

✓ Importação da biblioteca

```
1 import pandas as pd
```

✓ Criando um dataframe pandas a partir de listas

```
1 import pandas as pd
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                   columns=['nome','idade','altura','peso','sexo'])
9 display(df)
```

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |

✓ Exibindo informações gerais do dataframe pandas

```
1 # Exibindo o número de linhas e colunas de uma dataframe
2 import pandas as pd
3 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
4 idade = [45,7,65,64,42,37]
5 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
6 peso = [70,22,87,64,96,68]
7 sexo = ['M','F','M','F','M','F']
8 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
9                   columns=['nome','idade','altura','peso','sexo'])
10 df.shape
```

```
→ (6, 5)
```

```
1 # Exibindo uma linha do dataframe aleatoriamente
2 import pandas as pd
3 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
4 idade = [45,7,65,64,42,37]
5 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
6 peso = [70,22,87,64,96,68]
7 sexo = ['M','F','M','F','M','F']
8 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
9                   columns=['nome','idade','altura','peso','sexo'])
10 df.sample()
```

```
→
→ nome    idade   altura   peso   sexo
→ 2 Pedro    65     1.75     87     M
```

```
1 # Exibindo mais uma linha do dataframe aleatoriamente
2 import pandas as pd
3 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
4 idade = [45,7,65,64,42,37]
5 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
6 peso = [70,22,87,64,96,68]
7 sexo = ['M','F','M','F','M','F']
8 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
9                   columns=['nome','idade','altura','peso','sexo'])
10 df.sample(3)
```

```
→
→ nome    idade   altura   peso   sexo
→ 5 Ester    37     1.73     68     F
→ 4 Eduardo  42     1.82     96     M
→ 0 Douglas  45     1.85     70     M
```

```

1 # Exibindo os tipos de dados em um dataframe
2 import pandas as pd
3 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
4 idade = [45,7,65,64,42,37]
5 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
6 peso = [70,22,87,64,96,68]
7 sexo = ['M','F','M','F','M','F']
8 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
9                     columns=['nome','idade','altura','peso','sexo'])
10 df.dtypes

```

```

→ nome      object
idade     int64
altura    float64
peso      int64
sexo      object
dtype: object

```

```

1 #Exibindo os tipos de dados e outras informações do dataframe
2 import pandas as pd
3 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
4 idade = [45,7,65,64,42,37]
5 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
6 peso = [70,22,87,64,96,68]
7 sexo = ['M','F','M','F','M','F']
8 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
9                     columns=['nome','idade','altura','peso','sexo'])
10 df.info()

```

```

→ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 5 columns):
 #   Column Non-Null Count Dtype  
--- 
 0   nome    6 non-null    object  
 1   idade   6 non-null    int64   
 2   altura  6 non-null    float64 
 3   peso    6 non-null    int64   
 4   sexo    6 non-null    object  
dtypes: float64(1), int64(2), object(2)
memory usage: 368.0+ bytes

```

```

1 # Exibir informações do índice do dataframe
2 import pandas as pd
3 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
4 idade = [45,7,65,64,42,37]
5 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
6 peso = [70,22,87,64,96,68]
7 sexo = ['M','F','M','F','M','F']
8 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
9                     columns=['nome','idade','altura','peso','sexo'])
10 df.index

```

```

→ RangeIndex(start=0, stop=6, step=1)

```

```

1 # Criando um dataframe com rótulo
2 import pandas as pd
3 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
4 idade = [45,7,65,64,42,37]
5 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
6 peso = [70,22,87,64,96,68]
7 sexo = ['M','F','M','F','M','F']
8 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
9                     columns=['nome','idade','altura','peso','sexo'])
10 medidas = df.describe()
11 display(meidas)

```

| | idade | altura | peso |
|-------|-----------|----------|-----------|
| count | 6.000000 | 6.000000 | 6.000000 |
| mean | 43.333333 | 1.675000 | 67.833333 |
| std | 21.266562 | 0.22731 | 25.615750 |
| min | 7.000000 | 1.23000 | 22.000000 |
| 25% | 38.250000 | 1.68500 | 65.000000 |
| 50% | 43.500000 | 1.74000 | 69.000000 |
| 75% | 59.250000 | 1.80250 | 82.750000 |
| max | 65.000000 | 1.85000 | 96.000000 |

```

1 # Exibir informações do índice do dataframe com rótulo
2 import pandas as pd
3 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
4 idade = [45,7,65,64,42,37]
5 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
6 peso = [70,22,87,64,96,68]
7 sexo = ['M','F','M','F','M','F']
8 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
9                     columns=['nome','idade','altura','peso','sexo'])
10 medidas.index

```

→ Index(['count', 'mean', 'std', 'min', '25%', '50%', '75%', 'max'], dtype='object')

```

1 # Exibindo apenas as primeiras linhas de um dataframe
2 import pandas as pd
3 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
4 idade = [45,7,65,64,42,37]
5 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
6 peso = [70,22,87,64,96,68]
7 sexo = ['M','F','M','F','M','F']
8 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
9                     columns=['nome','idade','altura','peso','sexo'])
10 df.head()

```

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |

```

1 # Exibindo apenas as últimas linhas de um dataframe
2 import pandas as pd
3 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
4 idade = [45,7,65,64,42,37]
5 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
6 peso = [70,22,87,64,96,68]
7 sexo = ['M','F','M','F','M','F']
8 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
9                     columns=['nome','idade','altura','peso','sexo'])
10 df.tail()

```

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |

✓ Fatiamento de linhas de um dataframe

```

1 # Exibir as informações de uma linha por rótulo
2 import pandas as pd
3 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
4 idade = [45,7,65,64,42,37]
5 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
6 peso = [70,22,87,64,96,68]
7 sexo = ['M','F','M','F','M','F']
8 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
9                     columns=['nome','idade','altura','peso','sexo'])
10 #display(df)
11 df.loc[0]

```

→ nome Douglas
 idade 45
 altura 1.85
 peso 70
 sexo M
 Name: 0, dtype: object

```

1 # Criando um df com rótulo do índice como texto
2 import pandas as pd
3 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
4 idade = [45,7,65,64,42,37]
5 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
6 peso = [70,22,87,64,96,68]
7 sexo = ['M','F','M','F','M','F']
8 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
9                     columns=['nome','idade','altura','peso','sexo'])
10 medidas = df.describe()

```

```

10 medidas = pd.read_csv('dados.csv')
11 #display(meidas)
12 #exibir as informações de uma linha pelo rótulo
13 medidas.loc['count']

→ idade      6.0
    altura     6.0
    peso       6.0
Name: count, dtype: float64

1 '''
2 Temos
3 1 - df (todas as informações)
4 2 - medidas (tabela só com as estatísticas )
5 '''

1 '''
2 Resumo da seção o método loc procura um rótulo, que por sua
3 vez dependendo do dataframe, poder ser um rótulo de texto, ou
4 um rótulo de número, quando usamos a função display, sempre
5 será exibido o rótulo da linha, seja ele texto ou número.
6 '''

1 # guardando os valores fatiados em um objeto pandas
2 '''
3 Ao fatiarmos um dataframe pandas e armazená-lo, automaticamente o fatiamento
4 será convertido em outro dataframe (subtabela da tabela original)
5 '''

6 import pandas as pd
7 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
8 idade = [45,7,65,64,42,37]
9 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
10 peso = [70,22,87,64,96,68]
11 sexo = ['M','F','M','F','M','F']
12 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
13                     columns=['nome','idade','altura','peso','sexo'])
14 display(df)
15 sub_tabela = df.loc[0]
16 display(sub_tabela)

```

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |

```

nome      Douglas
idade     45
altura    1.85
peso      70
sexo      M
Name: 0, dtype: object

```

```

1 # guardando os valores fatiados em um objeto Python puro (lista)
2 import pandas as pd
3 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
4 idade = [45,7,65,64,42,37]
5 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
6 peso = [70,22,87,64,96,68]
7 sexo = ['M','F','M','F','M','F']
8 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
9                     columns=['nome','idade','altura','peso','sexo'])
10 display(df)
11 sub_tabela = list(df.loc[0])
12 print(sub_tabela)

```

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |

```

['Douglas', 45, 1.85, 70, 'M']

```

```

1 # Exibindo mais de uma linha pelo rótulo
2 import pandas as pd
3 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
4 idade = [45,7,65,64,42,37]
5 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
6 peso = [70,22,87,64,96,68]
7 sexo = ['M','F','M','F','M','F']
8 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
9                     columns=['nome','idade','altura','peso','sexo'])
10 display(df)
11 df.loc[[0,2,4]]

```

→

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 4 | Eduardo | 42 | 1.82 | 96 | M |

```

1 # armazenando mais de uma linha pelo rótulo
2 import pandas as pd
3 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
4 idade = [45,7,65,64,42,37]
5 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
6 peso = [70,22,87,64,96,68]
7 sexo = ['M','F','M','F','M','F']
8 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
9                     columns=['nome','idade','altura','peso','sexo'])
10 display(df)
11 sub2 = df.loc[[0,2,4]]
12 display(sub2)

```

→

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 4 | Eduardo | 42 | 1.82 | 96 | M |

```

1 """
2 Agora vamos supor que desejamos armazenar uma subtabela da tabela medidas
3 apenas com as linhas de média e desvio padrão, fatiadas pelo rótulo
4 """
5 import pandas as pd
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 medidas = df.describe()
14 display(medidas)
15 sub3 = medidas.loc[['mean','std']]
16 display(sub3)

```

→

| | idade | altura | peso |
|-------|-----------|----------|-----------|
| count | 6.000000 | 6.000000 | 6.000000 |
| mean | 43.333333 | 1.675000 | 67.833333 |
| std | 21.266562 | 0.22731 | 25.615750 |
| min | 7.000000 | 1.230000 | 22.000000 |
| 25% | 38.250000 | 1.685000 | 65.000000 |
| 50% | 43.500000 | 1.740000 | 69.000000 |
| 75% | 59.250000 | 1.80250 | 82.750000 |
| max | 65.000000 | 1.85000 | 96.000000 |
| | idade | altura | peso |
| mean | 43.333333 | 1.67500 | 67.833333 |
| std | 21.266562 | 0.22731 | 25.615750 |

```

1 '''
2 Agora vamos supor que desejamos armazenar uma subtabela da tabela medidas
3 apenas com as linhas da tabela original, fatiadas pelo rótulo e exibidas
4 e armazenadas em ordem diferente da original
5 '''
6 import pandas as pd
7 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
8 idade = [45, 7, 65, 64, 42, 37]
9 altura = [1.85, 1.23, 1.75, 1.67, 1.82, 1.73]
10 peso = [70, 22, 87, 64, 96, 68]
11 sexo = ['M', 'F', 'M', 'F', 'M', 'F']
12 df = pd.DataFrame(zip(nome, idade, altura, peso, sexo),
13                     columns=['nome', 'idade', 'altura', 'peso', 'sexo'])
14 display(meidas)
15 sub4 = medidas.loc[['min', '25%', '50%', '75%', 'max', 'count', 'mean', 'std']]
16 display(sub4)

```

| | idade | altura | peso |
|--------------|-----------|----------|-----------|
| count | 6.000000 | 6.000000 | 6.000000 |
| mean | 43.333333 | 1.675000 | 67.833333 |
| std | 21.266562 | 0.22731 | 25.615750 |
| min | 7.000000 | 1.23000 | 22.000000 |
| 25% | 38.250000 | 1.68500 | 65.000000 |
| 50% | 43.500000 | 1.74000 | 69.000000 |
| 75% | 59.250000 | 1.80250 | 82.750000 |
| max | 65.000000 | 1.85000 | 96.000000 |
| | idade | altura | peso |
| min | 7.000000 | 1.23000 | 22.000000 |
| 25% | 38.250000 | 1.68500 | 65.000000 |
| 50% | 43.500000 | 1.74000 | 69.000000 |
| 75% | 59.250000 | 1.80250 | 82.750000 |
| max | 65.000000 | 1.85000 | 96.000000 |
| count | 6.000000 | 6.000000 | 6.000000 |
| mean | 43.333333 | 1.67500 | 67.833333 |
| std | 21.266562 | 0.22731 | 25.615750 |

```

1 # Exibindo linhas por índices em um dataframe com rótulo e índice numérico
2 import pandas as pd
3 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
4 idade = [45, 7, 65, 64, 42, 37]
5 altura = [1.85, 1.23, 1.75, 1.67, 1.82, 1.73]
6 peso = [70, 22, 87, 64, 96, 68]
7 sexo = ['M', 'F', 'M', 'F', 'M', 'F']
8 df = pd.DataFrame(zip(nome, idade, altura, peso, sexo),
9                     columns=['nome', 'idade', 'altura', 'peso', 'sexo'])
10 df.iloc[0]

```

| | nome | idade | altura | peso | sexo |
|------------------------|---------|-------|--------|------|------|
| Name: 0, dtype: object | Douglas | 45 | 1.85 | 70 | M |

```

1 # Exibindo linhas por indices em um dataframe com rótulo textual e índice numérico
2 import pandas as pd
3 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
4 idade = [45, 7, 65, 64, 42, 37]
5 altura = [1.85, 1.23, 1.75, 1.67, 1.82, 1.73]
6 peso = [70, 22, 87, 64, 96, 68]
7 sexo = ['M', 'F', 'M', 'F', 'M', 'F']
8 df = pd.DataFrame(zip(nome, idade, altura, peso, sexo),
9                     columns=['nome', 'idade', 'altura', 'peso', 'sexo'])
10 medidas = df.describe()
11 display(meidas)
12 medidas.iloc[0]

```

```

    idade  altura      peso
count   6.000000  6.000000  6.000000
mean   43.333333  1.675000  67.833333
std    21.266562  0.227310  25.615750
min    7.000000  1.230000  22.000000
25%   38.250000  1.685000  65.000000
50%   43.500000  1.740000  69.000000
75%   59.250000  1.802500  82.750000
max   65.000000  1.850000  96.000000
idade      6.0
altura     6.0
peso       6.0
Name: count, dtype: float64

```

```

1 # Exibindo mais de uma linha por índices em um dataframe com rótulo textual e índice numérico
2 import pandas as pd
3 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
4 idade = [45,7,65,64,42,37]
5 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
6 peso = [70,22,87,64,96,68]
7 sexo = ['M','F','M','F','M','F']
8 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
9                   columns=['nome','idade','altura','peso','sexo'])
10 medidas = df.describe()
11 display(meidas)
12 medidas.iloc[[0,1,2]]

```

```

    idade  altura      peso
count   6.000000  6.000000  6.000000
mean   43.333333  1.675000  67.833333
std    21.266562  0.227310  25.615750
min    7.000000  1.230000  22.000000
25%   38.250000  1.685000  65.000000
50%   43.500000  1.740000  69.000000
75%   59.250000  1.802500  82.750000
max   65.000000  1.850000  96.000000
    idade  altura      peso
count   6.000000  6.000000  6.000000
mean   43.333333  1.675000  67.833333
std    21.266562  0.227310  25.615750

```

✓ Fatiamento de colunas de um dataframe

```

1 # Fatiando uma coluna pelo seu rótulo
2 ''
3 Agora vamos buscar todas as linhas do dataframe original para a coluna
4 nome
5 ''
6 import pandas as pd
7 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
8 idade = [45,7,65,64,42,37]
9 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
10 peso = [70,22,87,64,96,68]
11 sexo = ['M','F','M','F','M','F']
12 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
13                   columns=['nome','idade','altura','peso','sexo'])
14 df.loc[:, 'nome']

    0    Douglas
    1    Daniela
    2      Pedro
    3      Maria
    4    Eduardo
    5      Ester
Name: nome, dtype: object

```

```

1 # Armazenando a tabela anterior em uma variável (sub5)
2 '''
3 Agora vamos buscar todas as linhas do dataframe original para a coluna
4 nome
5 '''
6 import pandas as pd
7 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
8 idade = [45,7,65,64,42,37]
9 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
10 peso = [70,22,87,64,96,68]
11 sexo = ['M','F','M','F','M','F']
12 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
13                     columns=['nome','idade','altura','peso','sexo'])
14 sub5 = df.loc[:, 'nome']
15 display(sub5)

```

| | nome |
|---|---------|
| 0 | Douglas |
| 1 | Daniela |
| 2 | Pedro |
| 3 | Maria |
| 4 | Eduardo |
| 5 | Ester |

Name: nome, dtype: object

```

1 # Fatiando mais de uma coluna pelo seu rótulo
2 '''
3 Agora vamos buscar todas as linhas do dataframe original para as colunas
4 nome e idade
5 '''
6 import pandas as pd
7 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
8 idade = [45,7,65,64,42,37]
9 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
10 peso = [70,22,87,64,96,68]
11 sexo = ['M','F','M','F','M','F']
12 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
13                     columns=['nome','idade','altura','peso','sexo'])
14 df.loc[:,['nome', 'idade']]

```

| | nome | idade |
|---|---------|-------|
| 0 | Douglas | 45 |
| 1 | Daniela | 7 |
| 2 | Pedro | 65 |
| 3 | Maria | 64 |
| 4 | Eduardo | 42 |
| 5 | Ester | 37 |

```

1 # Fatiando mais de uma coluna pelo seu rótulo
2 '''
3 Agora vamos buscar as linhas 0, 2 e 4 do dataframe original para as colunas
4 nome e idade
5 '''
6 import pandas as pd
7 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
8 idade = [45,7,65,64,42,37]
9 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
10 peso = [70,22,87,64,96,68]
11 sexo = ['M','F','M','F','M','F']
12 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
13                     columns=['nome','idade','altura','peso','sexo'])
14 df.loc[[0,2,4],['nome','idade']]

```

| | nome | idade |
|---|---------|-------|
| 0 | Douglas | 45 |
| 2 | Pedro | 65 |
| 4 | Eduardo | 42 |

```

1 '''
2 Agora vamos criar a tabela medidas como um describe da tabela original
3 e fatiar pelo rótulo o desvio padrão e média para as idades e para o peso
4 '''
5 import pandas as pd
6 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 medidas = df.describe()
14 #display(medidas)
15 medidas.loc[['std', 'mean'],['idade', 'peso']]

```

```
→ idade      peso  
std   21.266562 25.615750  
mean  43.333333 67.833333
```

```
1 # Exibindo colunas pelo índice  
2 '''  
3 Vamos exibir a coluna nome e todas as suas linhas  
4 '''  
5 import pandas as pd  
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']  
7 idade = [45,7,65,64,42,37]  
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]  
9 peso = [70,22,87,64,96,68]  
10 sexo = ['M','F','M','F','M','F']  
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),  
12                      columns=['nome','idade','altura','peso','sexo'])  
13 df.iloc[:,0]
```

```
→ 0    Douglas  
1    Daniela  
2    Pedro  
3    Maria  
4    Eduardo  
5    Ester  
Name: nome, dtype: object
```

```
1 # Exibindo colunas pelo índice  
2 '''  
3 Vamos exibir a coluna nome, a coluna idade e todas as suas linhas  
4 '''  
5 import pandas as pd  
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']  
7 idade = [45,7,65,64,42,37]  
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]  
9 peso = [70,22,87,64,96,68]  
10 sexo = ['M','F','M','F','M','F']  
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),  
12                      columns=['nome','idade','altura','peso','sexo'])  
13 df.iloc[:,[0,1]]
```

```
→ nome  idade  
0 Douglas  45  
1 Daniela   7  
2 Pedro     65  
3 Maria     64  
4 Eduardo   42  
5 Ester     37
```

```
1 # Alternativa 01 para exibir todas as linhas de uma coluna  
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']  
3 idade = [45,7,65,64,42,37]  
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]  
5 peso = [70,22,87,64,96,68]  
6 sexo = ['M','F','M','F','M','F']  
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),  
8                      columns=['nome','idade','altura','peso','sexo'])  
9 df['nome']
```

```
→ 0    Douglas  
1    Daniela  
2    Pedro  
3    Maria  
4    Eduardo  
5    Ester  
Name: nome, dtype: object
```

```
1 # Alternativa 02 para exibir todas as linhas de uma coluna  
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']  
3 idade = [45,7,65,64,42,37]  
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]  
5 peso = [70,22,87,64,96,68]  
6 sexo = ['M','F','M','F','M','F']  
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),  
8                      columns=['nome','idade','altura','peso','sexo'])  
9 df.nome
```

```
→ 0    Douglas  
1    Daniela  
2    Pedro  
3    Maria  
4    Eduardo  
5    Ester  
Name: nome, dtype: object
```

✓ Fatiamento de células (cruzamento de linha com coluna) de um dataframe

```
1 # Fatiando uma célula pelo seu rótulo
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                   columns=['nome','idade','altura','peso','sexo'])
9 ...
10 imagine que você deseja saber a idade de Douglas
11 ''
12 df.loc[0,'idade']
```

45

```
1 # Fatiando uma célula pelo seu índice
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                   columns=['nome','idade','altura','peso','sexo'])
9 display(df)
10 ''
11 imagine que você deseja saber a idade de Douglas
12 ''
13 df.iloc[0,1]
```

45

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |

```
1 ''
2 Agora vamos buscar a coluna idade pelo seu rótulo e a linha 0
3 pelo seu índice
4 ''
5 # Fatiando uma célula pelo seu índice
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 display(df)
14 df['idade'][0]
```

```
1 ''
2 Existe um método alternativo ao loc para exibir uma célula pelo
3 seus rótulos de linha e coluna
4 Exemplo:
5 Imagine que você deseja saber a idade presente na linha de rótulo 0
6 ''
7 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
8 idade = [45,7,65,64,42,37]
9 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
10 peso = [70,22,87,64,96,68]
11 sexo = ['M','F','M','F','M','F']
12 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
13                     columns=['nome','idade','altura','peso','sexo'])
14 df.at[0,'idade']
```

45

```

1 ''
2 Existe um método alternativo ao iloc para exibir uma célula pelos
3 seus índices de linha e coluna
4 Exemplo:
5 Imagine que você deseja saber a idade presente na linha de rótulo 0
6 ''
7 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
8 idade = [45,7,65,64,42,37]
9 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
10 peso = [70,22,87,64,96,68]
11 sexo = ['M', 'F', 'M', 'F', 'M', 'F']
12 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
13                     columns=['nome','idade','altura','peso','sexo'])
14 df.iat[0,1]

```

→ 45

▼ Operações que modificam o dataframe original

```

1 # renomear colunas método 01
2 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M', 'F', 'M', 'F', 'M', 'F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                     columns=['nome','idade','altura','peso','sexo'])
9 ''
10 Vamos renomear as colunas peso (massa) e sexo (genero)
11 ''
12 df.rename(columns={'peso':'massa', 'sexo':'genero'}, inplace=True)
13 display(df)

```

→

| | nome | idade | altura | massa | genero |
|---|---------|-------|--------|-------|--------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |

```

1 # renomear colunas método 02
2 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M', 'F', 'M', 'F', 'M', 'F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                     columns=['nome','idade','altura','peso','sexo'])
9 ''
10 Vamos renomear as colunas peso (massa) e sexo (genero)
11 ''
12 df = df.rename(columns={'peso':'massa', 'sexo':'genero'})
13 display(df)

```

→

| | nome | idade | altura | massa | genero |
|---|---------|-------|--------|-------|--------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |

```

1 # renomear colunas método 03
2 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M', 'F', 'M', 'F', 'M', 'F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                     columns=['nome','idade','altura','peso','sexo'])
9 ''
10 Vamos renomear as colunas peso (massa) e sexo (genero)
11 ''
12 df.rename({'peso':'massa', 'sexo':'genero'}, axis=1)
13 display(df)

```

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |

```

1 # renomear colunas método 04
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                   columns=['nome','idade','altura','peso','sexo'])
9 ...
10 Vamos renomear as colunas peso (massa) e sexo (genero)
11 ''
12 df = df.rename({'peso':'massa','sexo':'genero'}, axis='columns')
13 display(df)

```

| | nome | idade | altura | massa | genero |
|---|---------|-------|--------|-------|--------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |

```

1 # renomear colunas método 05
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                   columns=['nome','idade','altura','peso','sexo'])
9 ...
10 Vamos renomear as colunas peso (massa) e depois sexo (genero)
11 ''
12 df.columns.values[3] = 'massa'
13 df.columns.values[4] = 'genero'
14 display(df)

```

| | nome | idade | altura | massa | genero |
|---|---------|-------|--------|-------|--------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |

```

1 # renomeando linhas método 01
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                   columns=['nome','idade','altura','peso','sexo'])
9 ...
10 Agora vamos supor que desejamos renomear a linha 0 (pai) e
11 a linha 1 (filha)
12 ''
13 df = df.rename({0:'pai',1:'filha'}, axis=0)
14 display(df)

```

| | | nome | idade | altura | peso | sexo |
|-------|---------|------|-------|--------|------|------|
| pai | Douglas | 45 | 1.85 | 70 | M | |
| filha | Daniela | 7 | 1.23 | 22 | F | |
| 2 | Pedro | 65 | 1.75 | 87 | M | |
| 3 | Maria | 64 | 1.67 | 64 | F | |
| 4 | Eduardo | 42 | 1.82 | 96 | M | |
| 5 | Ester | 37 | 1.73 | 68 | F | |

```

1 # renomeando linhas método 02
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                   columns=['nome','idade','altura','peso','sexo'])
9 ...
10 Agora vamos supor que desejamos renomear a linha 0 (pai) e
11 a linha 1 (filha)
12 ''
13 df = df.rename({0:'pai',1:'filha'}, axis='rows')
14 display(df)

```

| | | nome | idade | altura | peso | sexo |
|-------|---------|------|-------|--------|------|------|
| pai | Douglas | 45 | 1.85 | 70 | M | |
| filha | Daniela | 7 | 1.23 | 22 | F | |
| 2 | Pedro | 65 | 1.75 | 87 | M | |
| 3 | Maria | 64 | 1.67 | 64 | F | |
| 4 | Eduardo | 42 | 1.82 | 96 | M | |
| 5 | Ester | 37 | 1.73 | 68 | F | |

```

1 # resetando os rótulos do índice
2 ''
3 Ao aplicarmos o método reset_index criamos um novo índice com
4 rótulo numérico e o rótulo anterior vira uma nova coluna no dataframe
5 ''
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 df = df.rename({0:'pai',1:'filha'}, axis='rows')
14 display(df)
15 df.reset_index(inplace=True)
16 display(df)

```

| | | nome | idade | altura | peso | sexo |
|-------|---------|---------|-------|--------|------|------|
| pai | Douglas | 45 | 1.85 | 70 | M | |
| filha | Daniela | 7 | 1.23 | 22 | F | |
| 2 | Pedro | 65 | 1.75 | 87 | M | |
| 3 | Maria | 64 | 1.67 | 64 | F | |
| 4 | Eduardo | 42 | 1.82 | 96 | M | |
| 5 | Ester | 37 | 1.73 | 68 | F | |
| index | | nome | idade | altura | peso | sexo |
| 0 | pai | Douglas | 45 | 1.85 | 70 | M |
| 1 | filha | Daniela | 7 | 1.23 | 22 | F |
| 2 | 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | 5 | Ester | 37 | 1.73 | 68 | F |

```

1 # excluindo colunas
2 '''
3 Ao aplicarmos o método reset_index criamos um novo índice com
4 rótulo numérico e o rótulo anterior vira uma nova coluna no dataframe
5 '''
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 df = df.rename({0:'pai',1:'filha'}, axis='rows')
14 display(df)
15 df.reset_index(inplace=True)
16 display(df)
17 '''
18 Agora vamos excluir a coluna gerada automaticamente no momento
19 em que resetamos o índice
20 '''
21 df.drop('index',axis=1,inplace=True)
22 display(df)

```

→

| | nome | idade | altura | peso | sexo |
|-------|---------|---------|--------|------|------|
| pai | Douglas | 45 | 1.85 | 70 | M |
| filha | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |
| index | nome | idade | altura | peso | sexo |
| 0 | pai | Douglas | 45 | 1.85 | 70 |
| 1 | filha | Daniela | 7 | 1.23 | 22 |
| 2 | 2 | Pedro | 65 | 1.75 | 87 |
| 3 | 3 | Maria | 64 | 1.67 | 64 |
| 4 | 4 | Eduardo | 42 | 1.82 | 96 |
| 5 | 5 | Ester | 37 | 1.73 | 68 |
| nome | idade | altura | peso | sexo | |
| 0 | Douglas | 45 | 1.85 | 70 | |
| 1 | Daniela | 7 | 1.23 | 22 | |
| 2 | Pedro | 65 | 1.75 | 87 | |
| 3 | Maria | 64 | 1.67 | 64 | |
| 4 | Eduardo | 42 | 1.82 | 96 | |
| 5 | Ester | 37 | 1.73 | 68 | |

```

1 # excluindo linhas
2 '''
3 Ao aplicarmos o método reset_index criamos um novo índice com
4 rótulo numérico e o rótulo anterior vira uma nova coluna no dataframe
5 '''
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 display(df)
14 '''
15 Agora vamos excluir a coluna gerada automaticamente no momento
16 em que resetamos o índice
17 '''
18 df.drop(0,axis=0,inplace=True)
19 display(df)

```

```

→ nome  idade  altura  peso  sexo
0 Douglas  45   1.85   70   M
1 Daniela   7    1.23   22   F
2 Pedro     65   1.75   87   M
3 Maria     64   1.67   64   F
4 Eduardo   42   1.82   96   M
5 Ester     37   1.73   68   F
→ nome  idade  altura  peso  sexo
1 Daniela   7    1.23   22   F
2 Pedro     65   1.75   87   M
3 Maria     64   1.67   64   F
4 Eduardo   42   1.82   96   M
5 Ester     37   1.73   68   F

```

```

1 # substituindo o valor de uma célula
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                   columns=['nome','idade','altura','peso','sexo'])
9 display(df)
10 ''
11 Imagine que você deseja substituir a idade de Maria (64) pelo
12 valor 65
13 ''
14 df['idade'][3] = 65
15 display(df)

```

```

→ nome  idade  altura  peso  sexo
0 Douglas  45   1.85   70   M
1 Daniela   7    1.23   22   F
2 Pedro     65   1.75   87   M
3 Maria     64   1.67   64   F
4 Eduardo   42   1.82   96   M
5 Ester     37   1.73   68   F
<ipython-input-135-e87c1940991d>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
df['idade'][3] = 65

```

```

→ nome  idade  altura  peso  sexo
0 Douglas  45   1.85   70   M
1 Daniela   7    1.23   22   F
2 Pedro     65   1.75   87   M
3 Maria     65   1.67   64   F
4 Eduardo   42   1.82   96   M
5 Ester     37   1.73   68   F

```

```

1 # substituindo o valor de uma célula
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                   columns=['nome','idade','altura','peso','sexo'])
9 display(df)
10 ''
11 Imagine que você deseja substituir a idade de Maria (64) pelo
12 valor 65
13 ''
14 df.idade[3] = 65
15 display(df)

```

```

    nome  idade  altura  peso  sexo
0 Douglas   45     1.85    70    M
1 Daniela    7     1.23    22    F
2 Pedro      65     1.75    87    M
3 Maria       64     1.67    64    F
4 Eduardo     42     1.82    96    M
5 Ester       37     1.73    68    F
<ipython-input-136-54817e6b636c>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
df.idade[3] = 65
    nome  idade  altura  peso  sexo
0 Douglas   45     1.85    70    M
1 Daniela    7     1.23    22    F
2 Pedro      65     1.75    87    M
3 Maria       65     1.67    64    F
4 Eduardo     42     1.82    96    M
5 Ester       37     1.73    68    F

```

```

1 # substituindo o valor de uma célula
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                   columns=['nome','idade','altura','peso','sexo'])
9 display(df)
10 ''
11 Imagine que você deseja substituir a idade de Maria (64) pelo
12 valor 65
13 ''
14 df.iat[3,1] = 65
15 display(df)

```

```

    nome  idade  altura  peso  sexo
0 Douglas   45     1.85    70    M
1 Daniela    7     1.23    22    F
2 Pedro      65     1.75    87    M
3 Maria       64     1.67    64    F
4 Eduardo     42     1.82    96    M
5 Ester       37     1.73    68    F
    nome  idade  altura  peso  sexo
0 Douglas   45     1.85    70    M
1 Daniela    7     1.23    22    F
2 Pedro      65     1.75    87    M
3 Maria       65     1.67    64    F
4 Eduardo     42     1.82    96    M
5 Ester       37     1.73    68    F

```

```

1 # substituindo o valor de uma célula
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                   columns=['nome','idade','altura','peso','sexo'])
9 display(df)
10 ''
11 Imagine que você deseja substituir a idade de Maria (64) pelo
12 valor 65
13 ''
14 df.at[3,'idade'] = 65
15 display(df)

```

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 65 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |

```

1 # ordenando de forma crescente um dataframe de acordo com um atributo
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                   columns=['nome','idade','altura','peso','sexo'])
9 display(df)
10 df = df.sort_values('idade')
11 display(df)

```

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 5 | Ester | 37 | 1.73 | 68 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |

```

1 # ordenando de forma decrescente um dataframe de acordo com um atributo
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                   columns=['nome','idade','altura','peso','sexo'])
9 display(df)
10 df = df.sort_values('idade',ascending=False)
11 display(df)

```

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |
| | nome | idade | altura | peso | sexo |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |
| 1 | Daniela | 7 | 1.23 | 22 | F |

```

1 '''
2 Exercício:
3 Crie um dataframe com as seguintes medidas descritivas a partir do
4 df original:
5 1 - contagem - (count)
6 2 - media - (mean)
7 3 - mediana - (criar linha nova)
8 4 - moda - (criar linha nova)
9 5 - variância - (criar linha nova)
10 6 - desvio padrão - (std)
11 7 - valor mínimo - (min)
12 8 - Q1 - (25%)
13 9 - Q2 - (50%)
14 10 -Q3 - (75%)
15 11 -Valor máximo (max)
16 12 -AIQ - (criar linha nova)
17 13 -LI - (criar linha nova)
18 14 -LS -(criar linha nova)
19 '''

20 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
21 idade = [45,7,65,64,42,37]
22 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
23 peso = [70,22,87,64,96,68]
24 sexo = ['M','F','M','F','M','F']
25 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
26                     columns=['nome','idade','altura','peso','sexo'])
27 medidas = df.describe()
28 mediana =[df.idade.median(),df.altura.median(),df.peso.median()]
29 moda =[[df.idade.mode()][0][0],[df.altura.mode()][0][0],[df.peso.mode()][0][0]]
30 variancia =[df.idade.var(),df.altura.var(),df.peso.var()]
31 q1 = list(medidas.loc['25%'])
32 q3 = list(medidas.loc['75%'])
33 aiq = []
34 for i in range(len(q1)):
35     aiq.append(q3[i]-q1[i])
36 li = []
37 for i in range(len(q1)):
38     li.append(q1[i]+1.5*aiq[i])
39 ls = []
40 for i in range(len(q3)):
41     ls.append(q3[i]+1.5*aiq[i])
42 medidas.loc['mediana'] = mediana
43 medidas.loc['moda'] = moda
44 medidas.loc['variancia'] = variancia
45 medidas.loc['aiq'] = aiq
46 medidas.loc['li'] = li
47 medidas.loc['ls'] = ls
48 medidas = medidas.loc[['count','mean','mediana','moda','variancia','std',
49                         'min','25%','50%','75%','max','aiq','li','ls']]
50 display(meidas)
51 medidas = medidas.rename({'count':'contagem',
52                             'mean':'media','std':'desvio padrao',
53                             'min':'valor minimo', '25%':'Q1', '50%':'Q2','75%':'Q3',
54                             'max': 'valor maximo'}, axis=0)
55 display(meidas)

```

| | idade | altura | peso |
|---------------|------------|---------|------------|
| count | 6.000000 | 6.00000 | 6.000000 |
| mean | 43.333333 | 1.67500 | 67.833333 |
| mediana | 43.500000 | 1.74000 | 69.000000 |
| moda | 7.000000 | 1.23000 | 22.000000 |
| variancia | 452.266667 | 0.05167 | 656.166667 |
| std | 21.266562 | 0.22731 | 25.615750 |
| min | 7.000000 | 1.23000 | 22.000000 |
| 25% | 38.250000 | 1.68500 | 65.000000 |
| 50% | 43.500000 | 1.74000 | 69.000000 |
| 75% | 59.250000 | 1.80250 | 82.750000 |
| max | 65.000000 | 1.85000 | 96.000000 |
| aiq | 21.000000 | 0.11750 | 17.750000 |
| li | 6.750000 | 1.50875 | 38.375000 |
| ls | 90.750000 | 1.97875 | 109.375000 |
| | idade | altura | peso |
| contagem | 6.000000 | 6.00000 | 6.000000 |
| media | 43.333333 | 1.67500 | 67.833333 |
| mediana | 43.500000 | 1.74000 | 69.000000 |
| moda | 7.000000 | 1.23000 | 22.000000 |
| variancia | 452.266667 | 0.05167 | 656.166667 |
| desvio padrao | 21.266562 | 0.22731 | 25.615750 |
| valor minimo | 7.000000 | 1.23000 | 22.000000 |
| Q1 | 38.250000 | 1.68500 | 65.000000 |
| Q2 | 43.500000 | 1.74000 | 69.000000 |
| Q3 | 59.250000 | 1.80250 | 82.750000 |
| valor maximo | 65.000000 | 1.85000 | 96.000000 |
| aiq | 21.000000 | 0.11750 | 17.750000 |
| li | 6.750000 | 1.50875 | 38.375000 |
| ls | 90.750000 | 1.97875 | 109.375000 |

▼ Agrupamentos

```

1 import pandas as pd
2 nome = ['Douglas','Daniela','Douglas','Maria','Douglas','Ester']
3 dependentes = [1,0,2,2,2,3]
4 salario = [1500,9000,1500,8000,2000,5000]
5 cargo = ['analista','gerente','analista','gerente','analista','coordenadora']
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,dependentes,salario,cargo,sexo),
8                   columns=['nome','dependentes','salario','cargo','sexo'])
9 display(df)

```

| | nome | dependentes | salario | cargo | sexo |
|---|---------|-------------|---------|--------------|------|
| 0 | Douglas | 1 | 1500 | analista | M |
| 1 | Daniela | 0 | 9000 | gerente | F |
| 2 | Douglas | 2 | 1500 | analista | M |
| 3 | Maria | 2 | 8000 | gerente | F |
| 4 | Douglas | 2 | 2000 | analista | M |
| 5 | Ester | 3 | 5000 | coordenadora | F |

```

1 # Verificando os valores únicos de um atributo(distintos)
2 df.nome.unique()
3
4 array(['Douglas', 'Daniela', 'Maria', 'Ester'], dtype=object)

1 ''
2 Consulta que retorna o número de dependentes agrupado pr nomes
3 ''
4 df[['nome','dependentes']].groupby('nome').sum()

```

```
→ dependentes
```

| nome | |
|---------|---|
| Daniela | 0 |
| Douglas | 5 |
| Ester | 3 |
| Maria | 2 |

```
1 '''
2 Essa consulta seleciona nome e salario, agrupando os
3 resultados por nome e exibindo nomes únicos (distintos) e
4 suas respectivas somas de salários
5 '''
6 df[['nome','salario']].groupby('nome').sum()
```

```
→ salario
```

| nome | |
|---------|------|
| Daniela | 9000 |
| Douglas | 5000 |
| Ester | 5000 |
| Maria | 8000 |

```
1 '''
2 Consulta que retorna nome e salário agrupando por nomes
3 e exibindo a média dos salários agrupados
4 '''
5 df[['nome','salario']].groupby('nome').mean()
```

```
→ salario
```

| nome | |
|---------|-------------|
| Daniela | 9000.000000 |
| Douglas | 1666.666667 |
| Ester | 5000.000000 |
| Maria | 8000.000000 |

```
1 '''
2 Selecionando as colunas cargo e salário e exibindo as medianas dos
3 salários agrupados por cargo
4 '''
5 df[['cargo','salario']].groupby('cargo').median()
```

```
→ salario
```

| cargo | |
|--------------|--------|
| analista | 1500.0 |
| coordenadora | 5000.0 |
| gerente | 8500.0 |

```
1 '''
2 Selecionando as colunas cargo e salário e exibindo as médias dos
3 salários agrupados por cargo
4 '''
5 df[['cargo','salario']].groupby('cargo').mean()
```

```
→ salario
```

| cargo | |
|--------------|-------------|
| analista | 1666.666667 |
| coordenadora | 5000.000000 |
| gerente | 8500.000000 |

```
1 '''
2 Selecionando as colunas cargo e salário e exibindo o Q1 (25%) dos
3 salários agrupados por cargo
4 '''
5 df[['cargo','salario']].groupby('cargo').quantile(0.99)
```

```
→ salario
```

| cargo | |
|--------------|--------|
| analista | 1990.0 |
| coordenadora | 5000.0 |
| gerente | 8990.0 |

✓ Criando novas colunas

```
1 ''
2 Imagine que a partir dos dados de peso e altura da base a seguir você deseja
3 criar uma nova coluna no df com o resultado do imc para cada registro
4 ''
5 import pandas as pd
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 display(df)
14 df['imc'] = round(df['peso']/df['altura'],2)
15 display(df)
```

```
→ nome  idade  altura  peso  sexo
  0 Douglas  45    1.85   70    M
  1 Daniela   7    1.23   22    F
  2 Pedro     65    1.75   87    M
  3 Maria     64    1.67   64    F
  4 Eduardo   42    1.82   96    M
  5 Ester     37    1.73   68    F
      nome  idade  altura  peso  sexo   imc
  0 Douglas  45    1.85   70    M  37.84
  1 Daniela   7    1.23   22    F  17.89
  2 Pedro     65    1.75   87    M  49.71
  3 Maria     64    1.67   64    F  38.32
  4 Eduardo   42    1.82   96    M  52.75
  5 Ester     37    1.73   68    F  39.31
```

✓ Inserindo uma linha de total em um DF

```
1 ''
2 Imagine que você deseja criar uma linha com o total das colunas numéricas
3 ''
4 import pandas as pd
5 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
6 idade = [45,7,65,64,42,37]
7 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
8 peso = [70,22,87,64,96,68]
9 sexo = ['M','F','M','F','M','F']
10 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
11                     columns=['nome','idade','altura','peso','sexo'])
12 display(df)
13 df.loc[6] = ['Soma',df['idade'].sum(),df['altura'].sum(),df['peso'].sum(),'-']
14 display(df)
```

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |
| 6 | Soma | 260 | 10.05 | 407 | - |

▼ Fazendo filtros em um DF

```

1 ''
2 Filtrar somente os registros e seus atributos para os eleemntos com 40 anos
3 ou mais
4 ''
5 import pandas as pd
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 display(df)
14 filtro01 = df.loc[df['idade']>=40,['nome','idade']]
15 display(filtro01)

```

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 1 | Daniela | 7 | 1.23 | 22 | F |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 3 | Maria | 64 | 1.67 | 64 | F |
| 4 | Eduardo | 42 | 1.82 | 96 | M |
| 5 | Ester | 37 | 1.73 | 68 | F |

| | nome | idade |
|---|---------|-------|
| 0 | Douglas | 45 |
| 2 | Pedro | 65 |
| 3 | Maria | 64 |
| 4 | Eduardo | 42 |

```

1 ''
2 Filtrar somente os registros e seus atributos para os eleemntos com 40 anos
3 ou mais usando o método query
4 ''
5 import pandas as pd
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 display(df)
14 filtro02 = df.query('idade >= 40')
15 filtro02 = filtro02[['nome','idade']]
16 display(filtro02)

```

```

    nome  idade  altura  peso  sexo
0 Douglas     45     1.85    70    M
1 Daniela      7     1.23    22    F
2 Pedro        65     1.75    87    M
3 Maria        64     1.67    64    F
4 Eduardo      42     1.82    96    M
5 Ester         37     1.73    68    F

```

| | nome | idade |
|---|---------|-------|
| 0 | Douglas | 45 |
| 2 | Pedro | 65 |
| 3 | Maria | 64 |
| 4 | Eduardo | 42 |

```

1 """
2 Filtrando colunas qualitativas usando o método query
3 """
4 import pandas as pd
5 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
6 idade = [45,7,65,64,42,37]
7 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
8 peso = [70,22,87,64,96,68]
9 sexo = ['M','F','M','F','M','F']
10 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
11                     columns=['nome','idade','altura','peso','sexo'])
12 display(df)
13 df.query('sexo == "M"')

```

```

    nome  idade  altura  peso  sexo
0 Douglas     45     1.85    70    M
1 Daniela      7     1.23    22    F
2 Pedro        65     1.75    87    M
3 Maria        64     1.67    64    F
4 Eduardo      42     1.82    96    M
5 Ester         37     1.73    68    F

```

| | nome | idade | altura | peso | sexo |
|---|---------|-------|--------|------|------|
| 0 | Douglas | 45 | 1.85 | 70 | M |
| 2 | Pedro | 65 | 1.75 | 87 | M |
| 4 | Eduardo | 42 | 1.82 | 96 | M |

```

1 """
2 Filtrando colunas com espaço no nome pelo método query
3 """
4 import pandas as pd
5 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
6 idade = [45,7,65,64,42,37]
7 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
8 peso = [70,22,87,64,96,68]
9 sexo = ['M','F','M','F','M','F']
10 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
11                     columns=['nome','idade funcionario','altura','peso','sexo'])
12 display(df)
13 filtro03 = df.query(`idade funcionario` >= 40')
14 display(filtro03)

```

```

    nome  idade funcionario  altura  peso  sexo
0 Douglas                      45     1.85    70    M
1 Daniela                      7     1.23    22    F
2 Pedro                        65     1.75    87    M
3 Maria                        64     1.67    64    F
4 Eduardo                      42     1.82    96    M
5 Ester                        37     1.73    68    F

```

| | nome | idade | funcionario | altura | peso | sexo |
|---|---------|-------|-------------|--------|------|------|
| 0 | Douglas | | 45 | 1.85 | 70 | M |
| 2 | Pedro | | 65 | 1.75 | 87 | M |
| 3 | Maria | | 64 | 1.67 | 64 | F |
| 4 | Eduardo | | 42 | 1.82 | 96 | M |

✓ Unindo dfs

```
1 ''
2 Filtrando colunas com espaço no nome pelo método query
3 ''
4 import pandas as pd
5 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
6 idade = [45,7,65,64,42,37]
7 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
8 peso = [70,22,87,64,96,68]
9 sexo = ['M','F','M','F','M','F']
10 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
11                     columns=['nome','idade','altura','peso','sexo'])
12 df['imc'] = round(df['peso']/df['altura']**2,2)
13 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
14 display(df)
```

| | nome | idade | altura | massa | genero | imc |
|---|---------|-------|--------|-------|--------|-------|
| 0 | Douglas | 45 | 1.85 | 70 | M | 20.45 |
| 1 | Daniela | 7 | 1.23 | 22 | F | 14.54 |
| 2 | Pedro | 65 | 1.75 | 87 | M | 28.41 |
| 3 | Maria | 64 | 1.67 | 64 | F | 22.95 |
| 4 | Eduardo | 42 | 1.82 | 96 | M | 28.98 |
| 5 | Ester | 37 | 1.73 | 68 | F | 22.72 |

```
1 df2 = pd.read_csv('https://docs.google.com/spreadsheets/d/e/2PACX-1vRg09sDMtHT1fErHtoNumxvPsvHpeChBW3JCLtvwCxHheP7qwC1ZuTQtAPC21HlosGW9Ct3zdCULi/put')
2 display(df2)
```

| | nome | idade | altura | massa | genero | imc |
|---|---------|-------|--------|-------|--------|-------|
| 0 | Joaquim | 56 | 1.67 | 69 | M | 24.74 |
| 1 | Aurora | 34 | 1.55 | 53 | F | 22.06 |
| 2 | Manoel | 23 | 1.76 | 68 | M | 21.95 |

```
1 import pandas as pd
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                     columns=['nome','idade','altura','peso','sexo'])
9 df['imc'] = round(df['peso']/df['altura']**2,2)
10 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
11 print('***** DF 01 - *****')
12 display(df)
13 df2 = pd.read_csv('https://docs.google.com/spreadsheets/d/e/2PACX-1vRg09sDMtHT1fErHtoNumxvPsvHpeChBW3JCLtvwCxHheP7qwC1ZuTQtAPC21HlosGW9Ct3zdCULi/put')
14 print('***** DF 02 - *****')
15 display(df2)
16 ''
17 agora queremos unir df e df2 e para isso vamos utilizar o método concat
18 ''
19 df3 = pd.concat([df,df2])
20 print('***** DF 03 - *****')
21 display(df3)
22 ''
23 Observe que ao unir os dois dfs os índices de cada df foram preservados,
24 assim, vamos reorganizá-los
25 ''
26 df3 = df3.sort_values('nome')
27 df3.reset_index(drop=True, inplace=True)
28 print('***** DF 03 - *****')
29 display(df3)
```

```

→***** DF 01 - *****
    nome  idade  altura  massa  genero  imc
 0 Douglas   45     1.85    70      M 20.45
 1 Daniela    7     1.23    22      F 14.54
 2 Pedro     65     1.75    87      M 28.41
 3 Maria      64     1.67    64      F 22.95
 4 Eduardo    42     1.82    96      M 28.98
 5 Ester      37     1.73    68      F 22.72
***** DF 02 - *****
    nome  idade  altura  massa  genero  imc
 0 Joaquim   56     1.67    69      M 24.74
 1 Aurora    34     1.55    53      F 22.06
 2 Manoel    23     1.76    68      M 21.95
***** DF 03 - *****
    nome  idade  altura  massa  genero  imc
 0 Douglas   45     1.85    70      M 20.45
 1 Daniela    7     1.23    22      F 14.54
 2 Pedro     65     1.75    87      M 28.41
 3 Maria      64     1.67    64      F 22.95
 4 Eduardo    42     1.82    96      M 28.98
 5 Ester      37     1.73    68      F 22.72
 0 Joaquim   56     1.67    69      M 24.74
 1 Aurora    34     1.55    53      F 22.06
 2 Manoel    23     1.76    68      M 21.95
***** DF 03 - *****
    nome  idade  altura  massa  genero  imc
 0 Aurora    34     1.55    53      F 22.06
 1 Daniela    7     1.23    22      F 14.54
 2 Douglas   45     1.85    70      M 20.45
 3 Eduardo    42     1.82    96      M 28.98
 4 Ester      37     1.73    68      F 22.72
 5 Joaquim   56     1.67    69      M 24.74
 6 Manoel    23     1.76    68      M 21.95
 7 Maria      64     1.67    64      F 22.95

```

✓ Substituindo valores de uma coluna

```

1 '''
2 Imagine que você deseja substituir todos os valores M da coluna genero,
3 por masculino e valores F por feminino
4 '''
5 import pandas as pd
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                      columns=['nome','idade','altura','peso','sexo'])
13 df['imc'] = round(df['peso']/df['altura']**2,2)
14 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
15 df['genero'] = df['genero'].map({'M':'masculino','F':'feminino'})
16 display(df)

```

```

→
    nome  idade  altura  massa  genero  imc
 0 Douglas   45     1.85    70  masculino 20.45
 1 Daniela    7     1.23    22  feminino 14.54
 2 Pedro     65     1.75    87  masculino 28.41
 3 Maria      64     1.67    64  feminino 22.95
 4 Eduardo    42     1.82    96  masculino 28.98
 5 Ester      37     1.73    68  feminino 22.72

```

```

1 '''
2 Imagine que você deseja substituir todos os valores M da coluna genero,
3 por 0 e valores F por 1
4 '''
5 import pandas as pd
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                      columns=['nome','idade','altura','peso','sexo'])
13 df['imc'] = round(df['peso']/df['altura']**2,2)
14 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
15 df['genero'] = df['genero'].map({'M':0,'F':1})
16 display(df)

```

| | nome | idade | altura | massa | genero | imc |
|---|---------|-------|--------|-------|--------|-------|
| 0 | Douglas | 45 | 1.85 | 70 | 0 | 20.45 |
| 1 | Daniela | 7 | 1.23 | 22 | 1 | 14.54 |
| 2 | Pedro | 65 | 1.75 | 87 | 0 | 28.41 |
| 3 | Maria | 64 | 1.67 | 64 | 1 | 22.95 |
| 4 | Eduardo | 42 | 1.82 | 96 | 0 | 28.98 |
| 5 | Ester | 37 | 1.73 | 68 | 1 | 22.72 |

✓ Agrupando variáveis quantitativas

```

1 '''
2 Vamos supor que desejamos criar um nova coluna com as faixas etárias a partir
3 dos registros de idade, por exemplo: nessa nova coluna um registro cuja idade
4 fosse 42 anos ficaria classificado em uma faixa etária de 41 a 50 anos por
5 exemplo
6 '''
7 import pandas as pd
8 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
9 idade = [45,7,65,64,42,37]
10 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
11 peso = [70,22,87,64,96,68]
12 sexo = ['M','F','M','F','M','F']
13 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
14                      columns=['nome','idade','altura','peso','sexo'])
15 df['imc'] = round(df['peso']/df['altura']**2,2)
16 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
17 print('***** DF 01 - *****')
18 display(df)
19 faixas = ['01 a 10','11 - 20','21 - 30','31 - 40','41 - 50','51 - 60','61 - 70']
20 df['faixa_etaria'] = pd.cut(x=df['idade'],bins=[0,10,20,30,40,50,60,70],labels=faixas)
21 '''
22 O método cut, exige três argumentos:
23 1 - x: a coluna do df original que será usada como referência para criar a nova
24 2 - bins: uma lista com os valores que serão utilizado na variável x como ponto
25 de corte para a classificação das faixas
26 3 - labels: uma lista que contém todos os possíveis valores da nova coluna
27 '''
28 print('***** DF 01 - Trannsformado *****')
29 display(df)

```

| | nome | idade | altura | massa | genero | imc | faixa_etaria |
|---|---------|-------|--------|-------|--------|-------|--------------|
| 0 | Douglas | 45 | 1.85 | 70 | M | 20.45 | 41 - 50 |
| 1 | Daniela | 7 | 1.23 | 22 | F | 14.54 | 01 a 10 |
| 2 | Pedro | 65 | 1.75 | 87 | M | 28.41 | 61 - 70 |
| 3 | Maria | 64 | 1.67 | 64 | F | 22.95 | 61 - 70 |
| 4 | Eduardo | 42 | 1.82 | 96 | M | 28.98 | 41 - 50 |
| 5 | Ester | 37 | 1.73 | 68 | F | 22.72 | 31 - 40 |

✓ Inserindo nova coluna em uma posição específica

```
1 ''
2 Imagine que você deseja inserir uma nova coluna com os cargos
3 dos funcionários, mas deseja que essa nova coluna fique posicionada
4 imediatamente após a coluna nome, sem excluir as demais colunas, ou seja,
5 "empurrando" as colunas para frente
6 ''
7 import pandas as pd
8 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
9 idade = [45,7,65,64,42,37]
10 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
11 peso = [70,22,87,64,96,68]
12 sexo = ['M','F','M','F','M','F']
13 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
14                     columns=['nome','idade','altura','peso','sexo'])
15 df['imc'] = round(df['peso']/df['altura']**2,2)
16 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
17 print('***** DF 01 - *****')
18 display(df)
19 df.insert(1,'cargo', ['analista','gerente','coordenador','analista','gerente',
20                      'diretor'])
21 print('***** DF 01 - com cargo *****')
22 display(df)
```

```
***** DF 01 - *****
  nome  idade  altura  massa  genero    imc
0 Douglas     45     1.85    70      M  20.45
1 Daniela      7     1.23    22      F  14.54
2 Pedro        65     1.75    87      M  28.41
3 Maria         64     1.67    64      F  22.95
4 Eduardo       42     1.82    96      M  28.98
5 Ester          37     1.73    68      F  22.72
***** DF 01 - com cargo *****
  nome      cargo  idade  altura  massa  genero    imc
0 Douglas    analista     45     1.85    70      M  20.45
1 Daniela    gerente      7     1.23    22      F  14.54
2 Pedro      coordenador    65     1.75    87      M  28.41
3 Maria      analista      64     1.67    64      F  22.95
4 Eduardo    gerente       42     1.82    96      M  28.98
5 Ester       diretor       37     1.73    68      F  22.72
```

✓ Convertendo dados de um atributo

```
1 ''
2 Imagine que você deseja converter os valores da coluna massa
3 que originalmente estão como tipo int em tipo float
4 ''
5 import pandas as pd
6 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 df['imc'] = round(df['peso']/df['altura']**2,2)
14 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
15 print('***** DF 01 - com massa tipo int *****')
16 print(df.dtypes)
17 df['massa'] = df['massa'].astype('float')
18 print('***** DF 01 - com massa tipo float *****')
19 print(df.dtypes)
```

✓ Reordenando a posição das colunas em um dataframe

```

1 '''
2 Imagine que você deseja converter os valores da coluna massa
3 que originalmente estão como tipo int em tipo float
4 '''
5 import pandas as pd
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 df['imc'] = round(df['peso']/df['altura']**2,2)
14 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
15 print('***** DF 01 - posição original *****')
16 display(df)
17 '''
18 vamos supor que desejamos colocar imc antes de genero
19 '''
20 df = df[['nome','idade','altura','massa','imc','genero']]
21 print('***** DF 01 - posição reordenada *****')
22 display(df)

```

✓ Criando distribuições de frequência

```

1 '''
2 Imagine que você deseja criar uma distribuição de frequência para o
3 atributo faixa etária
4 '''
5 import pandas as pd
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 df['imc'] = round(df['peso']/df['altura']**2,2)
14 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
15 faixas = ['01 a 10','11 - 20','21 - 30','31 - 40','41 - 50','51 - 60','61 - 70']
16 df['faixa_etaria'] = pd.cut(x=df['idade'],bins=[0,10,20,30,40,50,60,70],labels=faixas)
17 '''
18 O método cut, exige três argumentos:
19 1 - x: a coluna do df orginal que será usada como referência para criar a nova
20 2 - bins: uma lista com os valores que serão utilizado na variável x como ponto
21 de corte para a classificação das faixas
22 3 - labels: uma lista que contém todos os possíveis valores da nova coluna
23 '''
24 print('***** DF 01 - Trannsformado *****')
25 display(df)
26 df.groupby('faixa_etaria')['faixa_etaria'].count()

```

→ ***** DF 01 - Trannsformado *****

| | nome | idade | altura | massa | genero | imc | faixa_etaria |
|---|---------|-------|--------|-------|--------|-------|--------------|
| 0 | Douglas | 45 | 1.85 | 70 | M | 20.45 | 41 - 50 |
| 1 | Daniela | 7 | 1.23 | 22 | F | 14.54 | 01 a 10 |
| 2 | Pedro | 65 | 1.75 | 87 | M | 28.41 | 61 - 70 |
| 3 | Maria | 64 | 1.67 | 64 | F | 22.95 | 61 - 70 |
| 4 | Eduardo | 42 | 1.82 | 96 | M | 28.98 | 41 - 50 |
| 5 | Ester | 37 | 1.73 | 68 | F | 22.72 | 31 - 40 |

faixa_etaria

| | |
|---------|---|
| 01 a 10 | 1 |
| 11 - 20 | 0 |
| 21 - 30 | 0 |
| 31 - 40 | 1 |
| 41 - 50 | 2 |
| 51 - 60 | 0 |
| 61 - 70 | 2 |

Name: faixa_etaria, dtype: int64

```

1 import pandas as pd
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                     columns=['nome','idade','altura','peso','sexo'])
9 df['imc'] = round(df['peso']/df['altura']**2,2)
10 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
11 faixas = ['01 - 10','11 - 20','21 - 30','31 - 40','41 - 50','51 - 60','61 - 70']
12 df['faixa_etaria'] = pd.cut(x=df['idade'],bins=[0,10,20,30,40,50,60,70],labels=faixas)
13 df

```

| | nome | idade | altura | massa | genero | imc | faixa_etaria |
|---|---------|-------|--------|-------|--------|-------|--------------|
| 0 | Douglas | 45 | 1.85 | 70 | M | 20.45 | 41 - 50 |
| 1 | Daniela | 7 | 1.23 | 22 | F | 14.54 | 01 - 10 |
| 2 | Pedro | 65 | 1.75 | 87 | M | 28.41 | 61 - 70 |
| 3 | Maria | 64 | 1.67 | 64 | F | 22.95 | 61 - 70 |
| 4 | Eduardo | 42 | 1.82 | 96 | M | 28.98 | 41 - 50 |
| 5 | Ester | 37 | 1.73 | 68 | F | 22.72 | 31 - 40 |

```
1 tabela = df.groupby('faixa_etaria')['faixa_etaria'].count()
2 tabela
```

```
→ faixa_etaria
01 - 10    1
11 - 20    1
21 - 30    1
31 - 40    1
41 - 50    1
51 - 60    1
61 - 70    1
Name: faixa_etaria, dtype: int64
```

```
1 """
2 No bloco anterior, ao aplicar o groupby nós geramos um objeto que possui uma única
3 dimensão, mesmo mostrando "duas colunas", assim, se nosso objetivo for guardar
4 essas frequências em uma tabela, devemos fazer o seguinte:
5 """
6 import pandas as pd
7 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester','Zé']
8 idade = [45,7,65,54,25,37,16]
9 altura = [1.85,1.23,1.75,1.67,1.82,1.73,1.68]
10 peso = [70,22,87,64,96,68,70]
11 sexo = ['M','F','M','F','M','F','M']
12 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
13                     columns=['nome','idade','altura','peso','sexo'])
14 df['imc'] = round(df['peso']/df['altura']**2,2)
15 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
16 faixas = ['01 - 10','11 - 20','21 - 30','31 - 40','41 - 50','51 - 60','61 - 70']
17 df['faixa_etaria'] = pd.cut(x=df['idade'],bins=[0,10,20,30,40,50,60,70],labels=faixas)
18 """
19 1 - gerar uma lista com os valores únicos da variável que estamos contando e
20 armazená-la em uma variável
21 """
22 xpto = df['faixa_etaria'].unique()
23
24 """
25 2 - Realizar o groupby e armazená-lo em uma variável
26 """
27 freq = df.groupby('faixa_etaria')['faixa_etaria'].count()
28
29 """
30 3 - Criar um dataframe com as duas variáveis armazenadas anteriormente
31 """
32 df3 = pd.DataFrame(zip(xpto,freq),columns=['Faixa Etária','Frequência'])
33 display(df3)
34 df3 = df3.sort_values(by='Faixa Etária')
35 display(df3)
36 """
37 Nesse processo temo um PROBLEMA:
38 ao gerar um objeto pelo .unique() teremos uma lista só com os valores únicos
39 que aparecem na variável faixa etária.
40 No entanto quando geramos o segundo objeto pelo .groupby podemos ter uma lista
41 maior que a lista gerada pelo unique, pois o .groupby contará também as categorias
42 listadas cuja frequencia for 0 , ou seja, aqueles que não aparecerão na lista do
43 .unique, e como resultado teremos duas listas que terão tamanhos diferentes
44 e ao usarmos o .pd.DataFrame a união será feita com base na lista de menos elementos
45 """
```

→ Faixa Etária Frequência

| | Faixa Etária | Frequência |
|---|--------------|------------|
| 0 | 41 - 50 | 1 |
| 1 | 01 - 10 | 1 |
| 2 | 61 - 70 | 1 |
| 3 | 51 - 60 | 1 |
| 4 | 21 - 30 | 1 |
| 5 | 31 - 40 | 1 |
| 6 | 11 - 20 | 1 |

→ Faixa Etária Frequência

| | Faixa Etária | Frequência |
|---|--------------|------------|
| 1 | 01 - 10 | 1 |
| 6 | 11 - 20 | 1 |
| 4 | 21 - 30 | 1 |
| 5 | 31 - 40 | 1 |
| 0 | 41 - 50 | 1 |
| 3 | 51 - 60 | 1 |
| 2 | 61 - 70 | 1 |

1 ''''

2 Exercício:

3 faça o mesmo procedimento do exercício anterior para a variável genero

4 ''''

5 import pandas as pd

6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']

7 idade = [45,7,65,64,42,37]

8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]

9 peso = [70,22,87,64,96,68]

10 sexo = ['M','F','M','F','M','F']

11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),

12 columns=['nome','idade','altura','peso','sexo'])

13 df['imc'] = round(df['peso']/df['altura']**2,2)

14 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)

15 faixas = ['01 - 10','11 - 20','21 - 30','31 - 40','41 - 50','51 - 60','61 - 70']

16 df['faixa_etaria'] = pd.cut(x=df['idade'],bins=[0,10,20,30,40,50,60,70],labels=faixas)

17 ''''

18 1 - gerar uma lista com os valores únicos da variável que estamos contando e

19 armazená-la em uma variável

20 ''''

21 gender = df['genero'].unique()

22

23 ''''

24 2 - Realizar o groupby e armazená-lo em uma variável

25 ''''

26 freq = df.groupby('genero')['genero'].count()

27

28 ''''

29 3 - Criar um dataframe com as duas variáveis armazenadas anteriormente

30 ''''

31 df4 = pd.DataFrame(zip(gender,freq),columns=['Gênero','Frequência'])

32 display(df4)

33 df4 = df4.sort_values(by='Gênero')

34 display(df4)

→ Gênero Frequência

| | Gênero | Frequência |
|---|--------|------------|
| 0 | M | 3 |
| 1 | F | 3 |

→ Gênero Frequência

| | Gênero | Frequência |
|---|--------|------------|
| 1 | F | 3 |
| 0 | M | 3 |

```

1 '''
2 Exercício:
3 faça o mesmo procedimento do exercício anterior para a variável nome
4 '''
5 import pandas as pd
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 df['imc'] = round(df['peso']/df['altura']**2,2)
14 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
15 faixas = ['01 - 10','11 - 20','21 - 30','31 - 40','41 - 50','51 - 60','61 - 70']
16 df['faixa_etaria'] = pd.cut(x=df['idade'],bins=[0,10,20,30,40,50,60,70],labels=faixas)
17 '''
18 1 - gerar uma lista com os valores únicos da variável que estamos contando e
19 armazená-la em uma variável
20 '''
21 gender = df['nome'].unique()
22
23 '''
24 2 - Realizar o groupby e armazená-lo em uma variável
25 '''
26 freq = df.groupby('nome')['nome'].count()
27
28 '''
29 3 - Criar um dataframe com as duas variáveis armazenadas anteriormente
30 '''
31 df5 = pd.DataFrame(zip(gender,freq),columns=['Nome','Frequência'])
32 display(df5)
33 df5 = df5.sort_values(by='Nome')
34 display(df5)

```

 Nome Frequência

| | | |
|---|---------|---|
| 0 | Douglas | 1 |
| 1 | Daniela | 1 |
| 2 | Pedro | 1 |
| 3 | Maria | 1 |
| 4 | Eduardo | 1 |
| 5 | Ester | 1 |

Nome Frequência

| | | |
|---|---------|---|
| 1 | Daniela | 1 |
| 0 | Douglas | 1 |
| 4 | Eduardo | 1 |
| 5 | Ester | 1 |
| 3 | Maria | 1 |
| 2 | Pedro | 1 |

```

1 ''
2 Agora vamos ver um método para criar a coluna de frequênciapercentual
3 tendo como referênciada coluna frequênci
4 ''
5
6 import pandas as pd
7 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
8 idade = [45,7,65,64,42,37]
9 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
10 peso = [70,22,87,64,96,68]
11 sexo = ['M', 'F', 'M', 'F', 'M', 'F']
12 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
13                     columns=['nome','idade','altura','peso','sexo'])
14 df['imc'] = round(df['peso']/df['altura']**2,2)
15 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
16 faixas = ['01 - 10', '11 - 20', '21 - 30', '31 - 40', '41 - 50', '51 - 60', '61 - 70']
17 df['faixa_etaria'] = pd.cut(x=df['idade'],bins=[0,10,20,30,40,50,60,70],labels=faixas)
18 ''
19 1 - gerar uma lista com os valores únicos da variável que estamos contando e
20 armazená-la em uma variável
21 ''
22 gender = df['nome'].unique()
23
24 ''
25 2 - Realizar o groupby e armazená-lo em uma variável
26 ''
27 freq = df.groupby('nome')['nome'].count()
28
29 ''
30 3 - Criar um dataframe com as duas variáveis armazenadas anteriormente
31 ''
32
33 ''
34 4 - Criar a coluna de frquencia percentual
35 ''
36
37 df5 = pd.DataFrame(zip(gender,freq),columns=['Nome','Frequênci'])
38 #display(df5)
39 df5 = df5.sort_values(by='Nome')
40 #display(df5)
41 df5['Frequênci %'] = round((df5['Frequênci']/df5['Frequênci'].sum())*100,2)
42 df5

```

| | Nome | Frequênci | Frequênci % |
|---|---------|-----------|-------------|
| 1 | Daniela | 1 | 16.67 |
| 0 | Douglas | 1 | 16.67 |
| 4 | Eduardo | 1 | 16.67 |
| 5 | Ester | 1 | 16.67 |
| 3 | Maria | 1 | 16.67 |
| 2 | Pedro | 1 | 16.67 |

```

1 ''
2 Calculando a freuência percentual para a variável gênero
3 ''
4 import pandas as pd
5 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
6 idade = [45,7,65,64,42,37]
7 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
8 peso = [70,22,87,64,96,68]
9 sexo = ['M', 'F', 'M', 'F', 'M', 'F']
10 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
11                     columns=['nome','idade','altura','peso','sexo'])
12 df['imc'] = round(df['peso']/df['altura']**2,2)
13 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
14 faixas = ['01 - 10', '11 - 20', '21 - 30', '31 - 40', '41 - 50', '51 - 60', '61 - 70']
15 df['faixa_etaria'] = pd.cut(x=df['idade'],bins=[0,10,20,30,40,50,60,70],labels=faixas)
16 ''
17 1 - gerar uma lista com os valores únicos da variável que estamos contando e
18 armazená-la em uma variável
19 ''
20 gender = df['genero'].unique()
21
22 ''
23 2 - Realizar o groupby e armazená-lo em uma variável
24 ''
25 freq = df.groupby('genero')['genero'].count()
26
27 ''
28 3 - Criar um dataframe com as duas variáveis armazenadas anteriormente
29 ''
30 df6 = pd.DataFrame(zip(gender,freq),columns=['Gênero','Frequênci'])
31 df6 = df6.sort_values(by='Gênero')
32 df6['Frequênci %'] = round((df6['Frequênci']/df6['Frequênci'].sum())*100,2)
33 df6.loc[2] = ['Total',df6['Frequênci'].sum(),df6['Frequênci %'].sum()]
34 df6

```

| | Gênero | Frequência | Frequência % |
|---|--------|------------|--------------|
| 1 | F | 3 | 50.0 |
| 0 | M | 3 | 50.0 |
| 2 | Total | 6 | 100.0 |

```

1 '''
2 Exercício:
3 faça o mesmo procedimento do exercício anterior para a variável faixa etária
4 '''
5 import pandas as pd
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester','Zé']
7 idade = [45,7,65,54,25,37,16]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73,1.68]
9 peso = [70,22,87,64,96,68,70]
10 sexo = ['M','F','M','F','M']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 df['imc'] = round(df['peso']/df['altura']**2,2)
14 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
15 faixas = ['01 - 10','11 - 20','21 - 30','31 - 40','41 - 50','51 - 60','61 - 70']
16 df['faixa_etaria'] = pd.cut(x=df['idade'],bins=[0,10,20,30,40,50,60,70],labels=faixas)
17 '''
18 1 - gerar uma lista com os valores únicos da variável que estamos contando e
19 armazená-la em uma variável
20 '''
21 xpto = df['faixa_etaria'].unique()
22
23 '''
24 2 - Realizar o groupby e armazená-lo em uma variável
25 '''
26 freq = df.groupby('faixa_etaria')['faixa_etaria'].count()
27
28 '''
29 3 - Criar um dataframe com as duas variáveis armazenadas anteriormente
30 '''
31 df7 = pd.DataFrame(zip(xpto,freq),columns=['Faixa Etária','Frequência'])
32 display(df7)
33 df7 = df7.sort_values(by='Faixa Etária')
34 display(df7)
35 df7['Frequência %'] = round((df7['Frequência']/df7['Frequência'].sum())*100,2)
36 df7.loc[7] = ['Total',df7['Frequência'].sum(),df7['Frequência %'].sum()]
37 display(df7)

```

→ Faixa Etária Frequência

| | | |
|---|---------|---|
| 0 | 41 - 50 | 1 |
| 1 | 01 - 10 | 1 |
| 2 | 61 - 70 | 1 |
| 3 | 51 - 60 | 1 |
| 4 | 21 - 30 | 1 |
| 5 | 31 - 40 | 1 |
| 6 | 11 - 20 | 1 |

Faixa Etária Frequência

| | | |
|---|---------|---|
| 1 | 01 - 10 | 1 |
| 6 | 11 - 20 | 1 |
| 4 | 21 - 30 | 1 |
| 5 | 31 - 40 | 1 |
| 0 | 41 - 50 | 1 |
| 3 | 51 - 60 | 1 |
| 2 | 61 - 70 | 1 |

Faixa Etária Frequência Frequência %

| | | | |
|---|---------|---|--------|
| 1 | 01 - 10 | 1 | 14.29 |
| 6 | 11 - 20 | 1 | 14.29 |
| 4 | 21 - 30 | 1 | 14.29 |
| 5 | 31 - 40 | 1 | 14.29 |
| 0 | 41 - 50 | 1 | 14.29 |
| 3 | 51 - 60 | 1 | 14.29 |
| 2 | 61 - 70 | 1 | 14.29 |
| 7 | Total | 7 | 100.03 |

```

1 '''
2 Recapitulando o problema do uso do groupby para criar distribuições de
3 frequências para variáveis quantitativas
4 '''
5
6 '''
7 Nesse processo temo um PROBLEMA:
8 ao gerar um objeto pelo .unique() teremos uma lista só com os valores únicos
9 que aparecem na variável faixa etária.
10 No entanto quando geramos o segundo objeto pelo .groupby podemos ter uma lista
11 maior que a lista gerada pelo unique, pois o .groupby contará também as categorias
12 listadas cuja frequência for 0 , ou seja, aqueles que não aparecerão na lista do
13 .unique, e como resultado teremos duas listas que terão tamanhos diferentes
14 e ao usarmos o .pd.DataFrame a união será feita com base na lista de menos elementos
15 '''
16 import pandas as pd
17 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
18 idade = [45,7,65,64,42,37]
19 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
20 peso = [70,22,87,64,96,68]
21 sexo = ['M','F','M','F','M','F']
22 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
23                     columns=['nome','idade','altura','peso','sexo'])
24 df['imc'] = round(df['peso']/df['altura']**2,2)
25 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
26 faixas = ['01 - 10','11 - 20','21 - 30','31 - 40','41 - 50','51 - 60','61 - 70']
27 df['faixa_etaria'] = pd.cut(x=df['idade'],bins=[0,10,20,30,40,50,60,70],labels=faixas)
28 freq = df['faixa_etaria'].value_counts()
29 display(freq)
30 percentual = df['faixa_etaria'].value_counts(normalize=True)*100
31 display(percentual)

```

→ faixa_etaria

| | |
|---------|---|
| 41 - 50 | 2 |
| 61 - 70 | 2 |
| 01 - 10 | 1 |
| 31 - 40 | 1 |
| 11 - 20 | 0 |
| 21 - 30 | 0 |
| 51 - 60 | 0 |

Name: count, dtype: int64

faixa_etaria

| | |
|---------|-----------|
| 41 - 50 | 33.333333 |
| 61 - 70 | 33.333333 |
| 01 - 10 | 16.666667 |
| 31 - 40 | 16.666667 |
| 11 - 20 | 0.000000 |
| 21 - 30 | 0.000000 |
| 51 - 60 | 0.000000 |

Name: proportion, dtype: float64

```

1 '''
2 Vamos criar a frequência percentual para a variável gênero pelo método
3 value_counts
4 '''
5 import pandas as pd
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 df['imc'] = round(df['peso']/df['altura']**2,2)
14 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
15 faixas = ['01 - 10','11 - 20','21 - 30','31 - 40','41 - 50','51 - 60','61 - 70']
16 df['faixa_etaria'] = pd.cut(x=df['idade'],bins=[0,10,20,30,40,50,60,70],labels=faixas)
17 freq = df['genero'].value_counts()
18 percentual = df['genero'].value_counts(normalize=True)*100
19 generos = pd.DataFrame({'Frequência Absoluta':freq,'Frequência Percentual':percentual})
20 generos.reset_index(inplace=True)
21 generos.loc[2] = ['TOTAL',generos['Frequência Absoluta'].sum(),
22                   generos['Frequência Percentual'].sum()]
23 display(generos)

```

→ genero Frequência Absoluta Frequência Percentual

| genero | Frequência Absoluta | Frequência Percentual |
|---------|---------------------|-----------------------|
| 0 M | 3 | 50.0 |
| 1 F | 3 | 50.0 |
| 2 TOTAL | 6 | 100.0 |

```

1 '''
2 Vamos criar a frequência percentual para a variável nome pelo método
3 value_counts
4 '''
5 import pandas as pd
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 df['imc'] = round(df['peso']/df['altura']**2,2)
14 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
15 faixas = ['01 - 10','11 - 20','21 - 30','31 - 40','41 - 50','51 - 60','61 - 70']
16 df['faixa_etaria'] = pd.cut(x=df['idade'],bins=[0,10,20,30,40,50,60,70],labels=faixas)
17 freq = df['nome'].value_counts() # aqui estamos criando um objeto com a contagem de todos os valores únicos da variável de interesse
18 percentual = df['nome'].value_counts(normalize=True)*100 # aqui estamos criando um objeto com a contagem percentual de todos os valores únicos da va
19 nomes = pd.DataFrame({'Frequência Absoluta':freq,'Frequência Percentual':percentual}) # aqui estamos criando um dataframe com os dois objetos criado
20 nomes.reset_index(inplace=True)
21 nomes.loc[6] = ['TOTAL',nomes['Frequência Absoluta'].sum(),
22                  nomes['Frequência Percentual'].sum()] # Aqui estamos inserindo uma linha de total na tabela de frequências
23 display(nomes)

```

| | nome | Frequência Absoluta | Frequência Percentual |
|---|---------|---------------------|-----------------------|
| 0 | Douglas | 1 | 16.666667 |
| 1 | Daniela | 1 | 16.666667 |
| 2 | Pedro | 1 | 16.666667 |
| 3 | Maria | 1 | 16.666667 |
| 4 | Eduardo | 1 | 16.666667 |
| 5 | Ester | 1 | 16.666667 |
| 6 | TOTAL | 6 | 100.000000 |

```

1 '''
2 Vamos criar a frequência percentual para a variável faixa etária pelo método
3 value_counts
4 '''
5 import pandas as pd
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                     columns=['nome','idade','altura','peso','sexo'])
13 df['imc'] = round(df['peso']/df['altura']**2,2)
14 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
15 faixas = ['01 - 10','11 - 20','21 - 30','31 - 40','41 - 50','51 - 60','61 - 70']
16 df['faixa_etaria'] = pd.cut(x=df['idade'],bins=[0,10,20,30,40,50,60,70],labels=faixas)
17 freq = df['faixa_etaria'].value_counts() # aqui estamos criando um objeto com a contagem de todos os valores únicos da variável de interesse
18 percentual = df['faixa_etaria'].value_counts(normalize=True)*100 # aqui estamos criando um objeto com a contagem percentual de todos os valores únic
19 idades = pd.DataFrame({'Frequência Absoluta':freq,'Frequência Percentual':percentual}) # aqui estamos criando um dataframe com os dois objetos criad
20 idades = idades.sort_values(by='faixa_etaria')
21 idades.reset_index(inplace=True)
22
23 '''
24 Quando temos uma variável quantitativa a distribuição de frequência deve ter
25 duas colunas a mais:
26 1 - Frequencia acumulada
27 2 - Frequencia acumulada %
28 '''
29 idades['Frequência Acumulada'] = idades['Frequência Absoluta'].cumsum() # Esta linha cria a coluna de frequencia acumulada
30 idades['Frequência Acumulada %'] = (idades['Frequência Acumulada']/idades['Frequência Absoluta'].sum())*100 # Esta linha cria a coluna de frequencia
31
32 idades.loc[7] = ['TOTAL',idades['Frequência Absoluta'].sum(),
33                  idades['Frequência Percentual'].sum(),'-','-'] # Aqui estamos inserindo uma linha de total na tabela de frequências
34 display(idades)
35
36

```

| | faixa_etaria | Frequência Absoluta | Frequência Percentual | Frequência Acumulada | Frequência Acumulada % |
|---|--------------|---------------------|-----------------------|----------------------|------------------------|
| 0 | 01 - 10 | 1 | 16.666667 | 1 | 16.666667 |
| 1 | 11 - 20 | 0 | 0.000000 | 1 | 16.666667 |
| 2 | 21 - 30 | 0 | 0.000000 | 1 | 16.666667 |
| 3 | 31 - 40 | 1 | 16.666667 | 2 | 33.333333 |
| 4 | 41 - 50 | 2 | 33.333333 | 4 | 66.666667 |
| 5 | 51 - 60 | 0 | 0.000000 | 4 | 66.666667 |
| 6 | 61 - 70 | 2 | 33.333333 | 6 | 100.0 |
| 7 | TOTAL | 6 | 100.000000 | - | - |

```

1 '''
2 considerando a base de dados a segui, construa as distribuções de frequências
3 adequadas para cada tipo de variável, no caso da variável faturamento
4 no pandas construa uma nova coluna classificado o faturamento por faixas
5 sendo as seguintes faixas:
6 [10001 - 12000, 12001 - 14000, 14001 - 16000, 16001 - 18000, 18001 - 20000]
7 '''
8 df = pd.read_csv('https://docs.google.com/spreadsheets/d/e/2PACX-1vTFGertCc67cL9ojU5rRq8qyCBojGKrH7m2zU7hqYJ1AKRm-oBpTNjRhGztmhYrzJApq3-FNbxBEPz3/pu
9 display(df)

```

✓ Gabarito do exercício

```

1 import pandas as pd
2 df = pd.read_csv('https://docs.google.com/spreadsheets/d/e/2PACX-1vTFGertCc67cL9ojU5rRq8qyCBojGKrH7m2zU7hqYJ1AKRm-oBpTNjRhGztmhYrzJApq3-FNbxBEPz3/pu
3 freq = df['genero'].value_counts() # aqui estamos criando um objeto com a contagem de todos os valores únicos da variável de interesse
4 percentual = df['genero'].value_counts(normalize=True)*100 # aqui estamos criando um objeto com a contagem percentual de todos os valores únicos da
5 generos = pd.DataFrame({'Frequência Absoluta':freq,'Frequência Percentual':percentual}) # aqui estamos criando um dataframe com os dois objetos cria
6 generos.reset_index(inplace=True)
7 generos.loc[2] = ['TOTAL',generos['Frequência Absoluta'].sum(),
8                 generos['Frequência Percentual'].sum()] # Aqui estamos inserindo uma linha de total na tabela de frequências
9 display(generos)

```

| | genero | Frequência Absoluta | Frequência Percentual |
|---|--------|---------------------|-----------------------|
| 0 | f | 11 | 55.0 |
| 1 | m | 9 | 45.0 |
| 2 | TOTAL | 20 | 100.0 |

```

1 import pandas as pd
2 df = pd.read_csv('https://docs.google.com/spreadsheets/d/e/2PACX-1vTFGertCc67cL9ojU5rRq8qyCBojGKrH7m2zU7hqYJ1AKRm-oBpTNjRhGztmhYrzJApq3-FNbxBEPz3/pu
3 freq = df['departamento'].value_counts() # aqui estamos criando um objeto com a contagem de todos os valores únicos da variável de interesse
4 percentual = df['departamento'].value_counts(normalize=True)*100 # aqui estamos criando um objeto com a contagem percentual de todos os valores únicos
5 departamentos = pd.DataFrame({'Frequência Absoluta':freq,'Frequência Percentual':percentual}) # aqui estamos criando um dataframe com os dois objeo
6 departamentos.reset_index(inplace=True)
7 departamentos.loc[3] = ['TOTAL',departamentos['Frequência Absoluta'].sum(),
8                         departamentos['Frequência Percentual'].sum()] # Aqui estamos inserindo uma linha de total na tabela de frequências
9 display(departamentos)

```

| | departamento | Frequência Absoluta | Frequência Percentual |
|---|--------------|---------------------|-----------------------|
| 0 | eletronico | 8 | 40.0 |
| 1 | ud | 6 | 30.0 |
| 2 | info | 6 | 30.0 |
| 3 | TOTAL | 20 | 100.0 |

```

1 import pandas as pd
2 df = pd.read_csv('https://docs.google.com/spreadsheets/d/e/2PACX-1vTFGertCc67cL9ojU5rRq8qyCBojGKrH7m2zU7hqYJ1AKRm-oBpTNjRhGztmhYrzJApq3-FNbxBEPz3/pu
3 freq = df['uf'].value_counts() # aqui estamos criando um objeto com a contagem de todos os valores únicos da variável de interesse
4 percentual = df['uf'].value_counts(normalize=True)*100 # aqui estamos criando um objeto com a contagem percentual de todos os valores únicos da vari
5 ufs = pd.DataFrame({'Frequência Absoluta':freq,'Frequência Percentual':percentual}) # aqui estamos criando um dataframe com os dois objetos criados
6 ufs.reset_index(inplace=True)
7 ufs.loc[11] = ['TOTAL',ufs['Frequência Absoluta'].sum(),
8                 ufs['Frequência Percentual'].sum()] # Aqui estamos inserindo uma linha de total na tabela de frequências
9 display(ufs)

```

| | uf | Frequência Absoluta | Frequência Percentual |
|----|-------|---------------------|-----------------------|
| 0 | sp | 6 | 30.0 |
| 1 | mg | 3 | 15.0 |
| 2 | ba | 3 | 15.0 |
| 3 | rj | 1 | 5.0 |
| 4 | ce | 1 | 5.0 |
| 5 | al | 1 | 5.0 |
| 6 | pe | 1 | 5.0 |
| 7 | sc | 1 | 5.0 |
| 8 | rs | 1 | 5.0 |
| 9 | pr | 1 | 5.0 |
| 10 | es | 1 | 5.0 |
| 11 | TOTAL | 20 | 100.0 |

```

1 '''
2 Código para contar a quantidade de opções únicas de um atributo
3 '''
4 est = df['uf'].unique()
5 len(est)

→ 11

1 import pandas as pd
2 df = pd.read_csv('https://docs.google.com/spreadsheets/d/e/2PACX-1vTFGertCc67cL9ojU5rRq8qyCBojGKrH7m2zU7hqYJ1AKRm-oBpTNjRhGztmhYrzJApq3-FNbxBEPz3/pu
3 faixas = ['10001 - 12000', '12001 - 14000', '14001 - 16000', '16001 - 18000', '18001 - 20000']
4 df['faturamentos'] = pd.cut(x=df['faturamento'], bins=[10000, 12000, 14000, 16000, 18000, 20000], labels=faixas)
5 freq = df['faturamentos'].value_counts() # aqui estamos criando um objeto com a contagem de todos os valores únicos da variável de interesse
6 percentual = df['faturamentos'].value_counts(normalize=True)*100 # aqui estamos criando um objeto com a contagem percentual de todos os valores únicos
7 fatur = pd.DataFrame({'Frequência Absoluta':freq,'Frequência Percentual':percentual}) # aqui estamos criando um dataframe com os dois objetos criados
8 fatur = fatur.sort_values(by='faturamentos')
9 fatur.reset_index(inplace=True)
10
11 '''
12 Quando temos uma variável quantitativa a distribuição de frequência deve ter
13 duas colunas a mais:
14 1 - Frequência acumulada
15 2 - Frequência acumulada %
16 '''
17 fatur['Frequência Acumulada'] = fatur['Frequência Absoluta'].cumsum() # Esta linha cria a coluna de frequencia acumulada
18 fatur['Frequência Acumulada %'] = (fatur['Frequência Acumulada']/fatur['Frequência Absoluta'].sum())*100 # Esta linha cria a coluna de frequencia acumulada %
19
20 fatur.loc[5] = ['TOTAL',fatur['Frequência Absoluta'].sum(),
21                 fatur['Frequência Percentual'].sum(),'-','-' ] # Aqui estamos inserindo uma linha de total na tabela de frequências
22 display(fatur)

```

| | faturamentos | Frequência Absoluta | Frequência Percentual | Frequência Acumulada | Frequência Acumulada % |
|---|---------------|---------------------|-----------------------|----------------------|------------------------|
| 0 | 10001 - 12000 | 3 | 15.0 | 3 | 15.0 |
| 1 | 12001 - 14000 | 4 | 20.0 | 7 | 35.0 |
| 2 | 14001 - 16000 | 2 | 10.0 | 9 | 45.0 |
| 3 | 16001 - 18000 | 5 | 25.0 | 14 | 70.0 |
| 4 | 18001 - 20000 | 6 | 30.0 | 20 | 100.0 |
| 5 | TOTAL | 20 | 100.0 | - | - |

▼ Operações Especiais com Groupby - 29/04/2024

```

1 import pandas as pd
2 df = pd.read_csv('https://docs.google.com/spreadsheets/d/e/2PACX-1vTFGertCc67cL9ojU5rRq8qyCBojGKrH7m2zU7hqYJ1AKRm-oBpTNjRhGztmhYrzJApq3-FNbxBEPz3/pu
3 group = df.groupby('departamento')

1 group = df.groupby('departamento')['faturamento']

1 group.mean()

→ departamento
eletro    14511.625000
info     17443.666667
ud       15603.333333
Name: faturamento, dtype: float64

1 import pandas as pd
2 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
3 idade = [45,7,65,64,42,37]
4 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
5 peso = [70,22,87,64,96,68]
6 sexo = ['M','F','M','F','M','F']
7 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
8                   columns=['nome','idade','altura','peso','sexo'])
9 df['imc'] = round(df['peso']/df['altura']**2,2)
10 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
11 faixas = ['01 - 10','11 - 20','21 - 30','31 - 40','41 - 50','51 - 60','61 - 70']
12 df['faixa_etaria'] = pd.cut(x=df['idade'],bins=[0,10,20,30,40,50,60,70],labels=faixas)
13 df

```

| | nome | idade | altura | massa | genero | imc | faixa_etaria |
|---|---------|-------|--------|-------|--------|-------|--------------|
| 0 | Douglas | 45 | 1.85 | 70 | M | 20.45 | 41 - 50 |
| 1 | Daniela | 7 | 1.23 | 22 | F | 14.54 | 01 - 10 |
| 2 | Pedro | 65 | 1.75 | 87 | M | 28.41 | 61 - 70 |
| 3 | Maria | 64 | 1.67 | 64 | F | 22.95 | 61 - 70 |
| 4 | Eduardo | 42 | 1.82 | 96 | M | 28.98 | 41 - 50 |
| 5 | Ester | 37 | 1.73 | 68 | F | 22.72 | 31 - 40 |

```

1 agrupamento = df.groupby('genero')[['massa','altura','idade']]
2 display(agrupamento)
→ <pandas.core.groupby.generic.DataFrameGroupBy object at 0x7aba4d2f2f80>

1 agrupamento.mean()
→
    massa    altura    idade
genero
_____
F      51.333333  1.543333  36.000000
M      84.333333  1.806667  50.666667

1 agrupamento.count()
→
    massa  altura  idade
genero
_____
F        3       3       3
M        3       3       3

1 '''
2 Utilizando o conceito de agrupamento faça uma distribuição de frequência das
3 faixas etárias
4 '''
5 import pandas as pd
6 nome = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
7 idade = [45,7,65,64,42,37]
8 altura = [1.85,1.23,1.75,1.67,1.82,1.73]
9 peso = [70,22,87,64,96,68]
10 sexo = ['M','F','M','F','M','F']
11 df = pd.DataFrame(zip(nome,idade,altura,peso,sexo),
12                      columns=['nome','idade','altura','peso','sexo'])
13 df['imc'] = round(df['peso']/df['altura']**2,2)
14 df.rename(columns={'peso':'massa','sexo':'genero'},inplace=True)
15 faixas = ['01 - 10','11 - 20','21 - 30','31 - 40','41 - 50','51 - 60','61 - 70']
16 df['faixa_etaria'] = pd.cut(x=df['idade'],bins=[0,10,20,30,40,50,60,70],labels=faixas)
17 idades = df.groupby('faixa_etaria')['idade']
18 idades.count()

→ faixa_etaria
01 - 10    1
11 - 20    0
21 - 30    0
31 - 40    1
41 - 50    2
51 - 60    0
61 - 70    2
Name: idade, dtype: int64

1 '''
2 agora imagine que você deseja saber a idade mínima de cada grupo, masculino e
3 feminino
4 '''
5 agrupamento.min()
→
    massa  altura  idade
genero
_____
F      22     1.23     7
M      70     1.75    42

1 '''
2 Imagine que você deseja saber qual é o valor que separa os 25% menores dos 75%
3 maiores valores para as colunas massa, genero e idade
4 '''
5 '''
6 nesse caso nosso objetivo é encontrar o primeiro quartil desses atributos,
7 agrupados por gêneros
8 '''
9 agrupamento.quantile(0.25)

→
    massa  altura  idade
genero
_____
F      43.0    1.450   22.0
M      78.5    1.785   43.5

```

```

1 ''
2 Imagine que você deseja saber qual é o valor que separa os 50% menores dos 50%
3 maiores valores para as colunas massa, genero e idade
4 ''
5 ''
6 nesse caso nosso objetivo é encontrar o primeiro quartil desses atributos,
7 agrupados por gêneros
8 ''
9 agrupamento.quantile(0.5)

```

→ massas altura idade

| genero | | | |
|--------|------|------|------|
| F | 64.0 | 1.67 | 37.0 |
| M | 87.0 | 1.82 | 45.0 |

```

1 ''
2 Imagine que você deseja saber qual é o valor que separa os 50% menores dos 50%
3 maiores valores para as colunas massa, genero e idade
4 ''
5 ''
6 nesse caso nosso objetivo é encontrar o primeiro quartil desses atributos,
7 agrupados por gêneros
8 ''
9 agrupamento.median()

```

→ massas altura idade

| genero | | | |
|--------|------|------|------|
| F | 64.0 | 1.67 | 37.0 |
| M | 87.0 | 1.82 | 45.0 |

```

1 ''
2 Imagine que você deseja saber qual é o valor que separa os 75% menores dos 25%
3 maiores valores para as colunas massa, genero e idade
4 ''
5 ''
6 nesse caso nosso objetivo é encontrar o primeiro quartil desses atributos,
7 agrupados por gêneros
8 ''
9 agrupamento.quantile(0.75)

```

→ massas altura idade

| genero | | | |
|--------|------|-------|------|
| F | 66.0 | 1.700 | 50.5 |
| M | 91.5 | 1.835 | 55.0 |

```

1 import pandas as pd
2 nome = ['Douglas', 'Daniela', 'Douglas', 'Maria', 'Douglas', 'Ester', 'Douglas', 'Ana',
3         'Douglas', 'Juliana', 'Celia']
4 idades = [21, 25, 30, 35, 35, 40, 41, 42, 43, 45, 52]
5 dependentes = [1,0,2,2,2,3,1,2,0,4,2]
6 salario = [1500,9000,1500,8000,2000,5000,2000,7000,3500,9000,11000]
7 cargo = ['analista', 'gerente', 'analista', 'gerente', 'analista', 'coordenadora',
8           'analista', 'coordenadora', 'analista', 'diretora', 'diretora']
9 sexo = ['M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'F']
10 df = pd.DataFrame(zip(nome, idades, dependentes, salario, cargo, sexo),
11                     columns=['nome', 'idades', 'dependentes', 'salario', 'cargo', 'sexo'])
12 df

```

→ nome idades dependentes salario cargo sexo

| 0 | Douglas | 21 | 1 | 1500 | analista | M |
|----|---------|----|---|-------|--------------|---|
| 1 | Daniela | 25 | 0 | 9000 | gerente | F |
| 2 | Douglas | 30 | 2 | 1500 | analista | M |
| 3 | Maria | 35 | 2 | 8000 | gerente | F |
| 4 | Douglas | 35 | 2 | 2000 | analista | M |
| 5 | Ester | 40 | 3 | 5000 | coordenadora | F |
| 6 | Douglas | 41 | 1 | 2000 | analista | M |
| 7 | Ana | 42 | 2 | 7000 | coordenadora | F |
| 8 | Douglas | 43 | 0 | 3500 | analista | M |
| 9 | Juliana | 45 | 4 | 9000 | diretora | F |
| 10 | Celia | 52 | 2 | 11000 | diretora | F |

```
1 cargos = df.groupby('cargo')[['idades', 'salario']]
```

```
1 '''
2 Neste bloco temos a média de idades e salário agrupada por cargo
3 '''
4 cargos.mean()
```

→ idades salario

| cargo | idade | salario |
|--------------|-------|---------|
| analista | 34.0 | 2100.0 |
| coordenadora | 41.0 | 6000.0 |
| diretora | 48.5 | 10000.0 |
| gerente | 30.0 | 8500.0 |

```
1 '''
2 Neste bloco temos a mediana de idades e salário agrupada por cargo
3 '''
4 cargos.median()
```

→ idades salario

| cargo | idade | salario |
|--------------|-------|---------|
| analista | 35.0 | 2000.0 |
| coordenadora | 41.0 | 6000.0 |
| diretora | 48.5 | 10000.0 |
| gerente | 30.0 | 8500.0 |

```
1 idades_cargos = df.groupby('cargo')[['idades']]
```

```
1 idades_cargos.describe()
```

→ idades

| cargo | count | mean | std | min | 25% | 50% | 75% | max |
|--------------|-------|------|----------|------|-------|------|-------|------|
| analista | 5.0 | 34.0 | 8.888194 | 21.0 | 30.00 | 35.0 | 41.00 | 43.0 |
| coordenadora | 2.0 | 41.0 | 1.414214 | 40.0 | 40.50 | 41.0 | 41.50 | 42.0 |
| diretora | 2.0 | 48.5 | 4.949747 | 45.0 | 46.75 | 48.5 | 50.25 | 52.0 |
| gerente | 2.0 | 30.0 | 7.071068 | 25.0 | 27.50 | 30.0 | 32.50 | 35.0 |

```
1 '''
2 Imagine que você deseja calcular as medidas descritivas agrupadas por cargos
3 tanto do atributo idades como do atributo salario
4 '''
5 idades_salarios_cargos = df.groupby('cargo')[['idades', 'salario']]
6 idades_salarios_cargos.describe()
```

→ idades salario

| cargo | idades | salario | | | | | | | | | | |
|--------------|--------|---------|----------|------|-------|------|-------|------|-------|---------|-------------|-----|
| | count | mean | std | min | 25% | 50% | 75% | max | count | mean | std | min |
| analista | 5.0 | 34.0 | 8.888194 | 21.0 | 30.00 | 35.0 | 41.00 | 43.0 | 5.0 | 2100.0 | 821.583836 | 15 |
| coordenadora | 2.0 | 41.0 | 1.414214 | 40.0 | 40.50 | 41.0 | 41.50 | 42.0 | 2.0 | 6000.0 | 1414.213562 | 50 |
| diretora | 2.0 | 48.5 | 4.949747 | 45.0 | 46.75 | 48.5 | 50.25 | 52.0 | 2.0 | 10000.0 | 1414.213562 | 90 |
| gerente | 2.0 | 30.0 | 7.071068 | 25.0 | 27.50 | 30.0 | 32.50 | 35.0 | 2.0 | 8500.0 | 707.106781 | 80 |

```
1 import pandas as pd
2 nome = ['Douglas', 'Daniela', 'Pedro', 'Maria', 'Eduardo', 'Ester']
3 idade = [45, 76, 64, 42, 37]
4 altura = [1.85, 1.23, 1.75, 1.67, 1.82, 1.73]
5 peso = [70, 22, 87, 64, 96, 68]
6 sexo = ['M', 'F', 'M', 'F', 'M', 'F']
7 df = pd.DataFrame(zip(nome, idade, altura, peso, sexo),
8                   columns=['nome', 'idade', 'altura', 'peso', 'sexo'])
```

▼ Atualização 03/05/2024

```
1 import pandas as pd
```

```

1 '''
2 Atualização 03/05/2024
3 '''
4 '''
5 Utilizando o conceito de agrupamento faça uma distribuição de frequência das
6 faixas etárias
7 '''
8 faixas = ['00 - 10','11 - 20','21 - 30','31 - 40']
9 df['faixa'] = pd.cut(x=df['Total de anos trabalhados'],bins=[-1,10,20,30,40],labels=faixas)# quando a variável for quanti inteira os bins são excluídos
10 idades = df.groupby('faixa')['Total de anos trabalhados']
11 idades.count()

faixa
00 - 10    923
11 - 20    340
1 df = pd.read_csv('https://docs.google.com/spreadsheets/d/e/2PACX-1vRvUmZWGvFp6i7t7pul-Ty44jNEz9dykaC99PDvWDu3LL5Jna4vbM3RTUbU7bHhsgwe0t1ddAHto1X0/pu
Name: Total de anos trabalhados, dtype: int64
1 df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Idade            1470 non-null   int64  
 1   Demissão          1470 non-null   object  
 2   Viagem de negócios 1470 non-null   object  
 3   Diárias recebidas 1470 non-null   int64  
 4   Departamento      1470 non-null   object  
 5   Distância de Casa 1470 non-null   int64  
 6   Escolaridade       1470 non-null   int64  
 7   Área de Formação   1470 non-null   object  
 8   Número de contratos de trabalho na empresa 1470 non-null   int64  
 9   ID Funcionário     1470 non-null   int64  
 10  Nível de Satisfação com o ambiente de trabalho 1470 non-null   int64  
 11  Gênero             1470 non-null   object  
 12  Salário por hora   1470 non-null   int64  
 13  Nível de envolvimento com o trabalho 1470 non-null   int64  
 14  Nível hierárquico   1470 non-null   int64  
 15  Cargo              1470 non-null   object  
 16  Satisfação com o trabalho 1470 non-null   int64  
 17  Estado civil        1470 non-null   object  
 18  Renda mensal        1470 non-null   int64  
 19  Salário Mensal      1470 non-null   int64  
 20  qtde de empresas que já trabalhou 1470 non-null   int64  
 21  Maior de idade       1470 non-null   object  
 22  Faz hora extra       1470 non-null   object  
 23  percentual de aumento de salário 1470 non-null   int64  
 24  score de performance 1470 non-null   int64  
 25  Satisfação nas relações Não trabalho 1470 non-null   int64  
 26  Jornada padrão de trabalho 1470 non-null   int64  
 27  opção de remuneração variável em ações 1470 non-null   int64  
 28  Total de anos trabalhados 1470 non-null   int64  
 29  Qtde de treinamentos realizados no último ano 1470 non-null   int64  
 30  Equilíbrio vida trabalho 1470 non-null   int64  
 31  Total de anos trabalhados na empresa 1470 non-null   int64  
 32  Anos trabalhados na função atual 1470 non-null   int64  
 33  Anos desde a última promoção 1470 non-null   int64  
 34  Anos trabalhando com o atual gestor 1470 non-null   int64  
dtypes: int64(26), object(9)
memory usage: 402.1+ KB

```

1 Comece a programar ou gere código com IA.

..