



## **POLO CHAPADA - MANAUS - AM/UNIVERSIDADE ESTÁCIO DE SÁ**

### **Missão Prática | Mundo 5 | Nível 5**

Curso: Desenvolvimento Full Stack

Disciplina Nível5: DGT2823 - Tecnologias para desenv. de soluções de big data.

Aluno: Gilberto Ferreira da Silva Junior Matrícula: 202307014923

#### **1º Título da Prática: Tecnologias para desenv. de soluções de big data.**

#### **2º Objetivo da Prática:**

1. Descrever como ler um arquivo CSV usando a biblioteca Pandas (Python);
2. Descrever como criar um subconjunto de dados a partir de um conjunto existente usando a biblioteca Pandas (Python);
3. Descrever como configurar o número máximo de linhas a serem exibidas na visualização de um conjunto de dados usando a biblioteca Pandas (Python);
4. 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);

#### **Códigos solicitados:**

<https://github.com/GilbertoFSJunior/Trabalho-Pr-tico-DGT2823-Tecnologias-para-desenv.-de-solu-es-de-big-data.git>

# Sumário

Curso: Desenvolvimento Full Stack .....1 Aluno: Gilberto Ferreira da Silva Junior Matrícula: 202307014923 .....1 1º INTRODUÇÃO .....3

2. OBJETIVOS .....4

|  |    |
|--|----|
| 2.1 Objetivos Gerais .....                                   | 4  |
| 2.2 Objetivos Específicos.....                               | 4  |
| 3. METODOLOGIA.....  | 4  |
| 3.1 Ambiente de Desenvolvimento.....                         | 4  |
| 3.2 DataFrame Utilizado .....                                | 4  |
| 3.3 Problemas Identificados no Dataset.....                  | 4  |
| 4. MICROATIVIDADES .....                                     | 5  |
| 4.1 Microatividade 1: Leitura de CSV .....                   | 5  |
| 4.2 Microatividade 2: Subconjunto de Dados .....             | 8  |
| 4.3 Microatividade 3: Configuração de Visualização .....     | 12 |
| 4.4 Microatividade 4: Primeiras e Últimas Linhas .....       | 13 |
| 4.5 Microatividade 5: Informações Gerais .....               | 15 |
| 5. TRABALHO PRÁTICO FINAL - DGT2823.....                     | 18 |
| 5.1 Passo 1: Preparação do Dataset .....                     | 18 |
| 5.2 Passo 2-4: Leitura do CSV .....                          | 18 |
| 5.3 Passo 5: Verificação dos Dados.....                      | 19 |
| 5.4 Passo 6: Cópia do Dataset.....                           | 19 |
| 5.5 Passo 7: Tratamento de Valores Nulos em 'Calories' ..... | 19 |
| 5.6 Passo 8: Tratamento Inicial da Coluna 'Date'.....        | 20 |
| 5.7 Passo 9: Correção do Primeiro Erro.....                  | 20 |
| 5.8 Passo 10: Correção de Formato Inconsistente .....        | 21 |
| 5.9 Passo 11: Conversão Final .....                          | 21 |
| 5.10 Passo 12: Remoção de Registros Nulos.....               | 21 |
| 5.11 Passo 13: Verificação Final .....                       | 22 |

## 1º INTRODUÇÃO

Este relatório tem como finalidade apresentar as atividades desenvolvidas no âmbito da disciplina **DGT2823 – Tecnologias para Desenvolvimento de Soluções de Big Data**, com foco na aplicação prática dos conceitos de manipulação e limpeza de dados. As atividades foram conduzidas com o uso da biblioteca **Pandas**, amplamente utilizada para o processamento e análise de dados na linguagem de programação **Python**.

O principal objetivo deste trabalho é demonstrar, de forma prática, o conhecimento adquirido sobre técnicas de tratamento de dados, por meio da identificação, limpeza e padronização de valores inconsistentes e ausentes em um conjunto de dados realista. A atividade visa proporcionar uma experiência próxima à realidade enfrentada por profissionais da área de ciência de dados.

As tarefas foram desenvolvidas no ambiente local utilizando a IDE - Visual Studio Code e o terminal CLI da mesma, para exibição dos dados.

O conjunto de dados utilizado contém informações relacionadas à prática de **exercícios físicos**. As variáveis presentes incluem:

- **Duração da atividade (em minutos)**
- **Data da atividade**
- **Frequência cardíaca média (pulso)**
- **Frequência cardíaca máxima**
- **Quantidade de calorias queimadas**

Este dataset foi propositalmente elaborado com **inconsistências** e **valores nulos**, com o intuito de simular um cenário verossímil de dados brutos, como frequentemente ocorre em aplicações reais. Isso permite aplicar técnicas de limpeza, tratamento e preparação de dados antes de prosseguir com análises mais complexas.

## 2. OBJETIVOS

### 2.1 Objetivos Gerais

- Aplicar técnicas de manipulação de dados utilizando a biblioteca Pandas
- Realizar limpeza e tratamento de dados inconsistentes
- Demonstrar proficiência em análise exploratória de dados

### 2.2 Objetivos Específicos

- Descrever como ler arquivos CSV usando Pandas
- Criar subconjuntos de dados a partir de conjuntos existentes
- Configurar opções de visualização de dados
- Exibir informações estatísticas e estruturais de datasets
- Tratar valores nulos e inconsistentes
- Converter tipos de dados adequadamente
- Validar a qualidade dos dados após tratamento

## 3. METODOLOGIA

### 3.1 Ambiente de Desenvolvimento

- Visual Studio Code: IDE
- Linguagem: Python 3.13.3
- Biblioteca Principal: Pandas
- Formato de Dados: CSV (Comma-Separated Values)

### 3.2 DataFrame Utilizado

```
=== INFORMAÇÕES ADICIONAIS SOBRE OS DADOS ===
Formato do DataFrame: (32, 6)
Colunas: ['ID', 'Duration', 'Date', 'Pulse', 'Maxpulse', 'Calories']
Tipos de dados:
ID          int64
Duration    int64
Date        object
Pulse       int64
Maxpulse    int64
Calories    float64
dtype: object
```

### 3.3 Problemas Identificados no Dataset

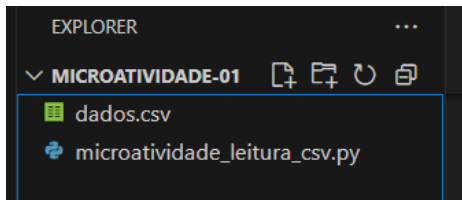
- Valores nulos (NaN) nas colunas Calories e Date
- Formato inconsistente de data (linha 26: "20201226")
- Necessidade de conversão de tipos de dados

## 4. MICROATIVIDADES

### 4.1 Microatividade 1: Leitura de CSV

Objetivo: Demonstrar a leitura de arquivos CSV utilizando a biblioteca Pandas.

Pasta contendo os dados.csv e o arquivo microatividade\_leitura\_csv.py.



#### Código Implementado Parte1:

```
1 # Microatividade 1: Descrever como ler um arquivo CSV usando a biblioteca Pandas
2 # Analista de Dados - Leitura de dados externos
3 |
4 # PROCEDIMENTO 2.1: Importe a biblioteca pandas
5 import pandas as pd
6
7 print("=== MICROATIVIDADE 1: LEITURA DE ARQUIVO CSV COM PANDAS ===\n")
8 print("✓ Passo 2.1: Biblioteca pandas importada com sucesso")
9
10 # PROCEDIMENTO 2.2: Cria uma variável
11 dados_lidos = None
12 print("✓ Passo 2.2: Variável 'dados_lidos' criada")
13
14 # PROCEDIMENTO 2.3 e 2.4: Leia o conteúdo do arquivo CSV e atribua à variável
15 print("\n--- LEITURA DO ARQUIVO CSV ---")
16
17 # DEMONSTRAÇÃO COM ARQUIVO REAL:
18 # Para ler um arquivo CSV real localizado na pasta raiz do projeto, use:
19 dados_lidos = pd.read_csv('dados.csv', sep=';', engine='python', encoding='utf-8')
20 print("✓ Passo 2.3 e 2.4: Arquivo CSV lido e dados atribuídos à variável 'dados_lidos'")
21
22 # Informações sobre os parâmetros utilizados
23 print("\n--- PARÂMETROS UTILIZADOS NA LEITURA ---")
24 print("• sep=';' → Define o separador de colunas como ponto e vírgula")
25 print("• engine='python' → Especifica o motor de análise Python")
26 print("• encoding='utf-8' → Define a codificação de caracteres")
```

#### Código Implementado Parte2:

```
28 # PROCEDIMENTO 2.5: Imprima/exiba em tela os dados da variável
29 print("\n--- PASSO 2.5: EXIBIÇÃO DOS DADOS LIDOS ---")
30
31 print("=== DADOS COMPLETOS DO ARQUIVO CSV ===")
32 print(dados_lidos)
33
34 print("\n=== INFORMAÇÕES ADICIONAIS SOBRE OS DADOS ===")
35 print(f"Formato do DataFrame: {dados_lidos.shape}")
36 print(f"Colunas: {list(dados_lidos.columns)}")
37 print(f"Tipos de dados:")
38 print(dados_lidos.dtypes)
39
40 print("\n=== PRIMEIRAS 5 LINHAS ===")
41 print(dados_lidos.head())
42
43 print("\n=== ÚLTIMAS 5 LINHAS ===")
44 print(dados_lidos.tail())
45
46 print("\n=== ESTATÍSTICAS BÁSICAS ===")
47 print(dados_lidos.describe())
48
49 print("\n=== MICROATIVIDADE 1 CONCLUÍDA COM SUCESSO! ===")
50 print("✓ Biblioteca pandas importada")
51 print("✓ Variável criada")
52 print("✓ Arquivo CSV lido com parâmetros especificados")
53 print("✓ Dados atribuídos à variável")
54 print("✓ Conteúdo exibido em tela")
55
```

## Resultado:

### Saída terminal Parte1:

```
=== MICROATIVIDADE 1: LEITURA DE ARQUIVO CSV COM PANDAS ===

✓ Passo 2.1: Biblioteca pandas importada com sucesso
✓ Passo 2.2: Variável 'dados_lidos' criada

--- LEITURA DO ARQUIVO CSV ---
✓ Passo 2.3 e 2.4: Arquivo CSV lido e dados atribuídos à variável 'dados_lidos'

--- PARÂMETROS UTILIZADOS NA LEITURA ---
• sep=';' → Define o separador de colunas como ponto e vírgula
• engine='python' → Especifica o motor de análise Python
• encoding='utf-8' → Define a codificação de caracteres
```

### Saída terminal Parte2: Exibição dos dados lidos:

```
--- PASSO 2.5: EXIBIÇÃO DOS DADOS LIDOS ---
=== DADOS COMPLETOS DO ARQUIVO CSV ===
```

|    | ID | Duration | Date         | Pulse | Maxpulse | Calories |
|----|----|----------|--------------|-------|----------|----------|
| 0  | 0  | 60       | '2020/12/01' | 110   | 130      | 4091.0   |
| 1  | 1  | 60       | '2020/12/02' | 117   | 145      | 4790.0   |
| 2  | 2  | 60       | '2020/12/03' | 103   | 135      | 3400.0   |
| 3  | 3  | 45       | '2020/12/04' | 109   | 175      | 2824.0   |
| 4  | 4  | 45       | '2020/12/05' | 117   | 148      | 4060.0   |
| 5  | 5  | 60       | '2020/12/06' | 102   | 127      | 3000.0   |
| 6  | 6  | 60       | '2020/12/07' | 110   | 136      | 3740.0   |
| 7  | 7  | 450      | '2020/12/08' | 104   | 134      | 2533.0   |
| 8  | 8  | 30       | '2020/12/09' | 109   | 133      | 1951.0   |
| 9  | 9  | 60       | '2020/12/10' | 98    | 124      | 2690.0   |
| 10 | 10 | 60       | '2020/12/11' | 103   | 147      | 3293.0   |
| 11 | 11 | 60       | '2020/12/12' | 100   | 120      | 2507.0   |
| 12 | 12 | 60       | '2020/12/12' | 100   | 120      | 2507.0   |
| 13 | 13 | 60       | '2020/12/13' | 106   | 128      | 3453.0   |
| 14 | 14 | 60       | '2020/12/14' | 104   | 132      | 3793.0   |
| 15 | 15 | 60       | '2020/12/15' | 98    | 123      | 2750.0   |
| 16 | 16 | 60       | '2020/12/16' | 98    | 120      | 2152.0   |
| 17 | 17 | 60       | '2020/12/17' | 100   | 120      | 3000.0   |
| 18 | 18 | 45       | '2020/12/18' | 90    | 112      | NaN      |
| 19 | 19 | 60       | '2020/12/19' | 103   | 123      | 3230.0   |
| 20 | 20 | 45       | '2020/12/20' | 97    | 125      | 2430.0   |
| 21 | 21 | 60       | '2020/12/21' | 108   | 131      | 3642.0   |
| 22 | 22 | 45       | NaN          | 100   | 119      | 2820.0   |
| 23 | 23 | 60       | '2020/12/23' | 130   | 101      | 3000.0   |
| 24 | 24 | 45       | '2020/12/24' | 105   | 132      | 2460.0   |
| 25 | 25 | 60       | '2020/12/25' | 102   | 126      | 3345.0   |
| 26 | 26 | 60       | 20201226     | 100   | 120      | 2500.0   |
| 27 | 27 | 60       | '2020/12/27' | 92    | 118      | 2410.0   |
| 28 | 28 | 60       | '2020/12/28' | 103   | 132      | NaN      |
| 29 | 29 | 60       | '2020/12/29' | 100   | 132      | 2800.0   |
| 30 | 30 | 60       | '2020/12/30' | 102   | 129      | 3803.0   |
| 31 | 31 | 60       | '2020/12/31' | 92    | 115      | 2430.0   |

### Saída terminal Parte 3:

Exemplo de como manusear a biblioteca pandas, realiza a leitura de dados de uma fonte externa e exibir seu conteúdo, criando subconjuntos de dados a partir de conjuntos existentes.

```
=== INFORMAÇÕES ADICIONAIS SOBRE OS DADOS ===
Formato do DataFrame: (32, 6)
Colunas: ['ID', 'Duration', 'Date', 'Pulse', 'Maxpulse', 'Calories']
Tipos de dados:
ID          int64
Duration    int64
Date        object
Pulse       int64
Maxpulse    int64
Calories    float64
dtype: object

=== PRIMEIRAS 5 LINHAS ===
   ID  Duration    Date  Pulse  Maxpulse  Calories
0   0         60 '2020/12/01'   110      130   4091.0
1   1         60 '2020/12/02'   117      145   4790.0
2   2         60 '2020/12/03'   103      135   3400.0
3   3         45 '2020/12/04'   109      175   2824.0
4   4         45 '2020/12/05'   117      148   4060.0

=== ÚLTIMAS 5 LINHAS ===
   ID  Duration    Date  Pulse  Maxpulse  Calories
27  27         60 '2020/12/27'    92      118   2410.0
28  28         60 '2020/12/28'   103      132     NaN
29  29         60 '2020/12/29'   100      132   2800.0
30  30         60 '2020/12/30'   102      129   3803.0
31  31         60 '2020/12/31'    92      115   2430.0

=== ESTATÍSTICAS BÁSICAS ===
      ID  Duration    Pulse  Maxpulse  Calories
count  32.000000  32.000000  32.000000  32.000000  30.000000
mean   15.500000  68.437500  103.500000  128.500000  3046.800000
std     9.380832  70.039591   7.832933  12.998759   660.037794
min     0.000000  30.000000  90.000000  101.000000  1951.000000
25%     7.750000  60.000000  100.000000  120.000000  2507.000000
50%    15.500000  60.000000  102.500000  127.500000  2912.000000
75%    23.250000  60.000000  106.500000  132.250000  3439.750000
max    31.000000  450.000000  130.000000  175.000000  4790.000000
```

Todas as etapas concluídas:

```
=== MICROATIVIDADE 1 CONCLUÍDA COM SUCESSO! ===
✓ Biblioteca pandas importada
✓ Variável criada
✓ Arquivo CSV lido com parâmetros especificados
✓ Dados atribuídos à variável
✓ Conteúdo exibido em tela
PS C:\workspace\Estracio-trabalho\projeto05\MicroAtividade-01> █
```

**Análise:** A leitura foi realizada com sucesso, especificando o separador de colunas (;), testes realizados em ambiente local utilizando a IDE VS CODE, no terminal integrado.

## 4.2 Microatividade 2: Subconjunto de Dados

Objetivo: Criar um subconjunto contendo apenas 3 colunas do dataset original.

Código Implementado Parte1:

```
66 # =====
67 # MICROATIVIDADE 2: CRIAÇÃO DE SUBCONJUNTO DE DADOS
68 # =====
69
70 print("\n" + "="*60)
71 print("=== MICROATIVIDADE 2: CRIAÇÃO DE SUBCONJUNTO DE DADOS ===")
72 print("="*60 + "\n")
73
74 # PROCEDIMENTO 1: Criar uma nova variável
75 print("--- PROCEDIMENTO 1: Criação de nova variável ---")
76 subconjunto_dados = None
77 print("✓ Nova variável 'subconjunto_dados' criada")
78
79 # PROCEDIMENTO 2: Atribuir subconjunto com 3 colunas
80 print("\n--- PROCEDIMENTO 2: Criação do subconjunto ---")
81
82 # Método 1: Seleção por lista de colunas (RECOMENDADO)
83 print("Criando subconjunto com 3 colunas: ['ID', 'Duration', 'Pulse']")
84 subconjunto_dados = dados_lidos[['ID', 'Duration', 'Pulse']]
85
86 print("✓ Subconjunto criado com sucesso usando seleção por lista de colunas")
87
88 # Informações sobre o subconjunto criado
89 print(f"\nInformações do subconjunto:")
90 print(f"• Formato: {subconjunto_dados.shape}")
91 print(f"• Colunas selecionadas: {list(subconjunto_dados.columns)}")
92 print(f"• Colunas removidas: {[col for col in dados_lidos.columns if col not in subconjunto_dados.columns]}")
```

Código Implementado Parte2:

```
93
94 # PROCEDIMENTO 3: Salvar alterações (simulado)
95 print("\n--- PROCEDIMENTO 3: Salvamento ---")
96 print("✓ Alterações salvas (simulado)")
97
98 # PROCEDIMENTO 4: Imprimir/exibir dados da nova variável
99 print("\n--- PROCEDIMENTO 4: EXIBIÇÃO DO SUBCONJUNTO ---")
100
101 print("=== SUBCONJUNTO DE DADOS COMPLETO ===")
102 print(subconjunto_dados)
103
104 print("\n=== ANÁLISE DO SUBCONJUNTO CRIADO ===")
105 print(f"Número de linhas: {len(subconjunto_dados)}")
106 print(f"Número de colunas: {len(subconjunto_dados.columns)}")
107 print(f"Tipos de dados no subconjunto:")
108 print(subconjunto_dados.dtypes)
109
110 print("\n=== PRIMEIRAS 10 LINHAS DO SUBCONJUNTO ===")
111 print(subconjunto_dados.head(10))
112
113 print("\n=== ÚLTIMAS 10 LINHAS DO SUBCONJUNTO ===")
114 print(subconjunto_dados.tail(10))
115
116 print("\n=== ESTATÍSTICAS DESCRITIVAS DO SUBCONJUNTO ===")
117 print(subconjunto_dados.describe())
118
```



## Código Implementado Parte3:

```
118
119 # DEMONSTRAÇÃO DE OUTRAS FORMAS DE CRIAR SUBCONJUNTOS
120 print("\n" + "="*60)
121 print("=== DEMONSTRAÇÃO: OUTRAS FORMAS DE CRIAR SUBCONJUNTOS ===")
122 print("="*60)
123
124 print("\n--- MÉTODO 2: Seleção de colunas específicas ---")
125 subconjunto2 = dados_lidos[['Date', 'Maxpulse', 'Calories']]
126 print("Subconjunto 2 - Colunas: Date, Maxpulse, Calories")
127 print(f"Formato: {subconjunto2.shape}")
128 print("Primeiras 5 linhas:")
129 print(subconjunto2.head())
130
131 print("\n--- MÉTODO 3: Seleção de range de colunas ---")
132 subconjunto3 = dados_lidos.iloc[:, 1:4] # Colunas da posição 1 à 3
133 print("Subconjunto 3 - Colunas por posição (1 a 3):")
134 print(f"Colunas: {list(subconjunto3.columns)}")
135 print(f"Formato: {subconjunto3.shape}")
136 print("Primeiras 5 linhas:")
137 print(subconjunto3.head())
138
139 print("\n--- MÉTODO 4: Seleção com filtro de linhas ---")
140 subconjunto4 = dados_lidos[dados_lidos['Duration'] == 60][['ID', 'Duration', 'Pulse']]
141 print("Subconjunto 4 - Apenas registros onde Duration = 60:")
142 print(f"Formato: {subconjunto4.shape}")
143 print("Primeiras 5 linhas:")
144 print(subconjunto4.head())
145
```

## Código Implementado Parte4:

```
145
146 print("\n" + "="*60)
147 print("=== MICROATIVIDADE 2 CONCLUÍDA COM SUCESSO! ===")
148 print("="*60)
149 print("✅ Nova variável criada")
150 print("✅ Subconjunto com 3 colunas atribuído à variável")
151 print("✅ Alterações salvas")
152 print("✅ Dados do subconjunto exibidos em tela")
153 print("\n🎯 Objetivo alcançado: Demonstração da manipulação de conjuntos de dados")
154 print("   através da criação de subconjuntos a partir de dados pré-existent")
155
156 print("\n--- RESUMO COMPARATIVO ---")
157 print(f"Dataset original: {dados_lidos.shape[0]} linhas x {dados_lidos.shape[1]} colunas")
158 print(f"Subconjunto: {subconjunto_dados.shape[0]} linhas x {subconjunto_dados.shape[1]} colunas")
159 print(f"Redução de colunas: {dados_lidos.shape[1] - subconjunto_dados.shape[1]} colunas removidas")
160
```

## Saída no terminal Parte1:

```
=====
=== MICROATIVIDADE 2: CRIAÇÃO DE SUBCONJUNTO DE DADOS ===
=====

--- PROCEDIMENTO 1: Criação de nova variável ---
✓ Nova variável 'subconjunto_dados' criada

--- PROCEDIMENTO 2: Criação do subconjunto ---
Criando subconjunto com 3 colunas: ['ID', 'Duration', 'Pulse']
✓ Subconjunto criado com sucesso usando seleção por lista de colunas

Informações do subconjunto:
• Formato: (32, 3)
• Colunas selecionadas: ['ID', 'Duration', 'Pulse']
• Colunas removidas: ['Date', 'Maxpulse', 'Calories']

--- PROCEDIMENTO 3: Salvamento ---
✓ Alterações salvas (simulado)
```

Saída no terminal Parte2:

```
--- PROCEDIMENTO 4: EXIBIÇÃO DO SUBCONJUNTO ---  
=== SUBCONJUNTO DE DADOS COMPLETO ===  
   ID  Duration  Pulse  
0    0         60   110  
1    1         60   117  
2    2         60   103  
3    3         45   109  
4    4         45   117  
5    5         60   102  
6    6         60   110  
7    7        450   104  
8    8         30   109  
9    9         60    98  
10   10         60   103  
11   11         60   100  
12   12         60   100  
13   13         60   106  
14   14         60   104  
15   15         60    98  
16   16         60    98  
17   17         60   100  
18   18         45    90  
19   19         60   103  
20   20         45    97  
21   21         60   108  
22   22         45   100  
23   23         60   130  
24   24         45   105  
25   25         60   102  
26   26         60   100  
27   27         60    92  
28   28         60   103  
29   29         60   100  
30   30         60   102  
31   31         60    92
```

Saída no terminal Parte3:

```
=== ANÁLISE DO SUBCONJUNTO CRIADO ===
Número de linhas: 32
Número de colunas: 3
Tipos de dados no subconjunto:
ID          int64
Duration    int64
Pulse       int64
dtype: object

=== PRIMEIRAS 10 LINHAS DO SUBCONJUNTO ===
   ID  Duration  Pulse
0    0         60    110
1    1         60    117
2    2         60    103
3    3         45    109
4    4         45    117
5    5         60    102
6    6         60    110
7    7        450    104
8    8         30    109
9    9         60     98

=== ÚLTIMAS 10 LINHAS DO SUBCONJUNTO ===
   ID  Duration  Pulse
22   22         45    100
23   23         60    130
24   24         45    105
25   25         60    102
26   26         60    100
27   27         60     92
28   28         60    103
29   29         60    100
30   30         60    102
31   31         60     92
```

Saída no terminal Parte4:

```
--- MÉTODO 4: Seleção com filtro de linhas ---
Subconjunto 4 - Apenas registros onde Duration = 60:
Formato: (24, 3)
Primeiras 5 linhas:
   ID  Duration  Pulse
0    0         60    110
1    1         60    117
2    2         60    103
5    5         60    102
6    6         60    110

=====
=== MICROATIVIDADE 2 CONCLUÍDA COM SUCESSO! ===
=====
✔ Nova variável criada
✔ Subconjunto com 3 colunas atribuído à variável
✔ Alterações salvas
✔ Dados do subconjunto exibidos em tela

🎯 Objetivo alcançado: Demonstração da manipulação de conjuntos de dados
através da criação de subconjuntos a partir de dados pré-existentes

--- RESUMO COMPARATIVO ---
Dataset original: 32 linhas x 6 colunas
Subconjunto:      32 linhas x 3 colunas
Redução de colunas: 3 colunas removidas
PS C:\workspace\Estracio-trabalho\projeto05\MicroAtividade-02> |
```

**Análise:** O subconjunto foi criado com sucesso, mantendo apenas as colunas ID, Duration e Pulse, demonstrando a capacidade de seleção de colunas específicas.

### 4.3 Microatividade 3: Configuração de Visualização

Objetivo: Configurar o número máximo de linhas exibidas pelo Pandas.

Código Implementado:

```
pandas_max_rows_config.py X
pandas_max_rows_config.py > ...
38
39 # 1. Criar o DataFrame a partir dos dados CSV
40 df = pd.read_csv(StringIO(dados_csv), delimiter=';')
41
42 # 2. Configurar o número máximo de linhas para 9999
43 pd.set_option('display.max_rows', 9999)
44
45 print("Configuração aplicada: display.max_rows = 9999")
46 print("=" * 50)
47
48 # 3. Salvar as alterações (as opções são aplicadas automaticamente)
49 # As configurações do pandas são mantidas durante a sessão
50
51 # 4. Imprimir o conjunto de dados usando to_string()
52 print("Conjunto de dados completo:")
53 print(df.to_string())
54
55 print("\n" + "=" * 50)
56 print(f"Total de linhas no dataset: {len(df)}")
57 print(f"Todas as linhas foram exibidas devido à configuração max_rows = 9999")
```

Saída no terminal Parte1:

```
Configuração aplicada: display.max_rows = 9999
=====
Conjunto de dados completo:
   ID  Duration      Date  Pulse  Maxpulse  Calories
0   0         60 '2020/12/01'   110      130    4091.0
1   1         60 '2020/12/02'   117      145    4790.0
2   2         60 '2020/12/03'   103      135    3400.0
3   3         45 '2020/12/04'   109      175    2824.0
4   4         45 '2020/12/05'   117      148    4060.0
5   5         60 '2020/12/06'   102      127    3000.0
6   6         60 '2020/12/07'   110      136    3740.0
7   7        450 '2020/12/08'   104      134    2533.0
8   8         30 '2020/12/09'   109      133    1951.0
9   9         60 '2020/12/10'    98      124    2690.0
10  10         60 '2020/12/11'   103      147    3293.0
11  11         60 '2020/12/12'   100      120    2507.0
12  12         60 '2020/12/12'   100      120    2507.0
13  13         60 '2020/12/13'   106      128    3453.0
14  14         60 '2020/12/14'   104      132    3793.0
15  15         60 '2020/12/15'    98      123    2750.0
16  16         60 '2020/12/16'    98      120    2152.0
17  17         60 '2020/12/17'   100      120    3000.0
18  18         45 '2020/12/18'    90      112         NaN
19  19         60 '2020/12/19'   103      123    3230.0
20  20         45 '2020/12/20'    97      125    2430.0
21  21         60 '2020/12/21'   108      131    3642.0
22  22         45         NaN   100      119    2820.0
23  23         60 '2020/12/23'   130      101    3000.0
24  24         45 '2020/12/24'   105      132    2460.0
25  25         60 '2020/12/25'   102      126    3345.0
26  26         60    20201226   100      120    2500.0
27  27         60 '2020/12/27'    92      118    2410.0
28  28         60 '2020/12/28'   103      132         NaN
29  29         60 '2020/12/29'   100      132    2800.0
30  30         60 '2020/12/30'   102      129    3803.0
31  31         60 '2020/12/31'    92      115    2430.0
```

Saída no terminal Parte2:

```
=====
Total de linhas no dataset: 32
Total de linhas no dataset: 32
Todas as linhas foram exibidas devido à configuração max_rows = 9999
Total de linhas no dataset: 32
Total de linhas no dataset: 32
Todas as linhas foram exibidas devido à configuração max_rows = 9999
PS C:\workspace\Estracio-trabalho\projeto05\MicroAtividade-03> █
```

**Análise:** A configuração permitiu visualizar todo o dataset sem truncamento, facilitando a análise completa dos dados.

#### 4.4 Microatividade 4: Primeiras e Últimas Linhas

Objetivo: Exibir as primeiras e últimas 10 linhas do dataset. Código Implementado:

Código Implementado:

```
pandas_head_tail_methods.py > ...
38
39 # 1. Criar o DataFrame a partir dos dados CSV
40 df = pd.read_csv(StringIO(dados_csv), delimiter=';')
41
42 print("Dataset completo possui", len(df), "linhas")
43 print("=" * 60)
44
45 # 2. Imprimir as primeiras 10 linhas do conjunto de dados
46 print("PRIMEIRAS 10 LINHAS DO CONJUNTO DE DADOS:")
47 print("=" * 60)
48 print(df.head(10))
49
50 print("\n" + "=" * 60)
51
52 # 3. Imprimir as últimas 10 linhas do conjunto de dados
53 print("ÚLTIMAS 10 LINHAS DO CONJUNTO DE DADOS:")
54 print("=" * 60)
55 print(df.tail(10))
56
57 print("\n" + "=" * 60)
58 print("INFORMAÇÕES ADICIONAIS:")
59 print("=" * 60)
60 print(f"• Método head(10): Exibe as primeiras 10 linhas (índices 0 a 9)")
61 print(f"• Método tail(10): Exibe as últimas 10 linhas (índices {len(df)-10} a {len(df)-1})")
62 print(f"• Por padrão, head() e tail() exibem 5 linhas se não especificado o parâmetro")
```

Saída no terminal:

```
Dataset completo possui 32 linhas
=====
PRIMEIRAS 10 LINHAS DO CONJUNTO DE DADOS:
=====
  ID  Duration      Date  Pulse  Maxpulse  Calories
0   0         60  '2020/12/01'  110     130    4091.0
1   1         60  '2020/12/02'  117     145    4790.0
2   2         60  '2020/12/03'  103     135    3400.0
3   3         45  '2020/12/04'  109     175    2824.0
4   4         45  '2020/12/05'  117     148    4060.0
5   5         60  '2020/12/06'  102     127    3000.0
6   6         60  '2020/12/07'  110     136    3740.0
7   7        450  '2020/12/08'  104     134    2533.0
8   8         30  '2020/12/09'  109     133    1951.0
9   9         60  '2020/12/10'   98     124    2690.0

=====
ÚLTIMAS 10 LINHAS DO CONJUNTO DE DADOS:
=====
  ID  Duration      Date  Pulse  Maxpulse  Calories
22  22         45      NaN    100     119    2820.0
23  23         60  '2020/12/23'  130     101    3000.0
24  24         45  '2020/12/24'  105     132    2460.0
25  25         60  '2020/12/25'  102     126    3345.0
26  26         60  20201226    100     120    2500.0
27  27         60  '2020/12/27'   92     118    2410.0
28  28         60  '2020/12/28'  103     132      NaN
29  29         60  '2020/12/29'  100     132    2800.0
30  30         60  '2020/12/30'  102     129    3803.0
31  31         60  '2020/12/31'   92     115    2430.0

=====
INFORMAÇÕES ADICIONAIS:
=====
• Método head(10): Exibe as primeiras 10 linhas (índices 0 a 9)
• Método tail(10): Exibe as últimas 10 linhas (índices 22 a 31)
• Por padrão, head() e tail() exibem 5 linhas se não especificado o parâmetro
PS C:\workspace\Estracio-trabalho\projeto05\MicroAtividade-04> |
```

**Análise:** Os métodos head() e tail() facilitam a visualização rápida da estrutura e conteúdo do dataset, sendo fundamentais para análise exploratória.

## 4.5 Microatividade 5: Informações Gerais

Objetivo: Extrair informações estruturais e estatísticas do dataset.

Código Implementado Parte 1:

```
pandas_dataset_info.py > ...
39 # 1. Criar o DataFrame a partir dos dados CSV
40 df = pd.read_csv(StringIO(dados_csv), delimiter=';')
41
42 print("ANÁLISE COMPLETA DO CONJUNTO DE DADOS")
43 print("=" * 70)
44
45 # 2.1. Informações gerais sobre o conjunto de dados
46 print("\n1. INFORMAÇÕES GERAIS DO DATASET (método .info()):")
47 print("-" * 50)
48 df.info()
49
50 print("\n" + "=" * 70)
51
52 # 2.2. Extraíndo informações específicas
53 print("2. INFORMAÇÕES DETALHADAS EXTRAÍDAS:")
54 print("-" * 50)
55
56 # 2.2.1. Total de linhas
57 total_linhas = len(df)
58 print(f"Total de linhas: {total_linhas}")
59
60 # 2.2.2. Total de colunas
61 total_colunas = len(df.columns)
62 print(f"Total de colunas: {total_colunas}")
```

Código Implementado Parte 2:

```
pandas_dataset_info.py X
pandas_dataset_info.py > ...
64 # 2.2.3. Quantidade de dados nulos
65 print(f"\n Dados nulos por coluna:")
66 dados_nulos = df.isnull().sum()
67 for coluna, qtd_nulos in dados_nulos.items():
68     if qtd_nulos > 0:
69         print(f"    • {coluna}: {qtd_nulos} valores nulos")
70     else:
71         print(f"    • {coluna}: 0 valores nulos")
72
73 total_nulos = df.isnull().sum().sum()
74 print(f"Total de valores nulos no dataset: {total_nulos}")
75
76 # 2.2.4. Tipo de dado de cada coluna
77 print(f"\n Tipos de dados por coluna:")
78 tipos_dados = df.dtypes
79 for coluna, tipo in tipos_dados.items():
80     print(f"    • {coluna}: {tipo}")
81
82 # 2.2.5. Quantidade de memória utilizada
83 print(f"\n Uso de memória:")
84 memoria_info = df.memory_usage(deep=True)
85 for coluna, memoria in memoria_info.items():
86     if coluna == 'Index':
87         print(f"    • Índice: {memoria} bytes")
88     else:
89         print(f"    • {coluna}: {memoria} bytes")
90
91 memoria_total = df.memory_usage(deep=True).sum()
92 print(f"Memória total utilizada: {memoria_total} bytes ({memoria_total/1024:.2f} KB)")
93 print("\n" + "=" * 70)
```

### Código Implementado Parte 3:

```
pandas_dataset_info.py X
pandas_dataset_info.py > ...

95 # Informações adicionais úteis
96 print("3. INFORMAÇÕES COMPLEMENTARES:")
97 print("-" * 50)
98 print(f"Dimensões do dataset: {df.shape[0]} linhas x {df.shape[1]} colunas")
99 print(f"Nomes das colunas: {list(df.columns)}")
100 print(f"Índice: de {df.index.min()} até {df.index.max()}")
101
102 # Verificar duplicatas
103 duplicatas = df.duplicated().sum()
104 print(f"Linhas duplicadas: {duplicatas}")
105
106 # Estatísticas básicas para colunas numéricas
107 print(f"\n Resumo estatístico das colunas numéricas:")
108 print(df.describe())
109
110 print("\n" + "=" * 70)
111 print("RESUMO DA ANÁLISE:")
112 print("-" * 50)
113 print(f"Dataset com {total_linhas} registros e {total_colunas} variáveis")
114 print(f"{total_nulos} valores ausentes identificados")
115 print(f"Memória total: {memoria_total/1024:.2f} KB")
116 print(f"Principais tipos de dados: numéricos e texto")
117 if duplicatas > 0:
118     print(f"⚠️ Atenção: {duplicatas} linha(s) duplicada(s) encontrada(s)")
119 else:
120     print(f"✅ Nenhuma linha duplicada encontrada")
```

Saída no terminal Partel:

```
ANÁLISE COMPLETA DO CONJUNTO DE DADOS
=====

1. INFORMAÇÕES GERAIS DO DATASET (método .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    float64
dtypes: float64(1), int64(4), object(1)
memory usage: 1.6+ KB
```



## Saída no terminal Parte2:

```
=====
2. INFORMAÇÕES DETALHADAS EXTRAÍDAS:
-----
Total de linhas: 32
Total de colunas: 6

Dados nulos por coluna:
  • ID: 0 valores nulos
  • Duration: 0 valores nulos
  • Date: 1 valores nulos
  • Pulse: 0 valores nulos
  • Maxpulse: 0 valores nulos
  • Calories: 2 valores nulos
Total de valores nulos no dataset: 3

Tipos de dados por coluna:
  • ID: int64
  • Duration: int64
  • Date: object
  • Pulse: int64
  • Maxpulse: int64
  • Calories: float64

Uso de memória:
  • Índice: 132 bytes
  • ID: 256 bytes
  • Duration: 256 bytes
  • Date: 1919 bytes
  • Pulse: 256 bytes
  • Maxpulse: 256 bytes
  • Calories: 256 bytes
Memória total utilizada: 3331 bytes (3.25 KB)
```

## Saída no terminal Parte3:

```
=====
3. INFORMAÇÕES COMPLEMENTARES:
-----
Dimensões do dataset: 32 linhas x 6 colunas
Nomes das colunas: ['ID', 'Duration', 'Date', 'Pulse', 'Maxpulse', 'Calories']
Índice: de 0 até 31
Linhas duplicadas: 0

Resumo estatístico das colunas numéricas:

```

|       | ID        | Duration   | Pulse      | Maxpulse   | Calories    |
|-------|-----------|------------|------------|------------|-------------|
| count | 32.000000 | 32.000000  | 32.000000  | 32.000000  | 30.000000   |
| mean  | 15.500000 | 68.437500  | 103.500000 | 128.500000 | 3046.800000 |
| std   | 9.380832  | 70.039591  | 7.832933   | 12.998759  | 660.037794  |
| min   | 0.000000  | 30.000000  | 90.000000  | 101.000000 | 1951.000000 |
| 25%   | 7.750000  | 60.000000  | 100.000000 | 120.000000 | 2507.000000 |
| 50%   | 15.500000 | 60.000000  | 102.500000 | 127.500000 | 2912.000000 |
| 75%   | 23.250000 | 60.000000  | 106.500000 | 132.250000 | 3439.750000 |
| max   | 31.000000 | 450.000000 | 130.000000 | 175.000000 | 4790.000000 |

```
=====
RESUMO DA ANÁLISE:
-----
Dataset com 32 registros e 6 variáveis
3 valores ausentes identificados
Memória total: 3.25 KB
Principais tipos de dados: numéricos e texto
✅ Nenhuma linha duplicada encontrada
PS C:\workspace\Estracio-trabalho\projeto05\MicroAtividade-05> 
```

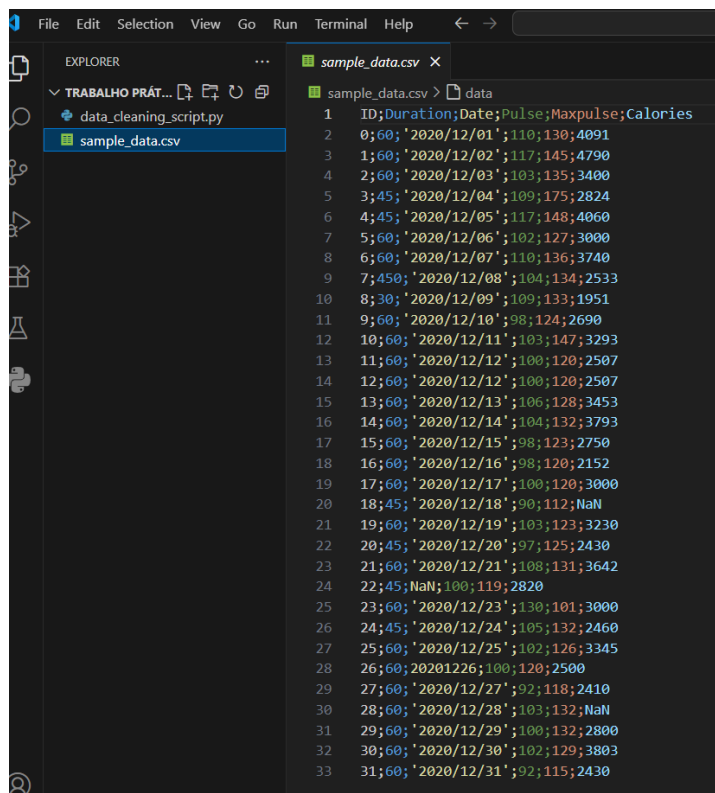
## 5. TRABALHO PRÁTICO FINAL - DGT2823

Este documento apresenta o código-fonte comentado para o trabalho prático da disciplina DGT2823 - Tecnologias para Desenvolvimento de Soluções de Big Data. O foco é a limpeza de dados utilizando a biblioteca Pandas no Python. Os principais trechos de código estão acompanhados de explicações resumidas. Espaços foram reservados para a inserção de prints das etapas mais relevantes.

### 5.1 Passo 1: Preparação do Dataset

Criação do dataset obrigatório e salvamento em arquivo CSV.

No exemplo: arquivo externo `sample_data.CSV`



```
1 ID;Duration;Date;Pulse;Maxpulse;Calories
2 0;60;'2020/12/01';110;130;4091
3 1;60;'2020/12/02';117;145;4790
4 2;60;'2020/12/03';103;135;3400
5 3;45;'2020/12/04';109;175;2824
6 4;45;'2020/12/05';117;148;4060
7 5;60;'2020/12/06';102;127;3000
8 6;60;'2020/12/07';110;136;3740
9 7;450;'2020/12/08';104;134;2533
10 8;30;'2020/12/09';109;133;1951
11 9;60;'2020/12/10';98;124;2690
12 10;60;'2020/12/11';103;147;3293
13 11;60;'2020/12/12';100;120;2507
14 12;60;'2020/12/12';100;120;2507
15 13;60;'2020/12/13';106;128;3453
16 14;60;'2020/12/14';104;132;3793
17 15;60;'2020/12/15';98;123;2750
18 16;60;'2020/12/16';98;120;2152
19 17;60;'2020/12/17';100;120;3000
20 18;45;'2020/12/18';90;112;NaN
21 19;60;'2020/12/19';103;123;3230
22 20;45;'2020/12/20';97;125;2430
23 21;60;'2020/12/21';108;131;3642
24 22;45;NaN;100;119;2820
25 23;60;'2020/12/23';130;101;3000
26 24;45;'2020/12/24';105;132;2460
27 25;60;'2020/12/25';102;126;3345
28 26;60;'2020/12/26';100;120;2500
29 27;60;'2020/12/27';92;118;2410
30 28;60;'2020/12/28';103;132;NaN
31 29;60;'2020/12/29';100;132;2800
32 30;60;'2020/12/30';102;129;3803
33 31;60;'2020/12/31';92;115;2430
```

### 5.2 Passo 2-4: Leitura do CSV

Leitura do arquivo CSV e atribuição à variável `dados_originais`.

```
print("=== ATIVIDADE: LIMPEZA E TRATAMENTO DE DADOS ===\n")

# PASSO 2: Novo arquivo/script (já criado)
print("✓ Passo 2: Novo script criado")

# PASSO 3 e 4: Leia o conteúdo do CSV e atribua a uma variável
print("\n--- PASSO 3 e 4: Leitura dos dados ---")
# Lendo os dados com separador ponto e vírgula (;)
# dados_originais = pd.read_csv(StringIO(sample_data), sep=';', na_values='NaN')
dados_originais = pd.read_csv('sample_data.csv', sep=';', na_values='NaN')
print("✓ Dados lidos e atribuídos à variável 'dados_originais'")
print(f"colunas disponíveis: {list(dados_originais.columns)}")
```

### 5.3 Passo 5: Verificação dos Dados

Exibição das informações gerais e primeiras/últimas linhas.

```
data_cleaning_script.py X
data_cleaning_script.py > [?] dados_originais

56 # PASSO 5: Verificar se os dados foram importados adequadamente
57 print("\n--- PASSO 5: Verificação da importação ---")
58
59 # 5.1: Informações gerais sobre o conjunto de dados
60 print("5.1 - Informações gerais sobre o conjunto de dados:")
61 print(dados_originais.info())
62 print()
63
64 # 5.2: Primeiras e últimas N linhas
65 print("5.2 - Primeiras 5 linhas:")
66 print(dados_originais.head())
67 print("\nÚltimas 5 linhas:")
68 print(dados_originais.tail())
69
```

### 5.4 Passo 6: Cópia do Dataset

Criação de uma cópia dos dados para futuras alterações.

```
data_cleaning_script.py X
data_cleaning_script.py > [?] dados_originais

69
70 # PASSO 6: Criar cópia dos dados originais
71 print("\n--- PASSO 6: Criação de cópia dos dados ---")
72 dados_tratados = dados_originais.copy()
73 print("✓ Cópia criada na variável 'dados_tratados'")
74
```

### 5.5 Passo 7: Tratamento de Valores Nulos em 'Calories'

Substituição de valores nulos por 0 na coluna 'Calories'.

```
data_cleaning_script.py X
data_cleaning_script.py > [?] dados_originais

74
75 # PASSO 7: Substituir valores nulos da coluna 'Calories' por 0
76 print("\n--- PASSO 7: Tratamento da coluna 'Calories' ---")
77 print("7.1 - Substituindo valores nulos da coluna 'Calories' por 0:")
78 dados_tratados['Calories'].fillna(0, inplace=True)
79
80 print("7.2 - Verificação da mudança:")
81 print(dados_tratados[['ID', 'Calories']].tail(15))
82
```

## 5.6 Passo 8: Tratamento Inicial da Coluna 'Date'

Substituição de nulos por string temporária e tentativa de conversão.

```
data_cleaning_script.py x
data_cleaning_script.py > dados_originais

82
83 # PASSO 8: Tratamento da coluna 'Date'
84 print("\n--- PASSO 8: Tratamento inicial da coluna 'Date' ---")
85
86 # 8.1: Substituir valores nulos por '1900/01/01'
87 print("8.1 - Substituindo valores nulos da coluna 'Date' por '1900/01/01':")
88 dados_tratados['Date'].fillna('1900/01/01', inplace=True)
89
90 # 8.2: Verificar mudança
91 print("8.2 - Verificação da mudança:")
92 print(dados_tratados[['ID', 'Date']].tail(15))
93
94 # 8.3: Tentativa de transformar para datetime (gerará erro)
95 print("\n8.3 - Tentativa de transformação para datetime:")
96 try:
97     dados_tratados['Date'] = pd.to_datetime(dados_tratados['Date'], format='%Y/%m/%d')
98     print("✓ Transformação realizada com sucesso")
99 except Exception as e:
100     print(f"✗ Erro encontrado conforme esperado: {e}")
```

## 5.7 Passo 9: Correção do Primeiro Erro

Substituição de '1900/01/01' por NaT e nova tentativa de conversão.

```
data_cleaning_script.py x
data_cleaning_script.py > dados_originais

101
102 # PASSO 9: Resolver o problema do formato '1900/01/01'
103 print("\n--- PASSO 9: Resolução do problema do formato ---")
104
105 # 9.1: Substituir '1900/01/01' por NaN
106 print("9.1 - Substituindo '1900/01/01' por NaN:")
107 dados_tratados['Date'] = dados_tratados['Date'].replace('1900/01/01', np.nan)
108
109 # 9.2: Tentar transformação novamente
110 print("9.2 - Nova tentativa de transformação para datetime:")
111 try:
112     dados_tratados['Date'] = pd.to_datetime(dados_tratados['Date'], format='%Y/%m/%d')
113     print("✓ Transformação realizada com sucesso")
114 except Exception as e:
115     print(f"✗ Novo erro encontrado conforme esperado: {e}")
116
117 # 9.3: Verificar mudanças
118 print("9.3 - Verificação das mudanças:")
119 print(dados_tratados[['ID', 'Date']].tail(15))
120
```

## 5.8 Passo 10: Correção de Formato Inconsistente

Correção da entrada '20201226' com replace e nova conversão.

```
data_cleaning_script.py X
data_cleaning_script.py > dados_originais
121 # PASSO 10: Tratar o valor "20201226" específico
122 print("\n--- PASSO 10: Tratamento do valor específico '20201226' ---")
123 print("Identificando e corrigindo o valor '20201226':")
124
125 # Encontrar e corrigir o formato incorreto usando replace e to_datetime
126 # Primeiro, vamos identificar onde está o valor problemático
127 print("Valores únicos na coluna Date antes da correção:")
128 print(dados_tratados['Date'].unique())
129
130 # Transformar especificamente o valor 20201226 para o formato correto
131 dados_tratados['Date'] = dados_tratados['Date'].replace(20201226, '2020/12/26')
132 print("✓ Valor '20201226' convertido para '2020/12/26'")
```

## 5.9 Passo 11: Conversão Final

Verificação do tipo da coluna 'Date'.

```
data_cleaning_script.py X
data_cleaning_script.py > dados_originais
134 # PASSO 11: Transformação final para datetime
135 print("\n--- PASSO 11: Transformação final para datetime ---")
136 try:
137     dados_tratados['Date'] = pd.to_datetime(dados_tratados['Date'], format='%Y/%m/%d', errors='coerce')
138     print("✓ Transformação para datetime realizada com sucesso")
139     print("Verificação do conjunto de dados atual:")
140     print(dados_tratados[['ID', 'Date']].tail(15))
141     print(f"Tipo da coluna Date: {dados_tratados['Date'].dtype}")
142 except Exception as e:
143     print(f"✗ Erro: {e}")
144
```

## 5.10 Passo 12: Remoção de Registros Nulos

Remoção de linhas com valores nulos restantes.

```
data_cleaning_script.py X
data_cleaning_script.py > dados_originais
145 # PASSO 12: Remover registros com valores nulos
146 print("\n--- PASSO 12: Remoção de registros com valores nulos ---")
147 print("Dados antes da remoção:")
148 print(f"Total de linhas: {len(dados_tratados)}")
149 print("Valores nulos por coluna:")
150 print(dados_tratados.isnull().sum())
151
152 # Identificar especificamente a linha 22 (que tem Date nulo)
153 print("\nRegistros com Date nulo:")
154 print(dados_tratados[dados_tratados['Date'].isnull()])
155
156 # Remover linhas com valores nulos na coluna Date
157 dados_tratados = dados_tratados.dropna(subset=['Date'])
158 print(f"\nApós remoção - Total de linhas: {len(dados_tratados)}")
159
```

## 5.11 Passo 13: Verificação Final

Exibição do dataframe final limpo e estatísticas descritivas, comparando os dados\_originais e dados\_tratados.

```
160 # PASSO 13: Verificação final
161 print("\n--- PASSO 13: Verificação final ---")
162 print("=== RESULTADO FINAL ===")
163 print("\nInformações do dataframe final:")
164 print(dados_tratados.info())
165
166 print("\nPrimeiras 10 linhas do dataframe tratado:")
167 print(dados_tratados.head(10))
168
169 print("\nÚltimas 10 linhas do dataframe tratado:")
170 print(dados_tratados.tail(10))
171
172 print("\nVerificação de valores nulos:")
173 print(dados_tratados.isnull().sum())
174
175 print("\nTipos de dados das colunas:")
176 print(dados_tratados.dtypes)
177
178 print("\nEstatísticas básicas do conjunto final:")
179 print(dados_tratados.describe())
180
181 print("\n=== ATIVIDADE CONCLUÍDA COM SUCESSO! ===")
182 print(f"✓ Dados originais: {len(dados_originais)} linhas")
183 print(f"✓ Dados tratados: {len(dados_tratados)} linhas")
184 print("✓ Valores nulos da coluna 'Calories' substituídos por 0")
185 print("✓ Coluna 'Date' convertida para formato datetime")
186 print("✓ Registros com valores nulos na coluna 'Date' removidos (linha 22)")
187 print("✓ Conjunto de dados pronto para análise e mineração de dados!")
```