Project Objective

The objectives of the project are:

1. To visualize which store has highest weekly sales
2. To know how many holidays were there in the given time
3. To visualize the distribution of features
4. To visualize which features are affecting the weekly sales
5. To visualize which year has generated more weekly sales
6. To visualize yearly trend of sales
7. To visualize monthly trend of sales per year
8. To visualize weekly trend of sales per month per year
9. To visualize the feature affecting the weekly sales per year and comparing them
10. To find the best evaluation model
11. To predict the model using prophet with regressors
12. To verify the data obtained from prophet with the best evaluation model
13. To forecast the weekly sales for 12 weeks

Assumptions

1. The Random Forest regression and XG Boost regression model used here is a very plain and simple model. No extra parameters are added.
2. Five features were added as regressors to make prophet model better, as there were some relationships between the weekly sales and other five features. Those five features are temperature, unemployment, cpi, fuel price, and holiday flag.
3. Out of three best models for forecasting, one model was selected based on the accuracy. With the help of the regressors added to the prophet model, the model will make predictions. Those predictions will be evaluated by using the selected evaluation algorithm.
4. After evaluating the forecasting model, if the accuracy of the evaluation algorithm is good, then the obtained forecasting result will be best.

Importing Libraries...

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

Reading Data...

In [2]:
```
data = pd.read_csv("/kaggle/input/walmart-sales/Walmart (1).csv")
data.head()
```

Out[2]:

	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

Data Description

The available dataset is Walmart dataset, which have following features:

1. Store – Store Number
2. Date – Week of Sales
3. Weekly_Sales – Sales for the given store in that week
4. Holiday_Flag – If it is a holiday week
5. Temperature – Temperature on the day of sale
6. Fuel_Price – Cost of the fuel in the region
7. CPI – Consumer Price Index
8. Unemployment – Unemployment Rate

Checking for nulls

In [3]:
```
data.isnull().sum()
```

Out[3]:
```
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
dtype: int64
```

In [4]:
```
data.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
```

In [5]:
```
data['Date'] = pd.to_datetime(data['Date'])
```

In [6]:
```
data.shape
```

Out[6]:
```
(6435, 8)
```

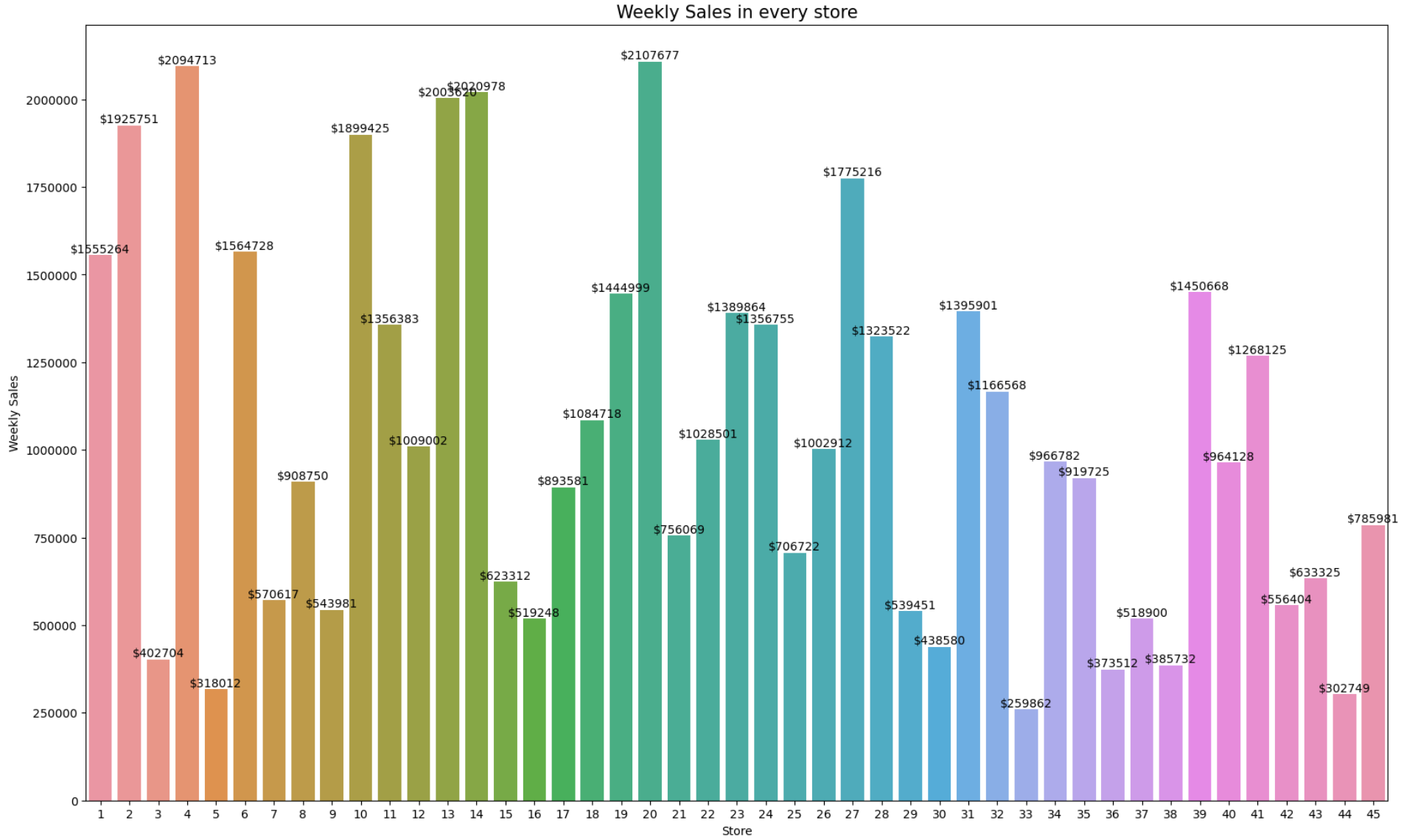
In [7]:
```
data['Store'].nunique()
```

Out[7]:
```
45
```

Weekly Sales in every store

In [8]:
```
plt.subplots(figsize=(20,12))
ax = sns.barplot(data,x = 'Store',y = 'Weekly_Sales',ci = None)
ax.bar_label(ax.containers[0],fmt='$%.0f')
```

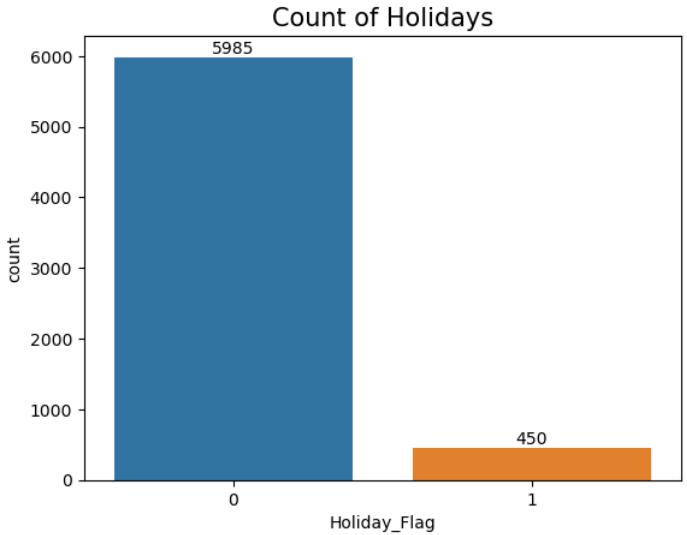
```
plt.ticklabel_format(useOffset=False,style='plain',axis='y')
plt.title("Weekly Sales in every store",fontsize = 15)
plt.ylabel("Weekly Sales")
plt.show()
```



Count of Holidays

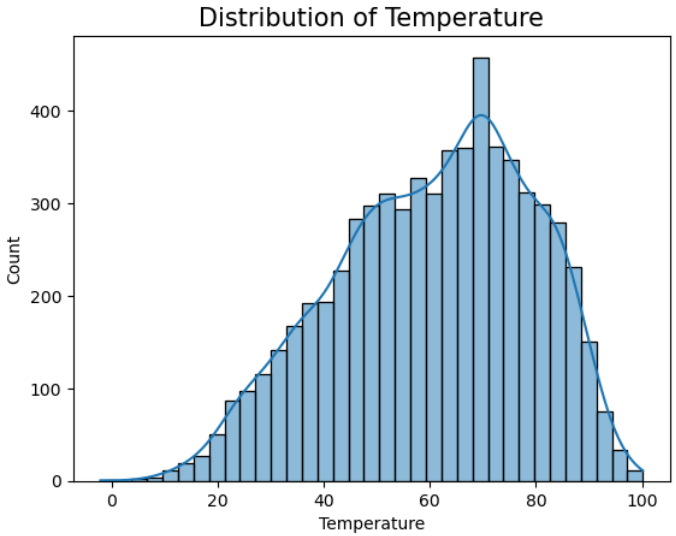
In [9]:
```
ax = sns.countplot(data, x = 'Holiday_Flag')
ax.bar_label(ax.containers[0])
```

```
plt.title("Count of Holidays",fontsize=15)
plt.show()
```



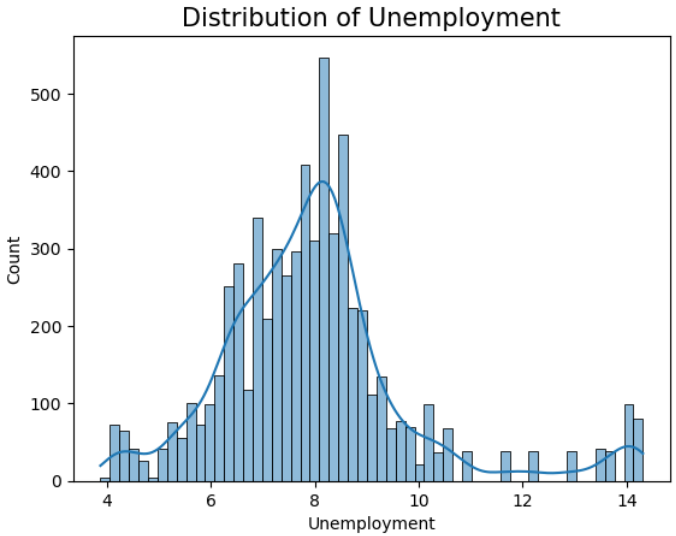
Distribution of Temperature

In [10]:
```
sns.histplot(data, x='Temperature',kde=True)
plt.title("Distribution of Temperature",fontsize=15)
plt.show()
```



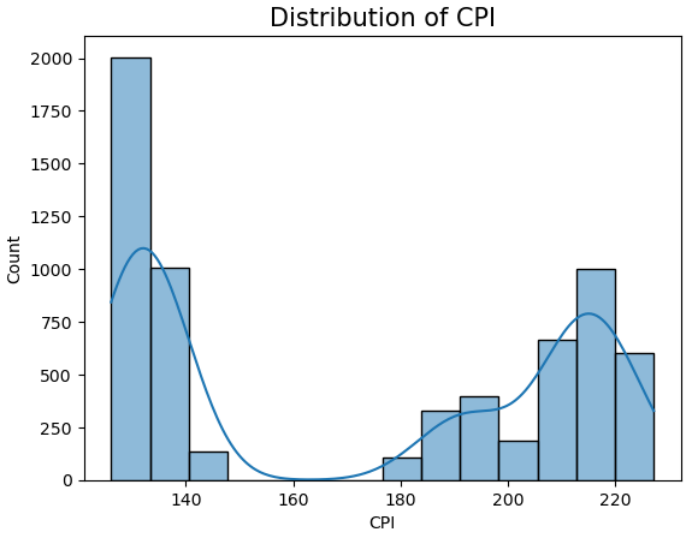
Distribution of Unemployment

In [11]:
```
sns.histplot(data, x='Unemployment',kde=True)
plt.title("Distribution of Unemployment",fontsize=15)
plt.show()
```



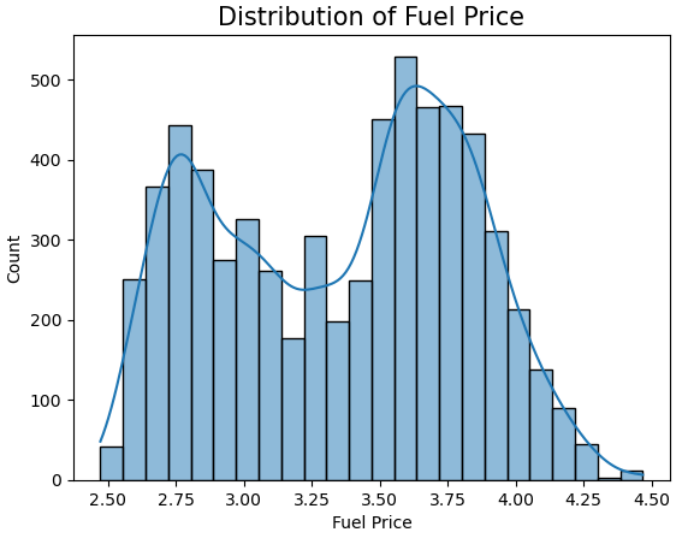
Distribution of CPI

In [12]:
```
sns.histplot(data, x='CPI',kde=True)
plt.title("Distribution of CPI",fontsize=15)
plt.show()
```



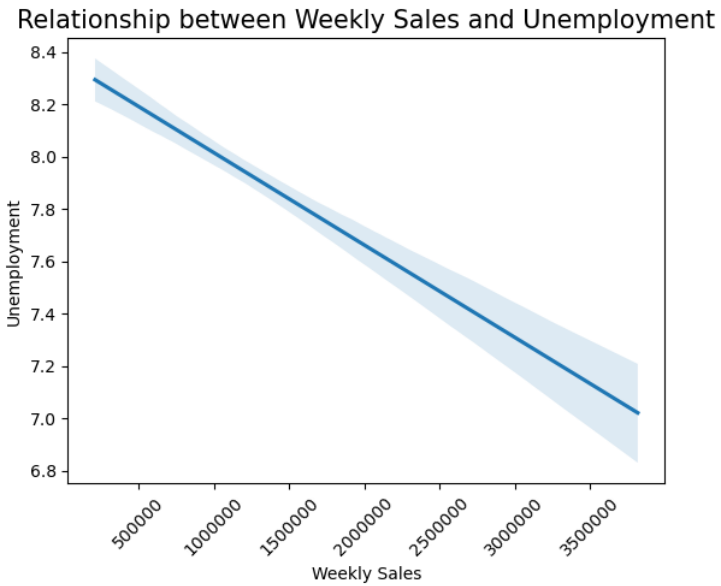
Distribution of Fuel Price

In [13]:
```
sns.histplot(data, x='Fuel_Price',kde=True)
plt.title("Distribution of Fuel Price",fontsize=15)
plt.xlabel("Fuel Price")
plt.show()
```



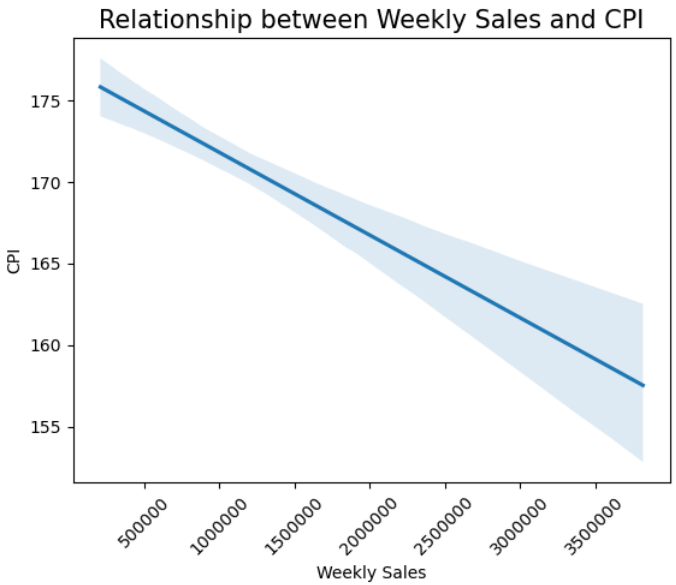
Relationship between Weekly Sales and Unemployment

In [14]:
```
sns.regplot(data,x='Weekly_Sales',y = 'Unemployment',scatter=False)
plt.ticklabel_format(useOffset=False,style='plain',axis='x')
plt.xticks(rotation = 45)
plt.title("Relationship between Weekly Sales and Unemployment",fontsize = 15)
plt.xlabel("Weekly Sales")
plt.show()
```



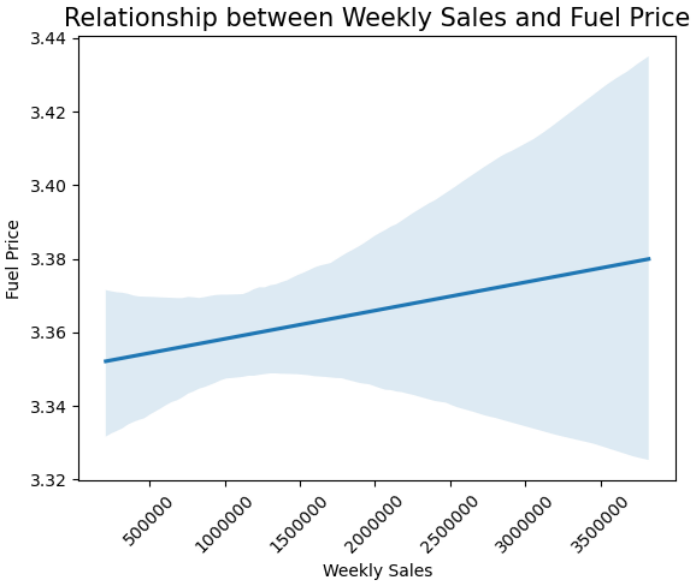
Relationship between Weekly Sales and CPI

In [15]:
```
sns.regplot(data,x='Weekly_Sales',y = 'CPI',scatter=False)
plt.ticklabel_format(useOffset=False,style='plain',axis='x')
plt.xticks(rotation = 45)
plt.title("Relationship between Weekly Sales and CPI",fontsize = 15)
plt.xlabel("Weekly Sales")
plt.show()
```



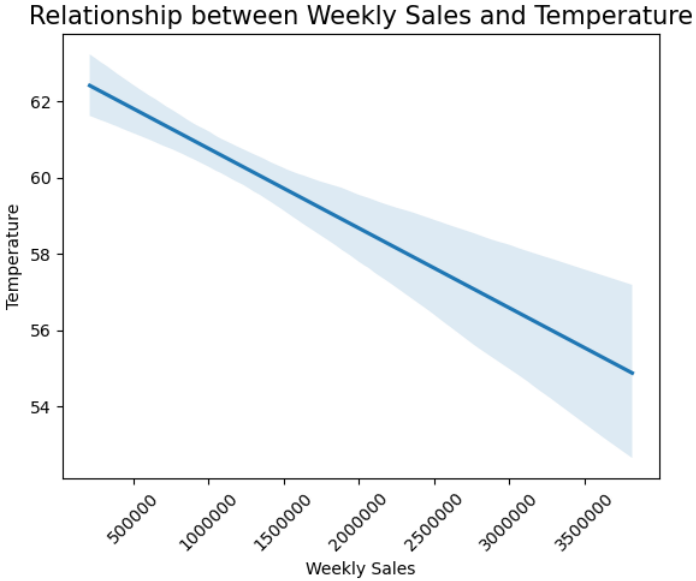
Relationship between Weekly Sales and Fuel Price

In [16]:
```
sns.regplot(data,x='Weekly_Sales',y = 'Fuel_Price',scatter=False)
plt.ticklabel_format(useOffset=False,style='plain',axis='x')
plt.xticks(rotation = 45)
plt.title("Relationship between Weekly Sales and Fuel Price",fontsize = 15)
plt.xlabel("Weekly Sales")
plt.ylabel("Fuel Price")
plt.show()
```



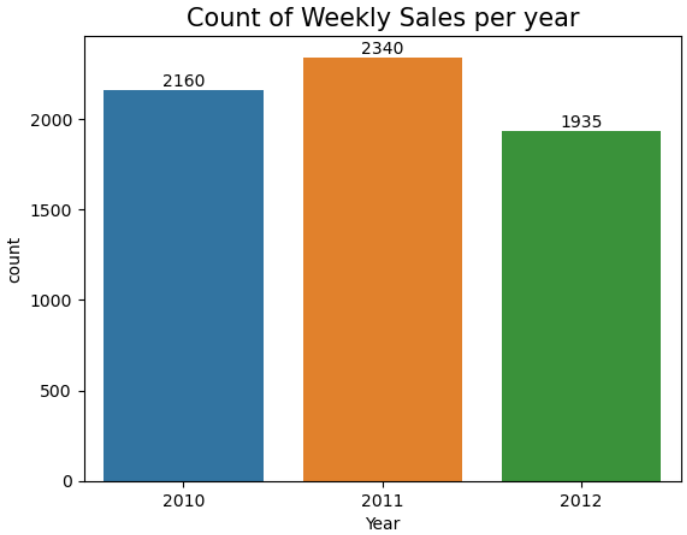
Relationship between Weekly Sales and Temperature

In [17]:
```
sns.regplot(data,x='Weekly_Sales',y = 'Temperature',scatter=False)
plt.ticklabel_format(useOffset=False,style='plain',axis='x')
plt.xticks(rotation = 45)
plt.title("Relationship between Weekly Sales and Temperature",fontsize = 15)
plt.xlabel("Weekly Sales")
plt.show()
```



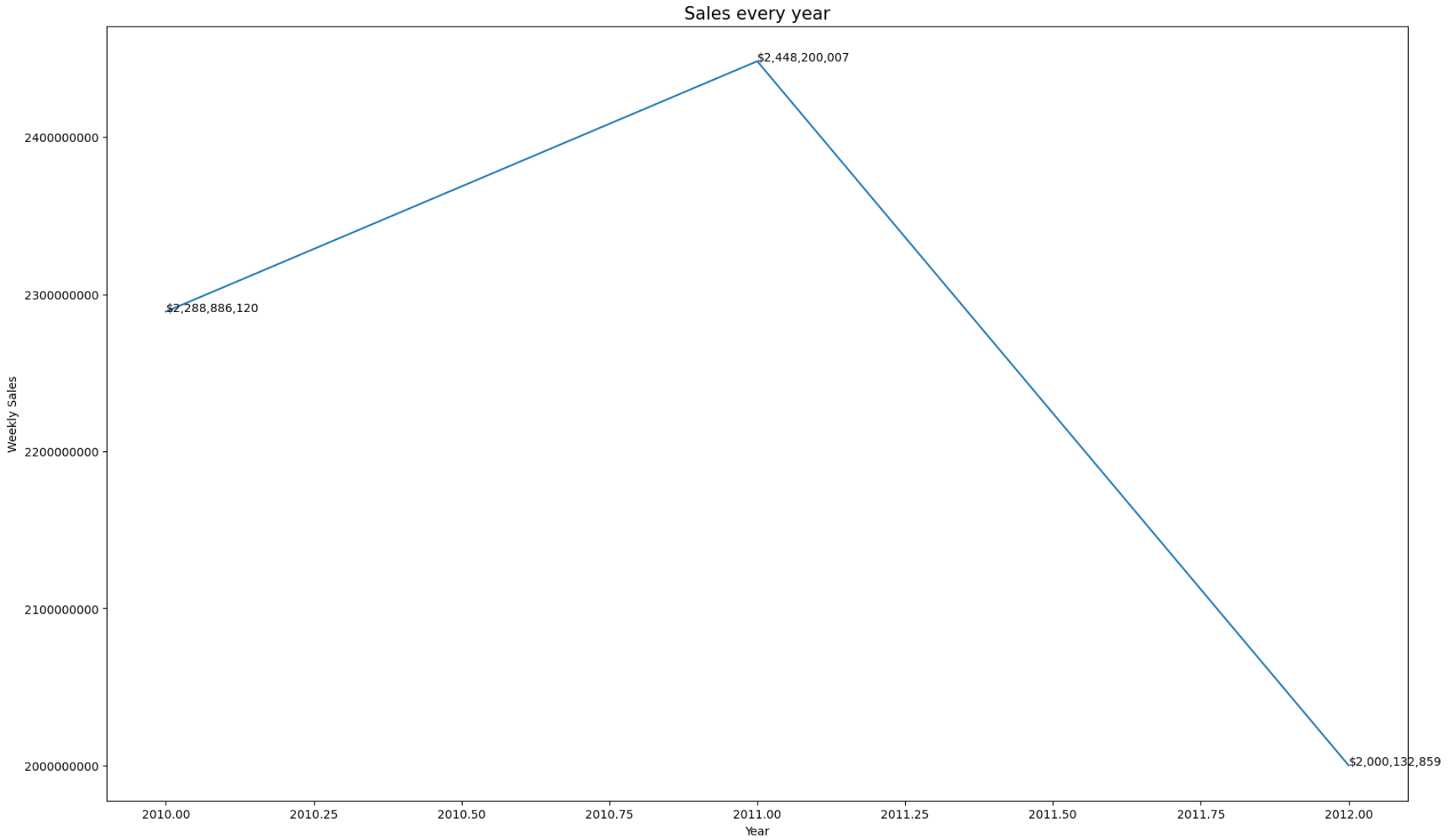
Count of Weekly Sales per year

In [18]:
```
data['Year'] = data['Date'].dt.year
ax = sns.countplot(data,x='Year')
ax.bar_label(ax.containers[0])
plt.title("Count of Weekly Sales per year", fontsize = 15)
plt.show()
```



Sales every year

In [19]:
```
v0 = data.groupby('Year')['Weekly_Sales'].sum().reset_index()
plt.subplots(figsize = (20,12))
ax = sns.lineplot(v0,x = 'Year',y = 'Weekly_Sales')
for x,y in zip(v0['Year'],v0['Weekly_Sales']):
    plt.text(x = x, y = y, s = '${:,.0f}'.format(y))
plt.ticklabel_format(useOffset=False,style='plain',axis='y')
plt.title("Sales every year",fontsize = 15)
plt.ylabel("Weekly Sales")
plt.show()
```



In [20]:
```
data['Month'] = data['Date'].dt.month
data['MonthName'] = data['Date'].dt.month_name()
data['Week'] = data['Date'].dt.week
```

Year 2010

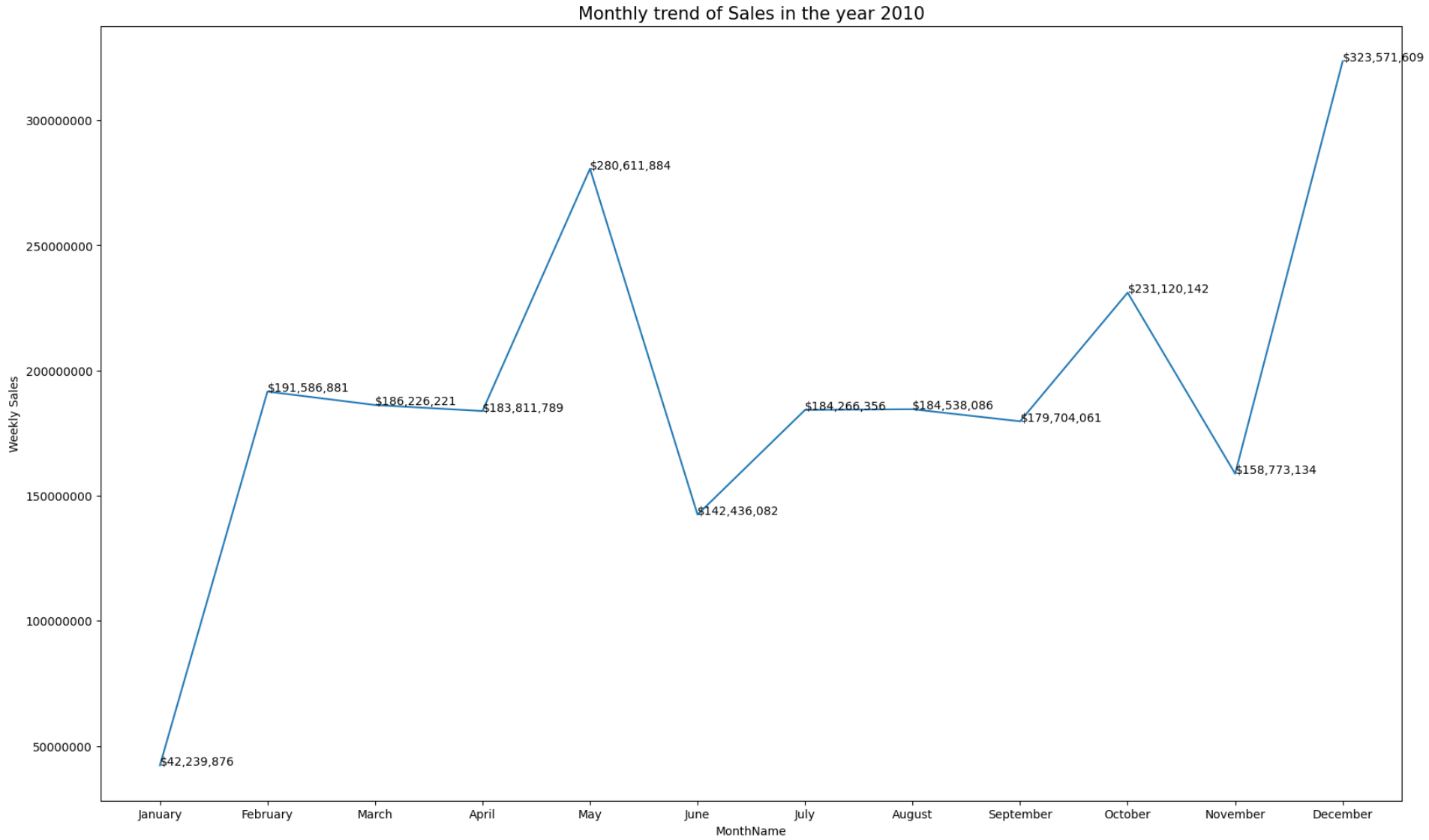
In [21]:
```
data_2010 = data[data['Year']==2010].sort_values(by='Month')
data_2010.head()
```

Out[21]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Year	Month	MonthName	Week
3323	24	2010-01-10	1215273.20	0	66.88	2.840	132.756800	8.275	2010	1	January	1
1607	12	2010-01-10	850936.26	0	85.20	3.001	126.234600	14.313	2010	1	January	1
34	1	2010-01-10	1453329.50	0	71.89	2.603	211.671989	7.838	2010	1	January	1
5611	40	2010-01-10	891152.33	0	62.01	2.717	132.756800	5.287	2010	1	January	1
3895	28	2010-01-10	1203080.41	0	85.20	3.001	126.234600	14.313	2010	1	January	1

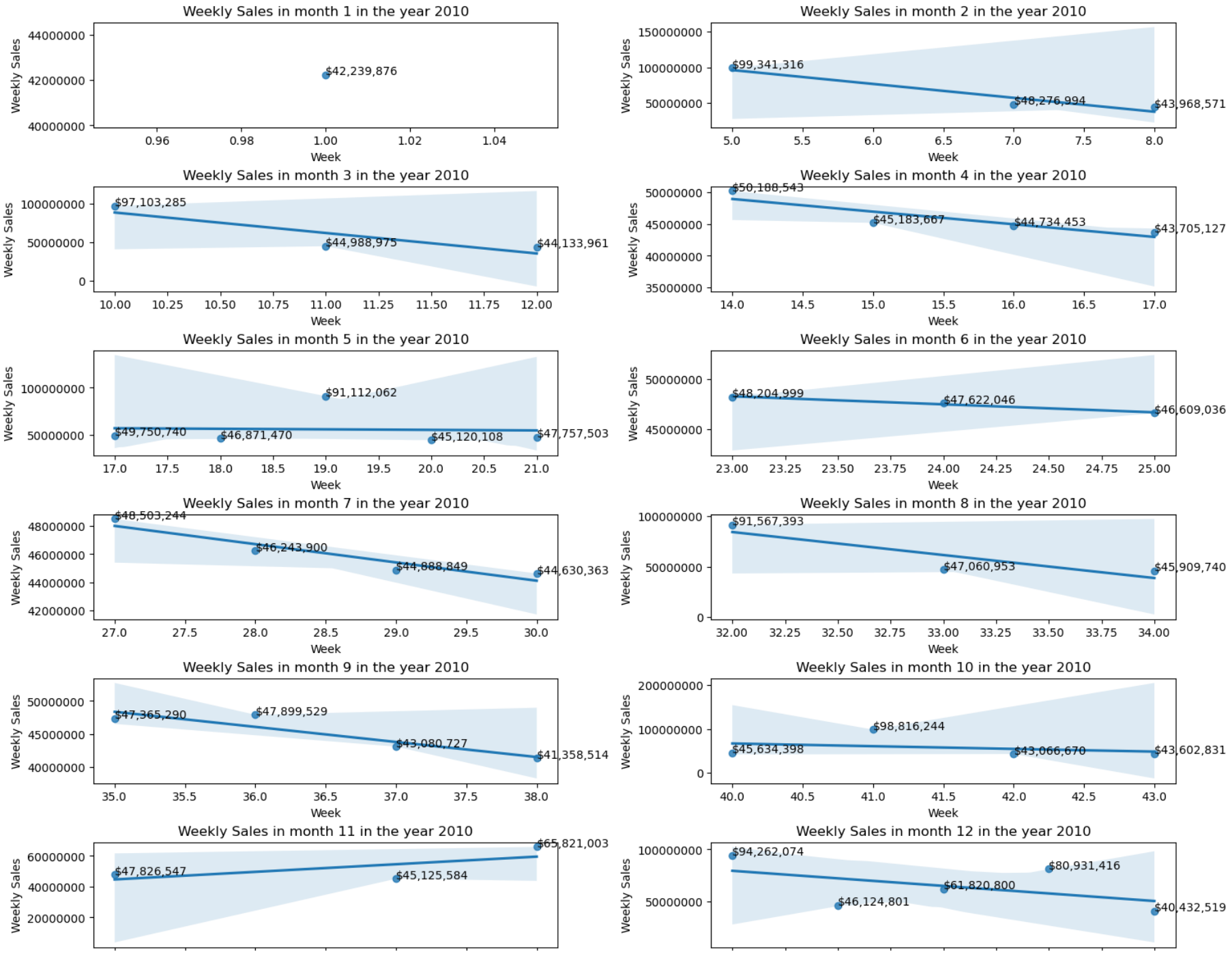
In [22]:

```
v1 = data_2010.groupby(['MonthName','Month'])['Weekly_Sales'].sum().reset_index().sort_values(by='Month')
plt.subplots(figsize = (20,12))
ax = sns.lineplot(v1,x = 'MonthName',y = 'Weekly_Sales')
for x,y in zip(v1['MonthName'],v1['Weekly_Sales']):
    plt.text(x = x, y = y, s = '${:,.0f}'.format(y))
plt.ticklabel_format(useOffset=False,style='plain',axis='y')
plt.title("Monthly trend of Sales in the year 2010",fontsize = 15)
plt.ylabel("Weekly Sales")
plt.show()
```



