Retail Analysis with Walmart Data

January 25, 2024

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import dates
from datetime import datetime
```

[2]:
```
walmart=pd.read_csv('Walmart_Store_sales.csv')
```

[3]:
```
walmart.head(10)
```

[3]:
```
   Store        Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
0      1  05-02-2010    1643690.90             0        42.31       2.572
1      1  12-02-2010    1641957.44             1        38.51       2.548
2      1  19-02-2010    1611968.17             0        39.93       2.514
3      1  26-02-2010    1409727.59             0        46.63       2.561
4      1  05-03-2010    1554806.68             0        46.50       2.625
5      1  12-03-2010    1439541.59             0        57.79       2.667
6      1  19-03-2010    1472515.79             0        54.58       2.720
7      1  26-03-2010    1404429.92             0        51.45       2.732
8      1  02-04-2010    1594968.28             0        62.27       2.719
9      1  09-04-2010    1545418.53             0        65.86       2.770

          CPI  Unemployment
0  211.096358         8.106
1  211.242170         8.106
2  211.289143         8.106
3  211.319643         8.106
4  211.350143         8.106
5  211.380643         8.106
6  211.215635         8.106
7  211.018042         8.106
8  210.820450         7.808
9  210.622857         7.808
```

[4]:
```
walmart.ndim
```

```
[4]: 2
```

```
[5]: type(walmart)
```

```
[5]: pandas.core.frame.DataFrame
```

```
[6]: walmart.shape
```

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

```
[7]: walmart.size
```

```
[7]: 51480
```

```
[8]: walmart.dtypes
```

```
[8]: Store              int64
     Date              object
     Weekly_Sales     float64
     Holiday_Flag       int64
     Temperature      float64
     Fuel_Price       float64
     CPI              float64
     Unemployment     float64
     dtype: object
```

```
[9]: walmart.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
```

```
[10]: walmart=pd.read_csv('Walmart_Store_sales.csv')
```

```
[11]: walmart.head(10)
```

[11]:

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

          CPI  Unemployment
0  211.096358         8.106
1  211.242170         8.106
2  211.289143         8.106
3  211.319643         8.106
4  211.350143         8.106
5  211.380643         8.106
6  211.215635         8.106
7  211.018042         8.106
8  210.820450         7.808
9  210.622857         7.808
```

[12]:

```python
walmart.loc[9,'Temperature']
```

[12]:

```
65.86
```

[13]:

```python
#Checking for missing values
walmart.isnull().sum()
```

[13]:

```
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
dtype: int64
```

[14]:

```python
walmart
```

[14]:

```
      Store        Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
0         1  05-02-2010    1643690.90             0        42.31       2.572
1         1  12-02-2010    1641957.44             1        38.51       2.548
2         1  19-02-2010    1611968.17             0        39.93       2.514
```

```
3        1  26-02-2010   1409727.59            0        46.63       2.561
4        1  05-03-2010   1554806.68            0        46.50       2.625
…      …        …            …         …         …        …
6430    45  28-09-2012    713173.95            0        64.88       3.997
6431    45  05-10-2012    733455.07            0        64.89       3.985
6432    45  12-10-2012    734464.36            0        54.47       4.000
6433    45  19-10-2012    718125.53            0        56.47       3.969
6434    45  26-10-2012    760281.43            0        58.85       3.882

             CPI  Unemployment
0     211.096358         8.106
1     211.242170         8.106
2     211.289143         8.106
3     211.319643         8.106
4     211.350143         8.106
…            …           …
6430  192.013558         8.684
6431  192.170412         8.667
6432  192.327265         8.667
6433  192.330854         8.667
6434  192.308899         8.667

[6435 rows x 8 columns]
```

[15]:

```
# # Convert date to datetime format and show dataset information
walmart['Date']=pd.to_datetime(walmart['Date'])
walmart.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
```

```
/tmp/ipykernel_76/4019288052.py:2: UserWarning: Parsing dates in DD/MM/YYYY
format when dayfirst=False (the default) was specified. This may lead to
inconsistently parsed dates! Specify a format to ensure consistent parsing.
  walmart['Date']=pd.to_datetime(walmart['Date'])
```

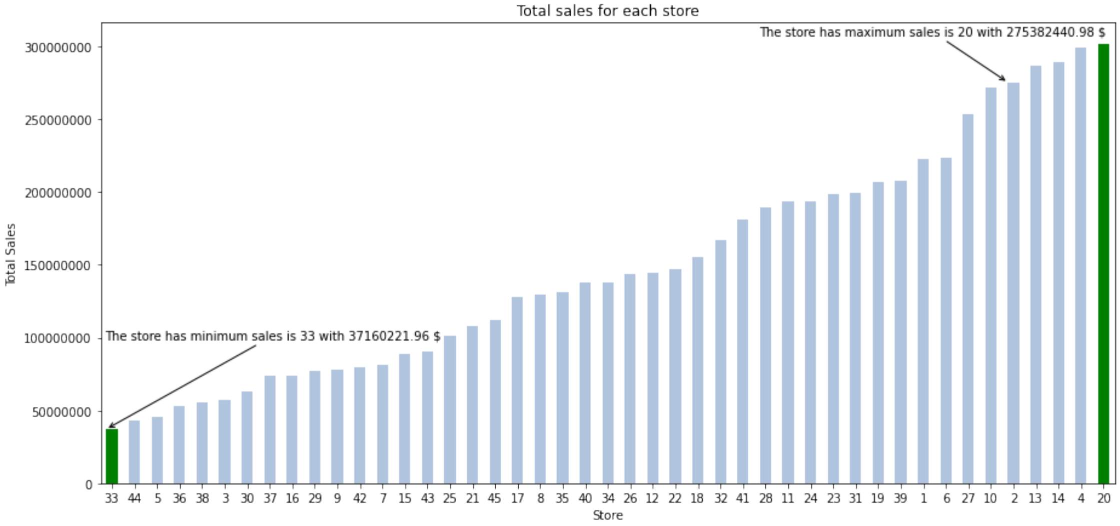
[16]:
```
# Splitting date and create new columns
walmart['Day']=pd.DatetimeIndex(walmart['Date']).day
walmart['Month']=pd.DatetimeIndex(walmart['Date']).month
walmart['Year']=pd.DatetimeIndex(walmart['Date']).year
walmart
```

[16]:
```
      Store        Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
0         1  2010-05-02    1643690.90             0        42.31       2.572
1         1  2010-12-02    1641957.44             1        38.51       2.548
2         1  2010-02-19    1611968.17             0        39.93       2.514
3         1  2010-02-26    1409727.59             0        46.63       2.561
4         1  2010-05-03    1554806.68             0        46.50       2.625
…       …         …            …             …          …          …
6430     45  2012-09-28     713173.95             0        64.88       3.997
6431     45  2012-05-10     733455.07             0        64.89       3.985
6432     45  2012-12-10     734464.36             0        54.47       4.000
6433     45  2012-10-19     718125.53             0        56.47       3.969
6434     45  2012-10-26     760281.43             0        58.85       3.882

             CPI  Unemployment  Day  Month  Year
0     211.096358         8.106    2      5  2010
1     211.242170         8.106    2     12  2010
2     211.289143         8.106   19      2  2010
3     211.319643         8.106   26      2  2010
4     211.350143         8.106    3      5  2010
…          …            …   …    …    …
6430  192.013558         8.684   28      9  2012
6431  192.170412         8.667   10      5  2012
6432  192.327265         8.667   10     12  2012
6433  192.330854         8.667   19     10  2012
6434  192.308899         8.667   26     10  2012

[6435 rows x 11 columns]
```

[17]:
```
# WHICH STORE HAS MAXIMUM SALES?

plt.figure(figsize=(15,7))
total_sales=walmart.groupby('Store')['Weekly_Sales'].sum().sort_values()
total_sales_array=np.array(total_sales)

clr=['lightsteelblue'if((x < max(total_sales))and(x > min(total_sales_array)))
         else'green'for x in total_sales_array]

graph=total_sales.plot(kind='bar', color=clr);

# Store with minimum sales
a=graph.patches[0]
```

```
print(type(a.get_height()))
graph.annotate("The store has minimum sales is 33 with {0:.2f} $".format((a.
 ↪get_height())), xy=(a.get_x(), a.get_height()), xycoords='data',
               xytext=(0.17, 0.32), textcoords='axes fraction',
            arrowprops=dict(arrowstyle="->", connectionstyle="arc3"),
            horizontalalignment='center', verticalalignment='center')

# Store with maximum sales

a=graph.patches[40]
graph.annotate("The store has maximum sales is 20 with {0:.2f} $".format((a.
 ↪get_height())), xy=(a.get_x(), a.get_height()), xycoords='data',
               xytext=(0.82, 0.98), textcoords='axes fraction',
            arrowprops=dict(arrowstyle="->", connectionstyle="arc3"),
            horizontalalignment='center', verticalalignment='center')
# Plotting properties

plt.xticks(rotation=0)
plt.ticklabel_format(useOffset=False, style='plain', axis='y')
plt.title('Total sales for each store')
plt.xlabel('Store')
plt.ylabel('Total Sales');
```

```
<class 'numpy.float64'>
```



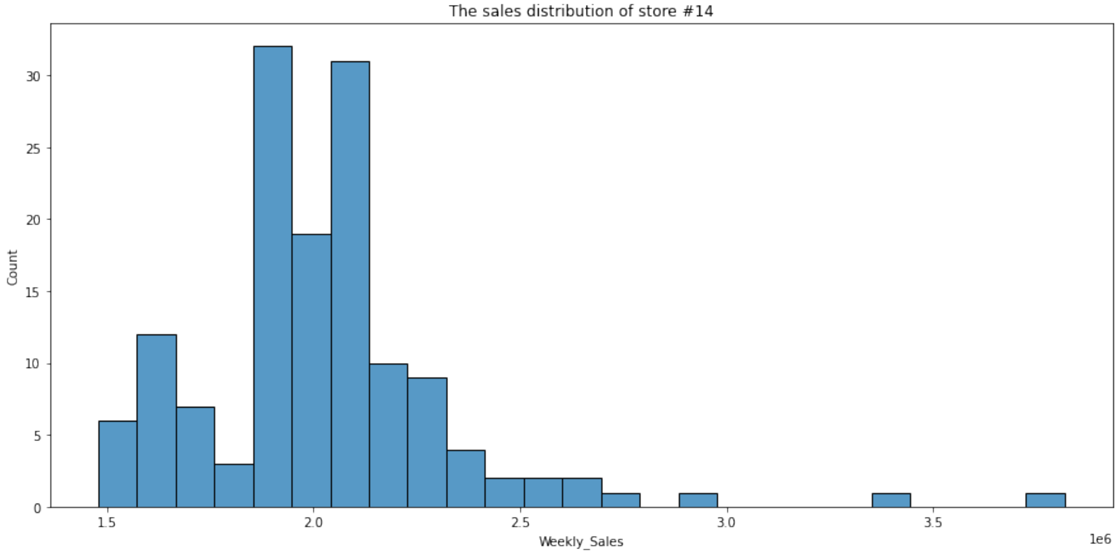
[18]:

```
# Which store has maximum standard deviation i.e., the sales vary a lot. Also,␣
 ↪find out the coefficient of mean to standard deviation ??
```

```
data_std=pd.DataFrame(walmart.groupby('Store')['Weekly_Sales'].std().
 ↪sort_values(ascending=False))
print("The store has maximum standard deviation is "+str(data_std.head(1).
 ↪index[0])+" with {0:.0f} $".format(data_std.head(1).Weekly_Sales[data_std.
 ↪head(1).index[0]]))
```

```
The store has maximum standard deviation is 14 with 317570 $
```

[19]:
```
# Distribution of store has maximum standard deviation


plt.figure(figsize=(15,7))
sns.histplot(walmart[walmart['Store']==data_std.head(1).
 ↪index[0]]['Weekly_Sales'])
plt.title('The sales distribution of store #'+str(data_std.head(1).index[0]));
```



[20]:
```
# Coefficient of mean to standard deviation

coef_mean_std = pd.DataFrame(walmart.groupby('Store')['Weekly_Sales'].std() /␣
 ↪walmart.groupby('Store')['Weekly_Sales'].mean())
coef_mean_std = coef_mean_std.rename(columns={'Weekly_Sales':'Coefficient of␣
 ↪mean to standard deviation'})
coef_mean_std
```

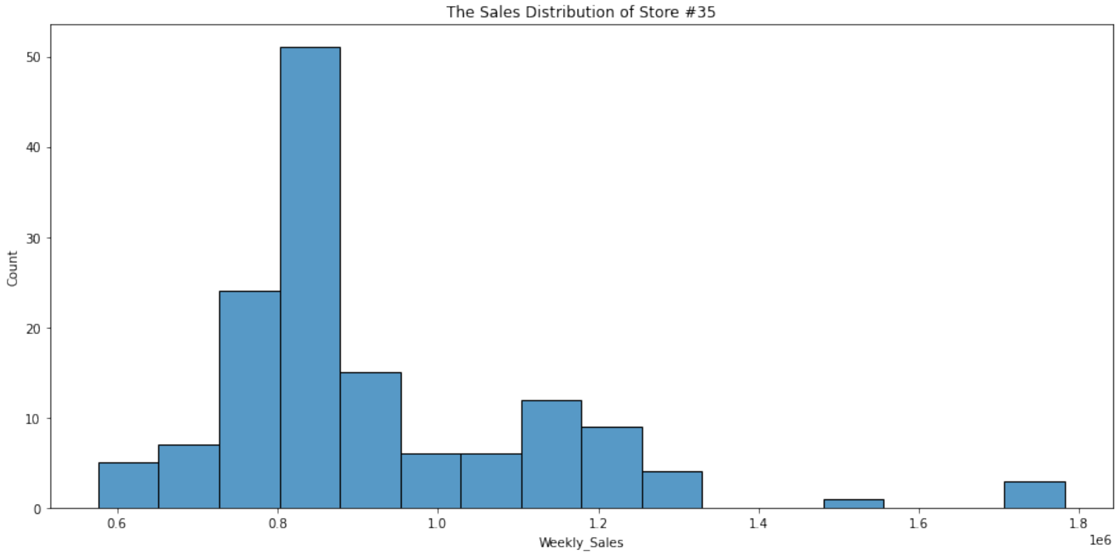
[20]:
```
       Coefficient of mean to standard deviation
Store
1                                       0.100292
2                                       0.123424
```

```
3                                       0.115021
4                                       0.127083
5                                       0.118668
6                                       0.135823
7                                       0.197305
8                                       0.116953
9                                       0.126895
10                                      0.159133
11                                      0.122262
12                                      0.137925
13                                      0.132514
14                                      0.157137
15                                      0.193384
16                                      0.165181
17                                      0.125521
18                                      0.162845
19                                      0.132680
20                                      0.130903
21                                      0.170292
22                                      0.156783
23                                      0.179721
24                                      0.123637
25                                      0.159860
26                                      0.110111
27                                      0.135155
28                                      0.137330
29                                      0.183742
30                                      0.052008
31                                      0.090161
32                                      0.118310
33                                      0.092868
34                                      0.108225
35                                      0.229681
36                                      0.162579
37                                      0.042084
38                                      0.110875
39                                      0.149908
40                                      0.123430
41                                      0.148177
42                                      0.090335
43                                      0.064104
44                                      0.081793
45                                      0.165613
```

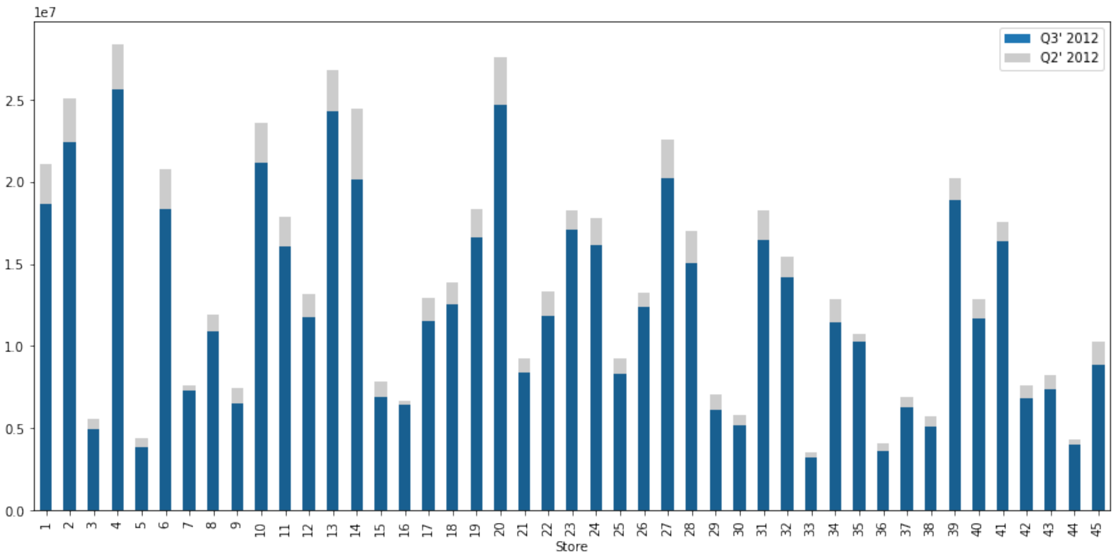
[21]:

```
# Distribution of store has maximum coefficient of mean to standard deviation
```

```
coef_mean_std_max = coef_mean_std.sort_values(by='Coefficient of mean to␣
 ↪standard deviation')
plt.figure(figsize=(15,7))
sns.histplot(walmart[walmart['Store'] == coef_mean_std_max.tail(1).
 ↪index[0]]['Weekly_Sales'])
plt.title('The Sales Distribution of Store #'+str(coef_mean_std_max.tail(1).
 ↪index[0]));
```



[22]:
```
# Which store/s has good quarterly growth rate in Q3’2012

plt.figure(figsize=(15,7))

# Sales for third quarterly in 2012
Q3=walmart[(walmart['Date']>'2012-07-01')&(walmart['Date']<'2012-09-30')].
 ↪groupby('Store')['Weekly_Sales'].sum()

# Sales for second quarterly in 2012
Q2=walmart[(walmart['Date']>'2012-04-01')&(walmart['Date']<'2012-06-30')].
 ↪groupby('Store')['Weekly_Sales'].sum()
# Plotting the difference between sales for second and third quarterly
Q2.plot(ax=Q3.plot(kind='bar',legend=True),kind='bar',color='black',alpha=0.
 ↪2,legend=True);
plt.legend(["Q3' 2012", "Q2' 2012"]);
```

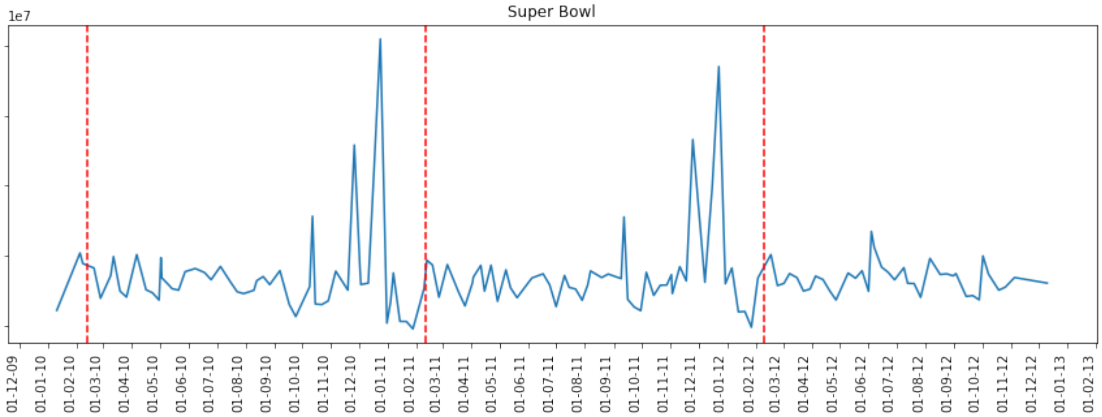


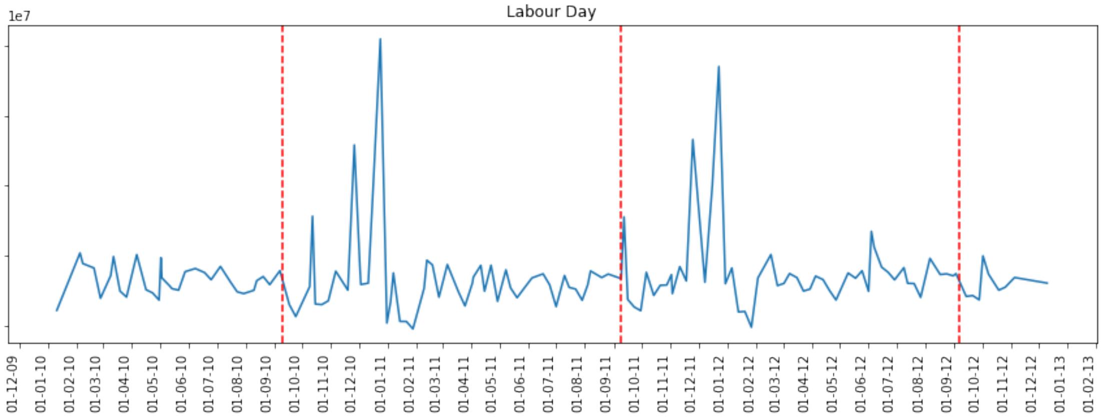
```
[23]: #  store/s has good quarterly growth rate in Q3’2012 - .
       ↪sort_values(by='Weekly_Sales')
      print('Store have good quarterly growth rate in Q3’2012 is Store '+str(Q3.
       ↪idxmax())+' With '+str(Q3.max())+' $')
```

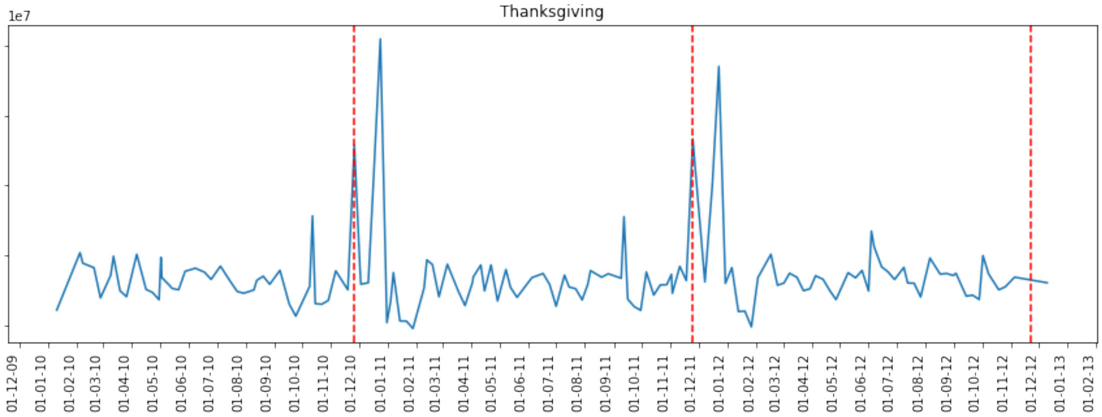
```
Store have good quarterly growth rate in Q3’2012 is Store 4 With 25652119.35 $
```

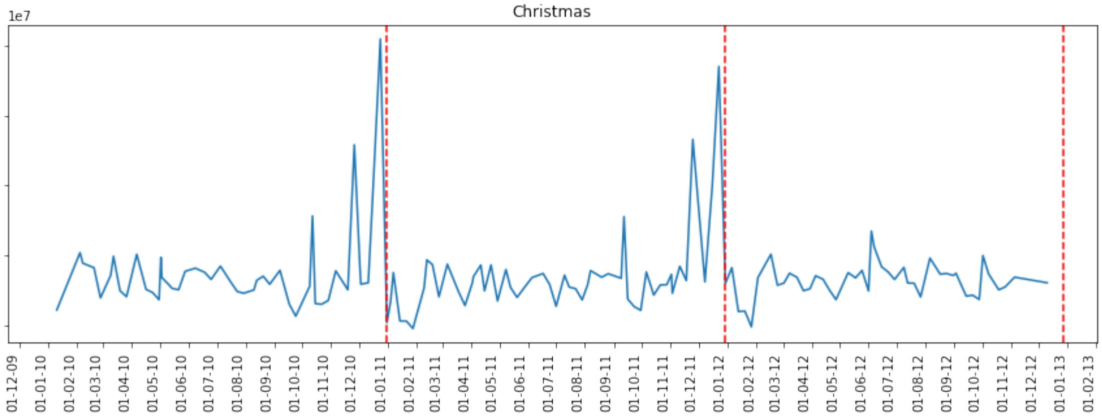
```
[27]: # 4. Some holidays have a negative impact on sales. Find out holidays which␣
       ↪have higher sales than the mean sales in non-holiday season for all stores␣
       ↪together

      # Holiday Events

      # Super Bowl: 12-Feb-10, 11-Feb-11, 10-Feb-12, 8-Feb-13
      # Labour Day: 10-Sep-10, 9-Sep-11, 7-Sep-12, 6-Sep-13
      # Thanksgiving: 26-Nov-10, 25-Nov-11, 23-Nov-12, 29-Nov-13
      # Christmas: 31-Dec-10, 30-Dec-11, 28-Dec-12, 227-Dec-13




      def plot_line(df,holiday_dates,holiday_label):
          fig, ax = plt.subplots(figsize = (15,5))
          ax.plot(df['Date'],df['Weekly_Sales'],label=holiday_label)

          for day in holiday_dates:
              day = datetime.strptime(day, '%d-%m-%Y')
              plt.axvline(x=day, linestyle='--', c='r')
```

```
    plt.title(holiday_label)
    x_dates = df['Date'].dt.strftime('%Y-%m-%d').sort_values().unique()
    xfmt = dates.DateFormatter('%d-%m-%y')
    ax.xaxis.set_major_formatter(xfmt)
    ax.xaxis.set_major_locator(dates.DayLocator(1))
    plt.gcf().autofmt_xdate(rotation=90)
    plt.show()


total_sales = walmart.groupby('Date')['Weekly_Sales'].sum().reset_index()
Super_Bowl =['12-2-2010', '11-2-2011', '10-2-2012']
Labour_Day =  ['10-9-2010', '9-9-2011', '7-9-2012']
Thanksgiving =  ['26-11-2010', '25-11-2011', '23-11-2012']
Christmas = ['31-12-2010', '30-12-2011', '28-12-2012']

plot_line(total_sales,Super_Bowl,'Super Bowl')
plot_line(total_sales,Labour_Day,'Labour Day')
plot_line(total_sales,Thanksgiving,'Thanksgiving')
plot_line(total_sales,Christmas,'Christmas')
```







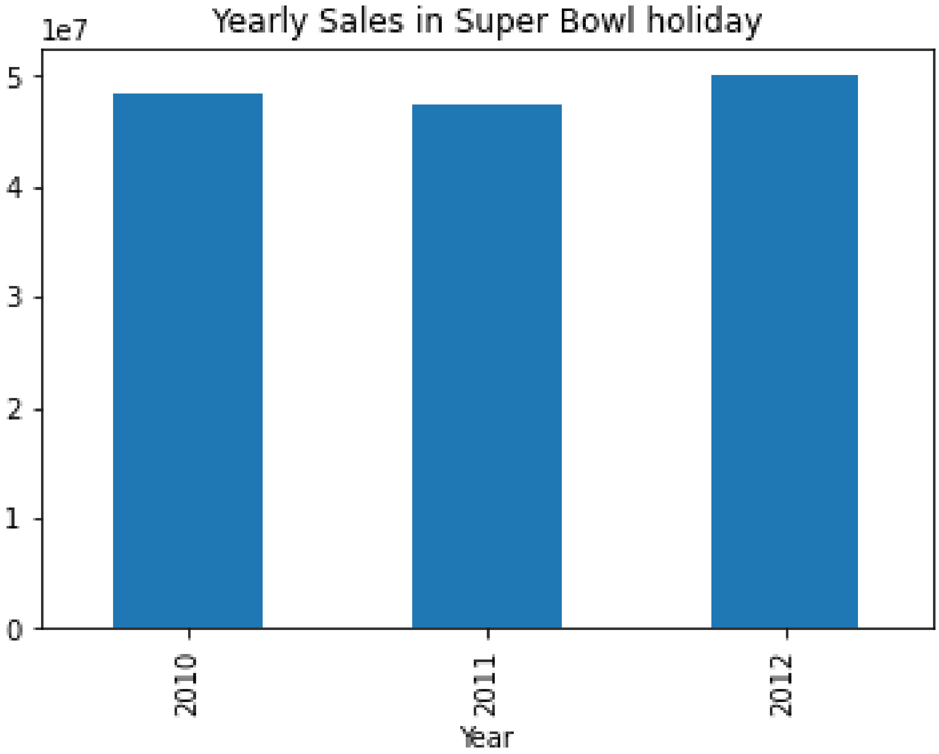


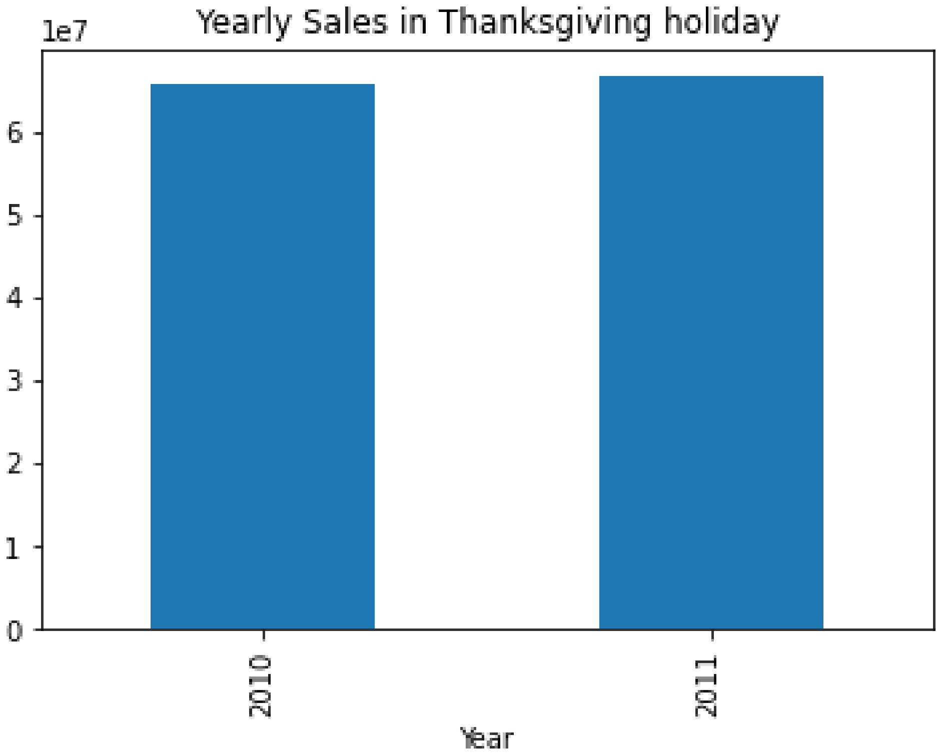
[32]:
```
# The sales increased during thanksgiving. And the sales decreased during␣
 ↪christmas.

walmart.loc[walmart.Date.isin(Super_Bowl)]

# Yearly Sales in holidays

Super_Bowl_df = pd.DataFrame(walmart.loc[walmart.Date.isin(Super_Bowl)].
 ↪groupby('Year')['Weekly_Sales'].sum())
Thanksgiving_df = pd.DataFrame(walmart.loc[walmart.Date.isin(Thanksgiving)].
 ↪groupby('Year')['Weekly_Sales'].sum())
Labour_Day_df = pd.DataFrame(walmart.loc[walmart.Date.isin(Labour_Day)].
 ↪groupby('Year')['Weekly_Sales'].sum())
Christmas_df = pd.DataFrame(walmart.loc[walmart.Date.isin(Christmas)].
 ↪groupby('Year')['Weekly_Sales'].sum())
```

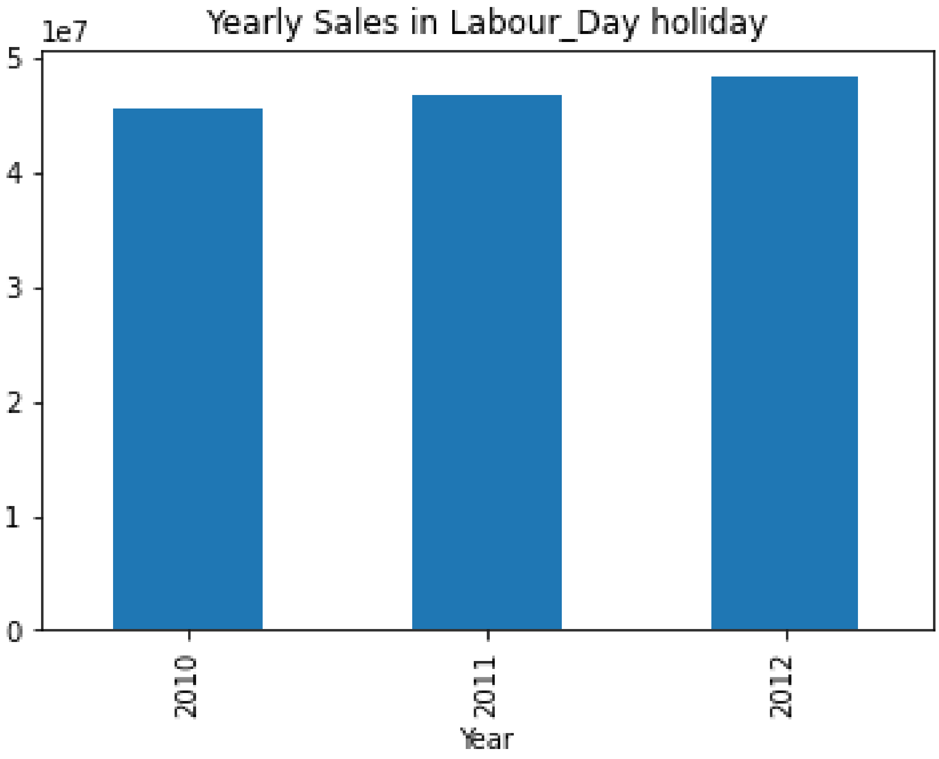
```
Super_Bowl_df.plot(kind='bar',legend=False,title='Yearly Sales in Super Bowl␣
  ↪holiday')
Thanksgiving_df.plot(kind='bar',legend=False,title='Yearly Sales in␣
  ↪Thanksgiving holiday')
Labour_Day_df.plot(kind='bar',legend=False,title='Yearly Sales in Labour_Day␣
  ↪holiday')
Christmas_df.plot(kind='bar',legend=False,title='Yearly Sales in Christmas␣
  ↪holiday')
```

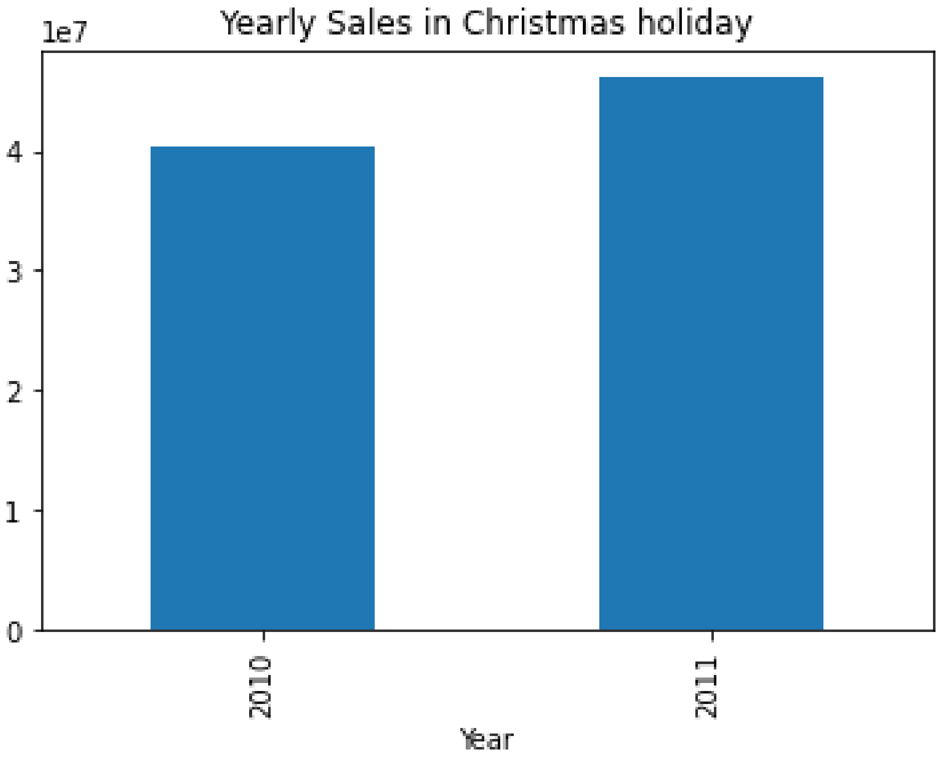
```
/tmp/ipykernel_76/644028570.py:8: UserWarning: Parsing dates in DD/MM/YYYY
format when dayfirst=False (the default) was specified. This may lead to
inconsistently parsed dates! Specify a format to ensure consistent parsing.
  Thanksgiving_df = pd.DataFrame(walmart.loc[walmart.Date.isin(Thanksgiving)].gr
oupby('Year')['Weekly_Sales'].sum())
/tmp/ipykernel_76/644028570.py:10: UserWarning: Parsing dates in DD/MM/YYYY
format when dayfirst=False (the default) was specified. This may lead to
inconsistently parsed dates! Specify a format to ensure consistent parsing.
  Christmas_df = pd.DataFrame(walmart.loc[walmart.Date.isin(Christmas)].groupby(
'Year')['Weekly_Sales'].sum())
```

```
[32]: <AxesSubplot: title={'center': 'Yearly Sales in Christmas holiday'},
      xlabel='Year'>
```



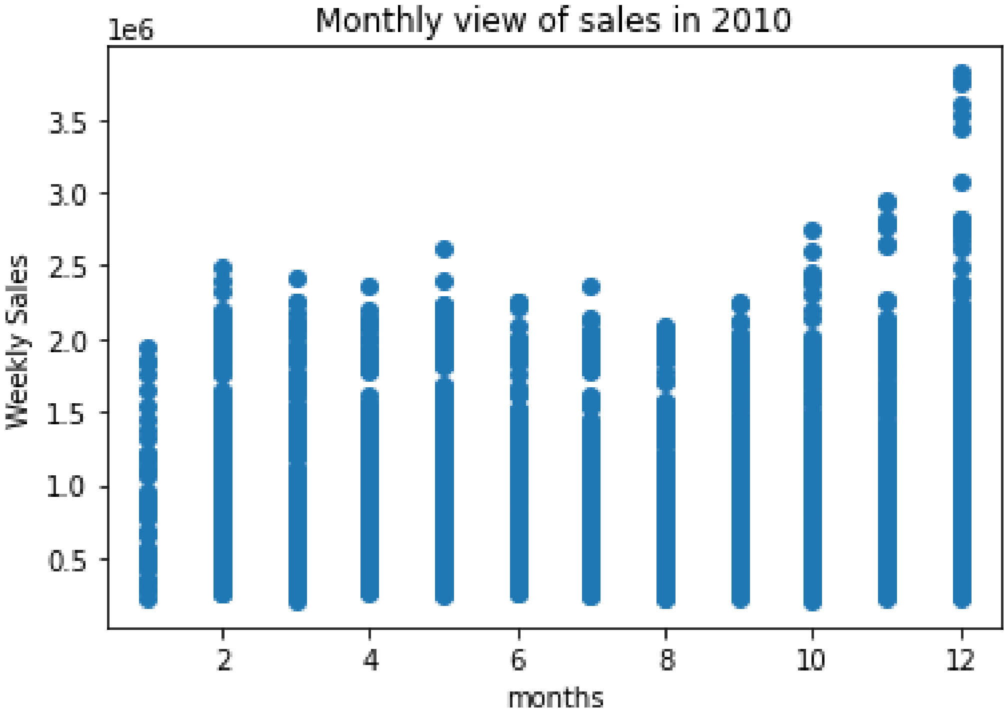

Yearly Sales in Thanksgiving holiday
1e7
6
5
4
3
2
1
0
2010
2011
Year

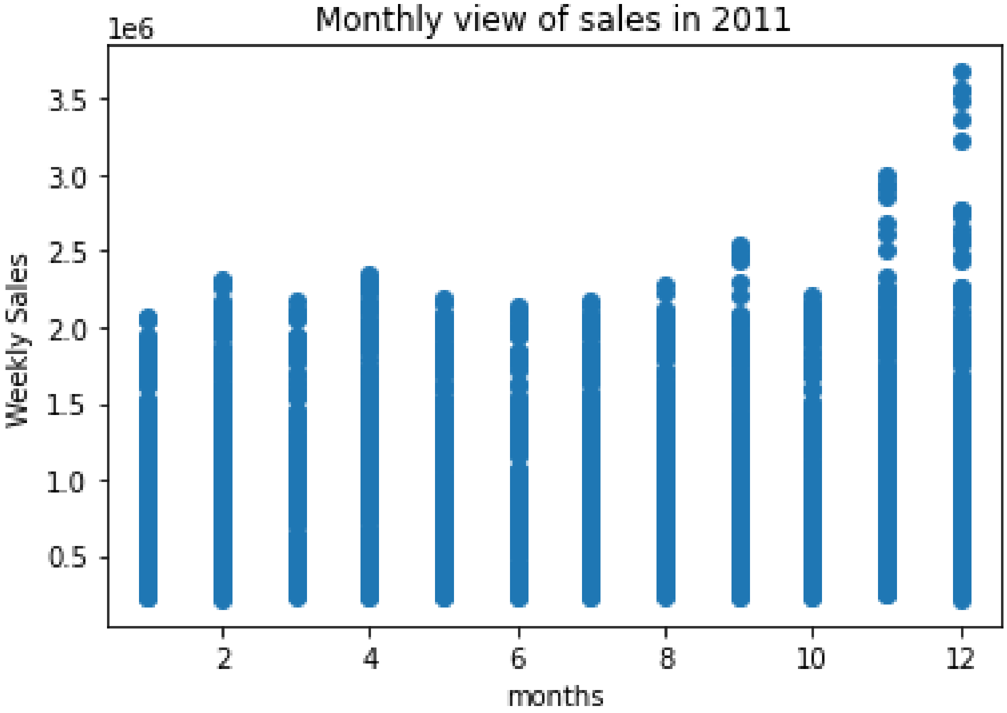

Yearly Sales in Labour_Day holiday
1e7
5
4
3
2
1
0
2010
2011
2012
Year

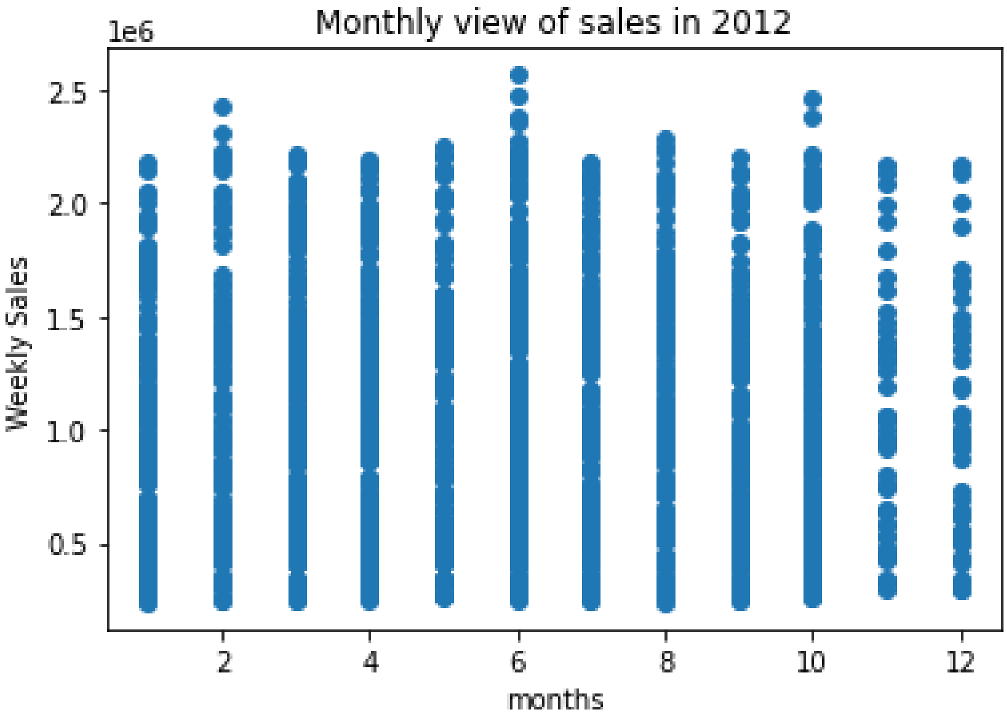

Yearly Sales in Christmas holiday
1e7
4
3
2
1
0
2010
2011
Year

[33]:

```
# 5. Provide a monthly and semester view of sales in units and give insights

# Monthly view of sales for each years

plt.scatter(walmart[walmart.Year==2010]["Month"],walmart[walmart.
 ↪Year==2010]["Weekly_Sales"])
plt.xlabel("months")
plt.ylabel("Weekly Sales")
plt.title("Monthly view of sales in 2010")
plt.show()

plt.scatter(walmart[walmart.Year==2011]["Month"],walmart[walmart.
 ↪Year==2011]["Weekly_Sales"])
plt.xlabel("months")
plt.ylabel("Weekly Sales")
plt.title("Monthly view of sales in 2011")
plt.show()

plt.scatter(walmart[walmart.Year==2012]["Month"],walmart[walmart.
 ↪Year==2012]["Weekly_Sales"])
plt.xlabel("months")
plt.ylabel("Weekly Sales")
plt.title("Monthly view of sales in 2012")
plt.show()
```

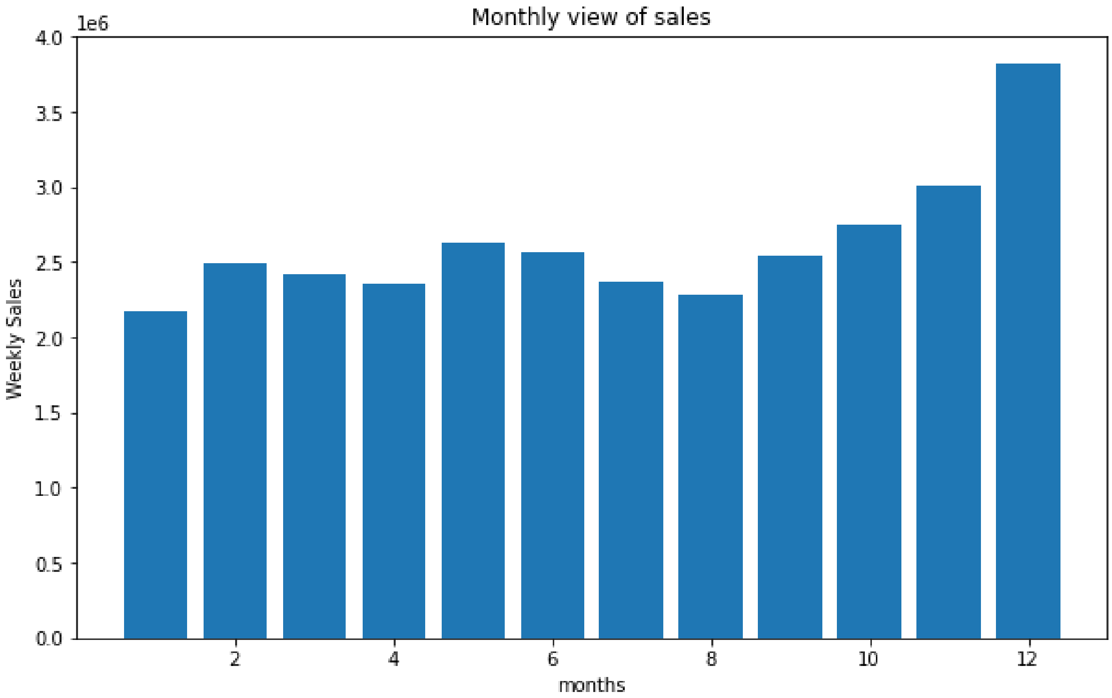

Monthly view of sales in 2010
1e6
Weekly Sales
3.5
3.0
2.5
2.0
1.5
1.0
0.5
2
4
6
8
10
12
months


Monthly view of sales in 2011
1e6
Weekly Sales
3.5
3.0
2.5
2.0
1.5
1.0
0.5
2
4
6
8
10
12
months



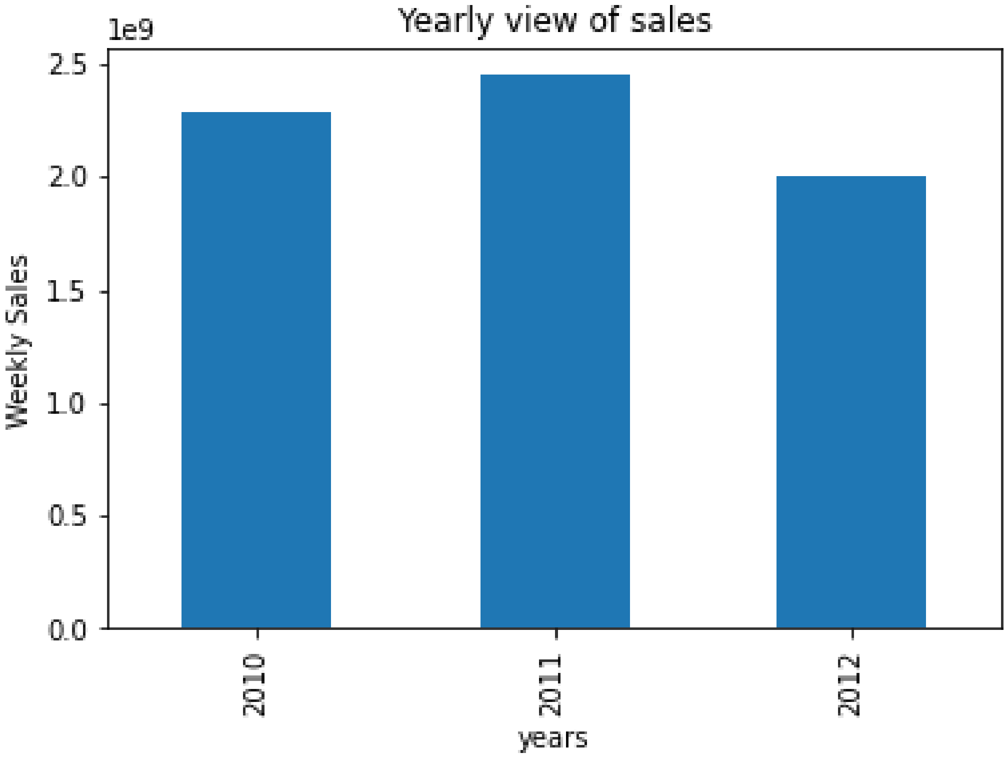
```
[34]: # Monthly view of sales for all years

      plt.figure(figsize=(10,6))
      plt.bar(walmart["Month"],walmart["Weekly_Sales"])
      plt.xlabel("months")
      plt.ylabel("Weekly Sales")
      plt.title("Monthly view of sales")
```

```
[34]: Text(0.5, 1.0, 'Monthly view of sales')
```

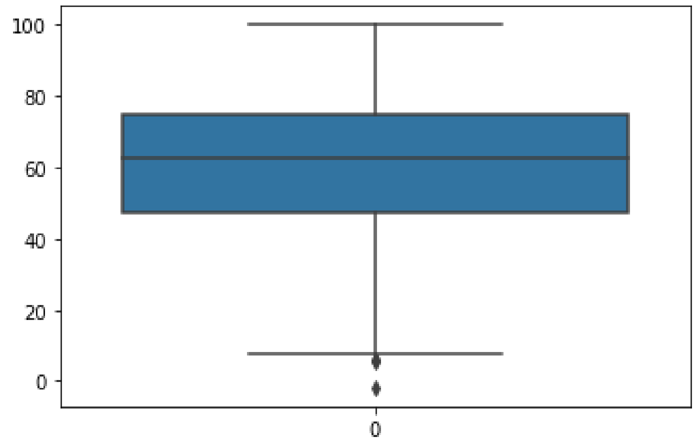


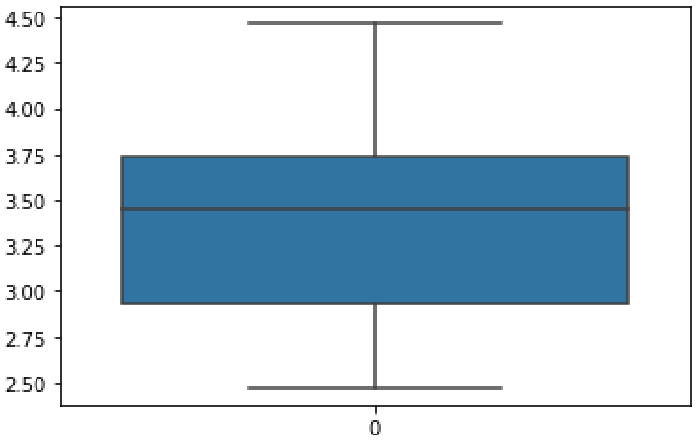
[36]:
```
# Yearly view of sales

plt.figure(figsize=(10,6))
walmart.groupby("Year")[["Weekly_Sales"]].sum().plot(kind='bar',legend=False)
plt.xlabel("years")
plt.ylabel("Weekly Sales")
plt.title("Yearly view of sales");
```

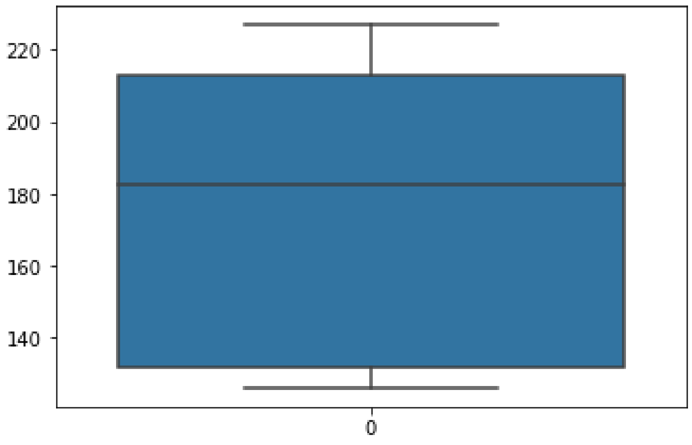
```
<Figure size 720x432 with 0 Axes>
```

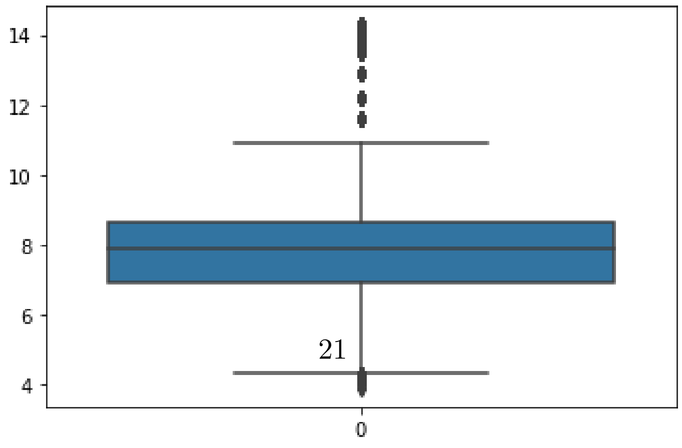


[38]:
```
# Build prediction models to forecast demand (Modeling)
fig, axs = plt.subplots(4,figsize=(6,18))
X = walmart[['Temperature','Fuel_Price','CPI','Unemployment']]
for i,column in enumerate(X):
    sns.boxplot(walmart[column], ax=axs[i])
```


100
80
60
40
20
0
0


4.50
4.25
4.00
3.75
3.50
3.25
3.00
2.75
2.50
0


220
200
180
160
140
0


14
12
10
8
6
4
0

[39]:

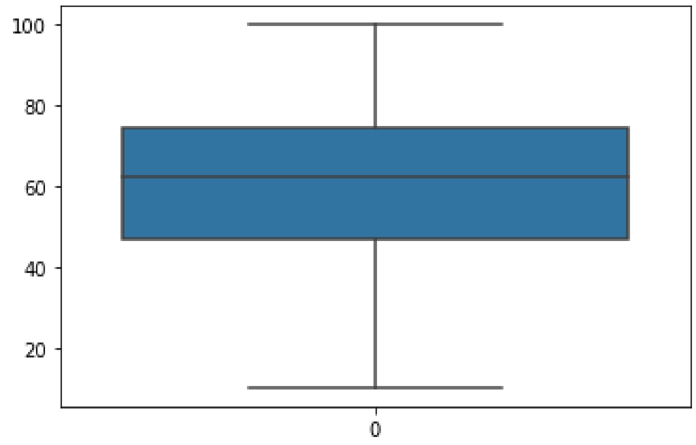
```
# DROP THE OUTLIERS

new = walmart[(walmart['Unemployment']<10) & (walmart['Unemployment']>4.5) &␣
 ↪(walmart['Temperature']>10)]
new
```

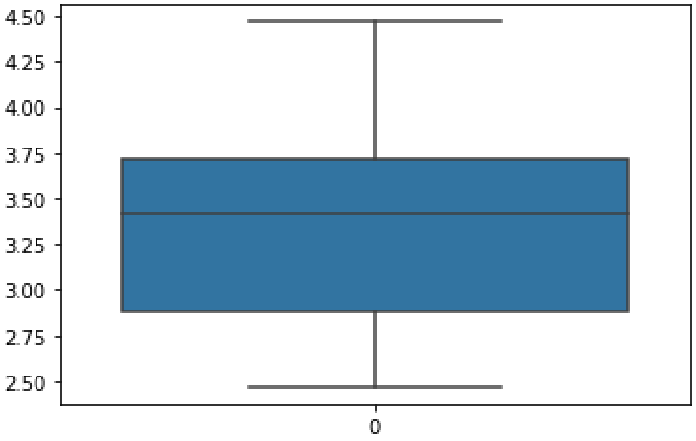
[39]:

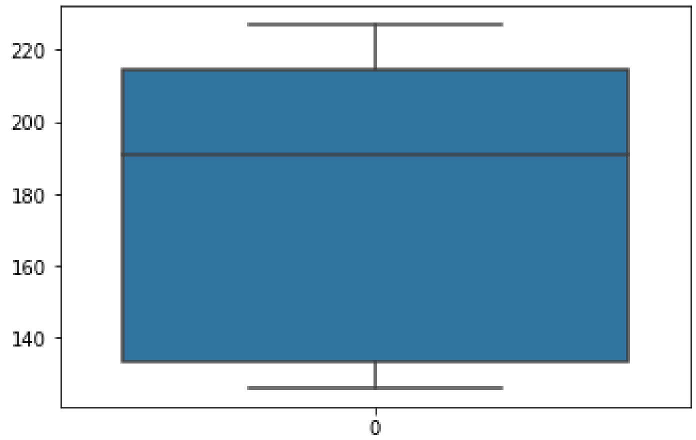
```
      Store        Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
0         1  2010-05-02    1643690.90             0        42.31       2.572
1         1  2010-12-02    1641957.44             1        38.51       2.548
2         1  2010-02-19    1611968.17             0        39.93       2.514
3         1  2010-02-26    1409727.59             0        46.63       2.561
4         1  2010-05-03    1554806.68             0        46.50       2.625
…       …         …            …             …          …          …
6430     45  2012-09-28     713173.95             0        64.88       3.997
6431     45  2012-05-10     733455.07             0        64.89       3.985
6432     45  2012-12-10     734464.36             0        54.47       4.000
6433     45  2012-10-19     718125.53             0        56.47       3.969
6434     45  2012-10-26     760281.43             0        58.85       3.882

             CPI  Unemployment  Day  Month  Year
0     211.096358         8.106    2      5  2010
1     211.242170         8.106    2     12  2010
2     211.289143         8.106   19      2  2010
3     211.319643         8.106   26      2  2010
4     211.350143         8.106    3      5  2010
…            …            …   …     …    …
6430  192.013558         8.684   28      9  2012
6431  192.170412         8.667   10      5  2012
6432  192.327265         8.667   10     12  2012
6433  192.330854         8.667   19     10  2012
6434  192.308899         8.667   26     10  2012

[5658 rows x 11 columns]
```

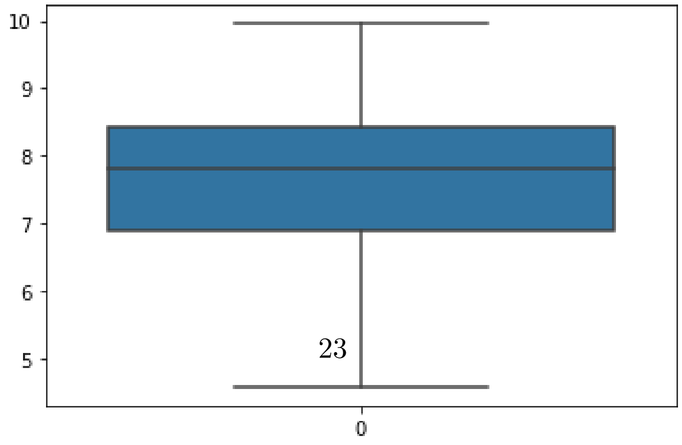
[40]:

```
# CHECKING THE OUTLIERS

fig, axs = plt.subplots(4,figsize=(6,18))
B = new[['Temperature','Fuel_Price','CPI','Unemployment']]
for i,column in enumerate(X):
    sns.boxplot(new[column], ax=axs[i])
```


100
80
60
40
20
0


4.50
4.25
4.00
3.75
3.50
3.25
3.00
2.75
2.50
0


220
200
180
160
140
0


10
9
8
7
6
5
0

[41]:

```
# STATISTICAL MODEL

# Build prediction model for forcast demand

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.linear_model import LinearRegression
```

[44]:

```
# Selecting features and target

X = new[['Store','Fuel_Price','CPI','Unemployment','Day','Month','Year']]
Y = new['Weekly_Sales']
```

[46]:

```
# Split data to train and test (0.80:0.20)
X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size=0.2)
```

[58]:

```
# Linear regression model

x=walmart.drop(["Weekly_Sales","Date"],axis=1)
y=walmart["Weekly_Sales"]
```

[59]:

```
linreg=LinearRegression(n_jobs=-1)
```

[61]:

```
from sklearn import model_selection
```

[62]:

```
xtrain,xtest,ytrain,ytest=model_selection.train_test_split(x,y,test_size=0.
  ↪4,random_state=42)
```

[63]:

```
linreg.fit(xtrain,ytrain)
```

[63]:

```
LinearRegression(n_jobs=-1)
```

[64]:

```
print(linreg.intercept_)
print(linreg.coef_)
```

```
84171361.04320997
[-15076.05743532  14940.56392359   -744.77138548  49882.84830669
  -2178.75498529 -26725.92004156  -1452.88018785  11680.40062841
 -40959.56523516]
```

[65]:

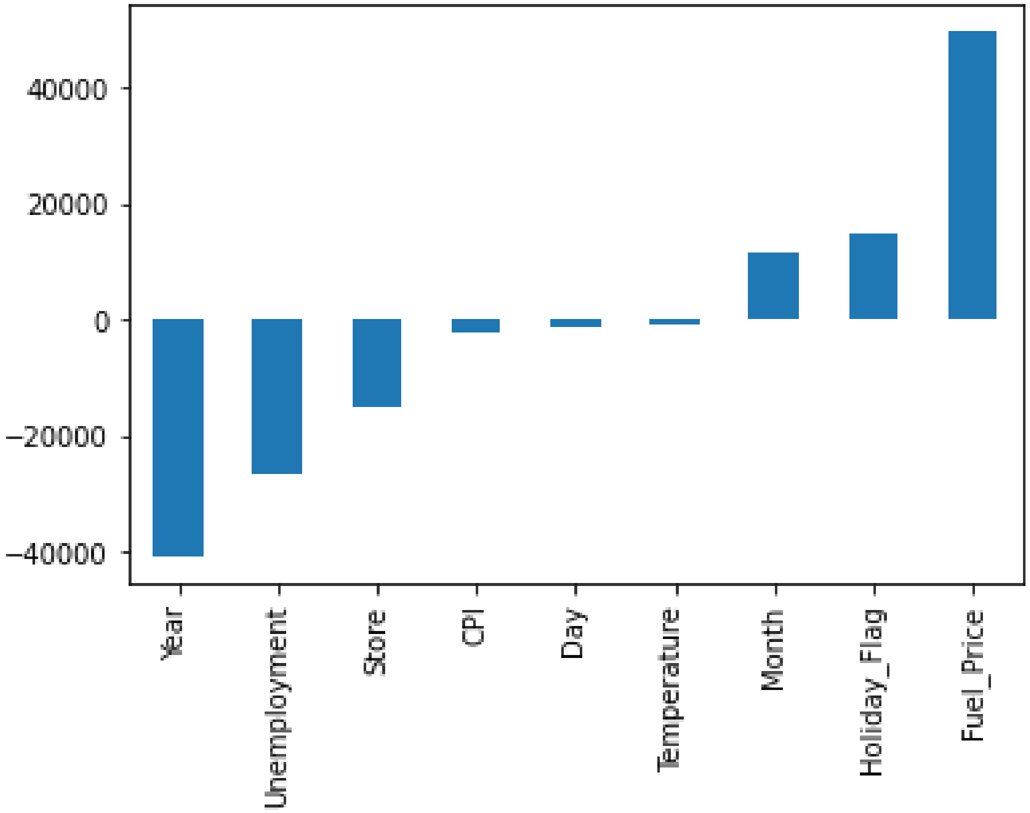
```
x.columns
```

```
[65]: Index(['Store', 'Holiday_Flag', 'Temperature', 'Fuel_Price', 'CPI',
              'Unemployment', 'Day', 'Month', 'Year'],
             dtype='object')
```

```
[66]: features=['Store', 'Holiday_Flag', 'Temperature', 'Fuel_Price', 'CPI',␣
        ↪'Unemployment', 'Year', 'Month'],
```

```
[67]: relation=pd.Series(linreg.coef_,x.columns).sort_values()
      relation.plot(kind="bar")
```

```
[67]: <AxesSubplot: >
```



```
[68]: # The plot shows that fuel price have greater positive impact on weekly sales.␣
        ↪Unemployment also has certain negative impact on weekly sales. CPI has least␣
        ↪impact towards weekly sales.

      print(format(linreg.score(xtest,ytest)))

      0.14950449647465958
```

[72]:
```
from math import sqrt
from sklearn.metrics import mean_squared_error
```

[73]:
```
print(sqrt(mean_squared_error(ytrain,linreg.predict(xtrain))))
```

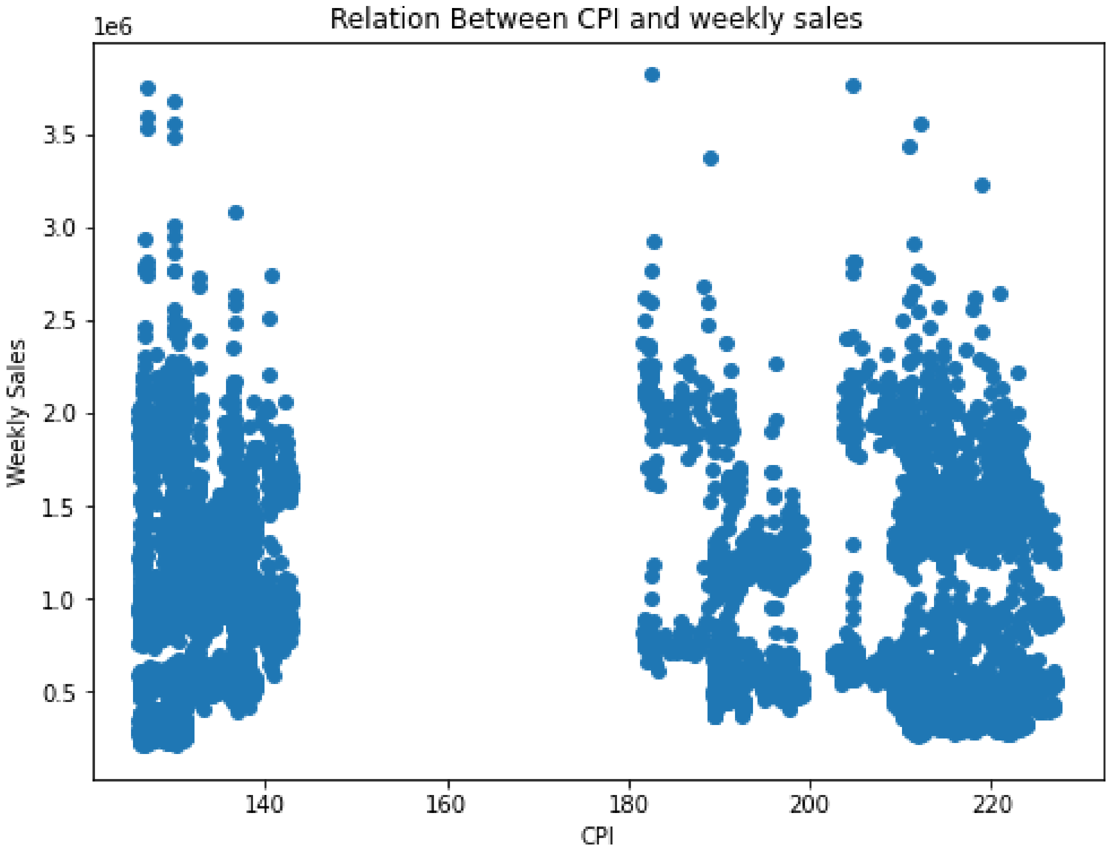
```
522476.30985960393
```

[74]:
```
print(sqrt(mean_squared_error(ytest,linreg.predict(xtest))))
```

```
519815.2248616646
```

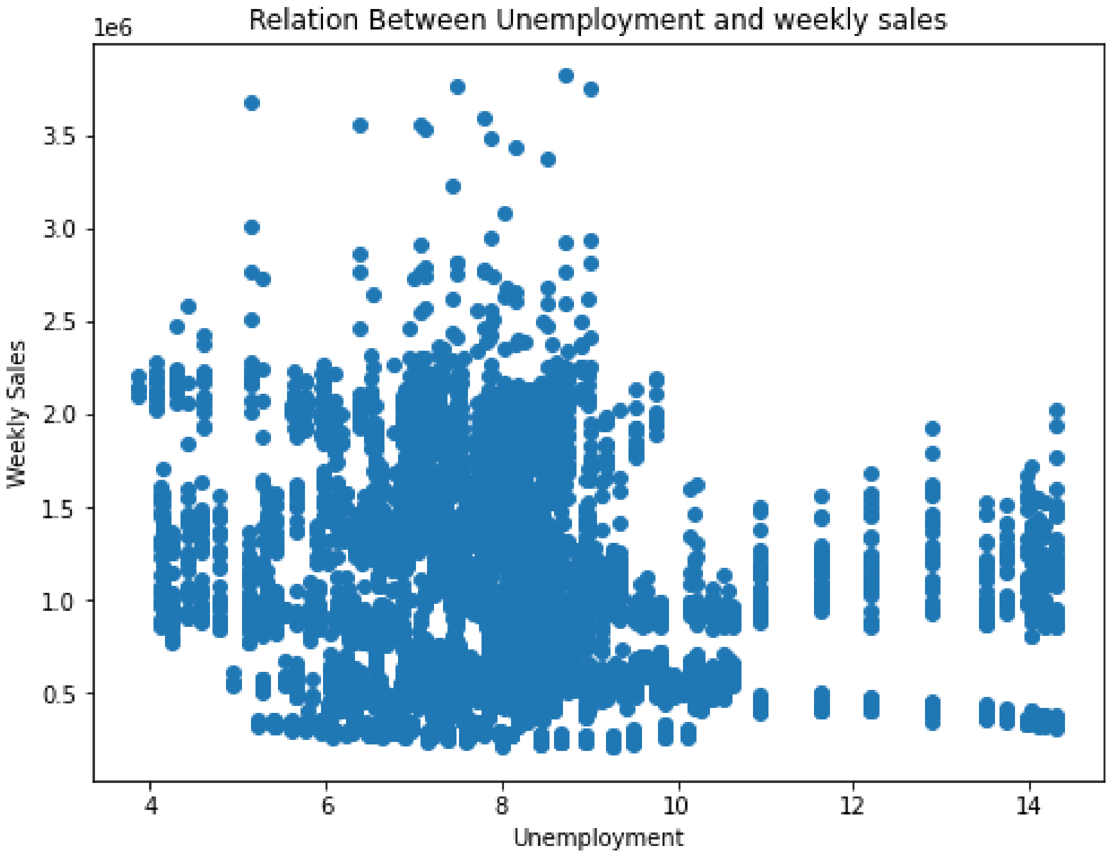
[75]:
```
plt.figure(figsize=(8,6))
plt.scatter(walmart["CPI"],walmart["Weekly_Sales"])
plt.title("Relation Between CPI and weekly sales")
plt.xlabel("CPI")
plt.ylabel("Weekly Sales")
```

[75]:
```
Text(0, 0.5, 'Weekly Sales')
```



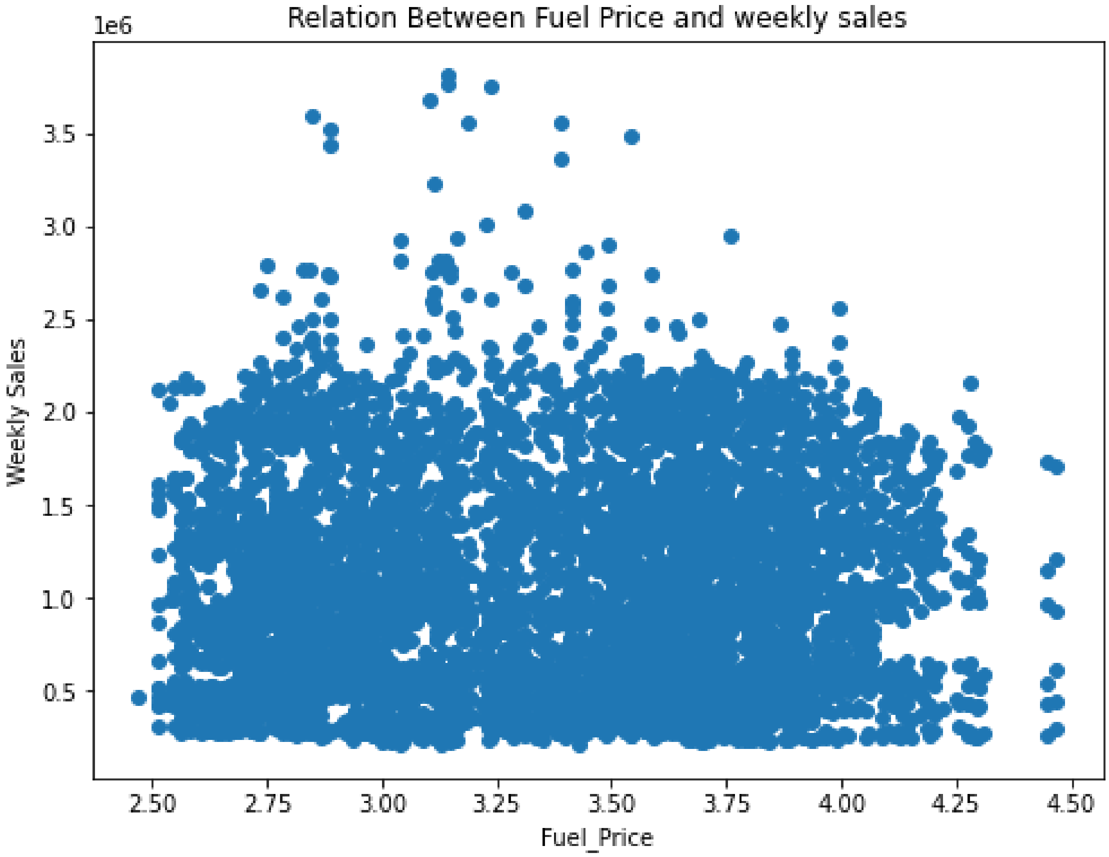
[76]:
```
plt.figure(figsize=(8,6))
plt.scatter(walmart["Unemployment"],walmart["Weekly_Sales"])
plt.title("Relation Between Unemployment and weekly sales")
plt.xlabel("Unemployment")
plt.ylabel("Weekly Sales")
```

[76]:
```
Text(0, 0.5, 'Weekly Sales')
```



[77]:
```
plt.figure(figsize=(8,6))
plt.scatter(walmart["Fuel_Price"],walmart["Weekly_Sales"])
plt.title("Relation Between Fuel Price and weekly sales")
plt.xlabel("Fuel_Price")
plt.ylabel("Weekly Sales")
```

[77]:
```
Text(0, 0.5, 'Weekly Sales')
```



[78]:
```
# CHANGING DATES INTO DAYS

walmart['days'] = walmart['Date'].dt.day_name()
```

[80]:
```
walmart
```

[80]:
```
      Store       Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
0         1 2010-05-02    1643690.90             0        42.31       2.572
1         1 2010-12-02    1641957.44             1        38.51       2.548
2         1 2010-02-19    1611968.17             0        39.93       2.514
3         1 2010-02-26    1409727.59             0        46.63       2.561
4         1 2010-05-03    1554806.68             0        46.50       2.625
…       …          …             …           …          …           …
6430     45 2012-09-28     713173.95             0        64.88       3.997
6431     45 2012-05-10     733455.07             0        64.89       3.985
6432     45 2012-12-10     734464.36             0        54.47       4.000
6433     45 2012-10-19     718125.53             0        56.47       3.969
6434     45 2012-10-26     760281.43             0        58.85       3.882

             CPI  Unemployment  Day  Month  Year      days
```

```
0     211.096358         8.106    2      5  2010    Sunday
1     211.242170         8.106    2     12  2010  Thursday
2     211.289143         8.106   19      2  2010    Friday
3     211.319643         8.106   26      2  2010    Friday
4     211.350143         8.106    3      5  2010    Monday
…          …           …  …    …   …       …
6430  192.013558         8.684   28      9  2012    Friday
6431  192.170412         8.667   10      5  2012  Thursday
6432  192.327265         8.667   10     12  2012    Monday
6433  192.330854         8.667   19     10  2012    Friday
6434  192.308899         8.667   26     10  2012    Friday

[6435 rows x 12 columns]
```

[ ]: