

```
In [48]: # Libraries for Extract, Treat and Load
import numpy as np, pandas as pd

# Libraries for Data Visualization
import matplotlib.pyplot as plt, seaborn as sns, plotly.express as px

# Libraries for Data Science
import statsmodels.api as sm
```

☒ Objetivo do Projeto Você deve investigar os fatores que influenciam o total de vendas (vendas) e responder à seguinte pergunta:

"Quais variáveis estão mais associadas com o desempenho de vendas mensal das lojas?"

✓ Tarefas Esperadas Análise Exploratória dos Dados (EDA):

- Histograma ou boxplot das variáveis numéricas.
- Gráfico de linha da variação das vendas ao longo dos meses.
- Comparações entre regiões (ex: vendas médias por região com seaborn.barplot).
- Correlação entre as variáveis (heatmap).

🔍 Visualizações com plotly express:

- Gráfico de dispersão entre clientes e vendas, com cores por região (use px.scatter).
- Linha de tendência com trendline='ols' para visualização de regressão.

✓ Modelo de Regressão Linear com statsmodels:

- Faça a regressão de vendas como variável dependente.
- Inclua como variáveis independentes: clientes, descontos, campanha\_marketing, preco\_medio\_produto, indice\_satisfacao, regiao (usar dummies)

Interprete os coeficientes e valores-p.

Analise os resíduos.

Conclusão:

- Quais variáveis são estatisticamente significantes?
- Como a empresa pode usar essas informações para aumentar suas vendas?

📌 Exemplo de Gráficos Esperados

- histograma: distribuição das vendas
- boxplot: comparação das vendas por região
- scatterplot: vendas vs. clientes com cores por região

- lineplot: tendência de vendas ao longo dos meses
- heatmap: correlação entre variáveis numéricas

```
In [49]: # Importando dados
FactSales = pd.read_parquet('Sales.parquet')

# Lendo o cabeçalho da base de dados
FactSales.head()
```

Out[49]:

	<b>IdLoja</b>	<b>MesAnoVenda</b>	<b>Regiao</b>	<b>Clientes</b>	<b>DescontoPercent</b>	<b>CampanhaMarketing</b>	<b>Pre</b>
<b>0</b>	1	2024-01	Sudeste	1475	5.0		1
<b>1</b>	1	2024-02	Sudeste	689	12.0		0
<b>2</b>	1	2024-03	Sudeste	1457	23.0		0
<b>3</b>	1	2024-04	Sudeste	1186	14.0		0
<b>4</b>	1	2024-05	Sudeste	1457	30.0		0

◀ ▶

```
In [50]: # Verificando se há valores nulos
```

```
print(FactSales.isnull().sum(), FactSales.isna().sum(), sep='-----')
```

```

IdLoja          0
MesAnoVenda     0
Regiao          0
Clientes        0
DescontoPercent 0
CampanhaMarketing 0
PrecoMedio       0
IndiceSatisfacao 0
Vendas          0
dtype: int64
-----
IdLoja          0
MesAnoVenda     0
Regiao          0
Clientes        0
DescontoPercent 0
CampanhaMarketing 0
PrecoMedio       0
IndiceSatisfacao 0
Vendas          0
dtype: int64
```

```
In [51]: # Resumo informativo da tabela
```

```
FactSales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 720 entries, 0 to 719
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   IdLoja            720 non-null    int64  
 1   MesAnoVenda       720 non-null    object  
 2   Regiao             720 non-null    object  
 3   Clientes           720 non-null    int32  
 4   DescontoPercent   720 non-null    float64 
 5   CampanhaMarketing 720 non-null    int32  
 6   PrecoMedio          720 non-null    float64 
 7   IndiceSatisfacao  720 non-null    float64 
 8   Vendas              720 non-null    float64 
dtypes: float64(4), int32(2), int64(1), object(2)
memory usage: 45.1+ KB
```

```
In [52]: from cycler import cycler

cores = plt.get_cmap('Pastel1').colors

FactSales.columns
```

```
Out[52]: Index(['IdLoja', 'MesAnoVenda', 'Regiao', 'Clientes', 'DescontoPercent',
               'CampanhaMarketing', 'PrecoMedio', 'IndiceSatisfacao', 'Vendas'],
               dtype='object')
```

```
In [53]: colors = px.colors.qualitative.Pastel

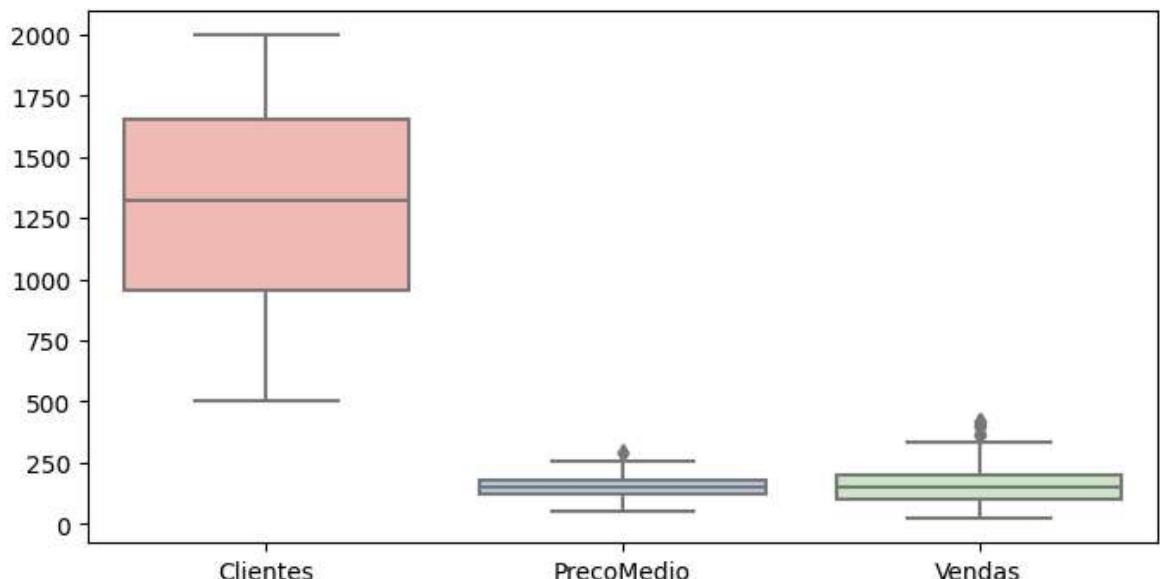
fig = px.box(FactSales[['Clientes', 'DescontoPercent', 'CampanhaMarketing', 'PrecoMedio'],
                       color_discrete_sequence=colors
                      ])

fig.show()
```

```
In [54]: plt.figure(figsize=(8, 4))

sns.boxplot(FactSales[['Clientes', 'PrecoMedio', 'Vendas']])
```

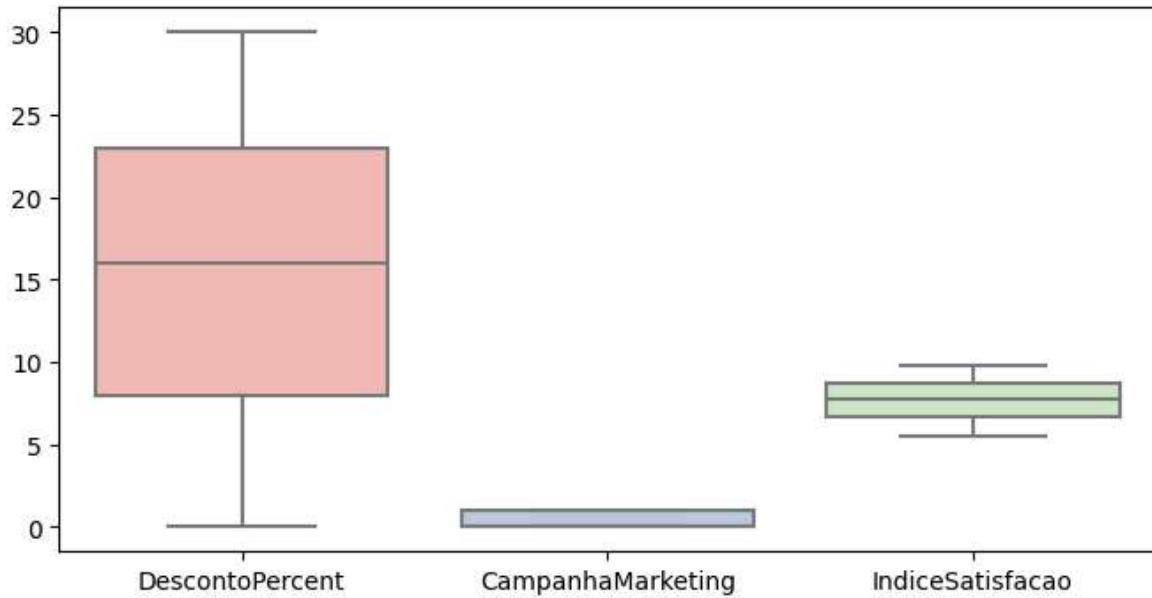
```
Out[54]: <Axes: >
```



```
In [55]: plt.figure(figsize=(8, 4))

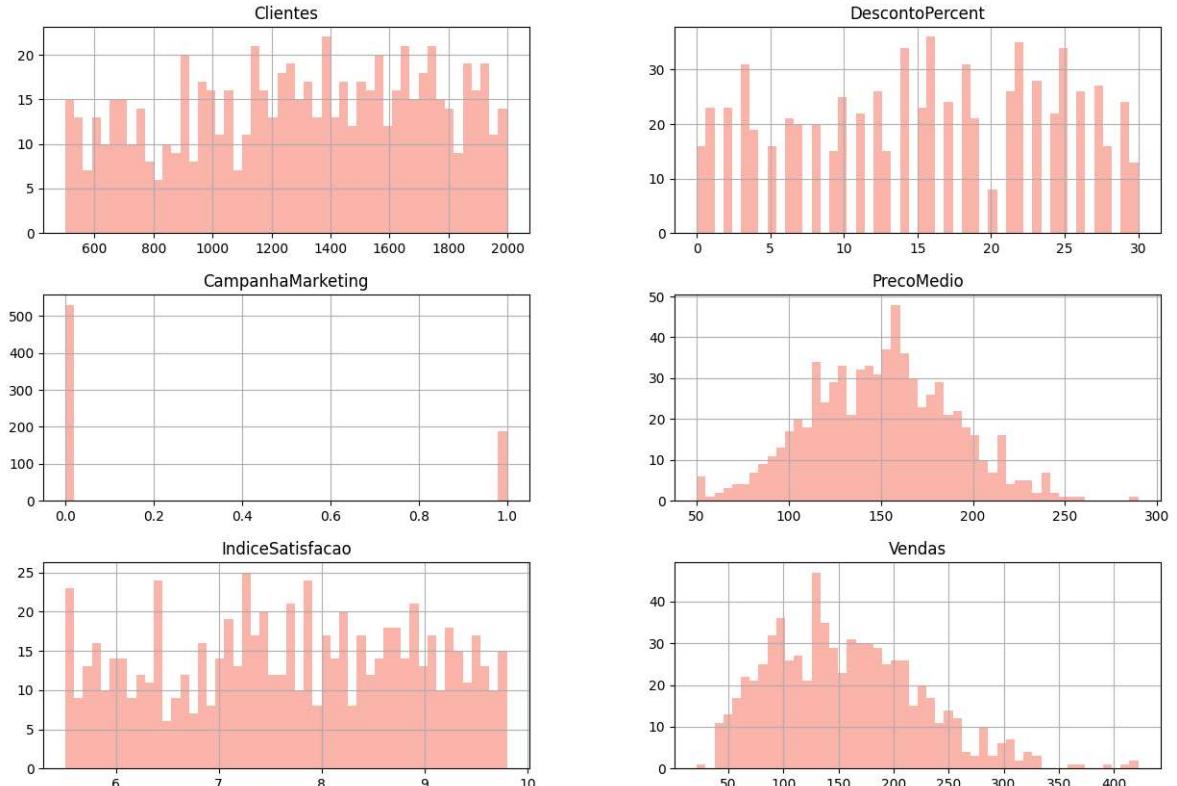
sns.boxplot(FactSales[['DescontoPercent', 'CampanhaMarketing', 'IndiceSatisfacao']])
```

Out[55]: <Axes: >



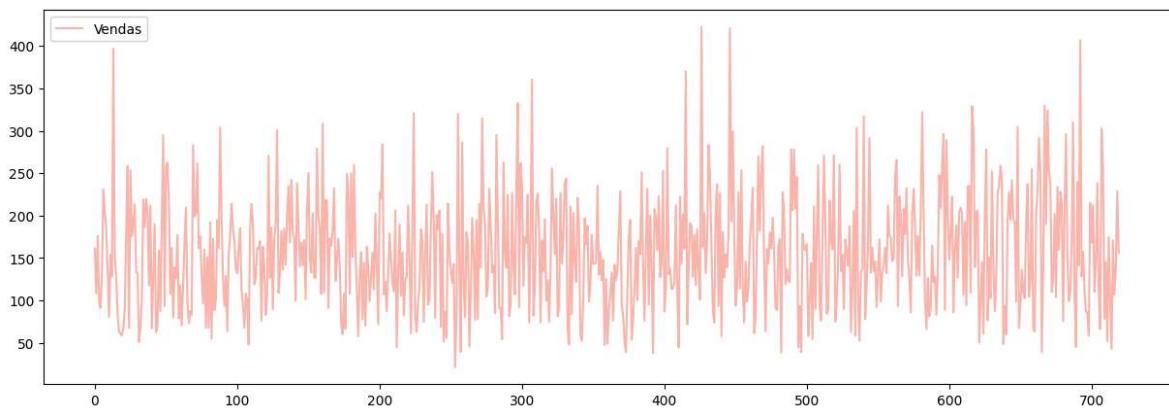
```
In [56]: FactSales[['Clientes', 'DescontoPercent', 'CampanhaMarketing', 'PrecoMedio', 'IndiceSatisfacao']]
```

```
Out[56]: array([['Clientes'],
   ['DescontoPercent'],
   ['CampanhaMarketing'],
   ['PrecoMedio'],
   ['IndiceSatisfacao'],
   ['Vendas']], dtype=object)
```



```
In [57]: FactSales[['MesAnoVenda', 'Vendas']].plot(kind='line', figsize=(15,5))
```

Out[57]: &lt;Axes: &gt;



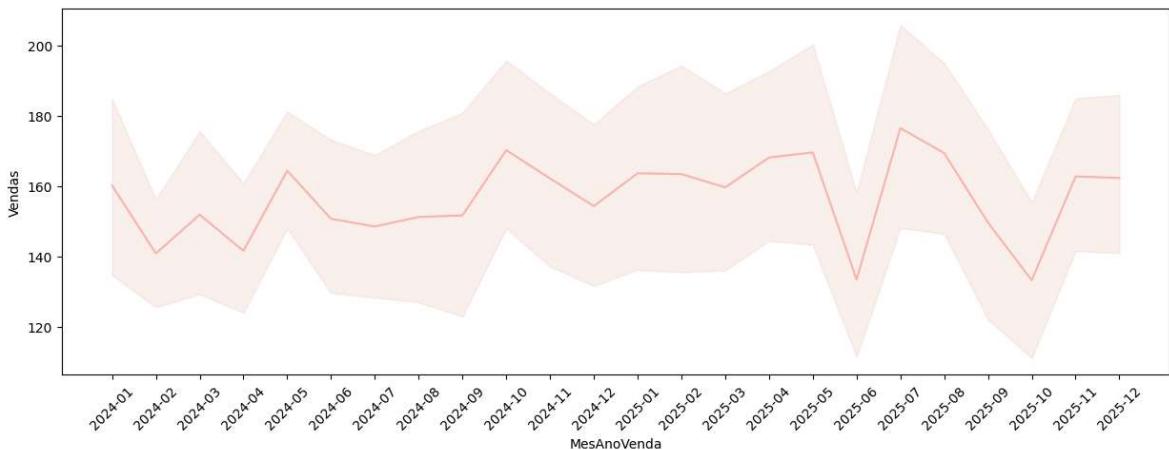
```
In [58]: plt.figure(figsize=(15,5))
sns.lineplot(FactSales[['MesAnoVenda', 'Vendas']], x='MesAnoVenda', y='Vendas')
plt.tick_params(axis='x', labelrotation=45)
plt.show()
```

c:\Users\adrya\anaconda3\Lib\site-packages\seaborn\\_oldcore.py:1119: FutureWarning:

use\_inf\_as\_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.

c:\Users\adrya\anaconda3\Lib\site-packages\seaborn\\_oldcore.py:1119: FutureWarning:

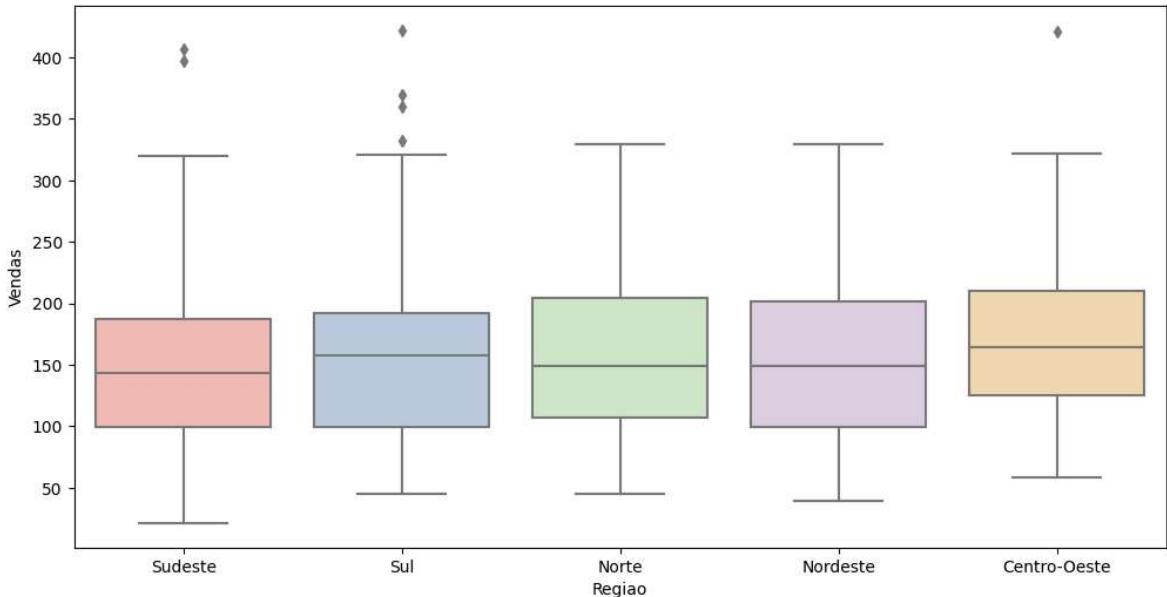
use\_inf\_as\_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.



```
In [59]: plt.rc('axes', prop_cycle=cycler('color', cores))
```

```
plt.figure(figsize=(12,6))
sns.boxplot(data=FactSales[['Vendas', 'Regiao']], x='Regiao', y='Vendas')
```

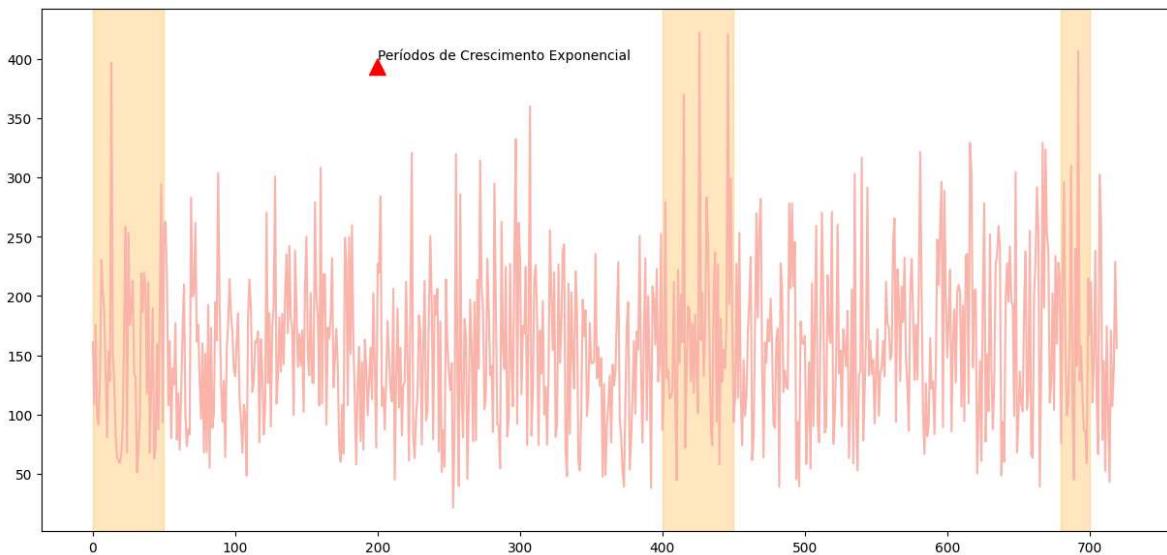
Out[59]: &lt;Axes: xlabel='Regiao', ylabel='Vendas'&gt;



```
In [60]: colors = px.colors.qualitative.Pastel
fig = px.box(FactSales[['Vendas', 'Regiao']], x='Regiao', y='Vendas', color_discrete_sequence=colors)

fig.update_yaxes(tickprefix="R$ ", showgrid=True)
```

```
In [61]: plt.figure(figsize=(15,7))
plt.plot(FactSales['Vendas'])
plt.axvspan(0, 50, alpha=0.25, color='orange')
plt.axvspan(400, 450, alpha=0.25, color='orange')
plt.axvspan(680, 700, alpha=0.25, color='orange')
plt.annotate('Períodos de Crescimento Exponencial', xy=(200,400), arrowprops={'color': 'red', 'arrowheadstyle': 'triangle-down', 'width': 2}, color='red')
plt.show()
```



```
In [62]: colors = px.colors.sequential.Cividis

fig = px.line(FactSales['Vendas'], color_discrete_sequence=colors)

fig.add_shape(
    type="rect",
    x0=400, x1=600, # intervalo no eixo x
    y0=0, y1=1, # cobre toda a altura do gráfico
    xref='x', yref='paper', # 'paper' faz com que vá de 0 a 1 no eixo y
```

```
        fillcolor="LightSalmon",
        opacity=0.3,
        layer="below",
        line_width=3,
    )

fig.show()
```

```
In [63]: # 'Lowess', 'rolling', 'ewm', 'expanding', 'ols'

colors = px.colors.qualitative.Safe

fig = px.scatter(
    FactSales[['Clientes', 'Vendas', 'Regiao']],
    x='Clientes',
    y='Vendas',
    color='Regiao',
    color_discrete_sequence=colors,
    trendline='ols',
    animation_frame='Regiao'
)

fig.layout.updatemenus[0].buttons[0].args[1]['frame']['duration'] = 2000
fig.layout.updatemenus[0].buttons[0].args[1]['transition']['duration'] = 5000

fig.show()
```

```
In [64]: # 'Lowess', 'rolling', 'ewm', 'expanding', 'ols'

colors = px.colors.qualitative.Safe

fig = px.scatter(
    FactSales[['Clientes', 'Vendas', 'Regiao']],
    x='Clientes',
    y='Vendas',
    color='Regiao',
    color_discrete_sequence=colors,
    trendline='lowess',
    animation_frame='Regiao'
)

fig.layout.updatemenus[0].buttons[0].args[1]['frame']['duration'] = 2000
fig.layout.updatemenus[0].buttons[0].args[1]['transition']['duration'] = 5000

fig.show()
```

```
In [65]: BaseAuxiliar = FactSales[['Vendas', 'Regiao', 'Clientes', 'DescontoPercent', 'Carro']]
BaseAuxiliar = pd.get_dummies(BaseAuxiliar, dtype=int, drop_first=True)

BaseAuxiliar.corr()
```

Out[65]:

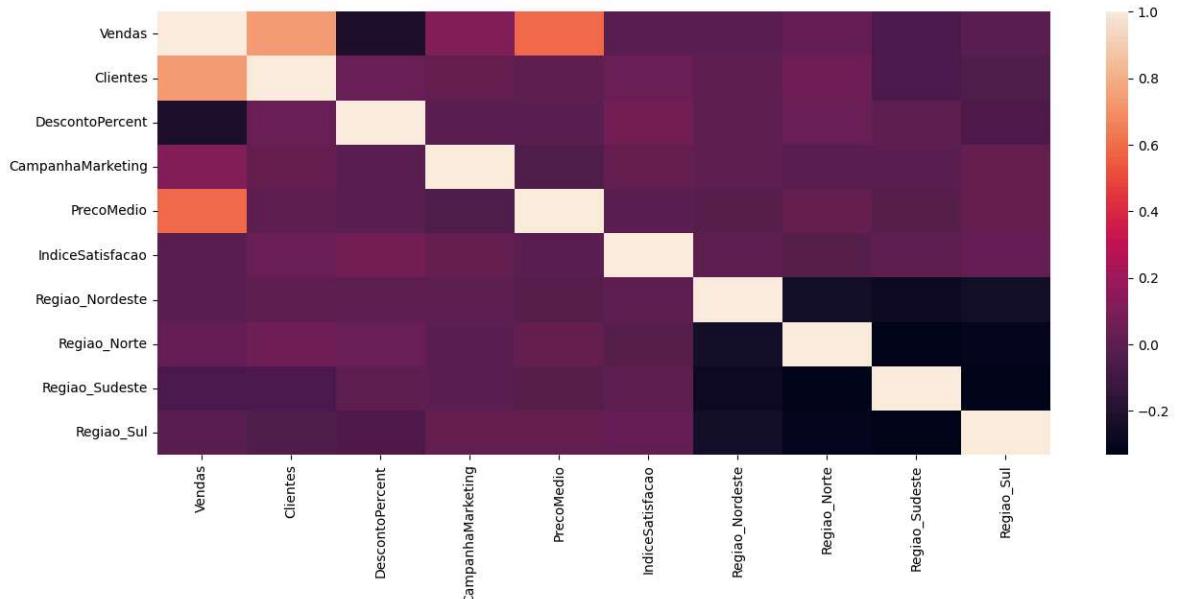
	Vendas	Clientes	DescontoPercent	CampanhaMarketing	Precc
Vendas	1.000000	0.729536	-0.213872	0.104656	0.
Clientes	0.729536	1.000000	0.030508	0.017694	-0.
DescontoPercent	-0.213872	0.030508	1.000000	-0.008279	-0.
CampanhaMarketing	0.104656	0.017694	-0.008279	1.000000	-0.
PrecoMedio	0.592785	-0.000937	-0.016978	-0.044454	1.
IndiceSatisfacao	-0.011278	0.028239	0.058259	0.012872	-0.
Regiao_Nordeste	-0.007910	0.005422	0.001219	-0.004235	-0.
Regiao_Norte	0.023442	0.043178	0.031233	-0.015673	0.
Regiao_Sudeste	-0.056867	-0.058158	0.000774	-0.017132	-0.
Regiao_Sul	-0.007873	-0.036234	-0.051810	0.014180	0.



In [66]: plt.figure(figsize=(15, 6))

sns.heatmap(BaseAuxiliar.corr())

plt.show()



In [67]: y = BaseAuxiliar['Vendas']

X = sm.add\_constant(BaseAuxiliar.drop(columns='Vendas'))

modelo1 = sm.OLS(y, X)

resultado1 = modelo1.fit()

resultado1.summary()

Out[67]:

## OLS Regression Results

<b>Dep. Variable:</b>	Vendas	<b>R-squared:</b>	0.950				
<b>Model:</b>	OLS	<b>Adj. R-squared:</b>	0.949				
<b>Method:</b>	Least Squares	<b>F-statistic:</b>	1485.				
<b>Date:</b>	Mon, 04 Aug 2025	<b>Prob (F-statistic):</b>	0.00				
<b>Time:</b>	19:43:57	<b>Log-Likelihood:</b>	-2987.6				
<b>No. Observations:</b>	720	<b>AIC:</b>	5995.				
<b>Df Residuals:</b>	710	<b>BIC:</b>	6041.				
<b>Df Model:</b>	9						
<b>Covariance Type:</b>	nonrobust						
		coef	std err	t	P> t	[0.025	0.975]
<b>const</b>	-129.2326	5.149	-25.097	0.000	-139.342	-119.123	
<b>Clientes</b>	0.1197	0.001	86.865	0.000	0.117	0.122	
<b>DescontoPercent</b>	-1.7681	0.067	-26.489	0.000	-1.899	-1.637	
<b>CampanhaMarketing</b>	18.1179	1.311	13.818	0.000	15.544	20.692	
<b>PrecoMedio</b>	1.0485	0.015	70.418	0.000	1.019	1.078	
<b>IndiceSatisfacao</b>	-0.8317	0.475	-1.752	0.080	-1.764	0.100	
<b>Regiao_Nordeste</b>	2.8235	2.311	1.222	0.222	-1.713	7.360	
<b>Regiao_Norte</b>	0.1320	2.180	0.061	0.952	-4.148	4.412	
<b>Regiao_Sudeste</b>	2.3932	2.147	1.115	0.265	-1.823	6.609	
<b>Regiao_Sul</b>	0.8317	2.186	0.381	0.704	-3.459	5.122	
<b>Omnibus:</b>	137.075	<b>Durbin-Watson:</b>	2.124				
<b>Prob(Omnibus):</b>	0.000	<b>Jarque-Bera (JB):</b>	579.421				
<b>Skew:</b>	0.813	<b>Prob(JB):</b>	1.51e-126				
<b>Kurtosis:</b>	7.083	<b>Cond. No.</b>	1.32e+04				

Notes:

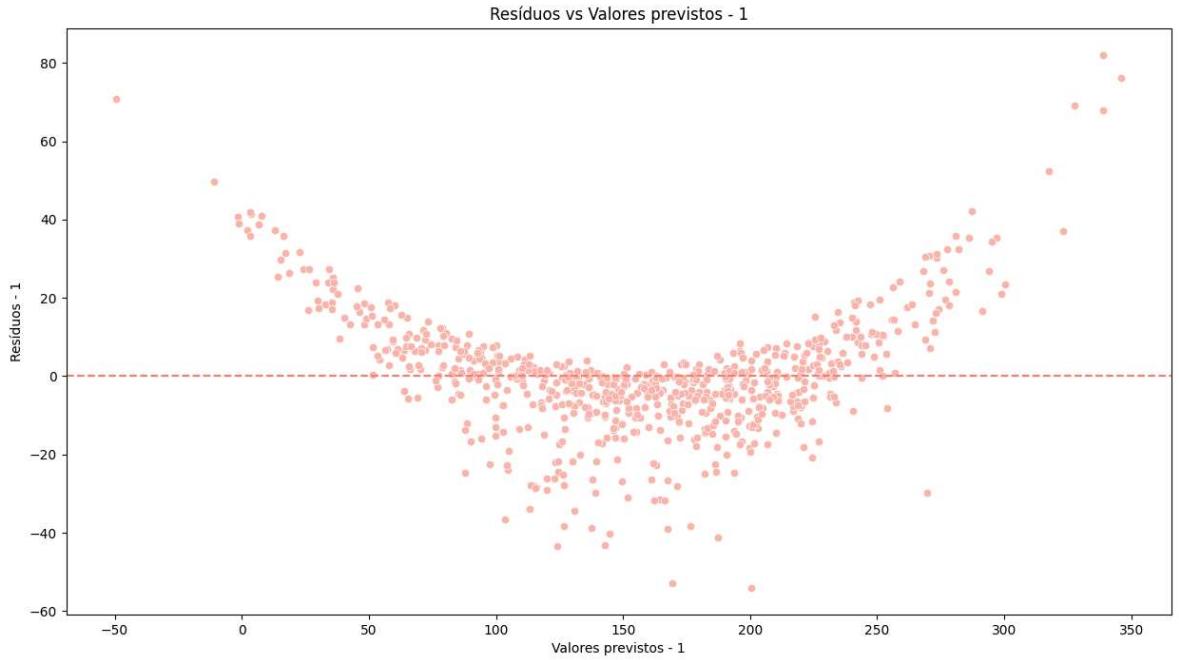
- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 1.32e+04. This might indicate that there are strong multicollinearity or other numerical problems.

In [111...]

```
residuos1 = resultado1.resid
previstos1 = resultado1.fittedvalues
plt.figure(figsize=(15,8))
```

```
sns.scatterplot(x=previstos1, y=residuos1)
plt.axhline(y=0, color='salmon', linestyle='--')
plt.xlabel('Valores previstos - 1')
plt.ylabel('Resíduos - 1')
plt.title('Resíduos vs Valores previstos - 1')

plt.show()
```



```
In [69]: y = BaseAuxiliar['Vendas']

X = sm.add_constant(BaseAuxiliar.drop(columns=['Vendas', 'IndiceSatisfacao', 'Re'])

modelo2 = sm.OLS(y, X)

resultado2 = modelo2.fit()

resultado2.summary()
```

Out[69]:

## OLS Regression Results

<b>Dep. Variable:</b>	Vendas	<b>R-squared:</b>	0.949			
<b>Model:</b>	OLS	<b>Adj. R-squared:</b>	0.949			
<b>Method:</b>	Least Squares	<b>F-statistic:</b>	3331.			
<b>Date:</b>	Mon, 04 Aug 2025	<b>Prob (F-statistic):</b>	0.00			
<b>Time:</b>	19:43:57	<b>Log-Likelihood:</b>	-2991.0			
<b>No. Observations:</b>	720	<b>AIC:</b>	5992.			
<b>Df Residuals:</b>	715	<b>BIC:</b>	6015.			
<b>Df Model:</b>	4					
<b>Covariance Type:</b>	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
<b>const</b>	-133.6334	3.117	-42.870	0.000	-139.753	-127.513
<b>Clientes</b>	0.1195	0.001	87.026	0.000	0.117	0.122
<b>DescontoPercent</b>	-1.7764	0.067	-26.670	0.000	-1.907	-1.646
<b>CampanhaMarketing</b>	18.0425	1.312	13.756	0.000	15.467	20.618
<b>PrecoMedio</b>	1.0469	0.015	70.400	0.000	1.018	1.076
<b>Omnibus:</b>	137.584	<b>Durbin-Watson:</b>	2.112			
<b>Prob(Omnibus):</b>	0.000	<b>Jarque-Bera (JB):</b>	571.329			
<b>Skew:</b>	0.822	<b>Prob(JB):</b>	8.66e-125			
<b>Kurtosis:</b>	7.042	<b>Cond. No.</b>	7.43e+03			

Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 7.43e+03. This might indicate that there are strong multicollinearity or other numerical problems.

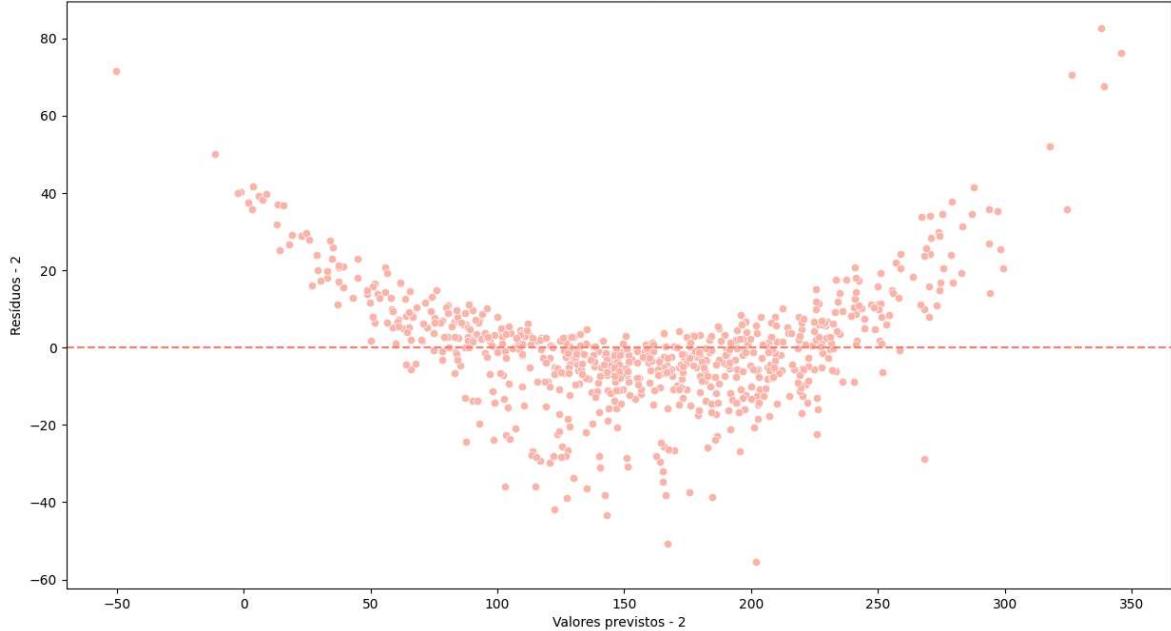
In [110...]

```
residuos2 = resultado2.resid

previstos2 = resultado2.fittedvalues

plt.figure(figsize=(15,8))
sns.scatterplot(x=previstos2, y=residuos2)
plt.axhline(y=0, color='salmon', linestyle='--')
plt.xlabel('Valores previstos - 2')
plt.ylabel('Resíduos - 2')
plt.title('Resíduos vs Valores previstos - 2')
plt.show()
```

Resíduos vs Valores previstos - 2



```
In [109]: fig = px.scatter(x=previstos2,
                      y=residuos2,
                      labels={'x':'Previstos - 2', 'y':'Resíduos - 2'},
                      title='Resíduos vs Valores previstos - 2'
                     )

fig.add_hline(y=0,
              line={'dash': 'dash',
                    }
                 )

fig.show()
```

Conclusão:

- Quais variáveis são estatisticamente significantes?

Resposta: As variáveis mais significantes são CampanhaMarketing com coeficiente 18.0425, DescontoPercent com coeficiente -1.7764, PrecoMedio com coeficiente 1.0469 e com o ajuste realizado nos modelos acima, o modelo 2 se configura como mais confiável, que apesar do mesmo R<sup>2</sup> de 0.949, desconsidera as variáveis irrelevantes.

- Como a empresa pode usar essas informações para aumentar suas vendas?

Resposta: A empresa poderá direcionar seus esforços em mais campanhas de marketing, uma vez que tem um potencial maior na atração de clientes e efetivação de vendas, ainda, é possível dedicar tempo na elaboração de políticas melhores de desconto e até cross-selling, tendo em vista que os descontos oferecidos podem estar tempo um impacto negativo nas vendas.