Importing libraries

In [1]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
import xgboost as xgb
```

Loading Data¶

In [ ]:
```

```

In [2]:
```
# Load datasets
train = pd.read_csv('train.csv')
features = pd.read_csv('features.csv')
stores = pd.read_csv('stores.csv')
```

In [3]:
```
train.head()
```

Out[3]:

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

In [4]:
```
features.head()
```

Out[4]:

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

In [5]:

```
stores.head()
```

Out[5]:

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

In [6]:

```
# Convert 'Date' to datetime format
train['Date'] = pd.to_datetime(train['Date'])
features['Date'] = pd.to_datetime(features['Date'])
```

In [ ]:

```

```

In [7]:

```
# Show the first few rows of the merged dataset
train.head()
```

Out[7]:

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

In [8]:

```
# Summary of the dataset (data types, missing values, etc.)
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 421570 entries, 0 to 421569
Data columns (total 5 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
dtypes: bool(1), datetime64[ns](1), float64(1), int64(2)
memory usage: 13.3 MB
```

In [9]:

```
# Statistical summary of numerical columns
train.describe()
```

Out[9]:

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

In [10]:
```
# Assuming 'features' is DataFrame
features[['MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5']] = features[['MarkDown1', 'MarkDown2', 'MarkDown3', 'M
```

In [11]:
```
features.CPI.head()
```

Out[11]:
```
0    211.096358
1    211.242170
2    211.289143
3    211.319643
4    211.350143
Name: CPI, dtype: float64
```

In [12]:
```
features['CPI'] = features['CPI'].interpolate()
```

In [13]:
```
features.Unemployment.head()
```

Out[13]:
```
0    8.106
1    8.106
2    8.106
3    8.106
4    8.106
Name: Unemployment, dtype: float64
```

In [14]:
```
features['Unemployment'] = features['Unemployment'].interpolate()
```

In [15]:
```
features.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   datetime64[ns]
 2   Temperature   8190 non-null   float64
 3   Fuel_Price    8190 non-null   float64
 4   MarkDown1     8190 non-null   float64
 5   MarkDown2     8190 non-null   float64
 6   MarkDown3     8190 non-null   float64
 7   MarkDown4     8190 non-null   float64
 8   MarkDown5     8190 non-null   float64
 9   CPI           8190 non-null   float64
 10  Unemployment  8190 non-null   float64
 11  IsHoliday     8190 non-null   bool
dtypes: bool(1), datetime64[ns](1), float64(9), int64(1)
memory usage: 712.0 KB
```

In [16]:
```
train.head()
```

Out[16]:

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

In [17]:
```
features.drop('IsHoliday',axis=1,inplace=True)
```

In [18]:
```
features.head()
```

Out[18]:

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

In [ ]:

In [ ]:

In [19]:

```
# Merge datasets
train_set = pd.merge(train, features, on=['Date', 'Store'], how='inner')
```

In [20]:

```
train_set.head()
```

Out[20]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI
0	1	1	2010-02-05	24924.50	False	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358
1	1	2	2010-02-05	50605.27	False	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358
2	1	3	2010-02-05	13740.12	False	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358
3	1	4	2010-02-05	39954.04	False	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358
4	1	5	2010-02-05	32229.38	False	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358

In [21]:

```
# Add the week of the year
train_set['Week_of_Year'] = train_set['Date'].dt.isocalendar().week
```

In [22]:
```
train_set.head()
```

Out[22]:

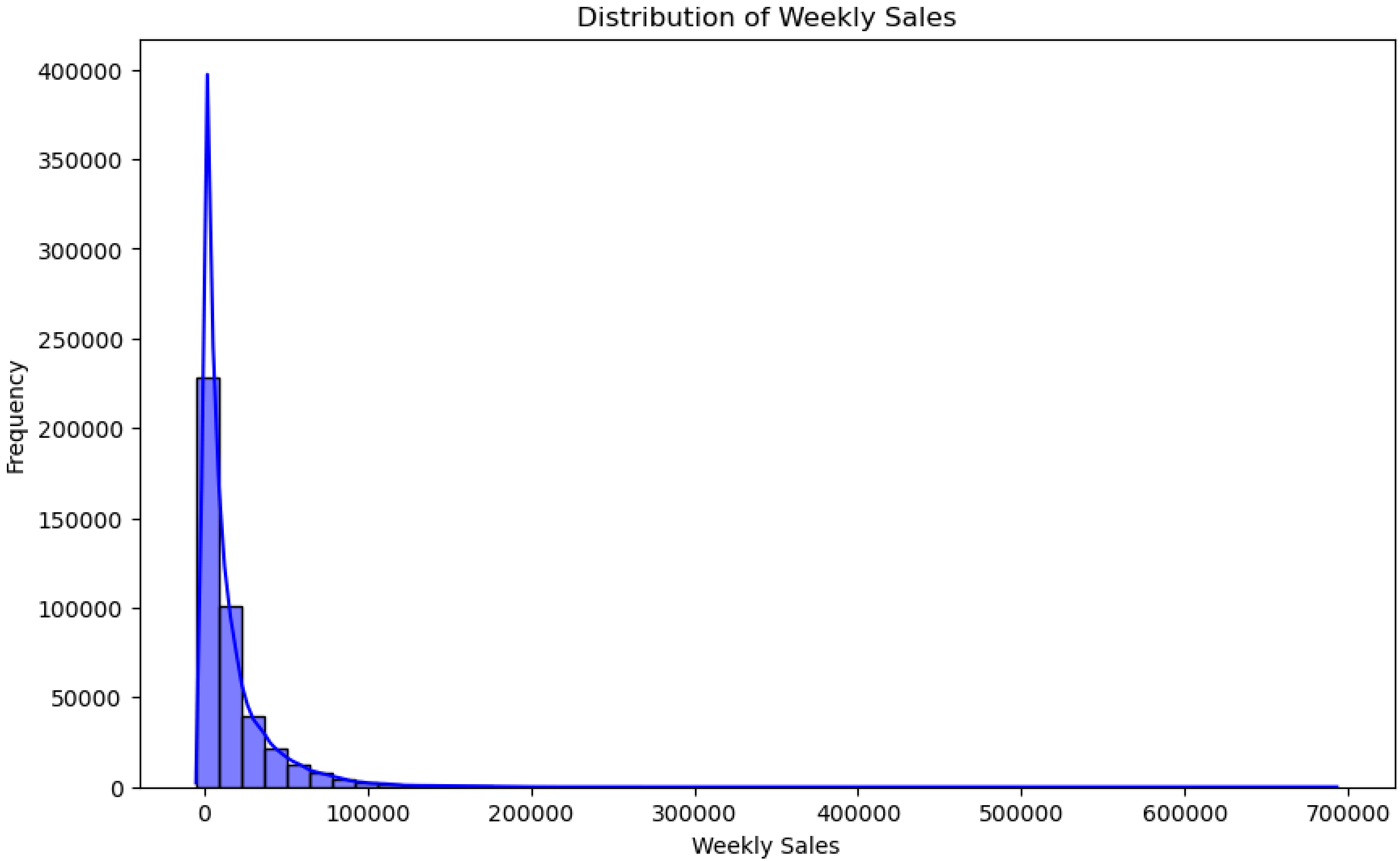
	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI
0	1	1	2010-02-05	24924.50	False	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358
1	1	2	2010-02-05	50605.27	False	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358
2	1	3	2010-02-05	13740.12	False	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358
3	1	4	2010-02-05	39954.04	False	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358
4	1	5	2010-02-05	32229.38	False	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358

In [23]:
```
print(train_set.isnull().sum())
```

```
Store           0
Dept            0
Date            0
Weekly_Sales    0
IsHoliday       0
Temperature     0
Fuel_Price      0
MarkDown1       0
MarkDown2       0
MarkDown3       0
MarkDown4       0
MarkDown5       0
CPI             0
Unemployment    0
Week_of_Year    0
dtype: int64
```

EDA

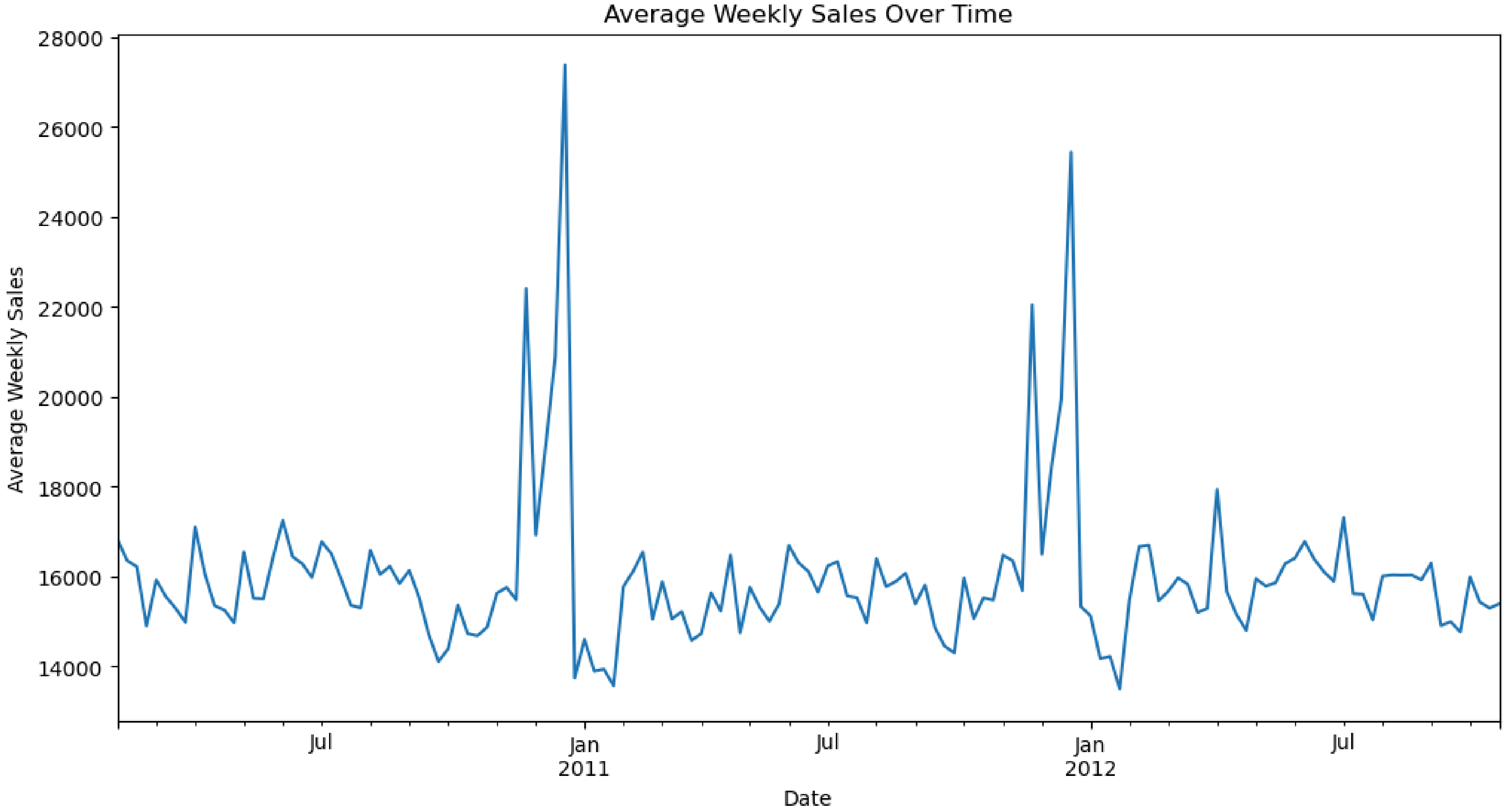
In [24]:
```
# Import visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns

# Plot distribution of Weekly Sales
plt.figure(figsize=(10, 6))
sns.histplot(train_set['Weekly_Sales'], bins=50, kde=True, color='blue')
plt.title('Distribution of Weekly Sales')
plt.xlabel('Weekly Sales')
plt.ylabel('Frequency')
plt.show()
```



Average Weekly Sales over time

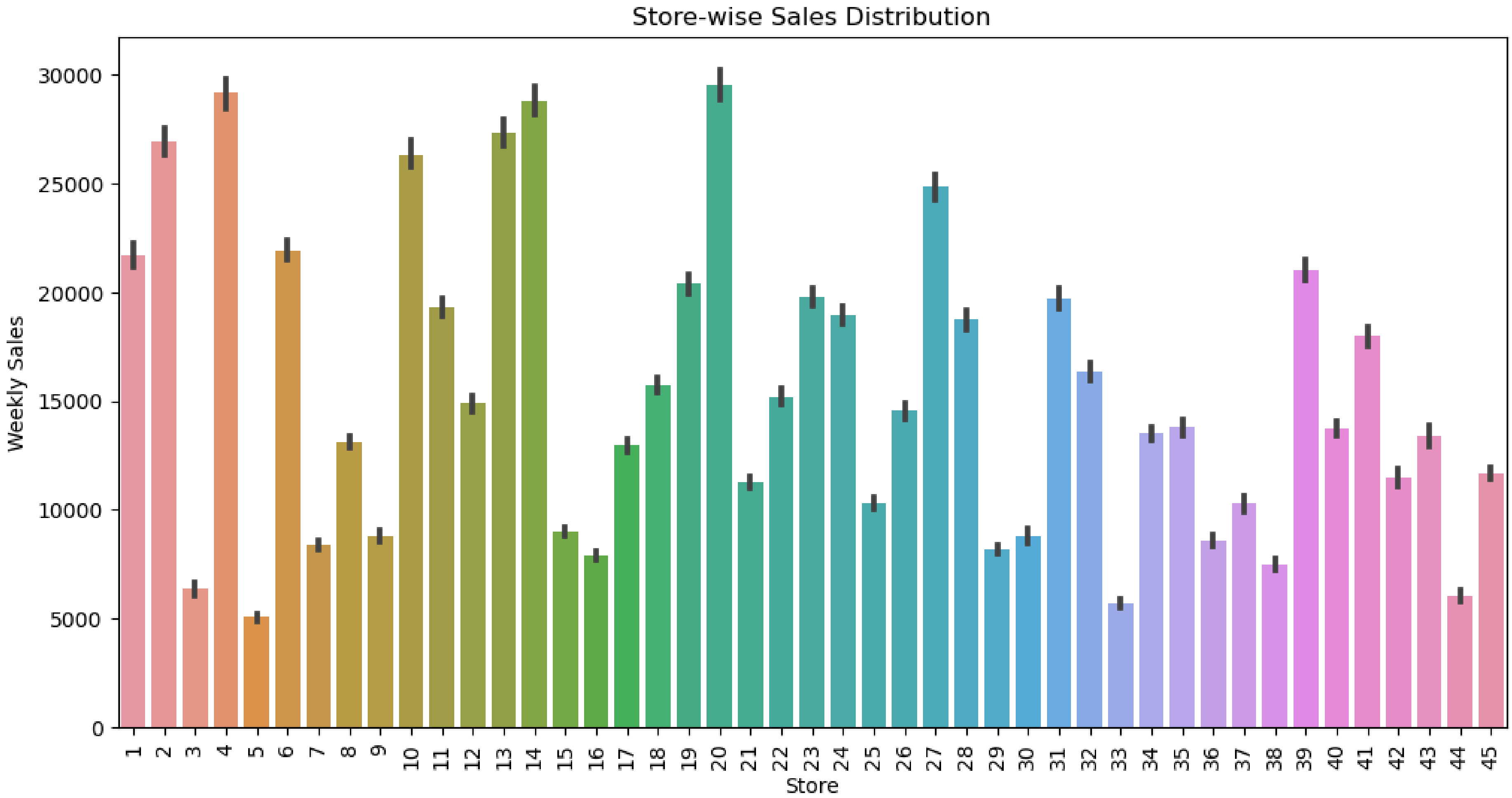
In [25]:
```
# Group data by date and calculate mean weekly sales
sales_over_time = train_set.groupby('Date')['Weekly_Sales'].mean()

# Plot sales over time
plt.figure(figsize=(12, 6))
```

```
sales_over_time.plot()
plt.title('Average Weekly Sales Over Time')
plt.xlabel('Date')
plt.ylabel('Average Weekly Sales')
plt.show()
```



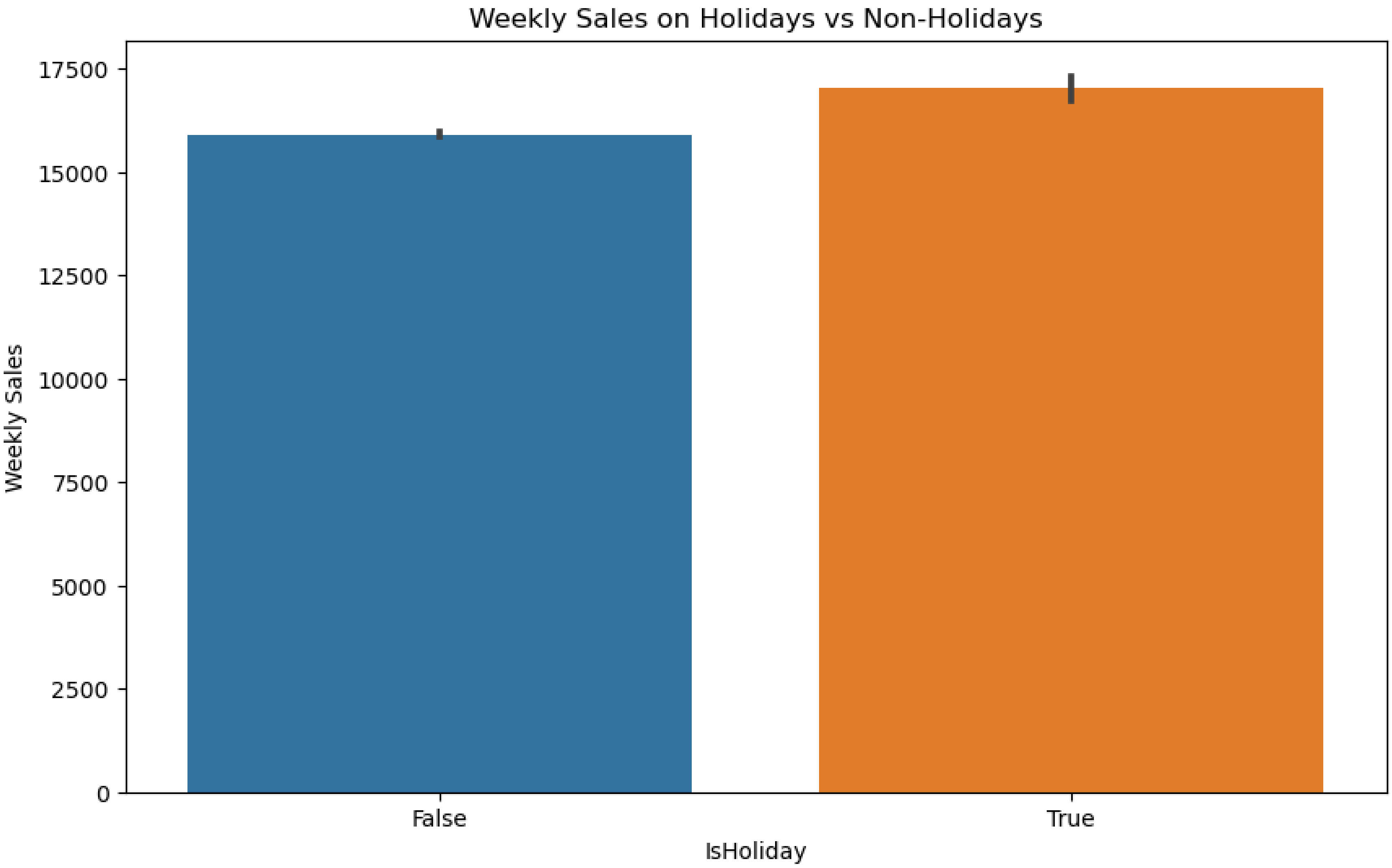
Weekly Sales by Stores

In [26]:
```
# Boxplot for Store-wise sales distribution
plt.figure(figsize=(12, 6))
sns.barplot(x='Store', y='Weekly_Sales', data=train_set)
plt.title('Store-wise Sales Distribution')
plt.xlabel('Store')
plt.ylabel('Weekly Sales')
plt.xticks(rotation=90)
plt.show()
```



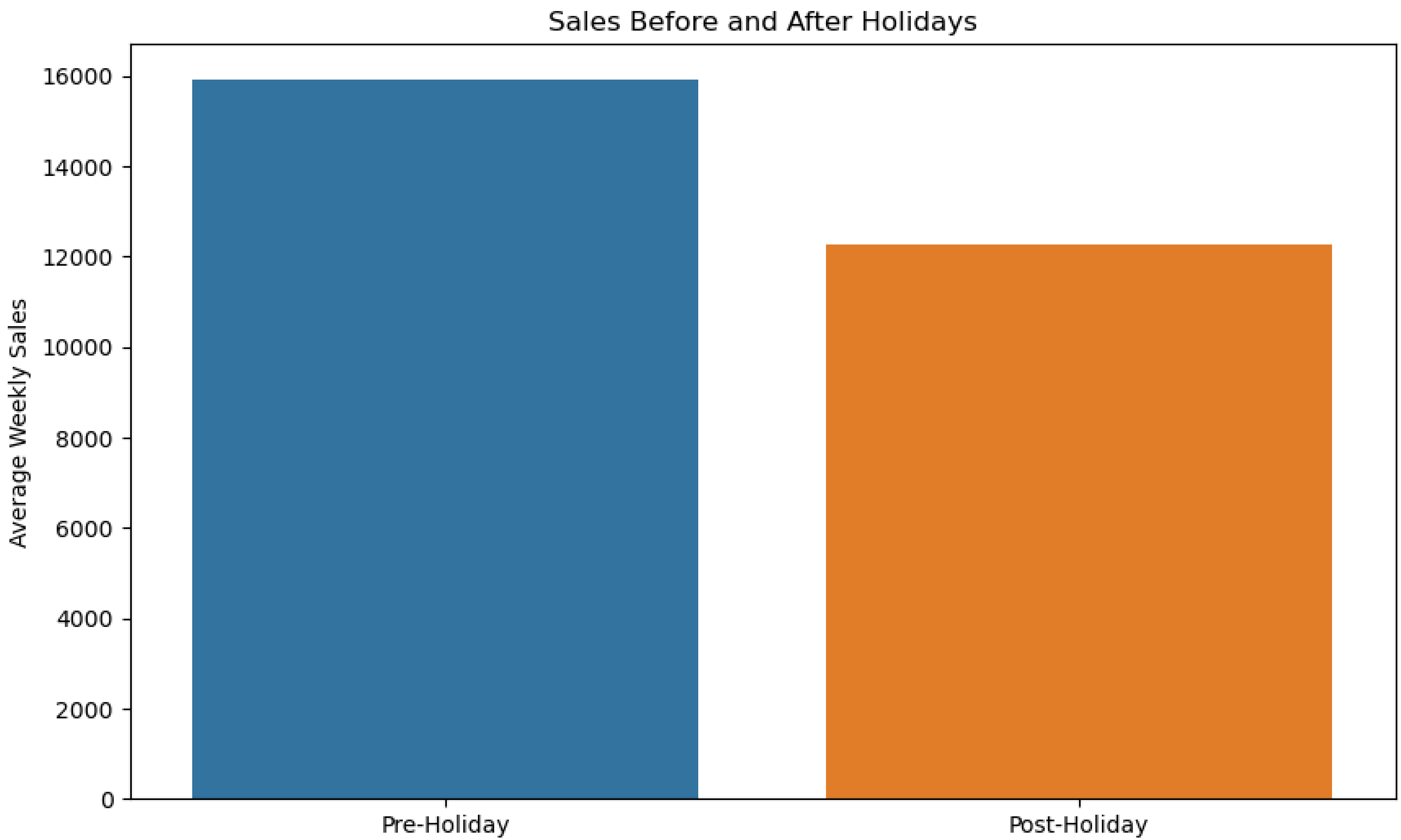
Impact of Holidays on Sales

In [27]:
```
# Boxplot for Weekly Sales on holidays vs non-holidays
plt.figure(figsize=(10, 6))
sns.barplot(x='IsHoliday', y='Weekly_Sales', data=train_set)
plt.title('Weekly Sales on Holidays vs Non-Holidays')
plt.xlabel('IsHoliday')
plt.ylabel('Weekly Sales')
plt.show()
```



Average Weekly Sales Pre-Holidays and Post-Holidays

In [28]:
```
# Shift the Date column to create pre-holiday and post-holiday analysis
train_set['Pre_Holiday'] = train_set['IsHoliday'].shift(1).fillna(False)
train_set['Post_Holiday'] = train_set['IsHoliday'].shift(-1).fillna(False)

# Group by pre-holiday and post-holiday
pre_post_sales = train_set.groupby(['Pre_Holiday', 'Post_Holiday'])['Weekly_Sales'].mean().reset_index()

# Plot pre-holiday and post-holiday sales
plt.figure(figsize=(10, 6))
sns.barplot(x=['Pre-Holiday', 'Post-Holiday'], y=[pre_post_sales['Weekly_Sales'][0], pre_post_sales['Weekly_Sales'][1]])
plt.title('Sales Before and After Holidays')
plt.ylabel('Average Weekly Sales')
plt.show()
```



In [26]:
```
train_set_holiday = train_set.loc[train_set['IsHoliday']==True]
train_set_holiday['Date'].unique()
```

Out[26]:
```
<DatetimeArray>
['2010-02-12 00:00:00', '2010-09-10 00:00:00', '2010-11-26 00:00:00',
 '2010-12-31 00:00:00', '2011-02-11 00:00:00', '2011-09-09 00:00:00',
 '2011-11-25 00:00:00', '2011-12-30 00:00:00', '2012-02-10 00:00:00',
 '2012-09-07 00:00:00']
Length: 10, dtype: datetime64[ns]
```

In [27]:
```
train_set_not_holiday = train_set.loc[train_set['IsHoliday']==False]
train_set_not_holiday['Date'].nunique()
```

Out[27]:
```
133
```

The figures do not include all holidays. There are four holiday values, which include:

Super Bowl: 12-February-10, 11-February-11, 10-February-12, and 8-February-13

Labour Day: 10-September-10, 9-September-11, 7-September-12, and 6-September-13

Thanksgiving: 26-November-10, 25-November-11, 23-November-12, and 29-November-13

Christmas is on December 31st, 30th, 28th, and 27th.

Following the festivities on September 7, 2012, a prediction test set has been created. When we look at the data, we see that average weekly sales for holidays are much greater than on non-holiday days. In train statistics, there are 133 weeks of non-holiday and 10 weeks of holiday.

I'd like to see the distinctions between vacation categories. So, I construct four new columns for different types of holidays and populate them with boolean values. If the date falls on one of these holidays, it is true; otherwise, it is false.

In [29]:
```
# Super bowl dates in train set
train_set.loc[(train_set['Date'] == '2010-02-12')|(train_set['Date'] == '2011-02-11')|(train_set['Date'] == '2012-02-10'),'Super_
train_set.loc[(train_set['Date'] != '2010-02-12')&(train_set['Date'] != '2011-02-11')&(train_set['Date'] != '2012-02-10'),'Super_
```

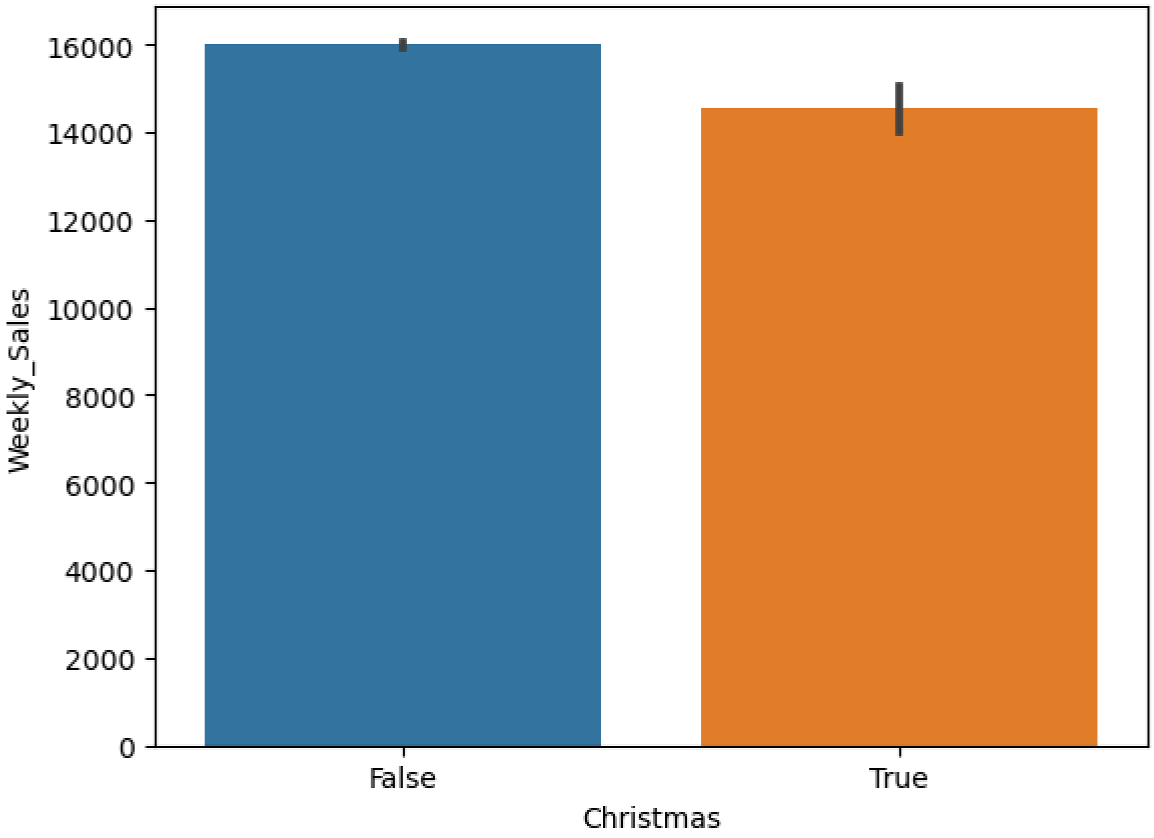
In [31]:
```
# Labor day dates in train set
train_set.loc[(train_set['Date'] == '2010-09-10')|(train_set['Date'] == '2011-09-09')|(train_set['Date'] == '2012-09-07'),'Labor_
train_set.loc[(train_set['Date'] != '2010-09-10')&(train_set['Date'] != '2011-09-09')&(train_set['Date'] != '2012-09-07'),'Labor_
```

In [33]:
```
# Thanksgiving dates in train set
train_set.loc[(train_set['Date'] == '2010-11-26')|(train_set['Date'] == '2011-11-25'),'Thanksgiving'] = True
train_set.loc[(train_set['Date'] != '2010-11-26')&(train_set['Date'] != '2011-11-25'),'Thanksgiving'] = False
```

In [34]:
```
#Christmas dates in train set
train_set.loc[(train_set['Date'] == '2010-12-31')|(train_set['Date'] == '2011-12-30'),'Christmas'] = True
train_set.loc[(train_set['Date'] != '2010-12-31')&(train_set['Date'] != '2011-12-30'),'Christmas'] = False
```

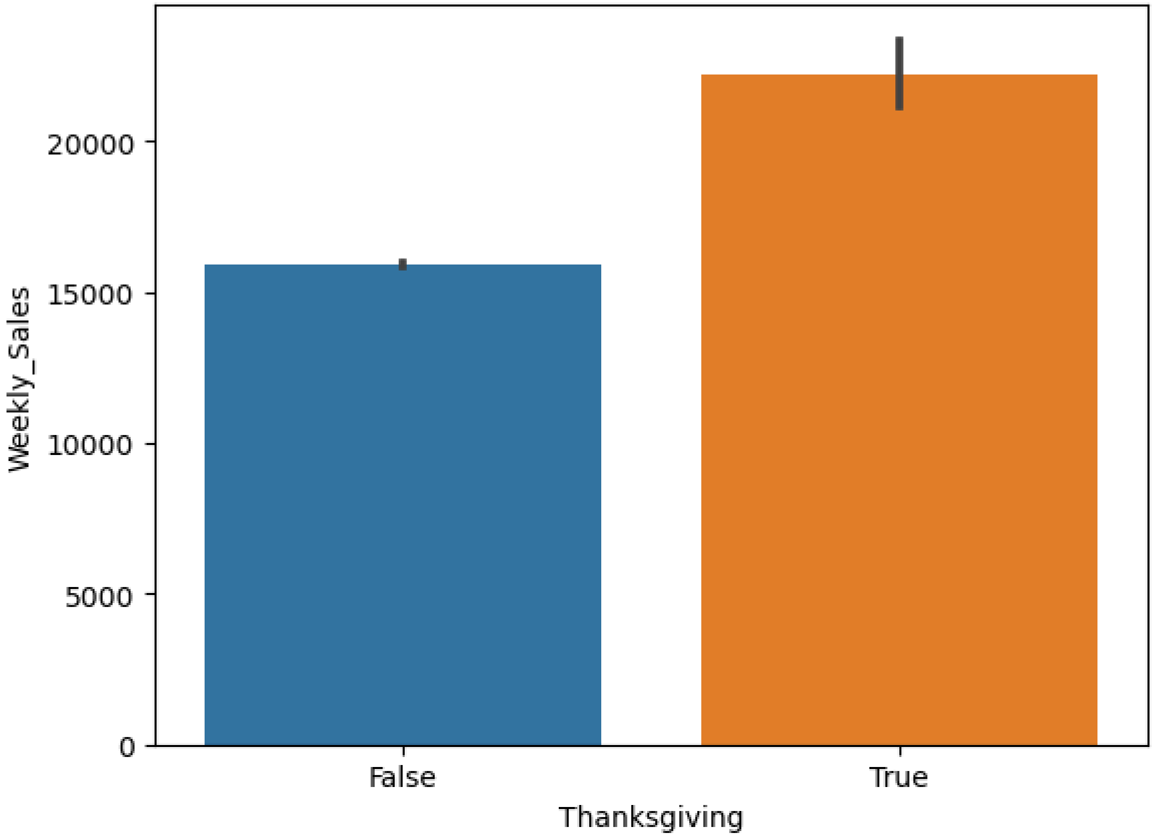
In [36]:
```
sns.barplot(x='Christmas', y='Weekly_Sales', data=train_set) # Christmas holiday vs not-Christmas
```

Out[36]: <Axes: xlabel='Christmas', ylabel='Weekly_Sales'>



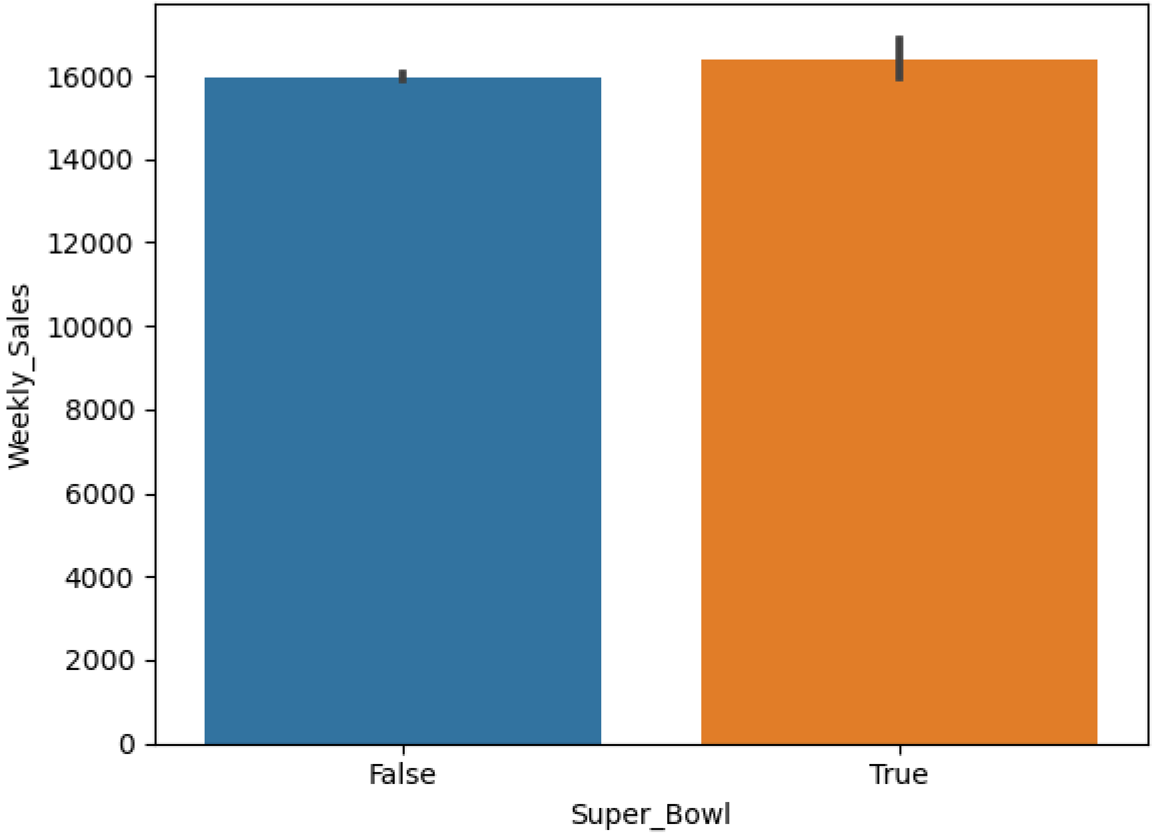
In [39]:
```
# Thanksgiving holiday vs not-thanksgiving
sns.barplot(x='Thanksgiving', y='Weekly_Sales', data=train_set )
```

Out[39]: <Axes: xlabel='Thanksgiving', ylabel='Weekly_Sales'>



In [40]:
```python
# Super bowl holiday vs not-super bowl
sns.barplot(x='Super_Bowl', y='Weekly_Sales', data=train_set)
```

Out[40]:
```
<Axes: xlabel='Super_Bowl', ylabel='Weekly_Sales'>
```

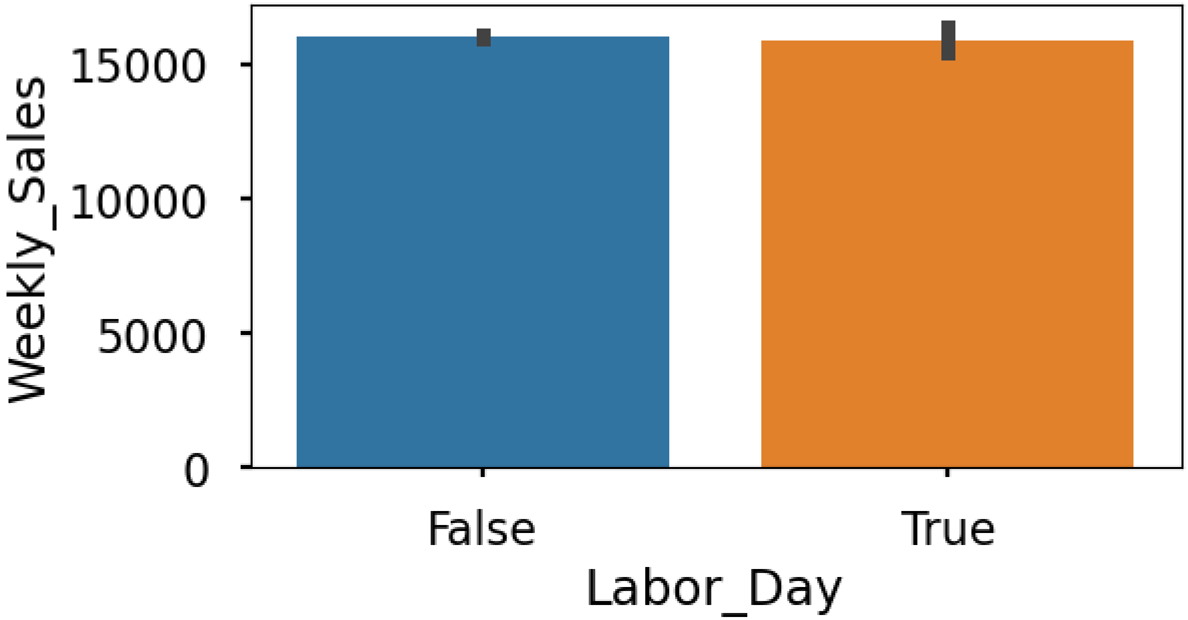


In [108…

```
# Labor day holiday vs not-labor day
plt.figure(figsize=(6,3))
sns.barplot(x='Labor_Day', y='Weekly_Sales', data=train_set)
```

Out[108]:

```
<Axes: xlabel='Labor_Day', ylabel='Weekly_Sales'>
```



The graphs illustrate that Labour Day and Christmas do not enhance weekly average sales. There is a beneficial influence on sales during the Super Bowl, but the biggest difference occurs during Thanksgiving. I believe that consumers prefer to buy Christmas gifts 1-2 weeks before the holiday, hence sales during the Christmas week are unaffected. In addition, Black Friday discounts take place during the Thanksgiving week.

In [44]:
```
# Merge datasets
df_train = pd.merge(train_set, stores, on=['Store'], how='inner')
```

In [46]:
```
df_train.head()
```

Out[46]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	...	MarkDown5	CPI	Unemploy
0	1	1	2010-02-05	24924.50	False	42.31	2.572	0.0	0.0	0.0	...	0.0	211.096358	
1	1	2	2010-02-05	50605.27	False	42.31	2.572	0.0	0.0	0.0	...	0.0	211.096358	
2	1	3	2010-02-05	13740.12	False	42.31	2.572	0.0	0.0	0.0	...	0.0	211.096358	
3	1	4	2010-02-05	39954.04	False	42.31	2.572	0.0	0.0	0.0	...	0.0	211.096358	
4	1	5	2010-02-05	32229.38	False	42.31	2.572	0.0	0.0	0.0	...	0.0	211.096358	

5 rows × 21 columns

Type Effect on Holidays

There are three different store types in the data as A, B and C.

In [47]:
```
df_train.groupby(['Christmas','Type'])['Weekly_Sales'].mean()  # Avg weekly sales for types on Christmas
```

Out[47]:
```
Christmas  Type
False      A       20126.297990
           B       12249.152357
           C        9541.691864
True       A       18231.031306
           B       11394.051524
           C        7963.228980
Name: Weekly_Sales, dtype: float64
```

In [48]:
```
df_train.groupby(['Labor_Day','Type'])['Weekly_Sales'].mean()  # Avg weekly sales for types on Labor Day
```

Out[48]:
```
Labor_Day  Type
False      A       20102.291095
           B       12241.858749
           C        9512.019024
True       A       19973.219881
           B       12013.482757
           C        9871.225746
Name: Weekly_Sales, dtype: float64
```

In [49]:
```
df_train.groupby(['Thanksgiving','Type'])['Weekly_Sales'].mean()  # Avg weekly sales for types on Thanksgiving
```

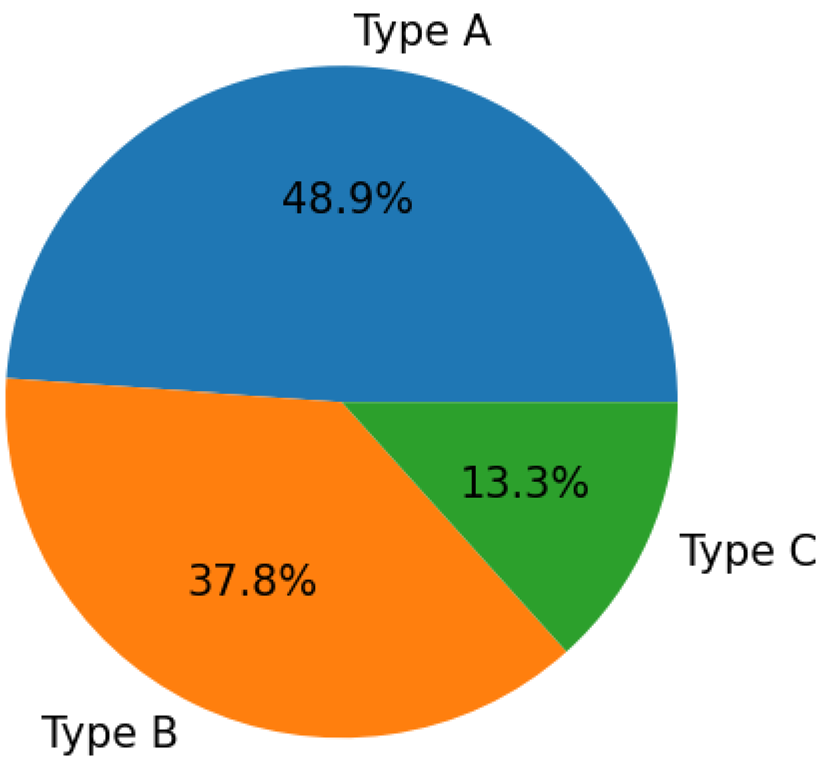
Out[49]:
```
Thanksgiving  Type
False         A       19995.309014
              B       12144.563438
              C        9517.272388
True          A       27370.728296
              B       18661.296519
              C        9679.900152
Name: Weekly_Sales, dtype: float64
```

In [ ]:

In [50]:
```
df_train.groupby(['Super_Bowl','Type'])['Weekly_Sales'].mean()  # Avg weekly sales for types on Super Bowl
```

Out[50]:
```
Super_Bowl  Type
False       A       20088.683671
            B       12233.518469
            C        9506.055492
True        A       20603.690832
            B       12401.718198
            C       10156.204711
Name: Weekly_Sales, dtype: float64
```

In [52]:
```
import matplotlib as mpl
data = [48.88, 37.77 , 13.33 ]  #percentages
my_labels = 'Type A','Type B', 'Type C' # labels
plt.pie(data,labels=my_labels,autopct='%1.1f%%', textprops={'fontsize': 15}) #plot pie type and bigger the labels
plt.axis('equal')
mpl.rcParams.update({'font.size': 20}) #bigger percentage labels

plt.show()
```



In [53]:
```
df_train.groupby('IsHoliday')['Weekly_Sales'].mean()
```

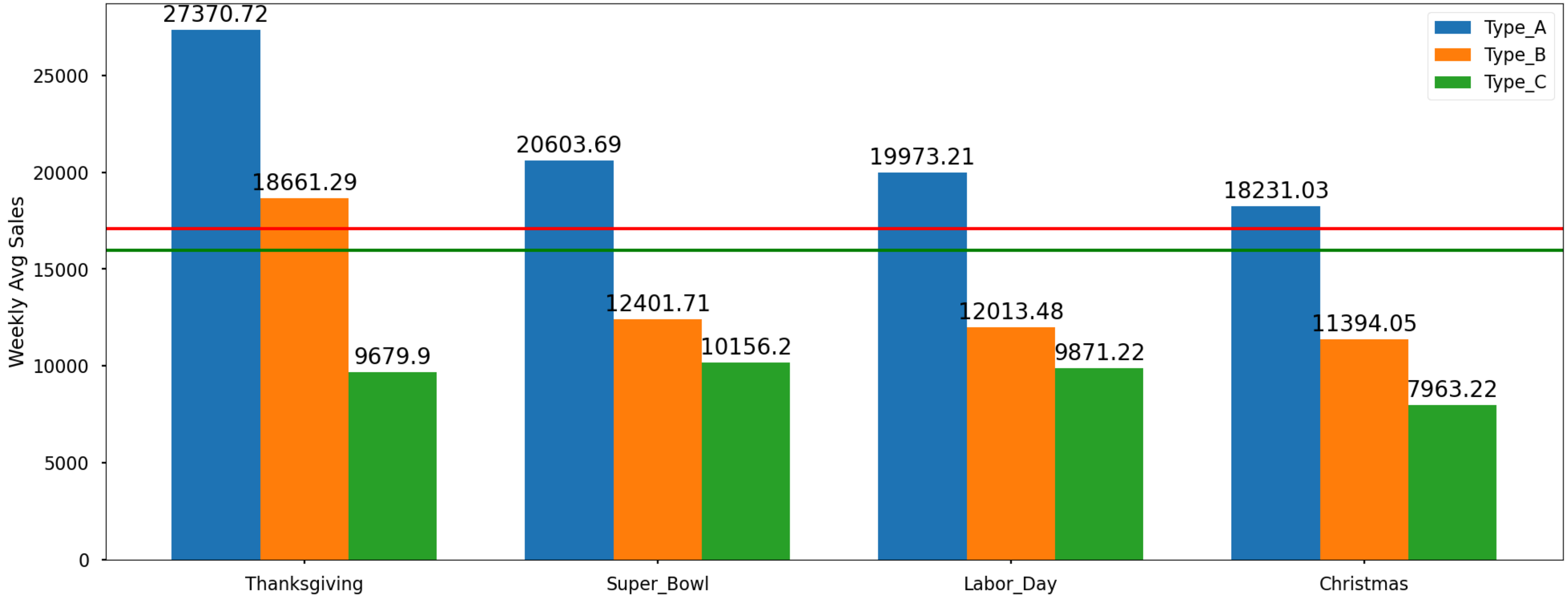
Out[53]:
```
IsHoliday
False    15901.445069
True     17035.823187
Name: Weekly_Sales, dtype: float64
```

Nearly, half of the stores are belongs to Type A.

In [60]:
```
# Plotting avg wekkly sales according to holidays by types
plt.style.use('seaborn-poster')
labels = ['Thanksgiving', 'Super_Bowl', 'Labor_Day', 'Christmas']
A_means = [27370.72, 20603.69, 19973.21, 18231.03]
B_means = [18661.29, 12401.71, 12013.48, 11394.05]
C_means = [9679.90,10156.20,9871.22,7963.22]

x = np.arange(len(labels))  # the label locations
width = 0.25  # the width of the bars
```

```
fig, ax = plt.subplots(figsize=(20, 8))
rects1 = ax.bar(x - width, A_means, width, label='Type_A')
rects2 = ax.bar(x , B_means, width, label='Type_B')
rects3 = ax.bar(x + width, C_means, width, label='Type_C')

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Weekly Avg Sales')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()


def autolabel(rects):
    """Attach a text label above each bar in *rects*, displaying its height."""
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')


autolabel(rects1)
autolabel(rects2)
autolabel(rects3)

plt.axhline(y=17094.30,color='r') # holidays avg
plt.axhline(y=15952.82,color='green') # not-holiday avg

fig.tight_layout()

plt.show()
```

```
C:\Users\aapat\AppData\Local\Temp\ipykernel_31572\2303633901.py:2: MatplotlibDeprecationWarning: The seaborn styles shipped by M
atplotlib are deprecated since 3.6, as they no longer correspond to the styles shipped by seaborn. However, they will remain ava
ilable as 'seaborn-v0_8-<style>'. Alternatively, directly use the seaborn API instead.
  plt.style.use('seaborn-poster')
```



The graph shows that the biggest sales average occurs during the Thanksgiving week in between holidays. And, for all holidays, Type A stores have the biggest sales.

In [61]:
```
df_train.sort_values(by='Weekly_Sales',ascending=False).head(5)
```

Out[61]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	...	MarkDown5	CPI	Une
90645	10	72	2010-11-26	693099.36	True	55.33	3.162	0.00	0.0	0.00	...	0.00	126.669267	
337053	35	72	2011-11-25	649770.18	True	47.88	3.492	1333.24	0.0	58563.24	...	6386.86	140.421786	
94393	10	72	2011-11-25	630999.19	True	60.68	3.760	174.72	329.0	141630.61	...	1009.98	129.836400	
333594	35	72	2010-11-26	627962.93	True	46.67	3.039	0.00	0.0	0.00	...	0.00	136.689571	
131088	14	72	2010-11-26	474330.10	True	46.15	3.039	0.00	0.0	0.00	...	0.00	182.783277	

5 rows × 21 columns

Also, the top five highest weekly sales relate to the Thanksgiving week.

Size - Type Relation

In [62]:

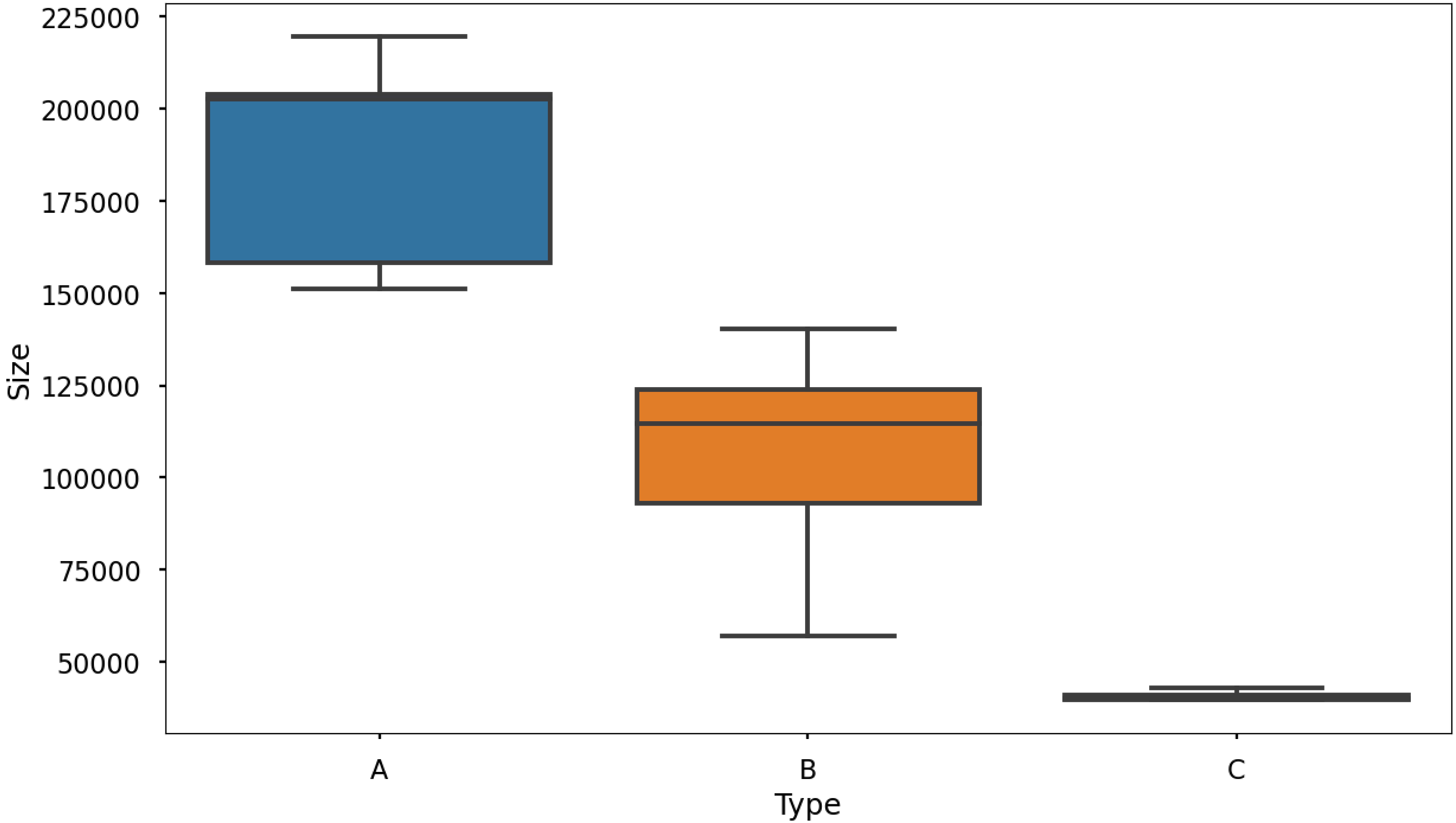
```
stores.groupby('Type').describe()['Size'].round(2) # See the Size-Type relation
```

Out[62]:

Type	count	mean	std	min	25%	50%	75%	max
A	22.0	177247.73	49392.62	39690.0	155840.75	202406.0	203819.0	219622.0
B	17.0	101190.71	32371.14	34875.0	93188.00	114533.0	123737.0	140167.0
C	6.0	40541.67	1304.15	39690.0	39745.00	39910.0	40774.0	42988.0

In [65]:

```
plt.figure(figsize=(14,8)) # To see the type-size relation
fig = sns.boxplot(x='Type', y='Size', data=df_train, showfliers=False)
```



In [69]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 421570 entries, 0 to 421569
Data columns (total 21 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  Week_of_Year  421570 non-null  UInt32
 15  Super_Bowl    421570 non-null  object
 16  Labor_Day     421570 non-null  object
 17  Thanksgiving  421570 non-null  object
 18  Christmas     421570 non-null  object
 19  Type          421570 non-null  object
 20  Size          421570 non-null  int64
dtypes: UInt32(1), bool(1), datetime64[ns](1), float64(10), int64(3), object(5)
memory usage: 63.5+ MB
```

In [75]:

```
df_train["Date"] = pd.to_datetime(df_train["Date"]) # convert to datetime
df_train['month'] =df_train['Date'].dt.month
df_train['year'] =df_train['Date'].dt.year
```

In [76]:

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 421570 entries, 0 to 421569
Data columns (total 23 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  Week_of_Year  421570 non-null  UInt32
 15  Super_Bowl    421570 non-null  object
 16  Labor_Day     421570 non-null  object
 17  Thanksgiving  421570 non-null  object
 18  Christmas     421570 non-null  object
 19  Type          421570 non-null  object
 20  Size          421570 non-null  int64
 21  month         421570 non-null  int32
 22  year          421570 non-null  int32
dtypes: UInt32(1), bool(1), datetime64[ns](1), float64(10), int32(2), int64(3), object(5)
memory usage: 66.7+ MB
```

In [77]:
```
df_train.head()
```

Out[77]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	...	Unemployment	Week_of_Year	Sup
0	1	1	2010-02-05	24924.50	False	42.31	2.572	0.0	0.0	0.0	...	8.106	5	
1	1	2	2010-02-05	50605.27	False	42.31	2.572	0.0	0.0	0.0	...	8.106	5	
2	1	3	2010-02-05	13740.12	False	42.31	2.572	0.0	0.0	0.0	...	8.106	5	
3	1	4	2010-02-05	39954.04	False	42.31	2.572	0.0	0.0	0.0	...	8.106	5	
4	1	5	2010-02-05	32229.38	False	42.31	2.572	0.0	0.0	0.0	...	8.106	5	

5 rows × 23 columns

In [79]:
```
df_train.groupby('month')['Weekly_Sales'].mean() # to see the best months for sales
```

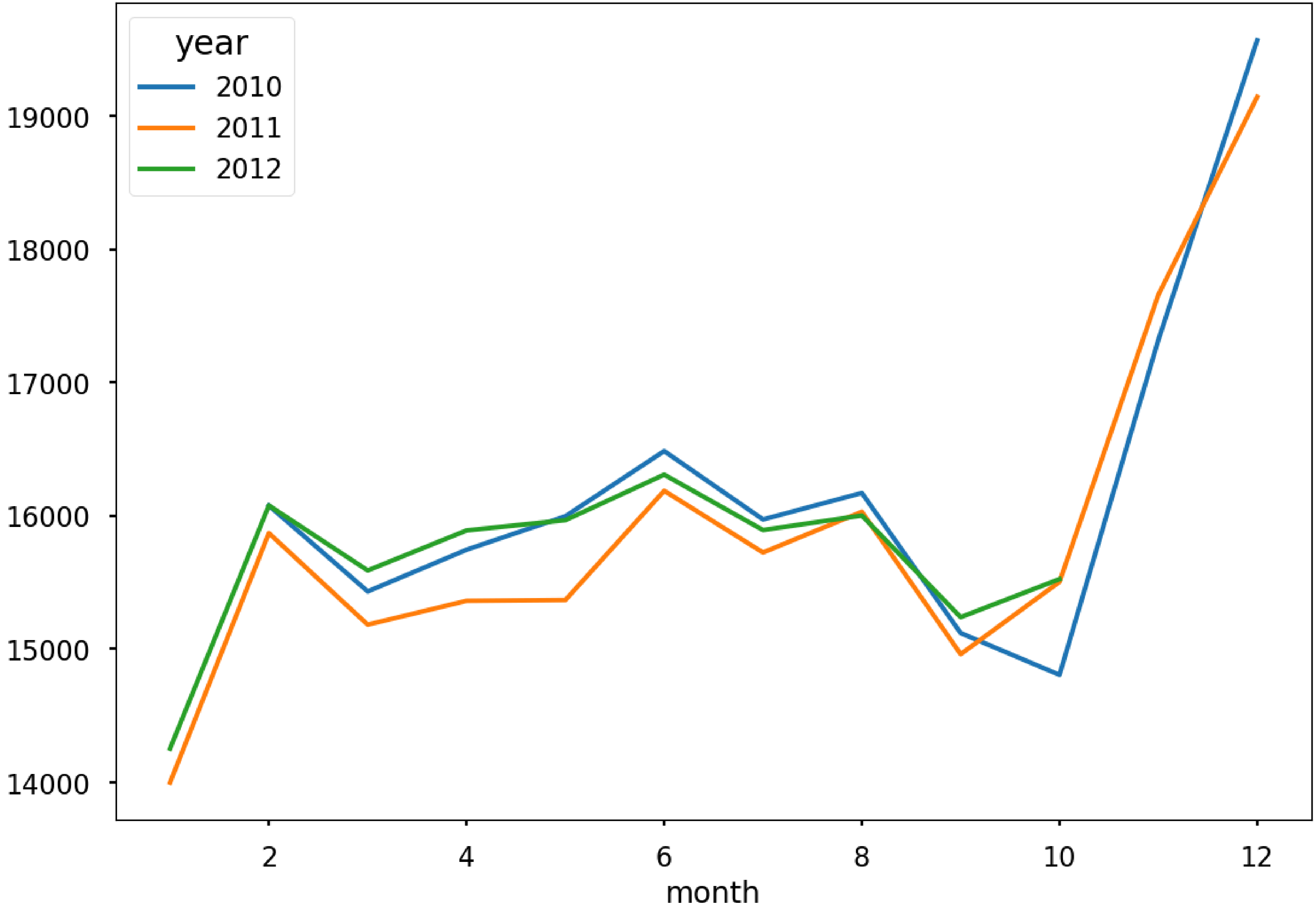
Out[79]:
```
month
1     14126.075111
2     16008.779217
3     15416.657597
4     15650.338357
5     15776.337202
6     16326.137002
7     15861.419650
8     16062.516933
9     15095.886154
10    15243.855576
11    17491.031424
12    19355.702141
Name: Weekly_Sales, dtype: float64
```

In [80]:
```
df_train.groupby('year')['Weekly_Sales'].mean() # to see the best years for sales
```

Out[80]:
```
year
2010    16270.275737
2011    15954.070675
2012    15694.948597
Name: Weekly_Sales, dtype: float64
```

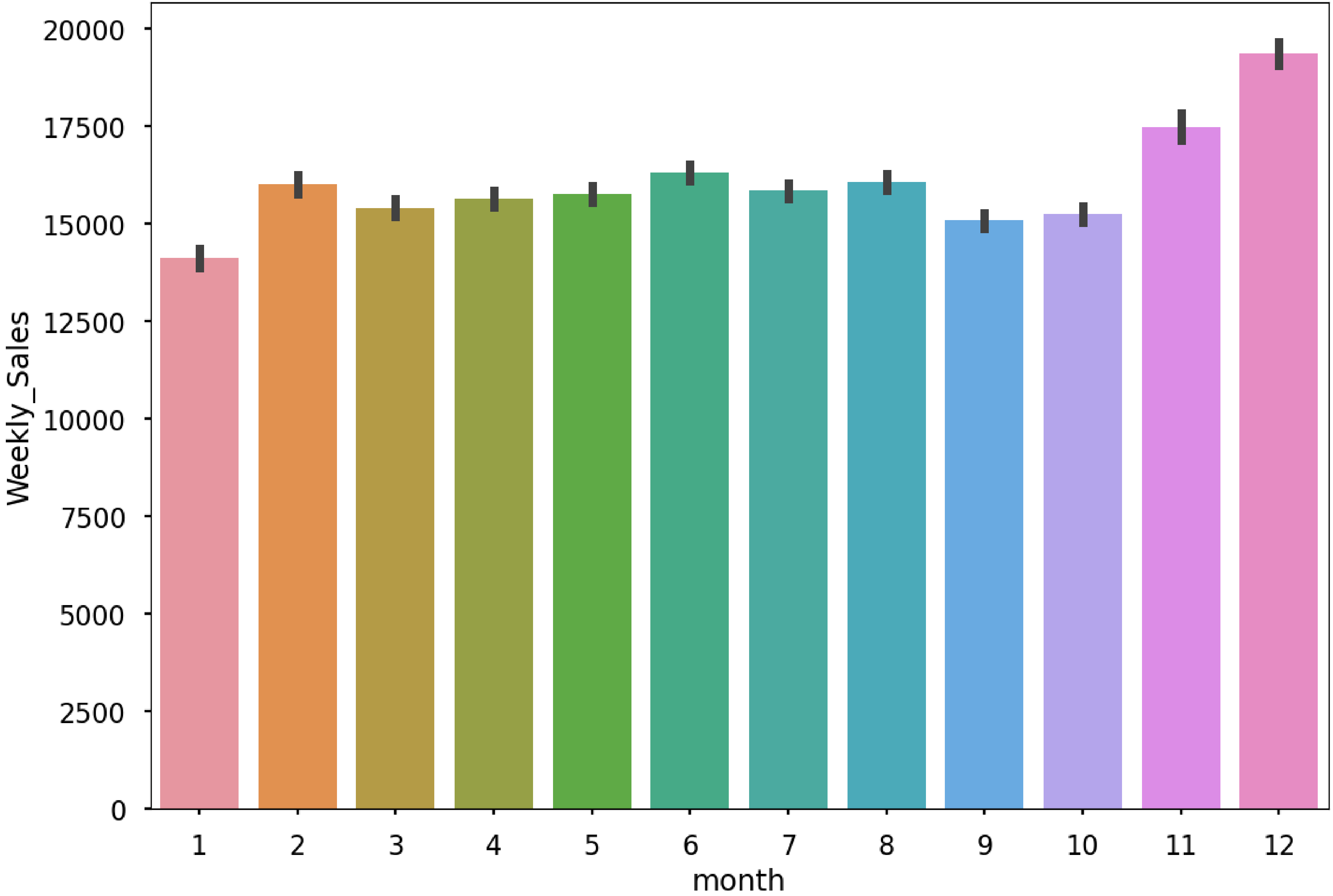
In [85]:
```
monthly_sales = pd.pivot_table(df_train, values = "Weekly_Sales", columns = "year", index = "month")
monthly_sales.plot()
```

Out[85]:
```
<Axes: xlabel='month'>
```



The graph shows that 2011 sales were lower than 2010 in general. When we look at the mean sales, we can see that 2010 had greater numbers, but 2012 lacks statistics on the higher sales months of November and December. Despite the fact that 2012 had no last two months' sales, the average is similar to 2010. If we add the 2012 results, it will most likely come in top.

In [87]:
```
fig = sns.barplot(x='month', y='Weekly_Sales', data=df_train)
```



According to the graph above, the strongest months for sales are December and November. The biggest values belong to the Thanksgiving holiday, but when we consider an average, it is clear that December has the finest value.

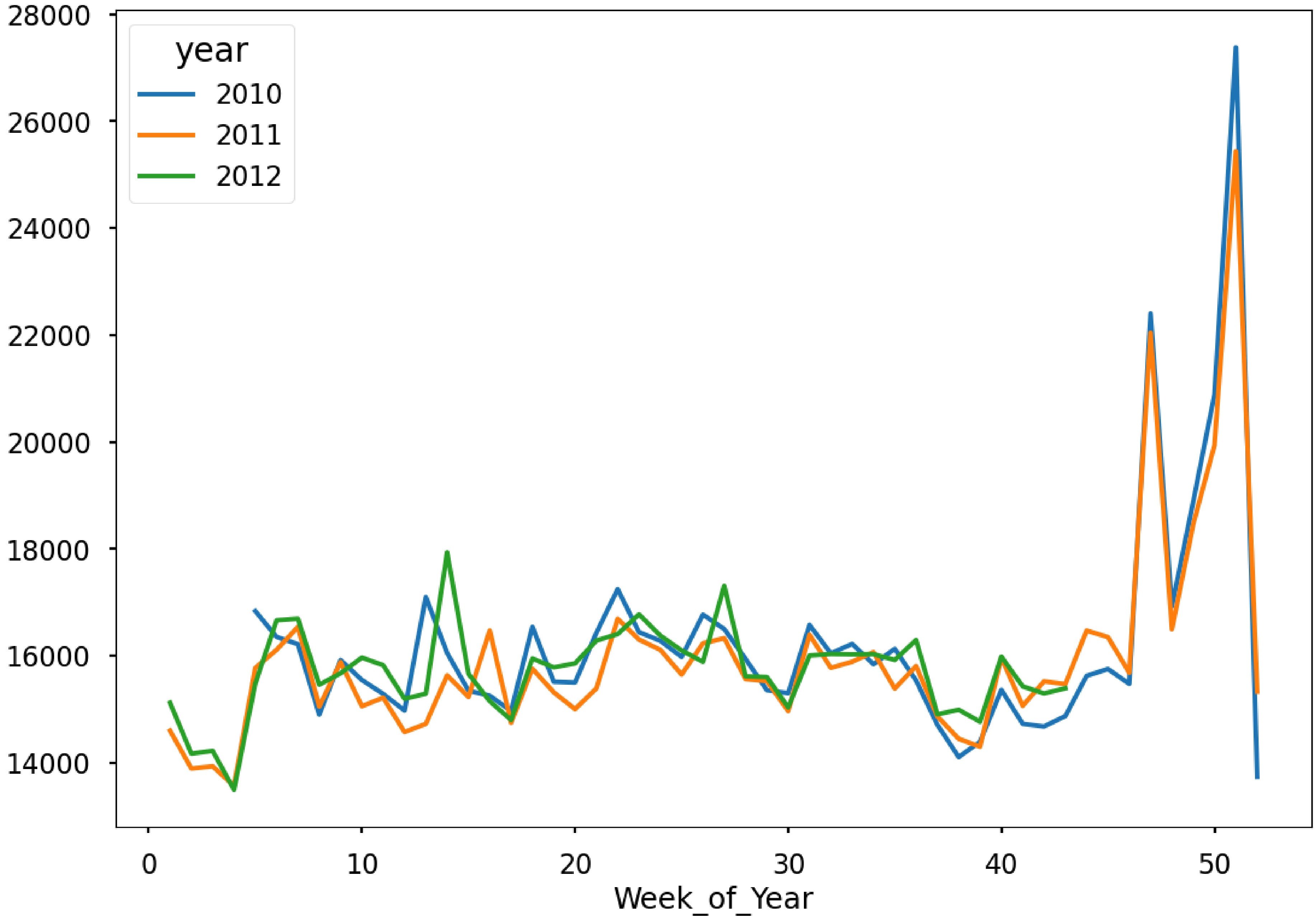
In [88]:
```
df_train.groupby('Week_of_Year')['Weekly_Sales'].mean().sort_values(ascending=False).head()
```

Out[88]:
```
Week_of_Year
51    26396.399283
47    22220.944538
50    20413.010012
49    18668.667017
22    16779.736413
Name: Weekly_Sales, dtype: float64
```

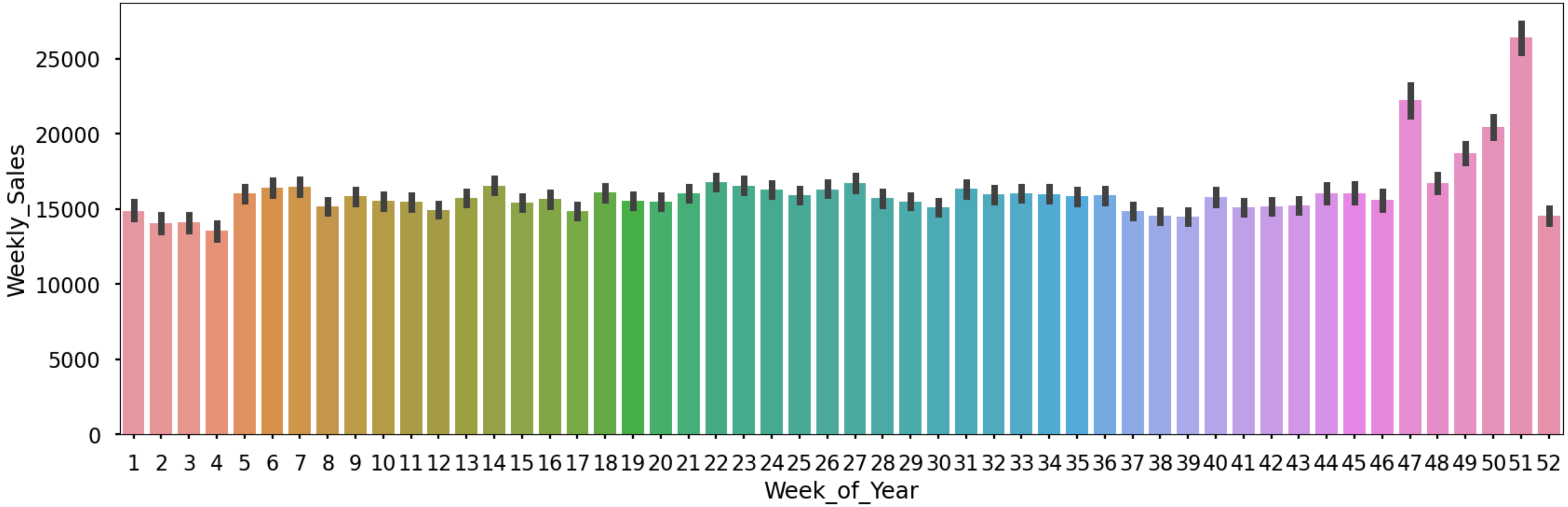
The top five sales averages by week are from 1-2 weeks before Christmas, Thanksgiving, Black Friday, and the end of May, when schools are closed.

In [89]:
```
weekly_sales = pd.pivot_table(df_train, values = "Weekly_Sales", columns = "year", index = "Week_of_Year")
weekly_sales.plot()
```

Out[89]:
```
<Axes: xlabel='Week_of_Year'>
```



In [92]:
```
plt.figure(figsize=(20,6))
fig = sns.barplot(x='Week_of_Year', y='Weekly_Sales', data=df_train)
```

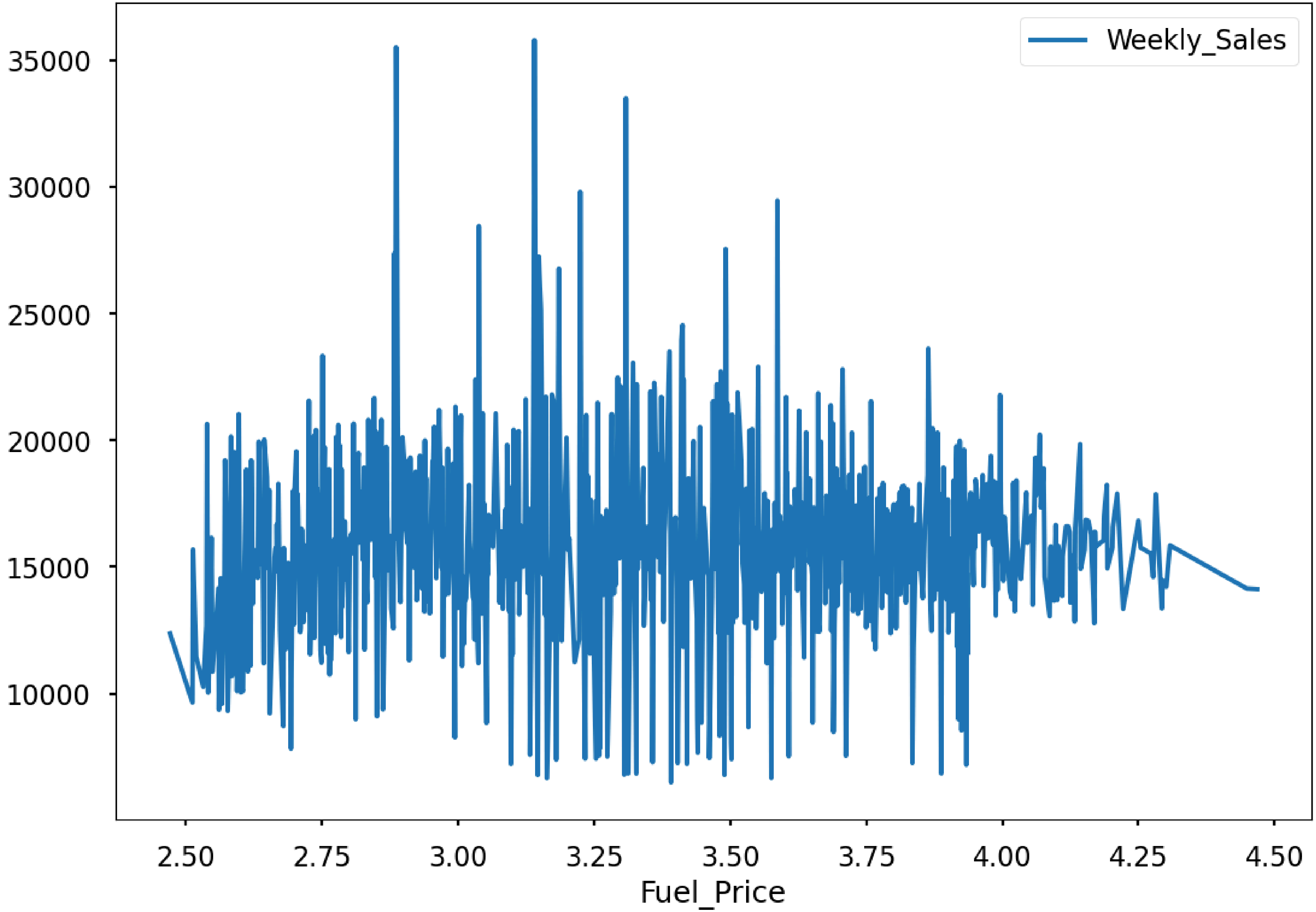


The graphs show that the 51st and 47th weeks have much higher averages due to the Christmas, Thanksgiving, and Black Friday effects.

Fuel Price, CPI , Unemployment , Temperature Effects

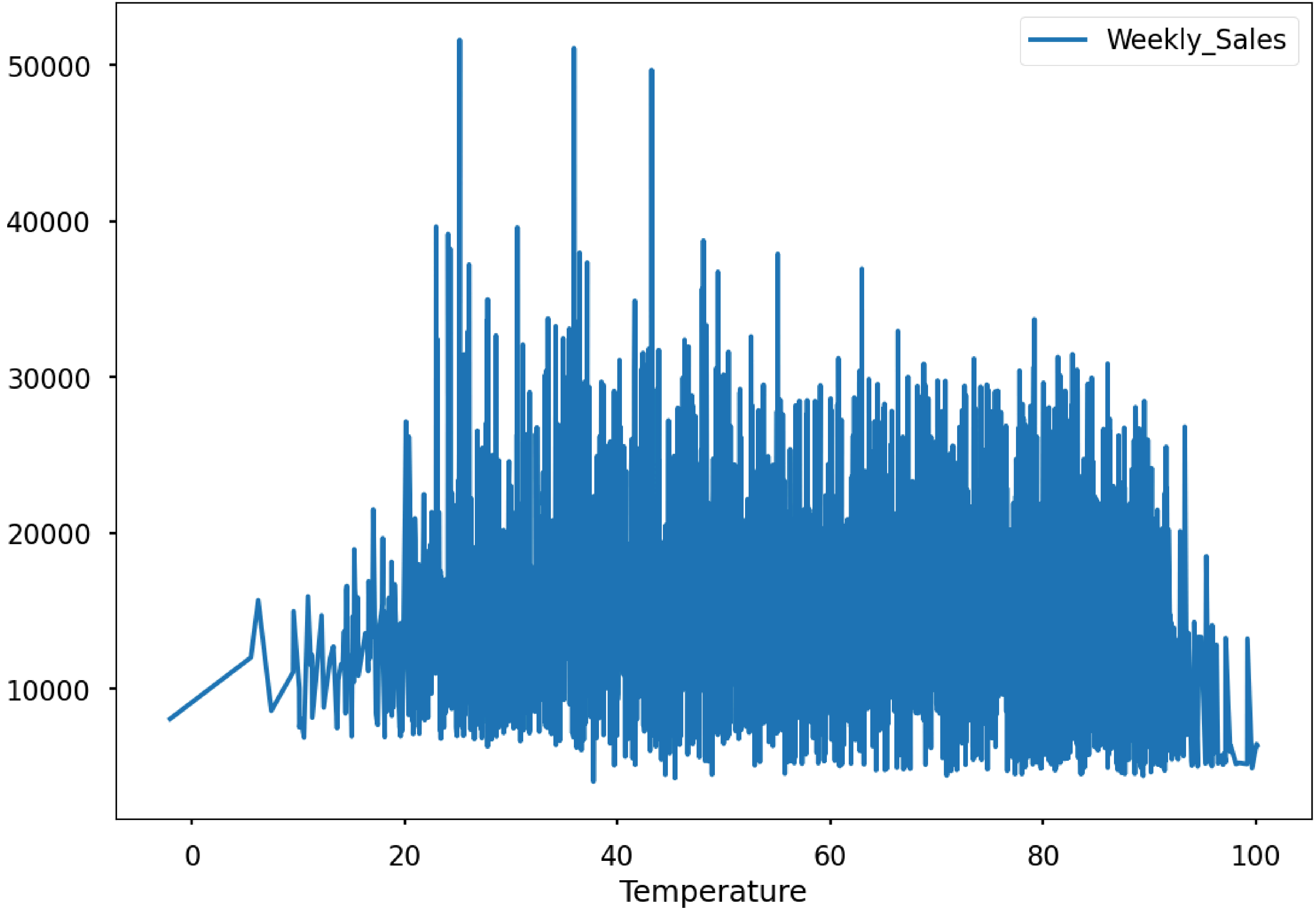
In [94]:
```
fuel_price = pd.pivot_table(df_train, values = "Weekly_Sales", index= "Fuel_Price")
fuel_price.plot()
```

Out[94]:
```
<Axes: xlabel='Fuel_Price'>
```



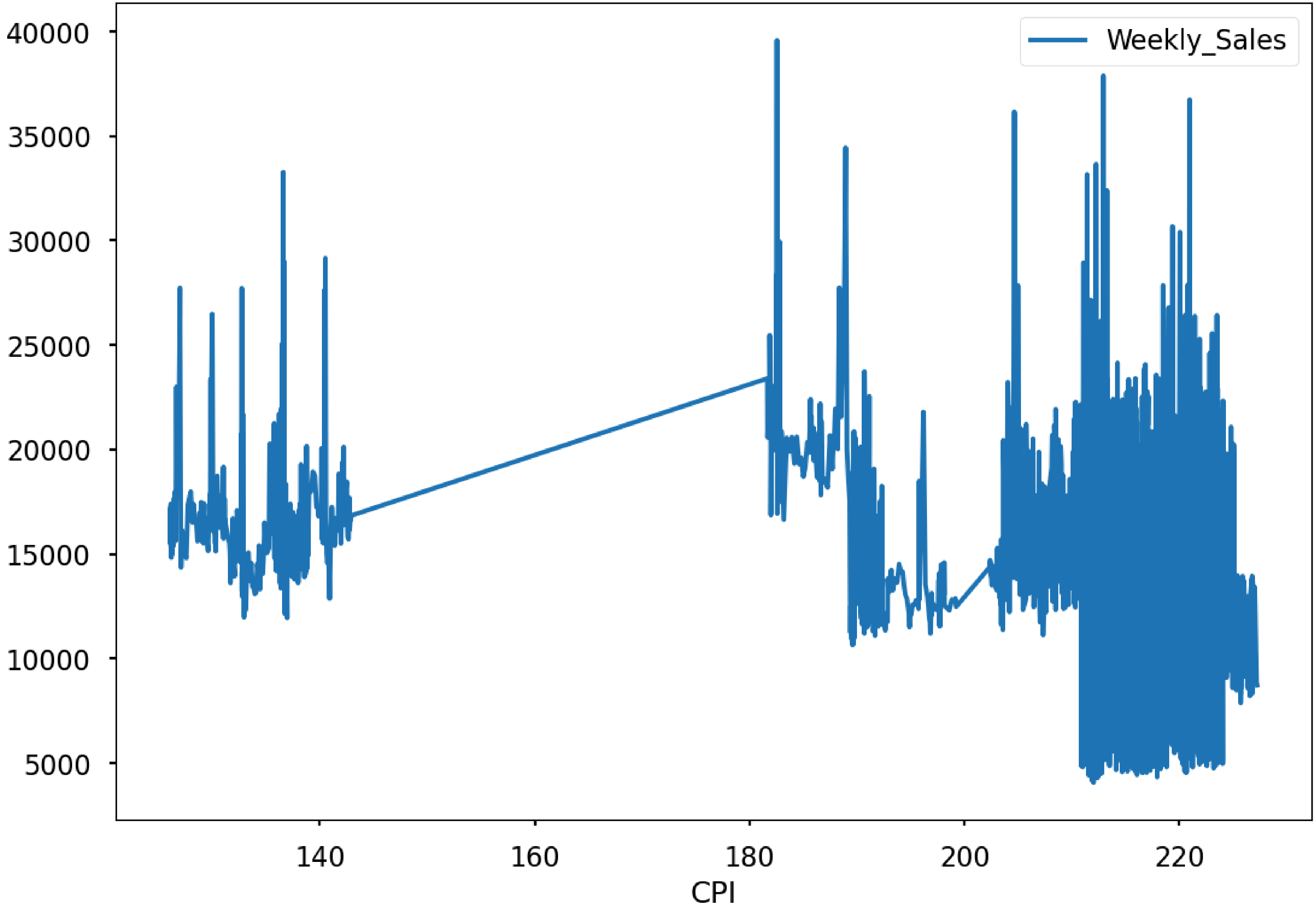
In [95]:
```
temp = pd.pivot_table(df_train, values = "Weekly_Sales", index= "Temperature")
temp.plot()
```

Out[95]: <Axes: xlabel='Temperature'>



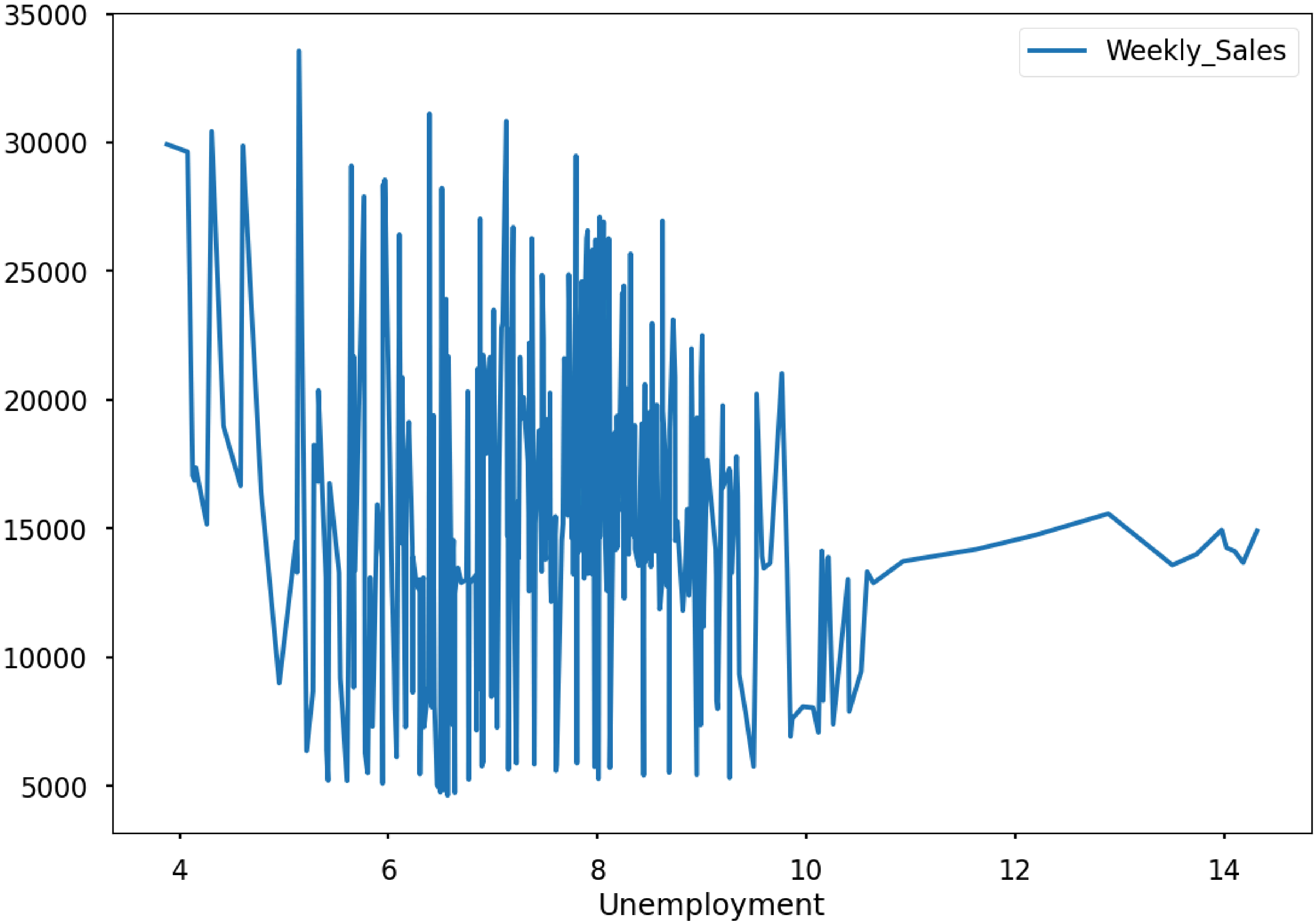
In [96]:
```
CPI = pd.pivot_table(df_train, values = "Weekly_Sales", index= "CPI")
CPI.plot()
```

Out[96]:
```
<Axes: xlabel='CPI'>
```



In [97]:
```
unemployment = pd.pivot_table(df_train, values = "Weekly_Sales", index= "Unemployment")
unemployment.plot()
```

Out[97]: <Axes: xlabel='Unemployment'>



The graphs show that there are no significant connections between CPI, temperature, unemployment rate, fuel price, and weekly sales. There is no data for CPI between 140 and 180.

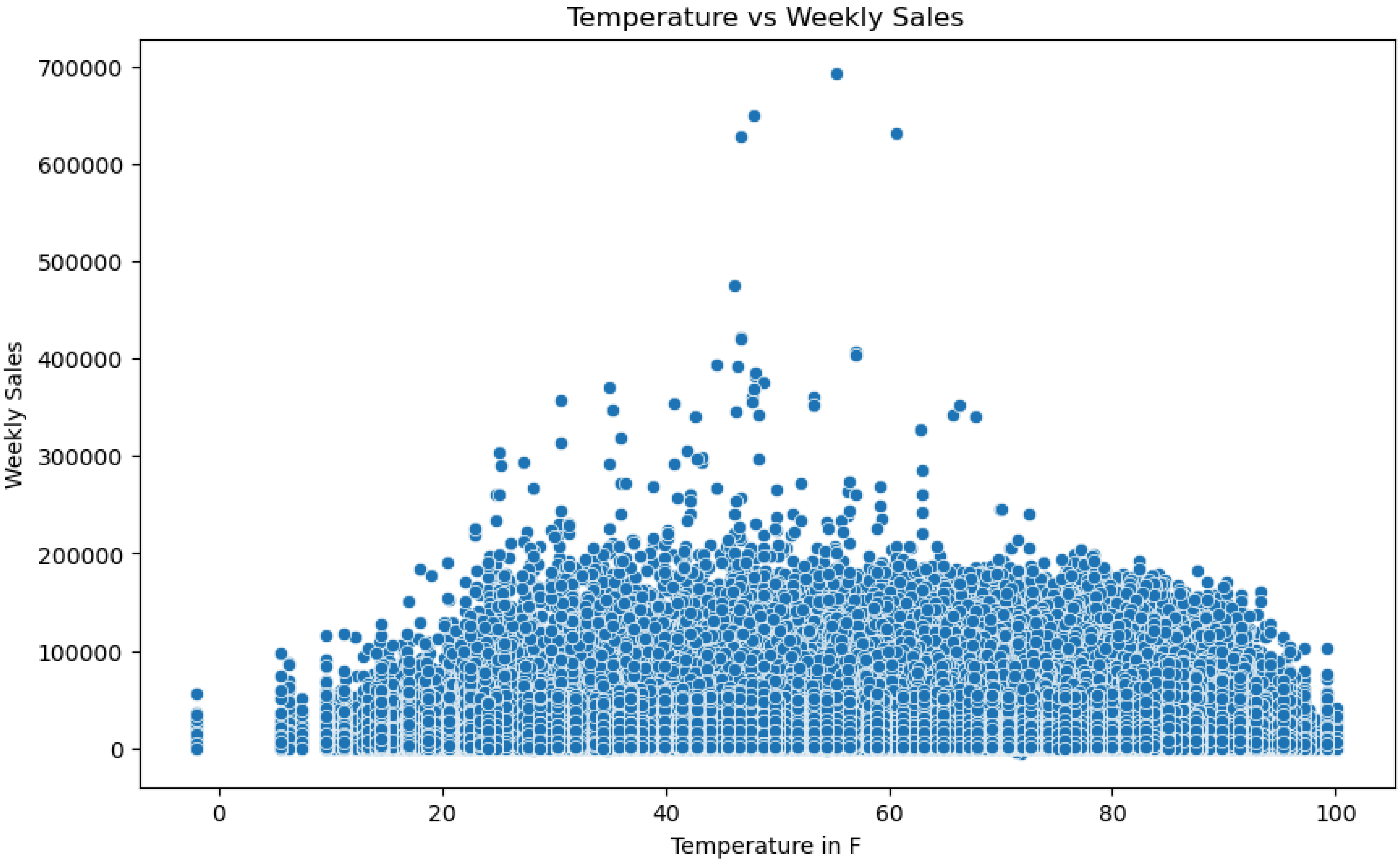
In [ ]:

Temperature vs Weekly Sales

In [ ]:

In [29]:

```
# Scatter plot for Temperature vs Weekly Sales
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Temperature', y='Weekly_Sales', data=train_set)
plt.title('Temperature vs Weekly Sales')
plt.xlabel('Temperature in F')
plt.ylabel('Weekly Sales')
plt.show()
```



In [30]:

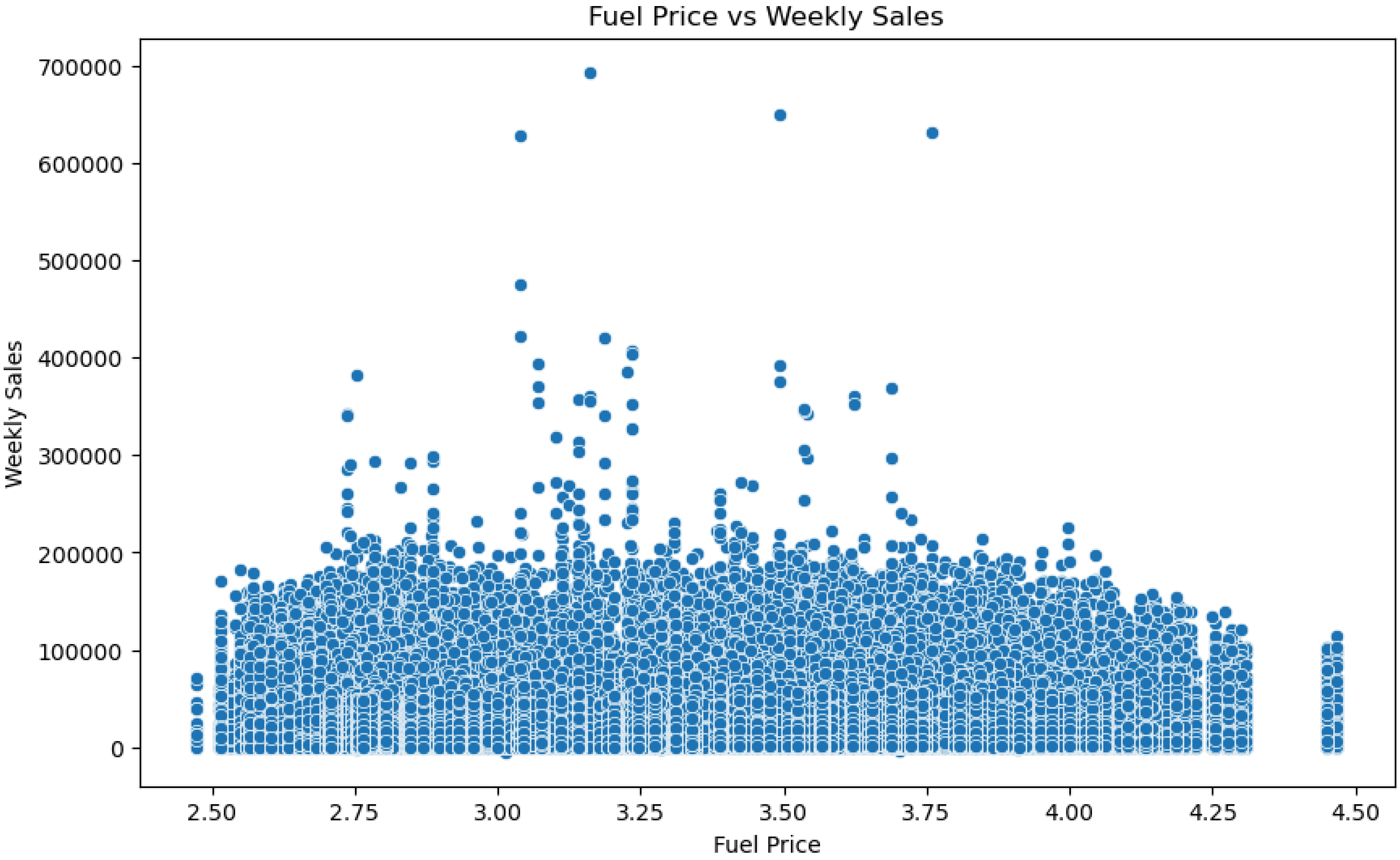
```

```

Fuel Price vs Weekly Sales

In [41]:

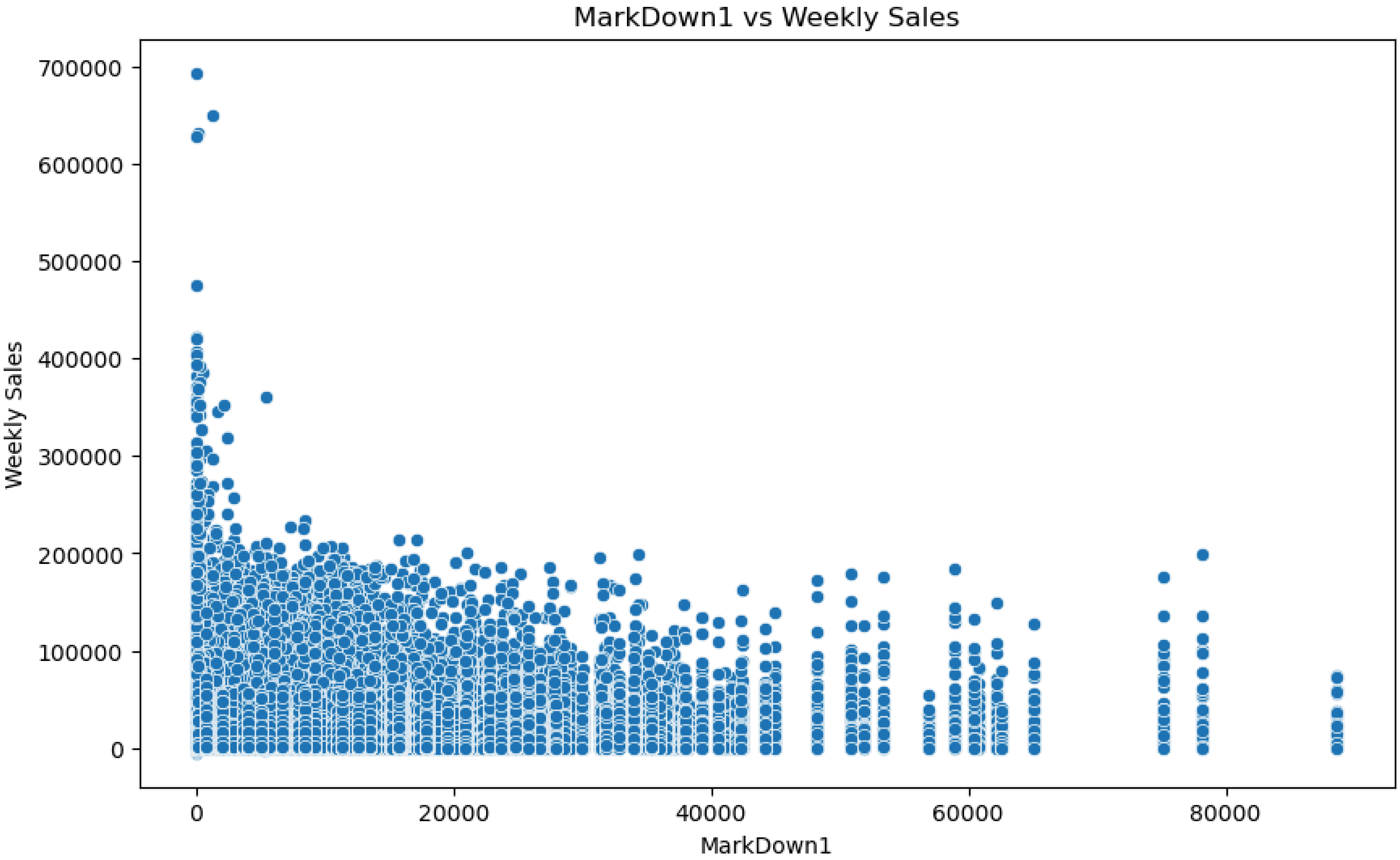
```
# Scatter plot for Fuel Price vs Weekly Sales
plt.figure(figsize=(10, 6))
```

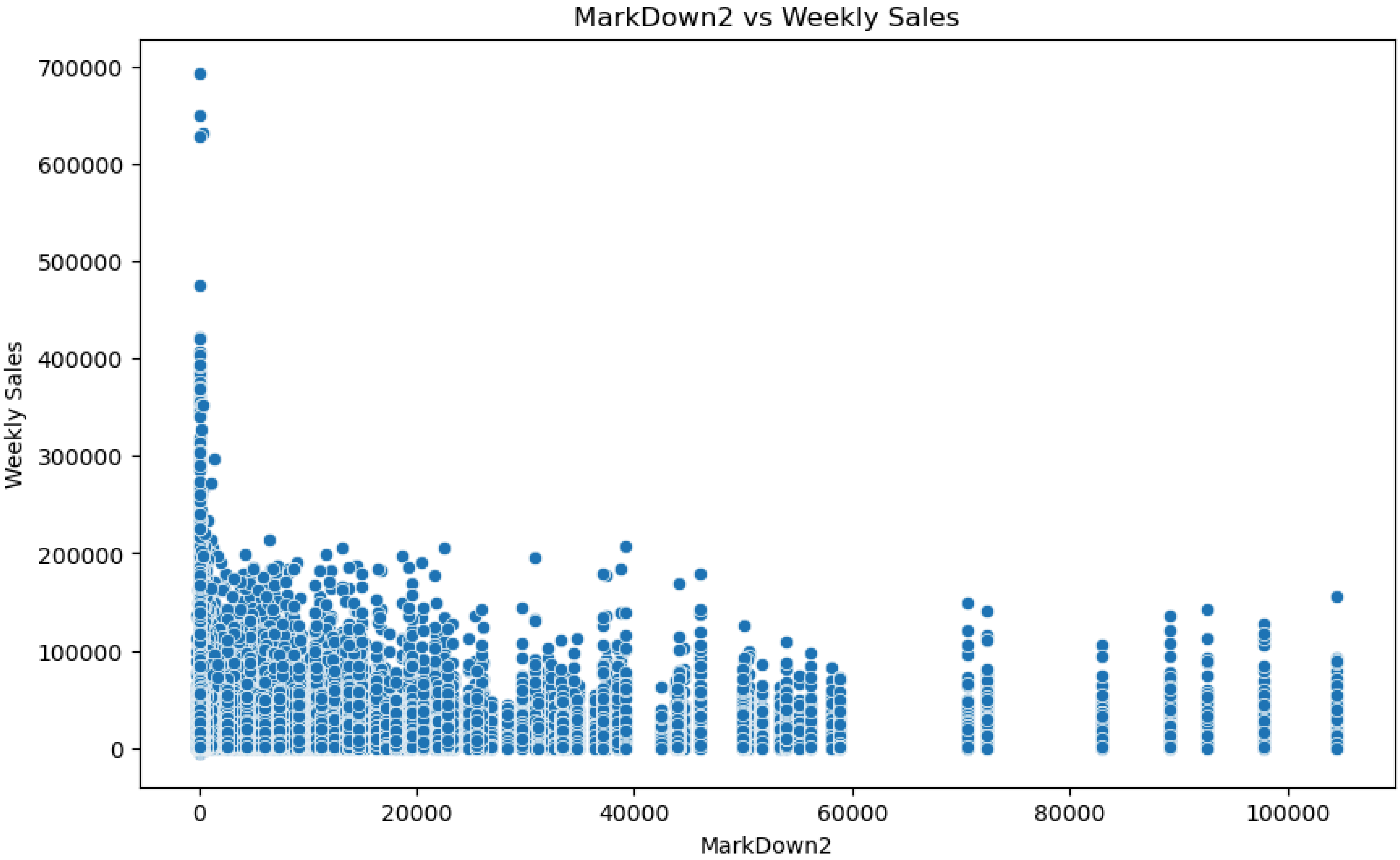
```
sns.scatterplot(x='Fuel_Price', y='Weekly_Sales', data=train_set)
plt.title('Fuel Price vs Weekly Sales')
plt.xlabel('Fuel Price')
plt.ylabel('Weekly Sales')
plt.show()
```

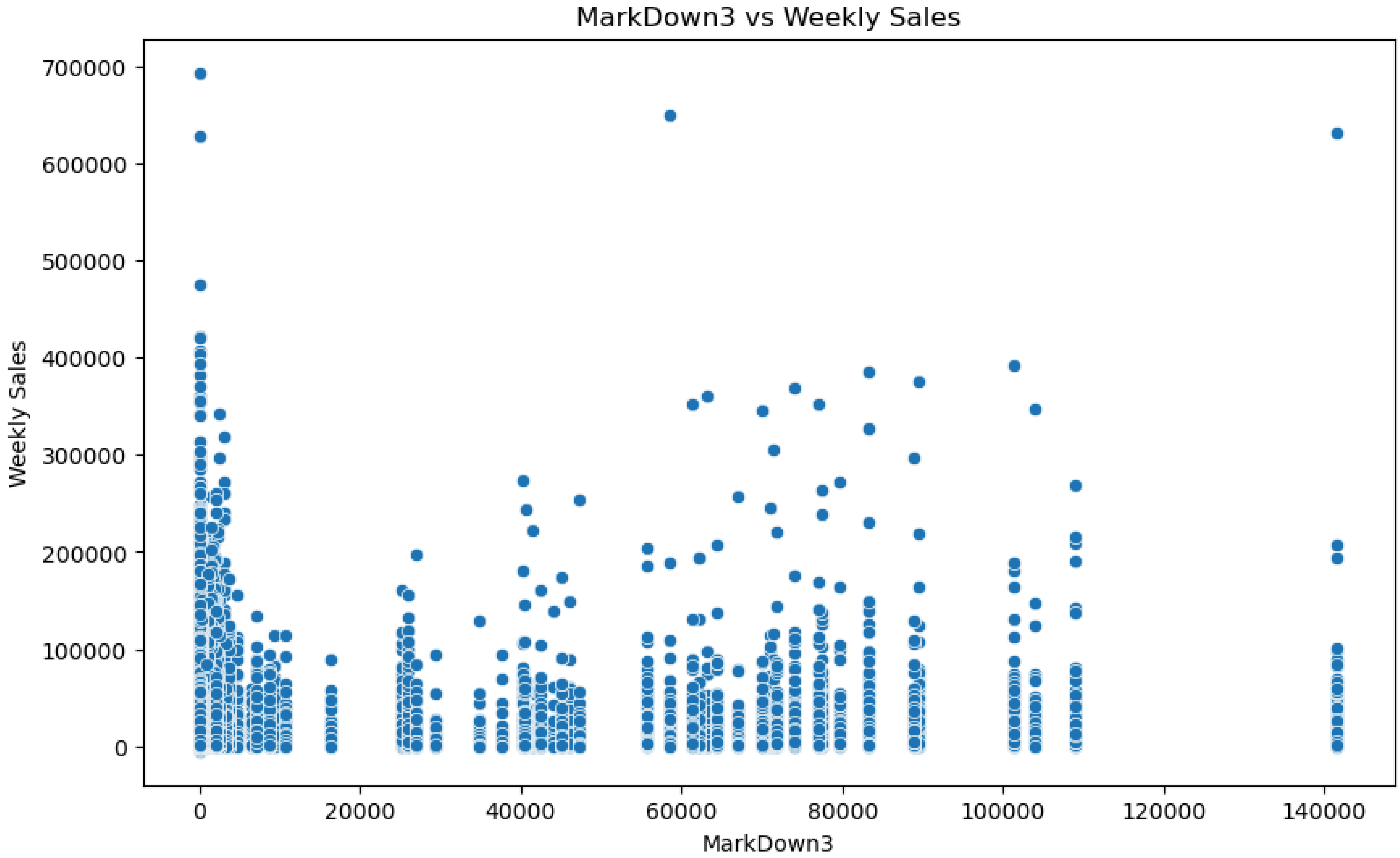


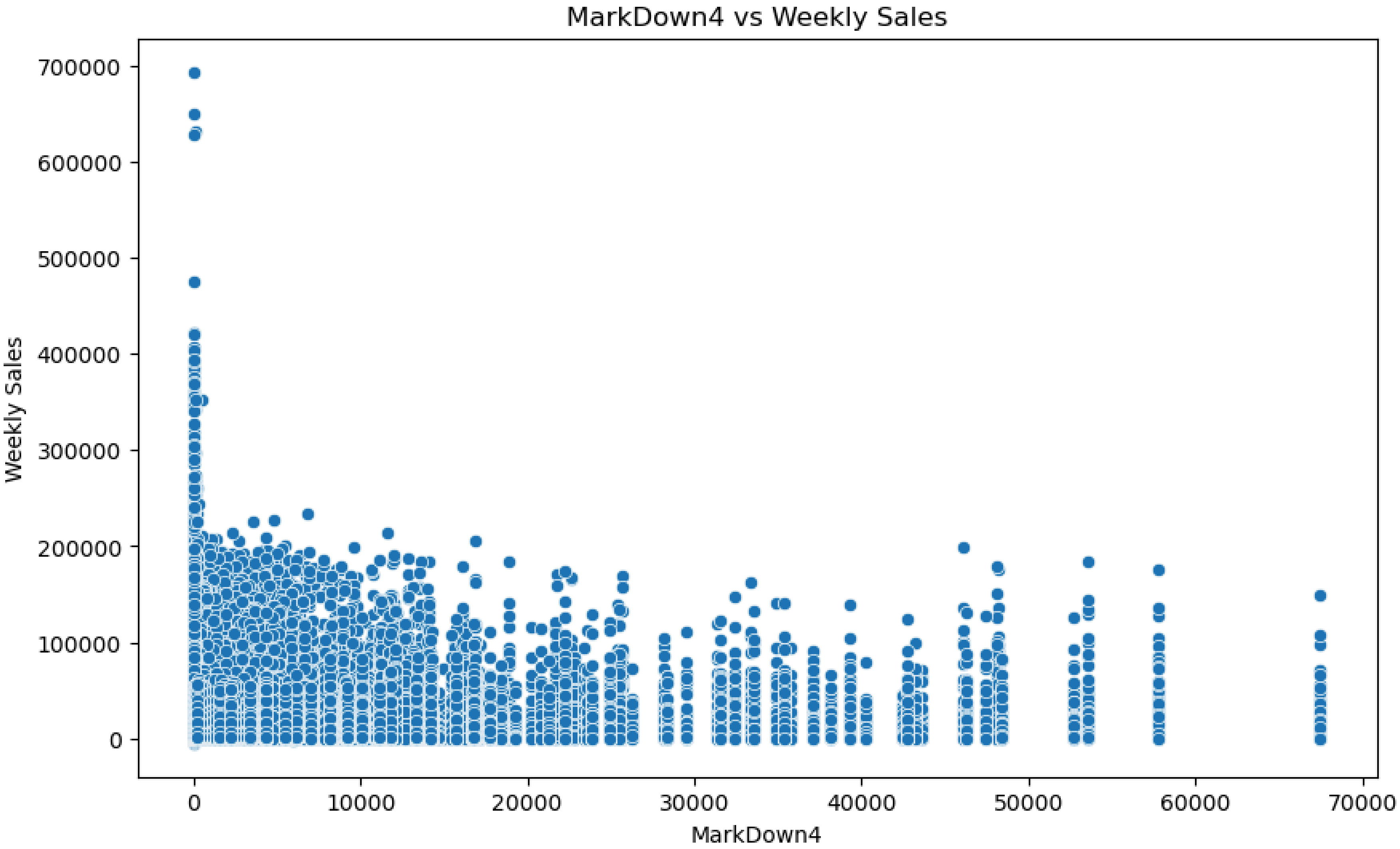
Markdowns vs Weekly Sales

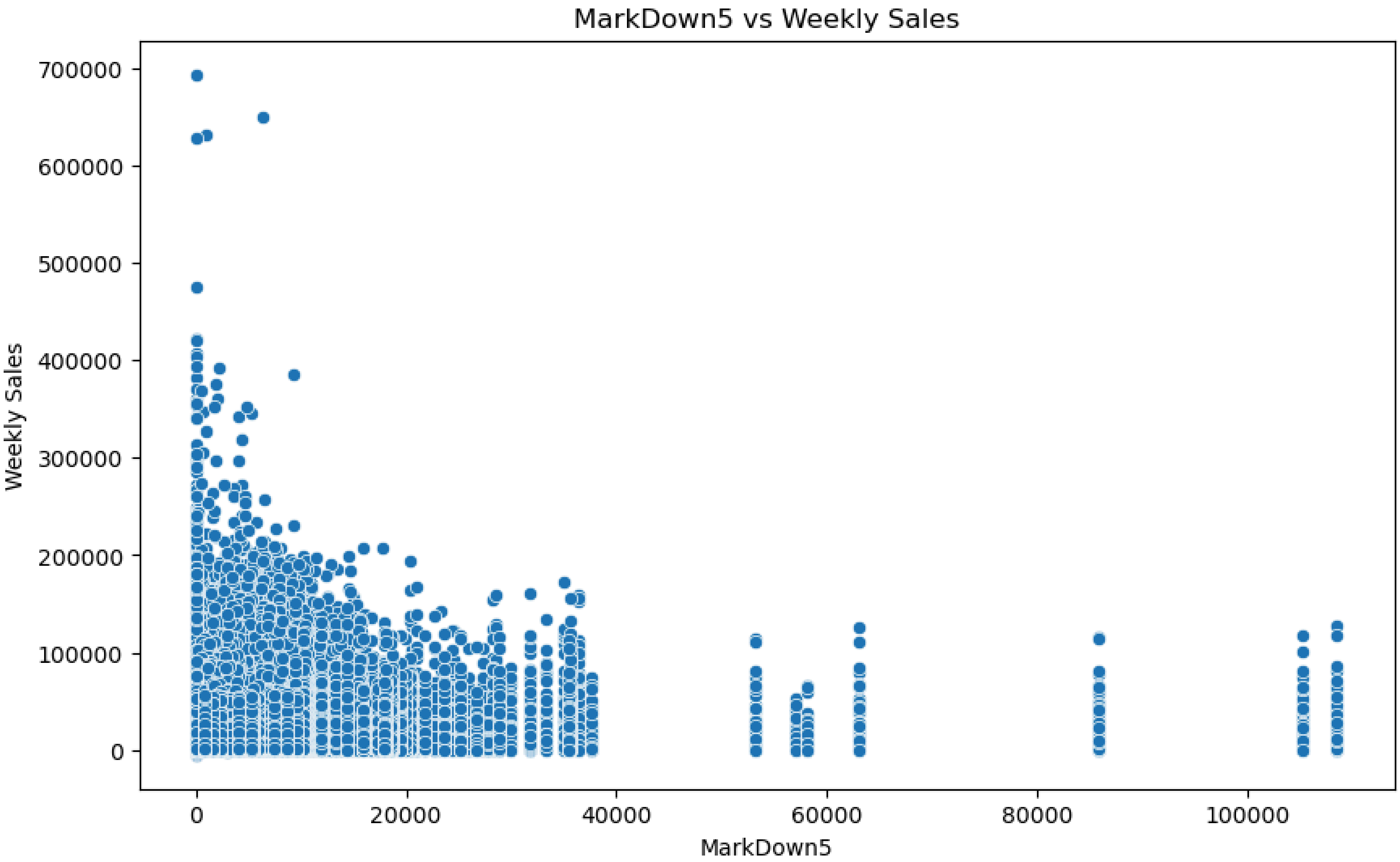
In [32]:
```
# Scatter plots for Markdown1, Markdown2, Markdown3, Markdown4, Markdown5 vs Weekly Sales
markdowns = ['MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5']

for markdown in markdowns:
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x=markdown, y='Weekly_Sales', data=train_set)
    plt.title(f'{markdown} vs Weekly Sales')
    plt.xlabel(markdown)
    plt.ylabel('Weekly Sales')
    plt.show()
```


MarkDown1 vs Weekly Sales
Weekly Sales
700000
600000
500000
400000
300000
200000
100000
0
0
20000
40000
60000
80000
MarkDown1


MarkDown2 vs Weekly Sales
Weekly Sales
700000
600000
500000
400000
300000
200000
100000
0
0
20000
40000
60000
80000
100000
MarkDown2


MarkDown3 vs Weekly Sales
700000
600000
500000
400000
300000
200000
100000
0
Weekly Sales
0
20000
40000
60000
80000
100000
120000
140000
MarkDown3


MarkDown4 vs Weekly Sales
Weekly Sales
700000
600000
500000
400000
300000
200000
100000
0
0
10000
20000
30000
40000
50000
60000
70000
MarkDown4

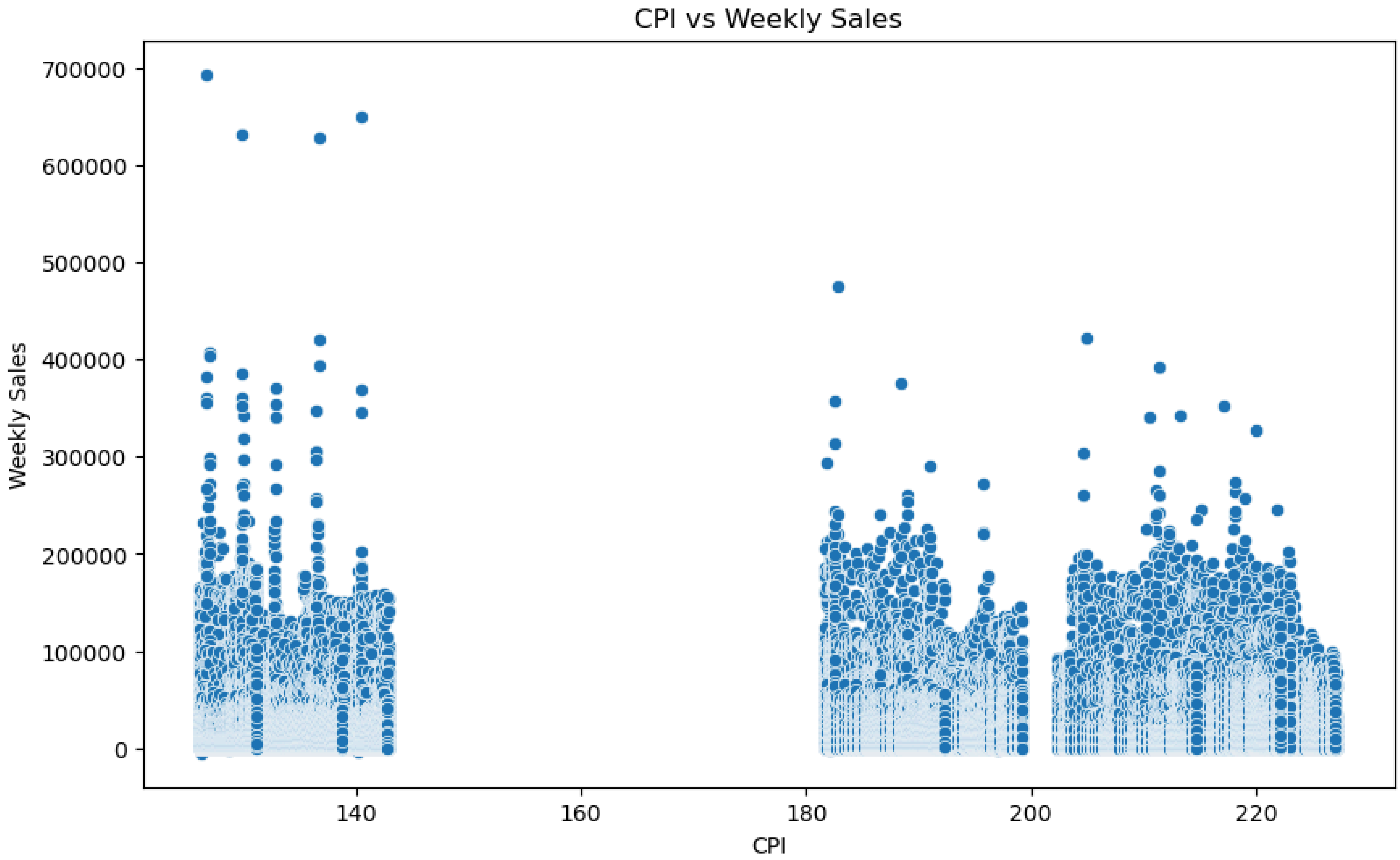


Unemployment and CPI vs Sales

In [33]:
```
# Scatter plot for Unemployment vs Weekly Sales
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Unemployment', y='Weekly_Sales', data=train_set)
plt.title('Unemployment vs Weekly Sales')
plt.xlabel('Unemployment')
```

```python
plt.ylabel('Weekly Sales')
plt.show()

# Scatter plot for CPI vs Weekly Sales
plt.figure(figsize=(10, 6))
sns.scatterplot(x='CPI', y='Weekly_Sales', data=train_set)
plt.title('CPI vs Weekly Sales')
plt.xlabel('CPI')
plt.ylabel('Weekly Sales')
plt.show()
```

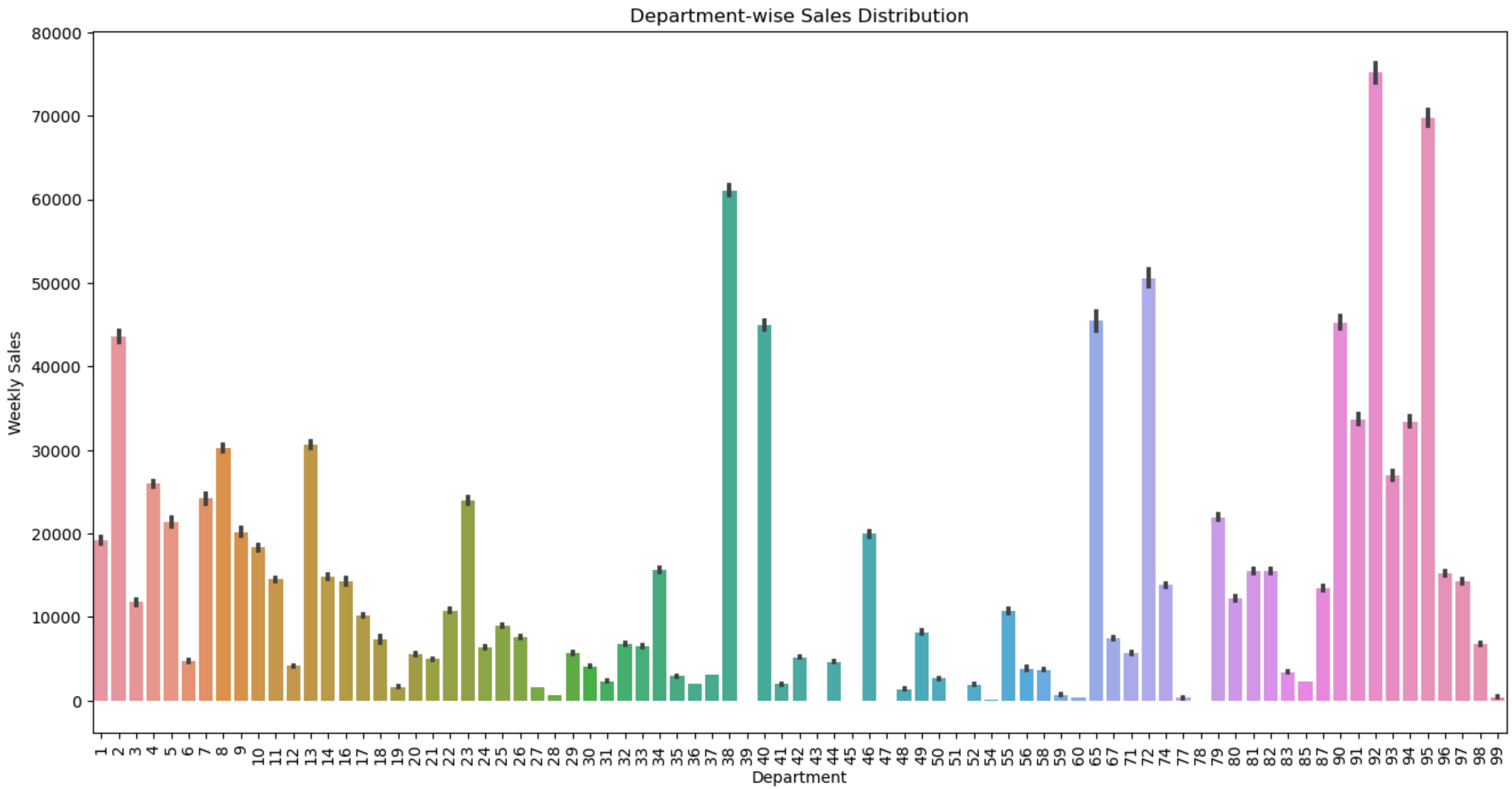

Unemployment vs Weekly Sales
Weekly Sales
Unemployment
0
100000
200000
300000
400000
500000
600000
700000
4
6
8
10
12
14



Weekly Sales by Department

In [34]:
```
# Boxplot for Department-wise sales distribution
plt.figure(figsize=(16, 8))
sns.barplot(x='Dept', y='Weekly_Sales', data=train_set)
plt.title('Department-wise Sales Distribution')
plt.xlabel('Department')
```

```
plt.ylabel('Weekly Sales')
plt.xticks(rotation=90)
plt.show()
```



Correlation Matrix

In [121…

```
drop_col = ['Super_Bowl','Labor_Day','Thanksgiving','Christmas','IsHoliday', 'Type']
df_train.drop(drop_col, axis=1, inplace=True) # dropping columns
```

In [124…

```
df_train.info()
```

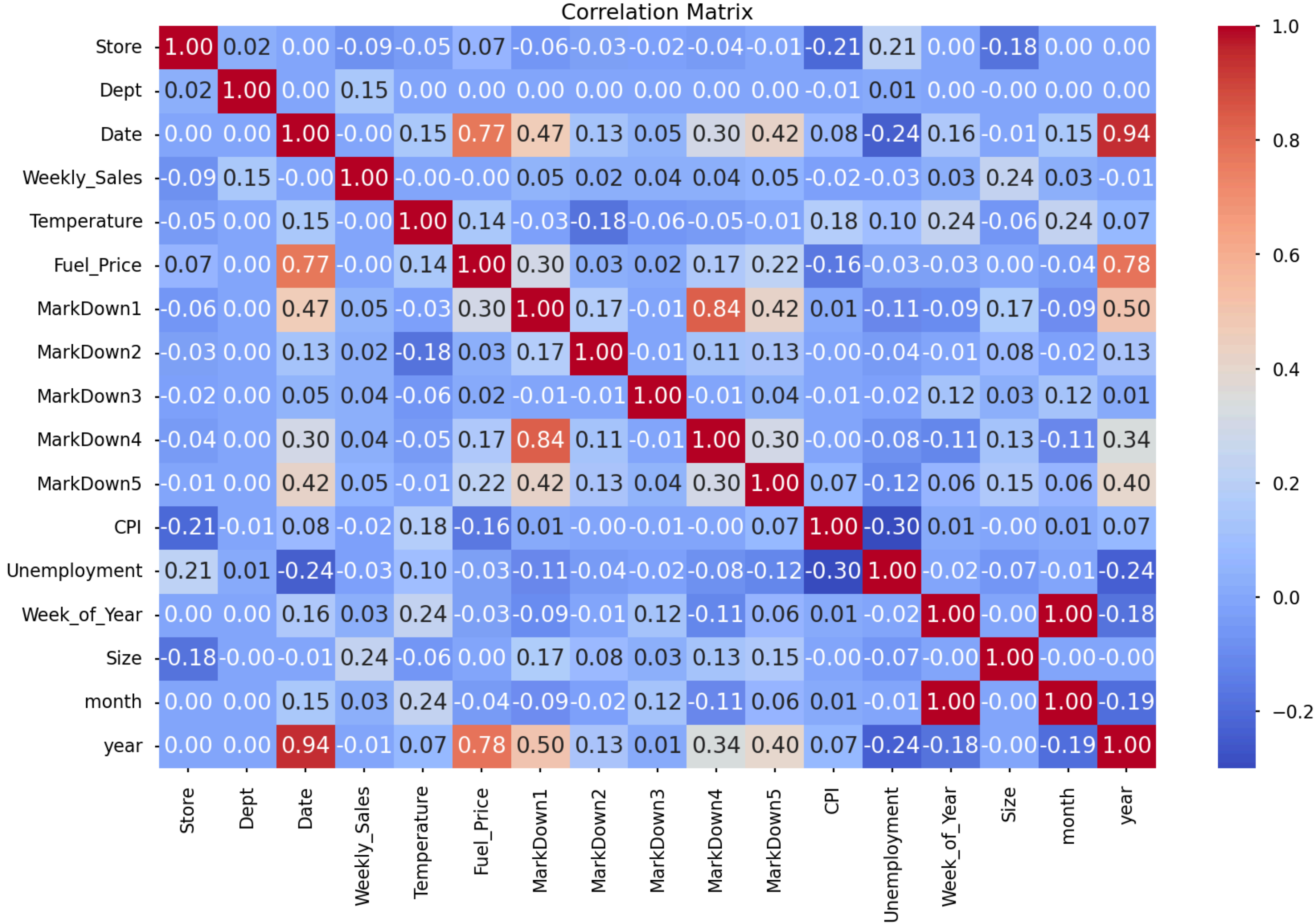
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 421570 entries, 0 to 421569
Data columns (total 17 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Store         421570 non-null  int64
 1   Dept          421570 non-null  int64
 2   Date          421570 non-null  datetime64[ns]
 3   Weekly_Sales  421570 non-null  float64
 4   Temperature   421570 non-null  float64
 5   Fuel_Price    421570 non-null  float64
 6   MarkDown1     421570 non-null  float64
 7   MarkDown2     421570 non-null  float64
 8   MarkDown3     421570 non-null  float64
 9   MarkDown4     421570 non-null  float64
 10  MarkDown5     421570 non-null  float64
 11  CPI           421570 non-null  float64
 12  Unemployment  421570 non-null  float64
 13  Week_of_Year  421570 non-null  UInt32
 14  Size          421570 non-null  int64
 15  month         421570 non-null  int32
 16  year          421570 non-null  int32
dtypes: UInt32(1), datetime64[ns](1), float64(10), int32(2), int64(3)
memory usage: 50.3 MB
```

In [ ]:

```
```

In [122…

```
# Correlation matrix
plt.figure(figsize=(20, 12))
corr_matrix = df_train.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```



This correlation matrix provides insights into the relationships between various features in your retail dataset. Let me explain some key observations:

Strong Positive Correlations:

Date and Year (0.94): This is expected as the year is derived from the date. Date and Fuel_Price (0.77): Suggests fuel prices have generally increased over time. MarkDown1 and MarkDown4 (0.84): Indicates these two types of markdowns often occur together. Week_of_Year and Month (1.00): Perfect correlation as these are directly related.

Moderate Positive Correlations:

Date and various MarkDowns: Suggests promotional activities have increased over time. Size and Weekly_Sales (0.24): Larger stores tend to have slightly higher sales. Temperature and Week_of_Year/Month (0.24): Reflects seasonal temperature changes.

Weak to Moderate Negative Correlations:

CPI and Unemployment (-0.30): As unemployment decreases, CPI tends to increase slightly. Store and CPI/Unemployment (-0.21): Might indicate regional economic differences.

Weak or No Correlations:

Most features have weak correlations with Weekly_Sales, with Size having the strongest (0.24). Dept has very low correlations with most features, suggesting department-specific factors may be important.

Interesting Observations:

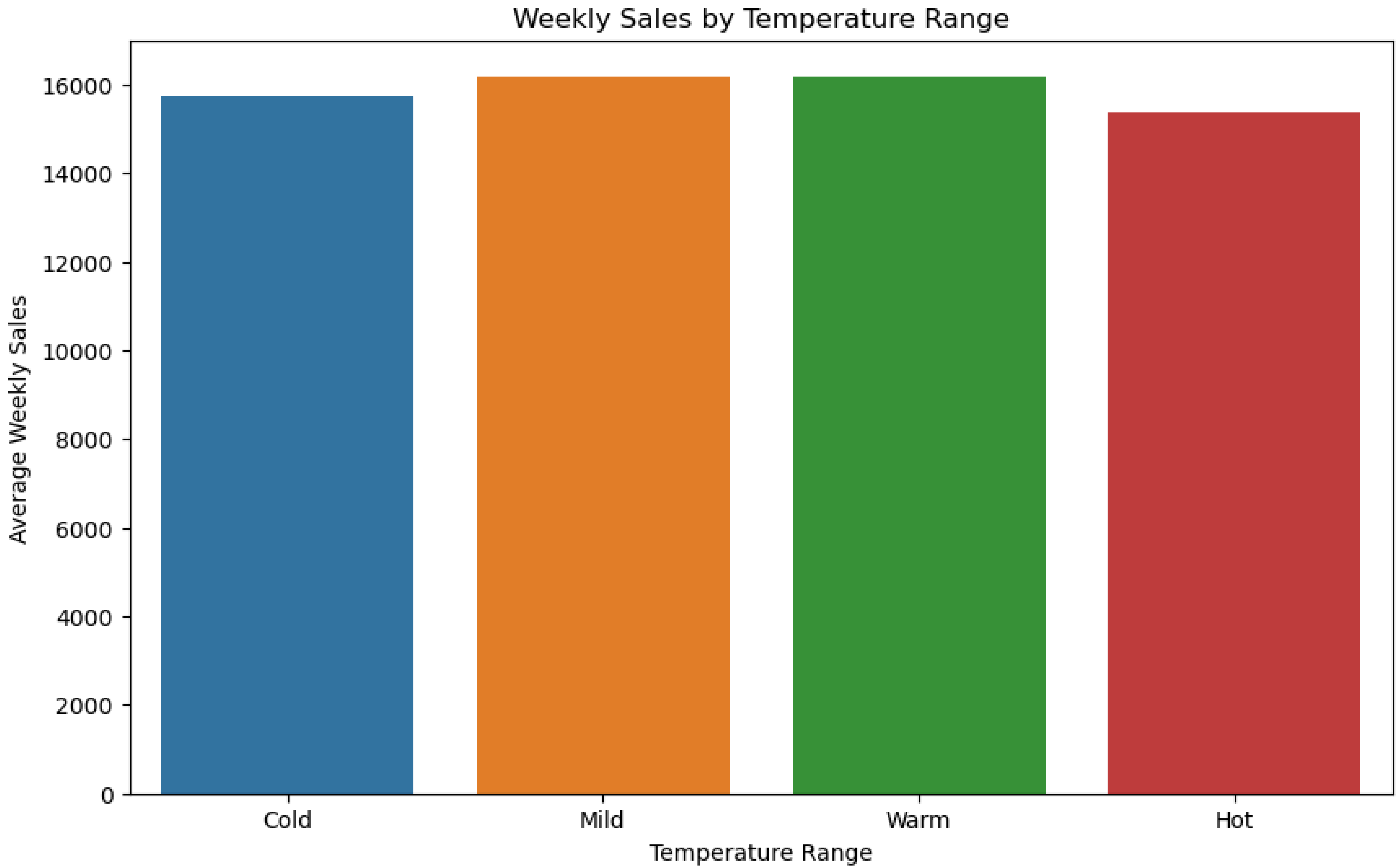
Temperature has a weak positive correlation with CPI (0.18) and Unemployment (0.10). MarkDowns are positively correlated with each other and with Date/Year, suggesting increased promotional activities over time.

Potential Multicollinearity:

High correlations between Date, Year, and Fuel_Price may cause issues in some models. Perfect correlation between Week_of_Year and Month might require using only one of these features in models.

In [39]:
```
# Create temperature bins for analysis
train_set['Temp_Bin'] = pd.cut(train_set['Temperature'], bins=[0, 40, 60, 80, 100], labels=['Cold', 'Mild', 'Warm', 'Hot'])

#Group by temperature bins and calculate mean sales
temp_sales = train_set.groupby('Temp_Bin')['Weekly_Sales'].mean().reset_index()
```

```
# Plot temperature bins vs weekly sales
plt.figure(figsize=(10, 6))
sns.barplot(x='Temp_Bin', y='Weekly_Sales', data=temp_sales)
plt.title('Weekly Sales by Temperature Range')
plt.xlabel('Temperature Range')
plt.ylabel('Average Weekly Sales')
plt.show()
```



Sales Trends Weekly

In [ ]:

```

```

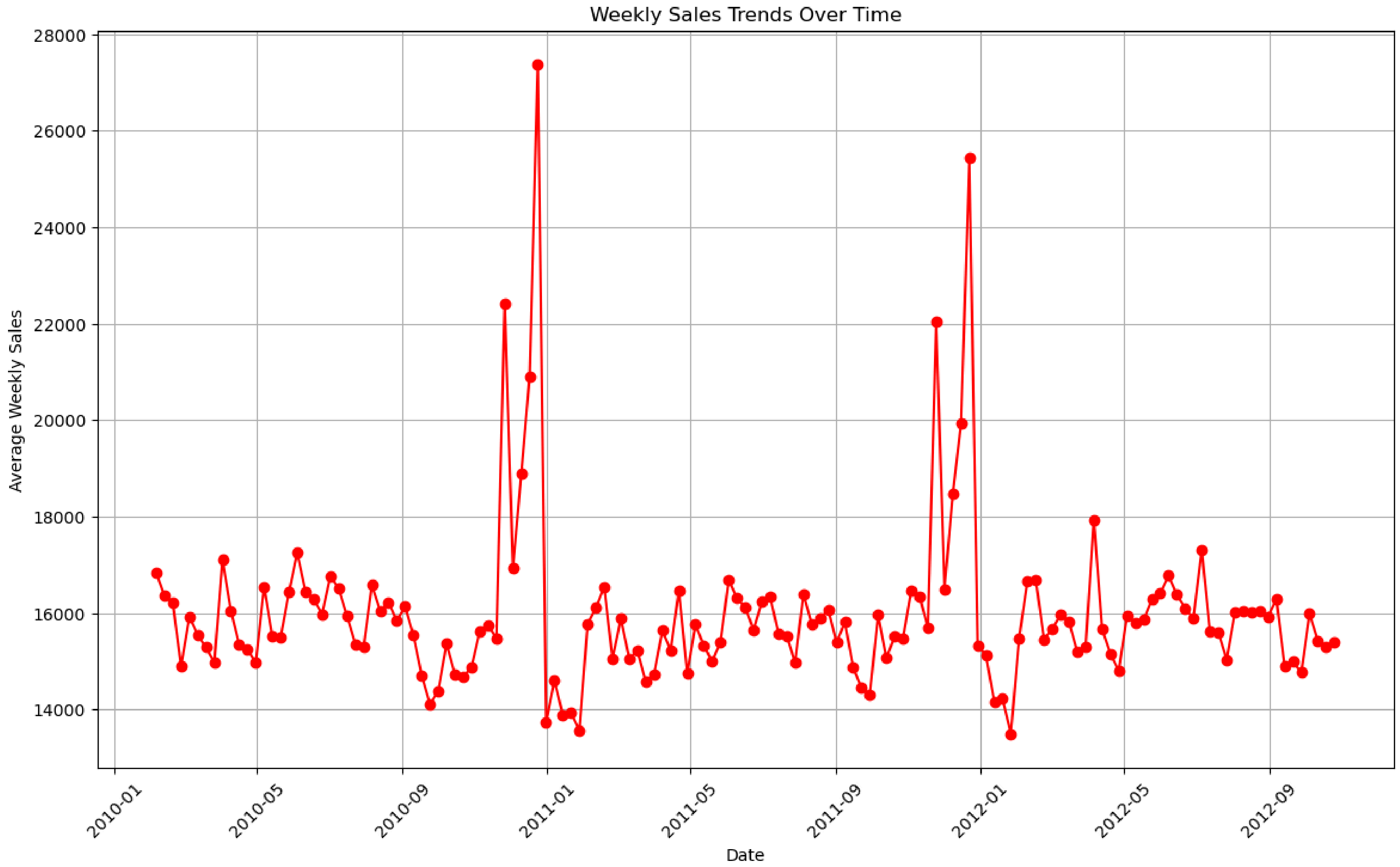
In [36]:

```
# Group by Date and calculate the average Weekly Sales for each Date
weekly_sales_by_date = train_set.groupby('Date')['Weekly_Sales'].mean().reset_index()

# Check the grouped data
print(weekly_sales_by_date.head(14))
```

```
         Date  Weekly_Sales
0  2010-02-05  16836.121997
1  2010-02-12  16352.056032
2  2010-02-19  16216.658979
3  2010-02-26  14899.549688
4  2010-03-05  15921.015727
5  2010-03-12  15546.850545
6  2010-03-19  15286.773578
7  2010-03-26  14975.894486
8  2010-04-02  17098.620298
9  2010-04-09  16050.589780
10 2010-04-16  15347.713003
11 2010-04-23  15252.114749
12 2010-04-30  14967.509147
13 2010-05-07  16542.716071
```

In [37]:

```
import matplotlib.pyplot as plt

# Plotting Weekly Sales vs Date
plt.figure(figsize=(14, 8))
plt.plot(weekly_sales_by_date['Date'], weekly_sales_by_date['Weekly_Sales'], marker='o', color='red')

# Adding titles and labels
plt.title('Weekly Sales Trends Over Time')
plt.xlabel('Date')
plt.ylabel('Average Weekly Sales')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.grid(True)
plt.show()
```


Weekly Sales Trends Over Time
Average Weekly Sales
28000
26000
24000
22000
20000
18000
16000
14000
Date
2010-01
2010-05
2010-09
2011-01
2011-05
2011-09
2012-01
2012-05
2012-09

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: