

PYSPARK

Grimaldo Oliveira

O que é o PYSPARK

1. É denominada como uma biblioteca Spark que possui a linguagem Python para executar o aplicativo Python usando recursos Apache Spark.
2. PySpark é uma API Python para Apache Spark.
3. Apache Spark é um mecanismo de processamento analítico para aplicações de processamento de dados distribuídos em larga escala e aprendizado de máquina.
4. Detalhe importante é que para executar o PySpark você também precisa que o Java seja instalado junto com Python.

Sites Importantes

Anaconda

<https://www.anaconda.com/products/individual>

Spark

<https://spark.apache.org/downloads.html>

Java JDK

<https://www.oracle.com/java/technologies/javase-downloads.html>

Ecosystem

Para que o Pyspark funcione
precisaremos instalar alguns
softwares: Apache Spark + Anaconda
+ JAVA.

APACHE SPARK



ANACONDA



JAVA

PYSPARK

Entendo melhor quais as vantagens de trabalhar com Pyspark.

PYSPARK

Lançado com o intuito de fornecer a colaboração entre Apache Spark e Python, onde fora desenvolvido uma API Python para Spark.

O uso da biblioteca Pyspark possui diversas vantagens:

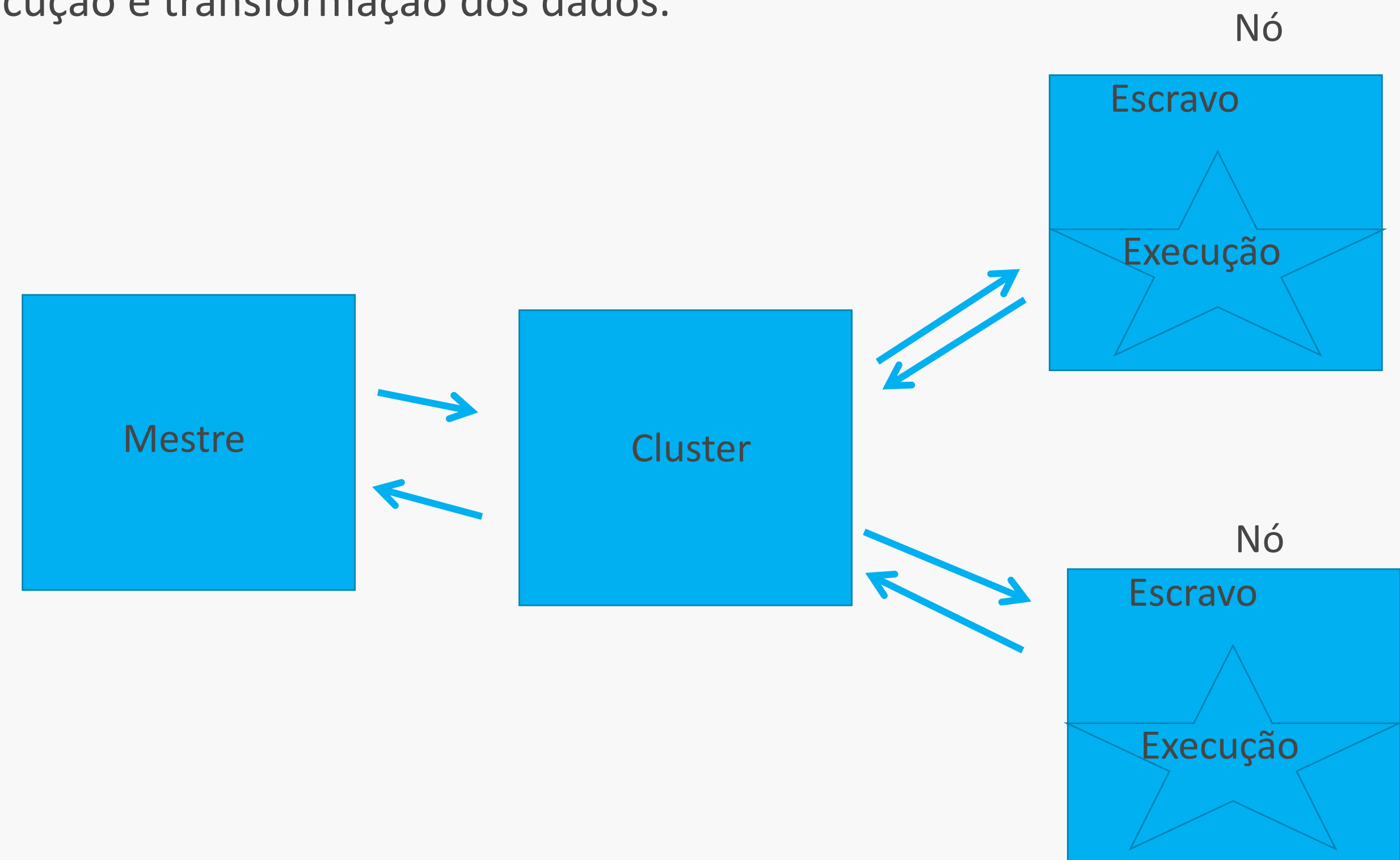
- É um mecanismo de processamento distribuído , na memória, que permite o processamento de dados de forma eficiente e de características distribuída.
- Com o uso do PySpark, é possível o processamento de dados em Hadoop (HDFS), AWS S3 e outros sistemas de arquivos.
- Possui bibliotecas de aprendizado de máquina e gráficos.
- Geralmente as aplicações criadas e executadas no PySpark são 100x mais rápidas que outras em sistemas de dados conhecidos.

Arquitetura do PYSPARK

Como funciona o Pyspark.

PYSPARK

Toda a execução dos scripts são realizados dentro do Apache Spark (arquitetura mestre-escravo), que distribui o processamento dentro de um ambiente de cluster que são interligados aos nós que realizam a execução e transformação dos dados.



SPARK

Faz todo o gerenciamento e distribuição do processamento.

Apache Spark

Apache Spark é o principal **mecanismo** de análise unificado para Big Data e aprendizado de máquina que existe no mundo, sendo utilizado pelas grandes corporações. Explorando nas suas execuções o uso de memória e outras otimizações. Anteriormente as empresas utilizavam o Hadoop.



Cluster

É onde os recursos para operacionalização são distribuídos.

Cluster

É um conjunto de recursos e configurações em que você cria os seus projetos, dentro dos chamados notebooks, é possível dentro de um cluster executar cargas de trabalho e analisar seus dados.



Módulos e pacotes PYSPARK

O que compõem o Pyspark?

PYSPARK

Vamos entender quais são os módulos que existem no Pyspark

- PySpark RDD
- PySpark DataFrame and SQL
- PySpark Streaming
- PySpark MLib
- PySpark GraphFrames
- PySpark Resource

Instalação

Para que o Pyspark funcione
precisaremos instalar alguns
softwares: Apache Spark + Anaconda
+ JAVA.

APACHE SPARK



ANACONDA



JAVA

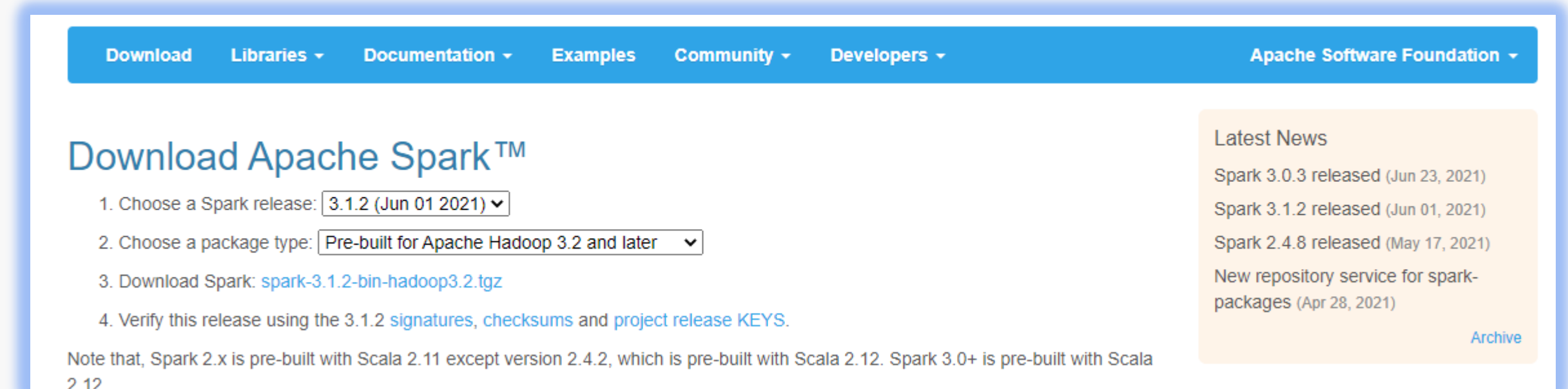
Começando a trabalhar

Vamos baixar o Apache Spark primeiro.

Instalando o Apache Spark

Para criarmos e executarmos nossos scripts em Pyspark, será necessário a instalação do Python, Spark e do Java. Neste treinamento trabalharemos com as versões para windows.

<https://spark.apache.org/downloads.html>



3. Download Spark: [spark-3.1.2-bin-hadoop3.2.tgz](#)

Após o download, descompacte usando usando o software 7-ZIP.

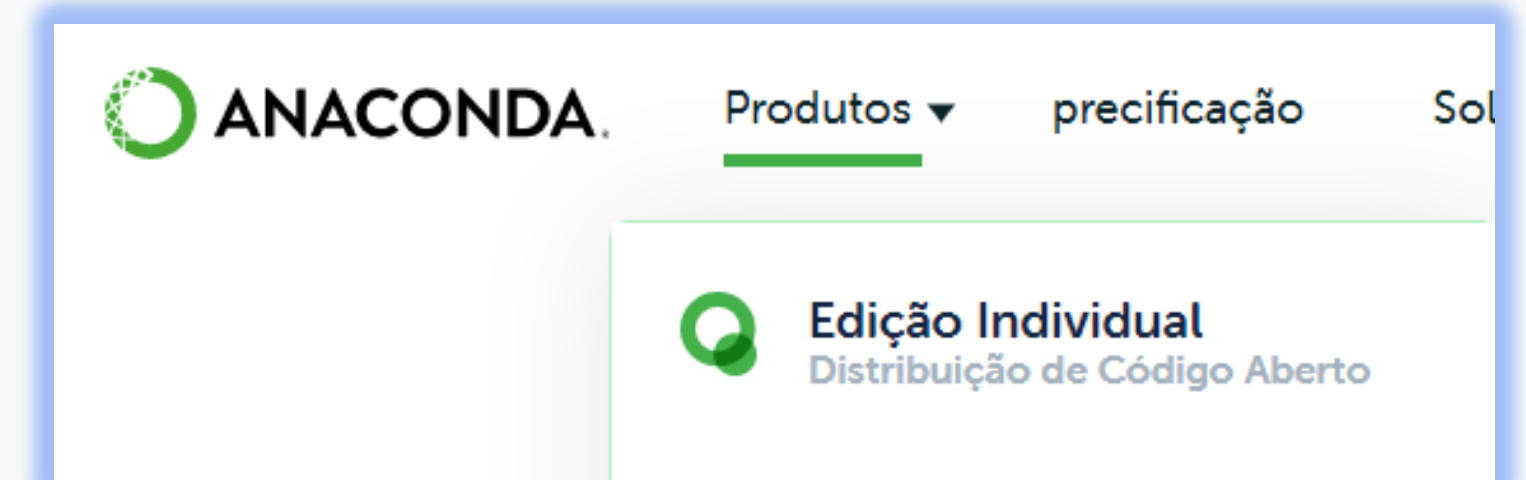
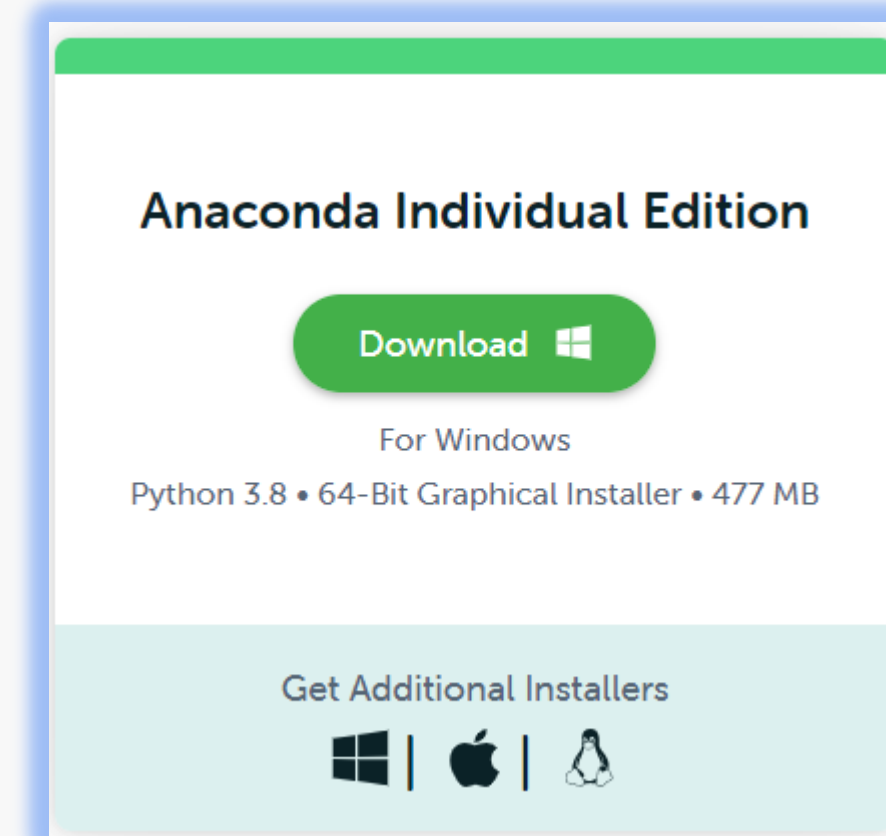
Começando a trabalhar

Vamos baixar o Anaconda.

Instalando o Anaconda

É uma ferramenta que possui a distribuição do Python já integrada com diversos pacotes instalados, possui a distribuição Individual (versão gratuita), facilitando a vida de quem trabalha com ciência dos dados. Inclui python, spyder IDE e notebook Jupyter.

<https://www.anaconda.com/products/individual>



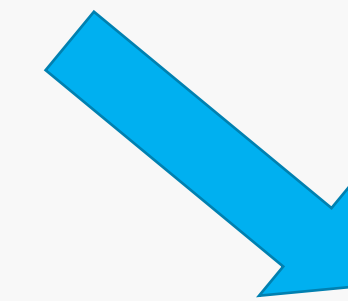
Começando a trabalhar

Vamos baixar o JAVA JDK.

Instalando o JAVA JDK

É um programa que inclui ferramentas como Máquina Virtual Java (JVM), bibliotecas e outros arquivos que suportam a execução de programas criados em java.

<https://www.oracle.com/java/technologies/javase-downloads.html>



Java SE Downloads

Java Platform, Standard Edition

Java SE 16






Java SE 16.0.1 is the latest release for the Java SE Platform

- Documentation
- Installation Instructions
- Release Notes
- Oracle License

Oracle JDK

- ↓ JDK Download
- ↓ Documentation Download



Java SE Development Kit 16.0.1		
This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE		
Product / File Description	File Size	Download
Linux ARM 64 RPM Package	144.87 MB	 jdk-16.0.1_linux-aarch64_bin.rpm
Linux ARM 64 Compressed Archive	160.72 MB	 jdk-16.0.1_linux-aarch64_bin.tar.gz
Linux x64 Debian Package	146.16 MB	 jdk-16.0.1_linux-x64_bin.deb
Linux x64 RPM Package	152.99 MB	 jdk-16.0.1_linux-x64_bin.rpm
Linux x64 Compressed Archive	170.02 MB	 jdk-16.0.1_linux-x64_bin.tar.gz

Ajustes das variáveis de ambiente do windows

É necessário para que os programas executem sem problemas.

Variáveis de ambiente do windows

As variáveis de ambiente permitem que qualquer parte do sistema operacional windows, possamos executar os seus programas.

```
# Variáveis para JDK
JAVA_HOME = C:\Java\jdk-16.0.1
PATH = C:\Java\jdk-16.0.1\bin

# Variáveis para Spark
SPARK_HOME = C:\spark
PATH = C:\spark\bin
PYSPARK_PYTHON = python3
PYSPARK_DRIVER_PYTHON = jupyter
PYSPARK_DRIVER_PYTHON_OPTS = notebook

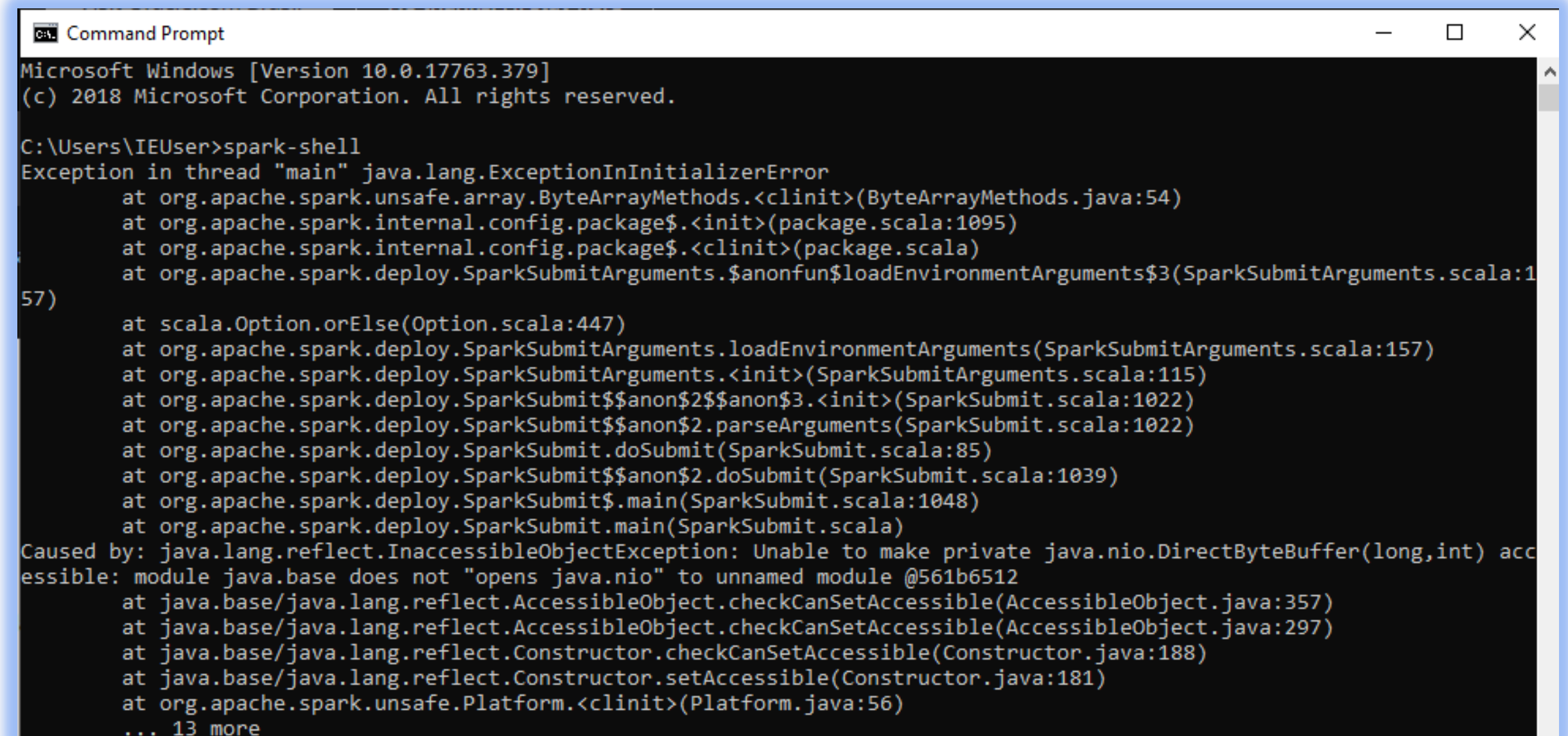
#Criação da pasta do hadoop
1) Criação do diretório: C:\Hadoop\bin
2) Copiar o programa wintools para o diretório C:\Hadoop\bin
3) Criação de nova variável de ambiente HADOOP_HOME, apontando para C:\Hadoop
   HADOOP_HOME = C:\Hadoop
4) Acrescenta o caminho de acesso ao Hadoop na variável de ambiente PATH
   PATH = C:\Hadoop\bin
```


Ajustes em caso de erro

É necessário para que os programas executem sem problemas.

Testar as aplicações

Possíveis ajustes nas aplicações



```
Command Prompt
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\IEUser>spark-shell
Exception in thread "main" java.lang.ExceptionInInitializerError
    at org.apache.spark.unsafe.array.ByteArrayMethods.<clinit>(ByteArrayMethods.java:54)
    at org.apache.spark.internal.config.package$.<init>(package.scala:1095)
    at org.apache.spark.internal.config.package$.<clinit>(package.scala)
    at org.apache.spark.deploy.SparkSubmitArguments.$anonfun$loadEnvironmentArguments$3(SparkSubmitArguments.scala:1
57)
    at scala.Option.getOrElse(Option.scala:447)
    at org.apache.spark.deploy.SparkSubmitArguments.loadEnvironmentArguments(SparkSubmitArguments.scala:157)
    at org.apache.spark.deploy.SparkSubmitArguments.<init>(SparkSubmitArguments.scala:115)
    at org.apache.spark.deploy.SparkSubmit$$anon$2$$anon$3.<init>(SparkSubmit.scala:1022)
    at org.apache.spark.deploy.SparkSubmit$$anon$2.parseArguments(SparkSubmit.scala:1022)
    at org.apache.spark.deploy.SparkSubmit.doSubmit(SparkSubmit.scala:85)
    at org.apache.spark.deploy.SparkSubmit$$anon$2.doSubmit(SparkSubmit.scala:1039)
    at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:1048)
    at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
Caused by: java.lang.reflect.InaccessibleObjectException: Unable to make private java.nio.DirectByteBuffer(long,int) acc
essible: module java.base does not "opens java.nio" to unnamed module @561b6512
    at java.base/java.lang.reflect.AccessibleObject.checkCanSetAccessible(AccessibleObject.java:357)
    at java.base/java.lang.reflect.AccessibleObject.checkCanSetAccessible(AccessibleObject.java:297)
    at java.base/java.lang.reflect.Constructor.checkCanSetAccessible(Constructor.java:188)
    at java.base/java.lang.reflect.Constructor.setAccessible(Constructor.java:181)
    at org.apache.spark.unsafe.Platform.<clinit>(Platform.java:56)
    ... 13 more
```

Entendendo funcionamento do Spark

O que é o Spark Context.

Spark Context

É considerado como o ponto de entrada entre o Spark e PySpark.
É necessário para a criação dos scripts que serão desenvolvidos no Pyspark. Para se conectar a fonte de dados e ao Spark utilizando o seu cluster é necessário criar o SPARK CONTEXT

Spark Context



Entendendo funcionamento do Spark

O que é o RDD.

RDD

RDD (Resilient Distributed Dataset) é a estrutura de dados que permite o armazenamento de dados dentro do Spark. Os RDDs são coleções de dados distribuídas que uma vez criados, não podem ser mais alterados(inmutáveis). Cada conjunto de dados em RDD é dividido em partições lógicas, que podem ser computadas em diferentes nós do cluster. É necessário que existe o Spark Context para se criar um RDD. Usado para dados estruturados e não estruturados.

Spark Context

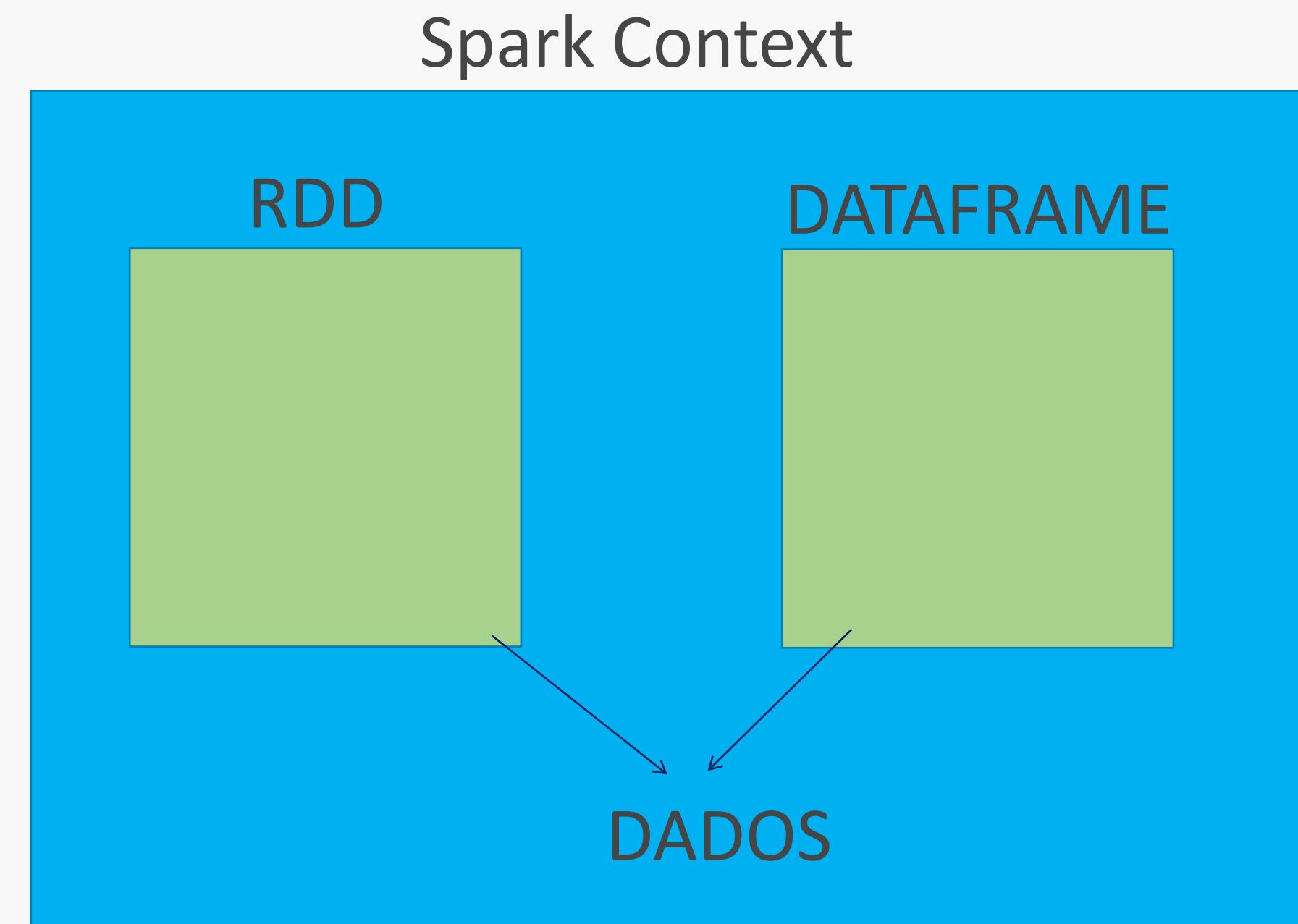


Entendendo funcionamento do Spark

O que é o Dataframe.

Dataframe

É um conjunto de dados organizado em colunas. É totalmente equivalente ao que conhecemos como tabelas de banco de dados. Usados para dados estruturados e semiestruturados.

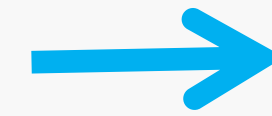


Detalhes importantes

Conheça algumas diferenças e detalhes importantes sobre RDD, Dataframe e Spark Context.

Importante Saber

Peculiaridades sobre RDD, Dataframe, Spark Context



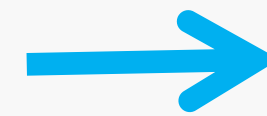
É possível criar um RDD e transformá-lo em Dataframe e vice-versa.



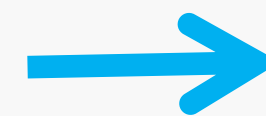
Ao subir o Jupiter notebook com o comando Pyspark, o SPARK Context é criado automaticamente.



Um Dataframe é um RDD de objetos linha, cada um representando um registro



Há outra forma de armazenamento é o Spark Dataset, mas somente tem suporte para JAVA e SCALA.



Em operações com dados muitas vezes não é possível guardá-los em um objeto Dataframe, (ex: JSON), os RDDs podem ser usados para pré-processar os dados com um grande volume de dados e depois podem ser transformados em Dataframes.



Existem transformações nos dados que podem ser realizadas em RDD que não são possíveis nos Dataframes.

Começando a trabalhar

Vamos começar com o uso do RDD.

Iniciando os trabalhos

Primeiramente vamos criar coleções de dados simples e depois trabalharemos com arquivos maiores.

```
Treinamento Pyspark

Primeiro script1 - RDD

In [ ]: import sys
        print(sys.version)

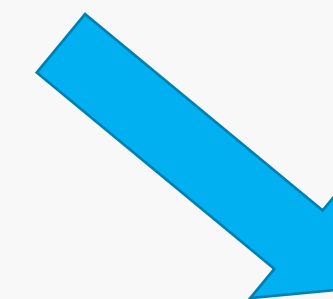
In [ ]: print(sc.version)

In [ ]: #Create RDD from parallelize
        paises = ["Brasil", "Italia", "Russia", "Noruega", "Espanha", "Mexico"]
        dadosrdd = spark.sparkContext.parallelize(paises)

In [ ]: dadosrdd.collect()

In [ ]: dadosrdd.count()

In [ ]: dadosrdd2 = spark.sparkContext.parallelize(paises)
        print("Particoes: " + str(dadosrdd2.getNumPartitions()))
```

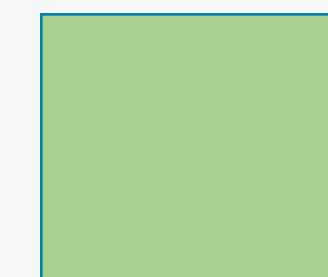


sparkContext.parallelize()

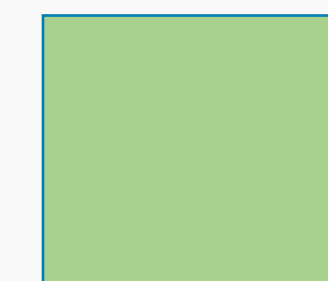
Lista Python



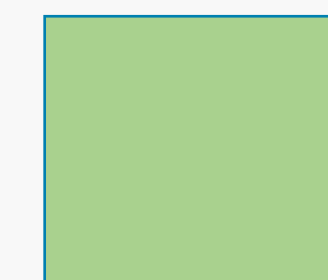
RDD



Partição 1



Partição 2



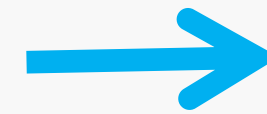
Partição 3

Entendendo operações

Compreendendo operações no
Pyspark.

Importante Saber

Detalhes sobre operações que realizaremos



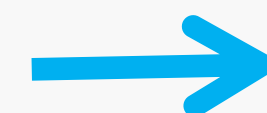
Collect() : Operação utilizada para recuperar todos os elementos do RDD/DataFrame (de todos os nós) para o nó principal.



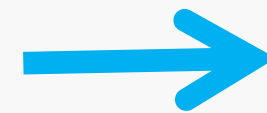
Count() : Retorna o número de linhas em um RDD/Dataframe



Parallelize() : Usado para criar um RDD a partir de uma coleção de listas. Partições são unidades básicas de paralelismo em PySpark. RDDs em PySpark são uma coleção de partições.



getNumPartitions() : Determina o número de partições RDD's atuais de uma coleção.



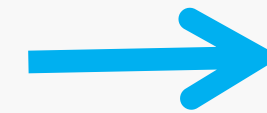
textFile () : Permite a leitura de um arquivo texto , indicando o local de origem do dado.



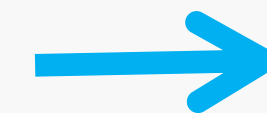
take() : Os primeiros 'x' elementos de um RDD



first() : Retorna o primeiro elemento de um RDD ou primeira linha de um Dataframe.



filter() : Retorna um novo RDD contendo apenas os elementos que satisfazem a condição.



cache() : Persiste os dados do RDD na memória.

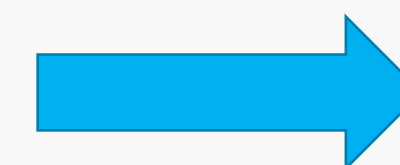
Entendendo a função Lambda

Qual a finalidade?

Função Lambda

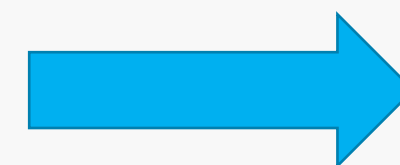
O Python suporta a criação de funções anônimas (ou seja, funções definidas sem nome), usando uma construção chamada "lambda".

```
def fmult(x):  
    return x*5
```



```
Print(fmult(5))  
25
```

```
fmt = lambda x: x*5
```



```
Print(fmt(5))  
25
```

1- Observe que a definição lambda não inclui uma instrução de "retorno" – ela sempre contém uma única expressão que é devolvida.

2- Você pode usar expressão condicional em uma função lambda ou/e ter mais de um argumento de entrada.

```
numeros = range(1, 35)
```

```
verifica_menor_zero = list(filter(lambda x: x < 0, numeros))
```

3- Uma função lambda pode ter mais de um argumento de entrada.

```
f = lambda x,y: ["ATIVO",x,y] if x>40 and y<140 else ["FALHA",x,y]
```

Entendendo operações

Compreendendo operações no
Pyspark.

Importante Saber

Detalhes sobre operações que realizaremos



flatMap(): É uma operação de transformação que retorna um RDD/DataFrame (colunas de DataFrame ou dados do RDD) depois de aplicar a função em cada elemento e retorna um novo RDD/DataFrame.



For: Comando de repetição utilizando em Python. Percorre os elementos de uma coleção de dados.



map() : É também uma operação que por meio de uma função de transformação realizada em cada elemento do RDD/DataFrame e retorna um novo RDD.

A diferença entre map e flatmap é que o **map** é uma operação que produz um valor de saída para cada valor de entrada, enquanto a **flatMap** é uma operação que produz um número arbitrário (zero ou mais) para cada valor de entrada.



from operator import add: Importa o módulo de operações do python trazendo a função de adição de elementos.



reduceByKey(): A transformação do SPARK RDD é usada para realizar merge(mesclar) os valores de cada chave usando uma função de redução associativa. Conceito de CHAVE,VALOR.



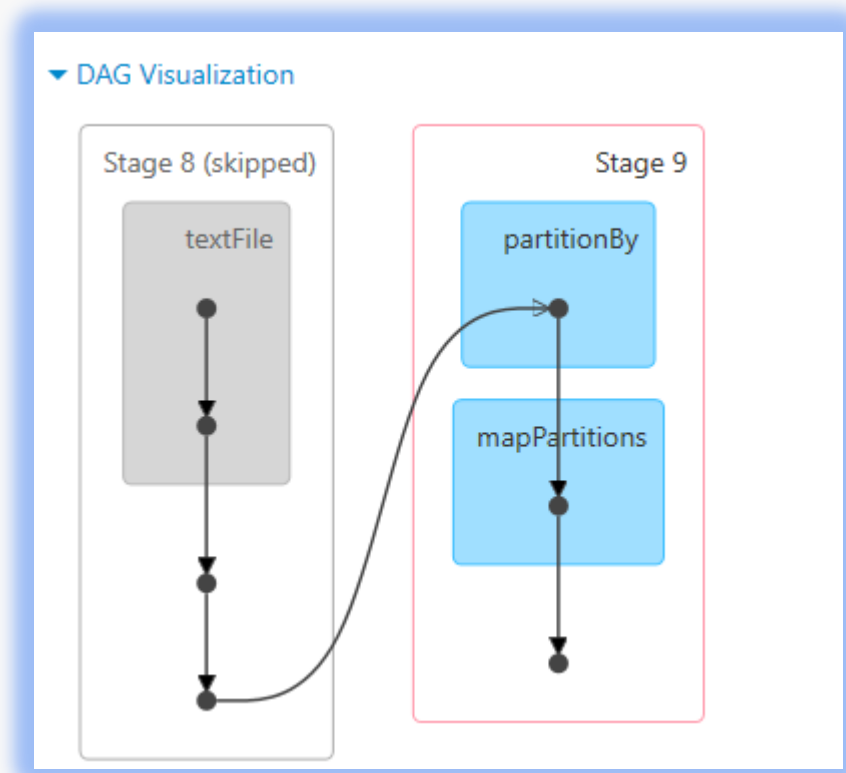
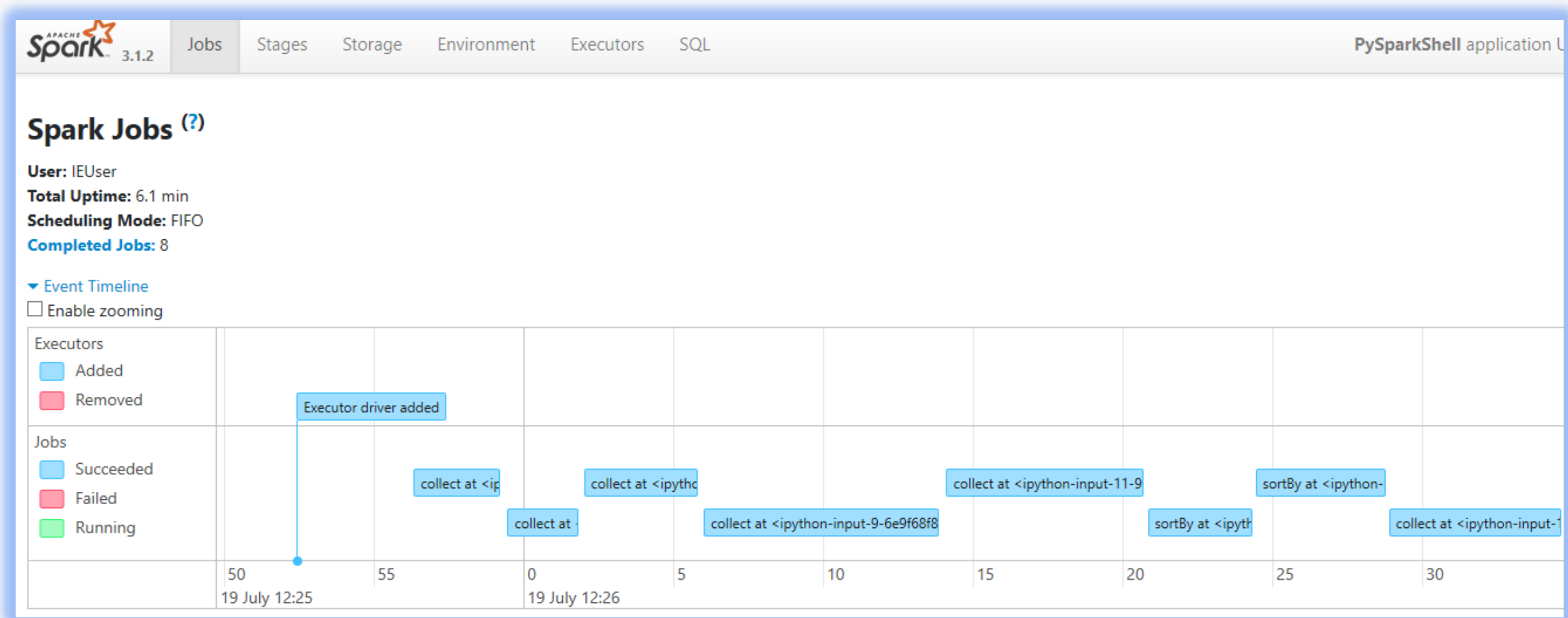
sortBy() : Classifica os elementos em um RDD de acordo com a função (keyfunc) do dado.

Monitoramento dos scripts

Spark UI ou Web UI, onde faz-se o monitoramento dos scripts.

Spark UI

Ferramenta de monitoramento ligada ao Spark



Summary												
	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write
Active(1)	0	27.2 KiB / 434.4 MiB	0.0 B	3	0	0	20	20	1.2 min (0.0 ms)	9.4 KiB	10.4 KiB	4.1 KiB
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B
Total(1)	0	27.2 KiB / 434.4 MiB	0.0 B	3	0	0	20	20	1.2 min (0.0 ms)	9.4 KiB	10.4 KiB	4.1 KiB

Transformações

JOIN, LEFT e RIGHT

Vamos utilizar o RDD para a união de coleções de dados.

Trabalhando com operações

A união de dados é muito comum em banco de dados, faremos estas transformações em dois RDDs

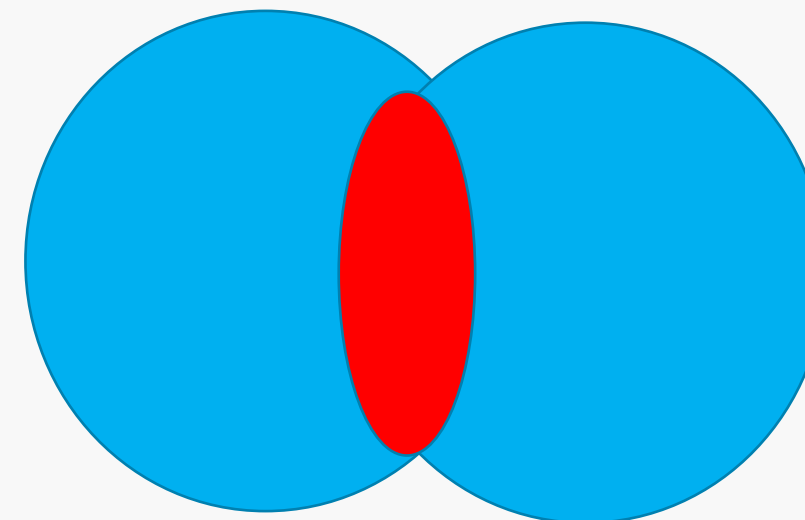
UTILIZANDO OPERAÇÕES DE JOIN, LEFT e RIGHT

```
In [19]: #Leitura dos arquivos de profissionais e salarios
profissional=spark.sparkContext.textFile("/Treinamento/Dados/Folhapagamento/profissionais.txt")
salario=spark.sparkContext.textFile("/Treinamento/Dados/Folhapagamento/salario.txt")
```

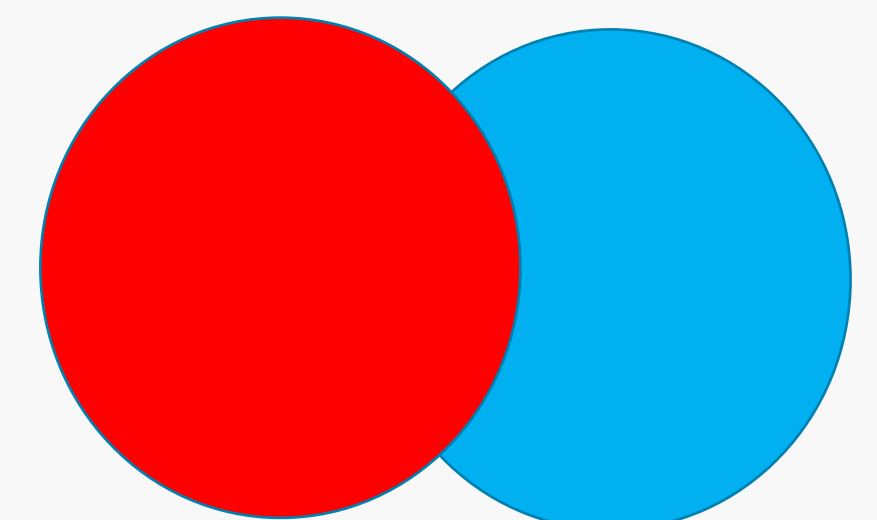
```
In [20]: # Exibindo os dados sobre os profissionais
profissional.collect()
```

```
Out[20]: ['Carlos,oncologista,hospital',
'Ana,dentista,clinica',
'Fernanda,enfermeira,hospital',
'Sandra,pediatra,clinica',
'Fatima,dentista,clinica',
'Gilmar,cardiologista,hospital',
'Fabio,pediatra,clinica',
'Hilton,enfermeiro,clinica',
'Daiane,dentista,clinica',
'Paulo,farmaceutico,clinica',
'Gilberto,pediatra,hospital']
```

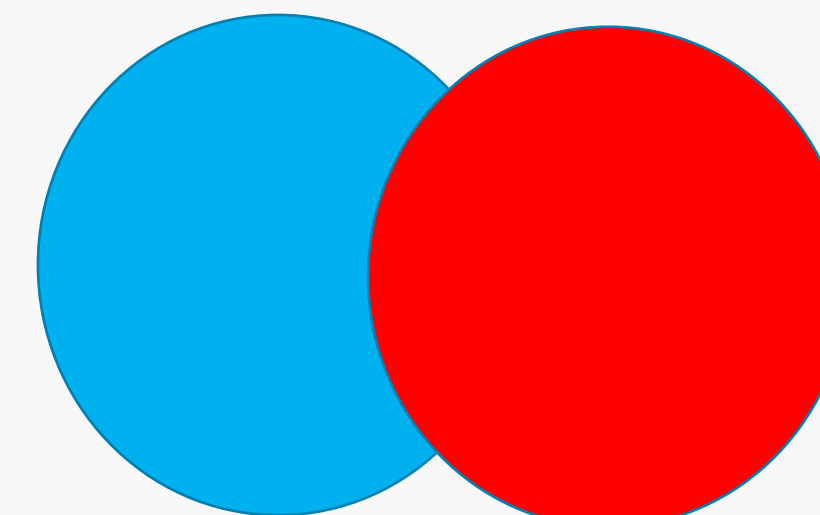
JOIN



LEFT OUTER JOIN



RIGHT OUTER JOIN



Entendendo operações

Compreendendo transformações no
Pyspark.

Importante Saber

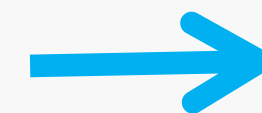
Detalhes sobre transformações que realizaremos



`join()` : Transformação utilizada para retornar os registros correspondentes em ambos os RDDs. Digamos que um RDD (C,V1) e outro RDD contenha (C,V2) e depois junte-se internamente retornando o RDD (C,(V1,V2)).



`leftOuterJoin()` : Transformação utilizada para retornar apenas os registros correspondentes ao primeiro RDD e que existam ou não no segundo RDD. Digamos que um RDD (C,V1) e (C1,V2) outro RDD contenha (C1,V3) e (C2,V4) e depois se juntem retornando o RDD (C1,(V2,V3)),(C,V1)



`rightOuterJoin()` : Transformação utilizada para retornar apenas os registros correspondentes ao segundo RDD e que existam ou não no primeiro RDD. Digamos que um RDD (C,V1) e (C1,V2) outro RDD contenha (C1,V3) e (C2,V4) e depois se juntem retornando o RDD (C1,(V2,V3)),(C2,V4)

Entendendo funcionamento do Spark

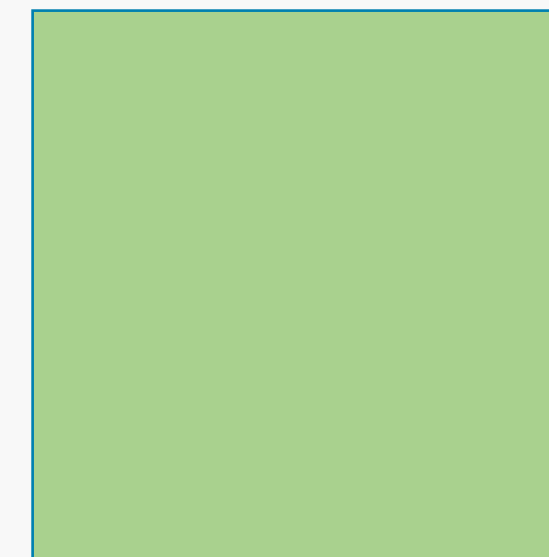
O que é um Dataframe.

Dataframe

É uma estrutura de dados que permite a guarda de dados de forma bidimensional com colunas de tipos de dados que podem ser diferentes. você pode criar um PySpark DataFrame a partir de fontes de dados como TXT, JSON, CSV, Parquet, XML lendo os formatos HDFS, S3, Azure Blob, dentre outros.

Um PySpark Dataframe pode ser criado passando uma lista, um Dataframe Pandas e por meio de um RDD que consiste em trazer uma lista.

Dataframe



Entendendo funcionamento do Spark

O que é um SparkSession.

SparkSession

O SparkSession é um novo objeto adicionado desde a versão Spark 2.x.

O SparkSession tornou-se um ponto de entrada para a Spark. Utilizado principalmente para a operação SQL no Dataframe.

Internamente, o SparkSession cria um novo SparkContext para todas as operações (trabalhar com RDD, Dataframe e Dataset).

Antes poderíamos criar principalmente apenas RDDs usando o Spark Context.

Assim como o SparkContext o SparkSession também é criado ao inicializar o Spark, mas você pode definir novas sessões no Spark. É possível até na criação da sessão, informar sobre o nó de execução e o total de memória a ser alocada.

```
from pyspark.sql import SparkSession  
spark = SparkSession.builder.getOrCreate()
```

Começando a trabalhar

Vamos começar a trabalhar com
Dataframe.

Iniciando os trabalhos

Primeiramente vamos criar coleções de dados simples e depois trabalharemos com arquivos maiores.

Criação de Dataframe

```
In [29]: #Carregando dados em um Dataframe e Listando
from datetime import datetime, date
from pyspark.sql import Row

datf=spark.createDataFrame([
    Row(Cidade='Salvador', Populacao=6., Time='Bahia', Dat_cadastro=date(2021, 10, 10)),
    Row(Cidade='Sao Paulo',Populacao=45., Time='Sao Paulo', Dat_cadastro=date(2021, 5, 5)),
    Row(Cidade='Recife', Populacao=3., Time='Sport',Dat_cadastro=date(2021, 7, 2)),
    Row(Cidade='Maceio', Populacao=2., Time='CSA', Dat_cadastro=date(2021, 8, 1)),
])
datf
```

```
Out[29]:
```

Cidade	Populacao	Time	Dat_cadastro
Salvador	6.0	Bahia	2021-10-10
Sao Paulo	45.0	Sao Paulo	2021-05-05
Recife	3.0	Sport	2021-07-02
Maceio	2.0	CSA	2021-08-01

Carregando dados de um arquivo

Vamos carregar dados de um arquivo texto, o food_coded.csv.

Analizando dados sobre alimentos

Vamos começar a gerar informações e analisar os dados no Dataframe

Carregando dados de um arquivo para um Dataframe

```
In [ ]: #importando sparksession
        from pyspark.sql import SparkSession

        #criando um objeto sparksession object e um appName
        spark=SparkSession.builder.appName("fooddf").getOrCreate()

In [ ]: # Fazendo a leitura do arquivo food_coded.csv
        df = spark.read.option("header", "true").csv("/Treinamento/Dados/food_coded.csv")

In [ ]: # Exibindo as 10 primeiras linhas
        df.show(10)

In [ ]: # exibindo o Schema
        df.printSchema()
```


Trabalhando com SPARK SQL

Criando consultas nos dados.

Analizando dados sobre Bikes da Índia

Vamos começar a gerar informações e analisar os dados com o uso do SPARK SQL

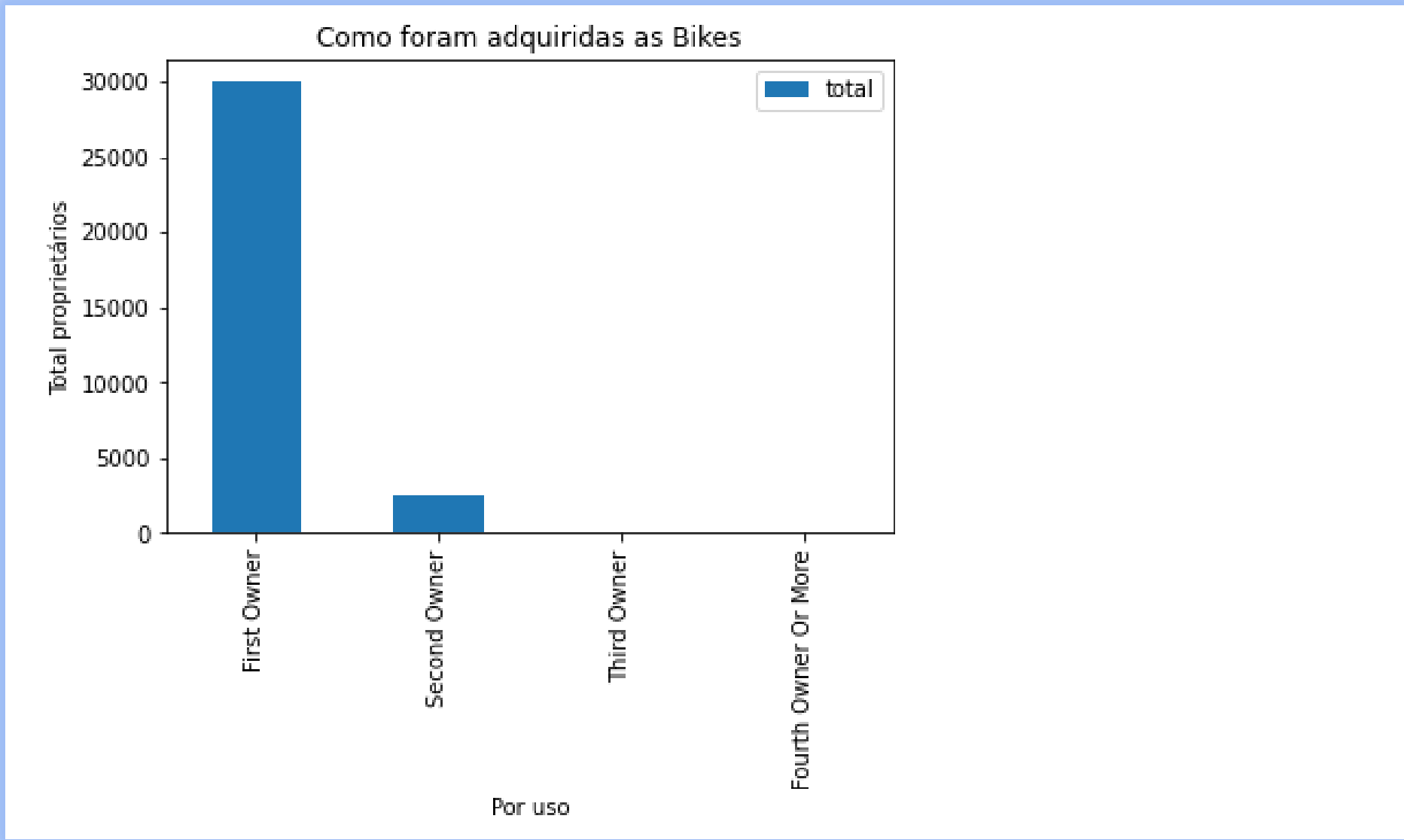
```
In [ ]: #importando sparksession
from pyspark.sql import SparkSession

#criando um objeto sparksession object e um appName
sparkSession=SparkSession.builder.appName("bikes").getOrCreate()

In [ ]: # Fazendo a leitura do arquivo Bikes.csv
bikes = sparkSession.read.option("header", "true").csv("/Treinamento/Dados/Bikes.csv")

In [48]: # Criando uma tabela temporária em memória com os dados e utilizando consulta SQL
bikes.createOrReplaceTempView("tab_bikes")
sparkSession.sql("select * from tab_bikes").show()
```

bike_name	price	city	kms_driven	owner	age	power	brand
TVS Star City Plu...	35000.0	Ahmedabad	17654.0	First Owner	3.0	110.0	TVS
Royal Enfield Cla...	119900.0	Delhi	11000.0	First Owner	4.0	350.0	Royal Enfield
Triumph Daytona 675R	600000.0	Delhi	110.0	First Owner	8.0	675.0	Triumph
TVS Apache RTR 180cc	65000.0	Bangalore	16329.0	First Owner	4.0	180.0	TVS
Yamaha FZ S V 2.0...	80000.0	Bangalore	10000.0	First Owner	3.0	150.0	Yamaha



Trabalhando com arquivos JSON

Criando consultas nos dados em arquivos JSON.

Analizando dados sobre análise de sentimentos

Vamos começar a gerar informações e analisar os dados com a leitura de um arquivo JSON, utilizando ajustes em SPARK SQL.

Carregando dados de um arquivo JSON

- Vamos analisar dados dentro de um arquivo JSON, dados mensagens sobre análise de sentimentos:

1. Carregar o arquivo de ados JSON
2. Entender o Schema gerado pelo arquivo
3. Separá-lo em fragmentos
4. Criar uma visão com os dados para anlaisar dentro do SPARK SQL
5. Mostrar as frases com sentimentos negativos

```
In [ ]: #importando sparksession
from pyspark.sql import SparkSession

#criando um objeto sparksession object e um appName
sparkSession=SparkSession.builder.appName("sentimento").getOrCreate()
```

```
In [ ]: # Carregando os dados arquivo JSON
path = "/Treinamento/Dados/sentimento.json"
sentimentoDF = sparkSession.read.json(path,multiline = "true")
```

```
In [50]: #Pesquisando dentre os dados todos que possuem o tipo de sentimento negativo
sparkSession.sql("select context, question from tipo_sentimento where tps = 'negative']").show()
```

```
+-----+-----+
|      context|  question|
+-----+-----+
| i dont think you...|[negative]|
|headache  wanna s...|[negative]|
|   miss you my dear|[negative]|
|today dan bought ...|[negative]|
| oo noo thats not...|[negative]|
|misses her phone....|[negative]|
|so i have like no...|[negative]|
|also i popped the...|[negative]|
|              uh oh|[negative]|
```


Trabalhando com SPARK Streaming

Entendendo o funcionamento.

Analizando dados com SPARK Streaming

Os dados de entrada são transmitidos e são divididos em pacotes dentro do SPARK. O SPARK Streaming é uma extensão da API do SPARK.

DADOS EM
BATCH

DADOS EM
STREAMING

PRÁTICA

ENVIE AO PROFESSOR

PREPARE UM ESTUDO

Carregando qualquer arquivo do site **kaggle.com** e crie um scripts utilizando dados em RDD, Dataframe e SPARK SQL e analise seus dados.

Muito boa sorte e conte comigo!

Criação de Dataframe

```
In [29]: #Carregando dados em um Dataframe e Listando
from datetime import datetime, date
from pyspark.sql import Row

datf=spark.createDataFrame([
    Row(Cidade='Salvador', Populacao=6., Time='Bahia', Dat_cadastro=date(2021, 10, 10)),
    Row(Cidade='Sao Paulo',Populacao=45., Time='Sao Paulo', Dat_cadastro=date(2021, 5, 5)),
    Row(Cidade='Recife', Populacao=3., Time='Sport',Dat_cadastro=date(2021, 7, 2)),
    Row(Cidade='Maceio', Populacao=2., Time='CSA', Dat_cadastro=date(2021, 8, 1)),
])
datf
```

Out[29]:

Cidade	Populacao	Time	Dat_cadastro
Salvador	6.0	Bahia	2021-10-10
Sao Paulo	45.0	Sao Paulo	2021-05-05
Recife	3.0	Sport	2021-07-02
Maceio	2.0	CSA	2021-08-01

```
In [50]: #Pesquisando dentre os dados todos que possuem o tipo de sentimento negativo
sparkSession.sql("select context, question from tipo_sentimento where tps = 'negative'").show()

+-----+-----+
|      context| question|
+-----+-----+
| i dont think you...|[negative]|
|headache  wanna s...|[negative]|
|  miss you my dear|[negative]|
|today dan bought ...|[negative]|
| oo noo thats not...|[negative]|
|misses her phone....|[negative]|
|so i have like no...|[negative]|
|also i popped the...|[negative]|
|          uh oh|[negative]|
```