

## ✓ Extracção


```
import pandas as pd
import requests
```

```
# URL da API
url = 'https://raw.githubusercontent.com/alura-cursos/challenge2-data-science/refs/heads/main/TelecomX_Data.json'
```

```
# Coleta dos dados
response = requests.get(url)
data_json = response.json()
```

```
# Conversão para DataFrame
df = pd.json_normalize(data_json)
```


```
# Visualização inicial
df.head()
```



	customerID	Churn	customer.gender	customer.SeniorCitizen	customer.Partner	customer.Dependents	customer.tenure	phone.PhoneSe
0	0002-ORFBO	No	Female	0	Yes	Yes	9	
1	0003-MKNFE	No	Male	0	No	No	9	
2	0004-TLHLJ	Yes	Male	0	No	No	4	
3	0011-IGKFF	Yes	Male	1	Yes	No	13	
4	0013-EXCHZ	Yes	Female	1	Yes	No	3	

5 rows × 21 columns

```
df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7267 entries, 0 to 7266
Data columns (total 21 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   customerID                               7267 non-null   object
1   Churn                                     7267 non-null   object
2   customer.gender                           7267 non-null   object
3   customer.SeniorCitizen                    7267 non-null   int64
4   customer.Partner                           7267 non-null   object
5   customer.Dependents                       7267 non-null   object
6   customer.tenure                           7267 non-null   int64
7   phone.PhoneService                        7267 non-null   object
8   phone.MultipleLines                       7267 non-null   object
9   internet.InternetService                  7267 non-null   object
10  internet.OnlineSecurity                    7267 non-null   object
11  internet.OnlineBackup                      7267 non-null   object
12  internet.DeviceProtection                  7267 non-null   object
13  internet.TechSupport                       7267 non-null   object
14  internet.StreamingTV                       7267 non-null   object
15  internet.StreamingMovies                   7267 non-null   object
16  account.Contract                          7267 non-null   object
17  account.PaperlessBilling                   7267 non-null   object
18  account.PaymentMethod                      7267 non-null   object
19  account.Charges.Monthly                    7267 non-null   float64
20  account.Charges.Total                      7267 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.2+ MB
```

## ✓ Transformação

```
# Padronizar nomes das colunas (tudo minúsculo e sem pontos)
df.columns = [col.lower().replace('.', '_') for col in df.columns]
```

```
df.info() # Mostra quantas colunas, tipos e se há valores nulos
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7267 entries, 0 to 7266
Data columns (total 21 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   customerid                           7267 non-null   object
1   churn                                7267 non-null   object
2   customer_gender                       7267 non-null   object
3   customer_seniorcitizen                7267 non-null   int64
4   customer_partner                      7267 non-null   object
5   customer_dependents                   7267 non-null   object
6   customer_tenure                       7267 non-null   int64
7   phone_phoneservice                    7267 non-null   object
8   phone_multiplelines                   7267 non-null   object
9   internet_internetsevice               7267 non-null   object
10  internet_onlinesecurity                7267 non-null   object
11  internet_onlinebackup                  7267 non-null   object
12  internet_deviceprotection              7267 non-null   object
13  internet_techsupport                   7267 non-null   object
14  internet_streamingtv                   7267 non-null   object
15  internet_streamingmovies               7267 non-null   object
16  account_contract                       7267 non-null   object
17  account_paperlessbilling               7267 non-null   object
18  account_paymentmethod                  7267 non-null   object
19  account_charges_monthly                7267 non-null   float64
20  account_charges_total                  7267 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.2+ MB

```

```

# Corrigir a coluna de gastos totais para ser numérica
df['account_charges_total'] = pd.to_numeric(df['account_charges_total'], errors='coerce')

```

```
df.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7267 entries, 0 to 7266
Data columns (total 21 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   customerid                           7267 non-null   object
1   churn                                7267 non-null   object
2   customer_gender                       7267 non-null   object
3   customer_seniorcitizen                7267 non-null   int64
4   customer_partner                      7267 non-null   object
5   customer_dependents                   7267 non-null   object
6   customer_tenure                       7267 non-null   int64
7   phone_phoneservice                    7267 non-null   object
8   phone_multiplelines                   7267 non-null   object
9   internet_internetsevice               7267 non-null   object
10  internet_onlinesecurity                7267 non-null   object
11  internet_onlinebackup                  7267 non-null   object
12  internet_deviceprotection              7267 non-null   object
13  internet_techsupport                   7267 non-null   object
14  internet_streamingtv                   7267 non-null   object
15  internet_streamingmovies               7267 non-null   object
16  account_contract                       7267 non-null   object
17  account_paperlessbilling               7267 non-null   object
18  account_paymentmethod                  7267 non-null   object
19  account_charges_monthly                7267 non-null   float64
20  account_charges_total                  7256 non-null   float64
dtypes: float64(2), int64(2), object(17)
memory usage: 1.2+ MB

```

```

# Ver quantos valores estão faltando em cada coluna
print(df.isnull().sum())

```

```

customerid      0
churn           0
customer_gender 0
customer_seniorcitizen 0
customer_partner 0
customer_dependents 0
customer_tenure 0
phone_phoneservice 0
phone_multiplelines 0
internet_internetsevice 0
internet_onlinesecurity 0
internet_onlinebackup 0
internet_deviceprotection 0
internet_techsupport 0
internet_streamingtv 0
internet_streamingmovies 0
account_contract 0
account_paperlessbilling 0

```

```

account_paymentmethod      0
account_charges_monthly    0
account_charges_total      11
dtype: int64

```

```

# Remover linhas que têm dados faltando (opcional, dependendo do caso)
df = df.dropna()

```

```
print(df['churn'].unique())
```

```

↗ ['No' 'Yes' '']

```

```
print(df['churn'].value_counts(dropna=False))
```

```

↗ churn
No      5163
Yes     1869
      224
Name: count, dtype: int64

```

```
import numpy as np
```

```
df['churn'] = df['churn'].astype(str).str.strip().str.lower()
```

```

# Passo 2: Substituir os valores válidos (inglês e português) para 'Sim' e 'Não'
df['churn'] = df['churn'].replace({
    'yes': 'Sim',
    'no': 'Não',
    'sim': 'Sim',
    'não': 'Não'
})

```

```
print(df['churn'].value_counts(dropna=False))
```

```

↗ churn
Não     5163
Sim     1869
      224
Name: count, dtype: int64

```

```
dados_xtel_sem_nulos = df[df['churn'].notna()]
```

```
dados_xtel_sem_nulos = dados_xtel_sem_nulos[dados_xtel_sem_nulos['churn'] != ''].copy()
```

```
print(dados_xtel_sem_nulos.columns)
```

```

↗ Index(['customerid', 'churn', 'customer_gender', 'customer_seniorcitizen',
        'customer_partner', 'customer_dependents', 'customer_tenure',
        'phone_phoneservice', 'phone_multiplelines', 'internet_internetservice',
        'internet_onlinesecurity', 'internet_onlinebackup',
        'internet_deviceprotection', 'internet_techsupport',
        'internet_streamingtv', 'internet_streamingmovies', 'account_contract',
        'account_paperlessbilling', 'account_paymentmethod',
        'account_charges_monthly', 'account_charges_total'],
        dtype='object')

```

```
print(dados_xtel_sem_nulos.info())
```

```

↗ <class 'pandas.core.frame.DataFrame'>
Index: 7032 entries, 0 to 7266
Data columns (total 21 columns):
#   Column                Non-Null Count  Dtype
---  -
0   customerid            7032 non-null  object
1   churn                 7032 non-null  object
2   customer_gender       7032 non-null  object
3   customer_seniorcitizen 7032 non-null  int64
4   customer_partner      7032 non-null  object
5   customer_dependents   7032 non-null  object
6   customer_tenure       7032 non-null  int64
7   phone_phoneservice    7032 non-null  object
8   phone_multiplelines   7032 non-null  object

```

```

9  internet_internet service  7032 non-null  object
10 internet_online security  7032 non-null  object
11 internet_online backup    7032 non-null  object
12 internet_device protection 7032 non-null  object
13 internet_tech support     7032 non-null  object
14 internet_streaming tv     7032 non-null  object
15 internet_streaming movies 7032 non-null  object
16 account_contract         7032 non-null  object
17 account_paperless billing 7032 non-null  object
18 account_payment method    7032 non-null  object
19 account_charges_monthly   7032 non-null  float64
20 account_charges_total     7032 non-null  float64
dtypes: float64(2), int64(2), object(17)
memory usage: 1.2+ MB
None

```

```

for coluna in dados_xtel_sem_nulos.columns:
    # Verifica se a coluna é do tipo 'object' (geralmente strings)
    if dados_xtel_sem_nulos[coluna].dtype == 'object':
        print(f"\n--- Coluna: {coluna} ---")
        # Pega os valores únicos e imprime
        print(dados_xtel_sem_nulos[coluna].unique())
        # Também pode ser útil ver a contagem de cada valor
        # print(dados_xtel_sem_nulos[coluna].value_counts())

```



```

--- Coluna: customerid ---
['0002-ORFBO' '0003-MKNFE' '0004-TLHLJ' ... '9992-UJOEL' '9993-LHIEB'
'9995-HOTOH']

```

```

--- Coluna: churn ---
['Não' 'Sim']

```

```

--- Coluna: customer_gender ---
['Female' 'Male']

```

```

--- Coluna: customer_partner ---
['Yes' 'No']

```

```

--- Coluna: customer_dependents ---
['Yes' 'No']

```

```

--- Coluna: phone_phoneservice ---
['Yes' 'No']

```

```

--- Coluna: phone_multiplelines ---
['No' 'Yes' 'No phone service']

```

```

--- Coluna: internet_internet service ---
['DSL' 'Fiber optic' 'No']

```

```

--- Coluna: internet_online security ---
['No' 'Yes' 'No internet service']

```

```

--- Coluna: internet_online backup ---
['Yes' 'No' 'No internet service']

```

```

--- Coluna: internet_device protection ---
['No' 'Yes' 'No internet service']

```

```

--- Coluna: internet_tech support ---
['Yes' 'No' 'No internet service']

```

```

--- Coluna: internet_streaming tv ---
['Yes' 'No' 'No internet service']

```

```

--- Coluna: internet_streaming movies ---
['No' 'Yes' 'No internet service']

```

```

--- Coluna: account_contract ---
['One year' 'Month-to-month' 'Two year']

```

```

--- Coluna: account_paperless billing ---
['Yes' 'No']

```

```

--- Coluna: account_payment method ---
['Mailed check' 'Electronic check' 'Credit card (automatic)'
'Bank transfer (automatic)']

```

```

# --- COLUNA: customer_gender ---
traducao_gender = {
    'Female': 'Feminino',
    'Male': 'Masculino'
}
dados_xtel_sem_nulos['customer_gender'] = dados_xtel_sem_nulos['customer_gender'].map(traducao_gender)

```

```
# --- COLUNA: customer_partner ---
traducao_partner = {
    'Yes': 'Sim',
    'No': 'Não'
}
dados_xtel_sem_nulos['customer_partner'] = dados_xtel_sem_nulos['customer_partner'].map(traducao_partner)

# --- COLUNA: customer_dependents ---
traducao_dependents = {
    'Yes': 'Sim',
    'No': 'Não'
}
dados_xtel_sem_nulos['customer_dependents'] = dados_xtel_sem_nulos['customer_dependents'].map(traducao_dependents)

# --- COLUNA: phone_phoneservice ---
traducao_phoneservice = {
    'Yes': 'Sim',
    'No': 'Não'
}
dados_xtel_sem_nulos['phone_phoneservice'] = dados_xtel_sem_nulos['phone_phoneservice'].map(traducao_phoneservice)

# --- COLUNA: phone_multiplelines ---
traducao_multiplelines = {
    'No': 'Não',
    'Yes': 'Sim',
    'No phone service': 'Sem serviço de telefone'
}
dados_xtel_sem_nulos['phone_multiplelines'] = dados_xtel_sem_nulos['phone_multiplelines'].map(traducao_multiplelines)

# --- COLUNA: internet_internet service ---
traducao_internet service = {
    'DSL': 'DSL', # Mantém, pois já é um termo técnico comum
    'Fiber optic': 'Fibra óptica',
    'No': 'Não'
}
dados_xtel_sem_nulos['internet_internet service'] = dados_xtel_sem_nulos['internet_internet service'].map(traducao_internet service)

# --- COLUNA: internet_online security ---
traducao_online security = {
    'No': 'Não',
    'Yes': 'Sim',
    'No internet service': 'Sem serviço de internet'
}
dados_xtel_sem_nulos['internet_online security'] = dados_xtel_sem_nulos['internet_online security'].map(traducao_online security)

# --- COLUNA: internet_online backup ---
traducao_online backup = {
    'Yes': 'Sim',
    'No': 'Não',
    'No internet service': 'Sem serviço de internet'
}
dados_xtel_sem_nulos['internet_online backup'] = dados_xtel_sem_nulos['internet_online backup'].map(traducao_online backup)

# --- COLUNA: internet_device protection ---
traducao_device protection = {
    'No': 'Não',
    'Yes': 'Sim',
    'No internet service': 'Sem serviço de internet'
}
dados_xtel_sem_nulos['internet_device protection'] = dados_xtel_sem_nulos['internet_device protection'].map(traducao_device protection)

# --- COLUNA: internet_tech support ---
traducao_tech support = {
    'Yes': 'Sim',
    'No': 'Não',
    'No internet service': 'Sem serviço de internet'
}
dados_xtel_sem_nulos['internet_tech support'] = dados_xtel_sem_nulos['internet_tech support'].map(traducao_tech support)

# --- COLUNA: internet_streaming tv ---
traducao_streaming tv = {
    'Yes': 'Sim',
    'No': 'Não',
```

```

    'No internet service': 'Sem serviço de internet'
}
dados_xtel_sem_nulos['internet_streamingtv'] = dados_xtel_sem_nulos['internet_streamingtv'].map(traducao_streamingtv)

# --- COLUNA: internet_streamingmovies ---
traducao_streamingmovies = {
    'No': 'Não',
    'Yes': 'Sim',
    'No internet service': 'Sem serviço de internet'
}
dados_xtel_sem_nulos['internet_streamingmovies'] = dados_xtel_sem_nulos['internet_streamingmovies'].map(traducao_streamingmovies)


# --- COLUNA: account_contract ---
traducao_contract = {
    'One year': 'Anual',
    'Month-to-month': 'Mensal',
    'Two year': 'Bienal'
}
dados_xtel_sem_nulos['account_contract'] = dados_xtel_sem_nulos['account_contract'].map(traducao_contract)

# --- COLUNA: account_paperlessbilling ---
traducao_paperlessbilling = {
    'Yes': 'Sim',
    'No': 'Não'
}
dados_xtel_sem_nulos['account_paperlessbilling'] = dados_xtel_sem_nulos['account_paperlessbilling'].map(traducao_paperlessbilling)

# --- COLUNA: account_paymentmethod ---
traducao_paymentmethod = {
    'Mailed check': 'Cheque enviado',
    'Electronic check': 'Cheque eletrônico',
    'Credit card (automatic)': 'Cartão de crédito',
    'Bank transfer (automatic)': 'Transferência bancária'
}
dados_xtel_sem_nulos['account_paymentmethod'] = dados_xtel_sem_nulos['account_paymentmethod'].map(traducao_paymentmethod)

print("Todas as traduções foram aplicadas!")
print("\nVocê pode verificar as primeiras linhas para confirmar:")
print(dados_xtel_sem_nulos.head())

```

 Todas as traduções foram aplicadas!

Você pode verificar as primeiras linhas para confirmar:

	customerid	churn	customer_gender	customer_seniorcitizen	customer_partner	\
0	0002-ORFBO	Não	Feminino	0	Sim	
1	0003-MKNFE	Não	Masculino	0	Não	
2	0004-TLHLJ	Sim	Masculino	0	Não	
3	0011-IGKFF	Sim	Masculino	1	Sim	
4	0013-EXCHZ	Sim	Feminino	1	Sim	

	customer_dependents	customer_tenure	phone_phoneservice	phone_multiplelines	\
0	Sim	9	Sim	Não	
1	Não	9	Sim	Sim	
2	Não	4	Sim	Não	
3	Não	13	Sim	Não	
4	Não	3	Sim	Não	

	internet_internetservice	... internet_onlinebackup	\
0	DSL	...	Sim
1	DSL	...	Não
2	Fibra óptica	...	Não
3	Fibra óptica	...	Sim
4	Fibra óptica	...	Não

	internet_deviceprotection	internet_techsupport	internet_streamingtv	\
0	Não	Sim	Sim	
1	Não	Não	Não	
2	Sim	Não	Não	
3	Sim	Não	Sim	
4	Não	Sim	Sim	

	internet_streamingmovies	account_contract	account_paperlessbilling	\
0	Não	Anual	Sim	
1	Sim	Mensal	Não	
2	Não	Mensal	Sim	
3	Sim	Mensal	Sim	
4	Não	Mensal	Sim	

	account_paymentmethod	account_charges_monthly	account_charges_total
0	Cheque enviado	65.6	593.30
1	Cheque enviado	59.9	542.40
2	Cheque eletrônico	73.9	280.85

3	Cheque eletrônico	98.0	1237.85
4	Cheque enviado	83.9	267.40

[5 rows x 21 columns]

```
# Selecionar as colunas numéricas e criar um novo DataFrame temporário para o heatmap
# Use .copy() para garantir que você esteja trabalhando com uma cópia independente
df_numericas_para_heatmap = dados_xtel_sem_nulos[[
    'customer_seniorcitizen',
    'customer_tenure',
    'account_charges_monthly',
    'account_charges_total'
]].copy()
```

```
# 2. Renomear as colunas do DataFrame para o português
df_numericas_para_heatmap = df_numericas_para_heatmap.rename(columns={
    'customer_seniorcitizen': 'Cliente Idoso',
    'customer_tenure': 'Tempo como Cliente (meses)',
    'account_charges_monthly': 'Cobranças Mensais (R$)',
    'account_charges_total': 'Cobranças Totais (R$)'
})
```

## ▼ Carga e análise

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

```
# Deixar os gráficos com estilo bonito e fácil de ler
sns.set(style='whitegrid')
```

```
# Definir cores padrão para os gráficos
cores = ['royalblue', 'purple', 'gray', 'navy']
```

```
# Contar os valores únicos novamente, desta vez para dados_xtel_sem_nulos
print(dados_xtel_sem_nulos['churn'].unique())
```

```
↳ ['Não' 'Sim']
```

```
print("Análise Descritiva das Colunas Numéricas do DataFrame:")
print(dados_xtel_sem_nulos.describe())
```

```
↳ Análise Descritiva das Colunas Numéricas do DataFrame:
```

	customer_seniorcitizen	customer_tenure	account_charges_monthly \
count	7032.000000	7032.000000	7032.000000
mean	0.162400	32.421786	64.798208
std	0.368844	24.545260	30.085974
min	0.000000	1.000000	18.250000
25%	0.000000	9.000000	35.587500
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.862500
max	1.000000	72.000000	118.750000

	account_charges_total
count	7032.000000
mean	2283.300441
std	2266.771362
min	18.800000
25%	401.450000
50%	1397.475000
75%	3794.737500
max	8684.800000

```
# plt.subplots(linhas, colunas, figsize=(largura, altura))
fig, axes = plt.subplots(1, 2, figsize=(12, 5)) # 1 linha, 2 colunas. Ajustei o tamanho total.
```

```
# --- GRÁFICO 1: Barras de Contagem ---
sns.countplot(data=dados_xtel_sem_nulos, x='churn', palette=cores, ax=axes[0]) # ax=axes[0] diz para desenhar no primeiro subplot
axes[0].set_title('Contagem de Clientes que cancelaram', fontsize=12)
axes[0].set_xlabel('', fontsize=10) # Use set_ para os eixos em subplots
axes[0].set_ylabel('Quantidade de Clientes', fontsize=12)
axes[0].tick_params(axis='x', labelsiz=10) # Ajustar o tamanho da fonte dos rótulos do eixo X
axes[0].tick_params(axis='y', labelsiz=10) # Ajustar o tamanho da fonte dos rótulos do eixo Y
axes[0].set_ylim(0,6000) # Manter a escala consistente
```

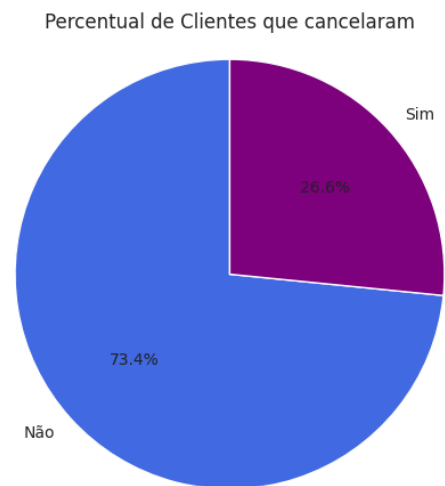
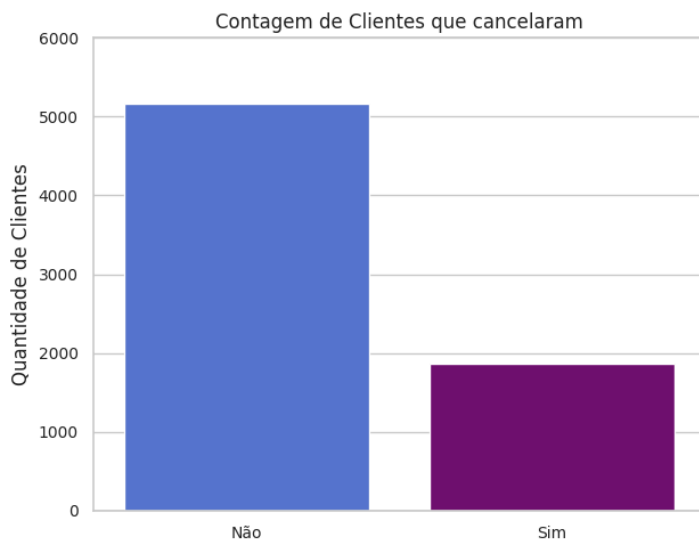
```
# --- GRÁFICO 2: Pizza de Percentual ---
axes[1].pie(contagem_e_percentual_churn,
            labels=contagem_e_percentual_churn.index,
```

```

autopct='%1.1f%%',
startangle=90,
colors=cores_pizza,
textprops={'fontsize': 10}) # Ajusta o tamanho da fonte dos percentuais nas fatias
axes[1].set_title('Percentual de Clientes que cancelaram', fontsize=12) # Use set_ para o título em subplots
axes[1].axis('equal') # Garante que o gráfico de pizza seja um círculo perfeito

plt.tight_layout() # Ajusta o layout para evitar sobreposição de elementos
plt.show()

```



```
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
```

```
import warnings
warnings.filterwarnings('ignore')
```

```

# --- GRÁFICO 1: Cancelamento por Tipo de Contrato ---
sns.countplot(data=dados_xtel_sem_nulos, x='account_contract', hue='churn', palette=cores, ax=axes[0])
axes[0].set_title('Cancelamento por Tipo de Contrato', fontsize=12)
axes[0].set_xlabel('Tipo de Contrato', fontsize=10)
axes[0].set_ylabel('Quantidade de Clientes', fontsize=10)

```

```

# Define o labelsizes para o eixo X do primeiro gráfico (8)
axes[0].tick_params(axis='x', labelsizes=8)
# Aplica rotação e alinhamento para os rótulos do eixo X do primeiro gráfico
plt.setp(axes[0].get_xticklabels(), rotation=0, ha='center')

```

```

# Mantém labelsizes=10 para o eixo Y do primeiro gráfico
axes[0].tick_params(axis='y', labelsizes=8)

axes[0].set_ylim(0, 3000)
axes[0].legend(title='Cancelou?', fontsize=10, title_fontsize=10)

```

```

# --- GRÁFICO 2: Cancelamento por Forma de Pagamento ---
sns.countplot(data=dados_xtel_sem_nulos, x='account_paymentmethod', hue='churn', palette=cores, ax=axes[1])
axes[1].set_title('Cancelamento por Forma de Pagamento', fontsize=12)
axes[1].set_xlabel('Forma de Pagamento', fontsize=10)
axes[1].set_ylabel('Quantidade de Clientes', fontsize=10)

```

```

# Correção aqui também:
# Define o labelsizes para o eixo X do segundo gráfico (10)
axes[1].tick_params(axis='x', labelsizes=8)
# Aplica rotação e alinhamento para os rótulos do eixo X do segundo gráfico
plt.setp(axes[1].get_xticklabels(), rotation=0, ha='center')

```

```

# Mantém labelsizes=10 para o eixo Y do segundo gráfico
axes[1].tick_params(axis='y', labelsizes=8)

axes[1].set_ylim(0, 3000)
axes[1].legend(title='Cancelou?', fontsize=10, title_fontsize=10)

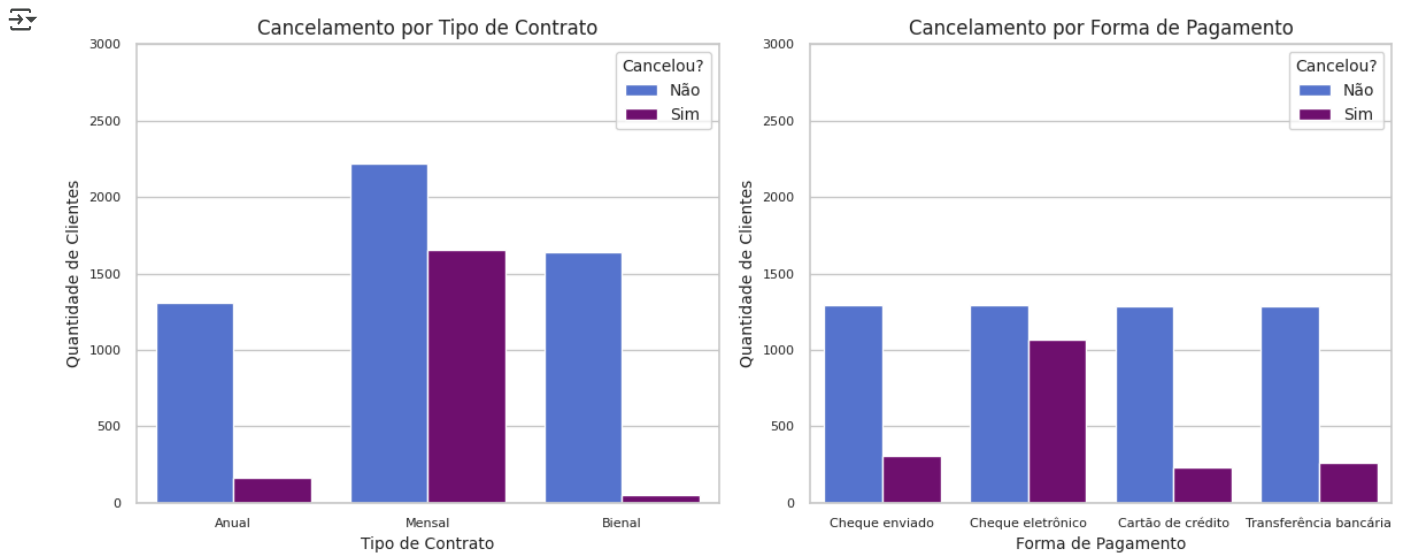
```

```

plt.tight_layout()
plt.show()

```





```
# --- Filtrar o DataFrame para incluir APENAS clientes que cancelaram ('Sim') ---
# Isso é crucial para que o gráfico de pizza mostre a distribuição SOMENTE dos cancelamentos
dados_cancelados = dados_xtel_sem_nulos[dados_xtel_sem_nulos['churn'] == 'Sim'].copy()

# --- Criação dos Subplots (1 linha, 2 colunas) ---
fig, axes = plt.subplots(1, 2, figsize=(12, 5)) # Aumentei um pouco o tamanho para as pizzas

# --- GRÁFICO 1: Percentual de Cancelamentos por Tipo de Contrato (Pizza) ---
contagem_contrato_cancelado = dados_cancelados['account_contract'].value_counts(normalize=True) * 100
cores_contrato_cancelado = [cores[0], cores[1], cores[2]] # Assumindo que você tem 3 cores para os 3 tipos de contrato

axes[0].pie(contagem_contrato_cancelado,
            labels=contagem_contrato_cancelado.index,
            autopct='%1.1f%%', # Mostra o percentual com uma casa decimal
            startangle=90,
            colors=cores_contrato_cancelado,
            textprops={'fontsize': 10}) # Ajusta o tamanho da fonte dos percentuais

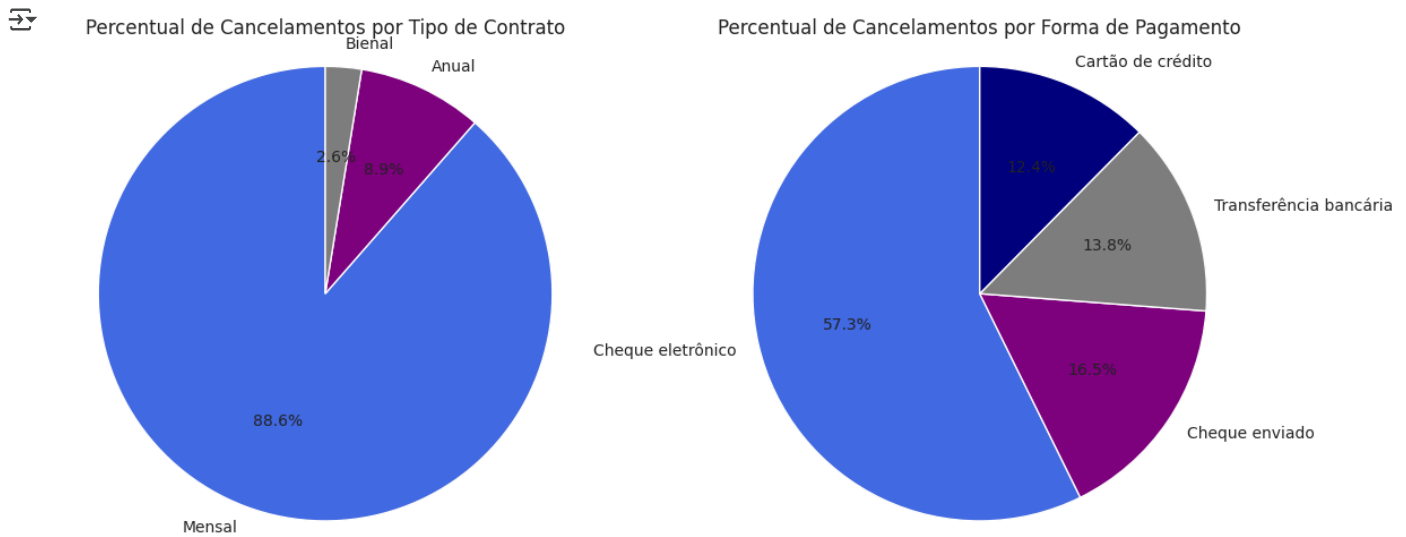
axes[0].set_title('Percentual de Cancelamentos por Tipo de Contrato', fontsize=12)
axes[0].axis('equal') # Garante que o círculo seja perfeito

# --- GRÁFICO 2: Percentual de Cancelamentos por Forma de Pagamento (Pizza) ---
contagem_pagamento_cancelado = dados_cancelados['account_paymentmethod'].value_counts(normalize=True) * 100
cores_pagamento_cancelado = [cores[0], cores[1], cores[2], cores[3]] # Assumindo que você tem 4 cores para 4 formas de pagamento

axes[1].pie(contagem_pagamento_cancelado,
            labels=contagem_pagamento_cancelado.index,
            autopct='%1.1f%%',
            startangle=90,
            colors=cores_pagamento_cancelado,
            textprops={'fontsize': 10})

axes[1].set_title('Percentual de Cancelamentos por Forma de Pagamento', fontsize=12)
axes[1].axis('equal') # Garante que o círculo seja perfeito

plt.tight_layout() # Ajusta o layout para evitar sobreposição
plt.show()
```



```
# --- 1. Filtrar o DataFrame para incluir APENAS clientes que cancelaram ('Sim') ---
# Isso é para o gráfico de pizza que vai mostrar a proporção de gênero DENTRO dos cancelamentos
dados_cancelados = dados_xtel_sem_nulos[dados_xtel_sem_nulos['churn'] == 'Sim'].copy()

# --- 2. Calcular o percentual de cancelamentos por gênero para o gráfico de pizza ---
# Conta a frequência de cada gênero APENAS entre os cancelados e normaliza para percentual
contagem_genero_cancelado = dados_cancelados['customer_gender'].value_counts(normalize=True) * 100

# --- NOVAS CORES PARA O GRÁFICO DE PIZZA DE GÊNERO ---
# Definindo as cores diretamente como strings (nomes de cores ou códigos hexadecimais)
# Assumindo que a ordem da contagem_genero_cancelado.index será 'Feminino', 'Masculino' (ou similar)
# Verifique a saída de 'print(contagem_genero_cancelado.index)' se tiver dúvidas sobre a ordem
cores_pizza_genero = ['#CCA1FF', '#88ABFD'] # 'HotPink' para Feminino, 'Gray' para Masculino

# Se a ordem for 'Masculino', 'Feminino', você inverteria:
# cores_pizza_genero = ['#808080', '#FF69B4']

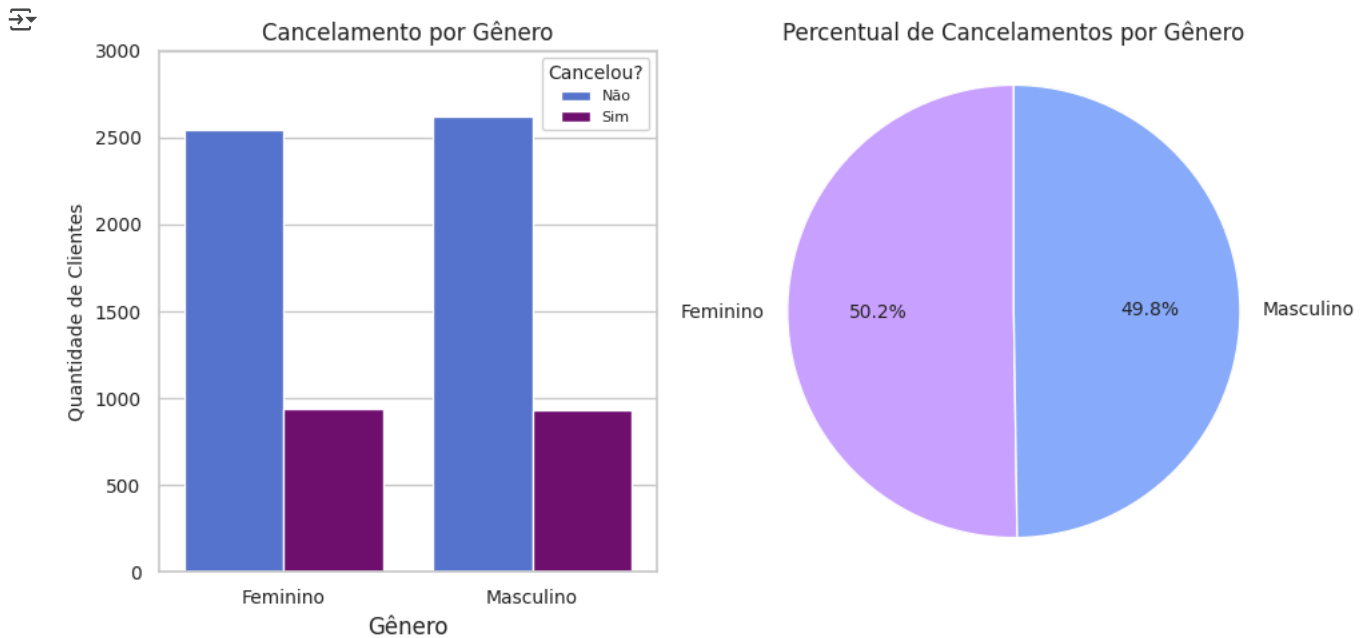
# --- 3. Criação dos Subplots (1 linha, 2 colunas) ---
fig, axes = plt.subplots(1, 2, figsize=(10, 5)) # Tamanho ajustado para ambos os gráficos

# --- GRÁFICO 1 (Esquerda): Contagem de Clientes por Gênero e Churn (Barras) ---
sns.countplot(data=dados_xtel_sem_nulos, x='customer_gender', hue='churn', palette=cores, ax=axes[0])
axes[0].set_title('Cancelamento por Gênero', fontsize=12) # Título maior para legibilidade
axes[0].set_xlabel('Gênero', fontsize=12)
axes[0].set_ylabel('Quantidade de Clientes', fontsize=10)
axes[0].tick_params(axis='x', labelsize=10) # Rótulos do eixo X (Male/Female)
axes[0].tick_params(axis='y', labelsize=10)
axes[0].set_ylim(0, 3000) # Mantém a escala original
axes[0].legend(title='Cancelou?', fontsize=8, title_fontsize=10)

# --- GRÁFICO 2 (Direita): Percentual de Cancelamentos por Gênero (Pizza) ---
axes[1].pie(contagem_genero_cancelado,
            labels=contagem_genero_cancelado.index, # Rótulos das fatias (Gêneros)
            autopct='%1.1f%%', # Formato para exibir o percentual (ex: 50.5%)
            startangle=90, # Onde a primeira fatia começa
            colors=cores_pizza_genero, # Cores para as fatias
            textprops={'fontsize': 10}) # Tamanho da fonte dos percentuais na fatia

axes[1].set_title('Percentual de Cancelamentos por Gênero', fontsize=12)
axes[1].axis('equal') # Garante que o gráfico de pizza seja um círculo perfeito

plt.tight_layout() # Ajusta o layout para evitar sobreposição
plt.show()
```



```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

# --- Importante: Certifique-se de que 'dados_xtel_sem_nulos'
# --- está limpo e que a sua lista 'cores' está definida.
# --- (Ex: cores = ['royalblue', 'purple'])
cores = ['royalblue', 'purple'] # Definindo cores de exemplo para 'Não' e 'Sim'

# --- Criação dos Subplots (1 linha, 2 colunas) ---
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

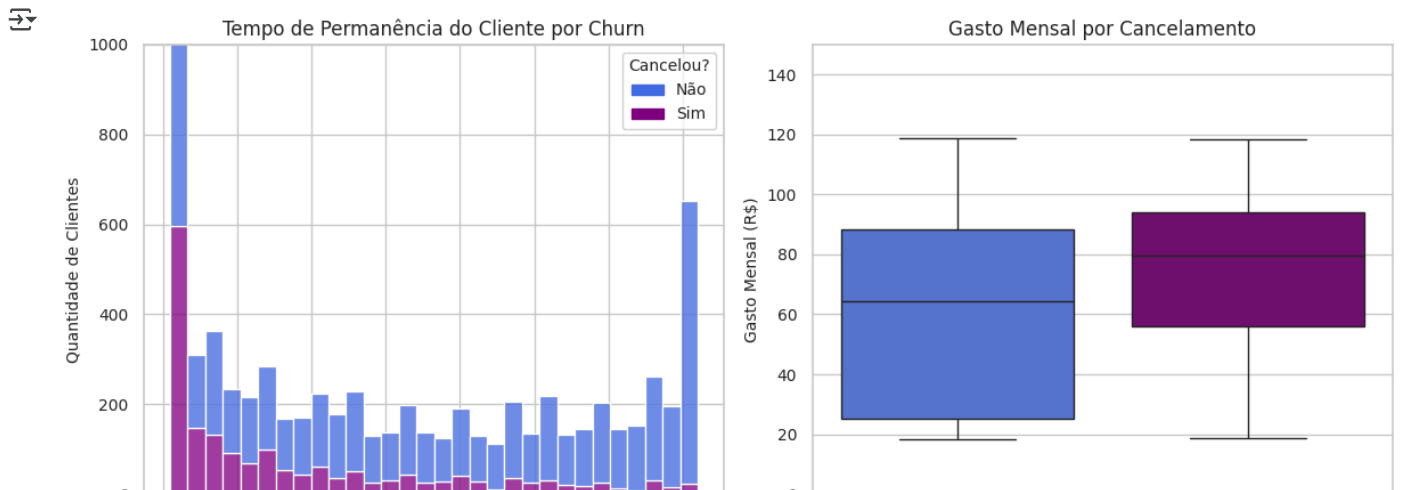
# --- GRÁFICO 1 (Esquerda): Histograma de Tempo de Permanência do Cliente ---
sns.histplot(data=dados_xtel_sem_nulos, x='customer_tenure', bins=30, hue='churn',
             multiple='stack', palette=cores, ax=axes[0])
axes[0].set_title('Tempo de Permanência do Cliente por Churn', fontsize=12)
axes[0].set_xlabel('Meses como Cliente', fontsize=10)
axes[0].set_ylabel('Quantidade de Clientes', fontsize=10)
axes[0].tick_params(axis='x', labelsize=10)
axes[0].tick_params(axis='y', labelsize=10)
axes[0].set_ylim(0, 1000) # Mantém o limite que você já tinha definido

# --- CORREÇÃO DA LEGENDA: Adicionando manualmente para garantir ---
# Criar os 'patches' (pequenos retângulos de cor) para a legenda
patch_nao = plt.matplotlib.patches.Patch(color=cores[0], label='Não')
patch_sim = plt.matplotlib.patches.Patch(color=cores[1], label='Sim')

# Adicionar a legenda ao eixo, passando os patches
axes[0].legend(handles=[patch_nao, patch_sim], title='Cancelou?', fontsize=10, title_fontsize=10)

# --- GRÁFICO 2 (Direita): Boxplot de Gasto Mensal por Cancelamento ---
sns.boxplot(data=dados_xtel_sem_nulos, x='churn', y='account_charges_monthly',
            palette=cores, ax=axes[1])
axes[1].set_title('Gasto Mensal por Cancelamento', fontsize=12)
axes[1].set_xlabel('Cancelou?', fontsize=10)
axes[1].set_ylabel('Gasto Mensal (R$)', fontsize=10)
axes[1].tick_params(axis='x', labelsize=10)
axes[1].tick_params(axis='y', labelsize=10)
axes[1].set_ylim(0, 150) # Mantém o limite que você já tinha definido

plt.tight_layout() # Ajusta o layout para evitar sobreposição
plt.show()
```



```
plt.figure(figsize=(8, 5))
sns.heatmap(df_numericas_para_heatmap.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.xticks(fontsize=8) # Ajuste o valor '8' para o tamanho desejado
plt.yticks(fontsize=8, rotation=0) # Ajuste o valor '8' para o tamanho desejado e rotation=0 para deixá-los na horizontal
plt.title('Correlação entre Variáveis Numéricas')
plt.show()
```

