Capítulo 1 – Probabilidade

Uma urna contém 5 bolas vermelhas e 7 azuis. Qual a probabilidade de retirar uma bola vermelha?

```python
import numpy as np

bolas = ['R','R','R','R','R','B','B','B','B','B','B','B']
n_sim = 10000
resultados = []

for _ in range(n_sim):
    np.random.shuffle(bolas)
    sorteio = bolas[0]
    resultados.append(sorteio)

resultados = np.array(resultados)
porcetagem_vermelha = np.mean(resultados == "R")

print(f"Probabilidade de sair bola vermelha: {porcetagem_vermelha:.4%}")
```

```
Probabilidade de sair bola vermelha: 41.4400%
```

#1.1 calcular a probabilidade de sair 2 vermelhas na sequência (com e sem reposição)

```python
bolas_calcular = np.array(['R','R','R','R','R','B','B','B','B','B','B','B'])

#sem reposição

sucessos_sem = 0
for _ in range(n_sim):
    np.random.shuffle(bolas_calcular)
    if bolas[0] == "R" and bolas[1] == "R":
        sucessos_sem += 1

prob_sem = sucessos_sem / n_sim

#com reposição

sucessos_com = 0
for _ in range(n_sim):
    primeira_saida = np.random.choice(bolas_calcular)
    segunda_saida = np.random.choice(bolas_calcular)
    if primeira_saida == "R" and segunda_saida == "R":
        sucessos_com += 1

prob_com = sucessos_com / n_sim

print(f"Probabilidade (2 vermelhas sem reposição): {prob_sem:.4f}")
print(f"Probabilidade (2 vermelhas com reposição): {prob_com:.4f}")
```

```
Probabilidade (2 vermelhas sem reposição): 0.0000
Probabilidade (2 vermelhas com reposição): 0.1682
```

Lançamento de um dado. Qual a probabilidade de sair um número maior que 3 ao lançar um dado de 6 lados?

```python
import numpy as np
lancamentos = np.random.randint(1, 7, size=5000)
prob_estimada = np.mean(lancamentos > 3)
print(f"Probabilidade estimada de sair número maior que 3: {prob_estimada:.4%}")
```

```
Probabilidade estimada de sair número maior que 3: 49.2200%
```

#3. Lançamento de duas moedas. Qual a probabilidade de sair exatamente uma cara ao lançar duas moedas?

```python
import random


opcoes = ["A","B"]
contagem = 0

for _ in range(10000):
    moeda1 = random.choice(opcoes)
    moeda2 = random.choice(opcoes)

    if(moeda1, moeda2).count("A") == 1:
        contagem +=1

probabilidade_cara = contagem / 10000
print(f"Probabilidade estimada de sair exatamente cara: {probabilidade_cara:.4%}")
```

```
Probabilidade estimada de sair exatamente cara: 49.5000%
```

#4. Probabilidade condicional. Uma urna tem 3 bolas vermelhas e 2 verdes. Se uma bola é retirada sem reposição e sai vermelha, qual a probabilidade da próxima ser verde?

```python
import numpy as np

bolas = np.array(['R', 'R', 'R', 'G', 'G'])
sucessos = 0
total = 0

for _ in range(n_sim):
    np.random.shuffle(bolas)
    primeira = bolas[0]
    segunda = bolas[1]

    if primeira == 'R':
        total += 1
        if segunda == 'G':
            sucessos += 1

P_sim = sucessos / total
print(f"Probabilidade estimada de depo : {P_sim:.4%}")
```

```
Probabilidade estimada de depo : 50.4385%
```

Capítulo 2 – Estatística Descritiva com Pandas

#5. Média, mediana e moda com o conjunto de dados: [10, 15, 20, 20, 25, 30, 35], calcule média, mediana e moda utilizando numpy, pandas.

```python
import pandas as pd
import numpy as np
import statistics

dados = [10, 15, 20, 20, 25, 30, 35]

print("Média:", statistics.mean(dados))
print("Médiana:", statistics.median(dados))
print("Moda:", statistics.mode(dados))
print("Desvio padrão:", np.std(dados))
print("Variância:", np.var(dados))
```

```
Média: 22.142857142857142
Médiana: 20
Moda: 20
Desvio padrão: 7.953949089757175
Variância: 63.26530612244898
```

#Variância e desvio padrão Calcule a variância e o desvio padrão do conjunto: [2, 4, 4, 4, 5, 5, 7, 9], usando pandas.

```
dados_1 = [2,4,4,4,5,5,7,9]

print("Desvio padrão:", np.std(dados_1))
print("Variância:", np.var(dados_1))
```

```
Desvio padrão: 7.953949089757175
Variância: 63.26530612244898
```

7. Medidas resumo Para os dados [5, 7, 8, 5, 10, 12, 15], calcule:

o Média

o Mediana

o Valor mínimo

o Valor máximo

o Amplitude

```
dados_2 = [5,6,8,5,10,12,15]

media = np.mean(dados_2)
mediana = np.median(dados_2)
valor_minimo = np.min(dados_2)
valor_maximo = np.max(dados_2)
amplitude = valor_maximo - valor_minimo

print("Média:", media)
print("Mediana:", mediana)
print("Valor mínimo:", valor_minimo)
print("Valor máximo:", valor_maximo)
print("Amplitude:", amplitude)
```

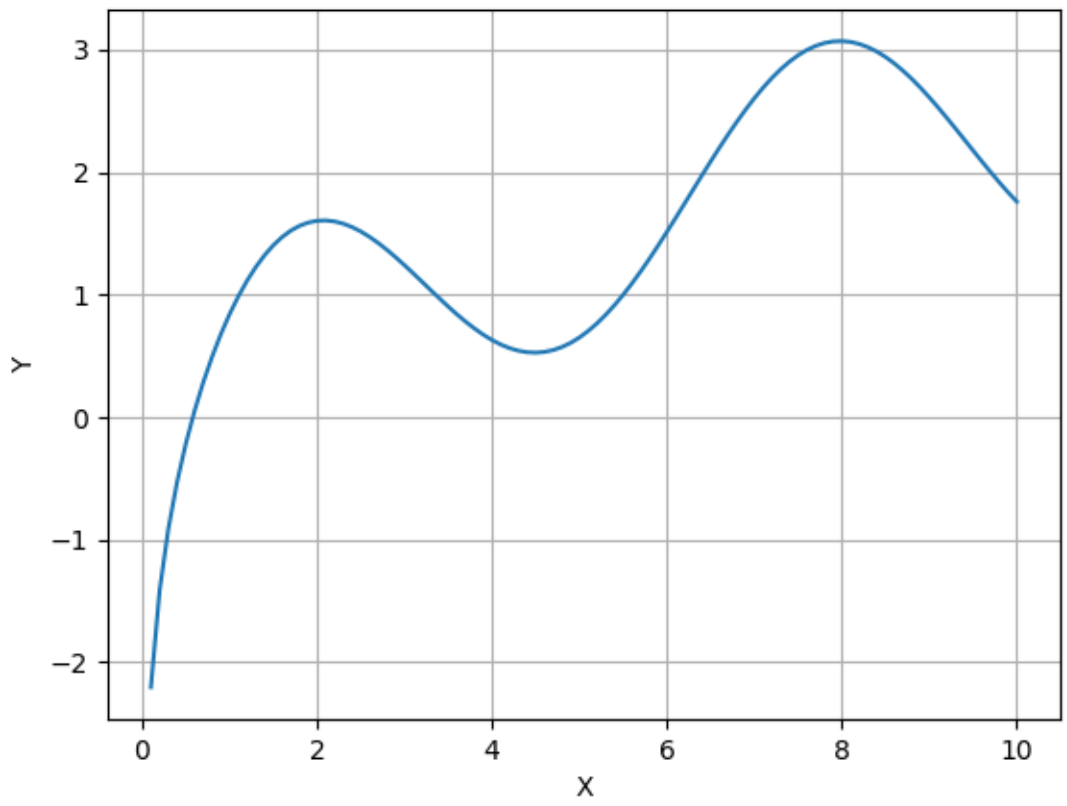
```
Média: 8.714285714285714
Mediana: 8.0
Valor mínimo: 5
Valor máximo: 15
Amplitude: 10
```

Capítulo 3 – Funções Trigonométricas e Logarítmicas

Capítulo 3 – Funções Trigonométricas e Logarítmicas

8. Enunciado: Usando numpy, gere uma série de valores x (100 números entre 0.1 e 10) e calcule y = sin(x) + log(x). Plote o gráfico.

```python
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0.1, 10, 100)

y =np.sin(x) + np.log(x)

plt.plot(x,y)
plt.xlabel("X")
plt.ylabel("Y")
plt.grid(True)
plt.show()
```



Capítulo 4 – Regressão Linear

9. Horas de estudo vs. notas Considere os dados:

o Horas de estudo: [1, 2, 3, 4, 5]

o Notas: [2, 4, 5, 4, 6] Ajuste uma regressão linear simples e interprete o coeficiente angular

```python
import numpy as np
from sklearn.linear_model import LinearRegression

x1 = np.array([1,2,3,4,5]).reshape(-1,1)
y1 = np.array([2,4,5,4,6])
modelo1 = LinearRegression()
modelo1.fit(x1,y1)
print("Coeficiente angular:", modelo1.coef_[0])
print("Intercepto:", modelo1.intercept_)
```

```
Coeficiente angular: 0.8000000000000002
Intercepto: 1.7999999999999998
```

#10. Preço vs. tamanho de imóveis Considere os dados:

o Tamanho (m²): [50, 60, 70, 80, 90]

o Preço (mil reais): [150, 200, 210, 240, 280] Ajuste um modelo de regressão linear e estime o preço para um imóvel de 100 m²

```
tamanho = np.array([50,60,70,80,90])
preco = np.array([150,200,210,240,280])

a_1, b_1 = np.polyfit(tamanho, preco, 1)

preco_100 = a_1 * 100 + b_1

print(f"Coeficiente angular (a): `{a_1:.2f}")
print(f"Coeficiente linear (b): `{b_1:.2f}")
print(f"Preço estimado para 100 m²: {preco_100:.2f} mill reais")
```

```
Coeficiente angular (a): `3.00
Coeficiente linear (b): `6.00
Preço estimado para 100 m²: 306.00 mill reais
```

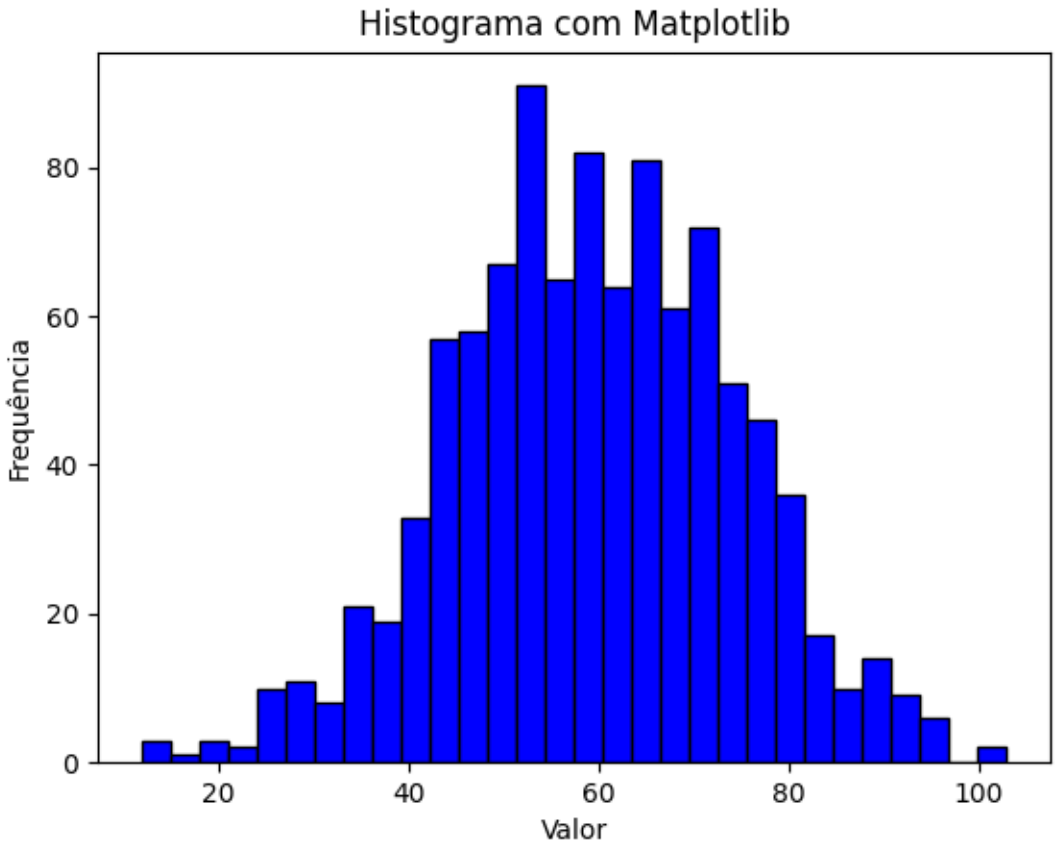
Capítulo 5 – Visualizações com Matplotlib e Seaborn

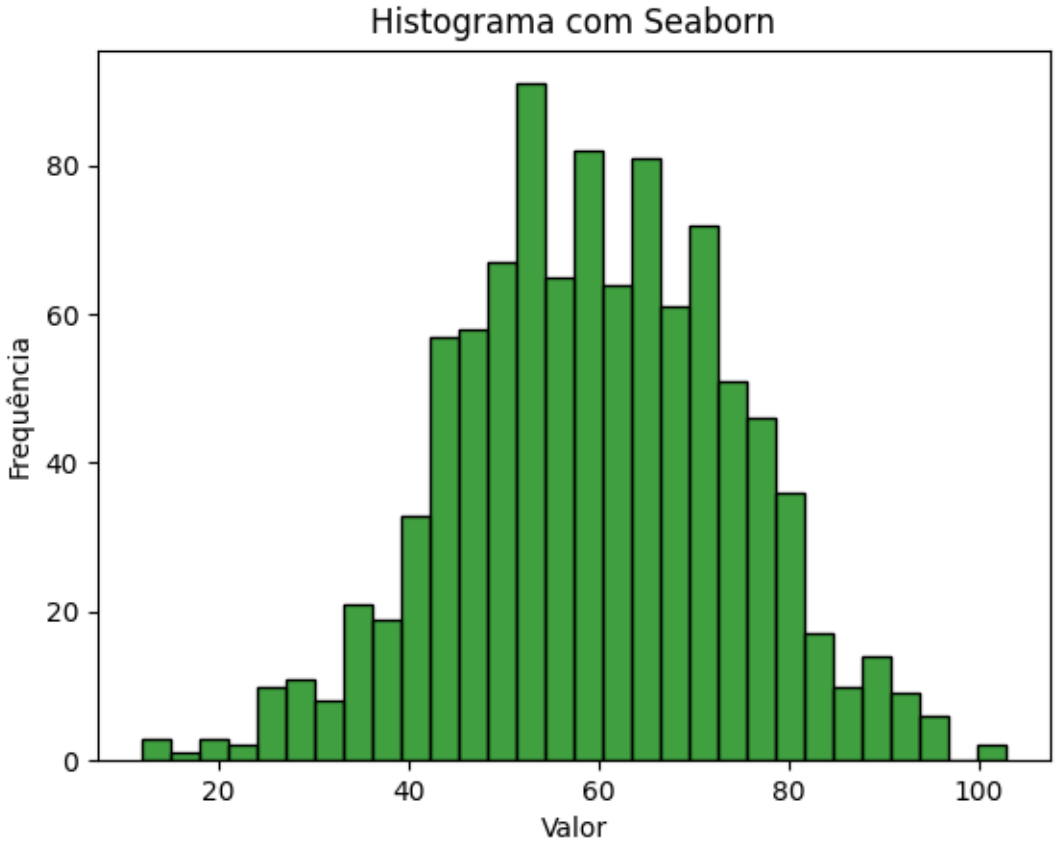
11.Histograma (Matplotlib e Seaborn) Gere 1000 números aleatórios com distribuição normal (média 60, desvio padrão 15).

o Plote um histograma com matplotlib.

o Plote o mesmo histograma com seabor

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

dados = np.random.normal(loc=60, scale=15, size=1000)

plt.hist(dados, bins=30, color="blue", edgecolor = "black")
plt.title("Histograma com Matplotlib")
plt.xlabel("Valor")
plt.ylabel("Frequência")
plt.show()

sns.histplot(dados, bins=30, color="green", kde=False)
plt.title("Histograma com Seaborn")
plt.xlabel("Valor")
plt.ylabel("Frequência")
plt.show()
```



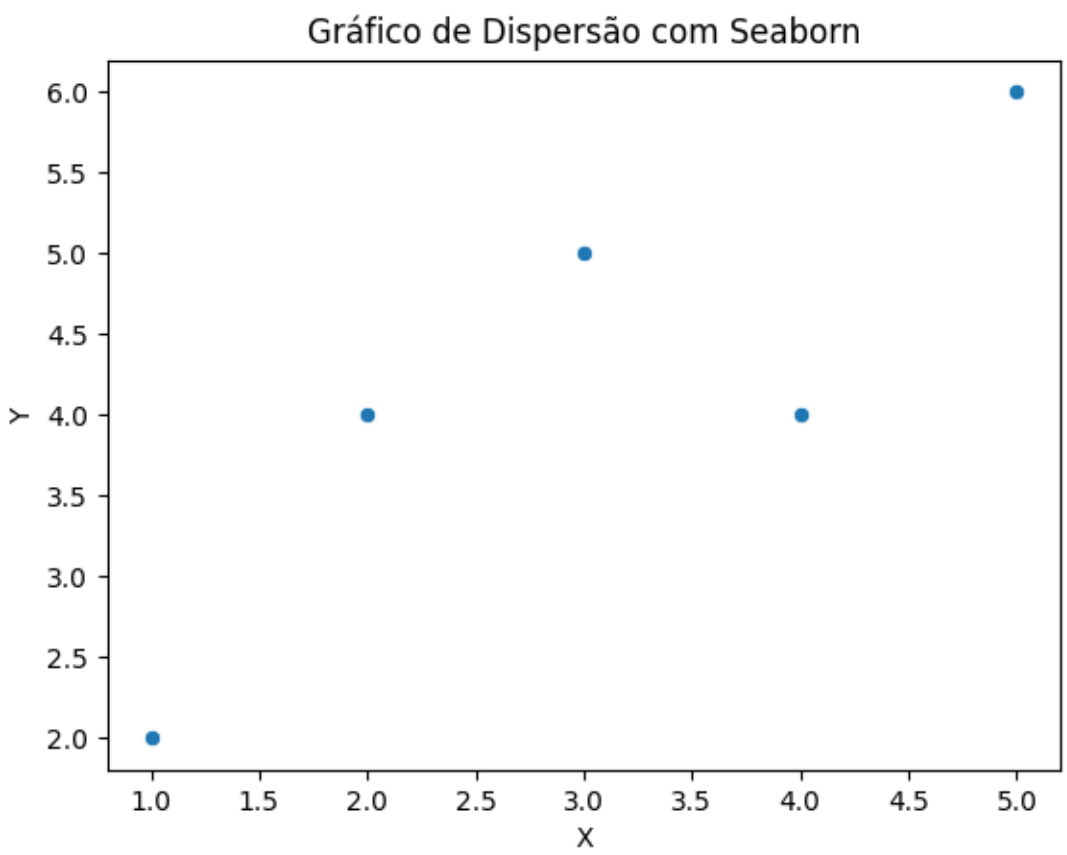


#12.Gráfico de dispersão (Seaborn) Considere os dados:

o X = [1, 2, 3, 4, 5]

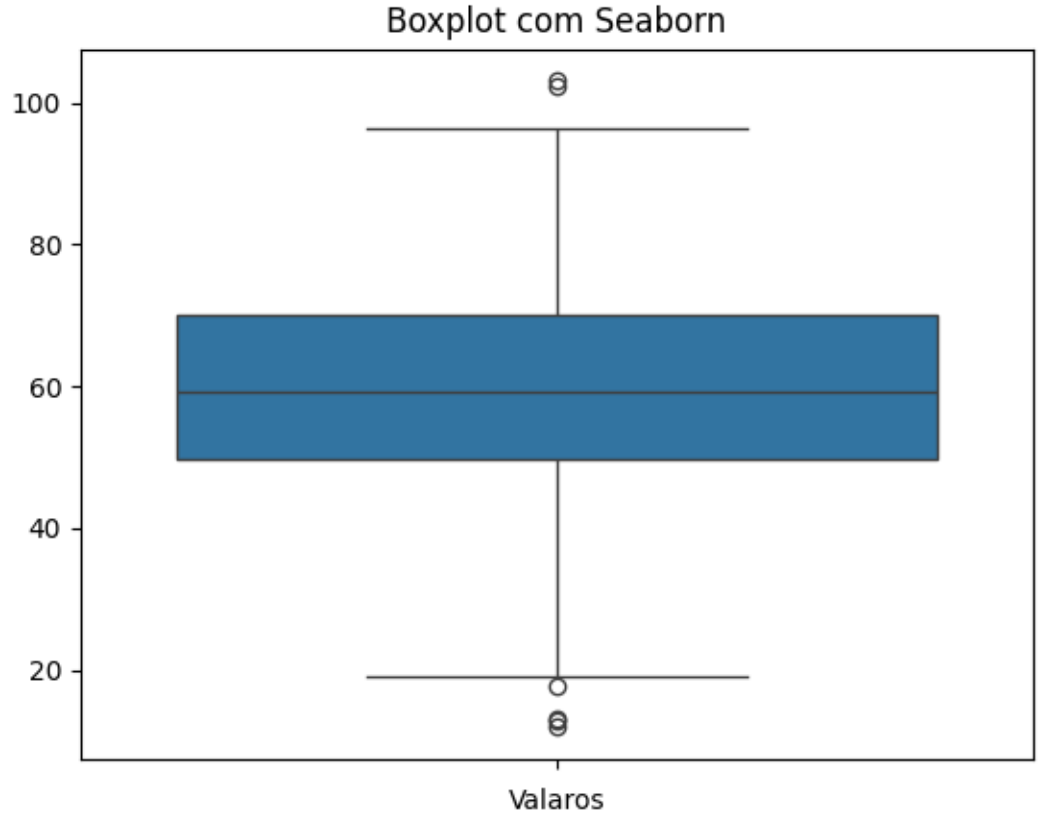
o Y = [2, 4, 5, 4, 6] Plote o gráfico de dispersão (scatter plot) usando seaborn.

```python
dados_1 = pd.DataFrame({
    "X": [1,2,3,4,5],
    "Y": [2,4,5,4,6],
})

sns.scatterplot(data=dados_1, x="X", y="Y")
plt.title("Gráfico de Dispersão com Seaborn")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```



#13.Boxplot (Seaborn) Construa um boxplot para os dados: [7, 8, 5, 6, 12, 14, 15, 8, 9, 10] com seaborn

```
dados_2 = [7,8,5,6,12,14,15,8,9,10]

sns.boxplot(data=dados)
plt.title("Boxplot com Seaborn")
plt.xlabel("Valaros")
plt.show()
```



Exercícios – CSV “industria.csv”

#Carregue o arquivo industria.csv no Python usando pandas.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np


df = pd.read_csv("projeto_menu/industria.csv")
```

Exercício 1 – Receita total por fábrica

Calcule a receita total de cada fábrica.

```
receita_frabrica = df.groupby("Fabrica")["Receita"].sum().reset_index()

print(f"Receita total por fábrica:",receita_frabrica)
```

#Gere um gráfico de barras mostrando a receita total por fábrica.

```
plt.figure(figsize=(10,6))
plt.bar(receita_frabrica["Fabrica"], receita_frabrica["Receita"], color="skyblue")

plt.title("Receita Total por Fábrica")
plt.ylabel("Receita Total (R$)")
plt.tight_layout()
plt.show(block=False)
```


#Qual fábrica teve a maior receita?

```
fabrica_maior = receita_frabrica.loc[receita_frabrica["Receita"].idxmax()]

print("Fábrica com maior receita:")
print(fabrica_maior)
```

```
Fábrica com maior receita:
Fabrica    Fabrica A
Receita       269000
```

#Qual a diferença entre a fábrica com maior receita e a de menor receita?

```
maior_receita = receita_frabrica["Receita"].max()
menor_receita = receita_frabrica["Receita"].min()

receita_diferente = maior_receita - menor_receita

print(f"A diferença entre a maior e a menor receita é de R${receita_diferente:.2f}")
```

```
A diferença entre a maior e a menor receita é de R$147000.00
```

#Exercício 2 – Receita média por produto

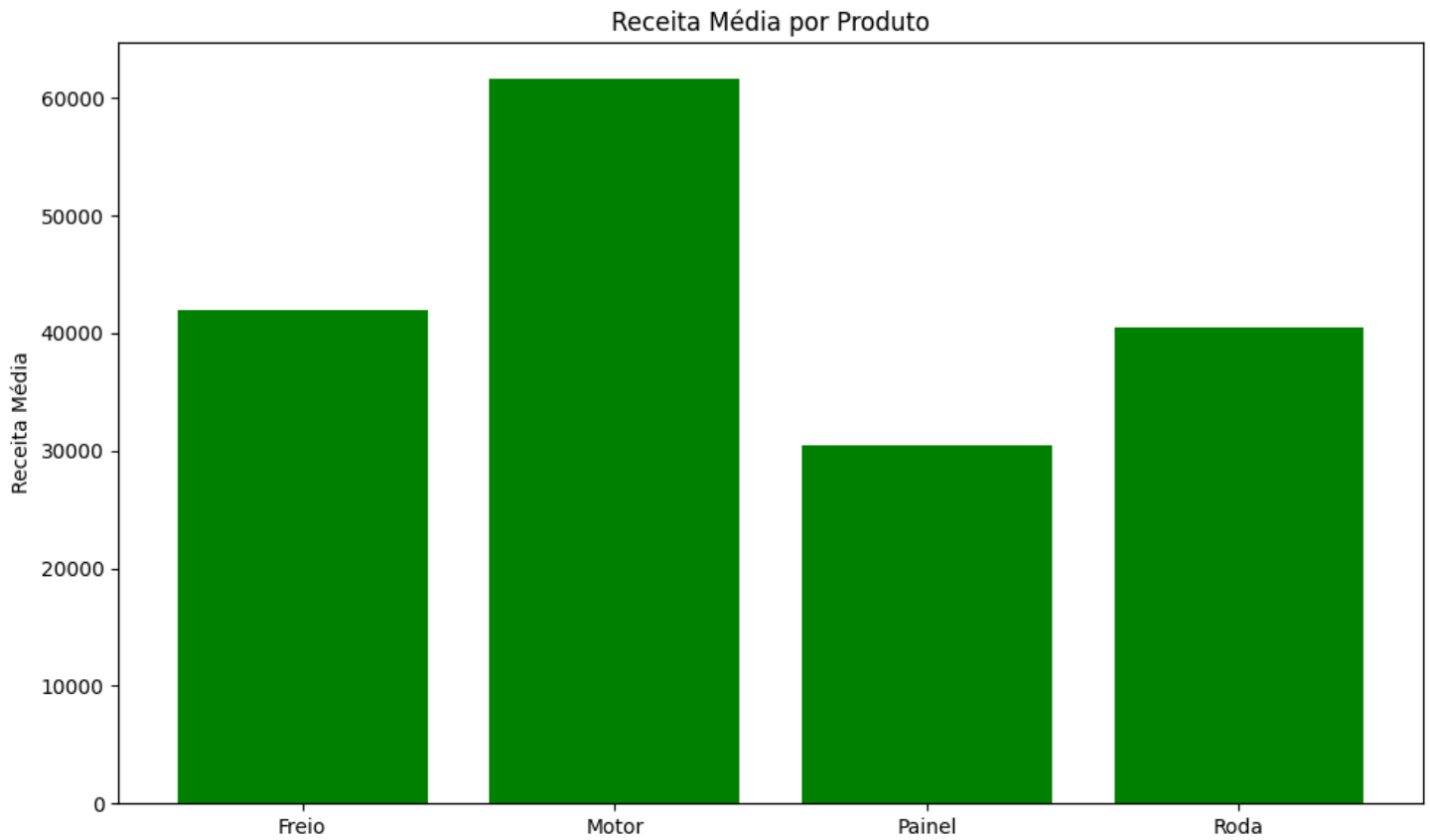
#Calcule a receita média de cada produto.

```
receita_media_produto = df.groupby("Produto")["Receita"].mean().reset_index()

print(f"Receita média por produto: ", receita_media_produto)
```

```
Receita média por produto:    Produto       Receita
0   Freio  42000.000000
1   Motor  61666.666667
2  Painel  30500.000000
3    Roda  40500.000000
```

#Gere um gráfico de barras mostrando a receita média por produto.

```
plt.figure(figsize=(10,6))
plt.bar(receita_media_produto["Produto"], receita_media_produto["Receita"], color = "green")

plt.title("Receita Média por Produto")
plt.ylabel("Produto")
plt.ylabel("Receita Média")
plt.tight_layout()
plt.show(block=False)
```



#Qual produto tem a maior receita média?

```
produto_maior_media = receita_media_produto.loc[receita_media_produto["Receita"].idxmax()]

print(f"Produto com a maior receita média é: ", produto_maior_media)
```

```
Produto com a maior receita média é:  Produto          Motor
Receita    61666.666667
```

Exercício 3 – Quantidade vendida total por mês

#Crie uma coluna Mes a partir da data de cada registro.

```
df["Data"] = pd.to_datetime(df["Data"])
df["Mes"] = df["Data"].dt.month

print(df[["Data", "Mes"]].head())
```

```
        Data  Mes
0 2025-01-10    1
1 2025-01-10    1
2 2025-01-15    1
3 2025-02-05    2
4 2025-02-12    2
```

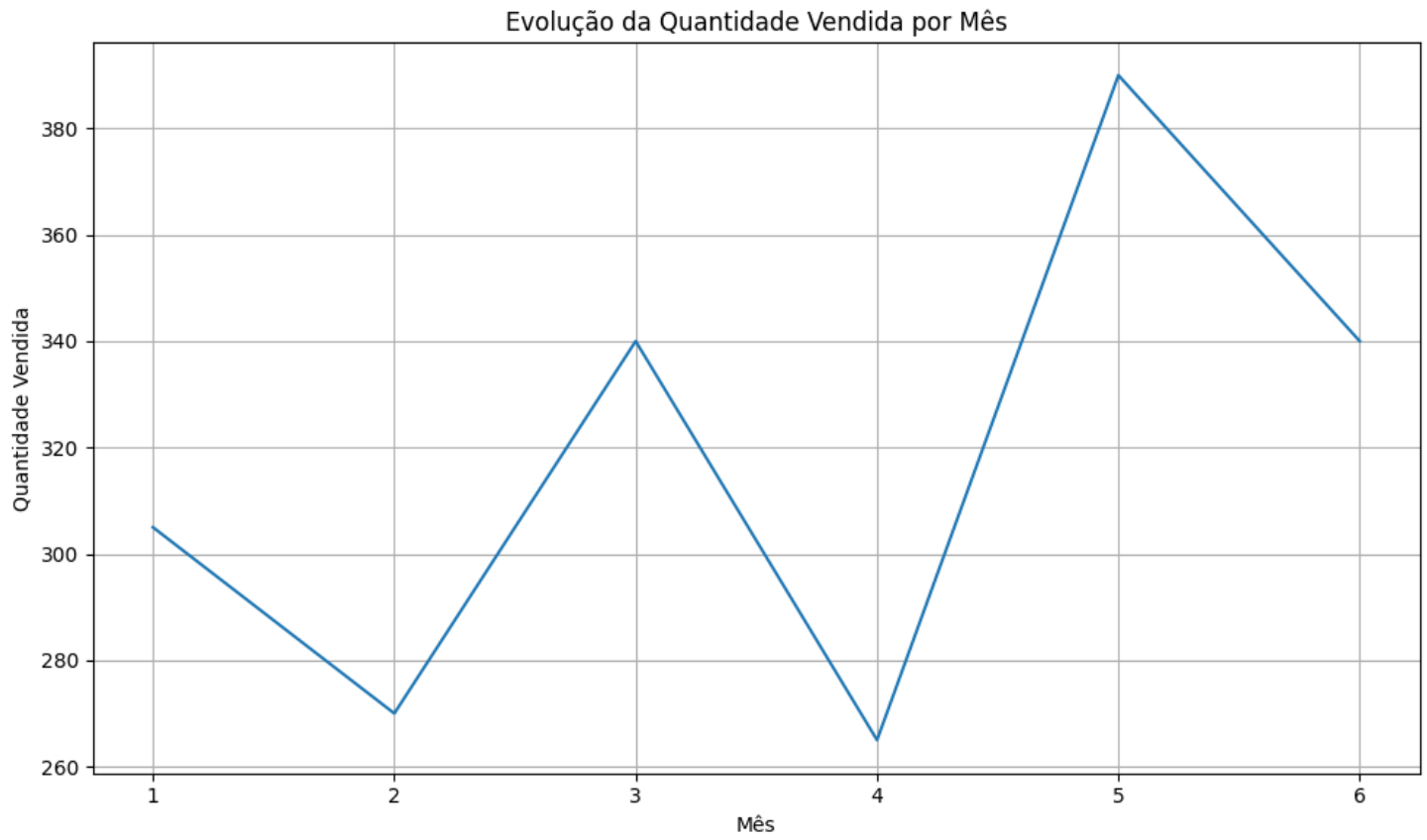
#Calcule a quantidade total vendida por mês.

```
quantidade_vendida_por_mes = df.groupby("Mes")["Quantidade_Vendida"].sum().reset_index()

print("Quantidade total vendida por mês: ")
print(quantidade_vendida_por_mes)
```

```
Quantidade total vendida por mês:
   Mes  Quantidade_Vendida
0    1                 305
1    2                 270
2    3                 340
3    4                 265
4    5                 390
5    6                 340
```

#Gere um gráfico de linha mostrando a evolução da quantidade vendida ao longo dos meses.

```
plt.figure(figsize=(10,6))
plt.plot(quantidade_vendida_por_mes["Mes"], quantidade_vendida_por_mes["Quantidade_Vendida"])

plt.title("Evolução da Quantidade Vendida por Mês")
plt.xlabel("Mês")
plt.ylabel("Quantidade Vendida")
plt.grid(True)
plt.tight_layout()
plt.show(block=False)
```



#Qual mês teve a maior quantidade vendida?

```
mes_maior_venda = quantidade_vendida_por_mes.loc[quantidade_vendida_por_mes["Quantidade_Vendida"].idxmax()]

print(f"O mês com a maior quantidade vendida foi:")
print(mes_maior_venda)

#Existe tendência de aumento ou diminuição ao longo do período?

primeiro_mes = quantidade_vendida_por_mes.iloc[0]["Quantidade_Vendida"]
ultimo_mes = quantidade_vendida_por_mes.iloc[-1]["Quantidade_Vendida"]

if ultimo_mes > primeiro_mes:
    print("Tendência de aumento na quantidade vendida.")
elif ultimo_mes < primeiro_mes:
    print("Tendência de diminuição na quantidade vendida.")
else:
    print("Sem tendência clara na quantidade vendida.")
```

```
Tendência de aumento na quantidade vendida.
```

Exercício 4 – Lucro médio por fábrica

#Crie uma coluna Lucro = Receita - Custo.

```
df["Lucro"] = df["Receita"] - df["Custo"]
print(df[["Receita", "Custo", "Lucro"]].head())
```

```
   Receita  Custo  Lucro
0    50000  32000  18000
1    25000  15000  10000
2    65000  40000  25000
3    36000  21000  15000
4    30000  18000  12000
```

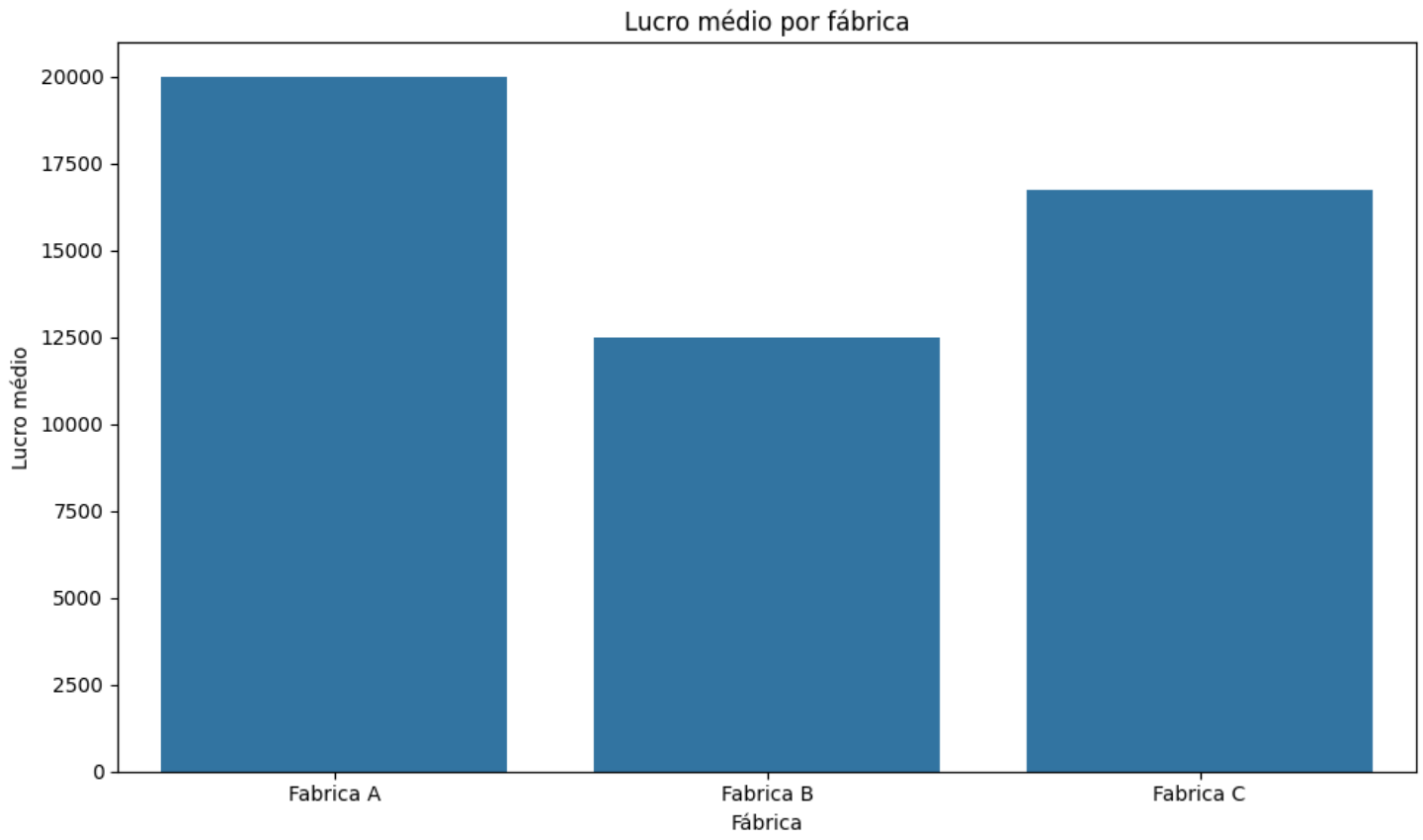
#Calcule o lucro médio por fábrica usando groupby.

```
lucro_medio_por_fabrica = df.groupby("Fabrica")["Lucro"].mean().reset_index()

print("Lucro médio por fábrica:")
print(lucro_medio_por_fabrica)
```

```
Lucro médio por fábrica:
     Fabrica    Lucro
0  Fabrica A  20000.0
1  Fabrica B  12500.0
2  Fabrica C  16750.0
```

#Gere um gráfico de barras usando seaborn mostrando o lucro médio por fábrica.

```
plt.figure(figsize=(10,6))
sns.barplot(x="Fabrica", y="Lucro", data=lucro_medio_por_fabrica)

plt.title("Lucro médio por fábrica")
plt.xlabel("Fábrica")
plt.ylabel("Lucro médio")

plt.tight_layout()
plt.show(block=False)
```



#Qual fábrica é mais lucrativa em média?

```
fabrica_mais_lucrativa = lucro_medio_por_fabrica.loc[lucro_medio_por_fabrica["Lucro"].idxmax()]

print("A fábrica mais lucrativa em média é:")
print(fabrica_mais_lucrativa)
```

```
A fábrica mais lucrativa em média é:
Fabrica    Fabrica A
Lucro        20000.0
```

#Alguma fábrica apresenta lucro negativo em algum registro?

```
print("Existe lucro negativo? ", (df["Lucro"]< 0).any)
```

```
Existe lucro negativo?  <bound method Series.any of 0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
```

Exercício 5 – Receita total por fábrica e produto

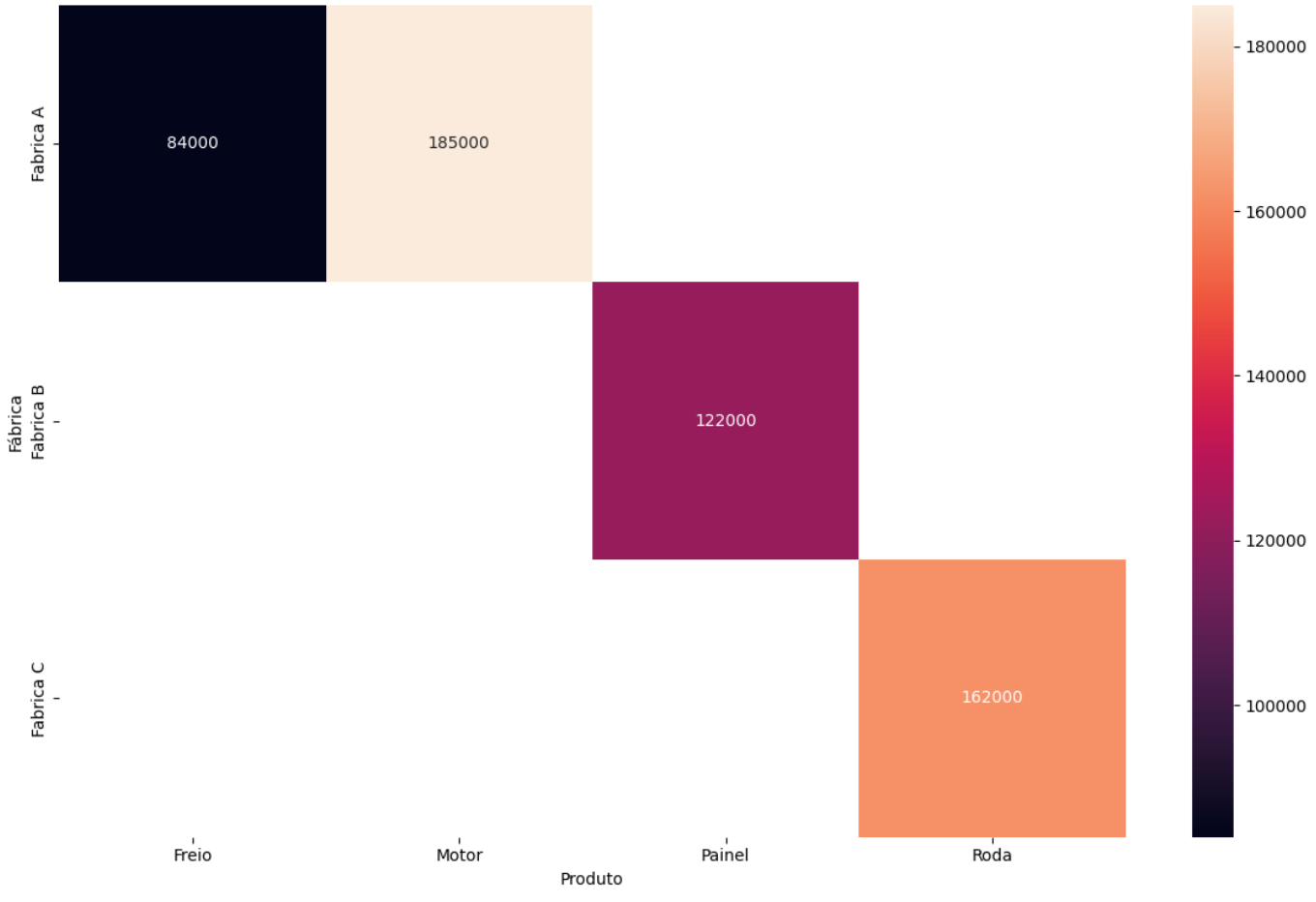
#Use groupby ou pivot_table para calcular a receita total para cada combinação de fábrica e produto.

```
receita_total = df.groupby(["Fabrica", "Produto"])["Receita"].sum().reset_index()
print(receita_total)
```

	Fabrica	Produto	Receita
0	Fabrica A	Freio	84000
1	Fabrica A	Motor	185000
2	Fabrica B	Painel	122000
3	Fabrica C	Roda	162000

#Gere um heatmap com seaborn mostrando a receita por fábrica (linhas) e produto (colunas).

```
receita_pivot = pd.pivot_table(df, values="Receita", index="Fabrica", columns="Produto", aggfunc="sum")


plt.figure(figsize=(12,8))
sns.heatmap(receita_pivot, annot=True, fmt=".0f")

plt.title("Receita total por fábrica e produto")
plt.xlabel("Produto")
plt.ylabel("Fábrica")

plt.tight_layout()
plt.show(block=False)
input("Presione Enter para fechar tudo...")
plt.close("all")
```



#Qual produto gera mais receita em cada fábrica?

```
receita = df.groupby(["Fabrica", "Produto"])["Receita"].sum()

maior_receita_fabrica = receita.groupby(level=0).idxmax()

print(maior_receita_fabrica)
```

```
Fabrica A     (Fabrica A, Motor)
Fabrica B    (Fabrica B, Painel)
Fabrica C      (Fabrica C, Roda)
```

#Existe algum produto que não foi produzido ou vendido em alguma fábrica?

```
fabricas = df["Fabrica"].unique()
produtos = df["Produto"].unique()

combinacoes_reais = df[["Fabrica", "Produto"]].drop_duplicates()

total_combinacoes = len(fabricas) * len(produtos)
combinacoes_existentes = len(combinacoes_reais)

if combinacoes_existentes < total_combinacoes:
    print("Existe pelo menos um produto que não foi produzido/vendido em alguma fábrica.")
else:
    print("Todos os produtos foram produzidos/vendidos em todas as fábricas.")
```

```
Existe pelo menos um produto que não foi produzido/vendido em alguma fábrica.
```