In [1]:
```
# Importing required Libraries.

import pandas as pd
from datetime import date
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [2]:
```
# Import Data Set.

import io
%cd "C:\Users\gkoppadx\OneDrive - Intel Corporation\Desktop\simple\1577429980_walmart_s
```

```
C:\Users\gkoppadx\OneDrive - Intel Corporation\Desktop\simple\1577429980_walmart_store_s
ales
```

In [3]:
```
# Read the CSV file.

walmartsales=pd.read_csv("Walmart_Store_sales.csv")
```

In [4]:
```
# Understand dataset.

walmartsales.head()
```

Out[4]:

| | 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 |

In [5]:
```
# Basic information about our dataset.

walmartsales.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 [6]:
```
#Maximum value in each column.

walmartsales.max()
```

Out[6]:
```
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 [7]:
```
walmartsales.describe()
```

Out[7]:

| | Store | Weekly_Sales | Holiday_Flag | Temperature | Fuel_Price | CPI | Unemployment |
|---|---|---|---|---|---|---|---|
| **count** | 6435.000000 | 6.435000e+03 | 6435.000000 | 6435.000000 | 6435.000000 | 6435.000000 | 6435.000000 |
| **mean** | 23.000000 | 1.046965e+06 | 0.069930 | 60.663782 | 3.358607 | 171.578394 | 7.999151 |
| **std** | 12.988182 | 5.643666e+05 | 0.255049 | 18.444933 | 0.459020 | 39.356712 | 1.875885 |
| **min** | 1.000000 | 2.099862e+05 | 0.000000 | -2.060000 | 2.472000 | 126.064000 | 3.879000 |
| **25%** | 12.000000 | 5.533501e+05 | 0.000000 | 47.460000 | 2.933000 | 131.735000 | 6.891000 |
| **50%** | 23.000000 | 9.607460e+05 | 0.000000 | 62.670000 | 3.445000 | 182.616521 | 7.874000 |
| **75%** | 34.000000 | 1.420159e+06 | 0.000000 | 74.940000 | 3.735000 | 212.743293 | 8.622000 |
| **max** | 45.000000 | 3.818686e+06 | 1.000000 | 100.140000 | 4.468000 | 227.232807 | 14.313000 |

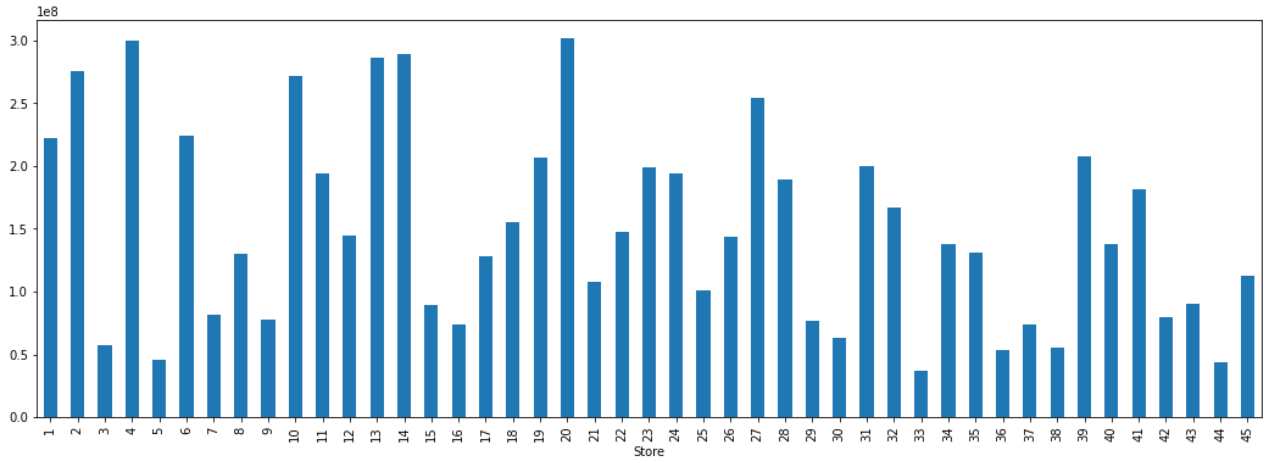
In [8]:
```
# store having maximum weekly sales.

sales_list= pd.DataFrame(walmartsales.groupby(['Store'])['Weekly_Sales'].sum())
sales_list.reset_index()
max_sales=sales_list.loc[sales_list['Weekly_Sales'] ==  sales_list['Weekly_Sales'].max(
max_sales

# We can see that store 20 has maximum weekly sales.
```

Out[8]:

| | Weekly_Sales |
|---|---|
| **Store** | |
| **20** | 3.013978e+08 |

In [9]:
```
# Plot showing weekly sales against stores.

plt.figure(figsize=(18,6))
walmartsales.groupby(['Store'])['Weekly_Sales'].sum().plot(kind='bar')
```

Out[9]:
```
<AxesSubplot:xlabel='Store'>
```



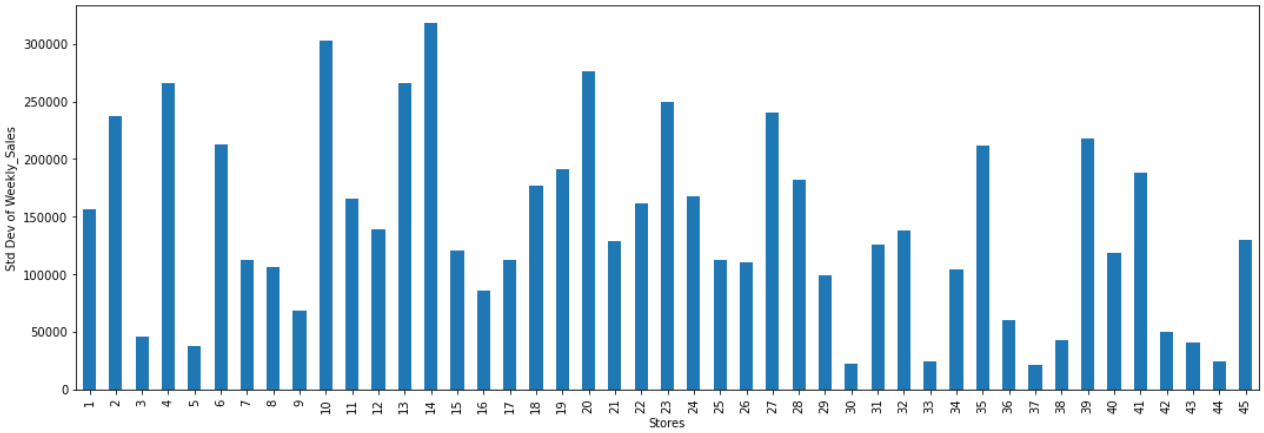

In [10]:
```
# store having maximum standard deviation i.e., the sales vary a lot. Also, finding out

maxstd=pd.DataFrame(walmartsales.groupby('Store').agg({'Weekly_Sales':['std','mean','va
maxstd = maxstd.reset_index()

maxstd['CoV'] =(maxstd[('Weekly_Sales','std')]/maxstd[('Weekly_Sales','mean')]) *100

# Finding the store with maximum standard deviation.

maxstd.loc[maxstd[('Weekly_Sales','std')]==maxstd[('Weekly_Sales','std')].max()]

