Project Name - Integrated Retail Analytics for Store Optimization

Project Type - Forecasting, Analysis and Model building

Contribution - Individual

Team Member 1 - Sudip Bairagi

Project Summary -

In today’s fast-paced and data-driven retail environment, leveraging machine learning and advanced data analysis techniques has become essential for maintaining a competitive edge. The primary objective is to harness the power of these technologies to optimize store performance, improve operational efficiency, forecast customer demand with greater accuracy, and deliver superior customer experiences. By applying predictive modeling, clustering algorithms, and real-time analytics, businesses can gain actionable insights that drive informed decision-making and strategic planning across all levels of the organization.

One of the most impactful applications of machine learning is in **demand forecasting**. Traditional methods often rely on historical sales data alone, but machine learning models incorporate a variety of variables—such as seasonality, promotional activity, economic indicators, customer behavior, and even weather patterns—to provide more nuanced and accurate predictions. Accurate demand forecasts allow stores to maintain optimal inventory levels, minimize stockouts or overstocking, reduce waste, and streamline supply chain management. This leads to increased profitability and improved customer satisfaction by ensuring that the right products are available at the right time.

In addition to operational optimization, **enhancing the customer experience** through data-driven strategies is a central goal. By segmenting customers based on purchasing patterns, demographics, behavior, and preferences, retailers can develop highly targeted and personalized marketing campaigns. Machine learning techniques such as k-means clustering, decision trees, and neural networks can be employed to identify and profile distinct customer segments. This enables marketers to tailor their messaging, product recommendations, and promotional offers to align with the unique needs and interests of each group, thereby improving engagement and conversion rates.

Furthermore, **personalized marketing strategies** powered by machine learning lead to deeper customer relationships and greater brand loyalty. For instance, recommendation engines, often based on collaborative filtering and content-based algorithms, analyze a customer's previous interactions and compare them with similar users to suggest products that are likely to resonate. These tailored experiences not only drive sales but also enhance the perceived value of the brand, creating a more satisfying and relevant shopping journey for each individual.

Machine learning also supports **real-time decision-making**, enabling businesses to respond dynamically to market trends and customer behaviors as they unfold. From adjusting pricing strategies to managing labor deployment in stores, the ability to analyze large volumes of data in real time facilitates a more agile and responsive business model.

Ultimately, the integration of machine learning and data analytics into retail operations empowers organizations to transition from reactive to proactive strategies. It provides a holistic view of the business, unifies insights across departments, and supports continuous improvement. By embracing these technologies, retailers can not only improve store performance and predict demand with precision but also foster meaningful customer relationships through data-driven personalization. The result is a more efficient, customer-centric, and future-ready business that is well-positioned for sustainable growth in an increasingly competitive marketplace.

GitHub Link -

Provide your GitHub Link here.

Problem Statement

Project Objective: To utilize machine learning and data analysis techniques to optimize store performance, forecast demand, and enhance customer experience through segmentation and personalized marketing strategies.

Project Components:

Anomaly Detection in Sales Data: Identify unusual sales patterns across stores and departments. Investigate potential causes (e.g., holidays, markdowns, economic indicators). Implement anomaly handling strategies to clean the data for further analysis.

Time-Based Anomaly Detection: Analyze sales trends over time. Detect seasonal variations and holiday effects on sales. Use time-series analysis for understanding store and department performance over time.

Data Preprocessing and Feature Engineering: Handle missing values, especially in the MarkDown data. Create new features that could influence sales (e.g., store size/type, regional factors).

Customer Segmentation Analysis: Segment stores or departments based on sales patterns, markdowns, and regional features. Analyze segment-specific trends and characteristics.

Market Basket Analysis: Although individual customer transaction data is not available, infer potential product associations within departments using sales data. Develop cross-selling strategies based on these inferences.

Demand Forecasting: Build models to forecast weekly sales for each store and department. Incorporate factors like CPI, unemployment rate, fuel prices, and store/dept attributes. Explore short-term and long-term forecasting models.

Impact of External Factors: Examine how external factors (economic indicators, regional climate) influence sales. Incorporate these insights into the demand forecasting models.

Personalization Strategies: Develop personalized marketing strategies based on the markdowns and store segments. Propose inventory management strategies tailored to store and department needs.

Segmentation Quality Evaluation: Evaluate the effectiveness of the customer segmentation. Use metrics to assess the quality of segments in terms of homogeneity and separation.

Real-World Application and Strategy Formulation: Formulate a comprehensive strategy for inventory management, marketing, and store optimization based on the insights gathered. Discuss potential real-world challenges in implementing these strategies.

Tools and Techniques: Machine Learning (e.g., clustering, time-series forecasting models, association rules). Data Preprocessing and Visualization. Statistical Analysis.

Deliverables: A detailed report with analysis, insights, and strategic recommendations. Predictive models for sales forecasting and anomaly detection. Segmentation analysis and market basket insights. Code and data visualizations to support findings.

1. Know Your Data

Import Libraries

```
import pandas as pd
```

Dataset Loading

```
# Load Dataset from data folder
features_df = pd.read_csv('data/Features data set.csv')
sales_data_df = pd.read_csv('data/sales data-set.csv')
stores_df = pd.read_csv('data/stores data-set.csv')
```

Dataset First View

```
# Dataset First Look
features_df.head()
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment
0	1	05/02/2010	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106
1	1	12/02/2010	38.51	2.548	NaN	NaN	NaN	NaN	NaN	211.242170	8.106
2	1	19/02/2010	39.93	2.514	NaN	NaN	NaN	NaN	NaN	211.289143	8.106
3	1	26/02/2010	46.63	2.561	NaN	NaN	NaN	NaN	NaN	211.319643	8.106
4	1	05/03/2010	46.50	2.625	NaN	NaN	NaN	NaN	NaN	211.350143	8.106

Next steps: Generate code with features_df View recommended plots New interactive sheet

```
features_df.describe()
```

	Store	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	U
count	8190.000000	8190.000000	8190.000000	4032.000000	2921.000000	3613.000000	3464.000000	4050.000000	7605.000000	
mean	23.000000	59.356198	3.405992	7032.371786	3384.176594	1760.100180	3292.935886	4132.216422	172.460809	
std	12.987966	18.678607	0.431337	9262.747448	8793.583016	11276.462208	6792.329861	13086.690278	39.738346	
min	1.000000	-7.290000	2.472000	-2781.450000	-265.760000	-179.260000	0.220000	-185.170000	126.064000	
25%	12.000000	45.902500	3.041000	1577.532500	68.880000	6.600000	304.687500	1440.827500	132.364839	
50%	23.000000	60.710000	3.513000	4743.580000	364.570000	36.260000	1176.425000	2727.135000	182.764003	
75%	34.000000	73.880000	3.743000	8923.310000	2153.350000	163.150000	3310.007500	4832.555000	213.932412	
max	45.000000	101.950000	4.468000	103184.980000	104519.540000	149483.310000	67474.850000	771448.100000	228.976456	

```
sales_data_df.head()
```

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

```
sales_data_df.describe()
```

	Store	Dept	Weekly_Sales
count	421570.000000	421570.000000	421570.000000
mean	22.200546	44.260317	15981.258123
std	12.785297	30.492054	22711.183519
min	1.000000	1.000000	-4988.940000
25%	11.000000	18.000000	2079.650000
50%	22.000000	37.000000	7612.030000
75%	33.000000	74.000000	20205.852500
max	45.000000	99.000000	693099.360000

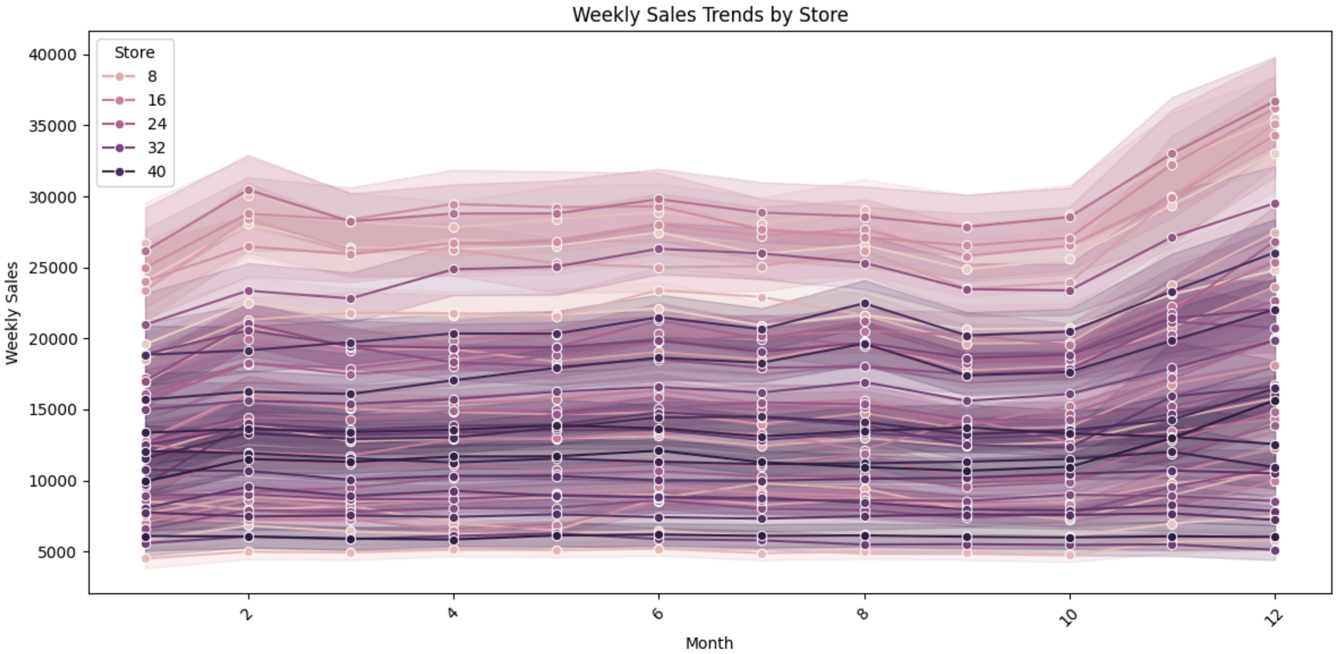
```
stores_df.head()
```

	Store	Type	Size
0	1	A	151315
1	2	A	202307
2	3	B	37392
3	4	A	205863
4	5	B	34875

