Technical Case Study

Will Sparrow

Adapted from a technical case prompt. Names, values, and variables have been modified.

Overview

- I recently conducted an exploratory and predictive analysis on a multi-location retail dataset to identify patterns in weekly sales, evaluate store segmentation, and explore forecasting approaches influenced by seasonal/economic factors.

- This project was completed as part of a technical case study.

- Dataset structure and task framing have been modified and anonymized.

Adapted from a technical case prompt. Names, values, and variables have been modified.

Table of Contents

Adapted from a technical case prompt. Names, values, and variables have been modified.

The Data – Weekly Sales

- ~40 Total Stores
  - ~60 departments per store
- Time Range: 2010 - 2013
- Interval: Weekly
- H_Period shows which weeks are Holidays

Location_ID	Dept	Date	Weekly_Sales	H_Period
1	1	2/5/10	24924.5	FALSE
1	1	2/12/10	46039.49	TRUE
1	1	2/19/10	41595.55	FALSE
1	1	2/26/10	19403.54	FALSE
1	1	3/5/10	21827.9	FALSE
1	1	3/12/10	21043.39	FALSE
1	1	3/19/10	22136.64	FALSE
1	1	3/26/10	26229.21	FALSE
1	1	4/2/10	57258.43	FALSE

Weekly Sales

Adapted from a technical case prompt. Names, values, and variables have been modified.

The Data – Factors & Store Type

- Economic and Regional Factors
  - For each week, displays the reported measures of four economic and/or regional variables: **temperature**, **gas price**, the **Consumer Price Index (CPI)**, and **unemployment** rate.

Location_ID	Date	Avg_Reg_Temp	Gas_Price	CPI_Index	Unemp_Pct
1	2/5/10	42.31	2.57	211.10	8.11
1	2/12/10	38.51	2.55	211.24	8.11
1	2/19/10	39.93	2.51	211.29	8.11
1	2/26/10	46.63	2.56	211.32	8.11
1	3/5/10	46.5	2.63	211.35	8.11
1	3/12/10	57.79	2.67	211.38	8.11
1	3/19/10	54.58	2.72	211.22	8.11
1	3/26/10	51.45	2.73	211.02	8.11
1	4/2/10	62.27	2.72	210.82	7.81

Economic and Regional Factors

- Store Type
  - Displays the type and size of store

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

Store Type

Adapted from a technical case prompt. Names, values, and variables have been modified.

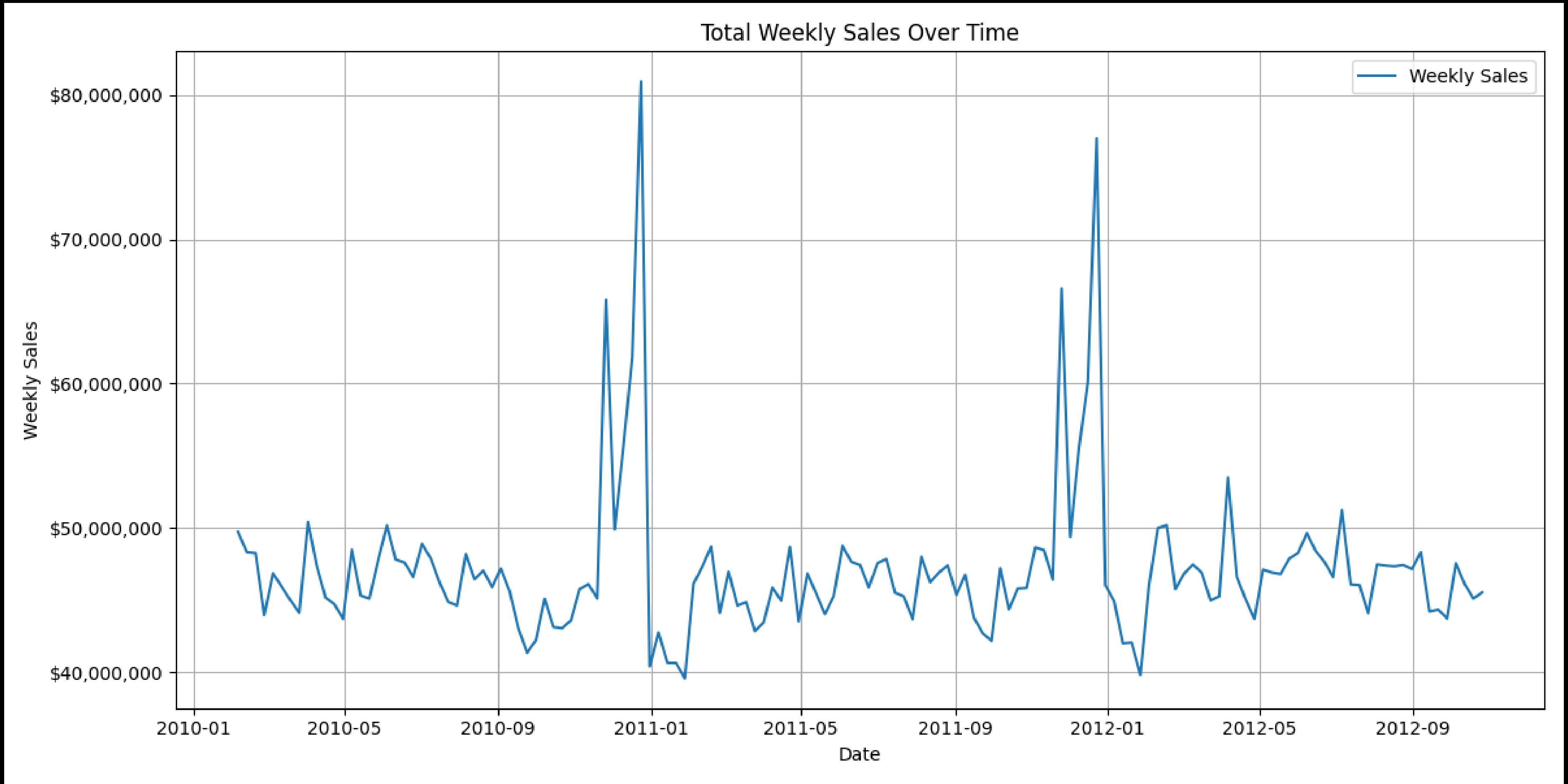
Weekly Sales Insights & Predictive Approach

- On the next few slides, I investigate sales trends through exploratory analysis, focusing on whether holiday periods and regional or economic factors meaningfully influence performance

- I also outline a potential forecasting model that could be used to predict future sales, including the general approach and evaluation considerations.

Weekly Sales Insights & Predictive Approach

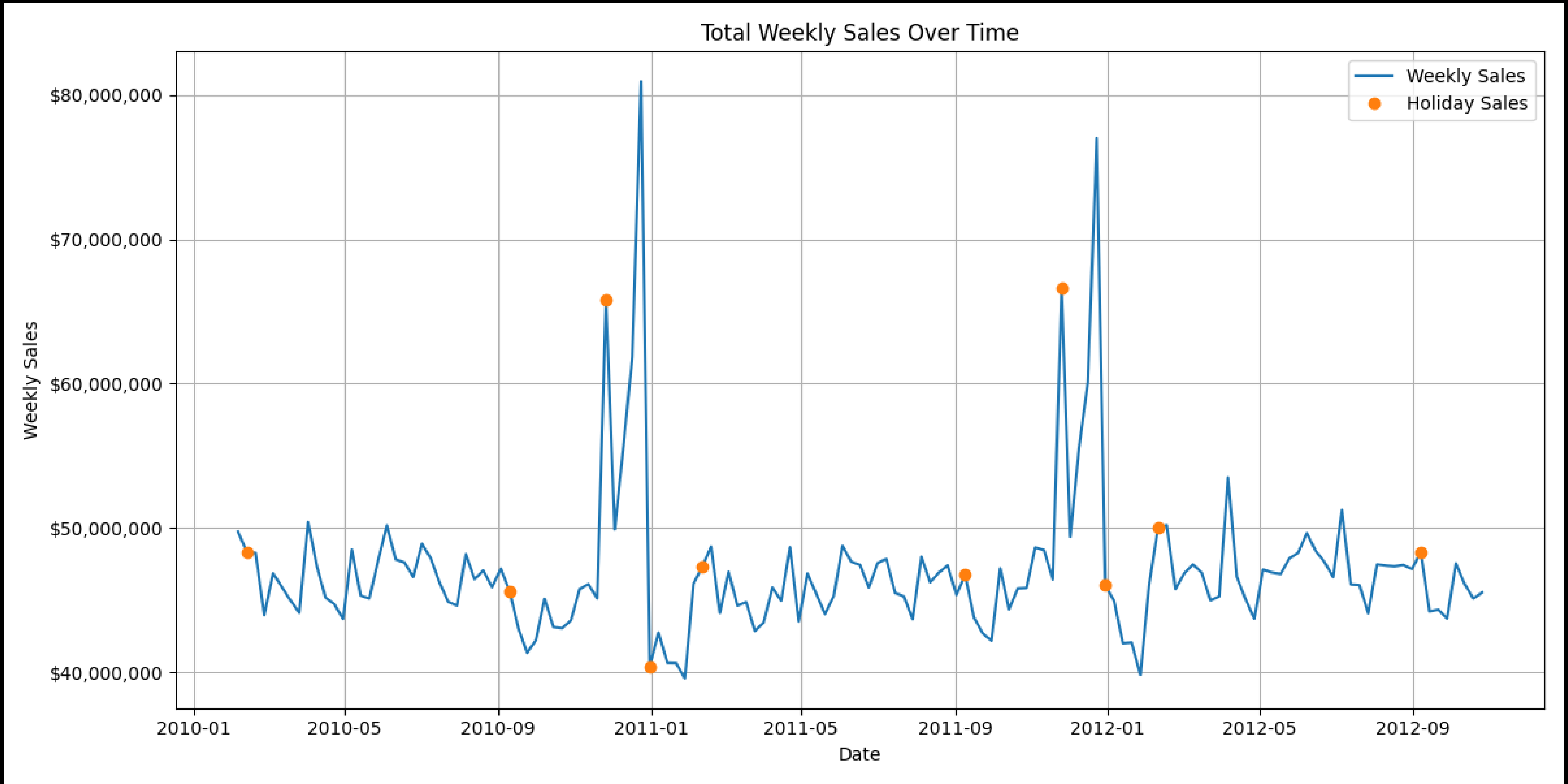
- Weekly sales over time across ALL stores/deps



Adapted from a technical case prompt. Names, values, and variables have been modified.

Weekly Sales Insights & Predictive Approach

- Holidays highlighted by orange dots
- These points are some of the highest weekly sales numbers
- Often a sharp dip following holiday



Adapted from a technical case prompt. Names, values, and variables have been modified.

Weekly Sales Insights & Predictive Approach

- Visually, it appears that holidays have an impact, but let’s run a quick test as well.
- After comparing the average weekly sales for non-holidays versus holidays, we get:

H_Period	
False	15901.445069
True	17035.823187

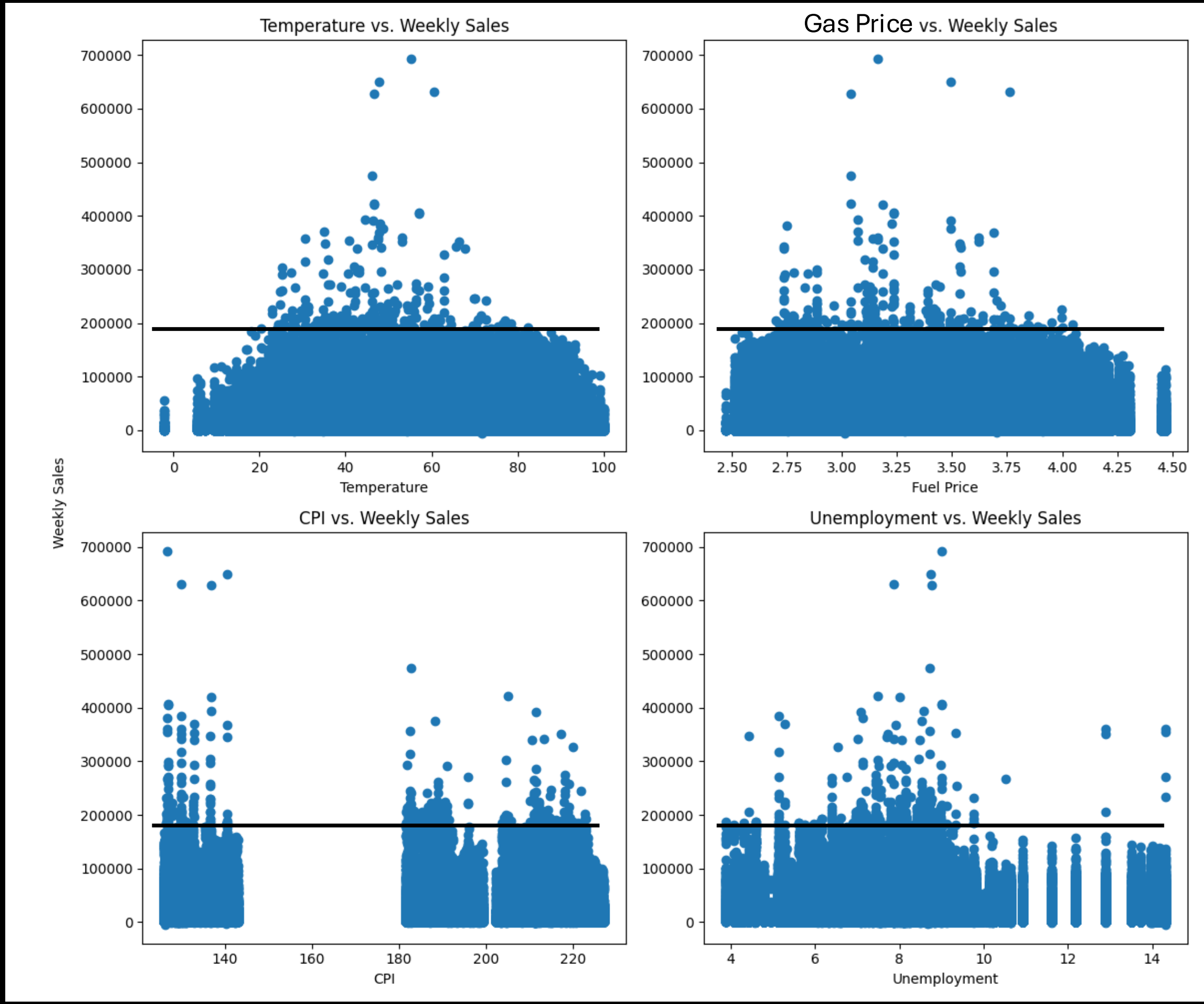
During holiday weeks, sales are ~7% higher

Adapted from a technical case prompt. Names, values, and variables have been modified.

Weekly Sales Insights & Predictive Approach

- All four regional/economic factors plotted against weekly sales shows no correlation

- Temperature, gas prices, CPI, and unemployment rate therefore DO NOT affect sales across all stores



Adapted from a technical case prompt. Names, values, and variables have been modified.

Weekly Sales Insights & Predictive Approach

The below model is a way in which we can predict future sales to account for seasonality of data:

- Methodology: ARIMA model
  - We can use an ARIMA model which takes into account seasonality to forecast weekly sales. This model looks at past data and can use patterns such as the sales output during holiday weeks, to help predict future sales.
- Validation Techniques: Test-Train-Split
  - Because the data is time based, we must sort the data chronologically by the date field. Then take the first 80% of data to train the model, tune the various parameters of the model, and use the last 20% to test the model.
- Metrics to Evaluate Model: Mean Absolute Error
  - After using the test set to generate predictions, take the absolute value between each prediction and the actual weekly sales value and divide that by the number of predictions made. This will give you on average, how far each prediction is away from its actual sales value.

Adapted from a technical case prompt. Names, values, and variables have been modified.

Store Classification & Efficiency Analysis

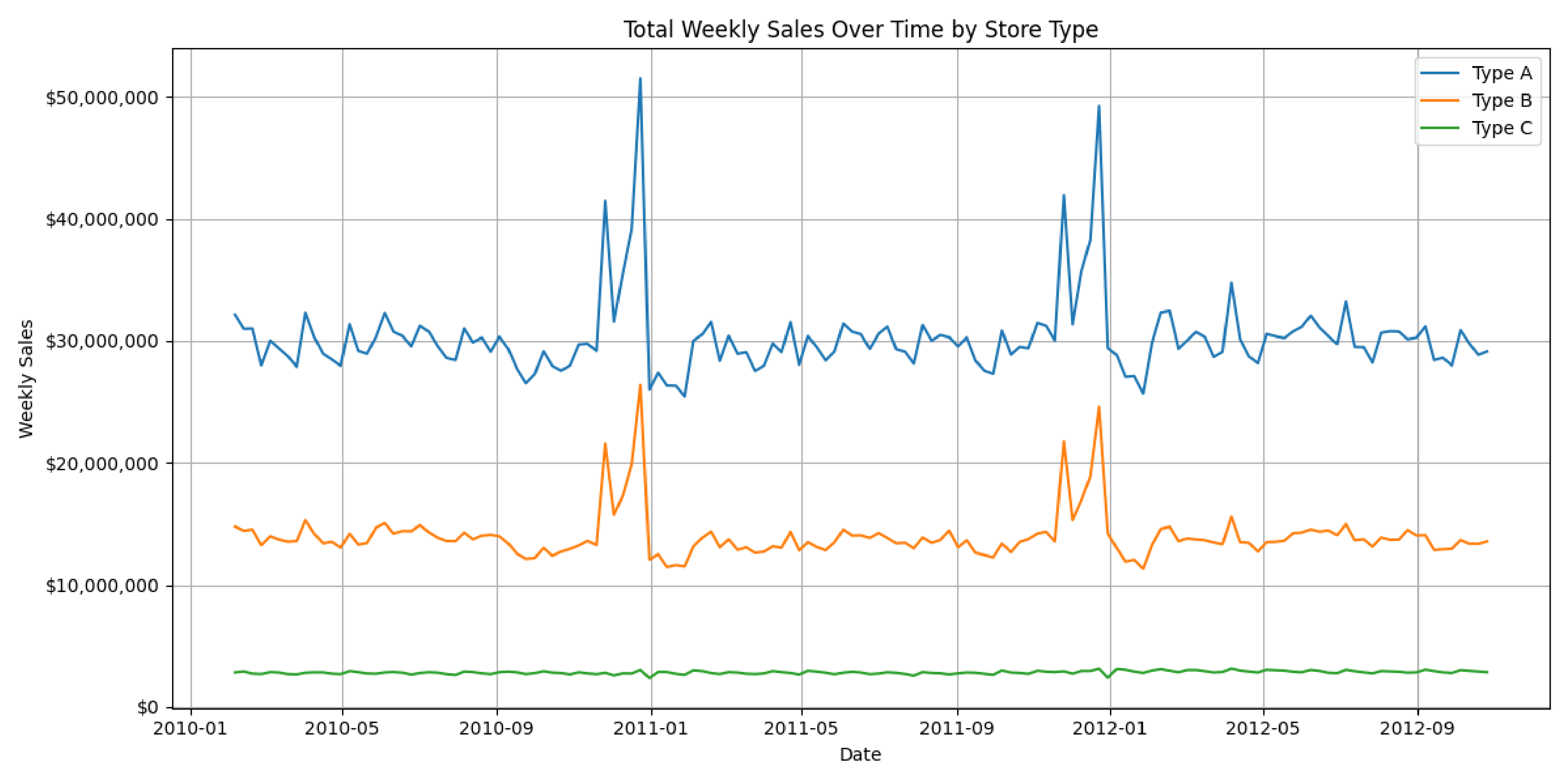
- On the following slides, I segment the stores by size, classification, and regional characteristics, then analyze sales performance across each group to identify the factors most associated with differences in outcomes.

Adapted from a technical case prompt. Names, values, and variables have been modified.

Store Classification & Efficiency Analysis

- We will be looking at four different variables:
  - Store Size
    - For simplicity's sake, I have assumed Store Type and Store Size are the same variable
  - Temperature
  - Gas Prices
  - CPI

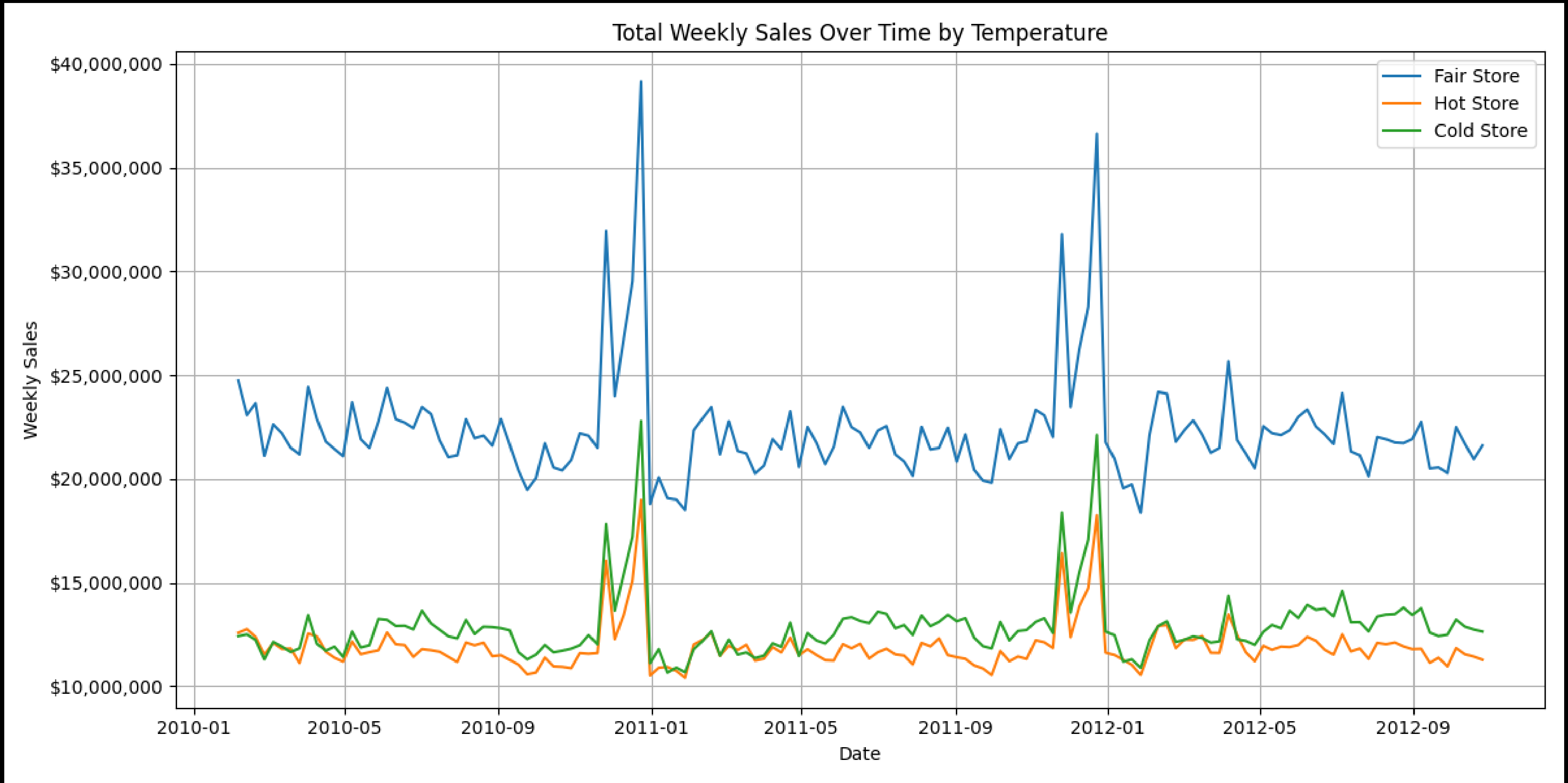
Store Classification & Efficiency Analysis



- Size
  - Type A stores are greater than 150,000
  - Type B are stores between 50,000 and 150,000
  - Type C are stores less than 50,000
  - Larger the store the better the sales

Adapted from a technical case prompt. Names, values, and variables have been modified.

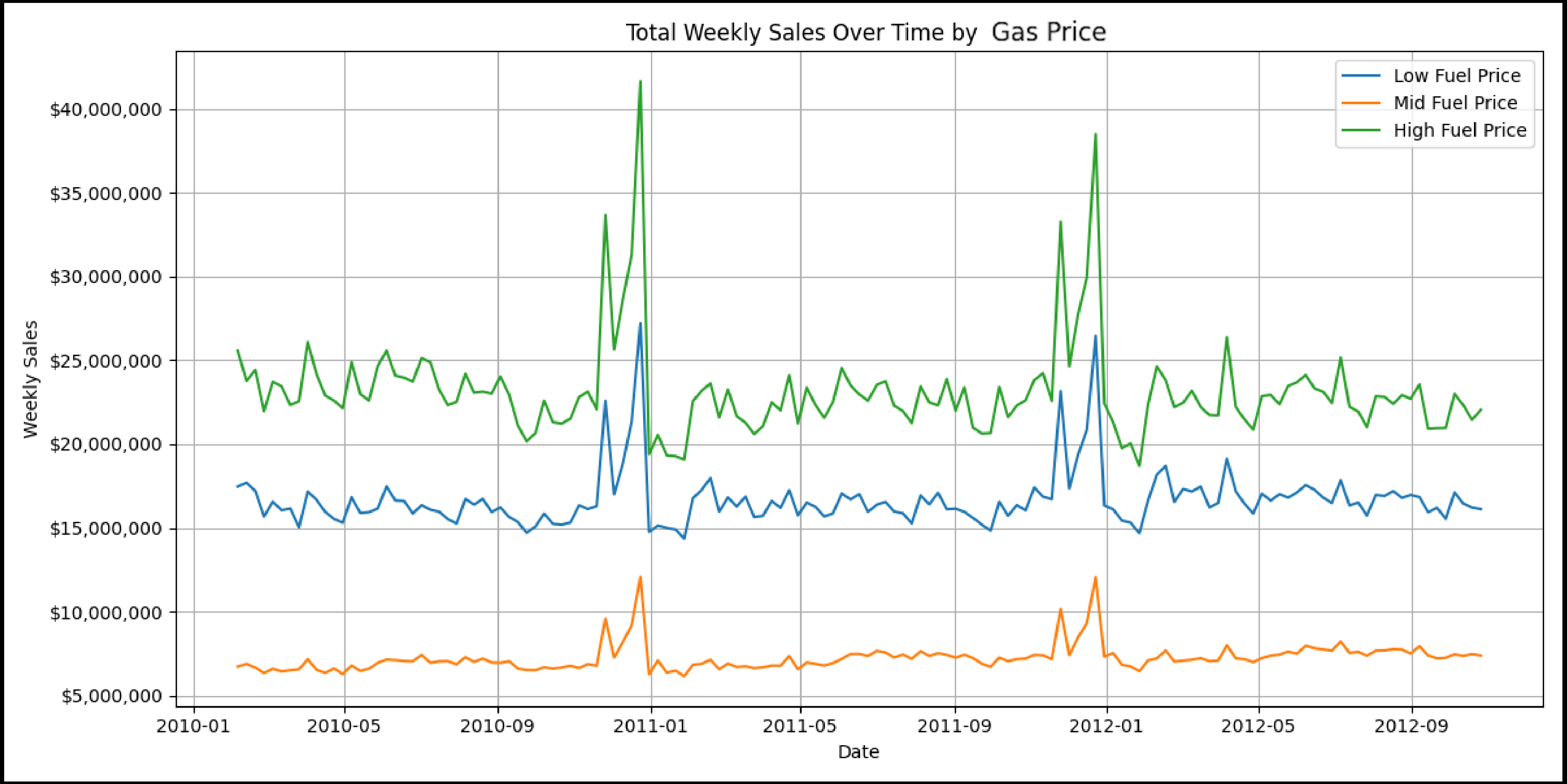
Store Classification & Efficiency Analysis



- Temperature
  - Stores in milder climates perform the best
  - Both hot/cold stores perform about the same

Adapted from a technical case prompt. Names, values, and variables have been modified.

Store Classification & Efficiency Analysis

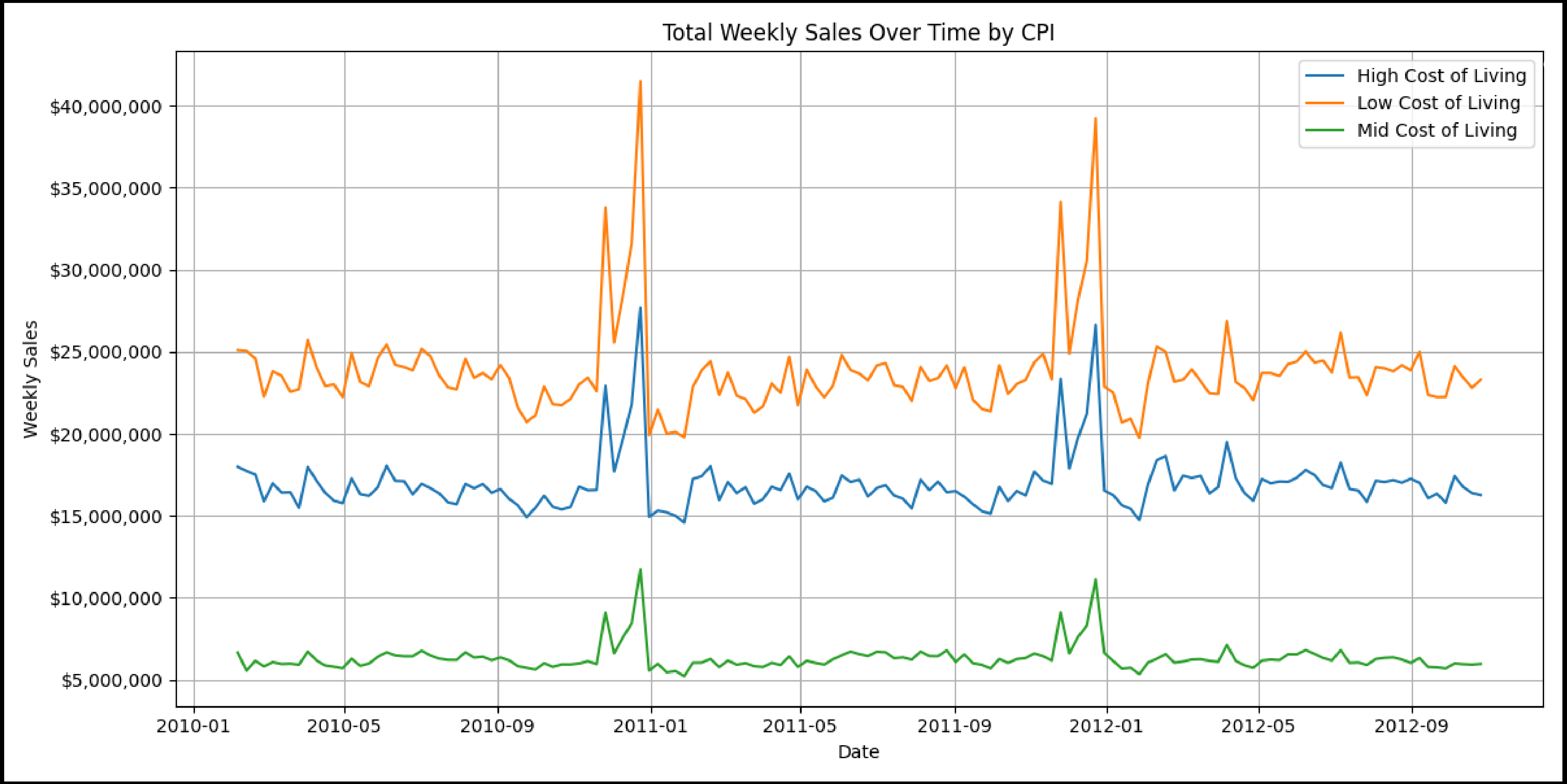


Gas Price

- Stores in regions with high gas prices perform the best
- Stores in. regions with low gas prices perform second best
- Stores in regions with mid gas prices perform last

Adapted from a technical case prompt. Names, values, and variables have been modified.

Store Classification & Efficiency Analysis



CPI

- Stores in LCOL regions perform best
- Stores in HCOL regions perform second best

Adapted from a technical case prompt. Names, values, and variables have been modified.

Store Classification & Efficiency Analysis

Variable	Effect	Potential Explanation
Store Size	The bigger the store size, the more weekly sales.	Assuming this is measured in square footage, the bigger the store is as it relates to area, the more inventory you can hold, and shoppers can ultimately buy.
Temperature	Stores in “fair” regions (milder temps) do much better than both hot/cold stores.	Shoppers are able to do more shopping year round if the weather is fair. They are less likely to go out and purchase things in extremely hot/cold weather.
Fuel Prices	Stores in area with high gas prices perform the best, then stores with low gas prices, and finally stores with mid gas prices.	Shoppers in an area with higher gas prices may have more absolute income (e.g., California residents in major city).
CPI	Stores in lower cost of living areas perform best, then stores in higher cost of living, and finally stores in mid cost of living.	The types of good this store sells, could be less luxury based, and cater more to lower income families.

Adapted from a technical case prompt. Names, values, and variables have been modified.

Improving Models with External Inputs

- On the following slide, I explore how incorporating external data sources could potentially strengthen the predictive models by adding additional context and improving forecast accuracy. I also outline the practical challenges and limitations involved in integrating these data sources with the existing dataset.

Adapted from a technical case prompt. Names, values, and variables have been modified.

Improving Models with External Inputs

Additional Data Sources	How Can Data Add Context or Improve Forecast/Operational Efficiency?	Potential Limitations of Additional Data Sources
Demographic Data	Allows us to understand the characteristics (income, age, etc.) of the surrounding population around our stores. Households and/or individuals with higher income may spend more.	Census data is not always accurate, up to date, and certain measures you might be interested in may not be reported.
Marketing Data	Helps us understand the impact of marketing campaigns on store sales. A recently released commercial or email campaign may result in a spike in weekly sales.	Difficult to quantify if marketing campaigns had direct effect on consumers. Additionally, dollars spent on marketing might not yield dollars spent on goods until much later.
Social Media Data	Helps us understand what the current sentiment of the company is and how it is trending. If the sentiment is negative, then you might expect weekly sales to decline (e.g., boycott of companies).	Sentiment analysis can be very noisy and unreliable. Additionally, general negativity (or positivity) for a company might not be able to be measured regionally.

Adapted from a technical case prompt. Names, values, and variables have been modified.

Python Script

- The following slides will show Python code I used to conduct the analyses.

Adapted from a technical case prompt. Names, values, and variables have been modified.

```python
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

# Read in data
holiday_df = pd.read_excel("Data Science - Holiday.xlsx")
macro_factors_df = pd.read_excel("Data Science - Macro Factors.xlsx")
store_sales_df = pd.read_excel("Data Science - Store Sales.xlsx")
store_type_df = pd.read_excel("Data Science - Store Type.xlsx")

# Combine store_sales_df with macro_factors_df
combined_df = pd.merge(
    store_sales_df,
    macro_factors_df.drop("H_Period", axis=1),  # keep H_Period col from store_sales_df
    on=["Location_ID", "Date"],
    how="left"
)

# Add store type and size to combined df
combined_df = pd.merge(
    combined_df,
    store_type_df,
    on="Location_ID",
    how="left"
)

# Reorder cols so that it reads better for me and save to csv so I can open file and explore
combined_df = combined_df[["Location_ID", "Dept", "Type","Size", "Date",
                           "Weekly_Sales", "H_Period", "Avg_Reg_Temp", "Gas_Price", "CPI_Index", "Unemp_Pct"]]
combined_df.to_csv("Final Combined Sales.csv", index=None)
```

Adapted from a technical case prompt. Names, values, and variables have been modified.

```python
## Conduct exploratory data analysis to understand sales trends and the impact of holidays and regional economic factors.
# Filter for only holiday rows
holidays = combined_df.loc[combined_df["H_Period"] == True]

# Drop duplicates to keep only unique dates
unique_holidays = holidays[["Date", "H_Period"]].drop_duplicates()  # Assumes that all locations/depts observe same holiday schedule

# Convert to dictionary
holiday_dict = dict(zip(unique_holidays["Date"], unique_holidays["H_Period"]))

# Group by and plot weekly sales across all locations/departments
grouped_by_week_sales = combined_df.groupby("Date")["Weekly_Sales"].sum()

# Add holiday sales which just returns the weekly_sales number if that row's date is a holiday week
weekly_sales_df = grouped_by_week_sales.copy()
weekly_sales_df = weekly_sales_df.to_frame(name="Weekly_Sales").reset_index()
weekly_sales_df["Holiday Sales"] = weekly_sales_df["Weekly_Sales"][
    weekly_sales_df["Date"].apply(lambda d: d in holiday_dict)
]

# Plot weekly sales over time across all locations/depts and shows which weeks are holidays
fig1, ax1 = plt.subplots(figsize=(12, 6))
ax1.plot(weekly_sales_df["Date"], weekly_sales_df["Weekly_Sales"], label="Weekly Sales")
ax1.plot(weekly_sales_df["Date"], weekly_sales_df["Holiday Sales"], "o", label="Holiday Sales")
ax1.set_title("Total Weekly Sales Over Time")
ax1.set_xlabel("Date")
ax1.set_ylabel("Weekly Sales")
# Format Y-axis as dollars
ax1.yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: f"${x:,.0f}"))
ax1.grid(True)
ax1.legend()

fig1.tight_layout()
fig1.savefig("weekly_sales_and_holidays.png")  # Save the figure
# plt.show()
plt.close(fig1)
```

Adapted from a technical case prompt. Names, values, and variables have been modified.

```python
# Avg holiday versus non-holiday weekly sales
holiday_v_nonholiday_sales = combined_df.groupby("H_Period")["Weekly_Sales"].mean()
# print(holiday_v_nonholiday_sales) #~7% higher sales numbers when its holiday week

# Create 2x2 scatter plot grid
fig2, axes = plt.subplots(2, 2, figsize=(12, 10))  # 2 rows, 2 columns

axes[0, 0].scatter(combined_df["Avg_Reg_Temp"], combined_df["Weekly_Sales"])
axes[0, 0].set_title("Avg Reg Temp vs. Weekly Sales")
axes[0, 0].set_xlabel("Avg Reg Temp")

axes[0, 1].scatter(combined_df["Gas_Price"], combined_df["Weekly_Sales"])
axes[0, 1].set_title("Gas Price vs. Weekly Sales")
axes[0, 1].set_xlabel("Gas Price")

axes[1, 0].scatter(combined_df["CPI_Index"], combined_df["Weekly_Sales"])
axes[1, 0].set_title("CPI Index vs. Weekly Sales")
axes[1, 0].set_xlabel("CPI Index")

axes[1, 1].scatter(combined_df["Unemp_Pct"], combined_df["Weekly_Sales"])
axes[1, 1].set_title("Unemp Pct vs. Weekly Sales")
axes[1, 1].set_xlabel("Unemp Pct")

# Shared y-axis label
fig2.text(0.04, 0.5, "Weekly Sales", va="center", rotation="vertical")

fig2.tight_layout(rect=[0.05, 0, 1, 1])
fig2.savefig("economic_factors_vs_weekly_sales.png")  # Save the figure
# plt.show()
plt.close(fig2)
```

Adapted from a technical case prompt. Names, values, and variables have been modified.

```python
## Segment the stores based on type, size, and regional characteristics.
# Multi line plot
def multi_line_plot(df, segment):
    fig, ax = plt.subplots(figsize=(12, 6))
    unique_values = df[segment].unique()

    for val in unique_values:
        sub_df = df[df[segment] == val]
        sales_by_date = sub_df.groupby("Date")["Weekly_Sales"].sum().sort_index()
        ax.plot(sales_by_date.index, sales_by_date.values, label=f"{val}")

    if segment.endswith("_Classification_"):
        segment = segment.replace("_Classification_", "")

    ax.set_title(f"Total Weekly Sales Over Time by {segment}")
    ax.set_xlabel("Date")
    ax.set_ylabel("Weekly Sales")
    ax.legend()
    ax.grid(True)
    ax.yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: f"${x:,.0f}"))

    fig.tight_layout()
    fig.savefig(f"weekly_sales_by_{segment}.png")
```

Adapted from a technical case prompt. Names, values, and variables have been modified.

```python
# Split by regional factors (excluding unemployment because it is national/macro factor)
regional_factors_agg_df = combined_df.groupby("Location_ID")[["Avg_Reg_Temp", "Gas_Price", "CPI_Index"]].agg(
    ["min", "median", "max"]
)
# Add a column to classify locations based on their average temp
def classify_temperature(temp):
    if temp < 55:
        return "Cold Store"
    elif 55 <= temp <= 70:
        return "Fair Store"
    else:
        return "Hot Store"


# Add a column to classify locations based on their average CPI
def classify_cpi(cpi):
    if cpi < 140:
        return "Low Cost of Living"
    elif 140 <= cpi <= 200:
        return "Mid Cost of Living"
    else:
        return "High Cost of Living"


# Add a column to classify locations based on their average gas price
def classify_fuel_price(fuel_price):
    if fuel_price < 3.3:
        return "Low Fuel Price"
    elif 3.3 <= fuel_price <= 3.5:
        return "Mid Fuel Price"
    else:
        return "High Fuel Price"


# Apply classification
regional_factors_agg_df["Avg_Reg_Temp_Classification"] = regional_factors_agg_df[("Avg_Reg_Temp", "median")].apply(
    classify_temperature
)
regional_factors_agg_df["CPI_Index_Classification"] = regional_factors_agg_df[("CPI_Index", "median")].apply(
    classify_cpi
)
regional_factors_agg_df["Gas_Price_Classification"] = regional_factors_agg_df[("Gas_Price", "median")].apply(
    classify_fuel_price
)
```

Adapted from a technical case prompt. Names, values, and variables have been modified.

```python
# Count unique entries for each classification column
temp_counts = regional_factors_agg_df["Avg_Reg_Temp_Classification"].value_counts()
cpi_counts = regional_factors_agg_df["CPI_Index_Classification"].value_counts()
fuel_counts = regional_factors_agg_df["Gas_Price_Classification"].value_counts()

print("Avg Reg Temp Classification Counts:\n", temp_counts)
print("\nCPI Index Classification Counts:\n", cpi_counts)
print("\nGas Price Classification Counts:\n", fuel_counts)

# Flatten the data
regional_factors_agg_df.columns = ["_".join(col).strip() if isinstance(col, tuple) else col for col in regional_factors_agg_df.columns]

# Now the DataFrame has singular column names
regional_factors_agg_df = regional_factors_agg_df.reset_index()

# Select relevant cols and merge
combined_df = pd.merge(
    combined_df,
    regional_factors_agg_df[
        [
            "Location_ID",
            "Avg_Reg_Temp_Classification_",
            "CPI_Index_Classification_",
            "Gas_Price_Classification_",
        ]
    ],
    on="Location_ID",
    how="left",
)

multi_line_plot(combined_df, "Avg_Reg_Temp_Classification_")
multi_line_plot(combined_df, "CPI_Index_Classification_")
multi_line_plot(combined_df, "Gas_Price_Classification_")
multi_line_plot(combined_df, "Type")
```

Adapted from a technical case prompt. Names, values, and variables have been modified.