Name: CH.VENKATA ROHITH
Title of the project : Walmart Store Sales.

In [23]:

```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from scipy.special import boxcox1p
import seaborn as sns
```

Problem statement: predict the sales level for future from the Walmart DataSet

sorting the data by date.

In [24]:

```
#Getting data
data=pd.read_csv("Walmart_Store_sales_Dataset.csv")

#sorting values of Data
data=data.sort_values(by='Date')
data.head(10)
```

Out[24]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
5208	37	01-04-2011	534578.78	0	67.31	3.524	213.577670	8.300
1204	9	01-04-2011	520962.14	0	56.12	3.524	218.445164	6.380
1776	13	01-04-2011	1864238.64	0	42.49	3.487	128.719935	7.193
2634	19	01-04-2011	1305950.22	0	30.68	3.811	134.068258	7.658
6066	43	01-04-2011	611585.54	0	67.79	3.524	206.673311	10.581
3349	24	01-04-2011	1163803.30	0	35.73	3.811	134.068258	8.212
2920	21	01-04-2011	732056.37	0	56.36	3.524	214.488691	7.931
3778	27	01-04-2011	1628868.28	0	37.27	3.811	137.955893	7.725
346	3	01-04-2011	374556.08	0	68.76	3.524	218.211418	7.574
4064	29	01-04-2011	475615.26	0	36.04	3.638	134.068258	9.966

1. **Analyzing the data**

In [25]:

```
data.min()
```

Out[25]:

```
Store                    1
Date            01-04-2011
Weekly_Sales      209986.25
Holiday_Flag              0
Temperature           -2.06
Fuel_Price            2.472
CPI                 126.064
```

```
Unemployment         3.879
dtype: object
```

In [26]:

```
data.mean()
```

Out[26]:

```
Store           2.300000e+01
Weekly_Sales    1.046965e+06
Holiday_Flag    6.993007e-02
Temperature     6.066378e+01
Fuel_Price      3.358607e+00
CPI             1.715784e+02
Unemployment    7.999151e+00
dtype: float64
```

In [27]:

```
data.median()
```

Out[27]:

```
Store              23.000000
Weekly_Sales    960746.040000
Holiday_Flag        0.000000
Temperature        62.670000
Fuel_Price          3.445000
CPI               182.616521
Unemployment        7.874000
dtype: float64
```

In [28]:

```
data.max()
```

Out[28]:

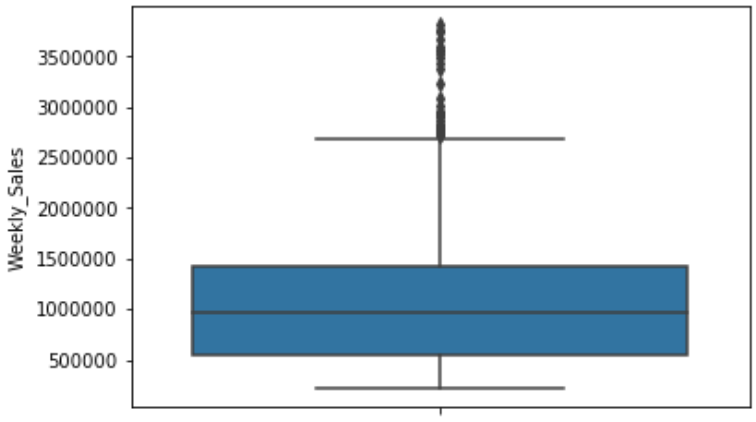
```
Store                  45
Date           31-12-2010
Weekly_Sales    3818686.45
Holiday_Flag             1
Temperature         100.14
Fuel_Price           4.468
CPI             227.232807
Unemployment        14.313
dtype: object
```

In [29]:

```
sns.boxplot(y='Weekly_Sales',data=data)
```

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f83bdbe3810>
```

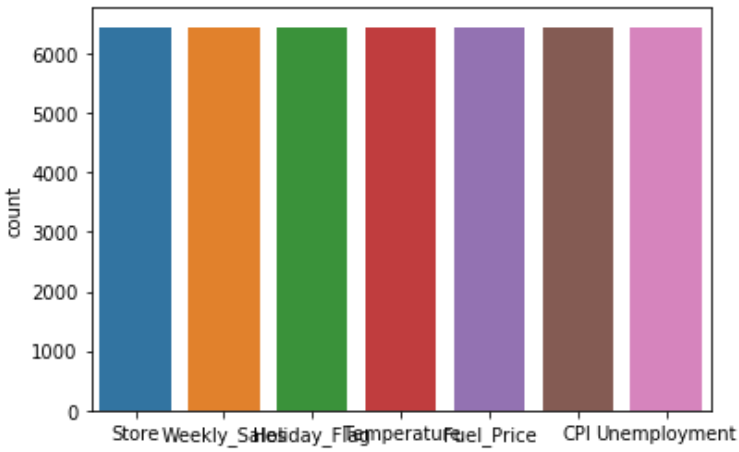


In [30]:

```
sns.countplot( data=data)
```

Out[30]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f83bd8ec0d0>
```

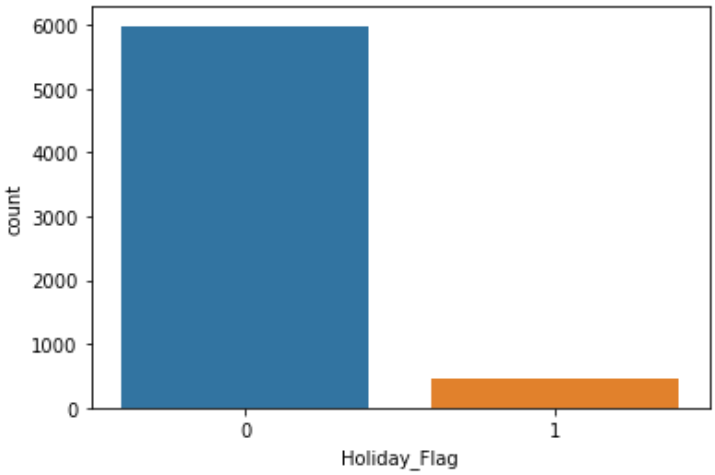


In [31]:

```
sns.countplot(x="Holiday_Flag", data=data)
```

Out[31]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f83bd862bd0>
```



Now, we check the null values of data

In [32]:

```
data.isnull().sum()
```

Out[32]:

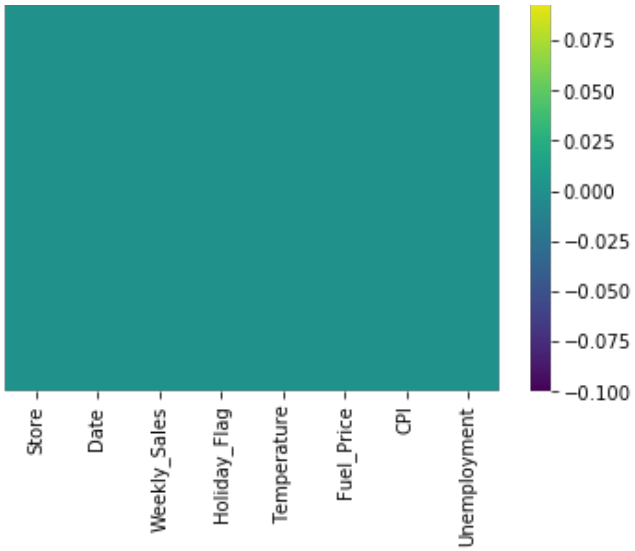
```
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
dtype: int64
```

In [33]:

```
sns.heatmap(data.isnull(),yticklabels=False, cmap="viridis")
```

Out[33]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f83bd843890>
```



Again checking for null values and verify that data is clean.

In [34]:

```
data.isnull().sum()
```

Out[34]:

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

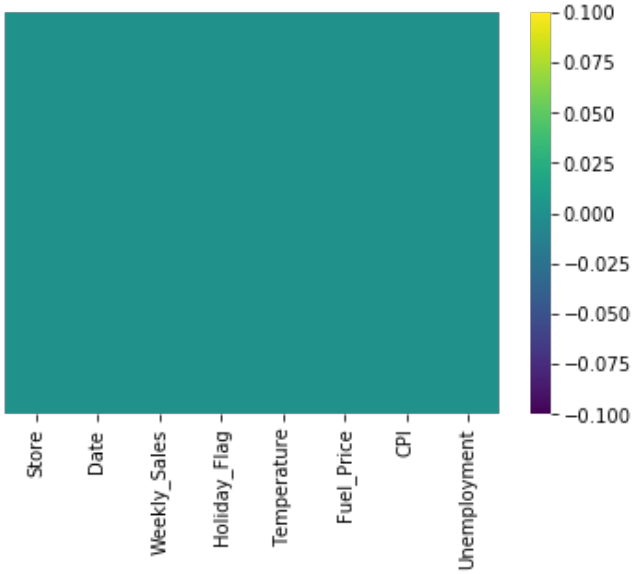
Heatmap for verifying there are no null values in data.

In [35]:

```
sns.heatmap(data.isnull(),yticklabels=False, cmap="viridis")
```

Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f83bd826790>
```



Converting Holiday_Flag in Holiday which is integer and 1 for holiday and 0 otherwise.

In [36]:

```
data['Holiday']=[int(i) for i in list(data.Holiday_Flag)]
data.head(10)
```

Out[36]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Holiday
5208	37	01-04-2011	534578.78	0	67.31	3.524	213.577670	8.300	0
1204	9	01-04-2011	520962.14	0	56.12	3.524	218.445164	6.380	0
1776	13	01-04-2011	1864238.64	0	42.49	3.487	128.719935	7.193	0
2634	19	01-04-2011	1305950.22	0	30.68	3.811	134.068258	7.658	0
6066	43	01-04-2011	611585.54	0	67.79	3.524	206.673311	10.581	0
3349	24	01-04-2011	1163803.30	0	35.73	3.811	134.068258	8.212	0
2920	21	01-04-2011	732056.37	0	56.36	3.524	214.488691	7.931	0
3778	27	01-04-2011	1628868.28	0	37.27	3.811	137.955893	7.725	0
346	3	01-04-2011	374556.08	0	68.76	3.524	218.211418	7.574	0
4064	29	01-04-2011	475615.26	0	36.04	3.638	134.068258	9.966	0

Converting type to an integer by one-hot encoding. The first column is also removed because we know of both columns B and C are 0 then it is A-type. So B=1 and C=0 for B.B=0 and C=1 for C.B=0 and C=0 for A.

Type_dummy=pd.get_dummies(data,drop_first=True) Type_dummy.head(10)

About ML:Here, we use the dataset of Walmart sales to forecast future sales using machine learning in Python. Linear regression use to forecast sales. Numpy, Pandas, Sklearn, Scipy, Seaborn Python libraries used in this program. We implement in three steps first to import libraries second by using that libraries prepare data and third forecast.

Splitting the train and test data

In [37]:

```
#splitting data in input and output
X=data.drop(['Weekly_Sales','Store','Date'],axis=1)
y=data['Weekly_Sales']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3)
```

Applying linear regression and fit the training data into it.

In [38]:

```
LR=LinearRegression(normalize=True)
LR.fit(X_train,y_train)
```

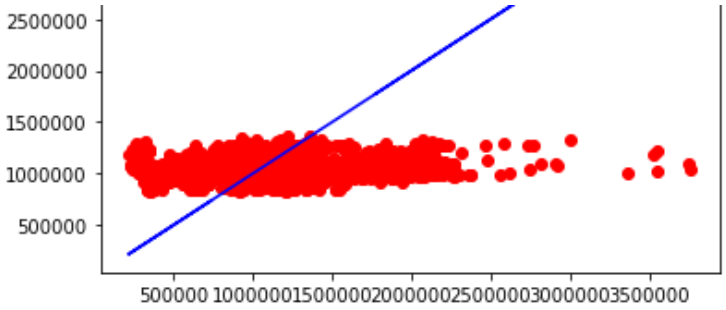
Out[38]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=True)
```

Predicting the data for test value as per linear regression.

In [39]:

```
y_pred=LR.predict(X_test)
plt.plot(y_test,y_pred,'ro')
plt.plot(y_test,y_test,'b-')
plt.show()
```



Evaluating the model by calculating errors by the root mean square error and R -squared.

In [40]:

```
Root_mean_square_error=np.sqrt(np.mean(np.square(y_test-y_pred)))
print(Root_mean_square_error)
```

```
563676.4571694164
```

In [41]:

```
from sklearn.metrics import r2_score
r2=r2_score(y_test,y_pred)
print(r2)
```

```
0.010030201239483283
```

In [ ]:

```

```