Escola Korú – Powered by Ifood

Grupo 4

Análise de Dados

OSASCO
18/12/2023

Resumo Executivo

Este documento tem por objetivo trazer uma análise detalhada sobre os registros de contidos no conjunto de dados "Brazilian E-Commerce Public Dataset by Olist" que contempla o ano de 2016 a 2018, trazendo a partir dela insights valiosos para aprimorar as operações de negócios, otimizar a logística e melhorar a experiência do cliente.

Metodologia

O projeto será dividido em três etapas:

Preparação dos dados: Nessa etapa faremos
- Importação
- Limpeza
- Transformação

Análise de dados: Nessa etapa serão realizadas:
- Análise Exploratória e descritiva dos dados
- Identificação de padrões e tendências

Visualização de dados: Nessa etapa, os insights gerados na etapa anterior serão apresentados de forma visual para facilitar a compreensão dos tomadores de decisão.

Resultados esperados

Ao final do projeto, espera-se gerar os seguintes resultados:

- **Melhorias na recomendação de produtos e personalização da experiência do cliente:** Os insights gerados nesta etapa ajudarão a empresa a recomendar produtos mais relevantes para os clientes e a personalizar a experiência de compra de acordo com os interesses e preferências de cada cliente.
- **Otimização da gestão de inventário e operações logísticas:** Os insights gerados nesta etapa ajudarão a empresa a otimizar o estoque, reduzir os custos logísticos e melhorar o tempo de entrega.
- **Melhoria da tomada de decisão estratégica e operacional:** As visualizações de dados criadas nesta etapa ajudarão os tomadores de decisão a entender melhor o negócio e a tomar decisões mais assertivas.

Cronograma

O projeto será executado no período de um mês, tendo por base o seguinte cronograma estimado:

- **Semanas 1-2:** Preparação dos dados
- **Semanas 3-4:** Análise de dados

- **Semanas 5-6:** Visualização de dados

Recursos necessários

Para executar o projeto, serão necessários os seguintes recursos:

- Computador com processador Intel Core i5 ou superior
- 16 GB de RAM
- Sistema operacional Windows ou macOS
- Softwares Python, PySpark
- **Conjunto de dados "Brazilian E-Commerce Public Dataset by Olist"**

Sobre a Olist

A Olist é uma plataforma que atua como um facilitador no universo do comércio online. Sua proposta inovadora é conectar vendedores e compradores, proporcionando uma experiência mais simples e eficiente. Ao criar um ambiente virtual que reúne uma diversidade de produtos de diferentes fornecedores, a Olist oferece aos consumidores uma ampla gama de opções em um só lugar.

A plataforma busca simplificar o processo de venda para os comerciantes, permitindo que foquem em seus produtos, enquanto a Olist cuida de aspectos como logística e visibilidade online. Essa abordagem visa criar um ecossistema colaborativo, onde tanto vendedores quanto consumidores encontram benefícios mútuos, promovendo assim uma experiência de comércio mais dinâmica e inclusiva.

Em 2017/2018 o Brasil integrava o a 9º posição do ranking dos 10 países com maior receita de e-commerce por usuário, conforme matéria da Forbes.

A Olist, sempre atenta aos movimentos de mercado, vem evoluindo desde a sua criação em 2007. Hoje já conta com mais de 40 mil lojas parceiras, 6,5Mi de produtos cadastrados e mais de 12Mi de vendas.

Sobre o conjunto de dados

Dados Públicos de E-Commerce Brasileiro da Olist - O conjunto de dados tem informações de 100 mil pedidos de **2016 a 2018** feitos em vários marketplaces no Brasil. Suas características permitem visualizar um pedido de várias dimensões: desde o status do pedido, preço, pagamento e desempenho do frete até a localização do cliente, atributos do produto e, finalmente, comentários escritos pelos clientes. Também há um conjunto de dados de geolocalização que relaciona CEPs brasileiros com coordenadas lat/lng.

Código e Análise

Abaixo estão trechos de códigos e análises, que serão utilizados em diferentes etapas do projeto:

1 **# importando as bibliotecas**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import folium
import seaborn as sns
```

Esse código em Python importa algumas bibliotecas para análise de dados, visualização e mapas geográficos.

pandas (pd):

Biblioteca para manipulação e análise de dados. Ela oferece estruturas de dados flexíveis, como DataFrames, que facilitam a manipulação e análise de conjuntos de dados.

numpy (np):

Biblioteca para computação numérica em Python. Fornece suporte para arrays e matrizes, além de funções matemáticas para operações eficientes em grandes conjuntos de dados.

matplotlib.pyplot (plt):

Biblioteca para criar gráficos e visualizações. O módulo pyplot fornece uma interface semelhante à do MATLAB para a criação de gráficos.

folium:

Biblioteca para criar mapas interativos em Python. É especialmente útil para a visualização de dados geográficos usando mapas interativos baseados na web.

seaborn (sns):

Biblioteca de visualização de dados baseada no matplotlib. Ela fornece uma interface de alto nível para a criação de gráficos estatísticos atraentes e informativos.

2 # Carregando os conjuntos de dados do diretório do e-commerce em variáveis individuais para facilitar a manipulação.

```
diretorio_ecommerce = 'C:/Users/gisle/Dropbox/Engenharia de dados/Módulo III/brazilian_ecommerce/'

df_item = pd.read_csv(diretorio_ecommerce + 'olist_order_items_dataset.csv')
df_reviews = pd.read_csv(diretorio_ecommerce + 'olist_order_reviews_dataset.csv')
```

```
df_orders = pd.read_csv(diretorio_ecommerce + 'olist_orders_dataset.csv')
df_products          =          pd.read_csv(diretorio_ecommerce          +
'olist_products_dataset.csv')
df_geolocation         =         pd.read_csv(diretorio_ecommerce         +
'olist_geolocation_dataset.csv')
df_sellers          =          pd.read_csv(diretorio_ecommerce          +
'olist_sellers_dataset.csv')
df_order_pay          =          pd.read_csv(diretorio_ecommerce          +
'olist_order_payments_dataset.csv')
df_customers          =          pd.read_csv(diretorio_ecommerce          +
'olist_customers_dataset.csv')
df_category          =          pd.read_csv(diretorio_ecommerce          +
'product_category_name_translation.csv')
```

Esse código carrega conjuntos de dados da base que será analisada, em variáveis individuais usando a biblioteca pandas. Cada conjunto de dados é lido de um arquivo CSV correspondente e armazenado em um DataFrame do pandas.

diretorio_ecommerce:

Uma variável que armazena o caminho do diretório onde os dados do e-commerce estão localizados. Neste caso, o diretório é 'C:/Users/gisle/Dropbox/Engenharia de dados/Módulo III/brazilian_ecommerce/'.

DataFrames: Para a atividade proposta, todos os Dataframes utilizados serão estruturas de dados bidimensionais, semelhantes a tabelas. São eles:

df_item: Contém dados sobre os itens dos pedidos.

df_reviews: Contém dados sobre as avaliações dos pedidos.

df_orders: Contém dados sobre os pedidos.

df_products: Contém dados sobre os produtos.

df_geolocation: Contém dados sobre a geolocalização.

df_sellers: Contém dados sobre os vendedores.

df_order_pay: Contém dados sobre os pagamentos dos pedidos.

df_customers: Contém dados sobre os clientes.

df_category: Contém traduções das categorias de produtos.

Leitura dos arquivos CSV:

Cada linha utiliza a função pd.read_csv para ler um arquivo CSV correspondente ao conjunto de dados e carregá-lo em um DataFrame do pandas. A função pd.read_csv é

usada para ler dados de arquivos CSV (um formato de arquivo simples usado para armazenar dados tabulares) e criar DataFrames.

No geral, esse código prepara o ambiente de trabalho carregando conjuntos de dados relacionados ao e-commerce em DataFrames, o que facilitará a manipulação e análise desses dados ao longo do código.

3 # Lista de DataFrames, cada um representando um conjunto de dados específico

```
datasets = [df_customers, df_geolocation, df_orders, df_item, df_order_pay,
            df_reviews, df_products, df_sellers]

# Nomes associados a cada DataFrame para identificação
names   =   ['df_customer',   'df_geolocation',   'df_orders',   'df_item',
'df_order_pay',
         'df_reviews', 'df_products', 'df_sellers']

# Lista vazia para armazenar informações sobre os conjuntos de dados
data_info = []


# Iteração pelos DataFrames e seus respectivos nomes para coletar
informações
for name, df in zip(names, datasets):
    # Coleta de informações sobre cada DataFrame
    info = {
        'conjunto_de_dados': name,
        'n_linhas': df.shape[0],  # Número de linhas no DataFrame
        'n_colunas': df.shape[1],  # Número de colunas no DataFrame
       'quantidade_de_nulos': df.isnull().sum().sum(),  # Quantidade total
de valores nulos
          'quantidade_de_colunas_com_nulos': len([col for col, null in
df.isnull().sum().items() if null > 0]),  # Número de colunas com valores
nulos
         'percentual_de_nulos': df.isnull().sum().sum() / (df.shape[0] *
df.shape[1]) * 100,  # Percentual de valores nulos no DataFrame
             'colunas_com_nulos':  ',  '.join([col  for  col,  null  in
df.isnull().sum().items() if null > 0])  # Nomes das colunas com valores
nulos
    }
    data_info.append(info)  # Adiciona as informações do DataFrame à lista
de informações

data_info = pd.DataFrame(data_info)  # Converte a lista de informações em
um DataFrame
data_info.style.background_gradient()   #  Aplica  um  estilo  visual  ao
DataFrame para melhorar a visualização
```

Este código coleta informações sobre o conjunto de dados, o número de linhas, o número de colunas, a quantidade de valores nulos, o número de colunas com valores nulos, o percentual de valores nulos e os nomes das colunas com valores nulos.

O código começa declarando uma lista de DataFrames, cada um representando um conjunto de dados específico. Também declara uma lista de nomes associados a cada DataFrame para identificação.

Em seguida, cria uma lista vazia para armazenar informações sobre os conjuntos de dados.

Depois, itera pelos DataFrames e seus respectivos nomes para coletar informações. Para cada DataFrame, o código coleta as seguintes informações:

- O nome do conjunto de dados
- O número de linhas no DataFrame
- O número de colunas no DataFrame
- A quantidade total de valores nulos no DataFrame
- O número de colunas com valores nulos no DataFrame
- O percentual de valores nulos no DataFrame
- Os nomes das colunas com valores nulos no DataFrame

O código armazena essas informações em um dicionário. Em seguida, adiciona o dicionário à lista de informações. Finalmente, o código converte a lista de informações em um DataFrame e aplica um estilo visual ao DataFrame para melhorar a visualização.

Aqui está a saída do código:

	conjunto_de_dados	n_linhas	n_colunas	quantidade_de_nulos	quantidade_de_colunas_com_nulos	percentual_de_nulos	colunas_com_nulos
0	df_customer	99441	5	0	0	0.000000	
1	df_geolocation	1000163	5	0	0	0.000000	
2	df_orders	99441	8	4908	3	0.616949	order_approved_at, order_delivered_carrier_date, order_delivered_customer_date
3	df_item	112650	7	0	0	0.000000	
4	df_order_pay	103886	5	0	0	0.000000	
5	df_reviews	99224	7	145903	2	21.006295	review_comment_title, review_comment_message
6	df_products	32951	9	2448	8	0.825468	product_category_name, product_name_lenght, product_description_lenght, product_photos_qty, product_weight_g, product_length_cm, product_height_cm, product_width_cm
7	df_sellers	3095	4	0	0	0.000000	

4 #verificação de onde estão os valores nulos no df_orders

```
nulos_orders = (df_orders.isnull().mean() *
100).round(2).sort_values(ascending=False)
print(nulos_orders)
```

Este código verifica onde estão os valores nulos no DataFrame. Ele pode ser usado para identificar as colunas que têm mais valores nulos. Essa informação pode ser útil para tomar decisões sobre como tratar esses valores.

Saída:

df_orders.order_delivered_customer_date	2.98
order_delivered_carrier_date	1.79
order_approved_at	0.16
order_id	0.00
customer_id	0.00
order_status	0.00
order_purchase_timestamp	0.00
order_estimated_delivery_date	0.00
dtype:	float64

5 #verificação de onde estão os nulos em df_products

```
nulos_products = (df_products.isnull().mean() *
100).round(2).sort_values(ascending=False)

print(nulos_products)
```

Verifica os valores nulos no DataFrame **df_products.** Teremos uma sequência de códigos que fará a verificação de nulos.

product_category_name	1.85
product_name_lenght	1.85
product_description_lenght	1.85
product_photos_qty	0.01
product_weight_g	0.01
product_length_cm	0.01
product_height_cm	0.01
product_width_cm	0.01
product_id	0.00

6 # Remoção dos nulos

```
df_products = df_products.dropna()
df_products.isnull().sum()
```

product_id	0
product_category_name	0
product_name_lenght	0
product_description_lenght	0
product_photos_qty	0

product_weight_g	0
product_length_cm	0
product_height_cm	0
product_width_cm	0
dtype: int64	

7 # Verificação de onde estão os nulos em df_reviews

```
nulos_reviews = (df_reviews.isnull().mean() * 100).round(2).sort_values(ascending=False)
print(nulos_reviews)
```

review_comment_title	88.34
review_comment_message	58.70
review_id	0.00
order_id	0.00
review_score	0.00
review_creation_date	0.00
review_answer_timestamp	0.00
dtype: float64	

8 # Remoção dos nulos: verificando se a coluna existe antes de removê-la

```
if 'review_comment_title' in df_reviews.columns:
    df_reviews = df_reviews.drop('review_comment_title', axis=1)

if 'review_comment_message' in df_reviews.columns:
    df_reviews = df_reviews.drop('review_comment_message', axis=1)
```

Este código remove colunas específicas do DataFrame **df_reviews** se essas colunas existirem. O código verifica se as colunas **review_comment_title** e **review_comment_message** existem no DataFrame. Em caso afirmativo, o código remove as colunas usando o método **drop**.

O método **drop** leva dois argumentos:

- **labels**: As colunas a serem removidas.
- **axis**: O eixo ao longo do qual remover as colunas. Em este caso, o eixo é 1, que representa as colunas.

9 # Verificando se a remoção foi bem sucedida

```
df_reviews.isnull().sum()
```

review_id	0
order_id	0
review_score	0

review_creation_date	0
review_answer_timestamp	0
dtype: int64	

10 # Coletando informações sobre as colunas e tipos de dados de dois DataFrames, df_orders e df_products

```
data_tipo = []

# Iterando pelos DataFrames e coletando informações sobre as colunas e tipos de dados
for name, df in zip(names, datasets):
    for column, data_type in zip(df.columns, df.dtypes):
        info = {
            'conjunto_de_dados': name,
            'nome_da_coluna': column,
            'tipo_dos_dados': data_type
        }
        data_tipo.append(info)

# Criando o DataFrame com as informações coletadas
data_tipo = pd.DataFrame(data_tipo)

# Exibindo o DataFrame com informações sobre colunas e tipos de dados
display(data_tipo)
```

Este código coleta informações sobre as colunas e tipos de dados de dois DataFrames, **df_orders** e **df_products**. O código cria uma lista vazia chamada **data_tipo**. Esta lista será usada para armazenar as informações coletadas.
Em seguida, o código itera pelos dois DataFrames, usando um **loop for** aninhado. O primeiro **loop for** itera pelos nomes dos DataFrames, que são armazenados na lista names. O segundo loop for itera pelas colunas e tipos de dados de cada DataFrame.

Para cada coluna e tipo de dados, o código cria um dicionário com as seguintes informações:

- conjunto_de_dados: O nome do DataFrame ao qual a coluna pertence.
- nome_da_coluna: O nome da coluna.
- tipo_dos_dados: O tipo de dados da coluna.

O código adiciona o dicionário à lista data_tipo.

Depois que todas as informações forem coletadas, o código cria um DataFrame a partir da lista data_tipo. O DataFrame é então exibido usando o método display().

A saída do código é a seguinte:

conjunto_de_dados	nome_da_coluna	tipo_dos_dados
df_customer	customer_id	object
df_customer	customer_unique_id	object
df_customer	customer_zip_code_prefix	int64
df_customer	customer_city	object
df_customer	customer_state	object
df_geolocation	geolocation_zip_code_prefix	int64
df_geolocation	geolocation_lat	float64
df_geolocation	geolocation_lng	float64
df_geolocation	geolocation_city	object
df_geolocation	geolocation_state	object
df_orders	order_id	object
df_orders	customer_id	object
df_orders	order_status	object
df_orders	order_purchase_timestamp	object
df_orders	order_approved_at	object
df_orders	order_delivered_carrier_date	object
df_orders	order_delivered_customer_date	object
df_orders	order_estimated_delivery_date	object
df_item	order_id	object
df_item	order_item_id	int64
df_item	product_id	object
df_item	seller_id	object
df_item	shipping_limit_date	object
df_item	Price	float64
df_item	freight_value	float64
df_order_pay	order_id	object
df_order_pay	payment_sequential	int64
df_order_pay	payment_type	object
df_order_pay	payment_installments	int64
df_order_pay	payment_value	float64
df_reviews	review_id	object
df_reviews	order_id	object
df_reviews	review_score	int64
df_reviews	review_comment_title	object
df_reviews	review_comment_message	object
df_reviews	review_creation_date	object
df_reviews	review_answer_timestamp	object
df_products	product_id	object
df_products	product_category_name	object
df_products	product_name_lenght	float64
df_products	product_description_lenght	float64
df_products	product_photos_qty	float64
df_products	product_weight_g	float64
df_products	product_length_cm	float64
df_products	product_height_cm	float64
df_products	product_width_cm	float64
df_sellers	seller_id	object

df_sellers	seller_zip_code_prefix	int64

11: Conversão do tipo de dado de object para datetime

```
coluna_datas    =    ['order_purchase_timestamp',    'order_approved_at',
'order_delivered_carrier_date',
                                          'order_estimated_delivery_date',
'order_delivered_customer_date']

print(df_orders[coluna_datas].dtypes)

# Aplicar a conversão para datetime
for colunas in coluna_datas:
    df_orders[colunas] = pd.to_datetime(df_orders[colunas])

df_item['shipping_limit_date']                                          =
pd.to_datetime(df_item['shipping_limit_date'])


# Após a conversão
print(df_orders[coluna_datas].dtypes)
```

Este código converte várias colunas de data em DataFrames para o tipo de dados **datetime**.

Passo a passo:

1. Definição da lista de colunas de data: Primeiro, o código define uma lista de strings chamada **coluna_datas**. Esta lista contém os nomes das colunas que o código irá converter para o tipo de dados **datetime**.
2. Verificação dos tipos de dados atuais: O código então usa o método **dtypes** para imprimir os tipos de dados atuais das colunas listadas em **coluna_datas**.
3. Conversão de colunas para **datetime**: Em seguida, o código itera sobre a lista **coluna_datas** usando um loop **for**. Dentro do loop, o código usa o método **to_datetime** do pandas para converter cada coluna para o tipo de dados **datetime**.
4. Conversão da coluna **shipping_limit_date**: O código então converte a coluna **shipping_limit_date** no DataFrame **df_item** para o tipo de dados **datetime** usando o mesmo método **to_datetime**.
5. Verificação dos tipos de dados após a conversão: Finalmente, o código usa novamente o método **dtypes** para imprimir os tipos de dados das colunas listadas em **coluna_datas** após a conversão.

Benefícios da conversão de colunas de data para **datetime**:

- Permitir cálculos com datas e horários.
- Facilitar a filtragem e agregação de dados com base em datas e horários.
- Melhorar a visualização de dados de data e hora.
- Respeita as boas práticas

Observações:

- O código assume que as colunas contêm valores de data e hora válidos. Se as colunas contiverem valores inválidos, o código pode gerar erros.
- O código não especifica nenhuma formatação de data e hora. Se a formatação for inconsistente, a conversão pode não ser bem-sucedida.

Antes da conversão:

order_purchase_timestamp	object
order_approved_at	object
order_delivered_carrier_date	object
order_estimated_delivery_date	object
order_delivered_customer_date	object

dtype: object

Após a conversão:

order_purchase_timestamp	datetime64[ns]
order_approved_at	datetime64[ns]
order_delivered_carrier_date	datetime64[ns]
order_estimated_delivery_date	datetime64[ns]
order_delivered_customer_date	datetime64[ns]

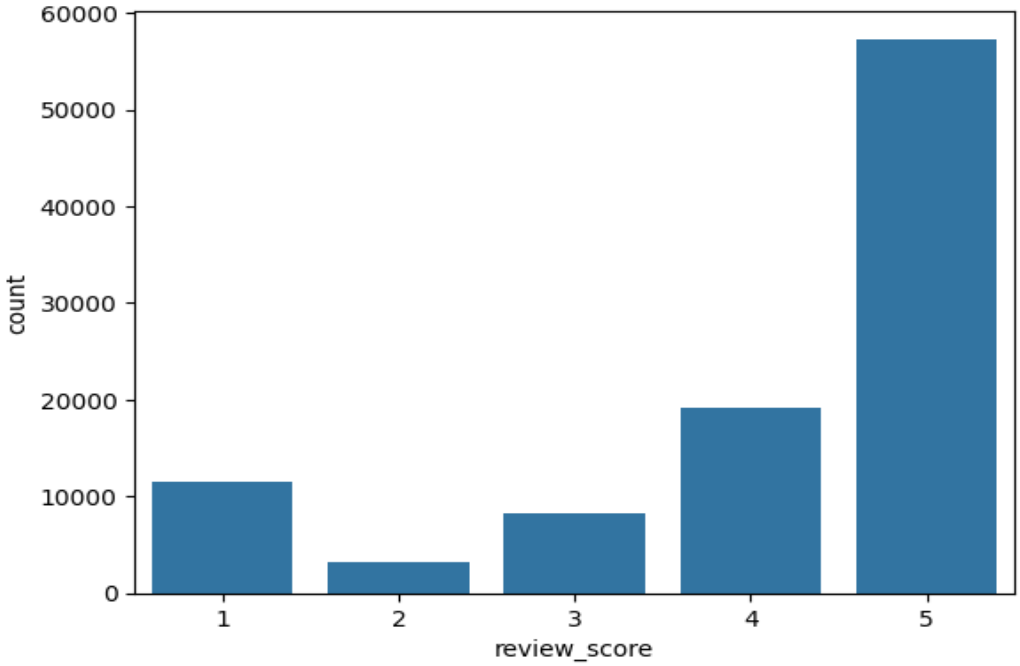
dtype: object

12: Criação de Gráfico das Avaliações

```
sns.countplot(x= 'review_score', data = df_reviews)
```

Este código cria um gráfico de barras que mostra a distribuição das notas das avaliações no DataFrame **df_reviews**

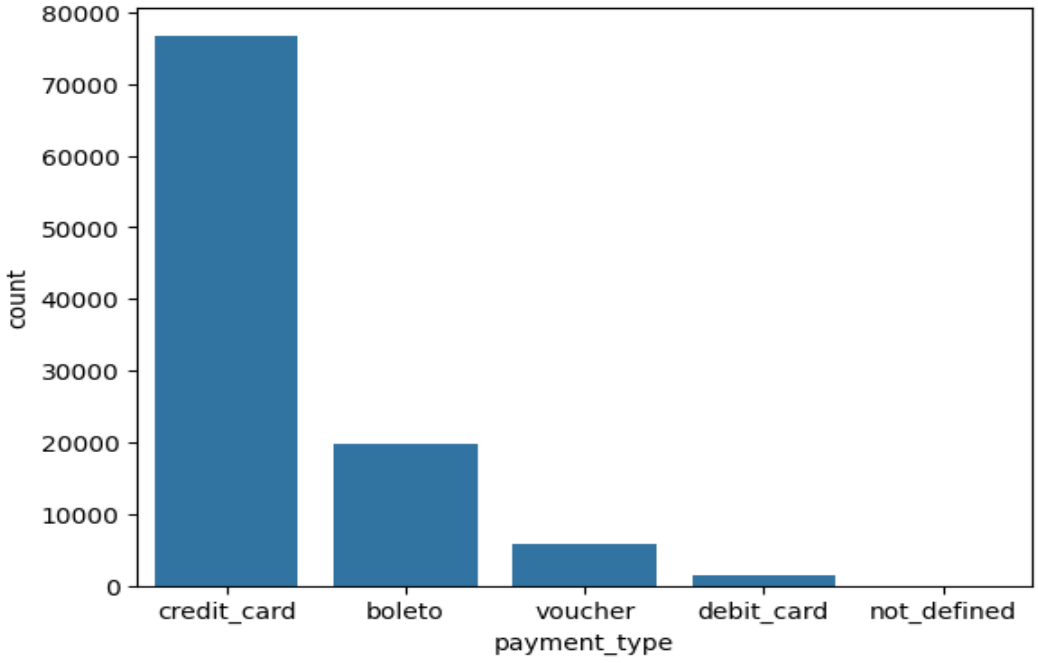
O gráfico gerado fornece uma visão geral rápida da distribuição das notas das avaliações dos usuários no DataFrame. O gráfico pode ser usado para identificar quaisquer tendências ou padrões na distribuição das pontuações.



13: # Criação de gráfico da distribuição das formas de pagamento dos pedidos

```
sns.countplot(x= 'payment_type', data = df_order_pay)
```

O código **sns.countplot(x= 'payment_type', data = df_order_pay)** cria um gráfico de barras que mostra a distribuição dos tipos de pagamento usados para as compras no e-commerce analisado, do DataFrame **df_order_pay**.



- A maior parte dos clientes preferem usar cartão de crédito para fazer suas compras. Isso pode se atribuir, mas não se limitando, à conveniência e segurança que o cartão de crédito oferece.
- O boleto bancário, segunda maior forma de pagamento escolhida entre o conjunto analisado, é um tipo de pagamento popular geralmente entre os clientes que não têm cartão de crédito ou não se sentem seguros informando os dados do cartão.
- Os vouchers são uma forma de pagamento que podem ser relacionados a compras promocionais ou de prováveis créditos de devoluções/reembolsos.
- Os tipos de pagamento débito e indefinido são menos comuns. Em relação ao cartão de débito, possíveis causas para sua menor predileção são (mas não se limitam):
    1. Facilidade de uso – em relação ao cartão e crédito, os sistemas de pagamento costumam solicitar autenticações adicionais para uso do cartão de débito.
    2. Recompensas e benefícios – é comum que cartões de crédito possuam programas de benefícios relacionados ao uso, o que é menor comum para cartões de débito.
    3. Indisponibilidade do uso – Muitos ecommerces não permitem o pagamento por meio e cartões de débito.

A Febraban (Federação Brasileira de Bancos) divulgou um ranking com as principais formas de pagamento do consumidor brasileiro. Conforme registra:

Análise dos meios de pagamento - 2022			
Número de transações		**Valores transacionados**	
Pix	24,1 bilhões	**TED**	R$ 40,7 trilhões
Cartão de crédito	18,2 bilhões	**Pix**	R$ 10,9 trilhões
Cartão de débito	15,6 bilhões	**Boleto**	R$ 5,3 trilhões
Boleto	4,03 bilhões	**Cartão de crédito**	R$ 2,09 trilhões
TED	1,01 bilhão	**Cartão de débito**	R$ 992 bilhões
Cheques	202,8 milhões	**Cheques**	R$ 666,8 bilhões
DOC	59 milhões	**DOC**	R$ 55,7 bilhões

Os dados referem-se o ano de 2022 e serve como referência posterior dos movimentos e preferências de compra do consumidor percebidos nos registros da Olist.

O gráfico de barras fornece uma visão geral útil da distribuição dos tipos de pagamento usados para as compras no DataFrame. Pode ser usado para identificar quaisquer tendências ou padrões na distribuição dos tipos de pagamento, o que pode ser um dado facilitador na tomada de decisões sobre como melhorar a experiência de pagamento para os clientes.

14 # Fatiando a distribuição dos pedidos por periodicidade

```
# Atributos da data da compra - Ano e Mês
df_orders['ano_da_compra'] = df_orders['order_purchase_timestamp'].dt.year
df_orders['mes_da_compra']                                              =
df_orders['order_purchase_timestamp'].dt.month
df_orders['nome_mes_da_compra']                                         =
df_orders['order_purchase_timestamp'].dt.strftime('%b')
df_orders['ano_e_mes_da_compra']                                        =
df_orders['order_purchase_timestamp'].dt.strftime('%Y%m')
df_orders['data_da_compra']                                             =
df_orders['order_purchase_timestamp'].dt.strftime('%Y%m%d')

# Atributos da data da compra - Dia e Dia da Semana
df_orders['dia_da_compra'] = df_orders['order_purchase_timestamp'].dt.day
df_orders['dia_da_semana_compra']                                       =
df_orders['order_purchase_timestamp'].dt.dayofweek
df_orders['nome_dia_compra']                                            =
df_orders['order_purchase_timestamp'].dt.strftime('%a')

# Atributos da data da compra - Hora e Período do Dia
df_orders['hora_da_compra']                                             =
df_orders['order_purchase_timestamp'].dt.hour
hours_bins = [-0.1, 6, 12, 18, 23]
hours_labels = ['Madrugada', 'Manhã', 'Tarde', 'Noite']
df_orders['periodo_da_compra']    =    pd.cut(df_orders['hora_da_compra'],
hours_bins, labels=hours_labels)

# Visualizar o novo DataFrame após as transformações
df_orders.head()
```

O código acima cria novos atributos no DataFrame **df_orders** para representar informações sobre a data e hora da compra. Os atributos são criados usando os métodos **dt.year**, **dt.month**, **dt.day**, **dt.dayofweek**, **dt.strftime**, **dt.hour** e **pd.cut** do pandas.

O código também imprime o cabeçalho do novo DataFrame para visualizar os novos atributos.

Aqui está uma explicação mais detalhada de como cada atributo é criado:

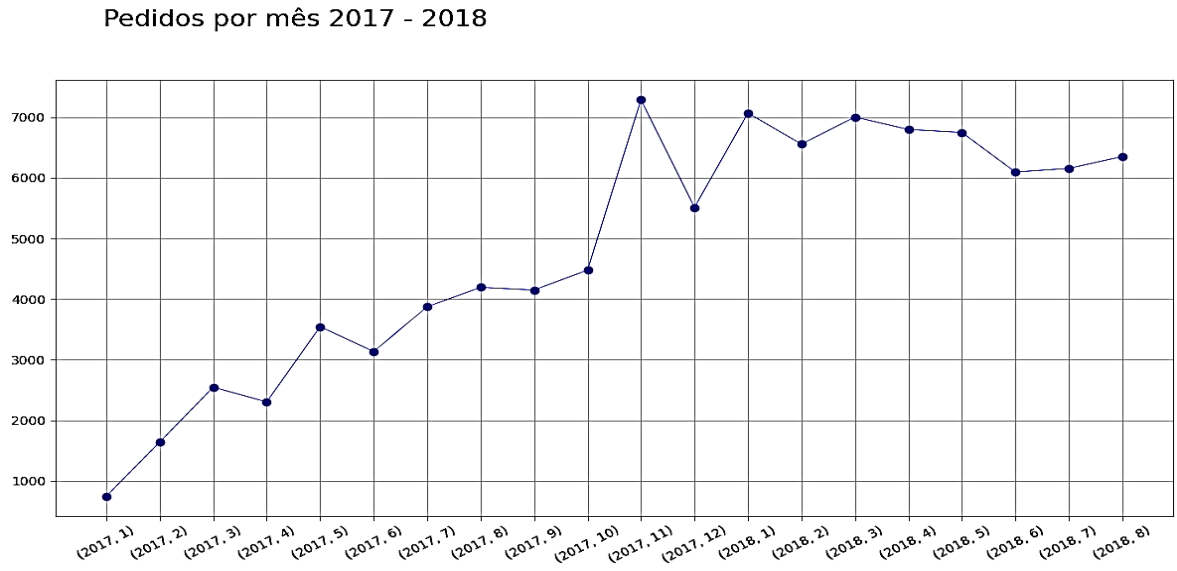
- **ano_da_compra**: Usa o método **dt.year** para extrair o ano da data de compra.
- **mes_da_compra**: Usa o método **dt.month** para extrair o mês da data de compra.
- **nome_mes_da_compra**: Usa o método **dt.strftime** para converter o mês da data de compra em um nome.
- **ano_e_mes_da_compra**: Usa o método **dt.strftime** para combinar o ano e o mês da data de compra em um único formato.
- **data_da_compra**: Usa o método **dt.strftime** para converter a data completa da compra em um formato específico.
- **dia_da_compra**: Usa o método **dt.day** para extrair o dia da data de compra.
- **dia_da_semana_compra**: Usa o método **dt.dayofweek** para extrair o dia da semana da data de compra.
- **nome_dia_compra**: Usa o método **dt.strftime** para converter o dia da semana da data de compra em um nome.
- **hora_da_compra**: Usa o método **dt.hour** para extrair a hora da data de compra.
- **periodo_da_compra**: Usa o método **pd.cut** para dividir as horas do dia em quatro períodos (madrugada, manhã, tarde e noite).

Esses novos atributos podem ser usados para realizar análises mais detalhadas sobre as compras. Por exemplo, podemos usar os atributos **ano_da_compra** e **mes_da_compra** para analisar as vendas por mês ou ano. Podemos usar os atributos **dia_da_semana_compra** e **nome_dia_compra** para analisar as vendas por dia da semana ou dia da semana. Podemos usar o atributo **periodo_da_compra** para analisar as vendas por período do dia.

15 # Criação de gráfico para representar os dados acima descritos

```
pedidos_mes = df_orders[df_orders['ano_da_compra'] != 2016].groupby(['ano_da_compra', 'mes_da_compra'])['order_id'].nunique()

fig, ax = plt.subplots(figsize=(12, 6))
line = ax.plot([str(idx) for idx in pedidos_mes.index], pedidos_mes, linewidth=0.5, color='blue', marker='o')

fig.text(0.09, 1.08, 'Pedidos por mês 2017 - 2018 ', fontsize=19)
ax.grid(True)

plt.xticks(rotation=30)
plt.tight_layout()
plt.show()
```

O código cria um gráfico de linha(abaixo) que mostra o número de pedidos por mês entre 2017 e 2018.



O gráfico mostra que o número de pedidos aumentou gradualmente de 2017 para 2018. O mês de novembro de 2017 teve o maior número de pedidos, seguido de janeiro de 2018 e março de 2018.

Aqui está uma análise mais detalhada do gráfico:

- O número de pedidos aumentou de 2017 para 2018. Isso pode ser devido a uma série de fatores, como o crescimento do mercado, o lançamento de novos produtos ou serviços, ou campanhas de marketing bem-sucedidas.

De acordo com a 36º edição do relatório Webshoppers, publicada pelo Ebit em 2017, o volume de pedidos online **cresceu 3,9%** em comparação aos anos de 2015 e 2016.

A mesma pesquisa aponta que o ano de 2017 teve início com grande aquecimento da economia e crescimento de **0,2% do PIB** (Produto Interno Bruto), segundo o **IBGE**.

Isso teve reflexo positivo nos e-commerces de forma geral, que apresentaram **crescimento de 10,3%** em relação ao ano anterior.

A pesquisa traz um dado interessante a respeito de momentos que ajudaram a alavancar os resultados de vendas nos e-commerces.

- O dia das mães movimentou **R$ 1,9 bilhão**, que representa um **crescimento de 16%** em comparação ao mesmo período do ano anterior (R$ 1,62 bilhão)

- O Dia dos Namorados de 2017 gerou um faturamento de R$ 1,71 bilhão, representando aumento de 5,1%.

- O Dia dos Pais 2017, movimentou R$ 1,94 bilhão no e-commerce, crescendo 10,1% em comparação ao ano anterior. Segundo Pedro Guasti, “por ser a primeira do segundo semestre, a data é um excelente termômetro para as vendas no restante do ano.

- O mês de novembro de 2017 teve o maior número de pedidos, seguido de janeiro de 2018 e março de 2018. Isso pode ser devido a fatores como as festas de fim de ano, o Black Friday e o Cyber Monday e liquidações de início de ano.

Segundo o Ebit, a Black Friday de 2017 gerou um **faturamento foi de R$ 2,1 bilhões**, crescendo **10,3%** em comparação aos R$ 1,9 bilhões do ano passado. O número de **pedidos** aumentou **14%**, indo **de 3,30 milhões** para **3,76 milhões**.

Em compensação, o ticket médio **caiu 3,1%**, de **R$ 580** para **R$ 562**. Essa queda pode ser explicada pelas ações promocionais mais agressivas na Black Friday, mais precisamente em categorias de produtos com maior valor agregado, que são as mais consumidas no comércio eletrônico.

Para Guasti, o maior destaque foi o grande aumento no volume de pedidos, que foi quase o dobro estimado pelo Ebit. Segundo ele, “ao contrário das duas Black Friday´s anteriores, que foram pautadas pelo crescimento no ticket médio, neste ano o grande vetor do crescimento foi no número de pedidos. Lojistas de todos os segmentos investiram em promoções na Black Friday com descontos reais, e isso atraiu o consumidor”.

- Voltando aos dados da base, os meses de junho, julho de 2018 tiveram uma queda maior no número de pedidos, voltando a crescer em agosto de 2018

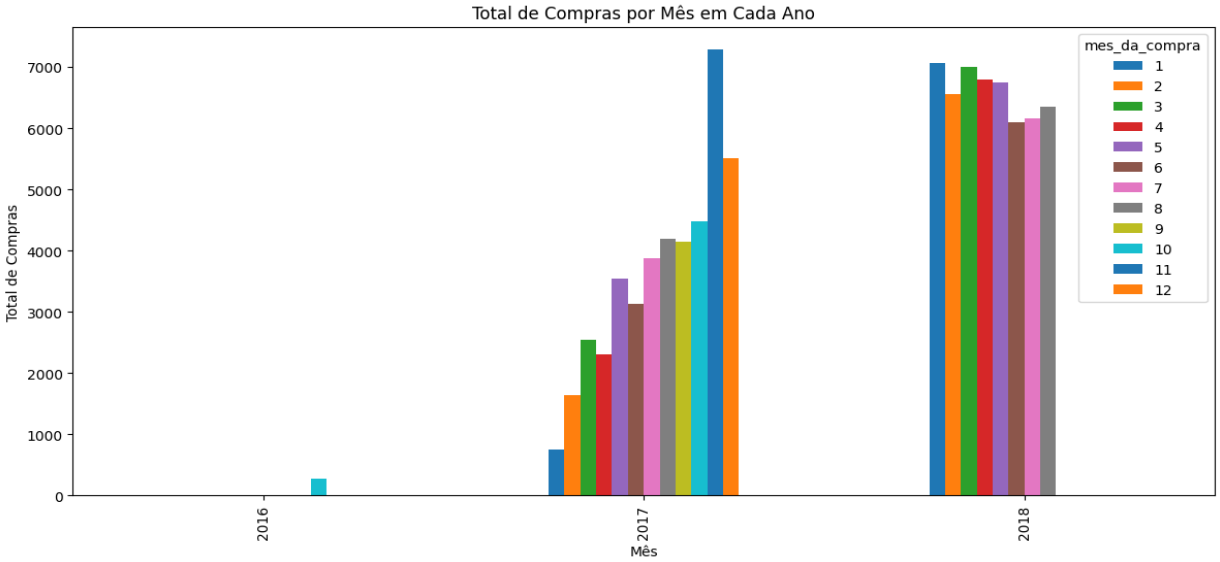
O gráfico acima pode ser usado para entender melhor o comportamento de compra dos clientes. Por exemplo, o gráfico pode ser usado para identificar os meses mais populares para compras, os produtos ou serviços que são mais populares em determinados meses, ou os fatores que podem estar influenciando o número de pedidos.

16 # Calculando total de compras por ano, mês e dia

```
# Total de compras por ano, mês e dia
total_compras_por_ano                                                =
df_orders['ano_da_compra'].value_counts().sort_index()
total_compras_por_mes         =         df_orders.groupby(['ano_da_compra',
'mes_da_compra']).size().unstack().fillna(0)
total_compras_por_dia                                                =
df_orders['data_da_compra'].value_counts().sort_index()

# Visualização - Total de compras por mês em cada ano
total_compras_por_mes.plot(kind='bar', figsize=(12, 6))
plt.xlabel('Mês')
plt.ylabel('Total de Compras')
plt.title('Total de Compras por Mês em Cada Ano')
plt.tight_layout()
plt.show()
```

O código acima calcula o total de compras por ano, mês e dia e, em seguida, cria uma visualização para mostrar o total de compras por mês em cada ano.

O gráfico de barras abaixo mostra o total de compras por mês em cada ano. O gráfico mostra que o número de compras aumentou de 2017 para 2018.



17 # Definindo uma métrica para identificar um possível outlier

```
limite_outlier = total_compras_por_dia.mean() + 2 *
total_compras_por_dia.std()

# Visualização destacando possíveis outliers
plt.figure(figsize=(12, 6))
plt.bar(total_compras_por_dia.index, total_compras_por_dia.values,
color='skyblue')
plt.xlabel('Dia')
plt.ylabel('Total de Compras')
plt.title('Total de Compras por Dia')

# Destacar possíveis outliers
outliers = total_compras_por_dia[total_compras_por_dia > limite_outlier]
plt.bar(outliers.index, outliers.values, color='red', label='Possível
Outlier')
plt.legend()
plt.grid(axis='y')
plt.show()

# Imprimir os dias possíveis outliers
print("Possíveis outliers:")
print(outliers)
```

O código acima define uma métrica para identificar um possível outlier, cria uma visualização destacando possíveis outliers e imprime as datas onde possíveis outliers foram percebidos.

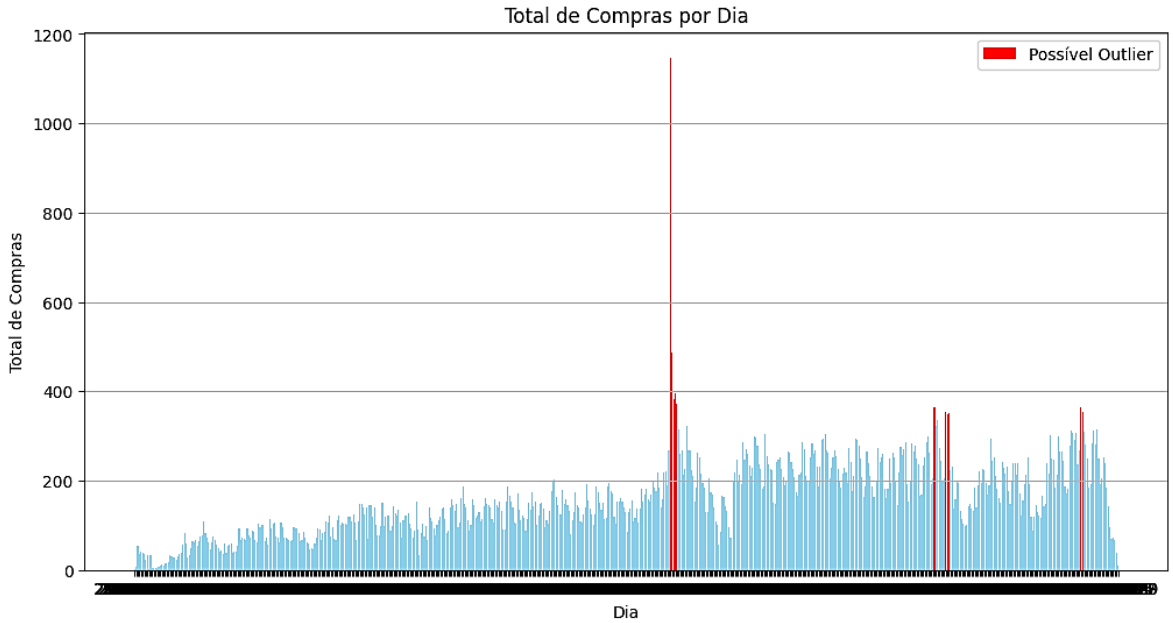
O primeiro bloco de código define uma métrica utilizada, que neste caso foi, o uso da média mais duas vezes o desvio padrão.

O segundo bloco de código cria uma visualização para destacar os dados coletados. O código usa a função **bar()** para plotar os dados do DataFrame **total_compras_por_dia**. O código também usa a função **legend()** para adicionar uma legenda ao gráfico.

O terceiro bloco de código destaca possíveis outliers. O código usa a função **[>]** para selecionar os valores do DataFrame **total_compras_por_dia** que são maiores que o limite de outlier.

O quarto bloco de código imprime os dias possíveis outliers. O código usa a função **print()** para imprimir o DataFrame **outliers**.

A saída do código é o gráfico de barras abaixo que mostra o total de compras por dia. O gráfico mostra que existem dias com valores muito acima do padrão de vendas. Esses dias podem se tratar de outliers.



Possíveis outliers:

data_da_compra

2017.11.24 1147

2017.11.25 487

2017.11.26 382

2017.11.27 395

2017.11.28 371

2018.05.07 363

2018.05.14 355

2018.05.15 348

2018.05.16 351

2018.08.06 363

2018.08.07 353

Name: count, dtype: int64

Aqui estão algumas hipóteses sobre o que pode ter ocorrido nestas datas em relação ao e-commerce no Brasil que gerou estes possíveis outliers:

- Promoções ou descontos especiais: É possível que estas datas tenham sido marcadas por promoções ou descontos especiais que tenham atraído um grande número de compradores. Por exemplo, as datas de 2017-11-24 a 2017-11-28 podem ter sido relacionadas à Black Friday, uma das maiores datas de compras do ano.
- Campanhas de marketing bem-sucedidas: Outra possibilidade é que estas datas tenham sido marcadas por campanhas de marketing bem-sucedidas que tenham alcançado um grande número de pessoas. Por exemplo, as datas de 2018-05-07 a 2018-05-16 podem ter sido relacionadas a campanhas de Dia das Mães ou Dia dos Pais.
- Eventos esportivos ou culturais: É possível que estas datas tenham sido marcadas por eventos esportivos ou culturais que tenham atraído um grande número de pessoas, aumentando a demanda por produtos e serviços. Por exemplo, as datas de 2018-08-06 e 2018-08-07 podem ter sido relacionadas às Olimpíadas de 2016, que aconteceram no Rio de Janeiro.

É importante ressaltar que estas são apenas hipóteses. Para confirmar ou refutar essas hipóteses, seria necessário analisar mais dados, como os produtos ou serviços que foram comprados nessas datas, o perfil dos compradores e as campanhas de marketing que foram realizadas.

Aqui estão algumas sugestões para analisar esses dados:

- Analisar os produtos ou serviços que foram comprados nessas datas: Isso pode ajudar a identificar se as promoções ou descontos foram direcionados a produtos ou serviços específicos.
- Analisar o perfil dos compradores: Isso pode ajudar a identificar se as campanhas de marketing foram direcionadas a um determinado público-alvo.
- Analisar as campanhas de marketing que foram realizadas: Isso pode ajudar a identificar se as campanhas foram bem-sucedidas em alcançar o público-alvo.

18 # Distribuição de compras ao longo do dia

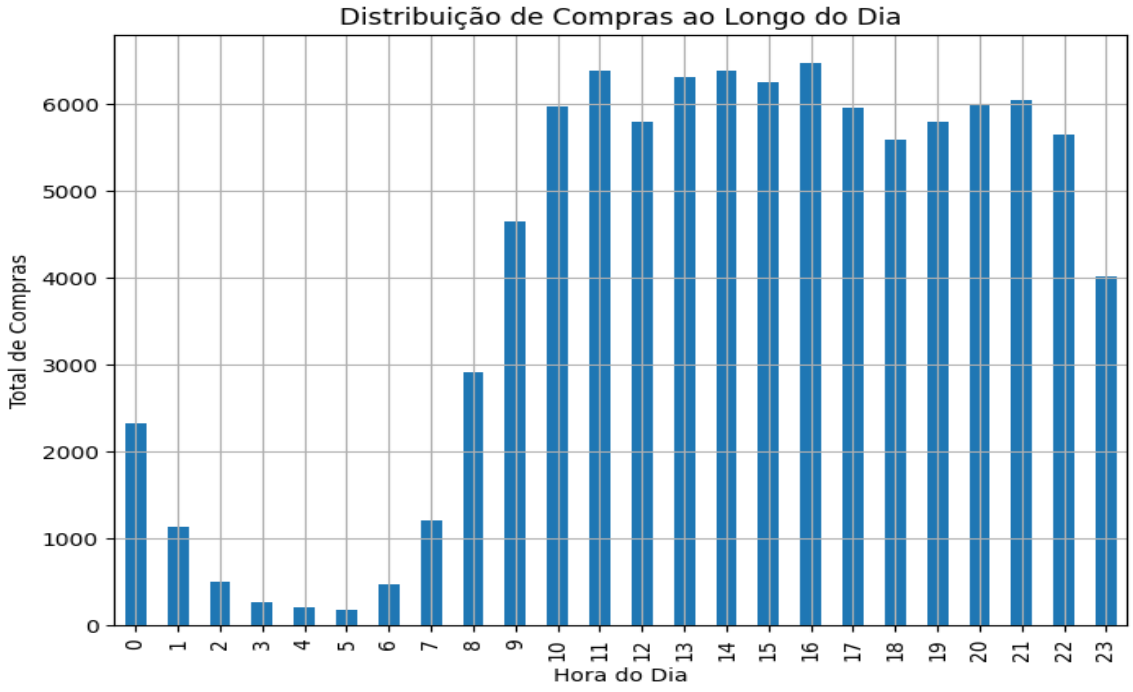
```
compras_por_hora = df_orders['hora_da_compra'].value_counts().sort_index()

# Visualização
plt.figure(figsize=(8, 6))
compras_por_hora.plot(kind='bar')
plt.xlabel('Hora do Dia')
plt.ylabel('Total de Compras')
plt.title('Distribuição de Compras ao Longo do Dia')
plt.grid(True)
plt.show()
```

O código acima calcula a distribuição de compras ao longo do dia e, em seguida, cria uma visualização para mostrar os resultados.

O primeiro bloco de código usa a função **value_counts()** para contar o número de vezes que cada valor aparece na coluna **hora_da_compra** do DataFrame **df_orders**.

O segundo bloco de código cria uma visualização usando a biblioteca **matplotlib**. A função **plot()** é usada para plotar, ou seja, gerar uma visualização dos dados.

A saída do código é um gráfico de barras que mostra a distribuição de compras ao longo do dia. O gráfico mostra que o maior número de compras ocorre no período da tarde, entre as 14h e as 20h. O menor número de compras ocorre no período da madrugada, entre as 4h e as 8h.



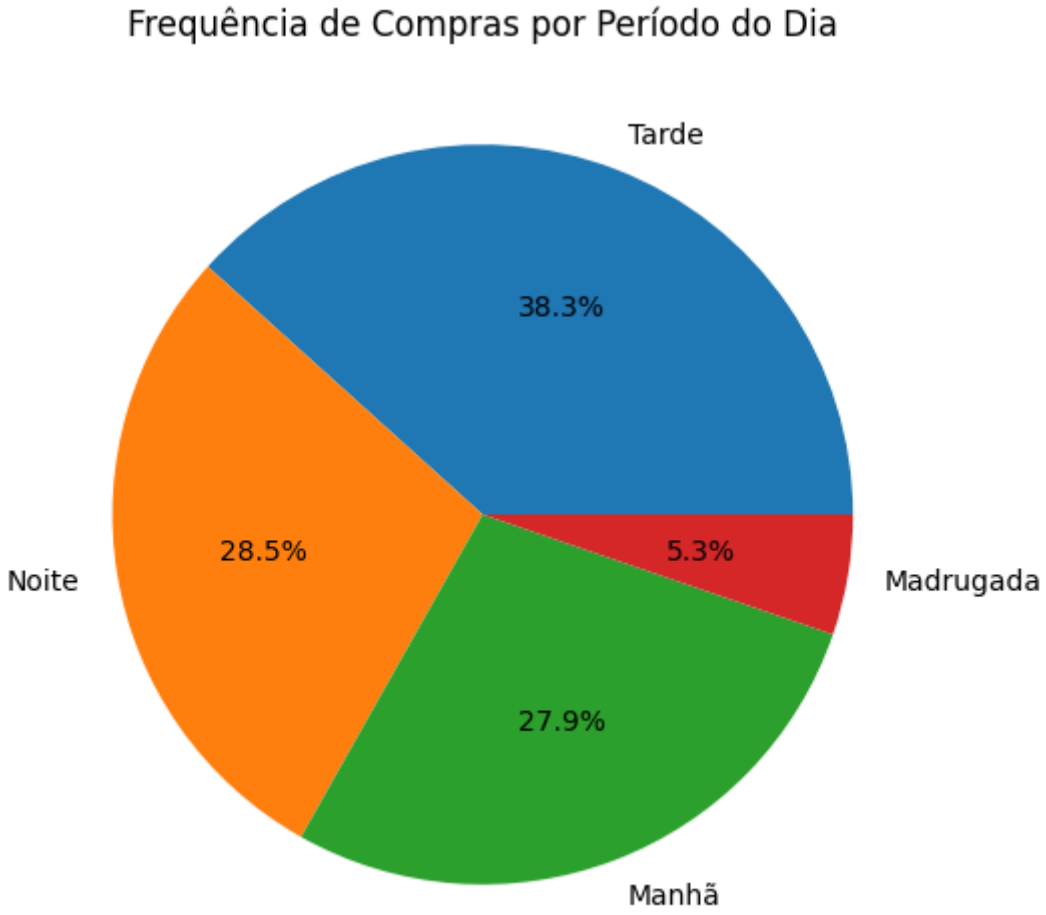
19 # Visualização da Frequência de compras por período do dia

```
frequencia_por_periodo = df_orders['periodo_da_compra'].value_counts()
# Visualização
plt.figure(figsize=(8, 6))
frequencia_por_periodo.plot(kind='pie', autopct='%1.1f%%')
plt.ylabel('')
plt.title('Frequência de Compras por Período do Dia')
plt.show()
```

O código acima calcula a frequência de compras por período do dia e, em seguida, cria uma visualização em gráfico de pizza para mostrar os resultados.

O primeiro bloco de código usa a função **value_counts()** para contar o número de vezes que cada valor aparece na coluna **periodo_da_compra** do DataFrame **df_orders**.

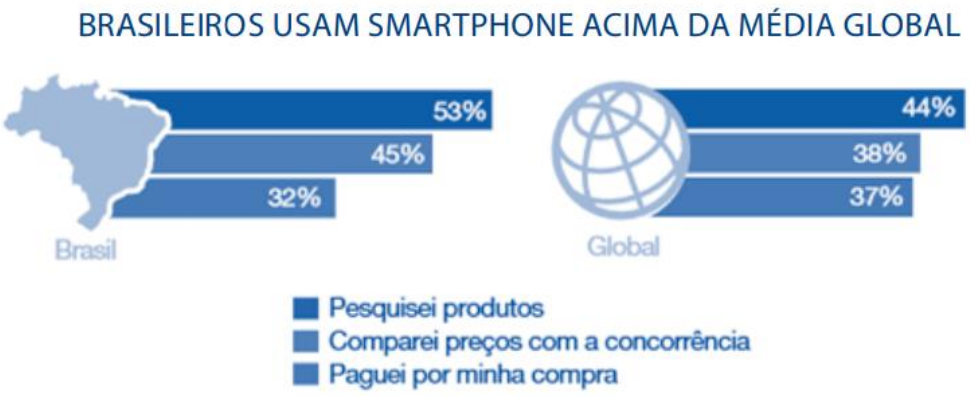
O segundo bloco de código cria uma visualização usando a biblioteca **matplotlib**. A função **plot()** é usada para plotar os dados. No caso específico, estamos usando um gráfico de pizza para plotar os dados do DataFrame **frequencia_por_periodo**.



A SBVC (Sociedade Brasileira de Varejo e Consumo) gerou em 2017 o ranking das 70 Maiores Empresas de E-commerce brasileiro, trazendo diversos dados relevantes sobre

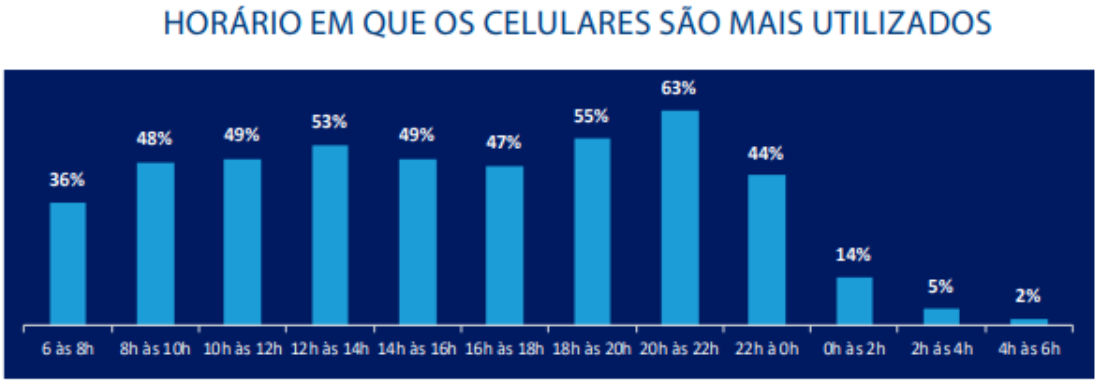
o perfil de compra do consumidor brasileiro, análise do poder de compra, situação do varejo, entre outras informações.

Os dados da base acompanham a tendência de mercado, o que pode ser percebido ao analisar a pesquisa, em sua página 12. Veja abaixo alguns dados relevantes que o levantamento gerado nos traz:



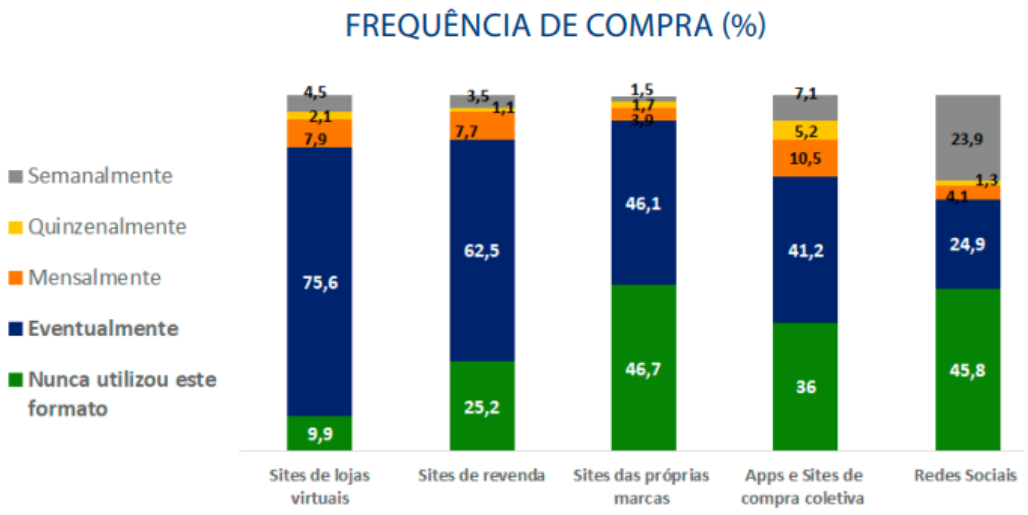
Fonte: PwC, Pesquisa Total Retail 2017

A pesquisa aponta: em sua Conforme o consumidor aprofunda seu rela - cionamento com as empresas do setor e percebe que suas marcas de preferência no varejo físico também estão disponíveis online, oferecendo experiências de consumo confiáveis, aumenta sua disposição em usar canais digitais de compra, o que cria novas oportu - nidades de relacionamento para as marcas nascidas no varejo físico, como também abre a possibilidade quanto internacionais (no Brasil, em 2016 um total de 21,2 milhões de consumidores gastou US$ 2,4 bilhões em compras em sites internacionais, segundo a Ebit). O fato de que hoje o e-commerce representa cerca de 8,1% das vendas do varejo online americano e apenas 2,5% no Brasil (segundo a Statista, com base em números dos órgãos oficiais de estatísticas de cada país) não reflete a relevância dos canais digitais no processo de compra. A jornada de consumo dos clientes inclui o digital, em algum ponto das etapas de pesquisa, consulta e escolha de produtos, em pratica - mente 100% dos casos. A pesquisa Total Retail 2017, produzida pela PwC, mostra que 53% dos brasileiros utilizam seus smartphones para pesquisar produtos, 45% compararam preços com a concorrência por meio do dispositivo móvel e 32% pagaram por suas compras. O brasileiro pesquisa e compara pelo celular mais que a média mundial, mesmo que não feche a compra por meios digitais. O fato é que os canais di - gitais se tornaram um poderoso auxílio no processo de compras dos consumidores.



Fonte: IBOPE Nielsen

E segue: O estudo “Varejo no Brasil: a influência do digital sobre o consumo”, realizado pelo Boston Consulting Group (BCG), indica que a quase totalidade dos 106 milhões de internautas usam a internet em alguma parte de seu processo de compra, mesmo que não seja especificamente para a realização da transação financeira. O maior impacto ocorre na fase de pré-compra, uma vez que de 60% a 70% das pesquisas em sites, redes sociais e ferramentas de busca dizem respeito à localização de marcas, produtos, serviços e lojas online. O smartphone, mais especificamente, é o principal meio de contato dos clientes com as marcas. Segundo o IBOPE Nielsen, os horários em que os celulares são mais utilizados são o almoço (53%), a saída do trabalho (55%) e após chegar em casa (63%), embora se mantenha relativamente alto durante todo o dia, demonstrando que o celular está na vida dos usuários o tempo todo e torna-se, assim, um canal relevante de contato com marcas, produtos e serviços.

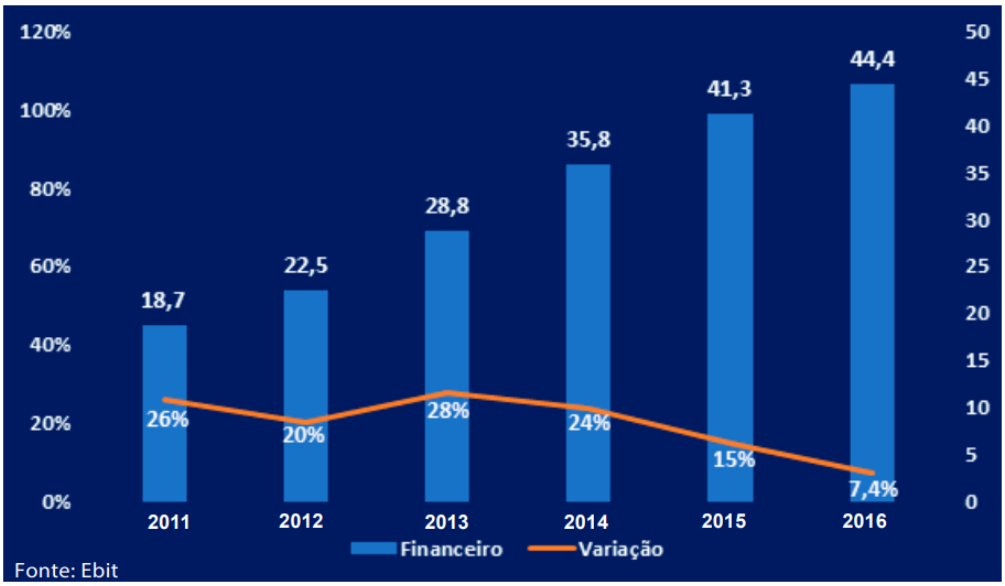


O estudo Omnishopper, desenvolvido pela SBVC e pela AGP Pesquisas para quantificar aspectos relacionados aos hábitos de compra online da população, barreiras e aspectos mais valorizados, mostra que a maioria dos consumidores entrevistados compra online por meio de varejistas online (apenas 9,9% dos pesquisados nunca usaram esse formato), mas que 46,7% nunca experimentaram os sites das marcas da indústria e 45,8% nunca compraram a partir das mídias sociais, mostrando que o perfil de consumo

online está bastante concentrado nas lojas virtuais. Segundo o estudo, porém, o uso das mídias sociais como canal de compra é mais relevante para os Millennials (31% deles afirmam consumir eventualmente por esse canal, contra 24,9% da média), enquanto os e- -commerces convencionais são preferidos pela geração X, o que pode ser um indicativo do caminho que o varejo online precisará percorrer nos próximos anos para se manter relevante junto à população mais jovem (que tem como característica um uso intenso das mídias sociais como canal de comunicação, relacionamento e, cada vez mais, consumo). O estudo também revelou que 85% dos consumidores percebe a loja virtual como mais barata, mas 35,5% acreditam que as marcas deveriam praticar o mesmo preço independente do canal de compra. Para uma parcela importante dos consumidores, o que importa é a marca, não o canal. Além disso, a possibilidade de trocar nas lojas físicas produtos comprados online é visto como um atributo importante para quase 70% dos consumidores entrevistados, reforçando a visão omnichannel que os consumidores já têm e que o varejo ainda busca implementar.

Em 2016, o comércio eletrônico brasileiro teve, pela primeira vez, um crescimento de apenas um dígito em suas vendas. De acordo com o 35º relatório Webshoppers, da Ebit, as vendas online somaram R$ 44,4 bilhões no ano passado, com um crescimento de 7,4% em termos nominais. Embora inferior ao ritmo de crescimento dos anos anteriores, o resultado foi bastante superior ao desempenho do varejo como um todo, que apresentou queda de 6,2%, segundo o IBGE.

TOTAL DE FATURAMENTO NO E-COMMERCE (EM R$ BILHÕES)



A pesquisa completa pode ser consultada no site oficial da SBVC.

Este tipo de dado pode ser utilizado para entender melhor o comportamento de compra dos seus clientes. Por exemplo, sabendo que a maioria das compras ocorre no período da tarde, é possível:

- Alocar mais recursos para atendimento ao cliente durante esse período.
- Enviar promoções e descontos durante esse período.
- Recomendar produtos e serviços que são mais populares nesse período.

Além disso, este tipo de dado pode ser utilizado para melhorar a experiência de compra dos seus clientes. Sabendo que a maioria das compras ocorre no período da tarde, melhorias podem ser implantadas, tais como:

- Oferecer um processo de checkout mais rápido e fácil durante esse período.
- Oferecer opções de entrega mais rápidas durante esse período.

Em resumo, saber o período do dia em que as compras ocorrem pode ajudar a:

- Melhorar a eficiência do atendimento ao cliente.
- Aumentar as vendas.
- Melhorar a experiência de compra dos clientes.

20 # Gerando o Total de vendas por dia da semana e em ordem correta

```
ordem_dias_semana = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
total_vendas_por_dia_semana                                          =
df_orders['nome_dia_compra'].value_counts().reindex(ordem_dias_semana)

# Visualização - Total de vendas por dia da semana
plt.figure(figsize=(8, 6))
total_vendas_por_dia_semana.plot(kind='bar', color='skyblue')
plt.xlabel('Dia da Semana')
plt.ylabel('Total de Vendas')
plt.title('Total de Vendas por Dia da Semana')
plt.grid(axis='y')
plt.show()
```

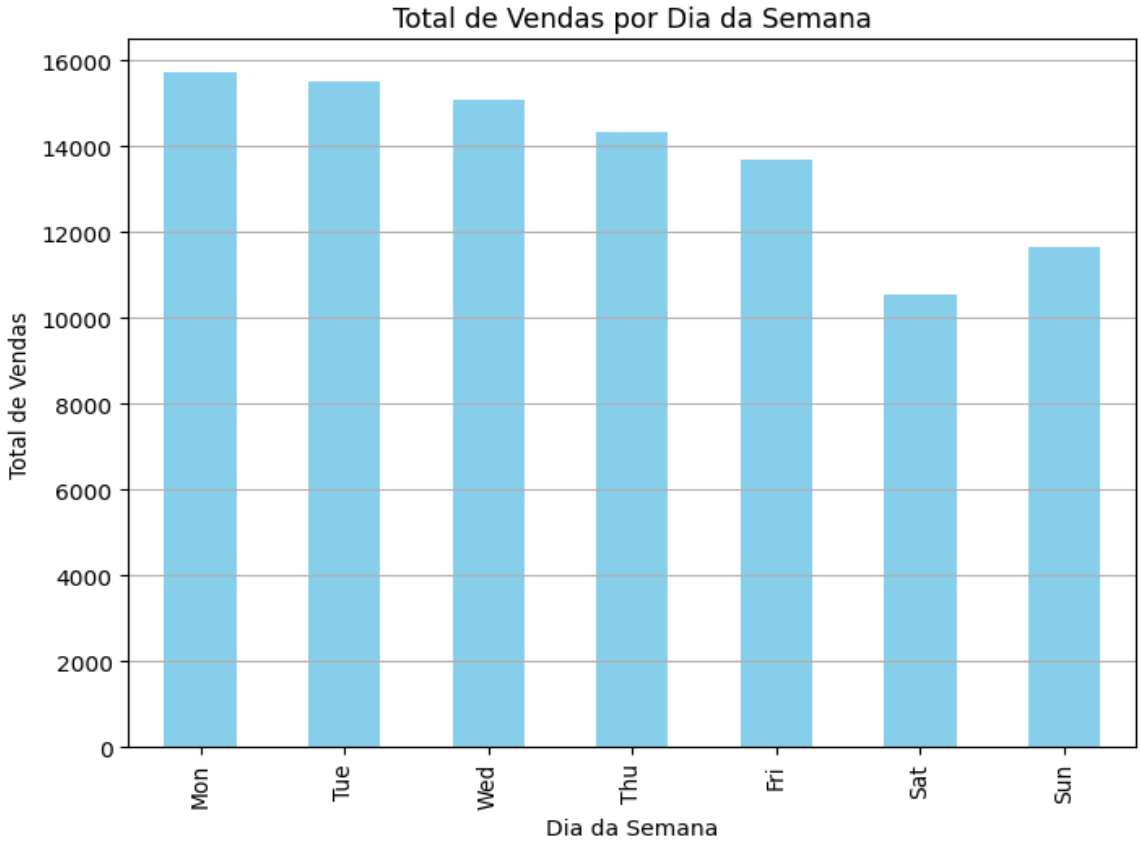
O código acima calcula o total de vendas por dia da semana e, em seguida, cria uma visualização para mostrar os resultados.

O primeiro bloco de código usa a função **value_counts()** para contar o número de vezes que cada valor aparece na coluna **nome_dia_compra** do DataFrame **df_orders**. Em seguida, o código usa a função **reindex()** para ordenar os valores por dia da semana. O segundo bloco de código cria uma visualização usando a biblioteca **matplotlib**. A função **plot()** é usada para plotar os dados. No caso específico, estamos usando um gráfico de barras para plotar os dados do DataFrame **total_vendas_por_dia_semana**.

A saída do código é o gráfico de barras abaixo, que mostra que o dia da semana com mais vendas é a segunda-feira,e o dia da semana com menos vendas é o sábado.

Esses dados ajudam a entender melhor o comportamento de compra dos seus clientes, auxiliando na tomada de decisões, que envolvam:

- Alocação de mais recursos para atendimento ao cliente durante o período com mais vendas
- Envio promoções e descontos durante esse período.
- Recomendação de produtos e serviços que são mais populares nesse período.
- Dar uma atenção maior aos procedimentos de logística nos dias com mais vendas.



21 # Trazendo categorias únicas em product_category_name

```
df_category['product_category_name'].unique()
```

O código **df_category['product_category_name'].unique()** é usado para obter os valores únicos da coluna **product_category_name** do DataFrame **df_category**.

Este código imprime todas as categorias de produtos distintas disponíveis na coluna **product_category_name**.

22 # Redistribuindo as categorias

```
categoria_ampla = {
```

```
'Saude e Beleza': ['beleza_saude', 'perfumaria', 'bebes',
'cuidados_pessoais', 'fraldas_higiene'],
'Tecnologia': ['informatica_acessorios', 'tablets_impressao_imagem',
'telefonia', 'telefonia_fixa', 'consoles_games', 'audio', 'pcs',
'eletronicos',
'eletrodomesticos','eletrodomesticos_2','eletroportateis'],
'Casa e Decoração': ['cama_mesa_banho', 'moveis_decoracao',
'utilidades_domesticas',
'moveis_cozinha_area_de_servico_jantar_e_jardim',
'moveis_escritorio','moveis_colchao_e_estofado', 'moveis_sala',
'moveis_quarto', 'moveis_externos', 'portateis_casa_forno_e_cafe',
'casa_conforto_2', 'casa_conforto', 'climatizacao'],
'Automotivo': ['automotivo'],
'Lazer e Entretenimento': ['esporte_lazer', 'brinquedos',
'instrumentos_musicais', 'cds_dvds_musicais', 'dvds_blu_ray', 'musica',
'artes_e_artesanato', 'artes'],
'Moda': ['fashion_bolsas_e_acessorios', 'fashion_calcados',
'fashion_roupa_masculina', 'fashion_roupa_feminina',
'fashion_underwear_e_moda_praia',
'Fashion_esporte','fashion_roupa_infanto_juvenil'],
'Culinaria': ['alimentos', 'la_cuisine', 'bebidas', 'flores',
'alimentos_bebidas'],
'Construção e Jardim': ['ferramentas_jardim',
'construcao_ferramentas_construcao', 'construcao_ferramentas_jardim',
'construcao_ferramentas_iluminacao','Construcao_ferramentas_ferramentas',
'construcao_ferramentas_seguranca', 'casa_construcao'],
'Livros e Cultura': ['papelaria', 'livros_tecnicos',
'livros_interesse_geral', 'livros_importados'],
'Eventos e Festas': ['relogios_presentes', 'artigos_de_festas',
'artigos_de_natal'],
'Negócio e Servicos': ['market_place', 'industria_comercio_e_negocios',
'agro_industria_e_comercio', 'seguros_e_servicos', 'cine_foto'],
'Sinalização e Segurança': ['sinalizacao_e_seguranca'],
'Petshop': ['pet_shop'],
'Produtos Diversos': ['cool_stuff']
}
```

A variável **categoria_ampla** é um dicionário que redistribui as categorias de produtos para uma lista de categorias específicas. Por exemplo, a chave **'Saude e Beleza**' mapeia para a lista de categorias específicas **['beleza_saude', 'perfumaria', 'bebes', 'cuidados_pessoais', 'fraldas_higiene']**.

Este dicionário é usado para agrupar categorias de produtos semelhantes. Por exemplo, todas as categorias de produtos relacionadas à saúde e beleza podem ser agrupadas na categoria ampla **'Saude e Beleza**'.

O Reagrupamento organiza melhor as categorias, ajudando a entender de forma mais objetiva o catálogo de produtos. Por exemplo, sabendo que as categorias amplas mais populares são 'Tecnologia' e 'Casa e Decoração', a Olist pode:

- Alocar mais recursos para marketing para essas categorias.
- Melhorar a experiência de compra para esses produtos.

Além disso, este tipo de dado pode ajudar a melhorar a experiência de compra dos seus clientes. Sabendo que as categorias amplas mais populares são 'Tecnologia' e 'Casa e Decoração', a Olist pode, por exemplo:

1. Oferecer recomendações de produtos mais relevantes para os clientes.
2. Oferecer descontos e promoções para esses produtos.

Em resumo, saber as categorias amplas de produtos pode ajudar a Olist a:

- Melhorar a eficiência do marketing.
- Melhorar a experiência de compra dos clientes.

23 # Agrupando dados relevantes

```
dados_agrupados = pd.merge(df_orders, df_item, on='order_id', how='left')
dados_agrupados = pd.merge(dados_agrupados, df_products, on='product_id',
how='left')
dados_agrupados       =       pd.merge(dados_agrupados,       df_customers,
on='customer_id', how='left')
dados_agrupados  =  pd.merge(dados_agrupados,  df_reviews,  on='order_id',
how='left')


dados_agrupados.head()
```

1. O código **dados_agrupados = pd.merge(df_orders, df_item, on='order_id', how='left')** combina os DataFrames **df_orders e df_item** usando a coluna **order_id** como chave de junção. O método **how='left'** garante que todas as linhas do DataFrame **df_orders** sejam mantidas, mesmo que não haja uma correspondência na coluna **order_id** no DataFrame **df_item**.

2. O código **dados_agrupados = pd.merge(dados_agrupados, df_products, on='product_id', how='left')** combina os DataFrames **dados_agrupados** e **df_products** usando a coluna **product_id** como chave de junção. O método **how='left'** garante que todas as linhas do DataFrame **dados_agrupados** sejam mantidas, mesmo que não haja uma correspondência na coluna **product_id** no DataFrame **df_products**.

3. O código **dados_agrupados = pd.merge(dados_agrupados, df_customers, on='customer_id', how='left')** combina os DataFrames **dados_agrupados** e **df_customers** usando a coluna **customer_id** como chave de junção. O método **how='left'** garante que todas as linhas do DataFrame **dados_agrupados** sejam mantidas, mesmo que não haja uma correspondência na coluna **customer_id** no DataFrame **df_customers**.

4. O código **dados_agrupados = pd.merge(dados_agrupados, df_reviews, on='order_id', how='left')** combina os DataFrames **dados_agrupados** e **df_reviews** usando a coluna **order_id** como chave de junção. O método **how='left'** garante que todas as linhas do DataFrame **dados_agrupados** sejam mantidas, mesmo que não haja uma correspondência na coluna **order_id** no DataFrame **df_reviews**.

Este DataFrame contém dados de todos os cinco DataFrames originais. As colunas são as seguintes:

- **order_id**: O ID do pedido
- **order_status**: O status do pedido
- **product_id**: O ID do produto
- **product_name**: O nome do produto
- **product_category_name**: A categoria do produto
- **customer_id**: O ID do cliente
- **customer_name**: O nome do cliente
- **review_score**: A pontuação da revisão
- **review_comment**: O comentário da revisão

Este DataFrame pode ser usado para realizar análises mais aprofundadas sobre os dados do Olist, como:

- Identificar tendências de vendas por categoria de produto.
- Analisar as taxas de cancelamento por categoria de produto.
- Avaliar a satisfação dos clientes por categoria de produto.

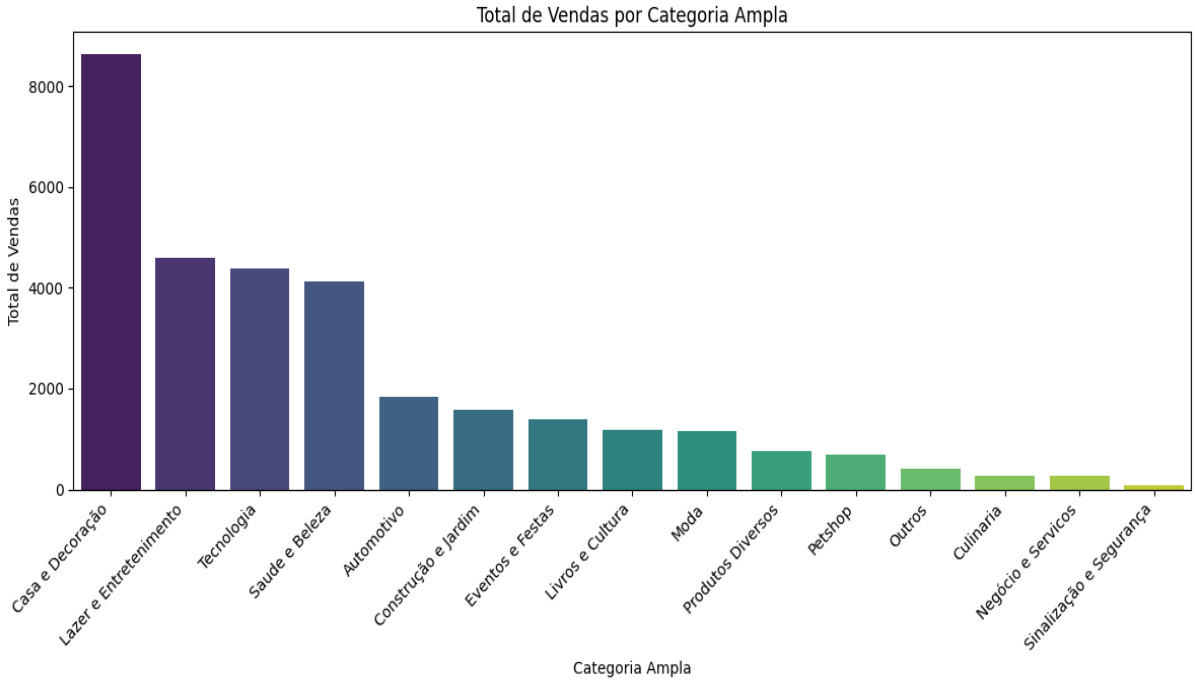
24 # Agrupando as categorias no DataFrame

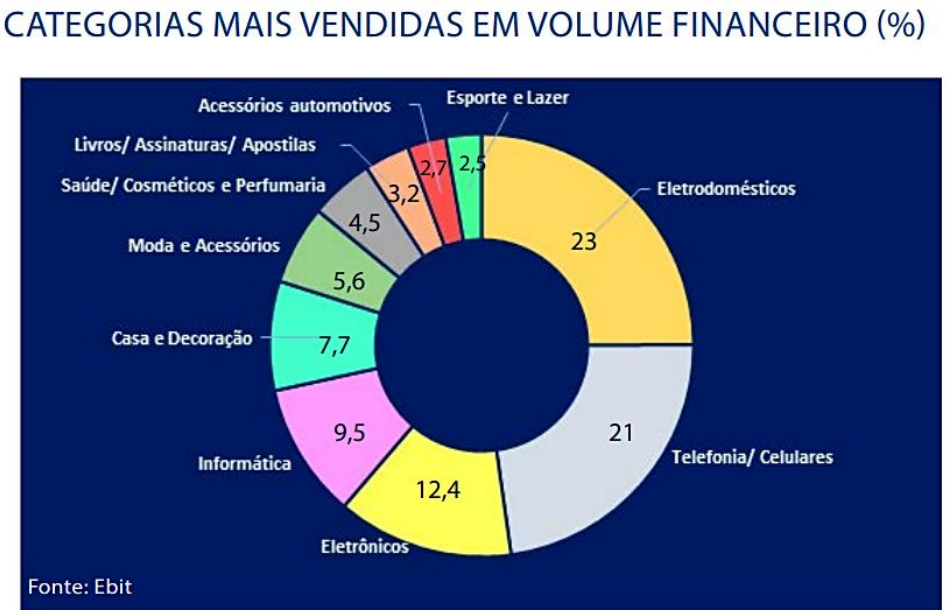
```
df_orders['categoria_ampla'] = df_products['product_category_name'].apply(lambda x: next((key for key, value in categoria_ampla.items() if x in value), 'Outros'))

# Contando as ocorrências por categoria ampla
contagem_por_categoria_ampla = df_orders['categoria_ampla'].value_counts()

# Visualização - Total de vendas por categoria ampla
plt.figure(figsize=(12, 6))
sns.barplot(x=contagem_por_categoria_ampla.index, y=contagem_por_categoria_ampla.values, palette='viridis')
plt.xlabel('Categoria Ampla')
```

```
plt.ylabel('Total de Vendas')
plt.title('Total de Vendas por Categoria Ampla')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

O código acima é usado para agrupar as categorias de produtos em categorias amplas e, em seguida, visualizar o total de vendas por categoria ampla.

O gráfico mostra que as categorias amplas de produtos com mais vendas são **Casa e Decoração** e **Lazer e Entretenimento**. A categoria **Tecnologia** é a terceira categoria com mais vendas.





Os dados acima foram extraídos da pesquisa feita pelo SBVC sobre as 70 maiores empresas de E-commerce brasileiro. A distribuição refere-se ao ano de 2016 e aponta que a Olist não segue as tendências de vendas do panorama geral em um volume considerável de categorias.

Este código pode ser usado para entender melhor as preferências de compra dos clientes da Olist.

Em 2018, o setor de Perfumaria e Cosméticos/ Saúde liderou o ranking em quantidade de pedidos, conforme aponta pesquisa divulgada pela Bis2Bis E-commerce.

A matéria segue dizendo:

Os segmentos de destaque em 2018

No tópico acima, nós já mostramos diversos dados que comprovam que o **e-commerce brasileiro cresceu em 2018**. Agora, é hora de falar dos segmentos que mais se destacaram nesse crescimento no ano passado, tanto em relação a **pedidos**, quanto em **faturamento**.

Em quantidade de pedidos, os cinco principais segmentos em 2018 foram:
1º Perfumaria e Cosméticos /Saúde: 16,4% de participação;
2º Moda e acessórios: 13,6% de participação;
3º Casa e decoração: 11,1% de participação;
4º Eletrodomésticos: 10,6% de participação; e
5º Livros / Assinaturas e Apostilas: 7,5% de participação.
Completando a lista dos 10 principais segmentos, na sequência estão: Telefonia / Celulares (7,1%); Esporte e Lazer (6,4%); Informática (5,2%); Alimentos e Bebidas (4,5%) e Eletrônicos (3,6%).

Em relação ao faturamento, as cinco principais categorias foram:
1º Eletrodomésticos: 19,6% de participação;
2º Telefonia / Celulares: 18,2 % de participação;
3º Casa de decoração: 10% de participação;
4º Informática: 9,6% de participação; e
5º Eletrônicos: 9,5% de participação.
Na sequência ainda estão: Perfumaria e Cosméticos /Saúde (6,8%); Moda e Acessórios (5,6%); Esporte e Lazer (3,6%); Acessórios Automotivos (2,5%) e Alimentos e Bebidas (2,2%).
O gráfico abaixo mostra a comparação entre 2017 e 2018 desses segmentos de destaque:

Ao saber quais categorias de produtos são mais populares, a Olist pode gerar uma série de insights valiosos para o e-commerce, contribuindo para estratégias mais eficientes e tomadas de decisão informadas que afetam diversos setores, entre eles:

- Segmentação de Mercado: Identificar as categorias mais populares ajuda a entender as preferências dos clientes. Isso permite segmentar o mercado de forma mais eficaz, adaptando estratégias de marketing e promoções específicas para cada categoria.

- Estoque e Planejamento de Produtos: Compreender quais categorias têm maior demanda facilita o planejamento de estoque. O e-commerce pode ajustar seus níveis de estoque, garantindo que produtos nas categorias mais populares estejam sempre disponíveis, evitando excessos ou escassez.

- Personalização da Experiência do Cliente:Conhecendo as categorias preferidas dos clientes, o e-commerce pode personalizar a experiência do usuário. Recomendações personalizadas, ofertas especiais e conteúdo relevante podem ser direcionados com base nas categorias de interesse do cliente.

- Estratégias de Marketing: Alocar recursos de marketing de maneira mais eficiente, concentrando esforços em promoções e campanhas para as categorias mais populares, pode melhorar o retorno sobre o investimento em marketing.

- Parcerias e Colaborações: Identificar categorias populares pode abrir oportunidades para parcerias estratégicas com fornecedores específicos ou marcas que são relevantes para essas categorias, fortalecendo a oferta de produtos.

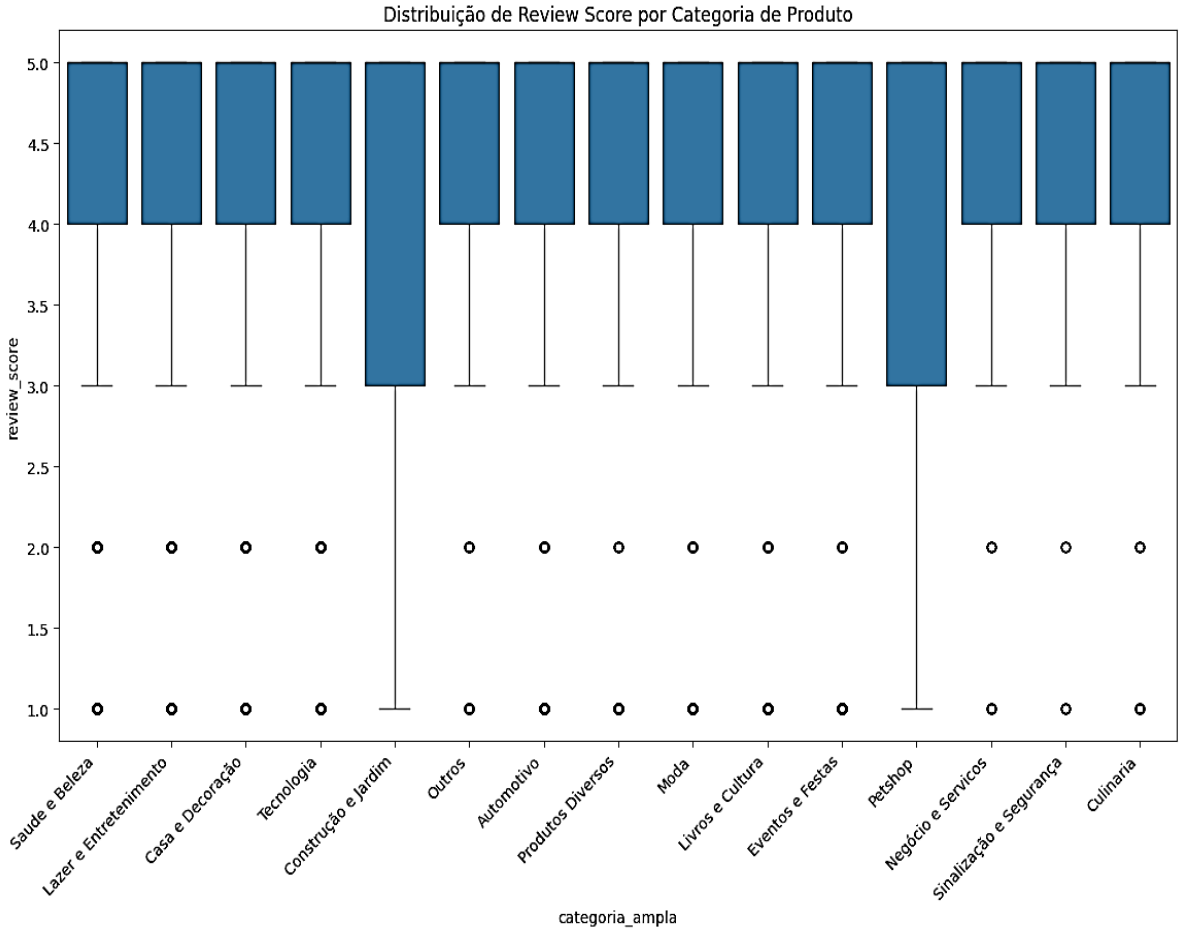
- Desenvolvimento de Novos Produtos: Compreender as categorias mais vendidas pode orientar o desenvolvimento de novos produtos. O e-commerce pode inovar dentro dessas categorias populares ou explorar categorias relacionadas para expandir seu portfólio de produtos.

- Análise Competitiva: Comparar as categorias de produtos mais vendidas com a concorrência pode oferecer insights sobre a posição do e-commerce no mercado. Identificar áreas onde a empresa está superando a concorrência ou identificar oportunidades para melhorias.

- Ciclos de Tendências: Observar as categorias mais vendidas ao longo do tempo pode ajudar a identificar tendências de mercado. Isso permite ao e-commerce antecipar mudanças nas preferências do consumidor e ajustar sua estratégia de produtos em conformidade.

25 # Gerando gráfico para definir totais de Avaliações por categoria de produto

```
plt.figure(figsize=(14, 8))
sns.boxplot(x='categoria_ampla', y='review_score', data=dados_agrupados)
plt.xticks(rotation=45, ha='right')
plt.title('Distribuição de Review Score por Categoria de Produto')
plt.show()
```



É possível perceber que as categorias que oferecem registros de avaliações negativas são Construção e Jardim e Petshop. As demais categorias tiveram avaliações entre 3 e 5.

Saber as avaliações por categoria em um e-commerce pode oferecer insights valiosos sobre a satisfação do cliente, a qualidade dos produtos e a eficácia das operações comerciais. Aqui estão algumas maneiras pelas quais a análise das avaliações por categoria pode ser benéfica:

- Melhoria da Qualidade do Produto: Identificar categorias com avaliações consistentemente baixas pode indicar problemas de qualidade nos produtos. Essa informação pode ser usada para focar em melhorias específicas em termos de design, fabricação ou sourcing de produtos nessas categorias.

- Gestão de Estoque: Avaliações negativas em categorias específicas podem estar relacionadas à falta de estoque, atrasos nas entregas ou problemas de disponibilidade. Monitorar avaliações por categoria ajuda a ajustar os níveis de estoque e melhorar a gestão da cadeia de suprimentos.

- Atendimento ao Cliente: Avaliações negativas podem estar relacionadas ao atendimento ao cliente. Identificar categorias com feedback negativo permite a implementação de melhorias nos processos de atendimento ao cliente, como tempos de resposta mais rápidos ou maior disponibilidade de suporte.

- Estratégias de Marketing: Categorias com avaliações positivas podem ser destacadas em campanhas de marketing para atrair mais clientes. O feedback positivo pode ser utilizado como uma vantagem competitiva em materiais de marketing.

- Reputação da Marca: Avaliações negativas em categorias específicas podem afetar a reputação da marca. Monitorar essas avaliações permite uma resposta rápida a problemas específicos e a construção de estratégias para restaurar a confiança do cliente.

- Desenvolvimento de Novos Produtos: Compreender as avaliações por categoria pode orientar o desenvolvimento de novos produtos. Se uma categoria específica é altamente avaliada, há oportunidades para expandir o portfólio nessa área. Se há áreas de melhoria, essas podem ser priorizadas no desenvolvimento de novos produtos.

- Análise Comparativa com a Concorrência: Comparar avaliações por categoria com a concorrência pode fornecer insights sobre como a empresa se destaca ou precisa melhorar em relação aos concorrentes.

- Feedback para Fornecedores e Parceiros: Se o e-commerce trabalha com vários fornecedores ou parceiros, as avaliações por categoria podem ser compartilhadas com eles para melhorar a qualidade dos produtos ou serviços fornecidos.

- Identificação de Tendências do Mercado: Mudanças nas avaliações por categoria ao longo do tempo podem indicar tendências do mercado. Isso permite ao e-commerce adaptar sua oferta de produtos e estratégias de marketing para acompanhar as mudanças nas preferências do consumidor.

26 # Trazendo o id dos 10 produtos mais vendidos, suas respectivas quantidades e categorias

```
# Combina as tabelas df_order e df_item com base na coluna order_id
merged_table = df_orders.merge(df_item, on='order_id')

# Converte as colunas 'product_id' para o tipo de dado 'object'
merged_table['product_id']                                        =
merged_table['product_id'].astype('object')
df_products['product_id']                                         =
df_products['product_id'].astype('object')

# Encontra os produtos mais vendidos e Cria um DataFrame com
'product_id' e 'count'
produtos_mais_vendidos                                            =
merged_table['product_id'].value_counts().head(10)
```

```
produtos_mais_vendidos_df = pd.DataFrame({'product_id':
produtos_mais_vendidos.index, 'count':
produtos_mais_vendidos.values})

# Junção (merge) para obter 'categoria_ampla' para os produtos
mais vendidos
totais_mais_vendidos = pd.merge(produtos_mais_vendidos_df,
df_products[['product_id', 'product_category_name']],
on='product_id')

# Adiciona a coluna categoria_ampla
totais_mais_vendidos['categoria_ampla'] =
totais_mais_vendidos['product_category_name'].apply(lambda x:
next((key for key, value in categoria_ampla.items() if x in
value), 'Outros'))

# Remove a coluna 'product_category_name' e ordena os totais da
contagem do maior para o menor
totais_mais_vendidos =
totais_mais_vendidos.drop('product_category_name',
axis=1).sort_values(by='count', ascending=False)
```

O código acima encontra os 10 produtos mais vendidos em uma base de dados de pedidos e itens. O código é dividido nas seguintes etapas:

1. Combinação das tabelas **df_order** e **df_item**. As duas tabelas são combinadas com base na coluna **order_id**. Isso cria uma nova tabela que contém todas as informações dos pedidos e itens.
2. Conversão das colunas **'product_id**' para o tipo de dado '**object**'. As colunas **product_id** das duas tabelas são convertidas para o tipo de dado **object**. Isso é necessário para que a função **value_counts()** funcione corretamente.
3. A função **value_counts()** é usada para encontrar os produtos mais vendidos. A função retorna um **objeto series** com o número de vezes que cada produto foi vendido.
4. Um DataFrame é criado com as colunas **product_id** e **count**. A coluna **product_id** contém os IDs dos produtos mais vendidos e a coluna **count** contém o número de vezes que cada produto foi vendido.
5. Junção (merge) para obter **'categoria_ampla**' para os produtos mais vendidos. O DataFrame criado na etapa anterior é unido com a tabela **df_products**. A união é feita com base na coluna **product_id**. Isso adiciona a coluna **product_category_name** ao DataFrame.
6. A coluna **categoria_ampla** é adicionada ao DataFrame. A coluna é preenchida com o valor da categoria ampla do produto.
7. A coluna **product_category_name** é removida do DataFrame. O DataFrame é então ordenado de acordo com a coluna **count**, do maior para o menor.

Retorna os seguintes dados:

product_id	count	categoria_ampla
aca2eb7d00ea1a7b8ebd4e68314663af	527	Casa e Decoração
99a4788cb24856965c36a24e339b6058	488	Casa e Decoração
422879e10f46682990de24d770e7f83d	484	Construção e Jardim
389d119b48cf3043d311335e499d9c6b	392	Construção e Jardim
368c6c730842d78016ad823897a372db	388	Construção e Jardim
53759a2ecddad2bb87a079a1f1519f73	373	Construção e Jardim
d1c427060a0f73f6b889a5c7c61f2ac4	343	Tecnologia
53b36df67ebb7c41585e8d54d6772e08	323	Eventos e Festas
154e7e31ebfa092203795c972e5804a6	281	Saúde e Beleza

27 # Representação visual dos dados acima (gráfico de barras)

```
# Defina o estilo do gráfico e Crie uma figura e eixos
sns.set(style="whitegrid")
plt.figure(figsize=(12  8))

# Contagem de quantas vezes cada categoria_ampla foi citada
contagem_categorias                                          =
totais_mais_vendidos['categoria_ampla'].value_counts()

# Plote o gráfico de barras horizontais com uma paleta de cores mais
alegre
ax            =            sns.barplot(x=contagem_categorias.values,
y=contagem_categorias.index, palette='Set2')

#  Adicione  anotações  para  indicar  o  número  de  vezes  que  cada
categoria_ampla foi citada
for index, value in enumerate(contagem_categorias.values):
    ax.text(value, index, f'{value} Produto{"os" if value > 1 else
""}', va='center', fontsize=10, color='black')

# Adicione rótulos e título
plt.xlabel('Contagem de Citações')
plt.ylabel('Categoria Ampla')
plt.title('Categorias com Maior Volume de Vendas')

# Exiba o gráfico
plt.show()
```

O código acima cria um gráfico de barras horizontais para visualizar a contagem de produtos vendidos em cada categoria ampla. O gráfico mostra que as categorias com maior volume de vendas são Construção e Jardim (4 produtos), Casa e Decoração (2 produtos) e Tecnologia (2 produtos).

O código funciona da seguinte forma:

A primeira linha define o estilo do gráfico como "**whitegrid**".

A segunda linha cria uma figura e eixos com tamanho de **12 polegadas por 8 polegadas**.

A terceira linha usa o método **value_counts()** do Pandas para contar quantas vezes cada **categoria_ampla** é encontrada no DataFrame **totais_mais_vendidos**.

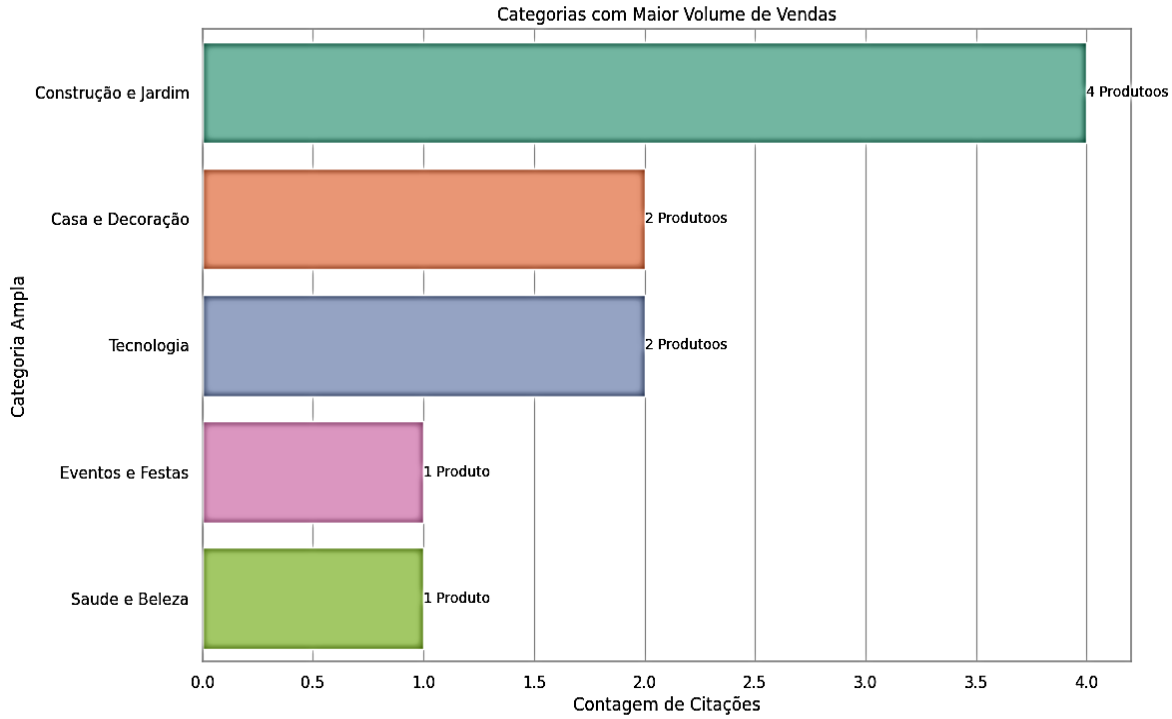
A quarta linha usa o método **barplot()** do **Seaborn** para plotar um gráfico de barras horizontais.

A quinta linha usa um **loop** para adicionar anotações ao gráfico, indicando o número de vezes que cada **categoria_ampla** foi citada.

A sexta linha adiciona rótulos e título ao gráfico.

A sétima linha exibe o gráfico.

O gráfico resultante, que segue abaixo, é útil para visualizar as categorias com maior volume de vendas. Ele pode ser usado para orientar decisões de marketing e vendas.



27 # Buscando o id dos 10 melhores vendedores, seus respectivos totais de vendas e cidades

```
# Merge dos DataFrames df_sellers e df_item
dados_vendas = pd.merge(df_sellers, df_item, on='seller_id',
how='left')

# Contagem do número de vendas por vendedor e seleção dos top 10
top_10_vendedores =
dados_vendas['seller_id'].value_counts().head(10).reset_index()
top_10_vendedores.columns = ['seller_id', 'count']
```

```
# Agrupamento das informações de dados_vendas e df_sellers para os top 10 vendedores
tabelas_combinadas = pd.merge(top_10_vendedores, df_sellers[['seller_id', 'seller_city']], on='seller_id', how='left')

# Preenchimento de 'Não encontrado' nas linhas correspondentes, caso haja vendedores não encontrados
tabelas_combinadas['seller_city'].fillna('Não encontrado', inplace=True)

print(tabelas_combinadas)
```

O código funciona da seguinte forma:

- A primeira linha usa o método **merge()** do Pandas para combinar os DataFrames **df_sellers** e **df_item**. O argumento **on='seller_id'** especifica que os dois DataFrames serão combinados pela coluna **seller_id**. O argumento **how='left'** especifica que todos os registros do DataFrame **df_sellers** serão incluídos no DataFrame combinado, mesmo que não haja um registro correspondente no DataFrame **df_item**.
- A segunda linha usa o método **value_counts()** do Pandas para contar quantas vezes cada valor aparece na coluna **seller_id** do DataFrame **dados_vendas**. O argumento **head(10)** especifica que apenas os 10 valores mais frequentes serão retornados. O argumento **reset_index()** converte o resultado em um DataFrame com duas colunas: **seller_id** e **count**.
- A terceira linha renomeia as colunas do DataFrame resultante.
- A quarta linha usa o método **merge()** do Pandas para combinar o DataFrame **top_10_vendedores** com o DataFrame **df_sellers**. O argumento **on='seller_id'** especifica que os dois DataFrames serão combinados pela coluna **seller_id**. O argumento **how='left'** especifica que todos os registros do DataFrame **top_10_vendedores** serão incluídos no DataFrame combinado, mesmo que não haja um registro correspondente no DataFrame **df_sellers**.
- A quinta linha usa o método **fillna()** do Pandas para preencher as linhas do DataFrame **tabelas_combinadas** nas quais a coluna **seller_city** é nula com o valor **'Não encontrado'**.
- A última linha imprime o DataFrame **tabelas_combinadas**.

O resultado do código é um DataFrame com as seguintes colunas:

- **seller_id**: O ID do vendedor.
- **count**: O número de vendas do vendedor.
- **seller_city**: A cidade de origem do vendedor.

Este DataFrame pode ser usado para identificar os vendedores mais bem-sucedidos e para entender melhor a distribuição geográfica das vendas.

Retorna os dados:

seller_id	count	seller_city
6560211a19b47992c3666cc44a7e94c0	2033	São Paulo
4a3ca9315b744ce9f8e9374361493884	1987	Ibitinga
1f50f920176fa81dab994f9023523100	1931	São José do Rio Preto
cc419e0650a3c5ba77189a1882b7556a	1775	Santo André
da8622b14eb17ae2831f4ac5b9dab84a	1551	Piracicaba
955fee9216a65b617aa5c0531780ce60	1499	São Paulo
1025f0e2d44d7041d6cf58b6550e0bfa	1428	São Paulo
7c67e1448b00f6e969d365cea6b010ab	1364	Itaquaquecetuba
ea8482cd71df3c1969d7b9473ff13abc	1203	São Paulo

28 # Representação visual dos dados acima

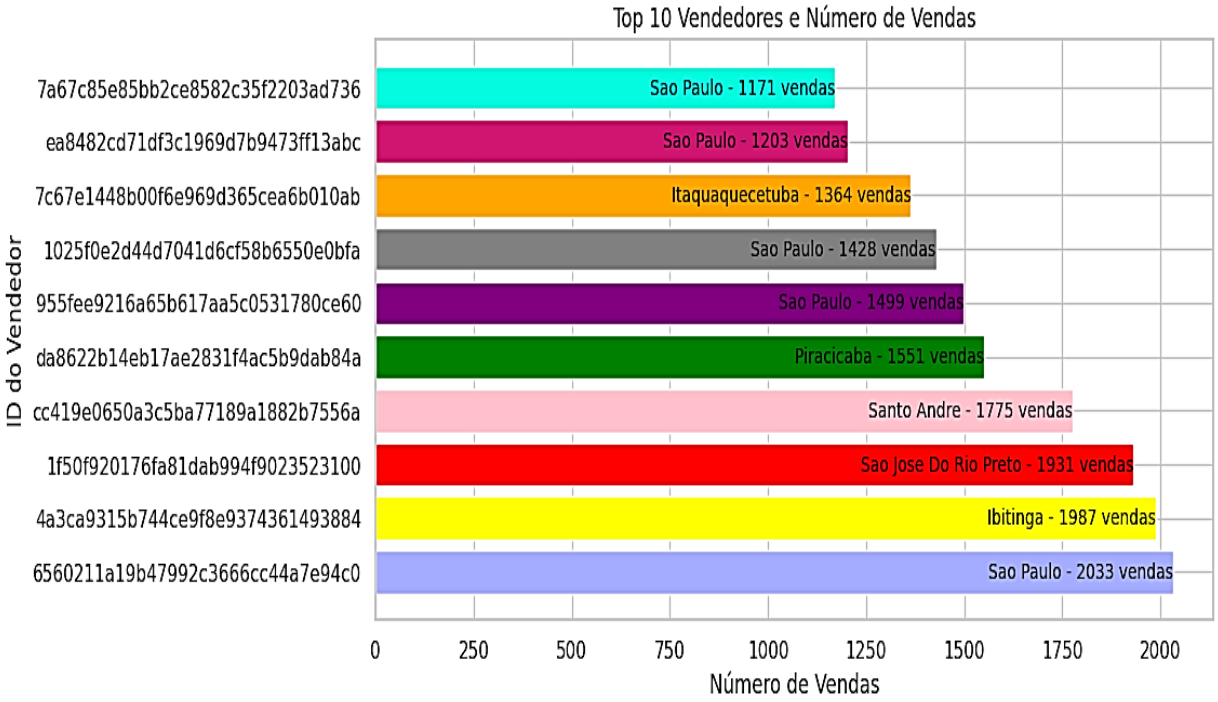
```
# Plota um gráfico de barras horizontais
plt.figure(figsize=(10  5))
colors  =  ['#a3acff',  'yellow',  'red',  'pink',  'green',  'purple',
'gray', 'orange', '#cf156f', '#03fcdf']
bars           =           plt.barh(tabelas_combinadas['seller_id'],
tabelas_combinadas['count'], color=colors)

# Adiciona a informação da cidade do vendedor dentro de cada barra
for bar, city, count in zip(bars, tabelas_combinadas['seller_city'],
tabelas_combinadas['count']):
    formatted_city = city.title()

# Capitaliza a primeira letra de cada palavra
         plt.text(count,    bar.get_y()    +    bar.get_height()/2
f'{formatted_city}  -  {count}  vendas',  há='right',  va='center',
color='black', fontsize=10)

plt.xlabel('Número de Vendas')
plt.ylabel('ID do Vendedor')
plt.title('Top 10 Vendedores e Número de Vendas')

# Exibe o gráfico
plt.show()
```

O código acima plota um gráfico de barras horizontais para visualizar os dados de vendas dos 10 melhores vendedores. O código usa a biblioteca Matplotlib para criar o gráfico.

A primeira linha do código cria uma figura com tamanho de 10 por 5 polegadas. A segunda linha define uma lista de cores para as barras do gráfico. A terceira linha usa a função **barh() para** plotar as barras do gráfico.

A quarta linha itera sobre as barras do gráfico, a cidade do vendedor e o número de vendas. Para cada barra, a função **title()** é usada para capitalizar a primeira letra de cada palavra da cidade. A função t**ext()** é usada para adicionar o texto da cidade e o número de vendas dentro da barra.

A quinta linha configura os rótulos dos eixos **x** e **y**. A sexta linha configura o título do gráfico. A sétima linha exibe o gráfico.



29 # Reagrupamento por cidade dos totais de vendas dos 10 melhores vendedores

```
# Encontra os vendedores com mais vendas
top_10_vendedores                                                =
tabelas_combinadas['seller_id'].value_counts().head(10)

# Cria um DataFrame com 'seller_id' e 'count'
top_10_vendedores_df          =          pd.DataFrame({'seller_id':
top_10_vendedores.index, 'count': top_10_vendedores.values})

#   Merge   para   adicionar   a   coluna   'seller_city'   à   base
top_10_vendedores_df
top_10_vendedores_df        =        pd.merge(top_10_vendedores_df,
df_sellers[['seller_id',      'seller_city']],      on='seller_id',
how='left', suffixes=('_top_10_vendedores', '_df_sellers'))

# Se houver vendedores não encontrados, preencha 'Não encontrado'
nas linhas correspondentes
top_10_vendedores_df['seller_city'].fillna('Não        encontrado',
inplace=True)

# Agrupa o DataFrame baseado na condição de que a 'seller_city' seja
'São Paulo'
```

```
agrupamento_df                                                    =
top_10_vendedores_df.groupby('seller_city')['count'].sum().reset_in
dex()

# Ordena o DataFrame com base na coluna 'count' em ordem decrescente
agrupamento_df       =       agrupamento_df.sort_values(by='count',
ascending=False)

# Converte a coluna 'seller_city' para tipo categórico com a ordem
desejada
ordem_cidades = ['sao paulo', 'ibitinga', 'sao jose do rio preto',
'santo andre', 'piracicaba', 'itaquaquecetuba']
agrupamento_df['seller_city']                                     =
agrupamento_df['seller_city'].astype(pd.CategoricalDtype(categories
=ordem_cidades, ordered=True))

# Ordena o DataFrame com base na nova ordem da coluna 'seller_city'
agrupamento_df = agrupamento_df.sort_values(by='seller_city')

print(agrupamento_df)
```

O código acima encontra os 10 vendedores com mais vendas e, em seguida, agrupa esses vendedores por cidade. O código usa biblioteca Pandas

O código funciona da seguinte forma:

- O código encontra os 10 IDs dos vendedores com mais vendas.
- cria um DataFrame com os 10 IDs dos vendedores e o número de vendas de cada vendedor.
- mescla o DataFrame criado com um DataFrame que contém a cidade de cada vendedor
- preenche os valores ausentes na coluna **seller_city** com o valor **Não encontrado**.
- agrupa o DataFrame pela coluna **seller_city**.
- ordena o DataFrame pela coluna **count** em ordem decrescente.
- converte a coluna **seller_city** para tipo categórico com a ordem desejada
- ordena o DataFrame pela coluna **seller_city** usando a ordem definida na linha anterior.

30 # Representação visual do código acima

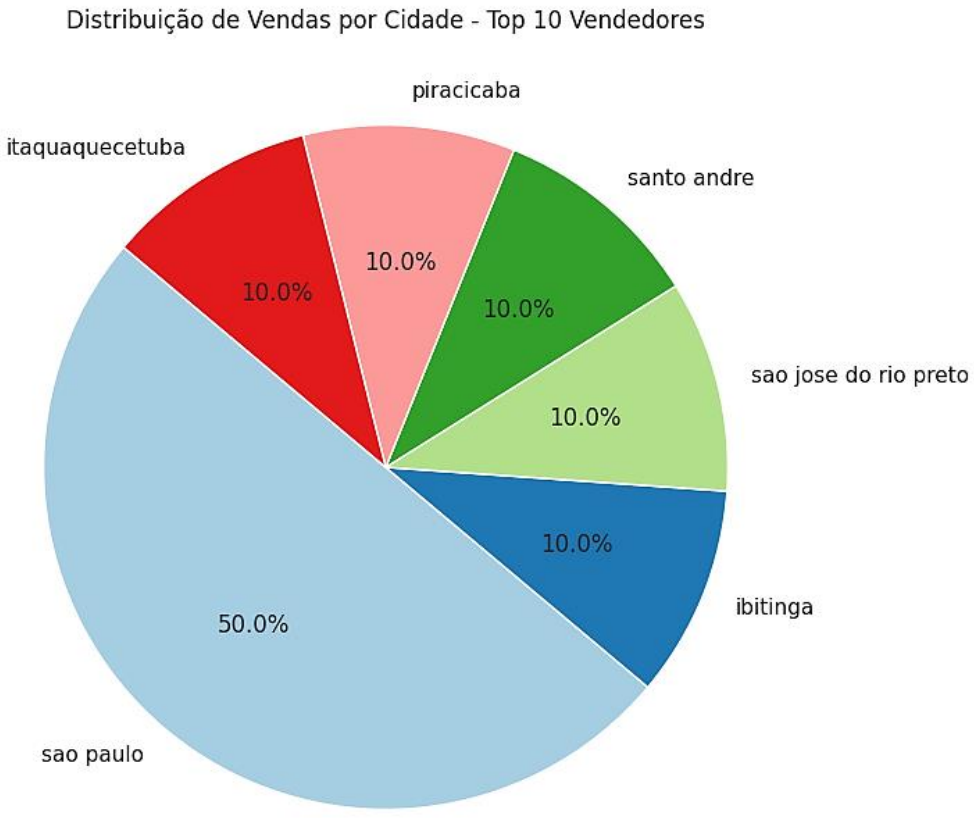
```
# Ordena o DataFrame com base na coluna 'count' em ordem decrescente
agrupamento_df       =       agrupamento_df.sort_values(by='count',
ascending=False)

# Cria um gráfico de pizza
plt.figure(figsize=(8, 8))
```

```
plt.pie(agrupamento_df['count'],
labels=agrupamento_df['seller_city'],            autopct='%1.1f%%',
startangle=140, colors=plt.cm.Paired.colors)
plt.title('Distribuição de Vendas por Cidade - Top 10 Vendedores')
plt.show()
```

A linha de código agrupamento**_df = agrupamento_df.sort_values(by='count', ascending=False)** ordena o DataFrame **agrupamento_df** pela coluna **count** em ordem decrescente. Isso significa que as cidades com mais vendas serão exibidas primeiro no gráfico de pizza.

A linha de código **plt.pie(agrupamento_df['count'], labels=agrupamento_df['seller_city'], autopct='%1.1f%%', startangle=140, colors=plt.cm.Paired.colors)** cria um gráfico de pizza usando os dados do DataFrame **agrupamento_df**.

A coluna **count** é usada para determinar o tamanho de cada fatia do gráfico. A coluna **seller_city** é usada para rotular cada fatia. A opção **autopct** é usada para exibir o percentual de vendas para cada cidade. A opção **startangle** é usada para definir o ângulo inicial do gráfico. A opção **colors** é usada para definir as cores das fatias do gráfico.



31 # Definindo a sazonalidade de compra por categoria ampla

```
#combinando as tabelas df_orders, df_item e df_products
```

```
categorias_vendas_compilado = pd.merge(pd.merge(df_item, df_orders[['order_id', 'order_delivered_customer_date']], on='order_id'), df_products[['product_id', 'product_category_name']], on='product_id')

#fazendo a conversão do formato das datas
categorias_vendas_compilado['order_delivered_customer_date'] = pd.to_datetime(categorias_vendas_compilado['order_delivered_customer_date'])

#fazendo o levantamento dos dados das vendas pelo order_delivered_customer_date
categorias_vendas_compilado['year'] = categorias_vendas_compilado['order_delivered_customer_date'].dt.year
categorias_vendas_compilado['month'] = categorias_vendas_compilado['order_delivered_customer_date'].dt.month

#agrupando os dados por ano, mês, e product_category_name
sazonalidade = categorias_vendas_compilado.groupby(['year', 'month', 'product_category_name']).size().reset_index(name='count')

# Adicione a coluna categoria_ampla a sazonalidade
sazonalidade['categoria_ampla'] = sazonalidade['product_category_name'].apply(lambda x: next((key for key, value in categoria_ampla.items() if x in value), 'Outros'))
```

O código acima combina três tabelas de dados para criar um DataFrame que contém as vendas de cada categoria de produto em cada mês do ano. O código usa a biblioteca Pandas.

O código funciona da seguinte forma:

1. O código combina as tabelas **df_orders**, **df_item** e **df_products** usando a coluna **product_id** como chave de mesclagem.
2. O código converte o formato das datas da coluna **order_delivered_customer_date** para o formato **datetime**.
3. O código adiciona duas colunas ao DataFrame **categorias_vendas_compilado**:
    - **year**: o ano da data de entrega do pedido
    - **month**: o mês da data de entrega do pedido
4. O código agrupa o DataFrame **categorias_vendas_compilado** por ano, mês e categoria de produto.
5. O código adiciona uma coluna ao DataFrame sazonalidade chamada **categoria_ampla**.

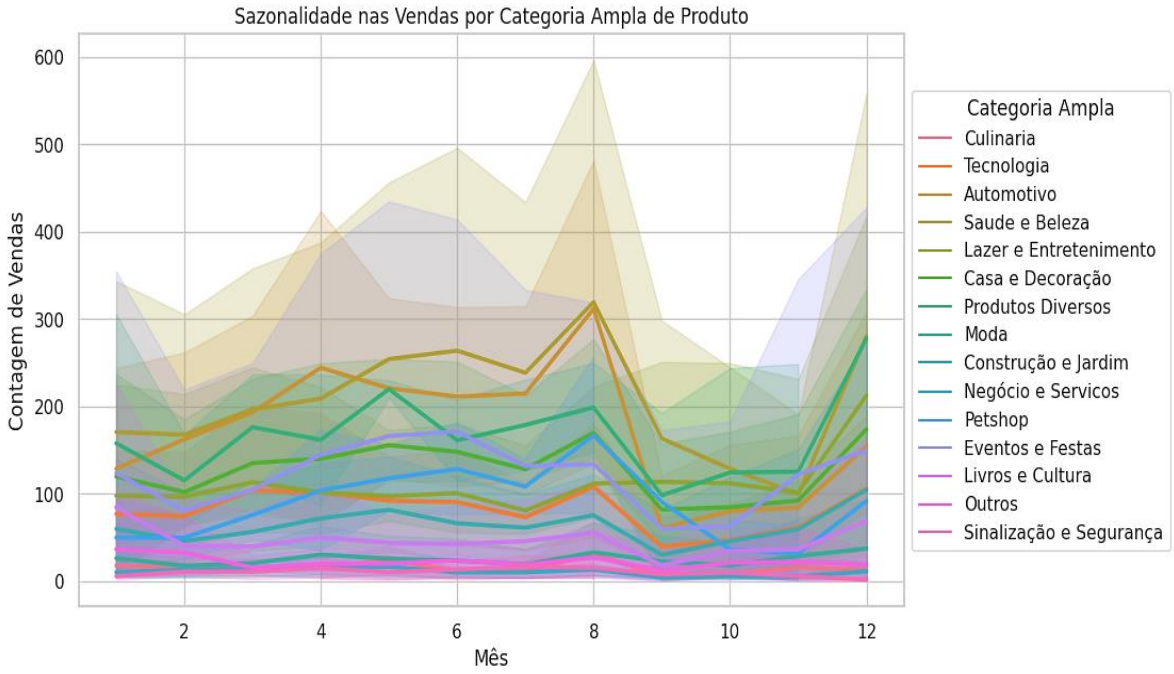
O resultado final do código é um DataFrame com as seguintes colunas:

- **year**
- **month**
- **product_category_name**
- **categoria_ampla**
- **count**

Este DataFrame pode ser usado para analisar a sazonalidade das vendas de produtos. Por exemplo, o código pode ser usado para determinar quais categorias de produtos têm as maiores vendas em determinados meses do ano.

32# Representação visual dos dados acima:

```
# Ajuste a figura para acomodar a legenda à direita do gráfico
plt.figure(figsize=(10, 6))

# Plote o gráfico de linha com linhas mais visíveis
sns.lineplot(x='month',      y='count',      hue='categoria_ampla',
data=sazonalidade, linewidth=2.5)

# Adicione a legenda à direita do gráfico
plt.legend(loc='center     left',     bbox_to_anchor=(1.0,     0.5),
title='Categoria Ampla')

plt.title('Sazonalidade nas Vendas por Categoria Ampla de Produto')
plt.xlabel('Mês')
plt.ylabel('Contagem de Vendas')
plt.show()
```



O setor **Saúde e Beleza** lidera o ranking nos meses de junho e agosto.

O setor **Casa e Decoração** segue um volume de vendas com certa estabilidade e sempre próximo dos maiores números, tendo um dos maiores totais de vendas do gráfico no mês de dezembro.

O setor **Automotivo** é o setor com o segundo maior volume de vendas no mês de agosto.

O setor de **Produtos Diversos** seguiu um fluxo de aumento, seguindo de queda em todos os meses do ano. Em dezembro, acompanhou de perto o volume de vendas do setor **Saúde e Beleza**.

33 #Totais absolutos de vendas por categoria ampla

```
# Agrupe os order_id por categoria_ampla e calcule os totais
contagem_categorias = df_orders.groupby('categoria_ampla')['order_id'].count().reset_index()
contagem_categorias.columns = ['categoria_ampla', 'total_vendas']

# Ordene o DataFrame pelos totais de vendas em ordem decrescente
contagem_categorias = contagem_categorias.sort_values(by='total_vendas', ascending=False)

print(contagem_categorias)
```

O código funciona da seguinte forma:

- O código agrupa o DataFrame **df_orders** por categoria ampla.
- A função **count()** é usada para contar o número de pedidos para cada categoria.

- A função **reset_index()** é usada para redefinir o índice do DataFrame para que ele contenha as colunas **categoria_ampla** e **total_vendas**.
- O código ordena o DataFrame **contagem_categorias** pelos totais de vendas em ordem decrescente.

O resultado final do código é um DataFrame com as seguintes colunas:

- **categoria_ampla**: a categoria ampla do pedido
- **total_vendas**: o número total de pedidos para a categoria ampla

Este DataFrame pode ser usado para analisar as vendas de produtos por categoria. Por exemplo, o código pode ser usado para determinar quais categorias de produtos têm as maiores vendas.

Retorna os dados:

Categoria ampla	Total de vendas
Casa e Decoração	8.906
Lazer e Entretenimento	4.717
Tecnologia	4.511
Saúde e Beleza	4.243
Automotivo	1.900
Construção e Jardim	1.635
Eventos e Festas	1.420
Livros e Cultura	1.219
Moda	1.202
Outros	1.030
Produtos Diversos	789
Petshop	719
Culinária	291
Sinalização e Segurança	93

34# Representação visual

```
# Defina o estilo do gráfico
sns.set(style="whitegrid")

# Crie um gráfico de linha com cores diferentes para cada categoria
e aumente o tamanho dos marcadores
plt.figure(figsize=(12, 8))
sns.lineplot(x='categoria_ampla',                y='total_vendas',
data=contagem_categorias,     marker='o',     hue='categoria_ampla',
palette='Set2', markers=True, ci=None, markersize=20)

# Adicione rótulos e título
plt.xlabel('Categoria Ampla')
plt.ylabel('Total de Vendas')
plt.title('Distribuição de Vendas por Categoria Ampla')

# Rotacione os rótulos no eixo x para melhor legibilidade
plt.xticks(rotation=45, ha='right')
```

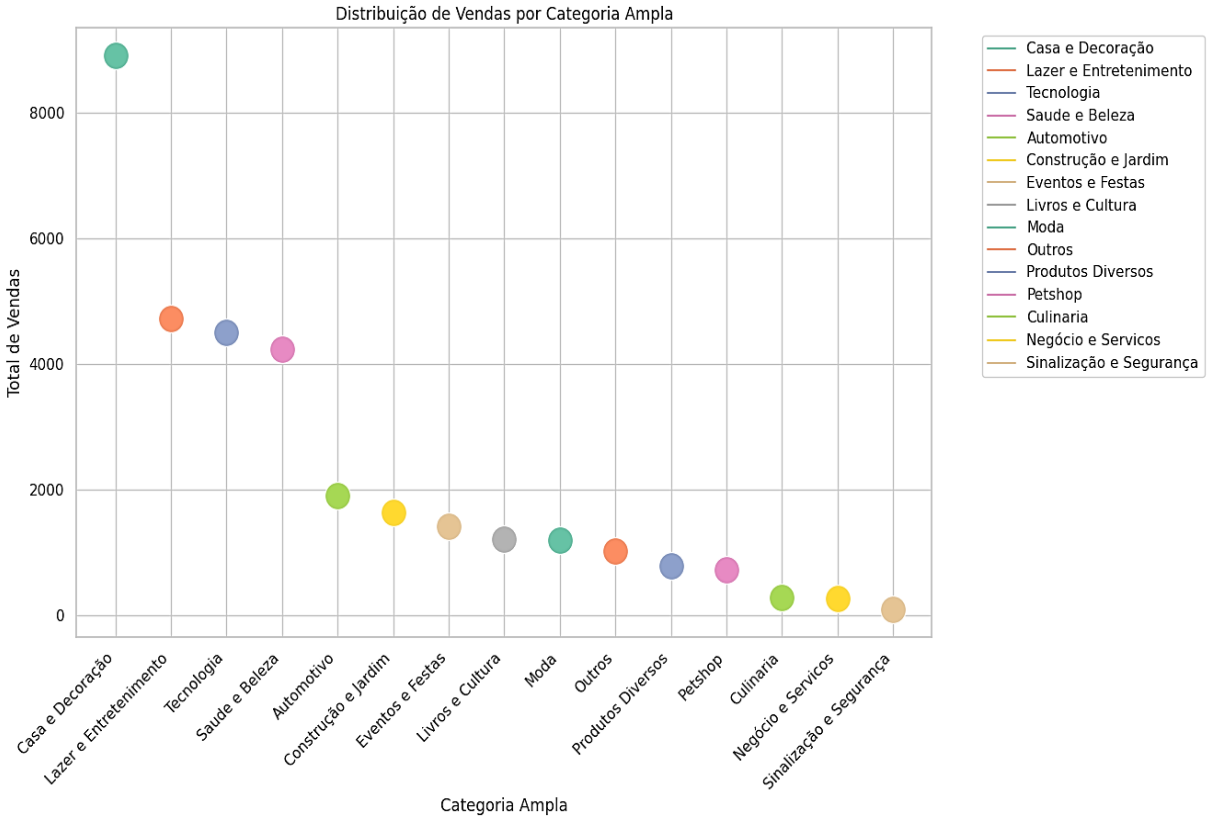
```
# Ajuste a legenda para ficar fora do gráfico
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')

# Exiba o gráfico
plt.show()
```

O código acima cria um gráfico de linha para visualizar a distribuição de vendas por categoria ampla.

As primeiras duas linhas definem o estilo do gráfico e o tamanho da figura. A linha 3 cria o gráfico de linha usando o pacote **Seaborn**. A linha 4 adiciona rótulos e título ao gráfico. A linha 5 rotaciona os rótulos no eixo x para melhor legibilidade. A linha 6 ajusta a legenda para ficar fora do gráfico. A linha 7 exibe o gráfico.

Mais especificamente, o código faz o seguinte:

- Linha 1: Define o estilo do gráfico para usar uma grade branca.
- Linha 2: Define o tamanho da figura para 12 polegadas de largura e 8 polegadas de altura.
- Linha 3: Cria o gráfico de linha usando o pacote Seaborn. O gráfico usa o eixo x para representar a categoria ampla e o eixo y para representar o total de vendas. Os pontos de dados são representados por marcadores de círculo. As linhas são coloridas de acordo com a categoria ampla.
- Linha 4: Adiciona rótulos aos eixos e um título ao gráfico.
- Linha 5: Rotaciona os rótulos no eixo x em 45 graus para melhor legibilidade.
- Linha 6: Ajusta a legenda para ficar fora do gráfico, na parte superior esquerda.
- Linha 7: Exibe o gráfico.



A Abrainc divulgou dados que apontaram crescimento de 9% do setor de Construção no primeiro semestre de 2017, com expectativa de aumento de 23% em relação ao ano anterior. O volume de vendas absolutas observado na Olist acompanha a tendência prevista.

35 # Gerando top 5 categorias com maior volume de avaliações com nota 5

```
# Etapa 1: Combine as tabelas df_orders e df_reviews usando a coluna order_id
merged_data = pd.merge(df_orders, df_reviews, on='order_id', how='left')

# Etapa 2: Filtre as avaliações com nota 5
avaliacoes_nota_5 = merged_data[merged_data['review_score'] == 5]

# Etapa 3: Agrupe por categoria_ampla e conte a quantidade de avaliações para cada categoria
contagem_avaliacoes_por_categoria = avaliacoes_nota_5.groupby('categoria_ampla').size().reset_index(name='count')

# Etapa 4: Ordene as categorias pelo número de avaliações em ordem decrescente
contagem_avaliacoes_por_categoria = contagem_avaliacoes_por_categoria.sort_values(by='count', ascending=False)
```

```
# Etapa 5: Selecione as top 5 categorias
top5_categorias = contagem_avaliacoes_por_categoria.head(5)

print(top5_categorias)
```

O código acima é usado para identificar as cinco categorias com mais avaliações de 5 estrelas.

As etapas do código são as seguintes:

- Combina as tabelas **df_orders** e **df_reviews** usando a coluna **order_id**. Isso cria uma nova tabela que contém os dados de pedidos e avaliações.
- Filtra as avaliações com nota 5. Isso cria uma nova tabela que contém apenas as avaliações com nota 5.
- Agrupa por **categoria_ampla** e conta a quantidade de avaliações para cada categoria. Isso cria uma nova tabela que contém o número de avaliações de 5 estrelas para cada categoria.
- Ordena as categorias pelo número de avaliações em ordem decrescente. Isso cria uma nova tabela que contém as categorias ordenadas pelo número de avaliações de 5 estrelas.
- Seleciona as top 5 categorias. Isso cria uma nova tabela que contém as cinco categorias com mais avaliações de 5 estrelas.

categoria_ampla	count
Casa e Decoração	5191
Lazer e Entretenimento	2759
Tecnologia	2607
Saúde e Beleza	2455
Automotivo	1091

36 # Representação visual

```
# Utilize uma paleta de cores diferente para cada barra
cores = sns.color_palette('husl', n_colors=len(top5_categorias))

# Crie um gráfico de barras
plt.figure(figsize=(10, 6))
bars = plt.bar(top5_categorias['categoria_ampla'],
top5_categorias['count'], color=cores)

# Adicione rótulos e título
plt.xlabel('Categoria Ampla')
```

```
plt.ylabel('Número de Avaliações (Nota 5)')
plt.title('Top 5 Categorias com Avaliações Nota 5')

# Adicione rótulos nas barras
for bar in bars:
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(),
str(int(bar.get_height())),
             ha='center', va='bottom', color='black', fontsize=10)

# Rotacione os rótulos do eixo x para melhor legibilidade
plt.xticks(rotation=45, ha='right')

# Exiba o gráfico
plt.show()
```

O código acima cria um gráfico de barras para visualizar as cinco categorias com mais avaliações de 5 estrelas.

- Linha 1: Define a paleta de cores para as barras do gráfico. A paleta de cores husl é usada, com um número de cores igual ao número de categorias na tabela top5_categorias.

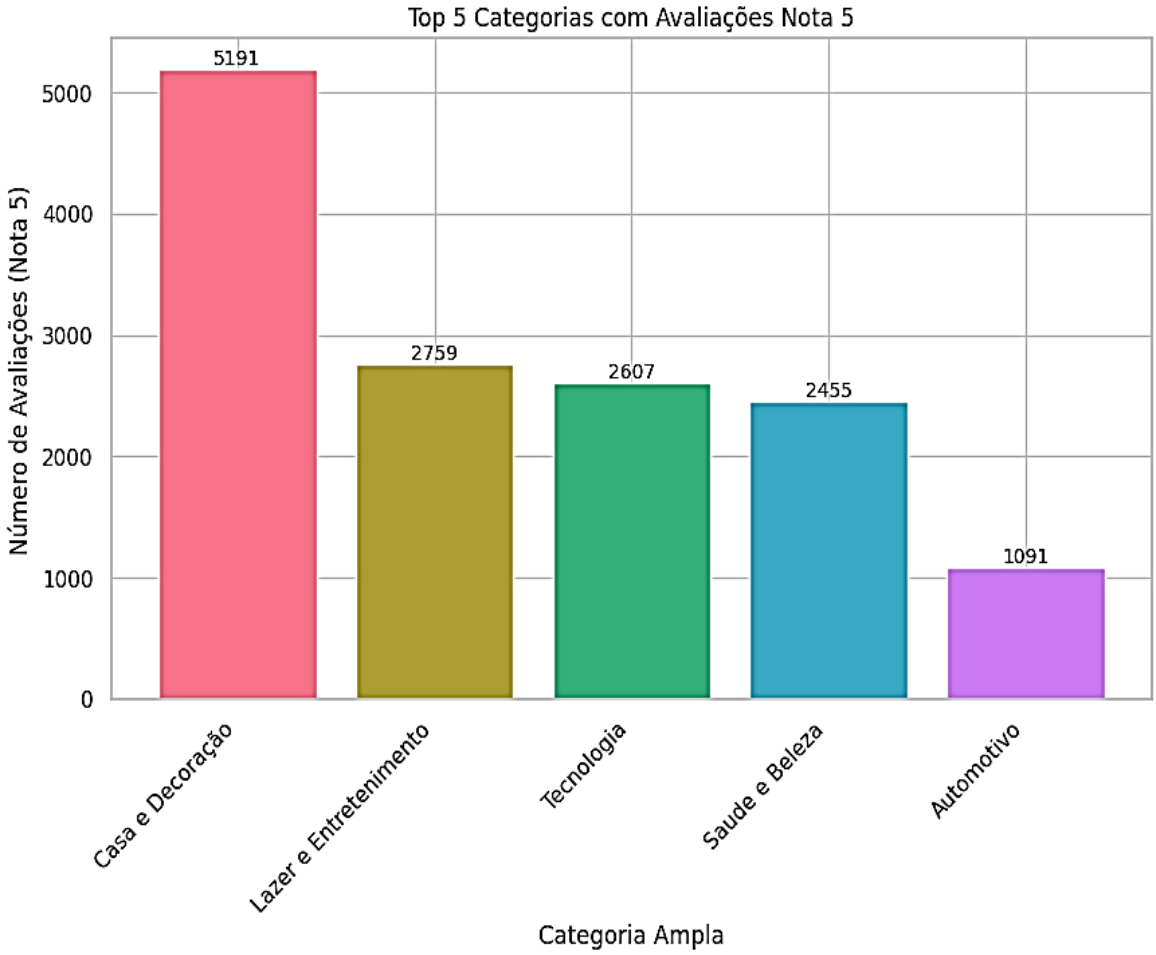
- Linha 2: Cria o gráfico de barras usando o pacote Matplotlib. O gráfico usa o eixo x para representar a categoria ampla e o eixo y para representar o número de avaliações. As barras são coloridas de acordo com a paleta de cores definida anteriormente.

- Linha 3: Adiciona rótulos aos eixos e um título ao gráfico. Os rótulos do eixo x são os nomes das categorias amplas. O rótulo do eixo y é o número de avaliações. O título do gráfico é "Top 5 Categorias com Avaliações Nota 5".

- Linha 4: Adiciona rótulos nas barras do gráfico. Os rótulos são o número de avaliações de 5 estrelas para cada categoria. Os rótulos são colocados no centro das barras, na parte inferior. A cor dos rótulos é preta e o tamanho da fonte é 10.

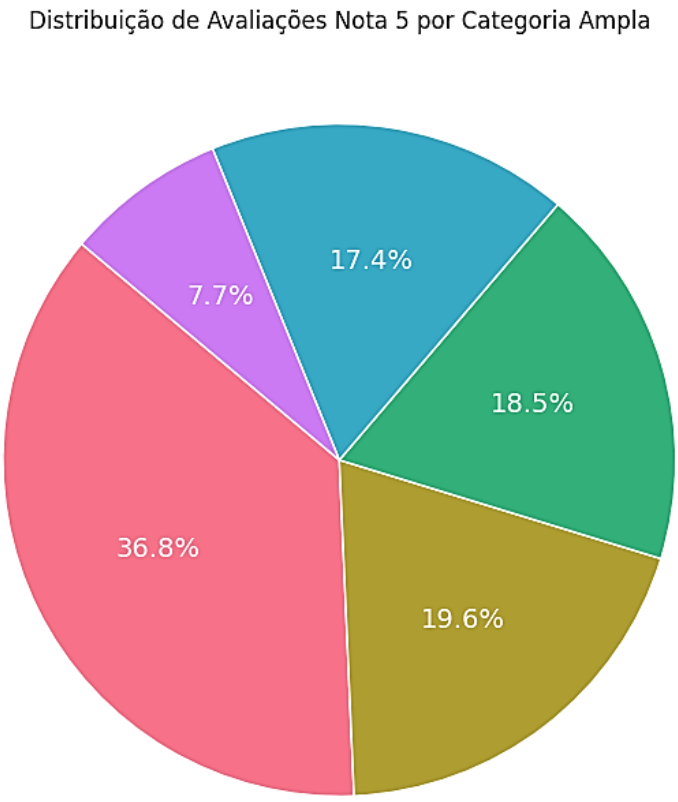
- Linha 5: Rotaciona os rótulos do eixo x em 45 graus para melhor legibilidade.

- Linha 6: Exibe o gráfico.



37 # Outro modelo de gráfico:

```
# Utilize uma paleta de cores diferente para cada categoria
cores = sns.color_palette('husl', n_colors=len(top5_categorias))

# Crie um gráfico de pizza
plt.figure(figsize=(10, 8))
wedges, texts, autotexts = plt.pie(
    top5_categorias['count'],
    labels=top5_categorias['categoria_ampla'],
    autopct='%1.1f%%',
    startangle=140,
    colors=cores,
    textprops=dict(color="w", size=14


# Adicione título
plt.title('Distribuição de Avaliações Nota 5 por Categoria Ampla')

# Exiba o gráfico
plt.show()
```



As 5 categorias com maior volume de avaliações positivas são também, respectivamente, as categorias com maior volume de vendas absolutas.

Alguns fatores podem ajudar a entender essa informação:

Demanda e Popularidade: A categoria **Casa e Decoração** é a líder em vendas e satisfação do cliente na base da Olist, diferente dos dados gerais do volume de compras do consumidor brasileiro, o que indica que essa categoria é destaque exclusivamente na Olist.

A Categoria **Lazer e Entretenimento**, que abrange **Esporte, brinquedos, instrumentos musicais, CDs/ DVDs, música e artesanato** também é líder em vendas e avaliações na Olist, sendo este um diferencial em relação ao e-commerce em geral, o que pode indicar um forte setor para aplicar investimentos.

Já a Categoria **Tecnologia**, que abrange **Informática e Acessórios, Tablets, Telefonia, Telefonia Fixa, consoles games, áudio, Pcs, eletrônicos, eletrodomésticos e eletroportáteis** acompanha a tendência do mercado geral, o que pode justificar tanto o volume de vendas, quanto a satisfação dos clientes.

Saúde e Beleza e **Automotivo**, são forte setores da Olist, não acompanhando a tendência de mercado geral.

Qualidade dos Produtos: Isso pode indicar que a qualidade dos produtos nessas categorias pode ser consistentemente alta, resultando em avaliações positivas. Clientes satisfeitos tendem a avaliar os produtos mais favoravelmente, contribuindo para uma média de avaliação mais alta.

Satisfação do Cliente: Pode haver um foco particular na satisfação do cliente nessas categorias, com as empresas implementando práticas e políticas que garantem uma experiência positiva para o cliente, desde a compra até o recebimento do produto.

Reputação da Marca: Se as marcas nessas categorias têm uma boa reputação no mercado, os consumidores podem estar mais propensos a comprar dessas marcas, confiando na qualidade e na experiência geral.

Avaliações como Fator de Compra: Se os consumidores consideram as avaliações como um fator importante na decisão de compra, é possível que produtos nessas categorias tenham sido avaliados positivamente, o que, por sua vez, impulsiona mais vendas.

Estratégias de Marketing Eficientes: As empresas que atuam nessas categorias podem ter implementado estratégias de marketing eficientes, aumentando a visibilidade e a atratividade de seus produtos.

Feedback Constante: A falta de avaliações com notas inferiores a 3 pode indicar que as empresas estão atentas ao feedback dos clientes e tomam medidas corretivas quando necessário, mantendo a satisfação do cliente em um nível elevado.

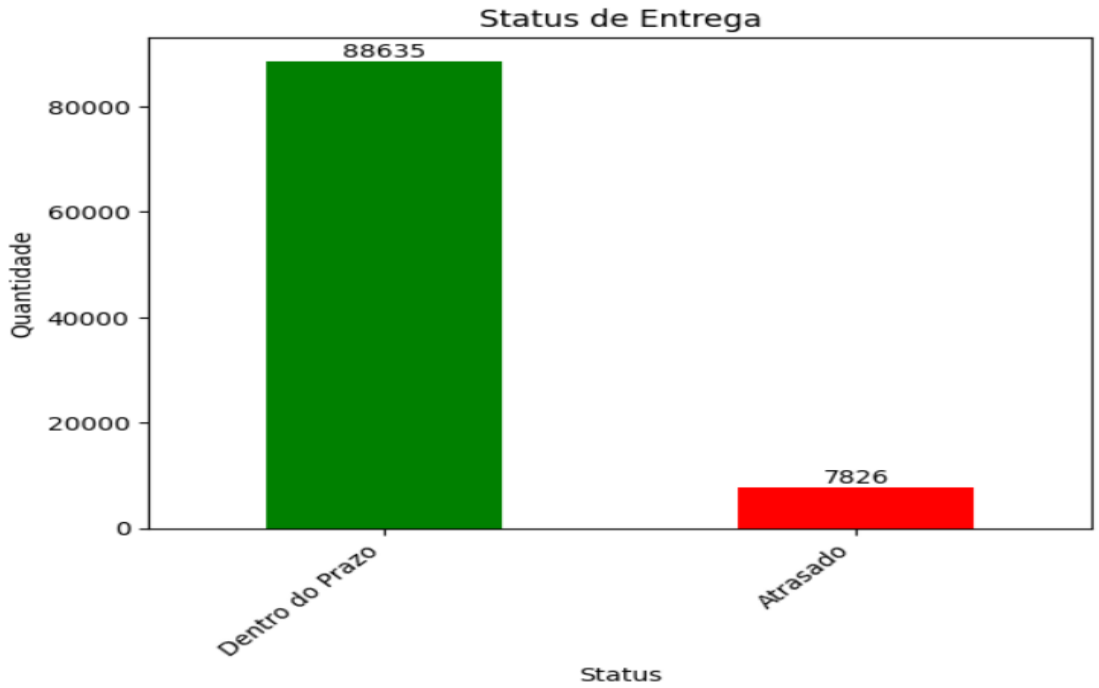
38 # Logística

```
df_orders['delivery_time_difference']                                    =
df_orders['order_estimated_delivery_date']                               -
df_orders['order_delivered_customer_date']

# Classificar como atrasado (negativo) ou dentro do prazo (positivo)
df_orders['delivery_status']                                             =
df_orders['delivery_time_difference'].apply(lambda x: 'Atrasado' if x.days
< 0 else 'Dentro do Prazo')

delivery_counts = df_orders['delivery_status'].value_counts()

delivery_counts.plot(kind='bar', color=['green', 'red'])
plt.title('Status de Entrega')
plt.xlabel('Status')
plt.ylabel('Quantidade')
plt.xticks(rotation=45, ha='right')


for i, value in enumerate(delivery_counts):
    plt.text(i, value + 0.1, str(value), ha='center', va='bottom')

plt.show()
```

Este código Python analisa dados de entrega de pedidos. Vejamos:

1. Calcula o tempo de atraso:
    - **df_orders['delivery_time_difference']**: cria uma coluna calculando a diferença entre a data estimada de entrega **(order_estimated_delivery_date)** e a data real de entrega **(order_delivered_customer_date)**.
2. Classifica entregas:
    - **df_orders['delivery_status']**: cria uma coluna chamada **delivery_status** aplicando uma função **lambda** ao **delivery_time_difference**. Se a diferença for negativa (entrega atrasada), classifica como "**Atrasado**", caso contrário, classifica como "**Dentro do Prazo**".
3. Analisa estatísticas:
    - **delivery_counts**: conta o número de ocorrências de cada tipo de **delivery_status**.
4. Visualiza os resultados:
    - Utiliza o método **plot(kind='bar')** para criar um gráfico de barras com as contagens de "**Atrasado**" e "**Dentro do Prazo**", utilizando cores verde e vermelho respectivamente.
    - Configura título, rótulos de eixos e rotaciona os nomes dos status para melhor leitura.
    - Adiciona o número de ocorrências acima de cada barra.

Em resumo, este código analisa se as entregas da Olist foram feitas dentro do prazo ou não, visualizando essa informação através de um gráfico de barras:



De acordo com matéria da CNN, a procura por galpões logísticos tem aumentado gradativamente e atingiu recordes em 2022.

No segundo trimestre de 2020, a taxa de absorção líquida (volume de locações) de galpões foi de 384% a mais do que no mesmo período do ano anterior.

Isso é um dos reflexos do fortalecimento do e-commerce, que gera novas demandas para solucionar problemas logísticos, trazendo mais agilidade e eficiência para as entregas.

Para contornar os problemas logísticos percebidos nas entregas da Olist, a criação de centros logísticos ou locação de galpões é uma solução em potencial.

Fatores externos e o comportamento de compra

Ao longo de toda a análise, é possível perceber momentos em que fatores externos geraram reflexo no comportamento de compra dos consumidores.

Datas comemorativas, como Dia das Mães e a Black Friday geraram aumentos expressivos nos volumes de vendas gerais e também nas vendas a Olist.
Algumas categorias que apresentam maior procura geral, também influenciam no volume de vendas.

O início do ano, bem como o mês de outrobro de todos os anos analisados, sofreram quedas consideráveis no volume de vendas, acompanhando uma queda geral dos setores. Um dos fatores de influência para esses resultados pode ser a preparação para gastos de final de ano (em outubro) e ao gasto excessivo gerado pelas compras de final de ano e início das aulas do ano seguinte (dezembro – janeiro – fevereiro)

AS PRINCIPAIS DATAS SAZONAIS/COMEMORATIVAS NO BRASIL

DATAS COMEMORATIVAS	PERÍODO	PEDIDOS	FATURAMENTO	TÍQUETE MÉDIO	VARIAÇÃO PEDIDOS	VARIAÇÃO FINANCEIRA	SHARE PEDIDOS DO ANO	SHARE FINANCEIRO DO ANO
DIA DO CONSUMIDOR	14/03/2018	521 mil	R$ 219 milhões	R$420	24%	8%	0,4%	0,4%
DIA DAS MÃES	28/04 a 12/05/2018	4,6 mi	R$ 2,11 bilhões	R$459	2%	12%	3,7%	4,0%
DIA DOS NAMORADOS	28/05 a 11/06/2018	3,8 mi	R$ 1,77 bilhões	R$464	-6%	4%	3,1%	3,3%
DIA DOS PAIS	28/07 a 11/08/2018	5,1 mi	R$ 2,09 bilhões	R$409	22%	8%	4,2%	3,9%
DIA DAS CRIANÇAS	27/09 a 11/10/2018	4,8 mi	R$ 1,97 bilhões	R$409	8%	6%	3,9%	3,7%
BLACK FRIDAY	22 e 23/11/2018	4,3 mi	R$ 2,6 bilhões	R$608	13%	23%	3,5%	4,9%
CYBER MONDAY	26/11/2018	752 mil	R$ 372 milhões	R$494	4%	20%	0,6%	0,7%
NATAL	10/12 a 24/12/2018	5,3 mi	R$ 2,54 bilhões	R$475	4%	18%	4,4%	4,8%

*Dia das crianças e Natal ajustado para 15 dias antes da data de comemoração.
Comparação realizada com o mesmo período do ano anterior (2018 vs. 2017) Fonte: Ebit | Nielsen

Fonte: Link

Insights do e-commerce no Brasil: diversidade, inovação e análise contínua

O e-commerce no Brasil vem crescendo de forma constante nos últimos anos, impulsionado por fatores como a crescente urbanização, a expansão da banda larga e a popularização dos smartphones. Em 2023, o e-commerce brasileiro deve atingir um faturamento de R$ 1,7 trilhão, representando um crescimento de 20% em relação ao ano anterior.

Para acompanhar esse crescimento, as empresas de e-commerce precisam estar atentas às tendências do mercado e às necessidades dos consumidores. Nesse sentido, a análise de dados é uma ferramenta essencial para obter insights que possam ajudar as empresas a tomar melhores decisões.

Diversidade nas Preferências de Pagamento:

Tanto na análise do e-commerce específico quanto na pesquisa mais ampla sobre os hábitos de pagamento dos brasileiros, fica evidente que há uma diversidade nas preferências de pagamento. Isso destaca a importância de compreender as nuances do comportamento do consumidor para atender às diversas necessidades.

Adoção Crescente de Tecnologias Emergentes:

Ambos os conjuntos de insights destacam a crescente adoção de tecnologias emergentes, como Pix e possivelmente o WhatsApp Pay. Isso sugere que os consumidores brasileiros estão abertos a inovações no setor de pagamentos e que as empresas precisam estar atentas a essas mudanças para oferecer soluções relevantes.

Preferência pelo Cartão de Crédito e Desafios do Cartão de Débito:

No contexto do e-commerce, o cartão de crédito é a opção preferida, enquanto o cartão de débito enfrenta desafios, incluindo autenticações adicionais e a falta de programas de benefícios. Essa tendência é consistente com as preferências gerais de pagamento no Brasil, destacando a relevância do crédito nas transações.

Importância da Análise Contínua:

Ambas as análises fazem referência a dados específicos em um determinado período, ressaltando a importância da análise contínua para acompanhar as mudanças nas preferências do consumidor. O comportamento de pagamento está em constante evolução, e as empresas precisam adaptar suas estratégias para permanecerem relevantes.

Crescimento Sólido do E-commerce:

O código revela um aumento consistente no número de compras ao longo dos anos, corroborando com o crescimento expressivo observado no setor de e-commerce,

conforme indicado pelo Ranking das 50 Maiores Empresas. O setor não apenas cresceu acima da média, mas também demonstrou resiliência em meio a um ambiente de crescimento econômico modesto.

Concentração nas Principais Empresas:

O Ranking destaca uma notável concentração nas 50 maiores empresas de e-commerce, que representam uma fatia significativa do mercado. Isso se alinha à tendência observada no código, onde o gráfico de barras evidencia a evolução das compras, indicando um possível aumento na participação das principais empresas ao longo do tempo.

Importância da Presença Multicanal:

O insight sobre a presença dominante de empresas multicanal nas 50 maiores destaca a importância da integração online e offline. Isso se relaciona com a análise do código, que, embora não explicitamente mencione integração, fornece uma base para explorar padrões e comportamentos de compra tanto online quanto offline.

Estratégias de Crescimento Variadas:

A variedade de estratégias de crescimento observada no Ranking, incluindo a presença de marketplaces e o sucesso de algumas empresas em atingir crescimento acima de 100%, ressalta a diversidade no setor de e-commerce. Isso pode ser reflexo das adaptações constantes das empresas aos desafios e oportunidades do mercado, como indicado pelo código que analisa dados específicos de compras.

Desafios e Oportunidades no Horizonte:

A presença crescente da Amazon e as oportunidades destacadas no setor, conforme mencionado no Ranking, indicam que o e-commerce brasileiro está sujeito a mudanças significativas. As empresas bem-sucedidas serão aquelas que conseguirem navegar eficientemente pelos desafios, como concorrência acirrada e evolução das preferências do consumidor.

- Como Melhorar a Satisfação dos clientes?

Insights Top 3 nichos

1º Perfumaria e Cosméticos /Saúde: 16,4%

Em **2018** teve um crescimento do E-commerce no Setor de Beleza

Houve um aumento notável nas vendas online de produtos de perfumaria e cosméticos, impulsionado pela crescente preferência dos consumidores por compras digitais, conveniência e variedade de opções.

Experiência do Cliente em Foco

Empresas focaram na melhoria da experiência do cliente online, oferecendo informações detalhadas sobre produtos, tutoriais de beleza e avaliações de clientes para aumentar a confiança do consumidor e incentivar as compras.

Personalização e Recomendações

Plataformas de e-commerce investiram em tecnologias de personalização, proporcionando recomendações de produtos baseadas em preferências individuais, histórico de compras e análises de beleza.

Expansão Internacional

Empresas no setor de beleza expandiram suas operações internacionalmente, aproveitando o e-commerce para alcançar consumidores em diferentes regiões, muitas vezes personalizando suas estratégias de marketing conforme as preferências culturais.

Ênfase na Sustentabilidade e Ingredientes Naturais

Houve uma crescente demanda por produtos de beleza sustentáveis e com ingredientes naturais. Empresas que enfatizaram esses atributos em suas linhas de produtos viram um aumento na preferência do consumidor.

Programas de Fidelidade e Ofertas Exclusivas

Programas de fidelidade e ofertas exclusivas foram estratégias comuns para incentivar a repetição de compras. Muitas marcas lançaram assinaturas mensais e programas de recompensas para fidelizar clientes.

Desafios com Devoluções e Testes Online

O setor enfrentou desafios relacionados a devoluções de produtos e a necessidade de encontrar soluções inovadoras para permitir que os clientes testassem virtualmente produtos que, tradicionalmente, eram experimentados em lojas físicas.

Esses insights refletem a dinâmica do e-commerce no setor de perfumaria e cosméticos em 2018, marcado por inovações tecnológicas, foco na experiência do cliente e a busca por diferenciação em um mercado cada vez mais competitivo.

2º Moda e acessórios: 13,6%

Ascensão do Mobile e-Commerce

Houve um aumento significativo nas transações de moda via dispositivos móveis, destacando a importância de plataformas otimizadas para smartphones e estratégias de marketing direcionadas a usuários de dispositivos móveis.

Experiência de Compra Personalizada

Empresas investiram em tecnologias de personalização para oferecer uma experiência de compra mais individualizada, incluindo recomendações de produtos com base no histórico de compras, preferências de estilo e comportamento do consumidor.

Influenciadores e Redes Sociais

A influência das redes sociais na moda online cresceu, com marcas utilizando influenciadores digitais para promover produtos, criar tendências e aumentar o alcance da audiência.
Tecnologias de Realidade Aumentada (RA):
Algumas plataformas de e-commerce de moda incorporaram tecnologias de realidade aumentada para permitir que os clientes visualizassem virtualmente roupas e acessórios, melhorando a experiência de compra online.

Sustentabilidade e Responsabilidade Social

A conscientização sobre práticas sustentáveis na indústria da moda aumentou, levando algumas empresas a destacarem suas iniciativas ecológicas e responsabilidade social para atrair consumidores preocupados com essas questões.

Compra por Impulso e Ofertas Relâmpago

Estratégias de marketing focadas em compras por impulso e ofertas relâmpago foram comuns, estimulando a sensação de urgência e incentivando a rápida tomada de decisão por parte dos consumidores.

Inteligência Artificial (IA) na Moda

Algumas plataformas incorporaram inteligência artificial para análise de tendências de moda, sugerindo estilos e previsões de moda com base em grandes conjuntos de dados e padrões de consumo.

Esses insights refletem a dinâmica do e-commerce no setor de moda e acessórios em 2018, caracterizado por inovações tecnológicas, foco na personalização da experiência do cliente e a crescente influência das redes sociais na tomada de decisões de compra.

3º Casa e decoração: 11,1%

Digitalização da Experiência de Compra

Houve uma crescente digitalização da experiência de compra no setor, com consumidores buscando inspiração online, explorando catálogos virtuais e efetuando compras por meio de plataformas de e-commerce especializadas.

Realidade Aumentada na Visualização de Produtos

Tecnologias de realidade aumentada foram incorporadas por algumas empresas para permitir que os consumidores visualizassem como os produtos de decoração ficariam em seus próprios espaços, melhorando a confiança na compra online.

Personalização e Customização

A tendência de personalização e customização de produtos ganhou destaque, permitindo que os consumidores criassem itens exclusivos de decoração de acordo com suas preferências e estilos.

Conteúdo Visual e Inspiracional

Empresas investiram em conteúdo visual e inspiracional, como tutoriais de decoração, blogs e vídeos, para envolver os consumidores e oferecer ideias práticas para transformar seus espaços.

Programas de Fidelidade e Descontos Exclusivos

Programas de fidelidade e descontos exclusivos foram estratégias comuns para incentivar a lealdade do cliente. Muitas marcas ofereceram benefícios especiais para clientes recorrentes e membros de programas de fidelidade.

Integração com Redes Sociais

A integração com redes sociais foi uma prática comum, permitindo que os consumidores compartilhassem suas compras e inspirações de decoração, gerando engajamento e reconhecimento de marca.

Análise de Dados para Tendências de Decoração

Empresas utilizaram análise de dados para identificar e antecipar tendências de decoração, adaptando seus estoques e ofertas de acordo com as preferências dos consumidores.

- Quais as melhorias relevantes para a logística?

Tempo de entrega x valor do frete

O prazo de entrega tornou-se um fator crucial no e-commerce, impactando diretamente a decisão de compra dos consumidores. Além do preço e qualidade do produto, a velocidade de entrega é um elemento-chave para conquistar clientes. Segundo um relatório do E-commerce Brasil, 50% dos usuários consideram o prazo de entrega como o principal influenciador na decisão de compra, e estima-se que até 2026, 25% das compras online serão entregues no mesmo dia. A ênfase no relatório recai sobre a necessidade de otimizar os processos de pós-venda para garantir prazos realistas e evitar descontentamento por parte dos consumidores. O desafio logístico no Brasil destaca a importância de investir em soluções eficientes para atender às expectativas crescentes dos clientes em relação à rapidez nas entregas.

Examinar a interação entre o prazo de entrega e o custo do frete no âmbito do e-commerce é crucial. Ao equilibrar de forma eficiente o tempo de entrega e o valor do frete, os varejistas de e-commerce podem oferecer uma experiência de compra mais satisfatória para os clientes, aumentar a fidelidade e permanecer competitivos no mercado.

Transparência na Informação

Fornecer informações claras sobre as opções de entrega disponíveis e os custos associados desde o início do processo de compra.

Indicar de maneira transparente as condições para frete grátis, se aplicável, e os prazos estimados de entrega.

Opções de Entrega Flexíveis

Oferecer opções de entrega flexíveis, como entrega expressa, entrega agendada ou entrega em pontos de coleta, para atender às diversas necessidades dos clientes.

Permitir que os clientes escolham entre opções de entrega mais rápidas pagando um valor adicional ou optem por prazos de entrega mais longos com custos de frete mais baixos.

Programas de Fidelidade e Benefícios para Frete

Criar ofertas especiais, como frete grátis para pedidos acima de um determinado valor, incentivando os clientes a aumentar o valor do carrinho de compras.

Armazenagem de carga em locais estratégicos

Em relação a diversas classificações para galpões logísticos ou Centros de Distribuição (CD), sua função primordial é a armazenagem estratégica de carga. É fundamental diferenciar entre armazenagem e estocagem, sendo esta última responsável por criar um estoque, enquanto a armazenagem se concentra no processo de armazenar mercadorias movimentadas frequentemente.

A localização estratégica dos CDs é de extrema importância, especialmente quando se refere à proximidade das áreas metropolitanas. Observa-se uma tendência significativa de grandes empresas de e-commerce concentrando investimentos em centros de distribuição localizados próximos aos principais pontos de consumo em todo o país.

Essa movimentação não apenas torna as grandes varejistas menos dependentes de serviços de entrega externos, mas também visa expandir suas operações por meio de armazéns menores em regiões urbanas próximas às suas lojas.

Algumas empresas que estão aprimorando a eficiência dos sistemas logísticos

A Via Varejo, possui o segundo maior centro de distribuição global em Jundiaí (SP).A Magazine Luiza inaugurou um CD no Rio Grande do Sul e está finalizou outro em Gravataí. Empresas como Amazon e Mercado Livre já adotaram essa estratégia, tornando os centros de distribuição um diferencial competitivo para entregas rápidas. Além disso, destaca-se a aquisição de pequenas lojas próximas da falência e processos de fusão e aquisições em tecnologia e logística

Rastreamento em Tempo Real

Fornecer informações de rastreamento em tempo real para que os clientes possam monitorar o status de suas entregas.

Enviar notificações proativas sobre atualizações no status do pedido, como saída do armazém, chegada à cidade de destino, etc.

Negociações com Transportadoras

Negociar tarifas competitivas com transportadoras para obter custos de frete mais vantajosos.

Avaliar parcerias estratégicas com empresas de logística para garantir tarifas preferenciais e eficiência nas entregas.

Avaliação de Desempenho

Realizar análises periódicas do desempenho logístico, considerando métricas como tempo médio de entrega, taxa de entregas no prazo e satisfação do cliente.

Utilizar feedback dos clientes para identificar áreas de melhoria e ajustar estratégias de entrega conforme necessário.

Educação do Cliente

Oferecer informações claras sobre como as escolhas do cliente influenciam o tempo de entrega e o custo do frete.

Inovação Tecnológica

Explorar soluções tecnológicas, como drones ou entregas automatizadas, para reduzir os tempos de entrega e otimizar os custos operacionais.

Fonte: https://agenciagnu.com.br/o-impacto-do-frete-no-e-commerce-e-como-otimiza-lo-com-eficiencia/

Recomendações

Com base nos insights apresentados, as empresas de e-commerce podem adotar as seguintes recomendações para melhorar seus resultados:

- Oferecer opções de pagamento diversificadas: as empresas devem oferecer uma variedade de opções de pagamento para atender às diferentes necessidades dos consumidores.
- Investir em tecnologias emergentes: as empresas devem estar atentas às tecnologias emergentes para oferecer soluções de pagamento inovadoras.
- Melhorar a experiência do cliente: as empresas devem se concentrar em melhorar a experiência do cliente, incluindo a facilidade de pagamento.

Ações específicas para impulsionar as vendas no e-commerce

Para impulsionar as vendas no e-commerce, as empresas devem se concentrar nas seguintes ações específicas:

- Melhorar a experiência do usuário: as empresas devem criar um site intuitivo e fácil de navegar, com um layout claro e uma navegação simples. O site também

deve ser otimizado para dispositivos móveis, pois cada vez mais pessoas estão comprando online por meio de smartphones e tablets.

- Oferecer frete transparente e opções de entrega claras: as empresas devem informar os clientes sobre os custos e prazos de entrega antes do checkout. Também devem oferecer uma variedade de opções de entrega, para atender às diferentes necessidades dos clientes.
- Implementar programas de fidelidade: os programas de fidelidade são uma ótima maneira de recompensar os clientes fiéis e incentivar as compras repetidas.
- Aproveitar as redes sociais: as redes sociais são uma ótima maneira de alcançar novos clientes e aumentar o reconhecimento da marca. As empresas devem criar conteúdo relevante e envolvente para seus seguidores.
- Destacar avaliações de clientes: as avaliações de clientes são uma ótima maneira de demonstrar a confiabilidade e a qualidade de um produto ou serviço. As empresas devem destacar as avaliações positivas em seu site e nas redes sociais.
- Realizar promoções: as promoções são uma ótima maneira de atrair novos clientes e incentivar as compras repetidas. As empresas devem realizar promoções regulares, com ofertas atraentes.
- Utilizar estratégias de remarketing: as estratégias de remarketing são uma ótima maneira de alcançar usuários que visitaram o site da empresa, mas não realizaram uma compra. As empresas podem usar o remarketing para exibir anúncios direcionados aos usuários que abandonaram o carrinho de compras ou que visualizaram um produto específico.

Análise contínua de dados

As empresas devem analisar continuamente dados de desempenho para ajustar e otimizar sua abordagem. A análise de dados pode ajudar as empresas a identificar tendências, oportunidades e áreas de melhoria.

Considerações finais

O e-commerce é um mercado em constante evolução. As empresas de e-commerce devem estar atentas às tendências do mercado e às necessidades dos consumidores para permanecerem competitivas. A análise de dados é uma ferramenta essencial para as empresas de e-commerce que desejam obter insights que possam ajudá-las a tomar melhores decisões.

Correlação com a Olist e Recomendações para Melhoria:

Considerando as recomendações apresentadas, a Olist, como empresa de e-commerce, pode adotar medidas específicas para aprimorar seu desempenho no mercado. Algumas sugestões alinhadas aos insights mencionados são:

Investimento em Tecnologias Emergentes:

Para permanecer na vanguarda do setor, a Olist deve estar atenta às tecnologias emergentes no e-commerce. A integração de soluções inovadoras de pagamento e automação pode diferenciar a empresa e atrair consumidores que buscam experiências modernas.

Melhoria da Experiência do Cliente:

Concentrando-se na otimização da experiência do cliente, a Olist pode investir na criação de um site intuitivo e amigável, otimizado para dispositivos móveis. A facilidade no processo de pagamento é crucial, e a simplificação do fluxo de compra pode melhorar significativamente a satisfação do cliente.

Frete Transparente e Opções de Entrega Claras:

A transparência nas informações sobre custos e prazos de entrega é fundamental. A Olist pode implementar uma política de frete transparente, fornecendo opções claras de entrega e comunicando essas informações de maneira proativa aos clientes.

Programas de Fidelidade Personalizados:

A introdução de programas de fidelidade específicos para a Olist pode ser uma estratégia eficaz para recompensar clientes fiéis e incentivar compras repetidas. Ofertas personalizadas com base no histórico de compras podem fortalecer o vínculo com os consumidores.

Estratégias de Marketing em Redes Sociais:

A Olist pode ampliar sua presença nas redes sociais, criando conteúdo relevante e envolvente para sua audiência. A utilização de plataformas como Instagram e Facebook pode ser uma maneira eficaz de atrair novos clientes e fortalecer o reconhecimento da marca.

Destaque de Avaliações de Clientes:

Destacar avaliações positivas em seu site e em plataformas de redes sociais pode construir confiança. A Olist pode incentivar os clientes a deixarem feedback e utilizar essas avaliações como uma ferramenta de marketing, destacando a qualidade de seus produtos e serviços.

Realização de Promoções Estratégicas:

A realização de promoções regulares, com ofertas atrativas, pode ser uma estratégia para atrair novos clientes e impulsionar as vendas. A Olist pode planejar campanhas sazonais e promover ofertas especiais para períodos específicos.

Utilização de Estratégias de Remarketing:

Implementar estratégias de remarketing pode ser crucial para recuperar potenciais clientes que abandonaram o carrinho de compras. A Olist pode criar anúncios direcionados a usuários que demonstraram interesse em produtos específicos, incentivando a conclusão da compra.

Análise Contínua de Dados:

A Olist deve adotar uma abordagem orientada por dados, analisando continuamente o desempenho de suas operações. A coleta e interpretação de dados podem proporcionar insights valiosos para ajustar estratégias, identificar tendências de mercado e áreas de melhoria.

Em conclusão, ao alinhar suas práticas com as recomendações mencionadas e adaptar essas estratégias ao contexto específico da Olist, a empresa estará mais bem posicionada para enfrentar os desafios e explorar as oportunidades no dinâmico mercado de e-commerce.

Fontes: https://materiais.opinionbox.com/infografico-pesquisa-meios-de-pagamento

https://tiinside.com.br/04/12/2018/sbvc-divulga-ranking-dos-50-maiores-e-commerces-no-brasil/

https://www.ecommercebrasil.com.br/artigos/estrategia-centros-de-distribuicao-e-commerce