In [32]:
```
import pandas as pd
df=pd.read_csv(r"website_wata.csv")
```

In [33]:
```
df
```

Out[33]:

	Page Views	Session Duration	Bounce Rate	Traffic Source	Time on Page	Previous Visits	Conversion Rate
0	5	11.051381	0.230652	Organic	3.890460	3	1.0
1	4	3.429316	0.391001	Social	8.478174	0	1.0
2	4	1.621052	0.397986	Organic	9.636170	2	1.0
3	5	3.629279	0.180458	Organic	2.071925	3	1.0
4	5	4.235843	0.291541	Paid	1.960654	5	1.0
...	...	...	...	...	...	...	...
1995	1	2.724513	0.207187	Referral	1.324206	2	1.0
1996	3	0.392856	0.095559	Organic	3.824416	1	1.0
1997	4	9.899823	0.446622	Organic	1.288675	1	1.0
1998	3	0.393319	0.278340	Paid	5.037584	2	1.0
1999	3	0.882638	0.338026	Direct	5.186908	3	1.0

2000 rows × 7 columns

In [34]:
```
df['Conversion Rate'].isnull().sum()
```

Out[34]:
```
0
```

In [35]:
```
df['Traffic Source'].unique()
```

Out[35]:
```
array(['Organic', 'Social', 'Paid', 'Direct', 'Referral'], dtype=object)
```

In [36]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Traffic Source'] = le.fit_transform(df['Traffic Source'])
```

In [37]:
```
df['Session Duration'] = df['Session Duration'].round(2)
df['Bounce Rate'] = df['Bounce Rate'].round(2)
df['Time on Page'] = df['Time on Page'].round(2)
```

In [38]:
```
df
```

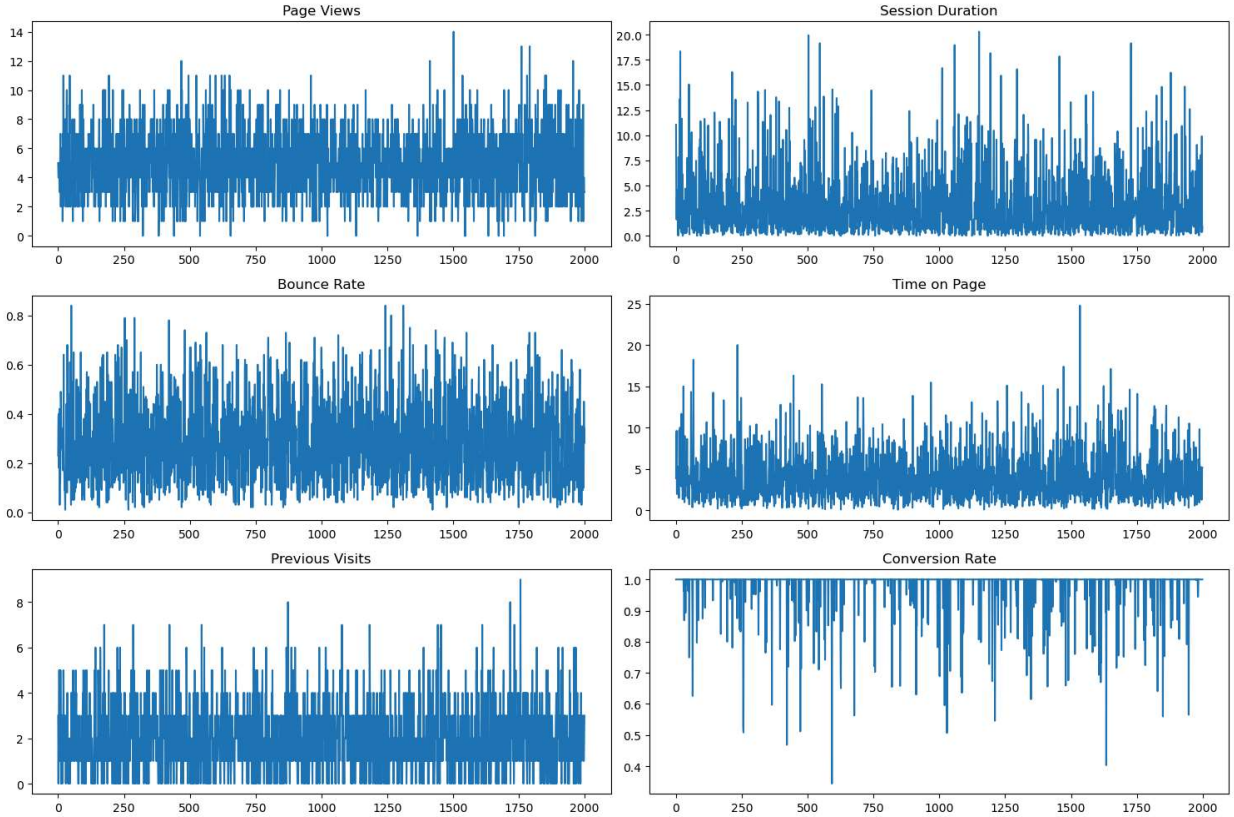
Out[38]:

	Page Views	Session Duration	Bounce Rate	Traffic Source	Time on Page	Previous Visits	Conversion Rate
0	5	11.05	0.23	1	3.89	3	1.0
1	4	3.43	0.39	4	8.48	0	1.0
2	4	1.62	0.40	1	9.64	2	1.0
3	5	3.63	0.18	1	2.07	3	1.0
4	5	4.24	0.29	2	1.96	5	1.0
...	...	...	...	...	...	...	...
1995	1	2.72	0.21	3	1.32	2	1.0
1996	3	0.39	0.10	1	3.82	1	1.0
1997	4	9.90	0.45	1	1.29	1	1.0
1998	3	0.39	0.28	2	5.04	2	1.0
1999	3	0.88	0.34	0	5.19	3	1.0

2000 rows × 7 columns

In [39]:
```
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
```

In [40]:
```
plt.figure(figsize=(15, 10))
plt.subplot(3, 2, 1)
df['Page Views'].plot(title='Page Views')
plt.subplot(3, 2, 2)
df['Session Duration'].plot(title='Session Duration')
plt.subplot(3, 2, 3)
df['Bounce Rate'].plot(title='Bounce Rate')
plt.subplot(3, 2, 4)
df['Time on Page'].plot(title='Time on Page')
plt.subplot(3, 2, 5)
df['Previous Visits'].plot(title='Previous Visits')
plt.subplot(3, 2, 6)
df['Conversion Rate'].plot(title='Conversion Rate')
plt.tight_layout()
plt.show()
```



In [43]:
```
df['Date'] = pd.date_range(start='2023-01-01', periods=df.shape[0], freq='D')
df.set_index('Date', inplace=True)
```

In [44]:
```
df
```

Out[44]:

Date	Page Views	Session Duration	Bounce Rate	Traffic Source	Time on Page	Previous Visits	Conversion Rate
2023-01-01	5	11.05	0.23	1	3.89	3	1.0
2023-01-02	4	3.43	0.39	4	8.48	0	1.0
2023-01-03	4	1.62	0.40	1	9.64	2	1.0
2023-01-04	5	3.63	0.18	1	2.07	3	1.0
2023-01-05	5	4.24	0.29	2	1.96	5	1.0
...	...	...	...	...	...	...	...
2028-06-18	1	2.72	0.21	3	1.32	2	1.0
2028-06-19	3	0.39	0.10	1	3.82	1	1.0
2028-06-20	4	9.90	0.45	1	1.29	1	1.0
2028-06-21	3	0.39	0.28	2	5.04	2	1.0
2028-06-22	3	0.88	0.34	0	5.19	3	1.0

2000 rows × 7 columns

In [45]:
```
df['2023-01']
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_12332\4022346285.py:1: FutureWarning: Indexing a DataFrame with a datetimelike index using a single string to slice the rows, like `frame[string]`, is deprecated and will be removed in a future version. Use `frame.loc[string]` instead.
  df['2023-01']
```

Out[45]:

Date	Page Views	Session Duration	Bounce Rate	Traffic Source	Time on Page	Previous Visits	Conversion Rate
2023-01-01	5	11.05	0.23	1	3.89	3	1.00000
2023-01-02	4	3.43	0.39	4	8.48	0	1.00000
2023-01-03	4	1.62	0.40	1	9.64	2	1.00000
2023-01-04	5	3.63	0.18	1	2.07	3	1.00000
2023-01-05	5	4.24	0.29	2	1.96	5	1.00000
2023-01-06	3	4.54	0.42	4	3.44	2	1.00000
2023-01-07	5	1.95	0.03	4	2.12	1	1.00000
2023-01-08	4	1.69	0.25	2	3.48	5	1.00000
2023-01-09	6	0.03	0.12	1	5.29	1	1.00000
2023-01-10	7	7.83	0.21	2	4.06	5	1.00000
2023-01-11	2	2.77	0.49	0	2.97	3	1.00000
2023-01-12	5	0.68	0.45	3	1.50	0	1.00000
2023-01-13	5	7.56	0.26	4	9.69	0	1.00000
2023-01-14	6	7.39	0.26	2	4.98	2	1.00000
2023-01-15	4	13.58	0.32	2	5.80	3	1.00000
2023-01-16	6	6.22	0.30	2	10.03	2	1.00000
2023-01-17	6	18.34	0.34	3	2.98	0	1.00000
2023-01-18	1	0.84	0.25	4	8.30	1	1.00000
2023-01-19	7	5.54	0.26	2	8.79	3	1.00000
2023-01-20	2	5.20	0.17	2	2.89	2	1.00000

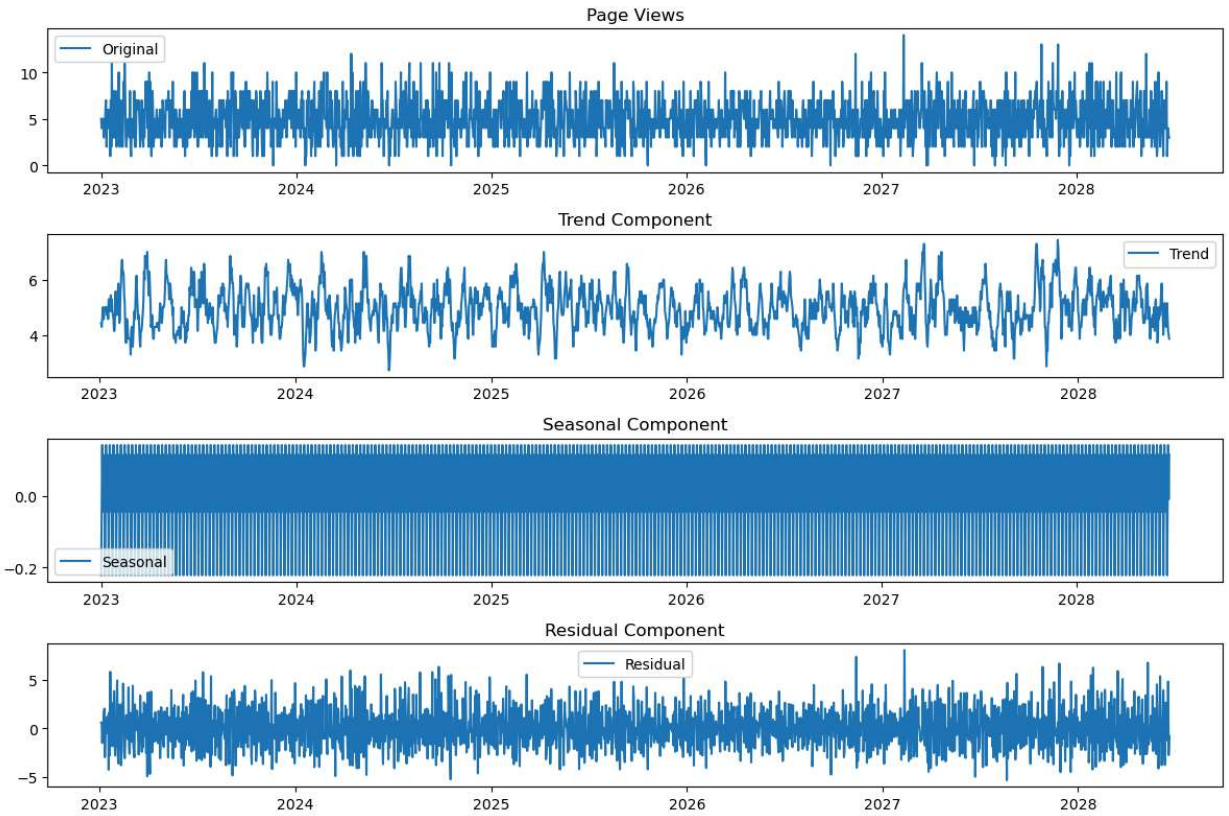
Date	Page Views	Session Duration	Bounce Rate	Traffic Source	Time on Page	Previous Visits	Conversion Rate
2023-01-21	11	0.79	0.25	1	11.69	5	1.00000
2023-01-22	4	0.31	0.64	1	4.16	4	1.00000
2023-01-23	3	11.65	0.29	0	6.03	1	1.00000
2023-01-24	8	1.05	0.18	1	3.73	2	1.00000
2023-01-25	3	6.14	0.09	4	5.08	2	1.00000
2023-01-26	3	0.01	0.12	3	1.34	3	1.00000
2023-01-27	5	0.06	0.06	4	7.05	1	1.00000
2023-01-28	8	1.88	0.01	0	2.11	2	1.00000
2023-01-29	3	0.82	0.16	3	15.02	3	1.00000
2023-01-30	2	3.51	0.14	4	1.55	0	0.96308
2023-01-31	5	2.50	0.12	1	3.30	2	1.00000

In [46]:
```
decomposition_page_views = sm.tsa.seasonal_decompose(df['Page Views'], model='additive
decomposition_session_duration = sm.tsa.seasonal_decompose(df['Session Duration'], mod
```

In [48]:
```
trend_page_views = decomposition_page_views.trend
seasonal_page_views = decomposition_page_views.seasonal
residual_page_views = decomposition_page_views.resid
```

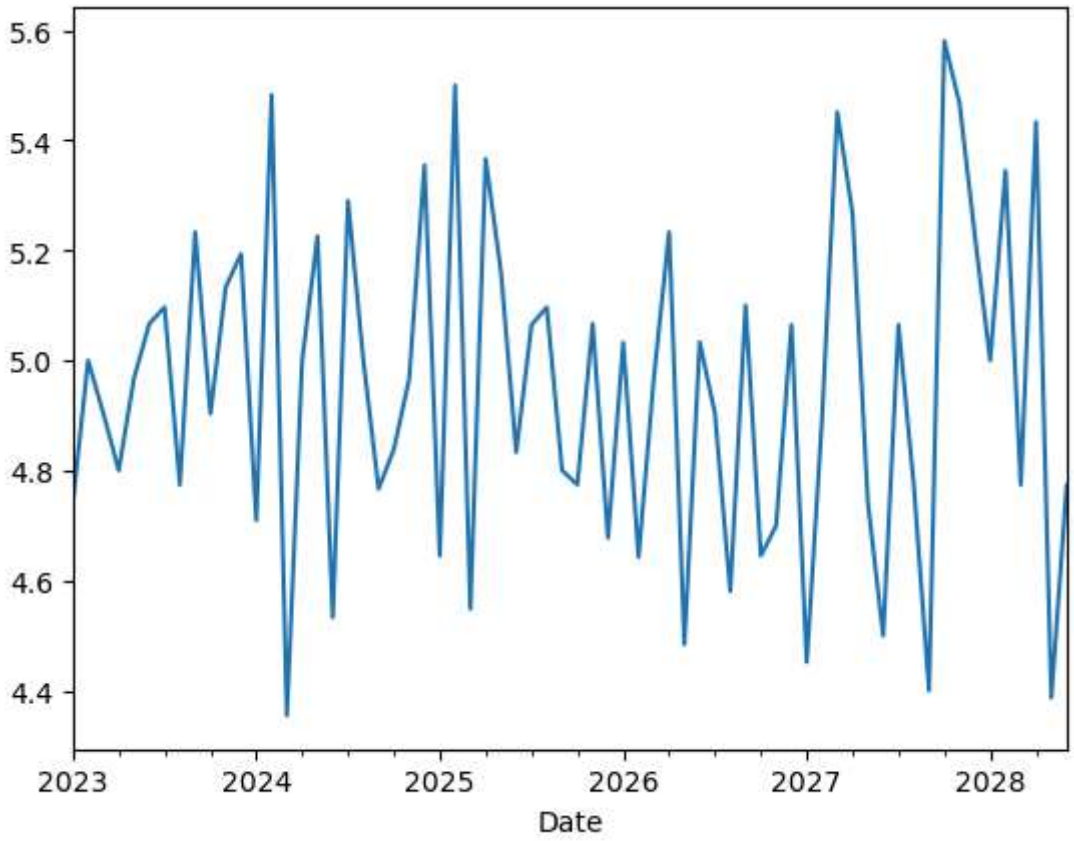
In [49]:
```
plt.figure(figsize=(12, 8))
plt.subplot(411)
plt.plot(df['Page Views'], label='Original')
plt.legend(loc='best')
plt.title('Page Views')

plt.subplot(412)
plt.plot(trend_page_views, label='Trend')
plt.legend(loc='best')
plt.title('Trend Component')

plt.subplot(413)
plt.plot(seasonal_page_views, label='Seasonal')
plt.legend(loc='best')
plt.title('Seasonal Component')
```

```
plt.subplot(414)
plt.plot(residual_page_views, label='Residual')
plt.legend(loc='best')
plt.title('Residual Component')

plt.tight_layout()
plt.show()

# ... similar for other attributes
```



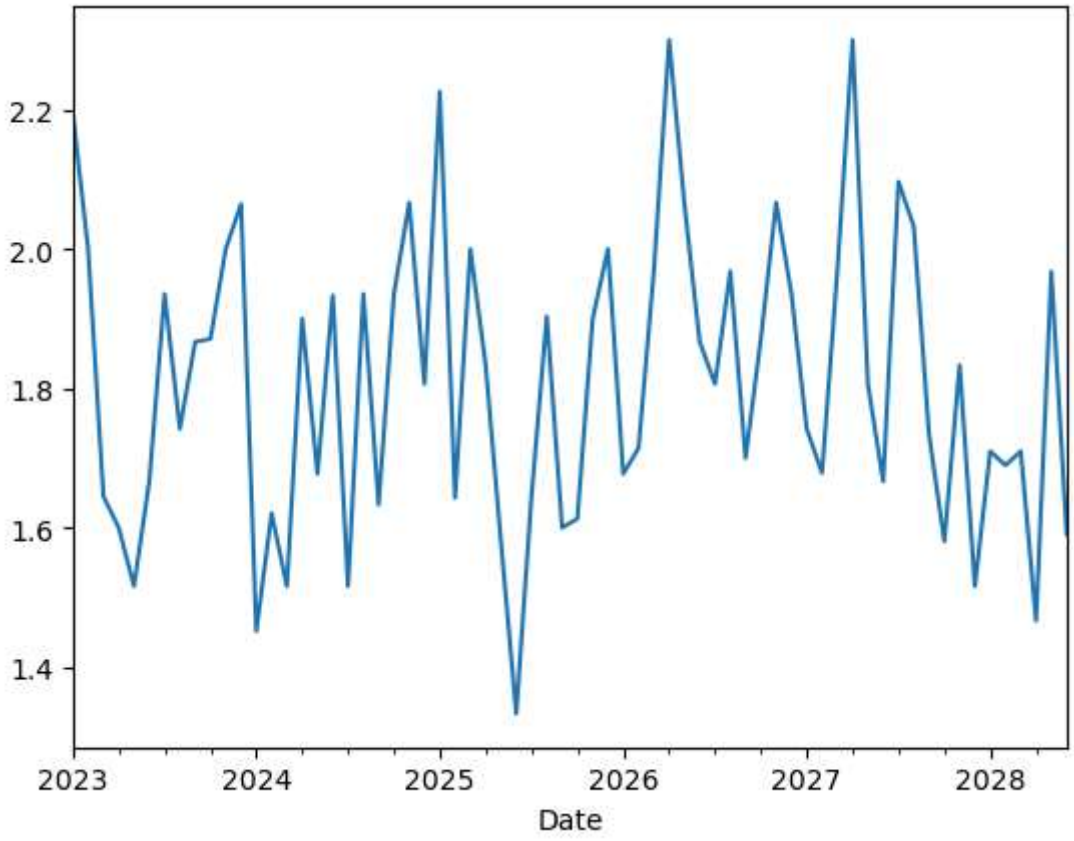
In [55]:
```
df["Page Views"].resample('M').mean().plot()
```

Out[55]:
```
<Axes: xlabel='Date'>
```



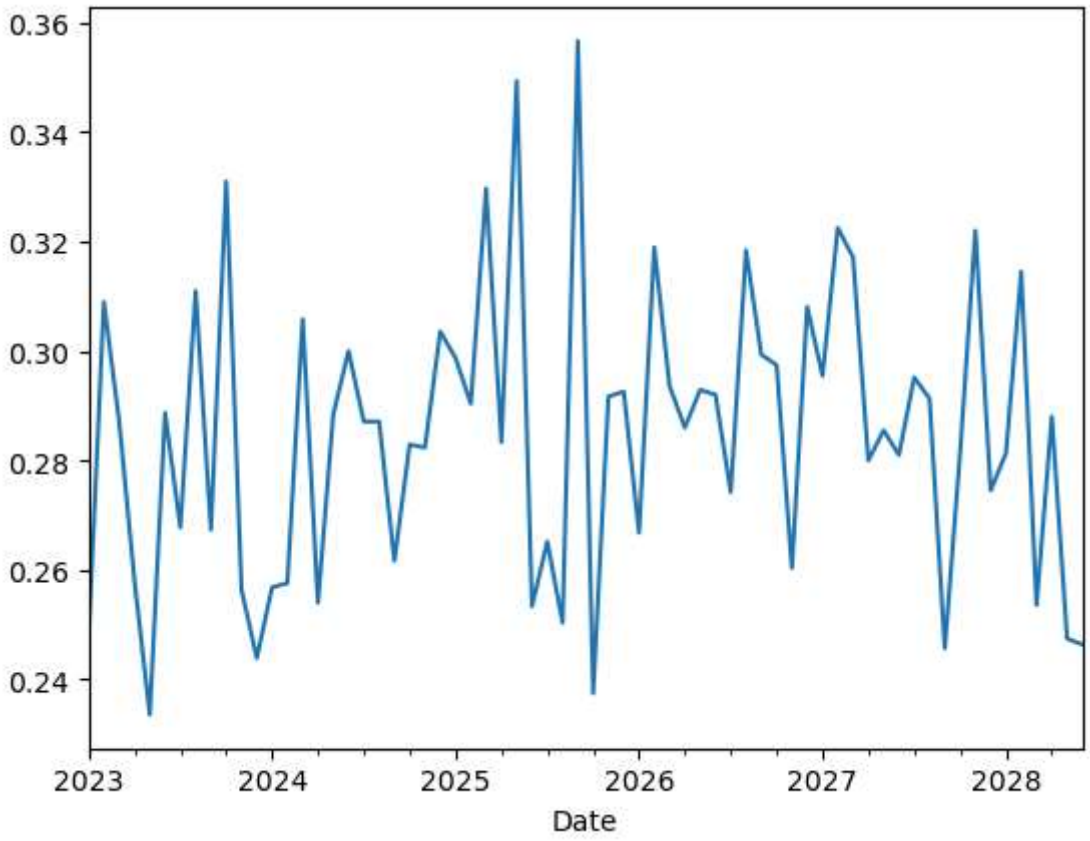
In [57]:
```
df["Traffic Source"].resample('M').mean().plot()
```

Out[57]:
```
<Axes: xlabel='Date'>
```



In [58]:
```
df["Bounce Rate"].resample('M').mean().plot()
```

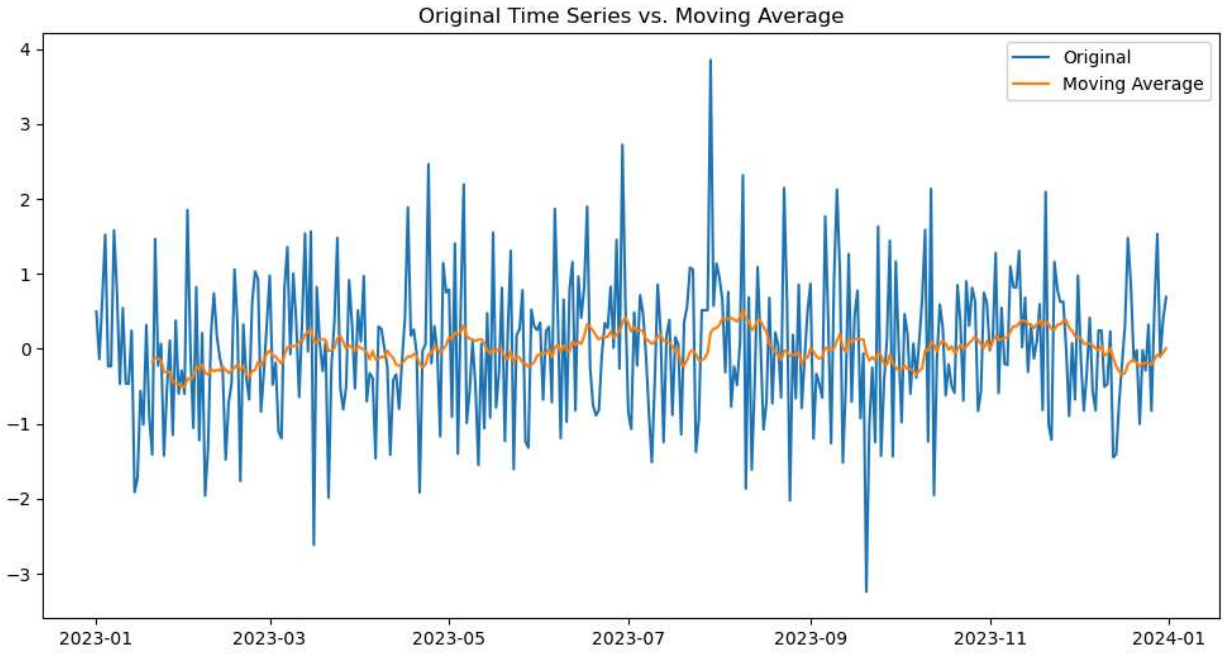
Out[58]:
```
<Axes: xlabel='Date'>
```



In [ ]:
```

```

In [ ]:
```

```

In [63]:
```
window_size = 20
moving_average = df['Session Duration'].rolling(window=window_size).mean()
plt.figure(figsize=(12, 6))
plt.plot(df['Session Duration'], label='Original')
plt.plot(moving_average, label='Moving Average')
plt.legend()
plt.title('Original Time Series vs. Moving Average')
plt.show()
```


Original Time Series vs. Moving Average
Original
Moving Average
4
3
2
1
0
−1
−2
−3
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01