01-01'))
entre_18_25_anos.count()

[53]: 209

[54]: entre_26_30_anos = df.select('team', 'birth_date', 'shirt_name').filter((col('birth_date') >= '1988-01-01') & (col('birth_date') < '1993-01-01'))
entre_26_30_anos.count()

[54]: 335

[51]: entre_31_35_anos = df.select('team', 'birth_date', 'shirt_name').filter((col('birth_date') >= '1983-01-01') & (col('birth_date') < '1988-01-01'))
entre_31_35_anos.count()

[51]: 175

[52]: entre_36_45_anos = df.select('team', 'birth_date', 'shirt_name').filter((col('birth_date') >= '1973-01-01') & (col('birth_date') < '1983-01-01'))
entre_36_45_anos.count()

[52]: 17

[68]: import matplotlib.pyplot as plt

quantidade = [entre_18_25_anos.count(), entre_26_30_anos.count(), entre_31_35_anos.count(), entre_36_45_anos.count()]
faixas = ['18-25 anos', '26-30 anos', '31-35 anos', '36-45 anos']

# Criando um gráfico de barras
plt.bar(faixas, quantidade, color='blue', alpha=0.7)
plt.xlabel('Faixa Etária')
plt.ylabel('Quantidade')
plt.title('Distribuição dos jogadores por faixa etária')

for i, valor in enumerate(contagens):
    plt.text(i, valor, str(valor), ha='center', va='bottom')

plt.show()
```

```
for i, valor in enumerate(contagens):  
    plt.text(i, valor, str(valor), ha='center', va='bottom')  
  
plt.show()
```

