Walmart sales EDA

In [5]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [7]:

```
data = pd.read_csv('./walmart-sales-dataset-of-45stores (1).csv', parse_dates=['Date'])
data.head(10)
```

Out[7]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployr
0	1	2010-05-02	1643690.90	0	42.31	2.572	211.096358	8
1	1	2010-12-02	1641957.44	1	38.51	2.548	211.242170	8
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8
4	1	2010-05-03	1554806.68	0	46.50	2.625	211.350143	8
5	1	2010-12-03	1439541.59	0	57.79	2.667	211.380643	8
6	1	2010-03-19	1472515.79	0	54.58	2.720	211.215635	8
7	1	2010-03-26	1404429.92	0	51.45	2.732	211.018042	8
8	1	2010-02-04	1594968.28	0	62.27	2.719	210.820450	7
9	1	2010-09-04	1545418.53	0	65.86	2.770	210.622857	7

In [8]:

```
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   datetime64[ns]
 2   Weekly_Sales  6435 non-null   float64       
 3   Holiday_Flag  6435 non-null   int64         
 4   Temperature   6435 non-null   float64       
 5   Fuel_Price    6435 non-null   float64       
 6   CPI           6435 non-null   float64       
 7   Unemployment  6435 non-null   float64       
dtypes: datetime64[ns](1), float64(5), int64(2)
memory usage: 402.3 KB
```

In [9]:

```
data.describe()
```

Out[9]:

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Une
count	6435.000000	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.000000	6
mean	23.000000	1.046965e+06	0.069930	60.663782	3.358607	171.578394	
std	12.988182	5.643666e+05	0.255049	18.444933	0.459020	39.356712	
min	1.000000	2.099862e+05	0.000000	-2.060000	2.472000	126.064000	
25%	12.000000	5.533501e+05	0.000000	47.460000	2.933000	131.735000	
50%	23.000000	9.607460e+05	0.000000	62.670000	3.445000	182.616521	
75%	34.000000	1.420159e+06	0.000000	74.940000	3.735000	212.743293	
max	45.000000	3.818686e+06	1.000000	100.140000	4.468000	227.232807	

In [10]:

```
data.shape
```

Out[10]:

```
(6435, 8)
```

In [11]:

```
data.columns
```

Out[11]:

```
Index(['Store', 'Date', 'Weekly_Sales', 'Holiday_Flag', 'Temperature',
       'Fuel_Price', 'CPI', 'Unemployment'],
      dtype='object')
```

Checking the null percentages

In [12]:

```
data.isnull().mean()*100
```

Out[12]:

```
Store           0.0
Date            0.0
Weekly_Sales    0.0
Holiday_Flag    0.0
Temperature     0.0
Fuel_Price      0.0
CPI             0.0
Unemployment    0.0
dtype: float64
```

Checking for duplicate values

In [13]:

```
data.duplicated().sum()
```

Out[13]:

```
0
```

Sorting the dataframe based on the date

In [14]:

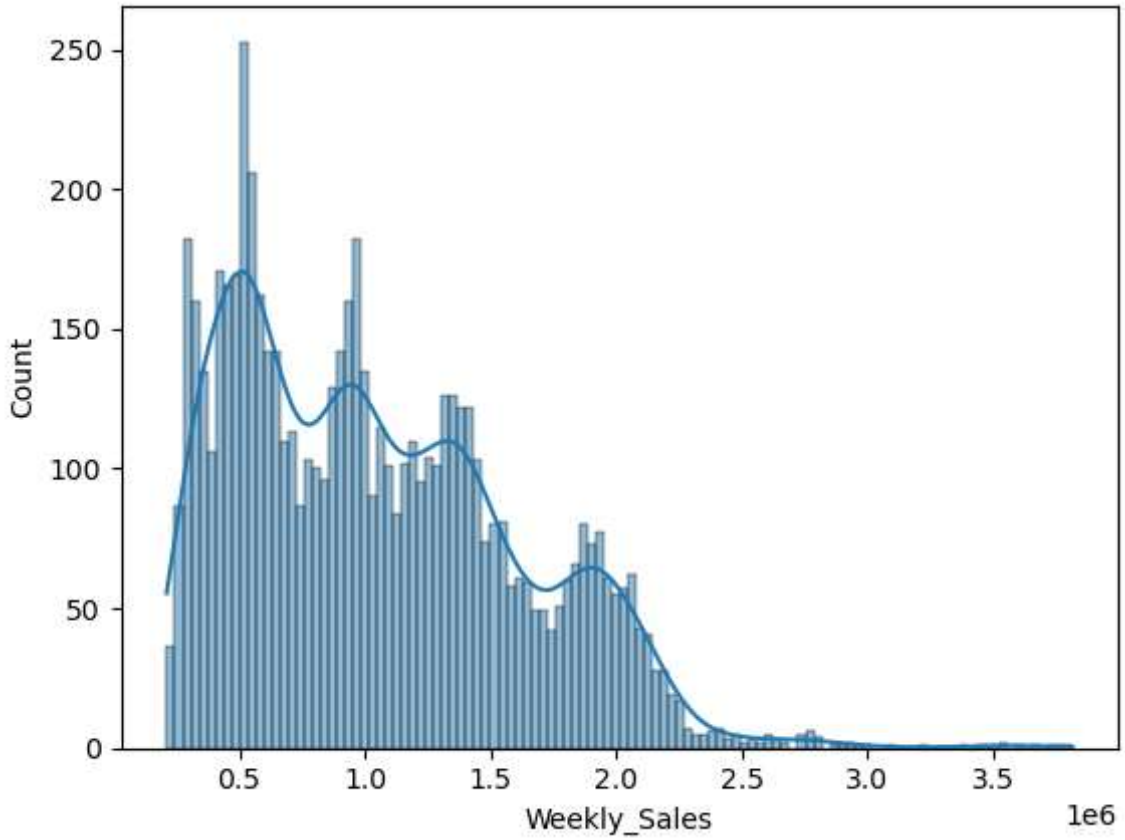
```
data.sort_values(by='Date', inplace=True)
data.reset_index(drop=True, inplace=True)
```

Univariate analysis

Histograms

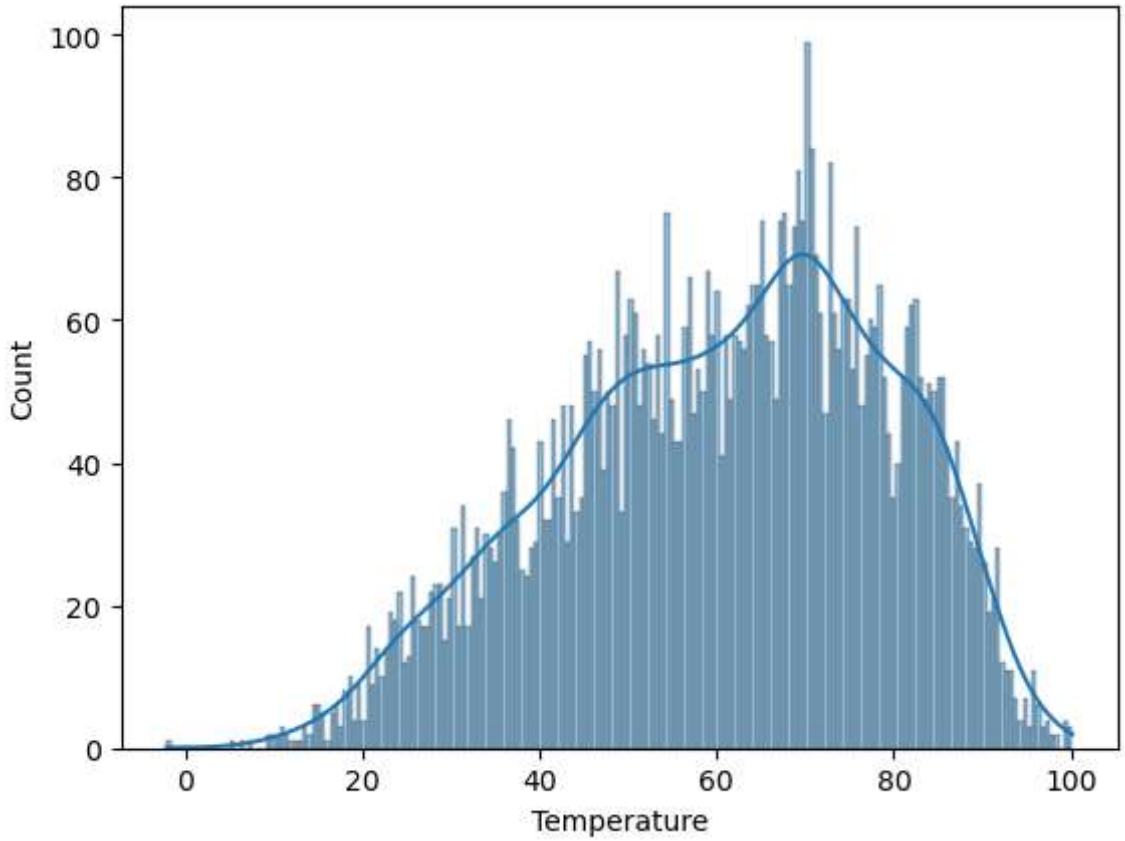
In [15]:

```
sns.histplot(data=data, kde=True, x='Weekly_Sales', bins=112)
plt.show()
```



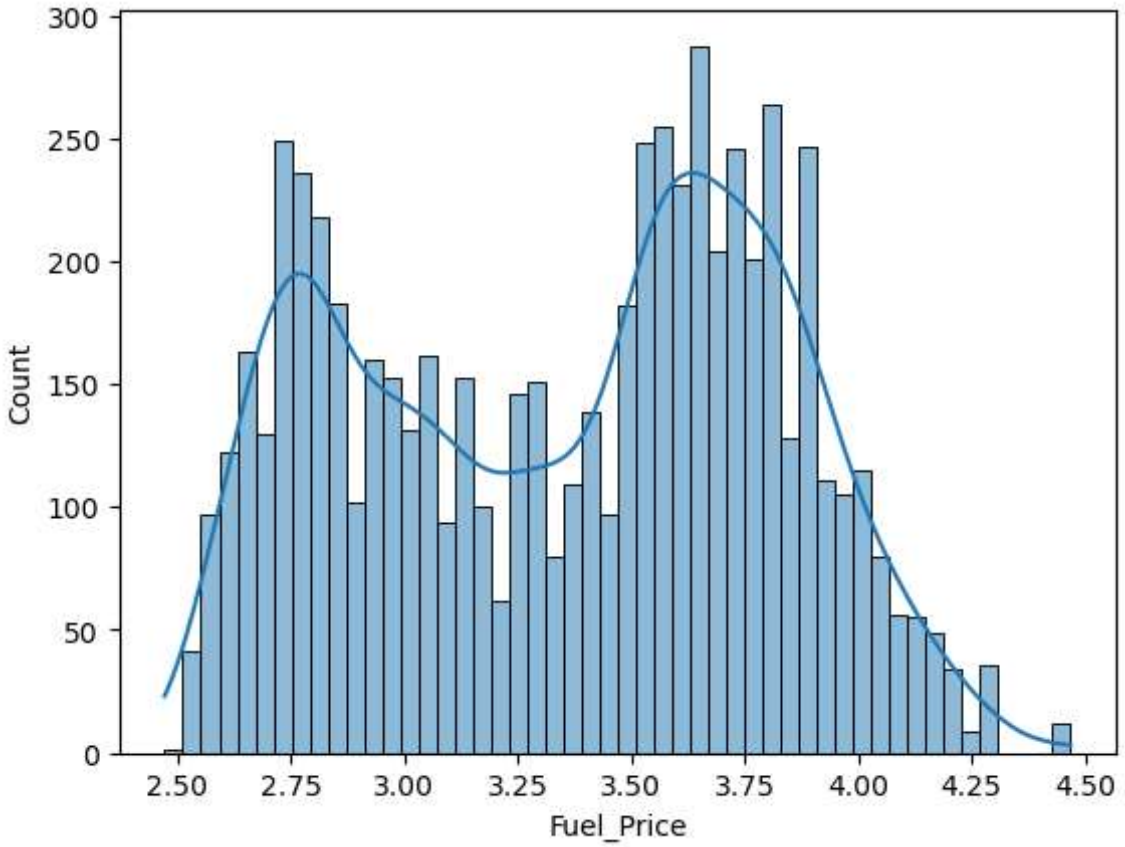
In [16]:

```
sns.histplot(data=data, kde=True, x='Temperature', bins=200)
plt.show()
```



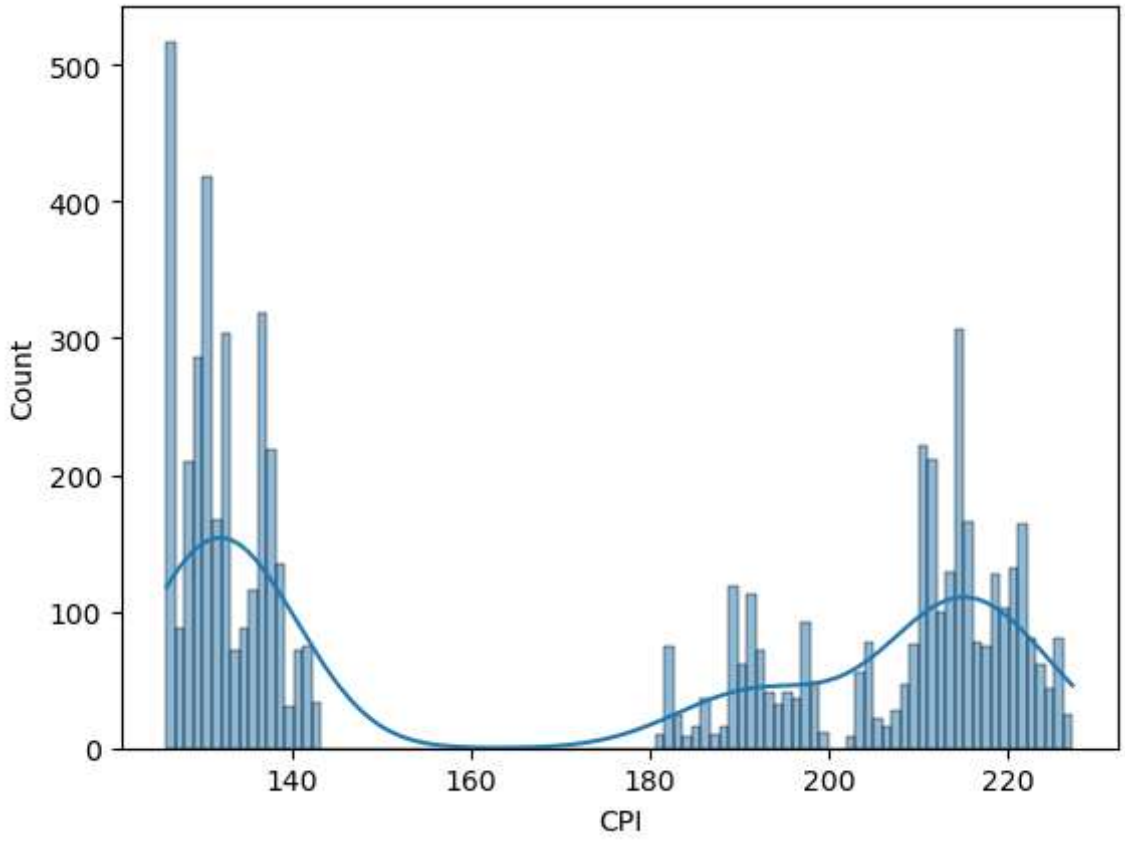
In [17]:

```
sns.histplot(data=data, kde=True, x='Fuel_Price', bins=50)
plt.show()
```



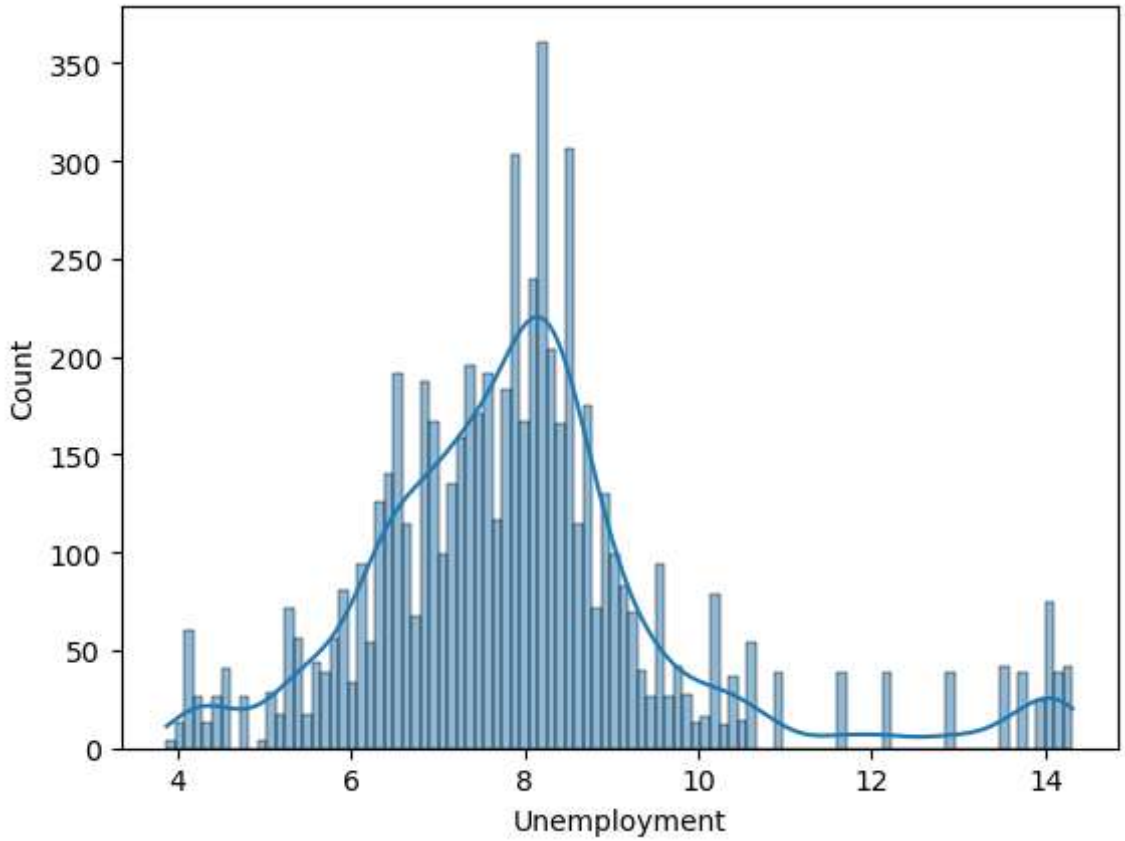
In [18]:

```
sns.histplot(data=data, kde=True, x='CPI', bins=100)
plt.show()
```



In [19]:

```
sns.histplot(data=data, kde=True, x='Unemployment', bins=100)
plt.show()
```



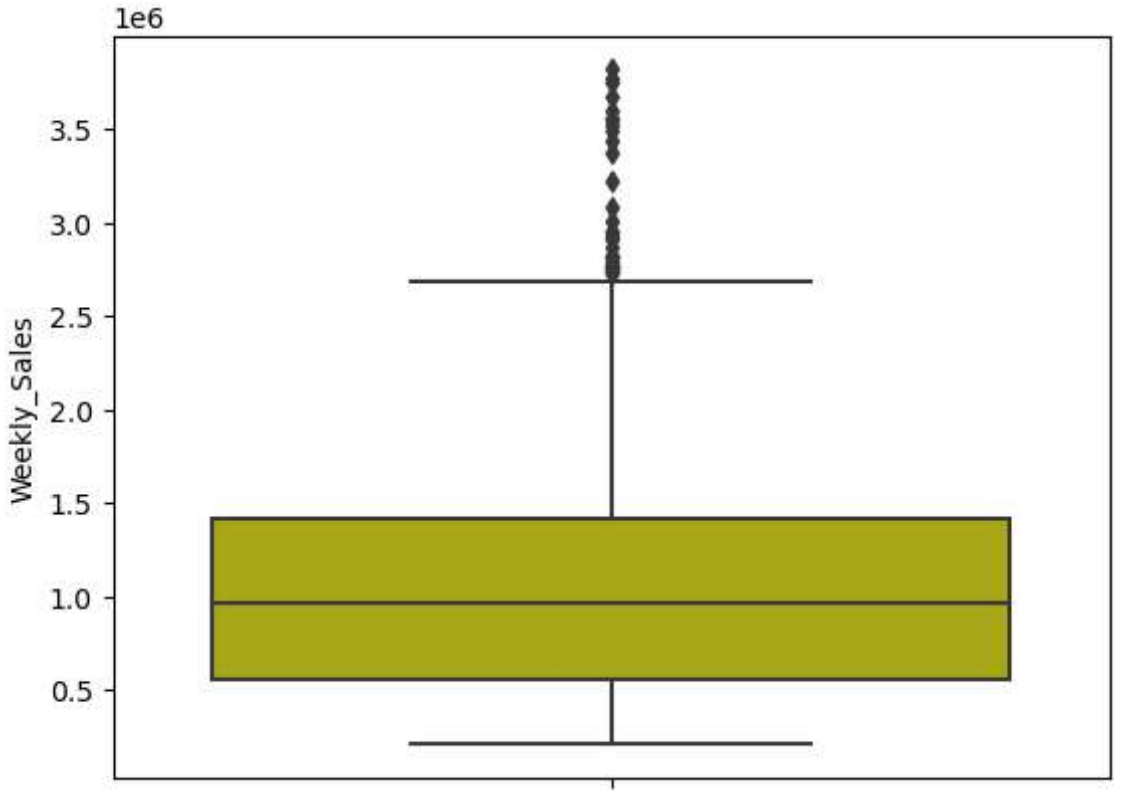
Box plots

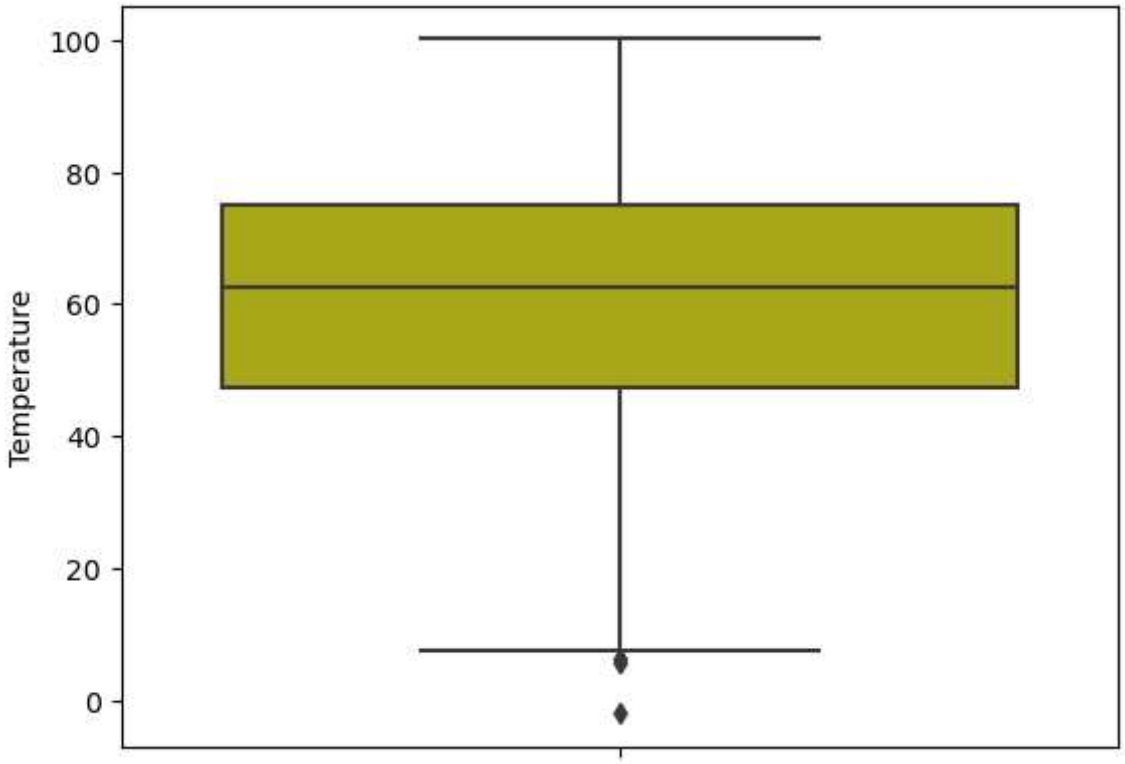
In [20]:

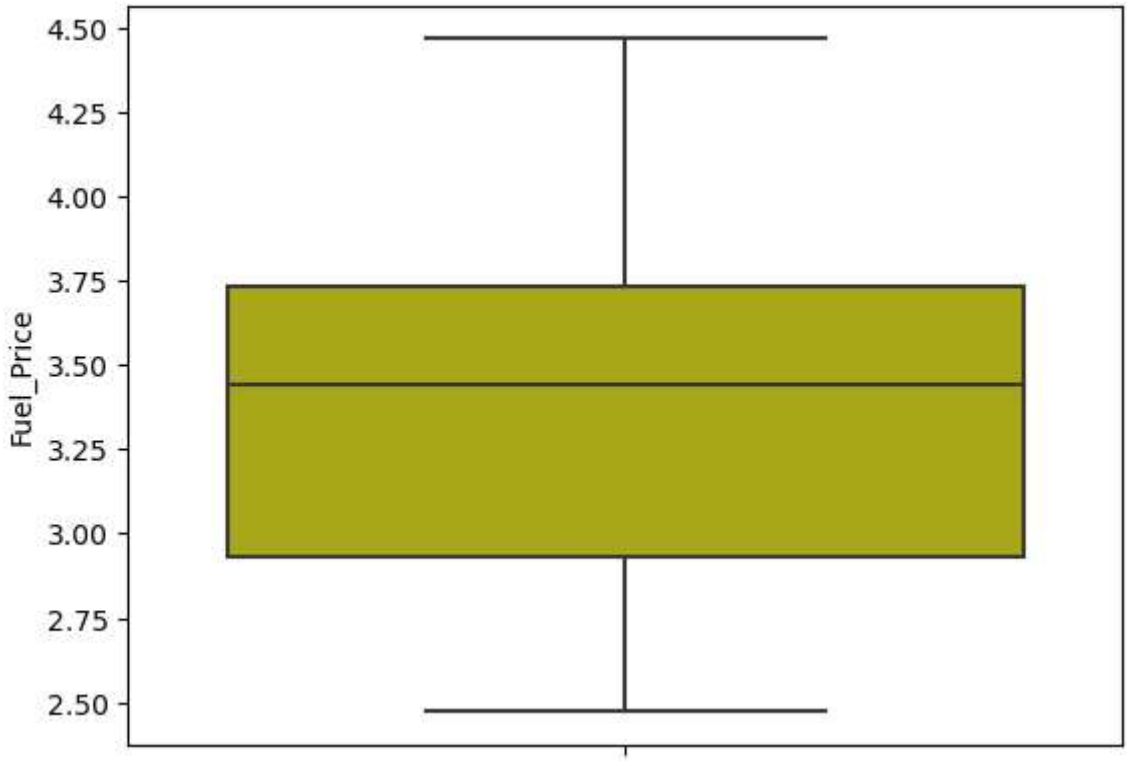
```
column_list = ['Weekly_Sales', 'Temperature',
       'Fuel_Price', 'CPI', 'Unemployment']
```

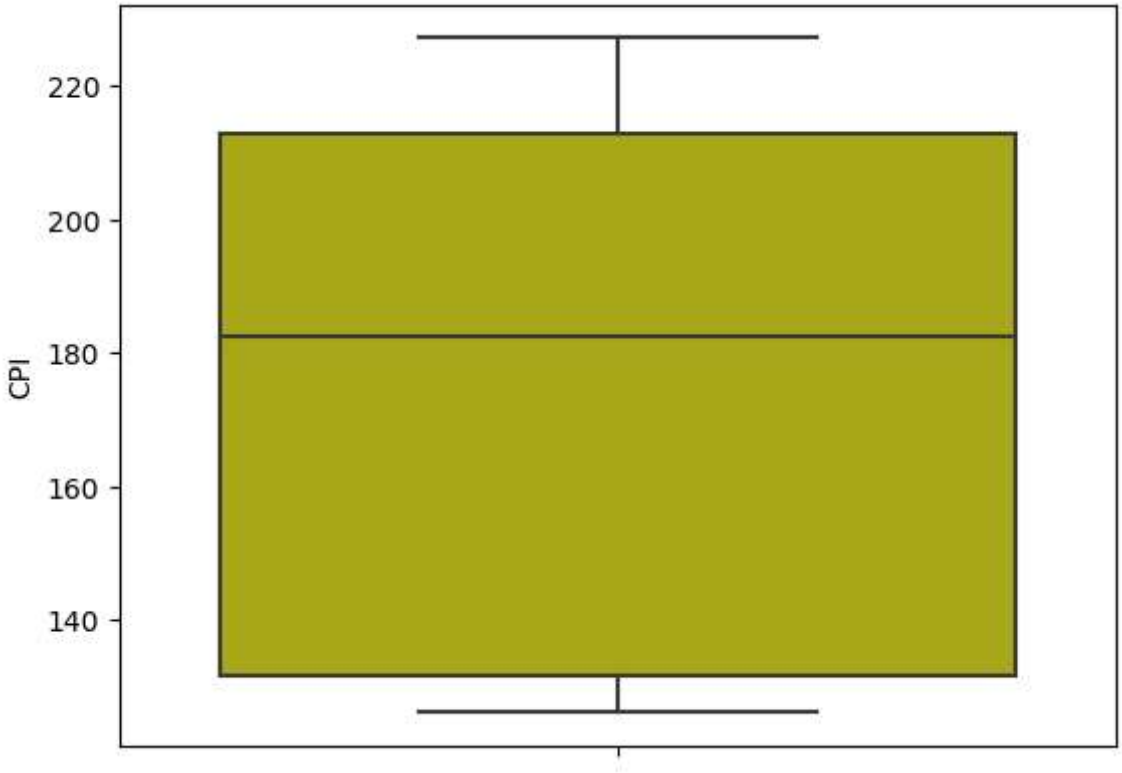
In [21]:

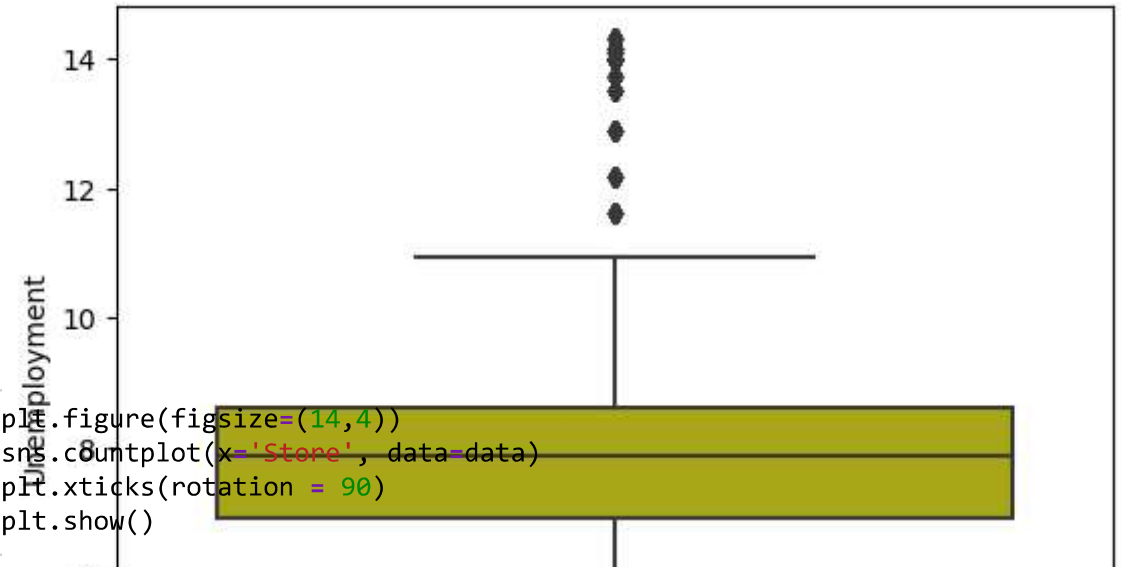
```
for i in column_list:
    sns.boxplot(y=data[i], data=data, color='y')
    plt.show()
```



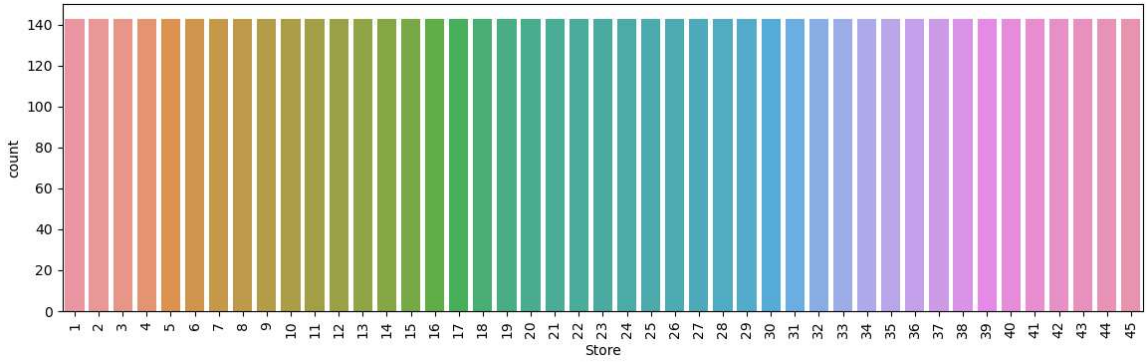



Fuel_Price
4.50
4.25
4.00
3.75
3.50
3.25
3.00
2.75
2.50


CPI
220
200
180
160
140

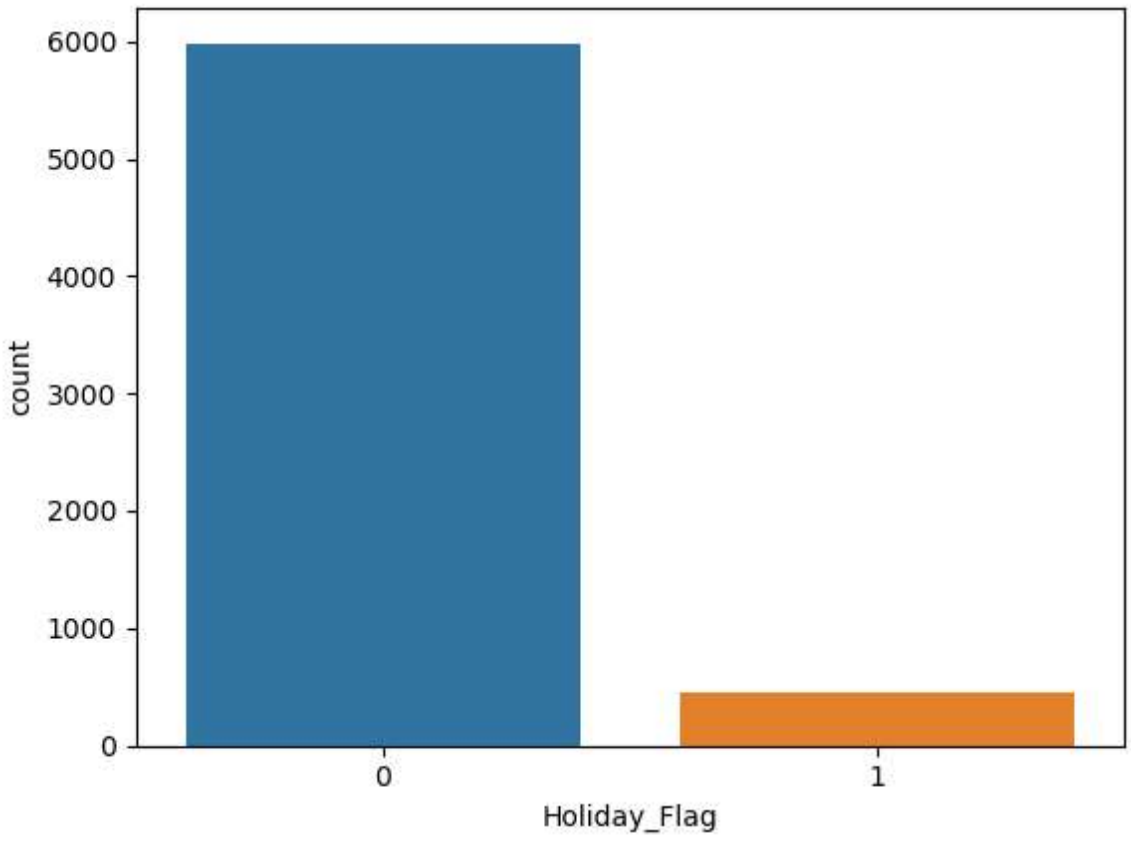



```
plt.figure(figsize=(14,4))
sns.countplot(x='Store', data=data)
plt.xticks(rotation = 90)
plt.show()
```



In [23]:

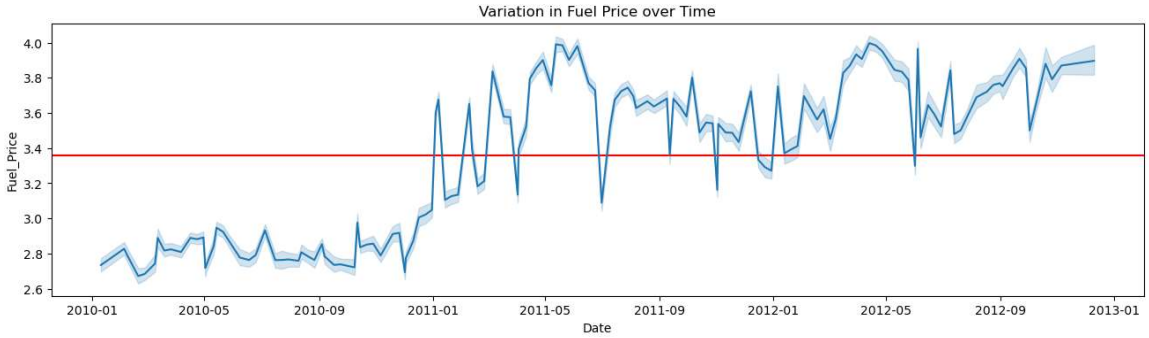
```
sns.countplot(x='Holiday_Flag', data=data)
plt.show()
```



Line Charts

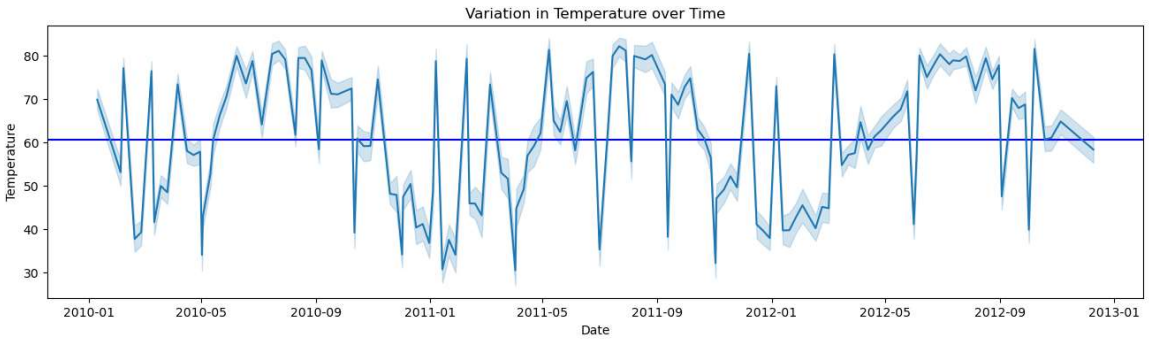
In [24]:

```
plt.figure(figsize=(16, 4))
plt.axhline(y=data.Fuel_Price.mean(), color = 'r')
sns.lineplot(x='Date', y='Fuel_Price', data=data)
plt.title("Variation in Fuel Price over Time")
plt.show()
```



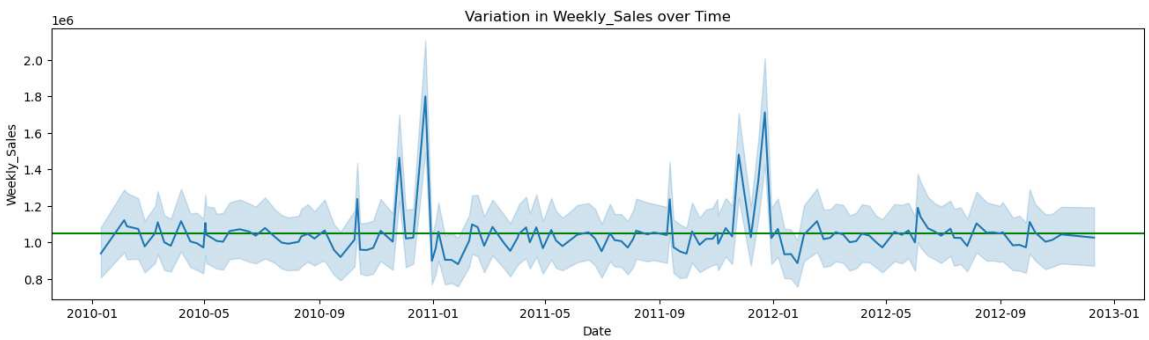
In [25]:

```
plt.figure(figsize=(16, 4))
plt.axhline(y=data.Temperature.mean(), color = 'b')
sns.lineplot(x='Date', y='Temperature', data=data)
plt.title("Variation in Temperature over Time")
plt.show()
```



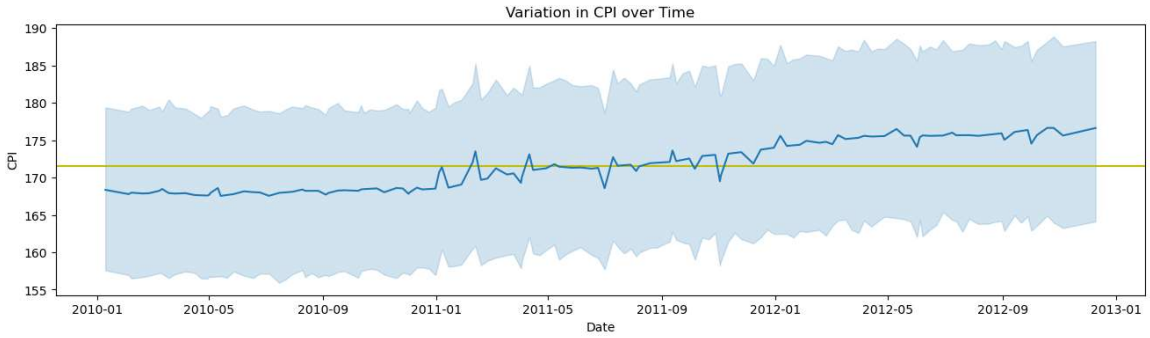
In [26]:

```
plt.figure(figsize=(16, 4))
plt.axhline(y=data.Weekly_Sales.mean(), color = 'g')
sns.lineplot(x='Date', y='Weekly_Sales', data=data)
plt.title("Variation in Weekly_Sales over Time")
plt.show()
```



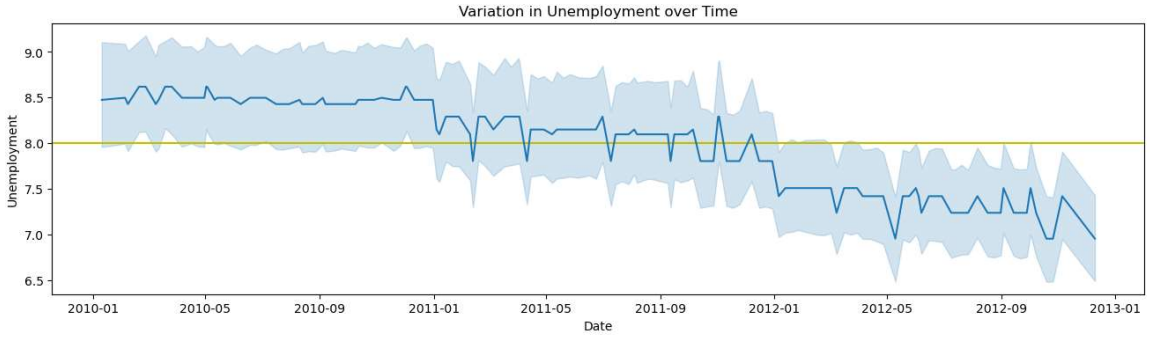
In [27]:

```
plt.figure(figsize=(16, 4))
plt.axhline(y=data.CPI.mean(), color = 'y')
sns.lineplot(x='Date', y='CPI', data=data)
plt.title("Variation in CPI over Time")
plt.show()
```



In [28]:

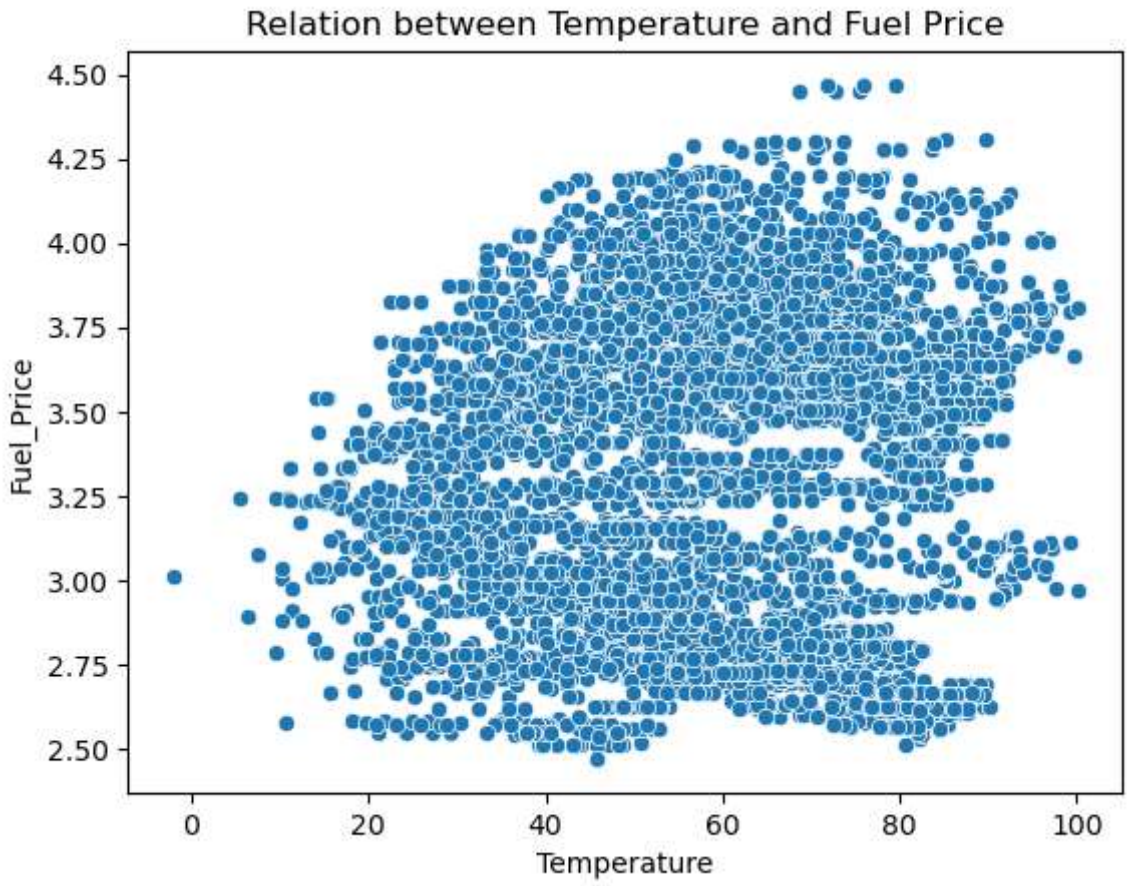
```
plt.figure(figsize=(16, 4))
plt.axhline(y=data.Unemployment.mean(), color = 'y')
sns.lineplot(x='Date', y='Unemployment', data=data)
plt.title("Variation in Unemployment over Time")
plt.show()
```



Scatter plots

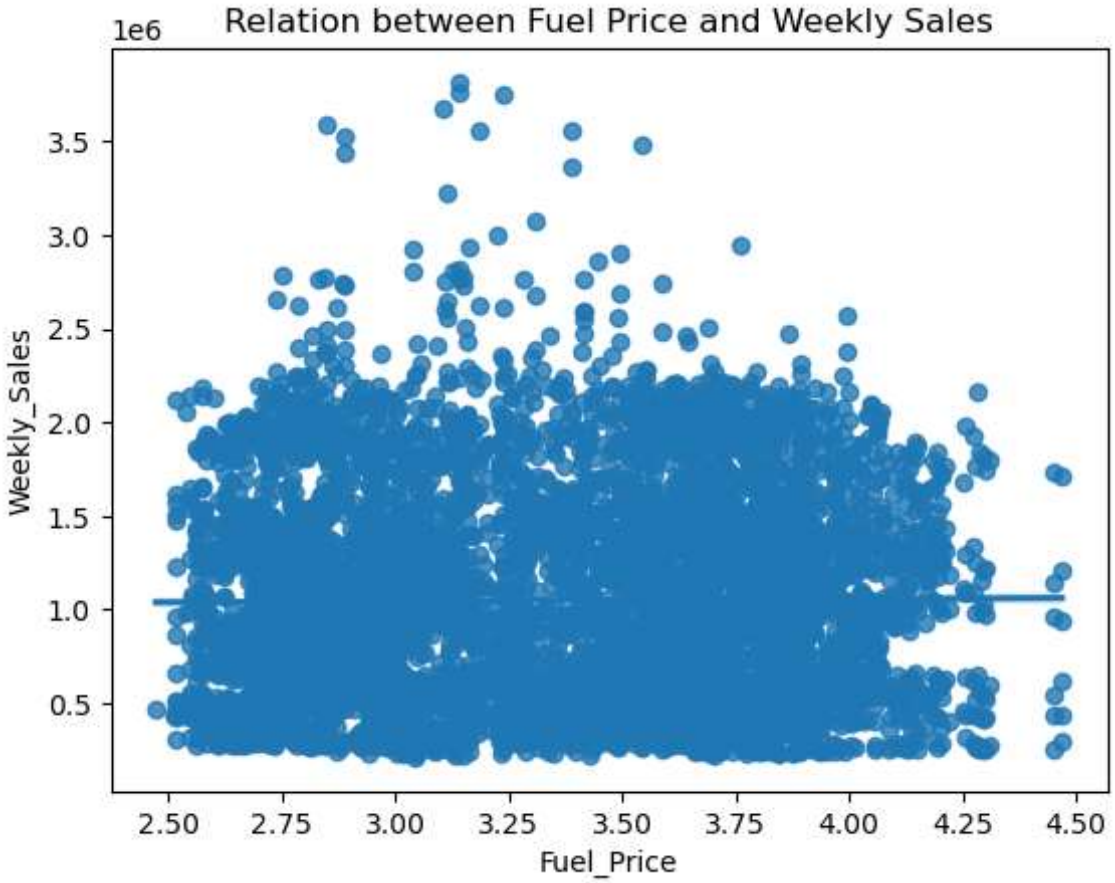
In [29]:

```
sns.scatterplot(x='Temperature', y='Fuel_Price', data=data)
plt.title("Relation between Temperature and Fuel Price")
plt.show()
```



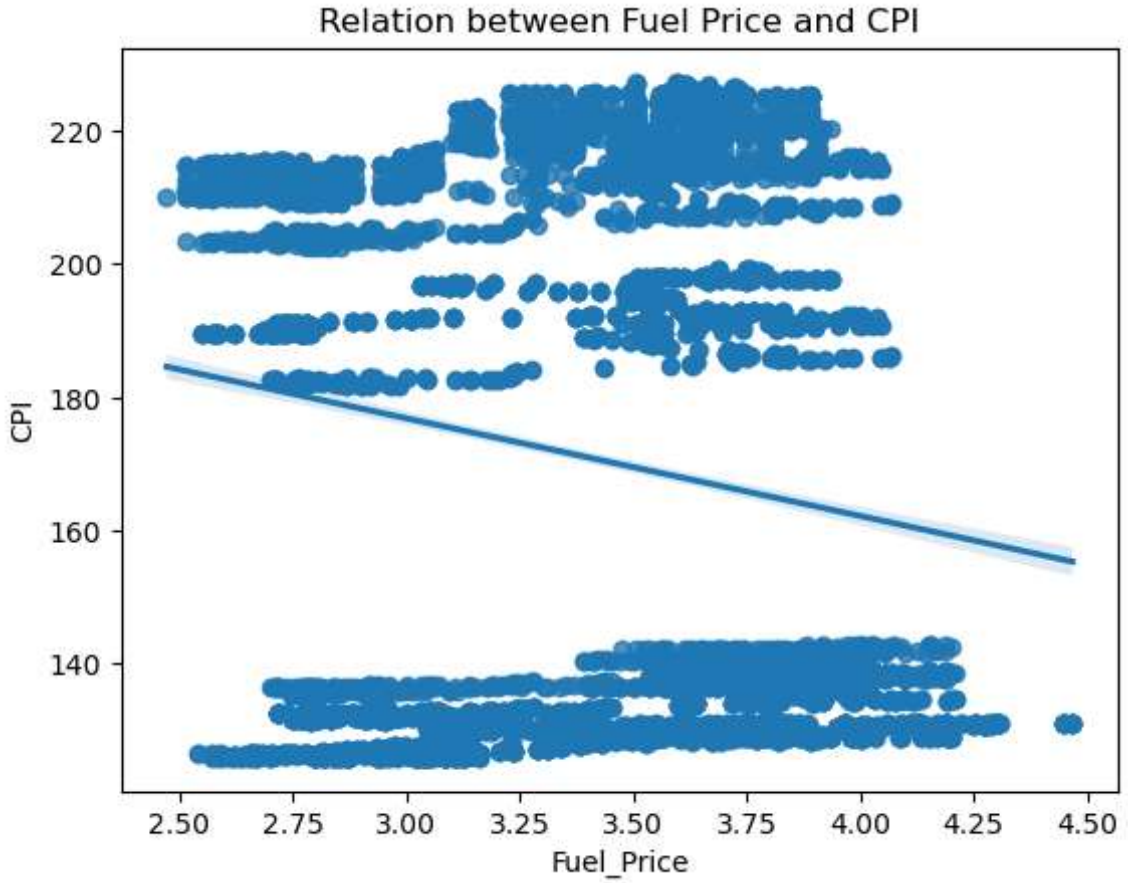
In [30]:

```
sns.regplot(x='Fuel_Price', y='Weekly_Sales',data=data)
plt.title("Relation between Fuel Price and Weekly Sales")
plt.show()
```



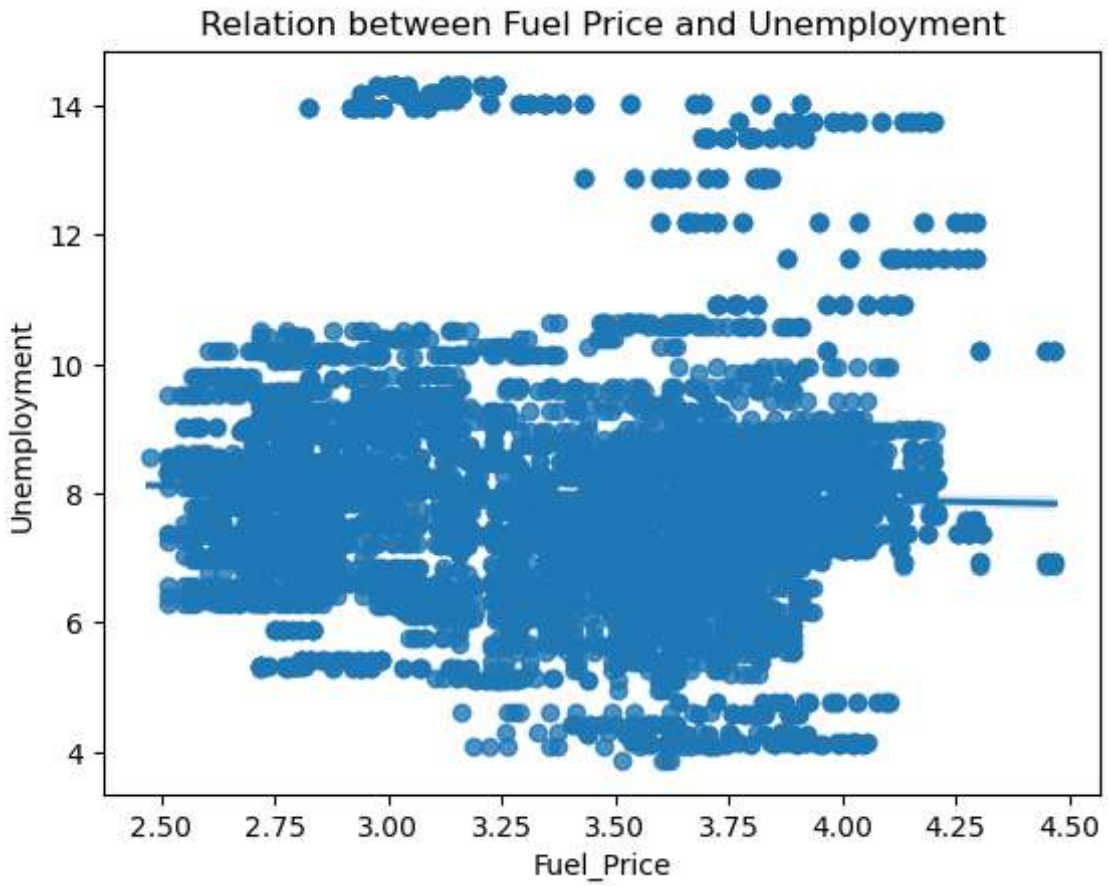
In [31]:

```
sns.regplot(x='Fuel_Price', y='CPI',data=data)
plt.title("Relation between Fuel Price and CPI")
plt.show()
```



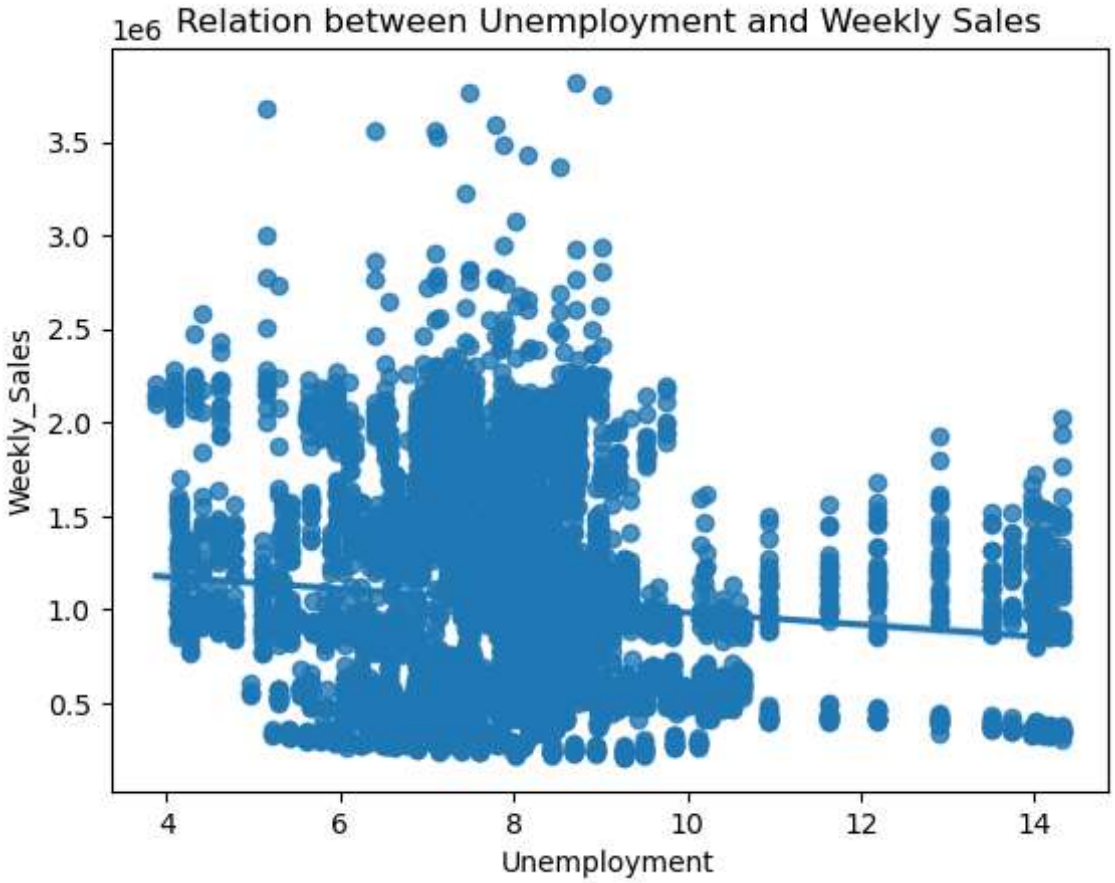
In [32]:

```
sns.regplot(x='Fuel_Price', y='Unemployment',data=data)
plt.title("Relation between Fuel Price and Unemployment")
plt.show()
```



In [33]:

```
sns.regplot(x='Unemployment',y='Weekly_Sales',data=data)
plt.title("Relation between Unemployment and Weekly Sales")
plt.show()
```



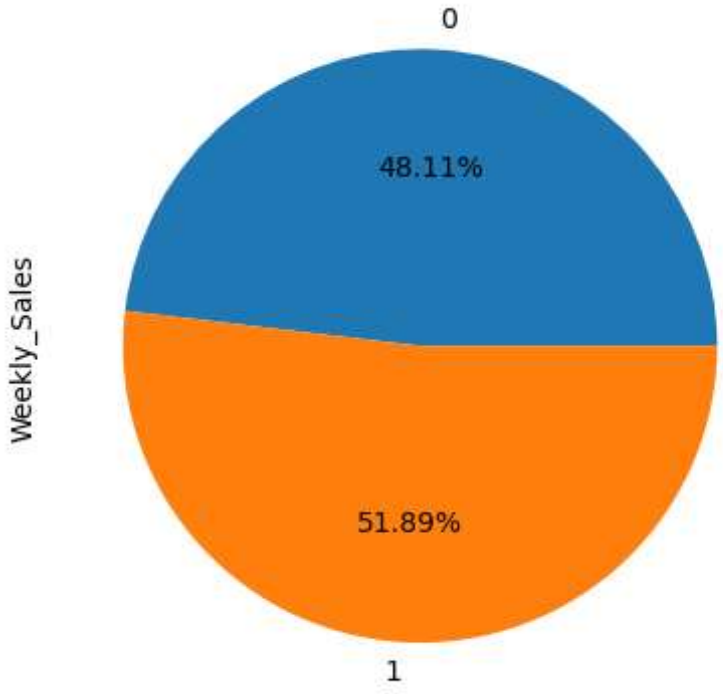
Pie Chart

In [34]:

```
Holiday_Sales = data.groupby(['Holiday_Flag'])['Weekly_Sales'].mean()
```

In [35]:

```
Holiday_Sales.plot.pie(autopct = '%1.2f%%');
```



Lets add new columns from the date column

In [36]:

```
data['year'] = data.Date.dt.year
data['month'] = data.Date.dt.month
data['week_day'] = data.Date.dt.weekday
```

In [37]:

```
data['month'] = data.month.replace({1:'January',
 2:'February',
 3:'March',
 4:'April',
 5:'May',
 6:'June',
7:'July',
 8:'August',
 9:'September',
10:'October',
11:'November',
 12:'December'})
data['week_day'] = data.week_day.replace({0:'Sunday',1:'Monday',2:'Tuesday',3:'Wednesday'
```

In [38]:

```
data
```

Out[38]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemp
0	5	2010-01-10	283178.12	0	71.10	2.603	212.226946	
1	15	2010-01-10	566945.95	0	59.69	2.840	132.756800	
2	42	2010-01-10	481523.93	0	86.01	3.001	126.234600	
3	33	2010-01-10	224294.39	0	91.45	3.001	126.234600	
4	36	2010-01-10	422169.47	0	74.66	2.567	210.440443	
...	...	...	...	...	...	...	...	
6430	41	2012-12-10	1409544.97	0	39.38	3.760	199.053937	
6431	16	2012-12-10	491817.19	0	43.26	3.760	199.053937	
6432	10	2012-12-10	1713889.11	0	76.03	4.468	131.108333	
6433	25	2012-12-10	697317.41	0	43.74	4.000	216.115057	
6434	2	2012-12-10	1900745.13	0	60.97	3.601	223.015426	

6435 rows × 11 columns

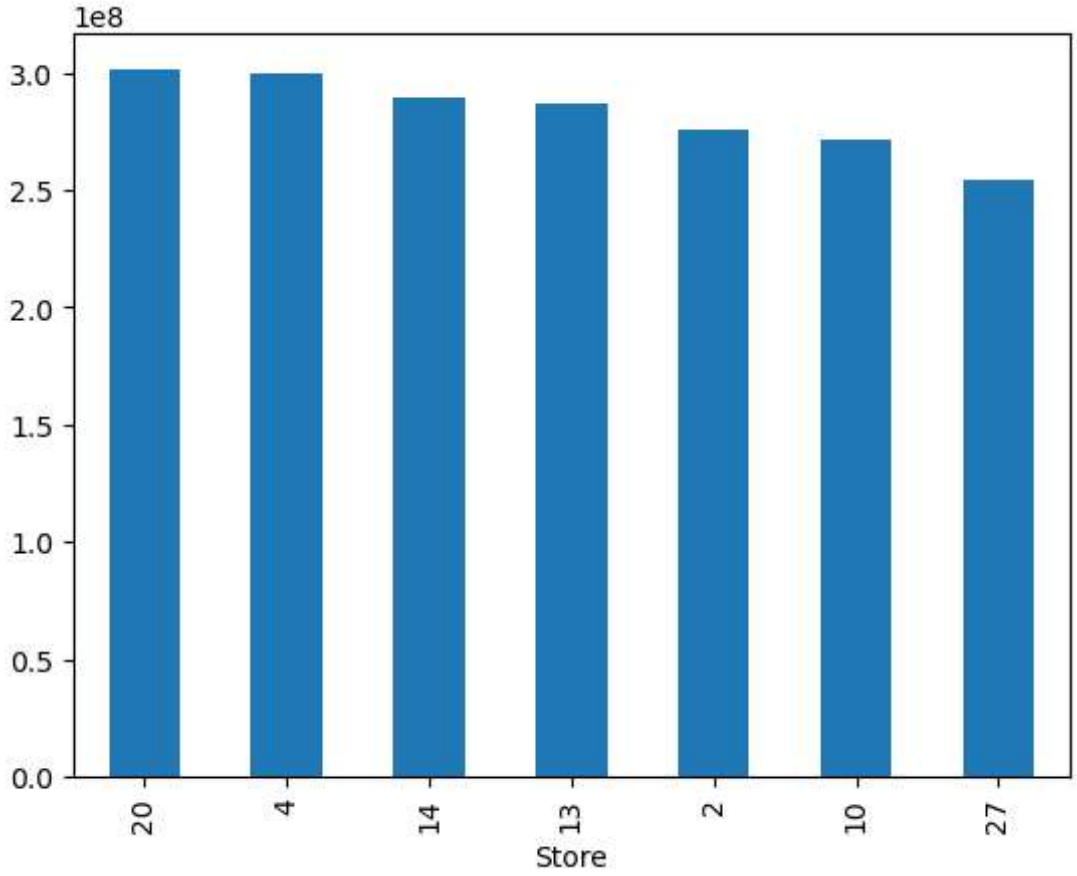
Bar plots

In [39]:

```
Store_Sales = data.groupby(['Store'])['Weekly_Sales'].sum()
```

In [40]:

```
Store_Sales.nlargest(7).plot.bar();
```

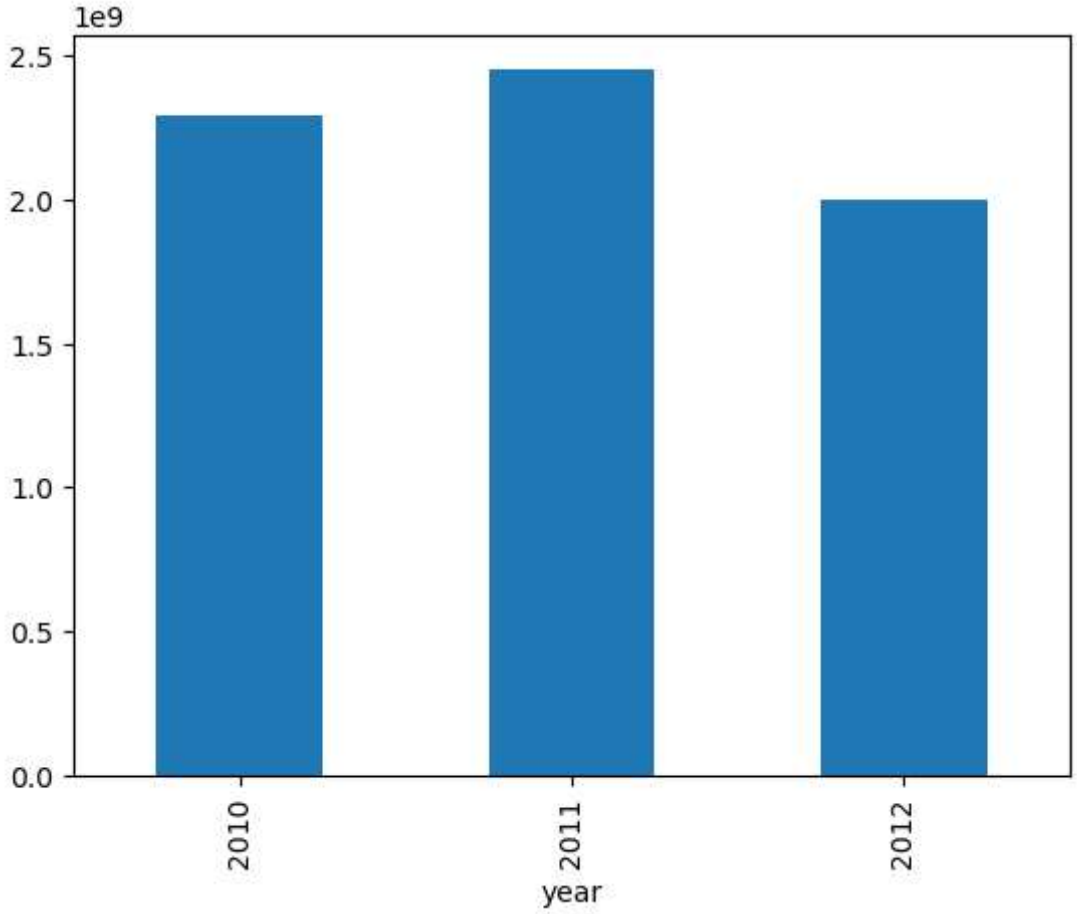


In [41]:

```
Year_Sales = data.groupby(['year'])['Weekly_Sales'].sum()
```

In [42]:

```
Year_Sales.plot.bar();
```

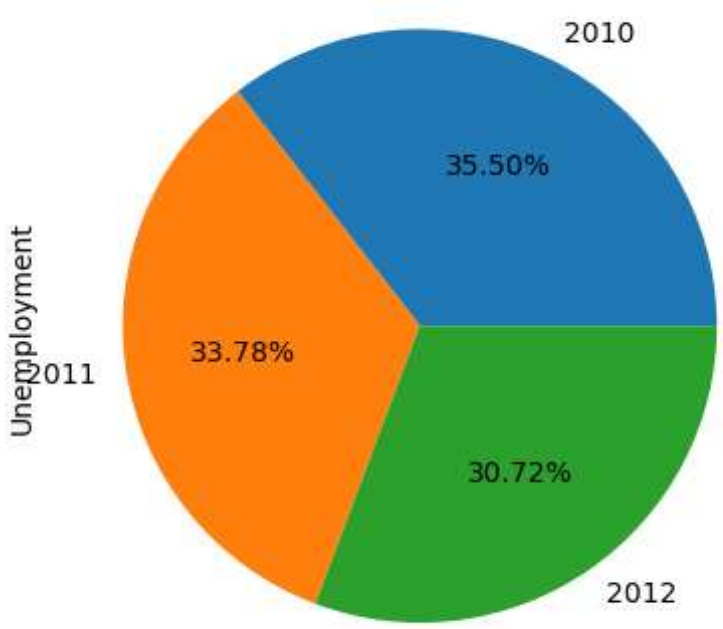


In [43]:

```
Year_unemployment = data.groupby(['year'])['Unemployment'].mean()
```

In [44]:

```
Year_unemployment.plot.pie(autopct = '%1.2f%%');
```

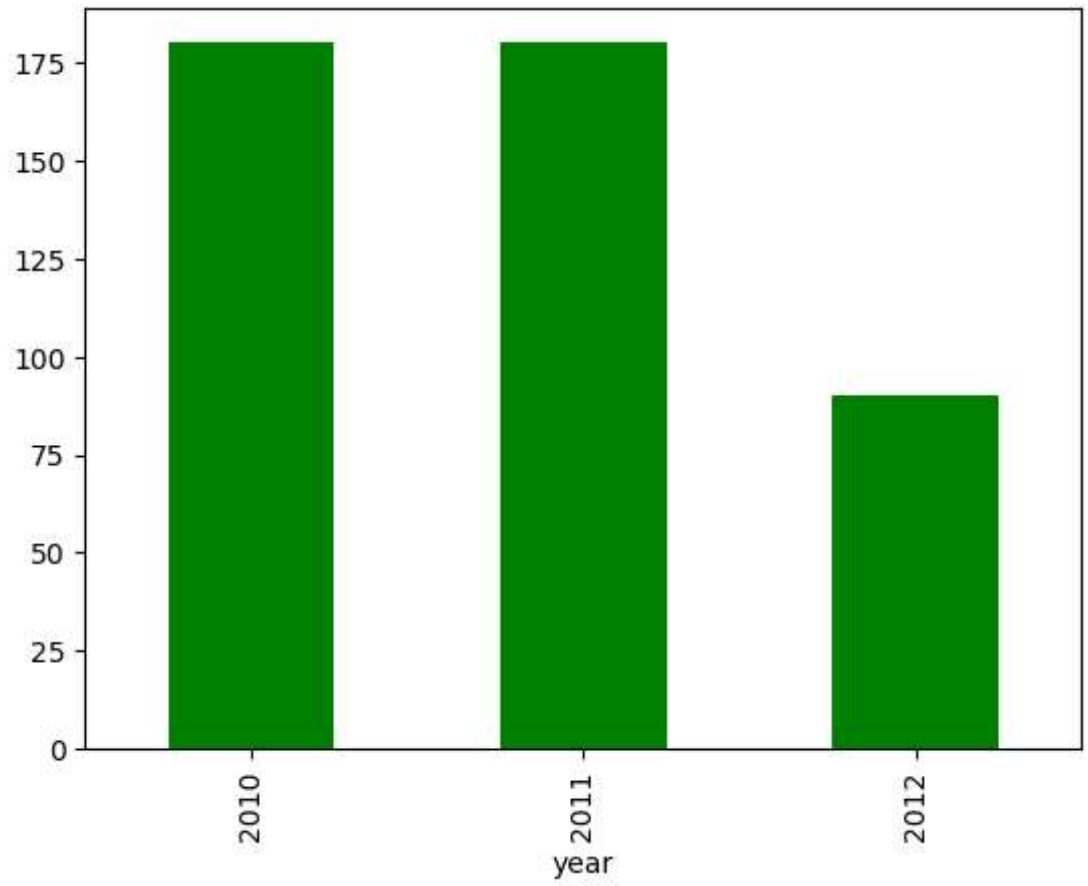


In [45]:

```
Year_Holiday = data.groupby(['year'])['Holiday_Flag'].sum()
```

In [46]:

```
Year_Holiday.plot.bar(color = 'g');
```

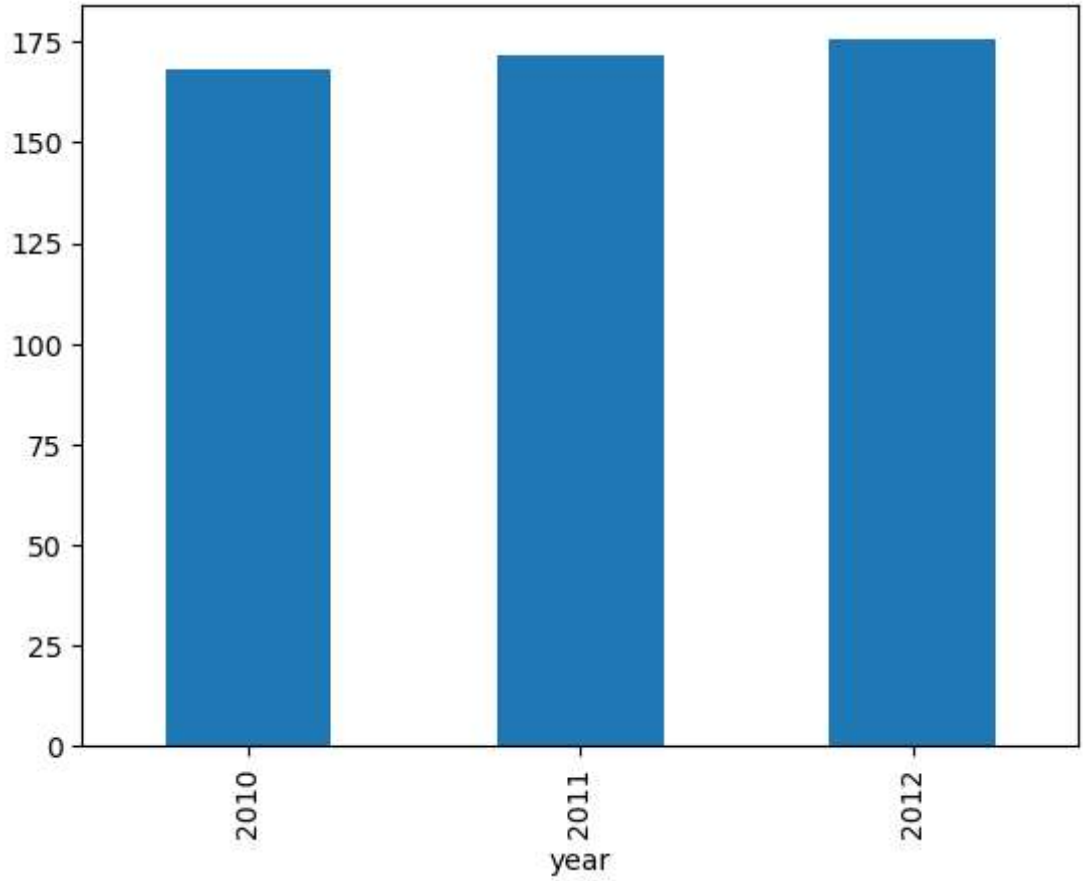


In [47]:

```
Year_CPI = data.groupby(['year'])['CPI'].mean()
```

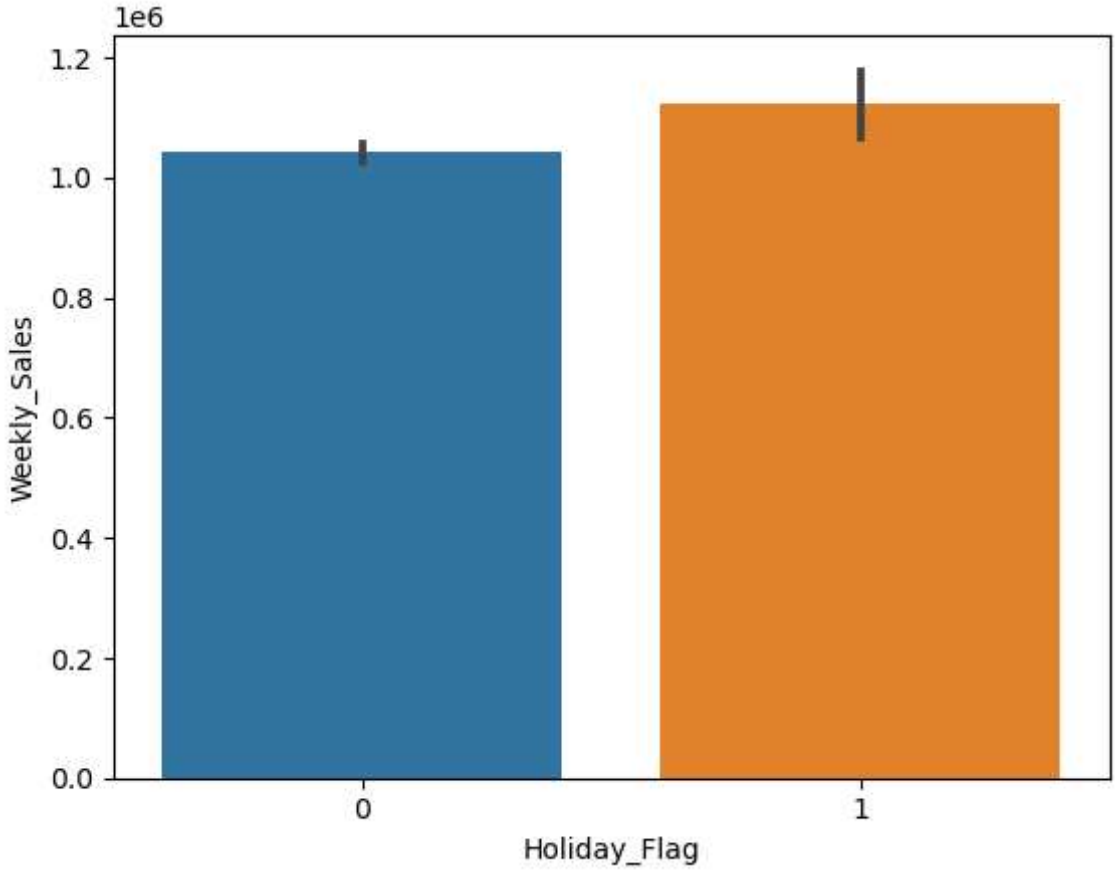
In [48]:

```
Year_CPI.plot.bar();
```



In [49]:

```
sns.barplot(x='Holiday_Flag', y='Weekly_Sales', data=data);
```



Year wise plots

In [50]:

```
Year_2010 = data[data.year == 2010]
Year_2011 = data[data.year == 2011]
Year_2012 = data[data.year == 2012]
```

In [51]:

```
Year_2010.drop(columns='year', inplace=True)
Year_2011.drop(columns='year', inplace=True)
Year_2012.drop(columns='year', inplace=True)
```

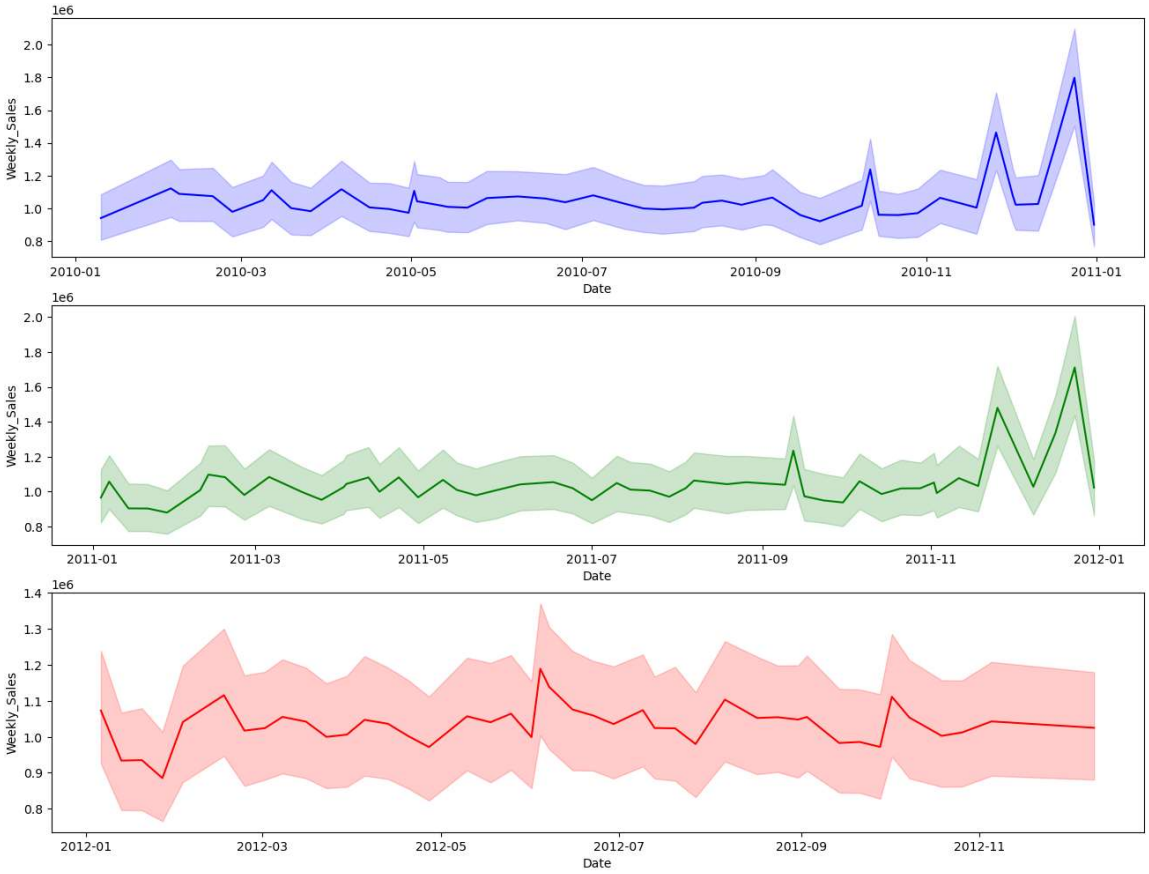
In [52]:

```
plt.figure(figsize=(16,12))
plt.subplot(3,1,1)
sns.lineplot(x='Date', y='Weekly_Sales', data=Year_2010, color = 'b')

plt.subplot(3,1,2)
sns.lineplot(x='Date', y='Weekly_Sales', data=Year_2011, color = 'g')

plt.subplot(3,1,3)
sns.lineplot(x='Date', y='Weekly_Sales', data=Year_2012, color = 'r')

plt.show()
```



Yearly Sales from each stores

In [53]:

```
Yearly_Sales_Store = data.groupby(['Store', 'year'])['Weekly_Sales'].sum()
```

In [54]:

```
Yearly_Sales_Store = Yearly_Sales_Store.reset_index()
```

In [55]:

```
Yearly_Sales_Store
```

Out[55]:

	Store	year	Weekly_Sales
0	1	2010	73278832.00
1	1	2011	80921918.83
2	1	2012	68202058.02
3	2	2010	95277864.19
4	2	2011	98607881.42
...	...	...	...
130	44	2011	15498194.67
131	44	2012	14187373.72
132	45	2010	38536343.37
133	45	2011	41135367.88
134	45	2012	32723630.17

135 rows × 3 columns

In [56]:

```
Yearly_Sales_Store.nlargest(5, 'Weekly_Sales')
```

Out[56]:

	Store	year	Weekly_Sales
10	4	2011	1.110923e+08
58	20	2011	1.098370e+08
40	14	2011	1.060963e+08
39	14	2010	1.054622e+08
37	13	2011	1.045375e+08

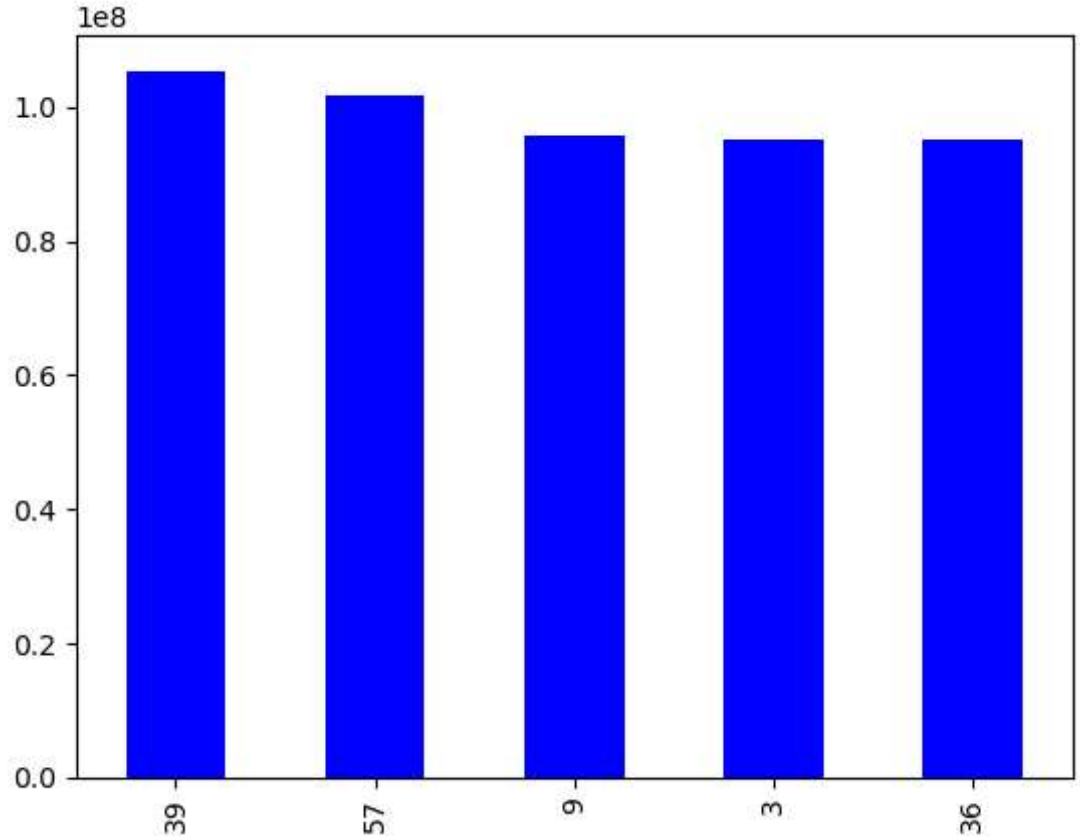
Top stores in each year

In [57]:

```
Yearly_Sales_Store_2010 = Yearly_Sales_Store[Yearly_Sales_Store.year == 2010]
Yearly_Sales_Store_2011 = Yearly_Sales_Store[Yearly_Sales_Store.year == 2011]
Yearly_Sales_Store_2012 = Yearly_Sales_Store[Yearly_Sales_Store.year == 2012]
```

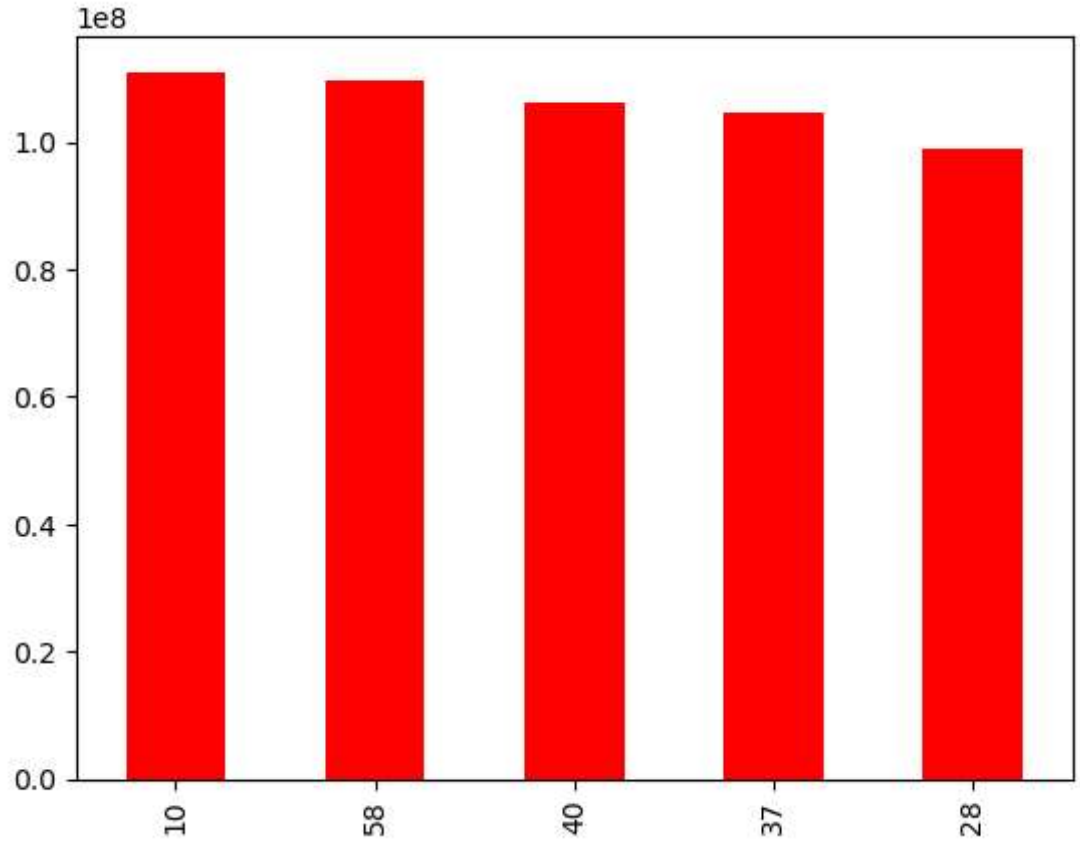
In [58]:

```
Yearly_Sales_Store_2010.Weekly_Sales.nlargest(5).plot.bar(color = 'b');
```



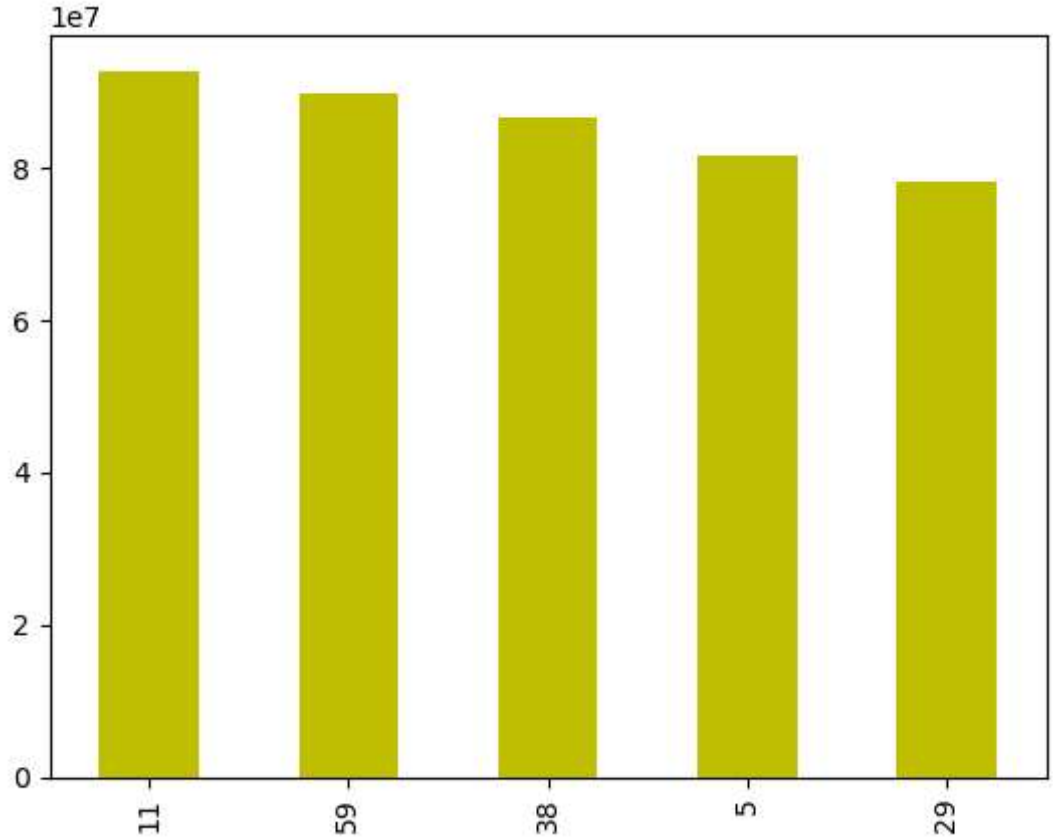
In [59]:

```
Yearly_Sales_Store_2011.Weekly_Sales.nlargest(5).plot.bar(color = 'r');
```



In [60]:

```
Yearly_Sales_Store_2012.Weekly_Sales.nlargest(5).plot.bar(color = 'y');
```



Temperature conversion (-2°F − 32) × 5/9

In [61]:

```
data['Celsius'] = [(Temp-32)*(5/9) for Temp in data.Temperature]
```

Lets create a new column called Heat

In [62]:

```
data['Heat'] = pd.cut(x=data.Celsius, bins=[-20, 10, 35], labels=['Cold', 'Warm'])
```

In [64]:

```
data
```

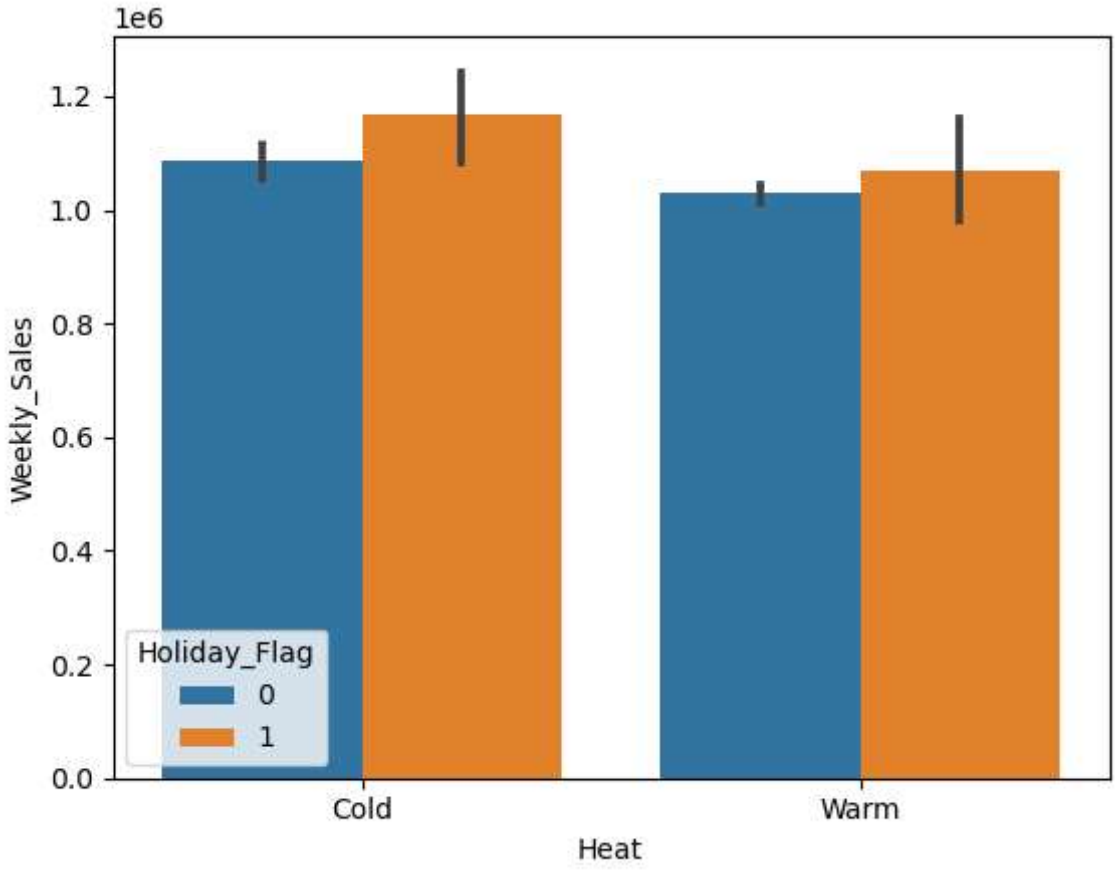
Out[64]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemp
0	5	2010-01-10	283178.12	0	71.10	2.603	212.226946	
1	15	2010-01-10	566945.95	0	59.69	2.840	132.756800	
2	42	2010-01-10	481523.93	0	86.01	3.001	126.234600	
3	33	2010-01-10	224294.39	0	91.45	3.001	126.234600	
4	36	2010-01-10	422169.47	0	74.66	2.567	210.440443	
...	...	...	...	...	...	...	...	
6430	41	2012-12-10	1409544.97	0	39.38	3.760	199.053937	
6431	16	2012-12-10	491817.19	0	43.26	3.760	199.053937	
6432	10	2012-12-10	1713889.11	0	76.03	4.468	131.108333	
6433	25	2012-12-10	697317.41	0	43.74	4.000	216.115057	
6434	2	2012-12-10	1900745.13	0	60.97	3.601	223.015426	

6435 rows × 13 columns

In [65]:

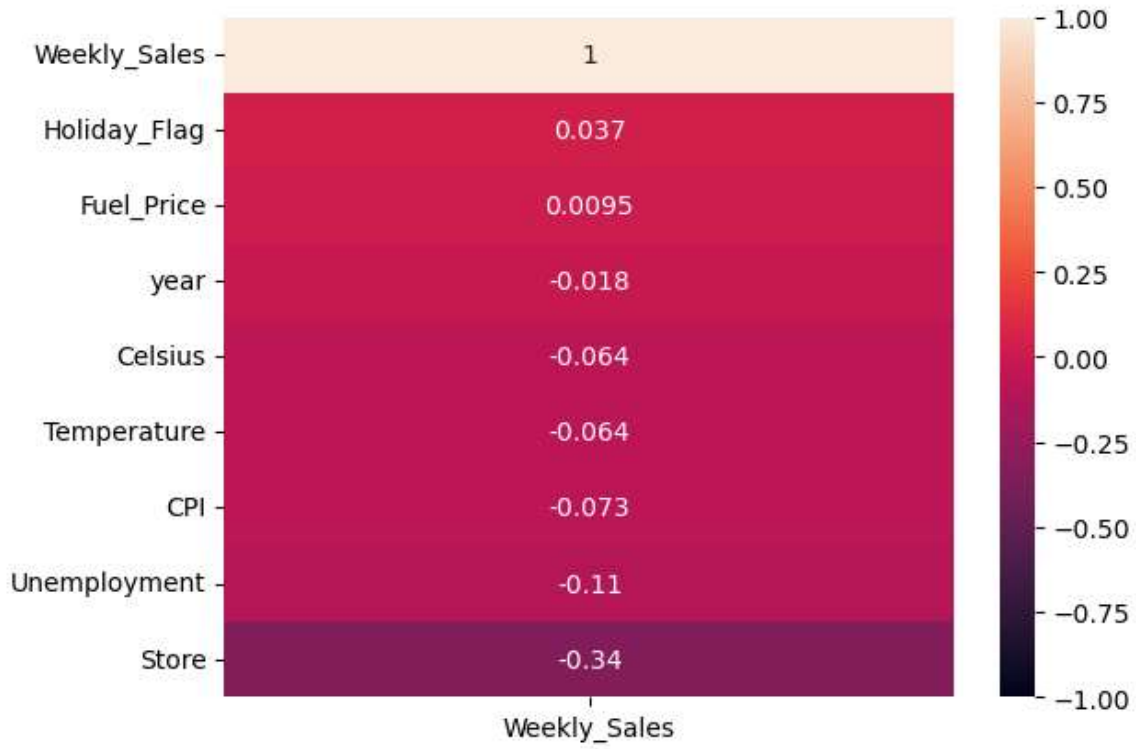
```
sns.barplot(x='Heat', y='Weekly_Sales', hue='Holiday_Flag', data=data)
plt.show()
```



Heat map

In [66]:

```
sns.heatmap(data.corr()[['Weekly_Sales']].sort_values(by='Weekly_Sales', ascending=False)
```



Plotting from pivot tables

In [67]:

```
Monthly_Sales = pd.pivot_table(data=data, values='Weekly_Sales',index='month',columns='ye
```

In [68]:

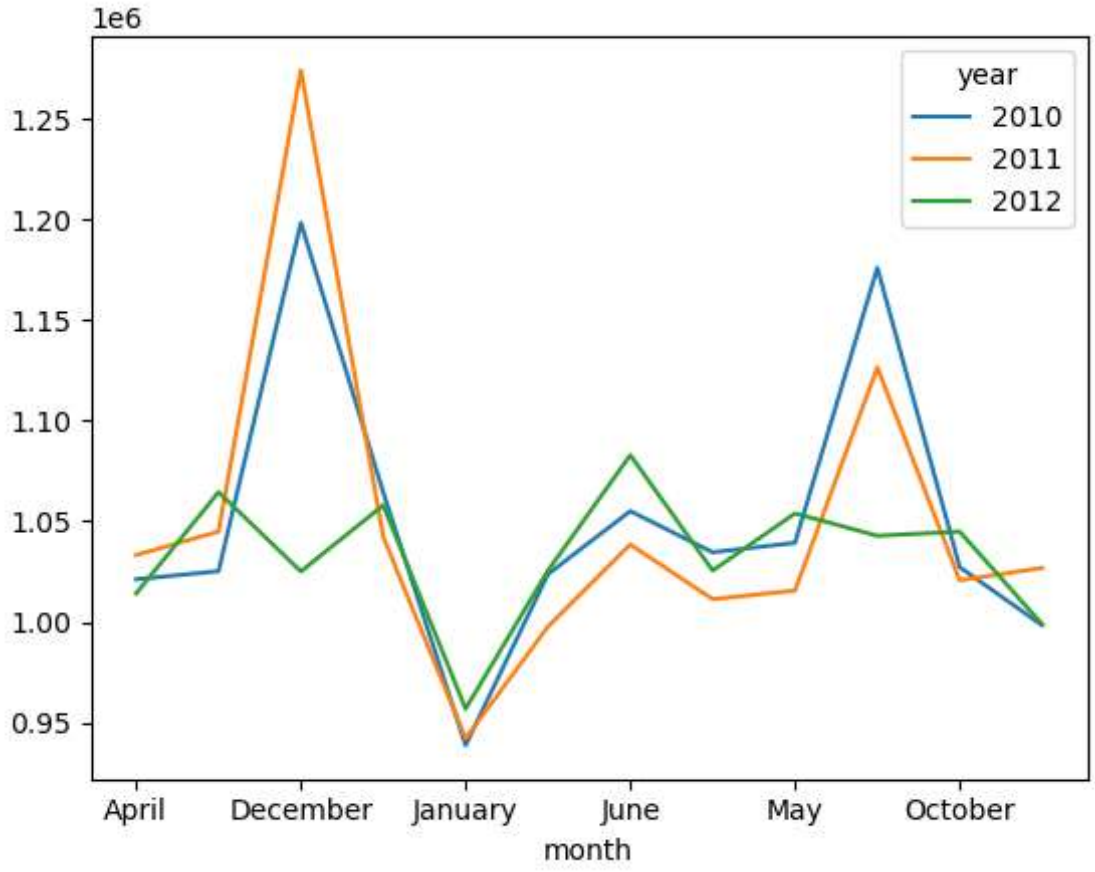
```
Monthly_Sales
```

Out[68]:

year	2010	2011	2012
month			
April	1.021177e+06	1.033220e+06	1.014127e+06
August	1.025212e+06	1.044895e+06	1.064514e+06
December	1.198413e+06	1.274311e+06	1.025078e+06
February	1.064372e+06	1.042273e+06	1.057997e+06
January	9.386639e+05	9.420697e+05	9.567817e+05
July	1.023702e+06	9.976049e+05	1.025480e+06
June	1.055082e+06	1.038471e+06	1.082920e+06
March	1.034590e+06	1.011263e+06	1.025510e+06
May	1.039303e+06	1.015565e+06	1.053948e+06
November	1.176097e+06	1.126535e+06	1.042797e+06
October	1.027201e+06	1.020663e+06	1.044885e+06
September	9.983559e+05	1.026810e+06	9.988663e+05

In [69]:

```
Monthly_Sales.plot();
```



*** Find total monthly and daily sales**

*** Plot monthly sales and other activities if suitable for each year**