PARTE 1 - LIMPEZA E TRATAMENTO DE DADOS

SEMANAS DO DESAFIO 11, 12 E 13 e outras referências.

ANDRE MOURA LIMA

Utilizando suas habilidades ninja em Python e bibliotecas como Pandas, Numpy, Seaborn e outr
deve:

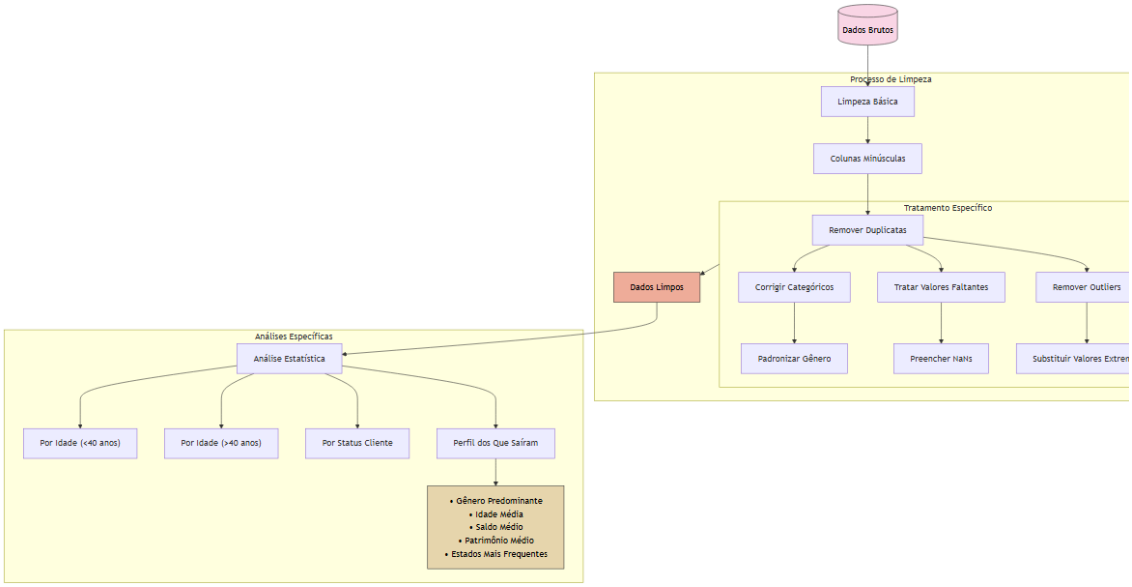
- Altere os nomes das colunas para letras minúsculas, utilizando a biblioteca pandas;
- Explore colunas categóricas, utilizando função ***groupby()*** e outras, visando aplicar os gráfic
  visualizações e a compreensão desses dados.
- Explore colunas numéricas, utilizando a função ***describe**(), **faça um *boxplot* ou outro g**
  que achar necessário, utilizando as bibliotecas pandas e **seaborn;
- **Corrigir os dados faltantes (*NANs)*** e preencher as lacunas com valores adequados.
- **Tratar os outliers, substituindo-os pelos valores adequados**.
- **Eliminar os duplicados, para** garantir a unicidade.
- **corrigir os dados categóricos, por exemplo,** "Mas" para "Masculino", "Fem" para "Femin
  outras inconsistências se houver, para adequá-los ao padrão.
- Mostre o tamanho dos dados importados, utilizando o *shape*.

PARTE 2 - ESTATISTICA:

- Apresente a média e a mediana do **saldo na conta** dos clientes abaixo de 40 anos;
- Apresente a média e a mediana do **saldo na conta** dos clientes acima de 40 anos;
- Apresente a média e a mediana do **saldo na conta** dos clientes que saíram e dos que
  permaneceram;
- Dos que saíram, mostre qual é o público predominante (Masculino ou Feminino), a idade, o
  na conta, patrimônio e os seus respectivos estados;

DESAFIO: ANÁLISE DE CLIENTES BANCÁRIOS – RELATÓRIO PARA O GERENTE

FLUXO DE TRABALHO - DIAGRAMA DE ATIVIDADES



IMPORTANDO PLANILHA - DADOS

In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [2]:
```
# Essas são as principais bibliotecas
!pip install pandas numpy seaborn openpyxl
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packag
(2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packag
(2.0.2)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packa
(0.13.2)
Requirement already satisfied: openpyxl in /usr/local/lib/python3.11/dist-pack
(3.1.5)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python
dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-
packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dis
packages (from pandas) (2025.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/
python3.11/dist-packages (from seaborn) (3.10.0)
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.11/dist-
packages (from openpyxl) (2.0.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/d
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/di
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-pac
(from matplotlib!=3.6.1,>=3.4->seaborn) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/d
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.2.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-pack
(from python-dateutil>=2.8.2->pandas) (1.17.0)
```

In [105…

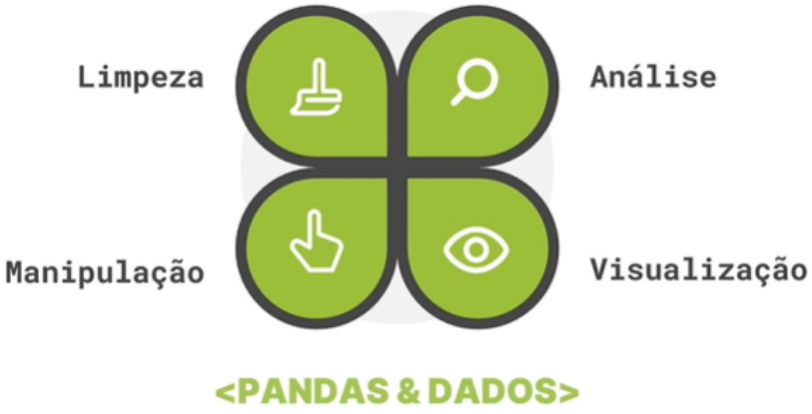
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
#Bibliotecas
```

In [152…

```python
import pandas as pd

# ID da sua planilha...
sheet_id = "1TUCtbo_xxkFqXoX93FT0JN5e1seirwYHOOq9t8ElfI0"

# Nome da aba que você quer ler (ex: 'Planilha1')...
sheet_name = "Planilha1"

# Montar a URL para ler como xlsx...
url = f"https://docs.google.com/spreadsheets/d/{sheet_id}/gviz/tq?tqx=out

# Ler o arquivo
dados = pd.read_csv(url)

# Mostrar o shape (linhas, colunas)...
print(f"Formato do DataFrame: {dados.shape}")

# Visualizar as primeiras linhas...
```

```
#dados.head()

# Definição importando aquivo local. Esse módulo pegar direto no driver m
```

```
Formato do DataFrame: (999, 12)
```



In [153…
```
dados.head(10)
```

Out[153…

	ID	PONTOS	ESTADO	GENERO	IDADE	BENS	SALDO NA CONTA	PRODUTOS	POSSUI CARTAO	AT
0	1	619	PI	Feminino	42	2	0	1	1	
1	2	608	CE	Feminino	41	1	8380786	1	0	
2	3	502	PI	Feminino	42	8	1596608	3	1	
3	4	699	PI	Feminino	39	1	0	2	0	
4	5	850	CE	Feminino	43	2	12551082	1	1	
5	6	645	CE	Masculino	44	8	11375578	2	1	
6	7	822	PI	Mas	50	7	0	2	1	
7	8	376	MA	Feminino	29	4	11504674	4	1	
8	9	501	PI	Masculino	44	4	14205107	2	0	
9	10	684	PI	Masculino	27	2	13460388	1	1	

read_excel(): Essa função é usada para ler arquivos do Excel (.xls ou .xlsx) e criar um DataFra
partir dos dados.

read - Visualizar apenas algumas linhas iniciais da nossa base de dados.

Para peguar as 5 últimas linhas.

In [154…
```
dados.tail()
```

Out[154…

	ID	PONTOS	ESTADO	GENERO	IDADE	BENS	SALDO NA CONTA	PRODUTOS	POSSU CARTAC
994	996	838	CE	Masculino	43	9	12310588	2	1
995	997	610	CE	Masculino	29	9	0	3	0
996	998	811	CE	Masculino	44	3	0	2	0
997	999	587	CE	Masculino	62	7	12128627	1	0
998	1000	811	MA	Feminino	28	4	16773882	2	1

TIPO DE DADOS

In [155…

```
type(dados)
```

Out[155…

pandas.core.frame.DataFrame

def __init__(data=None, index: Axes | None=None, columns: Axes | None=None, dtype: Dtype None=None, copy: bool | None=None) -> None

Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Data structure also contains labeled axes (rows and columns).
Arithmetic operations align on both row and column labels. Can be
thought of as a dict-like container for Series objects. The primary
pandas data structure.

Parameters
----
data : ndarray (structured or homogeneous), Iterable, dict, or DataFrame
    Dict can contain Series, arrays, constants, dataclass or list-like objects. If
    data is a dict, column order follows insertion-order. If a dict contains Series
    which have an index defined, it is aligned by its index. This alignment also
    occurs if data is a Series or a DataFrame itself. Alignment is done on
    Series/DataFrame inputs.

    If data is a list of dicts, column order follows insertion-order.

index : Index or array-like
    Index to use for resulting frame. Will default to RangeIndex if
    no indexing information part of input data and no index provided.
columns : Index or array-like
    Column labels to use for resulting frame when data does not have them,
    defaulting to RangeIndex(0, 1, 2, ..., n). If data contains column labels,
    will perform column selection instead.
dtype : dtype, default None
    Data type to force. Only a single dtype is allowed. If None, infer.
copy : bool or None, default None
    Copy data from inputs.
    For dict data, the default of None behaves like ``copy=True``. For DataFrame
    or 2d ndarray input, the default of None behaves like ``copy=False``.
    If data is a dict containing one or more Series (possibly of different dtypes),
    ``copy=False`` will ensure that these inputs are not copied.

    .. versionchanged:: 1.3.0

See Also

DataFrame.from_records : Constructor from tuples, also record arrays.
DataFrame.from_dict : From dicts of Series, arrays, or dicts.
read_csv : Read a comma-separated values (csv) file into DataFrame.

read_table : Read general delimited file into DataFrame.
read_clipboard : Read text from clipboard into DataFrame.

Notes

Please reference the :ref:`User Guide <basics.dataframe>` for more information.

Examples

Constructing DataFrame from a dictionary.

```
>>> d = {'col1': [1, 2], 'col2': [3, 4]}
>>> df = pd.DataFrame(data=d)
>>> df
   col1  col2
0     1     3
1     2     4
```

Notice that the inferred dtype is int64.

```
>>> df.dtypes
col1    int64
col2    int64
dtype: object
```

To enforce a single dtype:

```
>>> df = pd.DataFrame(data=d, dtype=np.int8)
>>> df.dtypes
col1    int8
col2    int8
dtype: object
```

Constructing DataFrame from a dictionary including Series:

```
>>> d = {'col1': [0, 1, 2, 3], 'col2': pd.Series([2, 3], index=[2, 3])}
>>> pd.DataFrame(data=d, index=[0, 1, 2, 3])
   col1  col2
0     0   NaN
1     1   NaN
2     2   2.0
3     3   3.0
```

Constructing DataFrame from numpy ndarray:

```
>>> df2 = pd.DataFrame(np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]),
...                    columns=['a', 'b', 'c'])
>>> df2
   a  b  c
0  1  2  3
```

1 - Altere os nomes das colunas para letras minúsculas, utilizando a biblioteca **pandas**;

In [156…
```
dados.columns
```

Out[156…
```
Index(['ID', 'PONTOS', 'ESTADO', 'GENERO', 'IDADE', 'BENS', 'SALDO NA CONTA'
       'PRODUTOS', 'POSSUI CARTAO', 'ATIVO', 'SALARIO ANUAL', 'SAIU'],
      dtype='object')
```

In [157…
```
dados.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              999 non-null    int64
 1   PONTOS          999 non-null    int64
 2   ESTADO          999 non-null    object
 3   GENERO          991 non-null    object
 4   IDADE           999 non-null    int64
 5   BENS            999 non-null    int64
 6   SALDO NA CONTA  999 non-null    int64
 7   PRODUTOS        999 non-null    int64
 8   POSSUI CARTAO   999 non-null    int64
 9   ATIVO           999 non-null    int64
 10  SALARIO ANUAL   992 non-null    float64
 11  SAIU            999 non-null    int64
dtypes: float64(1), int64(9), object(2)
memory usage: 93.8+ KB
```

In [158…
```
dados['PONTOS']
```

Out[158…

	PONTOS
0	619
1	608
2	502
3	699
4	850
...	...
994	838
995	610
996	811
997	587
998	811

999 rows × 1 columns

dtype: int64

In [159…
```
dados[['PONTOS', 'ESTADO']]
```

Out[159…

	PONTOS	ESTADO
0	619	PI
1	608	CE
2	502	PI
3	699	PI
4	850	CE
...	...	...
994	838	CE
995	610	CE
996	811	CE
997	587	CE
998	811	MA

999 rows × 2 columns

Alteração de Maiúscula para menúscula

In [160…
```
# Deixar todas as colunas em minúsculo
dados.columns = dados.columns.str.lower()

# Conferir
dados.head()
```

Out[160…

	id	pontos	estado	genero	idade	bens	saldo na conta	produtos	possui cartao	ativo	sa a
0	1	619	PI	Feminino	42	2	0	1	1	1	101348
1	2	608	CE	Feminino	41	1	8380786	1	0	1	112542
2	3	502	PI	Feminino	42	8	1596608	3	1	0	113931
3	4	699	PI	Feminino	39	1	0	2	0	0	93826
4	5	850	CE	Feminino	43	2	12551082	1	1	1	7908

2 - Corrigir os dados categóricos, por exemplo, "Mas" para "Masculino", "Fem" para "Feminino" inconsistências se houver, para adequá-los ao padrão.

In [161…
```
# Padronizar gênero
dados['genero'] = dados['genero'].replace({
    'Mas': 'Masculino',
    'Fem': 'Feminino',
    'M': 'Masculino',
    'F': 'Feminino'
})

# Verificar valores únicos após correção
print("\nValores únicos na coluna 'genero':")
print(dados['genero'].unique())
```

```
Valores únicos na coluna 'genero':
['Feminino' 'Masculino' nan]
```

Visualizar valores alterados

In [ ]:
```
# Exemplo de DataFrame
dados = pd.DataFrame({'genero': ['Mas', 'Fem', 'MASC', 'FEMIN', 'Masculin

# Substituir valores abreviados por formas completas
dados['genero'] = dados['genero'].replace({
    'Mas': 'Masculino',
    'Fem': 'Feminino',
    'MASC': 'Masculino',
    'FEMIN': 'Feminino'
})

# Visualizar os valores únicos após a substituição
valores_unicos = dados['genero'].unique()
print(valores_unicos)
```

```
['Masculino' 'Feminino']
```

3 - Eliminar os duplicados, para garantir a unicidade.

In [162…
```
# Verificar duplicados
print(f"Número de linhas duplicadas: {dados.duplicated().sum()}")

# Remover duplicados
dados.drop_duplicates(inplace=True)
print(f"Formato após remoção: {dados.shape}")
```

```
Número de linhas duplicadas: 1
Formato após remoção: (998, 12)
```

4 - Corrigir os dados faltantes (**NANs**) e preencher as lacunas com valores adequados.

In [163…
```
dados.isnull().sum()
```

Out[163…

	0
id	0
pontos	0
estado	0
genero	8
idade	0
bens	0
saldo na conta	0
produtos	0
possui cartao	0
ativo	0
salario anual	7
saiu	0

dtype: int64

In [164…
```
dados.fillna(0) #ver onde estao os valores nulos em qual coluna.
```

Out[164…

	id	pontos	estado	genero	idade	bens	saldo na conta	produtos	possui cartao	ativo	
0	1	619	PI	Feminino	42	2	0	1	1	1	1
1	2	608	CE	Feminino	41	1	8380786	1	0	1	1
2	3	502	PI	Feminino	42	8	1596608	3	1	0	1
3	4	699	PI	Feminino	39	1	0	2	0	0	
4	5	850	CE	Feminino	43	2	12551082	1	1	1	
...	...	...	...	...	...	...	...	...	...	...	
994	996	838	CE	Masculino	43	9	12310588	2	1	0	1
995	997	610	CE	Masculino	29	9	0	3	0	1	
996	998	811	CE	Masculino	44	3	0	2	0	1	
997	999	587	CE	Masculino	62	7	12128627	1	0	1	
998	1000	811	MA	Feminino	28	4	16773882	2	1	1	

998 rows × 12 columns

In [165…
```
dados = dados.fillna(0) # colocar zero nas colunas com valores nulos
```

In [166…
```
dados.isnull().sum()
```

Out[166…

	0
id	0
pontos	0
estado	0
genero	0
idade	0
bens	0
saldo na conta	0
produtos	0
possui cartao	0
ativo	0
salario anual	0
saiu	0

dtype: int64

In [167…
```
# Verificar quantidade de valores nulos
print(dados.isnull().sum())

# Preencher valores nulos
# Estratégia: para numéricos, preencher com a média; para categóricos, pr
for col in dados.columns:
    if dados[col].isnull().sum() > 0:
        if dados[col].dtype == 'object':
            dados[col].fillna(dados[col].mode()[0], inplace=True)
        else:
            dados[col].fillna(dados[col].mean(), inplace=True)
```

```
id                0
pontos            0
estado            0
genero            0
idade             0
bens              0
saldo na conta    0
produtos          0
possui cartao     0
ativo             0
salario anual     0
saiu              0
dtype: int64
```

In [ ]:
```
#dados.query('genero == 0 | ativo == 0') / remover colunas
```

In [168…
```
dados
```

Out[168…

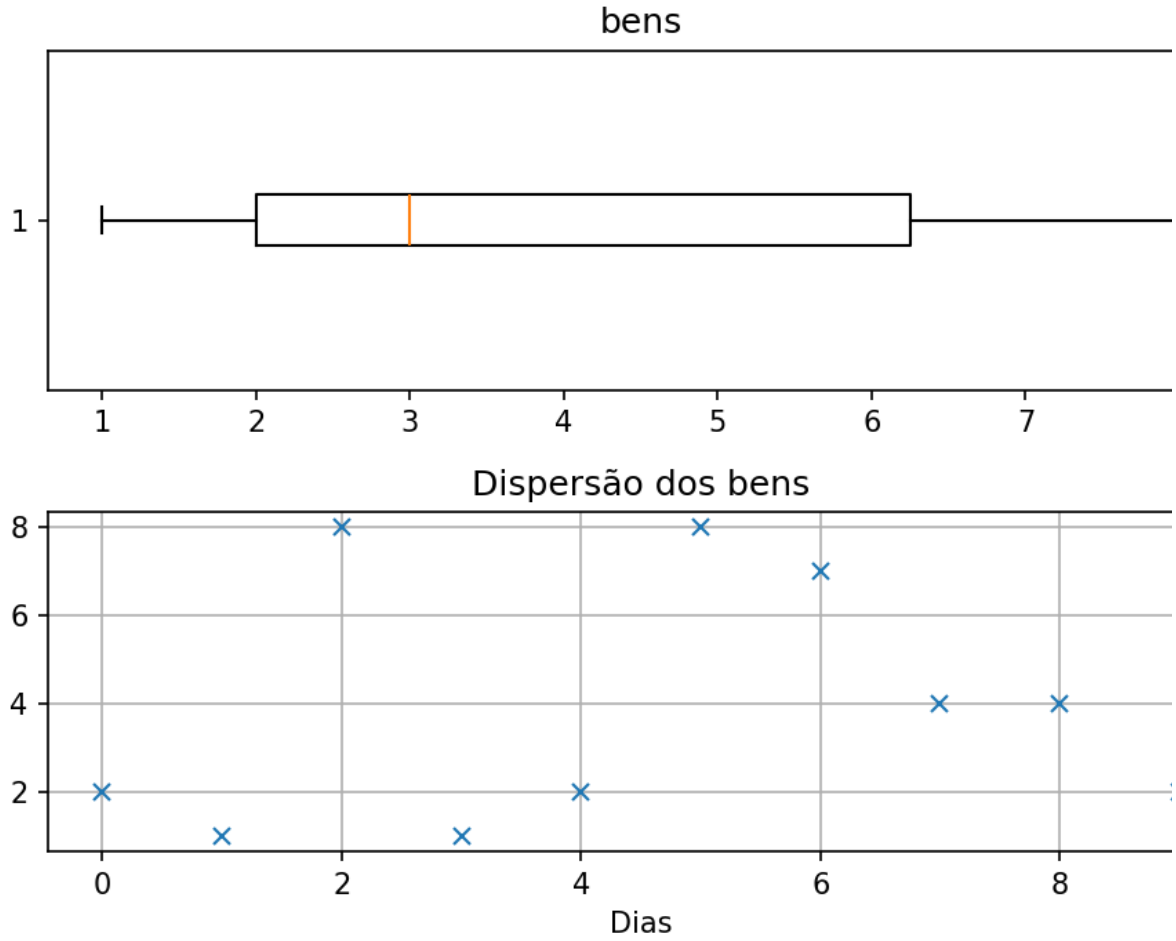
	id	pontos	estado	genero	idade	bens	saldo na conta	produtos	possui cartao	ativo	
0	1	619	PI	Feminino	42	2	0	1	1	1	1
1	2	608	CE	Feminino	41	1	8380786	1	0	1	1
2	3	502	PI	Feminino	42	8	1596608	3	1	0	1
3	4	699	PI	Feminino	39	1	0	2	0	0	
4	5	850	CE	Feminino	43	2	12551082	1	1	1	
...	...	...	...	...	...	...	...	...	...	...	
994	996	838	CE	Masculino	43	9	12310588	2	1	0	1
995	997	610	CE	Masculino	29	9	0	3	0	1	
996	998	811	CE	Masculino	44	3	0	2	0	1	
997	999	587	CE	Masculino	62	7	12128627	1	0	1	
998	1000	811	MA	Feminino	28	4	16773882	2	1	1	

998 rows × 12 columns

5 - Tratar os outliers, substituindo-os pelos valores adequados.

Metodo 2 outra referência:

In [26]:
```
# Dados de temperatura de exemplo (você pode substituir pelos seus)
dados = [2, 1, 8, 1, 2, 8, 7, 4, 4, 2]

# Conversão para array numpy
temp = np.array(dados)

# Plotagem
plt.figure(dpi=150)
plt.subplot(211)
plt.boxplot(temp, vert=False)
plt.title("bens")

plt.subplot(212)
plt.plot(temp, 'x')
plt.grid()
plt.title("Dispersão dos bens")
plt.xlabel("Dias")
plt.ylabel("bens")

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



Métodos para detecção de Outliers

- Univariado
  - Intervalo interquartil
  - Z-score
- Multivariado
  - Envelope Elíptico
  - Isolation Forest

Remoção e Manutenção de Outliers

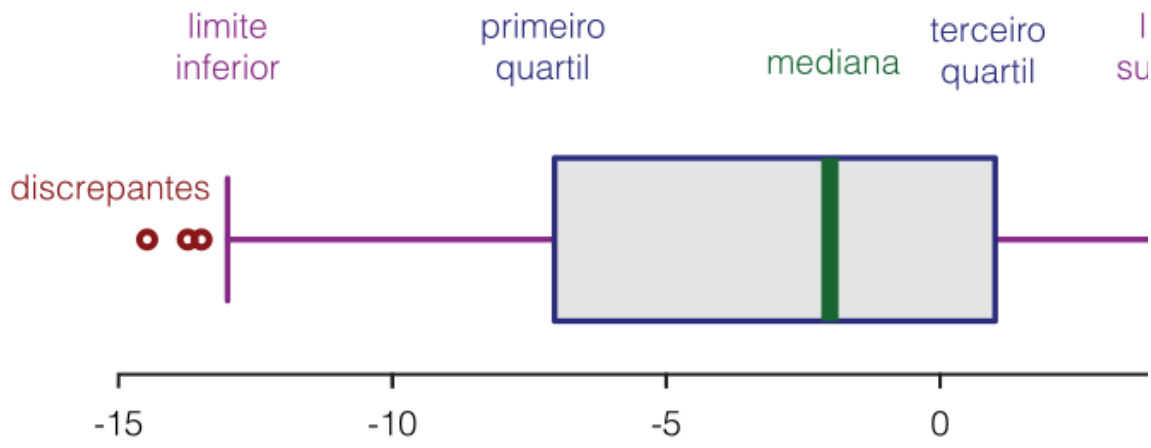
Após a detecção os outliers podem ser removidos ou mantidos.

Essa decisão dependerá das características do problema que se pretende resolver.

Caso Univariado

Remoção e Obtenção dos Outliers para uma única variável

Intervalor interquartil



Referêcia: Aqui

1º Passo: Calcular os quartis

- Q1: 25º percentil (quartil inferior)
- Q3: 75º percentil (quartil superior)

In [169…
```
Q1 = np.percentile(temp, 25)
Q3 = np.percentile(temp, 75)

print('Q1: %.2f' %Q1)
print('Q3: %.2f' %Q3)
```

```
Q1: 2.00
Q3: 6.25
```

2º Passo: Calcular Intervalor interquartil (IIQ) e os limites inferior e superior, utilizando uma cons
que geralmente recebe o valor 1,5.

In [170…
```
C   = 1.5
IIQ = Q3 - Q1
LI  = Q1 - C*IIQ
LS  = Q3 + C*IIQ

print('IIQ: %.2f' %IIQ)
print('Limites:')
print(' Inferior: %.2f' %LI)
print(' Superior: %.2f' %LS)
```

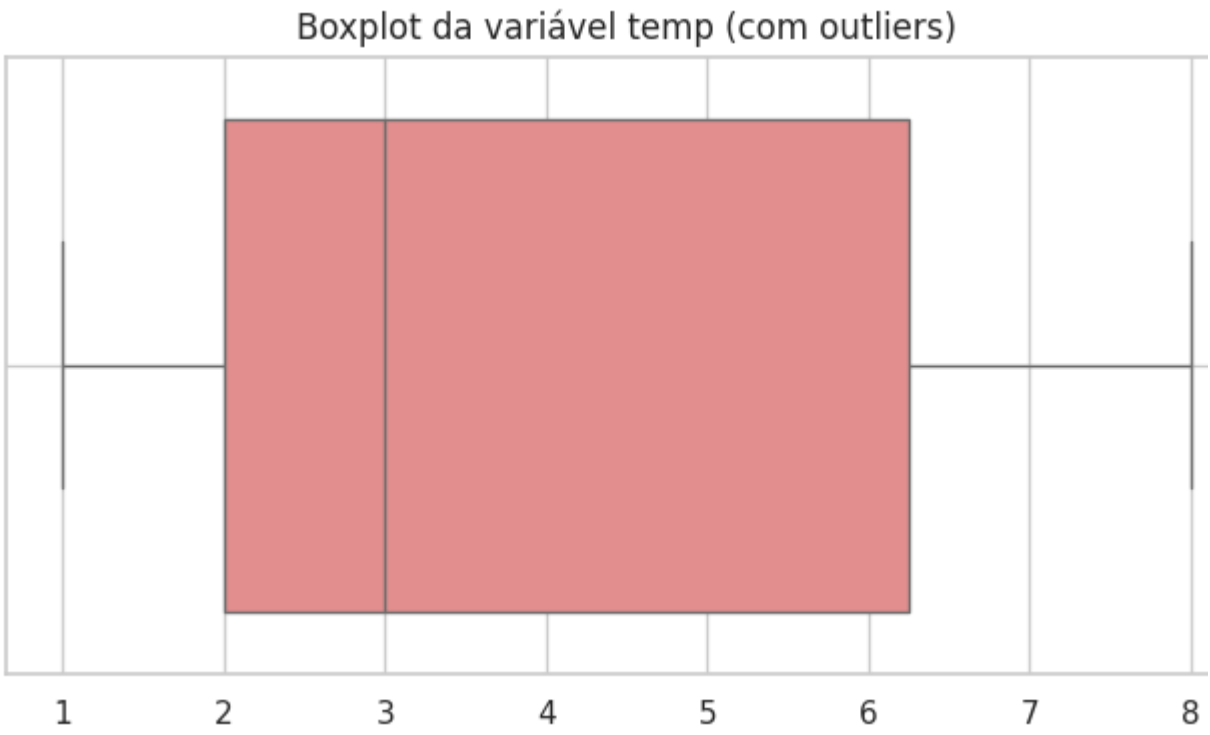
```
IIQ: 4.25
Limites:
 Inferior: -4.38
 Superior: 12.62
```

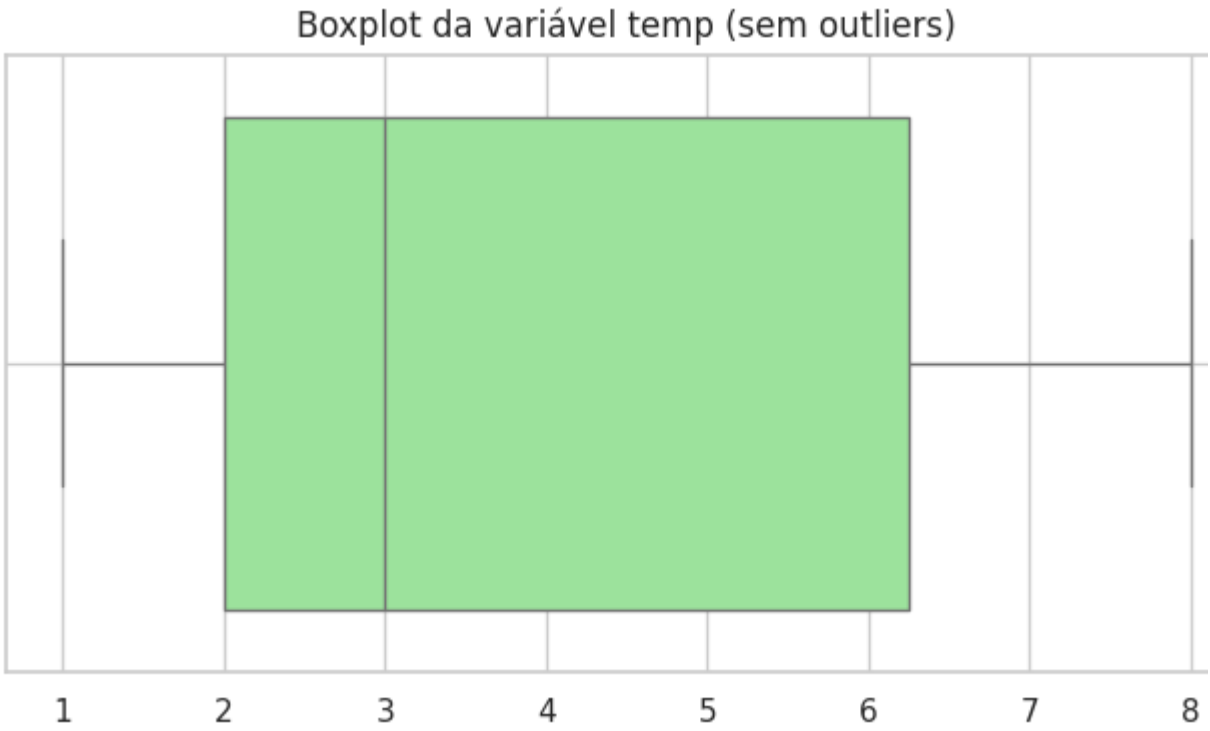
3º Passo: Obtém e Remove os Outliers, que são os valores abaixo do limite inferior ***LI*** ou acima limite superior ***LS***.

In [171…
```
#Obtém e Remove os Outliers
temp_sem_outliers_IIQ = []
outliers              = []
for t in temp:
    if t > LS or t < LI:
        outliers.append(t)
    else:
        temp_sem_outliers_IIQ.append(t)

print('Outliers: ')
print(outliers)
print('Sem Outliers: ')
print(temp_sem_outliers_IIQ)
```

```
Outliers:
[]
Sem Outliers:
[np.int64(2), np.int64(1), np.int64(8), np.int64(1), np.int64(2), np.int64(8),
np.int64(7), np.int64(4), np.int64(4), np.int64(2)]
```

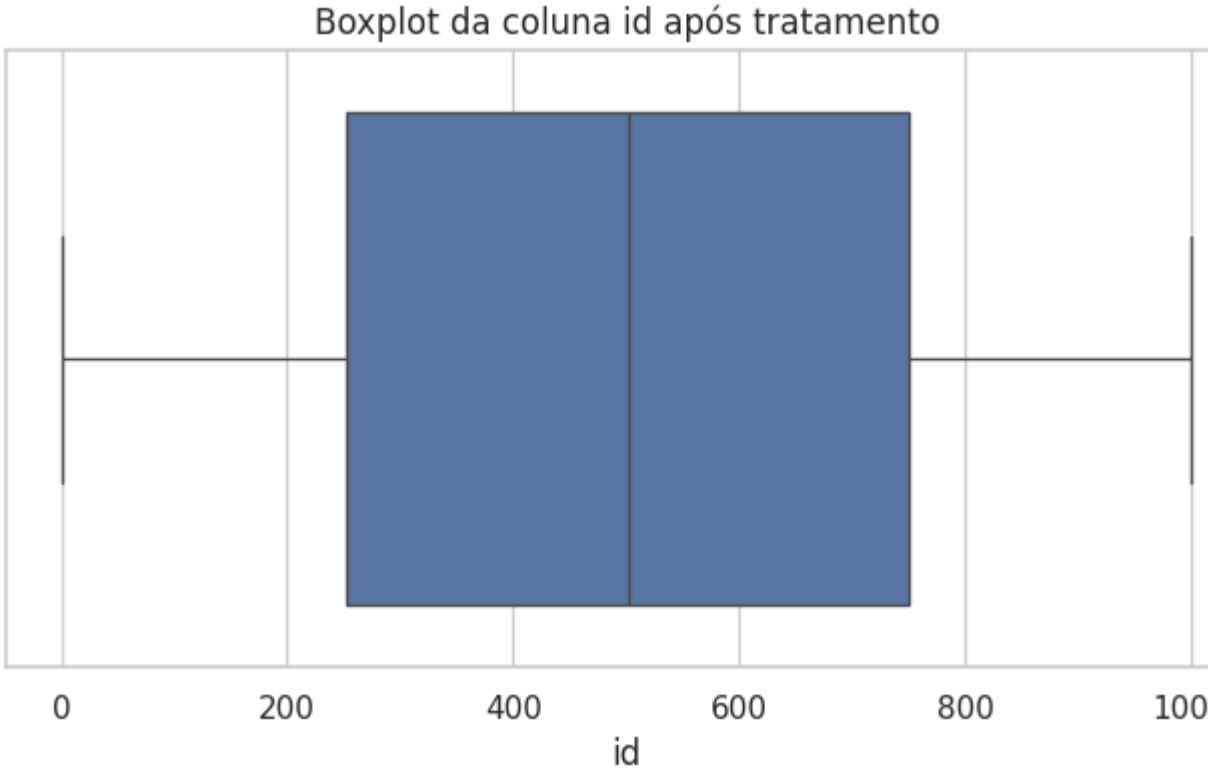
In [173…
```
#import matplotlib.pyplot as plt
#import seaborn as sns

# Boxplot com outliers
plt.figure(figsize=(8, 4))
sns.boxplot(x=temp, color='lightcoral')
plt.title('Boxplot da variável temp (com outliers)')
#plt.xlabel('Temperatura')
plt.grid(True)
plt.show()

# Boxplot sem outliers
plt.figure(figsize=(8, 4))
sns.boxplot(x=temp_sem_outliers_IIQ, color='lightgreen')
plt.title('Boxplot da variável temp (sem outliers)')
#plt.xlabel('Temperatura')
plt.grid(True)
plt.show()
```

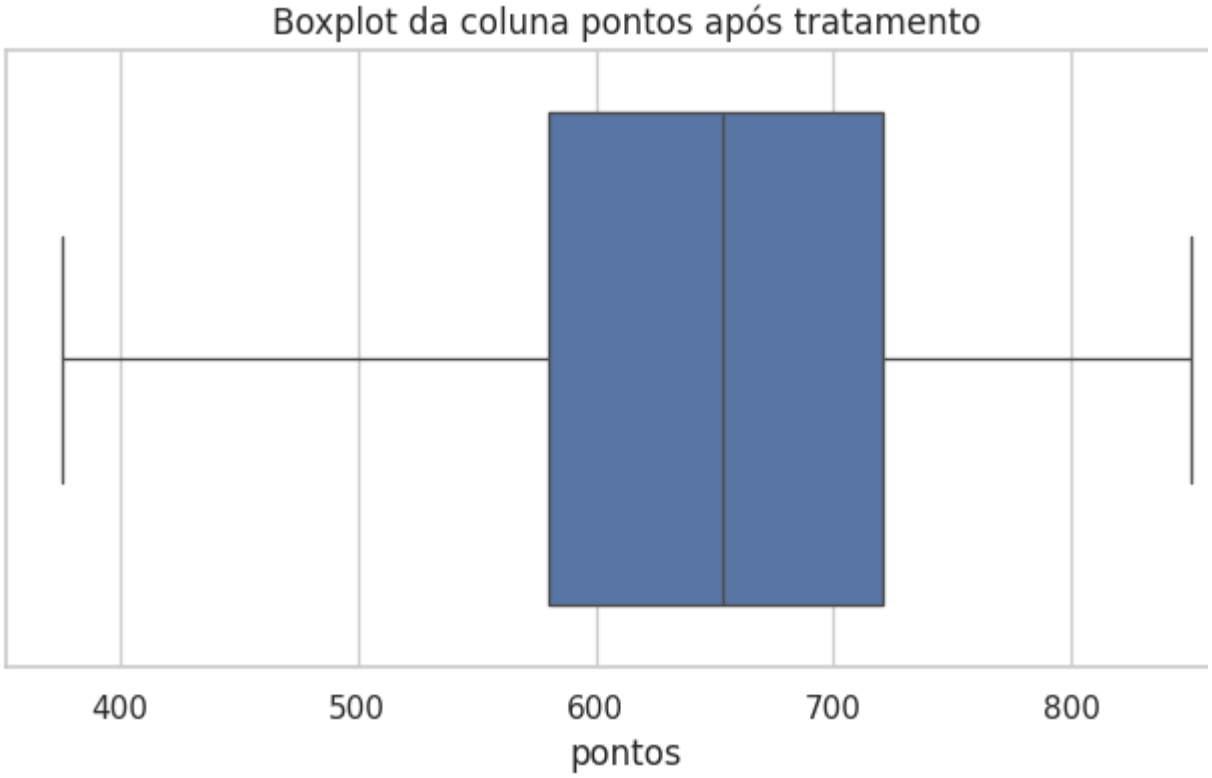


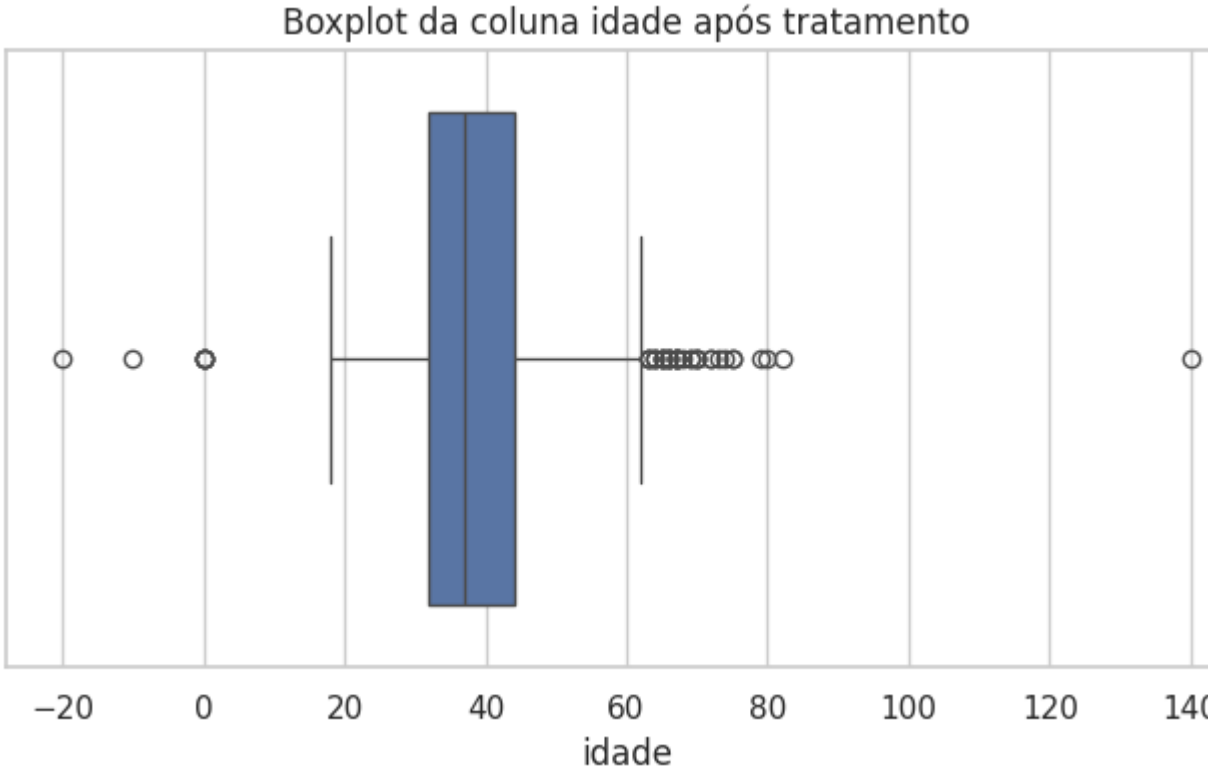


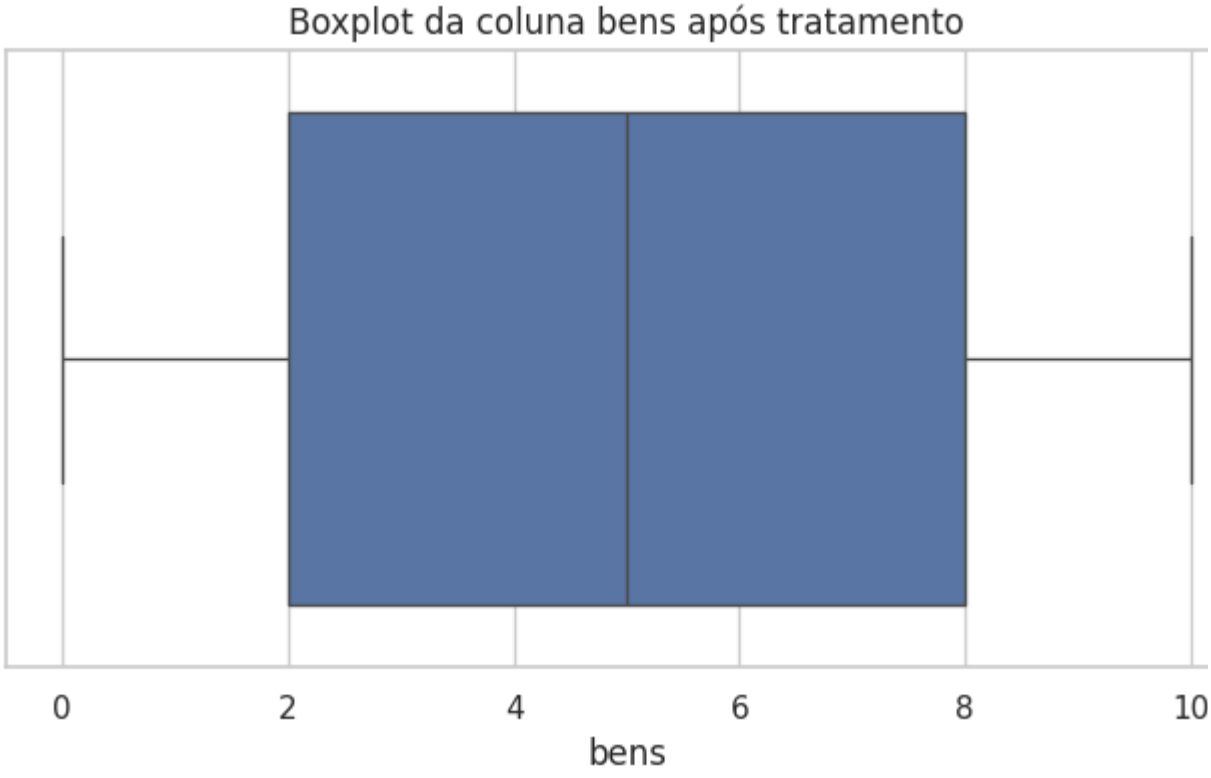
In [175…

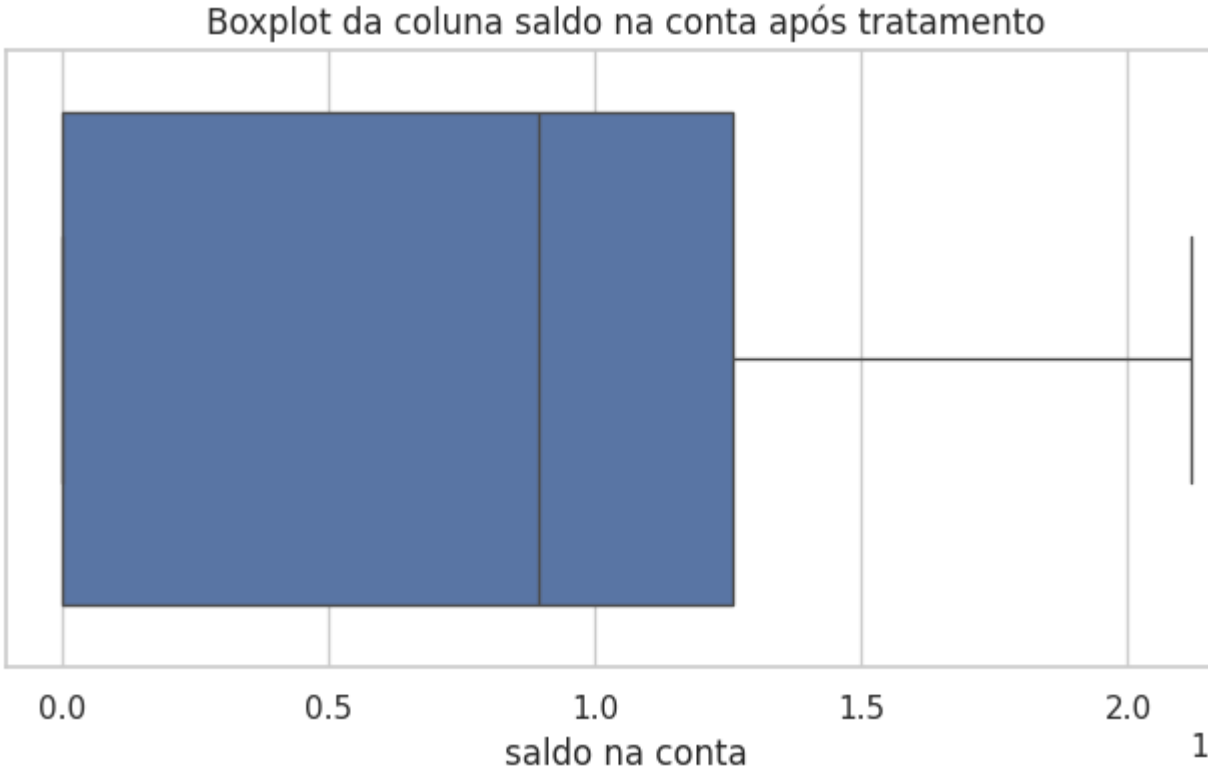
```
num_cols = dados.select_dtypes(include='number').columns  # Define as col
for col in num_cols:
    plt.figure(figsize=(8, 4))
    sns.boxplot(x=dados[col])
    plt.title(f'Boxplot da coluna {col} após tratamento')
    plt.show()
```

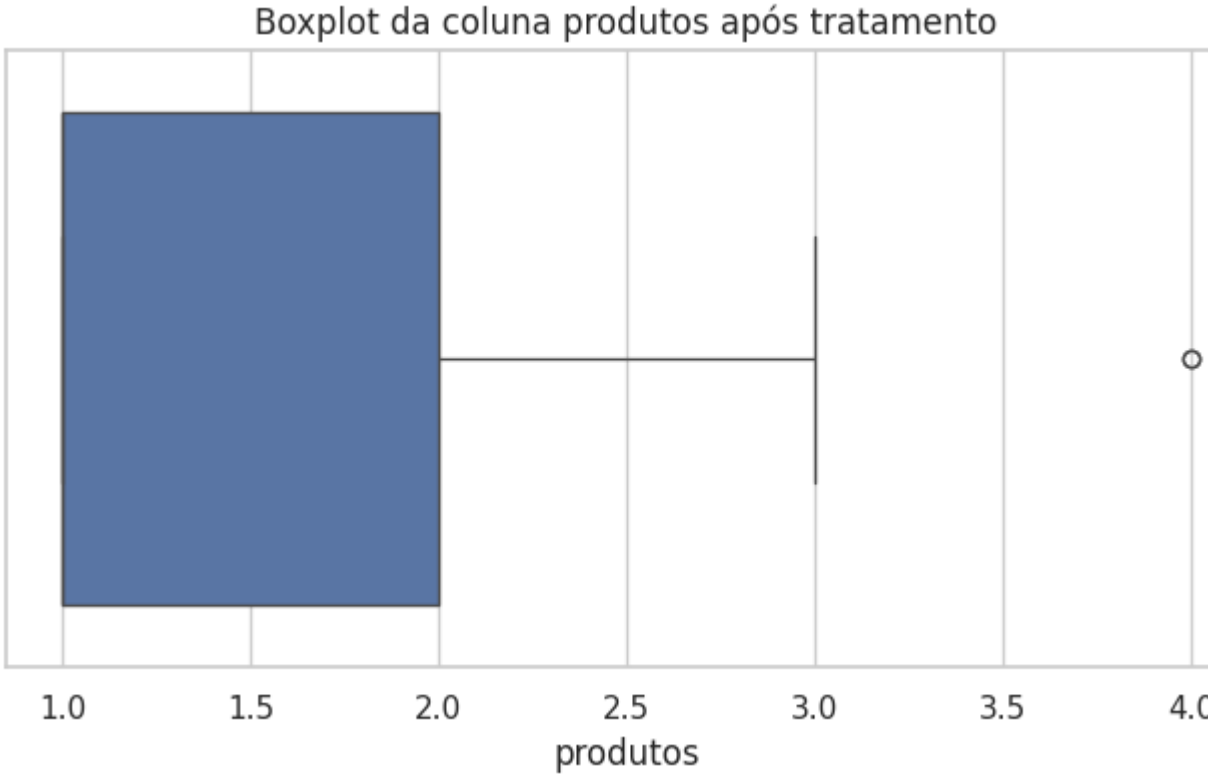

Boxplot da coluna id após tratamento
0
200
400
600
800
100
id

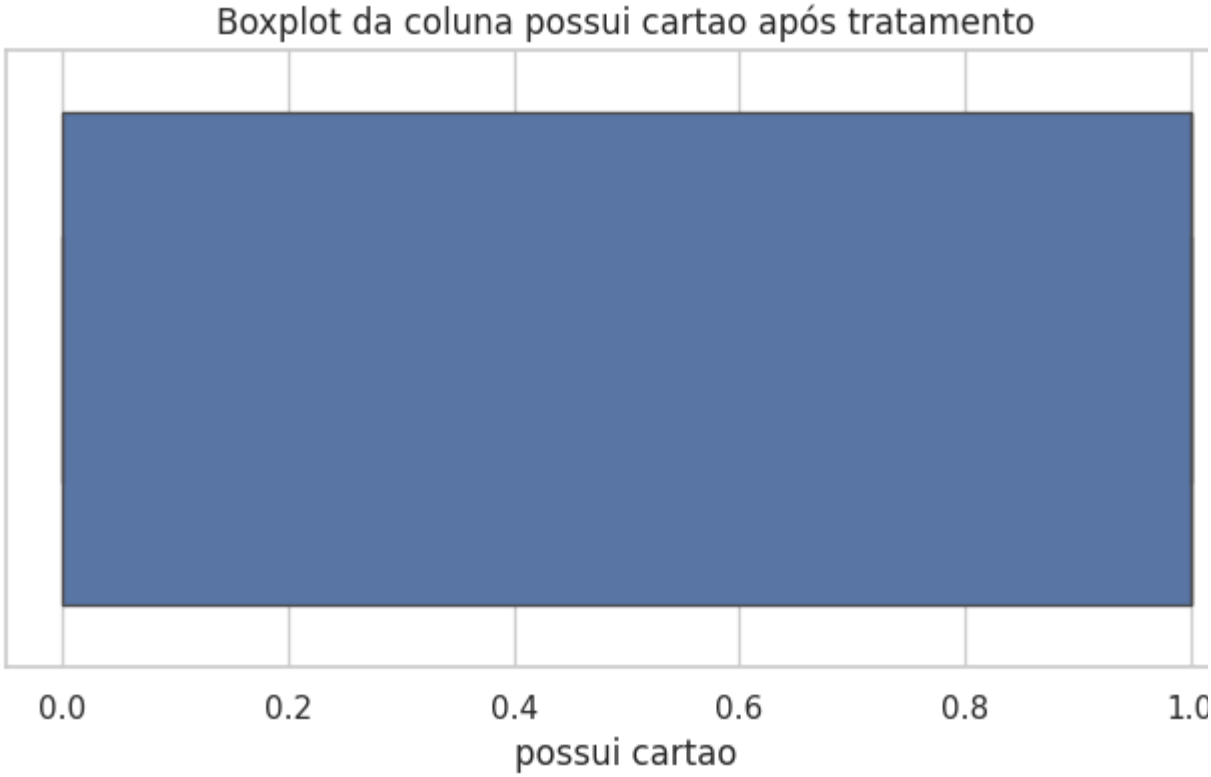

Boxplot da coluna pontos após tratamento
400
500
600
700
800
pontos

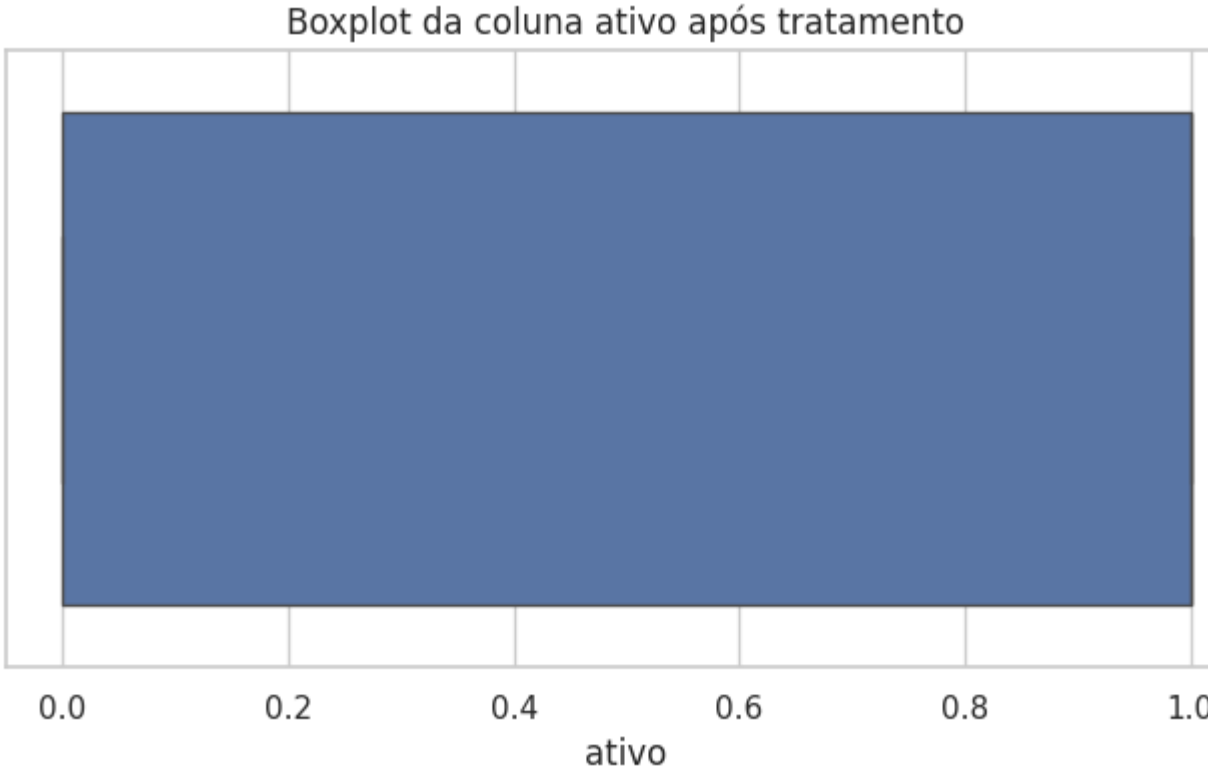

Boxplot da coluna idade após tratamento
−20
0
20
40
60
80
100
120
140
idade

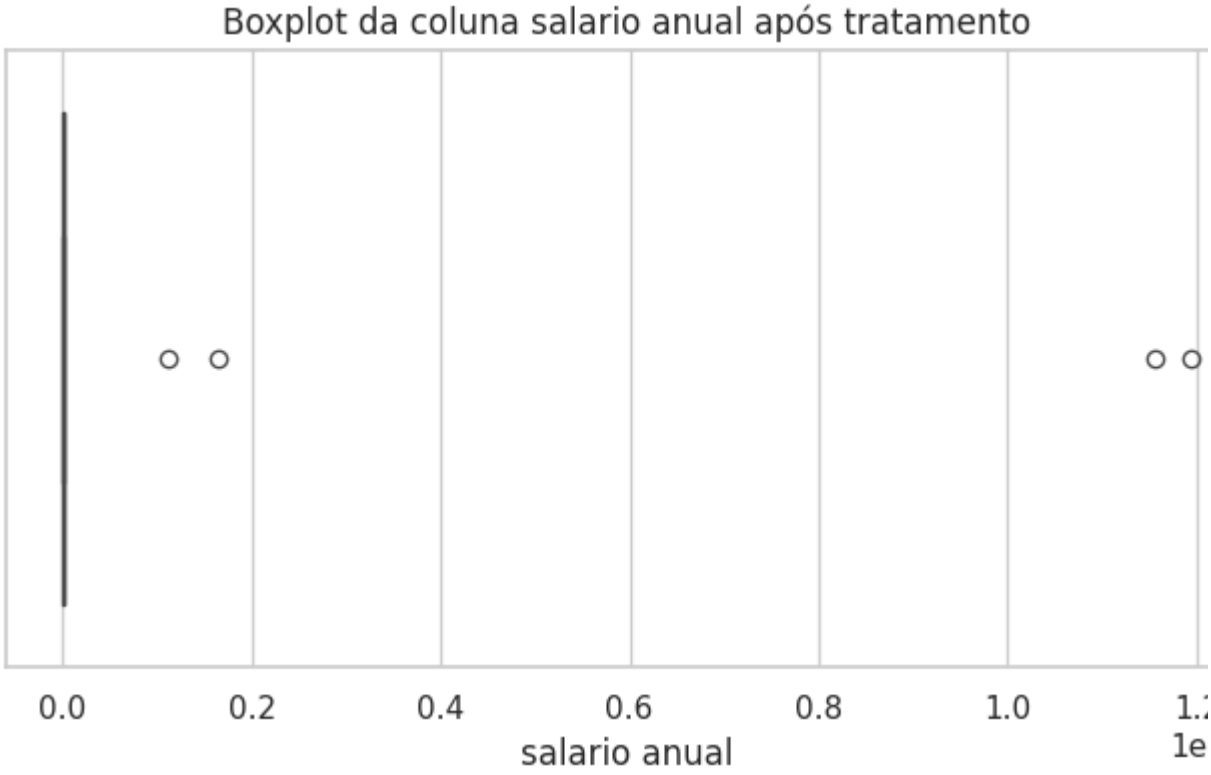

Boxplot da coluna bens após tratamento
0
2
4
6
8
10
bens


Boxplot da coluna saldo na conta após tratamento
0.0
0.5
1.0
1.5
2.0
saldo na conta


Boxplot da coluna produtos após tratamento
1.0
1.5
2.0
2.5
3.0
3.5
produtos


Boxplot da coluna possui cartao após tratamento
0.0
0.2
0.4
0.6
0.8
1.0
possui cartao


Boxplot da coluna ativo após tratamento
0.0
0.2
0.4
0.6
0.8
1.0
ativo


Boxplot da coluna salario anual após tratamento
0.0
0.2
0.4
0.6
0.8
1.0
salario anual
1e


Boxplot da coluna saiu após tratamento
0.0
0.2
0.4
0.6
0.8
1.0
saiu

Z-score

Z-score (Z) é uma medida relacionada a distância que um ponto está da média, em função des desvios padrão, ist

$$Z = \frac{X - \mu}{\sigma}$$

Se $Z < 0$ o dado observado está abaixo da média.

Se $Z > 0$ o dado observado está acima da média.

Valores de Z-score - Distribuição Normal

Percentual dos dados	Desvios padrão
68%	± 1
95%	± 2
99,7%	± 3

Cuidados a tomar:

- Deve ser utilizada se a distribuição dos dados for Gaussiana (Normal).
- Em pequenos conjuntos de dados (menor que 10) retorna valores não confiáveis
- É sensível a muitos Outliers. Os Z-scores tornam-se menos extremos.

1º Passo: Calcular o Z-score

In [30]:
```
z_scores = (temp - np.mean(temp)) / np.std(temp)

print('Z-scores: ', z_scores)
```

```
Z-scores:  [-0.71355998 -1.08911786  1.53978732 -1.08911786 -0.71355998  1.53
  1.16422944  0.03755579  0.03755579 -0.71355998]
```

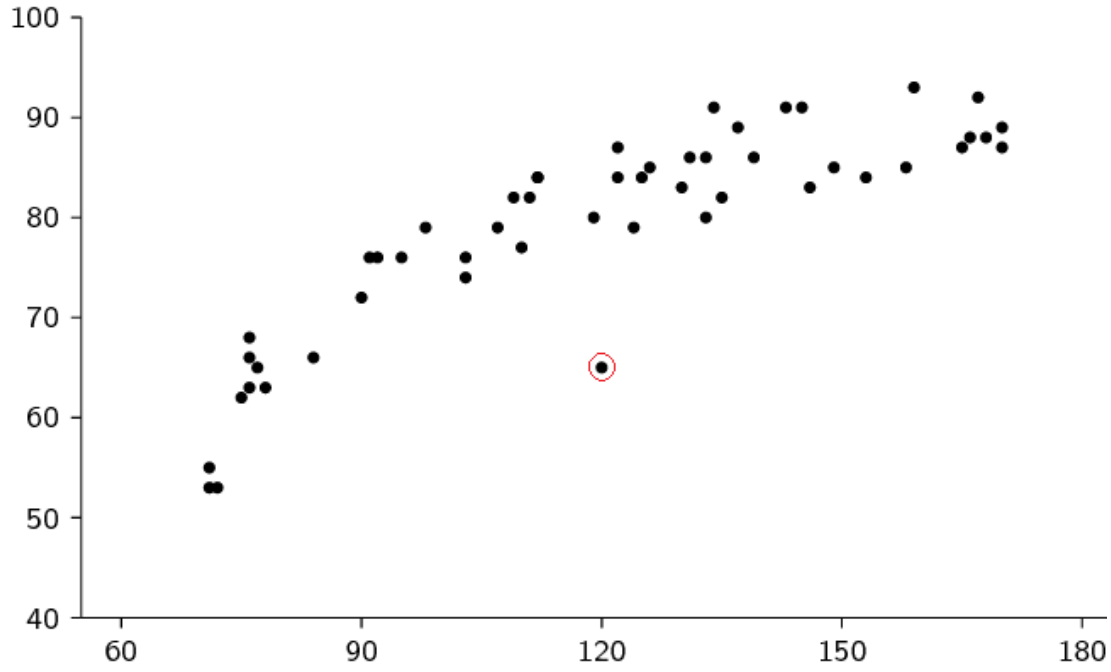
2º Passo: Remoção e Obtenção dos Outliers, ou seja, os Z-scores que ultrapassam o limiar ± 3

In [31]:
```
limiar = 3 #3 desvios padrão
temp_sem_outliers_Z_score = []
outliers                  = []
for i, z_score in enumerate(z_scores):
    if np.abs(z_score) > limiar:
        outliers.append(temp[i])
    else:
        temp_sem_outliers_Z_score.append(temp[i])

print('Outliers: ')
print(outliers)
print('Sem Outliers: ')
print(temp_sem_outliers_Z_score)
```

```
Outliers:
[]
Sem Outliers:
[np.int64(2), np.int64(1), np.int64(8), np.int64(1), np.int64(2), np.int64(8),
np.int64(7), np.int64(4), np.int64(4), np.int64(2)]
```

Caso Multivariado

Considera-se a remoção/detecção de Outliers em espaços de entrada de mais alta dimensão



In [32]:
```
from sklearn.datasets import load_diabetes

X, y = load_diabetes(return_X_y = True)

print('Nº Variáveis: %i' %X.shape[1])
print('Nº Amostras: %i' %X.shape[0])
```

```
Nº Variáveis: 10
Nº Amostras: 442
```

In [33]:
```
from sklearn.covariance import EllipticEnvelope

#Retorna -1 se a amostra é Outlier e 1 caso contrário
#O parâmetro 'contamination' é a proporção de Outliers no conjunto de dad
#Varia entre (0,0.5]
out_EE = EllipticEnvelope(contamination=0.1).fit_predict(X)

outliers = X[out_EE == -1, :]
X_out    = X[out_EE == 1, :]

print('Nº Outliers: %i' % outliers.shape[0])
print('\nDados sem Outliers')
print('Nº Variáveis: %i' %X_out.shape[1])
print('Nº Amostras: %i' %X_out.shape[0])
```

```
Nº Outliers: 45

Dados sem Outliers
Nº Variáveis: 10
Nº Amostras: 397
```

Regressão Linear

In [34]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
```

In [35]:
```
#Separa os dados de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,

#Aplica a Regressão Linear
lr = LinearRegression().fit(X_train, y_train)
y_pred = lr.predict(X_test)

print('Regressão Linear com Outliers')
print('MAE: %.4f' %mean_absolute_error(y_test, y_pred))
```

```
Regressão Linear com Outliers
MAE: 42.0026
```

In [36]:
```
from sklearn.ensemble import IsolationForest

#Retorna -1 se a amostra é Outlier e 1 caso contrário
out_IF = IsolationForest(contamination=0.1).fit_predict(X)

outliers = X[out_IF != 1 , :]
X_out    = X[out_IF != -1, :]

print('Nº Outliers: %i' % outliers.shape[0])
print('\nDados sem Outliers')
print('Nº Variáveis: %i' %X_out.shape[1])
print('Nº Amostras: %i' %X_out.shape[0])
```

```
Nº Outliers: 45

Dados sem Outliers
Nº Variáveis: 10
Nº Amostras: 397
```

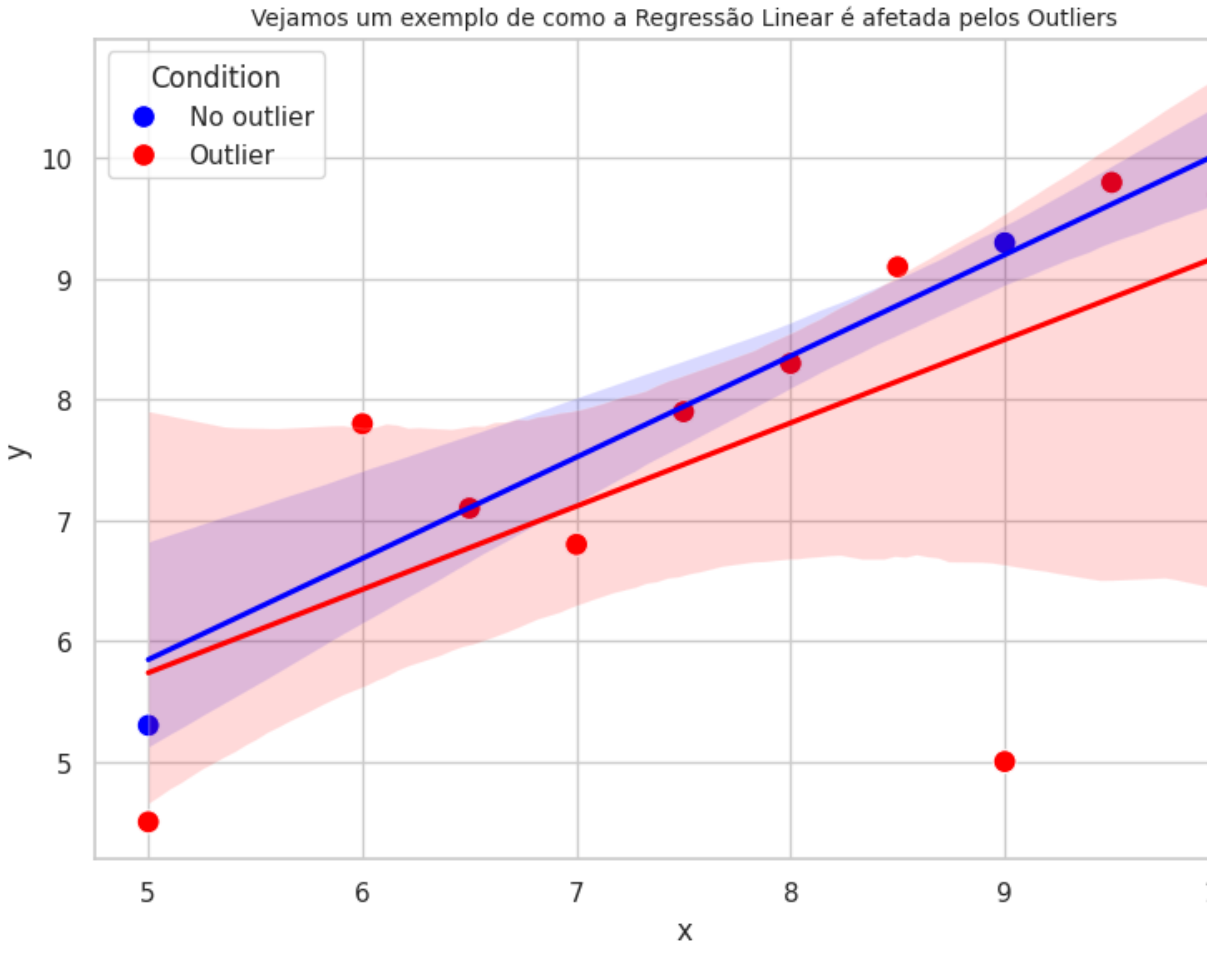
In [42]:
```
#Aplica a remoção de Outliers nos dados de treinamento
out_IF = IsolationForest(contamination=0.1).fit_predict(X_train)

X_train_wo = X_train[out_IF == 1, :]
y_train_wo = y_train[out_IF == 1]

#Aplica a Regressão Linear nos dados sem Outliers
lr = LinearRegression().fit(X_train_wo, y_train_wo)
y_pred_wo = lr.predict(X_test)

print('Regressão Linear sem Outliers')
print('MAE: %.4f' %mean_absolute_error(y_test, y_pred_wo))
```

```
Regressão Linear sem Outliers
MAE: 41.5361
```

Gráfico

In [41]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Dados sem outlier
```

```python
x = np.array([5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10])
y_no_outlier = np.array([5.3, 7.8, 7.1, 6.8, 7.9, 8.3, 9.1, 9.3, 9.8, 9.7

# Dados com outlier (modificando apenas dois pontos)
y_outlier = y_no_outlier.copy()
y_outlier[0] = 4.5   # outlier baixo
y_outlier[7] = 5.0   # outro outlier

# Criar DataFrame
df_no_outlier = pd.DataFrame({'x': x, 'y': y_no_outlier, 'Condition': 'No
df_outlier = pd.DataFrame({'x': x, 'y': y_outlier, 'Condition': 'Outlier'
df = pd.concat([df_no_outlier, df_outlier])

# Plot
plt.figure(figsize=(8, 6))
sns.set(style="whitegrid")

# Gráfico de dispersão
sns.scatterplot(data=df, x='x', y='y', hue='Condition', palette={'No outl

# Regressões
sns.regplot(x=x, y=y_no_outlier, scatter=False, color='blue')
sns.regplot(x=x, y=y_outlier, scatter=False, color='red')

plt.title("Vejamos um exemplo de como a Regressão Linear é afetada pelos
plt.xlabel("x")
plt.ylabel("y")
plt.legend(title="Condition")
plt.tight_layout()
plt.show()
```



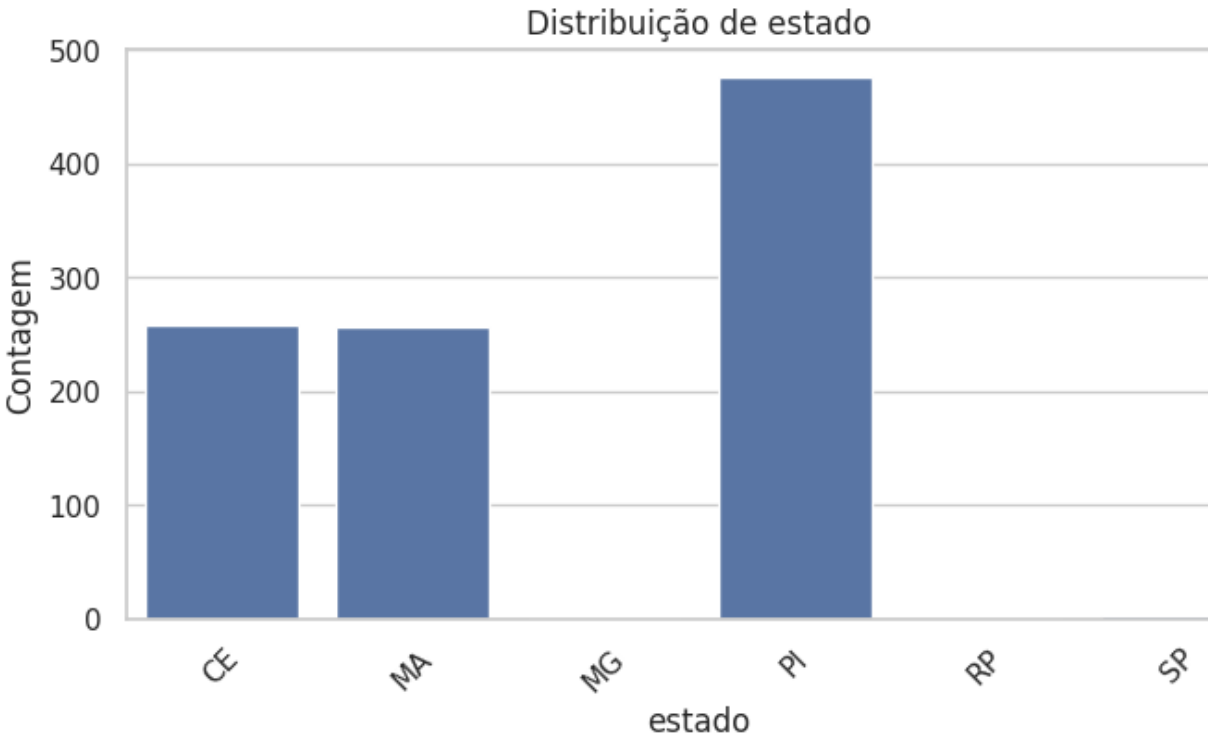
6 - Explore colunas categóricas, utilizando função **groupby()** e outras, visando aplicar os gráfic visualizações e a compreensão desses dados.

In [126…
```
dados.groupby('pontos')
```

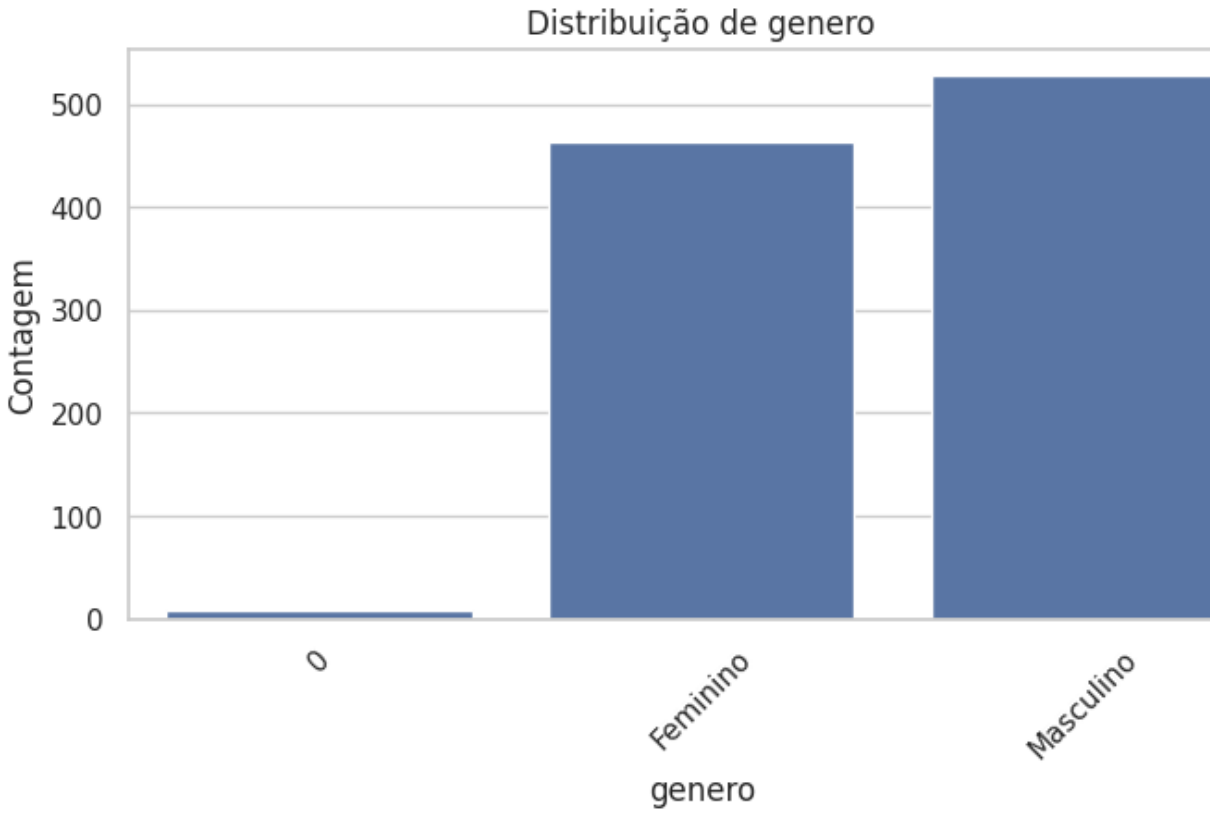
Out[126…
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7ff929fffd90>
```

In [127…
```
# Colunas categóricas
categorical_cols = ['estado', 'genero', 'possui cartao', 'ativo', 'saiu',

# Análise de frequência com groupby
for col in categorical_cols:
    print(f"\nDistribuição de {col}:")

    # Usando groupby para calcular as proporções (equivalente a value_cou
    freq_table = dados.groupby(col).size().div(len(dados)).reset_index(na
    print(freq_table.set_index(col).T)

    # Gráfico de barras usando groupby
    plt.figure(figsize=(8,4))

    # Agrupar e contar os valores
    plot_data = dados.groupby(col).size().reset_index(name='contagem')

    # Criar o gráfico de barras
    sns.barplot(data=plot_data, x=col, y='contagem')
    plt.title(f'Distribuição de {col}')
    plt.xticks(rotation=45)
    plt.ylabel('Contagem')
    plt.show()
```

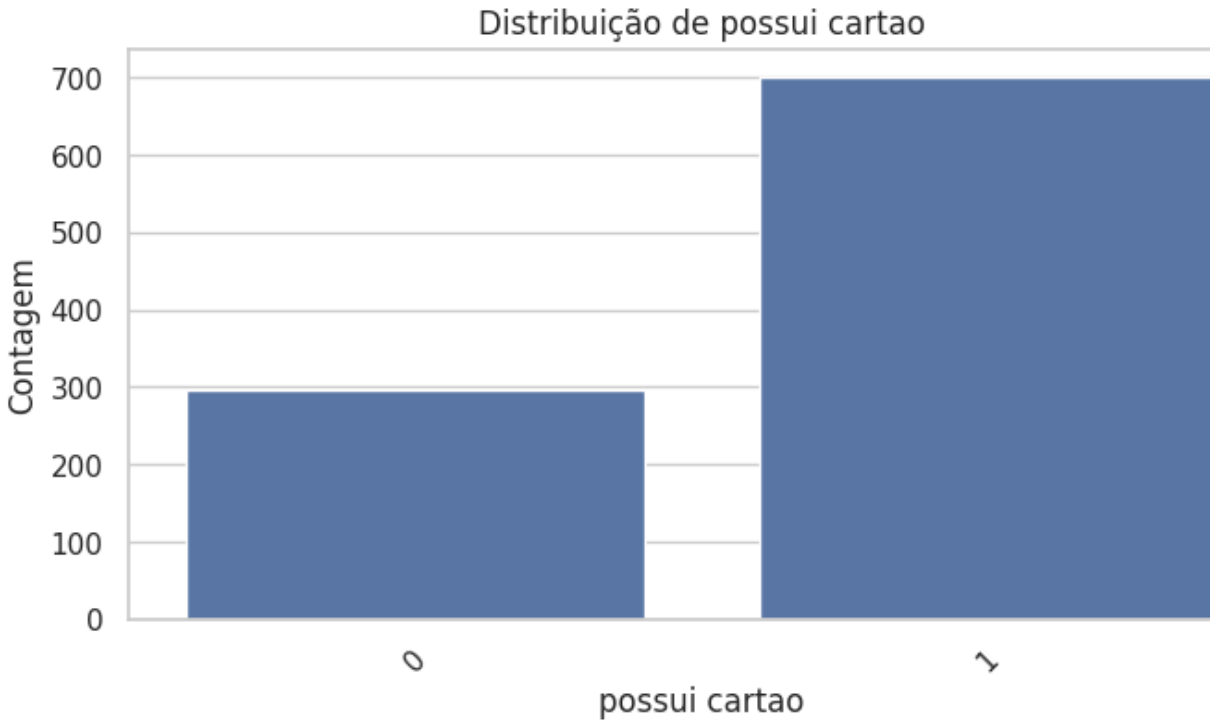
```
Distribuição de estado:
estado           CE        MA        MG        PI        RP        SP
proporcao  0.258517  0.257515  0.001002  0.477956  0.001002  0.004008
```



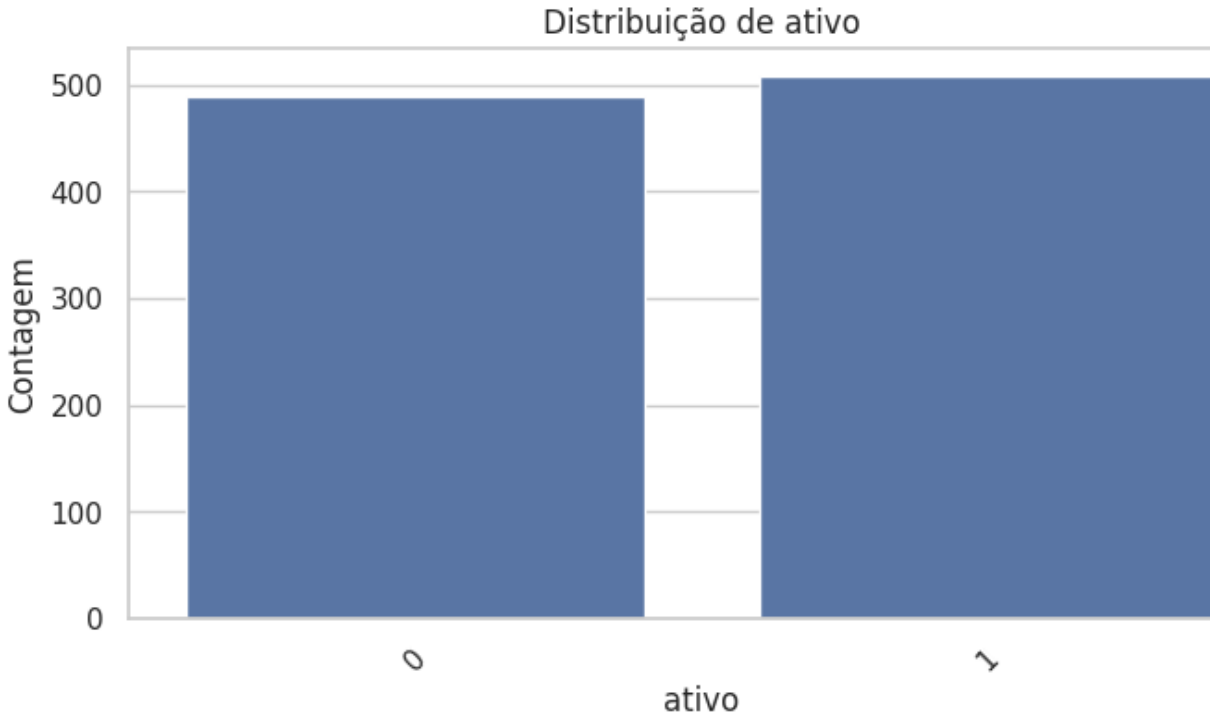
```
Distribuição de genero:
genero            0  Feminino  Masculino
proporcao  0.008016  0.463928   0.528056
```




```
Distribuição de possui cartao:
possui cartao         0         1
proporcao      0.296593  0.703407
```

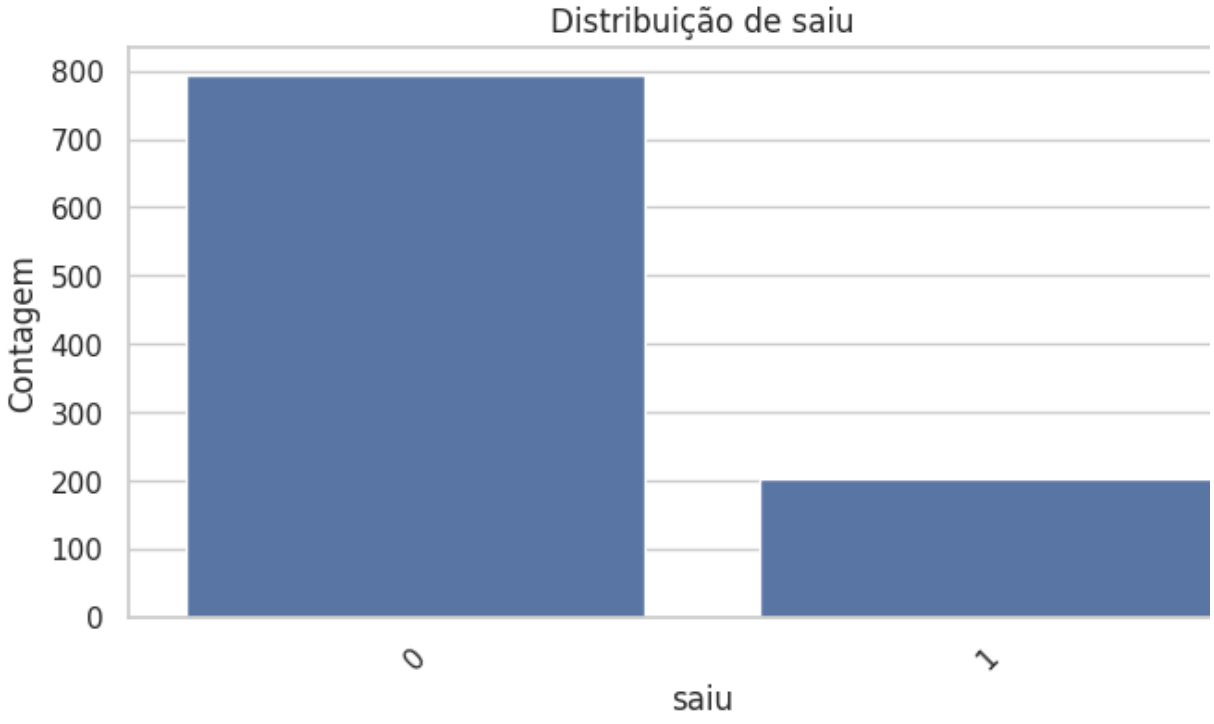



```
Distribuição de ativo:
ativo            0       1
proporcao  0.48998  0.51002
```



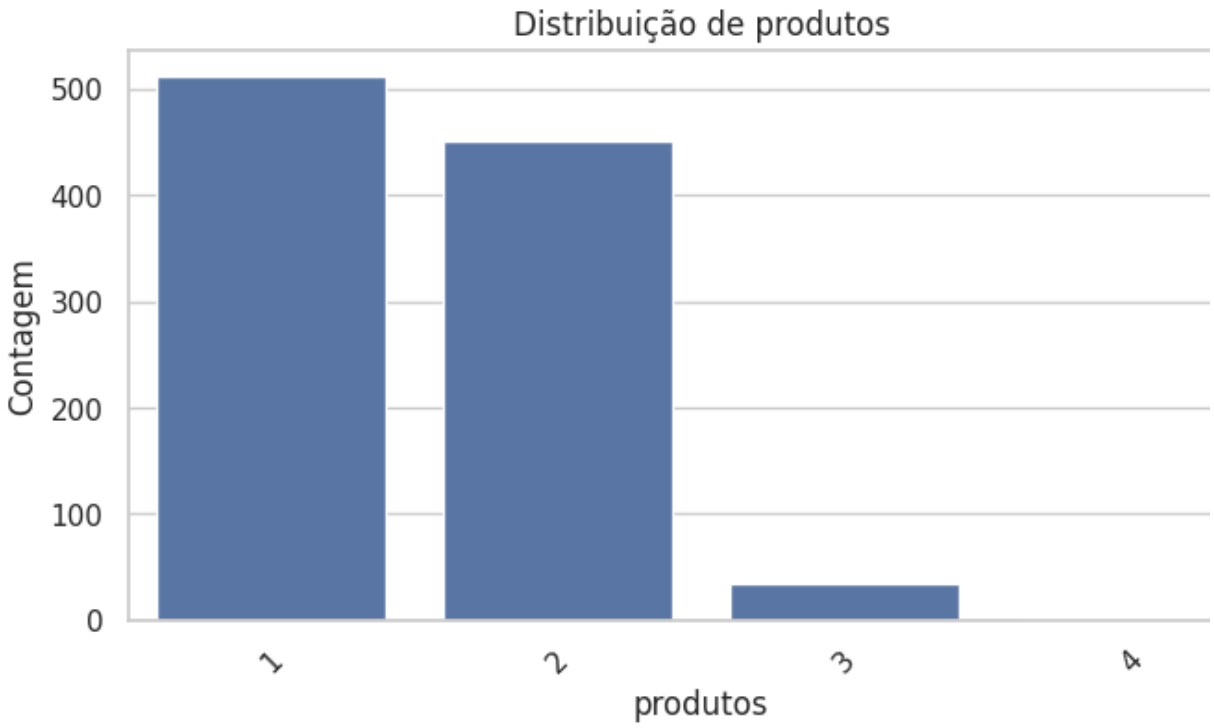
Distribuição de saiu:

saiu	0	1
proporcao	0.796593	0.203407



Distribuição de produtos:

produtos	1	2	3	4
proporcao	0.512024	0.451904	0.034068	0.002004

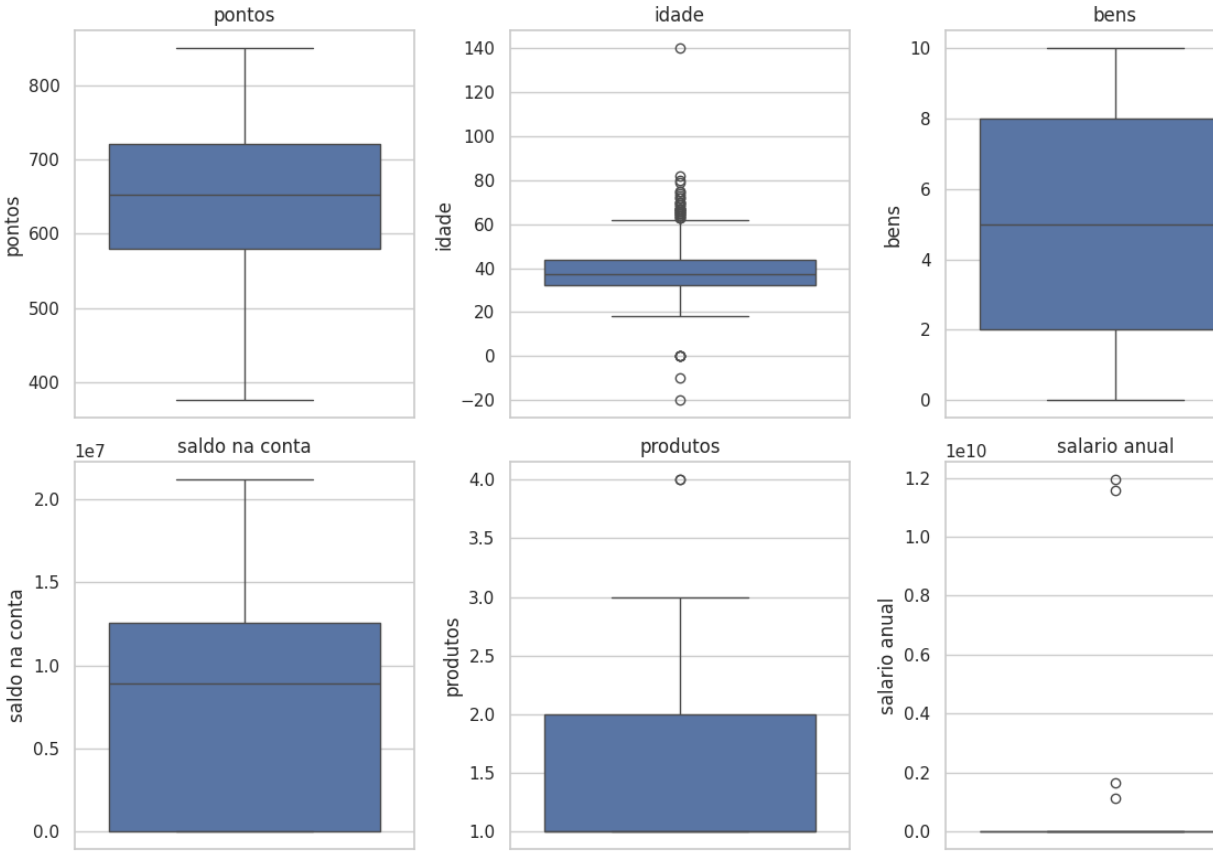


7 - Explore colunas numéricas, utilizando a função **describe()**, faça um boxplot ou outro gráfico achar necessário, utilizando as bibliotecas pandas e ** seaborn**;

In [177…

```
# Colunas numéricas
numeric_cols = ['pontos', 'idade', 'bens', 'saldo na conta', 'produtos',

# Estatísticas descritivas
print(dados[numeric_cols].describe())

# Boxplots para visualizar outliers
plt.figure(figsize=(12, 8))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(y=dados[col])
    plt.title(col)
plt.tight_layout()
plt.show()
```

```
           pontos       idade        bens  saldo na conta    produtos  \
count  998.000000  998.000000  998.000000    9.980000e+02  998.000000
mean   648.605210   38.907816    5.073146    7.162423e+06    1.526052
std     98.312117   11.406570    2.926320    6.314508e+06    0.574293
min    376.000000  -20.000000    0.000000    0.000000e+00    1.000000
25%    580.000000   32.000000    2.000000    0.000000e+00    1.000000
50%    653.000000   37.000000    5.000000    8.926348e+06    1.000000
75%    721.000000   44.000000    8.000000    1.258767e+07    2.000000
max    850.000000  140.000000   10.000000    2.117743e+07    4.000000

       salario anual
count   9.980000e+02
mean    3.505829e+07
std     5.289890e+08
min     0.000000e+00
25%     2.849010e+06
50%     8.637196e+06
75%     1.401381e+07
max     1.193469e+10
```



DADOS

In [178…

```
dados
```

Out[178…

	id	pontos	estado	genero	idade	bens	saldo na conta	produtos	possui cartao	ativo	
0	1	619	PI	Feminino	42	2	0	1	1	1	1
1	2	608	CE	Feminino	41	1	8380786	1	0	1	1
2	3	502	PI	Feminino	42	8	1596608	3	1	0	1
3	4	699	PI	Feminino	39	1	0	2	0	0	
4	5	850	CE	Feminino	43	2	12551082	1	1	1	
...	...	...	...	...	...	...	...	...	...	...	
994	996	838	CE	Masculino	43	9	12310588	2	1	0	1
995	997	610	CE	Masculino	29	9	0	3	0	1	
996	998	811	CE	Masculino	44	3	0	2	0	1	
997	999	587	CE	Masculino	62	7	12128627	1	0	1	
998	1000	811	MA	Feminino	28	4	16773882	2	1	1	

998 rows × 12 columns

8 - Mostre o tamanho dos dados importados, utilizando o shape.

In [179…
```
dados.shape
```

Out[179… (998, 12)

Tupla: Primeiro elemento a quantidade de linhas (0 A 999) e o segundo elemento as colunas (12

SALVAR: planilha na pasta de Origem do driver

In [ ]:
```
# Caminho onde você quer salvar o arquivo tratado
#caminho_saida = '/content/drive/MyDrive/Limpeza de dados/dados_tratados.

# Salvar os dados tratados
#dados.to_excel(caminho_saida, index=False)

#print(f"Arquivo salvo em: {caminho_saida}")
```

In [180…
```
# Caminhos onde você quer salvar os arquivos
caminho_excel = '/content/drive/MyDrive/Limpeza de dados/dados_tratados.x
caminho_csv = '/content/drive/MyDrive/Limpeza de dados/dados_tratados.csv

# Salvar como Excel
dados.to_excel(caminho_excel, index=False)

# Salvar como CSV
dados.to_csv(caminho_csv, index=False)

print(f"Arquivos salvos em:\n- Excel: {caminho_excel}\n- CSV: {caminho_cs
```

```
Arquivos salvos em:
- Excel: /content/drive/MyDrive/Limpeza de dados/dados_tratados.xlsx
- CSV: /content/drive/MyDrive/Limpeza de dados/dados_tratados.csv
```

PARTE 2 - ESTATISTICA:

- Apresente a média e a mediana do **saldo na conta** dos clientes abaixo de 40 anos;
- Apresente a média e a mediana do **saldo na conta** dos clientes acima de 40 anos;
- Apresente a média e a mediana do **saldo na conta** dos clientes que saíram e dos que permaneceram;
- Dos que saíram, mostre qual é o público predominante (Masculino ou Feminino), a idade, o na conta, patrimônio e os seus respectivos estados;

Lembre-se:

- Este desafio é uma oportunidade para você melhorar suas habilidades em tratamento de da análise exploratória.
- Utilize as ferramentas e técnicas aprendidas para transformar dados brutos em insights interessantes.
- Seja criativo e explore diferentes abordagens para alcançar os melhores resultados.
- Pesquise mais sobre as bibliotecas citadas para o melhor entendimento delas.
- Não se limite a dicas dadas, pois são somente orientações.

DADOS TRATADOS

1 - Apresente a média e a mediana do saldo na conta dos clientes abaixo de 40 anos;

In [181…

```
# Carregando os dados
caminho_arquivo = '/content/drive/MyDrive/Limpeza de dados/dados_tratados
dados = pd.read_csv(caminho_arquivo)

# Verificando as primeiras linhas para entender a estrutura
print(dados.head())
print("\nInformações sobre o dataframe:")
print(dados.info())
```

```
   id  pontos estado    genero  idade  bens  saldo na conta  produtos  \
0   1     619     PI  Feminino     42     2               0         1
1   2     608     CE  Feminino     41     1         8380786         1
2   3     502     PI  Feminino     42     8         1596608         3
3   4     699     PI  Feminino     39     1               0         2
4   5     850     CE  Feminino     43     2        12551082         1

   possui cartao  ativo  salario anual  saiu
0              1      1     10134888.0     1
1              0      1     11254258.0     0
2              1      0     11393157.0     1
3              0      0      9382663.0     0
4              1      1       790841.0     0

Informações sobre o dataframe:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 998 entries, 0 to 997
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              998 non-null    int64
 1   pontos          998 non-null    int64
 2   estado          998 non-null    object
 3   genero          998 non-null    object
 4   idade           998 non-null    int64
 5   bens            998 non-null    int64
 6   saldo na conta  998 non-null    int64
 7   produtos        998 non-null    int64
 8   possui cartao   998 non-null    int64
 9   ativo           998 non-null    int64
 10  salario anual   998 non-null    float64
 11  saiu            998 non-null    int64
dtypes: float64(1), int64(9), object(2)
memory usage: 93.7+ KB
None
```

In [204…

```python
# Filtrando clientes abaixo de 40 anos
abaixo_40 = dados[dados['idade'] < 40]

# Calculando média e mediana do saldo na conta
media_abaixo_40 = abaixo_40['saldo na conta'].mean() / 100
mediana_abaixo_40 = abaixo_40['saldo na conta'].median() / 100

print("\nClientes abaixo de 40 anos:")
print(f"Média do saldo na conta: R${media_abaixo_40:,.2f}")
print(f"Mediana do saldo na conta: R${mediana_abaixo_40:,.2f}")
```

```
Clientes abaixo de 40 anos:
Média do saldo na conta: R$70,154.28
Mediana do saldo na conta: R$82,293.82
```

2 - Apresente a média e a mediana do saldo na conta dos clientes acima de 40 anos;

In [205…

```python
# Filtrando clientes acima de 40 anos
acima_40 = dados[dados['idade'] >= 40]

# Calculando média e mediana do saldo na conta
media_acima_40 = acima_40['saldo na conta'].mean() / 100
mediana_acima_40 = acima_40['saldo na conta'].median() / 100
```

```
print("\nClientes acima de 40 anos:")
print(f"Média do saldo na conta: R${media_acima_40:,.2f}")
print(f"Mediana do saldo na conta: R${mediana_acima_40:,.2f}")
```

```
Clientes acima de 40 anos:
Média do saldo na conta: R$73,812.66
Mediana do saldo na conta: R$97,318.25
```

3 - Apresente a média e a mediana do saldo na conta dos clientes que saíram e dos que perma

In [206…

```
# Filtrando clientes que saíram (saiu = 1) e que permaneceram (saiu = 0)
clientes_sairam = dados[dados['saiu'] == 1]
clientes_permaneceram = dados[dados['saiu'] == 0]

# Calculando para os que saíram
if not clientes_sairam.empty:
    media_sairam = clientes_sairam['saldo na conta'].mean() / 100
    mediana_sairam = clientes_sairam['saldo na conta'].median() / 100
    print("\nClientes que saíram:")
    print(f"Média do saldo na conta: R${media_sairam:,.2f}")
    print(f"Mediana do saldo na conta: R${mediana_sairam:,.2f}")
else:
    print("\nNão há clientes que saíram na base de dados.")

# Calculando para os que permaneceram
media_permaneceram = clientes_permaneceram['saldo na conta'].mean() / 100
mediana_permaneceram = clientes_permaneceram['saldo na conta'].median() /
print("\nClientes que permaneceram:")
print(f"Média do saldo na conta: R${media_permaneceram:,.2f}")
print(f"Mediana do saldo na conta: R${mediana_permaneceram:,.2f}")
```

```
Clientes que saíram:
Média do saldo na conta: R$85,239.88
Mediana do saldo na conta: R$108,431.87

Clientes que permaneceram:
Média do saldo na conta: R$68,147.53
Mediana do saldo na conta: R$80,613.93
```

4 - Dos que saíram, mostre qual é o público predominante (Masculino ou Feminino), a idade, o s
conta, patrimônio e os seus respectivos estados;

In [199…

```
if not clientes_sairam.empty:
    # Contando gêneros dos que saíram
    contagem_genero = clientes_sairam['genero'].value_counts()

    # Verificando se há dados de gênero
    if not contagem_genero.empty:
        genero_predominante = contagem_genero.idxmax()
        print(f"\nGênero predominante entre os que saíram: {genero_predom

        # Criando um subconjunto apenas com o gênero predominante
        predominante = clientes_sairam[clientes_sairam['genero'] == gener

        # Calculando estatísticas
        idade_media = predominante['idade'].mean()
        saldo_medio = predominante['saldo na conta'].mean() / 100  # Conv
        patrimonio_medio = predominante['bens'].mean()
        estados = predominante['estado'].value_counts().head(3)  # Top 3
```

```
        print("\nCaracterísticas do público predominante que saiu:")
        print(f"• Idade média: {idade_media:.1f} anos")
        print(f"• Saldo médio na conta: R$ {saldo_medio:,.2f}")  # Format
        print(f"• Patrimônio médio (bens): {patrimonio_medio:.1f}")
        print("\nEstados mais frequentes:")
        for estado, count in estados.items():
            print(f"  - {estado}: {count} clientes")
    else:
        print("\nNão há informações de gênero para os clientes que saíram
else:
    print("\nNão há clientes que saíram na base de dados.")
```

```
Gênero predominante entre os que saíram: Feminino

Características do público predominante que saiu:
• Idade média: 43.7 anos
• Saldo médio na conta: R$ 82,019.14
• Patrimônio médio (bens): 4.7

Estados mais frequentes:
  - PI: 50 clientes
  - MA: 39 clientes
  - CE: 30 clientes
```

In [197…

```
if not clientes_sairam.empty:
    # Contando gêneros dos que saíram
    contagem_genero = clientes_sairam['genero'].value_counts()

    # Verificando se há dados de gênero
    if not contagem_genero.empty:
        genero_predominante = contagem_genero.idxmax()
        print(f"\nGênero predominante entre os que saíram: {genero_predom

        # Filtrando apenas os clientes com o gênero predominante
        predominante = clientes_sairam[clientes_sairam['genero'] == gener

        print("\nInformações dos clientes do gênero predominante que saír
        for idx, row in predominante.iterrows():
            print(f"\nCliente ID: {row['id'] if 'id' in row else idx}")
            print(f"Idade: {row['idade']} anos")
            print(f"Saldo na conta: R${row['saldo na conta'] / 100:,.2f}"
            print(f"Patrimônio (bens): {row['bens']}")
            print(f"Estado: {row['estado']}")
    else:
        print("\nNão há informações de gênero para os clientes que saíram
else:
    print("\nNão há clientes que saíram na base de dados.")
```

Gênero predominante entre os que saíram: Feminino

Informações dos clientes do gênero predominante que saíram:

Cliente ID: 1
Idade: 42 anos
Saldo na conta: R$0.00
Patrimônio (bens): 2
Estado: PI

Cliente ID: 3
Idade: 42 anos
Saldo na conta: R$15,966.08
Patrimônio (bens): 8
Estado: PI

Cliente ID: 8
Idade: 29 anos
Saldo na conta: R$115,046.74
Patrimônio (bens): 4
Estado: MA

Cliente ID: 23
Idade: 38 anos
Saldo na conta: R$0.00
Patrimônio (bens): 4
Estado: RP

Cliente ID: 31
Idade: 39 anos
Saldo na conta: R$0.00
Patrimônio (bens): 3
Estado: CE

Cliente ID: 36
Idade: 45 anos
Saldo na conta: R$134,264.04
Patrimônio (bens): 0
Estado: PI

Cliente ID: 42
Idade: 51 anos
Saldo na conta: R$122,522.32
Patrimônio (bens): 8
Estado: PI

Cliente ID: 44
Idade: 49 anos
Saldo na conta: R$131,394.56
Patrimônio (bens): 2
Estado: PI

Cliente ID: 47
Idade: 27 anos
Saldo na conta: R$112,045.67
Patrimônio (bens): 9
Estado: MA

Cliente ID: 48
Idade: 39 anos

Saldo na conta: R$13,784.38
Patrimônio (bens): 9
Estado: MA

Cliente ID: 59
Idade: 66 anos
Saldo na conta: R$0.00
Patrimônio (bens): 4
Estado: CE

Cliente ID: 90
Idade: 46 anos
Saldo na conta: R$0.00
Patrimônio (bens): 4
Estado: PI

Cliente ID: 92
Idade: 44 anos
Saldo na conta: R$0.00
Patrimônio (bens): 5
Estado: CE

Cliente ID: 106
Idade: 65 anos
Saldo na conta: R$0.00
Patrimônio (bens): 1
Estado: CE

Cliente ID: 107
Idade: 46 anos
Saldo na conta: R$107,073.27
Patrimônio (bens): 4
Estado: CE

Cliente ID: 120
Idade: 31 anos
Saldo na conta: R$107,818.63
Patrimônio (bens): 8
Estado: MA

Cliente ID: 128
Idade: 52 anos
Saldo na conta: R$0.00
Patrimônio (bens): 1
Estado: PI

Cliente ID: 140
Idade: 48 anos
Saldo na conta: R$21,314.62
Patrimônio (bens): 2
Estado: CE

Cliente ID: 141
Idade: 35 anos
Saldo na conta: R$129,490.36
Patrimônio (bens): 1
Estado: CE

Cliente ID: 146
Idade: 31 anos

Saldo na conta: R$40,915.55
Patrimônio (bens): 5
Estado: PI

Cliente ID: 153
Idade: 48 anos
Saldo na conta: R$0.00
Patrimônio (bens): 9
Estado: PI

Cliente ID: 168
Idade: 24 anos
Saldo na conta: R$113,034.22
Patrimônio (bens): 7
Estado: MA

Cliente ID: 170
Idade: 39 anos
Saldo na conta: R$0.00
Patrimônio (bens): 2
Estado: CE

Cliente ID: 181
Idade: 55 anos
Saldo na conta: R$161,608.81
Patrimônio (bens): 3
Estado: MA

Cliente ID: 186
Idade: 50 anos
Saldo na conta: R$0.00
Patrimônio (bens): 4
Estado: PI

Cliente ID: 192
Idade: 41 anos
Saldo na conta: R$56,214.85
Patrimônio (bens): 8
Estado: PI

Cliente ID: 205
Idade: 38 anos
Saldo na conta: R$129,022.06
Patrimônio (bens): 2
Estado: PI

Cliente ID: 208
Idade: 41 anos
Saldo na conta: R$89,763.84
Patrimônio (bens): 3
Estado: CE

Cliente ID: 229
Idade: 39 anos
Saldo na conta: R$74,596.15
Patrimônio (bens): 6
Estado: MA

Cliente ID: 239
Idade: 43 anos

Saldo na conta: R$11,622.05
Patrimônio (bens): 3
Estado: MA

Cliente ID: 240
Idade: 46 anos
Saldo na conta: R$0.00
Patrimônio (bens): 2
Estado: PI

Cliente ID: 247
Idade: 40 anos
Saldo na conta: R$123,497.58
Patrimônio (bens): 10
Estado: PI

Cliente ID: 270
Idade: 39 anos
Saldo na conta: R$0.00
Patrimônio (bens): 7
Estado: CE

Cliente ID: 295
Idade: 34 anos
Saldo na conta: R$112,822.26
Patrimônio (bens): 9
Estado: CE

Cliente ID: 299
Idade: 44 anos
Saldo na conta: R$0.00
Patrimônio (bens): 7
Estado: CE

Cliente ID: 329
Idade: 46 anos
Saldo na conta: R$95,441.27
Patrimônio (bens): 9
Estado: PI

Cliente ID: 335
Idade: 30 anos
Saldo na conta: R$112,013.81
Patrimônio (bens): 8
Estado: MA

Cliente ID: 340
Idade: 39 anos
Saldo na conta: R$165,272.13
Patrimônio (bens): 5
Estado: CE

Cliente ID: 341
Idade: 39 anos
Saldo na conta: R$115,301.31
Patrimônio (bens): 10
Estado: MA

Cliente ID: 342
Idade: 40 anos

Saldo na conta: R$129,502.49
Patrimônio (bens): 6
Estado: MA

Cliente ID: 363
Idade: 45 anos
Saldo na conta: R$150,842.93
Patrimônio (bens): 2
Estado: MA

Cliente ID: 381
Idade: 39 anos
Saldo na conta: R$0.00
Patrimônio (bens): 2
Estado: CE

Cliente ID: 401
Idade: 66 anos
Saldo na conta: R$135,842.41
Patrimônio (bens): 1
Estado: PI

Cliente ID: 408
Idade: 49 anos
Saldo na conta: R$141,434.04
Patrimônio (bens): 9
Estado: CE

Cliente ID: 415
Idade: 41 anos
Saldo na conta: R$181,461.48
Patrimônio (bens): 9
Estado: MA

Cliente ID: 418
Idade: 61 anos
Saldo na conta: R$110,368.03
Patrimônio (bens): 5
Estado: MA

Cliente ID: 422
Idade: 60 anos
Saldo na conta: R$115,924.89
Patrimônio (bens): 3
Estado: MA

Cliente ID: 435
Idade: 37 anos
Saldo na conta: R$114,754.08
Patrimônio (bens): 8
Estado: MA

Cliente ID: 450
Idade: 38 anos
Saldo na conta: R$6,806.58
Patrimônio (bens): 6
Estado: PI

Cliente ID: 465
Idade: 32 anos

Saldo na conta: R$133,950.37
Patrimônio (bens): 4
Estado: PI

Cliente ID: 469
Idade: 38 anos
Saldo na conta: R$130,878.75
Patrimônio (bens): 5
Estado: MA

Cliente ID: 495
Idade: 47 anos
Saldo na conta: R$0.00
Patrimônio (bens): 6
Estado: PI

Cliente ID: 496
Idade: 38 anos
Saldo na conta: R$170,061.92
Patrimônio (bens): 2
Estado: PI

Cliente ID: 501
Idade: 58 anos
Saldo na conta: R$0.00
Patrimônio (bens): 8
Estado: PI

Cliente ID: 516
Idade: 51 anos
Saldo na conta: R$136,188.78
Patrimônio (bens): 3
Estado: PI

Cliente ID: 518
Idade: 56 anos
Saldo na conta: R$0.00
Patrimônio (bens): 10
Estado: PI

Cliente ID: 522
Idade: 35 anos
Saldo na conta: R$211,774.31
Patrimônio (bens): 1
Estado: PI

Cliente ID: 530
Idade: 48 anos
Saldo na conta: R$152,827.99
Patrimônio (bens): 3
Estado: MA

Cliente ID: 540
Idade: 62 anos
Saldo na conta: R$114,931.35
Patrimônio (bens): 5
Estado: MA

Cliente ID: 541
Idade: 28 anos

Saldo na conta: R$111,071.36
Patrimônio (bens): 1
Estado: MA

Cliente ID: 546
Idade: 49 anos
Saldo na conta: R$0.00
Patrimônio (bens): 7
Estado: PI

Cliente ID: 557
Idade: 51 anos
Saldo na conta: R$154,962.99
Patrimônio (bens): 3
Estado: CE

Cliente ID: 565
Idade: 46 anos
Saldo na conta: R$0.00
Patrimônio (bens): 1
Estado: PI

Cliente ID: 568
Idade: 54 anos
Saldo na conta: R$0.00
Patrimônio (bens): 2
Estado: CE

Cliente ID: 575
Idade: 49 anos
Saldo na conta: R$0.00
Patrimônio (bens): 10
Estado: CE

Cliente ID: 583
Idade: 33 anos
Saldo na conta: R$108,431.87
Patrimônio (bens): 2
Estado: CE

Cliente ID: 586
Idade: 51 anos
Saldo na conta: R$119,741.77
Patrimônio (bens): 1
Estado: MA

Cliente ID: 587
Idade: 51 anos
Saldo na conta: R$100,946.71
Patrimônio (bens): 2
Estado: PI

Cliente ID: 591
Idade: 47 anos
Saldo na conta: R$157,296.02
Patrimônio (bens): 6
Estado: CE

Cliente ID: 592
Idade: 38 anos

Saldo na conta: R$144,606.22
Patrimônio (bens): 7
Estado: MA

Cliente ID: 600
Idade: 57 anos
Saldo na conta: R$162,448.69
Patrimônio (bens): 5
Estado: MA

Cliente ID: 602
Idade: 43 anos
Saldo na conta: R$0.00
Patrimônio (bens): 8
Estado: PI

Cliente ID: 617
Idade: 31 anos
Saldo na conta: R$0.00
Patrimônio (bens): 0
Estado: CE

Cliente ID: 619
Idade: 62 anos
Saldo na conta: R$140,745.33
Patrimônio (bens): 8
Estado: MA

Cliente ID: 632
Idade: 61 anos
Saldo na conta: R$0.00
Patrimônio (bens): 1
Estado: PI

Cliente ID: 633
Idade: 54 anos
Saldo na conta: R$115,988.86
Patrimônio (bens): 6
Estado: MA

Cliente ID: 645
Idade: 39 anos
Saldo na conta: R$135,134.99
Patrimônio (bens): 5
Estado: MA

Cliente ID: 646
Idade: 39 anos
Saldo na conta: R$133,102.92
Patrimônio (bens): 9
Estado: PI

Cliente ID: 647
Idade: 34 anos
Saldo na conta: R$42,157.08
Patrimônio (bens): 8
Estado: PI

Cliente ID: 689
Idade: 45 anos

Saldo na conta: R$126,674.81
Patrimônio (bens): 5
Estado: MA

Cliente ID: 690
Idade: 51 anos
Saldo na conta: R$136,294.97
Patrimônio (bens): 7
Estado: PI

Cliente ID: 699
Idade: 57 anos
Saldo na conta: R$106,138.33
Patrimônio (bens): 1
Estado: MA

Cliente ID: 716
Idade: 25 anos
Saldo na conta: R$8,660.55
Patrimônio (bens): 3
Estado: CE

Cliente ID: 722
Idade: 56 anos
Saldo na conta: R$209,767.31
Patrimônio (bens): 2
Estado: CE

Cliente ID: 723
Idade: 42 anos
Saldo na conta: R$129,634.25
Patrimônio (bens): 6
Estado: CE

Cliente ID: 731
Idade: 57 anos
Saldo na conta: R$0.00
Patrimônio (bens): 1
Estado: PI

Cliente ID: 736
Idade: 60 anos
Saldo na conta: R$128,981.07
Patrimônio (bens): 7
Estado: MA

Cliente ID: 745
Idade: 49 anos
Saldo na conta: R$88,915.37
Patrimônio (bens): 2
Estado: PI

Cliente ID: 763
Idade: 35 anos
Saldo na conta: R$124,151.09
Patrimônio (bens): 5
Estado: PI

Cliente ID: 764
Idade: 36 anos

Saldo na conta: R$7,725.35
Patrimônio (bens): 6
Estado: MA

Cliente ID: 771
Idade: 63 anos
Saldo na conta: R$114,715.71
Patrimônio (bens): 1
Estado: PI

Cliente ID: 772
Idade: 36 anos
Saldo na conta: R$129,748.54
Patrimônio (bens): 2
Estado: MA

Cliente ID: 778
Idade: 58 anos
Saldo na conta: R$116,922.25
Patrimônio (bens): 7
Estado: MA

Cliente ID: 807
Idade: 46 anos
Saldo na conta: R$0.00
Patrimônio (bens): 9
Estado: CE

Cliente ID: 819
Idade: 42 anos
Saldo na conta: R$123,331.36
Patrimônio (bens): 2
Estado: MA

Cliente ID: 825
Idade: 69 anos
Saldo na conta: R$137,453.43
Patrimônio (bens): 9
Estado: CE

Cliente ID: 840
Idade: 48 anos
Saldo na conta: R$0.00
Patrimônio (bens): 3
Estado: CE

Cliente ID: 845
Idade: 41 anos
Saldo na conta: R$0.00
Patrimônio (bens): 5
Estado: CE

Cliente ID: 853
Idade: 0 anos
Saldo na conta: R$97,133.92
Patrimônio (bens): 1
Estado: MA

Cliente ID: 858
Idade: 49 anos

Saldo na conta: R$134,956.02
Patrimônio (bens): 5
Estado: MA

Cliente ID: 863
Idade: 32 anos
Saldo na conta: R$172,448.77
Patrimônio (bens): 2
Estado: PI

Cliente ID: 869
Idade: -10 anos
Saldo na conta: R$170,833.46
Patrimônio (bens): 1
Estado: PI

Cliente ID: 873
Idade: 45 anos
Saldo na conta: R$129,818.39
Patrimônio (bens): 7
Estado: MA

Cliente ID: 882
Idade: 60 anos
Saldo na conta: R$0.00
Patrimônio (bens): 6
Estado: PI

Cliente ID: 883
Idade: 43 anos
Saldo na conta: R$0.00
Patrimônio (bens): 4
Estado: PI

Cliente ID: 885
Idade: 45 anos
Saldo na conta: R$45,144.43
Patrimônio (bens): 4
Estado: PI

Cliente ID: 895
Idade: 51 anos
Saldo na conta: R$0.00
Patrimônio (bens): 3
Estado: PI

Cliente ID: 896
Idade: 35 anos
Saldo na conta: R$125,884.95
Patrimônio (bens): 8
Estado: MA

Cliente ID: 900
Idade: 40 anos
Saldo na conta: R$102,967.41
Patrimônio (bens): 2
Estado: PI

Cliente ID: 907
Idade: 45 anos

Saldo na conta: R$0.00
Patrimônio (bens): 9
Estado: PI

Cliente ID: 909
Idade: 46 anos
Saldo na conta: R$104,947.72
Patrimônio (bens): 1
Estado: PI

Cliente ID: 944
Idade: 46 anos
Saldo na conta: R$0.00
Patrimônio (bens): 6
Estado: PI

Cliente ID: 948
Idade: 39 anos
Saldo na conta: R$0.00
Patrimônio (bens): 4
Estado: CE

Cliente ID: 949
Idade: 0 anos
Saldo na conta: R$118,590.41
Patrimônio (bens): 5
Estado: PI

Cliente ID: 951
Idade: 48 anos
Saldo na conta: R$73,309.38
Patrimônio (bens): 3
Estado: PI

Cliente ID: 956
Idade: 42 anos
Saldo na conta: R$156,371.61
Patrimônio (bens): 2
Estado: CE

Cliente ID: 966
Idade: 43 anos
Saldo na conta: R$115,888.04
Patrimônio (bens): 4
Estado: MA

Cliente ID: 978
Idade: 43 anos
Saldo na conta: R$132,558.26
Patrimônio (bens): 8
Estado: PI

Cliente ID: 985
Idade: 35 anos
Saldo na conta: R$128,100.28
Patrimônio (bens): 6
Estado: MA

Cliente ID: 991
Idade: 49 anos

Saldo na conta: R$168,197.66
Patrimônio (bens): 3
Estado: PI

ANDRE MOURA LIMA