```
# Estatísticas Descritivas e Tratamento dos Dados
```

In [1]:

```
#pip install pyodbc
# importar bibliotecas
import pandas as pd
import numpy as np
import io
```

2.1 ESTATISTICAS DESCRITIVAS ¶

In [2]:

```
#Importa arquivo final do tratamento de dados
IND_FINAL   = pd.read_excel(r'C:\Users\alesa\OneDrive - ANVISA - Agencia Nacional de
                            header=0, sheet_name="Planilha1")
assuntos = pd.read_excel(r'C:\Users\alesa\OneDrive - ANVISA - Agencia Nacional de Vi
                            header=0, sheet_name="ASSUNTOS")
```

In [4]:

```
# Selecionando variáveis de interesse
IND_FINAL_0 = IND_FINAL[[#'SAIDA_A',
                         #'SAIDA_M',
                         #'ENTRADA_A',
                         #'ENTRADA_M',
                         'CO_ASSUNTO',
                         #'NU_PROCESSO',
                         'NU_EXPEDIENTE',
                         'RECURSO',
                         'CANCELADO',
                         'DT_FINALIZACAO',
                         'DT_ENTRADA',
                         #'DT_INICIO_SITUACAO',
                         #'DT_FIM_SITUACAO',
                         'CICLO_GGTOX',
                         'DIAS',
                         #'NU_CNPJ_EMPRESA',
                         #'NO_RAZAO_SOCIAL_EMPRESA',
                         #'NO_PRODUTO',
                         #'TIPO_PUBLICACAO',
                         #'DT_FIM_SITUACAO_LAG'
                         ]]

IND_FINAL_0 = pd.DataFrame(IND_FINAL_0[(IND_FINAL_0['CICLO_GGTOX'] == "3.1.Analise") |
                                       (IND_FINAL_0['CICLO_GGTOX'] == '6.Tempo_total')])

IND_FINAL_0 = pd.DataFrame(IND_FINAL_0[(IND_FINAL_0['CICLO_GGTOX'] == "6.Tempo_total


IND_FINAL_0 = pd.DataFrame(IND_FINAL_0[(IND_FINAL_0['CO_ASSUNTO'] == 5065) |
                                       (IND_FINAL_0['CO_ASSUNTO'] == 5041)])

# Transformando Recurso e Cancelado maior que 1 em 1
IND_FINAL_0['RECURSO'] = np.where((IND_FINAL_0['RECURSO'] >= 1), 1, 0)
IND_FINAL_0['CANCELADO'] = np.where((IND_FINAL_0['CANCELADO'] >= 1), 1, 0)

IND_FINAL_0.info()
```

...

In [5]:

```
# Transpor os dados e gerando as duas bases de 5041 e 5065

df_transposed = IND_FINAL_0.pivot_table(index=['CO_ASSUNTO', 'NU_EXPEDIENTE', 'RECUR
                                       'DT_ENTRADA','DT_FINALIZACAO'],
                                columns='CICLO_GGTOX',
                                values='DIAS').reset_index()

# Renomear as colunas resultantes
df_transposed.columns.name = None

# Renomear as colunas adicionando um prefixo
df_transposed = df_transposed.rename(columns=lambda x: '' + str(x))

df_transposed.rename(columns={'3.1.Analise': 'ANALISE'}, inplace=True)
df_transposed.rename(columns={'6.Tempo_total': 'TOTAL'}, inplace=True)

assunto_tipo = assuntos[['CO_ASSUNTO','TIPO_PUBLICACAO']]

df_transposed = df_transposed.merge(assunto_tipo, on="CO_ASSUNTO", how="left")
df_transposed = pd.DataFrame(df_transposed[(df_transposed['TIPO_PUBLICACAO'] == "1.
df_transposed = df_transposed.drop(['TIPO_PUBLICACAO'], axis=1)
#df_transposed.head()
#df_transposed.info()

df_41 = pd.DataFrame(df_transposed[(df_transposed['CO_ASSUNTO'] == 5041)])
df_65 = pd.DataFrame(df_transposed[(df_transposed['CO_ASSUNTO'] == 5065)])

df_41.head()
```

...

In [6]:

```
#Verificando nulos
df_transposed.isnull().sum()
```

...

In [7]:

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

In [8]:

```
#Boxplot por categoria
sns.boxplot(x="CO_ASSUNTO",
            y="TOTAL",
            data=df_transposed)
plt.show()
```

...

In [9]:

```
sns.distplot(df_41['TOTAL'])
plt.title('Gráfico de Distribuição - 5041')
plt.show()
```

...

In [10]:

```
sns.distplot(df_65['TOTAL'])
plt.title('Gráfico de Distribuição - 5065')
plt.show()
```

...

In [12]:

```
# Gráfico de Dispersão

# Crie o gráfico de dispersão
plt.scatter(df_transposed.index, df_transposed['TOTAL'])

# Adicione rótulos aos eixos x e y
plt.xlabel('EXPEDIENTES')
plt.ylabel('DIAS')

# Adicione um título ao gráfico
plt.title('Gráfico de Dispersão - 5041 e 5065')

# Exiba o gráfico
plt.show()
```

...

In [19]:

```
# Gráfico de Dispersão

# Crie o gráfico de dispersão
plt.scatter(df_65.index, df_65['TOTAL'])

# Adicione rótulos aos eixos x e y
plt.xlabel('EXPEDIENTES')
plt.ylabel('DIAS')

# Adicione um título ao gráfico
plt.title('Gráfico de Dispersão - 5065 ')

# Exiba o gráfico
plt.show()

```

...

In [20]:

```
descricao_coluna = df_41['TOTAL'].describe()
print(descricao_coluna)
```

...

In [21]:

```
descricao_coluna = df_65['TOTAL'].describe()
print(descricao_coluna)
```

...

Detecção de anomalias - Outliers

In [22]:

```
#outliers semana 12 - detecção de anomalias
#importar bibliotecas
from sklearn.ensemble import IsolationForest
```

In [23]:

```
#df_transposed = pd.DataFrame(df_transposed[(df_transposed['CO_ASSUNTO'] == 5041)])

#Definir modelo de Isolation Forest - SOMENTE TOTAL
modelo=IsolationForest(n_estimators=100,max_samples='auto',random_state=0)
#Visualizar parâmetros do modelo
print(modelo.get_params())

variaveis_anomalia = ['TOTAL']

#Ajustar modelo multivariado
modelo.fit(df_transposed[variaveis_anomalia])

#Criar coluna chamada score
df_transposed['scores'] = modelo.decision_function(df_transposed[variaveis_anomalia]

df_transposed.info()
```

...

In [24]:

```
#Definir modelo de Isolation Forest - CO_ASSUNTO e TOTAL
modelo=IsolationForest(n_estimators=100,max_samples='auto',random_state=0)
#Visualizar parâmetros do modelo
print(modelo.get_params())

colunas_analise = ['TOTAL']

#Ajustar modelo multivariado
modelo.fit(df_transposed[colunas_analise])

#Criar coluna chamada score
df_transposed['scores'] = modelo.decision_function(df_transposed[colunas_analise])

df_transposed.head()


```

...

In [28]:

```
df_transposed.to_excel('anomalias.xlsx', sheet_name='Planilha1', index=False)
```

In [ ]:

```
# Exportando arquivo final para a modelagem
#TESTE.to_excel('TESTE.xlsx', sheet_name='Planilha1', index=False)
df.to_excel('df.xlsx', sheet_name='Planilha1', index=False)
```