In [160]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from scipy import stats
import statsmodels.api as sm


import pandas_profiling
```

In [161]:

```
df = pd.read_csv('Walmart_Store_sales.csv')
```

In [162]:

```
df.head()
```

Out[162]:

	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 [163]:

```
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 [165]:

```
df.columns
```

Out[165]:

```
Index(['Store', 'Date', 'Weekly_Sales', 'Holiday_Flag', 'Temperature',
       'Fuel_Price', 'CPI', 'Unemployment'],
      dtype='object')
```

In [166]:

```
df.shape
```

Out[166]:

Out[166]:

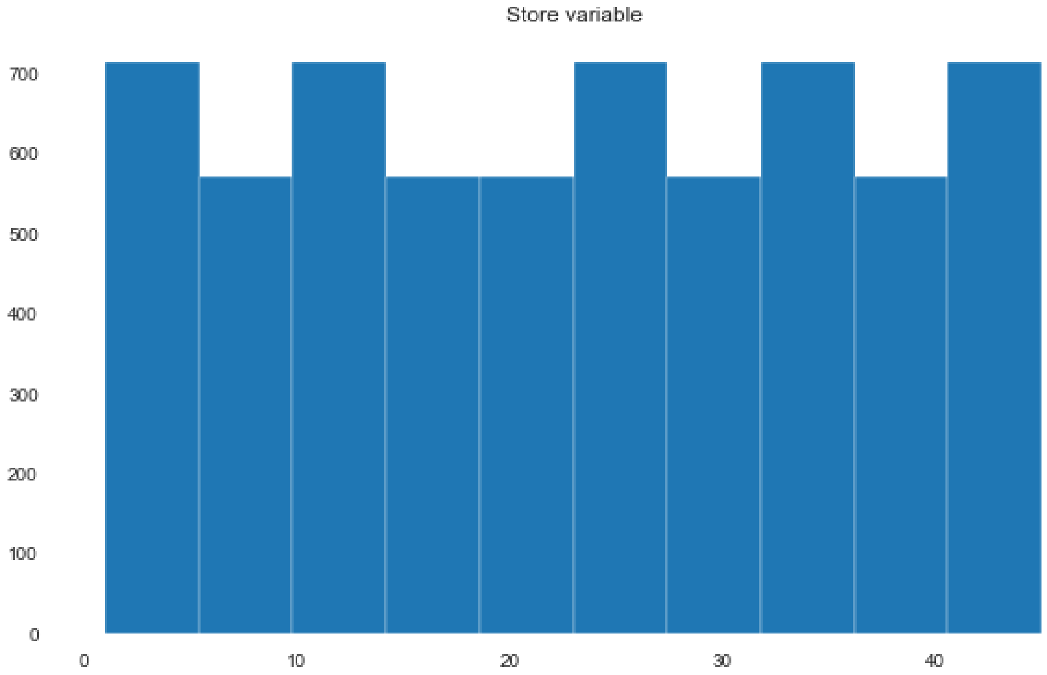
```
(6435, 8)
```

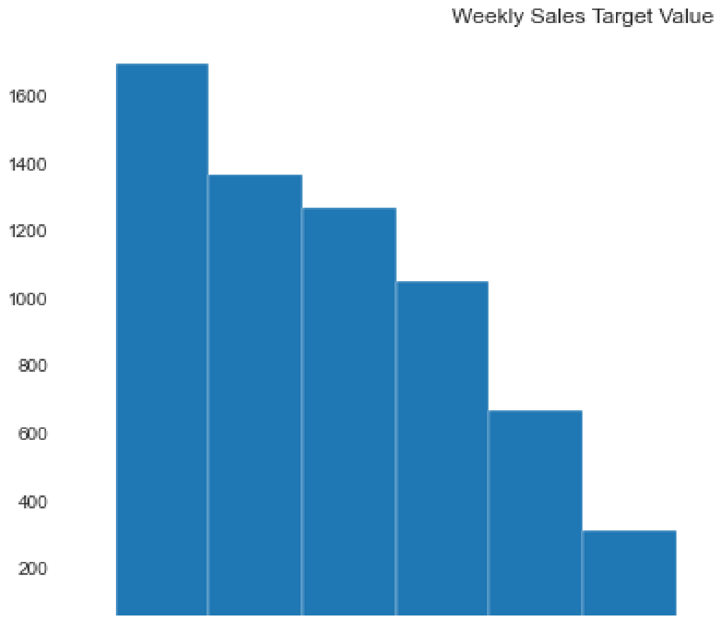
In [177]:

```
plt.figure(figsize=(10,6))
plt.hist(df['Store'])
plt.title('Store variable')

plt.figure(figsize=(10,6))
plt.hist(df['Weekly_Sales'])
plt.title('Weekly Sales Target Value')

plt.figure(figsize=(10,6))
plt.hist(df['Fuel_Price'])
plt.title('Fuel Price variable')

plt.figure(figsize=(10,6))
plt.hist(df['Unemployment'])
plt.title('Unemployment Rate variable')

plt.figure(figsize=(10,6))
plt.hist(df['CPI'])
plt.title('CPI variable')
```

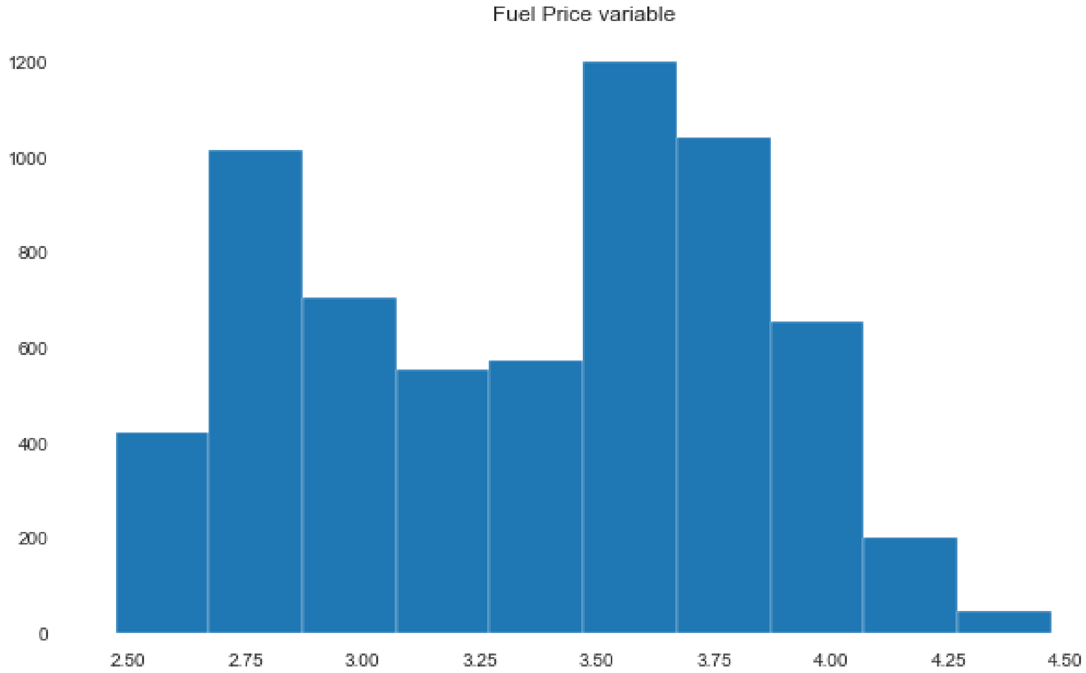
Out[177]:

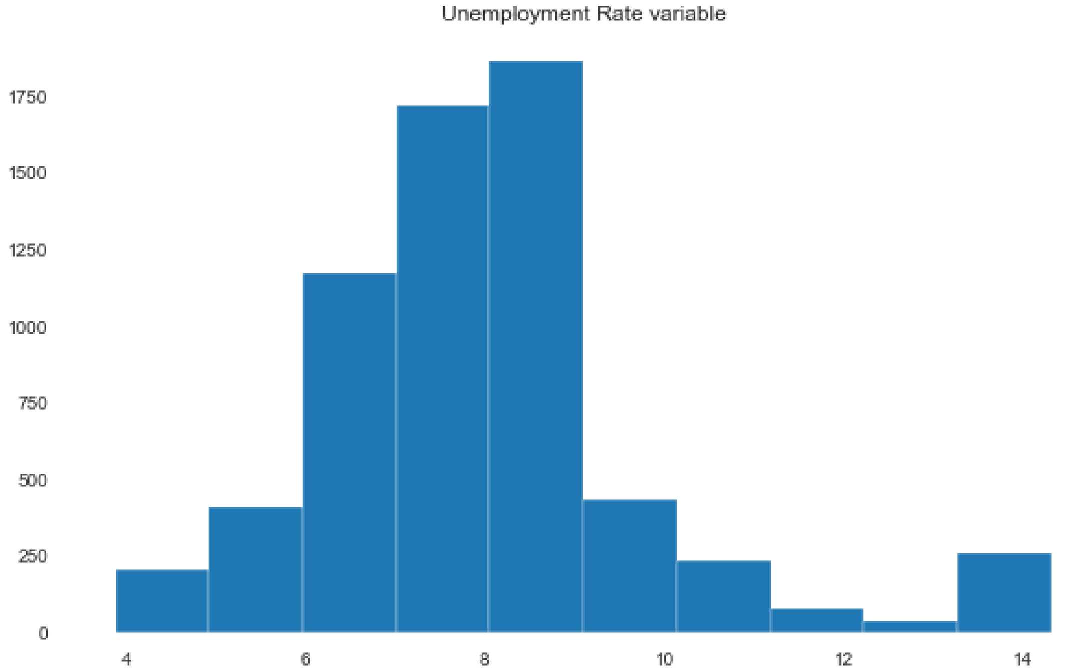
```
Text(0.5, 1.0, 'CPI variable')
```

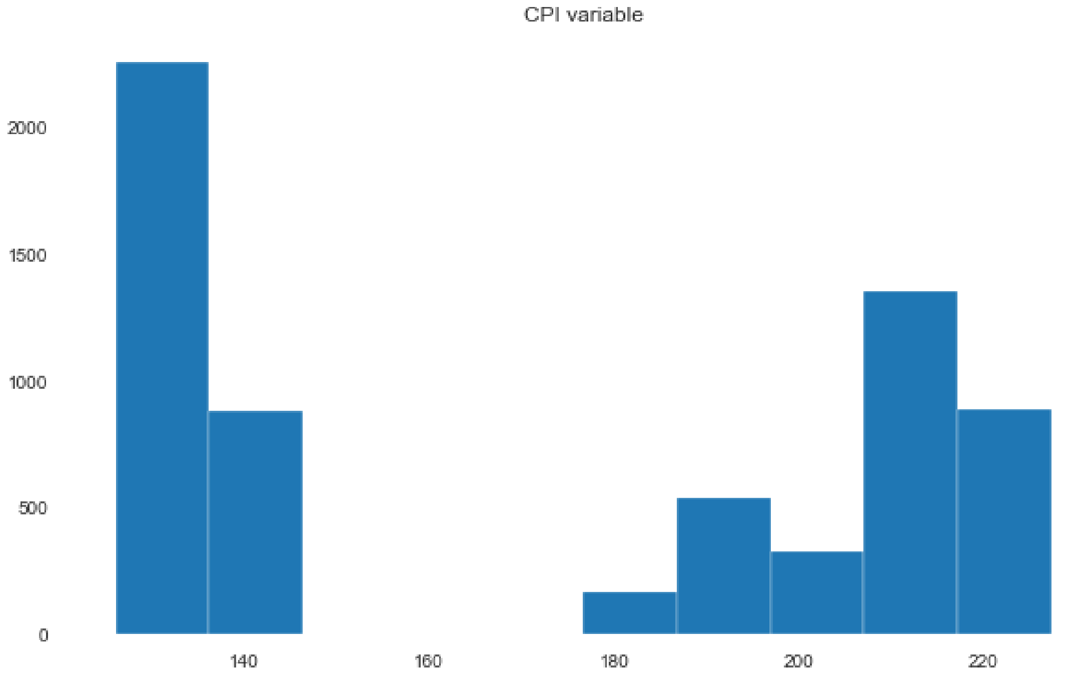




0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
1e6


Fuel Price variable
1200
1000
800
600
400
200
0
2.50
2.75
3.00
3.25
3.50
3.75
4.00
4.25
4.50


Unemployment Rate variable
1750
1500
1250
1000
750
500
250
0
4
6
8
10
12
14


CPI variable
2000
1500
1000
500
0
140
160
180
200
220

Descriptive Statistics anaysis using pandas profiling

In [115]:

```
pfr = pandas_profiling.ProfileReport(df)
pfr.to_file("Descriptive_Analysis_Walmart.html")
```

In [116]:

```
pfr
```

Out[116]:

Question 1. Which store has maximum sales ?

In [117]:

In [117]:

```
max_sales = df.groupby('Store')['Weekly_Sales'].sum()
max_sales.idxmax()
```

Out[117]:

```
20
```

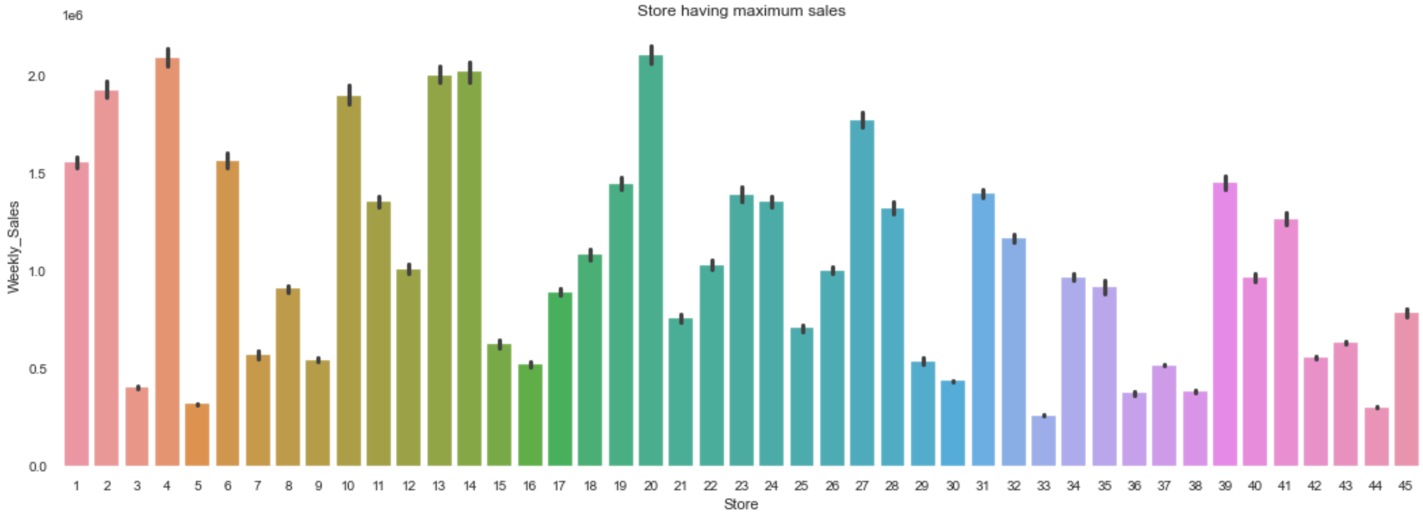
Store 20 has maximum Weekly Sales

In [118]:

```
plt.figure(figsize=(18,6))
sns.barplot(x=df.Store, y = df.Weekly_Sales)
plt.title('Store having maximum sales')
```

Out[118]:

```
Text(0.5, 1.0, 'Store having maximum sales')
```



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

In [119]:

```
max_std = df.groupby('Store')['Weekly_Sales'].std()
max_std.idxmax()
```

Out[119]:

```
14
```

In [120]:

```
max_cov = ((df.groupby('Store')['Weekly_Sales'].std())/(df.groupby('Store')['Weekly_Sale
s'].mean()))*100
max_cov.idxmax()
```

Out[120]:

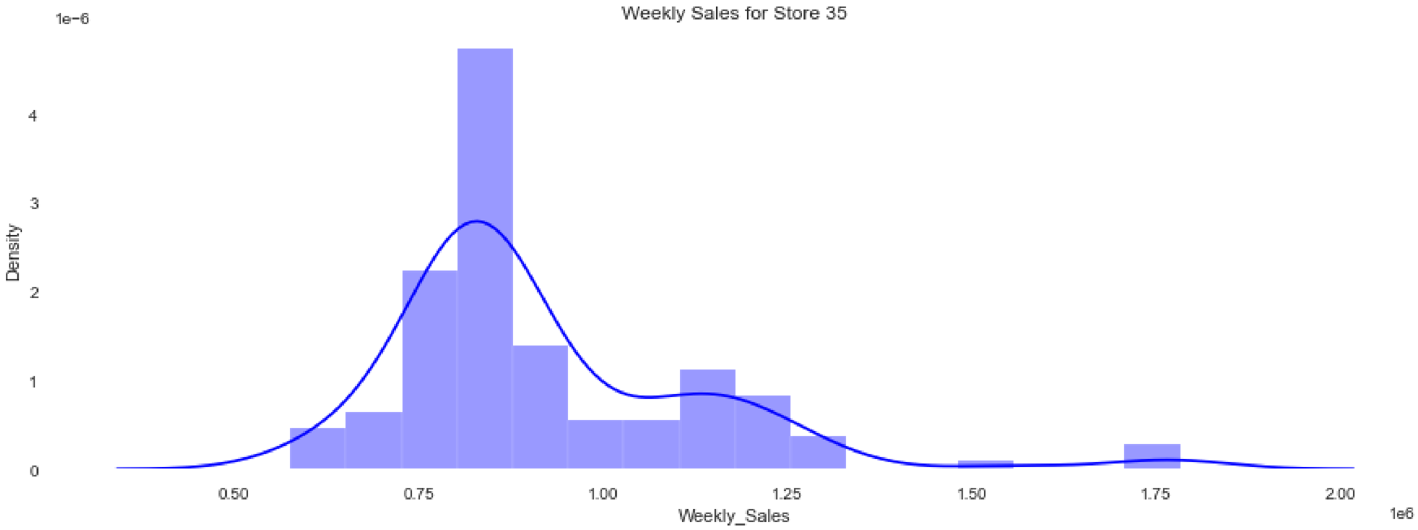
```
35
```

In [121]:

```
stores = df.groupby('Store')
store_35 = stores.get_group(35)
plt.figure(figsize=(15,5))
sns.distplot(store_35.Weekly_Sales, color='blue', label='Weekly Sales for Store 35')
plt.title('Weekly Sales for Store 35')
```

Out[121]:

```
Text(0.5, 1.0, 'Weekly Sales for Store 35')
```



Store 14 has maximum Standard deviation

Coefficient of mean to standard deviation is maximum for Store 35

In [122]:

```
from datetime import date
```

In [123]:

```
#Converting the data type of date column to dateTime

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

Q2data=df[(df['Date'] > Q2_date_from) & (df['Date'] < Q2_date_to)]
Q3data=df[(df['Date'] > Q3_date_from) & (df['Date'] < Q3_date_to)]

#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)

#finding the sum weekly sales of each store in Q2
Q3 = pd.DataFrame(Q3data.groupby('Store')['Weekly_Sales'].sum())
Q3.reset_index(inplace=True)
Q3.rename(columns={'Weekly_Sales': 'Q3_Weekly_Sales'},inplace=True)

#mergeing Q2 and Q3 data on Store as a common column
Q3_Growth= Q2.merge(Q3,how='inner',on='Store')

#Calculating Growth rate of each Store and collecting it into a dataframe
Q3_Growth['Growth_Rate'] =(Q3_Growth['Q3_Weekly_Sales'] - Q3_Growth['Q2_Weekly_Sales'])/
Q3_Growth['Q2_Weekly_Sales']
Q3_Growth['Growth_Rate']=round(Q3_Growth['Growth_Rate'],2)
Q3_Growth.sort_values('Growth_Rate',ascending=False).head(1)
```

Out[123]:

	Store	Q2_Weekly_Sales	Q3_Weekly_Sales	Growth_Rate
15	16	6626133.44	6441311.11	-0.03

In [124]:

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

Out[124]:

	Store	Q2_Weekly_Sales	Q3_Weekly_Sales	Growth_Rate
13	14	24427769.06	20140430.4	-0.18

In [125]:

```
Q3_Growth.head()
```

Out[125]:

	Store	Q2_Weekly_Sales	Q3_Weekly_Sales	Growth_Rate
0	1	21036965.58	18633209.98	-0.11
1	2	25085123.61	22396867.61	-0.11
2	3	5562668.16	4966495.93	-0.11
3	4	28384185.16	25652119.35	-0.10
4	5	4427262.21	3880621.88	-0.12

For Q3 the Store 16 has the least loss of 3% compared the other stores and store 14 has highest loss of 18%.

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

In [126]:

```
#finding the mean sales of non holiday and holiday
df.groupby('Holiday_Flag')['Weekly_Sales'].mean()
```

Out[126]:

```
Holiday_Flag
0    1.041256e+06
1    1.122888e+06
Name: Weekly_Sales, dtype: float64
```

In [127]:

```
# Marking the holiday dates
df['Date'] = pd.to_datetime(df['Date'])

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

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

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

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

```
#Calculating the mean sales during the holidays
Christmas_mean_sales=df[(df['Date'] == Christmas1) | (df['Date'] == Christmas2) | (df['D
ate'] == Christmas3) | (df['Date'] == Christmas4)]
Thanksgiving_mean_sales=df[(df['Date'] == Thanksgiving1) | (df['Date'] == Thanksgiving2)
 | (df['Date'] == Thanksgiving3) | (df['Date'] == Thanksgiving4)]
LabourDay_mean_sales=df[(df['Date'] == LabourDay1) | (df['Date'] == LabourDay2) | (df['D
ate'] == LabourDay3) | (df['Date'] == LabourDay4)]
SuperBowl_mean_sales=df[(df['Date'] == SuperBowl1) | (df['Date'] == SuperBowl2) | (df['D
ate'] == SuperBowl3) | (df['Date'] == SuperBowl4)]
Christmas_mean_sales

list_of_mean_sales = {'Christmas_mean_sales' : round(Christmas_mean_sales['Weekly_Sales']
.mean(),2),
'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' : round(df[df['Holiday_Flag'] == 0 ]['Weekly_Sales'].mean(),2
)}
list_of_mean_sales
```

Out[127]:

```
{'Christmas_mean_sales': 960833.11,
 'Thanksgiving_mean_sales': 1471273.43,
 'LabourDay_mean_sales': 1039182.83,
 'SuperBowl_mean_sales': nan,
 'Non holiday weekly sales': 1041256.38}
```

Thanksgiving has much higher sale than the rest of the holidays

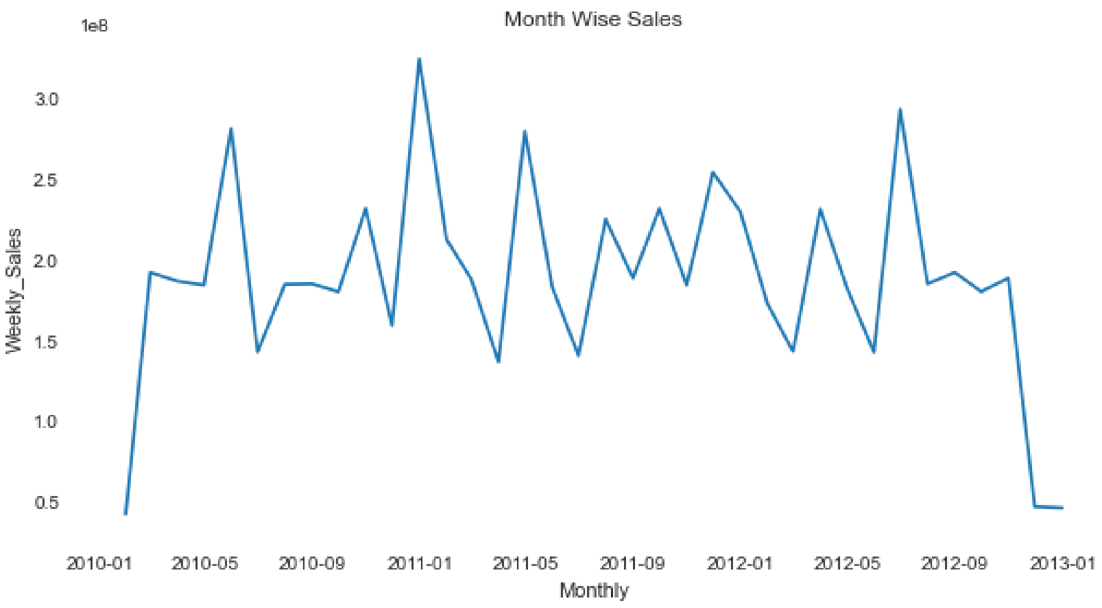
Question 5 Provide a monthly and semester view of sales in units and give insights

In [128]:

```
monthly = df.groupby(pd.Grouper(key='Date', freq='1M')).sum()  # groupby each 1 month
monthly=monthly.reset_index()
fig, ax = plt.subplots(figsize=(10,5))
X = monthly['Date']
Y = monthly['Weekly_Sales']
plt.plot(X,Y)
plt.title('Month Wise Sales')
plt.xlabel('Monthly')
plt.ylabel('Weekly_Sales')
```

Out[128]:

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



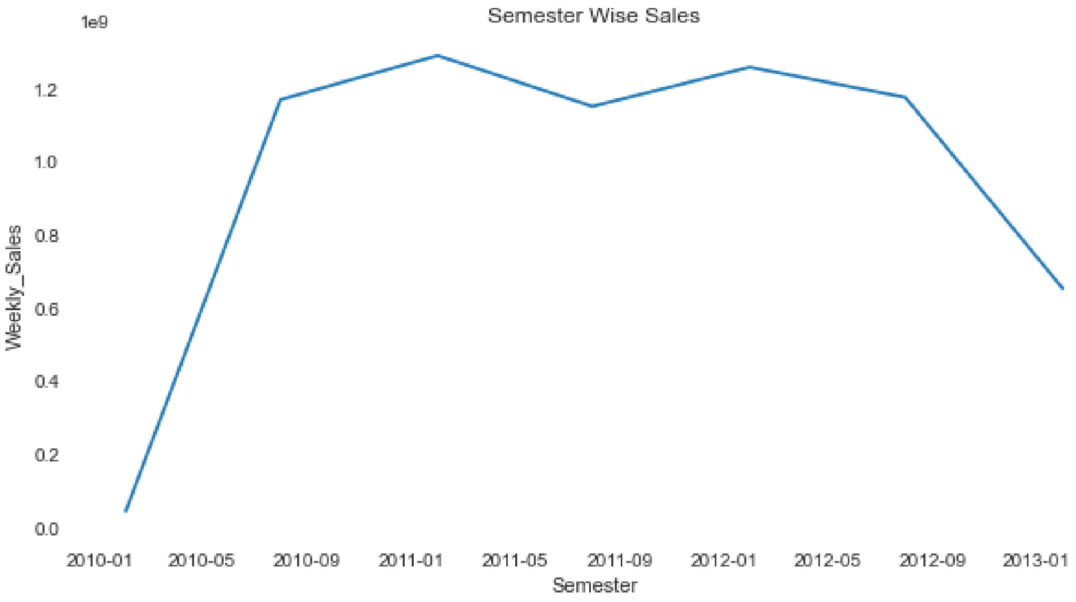
In [129]:

In [129]:

```
Semester = df.groupby(pd.Grouper(key='Date', freq='6M')).sum()
Semester = Semester.reset_index()
fig, ax = plt.subplots(figsize=(10,5))
X = Semester['Date']
Y = Semester['Weekly_Sales']
plt.plot(X,Y)
plt.title('Semester Wise Sales')
plt.xlabel('Semester')
plt.ylabel('Weekly_Sales')
```

Out[129]:

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



prediction models to forecast demand for Store 1

In [130]:

```
df_Store1= df[df['Store']==1]
```

In [131]:

```
df_Store1
```

Out[131]:

	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
...	...	...	...	...	...	...	...	...
138	1	2012-09-28	1437059.26	0	76.08	3.666	222.981658	6.908
139	1	2012-05-10	1670785.97	0	68.55	3.617	223.181477	6.573
140	1	2012-12-10	1573072.81	0	62.99	3.601	223.381296	6.573
141	1	2012-10-19	1508068.77	0	67.97	3.594	223.425723	6.573
142	1	2012-10-26	1493659.74	0	69.16	3.506	223.444251	6.573

143 rows × 8 columns

In [132]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [133]:

```
x = df_Store1.drop(['Weekly_Sales','Date'], axis=1)
y = df_Store1['Weekly_Sales']
```

In [151]:

```
date_cols = ['Date']
df = pd.read_csv('Walmart_Store_sales.csv', parse_dates=date_cols)
df['Days'] = pd.to_datetime(df['Date']).dt.day
df['Month'] = pd.to_datetime(df['Date']).dt.month
df['Year'] = pd.to_datetime(df['Date']).dt.year

df.head()
```

Out[151]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Days	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

Here I have created 3 columns for days, Month and year, as it was reading date as object.

Also for the last question , where I have to change dates into days, I have done it here itself in the prediction model.

In [152]:

```
x = df_Store1.drop(['Weekly_Sales','Date'], axis=1)
y = df_Store1['Weekly_Sales']
```

In [153]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.8,random_state = 4
2)
```

In [154]:

```
x_train.shape, y_test.shape
```

Out[154]:

```
((28, 6), (115,))
```

In [155]:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x,y)
```

Out[155]:

```
LinearRegression()
```

In [156]:

```
Walmart_Store1 = sm.OLS(y_train,x_train).fit()
```

In [157]:

```
Walmart_Store1.summary2()
```

Out[157]:

Model:	OLS	**Adj. R-squared:**	-0.095
Dependent Variable:	Weekly_Sales	**AIC:**	737.3294
Date:	2021-07-03 15:00	**BIC:**	745.3227
No. Observations:	28	**Log-Likelihood:**	-362.66
Df Model:	5	**F-statistic:**	0.5302
Df Residuals:	22	**Prob (F-statistic):**	0.751
R-squared:	0.108	**Scale:**	1.3259e+10

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Store	1292318.7578	3658419.9150	0.3532	0.7273	-6294779.7749	8879417.2905
Holiday_Flag	32614.8085	116387.6651	0.2802	0.7819	-208758.4356	273988.0527
Temperature	-1481.4080	1789.2722	-0.8279	0.4166	-5192.1315	2229.3156
Fuel_Price	61250.9567	92659.3812	0.6610	0.5155	-130912.8386	253414.7520
CPI	615.1554	13927.2249	0.0442	0.9652	-28268.1413	29498.4521
Unemployment	-7.4146	121110.0007	-0.0001	1.0000	-251174.1832	251159.3541

Omnibus:	4.478	**Durbin-Watson:**	1.552
Prob(Omnibus):	0.107	**Jarque-Bera (JB):**	2.849
Skew:	0.529	**Prob(JB):**	0.241
Kurtosis:	4.151	**Condition No.:**	38310

In [40]:

```
hypothesis = df.groupby('Store')[['Fuel_Price','Unemployment', 'CPI','Weekly_Sales', 'Holiday_Flag']]
factors  = hypothesis.get_group(1) #Filter by Store 1
day = [1]
for i in range (1,len(factors)):
    day.append(i*7)

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

Out[40]:

	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

	Fuel_Price	Unemployment	CPI	Weekly_Sales	Holiday_Flag	Day

143 rows × 6 columns

In [65]:

```
from scipy.stats import ttest_ind
```

Hypothesize if CPI has any impact on the sales

We will perform paired sample t-test between CPI and Sales , to check if CPI has any impact on Sales or Not.

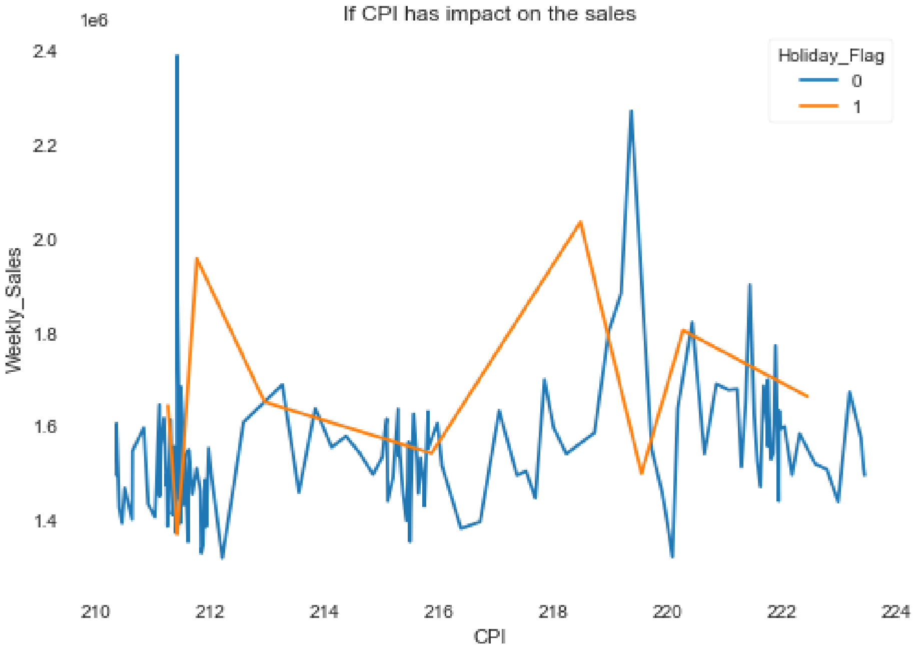
In [66]:

```
ttest,pval = stats.ttest_rel(factors['CPI'], factors['Weekly_Sales'])
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")

sns.lineplot(x='CPI', y = 'Weekly_Sales', data = factors, hue = 'Holiday_Flag')
plt.title('If CPI has impact on the sales')
```

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

Out[66]:

```
Text(0.5, 1.0, 'If CPI has impact on the sales')
```



Hypothesize if Unemployement Rate has any impact on the sales

We will perform paired sample t-test between Unemployment Rate and Weekly Sales , to check if Unemployment rate has any impact on Weekly Sales or Not.

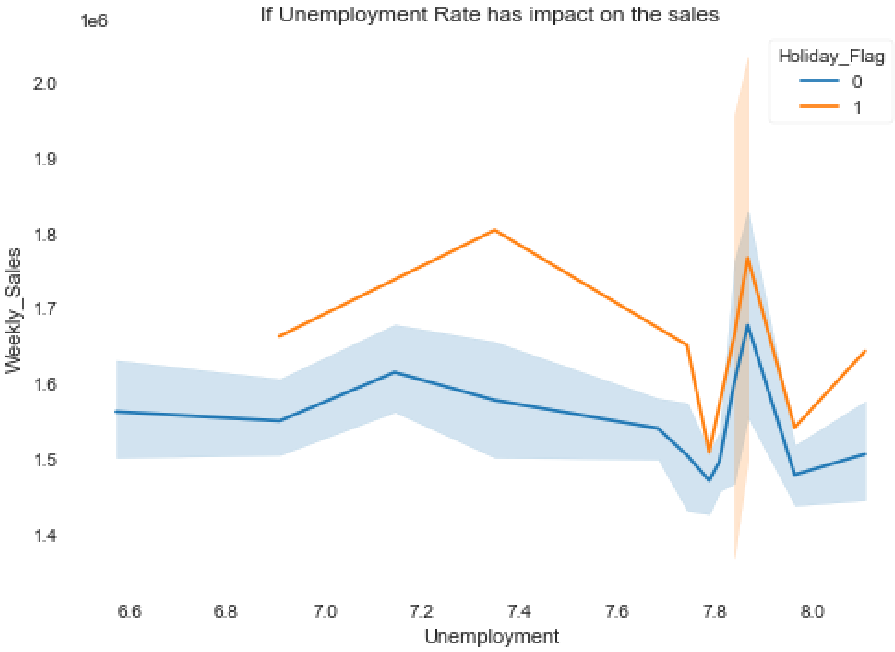
In [67]:

```
ttest,pval = stats.ttest_rel(factors['Unemployment'], factors['Weekly_Sales'])
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")

sns.lineplot(x='Unemployment', y = 'Weekly_Sales', data = factors, hue = 'Holiday_Flag')

plt.title('If Unemployment Rate has impact on the sales')
```

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

Out[67]:

```
Text(0.5, 1.0, 'If Unemployment Rate has impact on the sales')
```



Hypothesize if Fuel Price has any impact on the sales

We will perform paired sample t-test between Fuel Price and Weekly Sales , to check if Fuel Price has any impact on Weekly Sales or Not

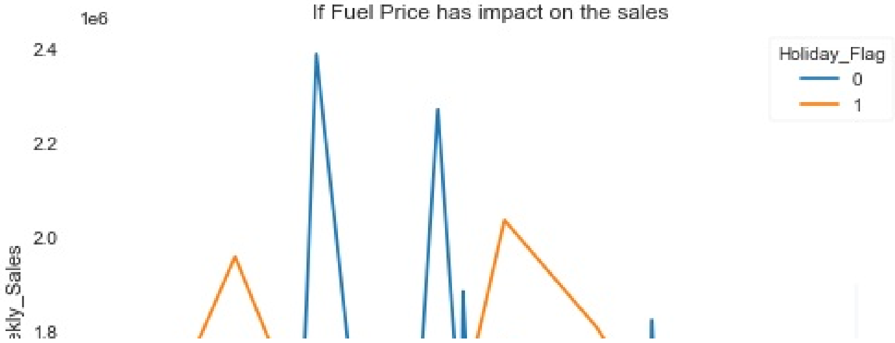
In [52]:

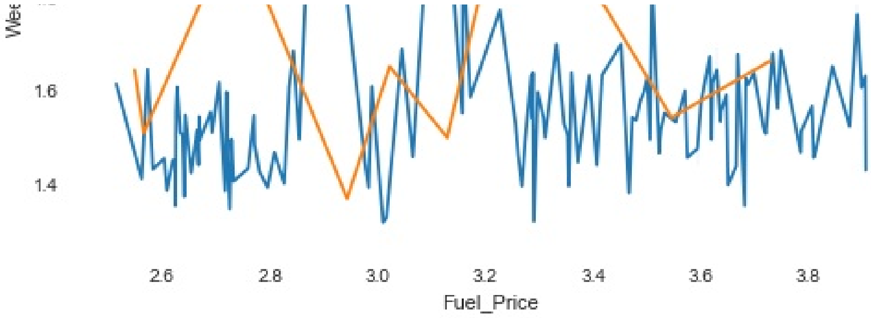
```
ttest,pval = stats.ttest_rel(factors['Fuel_Price'], factors['Weekly_Sales'])
print(pval)
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")

sns.lineplot(x='Fuel_Price', y = 'Weekly_Sales', data = factors, hue = 'Holiday_Flag')
plt.title('If Fuel Price has impact on the sales')
```

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

Out[52]:

```
Text(0.5, 1.0, 'If Fuel Price has impact on the sales')
```





In [158]:

```
# The part where , I have to change dates into days, Ihave done above the prediction mode
.
```