Análise Exploratória de Dados - 2024 🚀

- Projeto: **EDA - Análise Exploratória de Dados de Vendas Online**
- Curso: **Santander Coders 2023 | 2º Semestre**
- Módulo: **Técnicas de Programação I (PY)**
- **ADA Tech** em parceira com **Banco Santander**
- Facilitador: **Thiago**

Aluno: **Anderson Miranda**
ID: **1116003**

1. OBJETIVO:

O objetivo deste projeto é aplicar os conhecimentos adquiridos ao longo da disciplina de Técnicas de Programação I em um contexto prático, relevante e data-driven. Devendo realizar uma análise exploratória de dados de vendas online, utilizando um conjunto de dados real, a fim de extrair insights e entender melhor o fenômeno das vendas e de tudo aquilo que lhe diz respeito (clientes, fornecedores, produtos, pagamentos, etc.), usando apenas as bibliotecas Numpy e Pandas.

2. DATA SOURCE:

Vendas Online da Olist

Este dataset foi generosamente cedido pela Olist, a maior loja de departamentos dos mercados brasileiros. Olist conecta pequenas empresas de todo o Brasil a canais sem complicações e com um único contrato. Esses comerciantes podem vender seus produtos através da Olist Store e enviá-los diretamente aos clientes usando os parceiros logísticos da Olist.

Depois que um cliente compra o produto na Olist Store, um vendedor é notificado para atender ao pedido. Assim que o cliente recebe o produto, ou vence a data estimada de entrega, o cliente recebe por e-mail uma pesquisa de satisfação onde pode dar uma nota da experiência de compra e anotar alguns comentários.

Fonte: Kaggle - Vendas Online da Olist

1. Carregamento Principal

```
import pandas as pd
import numpy as np
```

```
# Define o path do dataset
path_dataset = "https://raw.githubusercontent.com/aluipio/ds_ada_santander_eda/main/dataset/"

# Carrega CSV
df_orders = pd.read_csv(path_dataset + "olist_orders_dataset.csv")
df_items = pd.read_csv(path_dataset + "olist_order_items_dataset.csv")
df_order_pay = pd.read_csv(path_dataset + "olist_order_payments_dataset.csv")
df_products = pd.read_csv(path_dataset + "olist_products_dataset.csv")
df_sellers = pd.read_csv(path_dataset + "olist_sellers_dataset.csv")
df_reviews = pd.read_csv(path_dataset + "olist_order_reviews_dataset.csv")
df_geolocation = pd.read_csv(path_dataset + "olist_geolocation_dataset.csv")
# df_customers = pd.read_csv(path_dataset + "olist_customers_dataset.csv")
# df_category = pd.read_csv(path_dataset + "product_category_name_translation.csv")
```

Depois de analisado os datasets, vamos estruturar um DataFrame com os dados de Ordens, Itens, Pagamentos, Produtos e Vendas, para iniciar as analises.

```
# Mesclar DataFrame central com os periféricos
df_full = df_orders.merge(df_items, on='order_id', how='left')
df_full = df_full.merge(df_order_pay, on='order_id', how='outer', validate='m:m')
df_full = df_full.merge(df_products, on='product_id', how='outer')
df_full = df_full.merge(df_sellers, on='seller_id', how='outer')
```

```
df_full.head()
```

	order_id	customer_id	order_status	o
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
3	128e10d95713541c87cd1a2e48201934	a20e8105f23924cd00833fd87daa0831	delivered	
4	0e7e841ddf8f8f2de2bad69267ecfbcf	26c7ac168e1433912a51b924fbd34d34	delivered	

5 rows × 29 columns

2. **Análise das informações**

```
1 # Dimensão da tabela
2 df_full.shape
```

```
(118434, 29)
```

```
1 # Visualizar informações do dataset
2 df_full.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 118434 entries, 0 to 118433
Data columns (total 29 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   order_id                       118434 non-null  object
 1   customer_id                    118434 non-null  object
 2   order_status                   118434 non-null  object
 3   order_purchase_timestamp       118434 non-null  object
 4   order_approved_at              118258 non-null  object
 5   order_delivered_carrier_date   116360 non-null  object
 6   order_delivered_customer_date  115037 non-null  object
 7   order_estimated_delivery_date  118434 non-null  object
 8   order_item_id                  117604 non-null  float64
 9   product_id                     117604 non-null  object
 10  seller_id                      117604 non-null  object
 11  shipping_limit_date            117604 non-null  object
 12  price                          117604 non-null  float64
 13  freight_value                  117604 non-null  float64
 14  payment_sequential             118431 non-null  float64
 15  payment_type                   118431 non-null  object
 16  payment_installments           118431 non-null  float64
 17  payment_value                  118431 non-null  float64
 18  product_category_name          115906 non-null  object
 19  product_name_lenght            115906 non-null  float64
 20  product_description_lenght     115906 non-null  float64
 21  product_photos_qty             115906 non-null  float64
 22  product_weight_g               117584 non-null  float64
 23  product_length_cm              117584 non-null  float64
 24  product_height_cm              117584 non-null  float64
 25  product_width_cm               117584 non-null  float64
 26  seller_zip_code_prefix         117604 non-null  float64
 27  seller_city                    117604 non-null  object
 28  seller_state                   117604 non-null  object
dtypes: float64(14), object(15)
memory usage: 27.1+ MB
```

```
1 # Visualizando parcela dos dados
2 df_full.head(5)
```

	order_id	customer_id	order_status	o
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
3	128e10d95713541c87cd1a2e48201934	a20e8105f23924cd00833fd87daa0831	delivered	
4	0e7e841ddf8f8f2de2bad69267ecfbcf	26c7ac168e1433912a51b924fbd34d34	delivered	

5 rows × 29 columns

```
1 # Verificar a integridade dos dados
2 df_full.isna().sum() * 100  / df_full.shape[0]
```

```
order_id                       0.000000
customer_id                    0.000000
```

```
order_status                    0.000000
order_purchase_timestamp        0.000000
order_approved_at               0.148606
order_delivered_carrier_date    1.751186
order_delivered_customer_date   2.868264
order_estimated_delivery_date   0.000000
order_item_id                   0.700812
product_id                      0.700812
seller_id                       0.700812
shipping_limit_date             0.700812
price                           0.700812
freight_value                   0.700812
payment_sequential              0.002533
payment_type                    0.002533
payment_installments            0.002533
payment_value                   0.002533
product_category_name           2.134522
product_name_lenght             2.134522
product_description_lenght      2.134522
product_photos_qty              2.134522
product_weight_g                0.717699
product_length_cm               0.717699
product_height_cm               0.717699
product_width_cm                0.717699
seller_zip_code_prefix          0.700812
seller_city                     0.700812
seller_state                    0.700812
dtype: float64
```

Verifica-se que temos cerca de 3% dos dados faltando, vamos estruturar

- vamos remover as linhas com dados ausentes para analisar apenas os dados integros;
- vamos remover as colunas com código de referência ou identificador;

```
# Remoção de dados vazios
df = df_full.dropna()

# Remoção de colunas com código identifiador
df = df.drop(['order_id', 'order_item_id', 'product_id', 'seller_id'], axis=1)
df.shape

print("Total de linhas removidas:", df_full.shape[0] - df.shape[0])
print("Quantidade de dados depois da exclusão:", df.shape)
```

```
Total de linhas removidas: 5044
Quantidade de dados depois da exclusão: (113390, 25)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 113390 entries, 0 to 118433
Data columns (total 25 columns):
 #   Column                         Non-Null Count   Dtype  
---  ------                         --------------   -----  
 0   customer_id                    113390 non-null  object 
 1   order_status                   113390 non-null  object 
 2   order_purchase_timestamp       113390 non-null  object 
 3   order_approved_at              113390 non-null  object 
 4   order_delivered_carrier_date   113390 non-null  object 
 5   order_delivered_customer_date  113390 non-null  object 
 6   order_estimated_delivery_date  113390 non-null  object 
 7   shipping_limit_date            113390 non-null  object 
 8   price                          113390 non-null  float64
 9   freight_value                  113390 non-null  float64
 10  payment_sequential             113390 non-null  float64
 11  payment_type                   113390 non-null  object 
 12  payment_installments           113390 non-null  float64
 13  payment_value                  113390 non-null  float64
 14  product_category_name          113390 non-null  object 
 15  product_name_lenght            113390 non-null  float64
 16  product_description_lenght     113390 non-null  float64
 17  product_photos_qty             113390 non-null  float64
 18  product_weight_g               113390 non-null  float64
 19  product_length_cm              113390 non-null  float64
 20  product_height_cm              113390 non-null  float64
 21  product_width_cm               113390 non-null  float64
 22  seller_zip_code_prefix         113390 non-null  float64
 23  seller_city                    113390 non-null  object 
 24  seller_state                   113390 non-null  object 
dtypes: float64(13), object(12)
memory usage: 22.5+ MB
```

2.1. Analisando os dados objects e categóricos

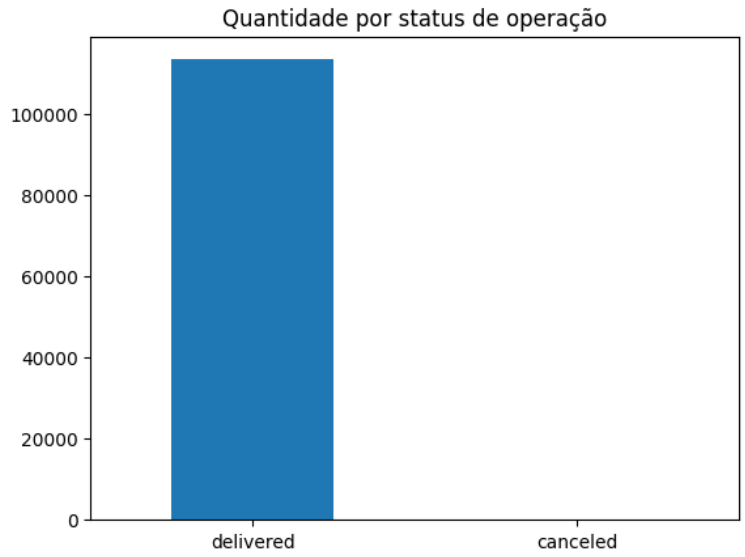
```
1 # Analisando os dados objects e categoricos
2 df.describe(include='object')
```

	customer_id	order_status	order_purchase_timestamp	orde
count	113390	113390	113390	
unique	95128	2	94624	
top	270c23a11d024a44c896d1894b261a83	delivered	2017-08-08 20:26:31	2017
freq	63	113383	63	

```
1 # Vamos analisar a distribuição dos status da ordem
2 df.order_status.value_counts(True)
```

```
delivered    0.999938
canceled     0.000062
Name: order_status, dtype: float64
```

```
1 df.order_status.value_counts().plot.bar(rot=0, title='Quantidade por status de operação');
```




```
1 # Categoria de produtos cancelados
2 mask = df.order_status == 'canceled'
3 df[mask].product_category_name
```

```
45835                      brinquedos
47081                      perfumaria
47124                      perfumaria
47401                   beleza_saude
79756     fashion_bolsas_e_acessorios
100100                  beleza_saude
111105                 esporte_lazer
Name: product_category_name, dtype: object
```

```
1 # Vamos analisar a distribuição de product_category_name, dos 10 mair
2 analise = df.product_category_name.value_counts().iloc[:10]
3 analise
```

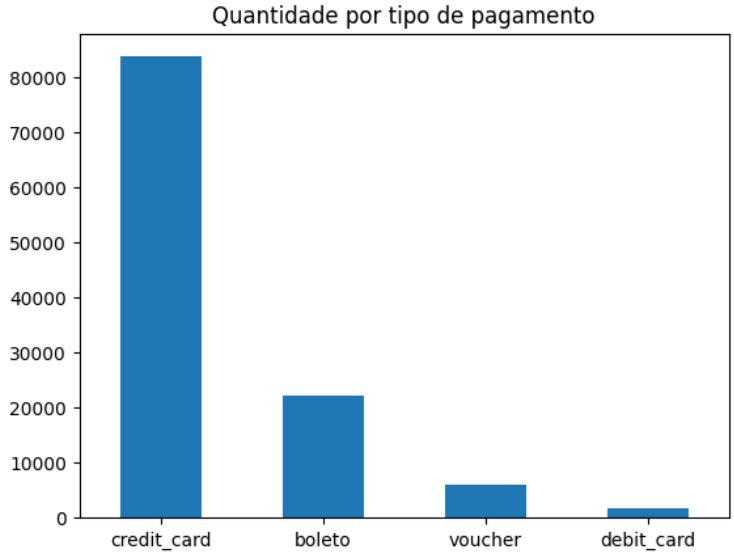
```
cama_mesa_banho           11649
beleza_saude               9761
esporte_lazer              8731
moveis_decoracao           8553
informatica_acessorios     7897
utilidades_domesticas      7172
relogios_presentes         6063
telefonia                  4601
ferramentas_jardim         4463
automotivo                 4283
Name: product_category_name, dtype: int64
```

```
1 analise.sort_values().plot.barh(title='10 categorias com mais vendas');
```


```
1 # Vamos analisar a distribuição de payment_type
2 analise = df.payment_type.value_counts()
3 analise
```

```
credit_card    83706
boleto         22047
voucher         6012
debit_card      1625
Name: payment_type, dtype: int64
```

```
1 analise.plot.bar(rot=0, title="Quantidade por tipo de pagamento");
```




```
1 # Vamos analisar a distribuição de seller_city, as 10 mais.
2 analise = df.seller_city.value_counts().iloc[:10]
3 analise
```

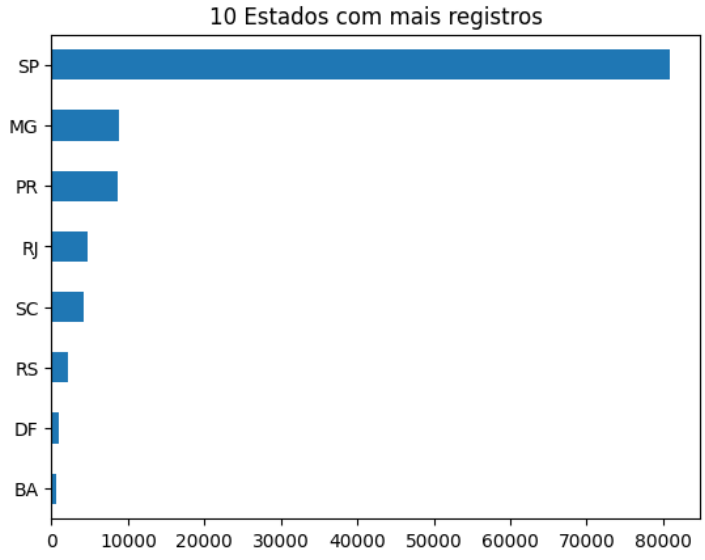
```
sao paulo               28297
ibitinga                 8060
curitiba                 3043
santo andre              3015
sao jose do rio preto    2640
belo horizonte           2527
rio de janeiro           2373
ribeirao preto           2271
maringa                  2230
guarulhos                2011
Name: seller_city, dtype: int64
```

```
1 analise.sort_values().plot.barh(title="10 cidades com mais registros");
```


```
1 # Vamos analisar a distribuição de seller_state
2 analise = df.seller_state.value_counts()
3 analise
```

```
SP    80870
MG     8815
PR     8733
RJ     4798
SC     4160
RS     2203
DF      919
BA      679
GO      527
PE      462
MA      406
ES      371
MT      146
CE       99
MS       60
RN       56
PB       40
RO       14
PI       11
SE       10
PA        8
AM        3
Name: seller_state, dtype: int64
```

```
1 analise.iloc[:8].sort_values().plot.barh(title="10 Estados com mais registros");
```



Pela quantidade de dados unicos, e sua distribuição, podemos considerar as features, como colunas categoricas para nossa analise:

- payment_type
- product_category_name
- seller_city

- seller_state

Contudo, a feature **order_status** aparente ser pouco relevante, uma vez que pouquissimas operações foram canceladas.

2.2 Analisando os dados numéricos

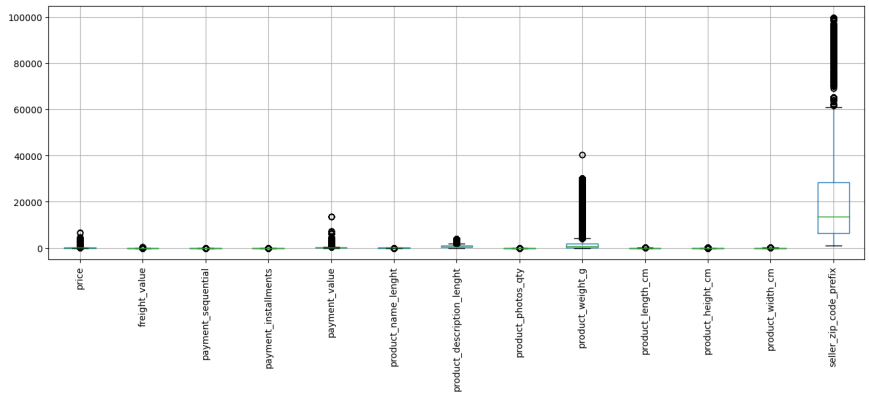
```
# Analisando os dados númericos
df.describe()
```

	price	freight_value	payment_sequential	payment_installments	payment_
count	113390.000000	113390.000000	113390.000000	113390.000000	113390.0
mean	120.184057	20.023985	1.089937	2.942367	172.2
std	182.761548	15.752500	0.682057	2.777802	266.5
min	0.850000	0.000000	1.000000	0.000000	0.0
25%	39.900000	13.080000	1.000000	1.000000	61.0
50%	74.900000	16.320000	1.000000	2.000000	108.1
75%	133.732500	21.200000	1.000000	4.000000	189.3
max	6735.000000	409.680000	26.000000	24.000000	13664.0

Análise de algumas features:

- **price**: a média do valor dos produtos é de R$ 120.08, uma vez que os dados são operações realizadas no Brasil;
- **freight_value**: valor médio do frete foi de R$ 20.00;
- **product_name_lenght**: trata-se da quantidade de letras do nome do produto, não tendo importância para nossa análise.

```
df.boxplot(rot=90, figsize=(16,5));
```




```
# Quantidade de produtos com valor acima da média
mask = df.price >= df.price.mean()
qtd_maior = df[mask].price.count()
print('Quantidade de produtos acima da média:', qtd_maior, '- Proporção:', qtd_maior*100 / df.shape[0], "%")
```

```
Quantidade de produtos acima da média: 31897 - Proporção: 28.130346591410177 %
```

```
# Quantidade de fretes com valor acima da média
mask = df.freight_value <= df.freight_value.mean()
qtd_maior = df[mask].freight_value.count()
print('Quantidade de Fretes abaixo:', qtd_maior, '- Proporção:', qtd_maior*100 / df.shape[0], "%")
```

```
Quantidade de Fretes abaixo: 81852 - Proporção: 72.18625981127083 %
```

Resposta aos quesitos

Pergunta 1: Há pagamentos do tipo “boleto” que tem mais de uma parcela?

```
# Quantidade de parcelas por tipo de pagamento
df_parcelas = df.groupby(['payment_type','payment_installments']).count()['order_status']
df_parcelas.unstack(level=0).T.fillna(0)
```

payment_installments	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.
payment_type									
boleto	0.0	22047.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
credit_card	3.0	26813.0	13199.0	11331.0	7713.0	5827.0	4456.0	1743.0	4869.
debit_card	0.0	1625.0	0.0	0.0	0.0	0.0	0.0	0.0	0.
voucher	0.0	6012.0	0.0	0.0	0.0	0.0	0.0	0.0	0.

4 rows × 24 columns

```
# Quantidade de parcelas por tipo de pagamento, com outro recurso.
pd.crosstab(df.payment_type, df.payment_installments)
```

payment_installments	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	...
payment_type											
boleto	0	22047	0	0	0	0	0	0	0	0	...
credit_card	3	26813	13199	11331	7713	5827	4456	1743	4869	701	...
debit_card	0	1625	0	0	0	0	0	0	0	0	...
voucher	0	6012	0	0	0	0	0	0	0	0	...

4 rows × 24 columns

```
# Quantidade de Boletos com mais de uma parcela
mask = (df.payment_type == 'boleto') & (df.payment_installments != 1)
print("Total de boletos com mais de uma parcela: ", df[mask].count().payment_type)
```

```
Total de boletos com mais de uma parcela:  0
```

Resposta: NÃO

Pergunta 2: Quais são exatamente os pagamentos que tem um valor maior ou menor do que o valor médio dos pagamentos registrados na tabela de pagamentos?

```
# Pagamento com valor MAIOR que a média
mask = df_order_pay.payment_value > df_order_pay.payment_value.mean()
print('Total de operações com pagamentos MAIOR que a média:', df_order_pay[mask].shape[0])
print('MAIOR pagamento:', df_order_pay['payment_value'].max())
df_order_pay[mask].head()
```

```
Total de operações com pagamentos MAIOR que a média: 31012
MAIOR pagamento: 13664.08
```

	order_id	payment_sequential	payment_type	payment_install
8	1f78449c87a54faf9e96e88ba1491fa9	1	credit_card	
10	d88e0d5fa41661ce03cf6cf336527646	1	credit_card	
15	12e5cfe0e4716b59afb0e0f4a3bd6570	1	credit_card	
18	8ac09207f415d55acff302df7d6a895c	1	credit_card	
21	4214cda550ece8ee66441f459dc33a8c	1	credit_card	

```
# Pagamento com valor MENOR que a média
mask = df_order_pay.payment_value < df_order_pay.payment_value.mean()
print('Total de operações com pagamentos MENOR que a média:', df_order_pay[mask].shape[0])
print('MENOR pagamento:', df_order_pay['payment_value'].min())
df_order_pay[mask].head()
```

```
Total de operações com pagamentos MENOR que a média: 72874
MENOR pagamento: 0.0
```

	order_id	payment_sequential	payment_type	payment_install
0	b81ef226f3fe1789b1e8b2acac839d17	1	credit_card	
1	a9810da82917af2d9aefd1278f1dcfa0	1	credit_card	
2	25e8ea4e93396b6fa0d3dd708e76c1bd	1	credit_card	
3	ba78997921bbcdc1373bb41e913ab953	1	credit_card	
4	42fdf880ba16b47b59251dd489d4441a	1	credit_card	

Pergunta 3: Quais são os clientes que provém de uma das 3 cidades mais comuns desta tabela?

```
# Três cidades mais comuns no dataset
cidades = df.seller_city.value_counts().iloc[:3]
cidades
```

```
sao paulo    28297
ibitinga      8060
curitiba      3043
Name: seller_city, dtype: int64
```

```
mask = df['seller_city'].isin(cidades.index)
print('Total de clientes das 3 cidade mais frequentes:', mask.sum())
print('Cidades:', list(cidades.index), '\n')
print('Clientes:')
df[mask]['customer_id'].head()
```

```
Total de clientes das 3 cidade mais frequentes: 39400
Cidades: ['sao paulo', 'ibitinga', 'curitiba']

Clientes:
2628    2f49811c845d9978f54ea3f61741516d
2629    dfbfddf21e93f87163d8695b85506112
2630    dfbfddf21e93f87163d8695b85506112
2631    dfbfddf21e93f87163d8695b85506112
2632    66d8f30fb4390d0b8d3e0b639bd367a9
Name: customer_id, dtype: object
```

Pergunta 4: Criar um ndarray e adicioná-lo a alguma tabela como uma nova coluna.

4.1. Tabela - df_products

```
# Verifica informações das features do DataFrame
df_products.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32951 entries, 0 to 32950
Data columns (total 9 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   product_id                  32951 non-null  object
 1   product_category_name       32341 non-null  object
 2   product_name_lenght         32341 non-null  float64
 3   product_description_lenght  32341 non-null  float64
 4   product_photos_qty          32341 non-null  float64
 5   product_weight_g            32949 non-null  float64
 6   product_length_cm           32949 non-null  float64
 7   product_height_cm           32949 non-null  float64
 8   product_width_cm            32949 non-null  float64
dtypes: float64(7), object(2)
memory usage: 2.3+ MB
```

```
# Recuperando medidas
prod_cumprimento = np.array(df_products.product_length_cm)
prod_altura = np.array(df_products.product_height_cm)
prod_largura = np.array(df_products.product_width_cm)
```

```
# Calculando o volume dos produtos
prod_volume = prod_cumprimento*prod_altura*prod_largura
prod_volume
```

```
array([ 2240., 10800.,  2430., ...,  5103.,  8060.,   168.])
```

```
# Medidas estatisticas
print("Tipo:", type(prod_volume))
print("Maximo:", np.max(prod_volume))
print("Mínimo:", np.min(prod_volume))
print("Média:", np.mean(prod_volume))
print("Mediana:", np.median(prod_volume))
print("Desvio Padrão:", np.std(prod_volume))
print("Variância:", np.var(prod_volume))
```

```
Tipo: <class 'numpy.ndarray'>
Maximo: nan
Mínimo: nan
Média: nan
Mediana: nan
Desvio Padrão: nan
Variância: nan
```

```
print("Quantidade de NaN:", np.isnan(prod_volume).sum())
```

```
Quantidade de NaN: 2
```

```
# Medidas estatisticas
print("Tipo:", type(prod_volume))
print("Maximo:", np.nanmax(prod_volume))
print("Mínimo:", np.nanmin(prod_volume))
print("Média:", np.nanmean(prod_volume))
print("Mediana:", np.nanmedian(prod_volume))
print("Desvio Padrão:", np.nanstd(prod_volume))
print("Variância:", np.nanvar(prod_volume))
```

```
Tipo: <class 'numpy.ndarray'>
Maximo: 296208.0
Mínimo: 168.0
Média: 16564.096694892105
Mediana: 6840.0
Desvio Padrão: 27056.631057479794
Variância: 732061284.1805801
```

```
# Medidas de Quartil
print("Quartil 25:", np.nanquantile(prod_volume, 0.25))
print("Quartil 50:", np.nanquantile(prod_volume, 0.5))
print("Quartil 75:", np.nanquantile(prod_volume, 0.75))
```

```
Quartil 25: 2880.0
Quartil 50: 6840.0
Quartil 75: 18480.0
```

```
# Inserir nova feature, ndArray, no DataFrame
df_products['product_volume'] = prod_volume
```

```
# Cria um array para descrição, com mesma dimensão do array de volumes
prod_volume_desc = np.array(['INDEFINIDO']*prod_volume.shape[0])

# Cria mascaras para os tamanhos
mask_pequeno = prod_volume <= np.nanquantile(prod_volume, 0.25)
mask_medio = (prod_volume > np.nanquantile(prod_volume, 0.25)) & (prod_volume < np.nanquantile(prod_volume, 0.75))
mask_grande = prod_volume >= np.nanquantile(prod_volume, 0.75)

# Descreve os tamanhos
prod_volume_desc[mask_pequeno] = 'PEQUENO'
prod_volume_desc[mask_medio] = 'MÉDIO'
prod_volume_desc[mask_grande] = 'GRANDE'

prod_volume_desc
```

```
array(['PEQUENO', 'MÉDIO', 'PEQUENO', ..., 'MÉDIO', 'MÉDIO', 'PEQUENO'],
      dtype='<U10')
```

```
# Inserir nova feature de descrição, ndArray, no DataFrame
df_products['product_volume_desc'] = prod_volume_desc
```

```
# Visualizando dados
df_products.head()
```

	product_id	product_category_name	product_name_lenght	produc
0	1e9e8ef04dbcff4541ed26657ea517e5	perfumaria	40.0	
1	3aa071139cb16b67ca9e5dea641aaa2f	artes	44.0	
2	96bd76ec8810374ed1b65e291975717f	esporte_lazer	46.0	
3	cef67bcfe19066a932b7673e239eb23d	bebes	27.0	
4	9dc1a7de274444849c219cff195d0b71	utilidades_domesticas	37.0	

Next steps: View recommended plots

4.2. Tabela - df_reviews

```
# Visualiza informações de tabela
df_reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99224 entries, 0 to 99223
Data columns (total 7 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   review_id                99224 non-null  object
 1   order_id                 99224 non-null  object
 2   review_score             99224 non-null  int64 
 3   review_comment_title     11568 non-null  object
 4   review_comment_message   40977 non-null  object
 5   review_creation_date     99224 non-null  object
 6   review_answer_timestamp  99224 non-null  object
dtypes: int64(1), object(6)
memory usage: 5.3+ MB
```

```
# Visualiza tabela
df_reviews.head()
```

	review_id	order_id	review_score	re
0	7bc2406110b926393aa56f80a40eba40	73fc7af87114b39712e6da79b0a377eb	4	
1	80e641a11e56f04c1ad469d5645fdfde	a548910a1c6147796b98fdf73dbeba33	5	
2	228ce5500dc1d8e020d8d1322874b6f0	f9e4b658b201a9f2ecdecbb34bed034b	5	
3	e64fb393e7b32834bb789ff8bb30750e	658677c97b385a9be170737859d3511b	5	
4	f7c4243c7fe1938f181bec41a392bdeb	8e6bfb81e283fa7e4f11123a3fb894f1	5	

```
# Conta conforme a nota
df_reviews.review_score.value_counts()
```

```
5    57328
4    19142
1    11424
3     8179
2     3151
Name: review_score, dtype: int64
```

```
# Cria ndArray
rev_score = np.array(df_reviews['review_score'])
rev_score
```

```
array([4, 5, 5, ..., 5, 4, 1])
```

```
# Criar coluna de booleanos que indique se a avaliação atingiu o valor mais alto (5) ou não.
mask = rev_score == 5
mask

# Cria novo ndArray
rev_score_desc = np.array(['NÃO']*rev_score.shape[0])
rev_score_desc[mask] = 'SIM'

rev_score_desc
```

```
array(['NÃO', 'SIM', 'SIM', ..., 'SIM', 'NÃO', 'NÃO'], dtype='<U3')
```

```
# Inserir nova feature de descrição, ndArray, no DataFrame
df_reviews['review_score_5'] = rev_score_desc
```

```
1 df_reviews.head()
```

	review_id	order_id	review_score	review_comment_title	review_comment_message	revi
0	7bc2406110b926393aa56f80a40eba40	73fc7af87114b39712e6da79b0a377eb	4	NaN	NaN	2
1	80e641a11e56f04c1ad469d5645fdfde	a548910a1c6147796b98fdf73dbeba33	5	NaN	NaN	2
2	228ce5500dc1d8e020d8d1322874b6f0	f9e4b658b201a9f2ecdecbb34bed034b	5	NaN	NaN	2
3	e64fb393e7b32834bb789ff8bb30750e	658677c97b385a9be170737859d3511b	5	NaN	Recebi bem antes do prazo estipulado.	2
4	f7c4243c7fe1938f181bec41a392bdeb	8e6bfb81e283fa7e4f11123a3fb894f1	5	NaN	Parabéns lojas lannister adorei comprar pela I...	2