Project on "Retail Analysis with Walmart Data" using python..

Dataset Description

This is the historical data that covers sales from 2010-02-05 to 2012-11-01, in the file Walmart_Store_sales. Within this file you will find the following fields:

Store - the store number

Date - the week of sales

Weekly_Sales - sales for the given store

Holiday_Flag - whether the week is a special holiday week 1 – Holiday week 0 – Non-holiday week

Temperature - Temperature on the day of sale

Fuel_Price - Cost of fuel in the region

CPI – Prevailing consumer price index

Unemployment - Prevailing unemployment rate

In [1]:
```
# Firstly importing all the required libraries...

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
%matplotlib inline
from scipy import  stats
```

In [2]:
```
#Sales data for 45 Walmart stores are available and data that covers sales from 2010-02-05 to 2012-11-01...
```

In [3]:
```
# Loading the dataset..
data=pd.read_csv('Downloads/1577429980_walmart_store_sales/Walmart_Store_sales.csv')
```

Basic understanding of dataset

In [4]:
```
#Show data set(First 5 rows only)
data.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]:
```
# Number of missing values
data.isna().sum()
```

Out[5]:
```
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
dtype: int64
```

In [6]:
```
# Data shape
data.shape
```

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

In [7]:
```
# Describe all mathematical details
data.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 [9]:
```
# Convert date to datetime format and show dataset information
data['Date'] =  pd.to_datetime(data['Date'],infer_datetime_format=True)
data.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 [10]:
```
# Splitting Date and create new columns (Day, Month, and Year)
data["Day"]= pd.DatetimeIndex(data['Date']).day
data['Month'] = pd.DatetimeIndex(data['Date']).month
data['Year'] = pd.DatetimeIndex(data['Date']).year
data
```

Out[10]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Day	Month	Year
0	1	2010-05-02	1643690.90	0	42.31	2.572	211.096358	8.106	2	5	2010
1	1	2010-12-02	1641957.44	1	38.51	2.548	211.242170	8.106	2	12	2010
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8.106	19	2	2010
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8.106	26	2	2010
4	1	2010-05-03	1554806.68	0	46.50	2.625	211.350143	8.106	3	5	2010
...	...	...	...	...	...	...	...	...	...	...	...
6430	45	2012-09-28	713173.95	0	64.88	3.997	192.013558	8.684	28	9	2012
6431	45	2012-05-10	733455.07	0	64.89	3.985	192.170412	8.667	10	5	2012
6432	45	2012-12-10	734464.36	0	54.47	4.000	192.327265	8.667	10	12	2012
6433	45	2012-10-19	718125.53	0	56.47	3.969	192.330854	8.667	19	10	2012
6434	45	2012-10-26	760281.43	0	58.85	3.882	192.308899	8.667	26	10	2012

6435 rows × 11 columns

Basic Statistics tasks

1.Which store has maximum sales

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

3.Which store/s has good quarterly growth rate in Q3'2012

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

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

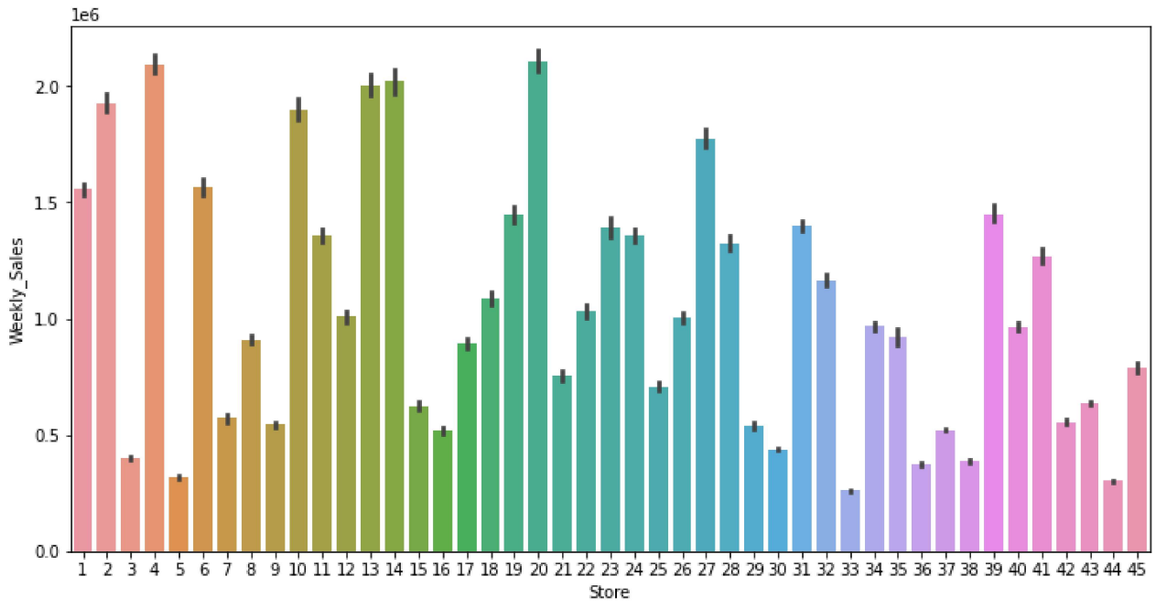
Task 1 - Which store has maximum sales..

In [11]:
```
# Which store has maximum sales.
max_sales=data.groupby('Store')['Weekly_Sales'].sum()
#max_sales.idxmax()
print("The Store {} has Maximum Sales.".format(max_sales.idxmax(),max_sales.max()))
```

```
The Store 20 has Maximum Sales.
```

In [12]:
```
#Plotting the maximum sales
plt.figure(figsize=(12,6))
sns.barplot(x=data.Store,y=data.Weekly_Sales)
```

Out[12]:
```
<AxesSubplot:xlabel='Store', ylabel='Weekly_Sales'>
```



Task 2 -

Part(1) Which store has maximum standard deviation i.e., the sales vary a lot.

Part(2) Also, find out the coefficient of mean to standard deviation

In [13]:
```
# Part 1- Which store has maximum standard deviation
data_std =data.groupby('Store')['Weekly_Sales'].std()
print("The store {} has maximum standard deviation".format(data_std.idxmax(),data_std.max()))
```

```
The store 14 has maximum standard deviation
```

In [14]:
```
# Part 2- Coefficient of mean to standard deviation
coef_of_mean_std = pd.DataFrame(data.groupby('Store')['Weekly_Sales'].std() / data.groupby('Store')['Weekly_Sales'].mean())
coef_of_mean_std = coef_of_mean_std.rename(columns={'Weekly_Sales':'Coefficient of mean to standard deviation'})
coef_of_mean_std.head()
```

Out[14]:

Store	Coefficient of mean to standard deviation
1	0.100292
2	0.123424
3	0.115021
4	0.127083
5	0.118668

In [15]:
```
#data_std =data.groupby('Store').agg({'Weekly_Sales':['mean','std']})
#data_std.head()
```

Task 3 - Which store/s has good quarterly growth rate in Q3’2012

In [16]:
```
# Sales for third quarterly in 2012
# Q1 = 1, 2, 3 ; Q2 = 4, 5, 6 ; Q3 = 7, 8, 9 ; Q4= 10, 11, 12

Q3_2012 = data[(data['Month'] > 6) & (data['Month'] < 10)].groupby('Store')['Weekly_Sales'].sum()
print("Store Number {} has Good Quartely Growth in Q3'2012".format(Q3_2012.idxmax(),Q3_2012.max()))
```

```
Store Number 4 has Good Quartely Growth in Q3'2012
```

Task 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, 27-Dec-13

In [17]:
```
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']

data.loc[data.Date.isin(Super_Bowl)]
```

Out[17]:

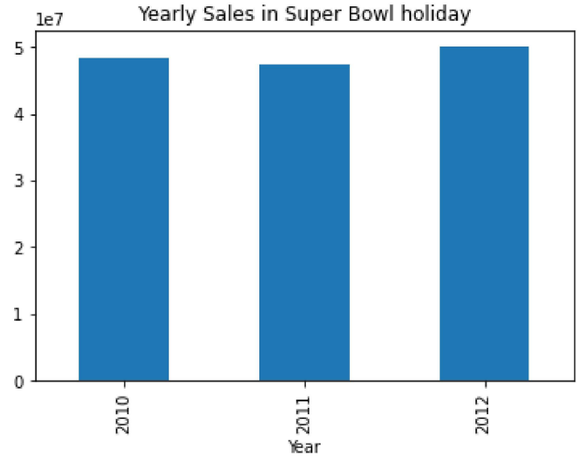
	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Day	Month	Year
1	1	2010-12-02	1641957.44	1	38.51	2.548	211.242170	8.106	2	12	2010
53	1	2011-11-02	1649614.93	1	36.39	3.022	212.936705	7.742	2	11	2011
105	1	2012-10-02	1802477.43	1	48.02	3.409	220.265178	7.348	2	10	2012
144	2	2010-12-02	2137809.50	1	38.49	2.548	210.897994	8.324	2	12	2010
196	2	2011-11-02	2168041.61	1	33.19	3.022	212.592862	8.028	2	11	2011
...	...	...	...	...	...	...	...	...	...	...	...
6202	44	2011-11-02	307486.73	1	30.83	3.034	127.859129	7.224	2	11	2011
6254	44	2012-10-02	325377.97	1	33.73	3.116	130.384903	5.774	2	10	2012
6293	45	2010-12-02	656988.64	1	27.73	2.773	181.982317	8.992	2	12	2010
6345	45	2011-11-02	766456.00	1	30.30	3.239	183.701613	8.549	2	11	2011
6397	45	2012-10-02	803657.12	1	37.00	3.640	189.707605	8.424	2	10	2012

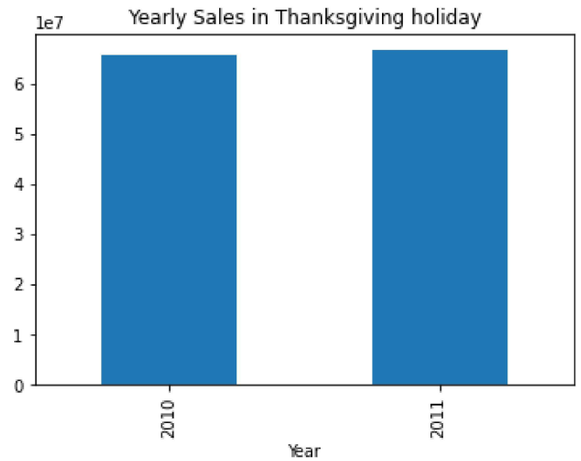
135 rows × 11 columns

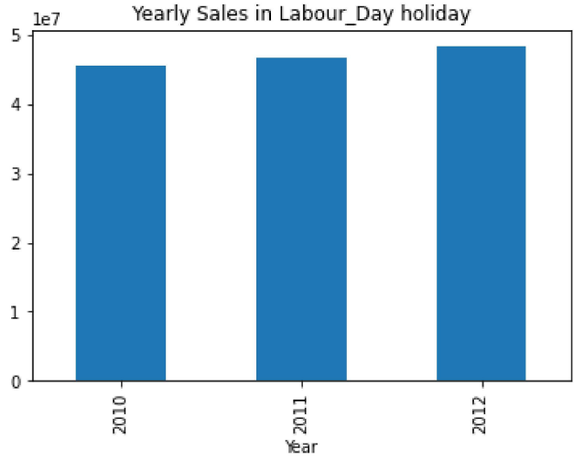
In [18]:
```
# Yearly Sales in holidays
Super_Bowl_df = pd.DataFrame(data.loc[data.Date.isin(Super_Bowl)].groupby('Year')['Weekly_Sales'].sum())
Thanksgiving_df = pd.DataFrame(data.loc[data.Date.isin(Thanksgiving)].groupby('Year')['Weekly_Sales'].sum())
Labour_Day_df = pd.DataFrame(data.loc[data.Date.isin(Labour_Day)].groupby('Year')['Weekly_Sales'].sum())
Christmas_df = pd.DataFrame(data.loc[data.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')
```

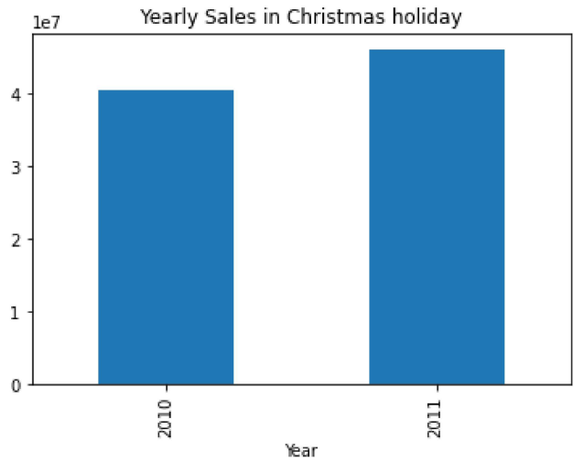
```
C:\Users\Admin\anaconda3\lib\site-packages\pandas\core\arrays\datetimes.py:339: UserWarning: Parsing '26-11-2010' in DD/MM/YYYY format. Provide format or specify infer_da
tetime_format=True for consistent parsing.
  return cls._from_sequence_not_strict(scalars, dtype=dtype, copy=copy)
C:\Users\Admin\anaconda3\lib\site-packages\pandas\core\arrays\datetimes.py:339: UserWarning: Parsing '25-11-2011' in DD/MM/YYYY format. Provide format or specify infer_da
tetime_format=True for consistent parsing.
  return cls._from_sequence_not_strict(scalars, dtype=dtype, copy=copy)
C:\Users\Admin\anaconda3\lib\site-packages\pandas\core\arrays\datetimes.py:339: UserWarning: Parsing '23-11-2012' in DD/MM/YYYY format. Provide format or specify infer_da
tetime_format=True for consistent parsing.
  return cls._from_sequence_not_strict(scalars, dtype=dtype, copy=copy)
C:\Users\Admin\anaconda3\lib\site-packages\pandas\core\arrays\datetimes.py:339: UserWarning: Parsing '31-12-2010' in DD/MM/YYYY format. Provide format or specify infer_da
tetime_format=True for consistent parsing.
  return cls._from_sequence_not_strict(scalars, dtype=dtype, copy=copy)
C:\Users\Admin\anaconda3\lib\site-packages\pandas\core\arrays\datetimes.py:339: UserWarning: Parsing '30-12-2011' in DD/MM/YYYY format. Provide format or specify infer_da
tetime_format=True for consistent parsing.
  return cls._from_sequence_not_strict(scalars, dtype=dtype, copy=copy)
C:\Users\Admin\anaconda3\lib\site-packages\pandas\core\arrays\datetimes.py:339: UserWarning: Parsing '28-12-2012' in DD/MM/YYYY format. Provide format or specify infer_da
tetime_format=True for consistent parsing.
  return cls._from_sequence_not_strict(scalars, dtype=dtype, copy=copy)
```

Out[18]:
```
<AxesSubplot:title={'center':'Yearly Sales in Christmas holiday'}, xlabel='Year'>
```


Yearly Sales in Super Bowl holiday
1e7
5
4
3
2
1
0
2010
2011
2012
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

In [19]:
```
# Mean of Sales of Holidays & Non-Holidays
data.groupby(data.Holiday_Flag).mean()
```

Out[19]:

Holiday_Flag	Store	Weekly_Sales	Temperature	Fuel_Price	CPI	Unemployment	Day	Month	Year
0	23.0	1.041256e+06	61.448124	3.368467	171.601725	7.993514	15.736842	6.172932	2010.977444
1	23.0	1.122888e+06	50.232044	3.227464	171.268092	8.074127	14.500000	10.500000	2010.800000

In [20]:
```
# Mean of Non Holidays
Mean=data.loc[(data['Holiday_Flag']==0)].Weekly_Sales.mean()
Mean
```

Out[20]:
```
1041256.3802088564
```

In [21]:
```
# List of Holidays where sales are higher than mean of sales of non holidays
Result= data[(data['Weekly_Sales']> Mean)&(data['Holiday_Flag']==1)]
Result
```

Out[21]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Day	Month	Year
1	1	2010-12-02	1641957.44	1	38.51	2.548	211.242170	8.106	2	12	2010
31	1	2010-10-09	1507460.69	1	78.69	2.565	211.495190	7.787	9	10	2010
42	1	2010-11-26	1955624.11	1	64.52	2.735	211.748433	7.838	26	11	2010
47	1	2010-12-31	1367320.01	1	48.43	2.943	211.404932	7.838	31	12	2010
53	1	2011-11-02	1649614.93	1	36.39	3.022	212.936705	7.742	2	11	2011
...	...	...	...	...	...	...	...	...	...	...	...
5819	41	2011-12-30	1264014.16	1	34.12	3.119	196.358610	6.759	30	12	2011
5825	41	2012-10-02	1238844.56	1	22.00	3.103	196.919506	6.589	2	10	2012
5855	41	2012-07-09	1392143.82	1	67.41	3.596	198.095048	6.432	9	7	2012
6334	45	2010-11-26	1182500.16	1	46.15	3.039	182.783277	8.724	26	11	2010
6386	45	2011-11-25	1170672.94	1	48.71	3.492	188.350400	8.523	25	11	2011

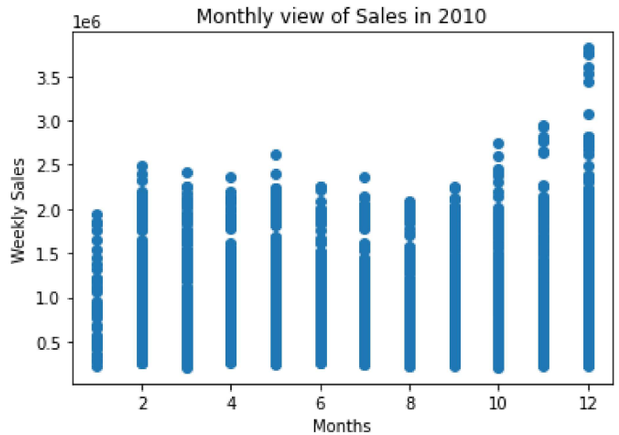
220 rows × 11 columns

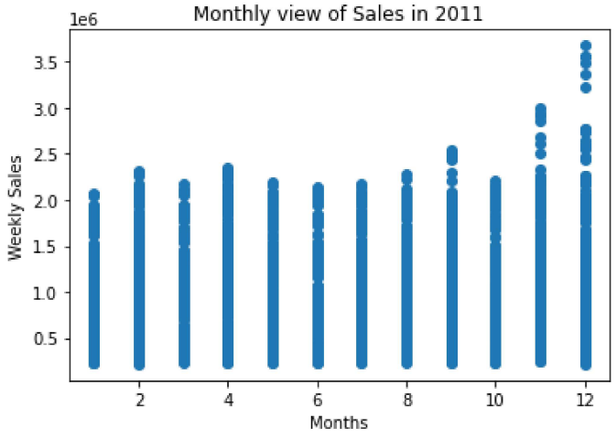
In [22]:
```
Result['Date'].unique
```

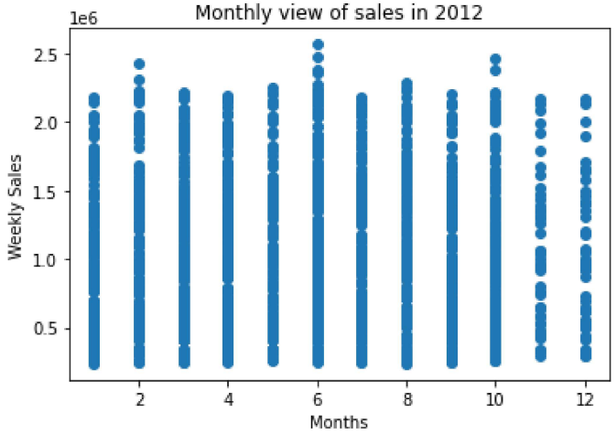
Out[22]:
```
<bound method Series.unique of 1      2010-12-02
31     2010-10-09
42     2010-11-26
47     2010-12-31
53     2011-11-02
          ...
5819   2011-12-30
5825   2012-10-02
5855   2012-07-09
6334   2010-11-26
6386   2011-11-25
Name: Date, Length: 220, dtype: datetime64[ns]>
```

Task 5 - Provide a monthly and semester view of sales in units and give insights

In [23]:
```
# Monthly view of sales for each years
plt.scatter(data[data.Year==2010]["Month"],data[data.Year==2010]["Weekly_Sales"])
plt.xlabel("Months")
plt.ylabel("Weekly Sales")
plt.title("Monthly view of Sales in 2010")
plt.show()

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

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

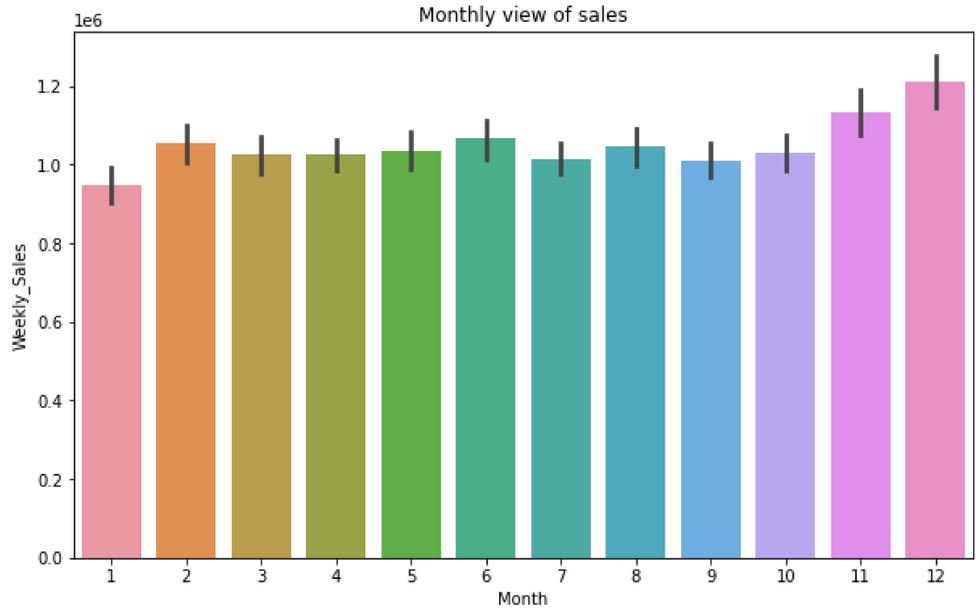



Monthly view of Sales in 2011
1e6
3.5
3.0
2.5
2.0
1.5
1.0
0.5
Weekly Sales
2
4
6
8
10
12
Months


Monthly view of sales in 2012
1e6
2.5
2.0
1.5
1.0
0.5
Weekly Sales
2
4
6
8
10
12
Months

In [24]:
```
# Monthly view of sales for all years
plt.figure(figsize=(10,6))
ax=sns.barplot(x="Month",y="Weekly_Sales",data=data)
plt.title("Monthly view of sales")
```

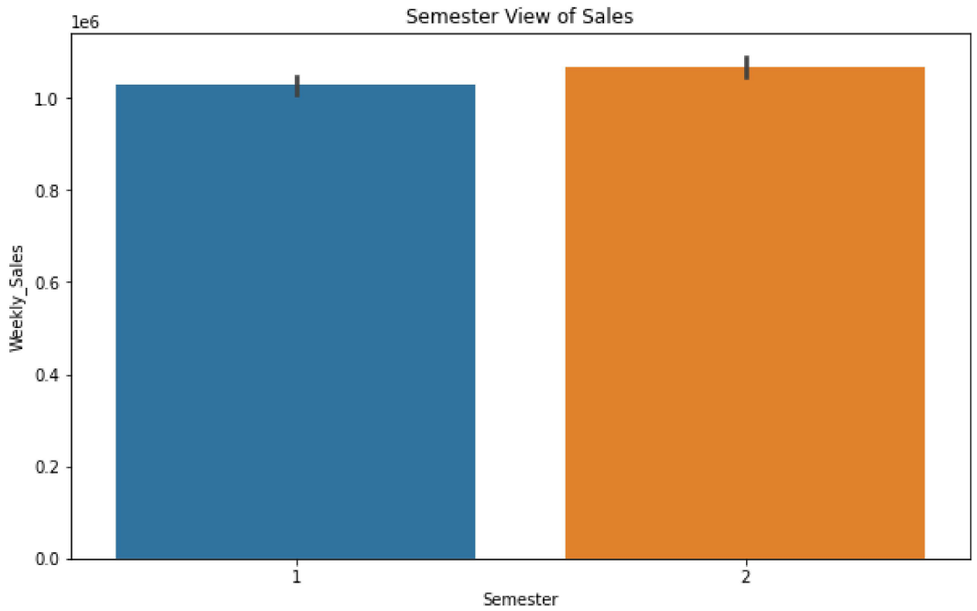
Out[24]: Text(0.5, 1.0, 'Monthly view of sales')



In [25]:
```
# For month 7 to 12 - Semester 2
# For month 1 to 6 - Semester 1
data['Semester']=np.where(data['Month']< 7,1,2)
```

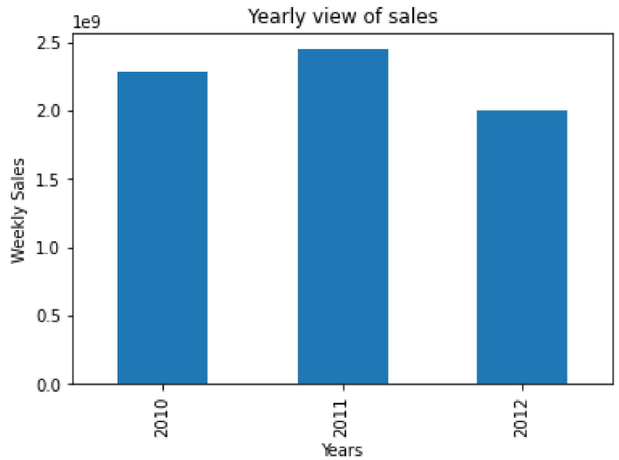
In [26]:
```
plt.figure(figsize=(10,6))
ax= sns.barplot(x="Semester",y="Weekly_Sales", data=data)
plt.title("Semester View of Sales")
```

Out[26]: Text(0.5, 1.0, 'Semester View of Sales')



In [27]:
```
plt.figure(figsize=(15,8))
data.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 1080x576 with 0 Axes>
```



Statistical Model

For Store 1 – Build prediction models to forecast demand

Linear Regression – Utilize variables like date and restructure dates as 1 for 5 Feb 2010 (starting from the earliest date in order). Hypothesize if CPI, unemployment, and fuel price have any impact on sales.

Change dates into days by creating new variable.

Select the model which gives best accuracy.

In [28]:
```
data.head()
```

Out[28]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Day	Month	Year	Semester
0	1	2010-05-02	1643690.90	0	42.31	2.572	211.096358	8.106	2	5	2010	1
1	1	2010-12-02	1641957.44	1	38.51	2.548	211.242170	8.106	2	12	2010	2
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8.106	19	2	2010	1
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8.106	26	2	2010	1
4	1	2010-05-03	1554806.68	0	46.50	2.625	211.350143	8.106	3	5	2010	1

Build Model

In [29]:
```
# Import sklearn
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
```

In [30]:
```
# Select features and target
X =data[data['Store'] ==1][['Store','Date']]
date_obj = data[data['Store'] ==1][['Date']]
date_obj.index +=1
X.Date = date_obj.index
X.head()
```

Out[30]:

	Store	Date
0	1	1
1	1	2
2	1	3
3	1	4
4	1	5

In [31]:
```
y_target = data[data['Store'] ==1]['Weekly_Sales']
y_target.head()
```

Out[31]:
```
0    1643690.90
1    1641957.44
2    1611968.17
3    1409727.59
4    1554806.68
Name: Weekly_Sales, dtype: float64
```

In [32]:
```
# Linear Regression Model
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,y_target,random_state=1)


from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(x_train,y_train)
y_pred= linreg.predict(x_test)

from sklearn import metrics

print('Accuracy:',linreg.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_pred)))
```

```
Accuracy: 5.402778835377909
Mean Absolute Error: 109573.90193034585
Mean Squared Error: 31711857054.0465
Root Mean Squared Error: 178078.23295969248
```

In [33]:
```
# features and target
feature_dataset = data[data['Store'] ==1][['Store','CPI','Unemployment','Fuel_Price']]
feature_dataset.head()
```

Out[33]:

	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 [34]:
```
response_set_cpi = data[data['Store'] ==1]['CPI'].astype('int64')
response_set_unemployment = data[data['Store'] ==1]['Unemployment'].astype('int64')


from sklearn.model_selection import train_test_split
x_train_cpi,x_test_cpi,y_train_cpi,y_test_cpi = train_test_split(feature_dataset,response_set_cpi,random_state=1)
x_train_unemp, x_test_unemp, y_train_unemp, y_test_unemp = train_test_split(feature_dataset,
                                                                            response_set_unemployment,random_state=1)
```

In [35]:
```
# Logistic Regression Model
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(max_iter=10000)

logreg.fit(x_train_cpi,y_train_cpi)
y_pred = logreg.predict(x_test_cpi)

logreg.fit(x_train_unemp,y_train_unemp)
```

Out[35]:
```
LogisticRegression(max_iter=10000)
```

In [36]:
```
y_pred_unemp = logreg.predict(x_test_unemp)

from sklearn import metrics
print(metrics.accuracy_score(y_test_cpi,y_pred))

print(metrics.accuracy_score(y_test_unemp,y_pred_unemp))
```

```
0.7222222222222222
0.9444444444444444
```

In [37]:
```
# Actual vs Predicted
print('cpi actual :', y_test_cpi.values[0:30])
print('cpi Predicted :', y_pred[0:30])
print('actual Unemployment :', y_test_unemp.values[0:30])
print('Predicted Unemployment :', y_pred_unemp[0:30])
```

```
cpi actual : [215 221 211 211 221 211 210 211 215 217 221 212 216 218 211 210 211 217
 215 211 212 217 221 219 214 211 211 219 215 219]
cpi Predicted : [215 221 211 211 221 211 211 211 215 215 221 211 215 218 211 211 211 217
 215 211 211 217 221 220 215 211 211 221 215 220]
actual Unemployment : [7 7 7 8 7 7 7 7 7 7 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7]
Predicted Unemployment : [7 7 7 7 6 7 7 7 7 7 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7]
```

In [38]:
```
data['Day'] = pd.to_datetime(data['Date']).dt.day_name()
data.head()
```

Out[38]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Day	Month	Year	Semester
0	1	2010-05-02	1643690.90	0	42.31	2.572	211.096358	8.106	Sunday	5	2010	1
1	1	2010-12-02	1641957.44	1	38.51	2.548	211.242170	8.106	Thursday	12	2010	2
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8.106	Friday	2	2010	1
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8.106	Friday	2	2010	1
4	1	2010-05-03	1554806.68	0	46.50	2.625	211.350143	8.106	Monday	5	2010	1