


```
# importando os comandos
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#importa o arquivo do projeto
from google.colab import files
```

```
uploaded=files.upload()
filename=next(iter(uploaded))
df=pd.read_excel(filename)
```


 Escolher Ficheiros

projetointegrador.xlsx

- projetointegrador.xlsx(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 12093 bytes, last modified: 13/05/2024 - 100% done

Saving projetointegrador.xlsx to projetointegrador.xlsx

```
# cabecalho do dataset
df.head(24)
```




	ID do Mês	Quantidade de Vendas	ID do Cliente	Custo de Operação	Valor Total de Vendas	Lucro Total das Vendas	Unnamed: 6	Unnamed: 7
0	2022-01-01	150	120	5000	35000	15000	NaN	22833.333333
1	2022-02-01	200	140	5500	40000	18000	NaN	NaN
2	2022-03-01	250	160	6000	45000	20000	NaN	NaN
3	2022-04-01	300	180	6500	50000	22000	NaN	NaN
4	2022-05-01	350	200	7000	55000	25000	NaN	NaN
5	2022-06-01	400	220	7500	60000	28000	NaN	NaN
6	2022-07-01	450	240	8000	55000	26000	NaN	NaN
7	2022-08-01	500	260	8500	53000	27000	NaN	NaN
8	2022-09-01	450	240	8000	52000	25000	NaN	NaN
9	2022-10-01	400	220	7500	49000	23000	NaN	NaN
10	2022-11-01	350	200	7000	47000	22000	NaN	NaN
11	2022-12-01	300	180	6500	45000	21000	NaN	NaN
12	2023-01-01	250	160	6000	43000	20000	NaN	NaN
13	2023-02-01	200	140	5500	40000	18000	NaN	NaN

Próximas etapas:

 Gerar código com df

 Ver gráficos recomendados

```
#estatísticas descritivas, como média, desvio padrão, máximo, mínimo e outras tendências centrais
print(df.describe())
```



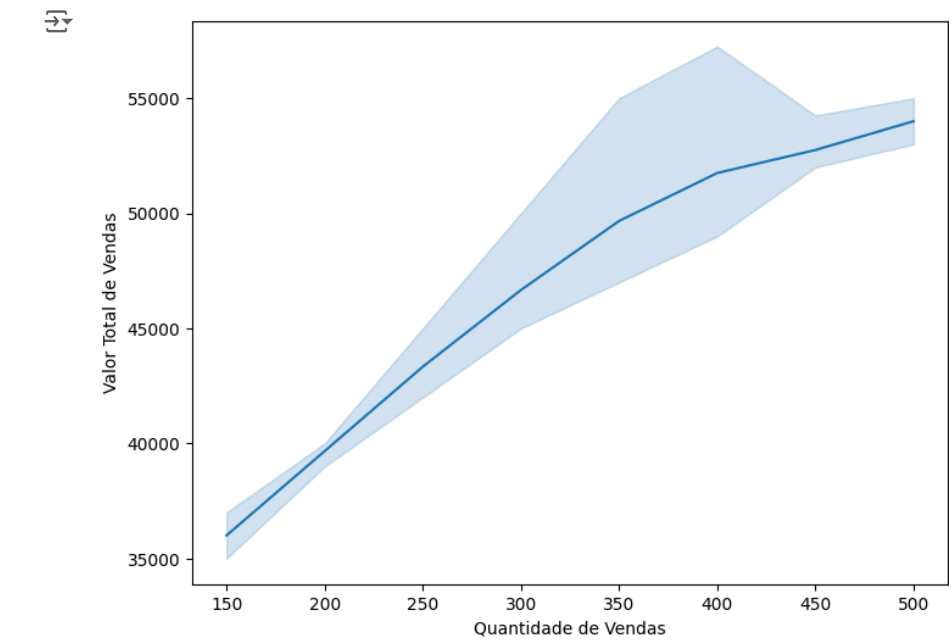
	ID do Mês	Quantidade de Vendas	ID do Cliente \
count	24	24.000000	24.000000
mean	2022-12-16 00:00:00	333.333333	193.333333
min	2022-01-01 00:00:00	150.000000	120.000000
25%	2022-06-23 12:00:00	250.000000	160.000000
50%	2022-12-16 12:00:00	350.000000	200.000000
75%	2023-06-08 12:00:00	412.500000	225.000000
max	2023-12-01 00:00:00	500.000000	260.000000
std	NaN	109.014026	43.605611

	Custo de Operação	Valor Total de Vendas	Lucro Total das Vendas \
count	24.000000	24.000000	24.000000

mean	6833.333333	47333.333333	22833.333333
min	5000.000000	35000.000000	15000.000000
25%	6000.000000	42750.000000	20000.000000
50%	7000.000000	48000.000000	22500.000000
75%	7625.000000	52000.000000	26000.000000
max	8500.000000	60000.000000	30000.000000
std	1090.140265	6424.590440	4156.364074

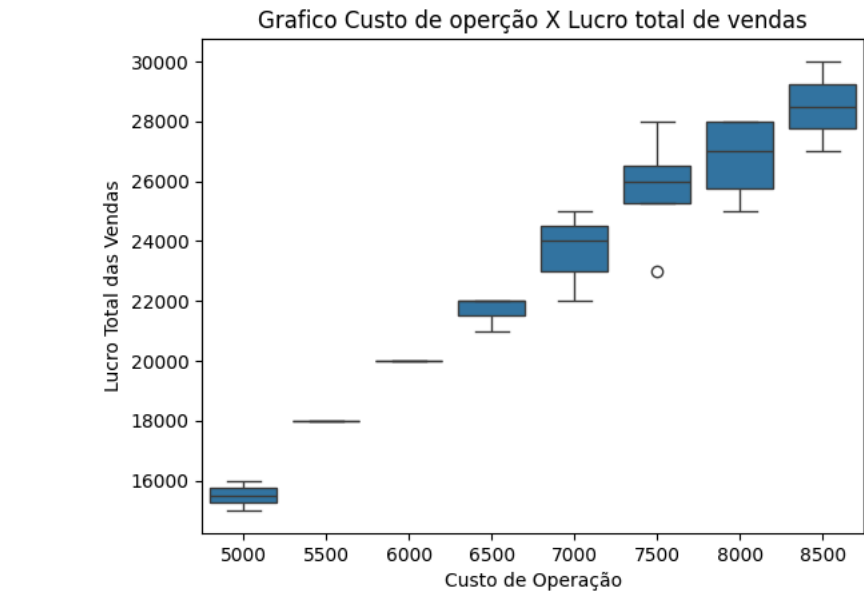
	Unnamed: 6	Unnamed: 7
count	0.0	1.000000
mean	NaN	22833.333333
min	NaN	22833.333333
25%	NaN	22833.333333
50%	NaN	22833.333333
75%	NaN	22833.333333
max	NaN	22833.333333
std	NaN	NaN

```
plt.figure(figsize=(8, 6))
sns.lineplot(df, x='Quantidade de Vendas',y='Valor Total de Vendas');
```

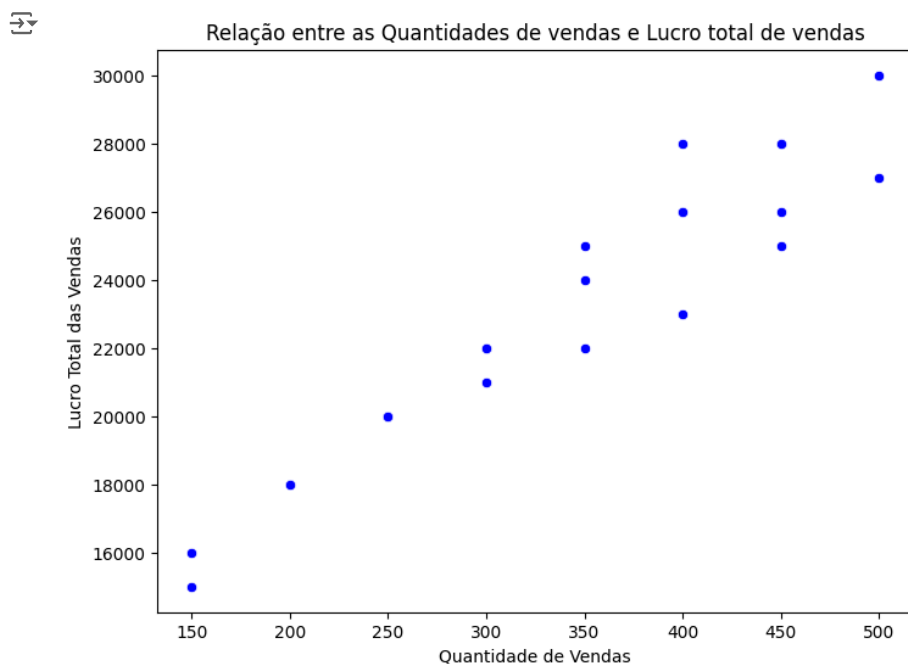


```
sns.boxplot(data=df,y='Lucro Total das Vendas',x='Custo de Operação');
plt.title('Grafico Custo de operação X Lucro total de vendas')
```

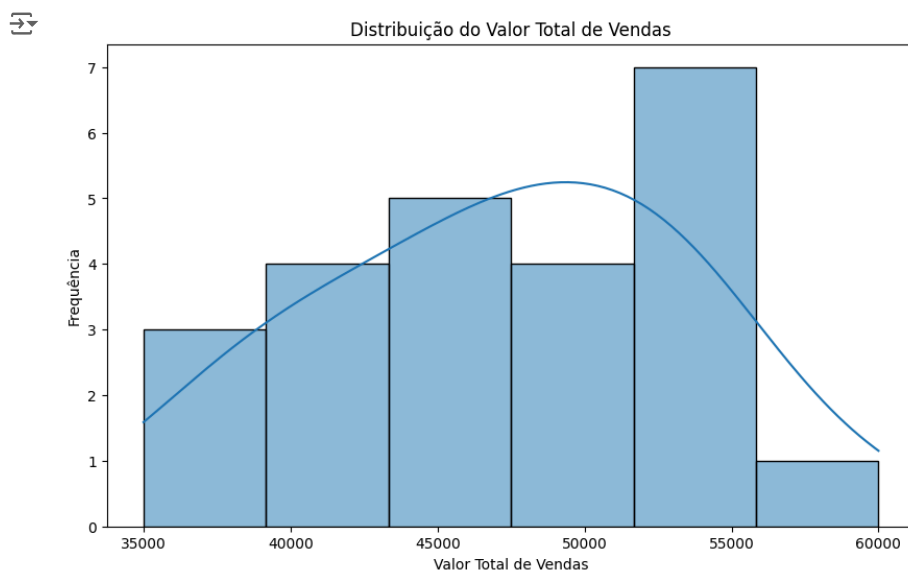
```
Text(0.5, 1.0, 'Grafico Custo de operação X Lucro total de vendas')
```



```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='Quantidade de Vendas', y='Lucro Total das Vendas', color='blue')
plt.title('Relação entre as Quantidades de vendas e Lucro total de vendas')
plt.xlabel('Quantidade de Vendas')
plt.ylabel('Lucro Total das Vendas')
plt.show()
```



```
plt.figure(figsize=(10, 6))
sns.histplot(df['Valor Total de Vendas'], kde=True)
plt.title('Distribuição do Valor Total de Vendas')
plt.xlabel('Valor Total de Vendas')
plt.ylabel('Frequência')
plt.show()
```



```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

```
# regressão linear
X = df[['Custo de Operação']]
```

```

y = df['Lucro Total das Vendas']
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.3, random_state=42)

#metricas


model = LinearRegression()

model.fit(X_train, y_train)
predictions = model.predict(X_val)

mae = mean_absolute_error(y_val, predictions)
mse = mean_squared_error(y_val, predictions)
rmse = mean_squared_error(y_val, predictions, squared=False) # Calculando RMSE a partir do MSE
r2 = r2_score(y_val, predictions)

print("Erro Médio Absoluto (MAE):", mae)
print("Erro Quadrático Médio (MSE):", mse)
print("Raiz do Erro Quadrático Médio (RMSE):", rmse)
print("R-quadrado (R²):", r2)

```


 Erro Médio Absoluto (MAE): 1020.7564575645761
 Erro Quadrático Médio (MSE): 1979701.3929548874
 Raiz do Erro Quadrático Médio (RMSE): 1407.0186185530338
 R-quadrado (R²): 0.8068584006873281

Comece a programar ou [gere código](#) com IA.

```

#PREVISAO DE LUCRO
X = df[['Quantidade de Vendas']]
y = df['Lucro Total das Vendas']
modelo=LinearRegression()
modelo.fit(X,y)


```

 **LinearRegression**
 LinearRegression()

```

clientes=180
lucro_previsto=modelo.predict([[clientes]])
print(f"com a quantidade de {clientes} clientes o lucro previsto em $ {lucro_previsto[0]:,.2f}")

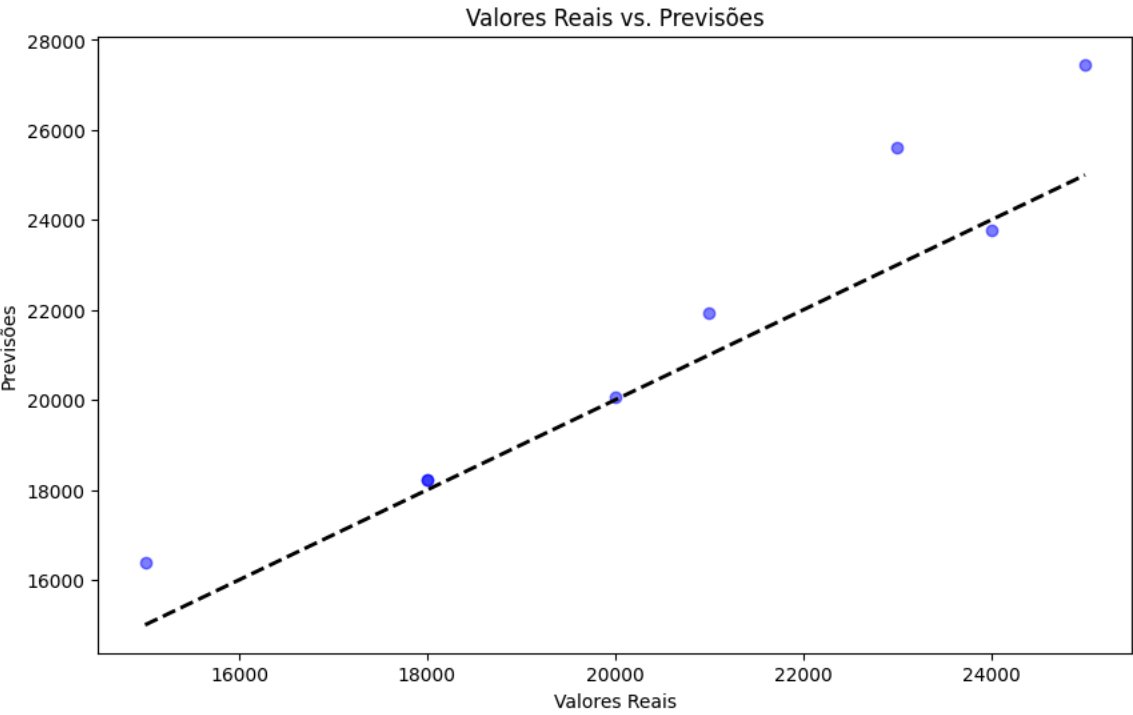
```

 com a quantidade de 180 clientes o lucro previsto em \$ 17,232.93
 /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
 warnings.warn(

```

plt.figure(figsize=(10, 6))
plt.scatter(y_val, predictions, color='blue', alpha=0.5)
plt.plot([y_val.min(), y_val.max()], [y_val.min(), y_val.max()], 'k--', lw=2)
plt.xlabel('Valores Reais')
plt.ylabel('Previsões')
plt.title('Valores Reais vs. Previsões')
plt.show()

```



Comece a programar ou [gere código](#) com IA.

Comece a programar ou [gere código](#) com IA.