

TRABALHO PRÁTICO

DGT2823 - Tecnologias para Desenvolvimento de Soluções de Big Data

UNIVERSIDADE: Estácio

CURSO: Desenvolvimento Full Stack

DISCIPLINA: DGT2823 - Tecnologias para Desenvolvimento de Soluções de Big Data

ALUNO: Alex Barroso Paz

MATRÍCULA: 2023 0615 1781

PROFESSOR/TUTOR: Jhonatan Alves

DATA: 12/06/2025

SUMÁRIO

DGT2823 - Tecnologias para Desenvolvimento de Soluções de Big Data	1
1. 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 Dataset Utilizado	5
3.3 Problemas Identificados no Dataset	5
4. MICROATIVIDADES	5
4.1 Microatividade 1: Leitura de CSV	5
4.2 Microatividade 2: Subconjunto de Dados	6
4.3 Microatividade 3: Configuração de Visualização	7
4.5 Microatividade 5: Informações Gerais	9
Trabalho Prático Final - DGT2823	11
Passo 1: Preparação do Dataset	12
Passos 2-4: Leitura do CSV	13
Passo 5: Verificação dos Dados	14
Passo 6: Cópia do Dataset	14
Passo 7: Tratamento de Valores Nulos em 'Calories'	15
Passo 8: Tratamento Inicial da Coluna 'Date'	16
Passo 9: Correção do Primeiro Erro	16
Passo 10: Correção de Formato Inconsistente	16
Passo 11: Conversão Final	16
Passo 12: Remoção de Registros Nulos	16
Passo 13: Verificação Final	17

1. INTRODUÇÃO

Este trabalho prático tem como finalidade demonstrar o conhecimento adquirido na disciplina DGT2823 Tecnologias para Desenvolvimento de Soluções de Big Data, com foco específico na manipulação e limpeza de dados utilizando a biblioteca Pandas da linguagem Python.

O trabalho foi desenvolvido no ambiente Google Colab, uma plataforma baseada em nuvem que oferece notebooks Jupyter gratuitos, permitindo a execução de código Python de forma interativa e colaborativa.

O dataset utilizado contém informações sobre exercícios físicos, incluindo duração, data, pulso, pulso máximo e calorias queimadas. Este conjunto de dados apresenta propositalmente inconsistências e valores nulos, representando um cenário realista de dados que necessitam de tratamento antes de serem utilizados em análises.

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

- **Plataforma:** Google Colab
- **Linguagem:** Python 3.x
- **Biblioteca Principal:** Pandas
- **Formato de Dados:** CSV (Comma-Separated Values)

3.2 Dataset Utilizado

O dataset contém 32 registros iniciais com as seguintes colunas:

- **ID:** Identificador único do exercício
- **Duration:** Duração do exercício em minutos
- **Date:** Data do exercício
- **Pulse:** Frequência cardíaca durante o exercício
- **Maxpulse:** Frequência cardíaca máxima atingida
- **Calories:** Calorias queimadas

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.

Código Implementado:

```
python import  
  
pandas as pd  
  
df_original =  
pd.read_csv('dados_exercicio.csv',  
sep=';',  
engine='python',  
encoding='utf-8')
```

Resultado:

Screenshot mostrando o dataset carregado

```
1. CRIANDO O DATASET...
✓ Dataset criado com sucesso!
Dataset carregado:
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
```

Análise: A leitura foi realizada com sucesso, especificando o separador de colunas (;), a engine Python para maior flexibilidade e a codificação UTF-8.

4.2 Microatividade 2: Subconjunto de Dados

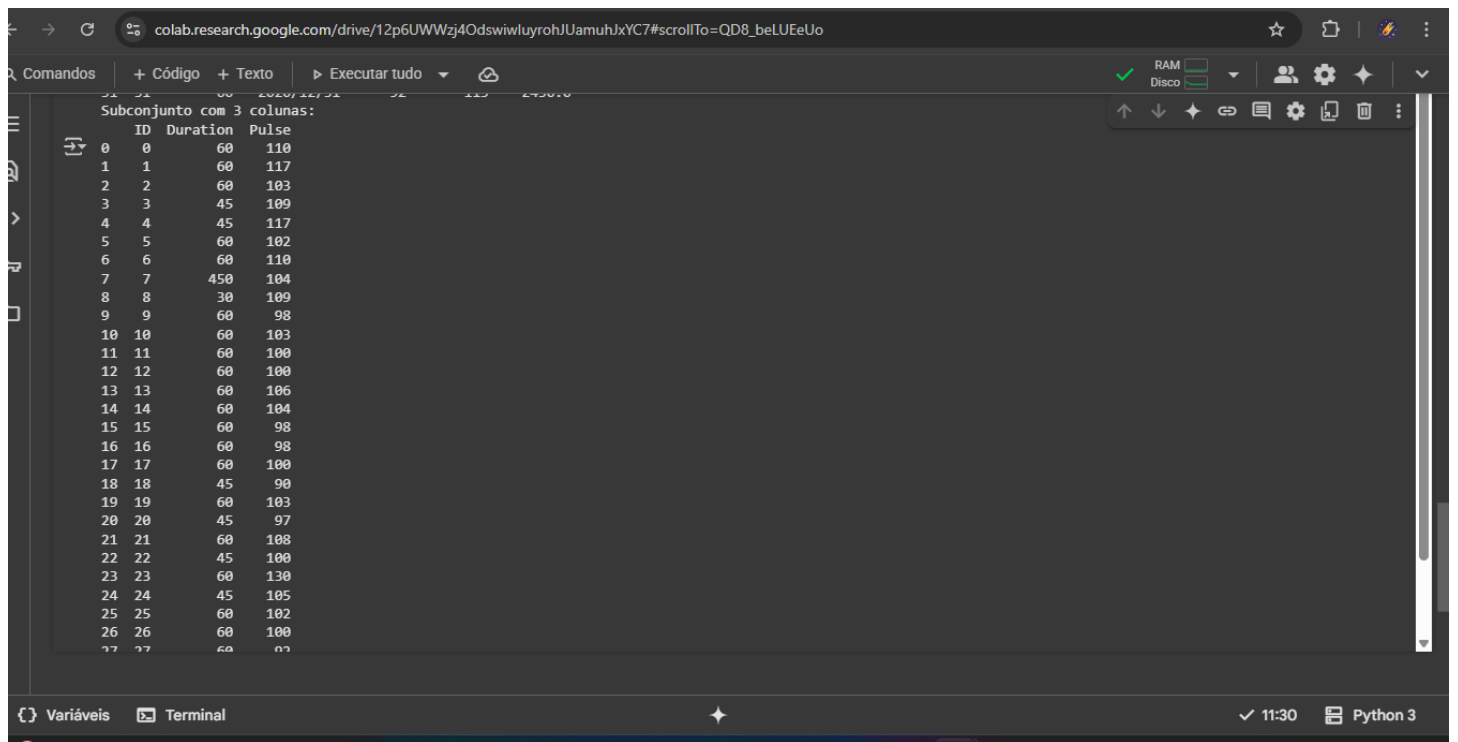
Objetivo: Criar um subconjunto contendo apenas 3 colunas do dataset original.

Código Implementado:

```
python df_subconjunto = df_original[['ID', 'Duration',
                                     'Pulse']]
```

Resultado:

Screenshot do subconjunto com 3 colunas



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:

```
python
pd.set_option('display.max_rows',
9999) print(df_original.to_string())
```

Resultado:

Screenshot mostrando todas as linhas sem truncamento

colab.research.google.com/drive/12pUWWZj40dswiwluyrohJUamuhJxYC7#scrollTo=QD8_beUEeUo

Comandos + Código + Texto ▶ Executar tudo

RAM
Disco

	Dataset	original	com	configuração de	max rows:	
	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

✓ 11:34 Python 3

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:

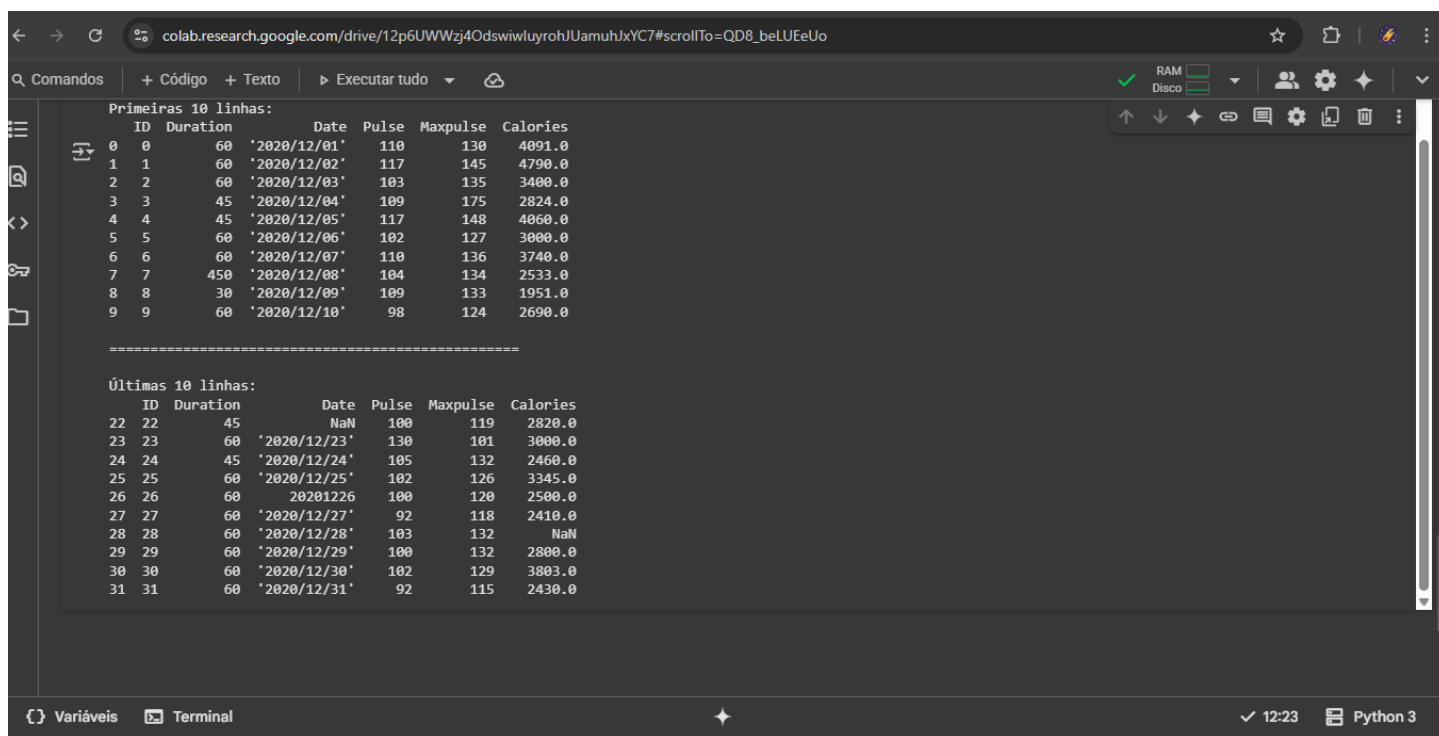
```
python print("Primeiras 10
linhas:")
print(df_original.head(10))

print("Últimas 10 linhas:")
print(df_original.tail(10))
```

Resultado:

Screenshot das primeiras 10 linhas

Screenshot das últimas 10 linhas



```
colab.research.google.com/drive/12p6UWWzj4OdswiwluyrohJUamuhJxYC7#scrollTo=QD8_beLUEeUo
```

Comandos + Código + Texto ▶ Executar tudo

Primeiras 10 linhas:

ID	Duration	Date	Pulse	Maxpulse	Calories
0	0	'2020/12/01'	110	130	4091.0
1	1	'2020/12/02'	117	145	4790.0
2	2	'2020/12/03'	103	135	3400.0
3	3	'2020/12/04'	109	175	2824.0
4	4	'2020/12/05'	117	148	4960.0
5	5	'2020/12/06'	102	127	3000.0
6	6	'2020/12/07'	110	136	3740.0
7	7	'2020/12/08'	104	134	2533.0
8	8	'2020/12/09'	109	133	1951.0
9	9	'2020/12/10'	98	124	2690.0

Últimas 10 linhas:

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

RAM Disco

Varáveis Terminal

12:23 Python 3

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:

```
python print("Informações  
gerais:")  
print(df_original.info())  
  
print("Valores nulos:")  
print(df_original.isnull().sum())  
  
print("Tipos de dados:")  
print(df_original.dtypes)
```

Resultado:

Screenshot das informações gerais

```
colab.research.google.com/drive/12p6UWWzj4OdswiwluyrohJUamuhJxYC7#scrollTo=QD8_beLUeUo

Comandos + Código + Texto ▶ Executar tudo

Informações gerais do dataset:
<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
None

=====

ANÁLISE DETALHADA:
Total de linhas: 32
Total de colunas: 6
Valores nulos por coluna:
ID          0
Duration    0
Date        1
Pulse       0
Maxpulse    0
Calories    2
dtype: int64

Tipos de dados:
ID          int64
```

PRINT - Screenshot dos valores nulos

```
colab.research.google.com/drive/12p6UWWzj4OdswiwluyrohJUamuhJxYC7#scrollTo=QD8_beLUeUo

Comandos + Código + Texto ▶ Executar tudo

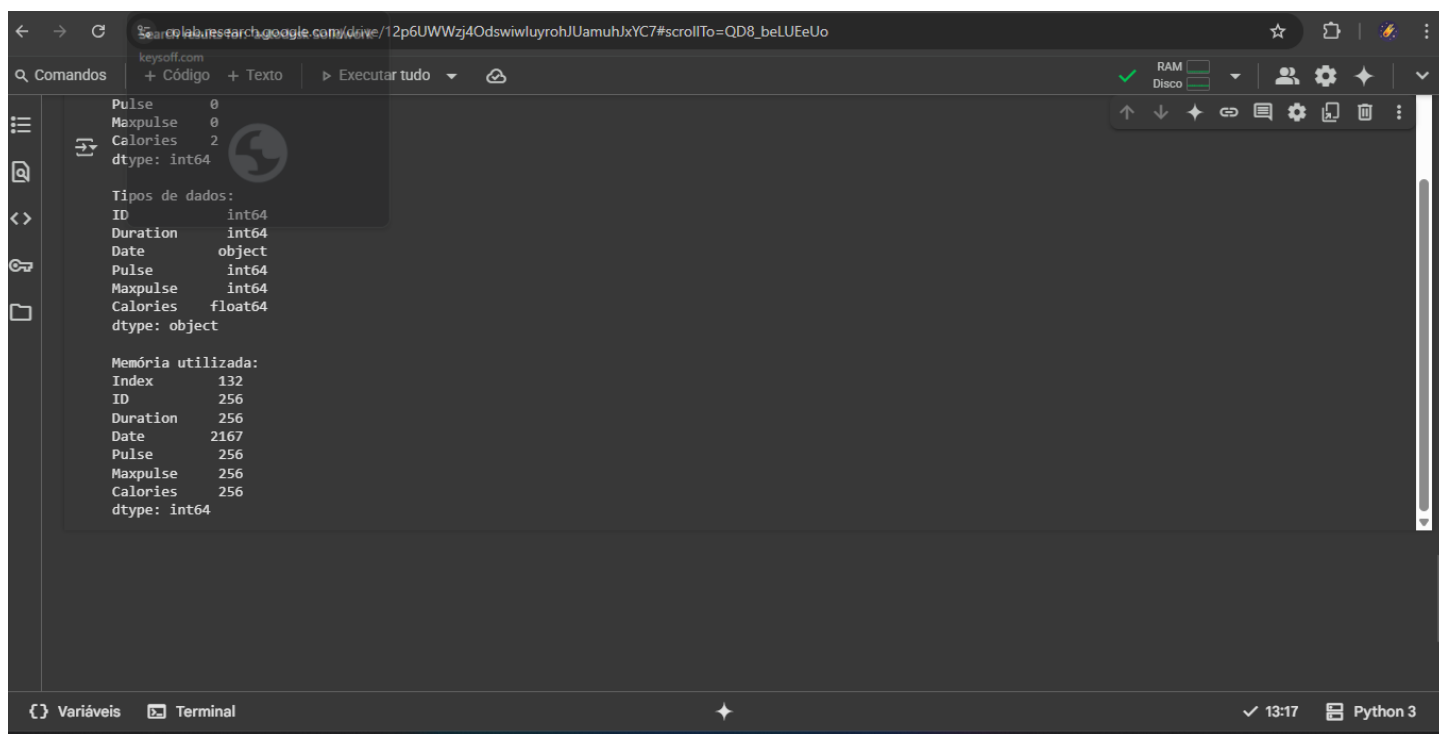
Informações gerais do dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
---  -
0    ID          32 non-null    int64
1    Duration    32 non-null    int64
2    Date        31 non-null    object
3    Pulse       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
None

=====

ANÁLISE DETALHADA:
Total de linhas: 32
Total de colunas: 6
Valores nulos por coluna:
ID          0
Duration    0
Date        1
Pulse       0
Maxpulse    0
Calories    2
dtype: int64

Tipos de dados:
ID          int64
```

PRINT - Screenshot dos tipos de dados



Análise:

- **Total de linhas:** 32
 - **Total de colunas:** 6
 - **Valores nulos identificados:** 3 registros (Calories: 2, Date: 1)
 - **Tipos de dados:** Majoritariamente object, necessitando conversões
-

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.

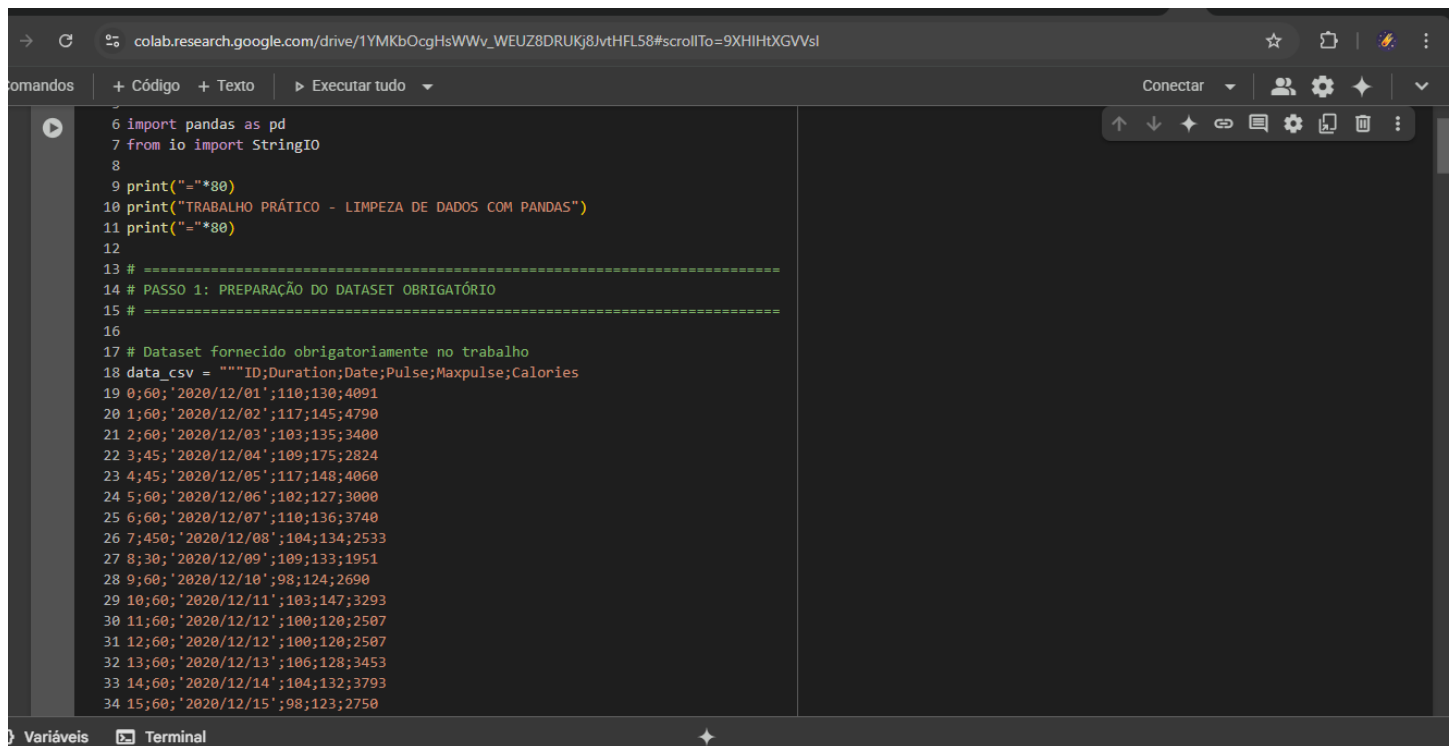
Passo 1: Preparação do Dataset

Criação do dataset obrigatório e salvamento em arquivo CSV.

```
data_csv = """ID;Duration;Date;Pulse;Maxpulse;Calories
... (demais linhas do CSV) ...
"""

with open('dados_exercicio.csv', 'w') as f:
    f.write(data_csv)
```

PRINT - Screenshot Criação do dataset



```
6 import pandas as pd
7 from io import StringIO
8
9 print("="*80)
10 print("TRABALHO PRÁTICO - LIMPEZA DE DADOS COM PANDAS")
11 print("="*80)
12
13 # =====
14 # PASSO 1: PREPARAÇÃO DO DATASET OBRIGATÓRIO
15 # =====
16
17 # Dataset fornecido obrigatoriamente no trabalho
18 data_csv = """ID;Duration;Date;Pulse;Maxpulse;Calories
19 0;60;'2020/12/01';110;130;4091
20 1;60;'2020/12/02';117;145;4790
21 2;60;'2020/12/03';103;135;3400
22 3;45;'2020/12/04';109;175;2824
23 4;45;'2020/12/05';117;148;4060
24 5;60;'2020/12/06';102;127;3000
25 6;60;'2020/12/07';110;136;3740
26 7;450;'2020/12/08';104;134;2533
27 8;30;'2020/12/09';109;133;1951
28 9;60;'2020/12/10';98;124;2690
29 10;60;'2020/12/11';103;147;3293
30 11;60;'2020/12/12';100;120;2507
31 12;60;'2020/12/12';100;120;2507
32 13;60;'2020/12/13';106;128;3453
33 14;60;'2020/12/14';104;132;3793
34 15;60;'2020/12/15';98;123;2750
```

PRINT - Screenshot Criação do dataset

```
Comandos + Código + Texto ▶ Executar tudo
28 9;60; '2020/12/10';98;124;2698
29 10;60; '2020/12/11';103;147;3293
30 11;60; '2020/12/12';100;120;2507
31 12;60; '2020/12/12';100;120;2507
32 13;60; '2020/12/13';106;128;3453
33 14;60; '2020/12/14';104;132;3793
34 15;60; '2020/12/15';98;123;2750
35 16;60; '2020/12/16';98;120;2152
36 17;60; '2020/12/17';100;120;3000
37 18;45; '2020/12/18';90;112;NaN
38 19;60; '2020/12/19';103;123;3230
39 20;45; '2020/12/20';97;125;2430
40 21;60; '2020/12/21';108;131;3642
41 22;45;NaN;100;119;2820
42 23;60; '2020/12/23';130;101;3000
43 24;45; '2020/12/24';105;132;2460
44 25;60; '2020/12/25';102;126;3345
45 26;60;20201226;100;120;2500
46 27;60; '2020/12/27';92;118;2410
47 28;60; '2020/12/28';103;132;NaN
48 29;60; '2020/12/29';100;132;2800
49 30;60; '2020/12/30';102;129;3803
50 31;60; '2020/12/31';92;115;2430""
51
52 # Salvar como arquivo CSV
53 with open('dados_exercicio.csv', 'w') as f:
54     f.write(data_csv)
55
56 print("✅ Dataset obrigatório criado!")
57

Variáveis Terminal
25°C
Parc ensolarado
```

Passos 2-4: Leitura do CSV

Leitura do arquivo CSV e atribuição à variável df_original.

```
df_original = pd.read_csv('dados_exercicio.csv',
                           sep=';',
                           engine='python',
                           encoding='utf-8')
```

PRINT - Screenshot Leitura do arquivo CSV

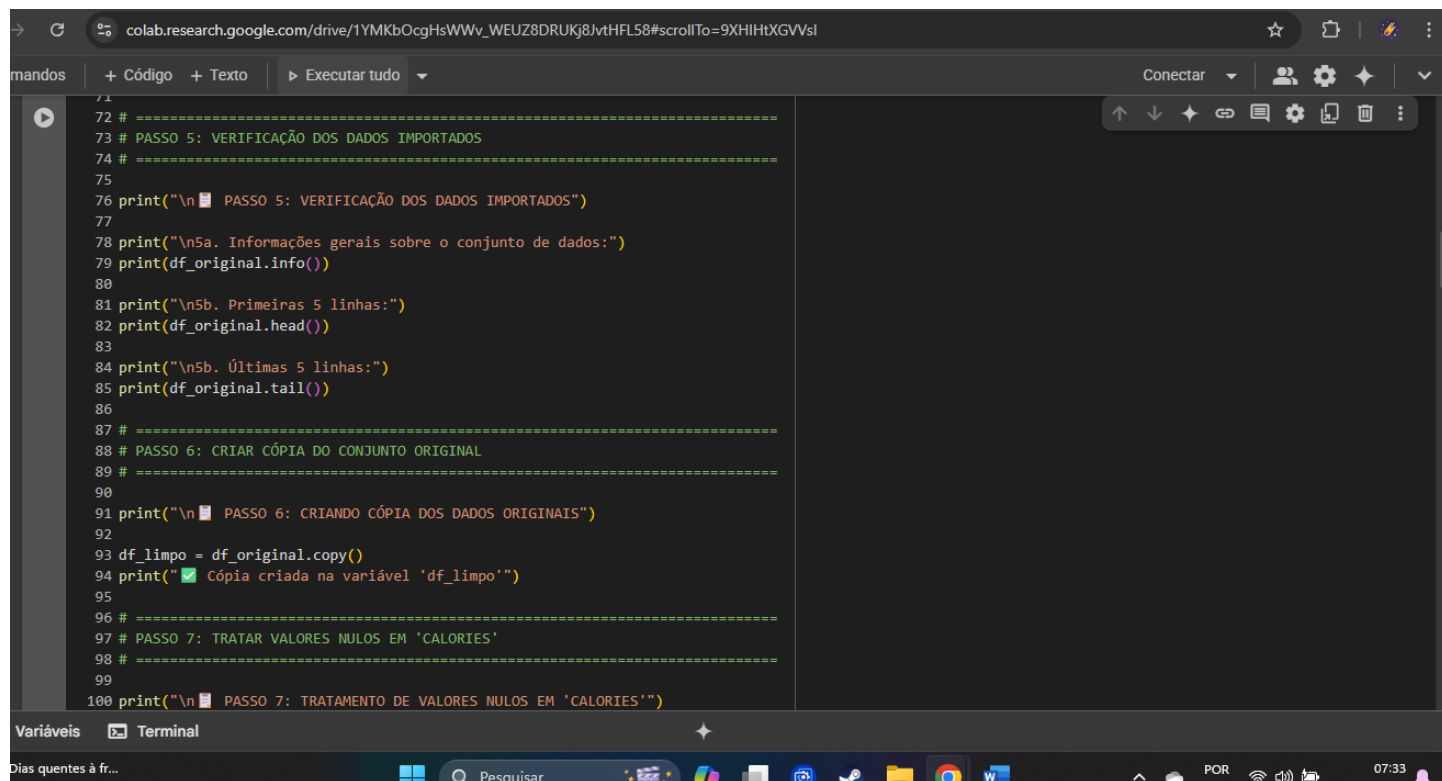
```
colab.research.google.com/drive/1YMKbOcgHsWWv_WEUZ8DRUKj8JvtHFL58#scrollTo=9XHIHtXGVVsl
Comandos + Código + Texto ▶ Executar tudo
58 # =====
59 # PASSOS 2-4: LEITURA DO CSV E ATRIBUIÇÃO A VARIÁVEL
60 # =====
61
62 print("\n 📄 PASSOS 2-4: LEITURA DO ARQUIVO CSV")
63
64 # Lendo o CSV com os parâmetros necessários
65 df_original = pd.read_csv('dados_exercicio.csv',
66                           sep=';', # Separador de colunas
67                           engine='python', # Engine de processamento
68                           encoding='utf-8') # Codificação
69
70 print("✅ Dados lidos e atribuídos à variável 'df_original'")
71
72 # =====
73 # PASSO 5: VERIFICAÇÃO DOS DADOS IMPORTADOS
74 # =====
75
76 print("\n 📄 PASSO 5: VERIFICAÇÃO DOS DADOS IMPORTADOS")
77
78 print("\n5a. Informações gerais sobre o conjunto de dados:")
79 print(df_original.info())
80
81 print("\n5b. Primeiras 5 linhas:")
82 print(df_original.head())
83
84 print("\n5b. Últimas 5 linhas:")
85 print(df_original.tail())
86
87 # =====
```

Passo 5: Verificação dos Dados

Exibição das informações gerais e primeiras/últimas linhas.

```
print(df_original.info())
print(df_original.head())
print(df_original.tail())
```

PRINT - Screenshot Verificação de dados

A screenshot of a Google Colab notebook interface. The browser address bar shows a Google Drive link. The notebook has tabs for 'Comandos', '+ Código', and '+ Texto'. The 'Comandos' tab is active, showing a list of commands. The code cell contains Python code for data verification, including comments in Portuguese and print statements for df.info(), df.head(), and df.tail(). The code is numbered from 72 to 100. The bottom of the notebook shows a 'Variáveis' tab and a 'Terminal' tab. The Windows taskbar is visible at the bottom of the screen.

```
72 # =====
73 # PASSO 5: VERIFICAÇÃO DOS DADOS IMPORTADOS
74 # =====
75
76 print("\n 📄 PASSO 5: VERIFICAÇÃO DOS DADOS IMPORTADOS")
77
78 print("\n5a. Informações gerais sobre o conjunto de dados:")
79 print(df_original.info())
80
81 print("\n5b. Primeiras 5 linhas:")
82 print(df_original.head())
83
84 print("\n5b. Últimas 5 linhas:")
85 print(df_original.tail())
86
87 # =====
88 # PASSO 6: CRIAR CÓPIA DO CONJUNTO ORIGINAL
89 # =====
90
91 print("\n 📄 PASSO 6: CRIANDO CÓPIA DOS DADOS ORIGINAIS")
92
93 df_limpo = df_original.copy()
94 print("✅ Cópia criada na variável 'df_limpo'")
95
96 # =====
97 # PASSO 7: TRATAR VALORES NULOS EM 'CALORIES'
98 # =====
99
100 print("\n 📄 PASSO 7: TRATAMENTO DE VALORES NULOS EM 'CALORIES'")
```

Passo 6: Cópia do Dataset

Criação de uma cópia dos dados para futuras alterações.

```
df_limpo = df_original.copy()
```

PRINT - Screenshot Cópia do Dataset

```
colab.research.google.com/drive/1YMKbOcgHsWWv_WEUZ8DRUKj8/vtHFL58#scrollTo=9XHtXGVVsl

Comandos + Código + Texto ▶ Executar tudo

87 # =====
88 # PASSO 6: CRIAR CÓPIA DO CONJUNTO ORIGINAL
89 # =====
90
91 print("\n 📄 PASSO 6: CRIANDO CÓPIA DOS DADOS ORIGINAIS")
92
93 df_limpo = df_original.copy()
94 print("✅ Cópia criada na variável 'df_limpo'")
95
96 # =====
97 # PASSO 7: TRATAR VALORES NULOS EM 'CALORIES'
98 # =====
99
100 print("\n 📄 PASSO 7: TRATAMENTO DE VALORES NULOS EM 'CALORIES'")
101
102 print("7a. Substituindo valores nulos da coluna 'Calories' por 0:")
103 df_limpo['Calories'] = df_limpo['Calories'].fillna(0)
104
105 print("7b. Verificando se a mudança foi aplicada:")
106 print(df_limpo)
107
108 # =====
109 # PASSO 8: TRATAR VALORES NULOS EM 'DATE'
110 # =====
111
112 print("\n 📄 PASSO 8: TRATAMENTO INICIAL DE VALORES NULOS EM 'DATE'")
113
114 print("8a. Substituindo valores nulos da coluna 'Date' por '1900/01/01':")
115 df_limpo['Date'] = df_limpo['Date'].fillna('1900/01/01')
116
```

Passo 7: Tratamento de Valores Nulos em 'Calories'

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

```
df_limpo['Calories'] = df_limpo['Calories'].fillna(0)
```

PRINT - Screenshot Valores nulos em Calories

```
colab.research.google.com/drive/1YMKbOcgHsWWv_WEUZ8DRUKj8/vtHFL58#scrollTo=9XHtXGVVsl

Comandos + Código + Texto ▶ Executar tudo

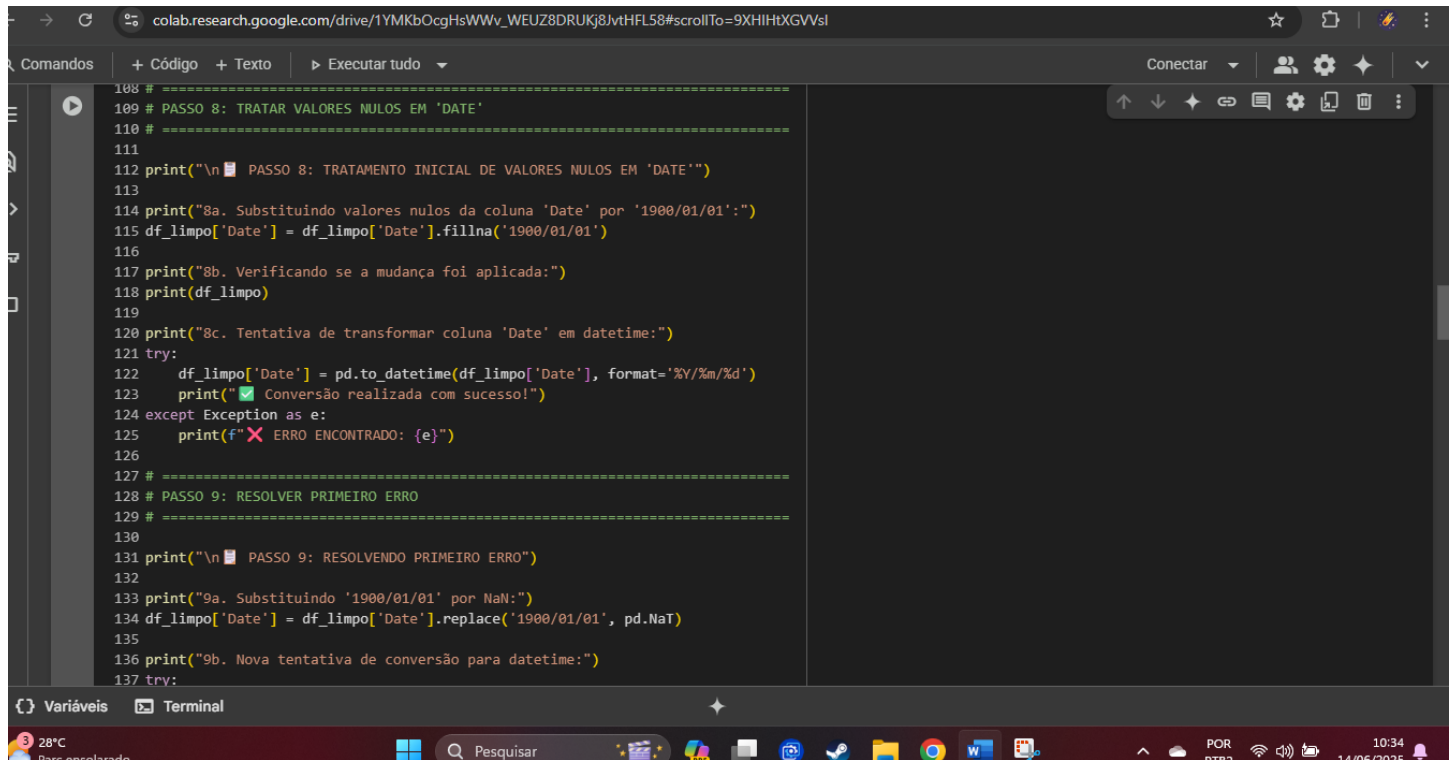
87 # =====
88 # PASSO 6: CRIAR CÓPIA DO CONJUNTO ORIGINAL
89 # =====
90
91 print("\n 📄 PASSO 6: CRIANDO CÓPIA DOS DADOS ORIGINAIS")
92
93 df_limpo = df_original.copy()
94 print("✅ Cópia criada na variável 'df_limpo'")
95
96 # =====
97 # PASSO 7: TRATAR VALORES NULOS EM 'CALORIES'
98 # =====
99
100 print("\n 📄 PASSO 7: TRATAMENTO DE VALORES NULOS EM 'CALORIES'")
101
102 print("7a. Substituindo valores nulos da coluna 'Calories' por 0:")
103 df_limpo['Calories'] = df_limpo['Calories'].fillna(0)
104
105 print("7b. Verificando se a mudança foi aplicada:")
106 print(df_limpo)
107
108 # =====
109 # PASSO 8: TRATAR VALORES NULOS EM 'DATE'
110 # =====
111
112 print("\n 📄 PASSO 8: TRATAMENTO INICIAL DE VALORES NULOS EM 'DATE'")
113
114 print("8a. Substituindo valores nulos da coluna 'Date' por '1900/01/01':")
115 df_limpo['Date'] = df_limpo['Date'].fillna('1900/01/01')
116
```

Passo 8: Tratamento Inicial da Coluna 'Date'

Substituição de nulos por string temporária e tentativa de conversão.

```
df_limpo['Date'] = df_limpo['Date'].fillna('1900/01/01')
df_limpo['Date'] = pd.to_datetime(df_limpo['Date'], format='%Y/%m/%d')
```

PRINT - Screenshot Inicial da coluna Date



```
108 # =====
109 # PASSO 8: TRATAR VALORES NULOS EM 'DATE'
110 # =====
111
112 print("\n 📅 PASSO 8: TRATAMENTO INICIAL DE VALORES NULOS EM 'DATE'")
113
114 print("8a. Substituindo valores nulos da coluna 'Date' por '1900/01/01':")
115 df_limpo['Date'] = df_limpo['Date'].fillna('1900/01/01')
116
117 print("8b. Verificando se a mudança foi aplicada:")
118 print(df_limpo)
119
120 print("8c. Tentativa de transformar coluna 'Date' em datetime:")
121 try:
122     df_limpo['Date'] = pd.to_datetime(df_limpo['Date'], format='%Y/%m/%d')
123     print("✅ Conversão realizada com sucesso!")
124 except Exception as e:
125     print(f"❌ ERRO ENCONTRADO: {e}")
126
127 # =====
128 # PASSO 9: RESOLVER PRIMEIRO ERRO
129 # =====
130
131 print("\n 📅 PASSO 9: RESOLVENDO PRIMEIRO ERRO")
132
133 print("9a. Substituindo '1900/01/01' por NaT:")
134 df_limpo['Date'] = df_limpo['Date'].replace('1900/01/01', pd.NaT)
135
136 print("9b. Nova tentativa de conversão para datetime:")
137 try:
```

Passo 9: Correção do Primeiro Erro

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

```
df_limpo['Date'] = df_limpo['Date'].replace('1900/01/01', pd.NaT)
df_limpo['Date'] = pd.to_datetime(df_limpo['Date'], format='%Y/%m/%d', errors='coerce')
```

Passo 10: Correção de Formato Inconsistente

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

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

Passo 11: Conversão Final

Verificação do tipo da coluna 'Date'.

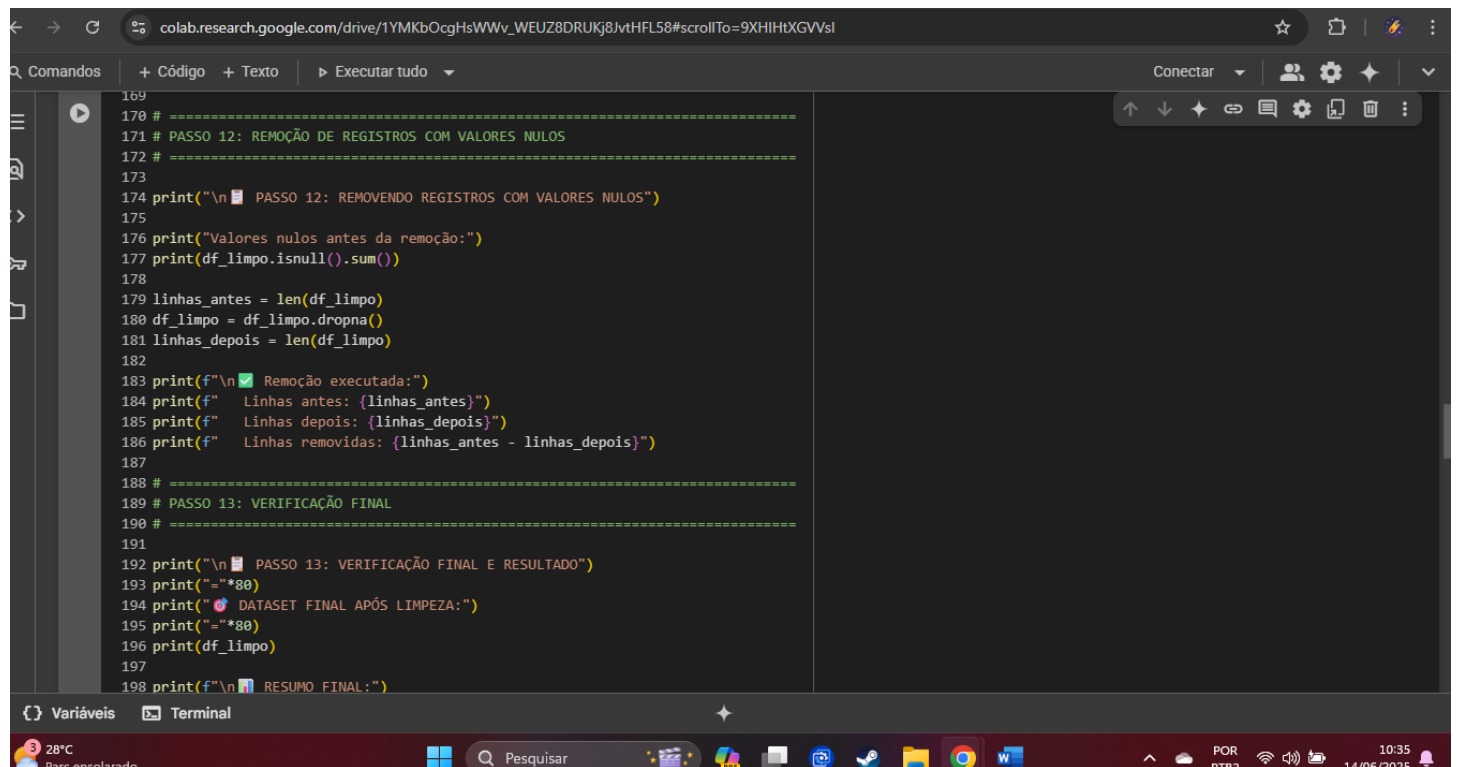
```
print(df_limpo['Date'].dtype)
```

Passo 12: Remoção de Registros Nulos

Remoção de linhas com valores nulos restantes.

```
df_limpo = df_limpo.dropna()
```


PRINT - Screenshot Remoção registro nulos



The screenshot shows a Google Colab notebook with the following code:

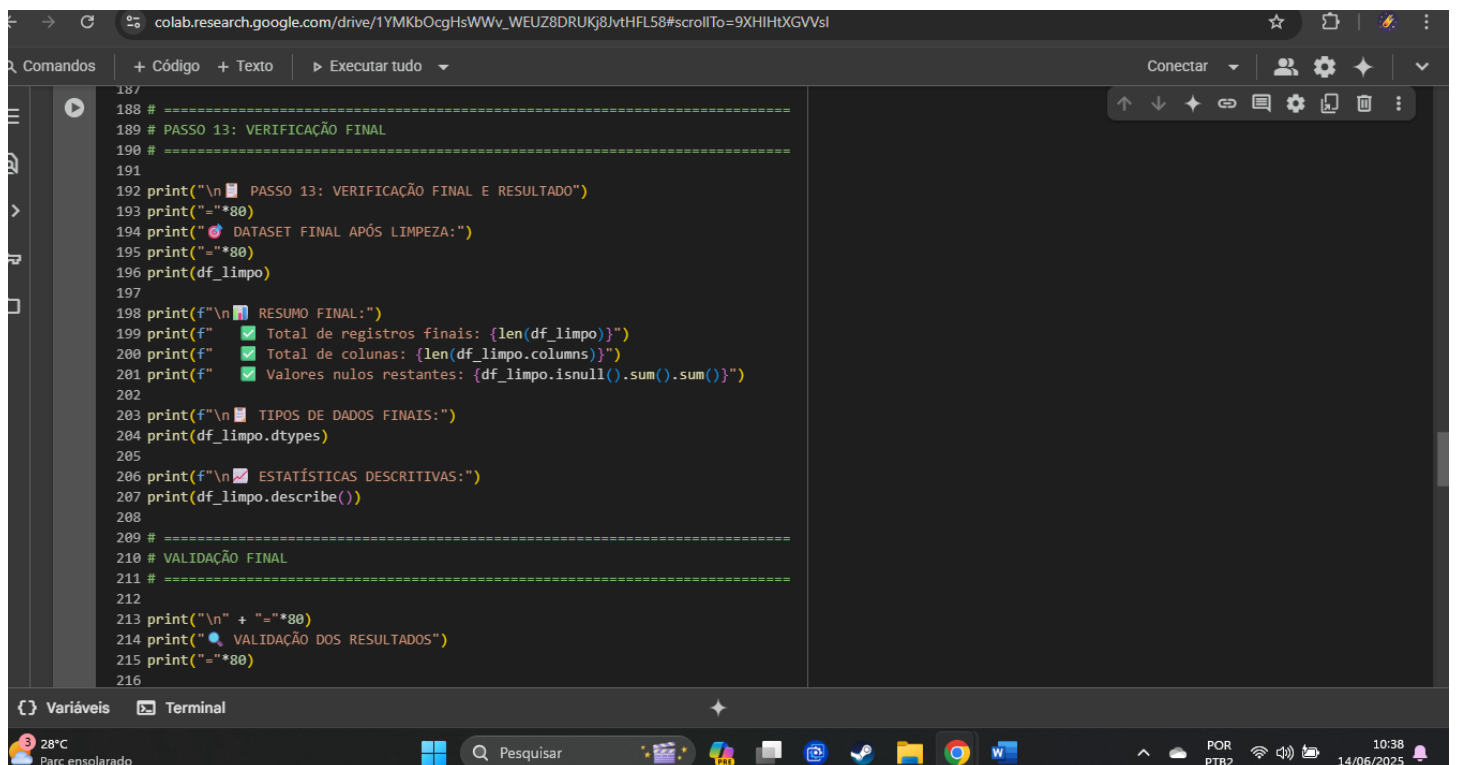
```
169
170 # =====
171 # PASSO 12: REMOÇÃO DE REGISTROS COM VALORES NULOS
172 # =====
173
174 print("\n 📄 PASSO 12: REMOVENDO REGISTROS COM VALORES NULOS")
175
176 print("Valores nulos antes da remoção:")
177 print(df_limpo.isnull().sum())
178
179 linhas_antes = len(df_limpo)
180 df_limpo = df_limpo.dropna()
181 linhas_depois = len(df_limpo)
182
183 print(f"\n ✅ Remoção executada:")
184 print(f"   Linhas antes: {linhas_antes}")
185 print(f"   Linhas depois: {linhas_depois}")
186 print(f"   Linhas removidas: {linhas_antes - linhas_depois}")
187
188 # =====
189 # PASSO 13: VERIFICAÇÃO FINAL
190 # =====
191
192 print("\n 📄 PASSO 13: VERIFICAÇÃO FINAL E RESULTADO")
193 print("="*80)
194 print("🔍 DATASET FINAL APÓS LIMPEZA:")
195 print("="*80)
196 print(df_limpo)
197
198 print(f"\n 📄 RESUMO FINAL:")
```

Passo 13: Verificação Final

Exibição do dataset final limpo e estatísticas descritivas.

```
print(df_limpo.info())
print(df_limpo.describe())
```

PRINT - Screenshot Verificação final



The screenshot shows a Google Colab notebook with the following code:

```
187
188 # =====
189 # PASSO 13: VERIFICAÇÃO FINAL
190 # =====
191
192 print("\n 📄 PASSO 13: VERIFICAÇÃO FINAL E RESULTADO")
193 print("="*80)
194 print("🔍 DATASET FINAL APÓS LIMPEZA:")
195 print("="*80)
196 print(df_limpo)
197
198 print(f"\n 📄 RESUMO FINAL:")
199 print(f"   ✅ Total de registros finais: {len(df_limpo)}")
200 print(f"   ✅ Total de colunas: {len(df_limpo.columns)}")
201 print(f"   ✅ Valores nulos restantes: {df_limpo.isnull().sum().sum()}")
202
203 print(f"\n 📄 TIPOS DE DADOS FINAIS:")
204 print(df_limpo.dtypes)
205
206 print(f"\n 📄 ESTATÍSTICAS DESCRITIVAS:")
207 print(df_limpo.describe())
208
209 # =====
210 # VALIDAÇÃO FINAL
211 # =====
212
213 print("\n" + "="*80)
214 print("🔍 VALIDAÇÃO DOS RESULTADOS")
215 print("="*80)
216
```