Seção 1: Preparação de Dados

Importação e Limpeza de Dados:

In [1]:
```
# Importando bibliotecas necessárias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.statespace.sarimax import SARIMAX
from pmdarima import auto_arima
from sklearn.metrics import mean_absolute_error, mean_squared_error
from scipy import stats
from statsmodels.tsa.arima.model import ARIMA
import plotly.express as px
import plotly.graph_objects as go
import statsmodels.api as sm

# Carregando o conjunto de dados financeiro
df = pd.read_csv('/Users/tsumano/Downloads/sistema de pedidos 05-09-2024.csv', encoding='latin1', sep=';')

# Visualizando as primeiras linhas do dataset
df.head()
```

Out[1]:

	Numero do pedido	Pedido SC	Agendamento	PDI	Cliente	Segmento	Data do recebimento	Data de finalização (desejada)	Produto (nu cat)	Quantidade no pedido	...	Custo de deslocamento	IPI	Prazo de pagamento	CNPJ para faturamento	Local de entrega	Nome	C
0	16605	MANUTENÇÃO	NaN	NaN	Experience Locadora	Locadoras	05/09/2024	12/09/2024	SUNTECH	1	...	NaN	30 DIAS	NaN	009.206.045-52	mesmo do cliente	Adson Souza da Silva	009.206.(
1	16604	REMOÇÃO	NaN	NaN	Localiza	Locadoras	05/09/2024	09/09/2024	SHE1F49	1	...	NaN	30 DIAS	NaN	NaN	mesmo do cliente	NaN	
2	16603	REMOÇÃO	NaN	NaN	Localiza	Locadoras	05/09/2024	09/09/2024	RMK4I90	1	...	NaN	30 DIAS	NaN	NaN	mesmo do cliente	NaN	
3	16602	REMOÇÃO	NaN	NaN	Localiza	Locadoras	05/09/2024	09/09/2024	RVI0F57	1	...	NaN	30 DIAS	NaN	NaN	mesmo do cliente	NaN	
4	16601	REMOÇÃO	NaN	NaN	Localiza	Locadoras	05/09/2024	09/09/2024	RVA2D50	1	...	NaN	30 DIAS	NaN	NaN	mesmo do cliente	NaN	

5 rows × 29 columns

Valores Ausentes

In [2]:
```
# Verificando informações gerais do dataset, como tipos de dados e valores ausentes
df.info()

# Identificando valores ausentes
df.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27404 entries, 0 to 27403
Data columns (total 29 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Numero do pedido                 27404 non-null  int64  
 1   Pedido SC                        13689 non-null  object 
 2   Agendamento                      546 non-null    object 
 3   PDI                              1098 non-null   object 
 4   Cliente                          27404 non-null  object 
 5   Segmento                         24933 non-null  object 
 6   Data do recebimento              27404 non-null  object 
 7   Data de finalização (desejada)   27399 non-null  object 
 8   Produto (nu cat)                 27404 non-null  object 
 9   Quantidade no pedido             27404 non-null  int64  
 10  Separados                        27404 non-null  object 
 11  Entrega                          27404 non-null  object 
 12  Previsão De Instalação           19925 non-null  object 
 13  Instalados                       27404 non-null  object 
 14  Local de instalações             27266 non-null  object 
 15  Status                           27404 non-null  object 
 16  Frete                            22049 non-null  object 
 17  Custo de cofre                   15419 non-null  object 
 18  Custo de instalação              17149 non-null  object 
 19  Custo de deslocamento            1495 non-null   object 
 20  IPI                              17240 non-null  object 
 21  Prazo de pagamento               0 non-null      float64
 22  CNPJ    para faturamento          10986 non-null  object 
 23  Local de entrega                 17107 non-null  object 
 24  Nome                             7670 non-null   object 
 25  CNPJ                             6622 non-null   object 
 26  Endereço                         7677 non-null   object 
 27  Tel                              1912 non-null   object 
 28  Observações                      13984 non-null  object 
dtypes: float64(1), int64(2), object(26)
memory usage: 6.1+ MB
```

Out[2]:
```
Numero do pedido                    0
Pedido SC                       13715
Agendamento                     26858
PDI                             26306
Cliente                             0
Segmento                         2471
Data do recebimento                 0
Data de finalização (desejada)      5
Produto (nu cat)                    0
Quantidade no pedido                0
Separados                           0
Entrega                             0
Previsão De Instalação           7479
Instalados                          0
Local de instalações              138
Status                              0
Frete                            5355
Custo de cofre                  11985
Custo de instalação             10255
Custo de deslocamento           25909
IPI                             10164
Prazo de pagamento              27404
CNPJ   para faturamento          16418
Local de entrega                10297
Nome                            19734
CNPJ                            20782
Endereço                        19727
Tel                             25492
Observações                     13420
dtype: int64
```

In [3]:
```python
# Removendo colunas irrelevantes com muitos valores ausentes
columns_to_remove = [
    'Pedido SC', 'Agendamento', 'PDI', 'Custo de deslocamento', 'Prazo de pagamento',
    'Nome', 'CNPJ', 'Endereço', 'Tel', 'Observações'
]
df = df.drop(columns=columns_to_remove)
```

In [4]:
```python
# Removendo caracteres não numéricos e convertendo para tipo numérico
df['Custo de cofre'] = df['Custo de cofre'].replace('[\D]', '', regex=True)  # Removendo caracteres não numéricos
df['Custo de cofre'] = pd.to_numeric(df['Custo de cofre'], errors='coerce')  # Convertendo para tipo numérico
# Preenchendo valores ausentes com a mediana
df['Custo de cofre'] = df['Custo de cofre'].fillna(df['Custo de cofre'].median())
```

In [5]:
```python
print(df['Custo de instalação'].unique())
```

```
['30' '150' '50' nan '0' '40' '20' '120' '70' '75' '350' '100' '180' '29'
 '60' '85' '19,54' '90' '170' '250' '25' '280' '200,4' '80' '300' '190'
 'SANGUESSUGA 178,' '150.00' 'KIT R r$39,00' '0.00' '450' '300.00' '200'
 '145' '380' '480' '30 DIAS' '295,6' '35' '230' '140' '220' '19,65'
 'KIT R R$39,00' '540' '550' '17' '43,4' '570' '350,OO' '580' '490'
 '250,00 DIARIA' '260' '317,70 +ST - BR2' '43.40' '43' '150,00 + 50,00 -'
 '320' '150,00 + 50,00' '50,00 SS' '150,00+ 50,00 SS' '150,00 + 50,00 S'
 '335,64' '43.,40' '240' '92,4' '17.00' '160' '66' '92' '91' '77,5'
 '92.40' '315,2' '87.50' '87,5' '37,5' '110' '55' '130' '92,40 + 09,90 SA'
 '84,6' '-' '290' '20 ,00' 'R$ 30.00' 'R$ 1.30 DE DESCO' '49,2' '144' '77'
 '175' '610' '311' '84' '225' '66.00' '340' '72,5' 'R$ 70,00' 'R$ 77,50'
 '58,22' '36' '209' '80,00 CADA' '40,00+ 260,00 DE' '1.00'
 '40,00 + 200,00 d' '380,00 APROVADO' 'R$ 30,00' '70,00+ DES 188,' '330'
 '400' '250,00 aprovado' '250,00APROVADO' '450,00 aprovado'
 '50 DESLOCAMENTO' '350,00aprovado' '310' '350,00 aprovado'
 '92,40 + 29,90 ub' '400,00 aprovado' '400,00APROVADA' '500,00+ 56,40'
 '310,00 APROVADO' '150,00+70,00' '92,40 + 220,00' '400 APROVADO'
 '550,00 aguardand' '92,40 + 250,00 D' '500' '350,00 APROVADO'
 '400 aprovado' '-350 APROVADO' '800,00 TOTAL' '-150' '500 APROVADA'
 '-300' '-300 APROVADA' '-450' '150,00 PAGO' '647,00 aguardand'
 '-450 APROVADO' '700,00 AGUARDAND' '92,40 +200' '300,00aprovada'
 '350 APROVADA' '84.60' '350 aprovado' '105' '350,00 aprovada'
 '450,00 APROVADA' '150 dias' '300,00 aprovada' '77,55' '300,00 aprovado'
 '42,5' '280,00 DESLOCAME' '350,00 remoção' '400 remoção = de'
 '400,00 remoção +' '77,50 + 274,40 D' '50,00+170,00 DES'
 '250,00 DESLOCAME' '200 DESLOCAMENTO' '72' '150 deslocamento'
 '77,50 + 126,84' '250 DESLOCAMENTO' '150 DESLOCAMENTO' '211,96 + 77,50 d'
 '173' '50,00 + 250,00 D' '100,00 deslocame' '200 deslocamento'
 '230,00 deslocame' '285' '150,00 deslocame' '100 deslocamento' '125'
 '125 DESLOCAMENTO' '450 deslocamento' '210' '250,00 deslocame'
 '77,50+ 487,20 DE' '100,00 DESLOCAME' '87,50.' '147,1' '500 deslocamento'
 '50,00 deslocamen' '40,00+ 250 deslo' '40,00+ 140,00 de'
 '40,00+ 280,00 DE' '40,00 + 665,60 d' '250 deslocamento'
 '100 DESLOCAMENTO' '4.600' '300 DESLOCAMENTO' '600,00 remoção'
 '125,00 DESLOCAME' '50,00 DESLOCAMEN' '1.000,00' '80,00 + 431,20'
 '350 deslocamento' '134' 'DESLOCAMENTO$450' '123,2' '460,32 NOW SHOW'
 '431,20 DESLOCAME' '850,00 REMOÇÃO' '300,00 deslocame' '150,00+ 250,00 d'
 '750 de deslocame' '220 deslocamento' '92,5' '92,50 deslocamen'
 '78,40 DEVOLUÇÃO' '92,7' '37' '87']
```

In [6]:

```python
# Limpando a coluna 'Custo de instalação'

# Função para limpar e converter valores
def limpar_custo(valor):
    if pd.isna(valor):
        return np.nan
    valor = str(valor).replace(',', '.').replace('R$', '').replace(' ', '').replace('+', '').replace('APROVADO', '').replace('DESLOCAMENTO', '').replace('DESLOCAME', '').repla
    try:
        return float(valor)
    except ValueError:
        return np.nan

# Aplicando a função de limpeza
df['Custo de instalação'] = df['Custo de instalação'].apply(limpar_custo)

# Preenchendo valores ausentes com a mediana
df['Custo de instalação'] = df['Custo de instalação'].fillna(df['Custo de instalação'].median())

# Verificando a limpeza e conversão
print("Amostra dos dados da coluna 'Custo de instalação':")
```

```
print(df['Custo de instalação'].head(10))

# Obtendo estatísticas descritivas da coluna 'Custo de instalação'
print("\nEstatísticas descritivas da coluna 'Custo de instalação':")
print(df['Custo de instalação'].describe())

# Verificando valores ausentes na coluna 'Custo de instalação'
print("\nNúmero de valores ausentes na coluna 'Custo de instalação':")
print(df['Custo de instalação'].isnull().sum())
```

```
Amostra dos dados da coluna 'Custo de instalação':
0     30.0
1    150.0
2    150.0
3    150.0
4    150.0
5    150.0
6    150.0
7    150.0
8     50.0
9     66.0
Name: Custo de instalação, dtype: float64

Estatísticas descritivas da coluna 'Custo de instalação':
count    27404.000000
mean        69.527526
std         56.316986
min       -450.000000
25%         40.000000
50%         66.000000
75%         87.500000
max        800.000000
Name: Custo de instalação, dtype: float64

Número de valores ausentes na coluna 'Custo de instalação':
0
```

In [7]:
```
# 1. Removendo prefixos e valores inválidos
df['Frete'] = df['Frete'].replace({'Nao': np.nan, 'A ver': np.nan}, regex=True)  # Remove valores inválidos
df['Frete'] = df['Frete'].replace(r'^R\$ ', '', regex=True)  # Remove prefixos como 'R$ '
df['Frete'] = df['Frete'].replace(r'[^\d,]', '', regex=True)  # Remove caracteres não numéricos, exceto vírgulas

# 2. Substituindo vírgulas por pontos e convertendo para float
df['Frete'] = df['Frete'].str.replace(',', '.', regex=False)  # Substitui vírgulas por pontos
df['Frete'] = pd.to_numeric(df['Frete'], errors='coerce')  # Converte para float, forçando erros a NaN

# 3. Preenchendo valores ausentes com a mediana
df['Frete'] = df['Frete'].fillna(df['Frete'].median())
```

In [8]:
```
# Substituindo texto e caracteres não numéricos por NaN
df['IPI'] = df['IPI'].replace({
    r'[^\d,]': np.nan,  # Substitui caracteres não numéricos por NaN
    'nan': np.nan,      # Substitui valores 'nan' como string por NaN
    '-': np.nan         # Substitui '-' por NaN
}, regex=True)

# 2. Substituindo vírgulas por pontos e convertendo para float
df['IPI'] = df['IPI'].str.replace(',', '.', regex=False)  # Substitui vírgulas por pontos
df['IPI'] = pd.to_numeric(df['IPI'], errors='coerce')      # Converte para float, forçando erros a NaN
```

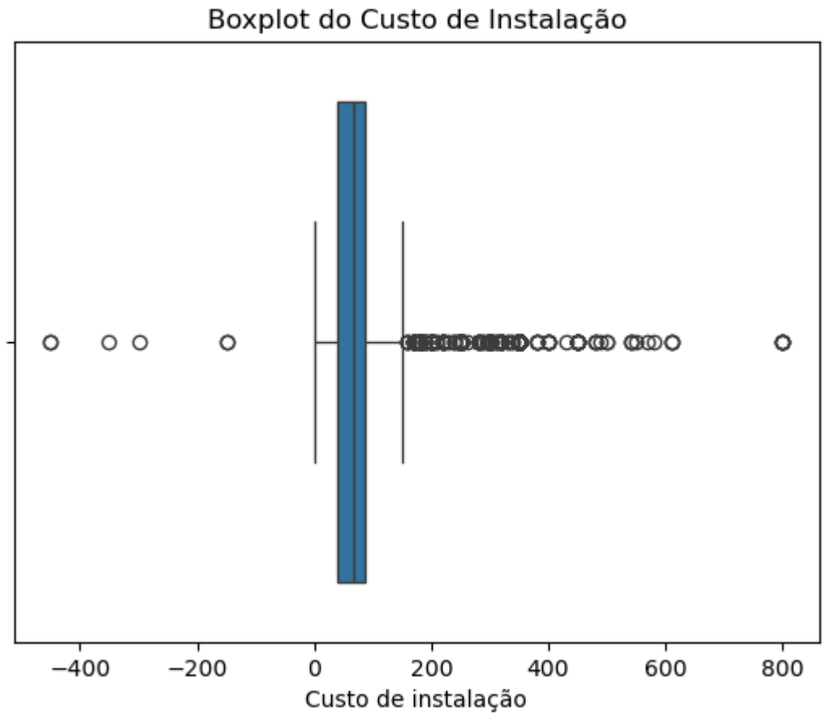
```
# 3. Preenchendo valores ausentes com a mediana
df['IPI'] = df['IPI'].fillna(df['IPI'].median())
```

In [9]:
```
# Preenchendo valores ausentes na coluna 'Segmento' com a moda
moda_segmento = df['Segmento'].mode()[0]
df['Segmento'] = df['Segmento'].fillna(moda_segmento)

# Removendo linhas com valores ausentes na coluna 'Data de finalização (desejada)'
df = df.dropna(subset=['Data de finalização (desejada)'])

# Preenchendo valores ausentes na coluna 'Previsão De Instalação' com 'Desconhecido'
df['Previsão De Instalação'] = df['Previsão De Instalação'].fillna('Desconhecido')

# Preenchendo valores ausentes na coluna 'Local de instalações' com 'Desconhecido'
df['Local de instalações'] = df['Local de instalações'].fillna('Desconhecido')

# Preenchendo valores ausentes na coluna 'Custo de cofre' com a mediana
df['Custo de cofre'] = df['Custo de cofre'].fillna(df['Custo de cofre'].median())

# Preenchendo valores ausentes na coluna 'CNPJ    para faturamento' com 'Desconhecido'
df['CNPJ    para faturamento'] = df['CNPJ    para faturamento'].fillna('Desconhecido')

# Preenchendo valores ausentes na coluna 'Local de entrega' com 'Desconhecido'
df['Local de entrega'] = df['Local de entrega'].fillna('Desconhecido')
```

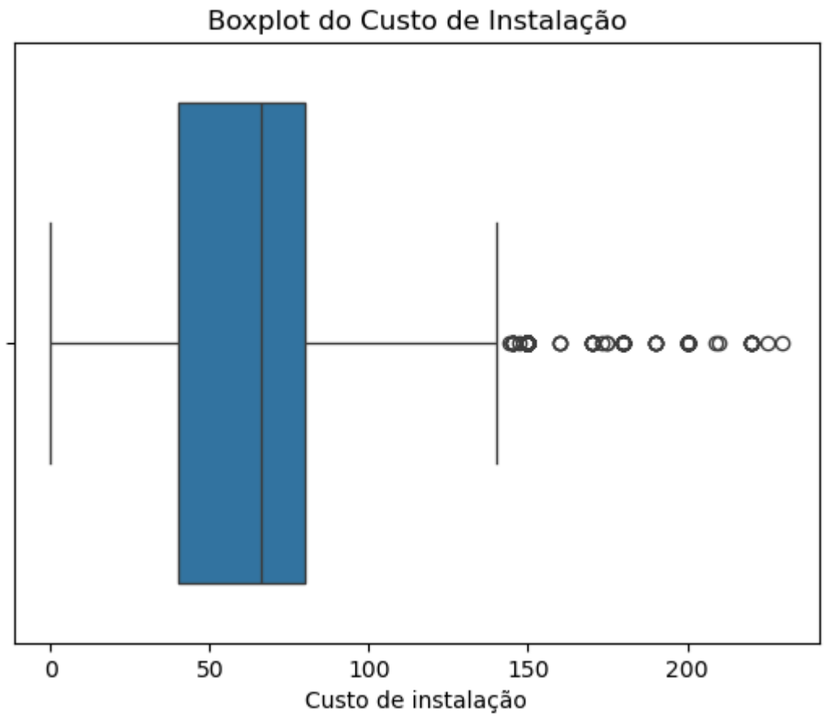
In [10]:
```
# Verificando se as colunas possuem multi-índice
if isinstance(df.columns, pd.MultiIndex):
    # Flattening multi-index columns
    df.columns = [' '.join(col).strip() for col in df.columns]

# Calculando a média e o desvio padrão do 'Custo de instalação'
mean = df['Custo de instalação'].mean()
std_dev = df['Custo de instalação'].std()

# Identificando os outliers
outliers = (df['Custo de instalação'] > mean + 3 * std_dev) | (df['Custo de instalação'] < mean - 3 * std_dev)

# Calculando a mediana do 'Custo de instalação'
mediana = df['Custo de instalação'].median()
```

Outliers

In [11]:
```
# Criando um boxplot para visualizar os outliers
sns.boxplot(x=df['Custo de instalação'])
plt.title('Boxplot do Custo de Instalação')
plt.show()
```



In [12]:
```
# Substituindo os outliers pela mediana
df.loc[outliers, 'Custo de instalação'] = mediana

# Criando um boxplot para visualizar se outliers foram corretamente tratados
sns.boxplot(x=df['Custo de instalação'])
plt.title('Boxplot do Custo de Instalação')
plt.show()
```



In [13]:
```
# Listando todas as colunas numéricas
numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns
print(numeric_columns)
```

```
Index(['Numero do pedido', 'Quantidade no pedido', 'Frete', 'Custo de cofre',
       'Custo de instalação', 'IPI'],
      dtype='object')
```

In [14]:
```
# Identificando outliers com base no IQR para cada coluna numérica
outliers_dict = {}

for col in numeric_columns:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Encontrando os outliers
    outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
    outliers_dict[col] = outliers

# Verificando as colunas que possuem outliers
for col, outliers in outliers_dict.items():
    if not outliers.empty:
        print(f"Outliers encontrados na coluna: {col}, Número de outliers: {len(outliers)}")
```

```
Outliers encontrados na coluna: Quantidade no pedido, Número de outliers: 3884
Outliers encontrados na coluna: Frete, Número de outliers: 1230
Outliers encontrados na coluna: Custo de cofre, Número de outliers: 9790
Outliers encontrados na coluna: Custo de instalação, Número de outliers: 2465
Outliers encontrados na coluna: IPI, Número de outliers: 152
```

Duplicadas

In [15]:
```python
# Verificando duplicadas no DataFrame inteiro
duplicatas = df[df.duplicated()]

# Exibindo o número de duplicadas encontradas
print(f"Número de observações duplicadas: {len(duplicatas)}")

# Exibindo as duplicadas, caso existam
if not duplicatas.empty:
    print(duplicatas)
else:
    print("Nenhuma observação duplicada encontrada.")
```

Número de observações duplicadas: 20

	Numero do pedido	Cliente	Segmento
4228	13361	Yalla Transportes e Serviços	Locadoras
17204	3993	Localiza	Locadoras
23366	1600	Localiza	Locadoras
24835	1080	Localiza	Locadoras
24836	1080	Localiza	Locadoras
24840	1079	Localiza	Locadoras
24842	1078	Localiza	Locadoras
24935	1067	Localiza	Locadoras
24936	1067	Localiza	Locadoras
25199	937	Localiza	Locadoras
25342	882	Localiza	Locadoras
26137	565	Localiza	Locadoras
26191	550	Localiza	Locadoras
26224	529	Localiza	Locadoras
26480	432	Localiza	Locadoras
26620	357	Unidas	Locadoras
27072	144	Localiza	Locadoras
27074	144	Localiza	Locadoras
27227	96	Localiza	Locadoras
27231	96	Localiza	Locadoras

	Data do recebimento	Data de finalização (desejada)	Produto (nu cat)
4228	27/03/2024	01/04/2024	SUNTECH+BLOQ
17204	03/10/2022	05/10/2022	BR172
23366	16/05/2022	17/05/2022	BR214
24835	01/04/2022	01/04/2022	BR221
24836	01/04/2022	01/04/2022	BR221
24840	01/04/2022	01/04/2022	BR222
24842	01/04/2022	01/04/2022	BR152
24935	31/03/2022	04/04/2022	BR218
24936	31/03/2022	04/04/2022	BR218
25199	23/03/2022	23/03/2022	BR153
25342	17/03/2022	22/03/2022	BR177
26137	17/02/2022	17/02/2022	BR153
26191	17/02/2022	17/02/2022	BR156
26224	16/02/2022	16/02/2022	BR167
26480	08/02/2022	11/02/2022	BR202
26620	03/02/2022	07/02/2022	BR158
27072	07/01/2022	13/01/2022	BR172
27074	07/01/2022	13/01/2022	BR171
27227	05/01/2022	07/01/2022	BR174
27231	05/01/2022	07/01/2022	BR161

	Quantidade no pedido	Separados	Entrega	Previsão De Instalação
4228	1	28/03/2024	28/03/2024	Desconhecido
17204	2	04/10/2022	04/10/2022	–
23366	1	17/05/2022	17/05/2022	–
24835	1	01/04/2022	01/04/2022	–
24836	1	01/04/2022	01/04/2022	–
24840	1	01/04/2022	01/04/2022	–
24842	1	01/04/2022	01/04/2022	–
24935	2	04/04/2022	05/04/2022	–
24936	2	04/04/2022	05/04/2022	–
25199	1	23/03/2022	23/03/2022	–
25342	2	18/03/2022	22/03/2022	–
26137	3	18/02/2022	18/02/2022	–
26191	1	17/02/2022	17/02/2022	–
26224	1	16/02/2022	16/02/2022	–

26480	3	10/02/2022	10/02/2022	-	
26620	1	04/02/2022	04/02/2022	-	
27072	2	13/01/2022	13/01/2022	-	
27074	10	13/01/2022	13/01/2022	-	
27227	4	11/01/2022	11/01/2022	-	
27231	4	11/01/2022	11/01/2022	-	

	Instalados	Local de instalações	Status	\
4228	02/04/2024	R. Lourenço Castanho, 193 - Vila Nova Conceiçã...	Done	
17204	-	KIP SOUND	Done	
23366	-	Endereço: Av. João Firmino, 165 - Assunção, Sã...	Done	
24835	-	LOCALIZA - PÁTIO CONSOLAÇÃO - LENILSON	Done	
24836	-	LOCALIZA - PÁTIO CONSOLAÇÃO - LENILSON	Done	
24840	-	LOCALIZA - ALPHAVILLE - KAIQUE	Done	
24842	-	LOCALIZA - SHOPPING SANTA CRUZ - EWERTON	Done	
24935	-	LOCALIZA - CPQ	Done	
24936	-	LOCALIZA - CPQ	Done	
25199	-	LOCALIZA - LAPA - LUIZ	Done	
25342	-	LOCALIZA - CONFINS	Done	
26137	-	LOCALIZA - VILA ANDRADE	Done	
26191	-	ZARP OSASCO	Done	
26224	-	LOCALIZA - SHOPPING D&D	Done	
26480	-	LOCALIZA - Rua Primavera, 1047, Bairro Rio Bra...	Done	
26620	-	RUA DOMINGOS DE MORAIS, 2564	Done	
27072	-	ZARP PAMPULHA	Done	
27074	-	ZARP PAMPULHA	Done	
27227	-	AVENIDA ARUANA 641 - TAMBORÉ/ FORNECEDOR TORQU...	Done	
27231	-	AVENIDA ARUANA 641 - TAMBORÉ/ FORNECEDOR TORQU...	Done	

	Frete	Custo de cofre	Custo de instalação	IPI	\
4228	91.15	2299.0	90.0	0.0	
17204	91.15	121.0	92.4	0.0	
23366	91.15	121.0	66.0	0.0	
24835	91.15	121.0	66.0	0.0	
24836	91.15	121.0	66.0	0.0	
24840	91.15	121.0	66.0	0.0	
24842	91.15	121.0	66.0	0.0	
24935	91.15	121.0	66.0	0.0	
24936	91.15	121.0	66.0	0.0	
25199	91.15	121.0	66.0	0.0	
25342	91.15	121.0	66.0	0.0	
26137	91.15	121.0	66.0	0.0	
26191	91.15	121.0	66.0	0.0	
26224	91.15	121.0	66.0	0.0	
26480	91.15	121.0	66.0	0.0	
26620	91.15	121.0	66.0	0.0	
27072	91.15	121.0	66.0	0.0	
27074	91.15	121.0	66.0	0.0	
27227	91.15	121.0	66.0	0.0	
27231	91.15	121.0	66.0	0.0	

	CNPJ	para faturamento	Local de entrega
4228	33.011.265/0001-44		mesmo do cliente
17204	Desconhecido		mesmo do cliente
23366	Desconhecido		Desconhecido
24835	Desconhecido		Desconhecido
24836	Desconhecido		Desconhecido
24840	Desconhecido		Desconhecido
24842	Desconhecido		Desconhecido
24935	Desconhecido		Desconhecido

```
24936           Desconhecido      Desconhecido
25199           Desconhecido      Desconhecido
25342           Desconhecido      Desconhecido
26137           Desconhecido      Desconhecido
26191           Desconhecido      Desconhecido
26224           Desconhecido      Desconhecido
26480           Desconhecido      Desconhecido
26620           Desconhecido      Desconhecido
27072           Desconhecido      Desconhecido
27074           Desconhecido      Desconhecido
27227           Desconhecido      Desconhecido
27231           Desconhecido      Desconhecido
```

In [16]:
```python
# Verificando o número de observações
print(f"Número de observações após remover duplicatas: {df.shape[0]}")
```

```
Número de observações após remover duplicatas: 27399
```

In [17]:
```python
# Removendo duplicatas e mantendo a primeira ocorrência
df = df.drop_duplicates()

# Verificando o número de observações após a remoção
print(f"Número de observações após remover duplicatas: {df.shape[0]}")
```

```
Número de observações após remover duplicatas: 27379
```

Seção 2: Análise de Dados

Principais Métricas Financeiras

In [18]:
```python
# Calculando a receita total
df['Receita'] = df['Quantidade no pedido'] * df['Custo de instalação']
receita_total = df['Receita'].sum()

# Identificando fatores importantes
fatores_importantes = {
    'Receita Média por Cliente': df.groupby('Cliente')['Receita'].mean().mean(),
    'Receita Média por Produto': df.groupby('Produto (nu cat)')['Receita'].mean().mean(),
    'Receita Média por Segmento': df.groupby('Segmento')['Receita'].mean().mean()
}

print(f"Receita Total: {receita_total}")
print(f"Fatores Importantes: {fatores_importantes}")
```

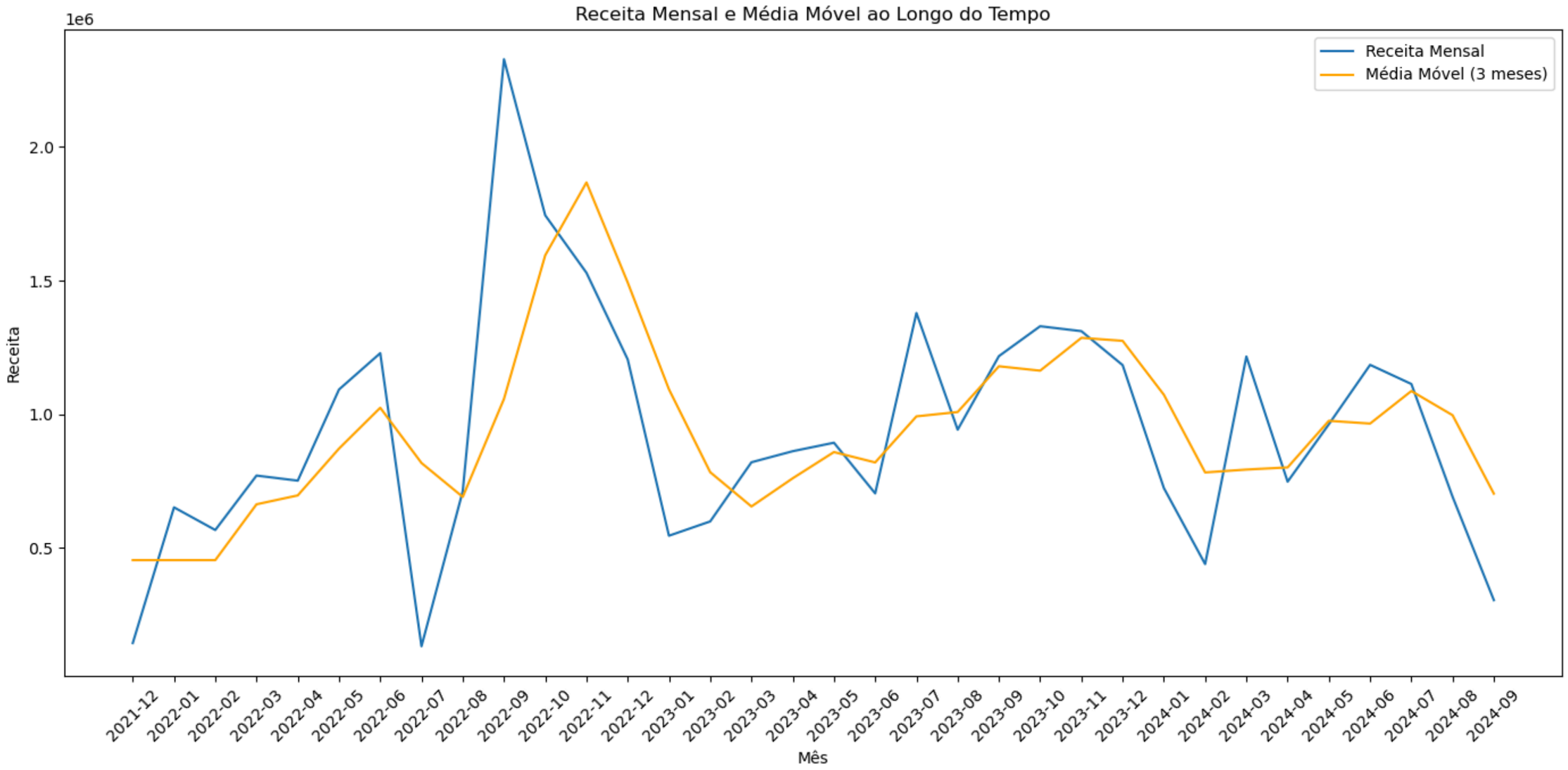
```
Receita Total: 32016270.446000002
Fatores Importantes: {'Receita Média por Cliente': 349.3897257885305, 'Receita Média por Produto': 181.9998452916312, 'Receita Média por Segmento': 555.3723616475808}
```

Visualização das Métricas ao Longo do Tempo

In [19]:
```python
# Convertendo a data para o formato datetime
df['Data de finalização (desejada)'] = pd.to_datetime(df['Data de finalização (desejada)'], format='%d/%m/%Y')

# Criando a coluna 'Mês' no formato de período
df['Mês'] = df['Data de finalização (desejada)'].dt.to_period('M').astype(str)  # Convertendo para string

# Verificando o tipo de dado da coluna 'Mês'
print(df['Mês'].dtype)
print(df['Mês'].unique())
```

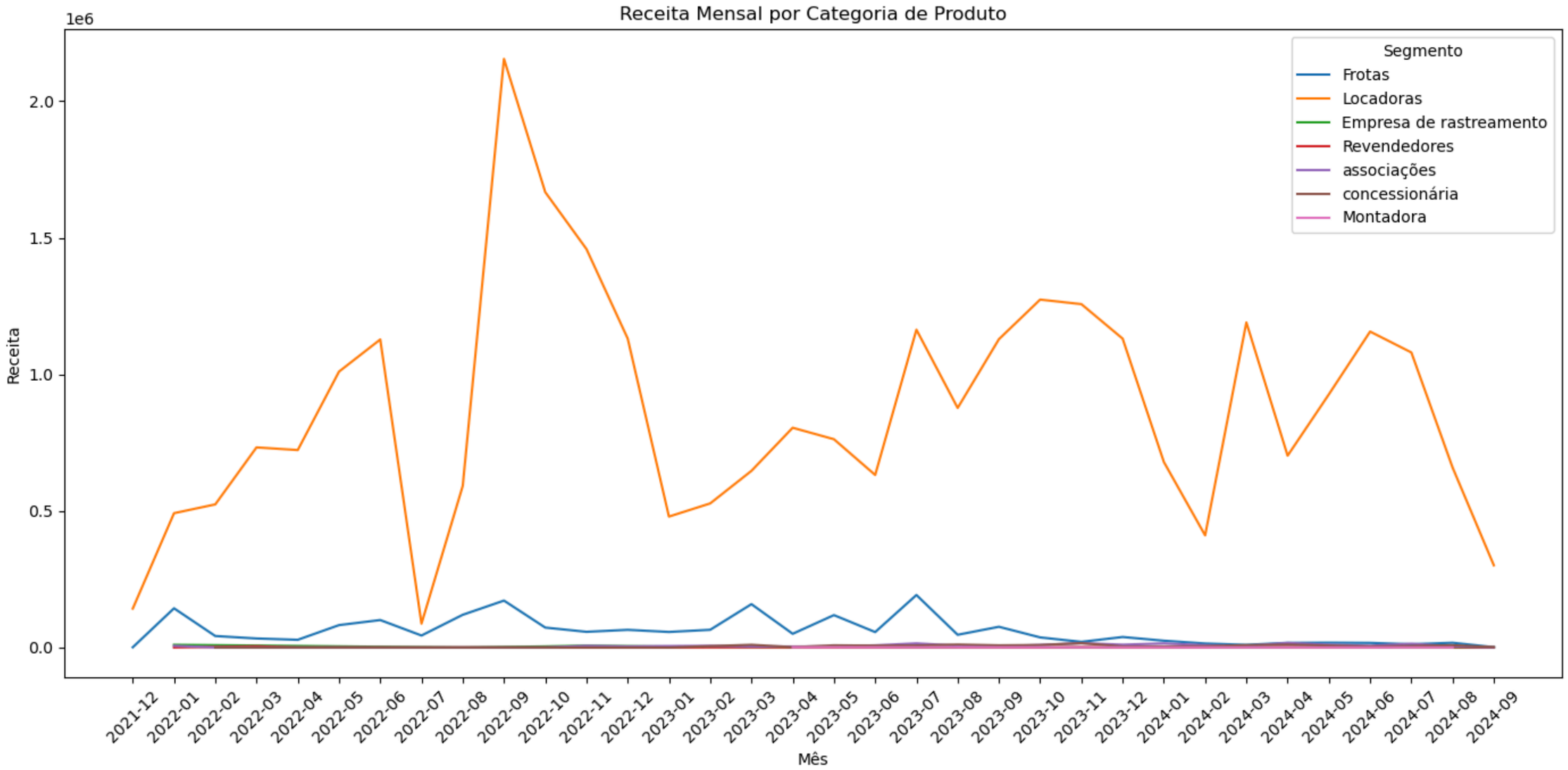
```python
# Verificando o tipo de dado da coluna 'Receita'
print(df['Receita'].dtype)
print(df['Receita'].unique())

# Corrigindo a coluna 'Receita'
df['Receita'] = pd.to_numeric(df['Receita'], errors='coerce')  # Convertendo para numérico
df = df.dropna(subset=['Receita'])  # Removendo linhas onde 'Receita' é NaN

# Agrupando e calculando a receita mensal
receita_mensal = df.groupby('Mês')['Receita'].sum().reset_index()

# Calculando a receita acumulada
receita_mensal['Receita Acumulada'] = receita_mensal['Receita'].cumsum()

# Calculando a média móvel de 3 meses
receita_mensal['Média Móvel (3 meses)'] = receita_mensal['Receita'].rolling(window=3).mean()

# Preenchendo valores ausentes na coluna 'Média Móvel (3 meses)' com o valor anterior (método de interpolação)
receita_mensal['Média Móvel (3 meses)'] = receita_mensal['Média Móvel (3 meses)'].bfill()

# Verificando o DataFrame
print(receita_mensal.head())

# Plotando a receita mensal e a média móvel
plt.figure(figsize=(14, 7))
sns.lineplot(data=receita_mensal, x='Mês', y='Receita', label='Receita Mensal')
sns.lineplot(data=receita_mensal, x='Mês', y='Média Móvel (3 meses)', label='Média Móvel (3 meses)', color='orange')
plt.title('Receita Mensal e Média Móvel ao Longo do Tempo')
plt.xlabel('Mês')
plt.ylabel('Receita')
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```

```
object
['2024-09' '2024-08' '2024-07' '2024-06' '2024-05' '2024-04' '2024-03'
 '2024-02' '2024-01' '2023-12' '2023-11' '2023-10' '2023-09' '2023-08'
 '2023-07' '2023-06' '2023-05' '2023-04' '2023-03' '2023-02' '2023-01'
 '2022-12' '2022-11' '2022-10' '2022-09' '2022-08' '2022-07' '2022-06'
 '2022-05' '2022-04' '2022-03' '2022-02' '2022-01' '2021-12']
float64
[   30.   150.    50. ... 20130. 17490. 29172.]
       Mês   Receita  Receita Acumulada  Média Móvel (3 meses)
0  2021-12  143352.0           143352.0               453662.0
1  2022-01  651222.0           794574.0               453662.0
2  2022-02  566412.0          1360986.0               453662.0
3  2022-03  770220.0          2131206.0               662618.0
4  2022-04  751146.0          2882352.0               695926.0
```

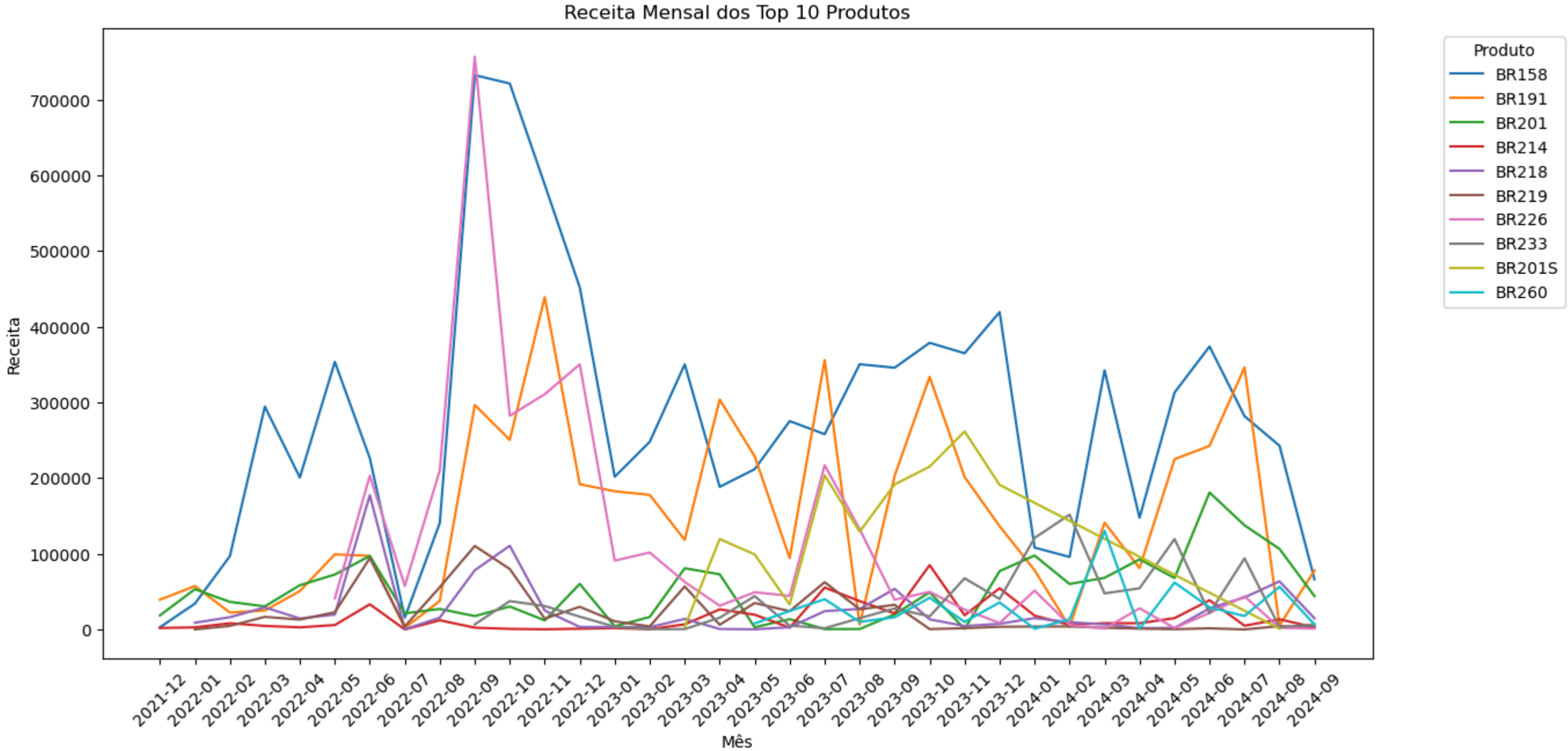


Análise de Tendências

In [20]:
```python
# Agrupando e calculando a receita total por segmento e mês
receita_categoria = df.groupby(['Mês', 'Segmento'])['Receita'].sum().reset_index()

# Criando um gráfico de linhas para cada categoria
plt.figure(figsize=(14, 7))
sns.lineplot(data=receita_categoria, x='Mês', y='Receita', hue='Segmento')
plt.title('Receita Mensal por Categoria de Produto')
plt.xlabel('Mês')
plt.ylabel('Receita')
plt.xticks(rotation=45)
plt.legend(title='Segmento')
plt.tight_layout()
plt.show()
```

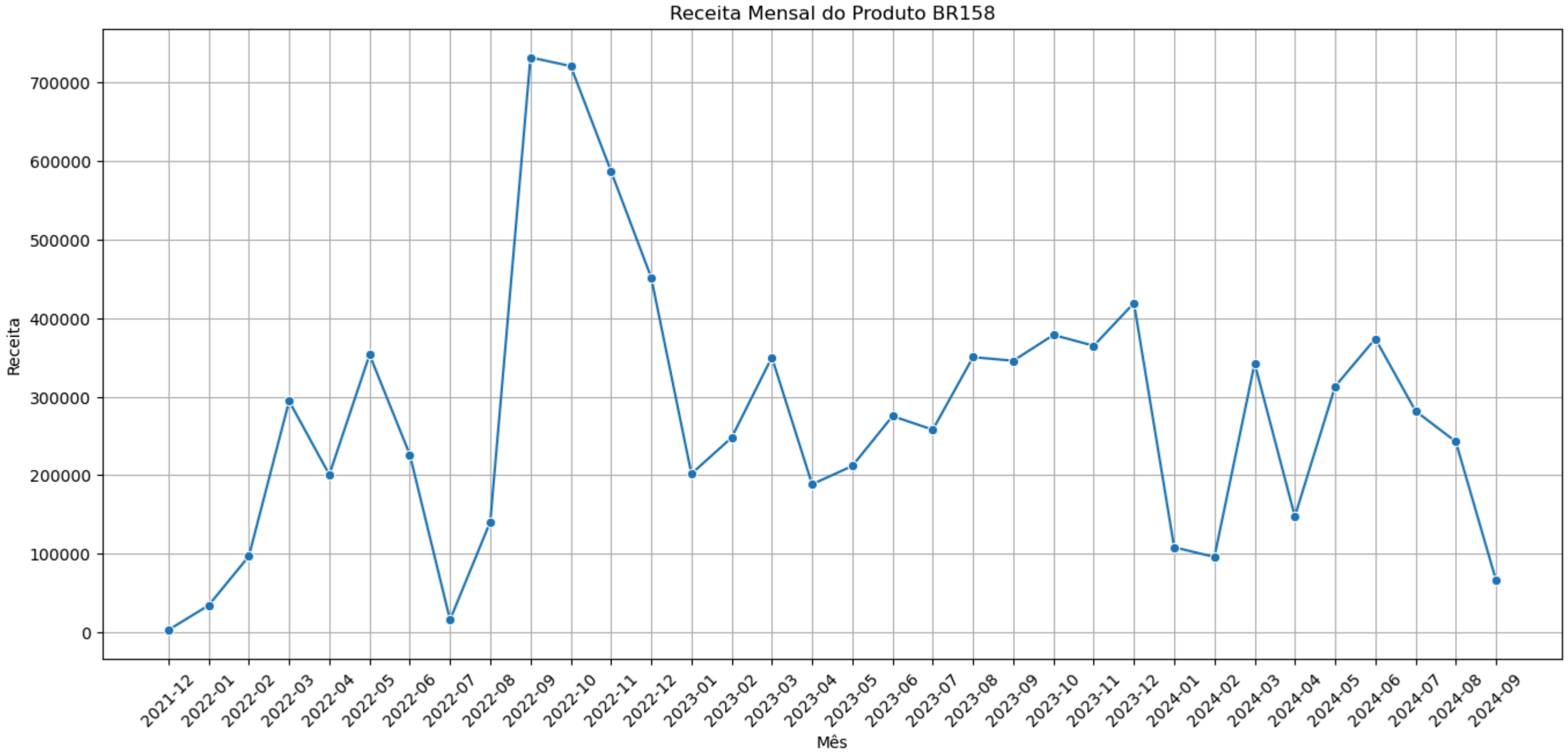


In [21]:
```python
# Agrupando e calculando a receita total por produto e mês
receita_produto = df.groupby(['Mês', 'Produto (nu cat)'])['Receita'].sum().reset_index()

# Selecionando os 10 produtos com maior receita total
top_produtos = receita_produto.groupby('Produto (nu cat)')['Receita'].sum().nlargest(10).index
receita_top_produto = receita_produto[receita_produto['Produto (nu cat)'].isin(top_produtos)]

# Criando um gráfico de linhas para os principais produtos
plt.figure(figsize=(14, 7))
sns.lineplot(data=receita_top_produto, x='Mês', y='Receita', hue='Produto (nu cat)')
plt.title('Receita Mensal dos Top 10 Produtos')
plt.xlabel('Mês')
plt.ylabel('Receita')
plt.xticks(rotation=45)
plt.legend(title='Produto', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout(pad=2.0)
plt.show()
```



In [22]:

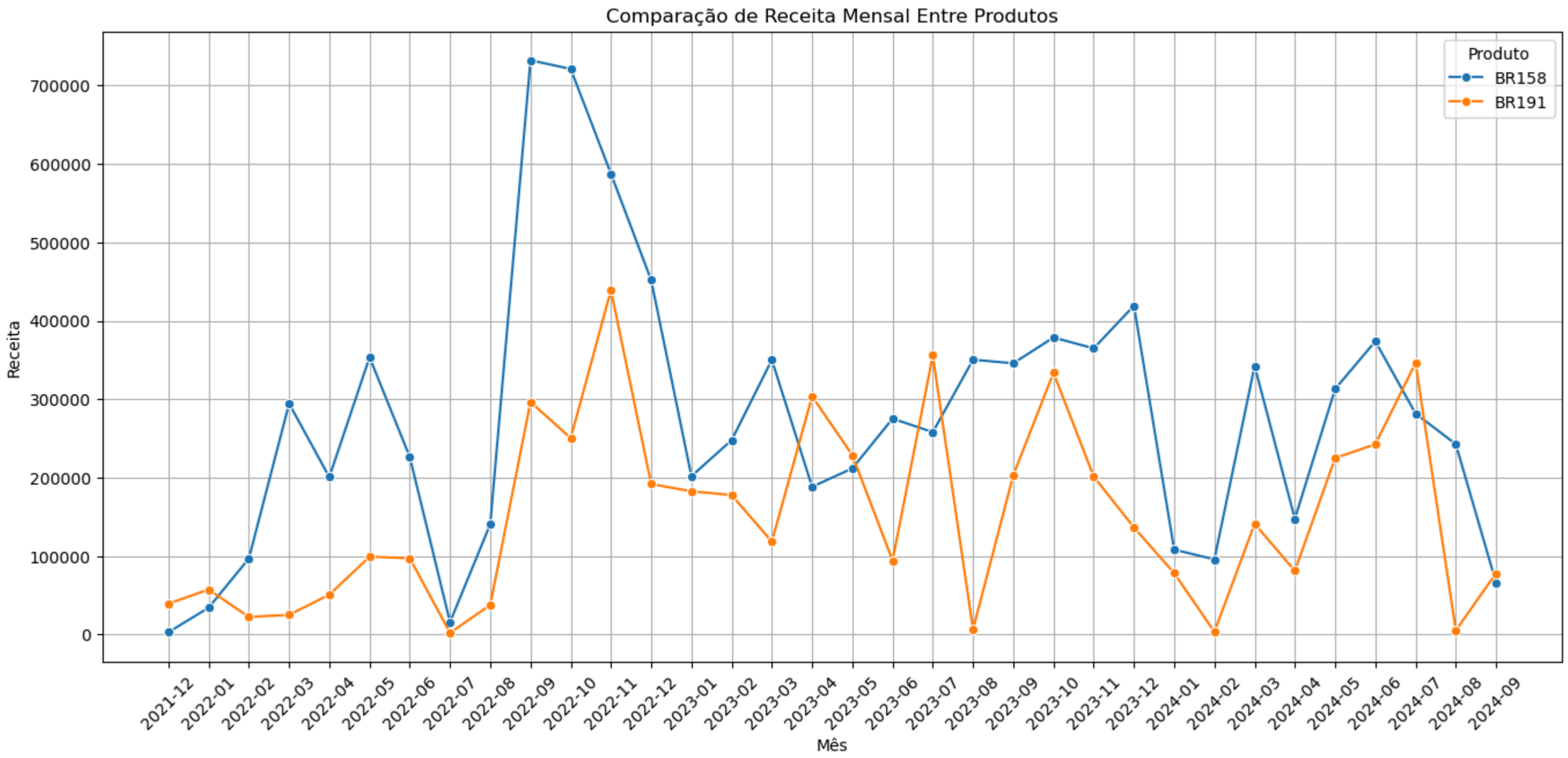
```
# Selecionando um produto específico para análise
produto_especifico = 'BR158'

# Filtrando dados para o produto específico
dados_produto = receita_produto[receita_produto['Produto (nu cat)'] == produto_especifico]

# Criando um gráfico de linha para o produto específico
plt.figure(figsize=(14, 7))
sns.lineplot(data=dados_produto, x='Mês', y='Receita', marker='o')
plt.title(f'Receita Mensal do Produto {produto_especifico}')
plt.xlabel('Mês')
plt.ylabel('Receita')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout(pad=2.0)
plt.show()
```



Análise Comparativa

In [23]:
```
# Selecionando múltiplos produtos para comparação
produtos_selecionados = ['BR158', 'BR191']  # Exemplo de códigos de produtos

# Filtrando dados para os produtos selecionados
dados_comparacao = receita_produto[receita_produto['Produto (nu cat)'].isin(produtos_selecionados)]

# Criando um gráfico de linha para comparação
plt.figure(figsize=(14, 7))
sns.lineplot(data=dados_comparacao, x='Mês', y='Receita', hue='Produto (nu cat)', marker='o')
plt.title('Comparação de Receita Mensal Entre Produtos')
plt.xlabel('Mês')
plt.ylabel('Receita')
plt.xticks(rotation=45)
plt.grid(True)
plt.legend(title='Produto')
```

```
plt.tight_layout(pad=2.0)
plt.show()
```



In [24]:
```
# Calculando estatísticas descritivas
estatisticas_produto = receita_produto.groupby('Produto (nu cat)')['Receita'].agg(['mean', 'median', 'std'])
print(estatisticas_produto)
```

```
                    mean  median        std
Produto (nu cat)
 ABM6B41            66.0    66.0        NaN
 BR152              70.0    70.0        NaN
 BR158              55.0    55.0  21.213203
 BR167              40.0    40.0        NaN
 BR167R             66.0    66.0        NaN
...                  ...     ...        ...
teste 1            198.0   198.0        NaN
teste1               0.0     0.0        NaN
teste2               0.0     0.0        NaN
tttt                66.0    66.0        NaN
wqr3i97             66.0    66.0        NaN

[5543 rows x 3 columns]
```

In [25]:

```
# Identificando outliers na receita
q1 = df['Receita'].quantile(0.25)
q3 = df['Receita'].quantile(0.75)
iqr = q3 - q1

outliers = df[(df['Receita'] < (q1 - 1.5 * iqr)) | (df['Receita'] > (q3 + 1.5 * iqr))]
print(outliers)
```

```
       Numero do pedido   Cliente   Segmento Data do recebimento  \
47                16572  Localiza  Locadoras          05/09/2024
48                16571    Movida  Locadoras          04/09/2024
73                16560  Localiza  Locadoras          04/09/2024
74                16560  Localiza  Locadoras          04/09/2024
75                16560  Localiza  Locadoras          04/09/2024
...                 ...       ...       ...                 ...
27396                9  Localiza  Locadoras          24/12/2021
27397                9  Localiza  Locadoras          24/12/2021
27398                9  Localiza  Locadoras          24/12/2021
27399                9  Localiza  Locadoras          24/12/2021
27403                6    Movida  Locadoras          23/12/2021

      Data de finalização (desejada) Produto (nu cat)  Quantidade no pedido  \
47                        2024-09-09           BR260                    55
48                        2024-09-05           BR233                    82
73                        2024-09-06           BR226                    12
74                        2024-09-06           BR219                    15
75                        2024-09-06           BR214                    21
...                              ...             ...                   ...
27396                     2021-12-28           BR167                   102
27397                     2021-12-28           BR161                   112
27398                     2021-12-28           BR191                   115
27399                     2021-12-29           BR155                   134
27403                     2021-12-30           BR201                   120

       Separados     Entrega Previsão De Instalação  ...  \
47            55  05/09/2024          Desconhecido  ...
48             0           -                     -  ...
73             0           -                     -  ...
74             0           -                     -  ...
75             0           -                     -  ...
...          ...         ...                   ...  ...
27396  28/12/2021  28/12/2021                     -  ...
27397  28/12/2021  28/12/2021                     -  ...
27398  28/12/2021  28/12/2021                     -  ...
27399  28/12/2021  28/12/2021                     -  ...
27403  08/01/2022  30/12/2021            07/01/2022  ...

         Local de instalações      Status  Frete  Custo de cofre  \
47     PDI TRANSZERO CAÇAPAVA  Processing  91.15         17765.0
48                PDI RENAULT  Processing  91.15          2004.0
73                OPERAÇÃO RJ  Processing  91.15         17765.0
74                OPERAÇÃO RJ  Processing  91.15         17765.0
75                OPERAÇÃO RJ  Processing  91.15         17765.0
...                       ...         ...    ...             ...
27396  ESTOQUE RIO DE JANEIRO        Done  91.15           121.0
27397  ESTOQUE RIO DE JANEIRO        Done  91.15           121.0
27398  ESTOQUE RIO DE JANEIRO        Done  91.15           121.0
27399  ESTOQUE RIO DE JANEIRO        Done  91.15           121.0
27403          PDI PIRACICABA        Done  91.15           121.0

       Custo de instalação  IPI  CNPJ    para faturamento  Local de entrega  \
47                    66.0  0.0       16.670.085/0001-55      Desconhecido
48                    66.0  0.0       07.976.147/0013-02  outro do cliente
73                    66.0  0.0       16.670.085/0001-55  mesmo do cliente
74                    66.0  0.0       16.670.085/0001-55  mesmo do cliente
75                    66.0  0.0       16.670.085/0001-55  mesmo do cliente
...                    ...  ...                      ...               ...
27396                 66.0  0.0             Desconhecido      Desconhecido
```

```
27397                66.0  0.0            Desconhecido      Desconhecido
27398                66.0  0.0            Desconhecido      Desconhecido
27399                66.0  0.0            Desconhecido      Desconhecido
27403                66.0  0.0            Desconhecido      Desconhecido

      Receita      Mês
47     3630.0  2024-09
48     5412.0  2024-09
73      792.0  2024-09
74      990.0  2024-09
75     1386.0  2024-09
...       ...      ...
27396  6732.0  2021-12
27397  7392.0  2021-12
27398  7590.0  2021-12
27399  8844.0  2021-12
27403  7920.0  2021-12

[4173 rows x 21 columns]
```

Análise Preditiva

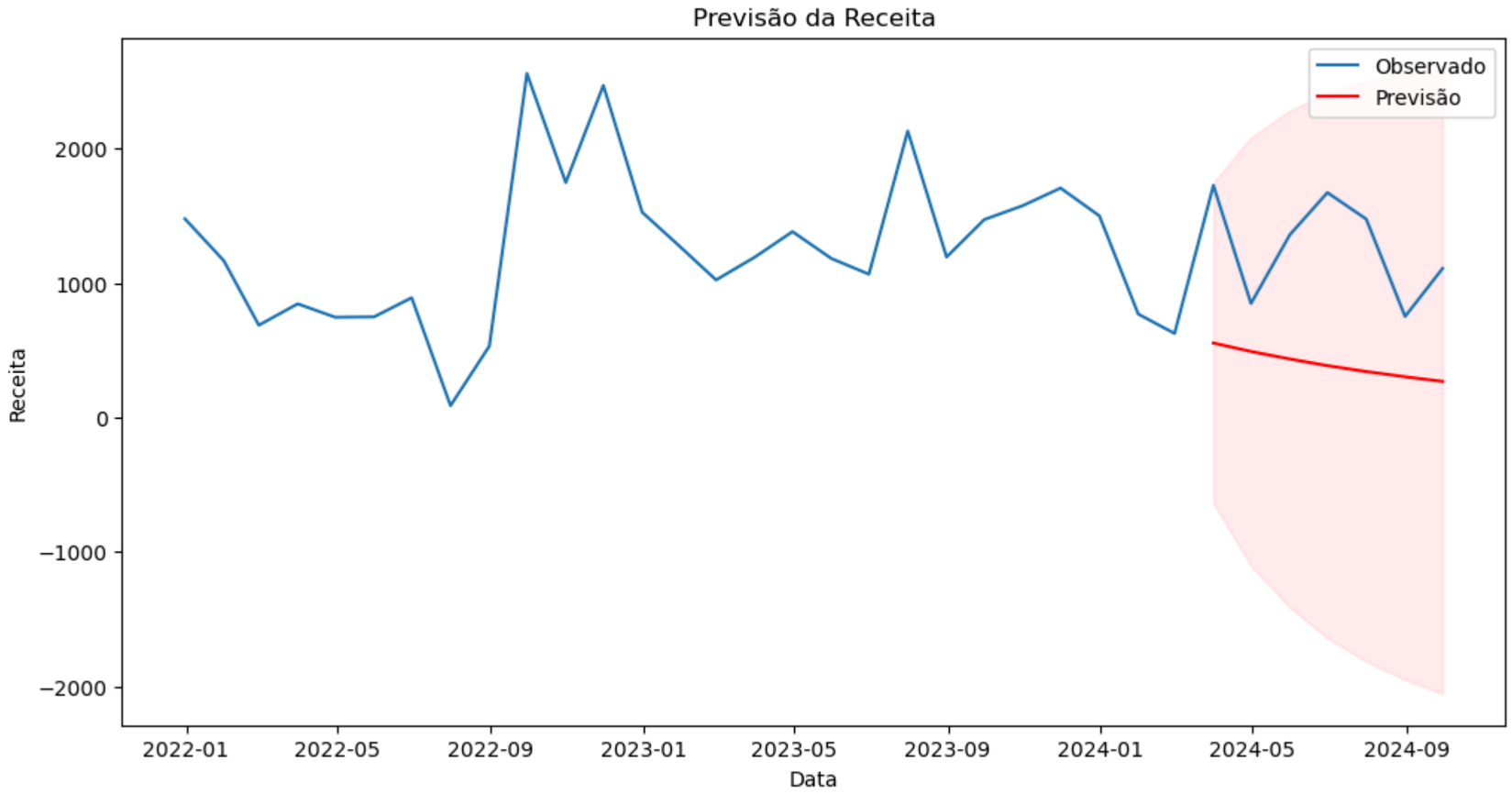
In [26]:
```python
# Ordenando por 'Produto (nu cat)' e 'Mês'
df_sorted = df.sort_values(by=['Produto (nu cat)', 'Mês'])
```

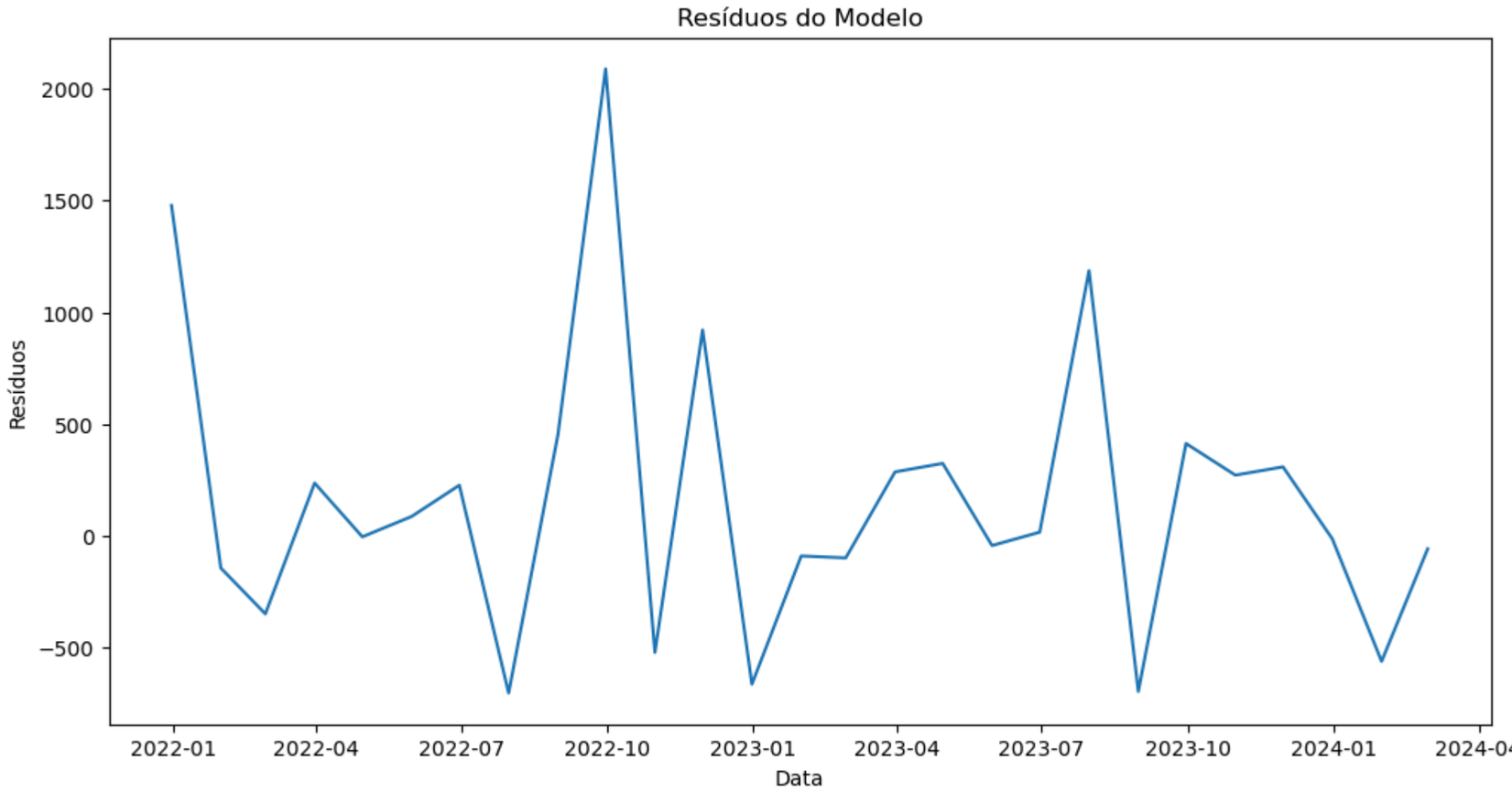
In [27]:
```python
# Convertendo a coluna de datas para datetime e configurando como índice
df['Data de finalização (desejada)'] = pd.to_datetime(df['Data de finalização (desejada)'])
df.set_index('Data de finalização (desejada)', inplace=True)

# Selecionando apenas as colunas numéricas
df_numeric = df.select_dtypes(include=[np.number])

# Resample para garantir que o índice está definido com uma frequência consistente
df_resampled = df_numeric.resample('ME').mean()  # Usando 'ME' para o último dia do mês

# Selecionando a coluna 'Receita' para análise
serie_temporal = df_resampled['Receita']

# Dividindo os dados em conjunto de treinamento e teste
train_size = int(len(serie_temporal) * 0.8)
train, test = serie_temporal[:train_size], serie_temporal[train_size:]

# Ajustando o modelo ARIMA com auto_arima
modelo_auto_arima = auto_arima(train, seasonal=False, stepwise=True, trace=True)
print(modelo_auto_arima.summary())

# Ajustando o modelo SARIMAX com os parâmetros encontrados
modelo_sarimax = SARIMAX(train, order=modelo_auto_arima.order)
resultado = modelo_sarimax.fit()
print(resultado.summary())

# Fazendo previsões
previsoes = resultado.get_forecast(steps=len(test))
previsoes_media = previsoes.predicted_mean
intervalo_confianca = previsoes.conf_int()

# Calculando erros
mae = mean_absolute_error(test, previsoes_media)
rmse = np.sqrt(mean_squared_error(test, previsoes_media))
```

```
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")

# Plotando os resultados
plt.figure(figsize=(12, 6))
plt.plot(serie_temporal.index, serie_temporal, label='Observado')
plt.plot(test.index, previsoes_media, color='red', label='Previsão')
plt.fill_between(test.index, intervalo_confianca.iloc[:, 0], intervalo_confianca.iloc[:, 1], color='pink', alpha=0.3)
plt.title('Previsão da Receita')
plt.xlabel('Data')
plt.ylabel('Receita')
plt.legend()
plt.show()

# Analisando os resíduos
residuos = resultado.resid
plt.figure(figsize=(12, 6))
plt.plot(residuos)
plt.title('Resíduos do Modelo')
plt.xlabel('Data')
plt.ylabel('Resíduos')
plt.show()

# Teste de normalidade dos resíduos
_, p_value = stats.normaltest(residuos)
print(f"P-valor do teste de normalidade dos resíduos: {p_value:.2f}")
```

```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0]             : AIC=inf, Time=0.12 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=468.415, Time=0.01 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=428.627, Time=0.02 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=451.989, Time=0.03 sec
 ARIMA(2,0,0)(0,0,0)[0]             : AIC=428.603, Time=0.02 sec
 ARIMA(3,0,0)(0,0,0)[0]             : AIC=430.480, Time=0.03 sec
 ARIMA(2,0,1)(0,0,0)[0]             : AIC=inf, Time=0.07 sec
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=428.235, Time=0.05 sec
 ARIMA(1,0,2)(0,0,0)[0]             : AIC=428.600, Time=0.08 sec
 ARIMA(0,0,2)(0,0,0)[0]             : AIC=444.827, Time=0.05 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=422.670, Time=0.03 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=421.155, Time=0.04 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=422.457, Time=0.01 sec
 ARIMA(0,0,2)(0,0,0)[0] intercept   : AIC=422.253, Time=0.06 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=420.549, Time=0.02 sec
 ARIMA(2,0,0)(0,0,0)[0] intercept   : AIC=422.591, Time=0.03 sec
 ARIMA(2,0,1)(0,0,0)[0] intercept   : AIC=424.508, Time=0.09 sec

Best model:  ARIMA(1,0,0)(0,0,0)[0] intercept
Total fit time: 0.756 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   27
Model:               SARIMAX(1, 0, 0)   Log Likelihood                -207.275
Date:                Wed, 11 Sep 2024   AIC                            420.549
Time:                        13:29:11   BIC                            424.437
Sample:                    12-31-2021   HQIC                           421.705
                         - 02-29-2024
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept    748.4536    285.260      2.624      0.009     189.354    1307.553
ar.L1          0.3900      0.228      1.711      0.087      -0.057       0.837
sigma2      2.791e+05   7.33e+04      3.810      0.000    1.36e+05    4.23e+05
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                11.34
Prob(Q):                              0.96   Prob(JB):                         0.00
Heteroskedasticity (H):               1.11   Skew:                             1.22
Prob(H) (two-sided):                  0.88   Kurtosis:                         5.03
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            2     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.86366D+00    |proj g|=  5.46266D-03

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
```

```
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    2      4      6      1     0     0   1.674D-05   7.863D+00
  F =   7.8634545866096710

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
                              SARIMAX Results
==============================================================================
Dep. Variable:                Receita   No. Observations:                   27
Model:               SARIMAX(1, 0, 0)   Log Likelihood                -212.313
Date:                Wed, 11 Sep 2024   AIC                            428.627
Time:                        13:29:11   BIC                            431.218
Sample:                    12-31-2021   HQIC                           429.397
                         - 02-29-2024
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8862      0.104      8.505      0.000       0.682       1.090
sigma2      3.702e+05   6.72e+04      5.510      0.000    2.39e+05    5.02e+05
===================================================================================
Ljung-Box (L1) (Q):                   2.86   Jarque-Bera (JB):                12.58
Prob(Q):                              0.09   Prob(JB):                         0.00
Heteroskedasticity (H):               1.79   Skew:                             1.21
Prob(H) (two-sided):                  0.40   Kurtosis:                         5.31
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
Mean Absolute Error (MAE): 880.90
Root Mean Squared Error (RMSE): 941.98
 This problem is unconstrained.
```

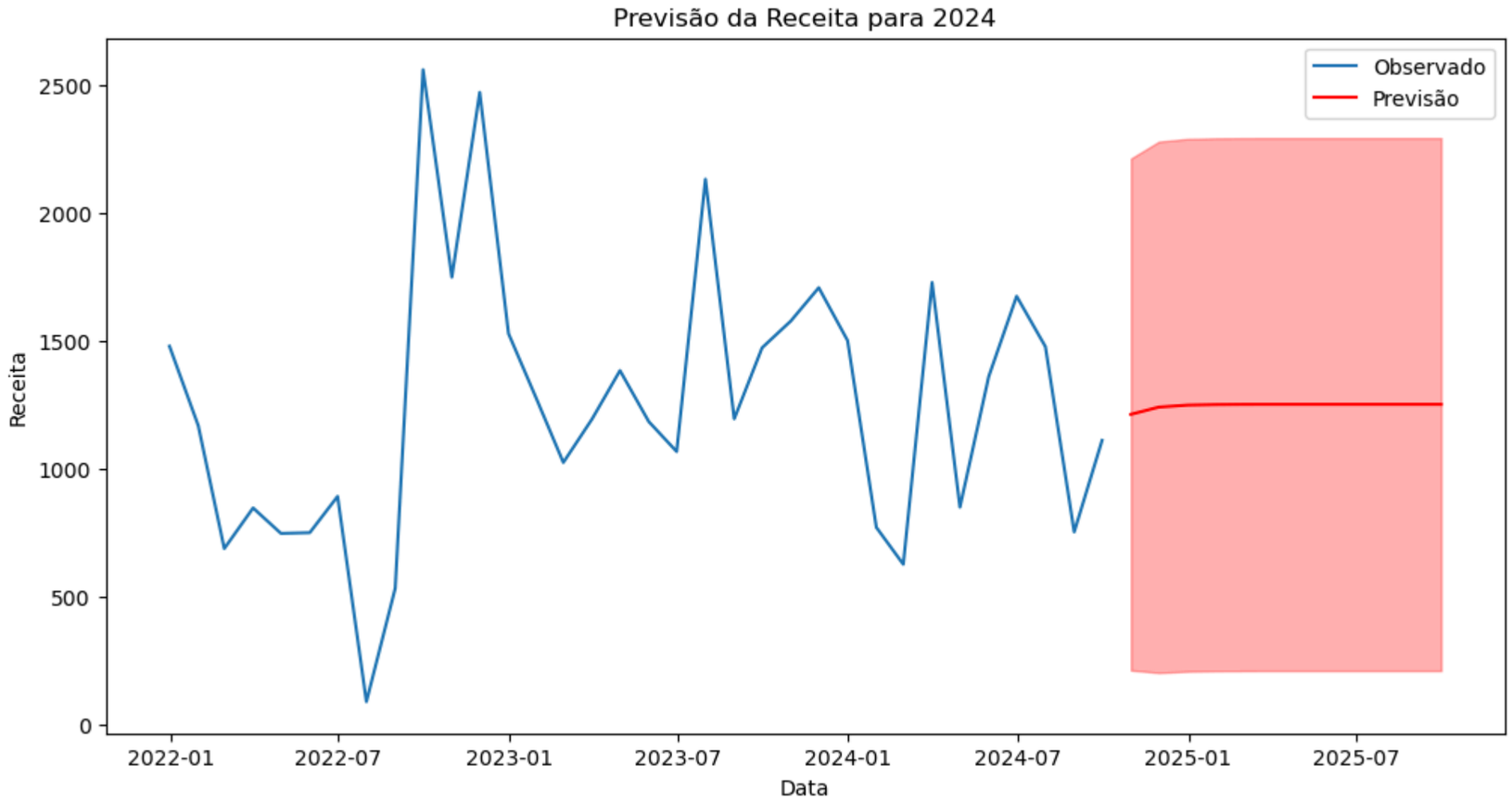

Previsão da Receita
Observado
Previsão
Receita
2000
1000
0
−1000
−2000
2022-01
2022-05
2022-09
2023-01
2023-05
2023-09
2024-01
2024-05
2024-09
Data




```
P-valor do teste de normalidade dos resíduos: 0.01
```

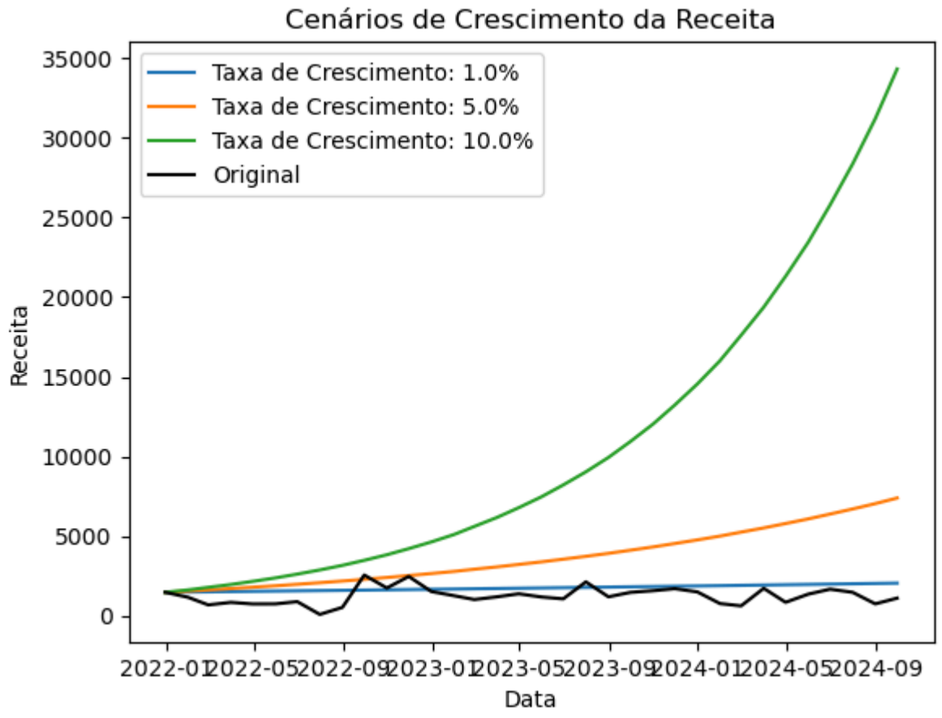
Previsão da Receita para 2024

In [28]:
```python
# Ajustando o modelo ARIMA
modelo = ARIMA(serie_temporal, order=(1, 0, 0))  # Ordem pode ser ajustada conforme o resultado da análise de AIC
resultado = modelo.fit()

# Previsão para o futuro
previsao = resultado.get_forecast(steps=12)  # Previsão para 12 meses
previsao_media = previsao.predicted_mean
previsao_intervalo = previsao.conf_int()

# Visualizando a previsão
plt.figure(figsize=(12, 6))
plt.plot(serie_temporal, label='Observado')
plt.plot(previsao_media, color='red', label='Previsão')
plt.fill_between(previsao_intervalo.index, previsao_intervalo.iloc[:, 0], previsao_intervalo.iloc[:, 1], color='red', alpha=0.3)
plt.title('Previsão da Receita para 2024')
plt.xlabel('Data')
plt.ylabel('Receita')
plt.legend()
plt.show()
```



Análise de Cenários

In [29]:
```python
# Ajustando parâmetros
def ajustar_crescimento(serie, taxa_crescimento):
    serie_ajustada = serie.copy()
    for i in range(1, len(serie_ajustada)):
        serie_ajustada.iloc[i] = serie_ajustada.iloc[i - 1] * (1 + taxa_crescimento)
    return serie_ajustada

# Analisando diferentes cenários
taxas_crescimento = [0.01, 0.05, 0.10]
for taxa in taxas_crescimento:
    serie_cenario = ajustar_crescimento(serie_temporal, taxa)
    plt.plot(serie_cenario, label=f'Taxa de Crescimento: {taxa*100}%')

plt.plot(serie_temporal, label='Original', color='black')
plt.title('Cenários de Crescimento da Receita')
plt.xlabel('Data')
plt.ylabel('Receita')
plt.legend()
plt.show()
```



Visualização e Relatórios

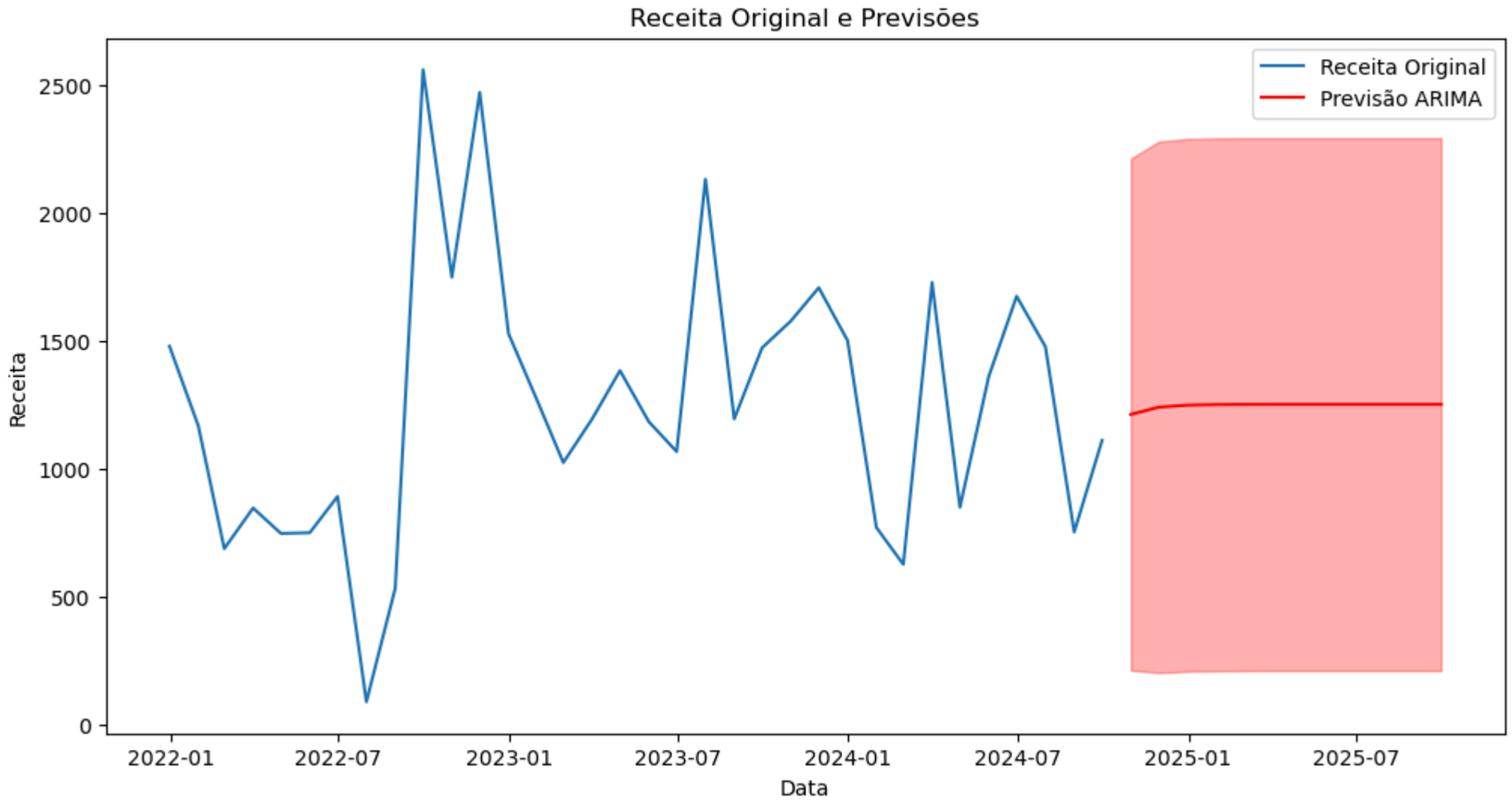
In [30]:

```python
# Obtendo os resíduos
residuos = resultado.resid

# Teste de Jarque-Bera para normalidade dos resíduos
jb_test = sm.stats.jarque_bera(residuos)

# Extraindo valores do teste
jb_stat = jb_test[0]  # Estatístico do teste
jb_pvalue = jb_test[1]  # P-valor

# Criando o relatório atualizado
print('Relatório de Previsão e Análise Preditiva')
print('==========================================')
print(f'\nModelo ARIMA selecionado: ARIMA(1,0,0)')
print(f'\nMédia da Receita Prevista para 2024:')
print(previsao_media.describe())
print(f'\nErro Médio Absoluto (MAE): {resultado.mse}')
print(f'\nErro Quadrático Médio (RMSE): {np.sqrt(resultado.mse)}')
print(f'\nP-valor do teste de Jarque-Bera dos resíduos: {jb_pvalue}')
```

```
Relatório de Previsão e Análise Preditiva
==================================

Modelo ARIMA selecionado: ARIMA(1,0,0)

Média da Receita Prevista para 2024:
count      12.000000
mean     1246.039084
std        11.333947
min      1211.439683
25%      1249.170396
50%      1250.508903
75%      1250.548876
max      1250.549973
Name: predicted_mean, dtype: float64

Erro Médio Absoluto (MAE): 254378.53595053294

Erro Quadrático Médio (RMSE): 504.3595304448335

P-valor do teste de Jarque-Bera dos resíduos: 0.040963514847225775
```

count 12.000000 mean 1246.039084 std 11.333947 min 1211.439683 25% 1249.170396 50% 1250.508903 75% 1250.548876 max 1250.549973 Name: predicted_mean, dtype: float64

In [31]:
```python
# Gráfico da série temporal e previsões
plt.figure(figsize=(12, 6))
sns.lineplot(data=serie_temporal, label='Receita Original')
sns.lineplot(data=previsao_media, label='Previsão ARIMA', color='red')
plt.fill_between(previsao_intervalo.index, previsao_intervalo.iloc[:, 0], previsao_intervalo.iloc[:, 1], color='red', alpha=0.3)
plt.title('Receita Original e Previsões')
plt.xlabel('Data')
plt.ylabel('Receita')
plt.legend()
plt.show()
```

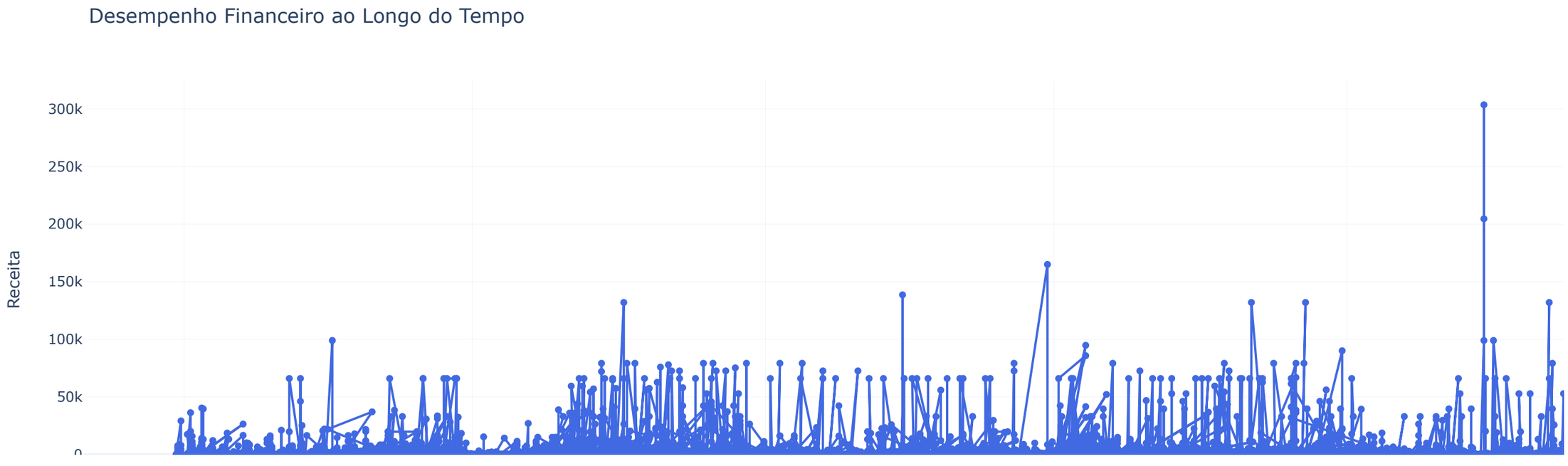


Gráficos Interativos

(só funciona no Jupyter Notebook)

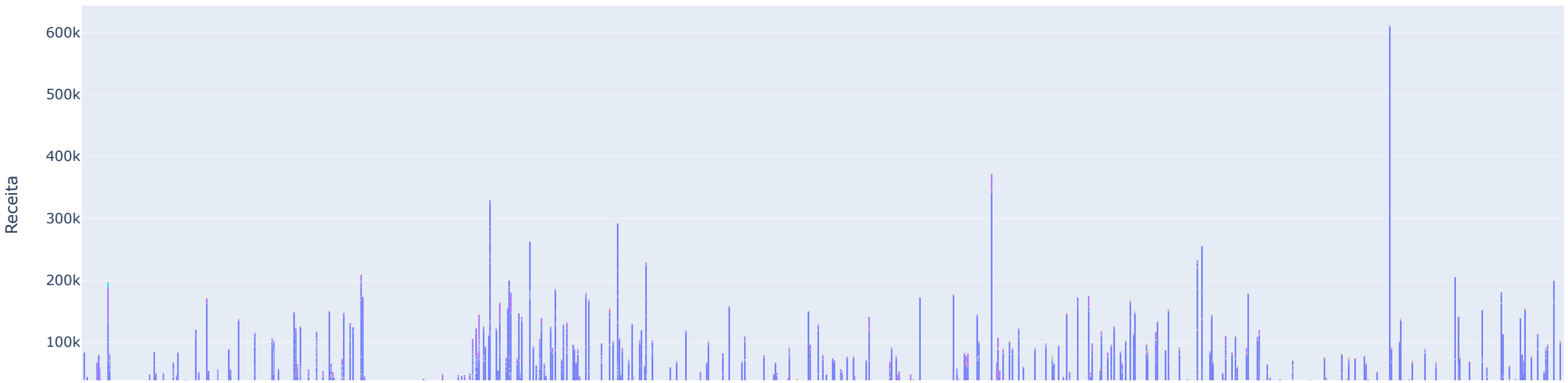
In [32]:
```
# Convertendo a coluna 'Data de finalização (desejada)' para datetime
df.index = pd.to_datetime(df.index, format='%Y-%m-%d')

# Criando uma série temporal para 'Receita'
serie_temporal = df[['Receita']].copy()

# Criando o gráfico interativo
fig = go.Figure()

# Adicionando a série temporal da receita ao gráfico
fig.add_trace(go.Scatter(x=serie_temporal.index, y=serie_temporal['Receita'],
                         mode='lines+markers',
                         name='Receita',
                         line=dict(color='royalblue', width=2),
                         marker=dict(size=6, color='royalblue', symbol='circle')))

# Atualizando o layout do gráfico
fig.update_layout(
    title='Desempenho Financeiro ao Longo do Tempo',
    xaxis_title='Data',
    yaxis_title='Receita',
    template='plotly_white'
```

```
)

# Exibindo o gráfico
fig.show()
```

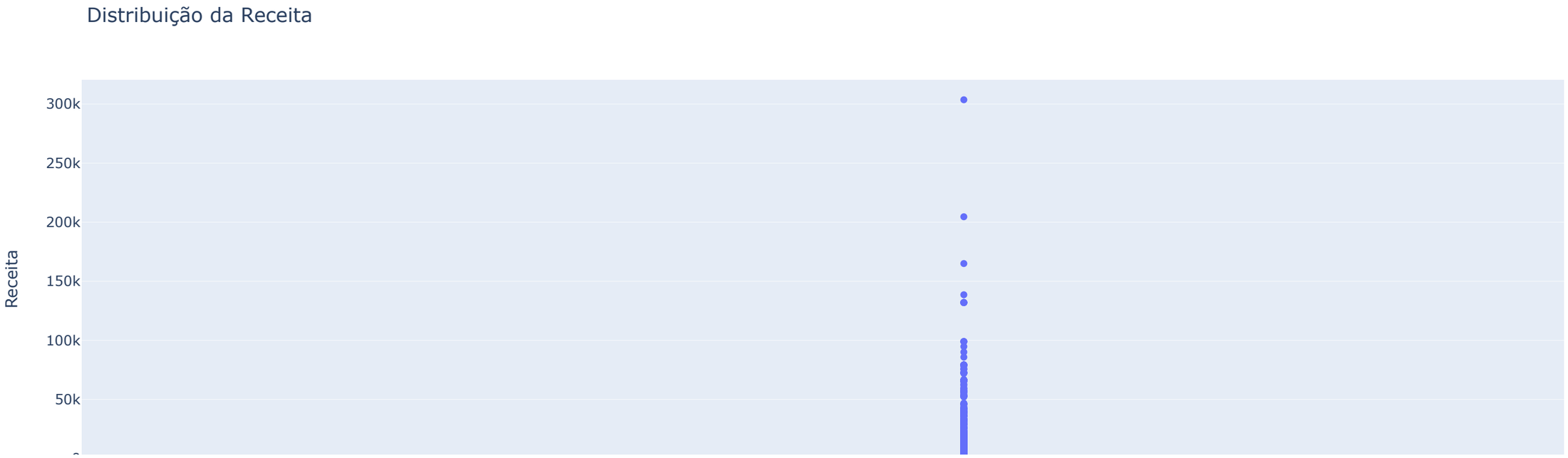


In [33]:
```
# Criando um gráfico de barras empilhadas por segmento
fig = px.bar(df, x=df.index, y='Receita', color='Segmento', title='Receita por Segmento ao Longo do Tempo',
             labels={'Receita': 'Receita', 'Data de finalização (desejada)': 'Data'})
fig.update_layout(barmode='stack')
fig.show()
```

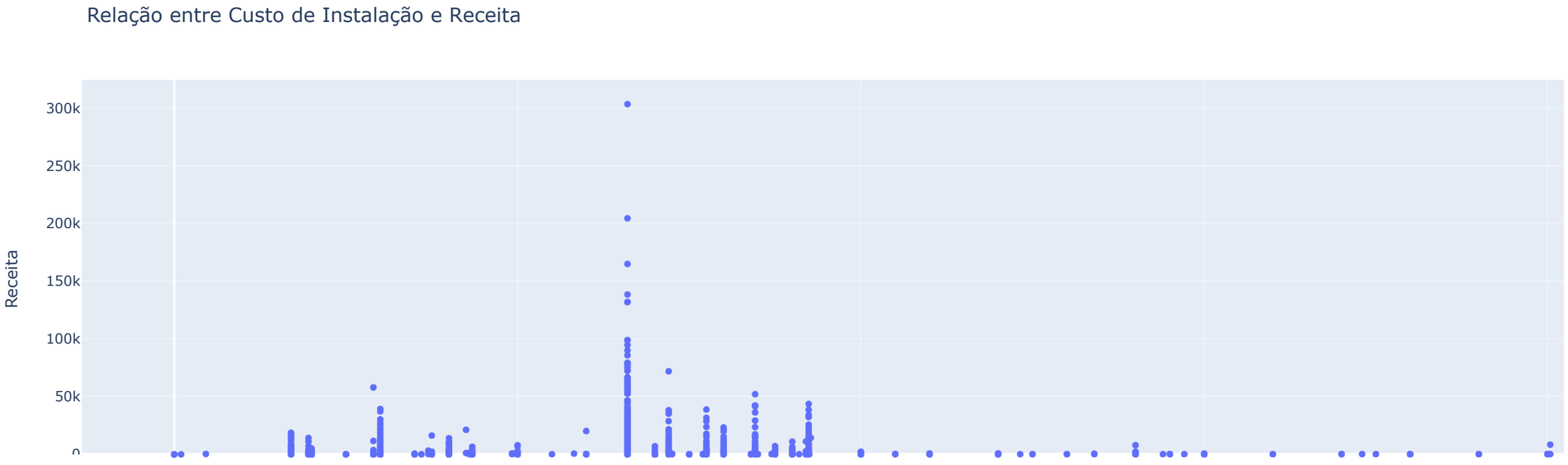
```
/Users/tsumano/opt/anaconda3/lib/python3.9/site-packages/plotly/express/_core.py:1979: FutureWarning:

When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future version of pandas. Pass `(name,)` instead of `name` to silence this wa
rning.

/Users/tsumano/opt/anaconda3/lib/python3.9/site-packages/_plotly_utils/basevalidators.py:106: FutureWarning:

The behavior of DatetimeProperties.to_pydatetime is deprecated, in a future version this will return a Series containing python datetime objects instead of an ndarray. To reta
in the old behavior, call `np.array` on the result
```



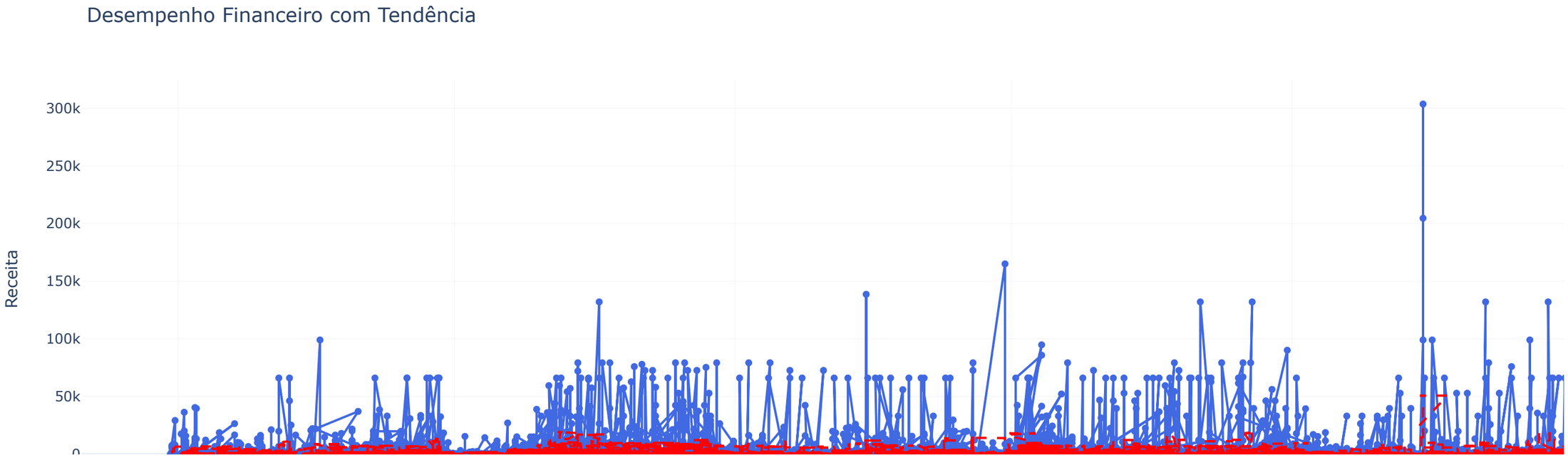
In [34]:
```
fig = px.box(df, y='Receita', title='Distribuição da Receita')
fig.show()
```



In [35]:
```
fig = px.scatter(df, x='Custo de instalação', y='Receita', title='Relação entre Custo de Instalação e Receita',
                 labels={'Custo de instalação': 'Custo de Instalação', 'Receita': 'Receita'})
fig.show()
```



In [36]:
```
fig = go.Figure()

# Linha da receita
fig.add_trace(go.Scatter(x=serie_temporal.index, y=serie_temporal['Receita'],
                         mode='lines+markers', name='Receita',
                         line=dict(color='royalblue', width=2),
                         marker=dict(size=6, color='royalblue', symbol='circle')))

# Linha de tendência
fig.add_trace(go.Scatter(x=serie_temporal.index, y=serie_temporal['Receita'].rolling(window=12).mean(),
                         mode='lines', name='Tendência',
                         line=dict(color='red', width=2, dash='dash')))

fig.update_layout(
    title='Desempenho Financeiro com Tendência',
    xaxis_title='Data',
    yaxis_title='Receita',
    template='plotly_white'
)

fig.show()
```



In [37]:

```python
# Preparando os dados para o gráfico de radar
dados_segmento = df.groupby('Segmento').agg({'Receita': 'sum'}).reset_index()

# Criando o gráfico de radar
fig = go.Figure()

# Adicionando cada segmento ao gráfico
for segmento in dados_segmento['Segmento']:
    dados = dados_segmento[dados_segmento['Segmento'] == segmento]
    fig.add_trace(go.Scatterpolar(
        r=dados['Receita'],
        theta=[segmento] * len(dados),
        fill='toself',
        name=segmento
    ))

# Atualizando o layout do gráfico
fig.update_layout(
    polar=dict(
        radialaxis=dict(visible=True, range=[0, dados_segmento['Receita'].max()]),
    ),
    showlegend=True,
    title='Comparação de Receita por Segmento'
)

fig.show()
```

Comparação de Receita por Segmento