

ANÁLISE DE DADOS COM PANDAS

INTRODUÇÃO AO PANDAS

Neste vídeo, a expert [Patrícia Pampanelli](#), Deep Learning Solutions Architect da NVIDIA, apresenta os conceitos iniciais da biblioteca Padas e detalha o processo necessário para a sua importação.



05:04



Até agora neste curso você aprendeu um pouco sobre as estruturas nativas do Python. Mas nem sempre vamos precisar fazer tudo do zero: para muitos dos problemas que existem, a comunidade já criou alguma ferramenta para ajudar com a solução. Uma dessas ferramentas é o Pandas que vamos ver agora.

Essa é uma introdução bem superficial sobre o que essa biblioteca pode fazer, só para dar um gostinho de sair do básico de Python e ver uma aplicação real.

Pandas é uma das bibliotecas mais populares para **Data Science**. Esta biblioteca tem bastante recurso para leitura, manipulação e visualização de dados.

ANÁLISE DE DADOS COM PANDAS

necessidade de uma ferramenta flexível e de alta performance para análises qualitativas de dados financeiros.

Em 2011, ele publica um paper intitulado "*pandas: a Foundational Python Library for Data Analysis and Statistics*" onde consolida sua visão, explica a origem e lista algumas das principais funcionalidades.

OK, MAS POR QUE O NOME PANDAS?

Logo na introdução do artigo, Wes explica que o nome da biblioteca vem de **Panel Data**, um termo atribuído a datasets multidimensionais comuns em econometria e estatística. Um exemplo clássico de um painel de dados, é um dataset de preços de ações. Nele é possível observar diversas variáveis e como elas se comportam diante de uma ou mais dimensões. Geralmente, essa dimensão é o tempo.

Utilizando a biblioteca Pandas você pode importar dados em diversos formatos (csv, xls, html, json, etc...), tratar esses dados, além de fazer transformações que irão te ajudar em diversos tipos de análise.

IMPORTANDO BIBLIOTECAS EM PYTHON

Para utilizar a biblioteca Pandas você primeiro precisa fazer sua instalação. Para nossa sorte, o Colab já vem com ela instalada, então podemos apenas importá-la.

A importação de bibliotecas em Python é feita utilizando o comando import:

```
import pandas
```

Para facilitar a utilização desta biblioteca você pode criar uma versão encurtada do nome da biblioteca. Vocês vão observar que o nome '**pd**' é o mais utilizado:

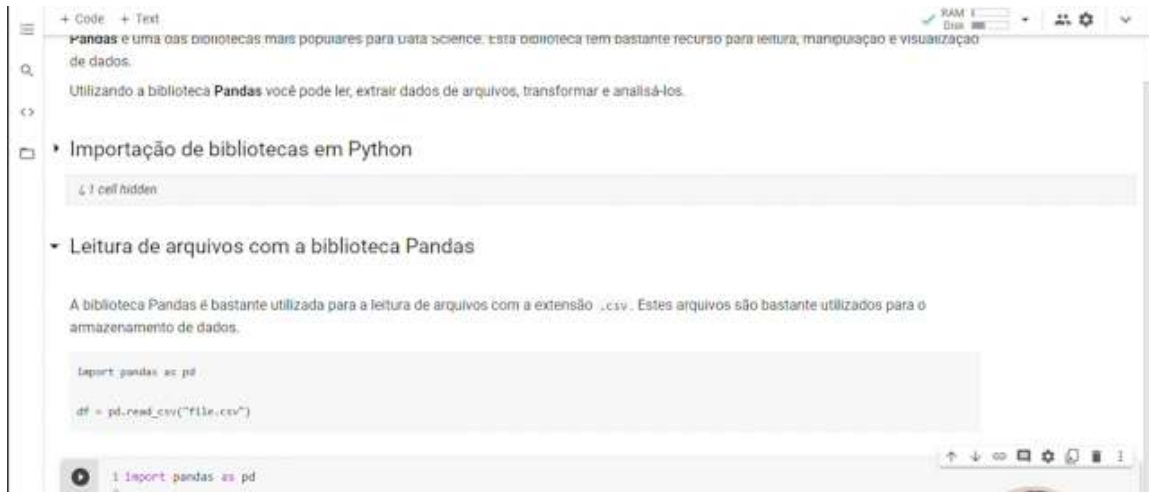
ANÁLISE DE DADOS COM PANDAS

AVANÇAR

ANÁLISE DE DADOS COM PANDAS

PANDAS SERIES E PANDAS DATAFRAMES

Nesta parte, Patrícia apresenta os dois principais objetos no Pandas: Pandas Series e Pandas DataFrame.



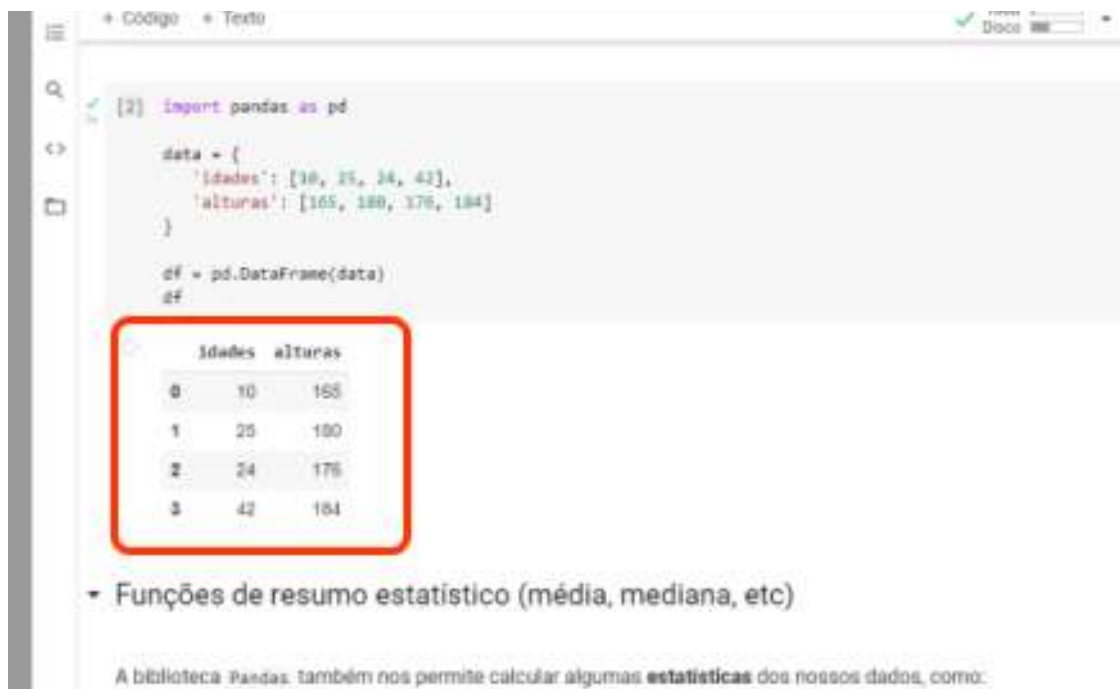
PANDAS SERIES E PANDAS DATAFRAMES

Esses são os dois principais tipos de objetos em pandas.

Podemos fazer uma analogia com uma planilha comum: o DataFrame seria a nossa planilha completa e uma Series seria uma coluna desta planilha.

No exemplo abaixo, temos um **DataFrame** com duas **Series** (idades e altura)

ANÁLISE DE DADOS COM PANDAS




Vamos ver como criar essas estruturas no Python.

LEITURA DE ARQUIVOS COM A BIBLIOTECA PANDAS

Um dos formatos mais comuns que podemos encontrar dados estruturados é o formato .csv. Vamos ver como fazer o carregamento de um arquivo no Colab e em seguida como fazer a leitura dele no pandas.

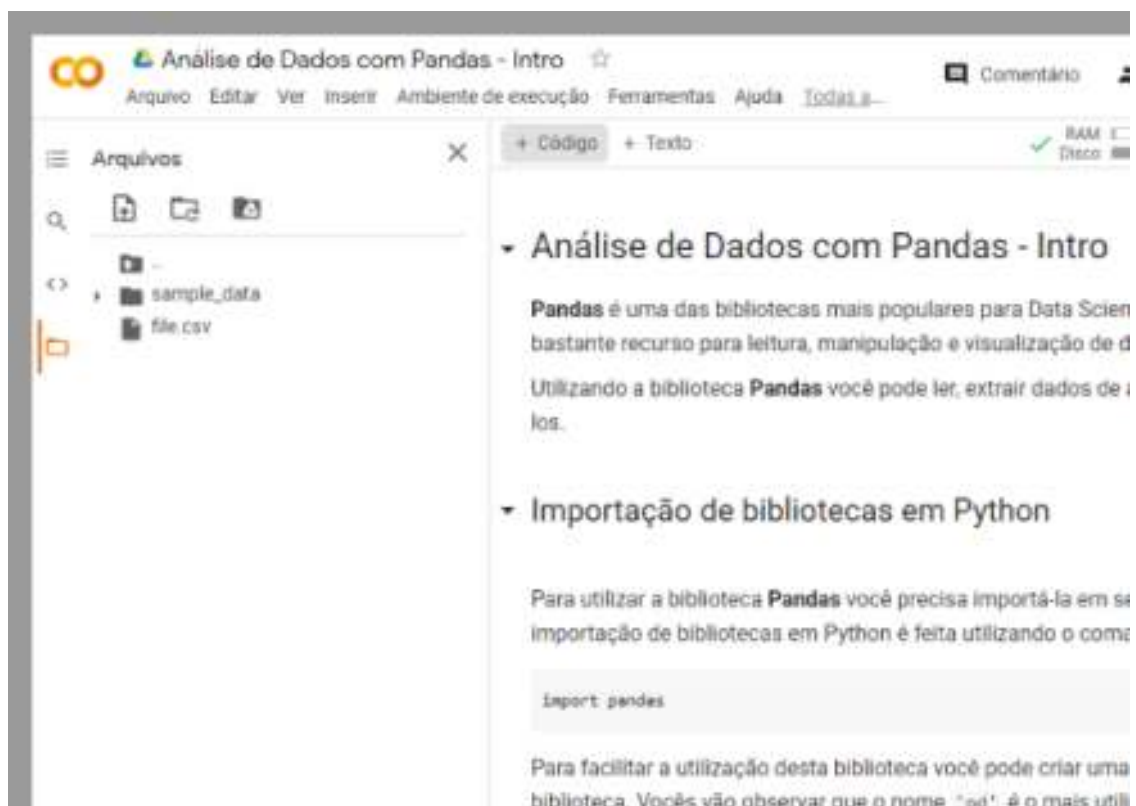
Faça o download do arquivo clicando [neste link](#).

Agora no Colab clique no ícone de pasta() para mostrar o menu lateral. Você verá um menu como este:

ANÁLISE DE DADOS COM PANDAS



Agora clique no primeiro ícone abaixo da palavra **Arquivos** e faça o upload do arquivo 'file.csv' que baixou anteriormente. Assim que terminar o upload, você o verá na estrutura de arquivos:



Lembre-se que todos os arquivos que você subir no Google colab serão apagados assim que a sua sessão se encerrar. Para fazer a leitura basta chamar a função do **.read_csv()** do pandas da seguinte forma.

ANÁLISE DE DADOS COM PANDAS

Porém o que fizemos no código anterior fez somente a leitura, precisamos guardar este objeto numa variável para que possamos fazer a manipulação desses dados. Vamos guardar então numa variável chamada **df**, mas poderíamos dar qualquer outro nome para esta variável.

```
df = pd.read_csv("file.csv")
```

Agora podemos apenas chamar a variável **df**, que veremos que os dados ficaram gravados nela.

INSERÇÃO MANUAL

Lembra dos dicionários e das listas que aprendemos nas aulas anteriores? Podemos usar essas estruturas que aprendemos para criar um DataFrame manualmente no pandas também. Vamos ver como replicar esse arquivo que importamos mas agora de maneira manual.

```
data = {  
    'idades': [10, 25, 24, 42],  
    'alturas': [165, 180, 176, 184]  
}  
df = pd.DataFrame(data)
```

No código acima, nós criamos um dicionário com dois elementos, a chave do primeiro é **'idades'** e o valor do primeiro é uma lista com os elementos **[10, 25, 24, 42]**. A chave do segundo elemento é **'alturas'** e o valor a seguinte lista: **[165, 180, 176, 184]**. E guardamos esse dicionário na variável **data**.

Passando essa variável que contém esse dicionário para o construtor **pd.DataFrame()**, o pandas irá criar um DataFrame igual aquele arquivo que já importamos.

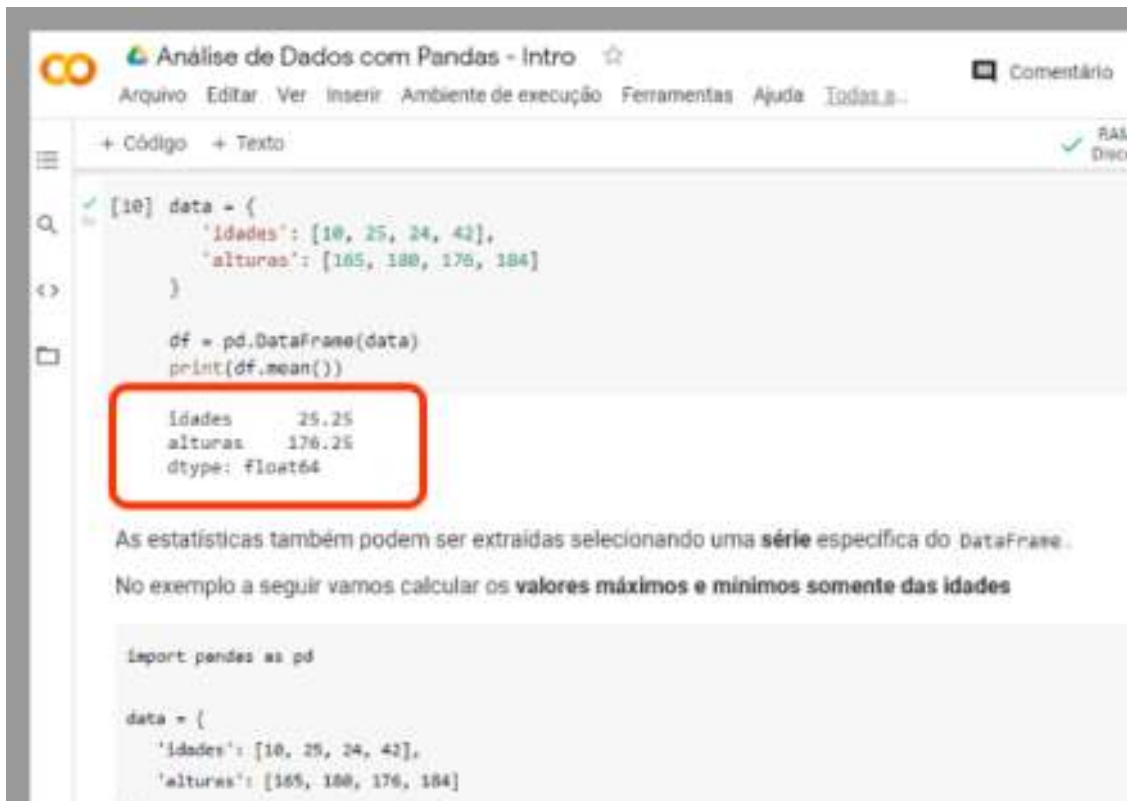
ANÁLISE DE DADOS COM PANDAS

A biblioteca Pandas também nos permite calcular algumas estatísticas dos nossos dados, como:

- média **.mean()**
- desvio padrão **.std()**
- soma **.sum()**
- valor máximo **.max()**
- valor mínimo **.min()**

Para calcular a média das Series que guardamos na variável **df** basta chamarmos:

```
df.mean()
```



```
[10] data = {  
    'idades': [10, 25, 24, 42],  
    'alturas': [165, 180, 176, 184]  
}  
  
df = pd.DataFrame(data)  
print(df.mean())  
  
idades      25.25  
alturas     176.25  
dtype: float64
```

As estatísticas também podem ser extraídas selecionando uma **série** específica do **DataFrame**.

No exemplo a seguir vamos calcular os **valores máximos e mínimos somente das idades**

```
import pandas as pd  
  
data = {  
    'idades': [10, 25, 24, 42],  
    'alturas': [165, 180, 176, 184]  
}
```


ANÁLISE DE DADOS COM PANDAS

A estrutura de dados DataFrame nos permite utilizar condicionais para filtrar algumas linhas que sejam de nosso interesse. Passando alguns operadores comparativos o Pandas irá retornar uma outra série com True para as linhas que satisfizeram a condição e False para as que não. Vamos ver um exemplo:

No exemplo a seguir nós vamos selecionar somente as linhas que contém idades maiores que 24, primeiro vamos selecionar apenas a coluna idades através do código: **df['idades']** e depois verificar quais elementos são maiores do que 24.

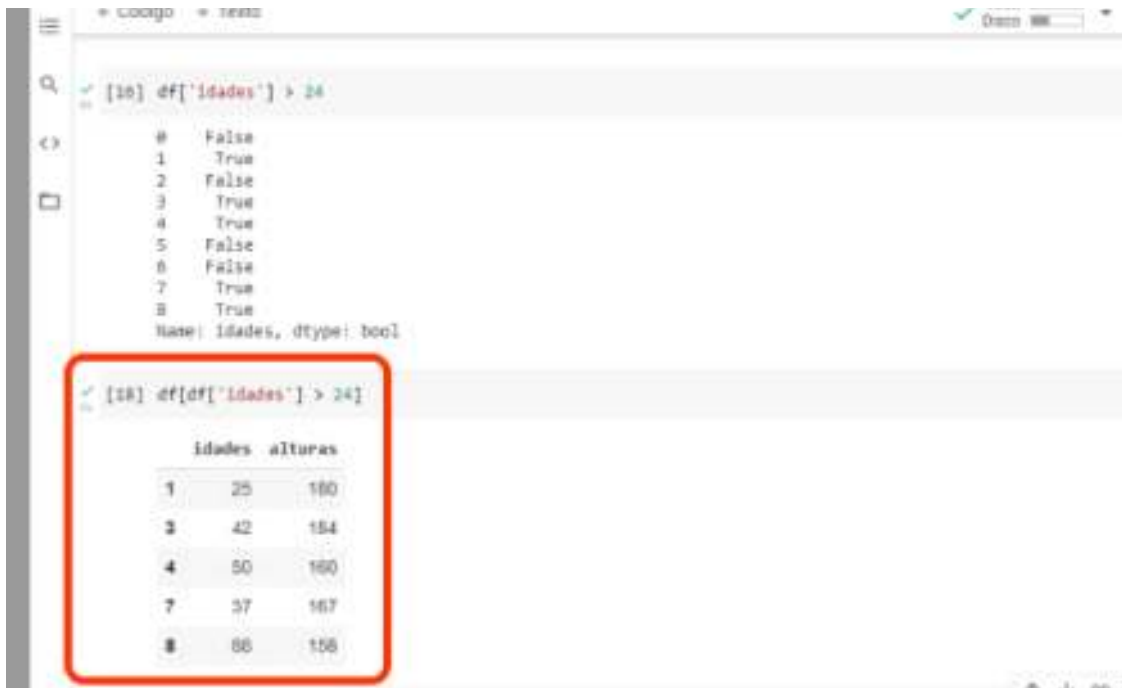
```
df['idades'] > 24
```



Com essa informação podemos passar essa série de volta para o DataFrame original para que a gente possa ver o dado filtrado, basta usar o operador colchetes.

```
df[df['idades'] > 24]
```

ANÁLISE DE DADOS COM PANDAS



VISUALIZAÇÃO DE HISTOGRAMAS COM PANDAS

A biblioteca Pandas também nos permite utilizar alguns recursos de visualização de dados. Esta é uma etapa bastante importante para Data Science. Uma das visualizações que pode ser bastante útil em um processo de **análise exploratória** de dados é o **Histograma**. Histogramas são gráficos que nos apresentam a frequência na qual determinada informação aparece em um conjunto de dados.

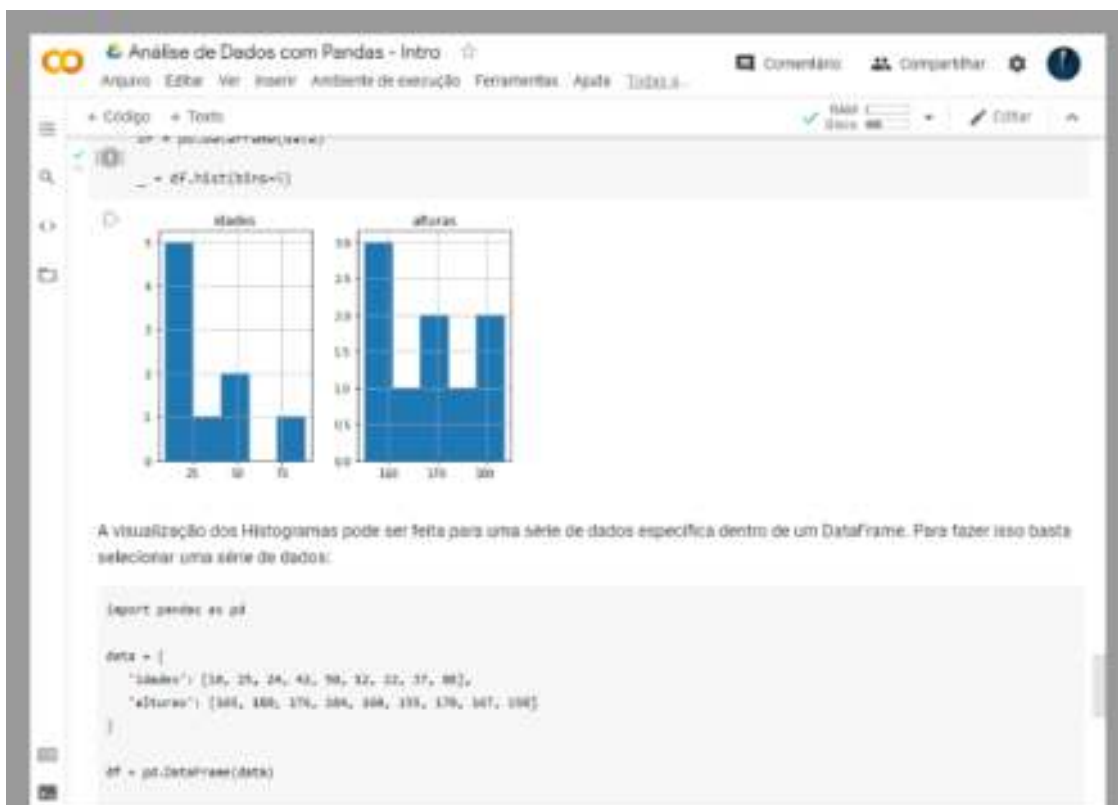
O **eixo horizontal** representa a informação que queremos avaliar ('idades' ou 'alturas') e o **eixo vertical** representa a frequência na qual esta informação aparece no DataFrame.

No exemplo a seguir estamos visualizando os Histogramas das idades e das alturas utilizando a função **hist()**:

Vamos criar um DataFrame um pouco maior para ver como exemplo:

```
data = {
    'idades': [10, 25, 24, 42, 50, 12, 22, 37, 88],
    'alturas': [165, 180, 176, 184, 160, 155, 170, 167,
158]
}
```

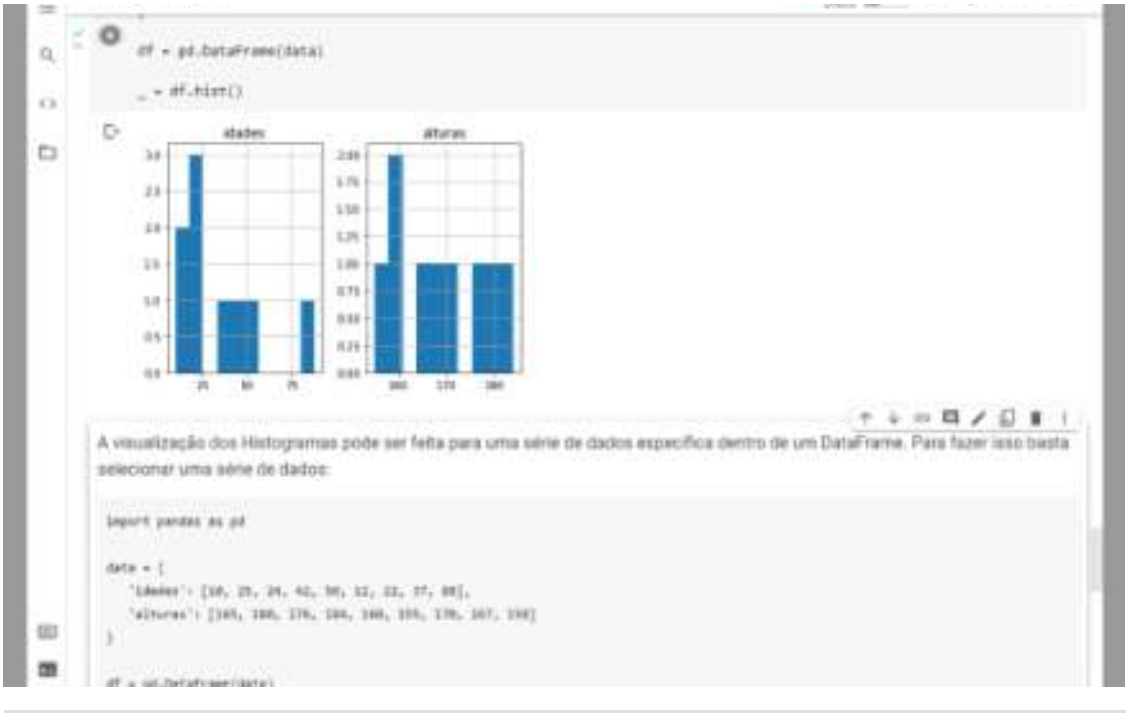
ANÁLISE DE DADOS COM PANDAS



Podemos controlar a quantidade de “caixinhas” que o histograma vai agrupar nosso dado através do parâmetro bins.

```
df.hist(bins=5)
```

ANÁLISE DE DADOS COM PANDAS



AVANÇAR