Conjunto de Dados da Olist

Este conjunto de dados detalha mais de 100.000 pedidos da Olist Store, cobrindo o período de 2016 a 2018. As informações foram coletadas em várias etapas da jornada do cliente, e incluem:

Status do pedido: o estado atual de cada pedido, desde a compra até a entrega.

Preço e frete: os custos associados a cada pedido.

Desempenho da entrega: informações sobre o tempo de envio e a eficiência da logística.

Localização: dados geográficos dos clientes e vendedores, permitindo análises regionais.

Avaliações e comentários: notas de satisfação e feedback escrito dos clientes.

Objetivos e Análises Possíveis

O principal objetivo do conjunto de dados é permitir uma análise aprofundada do mercado de e-commerce brasileiro. Com ele, é possível:

Entender o comportamento do cliente ao longo do tempo.

Analisar a satisfação do cliente e como ela se relaciona com fatores como tempo de entrega e localização.

Mapear a distribuição geográfica de vendedores e clientes no Brasil.

Identificar os fatores que influenciam a avaliação de um cliente.

Inicio do Projeto - Carregando os Dados

```
import pandas as pd

df_clientes = pd.read_csv('https://harve.com.br/praticas/olist/olist_customers_dataset.csv')
df_geolocalizacoes = pd.read_csv('https://harve.com.br/praticas/olist/olist_geolocation_dataset.csv')
df_itens_pedidos = pd.read_csv('https://harve.com.br/praticas/olist/olist_order_items_dataset.csv')
df_pagamentos = pd.read_csv('https://harve.com.br/praticas/olist/olist_order_payments_dataset.csv')
df_avaliacoes = pd.read_csv('https://harve.com.br/praticas/olist/olist_order_reviews_dataset.csv')
df_pedidos = pd.read_csv('https://harve.com.br/praticas/olist/olist_orders_dataset.csv')
df_produtos = pd.read_csv('https://harve.com.br/praticas/olist/olist_products_dataset.csv')
df_vendedores = pd.read_csv('https://harve.com.br/praticas/olist/olist_sellers_dataset.csv')
df_categoria_produtos = pd.read_csv('https://harve.com.br/praticas/olist/product_category_name_translation.csv')
```

Verificando quantos por cento dos pedidos tiveram atrasos?

```
# convertendo as data caso elas não forem convertidas
df_pedidos['order_delivered_customer_date'] = pd.to_datetime(df_pedidos['order_delivered_customer_date'])
df_pedidos['order_estimated_delivery_date'] = pd.to_datetime(df_pedidos['order_estimated_delivery_date'])

# Filtrando os pedidos que foram entregues
df_entregues = df_pedidos[df_pedidos['order_status'] == 'delivered'].copy()

# Contando  número de pedidos com atraso
pedidos_atrasados = df_entregues[df_entregues['order_delivered_customer_date'] > df_entregues['order_estimated_delivery_date'
numero_pedidos_atrasados = len(pedidos_atrasados)

# Obtendo o número total de pedidos
numero_total_pedidos = len(df_pedidos)

# Calculando a porcentagem
porcentagem_atraso = (numero_pedidos_atrasados / numero_total_pedidos) * 100

# Resultado
print(f"### Porcentagem de pedidos com atraso: {porcentagem_atraso:.2f}%")
```

```
### Porcentagem de pedidos com atraso: 7.87%
```

Foi levantada uma possível falha no sistema que permitia que pedidos com o status delivered não tivessem data de entrega no sistema. Consegue verificar se isso aconteceu? E em quantos casos isso ocorreu?

```
# Filtrando pedidos com o status 'delivered'
df_entregues = df_pedidos[df_pedidos['order_status'] == 'delivered'].copy()

# Verificando se a coluna de data de entrega existe
if 'order_delivered_customer_date' in df_entregues.columns:
    # Contando os casos em que a data de entrega está ausente
    casos_sem_data_entrega = df_entregues['order_delivered_customer_date'].isnull().sum()

    # Exibindo o resultado
    if casos_sem_data_entrega > 0:
        print(f"Sim, a falha ocorreu. Foram encontrados {casos_sem_data_entrega} pedidos com o status 'Entregue' sem uma data
    else:
        print("Não, a falha não foi encontrada. Nenhum pedido com o status 'Entregue' está sem data de entrega.")
else:
    print("A coluna 'order_delivered_customer_date' não foi encontrada no DataFrame. Por favor, verifique o nome da coluna e
```

```
Sim, a falha ocorreu. Foram encontrados 8 pedidos com o status 'Entregue' sem uma data de entrega registrada.
```

A equipe de logística quer saber qual é a dimensão máxima que uma caixa deve ter para não ter risco de não atender algum produto.

```
# Encontrando as dimensões máximas
max_altura = df_produtos['product_height_cm'].max()
max_largura = df_produtos['product_width_cm'].max()
max_comprimento = df_produtos['product_length_cm'].max()
max_peso = df_produtos['product_weight_g'].max()

# Exibindo as dimensões máximas para a caixa
print(f"### Para atender todos os produtos, a caixa deve ter as seguintes dimensões mínimas:")
print(f"Altura máxima: {max_altura:.2f} cm")
print(f"Largura máxima: {max_largura:.2f} cm")
print(f"Comprimento máximo: {max_comprimento:.2f} cm")
print(f"Peso máximo: {max_peso:.2f} g")
```

```
### Para atender todos os produtos, a caixa deve ter as seguintes dimensões mínimas:
Altura máxima: 105.00 cm
Largura máxima: 118.00 cm
Comprimento máximo: 105.00 cm
Peso máximo: 40425.00 g
```

Os pedidos estão crescendo ao longo do tempo? Eles estão estáveis em 2018 ou estão em ritmo de crescimento? Qual mês constatou o maior número de vendas?

```
# 1. Convertendo a coluna de data para o formato datetime
df_pedidos['order_purchase_timestamp'] = pd.to_datetime(df_pedidos['order_purchase_timestamp'])

# 2. Verificando o crescimento dos pedidos ao longo do tempo (por mês/ano)
pedidos_por_mes = df_pedidos.set_index('order_purchase_timestamp').resample('ME').size()
print("### Pedidos por mês ao longo do tempo:")
pedidos_ordenados = pedidos_por_mes.sort_values(ascending=True)
print(pedidos_ordenados)

# 3. Analisando o ritmo de crescimento em 2018
pedidos_2018 = df_pedidos[df_pedidos['order_purchase_timestamp'].dt.year == 2018]
pedidos_2018_por_mes = pedidos_2018.set_index('order_purchase_timestamp').resample('ME').size()
print("\n### Pedidos por mês em 2018:")
pedidos_ordenados_2018 = pedidos_2018_por_mes.sort_values(ascending=True)
print(pedidos_ordenados_2018)

# 4. Encontrando o mês com o maior número de vendas
mes_com_mais_vendas = pedidos_por_mes.idxmax()
numero_vendas_max = pedidos_por_mes.max()
print(f"\nO mês com o maior número de vendas foi: {mes_com_mais_vendas.strftime('%m-%Y')}, com um total de {numero_vendas_max
```

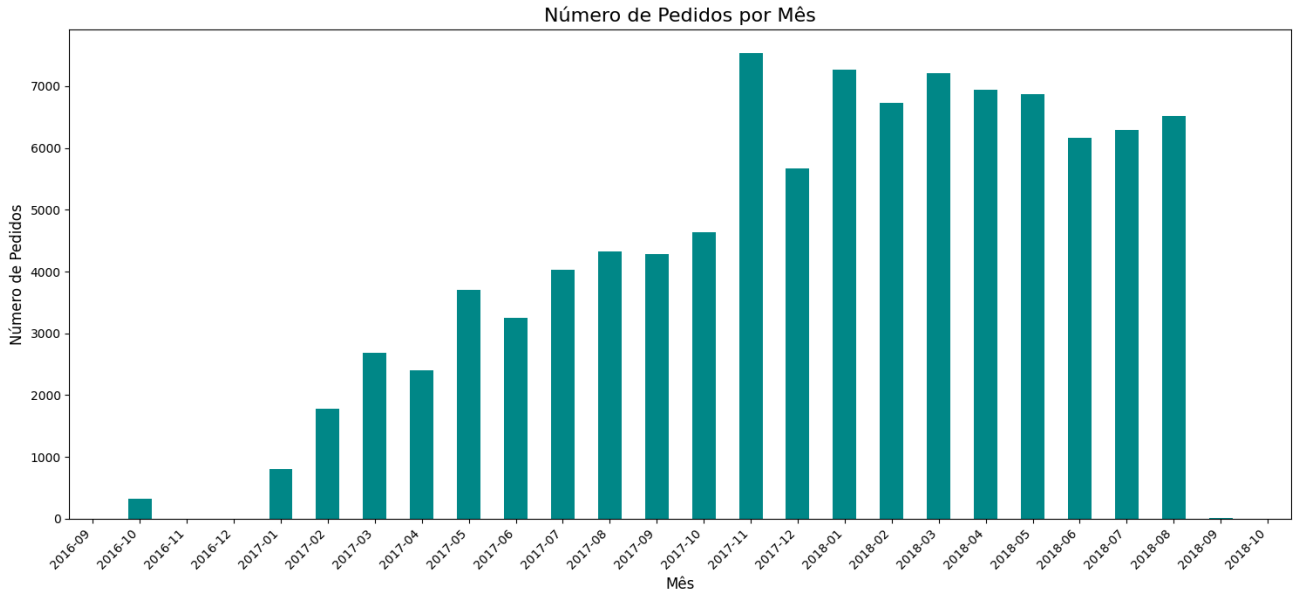
Mostrar saída oculta

Clique duas vezes (ou pressione "Enter") para editar

```
import matplotlib.pyplot as plt

# A linha de ordenação é removida para que os dados fiquem em ordem cronológica
# pedidos_ordenados = pedidos_por_mes.sort_values(ascending=True)
```

```
# O `resample` já retorna os dados em ordem de data, então
# usamos `pedidos_por_mes` diretamente
pedidos_por_mes.index = pedidos_por_mes.index.strftime('%Y-%m')

# Criando o gráfico de barras
plt.figure(figsize=(15, 7))
pedidos_por_mes.plot(kind='bar', color='darkcyan')

# Configurando o título e os rótulos dos eixos
plt.title('Número de Pedidos por Mês', fontsize=16)
plt.xlabel('Mês', fontsize=12)
plt.ylabel('Número de Pedidos', fontsize=12)

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

# Ajustando o layout para evitar que os rótulos sejam cortados
plt.tight_layout()

# Salvando o gráfico em um arquivo de imagem
plt.savefig('pedidos_por_mes.png')

print("O gráfico foi salvo como 'pedidos_por_mes.png', ordenado por data.")
```

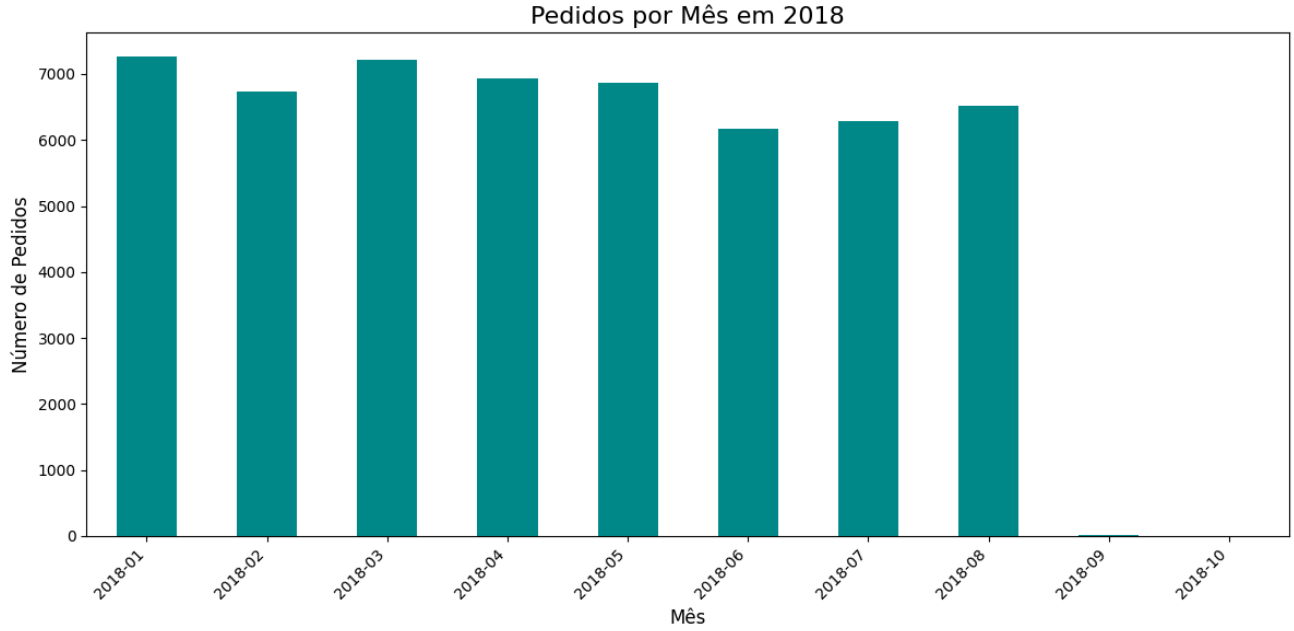
```
O gráfico foi salvo como 'pedidos_por_mes.png', ordenado por data.
```




```
import pandas as pd
import matplotlib.pyplot as plt

try:
    # 1. Certifique-se de que a coluna de data está no formato correto
    df_pedidos['order_purchase_timestamp'] = pd.to_datetime(df_pedidos['order_purchase_timestamp'])

    # 2. Filtrar os dados para o ano de 2018
    pedidos_2018 = df_pedidos[df_pedidos['order_purchase_timestamp'].dt.year == 2018].copy()

    # 3. **Definir a coluna de data como o índice do DataFrame**
    # Este é o passo crucial para o `resample` funcionar
    pedidos_2018.set_index('order_purchase_timestamp', inplace=True)

    # 4. Contar os pedidos por mês em 2018
    pedidos_2018_por_mes = pedidos_2018.resample('ME').size()

    # 5. Verificar se há dados para plotar
    if not pedidos_2018_por_mes.empty:
```

```
        # Agora o .strftime() funcionará, pois o índice é do tipo DatetimeIndex
        pedidos_2018_por_mes.index = pedidos_2018_por_mes.index.strftime('%Y-%m')

        # Criando o gráfico de barras
        plt.figure(figsize=(12, 6))
        pedidos_2018_por_mes.plot(kind='bar', color='darkcyan')

        # Configurando título e rótulos
        plt.title('Pedidos por Mês em 2018', fontsize=16)
        plt.xlabel('Mês', fontsize=12)
        plt.ylabel('Número de Pedidos', fontsize=12)

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

        # Ajustando o layout
        plt.tight_layout()

        # Salvando o gráfico em um arquivo de imagem
        plt.savefig('pedidos_2018.png')

        print("O gráfico foi salvo como 'pedidos_2018.png'.")
    else:
        print("Não há pedidos registrados para o ano de 2018 para gerar o gráfico.")

except KeyError:
    print("Erro: Verifique se a coluna 'order_purchase_timestamp' existe no seu DataFrame.")
```

```
O gráfico foi salvo como 'pedidos_2018.png'.
```



Retornando as 3 categorias com pior e melhor média de avaliação

```
import pandas as pd

# 1. Unindo os DataFrames
# Primeiro, unimos df_avaliacoes e df_itens_pedidos pelo 'order_id'
df_merged = pd.merge(df_avaliacoes, df_itens_pedidos, on='order_id', how='inner')

# Em seguida, unimos o resultado com df_produtos pelo 'product_id'
df_final = pd.merge(df_merged, df_produtos, on='product_id', how='inner')

# 2. Calculando a média de avaliação por categoria
media_por_categoria = df_final.groupby('product_category_name')['review_score'].mean()

# 3. Retornar as 3 piores e as 3 melhores categorias
piores_avaliacoes = media_por_categoria.nsmallest(3)
melhores_avaliacoes = media_por_categoria.nlargest(3)
```

```
print("### As Categorias com as Piores Médias de Avaliação ###")
print(piores_avaliacoes.to_string())

print("\n### As 3 Categorias com as Melhores Médias de Avaliação ###")
print(melhores_avaliacoes.to_string())
```

```
### As Categorias com as Piores Médias de Avaliação ###
product_category_name
seguros_e_servicos                              2.500000
fraldas_higiene                                 3.256410
portateis_cozinha_e_preparadores_de_alimentos   3.266667

### As 3 Categorias com as Melhores Médias de Avaliação ###
product_category_name
cds_dvds_musicais               4.642857
fashion_roupa_infanto_juvenil   4.500000
livros_interesse_geral          4.439421
```

Qual é a média em dias do tempo de entrega? E qual foi a entrega mais longa em dias?

```
import pandas as pd

# 1. Convertendo as colunas de data para o formato datetime
df_pedidos['order_purchase_timestamp'] = pd.to_datetime(df_pedidos['order_purchase_timestamp'])
df_pedidos['order_delivered_customer_date'] = pd.to_datetime(df_pedidos['order_delivered_customer_date'])

# 2. Calculando o tempo de entrega em dias
# O cálculo só será feito para pedidos com data de entrega registrada, por isso o .dropna()
df_pedidos['tempo_entrega'] = (df_pedidos['order_delivered_customer_date'] - df_pedidos['order_purchase_timestamp']).dt.days

# 3. Calculando a média e a entrega mais longa
media_entrega = df_pedidos['tempo_entrega'].mean()
entrega_mais_longa = df_pedidos['tempo_entrega'].max()

# 4. Exibindo os resultados
print(f"O tempo médio de entrega é de {media_entrega:.0f} dias.")
print(f"O tempo de entrega mais longo registrado foi de {entrega_mais_longa:.0f} dias.")
```

```
O tempo médio de entrega é de 12 dias.
O tempo de entrega mais longo registrado foi de 209 dias.
```

Qual é o tipo de pagamento mais comum?

```
# coluna 'payment_type' e os tipos
if 'payment_type' in df_pagamentos.columns:
    tipos_pagamento = df_pagamentos['payment_type'].value_counts()

    # 2. Encontrando o tipo de pagamento mais comum o primeiro da lista
    pagamento_mais_comum = tipos_pagamento.index[0]

    # 3. Exibindo o resultado
    print("### tipos de pagamento:")
    print(tipos_pagamento.to_string())
    print(f"\nO tipo de pagamento mais comum é: {pagamento_mais_comum}")
else:
    print("A coluna 'payment_type' não foi encontrada no DataFrame. Por favor, verifique o nome da coluna e tente novamente."
```

```
### tipos de pagamento:
payment_type
credit_card    76795
boleto         19784
voucher         5775
debit_card      1529
not_defined        3

O tipo de pagamento mais comum é: credit_card
```

Quais são os produtos que mais venderam em receita?

```
import pandas as pd

try:
    # Unindo os DataFrames para vincular o preço ao produto
```

```
    df_merged_receita = pd.merge(df_itens_pedidos, df_produtos, on='product_id', how='inner')

    # Calculando a receita total para cada produto
    receita_por_produto = df_merged_receita.groupby('product_id')['price'].sum().reset_index()

    # Unindo a receita com as informações do produto
    df_top_produtos = pd.merge(receita_por_produto, df_produtos[['product_id', 'product_category_name']], on='product_id', hou

    # Ordenando a receita do maior para o menor e selecionar os 10 primeiros
    top_10_produtos_com_nome = df_top_produtos.sort_values(by='price', ascending=False).head(10)

    # Exibindo o resultado
    print("Os 10 produtos que mais venderam por receita:")
    print(top_10_produtos_com_nome[['product_category_name', 'price']].to_string(index=False))

except NameError:
    print("Erro: Certifique-se de que os DataFrames 'df_itens_pedido' e 'df_produtos' estão carregados.")
except KeyError as e:
    print(f"Erro: A coluna {e} não foi encontrada. Por favor, verifique os nomes das colunas e tente novamente.")
```

```
Os 10 produtos que mais venderam por receita:
 product_category_name    price
          beleza_saude 63885.00
          beleza_saude 54730.20
                   pcs 48899.34
informatica_acessorios 47214.51
       cama_mesa_banho 43025.56
informatica_acessorios 41082.60
                 bebes 38907.32
            cool_stuff 37733.90
    relogios_presentes 37683.42
      moveis_decoracao 37608.90
```

Qual é a média de número de pedidos por cliente?

```
try:
    # Contando o número de pedidos únicos por cliente
    pedidos_por_cliente = df_pedidos.groupby('customer_id')['order_id'].nunique()

    # Calculando a média do número de pedidos por cliente
    media_pedidos_por_cliente = pedidos_por_cliente.mean()

    print(f"A média de pedidos por cliente é: {media_pedidos_por_cliente:.2f}")

except KeyError as e:
    print(f"Erro: A coluna {e} não foi encontrada no DataFrame. Por favor, verifique os nomes das colunas e tente novamente."
```

```
A média de pedidos por cliente é: 1.00
```

Quais são os estados atuais dos clientes e quantos pedidos temos de cada estado?

```
import pandas as pd

try:
    # Unindo os DataFrames de pedidos e clientes
    df_merged = pd.merge(df_pedidos, df_clientes, on='customer_id', how='inner')

    # Contando o número de pedidos por estado
    pedidos_por_estado = df_merged['customer_state'].value_counts()

    # Exibindo o resultado
    print("Número de pedidos por estado:")
    print(pedidos_por_estado.to_string())

except NameError:
    print("Erro: Certifique-se de que os DataFrames 'df_pedidos' e 'df_clientes' estão carregados.")
except KeyError as e:
    print(f"Erro: A coluna {e} não foi encontrada. Por favor, verifique os nomes das colunas e tente novamente.")
```

```
Número de pedidos por estado:
customer_state
SP    41746
RJ    12852
MG    11635
RS     5466
PR     5045
SC     3637
BA     3380
DF     2140
```

```
ES     2033
GO     2020
PE     1652
CE     1336
PA      975
MT      907
MA      747
MS      715
PB      536
PI      495
RN      485
AL      413
SE      350
TO      280
RO      253
AM      148
AC       81
AP       68
RR       46
```

Quantos pedidos foram cancelados?

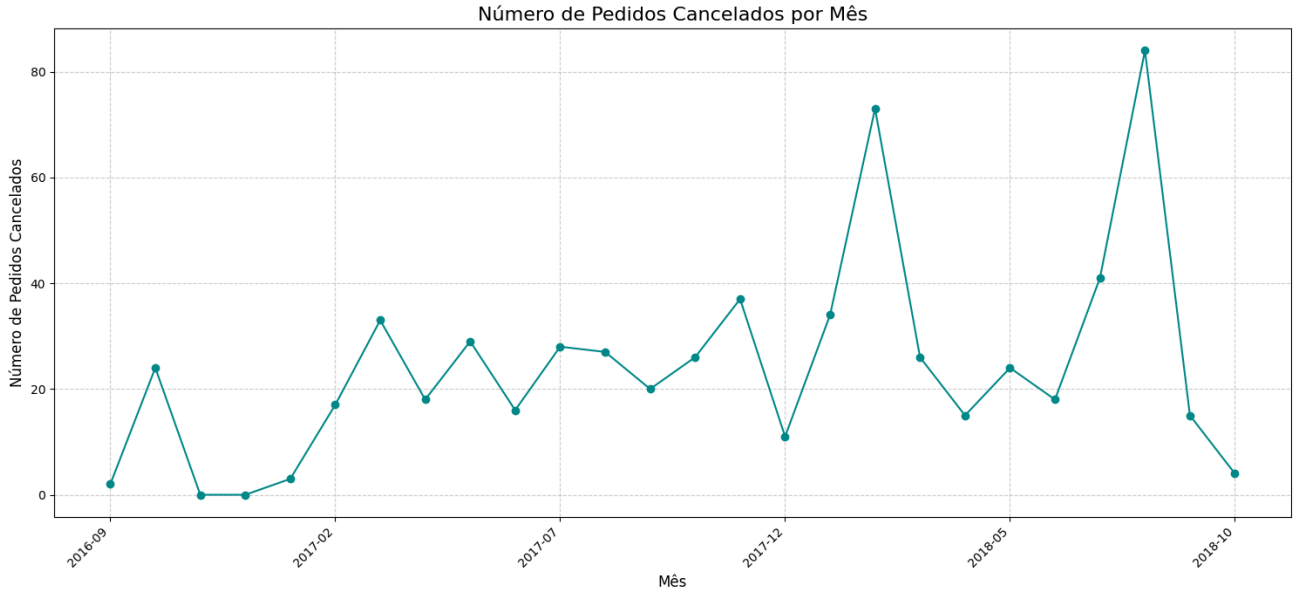
```
try:
    # Contando o número de pedidos onde o status é 'canceled'
    pedidos_cancelados = df_pedidos[df_pedidos['order_status'] == 'canceled']
    numero_pedidos_cancelados = len(pedidos_cancelados)

    print(f"O número total de pedidos cancelados é: {numero_pedidos_cancelados}")

except KeyError as e:
    print(f"Erro: A coluna 'order_status' não foi encontrada. Por favor, verifique o nome da coluna e tente novamente.")
```

```
O número total de pedidos cancelados é: 625
```

Gere um gráfico exibindo os pedidos cancelados por mês

```
import pandas as pd
import matplotlib.pyplot as plt

try:
    # 1. Filtrando os pedidos que foram cancelados
    pedidos_cancelados = df_pedidos[df_pedidos['order_status'] == 'canceled'].copy()

    # 2. Convertendo a coluna de data para o formato datetime
    pedidos_cancelados['order_purchase_timestamp'] = pd.to_datetime(pedidos_cancelados['order_purchase_timestamp'])

    # 3. Contando os pedidos cancelados por mês (o resample já ordena por data)
    cancelamentos_por_mes = pedidos_cancelados.set_index('order_purchase_timestamp').resample('ME').size()

    # 4. Formatando o índice para exibir apenas o ano e o mês
    cancelamentos_por_mes.index = cancelamentos_por_mes.index.strftime('%Y-%m')

    # 5. Criando o gráfico de linha
    # Apenas mudei 'bar' para 'line' e adicionei marcadores
    plt.figure(figsize=(15, 7))
    cancelamentos_por_mes.plot(kind='line', marker='o', color='darkcyan')

    # 6. Configurando o título e os rótulos dos eixos
    plt.title('Número de Pedidos Cancelados por Mês', fontsize=16)
    plt.xlabel('Mês', fontsize=12)
    plt.ylabel('Número de Pedidos Cancelados', fontsize=12)

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

    # 8. Adicionando uma grade para facilitar a leitura
    plt.grid(True, linestyle='--', alpha=0.6)

    # 9. Ajustando o layout
    plt.tight_layout()

    # 10. Salvando o gráfico em um arquivo de imagem
    plt.savefig('pedidos_cancelados_por_mes_linha.png')

    print("O gráfico de linha de pedidos cancelados por mês foi salvo.")

except KeyError:
    print("Erro: Verifique se as colunas 'order_status' e 'order_purchase_timestamp' existem no seu DataFrame.")
```

```
O gráfico de linha de pedidos cancelados por mês foi salvo.
```



Qual é a média do custo de frete por estado do cliente?

```
import pandas as pd

try:
    # 1. Unindo os DataFrames para vincular o frete ao estado do cliente
    df_merged = pd.merge(df_pedidos, df_clientes, on='customer_id', how='inner')
    df_final = pd.merge(df_merged, df_itens_pedidos, on='order_id', how='inner')

    # 2. Calcular a média do custo de frete por estado
    media_frete_por_estado = df_final.groupby('customer_state')['freight_value'].mean().sort_values(ascending=False)

    # 3. Exibir o resultado
    print("Média do custo de frete por estado (ordenado do maior para o menor):")
    print(media_frete_por_estado.to_string())

except NameError:
    print("Erro: Certifique-se de que os DataFrames 'df_pedidos', 'df_clientes' e 'df_itens_pedido' estão carregados.")
except KeyError as e:
    print(f"Erro: A coluna {e} não foi encontrada. Por favor, verifique os nomes das colunas e tente novamente.")
```

```
Média do custo de frete por estado (ordenado do maior para o menor):
customer_state
RR    42.984423
PB    42.723804
RO    41.069712
AC    40.073370
PI    39.147970
MA    38.257002
TO    37.246603
SE    36.653169
AL    35.843671
PA    35.832685
RN    35.652363
AP    34.006098
AM    33.205394
PE    32.917863
CE    32.714202
MT    28.166284
BA    26.363959
MS    23.374884
GO    22.766815
ES    22.058777
```

```
RS    21.735804
SC    21.470369
DF    21.041355
RJ    20.960924
MG    20.630167
PR    20.531652
SP    15.147275
```

Quantos por cento dos vendedores tiveram avaliação menor que 2?

```
import pandas as pd

try:
    # 1. Unindo os DataFrames para vincular a avaliação ao vendedor
    df_merged = pd.merge(df_avaliacoes, df_itens_pedidos, on='order_id', how='inner')

    # 2. Encontrar a avaliação mínima por vendedor para evitar duplicações de avaliações por pedido.
    # No entanto, a pergunta é sobre 'avaliação menor que 2', o que implica que basta ter uma avaliação menor que 2.
    # Logo, vamos apenas filtrar as avaliações e pegar os vendedores únicos

    # 3. Filtrar avaliações com nota menor que 2
    avaliacoes_baixas = df_merged[df_merged['review_score'] < 2].copy()

    # 4. Contar o número de vendedores únicos com avaliação baixa
    vendedores_unicos_com_nota_baixa = avaliacoes_baixas['seller_id'].nunique()

    # 5. Contar o número total de vendedores únicos
    total_vendedores_unicos = df_itens_pedidos['seller_id'].nunique()

    # 6. Calcular a porcentagem
    porcentagem_vendedores_baixa_avaliacao = (vendedores_unicos_com_nota_baixa / total_vendedores_unicos) * 100

    # 7. Exibir o resultado
    print(f"Número de vendedores com avaliação menor que 2: {vendedores_unicos_com_nota_baixa}")
    print(f"Número total de vendedores: {total_vendedores_unicos}")
    print(f"Porcentagem de vendedores com avaliação menor que 2: {porcentagem_vendedores_baixa_avaliacao:.2f}%")

except NameError:
    print("Erro: Certifique-se de que os DataFrames 'df_avaliacoes' e 'df_itens_pedido' estão carregados.")
except KeyError as e:
    print(f"Erro: A coluna {e} não foi encontrada. Por favor, verifique os nomes das colunas e tente novamente.")
```

```
Número de vendedores com avaliação menor que 2: 1797
Número total de vendedores: 3095
Porcentagem de vendedores com avaliação menor que 2: 58.06%
```

Quais são as 5 categorias com frete mais caro?

```
import pandas as pd

# Carregando os datasets
order_items = pd.read_csv("https://harve.com.br/praticas/olist/olist_order_items_dataset.csv")
products = pd.read_csv("https://harve.com.br/praticas/olist/olist_products_dataset.csv")

# Juntando os dois datasets pelo product_id
df = order_items.merge(products, on="product_id", how="left")

# Agrupando por categoria de produto e calcular o frete médio
frete_por_categoria = (
    df.groupby("product_category_name")["freight_value"]
    .sum()
    .sort_values(ascending=False)
)

# Selecionando as 5 categorias com maior frete médio
top5_categorias = frete_por_categoria.head(5)

# exibindo resultado
print("As 5 categorias com frete mais caro são:\n")
for categoria, frete in top5_categorias.items():
    print(f"{categoria}: R$ {frete:.2f}")
```

```
As 5 categorias com frete mais caro são:

cama_mesa_banho: R$ 204693.04
beleza_saude: R$ 182566.73
moveis_decoracao: R$ 172749.30
esporte_lazer: R$ 168607.51
```

```
informatica_acessorios: R$ 147318.08
```

O que mais influencia no preço do frete, o peso ou o tamanho do produto?

```
import pandas as pd

try:
    # 1. Unir os DataFrames para ter o frete e as dimensões/peso juntos
    df_analise_frete = pd.merge(df_itens_pedidos, df_produtos, on='product_id', how='inner')

    # 2. Criar uma coluna de volume para representar o tamanho do produto
    # Lidando com valores ausentes
    df_analise_frete.dropna(subset=['product_length_cm', 'product_height_cm', 'product_width_cm', 'product_weight_g', 'freigh

    df_analise_frete['volume_cm3'] = df_analise_frete['product_length_cm'] * df_analise_frete['product_height_cm'] * df_anali

    # 3. Calcular a correlação entre o frete e as variáveis de interesse
    correlacao_peso = df_analise_frete['freight_value'].corr(df_analise_frete['product_weight_g'])
    correlacao_volume = df_analise_frete['freight_value'].corr(df_analise_frete['volume_cm3'])

    # 4. Exibir os resultados e a conclusão
    print(f"Correlação entre o custo do frete e o peso: {correlacao_peso:.2f}")
    print(f"Correlação entre o custo do frete e o volume: {correlacao_volume:.2f}")

    if abs(correlacao_peso) > abs(correlacao_volume):
        print("\nO peso do produto tem uma correlação maior com o preço do frete do que o tamanho.")
    else:
        print("\nO tamanho do produto (volume) tem uma correlação maior com o preço do frete do que o peso.")

except NameError:
    print("Erro: Certifique-se de que os DataFrames 'df_itens_pedido' e 'df_produtos' estão carregados.")
except KeyError as e:
    print(f"Erro: A coluna {e} não foi encontrada. Por favor, verifique os nomes das colunas e tente novamente.")
```

```
Correlação entre o custo do frete e o peso: 0.61
Correlação entre o custo do frete e o volume: 0.59

O peso do produto tem uma correlação maior com o preço do frete do que o tamanho.
```

Quais são os dias da semana em que mais são realizados pedidos (orders)? Traga o dia da semana em forma de texto.

```
import pandas as pd

try:
    # Convertendo a coluna de data para o formato datetime
    df_pedidos['order_purchase_timestamp'] = pd.to_datetime(df_pedidos['order_purchase_timestamp'])

    # Extraindo o dia da semana como texto
    df_pedidos['dia_semana'] = df_pedidos['order_purchase_timestamp'].dt.day_name()

    # Contando o número de pedidos por dia da semana
    pedidos_por_dia_semana = df_pedidos['dia_semana'].value_counts()

    # Exibindo o resultado
    print("Número de pedidos por dia da semana:")
    print(pedidos_por_dia_semana.to_string())

except NameError:
    print("Erro: Certifique-se de que o DataFrame 'df_pedidos' está carregado.")
except KeyError as e:
    print(f"Erro: A coluna {e} não foi encontrada. Por favor, verifique os nomes das colunas e tente novamente.")
```

```
Número de pedidos por dia da semana:
dia_semana
Monday       16196
Tuesday      15963
Wednesday    15552
Thursday     14761
Friday       14122
Sunday       11960
Saturday     10887
```

Agente de IA para Auxiliar nas Respostas do Projeto.

```
pip install openai
```

Mostrar saída oculta

```
from openai import OpenAI
import time

# 1. Inicializando o cliente com a chave de API
client = OpenAI(api_key="sk-proj-Lu1tiQkpj1Cm1bb6uZsjfKNlJgRlXAXYaItKlcqVr71SUgp1AR-fPAF2yBudTCstlywGlR4ySwk-rg0A")#sua chave d

# 2. ID do seu assistente
assistant_id = "asst_2p6V02GJAdCt" #seu assistente

# --- Input para o conteúdo da mensagem com as informações para o agente fazer as buscas ---
user_question = input("Por favor, digite a sua pergunta para o assistente do projeto Olist: ")

# 3. Cria um thread (conversa)
thread = client.beta.threads.create()

# 4. Envia a mensagem para o thread usando a variável do usuário
client.beta.threads.messages.create(
    thread_id=thread.id,
    role="user",
    content=user_question
)

# 5. Rodando o assistente nesse thread
run = client.beta.threads.runs.create(
    thread_id=thread.id,
    assistant_id=assistant_id
)

# 6. Aguardando a resposta
while True:
    run_status = client.beta.threads.runs.retrieve(
        thread_id=thread.id,
        run_id=run.id,
    )
    if run_status.status == "completed":
        break
    elif run_status.status == "failed":
        print("A execução falhou.")
        break
    time.sleep(1)

# 7. Recupera as mensagens do thread
messages = client.beta.threads.messages.list(thread_id=thread.id)

# 8. Mostra a última resposta do assistente
for m in messages.data:
    if m.role == "assistant":
        print("Assistente:", m.content[0].text.value)
```

```
Por favor, digite a sua pergunta para o assistente do projeto Olist: Quais são os dias da semana em que mais são realizados pec
/tmp/ipython-input-3419490225.py:14: DeprecationWarning: The Assistants API is deprecated in favor of the Responses API
  thread = client.beta.threads.create()
/tmp/ipython-input-3419490225.py:17: DeprecationWarning: The Assistants API is deprecated in favor of the Responses API
  client.beta.threads.messages.create(
/tmp/ipython-input-3419490225.py:24: DeprecationWarning: The Assistants API is deprecated in favor of the Responses API
  run = client.beta.threads.runs.create(
/tmp/ipython-input-3419490225.py:31: DeprecationWarning: The Assistants API is deprecated in favor of the Responses API
  run_status = client.beta.threads.runs.retrieve(
/tmp/ipython-input-3419490225.py:43: DeprecationWarning: The Assistants API is deprecated in favor of the Responses API
  messages = client.beta.threads.messages.list(thread_id=thread.id)
Assistente: Os dias da semana em que mais são realizados pedidos (orders) são:

- Segunda-feira: 16,196 pedidos
- Terça-feira: 15,963 pedidos
- Quarta-feira: 15,552 pedidos
- Quinta-feira: 14,761 pedidos
- Sexta-feira: 14,122 pedidos
- Domingo: 11,960 pedidos
- Sábado: 10,887 pedidos

Esses são os dias da semana com a quantidade de pedidos realizados, em ordem decrescente. Essas informações foram extraídas do
```