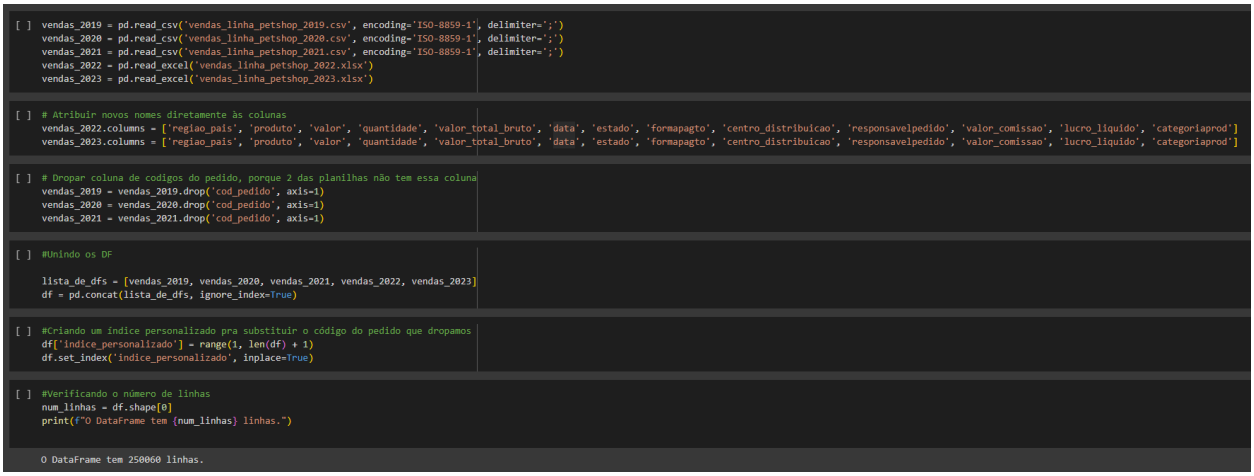
FIAP - Faculdade de Informática e Administração Paulista

- ***Initial information::***

It was necessary to process the dataset so that the columns had the same name and that they were consistent across the dataset. The 5 spreadsheets were merged into a single dataset.



```
[ ] vendas_2019 = pd.read_csv('vendas_linha_petshop_2019.csv', encoding='ISO-8859-1', delimiter=';')
    vendas_2020 = pd.read_csv('vendas_linha_petshop_2020.csv', encoding='ISO-8859-1', delimiter=';')
    vendas_2021 = pd.read_csv('vendas_linha_petshop_2021.csv', encoding='ISO-8859-1', delimiter=';')
    vendas_2022 = pd.read_excel('vendas_linha_petshop_2022.xlsx')
    vendas_2023 = pd.read_excel('vendas_linha_petshop_2023.xlsx')

[ ] # Atribuir novos nomes diretamente às colunas
    vendas_2022.columns = ['regiao_pais', 'produto', 'valor', 'quantidade', 'valor_total_bruto', 'data', 'estado', 'formapagto', 'centro_distribuicao', 'responsavelpedido', 'valor_comissao', 'lucro_liquido', 'categoriaprod']
    vendas_2023.columns = ['regiao_pais', 'produto', 'valor', 'quantidade', 'valor_total_bruto', 'data', 'estado', 'formapagto', 'centro_distribuicao', 'responsavelpedido', 'valor_comissao', 'lucro_liquido', 'categoriaprod']

[ ] # Dropar coluna de codigos do pedido, porque 2 das planilhas não tem essa coluna
    vendas_2019 = vendas_2019.drop('cod_pedido', axis=1)
    vendas_2020 = vendas_2020.drop('cod_pedido', axis=1)
    vendas_2021 = vendas_2021.drop('cod_pedido', axis=1)

[ ] #Unindo os DF

    lista_de_dfs = [vendas_2019, vendas_2020, vendas_2021, vendas_2022, vendas_2023]
    df = pd.concat(lista_de_dfs, ignore_index=True)

[ ] #Criando um índice personalizado pra substituir o código do pedido que dropamos
    df['indice_personalizado'] = range(1, len(df) + 1)
    df.set_index('indice_personalizado', inplace=True)

[ ] #Verificando o número de linhas
    num_linhas = df.shape[0]
    print(f"O DataFrame tem {num_linhas} linhas.")

    O DataFrame tem 250060 linhas.
```

1) *Regarding the provided data, are there missing data? Describe what was found. In situations like this, what needs to be done?*

Answer:

Yes, there are missing data in the columns "quantity", "gross_total_value", "commission_value", and "net_profit". There are 332 missing values in the "quantity" column, 20 in the gross_total_value" column, 21 in the "commission_value" column, and 21 in the "net_profit" column, as analyzed below:

```
[9]  missing_count = df.isnull().sum()
     print(missing_count)
     total_missing_count = missing_count.sum()
     print(f"Total de valores ausentes: {total_missing_count}")
```

```
regiao_pais              0
produto                  0
valor                    0
quantidade             332
valor_total_bruto       20
data                     0
estado                   0
formapagto               0
centro_distribuicao      0
responsavelpedido        0
valor_comissao          21
lucro_liquido           21
categoriaprod            0
dtype: int64
Total de valores ausentes: 394
```

In situations like this, it is necessary to analyze the null value found in detail and decide whether it should be eliminated or filled. In cases where the null data is in the "quantity" or "value" columns but the "gross_total_value" column is filled, it is possible to correct the missing data since gross_total_value = quantity * value. However, this is not always possible because if we had null data in the date column, for example, we would need to drop the rows with missing data.

```
[10] df_tratamento = df.copy()
```

```
[11] df_tratamento['valor'] = df_tratamento['valor'].apply(lambda x: x.replace(',', '.') if ',' in str(x) else x)
     df_tratamento['valor'] = pd.to_numeric(df_tratamento['valor'], errors='coerce')
     df_tratamento['valor_total_bruto'] = df_tratamento['valor_total_bruto'].apply(lambda x: x.replace(',', '.') if ',' in str(x) else x)
     df_tratamento['valor_total_bruto'] = pd.to_numeric(df_tratamento['valor_total_bruto'], errors='coerce')

     #pegando apenas quando temos quantidade null e o resto não null
     df_tratamento['quantidade_tratada'] = df_tratamento.valor_total_bruto / df_tratamento.valor
     df_tratamento['quantidade_tratada']

     #alguns valores ficaram claramente inconsistentes, a ser investigado posteriormente
     df_tratamento['quantidade'] = df_tratamento['quantidade'].fillna(df_tratamento['quantidade_tratada'])

     #validando que não tem mais nenhum nulo
     df_tratamento['quantidade'].isnull().sum()
```

```
0
```

```
[12] #agora vamos avaliar quando valor_total_bruto é NaN. Seria possível também fazer o cálculo se a coluna quantidade estivesse preenchida de maneira correta.
     #Como não está, será necessário dropar essas colunas.
     df_tratamento[df_tratamento['valor_total_bruto'].isna()]
     df_tratamento = df_tratamento.dropna(subset=['valor_total_bruto'])
```

```
[16] #revendo os missing values
     missing_count = df_tratamento.isnull().sum()
     print(missing_count)
     total_missing_count = missing_count.sum()
     print(f"Total de valores ausentes: {total_missing_count}")

     regiao_pais            0
     produto                0
     valor                  0
     quantidade             0
     valor_total_bruto      0
     data                   0
     estado                 0
     formapagto             0
     centro_distribuicao    0
     responsavelpedido      0
     valor_comissao         0
     lucro_liquido          0
     categoriaprod          0
     quantidade_tratada     0
     dtype: int64
     Total de valores ausentes: 0
```

After processing, we dropped the rows that still had null values.

In the last cell, we verified if the replacement had indeed worked. The code returned 0 missing values, proving that everything worked correctly.

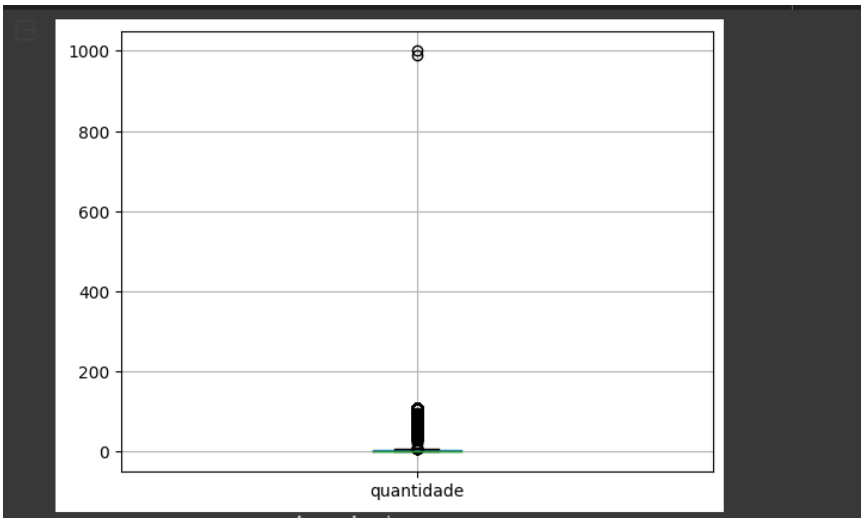
2) *Analyze the data from the perspective of the quantity column. Are there outliers in the provided data? Is it possible to identify anything related to sales regarding these outliers? Justify your answer.*

Resposta:

Yes, there are outliers in the data. Some quantities were negative, indicating data entry errors. It was necessary to preprocess the columns so that we could plot the data, and the outliers can be observed through the boxplot.

Exercício 2

```
[ ] #Transformando a coluna quantidade em numerica, substituindo possíveis pontos e letras antes de usar o coerce
    df['quantidade'] = df['quantidade'].astype(str)
    df['quantidade'] = pd.to_numeric(df['quantidade'].str.replace('[^\d.,]', '', regex=True), errors='coerce')

    #Identificar outliers usando o método IQR
    Q1 = df['quantidade'].quantile(0.25)
    Q3 = df['quantidade'].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = df[(df['quantidade'] < lower_bound) | (df['quantidade'] > upper_bound)]

    #Visualizar os outliers
    df.boxplot(column='quantidade')
    plt.show()

    #Imprimir os outliers
    print(outliers)

                          quantidade valor_total_bruto                 data  \
```



3) *In relation to the consistency of the gross_total_value data, what can be reflected about its contents? Are there inconsistent data? Justify how it is possible to correct them and perform this important activity, leaving this data ready for analysis.*

Answer:

Yes, there are inconsistent data in the gross_total_value column. Some of the spreadsheets used as a basis have a different name for the column in question, which was addressed by renaming these columns so that they are all the same. Other columns were renamed in the same cell, aiming to ensure that the analysis could be performed correctly.

Additionally, the data in the dataframe had variations in the representation of a monetary value: some had the symbol R$ in front, others had the cents separated by commas, others by periods. Ideally, we should have only numbers with cents separated by periods returning in all of them to facilitate possible calculations, as they can only be done with numbers.

There are also cases where the gross_total_value does not match the multiplication between the 'value' and 'quantity' fields. These cases were removed from the dataframe, as they represent incorrect data inputs.

```python
#Verificando o tipo de dado:
tipo_dado_1 = df['valor_total_bruto'].dtype
print(f"Tipo de dado da coluna 'valor_total_bruto': {tipo_dado_1}")

tipo_dado_2 = df['quantidade'].dtype
print(f"Tipo de dado da coluna 'quantidade': {tipo_dado_2}")

tipo_dado_3 = df['valor'].dtype
print(f"Tipo de dado da coluna 'valor': {tipo_dado_3}")

#transformando as colunas do valor bruto para numerica (para nao dar erro)
#combined_df['valor_total_bruto'] = pd.to_numeric(combined_df['valor_total_bruto'], errors='coerce')
```

```
Tipo de dado da coluna 'valor_total_bruto': object
Tipo de dado da coluna 'quantidade': float64
Tipo de dado da coluna 'valor': object
```

```python
#Transformando o valor_total_bruto em float para permitir cálculos
df['valor_total_bruto'] = df['valor_total_bruto'].str.replace(',', '.', regex=True)
df['valor_total_bruto'] = df['valor_total_bruto'].str.replace('[^\d.]', '', regex=True)
df['valor_total_bruto'] = df['valor_total_bruto'].astype(float)

#Transformando o valor em float para permitir cálculos
df['valor'] = df['valor'].str.replace(',', '.', regex=True)
df['valor'] = df['valor'].str.replace('[^\d.]', '', regex=True)
df['valor'] = pd.to_numeric(df['valor'], errors='coerce')

#Lida com valores infinitos substituindo por 0
df['quantidade'] = df['quantidade'].replace([float('inf'), float('-inf')], 0)

#Substitui valores não finitos por 0 antes de converter para int
df['quantidade'] = df['quantidade'].fillna(0).astype(int)

print(df)
```

```python
#Calcular o valor total bruto com base em valor e quantidade (total = valor * quantidade)
valor_total_calculado = df['valor'] * df['quantidade']

#Verificar se os valores na coluna 'valor_total_bruto' correspondem ao valor calculado
inconsistentes = df[valor_total_calculado != df['valor_total_bruto']]

#Limpar os casos com inconsistência
df = df[valor_total_calculado == df['valor_total_bruto']]

print(df)
```

```python
#Verificando o número de linhas
num_linhas = df.shape[0]
print(f"O DataFrame tem {num_linhas} linhas.")
```

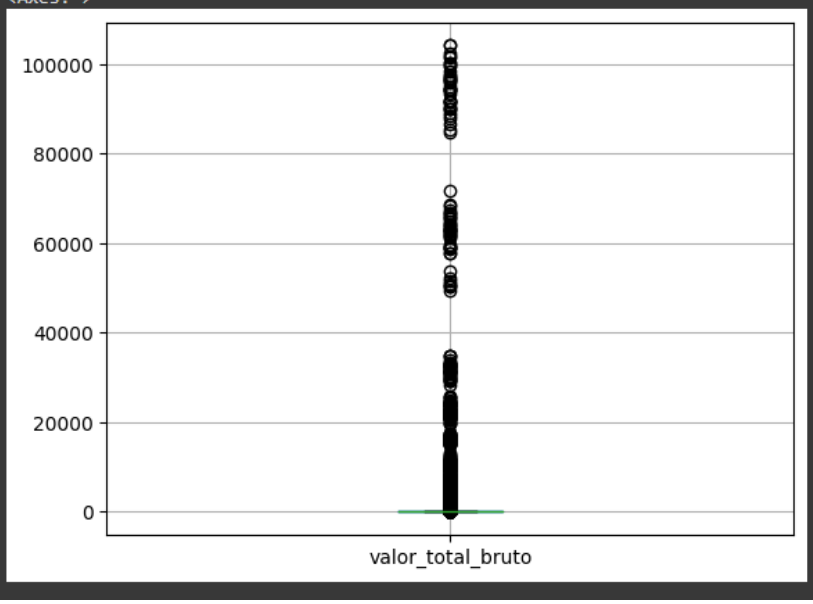
```
O DataFrame tem 92397 linhas.
```

```python
#Analisando valores mínimo, máximo, média e mediana:

min_valor = combined_df['valor_total_bruto'].min()
max_valor = combined_df['valor_total_bruto'].max()
print(f"Valor mínimo: {min_valor}")
print(f"Valor máximo: {max_valor}")

media = combined_df['valor_total_bruto'].mean()
mediana = combined_df['valor_total_bruto'].median()
print(f"Média: {media}")
print(f"Mediana: {mediana}")
```

```
Valor mínimo: 28.0
Valor máximo: 22484.0
Média: 394.27202472952087
Mediana: 84.0
```

```
[ ]  #box plot
     df[['valor_total_bruto']].boxplot()
```

```
<Axes: >
```




```
[ ]  df.head()
```

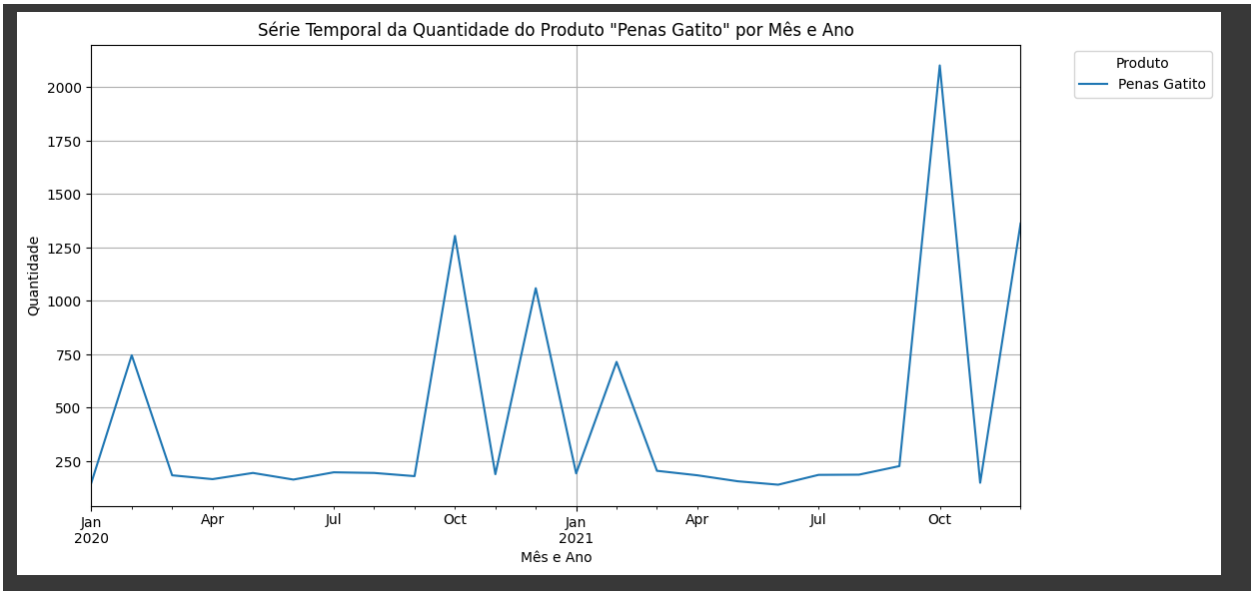
indice_personalizado	regiao_pais	produto	valor	quantidade	valor_total_bruto	data	estado	formapagto	centro_distribuicao	responsavelpedido	valor_comissao	lucro_liquido	categoriaprod
50284	Norte	Biscoito True Champion 300g	24.64	2	49.28	12/03/2020	Acre	Cartão Crédito	Rapid Pink	Carlos	1,76	26,4	Alimentação
50285	Norte	Biscoito True Champion 300g	23.52	2	47.04	10/04/2020	Amapá	Cartão Crédito	Rapid Pink	Maria Linda	1,68	25,2	Alimentação
50286	Norte	Biscoito True Champion 300g	24.64	4	98.56	07/08/2020	Pará	Dinheiro	Rapid Pink	Julia	3,52	52,8	Alimentação
50287	Norte	Biscoito True Champion 300g	21.28	4	85.12	10/01/2020	Rondônia	Pix	Rapid Pink	Yuri	3,04	45,6	Alimentação
50289	Norte	Biscoito True Champion 300g	21.28	4	85.12	04/05/2020	Roraima	Boleto Bancário	Rapid Pink	Adriana	3,04	45,6	Alimentação

```
[ ]  num_linhas = df.shape[0]
     print(f"O DataFrame tem {num_linhas} linhas.")
```

```
O DataFrame tem 92397 linhas.
```

4) *After performing data cleaning and preprocessing, select a product and apply time series analysis techniques to perform the following activities:*

a) Create a chart for the quantity series of the product by month and year.



b) Apply the stationarity test. Is there a need for transformation? Which one?

```
[ ]  #B. Aplicar o teste de estacionariedade.
     result = adfuller(grupo_por_mes_ano_produto)
     print('Estatística ADF:', result[0])
     print('P-Value:', result[1])
     print('Valores críticos:', result[4])

     if result[1] <= 0.05:
         print('A série é estacionária.')
     else:
         print('A série não é estacionária e pode precisar de transformação.')
```

```
Estatística ADF: -2.039245275812784
P-Value: 0.269634810328626
Valores críticos: {'1%': -3.769732625845229, '5%': -3.005425537190083, '10%': -2.6425009917355373}
A série não é estacionária e pode precisar de transformação.
```

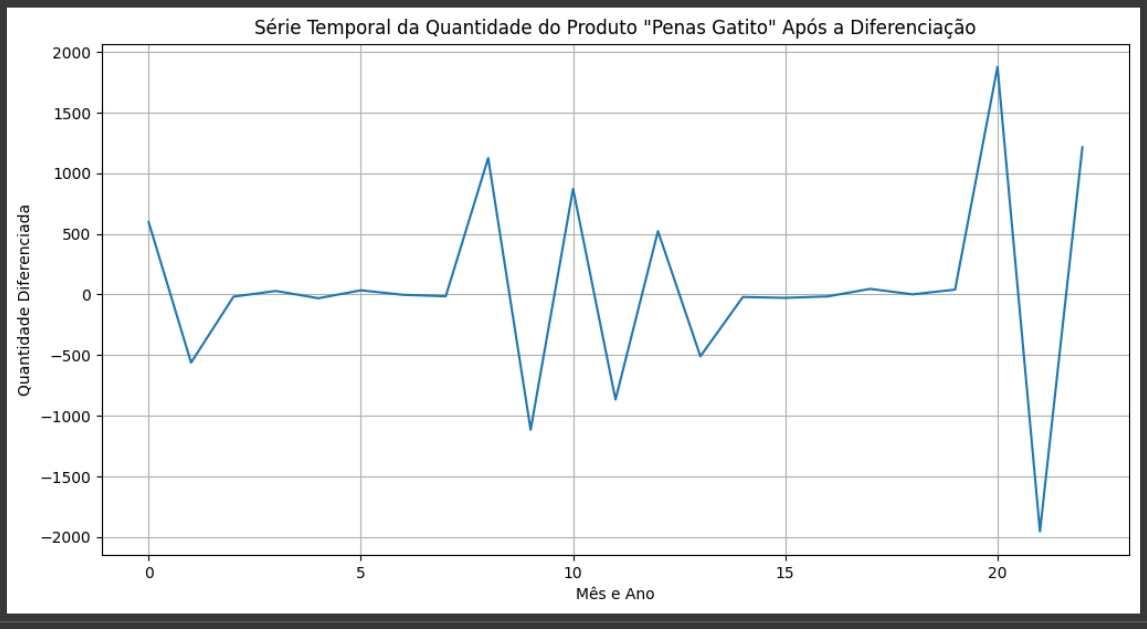
c) Apply the stationarity test to the transformed series?

```
[ ] #C. Aplicar o teste de estacionariedade da série transformada
    #Fazer a transformação
    serie_temporal_diferenciada = grupo_por_mes_ano_produto.diff(periods=1).dropna()
    result_diferenciado = adfuller(serie_temporal_diferenciada)

    #Imprimir resultado da transformação
    print('Estatística ADF após a transformação:', result_diferenciado[0])
    print('P-Value após a transformação:', result_diferenciado[1])
    print('Valores críticos após a transformação:', result_diferenciado[4])

    if result_diferenciado[1] <= 0.05:
        print('A série transformada é estacionária.')
    else:
        print('A série transformada não é estacionária e pode precisar de transformação adicional ou outra abordagem.')
```

```
Estatística ADF após a transformação: -11.900888467629898
Valor-p após a transformação: 5.578806122883552e-22
Valores críticos após a transformação: {'1%': -3.769732625845229, '5%': -3.005425537190083, '10%': -2.642500917355373}
A série transformada é estacionária.
```

d) Create the plot of the transformed series.

```
#D. Gráfico da série transformada
#Converter a série temporal para um formato numérico
serie_temporal_diferenciada = np.array(serie_temporal_diferenciada)

#Criar um gráfico da série transformada diferenciada
plt.figure(figsize=(12, 6))
plt.plot(serie_temporal_diferenciada)
plt.title(f'Série Temporal da Quantidade do Produto "{produto_escolhido}" Após a Diferenciação')
plt.xlabel('Mês e Ano')
plt.ylabel('Quantidade Diferenciada')
plt.grid(True)
plt.show()
```



5) *After applying data cleaning and preprocessing, calculate the mean and median per product category and the ranking of the best-selling products.*

```
[ ] #Após aplicar a limpeza e tratamento nos dados, calcule a média e mediana por categoria de produto e o ranking dos produtos mais vendidos.
    #Agrupar por categoria e calcular a media e mediana
    grupo_por_categoria = df.groupby('categoriaprod')
    media_por_categoria = grupo_por_categoria['quantidade'].mean()
    mediana_por_categoria = grupo_por_categoria['quantidade'].median()

    resumo_por_categoria = pd.DataFrame({'Média': media_por_categoria, 'Mediana': mediana_por_categoria})
    print(resumo_por_categoria)
```

categoriaprod	Média	Mediana
Acessório	3.135114	1.0
Alimentação	4.069991	2.0
Bebedouros e Comedouros	2.697003	1.0
Brinquedo	3.297683	1.0
Higiene e Limpeza	3.111727	1.0
Medicamento	3.554737	2.0
Petisco	3.608583	2.0

```
[ ] #Ranking de produtos mais vendidos
    grupo_por_produto = df.groupby('produto')
    total_vendido_por_produto = grupo_por_produto['quantidade'].sum()
    ranking_produtos = pd.DataFrame({'Total Vendido': total_vendido_por_produto})
    ranking_produtos_ordenado = ranking_produtos.sort_values(by='Total Vendido', ascending=False)

    print(ranking_produtos_ordenado)
```

produto	Total Vendido
Biscoito True Champion 300g	17823
Shampoo vegano para cachorro e gato de camomila...	15742
Ração Úmida Royal Canin Lata Veterinary Cães Ad...	15158
Bandana Disney Mickey Fábrica Petti	15005
Meias esportivas para cães para ambientes inter...	14912
Mordedor de Corda Bola	14552
Ração Royal Canin Exigent Gatos Adultos 1,5Kg	14166
Whiskas Petisco Temptations Anti Bola de Pelo 40g	14057
Vitamina E Granulado BigForce	13493
Antipulgas e Carrapatos MSD Bravecto para Pet d...	13265
Pote Petisco Para Cachorro Bifinho Sabor Carne ...	13219
Ração Royal Canin Club Performance para Cães Ad...	13097
Cama Coração Coroa Pet Nest Almofada Lavável	12521
Suplemento Alimentar Glutamina Mundo Animal Nut...	12337
Roupa para Gato Petti	12017
Bola Pet Vinil Big Blue	11933
Biscoito Pedigree Biscrok Multi para Cães Adult...	11740
Bebedouro e Comedouro Automático para Cães e Ga...	11657
Roupa para Cão Billaboard	11311
Escova Pet Rasqueadeira Tira Pelos Cachorro e G...	11227
Kit Banho e Tosa com Escova PetShop Câopeon	10658
Penas Gatito	10515
Cama média almofadada Pity	10411
Nutri Alimentador Inteligente Automático Câmera...	10391