

Python para dados

Pandas

Introdução ao Pandas

O que é pandas?

O pandas, um pacote Python dedicado à manipulação de dados, também possuí funcionalidades para a visualização de dados. Sua estrutura é fundamentada em dois pacotes essenciais do Python: NumPy e Matplotlib. Enquanto o NumPy oferece objetos de matriz multidimensional para facilitar a manipulação de dados, o Matplotlib proporciona ao pandas capacidades robustas de visualização de dados.

Com uma ampla base de usuários, o pandas registrou aproximadamente 14 milhões de downloads no PyPi em dezembro de 2019. Essa cifra representa praticamente toda a comunidade de ciência de dados em Python!



O que será estudado?

Introdução ao DataFrames:

- DataFrames são o coração do pandas.

Agregação de Dados para *Insights*:

- Discussão sobre como combinar dados para uma compreensão mais aprofundada.

Fatiamento e Indexação para Subconjuntos de DataFrames:

- Exploração de técnicas para extrair partes específicas dos DataFrames por meio de fatiamento e indexação.

Visualização, Tratamento de Dados Ausentes e Leitura de Dados:

- Aprendizado sobre a visualização de dados.
- Lidar com a ausência de dados.
- Ler dados e incorporá-los em um DataFrame.





Introdução a DataFrames



Formato de dados

Guardar informações pode ser feito de muitas maneiras, mas a mais comum é como uma tabela, que chamamos de "dados retangulares" ou "dados tabulares". Vamos imaginar isso como uma folha de papel dividida em linhas e colunas. No exemplo dos cachorros, cada cachorro é uma linha, e as coisas diferentes sobre eles, como nome e cor, são colunas.

| Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento |
|--------|------------------|----------|-------------|-----------|--------------------|
| Max | Labrador | Amarelo | 60 | 30 | 01/05/2018 |
| Bella | Poodle | Branco | 35 | 7 | 10/12/2019 |
| Rocky | Boxer | Marrom | 65 | 28 | 03/08/2017 |
| Luna | Golden Retriever | Dourado | 55 | 25 | 12/02/2016 |
| Duke | Bulldog | Tigrado | 40 | 22 | 05/09/2020 |
| Daisy | Beagle | Tricolor | 33 | 10 | 08/11/2015 |
| Milo | Shih Tzu | Cinza | 25 | 5 | 07/06/2018 |
| Zoe | Dachshund | Preto | 28 | 6 | 09/04/2019 |
| Oliver | Husky | Cinza | 60 | 24 | 02/03/2016 |
| Chloe | Rottweiler | Preto | 63 | 45 | 11/10/2017 |





Criando um dataframe



Utilizando listas e dicionários

Podemos criar dataframes de várias formas, umas das mais simples é utilizando dicionários e listas:

```
1 dados = {  
2     'Nome': ['Max', 'Bella', 'Rocky', 'Luna'],  
3     'Raça': ['Labrador', 'Poodle', 'Boxer', 'Golden Retriever'],  
4     'Cor': ['Amarelo', 'Branco', 'Marrom', 'Dourado'],  
5     'Peso (kg)': [60, 35, 65, 55],  
6     'Altura (cm)': [30, 7, 28, 25],  
7     'Data de Nascimento': ['01/05/2018', '10/12/2019', '03/08/2017', '12/02/2016']  
8 }
```



Utilizando listas e dicionários

Podemos criar dataframes de várias formas, umas das mais simples é utilizando dicionários e listas:

```
1 import pandas as pd\n\n1 dados = {\n2     'Nome': ['Max', 'Bella', 'Rocky', 'Luna'],\n3     'Raça': ['Labrador', 'Poodle', 'Boxer', 'Golden Retriever'],\n4     'Cor': ['Amarelo', 'Branco', 'Marrom', 'Dourado'],\n5     'Peso (kg)': [60, 35, 65, 55],\n6     'Altura (cm)': [30, 7, 28, 25],\n7     'Data de Nascimento': ['01/05/2018', '10/12/2019', '03/08/2017', '12/02/2016']\n8 }\n9\n10 # Criar o DataFrame\n11 df_cachorros = pd.DataFrame(dados)
```



Utilizando listas e dicionários

Podemos criar dataframes de várias formas, uma das mais simples é utilizando dicionários e listas:

```
1 import pandas as pd\n\n1 dados = {\n2     'Nome': ['Max', 'Bella', 'Rocky', 'Luna'],\n3     'Raça': ['Labrador', 'Poodle', 'Boxer', 'Golden Retriever'],\n4     'Cor': ['Amarelo', 'Branco', 'Marrom', 'Dourado'],\n5     'Peso (kg)': [60, 35, 65, 55],\n6     'Altura (cm)': [30, 7, 28, 25],\n7     'Data de Nascimento': ['01/05/2018', '10/12/2019', '03/08/2017', '12/02/2016']\n8 }\n9\n10 # Criar o DataFrame\n11 df_cachorros = pd.DataFrame(dados)\n12 df_cachorros
```

| | Nome | Raça | Cor | Peso (kg) | Altura (cm) | Data de Nascimento | |
|---|-------|------------------|---------|-----------|-------------|--------------------|--|
| 0 | Max | Labrador | Amarelo | 60 | 30 | 01/05/2018 | |
| 1 | Bella | Poodle | Branco | 35 | 7 | 10/12/2019 | |
| 2 | Rocky | Boxer | Marrom | 65 | 28 | 03/08/2017 | |
| 3 | Luna | Golden Retriever | Dourado | 55 | 25 | 12/02/2016 | |





Lendo um arquivo e transformando-os em dataframe



Existem vários tipos de arquivos que podem ser lidos diretamente em um dataframe...

```
1
2 import pandas as pd
3
4 #csv
5 df_csv = pd.read_csv('arquivo.csv')
6
7 #excel
8 df_excel = pd.read_excel('planilha.xlsx', sheet_name='nome_da_planilha')
9
10 #json
11 df_json = pd.read_json('arquivo.json')
12
13 #parquet
14 df_parquet = pd.read_parquet('arquivo.parquet')
15
16 #SQL
17 from sqlalchemy import create_engine
18
19 engine = create_engine('sqlite:///banco_de_dados.db')
20 query = 'SELECT * FROM tabela'
21 df_sql = pd.read_sql(query, engine)
```





Primeiros métodos para entender seu dataframe



df.head()

Quando a gente ganha uma tabela nova, é legal dar uma olhada rápida, né?

O pandas tem um truque chamado "head" que mostra as primeiras linhas da tabela. Se tiver muitas linhas, isso ajuda bastante!

| 1 df_cachorros.head() | | | | | | |
|-----------------------|-------|------------------|---------|-------------|-----------|--------------------|
| | Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento |
| 0 | Max | Labrador | Amarelo | 60 | 30 | 01/05/2018 |
| 1 | Bella | Poodle | Branco | 35 | 7 | 10/12/2019 |
| 2 | Rocky | Boxer | Marrom | 65 | 28 | 03/08/2017 |
| 3 | Luna | Golden Retriever | Dourado | 55 | 25 | 12/02/2016 |
| 4 | Duke | Bulldog | Tigrado | 40 | 22 | 05/09/2020 |



df.info()

Tem um jeito do pandas nos contar os nomes das colunas, que são como etiquetas, e se tem alguma informação faltando, ele usa um método chamado "info" para isso.

```
1 df_cachorros.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Nome             10 non-null    object  
 1   Raça             10 non-null    object  
 2   Cor              10 non-null    object  
 3   Altura (cm)     10 non-null    int64   
 4   Peso (kg)        10 non-null    int64   
 5   Data de Nascimento  10 non-null  datetime64[ns]
dtypes: datetime64[ns](1), int64(2), object(3)
memory usage: 608.0+ bytes
```



df.shape

A tabela do pandas tem um jeito de mostrar quantas linhas e quantas colunas ela tem, como se fosse um resuminho. Não precisa usar parênteses, é só chamar "shape".

```
1 df_cachorros.shape
```

```
(10, 6)
```



df.describe()

O pandas também sabe fazer as contas! Com o "describe", ele conta coisas legais sobre os números na tabela, tipo média e mediana. Ele até fala quantos números tem em cada coluna

```
1 df_cachorros.describe()
```

| | Altura (cm) | Peso (kg) |
|--------------|-------------|-----------|
| count | 10.000000 | 10.000000 |
| mean | 46.400000 | 20.200000 |
| std | 15.678719 | 13.011106 |
| min | 25.000000 | 5.000000 |
| 25% | 33.500000 | 7.750000 |
| 50% | 47.500000 | 23.000000 |
| 75% | 60.000000 | 27.250000 |
| max | 65.000000 | 45.000000 |



df.describe()

O pandas também sabe fazer as contas! Com o "describe", ele conta coisas legais sobre os números na tabela, tipo média e mediana. Ele até fala quantos números tem em cada coluna

```
1 df_cachorros.describe()
```

| | Altura (cm) | Peso (kg) |
|--------------|-------------|-----------|
| count | 10.000000 | 10.000000 |
| mean | 46.400000 | 20.200000 |
| std | 15.678719 | 13.011106 |
| min | 25.000000 | 5.000000 |
| 25% | 33.500000 | 7.750000 |
| 50% | 47.500000 | 23.000000 |
| 75% | 60.000000 | 27.250000 |
| max | 65.000000 | 45.000000 |





Componentes de um dataframe



Peças Especiais do DataFrame

A tabela do pandas é como um quebra-cabeça com três partes:

- .values:** os valores (que são os dados mesmo);
- .columns:** as etiquetas (labels) das colunas;
- .index:** etiquetas das linhas;

Isso ajuda a entender melhor cada pedacinho!



Peças Especiais do DataFrame

A tabela do pandas é como um quebra-cabeça com três partes:
melhor

Isso ajuda a entender
cada pedacinho!

.values: os valores (que são os dados mesmo);

.columns: as etiquetas (labels) das colunas;

.index: etiquetas das linhas;

df.values

```
1 df_cachorros.values

array([['Max', 'Labrador', 'Amarelo', 60, 30, '01/05/2018'],
       ['Bella', 'Poodle', 'Branco', 35, 7, '10/12/2019'],
       ['Rocky', 'Boxer', 'Marrom', 65, 28, '03/08/2017'],
       ['Luna', 'Golden Retriever', 'Dourado', 55, 25, '12/02/2016'],
       ['Duke', 'Bulldog', 'Tigrado', 40, 22, '05/09/2020'],
       ['Daisy', 'Beagle', 'Tricolor', 33, 10, '08/11/2015'],
       ['Milo', 'Shih Tzu', 'Cinza', 25, 5, '07/06/2018'],
       ['Zoe', 'Dachshund', 'Preto', 28, 6, '09/04/2019'],
       ['Oliver', 'Husky', 'Cinza', 60, 24, '02/03/2016'],
       ['Chloe', 'Rottweiler', 'Preto', 63, 45, '11/10/2017']],
      dtype=object)
```



Peças Especiais do DataFrame

A tabela do pandas é como um quebra-cabeça com três partes:
melhor

Isso ajuda a entender
cada pedacinho!

.values: os valores (que são os dados mesmo);

.columns: as etiquetas (labels) das colunas;

.index: etiquetas das linhas;

df.columns

```
1 df_cachorros.columns
```

```
Index(['Nome', 'Raça', 'Cor', 'Altura (cm)', 'Peso (kg)',  
       'Data de Nascimento'],  
      dtype='object')
```



Peças Especiais do DataFrame

A tabela do pandas é como um quebra-cabeça com três partes:
melhor

Isso ajuda a entender
cada pedacinho!

.values: os valores (que são os dados mesmo);

.columns: as etiquetas (labels) das colunas;

.index: etiquetas das linhas;

df.index

```
1 df_cachorros.index
```

```
RangeIndex(start=0, stop=10, step=1)
```





A filosofia Python de que deveria **existir apenas uma solução óbvia** para um certo problema de programação é desafiada com o Pandas, já que a biblioteca nos traz várias formas de resolver um só problema





Ordenação e subconjunto



Você pode ordenar as linhas usando o método `sort_values`, passando o nome da coluna pela qual você deseja ordenar...

Ordenação

`df.sort_values('coluna')`

| | Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento |
|---|--------|------------------|----------|-------------|-----------|--------------------|
| 6 | Milo | Shih Tzu | Cinza | 25 | 5 | 07/06/2018 |
| 7 | Zoe | Dachshund | Preto | 28 | 6 | 09/04/2019 |
| 5 | Daisy | Beagle | Tricolor | 33 | 10 | 08/11/2015 |
| 1 | Bella | Poodle | Branco | 35 | 7 | 10/12/2019 |
| 4 | Duke | Bulldog | Tigrado | 40 | 22 | 05/09/2020 |
| 3 | Luna | Golden Retriever | Dourado | 55 | 25 | 12/02/2016 |
| 0 | Max | Labrador | Amarelo | 60 | 30 | 01/05/2018 |
| 8 | Oliver | Husky | Cinza | 60 | 24 | 02/03/2016 |
| 9 | Chloe | Rottweiler | Preto | 63 | 45 | 11/10/2017 |
| 2 | Rocky | Boxer | Marrom | 65 | 28 | 03/08/2017 |



Definir o argumento ascending como False ordenará os dados de forma inversa, do cachorro mais pesado para o cachorro mais leve...

Ordenação

`df.sort_values('coluna', ascending=False)`

```
1 df_cachorros.sort_values('Altura (cm)', ascending=False)
```

| | Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento |
|---|--------|------------------|----------|-------------|-----------|--------------------|
| 2 | Rocky | Boxer | Marrom | 65 | 28 | 03/08/2017 |
| 9 | Chloe | Rottweiler | Preto | 63 | 45 | 11/10/2017 |
| 0 | Max | Labrador | Amarelo | 60 | 30 | 01/05/2018 |
| 8 | Oliver | Husky | Cinza | 60 | 24 | 02/03/2016 |
| 3 | Luna | Golden Retriever | Dourado | 55 | 25 | 12/02/2016 |
| 4 | Duke | Bulldog | Tigrado | 40 | 22 | 05/09/2020 |
| 1 | Bella | Poodle | Branco | 35 | 7 | 10/12/2019 |
| 5 | Daisy | Beagle | Tricolor | 33 | 10 | 08/11/2015 |
| 7 | Zoe | Dachshund | Preto | 28 | 6 | 09/04/2019 |
| 6 | Milo | Shih Tzu | Cinza | 25 | 5 | 07/06/2018 |



Podemos ordenar por várias variáveis passando uma lista de nomes de colunas para `sort_values`. Aqui, ordenamos primeiro por peso e depois por altura...

Ordenação por múltiplas variáveis

```
df.sort_values(['Altura (cm)', 'Peso (kg)'])
```

| | Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|---|--------|------------------|----------|-------------|-----------|--------------------|------------|
| 6 | Milo | Shih Tzu | Cinza | 25 | 5 | 2018-07-06 | 0.25 |
| 7 | Zoe | Dachshund | Preto | 28 | 6 | 2019-09-04 | 0.28 |
| 5 | Daisy | Beagle | Tricolor | 33 | 10 | 2015-08-11 | 0.33 |
| 1 | Bella | Poodle | Branco | 35 | 7 | 2019-10-12 | 0.35 |
| 4 | Duke | Bulldog | Tigrado | 40 | 22 | 2020-05-09 | 0.40 |
| 3 | Luna | Golden Retriever | Dourado | 55 | 25 | 2016-12-02 | 0.55 |
| 8 | Oliver | Husky | Cinza | 60 | 24 | 2016-02-03 | 0.60 |
| 0 | Max | Labrador | Amarelo | 60 | 30 | 2018-01-05 | 0.60 |
| 9 | Chloe | Rottweiler | Preto | 63 | 45 | 2017-11-10 | 0.63 |
| 2 | Rocky | Boxer | Marrom | 65 | 28 | 2017-03-08 | 0.65 |



Para alterar a direção da ordenação, passe uma lista para o argumento `ascending` para especificar em qual direção a ordenação deve ser feita para cada variável...

Ordenação por múltiplas variáveis

```
df.sort_values(['Altura (cm)', 'Peso (kg)'], ascending=[True, False])
```

| | Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) | grid icon | info icon |
|---|--------|------------------|----------|-------------|-----------|--------------------|------------|-----------|-----------|
| 6 | Milo | Shih Tzu | Cinza | 25 | 5 | 2018-07-06 | 0.25 | | |
| 7 | Zoe | Dachshund | Preto | 28 | 6 | 2019-09-04 | 0.28 | | |
| 5 | Daisy | Beagle | Tricolor | 33 | 10 | 2015-08-11 | 0.33 | | |
| 1 | Bella | Poodle | Branco | 35 | 7 | 2019-10-12 | 0.35 | | |
| 4 | Duke | Bulldog | Tigrado | 40 | 22 | 2020-05-09 | 0.40 | | |
| 3 | Luna | Golden Retriever | Dourado | 55 | 25 | 2016-12-02 | 0.55 | | |
| 0 | Max | Labrador | Amarelo | 60 | 30 | 2018-01-05 | 0.60 | | |
| 8 | Oliver | Husky | Cinza | 60 | 24 | 2016-02-03 | 0.60 | | |
| 9 | Chloe | Rottweiler | Preto | 63 | 45 | 2017-11-10 | 0.63 | | |
| 2 | Rocky | Boxer | Marrom | 65 | 28 | 2017-03-08 | 0.65 | | |



Subconjunto de colunas

Podemos querer focar em apenas uma coluna

```
1 df_cachorros['Nome']
```

| | |
|---|--------|
| 0 | Max |
| 1 | Bella |
| 2 | Rocky |
| 3 | Luna |
| 4 | Duke |
| 5 | Daisy |
| 6 | Milo |
| 7 | Zoe |
| 8 | Oliver |
| 9 | Chloe |

Name: Nome, dtype: object

Subconjunto de várias colunas

Para selecionar várias colunas, você precisa de dois pares de colchetes.

```
1 df_cachorros[['Nome', 'Altura (cm)']]
```

| | Nome | Altura (cm) |
|---|--------|-------------|
| 0 | Max | 60 |
| 1 | Bella | 35 |
| 2 | Rocky | 65 |
| 3 | Luna | 55 |
| 4 | Duke | 40 |
| 5 | Daisy | 33 |
| 6 | Milo | 25 |
| 7 | Zoe | 28 |
| 8 | Oliver | 60 |
| 9 | Chloe | 63 |



Subconjunto de linhas

Existem muitas maneiras diferentes de fazer subconjunto de linhas. A maneira mais comum de fazer isso é criando uma condição lógica para filtrar.

```
1 df_cachorros['Altura (cm)'] > 40  
0    True  
1   False  
2    True  
3    True  
4   False  
5   False  
6   False  
7   False  
8    True  
9    True  
Name: Altura (cm), dtype: bool
```

Podemos usar a condição lógica dentro de colchetes para fazer subconjunto de linhas que nos interessam.

```
1 df_cachorros[df_cachorros['Altura (cm)'] > 40]  
   Nome          Raça      Cor  Altura (cm)  Peso (kg) Data de Nascimento  
0   Max        Labrador  Amarelo       60        30     01/05/2018  
2  Rocky         Boxer  Marrom       65        28     03/08/2017  
3   Luna  Golden Retriever  Dourado       55        25    12/02/2016  
8  Oliver        Husky   Cinza       60        24    02/03/2016  
9  Chloe        Rottweiler  Preto       63        45    11/10/2017
```



Subconjunto com base em dados de texto

Também podemos fazer subconjuntos de linhas com base em dados de **texto**.

```
1 df_cachorros[df_cachorros['Raça'] == 'Boxer']
```

| Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento |
|---------|-------|--------|-------------|-----------|--------------------|
| 2 Rocky | Boxer | Marrom | 65 | 28 | 03/08/2017 |

Subconjunto com base em datas

Também podemos fazer subconjuntos de linhas com base em **datas**.



```
1 df_cachorros[df_cachorros['Data de Nascimento'] < '12/05/2017']
```

| | Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento |
|---|--------|------------------|----------|-------------|-----------|--------------------|
| 2 | Rocky | Boxer | Marrom | 65 | 28 | 2017-03-08 |
| 3 | Luna | Golden Retriever | Dourado | 55 | 25 | 2016-12-02 |
| 5 | Daisy | Beagle | Tricolor | 33 | 10 | 2015-08-11 |
| 8 | Oliver | Husky | Cinza | 60 | 24 | 2016-02-03 |
| 9 | Chloe | Rottweiler | Preto | 63 | 45 | 2017-11-10 |



Subconjunto com base em múltiplas condições

Para fazer um subconjunto das linhas que atendem a várias condições, você pode combinar condições usando operadores lógicos, como o operador "e" visto aqui. Isso significa que apenas as linhas que atendem a ambas as condições serão subconjugadas. Você também poderia fazer isso em uma única linha de código, mas também precisará adicionar parênteses em torno de cada condição.

```
1 is_husky = df_cachorros["Raça"] == 'Husky'  
2 is_preto = df_cachorros["Cor"] == 'Cinza'  
3 df_cachorros[is_husky & is_preto]
```

| | Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento | |
|---|--------|-------|-------|-------------|-----------|--------------------|------------|
| 8 | Oliver | Husky | Cinza | | 60 | 24 | 2016-02-03 |

```
1 df_cachorros[(df_cachorros["Raça"] == 'Husky') & (df_cachorros["Cor"] == 'Cinza')]
```

| | Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento | grid icon |
|---|--------|-------|-------|-------------|-----------|--------------------|------------|
| 8 | Oliver | Husky | Cinza | | 60 | 24 | 2016-02-03 |



Subconjunto usando .isin()

Se você quiser filtrar com base em múltiplos valores de uma variável categórica, a maneira mais fácil é usar o método `isin`. Isso aceita uma lista de valores para filtrar.

```
1 is_marrom_ou_cinza = df_cachorros["Cor"].isin(["Cinza", "Marrom"])
2 df_cachorros[is_marrom_ou_cinza]
```

| | Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento |  |  |
|---|--------|----------|--------|-------------|-----------|--------------------|---|---|
| 2 | Rocky | Boxer | Marrom | 65 | 28 | 2017-03-08 | | |
| 6 | Milo | Shih Tzu | Cinza | 25 | 5 | 2018-07-06 | | |
| 8 | Oliver | Husky | Cinza | 60 | 24 | 2016-02-03 | | |





Adicionando colunas



Adicionando uma nova coluna

Criar e adicionar novas colunas pode ter vários nomes, incluindo mutação de um DataFrame, transformação de um DataFrame e engenharia de características. Digamos que queiramos adicionar uma nova coluna ao nosso DataFrame com a altura de cada cachorro em metros, em vez de centímetros. No lado esquerdo do sinal de igual, usamos colchetes com o nome da nova coluna que queremos criar. No lado direito, temos o cálculo. Observe que tanto a coluna existente quanto a nova coluna que acabamos de criar estão no DataFrame.

```
1 df_cachorros['Altura (m)'] = df_cachorros['Altura (cm)'] / 100
2 df_cachorros
```

| | Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|---|-------|------------------|---------|-------------|-----------|--------------------|------------|
| 0 | Max | Labrador | Amarelo | 60 | 30 | 2018-01-05 | 0.60 |
| 1 | Bella | Poodle | Branco | 35 | 7 | 2019-10-12 | 0.35 |
| 2 | Rocky | Boxer | Marrom | 65 | 28 | 2017-03-08 | 0.65 |
| 3 | Luna | Golden Retriever | Dourado | 55 | 25 | 2016-12-02 | 0.55 |
| 4 | Duke | Bulldog | Tigrado | 40 | 22 | 2020-05-09 | 0.40 |



Múltiplas manipulações

O verdadeiro poder do pandas se revela quando você combina todas as habilidades que aprendeu até agora.

Vamos descobrir qual o peso dos cães com pelo menos 0.5 metros de altura, retorne o nome, altura em cm também.

```
1 dog_pequeno = df_cachorros[df_cachorros['Altura (m)'] <= 0.5]
2 dog_pequeno = dog_pequeno.sort_values('Altura (cm)', ascending=False)
3 dog_pequeno[['Nome', 'Altura (cm)', 'Peso (kg)']]
```

| | Nome | Altura (cm) | Peso (kg) |  |
|---|-------|-------------|-----------|--|
| 4 | Duke | 40 | 22 |  |
| 1 | Bella | 35 | 7 | |
| 5 | Daisy | 33 | 10 | |
| 7 | Zoe | 28 | 6 | |
| 6 | Milo | 25 | 5 | |



HORA DA PRÁTICA

Agregando datos

Extraindo estatísticas

Estatísticas resumidas, como o próprio nome sugere, são números que resumem e fornecem informações sobre seu conjunto de dados...

- **.median()**: Calcula a mediana de um conjunto de dados, indicando o valor central quando os dados estão ordenados.
- **.mode()**: Identifica o(s) valor(es) mais frequente(s) em um conjunto de dados.
- **.min()**: Retorna o valor mínimo em um conjunto de dados.
- **.max()**: Retorna o valor máximo em um conjunto de dados.
- **.var()**: Calcula a variância, uma medida da dispersão dos dados.
- **.std()**: Calcula o desvio padrão, indicando a dispersão média dos dados em relação à média.
- **.sum()**: Soma todos os valores em um conjunto de dados.
- **.quantile()**: Calcula um percentil específico em um conjunto de dados, indicando o valor abaixo do qual uma determinada porcentagem dos dados está.



Estatísticas resumidas, como o próprio nome sugere, são números que resumem e fornecem informações sobre seu conjunto de dados...

```
1 df_cachorros['Altura'].min()
```

5

```
1 df_cachorros['Altura'].max()
```

45

```
1 df_cachorros['Nome'].max()
```

'Zoe'



O método de agregação, ou agg, permite calcular estatísticas personalizadas...

```
1 def pct50(coluna):  
2     return coluna.quantile(0.50)
```

```
1 df_cachorros['Altura (cm)'].agg(pct50)
```

23.0

```
1 df_cachorros[['Altura (cm)', 'Peso (kg)']].agg(pct50)
```

```
Altura (cm)    23.0  
Peso (kg)      47.5  
dtype: float64
```

```
1 def max_min(coluna):  
2     return coluna.max(), coluna.min()
```

```
1 df_cachorros[['Altura (cm)', 'Peso (kg)']].agg(max_min)
```

| | Altura (cm) | Peso (kg) | grid icon |
|---|-------------|-----------|----------------|
| 0 | 45 | 65 | bar chart icon |
| 1 | 5 | 25 | |



O pandas também possui métodos para calcular estatísticas cumulativas. Chamar "cumsum" em uma coluna não retorna apenas um número, mas um número para cada linha do DataFrame...

O primeiro número
retornado, ou o número no
índice zero, é o peso do
primeiro cachorro. O próximo
número é a soma do peso
do primeiro e segundo
cachorros. O terceiro número
é a soma do peso do
primeiro, segundo e terceiro
cachorros, e assim por
diante. O último número é a
soma de todos os pesos dos
cachorros.

```
1 df_cachorros["Peso (kg)"]
```

| | |
|---|----|
| 0 | 60 |
| 1 | 35 |
| 2 | 65 |
| 3 | 55 |
| 4 | 40 |
| 5 | 33 |
| 6 | 25 |
| 7 | 28 |
| 8 | 60 |
| 9 | 63 |

Name: Peso (kg), dtype: int64

```
1 df_cachorros["Peso (kg)"].cumsum()
```

| | |
|---|-----|
| 0 | 60 |
| 1 | 95 |
| 2 | 160 |
| 3 | 215 |
| 4 | 255 |
| 5 | 288 |
| 6 | 313 |
| 7 | 341 |
| 8 | 401 |
| 9 | 464 |

Name: Peso (kg), dtype: int64



```
1 df_cachorros["Peso (kg)"]
```

```
0    60  
1    35  
2    65  
3    55  
4    40  
5    33  
6    25  
7    28  
8    60  
9    63
```

Name: Peso (kg), dtype: int64

```
1 df_cachorros["Peso (kg)"].cumsum()
```

```
0    60  
1    95  
2   160  
3   215  
4   255  
5   288  
6   313  
7   341  
8   401  
9   464
```

Name: Peso (kg), dtype: int64

O pandas também possui métodos para calcular estatísticas cumulativas. Chamar "cumsum" em uma coluna não retorna apenas um número, mas um número para cada linha do DataFrame...

O primeiro número retornado, ou o número no índice zero, é o peso do primeiro cachorro. O próximo número é a soma do peso do primeiro e segundo cachorros. O terceiro número é a soma do peso do primeiro, segundo e terceiro cachorros, e assim por diante. O último número é a soma de todos os pesos dos cachorros.





O pandas também possui métodos para outras estatísticas cumulativas, como o máximo cumulativo, mínimo cumulativo e o produto cumulativo. Todos eles retornam uma coluna inteira de um DataFrame, em vez de um único número.

- **.cumsum():** Calcula a soma cumulativa, fornecendo uma série de valores acumulados à medida que percorre um conjunto de dados.
- **.cummax():** Calcula o máximo cumulativo, retornando a série de valores máximos à medida que avança através de um conjunto de dados.
- **.cummin():** Calcula o mínimo cumulativo, resultando na série de valores mínimos à medida que se desloca através de um conjunto de dados.
- **.cumprod():** Calcula o produto cumulativo, gerando uma série de valores multiplicados cumulativamente à medida que percorre um conjunto de dados.





O pandas também possui métodos para outras estatísticas cumulativas, como o máximo cumulativo, mínimo cumulativo e o produto cumulativo. Todos eles retornam uma coluna inteira de um DataFrame, em vez de um único número.

```
1 df_cachorros["Peso (kg)"].cummin()
```

| | |
|---|----|
| 0 | 60 |
| 1 | 35 |
| 2 | 35 |
| 3 | 35 |
| 4 | 35 |
| 5 | 33 |
| 6 | 25 |
| 7 | 25 |
| 8 | 25 |
| 9 | 25 |

Name: Peso (kg), dtype: int64

```
✓ 1 df_cachorros["Peso (kg)"].cummax()
```

| | |
|---|----|
| 0 | 60 |
| 1 | 60 |
| 2 | 65 |
| 3 | 65 |
| 4 | 65 |
| 5 | 65 |
| 6 | 65 |
| 7 | 65 |
| 8 | 65 |
| 9 | 65 |

Name: Peso (kg), dtype: int64



Lidando com valores duplicados

Adicionando mais 3 linhas

```
1 df_cachorros
```

| | Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|----|--------|------------------|----------|-------------|-----------|--------------------|------------|
| 0 | Max | Labrador | Amarelo | 30 | 60 | 01/05/2018 | 0.30 |
| 1 | Bella | Poodle | Branco | 7 | 35 | 10/12/2019 | 0.07 |
| 2 | Rocky | Boxer | Marrom | 28 | 65 | 03/08/2017 | 0.28 |
| 3 | Luna | Golden Retriever | Dourado | 25 | 55 | 12/02/2016 | 0.25 |
| 4 | Duke | Bulldog | Tigrado | 22 | 40 | 05/09/2020 | 0.22 |
| 5 | Daisy | Beagle | Tricolor | 10 | 33 | 08/11/2015 | 0.10 |
| 6 | Milo | Shih Tzu | Cinza | 5 | 25 | 07/06/2018 | 0.05 |
| 7 | Zoe | Dachshund | Preto | 6 | 28 | 09/04/2019 | 0.06 |
| 8 | Oliver | Husky | Cinza | 24 | 60 | 02/03/2016 | 0.24 |
| 9 | Chloe | Rottweiler | Preto | 45 | 63 | 11/10/2017 | 0.45 |
| 10 | Luna | Golden Retriever | Dourado | 25 | 32 | 15/09/2016 | NaN |
| 11 | Oliver | Beagle | Tricolor | 12 | 18 | 22/04/2020 | NaN |
| 12 | Milo | Labrador | Preto | 35 | 55 | 07/11/2015 | NaN |



Excluindo duplicados

O método **drop_duplicates** do pandas remove linhas duplicadas de um DataFrame, permitindo a criação de um novo DataFrame sem repetições com base em critérios específicos definidos pelos valores em determinadas colunas. O argumento subset especifica as colunas para identificar duplicatas, e o resultado é um DataFrame sem as linhas duplicadas.

```
1 df_cachorros.drop_duplicates(subset="Nome")  
2
```

| | Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|---|--------|------------------|----------|-------------|-----------|--------------------|------------|
| 0 | Max | Labrador | Amarelo | 30 | 60 | 01/05/2018 | 0.30 |
| 1 | Bella | Poodle | Branco | 7 | 35 | 10/12/2019 | 0.07 |
| 2 | Rocky | Boxer | Marrom | 28 | 65 | 03/08/2017 | 0.28 |
| 3 | Luna | Golden Retriever | Dourado | 25 | 55 | 12/02/2016 | 0.25 |
| 4 | Duke | Bulldog | Tigrado | 22 | 40 | 05/09/2020 | 0.22 |
| 5 | Daisy | Beagle | Tricolor | 10 | 33 | 08/11/2015 | 0.10 |
| 6 | Milo | Shih Tzu | Cinza | 5 | 25 | 07/06/2018 | 0.05 |
| 7 | Zoe | Dachshund | Preto | 6 | 28 | 09/04/2019 | 0.06 |
| 8 | Oliver | Husky | Cinza | 24 | 60 | 02/03/2016 | 0.24 |
| 9 | Chloe | Rottweiler | Preto | 45 | 63 | 11/10/2017 | 0.45 |



Excluindo duplicados com mais de um critério

```
1 sub_raca_nome = df_cachorros.drop_duplicates(subset=["Nome", "Raça"])
2 sub_raca_nome
```

| | Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|----|--------|------------------|----------|-------------|-----------|--------------------|------------|
| 0 | Max | Labrador | Amarelo | 30 | 60 | 01/05/2018 | 0.30 |
| 1 | Bella | Poodle | Branco | 7 | 35 | 10/12/2019 | 0.07 |
| 2 | Rocky | Boxer | Marrom | 28 | 65 | 03/08/2017 | 0.28 |
| 3 | Luna | Golden Retriever | Dourado | 25 | 55 | 12/02/2016 | 0.25 |
| 4 | Duke | Bulldog | Tigrado | 22 | 40 | 05/09/2020 | 0.22 |
| 5 | Daisy | Beagle | Tricolor | 10 | 33 | 08/11/2015 | 0.10 |
| 6 | Milo | Shih Tzu | Cinza | 5 | 25 | 07/06/2018 | 0.05 |
| 7 | Zoe | Dachshund | Preto | 6 | 28 | 09/04/2019 | 0.06 |
| 8 | Oliver | Husky | Cinza | 24 | 60 | 02/03/2016 | 0.24 |
| 9 | Chloe | Rottweiler | Preto | 45 | 63 | 11/10/2017 | 0.45 |
| 11 | Oliver | Beagle | Tricolor | 12 | 18 | 22/04/2020 | NaN |
| 12 | Milo | Labrador | Preto | 35 | 55 | 07/11/2015 | NaN |



values_count()

O método **value_counts()** do pandas retorna uma contagem de valores únicos em uma coluna de um DataFrame, fornecendo uma série que lista os valores distintos junto com a frequência de cada valor. Isso é útil para entender a distribuição dos dados e identificar os valores mais comuns em uma determinada coluna.

```
1 sub_raca_nome['Raça'].value_counts()  
  
Labrador      2  
Beagle         2  
Poodle         1  
Boxer          1  
Golden Retriever 1  
Bulldog        1  
Shih Tzu       1  
Dachshund     1  
Husky          1  
Rottweiler    1  
Name: Raça, dtype: int64
```

```
1 sub_raca_nome['Raça'].value_counts(sort=True)  
  
Labrador      2  
Beagle         2  
Poodle         1  
Boxer          1  
Golden Retriever 1  
Bulldog        1  
Shih Tzu       1  
Dachshund     1  
Husky          1  
Rottweiler    1  
Name: Raça, dtype: int64
```



values_count()

O método **value_counts()** do pandas retorna uma contagem de valores únicos em uma coluna de um DataFrame, fornecendo uma série que lista os valores distintos junto com a frequência de cada valor. Isso é útil para entender a distribuição dos dados e identificar os valores mais comuns em uma determinada coluna.

normalize é um argumento opcional que, quando definido como True, retorna as proporções normalizadas em vez das contagens absolutas. Ele apresenta a porcentagem relativa de cada valor em relação ao total de observações

```
1 sub_raca_nome[ 'Raça' ].value_counts(normalize=True)

Labrador          0.166667
Beagle            0.166667
Poodle            0.083333
Boxer              0.083333
Golden Retriever  0.083333
Bulldog            0.083333
Shih Tzu           0.083333
Dachshund          0.083333
Husky              0.083333
Rottweiler         0.083333
Name: Raça, dtype: float64
```



Agrupando

Agrupando sem nenhum método de agrupamento

```
1 print(df_cachorros[df_cachorros['Cor'] == 'Amarelo']['Peso (kg)'].mean())
2 print(df_cachorros[df_cachorros['Cor'] == 'Branco']['Peso (kg)'].mean())
3 print(df_cachorros[df_cachorros['Cor'] == 'Marrom']['Peso (kg)'].mean())
4 print(df_cachorros[df_cachorros['Cor'] == 'Dourado']['Peso (kg)'].mean())
5 print(df_cachorros[df_cachorros['Cor'] == 'Tricolor']['Peso (kg)'].mean())
6 print(df_cachorros[df_cachorros['Cor'] == 'Cinza']['Peso (kg)'].mean())
7 print(df_cachorros[df_cachorros['Cor'] == 'Preto']['Peso (kg)'].mean())
```

```
60.0
35.0
65.0
43.5
25.5
42.5
48.66666666666664
```



Usando groupby

```
1 df_cachorros.groupby('Cor')['Peso (kg)'].mean()
```

```
Cor
Amarelo      60.000000
Branco       35.000000
Cinza        42.500000
Dourado      43.500000
Marrom       65.000000
Preto         48.666667
Tigrado      40.000000
Tricolor     25.500000
Name: Peso (kg), dtype: float64
```

O método **groupby** no pandas é utilizado para agrupar um DataFrame por uma ou mais colunas, permitindo a aplicação de operações em cada grupo separadamente. Ele cria um objeto de grupo que pode ser combinado com várias funções de agregação, como média, soma, mínimo, máximo, entre outras. Esse método é frequentemente usado em conjunto com outros métodos, como **agg**, para calcular estatísticas resumidas para cada grupo de dados.



Usando groupby com agg

```
1 df_cachorros.groupby('Cor')[['Peso (kg)']].agg([min, max, sum])
```

| | min | max | sum |
|----------|-----|-----|-----|
| Cor | | | |
| Amarelo | 60 | 60 | 60 |
| Branco | 35 | 35 | 35 |
| Cinza | 25 | 60 | 85 |
| Dourado | 32 | 55 | 87 |
| Marrom | 65 | 65 | 65 |
| Preto | 28 | 63 | 146 |
| Tigrado | 40 | 40 | 40 |
| Tricolor | 18 | 33 | 51 |

O método **groupby** no pandas é utilizado para agrupar um DataFrame por uma ou mais colunas, permitindo a aplicação de operações em cada grupo separadamente. Ele cria um objeto de grupo que pode ser combinado com várias funções de agregação, como média, soma, mínimo, máximo, entre outras. Esse método é frequentemente usado em conjunto com outros métodos, como **agg**, para calcular estatísticas resumidas para cada grupo de dados.



Usando groupby com múltiplas variáveis

```
1 df_cachorros.groupby(['Cor', 'Raça'])['Peso (kg)'].mean()
```

| Cor | Raça | Peso (kg) |
|----------|------------------|-----------|
| Amarelo | Labrador | 60.0 |
| Branco | Poodle | 35.0 |
| Cinza | Husky | 60.0 |
| | Shih Tzu | 25.0 |
| Dourado | Golden Retriever | 43.5 |
| Marrom | Boxer | 65.0 |
| Preto | Dachshund | 28.0 |
| | Labrador | 55.0 |
| | Rottweiler | 63.0 |
| Tigrado | Bulldog | 40.0 |
| Tricolor | Beagle | 25.5 |

Name: Peso (kg), dtype: float64

Você também pode agrupar por várias colunas e calcular estatísticas resumidas.



Usando groupby com múltiplas variáveis

```
1 df_cachorros.groupby(['Cor', 'Raça'])[['Peso (kg)', 'Altura (cm)']].mean()
```

| Cor | Raça | Peso (kg) | Altura (cm) | grid icon | refresh icon |
|----------|------------------|-----------|-------------|-----------|--------------|
| | | | | | |
| Amarelo | Labrador | 60.0 | 30.0 | | |
| Branco | Poodle | 35.0 | 7.0 | | |
| Cinza | Husky | 60.0 | 24.0 | | |
| | Shih Tzu | 25.0 | 5.0 | | |
| Dourado | Golden Retriever | 43.5 | 25.0 | | |
| Marrom | Boxer | 65.0 | 28.0 | | |
| Preto | Dachshund | 28.0 | 6.0 | | |
| | Labrador | 55.0 | 35.0 | | |
| | Rottweiler | 63.0 | 45.0 | | |
| Tigrado | Bulldog | 40.0 | 22.0 | | |
| Tricolor | Beagle | 25.5 | 11.0 | | |

Você também pode agrupar por várias colunas e calcular estatísticas resumidas.



Tabela Pivot

O que são tabelas pivot (ou dinâmicas)

Tabelas dinâmicas são outra maneira de calcular estatísticas resumidas agrupadas. Se você já usou uma planilha, há chances de que já tenha usado uma tabela dinâmica. Vamos ver como criar tabelas dinâmicas no pandas. É uma forma de reorganizar e resumir dados tabulares para análise. O método **pivot_table** no pandas permite criar facilmente tabelas dinâmicas.

```
1 df_cachorros.groupby('Cor')[ 'Peso (kg)' ].mean()
```

| Cor | Peso (kg) |
|----------|-----------|
| Amarelo | 60.000000 |
| Branco | 35.000000 |
| Cinza | 42.500000 |
| Dourado | 43.500000 |
| Marrom | 65.000000 |
| Preto | 48.666667 |
| Tigrado | 40.000000 |
| Tricolor | 25.500000 |

Name: Peso (kg), dtype: float64

```
1 df_cachorros.pivot_table(values="Peso (kg)",index="Cor")
```

| Cor | Peso (kg) |
|----------|-----------|
| Amarelo | 60.000000 |
| Branco | 35.000000 |
| Cinza | 42.500000 |
| Dourado | 43.500000 |
| Marrom | 65.000000 |
| Preto | 48.666667 |
| Tigrado | 40.000000 |
| Tricolor | 25.500000 |



Usando aggfunc

Se quisermos uma estatística resumida diferente, podemos usar o argumento **aggfunc** e passar uma função.

```
1 df_cachorros.pivot_table(values="Peso (kg)",index="Cor", aggfunc=np.median)
```

| | Peso (kg) |
|----------|-----------|
| Cor | |
| Amarelo | 60.0 |
| Branco | 35.0 |
| Cinza | 42.5 |
| Dourado | 43.5 |
| Marrom | 65.0 |
| Preto | 55.0 |
| Tigrado | 40.0 |
| Tricolor | 25.5 |



Usando aggfunc para múltiplas estatísticas

Se quisermos uma estatística resumida diferente, podemos usar o argumento **aggfunc** e passar uma função.

```
1 df_cachorros.pivot_table(values="Peso (kg)", index="Cor", aggfunc=[np.mean, np.median])
```

| Cor | mean | median |
|----------|-----------|-----------|
| | Peso (kg) | Peso (kg) |
| Amarelo | 60.000000 | 60.0 |
| Branco | 35.000000 | 35.0 |
| Cinza | 42.500000 | 42.5 |
| Dourado | 43.500000 | 43.5 |
| Marrom | 65.000000 | 65.0 |
| Preto | 48.666667 | 55.0 |
| Tigrado | 40.000000 | 40.0 |
| Tricolor | 25.500000 | 25.5 |



Pivot para múltiplas variáveis

Para agrupar por duas variáveis, podemos passar um segundo nome de variável para o argumento "columns". Embora o resultado pareça um pouco diferente do que tínhamos antes, ele contém os mesmos números

```
1 df_cachorros.groupby(["Cor", "Raça"])["Peso (kg)"].mean()
```

```
Cor      Raça
Amarelo  Labrador      60.0
Branco   Poodle        35.0
Cinza    Husky         60.0
         Shih Tzu       25.0
Dourado  Golden Retriever 43.5
Marrom   Boxer          65.0
Preto    Dachshund     28.0
         Labrador      55.0
         Rottweiler    63.0
Tigrado  Bulldog        40.0
Tricolor Beagle         25.5
Name: Peso (kg), dtype: float64
```

```
1 df_cachorros.pivot_table(values="Peso (kg)", index="Cor", columns="Raça")
```

| | Beagle | Boxer | Bulldog | Dachshund | Golden Retriever | Husky | Labrador | Poodle | Rottweiler | Shih Tzu |
|----------|--------|-------|---------|-----------|------------------|-------|----------|--------|------------|----------|
| Cor | | | | | | | | | | |
| Amarelo | NaN | NaN | NaN | NaN | | NaN | NaN | 60.0 | NaN | NaN |
| Branco | NaN | NaN | NaN | NaN | | NaN | NaN | 35.0 | NaN | NaN |
| Cinza | NaN | NaN | NaN | NaN | | NaN | 60.0 | NaN | NaN | 25.0 |
| Dourado | NaN | NaN | NaN | NaN | | 43.5 | NaN | NaN | NaN | NaN |
| Marrom | NaN | 65.0 | NaN | NaN | | NaN | NaN | NaN | NaN | NaN |
| Preto | NaN | NaN | NaN | 28.0 | | NaN | NaN | 55.0 | NaN | 63.0 |
| Tigrado | NaN | NaN | 40.0 | NaN | | NaN | NaN | NaN | NaN | NaN |
| Tricolor | 25.5 | NaN | NaN | NaN | | NaN | NaN | NaN | NaN | NaN |



Substituir valores faltos

Em vez de ter muitos valores ausentes em nossa tabela dinâmica, podemos preenchê-los usando o argumento **fill_value**. Aqui, todos os NaNs são preenchidos com zeros.

```
1 df_cachorros.pivot_table(values="Peso (kg)", index="Cor", columns="Raça", fill_value=0)
```

| | Raça | Beagle | Boxer | Bulldog | Dachshund | Golden Retriever | Husky | Labrador | Poodle | Rottweiler | Shih Tzu |
|-----------------|------|--------|-------|---------|-----------|------------------|-------|----------|--------|------------|----------|
| Cor | | | | | | | | | | | |
| Amarelo | 0.0 | 0 | 0 | 0 | | 0.0 | 0 | 60 | 0 | 0 | 0 |
| Branco | 0.0 | 0 | 0 | 0 | | 0.0 | 0 | 0 | 35 | 0 | 0 |
| Cinza | 0.0 | 0 | 0 | 0 | | 0.0 | 60 | 0 | 0 | 0 | 25 |
| Dourado | 0.0 | 0 | 0 | 0 | | 43.5 | 0 | 0 | 0 | 0 | 0 |
| Marrom | 0.0 | 65 | 0 | 0 | | 0.0 | 0 | 0 | 0 | 0 | 0 |
| Preto | 0.0 | 0 | 0 | 28 | | 0.0 | 0 | 55 | 0 | 63 | 0 |
| Tigrado | 0.0 | 0 | 40 | 0 | | 0.0 | 0 | 0 | 0 | 0 | 0 |
| Tricolor | 25.5 | 0 | 0 | 0 | | 0.0 | 0 | 0 | 0 | 0 | 0 |



Substituir valores faltos

Se definirmos o argumento **margins** como True, a última linha e a última coluna da tabela dinâmica contêm a média de todos os valores na coluna ou linha, excluindo os valores ausentes que foram preenchidos com zeros.

```
1 df_cachorros.pivot_table(values="Peso (kg)", index="Cor", columns="Raça", fill_value=0, margins=True)
```

| Raça | Beagle | Boxer | Bulldog | Dachshund | Golden Retriever | Husky | Labrador | Poodle | Rottweiler | Shih Tzu | All | |
|-----------------|--------|-------|---------|-----------|------------------|-------|----------|--------|------------|----------|-----------|--|
| Cor | | | | | | | | | | | | |
| Amarelo | 0.0 | 0 | 0 | 0 | 0.0 | 0 | 60.0 | 0 | 0 | 0 | 60.000000 | |
| Branco | 0.0 | 0 | 0 | 0 | 0.0 | 0 | 0.0 | 35 | 0 | 0 | 35.000000 | |
| Cinza | 0.0 | 0 | 0 | 0 | 0.0 | 60 | 0.0 | 0 | 0 | 25 | 42.500000 | |
| Dourado | 0.0 | 0 | 0 | 0 | 43.5 | 0 | 0.0 | 0 | 0 | 0 | 43.500000 | |
| Marrom | 0.0 | 65 | 0 | 0 | 0.0 | 0 | 0.0 | 0 | 0 | 0 | 65.000000 | |
| Preto | 0.0 | 0 | 0 | 28 | 0.0 | 0 | 55.0 | 0 | 63 | 0 | 48.666667 | |
| Tigrado | 0.0 | 0 | 40 | 0 | 0.0 | 0 | 0.0 | 0 | 0 | 0 | 40.000000 | |
| Tricolor | 25.5 | 0 | 0 | 0 | 0.0 | 0 | 0.0 | 0 | 0 | 0 | 25.500000 | |
| All | 25.5 | 65 | 40 | 28 | 43.5 | 60 | 57.5 | 35 | 63 | 25 | 43.769231 | |



HORA DA PRÁTICA

Índice e fatiando o dataframe

Alterando índices

Revendo o dataframe

```
1 df_cachorros2
```

| | Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|---|--------|------------------|----------|-------------|-----------|--------------------|------------|
| 0 | Max | Labrador | Amarelo | 30 | 60 | 01/05/2018 | 0.30 |
| 1 | Bella | Poodle | Branco | 7 | 35 | 10/12/2019 | 0.07 |
| 2 | Rocky | Boxer | Marrom | 28 | 65 | 03/08/2017 | 0.28 |
| 3 | Luna | Golden Retriever | Dourado | 25 | 55 | 12/02/2016 | 0.25 |
| 4 | Duke | Bulldog | Tigrado | 22 | 40 | 05/09/2020 | 0.22 |
| 5 | Daisy | Beagle | Tricolor | 10 | 33 | 08/11/2015 | 0.10 |
| 6 | Milo | Shih Tzu | Cinza | 5 | 25 | 07/06/2018 | 0.05 |
| 7 | Zoe | Dachshund | Preto | 6 | 28 | 09/04/2019 | 0.06 |
| 8 | Oliver | Husky | Cinza | 24 | 60 | 02/03/2016 | 0.24 |
| 9 | Chloe | Rottweiler | Preto | 45 | 63 | 11/10/2017 | 0.45 |



Alterando o índice

Podemos colocar uma das colunas como índice do datafram usando o método **set_index**.

```
1 df_cachorros2.set_index('Nome')
```

| | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|---------------|------------------|----------|-------------|-----------|--------------------|------------|
| Nome | | | | | | |
| Max | Labrador | Amarelo | 30 | 60 | 01/05/2018 | 0.30 |
| Bella | Poodle | Branco | 7 | 35 | 10/12/2019 | 0.07 |
| Rocky | Boxer | Marrom | 28 | 65 | 03/08/2017 | 0.28 |
| Luna | Golden Retriever | Dourado | 25 | 55 | 12/02/2016 | 0.25 |
| Duke | Bulldog | Tigrado | 22 | 40 | 05/09/2020 | 0.22 |
| Daisy | Beagle | Tricolor | 10 | 33 | 08/11/2015 | 0.10 |
| Milo | Shih Tzu | Cinza | 5 | 25 | 07/06/2018 | 0.05 |
| Zoe | Dachshund | Preto | 6 | 28 | 09/04/2019 | 0.06 |
| Oliver | Husky | Cinza | 24 | 60 | 02/03/2016 | 0.24 |
| Chloe | Rottweiler | Preto | 45 | 63 | 11/10/2017 | 0.45 |



Removendo um index

Para desfazer o que você acabou de fazer, você pode redifinir o índice - ou seja, removê-lo. Isso é feito via **reset_index**.

```
1 df_cachorros2.reset_index()
```

| | Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|---|--------|------------------|----------|-------------|-----------|--------------------|------------|
| 0 | Max | Labrador | Amarelo | 30 | 60 | 01/05/2018 | 0.30 |
| 1 | Bella | Poodle | Branco | 7 | 35 | 10/12/2019 | 0.07 |
| 2 | Rocky | Boxer | Marrom | 28 | 65 | 03/08/2017 | 0.28 |
| 3 | Luna | Golden Retriever | Dourado | 25 | 55 | 12/02/2016 | 0.25 |
| 4 | Duke | Bulldog | Tigrado | 22 | 40 | 05/09/2020 | 0.22 |
| 5 | Daisy | Beagle | Tricolor | 10 | 33 | 08/11/2015 | 0.10 |
| 6 | Milo | Shih Tzu | Cinza | 5 | 25 | 07/06/2018 | 0.05 |
| 7 | Zoe | Dachshund | Preto | 6 | 28 | 09/04/2019 | 0.06 |
| 8 | Oliver | Husky | Cinza | 24 | 60 | 02/03/2016 | 0.24 |
| 9 | Chloe | Rottweiler | Preto | 45 | 63 | 11/10/2017 | 0.45 |



Removendo um index

Para desfazer o que você acabou de fazer, você pode redefinir o índice - ou seja, removê-lo. Isso é feito via **reset_index**.

reset_index tem um argumento chamado "drop" que permite descartar um índice.

```
1 df_cachorros2.reset_index(drop=True)
```

| | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|---|------------------|----------|-------------|-----------|--------------------|------------|
| 0 | Labrador | Amarelo | 30 | 60 | 01/05/2018 | 0.30 |
| 1 | Poodle | Branco | 7 | 35 | 10/12/2019 | 0.07 |
| 2 | Boxer | Marrom | 28 | 65 | 03/08/2017 | 0.28 |
| 3 | Golden Retriever | Dourado | 25 | 55 | 12/02/2016 | 0.25 |
| 4 | Bulldog | Tigrado | 22 | 40 | 05/09/2020 | 0.22 |
| 5 | Beagle | Tricolor | 10 | 33 | 08/11/2015 | 0.10 |
| 6 | Shih Tzu | Cinza | 5 | 25 | 07/06/2018 | 0.05 |
| 7 | Dachshund | Preto | 6 | 28 | 09/04/2019 | 0.06 |
| 8 | Husky | Cinza | 24 | 60 | 02/03/2016 | 0.24 |
| 9 | Rottweiler | Preto | 45 | 63 | 11/10/2017 | 0.45 |



Usar index faz subset mais fáceis de se filtrar

`loc` ajuda a pegar um pedaço do dataframe. Entenderemos melhor dele nos próximos slides.

```
1 df_cachorros[df_cachorros['Nome'].isin(['Max', 'Zoe'])]
```

| | Nome | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|---|------|-----------|---------|-------------|-----------|--------------------|------------|
| 0 | Max | Labrador | Amarelo | 30 | 60 | 01/05/2018 | 0.30 |
| 7 | Zoe | Dachshund | Preto | 6 | 28 | 09/04/2019 | 0.06 |

```
1 df_cachorros2.loc[['Max', 'Zoe']]
```

| | Raça | Cor | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|------|-----------|---------|-------------|-----------|--------------------|------------|
| Nome | | | | | | |
| Max | Labrador | Amarelo | 30 | 60 | 01/05/2018 | 0.30 |
| Zoe | Dachshund | Preto | 6 | 28 | 09/04/2019 | 0.06 |



Não é preciso ser um valor único para ser índice...

```
1 df_cachorros2.set_index('Cor')
```

| | Nome | Raça | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|-----------------|--------|------------------|-------------|-----------|--------------------|------------|
| Cor | | | | | | |
| Amarelo | Max | Labrador | 30 | 60 | 01/05/2018 | 0.30 |
| Branco | Bella | Poodle | 7 | 35 | 10/12/2019 | 0.07 |
| Marrom | Rocky | Boxer | 28 | 65 | 03/08/2017 | 0.28 |
| Dourado | Luna | Golden Retriever | 25 | 55 | 12/02/2016 | 0.25 |
| Tigrado | Duke | Bulldog | 22 | 40 | 05/09/2020 | 0.22 |
| Tricolor | Daisy | Beagle | 10 | 33 | 08/11/2015 | 0.10 |
| Cinza | Milo | Shih Tzu | 5 | 25 | 07/06/2018 | 0.05 |
| Preto | Zoe | Dachshund | 6 | 28 | 09/04/2019 | 0.06 |
| Cinza | Oliver | Husky | 24 | 60 | 02/03/2016 | 0.24 |
| Preto | Chloe | Rottweiler | 45 | 63 | 11/10/2017 | 0.45 |



Subset de um índice duplicado

Será retornado todas as linhas correspondentes!

| 3 df_cachorros2.loc['Preto'] | | | | | | |
|------------------------------|-------|------------|-------------|-----------|--------------------|------------|
| | Nome | Raça | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
| Cor | | | | | | |
| Preto | Zoe | Dachshund | 6 | 28 | 09/04/2019 | 0.06 |
| Preto | Chloe | Rottweiler | 45 | 63 | 11/10/2017 | 0.45 |



Multinível de índice

```
1 df_cachorros2.set_index(['Cor', 'Raça'])
```

| Cor | Raça | Nome | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|----------|------------------|--------|-------------|-----------|--------------------|------------|
| | | | | | | |
| Amarelo | Labrador | Max | 30 | 60 | 01/05/2018 | 0.30 |
| Branco | Poodle | Bella | 7 | 35 | 10/12/2019 | 0.07 |
| Marrom | Boxer | Rocky | 28 | 65 | 03/08/2017 | 0.28 |
| Dourado | Golden Retriever | Luna | 25 | 55 | 12/02/2016 | 0.25 |
| Tigrado | Bulldog | Duke | 22 | 40 | 05/09/2020 | 0.22 |
| Tricolor | Beagle | Daisy | 10 | 33 | 08/11/2015 | 0.10 |
| Cinza | Shih Tzu | Milo | 5 | 25 | 07/06/2018 | 0.05 |
| Preto | Dachshund | Zoe | 6 | 28 | 09/04/2019 | 0.06 |
| Cinza | Husky | Oliver | 24 | 60 | 02/03/2016 | 0.24 |
| Preto | Rottweiler | Chloe | 45 | 63 | 11/10/2017 | 0.45 |



Subset de múltiplos níveis com uma lista

```
3 df_cachorros2.loc[['Preto', 'Amarelo']]
```

| | | Nome | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|---------|------------|-------|-------------|-----------|--------------------|------------|
| Cor | Raça | | | | | |
| Preto | Dachshund | Zoe | 6 | 28 | 09/04/2019 | 0.06 |
| | Rottweiler | Chloe | 45 | 63 | 11/10/2017 | 0.45 |
| Amarelo | Labrador | Max | 30 | 60 | 01/05/2018 | 0.30 |



Subset de múltiplos níveis com uma lista

Utilizando tuplas

```
1 df_cachorros2.loc[['Branco', 'Poodle'], ('Amarelo', 'Labrador')]
```

| | | | Nome | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|---------|----------|-------|------|-------------|-----------|--------------------|------------|
| Cor | Raça | | | | | | |
| Branco | Poodle | Bella | | 7 | 35 | 10/12/2019 | 0.07 |
| Amarelo | Labrador | Max | | 30 | 60 | 01/05/2018 | 0.30 |



Ordenando pelo valor do index

Você também pode ordenar por valores do índice usando **sort_index**. Por padrão, ele ordena todos os níveis do índice de externo para interno, em ordem crescente.

```
1 df_cachorros2.sort_index()
```

| | | | Nome | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|--|----------|------------------|--------|-------------|-----------|--------------------|------------|
| | Cor | Raça | | | | | |
| | Amarelo | Labrador | Max | 30 | 60 | 01/05/2018 | 0.30 |
| | Branco | Poodle | Bella | 7 | 35 | 10/12/2019 | 0.07 |
| | Cinza | Husky | Oliver | 24 | 60 | 02/03/2016 | 0.24 |
| | | Shih Tzu | Milo | 5 | 25 | 07/06/2018 | 0.05 |
| | Dourado | Golden Retriever | Luna | 25 | 55 | 12/02/2016 | 0.25 |
| | Marrom | Boxer | Rocky | 28 | 65 | 03/08/2017 | 0.28 |
| | Preto | Dachshund | Zoe | 6 | 28 | 09/04/2019 | 0.06 |
| | | Rottweiler | Chloe | 45 | 63 | 11/10/2017 | 0.45 |
| | Tigrado | Bulldog | Duke | 22 | 40 | 05/09/2020 | 0.22 |
| | Tricolor | Beagle | Daisy | 10 | 33 | 08/11/2015 | 0.10 |



Controlando o sort_index

Você pode controlar a ordenação passando listas para os argumentos **level** e **ascending**.

```
1 df_cachorros2.sort_index(level=['Cor', 'Raça'], ascending=[True, False])
```

| | | | Nome | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|----------|------------------|--------|------|-------------|------------|--------------------|------------|
| Cor | Raça | | | | | | |
| Amarelo | Labrador | Max | 30 | 60 | 01/05/2018 | 0.30 | |
| Branco | Poodle | Bella | 7 | 35 | 10/12/2019 | 0.07 | |
| Cinza | Shih Tzu | Milo | 5 | 25 | 07/06/2018 | 0.05 | |
| | Husky | Oliver | 24 | 60 | 02/03/2016 | 0.24 | |
| Dourado | Golden Retriever | Luna | 25 | 55 | 12/02/2016 | 0.25 | |
| Marrom | Boxer | Rocky | 28 | 65 | 03/08/2017 | 0.28 | |
| Preto | Rottweiler | Chloe | 45 | 63 | 11/10/2017 | 0.45 | |
| | Dachshund | Zoe | 6 | 28 | 09/04/2019 | 0.06 | |
| Tigrado | Bulldog | Duke | 22 | 40 | 05/09/2020 | 0.22 | |
| Tricolor | Beagle | Daisy | 10 | 33 | 08/11/2015 | 0.10 | |



Índices são controversos!

- Embora simplifiquem o código de subdivisão, existem algumas desvantagens. **Os valores do índice são apenas dados.**
- **Armazenar dados em várias formas torna mais difícil pensar sobre eles.** Existe um conceito chamado "dados organizados", onde os dados são armazenados em forma tabular - como em um DataFrame. Cada linha contém uma única observação, e cada variável é armazenada em sua própria coluna. Os índices violam a última regra, já que os valores do índice não têm sua própria coluna.
- **No pandas, a sintaxe para trabalhar com índices é diferente da sintaxe para trabalhar com colunas.** Usar duas sintaxes torna seu código.



Slice usando loc

O que é Slicing?

Slicing é uma técnica para selecionar elementos consecutivos de objetos. Você também pode cortar **DataFrames**, mas primeiro precisa ordenar o índice.

Aqui, o conjunto de dados de cachorros recebeu um índice de vários níveis de raça e cor; em seguida, o índice é ordenado com **sort_index**.

| 2 df_cachorros2 | | | | | | |
|-----------------|------------------|--------|-------------|-----------|--------------------|------------|
| | | Nome | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
| Cor | Raça | | | | | |
| Amarelo | Labrador | Max | 30 | 60 | 01/05/2018 | 0.30 |
| Branco | Poodle | Bella | 7 | 35 | 10/12/2019 | 0.07 |
| Cinza | Husky | Oliver | 24 | 60 | 02/03/2016 | 0.24 |
| | Shih Tzu | Milo | 5 | 25 | 07/06/2018 | 0.05 |
| Dourado | Golden Retriever | Luna | 25 | 55 | 12/02/2016 | 0.25 |
| Marrom | Boxer | Rocky | 28 | 65 | 03/08/2017 | 0.28 |
| Preto | Dachshund | Zoe | 6 | 28 | 09/04/2019 | 0.06 |
| | Rottweiler | Chloe | 45 | 63 | 11/10/2017 | 0.45 |
| Tigrado | Bulldog | Duke | 22 | 40 | 05/09/2020 | 0.22 |
| Tricolor | Beagle | Daisy | 10 | 33 | 08/11/2015 | 0.10 |



Slice Nível Externo do índice

Slicing é uma técnica para selecionar elementos consecutivos de objetos. Você também pode cortar **DataFrames**, mas primeiro precisa ordenar o índice.

Aqui, o conjunto de dados de cachorros recebeu um índice de vários níveis de raça e cor; em seguida, o índice é ordenado com **sort_index**.

| 2 df_cachorros2 | | | | | | |
|-----------------|------------------|--------|-------------|-----------|--------------------|------------|
| | | Nome | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
| Cor | Raça | | | | | |
| Amarelo | Labrador | Max | 30 | 60 | | |
| Branco | Poodle | Bella | 7 | 35 | | |
| Cinza | Husky | Oliver | 24 | 60 | | |
| | Shih Tzu | Milo | 5 | 25 | | |
| Dourado | Golden Retriever | Luna | 25 | 55 | | |
| Marrom | Boxer | Rocky | 28 | 65 | | |
| Preto | Dachshund | Zoe | 6 | 28 | | |
| | Rottweiler | Chloe | 45 | 63 | | |
| Tigrado | Bulldog | Duke | 22 | 40 | | |
| Tricolor | Beagle | Daisy | 10 | 33 | | |

| 1 df_cachorros2['Amarelo':'Marrom'] | | | | | | |
|-------------------------------------|------------------|--------|-------------|-----------|--------------------|------------|
| | | Nome | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
| Cor | Raça | | | | | |
| Amarelo | Labrador | Max | 30 | 60 | 01/05/2018 | 0.30 |
| Branco | Poodle | Bella | 7 | 35 | 10/12/2019 | 0.07 |
| Cinza | Husky | Oliver | 24 | 60 | 02/03/2016 | 0.24 |
| | Shih Tzu | Milo | 5 | 25 | 07/06/2018 | 0.05 |
| Dourado | Golden Retriever | Luna | 25 | 55 | 12/02/2016 | 0.25 |
| Marrom | Boxer | Rocky | 28 | 65 | 03/08/2017 | 0.28 |



Slice Nível Interno do Índice

A mesma técnica não funciona nos níveis internos do índice.

Aqui, tentar cortar de labrador a Boxer e retornar um DataFrame vazio em vez dos seis cachorros desejados. É importante entender o perigo aqui: o pandas não gera um erro para alertar sobre o problema.

```
2 df_cachorros2
```

| | | | Nome | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|----------|------------------|--------|------|-------------|------------|--------------------|------------|
| Cor | Raça | | | | | | |
| Amarelo | Labrador | Max | 30 | 60 | 01/05/2018 | 0.30 | |
| Branco | Poodle | Bella | 7 | 35 | 10/12/2019 | 0.07 | |
| Cinza | Husky | Oliver | 24 | 60 | 02/03/2016 | 0.24 | |
| | Shih Tzu | Milo | 5 | 25 | 07/01/2020 | 0.05 | |
| Dourado | Golden Retriever | Luna | 25 | 55 | 12/02/2019 | 0.25 | |
| Marrom | Boxer | Rocky | 28 | 65 | 03/04/2018 | 0.28 | |
| Preto | Dachshund | Zoe | 6 | 28 | 09/07/2019 | 0.06 | |
| | Rottweiler | Chloe | 45 | 63 | 11/08/2017 | 0.45 | |
| Tigrado | Bulldog | Duke | 22 | 40 | 05/09/2019 | 0.22 | |
| Tricolor | Beagle | Daisy | 10 | 33 | 08/10/2020 | 0.10 | |

```
1 df_cachorros2['Labrador':'Boxer']
```

| | | | Nome | Altura (cm) | Peso (kg) | Data de Nascimento | Altura (m) |
|----------|------------------|--------|------|-------------|------------|--------------------|------------|
| Cor | Raça | | | | | | |
| Amarelo | Labrador | Max | 30 | 60 | 01/05/2018 | 0.30 | |
| Branco | Poodle | Bella | 7 | 35 | 10/12/2019 | 0.07 | |
| Cinza | Husky | Oliver | 24 | 60 | 02/03/2016 | 0.24 | |
| | Shih Tzu | Milo | 5 | 25 | 07/01/2020 | 0.05 | |
| Dourado | Golden Retriever | Luna | 25 | 55 | 12/02/2019 | 0.25 | |
| Marrom | Boxer | Rocky | 28 | 65 | 03/04/2018 | 0.28 | |
| Preto | Dachshund | Zoe | 6 | 28 | 09/07/2019 | 0.06 | |
| | Rottweiler | Chloe | 45 | 63 | 11/08/2017 | 0.45 | |
| Tigrado | Bulldog | Duke | 22 | 40 | 05/09/2019 | 0.22 | |
| Tricolor | Beagle | Daisy | 10 | 33 | 08/10/2020 | 0.10 | |



Slice Nível Interno do Índice

Para cortar linhas no nível externo de um índice, Você chama **loc**, passando os valores inicial e final separados por dois pontos. A diferença em relação ao corte de listas é que, em vez de especificar números de linha, você especifica valores de índice. Além disso, observe que o valor está incluído.

| 2 df_cachorros2 | | | | | | |
|-----------------|------------------|--------|------|-------------|------------|--------------------|
| | | | Nome | Altura (cm) | Peso (kg) | Data de Nascimento |
| Cor | Raça | | | | | Altura (m) |
| Amarelo | Labrador | Max | 30 | 60 | 01/05/2018 | 0.30 |
| Branco | Poodle | Bella | 7 | 35 | 10/12/2019 | 0.07 |
| Cinza | Husky | Oliver | 24 | 60 | 02/03/2016 | 0.24 |
| | Shih Tzu | Milo | 5 | 25 | 07/06/2018 | 0.05 |
| Dourado | Golden Retriever | Luna | 25 | 55 | 12/02/2016 | 0.25 |
| Marrom | Boxer | Rocky | 28 | 65 | 03/08/2017 | 0.28 |
| Preto | Dachshund | Zoe | 6 | 28 | 09/07/2019 | 0.09 |
| | Rottweiler | Chloe | 45 | 63 | | |
| Tigrado | Bulldog | Duke | 22 | 40 | 05/03/2018 | 0.25 |
| Tricolor | Beagle | Daisy | 10 | 33 | | |

| 1 df_cachorros2[('Amarelo', 'Labrador'):('Marrom', 'Boxer')] | | | | | | |
|--|------------------|--------|------|-------------|------------|--------------------|
| | | | Nome | Altura (cm) | Peso (kg) | Data de Nascimento |
| Cor | Raça | | | | | Altura (m) |
| Amarelo | Labrador | Max | 30 | 60 | 01/05/2018 | 0.30 |
| Branco | Poodle | Bella | 7 | 35 | 10/12/2019 | 0.07 |
| Cinza | Husky | Oliver | 24 | 60 | 02/03/2016 | 0.24 |
| | Shih Tzu | Milo | 5 | 25 | 07/06/2018 | 0.05 |
| Dourado | Golden Retriever | Luna | 25 | 55 | 12/02/2016 | 0.25 |
| Marrom | Boxer | Rocky | 28 | 65 | 03/08/2017 | 0.28 |



Slice de colunas

- Como os DataFrames são objetos bidimensionais, você também pode cortar colunas.
- Isso é feito passando dois argumentos para loc.
- O caso mais simples envolve a seleção de colunas, mas mantendo todas as linhas.
- Para fazer isso, passe dois pontos como o primeiro argumento para loc.

```
1 df_cachorros2.loc[:, 'Nome':'Peso (kg)']
```

| | | Nome | Altura (cm) | Peso (kg) |
|----------|------------------|--------|-------------|-----------|
| | Cor | Raça | | |
| Amarelo | Labrador | Max | 30 | 60 |
| Branco | Poodle | Bella | 7 | 35 |
| Cinza | Husky | Oliver | 24 | 60 |
| | Shih Tzu | Milo | 5 | 25 |
| Dourado | Golden Retriever | Luna | 25 | 55 |
| Marrom | Boxer | Rocky | 28 | 65 |
| Preto | Dachshund | Zoe | 6 | 28 |
| | Rottweiler | Chloe | 45 | 63 |
| Tigrado | Bulldog | Duke | 22 | 40 |
| Tricolor | Beagle | Daisy | 10 | 33 |



Cortando Duas Vezes

- Você pode cortar linhas e colunas ao mesmo tempo: basta passar o corte apropriado para cada argumento.
- Aqui, você vê os dois cortes anteriores sendo feitos na mesma linha de código.

```
1 df_cachorros2.loc[['Amarelo', 'Labrador'],
2                   ('Dourado', 'Golden Retriever'),
3                   'Nome':'Peso (kg)']]
```

| | | | Nome | Altura (cm) | Peso (kg) |  |
|---|---------|------------------|--------|-------------|-----------|---|
| | Cor | Raça | | | |  |
| 1 | Amarelo | Labrador | Max | 30 | 60 |  |
| 2 | Branco | Poodle | Bella | 7 | 35 |  |
| 3 | Cinza | Husky | Oliver | 24 | 60 |  |
| | | Shih Tzu | Milo | 5 | 25 |  |
| | Dourado | Golden Retriever | Luna | 25 | 55 |  |



Slice usando iloc

Usando posição numérica de linhas e colunas para fazer um corte no dataframe

- Você também pode cortar DataFrames por número de linha ou coluna usando o método iloc.
- Isso usa uma sintaxe semelhante ao corte de listas, exceto que há dois argumentos: um para linhas e outro para colunas.
- Observe que, como no corte de listas, mas ao contrário de loc, os valores finais não estão incluídos no corte.



Usando posição numérica de linhas e colunas para fazer um corte no dataframe

- Você também pode cortar DataFrames por número de linha ou coluna usando o método iloc.
- Isso usa uma sintaxe semelhante ao corte de listas, exceto que há dois argumentos: um para linhas e outro para colunas.
- Observe que, como no corte de listas, mas ao contrário de loc, os valores finais não estão incluídos no corte.

```
1 df_cachorros.iloc[1:3, :4]
```

| | Nome | Raça | Cor | Peso (kg) |
|---|-------|--------|--------|-----------|
| 1 | Bella | Poodle | Branco | 35.0 |
| 2 | Rocky | Boxer | Marrom | 65.0 |



Gráficos, valores e faltosos

É possível plotar um gráfico dos dados?

SIM!!

Visualizando um agrupamento de dados

Agrupando os cachorros por raça e tirando a média de peso de cada raça.

```
1 media_raca = df_cachorros.groupby('Raça')['Peso'].mean()  
2 media_raca  
  
Raça  
Beagle           33.0  
Boxer            65.0  
Bulldog          40.0  
Dachshund        28.0  
Golden Retriever 55.0  
Husky             60.0  
Labrador          60.0  
Poodle            35.0  
Rottweiler        63.0  
Shih Tzu          25.0  
Name: Peso, dtype: float64
```



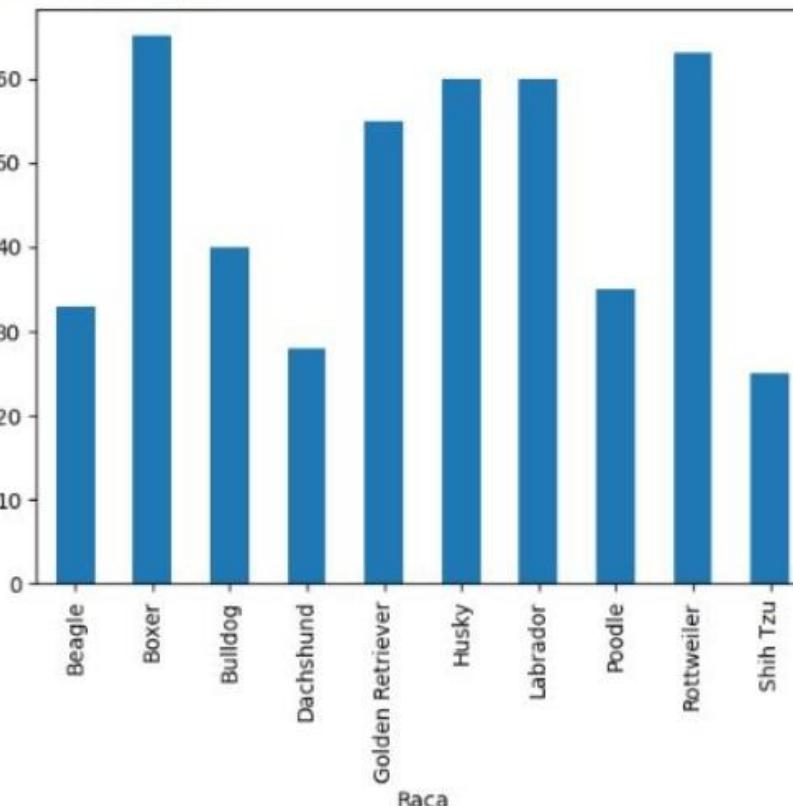
Visualizando um agrupamento de dados

Agrupando os cachorros por raça e tirando a média de peso de cada raça.

```
1 media_raca = df_cachorros.groupby('Raça')['Peso'].mean()  
2 media_raca  
  
Raça  
Beagle      33.0  
Boxer       65.0  
Bulldog     40.0  
Dachshund   28.0  
Golden Retriever  55.0  
Husky        60.0  
Labrador    60.0  
Poodle      35.0  
Rottweiler   63.0  
Shih Tzu    25.0  
Name: Peso, dtype: float64
```

```
1 media_raca.plot(kind='bar')
```

```
<Axes: xlabel='Raça'>
```



Import matplotlib

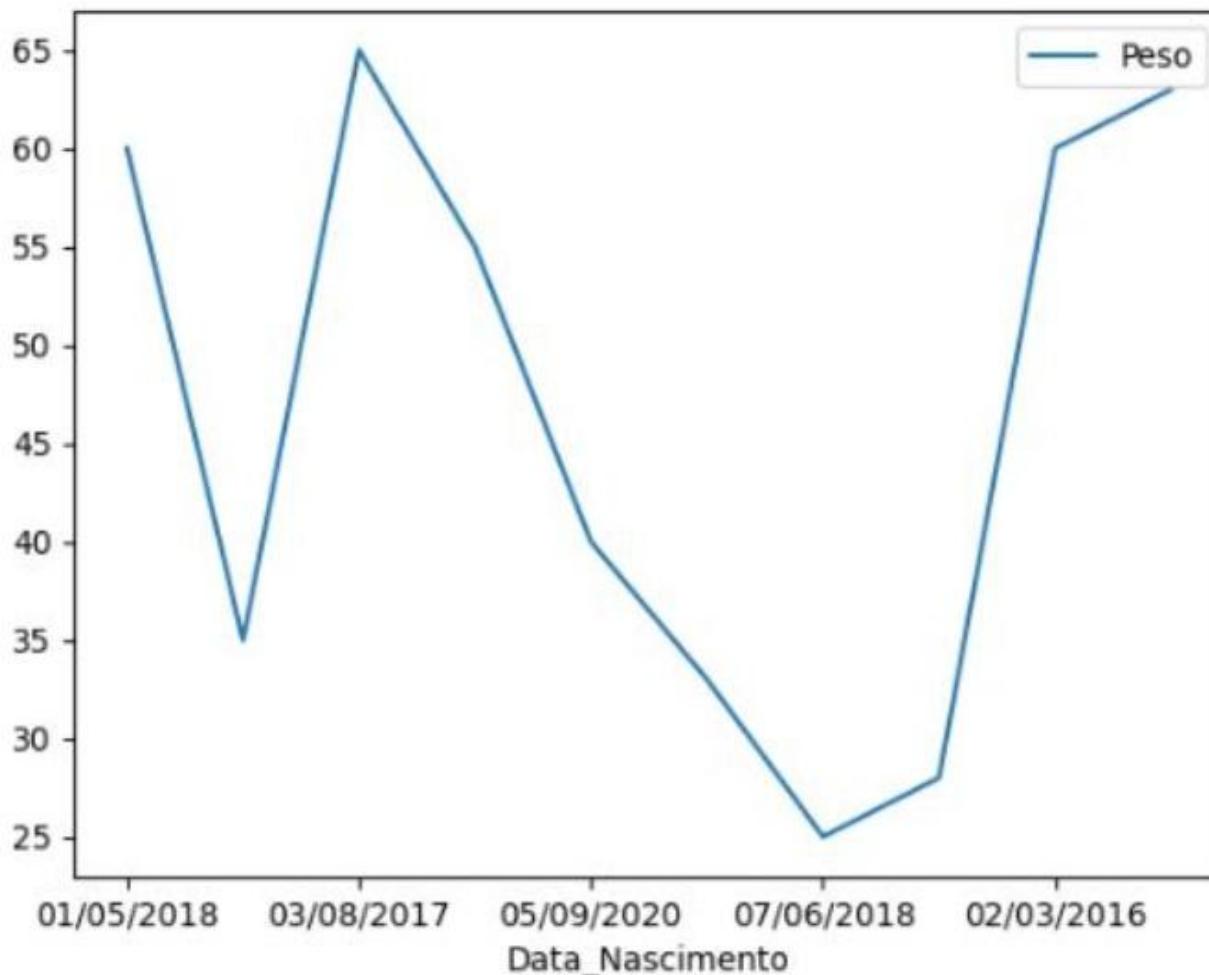
A biblioteca básica de visualização de dados no Python é o matplotlib. Será melhor estudado no curso de visualização de dados com Python.

```
1 from matplotlib import pyplot as plt
```



Plot de linha

```
1 df_cachorros.plot(x='Data_Nascimento', y='Peso', kind='line')
2 plt.show()
```



Lidando com valores faltosos

Dados não vêm perfeitos

Você pode receber um DataFrame com valores ausentes, por isso é importante saber como lidar com eles. A maioria dos dados não é perfeita – sempre existe a possibilidade de que faltem algumas peças no seu conjunto de dados. Em um DataFrame do pandas, os valores ausentes são indicados com N-a-N, que significa "não é um número".

| 1 df_cachorros | | | | | | |
|----------------|--------|------------------|----------|-----------|-------------|--------------------|
| | Nome | Raça | Cor | Peso (kg) | Altura (cm) | Data de Nascimento |
| 0 | Max | Labrador | Amarelo | NaN | 30 | 01/05/2018 |
| 1 | Bella | Poodle | Branco | 35.0 | 7 | 10/12/2019 |
| 2 | Rocky | Boxer | Marrom | 65.0 | 28 | 03/08/2017 |
| 3 | Luna | Golden Retriever | Dourado | 55.0 | 25 | 12/02/2016 |
| 4 | Duke | Bulldog | Tigrado | 40.0 | 22 | 05/09/2020 |
| 5 | Daisy | Beagle | Tricolor | NaN | 10 | 08/11/2015 |
| 6 | Milo | Shih Tzu | Cinza | 25.0 | 5 | 07/06/2018 |
| 7 | Zoe | Dachshund | Preto | 28.0 | 6 | 09/04/2019 |
| 8 | Oliver | Husky | Cinza | 60.0 | 24 | 02/03/2016 |
| 9 | Chloe | Rottweiler | Preto | 63.0 | 45 | 11/10/2017 |



Detectando valores nulos

.isna() é um método que pode ser utilizado para encontrar os valores nulos. Retorna "True" para registros nulos.

```
1 df_cachorros.isna()
```

| | Nome | Raça | Cor | Peso (kg) | Altura (cm) | Data de Nascimento |
|---|-------|-------|-------|-----------|-------------|--------------------|
| 0 | False | False | False | True | False | False |
| 1 | False | False | False | False | False | False |
| 2 | False | False | False | False | False | False |
| 3 | False | False | False | False | False | False |
| 4 | False | False | False | False | False | False |
| 5 | False | False | False | True | False | False |
| 6 | False | False | False | False | False | False |
| 7 | False | False | False | False | False | False |
| 8 | False | False | False | False | False | False |
| 9 | False | False | False | False | False | False |



E se o dataframe for maior? Como visualizar essa informação?

**any
sum
plot**

```
1 df_cachorros.isna().any()

Nome          False
Raça         False
Cor          False
Peso (kg)     True
Altura (cm)   False
Data de Nascimento False
dtype: bool
```



E se o dataframe for maior? Como visualizar essa informação?

any
sum
plot

```
1 df_cachorros.isna().any()
```

| | |
|--------------------|-------|
| Nome | False |
| Raça | False |
| Cor | False |
| Peso (kg) | True |
| Altura (cm) | False |
| Data de Nascimento | False |
| dtype: bool | |

```
1 df_cachorros.isna().sum()
```

| | |
|--------------------|---|
| Nome | 0 |
| Raça | 0 |
| Cor | 0 |
| Peso (kg) | 2 |
| Altura (cm) | 0 |
| Data de Nascimento | 0 |
| dtype: int64 | |



E se o dataframe for maior? Como visualizar essa informação?

**any
sum
plot**

```
1 df_cachorros.isna().any()
```

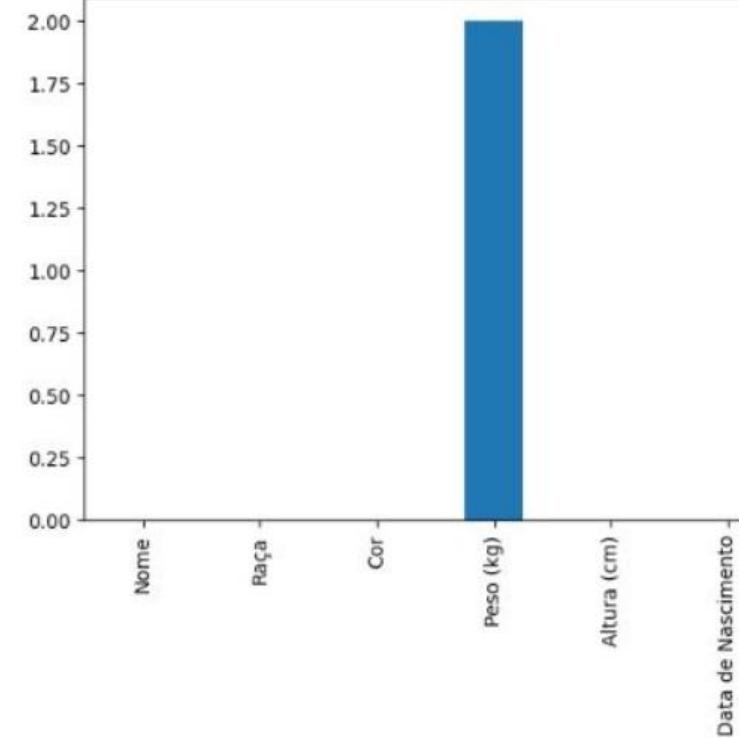
```
Nome          False
Raça         False
Cor           False
Peso (kg)     True
Altura (cm)   False
Data de Nascimento  False
dtype: bool
```

```
1 df_cachorros.isna().sum()
```

```
Nome          0
Raça         0
Cor           0
Peso (kg)     2
Altura (cm)   0
Data de Nascimento  0
dtype: int64
```

```
1 df_cachorros.isna().sum().plot(kind='bar')
```

<Axes:



Removendo dados nulos

O método **dropna** no pandas é utilizado para remover linhas ou colunas de um DataFrame que contenham valores nulos (NaN), facilitando a limpeza e preparação dos dados. Ele pode ser aplicado para remover linhas com pelo menos um NaN (padrão) ou colunas específicas, proporcionando flexibilidade na manipulação de conjuntos de dados.

```
df.dropna()    # Remove todas as linhas com pelo menos um NaN.  
df.dropna(axis=1)  # Remove todas as colunas com pelo menos um  
NaN.  
df.dropna(subset=['coluna1', 'coluna2'])  # Remove linhas com  
NaN apenas nas colunas especificadas.
```

| 1 df_cachorros.dropna() | | | | | | |
|-------------------------|--------|------------------|---------|-----------|-------------|--------------------|
| | Nome | Raça | Cor | Peso (kg) | Altura (cm) | Data de Nascimento |
| 1 | Bella | Poodle | Branco | 35.0 | 7 | 10/12/2019 |
| 2 | Rocky | Boxer | Marrom | 65.0 | 28 | 03/08/2017 |
| 3 | Luna | Golden Retriever | Dourado | 55.0 | 25 | 12/02/2016 |
| 4 | Duke | Bulldog | Tigrado | 40.0 | 22 | 05/09/2020 |
| 6 | Milo | Shih Tzu | Cinza | 25.0 | 5 | 07/06/2018 |
| 7 | Zoe | Dachshund | Preto | 28.0 | 6 | 09/04/2019 |
| 8 | Oliver | Husky | Cinza | 60.0 | 24 | 02/03/2016 |
| 9 | Chloe | Rottweiler | Preto | 63.0 | 45 | 11/10/2017 |



Substituindo valores nulos

O método **fillna** no pandas é utilizado para preencher valores nulos (NaN) em um DataFrame com valores específicos. Ele oferece a capacidade de substituir os valores ausentes por um número constante, pela média, moda, mediana; ou utilizar lógicas mais avançadas, proporcionando flexibilidade na manipulação de dados faltantes.

```
1 df_cachorros.fillna(df_cachorros.median(numeric_only=True))
```

| | Nome | Raça | Cor | Peso (kg) | Altura (cm) | Data de Nascimento |
|---|--------|------------------|----------|-----------|-------------|--------------------|
| 0 | Max | Labrador | Amarelo | 47.5 | 30 | 01/05/2018 |
| 1 | Bella | Poodle | Branco | 35.0 | 7 | 10/12/2019 |
| 2 | Rocky | Boxer | Marrom | 65.0 | 28 | 03/08/2017 |
| 3 | Luna | Golden Retriever | Dourado | 55.0 | 25 | 12/02/2016 |
| 4 | Duke | Bulldog | Tigrado | 40.0 | 22 | 05/09/2020 |
| 5 | Daisy | Beagle | Tricolor | 47.5 | 10 | 08/11/2015 |
| 6 | Milo | Shih Tzu | Cinza | 25.0 | 5 | 07/06/2018 |
| 7 | Zoe | Dachshund | Preto | 28.0 | 6 | 09/04/2019 |
| 8 | Oliver | Husky | Cinza | 60.0 | 24 | 02/03/2016 |
| 9 | Chloe | Rottweiler | Preto | 63.0 | 45 | 11/10/2017 |



Substituindo valores nulos

O método **fillna** no pandas é utilizado para preencher valores nulos (NaN) em um DataFrame com valores específicos. Ele oferece a capacidade de substituir os valores ausentes por um número constante, pela média, moda, mediana; ou utilizar lógicas mais avançadas, proporcionando flexibilidade na manipulação de dados faltantes.

```
1 df_cachorros.fillna(df_cachorros.median())  
<ipython-input-6-e27ac8ae6d41>:1: FutureWarning: The default value of numeric_d  
df_cachorros.fillna(df_cachorros.median())
```

| | Nome | Raça | Cor | Peso (kg) | Altura (cm) | Data de Nascimento |
|---|--------|------------------|----------|-----------|-------------|--------------------|
| 0 | Max | Labrador | Amarelo | 47.5 | 30 | 01/05/2018 |
| 1 | Bella | Poodle | Branco | 35.0 | 7 | 10/12/2019 |
| 2 | Rocky | Boxer | Marrom | 65.0 | 28 | 03/08/2017 |
| 3 | Luna | Golden Retriever | Dourado | 55.0 | 25 | 12/02/2016 |
| 4 | Duke | Bulldog | Tigrado | 40.0 | 22 | 05/09/2020 |
| 5 | Daisy | Beagle | Tricolor | 47.5 | 10 | 08/11/2015 |
| 6 | Milo | Shih Tzu | Cinza | 25.0 | 5 | 07/06/2018 |
| 7 | Zoe | Dachshund | Preto | 28.0 | 6 | 09/04/2019 |
| 8 | Oliver | Husky | Cinza | 60.0 | 24 | 02/03/2016 |
| 9 | Chloe | Rottweiler | Preto | 63.0 | 45 | 11/10/2017 |



HORA DA PRÁTICA