Inventory Demand Forecasting and Stock Optimization

Objective---

Predict monthly demand of products and optimize stock levels using EOQ,ROP and Safety Stock concetps.

Dataset Description---

Source: Kaggle(Superstore.csv)

Columns:Order Date,Category,Quantity,Sales,Profit etc

Time Range: 2014-2017

Size: 10,000 rows

Steps and Techniques Used---

1.Task--Exploratory Data Analysis (Tools--Pandas ,Matplotlib)

2.Task--Category-wise Demand Aggregation (Tools--GroupBy, Resampling)

3.Task--Demand Forecasting (Tools--Facebook Prophet)

4.Task--Inventory Optimization (Tools--EOQ,safety,Stock,ROP formulas)

5.Task--Visualizations (Tools--Matplotlib)

6.Task--Final Output+Report (Tools Jupyter Notebook)

Key Visualization---

Monthly demand trends

Forecast graph with Prophet

Reorder Point vs Monthly Demand bar Chart

■ Real-World Extension Note:

This project has been extended with real-world business implementation involving:

- Demand forecasting using Prophet/ARIMA models with MAPE evaluation
- Inventory stock optimization using EOQ and Reorder Point formulas
- Deployment plan via Streamlit dashboard for actionable insights

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv("Superstore.csv",encoding="ISO-8859-1")
df.head()
```

Out[2]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Countr
0	1	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	Unite Stat
1	2	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	Unite Stat
2	3	CA-2016-138688	6/12/2016	6/16/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	Unite Stat
3	4	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	Unite Stat
4	5	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	Unite Stat

5 rows × 21 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Row ID         9994 non-null   int64  
 1   Order ID       9994 non-null   object 
 2   Order Date     9994 non-null   object 
 3   Ship Date      9994 non-null   object 
 4   Ship Mode      9994 non-null   object 
 5   Customer ID    9994 non-null   object 
 6   Customer Name  9994 non-null   object 
 7   Segment        9994 non-null   object 
 8   Country        9994 non-null   object 
 9   City           9994 non-null   object 
 10  State          9994 non-null   object 
 11  Postal Code    9994 non-null   int64  
 12  Region         9994 non-null   object 
 13  Product ID     9994 non-null   object 
 14  Category       9994 non-null   object 
 15  Sub-Category   9994 non-null   object 
 16  Product Name   9994 non-null   object 
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64  
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
dtypes: float64(3), int64(3), object(15)
memory usage: 1.6+ MB
```

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

Out[4]:
```
Row ID           0
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
Country          0
City             0
State            0
Postal Code      0
Region           0
Product ID       0
Category         0
Sub-Category     0
Product Name     0
Sales            0
Quantity         0
Discount         0
Profit           0
dtype: int64
```

In [5]:
```
df['Order Date'] =pd.to_datetime(df['Order Date'])
df['Ship Date'] = pd.to_datetime(df['Ship Date'])
```

In [6]:
```
df['Order Month'] = df['Order Date'].dt.to_period('M')
```

In [7]:
```
print("\n Basic Info:\n",df.info())
print("\n Summary Stats:\n",df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9994 non-null   int64
 1   Order ID       9994 non-null   object
 2   Order Date     9994 non-null   datetime64[ns]
 3   Ship Date      9994 non-null   datetime64[ns]
 4   Ship Mode      9994 non-null   object
 5   Customer ID    9994 non-null   object
 6   Customer Name  9994 non-null   object
 7   Segment        9994 non-null   object
 8   Country        9994 non-null   object
 9   City           9994 non-null   object
 10  State          9994 non-null   object
 11  Postal Code    9994 non-null   int64
 12  Region         9994 non-null   object
 13  Product ID     9994 non-null   object
 14  Category       9994 non-null   object
 15  Sub-Category   9994 non-null   object
 16  Product Name   9994 non-null   object
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
 21  Order Month    9994 non-null   period[M]
dtypes: datetime64[ns](2), float64(3), int64(3), object(13), period[M](1)
memory usage: 1.7+ MB

 Basic Info:
 None

 Summary Stats:
            Row ID                     Order Date  \
count  9994.000000                           9994
mean   4997.500000  2016-04-30 00:07:12.259355648
min       1.000000            2014-01-03 00:00:00
25%    2499.250000            2015-05-23 00:00:00
50%    4997.500000            2016-06-26 00:00:00
75%    7495.750000            2017-05-14 00:00:00
max    9994.000000            2017-12-30 00:00:00
std    2885.163629                            NaN

                           Ship Date   Postal Code         Sales     Quantity  \
count                           9994   9994.000000   9994.000000  9994.000000
mean   2016-05-03 23:06:58.571142912  55190.379428    229.858001     3.789574
min              2014-01-07 00:00:00   1040.000000      0.444000     1.000000
25%              2015-05-27 00:00:00  23223.000000     17.280000     2.000000
50%              2016-06-29 00:00:00  56430.500000     54.490000     3.000000
75%              2017-05-18 00:00:00  90008.000000    209.940000     5.000000
max              2018-01-05 00:00:00  99301.000000  22638.480000    14.000000
std                              NaN  32063.693350    623.245101     2.225110

          Discount       Profit
count  9994.000000  9994.000000
```

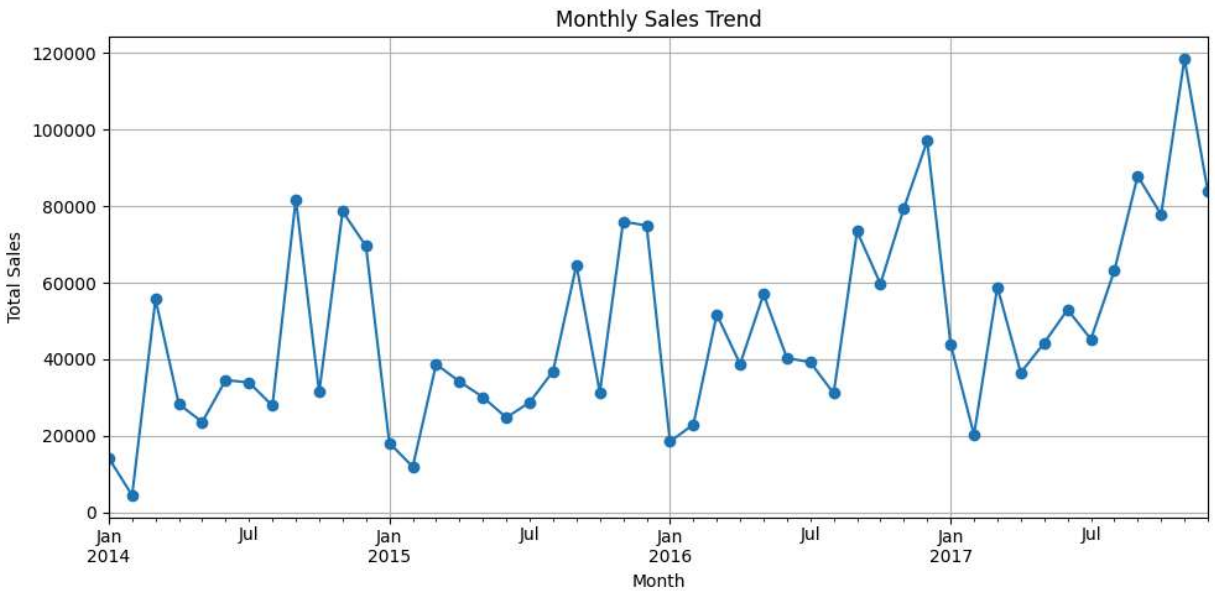
```
mean      0.156203    28.656896
min       0.000000 -6599.978000
25%       0.000000     1.728750
50%       0.200000     8.666500
75%       0.200000    29.364000
max       0.800000  8399.976000
std       0.206452   234.260108
```

In [8]:
```
top_products = df.groupby('Product Name')['Quantity'].sum().sort_values(ascending=F
print("\n Top 10 Products by Quantity Sold:\n",top_products)
```

```
 Top 10 Products by Quantity Sold:
 Product Name
Staples                                                      215
Staple envelope                                              170
Easy-staple paper                                            150
Staples in misc. colors                                       86
KI Adjustable-Height Table                                    74
Storex Dura Pro Binders                                       71
Avery Non-Stick Binders                                       71
GBC Premium Transparent Covers with Diagonal Lined Pattern    67
Situations Contoured Folding Chairs, 4/Set                    64
Staple-based wall hangings                                    62
Name: Quantity, dtype: int64
```

In [9]:
```
monthly_sales = df.groupby('Order Month')['Sales'].sum()

plt.figure(figsize=(10,5))
monthly_sales.plot(marker='o')
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.grid(True)
plt.tight_layout()
plt.show()
```

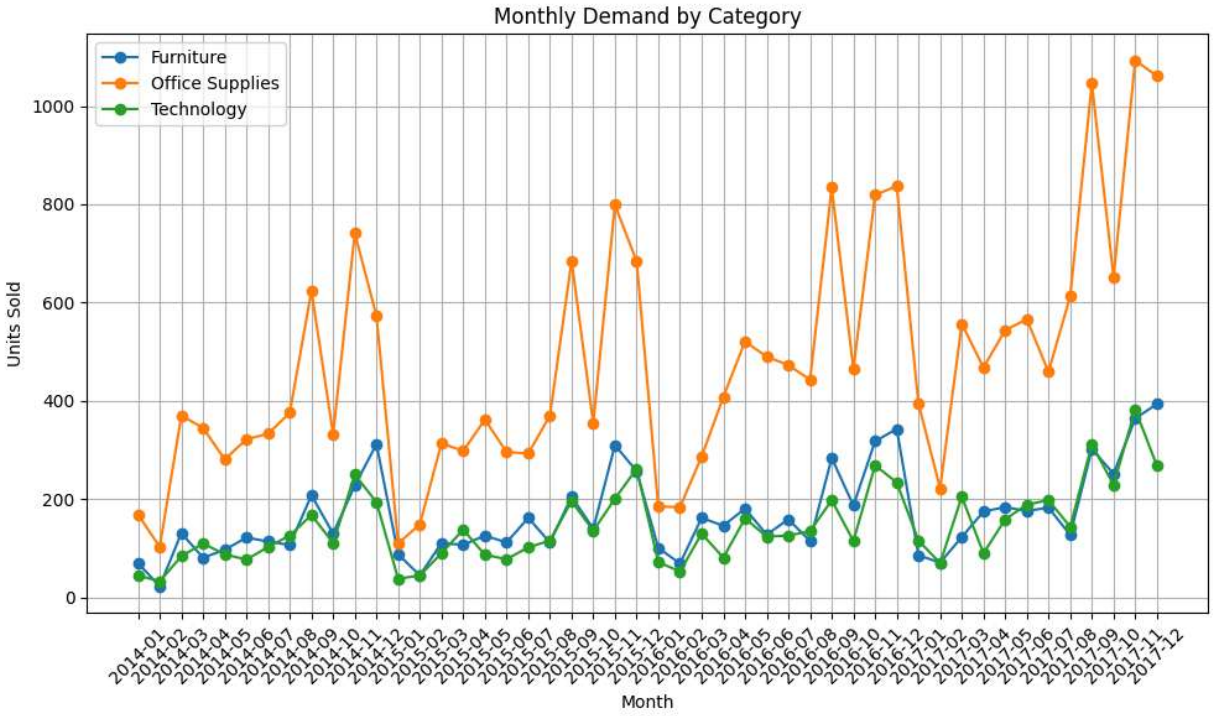


In [10]:
```
monthly_demand = df.groupby(['Category','Order Month'])['Quantity'].sum().reset_ind
monthly_demand['Order Month']=monthly_demand['Order Month'].astype(str)
```

```
print("Aggregated Monthly Demand(sample):\n",monthly_demand.head())
```

```
Aggregated Monthly Demand(sample):
     Category Order Month  Quantity
0  Furniture     2014-01        70
1  Furniture     2014-02        23
2  Furniture     2014-03       131
3  Furniture     2014-04        81
4  Furniture     2014-05        97
```

In [11]:
```
categories = monthly_demand['Category'].unique()
plt.figure(figsize=(10,6))
for cat in categories:
    subset=monthly_demand[monthly_demand['Category']==cat]
    plt.plot(subset['Order Month'],subset['Quantity'],marker='o',label=cat)
plt.xticks(rotation=45)
plt.title("Monthly Demand by Category")
plt.xlabel("Month")
plt.ylabel("Units Sold")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [14]:
```
df['Order Month'] = df['Order Month'].astype(str)
furniture_df =df[df['Category'] =='Furniture']
```

In [15]:
```
monthly_furniture = furniture_df.groupby('Order Month')['Quantity'].sum().reset_ind
monthly_furniture.columns =['ds','y']
monthly_furniture['ds'] = pd.to_datetime(monthly_furniture['ds'])
```

model=Prophet() model.fit(monthly_furniture)

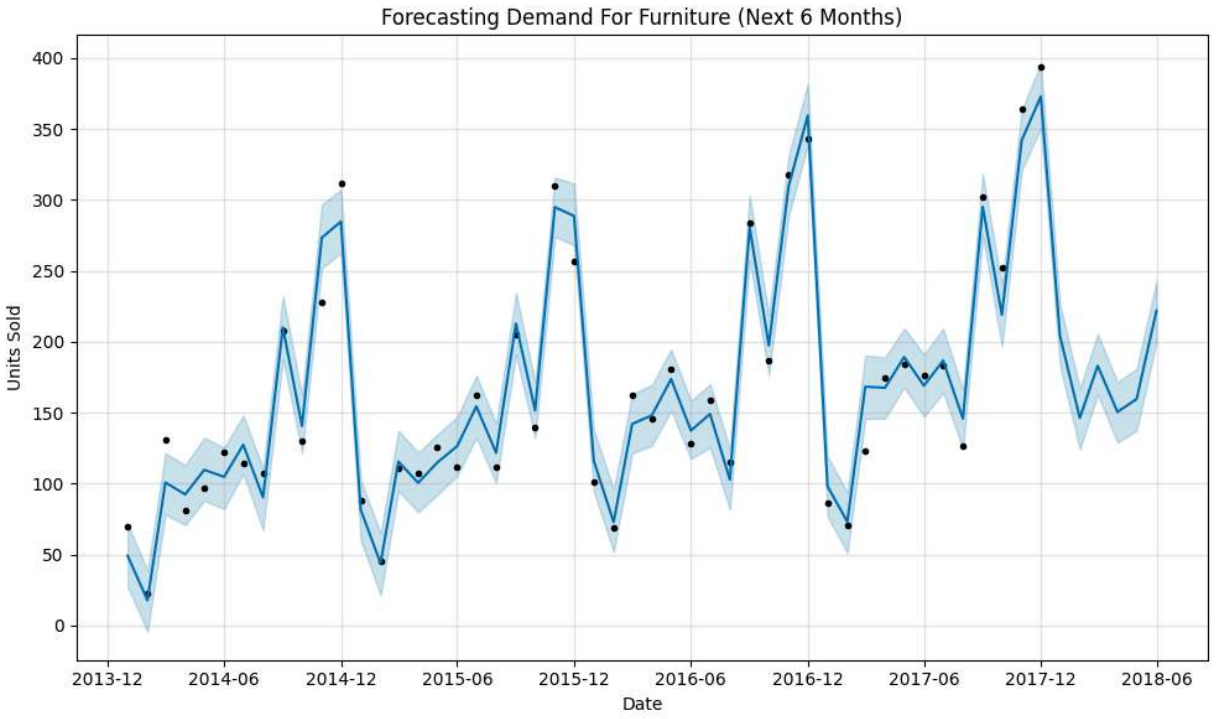
In [18]:
```
from prophet import Prophet
model=Prophet()
model.fit(monthly_furniture)
```

```
23:16:49 - cmdstanpy - INFO - Chain [1] start processing
23:16:49 - cmdstanpy - INFO - Chain [1] done processing
```

Out[18]: <prophet.forecaster.Prophet at 0x1c675d97890>

In [20]:
```
future = model.make_future_dataframe(periods=6,freq ='ME')
forecast = model.predict(future)
```

In [22]:
```
model.plot(forecast)
plt.title("Forecasting Demand For Furniture (Next 6 Months)")
plt.xlabel('Date')
plt.ylabel('Units Sold')
plt.tight_layout()
plt.show()
```



In [27]:
```
daily_demand =monthly_furniture['y'].mean()/30
std_dev = monthly_furniture['y'].std()
```

In [26]:
```
lead_time_days = 7
service_level =1.65

safety_stock = service_level * std_dev*(lead_time_days**0.5)
reorder_point = (daily_demand * lead_time_days)+safety_stock
```

In [28]:
```
ordering_cost =100
holding_cost =2
annual_demand = monthly_furniture['y'].sum()
```

```
eoq =((2* annual_demand * ordering_cost)/ holding_cost)**0.5
```

In [29]:
```
print(f"Safety Stock: {safety_stock: .2f}")
print(f"Reorder point: {reorder_point: .2f}")
print(f"EOQ: {eoq: .2f}")
```

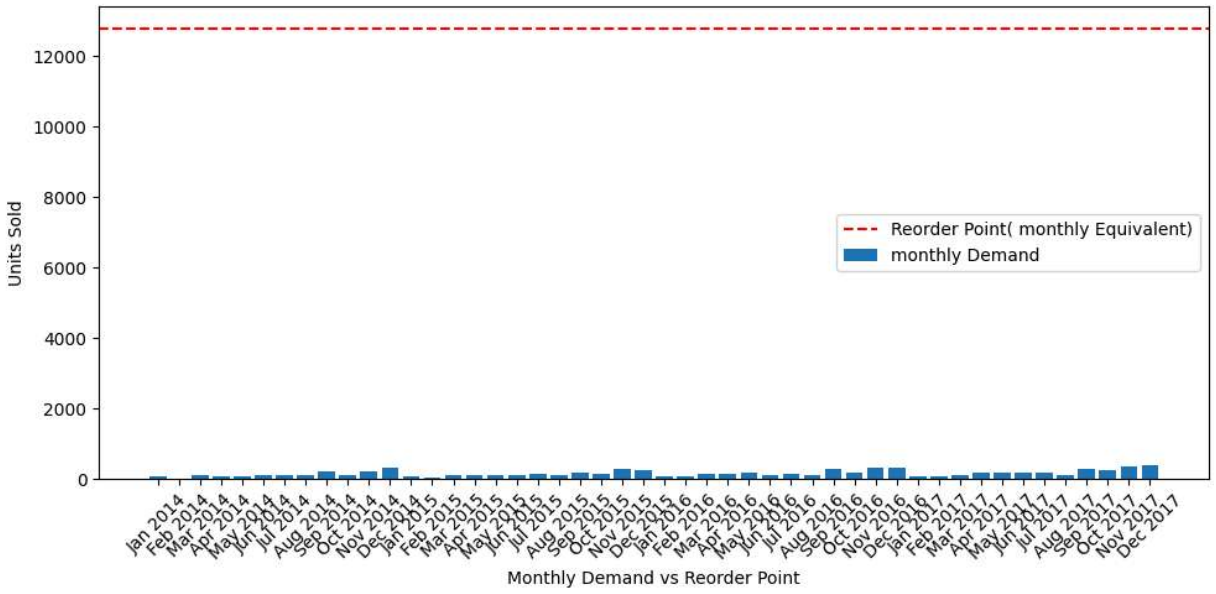
```
Safety Stock:  386.74
Reorder point:  425.77
EOQ:  895.99
```

In [31]:
```
import matplotlib.pyplot as plt

months = monthly_furniture['ds'].dt.strftime('%b %Y')
demand = monthly_furniture['y']

plt.figure(figsize=(10,5))
plt.bar(months,demand,label='monthly Demand')
plt.axhline(y=reorder_point*30 , color ='r',linestyle='--',label='Reorder Point( mo

plt.xticks(rotation=45)
plt.ylabel('Units Sold')
plt.xlabel('Monthly Demand vs Reorder Point')
plt.legend()
plt.tight_layout()
plt.show()
```



Conclusion---

This project successfully demonstrates real-world forecasting and inventory logic.The methodology can be used in any retail supply chain scenario.

In [ ]:

■ Real-World Implementation & Optimization

- Business logic added: Reorder Point (ROP) and Economic Order Quantity (EOQ)
- Forecasting with Prophet and ARIMA for future sales prediction
- Streamlit-based dashboard planned for product-wise stock tracking
- Practical usage in retail, e-commerce, warehouse, and pharma sectors
- Helps reduce overstocking and avoid stockouts