In [304…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly.express as px
```

In [305…
```
df = pd.read_csv(r'D:\desctop\retail_store_inventory.CSV')
```

In [306…
```
df.head()
```

Out[306…

	Date	Store ID	Product ID	Category	Region	Inventory Level	Units Sold	Units Ordered	Demand Forecast	Price	Discount
0	2022-01-01	S001	P0001	Groceries	North	231	127	55	135.47	33.50	20
1	2022-01-01	S001	P0002	Toys	South	204	150	66	144.04	63.01	20
2	2022-01-01	S001	P0003	Toys	West	102	65	51	74.02	27.99	10
3	2022-01-01	S001	P0004	Toys	North	469	61	164	62.18	32.72	10
4	2022-01-01	S001	P0005	Electronics	East	166	14	135	9.26	73.64	0

In [307…
```
df['Date'] = pd.to_datetime(df['Date'])
```

In [308…
```
for col in df.columns:
    print(col)
```

```
Date
Store ID
Product ID
Category
Region
Inventory Level
Units Sold
Units Ordered
Demand Forecast
Price
Discount
Weather Condition
Holiday/Promotion
Competitor Pricing
Seasonality
```

In [309…
```
daily_sales = df.groupby(df['Date'])['Units Sold'].sum().reset_index()
fig1 = px.line(daily_sales, x='Date', y='Units Sold', title='Sales Over Time', label
fig1.update_traces(line=dict(color='royalblue'))
fig1.show()
```

In [310…

```
df_sampled = df.sample(n=10000)
fig2 = px.histogram(df_sampled, x='Price', nbins=30, marginal='rug', title='Price Di
fig2.update_traces(marker=dict(color='skyblue'))
fig2.show()
```

In [311…

```
weather_sales = df.groupby('Weather Condition')['Units Sold'].sum().reset_index()
fig5 = px.pie(weather_sales, names='Weather Condition', values='Units Sold', title='
fig5.show()
```

In [312…

```
fig6 = px.scatter(df, x='Demand Forecast', y='Price', title='Demand Forecast vs Pric
fig6.update_traces(marker=dict(color='orange'))
fig6.show()
```

In [313…

```
fig7 = px.box(df, x='Discount', y='Units Sold', title='Discount vs Sales', labels={'
fig7.update_traces(marker=dict(color='purple'))
fig7.show()
```

In [314…

```
region_sales = df.groupby('Region')['Units Sold'].sum().reset_index()
fig1 = px.bar(region_sales, x='Region', y='Units Sold', title='Total Sales by Region
fig1.update_traces(marker=dict(color='lightseagreen'))
fig1.show()
```

In [315…

```
daily_forecast = df.groupby(df['Date'])['Demand Forecast'].mean().reset_index()
fig2 = px.line(daily_forecast, x='Date', y='Demand Forecast', title='Average Daily D
fig2.update_traces(line=dict(color='mediumvioletred'))
fig2.show()
```

In [316…

```
price_by_cat = df.groupby('Category')['Price'].mean().reset_index()
fig3 = px.bar(price_by_cat, x='Category', y='Price', title='Average Price per Catego
fig3.update_traces(marker=dict(color='royalblue'))
fig3.show()
```

In [317…

```
fig4 = px.histogram(df, x='Inventory Level', nbins=30, title='Inventory Level Distri
fig4.update_traces(marker=dict(color='skyblue'))
fig4.show()
```

In [318…

```
fig5 = px.histogram(df, x='Units Ordered', nbins=30, title='Units Ordered Distributi
fig5.update_traces(marker=dict(color='lightcoral'))
fig5.show()
```

In [319…

```
fig6 = px.histogram(df, x='Competitor Pricing', nbins=30, title='Competitor Pricing
fig6.update_traces(marker=dict(color='orange'))
fig6.show()
```

In [320…

```
promo_counts = df['Holiday/Promotion'].value_counts().sort_index().reset_index()
promo_counts.columns = ['Holiday/Promotion', 'count']

fig7 = px.bar(
    promo_counts,
    x='Holiday/Promotion',
    y='count',
    title='Operations by Holiday/Promotion',
    labels={'Holiday/Promotion': 'Holiday/Promotion (0=No, 1=Yes)', 'count': 'Number
)
fig7.update_traces(marker=dict(color='teal'))
fig7.show()
```

In [321…

```
daily = df.groupby('Date')[['Units Sold', 'Units Ordered']].sum().reset_index()
fig8 = px.line(daily, x='Date', y=['Units Sold', 'Units Ordered'], title='Sales vs U
fig8.update_traces(line=dict(color='green'))
fig8.show()
```

In [322…

```
fig10 = px.box(df, x='Category', y='Units Sold', title='Units Sold by Category', lab
fig10.update_traces(marker=dict(color='yellowgreen'))
fig10.show()
```

In [323…

```
daily_discount = df.groupby('Date')['Discount'].mean().reset_index()
fig13 = px.line(daily_discount, x='Date', y='Discount', title='Average Discount Over
fig13.update_traces(line=dict(color='indianred'))
fig13.show()
```

In [324…

```
inv_by_store = df.groupby('Store ID')['Inventory Level'].mean().reset_index()
fig14 = px.bar(inv_by_store, x='Store ID', y='Inventory Level', title='Average Inven
fig14.update_traces(marker=dict(color='mediumpurple'))
fig14.show()
```

In [325…

```
fig15 = px.histogram(df, x='Demand Forecast', nbins=30, title='Demand Forecast Distr
fig15.update_traces(marker=dict(color='gold'))
fig15.show()
```

In [326…

```
fig16 = px.scatter(df, x='Demand Forecast', y='Units Sold', title='Units Sold vs Dem
fig16.update_traces(marker=dict(color='darkviolet', opacity=0.4))
fig16.show()
```

In [327…

```
sales_by_season = df.groupby('Seasonality')['Units Sold'].mean().reset_index()
fig17 = px.bar(sales_by_season, x='Seasonality', y='Units Sold', title='Average Sale
fig17.update_traces(marker=dict(color='darkorange'))
fig17.show()
```

In [328…

```
cat_counts = df['Category'].value_counts().reset_index()
cat_counts.columns = ['Category', 'count']
fig18 = px.pie(cat_counts, names='Category', values='count', title='Category Proport
fig18.show()
```

In [329…

```
fig21 = px.histogram(df, x='Discount', nbins=5, title='Discount Distribution', label
fig21.update_traces(marker=dict(color='darkturquoise'))
fig21.show()
```

In [330…

```
sales_by_product = df.groupby('Product ID')['Units Sold'].sum().reset_index()
fig22 = px.bar(sales_by_product, x='Product ID', y='Units Sold', title='Units Sold b
fig22.update_traces(marker=dict(color='fuchsia'))
fig22.show()
```

In [331…

```
price_promo = df.groupby('Holiday/Promotion')['Price'].mean().reset_index()
fig24 = px.bar(price_promo, x='Holiday/Promotion', y='Price', title='Average Price i
fig24.update_traces(marker=dict(color='peru'))
fig24.show()
```

In [332…

```
daily = df.groupby('Date')[['Units Sold','Units Ordered']].sum().reset_index()
fig = px.line(daily, x='Date', y=['Units Sold', 'Units Ordered'],
title='Units Sold vs Units Ordered per Day')
fig.show()
```

In [333…

```
fig = px.box(df, x='Seasonality', y='Price',
title='Price by Season')
fig.show()
```

In [334…

```
pivot_df = df.pivot_table(index='Category', columns='Region', values='Units Sold', a
pivot_df = pivot_df.melt(id_vars='Category', var_name='Region', value_name='Units So
fig = px.bar(pivot_df, x='Category', y='Units Sold', color='Region',
title='Category Sales by Region', barmode='stack')
fig.show()
```

In [335…

```
fig = px.bar(df.groupby('Weather Condition')['Units Sold'].mean().reset_index(),
x='Weather Condition', y='Units Sold',
title='Average Sales by Weather Condition')
fig.show()
```

In [336…

```
fig = px.scatter(df, x='Inventory Level', y='Demand Forecast',
title='Inventory Level vs Demand Forecast', opacity=0.4)
fig.show()
```

In [337…

```
df['Month'] = df['Date'].dt.to_period('M')
monthly = df.groupby('Month')[['Units Ordered','Demand Forecast']].sum().reset_index
monthly['Month'] = monthly['Month'].astype(str)
```

```
fig = px.line(monthly, x='Month', y=['Units Ordered', 'Demand Forecast'],
title='Monthly Demand vs Orders')
fig.show()
```

In [338…

```
fig3 = px.box(df, x='Discount', y='Units Sold',
title=' Discount Impact on Sales')
fig3.show()
```

In [339…

```
fig4 = px.sunburst(df, path=['Region', 'Category'], values='Units Sold',
title=' Sales by Region and Product Category')
fig4.show()
```

In [340…

```
fig5 = px.scatter(df, x='Competitor Pricing', y='Price', color='Category',
title=' Product Price vs. Competitor Price by Category',
hover_data=['Units Sold', 'Discount'])
fig5.show()
```

In [341…

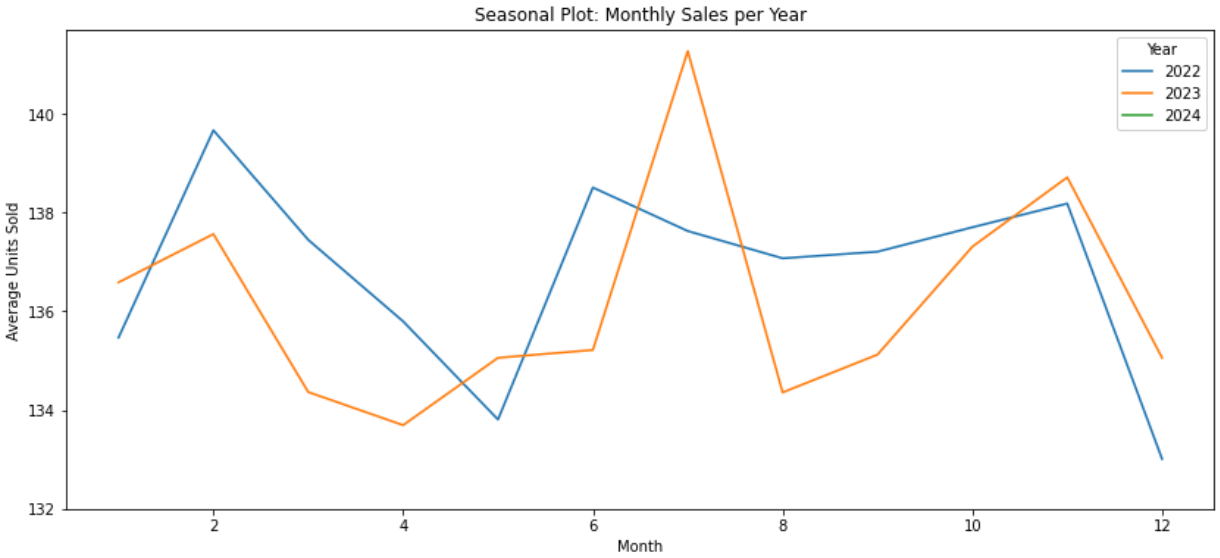
```
category_trend = df.groupby(['Date', 'Category'])['Units Sold'].sum().reset_index()
fig = px.line(category_trend, x='Date', y='Units Sold', color='Category',
title=' Sales Trend by Product Category Over Time')
fig.show()
```

In [342…

```
fig = px.scatter(df, x='Inventory Level', y='Units Sold', color='Category',
title=' Inventory Level vs. Units Sold',
hover_data=['Store ID', 'Product ID'])
fig.show()
```

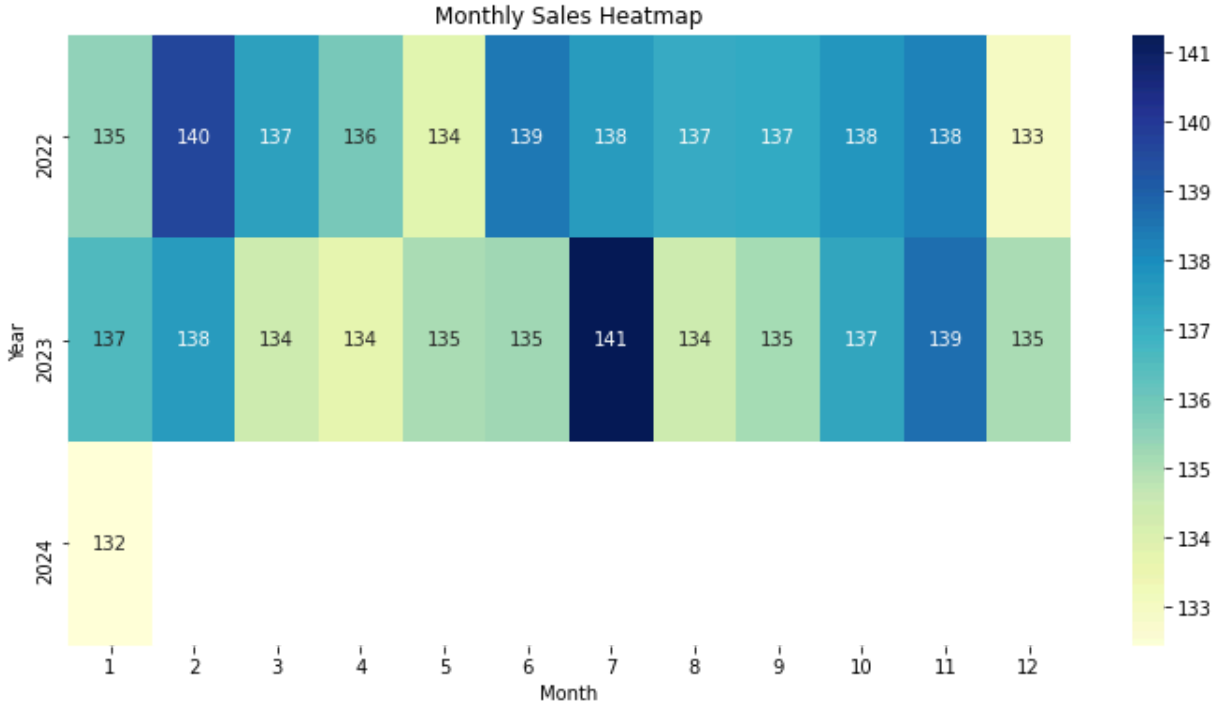
In [343…

```
df['Month'] = df['Date'].dt.month
df['Year'] = df['Date'].dt.year
monthly_avg = df.groupby(['Year', 'Month'])['Units Sold'].mean().reset_index()

plt.figure(figsize=(14,6))
sns.lineplot(data=monthly_avg, x='Month', y='Units Sold', hue='Year', palette='tab10
plt.title('Seasonal Plot: Monthly Sales per Year')
plt.xlabel('Month')
plt.ylabel('Average Units Sold')
plt.show()
```



In [344…

```
pivot = monthly_avg.pivot(index='Year', columns='Month', values='Units Sold')

plt.figure(figsize=(12,6))
sns.heatmap(pivot, annot=True, fmt=".0f", cmap="YlGnBu")
plt.title('Monthly Sales Heatmap')
plt.xlabel('Month')
plt.ylabel('Year')
plt.show()
```



In [ ]:

In [345…

```
df['Date'] = pd.to_datetime(df['Date'])

df.set_index('Date', inplace=True)

monthly_sales = df['Units Sold'].resample('M').sum()

fig = px.line(monthly_sales, title='Monthly Sales Trend')
```

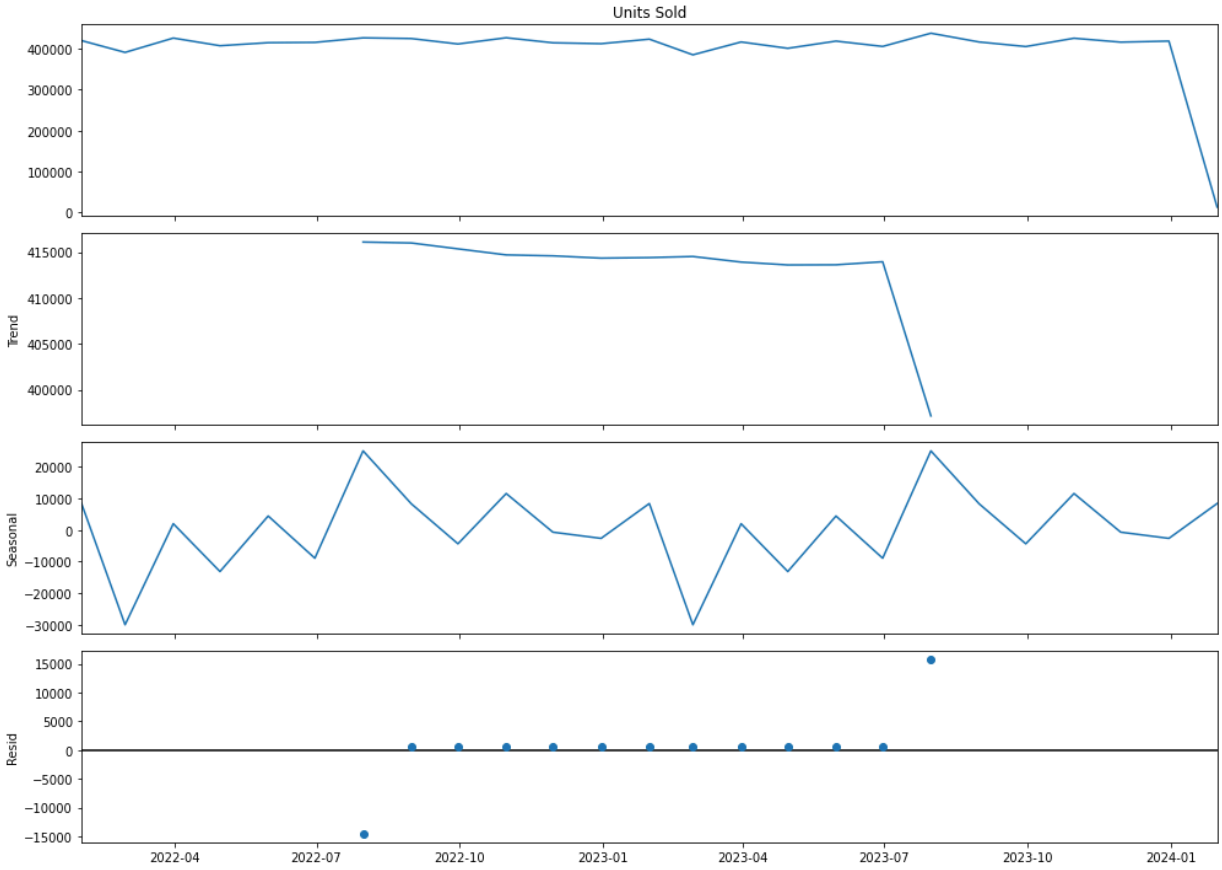
```
fig.update_layout(xaxis_title='Date', yaxis_title='Units Sold')
fig.show()
```

In [346…

```
df['Month'] = df.index.month
monthly_avg = df.groupby('Month')['Units Sold'].mean().reset_index()
fig = px.bar(monthly_avg, x='Month', y='Units Sold', title='Average Units Sold by Mo
fig.show()
```

In [347…

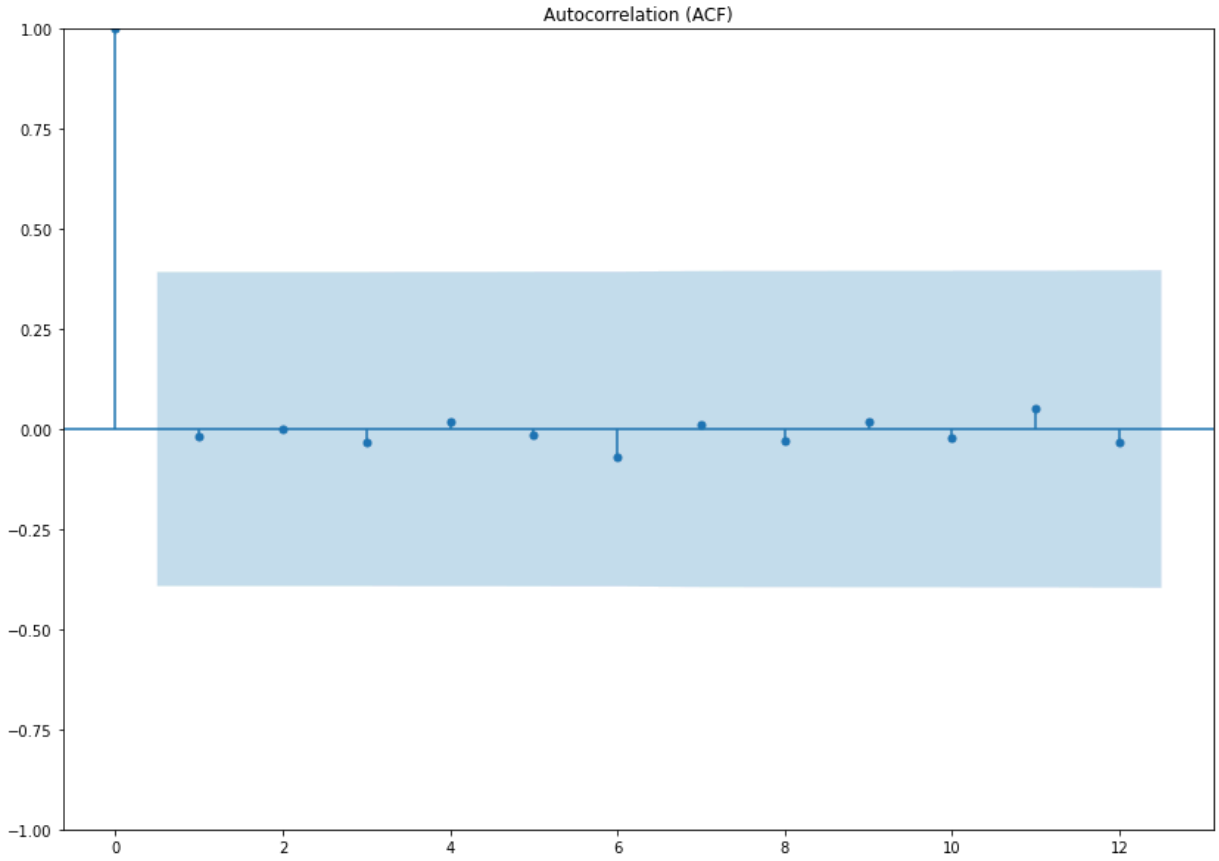
```
from statsmodels.tsa.seasonal import seasonal_decompose

result = seasonal_decompose(monthly_sales, model='additive')
plt.rcParams.update({'figure.figsize': (14, 10)})
result.plot()
plt.tight_layout()
plt.show()
```



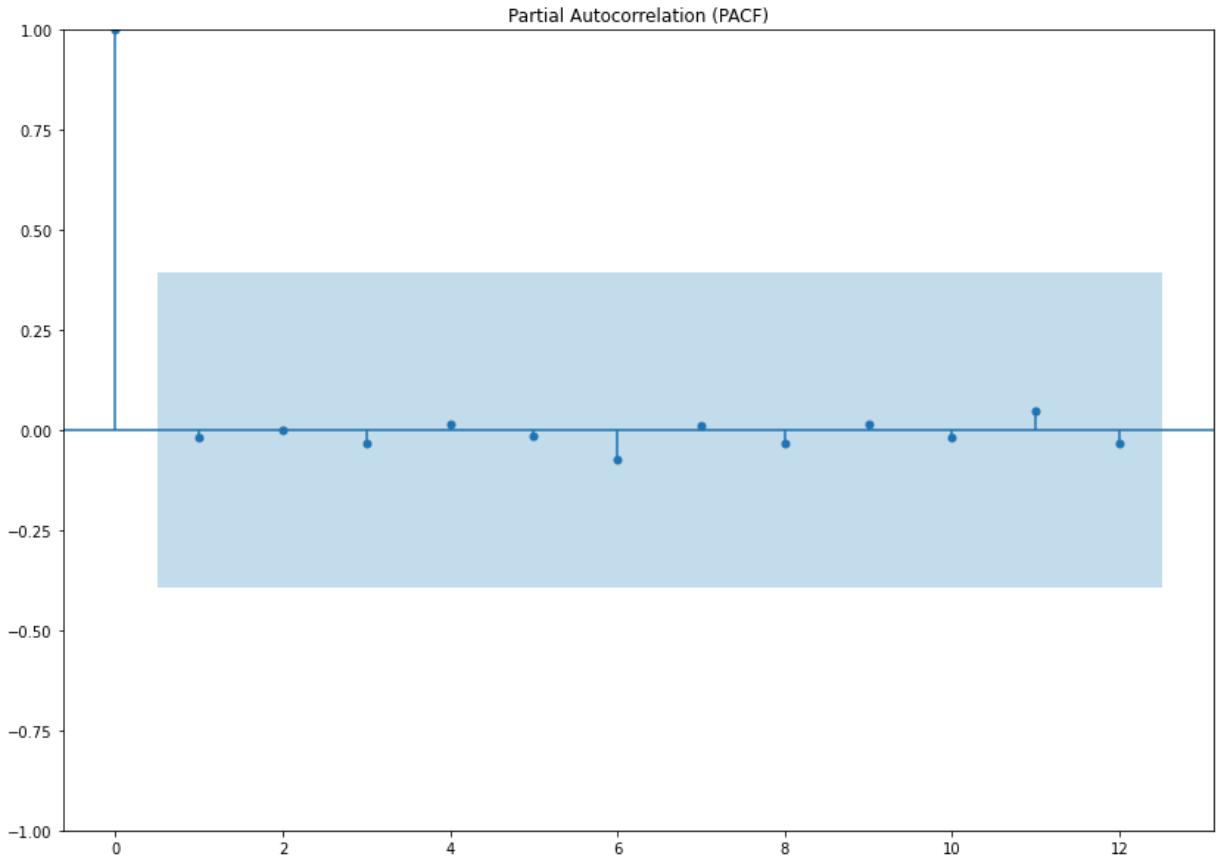
In [348…

```
n_obs = len(monthly_sales)
max_lags = n_obs // 2
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

plt.figure(figsize=(14,5))
plot_acf(monthly_sales, lags=max_lags)
plt.title("Autocorrelation (ACF)")
plt.show()

plt.figure(figsize=(14,5))
plot_pacf(monthly_sales, lags=max_lags)
plt.title("Partial Autocorrelation (PACF)")
plt.show()
```

```
<Figure size 1008x360 with 0 Axes>
```

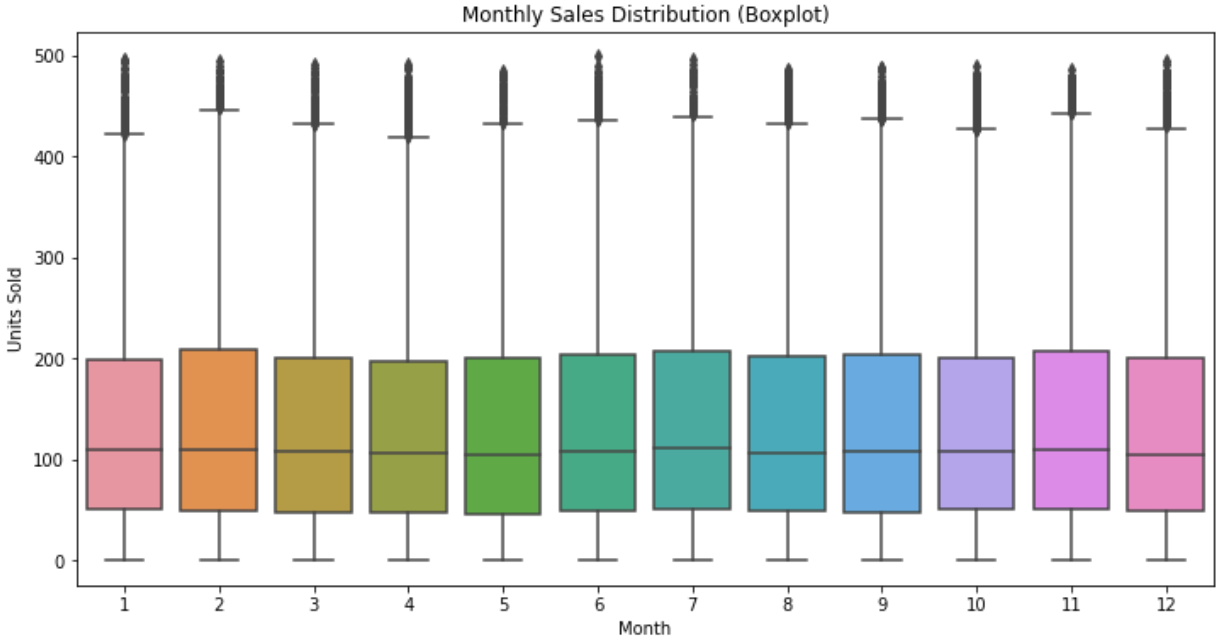



```
<Figure size 1008x360 with 0 Axes>
```



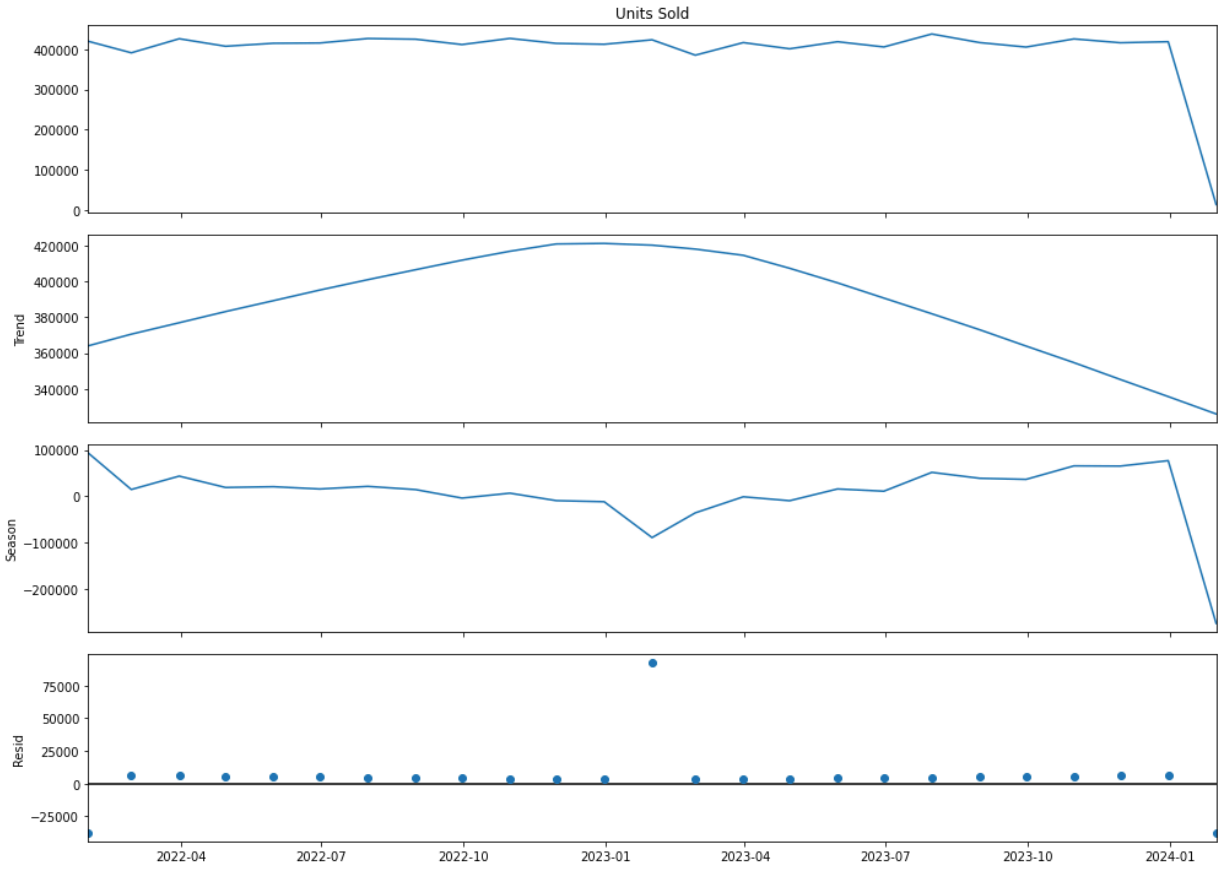
In [349…

```
plt.figure(figsize=(12,6))
sns.boxplot(x='Month', y='Units Sold', data=df)
plt.title('Monthly Sales Distribution (Boxplot)')
plt.xlabel('Month')
plt.ylabel('Units Sold')
plt.show()
```



In [350…

```
from statsmodels.tsa.seasonal import STL

stl = STL(monthly_sales, seasonal=13)
res = stl.fit()
res.plot()
plt.show()
```



In [ ]: