



RPG0033 - TRATANDO A IMENSIDÃO DOS DADOS

Paulo Wuéliton Horacio Fernandes (202402025919)

507 Polo Tucuruvi – São Paulo/SP

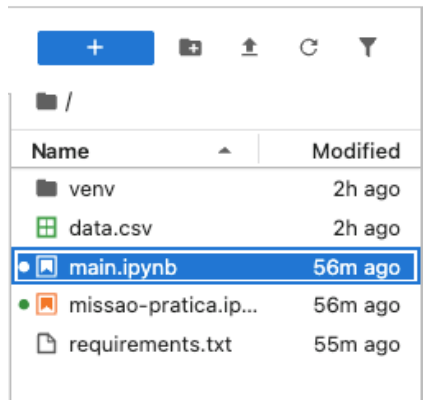
Nível 3: Tratando a Imensidão Dos Dados

Objetivo da Prática

- Descrever como ler um arquivo CSV usando a biblioteca Pandas (Python);
- Descrever como criar um subconjunto de dados a partir de um conjunto existente usando a biblioteca Pandas (Python);
- Descrever como configurar o número máximo de linhas a serem exibidas na visualização de um conjunto de dados usando a biblioteca Pandas (Python);
- Descrever como exibir as primeiras e últimas “N” linhas de um conjunto de dados usando a biblioteca Pandas (Python); Descrever como exibir informações gerais sobre as colunas, linhas e dados de um conjunto de dados usando a biblioteca Pandas (Python);

Microatividade 1: Descrever como ler um arquivo CSV usando a biblioteca Pandas (Python)

1. Foi criado e salvo um arquivo chamado data.csv contendo os dados do exercício



2. Foi instalada a biblioteca pandas no projeto e o JupyterLab para execução dos exercícios, as dependências foram adicionadas no projeto usando o comando freeze do pip e salvas no arquivo requirements.txt para que seja possível executar o projeto em outros computadores. Para execução do exercício foi utilizado o comando read_csv para acessar o arquivo data.csv e os dados lidos foram exibidos utilizando o comando print com o to_string do pandas para exibir em modo string.

Código:

```
import pandas as pd

data = ""

file = pd.read_csv('./data.csv', sep=';', engine='python', encoding='utf-8')

data = file

print(data.to_string())
```

Saída:

```
[19]: import pandas as pd

data = ''
file = pd.read_csv('./data.csv', sep=';', engine='python', encoding='utf-8')
data = file

print(data.to_string())
```

	ID	Duration	Date	Pulse	Maxpulse	Calories
0	0	60	'2020/12/01'	110	130	4091
1	1	60	'2020/12/02'	117	145	4790
2	2	60	'2020/12/03'	103	135	3400
3	3	45	'2020/12/04'	109	175	2824
4	4	45	'2020/12/05'	117	148	4060
5	5	60	'2020/12/06'	102	127	3000
6	6	60	'2020/12/07'	110	136	3740
7	7	450	'2020/12/08'	104	134	2533
8	8	30	'2020/12/09'	109	133	1951
9	9	60	'2020/12/10'	98	124	2690
10	10	60	'2020/12/11'	103	147	3293
11	11	60	'2020/12/12'	100	120	2507
12	12	60	'2020/12/12'	100	120	2507
13	13	60	'2020/12/13'	106	128	3453
14	14	60	'2020/12/14'	104	132	3793
15	15	60	'2020/12/15'	98	123	2750
16	16	60	'2020/12/16'	98	120	2152
17	17	60	'2020/12/17'	100	120	3000
18	18	45	'2020/12/18'	90	112	NaN
19	19	60	'2020/12/19'	103	123	3230
20	20	45	'2020/12/20'	97	125	2430 2
21	1	60	'2020/12/21'	108	131	3642
22	22	45	NaN	100	119	2820
23	23	60	'2020/12/23'	130	101	3000
24	24	45	'2020/12/24'	105	132	2460
25	25	60	'2020/12/25'	102	126	3345
26	26	60	20201226	100	120	2500
27	27	60	'2020/12/27'	92	118	2410
28	28	60	'2020/12/28'	103	132	NaN
29	29	60	'2020/12/29'	100	132	2800
30	30	60	'2020/12/30'	102	129	3803
31	31	60	'2020/12/31'	92	115	2430

Microatividade 2: Descrever como criar um subconjunto de dados a partir de um conjunto existente usando a biblioteca Pandas (Python)

1. Criada uma nova variável columns
2. Adicionados os dados apenas das 3 primeiras colunas ('ID', 'Duration', 'Date') e exibidos os dados utilizando o comando print:

Código:

```
columns = file[['ID', 'Duration', 'Date']]
```

```
print(columns.to_string())
```

Saída:

```
[5]: columns = file[['ID', 'Duration', 'Date']]
      print(columns.to_string())
```

	ID	Duration	Date
0	0	60	'2020/12/01'
1	1	60	'2020/12/02'
2	2	60	'2020/12/03'
3	3	45	'2020/12/04'
4	4	45	'2020/12/05'
5	5	60	'2020/12/06'
6	6	60	'2020/12/07'
7	7	450	'2020/12/08'
8	8	30	'2020/12/09'
9	9	60	'2020/12/10'
10	10	60	'2020/12/11'
11	11	60	'2020/12/12'
12	12	60	'2020/12/12'
13	13	60	'2020/12/13'
14	14	60	'2020/12/14'
15	15	60	'2020/12/15'
16	16	60	'2020/12/16'
17	17	60	'2020/12/17'
18	18	45	'2020/12/18'
19	19	60	'2020/12/19'
20	20	45	'2020/12/20'
21	1	60	'2020/12/21'
22	22	45	NaN
23	23	60	'2020/12/23'
24	24	45	'2020/12/24'
25	25	60	'2020/12/25'
26	26	60	20201226
27	27	60	'2020/12/27'
28	28	60	'2020/12/28'
29	29	60	'2020/12/29'
30	30	60	'2020/12/30'
31	31	60	'2020/12/31'

Microatividade 3: Descrever como configurar o número máximo de linhas a serem exibidas na visualização de um conjunto de dados usando a biblioteca Pandas (Python)

1. Utilizar a configuração `max_rows` para definir o limite de 9999:

Código:

```
pd.set_option('display.max_rows', 9999)
```

```
print(data.to_string())
```

Saída:

```
[21]: pd.set_option('display.max_rows', 9999)
      print(data.to_string())
```

	ID	Duration	Date	Pulse	Maxpulse	Calories
0	0	60	'2020/12/01'	110	130	4091
1	1	60	'2020/12/02'	117	145	4790
2	2	60	'2020/12/03'	103	135	3400
3	3	45	'2020/12/04'	109	175	2824
4	4	45	'2020/12/05'	117	148	4060
5	5	60	'2020/12/06'	102	127	3000
6	6	60	'2020/12/07'	110	136	3740
7	7	450	'2020/12/08'	104	134	2533
8	8	30	'2020/12/09'	109	133	1951
9	9	60	'2020/12/10'	98	124	2690
10	10	60	'2020/12/11'	103	147	3293
11	11	60	'2020/12/12'	100	120	2507
12	12	60	'2020/12/12'	100	120	2507
13	13	60	'2020/12/13'	106	128	3453
14	14	60	'2020/12/14'	104	132	3793
15	15	60	'2020/12/15'	98	123	2750
16	16	60	'2020/12/16'	98	120	2152
17	17	60	'2020/12/17'	100	120	3000
18	18	45	'2020/12/18'	90	112	NaN
19	19	60	'2020/12/19'	103	123	3230
20	20	45	'2020/12/20'	97	125	2430 2
21	1	60	'2020/12/21'	108	131	3642
22	22	45	NaN	100	119	2820
23	23	60	'2020/12/23'	130	101	3000
24	24	45	'2020/12/24'	105	132	2460
25	25	60	'2020/12/25'	102	126	3345
26	26	60	20201226	100	120	2500
27	27	60	'2020/12/27'	92	118	2410
28	28	60	'2020/12/28'	103	132	NaN
29	29	60	'2020/12/29'	100	132	2800
30	30	60	'2020/12/30'	102	129	3803
31	31	60	'2020/12/31'	92	115	2430

Microatividade 4: Descrever como exibir as primeiras e últimas “N” linhas de um conjunto de dados usando a biblioteca Pandas (Python)

1. Imprimir as 10 primeiras linhas do conjunto de dados utilizando o método head do pandas:

Código:

```
print("Primeiras 10 linhas")
print(file.head(10))
```

Saída:

```
[43]: print("Primeiras 10 linhas")
      print(file.head(10))
```

Primeiras 10 linhas

	ID	Duration	Date	Pulse	Maxpulse	Calories
0	0	60	'2020/12/01'	110	130	4091
1	1	60	'2020/12/02'	117	145	4790
2	2	60	'2020/12/03'	103	135	3400
3	3	45	'2020/12/04'	109	175	2824
4	4	45	'2020/12/05'	117	148	4060
5	5	60	'2020/12/06'	102	127	3000
6	6	60	'2020/12/07'	110	136	3740
7	7	450	'2020/12/08'	104	134	2533
8	8	30	'2020/12/09'	109	133	1951
9	9	60	'2020/12/10'	98	124	2690

2. Imprimir as últimas 10 linhas do conjunto de dados utilizando o método tail do pandas:

Código:

```
print("Últimas 10 linhas")  
  
print(file.tail(10))
```

Saída:

```
[44]: print("Últimas 10 linhas")  
      print(file.tail(10))  
  
Últimas 10 linhas  
      ID  Duration      Date  Pulse  Maxpulse  Calories  
22  22      45      NaN    100      119      2820  
23  23      60  '2020/12/23'  130      101      3000  
24  24      45  '2020/12/24'  105      132      2460  
25  25      60  '2020/12/25'  102      126      3345  
26  26      60  '2020/12/26'  100      120      2500  
27  27      60  '2020/12/27'   92      118      2410  
28  28      60  '2020/12/28'  103      132       NaN  
29  29      60  '2020/12/29'  100      132      2800  
30  30      60  '2020/12/30'  102      129      3803  
31  31      60  '2020/12/31'   92      115      2430
```

Microatividade 5: Descrever como exibir informações gerais sobre as colunas, linhas e dados de um conjunto de dados usando a biblioteca Pandas (Python)

1. Imprimir as informações gerais do conjunto de dados:

Código:

```
print("Informações gerais")  
  
file.info()
```

Saída:

```
[45]: print("Informações gerais")  
      file.info()  
  
Informações gerais  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 32 entries, 0 to 31  
Data columns (total 6 columns):  
#   Column      Non-Null Count  Dtype  
---  ---  
0   ID          32 non-null    int64  
1   Duration    32 non-null    int64  
2   Date        31 non-null    object  
3   Pulse       32 non-null    int64  
4   Maxpulse    32 non-null    int64  
5   Calories    30 non-null    object  
dtypes: int64(4), object(2)  
memory usage: 1.6+ KB
```

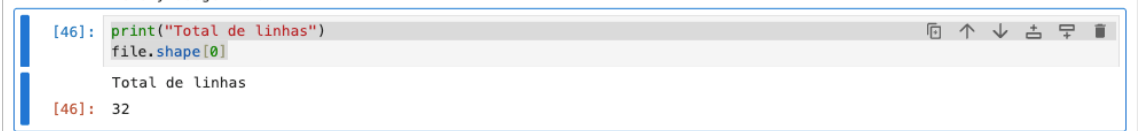
2. Imprimir o total de linhas do conjunto de dados:

Código:

```
print("Total de linhas")

file.shape[0]
```

Saída:

A screenshot of a Jupyter Notebook cell showing the execution of code to print the total number of lines. The code is: `[46]: print("Total de linhas")` followed by `file.shape[0]`. The output is: `Total de linhas` followed by `[46]: 32`.

```
[46]: print("Total de linhas")
      file.shape[0]

Total de linhas
[46]: 32
```

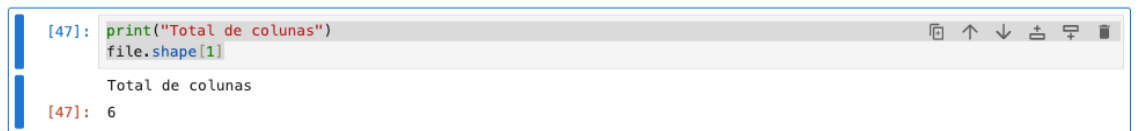
3. Imprimir o total de Colunas do conjunto de dados:

Código:

```
print("Total de colunas")

file.shape[1]
```

Saída:

A screenshot of a Jupyter Notebook cell showing the execution of code to print the total number of columns. The code is: `[47]: print("Total de colunas")` followed by `file.shape[1]`. The output is: `Total de colunas` followed by `[47]: 6`.

```
[47]: print("Total de colunas")
      file.shape[1]

Total de colunas
[47]: 6
```

4. Imprimir a quantidade de dados nulos:

Código:

```
print("Quantidade de dados nulos")

file.isnull().sum()
```

Saída:

A screenshot of a Jupyter Notebook cell showing the execution of code to print the number of null values. The code is: `[48]: print("Quantidade de dados nulos")` followed by `file.isnull().sum()`. The output is: `Quantidade de dados nulos` followed by a table of null counts for each column. The table has 7 rows: ID (0), Duration (0), Date (1), Pulse (0), Maxpulse (0), Calories (2), and dtype: int64.

```
[48]: print("Quantidade de dados nulos")
      file.isnull().sum()

Quantidade de dados nulos
[48]: ID          0
      Duration    0
      Date        1
      Pulse       0
      Maxpulse    0
      Calories    2
      dtype: int64
```

5. Imprimir a quantidade de memória utilizada:

Código:

```
print("Memória utilizada")

file.memory_usage(deep=True)
```

Saída:

```
dtype: int64

[7]: print("Memória utilizada")
file.memory_usage(deep=True)

Memória utilizada
[7]: Index      132
     ID          256
     Duration    256
     Date        1919
     Pulse        256
     Maxpulse     256
     Calories    1656
     dtype: int64
```

Missão Prática | Tratando a imensidão dos dados ☐

1. Ler o conteúdo do CSV e exibir as informações gerais sobre o conjunto de dados:

Código:

```
import pandas as pd

file = pd.read_csv('./data.csv', sep=';', encoding='utf-8')
print(file.info())
```

Saída:

```
[11]: import pandas as pd
file = pd.read_csv('./data.csv', sep=';', encoding='utf-8')

[12]: print(file.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
---  --
 0   ID          32 non-null    int64  
 1   Duration    32 non-null    int64  
 2   Date        31 non-null    object  
 3   Pulse       32 non-null    int64  
 4   Maxpulse    32 non-null    int64  
 5   Calories    30 non-null    object  
dtypes: int64(4), object(2)
memory usage: 1.6+ KB
None
```

2. Ler as 10 primeiras linhas e as últimas 10 linhas do conjunto de dados:

Código:

```
print(file.head(10))

print(file.tail(10))
```


Saída:

```
[13]: print(file.head(10))
      print(file.tail(10))
```

	ID	Duration	Date	Pulse	Maxpulse	Calories
0	0	60	'2020/12/01'	110	130	4091
1	1	60	'2020/12/02'	117	145	4790
2	2	60	'2020/12/03'	103	135	3400
3	3	45	'2020/12/04'	109	175	2824
4	4	45	'2020/12/05'	117	148	4060
5	5	60	'2020/12/06'	102	127	3000
6	6	60	'2020/12/07'	110	136	3740
7	7	450	'2020/12/08'	104	134	2533
8	8	30	'2020/12/09'	109	133	1951
9	9	60	'2020/12/10'	98	124	2690
	ID	Duration	Date	Pulse	Maxpulse	Calories
22	22	45	NaN	100	119	2820
23	23	60	'2020/12/23'	130	101	3000
24	24	45	'2020/12/24'	105	132	2460
25	25	60	'2020/12/25'	102	126	3345
26	26	60	20201226	100	120	2500
27	27	60	'2020/12/27'	92	118	2410
28	28	60	'2020/12/28'	103	132	NaN
29	29	60	'2020/12/29'	100	132	2800
30	30	60	'2020/12/30'	102	129	3803
31	31	60	'2020/12/31'	92	115	2430

3. Criar uma cópia dos dados em uma nova variável e substituir todos os valores nulos da coluna 'Calories' por 0:

Código:

```
copy = file
copy['Calories'] = copy['Calories'].fillna(0)

print(copy)
```

Saída:

```
[14]: copy = file
```

```
[15]: copy['Calories'] = copy['Calories'].fillna(0)  
print(copy)
```

	ID	Duration	Date	Pulse	Maxpulse	Calories
0	0	60	'2020/12/01'	110	130	4091
1	1	60	'2020/12/02'	117	145	4790
2	2	60	'2020/12/03'	103	135	3400
3	3	45	'2020/12/04'	109	175	2824
4	4	45	'2020/12/05'	117	148	4060
5	5	60	'2020/12/06'	102	127	3000
6	6	60	'2020/12/07'	110	136	3740
7	7	450	'2020/12/08'	104	134	2533
8	8	30	'2020/12/09'	109	133	1951
9	9	60	'2020/12/10'	98	124	2690
10	10	60	'2020/12/11'	103	147	3293
11	11	60	'2020/12/12'	100	120	2507
12	12	60	'2020/12/12'	100	120	2507
13	13	60	'2020/12/13'	106	128	3453
14	14	60	'2020/12/14'	104	132	3793
15	15	60	'2020/12/15'	98	123	2750
16	16	60	'2020/12/16'	98	120	2152
17	17	60	'2020/12/17'	100	120	3000
18	18	45	'2020/12/18'	90	112	0
19	19	60	'2020/12/19'	103	123	3230
20	20	45	'2020/12/20'	97	125	2430
21	1	60	'2020/12/21'	108	131	3642
22	22	45	NaN	100	119	2820
23	23	60	'2020/12/23'	130	101	3000
24	24	45	'2020/12/24'	105	132	2460
25	25	60	'2020/12/25'	102	126	3345
26	26	60	20201226	100	120	2500
27	27	60	'2020/12/27'	92	118	2410
28	28	60	'2020/12/28'	103	132	0
29	29	60	'2020/12/29'	100	132	2800
30	30	60	'2020/12/30'	102	129	3803
31	31	60	'2020/12/31'	92	115	2430

4. Substituir os valores nulos da coluna 'Date' por '1900/01/01', imprimir o resultado e transformar os dados da coluna 'Date' em datetime utilizando o método 'to_datetime':

Código:

```
copy['Date'] = copy['Date'].fillna("1900/01/01")  
  
print(copy)  
  
copy['Date'] = pd.to_datetime(copy['Date'])
```

Saída:

```
[16]: copy['Date'] = copy['Date'].fillna("1900/01/01")
print(copy)
copy['Date'] = pd.to_datetime(copy['Date'])
```

	ID	Duration	Date	Pulse	Maxpulse	Calories
0	0	60	'2020/12/01'	110	130	4091
1	1	60	'2020/12/02'	117	145	4790
2	2	60	'2020/12/03'	103	135	3400
3	3	45	'2020/12/04'	109	175	2824
4	4	45	'2020/12/05'	117	148	4060
5	5	60	'2020/12/06'	102	127	3000
6	6	60	'2020/12/07'	110	136	3740
7	7	450	'2020/12/08'	104	134	2533
8	8	30	'2020/12/09'	109	133	1951
9	9	60	'2020/12/10'	98	124	2690
10	10	60	'2020/12/11'	103	147	3293
11	11	60	'2020/12/12'	100	120	2507
12	12	60	'2020/12/12'	100	120	2507
13	13	60	'2020/12/13'	106	128	3453
14	14	60	'2020/12/14'	104	132	3793
15	15	60	'2020/12/15'	98	123	2750
16	16	60	'2020/12/16'	98	120	2152
17	17	60	'2020/12/17'	100	120	3000
18	18	45	'2020/12/18'	90	112	0
19	19	60	'2020/12/19'	103	123	3230
20	20	45	'2020/12/20'	97	125	2430 2
21	1	60	'2020/12/21'	108	131	3642
22	22	45	'1900/01/01'	100	119	2820
23	23	60	'2020/12/23'	130	101	3000
24	24	45	'2020/12/24'	105	132	2460
25	25	60	'2020/12/25'	102	126	3345
26	26	60	'2020/12/26'	100	120	2500
27	27	60	'2020/12/27'	92	118	2410
28	28	60	'2020/12/28'	103	132	0
29	29	60	'2020/12/29'	100	132	2800
30	30	60	'2020/12/30'	102	129	3803
31	31	60	'2020/12/31'	92	115	2430

```
File ~/Documents/python/missao-pratica-nivel3-mundo5/venv/lib/python3.13/site-packages/pandas/core/tools/datetim
es.py:1067, in to_datetime(arg, errors, dayfirst, yearfirst, utc, format, exact, unit, infer_datetime_format, or
igin, cache)
1065     result = arg.map(cache_array)
1066 else:
--> 1067     values = convert_listlike(arg._values, format)
1068     result = arg._constructor(values, index=arg.index, name=arg.name)
1069 elif isinstance(arg, (ABCDDataFrame, abc.MutableMapping)):
```

```
File ~/Documents/python/missao-pratica-nivel3-mundo5/venv/lib/python3.13/site-packages/pandas/core/tools/datetim
es.py:433, in _convert_listlike_datetimes(arg, format, name, utc, unit, errors, dayfirst, yearfirst, exact)
431 # 'format' could be inferred, or user didn't ask for mixed-format parsing.
432 if format is not None and format != "mixed":
--> 433     return _array_strptime_with_fallback(arg, name, utc, format, exact, errors)
435 result, tz_parsed = objects_to_datetime64(
436     arg,
437     dayfirst=dayfirst,
438     (...) 441     allow_object=True,
442 )
444 if tz_parsed is not None:
445     # We can take a shortcut since the datetime64 numpy array
446     # is in UTC
```

```
File ~/Documents/python/missao-pratica-nivel3-mundo5/venv/lib/python3.13/site-packages/pandas/core/tools/datetim
es.py:467, in _array_strptime_with_fallback(arg, name, utc, fmt, exact, errors)
466 def _array_strptime_with_fallback(
467     arg,
468     name,
469     (...) 462     errors: str,
470 ) -> Index:
471     """
472     Call array_strptime, with fallback behavior depending on 'errors'.
```

```
473     """
474     result, tz_out = array_strptime(arg, fmt, exact=exact, errors=errors, utc=utc)
475     if tz_out is not None:
476         unit = np.datetime_data(result.dtype)[0]
477     return result, tz_out, unit
478 File strptime.pyx:501, in pandas._libs.tslibs.strptime.array_strptime()
479 File strptime.pyx:451, in pandas._libs.tslibs.strptime.array_strptime()
480 File strptime.pyx:583, in pandas._libs.tslibs.strptime._parse_with_format()
```

```
ValueError: time data "1900/01/01" doesn't match format "%Y/%m/%d", at position 22. You might want to try:
- passing 'format' if your strings have a consistent format;
- passing 'format='ISO8601'' if your strings are all ISO8601 but not necessarily in exactly the same format;
- passing 'format='mixed'', and the format will be inferred for each element individually. You might want to
use 'dayfirst' alongside this.
```

5. Substituir o valor '1900/01/01' por 'NaN' da coluna 'Date'

Código:

```
copy['Date'] = copy['Date'].replace('1900/01/01', 'NaN')

print(copy)
```

Saída:

```
[7]: copy['Date'] = copy['Date'].replace('1900/01/01', 'NaN')
print(copy)
```

	ID	Duration	Date	Pulse	Maxpulse	Calories
0	0	60	'2020/12/01'	110	130	4091
1	1	60	'2020/12/02'	117	145	4790
2	2	60	'2020/12/03'	103	135	3400
3	3	45	'2020/12/04'	109	175	2824
4	4	45	'2020/12/05'	117	148	4060
5	5	60	'2020/12/06'	102	127	3000
6	6	60	'2020/12/07'	110	136	3740
7	7	450	'2020/12/08'	104	134	2533
8	8	30	'2020/12/09'	109	133	1951
9	9	60	'2020/12/10'	98	124	2690
10	10	60	'2020/12/11'	103	147	3293
11	11	60	'2020/12/12'	100	120	2507
12	12	60	'2020/12/12'	100	120	2507
13	13	60	'2020/12/13'	106	128	3453
14	14	60	'2020/12/14'	104	132	3793
15	15	60	'2020/12/15'	98	123	2750
16	16	60	'2020/12/16'	98	120	2152
17	17	60	'2020/12/17'	100	120	3000
18	18	45	'2020/12/18'	90	112	0
19	19	60	'2020/12/19'	103	123	3230
20	20	45	'2020/12/20'	97	125	2430 2
21	1	60	'2020/12/21'	108	131	3642
22	22	45	NaN	100	119	2820
23	23	60	'2020/12/23'	130	101	3000
24	24	45	'2020/12/24'	105	132	2460
25	25	60	'2020/12/25'	102	126	3345
26	26	60	'2020/12/26'	100	120	2500
27	27	60	'2020/12/27'	92	118	2410
28	28	60	'2020/12/28'	103	132	0
29	29	60	'2020/12/29'	100	132	2800
30	30	60	'2020/12/30'	102	129	3803
31	31	60	'2020/12/31'	92	115	2430

- Transformar os dados da coluna 'Date' para datetime usando o método 'to_datetime':

Código:

```
copy['Date'] = pd.to_datetime(copy['Date'])
```

Saída:

```

File ~/Documents/python/missao-pratica-nivel3-mundo5/venv/lib/python3.13/site-packages/pandas/core/tools/datetim
es.py:1067, in to_datetime(arg, errors, dayfirst, yearfirst, utc, format, exact, unit, infer_datetime_format, or
igin, cache)
    1065     result = arg.map(cache_array)
    1066 else:
-> 1067     values = convert_listlike(arg._values, format)
    1068     result = arg._constructor(values, index=arg.index, name=arg.name)
    1069 elif isinstance(arg, (ABCDDataFrame, abc.MutableMapping)):

File ~/Documents/python/missao-pratica-nivel3-mundo5/venv/lib/python3.13/site-packages/pandas/core/tools/datetim
es.py:433, in _convert_listlike_datetimes(arg, format, name, utc, unit, errors, dayfirst, yearfirst, exact)
    431 # 'format' could be inferred, or user didn't ask for mixed-format parsing.
    432 if format is not None and format != "mixed":
-> 433     return _array_strptime_with_fallback(arg, name, utc, format, exact, errors)
    435 result, tz_parsed = objects_to_datetime64(
    436     arg,
    437     dayfirst=dayfirst,
    (...) 441     allow_object=True,
    442 )
    444 if tz_parsed is not None:
    445     # We can take a shortcut since the datetime64 numpy array
    446     # is in UTC

File ~/Documents/python/missao-pratica-nivel3-mundo5/venv/lib/python3.13/site-packages/pandas/core/tools/datetim
es.py:467, in _array_strptime_with_fallback(arg, name, utc, fmt, exact, errors)
    466 def _array_strptime_with_fallback(
    467     arg,
    468     name,
    (...) 469     errors: str,
    470 ) -> Index:
    471     """
    472     Call array_strptime, with fallback behavior depending on 'errors'.
    473     """
-> 474     result, tz_out = array_strptime(arg, fmt, exact=exact, errors=errors, utc=utc)
    475     if tz_out is not None:
    476         unit = np.datetime_data(result.dtype)[0]

File strptime.pyx:501, in pandas._libs.tslibs.strptime.array_strptime()

File strptime.pyx:451, in pandas._libs.tslibs.strptime.array_strptime()

File strptime.pyx:583, in pandas._libs.tslibs.strptime._parse_with_format()

ValueError: time data "20201226" doesn't match format "%Y/%m/%d", at position 26. You might want to try:
- passing 'format' if your strings have a consistent format;
- passing 'format='ISO8601'' if your strings are all ISO8601 but not necessarily in exactly the same format;
- passing 'format='mixed'', and the format will be inferred for each element individually. You might want to
use 'dayfirst' alongside this.

```

- Transformar o valor '20201226' em uma data válida usando o 'replace' e o 'to_datetime' e converter todos os dados da coluna 'Date' para datas:

Código:

```

copy['Date'] = copy['Date'].replace('20201226', pd.to_datetime('2020/12/26',
format='%Y/%m/%d'))

copy['Date'] = pd.to_datetime(copy['Date'])

print(copy)

```

Saída:

```
[9]: copy['Date'] = copy['Date'].replace('20201226', pd.to_datetime('2020/12/26', format='%Y/%m/%d'))
copy['Date'] = pd.to_datetime(copy['Date'])
print(copy)
```

	ID	Duration	Date	Pulse	Maxpulse	Calories
0	0	60	2020-12-01	110	130	4091
1	1	60	2020-12-02	117	145	4790
2	2	60	2020-12-03	103	135	3400
3	3	45	2020-12-04	109	175	2824
4	4	45	2020-12-05	117	148	4060
5	5	60	2020-12-06	102	127	3000
6	6	60	2020-12-07	110	136	3740
7	7	450	2020-12-08	104	134	2533
8	8	30	2020-12-09	109	133	1951
9	9	60	2020-12-10	98	124	2690
10	10	60	2020-12-11	103	147	3293
11	11	60	2020-12-12	100	120	2507
12	12	60	2020-12-12	100	120	2507
13	13	60	2020-12-13	106	128	3453
14	14	60	2020-12-14	104	132	3793
15	15	60	2020-12-15	98	123	2750
16	16	60	2020-12-16	98	120	2152
17	17	60	2020-12-17	100	120	3000
18	18	45	2020-12-18	90	112	0
19	19	60	2020-12-19	103	123	3230
20	20	45	2020-12-20	97	125	2430 2
21	1	60	2020-12-21	108	131	3642
22	22	45	NaT	100	119	2820
23	23	60	2020-12-23	130	101	3000
24	24	45	2020-12-24	105	132	2460
25	25	60	2020-12-25	102	126	3345
26	26	60	2020-12-26	100	120	2500
27	27	60	2020-12-27	92	118	2410
28	28	60	2020-12-28	103	132	0
29	29	60	2020-12-29	100	132	2800
30	30	60	2020-12-30	102	129	3803
31	31	60	2020-12-31	92	115	2430

8. Remover os registros nulos:

Código:

```
copy = copy.dropna(subset=['Date'])
print(copy)
```

Saída:

```
[10]: copy = copy.dropna(subset=['Date'])
```

```
[11]: print(copy)
```

	ID	Duration	Date	Pulse	Maxpulse	Calories
0	0	60	2020-12-01	110	130	4091
1	1	60	2020-12-02	117	145	4790
2	2	60	2020-12-03	103	135	3400
3	3	45	2020-12-04	109	175	2824
4	4	45	2020-12-05	117	148	4060
5	5	60	2020-12-06	102	127	3000
6	6	60	2020-12-07	110	136	3740
7	7	450	2020-12-08	104	134	2533
8	8	30	2020-12-09	109	133	1951
9	9	60	2020-12-10	98	124	2690
10	10	60	2020-12-11	103	147	3293
11	11	60	2020-12-12	100	120	2507
12	12	60	2020-12-12	100	120	2507
13	13	60	2020-12-13	106	128	3453
14	14	60	2020-12-14	104	132	3793
15	15	60	2020-12-15	98	123	2750
16	16	60	2020-12-16	98	120	2152
17	17	60	2020-12-17	100	120	3000
18	18	45	2020-12-18	90	112	0
19	19	60	2020-12-19	103	123	3230
20	20	45	2020-12-20	97	125	2430 2
21	1	60	2020-12-21	108	131	3642
23	23	60	2020-12-23	130	101	3000
24	24	45	2020-12-24	105	132	2460
25	25	60	2020-12-25	102	126	3345
26	26	60	2020-12-26	100	120	2500
27	27	60	2020-12-27	92	118	2410
28	28	60	2020-12-28	103	132	0
29	29	60	2020-12-29	100	132	2800
30	30	60	2020-12-30	102	129	3803
31	31	60	2020-12-31	92	115	2430