In [2]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
df=pd.read_csv("Walmart_Store_sales.csv")
```

In [4]:
```
df.head(10)
```

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
5	1	12-03-2010	1439541.59	0	57.79	2.667	211.380643	8.106
6	1	19-03-2010	1472515.79	0	54.58	2.720	211.215635	8.106
7	1	26-03-2010	1404429.92	0	51.45	2.732	211.018042	8.106
8	1	02-04-2010	1594968.28	0	62.27	2.719	210.820450	7.808
9	1	09-04-2010	1545418.53	0	65.86	2.770	210.622857	7.808

In [5]:
```
df.describe()
```

Out[5]:

	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 [6]:
```
df.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 [7]:
```
df.dtypes
```

Out[7]:
```
Store             int64
Date             object
Weekly_Sales    float64
Holiday_Flag      int64
Temperature     float64
Fuel_Price      float64
CPI             float64
Unemployment    float64
dtype: object
```

Which store has maximum sale?

In [13]:
```
retail_data=df
```

In [14]:
```
print(df[df.Weekly_Sales == df.Weekly_Sales.max()])
```

```
      Store        Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
1905     14  24-12-2010    3818686.45             0        30.59       3.141   

            CPI  Unemployment quarter
1905  182.54459         8.724  2010Q4
```

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

In [15]:
```
maxstd=pd.DataFrame(df.groupby('Store').agg({'Weekly_Sales':['std','mean']}))
#Just resetting the index.
maxstd = maxstd.reset_index()
# coefficient of Variance = mean / standard Deviation
maxstd['COV'] = (maxstd[('Weekly_Sales','std')]/maxstd[('Weekly_Sales','mean')]) *100
maxstd.loc[maxstd[('Weekly_Sales','std')]==maxstd[('Weekly_Sales','std')].max()]
```

Out[15]:

	Store	Weekly_Sales	COV

		std	mean	
13	14	317569.949476	2.020978e+06	15.713674

Which store has a good quarterly growth rate?

In [16]:
```
# create a new column which shows the year and quarter
df['quarter'] = pd.PeriodIndex(df.Date, freq='Q')
T2012Q2 = df.loc[df['quarter'] == "2012Q2", ["Weekly_Sales", 'Store']]
T2012Q3 = df.loc[df['quarter'] == "2012Q3", ["Weekly_Sales", 'Store']]
T2012Q2_sum_per_store = pd.DataFrame(T2012Q2.groupby('Store')['Weekly_Sales'].sum())
T2012Q2_sum_per_store.reset_index(inplace=True)
T2012Q3_sum_per_store = pd.DataFrame(T2012Q3.groupby('Store')['Weekly_Sales'].sum())
T2012Q3_sum_per_store.reset_index(inplace=True)
T2012Q2_sum_per_store['Weekly_Sales_Q3'] = T2012Q3_sum_per_store['Weekly_Sales']
T2012Q2_sum_per_store['Growth Rate'] = ((T2012Q2_sum_per_store.Weekly_Sales_Q3 - T2012Q2_sum_per_store.Weekly_Sales)/T2012Q2_sum_per_store.Weekly_Sales)*100
T2012Q2_sum_per_store.loc[T2012Q2_sum_per_store['Growth Rate']==T2012Q2_sum_per_store['Growth Rate'].max()]
```

Out[16]:

	Store	Weekly_Sales	Weekly_Sales_Q3	Growth Rate
15	16	6626133.44	6441311.11	-2.789294

Find out the holiday that has the higher sales than the mean sales in non-holiday season II together

In [20]:
```
import datetime as dt
retail_data.Date = pd.to_datetime(retail_data.Date)
```

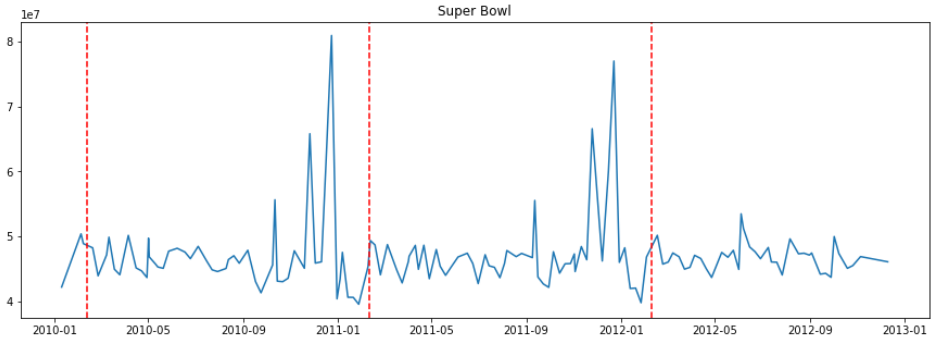
In [21]:
```
Holiday_Week = retail_data.loc[retail_data['Holiday_Flag']==1]
```

In [22]:
```
from matplotlib import pyplot as plt
import time
retail_data['Date'] = pd.to_datetime(retail_data.Date)
```

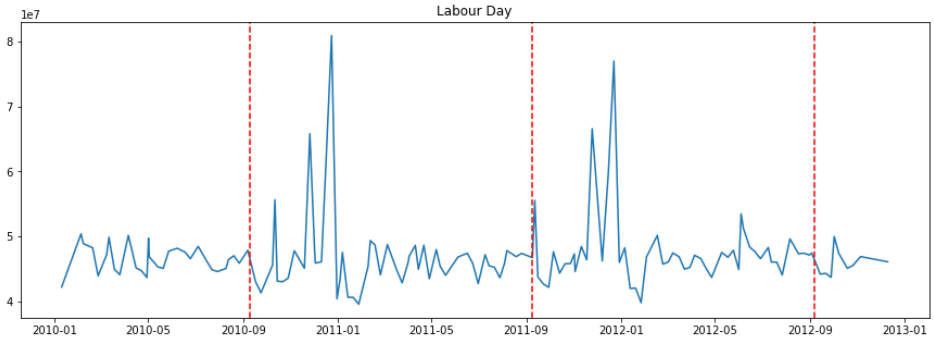
In [23]:
```
def plot_line(df, holiday_dates, holiday_label):
    fig, ax = plt.subplots(figsize=(15,5))
    ax.plot(df['Date'], df['Weekly_Sales'], label=holiday_label)

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

    plt.title(holiday_label)
    x_dates = df['Date'].dt.strftime('%Y-%m-%d').sort_values().unique()
    plt.show()
```

In [24]:
```
total_sales = retail_data.groupby('Date')['Weekly_Sales'].sum().reset_index()
Super_Bowl=['12-2-2010', '11-2-2011', '10-2-2012']
Labor_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']
```

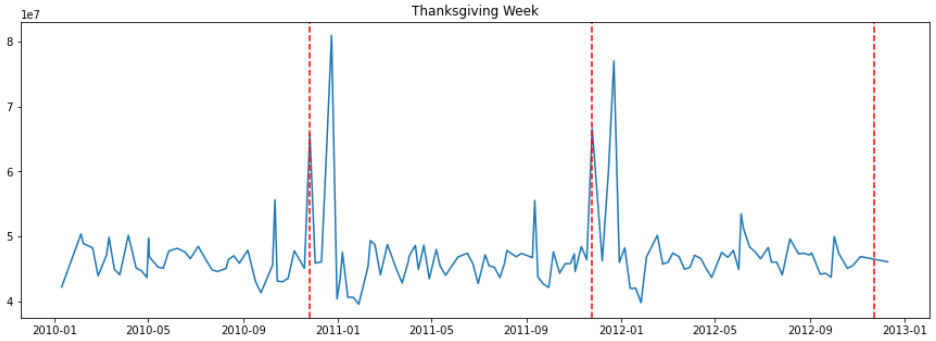
In [25]:
```
plot_line(total_sales, Super_Bowl, 'Super Bowl')
```



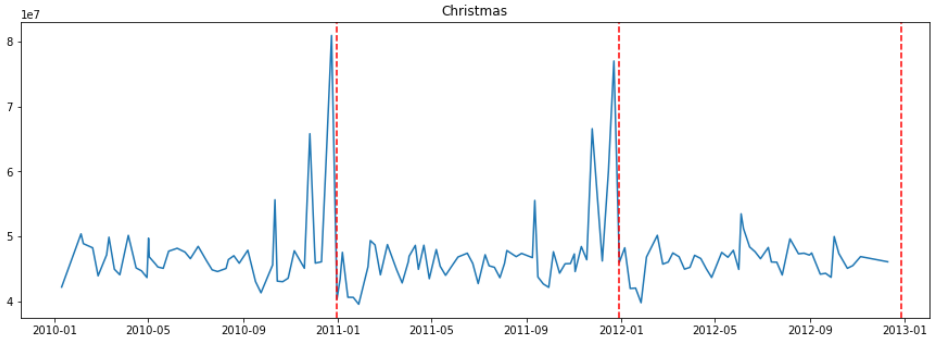
In [26]:
```
plot_line(total_sales, Labor_Day, 'Labour Day')
```



In [27]:
```
plot_line(total_sales, Thanksgiving, 'Thanksgiving Week')
```

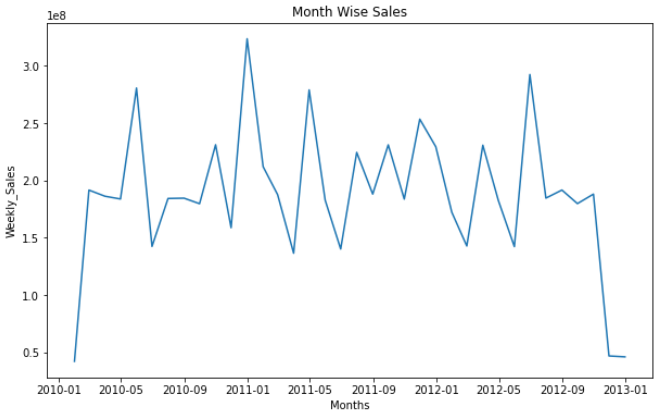


In [28]:
```
plot_line(total_sales, Christmas, 'Christmas')
```



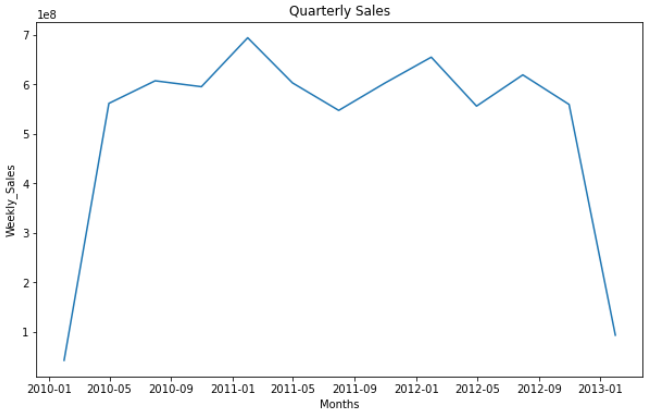
In [29]:
```
from matplotlib import pyplot as plt
pd.DatetimeIndex(retail_data['Date'])
monthly = retail_data.groupby(pd.Grouper(key='Date', freq='1M')).sum()# groupby each 1 month
monthly=monthly.reset_index()
fig, ax = plt.subplots(figsize=(10,6))
X = monthly['Date']
Y = monthly['Weekly_Sales']
plt.plot(X,Y)
plt.title('Month Wise Sales')
plt.xlabel('Months')
plt.ylabel('Weekly_Sales')
```

Out[29]:
```
Text(0, 0.5, 'Weekly_Sales')
```



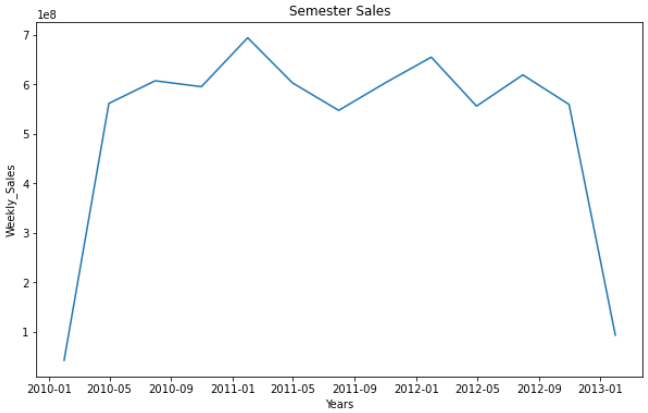
In [30]:
```
quarterly = retail_data.groupby(pd.Grouper(key='Date', freq='3M')).sum()# groupby each 3e month
quarterly=quarterly.reset_index()
fig, ax = plt.subplots(figsize=(10,6))
X = quarterly['Date']
Y = quarterly['Weekly_Sales']
plt.plot(X,Y)
plt.title('Quarterly Sales')
plt.xlabel('Months')
plt.ylabel('Weekly_Sales')
```

Out[30]:
```
Text(0, 0.5, 'Weekly_Sales')
```

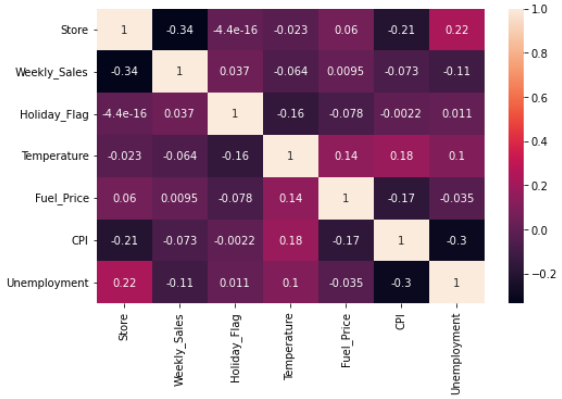


In [31]:
```
semesterly = retail_data.groupby(pd.Grouper(key='Date', freq='3M')).sum()# groupby each 3e month
semesterly=semesterly.reset_index()
fig, ax = plt.subplots(figsize=(10,6))
X = semesterly['Date']
Y = semesterly['Weekly_Sales']
plt.plot(X,Y)
plt.title('Semester Sales')
plt.xlabel('Years')
plt.ylabel('Weekly_Sales')
```

Out[31]:
```
Text(0, 0.5, 'Weekly_Sales')
```



In [32]:
```
# A little feature engineering
import seaborn as sns
corr = retail_data.corr()

plt.figure(figsize=(8, 5))
sns.heatmap(corr, annot=True)
plt.show()
```



In [33]:
```
retail_data['Day']=retail_data['Date'].dt.day
retail_data = retail_data.drop('Date',axis=1)
```

In [34]:
```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.metrics import confusion_matrix, r2_score
```

In [35]:
```
df_v2 = pd.get_dummies(retail_data, columns = ['Holiday_Flag','Store'])
y = df_v2['Weekly_Sales']
X= df_v2.drop(['Weekly_Sales', 'quarter'],axis=1)
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
```

In [36]:
```
ln_model = LinearRegression()
ln_model.fit(X_train, y_train)
```

Out[36]:
```
LinearRegression()
```

In [37]:
```
y_pred = ln_model.predict(X_test)
print("r2 score:", r2_score(y_test,y_pred))
```

```
r2 score: 0.9089914384804616
```

Submitted by SAMBIT MAHANTA

In [ ]:
```

```