In [44]:

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

In [7]:

```
df=pd.read_csv(r'C:\desktop\Walmart_Store_sales.csv')
```

In [8]:

```
df.head()
```

Out[8]:

	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 [9]:

```
df.shape
```

Out[9]:

```
(6435, 8)
```

In [10]:

```
df.describe()
```

Out[10]:

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemploy
count	6435.000000	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.000000	6435.0
mean	23.000000	1.046965e+06	0.069930	60.663782	3.358607	171.578394	7.9
std	12.988182	5.643666e+05	0.255049	18.444933	0.459020	39.356712	1.8
min	1.000000	2.099862e+05	0.000000	-2.060000	2.472000	126.064000	3.8
25%	12.000000	5.533501e+05	0.000000	47.460000	2.933000	131.735000	6.8
50%	23.000000	9.607460e+05	0.000000	62.670000	3.445000	182.616521	7.8
75%	34.000000	1.420159e+06	0.000000	74.940000	3.735000	212.743293	8.6
max	45.000000	3.818686e+06	1.000000	100.140000	4.468000	227.232807	14.3

In [11]:

```
df['Date'] = pd.to_datetime(df['Date'])
```

In [12]:

```
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   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 [14]:

```
df.isnull().sum()
```

Out[14]:

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

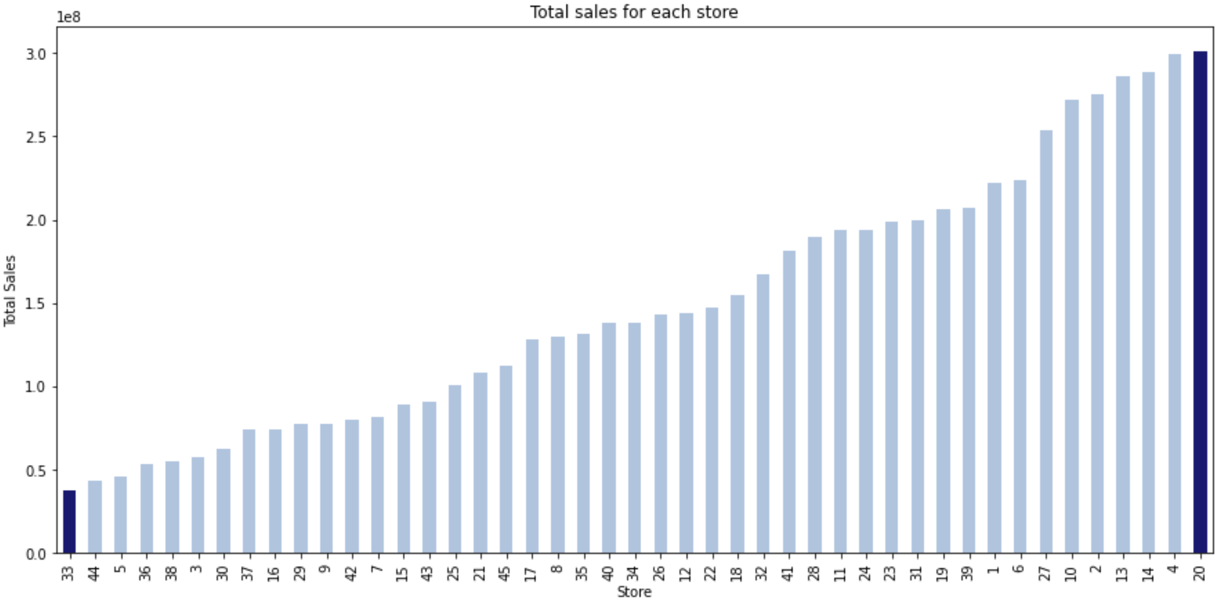
In [15]:

```
df["Day"]= pd.DatetimeIndex(df['Date']).day
df['Month'] = pd.DatetimeIndex(df['Date']).month
df['Year'] = pd.DatetimeIndex(df['Date']).year
df
```

Out[15]:

```
      Store  Date   Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price         CPI  Unemployme
   0      1  2010-    1643690.90             0        42.31       2.572  211.096358        8.1
             05-02
   1      1  2010-    1641957.44             1        38.51       2.548  211.242170        8.1
             12-02
   2      1  2010-    1611968.17             0        39.93       2.514  211.289143        8.1
             02-19
   3      1  2010-    1409727.59             0        46.63       2.561  211.319643        8.1
             02-26
   4      1  2010-    1554806.68             0        46.50       2.625  211.350143        8.1
             05-03
 ...    ...   ...            ...           ...          ...         ...         ...
6430     45  2012-     713173.95             0        64.88       3.997  192.013558        8.6
             09-28
6431     45  2012-     733455.07             0        64.89       3.985  192.170412        8.6
             05-10
6432     45  2012-     734464.36             0        54.47       4.000  192.327265        8.6
             12-10
6433     45  2012-     718125.53             0        56.47       3.969  192.330854        8.6
             10-19
6434     45  2012-     760281.43             0        58.85       3.882  192.308899        8.6
             10-26

6435 rows × 11 columns
```

In [18]:
```
#Task 1
#Maximum Sales
plt.figure(figsize=(15,7))
total_sales_for_each_store = df.groupby('Store')['Weekly_Sales'].sum().sort_value
total_sales_for_each_store_array = np.array(total_sales_for_each_store) # convert
clrs = ['lightsteelblue' if ((x < max(total_sales_for_each_store_array)) and (x
ax = total_sales_for_each_store.plot(kind='bar',color=clrs);
plt.title('Total sales for each store')
plt.xlabel('Store')
plt.ylabel('Total Sales');
```

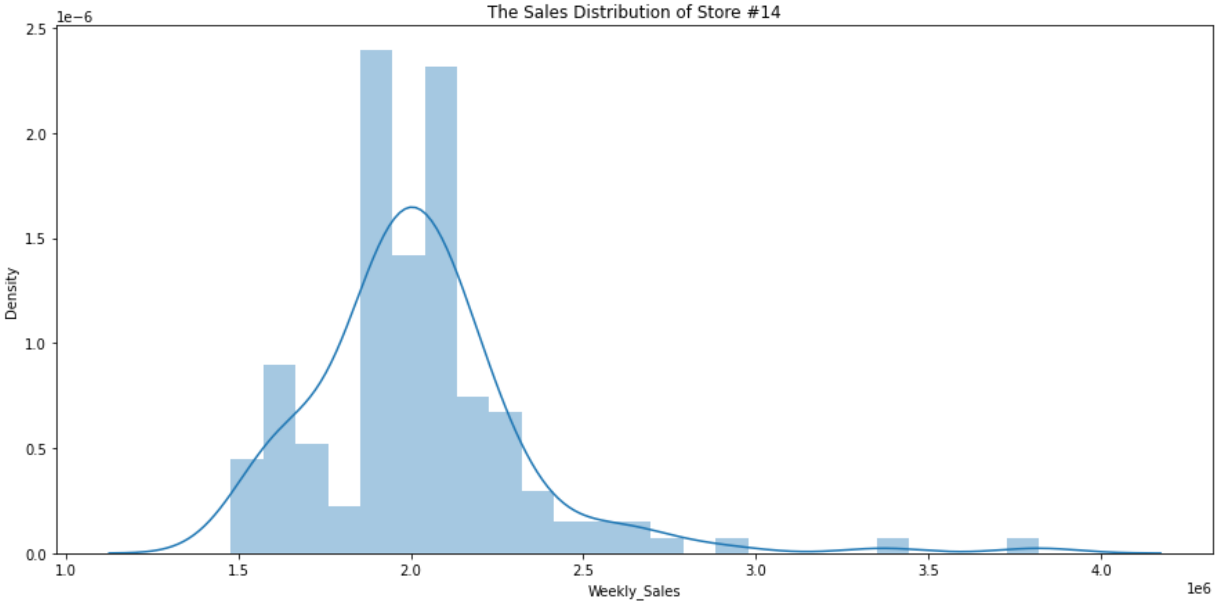


In [22]:
```
#Maximum Std.
df_std = pd.DataFrame(df.groupby('Store')['Weekly_Sales'].std().sort_values(ascen
print("The store has maximum standard deviation is "+str(df_std.head(1).index[0]
```

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

In [23]:
```
# Distribution of store has maximum standard deviation
plt.figure(figsize=(15,7))
sns.distplot(df[df['Store'] == df_std.head(1).index[0]]['Weekly_Sales'])
plt.title('The Sales Distribution of Store #'+ str(df_std.head(1).index[0]));
```

```
c:\users\bhavuk\appdata\local\programs\python\python37\lib\site-packages\seabor
n\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and
will be removed in a future version. Please adapt your code to use either `disp
lot` (a figure-level function with similar flexibility) or `histplot` (an axes-
level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [24]:

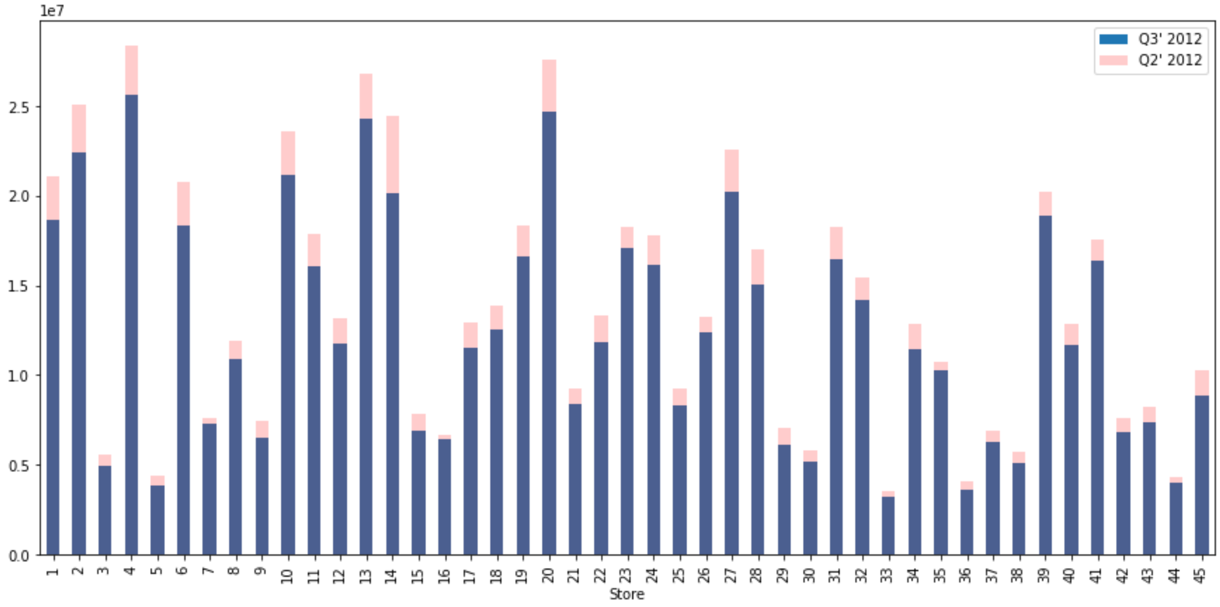
```
# Coefficient of mean to standard deviation
coef_mean_std = pd.DataFrame(df.groupby('Store')['Weekly_Sales'].std() / df.group
coef_mean_std = coef_mean_std.rename(columns={'Weekly_Sales':'Coefficient of mear
coef_mean_std
```

Out[24]:

	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

	Coefficient of mean to standard deviation
Store	
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

```
In [29]: plt.figure(figsize=(15,7))
         Q3 = df[(df['Date'] > '2012-07-01') & (df['Date'] < '2012-09-30')].groupby('Store
         Q2 = df[(df['Date'] > '2012-04-01') & (df['Date'] < '2012-06-30')].groupby('Store
         Q2.plot(ax=Q3.plot(kind='bar',legend=True),kind='bar',color='r',alpha=0.2,legend=
         plt.legend(["Q3' 2012", "Q2' 2012"]);
```

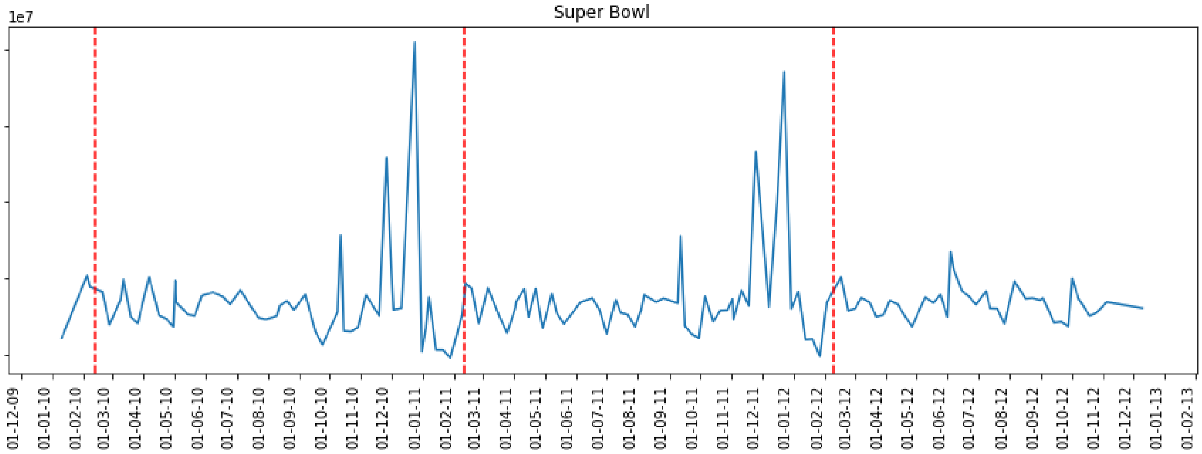


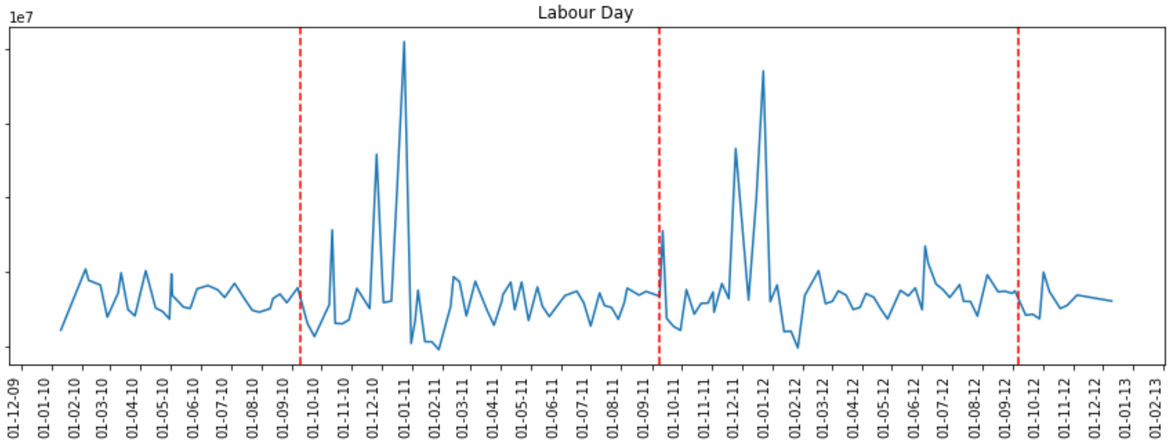
In [30]:
```
print('Store have good quarterly growth rate in Q3’2012 is Store '+str(Q3.idxmax(
```

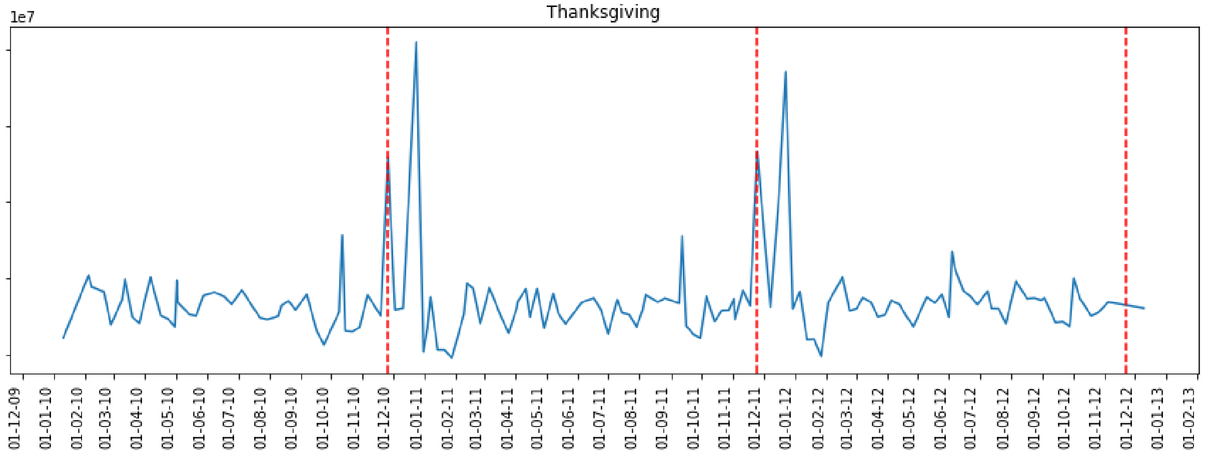
```
Store have good quarterly growth rate in Q3’2012 is Store 4 With 25652119.35 $
```

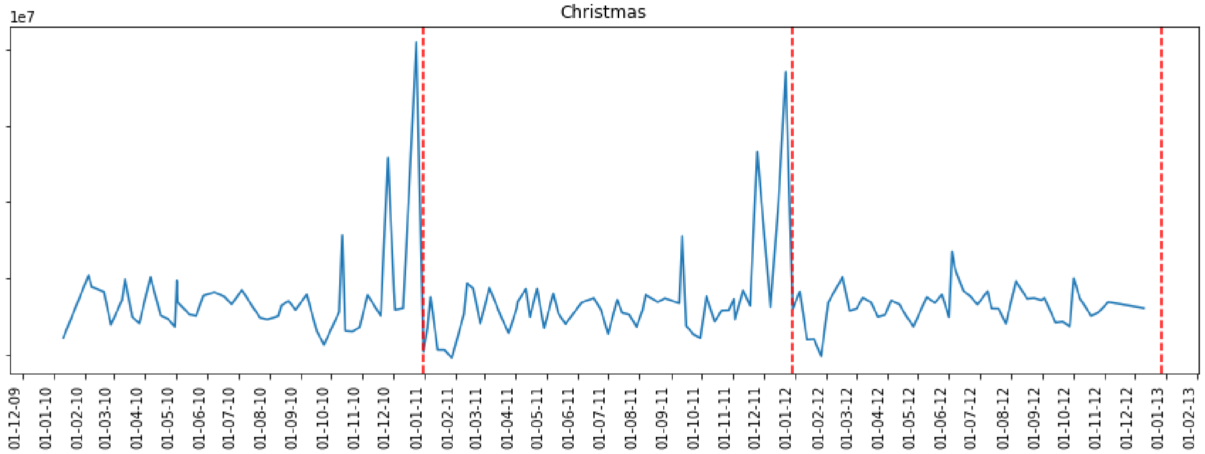
In [36]:

```
#Holidays event
total_sales = df.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']

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()


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')
```



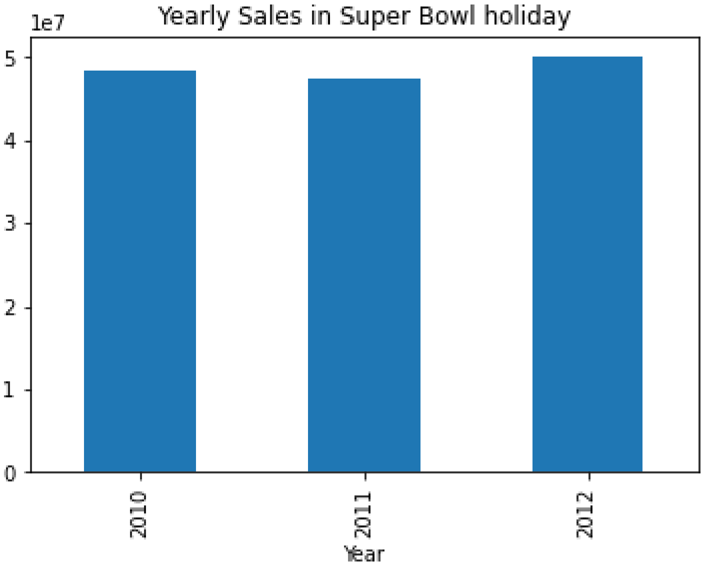

Labour Day
1e7
8
7
6
5
4
01-12-09
01-01-10
01-02-10
01-03-10
01-04-10
01-05-10
01-06-10
01-07-10
01-08-10
01-09-10
01-10-10
01-11-10
01-12-10
01-01-11
01-02-11
01-03-11
01-04-11
01-05-11
01-06-11
01-07-11
01-08-11
01-09-11
01-10-11
01-11-11
01-12-11
01-01-12
01-02-12
01-03-12
01-04-12
01-05-12
01-06-12
01-07-12
01-08-12
01-09-12
01-10-12
01-11-12
01-12-12
01-01-13
01-02-13

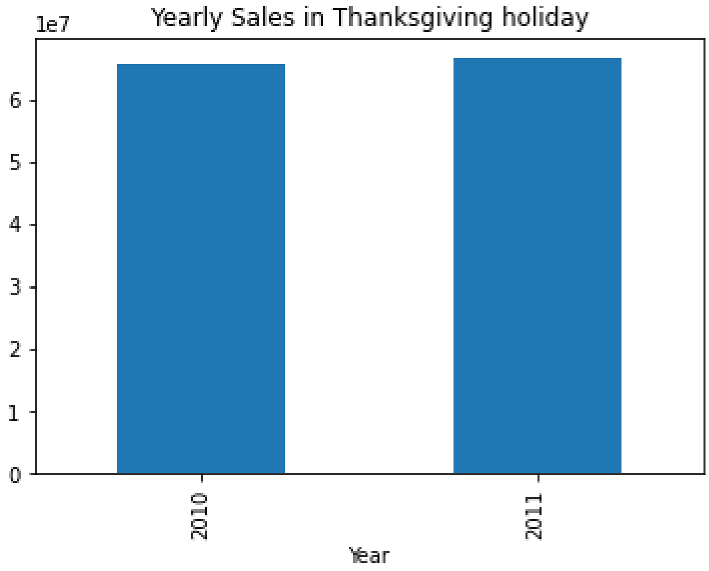

Thanksgiving
1e7
8
7
6
5
4
01-12-09
01-01-10
01-02-10
01-03-10
01-04-10
01-05-10
01-06-10
01-07-10
01-08-10
01-09-10
01-10-10
01-11-10
01-12-10
01-01-11
01-02-11
01-03-11
01-04-11
01-05-11
01-06-11
01-07-11
01-08-11
01-09-11
01-10-11
01-11-11
01-12-11
01-01-12
01-02-12
01-03-12
01-04-12
01-05-12
01-06-12
01-07-12
01-08-12
01-09-12
01-10-12
01-11-12
01-12-12
01-01-13
01-02-13

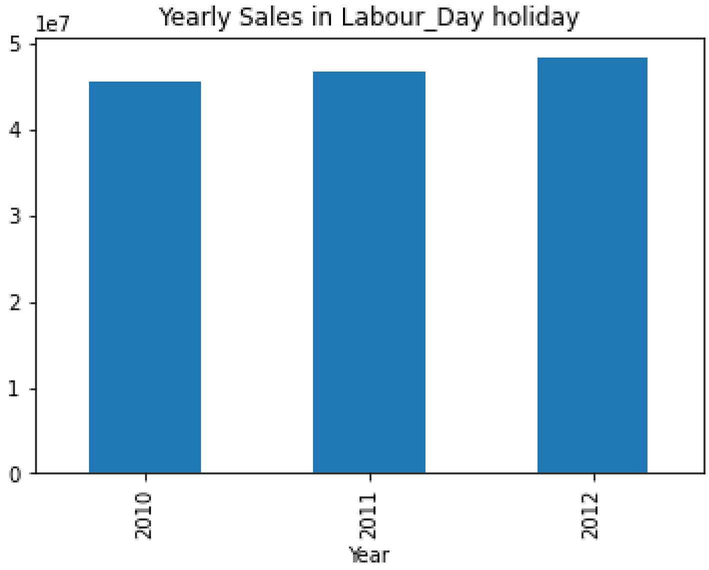

Christmas
1e7
8
7
6
5
4
01-12-09
01-01-10
01-02-10
01-03-10
01-04-10
01-05-10
01-06-10
01-07-10
01-08-10
01-09-10
01-10-10
01-11-10
01-12-10
01-01-11
01-02-11
01-03-11
01-04-11
01-05-11
01-06-11
01-07-11
01-08-11
01-09-11
01-10-11
01-11-11
01-12-11
01-01-12
01-02-12
01-03-12
01-04-12
01-05-12
01-06-12
01-07-12
01-08-12
01-09-12
01-10-12
01-11-12
01-12-12
01-01-13
01-02-13

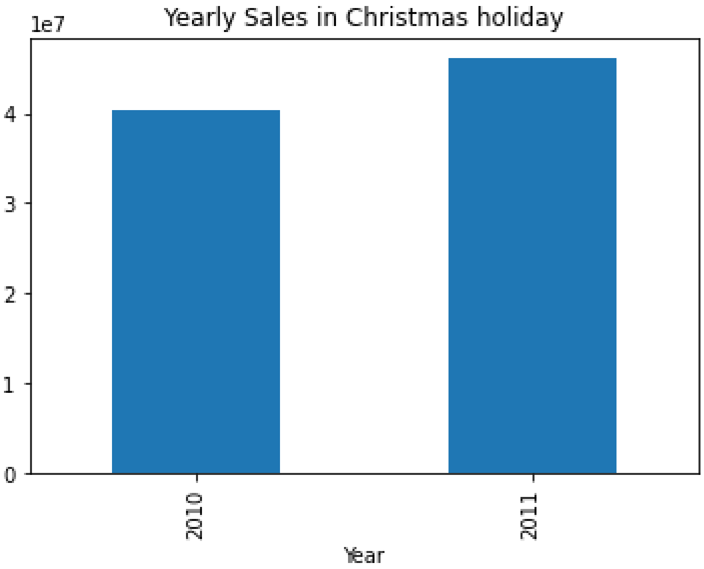
In [38]:
```
# Yearly Sales in holidays
Super_Bowl_df = pd.DataFrame(df.loc[df.Date.isin(Super_Bowl)].groupby('Year')['We
Thanksgiving_df = pd.DataFrame(df.loc[df.Date.isin(Thanksgiving)].groupby('Year')
Labour_Day_df = pd.DataFrame(df.loc[df.Date.isin(Labour_Day)].groupby('Year')['We
Christmas_df = pd.DataFrame(df.loc[df.Date.isin(Christmas)].groupby('Year')['Week

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

Out[38]:
```
<AxesSubplot:title={'center':'Yearly Sales in Christmas holiday'}, xlabel='Yea
r'>
```



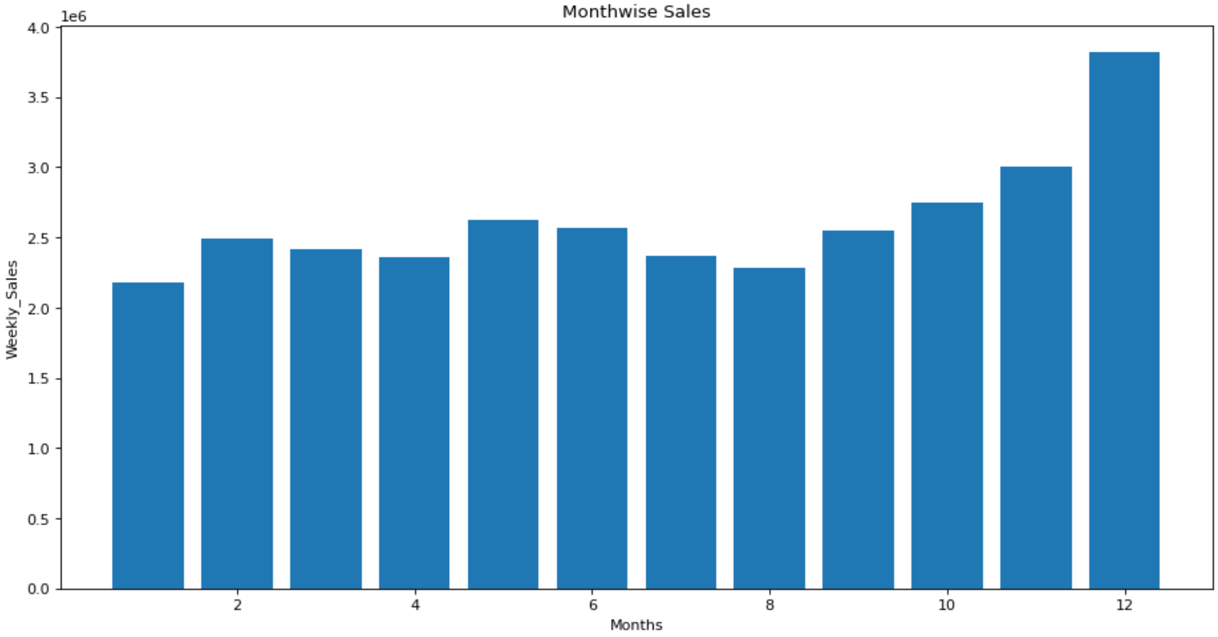

Yearly Sales in Thanksgiving holiday
1e7
6
5
4
3
2
1
0
2010
2011
Year


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


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

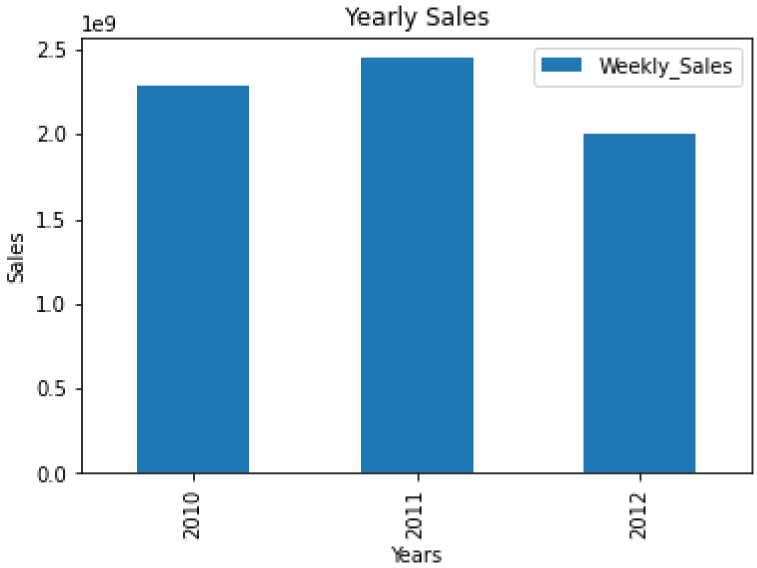
In [40]:
```
#Monthwise Sales
plt.figure(figsize=(14,7),dpi=80)
plt.bar(df['Month'],df['Weekly_Sales'])
plt.xlabel('Months')
plt.ylabel('Weekly_Sales')
plt.title('Monthwise Sales')
```

Out[40]:
```
Text(0.5, 1.0, 'Monthwise Sales')
```



In [41]:
```
plt.figure(figsize=(10,7),dpi=80)
df.groupby('Year')[['Weekly_Sales']].sum().plot(kind='bar',legend=True)
plt.xlabel('Years')
plt.ylabel('Sales')
plt.title('Yearly Sales')
```

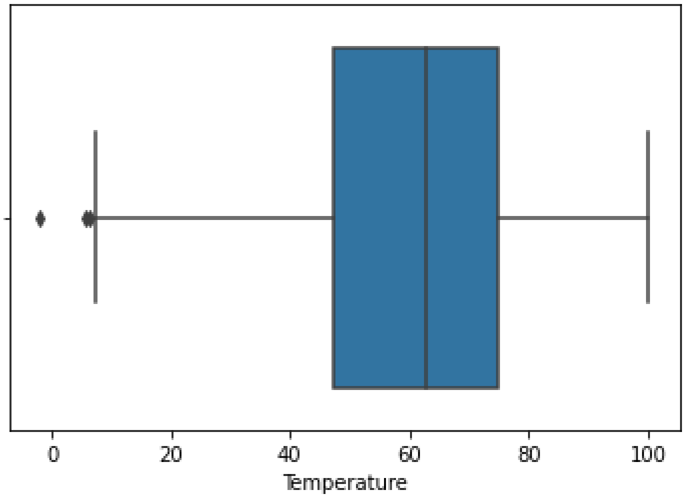
Out[41]:
```
Text(0.5, 1.0, 'Yearly Sales')

<Figure size 800x560 with 0 Axes>
```

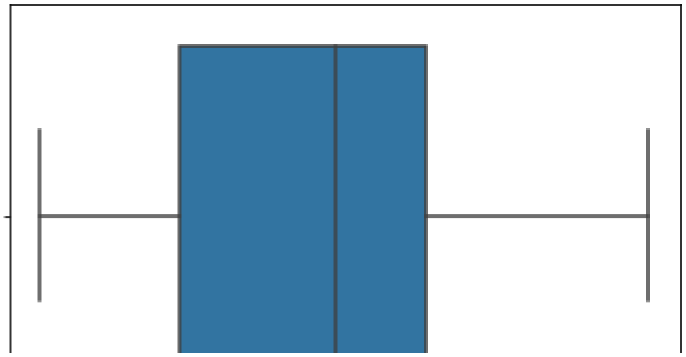


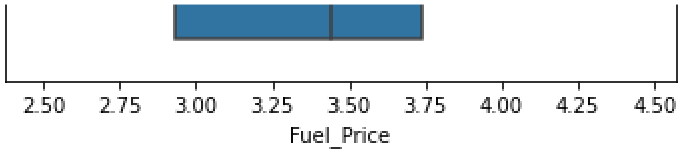
In [ ]:
```
#Insights
(1) Year 2010 has the highest sales and 2012 has the lowest sales.
(2) December month has the highest weekly sales.
(3) Year 2011 has the highest weekly sales.
```

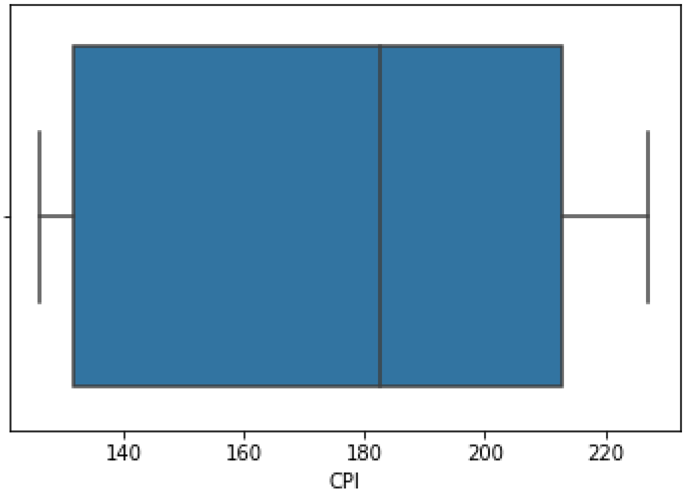
In [58]:
```
#Build prediction model
fig, axs = plt.subplots(4,figsize=(6,18))
X = df[['Temperature','Fuel_Price','CPI','Unemployment']]
for i,column in enumerate(X):
    sns.boxplot(df[column], ax=axs[i])
```

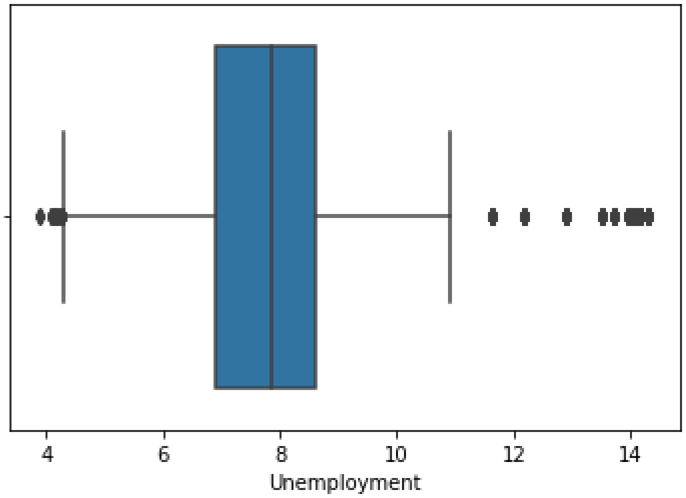
```
c:\users\bhavuk\appdata\local\programs\python\python37\lib\site-packages\seabor
n\_decorators.py:43: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
c:\users\bhavuk\appdata\local\programs\python\python37\lib\site-packages\seabor
n\_decorators.py:43: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
c:\users\bhavuk\appdata\local\programs\python\python37\lib\site-packages\seabor
n\_decorators.py:43: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
c:\users\bhavuk\appdata\local\programs\python\python37\lib\site-packages\seabor
n\_decorators.py:43: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
```











In [50]:
```
x_features_object = df[df['Store'] ==1][['Store','Date']]
date_obj = df[df['Store'] ==1][['Date']]
date_obj.index +=1
x_features_object.Date = date_obj.index
x_features_object.head()

y_target = df[df['Store'] ==1]['Weekly_Sales']
y_target.head()
```

Out[50]:
```
0    1643690.90
1    1641957.44
2    1611968.17
3    1409727.59
4    1554806.68
Name: Weekly_Sales, dtype: float64
```

In [51]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x_features_object,y_target,rand
```

In [66]:

```
from sklearn.linear_model import LinearRegression
from sklearn import metrics

linreg = LinearRegression()
linreg.fit(x_train,y_train)
feature_dataset = df[df['Store'] ==1][['Store','CPI','Unemployment','Fuel_Price']
feature_dataset.head()
```

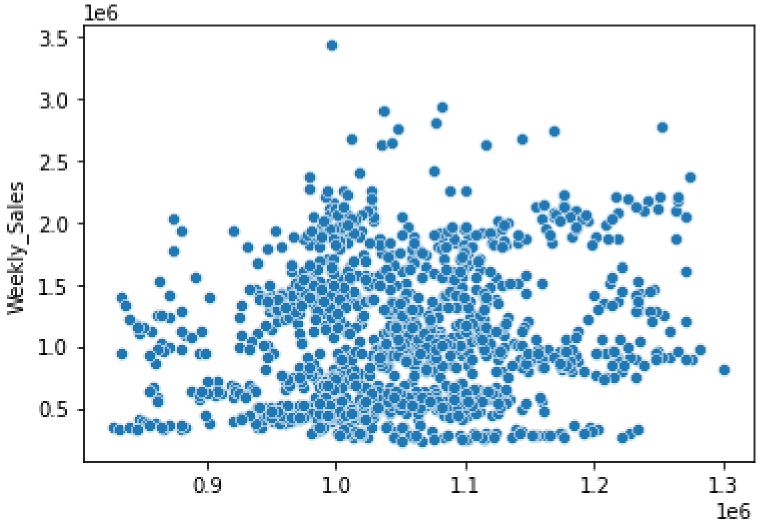
Out[66]:

	Store	CPI	Unemployment	Fuel_Price
0	1	211.096358	8.106	2.572
1	1	211.242170	8.106	2.548
2	1	211.289143	8.106	2.514
3	1	211.319643	8.106	2.561
4	1	211.350143	8.106	2.625

In [67]:

```
#Linear Regression
x = df[['Store','Fuel_Price','CPI','Unemployment','Day','Month','Year']]
y = df['Weekly_Sales']

x_train, x_test, y_train, y_test = train_test_split(X,y,test_size=0.2)

print('Linear Regression:')
print()
reg = LinearRegression()
reg.fit(x_train, y_train)
y_pred = reg.predict(x_test)
print('Accuracy:',reg.score(x_train, y_train)*100)


print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pr


sns.scatterplot(y_pred, y_test);
```

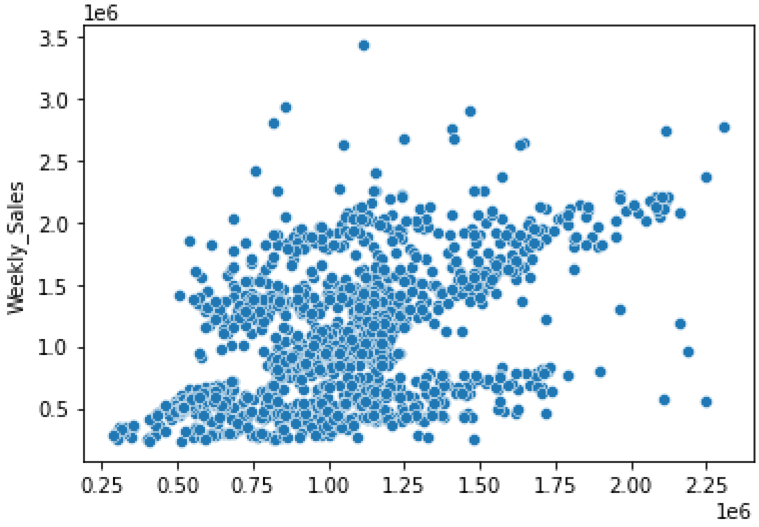
```
Linear Regression:

Accuracy: 2.385896233042928
Mean Absolute Error: 463162.07130399876
Mean Squared Error: 300375463241.1589
Root Mean Squared Error: 548065.1998085254

c:\users\bhavuk\appdata\local\programs\python\python37\lib\site-packages\seabor
n\_decorators.py:43: FutureWarning: Pass the following variables as keyword arg
s: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error
or misinterpretation.
  FutureWarning
```



In [75]:

```
# Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor

print('Random Forest Regressor:')
print()
rfr = RandomForestRegressor(n_estimators = 400,max_depth=15,n_jobs=5)
rfr.fit(x_train,y_train)
y_pred=rfr.predict(x_test)
print('Accuracy:',rfr.score(x_test, y_test)*100)

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pr


sns.scatterplot(y_pred, y_test);

print('Random Forest Regressor model gives better accuracy')
```

```
Random Forest Regressor:

Accuracy: 19.875987290547435
Mean Absolute Error: 365676.68323243014
Mean Squared Error: 246905928864.41336
Root Mean Squared Error: 496896.295885181
Random Forest Regressor model gives better accuracy


c:\users\bhavuk\appdata\local\programs\python\python37\lib\site-packages\seabor
n\_decorators.py:43: FutureWarning: Pass the following variables as keyword arg
s: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error
or misinterpretation.
  FutureWarning
```



In [71]:

```
df['Day'] = pd.to_datetime(df['Date']).dt.day_name()
df.head()
```

Out[71]:

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

In [ ]:

```
FINISH
```