# store with maximum standard deviation of 317569.949476 is 14.
```

Out[10]:

| | Store | Weekly_Sales | | | CoV |
|---|---|---|---|---|---|
| | | std | mean | var | |
| **13** | 14 | 317569.949476 | 2.020978e+06 | 1.008507e+11 | 15.713674 |

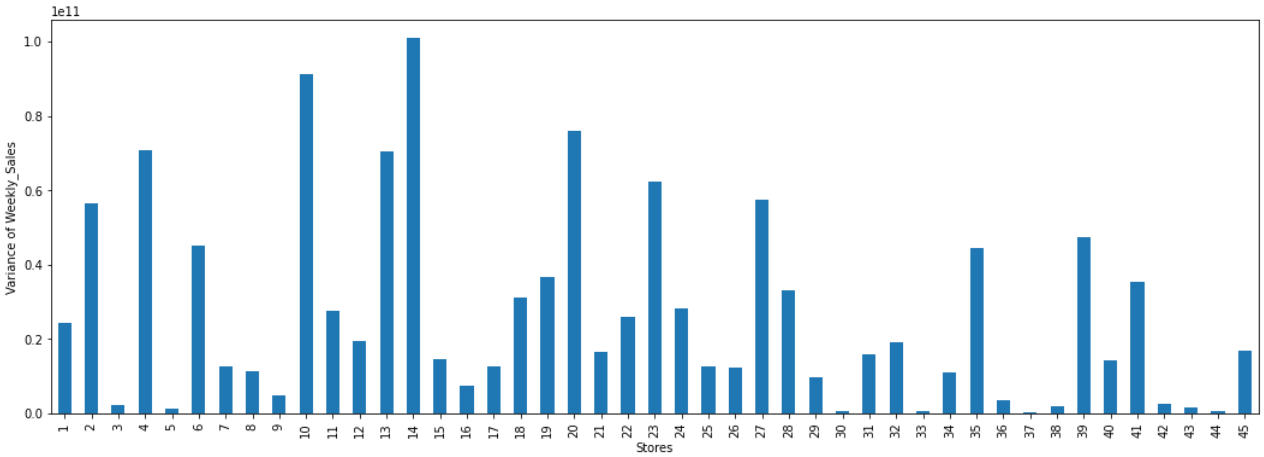
In [11]:
```
# Bar plot showing "Std Dev of Weekly_Sales" agianst "Stores"

plt.figure(figsize=(18,6))
walmartsales.Weekly_Sales.groupby(walmartsales.Store).std().plot(kind='bar')
plt.xlabel("Stores")
plt.ylabel("Std Dev of Weekly_Sales")
```

Out[11]:
```
Text(0, 0.5, 'Std Dev of Weekly_Sales')
```

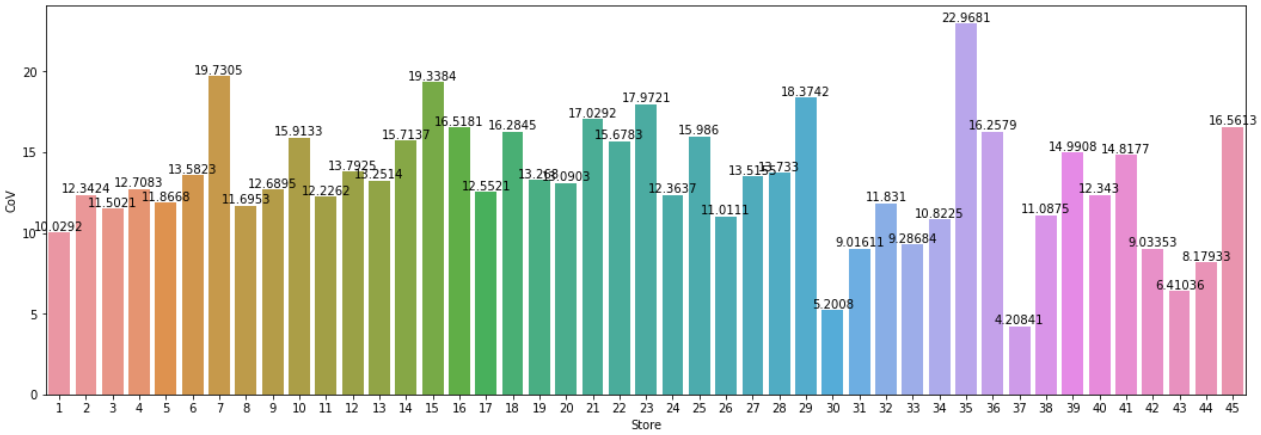



In [12]:
```
# Bar plot showing "var" agianst "Stores"

plt.figure(figsize=(18,6))
walmartsales.Weekly_Sales.groupby(walmartsales.Store).var().plot(kind='bar')
plt.xlabel("Stores")
plt.ylabel("Variance of Weekly_Sales")
```

Out[12]:
```
Text(0, 0.5, 'Variance of Weekly_Sales')
```




In [13]:
```
# Coefficient of mean to standard deviation

# Bar plot showing "CoV" agianst "Stores"

plt.figure(figsize=(18,6))
storeax=sns.barplot(x='Store',y='CoV',data=maxstd)
storeax.bar_label(storeax.containers[0]);
```




# Store/s having good quarterly growth rate in Q3'2012.

In [14]:
```
# Extracting Year, Month and Week from date column

walmartsales['Date'] = pd.to_datetime(walmartsales.Date,format='%d-%m-%Y')

walmartsales['Year'], walmartsales['Month'], walmartsales['Week'] = walmartsales['Date'
walmartsales
```

Out[14]:

| | Store | Date | Weekly_Sales | Holiday_Flag | Temperature | Fuel_Price | CPI | Unemployment | Y |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2010-02-05 | 1643690.90 | 0 | 42.31 | 2.572 | 211.096358 | 8.106 | 2 |
| **1** | 1 | 2010-02-12 | 1641957.44 | 1 | 38.51 | 2.548 | 211.242170 | 8.106 | 2 |
| **2** | 1 | 2010-02-19 | 1611968.17 | 0 | 39.93 | 2.514 | 211.289143 | 8.106 | 2 |
| **3** | 1 | 2010-02-26 | 1409727.59 | 0 | 46.63 | 2.561 | 211.319643 | 8.106 | 2 |
| **4** | 1 | 2010-03-05 | 1554806.68 | 0 | 46.50 | 2.625 | 211.350143 | 8.106 | 2 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **6430** | 45 | 2012-09-28 | 713173.95 | 0 | 64.88 | 3.997 | 192.013558 | 8.684 | 2 |
| **6431** | 45 | 2012-10-05 | 733455.07 | 0 | 64.89 | 3.985 | 192.170412 | 8.667 | 2 |
| **6432** | 45 | 2012-10-12 | 734464.36 | 0 | 54.47 | 4.000 | 192.327265 | 8.667 | 2 |
| **6433** | 45 | 2012-10-19 | 718125.53 | 0 | 56.47 | 3.969 | 192.330854 | 8.667 | 2 |
| **6434** | 45 | 2012-10-26 | 760281.43 | 0 | 58.85 | 3.882 | 192.308899 | 8.667 | 2 |

6435 rows × 11 columns

In [15]:
```
# Defining the start and end date of Q3 and Q2

Q3_date_from = pd.Timestamp(date(2012,7,1))
Q3_date_to = pd.Timestamp(date(2012,9,30))

Q2_date_from = pd.Timestamp(date(2012,4,1))
Q2_date_to = pd.Timestamp(date(2012,6,30))

# Collecting the data of Q3 and Q2 from original dataset.

Q3data=walmartsales[(walmartsales['Date'] >= Q3_date_from) & (walmartsales['Date'] <= Q
Q2data=walmartsales[(walmartsales['Date'] >= Q2_date_from) & (walmartsales['Date'] <= Q
```

In [16]:
```
# Finding the sum weekly sales of each store in Q3

Q3 = pd.DataFrame(Q3data.groupby('Store')['Weekly_Sales'].sum())
Q3.reset_index(inplace=True)
Q3.rename(columns={'Weekly_Sales': 'Q3_Weekly_Sales'},inplace=True)

# Finding the sum weekly sales of each store in Q2

Q2 = pd.DataFrame(Q2data.groupby('Store')['Weekly_Sales'].sum())
Q2.reset_index(inplace=True)
Q2.rename(columns={'Weekly_Sales': 'Q2_Weekly_Sales'},inplace=True)

# Mergeing Q2 and Q3 data on Store as a common column

Q3_Growth= Q2.merge(Q3,how='inner',on='Store')
```

In [17]:
```
Q3_Growth.head(3)
```

Out[17]:

| | Store | Q2_Weekly_Sales | Q3_Weekly_Sales |
|---|---|---|---|
| **0** | 1 | 20978760.12 | 20253947.78 |
| **1** | 2 | 25083604.88 | 24303354.86 |
| **2** | 3 | 5620316.49 | 5298005.47 |

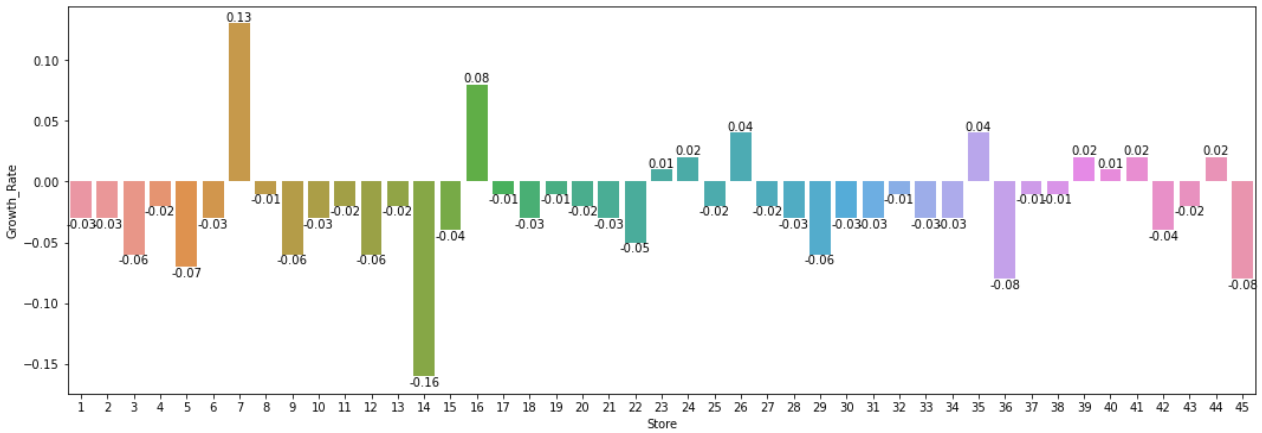
In [18]:
```
# Calculating Growth rate of each Store and collecting it into a dataframe

# Growth rate = ((Present value – Past value )/Past value )*100

Q3_Growth['Growth_Rate'] =(Q3_Growth['Q3_Weekly_Sales'] - Q3_Growth['Q2_Weekly_Sales'])
Q3_Growth['Growth_Rate']=round(Q3_Growth['Growth_Rate'],2)
Q3_Growth.head()
```

Out[18]:

| | Store | Q2_Weekly_Sales | Q3_Weekly_Sales | Growth_Rate |
|---|---|---|---|---|
| **0** | 1 | 20978760.12 | 20253947.78 | -0.03 |

| | Store | Q2_Weekly_Sales | Q3_Weekly_Sales | Growth_Rate |
|---|---|---|---|---|
| **1** | 2 | 25083604.88 | 24303354.86 | -0.03 |
| **2** | 3 | 5620316.49 | 5298005.47 | -0.06 |
| **3** | 4 | 28454363.67 | 27796792.46 | -0.02 |
| **4** | 5 | 4466363.69 | 4163790.99 | -0.07 |

In [19]:
```
# Bar plot showing "Growth_Rate" agianst "Stores"

plt.figure(figsize=(18,6))
storebx=sns.barplot(x='Store',y='Growth_Rate',data=Q3_Growth)
storebx.bar_label(storebx.containers[0]);
```




In [20]:
```
# Finding the store with highest Growth_Rate.

Q3_Growth.sort_values('Growth_Rate',ascending=False).head(1)

# Store 7 has made the highest growth.
```

Out[20]:

| | Store | Q2_Weekly_Sales | Q3_Weekly_Sales | Growth_Rate |
|---|---|---|---|---|
| **6** | 7 | 7290859.27 | 8262787.39 | 0.13 |

In [21]:
```
# Finding the store with lowest Growth_Rate.

Q3_Growth.sort_values('Growth_Rate',ascending=True).head(1)

# Store 14 has made the lowest growth.
```

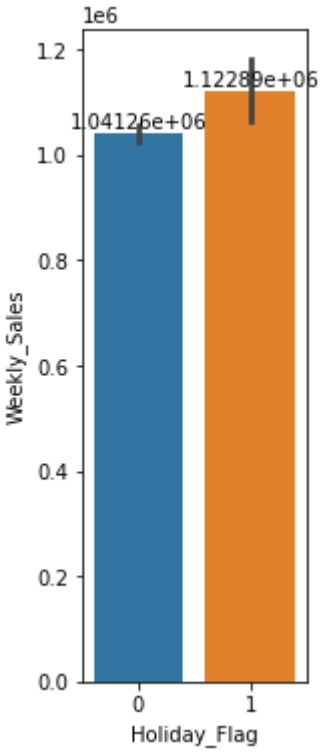
Out[21]:

| | Store | Q2_Weekly_Sales | Q3_Weekly_Sales | Growth_Rate |
|---|---|---|---|---|
| **13** | 14 | 25155535.41 | 21187560.65 | -0.16 |

In [22]:
```
# Finding the mean sales of non holiday and holiday.

walmartsales.groupby('Holiday_Flag')['Weekly_Sales'].mean()
```

Out[22]:
```
Holiday_Flag
0    1.041256e+06
1    1.122888e+06
Name: Weekly_Sales, dtype: float64
```

In [23]:
```
# Bar plot showing "Weekly_Sales" agianst "Holiday_Flag"

plt.figure(figsize=(2,6))
storecx=sns.barplot(x='Holiday_Flag',y='Weekly_Sales',data=walmartsales)
storecx.bar_label(storecx.containers[0]);
```




In [24]:
```
# Marking the holiday dates.

Christmas1 = pd.Timestamp(date(2010,12,31))
Christmas2 = pd.Timestamp(date(2011,12,30))
Christmas3 = pd.Timestamp(date(2012,12,28))
Christmas4 = pd.Timestamp(date(2013,12,27))

Thanksgiving1=pd.Timestamp(date(2010,11,26))
Thanksgiving2=pd.Timestamp(date(2011,11,25))
Thanksgiving3=pd.Timestamp(date(2012,11,23))
Thanksgiving4=pd.Timestamp(date(2013,11,29))

LabourDay1=pd.Timestamp(date(2010,9,10))
LabourDay2=pd.Timestamp(date(2011,9,9))
LabourDay3=pd.Timestamp(date(2012,9,7))
LabourDay4=pd.Timestamp(date(2013,9,6))

SuperBowl1=pd.Timestamp(date(2010,2,12))
SuperBowl2=pd.Timestamp(date(2011,2,11))
SuperBowl3=pd.Timestamp(date(2012,2,10))
SuperBowl4=pd.Timestamp(date(2013,2,8))
```

In [25]:
```
# Calculating the mean sales during the holidays.

Christmas_mean_sales=walmartsales[(walmartsales['Date'] == Christmas1) | (walmartsales[
Thanksgiving_mean_sales=walmartsales[(walmartsales['Date'] == Thanksgiving1) | (walmart
LabourDay_mean_sales=walmartsales[(walmartsales['Date'] == LabourDay1) | (walmartsales[
SuperBowl_mean_sales=walmartsales[(walmartsales['Date'] == SuperBowl1) | (walmartsales[
```

In [26]:
```
dict_of_mean_sales = {'Christmas_mean_sales' : round(Christmas_mean_sales['Weekly_Sales
'Thanksgiving_mean_sales': round(Thanksgiving_mean_sales['Weekly_Sales'].mean(),2),
'LabourDay_mean_sales' : round(LabourDay_mean_sales['Weekly_Sales'].mean(),2),
'SuperBowl_mean_sales':round(SuperBowl_mean_sales['Weekly_Sales'].mean(),2),
'Non holiday weekly sales' : walmartsales[walmartsales['Holiday_Flag'] == 0 ]['Weekly_S

dict_of_mean_sales # List of mean sales during the holidays and mean sales during the N

# We can see that during Thanksgiving, mean sales are high than the mean sales during N
```

Out[26]:
```
{'Christmas_mean_sales': 960833.11,
 'Thanksgiving_mean_sales': 1471273.43,
 'LabourDay_mean_sales': 1042427.29,
 'SuperBowl_mean_sales': 1079127.99,
 'Non holiday weekly sales': 1041256.3802088564}
```

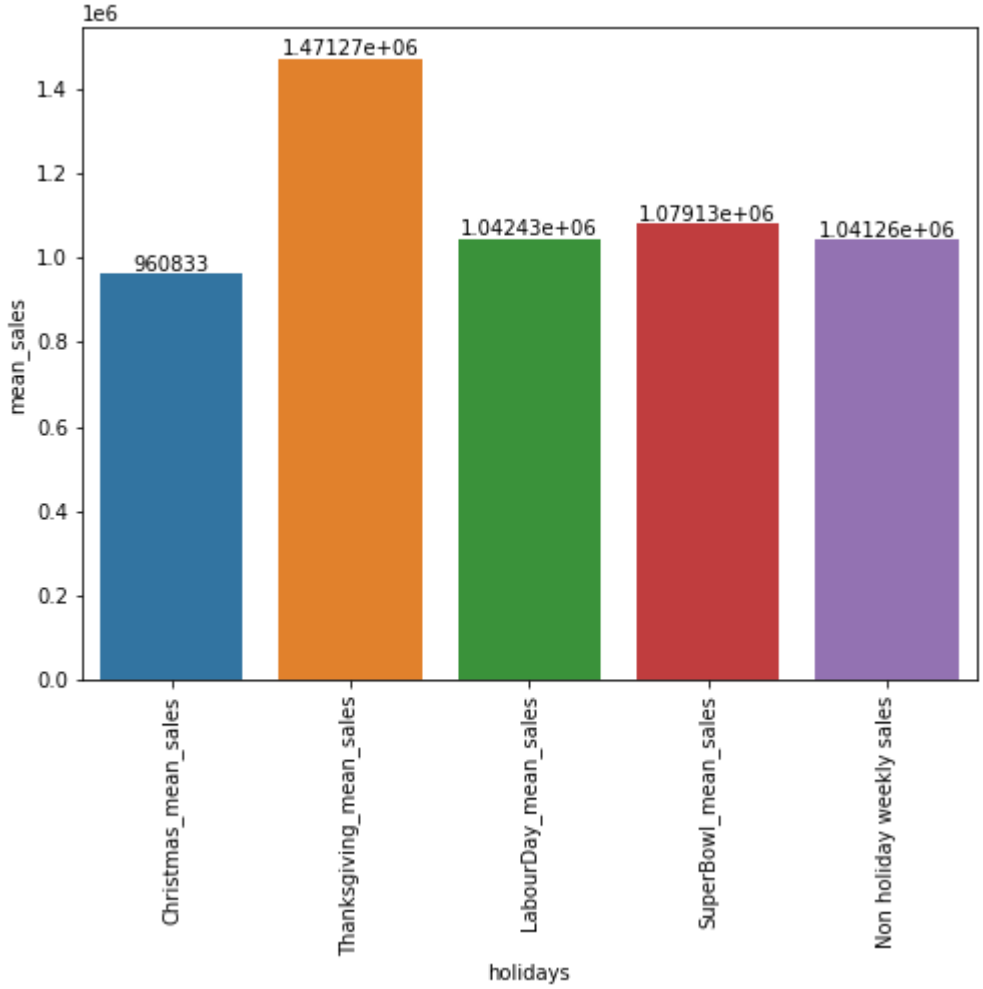
In [27]:
```
mean_sales_during_holidays_Nonholidays=pd.DataFrame(list(dict_of_mean_sales.items()),co
```

In [28]:
```
mean_sales_during_holidays_Nonholidays
```

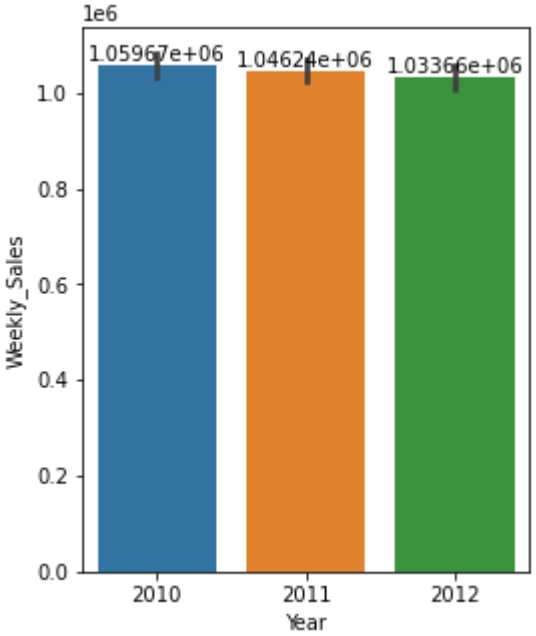
Out[28]:

| | holidays | mean_sales |
|---|---|---|
| **0** | Christmas_mean_sales | 9.608331e+05 |
| **1** | Thanksgiving_mean_sales | 1.471273e+06 |
| **2** | LabourDay_mean_sales | 1.042427e+06 |
| **3** | SuperBowl_mean_sales | 1.079128e+06 |
| **4** | Non holiday weekly sales | 1.041256e+06 |

In [29]:
```
# Bar plot showing mean sales during Holidays and Non Holidays.

plt.figure(figsize=(8,6))
storedx=sns.barplot(x='holidays',y='mean_sales',data=mean_sales_during_holidays_Nonholi
plt.xticks(rotation=90)
storedx.bar_label(storedx.containers[0]);
```

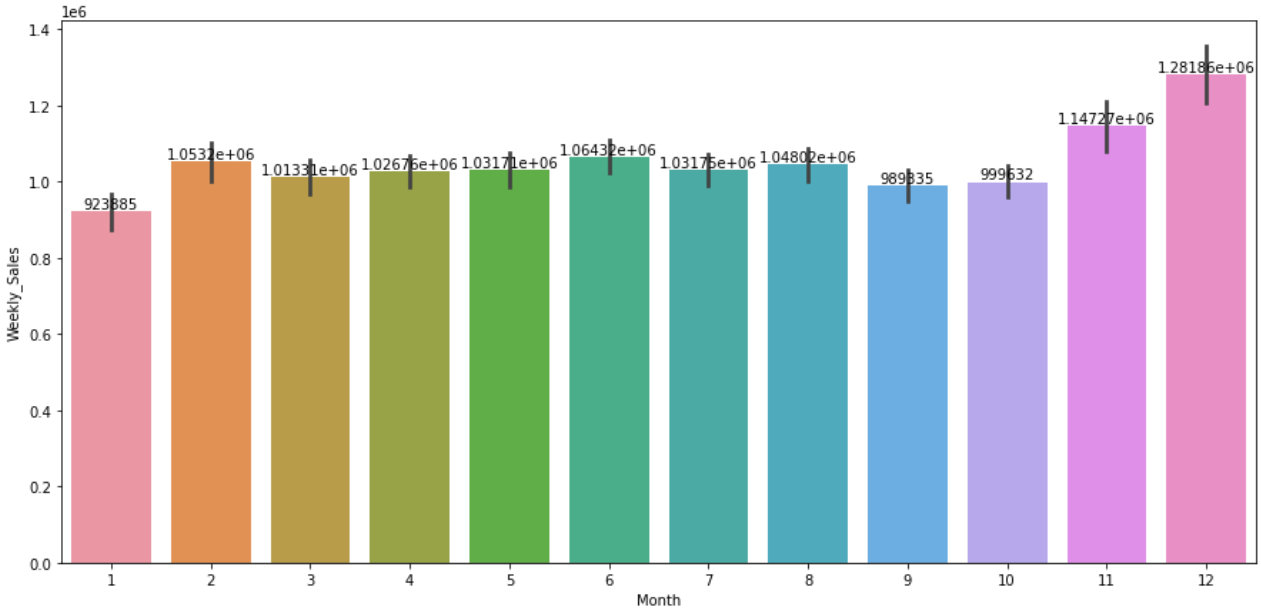



In [30]:
```
plt.figure(figsize=(4,5))
store_ex=sns.barplot(x='Year', y='Weekly_Sales', data=walmartsales); # Year wise averag
store_ex.bar_label(store_ex.containers[0]);
```



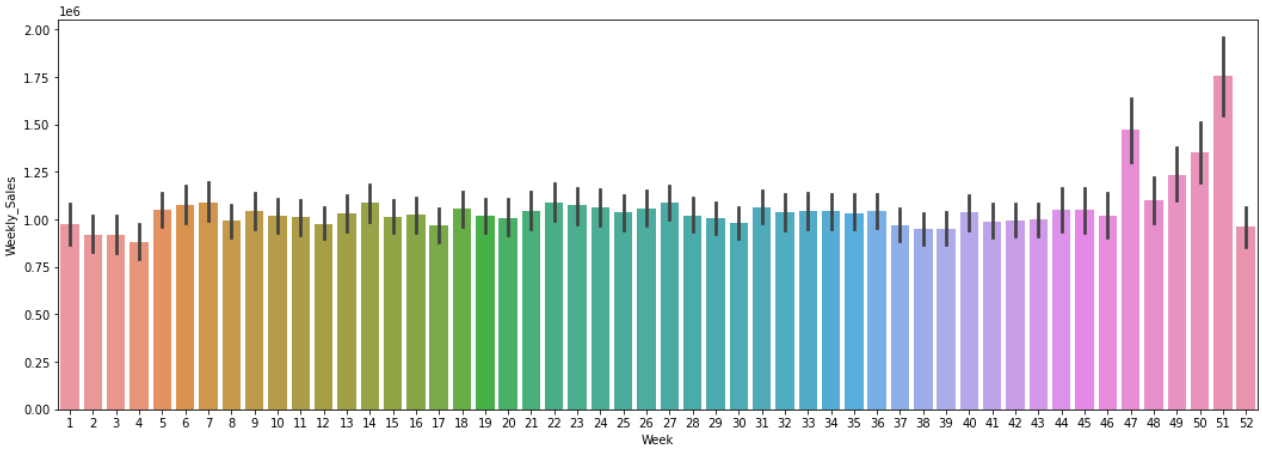

In [31]:
```
plt.figure(figsize=(15,7))
```

```
storefx=sns.barplot(x='Month', y='Weekly_Sales', data=walmartsales); # Month wise avera
storefx.bar_label(storefx.containers[0]);
```




In [32]:

```
plt.figure(figsize=(18,6))
sns.barplot(x='Week', y='Weekly_Sales', data=walmartsales); # Week wise average Weekly_
```

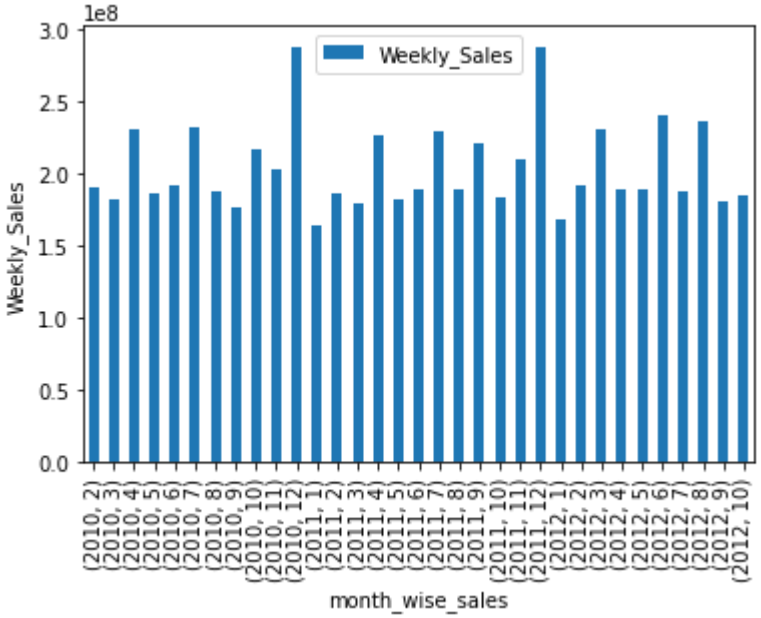



In [33]:

```
# Monthly sales.

Monthly_sales = walmartsales.groupby(['Year','Month']) \
.agg(Weekly_Sales = ('Weekly_Sales', 'sum')).plot(kind='bar')

plt.xlabel("month_wise_sales")
plt.ylabel("Weekly_Sales")
```

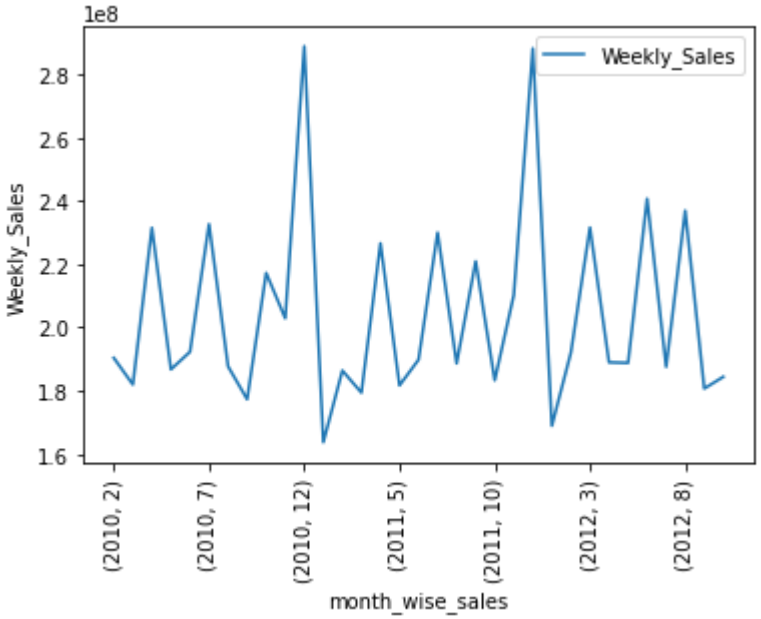
Out[33]:

```
Text(0, 0.5, 'Weekly_Sales')
```




In [34]:
```
# Monthly sales.

Monthly_sales = walmartsales.groupby(['Year','Month']) \
.agg(Weekly_Sales = ('Weekly_Sales', 'sum')).plot(kind='line')

plt.xlabel("month_wise_sales")
plt.xticks(rotation=90)
plt.ylabel("Weekly_Sales")

# We can observe from the Monthly Sales Graph that highest sum of sales is recorded in
```

Out[34]:
```
Text(0, 0.5, 'Weekly_Sales')
```




In [35]:
```
# using the to_period function
#walmartsales['quarter'] = walmartsales['Date'].dt.to_period('Q')
walmartsales['quarter'] = walmartsales['Date'].dt.quarter
```

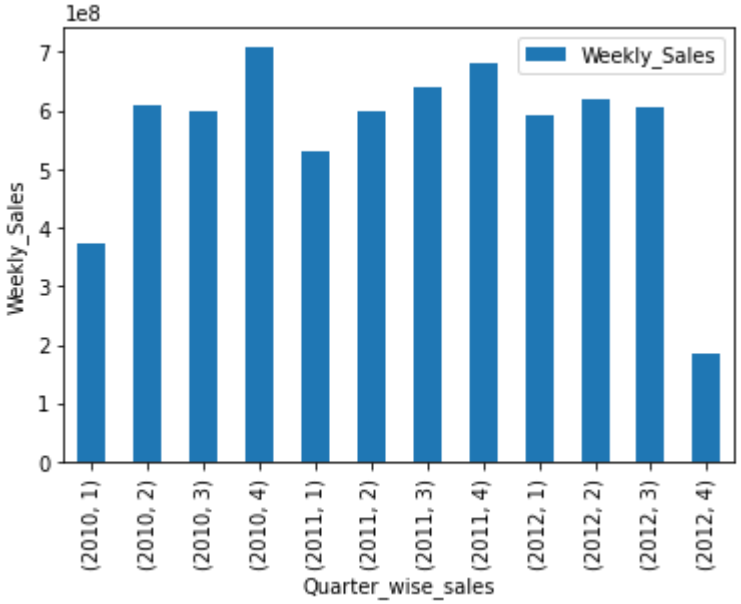
In [36]:
```
walmartsales.head(5)
```

Out[36]:

| | Store | Date | Weekly_Sales | Holiday_Flag | Temperature | Fuel_Price | CPI | Unemployment | Year |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2010-02-05 | 1643690.90 | 0 | 42.31 | 2.572 | 211.096358 | 8.106 | 2010 |
| **1** | 1 | 2010-02-12 | 1641957.44 | 1 | 38.51 | 2.548 | 211.242170 | 8.106 | 2010 |
| **2** | 1 | 2010-02-19 | 1611968.17 | 0 | 39.93 | 2.514 | 211.289143 | 8.106 | 2010 |
| **3** | 1 | 2010-02-26 | 1409727.59 | 0 | 46.63 | 2.561 | 211.319643 | 8.106 | 2010 |
| **4** | 1 | 2010-03-05 | 1554806.68 | 0 | 46.50 | 2.625 | 211.350143 | 8.106 | 2010 |

In [37]:
```
# Quarterly sales.

Quarter_sales = walmartsales.groupby(['Year','quarter']) \
.agg(Weekly_Sales = ('Weekly_Sales', 'sum')).plot(kind='bar')

plt.xlabel("Quarter_wise_sales")
plt.xticks(rotation=90)
plt.ylabel("Weekly_Sales")
```

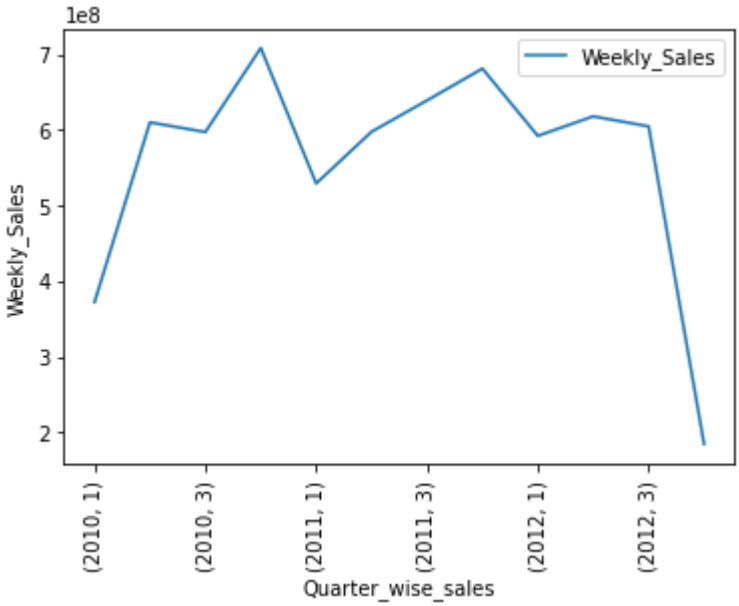
Out[37]:
```
Text(0, 0.5, 'Weekly_Sales')
```




In [38]:
```
# Quarterly sales.

Quarter_sales = walmartsales.groupby(['Year','quarter']) \
.agg(Weekly_Sales = ('Weekly_Sales', 'sum')).plot(kind='line')

plt.xlabel("Quarter_wise_sales")
plt.xticks(rotation=90)
```

```
plt.ylabel("Weekly_Sales")

# We can observe from the Quarterly Sales Graph that higest sum of sales is recorded in
```

Out[38]:
```
Text(0, 0.5, 'Weekly_Sales')
```




In [39]:
```
# using the to_period function
#walmartsales['semester']= walmartsales.Date.dt.year.astype(str) + 'S'+ np.where(walmar
walmartsales['semester']= np.where(walmartsales.Date.dt.quarter.gt(2),2,1).astype(str)
```

In [40]:
```
walmartsales.head(5)
```

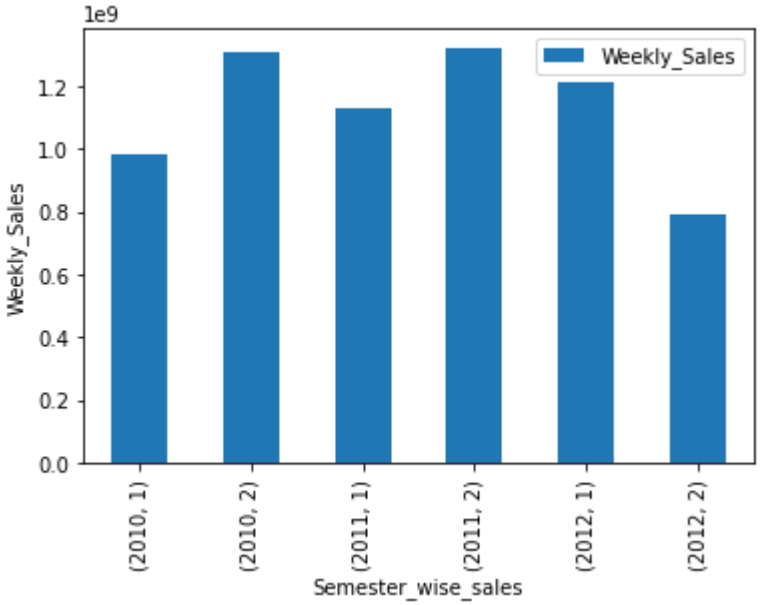
Out[40]:

| | Store | Date | Weekly_Sales | Holiday_Flag | Temperature | Fuel_Price | CPI | Unemployment | Year |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2010-02-05 | 1643690.90 | 0 | 42.31 | 2.572 | 211.096358 | 8.106 | 2010 |
| **1** | 1 | 2010-02-12 | 1641957.44 | 1 | 38.51 | 2.548 | 211.242170 | 8.106 | 2010 |
| **2** | 1 | 2010-02-19 | 1611968.17 | 0 | 39.93 | 2.514 | 211.289143 | 8.106 | 2010 |
| **3** | 1 | 2010-02-26 | 1409727.59 | 0 | 46.63 | 2.561 | 211.319643 | 8.106 | 2010 |
| **4** | 1 | 2010-03-05 | 1554806.68 | 0 | 46.50 | 2.625 | 211.350143 | 8.106 | 2010 |

In [41]:
```
# Semester sales.

Semester_sales = walmartsales.groupby(['Year','semester']) \
.agg(Weekly_Sales = ('Weekly_Sales', 'sum')).plot(kind='bar')

plt.xlabel("Semester_wise_sales")
```

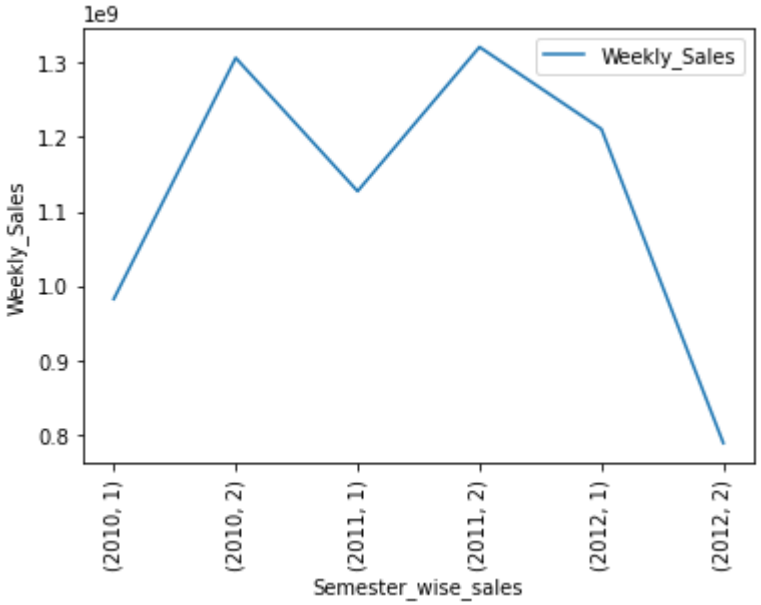
```
plt.xticks(rotation=90)
plt.ylabel("Weekly_Sales")
```

Out[41]:
```
Text(0, 0.5, 'Weekly_Sales')
```




In [42]:
```
# Semester sales.

Semester_sales = walmartsales.groupby(['Year','semester']) \
.agg(Weekly_Sales = ('Weekly_Sales', 'sum')).plot(kind='line')

plt.xlabel("Semester_wise_sales")
plt.xticks(rotation=90)
plt.ylabel("Weekly_Sales")

# We can Observe from Semester graph that at end of 2nd sem of 2011 sales are Highest.
```

Out[42]:
```
Text(0, 0.5, 'Weekly_Sales')
```




In [43]:

```
walmartsales.head()
```

Out[43]:

| | Store | Date | Weekly_Sales | Holiday_Flag | Temperature | Fuel_Price | CPI | Unemployment | Year |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2010-02-05 | 1643690.90 | 0 | 42.31 | 2.572 | 211.096358 | 8.106 | 2010 |
| **1** | 1 | 2010-02-12 | 1641957.44 | 1 | 38.51 | 2.548 | 211.242170 | 8.106 | 2010 |
| **2** | 1 | 2010-02-19 | 1611968.17 | 0 | 39.93 | 2.514 | 211.289143 | 8.106 | 2010 |
| **3** | 1 | 2010-02-26 | 1409727.59 | 0 | 46.63 | 2.561 | 211.319643 | 8.106 | 2010 |
| **4** | 1 | 2010-03-05 | 1554806.68 | 0 | 46.50 | 2.625 | 211.350143 | 8.106 | 2010 |

# Hypothesis of Factors like CPI, Unemployment and Fuel_price on Weekly_Sales, Creating a Day Column.

# Statistical Modelling For Store 1

In [44]:

```
#Let's Group the data.

hypothesis = walmartsales.groupby('Store')[['Fuel_Price','Unemployment', 'CPI','Weekly_
factors  = hypothesis.get_group(1)
day_arr = [1]
for i in range (1,len(factors)):
    day_arr.append(i*7)

factors['Day'] = day_arr.copy()
```

```
C:\Users\gkoppadx\AppData\Local\Temp/ipykernel_4940/4263428844.py:9: SettingWithCopyWarn
ing:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  factors['Day'] = day_arr.copy()
```

In [45]:

```
factors.head()
```

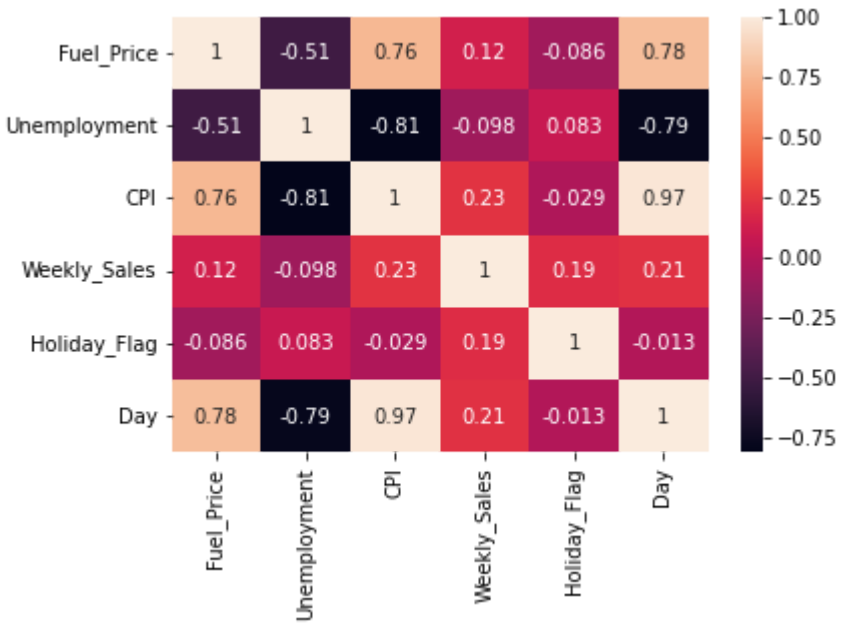
Out[45]:

| | Fuel_Price | Unemployment | CPI | Weekly_Sales | Holiday_Flag | Day |
|---|---|---|---|---|---|---|
| **0** | 2.572 | 8.106 | 211.096358 | 1643690.90 | 0 | 1 |
| **1** | 2.548 | 8.106 | 211.242170 | 1641957.44 | 1 | 7 |

| | Fuel_Price | Unemployment | CPI | Weekly_Sales | Holiday_Flag | Day |
|---|---|---|---|---|---|---|
| **2** | 2.514 | 8.106 | 211.289143 | 1611968.17 | 0 | 14 |
| **3** | 2.561 | 8.106 | 211.319643 | 1409727.59 | 0 | 21 |
| **4** | 2.625 | 8.106 | 211.350143 | 1554806.68 | 0 | 28 |

In [46]:
```
sns.heatmap(factors.corr(), annot = True)
```

Out[46]:
```
<AxesSubplot:>
```




In [47]:
```
# By looking at the heatmap we can conclude that CPI and Holiday_Flag is fairly strongl
```

# Hypothesis of CPI, FuelPrice, Unemployment with Weekly_Sales.

In [48]:
```
# Hypothesis Testing - CPI

from scipy import stats
ttest,pval = stats.ttest_rel(factors['Weekly_Sales'],factors['CPI'])

print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
3.106725927640744e-144
reject null hypothesis
```

In [49]:
```
# Hypothesis Testing - Fuel_Price
```

```
ttest,pval = stats.ttest_rel(factors['Weekly_Sales'],factors['Fuel_Price'])

print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
3.050079726743709e-144
reject null hypothesis
```

In [50]:

```
# Hypothesis Testing - Uneployment

ttest,pval = stats.ttest_rel(factors['Weekly_Sales'],factors['Unemployment'])

print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
3.0515405336011733e-144
reject null hypothesis
```

# Linear Regression Model

In [51]:

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

In [52]:

```
factors
```

Out[52]:

| | Fuel_Price | Unemployment | CPI | Weekly_Sales | Holiday_Flag | Day |
|---|---|---|---|---|---|---|
| **0** | 2.572 | 8.106 | 211.096358 | 1643690.90 | 0 | 1 |
| **1** | 2.548 | 8.106 | 211.242170 | 1641957.44 | 1 | 7 |
| **2** | 2.514 | 8.106 | 211.289143 | 1611968.17 | 0 | 14 |
| **3** | 2.561 | 8.106 | 211.319643 | 1409727.59 | 0 | 21 |
| **4** | 2.625 | 8.106 | 211.350143 | 1554806.68 | 0 | 28 |
| **...** | ... | ... | ... | ... | ... | ... |
| **138** | 3.666 | 6.908 | 222.981658 | 1437059.26 | 0 | 966 |
| **139** | 3.617 | 6.573 | 223.181477 | 1670785.97 | 0 | 973 |
| **140** | 3.601 | 6.573 | 223.381296 | 1573072.81 | 0 | 980 |
| **141** | 3.594 | 6.573 | 223.425723 | 1508068.77 | 0 | 987 |
| **142** | 3.506 | 6.573 | 223.444251 | 1493659.74 | 0 | 994 |

143 rows × 6 columns

In [53]:
```
walmartsales
```

Out[53]:

| | Store | Date | Weekly_Sales | Holiday_Flag | Temperature | Fuel_Price | CPI | Unemployment | Y |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2010-02-05 | 1643690.90 | 0 | 42.31 | 2.572 | 211.096358 | 8.106 | 2 |
| **1** | 1 | 2010-02-12 | 1641957.44 | 1 | 38.51 | 2.548 | 211.242170 | 8.106 | 2 |
| **2** | 1 | 2010-02-19 | 1611968.17 | 0 | 39.93 | 2.514 | 211.289143 | 8.106 | 2 |
| **3** | 1 | 2010-02-26 | 1409727.59 | 0 | 46.63 | 2.561 | 211.319643 | 8.106 | 2 |
| **4** | 1 | 2010-03-05 | 1554806.68 | 0 | 46.50 | 2.625 | 211.350143 | 8.106 | 2 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **6430** | 45 | 2012-09-28 | 713173.95 | 0 | 64.88 | 3.997 | 192.013558 | 8.684 | 2 |
| **6431** | 45 | 2012-10-05 | 733455.07 | 0 | 64.89 | 3.985 | 192.170412 | 8.667 | 2 |
| **6432** | 45 | 2012-10-12 | 734464.36 | 0 | 54.47 | 4.000 | 192.327265 | 8.667 | 2 |
| **6433** | 45 | 2012-10-19 | 718125.53 | 0 | 56.47 | 3.969 | 192.330854 | 8.667 | 2 |
| **6434** | 45 | 2012-10-26 | 760281.43 | 0 | 58.85 | 3.882 | 192.308899 | 8.667 | 2 |

6435 rows × 13 columns

In [54]:
```
# For Store 1

walmartsales['Day']=factors['Day']
walmartsales_1=walmartsales[(walmartsales.Store == 1)]
walmartsales_1
```

Out[54]:

| | Store | Date | Weekly_Sales | Holiday_Flag | Temperature | Fuel_Price | CPI | Unemployment | Ye |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2010-02-05 | 1643690.90 | 0 | 42.31 | 2.572 | 211.096358 | 8.106 | 20 |
| **1** | 1 | 2010-02-12 | 1641957.44 | 1 | 38.51 | 2.548 | 211.242170 | 8.106 | 20 |
| **2** | 1 | 2010-02-19 | 1611968.17 | 0 | 39.93 | 2.514 | 211.289143 | 8.106 | 20 |

| | Store | Date | Weekly_Sales | Holiday_Flag | Temperature | Fuel_Price | CPI | Unemployment | Ye |
|---|---|---|---|---|---|---|---|---|---|
| **3** | 1 | 2010-02-26 | 1409727.59 | 0 | 46.63 | 2.561 | 211.319643 | 8.106 | 20 |
| **4** | 1 | 2010-03-05 | 1554806.68 | 0 | 46.50 | 2.625 | 211.350143 | 8.106 | 20 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **138** | 1 | 2012-09-28 | 1437059.26 | 0 | 76.08 | 3.666 | 222.981658 | 6.908 | 20 |
| **139** | 1 | 2012-10-05 | 1670785.97 | 0 | 68.55 | 3.617 | 223.181477 | 6.573 | 20 |
| **140** | 1 | 2012-10-12 | 1573072.81 | 0 | 62.99 | 3.601 | 223.381296 | 6.573 | 20 |
| **141** | 1 | 2012-10-19 | 1508068.77 | 0 | 67.97 | 3.594 | 223.425723 | 6.573 | 20 |
| **142** | 1 | 2012-10-26 | 1493659.74 | 0 | 69.16 | 3.506 | 223.444251 | 6.573 | 20 |

143 rows × 14 columns

In [55]:
```
# Remove extra added columns

walmartsales_1 = walmartsales_1.drop(['Year','Month','Week','quarter','semester'], axis
walmartsales_1.head(3)
```

Out[55]:

| | Store | Date | Weekly_Sales | Holiday_Flag | Temperature | Fuel_Price | CPI | Unemployment | Day |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2010-02-05 | 1643690.90 | 0 | 42.31 | 2.572 | 211.096358 | 8.106 | 1.0 |
| **1** | 1 | 2010-02-12 | 1641957.44 | 1 | 38.51 | 2.548 | 211.242170 | 8.106 | 7.0 |
| **2** | 1 | 2010-02-19 | 1611968.17 | 0 | 39.93 | 2.514 | 211.289143 | 8.106 | 14.0 |

In [56]:
```
# Setup data
X = walmartsales_1.drop(['Weekly_Sales', 'Date'], axis=1)
y = walmartsales_1['Weekly_Sales']

## Split dataset into training and test set
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, random_state=1

# Fitting data to multiple Linear Regression
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[56]:
```
LinearRegression()
```

In [57]:
```
## Check out the score
regressor.score(X_test, y_test)
```

Out[57]:
```
0.051433013580189924
```

In [58]:
```
# Predict test result
y_pred = regressor.predict(X_test)
y_pred
```

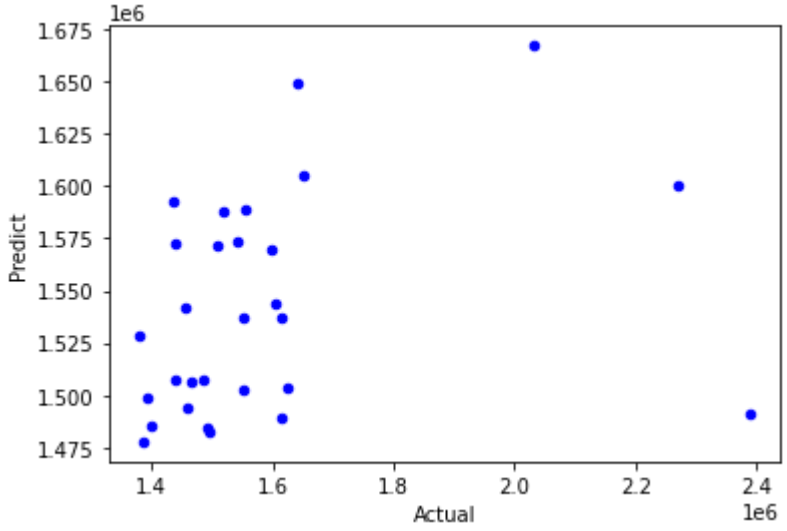
Out[58]:
```
array([1537030.77549693, 1542313.74108229, 1600504.4168272 ,
       1484449.68714447, 1506869.33257533, 1571809.11025818,
       1528110.97658201, 1494492.47724498, 1507666.7674199 ,
       1588580.83361967, 1491621.87947826, 1605106.75546803,
       1543691.82729157, 1507153.71929303, 1573895.00153055,
       1572351.0899279 , 1666951.35128518, 1569776.91859298,
       1503942.15612531, 1499041.1615855 , 1489032.9980949 ,
       1592436.55087211, 1649282.41614805, 1588225.59188752,
       1485542.30433994, 1502761.62125991, 1483010.89160762,
       1477888.90990766, 1537421.69997719])
```

In [59]:
```
## Function to check out the accuracy of the model
def mean_absolute_percentage_error(y_test, y_pred):
    y_test, y_pred = np.array(y_test), np.array(y_pred)
    errors = np.abs(y_test - y_pred)
    mape = np.mean(100 * (errors / y_test))
    print('Mean Absolute Percentage Error:', round(mape, 2), '%.')
    accuracy = 100 - mape
    print('Accuracy:', round(accuracy, 2), '%.')
```

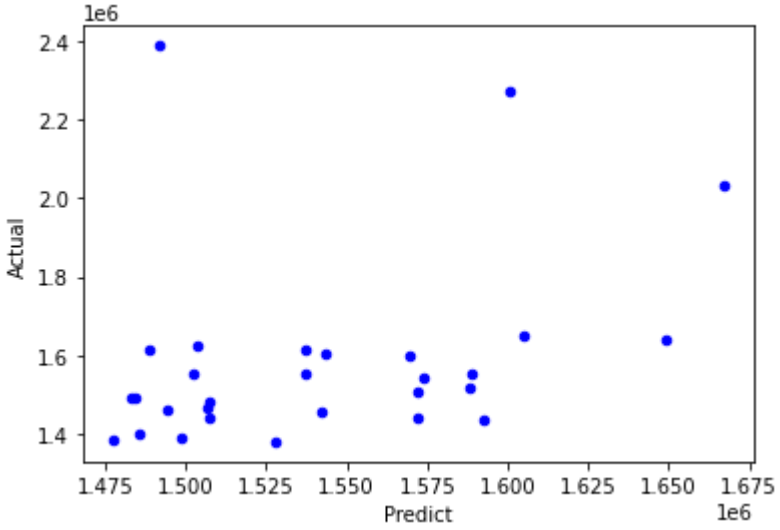
In [60]:
```
## Check out the accuracy of the model
mean_absolute_percentage_error(y_test, y_pred)
```

```
Mean Absolute Percentage Error: 6.95 %.
Accuracy: 93.05 %.
```

In [61]:
```
# Plot the Actual vs predicted values
y_test_pred_df = pd.DataFrame(list(zip(y_test, y_pred)), columns =['Actual', 'Predict']
y_test_pred_df
y_test_pred_df.plot(x="Actual", y="Predict", kind="scatter", color="blue");
```




In [62]:
```
# Plot the predicted vs actual values
y_test_pred_df.plot(x="Predict", y="Actual", kind="scatter", color="blue");
```




In [63]:
```
y_test_pred_df
```

Out[63]:

| | Actual | Predict |
|---|---|---|
| **0** | 1553191.63 | 1.537031e+06 |
| **1** | 1455090.69 | 1.542314e+06 |
| **2** | 2270188.99 | 1.600504e+06 |
| **3** | 1492418.14 | 1.484450e+06 |
| **4** | 1464693.46 | 1.506869e+06 |
| **5** | 1507460.69 | 1.571809e+06 |
| **6** | 1380020.27 | 1.528111e+06 |
| **7** | 1459409.10 | 1.494492e+06 |
| **8** | 1483784.18 | 1.507667e+06 |

| | Actual | Predict |
|---|---|---|
| **9** | 1555444.55 | 1.588581e+06 |
| **10** | 2387950.20 | 1.491622e+06 |
| **11** | 1649604.63 | 1.605107e+06 |
| **12** | 1604775.58 | 1.543692e+06 |
| **13** | 1438830.15 | 1.507154e+06 |
| **14** | 1540421.49 | 1.573895e+06 |
| **15** | 1439123.71 | 1.572351e+06 |
| **16** | 2033320.66 | 1.666951e+06 |
| **17** | 1597868.05 | 1.569777e+06 |
| **18** | 1624383.75 | 1.503942e+06 |
| **19** | 1391256.12 | 1.499041e+06 |
| **20** | 1615524.71 | 1.489033e+06 |
| **21** | 1437059.26 | 1.592437e+06 |
| **22** | 1641957.44 | 1.649282e+06 |
| **23** | 1517428.87 | 1.588226e+06 |
| **24** | 1399662.07 | 1.485542e+06 |
| **25** | 1551659.28 | 1.502762e+06 |
| **26** | 1494251.50 | 1.483011e+06 |
| **27** | 1385065.20 | 1.477889e+06 |
| **28** | 1614259.35 | 1.537422e+06 |

In [ ]: