In [4]:

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller

sns.set(style="whitegrid")
```

In [6]:

```
# Load dataset
df = pd.read_csv(r"C:\Users\dhars\Downloads\retail_sales.csv")

# Convert date column to datetime
df['date'] = pd.to_datetime(df['date'])

# Set date as index
df.set_index('date', inplace=True)

# Sort by date
df = df.sort_index()

df.head()
```

Out[6]:

	sales
date	
2022-01-01	188.47
2022-01-02	183.53
2022-01-03	211.68
2022-01-04	241.16
2022-01-05	219.39

In [7]:

```
df.info()
df.describe()
df.isnull().sum()
```

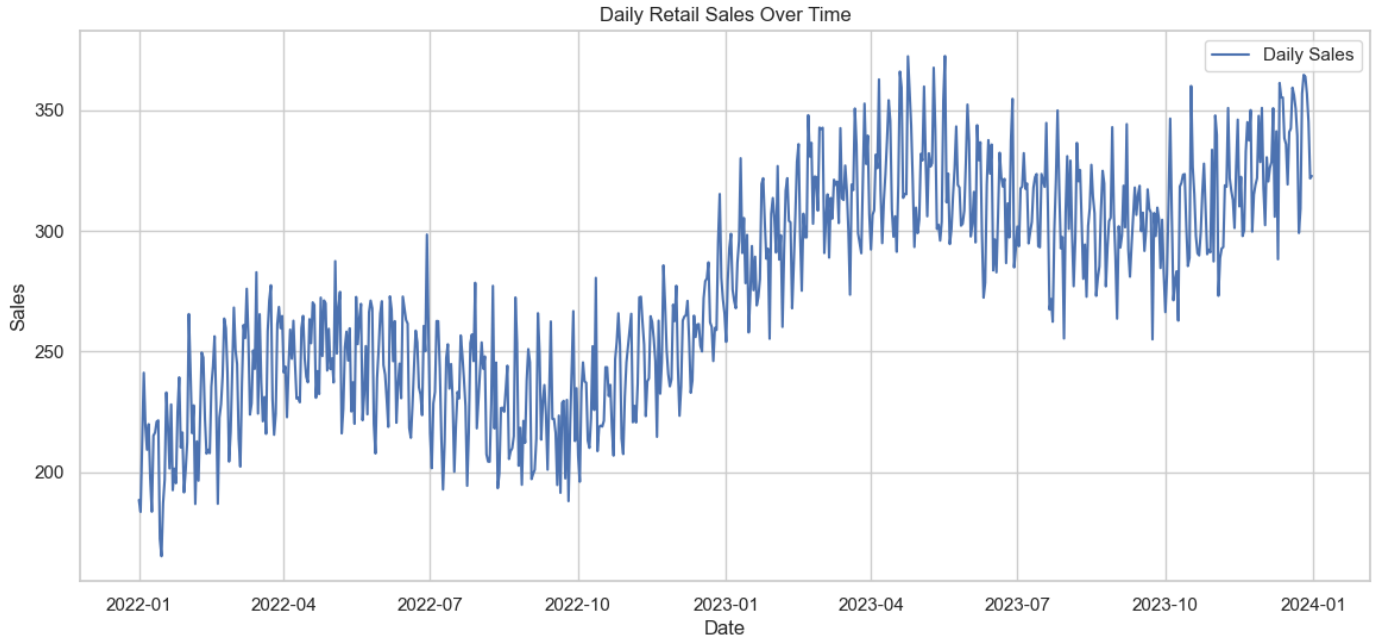
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 730 entries, 2022-01-01 to 2023-12-31
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   sales   730 non-null    float64
dtypes: float64(1)
memory usage: 11.4 KB
```

Out[7]:

```
sales    0
dtype: int64
```

In [8]:

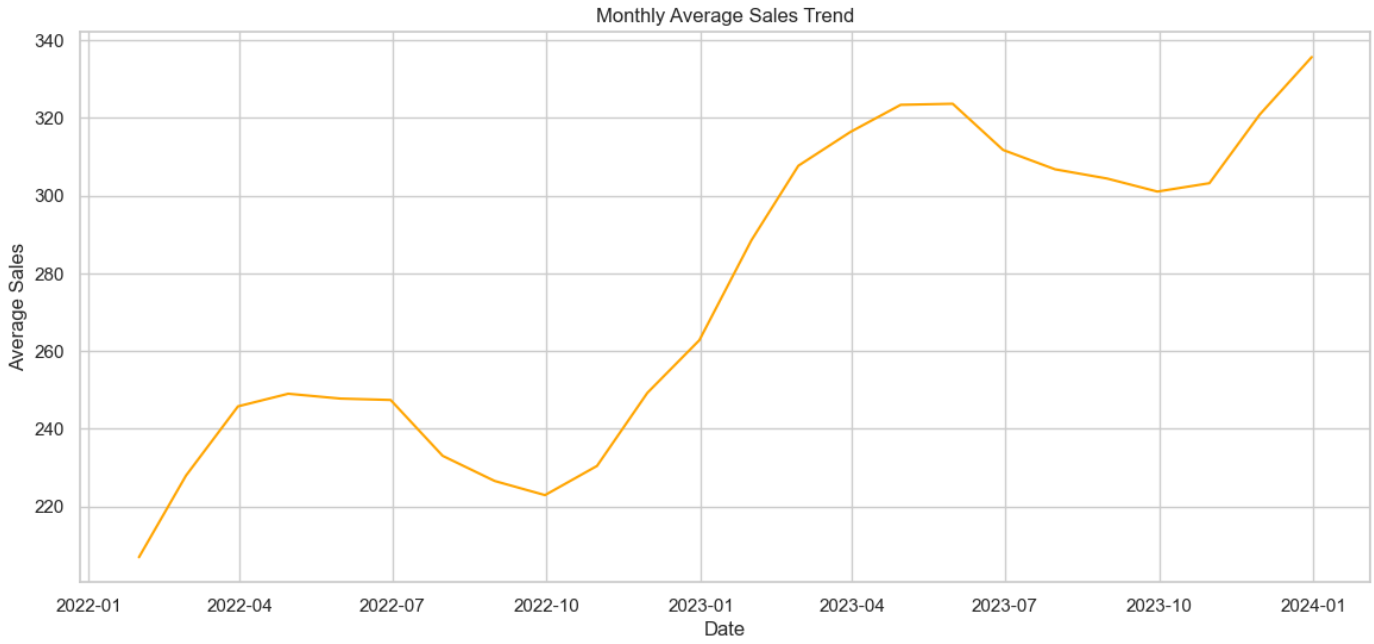
```
plt.figure(figsize=(14,6))
plt.plot(df.index, df['sales'], label='Daily Sales')
plt.title("Daily Retail Sales Over Time")
plt.xlabel("Date")
plt.ylabel("Sales")
plt.legend()
plt.show()
```



In [9]:

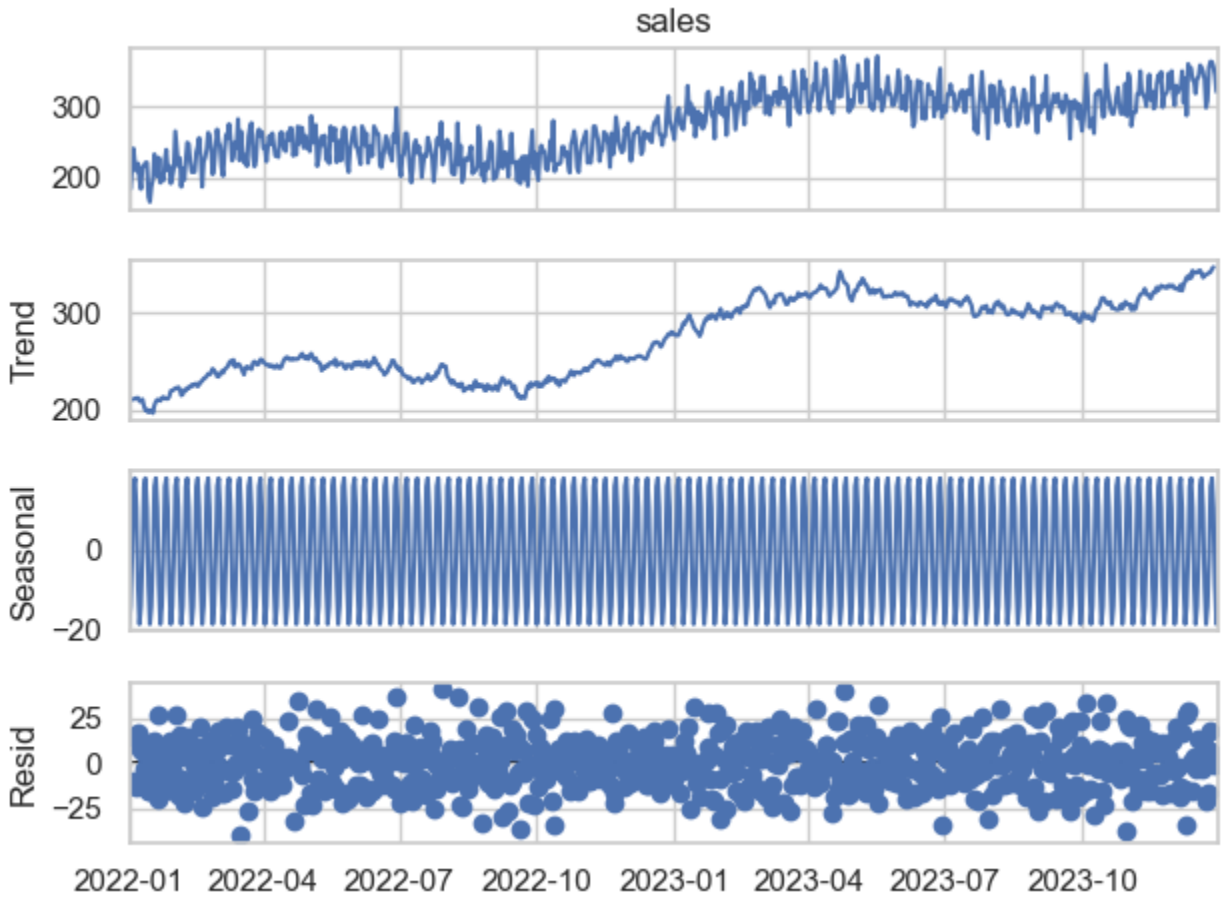
```
monthly_sales = df['sales'].resample('M').mean()

plt.figure(figsize=(14,6))
plt.plot(monthly_sales, color='orange')
plt.title("Monthly Average Sales Trend")
plt.xlabel("Date")
plt.ylabel("Average Sales")
plt.show()
```

```
C:\Users\dhars\AppData\Local\Temp\ipykernel_12220\3608894250.py:1: FutureWarning: 'M' is deprecated and will be removed in a future version, please use 'ME' instead.
  monthly_sales = df['sales'].resample('M').mean()
```



In [10]:

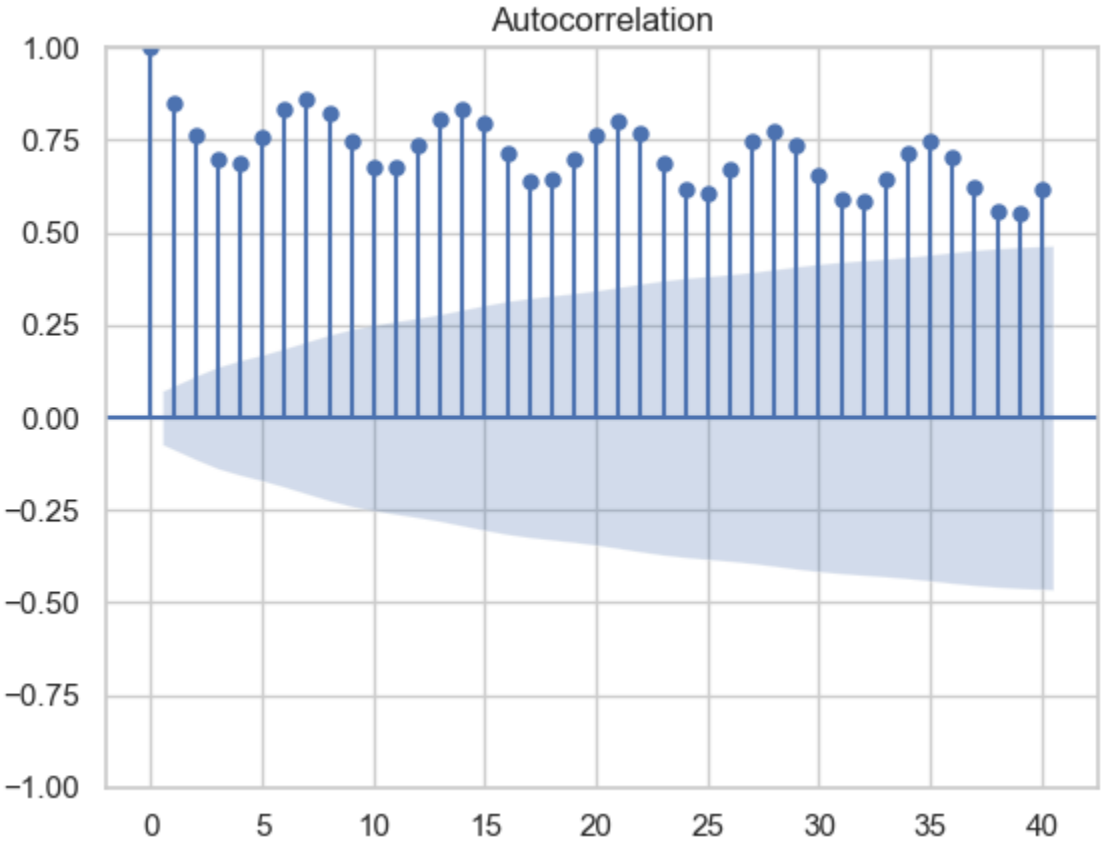
```
decomposition = seasonal_decompose(df['sales'], model='additive', period=7)

decomposition.plot()
plt.show()
```



In [11]:

```
plt.figure(figsize=(12,4))
plot_acf(df['sales'], lags=40)
plt.show()
```

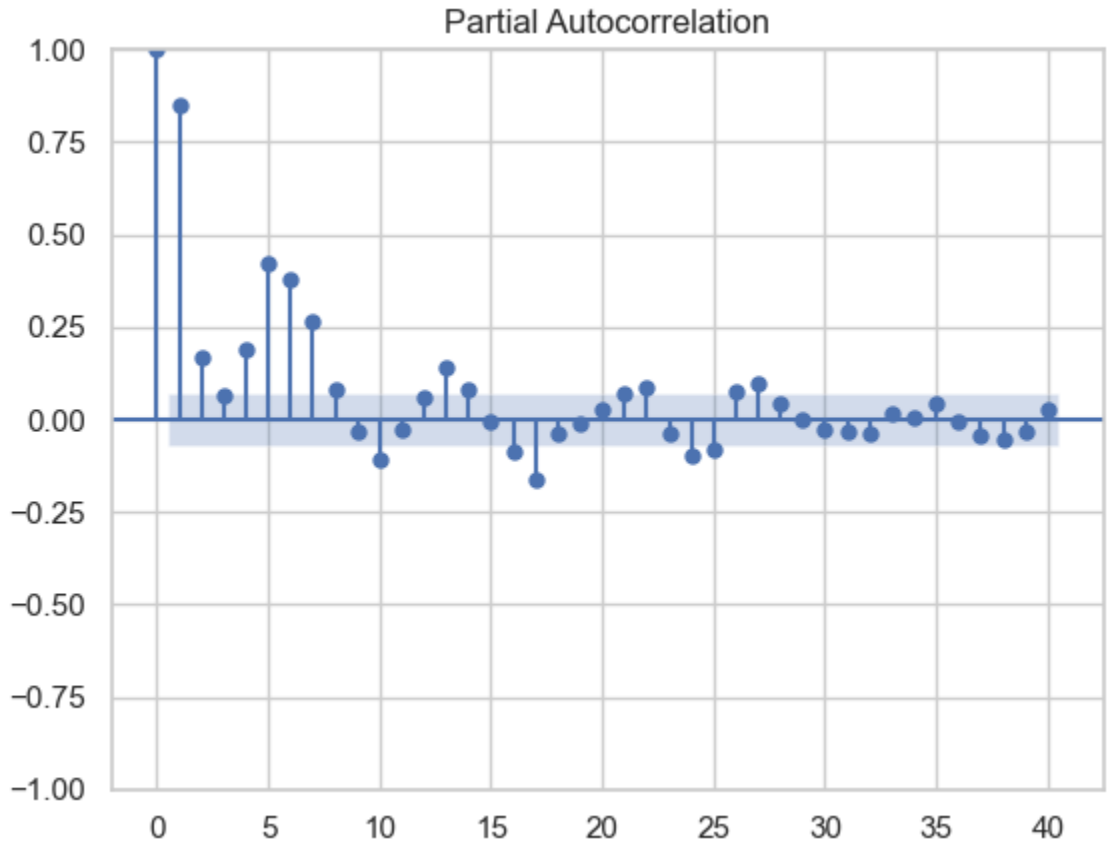
```
<Figure size 1200x400 with 0 Axes>
```



In [12]:

```
plt.figure(figsize=(12,4))
plot_pacf(df['sales'], lags=40, method='ywm')
plt.show()
```

```
<Figure size 1200x400 with 0 Axes>
```



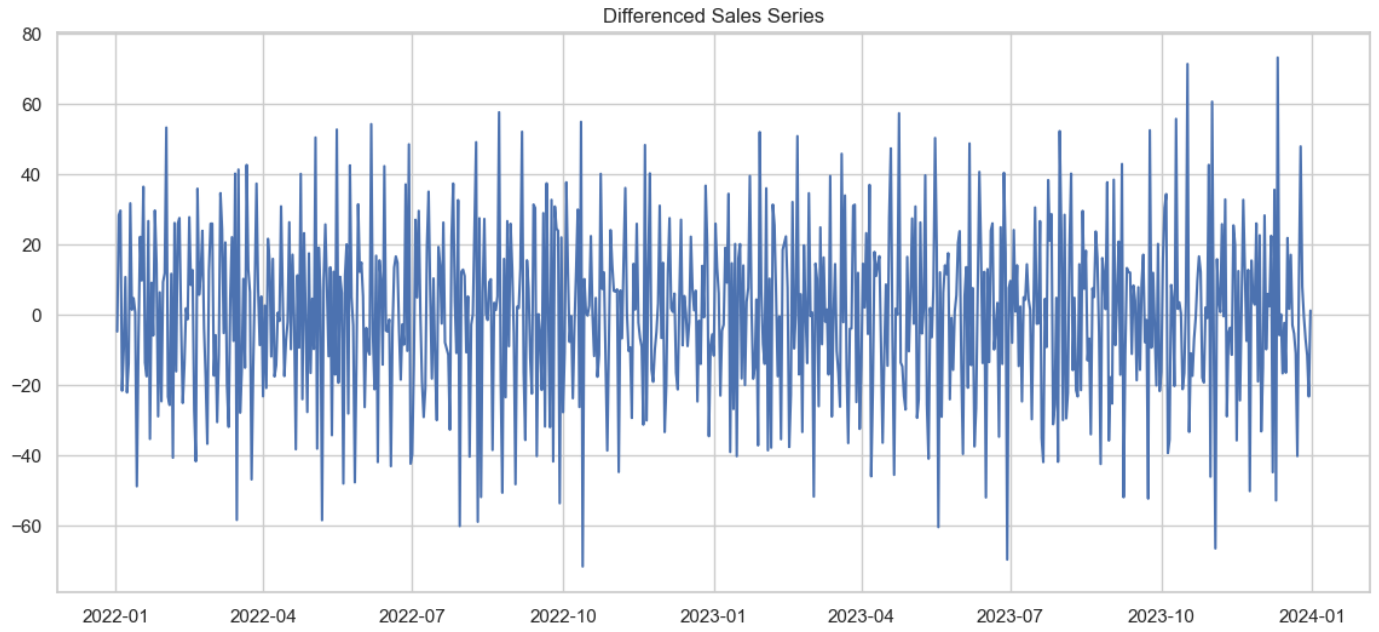
In [13]:

```
def adf_test(series):
    result = adfuller(series)

    print("ADF Statistic:", result[0])
    print("p-value:", result[1])

    if result[1] <= 0.05:
        print("Result: Data is stationary")
    else:
        print("Result: Data is NOT stationary")

adf_test(df['sales'])
```

```
ADF Statistic: -0.9357994761502129
p-value: 0.7759547036582954
Result: Data is NOT stationary
```

In [14]:

```
df['sales_diff'] = df['sales'].diff().dropna()

plt.figure(figsize=(14,6))
plt.plot(df['sales_diff'])
plt.title("Differenced Sales Series")
plt.show()
```



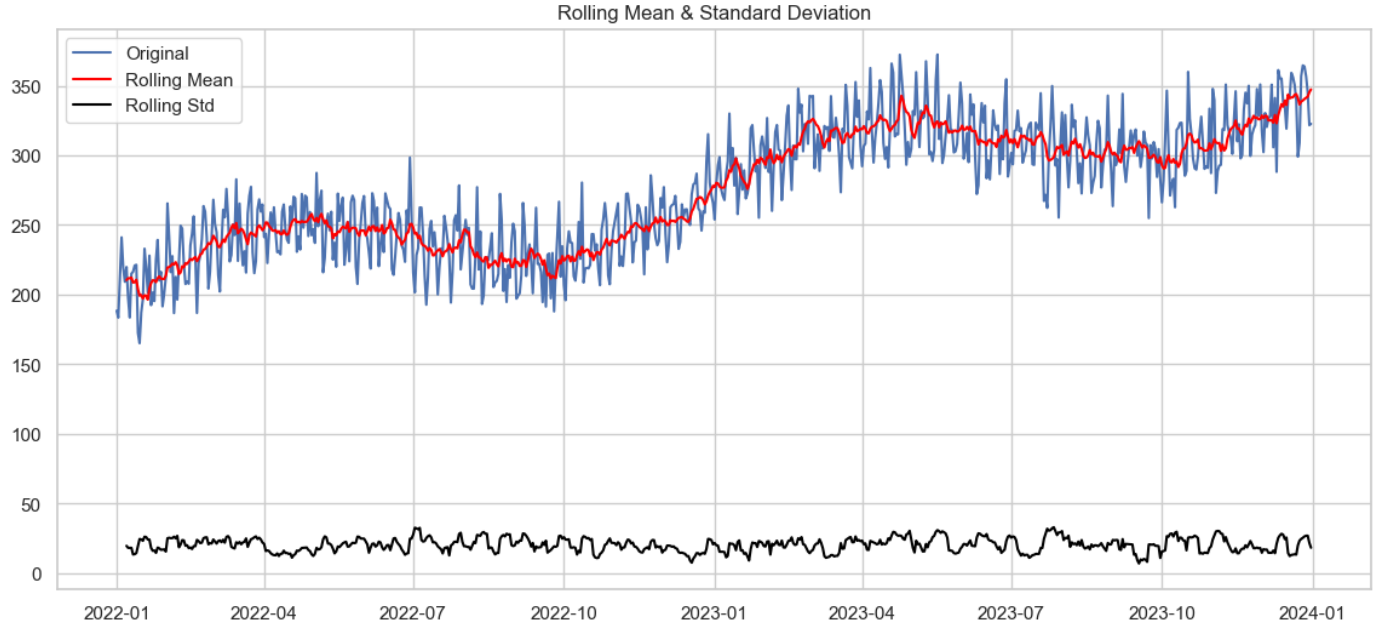
In [15]:

```
adf_test(df['sales_diff'].dropna())
```

```
ADF Statistic: -8.515080359959512
p-value: 1.1398326711003247e-13
Result: Data is stationary
```

In [16]:

```
rolling_mean = df['sales'].rolling(window=7).mean()
rolling_std = df['sales'].rolling(window=7).std()

plt.figure(figsize=(14,6))
plt.plot(df['sales'], label='Original')
plt.plot(rolling_mean, label='Rolling Mean', color='red')
plt.plot(rolling_std, label='Rolling Std', color='black')
plt.legend()
plt.title("Rolling Mean & Standard Deviation")
plt.show()
```



In [ ]: