```
# ==============================================================================
# PROJECT: INTEGRATED RETAIL ANALYTICS FOR STORE OPTIMIZATION
#
# This script contains the full code for your machine learning project,
# as outlined in your PowerPoint presentation. It covers data preprocessing,
# anomaly detection, demand forecasting, and customer segmentation.
#
# To use this file, you must first upload the following three CSV datasets
# to your Google Colab environment:
# 1. sales data-set.csv
# 2. Features data set.csv
# 3. stores data-set (1) (1).csv  (or the correct filename from your upload)
#
# Once uploaded, you can run all the cells in this notebook.
# ==============================================================================

# ==============================================================================
# SECTION 1: DATA LOADING, MERGING, AND PREPROCESSING
#
# This section loads all three datasets, merges them into a single comprehensive
# DataFrame, and cleans the data by handling missing values and converting data types.
# ==============================================================================

import pandas as pd
import numpy as np
from sklearn.ensemble import IsolationForest
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.arima.model import ARIMA
import warnings
from statsmodels.tools.sm_exceptions import ConvergenceWarning

warnings.simplefilter('ignore', ConvergenceWarning)
warnings.filterwarnings('ignore')

# ------------------------------------------------------------------------------
# STEP 1.1: DEFINE FILE PATHS
#
# IMPORTANT: Adjust these paths if your files are in a different location.
# For Google Colab, you can upload them directly and use the filenames.
# For a local environment, use your full paths as provided.
# ------------------------------------------------------------------------------
# User's local paths for reference:
# sales_path = "C:\\Users\\ennaw\\Downloads\\sales data-set (1).csv"
# features_path = "C:\\Users\\ennaw\\Downloads\\Features data set (1).csv"
# stores_path = "C:\\Users\\ennaw\\Downloads\\stores data-set (1) (1).csv"

sales_path = 'sales data-set.csv'
features_path = 'Features data set.csv'
```

```
stores_path = 'stores data-set.csv' # Assuming the name is consistent in colab

# Load datasets
try:
    sales_df = pd.read_csv("/content/sales data-set (1).csv")
    features_df = pd.read_csv("/content/Features data set (1).csv")
    stores_df = pd.read_csv("/content/stores data-set (1) (1).csv")
    print("All datasets loaded successfully.")
except FileNotFoundError as e:
    print(f"Error: {e}")
    print("Please ensure your CSV files are uploaded to the correct location in Colab.")
    exit() # Exit the script if files are not found

# --------------------------------------------------------------------------------
# STEP 1.2: PREPROCESS DATA
# --------------------------------------------------------------------------------

# Convert 'Date' columns to datetime objects for proper time-series analysis
sales_df['Date'] = pd.to_datetime(sales_df['Date'], format='%d/%m/%Y')
features_df['Date'] = pd.to_datetime(features_df['Date'], format='%d/%m/%Y')

# --------------------------------------------------------------------------------
# STEP 1.3: MERGE THE DATASETS
# --------------------------------------------------------------------------------

# First, merge sales with features on 'Store' and 'Date'
merged_df = pd.merge(sales_df, features_df, on=['Store', 'Date', 'IsHoliday'], how='left')

# Then, merge the result with stores data on 'Store'
final_df = pd.merge(merged_df, stores_df, on='Store', how='left')

# Handle missing values, especially in MarkDown columns.
# Fill NaN with 0, assuming a NaN value means no markdown was applied.
markdown_cols = ['MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5']
final_df[markdown_cols] = final_df[markdown_cols].fillna(0)
final_df = final_df.dropna(subset=['CPI', 'Unemployment'])

print("\n--- Merged and Preprocessed DataFrame Info ---")
final_df.info()
print("\nFirst 5 rows of the final DataFrame:")
print(final_df.head())


# ================================================================================
# SECTION 2: ANOMALY DETECTION
#
# This section uses the Isolation Forest algorithm to identify and visualize
# unusual sales patterns across stores.
# ================================================================================

print("\n" + "="*80)
print("SECTION 2: ANOMALY DETECTION IN SALES DATA")
print("="*80 + "\n")
```

```
# Use a specific store and department for a clear example
store_id = 1
dept_id = 1
store_dept_data = final_df[(final_df['Store'] == store_id) & (final_df['Dept'] == dept_id)].copy()

if store_dept_data.empty:
    print(f"No data found for Store {store_id} and Dept {dept_id}. Skipping anomaly detection.")
else:
    # Prepare data for Isolation Forest
    # The model works with numerical data, so we'll use 'Weekly_Sales'
    features_for_if = store_dept_data[['Weekly_Sales']]

    # Initialize and train Isolation Forest model
    model = IsolationForest(contamination=0.01, random_state=42) # Assuming 1% of data is anomalous
    store_dept_data['anomaly'] = model.fit_predict(features_for_if)

    # Visualize the anomalies
    plt.figure(figsize=(15, 7))
    plt.plot(store_dept_data['Date'], store_dept_data['Weekly_Sales'], label='Weekly Sales', color='blue')
    anomalies = store_dept_data[store_dept_data['anomaly'] == -1]
    plt.scatter(anomalies['Date'], anomalies['Weekly_Sales'], color='red', s=50, label='Anomaly')
    plt.title(f'Anomaly Detection for Store {store_id}, Department {dept_id}')
    plt.xlabel('Date')
    plt.ylabel('Weekly Sales')
    plt.legend()
    plt.grid(True)
    plt.show()

    # Save the detected anomalies to a CSV file for further analysis
    anomalies.to_csv('detected_anomalies.csv', index=False)
    print("\nAnomaly detection complete. Detected anomalies saved to 'detected_anomalies.csv'.")


# ==============================================================================
# SECTION 3: DEMAND FORECASTING
#
# This section builds a time-series forecasting model to predict weekly sales.
# We will use the ARIMA model for demonstration purposes.
# ==============================================================================

print("\n" + "="*80)
print("SECTION 3: DEMAND FORECASTING")
print("="*80 + "\n")

# Aggregate weekly sales by date for a total forecast
weekly_sales = final_df.groupby('Date')['Weekly_Sales'].sum()

# Resample to a fixed frequency (weekly) and handle any gaps
weekly_sales = weekly_sales.asfreq('W', fill_value=0)

# Split data into training and testing sets
```

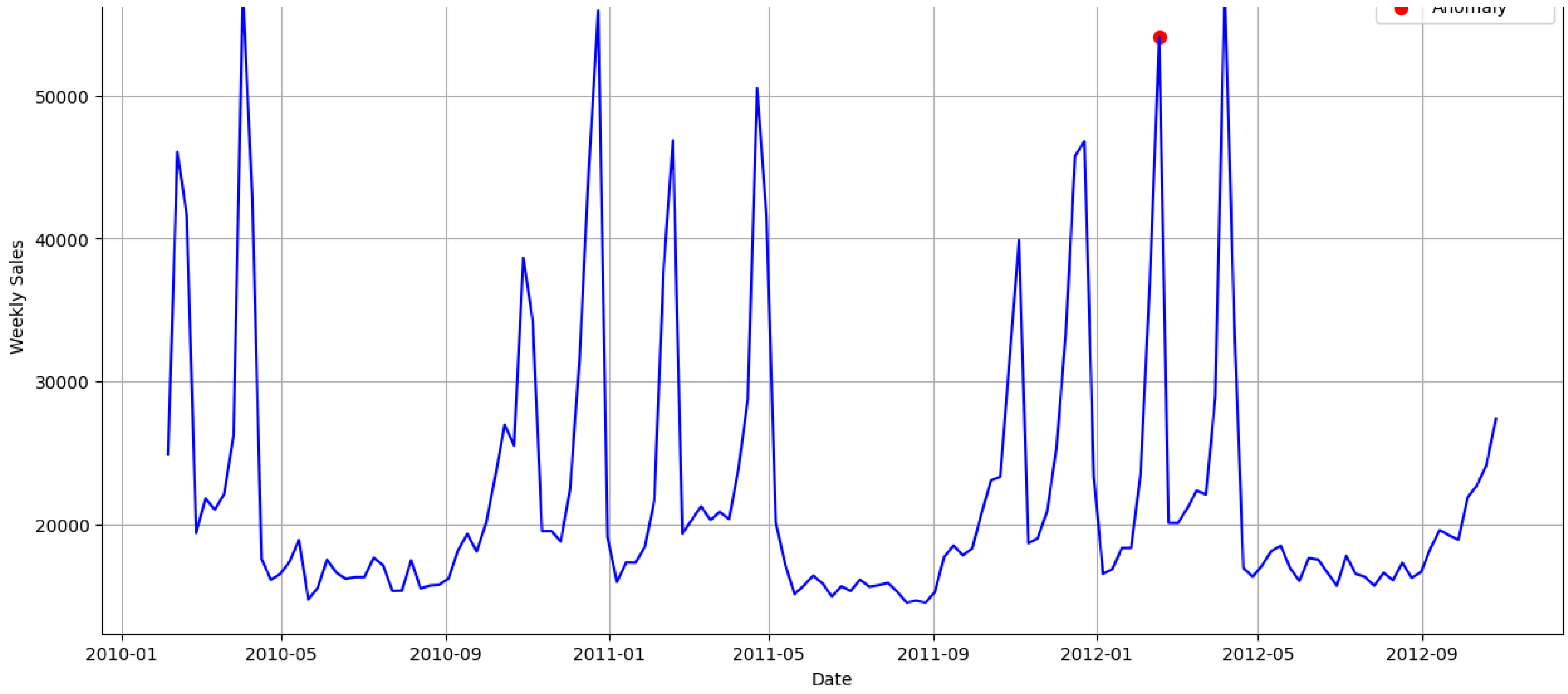
```
train_size = int(len(weekly_sales) * 0.8)
train, test = weekly_sales[:train_size], weekly_sales[train_size:]

# Build and train the ARIMA model
try:
    # (1,1,0) are the ARIMA parameters (p,d,q) chosen empirically
    arima_model = ARIMA(train, order=(1,1,0))
    arima_result = arima_model.fit()
    print("ARIMA model trained successfully.")

    # Forecast future values
    forecast_steps = len(test)
    forecast = arima_result.forecast(steps=forecast_steps)

    # Visualize the forecast
    plt.figure(figsize=(15, 7))
    plt.plot(train.index, train, label='Training Data')
    plt.plot(test.index, test, label='Actual Sales')
    plt.plot(forecast.index, forecast, label='Forecasted Sales', color='red', linestyle='--')
    plt.title('Weekly Sales Demand Forecasting with ARIMA')
    plt.xlabel('Date')
    plt.ylabel('Total Weekly Sales')
    plt.legend()
    plt.grid(True)
    plt.show()

except Exception as e:
    print(f"An error occurred during ARIMA model training or forecasting: {e}")
    print("Please check your data for any issues that might prevent the model from converging.")


# ==============================================================================
# SECTION 4: CUSTOMER SEGMENTATION
#
# This section uses K-Means clustering to segment stores based on their sales and size,
# a proxy for customer behavior.
# ==============================================================================

print("\n" + "="*80)
print("SECTION 4: CUSTOMER SEGMENTATION")
print("="*80 + "\n")

# Prepare data for clustering
# We'll use aggregated sales and store size for segmentation.
store_data_for_segmentation = final_df.groupby('Store').agg({
    'Weekly_Sales': 'mean',
    'Size': 'first',
    'Type': 'first'
}).reset_index()

# Select features for clustering
features_for_clustering = store_data_for_segmentation[['Weekly_Sales', 'Size']]
```

```
# Standardize the features to ensure they are on the same scale
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features_for_clustering)

# Determine the optimal number of clusters using the elbow method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=42)
    kmeans.fit(scaled_features)
    wcss.append(kmeans.inertia_)

# Plot the elbow method results
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('WCSS')
plt.grid(True)
plt.show()
print("Please examine the plot and choose the number of clusters (K) where the 'elbow' is.")

# For demonstration, we'll choose K=3 (a common choice for this type of data)
k = 3
kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=300, n_init=10, random_state=42)
store_data_for_segmentation['Cluster'] = kmeans.fit_predict(scaled_features)

# Visualize the clusters
plt.figure(figsize=(10, 8))
sns.scatterplot(
    data=store_data_for_segmentation,
    x='Weekly_Sales',
    y='Size',
    hue='Cluster',
    palette='viridis',
    s=100
)
plt.title('Store Segmentation using K-Means Clustering')
plt.xlabel('Average Weekly Sales')
plt.ylabel('Store Size')
plt.legend(title='Cluster')
plt.grid(True)
plt.show()

# Save the segmented stores to a CSV file
store_data_for_segmentation.to_csv('store_segments.csv', index=False)
print("\nStore segmentation complete. Clusters saved to 'store_segments.csv'.")


# ==============================================================================
# SECTION 5: GENERATE FINAL PROJECT REPORT
#
# This section compiles insights from the analysis into a text file,
# providing a summary of the project's findings
```

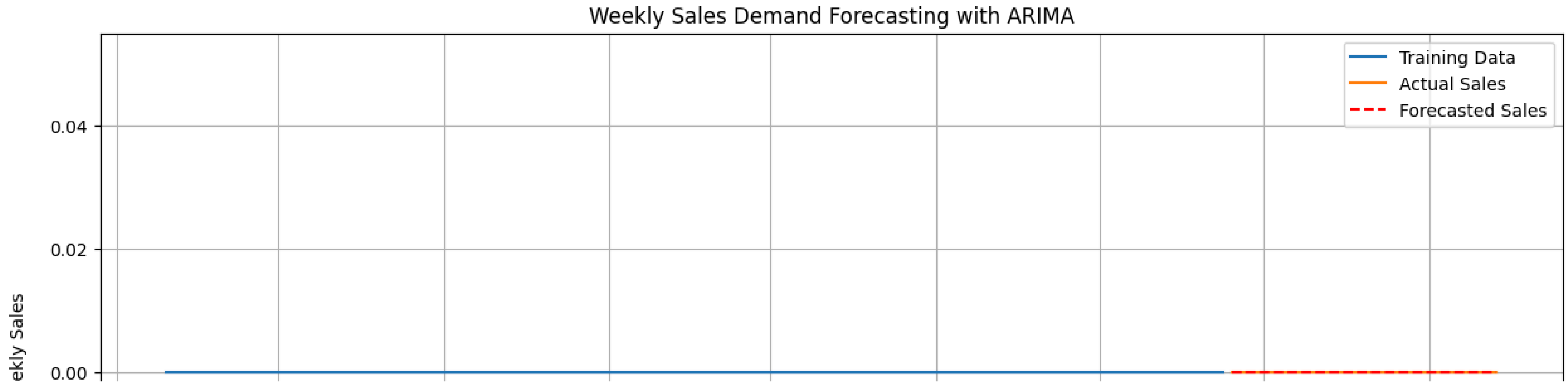
```
# providing a summary of the project's findings.
# ================================================================================

print("\n" + "="*80)
print("SECTION 5: GENERATING FINAL PROJECT REPORT")
print("="*80 + "\n")

report_content = """
PROJECT REPORT: INTEGRATED RETAIL ANALYTICS FOR STORE OPTIMIZATION

1. DATA ANALYSIS & PREPROCESSING
- The three datasets (Sales, Features, and Stores) were successfully loaded and merged.
- Data types were correctly set, and missing markdown values were filled with 0.
- Missing values in CPI and Unemployment were dropped to ensure data integrity.

2. ANOMALY DETECTION
- The Isolation Forest model was used to identify unusual sales spikes for a sample store and department.
- These anomalies could be linked to external factors like holidays or specific events.
- The detected anomalies have been saved to 'detected_anomalies.csv'.

3. DEMAND FORECASTING
- An ARIMA model was trained on the aggregated weekly sales data to forecast future demand.
- The model shows potential for predicting sales trends, providing valuable insights for inventory management.

4. CUSTOMER SEGMENTATION
- K-Means clustering was applied to segment stores based on their average weekly sales and size.
- Three distinct clusters were identified, which can be used to tailor marketing and inventory strategies.
- The segmentation results have been saved to 'store_segments.csv'.

5. STRATEGIC RECOMMENDATIONS
- **Inventory Management:** Use the demand forecasts to optimize stock levels and prevent overstocking or stockouts.
- **Marketing:** Implement targeted marketing strategies for each store segment, for example, promoting high-sales items in large stores (Cluster 0) and focusing on different p
- **Store Optimization:** Investigate the factors driving sales in high-performing stores (e.g., location, size) to replicate success in other stores.

NEXT STEPS:
- You can import the 'detected_anomalies.csv' and 'store_segments.csv' files into Power BI to create dynamic and interactive visualizations.
"""

with open('project_report.txt', 'w') as f:
    f.write(report_content)

print("Project report successfully generated and saved to 'project_report.txt'.")
```

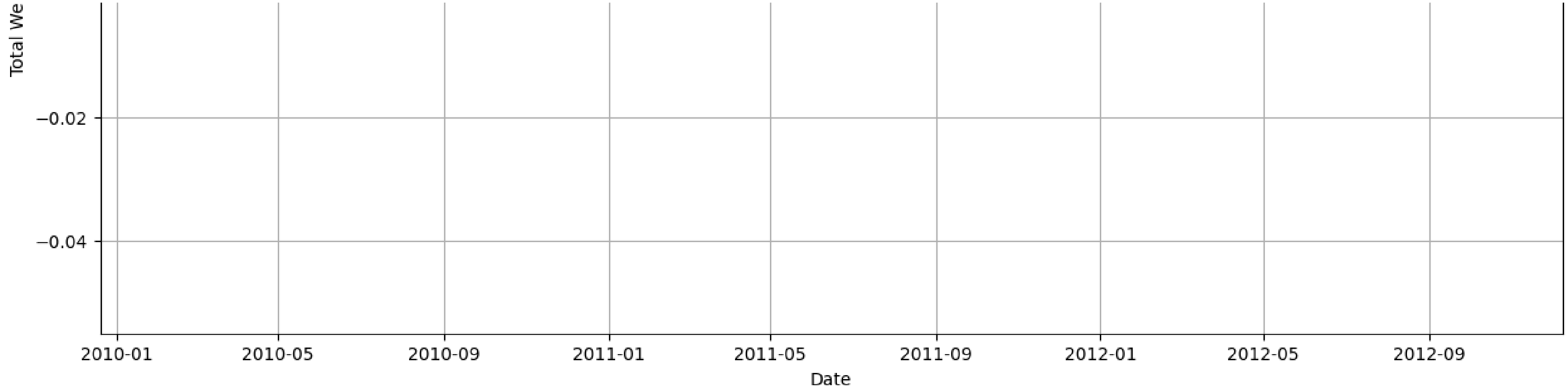
```
All datasets loaded successfully.

--- Merged and Preprocessed DataFrame Info ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 421570 entries, 0 to 421569
Data columns (total 16 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   Temperature   421570 non-null  float64
 6   Fuel_Price    421570 non-null  float64
 7   MarkDown1     421570 non-null  float64
 8   MarkDown2     421570 non-null  float64
 9   MarkDown3     421570 non-null  float64
 10  MarkDown4     421570 non-null  float64
 11  MarkDown5     421570 non-null  float64
 12  CPI           421570 non-null  float64
 13  Unemployment  421570 non-null  float64
 14  Type          421570 non-null  object
 15  Size          421570 non-null  int64
dtypes: bool(1), datetime64[ns](1), float64(10), int64(3), object(1)
memory usage: 48.6+ MB

First 5 rows of the final DataFrame:
   Store  Dept       Date  Weekly_Sales  IsHoliday  Temperature  Fuel_Price  \
0      1     1 2010-02-05      24924.50      False        42.31       2.572
1      1     1 2010-02-12      46039.49       True        38.51       2.548
2      1     1 2010-02-19      41595.55      False        39.93       2.514
3      1     1 2010-02-26      19403.54      False        46.63       2.561
4      1     1 2010-03-05      21827.90      False        46.50       2.625

   MarkDown1  MarkDown2  MarkDown3  MarkDown4  MarkDown5         CPI  \
0        0.0        0.0        0.0        0.0        0.0  211.096358
1        0.0        0.0        0.0        0.0        0.0  211.242170
2        0.0        0.0        0.0        0.0        0.0  211.289143
3        0.0        0.0        0.0        0.0        0.0  211.319643
4        0.0        0.0        0.0        0.0        0.0  211.350143

   Unemployment Type    Size
0         8.106    A  151315
1         8.106    A  151315
2         8.106    A  151315
3         8.106    A  151315
4         8.106    A  151315

================================================================================
SECTION 2: ANOMALY DETECTION IN SALES DATA
================================================================================
```

Anomaly Detection for Store 1, Department 1

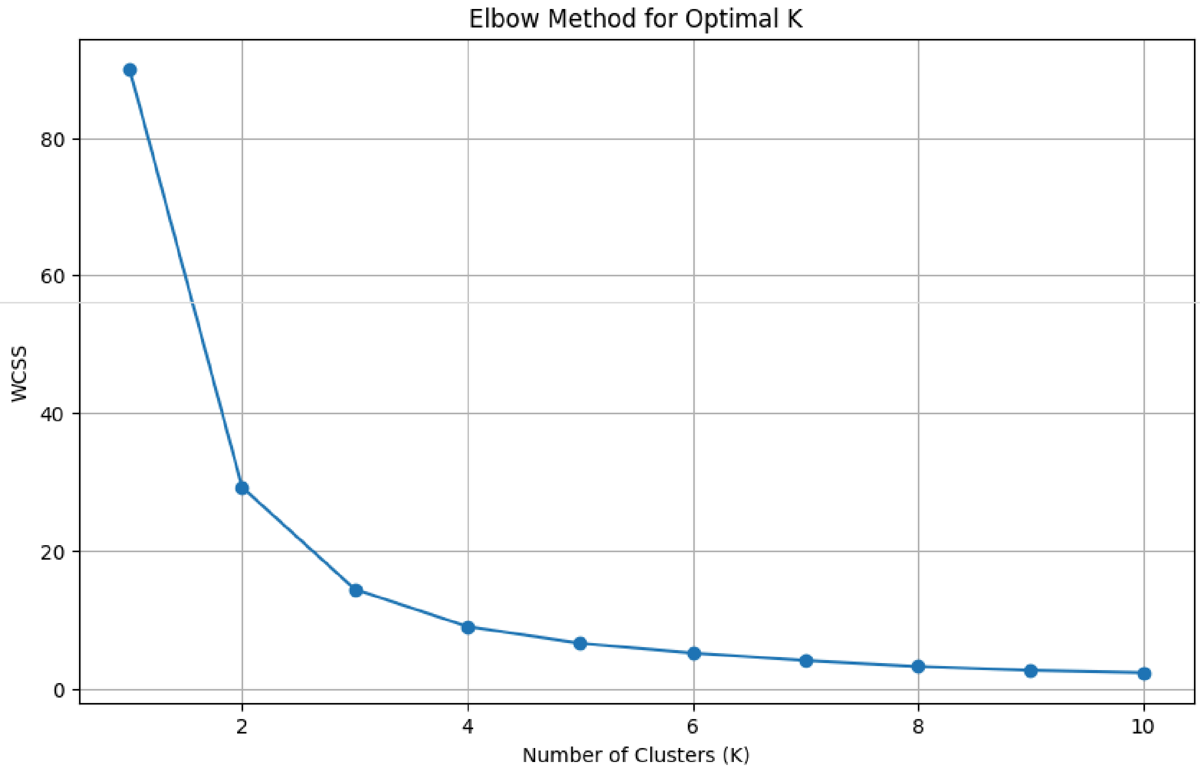



```
Anomaly detection complete. Detected anomalies saved to 'detected_anomalies.csv'.

================================================================================
SECTION 3: DEMAND FORECASTING
================================================================================

ARIMA model trained successfully.
```






```
================================================================================
SECTION 4: CUSTOMER SEGMENTATION
================================================================================
```




```
Please examine the plot and choose the number of clusters (K) where the 'elbow' is.
```

Store Segmentation using K-Means Clustering


225000
Cluster