

IA aplicada a Data Science: utilizando IA na construção de visualização de dados



Aula 1 - Conhecendo os dados

A **Zoop** é uma grande varejista que atende a todas as regiões do Brasil por meio do seu **e-commerce**. Ela é conhecida pela ampla variedade em seus produtos buscando atender a todo tipo de público. Ela possui uma série de centros de distribuição (CDs) para entrega de seus produtos, com grande parte situada no estado de São Paulo.

Para gerenciar o seu alcance, bem como o faturamento de seu setor em lojas online, ela consolida os dados em diferentes períodos de tempo e avalia esses dados para gerar insights e tomar algumas decisões estratégicas em seu negócio. Neste projeto, vamos ter acesso aos dados de parte da sua clientela do e-commerce dentro do ano de 2023.

Você, como **analista de dados** da empresa, precisará gerar visuais que auxiliem na construção de relatórios de acordo com algumas premissas passadas pelas partes interessadas realizando uma rápida análise do público que possuímos na loja virtual e do faturamento da empresa.

Problema de negócio:

O time de dados da **Zoop** precisa extrair os dados e gerar informações por meio de visuais que possam ser apresentados a diretoria da empresa apontando os dados de faturamento, perfil do cliente e outros indicadores que possam auxiliar na tomada de decisão em sua loja online.

Base de dados

Vamos importar duas bases de dados:

- Dados de clientes do e-commerce da Zoop, separados pelo código identificador da compra.

- Dados de vendas do e-commerce da Zoop em 2023, separados pelo código identificador da compra.

Esses dados serão lidos a partir do repositório compartilhado pelo GitHub.

Desafio

Você, como analista de dados do time de dados da Zoop, tem o desafio de extrair os dados de ambas as bases e construir visuais que possam agregar valor a apresentação dos resultados da Zoop em 2023. Para isso, serão repassados ao todo **7**

questionamentos que foram separados para que você possa contribuir na construção do storytelling das vendas da empresa.

Para agilizar o processo da análise exploratória dos dados (AED) e a criação dos visuais, utilizaremos a IA do **ChatGPT** como nossa assistente, tudo isso levando em conta o prazo curto para as análises e a qualidade da entrega.

1.2 - Obtendo os dados

Vamos inicialmente passar o link do repositório com as duas bases que utilizaremos no projeto:

```
In [787... url_clientes = "https://raw.githubusercontent.com/alura-cursos/ia-datascience-da
url_vendas = "https://raw.githubusercontent.com/alura-cursos/ia-datascience-data
```

Após esta etapa, vamos criar 2 dataframes com os dados dos clientes e das vendas da Zoop separadamente e dar uma pequena checada nas tabelas extraídas.

```
In [788... import pandas as pd
```

```
In [789... clientes = pd.read_csv(url_clientes)
```

```
In [790... clientes.head()
```

```
Out[790... 
```

	ID_compra	ID_cliente	cidade	uf	regiao	idade	sexo_biologico	cashback
0	904	8972	Natal	Rio Grande do Norte	Nordeste	49	Masculino	Não
1	4613	5120	São Paulo	São Paulo	Sudeste	49	Feminino	Não
2	5579	1847	São Paulo	São Paulo	Sudeste	24	Feminino	Não
3	2405	2846	São Paulo	São Paulo	Sudeste	38	Feminino	Sim
4	2485	2715	Salvador	Bahia	Nordeste	30	Feminino	Não

```
In [791... print(clientes.columns.tolist())
```

```
['ID_compra', 'ID_cliente', 'cidade', 'uf', 'regiao', 'idade', 'sexo_biologico', 'cashback', 'avaliacao_compra']
```

```
In [792...] vendas = pd.read_csv(url_vendas)
```

```
In [793...] vendas.tail()
```

```
Out[793...]
      ID_compra  data  horario  categoria  preco_unitario  quantidade  frete
6164      6141  2023-12-31  17:49  Eletrodomésticos      1509.9           5  377.48
6165      6148  2023-12-31  23:02    Papelaria e      14.0           2    3.36
        escritório
6166      6123  2023-12-31  21:25    Alimentos e      77.9           1    7.79
        bebidas
6167      6147  2023-12-31  21:38         Livros     142.9           4   17.15
6168      6122  2023-12-31  22:11    Beleza e      89.9           5   13.48
        cuidados
        pessoais
```



```
In [794...] print(vendas.columns.tolist())
```

```
['ID_compra', 'data', 'horario', 'categoria', 'preco_unitario', 'quantidade', 'frete', 'metodo_pagamento']
```

1.3 - Entendendo os dados

Agora, vamos explorar os dados compreendendo o que significam e como podemos relacioná-los na construção dos visuais.

Base de clientes

Prompt:

Vamos atuar como analista de dados de um e-commerce. Recebemos duas bases de dados com os dados dos clientes e das vendas em 2023, respectivamente.

Neste primeiro momento, vamos explorar a base clientes no formato de um DataFrame, chamado "clientes". Ela contém dados dos clientes que realizaram comprar na loja virtual, sendo eles: ID da compra, ID do cliente, a cidade, estado e região da compra, idade, sexo biológico, se participam do programa de cashback e avaliação da compra.

Queremos fazer uma breve exploração dos dados para compreender um pouco sobre eles. Lembrando que os dados desta base já estão limpos e tratados. Construa um código para executar esta etapa usando o Python.

Não é necessário utilizar "print()" para mostrar o resultado das consultas.

```
In [795... # Exibir as primeiras linhas do DataFrame
clientes.head()
```

Out[795...

	ID_compra	ID_cliente	cidade	uf	regiao	idade	sexo_biologico	cashback
0	904	8972	Natal	Rio Grande do Norte	Nordeste	49	Masculino	Não
1	4613	5120	São Paulo	São Paulo	Sudeste	49	Feminino	Não
2	5579	1847	São Paulo	São Paulo	Sudeste	24	Feminino	Não
3	2405	2846	São Paulo	São Paulo	Sudeste	38	Feminino	Sim
4	2485	2715	Salvador	Bahia	Nordeste	30	Feminino	Não



```
In [796... # Informações gerais sobre o DataFrame
clientes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6169 entries, 0 to 6168
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ID_compra              6169 non-null  int64
1   ID_cliente             6169 non-null  int64
2   cidade                 6169 non-null  object
3   uf                     6169 non-null  object
4   regiao                 6169 non-null  object
5   idade                  6169 non-null  int64
6   sexo_biologico         6169 non-null  object
7   cashback               6169 non-null  object
8   avaliacao_compra       6169 non-null  int64
dtypes: int64(4), object(5)
memory usage: 433.9+ KB
```

```
In [797... # Verificar estatísticas descritivas das variáveis numéricas e de texto
clientes.describe(include='all')
```

Out[797...

	ID_compra	ID_cliente	cidade	uf	regiao	idade	sexo_biologico
count	6169.000000	6169.000000	6169	6169	6169	6169.000000	6169
unique	NaN	NaN	96	27	5	NaN	2
top	NaN	NaN	São Paulo	São Paulo	Sudeste	NaN	Feminino
freq	NaN	NaN	458	1500	2666	NaN	3371
mean	3085.000000	5461.475280	NaN	NaN	NaN	38.260010	NaN
std	1780.981237	2610.959977	NaN	NaN	NaN	9.838951	NaN
min	1.000000	1005.000000	NaN	NaN	NaN	18.000000	NaN
25%	1543.000000	3148.000000	NaN	NaN	NaN	31.000000	NaN
50%	3085.000000	5436.000000	NaN	NaN	NaN	38.000000	NaN
75%	4627.000000	7748.000000	NaN	NaN	NaN	45.000000	NaN
max	6169.000000	9999.000000	NaN	NaN	NaN	70.000000	NaN



In [798...

```
# Verificar a distribuição de valores categóricos
clientes['sexo_biologico'].value_counts()
```

Out[798...

```
sexo_biologico
Feminino      3371
Masculino     2798
Name: count, dtype: int64
```

In [799...

```
# Verificar a distribuição de clientes por região
clientes['regiao'].value_counts()
```

Out[799...

```
regiao
Sudeste      2666
Nordeste     1882
Sul          819
Centro-Oeste 495
Norte        307
Name: count, dtype: int64
```

In [800...

```
# Verificar a distribuição de clientes por estado
clientes['uf'].value_counts()
```

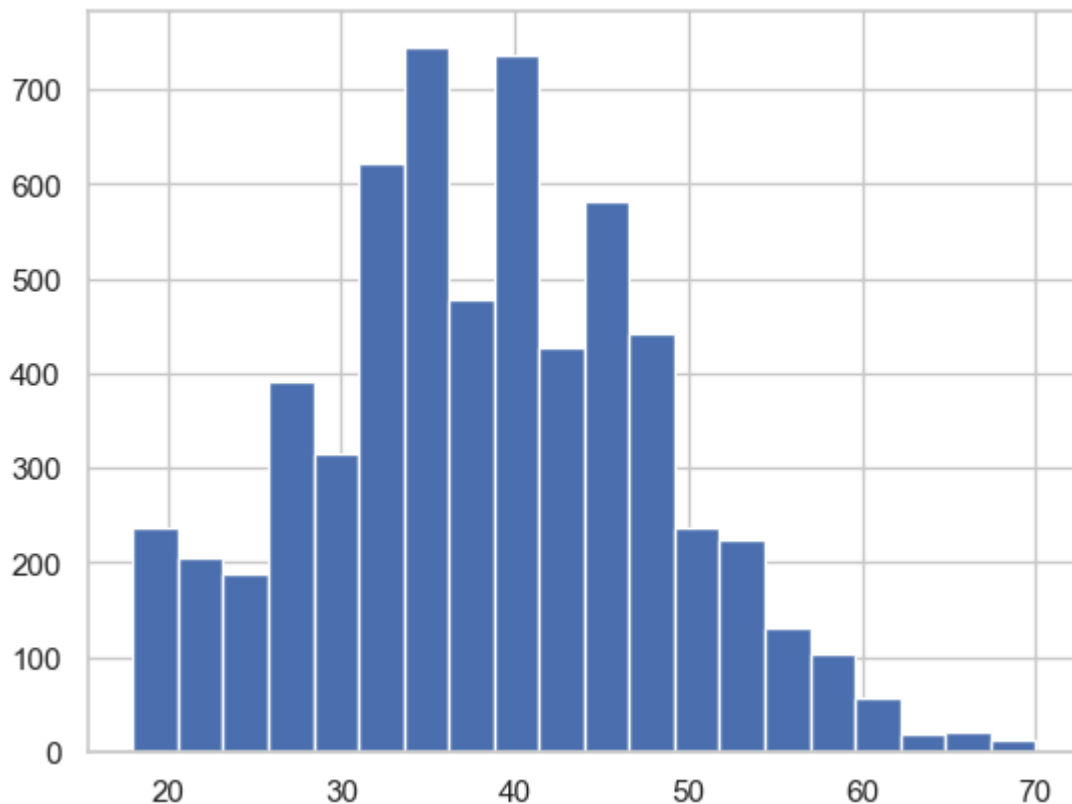
```
Out[800...  uf
            São Paulo      1500
            Rio de Janeiro  570
            Minas Gerais   478
            Bahia          453
            Pernambuco     411
            Paraná         383
            Rio Grande do Sul 329
            Ceará          284
            Paraíba        279
            Distrito Federal 263
            Rio Grande do Norte 244
            Goiás          120
            Espírito Santo  118
            Pará           111
            Santa Catarina  107
            Piauí          83
            Mato Grosso    58
            Mato Grosso do Sul 54
            Amazonas       51
            Maranhão       45
            Sergipe        43
            Alagoas        40
            Roraima        38
            Amapá          35
            Rondônia       32
            Acre           30
            Tocantins      10
            Name: count, dtype: int64
```

```
In [801... # Verificar a distribuição de clientes por cidade
clientes['cidade'].value_counts().head(10) # Top 10 cidades
```

```
Out[801... cidade
            São Paulo      458
            Salvador       382
            Rio de Janeiro  284
            Brasília       263
            Porto Alegre   260
            Belo Horizonte  259
            Fortaleza      258
            Curitiba       255
            Recife         252
            Campinas       250
            Name: count, dtype: int64
```

```
In [802... # Verificar a distribuição da idade
clientes['idade'].describe()
clientes['idade'].hist(bins=20)
```

```
Out[802... <AxesSubplot: >
```



```
In [803... # Verificar participação no programa de cashback
clientes['cashback'].value_counts(normalize=True)
```

```
Out[803... cashback
Não      0.576593
Sim      0.423407
Name: proportion, dtype: float64
```

```
In [804... # Verificar avaliação média das compras
clientes['avaliacao_compra'].describe()
clientes['avaliacao_compra'].value_counts().sort_index()
```

```
Out[804... avaliacao_compra
0      32
1      33
2      58
3      60
4     113
5      90
6     135
7     689
8     923
9    2519
10   1517
Name: count, dtype: int64
```

```
In [805... # Analisar a idade média por sexo
clientes.groupby('sexo_biologico')['idade'].mean()
```

```
Out[805... sexo_biologico
Feminino    38.318896
Masculino   38.189064
Name: idade, dtype: float64
```

```
In [806... # Analisar a avaliação média por região
clientes.groupby('regiao')['avaliacao_compra'].mean()
```

```
Out[806... regiao
Centro-Oeste    8.377778
Nordeste        8.391605
Norte           8.537459
Sudeste         8.476369
Sul             8.461538
Name: avaliacao_compra, dtype: float64
```

Base de vendas

```
In [807... # Analisar a taxa de participação no cashback por região
clientes.groupby('regiao')['cashback'].value_counts(normalize=True).unstack()
```

```
Out[807...      cashback      Não      Sim
regiao
Centro-Oeste  0.579798  0.420202
Nordeste      0.578108  0.421892
Norte         0.543974  0.456026
Sudeste       0.584396  0.415604
Sul           0.557998  0.442002
```

```
In [808... # Verificar quantos clientes únicos existem
clientes['ID_cliente'].nunique()
```

```
Out[808... 4453
```

Prompt:

Agora vamos explorar a base de vendas no formato de um DataFrame, chamado "vendas". Ela contém dados das vendas do e-commerce, sendo eles: ID da compra, data da compra, horário da compra, categoria do produto, preço unitário do produto, quantidade vendida, frete e o método do pagamento.

Queremos fazer uma breve exploração dos dados para compreender um pouco sobre eles. Lembrando que os dados desta base também já estão limpos e tratados. Construa um código para executar esta etapa usando o Python.

```
In [809... # Exibir as primeiras linhas do DataFrame
vendas.head()
```


Out[809...

	ID_compra	data	horario	categoria	preco_unitario	quantidade	frete	metodo
0	6	2023-01-03	20:40	Eletrônicos	4457.9	4	713.26	
1	5	2023-01-03	21:48	Ferramentas e construção	115.9	1	0.00	
2	9	2023-01-03	17:08	Beleza e cuidados pessoais	68.9	2	13.78	Cartão
3	4	2023-01-03	14:29	Ferramentas e construção	80.9	5	40.45	
4	8	2023-01-03	18:02	Beleza e cuidados pessoais	165.0	4	0.00	

In [810...

```
# Informações gerais sobre o DataFrame
vendas.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6169 entries, 0 to 6168
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ID_compra              6169 non-null   int64
1   data                   6169 non-null   object
2   horario                 6169 non-null   object
3   categoria               6169 non-null   object
4   preco_unitario         6169 non-null   float64
5   quantidade             6169 non-null   int64
6   frete                  6169 non-null   float64
7   metodo_pagamento      6169 non-null   object
dtypes: float64(2), int64(2), object(4)
memory usage: 385.7+ KB
```

In [811...

```
vendas["data"] = pd.to_datetime(vendas["data"], format="%Y-%m-%d")
```

In [812...

```
# Estatísticas descritivas para variáveis numéricas
vendas.describe()
```

Out[812...

	ID_compra	data	preco_unitario	quantidade	frete
count	6169.000000	6169	6169.000000	6169.000000	6169.000000
mean	3085.000000	2023-08-20 00:12:22.292105984	873.234738	3.004863	119.233832
min	1.000000	2023-01-03 00:00:00	2.900000	1.000000	0.000000
25%	1543.000000	2023-05-30 00:00:00	44.900000	2.000000	0.320000
50%	3085.000000	2023-08-31 00:00:00	93.900000	3.000000	7.550000
75%	4627.000000	2023-11-23 00:00:00	221.900000	4.000000	29.240000
max	6169.000000	2023-12-31 00:00:00	8000.000000	5.000000	4521.540000
std	1780.981237	NaN	1852.064342	1.420725	366.149225

In [813...

```
# Verificar a distribuição de categorias do produto
vendas['categoria'].value_counts()
```

Out[813...

```
categoria
Alimentos e bebidas      1495
Beleza e cuidados pessoais 1062
Casa e limpeza           1041
Eletrodomésticos        665
Eletrônicos              580
Papeleria e escritório   551
Ferramentas e construção 474
Livros                   301
Name: count, dtype: int64
```

In [814...

```
# Verificar os métodos de pagamento mais utilizados
vendas['metodo_pagamento'].value_counts(normalize=True)
```

Out[814...

```
metodo_pagamento
Cartão de Crédito      0.462149
PIX                    0.363754
Cartão de Débito       0.088669
Boleto                 0.085427
Name: proportion, dtype: float64
```

In [815...

```
# Verificar o intervalo de datas das vendas
vendas['data'].min(), vendas['data'].max()
```

Out[815...

```
(Timestamp('2023-01-03 00:00:00'), Timestamp('2023-12-31 00:00:00'))
```

In [816...

```
# Analisar vendas por mês
vendas['mês'] = vendas['data'].dt.to_period('M')
vendas['mês'].value_counts().sort_index()
```

```
Out[816...] mês
2023-01      252
2023-02      223
2023-03      253
2023-04      238
2023-05      614
2023-06      582
2023-07      270
2023-08      674
2023-09      260
2023-10      576
2023-11      913
2023-12     1314
Freq: M, Name: count, dtype: int64
```

```
In [817...] # Verificar os horários com mais compras
vendas['horario'].value_counts().sort_index()
```

```
Out[817...] horario
00:00        9
00:01        8
00:02       10
00:03       16
00:04       14
..
23:55       18
23:56       26
23:57       16
23:58       13
23:59       19
Name: count, Length: 659, dtype: int64
```

```
In [818...] # Calcular o faturamento por venda (preço unitário * quantidade + frete)
vendas['faturamento'] = vendas['preco_unitario'] * vendas['quantidade'] + vendas
```

```
In [819...] # Estatísticas do faturamento
vendas['faturamento'].describe()
```

```
Out[819...] count      6169.000000
mean       2752.311154
std        6590.867737
min         2.900000
25%        109.760000
50%        272.320000
75%        780.890000
max       43025.950000
Name: faturamento, dtype: float64
```

```
In [820...] # Faturamento médio por categoria
vendas.groupby('categoria')['faturamento'].mean().sort_values(ascending=False)
```

```
Out[820...] categoria
Eletrônicos                19010.163328
Eletrodomésticos          6839.987654
Ferramentas e construção   558.126751
Beleza e cuidados pessoais 415.866026
Casa e limpeza             317.998204
Livros                     260.399468
Alimentos e bebidas        161.647625
Papellaria e escritório    85.734338
Name: faturamento, dtype: float64
```

```
In [821...] # Faturamento total por método de pagamento
vendas.groupby('metodo_pagamento')['faturamento'].sum().sort_values(ascending=False)
```

```
Out[821...] metodo_pagamento
Cartão de Crédito      7769428.91
PIX                    6260073.08
Boleto                 1667076.30
Cartão de Débito       1282429.22
Name: faturamento, dtype: float64
```

```
In [822...] # Total de produtos vendidos por categoria
vendas.groupby('categoria')['quantidade'].sum().sort_values(ascending=False)
```

```
Out[822...] categoria
Alimentos e bebidas      4484
Beleza e cuidados pessoais 3183
Casa e limpeza           3100
Eletrodomésticos        1973
Eletrônicos              1763
Papellaria e escritório  1729
Ferramentas e construção 1396
Livros                   909
Name: quantidade, dtype: int64
```

1.4 - Tipos de gráficos

Já exploramos um pouco das duas bases do projeto e obtivemos alguns insights interessantes que podem ser importantes na construção de nossas visuais.

Agora, vamos tentar simplificar a nossa análise consolidando os dados em uma única tabela para a partir daí debruçar nas possibilidades de visuais combinando os dados. Será que o GPT pode nos ajudar aqui?

Prompt:

Vamos agora unir as duas bases do nosso projeto em uma só utilizando como chave a coluna "ID_compra". Além disso, vamos organizar as colunas na seguinte sequência:

- ID_compra
- data
- horario
- categoria

- preco_unitario
- quantidade
- frete
- metodo_pagamento
- ID_cliente
- idade
- sexo_biologico
- cidade
- uf
- regioao
- cashback
- avaliacao_compra

Construa um código para executar esta etapa usando o Python e a biblioteca Pandas.

```
In [823... # Unindo as bases de dados com base na coluna "ID compra"
df = pd.merge(vendas, clientes, on='ID_compra')
```

```
In [824... print(df.columns.tolist())
```

```
['ID_compra', 'data', 'horario', 'categoria', 'preco_unitario', 'quantidade', 'frete', 'metodo_pagamento', 'mês', 'faturamento', 'ID_cliente', 'cidade', 'uf', 'regiao', 'idade', 'sexo_biologico', 'cashback', 'avaliacao_compra']
```

```
In [825... # Reordenando as colunas
colunas_organizadas = [
    'ID_compra', 'data', 'horario', 'categoria', 'preco_unitario',
    'quantidade', 'frete', 'metodo_pagamento', 'ID_cliente', 'idade',
    'sexo_biologico', 'cidade', 'uf', 'regiao', 'cashback', 'avaliacao_compra'
]
```

```
In [826... df = df[colunas_organizadas]
df
```

Out[826...

	ID_compra	data	horario	categoria	preco_unitario	quantidade	frete
0	6	2023-01-03	20:40	Eletrônicos	4457.9	4	713.26
1	5	2023-01-03	21:48	Ferramentas e construção	115.9	1	0.00
2	9	2023-01-03	17:08	Beleza e cuidados pessoais	68.9	2	13.78
3	4	2023-01-03	14:29	Ferramentas e construção	80.9	5	40.45
4	8	2023-01-03	18:02	Beleza e cuidados pessoais	165.0	4	0.00
...
6164	6141	2023-12-31	17:49	Eletrodomésticos	1509.9	5	377.48
6165	6148	2023-12-31	23:02	Papelaria e escritório	14.0	2	3.36
6166	6123	2023-12-31	21:25	Alimentos e bebidas	77.9	1	7.79
6167	6147	2023-12-31	21:38	Livros	142.9	4	17.15
6168	6122	2023-12-31	22:11	Beleza e cuidados pessoais	89.9	5	13.48

6169 rows × 16 columns



In [827...

```
print(df.columns.tolist())
```

```
['ID_compra', 'data', 'horario', 'categoria', 'preco_unitario', 'quantidade', 'frete', 'metodo_pagamento', 'ID_cliente', 'idade', 'sexo_biologico', 'cidade', 'uf', 'regiao', 'cashback', 'avaliacao_compra']
```

Agora vamos pedir ao ChatGPT ideias de visuais que podemos construir com esses dados.

Prompt:

Agora com o nosso "df" já construído por meio das duas bases do projeto, vamos iniciar um novo processo. Queremos iniciar a etapa da geração dos visuais que possam auxiliar na análise do faturamento ao longo do ano, perfil do cliente e outros dados que possam ser relevantes à análise e construção do nosso storytelling.

As colunas de "df" são:

- ID_compra: número único atribuído a compra realizada, no formato int
- data: data da compra, no formato datetime
- horario: horário da compra no formato object
- categoria: categoria do produto, no formato object
- preco_unitario: preço de uma unidade do produto, no formato float
- quantidade: quantidade vendida do produto, no formato int
- frete: frete a ser pago no total da compra, no formato float
- metodo_pagamento: método de pagamento da compra, no formato object
- ID_cliente: número único atribuído ao cliente, no formato int
- idade: idade do cliente, no formato int
- sexo_biologico: sexo biológico do cliente, no formato object
- cidade: cidade do cliente, no formato object
- uf: unidade federativa do cliente, no formato object
- regioao: região do cliente, no formato object
- cashback: se o cliente participa ou não do programa de cashback da loja, no formato object
- avaliacao_compra: avaliação do cliente pela compra, no formato int

O objetivo é abstrair e buscar relações para construir visuais interessantes para a análise dos dados dos clientes e vendas da Zoop.

Com base nas informações fornecidas, quais tipos de visualizações seriam úteis para atingir o objetivo de analisar os dados?

Sugestões do ChatGPT:

Com base nas informações fornecidas sobre as colunas do DataFrame "df", aqui estão algumas sugestões de visualizações que podem ser úteis para analisar os dados dos clientes e vendas da Zoop:

1. Gráfico de Linha Temporal para Faturamento ao Longo do Ano:

- Utilize a coluna "data" para criar um gráfico de linha mostrando o faturamento total ao longo do ano. Isso pode ser mensal, semanal ou diário, dependendo da granularidade dos seus dados.

2. Gráfico de Barras para Categorias de Produtos Vendidos:

- Utilize a coluna "categoria" para criar um gráfico de barras mostrando a distribuição das vendas por categoria de produtos. Isso pode ajudar a identificar as categorias mais populares.

3. Gráfico de Barras ou Pizza para Métodos de Pagamento:

- Utilize a coluna "metodo_pagamento" para criar um gráfico de barras ou pizza mostrando a distribuição de métodos de pagamento utilizados pelos clientes.

4. Gráfico de Dispersão para Relacionamento entre Idade e Avaliação de Compra:

- Utilize as colunas "idade" e "avaliacao_compra" para criar um gráfico de dispersão e verificar se há alguma correlação entre a idade dos clientes e a

avaliação da compra.

5. Gráfico de Barras para Participação no Programa de Cashback:

- Utilize a coluna "cashback" para criar um gráfico de barras mostrando a participação dos clientes no programa de cashback.

6. Mapa de Calor para Correlações:

- Crie um mapa de calor usando as correlações entre diferentes variáveis. Isso pode ajudar a identificar padrões e relações entre as diferentes características dos clientes e as transações.

7. Gráfico de Boxplot para Preço Unitário por Categoria de Produto:

- Utilize as colunas "preco_unitario" e "categoria" para criar um gráfico de boxplot e identificar a variabilidade de preços em diferentes categorias de produtos.

8. Gráfico de Barras para Avaliações Médias por Região:

- Utilize as colunas "avaliacao_compra" e "regiao" para criar um gráfico de barras mostrando as avaliações médias das compras em diferentes regiões.

Essas são apenas algumas ideias iniciais, e a escolha das visualizações dependerá dos insights específicos que você busca e dos questionamentos que deseja responder com a análise dos dados.

O ChatGPT trouxe algumas ideias interessantes para criação dos gráficos.

Porém, como já possuímos alguns questionamentos para nos orientar na criação dos visuais, não vamos seguir todas as sugestões aqui apontadas.

Fica como uma sugestão para te auxiliar quando você estiver trabalhando com outras bases e estiver com dificuldade em pensar em quais visuais você poderá construir.

Observação: Lembrando que é muito provável que os gráficos apontados para mim, não sejam necessariamente iguais aos que aparecerá para você, dado a aleatoriedade das respostas de qualquer IA generativa.

Aula 2 - Visualizando os dados

2.1 - Definindo os visuais

Após a etapa de exploração dos dados, vamos ler as questões que foram levantadas na demanda que recebemos. Assim, começaremos a pensar nos visuais que iremos construir com o auxílio da IA.

Questionamentos

Vendas

1. Quais foram os métodos de pagamentos mais utilizados em 2023?
2. Qual foi o faturamento por categoria em 2023?
3. Como foram as vendas ao longo dos meses em 2023?
4. Como esta dividida a venda por trimestre por método de pagamento? Existe um período em que algum tipo de pagamento se sobressai?

Perfil do cliente

5. Qual a proporção da clientela que participa de nosso programa de cashback?
6. Como estão distribuídas as avaliações de produto? Temos bons números gerais?
7. Como está distribuído o nosso público por idade e sexo biológico? Existe um grupo que sobressai?

Inicialmente, vamos fazer a primeira versão dos gráficos, ainda não refinados com os elementos visuais, para mais a frente ir adicionando as diversas técnicas de storytelling.

1º Questionamento: Quais foram os métodos de pagamentos mais utilizados em 2023?

Vamos pedir ao ChatGPT para nos auxiliar na construção desse visual. Primeiro, vamos tratar o dado gerando uma tabela resumida (que também é uma espécie de visual) e em seguida construir o 1º gráfico.

| Prompt:

Temos um DataFrame chamado "df" e queremos construir um visual que apresente quais métodos de pagamentos foram os mais utilizados.

Em "df", temos a coluna "metodo_pagamento" com os 4 tipos de pagamento utilizado na compra.

Inicialmente, queremos construir uma tabela chamada "metodos_de_pagamento" que resuma os dados dessa coluna e, na sequência, desejamos criar uma visualização que apresente a quantidade desses métodos de pagamento presentes na tabela "metodo_pagamento".

Utilize as bibliotecas de visualização de dados do Python para criar essa visualização.

In [828...

```
# Criar a tabela de resumo
metodos_de_pagamento = df['metodo_pagamento'].value_counts().reset_index()
metodos_de_pagamento.columns = ['metodo_pagamento', 'quantidade']
metodos_de_pagamento
```

Out[828...

	metodo_pagamento	quantidade
0	Cartão de Crédito	2851
1	PIX	2244
2	Cartão de Débito	547
3	Boleto	527

In [829...

```
print(df.columns.tolist())
```

```
['ID_compra', 'data', 'horario', 'categoria', 'preco_unitario', 'quantidade', 'fr  
ete', 'metodo_pagamento', 'ID_cliente', 'idade', 'sexo_biologico', 'cidade', 'u  
f', 'regiao', 'cashback', 'avaliacao_compra']
```

In [830...

```
print(metodos_de_pagamento.columns.tolist())
```

```
['metodo_pagamento', 'quantidade']
```

In [831...

```
import matplotlib.pyplot as plt
import seaborn as sns

# Define o estilo
sns.set(style="whitegrid")

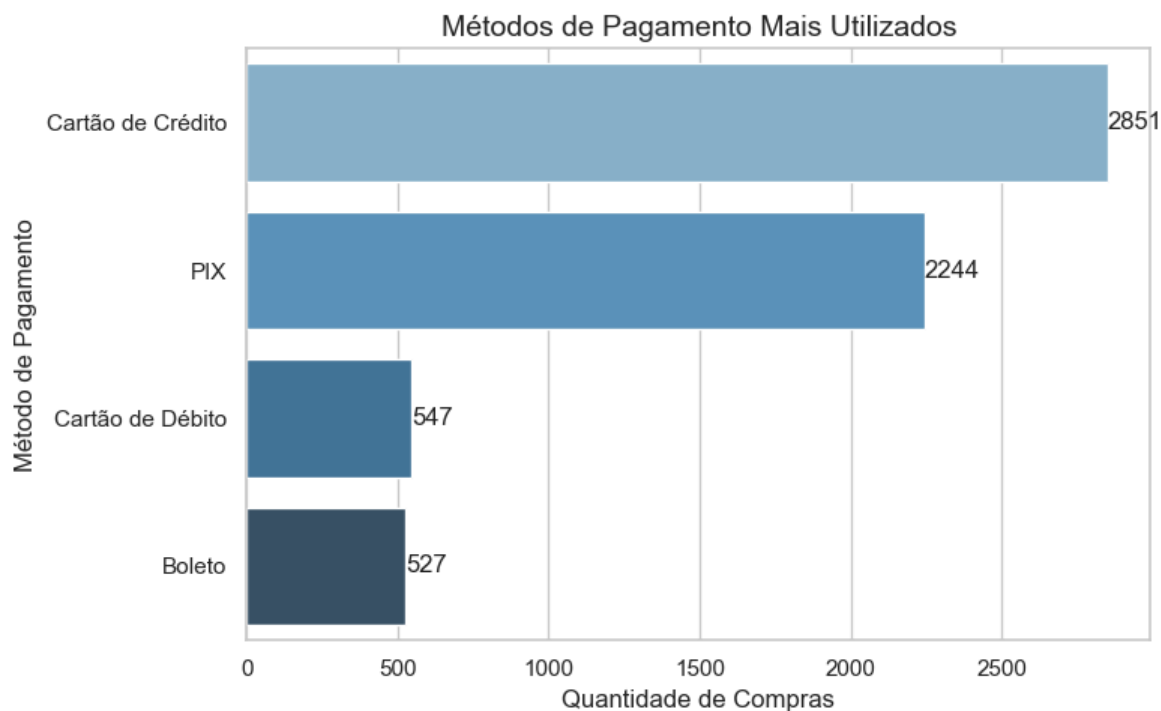
# Tamanho da figura
plt.figure(figsize=(8, 5))

# Cria o gráfico de barras
sns.barplot(
    data=metodos_de_pagamento,
    y='metodo_pagamento',
    x='quantidade',
    hue='metodo_pagamento',      # Adiciona hue igual ao y
    palette='Blues_d',
    legend=False                 # Remove a Legenda redundante
)

# Adiciona os rótulos
plt.title('Métodos de Pagamento Mais Utilizados', fontsize=14)
plt.xlabel('Quantidade de Compras')
plt.ylabel('Método de Pagamento')

# Adiciona os valores em cada barra
for index, row in metodos_de_pagamento.iterrows():
    plt.text(row['quantidade'] + 1, index, str(row['quantidade']), va='center')

plt.tight_layout()
plt.show()
```



Comentários:

Nós conseguimos observar que mesmo sem especificar o tipo de gráfico, pelo modo que apresentamos os dados ao ChatGPT ele conseguiu representar com sucesso um visual que traga as informações desejadas. Um gráfico de colunas realmente é o ideal para representação das quantidades de uma ou mais categorias de dados.

Vamos ajustar aos poucos os visuais para que fiquem adequados para o nosso público e que tenha a identidade visual da empresa.

2.2 - Comparando dados

Agora que já criamos nosso primeiro visual, podemos prosseguir com a construção dos próximos visuais. Aqui, vamos focar em gráficos de comparação para responder a mais dois questionamentos.

2º Questionamento: Qual foi o faturamento por categoria em 2023?

Observação: Para o faturamento, vamos considerar aqui o produto entre o preço unitário e a quantidade somadas ao frete de cada produto.

Prompt:

Temos um DataFrame chamado "df" e queremos construir um visual de gráfico de barras horizontais que apresente o faturamento por categoria da Zoop.

Em "df", temos as colunas:

- categoria: categorias dos produtos
- preco_unitario: preço de uma unidade do produto
- quantidade: quantidade vendida do produto
- frete: frete a ser pago no total da compra.

Desejamos criar uma visualização que apresente o faturamento de cada categoria, em que o faturamento é dado pelo produto entre o preço unitário e a quantidade somadas ao frete.

Utilize as bibliotecas de visualização de dados do Python para criar essa visualização.

faturamento=(preco_unitario×quantidade)+frete

In [832...

```
# 1. Criar coluna de faturamento por linha  
df['faturamento'] = (df['preco_unitario'] * df['quantidade']) + df['frete']  
df
```

Out[832...

	ID_compra	data	horario	categoria	preco_unitario	quantidade	frete
0	6	2023-01-03	20:40	Eletrônicos	4457.9	4	713.26
1	5	2023-01-03	21:48	Ferramentas e construção	115.9	1	0.00
2	9	2023-01-03	17:08	Beleza e cuidados pessoais	68.9	2	13.78
3	4	2023-01-03	14:29	Ferramentas e construção	80.9	5	40.45
4	8	2023-01-03	18:02	Beleza e cuidados pessoais	165.0	4	0.00
...
6164	6141	2023-12-31	17:49	Eletrodomésticos	1509.9	5	377.48
6165	6148	2023-12-31	23:02	Papelaria e escritório	14.0	2	3.36
6166	6123	2023-12-31	21:25	Alimentos e bebidas	77.9	1	7.79
6167	6147	2023-12-31	21:38	Livros	142.9	4	17.15
6168	6122	2023-12-31	22:11	Beleza e cuidados pessoais	89.9	5	13.48

6169 rows × 17 columns



In [833...

```
print(df.columns.tolist())
```

```
['ID_compra', 'data', 'horario', 'categoria', 'preco_unitario', 'quantidade', 'frete', 'metodo_pagamento', 'ID_cliente', 'idade', 'sexo_biologico', 'cidade', 'uf', 'regiao', 'cashback', 'avaliacao_compra', 'faturamento']
```

In [834...

```
# 2. Agrupar por categoria e somar o faturamento
```

```
faturamento_categoria = df.groupby('categoria')['faturamento'].sum().reset_index
```

In [835...

```
# 3. Ordenar do maior para o menor faturamento
```

```
faturamento_categoria = faturamento_categoria.sort_values(by='faturamento', ascending=False)
```

Out[835...

	categoria	faturamento
7	Papelaria e escritório	47239.62
6	Livros	78380.24
0	Alimentos e bebidas	241663.20
5	Ferramentas e construção	264552.08
2	Casa e limpeza	331036.13
1	Beleza e cuidados pessoais	441649.72
3	Eletrrodomésticos	4548591.79
4	Eletrônicos	11025894.73

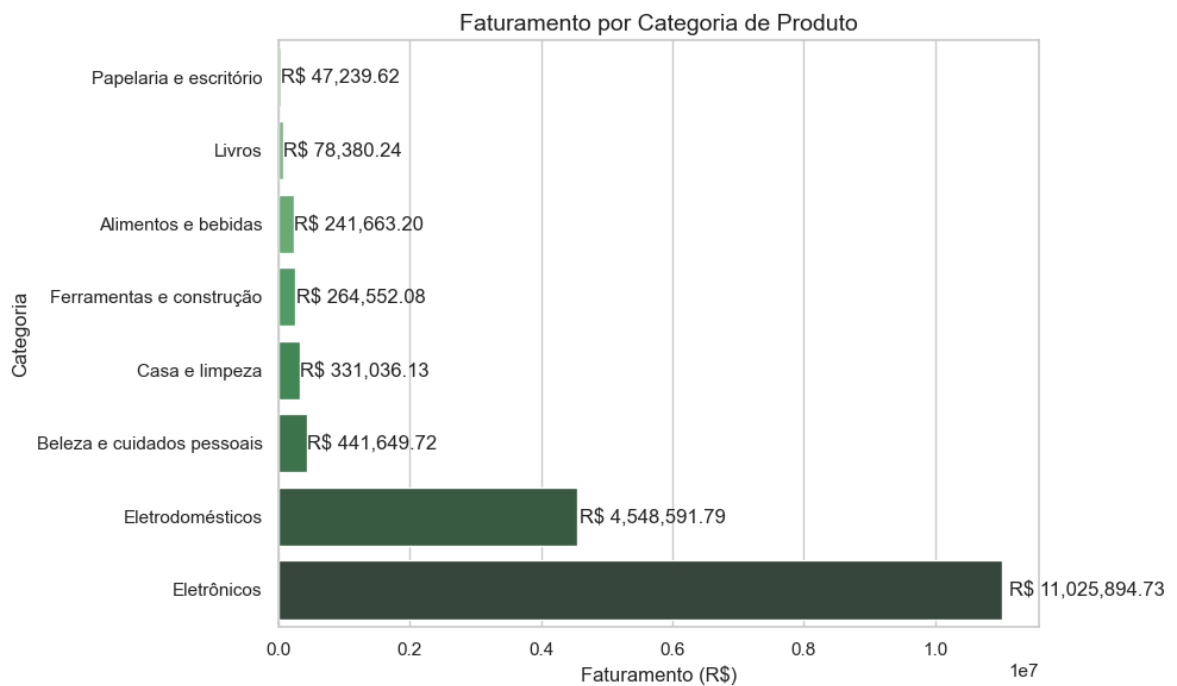
In [836...

```
# 4. Plotar gráfico de barras horizontais
plt.figure(figsize=(10, 6))
ax = sns.barplot(
    data=faturamento_categoria,
    x='faturamento',
    y='categoria',
    hue='categoria',          # Adiciona hue igual ao y
    palette='Greens_d',
    legend=False             # Remove a Legenda redundante
)

# Título e rótulos
plt.title('Faturamento por Categoria de Produto', fontsize=14)
plt.xlabel('Faturamento (R$)')
plt.ylabel('Categoria')

# Inserir rótulos de dados diretamente nas barras com base na altura real
for bar in ax.patches:
    width = bar.get_width()
    y = bar.get_y() + bar.get_height() / 2
    ax.text(
        width + (width * 0.01), # deslocamento proporcional
        y,
        f"R$ {width:,.2f}",
        va='center'
    )

plt.tight_layout()
plt.show()
```



Comentários:

Aqui especificamos o tipo de gráfico, guiando o ChatGPT para criação do nosso visual. Podemos notar que em um gráfico de barras o foco é maior nas categorias, além de ser uma forma mais fluida de ler textos que possam ser extensos demais para um gráfico de coluna.

3º Questionamento: Como foram as vendas ao longo dos meses em 2023?

Observação: É importante especificar como a sua variável tempo está especificada dentro da base para o ChatGPT alcançar resultados melhores na sugestão do visual

Prompt:

Temos um DataFrame chamado "df" e queremos construir um visual de gráfico de linha que apresente as vendas totais mensais da Zoop.

Em "df", temos as colunas:

- data: com as datas das compras no formato datetime (`aaaa-mm-dd`)
- faturamento: dado pelo produto entre o preço unitário e a quantidade somadas ao frete de cada venda.

Desejamos criar uma visualização que apresente as vendas por mês. Primeiro, agrupe os dados por mês e depois crie uma nova coluna chamada "mes" que receba o nome de cada mês, traduzindo os meses do índice por meio do dicionário abaixo. Utilize a coluna "data" para o agrupamento dos dados e construção do gráfico. A coluna "mes" só deve ser utilizada para alterar os rótulos do eixo x.

```
meses = {
    'January': 'Jan',
    'February': 'Fev',
    'March': 'Mar',
    'April': 'Abr',
    'May': 'Mai',
    'June': 'Jun',
    'July': 'Jul',
    'August': 'Ago',
    'September': 'Set',
    'October': 'Out',
    'November': 'Nov',
    'December': 'Dez'
}
```

Utilize as bibliotecas de visualização de dados do Python para criar essa visualização.

```
In [837... # 1. Garantir que a coluna data está no formato datetime
df['data'] = pd.to_datetime(df['data'])
```

```
In [838... # 2. Agrupa por mês e soma o 'faturamento'
vendas_mensais = df.groupby(pd.Grouper(key='data', freq='MS'))['faturamento'].sum()
vendas_mensais
```

```
Out[838...
      data  faturamento
0  2023-01-01    586383.51
1  2023-02-01    445347.92
2  2023-03-01    572803.02
3  2023-04-01    554309.04
4  2023-05-01    1608193.84
5  2023-06-01    1543645.78
6  2023-07-01     810660.53
7  2023-08-01    2038069.76
8  2023-09-01     821239.89
9  2023-10-01    1578166.95
10 2023-11-01    2696705.50
11 2023-12-01    3723481.77
```

```
In [839... # 3. Dicionário de tradução dos meses
meses = {
    'January': 'Jan',
    'February': 'Fev',
    'March': 'Mar',
    'April': 'Abr',
    'May': 'Mai',
    'June': 'Jun',
    'July': 'Jul',
```



```

    'August': 'Ago',
    'September': 'Set',
    'October': 'Out',
    'November': 'Nov',
    'December': 'Dez'
}

```

In [840... *# 4. Cria uma nova coluna 'mes' com os nomes dos meses traduzidos*

```

vendas_mensais['mes'] = vendas_mensais['data'].dt.month_name().map(meses)
vendas_mensais

```

Out[840...

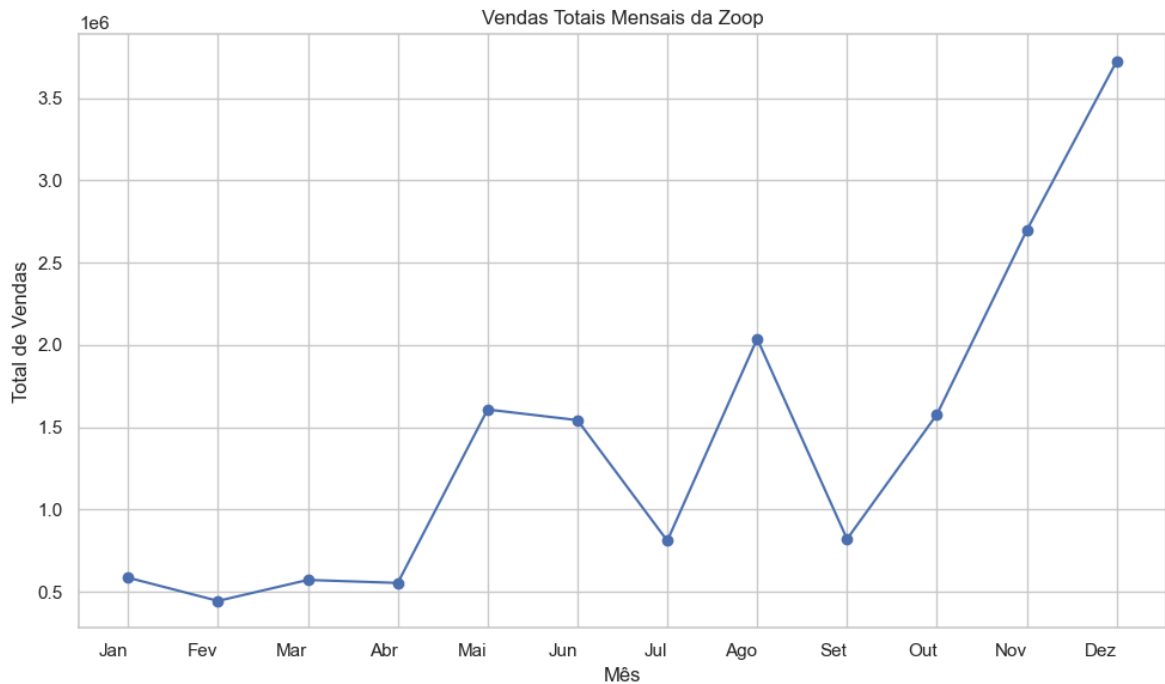
	data	faturamento	mes
0	2023-01-01	586383.51	Jan
1	2023-02-01	445347.92	Fev
2	2023-03-01	572803.02	Mar
3	2023-04-01	554309.04	Abr
4	2023-05-01	1608193.84	Mai
5	2023-06-01	1543645.78	Jun
6	2023-07-01	810660.53	Jul
7	2023-08-01	2038069.76	Ago
8	2023-09-01	821239.89	Set
9	2023-10-01	1578166.95	Out
10	2023-11-01	2696705.50	Nov
11	2023-12-01	3723481.77	Dez

In [841... *# 5. Plotar gráfico de Linha*

```

plt.figure(figsize=(10, 6)) # Ajusta o tamanho da figura
plt.plot(vendas_mensais['mes'], vendas_mensais['faturamento'], marker='o', lines
plt.title('Vendas Totais Mensais da Zoop')
plt.xlabel('Mês')
plt.ylabel('Total de Vendas')
plt.grid(True) # Adiciona uma grade
plt.xticks(rotation=0, ha='right') # Rotaciona os rótulos do eixo x para melhor
plt.tight_layout() # Ajusta o layout para evitar que os rótulos sejam cortados

```



Comentários:

Quando possuímos uma quantidade de períodos maior para os nossos dados, vale a pena recorrer a gráficos de linha(s) que é ideal para séries temporais. Como nossos dados são mensais, utilizar um gráfico de colunas, por exemplo, reduziria a percepção de uma sequência dos dados. Linhas dão uma maior sensação de continuidade.

Sinta-se livre para elaborar algumas hipóteses dos momentos de alta e baixa das compras.

2.3 - Compondo dados dinâmicos

Construímos alguns visuais interessantes para comparação dos dados, trazendo os faturamentos da Zoop. Agora, vamos partir para uma nova categoria de visuais: composição.

Vamos uma composição dinâmica dos dados. Para isso, vamos adicionar mais variáveis à análise, trazendo 3 diferentes variáveis sendo uma delas categórica.

4º Questionamento: Como esta dividida a venda por trimestre por método de pagamento? Existe um período em que algum tipo de pagamento se sobressai?

Prompt:

Temos um DataFrame chamado "df" e queremos construir um visual de gráfico de barras empilhadas que apresente as vendas por trimestre em relação ao método de pagamento da Zoop.

Em "df", temos as colunas:

- data: com as datas das compras no formato datetime (aaaa-mm-dd)
- faturamento: com os valores das vendas
- metodo_pagamento: com o método de pagamento escolhido na compra

Desejamos criar uma visualização que apresente as vendas por trimestre, agrupando os métodos de pagamento de cada trimestre correspondente a cada compra.

Utilize as bibliotecas de visualização de dados do Python para criar essa visualização.

In [842...

```
# Garante que a coluna 'data' seja do tipo datetime
df['data'] = pd.to_datetime(df['data'])
```

In [843...

```
# Extrai o trimestre e o converte para string
df['trimestre'] = df['data'].dt.to_period('Q').astype(str)
df
```

Out[843...

	ID_compra	data	horario	categoria	preco_unitario	quantidade	frete
0	6	2023-01-03	20:40	Eletrônicos	4457.9	4	713.26
1	5	2023-01-03	21:48	Ferramentas e construção	115.9	1	0.00
2	9	2023-01-03	17:08	Beleza e cuidados pessoais	68.9	2	13.78
3	4	2023-01-03	14:29	Ferramentas e construção	80.9	5	40.45
4	8	2023-01-03	18:02	Beleza e cuidados pessoais	165.0	4	0.00
...
6164	6141	2023-12-31	17:49	Eletrodomésticos	1509.9	5	377.48
6165	6148	2023-12-31	23:02	Papelaria e escritório	14.0	2	3.36
6166	6123	2023-12-31	21:25	Alimentos e bebidas	77.9	1	7.79
6167	6147	2023-12-31	21:38	Livros	142.9	4	17.15
6168	6122	2023-12-31	22:11	Beleza e cuidados pessoais	89.9	5	13.48

6169 rows × 8 columns



```
In [844... print(df.columns.tolist())
```

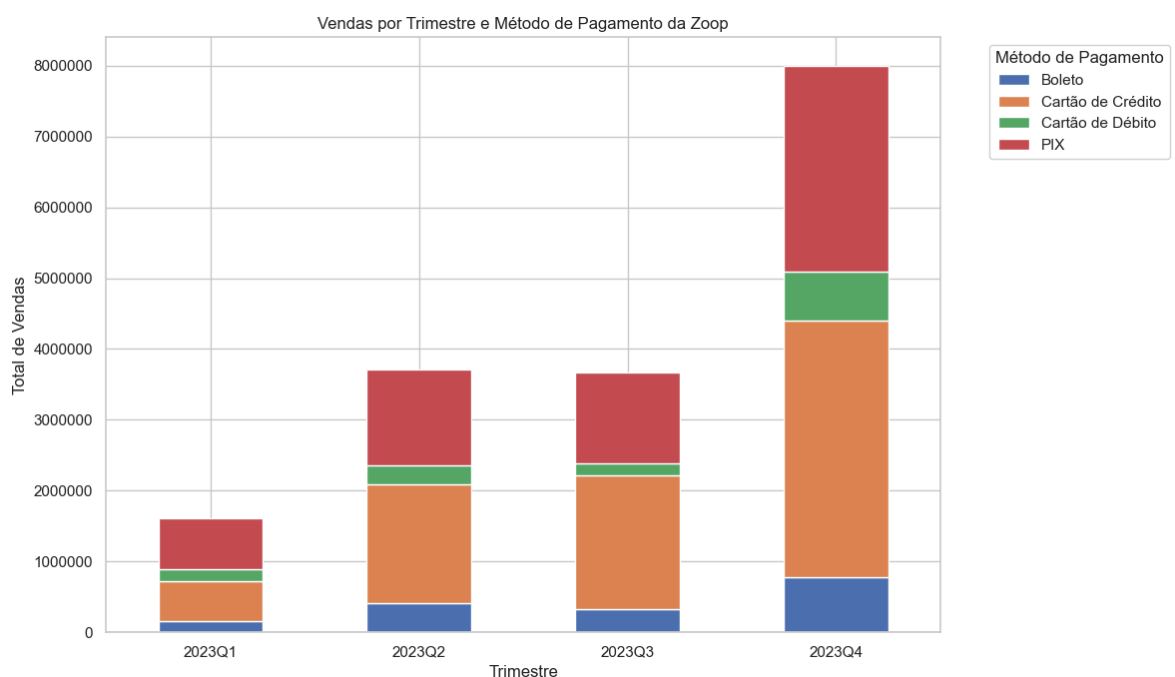
```
['ID_compra', 'data', 'horario', 'categoria', 'preco_unitario', 'quantidade', 'fr  
ete', 'metodo_pagamento', 'ID_cliente', 'idade', 'sexo_biologico', 'cidade', 'u  
f', 'regiao', 'cashback', 'avaliacao_compra', 'faturamento', 'trimestre']
```

```
In [845... df.columns = df.columns.str.strip()
```

```
In [846... # Agrupa por trimestre e método de pagamento, e então soma o faturamento.  
df_tri = df.groupby(['trimestre', 'metodo_pagamento'])['faturamento'].sum().unsta  
df_tri
```

```
Out[846... metodo_pagamento    Boleto  Cartão de Crédito  Cartão de Débito    PIX  
  
trimestre  
2023Q1  159392.78          567801.62          160984.48  716355.57  
2023Q2  408176.22          1675152.84          271050.18  1351769.42  
2023Q3  323562.78          1899375.64          166394.70  1280637.06  
2023Q4  775944.52          3627098.81          683999.86  2911311.03
```

```
In [847... # Cria o gráfico de barras empilhadas  
plt.figure(figsize=(12, 7)) # Define o tamanho da figura  
df_grouped.plot(kind='bar', stacked=True, ax=plt.gca()) # ax=plt.gca() garante q  
  
plt.title('Vendas por Trimestre e Método de Pagamento da Zoop')  
plt.xlabel('Trimestre')  
plt.ylabel('Total de Vendas')  
plt.xticks(rotation=0, ha='center') # Rotaciona os rótulos do eixo x para melhor  
plt.ticklabel_format(style='plain', axis='y') # Evita notação científica no eixo  
# Move a legenda para fora da área do gráfico para não sobrepor os dados  
plt.legend(title='Método de Pagamento', bbox_to_anchor=(1.05, 1), loc='upper lef  
plt.tight_layout() # Ajusta o layout para evitar que os rótulos e a legenda seja
```



Comentários:

O gráfico de colunas empilhadas adiciona à comparação entre períodos ou categorias a comparação também entre os dados de cada período/categoria. É como se colocássemos uma "lupa" em cada coluna e visualizássemos a composição de cada uma delas de acordo com a variável desejada (aqui, as formas de pagamento).

Sinta-se livre para elaborar algumas hipóteses das vendas pelo trimestre combinados ao tipo de pagamento e quais insights podemos retirar dessa base.

Aula 3 - Gerando mais visuais

Agora, vamos nos concentrar nos questionamentos voltados ao perfil dos(as) clientes, como avaliam as compras dentro da Zoop e se participam de nosso programa de cashback.

3.1 - Compendo dados estáticos

Vamos inicialmente trazer um exemplo de como podemos utilizar a comparação de dados estáticos, ou seja, um recorte do tempo ou das variáveis que possuímos.

5º Questionamento: Qual a proporção da clientela que participa de nosso programa de cashback?

Prompt:

Temos um DataFrame chamado "df" e queremos construir um visual que apresente a proporção dos(as) usuários(as) que estão aderindo o programa de cashback da Zoop.

Em "df", temos a coluna "cashback" que dispõe de tais dados.

Utilize as bibliotecas de visualização de dados do Python para criar essa visualização.

```
In [848... # Calcula a contagem de cada valor na coluna 'cashback' e normaliza para obter a
proporcoes_cashback = clientes['cashback'].value_counts(normalize=True)

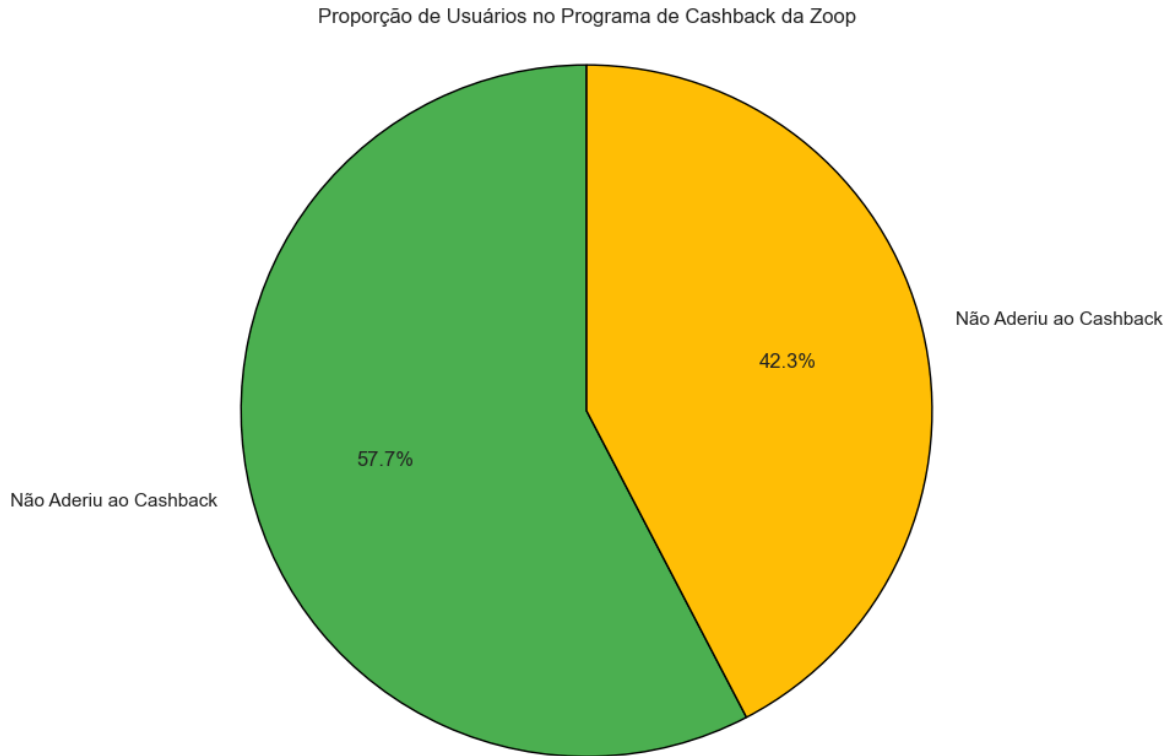
In [849... labels = ['Aderiu ao Cashback' if val == True else 'Não Aderiu ao Cashback' for

In [850... colors = ['#4CAF50', '#FFC107'] # Verde para "Aderiu", Amarelo para "Não Aderiu"

In [851... # Cria o gráfico de pizza
plt.figure(figsize=(8, 8)) # Define o tamanho da figura para que o gráfico fique
plt.pie(
    proporcoes_cashback, # Os valores a serem plotados
    labels=labels,       # Os rótulos para cada fatia
    autopct='%1.1f%%',   # Formato para exibir as porcentagens nas fatias (ex: 7
    startangle=90,        # Ângulo inicial da primeira fatia (começa no topo)
    colors=colors,        # As cores definidas
```

```
wedgeprops={'edgecolor': 'black'} # Adiciona uma borda preta em cada fatia p
)
plt.title('Proporção de Usuários no Programa de Cashback da Zoop') # Título do g
plt.axis('equal') # Garante que o gráfico de pizza seja desenhado como um círcul
```

```
Out[851... (np.float64(-1.0999984793891773),
np.float64(1.0999911023562172),
np.float64(-1.099999856932745),
np.float64(1.099999993187273))
```



Prompt:

Você conseguiria inverter as cores e alterar o gráfico anterior para um do tipo rosca?

Utilize as bibliotecas de visualização de dados do Python para criar essa visualização.

```
In [852... # Amarelo para "Aderiu", Verde para "Não Aderiu"
colors = [ '#4CAF50', '#FFC107' ]

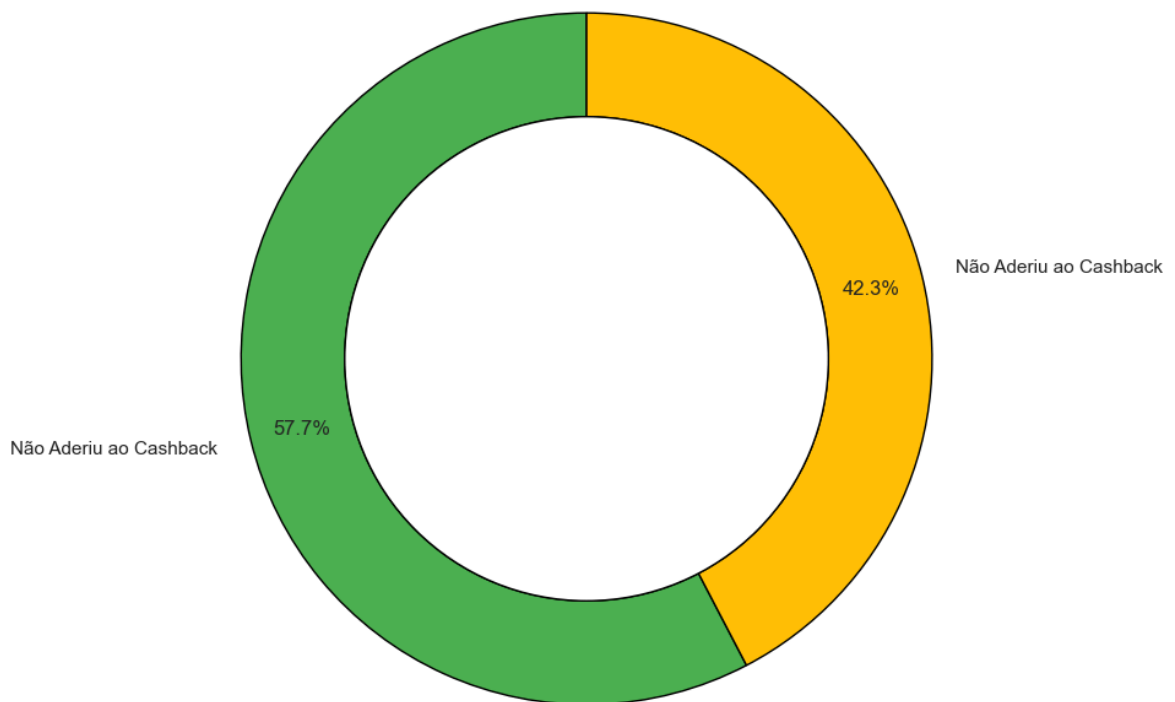
# Cria o gráfico de rosca (donut chart)
plt.figure(figsize=(8, 8)) # Define o tamanho da figura
plt.pie(
    proporcoes_cashback, # Os valores a serem plotados
    labels=labels,       # Os rótulos para cada fatia
    autopct='%1.1f%%',   # Formato para exibir as porcentagens nas fatias (ex: 7
    startangle=90,        # Ângulo inicial da primeira fatia (começa no topo)
    colors=colors,        # As cores definidas
    pctdistance=0.85,     # Distância das porcentagens do centro
    wedgeprops=dict(width=0.3, edgecolor='black') # Propriedades para criar a "r
)

# Desenha um círculo branco no centro para criar o efeito de rosca
centre_circle = plt.Circle((0,0), 0.70, fc='white', edgecolor='black')
fig = plt.gcf()
```

```
fig.gca().add_artist(centre_circle)
```

```
plt.title('Proporção de Usuários no Programa de Cashback da Zoop (Gráfico de Rosca)')  
plt.axis('equal') # Garante que o gráfico seja desenhado como um círculo perfeito  
plt.show()
```

Proporção de Usuários no Programa de Cashback da Zoop (Gráfico de Rosca)



Comentários:

Olha só! Mais uma vez a IA nos apresentou uma boa noção de qual gráfico aplicar para esse ponto, mas você conseguiu notar que "ela" não faz ideia dos dados que possuímos?

Precisamos estar atentos(as) a isso, pois passamos apenas os nomes das colunas e que dados existiam lá. Ela não consegue saber o que é um valor de caráter positivo ou negativo, por exemplo. Por isso, o "não" estava em azul e o "sim" em vermelho. No prompt seguinte, demos uma orientação para inverter as cores (ainda sem falar o real valor dos dados) e pedimos uma alteração do visual.

Que tal você experimentar fazer outros pedidos e ver como o ChatGPT se comporta?

Falando do gráfico, gráficos de setores são ideais para poucas categorias de dados, se possível dados dicotômicos (2 categorias, como sim e não, gato e cachorro...). Eles representam a proporção dos dados de maneira estática.

3.2 - Distribuindo dados

Com o auxílio da IA, vamos gerar mais alguns gráficos que serão importantes na construção do storytelling do relatório da Zoop como requisitado. Partindo agora, da

distribuição de dados, vamos analisar o perfil dos clientes e como eles avaliam as compras na Zoop

6º Questionamento: Como estão distribuídas as avaliações de produto? Temos bons números gerais?

Vamos pedir ao ChatGPT para nos auxiliar na construção desse visual. Vamos ver se esta IA consegue representar tais dados com um visual adequado.

Prompt:

Temos um DataFrame chamado "df" e queremos construir um visual que apresente a distribuição das avaliações do público em relação as compras feitas na loja online.

Em "df", temos a coluna "avaliacao_compra" com as notas dadas por cada usuário em relação a compra que fizeram que variam de 0 a 10 em números inteiros.

Desejamos criar uma visualização que apresente a distribuição desses dados.

Utilize as bibliotecas de visualização de dados do Python para criar essa visualização.

```
In [853... # Calcula a frequência de cada avaliação
distribuicao_avaliacoes = clientes['avaliacao_compra'].value_counts().sort_index

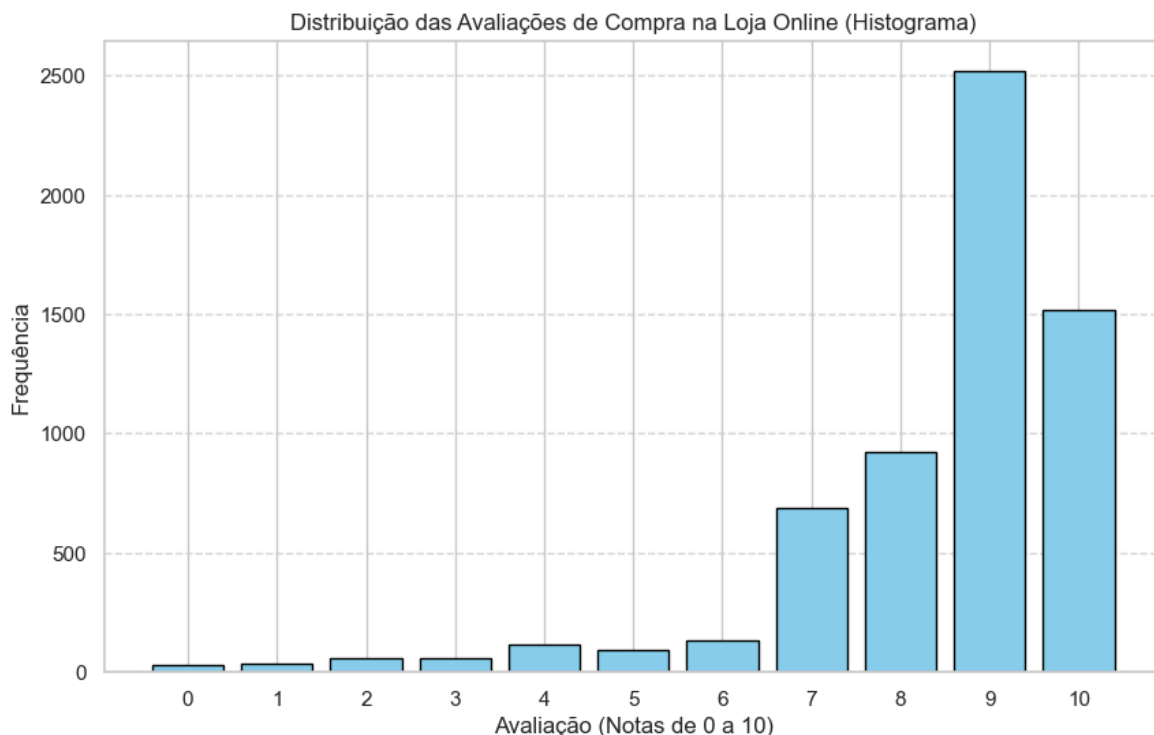
In [854... import numpy as np

In [855... # Define os 'bins' (barras) para o histograma
min_rating = clientes['avaliacao_compra'].min()
max_rating = clientes['avaliacao_compra'].max()
bins = np.arange(min_rating - 0.5, max_rating + 1.5, 1)

In [856... # Cria o histograma
plt.figure(figsize=(10, 6)) # Define o tamanho da figura para melhor visualizaçã
plt.hist(
    clientes['avaliacao_compra'], # Os dados a serem plotados
    bins=bins,                  # Os 'bins' definidos
    color='skyblue',            # Cor das barras
    edgecolor='black',          # Cor da borda das barras
    rwidth=0.8                  # Largura relativa das barras (para criar um pequeno
)

plt.title('Distribuição das Avaliações de Compra na Loja Online (Histograma)') #
plt.xlabel('Avaliação (Notas de 0 a 10)') # Rótulo do eixo X
plt.ylabel('Frequência') # Rótulo do eixo Y (representa o número de ocorrências)
plt.xticks(np.arange(0, 11)) # Define os 'ticks' do eixo X para cada número inte
plt.grid(axis='y', linestyle='--', alpha=0.7) # Adiciona linhas de grade horizon

plt.show()
```

Comentários:

Mais uma vez não precisamos especificar diretamente o tipo de gráfico, pois ao citar distribuição de valores numéricos de uma simples coluna o ChatGPT conseguiu interpretar e trazer o código para um histograma de coluna. Este gráfico é ideal para apresentar a distribuição de uma amostra de dados ou população. Cada coluna aqui representa a frequência de repetição de um valor.

Quer experimentar fazendo o mesmo apenas com as idades?

3.3 - Distribuindo por característica

Vamos agora para a criação do nosso último visual. Subindo mais um degrau na distribuição de dados, iremos adicionar uma variável categórica ao problema. Vamos ver como a IA se comporta aqui?

7º Questionamento: Como está distribuído o nosso público por idade e sexo biológico? Existe um grupo que sobressai?

Prompt:

Temos um DataFrame chamado "df" e queremos construir um visual que apresente a distribuição do nosso público por idade e sexo biológico.

Em "df", temos as colunas:

- idade: idade da pessoa que realizou a compra

- `sexo_biologico`: sexo biológico da pessoa que realizou a compra, com dados categóricos

Desejamos criar uma visualização que apresente a distribuição desses dados.

Utilize as bibliotecas de visualização de dados do Python para criar essa visualização.

```
In [857... # Garante que a coluna 'sexo_biologico' seja tratada como categórica pelo pandas
clientes['sexo_biologico'] = clientes['sexo_biologico'].astype('category')

In [858... # Prepara os dados separando por sexo biológico
idade_feminino = clientes[clientes['sexo_biologico'] == 'Feminino']['idade']
idade_masculino = clientes[clientes['sexo_biologico'] == 'Masculino']['idade']

In [859... # Define os 'bins' (intervalos) para o histograma.
# Isso ajuda a agrupar as idades em faixas e garantir que as barras do histogram
min_idade_geral = clientes['idade'].min()
max_idade_geral = clientes['idade'].max()
bins = np.arange(min_idade_geral, max_idade_geral + 2, 2) # Bins de 2 em 2 anos

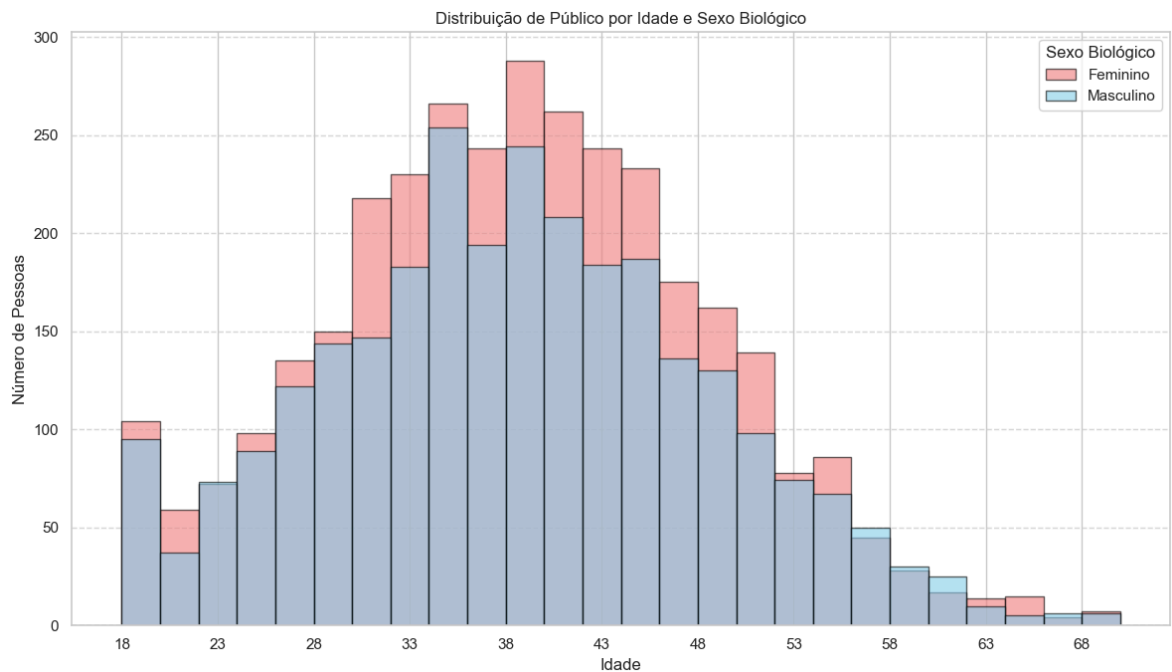
In [860... # Cria o histograma
plt.figure(figsize=(12, 7)) # Define o tamanho da figura para melhor visualizaçã

# Plota o histograma para o sexo feminino
plt.hist(
    idade_feminino,
    bins=bins,
    alpha=0.6,          # Transparência para permitir ver a sobreposição
    label='Feminino',
    color='lightcoral', # Cor para o sexo feminino
    edgecolor='black'   # Borda das barras
)

# Plota o histograma para o sexo masculino
plt.hist(
    idade_masculino,
    bins=bins,
    alpha=0.6,          # Transparência para permitir ver a sobreposição
    label='Masculino',
    color='skyblue',    # Cor para o sexo masculino
    edgecolor='black'   # Borda das barras
)

plt.title('Distribuição de Público por Idade e Sexo Biológico') # Título do gráf
plt.xlabel('Idade') # Rótulo do eixo X
plt.ylabel('Número de Pessoas') # Rótulo do eixo Y
plt.xticks(np.arange(min_idade_geral, max_idade_geral + 1, 5)) # Define os 'tick
plt.legend(title='Sexo Biológico') # Adiciona a Legenda para identificar as core
plt.grid(axis='y', linestyle='--', alpha=0.7) # Adiciona linhas de grade horizon
plt.tight_layout() # Ajusta o layout para evitar que elementos sejam cortados

plt.show()
```



O gráfico gerado é pouco prático, pois compila os dados do sexo biológico "Masculino" e "Feminino" em uma mesma barra! Precisamos trazer mais uma instrução ao GPT para escolher um visual mais adequado.

Prompt:

Acho que podemos melhorar esse gráfico. Que tal mudar para um boxplot?

In [861...

```
# Cria o boxplot
fig, ax = plt.subplots(figsize=(10, 6)) # Cria uma figura e um conjunto de eixos

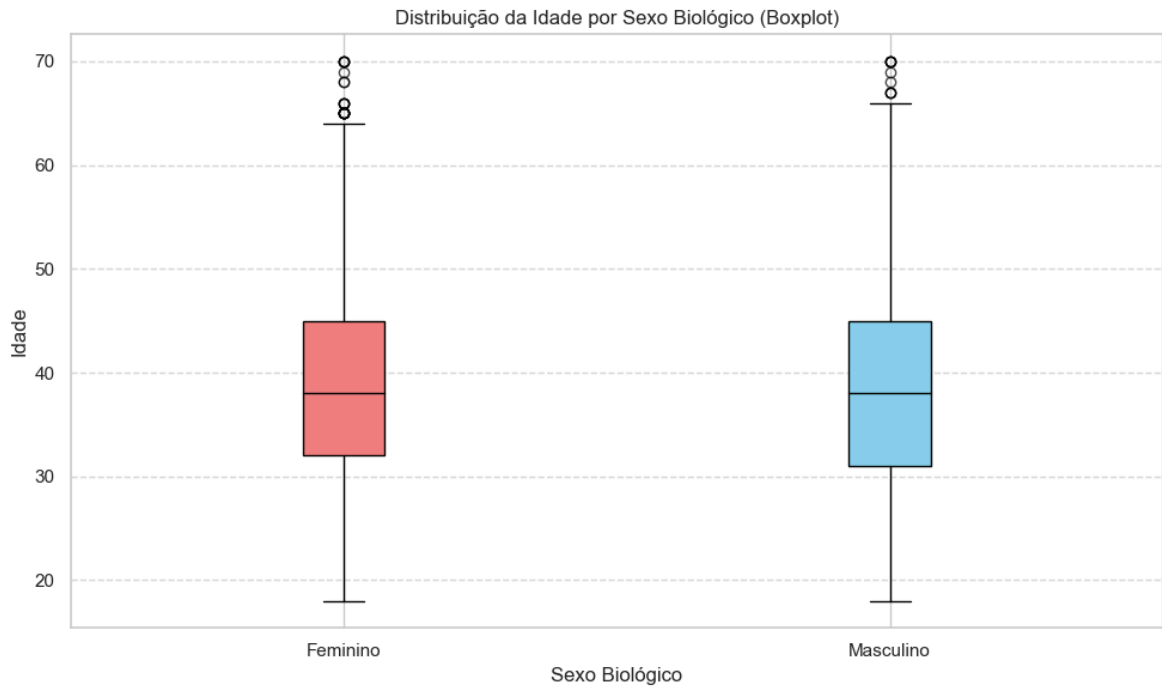
# Dados para o boxplot devem ser uma lista de arrays/series
boxplot_data = [idade_feminino, idade_masculino]
labels = ['Feminino', 'Masculino'] # Rótulos para cada boxplot

bp = ax.boxplot(
    boxplot_data,
    labels=labels,
    patch_artist=True, # Permite colorir as caixas
    medianprops=dict(color='black'), # Cor da linha da mediana
    whiskerprops=dict(color='black'), # Cor dos "bigodes"
    capprops=dict(color='black'), # Cor das "tampas" dos bigodes
    flierprops=dict(markeredgecolor='black', marker='o', markersize=6, alpha=0.7)
)

# Define as cores para cada caixa (box) individualmente
colors = ['lightcoral', 'skyblue'] # Cor para 'Feminino' e 'Masculino'
for patch, color in zip(bp['boxes'], colors):
    patch.set_facecolor(color)

plt.title('Distribuição da Idade por Sexo Biológico (Boxplot)') # Título do gráfico
plt.xlabel('Sexo Biológico') # Rótulo do eixo X
plt.ylabel('Idade') # Rótulo do eixo Y
plt.grid(axis='y', linestyle='--', alpha=0.7) # Adiciona linhas de grade horizontais
plt.tight_layout() # Ajusta o layout para evitar cortes
```

```
plt.show()
```



Comentários:

Dessa vez foi necessário guiar a IA a fim de gerar um visual melhor para nossas análises.

O gráfico aqui representado é um boxplot (diagrama de caixa) que é ideal para representar um conjunto de dados de acordo com 5 medidas de interesse: valor máximo, mínimo, 1º quartil (onde estão situados os primeiros 25% dos dados), 2º quartil ou mediana (onde estão situados 50% dos dados) e 3º quartil (onde estão situados 75% dos dados).

Lembra do `describe()` ? Este gráfico apresenta em um único elemento boa parte dos dados que a função exibe em tabela.

Concluimos a 1ª versão dos 7 visuais que utilizaremos em nosso projeto. Agora, vamos partir para a aplicação das técnicas de storytelling para melhorar os visuais e deixá-los com a identidade visual da Zoop antes da exportação.

Modulo 3 - Gerando mais visuais - aula 8 - Faça como eu fiz: gráfico de árvore (treemap)

Prompt

Temos um DataFrame chamado "df" e queremos construir um Treemap que apresente o faturamento total por região e estado.

Em "df", temos as colunas:

regiao: com as regiões dos(as) clientes. Com os valores de "Centro-Oeste", "Nordeste", "Norte", "Sudeste" e "Sul". uf: com os estados dos(as) clientes. faturamento: com os valores das vendas. Crie um novo DataFrame chamado "vendas_2023" que filtre o df para manter apenas as colunas "regiao", "uf" e "faturamento" e crie um gráfico Treemap a partir desta base.

Para path passe apenas a coluna "uf", para values a coluna "faturamento" e para color a coluna "regiao". Para colorir cada região utilize as seguintes cores:

Centro-Oeste: "#0081a7" Nordeste: "#00afb9" Norte: "#fdfcdc" Sudeste: "#fed9b7" Sul: "#f07167" Ajuste a largura (width = 1400) e altura (height = 600) do Treemap e adicione os valores dentro dos retângulos. Utilize a biblioteca Plotly no Python para criar essa visualização.

In [862...

df

Out[862...

	ID_compra	data	horario	categoria	preco_unitario	quantidade	frete
0	6	2023-01-03	20:40	Eletrônicos	4457.9	4	713.26
1	5	2023-01-03	21:48	Ferramentas e construção	115.9	1	0.00
2	9	2023-01-03	17:08	Beleza e cuidados pessoais	68.9	2	13.78
3	4	2023-01-03	14:29	Ferramentas e construção	80.9	5	40.45
4	8	2023-01-03	18:02	Beleza e cuidados pessoais	165.0	4	0.00
...
6164	6141	2023-12-31	17:49	Eletrodomésticos	1509.9	5	377.48
6165	6148	2023-12-31	23:02	Papelaria e escritório	14.0	2	3.36
6166	6123	2023-12-31	21:25	Alimentos e bebidas	77.9	1	7.79
6167	6147	2023-12-31	21:38	Livros	142.9	4	17.15
6168	6122	2023-12-31	22:11	Beleza e cuidados pessoais	89.9	5	13.48

6169 rows × 18 columns



In [863...

metodos_de_pagamento

Out[863...

	metodo_pagamento	quantidade
0	Cartão de Crédito	2851
1	PIX	2244
2	Cartão de Débito	547
3	Boleto	527

In [864...

```
import pandas as pd
import plotly.express as px

# Supondo que você já tenha o DataFrame df
# Filtrar as colunas desejadas
vendas_2023 = df[['regiao', 'uf', 'faturamento']]

# Criar o gráfico Treemap
fig = px.treemap(vendas_2023, path=['regiao', 'uf'], values='faturamento', color_discrete_map={'Centro-Oeste': '#0081a7',
                                                    'Nordeste': '#00afb9',
                                                    'Norte': '#fdfcdc',
                                                    'Sudeste': '#fed9b7',
                                                    'Sul': '#f07167'})

# Atualizando o layout para adicionar título
fig.update_layout(width=1400, height=600, title='Faturamento Total por Região e')

# Adicionar os valores dentro dos retângulos
fig.update_traces(textinfo='label+value')

# Exibir o gráfico
fig.show()
```

Aula 4 - Técnicas de storytelling

4.1 - Definindo as cores do projeto

Agora, vamos explorar a importância das cores na visualização de dados, discutir como as pessoas percebem e interpretam as informações visuais, e abordar a diferença entre a capacidade da inteligência artificial e a visão humana no que diz respeito à escolha de cores e acessibilidade.

Paleta de cores da Zoop:



Paleta do projeto

Vermelho

VERMELHO_1 VERMELHO_2 VERMELHO_3

#e23155 #cc2c4e #b32742



Azul

AZUL_1 AZUL_2 AZUL_3

#203f75 #1c3867 #19325b



Cinza

BRANCO CINZA_1 CINZA_2 CINZA_3 CINZA_4 CINZA_5

#ffffff #ebebeb #d9d9d9 #cccccc #555655 #231f20



Aqua

AQUA_1 AQUA_2 AQUA_3

#addcd4 #9fccc5 #96bfb9



```
# Adicionando as cores como variáveis do projeto
VERMELHO_1, VERMELHO_2, VERMELHO_3 = "#e23155", "#cc2c4e", "#b32742"
AZUL_1, AZUL_2, AZUL_3 = "#203f75", "#1c3867", "#19325b"
BRANCO, CINZA_1, CINZA_2, CINZA_3, CINZA_4, CINZA_5 = "#ffffff", "#ebebeb",
AQUA_1, AQUA_2, AQUA_3 = "#addcd4", "#9fccc5", "#96bfb9"
```

Por que definimos nós mesmos as cores do projeto?

1. Percepção humana das cores:

Nosso cérebro interpreta as cores de maneira única, associando emoções e significados a cada tonalidade. É importante na escolha das cores para representar dados considerar a percepção humana para garantir que as informações sejam transmitidas de maneira clara e eficaz. Além de seguir a identidade visual da Zoop, ou seja, deixar aquela sensação de que aquele visual é da Zoop.

2. Acessibilidade:

Certas combinações de cores podem ser difíceis de serem percebidas por pessoas com deficiências visuais. Nossa responsabilidade como desenvolvedores é garantir que nossas visualizações sejam inclusivas. Vamos seguir aqui algumas diretrizes e recomendações para criar gráficos acessíveis.

3. IA vs. escolha de cores:

Enquanto a IA pode realizar tarefas complexas, como análise de dados e reconhecimento de padrões, ela ainda não possui o discernimento humano quando se trata de escolher cores significativas e garantir a acessibilidade. É aqui que nossa intervenção criativa e consideração humana entram em cena.

4. Experiência visual:

As pessoas conseguem entender as informações transmitidas pelos gráficos? Existe alguma "briga" entre as cores que prejudica a clareza da mensagem? Vamos explorar técnicas para melhorar a experiência visual, como contraste adequado, pela escolha cuidadosa das cores da paleta e destaque de elementos-chave.

Vamos agora mergulhar na prática e incorporar esses conceitos ao nosso projeto?

4.2 - Adicionando rótulos e labels

Já criamos anteriormente os 7 visuais e definimos as cores do projeto. Agora, vamos escolher apenas um dos visuais para compreender como podemos utilizar a IA para

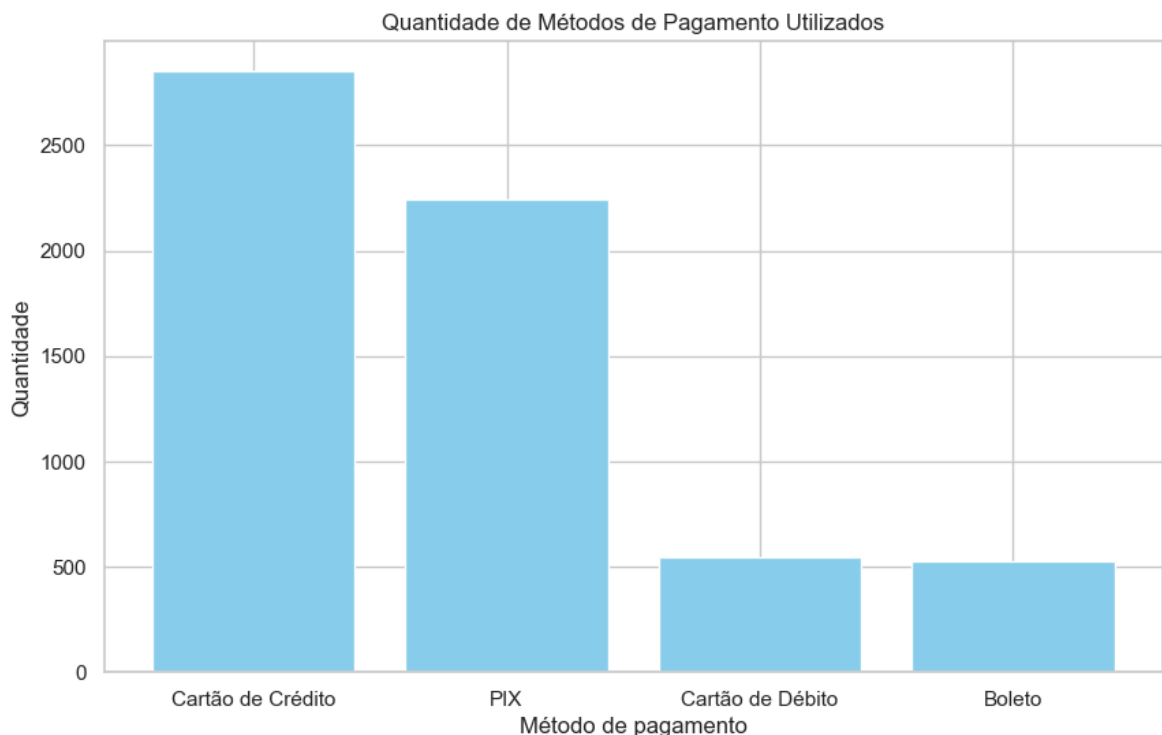
adicionar as cores ideais, os rótulos dos dados, títulos e labels dos eixos.

Vamos então pegar o código daquele primeiro gráfico (**Métodos de pagamento utilizados**) e explorar essas alterações por meio da utilização da IA

In [866...

```
# Visualização
plt.figure(figsize=(10, 6))
plt.bar(metodos_de_pagamento['metodo_pagamento'], metodos_de_pagamento['quantida
plt.title('Quantidade de Métodos de Pagamento Utilizados')
plt.xlabel('Método de pagamento')
plt.ylabel('Quantidade')
plt.xticks(rotation=0, ha='center')

# Exibir o gráfico
plt.show()
```



Prompt:

Agora chegou a hora de refinarmos as nossas visualizações. Vamos começar com o 1º gráfico que fizemos, aquele com os métodos de pagamento?

O nosso gráfico precisa seguir a identidade visual da Zoop, com o fundo da figura e do plot na cor CINZA_1, as colunas na cor VERMELHO_1 e o tamanho da figura deve ser de (10, 6).

O título da figura é "Métodos de pagamentos mais utilizados em 2023" com tamanho 18 e sua cor CINZA_5.

Os rótulos do eixo y (yticks) devem ser suprimidos assim como o label do eixo y (ylabel) e os rótulos e label do eixo X (xticks e xlabel) devem ter tamanho 12 com a cor AZUL_1. As bordas do topo, lado direito e esquerdo do gráfico devem ser suprimidas.

Além disso, é necessário exibir o total de cada método de pagamento acima de cada coluna, com o texto também na cor AZUL_1 e tamanho 12.

Utilize as bibliotecas de visualização de dados do Python para criar essa visualização. Lembre-se que as variáveis das cores que estamos utilizando possui valores de cores já em hexadecimal.

In [867...

```
# Identidade visual Zoop
CINZA_1 = "#E6E6E6"
VERMELHO_1 = "#FF4D4F"
CINZA_5 = "#555555"
AZUL_1 = "#1890FF"

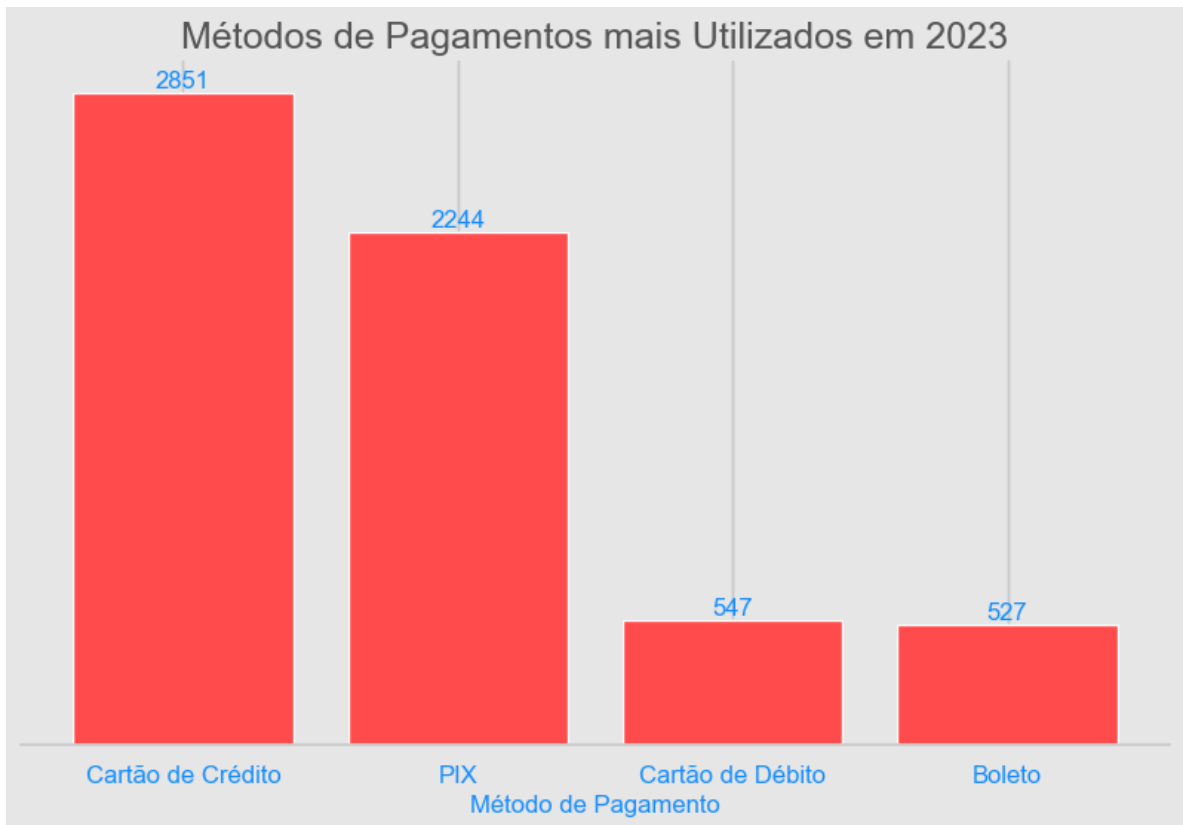
# Agrupar por método de pagamento e contar o total de ocorrências
metodos_de_pagamento = df['metodo_pagamento'].value_counts()

# Configurações do gráfico
plt.figure(figsize=(10, 6), facecolor=CINZA_1)
ax = plt.axes()
ax.set_facecolor(CINZA_1)
plt.bar(metodos_de_pagamento.index, metodos_de_pagamento, color=VERMELHO_1)

# Adicionar texto com o total acima de cada barra
for i, v in enumerate(metodos_de_pagamento):
    plt.text(i, v + 0.1, str(v), ha='center', va='bottom', color=AZUL_1, fontsize=12)

# Configurações adicionais
plt.title('Métodos de Pagamentos mais Utilizados em 2023', fontsize=18, color=CINZA_5)
plt.xticks(fontsize=12, color=AZUL_1)
plt.yticks([])
plt.xlabel('Método de Pagamento', fontsize=12, color=AZUL_1)
ax.spines[['top', 'right', 'left']].set_visible(False)

plt.show()
```



Comentários:

Uau! Parece outro gráfico não é mesmo?

Retiramos muitos elementos distratores e adicionamos apenas a informação que nos interessava. Além disso mudamos o visual de um estado genérico para um formato que casa com a identidade da Zoop, deixando nosso gráfico pronto para ser utilizado, por exemplo, no relatório das vendas.

Mais a frente traremos essas ações para os outros visuais. O nosso foco aqui é entender as técnicas de storytelling e como aliar elas a IA pode ser bastante útil.

4.3 - Adicionando textos

Já refinamos um dos gráficos na etapa anterior. Agora, vamos trazer esse gráfico, adicionando mais um elemento que pode ser importante às análises: textos informativos. Vamos utilizar a IA novamente para esse processo.

Prompt:

O gráfico ficou muito bom!

Agora quero adicionar um texto dentro da área do plot, vamos lá?

Para isso, some os valores das duas primeiras colunas, divida pela soma de todas as colunas e passe para o formato de porcentagem armazenando na variável "percentual"

Agora, na posição x = 2 e y = 2500, adicione o seguinte texto com tamanho 10 e cor AZUL_3: `f"$\\bf{percentual:.2f}$% dos clientes utilizam Cartão de Crédito ou PIX \npara pagamento. Seria interessante recorrer a parcerias\ncom bancos para a construção de um Zoop Pay a fim\nde fidelizar a nossa clientela."`.

```
In [868... metodos_de_pagamento = df['metodo_pagamento'].value_counts()
```

```
In [869... # Calcular percentual
total_clientes = metodos_de_pagamento.sum()
percentual = (metodos_de_pagamento.iloc[0] + metodos_de_pagamento.iloc[1]) / tot
```

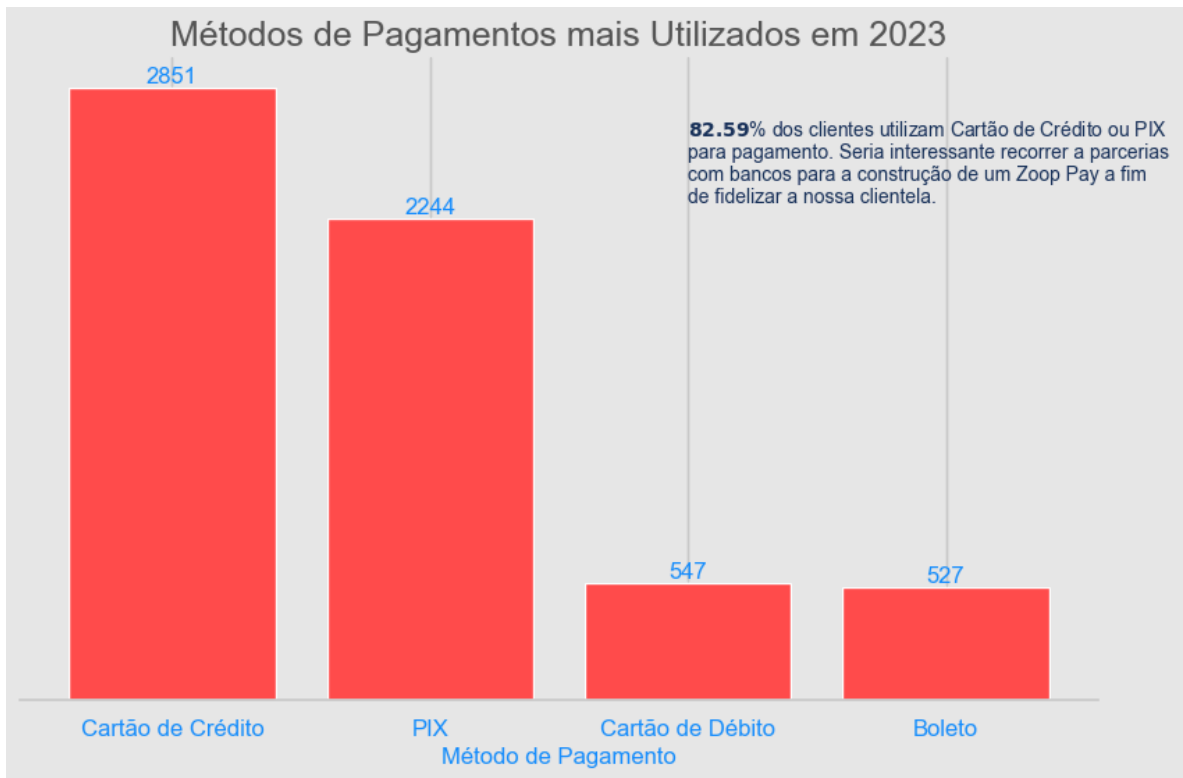
```
In [870... # Configurações do gráfico
plt.figure(figsize=(10, 6), facecolor=CINZA_1)
ax = plt.axes()
ax.set_facecolor(CINZA_1)
plt.bar(metodos_de_pagamento.index, metodos_de_pagamento, color=VERMELHO_1)

# Adicionar texto com o total acima de cada barra
for i, v in enumerate(metodos_de_pagamento):
    plt.text(i, v + 0.1, str(v), ha='center', va='bottom', color=AZUL_1, fontsize=10)

# Adicionar texto dentro da área do plot
plt.text(2, 2500, f"$\\bf{percentual:.2f}$% dos clientes utilizam Cartão de Crédito ou PIX \npara pagamento. Seria interessante recorrer a parcerias\ncom bancos para a construção de um Zoop Pay a fim\nde fidelizar a nossa clientela.",
        ha='left', va='center', color=AZUL_3, fontsize=10)

# Configurações adicionais
plt.title('Métodos de Pagamentos mais Utilizados em 2023', fontsize=18, color=CINZA_1)
plt.xticks(fontsize=12, color=AZUL_1)
plt.yticks([])
plt.xlabel('Método de Pagamento', fontsize=12, color=AZUL_1)
ax.spines[['top', 'right', 'left']].set_visible(False)

plt.show()
```



Comentários:

Aproveitamos a área útil para trazer um *insight* que tivemos ao descrever os dados dos métodos de pagamento. Isso já aumenta e muito a relevância do dado representado e fornece mais informação a quem lê.

4.4 - Adicionando elementos visuais

Bem bacana o uso dos textos informativos, não é mesmo? E se agora nós adicionarmos outros elementos visuais a um de nossos gráficos?

Vamos escolher agora o gráfico de linha (**Vendas totais mensais da Zoop**) e aplicar mais uma técnica a ele por meio da utilização da IA .

```
In [871... # 1. Garantir que a coluna data está no formato datetime
df['data'] = pd.to_datetime(df['data'])
# 2. Agrupa por mês e soma o 'faturamento'
vendas_mensais = df.groupby(pd.Grouper(key='data', freq='MS'))['faturamento'].sum()
# 3. Dicionário de tradução dos meses
meses = {
    'January': 'Jan',
    'February': 'Fev',
    'March': 'Mar',
    'April': 'Abr',
    'May': 'Mai',
    'June': 'Jun',
    'July': 'Jul',
    'August': 'Ago',
    'September': 'Set',
    'October': 'Out',
```

```

    'November': 'Nov',
    'December': 'Dez'
}
# 4. Cria uma nova coluna 'mes' com os nomes dos meses traduzidos
vendas_mensais['mes'] = vendas_mensais['data'].dt.month_name().map(meses)

```

In [872...

vendas_mensais

Out[872...

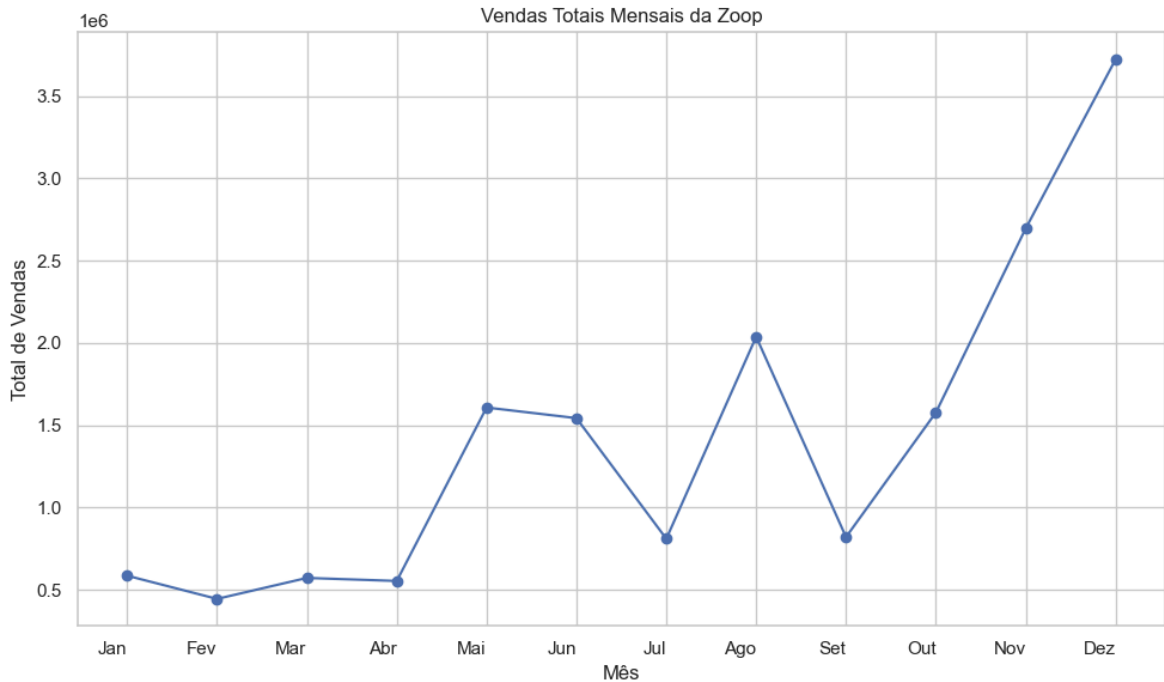
	data	faturamento	mes
0	2023-01-01	586383.51	Jan
1	2023-02-01	445347.92	Fev
2	2023-03-01	572803.02	Mar
3	2023-04-01	554309.04	Abr
4	2023-05-01	1608193.84	Mai
5	2023-06-01	1543645.78	Jun
6	2023-07-01	810660.53	Jul
7	2023-08-01	2038069.76	Ago
8	2023-09-01	821239.89	Set
9	2023-10-01	1578166.95	Out
10	2023-11-01	2696705.50	Nov
11	2023-12-01	3723481.77	Dez

In [873...

```

# 5. Plotar gráfico de linha
plt.figure(figsize=(10, 6)) # Ajusta o tamanho da figura
plt.plot(vendas_mensais['mes'], vendas_mensais['faturamento'], marker='o', lines
plt.title('Vendas Totais Mensais da Zoop')
plt.xlabel('Mês')
plt.ylabel('Total de Vendas')
plt.grid(True) # Adiciona uma grade
plt.xticks(rotation=0, ha='right') # Rotaciona os rótulos do eixo x para melhor
plt.tight_layout() # Ajusta o layout para evitar que os rótulos sejam cortados

```



Prompt:

Vamos explorar mais um dos gráficos que fizemos? Agora vamos utilizar aquele com as Vendas totais mensais da Zoop. A variável que passa os valores mensais é:

```
df['mes'] = df['data'].dt.strftime('%B')
vendas_mensais = df.groupby('mes')
['faturamento'].sum().reindex(meses.keys())
```

Em que `df["data"]` correspondente as datas no formato datetime e `df['faturamento']` os valores das vendas.

E o que gerou o gráfico original foi

```
plt.plot(vendas_mensais.index, vendas_mensais, marker='o',
color='#1890FF', linewidth=2)
```

O nosso gráfico precisa seguir a identidade visual da Zoop, com o fundo da figura e do plot na cor CINZA_1, a linha na cor AZUL_1 e o tamanho da figura deve ser de (12, 6).

O título da figura é "Vendas totais mensais em 2023" com tamanho 18 e sua cor CINZA_5.

Os rótulos do eixo y (yticks) devem estar o formato "R\$ valor M" com duas casas decimais, tamanho 12 e cor AZUL_1.

Os rótulos do eixo X (xticks) devem ter tamanho 12 com a cor AZUL_1. Já os labels do eixo x e y (xlabel e ylabel) devem ser suprimidos. As bordas do topo e lado direito do gráfico devem ser suprimidas.

Utilize as bibliotecas de visualização de dados do Python para criar essa visualização. Lembre-se que as variáveis das cores que estamos utilizando possui valores de cores já

em hexadecimal.

```
In [874... # 1. Garantir que a coluna data está no formato datetime
df['data'] = pd.to_datetime(df['data'])
# 2. Agrupa por mês e soma o 'faturamento'
vendas_mensais = df.groupby(pd.Grouper(key='data', freq='MS'))['faturamento'].sum()
# 3. Dicionário de tradução dos meses
meses = {
    'January': 'Jan',
    'February': 'Fev',
    'March': 'Mar',
    'April': 'Abr',
    'May': 'Mai',
    'June': 'Jun',
    'July': 'Jul',
    'August': 'Ago',
    'September': 'Set',
    'October': 'Out',
    'November': 'Nov',
    'December': 'Dez'
}
# 4. Cria uma nova coluna 'mes' com os nomes dos meses traduzidos
vendas_mensais['mes'] = vendas_mensais['data'].dt.month_name().map(meses)
```

```
In [875... import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

# Cores definidas
CINZA_1 = '#F5F5F5'
CINZA_5 = '#595959'
AZUL_1 = '#1890FF'

# Criação da figura
fig, ax = plt.subplots(figsize=(12, 6), facecolor=CINZA_1)
ax.set_facecolor(CINZA_1)

# Plotagem da linha
ax.plot(vendas_mensais['mes'], vendas_mensais['faturamento'], marker='o', color=CINZA_5)

# Título
ax.set_title('Vendas totais mensais em 2023', fontsize=18, color=CINZA_5)

# Remoção dos labels dos eixos
ax.set_xlabel('')
ax.set_ylabel('')

# Estilização dos rótulos do eixo Y
ax.yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, _: f'R$ {x/1e6:.2f}'))
ax.tick_params(axis='y', labelsize=12, colors=AZUL_1)

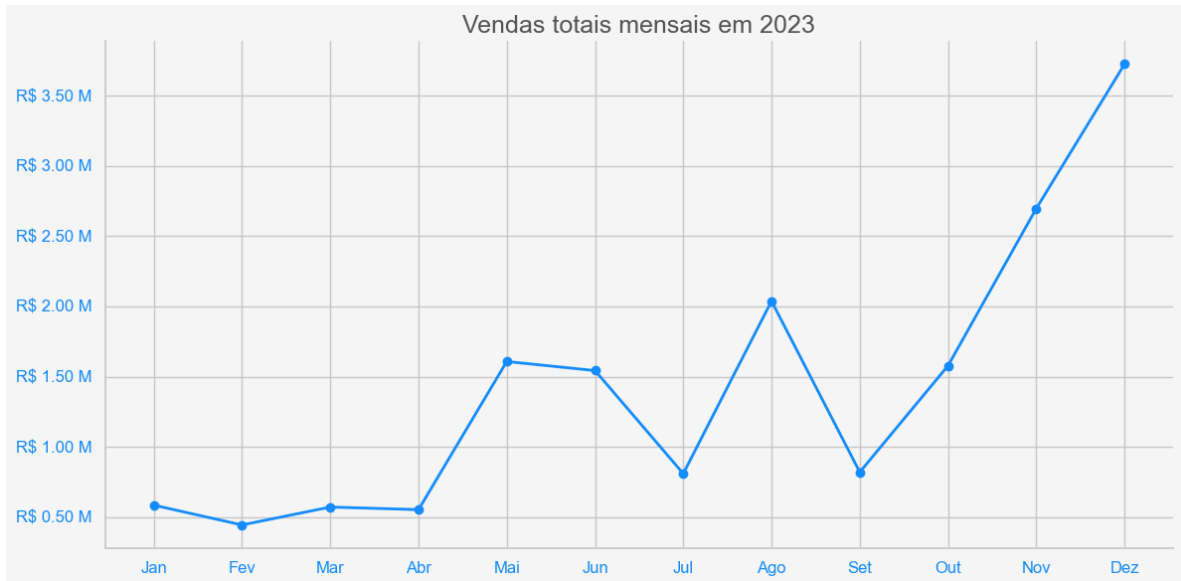
# Estilização dos rótulos do eixo X
ax.tick_params(axis='x', labelsize=12, colors=AZUL_1)

# Remoção das bordas superiores e direita
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)

# Exibição
```



```
plt.tight_layout()
plt.show()
```



Prompt:

Bacana! Agora que o nosso gráfico já está com a identidade visual, vamos adicionar alguns elementos visuais.

Desenhe 4 círculos de raio 200, transparentes com a borda na cor VERMELHO_1 e com o centro nas coordenadas do 5º, 8º, 11º e 12º pontos, respectivamente. No lado esquerdo de cada círculo enumere de 1 a 4.

In [876...

```
# 1. Garantir que a coluna data está no formato datetime
df['data'] = pd.to_datetime(df['data'])
# 2. Agrupa por mês e soma o 'faturamento'
vendas_mensais = df.groupby(pd.Grouper(key='data', freq='MS'))['faturamento'].sum()
# 3. Dicionário de tradução dos meses
meses = {
    'January': 'Jan',
    'February': 'Fev',
    'March': 'Mar',
    'April': 'Abr',
    'May': 'Mai',
    'June': 'Jun',
    'July': 'Jul',
    'August': 'Ago',
    'September': 'Set',
    'October': 'Out',
    'November': 'Nov',
    'December': 'Dez'
}
# 4. Cria uma nova coluna 'mes' com os nomes dos meses traduzidos
vendas_mensais['mes'] = vendas_mensais['data'].dt.month_name().map(meses)
```

In [877...

```
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import matplotlib.patches as patches

# Cores definidas
```

```

CINZA_1 = '#F5F5F5'
CINZA_5 = '#595959'
AZUL_1 = '#1809FF'
VERMELHO_1 = '#FF0000'

# Criação da figura
fig, ax = plt.subplots(figsize=(12, 6), facecolor=CINZA_1)
ax.set_facecolor(CINZA_1)

# Plotagem da linha
ax.plot(vendas_mensais['mes'], vendas_mensais['faturamento'], marker='o', color=

# Título
ax.set_title('Vendas totais mensais em 2023', fontsize=18, color=CINZA_5)

# Remoção dos Labels dos eixos
ax.set_xlabel('')
ax.set_ylabel('')

# Estilização dos rótulos do eixo Y
ax.yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, _: f'R$ {x/1e6:.2f}')
ax.tick_params(axis='y', labelsize=12, colors=AZUL_1)

# Estilização dos rótulos do eixo X
ax.tick_params(axis='x', labelsize=12, colors=AZUL_1)

# Remoção das bordas superiores e direita
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)

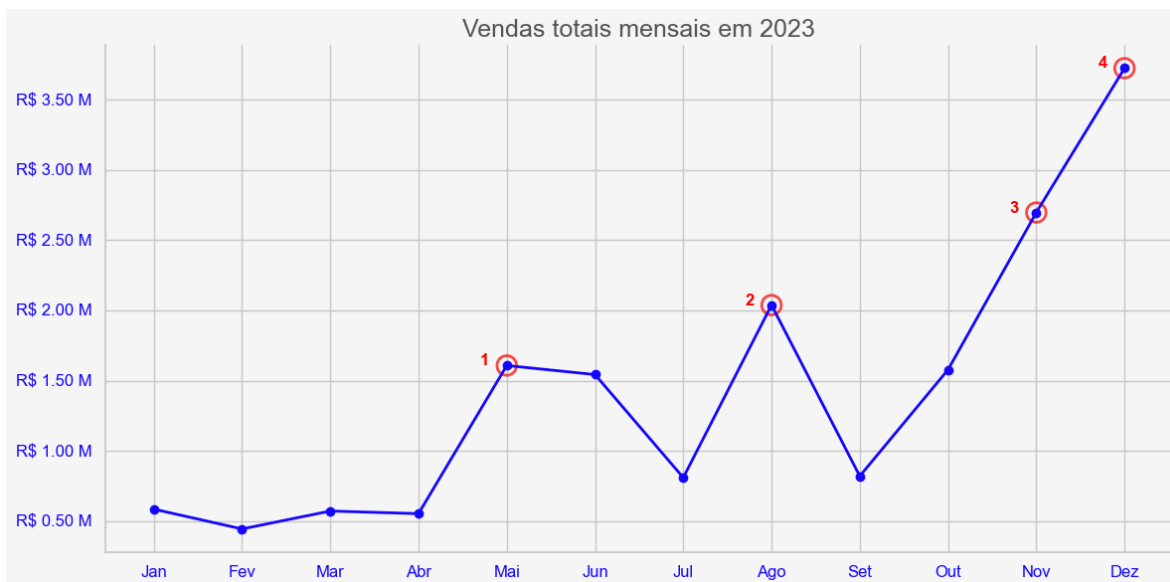
# --- Adiciona círculos ao redor de pontos específicos ---
# Adicionar círculos numerados
for i, point in enumerate([(4, vendas_mensais.iloc[4]),
                           (7, vendas_mensais.iloc[7]),
                           (10, vendas_mensais.iloc[10]),
                           (11, vendas_mensais.iloc[11])], start=1):

    plt.scatter(point[0], point[1]['faturamento'], s=200,
                facecolors='none', edgecolors=VERMELHO_1, linewidths=2, alpha=0.

    plt.text(point[0] - 0.3, point[1]['faturamento'] + 0.1, str(i),
            fontsize=12, color=VERMELHO_1, fontweight='bold')

# Exibição
plt.tight_layout()
plt.show()

```



Comentários:

Adicionamos 4 círculos representando os pontos com maior volume de vendas da Zoop em 2023. Podemos pensar em algumas hipóteses:

1. No mês de Maio, temos uma das datas mais relevantes em vendas: Dia das mães; o que pode justificar o aumento das vendas da Zoop
2. No mês de Agosto, temos 2 situações que podem justificar, a volta às aulas no 2º semestre em alguns estados brasileiros e o Dia dos Pais.
3. No mês de Novembro, fica o destaque a Black Friday um dos principais períodos de alta das vendas, considerado de grande relevância ao Varejo.
4. No mês de Dezembro, temos o Natal e as festividades de Ano Novo. considerados, não só na Zoop, o período de maior vendas do Varejo.

Fica como dica investigar mais os dados e chegar as suas próprias conclusões.

Concluimos assim 2 visuais aplicando as técnicas de storytelling, deixando-os com a identidade visual da Zoop e complementando-os com informações relevantes para a interpretação dos dados. Agora, vamos seguir ajustando os outros 5 visuais para no final partimos para a exportação dos gráficos.

Aula 5 - Concluindo o projeto

5.1 - Ajustando os visuais de comparação

Antes de ajustarmos os últimos 5 visuais de nosso projeto, vamos fazer uma pequena modificação nos 2 primeiros passando-os para função, o que facilitará mais a frente quando realizarmos a exportação dos visuais.

Gráfico 1 - Métodos de pagamentos mais utilizados em 2023:

In [878...

```
def metodos_pag():  
    # Agrupar por método de pagamento e contar o total de ocorrências  
    metodos_de_pagamento = df['metodo_pagamento'].value_counts()  
  
    # Calcular percentual  
    total_clientes = metodos_de_pagamento.sum()  
    percentual = (metodos_de_pagamento[0] + metodos_de_pagamento[1]) / total_clientes  
  
    # Configurações do gráfico  
    plt.figure(figsize=(10, 6), facecolor=CINZA_1)  
    ax = plt.axes()  
    ax.set_facecolor(CINZA_1)  
    plt.bar(metodos_de_pagamento.index, metodos_de_pagamento, color=VERMELHO_1)  
  
    # Adicionar texto com o total acima de cada barra  
    for i, v in enumerate(metodos_de_pagamento):  
        plt.text(i, v + 0.1, str(v), ha='center', va='bottom', color=AZUL_1, fontweight='bold')  
  
    # Adicionar texto dentro da área do plot  
    plt.text(2, 2500, f"$\\bf{{percentual:.2f}}\\% dos clientes utilizam Cartão de Crédito",  
            ha='left', va='center', color=AZUL_3, fontsize=10)  
  
    # Configurações adicionais  
    plt.title('Métodos de Pagamentos mais Utilizados em 2023', fontsize=18, color=VERMELHO_1)  
    plt.xticks(fontsize=12, color=AZUL_1)  
    plt.yticks([])  
    plt.xlabel('Método de Pagamento', fontsize=12, color=AZUL_1)  
    ax.spines[['top', 'right', 'left']].set_visible(False)  
    plt.grid(False)  
  
    return plt
```

In [895...

```
metodos_pag()
```

Out[895...

```
<module 'matplotlib.pyplot' from 'C:\\Users\\joaoc\\AppData\\Roaming\\Python\\Python313\\site-packages\\matplotlib\\pyplot.py'>
```

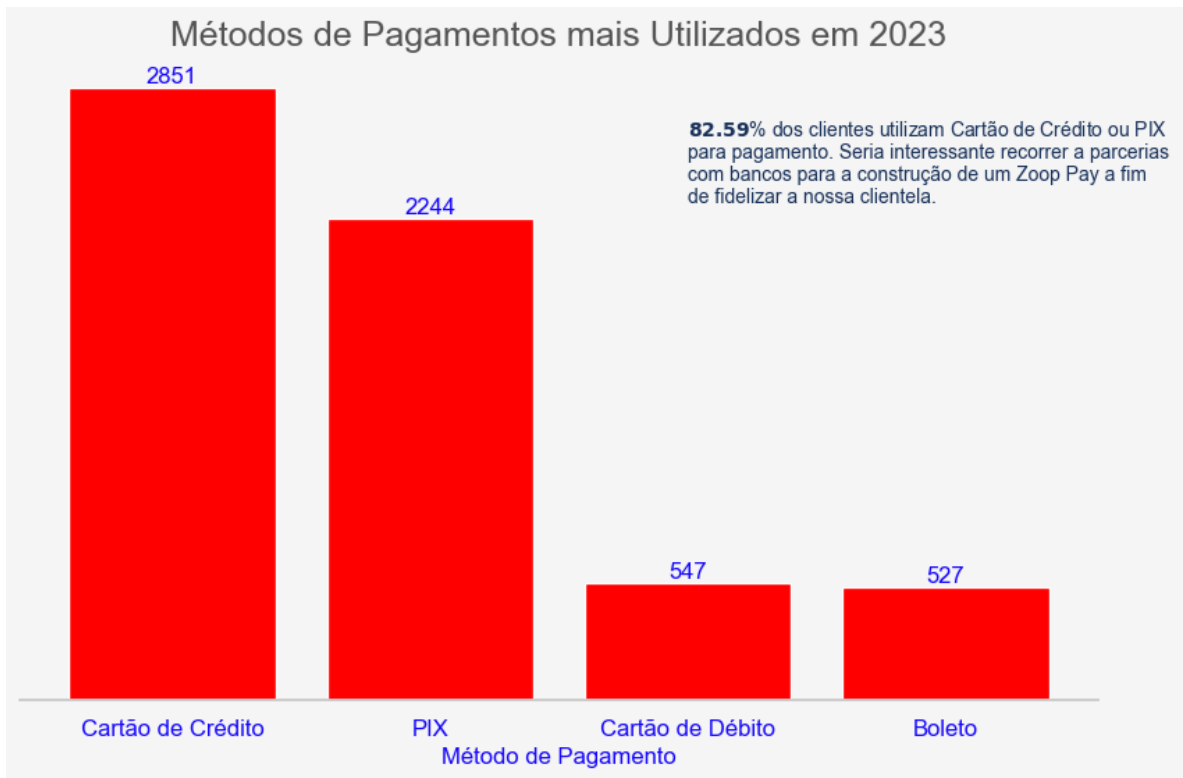


Gráfico 2 - Vendas totais mensais em 2023:

In [897...

```
def vendas_mensais():
    import matplotlib.pyplot as plt
    import matplotlib.ticker as ticker
    import matplotlib.patches as patches

    # 1. Garantir que a coluna data está no formato datetime
    df['data'] = pd.to_datetime(df['data'])
    # 2. Agrupa por mês e soma o 'faturamento'
    vendas_mensais = df.groupby(pd.Grouper(key='data', freq='MS'))['faturamento'].
    # 3. Dicionário de tradução dos meses
    meses = {
        'January': 'Jan',
        'February': 'Fev',
        'March': 'Mar',
        'April': 'Abr',
        'May': 'Mai',
        'June': 'Jun',
        'July': 'Jul',
        'August': 'Ago',
        'September': 'Set',
        'October': 'Out',
        'November': 'Nov',
        'December': 'Dez'
    }

    # 4. Cria uma nova coluna 'mes' com os nomes dos meses traduzidos
    vendas_mensais['mes'] = vendas_mensais['data'].dt.month_name().map(meses)

    # Cores definidas
    CINZA_1 = '#F5F5F5'
    CINZA_5 = '#595959'
    AZUL_1 = '#1809FF'
    VERMELHO_1 = '#FF0000'
```

```

# Criação da figura
fig, ax = plt.subplots(figsize=(12, 6), facecolor=CINZA_1)
ax.set_facecolor(CINZA_1)

# Plotagem da Linha
ax.plot(vendas_mensais['mes'], vendas_mensais['faturamento'], marker='o', color=CINZA_1)

# Título
ax.set_title('Vendas totais mensais em 2023', fontsize=18, color=CINZA_5)

# Remoção dos Labels dos eixos
ax.set_xlabel('')
ax.set_ylabel('')

# Estilização dos rótulos do eixo Y
ax.yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, _: f'R$ {x/1e6:.2f}'))
ax.tick_params(axis='y', labelsize=12, colors=AZUL_1)

# Estilização dos rótulos do eixo X
ax.tick_params(axis='x', labelsize=12, colors=AZUL_1)

# Remoção das bordas superiores e direita
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)

# --- Adiciona círculos ao redor de pontos específicos ---
# Adicionar círculos numerados
for i, point in enumerate([(4, vendas_mensais.iloc[4]),
                           (7, vendas_mensais.iloc[7]),
                           (10, vendas_mensais.iloc[10]),
                           (11, vendas_mensais.iloc[11])], start=1):

    plt.scatter(point[0], point[1]['faturamento'], s=200,
                facecolors='none', edgecolors=VERMELHO_1, linewidths=2, alpha=0.5)

    plt.text(point[0] - 0.3, point[1]['faturamento'] + 0.1, str(i),
            fontsize=12, color=VERMELHO_1, fontweight='bold')

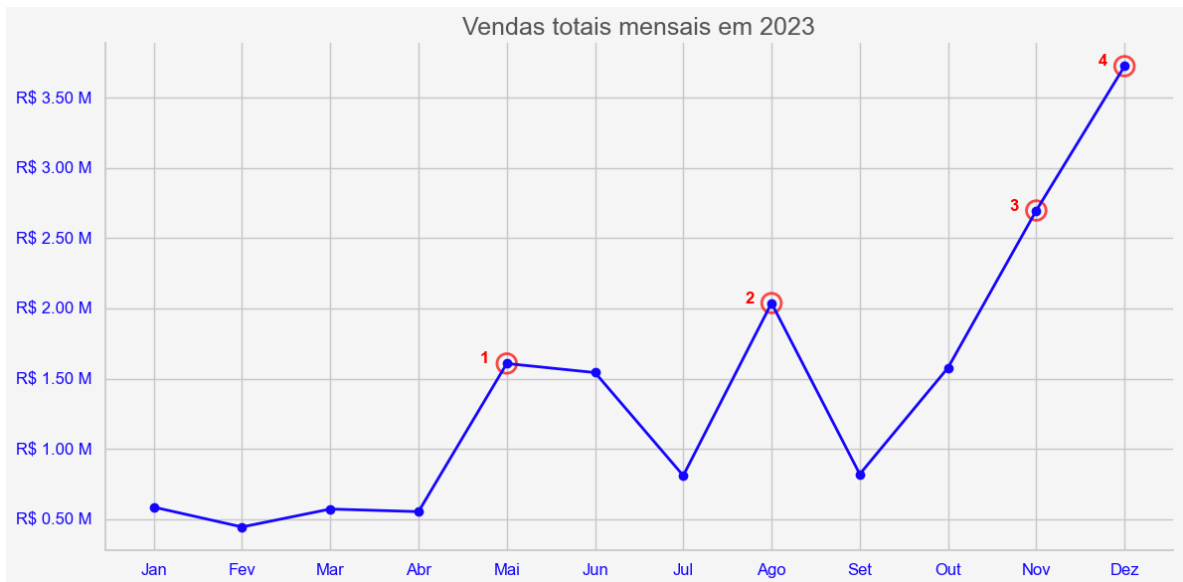
# Exibição
plt.tight_layout()

return plt

```

In [898... vendas_mensais()]

Out[898... <module 'matplotlib.pyplot' from 'C:\\Users\\joaoc\\AppData\\Roaming\\Python\\Python313\\site-packages\\matplotlib\\pyplot.py'>



Agora que salvamos os 2 primeiros gráficos em função, vamos modificar mais um, **Faturamento por categoria de produto em 2023**, utilizando as técnicas de storytelling que já aprendemos.

Gráfico 3 - Faturamento por categoria de produto em 2023:

Vamos dar prosseguimento aos gráficos gerados ao longo das aulas, partindo agora do gráfico de faturamento por categoria de produto aplicando as técnicas de storytelling que aprendemos aqui no curso por meio da utilização da IA

In [880...

```
import matplotlib.pyplot as plt

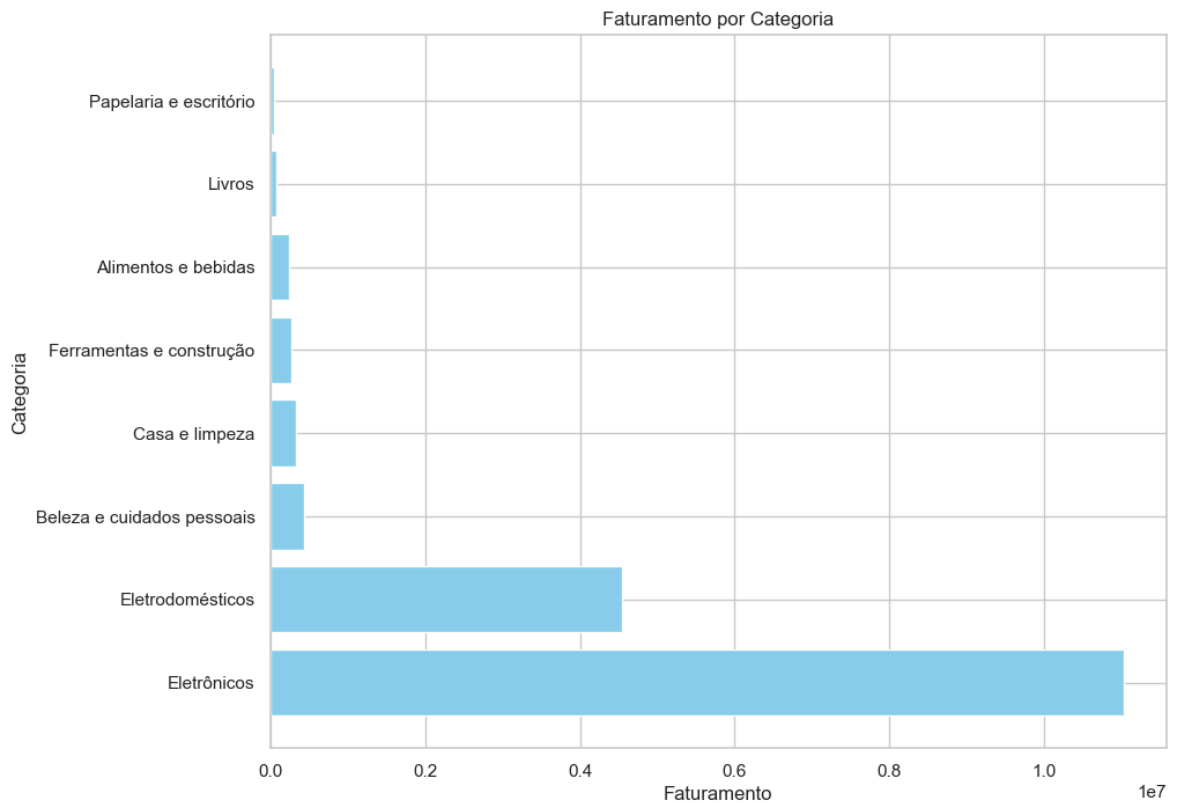
# Calcular o faturamento para cada categoria
df['faturamento'] = (df['preco_unitario'] * df['quantidade']) + df['frete']

# Agrupar por categoria e calcular o faturamento total
faturamento_por_categoria = df.groupby('categoria')['faturamento'].sum().reset_index()

# Ordenar por faturamento
faturamento_por_categoria = faturamento_por_categoria.sort_values(by='faturamento', ascending=False)

# Visualização
plt.figure(figsize=(10, 8))
plt.barh(faturamento_por_categoria['categoria'], faturamento_por_categoria['faturamento'])
plt.title('Faturamento por Categoria')
plt.xlabel('Faturamento')
plt.ylabel('Categoria')

plt.show()
```



Prompt:

Vamos partir para mais um gráfico? Agora, vamos refinar o gráfico de faturamento por categoria. Lembre que esse gráfico foi de barras horizontais.

O nosso gráfico precisa seguir a identidade visual da Zoop, com o fundo da figura e do plot na cor CINZA_1, as colunas na cor AQUA_1 e o tamanho da figura deve ser de (10, 6).

O título da figura é "Faturamento por categoria de produto em 2023" com tamanho 18 e sua cor CINZA_5.

Os rótulos do eixo x (xticks) devem ser suprimidos assim como os labels do eixo x e y (xlabel e ylabel) e os rótulos do eixo y (yticks) devem ter tamanho 12 com a cor AZUL_1. Todas as bordas do gráfico devem ser suprimidas.

Além disso, é necessário exibir o total do faturamento de cada categoria do lado direito de cada barra, com o texto também na cor AZUL_1 e tamanho 12.

Vamos adicionar também um texto dentro da área do plot. Para isso, divida o valor da 1ª barra pela soma de todas as barras e passe para o formato de porcentagem armazenando na variável "percentual"

Agora, na posição $x = 0.4e7$ e $y = 5$, adicione o seguinte texto com tamanho 12 e cor AZUL_3: `f"$\bf{percentual:.2f}$% das vendas correspondem apenas as vendas de Eletrônicos.\nIsso é equivalente a quase o dobro em relação a todas as outras categorias\npresentes no portfólio da Zoop."`.

Utilize as bibliotecas de visualização de dados do Python para criar essa visualização.

In [881...

```
def faturamento_por_categoria():
    import matplotlib.pyplot as plt

    # Agrupar por categoria e calcular o faturamento total
    faturamento_por_categoria = df.groupby('categoria')['faturamento'].sum().sort_index()

    # Calcular percentual
    total_faturamento = faturamento_por_categoria.sum()
    percentual = faturamento_por_categoria.iloc[0] / total_faturamento * 100

    # Configurações do gráfico
    plt.figure(figsize=(10, 6), facecolor=CINZA_1)
    ax = plt.axes()
    ax.set_facecolor(CINZA_1)
    bars = plt.barh(faturamento_por_categoria.index, faturamento_por_categoria, color=CINZA_5)

    # Adicionar rótulos e título
    plt.title('Faturamento por Categoria de Produto em 2023', fontsize=18, color=CINZA_5)
    plt.xlabel('')
    plt.ylabel('')
    plt.yticks(fontsize=12, color=CINZA_5)
    plt.xticks([])
    ax.spines[['top', 'right', 'left', 'bottom']].set_visible(False)

    # Adicionar total do faturamento acima de cada barra
    for bar in bars:
        plt.text(bar.get_width() + 1e5, bar.get_y() + bar.get_height()/2,
                 f'R$ {bar.get_width()/1e6:.2f} M',
                 ha='left', va='center', color=CINZA_5, fontsize=12)

    # Adicionar texto dentro da área do plot
    plt.text(0.4e7, 6, f"${\\bf{percentual:.2f}}\% das vendas correspondem apenas a {percentual:.2f}\% das vendas de Eletrônicos. Isso é equivalente a quase o dobro em relação a todas as outras categorias presentes no portfólio da Zoop.",
            ha='left', va='center', color=CINZA_5, fontsize=12)

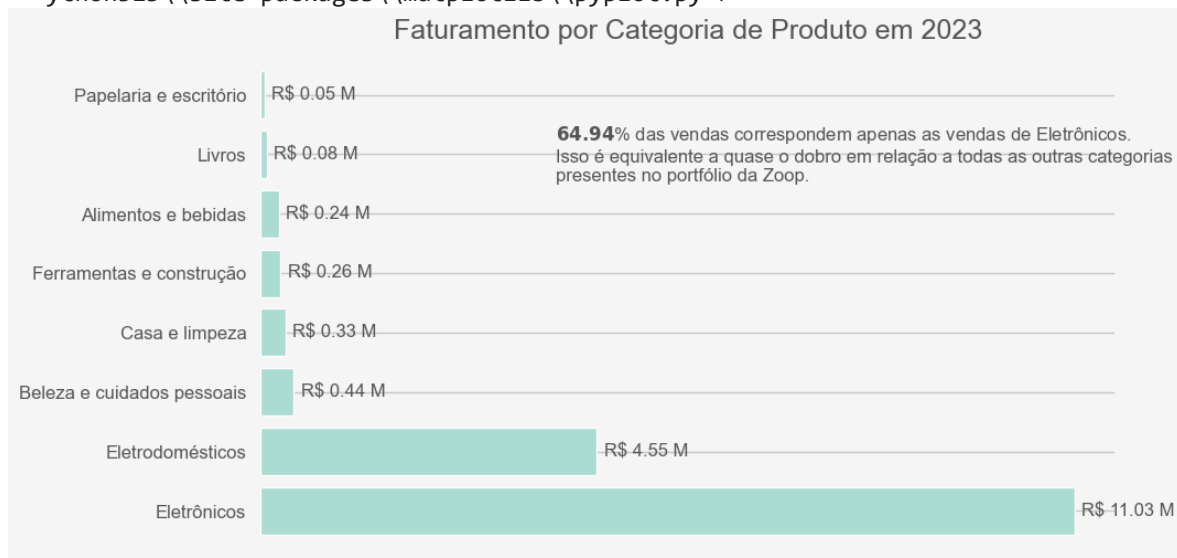
    return plt
```

In [882...

faturamento_por_categoria()

Out[882...

<module 'matplotlib.pyplot' from 'C:\\Users\\joaoc\\AppData\\Roaming\\Python\\Python313\\site-packages\\matplotlib\\pyplot.py'>



Comentários:

Fizemos com um único prompt a inserção dos rótulos, alteração das cores e adicionamos um texto explicativo. Com a sequência de explicações para o ChatGPT ele nota alguns padrões na construção e cria parte dos códigos mesmo sem mais detalhes.

5.2 - Ajustando os visuais de composição

Já chegamos a praticamente metade dos gráficos refinados. Agora, vamos neste tópico ajustar mais 2 aplicando as técnicas de storytelling que aprendemos aqui no curso por meio da utilização da IA.

Gráfico 4 - Vendas por trimestre e método de pagamento:

In [883...

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

# Supondo que você tenha o DataFrame "df" carregado

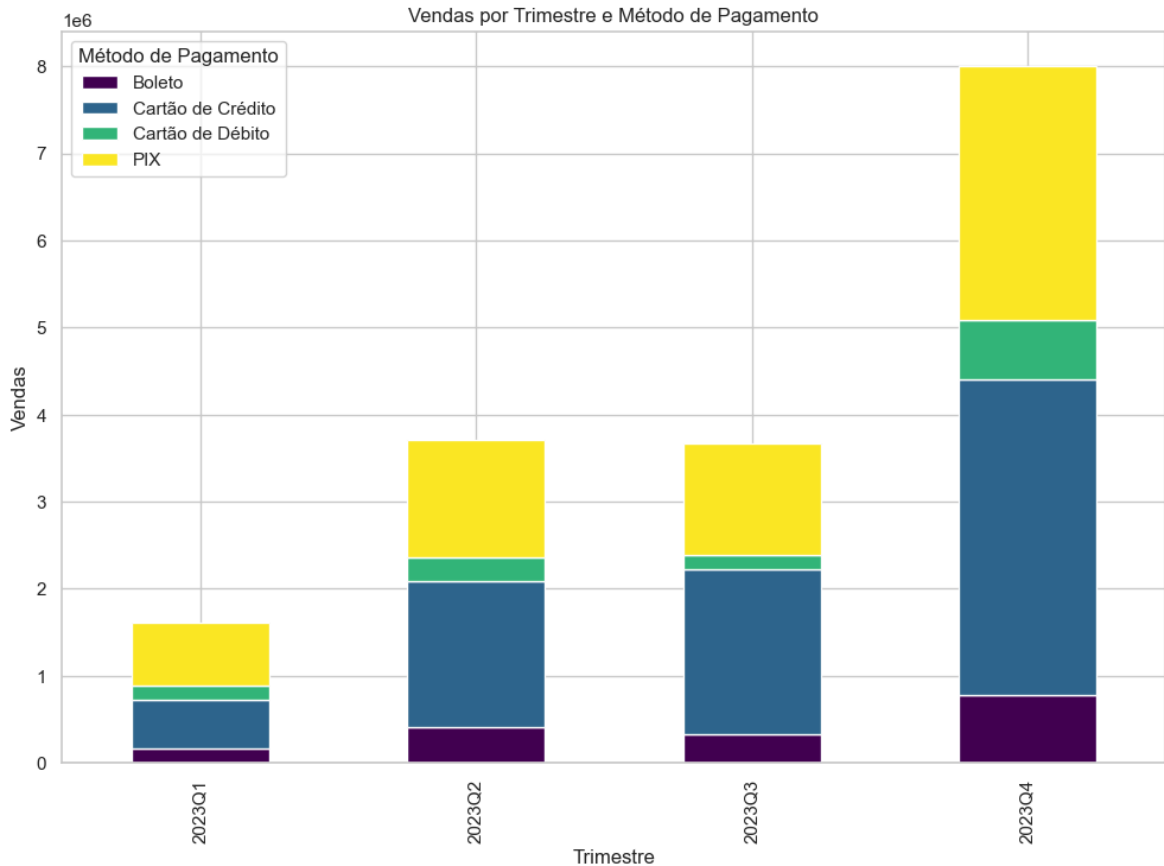
# Converter a coluna 'data' para o formato datetime, se ainda não estiver
df['data'] = pd.to_datetime(df['data'])

# Criar uma nova coluna para identificar o trimestre
df['trimestre'] = df['data'].dt.to_period('Q')

# Agrupar por trimestre e método de pagamento, somando o faturamento
vendas_por_trimestre = df.groupby(['trimestre', 'metodo_pagamento'])['faturament

# Visualização em gráfico de barras empilhadas
vendas_por_trimestre.plot(kind='bar', stacked=True, figsize=(12, 8), colormap='v
plt.title('Vendas por Trimestre e Método de Pagamento')
plt.xlabel('Trimestre')
plt.ylabel('Vendas')
plt.legend(title='Método de Pagamento')

plt.show()
```



Prompt:

Vamos agora para o gráfico de Vendas por Trimestre e Método de Pagamento que produzimos anteriormente. Vamos refiná-lo usando as seguintes orientações.

O nosso gráfico precisa seguir a identidade visual da Zoop, com o fundo da figura e do plot na cor CINZA_1 e o tamanho da figura deve ser de (10, 6). A sequência de cores das colunas empilhadas (palette) é [BRANCO, CINZA_3, AQUA_1, AQUA_3].

O título da figura é "Vendas por Trimestre e Método de Pagamento em 2023" com tamanho 18 e sua cor CINZA_5.

Os rótulos do eixo x (xticks) precisam estar em 0 graus e ter tamanho 12 com a cor AZUL_1. Os rótulos do eixo y (yticks) devem estar no formato f"R\$ valor M" sem casas decimais e os labels do eixo x e y (xlabel e ylabel) devem ser suprimidos. Todas as bordas do gráfico devem ser suprimidas(['top','bottom','left','right']).

Adicione também um grid tracejado em y

Utilize as bibliotecas de visualização de dados do Python para criar essa visualização.

In [884...

```
def vendas_tri_metodo():
    # Agrupar por trimestre e método de pagamento e calcular o faturamento total
    vendas_por_trimestre_metodo = df.groupby(['trimestre', 'metodo_pagamento'])['faturamento'].sum()

    # Configurações do gráfico
    plt.figure(figsize=(10, 6), facecolor=CINZA_1)
    ax = plt.axes()
```

```

ax.set_facecolor(CINZA_1)

# Plotar o gráfico de barras empilhadas
vendas_por_trimestre_metodo.plot(kind='bar', stacked=True, color=[BRANCO, CINZA_5, CINZA_4, CINZA_3])

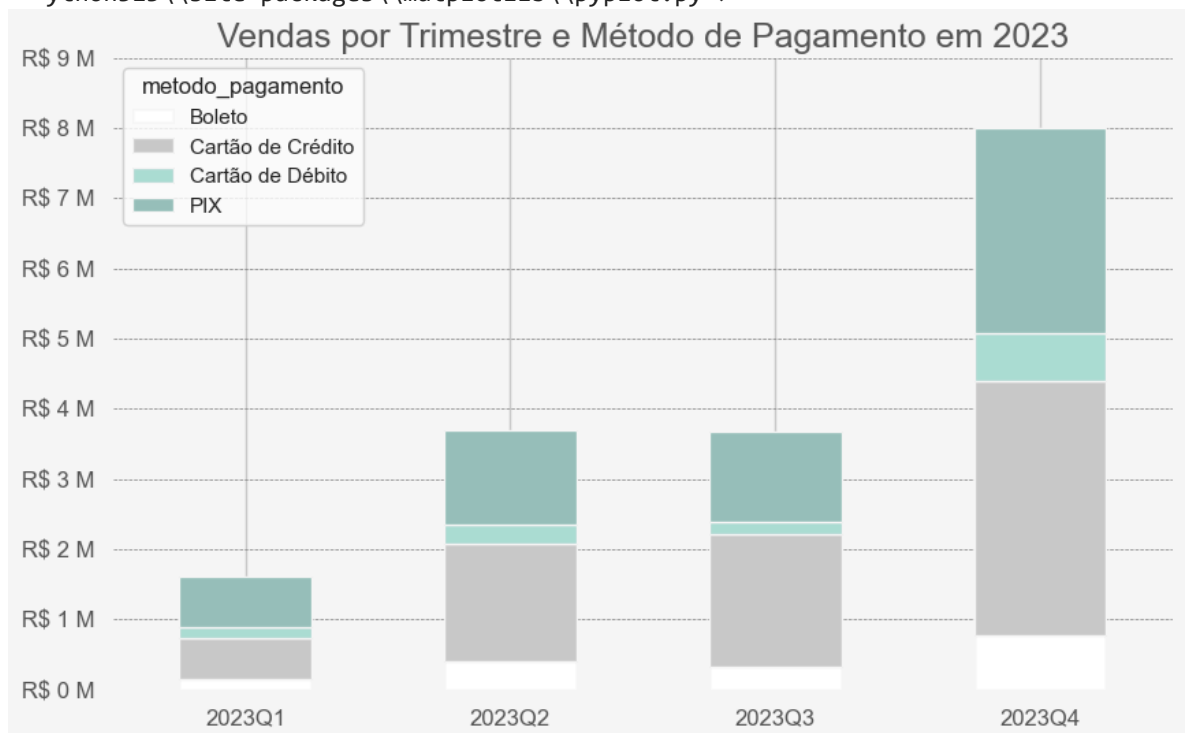
# Adicionar rótulos e título
plt.title('Vendas por Trimestre e Método de Pagamento em 2023', fontsize=18, color=BRANCO)
plt.xlabel('')
plt.ylabel('')
plt.xticks(rotation=0, fontsize=12, color=CINZA_5)
plt.yticks(ax.get_yticks(), ['R$ {:.0f} M'.format(val/1e6) for val in ax.get_yticks()])
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)
ax.spines['bottom'].set_visible(False)
ax.yaxis.grid(color='gray', linestyle='--', linewidth=0.5)
ax.set_axisbelow(True) # adicionar grid atrás

return plt

```

In [885... vendas_tri_metodo()]

Out[885... <module 'matplotlib.pyplot' from 'C:\\Users\\joaoc\\AppData\\Roaming\\Python\\Python313\\site-packages\\matplotlib\\pyplot.py'>



Comentários:

Ajustamos o gráfico de barras empilhadas representando com cores mais brandas os valores dos métodos de pagamento. Podemos notar coisas interessantes nesse gráfico, como: 2º e 4º tri como sendo os momentos em que a Zoop fatura mais com vendas e, também, o aumento do pagamento das compras feitas pelos clientes por meio do cartão de crédito.

Gráfico 5 - Clientes participantes do programa de cashback:

Agora chegou a hora do gráfico de setores. Ele já esta bem encaminhado, mas precisamos realizar alguns ajustes.

In [886...

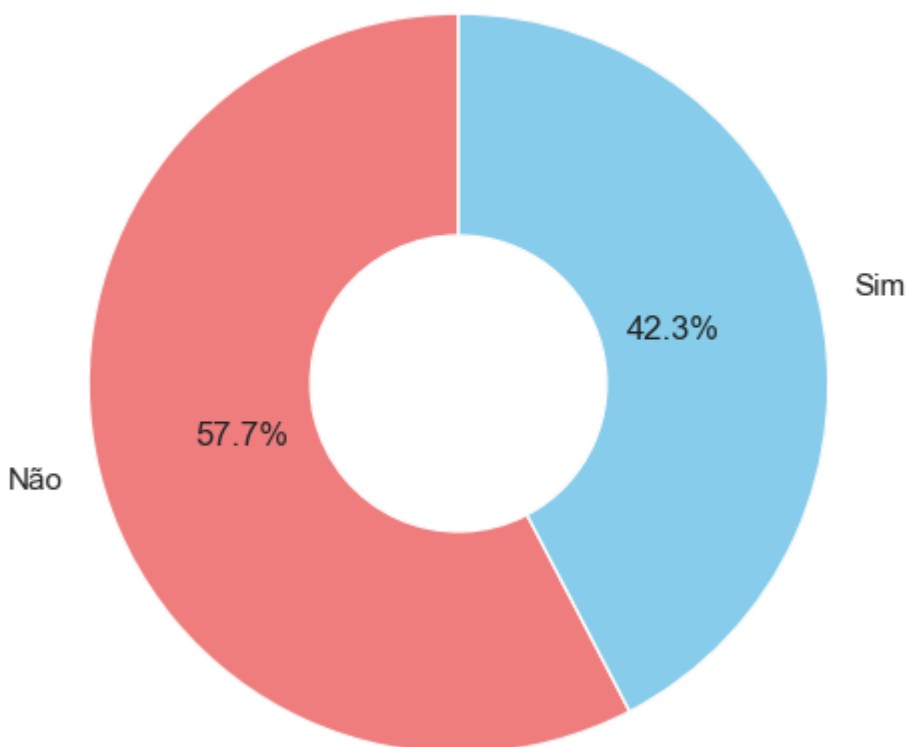
```
import matplotlib.pyplot as plt

# Calcular a proporção de adesão ao cashback
proporcao_cashback = df['cashback'].value_counts(normalize=True)

# Cores invertidas
cores_invertidas = ['lightcoral', 'skyblue']

# Visualização em gráfico de rosca (donut)
plt.figure(figsize=(6, 6))
plt.pie(proporcao_cashback, labels=proporcao_cashback.index, autopct='%1.1f%%',
plt.title('Proporção de Usuários Aderindo ao Cashback')
plt.show()
```

Proporção de Usuários Aderindo ao Cashback



Prompt:

Agora, vamos refinar o gráfico de rosca que produzimos para a proporção de usuários aderindo ao programa de cashback.

O nosso gráfico precisa seguir a identidade visual da Zoop, com o fundo da figura e do plot na cor CINZA_1, a cor da fatia de valor "Não" igual a CINZA_3, a cor da fatia de valor "Sim" igual a AZUL_1 e o tamanho da figura deve ser de (6, 6).

O título da figura é "Participam do programa de cashback?" com tamanho 18 e sua cor CINZA_5.

Os rótulos das fatias precisam ter a mesma cor das fatias e tamanho 12 e as porcentagens precisam estar na cor BRANCO, também com tamanho 12.

A propriedade da largura da rosca (`wedgeprops`) precisa ser de 0.4, a posição das porcentagens dentro da fatia (`pctdistance`) precisa ser de 0.8, o ângulo de início de 180 (`startangle`) e sem explosão de fatias.

Utilize as bibliotecas de visualização de dados do Python para criar essa visualização.

```
In [887... def cashback():
    # Calcular a proporção de usuários aderindo ao programa de cashback
    proporcao_cashback = df['cashback'].value_counts(normalize=True)

    # Configurações do gráfico de rosca
    fig, ax = plt.subplots(figsize=(6, 6), facecolor=CINZA_1)
    ax.set_facecolor(CINZA_1)

    # Plotar o gráfico de rosca
    wedges, texts, autotexts = ax.pie(proporcao_cashback, labels=proporcao_cashback.index,
                                      wedgeprops=dict(width=0.4, edgecolor=CINZA_1))

    # Adicionar rótulos e título
    plt.title('Participam do Programa de Cashback?', fontsize=18, color=CINZA_5)

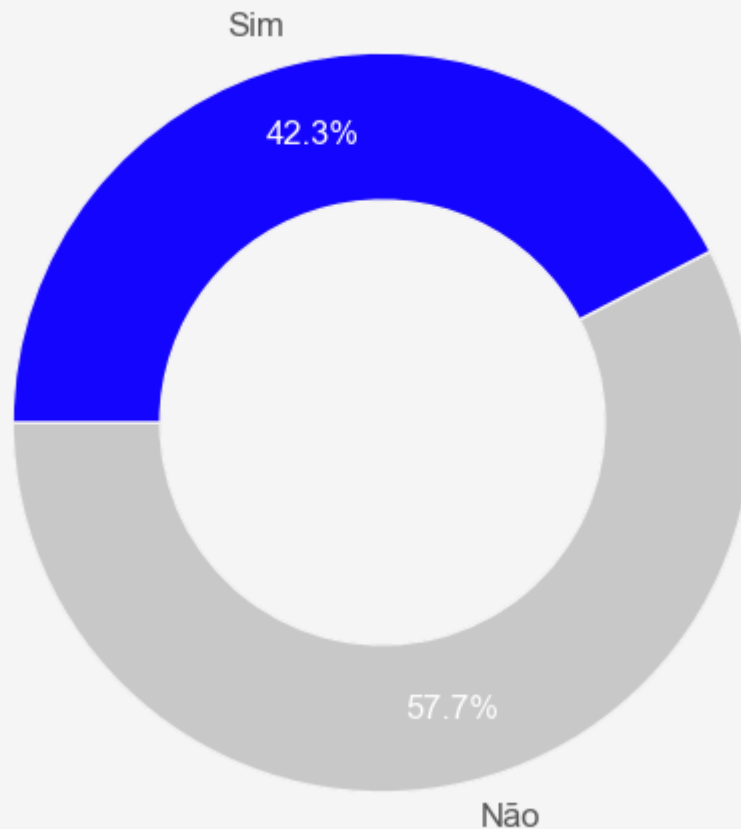
    # Configurar rótulos e porcentagens com as cores especificadas
    for text, autotext, color in zip(texts, autotexts, [CINZA_5, CINZA_5]):
        text.set_color(color)
        text.set_fontsize(12)
        autotext.set_color(BRANCO)
        autotext.set_fontsize(12)

    return plt
```

```
In [888... cashback()
```

```
Out[888... <module 'matplotlib.pyplot' from 'C:\\Users\\joaoc\\AppData\\Roaming\\Python\\Python313\\site-packages\\matplotlib\\pyplot.py'>
```

Participam do Programa de Cashback?



Comentários:

Alteramos mais um gráfico, agora o de rosca, para representar a proporção dos(a) nossos(as) clientes. Demos maior destaque ao "Sim", utilizando a cor como recurso de ponto focal.

5.3 - Ajustando os visuais de distribuição

Agora vamos para os últimos 2 gráficos antes da exportação dos visuais. Vamos aplicar mais técnicas de storytelling que aprendemos por meio da utilização da IA.

Gráfico 6 - Avaliações dos produtos:

Agora vamos para a distribuição das avaliações dos produtos em que criamos um histograma. Vamos adicionar mais um novo elemento visual e realizar alguns ajustes.

In [889...

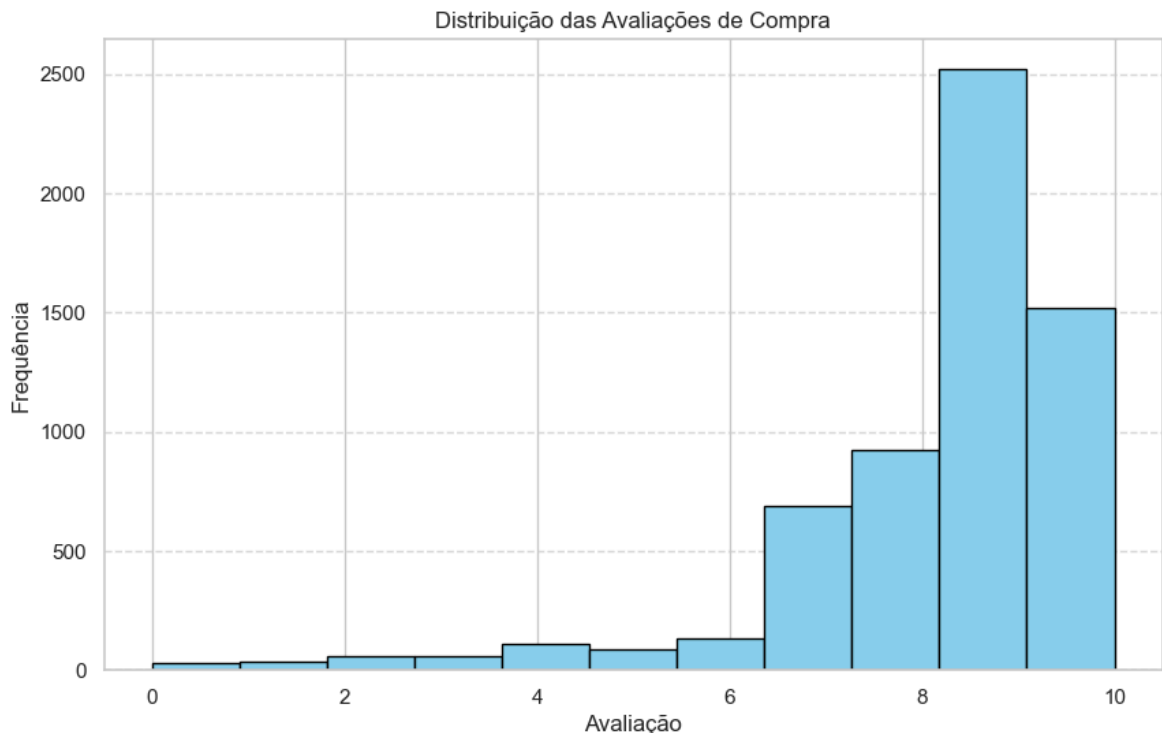
```
import matplotlib.pyplot as plt

# Supondo que você tenha o DataFrame "df" carregado

# Visualização em histograma
plt.figure(figsize=(10, 6))
```

```
plt.hist(df['avaliacao_compra'], bins=11, edgecolor='black', color='skyblue')
plt.title('Distribuição das Avaliações de Compra')
plt.xlabel('Avaliação')
plt.ylabel('Frequência')
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()
```



Prompt:

Vamos agora para o gráfico da Distribuição das Avaliações de Compra que produzimos anteriormente. Vamos refiná-lo usando as seguintes orientações.

O nosso gráfico precisa seguir a identidade visual da Zoop, com o fundo da figura e do plot na cor CINZA_1, a cor das colunas VERMELHO_1 e o tamanho da figura deve ser de (10, 6).

O título da figura é "Distribuição das avaliações das compras em 2023" com tamanho 18 e sua cor CINZA_5.

Os rótulos do eixo x (xticks) precisa estar no range de 0 a 10 com bins de 11 bins e ter tamanho 12 com a cor AZUL_1. Os rótulos do eixo y (yticks) devem ter tamanho 12 com a cor AZUL_1 e os labels do eixo x e y (xlabel e ylabel) também devem ter tamanho 12 com a cor AZUL_1. Todas as bordas do gráfico devem ser suprimidas(['top','bottom','left','right']).

O grid do y tem que estar atrás do gráfico (`ax.set_axisbelow(True)`).

Vamos adicionar uma caixa de anotação (`ax.annotate()`) com o fundo na cor BRANCO e a borda na cor CINZA_3 dentro da área do plot. O texto da anotação é


```
f"Média de avaliações =
{df['avaliacao_compra'].mean().round(2)}\n
Avaliação mais comum =
{df['avaliacao_compra'].value_counts().idxmax()}\n
Frequência da mais comum =
{df['avaliacao_compra'].value_counts().max()}"
```

na cor AZUL_1. Adicione a caixa da anotação com o texto na posição xytext=(2, 1800) e coloque uma seta, na cor AZUL_1, saindo da caixa até a posição xy=(9, 2250).

Utilize as bibliotecas de visualização de dados do Python para criar essa visualização.

In [890...

```
import matplotlib.pyplot as plt
import seaborn as sns

def avaliacao_compra():
    # Configurações do gráfico
    plt.figure(figsize=(10, 6), facecolor=CINZA_2)
    ax = plt.axes()
    ax.set_facecolor(CINZA_2)

    # Plotar o histograma das avaliações
    sns.histplot(df['avaliacao_compra'], bins=11, color=VERMELHO_1, kde=False, a

    # Adicionar rótulos e título
    plt.title('Distribuição das Avaliações das Compras em 2023', fontsize=18, co
    plt.xlabel('Avaliação', fontsize=12, color=CINZA_5)
    plt.ylabel('Frequência', fontsize=12, color=CINZA_5)
    plt.xticks(range(11), fontsize=12, color=CINZA_5)
    plt.yticks(fontsize=12, color=CINZA_5)
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.spines['left'].set_visible(False)
    ax.spines['bottom'].set_visible(False)
    ax.set_axisbelow(True)
    ax.yaxis.grid(color='gray', linestyle='--', linewidth=0.5)

    # Adicionar caixa de anotação
    media_avaliacao = df['avaliacao_compra'].mean().round(2)
    avaliacao_mais_comum = df['avaliacao_compra'].value_counts().idxmax()
    freq_avaliacao_mais_comum = df['avaliacao_compra'].value_counts().max()

    texto_annotacao = f"Média de avaliações = {media_avaliacao}\nAvaliação mais c

    ax.annotate(texto_annotacao, xy=(9, 2250), xytext=(2, 1800), fontsize=12, col
                bbox=dict(boxstyle="round,pad=0.3", edgecolor=CINZA_5, facecolor
                arrowprops=dict(facecolor=CINZA_5, arrowstyle="->", connectionst

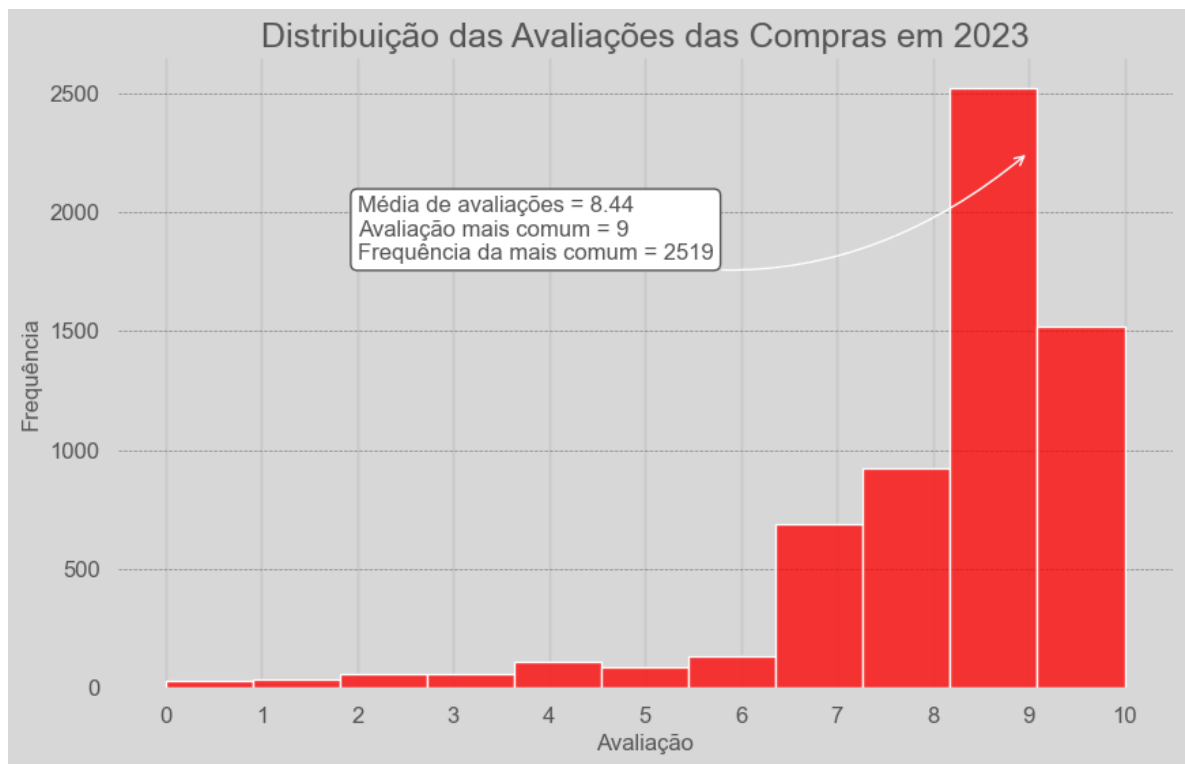
    return plt
```

In [891...

```
avaliacao_compra()
```

Out[891...

```
<module 'matplotlib.pyplot' from 'C:\\Users\\joaoc\\AppData\\Roaming\\Python\\P
ython313\\site-packages\\matplotlib\\pyplot.py'>
```



Comentários:

Mais uma novidade aqui em nosso gráfico! Adicionamos o recurso de bloco de anotações que é muito bom para colocar informações a respeito do dado. Com ele é possível utilizar as setas como indicadores para destaque.

Gráfico 7 - Distribuição do público por idade e sexo biológico:

Agora vamos para o nosso último visual!

```
In [892... import matplotlib.pyplot as plt
import seaborn as sns

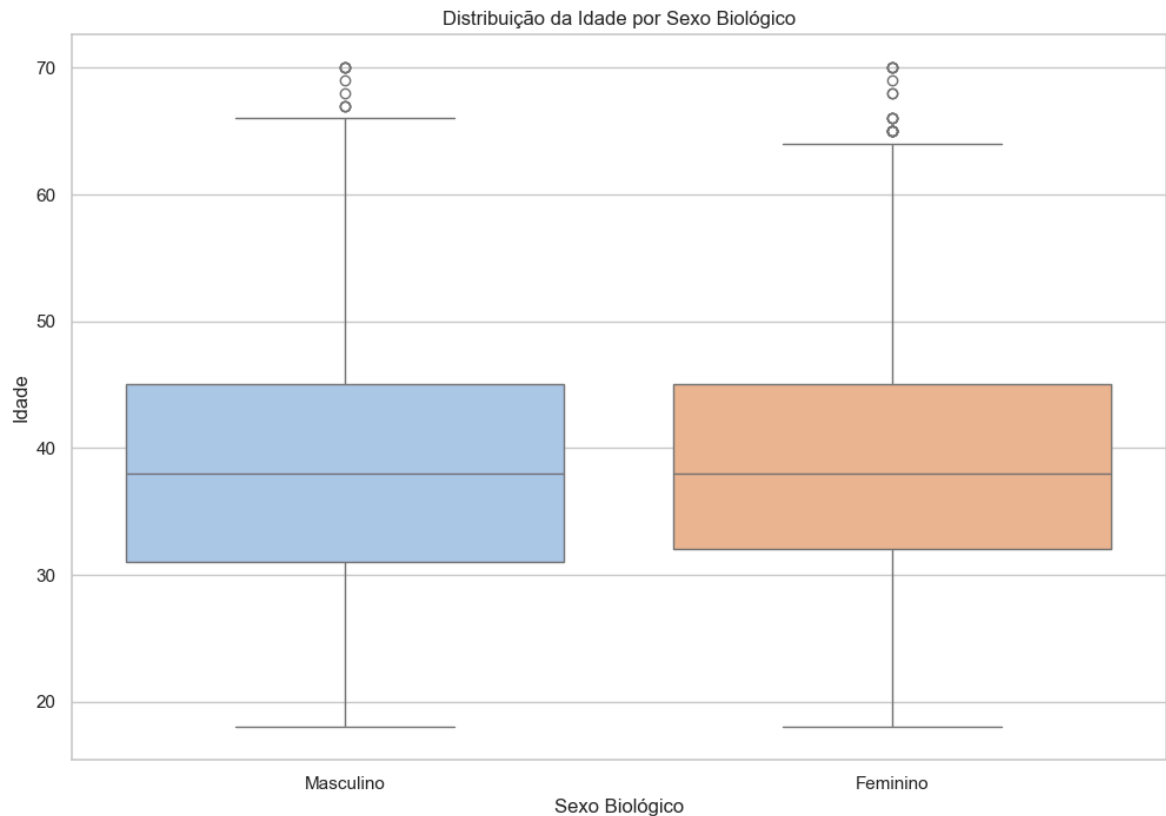
# Supondo que você tenha o DataFrame "df" carregado

# Visualização em boxplot
plt.figure(figsize=(12, 8))
sns.boxplot(data=df, x='sexo_biologico', y='idade', palette='pastel')
plt.title('Distribuição da Idade por Sexo Biológico')
plt.xlabel('Sexo Biológico')
plt.ylabel('Idade')

plt.show()
```

C:\Users\joaoc\AppData\Local\Temp\ipykernel_28956\3149611002.py:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v 0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.



Prompt:

Vamos agora para o gráfico de boxplot da Distribuição da Idade por Sexo Biológico que produzimos anteriormente. Vamos refiná-lo usando as seguintes orientações.

O nosso gráfico precisa seguir a identidade visual da Zoop, com o fundo da figura e do plot na cor CINZA_1, e o tamanho da figura deve ser de (10, 6).

O título da figura é "Distribuição da Idade por Sexo Biológico dos(as) clientes" com tamanho 18 e sua cor CINZA_5.

A coluna 'sexo_biologico' é representada no eixo x e a 'idade' no eixo y. A paleta de cores para os valores de x é [AZUL_1, AQUA_1].

O label do eixo x (xlabel) deve ser suprimido. Os rótulos do eixo x e y (xticks e yticks) devem ter tamanho 12 com a cor AZUL_1 e o label do eixo y (ylabel) também deve ter tamanho 12 com a cor AZUL_1. Todas as bordas do gráfico devem ser suprimidas e o grid do eixo y tem que estar atrás do gráfico (`ax.set_axisbelow(True)`).

Vamos adicionar duas caixas de anotação (`ax.annotate()`) com o fundo na cor BRANCO e a borda na cor CINZA_3 dentro da área do plot.

O texto da primeira anotação na cor AZUL_1 é

```
f"Mínimo = df.groupby('sexo_biologico')['idade'].min()[1]\n
1º quartil = df.groupby('sexo_biologico')['idade'].quantile(.25)\n
[1]\n
2º quartil = df.groupby('sexo_biologico')['idade'].median()[1]\n
3º quartil = df.groupby('sexo_biologico')['idade'].quantile(.75)\n
[1]\n
Máximo = df.groupby('sexo_biologico')['idade'].max()[1]"
```

Adicione a caixa da anotação com o texto na posição `xytext=(-0.4, 50)` e coloque uma seta, na cor AZUL_1, saindo da caixa até a posição `xy=(-0.1, 45)`.

O texto da segunda anotação na cor AZUL_1 é

```
f"Mínimo = df.groupby('sexo_biologico')['idade'].min()[0]\n
1º quartil = df.groupby('sexo_biologico')['idade'].quantile(.25)\n
[0]\n
2º quartil = df.groupby('sexo_biologico')['idade'].median()[0]\n
3º quartil = df.groupby('sexo_biologico')['idade'].quantile(.75)\n
[0]\n
Máximo = df.groupby('sexo_biologico')['idade'].max()[0]"
```

Adicione a caixa da anotação com o texto na posição `xytext=(0.6, 50)` e coloque uma seta, na cor AZUL_1, saindo da caixa até a posição `xy=(0.9, 45)`.

Utilize as bibliotecas de visualização de dados do Python para criar essa visualização.

In [893...

```
import matplotlib.pyplot as plt
import seaborn as sns

def dist_idade_sexo():
    # Configurações do gráfico
    plt.figure(figsize=(10, 6), facecolor=CINZA_1)
    ax = plt.axes()
    ax.set_facecolor(CINZA_1)

    # Plotar o boxplot da distribuição da idade por sexo biológico
    sns.boxplot(data=df, x='sexo_biologico', y='idade', hue='sexo_biologico', pa

    # Adicionar rótulos e título
    plt.title('Distribuição da Idade por Sexo Biológico dos(as) clientes', fonts
    plt.xlabel('')
    plt.ylabel('Idade', fontsize=12, color=CINZA_5)
    plt.xticks(fontsize=12, color=CINZA_5)
    plt.yticks(fontsize=12, color=CINZA_5)
    ax.spines[['top', 'right', 'left', 'bottom']].set_visible(False)
    ax.set_axisbelow(True)
    ax.yaxis.grid(color='gray', linestyle='--', linewidth=0.5)

    # Adicionar caixas de anotação
    texto_annotacao1 = f"Mínimo = {df.groupby('sexo_biologico')['idade'].min()[1]}
```

```

f"1º quartil = {df.groupby('sexo_biologico')['idade'].quantil(1)}
f"2º quartil = {df.groupby('sexo_biologico')['idade'].median()}
f"3º quartil = {df.groupby('sexo_biologico')['idade'].quantil(3)}
f"Máximo = {df.groupby('sexo_biologico')['idade'].max()[1]}"

texto_annotacao2 = f"Mínimo = {df.groupby('sexo_biologico')['idade'].min()[0]}
f"1º quartil = {df.groupby('sexo_biologico')['idade'].quantil(1)}
f"2º quartil = {df.groupby('sexo_biologico')['idade'].median()}
f"3º quartil = {df.groupby('sexo_biologico')['idade'].quantil(3)}
f"Máximo = {df.groupby('sexo_biologico')['idade'].max()[0]}"

ax.annotate(texto_annotacao1, xy=(-0.1, 45), xytext=(-0.4, 50), fontsize=12,
            bbox=dict(boxstyle="round,pad=0.3", edgecolor=CINZA_5, facecolor="white"),
            arrowprops=dict(facecolor=CINZA_5, arrowstyle="->", connectionstyle="arc, radius=100"))

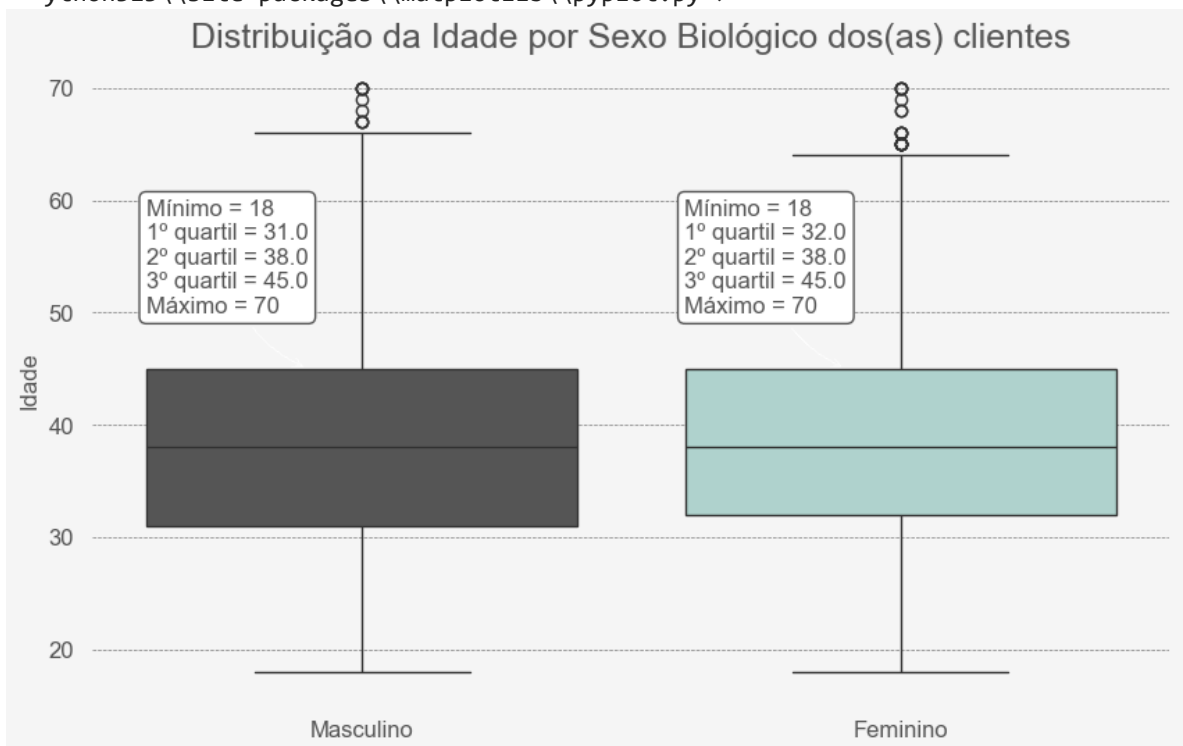
ax.annotate(texto_annotacao2, xy=(0.9, 45), xytext=(0.6, 50), fontsize=12,
            bbox=dict(boxstyle="round,pad=0.3", edgecolor=CINZA_5, facecolor="white"),
            arrowprops=dict(facecolor=CINZA_5, arrowstyle="->", connectionstyle="arc, radius=100"))

return plt

```

In [894... dist_idade_sexo()

Out[894... <module 'matplotlib.pyplot' from 'C:\\Users\\joaoc\\AppData\\Roaming\\Python\\Python313\\site-packages\\matplotlib\\pyplot.py'>



Comentários:

Para nosso último gráfico, criamos novamente o bloco de anotações apontando para os valores que são exibidos em um boxplot. Notamos que a distribuição entre o clientes do sexo biológico masculino e feminino está bem próxima com apenas um ano a mais no intervalo interquartil (1º a 3º) no caso do sexo biológico masculino. A Zoop tem na mediana a clientela dentro dos 38 anos de idade para ambos sexos.

5.4 - Exportando os visuais

Pronto! Agora que finalizamos os ajustes de todos os gráficos requisitados, vamos gerar um último código para exportar cada um dos gráficos que transformamos em funções no formato de imagem (PNG).

Prompt:

Agora, vamos para a nossa ultima etapa: salvar as visualizações como imagem. Coloquei cada um dos gráficos que geramos dentro de uma função e quero que seja aplicada a cada uma delas o seguinte procedimento:

```
nome_imagem = nome_funcao()
nome_imagem.savefig('nome_imagem.png', bbox_inches='tight')
nome_imagem.close()
```

As funções são:

- metodos_pag()
- vendas_mensais()
- faturamento_por_categoria()
- vendas_tri_metodo()
- cashback()
- avaliacao_compra()
- dist_idade_sexo()

Troque a variável `nome_imagem` por `grafico_` seguido do nome da função que gera o gráfico.

Utilize a linguagem Python para criar o código e não exiba as imagens.

In [899...

```
# Salvar os gráficos gerados por cada função como imagem

grafico_metodos_pag = metodos_pag()
grafico_metodos_pag.savefig('grafico_metodos_pag.png', bbox_inches='tight')
grafico_metodos_pag.close()

grafico_vendas_mensais = vendas_mensais()
grafico_vendas_mensais.savefig('grafico_vendas_mensais.png', bbox_inches='tight')
grafico_vendas_mensais.close()

grafico_faturamento_por_categoria = faturamento_por_categoria()
grafico_faturamento_por_categoria.savefig('grafico_faturamento_por_categoria.png')
grafico_faturamento_por_categoria.close()

grafico_vendas_tri_metodo = vendas_tri_metodo()
grafico_vendas_tri_metodo.savefig('grafico_vendas_tri_metodo.png', bbox_inches='tight')
grafico_vendas_tri_metodo.close()

grafico_cashback = cashback()
```

```
grafico_cashback.savefig('grafico_cashback.png', bbox_inches='tight')
grafico_cashback.close()

grafico_avaliacao_compra = avaliacao_compra()
grafico_avaliacao_compra.savefig('grafico_avaliacao_compra.png', bbox_inches='tight')
grafico_avaliacao_compra.close()

grafico_dist_idade_sexo = dist_idade_sexo()
grafico_dist_idade_sexo.savefig('grafico_dist_idade_sexo.png', bbox_inches='tight')
grafico_dist_idade_sexo.close()
```

Agora é a sua vez! Que tal baixar os gráficos e gerar um relatório trazendo alguns insights e explicando os resultados?

Espero que tenha curtido este curso e faça bom proveito do conteúdo!