```
stores_df.describe()
```

	Store	Size
count	45.000000	45.000000
mean	23.000000	130287.600000
std	13.133926	63825.271991
min	1.000000	34875.000000
25%	12.000000	70713.000000
50%	23.000000	126512.000000
75%	34.000000	202307.000000
max	45.000000	219622.000000

```
# Create a trends plot for the sales data. Plot the monthly sales trends for each store.
sales_data_df['Date'] = pd.to_datetime(sales_data_df['Date'], format='%d/%m/%Y')
sales_data_df['Year'] = sales_data_df['Date'].dt.year
sales_data_df['Month'] = sales_data_df['Date'].dt.month
sales_data_df['Weekly_Sales'] = sales_data_df['Weekly_Sales'].astype(float)

import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(12, 6))
sns.lineplot(data=sales_data_df, x='Month', y='Weekly_Sales', hue='Store', marker='o')
plt.title('Weekly Sales Trends by Store')
# plt.xlabel('Date')
plt.ylabel('Weekly Sales')
plt.xticks(rotation=45)
plt.legend(title='Store')
plt.tight_layout()
plt.show()
```



Dataset Rows & Columns count

```
# Dataset Rows & Columns count
print("Shape of features_df:", features_df.shape)
print("Shape of sales_data_df:", sales_data_df.shape)
print("Shape of stores_df:", stores_df.shape)
```

```
Shape of features_df: (8190, 12)
Shape of sales_data_df: (421570, 7)
Shape of stores_df: (45, 3)
```

Dataset Information

```
# Dataset Info
print(features_df.info())
print(sales_data_df.info())
print(stores_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8190 entries, 0 to 8189
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         8190 non-null   int64
 1   Date          8190 non-null   object
 2   Temperature   8190 non-null   float64
 3   Fuel_Price    8190 non-null   float64
 4   MarkDown1     4032 non-null   float64
 5   MarkDown2     2921 non-null   float64
```

```
 6   MarkDown3     3613 non-null   float64
 7   MarkDown4     3464 non-null   float64
 8   MarkDown5     4050 non-null   float64
 9   CPI           7605 non-null   float64
 10  Unemployment  7605 non-null   float64
 11  IsHoliday     8190 non-null   bool
dtypes: bool(1), float64(9), int64(1), object(1)
memory usage: 712.0+ KB
None
<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   Year          421570 non-null  int32
 6   Month         421570 non-null  int32
dtypes: bool(1), datetime64[ns](1), float64(1), int32(2), int64(2)
memory usage: 16.5 MB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45 entries, 0 to 44
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Store   45 non-null     int64
 1   Type    45 non-null     object
 2   Size    45 non-null     int64
dtypes: int64(2), object(1)
memory usage: 1.2+ KB
None
```

Duplicate Values

```
# Dataset Duplicate Value Count
```

Missing Values/Null Values

```
# Find Missing Values/Null Values Count for features_df, sales_df and stores_df
print("Missing Values in features_df:")
print(features_df.isnull().sum())
print("\nMissing Values in sales_df:")
print(sales_data_df.isnull().sum())
print("\nMissing Values in stores_df:")
print(stores_df.isnull().sum())
```

```
Missing Values in features_df:
Store              0
Date               0
Temperature        0
Fuel_Price         0
MarkDown1       4158
MarkDown2       5269
MarkDown3       4577
MarkDown4       4726
MarkDown5       4140
CPI              585
Unemployment     585
IsHoliday          0
dtype: int64

Missing Values in sales_df:
Store           0
Dept            0
Date            0
Weekly_Sales    0
IsHoliday       0
Year            0
Month           0
dtype: int64

Missing Values in stores_df:
Store    0
Type     0
Size     0
dtype: int64
```

```
# Convert null values
features_df['MarkDown1'] = features_df['MarkDown1'].fillna(0)
features_df['MarkDown2'] = features_df['MarkDown2'].fillna(0)
features_df['MarkDown3'] = features_df['MarkDown3'].fillna(0)
features_df['MarkDown4'] = features_df['MarkDown4'].fillna(0)
features_df['MarkDown5'] = features_df['MarkDown5'].fillna(0)
features_df['CPI'] = features_df['CPI'].fillna(0)
features_df['Unemployment'] = features_df['Unemployment'].fillna(0)
features_df['IsHoliday'] = features_df['IsHoliday'].fillna(0)
```

```
# encode categorical columns of features_df
features_df = pd.get_dummies(features_df, columns=['IsHoliday'], drop_first=True)
```

```
# LabelEncoder on isHoliday_True
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
features_df['IsHoliday_True'] = label_encoder.fit_transform(features_df['IsHoliday_True'])
features_df.head()
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment
0	1	05/02/2010	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358	8.106
1	1	12/02/2010	38.51	2.548	0.0	0.0	0.0	0.0	0.0	211.242170	8.106
2	1	19/02/2010	39.93	2.514	0.0	0.0	0.0	0.0	0.0	211.289143	8.106
3	1	26/02/2010	46.63	2.561	0.0	0.0	0.0	0.0	0.0	211.319643	8.106
4	1	05/03/2010	46.50	2.625	0.0	0.0	0.0	0.0	0.0	211.350143	8.106

Next steps: Generate code with features_df View recommended plots New interactive sheet

2. Anomaly Detection

Use ***IsolationForest*** to fnd Anomaly

```
# detect anamoly in features_df data using isolationForest
from sklearn.ensemble import IsolationForest
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score

# create a scikit-learn pipeline with StandardScaler and IsolationForest and fit with features_df data


#Create the pipeline

pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('isolation_forest', IsolationForest(contamination='auto', random_state=42))
])

# Select numerical columns for fitting (excluding 'Store' and 'IsHoliday' which are not suitable for direct scaling/anomaly
# Assuming 'Date' has been converted to datetime and is not included in numerical features for this step
numerical_features = features_df.select_dtypes(include=np.number).columns.tolist()
features_to_fit = features_df[numerical_features].drop(columns=['Store'])


# Fit the pipeline to the data
pipeline.fit(features_to_fit)

# Predict anomalies (-1 for outliers, 1 for inliers)
features_df['anomaly'] = pipeline.predict(features_to_fit)

# Display the number of anomalies detected
print(f"Number of anomalies detected: {features_df[features_df['anomaly'] == -1].shape[0]}")

# Display the rows with anomalies
print("Anomalies in features_df:")
display(features_df[features_df['anomaly'] == -1].head())
```

```
Number of anomalies detected: 666
Anomalies in features_df:
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployme
94	1	25/11/2011	60.14	3.236	410.31	98.00	55805.51	8.00	554.92	218.467621	7.8
95	1	02/12/2011	48.91	3.172	5629.51	68.00	1398.11	2084.64	20475.32	218.714733	7.8
99	1	30/12/2011	44.55	3.129	5762.10	46011.38	260.36	983.65	4735.78	219.535990	7.8
104	1	03/02/2012	56.55	3.360	34577.06	3579.21	160.53	32403.87	5630.40	220.172015	7.3
105	1	10/02/2012	48.02	3.409	13925.06	6927.23	101.64	8471.88	6886.04	220.265178	7.3

```
# sort features_df
features_df.sort_values(by='Date', inplace=True)
features_df.head()
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemploym
7436	41	01/02/2013	30.86	2.998	19679.74	1148.08	412.8	30618.76	6293.68	200.891935	5.
5434	30	01/02/2013	55.38	3.244	1019.93	11.72	16.6	36.53	647.71	223.869197	6.
3068	17	01/02/2013	28.23	3.029	17447.15	696.14	123.2	26529.19	1695.28	132.153710	5.
4888	27	01/02/2013	34.17	3.806	26226.09	2773.71	52.0	51587.03	3215.40	142.868066	7.
6708	37	01/02/2013	63.45	3.244	213.76	12.81	15.1	0.00	1063.60	222.905843	6.

```
# filer features df where anomly = 1
features_df_anomaly = features_df[features_df['anomaly'] == -1]
features_df_anomaly.head()
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemploym
7436	41	01/02/2013	30.86	2.998	19679.74	1148.08	412.80	30618.76	6293.68	200.891935	5.
3068	17	01/02/2013	28.23	3.029	17447.15	696.14	123.20	26529.19	1695.28	132.153710	5.
4888	27	01/02/2013	34.17	3.806	26226.09	2773.71	52.00	51587.03	3215.40	142.868066	7.
4342	24	01/02/2013	30.77	3.806	16578.16	3771.20	248.70	36615.03	1773.02	138.796613	8.
2340	13	01/02/2013	30.44	3.029	21473.20	1904.19	424.53	26859.39	3342.06	132.153710	5.

```
features_df_anomaly.describe()
```

	Store	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Une
count	666.000000	666.000000	666.000000	666.000000	666.000000	666.000000	666.000000	666.000000	666.000000	
mean	20.085586	53.853348	3.534926	15634.686532	7183.673874	8783.868544	9281.512477	8381.296231	130.841685	
std	11.867446	21.171288	0.269592	17355.696498	15128.040310	25056.963005	13016.587713	31282.320209	83.124091	
min	1.000000	-6.080000	2.572000	0.000000	-5.980000	-89.100000	0.000000	0.000000	0.000000	
25%	11.000000	36.235000	3.385250	4752.677500	79.625000	58.917500	602.117500	2082.722500	127.326371	
50%	20.000000	52.265000	3.560500	9758.135000	828.725000	205.150000	4418.360000	4443.970000	138.424667	
75%	28.000000	72.740000	3.753000	20262.670000	4523.302500	644.060000	11696.832500	8058.267500	213.023622	
max	45.000000	101.950000	4.178000	103184.980000	104519.540000	149483.310000	67474.850000	771448.100000	228.020781	

```
features_df_anomaly.dtypes
```

	0
Store	int64
Date	object
Temperature	float64
Fuel_Price	float64
MarkDown1	float64
MarkDown2	float64
MarkDown3	float64
MarkDown4	float64
MarkDown5	float64
CPI	float64
Unemployment	float64
IsHoliday_True	int64
anomaly	int64

dtype: object

```
# count anomaly for each month
# convert Date object to dd/mm/yyyy Datetime format
features_df_anomaly['Date'] = pd.to_datetime(features_df_anomaly['Date'], format='%d/%m/%Y')
features_df_anomaly['Month'] = features_df_anomaly['Date'].dt.month
monthly_anomaly_count = features_df_anomaly.groupby('Month').size()
monthly_anomaly_count
```

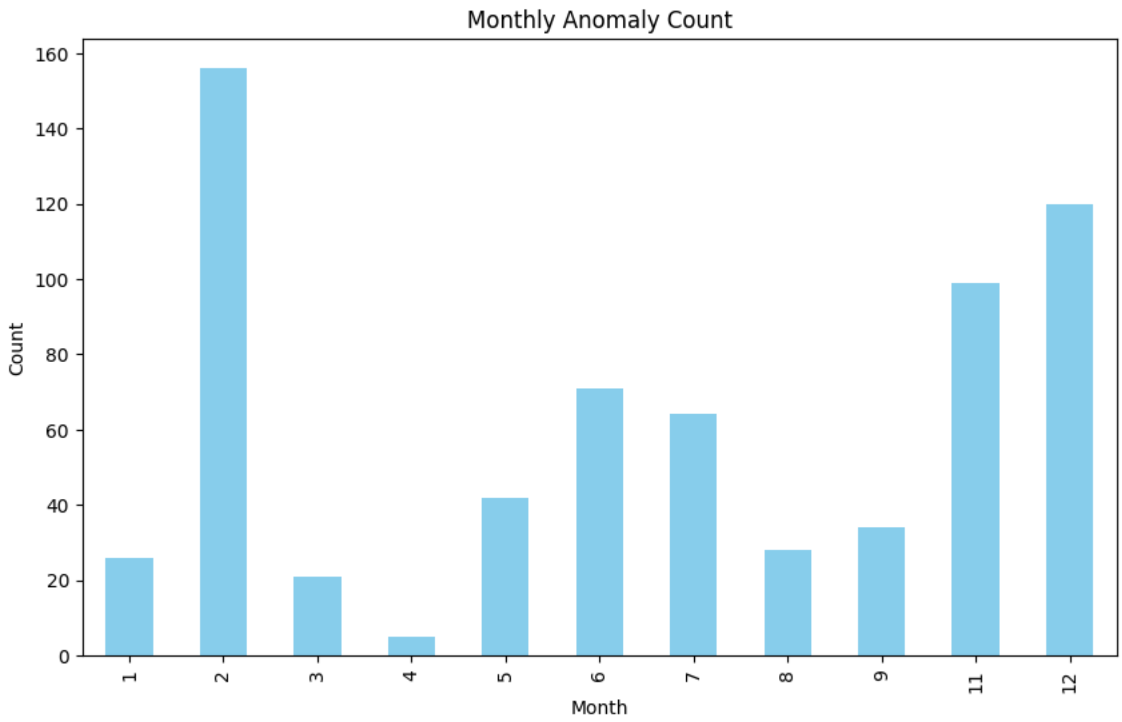
```
/tmp/ipython-input-29-2907972600.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
  features_df_anomaly['Date'] = pd.to_datetime(features_df_anomaly['Date'], format='%d/%m/%Y')
/tmp/ipython-input-29-2907972600.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
  features_df_anomaly['Month'] = features_df_anomaly['Date'].dt.month
```

Month	0
1	26
2	156
3	21
4	5
5	42
6	71
7	64
8	28
9	34
11	99
12	120

dtype: int64

```
# Visualize monthly_anomaly_count
plt.figure(figsize=(10, 6))
monthly_anomaly_count.plot(kind='bar', color='skyblue')
plt.title('Monthly Anomaly Count')
plt.xlabel('Month')
plt.ylabel('Count')
```

```
Text(0, 0.5, 'Count')
```




```
features_df.describe()
```

	Store	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	U
count	8190.000000	8190.000000	8190.000000	8190.000000	8190.000000	8190.000000	8190.000000	8190.000000	8190.000000	
mean	23.000000	59.356198	3.405992	3462.090725	1206.981664	776.464219	1392.763115	2043.403725	160.142180	
std	12.987966	18.678607	0.431337	7388.916286	5495.556015	7539.953758	4707.111488	9431.223215	58.645545	
min	1.000000	-7.290000	2.472000	-2781.450000	-265.760000	-179.260000	0.000000	-185.170000	0.000000	
25%	12.000000	45.902500	3.041000	0.000000	0.000000	0.000000	0.000000	0.000000	131.051167	
50%	23.000000	60.710000	3.513000	0.000000	0.000000	0.000000	0.000000	0.000000	140.587450	
75%	34.000000	73.880000	3.743000	4639.585000	98.590000	24.220000	774.692500	2680.295000	212.766994	
max	45.000000	101.950000	4.468000	103184.980000	104519.540000	149483.310000	67474.850000	771448.100000	228.976456	

3. Understanding comprehensive sales data

```
sales_data_df.head()
```

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

```
features_df_anomaly.head()
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment
7436	41	2013-02-01	30.86	2.998	19679.74	1148.08	412.80	30618.76	6293.68	200.891935	5.771
3068	17	2013-02-01	28.23	3.029	17447.15	696.14	123.20	26529.19	1695.28	132.153710	5.275
4888	27	2013-02-01	34.17	3.806	26226.09	2773.71	52.00	51587.03	3215.40	142.868066	7.945

```
features_df.head()
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemploym
7436	41	01/02/2013	30.86	2.998	19679.74	1148.08	412.8	30618.76	6293.68	200.891935	5.
5434	30	01/02/2013	55.38	3.244	1019.93	11.72	16.6	36.53	647.71	223.869197	6.
3068	17	01/02/2013	28.23	3.029	17447.15	696.14	123.2	26529.19	1695.28	132.153710	5.
4888	27	01/02/2013	34.17	3.806	26226.09	2773.71	52.0	51587.03	3215.40	142.868066	7.
6708	37	01/02/2013	63.45	3.244	213.76	12.81	15.1	0.00	1063.60	222.905843	6.

```
# check duplicates in sales_data_df
sales_data_df.isnull().sum()
```

	0
Store	0
Dept	0
Date	0
Weekly_Sales	0
IsHoliday	0
Year	0
Month	0

dtype: int64

```
features_df.duplicated().sum()
```

```
np.int64(0)
```

```
features_df.isnull().sum()
```

	0
Store	0
Date	0
Temperature	0
Fuel_Price	0
MarkDown1	0
MarkDown2	0
MarkDown3	0
MarkDown4	0
MarkDown5	0
CPI	0
Unemployment	0
IsHoliday_True	0
anomaly	0

dtype: int64

```
# merge sales_df and features_df on Store and Date colums
features_df['Date'] = pd.to_datetime(features_df['Date'], format='%d/%m/%Y')
sales_data_df['Date'] = pd.to_datetime(sales_data_df['Date'], format='%d/%m/%Y')
sales_features_df = pd.merge(sales_data_df, features_df, on=['Store', 'Date'], how='inner')
display(sales_features_df.head())
```

	Store	Dept	Date	Weekly_Sales	IsHoliday	Year	Month	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDo
0	1	1	2010-02-05	24924.50	False	2010	2	42.31	2.572	0.0	0.0	0.0	
1	1	1	2010-02-12	46039.49	True	2010	2	38.51	2.548	0.0	0.0	0.0	
2	1	1	2010-02-19	41595.55	False	2010	2	39.93	2.514	0.0	0.0	0.0	

```
sales_data_df.describe()
```

	Store	Dept	Date	Weekly_Sales	Year	Month
count	421570.000000	421570.000000	421570	421570.000000	421570.000000	421570.000000
mean	22.200546	44.260317	2011-06-18 08:30:31.963375104	15981.258123	2010.968591	6.449510
min	1.000000	1.000000	2010-02-05 00:00:00	-4988.940000	2010.000000	1.000000
25%	11.000000	18.000000	2010-10-08 00:00:00	2079.650000	2010.000000	4.000000
50%	22.000000	37.000000	2011-06-17 00:00:00	7612.030000	2011.000000	6.000000
75%	33.000000	74.000000	2012-02-24 00:00:00	20205.852500	2012.000000	9.000000
max	45.000000	99.000000	2012-10-26 00:00:00	693099.360000	2012.000000	12.000000
std	12.785297	30.492054	NaN	22711.183519	0.796876	3.243217

```
sales_features_df.describe()
```

	Store	Dept	Date	Weekly_Sales	Year	Month	Temperature	Fuel_Price	M
count	421570.000000	421570.000000	421570	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000	421
mean	22.200546	44.260317	2011-06-18 08:30:31.963375104	15981.258123	2010.968591	6.449510	60.090059	3.361027	2
min	1.000000	1.000000	2010-02-05 00:00:00	-4988.940000	2010.000000	1.000000	-2.060000	2.472000	
25%	11.000000	18.000000	2010-10-08 00:00:00	2079.650000	2010.000000	4.000000	46.680000	2.933000	
50%	22.000000	37.000000	2011-06-17 00:00:00	7612.030000	2011.000000	6.000000	62.090000	3.452000	
75%	33.000000	74.000000	2012-02-24 00:00:00	20205.852500	2012.000000	9.000000	74.280000	3.738000	2

```
# Duplicates in sales_features_df
sales_features_df.duplicated().sum()
```

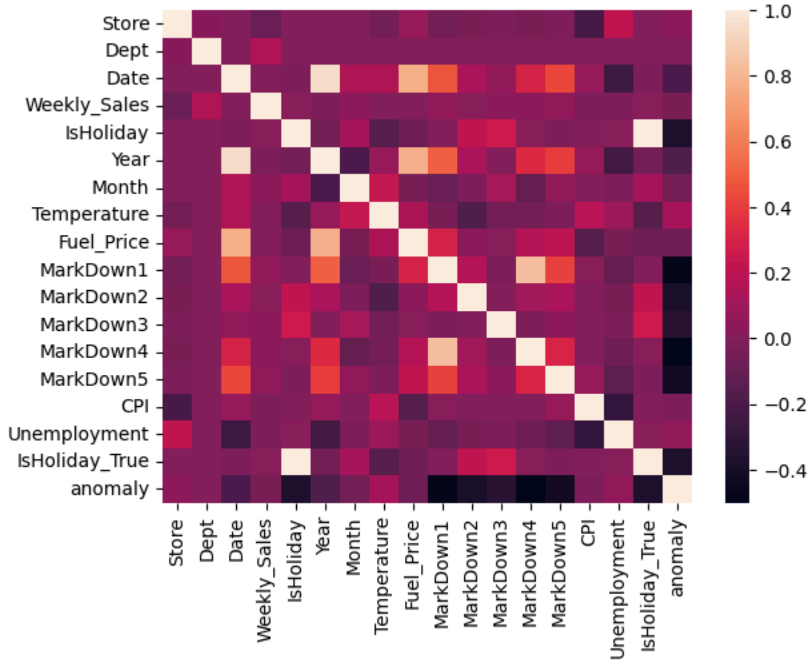
```
np.int64(0)
```

```
# Show co-relation between weekly_sales and Month, temperature,IsHoliday_True, CPI and Unemployment
sales_features_df.corr()
```

	Store	Dept	Date	Weekly_Sales	IsHoliday	Year	Month	Temperature	Fuel_Price	MarkDown1
Store	1.000000	0.024004	0.003362	-0.085195	-0.000548	0.002997	0.001011	-0.050097	0.065290	-0.059844
Dept	0.024004	1.000000	0.004054	0.148032	0.000916	0.003738	0.000904	0.004437	0.003572	0.001494
Date	0.003362	0.004054	1.000000	-0.000663	-0.013017	0.941467	0.146422	0.147064	0.771913	0.470865
Weekly_Sales	-0.085195	0.148032	-0.000663	1.000000	0.012774	-0.010111	0.028409	-0.002312	-0.000120	0.047172
IsHoliday	-0.000548	0.000916	-0.013017	0.012774	1.000000	-0.056746	0.123376	-0.155949	-0.078281	-0.003521
Year	0.002997	0.003738	0.941467	-0.010111	-0.056746	1.000000	-0.194288	0.065814	0.779633	0.501044
Month	0.001011	0.000904	0.146422	0.028409	0.123376	-0.194288	1.000000	0.235983	-0.040876	-0.089206
Temperature	-0.050097	0.004437	0.147064	-0.002312	-0.155949	0.065814	0.235983	1.000000	0.143859	-0.026415
Fuel_Price	0.065290	0.003572	0.771913	-0.000120	-0.078281	0.779633	-0.040876	0.143859	1.000000	0.297056
MarkDown1	-0.059844	0.001494	0.470865	0.047172	-0.003521	0.501044	-0.089206	-0.026415	0.297056	1.000000
MarkDown2	-0.033829	0.000587	0.127975	0.020716	0.207604	0.131867	-0.019360	-0.179672	0.029153	0.174868
MarkDown3	-0.020331	0.001475	0.048749	0.038562	0.266471	0.006789	0.116031	-0.056026	0.018615	-0.014411
MarkDown4	-0.042724	0.001937	0.297472	0.037467	0.011565	0.335340	-0.105569	-0.050281	0.166622	0.838904
MarkDown5	-0.012452	0.002668	0.423599	0.050465	-0.015235	0.402964	0.055770	-0.014752	0.215420	0.415050
CPI	-0.211088	-0.007477	0.077001	-0.020921	-0.001944	0.074544	0.005282	0.182112	-0.164210	0.010915
Unemployment	0.208552	0.007837	-0.243370	-0.025864	0.010460	-0.237161	-0.012444	0.096730	-0.033853	-0.105168
IsHoliday_True	-0.000548	0.000916	-0.013017	0.012774	1.000000	-0.056746	0.123376	-0.155949	-0.078281	-0.003521
anomaly	0.037602	-0.001815	-0.196483	-0.038819	-0.376093	-0.177279	-0.057118	0.124630	-0.072092	-0.486622

```
# Craete a sns graph of this corr .

sns.heatmap(sales_features_df.corr())

# make the graph big
plt.figure(figsize=(20, 16))
```

<Figure size 2000x1600 with 0 Axes>

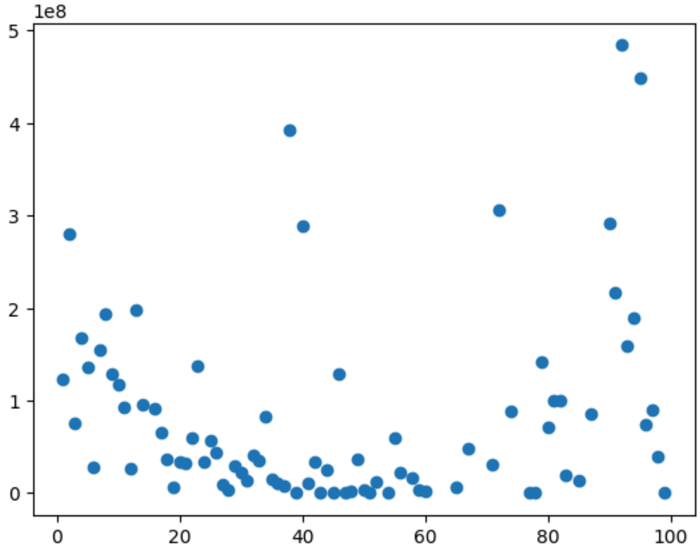


<Figure size 2000x1600 with 0 Axes>

Analysis of data

```
# Sum weekly sales per dept
sales_features_df.groupby('Dept')['Weekly_Sales'].sum().sort_values(ascending=False)
# put it in a scatter plot
plt.scatter(sales_features_df.groupby('Dept')['Weekly_Sales'].sum().sort_values(ascending=False).index, sales_features_df.gro
```

<matplotlib.collections.PathCollection at 0x7a3c72476290>




```
# merge sales and stores data on store
sales_stores_df = pd.merge(sales_data_df, stores_df, on='Store', how='inner')
sales_stores_df.head()
```

	Store	Dept	Date	Weekly_Sales	IsHoliday	Year	Month	Type	Size
0	1	1	2010-02-05	24924.50	False	2010	2	A	151315
1	1	1	2010-02-12	46039.49	True	2010	2	A	151315
2	1	1	2010-02-19	41595.55	False	2010	2	A	151315
3	1	1	2010-02-26	19403.54	False	2010	2	A	151315
4	1	1	2010-03-05	21827.90	False	2010	3	A	151315

```
# marge stores data with sales_features_df
sales_features_stores_df = pd.merge(sales_features_df, stores_df, on='Store', how='inner')
sales_features_stores_df.head()
```

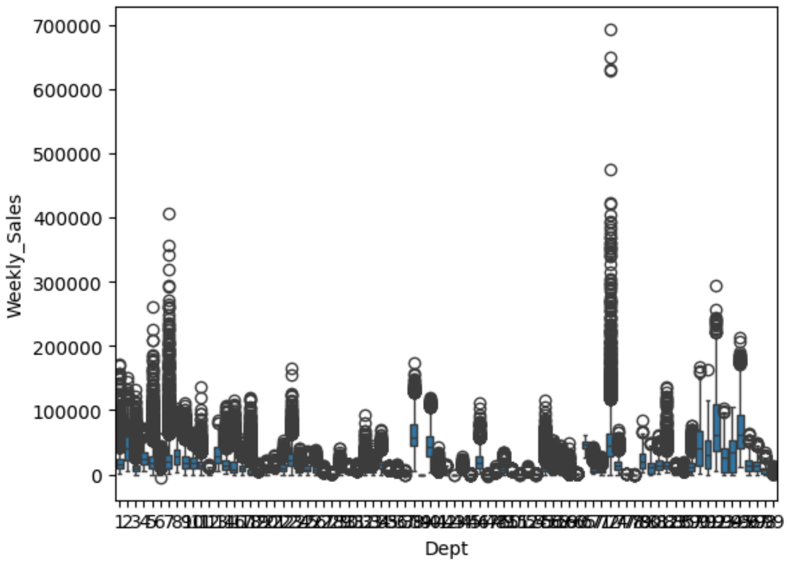
	Store	Dept	Date	Weekly_Sales	IsHoliday	Year	Month	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkD
0	1	1	2010-02-05	24924.50	False	2010	2	42.31	2.572	0.0	0.0	0.0	
1	1	1	2010-02-12	46039.49	True	2010	2	38.51	2.548	0.0	0.0	0.0	
2	1	1	2010-02-19	41595.55	False	2010	2	39.93	2.514	0.0	0.0	0.0	

3. ***Data Wrangling***

Data Wrangling Code

```
# show bi-variate relationship with Weekly_Sales and Dept
import seaborn as sns
sns.boxplot(x='Dept', y='Weekly_Sales', data=sales_features_stores_df)
```

<Axes: xlabel='Dept', ylabel='Weekly_Sales'>



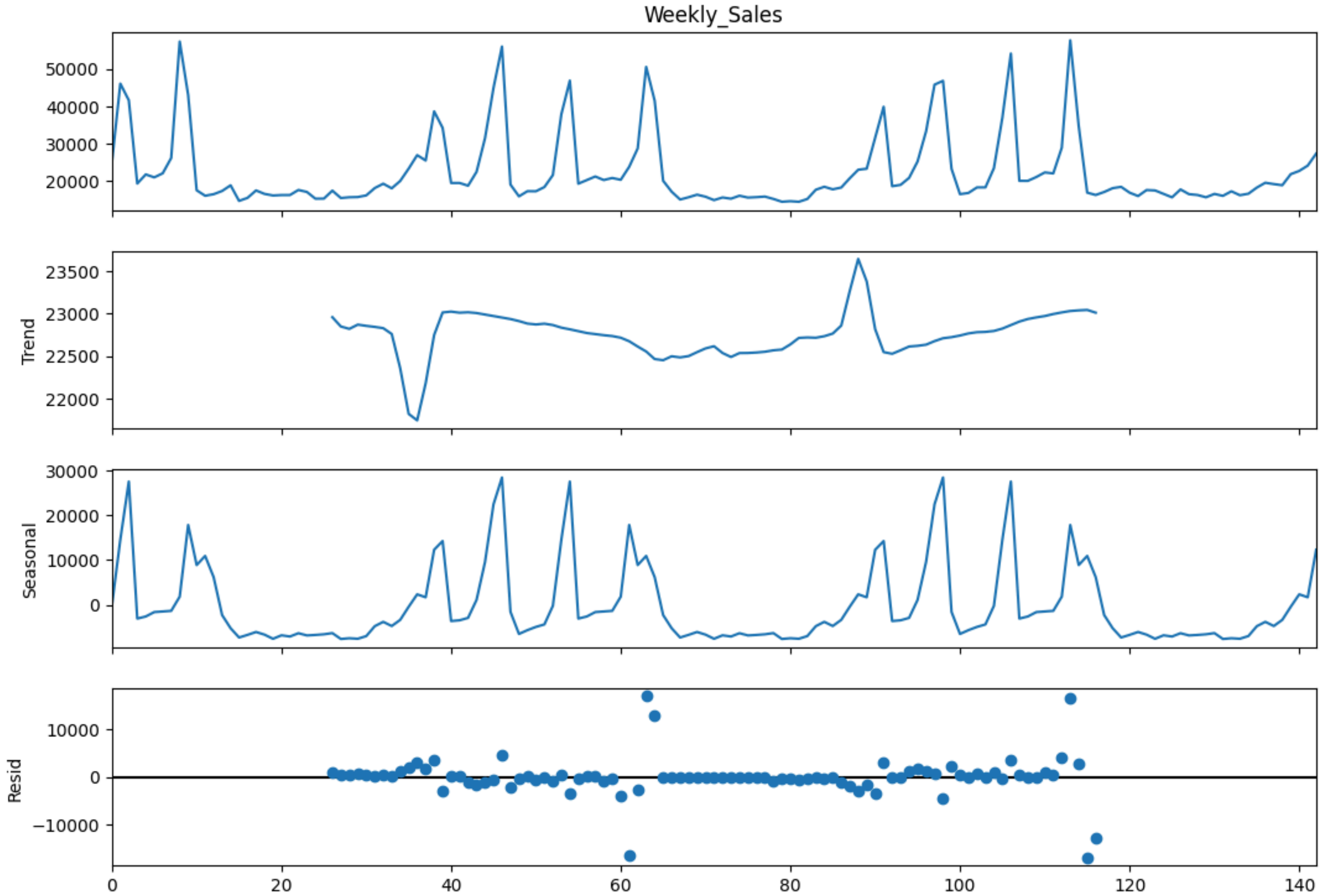

```
# sort sales_data_df on Date
sales_data_df.sort_values(by='Date', inplace=True)
sales_data_df.head()
```

	Store	Dept	Date	Weekly_Sales	IsHoliday	Year	Month
0	1	1	2010-02-05	24924.50	False	2010	2
277665	29	5	2010-02-05	15552.08	False	2010	2
277808	29	6	2010-02-05	3200.22	False	2010	2
277951	29	7	2010-02-05	10820.05	False	2010	2
278094	29	8	2010-02-05	20055.64	False	2010	2

Analyse Seaonality and Trends

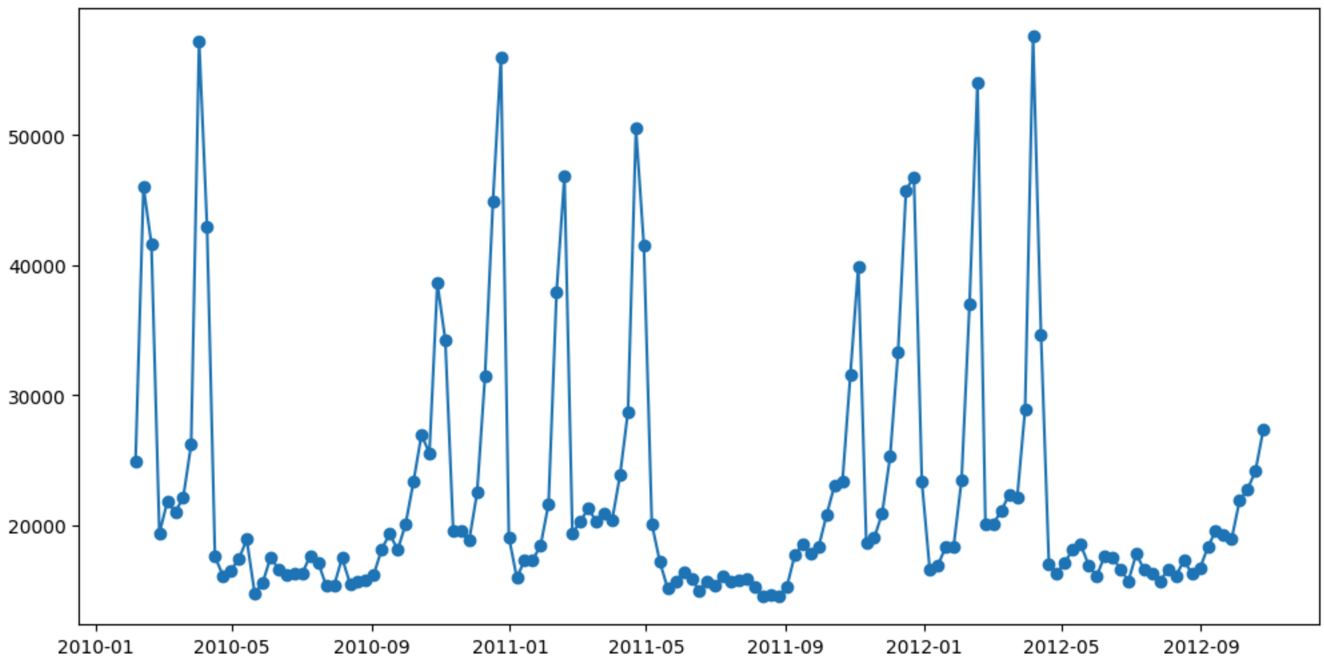
```
from statsmodels.tsa.seasonal import seasonal_decompose
sales_data_df_store_1_dept_1 = sales_data_df[sales_data_df['Store'] == 1][sales_data_df['Dept'] == 1]
#remove duplicates from sales_df_store_1_dept_1
sales_data_df_store_1_dept_1 = sales_data_df_store_1_dept_1.drop_duplicates(subset=['Date'])
#sort on date
sales_data_df_store_1_dept_1 = sales_data_df_store_1_dept_1.sort_values(by='Date')

decomposition = seasonal_decompose(
    sales_data_df_store_1_dept_1['Weekly_Sales'],
    model='additive',  # Use 'multiplicative' if seasonality grows with trend
    period=52  # Weekly data → yearly seasonality (52 weeks)
)

# Plot decomposition
fig = decomposition.plot()
fig.set_size_inches(12, 8)
plt.show()
```

```
/tmp/ipython-input-17-1292653360.py:2: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  sales_data_df_store_1_dept_1 = sales_data_df[sales_data_df['Store'] == 1][sales_data_df['Dept'] == 1]
```




```
# plot line plot of Weekly_sales and Date
plt.figure(figsize=(12, 6))
plt.plot(sales_data_df_store_1_dept_1['Date'], sales_data_df_store_1_dept_1['Weekly_Sales'], marker='o', linestyle='-')
```

```
[<matplotlib.lines.Line2D at 0x78c13a508810>]
```




```
sales_data_df_store_1_dept_1.describe()
```

	Store	Dept	Date	Weekly_Sales	Year	Month
count	143.0	143.0	143	143.000000	143.000000	143.000000
mean	1.0	1.0	2011-06-17 00:00:00	22513.322937	2010.965035	6.447552
min	1.0	1.0	2010-02-05 00:00:00	14537.370000	2010.000000	1.000000
25%	1.0	1.0	2010-10-11 12:00:00	16494.630000	2010.000000	4.000000
50%	1.0	1.0	2011-06-17 00:00:00	18535.480000	2011.000000	6.000000
75%	1.0	1.0	2012-02-20 12:00:00	23214.215000	2012.000000	9.000000
max	1.0	1.0	2012-10-26 00:00:00	57592.120000	2012.000000	12.000000
std	0.0	0.0	NaN	9854.349032	0.799759	3.249438

```
sales_features_stores_df.describe()
```

	Store	Dept	Date	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2
count	421570.000000	421570.000000	421570	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000
mean	22.200546	44.260317	2011-06-18 08:30:31.963375104	15981.258123	60.090059	3.361027	2590.074819	879.974298
min	1.000000	1.000000	2010-02-05 00:00:00	-4988.940000	-2.060000	2.472000	0.000000	-265.760000
25%	11.000000	18.000000	2010-10-08 00:00:00	2079.650000	46.680000	2.933000	0.000000	0.000000
50%	22.000000	37.000000	2011-06-17 00:00:00	7612.030000	62.090000	3.452000	0.000000	0.000000
75%	[illegible]	[illegible]	2012-02-24	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

```
sales_features_stores_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 421570 entries, 0 to 421569
Data columns (total 22 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   Year            421570 non-null  int32
 6   Month           421570 non-null  int32
 7   Temperature     421570 non-null  float64
 8   Fuel_Price      421570 non-null  float64
 9   MarkDown1       421570 non-null  float64
```

```
 10  MarkDown2       421570 non-null  float64
 11  MarkDown3       421570 non-null  float64
 12  MarkDown4       421570 non-null  float64
 13  MarkDown5       421570 non-null  float64
 14  CPI             421570 non-null  float64
 15  Unemployment    421570 non-null  float64
 16  IsHoliday_True  421570 non-null  int64
 17  anomaly         421570 non-null  int64
 18  Type            421570 non-null  object
 19  Size            421570 non-null  int64
 20  Dept_Type       421570 non-null  float64
 21  Cluster         421570 non-null  int32
dtypes: bool(1), datetime64[ns](1), float64(11), int32(3), int64(5), object(1)
memory usage: 63.1+ MB
```

Start coding or generate with AI.

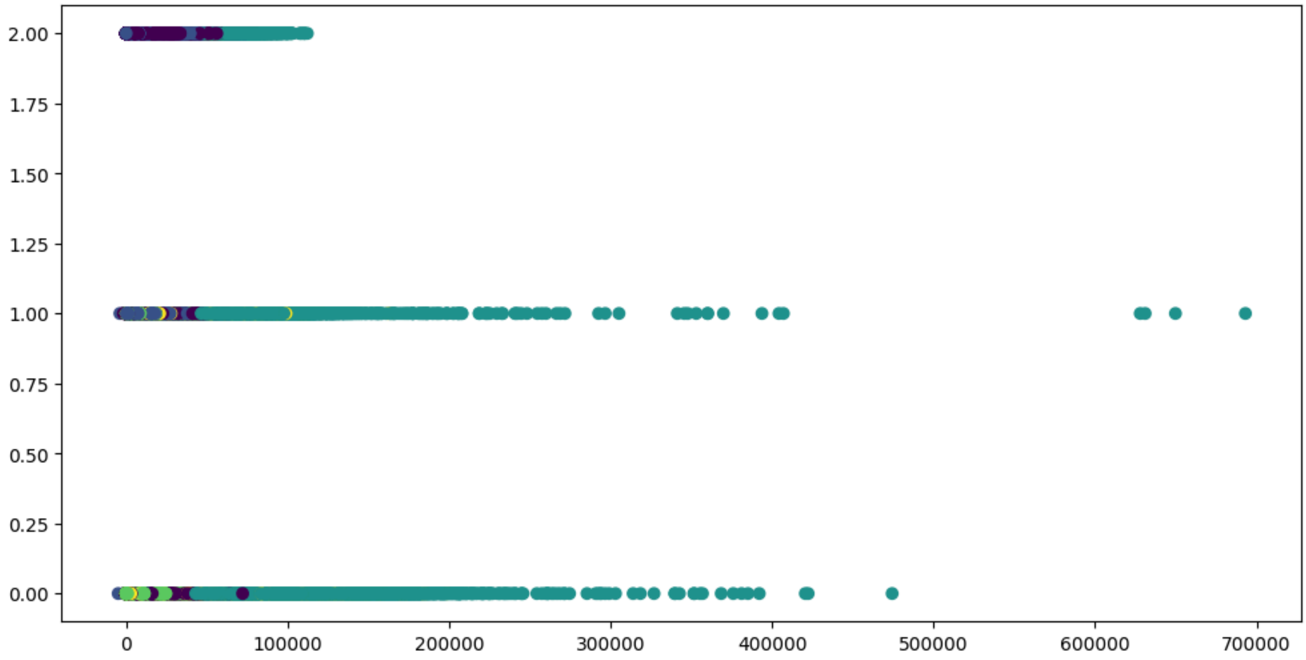
Customer Segmentaion - Department and Store segmentaions

```
# find distinct Type
sales_features_stores_df['Type'].unique()
```

```
array(['A', 'B', 'C'], dtype=object)
```

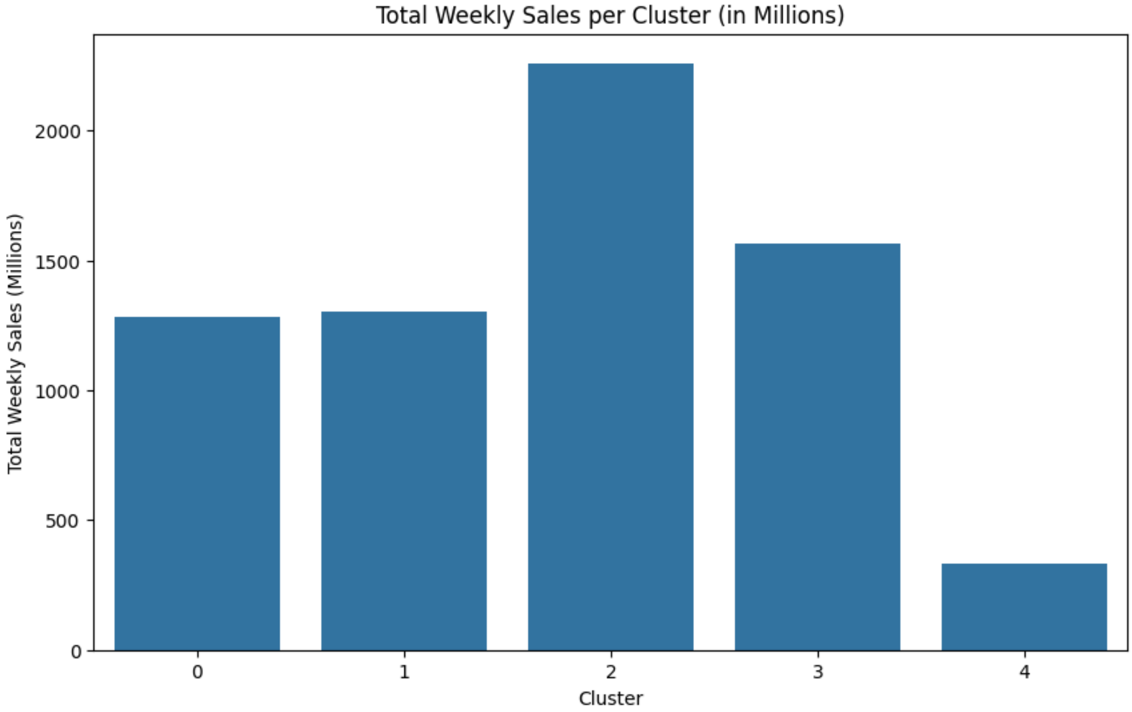
```
# Implement K-Means unsupervised clustering on sales_features_store_df df
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt


# LabelEncoder for Deprtment Type, year and Month columns
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
sales_features_stores_df['Dept_Type'] = label_encoder.fit_transform(sales_features_stores_df['Type'])
sales_features_stores_df['Lab_Year'] = label_encoder.fit_transform(sales_features_stores_df['Year'])
sales_features_stores_df['Lab_Month'] = label_encoder.fit_transform(sales_features_stores_df['Month'])

sales_features_stores_scaled_df = sales_features_stores_df.copy()

# Scale the data
scaler = StandardScaler()
scaled_features = ['Weekly_Sales', 'Temperature', 'Fuel_Price', 'MarkDown1', 'MarkDown2','MarkDown3','MarkDown4','MarkDown5'
sales_features_stores_scaled_df[scaled_features] = scaler.fit_transform(sales_features_stores_df[scaled_features])


segment_model = KMeans(n_clusters=5, random_state=42)
segment_model.fit(sales_features_stores_scaled_df[['Weekly_Sales', 'Temperature', 'Fuel_Price', 'MarkDown1', 'MarkDown2','Ma

# Evaluate and plot the clusters
sales_features_stores_df['Cluster'] = segment_model.labels_
sales_features_stores_df.head()

# Plot a b
plt.figure(figsize=(12, 6))
plt.scatter(sales_features_stores_df['Weekly_Sales'],sales_features_stores_df['Dept_Type'], c=sales_features_stores_df['Clus
```

```
<matplotlib.collections.PathCollection at 0x7a3c70b131d0>
```




```
# group Dept with cluster and total weekly_sales
result_df = sales_features_stores_df.groupby('Cluster')['Weekly_Sales'].sum().reset_index()
result_df['Weekly_sales_MN'] = result_df['Weekly_Sales'] / 1000000

# Display the result
display(result_df)

# Plot the total weekly sales per cluster
plt.figure(figsize=(10, 6))
sns.barplot(data=result_df, x='Cluster', y='Weekly_sales_MN')
plt.title('Total Weekly Sales per Cluster (in Millions)')
plt.xlabel('Cluster')
plt.ylabel('Total Weekly Sales (Millions)')
plt.show()
```

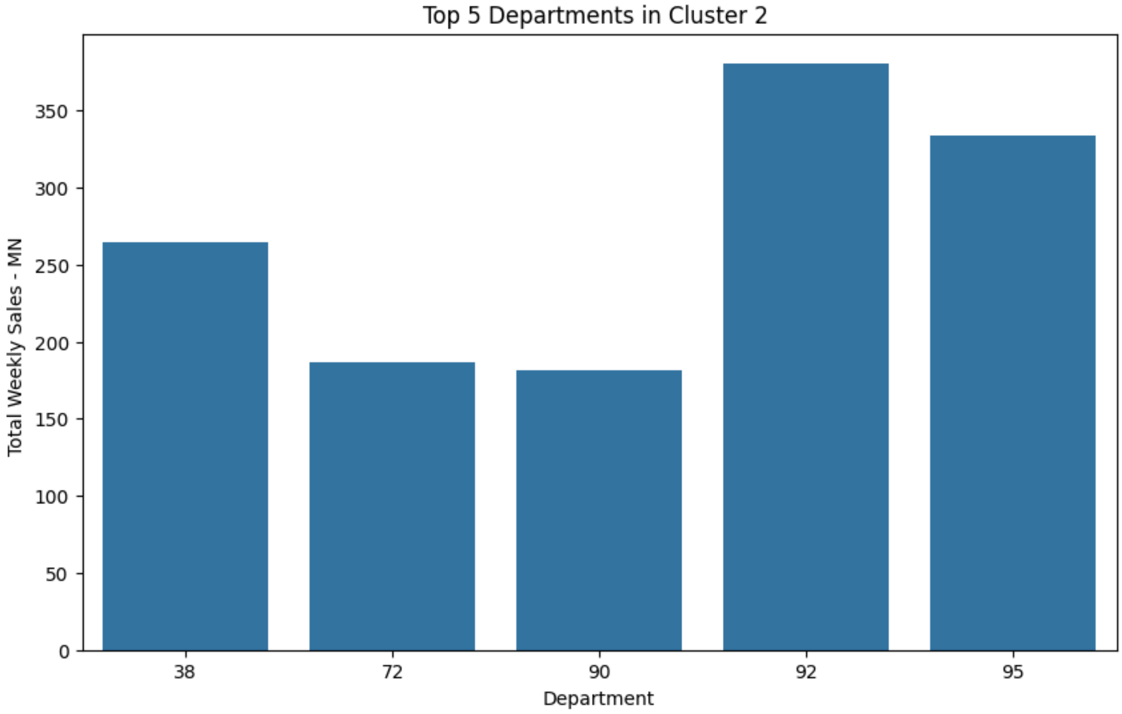
	Cluster	Weekly_Sales	Weekly_sales_MN
0	0	1.279643e+09	1279.643213
1	1	1.303520e+09	1303.519779
2	2	2.259100e+09	2259.099696
3	3	1.562810e+09	1562.809907
4	4	3.321464e+08	332.146392



Next steps: Generate code with result_df | View recommended plots | New interactive sheet

```
# Top 5 depts in Cluster 2
dept_df= sales_features_stores_df[sales_features_stores_df['Cluster'] == 2].groupby('Dept')['Weekly_Sales'].sum().sort_values
weekly_sales_df = dept_df.values/1000000

# Plot this as a bar chart
plt.figure(figsize=(10, 6))
sns.barplot(x=dept_df.index, y=weekly_sales_df)
plt.title('Top 5 Departments in Cluster 2 ')
plt.xlabel('Department')
plt.ylabel('Total Weekly Sales - MN ')
```

Text(0, 0.5, 'Total Weekly Sales - MN ')



T-SNE CLustering and Plot

```
from sklearn.manifold import TSNE

tsne = TSNE(n_components=5, perplexity=30, random_state=42)
tsne_result = tsne.fit_transform(sales_features_stores_df[['Weekly_Sales', 'Temperature', 'Fuel_Price', 'MarkDown1', 'MarkDo
                                                            'MarkDown3', 'MarkDown4', 'MarkDown5', 'CPI', 'Unemployment', 'D
sales_features_stores_df['TSNE1'] = tsne_result[:, 0]
sales_features_stores_df['TSNE2'] = tsne_result[:, 1]

plt.figure(figsize=(10, 6))
sns.scatterplot(data=sales_features_stores_df, x='TSNE1', y='TSNE2', hue='Cluster', palette='tab10')
plt.title('KMeans Clusters (t-SNE Reduced)')
plt.show()
```

Demand Forcasting

**4. Prepare for Time Series Model - State Space Model : STS **

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.statespace.structural import UnobservedComponents
from sklearn.preprocessing import StandardScaler # Import StandardScaler

# Load data (replace with your DataFrame name)
df = sales_features_stores_df.copy()

# 1. DATA PREPARATION
# Filter for Store 1, Dept 1
sales_feature_store_1_Dept_1 = df.loc[(df['Store'] == 1) & (df['Dept'] == 1)]

# Convert Date to datetime and set as index
sales_feature_store_1_Dept_1['Date'] = pd.to_datetime(sales_feature_store_1_Dept_1['Date'])
# Drop duplicate dates before setting as index
sales_feature_store_1_Dept_1.drop_duplicates(subset=['Date'], inplace=True)
sales_feature_store_1_Dept_1.set_index('Date', inplace=True)
sales_feature_store_1_Dept_1.sort_index(inplace=True)

# Resample to weekly frequency and fill missing values (e.g., with 0 or forward fill)
# Filling with 0 assumes no sales on missing weeks, which might be appropriate depending on the context.
#store_dept_df_resampled = store_dept_df.resample('W').asfreq().fillna(0)
```

```
sales_feature_store_1_Dept_1.describe()
```

```
/tmp/ipython-input-152-1768717614.py:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
  sales_feature_store_1_Dept_1['Date'] = pd.to_datetime(sales_feature_store_1_Dept_1['Date'])
/tmp/ipython-input-152-1768717614.py:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
  sales_feature_store_1_Dept_1.drop_duplicates(subset=['Date'], inplace=True)
```

	Store	Dept	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	
count	143.0	143.0	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	1
mean	1.0	1.0	22513.322937	68.306783	3.219699	2885.518042	863.882867	428.461678	1336.016224	1789.869930	2
std	0.0	0.0	9854.349032	14.250486	0.427313	5522.291795	4481.220951	4665.898328	3989.624600	3100.360433	
min	1.0	1.0	14537.370000	35.400000	2.514000	0.000000	0.000000	0.000000	0.000000	0.000000	2
25%	1.0	1.0	16494.630000	58.265000	2.764500	0.000000	0.000000	0.000000	0.000000	0.000000	2
50%	1.0	1.0	18535.480000	69.640000	3.290000	0.000000	0.000000	0.000000	0.000000	0.000000	2
75%	1.0	1.0	23214.215000	80.485000	3.594000	4469.405000	10.740000	7.480000	827.540000	3330.625000	2
max	1.0	1.0	57592.120000	91.650000	3.907000	34577.060000	46011.380000	55805.510000	32403.870000	20475.320000	2

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.statespace.structural import UnobservedComponents
from sklearn.preprocessing import StandardScaler # Import StandardScaler

# Load data (replace with your DataFrame name)
df = sales_features_stores_df

# 1. DATA PREPARATION
# Filter for Store 1, Dept 1
store_dept_df = df[(df['Store'] == 1) & (df['Dept'] == 1)].copy()


# Select features: Target + Exogenous Variables
exog_vars = ['IsHoliday_True', 'Temperature', 'Fuel_Price',
             'MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5', 'CPI', 'Unemployment'


target = 'Weekly_Sales'

# Scale exogenous variables
scaler = StandardScaler()
store_dept_df[exog_vars] = scaler.fit_transform(store_dept_df[exog_vars])

# Split into train/test (last 12 weeks for testing)
test_size = 18
train = store_dept_df.iloc[:-test_size]
test = store_dept_df.iloc[-test_size:]

print("Train Shape:", train.shape)
print("Test Shape:", test.shape)

# 2. MODEL TRAINING
# Initialize STS model with components:
# - Stochastic level (llevel)
# - Deterministic linear trend
# - Yearly seasonality (52 weeks)
# - Exogenous regressors
model = UnobservedComponents(
    endog=train[target],
    exog=train[exog_vars],
    level='llevel',
    trend=True,
    seasonal=52
)

# Fit model (disable output with disp=0)
results = model.fit(disp=0)
print(results.summary())

# 3. FORECASTING
# Forecast next 12 weeks using test-set exogenous variables
```

```
forecast = results.get_forecast(
    steps=test_size,
    exog=test[exog_vars]
)
forecast_mean = forecast.predicted_mean
conf_int = forecast.conf_int()

# 4. VISUALIZATION
plt.figure(figsize=(12, 6))
plt.plot(train.index, train[target], label='Train')
plt.plot(test.index, test[target], label='Actual', color='green')
plt.plot(forecast_mean.index, forecast_mean, label='Forecast', color='red')
plt.fill_between(conf_int.index, conf_int.iloc[:,0], conf_int.iloc[:,1], alpha=0.2)
plt.title('STS Forecast: Weekly Sales (Store 1, Dept 1)')
plt.legend()
plt.show()

# 5. EVALUATION
from sklearn.metrics import mean_absolute_error, mean_squared_error

mae = mean_absolute_degree = mean_absolute_error(test[target], forecast_mean)
rmse = np.sqrt(mean_squared_error(test[target], forecast_mean))
print(f"MAE: {mae:.2f}, RMSE: {rmse:.2f}")
```

```
Train Shape: (125, 18)
Test Shape: (18, 18)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/statespace/structural.py:426: SpecificationWarning: Value of `t
  warn("Value of `%s` may be overridden when the trend"
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/filters/hp_filter.py:100: SparseEfficiencyWarning: spsolve requ
  trend = spsolve(I+lamb*K.T.dot(K), x, use_umfpack=use_umfpack)
                            Unobserved Components Results
=====================================================================================
Dep. Variable:                 Weekly_Sales   No. Observations:                  125
Model:                          local level   Log Likelihood                -770.511
                   + stochastic seasonal(52)   AIC                           1567.023
Date:                      Fri, 25 Jul 2025   BIC                           1596.799
Time:                              16:24:38   HQIC                          1578.889
Sample:                                   0
                                      - 125
Covariance Type:                        opg
=======================================================================================
                          coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------------
sigma2.irregular     5.678e+07   1.83e+07      3.102      0.002    2.09e+07    9.27e+07
sigma2.level         4.044e+06    4.6e+06      0.878      0.380   -4.98e+06    1.31e+07
sigma2.seasonal      2.944e+06   3.04e+06      0.970      0.332   -3.01e+06    8.89e+06
beta.IsHoliday_True   281.9659   6063.638      0.047      0.963   -1.16e+04    1.22e+04
beta.Temperature    -2724.4657   3033.268     -0.898      0.369   -8669.562    3220.631
beta.Fuel_Price       503.4676   4674.347      0.108      0.914   -8658.084    9665.019
beta.MarkDown1        281.3994   3422.209      0.082      0.934   -6426.007    6988.806
beta.MarkDown2       -749.5586   1579.327     -0.475      0.635   -3844.982    2345.865
beta.MarkDown3       -689.6742   2810.170     -0.245      0.806   -6197.507    4818.159
beta.MarkDown4       -148.9296   2983.708     -0.050      0.960   -5996.890    5699.031
beta.MarkDown5       -720.6744   1942.306     -0.371      0.711   -4527.524    3086.175
beta.CPI             2747.3569   1.21e+04      0.228      0.820   -2.09e+04    2.64e+04
beta.Unemployment    1605.7073   4026.913      0.399      0.690   -6286.896    9498.311
===================================================================================
Ljung-Box (L1) (Q):                  11.03   Jarque-Bera (JB):                53.66
Prob(Q):                              0.00   Prob(JB):                         0.00
Heteroskedasticity (H):               1.14   Skew:                             1.48
Prob(H) (two-sided):                  0.75   Kurtosis:                         5.98
===================================================================================
```