Problem Statement 1:

A retail store that has multiple outlets across the country are facing issues in managing the inventory - to match the demand with respect to supply. You are a data scientist, who has to come up with useful insights using the data and make prediction models to forecast the sales for X number of months/years.

Dataset Information: The walmart.csv contains 6435 rows and 8 columns

1. Using the above data, come up with useful insights that can be used by each of the stores to improve in various areas.
2. Forecast the sales for each store for the next 12 weeks

Problem Statement 2:

An online retail store is trying to understand the various customer purchase patterns for their firm, you are required to give enough evidence based insights to provide the same.

Dataset Information: The online_retail.csv contains 387961 rows and 8 columns

1. Using the above data, find useful insights about the customer purchasing history that can be an added advantage for the online retailer.
2. Segment the customers based on their purchasing behavior

```
1 from google.colab import files
2 uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
1 import pandas as pd
2 df = pd.read_csv("Walmart (1).csv")
3 df.head()
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         6435 non-null   int64
 1   Date          6435 non-null   object
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
dtypes: float64(5), int64(2), object(1)
memory usage: 402.3+ KB
```

```
1 df.describe()
```

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
count	6435.000000	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000
mean	23.000000	1.046965e+06	0.069930	60.663782	3.358607	171.578394	7.999151
std	12.988182	5.643666e+05	0.255049	18.444933	0.459020	39.356712	1.875885
min	1.000000	2.099862e+05	0.000000	-2.060000	2.472000	126.064000	3.879000
25%	12.000000	5.533501e+05	0.000000	47.460000	2.933000	131.735000	6.891000
50%	23.000000	9.607460e+05	0.000000	62.670000	3.445000	182.616521	7.874000
75%	34.000000	1.420159e+06	0.000000	74.940000	3.735000	212.743293	8.622000
max	45.000000	3.818686e+06	1.000000	100.140000	4.468000	227.232807	14.313000

```
1 df.isnull().sum()
```

	0
Store	0
Date	0
Weekly_Sales	0
Holiday_Flag	0
Temperature	0
Fuel_Price	0
CPI	0
Unemployment	0

dtype: int64

```
1 df['Date'] = pd.to_datetime(df['Date'], dayfirst=True)
2 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         6435 non-null   int64
 1   Date          6435 non-null   datetime64[ns]
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
dtypes: datetime64[ns](1), float64(5), int64(2)
memory usage: 402.3 KB
```

```
1 df = df.sort_values("Date")
2 df.head()
3 df.tail()
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
3574	25	2012-10-26	688940.94	0	56.69	3.882	216.151590	7.293
714	5	2012-10-26	319550.77	0	71.70	3.506	224.037814	5.422
5719	40	2012-10-26	921264.52	0	49.65	3.917	138.728161	4.145
2573	18	2012-10-26	1127516.25	0	56.09	3.917	138.728161	8.243
6434	45	2012-10-26	760281.43	0	58.85	3.882	192.308899	8.667

```
1 df = df.sort_values(by=["Date", "Store"], ascending=[True, True])
2 df.head()
3 df.tail()
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
5862	41	2012-10-26	1316542.59	0	41.80	3.686	199.219532	6.195
6005	42	2012-10-26	514756.08	0	70.50	4.301	131.193097	6.943
6148	43	2012-10-26	587603.55	0	69.17	3.506	214.741539	8.839
6291	44	2012-10-26	361067.07	0	46.97	3.755	131.193097	5.217
6434	45	2012-10-26	760281.43	0	58.85	3.882	192.308899	8.667

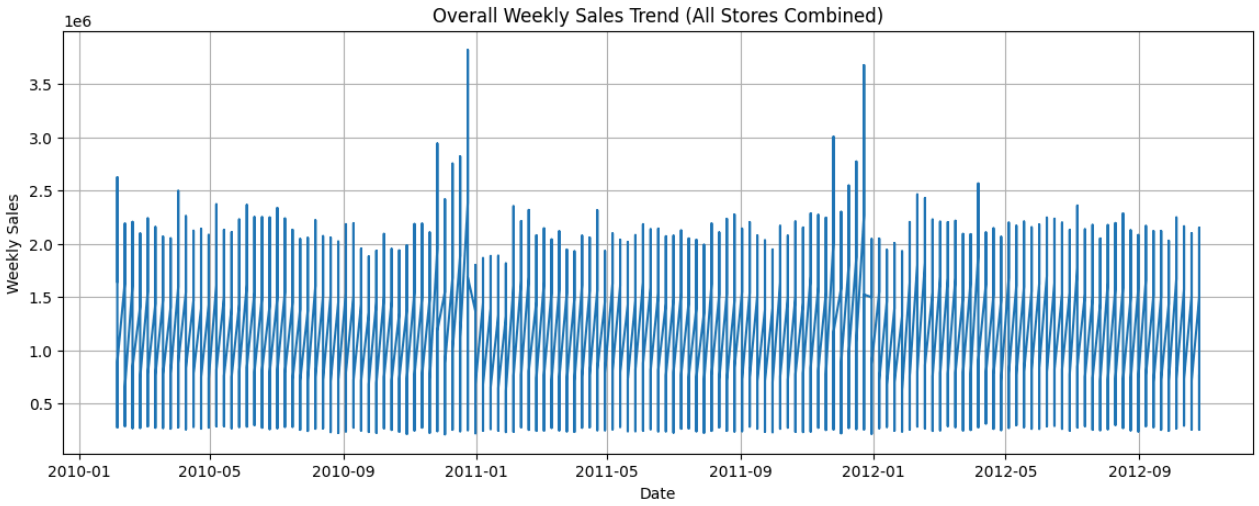
```
1 df['Date'].min(), df['Date'].max()
```

```
(Timestamp('2010-02-05 00:00:00'), Timestamp('2012-10-26 00:00:00'))
```

```
1 df = df.sort_values(by=["Date", "Store"], ascending=[True, True])
2 print("Earliest date:", df['Date'].head(5))
3 print("Latest date:", df['Date'].tail(5))
```

```
Earliest date: 0     2010-02-05
143   2010-02-05
286   2010-02-05
429   2010-02-05
572   2010-02-05
Name: Date, dtype: datetime64[ns]
Latest date: 5862   2012-10-26
6005   2012-10-26
6148   2012-10-26
6291   2012-10-26
6434   2012-10-26
Name: Date, dtype: datetime64[ns]
```

```
1 import matplotlib.pyplot as plt
2
3 plt.figure(figsize=(14,5))
4 plt.plot(df['Date'], df['Weekly_Sales'])
5 plt.title("Overall Weekly Sales Trend (All Stores Combined)")
6 plt.xlabel("Date")
7 plt.ylabel("Weekly Sales")
8 plt.grid(True)
9 plt.show()
```

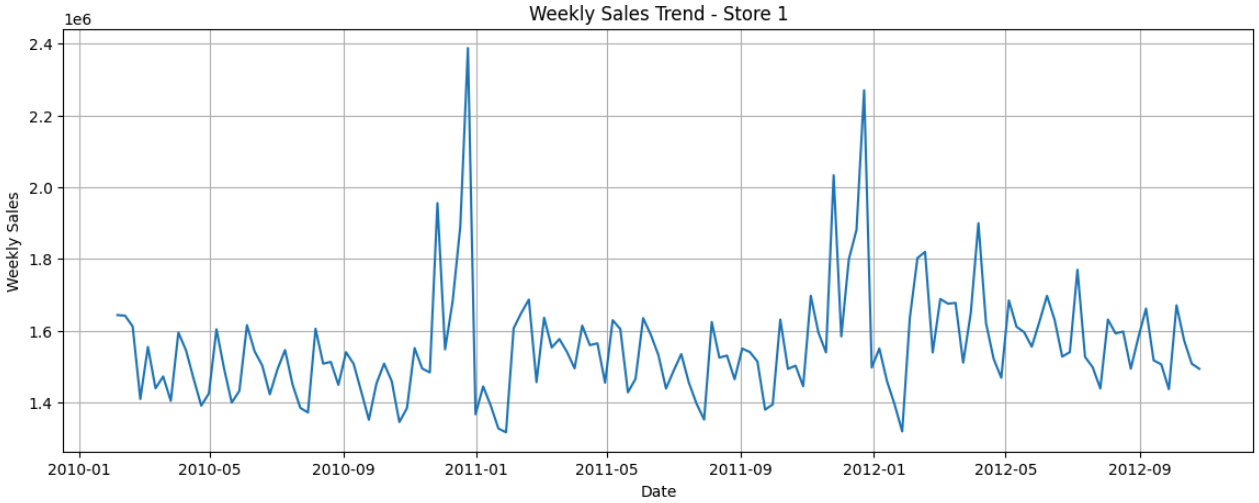



```
1 store1 = df[df['Store'] == 1]
2 store1 = store1.sort_values("Date")
3 store1_ts = store1.set_index("Date")["Weekly_Sales"]
4 store1_ts.head()
```

Date	Weekly_Sales
2010-02-05	1643690.90
2010-02-12	1641957.44
2010-02-19	1611968.17
2010-02-26	1409727.59
2010-03-05	1554806.68

dtype: float64

```
plt.figure(figsize=(14,5))
plt.plot(store1_ts)
plt.title("Weekly Sales Trend - Store 1")
plt.xlabel("Date")
plt.ylabel("Weekly Sales")
plt.grid(True)
plt.show()
```




```
!pip install pmdarima
```

```
Collecting pmdarima
  Downloading pmdarima-2.1.1-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl.metadata (8.5
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.12/dist-packages (from pmdarima) (1.5.2)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/python3.12/dist-packages (from pmdarima) (
Requirement already satisfied: numpy>=1.21.6 in /usr/local/lib/python3.12/dist-packages (from pmdarima) (2.0.2)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.12/dist-packages (from pmdarima) (2.2.2)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.12/dist-packages (from pmdarima) (1.6.1)
Requirement already satisfied: scipy>=1.13.0 in /usr/local/lib/python3.12/dist-packages (from pmdarima) (1.16.3)
Requirement already satisfied: statsmodels>=0.14.5 in /usr/local/lib/python3.12/dist-packages (from pmdarima) (0.14.5)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.12/dist-packages (from pmdarima) (2.5.0)
Requirement already satisfied: setuptools!=50.0.0,>=42 in /usr/local/lib/python3.12/dist-packages (from pmdarima) (75.2.0)
Requirement already satisfied: packaging>=17.1 in /usr/local/lib/python3.12/dist-packages (from pmdarima) (25.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.19->pmdarim
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.19->pmdarima) (2025.2
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.19->pmdarima) (2025
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn>=0.22->pmd
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.12/dist-packages (from statsmodels>=0.14.5->pmdarima)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas>=0.1
Downloading pmdarima-2.1.1-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl (689 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 689.1/689.1 kB 22.2 MB/s eta 0:00:00
Installing collected packages: pmdarima
Successfully installed pmdarima-2.1.1
```

```
from pmdarima import auto_arima

model = auto_arima(
    store1_ts,
    seasonal=True,
    m=52,   # 52 weeks = 1 year seasonality
    trace=True,
    error_action="ignore",
```

```
    suppress_warnings=True
)

model
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[52] intercept   : AIC=3786.582, Time=7.38 sec
 ARIMA(0,1,0)(0,0,0)[52] intercept   : AIC=3850.005, Time=0.05 sec
 ARIMA(1,1,0)(1,0,0)[52] intercept   : AIC=3795.624, Time=1.56 sec
 ARIMA(0,1,1)(0,0,1)[52] intercept   : AIC=3798.433, Time=1.18 sec
 ARIMA(0,1,0)(0,0,0)[52]             : AIC=3848.013, Time=0.03 sec
 ARIMA(2,1,2)(0,0,1)[52] intercept   : AIC=3795.311, Time=5.01 sec
 ARIMA(2,1,2)(1,0,0)[52] intercept   : AIC=3786.204, Time=3.13 sec
 ARIMA(2,1,2)(0,0,0)[52] intercept   : AIC=3819.024, Time=0.17 sec
 ARIMA(2,1,2)(2,0,0)[52] intercept   : AIC=3786.759, Time=56.59 sec
 ARIMA(2,1,2)(2,0,1)[52] intercept   : AIC=3788.572, Time=25.41 sec
 ARIMA(1,1,2)(1,0,0)[52] intercept   : AIC=3785.525, Time=5.85 sec
 ARIMA(1,1,2)(0,0,0)[52] intercept   : AIC=3819.410, Time=0.32 sec
 ARIMA(1,1,2)(2,0,0)[52] intercept   : AIC=inf, Time=37.78 sec
 ARIMA(1,1,2)(1,0,1)[52] intercept   : AIC=3786.145, Time=6.79 sec
 ARIMA(1,1,2)(0,0,1)[52] intercept   : AIC=3794.915, Time=3.25 sec
 ARIMA(1,1,2)(2,0,1)[52] intercept   : AIC=3788.057, Time=31.97 sec
 ARIMA(0,1,2)(1,0,0)[52] intercept   : AIC=3789.312, Time=1.78 sec
 ARIMA(1,1,1)(1,0,0)[52] intercept   : AIC=3778.591, Time=3.59 sec
 ARIMA(1,1,1)(0,0,0)[52] intercept   : AIC=3819.035, Time=0.10 sec
 ARIMA(1,1,1)(2,0,0)[52] intercept   : AIC=3778.695, Time=22.93 sec
 ARIMA(1,1,1)(1,0,1)[52] intercept   : AIC=3778.757, Time=4.73 sec
 ARIMA(1,1,1)(0,0,1)[52] intercept   : AIC=3789.828, Time=1.97 sec
 ARIMA(1,1,1)(2,0,1)[52] intercept   : AIC=3780.695, Time=16.17 sec
 ARIMA(0,1,1)(1,0,0)[52] intercept   : AIC=3785.016, Time=1.12 sec
 ARIMA(2,1,1)(1,0,0)[52] intercept   : AIC=3779.213, Time=3.04 sec
 ARIMA(0,1,0)(1,0,0)[52] intercept   : AIC=3808.891, Time=0.75 sec
 ARIMA(2,1,0)(1,0,0)[52] intercept   : AIC=3791.364, Time=2.86 sec
 ARIMA(1,1,1)(1,0,0)[52]             : AIC=3776.592, Time=1.95 sec
 ARIMA(1,1,1)(0,0,0)[52]             : AIC=3817.037, Time=0.08 sec
 ARIMA(1,1,1)(2,0,0)[52]             : AIC=3776.695, Time=20.87 sec
 ARIMA(1,1,1)(1,0,1)[52]             : AIC=3776.758, Time=4.50 sec
 ARIMA(1,1,1)(0,0,1)[52]             : AIC=3787.828, Time=1.69 sec
 ARIMA(1,1,1)(2,0,1)[52]             : AIC=3778.696, Time=14.79 sec
 ARIMA(0,1,1)(1,0,0)[52]             : AIC=3783.017, Time=0.92 sec
 ARIMA(1,1,0)(1,0,0)[52]             : AIC=3793.660, Time=0.81 sec
 ARIMA(2,1,1)(1,0,0)[52]             : AIC=3777.258, Time=2.68 sec
 ARIMA(1,1,2)(1,0,0)[52]             : AIC=3784.011, Time=5.00 sec
 ARIMA(0,1,0)(1,0,0)[52]             : AIC=inf, Time=0.79 sec
 ARIMA(0,1,2)(1,0,0)[52]             : AIC=3787.462, Time=1.38 sec
 ARIMA(2,1,0)(1,0,0)[52]             : AIC=3789.367, Time=1.21 sec
 ARIMA(2,1,2)(1,0,0)[52]             : AIC=3784.522, Time=2.97 sec

Best model:  ARIMA(1,1,1)(1,0,0)[52]
Total fit time: 305.481 seconds
```

```
▾ ARIMA
ARIMA(1,1,1)(1,0,0)[52]
```

```
n_periods = 12
forecast, conf_int = model.predict(n_periods=n_periods, return_conf_int=True)

forecast
```

	0
2012-11-02	1.638781e+06
2012-11-09	1.596926e+06
2012-11-16	1.573151e+06
2012-11-23	1.818433e+06
2012-11-30	1.597295e+06
2012-12-07	1.703985e+06
2012-12-14	1.744323e+06
2012-12-21	1.936403e+06
2012-12-28	1.554968e+06
2013-01-04	1.581093e+06
2013-01-11	1.536289e+06
2013-01-18	1.504101e+06

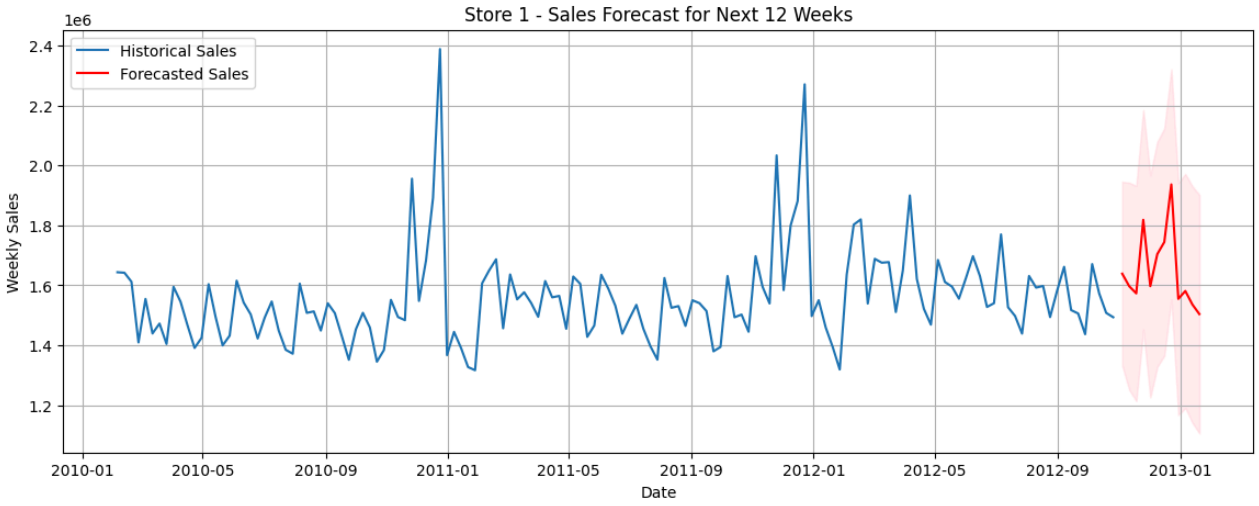
dtype: float64

```
import numpy as np

# Create future date index
last_date = store1_ts.index[-1]
future_dates = pd.date_range(last_date, periods=n_periods+1, freq='W')[1:]

# Plot
plt.figure(figsize=(14,5))
plt.plot(store1_ts.index, store1_ts, label='Historical Sales')
plt.plot(future_dates, forecast, label='Forecasted Sales', color='red')
plt.fill_between(future_dates,
                 conf_int[:, 0],
                 conf_int[:, 1],
                 color='pink', alpha=0.3)

plt.title("Store 1 - Sales Forecast for Next 12 Weeks")
plt.xlabel("Date")
plt.ylabel("Weekly Sales")
plt.legend()
plt.grid(True)
plt.show()
```




```
import joblib
joblib.dump(model, "store1_forecast_model.pkl")
```

```
['store1_forecast_model.pkl']
```

```
all_forecasts = {}

for store_id in df['Store'].unique():
    print(f"Processing Store {store_id}...")

    # Filter store data
    store_df = df[df['Store'] == store_id].sort_values("Date")
    store_ts = store_df.set_index("Date")["Weekly_Sales"]

    # Fit model
    model = auto_arima(
        store_ts,
        seasonal=True,
        m=52,
        error_action="ignore",
        suppress_warnings=True
    )

    # Forecast next 12 weeks
    forecast = model.predict(n_periods=12)

    # Create future dates
    last_date = store_ts.index[-1]
    future_dates = pd.date_range(last_date, periods=13, freq='W')[1:]

    # Store results
    all_forecasts[store_id] = pd.DataFrame({
        "Store": store_id,
```

```
        "Date": future_dates,
        "Forecast_Weekly_Sales": forecast
    })

# Combine all stores' forecast into one DataFrame
final_forecast_df = pd.concat(all_forecasts.values(), ignore_index=True)

# Show top rows
final_forecast_df.head()
```

```
Processing Store 1...
Processing Store 2...
Processing Store 3...
Processing Store 4...
Processing Store 5...
Processing Store 6...
Processing Store 7...
Processing Store 8...
Processing Store 9...
Processing Store 10...
Processing Store 11...
Processing Store 12...
Processing Store 13...
Processing Store 14...
Processing Store 15...
Processing Store 16...
Processing Store 17...
Processing Store 18...
Processing Store 19...
Processing Store 20...
Processing Store 21...
Processing Store 22...
Processing Store 23...
Processing Store 24...
Processing Store 25...
Processing Store 26...
Processing Store 27...
Processing Store 28...
Processing Store 29...
Processing Store 30...
Processing Store 31...
Processing Store 32...
Processing Store 33...
Processing Store 34...
Processing Store 35...
Processing Store 36...
Processing Store 37...
Processing Store 38...
Processing Store 39...
Processing Store 40...
Processing Store 41...
Processing Store 42...
Processing Store 43...
Processing Store 44...
Processing Store 45...
```

	Store	Date	Forecast_Weekly_Sales
0	1	2012-11-04	1.638781e+06
1	1	2012-11-11	1.596926e+06
2	1	2012-11-18	1.573151e+06
3	1	2012-11-25	1.818433e+06
4	1	2012-12-02	1.597295e+06

```
final_forecast_df.to_csv("Walmart_All_Stores_Forecast.csv", index=False)
```

```
report = """
# 📘 Final Project Report
### Capstone Project – Walmart Sales Forecasting & Online Retail Customer Segmentation
By: Bharath G

---

## a. Problem Statement

### Problem Statement 1 – Walmart
A retail chain with multiple stores across the country is facing inventory management problems. They are unable to matc

### Problem Statement 2 – Online Retail
An online retail company wants to understand customer purchasing behavior. The goal is to analyze purchase history and

---
```

```
## b. Project Objective

### Walmart
- Understand sales patterns across all stores.
- Identify trends and weekly seasonality.
- Build forecasting models (12 weeks ahead).
- Save models and forecasts.

### Online Retail
- Analyze customer behavior.
- Perform RFM analysis.
- Cluster customers.
- Save segmentation model and results.

---

## c. Data Description

### Walmart Dataset
- 6435 rows, 8 columns
- Columns: Store, Date, Weekly_Sales, Holiday_Flag, Temperature, Fuel_Price, CPI, Unemployment

### Online Retail Dataset
- 387,961 rows, 8 columns
- Columns: Invoice, StockCode, Description, Quantity, InvoiceDate, Price, CustomerID, Country

---

## d. Data Pre-processing Steps

### Walmart
- Converted Date column into datetime
- Sorted by Date & Store
- Checked missing values
- Basic plots for trend and seasonality

### Online Retail
- Removed missing CustomerID
- Converted InvoiceDate to datetime
- Removed returns (negative quantities)
- Added TotalPrice = Quantity * Price

---

## e. Choosing the Algorithm

### Walmart
SARIMA (Seasonal ARIMA) via auto_arima for forecasting.

### Online Retail
K-Means clustering for customer segmentation.

---

## f. Motivation for Choosing Algorithm

### SARIMA
- Weekly seasonality
- Auto ARIMA picks best parameters
- Works well for retail forecasting

### K-Means
- Simple, effective clustering for RFM
- Works well for numerical data

---

## g. Assumptions

### Walmart
- Past patterns continue
- Holidays affect sales similarly
- No major structural changes

### Online Retail
- RFM captures buying behavior
- Data represents typical customer patterns

---

## h. Model Evaluation

### Walmart
```

```
- Auto ARIMA AIC selection
- Historical vs Forecast plots
- Confidence intervals

### Online Retail
- Cluster statistics (mean RFM per cluster)
- Distribution of customers across clusters

---

## i. Inferences

### Walmart
- Strong weekly seasonality
- Holiday spikes in Nov–Dec
- Store-level differences in sales
- Forecast helps inventory planning

### Online Retail
- Small group of customers generate large revenue
- Clear segments: loyal, recent, at-risk, low-value
- Marketing can be targeted by RFM clusters

---

## j. Future Possibilities

### Walmart
- SARIMAX with external features
- LSTM deep learning models
- BI dashboards for stores

### Online Retail
- Customer churn prediction
- Product recommendation system
- Hierarchical clustering

---

## Saved Models

### Walmart
- store1_forecast_model.pkl
- Store1_12week_forecast.csv
- Walmart_All_Stores_Forecast.csv

### Online Retail
- rfm_scaler.pkl
- rfm_kmeans.pkl
- OnlineRetail_Customer_Segments.csv

---

# End of Project Report
"""

print(report)
```

```
# 📘 Final Project Report
### Capstone Project – Walmart Sales Forecasting & Online Retail Customer Segmentation
**Author:** Bharath G.

---

## a. Problem Statement

### Problem Statement 1 – Walmart
A retail chain with multiple stores across the country is facing inventory management problems. They are unable to match s

### Problem Statement 2 – Online Retail
An online retail company wants to understand customer purchasing behavior. The goal is to analyze purchase history and seg

---

## b. Project Objective

### Walmart
- Understand sales patterns across all stores.
- Identify trends and weekly seasonality.
- Build forecasting models (12 weeks ahead).
- Save models and forecasts.
```

```
### Online Retail
- Analyze customer behavior.
- Perform RFM analysis.
- Cluster customers.
- Save segmentation model and results.

---

## c. Data Description

### Walmart Dataset
- 6435 rows, 8 columns
- Columns: Store, Date, Weekly_Sales, Holiday_Flag, Temperature, Fuel_Price, CPI, Unemployment

### Online Retail Dataset
- 387,961 rows, 8 columns
- Columns: Invoice, StockCode, Description, Quantity, InvoiceDate, Price, CustomerID, Country

---

## d. Data Pre-processing Steps

### Walmart
```