PASSO A PASSO – SEMANTIX DATA SCIENCE ACADEMY – DATA SCIENCE FUNDAMENTALS

ESSENCIAIS

BIG DATA ECOSYSTEM COM DOCKER-Exercícios Respondidos

Ambiente para estudo dos principais frameworks big data em docker oferecido pela SEMANTIX no seu treinamento. O objetivo manter o registros dos estudos e exercícios para uma consulta no futuro.

PRÉ-REQUISITO

Ter instalado o docker e ubunto(ou outra versão do linux). Neste lab foi utilizado Windows 10 Home Single Language, versão 21H2, compilação do S.O 19044.1526.

Esse setup vai criar dockers com os frameworks HDFS, HBase, Hive, Presto, Spark, Jupyter, Hue, Mongodb, Metabase, Nifi, kafka, Mysql e Zookeeper.

SETUP

OBS: Esse passo deve ser realizado apena uma vez. Após o ambiente criado, utilizar o docker-compose para iniciar os containers como mostrado no tópico INICIANDO O AMBIENTE

Criação do diretório docker:

OBS: A criação do diretório é importante para os mapeamentos necessários

- No Linux:
    - Criar o diretório na home do usuário ex: /home/user/docker

Em um terminal/DOS, dentro diretório docker, realizar o clone do projeto no github

```
git clone https://github.com/rodrigo-reboucas/docker-bigdata.git
```

INICIANDO O AMBIENTE

No terminal, no diretorio bigdata_docker, executar o docker-compose

```
docker-compose up -d
```

Verificar imagens

```
docker image ls
```

Verificar containers

```
docker container ls
```

SOLUCIONANDO PROBLEMAS

Parar um containers

```
docker stop [nome do container]
```

Parar todos containers

docker stop $(docker ps -a -q)

Remover um container

docker rm [nome do container]

Remover todos containers

docker rm $(docker ps -a -q)

Dados do containers

docker container inspect [nome do container]

Iniciar um container

docker-compose up -d [nome do container]

Iniciar todos os containers

docker-compose up -d

Acessar log do container

docker container logs [nome do container]

Acesso WebUI dos Frameworks

- HDFS *http://localhost:50070*
- Presto *http://localhost:8080*
- Hbase *http://localhost:16010/master-status*
- Mongo Express *http://localhost:8081*
- Kafka Manager *http://localhost:9000*
- Metabase *http://localhost:3000*
- Nifi *http://localhost:9090*
- Jupyter Spark *http://localhost:8889*
- Hue *http://localhost:8888*
- Spark *http://localhost:4040*

Acesso por shell

HDFS

docker exec -it datanode bash

HBase

docker exec -it hbase-master bash

Sqoop

docker exec -it datanode bash

Kafka

docker exec -it kafka bash

Acesso JDBC

MySQL

jdbc:mysql://database/employees

Hive

jdbc:hive2://hive-server:10000/default

Presto

jdbc:presto://presto:8080/hive/default

Usuários e senhas

Hue

Usuário: admin

Senha: admin

Metabase

Usuário: bigdata@class.com

Senha: bigdata123

MySQL

Usuário: root

Senha: secret

MongoDB

Usuário: root

Senha: root

Authentication Database: admin

Imagens

Docker Hub

Documentação Oficial

- https://zookeeper.apache.org/
- https://kafka.apache.org/
- https://nifi.apache.org/
- https://prestodb.io/

- https://spark.apache.org/
- https://www.mongodb.com/
- https://www.metabase.com/
- https://jupyter.org/
- https://hbase.apache.org/
- https://sqoop.apache.org/
- https://hadoop.apache.org/
- https://hive.apache.org/
- https://gethue.com/
- https://github.com/yahoo/CMAK
- https://www.docker.com/

Este repositório é um fork do fabiogjardim

AULA 3: HDFS

1. Iniciar o cluster de Big Data

```
cd docker-bigdata
docker-compose up -d
```

2. Baixar os dados dos exercícios do treinamento

```
cd input
sudo git clone https://github.com/rodrigo-reboucas/exercises-data.git
```

3. Acessar o container do namenode

```
docker exec -it namenode bash
```

4. Criar a estrutura de pastas apresentada a baixo pelo comando: $ hdfs dfs -ls -R /

user/aluno/

<nome>

data

recover

delete

```
hdfs dfs -mkdir -p /user/heliton/data/
hdfs dfs -mkdir -p /user/aluno/heliton/recover
hdfs dfs -mkdir -p /user/aluno/heliton/delete
```

5. Enviar a pasta “/input/exercises-data/escola” e o arquivo “/input/exercises-data/entrada1.txt” para data

```
hdfs dfs -put /input/exercises-data/escola/ /user/aluno/heliton/data/
hdfs dfs -put /input/exercises-data/entrada1.txt /user/aluno/heliton/data/
```

6. Mover o arquivo “entrada1.txt” para recover

```
hdfs dfs -mv /user/aluno/heliton/data/entrada1.txt /user/aluno/heliton/recover/
```

7. Baixar o arquivo do hdfs “escola/alunos.json” para o sistema local /

```
hdfs dfs -get /user/aluno/heliton/data/escola/alunos.json /
```

8. Deletar a pasta recover

```
hdfs dfs -rm -r /user/aluno/heliton/recover
```

9. Deletar permanentemente o delete

```
hdfs dfs -rm -r -skipTrash /user/aluno/heliton/delete
```

10. Procurar o arquivo “alunos.csv” dentro do /user

```
hdfs dfs -find / -name alunos.csv
```

11. Mostrar o último 1KB do arquivo “alunos.csv”

```
hdfs dfs -tail /user/aluno/heliton/data/escola/alunos.csv

Outra forma de visualizar com tail
hdfs dfs -cat /user/aluno/heliton/data/escola/alunos.csv | tail -n 10
```

12. Mostrar as 2 primeiras linhas do arquivo “alunos.csv”

```
hdfs dfs -cat /user/aluno/heliton/data/escola/alunos.csv | head -n 2
```

13. Verificação de soma das informações do arquivo “alunos.csv”

```
hdfs dfs -checksum /user/aluno/heliton/data/escola/alunos.csv
/user/aluno/heliton/data/escola/alunos.csv    MD5-of-0MD5-of-512CRC32C        00000200000000000000000164b9235a4d65a1e8ebfe12eb97ac471
```

14. Criar um arquivo em branco com o nome de “test” no data

```
  hdfs dfs -touchz /user/aluno/heliton/data/test
 -rw-r--r--   3 root supergroup          0 2022-05-19 16:54 /user/aluno/helit
on/data/test
```

15. Alterar o fator de replicação do arquivo “test” para 2

```
 hdfs dfs -setrep 2 /user/aluno/heliton/data/test
 -rw-r--r--   2 root supergroup          0 2022-05-19 16:25 /user/heliton/dat
a/test
```

16. Ver as informações do arquivo “alunos.csv”

```
 hdfs dfs -find / -name alunos.csv
 /user/aluno/heliton/data/escola/alunos.csv
 hdfs dfs -help stat
 hdfs dfs -stat /user/aluno/heliton/data/escola/alunos.csv
```

17. Exibir o espaço livre do data e o uso do disco

```
 Espaço livre diretorio data
 hdfs dfs -df -h /user/aluno/heliton/data/
 Filesystem               Size    Used  Available  Use%
 hdfs://namenode:8020  251.0 G  29.7 M    224.0 G    0%
 Uso do disco
 hdfs dfs -du -h /user/aluno/heliton/data/
 29.2 M  /user/aluno/heliton/data/escola
 0       /user/aluno/heliton/data/test
```

AULA 4

Exercício criação da tabela raw

1. Enviar o arquivo local “/input/exercises-data/populacaoLA/populacaoLA.csv” para o diretório no HDFS “/user/aluno/<nome>/data/populacao”

```
 hdfs dfs -mkdir /user/aluno/heliton/data/populacao
 hdfs dfs -put /input/exercises-data/populacaoLA/populacaoLA.csv /user/aluno/
heliton/data/populacao/
```

2. Listar os bancos de dados no Hive

```
 docker exec -it hive-server bash
 beeline --help
 beeline -u jdbc:hive2://localhost:10000

 show databases;
+----------------+
| database_name  |
```

```
+----------------+
| default        |
+----------------+
```

3. Criar o banco de dados <nome>

```
 create database heliton;

 show databases;
+----------------+
| database_name  |
+----------------+
| default        |
| heliton        |
+----------------+
```

4. Criar a Tabela Hive no BD <nome>

Tabela interna: pop
Campos:
zip_code - int
total_population - int
median_age - float
total_males - int
total_females - int
total_households - int
average_household_size - float
Propriedades
Delimitadores: Campo ‘,’ | Linha ‘\n’
Sem Partição
Tipo do arquivo: Texto
tblproperties("skip.header.line.count"="1")’

```
use heliton;

create table pop (
zip_code int,
total_population int,
median_age float,
total_males int,
total_females int,
total_households int,
average_household_size float
)
row format delimited
fields terminated by ','
lines terminated by '\n'
stored as textfile
tblproperties("skip.header.line.count"="1");
```

5. Visualizar a descrição da tabela pop

```
desc pop;
+-------------------------+------------+----------+
|        col_name         | data_type  | comment  |
+-------------------------+------------+----------+
| zip_code                | int        |          |
| total_population        | int        |          |
| median_age              | float      |          |
| total_males             | int        |          |
| total_females           | int        |          |
| total_households        | int        |          |
| average_household_size  | float      |          |
+-------------------------+------------+----------+

desc formatted pop;
+-------------------------------+-----------------------------------------
--------+-----------------------------+
|           col_name            |                    data_typ
e                       |           comment           |
+-------------------------------+-----------------------------------------
--------+-----------------------------+
| # col_name                    | data_typ
e                                        | comment                     |
|                               | NUL
L                                           | NUL
L                         |
| zip_code                      | in
t                                           |
  |
| total_population              | in
t                                           |
  |
| median_age                    | floa
t                                          |
|
| total_males                   | in
t                                           |
  |
| total_females                 | in
t                                           |
  |
| total_households              | in
t                                           |
  |
| average_household_size        | floa
t                                          |
|
|                               | NUL
L                                           | NUL
L                         |
| # Detailed Table Information  | NUL
L                                           | NUL
L                         |
| Database:                     | helito
n                                         | NULL                        |
| Owner:                        | roo
```

```
t                                                  | NUL
L                        |
| CreateTime:                    | Fri May 20 16:38:44 UTC 202
2                       | NULL                           |
| LastAccessTime:                | UNKNOW
N                                          | NULL                            |
| Retention:                     |
0                                              | NUL
L                        |
| Location:                      | hdfs://namenode:8020/user/hive/warehouse/he
liton.db/pop | NULL                          |
| Table Type:                    | MANAGED_TABL
E                                       | NULL                            |
| Table Parameters:              | NUL
L                                          | NUL
L                        |
|                                | COLUMN_STATS_ACCURAT
E                                | {\"BASIC_STATS\":\"true\"}  |
|                                | numFile
s                                          | 0                               |
|                                | numRow
s                                           | 0                               |
|                                | rawDataSiz
e                                          | 0                              |
|                                | skip.header.line.coun
t                                | 1                              |
|                                | totalSiz
e                                           | 0                               |
|                                | transient_lastDdlTim
e                                | 1653064724                   |
|                                | NUL
L                                          | NUL
L                        |
| # Storage Information          | NUL
L                                          | NUL
L                        |
| SerDe Library:                 | org.apache.hadoop.hive.serde2.lazy.LazySimp
leSerDe | NULL                              |
| InputFormat:                   | org.apache.hadoop.mapred.TextInputForma
t           | NULL                               |
| OutputFormat:                  | org.apache.hadoop.hive.ql.io.HiveIgnoreKeyT
extOutputFormat | NULL                           |
| Compressed:                    | N
o                                              | NUL
L                        |
| Num Buckets:                   | -
1                                          | NUL
L                        |
| Bucket Columns:                | [
]                                          | NUL
L                        |
| Sort Columns:                  | [
]                                          | NUL
L                        |
| Storage Desc Params:           | NUL
L                                          | NUL
L                        |
|                                | field.deli
m                                         | ,                              |
```

```
|                                | line.deli
m                                        | \n                          |
|                                | serialization.forma
t                                | ,                             |
+--------------------------------+-----------------------------------------
--------+----------------------------+
```

Exercicio Inserir Dados na Tabela Raw

1.Visualizar a descrição da tabela pop do banco de dados <nome>

```
use heliton;
desc pop;
+-------------------------+------------+----------+
|        col_name         | data_type  | comment  |
+-------------------------+------------+----------+
| zip_code                | int        |          |
| total_population        | int        |          |
| median_age              | float      |          |
| total_males             | int        |          |
| total_females           | int        |          |
| total_households        | int        |          |
| average_household_size  | float      |          |
+-------------------------+------------+----------+

desc formatted pop;
+--------------------------------+-----------------------------------------
--------+----------------------------+
|           col_name             |                     data_typ
e                       |           comment            |
+--------------------------------+-----------------------------------------
--------+----------------------------+
| # col_name                     | data_typ
e                                        | comment                     |
|                                | NUL
L                                            | NUL
L                          |
| zip_code                       | in
t                                             |
  |
| total_population               | in
t                                             |
  |
| median_age                     | floa
t                                           |
|
| total_males                    | in
t                                             |
  |
| total_females                  | in
t                                             |
  |
| total_households               | in
t                                             |
  |
| average_household_size         | floa
t                                           |
```

```
|
|                               | NUL
L                                              | NUL
L                       |
| # Detailed Table Information  | NUL
L                                              | NUL
L                       |
| Database:                     | helito
n                                           | NULL                          |
| Owner:                        | roo
t                                              | NUL
L                       |
| CreateTime:                   | Fri May 20 16:38:44 UTC 202
2                      | NULL                          |
| LastAccessTime:               | UNKNOW
N                                           | NULL                          |
| Retention:                    |
0                                                 | NUL
L                       |
| Location:                     | hdfs://namenode:8020/user/hive/warehouse/he
liton.db/pop | NULL                          |
| Table Type:                   | MANAGED_TABL
E                                     | NULL                          |
| Table Parameters:             | NUL
L                                              | NUL
L                       |
|                               | COLUMN_STATS_ACCURAT
E                              | {\"BASIC_STATS\":\"true\"}  |
|                               | numFile
s                                         | 0                             |
|                               | numRow
s                                           | 0                             |
|                               | rawDataSiz
e                                       | 0                             |
|                               | skip.header.line.coun
t                             | 1                             |
|                               | totalSiz
e                                          | 0                             |
|                               | transient_lastDdlTim
e                              | 1653064724                    |
|                               | NUL
L                                              | NUL
L                       |
| # Storage Information         | NUL
L                                              | NUL
L                       |
| SerDe Library:                | org.apache.hadoop.hive.serde2.lazy.LazySimp
leSerDe | NULL                          |
| InputFormat:                  | org.apache.hadoop.mapred.TextInputForma
t           | NULL                          |
| OutputFormat:                 | org.apache.hadoop.hive.ql.io.HiveIgnoreKeyT
extOutputFormat | NULL                          |
| Compressed:                   | N
o                                                 | NUL
L                       |
| Num Buckets:                  | -
1                                                 | NUL
L                       |
| Bucket Columns:               | [
```

```
]                                                  | NUL
L                        |
| Sort Columns:                  | [
]                                                  | NUL
L                        |
| Storage Desc Params:           | NUL
L                                                | NUL
L                        |
|                                | field.deli
m                                       | ,                              |
|                                | line.deli
m                                        | \n                              |
|                                | serialization.forma
t                                | ,                             |
+--------------------------------+--------------------------------------------
--------+-----------------------------+
```

2.Selecionar os 10 primeiros registros da tabela pop

```
select *
from pop
limit 10;
+---------------+----------------------+-----------------+-----------------
+--------------------+----------------------+-----------------------------+
| pop.zip_code  | pop.total_population  | pop.median_age  | pop.total_male
s  | pop.total_females  | pop.total_households  | pop.average_household_siz
e  |
+---------------+----------------------+-----------------+-----------------
+--------------------+----------------------+-----------------------------+
+---------------+----------------------+-----------------+-----------------
+--------------------+----------------------+-----------------------------+
```

3.Carregar o arquivo do HDFS “/user/aluno/<nome>/data/população/populacaoLA.csv” para a tabela Hive pop

```
load data inpath '/user/aluno/heliton/data/populacao/populacaoLA.csv' overwri
te into table pop;
```

4.Selecionar os 10 primeiros registros da tabela pop

```
select *
from pop
limit 10;
+---------------+----------------------+-----------------+-----------------
+--------------------+----------------------+-----------------------------+
| pop.zip_code  | pop.total_population  | pop.median_age  | pop.total_male
s  | pop.total_females  | pop.total_households  | pop.average_household_siz
e  |
+---------------+----------------------+-----------------+-----------------
+--------------------+----------------------+-----------------------------+
| 91371         | 1                    | 73.5            |
0                | 1                  | 1                    | 1.
0                        |
| 90001         | 57110                | 26.6            | 2846
8            | 28642              | 12971                | 4.
4                        |
```

```
| 90002         | 51223                | 25.5            | 2487
6            | 26347             | 11731                | 4.3
6                      |
| 90003         | 66266                | 26.3            | 3263
1            | 33635             | 15642                | 4.2
2                      |
| 90004         | 62180                | 34.8            | 3130
2            | 30878             | 22547                | 2.7
3                      |
| 90005         | 37681                | 33.9            | 1929
9            | 18382             | 15044                | 2.
5                       |
| 90006         | 59185                | 32.4            | 3025
4            | 28931             | 18617                | 3.1
3                      |
| 90007         | 40920                | 24.0            | 2091
5            | 20005             | 11944                | 3.
0                       |
| 90008         | 32327                | 39.7            | 1447
7            | 17850             | 13841                | 2.3
3                      |
| 90010         | 3800                 | 37.8            | 187
4             | 1926              | 2014                 | 1.8
7                      |
+---------------+----------------------+-----------------+------------------
+--------------------+-----------------------+-----------------------------+
```

5.Contar a quantidade de registros da tabela pop

```
select count(*) as qtd_registros from pop;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the f
uture versions. Consider using a different execution engine (i.e. spark, tez)
or using Hive 1.X releases.
+----------------+
| qtd_registros  |
+----------------+
| 319            |
+----------------+
```

AULA 6

Hive - Seleção de Tabelas

1. Selecionar os 10 primeiros registros da tabela nascimento pelo ano de 2016.

select * from nascimento where ano=2016;

2. Contar a quantidade de nomes de crianças nascidas em 2017

select count(nome) from nascimento where ano=2017;

3. Contar a quantidade de crianças nascidas em 2017

select sum(frequencia) from nascimento where ano=2017;

4. Contar a quantidade de crianças nascidas por sexo no ano de 2015

select count(sexo) from nascimento where ano=2015;

5. Mostrar por ordem de ano decrescente a quantidade de crianças nascidas por sexo

select ano, sexo, sum(frequencia) as qtd_decrescente from nascimento group by ano, sexo order by ano desc;

6. Mostrar por ordem de ano decrescente a quantidade de crianças nascidas por sexo com o nome iniciado com ‘A’

select ano, sexo, sum(frequencia) as qtd_decrescente from nascimento where nome like 'A%' group by ano, sexo order by ano desc;

7. Qual nome e quantidade das 5 crianças mais nascidas em 2016

select nome, max(frequencia) as qnt_mais_nasc from nascimento where ano=2016 group by nome order by qnt_mais_nasc desc limit 5;

8. Qual nome e quantidade das 5 crianças mais nascidas em 2016 do sexo masculino e feminino

select nome, max(frequencia) as qnt_mais_nasc, sexo from nascimento where ano=2016 group by nome, sexo order by qnt_mais_nasc desc limit 5;

AULA 6, ATIVIDADE 2

Hive - Criação de Tabelas Otimizadas

1. Usar o banco de dados <nome>

USE vinicius;

2. Selecionar os 10 primeiros registros da tabela pop

SELECT * FROM pop LIMIT 10;

3. Criar a tabela pop_parquet no formato parquet para ler os dados da tabela pop

```
create table pop_parquet(
zip_code int,
total_population int,
median_age float,
total_males int,
total_females int,
total_households int,
average_household_size float
)
stored as parquet;
```

4. Inserir os dados da tabela pop na pop_parquet

```
insert into pop_parquet (select * from pop);
```

5. Contar os registros da tabela pop_parquet

```
select count(*) from pop;
```

select count(*) from pop_parquet; → muito mais rápido

6. Selecionar os 10 primeiros registros da tabela pop_parquet

```
select * from pop_parquet limit 10;
```

7. Criar a tabela pop_parquet_snappy no formato parquet com compressão Snappy para ler os dados da tabela pop

```
create table pop_parquet_snappy(
zip_code int,
total_population int,
median_age float,
total_males int,
```

total_females int,

total_households int,

average_households_size float

)

stored as parquet

tblproperties('parquet.compress'='SNAPPY');

8. Inserir os dados da tabela pop na pop_parquet_snappy

insert into pop_parquet_snappy (select * from pop);

9. Contar os registros da tabela pop_parquet_snappy

select count(*) from pop_parquet_snappy;

10. Selecionar os 10 primeiros registros da tabela pop_parquet_snappy

select * from pop_parquet_snappy limit 10;

11. Comparar as tabelas pop, pop_parquet e pop_parquet_snappy no HDFS.

Entra no bash do namenode (docker exec –it namenode bash)

hdfs dfs –ls /user/hive/warehouse/vinicius.db →listar

hdfs dfs -du -h /user/hive/warehouse/vinicius.db →conferir tamanho

AULA 7

Sqoop - Pesquisa e Criação de Tabelas

Todos os comandos precisam ser executados pelo Sqoop.

1. Mostrar todos os databases

sqoop list-databases --connect jdbc:mysql://database --username root --password secret

2. Mostrar todas as tabelas do bd employees

sqoop list-tables --connect jdbc:mysql://database/employees --user

name root --password secret

3. Inserir os valores ('d010', 'BI') na tabela departments do bd employees

sqoop eval --connect jdbc:mysql://database/employees --user

name root --password secret –query (inserir query aqui, deu preguiça pq qnd vou colar o –connect... o meu faz a pesquisa direto)

4. Pesquisar todos os registros da tabela departments

5. Criar a tabela benefits(cod int(2) AUTO_INCREMENT PRIMARY KEY, name varchar(30)) no bd employees

6. Inserir os valores (null,'food vale') na tabela benefits

7. Pesquisar todos os registros da tabela benefits

AULA 8

Sqoop - Importação BD Employees

1. Pesquisar os 10 primeiros registros da tabela employees do banco de dados employees

sqoop eval --connect jdbc:mysql://database/employees --username root --password secret --query "select * from employees limit 10"

2. Realizar as importações referentes a tabela employees e para validar cada questão, é necessário visualizar no HDFS*

A. Importar a tabela employees, no warehouse /user/hive/warehouse/db_test_a

   sqoop import --table employees --connect jdbc:mysql://database/employees --username root --password secret --warehouse-dir /user/hive/warehouse/db_test_a

hdfs dfs -cat /user/hive/warehouse/db_test_a/employees/part-m-00000 | head -n 3

hdfs dfs -cat /user/hive/warehouse/db_test_a/employees/part-m-00001 | head -n 3

hdfs dfs -cat /user/hive/warehouse/db_test_a/employees/part-m-00002 | head -n 3

hdfs dfs -cat /user/hive/warehouse/db_test_a/employees/part-m-00003 | head -n 3

B. Importar todos os funcionários do gênero masculino, no warehouse /user/hive/warehouse/db_test_b

sqoop import --table employees --connect jdbc:mysql://database/employees --username root --password secret --where "gender='M'" --warehouse-dir /user/hive/warehouse/db_test_b

C. importar o primeiro e o último nome dos funcionários com os campos separados por tabulação, no warehouse /user/hive/warehouse/db_test_c

sqoop import --table employees --connect jdbc:mysql://database/employees --username root --password secret --columns "first_name, last_name" --fields-terminated-by '\t' --warehouse-dir /user/hive/warehouse/db_test_c

D. Importar o primeiro e o último nome dos funcionários com as linhas separadas por “ : ” e salvar no mesmo diretório da questão 2.C

sqoop import --table employees --connect jdbc:mysql://database/employees --username root --password secret --columns "first_name, last_name" --lines-terminated-by ':' --warehouse-dir /user/hive/warehouse/db_test_c --delete-target-dir

* Dica para visualizar no HDFS:

$ hdfs dfs -cat /.../db_test/nomeTabela/part-m-00000 | head -n 5

3. Clicar no botão de **Enviar Tarefa**, e enviar o texto: **ok**

AULA 8

Sqoop - Importação BD Employees- Otimização

Realizar com uso do **MySQL**

1. Criar a tabela cp_titles_date, contendo a cópia da tabela titles com os campos title e to_date

create table cp_titles_date (select title, to_date from titles);

2. Pesquisar os 15 primeiros registros da tabela cp_titles_date

select * from cp_titles_date limit 15;

3. Alterar os registros do campo to_date para null da tabela cp_titles_date, quando o título for igual a Staff

update cp_titles_date set to_date=NULL where title='Staff';

Realizar com uso do **Sqoop** - Importações no warehouse **/user/hive/warehouse/db_test_<numero_questao>** e visualizar no HDFS

4. Importar a tabela titles com 8 mapeadores no formato parquet

sqoop import --table titles --connect jdbc:mysql://database/employees --username root --password secret -m 8 --as-parquetfile --warehouse-dir /user/hive/warehouse/db_test2_4

5. Importar a tabela titles com 8 mapeadores no formato parquet e compressão snappy

sqoop import --table titles --connect jdbc:mysql://database/employees --username root --password secret -m 8 --as-parquetfile --warehouse-dir /user/hive/warehouse/db_test2_5 --compress --compression-codec org.apache.hadoop.io.compress.SnappyCodec

6. Importar a tabela cp_titles_date com 4 mapeadores (erro)

sqoop import --table cp_titles_date --connect jdbc:mysql://database/employees --username root --password secret -m 4 --warehouse-dir /user/hive/warehouse/db_test2_6

A. Importar a tabela cp_titles_date com 4 mapeadores divididos pelo campo título no warehouse **/user/hive/warehouse/db_test2_title**

   sqoop import -Dorg.apache.sqoop.splitter.allow_text_splitter=true --table cp_titles_date --connect jdbc:mysql://database/employees --username root --password secret -m 4 --warehouse-dir /user/hive/warehouse/db_test2_title --split-by title

B. Importar a tabela cp_titles_date com 4 mapeadores divididos pelo campo data no warehouse **/user/hive/warehouse/db_test2_date**

sqoop import --table cp_titles_date --connect jdbc:mysql://database/employees --username root --password secret -m 4 --warehouse-dir /user/hive/warehouse/db_test2_date –split-by to_date

C. Qual a diferença dos registros nulos entre as duas importações?

FODASE

AULA 9, E1

Sqoop - Importação BD Sakila – Carga Incremental

Realizar com uso do **MySQL**

1. Criar a tabela cp_rental_append, contendo a cópia da tabela rental com os campos rental_id e rental_date

create table cp_rental_append select rental_id, rental_date from rental;

2 .Criar a tabela cp_rental_id e cp_rental_date, contendo a cópia da tabela cp_rental_append

create table cp_rental_id select * from cp_rental_append;

create table cp_rental_date select * from cp_rental_append;

Realizar com uso do **Sqoop** - Importações no warehouse /user/hive/warehouse/db_test3 e visualizar no HDFS

3. Importar as tabelas cp_rental_append, cp_rental_id e cp_rental_date com 1 mapeador

sqoop import --connect jdbc:mysql://database/sakila --username root -password secret --warehouse-dir /user/hive/warehouse/db_test3 -m 1 --table cp_rental_append

sqoop import --connect jdbc:mysql://database/sakila --username root -password secret --warehouse-dir /user/hive/warehouse/db_test3 -m 1 --table cp_rental_id

sqoop import --connect jdbc:mysql://database/sakila --username root -password secret --warehouse-dir /user/hive/warehouse/db_test3 -m 1 --table cp_rental_date

Realizar com uso do **MySQL**

4. Executar o sql /db-sql/sakila/insert_rental.sql no container do database

$ docker exec -it database bash

$ cd /db-sql/sakila

$ mysql -psecret < insert_rental.sql

Realizar com uso do **Sqoop** - Importações no warehouse /user/hive/warehouse/db_test3 e visualizar no HDFS

hdfs dfs -ls -R /user/hive/warehouse/db_test3

mysql -psecret < insert_rental.sql

5. Atualizar a tabela cp_rental_append no HDFS anexando os novos arquivos

sqoop eval --connect jdbc:mysql://database/sakila --username root -password secret --query "select *from cp_rental_append order by rental_id desc limit 5"

6. Atualizar a tabela cp_rental_id no HDFS de acordo com o último registro de rental_id, adicionando apenas os novos dados.

7. Atualizar a tabela cp_rental_date no HDFS de acordo com o último registro de rental_date, atualizando os registros a partir desta data.

AULA 9, E2

Sqoop - Importação para o Hive e Exportação - BD Employees

1.Importar a tabela employees.titles do MySQL para o diretório /user/aluno/<nome>/data com 1 mapeador.

sqoop import --table titles --connect jdbc:mysql://database/employees --username root --password secret --warehouse-dir /user/alunos/vinicius/data -m 1

2. Importar a tabela employees.titles do MySQL para uma **tabela Hive** no banco de dados seu nome com 1 mapeador.

sqoop import --table titles --connect jdbc:mysql://database/employees --username root --password secret -m 1 --hive-import --hive-table vinicius.titles

hdfs dfs -ls /user/hive/warehouse/vinicius.db

3. Selecionar os 10 primeiros registros da tabela titles no **Hive**.

No bash do hive:

use vinicius;

select * from titles limit 10;

4. Deletar os registros da tabela employees.titles do MySQL e verificar se foram apagados, através do Sqoop

sqoop eval --connect jdbc:mysql://database/employees --username root --password secret --query "select * from titles limit 10"

sqoop eval --connect jdbc:mysql://database/employees --username root --password secret --query "truncate table titles"

sqoop eval --connect jdbc:mysql://database/employees --username root --password secret --query "select * from titles limit 10"

5. Exportar os dados do diretório /user/hive/warehouse/<nome>.db/data/titles para a tabela do MySQL employees.titles.

sqoop export --table titles --connect jdbc:mysql://database/employees --username root --password secret --export-dir /user/aluno/vinicius/data/titles

6. Selecionar os 10 primeiros registros registros da tabela employees.titles do MySQL.

sqoop eval --connect jdbc:mysql://database/employees --username root --password secret --query "select * from titles limit 10"

AULA 11

Spark - Exercícios de DataFrame

1. Enviar o diretório local “/input/exercises-data/juros_selic” para o HDFS em “/user/aluno/<nome>/data”

root@namenode:/#

root@namenode:/# hdfs dfs -put /input/exercises-data/juros_selic/ /user/aluno/vinicius/data

root@namenode:/# hdfs dfs -ls /user/aluno/vinicius/data

2. Criar o DataFrame jurosDF para ler o arquivo no HDFS “/user/aluno/<nome>/data/juros_selic/juros_selic.json”

vinysiq@LAPTOP-27CLHU1L:~/treinamento-bigdata/docker-bigdata$ docker exec -it spark bash

root@spark:/# spark-shell

scala> val jurosDF = spark.read.json("/user/aluno/vinicius/data/juros_selic/juros_selic.json")

3. Visualizar o Schema do jurosDF

scala> jurosDF.printSchema()

4. Mostrar os 5 primeiros registros do jutosDF

scala> jurosDF.show(5)

scala> jurosDF.show(5, false)

5. Contar a quantidade de registros do jurosDF

scala> jurosDF.count()

6. Criar o DataFrame jurosDF10 para filtrar apenas os registros com o campo “valor” maior que 10

scala> val jurosDF10 = jurosDF.where("valor > 10")

7. Salvar o DataFrame jurosDF10 como tabela Hive “<nome>.tab_juros_selic”

scala> jurosDF10.write.saveAsTable("vinicius.tab_juros_selic")

8. Criar o DataFrame jurosHiveDF para ler a tabela “<nome>.tab_juros_selic”

scala> val jurosHiveDF = spark.read.table("vinicius.tab_juros_selic")

9. Visualizar o Schema do jurosHiveDF

scala> jurosHiveDF.printSchema

10. Mostrar os 5 primeiros registros do jurosHiveDF

scala> jurosHiveDF.show(5)

11. Salvar o DataFrame jurosHiveDF no HDFS no diretório “/user/aluno/nome/data/save_juros” no formato parquet

scala> jurosHiveDF.write.save("/user/aluno/vinicius/data/save_juros")

12. Visualizar o save_juros no HDFS

vinysiq@LAPTOP-27CLHU1L:~/treinamento-bigdata/docker-bigdata$ docker exec -it namenode bash

root@namenode:/# hdfs dfs -ls /user/aluno/vinicius/data/save_juros

13. Criar o DataFrame jurosHDFS para ler o diretório do “save_juros” da questão 8

vinysiq@LAPTOP-27CLHU1L:~/treinamento-bigdata/docker-bigdata$ docker exec -it spark bash

root@spark:/# spark-shell

scala> val jurosHDFS = spark.read.load("/user/aluno/vinicius/data/save_juros")

14. Visualizar o Schema do jurosHDFS

scala> jurosHDFS.printSchema

15. Mostrar os 5 primeiros registros do jurosHDFS

scala> jurosHDFS.show(5)

AULA 12

Spark - Exercícios de Esquema e Join

1. Criar o DataFrame **alunosDF** para ler o arquivo no hdfs “/user/aluno/<nome>/data/escola/alunos.csv” sem usar as “option”

scala> val alunosDF = spark.read.csv("/user/aluno/vinicius/data/escola/alunos.csv")

2. Visualizar o esquema do alunosDF

alunosDF.printSchema

3. Criar o DataFrame **alunosDF** para ler o arquivo “/user/aluno/<nome>/data/escola/alunos.csv” com a opção de Incluir o cabeçalho

scala> val alunosDF = spark.read.option("header","true").csv("/user/aluno/vinicius/data/escola/alunos.csv")

4. Visualizar o esquema do alunosDF

alunosDF.printSchema

5. Criar o DataFrame **alunosDF** para ler o arquivo “/user/aluno/<nome>/data/escola/alunos.csv” com a opção de Incluir o cabeçalho e inferir o esquema

scala> val alunosDF = spark.read.option("header","true").option("inferSchema","true").csv("/user/aluno/vinicius/data/escola/alunos.csv")

6. Visualizar o esquema do alunosDF

alunosDF.printSchema

7. Salvar o DaraFrame alunosDF como tabela Hive “tab_alunos” no banco de dados <nome>

scala> alunosDF.write.saveAsTable("vinicius.tab_alunos")

8. Criar o DataFrame **cursosDF** para ler o arquivo “/user/aluno/<nome>/data/escola/cursos.csv” com a opção de Incluir o cabeçalho e inferir o esquema

scala> val cursosDF = spark.read.option("header","true").option("inferSchema","true").csv("/user/aluno/vinicius/data/escola/cursos.csv")

9. Criar o DataFrame **alunos_cursosDF** com o inner join do alunosDF e cursosDF quando o id_curso dos 2 forem o mesmo

scala> val aluno_cursosDF = alunosDF.join(cursosDF, "id_curso")

10. Visualizar os dados, o esquema e a quantidade de registros do alunos_cursosDF

scala> aluno_cursosDF.show(10)

Spark - Exercícios da API Catalog

Realizar os exercícios usando a API Catalog.

1. Visualizar todos os banco de dados

spark.catalog.listDatabases.show

2. Definir o banco de dados “seu-nome” como principal

spark.catalog.setCurrentDatabase("vinicius")

3. Visualizar todas as tabelas do banco de dados “seu-nome”

spark.catalog.listTables.show

4. Visualizar as colunas da tabela tab_alunos

scala> spark.catalog.listColumns("tab_alunos").show

5. Visualizar os 10 primeiros registos da tabela "tab_alunos" com uso do spark.sql

scala> spark.sql("select * from tab_alunos limit 10").show()

scala> spark.read.table("tab_alunos").show(10)

Spark - Exercícios de SQL Queries vs Operações de DataFrame

Realizar as seguintes consultas usando SQL queries e transformações de DataFrame na tabela “tab_alunos” no banco de dados <nome>

1. Visualizar o id e nome dos 5 primeiros registros

scala> spark.catalog.setCurrentDatabase("vinicius")

scala> spark.catalog.listColumns("tab_alunos").show

2. Visualizar o id, nome e ano quando o ano de ingresso for maior ou igual a 2018

scala> spark.sql("select id_discente, nome from tab_alunos limit 5").show(false)

scala> spark.catalog.listColumns("tab_alunos").show(false)

3. Visualizar por ordem alfabética do nome o id, nome e ano quando o ano de ingresso for maior ou igual a 2018

scala> val alunosHiveDF = spark.read.table("tab_alunos")

scala> spark.sql("select id_discente,nome,ano_ingresso from tab_alunos where ano_ingresso >= 2018").show(false)

scala> alunosHiveDF.select("id_discente","nome","ano_ingresso").where("ano_ingresso >= 2018").show(false)

scala> spark.sql("select id_discente,nome,ano_ingresso from tab_alunos where ano_ingresso >= 2018 order by nome desc").show(false)

scala>alunosHiveDF.select("id_discente","nome","ano_ingresso").where("ano_ingresso >= 2018").orderBy($"nome".desc).show(false)

4. Contar a quantidade de registros do item anterior

```
scala> spark.sql("select count(id_discente) from tab_alunos where ano_ingresso >= 2018").show
```

```
scala> alunosHiveDF.select("id_discente","nome","ano_ingresso").where("ano_ingresso >= 2018").orderBy($"nome".desc).count()
```