1. **Introdução**
   1. **Google Colab**

- O que é o Google Colab? Ok

O Colab permite que qualquer pessoa escreva e execute código Python pelo navegador e é especialmente adequado para aprendizado de máquina e análise de dados. Tem uma interface de notebook em ambiente colaborativo.

Link: <https://colab.research.google.com/>

* 1. **SparkSession**

- SparkSession

- O que é o driver?

É a maquina principal onde ocorre o gerenciamento de criação e quem executará o código. Neste código possui o spark session que é o ponto de entrada. O Spark Session pega as suas tarefas e as divide em tarefas menores que serão tratadas pelos executores.

- O que é o **cluster manager**?

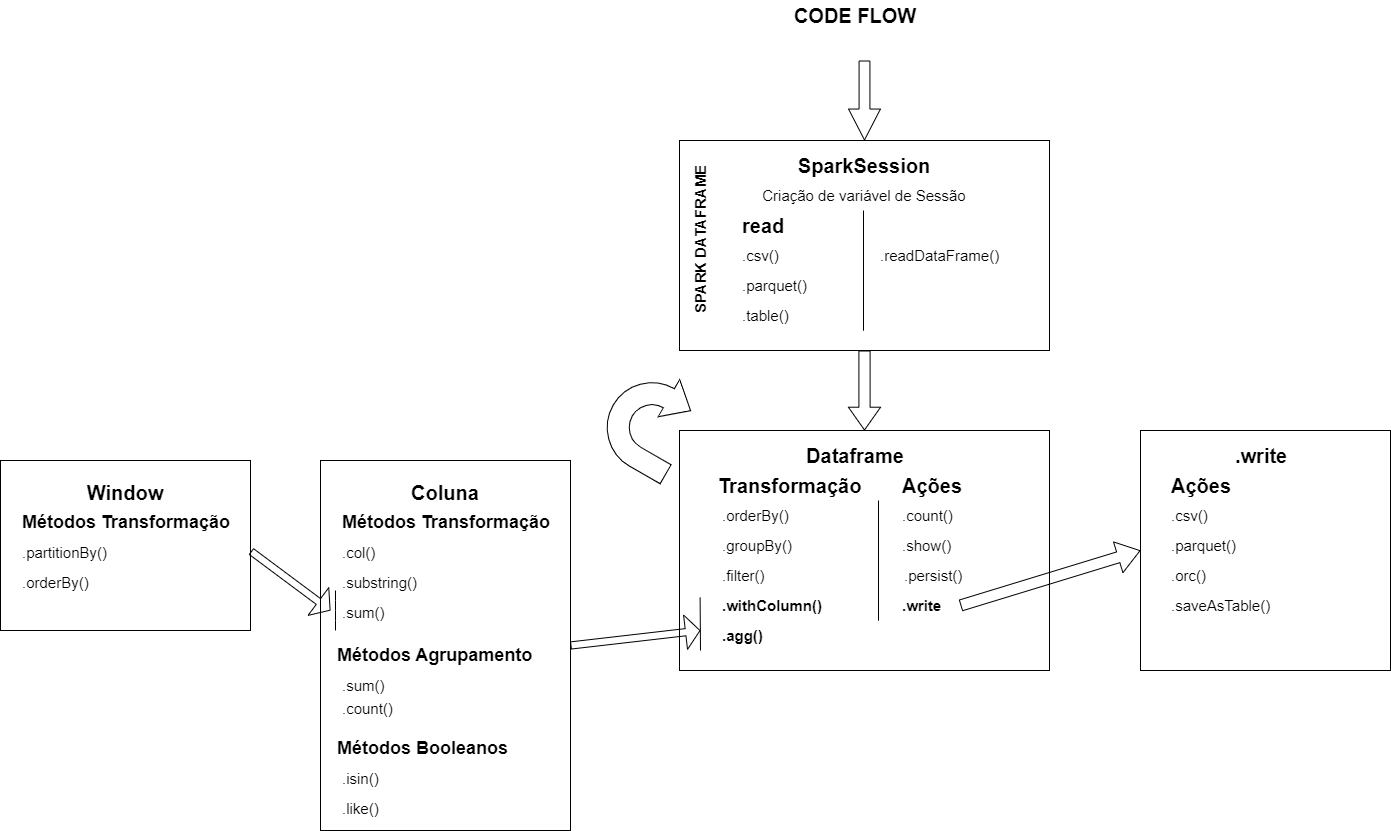
É o gerenciador de tarefas (ETL) e jobs que só é necessário se o Spark for executado de forma distribuída e também é responsável pela administração das máquinas que serão utilizadas como workers. O gerenciador de cluster se comunica com o driver e os executores para gerenciar a alocação de recursos, divisão do programa e execução do programa.

- O que são os **work nodes**?

São as máquinas conectadas ao cluster que são responsáveis pela execução das tarefas designadas pelo cluster manager. Se o Spark for executado de forma local, a máquina fará o papel de Driver Program e de Worker.

Arquitetura do Spark

A arquitetura de um código utilizando o spark é constituidas utilizando alguns objetos como: dataframe, colunas e entre outros. Para iniciarmos na hora de codar, nosso ponto de inicio para construir um dataframe é o SparkSession (driver program). O spark session é responsável por construir uma variável a qual conectará o ambiente do cluster, ou seja, a integração de sistemas para efetuar a carga dos dados (processo de ETL), possibilitando a leitura de dataframes.





Instalação do Spark

# instalar as dependencias

# baixar a versão mais recente do java8;

!apt-get install openjdk-8-jdk-headless -qq > /dev/null

#download via stp do hadoop

!wget -q https://dlcdn.apache.org/spark/spark-3.4.2/spark-3.4.2-bin-hadoop3.tgz

!tar xf spark-3.4.2-bin-hadoop3.tgz

# instalar o Findspark

!pip install -q findspark

import os

os.environ["JAVA\_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"

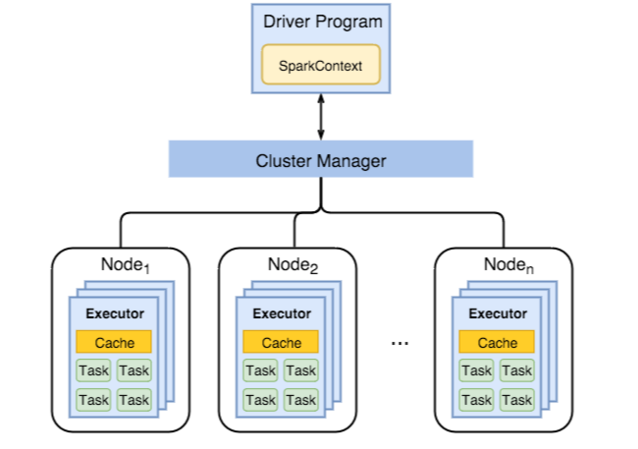
os.environ["SPARK\_HOME"] = "/content/sample\_data/spark-3.3.2-bin-hadoop3"

# utilizando findspark para que o spark seja "importavel"

import findspark

findspark.init('spark-3.4.2-bin-hadoop3')

Cluster - O spark conecta nosso código executado no driver ao cluster atráves do SparkSession.

 Arquitetura de Objetos do SparkSession

1. Criando Dataframe
   1. Dataframe com arquivo CSV

Iniciando a variável SparkSession

from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate();

dir(spark)

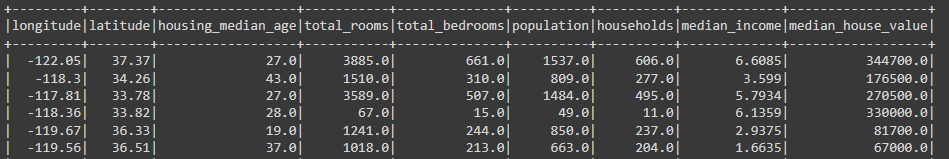
!ls sample\_data

Lendo ao arquivo CSV e colocando em um dataframe:

# @title Texto de título padrão

df = spark.read.csv(path='/content/sample\_data/california\_housing\_test.csv',sep=',',header=True, inferSchema=True)

df.show()

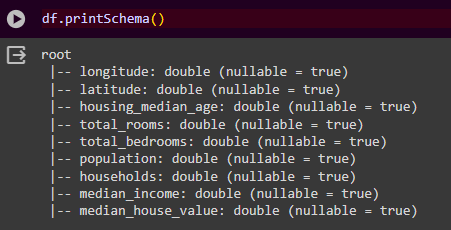


**Path**: É o diretório que o arquivo encontra-se armazenado;

**Sep**: É o separador que se utiliza para que as colunas sejam separadas dentro do arquivo;

**Header**: É o boleano que utiliza a primeira linha para a nomeação das colunas do arquivo;

**InferSchema**: Presume o datatype baseando-se nas colunas que estão nele;



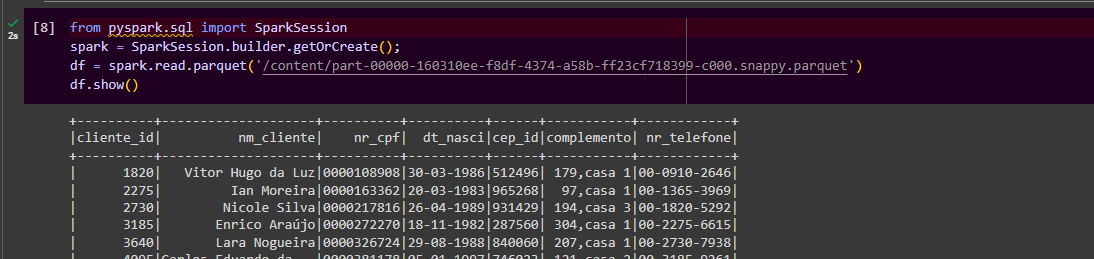
* 1. Dataframe com arquivo ORC

Parquet é um formato de arquivo colunar que fornece optimizações para acelerar as consultas e é um formato muito mais eficiente do que CSV ou JSON, suportado por muitos sistemas de processamento de dados. É compatível com a maioria das estruturas de processamento de dados. Fornece esquemas eficientes de compressão e codificação de dados com desempenho melhorado para lidar com dados complexos em massa.

Arquivos nesse formato reduzem o armazenamento de dados em média em 75%. O Spark por padrão suporta Parquet na sua biblioteca, logo não precisamos de adicionar quaisquer bibliotecas de dependência.

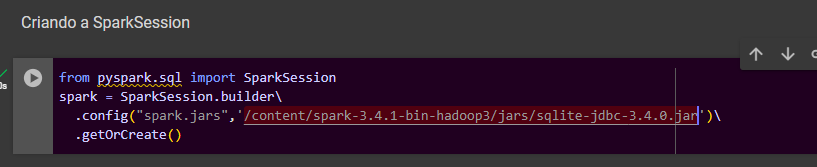
Abaixo estão algumas das vantagens de usar o Parquet. A combinação destes benefícios com Spark melhora o desempenho e dá a capacidade de trabalhar com estrutura de arquivos. Reduz as operações de IO (maior eficiência as transformações).

Obtém colunas específicas a que precisa de acessar (particionamentos). Consome menos espaço. Suporta Datatype nas colunas.



* 1. Conexão JDBC

Para teste de JBDC necessitaremos copiar o arquivo driver (.jar) que faz a conexão com o SQLite no seguinte caminho “/content/spark-3.4.1-bin-hadoop3/jars/sqlite-jdbc-3.4.0.jar”



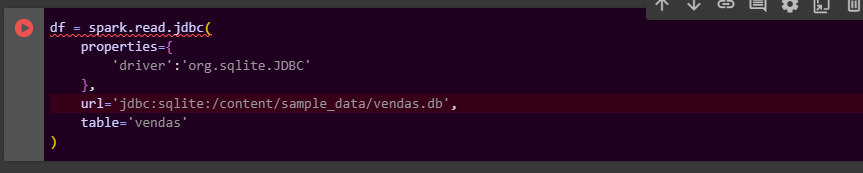
Parâmetros importantes JDBC:

Properties: dicionário que deverá conter o driver

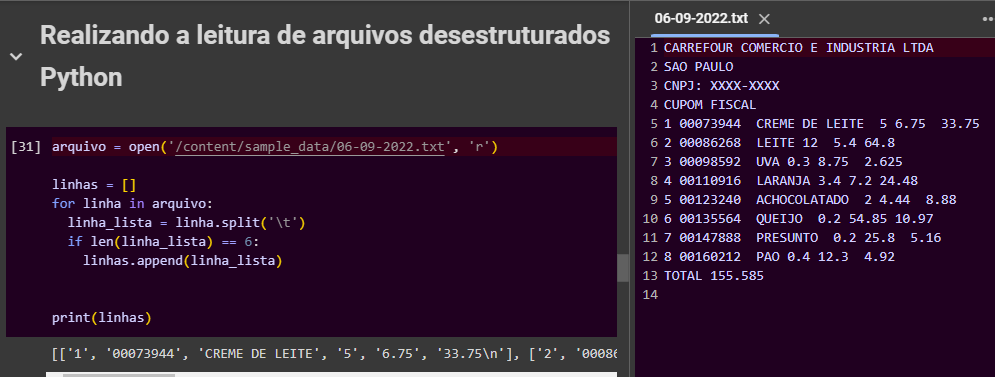
url: url do banco de dados

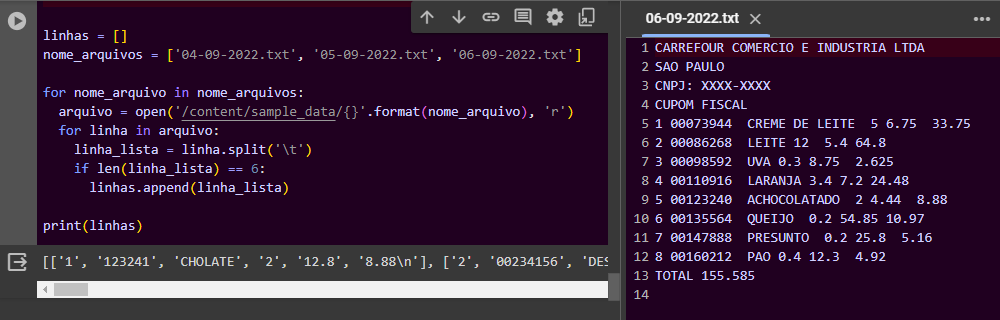
table: tabela dentro do banco de dados

Realizando a leitura de um banco de dados com JDBC



* 1. Lidando com dados desestruturados
     1. Realizando a leitura de arquivos desestruturados com Python





linhas = []

nome\_arquivos = ['04-09-2022.txt', '05-09-2022.txt', '06-09-2022.txt']

for nome\_arquivo in nome\_arquivos:

  arquivo = open('/content/sample\_data/{}'.format(nome\_arquivo), 'r')

  for linha in arquivo:

    linha\_lista = linha.split('\t')

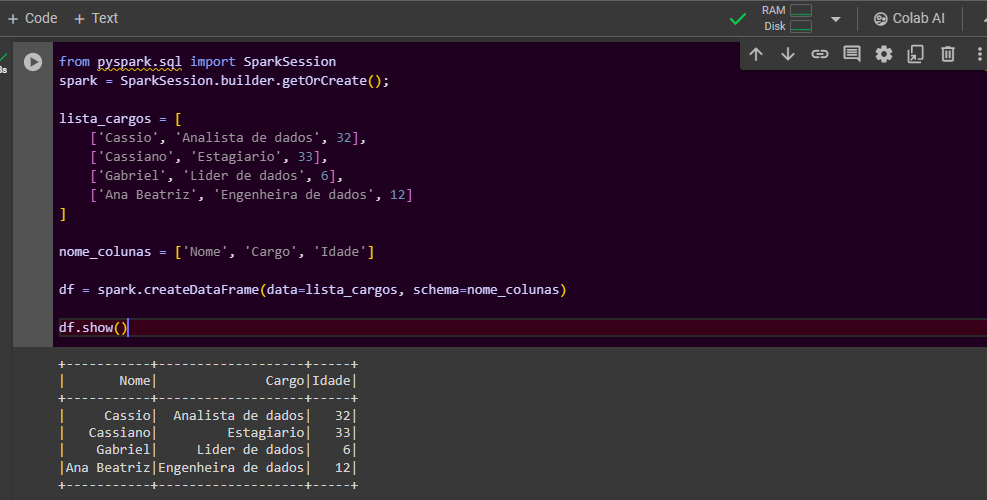
    if len(linha\_lista) == 6:

      linhas.append(linha\_lista)

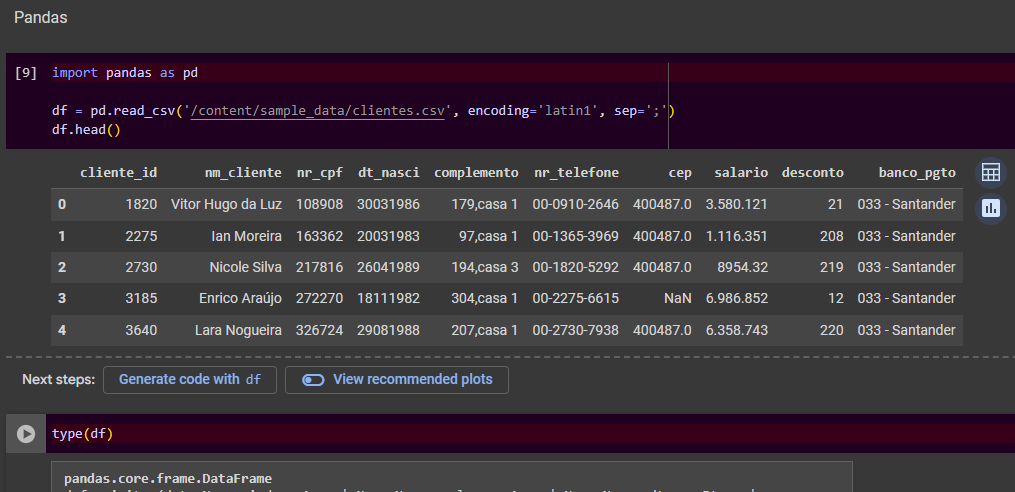
print(linhas)

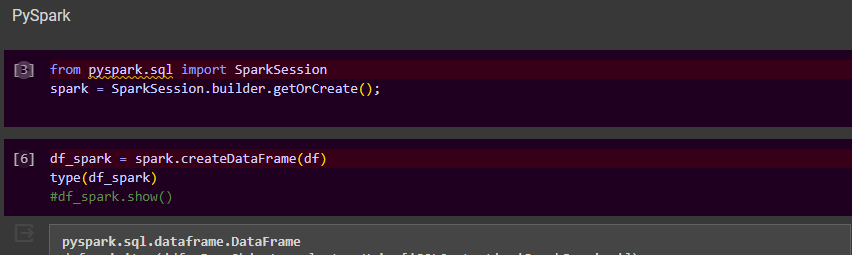
* + 1. Spark Create Dataframe

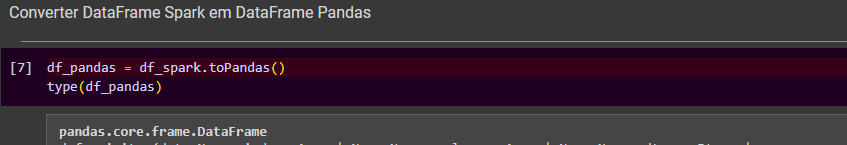
Criação de um dataframe a partir de uma lista de lista com o nome das colunas utilizando o Spark.



* + 1. Pandas Dataframe para Spark Dataframe

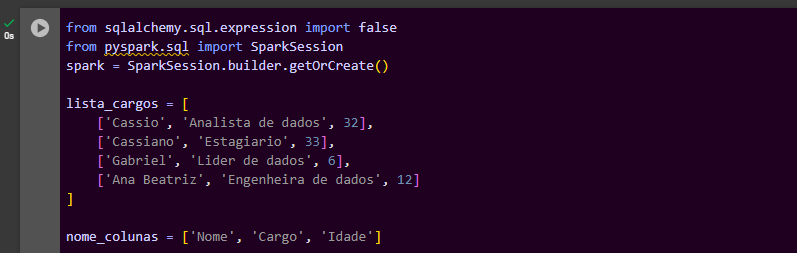


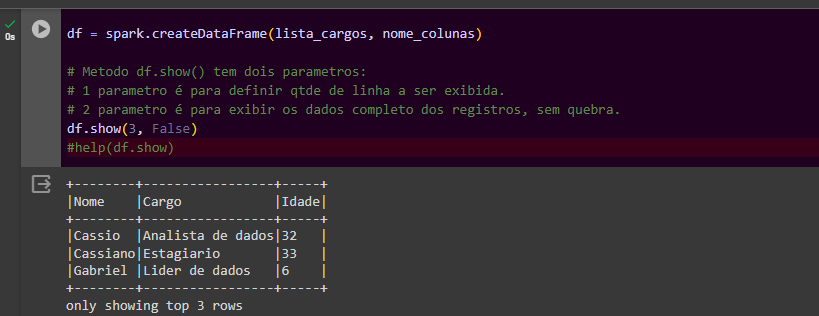




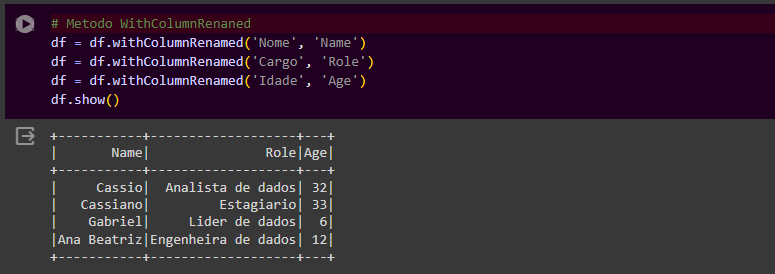
1. Métodos do DataFrame
   1. Método WithColumnRenamed e Select

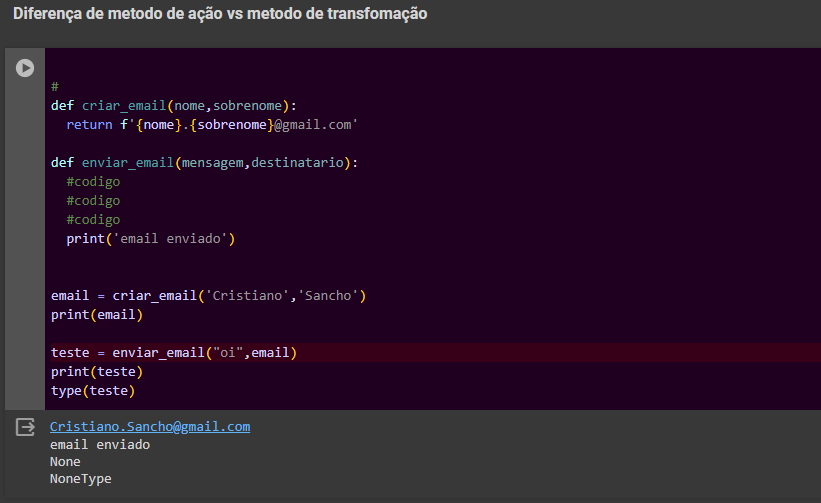
Criando dataframe através de uma lista

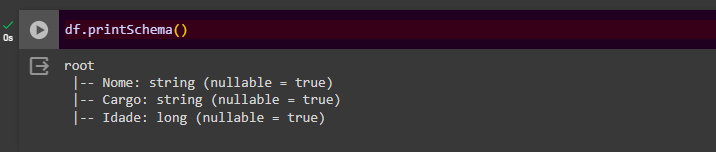




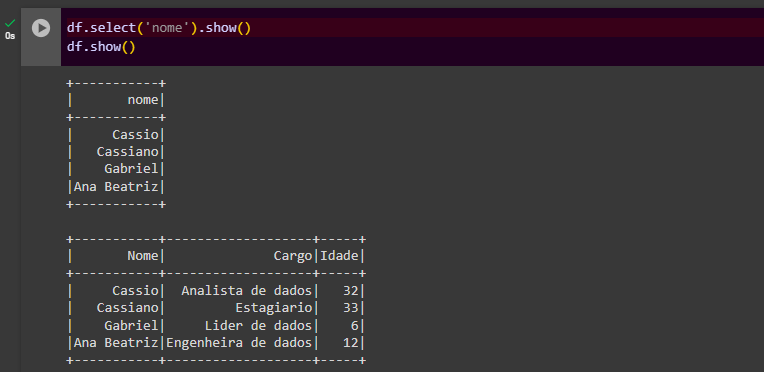
Renomeando os campos do dataframe



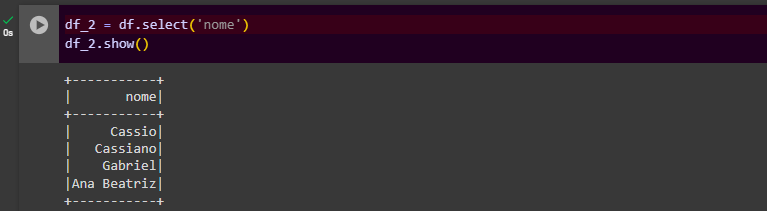




Método select

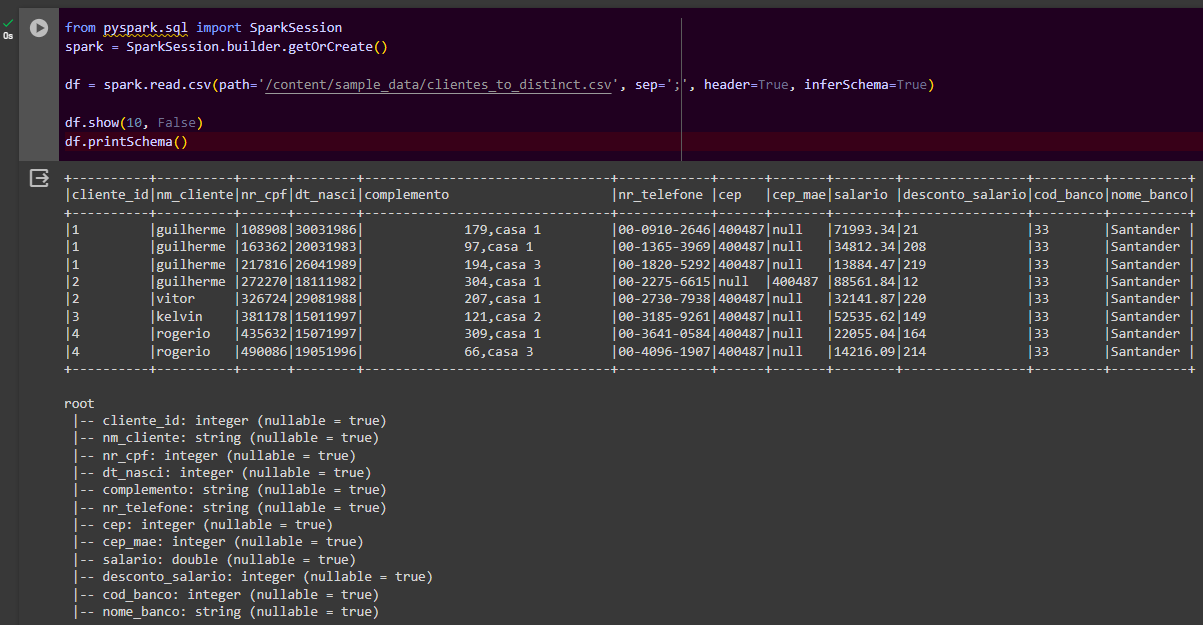


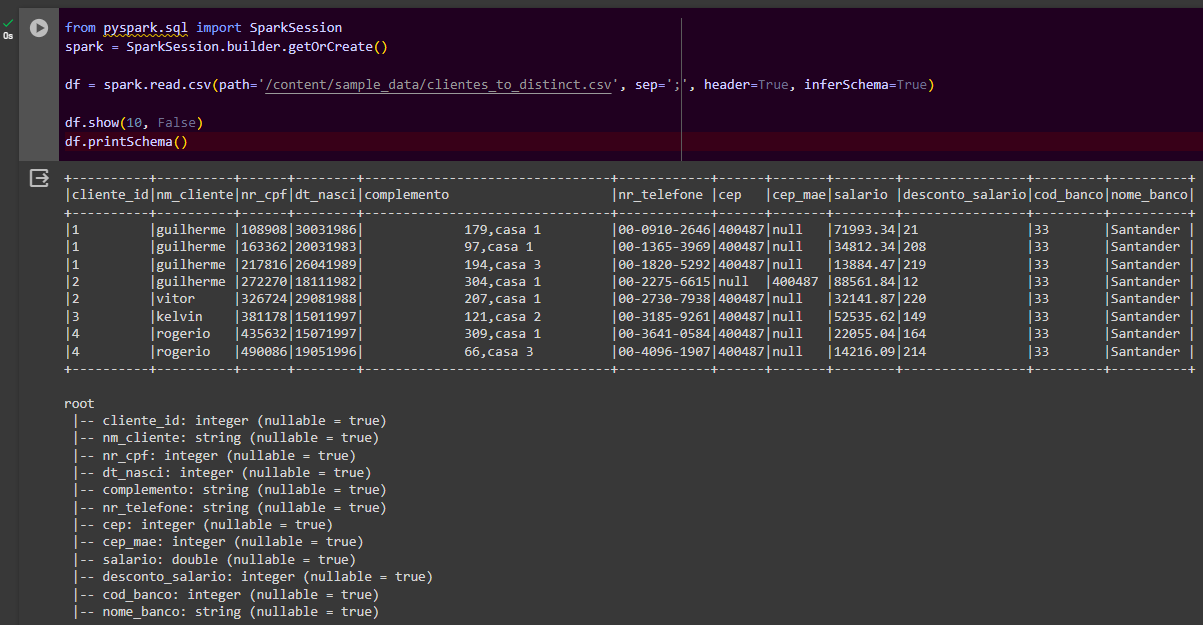
Posso atribuir um dataframe com colunas selecionada para uma variável nova, onde nesta nova variável teremos apenas o dataframe com o que for selecionado.



* 1. Alinhamento de método e como usar Count, OrderBy, Tak, Collect e ToPandas

Criação do dataframe com separador, nome das colunas (header) e tipo de dados (inferSchema)

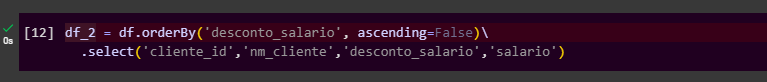


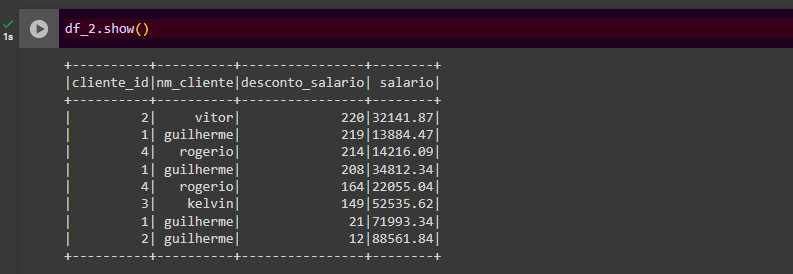


Metodo de ação - OrdeBy

Ordenar o dataframe pela colunas de chave e retorna um novo dataframe.

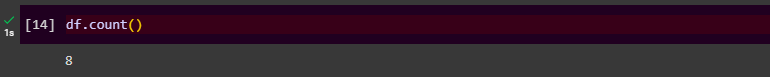
No código acima fizemos um alinhamento de método para melhor leitura do código.





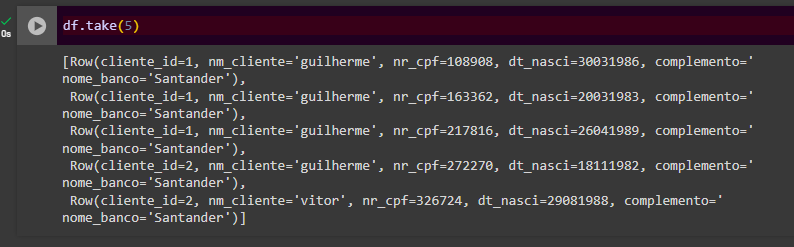
Metodo de ação - count()

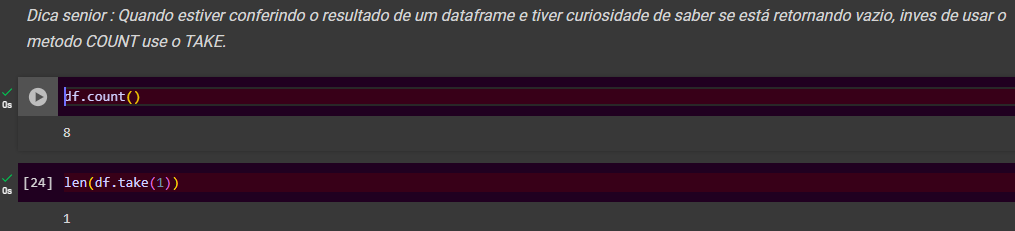
retorna a contagem de linhas de um dataframe.



Metodo de ação - take()

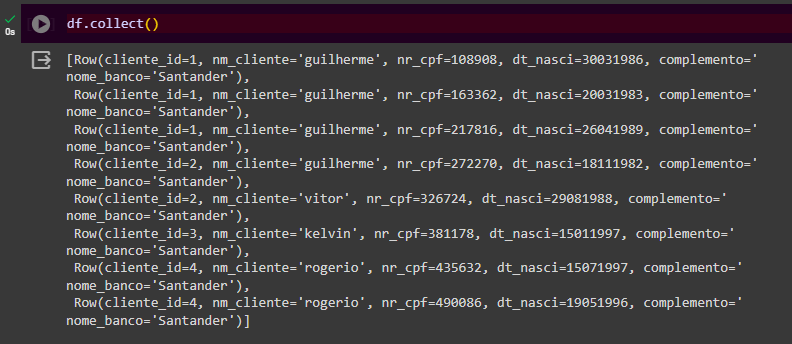
Mostrar a primeira linha em formato de uma Lista de um dataframe.





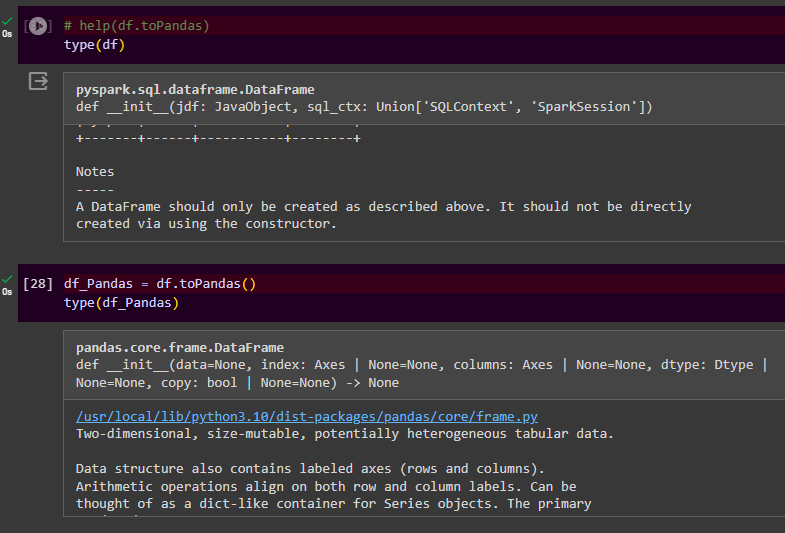
Metodo de ação - collect()

Retorna as linhas do dataframe em uma lista do python, semelhante ao take, no entanto, o collect traz tudo numa lista. CUIDADO: Se o dataframe for muito grande, pode prejudicar a memoria do driver.

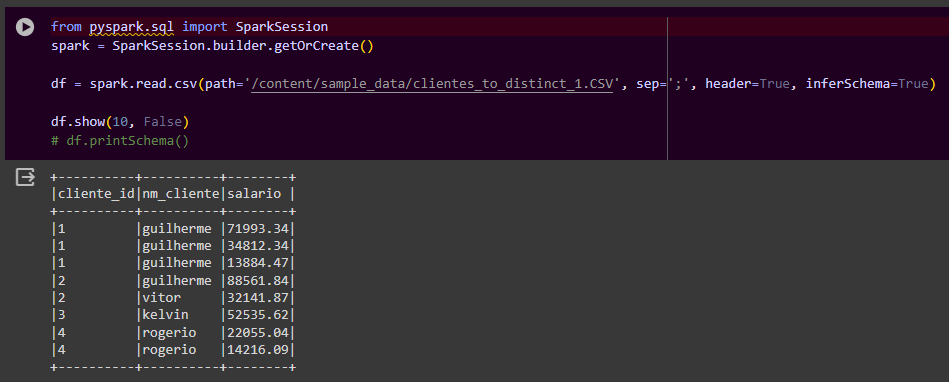


Metodo de ação - toPandas

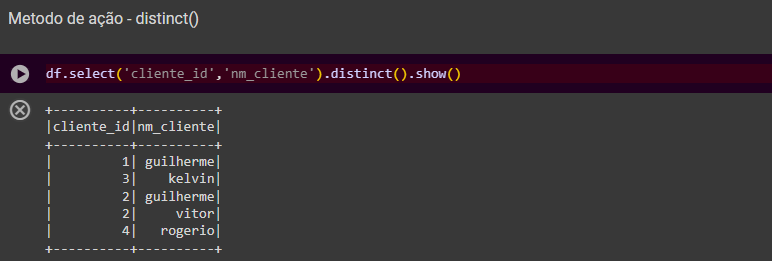
Converte um Spark Dataframe em um dataframe do Pandas.



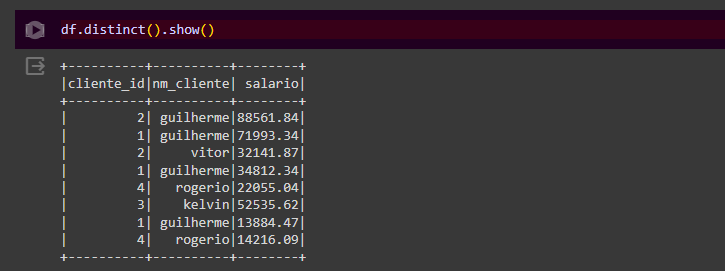
* 1. Metodos DropDuplicates VS Distinct



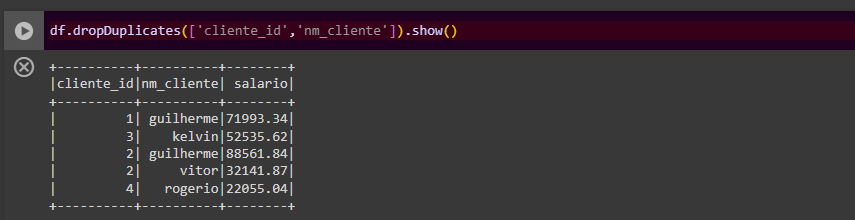
Carregamos um arquivo com dados duplicados nas primeiras colunas.



Qdo usamos o método Select para selecionar as duas primeiras colunas e incluímos o método Distinct, faz o agrupamento corretamente.

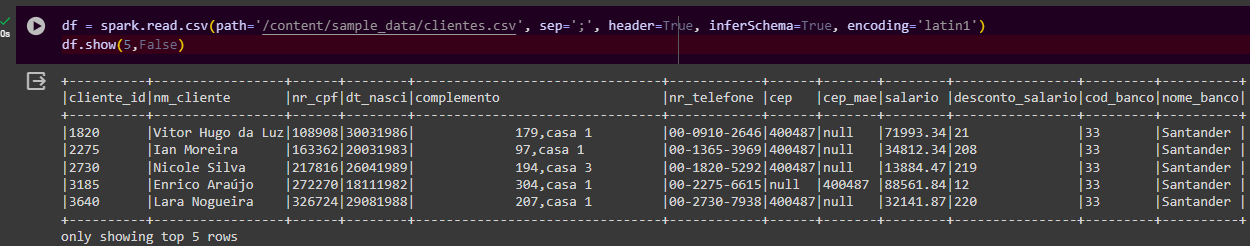


No entanto, quando fazemos o distinct no dataframe completo, não ocorre o agrupamento, pois o campo salário tem valores diversos.

Para resolver isso, usa-se o método DROPDUPLICATES para excluir as linhas duplicadas.

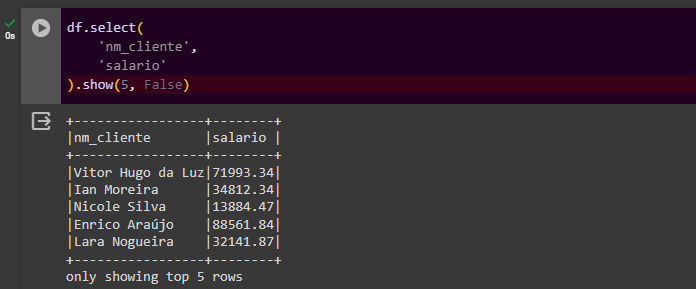
1. Colunas
   1. Trabalhando com Colunas

Vamos criar um dataframe



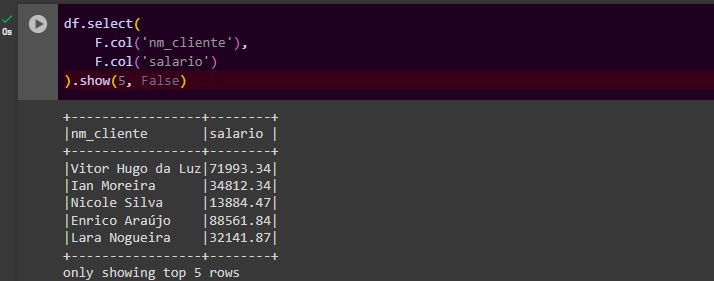
Método SELECT recebe de argumento dois tipos de parâmetros:

1. O nome da coluna em uma string.



1. Um objeto de coluna.

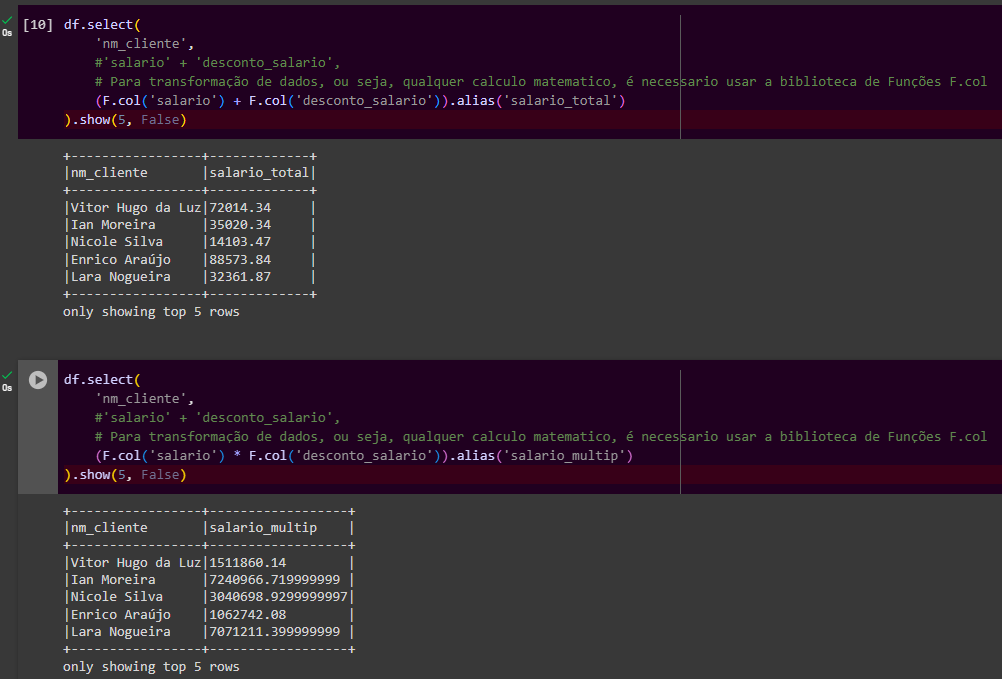
Para criar um objeto de coluna, usamos sempre os métodos da biblioteca Functions.



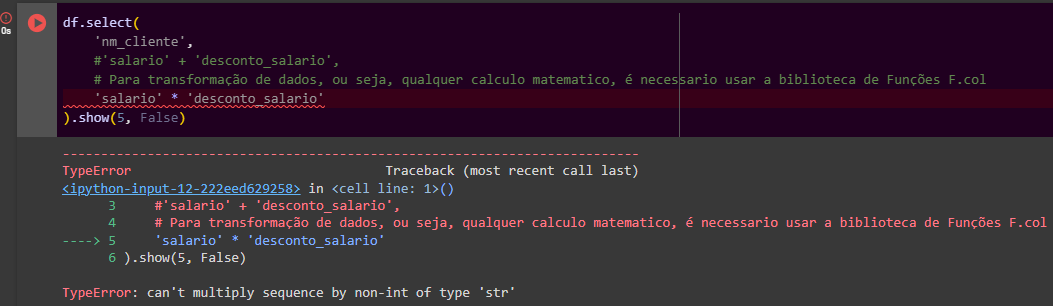
F.col(\*) – Retorna um objeto de coluna. Este objeto aponta para alguma coluna do dataframe.

Atenção: Priorize o uso de objeto de coluna em vez de passar uma string, pois é mais comum os métodos do dataframe aceitarem objetos de colunas.

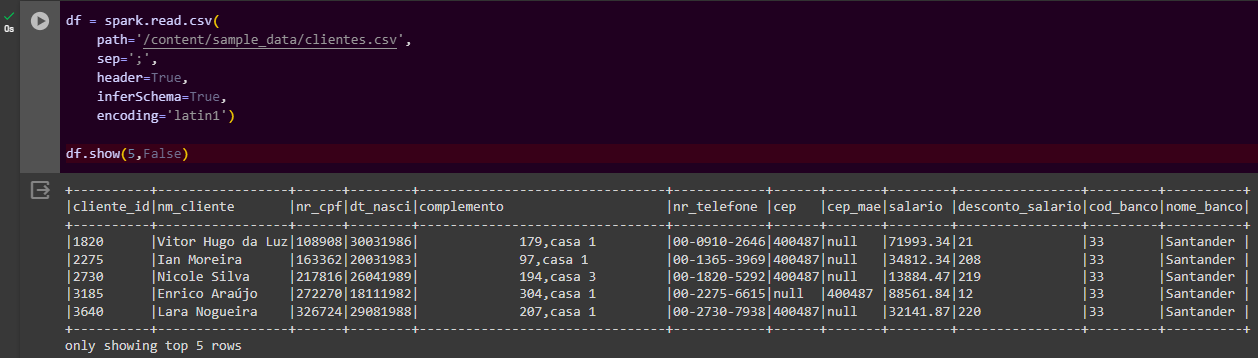
Os métodos de transformação usam mais um objeto de coluna que string, pois no objeto é mais adequado para fazer operações matemática.



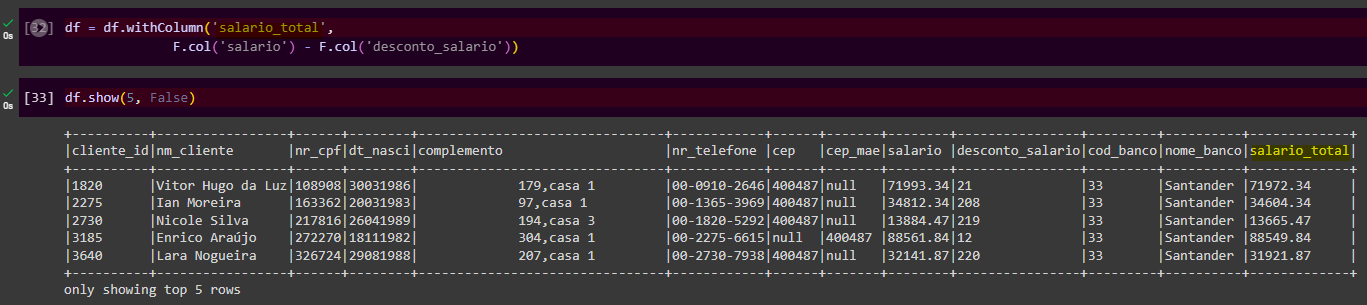
Principal motivo de não usar string é que ocorre erro nas operações matemática com colunas em string, pois faz a concatenação dos campos erroneamente.



* 1. Como criar uma nova colunas com withColumn



O método WITHCOLUMN é responsável por criar uma nova coluna e o retorno dela é um novo dataframe com essa coluna.



* 1. Métodos Colunas String – 39min

ok

* 1. Trabalhando com colunas Datetime – 25min

ok

1. Trabalhando com dataframe
   1. Aprendendo métodos de dataframe – Parte 1 – 2hs

ok

* 1. Aprendendo métodos de dataframe – Parte 2 - 1,4

Ok

1. Trabalhando com Booleanos em Colunas – 1:53 + 2hs
2. Window Function – 1h
3. Performance e Tunning – 1h
4. Pipeline – 1:43 + 1:44