Modelo Previsão Tempo Total

In [1]:

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

In [2]:

```
# Tratamento da base: agrupando a serie por ano e mês
df_0   = pd.read_excel(r'C:\Users\alesa\OneDrive - ANVISA - Agencia Nacional de Vigi
                       header=0, sheet_name="Planilha1")

df_0['DATA_FINALIZACAO'] = df_0['DT_FINALIZACAO'].dt.date
df_0['DATA_FINALIZACAO_AM'] = df_0['DT_FINALIZACAO'].dt.strftime('%Y-%m')

df_41 = pd.DataFrame(df_0[(df_0['CO_ASSUNTO'] == 5041)])
df_65 = pd.DataFrame(df_0[(df_0['CO_ASSUNTO'] == 5065)])

#df = df_0.groupby(['CO_ASSUNTO','CO_ASSUNTO_T','DATA_FINALIZACAO'])['TOTAL'].mean()
df =    df_0.groupby(['CO_ASSUNTO','CO_ASSUNTO_T','DATA_FINALIZACAO_AM'])['TOTAL'].m
df41 = df_41.groupby(['CO_ASSUNTO','CO_ASSUNTO_T','DATA_FINALIZACAO_AM'])['TOTAL'].m
df65 = df_65.groupby(['CO_ASSUNTO','CO_ASSUNTO_T','DATA_FINALIZACAO_AM'])['TOTAL'].m

#df65.head()

```

Séries Temporais

In [3]:

```
# Verificando a estacionaridade da série

from statsmodels.tsa.stattools import adfuller
import matplotlib.pyplot as plt

```

In [4]:

```
#result = adfuller(df['TOTAL'])
result41 = adfuller(df41['TOTAL'])
result65 = adfuller(df65['TOTAL'])

# Extraia os resultados do teste
#p_value = result[1]
p_value41 = result41[1]
p_value65 = result65[1]


# Imprima os resultados
#print(f"Valor p: {p_value}")
print(f"Valor p41: {p_value41}")
print(f"Valor p65: {p_value65}")

```

```
Valor p41: 0.011022706431745681
Valor p65: 0.0002491862947946675
```

In [5]:

```
# Verificar a estacionariedade com valor-p
#if p_value < 0.05:
#    print("A série é estacionária.")
#else:
#    print("A série não é estacionária.")

if p_value41 < 0.05:
    print("A série41 é estacionária.")
else:
    print("A série41 não é estacionária.")

if p_value65 < 0.05:
    print("A série65 é estacionária.")
else:
    print("A série65 não é estacionária.")
```

```
A série41 é estacionária.
A série65 é estacionária.
```

In [6]:

```
#### Gráficos ACF e PACF
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
```

In [7]:

```
# Definir a coluna ano como índice
df41.set_index('DATA_FINALIZACAO_AM',inplace=True)
# Alterar o nome do índice de Any para ano
# Alterar o nome do índice de Any para ano
df41.index.name = 'AnoMes'
df41 = df41[['TOTAL']]
```

In [8]:

```
# Definir a coluna ano como índice
df65.set_index('DATA_FINALIZACAO_AM',inplace=True)
# Alterar o nome do índice de Any para ano
# Alterar o nome do índice de Any para ano
df65.index.name = 'AnoMes'
df65 = df65[['TOTAL']]
#df65.head()
```

In [11]:

```
# Verificando Autocorrelação
fig, ax = plt.subplots(figsize=(8,5))
plot_acf(df65['TOTAL'], lags=30, ax=ax)
plt.title('AUTOCORRELAÇÃO 5065')
plt.show()

```

...

In [12]:

```
# Verificando Autocorrelação Parcial
fig, ax = plt.subplots(figsize=(8,5))
plot_pacf(df65['TOTAL'], lags=30, ax=ax)
plt.title('AUTOCORRELAÇÃO PARCIAL 5065')
plt.show()
```

...

Modelo Autoregressivo (AR)

In [21]:

```
# AUTOREGRESSIVO
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
```

In [24]:

```
ar_modelo = AutoReg(df65[['TOTAL']], lags=2).fit()
ar_modelo.summary()
```

...

In [25]:

```
# Previsão do Modelo
pred = ar_modelo.predict(start=len(df65), end=len(df65)+2, dynamic=False)
pred
```

...

In [26]:

```
# Gráfico de previsão

# Ajustar o modelo ARIMA aos dados
#model = ARIMA(df41['TOTAL'], order=order)
#model_fit = model.fit()

ar_modelo = AutoReg(df41[['TOTAL']], lags=2).fit()
#modelo_ma = sm.tsa.arima.ARIMA(df65[['TOTAL']], order=(0,0,1)) # Ordem do modelo MA

# Fazer previsões
n_steps = 3  # Número de passos futuros a serem previstos
forecast = ar_modelo.forecast(steps=n_steps)
#forecast1 = modelo_ma.forecast(steps=n_steps)

# Plotar os dados originais e as previsões
plt.plot(df41['TOTAL'], label='Dados Originais')
#plt.plot(np.arange(len(df65['TOTAL']), len(df65['TOTAL']) + n_steps), forecast, for
plt.plot(np.arange(len(df41['TOTAL']), len(df41['TOTAL']) + n_steps), forecast, labe
plt.title('PREVISÕES 5041')
plt.legend()
plt.show()
```

...

Modelo de Médias Moveis (MA)

In [18]:

```
import statsmodels.api as sm
```

In [19]:

```
#MODELO MA
modelo_ma = sm.tsa.arima.ARIMA(df65[['TOTAL']], order=(0,0,1)) # Ordem do modelo MA
resultado_ma = modelo_ma.fit()
print(resultado_ma.summary())
```

...

Modelo de Média Movel Autoregressivo (ARMA)

In [90]:

```
#MODELO ARMA
modelo_ma = sm.tsa.arima.ARIMA(df65[['TOTAL']], order=(0,1,1)) # Ordem do modelo MA

resultado_ma = modelo_ma.fit()
print(resultado_ma.summary())
```

...