Task 7: Time Series Breakdown of Retail Sales

This notebook performs time series analysis on Walmart's departmental sales data. It includes data aggregation to monthly sales, visualization of trends using rolling averages, and sales forecasting using Simple Exponential Smoothing. The aim is to uncover patterns, seasonal trends, and provide a basic predictive model for future sales behavior.

In [42]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

1 Load and Explore the Data

In [2]:
```
df=pd.read_csv("train.csv")
df.head()
```

Out[2]:

	Store	Dept	Date	Weekly_Sales	IsHoliday
0	1	1	2010-02-05	24924.50	False
1	1	1	2010-02-12	46039.49	True
2	1	1	2010-02-19	41595.55	False
3	1	1	2010-02-26	19403.54	False
4	1	1	2010-03-05	21827.90	False

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 421570 entries, 0 to 421569
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   Store         421570 non-null  int64  
 1   Dept          421570 non-null  int64  
 2   Date          421570 non-null  object 
 3   Weekly_Sales  421570 non-null  float64
 4   IsHoliday     421570 non-null  bool   
dtypes: bool(1), float64(1), int64(2), object(1)
memory usage: 13.3+ MB
```

In [4]:
```
df.isna().sum()
```

Out[4]:
```
Store           0
Dept            0
Date            0
Weekly_Sales    0
IsHoliday       0
dtype: int64
```

In [5]:
```
df.duplicated().sum()
```

Out[5]:
```
0
```

In [7]:
```
df.dtypes
```

Out[7]:
```
Store             int64
Dept              int64
Date             object
Weekly_Sales    float64
IsHoliday          bool
dtype: object
```

2 Convert to Monthly Sales

In [13]:
```
df['Date']=pd.to_datetime(df["Date"])
df['Month'] = df['Date'].dt.to_period('M')
monthly_sales = df.groupby('Month')['Weekly_Sales'].sum().reset_index()
monthly_sales['Month'] = monthly_sales['Month'].dt.to_timestamp()
```

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 421570 entries, 0 to 421569
Data columns (total 7 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Store         421570 non-null  int64
 1   Dept          421570 non-null  int64
 2   Date          421570 non-null  datetime64[ns]
 3   Weekly_Sales  421570 non-null  float64
 4   IsHoliday     421570 non-null  bool
 5   Month         421570 non-null  period[M]
 6   Year          421570 non-null  int32
dtypes: bool(1), datetime64[ns](1), float64(1), int32(1), int64(2), period[M](1)
memory usage: 18.1 MB
```

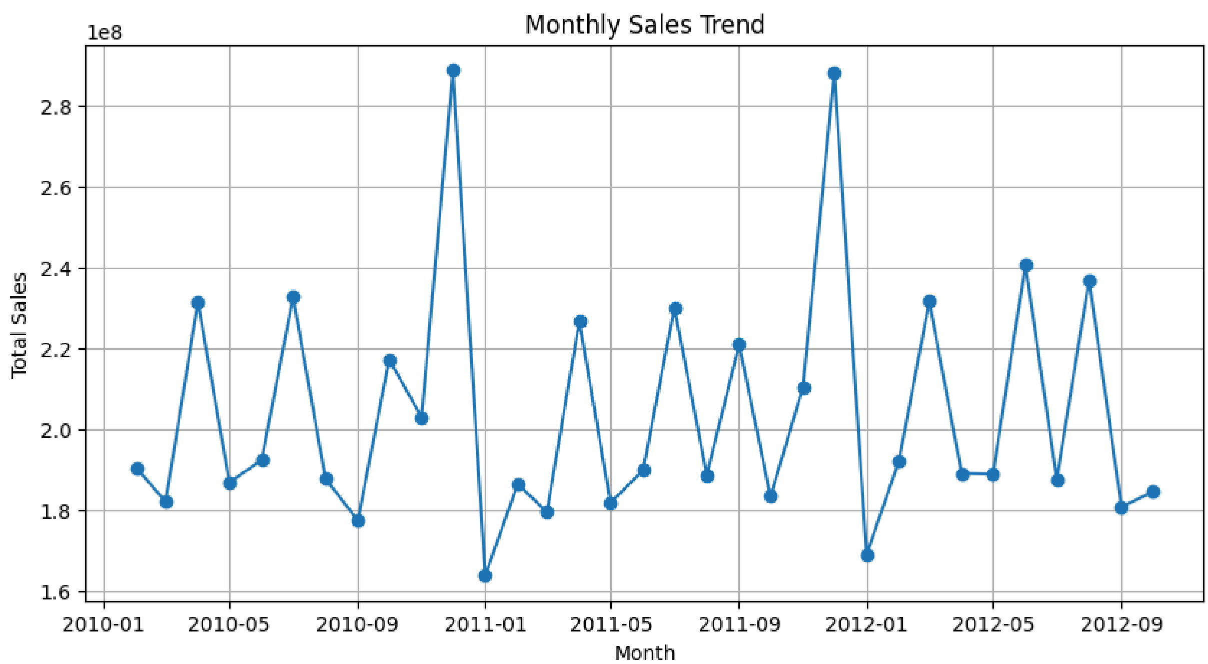
In [15]:
```
df.tail()
```

Out[15]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Month	Year
421565	45	98	2012-09-28	508.37	False	2012-09	2012
421566	45	98	2012-10-05	628.10	False	2012-10	2012
421567	45	98	2012-10-12	1061.02	False	2012-10	2012
421568	45	98	2012-10-19	760.01	False	2012-10	2012
421569	45	98	2012-10-26	1076.80	False	2012-10	2012

4. Plot Trend Over Time

In [16]:
```
plt.figure(figsize=(10, 5))
plt.plot(monthly_sales['Month'], monthly_sales['Weekly_Sales'], marker='o')
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.grid(True)
plt.show()
```

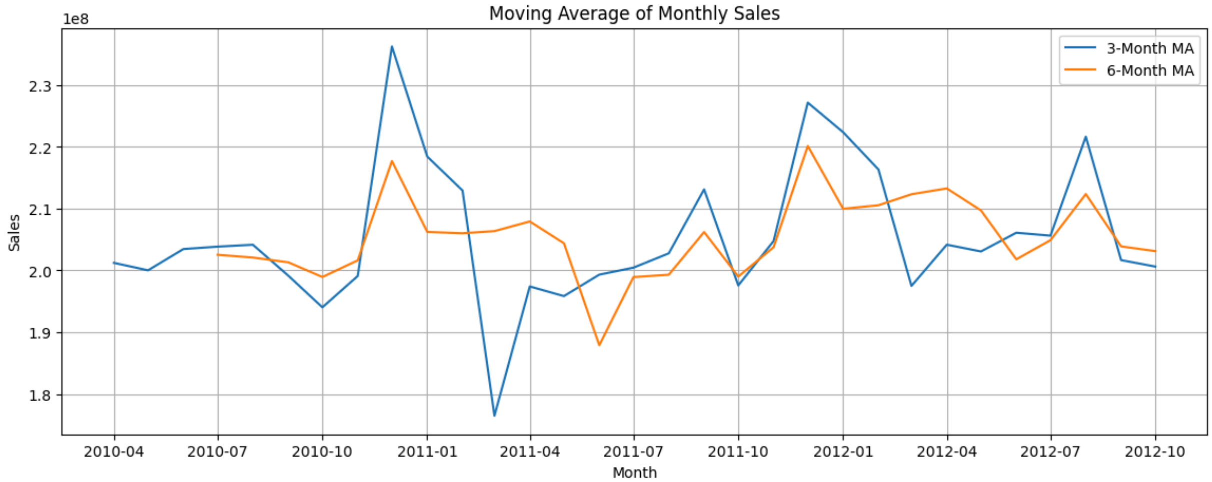


5. Add Moving Averages

In [17]:
```
monthly_sales['Rolling_Mean_3'] = monthly_sales['Weekly_Sales'].rolling(window=3).m
monthly_sales['Rolling_Mean_6'] = monthly_sales['Weekly_Sales'].rolling(window=6).m
```

In [21]:
```
plt.figure(figsize=(14, 5))
plt.plot(monthly_sales['Month'], monthly_sales['Rolling_Mean_3'], label='3-Month MA
plt.plot(monthly_sales['Month'], monthly_sales['Rolling_Mean_6'], label='6-Month MA
plt.legend()
plt.xlabel('Month')
plt.ylabel('Sales')
```

```
plt.title('Moving Average of Monthly Sales')
plt.grid(True)
plt.show()
```



6.Breakdown by Product/Region

Break down revenue by product and region over time and then the visualization

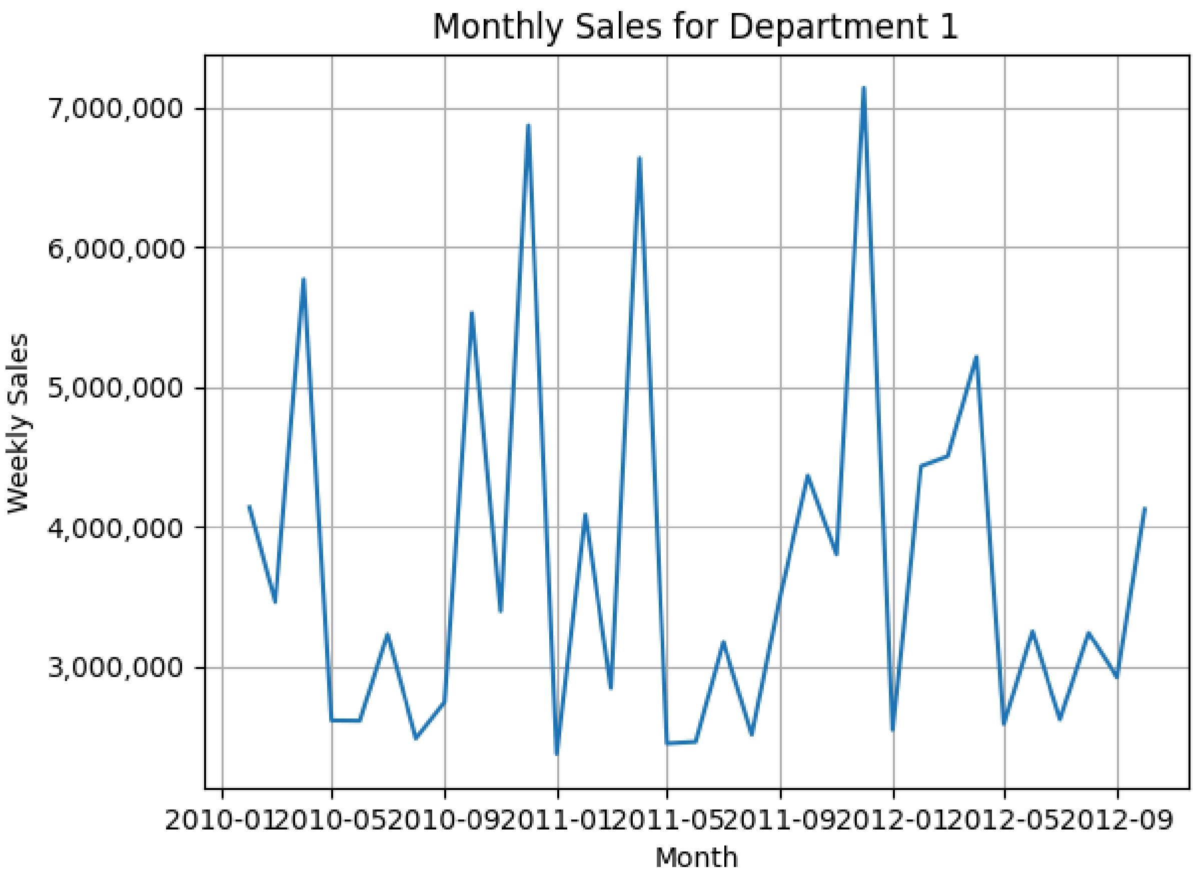
In [22]:
```
dept_sales = df.groupby(['Month', 'Dept'])['Weekly_Sales'].sum().reset_index()
```

In [27]:
```
dept_sales['Month'] = dept_sales['Month'].dt.to_timestamp()
```

In [28]:
```
store_sales = df.groupby(['Month', 'Store'])['Weekly_Sales'].sum().reset_index()
```

In [32]:
```
sns.lineplot(data=dept_sales[dept_sales['Dept'] == 1], x='Month', y='Weekly_Sales')
plt.gca().yaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))

plt.title('Monthly Sales for Department 1')
plt.xlabel('Month')
plt.ylabel('Weekly Sales')
plt.grid(True)
plt.show()
```



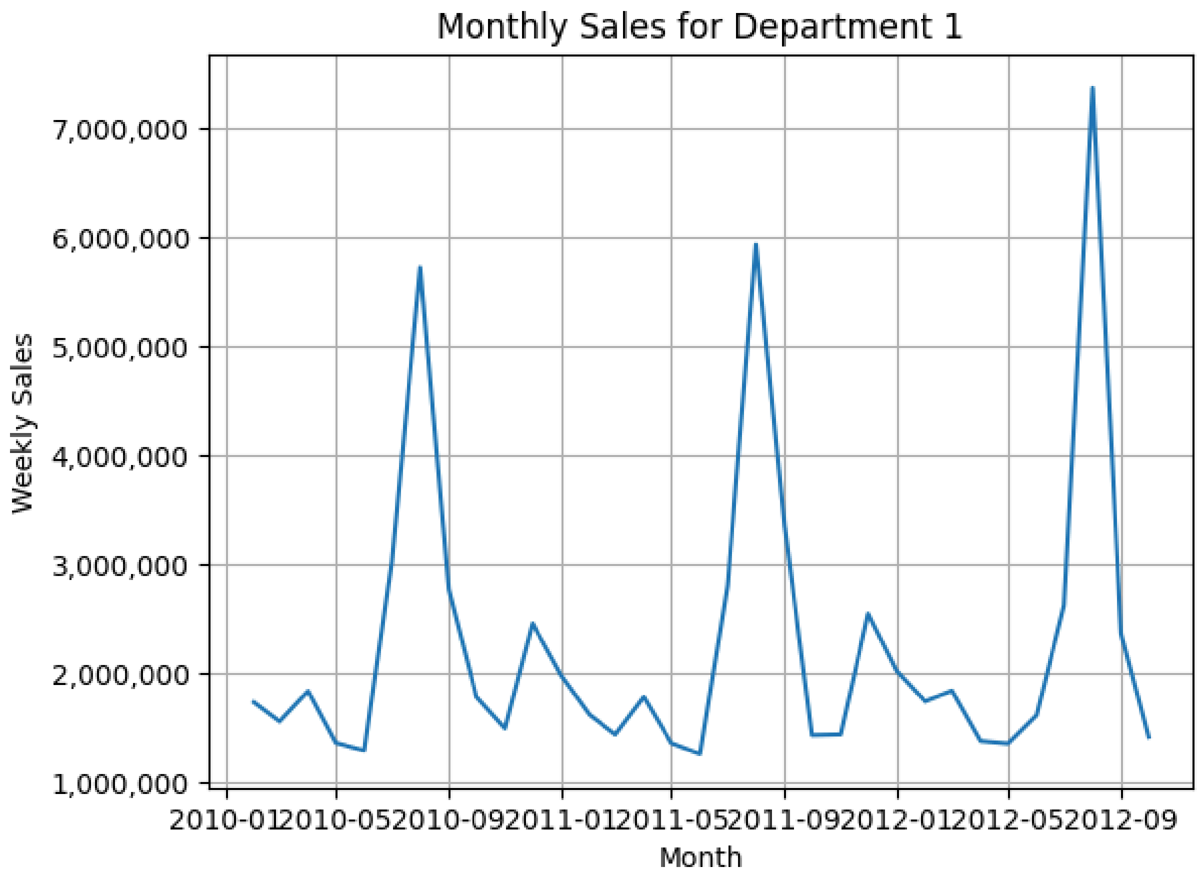
In [31]:
```
dept_sales.head()
```

Out[31]:

	Month	Dept	Weekly_Sales
0	2010-02-01	1	4138664.75
1	2010-02-01	2	7658267.17
2	2010-02-01	3	1739169.28
3	2010-02-01	4	4470499.52
4	2010-02-01	5	4231371.27

In [33]:
```
sns.lineplot(data=dept_sales[dept_sales['Dept'] == 3], x='Month', y='Weekly_Sales')
plt.gca().yaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))

plt.title('Monthly Sales for Department 1')
plt.xlabel('Month')
plt.ylabel('Weekly Sales')
plt.grid(True)
plt.show()
```

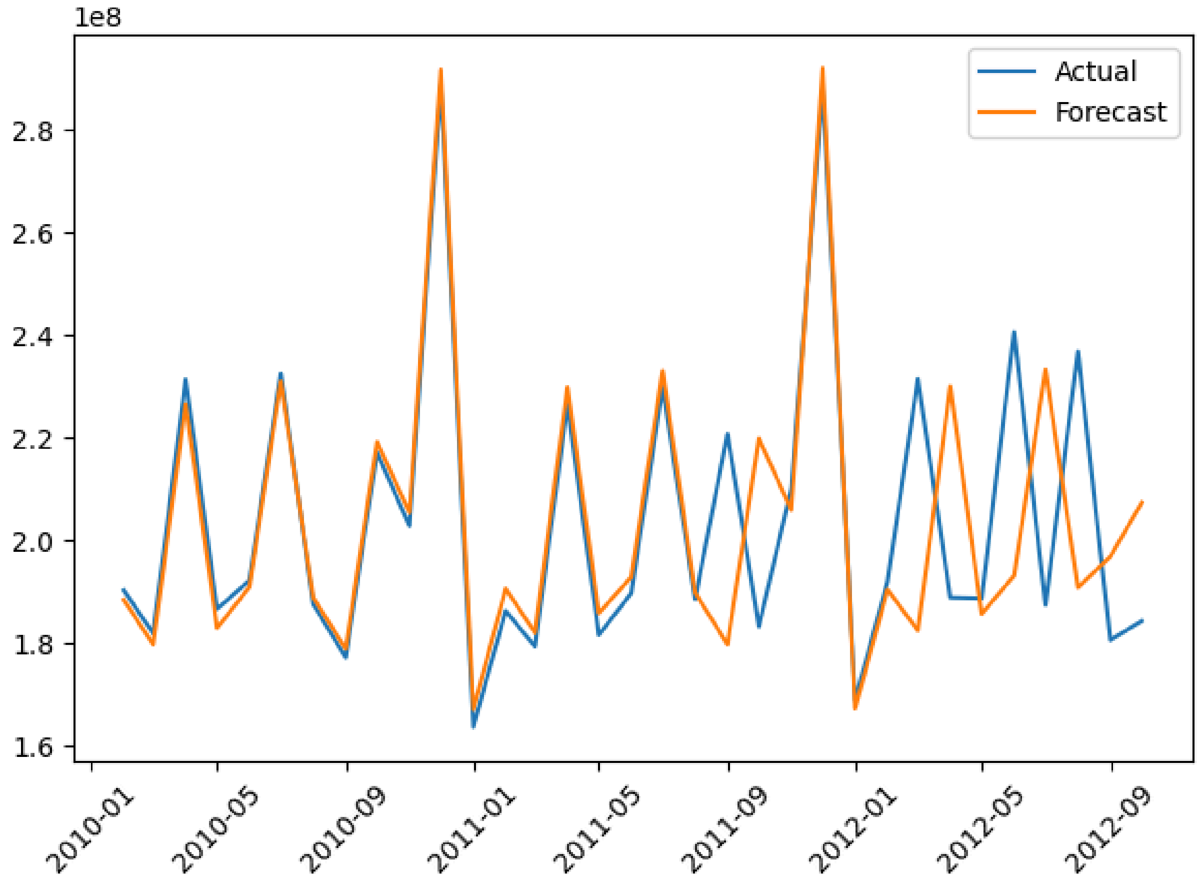


7. Holt-Winters (Triple Exponential Smoothing) - Bonus

It captures trend & seasonality.

In [41]:
```
model = ExponentialSmoothing(monthly_sales['Weekly_Sales'], trend='add', seasonal='
fitted = model.fit()
monthly_sales['Forecast'] = fitted.fittedvalues

plt.plot(monthly_sales['Month'], monthly_sales['Weekly_Sales'], label='Actual')
plt.plot(monthly_sales['Month'], monthly_sales['Forecast'], label='Forecast')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```


1e8
Actual
Forecast
2.8
2.6
2.4
2.2
2.0
1.8
1.6
2010-01
2010-05
2010-09
2011-01
2011-05
2011-09
2012-01
2012-05
2012-09

In [ ]: