POLO BARRA WORLD

CURSO DESENVOLVIMENTO FULL STACK

Disciplina: Nível 5– Tecnologias para desenv. De Soluções de Big Data

Turma: 9001 / 5º Semestre

Aluno: Paola Savedra Barreiros

Matrícula: 2023.0701.473-7

Repositório Github: https://github.com/pasavedra/DGT2823-Tecnologias-para-desenv.-de-solucoes-de-big-data

Missão Prática | Nível 3 | Mundo 5

Título da prática: DGT2823 - Tecnologias para desenv. de soluções de big data

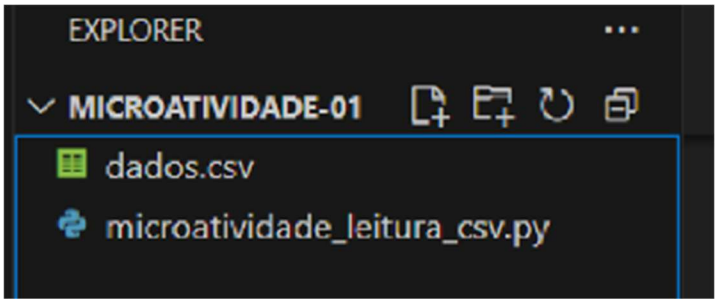
Objetivo da Prática:

- 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 Panda (Python); Descrever como exibir informações gerais sobre colunas,

Linhas e dados de um conjunto de dados usando a biblioteca Pandas (Python)

Microatividades

Microatividades 01



Parte 01:

```python
# Microatividade 1: Descrever como ler um arquivo CSV usando a biblioteca Pandas
# Analista de Dados - Leitura de dados externos

# PROCEDIMENTO 2.1: Importe a biblioteca pandas
import pandas as pd

print("=== MICROATIVIDADE 1: LEITURA DE ARQUIVO CSV COM PANDAS ===\n")
print("✓ Passo 2.1: Biblioteca pandas importada com sucesso")

# PROCEDIMENTO 2.2: Cria uma variável
dados_lidos = None
print("✓ Passo 2.2: Variável 'dados_lidos' criada")

# PROCEDIMENTO 2.3 e 2.4: Leia o conteúdo do arquivo CSV e atribua à variável
print("\n--- LEITURA DO ARQUIVO CSV ---")

# DEMONSTRAÇÃO COM ARQUIVO REAL:
# Para ler um arquivo CSV real localizado na pasta raiz do projeto, use:
dados_lidos = pd.read_csv('dados.csv', sep=';', engine='python', encoding='utf-8'
print("✓ Passo 2.3 e 2.4: Arquivo CSV lido e dados atribuídos à variável 'dados_l

# Informações sobre os parâmetros utilizados
print("\n--- PARÂMETROS UTILIZADOS NA LEITURA ---")
print("• sep=';'        → Define o separador de colunas como ponto e vírgula")
print("• engine='python' → Especifica o motor de análise Python")
print("• encoding='utf-8' → Define a codificação de caracteres")
```

Parte 02:

```python
# PROCEDIMENTO 2.5: Imprima/exiba em tela os dados da variável
print("\n--- PASSO 2.5: EXIBIÇÃO DOS DADOS LIDOS ---")

print("=== DADOS COMPLETOS DO ARQUIVO CSV ===")
print(dados_lidos)

print("\n=== INFORMAÇÕES ADICIONAIS SOBRE OS DADOS ===")
print(f"Formato do DataFrame: {dados_lidos.shape}")
print(f"Colunas: {list(dados_lidos.columns)}")
print(f"Tipos de dados:")
print(dados_lidos.dtypes)

print("\n=== PRIMEIRAS 5 LINHAS ===")
print(dados_lidos.head())

print("\n=== ÚLTIMAS 5 LINHAS ===")
print(dados_lidos.tail())

print("\n=== ESTATÍSTICAS BÁSICAS ===")
print(dados_lidos.describe())

print("\n=== MICROATIVIDADE 1 CONCLUÍDA COM SUCESSO! ===")
print("✅ Biblioteca pandas importada")
print("✅ Variável criada")
print("✅ Arquivo CSV lido com parâmetros especificados")
print("✅ Dados atribuídos à variável")
print("✅ Conteúdo exibido em tela")
```

Resultado

Saída Terminal Parte 01:

```
=== 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

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

Saída Terminal Parte 02:

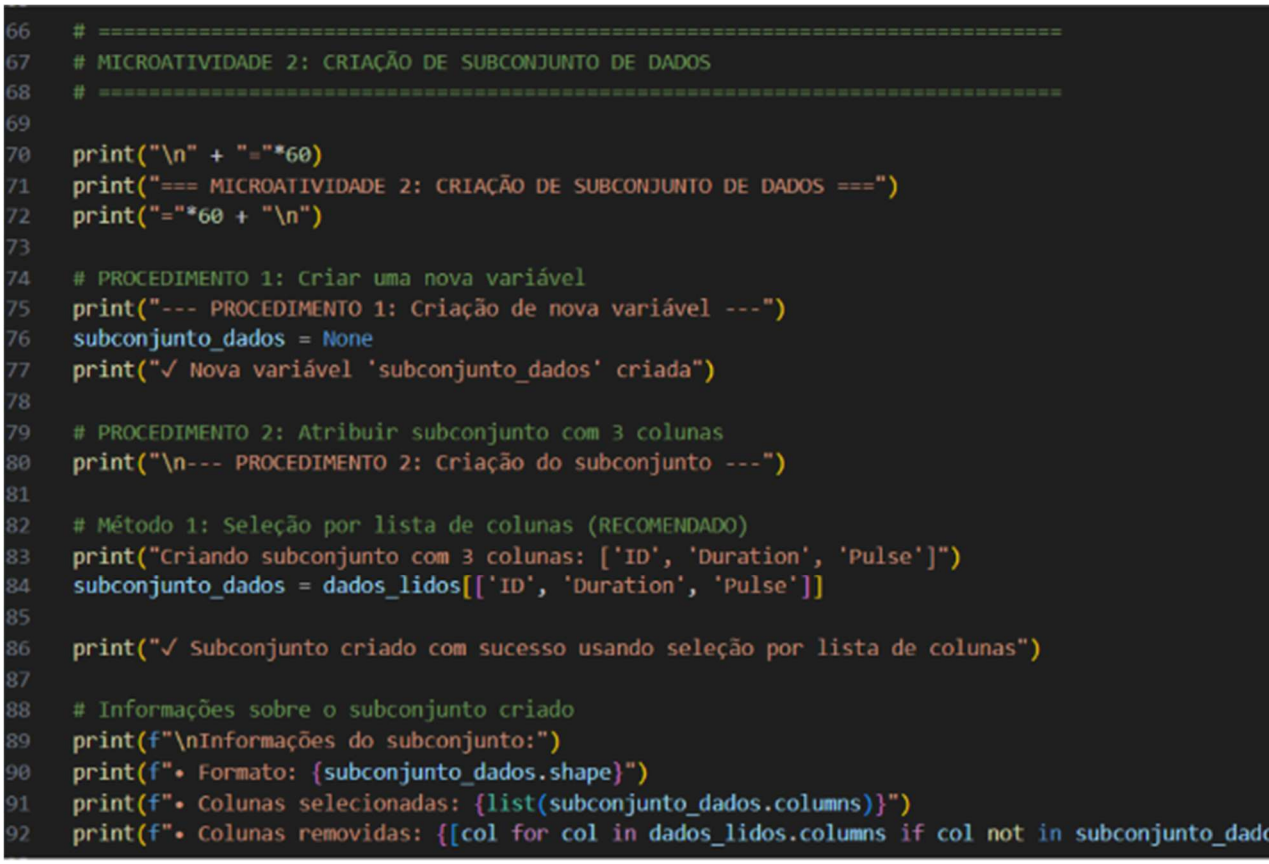
```
--- PASSO 2.5: EXIBIÇÃO DOS DADOS LIDOS ---
=== DADOS COMPLETOS DO ARQUIVO CSV ===
    ID  Duration          Date  Pulse  Maxpulse  Calories
0    0        60  '2020/12/01'    110       130    4091.0
1    1        60  '2020/12/02'    117       145    4790.0
2    2        60  '2020/12/03'    103       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:

```
=== INFORMAÇÕES ADICIONAIS SOBRE OS DADOS ===
Formato do DataFrame: (32, 6)
Colunas: ['ID', 'Duration', 'Dale', '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
```

Microatividades 02:

Código Parte 01:



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

print("\n" + "="*60)
print("=== MICROATIVIDADE 2: CRIAÇÃO DE SUBCONJUNTO DE DADOS ===")
print("="*60 + "\n")

# PROCEDIMENTO 1: Criar uma nova variável
print("--- PROCEDIMENTO 1: Criação de nova variável ---")
subconjunto_dados = None
print("✓ Nova variável 'subconjunto_dados' criada")

# PROCEDIMENTO 2: Atribuir subconjunto com 3 colunas
print("\n--- PROCEDIMENTO 2: Criação do subconjunto ---")

# Método 1: Seleção por lista de colunas (RECOMENDADO)
print("Criando subconjunto com 3 colunas: ['ID', 'Duration', 'Pulse']")
subconjunto_dados = dados_lidos[['ID', 'Duration', 'Pulse']]

print("✓ Subconjunto criado com sucesso usando seleção por lista de colunas")

# Informações sobre o subconjunto criado
print(f"\nInformações do subconjunto:")
print(f"• Formato: {subconjunto_dados.shape}")
print(f"• Colunas selecionadas: {list(subconjunto_dados.columns)}")
print(f"• Colunas removidas: {[col for col in dados_lidos.columns if col not in subconjunto_dado
```

Código Parte 02:

```python
# PROCEDIMENTO 3: Salvar alterações (simulado)
print("\n--- PROCEDIMENTO 3: Salvamento ---")
print("✓ Alterações salvas (simulado)")

# PROCEDIMENTO 4: Imprimir/exibir dados da nova variável
print("\n--- PROCEDIMENTO 4: EXIBIÇÃO DO SUBCONJUNTO ---")

print("=== SUBCONJUNTO DE DADOS COMPLETO ===")
print(subconjunto_dados)

print("\n=== ANÁLISE DO SUBCONJUNTO CRIADO ===")
print(f"Número de linhas: {len(subconjunto_dados)}")
print(f"Número de colunas: {len(subconjunto_dados.columns)}")
print(f"Tipos de dados no subconjunto:")
print(subconjunto_dados.dtypes)

print("\n=== PRIMEIRAS 10 LINHAS DO SUBCONJUNTO ===")
print(subconjunto_dados.head(10))

print("\n=== ÚLTIMAS 10 LINHAS DO SUBCONJUNTO ===")
print(subconjunto_dados.tail(10))

print("\n=== ESTATÍSTICAS DESCRITIVAS DO SUBCONJUNTO ===")
print(subconjunto_dados.describe())
```

Código Parte 03:

```python
# DEMONSTRAÇÃO DE OUTRAS FORMAS DE CRIAR SUBCONJUNTOS
print("\n" + "="*60)
print("=== DEMONSTRAÇÃO: OUTRAS FORMAS DE CRIAR SUBCONJUNTOS ===")
print("="*60)

print("\n--- MÉTODO 2: Seleção de colunas específicas ---")
subconjunto2 = dados_lidos[['Date', 'Maxpulse', 'Calories']]
print("Subconjunto 2 - Colunas: Date, Maxpulse, Calories")
print(f"Formato: {subconjunto2.shape}")
print("Primeiras 5 linhas:")
print(subconjunto2.head())

print("\n--- MÉTODO 3: Seleção de range de colunas ---")
subconjunto3 = dados_lidos.iloc[:, 1:4]  # Colunas da posição 1 à 3
print("Subconjunto 3 - Colunas por posição (1 a 3):")
print(f"Colunas: {list(subconjunto3.columns)}")
print(f"Formato: {subconjunto3.shape}")
print("Primeiras 5 linhas:")
print(subconjunto3.head())

print("\n--- MÉTODO 4: Seleção com filtro de linhas ---")
subconjunto4 = dados_lidos[dados_lidos['Duration'] == 60][['ID', 'Duration', 'Pulse']]
print("Subconjunto 4 - Apenas registros onde Duration = 60:")
print(f"Formato: {subconjunto4.shape}")
print("Primeiras 5 linhas:")
print(subconjunto4.head())
```

Código Parte 04:

```
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-existentes")
155
156   print("\n--- RESUMO COMPARATIVO ---")
157   print(f"Dataset original:  {dados_lidos.shape[0]} linhas × {dados_lidos.shape[1]} colunas")
158   print(f"Subconjunto:       {subconjunto_dados.shape[0]} linhas × {subconjunto_dados.shape[1]} colunas")
159   print(f"Redução de colunas: {dados_lidos.shape[1] - subconjunto_dados.shape[1]} colunas removidas")
```

Saída Terminal Parte 01:

```
============================================================
=== 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 Terminal Parte 02:

```
--- 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 Terminal Parte 03:

```
--- 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 Terminal Parte 04:

```
--- 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
```

Microatividades 03:

Código:

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("Todas as linhas foram exibidas devido à configuração max_rows = 9999")
```

Saída Parte 01:

```
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 Parte 2:

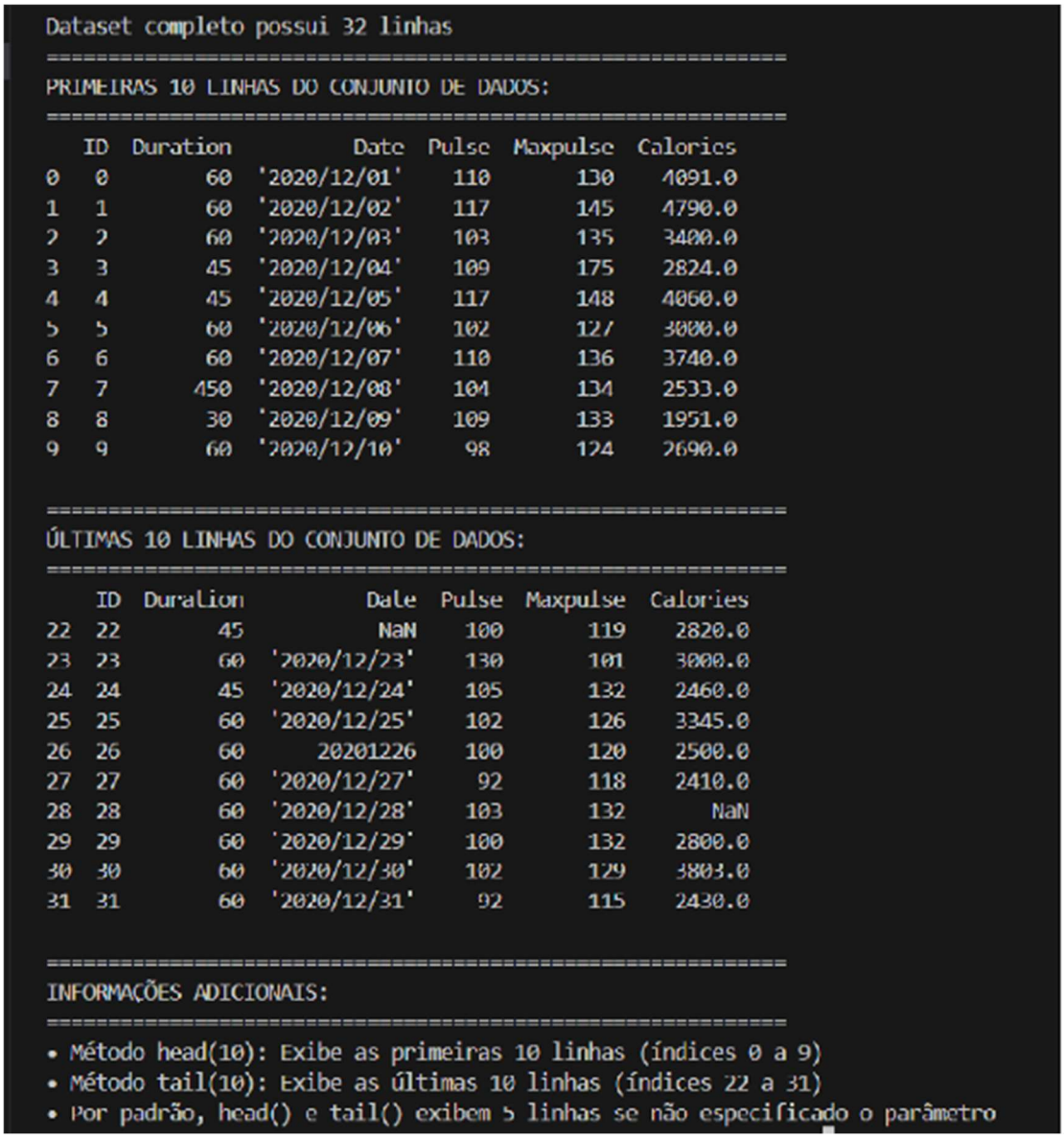
```
================================================
Total de linhas no dataset: 32
Total de linhas no dataset: 32
Todas as linhas foram exibidas devido à configuração max_rows = 9999
Total de linhas no dataset: 32
Total de linhas no dataset: 32
Todas as linhas foram exibidas devido à configuração max_rows = 9999
```

Microatividades 04:

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:



```
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
```

Microatividade 05:

Código Parte 01:

pandas_dataset_info.py > ...

```python
# 1. Criar o DataFrame a partir dos dados CSV
df = pd.read_csv(StringIO(dados_csv), delimiter=';')

print("ANÁLISE COMPLETA DO CONJUNTO DE DADOS")
print("=" * 70)

# 2.1. Informações gerais sobre o conjunto de dados
print("\n1. INFORMAÇÕES GERAIS DO DATASET (método .info()):")
print("-" * 50)
df.info()

print("\n" + "=" * 70)

# 2.2. Extraindo informações específicas
print("2. INFORMAÇÕES DETALHADAS EXTRAÍDAS:")
print("-" * 50)

# 2.2.1. Total de linhas
total_linhas = len(df)
print(f"Total de linhas: {total_linhas}")

# 2.2.2. Total de colunas
total_colunas = len(df.columns)
print(f"Total de colunas: {total_colunas}")
```

Código Parte 02:

pandas_dataset_info.py

```python
# 2.2.3. Quantidade de dados nulos
print(f"\n Dados nulos por coluna:")
dados_nulos = df.isnull().sum()
for coluna, qtd_nulos in dados_nulos.items():
    if qtd_nulos > 0:
        print(f"   • {coluna}: {qtd_nulos} valores nulos")
    else:
        print(f"   • {coluna}: 0 valores nulos")

total_nulos = df.isnull().sum().sum()
print(f"Total de valores nulos no dataset: {total_nulos}")

# 2.2.4. Tipo de dado de cada coluna
print(f"\n Tipos de dados por coluna:")
tipos_dados = df.dtypes
for coluna, tipo in tipos_dados.items():
    print(f"   • {coluna}: {tipo}")

# 2.2.5. Quantidade de memória utilizada
print(f"\n Uso de memória:")
memoria_info = df.memory_usage(deep=True)
for coluna, memoria in memoria_info.items():
    if coluna == 'Index':
        print(f"   • Índice: {memoria} bytes")
    else:
        print(f"   • {coluna}: {memoria} bytes")

memoria_total = df.memory_usage(deep=True).sum()
print(f"Memória total utilizada: {memoria_total} bytes ({memoria_total/1024:.2f} KB)")
print("\n" + "=" * 70)
```

Código Parte 03:

pandas_dataset_info.py

```python
# Informações adicionais úteis
print("3. INFORMAÇÕES COMPLEMENTARES:")
print("-" * 50)
print(f"Dimensões do dataset: {df.shape[0]} linhas x {df.shape[1]} colunas")
print(f"Nomes das colunas: {list(df.columns)}")
print(f"Índice: de {df.index.min()} até {df.index.max()}")

# Verificar duplicatas
duplicatas = df.duplicated().sum()
print(f"Linhas duplicadas: {duplicatas}")

# Estatísticas básicas para colunas numéricas
print(f"\n Resumo estatístico das colunas numéricas:")
print(df.describe())

print("\n" + "-" * 70)
print("RESUMO DA ANÁLISE:")
print("-" * 50)
print(f"Dataset com {total_linhas} registros e {total_colunas} variáveis")
print(f"{total_nulos} valores ausentes identificados")
print(f"Memória total: {memoria_total/1024:.2f} KB")
print(f"Principais tipos de dados: numéricos e texto")
if duplicatas > 0:
    print(f"⚠️  Atenção: {duplicatas} linha(s) duplicada(s) encontrada(s)")
else:
    print(f"✅ Nenhuma linha duplicada encontrada")
```

Saída Parte 01:

```
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 Parte 02:

```
----------------------------------------------------------------
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 Parte 03:

```
--------------------------------------------------------------------------------
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: 1.25 KB
Principais tipos de dados: numéricos e texto
✅ Nenhuma linha duplicada encontrada
```