

Programa de Estágio Quod | Teste de Analytics

Com intuito de demonstrar habilidade técnica e visão acerca dos dados a fim de alcançar a oportunidade oferecida pela Quod. Talk Data.

Neste relatório trouxe:

- Parte 1: Programação em Python
- Parte 2: SQL

Feito por: Arthur silva Capistrano

LinkedIn: www.linkedin.com/in/arthur-capistrano/

Contato: (81) 9.8114-2081

Medium: medium.com/@arthur.capistrano12

Cidade: Olinda - PE

Github: github.com/ArthurCapistrano

```
# Importando bibliotecas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
import random
```

Parte 1.

- Script dataset fictício:

```
# Criando Script para montar um dataset
def criar_dataset_vendas(n_linhas, produtos, categoria_produto, categoria_preco, data_inicio, data_fim, duplicadas= 5, nulos= 10):

    # Para que os valores sejam reproduzíveis
    random.seed(42)

    # gerando uma data aleatória
    def criar_date(inicio, fim):
        inicio = int(inicio.timestamp())
        fim = int(fim.timestamp())
        # gerar um tempo em segundos aleatório
        timestamp = random.randint(inicio, fim)
        # converte para datetime
        data = datetime.fromtimestamp(timestamp)
        return data

    # Criando e populando colunas
    dados = {
        "ID": np.arange(n_linhas),
        "data": [criar_date(data_inicio, data_fim).strftime("%d-%m-%Y") for _ in range(n_linhas)],
        "produto": [random.choice(produtos) for _ in range(n_linhas)],
        "quantidade": [random.randint(1, 50) for _ in range(n_linhas)],
    }

    # Criando coluna de categoria e preco. Se o valor não estiver no dicionário, então ficamos com nan
    dados["categoria"] = [categoria_produto.get(produto, np.nan) for produto in dados["produto"]]

    dados["preco"] = [categoria_preco.get(categoria, np.nan) for categoria in dados["produto"]]

    # Criando o df pandas
    df_vendas = pd.DataFrame(dados)

    # Adicionando valores nulos em posições aleatórias
    for cols in ["data", "produto", "quantidade", "categoria", "preco"]:
        for _ in range(nulos):
            ind_aleatorio = random.randint(0, n_linhas - 1)
            pos_col = df_vendas.columns.get_loc(cols) # posição numérica da coluna
            df_vendas.iloc[ind_aleatorio, pos_col] = np.nan

    # Escolhendo linhas aleatórias para duplicar
    linhas_duplicadas = df_vendas.sample(n= duplicadas)
    # Concatendo com o dataframe
    df_vendas = pd.concat([df_vendas, linhas_duplicadas], ignore_index= True)
```

```
return df_vendas
```

Criando o dataset:

```
# Criando o dataset

# Lista de produtos para vendas
produtos = ["Banana", "Maçã", "Pera", "Abacaxi", "Laranja", "Manga", "Uva"]

# Dicionário para categorizar produtos
categorias = {
    "Banana": "Frutas Tropicais",
    "Maçã": "Frutas de Clima Frio",
    "Pera": "Frutas de Clima Frio",
    "Abacaxi": "Frutas Tropicais",
    "Laranja": "Frutas Cítricas",
    "Manga": "Frutas Tropicais",
    "Uva": "Frutas Cítricas",
}

# Dicionário para preços a part
precos = {
    "Banana": 3.0,
    "Maçã": 2.0,
    "Pera": 3.5,
    "Abacaxi": 7.0,
    "Laranja": 5.0,
    "Manga": 6.0,
    "Uva": 4.0,
}

# Intervalo de datas
data_inicio = datetime(2023, 1, 1)
data_fim = datetime(2023, 12, 31)
```

- 1º Vista dos Dados

```
# Criando o df
df_vendas = criar_dataset_vendas(100, produtos, categorias, precos, data_inicio, data_fim, 5, 10)

# Colocando o ID como index
df_vendas.set_index("ID", inplace= True)

# 1º vista dos dados:
df_vendas.head(20)
```

ID	data	produto	quantidade	categoria
0	06-09-2023	Manga	47	Frutas Tropicais
1	13-02-2023	Pera	47	Frutas de Clima Frio
2	10-01-2023	Manga	17	Frutas Tropicais
3	15-10-2023	Manga	33	Frutas Tropicais
4	17-04-2023	Manga	49	Frutas Tropicais
5	06-04-2023	Banana	12	Frutas Tropicais
6	28-03-2023	null	33	null
7	24-02-2023	null	7	Frutas Tropicais
8	14-10-2023	Maçã	41	Frutas de Clima Frio
9	09-02-2023	Laranja	20	null
10	20-09-2023	Manga	41	Frutas Tropicais
11	15-10-2023	Maçã	33	Frutas de Clima Frio
12	null	null	39	null
13	31-07-2023	Abacaxi	null	Frutas Tropicais
14	03-02-2023	Abacaxi	10	Frutas Tropicais

20 rows ↓

```
# Olhando info extras
df_vendas.info()
```

```

<class 'pandas.core.frame.DataFrame'>
Index: 105 entries, 0 to 71
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   data        96 non-null    object  
 1   produto     96 non-null    object  
 2   quantidade  95 non-null    float64 
 3   categoria   94 non-null    object  
 4   preco       94 non-null    float64 
dtypes: float64(2), object(3)
memory usage: 4.9+ KB

```

Como forma saudável e boa prática, os **valores faltantes** não ser lidos, mas não apenas isso, também vamos converter os dados para os tipos corretos e até pensando em uma manutenção do uso de memória. É algo que para datasets pequenos como esse não é prejudicial, mas para maiores e mais pesados, pode ser uma mudança importante no tempo gasto de espera de querys, etc.

```
# Olhando variáveis numéricas
df_vendas.describe()
```

	quantidade	preco
count		95
mean		25.6105263158
std		15.1441967938
min		1
25%		12.5
50%		28
75%		37.5
max		50

8 rows ↓

Nesta primeira visualização, podemos olhar pequenos detalhes como:

- **Quantidade:**
 - > A maior quantidade que apareceu em um pedido foi de 50 unidades
 - > A média de quantidade comprada foi de 25.61, entretanto este é um valor que pode estar sendo distorcido por outliers, algo a se visualizar na Exploração.
 - > 95 valores encontrados, o que bate com nosso `info` acima
- **Preço:**
 - > O produto mais caro custa 7 reais, o mais barato 2 reais.
 - > O preço médio no dataset está em 4.5.
 - > 94 valores encontrados, o que também confere com o `info` acima.

```
# Olhando o value counts de produtos
df_vendas["produto"].value_counts()
```

produto	count
Manga	
Abacaxi	
Laranja	
Maçã	
Pera	
Uva	
Banana	

7 rows ↓

Podemos reconhecer a manga como produto com maior aparições de venda (isto é diferente de quantidade de vendas).

```
# Olhando o value counts de produtos
df_vendas["categoria"].value_counts()
```

categoria	count
Frutas Tropicais	
Frutas de Clima Frio	
Frutas Cítricas	

3 rows ↓

Tais valores conferem com o que estamos vendendo em `df_vendas["produto"].value_counts()`, uma vez Manga e Abacaxi, são os mais vendidos, sendo ambos "Frutas Tropicais"

- Tratamento

Conhecendo valores faltantes:

```
# Conferindo quantidade de valores faltantes
val_faltantes = df_vendas.isna().sum()
print(f"Temos cerca de: {val_faltantes.sum()} valores faltantes\n")
print(f"Estes estão distribuídos em:\n{val_faltantes}")

# Mudando nome da serie e printando
val_faltantes = val_faltantes.reset_index()

# Renomeando colunas
val_faltantes.columns = ["Colunas", "Qtd Valores nulos"]

Temos cerca de: 50 valores faltantes

Estes estão distribuídos em:
data         9
produto      9
quantidade   10
categoria    11
preco        11
dtype: int64
```

Visualização de valores faltantes:

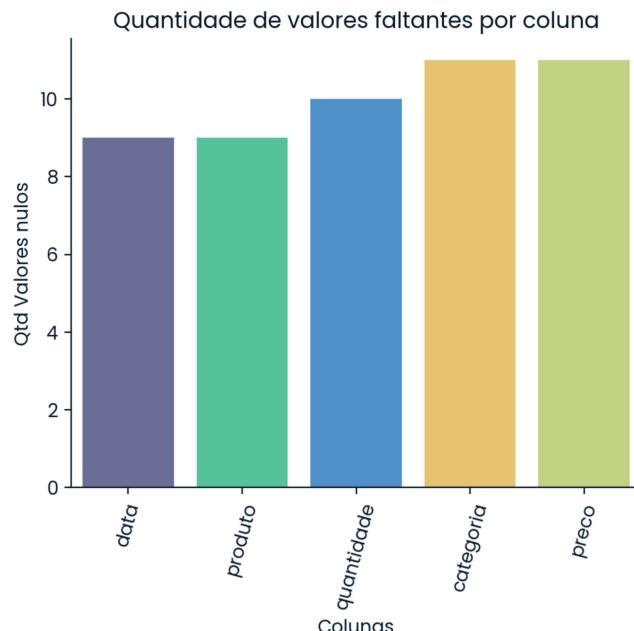
```
# Visualizando valores faltantes

# Criando o gráfico
sns.catplot(
    data= val_faltantes,
    kind= "bar",
    x= "Colunas",
    y= "Qtd Valores nulos",
    hue= "Colunas"
)

# Adicionando título
plt.title("Quantidade de valores faltantes por coluna")

# Rotacionando x para nomes não se sobreponham
plt.xticks(rotation= 75)

# Ajustando e plotando gráfico
plt.tight_layout()
plt.show()
```



Tratando Valores Faltantes

1 - Caso haja 4 valores nulos na linha, então vamos remover:

```
# Quantidade de nulos antes:  
print(f"Antes da remoção de nulas tínhamos:{df_vendas.isna().sum().sum()} nulos\n")  
  
# Removendo linha se tiver 4 valores nulos  
df_vendas.dropna(thresh= 1, inplace= True)  
  
# Quantidade de nulos após:  
print(f"Após da remoção de linhas com 4 valores nulos temos:{df_vendas.isna().sum().sum()} nulos")  
  
Antes da remoção de nulas tínhamos:50 nulos  
  
Após da remoção de linhas com 4 valores nulos temos:50 nulos
```

- Isso mostra que em nossas linhas que continham valores faltantes, nenhuma linha única tinha 4 valores faltantes

```
# Olhando o dataframe que contém algum valor nulo  
df_vendas[df_vendas.isna().any(axis=1)]
```

ID	data	produto	quantidade	categoria
1	13-02-2023	Pera		47 Frutas de Clima Frio
6	28-03-2023	null		33 null
7	24-02-2023	null		7 Frutas Tropicais
9	09-02-2023	Laranja		20 null
11	15-10-2023	Maçã		33 Frutas de Clima Frio
12	null	null		39 null
13	31-07-2023	Abacaxi		null Frutas Tropicais
17	13-01-2023	Manga		null Frutas Tropicais
21	01-04-2023	Pera		1 Frutas de Clima Frio
23	22-08-2023	Uva		null Frutas Cítricas
24	null	Uva		null Frutas Cítricas
27	06-10-2023	Uva		24 Frutas Cítricas
30	31-07-2023	Pera		4 Frutas de Clima Frio
31	12-06-2023	Abacaxi		null null
35	19-04-2023	Laranja		47 null

40 rows ↓

2 - Valores faltantes em Data:

A abordagem que escolhi para preencher valores faltantes na coluna de "data" foi o `ffill`. Tomei essa decisão uma vez que estamos lidando com dados de vendas onde, supostamente, deveríamos ter uma constância de entradas, visto que é possível várias vendas em uma mesma data.

Por conta disso, irei realizar a transformação para o tipo datetime (como visto em `data.info`, este campo está como tipo object pandas) e então ordenação do nosso data frame de vendas por datetime e, por fim, o preenchimento dos valores faltantes

```
# Mostrando o tipo que está a coluna de data  
print(f"O tipo da nossa coluna 'data' está como: {df_vendas['data'].dtype}\n")  
  
# Convertendo data para tipo datetime  
df_vendas["data"] = pd.to_datetime(df_vendas["data"], format= "%d-%m-%Y")  
print(f"Após a conversão o tipo da nossa coluna 'data' está como: {df_vendas['data'].dtype}\n")  
  
# Ordenando o df através das datas  
df_vendas = df_vendas.sort_values("data")  
  
O tipo da nossa coluna 'data' está como: object  
  
Após a conversão o tipo da nossa coluna 'data' está como: datetime64[ns]
```

```
# Quantidade de valores faltantes antes de preenchermos  
print(f"Na coluna data tínhamos: {df_vendas['data'].isna().sum().sum()} valores faltantes\n")  
  
# Preenchendo os valores faltantes em Data com ffill  
df_vendas["data"] = df_vendas["data"].ffill()  
  
# Conferindo se ainda temos valores nulos na coluna data
```

```
print(f"Após o preenchimento temos {df_vendas['data'].isna().sum().sum()} valores faltantes")
```

Na coluna data tinhemos: 9 valores faltantes

Após o preenchimento temos 0 valores faltantes

```
# Conferindo como ficou o df  
df_vendas.head(10)
```

ID	data	produto	quantidade	categoria
38	2023-01-03T00:00:00.000	Maçã		49 Frutas de Clima Frio
2	2023-01-10T00:00:00.000	Manga		17 Frutas Tropicais
18	2023-01-12T00:00:00.000	Laranja		35 Frutas Cítricas
17	2023-01-13T00:00:00.000	Manga		null Frutas Tropicais
80	2023-01-18T00:00:00.000	Pera		41 Frutas de Clima Frio
79	2023-01-28T00:00:00.000	Laranja		46 Frutas Cítricas
67	2023-01-31T00:00:00.000	Maçã		16 Frutas de Clima Frio
85	2023-01-31T00:00:00.000	Manga		22 Frutas Tropicais
14	2023-02-03T00:00:00.000	Abacaxi		10 Frutas Tropicais
51	2023-02-06T00:00:00.000	null		14 Frutas de Clima Frio

10 rows ↓

3 - Valores faltantes em produto, categoria e preço:

Como estamos lidando com dados de vendas e é clara a relação entre as colunas (por exemplo, um produto tem uma categoria e um preço), o preenchimento desses dados faltantes é ideal que possamos identificar essas relações.

Em outras situações, poderíamos utilizar a *imputação de moda, mediana* ou até mesmo o uso de um *SimpleImputer do sklearn* como forma de trazer valores mais próximos da realidade.

Porém, no nosso caso, o entendimento da relação entre as colunas será o suficiente para trazer uma boa imputação dos valores faltantes

```
# Olhando a relação entre as 3 colunas  
df_vendas.groupby(["produto", "categoria", "preco"]).size()
```

produto	categoria	preco
Abacaxi	Frutas Tropicais	
Banana	Frutas Tropicais	
Laranja	Frutas Cítricas	
Manga	Frutas Tropicais	
Maçã	Frutas de Clima Frio	
Pera	Frutas de Clima Frio	
Uva	Frutas Cítricas	

Agrupei dessa forma, pois consigo visualizar se há algum padrão que está fugindo. Como não foi encontrado, sabemos que, por exemplo, um abacaxi está na categoria Frutas Tropicais e possui o preço único de 7 reais.

```
# Fazendo dicionário com as relações:
```

```
# Relação produto com categoria  
rel_prod_categ = df_vendas.groupby("produto")["categoria"].apply(lambda x:x.iloc[0]).to_dict() # Faço o groupby como acima, seleciono a linha, pego o 1º valor associado a coluna que usamos para agrupar e então transforma em dicionario
```

```
# Relação produto com o seu preço  
rel_prod_preco = df_vendas.groupby("produto")["preco"].apply(lambda x:x.iloc[0]).to_dict()
```

```
# Relação do preço com o produto  
rel_preco_prod = df_vendas.groupby("preco")["produto"].apply(lambda x: x.iloc[0]).to_dict()
```

```
# Olhando os dicionários:
```

```
print("Dicionário Produto -> Categoria:\n")  
display(rel_prod_categ)  
print("\nDicionário Produto -> Preço:\n")  
display(rel_prod_preco)  
print("\nDicionário Preço -> Produto:\n")  
display(rel_preco_prod)
```

Dicionário Produto -> Categoria:

```
{'Abacaxi': 'Frutas Tropicais',
'Banana': 'Frutas Tropicais',
'Laranja': 'Frutas Cítricas',
'Manga': 'Frutas Tropicais',
'Maçã': 'Frutas de Clima Frio',
'Pera': 'Frutas de Clima Frio',
'Uva': 'Frutas Cítricas'}
```

Dicionário Produto -> Preço:

```
{'Abacaxi': 7.0,
'Banana': 3.0,
'Laranja': 5.0,
'Manga': 6.0,
'Maçã': 2.0,
'Pera': 3.5,
'Uva': 4.0}
```

Dicionário Preço -> Produto:

```
{2.0: 'Maçã',
3.0: 'Banana',
3.5: 'Pera',
4.0: 'Uva',
5.0: 'Laranja',
6.0: 'Manga',
7.0: 'Abacaxi'}
```

Preenchendo os valores faltantes:

```
# Quantos valores faltantes ainda tínhamos:
print(f"Ainda tínhamos {df_vendas.isna().sum().sum()}...\n")

# Faltantes na col preço
df_vendas["preco"] = df_vendas["preco"].fillna(df_vendas["produto"].map(rel_prod_preco))

# Faltantes na col produto
df_vendas["produto"] = df_vendas["produto"].fillna(df_vendas["preco"].map(rel_preco_prod))

# Faltantes em categoria
df_vendas["categoria"] = df_vendas["categoria"].fillna(df_vendas["produto"].map(rel_prod_categ))

# Quantos valores faltantes temos após a conversão
print(f"Após a imputação inicial temos {df_vendas[['produto', 'categoria', 'preco']].isna().sum().sum()} valores faltantes")

Ainda tínhamos 41...

Após a imputação inicial temos 2 valores faltantes
```

```
# Onde estão os valores faltantes
df_vendas[['produto', 'categoria', 'preco']].isna().sum()
```

produto	categoria	preco

Sabendo o produto, conseguimos conhecer o preço e sua categoria. Mas, antes disso, vamos buscar essa linha em que ainda há um produto nulo.

```
# Conferindo qual é a linha com produto faltante
df_vendas[df_vendas["produto"].isna()]
```

ID	data	produto	quantidade	categoria
51	2023-02-06T00:00:00.000	null		14 Frutas de

Como está é uma única linha do nosso dataframe, vou realizar seu drop.

Caso esse volume representasse uma grande quantidade de linhas que estivessem com suas colunas "produto" e "preco" ausentes, poderia ser interessante usar imputação da moda/mediana desses respectivos valores, ou, uma outra opção usar modelo de predição para preencher estes valores, apesar de que teríamos uma

limitação pela quantidade de dados.

```
# Removendo esta linha única
df_vendas = df_vendas.dropna(subset= "produto")

# Olhando faltantes em produtos:
print(f"Feito isso temos {df_vendas['produto'].isna().sum()} valores faltantes na coluna produto\n")

# Olhando dnv a quantidade de nulos em produto, categoria e preco
print(f"Após esse drop, temos [{df_vendas[['produto', 'categoria', 'preco']].isna().sum().sum()}] valores faltantes nas colunas Produto, Categoria e Preco")

Feito isso temos 0 valores faltantes na coluna produto

Após esse drop, temos [0] valores faltantes nas colunas Produto, Categoria e Preco
```

```
# Conferindo a quantida de valores faltantes no df
print("Temos então tais valores faltantes:\n")
display(df_vendas.isna().sum())
```

Temos então tais valores faltantes:

	0
data	
produto	
quantidade	
categoria	
preco	

Podemos ver que resta apenas 10 valores faltantes, isto na coluna "quantidade" apenas.

4 - Valores faltantes na coluna quantidade:

Como quantidade não é um valor o qual podemos nos referenciar em outra coluna (ao menos não nesse dataset), então vou imputar a quatidade apartir da mediana de quantidade do produto da linha

```
# Pegando a mediana da quantidade por produto
mediana_qtd_prod = df_vendas.groupby("produto")["quantidade"].apply(lambda x: x.median()).to_dict()

# Conferindo o dicionário
display(mediana_qtd_prod)

{'Abacaxi': 34.0,
'Banana': 10.5,
'Laranja': 36.0,
'Manga': 33.0,
'Maçã': 26.0,
'Pera': 36.0,
'Uva': 23.0}
```

```
# Preenchendo os valores nulos de quantidade usando o dicionário acima
df_vendas["quantidade"] = df_vendas["quantidade"].fillna(df_vendas["produto"].map(mediana_qtd_prod))
```

```
# Conferindo o total de valores faltantes no df
print("Temos então valores faltantes:\n")
display(df_vendas.isna().sum())
```

Temos então valores faltantes:

	0
data	
produto	
quantidade	
categoria	
preco	

1 - Conferindo linhas duplicadas:

```
# Conferir as linhas duplicadas pelo ID  
df_vendas[df_vendas.index.duplicated()]
```

ID	data	produto	quantidade	cc
96	2023-03-05T00:00:00.000	Abacaxi		37 Fr
13	2023-07-31T00:00:00.000	Abacaxi		34 Fr
71	2023-09-02T00:00:00.000	Manga		2 Fr
70	2023-11-19T00:00:00.000	Uva		22 Fr
60	2023-12-25T00:00:00.000	Laranja		43 Fr

5 rows ↴

2 - Removendo linhas duplicadas:

Como o ID deve ser único, então vamos nos guiar também utilizando o ID como uma coluna para ter certeza que a linha inteira está duplicada.

```
# Remover as linhas duplicadas pelo ID
```

```
# Transformando o id em uma coluna  
df_vendas = df_vendas.reset_index()
```

```
# Removendo pelo id  
df_vendas = df_vendas.drop_duplicates()
```

```
# Coloca id novamente como indice  
df_vendas = df_vendas.set_index("ID")
```

```
# Conferindo df de linhas duplicadas  
print("Agora de linhas duplicadas temos:\n")  
display(df_vendas[df_vendas.index.duplicated()])
```

Agora de linhas duplicadas temos:

Your query ran successfully but returned no results.

Conversão de tipo de dados

Como pudemos ver no `info` temos dados que estão como float 64 e object. Fora a data que não estava em datetime, mas que já solucionamos juntos com seus valores faltantes.

Vamos então olhar para as variáveis object e float64.

A ideia é transformar o object pandas as variáveis do tipo "category" que ocupam menos espaço na memória e nos auxiliará na manipulação, caso necessário. Float64 apenas mudar para guardarmos memória, uma vez que nossos dados não estão exigindo.

1 - Vendos os tipos das colunas:

```
# Conferindo os tipos das colunas  
df_vendas.dtypes
```

```
data      datetime64[ns]  
produto    object  
quantidade   float64  
categoria     object  
preco       float64  
dtype: object
```

2 - Convertendo:

- object → category
- float64 → float16

```
# Fazendo as conversões
```

```
df_vendas = df_vendas.apply(lambda col: col.astype("category") if col.dtype == "object" else  
                           col.astype("float32") if col.dtype == float  
                           else col)
```

```
# Conferindo a conversão
```

```
df_vendas.dtypes
```

```
data      datetime64[ns]
produto    category
quantidade float32
categoria   category
preco      float32
dtype: object
```

Salvando dataset Tratado

Guardando o dataset limpo em data_clean.csv

```
# Salvando o df já limpo em data_clean.csv
df_vendas.to_csv("data_clean.csv")
```

Cálculo total de vendas:

Nada mais é que a quantidade total de vendas (Quantidade * Preço) por produto

1 - Realizando o Agrupamento:

```
# Agrupando por produto conheno o total vendido por produto e o seu preço
df_vendas_group = df_vendas.groupby("produto")[["quantidade", "preco"]].agg({"quantidade": "sum", "preco": "mean"})
```

```
# Vendo como fica o agrupamento:
display(df_vendas_group)
```

produto	quantidade
Abacaxi	4
Banana	1
Laranja	4
Manga	5
Maçã	3
Pera	3
Uva	2

2 - Criando a tabela de Total Vendido por Produto:

```
# Criando o df com o total vendido por produto
total = df_vendas_group["quantidade"] * df_vendas_group["preco"]
total = pd.DataFrame(total, columns= ["Total Vendido p Produto"])
```

```
total = total.sort_values(by= "Total Vendido p Produto", ascending= False)
# Vendo tabela com total vendido por produto:
display(total)
```

produto	Total Vendido p Produto
Manga	
Abacaxi	
Laranja	
Pera	
Uva	
Maçã	
Banana	

Identificando produto com maior número de vendas:

```
# Identificando o produto que tem o maior nûm de vendas:
print(f"O produto com maior nûmero de vendas é '{total.index[0]}' com cerca de: {total['Total Vendido p Produto'][0]} reais vendidos")
```

```
O produto com maior nûmero de vendas é 'Manga' com cerca de: 3372.0 reais vendidos
```

Parte 1.2.

Realização da Análise Exploratória - Conhecendo os dados ao longo do tempo:

Para conseguir fazer de forma mais clara a visualização ao longo do tempo, escolhi criar novas colunas como "meses" (nome dos meses de forma categoria), "valor_venda" (será a quantidade gasta em uma venda) e por fim, o "total_gasto_produto" (será um groupby produto e valor_venda que irá trazer o valor gasto por produto de acordo com os meses)

- A primeira ideia que trarei será criar uma coluna com os valores dos meses utilizando o `pd.cut` como uma coluna de dados categóricos.

```
# Criando os labels dos nomes dos meses que vão ser "mapeados" com o cut
labels= ["Janeiro", "Fevereiro", "Março", "Abril", "Maio", "Junho", "Julho", "Agosto", "Setembro", "Outubro", "Novembro", "Dezembro"]
```

```
# Quantidade de bins, para o cut, sempre 1 a mais que as labels
bins = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
```

```
# Criando os bins para compensar com os meses
```

```
df_vendas["meses"] = pd.cut(
    df_vendas["data"].dt.month, #Pegando o valor numéricos dos meses de cada data
    bins=bins,
    labels=labels,
    right=False
)
```

```
# Conferindo o df com a nova coluna
```

```
df_vendas.head()
```

ID	data	produto	quantidade	categoria
38	2023-01-03T00:00:00.000	Maçã		49 Frutas de Clima Frio
2	2023-01-10T00:00:00.000	Manga		17 Frutas Tropicais
18	2023-01-12T00:00:00.000	Laranja		35 Frutas Cítricas
17	2023-01-13T00:00:00.000	Manga		33 Frutas Tropicais
80	2023-01-18T00:00:00.000	Pera		41 Frutas de Clima Frio

5 rows ↓

- Criando coluna com o valor pago de cada venda: Utilizando como uma forma de auxiliar nossa visualização dos dados.

```
# Criando o valor total de uma venda
```

```
df_vendas["valor_venda"] = df_vendas["quantidade"] * df_vendas["preco"]
```

```
# Conferindo esta coluna
display(df_vendas)
```

ID	data	produto	quantidade	categoria	prec
38	2023-01-03T00:00:00.000	Maçã		49 Frutas de Clima Frio	
2	2023-01-10T00:00:00.000	Manga		17 Frutas Tropicais	
18	2023-01-12T00:00:00.000	Laranja		35 Frutas Cítricas	
17	2023-01-13T00:00:00.000	Manga		33 Frutas Tropicais	
80	2023-01-18T00:00:00.000	Pera		41 Frutas de Clima Frio	
79	2023-01-28T00:00:00.000	Laranja		46 Frutas Cítricas	
67	2023-01-31T00:00:00.000	Maçã		16 Frutas de Clima Frio	
85	2023-01-31T00:00:00.000	Manga		22 Frutas Tropicais	
14	2023-02-03T00:00:00.000	Abacaxi		10 Frutas Tropicais	
19	2023-02-06T00:00:00.000	Maçã		50 Frutas de Clima Frio	
53	2023-02-07T00:00:00.000	Laranja		49 Frutas Cítricas	
89	2023-02-09T00:00:00.000	Uva		18 Frutas Cítricas	
50	2023-02-09T00:00:00.000	Laranja		28 Frutas Cítricas	
9	2023-02-09T00:00:00.000	Laranja		20 Frutas Cítricas	
1	2023-02-13T00:00:00.000	Pera		47 Frutas de Clima Frio	

99 rows ↓

- `total_gasto_produto`: Fazendo a quantidade gasta, por produto, de acordo com os meses. Com isso, utilizar para o gráfico de linhas
- `df_total_quantidade_group`: Fazendo a quantidade total de produtos vendidos de acordo com os meses.

```
# Salvando o índice, após esse processo perderia o índice
df_vendas["ID"] = df_vendas.index
```

```
# Fazendo o groupby. Agrupando por produto e meses, então usar a soma do valor venda
df_total_gasto_group = df_vendas.groupby(["produto", "meses"])["valor_venda"].sum().reset_index()
```

```
# Alterando a coluna
```

```
df_total_gasto_group = df_total_gasto_group.rename(columns={"valor_venda": "total_gasto"})
```

```
# Olhando o groupby
```

```
display(df_total_gasto_group)
```

	produto	meses	total_gasto
0	Abacaxi	Janeiro	0
1	Abacaxi	Fevereiro	0
2	Abacaxi	Março	0
3	Abacaxi	Abril	0
4	Abacaxi	Maio	0
5	Abacaxi	Junho	0
6	Abacaxi	Julho	0
7	Abacaxi	Agosto	0
8	Abacaxi	Setembro	0
9	Abacaxi	Outubro	0
10	Abacaxi	Novembro	0
11	Abacaxi	Dezembro	0
12	Banana	Janeiro	0
13	Banana	Fevereiro	0
14	Banana	Março	0

84 rows ↓

O primeiro ponto que é possível notar está em valores 0. Estes valores 0 representam que não houve vendas de determinado produto no mês em específico. Isto ocasionado pela aleatoriedade gerada dos dados e pela pequena quantidade de dados.

Isto, obviamente, vai causar estranheza nos gráficos de linhas. Contudo, é válido falar que não é algo impossível de ser verídico em comércios menores.

```
# Juntar com nosso df_vendas
```

```
df_vendas = df_vendas.merge(df_total_gasto_group, on= ["produto", "meses"], how= "left")
```

```
# Estabelecendo o índice com ID novamente
```

```
df_vendas = df_vendas.set_index(["ID"])
```

```
# Conferindo o df_vendas
```

```
df_vendas.head(20)
```

ID	data	produto	quantidade	categoria	preco	meses
38	2023-01-03T00:00:00.000	Maçã	49	Frutas de Clima Frio	2	Janeiro
2	2023-01-10T00:00:00.000	Manga	17	Frutas Tropicais	6	Janeiro
18	2023-01-12T00:00:00.000	Laranja	35	Frutas Cítricas	5	Janeiro
17	2023-01-13T00:00:00.000	Manga	33	Frutas Tropicais	6	Janeiro
80	2023-01-18T00:00:00.000	Pera	41	Frutas de Clima Frio	3.5	Janeiro
79	2023-01-28T00:00:00.000	Laranja	46	Frutas Cítricas	5	Janeiro
67	2023-01-31T00:00:00.000	Maçã	16	Frutas de Clima Frio	2	Janeiro
85	2023-01-31T00:00:00.000	Manga	22	Frutas Tropicais	6	Janeiro
14	2023-02-03T00:00:00.000	Abacaxi	10	Frutas Tropicais	7	Fevereiro
19	2023-02-06T00:00:00.000	Maçã	50	Frutas de Clima Frio	2	Fevereiro
53	2023-02-07T00:00:00.000	Laranja	49	Frutas Cítricas	5	Fevereiro
89	2023-02-09T00:00:00.000	Uva	18	Frutas Cítricas	4	Fevereiro
50	2023-02-09T00:00:00.000	Laranja	28	Frutas Cítricas	5	Fevereiro
9	2023-02-09T00:00:00.000	Laranja	20	Frutas Cítricas	5	Fevereiro
1	2023-02-13T00:00:00.000	Pera	47	Frutas de Clima Frio	3.5	Fevereiro

20 rows ↓

Valores repetidos são esperados, assim como valores que acabam coincidindo em "valor_venda" e "total_gasto", significando uma única venda no mês.

1 - Gráfico de linhas df_vendas: Quantidade por Produto

```
# Criando dicionário para usar em pallet
```

```
cores_produto = {
```

```
    "Uva": "purple",
    "Laranja": "orange",
    "Manga": "yellow",
    "Maçã": "green",
    "Abacaxi": "gold",
```

```

"Pera": "lightgreen",
"Banana": "yellowgreen",
}

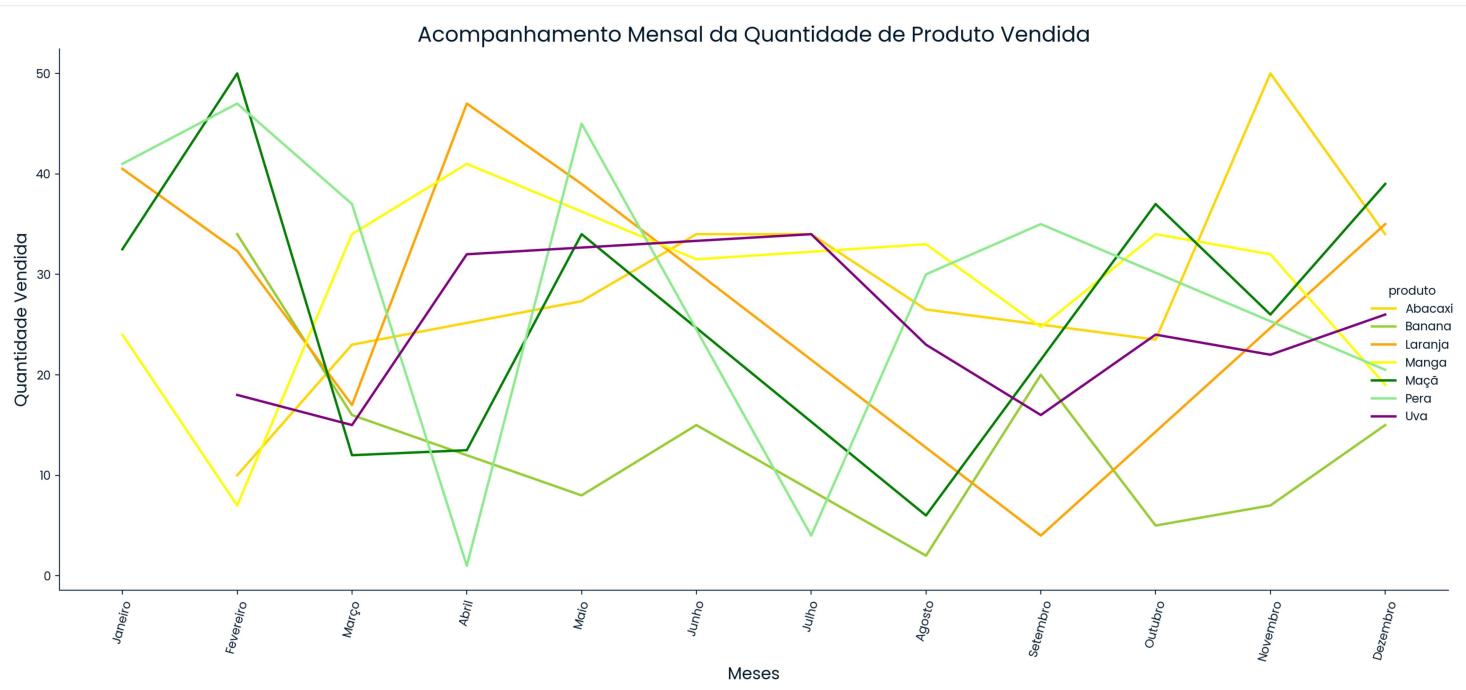
# Criando o gráfico de linha
sns.relplot(
    data=df_vendas,
    x="meses",
    y="quantidade",
    kind="line",
    hue="produto",
    height=8,
    aspect=2,
    palette=cores_produto,
    linewidth=2,
    ci= None
)

# Nomeando nossa figura
plt.title("Acompanhamento Mensal da Quantidade de Produto Vendida", fontsize=18)
plt.xlabel("Meses", fontsize= 14)
plt.ylabel("Quantidade Vendida", fontsize= 14)

# rotação da letra para x da figura
plt.xticks(rotation=75)

# Plotando o gráfico
plt.tight_layout()
plt.show()

```



Com este gráfico podemos notar, em certo grau, a imprevisibilidade dos dados, não havendo uma tendência clara como o aumento contínuo da quantidade de produtos que tem sido vendidos. Isto é comum em casos onde:

- Conjunto de dados pequenos.
- Abordando um conjunto de dados em um curto prazo. A exemplo disso, são operações da bolsa de valores. Períodos curtos podem esconder tendências.
- Dados que não apresentam padrões relacionados ao tempo.

Vou olhar separadamente cada um dos produtos e notar a **quantidade de vezes vendidas** e o **valor total gasto**.

Olhando afundo - Gráficos de linhas para cada produto:

(Levando em consideração que é um df criado, vou criar minhas devidas suposições e insights)

Abacaxi:

- Quantidade Total vendido ao longo dos meses:

```

# Filtrando o df para apenas abacaxi
df_abacaxi = df_vendas[df_vendas['produto'] == "Abacaxi"]

# Criando um df agrupado por meses e com a quantidade total vendida por mes
df_abacaxi_group = df_abacaxi.groupby("meses", as_index=False)[["quantidade"]].sum()

```

```

# Renomeando a coluna do groupby
df_abacaxi_group = df_abacaxi_group.rename(columns= {"quantidade": "total_quantidade"})

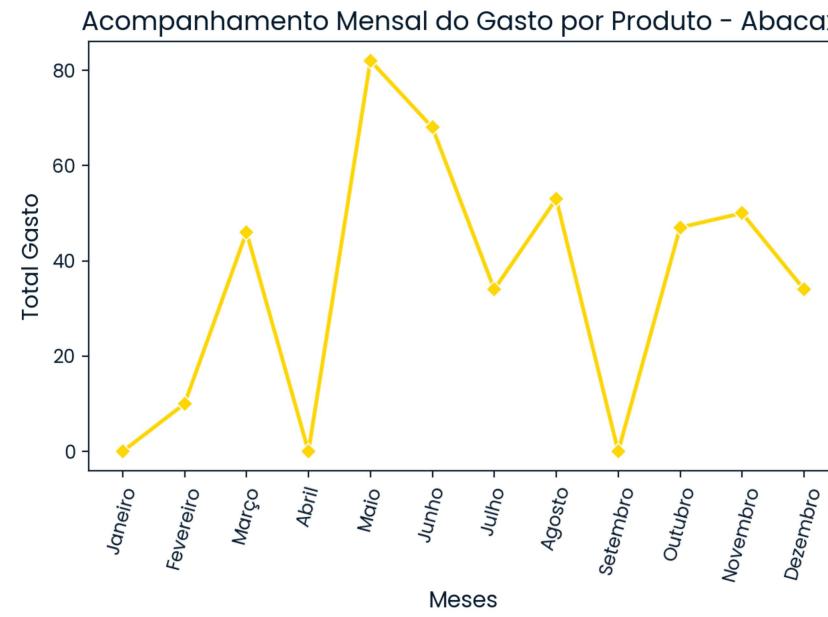
# Criando o gráfico de linhas
sns.lineplot(
    data=df_abacaxi_group,
    x="meses",
    y="total_quantidade",
    marker="D",
    linewidth=2,
    color="gold"
)

# Nomeando o gráfico
plt.title("Acompanhamento Mensal do Gasto por Produto - Abacaxi", fontsize=14)
plt.xlabel("Meses", fontsize=12)
plt.ylabel("Total Gasto", fontsize=12)

# Rotação da letra para x da figura
plt.xticks(rotation=75)

# Plotando o gráfico
plt.tight_layout()
plt.show()

```



- Valor Total vendido ao longo dos meses:

```

# Criando o gráfico de linhas
sns.lineplot(
    data=df_abacaxi,
    x="meses",
    y="total_gasto",
    marker="o",
    linewidth=2,
    color="gold"
)

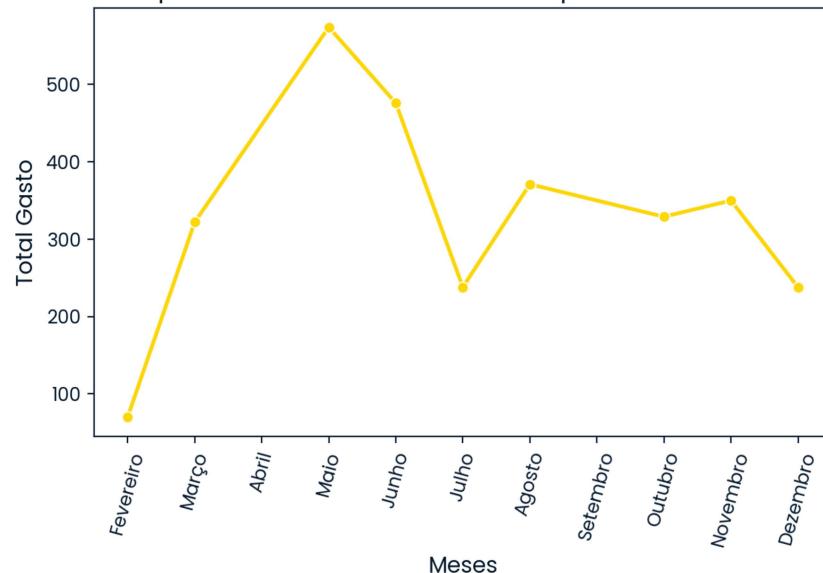
# Nomeando o gráfico
plt.title("Acompanhamento Mensal do Gasto por Produto - Abacaxi", fontsize=14)
plt.xlabel("Meses", fontsize=12)
plt.ylabel("Total Gasto", fontsize=12)

# Rotação da letra para x da figura
plt.xticks(rotation=75)

# Plotando o gráfico
plt.tight_layout()
plt.show()

```

Acompanhamento Mensal do Gasto por Produto - Abacaxi



Anotações - Produto Abacaxi:

Para abacaxi é claro um pico de vendas no período do 2º trimestre. Isto pode ser resultante de diversos fatores.

Suposições/Padrões:

- Como se trata de um df relacionado a frutas e, é de conhecimento comum, variações como está vista pode ser referente, por exemplo, ao período em que a fruta está em boa qualidade e maior abundância para a venda.
- Outra suposição a ser apontada, está na condições climáticas, em um df orgânico com a disposição de dados ao longo de vários anos, podemos ter a sorte de notar eventos climáticos e como eles podem atingir este tipo de df.

Insight:

- Relacionada a este tipo de df, a sazonalidade tem papel chave, pois a busca um certo tipo de fruta a depender do mês em que estamos é preferível frente a outras. Por exemplo, em períodos de calor o consumo do abacaxi pode ser a opção, frente a uva, por exemplo.

Uva:

- Quantidade Total Vendida ao longo dos meses:

```
# Filtrando o df para apenas Uva
df_uva = df_vendas[df_vendas['produto'] == "Uva"]

# Criando um df agrupado por meses e com a quantidade total vendida por mês
df_uva_group = df_uva.groupby("meses", as_index=False)[["quantidade"]].sum()

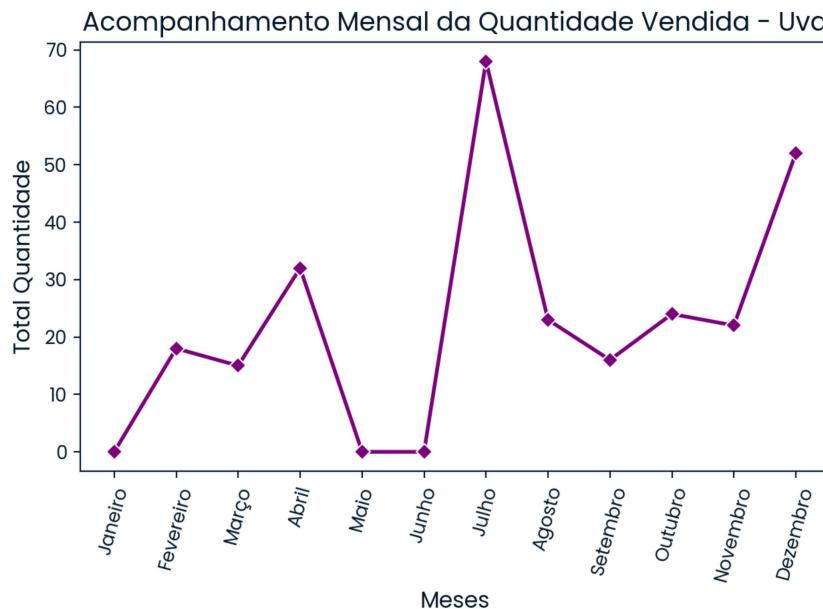
# Renomeando a coluna do groupby
df_uva_group = df_uva_group.rename(columns={"quantidade": "total_quantidade"})

# Criando o gráfico de linhas
sns.lineplot(
    data=df_uva_group,
    x="meses",
    y="total_quantidade",
    marker="D",
    linewidth=2,
    color="purple"
)

# Nomeando o gráfico
plt.title("Acompanhamento Mensal da Quantidade Vendida - Uva", fontsize=14)
plt.xlabel("Meses", fontsize=12)
plt.ylabel("Total Quantidade", fontsize=12)

# Rotação da letra para x da figura
plt.xticks(rotation=75)

# Plotando o gráfico
plt.tight_layout()
plt.show()
```



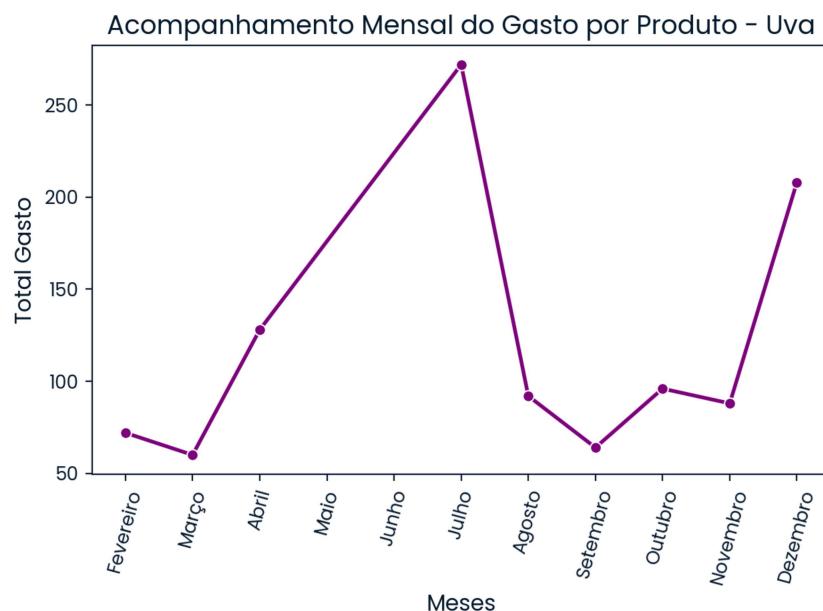
- Valor total Vendido por Mês:

```
# Criando o gráfico de linhas
sns.lineplot(
    data=df_uva,
    x="meses",
    y="total_gasto",
    marker="o",
    linewidth=2,
    color="purple"
)

# Nomeando o gráfico
plt.title("Acompanhamento Mensal do Gasto por Produto - Uva", fontsize=14)
plt.xlabel("Meses", fontsize=12)
plt.ylabel("Total Gasto", fontsize=12)

# Rotação da letra para x da figura
plt.xticks(rotation=75)

# Plotando o gráfico
plt.tight_layout()
plt.show()
```



Anotações - Produto Uva:

Suposições/Padrões:

- A visualização de um pico de vendas em uva para o final do ano é esperado uma vez que bate com datas festivas como o natal e o ano novo.
- Como de praxe para frutas, a sazonalidade também tem influência, períodos que tem mais chuvas e frio podem também ser responsáveis por esses aumentos.

Insight:

- Como já temos essa tendência vista para o final de ano e, provavelmente relacionado as datas comemorativas, trazer publicidades que relacionem uvas a estas datas podem trazer uma visibilidade ainda maior para um público que já está mais disposto a comprar.

Laranja

- Quantidade Total vendido ao longo dos meses:

```
# Filtrando o df para apenas Uva
df_laranja = df_vendas[df_vendas['produto'] == "Laranja"]

# Criando um df agrupado por meses e com a quantidade total vendida por mês
df_laranja_group = df_laranja.groupby("meses", as_index=False)[["quantidade"]].sum()

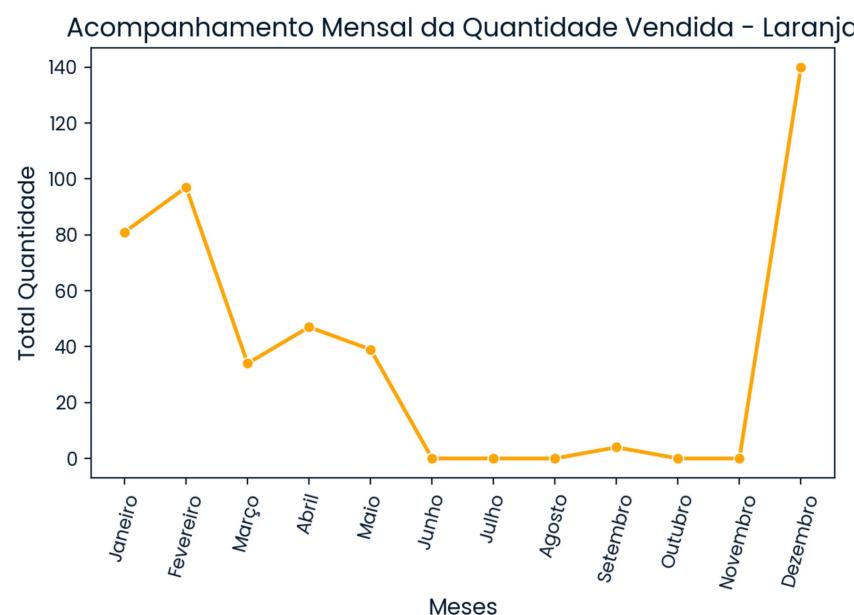
# Renomeando a coluna do groupby
df_laranja_group = df_laranja_group.rename(columns={"quantidade": "total_quantidade"})

# Criando o gráfico de linhas
sns.lineplot(
    data=df_laranja_group,
    x="meses",
    y="total_quantidade",
    marker="o",
    linewidth=2,
    color="orange"
)

# Nomeando o gráfico
plt.title("Acompanhamento Mensal da Quantidade Vendida - Laranja", fontsize=14)
plt.xlabel("Meses", fontsize=12)
plt.ylabel("Total Quantidade", fontsize=12)

# Rotação da letra para x da figura
plt.xticks(rotation=75)

# Plotando o gráfico
plt.tight_layout()
plt.show()
```



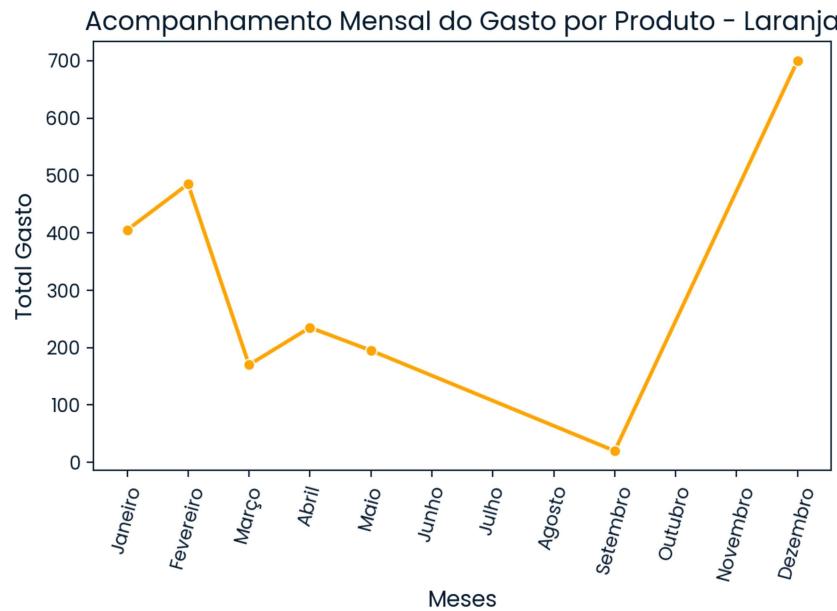
- Valor total Vendido por Mês:

```
# Criando o gráfico de linhas
sns.lineplot(
    data=df_laranja,
    x="meses",
    y="total_gasto",
    marker="o",
    linewidth=2,
    color="orange"
)

# Nomeando o gráfico
plt.title("Acompanhamento Mensal do Gasto por Produto - Laranja", fontsize=14)
plt.xlabel("Meses", fontsize=12)
plt.ylabel("Total Gasto", fontsize=12)
```

```
# Rotação da letra para x da figura  
plt.xticks(rotation=75)
```

```
# Plotando o gráfico  
plt.tight_layout()  
plt.show()
```



Anotações - Produto Laranja:

Neste gráfico para laranja é clara o pulo de dados em dezembro. O qual vai diminuindo ao longo do ano.

Suposições/Padrões:

- Provavelmente devido a quantidade de dados possíveis (problemática também devido a dados criados, não orgânicos) a visualização de laranjas não coincide com os períodos mais comuns de consumo algo em torno de maio e agosto
- Padrão comum de consumo de frutas cítricas e especialmente a laranja, a qual é muito conhecida por seus fatores de saúde, está em épocas mais frias por conta da maior quantidade de pessoas pegam gripes.

Insight:

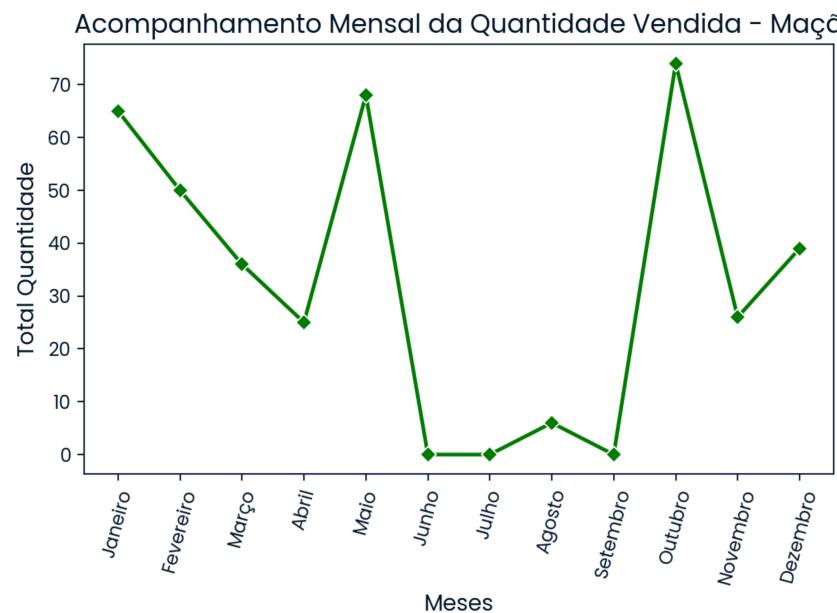
- A laranja normalmente é uma fruta que podemos identificar com maior facilidade seus períodos de sazonalidade. Por conta disso, realizar planejamentos voltados à certas épocas específicas e publicidades voltadas a públicos que buscam saúde, poderia ser uma grande virada de chave para o sucesso de vendas em determinado período.

Maçã

- Quantidade Total vendido ao longo dos meses:

```
# Filtrando o df para apenas maçã  
df_maca = df_vendas[df_vendas['produto'] == "Maçã"]  
  
# Criando um df agrupado por meses e com a quantidade total vendida por mes  
df_maca_group = df_maca.groupby("meses", as_index=False)[["quantidade"]].sum()  
  
# Renomenando a coluna do groupby  
df_maca_group = df_maca_group.rename(columns={"quantidade": "total_quantidade"})  
  
# Criando o gráfico de linhas  
sns.lineplot(  
    data=df_maca_group,  
    x="meses",  
    y="total_quantidade",  
    marker="D",  
    linewidth=2,  
    color="green")  
  
# Nomeando o gráfico  
plt.title("Acompanhamento Mensal da Quantidade Vendida - Maçã", fontsize=14)  
plt.xlabel("Meses", fontsize=12)  
plt.ylabel("Total Quantidade", fontsize=12)  
  
# Rotação da letra para x da figura  
plt.xticks(rotation=75)
```

```
# Plotando o gráfico
plt.tight_layout()
plt.show()
```



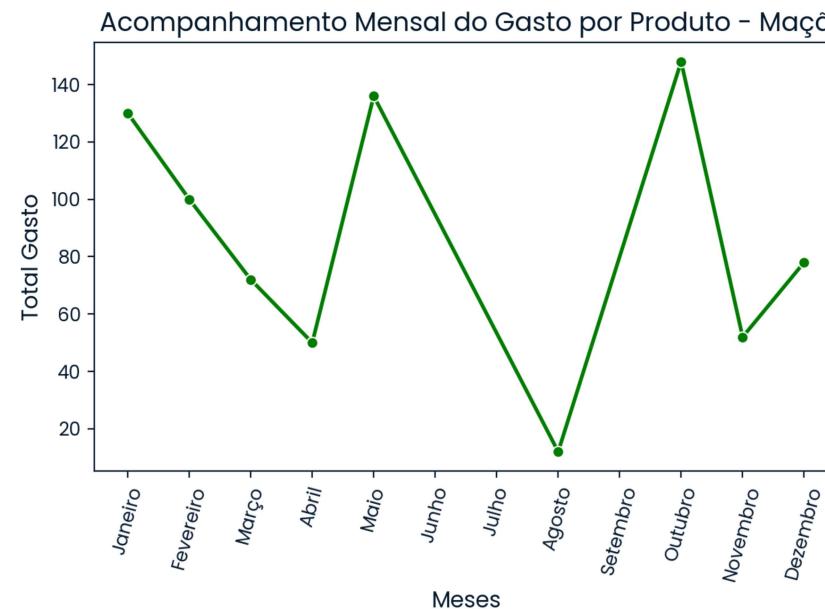
- Valor total Vendido por Mês:

```
# Criando o gráfico de linhas
sns.lineplot(
    data=df_maca,
    x="meses",
    y="total_gasto",
    marker="o",
    linewidth=2,
    color="green"
)

# Nomeando o gráfico
plt.title("Acompanhamento Mensal do Gasto por Produto - Maçã", fontsize=14)
plt.xlabel("Meses", fontsize=12)
plt.ylabel("Total Gasto", fontsize=12)

# Rotação da letra para x da figura
plt.xticks(rotation=75)

# Plotando o gráfico
plt.tight_layout()
plt.show()
```



Anotações - Produto Maçã:

Suposições/Padrões:

- Provavelmente também afetada pelos dados não orgânicos, temos grandes variações na quantidade de maçãs vendidas ao longo do ano, consequentemente, afetando a total arrecadado com a maçã.

Insight:

- Essas quedas bruscas em relação às vendas de maçãs, demonstram um espaço para estratégias de marketing voltadas à atração do público a tais benefícios, como também a investigação profunda do que pode estar acontecendo ao longo do ano que motive tais variações.

Manga

- Quantidade Total vendido ao longo dos meses:

```
# Filtrando o df para apenas manga
df_manga = df_vendas[df_vendas['produto'] == "Manga"]

# Criando um df agrupado por meses e com a quantidade total vendida por mês
df_manga_group = df_manga.groupby("meses", as_index=False)[["quantidade"]].sum()

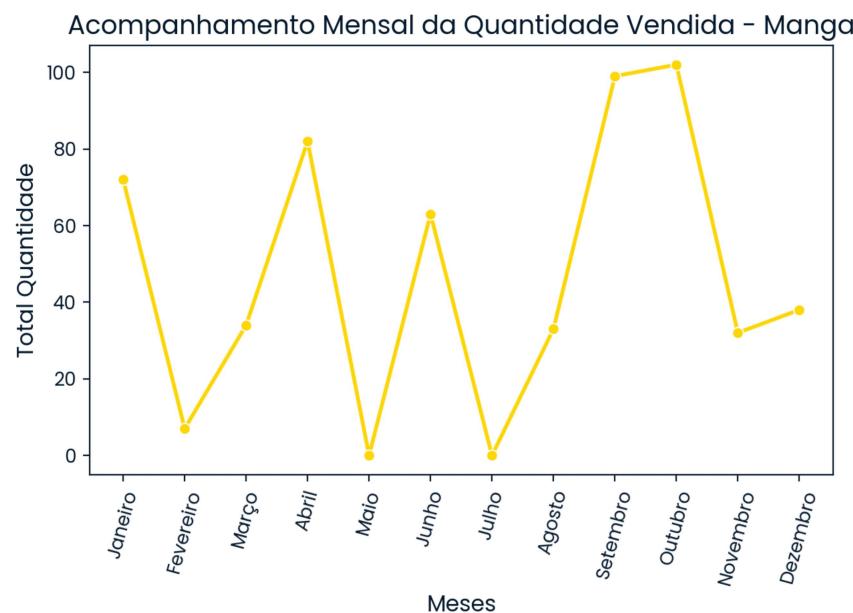
# Renomeando a coluna do groupby
df_manga_group = df_manga_group.rename(columns={"quantidade": "total_quantidade"})

# Criando o gráfico de linhas
sns.lineplot(
    data=df_manga_group,
    x="meses",
    y="total_quantidade",
    marker="o",
    linewidth=2,
    color="gold"
)

# Nomeando o gráfico
plt.title("Acompanhamento Mensal da Quantidade Vendida - Manga", fontsize=14)
plt.xlabel("Meses", fontsize=12)
plt.ylabel("Total Quantidade", fontsize=12)

# Rotação da letra para x da figura
plt.xticks(rotation=75)

# Plotando o gráfico
plt.tight_layout()
plt.show()
```



- Valor total Vendido por Mês:

```
# Criando o gráfico de linhas
sns.lineplot(
    data=df_manga,
    x="meses",
    y="total_gasto",
    marker="o",
    linewidth=2,
    color="gold"
```

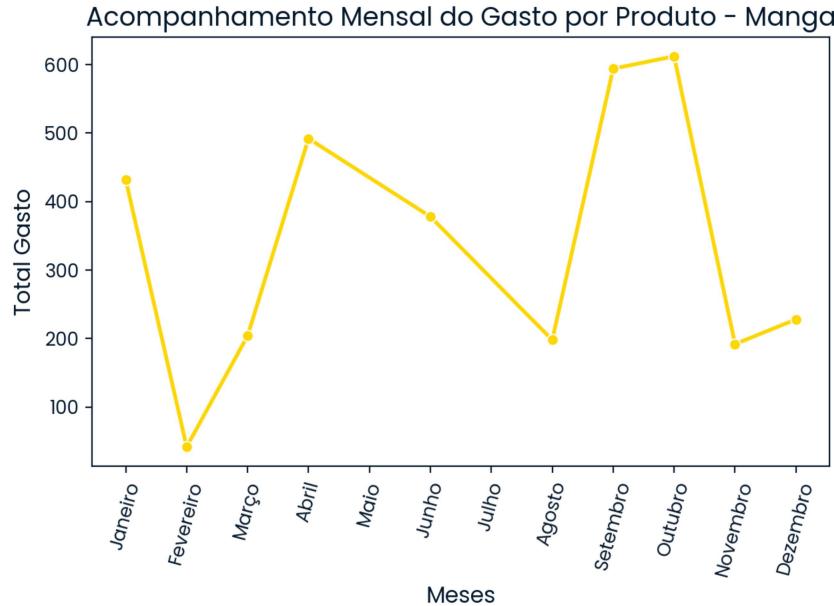
```

)
# Nomeando o gráfico
plt.title("Acompanhamento Mensal do Gasto por Produto - Manga", fontsize=14)
plt.xlabel("Meses", fontsize=12)
plt.ylabel("Total Gasto", fontsize=12)

# Rotação da letra para x da figura
plt.xticks(rotation=75)

# Plotando o gráfico
plt.tight_layout()
plt.show()

```



Anotações - Produto Manga:

Suposições/Padrões:

- Apesar da notável variação da quantidade vendida ao longo dos meses, isto parece reflexo de poucos dados, mas que ainda assim traz um padrão de consistência ao longo dos meses pelo ano.
- Consequência destas consistência ao longo do ano, será a receita mais estável ao longo do ano.

Insight:

- Visto um maior solidez em receita ao longo do ano, a empresa pode utilizar isso como forma de tentar diversificar em novos produtos que tenham o sabor "Manga" ou outras frutas que tenham maior semelhança, uma vez que é, se parece ser uma demanda mais constante.

Pera

- Quantidade Total vendido ao longo dos meses:

```

# Filtrando o df para apenas pera
df_pera = df_vendas[df_vendas['produto'] == "Pera"]

# Criando um df agrupado por meses e com a quantidade total vendida por mes
df_pera_group = df_pera.groupby("meses", as_index=False)[["quantidade"]].sum()

# Renomenando a coluna do groupby
df_pera_group = df_pera_group.rename(columns={"quantidade": "total_quantidade"})

# Criando o gráfico de linhas
sns.lineplot(
    data=df_pera_group,
    x="meses",
    y="total_quantidade",
    marker="o",
    linewidth=2,
    color="lightgreen"
)

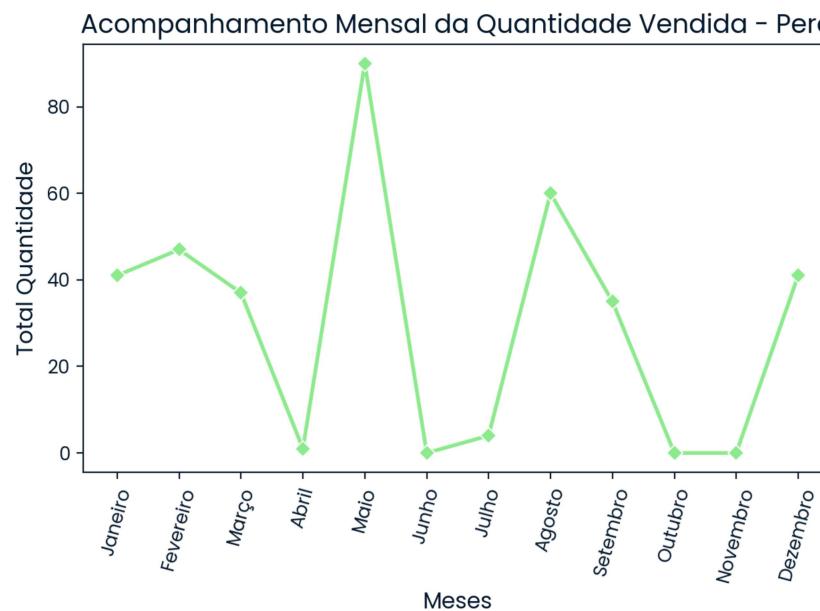
# Nomeando o gráfico
plt.title("Acompanhamento Mensal da Quantidade Vendida - Pera", fontsize=14)
plt.xlabel("Meses", fontsize=12)
plt.ylabel("Total Quantidade", fontsize=12)

# Rotação da letra para x da figura
plt.xticks(rotation=75)

```

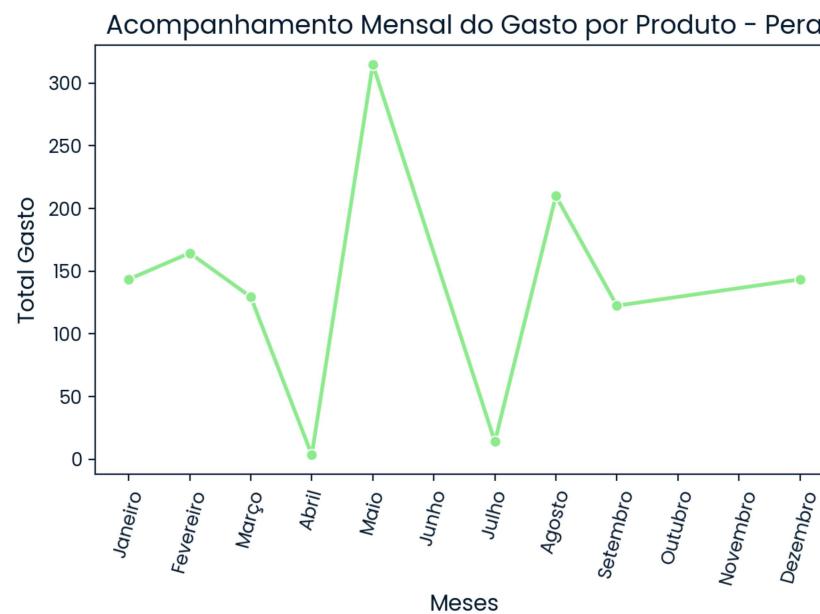
```
plt.xticks(rotation=75)
```

```
# Plotando o gráfico  
plt.tight_layout()  
plt.show()
```



- Valor total Vendido por Mês:

```
# Criando o gráfico de linhas  
sns.lineplot(  
    data=df_pera,  
    x="meses",  
    y="total_gasto",  
    marker="o",  
    linewidth=2,  
    color="lightgreen"  
)  
  
# Nomeando o gráfico  
plt.title("Acompanhamento Mensal do Gasto por Produto - Pera", fontsize=14)  
plt.xlabel("Meses", fontsize=12)  
plt.ylabel("Total Gasto", fontsize=12)  
  
# Rotação da letra para x da figura  
plt.xticks(rotation=75)  
  
# Plotando o gráfico  
plt.tight_layout()  
plt.show()
```



Anotações - Produto Pera:

Suposições/Padrões:

- É notável pico de vendas durante o período do 2º trimestre do ano.
- Outros períodos do ano se mostraram com maiores variações, apesar do 3º trimestre também trazer uma maior estabilidade.

Insight:

- Uma movimentação que poderia ser realizada é trazer ênfase ao 2º e 3º trimestre do ano voltado à pera. Esse planejamento poderia trazer uma pontencialização dos lucros dentro deste período. Podendo servir de base para novas ações ou até suporte financeiro no meio do ano para outras áreas da empresa.

Banana

- Quantidade Total vendido ao longo dos meses:

```
# Filtrando o df para apenas banana
df_banana = df_vendas[df_vendas['produto'] == "Banana"]

# Criando um df agrupado por meses e com a quantidade total vendida por mes
df_banana_group = df_banana.groupby("meses", as_index=False)[["quantidade"]].sum()

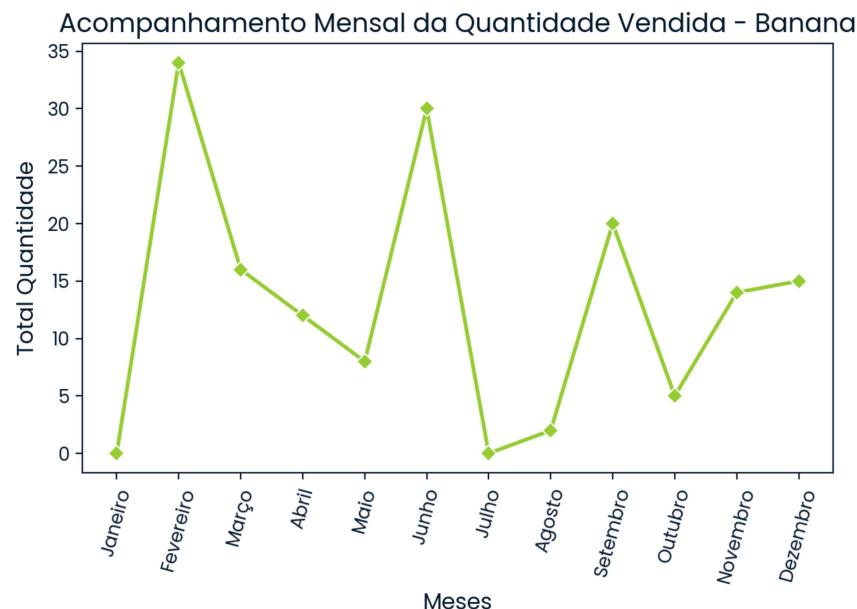
# Renomenando a coluna do groupby
df_banana_group = df_banana_group.rename(columns={"quantidade": "total_quantidade"})

# Criando o gráfico de linhas
sns.lineplot(
    data=df_banana_group,
    x="meses",
    y="total_quantidade",
    marker="D",
    linewidth=2,
    color="yellowgreen"
)

# Nomeando o gráfico
plt.title("Acompanhamento Mensal da Quantidade Vendida - Banana", fontsize=14)
plt.xlabel("Meses", fontsize=12)
plt.ylabel("Total Quantidade", fontsize=12)

# Rotação da letra para x da figura
plt.xticks(rotation=75)

# Plotando o gráfico
plt.tight_layout()
plt.show()
```



- Valor total Vendido por Mês:

```
# Criando o gráfico de linhas
sns.lineplot(
    data=df_banana,
    x="meses",
    y="total_gasto",
```

```

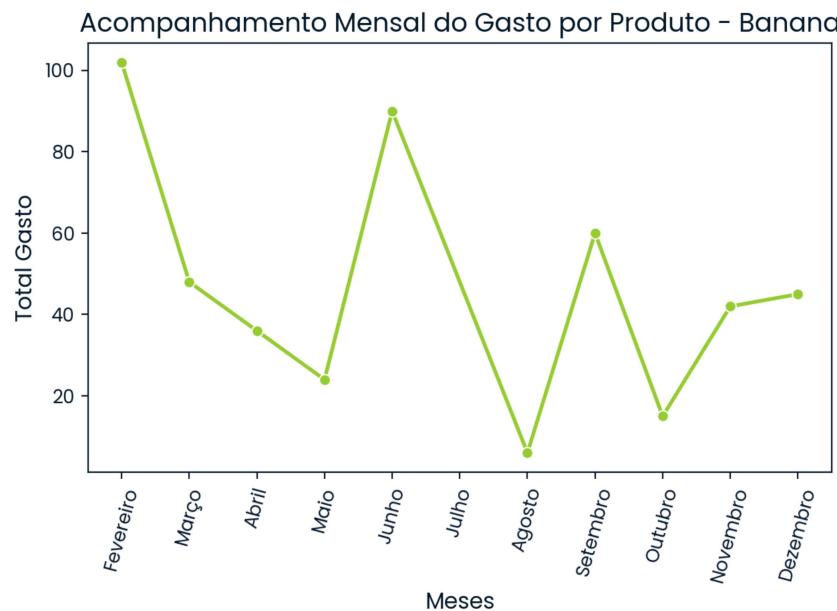
marker="o",
linewidth=2,
color="yellowgreen"
)

# Nomeando o gráfico
plt.title("Acompanhamento Mensal do Gasto por Produto - Banana", fontsize=14)
plt.xlabel("Meses", fontsize=12)
plt.ylabel("Total Gasto", fontsize=12)

# Rotação da letra para x da figura
plt.xticks(rotation=75)

# Plotando o gráfico
plt.tight_layout()
plt.show()

```



Anotações - Produto Banana:

Suposições/Padrões:

- A tendência notável nestes dados está com um bom começo de ano a partir de fevereiro e há uma descida com padrões de velas ao longo do ano. Com isso, há períodos de grandes quedas e grandes retornos.

Insight:

- Com esse padrão notado, poderíamos buscar criar estratégias de manutenção da quantidade de vendas ao longo do ano, respeitando claro a sazonalidade da fruta, mas que não trouxesse diminuições bruscas ao longo dos meses do ano.
- Algo como promoções em épocas mais próximas ao final do ano, publicidades educacionais da implementação do produto em uma rotina saudável e agitada, entre outras ideias que poderiam ser trazidas a fim de garantir essa manutenção das vendas.

Parte 2.

1 - Encontrando Produto e seu Total de Vendas:

Usando sql para fazer uma busca nos dados que preparamos com python.

Lógica:

- A ideia dessa query foi agrupar por produto e categoria e então fazer o cálculo do total de vendas. Por fim, ainda ordenando do maior para o menor de acordo com o total de vendas. Com isso, trazemos o select com produto, categoria e o total de vendas o que atende aos requisitos pedidos.

⊕ DataFrames and CSVs DataFrame as

```

SELECT produto, categoria, SUM(quantidade * preco) as totalVendas
FROM 'data_clean.csv'
GROUP BY produto, categoria
ORDER BY totalVendas DESC;

```

v	produto	v	categoria
0	Manga		Frutas Tropicais
1	Abacaxi		Frutas Tropicais