

Título: Relatório instalação e configuração ambiente cluster Spark

Aluno: Rafael De Pauli Baptista

Disciplina: Introdução a Big Data

Relatório detalhando a instalação e configuração de um cluster Spark. Esse cluster é constituído de dois workers, sendo eles:

- 1 - Máquina virtual no ambiente VirtualBox preparada no exercício 1 da matéria;
- 2 - WSL (Windows Subsystem for Linux).

Recursos utilizados:

- Máquina virtual no ambiente VirtualBox preparada no exercício 1 da matéria;
- [spark-3.5.6-bin-hadoop3](#)

Procedimento:

1. Instalação do Spark 3.5.6

No exercício 1 da matéria, foi preparado um ambiente com a versão Java 1.8 e Hadoop 3.3.1. Devido a isso, foi utilizado a versão Spark 3.5.6 para manter a compatibilidade com o ambiente já preparado. A versão Spark 4.0.0 possui a dependência da versão Java 17 ou posterior.

1.1. Configuração do Hadoop 3.3.1 na na máquina virtual no ambiente VirtualBox

Saindo da premissa que o Hadoop 3.3.1 foi pré-instalado na máquina virtual do ambiente VirtualBox, realizar os seguintes ajustes:

- Editar arquivo **\$HADOOP_HOME/etc/hadoop/core-site.xml**:

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://192.168.0.13:9000</value>
  </property>
</configuration>
```

- Editar arquivo **\$HADOOP_HOME/etc/hadoop/hdfs-site.xml**:

```
<configuration>
  <property>
    <name>dfs.replication</name>
```

```

        <value>1</value>
    </property>
</property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/usr/local/hadoop/hadoop_data/hdfs/namenode</value>
</property>
</property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/usr/local/hadoop/hadoop_data/hdfs/datanode</value>
</property>

<!-- Trecho abaixo inserido para funcionar em cluster -->
<property>
    <name>dfs.datanode.ipc.address</name>
    <value>192.168.0.13:9867</value> </property>
<property>
    <name>dfs.datanode.http.address</name>
    <value>192.168.0.13:9864</value>
</property>
<property>
    <name>dfs.datanode.address</name>
    <value>192.168.0.13:9866</value>
</property>
</configuration>

```

- Editar arquivo **\$HADOOP_HOME/etc/hadoop/yarn-site.xml**:

```

<configuration>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>192.168.0.13</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
  <property>
    <name>yarn.nodemanager.env-whitelist</name>

    <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PRE
    PEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_HOME,PATH,LANG,TZ,HADOOP_MAPRED_HOME</value>
  >
  </property>
</configuration>

```

Obs: O IP da máquina configurada é o 192.168.0.13 (máquina virtual do ambiente VirtualBox). Esse IP deve ser alterado conforme o IP da máquina virtual do ambiente VirtualBox.

Essas configurações darão embasamento para a correta instalação e execução do Spark 3.5.6 no modo cluster.

1.2. Instalação Spark Master e Worker na máquina virtual no ambiente VirtualBox

- Realizar download do spark-3.5.6-bin-hadoop3;
- Copiar o arquivo para o diretório `$ sudo cp spark-3.5.6-bin-hadoop3.tar.gz /usr/local`;
- Acessar diretório `$ cd /usr/local` e descompactar o arquivo copiado `$ sudo tar -czfv spark-3.5.6-bin-hadoop3.tar.gz`;
- Editar arquivo `~/.bashrc` e adicionar as seguintes linhas:
 - `export SPARK_HOME=/usr/local/spark-3.5.6-bin-hadoop3/`
 - `export PATH=$PATH:$SPARK_HOME/bin`
- Utilizar script abaixo para iniciar Spark Connector. Isso nos permitirá acessar remotamente o cluster Spark a partir de qualquer computador:

```
#!/bin/bash

# Este script inicia o Spark Connect Server com os pacotes e configurações
corretas.

# Defina o caminho base da sua instalação do Spark
# Certifique-se de que este caminho está correto para a sua VM
SPARK_HOME="/usr/local/spark-3.5.6-bin-hadoop3"

# Defina o IP do seu Spark Master (o mesmo que você está usando para a UI e para
conectar via PySpark)
SPARK_MASTER_IP="bigdata-hadoop"

# Porta padrão para o Spark Master
SPARK_MASTER_PORT="7077"

# Pacote do Spark Connect (verifique a versão exata se houver problemas)
# Geralmente, a versão do spark-connect_2.12 é a mesma da sua versão principal do
Spark (ex: 3.5.x)
SPARK_CONNECT_PACKAGE="org.apache.spark:spark-connect_2.12:3.5.1"
# Se 3.5.1 não funcionar, tente 3.5.0 ou a versão exata do seu Spark (3.5.6)
# SPARK_CONNECT_PACKAGE="org.apache.spark:spark-connect_2.12:3.5.6"

echo "Iniciando o Spark Connect Server..."
echo "SPARK_HOME: $SPARK_HOME"
echo "Spark Master: spark://$SPARK_MASTER_IP:$SPARK_MASTER_PORT"
echo "Spark Connect Package: $SPARK_CONNECT_PACKAGE"

# Executa o script start-connect-server.sh com os parâmetros necessários
"$SPARK_HOME"/sbin/start-connect-server.sh \
  --packages "$SPARK_CONNECT_PACKAGE" \
  --master "spark://$SPARK_MASTER_IP:$SPARK_MASTER_PORT"
```


```
echo "Verifique o log para o status de inicialização do Spark Connect Server."
echo "O servidor deve estar escutando na porta 15002."
echo "Você pode verificar com: sudo netstat -tulnp | grep 15002"
```

- Iniciar ambiente Hadoop conforme descrito nas instruções do exercício 1 da matéria;
- Iniciar ambiente Spark Cluster:
 - /usr/local/spark-3.5.6-bin-hadoop3/sbin/start-all.sh;
 - Iniciar Spark Connector;
- Verificar execução do ambiente Spark Cluster:
 - Executar comando `$ jps` e verificar os processos Hadoop e Spark;

```
bigdata@bigdata-hadoop:~/Desktop$ jps
3760 DataNode
5169 Jps
4306 NodeManager
4899 SparkSubmit
4534 Master
4200 ResourceManager
4392 JobHistoryServer
3928 SecondaryNameNode
5065 CoarseGrainedExecutorBackend
3657 NameNode
4828 Worker
```

- Verificar execução do Master, Worker e Spark Connector usando o UI do Spark:

← → 🔍 Não seguro 192.168.0.13:8080

 **Spark Master at spark://0.0.0.0:7077**

URL: spark://0.0.0.0:7077

Alive Workers: 1

Cores in use: 6 Total, 6 Used

Memory in use: 9.7 GiB Total, 1024.0 MiB Used

Resources in use:

Applications: 1 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

~ Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20250709173848-192.168.0.13-45393	192.168.0.13:45393	ALIVE	6 (6 Used)	9.7 GiB (1024.0 MiB Used)	

~ Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20250709173857-0000	(kill) <u>Spark Connect server</u>	6	1024.0 MiB		2025/07/09 17:38:57	bigdata	RUNNING	8,1 min

~ Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

1.3. Instalação Spark Worker no WSL (Windows Subsystem for Linux)

Estamos saindo da premissa que o WSL (Windows Subsystem for Linux) foi pré-instalado na máquina hospedeira que roda o ambiente VirtualBox. **Obs:** O IP da máquina virtual do ambiente VirtualBox configurada é o 192.168.0.13. Esse IP deve ser alterado conforme o IP da máquina virtual do ambiente VirtualBox detalhado no item **1.2**.

- Acessar diretório `$ cd /usr/local;`
- Copiar Hadoop configurado no passo 1.2 para o WSL `$sudo scp -r bigdata@192.168.0.13:/usr/local/hadoop ./`
- Copiar Spark configurado no passo 1.2 para o WSL `$sudo scp -r bigdata@192.168.0.13:/usr/local/spark-3.5.6-bin-hadoop3 ./`
- Acrescentar as seguintes configurações no arquivo `~/bashrc`:

```
export JAVA_HOME="/usr/lib/jvm/java-11-openjdk-amd64"
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
unset JAVA_TOOL_OPTIONS
export PDSH_RCMD_TYPE=ssh

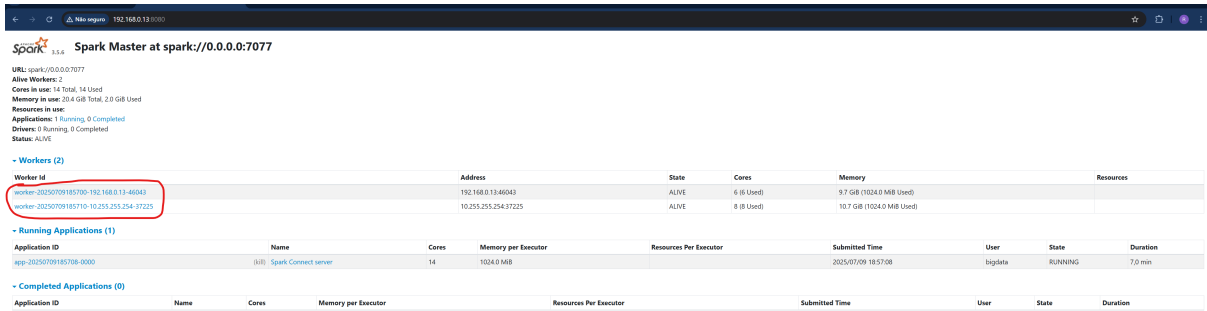
# Config Spark
export SPARK_HOME=/usr/local/spark-3.5.6-bin-hadoop3/
export PATH=$PATH:$SPARK_HOME/bin
export PATH=$PATH:$SPARK_HOME/sbin
```

- **Obs:** Ajustar caminho da variável **JAVA_HOME** conforme o caminho da sua versão Java instalada no WSL. Vale lembrar que o ambiente preparado só funciona com a versão Java 1.8 ou 11.
- Testar conectividade hadoop `$ hdfs dfs -ls /`

```
airflow@LAPTOP-88ATPCMH: $ hdfs dfs -ls /
2025-07-09 18:59:01,131 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r-- 1 bigdata supergroup 16822301 2025-07-07 19:54 /employees_hadoop.csv
drwxrwx--- 0 bigdata supergroup 0 2025-06-28 16:28 /tmp
airflow@LAPTOP-88ATPCMH: $
```

- Iniciar Spark Worker `$ start-worker.sh spark://192.168.0.13:7077`
 - Endereço IP informado é do Spark Master configurado no item **1.2**.
 - Caso o WSL não consiga se conectar ao Spark Master, reiniciar interface de rede do WSL usando os comando que devem ser executados no prompt do **Windows**:
 - `DISM /Online /Enable-Feature /All /FeatureName:Microsoft-Windows-Subsystem-Linux`
 - `DISM /Online /Enable-Feature /All /FeatureName:VirtualMachinePlatform`

- Verificar Spark Cluster possui 2 Workers usando o UI do Spark:



Spark Master at spark://0.0.0.0:7077

URL: spark://0.0.0.0:7077

Alive Workers: 2

Cores in use: 14 Total: 14 Used

Memory in use: 20.4 GB Total: 2.0 GB Used

Resources in use:

Applications: 1 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers (2)

Worker id	Address	State	Cores	Memory	Resources
worker-20250709185700-192.168.0.13-4043	192.168.0.13:4043	ALIVE	6 (6 Used)	9.7 GB (1024.0 MB Used)	
worker-20250709185710-10.255.255.254-37225	10.255.255.254:37225	ALIVE	8 (8 Used)	10.7 GB (1024.0 MB Used)	

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20250709185708-0000	(S-R) Spark Connect server	14	1024.0 MB		2025/07/09 18:57:08	logdata	RUNNING	7.0 min

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

1.4. Execução dos exercícios propostos

- Word Count Job em shell Scala

```
bigdata@bigdata-hadoop:~$ spark-shell
25/07/09 19:11:45 WARN Utils: Your hostname, bigdata-hadoop resolves to a loopback address: 127.0.1.1; using 192.168.0.13 instead (on interface enp0s3)
25/07/09 19:11:45 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/07/09 19:12:02 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
25/07/09 19:12:05 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
Spark context Web UI available at http://192.168.0.13:4041
Spark context available as 'sc' (master = local[*], app id = local-1752099125353).
Spark session available as 'spark'.
Welcome to

  ____
 /  __ \
/   /  \
/_____/    version 3.5.6

Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 1.8.0_452)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val texto = sc.textFile("/home/bigdata/pg2701.txt")
texto: org.apache.spark.rdd.RDD[String] = /home/bigdata/pg2701.txt MapPartitionsRDD[1] at textFile at <console>:23

scala> val palavras = texto.flatMap(line => line.split(" "))
palavras: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <console>:23

scala> val contadores = palavras.map(palavra => (palavra,1)).reduceByKey(_ + _)
contadores: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:23

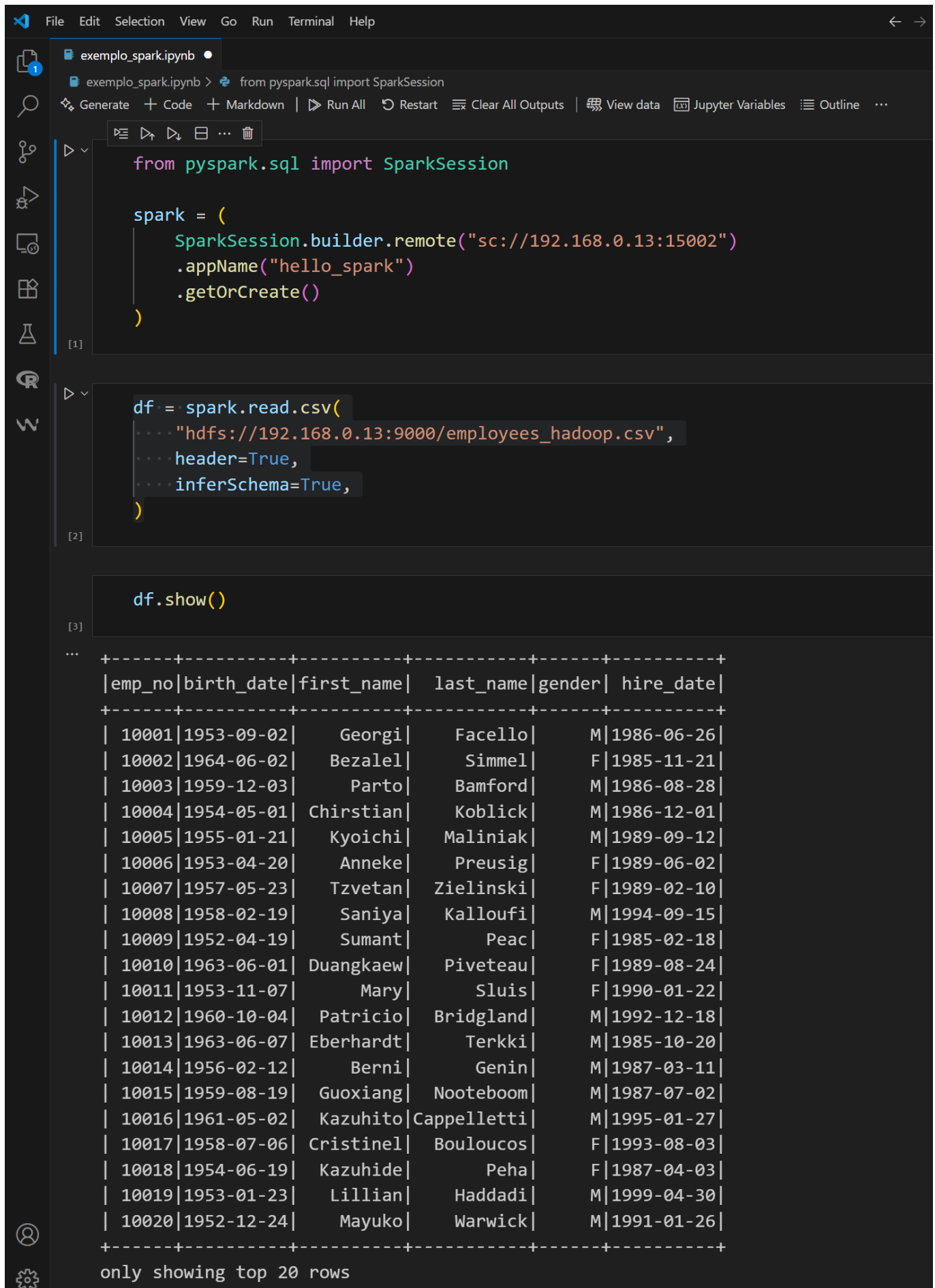
scala> contadores.saveAsTextFile("/home/bigdata/result")

scala> |
```

- Consultas sobre o CSV de Employees em PySpark

```
>>> df.createOrReplaceTempView("employees")
>>> spark.sql("select * from employees where emp_no = 10002").show()
+-----+-----+-----+-----+-----+-----+
|emp_no|birth_date|first_name|last_name|gender|hire_date|
+-----+-----+-----+-----+-----+-----+
| 10002|1964-06-02|Bezalel|Simmel|F|1985-11-21|
+-----+-----+-----+-----+-----+-----+
```

- 7 / 9



The screenshot shows a Jupyter Notebook with three cells. The first cell imports SparkSession. The second cell creates a SparkSession with a remote connection and app name. The third cell reads a CSV file and displays the first 20 rows of the resulting DataFrame.

```
from pyspark.sql import SparkSession

spark = (
    SparkSession.builder.remote("sc://192.168.0.13:15002")
    .appName("hello_spark")
    .getOrCreate()
)
```

```
df = spark.read.csv(
    "hdfs://192.168.0.13:9000/employees_hadoop.csv",
    header=True,
    inferSchema=True,
)
```

```
df.show()
```

emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
10006	1953-04-20	Anneke	Preusig	F	1989-06-02
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
10009	1952-04-19	Sumant	Peac	F	1985-02-18
10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24
10011	1953-11-07	Mary	Sluis	F	1990-01-22
10012	1960-10-04	Patricio	Bridgland	M	1992-12-18
10013	1963-06-07	Eberhardt	Terkki	M	1985-10-20
10014	1956-02-12	Berni	Genin	M	1987-03-11
10015	1959-08-19	Guoxiang	Nooteboom	M	1987-07-02
10016	1961-05-02	Kazuhito	Cappelletti	M	1995-01-27
10017	1958-07-06	Cristinel	Bouloucos	F	1993-08-03
10018	1954-06-19	Kazuhide	Peha	F	1987-04-03
10019	1953-01-23	Lillian	Haddadi	M	1999-04-30
10020	1952-12-24	Mayuko	Warwick	M	1991-01-26

only showing top 20 rows

File Edit Selection View Go Run Terminal Help

exemplo_spark.ipynb

exemplo_spark.ipynb > from pyspark.sql import SparkSession

Generate + Code + Markdown | ▶ Run All ↺ Restart ≡ Clear All Outputs | 🔍 View data 📄 Jupyter Variables 📖 Outline ...

from pyspark.sql.functions import max

df.select(max("hire_date")).show()

[4]

... +-----+
|max(hire_date)|
+-----+
| 2000-01-28 |
+-----+

df.filter((df.emp_no < 10020) & (df.gender == "M") & (df.first_name.startswith("G"))).show()

[5]

... +-----+-----+-----+-----+-----+-----+
|emp_no|birth_date|first_name|last_name|gender| hire_date|
+-----+-----+-----+-----+-----+-----+
| 10001|1953-09-02| Georgi| Facello| M|1986-06-26|
| 10015|1959-08-19| Guoxiang|Nooteboom| M|1987-07-02|
+-----+-----+-----+-----+-----+-----+