

Desafio Data Wrangling e Pipeline

Bootcamp Business Intelligence

Extração e Exploração

##1. Leitura e Exploração Inicial

```
from google.colab import drive
drive.mount('/content/drive')
```

##1. Leia o arquivo CSV e visualize as 5 primeiras linhas do dataset.

```
import pandas as pd
dataframe = pd.read_csv("/content/vendas.csv")
print(dataframe.head())
```



##2. Verifique o número de linhas e colunas.

```
contLinha = len(dataframe.axes[0])
contColuna = len(dataframe.axes[1])
print("Número de colunas:"+str(contColuna))
print("Número de linhas:"+str(contLinha))
```



Número de colunas:7
Número de linhas:2000



Exploração

▶ ##3. Descubra o tipo de dados (dtype) de cada coluna.

```
tipo_colunas = dataframe.dtypes  
print(tipo_colunas)
```

```
⇒ id_venda          int64  
   data_venda       object  
   cliente          object  
   produto          object  
   quantidade       object  
   preco_unitario   float64  
   categoria        object  
   dtype: object
```

▶ ##4. Conte os valores ausentes existentes em cada coluna.

```
valores_ausentes = dataframe.isnull().sum()
```

Exibir o resultado

```
print("Valores ausentes por coluna:\n")  
print(valores_ausentes)
```

⇒ Valores ausentes por coluna:

```
id_venda          0  
data_venda        0  
cliente           335  
produto           0  
quantidade        0  
preco_unitario    0  
categoria         52  
dtype: int64
```



Filtragem e ordenação

▶ #1. Filtre apenas as linhas onde o preço unitário é maior que 100.

```
preco_maior_que_100 = dataframe['preco_unitario'] > 100
print(dataframe[preco_maior_que_100])
```

	id_venda	data_venda	cliente	produto	quantidade	\
0	1	2025/06/01	Fernanda Lima	iPhone 13	5	
2	3	04-04-2025	Maria Silva	Notebook Dell	2	
4	5	15-09-2025	Fernanda Lima	Mouse Gamer	3	
7	8	2025/01/22	Pedro Costa	Mochila	1	
9	10	2025-02-03	Pedro Costa	Notebook Dell	2	
...	
1987	1988	10-09-2025	Pedro Costa	iPhone 13	5	
1991	1992	2025-04-05	Pedro Costa	Impressora HP	4	
1994	1995	2025/04/19	Maria Silva	Monitor LG	2	
1995	1996	2025-07-20	Pedro Costa	iPhone 13	3	
1999	2000	2025/12/27	Ana Pereira	iPhone 13	três	

▶ #2. Ordene o dataset pelo valor do preço em ordem decrescente.

```
# Ordenar pelo preço unitário em ordem decrescente
df_ordenado = dataframe.sort_values(by='preco_unitario', ascending=False)

# Exibir as 5 primeiras linhas para verificação
print(df_ordenado.head())
```

	id_venda	data_venda	cliente	produto	quantidade	\
25	26	2025-06-07	NaN	iPhone 13	4	
0	1	2025/06/01	Fernanda Lima	iPhone 13	5	
1982	1983	01-04-2025	Ana Pereira	iPhone 13	4	
24	25	2025-07-23	Pedro Costa	iPhone 13	3	
19	20	26-09-2025	Maria Silva	iPhone 13	3	

	preco_unitario	categoria
25	5500.0	Eletrônicos
0	5500.0	Eletrônicos
1982	5500.0	Eletrônicos
24	5500.0	Eletrônicos
19	5500.0	Eletrônicos



Transformação

```
▶ # Padronizar a coluna de data (YYYY-MM-DD).
dataframe['data_venda'] = (
    dataframe['data_venda']
    .astype(str)
    .str.strip()
    .str.replace(r'[/\.]', '-', regex=True)
)

datas = pd.to_datetime(dataframe['data_venda'], errors='coerce', format='%Y-%m-%d')

mask_nat = datas.isna()
datas[mask_nat] = pd.to_datetime(dataframe.loc[mask_nat, 'data_venda'], errors='coerce', dayfirst=True)

dataframe['data_venda'] = datas.dt.strftime('%Y-%m-%d')

print(dataframe['data_venda'].head())
```


```
↩ 0    2025-06-01
   1    2025-05-02
   2    2025-04-04
   3    2025-06-21
   4    2025-09-15
   Name: data_venda, dtype: object
```



Transformação

```
# Substituir valores nulos por "Não informado".
newDataframe = dataframe.fillna('Não Informado')
```

```
print(newDataframe.head())
```



	id_venda	data_venda	cliente	produto	quantidade	\
0	1	2025-06-01	Fernanda Lima	iPhone 13	5	
1	2	2025-05-02	João Souza	Caderno	1	
2	3	2025-04-04	Maria Silva	Notebook Dell	2	
3	4	2025-06-21	Fernanda Lima	Caderno	1	
4	5	2025-09-15	Fernanda Lima	Mouse Gamer	3	

	preco_unitario	categoria
0	5500.0	Eletrônicos
1	25.0	Papelaria
2	3500.0	Eletrônicos
3	25.0	Papelaria
4	200.0	Eletrônicos

```
# Corrigir quantidade para ser sempre um número inteiro.
dataframe['quantidade'].unique()
```

```
dataframe['quantidade'] = dataframe['quantidade'].replace('três', '3')
```

```
dataframe['quantidade'] = pd.to_numeric(dataframe['quantidade'])
dataframe.dtypes
```



	0
id_venda	int64
data_venda	object
cliente	object
produto	object
quantidade	int64
preco_unitario	float64
categoria	object

dtype: object



Transformação

▶ # Remover ou corrigir preços negativos.
if 'preco_unitario' in dataframe.columns:
 dataframe = dataframe[dataframe['preco_unitario'] >= 0]

▶ # Criar uma nova coluna valor_total = quantidade * preco_unitario.
if 'quantidade' in dataframe.columns and 'preco_unitario' in dataframe.columns:
 dataframe['valor_total'] = dataframe['quantidade'] * dataframe['preco_unitario']



Carga

```
# Criar um banco SQLite (arquivo vendas.db).  
conexao = sql.connect('vendas.db')  
print("Banco de dados 'vendas.db' criado/conectado com sucesso. \n\n")
```

```
# Salvar os dados tratados em uma tabela tb_vendas.  
dataframe.to_sql("tb_vendas", conexao, if_exists="replace", index=False)  
print("Dados carregados na tabela 'tb_vendas' do banco de dados 'vendas.db'.\n\n")
```



Carga

Desafio Extra

```
# Criar uma segunda tabela tb_clientes contendo apenas clientes únicos.
df_clientes = dataframe[["cliente"]].drop_duplicates().reset_index(drop=True)
df_clientes.to_sql("tb_clientes", conexao, if_exists="replace", index=False)
print("Tabela 'tb_clientes' criada com clientes únicos no banco de dados 'vendas.db'.\n\n")

# Relacionar tb_vendas com tb_clientes via chave estrangeira.
dataframe = dataframe.merge(df_clientes, on="cliente", how="left")
print("Relacionamento lógico entre tb_vendas e tb_clientes criado (coluna: cliente). \n\n")
```



Carga

Desafio Extra

Escrever uma consulta SQL que mostre: total de vendas por categoria.

```
query = """
```

```
    SELECT
```

```
        categoria,
```

```
        SUM(valor_total) AS total_vendas
```

```
    FROM tb_vendas
```

```
    GROUP BY categoria;
```

```
"""
```

```
df_resultado = pd.read_sql_query(query, conexao)
```

```
print("Resultado da consulta SQL - Total de vendas por categoria:\n")
```

```
print(df_resultado.to_string())
```



Carga

```
# Adicional gerar um relatório em arquivo texto com um resumo do banco de dados.
print("\n\nGerando relatório do banco de dados...")
print("Relatório gerado com sucesso: relatorio_banco_de_dados.txt\n")
```

```
arquivo_saida="relatorio_banco_de_dados.txt"
tabelas = pd.read_sql_query(
    "SELECT name FROM sqlite_master WHERE type='table';", conexao
)
```

```
with open(arquivo_saida, "w", encoding="utf-8") as f:
```

```
    f.write(f"📄 RELATÓRIO DO BANCO: vendas.db\n")
    f.write("="*60 + "\n\n")
    f.write("Tabelas encontradas:\n")
    f.write(str(tabelas) + "\n\n")
```

```
# Para cada tabela, exibir resumo e 15 primeiros registros
```

```
for tabela in tabelas['name']:
```

```
    f.write(f"📁 Tabela: {tabela}\n")
    f.write("-"*60 + "\n")
```

```
# Esquema (colunas e tipos)
```

```
schema = pd.read_sql_query(f"PRAGMA table_info({tabela});", conexao)
f.write("📋 Estrutura:\n")
f.write(schema.to_string(index=False) + "\n\n")
```

```
# Primeiras linhas
```

```
df_preview = pd.read_sql_query(f"SELECT * FROM {tabela} LIMIT 30;", conexao)
f.write("🌀 Primeiros registros:\n")
f.write(df_preview.to_string(index=False) + "\n\n")
```

```
# Contagem de registros
```

```
count = pd.read_sql_query(f"SELECT COUNT(*) AS total FROM {tabela};", conexao)
f.write(f"📊 Total de registros: {count['total'][0]}\n")
f.write("-"*60 + "\n\n")
```

```
conexao.close()
```



Carga

RELATÓRIO DO BANCO: vendas.db

Tabelas encontradas:

```
0  tb_vendas
1  tb_clientes
```

Tabela: tb_vendas

Estrutura:

cid	name	type	notnull	dflt_value	pk
0	id_venda	INTEGER	0	None	0
1	data_venda	TEXT	0	None	0
2	cliente	TEXT	0	None	0
3	produto	TEXT	0	None	0
4	quantidade	INTEGER	0	None	0
5	preco_unitario	REAL	0	None	0
6	categoria	TEXT	0	None	0
7	valor_total	REAL	0	None	0

Primeiros registros:

id_venda	data_venda	cliente	produto	quantidade	preco_unitario	categoria	valor_total
1	2025-06-01	Fernanda Lima	iPhone 13	5	5500.0	Eletrônicos	27500.0
2	2025-05-02	João Souza	Caderno	1	25.0	Papelaria	25.0
3	2025-04-04	Maria Silva	Notebook Dell	2	3500.0	Eletrônicos	7000.0
4	2025-06-21	Fernanda Lima	Caderno	1	25.0	Papelaria	25.0
5	2025-09-15	Fernanda Lima	Mouse Gamer	3	200.0	Eletrônicos	600.0
6	2025-01-22	Fernanda Lima	Caderno	5	25.0	Papelaria	125.0
7	2025-05-22	Fernanda Lima	Caneta Azul	3	2.5	Papelaria	7.5



Organização do Squad





Gestora de Atividades:

- Sorteio sobre quem ficaria responsável por cada atividade;
- Uso do Trello para gestão;
- Acompanhamento através do Discord.



Kanban

Desafio - Construção De Um Pipeline De Dados 000 ▾ IM DM AS AL +3 🔍 ⚡ ☰ ☆ 🔒 + Compartilhar ...

A Fazer

Corrigir a quantidade para ser sempre um número inteiro.
🕒 8 de out. - 19 de out. VA

Relacionar tb_vendas com tb_clientes por meio de uma chave estrangeira.

+ Adicionar um cartão 📄

Fazendo

Conte os valores ausentes existentes em cada coluna.
🕒 8 de out. - 19 de out.

Apresentação

🔔 1 👁 🕒 20 de out. - 23 de out. IM 👤 AS

+ Adicionar um cartão 📄

Concluído

✓ Escrever uma consulta SQL que mostre o total de vendas por categoria

🕒 8 de out. - 19 de out. VA

✓ Salvar os dados tratados em uma tabela chamada tb_vendas

+ Adicionar um cartão 📄

+ Adicionar outra lista



Repositório do Github

The screenshot shows the GitHub repository page for 'Desafio-Data-Wrangling-e-Pipeline' by user 'isabelamenezs'. The repository is public and has 2 stars and 2 forks. The main branch is 'main'. The repository contains a README.md file and a pipeline.py file. The README.md file is titled 'Desafio Data Wrangling & Pipeline de Dados' and describes the project as a data pipeline for cleaning, transforming, and loading sales data into a SQLite database. The pipeline.py file is a Python script for data wrangling.

Repository: Desafio-Data-Wrangling-e-Pipeline (Public)

Branches: main (1 Branch), Tags: 0 Tags

Files:

File	Commit	Time
dados	Create vendas_raw.csv	last week
etl	Update transform.py	1 hour ago
README.md	Update README.md	13 minutes ago
pipeline.py	Update pipeline.py	2 days ago

About: Construir um pipeline de dados em Python para limpar, transformar e carregar informações de vendas em um banco SQLite.

Releases: No releases published. [Create a new release](#)

<https://github.com/isabelamenezs/Desafio-Data-Wrangling-e-Pipeline>



Facilidades e Dificuldades



Obstáculos

- Desistência de 3 integrantes
- Aprendizado de novas plataformas
- Dificuldades para realizar algumas das questões
- Gestão de tempo





Facilidades

- Proatividade do time
- Organização
- Background em tecnologia
- Comunicação



Obrigada! :)