



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:

Missão Prática | Nível 3 | Mundo 5

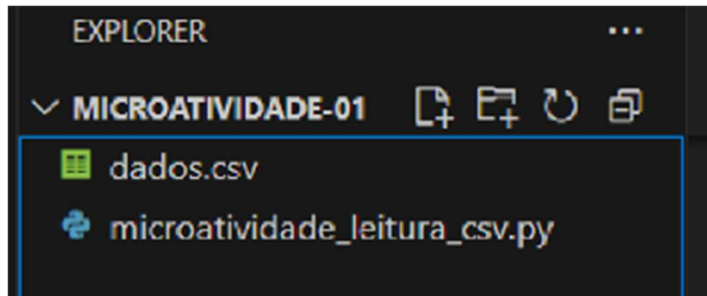
Título da prática: DGT2823 - Tecnologias para desenv. de soluções de big data

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:

```
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")
```

Parte 02:

```

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")

```

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 por
• 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', '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

Microatividades 02:

Código Parte 01:

```

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 Parte 02:

```

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())

```

Código Parte 03:

```

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())

```


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-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 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-existent

--- 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 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 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 > ...
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 Parte 02:

```

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 Parte 03:

```

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"Nomens 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 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:

