In [1]:
```
# importing libraries
import pandas as pd
import numpy as np
```
In [2]:
```
df = pd.read_csv('Walmart_Store_sales.csv')
```
In [3]:
```
df.head()
```
Out[3]:

	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 [4]:
```
df.shape
```
Out[4]:
```
(6435, 8)
```
In [5]:
```
df.dtypes
```
Out[5]:
```
Store              int64
Date              object
Weekly_Sales     float64
Holiday_Flag       int64
Temperature      float64
Fuel_Price       float64
CPI              float64
Unemployment     float64
dtype: object
```
In [6]:
```
# changing the columns to lower case

df.columns  = df.columns.str.lower()
```
In [7]:
```
df.columns
```
Out[7]:
```
Index(['store', 'date', 'weekly_sales', 'holiday_flag', 'temperature',
       'fuel_price', 'cpi', 'unemployment'],
      dtype='object')
```

1) Which store has maximum values

In [8]:
```
df.groupby('store')['weekly_sales'].sum().head()
```
Out[8]:
```
store
1    2.224028e+08
2    2.753824e+08
3    5.758674e+07
4    2.995440e+08
5    4.547569e+07
Name: weekly_sales, dtype: float64
```
In [9]:
```
max_sales = df.groupby('store')['weekly_sales'].sum()
```
In [10]:
```
max_sales.head()
```
Out[10]:
```
store
1    2.224028e+08
2    2.753824e+08
3    5.758674e+07
4    2.995440e+08
5    4.547569e+07
Name: weekly_sales, dtype: float64
```
In [11]:
```
max_sales.index[max_sales.argmax()]
```
Out[11]:
```
20
```
Conclusion :

```
- Store 20 has maximum sales
```
In [ ]:

2)Store having maximum standard deviation

In [12]:
```
max_std = df.groupby('store')['weekly_sales'].std()
max_std.head()
```
Out[12]:
```
store
1    155980.767761
2    237683.694682
3     46319.631557
4    266201.442297
5     37737.965745
Name: weekly_sales, dtype: float64
```
In [13]:
```
max_std.index[max_std.argmax()]
```
Out[13]:
```
14
```
Conclusion :

```
- Store 14 has maximum standard deviation
```

coefficient of mean to standard deviation

In [14]:
```
max_mean = df.groupby('store')['weekly_sales'].mean()
max_mean.head()
```
Out[14]:
```
store
1    1.555264e+06
2    1.925751e+06
3    4.027044e+05
4    2.094713e+06
5    3.180118e+05
Name: weekly_sales, dtype: float64
```
In [15]:
```
cv = max_std/max_mean * 100
```

Coefficient of variation - overall weeklysales

In [16]:
```
cv.head()
```
Out[16]:
```
store
1    10.029212
2    12.342388
3    11.502141
4    12.708254
5    11.866844
Name: weekly_sales, dtype: float64
```
In [17]:
```
mean_14 = df.groupby('store')['weekly_sales'].get_group(14).mean()
```
In [18]:
```
std_14 = max_std[max_std.argmax()]
```

coefficient of variation - 14th store

In [19]:
```
cv_14 = mean_14/std_14
cv_14
```
Out[19]:
```
7.61177081257525
```
In [ ]:

3) Store/s having good quarterly growth rate in Q3'2012

In [20]:
```
# Extraction year and month from the date variable

df['year']=pd.DatetimeIndex(df['date']).year
df['month']=pd.DatetimeIndex(df['date']).month
```
In [21]:
```
df.head()
```
Out[21]:

	store	date	weekly_sales	holiday_flag	temperature	fuel_price	cpi	unemployment	year	month
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106	2010	5
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106	2010	12
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106	2010	2
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106	2010	2
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106	2010	5

In [22]:
```
# Group_by 2012

df['quartile'] = 0
quarterly = df.groupby('year').get_group(2012)
quarterly.head()
```
Out[22]:

	store	date	weekly_sales	holiday_flag	temperature	fuel_price	cpi	unemployment	year	month	quartile
100	1	06-01-2012	1550369.92	0	49.01	3.157	219.714258	7.348	2012	6	0
101	1	13-01-2012	1459601.17	0	48.53	3.261	219.892526	7.348	2012	1	0
102	1	20-01-2012	1394393.84	0	54.11	3.268	219.985689	7.348	2012	1	0
103	1	27-01-2012	1319325.59	0	54.26	3.290	220.078852	7.348	2012	1	0
104	1	03-02-2012	1636339.65	0	56.55	3.360	220.172015	7.348	2012	3	0

In [23]:
```
pd.options.mode.chained_assignment = None
```
In [24]:
```
# Defining Quarterly range using for-loop

for i in quarterly['month']:
    if i in [4,5,6]:
        quarterly['quartile'][quarterly[quarterly['month']==i].index] = 'q2'
    elif i in [7,8,9]:
        quarterly['quartile'][quarterly[quarterly['month']==i].index] = 'q3'
```
In [25]:
```
quarterly.head()
```
Out[25]:

	store	date	weekly_sales	holiday_flag	temperature	fuel_price	cpi	unemployment	year	month	quartile
100	1	06-01-2012	1550369.92	0	49.01	3.157	219.714258	7.348	2012	6	q2
101	1	13-01-2012	1459601.17	0	48.53	3.261	219.892526	7.348	2012	1	0
102	1	20-01-2012	1394393.84	0	54.11	3.268	219.985689	7.348	2012	1	0
103	1	27-01-2012	1319325.59	0	54.26	3.290	220.078852	7.348	2012	1	0
104	1	03-02-2012	1636339.65	0	56.55	3.360	220.172015	7.348	2012	3	0

In [26]:
```
# grouping the q2 datas

q2 = quarterly.groupby('quartile').get_group('q2').groupby('store')['weekly_sales'].sum()
q2.head()
```
Out[26]:
```
store
1    21036965.58
2    25085123.61
3     5562668.16
4    28384185.16
5     4427262.21
Name: weekly_sales, dtype: float64
```
In [27]:
```
# Grouping q3 datas

q3 = quarterly.groupby('quartile').get_group('q3').groupby('store')['weekly_sales']
q3 = q3.sum()
q3.head()
```
Out[27]:
```
store
1    18633209.98
2    22396867.61
3     4966495.93
4    25652119.35
5     3880621.88
Name: weekly_sales, dtype: float64
```
In [28]:
```
Q3_total = q3 -q2
Q3_total.head()
```
Out[28]:
```
store
1   -2403755.60
2   -2688256.00
3    -596172.23
4   -2732065.81
5    -546640.33
Name: weekly_sales, dtype: float64
```
In [29]:
```
Q3_total.index[Q3_total.argmax()]
```
Out[29]:
```
16
```
Conclusion:

- Quarterly growth rate for stores is not good in 2012

In [ ]:

4) Find out holidays which have higher sales than the mean sales in non-holiday season for all stores together.

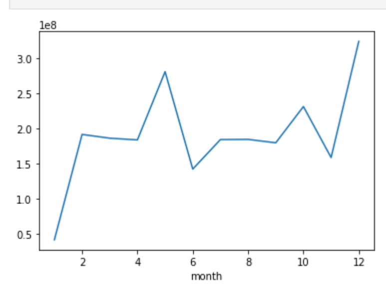
In [30]:
```
copy_data = pd.read_csv('Walmart_Store_sales.csv')
```
In [31]:
```
copy_data['Date'] = pd.to_datetime(copy_data['Date'],format='%d-%m-%Y')
```
In [32]:
```
# creating dataframe for superbowl holidays sales

superbowl_df = copy_data[(copy_data['Date']=='2010-02-12')|(copy_data['Date']=='2011-02-11')|(copy_data['Dat
```
In [33]:
```
# creating dataframe for labour_day holidays sales

labour_day_df = copy_data[(copy_data['Date']=='2010-09-10') |(copy_data['Date']=='2011-09-09')|(copy_data['D
```
In [34]:
```
# creating dataframe for thanksgiving holidays sales

thanksgiving_df = copy_data[(copy_data['Date']=='2010-11-26') |(copy_data['Date']=='2011-11-25')|(copy_data[
```
In [35]:
```
# creating dataframe for christmas holidays sales

christmas_df = copy_data[(copy_data['Date']=='2010-12-31') |(copy_data['Date']=='2011-12-30')|(copy_data['Da
                                |(copy_data['Date']=='2013-12-27') ]
```
In [36]:
```
superbowl_df['Weekly_Sales'].mean() > thanksgiving_df['Weekly_Sales'].mean()
```
Out[36]:
```
False
```
In [37]:
```
superbowl_df['Weekly_Sales'].mean() > labour_day_df['Weekly_Sales'].mean()
```
Out[37]:
```
True
```
In [38]:
```
superbowl_df['Weekly_Sales'].mean() > christmas_df['Weekly_Sales'].mean()
```
Out[38]:
```
True
```
Mean sales of Non-holiday sales

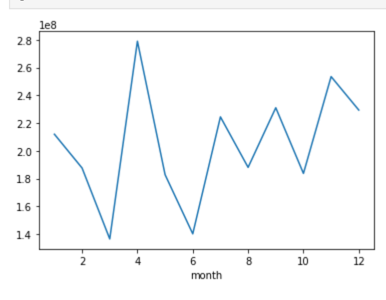
In [39]:
```
# Grouping Non holidays using holiday_flag from dataset

holiday_grp = copy_data.groupby('Holiday_Flag').get_group(0)[['Date','Weekly_Sales']]
```
In [40]:
```
holiday_grp.Weekly_Sales.mean()
```
Out[40]:
```
1041256.3802088564
```
In [41]:
```
superbowl_df['Weekly_Sales'].mean() > holiday_grp.Weekly_Sales.mean()
```
Out[41]:
```
True
```
In [42]:
```
labour_day_df['Weekly_Sales'].mean() > holiday_grp['Weekly_Sales'].mean()
```
Out[42]:
```
True
```
In [43]:
```
thanksgiving_df['Weekly_Sales'].mean() > holiday_grp['Weekly_Sales'].mean()
```
Out[43]:
```
True
```
In [44]:
```
christmas_df['Weekly_Sales'].mean() > holiday_grp['Weekly_Sales'].mean()
```
Out[44]:
```
False
```

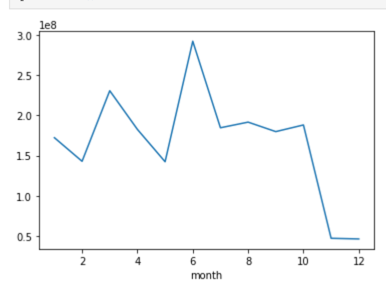
Thanksgiving, superbowl,Labour days has higher mean sales than mean of non-holidays for all stores together

In [ ]:

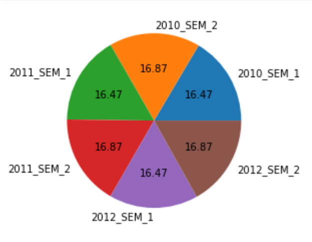
Provide a monthly and semester view of sales in units and give insights

In [45]:
```
import matplotlib.pyplot as plt
```
In [46]:
```
# grouping 2010 datas
year10 = df.groupby('year').get_group(2010).groupby('month')['weekly_sales'].sum()
year10.head()
```
Out[46]:
```
month
1    4.223988e+07
2    1.915869e+08
3    1.862262e+08
4    1.838118e+08
5    2.806119e+08
Name: weekly_sales, dtype: float64
```
In [47]:
```
year10.plot(x='month',y='weekly_sales')
# plt.xticks(year10['month'],rotation='vertical',size=10)
# plt.ylabel("Weekly_Sales $")
plt.show()
```


In [48]:
```
year11 = df.groupby('year').get_group(2011).groupby('month')['weekly_sales'].sum()
year11.head()
```
Out[48]:
```
month
1    2.119657e+08
2    1.876092e+08
3    1.365205e+08
4    2.789693e+08
5    1.828017e+08
Name: weekly_sales, dtype: float64
```
In [49]:
```
year11.plot(x='month',y='weekly_sales')
plt.show()
```


In [50]:
```
year12 = df.groupby('year').get_group(2012).groupby('month')['weekly_sales'].sum()
year12.head()
```
Out[50]:
```
month
1    1.722207e+08
2    1.428296e+08
3    2.307397e+08
4    1.825428e+08
5    1.422830e+08
Name: weekly_sales, dtype: float64
```
In [51]:
```
year12.plot(x='month',y='weekly_sales')
plt.show()
```


Semester view of sales

In [52]:
```
sem1_2010 = df.groupby('year').get_group(2010)
# Two semesters grouping of 2010
sem1_2010 = df[df['month'].isin([1,2,3,4,5,6])]
sem2_2010 = df[df['month'].isin([7,8,9,10,11,12])]
```
In [53]:
```
sem1_2011 = df.groupby('year').get_group(2011)
# Semester grouping of 2011
sem1_2011 = df[df['month'].isin([1,2,3,4,5,6])]
sem2_2011 = df[df['month'].isin([7,8,9,10,11,12])]
```
In [54]:
```
sem1_2012 = df.groupby('year').get_group(2012)
# Semester grouping of 2012
sem1_2012 = df[df['month'].isin([1,2,3,4,5,6])]
sem2_2012 = df[df['month'].isin([7,8,9,10,11,12])]
```
In [55]:
```
# Pie-chart view of semester view of 2010,2011,2012

lst = [sem1_2010['weekly_sales'].sum(),sem2_2010['weekly_sales'].sum(),
       sem1_2011['weekly_sales'].sum(),sem2_2011['weekly_sales'].sum(),
       sem1_2012['weekly_sales'].sum(),sem2_2012['weekly_sales'].sum()]

label = ['2010_SEM_1','2010_SEM_2','2011_SEM_1','2011_SEM_2','2012_SEM_1','2012_SEM_2']
plt.pie(lst,labels=label,autopct='%.2f')
plt.show()
```


In [ ]:

Statistical Task

Utilize variables like date and restructure dates as 1 for 5 Feb 2010

In [56]:
```
store1_restruct = df.groupby('year').get_group(2010).groupby('store').get_group(1)
store1_restruct.head()
```
Out[56]:

	store	date	weekly_sales	holiday_flag	temperature	fuel_price	cpi	unemployment	year	month	quartile
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106	2010	5	0
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106	2010	12	0
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106	2010	2	0
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106	2010	2	0
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106	2010	5	0

In [57]:
```
restructured = []
for i in range(1,len(store1_restruct['weekly_sales'])+1):
    restructured.append(i)

store1_restruct['restructured'] = restructured
store1_restruct.drop(columns=['year','quartile','month'],inplace=True)
store1_restruct.head()
```
Out[57]:

	store	date	weekly_sales	holiday_flag	temperature	fuel_price	cpi	unemployment	restructured
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106	1
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106	2
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106	3
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106	4
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106	5

Linear model for hypothesizing between variables

In [58]:
```
store_1 = df.groupby('store').get_group(1)
store_1.tail()
```
Out[58]:

	store	date	weekly_sales	holiday_flag	temperature	fuel_price	cpi	unemployment	year	month	quartile
138	1	28-09-2012	1437059.26	0	76.08	3.666	222.981658	6.908	2012	9	0
139	1	05-10-2012	1670785.97	0	68.55	3.617	223.181477	6.573	2012	5	0
140	1	12-10-2012	1573072.81	0	62.99	3.601	223.381296	6.573	2012	12	0
141	1	19-10-2012	1508068.77	0	67.97	3.594	223.425723	6.573	2012	10	0
142	1	26-10-2012	1493659.74	0	69.16	3.506	223.444251	6.573	2012	10	0

Model Development

In [59]:
```
from sklearn.model_selection import train_test_split as split
```
In [60]:
```
train,test = split(store_1,test_size=0.30,random_state = 12)
```
In [61]:
```
from sklearn.linear_model import LinearRegression
lr  = LinearRegression()
```
In [62]:
```
x = train[['cpi','unemployment','fuel_price']]
y = train['weekly_sales']
lr.fit(x,y)
```
Out[62]:
```
LinearRegression()
```
In [63]:
```
x_test = test[['cpi','unemployment','fuel_price']]
y_test = test['weekly_sales']

ypred = lr.predict(x_test)
```
In [64]:
```
print(lr.score(x,y))
```
```
0.10681532848100328
```
In [65]:
```
print(x.shape,y.shape,x_test.shape,y_test.shape)
```
```
(100, 3) (100,) (43, 3) (43,)
```
In [66]:
```
# shapes of predicted and test variables
print(ypred.shape,y_test.shape)
```
```
(43,) (43,)
```
In [67]:
```
# creating OLS summary
from statsmodels.formula.api import ols

mod= ols('weekly_sales ~ cpi + unemployment + fuel_price',data=train).fit()
mod.summary()
```
Out[67]:

OLS Regression Results

Dep. Variable:	weekly_sales	**R-squared:**	0.107
Model:	OLS	**Adj. R-squared:**	0.079
Method:	Least Squares	**F-statistic:**	3.827
Date:	Mon, 19 Oct 2020	**Prob (F-statistic):**	0.0123
Time:	11:07:23	**Log-Likelihood:**	-1313.5
No. Observations:	100	**AIC:**	2635.
Df Residuals:	96	**BIC:**	2645.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-2.097e+06	1.72e+06	-1.222	0.225	-5.5e+06	1.31e+06
cpi	1.547e+04	6720.811	2.302	0.023	2130.531	2.88e+04
unemployment	5.168e+04	5.61e+04	0.921	0.359	-5.97e+04	1.63e+05
fuel_price	-2.827e+04	4.6e+04	-0.615	0.540	-1.19e+05	6.29e+04

Omnibus:	19.798	**Durbin-Watson:**	2.092
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	27.195
Skew:	0.951	**Prob(JB):**	1.24e-06
Kurtosis:	4.706	**Cond. No.**	2.97e+04

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.97e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

conclusion from linearity

From applying the method, we can see that there is only 10% good fit relation.

Hypothesis:

```
- Ho : cpi,unemployment,fuel_price have impact on sales
- Ha : Have no impact on sales
```
Conclusion:

```
- Since the p_values of [cpi,unemployment,fuel_price] > alpha
- We fail to reject null hypothesis.
```

Change dates into days by creating new variable.

In [68]:
```
day_name = copy_data['Date'].dt.day_name()
```
In [69]:
```
copy_data.insert(2,'Day_name',day_name)
```
In [70]:
```
copy_data.head()
```
Out[70]:

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

In [ ]: