🔍 Análise Exploratória dos Dados

▼ Importação e junção dos arquivos CSV:

Nessa etapa, importei as bibliotecas necessárias para realizar a junção de todos os arquivos, já que o dataset que encontrei apenas disponibilizava os arquivos separadamente por categoria de produto:

```
import os  # importa a biblioteca os, que permite interagir com o sistema operacional
import glob  # importa a biblioteca glob, que permite encontrar todos os arquivos que correspondem a um padrão de nome
import pandas as pd  # importa a biblioteca pandas e a renomeia como pd

os.chdir(r"C:\Users\Maiara\OneDrive\MACKENZIE\Amazon")  # define o diretório de trabalho para a pasta "Amazon" no OneDrive

extension = 'csv'  # define a extensão desejada para os arquivos a serem lidos como "csv"
all_filenames = [i for i in glob.glob('*.{}'.format(extension))]  # cria uma lista com o nome de todos os arquivos na pasta atual com a extensão "csv"

df = pd.concat([pd.read_csv(f) for f in all_filenames])  # concatena todos os arquivos da lista "all_filenames" em um único DataFrame "df"
df.to_csv("df.csv", index=False, encoding='utf-8-sig')  # salva o DataFrame "df" em um arquivo CSV chamado "df.csv", sem incluir o índice e usando a codificação "utf-8-sig"
```

▼ Número de exemplares (linhas) e dimensões (colunas):

Após realizar a junção e concatenação de todos os arquivos para o Dataframe, verifiquei as dimensões do Dataframe para validar o **volume final dos meus dados** e se eu poderia trabalhar dessa forma ou teria que utilizar **técnicas de amostragem ou redução de dimensionalidade para torná-lo mais** `gerenciável`. Além disso, saber as dimensões dos dados também me ajudam a determinar a complexidade da análise. Caso o conjunto tivesse **muitas variáveis**, poderia ser necessário utilizar técnicas mais sofisticadas de análise exploratória, como análise de componentes principais ou agrupamento hierárquico. Já no caso de o conjunto de dados ter **poucas variáveis**, técnicas mais simples de análise exploratória, como tabelas de frequência e gráficos de barras, podem ser suficientes.

Caso houvessem muitos **dados faltantes ou duplicados**, eu conseguiria identificar logo “de cara” também conhecendo o tipo de negócio em que atuo:

```
df.shape
```

▼ Tipos de dados:

Nessa etapa, utilizei o método dtypes tendo em mente que cada tipo de dado tem suas próprias características e propriedades, e essas características afetam diretamente as `operações` que podem ser realizadas com esses dados. Além disso, o tipo de dado também pode afetar o **desempenho do programa e o uso de memória** e isso é importante já que o meu Dataset possui 1175955 linhas, o que já o torna grande podendo apresentar desafios em termos de processamento.

Um outro motivo pelo qual utilizei a função foi a própria **precisão das operações matemáticas** que serão realizadas com esses valores. Sempre preciso saber se estou trabalhando com um tipo float ou int, por exemplo:

```
df.dtypes # A função "dtypes" retorna uma série com os tipos de dados de cada coluna
```

```
name              object
main_category     object
sub_category      object
image             object
link              object
ratings           object
no_of_ratings     object
discount_price    object
actual_price      object
Unnamed: 0       float64
dtype: object
```

▼ Valores perdidos ou incorretos:

Assim que visualizei o Dataframe, identifiquei uma coluna chamada "Unnamed: 0", que não possuía nenhum valor. Como essa coluna **não me traria informações importantes para a compreensão dos dados e dos** `padrões` **presentes neles**, utilizei o método “Drop” do Pandas para excuí-la do Dataframe na intenção de **simplificar a análise, reduzir a quantidade de dados que precisam ser processados e melhorar a legibilidade** dos resultados:

```
df = df.drop('Unnamed: 0', axis=1) # A função "drop()" é usada para remover uma coluna ou linha do DataFrame usando o eixo 1 (axis=1), que indica a coluna.
# O resultado é atribuído novamente à variável "df".
```

main_category	sub_category	image	link	ratings	no_of_ratings	discount_price	actual_price	Unnamed: 0
appliances	Air Conditioners	https://m.media-amazon.com/images/I/31UISB90sY...	https://www.amazon.in/Lloyd-Inverter-Convertib...	4.2	2,255	₹32,999	₹58,990	NaN
appliances	Air Conditioners	https://m.media-amazon.com/images/I/51JFb7FctD...	https://www.amazon.in/LG-Convertible-Anti-Viru...	4.2	2,948	₹46,490	₹75,990	NaN
appliances	Air Conditioners	https://m.media-amazon.com/images/I/51JFb7FctD...	https://www.amazon.in/LG-Inverter-Convertible-...	4.2	1,206	₹34,490	₹61,990	NaN
appliances	Air Conditioners	https://m.media-amazon.com/images/I/51JFb7FctD...	https://www.amazon.in/LG-Convertible-Anti-Viru...	4.0	69	₹37,990	₹68,990	NaN
appliances	Air Conditioners	https://m.media-amazon.com/images/I/41IrtqXPiW...	https://www.amazon.in/Carrier-Inverter-Split-C...	4.1	630	₹34,490	₹67,790	NaN

Assim que abri o DataFrame e utilizei o método unique() para entender o porquê o campo estava sendo interpretado pelo Python como “Object” encontrei o valor “₹” e textos aonde deveriam haver apenas números na coluna “ratings”, soube que teria que `tratar` **esses dados**, pois **não seria possível convertê-los em um tipo numérico para realizar as operações necessárias** além de causar erros ou **distorções em cálculos estatísticos e gráficos** e dificultar a própria **compreensão dos dados**. Como se trata de uma coluna de avaliação do produto, não fazia sentido valores monetários e valores do tipo texto, então optei por substituir por um valor ausente, indicando que por algum motivo, no processo de raspagem, aqueles dados se perderam:

```
# Tratando a coluna ratings

print(df['ratings'].unique())
# Printa uma lista de valores únicos encontrados na coluna 'ratings' para identificar valores não numéricos

import numpy as np
# Importa a biblioteca NumPy e a renomeia como "np".

df['ratings'] = df['ratings'].replace(['Get', 'FREE','₹68.99','₹65','₹70','₹100','₹99','₹2.99'], np.nan).astype(float)
# A função "replace()" é usada para substituir os valores 'Get', 'FREE', '₹68.99', '₹65', '₹70', '₹100', '₹99' e '₹2.99' pelo valor NaN, que indica um valor ausente.
# Em seguida, a função "astype()" é usada para converter a coluna para o tipo de dados float, que é o tipo mais adequado para a coluna 'ratings'.
# O resultado é atribuído novamente à coluna 'ratings' do DataFrame 'df'.
```

Na coluna “no_of_ratings” além dos campos do tipo texto encontrados anteriormente serem em uma quantidade muito maior nesse campo e tornando muito demorada sua listagem para utilizar um método simples como o replace(), eu também tive **problemas com casas decimais incorretas** e “ , “ sendo utilizadas como separador das casas decimais no lugar do “ . “ que é como o Python entende originalmente. Nesse caso, tive que fazer uso de duas `funções` para resolver o problema:

```
#Tratando a coluna no_of_ratings

def change_value(valor):
    # Verifica se o valor é uma string com mais de 10 caracteres
    if isinstance(valor, str) and len(valor) > 10:
        return '0'  # Substitui o valor por 0
    # Verifica se o valor é um número int ou float com mais de 10 dígitos
    elif isinstance(valor, (int, float)) and len(str(valor)) > 10:
        return 0  # Substitui o valor por 0
    else:
        return valor  # Retorna o valor original

# Aplica a função change_value à coluna 'no_of_ratings' do df usando a função lambda
df['no_of_ratings'] = df['no_of_ratings'].apply(lambda x: change_value(x))

# Substitui as vírgulas por pontos na coluna 'no_of_ratings' e converte os valores para o tipo float
df['no_of_ratings'] = df['no_of_ratings'].str.replace(',', '.').astype(float)
```

Para tratar a coluna “discount_price”, além do método replace() para tratar o problema com o '₹' e o uso de funções para a **grande quantidade de textos no lugar onde deveriam existir valores numéricos**, também precisei realizar o uso de funções para a correção das casas decimais como no campo anterior:

```
#Tratando a coluna discount_price

#Remove o sinal '₹' do campo
df['discount_price'] = df['discount_price'].str.replace('₹', '')

def correct_decimal(valor):
    if isinstance(valor, str) and '.' in valor:
        valor = valor.replace('.', ',', 1) #substitui o segundo ponto decimal por uma vírgula
    return valor

#Aplicar a função a cada valor da coluna
df['discount_price'] = df['discount_price'].apply(correct_decimal)

def convert_to_float(valor):
    if isinstance(valor, str):
        valor = valor.replace(',', '.')  # substituir vírgula por ponto
    try:
        valor = float(valor)  # converte pra float
    except ValueError:
        valor = float('NaN')  # converte valor inválido pra "NOT A NUMBER"
```

```
    return valor

df['discount_price'] = df['discount_price'].apply(convert_to_float) #Aplica a função pra cada valor da coluna
```

Para tratar o campo “actual_price” foi exatamente o mesmo processo pois os problemas eram exatamente os mesmos:

```
# Tratando a coluna actual_price

df['actual_price'] = df['actual_price'].str.replace('₹', '') #Remove o sinal '₹' do campo

def correct_decimal(valor):
    if isinstance(valor, str) and '.' in valor: # substitui o segundo ponto decimal por uma vírgula
        valor = valor.replace('.', ',', 1)
    return valor


df['actual_price'] = df['actual_price'].apply(correct_decimal) # aplica a função a cada valor da coluna

def convert_to_float(valor):
    if isinstance(valor, str):
        valor = valor.replace(',', '.')  # substitui vírgula por ponto
    try:
        valor = float(valor)  # converte pra float
    except ValueError:
        valor = float('NaN')  # converte valor inválido pra "NOT A NUMBER"
    return valor

df['actual_price'] = df['actual_price'].apply(convert_to_float) # aplica a função a cada valor da coluna
```

Resultado após to tratamento dos campos:

name	main_category	sub_category	image	link	ratings	no_of_ratings	discount_price	actual_price
Lloyd 1.5 Ton 3 Star Inverter Split Ac (5 In 1...	appliances	Air Conditioners	https://m.media-amazon.com/images/I/31UISB90sY...	https://www.amazon.in/Lloyd-Inverter-Convertib...	4.2	2.255	32.999	58.990
LG 1.5 Ton 5 Star AI DUAL Inverter Split AC (C...	appliances	Air Conditioners	https://m.media-amazon.com/images/I/51JFb7FctD...	https://www.amazon.in/LG-Convertible-Anti-Viru...	4.2	2.948	46.490	75.990
LG 1 Ton 4 Star Ai Dual Inverter Split Ac (Cop...	appliances	Air Conditioners	https://m.media-amazon.com/images/I/51JFb7FctD...	https://www.amazon.in/LG-Inverter-Convertible-...	4.2	1.206	34.490	61.990
LG 1.5 Ton 3 Star AI DUAL Inverter Split AC (C...	appliances	Air Conditioners	https://m.media-amazon.com/images/I/51JFb7FctD...	https://www.amazon.in/LG-Convertible-Anti-Viru...	4.0	69.000	37.990	68.990
Carrier 1.5 Ton 3 Star Inverter Split AC (Copp...	appliances	Air Conditioners	https://m.media-amazon.com/images/I/41IrtqXPiW...	https://www.amazon.in/Carrier-Inverter-Split-C...	4.1	630.000	34.490	67.790
...	...	...	...	...	...	...	...	...
Adidas Regular Fit Men's Track Tops	sports & fitness	Yoga	https://m.media-amazon.com/images/I/71tHAR9plY...	https://www.amazon.in/Adidas-Regular-Mens-Trac...	3.2	9.000	3.449	4.599
Redwolf Noice Toit Smort - Hoodie (Black)	sports & fitness	Yoga	https://m.media-amazon.com/images/I/41pKrMZ5lQ...	https://www.amazon.in/Redwolf-Noice-Smort-Cott...	2.0	2.000	1.199	1.999
Redwolf Schrute Farms B&B - Hoodie (Navy Blue)	sports & fitness	Yoga	https://m.media-amazon.com/images/I/41n9u+zNSc...	https://www.amazon.in/Redwolf-Schrute-Farms-Ho...	4.0	1.000	1.199	1.999
Puma Men Shorts	sports & fitness	Yoga	https://m.media-amazon.com/images/I/51LoWv5JDt...	https://www.amazon.in/Puma-Woven-Short-5208526...	4.4	37.000	NaN	NaN
Mothercare Printed Cotton Elastane Girls Infan	sports & fitness	Yoga	https://m.media-amazon.com/images/I/814X-sI3DB...	https://www.amazon.in/Mothercare-Carrot-Regula...	4.6	5.000	1.039	1.299

▼ Medidas de posição e dispersão:

As medidas de posição e dispersão quando avaliadas sob o conjunto total dos dados podem não trazer as respostas para as perguntas que procuramos responder pois oferecem uma visão `geral` dos dados, mas de qualquer forma temos certos insights e podemos fazer observações como:

A **média** de 3.84 indica que a avaliação média tende a ser um pouco acima do valor médio (que seria 2.5 em uma escala de 1 a 5). Isso pode indicar que, de maneira geral, **as pessoas que fazem as avaliações tendem a ser** `positivas` em relação ao item avaliado.

O **desvio padrão** de 0.75 indica que as avaliações tendem a se concentrar em torno da média, com poucas avaliações muito baixas ou muito altas.

```
df.describe()
```

	ratings	no_of_ratings	discount_price	actual_price
count	774717.000000	787341.000000	1.039503e+06	1.122810e+06
mean	3.839880	81.522315	3.348425e+02	2.566563e+02
std	0.753968	166.752112	3.007437e+02	3.709120e+02
min	1.000000	0.000000	1.000000e+00	0.000000e+00
25%	3.500000	3.000000	6.000000e+00	1.999000e+00
50%	3.900000	12.000000	2.990000e+02	4.999000e+00
75%	4.300000	65.588000	5.460000e+02	5.490000e+02
max	5.000000	999.000000	9.999900e+02	9.999900e+02

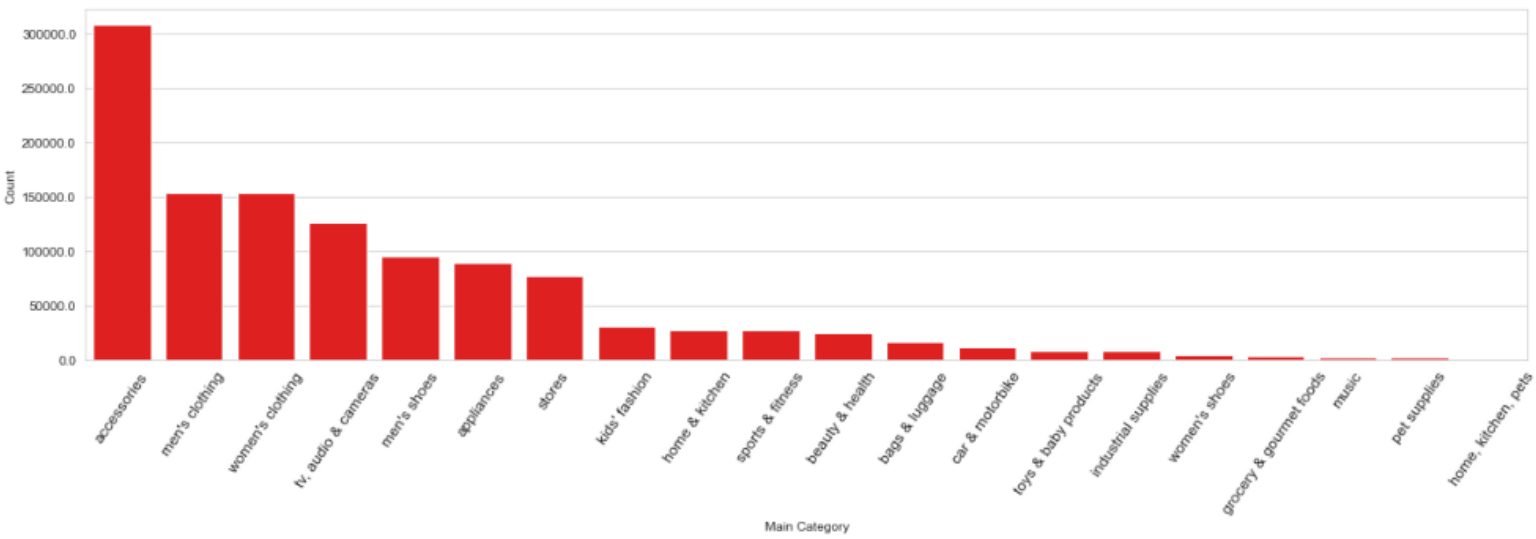
▼ Distribuição e frequência:

Com um **gráfico de frequência** conseguimos identificar que a maior quantidade de vendas das categorias avaliadas foi a categoria “`acessories`” com mais de 300.000, “men’s clothing” e “women’s clothing” com 150.000, e “tv, audio e cameras” com mais de 100.000 e “men’s shoes” com quase 100.000 vendas e a menor quantidade foi “home, kitchen, pets” com 0 vendas no período.

Isso pode nos indicar que:

- A categoria "accessories" pode ser a mais popular entre os consumidores, gerando mais vendas;
- As categorias "men's clothing" e "women's clothing" também são populares, indicando que roupas são um item comum nas compras online;
- O aumento na `demanda` por "tv, audio e cameras" **pode estar relacionada** ao aumento da produção de conteúdo online, tanto por influenciadores quanto por empresas que buscam se comunicar com seu público de forma mais dinâmica e atrativa.
- A categoria "men's shoes" aparenta ser `popular` entre os consumidores masculinos e em contrapartida a categoria “women’s shoes” foi pouco procurada pelo público feminino podendo indicar **indisponibilidade dos produtos procurados, valores elevados e pouco atrativos**, pode indicar que a oferta de produtos para mulheres não tenha atendido às necessidades e preferências das consumidoras ou até que houveram promoções focadas no público masculino nesse período.
- A **ausência de vendas** nas categorias "home, kitchen, pets", “pet supplies”, “music” e “grocery e gourmet foods” pode `indicar` falta de interesse nessas categorias ou falta de ofertas de produtos adequados para essas categorias.

```
import seaborn as sns  # Importa a biblioteca Seaborn
import pandas as pd  # Importa a biblioteca Pandas
import matplotlib.pyplot as plt # Importa o método pyplot

sns.set_style('whitegrid')

# Obtém a contagem de cada valor único da coluna 'main_category' e cria uma tabela de frequência
count = df['main_category'].value_counts().reset_index()
count.columns = ['main_category', 'count']

# Plota a tabela de frequência como um gráfico de barras
fig, ax = plt.subplots(figsize=(20, 5))
g = sns.barplot(data=count, x='main_category', y='count', color='red', ax=ax)

g.set(xlabel='Main Category', ylabel='Count')
g.set_xticklabels(count['main_category'], rotation=55, fontsize=12)
g.set_yticklabels(g.get_yticks(), fontsize=10)
g.tick_params(axis='y', labelsize=10)

plt.show()
```

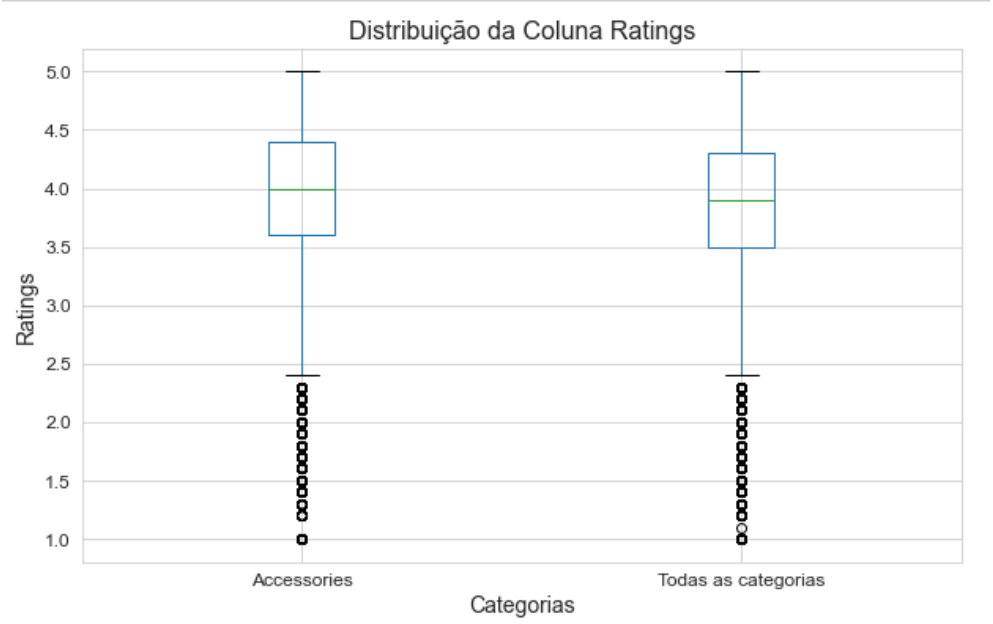


Observando a **distribuição das avaliações**, é possível concluir que a categoria "accessories" apresenta uma `distribuição de ratings` mais concentrada em valores próximos a 4.0, enquanto que as demais categorias (incluindo "Todas as categorias") possuem uma distribuição um pouco mais variada.

Isso pode indicar que os produtos da categoria "accessories" possuem uma qualidade consistente e/ou que os consumidores **tendem a avaliá-los de forma similar**, enquanto que os produtos de outras categorias apresentam uma variedade maior em termos de qualidade e/ou as avaliações dos consumidores são mais diversas. No entanto, é **importante analisar outros aspectos dos dados para confirmar ou refutar essas hipóteses**.

Analisando os `Outliers`, é possível entender que a categoria “acessories” possui praticamente o mesmo desempenho das demais categorias, com poucas avaliações abaixo de 2.5.

```
# Cria uma figura com tamanho 10x6 polegadas
fig = plt.figure(figsize=(10,6))

# Plota o boxplot com tamanho aumentado para a categoria "accessories"
df[df['main_category'] == "accessories"].boxplot(column='ratings', figsize=(10,6), positions=[1])

# Plota o boxplot com tamanho aumentado para todas as categorias
df.boxplot(column='ratings', figsize=(10,6), positions=[2])

# Adiciona um título e labels aos eixos x e y
plt.title('Distribuição da Coluna Ratings', fontsize=16)
plt.xlabel('Categorias', fontsize=14)
plt.ylabel('Ratings', fontsize=14)

# Define o label do eixo x como o nome da categoria
plt.xticks([1, 2], ["Accessories", "Todas as categorias"], fontsize=12)

# Adiciona uma grade ao gráfico
plt.grid(True)

# Aumenta o tamanho das fontes dos eixos x e y
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

# Exibe o gráfico
plt.show()
```



▼ Correlações:

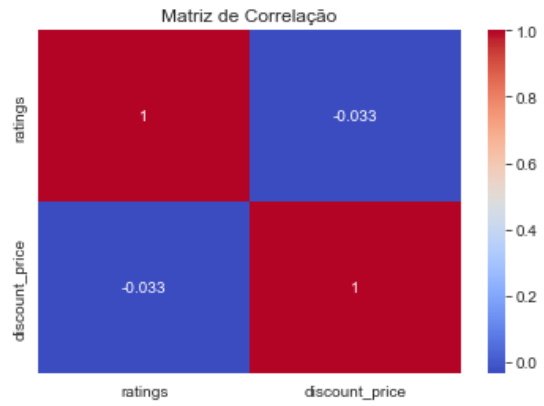
Para verificar a correlação entre a quantidade de avaliações e o preço do produto com desconto, ou seja, se os produtos mais caros tendem a ter menos avaliações do que os produtos mais baratos empreguei a **matriz de correlação** com um “mapa de calor” e obtive uma correlação negativa de -0.033.

Isso significa que, em média, os produtos mais caros não tendem a ter menos avaliações do que os produtos mais baratos.

A correlação negativa significa que as duas variáveis se movem em direções opostas: quando uma aumenta, a outra tende a diminuir. No caso, **a correlação negativa mostra que, em geral, à medida que o preço do produto com desconto aumenta, a quantidade de avaliações tende a diminuir, mas essa relação é muito fraca.** Por exemplo, pode haver produtos caros que têm muitas avaliações e produtos baratos que têm poucas avaliações.

Vale lembrar sempre que a `correlação não implica causalidade`, ou seja, a correlação negativa não significa necessariamente que os produtos mais caros tenham menos avaliações por causa do preço. **Outros fatores, como a qualidade do produto, a marca ou a popularidade do vendedor, podem estar afetando tanto o preço quanto a quantidade de avaliações.**

```
# Seleciona apenas as colunas 'no_of_ratings' e 'discount_price'
cols = ['ratings', 'discount_price']
df_selected = df[cols]

# Calcula a matriz de correlação
corr= df_selected.corr()

# Cria um heatmap com a matriz de correlação
sns.heatmap(corr, annot=True, cmap='coolwarm')

# Define o título do gráfico
plt.title('Matriz de Correlação')

# Exibe o gráfico
plt.show()
```



▼ Anomalias e outliers:

Esse gráfico de dispersão **mostra a relação entre a avaliação média dos produtos em cada categoria e o número de avaliações recebidas por esses produtos.** Cada ponto no gráfico representa uma categoria de produtos e a posição vertical do ponto indica a `avaliação média` dos produtos nessa categoria, enquanto a posição horizontal do ponto indica o número médio de avaliações recebidas por produto nessa categoria.

Ao analisar o gráfico, podemos observar que as categorias que têm uma avaliação média mais alta estão geralmente localizadas mais acima no gráfico, indicando que essas categorias têm uma avaliação média maior do que as categorias localizadas mais abaixo. Além disso, podemos ver que as categorias que têm um n**úmero médio de avaliações** mais alto estão geralmente localizadas mais à direita no gráfico, indicando que essas categorias tendem a receber mais avaliações do que as categorias localizadas mais à esquerda.

O gráfico de dispersão nos ajuda a identificar se existe uma relação entre o número de avaliações recebidas pelos produtos e a avaliação média desses produtos. Também pode nos auxiliar ao comparar o desempenho das diferentes categorias de produtos em termos de avaliação média e número de avaliações recebidas.

```
# Agrupar o dataframe por categoria e calcular a média de ratings e no_of_ratings em cada grupo
grouped = df.groupby('main_category')[['ratings', 'no_of_ratings']].mean()

# Criar uma nova figura com tamanho 10x8 polegadas
plt.figure(figsize=(13, 8))

# Plotar o gráfico de dispersão para cada categoria
for category, data in grouped.iterrows():
    x = data['no_of_ratings']
    y = data['ratings']
    plt.scatter(x, y, label=category, s=50)  # aumentar o tamanho dos pontos

# Definir os rótulos dos eixos
plt.xlabel('Número de Avaliações')
plt.ylabel('Avaliação Média')

# Adicionar uma legenda com as categorias
plt.legend()

# Mostrar o gráfico
plt.show()
```