In [23]:
```
plt.subplots(6,2,figsize=(15,12),constrained_layout = 'True')
for i in range(1,13):
    vm = data_2010[data_2010['Month']==i]
    vw = vm.groupby('Week')['Weekly_Sales'].sum().reset_index()

    plt.subplot(6,2,i)
    ax = sns.regplot(vw,x='Week',y = 'Weekly_Sales')
    for x,y in zip(vw['Week'],vw['Weekly_Sales']):
        plt.text(x = x, y = y, s = '${:,.0f}'.format(y))
    plt.ticklabel_format(useOffset=False,style='plain',axis='y')
    plt.title(f'Weekly Sales in month {i} in the year 2010')
```

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


Weekly Sales in month 1 in the year 2010
Weekly Sales
Week
$42,239,876
Weekly Sales in month 2 in the year 2010
$99,341,316
$48,276,994
$43,968,571
Weekly Sales in month 3 in the year 2010
$97,103,285
$44,988,975
$44,133,961
Weekly Sales in month 4 in the year 2010
$50,188,543
$45,183,667
$44,734,453
$43,705,127
Weekly Sales in month 5 in the year 2010
$91,112,062
$49,750,740
$46,871,470
$45,120,108
$47,757,503
Weekly Sales in month 6 in the year 2010
$48,204,999
$47,622,046
$46,609,036
Weekly Sales in month 7 in the year 2010
$48,503,244
$46,243,900
$44,888,849
$44,630,363
Weekly Sales in month 8 in the year 2010
$91,567,393
$47,060,953
$45,909,740
Weekly Sales in month 9 in the year 2010
$47,365,290
$47,899,529
$43,080,727
$41,358,514
Weekly Sales in month 10 in the year 2010
$98,816,244
$45,634,398
$43,066,670
$43,602,831
Weekly Sales in month 11 in the year 2010
$65,821,003
$47,826,547
$45,125,584
Weekly Sales in month 12 in the year 2010
$94,262,074
$80,931,416
$61,820,800
$46,124,801
$40,432,519

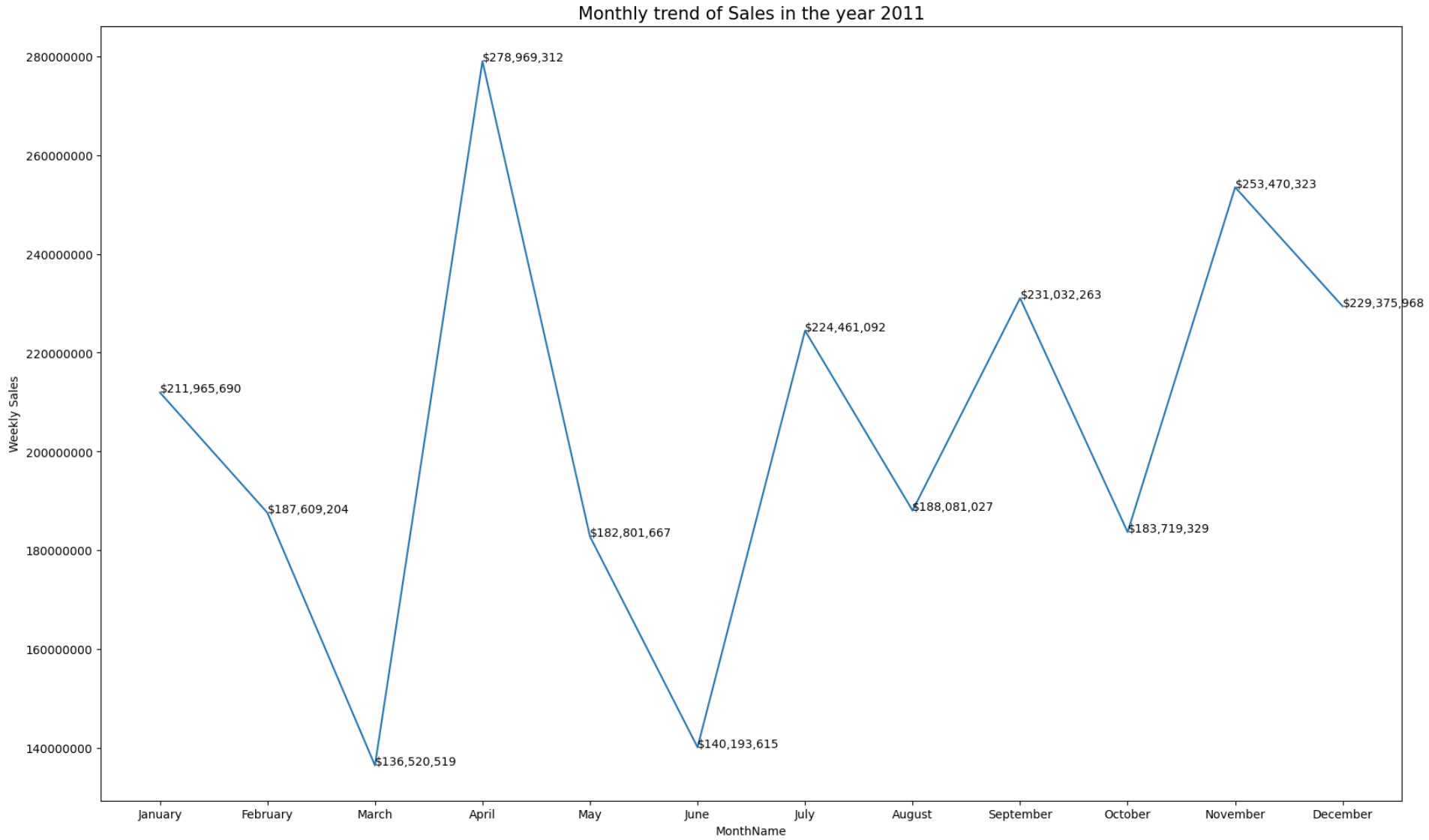
Year 2011

In [24]:
```
data_2011 = data[data['Year']==2011].sort_values(by='Month')
data_2011.head()
```

Out[24]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Year	Month	MonthName	Week
3219	23	2011-01-07	1492507.44	0	65.65	3.815	135.446800	4.584	2011	1	January	1
2062	15	2011-01-04	542556.05	0	30.34	3.811	134.068258	7.658	2011	1	January	1
4636	33	2011-01-04	232769.09	0	71.41	3.772	128.719935	8.687	2011	1	January	1
5936	42	2011-01-07	506343.83	0	95.36	3.842	129.089400	8.257	2011	1	January	1
2053	15	2011-01-28	481119.60	0	19.61	3.402	133.105968	7.771	2011	1	January	4

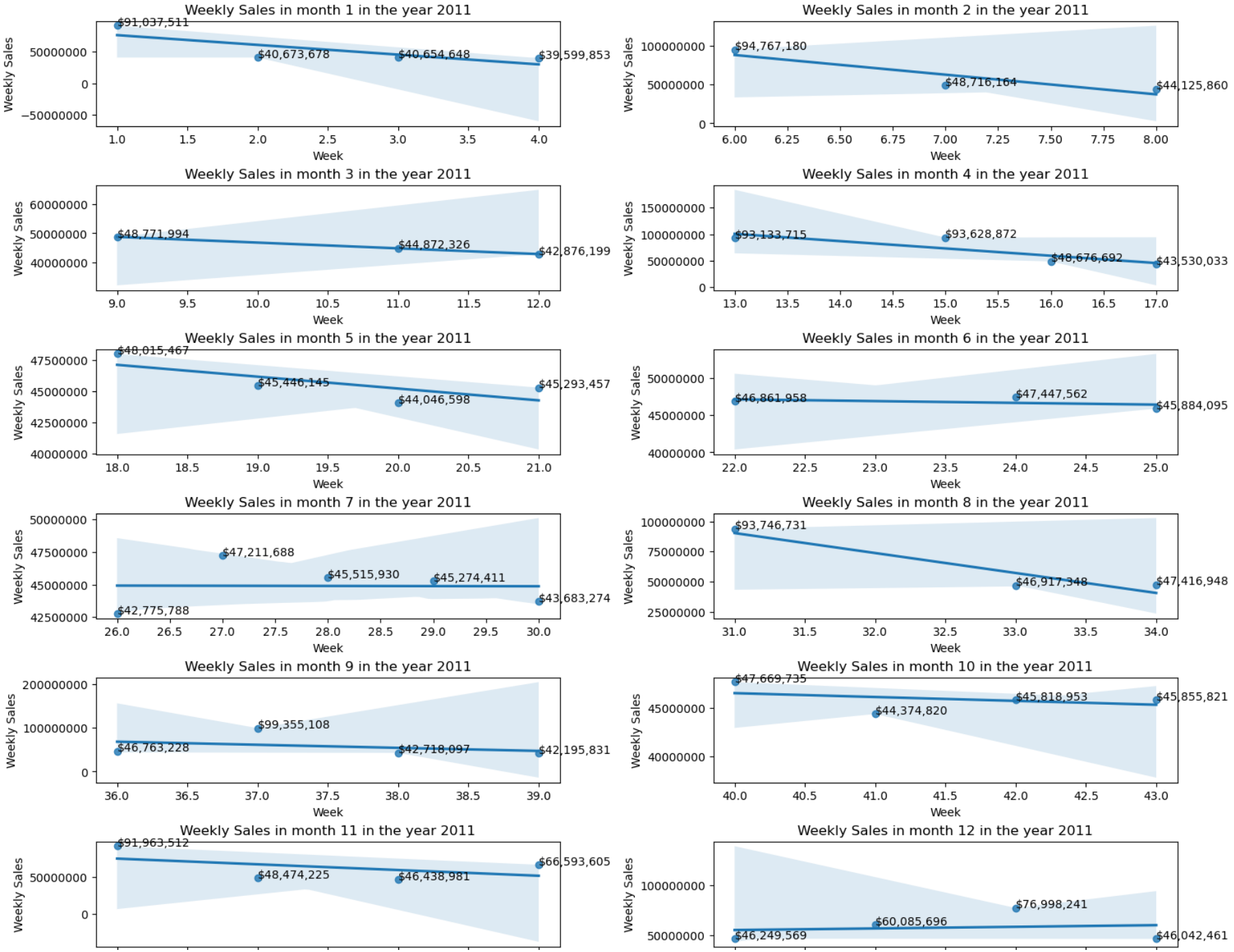
In [25]:
```
v2 = data_2011.groupby(['MonthName','Month'])['Weekly_Sales'].sum().reset_index().sort_values(by='Month')
plt.subplots(figsize = (20,12))
ax = sns.lineplot(v2,x = 'MonthName',y = 'Weekly_Sales')
for x,y in zip(v2['MonthName'],v2['Weekly_Sales']):
    plt.text(x = x, y = y, s = '${:,.0f}'.format(y))
plt.ticklabel_format(useOffset=False,style='plain',axis='y')
plt.title("Monthly trend of Sales in the year 2011",fontsize=15)
plt.ylabel("Weekly Sales")
plt.show()
```



In [26]:

```
plt.subplots(6,2,figsize=(15,12),constrained_layout = 'True')
for i in range(1,13):
    vm = data_2011[data_2011['Month']==i]
    vw = vm.groupby('Week')['Weekly_Sales'].sum().reset_index()

    plt.subplot(6,2,i)
    ax = sns.regplot(vw,x='Week',y = 'Weekly_Sales')
    for x,y in zip(vw['Week'],vw['Weekly_Sales']):
        plt.text(x = x, y = y, s = '${:,.0f}'.format(y))
    plt.ticklabel_format(useOffset=False,style='plain',axis='y')
    plt.title(f'Weekly Sales in month {i} in the year 2011')
```

```
    plt.ylabel("Weekly Sales")
plt.show()
```


Weekly Sales in month 1 in the year 2011
$91,037,511
$40,673,678
$40,654,648
$39,599,853
Weekly Sales in month 2 in the year 2011
$94,767,180
$48,716,164
$44,125,860
Weekly Sales in month 3 in the year 2011
$48,771,994
$44,872,326
$42,876,199
Weekly Sales in month 4 in the year 2011
$93,133,715
$93,628,872
$48,676,692
$43,530,033
Weekly Sales in month 5 in the year 2011
$48,015,467
$45,446,145
$44,046,598
$45,293,457
Weekly Sales in month 6 in the year 2011
$46,861,958
$47,447,562
$45,884,095
Weekly Sales in month 7 in the year 2011
$47,211,688
$45,515,930
$45,274,411
$42,775,788
$43,683,274
Weekly Sales in month 8 in the year 2011
$93,746,731
$46,917,348
$47,416,948
Weekly Sales in month 9 in the year 2011
$99,355,108
$46,763,228
$42,718,097
$42,195,831
Weekly Sales in month 10 in the year 2011
$47,669,735
$44,374,820
$45,818,953
$45,855,821
Weekly Sales in month 11 in the year 2011
$91,963,512
$48,474,225
$46,438,981
$66,593,605
Weekly Sales in month 12 in the year 2011
$76,998,241
$60,085,696
$46,249,569
$46,042,461
Weekly Sales
Week

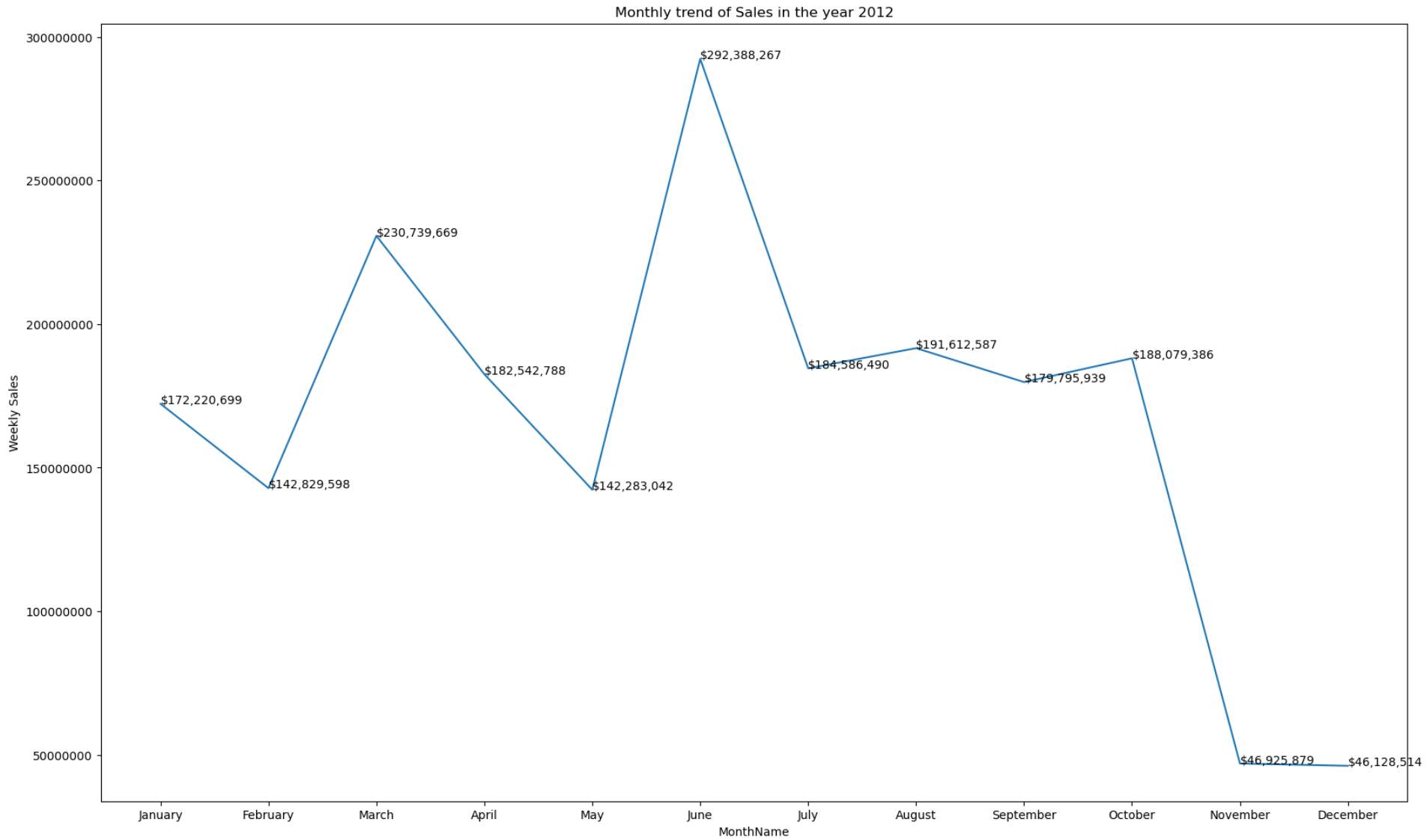
Year 2012

In [27]:
```
data_2012 = data[data['Year']==2012].sort_values(by='Month')
data_2012.head()
```

Out[27]:

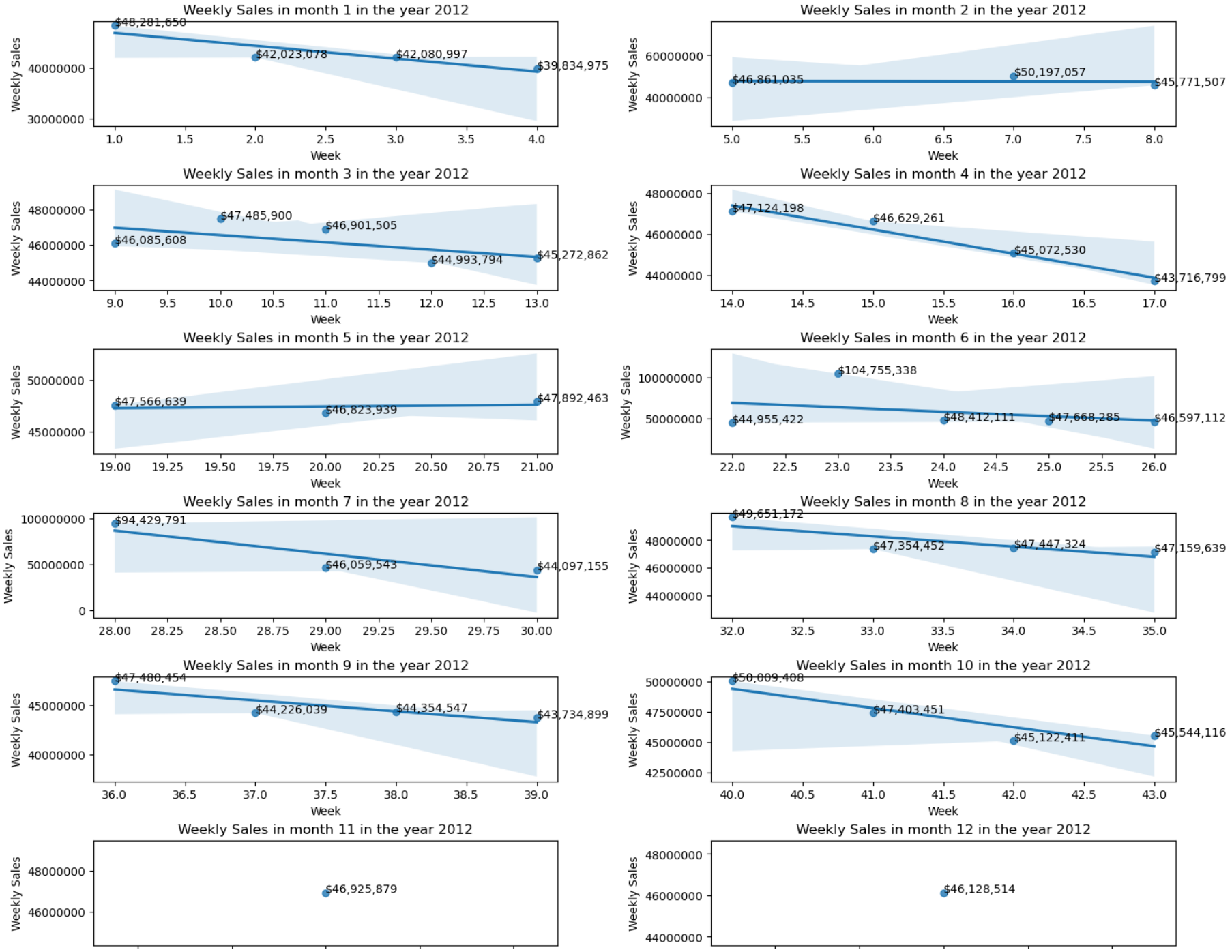
	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Year	Month	MonthName	Week
3267	23	2012-01-06	1476144.34	0	66.22	3.798	138.113807	4.125	2012	1	January	1
2104	15	2012-01-20	492721.85	0	21.39	3.705	136.856419	7.943	2012	1	January	3
2103	15	2012-01-13	454183.42	0	36.26	3.666	136.753000	7.943	2012	1	January	2
2695	19	2012-01-06	1450733.29	0	68.18	3.915	138.113807	8.150	2012	1	January	1
4554	32	2012-01-06	1157557.79	0	62.84	3.764	197.621895	8.090	2012	1	January	1

In [28]:
```
v3 = data_2012.groupby(['MonthName','Month'])['Weekly_Sales'].sum().reset_index().sort_values(by='Month')
plt.subplots(figsize = (20,12))
ax = sns.lineplot(v3,x = 'MonthName',y = 'Weekly_Sales')
for x,y in zip(v3['MonthName'],v3['Weekly_Sales']):
    plt.text(x = x, y = y, s = '${:,.0f}'.format(y))
plt.ticklabel_format(useOffset=False,style='plain',axis='y')
plt.title("Monthly trend of Sales in the year 2012")
plt.ylabel("Weekly Sales")
plt.show()
```



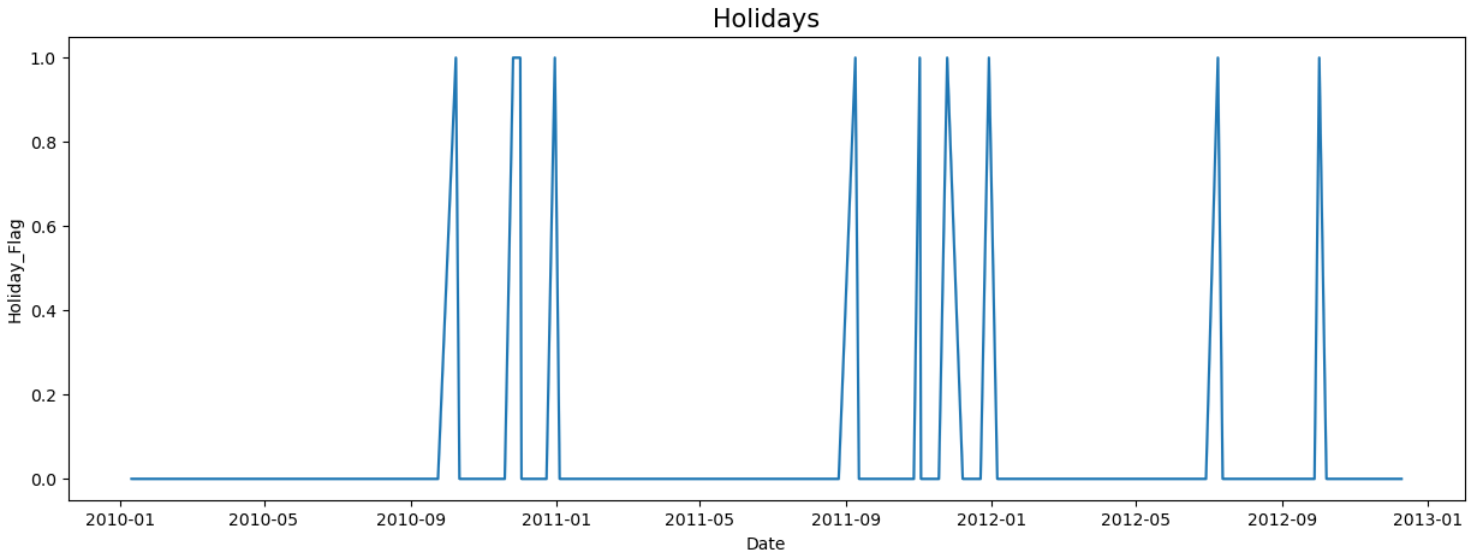
In [29]:
```
plt.subplots(6,2,figsize=(15,12),constrained_layout = 'True')
for i in range(1,13):
    vm = data_2012[data_2012['Month']==i]
    vw = vm.groupby('Week')['Weekly_Sales'].sum().reset_index()

    plt.subplot(6,2,i)
    ax = sns.regplot(vw,x='Week',y = 'Weekly_Sales')
    for x,y in zip(vw['Week'],vw['Weekly_Sales']):
        plt.text(x = x, y = y, s = '${:,.0f}'.format(y))
    plt.ticklabel_format(useOffset=False,style='plain',axis='y')
    plt.title(f'Weekly Sales in month {i} in the year 2012')
```

```
    plt.ylabel("Weekly Sales")
plt.show()
```

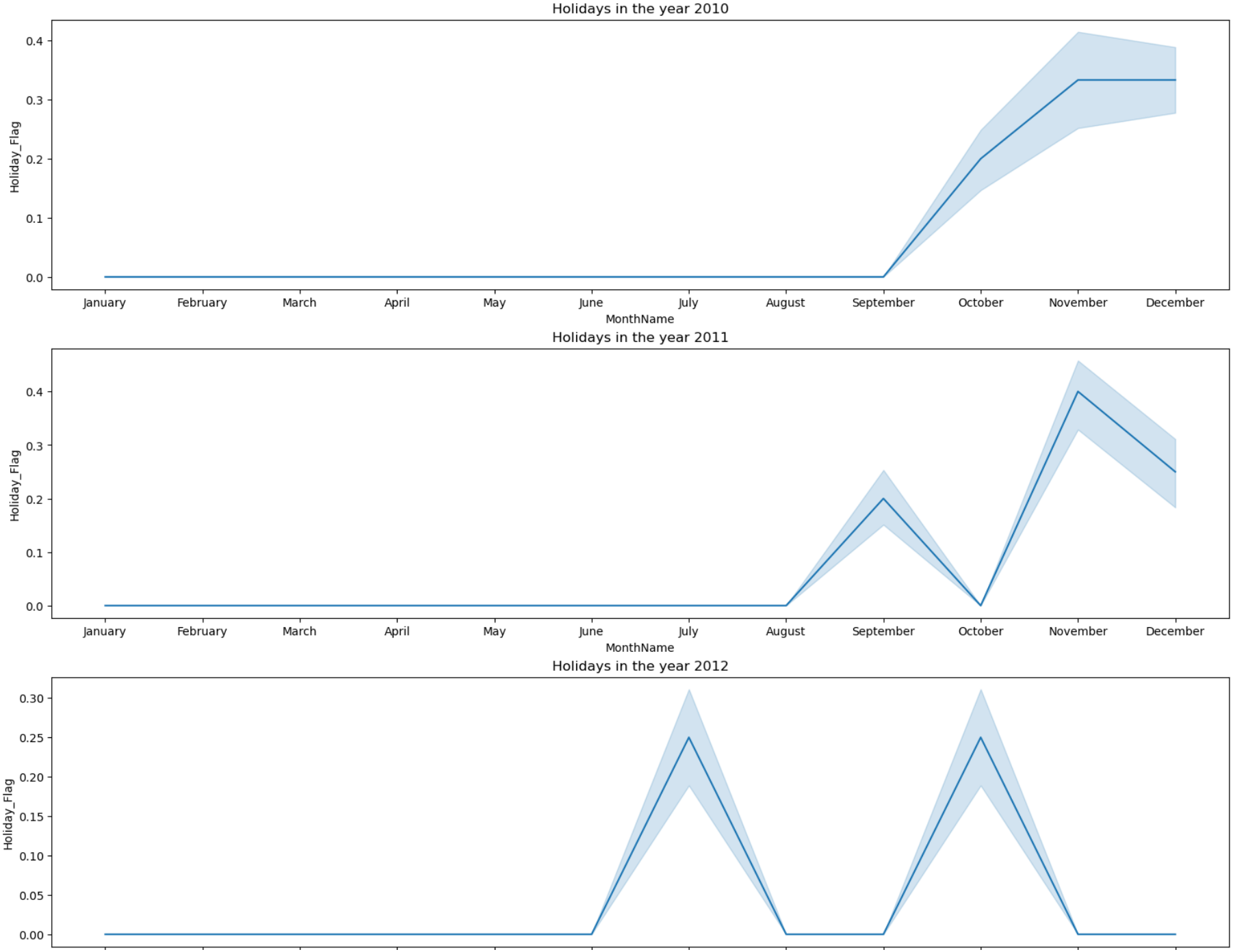

Weekly Sales in month 1 in the year 2012
$48,281,650
$42,023,078
$42,080,997
$39,834,975
Weekly Sales in month 2 in the year 2012
$46,861,035
$50,197,057
$45,771,507
Weekly Sales in month 3 in the year 2012
$46,085,608
$47,485,900
$46,901,505
$44,993,794
$45,272,862
Weekly Sales in month 4 in the year 2012
$47,124,198
$46,629,261
$45,072,530
$43,716,799
Weekly Sales in month 5 in the year 2012
$47,566,639
$46,823,939
$47,892,463
Weekly Sales in month 6 in the year 2012
$104,755,338
$44,955,422
$48,412,111
$47,668,285
$46,597,112
Weekly Sales in month 7 in the year 2012
$94,429,791
$46,059,543
$44,097,155
Weekly Sales in month 8 in the year 2012
$49,651,172
$47,354,452
$47,447,324
$47,159,639
Weekly Sales in month 9 in the year 2012
$47,480,454
$44,226,039
$44,354,547
$43,734,899
Weekly Sales in month 10 in the year 2012
$50,009,408
$47,403,451
$45,122,411
$45,544,116
Weekly Sales in month 11 in the year 2012
$46,925,879
Weekly Sales in month 12 in the year 2012
$46,128,514
Weekly Sales
Week

Holidays

In [30]:
```
plt.subplots(figsize = (15,5))
sns.lineplot(data,x = 'Date', y = 'Holiday_Flag')
plt.title("Holidays",fontsize=15)
plt.show()
```

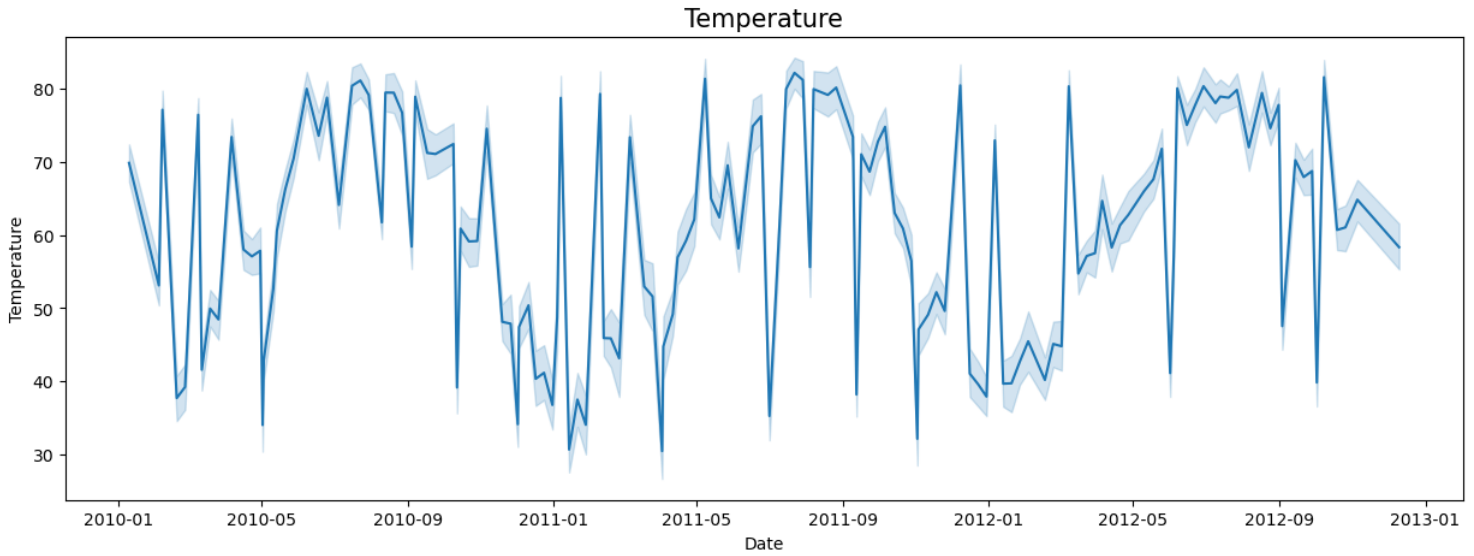


In [31]:
```
plt.subplots(3,1,figsize = (15,12),constrained_layout = True)
plt.subplot(3,1,1)
sns.lineplot(data_2010,x='MonthName',y = 'Holiday_Flag')
plt.title('Holidays in the year 2010')
plt.subplot(3,1,2)
sns.lineplot(data_2011,x='MonthName',y = 'Holiday_Flag')
plt.title('Holidays in the year 2011')
plt.subplot(3,1,3)
sns.lineplot(data_2012,x='MonthName',y = 'Holiday_Flag')
plt.title('Holidays in the year 2012')
plt.show()
```

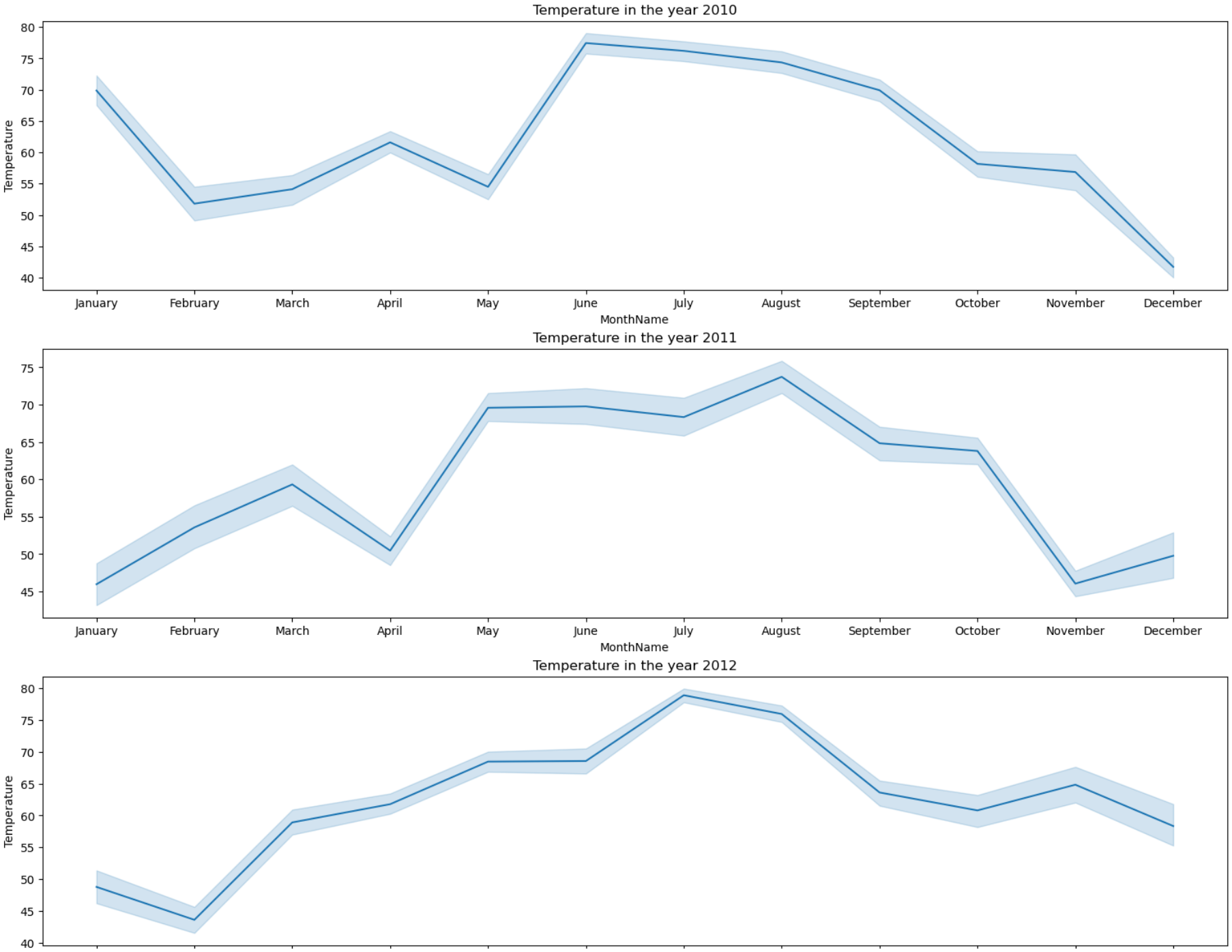

Holidays in the year 2010
Holiday_Flag
0.0
0.1
0.2
0.3
0.4
January
February
March
April
May
June
July
August
September
October
November
December
MonthName
Holidays in the year 2011
Holiday_Flag
0.0
0.1
0.2
0.3
0.4
January
February
March
April
May
June
July
August
September
October
November
December
MonthName
Holidays in the year 2012
Holiday_Flag
0.00
0.05
0.10
0.15
0.20
0.25
0.30

Temperature

In [32]:
```
plt.subplots(figsize = (15,5))
sns.lineplot(data,x = 'Date', y = 'Temperature')
plt.title("Temperature",fontsize=15)
plt.show()
```

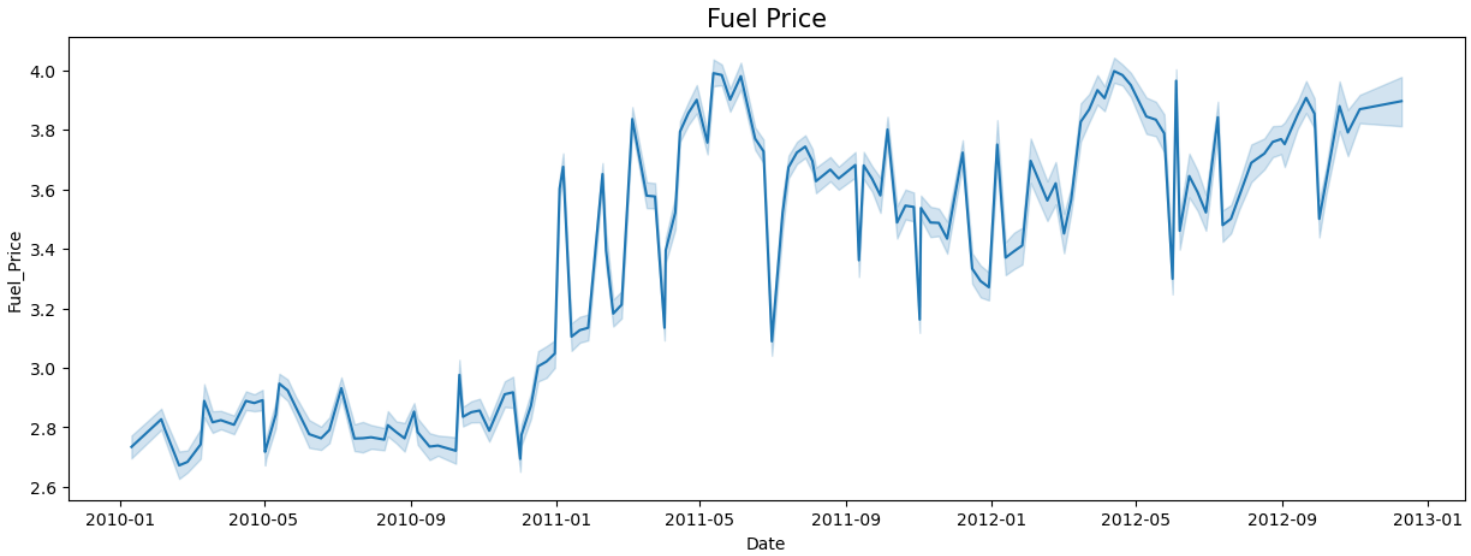


In [33]:
```
plt.subplots(3,1,figsize = (15,12),constrained_layout = True)
plt.subplot(3,1,1)
sns.lineplot(data_2010,x='MonthName',y = 'Temperature')
plt.title('Temperature in the year 2010')
plt.subplot(3,1,2)
sns.lineplot(data_2011,x='MonthName',y = 'Temperature')
plt.title('Temperature in the year 2011')
plt.subplot(3,1,3)
sns.lineplot(data_2012,x='MonthName',y = 'Temperature')
plt.title('Temperature in the year 2012')
plt.show()
```


Temperature in the year 2010
Temperature
80
75
70
65
60
55
50
45
40
January
February
March
April
May
June
July
August
September
October
November
December
MonthName
Temperature in the year 2011
Temperature
75
70
65
60
55
50
45
January
February
March
April
May
June
July
August
September
October
November
December
MonthName
Temperature in the year 2012
Temperature
80
75
70
65
60
55
50
45
40

Fuel Price

In [34]:
```
plt.subplots(figsize = (15,5))
sns.lineplot(data,x = 'Date', y = 'Fuel_Price')
plt.title("Fuel Price",fontsize=15)
plt.show()
```

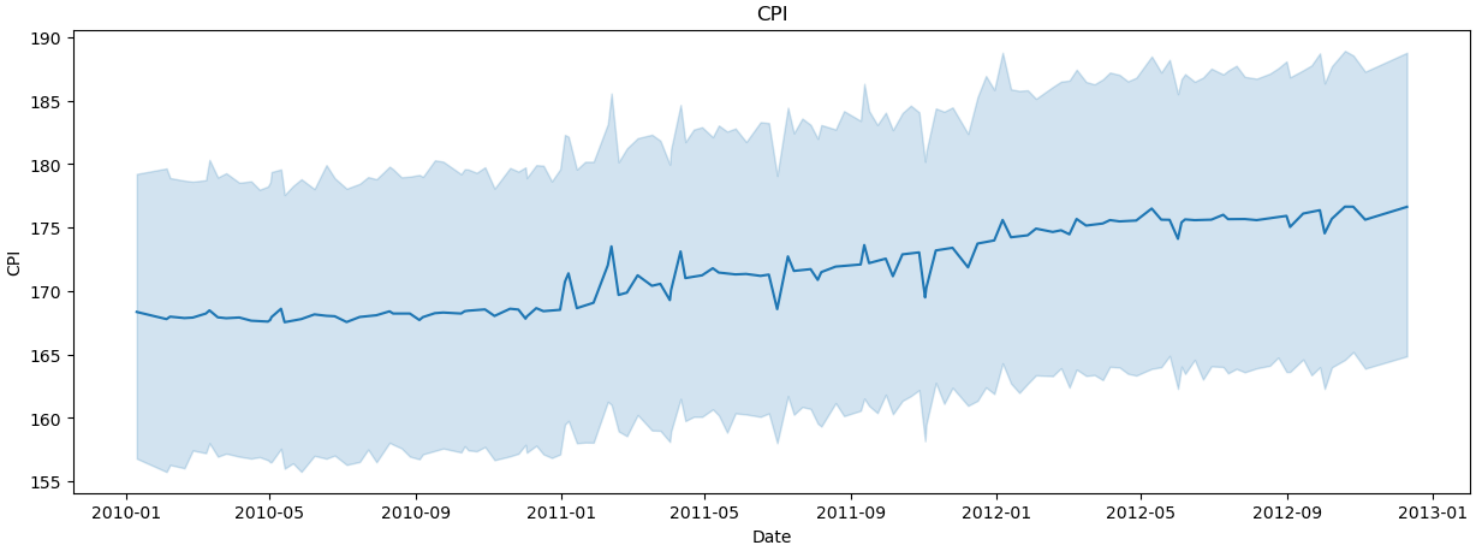


In [35]:
```
plt.subplots(3,1,figsize = (15,12),constrained_layout = True)
plt.subplot(3,1,1)
sns.lineplot(data_2010,x='MonthName',y = 'Fuel_Price')
plt.title('Fuel Price in the year 2010')
plt.subplot(3,1,2)
sns.lineplot(data_2011,x='MonthName',y = 'Fuel_Price')
plt.title('Fuel Price in the year 2011')
plt.subplot(3,1,3)
sns.lineplot(data_2012,x='MonthName',y = 'Fuel_Price')
plt.title('Fuel Price in the year 2012')
plt.show()
```

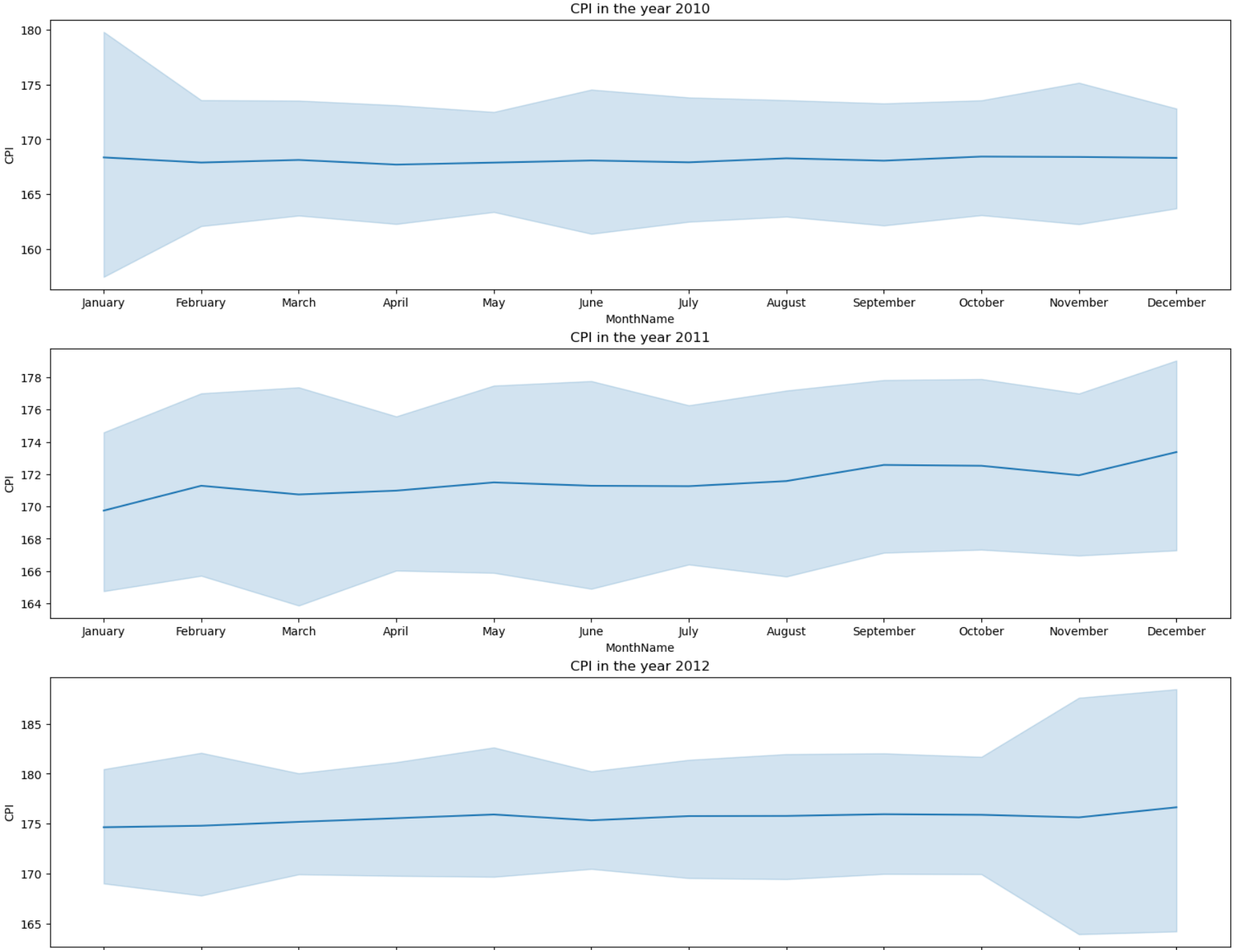

Fuel Price in the year 2010
Fuel_Price
2.90
2.85
2.80
2.75
2.70
January
February
March
April
May
June
July
August
September
October
November
December
MonthName
Fuel Price in the year 2011
Fuel_Price
3.9
3.8
3.7
3.6
3.5
3.4
3.3
January
February
March
April
May
June
July
August
September
October
November
December
MonthName
Fuel Price in the year 2012
Fuel_Price
4.0
3.9
3.8
3.7
3.6
3.5

CPI

In [36]:
```
plt.subplots(figsize = (15,5))
sns.lineplot(data,x = 'Date', y = 'CPI')
plt.title("CPI")
plt.show()
```

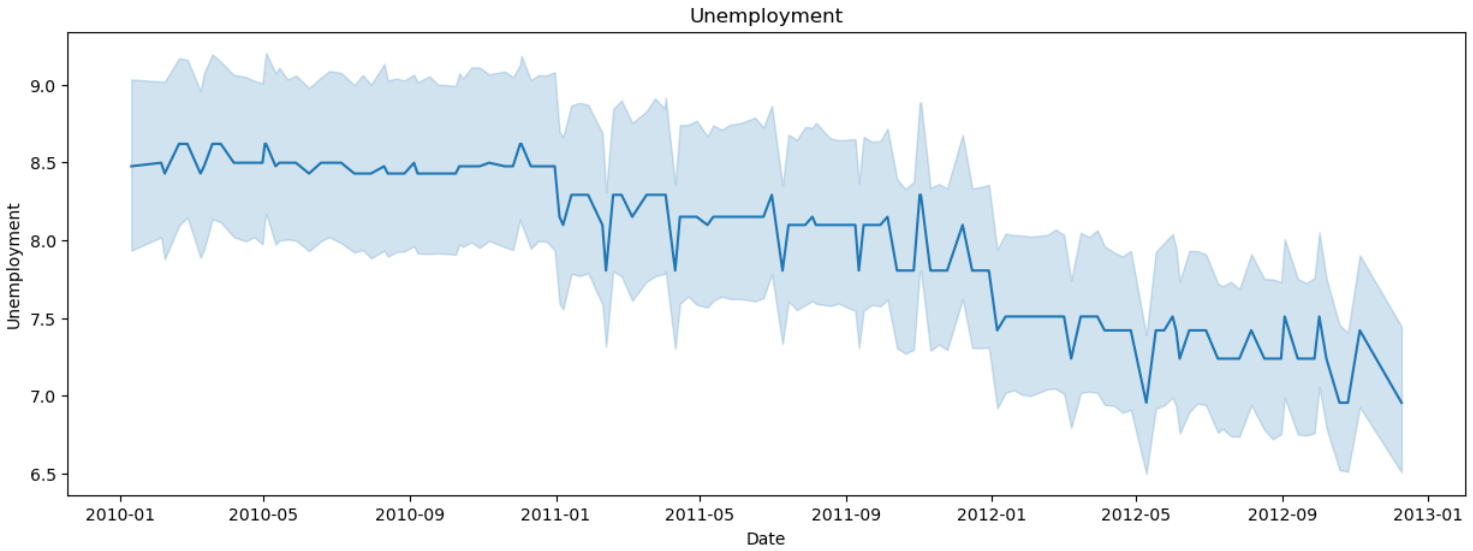


In [37]:
```
plt.subplots(3,1,figsize = (15,12),constrained_layout = True)
plt.subplot(3,1,1)
sns.lineplot(data_2010,x='MonthName',y = 'CPI')
plt.title('CPI in the year 2010')
plt.subplot(3,1,2)
sns.lineplot(data_2011,x='MonthName',y = 'CPI')
plt.title('CPI in the year 2011')
plt.subplot(3,1,3)
sns.lineplot(data_2012,x='MonthName',y = 'CPI')
plt.title('CPI in the year 2012')
plt.show()
```

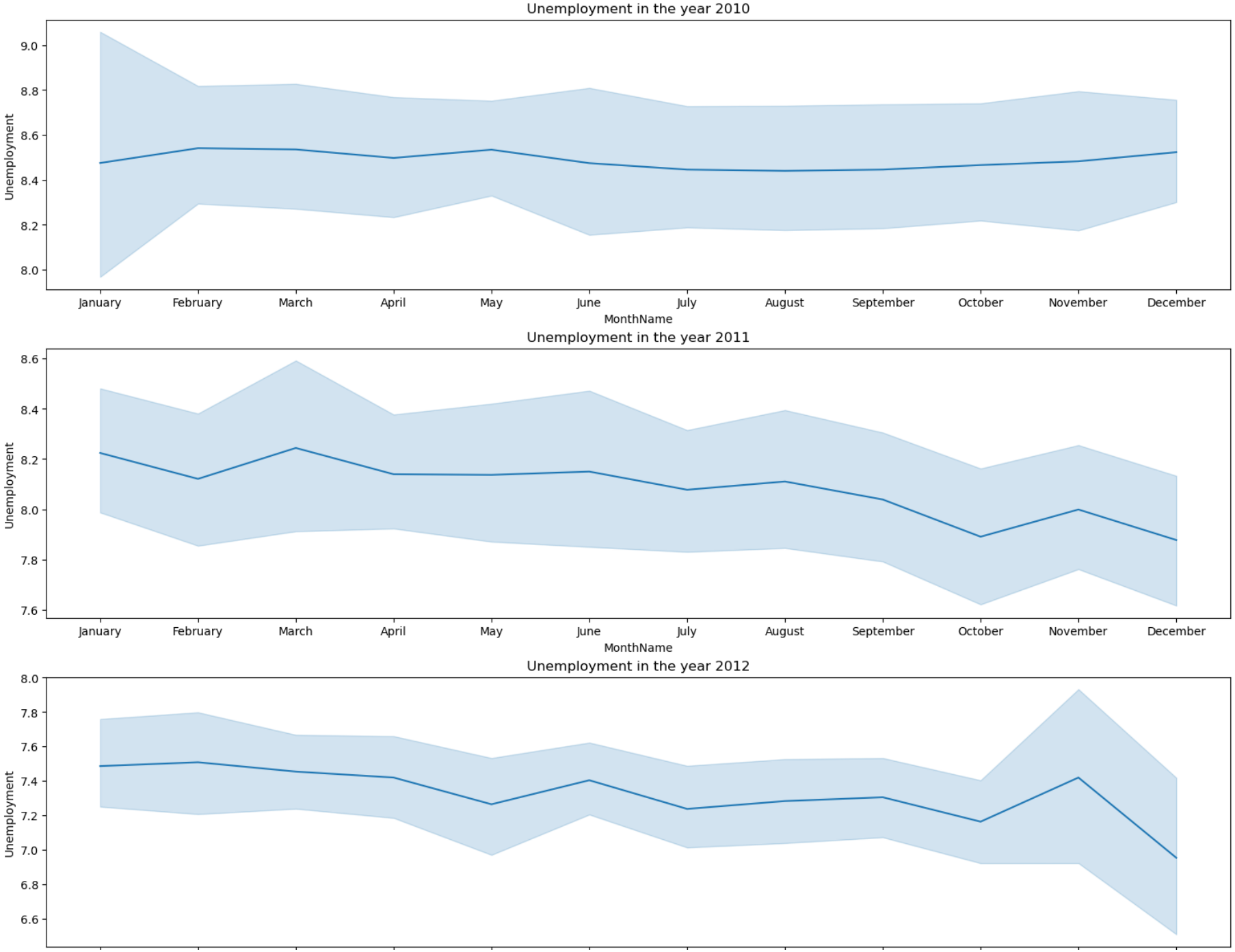

CPI in the year 2010
CPI
MonthName
January
February
March
April
May
June
July
August
September
October
November
December
CPI in the year 2011
CPI
MonthName
CPI in the year 2012
CPI

Unemployment

In [38]:
```
plt.subplots(figsize = (15,5))
sns.lineplot(data,x = 'Date', y = 'Unemployment')
plt.title("Unemployment")
plt.show()
```

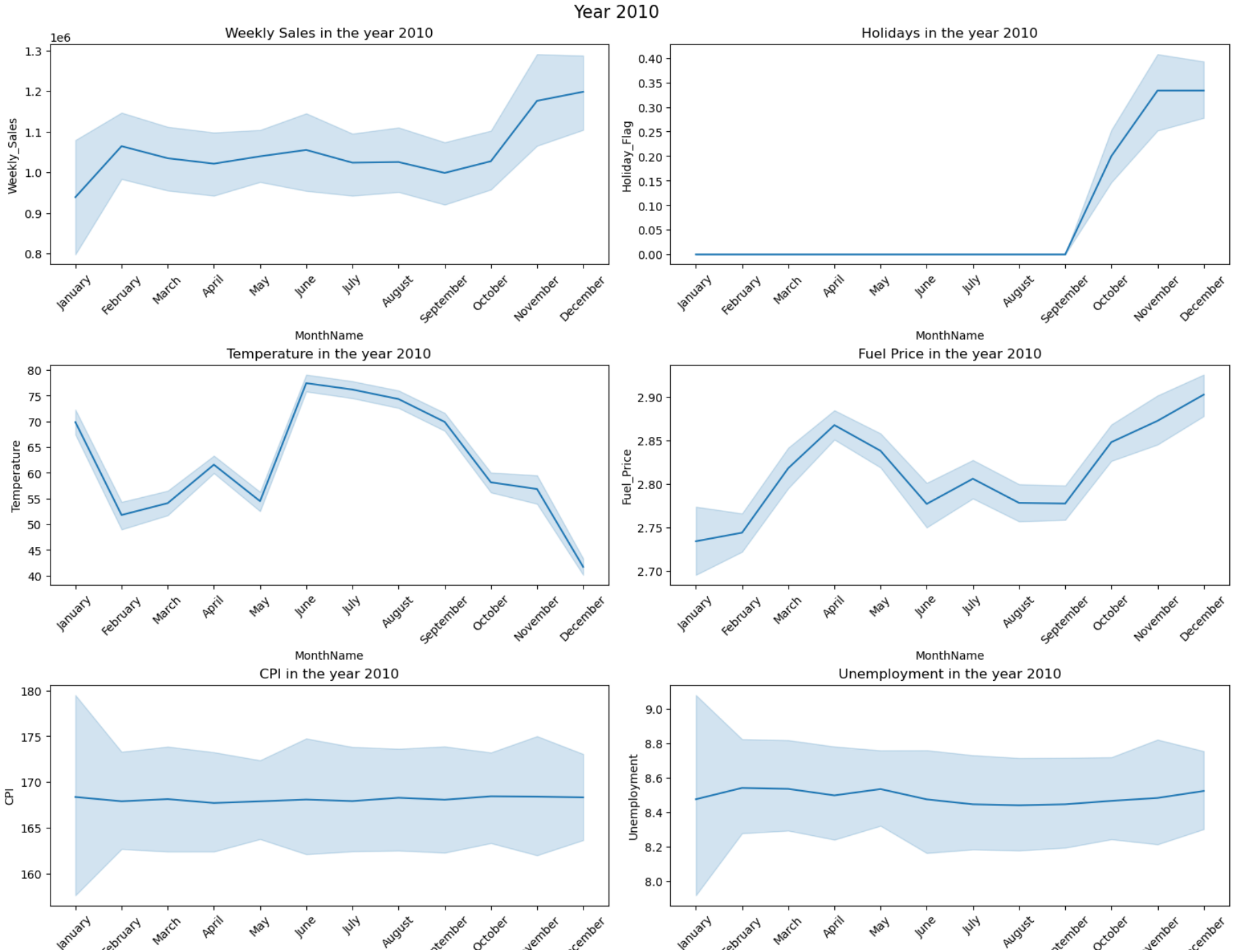


In [39]:
```
plt.subplots(3,1,figsize = (15,12),constrained_layout = True)
plt.subplot(3,1,1)
sns.lineplot(data_2010,x='MonthName',y = 'Unemployment')
plt.title('Unemployment in the year 2010')
plt.subplot(3,1,2)
sns.lineplot(data_2011,x='MonthName',y = 'Unemployment')
plt.title('Unemployment in the year 2011')
plt.subplot(3,1,3)
sns.lineplot(data_2012,x='MonthName',y = 'Unemployment')
plt.title('Unemployment in the year 2012')
plt.show()
```


Unemployment in the year 2010
Unemployment
9.0
8.8
8.6
8.4
8.2
8.0
January
February
March
April
May
June
July
August
September
October
November
December
MonthName
Unemployment in the year 2011
Unemployment
8.6
8.4
8.2
8.0
7.8
7.6
January
February
March
April
May
June
July
August
September
October
November
December
MonthName
Unemployment in the year 2012
Unemployment
8.0
7.8
7.6
7.4
7.2
7.0
6.8
6.6

Visualizing features in year 2010

In [40]:
```
plt.subplots(3,2,figsize = (15,12),constrained_layout = True)
plt.suptitle("Year 2010", fontsize=15)
plt.subplot(3,2,1)
sns.lineplot(data_2010,x='MonthName',y = 'Weekly_Sales')
plt.xticks(rotation = 45)
plt.title('Weekly Sales in the year 2010')
plt.subplot(3,2,2)
sns.lineplot(data_2010,x='MonthName',y = 'Holiday_Flag')
plt.xticks(rotation = 45)
plt.title('Holidays in the year 2010')
plt.subplot(3,2,3)
sns.lineplot(data_2010,x='MonthName',y = 'Temperature')
plt.xticks(rotation = 45)
plt.title('Temperature in the year 2010')
plt.subplot(3,2,4)
sns.lineplot(data_2010,x='MonthName',y = 'Fuel_Price')
plt.xticks(rotation = 45)
plt.title('Fuel Price in the year 2010')
plt.subplot(3,2,5)
sns.lineplot(data_2010,x='MonthName',y = 'CPI')
plt.xticks(rotation = 45)
plt.title('CPI in the year 2010')
plt.subplot(3,2,6)
sns.lineplot(data_2010,x='MonthName',y = 'Unemployment')
plt.xticks(rotation = 45)
plt.title('Unemployment in the year 2010')
plt.show()
```


Year 2010
Weekly Sales in the year 2010
1e6
Weekly_Sales
MonthName
Holidays in the year 2010
Holiday_Flag
MonthName
Temperature in the year 2010
Temperature
MonthName
Fuel Price in the year 2010
Fuel_Price
MonthName
CPI in the year 2010
CPI
Unemployment in the year 2010
Unemployment
January
February
March
April
May
June
July
August
September
October
November
December

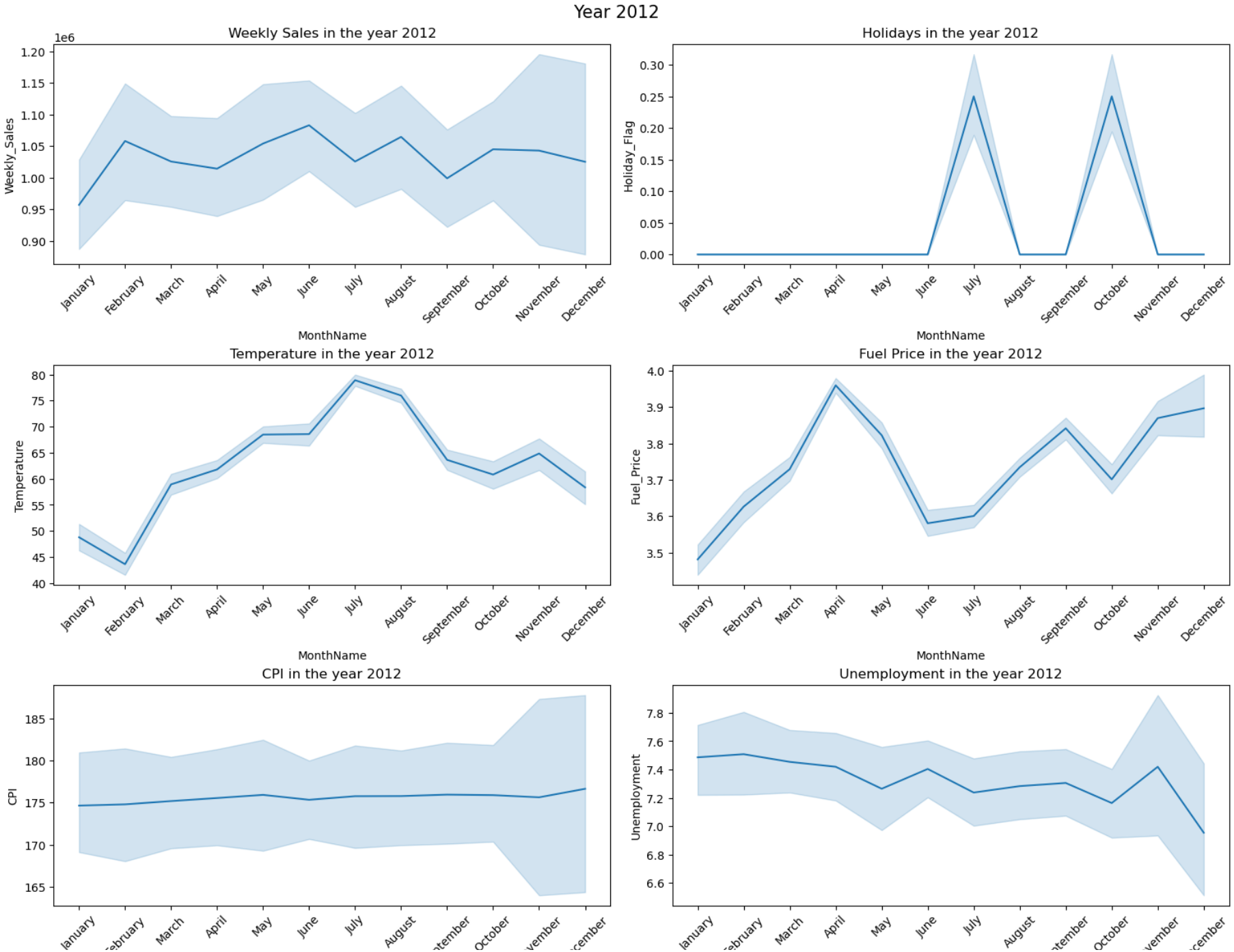
Visualizing features in year 2011

In [41]:
```
plt.subplots(3,2,figsize = (15,12),constrained_layout = True)
plt.suptitle("Year 2011", fontsize=15)
plt.subplot(3,2,1)
sns.lineplot(data_2011,x='MonthName',y = 'Weekly_Sales')
plt.xticks(rotation = 45)
plt.title('Weekly Sales in the year 2011')
plt.subplot(3,2,2)
sns.lineplot(data_2011,x='MonthName',y = 'Holiday_Flag')
plt.xticks(rotation = 45)
plt.title('Holidays in the year 2011')
plt.subplot(3,2,3)
sns.lineplot(data_2011,x='MonthName',y = 'Temperature')
plt.xticks(rotation = 45)
plt.title('Temperature in the year 2011')
plt.subplot(3,2,4)
sns.lineplot(data_2011,x='MonthName',y = 'Fuel_Price')
plt.xticks(rotation = 45)
plt.title('Fuel Price in the year 2011')
plt.subplot(3,2,5)
sns.lineplot(data_2011,x='MonthName',y = 'CPI')
plt.xticks(rotation = 45)
plt.title('CPI in the year 2011')
plt.subplot(3,2,6)
sns.lineplot(data_2011,x='MonthName',y = 'Unemployment')
plt.xticks(rotation = 45)
plt.title('Unemployment in the year 2011')
plt.show()
```


Year 2011
Weekly Sales in the year 2011
1e6
Weekly_Sales
MonthName
Holidays in the year 2011
Holiday_Flag
MonthName
Temperature in the year 2011
Temperature
MonthName
Fuel Price in the year 2011
Fuel_Price
MonthName
CPI in the year 2011
CPI
Unemployment in the year 2011
Unemployment
January
February
March
April
May
June
July
August
September
October
November
December

Visualizing features in year 2012

In [42]:
```
plt.subplots(3,2,figsize = (15,12),constrained_layout = True)
plt.suptitle("Year 2012", fontsize=15)
plt.subplot(3,2,1)
sns.lineplot(data_2012,x='MonthName',y = 'Weekly_Sales')
plt.xticks(rotation = 45)
plt.title('Weekly Sales in the year 2012')
plt.subplot(3,2,2)
sns.lineplot(data_2012,x='MonthName',y = 'Holiday_Flag')
plt.xticks(rotation = 45)
plt.title('Holidays in the year 2012')
plt.subplot(3,2,3)
sns.lineplot(data_2012,x='MonthName',y = 'Temperature')
plt.xticks(rotation = 45)
plt.title('Temperature in the year 2012')
plt.subplot(3,2,4)
sns.lineplot(data_2012,x='MonthName',y = 'Fuel_Price')
plt.xticks(rotation = 45)
plt.title('Fuel Price in the year 2012')
plt.subplot(3,2,5)
sns.lineplot(data_2012,x='MonthName',y = 'CPI')
plt.xticks(rotation = 45)
plt.title('CPI in the year 2012')
plt.subplot(3,2,6)
sns.lineplot(data_2012,x='MonthName',y = 'Unemployment')
plt.xticks(rotation = 45)
plt.title('Unemployment in the year 2012')
plt.show()
```


Year 2012
Weekly Sales in the year 2012
1e6
Weekly_Sales
MonthName
Holidays in the year 2012
Holiday_Flag
MonthName
Temperature in the year 2012
Temperature
MonthName
Fuel Price in the year 2012
Fuel_Price
MonthName
CPI in the year 2012
CPI
Unemployment in the year 2012
Unemployment
January
February
March
April
May
June
July
August
September
October
November
December

Preparing data for choosing regression model

In [43]:
```
reg_data = data.drop(columns=['Date','MonthName'])
reg_data.head()
```

Out[43]:

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Year	Month	Week
0	1	1643690.90	0	42.31	2.572	211.096358	8.106	2010	5	17
1	1	1641957.44	1	38.51	2.548	211.242170	8.106	2010	12	48
2	1	1611968.17	0	39.93	2.514	211.289143	8.106	2010	2	7
3	1	1409727.59	0	46.63	2.561	211.319643	8.106	2010	2	8
4	1	1554806.68	0	46.50	2.625	211.350143	8.106	2010	5	18

In [44]:
```
reg_data.shape
```

Out[44]:
```
(6435, 10)
```

In [45]:
```
X = reg_data.drop(columns='Weekly_Sales')
y = reg_data[['Weekly_Sales']]
```

Data Normalization

In [46]:
```
from sklearn.preprocessing import MinMaxScaler
scale = MinMaxScaler()
X = pd.DataFrame(scale.fit_transform(X),columns=X.columns)
X.head()
```

Out[46]:

	Store	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Year	Month	Week
0	0.0	0.0	0.434149	0.050100	0.840500	0.405118	0.0	0.363636	0.313725
1	0.0	1.0	0.396967	0.038076	0.841941	0.405118	0.0	1.000000	0.921569
2	0.0	0.0	0.410861	0.021042	0.842405	0.405118	0.0	0.090909	0.117647
3	0.0	0.0	0.476419	0.044589	0.842707	0.405118	0.0	0.090909	0.137255
4	0.0	0.0	0.475147	0.076653	0.843008	0.405118	0.0	0.363636	0.333333

In [47]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[47]:
```
((5148, 9), (1287, 9), (5148, 1), (1287, 1))
```

Importing regression models

In [48]:
```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
```

In [49]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
```

Helper functions for evaluation metrics

In [50]:
```
def evaluation_metrics(y_test,y_pred):
    mse = mean_squared_error(y_test,y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_test,y_pred)
    r2score = r2_score(y_test,y_pred)

    print(f'Mean Squared Error = {mse}')
    print(f'Mean Absolute Error = {mae}')
    print(f'Root Mean Squared Error = {rmse}')
    print(f'r2 score = {r2score}')
```

Linear Regression

In [51]:
```
linear_model = LinearRegression(normalize=False)
linear_model.fit(x_train,y_train)
linear_pred = linear_model.predict(x_test)
```

In [52]:
```
evaluation_metrics(y_test,linear_pred)
```

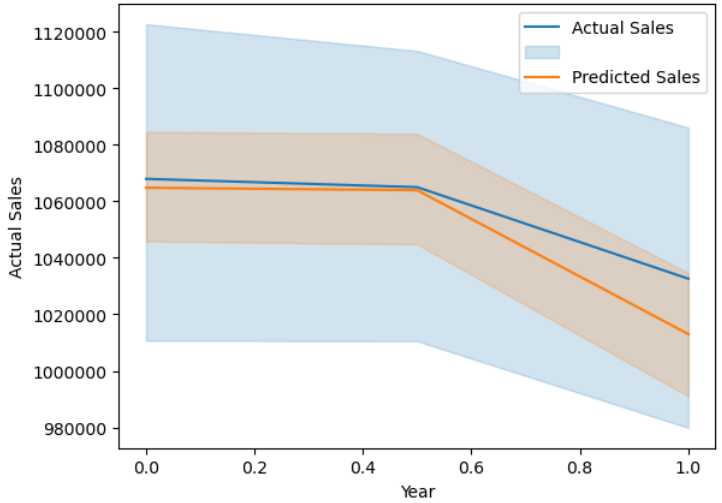
```
Mean Squared Error = 270465205323.04138
Mean Absolute Error = 433301.6691152136
Root Mean Squared Error = 520062.6936466808
r2 score = 0.12726215663704188
```

In [53]:
```
Comparison = x_test.copy("deep")
Comparison['Actual Sales'] = y_test
Comparison['Predicted Sales'] = linear_pred
Comparison.head()
```

Out[53]:

	Store	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Year	Month	Week	Actual Sales	Predicted Sales
6320	1.000000	0.0	0.763894	0.153808	0.558992	0.466168	0.0	0.636364	0.627451	708568.29	6.590440e+05
6003	0.931818	0.0	0.764090	1.000000	0.049861	0.293655	1.0	1.000000	0.960784	612379.90	9.571235e+05
3617	0.568182	1.0	0.295205	0.299599	0.066947	0.409239	0.0	0.909091	0.901961	1286833.62	1.211847e+06
1855	0.272727	0.0	0.624755	0.672846	0.049538	0.166954	1.0	0.363636	0.352941	2041918.74	1.322186e+06
5837	0.909091	0.0	0.537378	0.716934	0.707158	0.255703	1.0	0.272727	0.254902	1359770.73	7.211729e+05

In [54]:
```
sns.lineplot(Comparison, x = 'Year',y = 'Actual Sales')
sns.lineplot(Comparison, x = 'Year',y = 'Predicted Sales')
plt.ticklabel_format(useOffset=False,style='plain',axis='y')
plt.legend(['Actual Sales','','Predicted Sales'])
plt.show()
```



In [55]:
```
from sklearn.model_selection import cross_val_score
linear_scores = cross_val_score(linear_model,x_train,y_train,cv=10)
np.mean(linear_scores)
```

Out[55]:
```
0.14592723103006416
```

Random Forest Regression

In [56]:
```
rf_model = RandomForestRegressor()
rf_model.fit(x_train,y_train)
rf_pred = rf_model.predict(x_test)
```

In [57]:
```
evaluation_metrics(y_test,rf_pred)
```

```
Mean Squared Error = 12450621831.237871
Mean Absolute Error = 66669.47684382283
Root Mean Squared Error = 111582.35447971991
r2 score = 0.9598243003844288
```

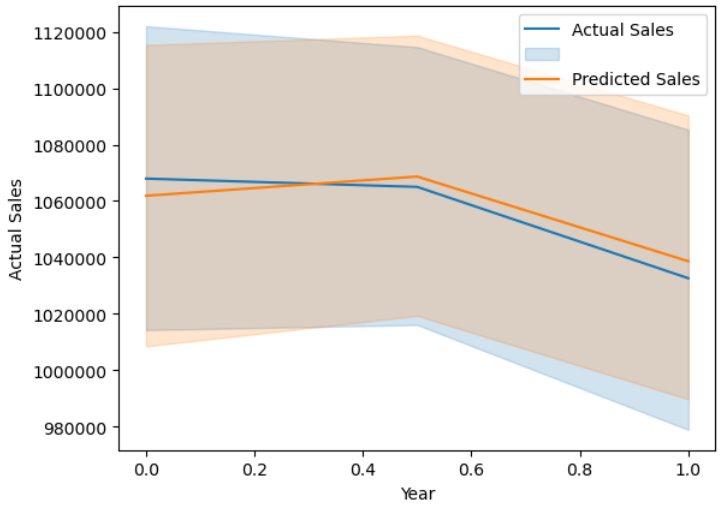
In [58]:
```
Comparison = x_test.copy("deep")
Comparison['Actual Sales'] = y_test
Comparison['Predicted Sales'] = rf_pred
Comparison.head()
```

Out[58]:

	Store	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Year	Month	Week	Actual Sales	Predicted Sales
6320	1.000000	0.0	0.763894	0.153808	0.558992	0.466168	0.0	0.636364	0.627451	708568.29	7.257241e+05
6003	0.931818	0.0	0.764090	1.000000	0.049861	0.293655	1.0	1.000000	0.960784	612379.90	5.779327e+05
3617	0.568182	1.0	0.295205	0.299599	0.066947	0.409239	0.0	0.909091	0.901961	1286833.62	1.112256e+06
1855	0.272727	0.0	0.624755	0.672846	0.049538	0.166954	1.0	0.363636	0.352941	2041918.74	2.055963e+06
5837	0.909091	0.0	0.537378	0.716934	0.707158	0.255703	1.0	0.272727	0.254902	1359770.73	1.344048e+06

In [59]:
```
sns.lineplot(Comparison, x = 'Year',y = 'Actual Sales')
sns.lineplot(Comparison, x = 'Year',y = 'Predicted Sales')
plt.ticklabel_format(useOffset=False,style='plain',axis='y')
```

```
plt.legend(['Actual Sales','','Predicted Sales'])
plt.show()
```



In [60]:
```
rf_scores = cross_val_score(rf_model,x_train,y_train,cv=10)
np.mean(rf_scores)
```

Out[60]:
```
0.9468933794404215
```

XG Boost Regression

In [61]:
```
xgb_model = XGBRegressor()
xgb_model.fit(x_train,y_train)
xgb_pred = xgb_model.predict(x_test)
```

In [62]:
```
evaluation_metrics(y_test,xgb_pred)
```

```
Mean Squared Error = 6480600305.456524
Mean Absolute Error = 53042.5374694056
Root Mean Squared Error = 80502.1757808851
r2 score = 0.9790883817105933
```

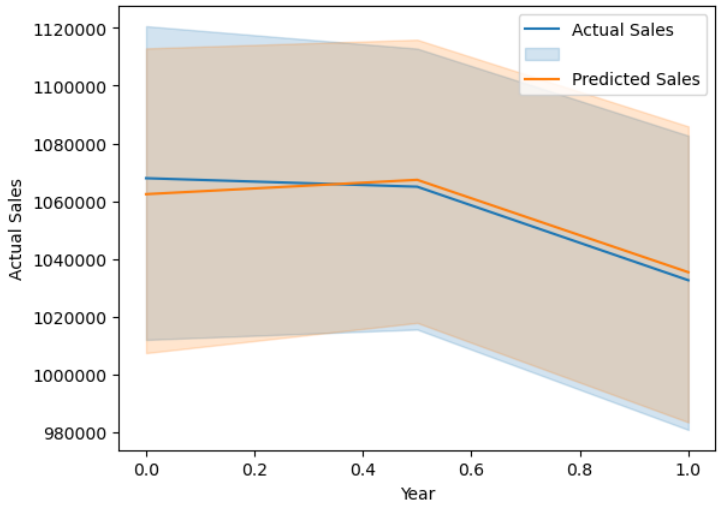
In [63]:
```
Comparison = x_test.copy("deep")
Comparison['Actual Sales'] = y_test
Comparison['Predicted Sales'] = xgb_pred
Comparison.head()
```

Out[63]:

	Store	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Year	Month	Week	Actual Sales	Predicted Sales
6320	1.000000	0.0	0.763894	0.153808	0.558992	0.466168	0.0	0.636364	0.627451	708568.29	7.431457e+05
6003	0.931818	0.0	0.764090	1.000000	0.049861	0.293655	1.0	1.000000	0.960784	612379.90	5.352404e+05
3617	0.568182	1.0	0.295205	0.299599	0.066947	0.409239	0.0	0.909091	0.901961	1286833.62	1.462036e+06
1855	0.272727	0.0	0.624755	0.672846	0.049538	0.166954	1.0	0.363636	0.352941	2041918.74	2.017106e+06
5837	0.909091	0.0	0.537378	0.716934	0.707158	0.255703	1.0	0.272727	0.254902	1359770.73	1.359186e+06

In [64]:
```
sns.lineplot(Comparison, x = 'Year',y = 'Actual Sales')
sns.lineplot(Comparison, x = 'Year',y = 'Predicted Sales')
plt.ticklabel_format(useOffset=False,style='plain',axis='y')
```

```
plt.legend(['Actual Sales','','Predicted Sales'])
plt.show()
```



In [65]:
```
xgb_scores = cross_val_score(xgb_model,x_train,y_train,cv=10)
np.mean(xgb_scores)
```

Out[65]:
```
0.9692994682576972
```

Installing Prophet

In [66]:
```
pip install prophet
```

```
Requirement already satisfied: prophet in /opt/conda/lib/python3.7/site-packages (1.1.1)
Requirement already satisfied: wheel>=0.37.0 in /opt/conda/lib/python3.7/site-packages (from prophet) (0.38.4)
Requirement already satisfied: setuptools>=42 in /opt/conda/lib/python3.7/site-packages (from prophet) (59.8.0)
Requirement already satisfied: setuptools-git>=1.2 in /opt/conda/lib/python3.7/site-packages (from prophet) (1.2)
Requirement already satisfied: matplotlib>=2.0.0 in /opt/conda/lib/python3.7/site-packages (from prophet) (3.5.3)
Requirement already satisfied: cmdstanpy>=1.0.4 in /opt/conda/lib/python3.7/site-packages (from prophet) (1.1.0)
Requirement already satisfied: pandas>=1.0.4 in /opt/conda/lib/python3.7/site-packages (from prophet) (1.3.5)
Requirement already satisfied: python-dateutil>=2.8.0 in /opt/conda/lib/python3.7/site-packages (from prophet) (2.8.2)
Requirement already satisfied: numpy>=1.15.4 in /opt/conda/lib/python3.7/site-packages (from prophet) (1.21.6)
Requirement already satisfied: holidays>=0.14.2 in /opt/conda/lib/python3.7/site-packages (from prophet) (0.21.13)
Requirement already satisfied: convertdate>=2.1.2 in /opt/conda/lib/python3.7/site-packages (from prophet) (2.4.0)
Requirement already satisfied: LunarCalendar>=0.0.9 in /opt/conda/lib/python3.7/site-packages (from prophet) (0.0.9)
Requirement already satisfied: tqdm>=4.36.1 in /opt/conda/lib/python3.7/site-packages (from prophet) (4.64.1)
Requirement already satisfied: pymeeus<=1,>=0.3.13 in /opt/conda/lib/python3.7/site-packages (from convertdate>=2.1.2->prophet) (0.5.12)
Requirement already satisfied: backports.zoneinfo in /opt/conda/lib/python3.7/site-packages (from holidays>=0.14.2->prophet) (0.2.1)
Requirement already satisfied: korean-lunar-calendar in /opt/conda/lib/python3.7/site-packages (from holidays>=0.14.2->prophet) (0.3.1)
Requirement already satisfied: hijri-converter in /opt/conda/lib/python3.7/site-packages (from holidays>=0.14.2->prophet) (2.2.4)
Requirement already satisfied: pytz in /opt/conda/lib/python3.7/site-packages (from LunarCalendar>=0.0.9->prophet) (2022.7.1)
Requirement already satisfied: ephem>=3.7.5.3 in /opt/conda/lib/python3.7/site-packages (from LunarCalendar>=0.0.9->prophet) (4.1.4)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.7/site-packages (from matplotlib>=2.0.0->prophet) (9.4.0)
Requirement already satisfied: pyparsing>=2.2.1 in /opt/conda/lib/python3.7/site-packages (from matplotlib>=2.0.0->prophet) (3.0.9)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.7/site-packages (from matplotlib>=2.0.0->prophet) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.7/site-packages (from matplotlib>=2.0.0->prophet) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.7/site-packages (from matplotlib>=2.0.0->prophet) (23.0)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.7/site-packages (from matplotlib>=2.0.0->prophet) (4.38.0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packages (from python-dateutil>=2.8.0->prophet) (1.16.0)
Requirement already satisfied: typing-extensions in /opt/conda/lib/python3.7/site-packages (from kiwisolver>=1.0.1->matplotlib>=2.0.0->prophet) (4.4.0)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
Note: you may need to restart the kernel to use updated packages.
```

Preparing data for forecasting

In [67]:
```
time_data = data[['Date','Weekly_Sales']]
time_data.columns = ['ds','y']
time_data.head()
```

Out[67]:

	ds	y
0	2010-05-02	1643690.90
1	2010-12-02	1641957.44
2	2010-02-19	1611968.17
3	2010-02-26	1409727.59
4	2010-05-03	1554806.68

In [68]:
```
data.columns
```

Out[68]:
```
Index(['Store', 'Date', 'Weekly_Sales', 'Holiday_Flag', 'Temperature',
       'Fuel_Price', 'CPI', 'Unemployment', 'Year', 'Month', 'MonthName',
       'Week'],
      dtype='object')
```

In [69]:
```
regressors_df = data[['Date','Holiday_Flag','Temperature','Fuel_Price','CPI','Unemployment']]
regressors_df = regressors_df.rename(columns={'Date':'ds'})
merged_df = pd.merge(time_data,regressors_df,on='ds',how='left')
```

In [70]:
```
merged_df.head()
```

Out[70]:

	ds	y	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	2010-05-02	1643690.9	0	42.31	2.572	211.096358	8.106
1	2010-05-02	1643690.9	0	40.19	2.572	210.752605	8.324
2	2010-05-02	1643690.9	0	45.71	2.572	214.424881	7.368
3	2010-05-02	1643690.9	0	43.76	2.598	126.442065	8.623
4	2010-05-02	1643690.9	0	39.70	2.572	211.653972	6.566

Adding Regressors

In [71]:
```
from prophet import Prophet
m = Prophet(daily_seasonality=True)
m.add_regressor('Holiday_Flag')
m.add_regressor('Temperature')
m.add_regressor('Fuel_Price')
m.add_regressor('CPI')
m.add_regressor('Unemployment')
model_fit = m.fit(merged_df)
```

```
13:04:36 - cmdstanpy - INFO - Chain [1] start processing
13:06:35 - cmdstanpy - INFO - Chain [1] done processing
```

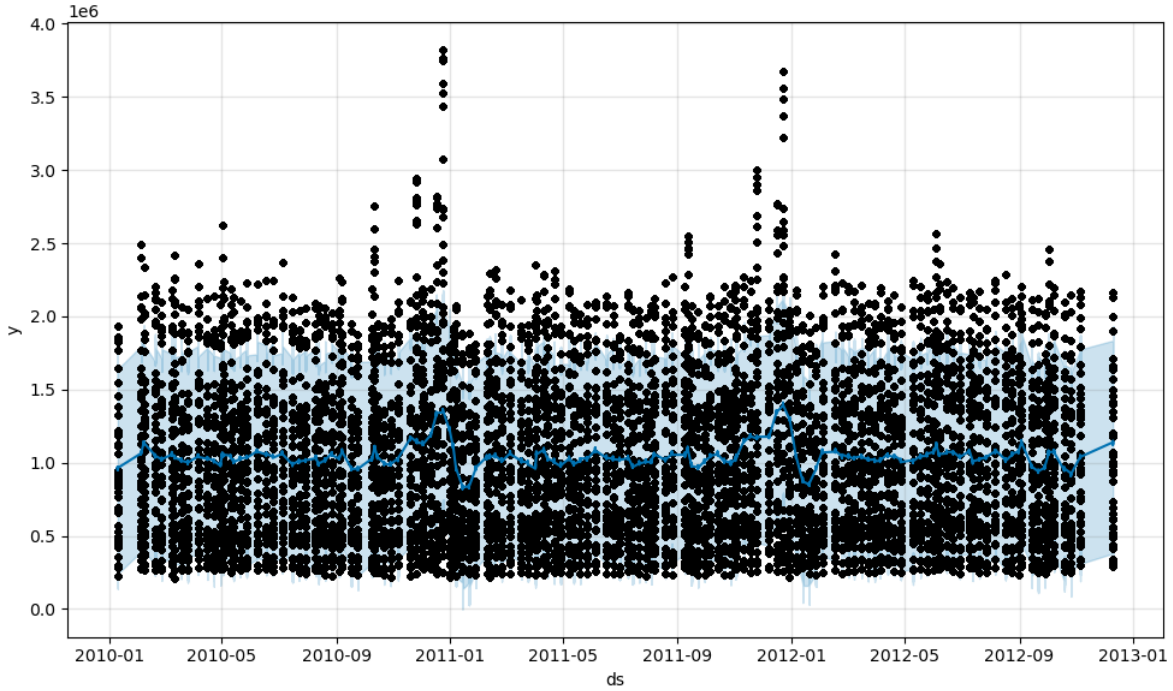
Predicting Model

In [72]:
```
forecast = model_fit.predict(merged_df)
forecast[['ds','yhat','Holiday_Flag','Temperature','Fuel_Price','CPI','Unemployment']].tail()
```

Out[72]:

	ds	yhat	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
289570	2012-12-10	1.127929e+06	0.0	-5213.135863	-8776.555902	-1177.894957	1441.991151
289571	2012-12-10	1.135707e+06	0.0	-4337.707708	-1917.604337	1469.024776	-1160.981593
289572	2012-12-10	1.139140e+06	0.0	6266.268817	-5303.568893	-950.775300	-2526.378779
289573	2012-12-10	1.142118e+06	0.0	7282.895062	-5303.568893	-950.775300	-564.480662
289574	2012-12-10	1.139434e+06	0.0	340.655659	-1933.426602	-1177.894957	550.517232

In [73]:
```
model_fit.plot(forecast);
```



In [74]:
```
model_fit.plot_components(forecast);
```

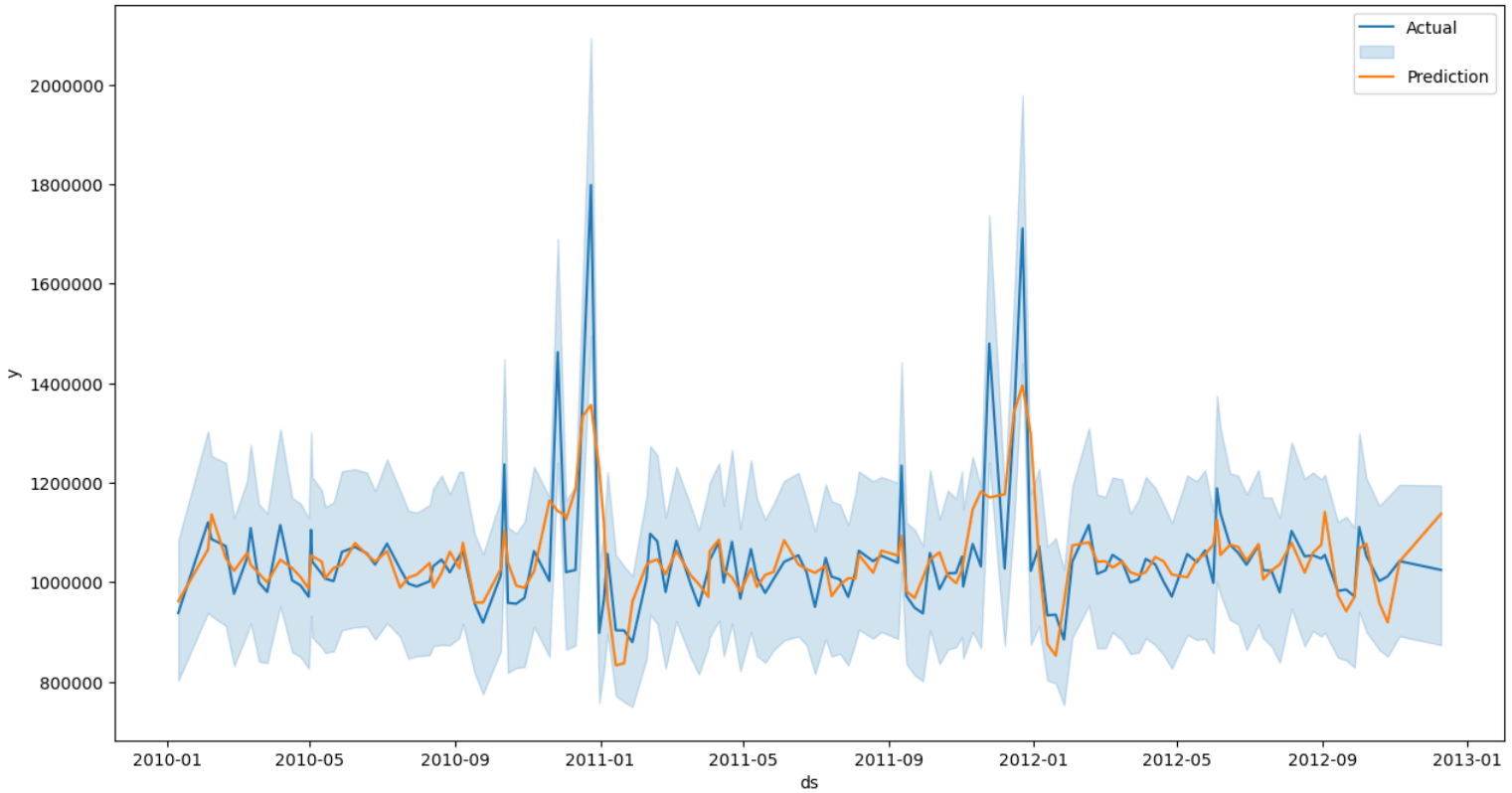

trend
450000
400000
350000
300000
2010-01
2010-05
2010-09
2011-01
2011-05
2011-09
2012-01
2012-05
2012-09
2013-01
ds
weekly
40000
20000
0
−20000
−40000
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week
yearly
300000
200000
100000
0
−100000
−200000
−300000
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year
daily
600000
400000
200000
0
−200000
00:00:00
03:25:42
06:51:25
10:17:08
13:42:51
17:08:34
20:34:17
00:00:00

In [75]:
```
predictions = forecast[['ds','yhat']]
predictions = predictions.rename(columns = {'ds':'Date','yhat':'Weekly_Sales'})
predictions.head()
```

Out[75]:

	Date	Weekly_Sales
0	2010-01-10	962791.406610
1	2010-01-10	955604.109832
2	2010-01-10	962338.212445
3	2010-01-10	962623.013203
4	2010-01-10	961659.148182

In [76]:
```
plt.subplots(figsize = (15,8))
sns.lineplot(time_data,x = 'ds',y='y',legend='full')
sns.lineplot(predictions,x = 'Date',y = 'Weekly_Sales',legend='full')
plt.legend(labels = ['Actual','','Prediction'])
plt.ticklabel_format(useOffset=False,style='plain',axis='y')
plt.show()
```



In [77]:
```
predictions['Date'] = pd.to_datetime(predictions['Date'])
predictions['Year'] = predictions['Date'].dt.year
predictions['Month'] = predictions['Date'].dt.month
predictions['MonthName'] = predictions['Date'].dt.month_name()
predictions['Week'] = predictions['Date'].dt.week
```

In [78]:
```
predictions.head()
```

Out[78]:

	Date	Weekly_Sales	Year	Month	MonthName	Week
0	2010-01-10	962791.406610	2010	1	January	1
1	2010-01-10	955604.109832	2010	1	January	1
2	2010-01-10	962338.212445	2010	1	January	1
3	2010-01-10	962623.013203	2010	1	January	1
4	2010-01-10	961659.148182	2010	1	January	1

In [79]:
```
predictions.shape
```

Out[79]:
```
(289575, 6)
```

In [80]:
```
x_time  = predictions.drop(columns=['Date','Weekly_Sales','MonthName'])
y_time = predictions[['Weekly_Sales']]
```

In [81]:
```
x_time_train,x_time_test,y_time_train,y_time_test = train_test_split(x_time,y_time,test_size=0.2,random_state=20)
```

Evaluating Model

In [82]:
```
xgb_time_model = XGBRegressor()
xgb_time_model.fit(x_time_train,y_time_train)
xgb_time_pred = xgb_time_model.predict(x_time_test)
```

In [83]:
```
x_time_train.shape,y_time_train.shape,x_time_test.shape,y_time_test.shape,xgb_time_pred.shape
```

Out[83]:
```
((231660, 3), (231660, 1), (57915, 3), (57915, 1), (57915,))
```

In [84]:
```
evaluation_metrics(y_time_test,xgb_time_pred)
```

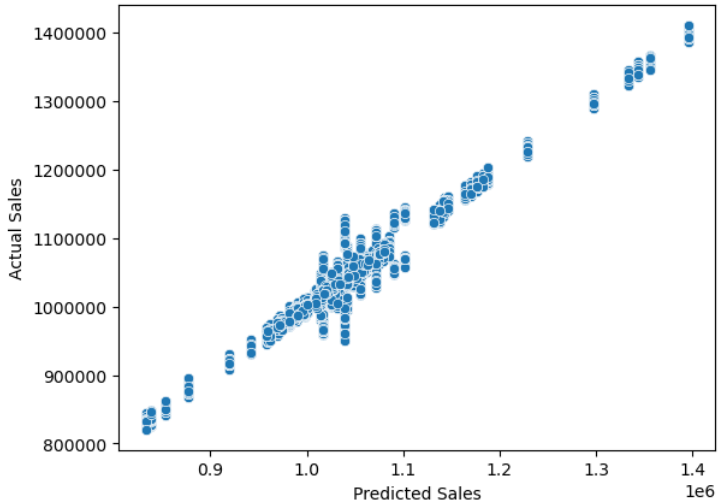
```
Mean Squared Error = 279145472.6275845
Mean Absolute Error = 8950.858743231493
Root Mean Squared Error = 16707.647130209105
r2 score = 0.9600973297512951
```

In [85]:
```
Comparison = x_time_test.copy("deep")
Comparison['Actual Sales'] = y_time_test
Comparison['Predicted Sales'] = xgb_time_pred
Comparison.head()
```

Out[85]:

	Year	Month	Week	Actual Sales	Predicted Sales
251272	2012	6	26	1.046945e+06	1.043121e+06
212922	2012	2	7	1.086615e+06	1.080605e+06
155918	2011	7	29	9.872480e+05	9.950811e+05
234202	2012	4	17	1.012230e+06	1.015952e+06
216083	2012	2	8	1.038634e+06	1.041675e+06

In [86]:
```
sns.scatterplot(Comparison,x='Predicted Sales',y='Actual Sales')
plt.ticklabel_format(useOffset=False,style='plain',axis='y')
plt.show()
```



In [87]:
```
xgb_time_scores = cross_val_score(xgb_time_model,x_time_train,y_time_train,cv=10)
np.mean(xgb_time_scores)
```

Out[87]:
```
0.9596304514463403
```

Forecasting Model for 12 weeks

In [88]:
```
m2 = Prophet(daily_seasonality=True)
m2.fit(time_data)
future = m2.make_future_dataframe(periods = 84)
future.tail(10)
```

```
13:09:25 - cmdstanpy - INFO - Chain [1] start processing
13:09:26 - cmdstanpy - INFO - Chain [1] done processing
```

Out[88]:

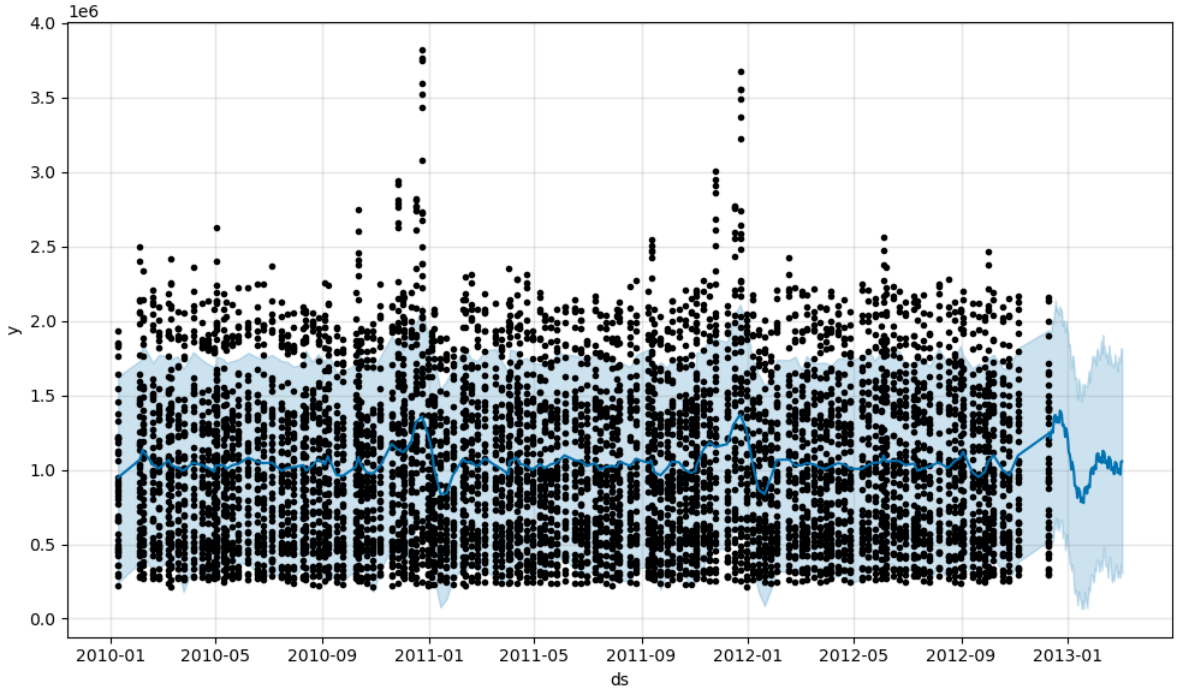
	ds
217	2013-02-23
218	2013-02-24
219	2013-02-25
220	2013-02-26
221	2013-02-27
222	2013-02-28
223	2013-03-01
224	2013-03-02
225	2013-03-03
226	2013-03-04

In [89]:
```
forecast2 = m2.predict(future)
forecast2[['ds','yhat','yhat_upper','yhat_lower']].tail()
```

Out[89]:

	ds	yhat	yhat_upper	yhat_lower
222	2013-02-28	1.003895e+06	1.757626e+06	322727.153306
223	2013-03-01	1.003094e+06	1.718388e+06	295393.524561
224	2013-03-02	9.681256e+05	1.749347e+06	277325.002089
225	2013-03-03	1.054166e+06	1.818416e+06	363605.770942
226	2013-03-04	1.056829e+06	1.801762e+06	307479.111988

In [90]:
```
m2.plot(forecast2);
```



In [91]:
```
m2.plot_components(forecast2);
```


trend
450000
445000
440000
435000
430000
2010-01
2010-05
2010-09
2011-01
2011-05
2011-09
2012-01
2012-05
2012-09
2013-01
ds
weekly
40000
20000
0
−20000
−40000
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week
yearly
300000
200000
100000
0
−100000
−200000
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year
daily
600000
400000
200000
0
−200000
00:00:00
03:25:42
06:51:25
10:17:08
13:42:51
17:08:34
20:34:17
00:00:00

Conclusion

1. XG Boost Regression is not only fast but also efficient for time series model evaluation. It provides best results, and the company can trust the forecasting.
2. High sales in the forecasting were observed in the end of the year 2012, and for the upcoming weeks, sales will fall down but recover quickly.
3. This forecasting tells that there is a presence of seasonality. Every end of the year is good for sales, and every start of the year brings some fall in the sales and then recovery.