```
from google.colab import files
import pandas as pd
uploaded = files.upload()
df = pd.read_csv('stores_sales_forecasting.csv', encoding='latin1')
print(df.head())
```

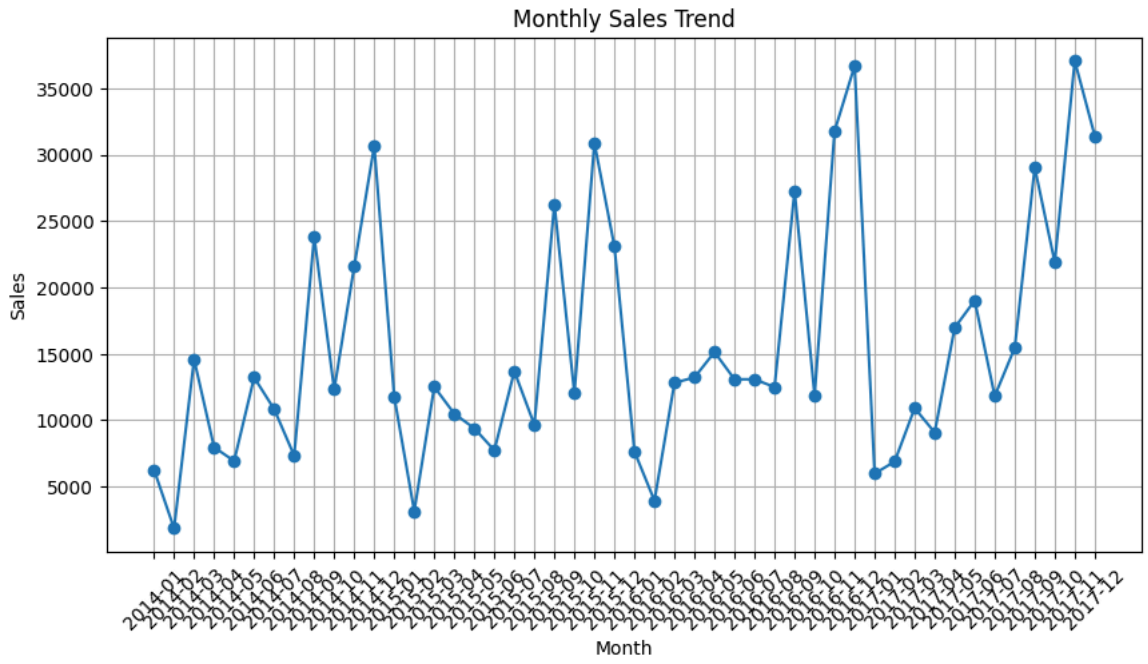
```
Choose Files stores_sale…ecasting.csv
stores_sales_forecasting.csv(text/csv) - 494695 bytes, last modified: 4/12/2024 - 100% done
Saving stores_sales_forecasting.csv to stores_sales_forecasting (4).csv
   Row ID        Order ID  Order Date   Ship Date       Ship Mode Customer ID  \
0       1  CA-2016-152156   11/8/2016  11/11/2016    Second Class    CG-12520
1       2  CA-2016-152156   11/8/2016  11/11/2016    Second Class    CG-12520
2       4  US-2015-108966  10/11/2015  10/18/2015  Standard Class    SO-20335
3       6  CA-2014-115812    6/9/2014   6/14/2014  Standard Class    BH-11710
4      11  CA-2014-115812    6/9/2014   6/14/2014  Standard Class    BH-11710

     Customer Name   Segment        Country             City  ... Postal Code  \
0      Claire Gute  Consumer  United States        Henderson  ...       42420
1      Claire Gute  Consumer  United States        Henderson  ...       42420
2   Sean O'Donnell  Consumer  United States  Fort Lauderdale  ...       33311
3  Brosina Hoffman  Consumer  United States      Los Angeles  ...       90032
4  Brosina Hoffman  Consumer  United States      Los Angeles  ...       90032

   Region       Product ID   Category Sub-Category  \
0   South  FUR-BO-10001798  Furniture    Bookcases
1   South  FUR-CH-10000454  Furniture       Chairs
2   South  FUR-TA-10000577  Furniture       Tables
3    West  FUR-FU-10001487  Furniture  Furnishings
4    West  FUR-TA-10001539  Furniture       Tables

                                   Product Name      Sales  Quantity  \
0                 Bush Somerset Collection Bookcase   261.9600         2
1  Hon Deluxe Fabric Upholstered Stacking Chairs,...   731.9400         3
2      Bretford CR4500 Series Slim Rectangular Table   957.5775         5
3  Eldon Expressions Wood and Plastic Desk Access...    48.8600         7
4           Chromcraft Rectangular Conference Tables  1706.1840         9

   Discount    Profit
0      0.00   41.9136
1      0.00  219.5820
2      0.45 -383.0310
3      0.00   14.1694
4      0.20   85.3092

[5 rows x 21 columns]
```

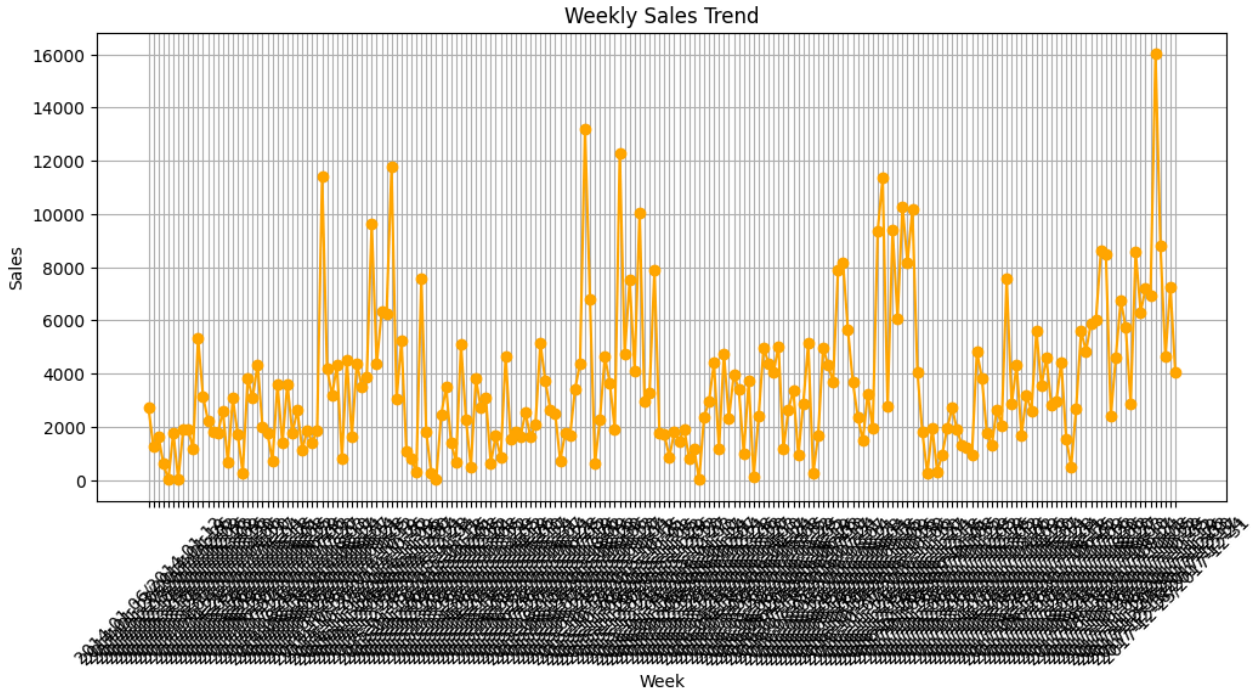
```
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Month'] = df['Order Date'].dt.to_period('M')
df['Week']  = df['Order Date'].dt.to_period('W')
monthly_sales = df.groupby('Month')['Sales'].sum().reset_index()
weekly_sales = df.groupby('Week')['Sales'].sum().reset_index()
print(monthly_sales.head())
print(weekly_sales.head())
```

```
     Month      Sales
0  2014-01   6242.525
1  2014-02   1839.658
2  2014-03  14573.956
3  2014-04   7944.837
4  2014-05   6912.787
                   Week     Sales
0  2014-01-06/2014-01-12  2712.428
1  2014-01-13/2014-01-19  1250.473
2  2014-01-20/2014-01-26  1655.958
3  2014-01-27/2014-02-02   623.666
4  2014-02-03/2014-02-09    14.560
```

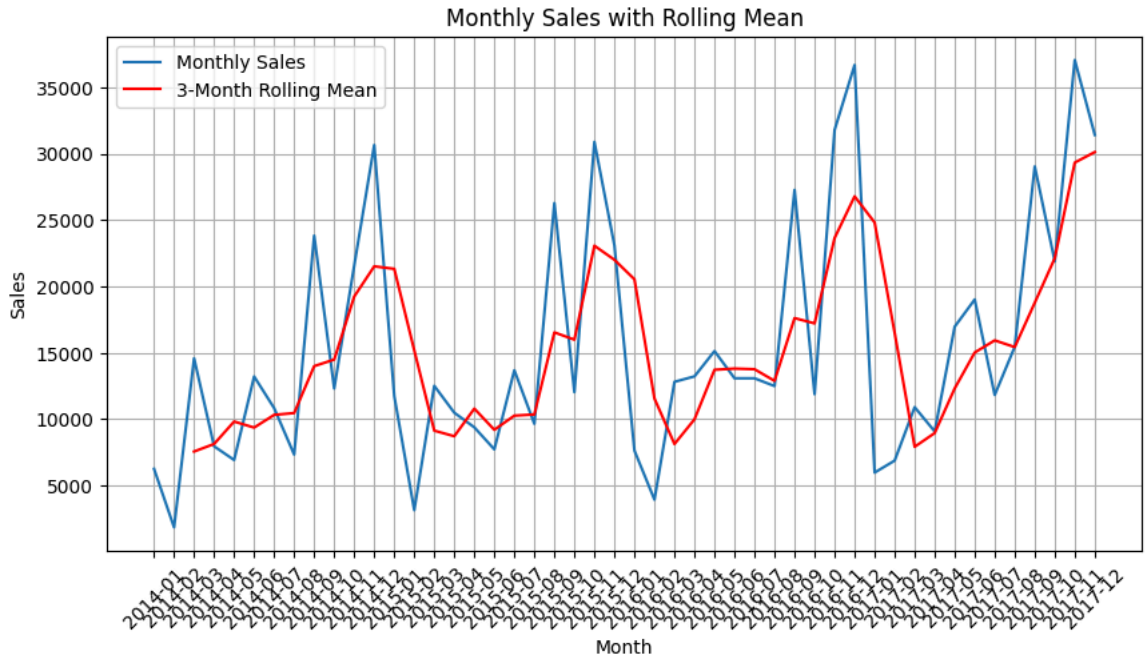
```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,5))
plt.plot(monthly_sales['Month'].astype(str), monthly_sales['Sales'], marker='o')
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
plt.figure(figsize=(12,5))
plt.plot(weekly_sales['Week'].astype(str), weekly_sales['Sales'], marker='o', color='orange')
```

```
plt.title('Weekly Sales Trend')
plt.xlabel('Week')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```





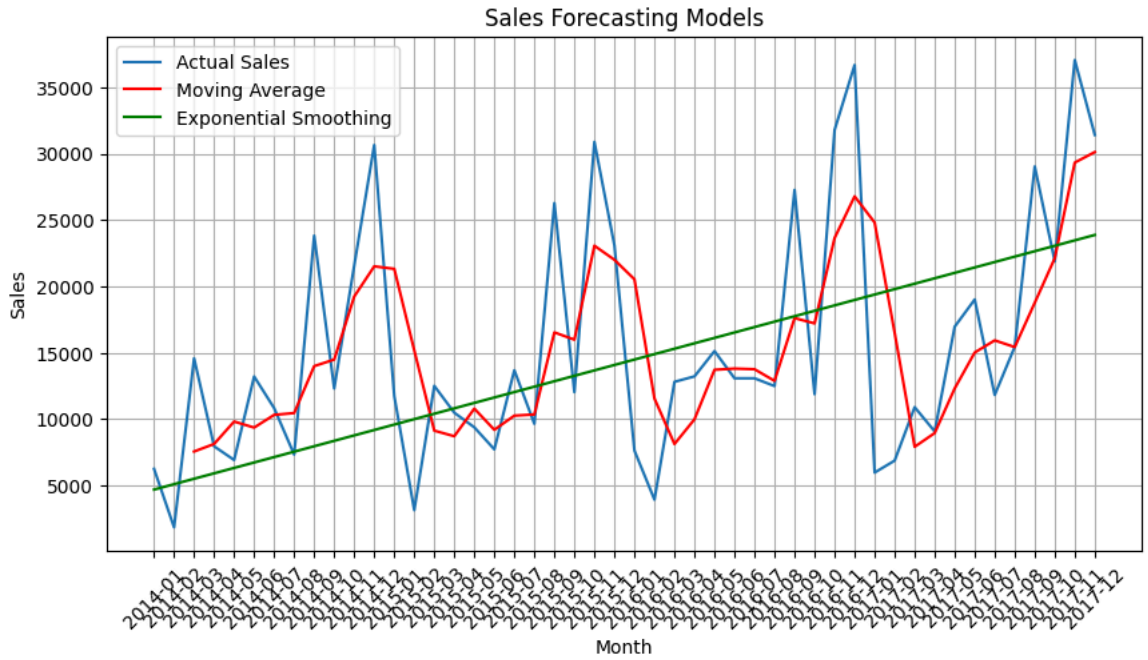

```
monthly_sales['RollingMean'] = monthly_sales['Sales'].rolling(window=3).mean()
import matplotlib.pyplot as plt
plt.figure(figsize=(10,5))
plt.plot(monthly_sales['Month'].astype(str), monthly_sales['Sales'], label='Monthly Sales')
plt.plot(monthly_sales['Month'].astype(str), monthly_sales['RollingMean'], label='3-Month Rolling Mean', color='red')
plt.title('Monthly Sales with Rolling Mean')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.show()
```




```
df = df.sort_values('Order Date')
cutoff_date = '2023-07-01'   # adjust based on dataset timeline
train = df[df['Order Date'] < cutoff_date]
test  = df[df['Order Date'] >= cutoff_date]
print("Train range:", train['Order Date'].min(), "to", train['Order Date'].max())
print("Test range:", test['Order Date'].min(), "to", test['Order Date'].max())
print("Train size:", len(train), "Test size:", len(test))
```

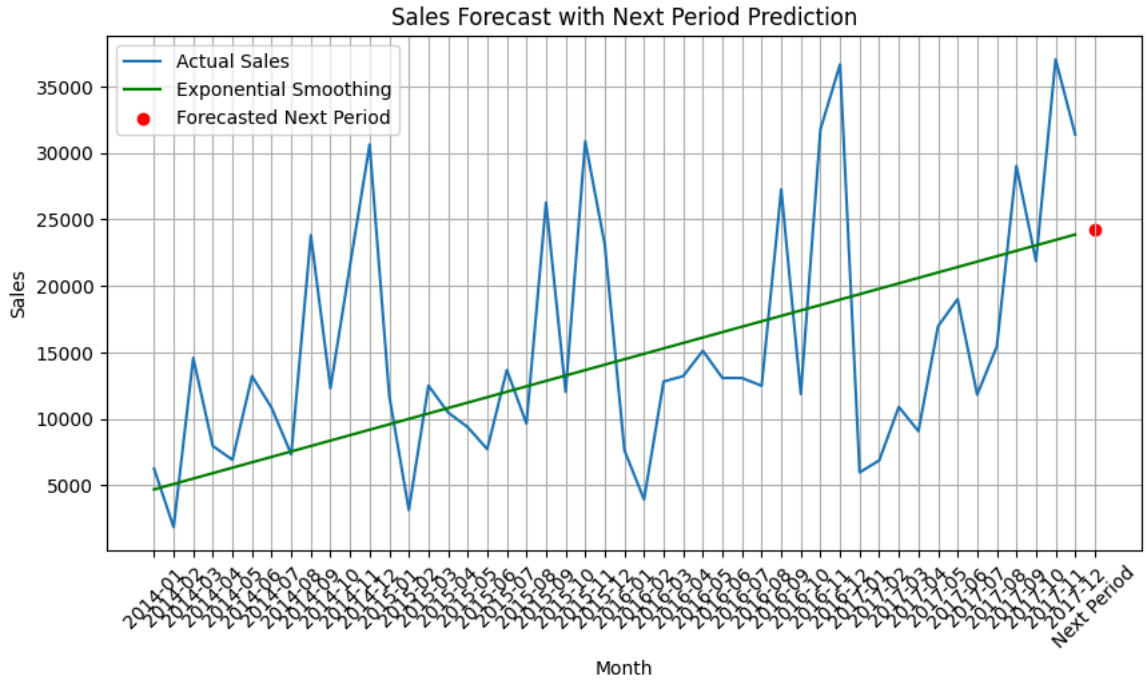
```
Train range: 2014-01-06 00:00:00 to 2017-12-30 00:00:00
Test range: NaT to NaT
Train size: 2121 Test size: 0
```

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing
monthly_sales['Month'] = monthly_sales['Month'].astype(str)
monthly_sales['MA_Forecast'] = monthly_sales['Sales'].rolling(window=3).mean()
model = ExponentialSmoothing(monthly_sales['Sales'], trend='add', seasonal=None)
fit = model.fit()
monthly_sales['ES_Forecast'] = fit.fittedvalues
import matplotlib.pyplot as plt
plt.figure(figsize=(10,5))
plt.plot(monthly_sales['Month'], monthly_sales['Sales'], label='Actual Sales')
plt.plot(monthly_sales['Month'], monthly_sales['MA_Forecast'], label='Moving Average', color='red')
plt.plot(monthly_sales['Month'], monthly_sales['ES_Forecast'], label='Exponential Smoothing', color='green')
plt.title('Sales Forecasting Models')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.show()
```




```
forecast_next = fit.forecast(1)
print("Next period forecasted sales:", forecast_next.iloc[0])
future_months = list(monthly_sales['Month']) + ['Next Period']
future_sales  = list(monthly_sales['Sales']) + [None]
future_es     = list(monthly_sales['ES_Forecast']) + [forecast_next.iloc[0]]
import matplotlib.pyplot as plt
plt.figure(figsize=(10,5))
plt.plot(monthly_sales['Month'], monthly_sales['Sales'], label='Actual Sales')
plt.plot(monthly_sales['Month'], monthly_sales['ES_Forecast'], label='Exponential Smoothing', color='green')
plt.scatter('Next Period', forecast_next.iloc[0], color='red', label='Forecasted Next Period')
plt.title('Sales Forecast with Next Period Prediction')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.show()
```

```
Next period forecasted sales: 24278.65055470222
```




```
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error
actual = monthly_sales['Sales']
forecast = monthly_sales['ES_Forecast']
```

```
mae = mean_absolute_error(actual, forecast)
mape = mean_absolute_percentage_error(actual, forecast)

print("MAE:", mae)
print("MAPE:", mape)
```

```
MAE: 6778.404827899402
MAPE: 0.5631392347178679
```

```
export_df = monthly_sales[['Month','Sales','MA_Forecast','ES_Forecast']]
export_df.to_csv('forecast_results.csv', index=False)

print("Forecast results exported to forecast_results.csv")
```