Retail Analysis with Walmart Data

August 19, 2020

[1]:
```
# Importing required libraries
import pandas as pd
from datetime import date
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
```

[2]:
```
# Import Data set

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

[3]:
```
# default display the first five rows from the dataset
df_walmart.head()
```

[3]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	\
0	1	05-02-2010	1643690.90	0	42.31	2.572	
1	1	12-02-2010	1641957.44	1	38.51	2.548	
2	1	19-02-2010	1611968.17	0	39.93	2.514	
3	1	26-02-2010	1409727.59	0	46.63	2.561	
4	1	05-03-2010	1554806.68	0	46.50	2.625	

	CPI	Unemployment
0	211.096358	8.106
1	211.242170	8.106
2	211.289143	8.106
3	211.319643	8.106
4	211.350143	8.106

[4]:
```
#basic infomation about dataset
df_walmart.info()
df_walmart.shape
```

```
<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
```

[4]:
```
(6435, 8)
```

[5]:
```
# find the maximum value in each column
df_walmart.max()
```

[5]:
```
Store                    45
Date             31-12-2010
Weekly_Sales    3.81869e+06
Holiday_Flag              1
Temperature          100.14
Fuel_Price            4.468
CPI                 227.233
Unemployment         14.313
dtype: object
```

[6]:
```
# which store has maximum sales in this dataset?
df_walmart.loc[df_walmart['Weekly_Sales'] == df_walmart['Weekly_Sales'].max()]
#maximum weekly sales
```

[6]:
```
      Store        Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
1905     14  24-12-2010    3818686.45             0        30.59       3.141

            CPI  Unemployment
1905  182.54459         8.724
```

[7]:
```
# Which store has maximum standard deviation i.e. the sales vary a lot. Also,␣
 ↪find out the coefficient of varience
#(cov)
# Grouping by store and finding the standard deviation and mean of each store.

maxstd=pd.DataFrame(df_walmart.groupby('Store').agg({'Weekly_Sales':
 ↪['std','mean']}))

# resetting the index.
maxstd=maxstd.reset_index()

# Now we know that COV is  std/mean we are doing this for each store.
```

```
maxstd['Cov'] = (maxstd[('Weekly_Sales','std')]/
 ↪maxstd[('Weekly_Sales','mean')])*100

#finding the store with maximum standard deviation.
maxstd.loc[maxstd[('Weekly_Sales','std')]==maxstd[('Weekly_Sales','std')].max()]
```

[7]:

	Store	Weekly_Sales		Cov
		std	mean	
13	14	317569.949476	2.020978e+06	15.713674

[8]:
```
# which store has good quarterly growth rate in Q3'2012
# converting the data type of date column to datetime
df_walmart['Date']= pd.to_datetime(df_walmart['Date'])
df_walmart.head()

# defining the start and end date of Q3 and Q2

Q2_date_from = pd.Timestamp(date(2012,4,1))
Q2_date_to = pd.Timestamp(date(2012,6,30))

Q3_date_from = pd.Timestamp(date(2012,7,1))
Q3_date_to = pd.Timestamp(date(2012,9,30))


#collecting the data of Q3 and Q2 from original dataset.
Q2data=df_walmart[(df_walmart['Date']> Q2_date_from) &␣
 ↪(df_walmart['Date']<Q2_date_to)]
Q3data=df_walmart[(df_walmart['Date']> Q3_date_from) &␣
 ↪(df_walmart['Date']<Q3_date_to)]

# Finding 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 sum weekly sales of each store in Q3

Q3 = pd.DataFrame(Q3data.groupby('Store')['Weekly_Sales'].sum())
Q3.reset_index(inplace=True)
Q3.rename(columns={'Weekly_Sales': 'Q3_Weekly_Sales'},inplace=True)

#merging Q2 and Q3 data on store as a common column
Q3_growth = Q2.merge(Q3,how='inner',on='Store')
```

[9]:
```
# 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)
```

[9]:

	Store	Q2_Weekly_Sales	Q3_Weekly_Sales	Growth_Rate
15	16	6626133.44	6441311.11	-0.03

[10]:
```
Q3_growth.sort_values('Growth_Rate',ascending=False).tail(1)
```

[10]:

	Store	Q2_Weekly_Sales	Q3_Weekly_Sales	Growth_Rate
13	14	24427769.06	20140430.4	-0.18

[11]:
```
#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
#finding the mean sales of holiday and non holiday

df_walmart.groupby('Holiday_Flag')['Weekly_Sales'].mean()
```

[11]:
```
Holiday_Flag
0    1.041256e+06
1    1.122888e+06
Name: Weekly_Sales, dtype: float64
```

[12]:
```
#marking the holiday dates
Christmas1 = pd.Timestamp(date(2010,12,31) )
Christmas2 = pd.Timestamp(date(2011,12,30) )
Christmas3 = pd.Timestamp(date(2012,12,28) )
Christmas4 = pd.Timestamp(date(2013,12,27) )

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

LabourDay1=pd.Timestamp(date(2010,2,10) )
LabourDay2=pd.Timestamp(date(2011,2,9) )
LabourDay3=pd.Timestamp(date(2012,2,7) )
LabourDay4=pd.Timestamp(date(2013,2,6) )

SuperBowl1=pd.Timestamp(date(2010,9,12) )
SuperBowl2=pd.Timestamp(date(2011,9,11) )
SuperBowl3=pd.Timestamp(date(2012,9,10) )
```

```
SuperBowl4=pd.Timestamp(date(2013,9,8) )
```

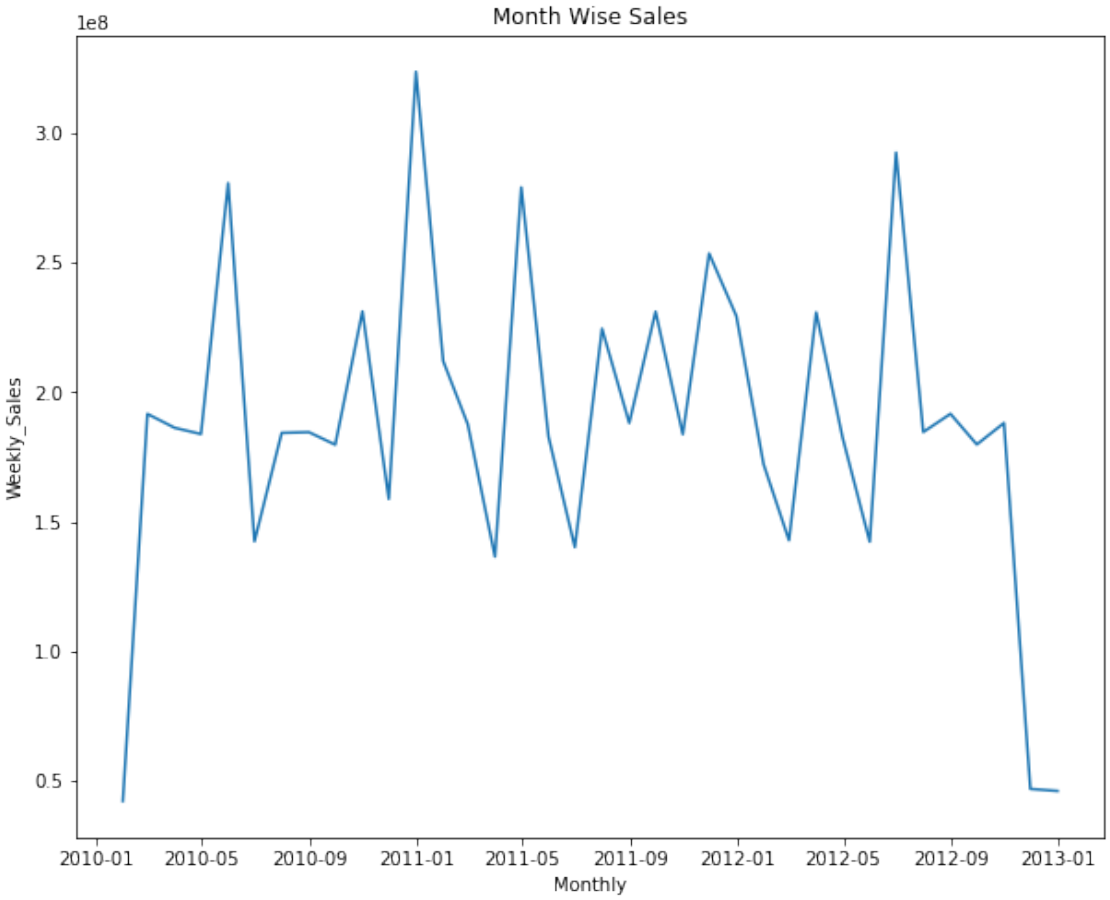
[13]:
```
#Calculating the mean sales during the holidays
Christmas_mean_sales=df_walmart[(df_walmart['Date'] == Christmas1) |␣
 ↪(df_walmart['Date'] == Christmas2) | (df_walmart['Date'] == Christmas3) |␣
 ↪(df_walmart['Date'] == Christmas4)]
Thanksgiving_mean_sales=df_walmart[(df_walmart['Date'] == Thanksgiving1) |␣
 ↪(df_walmart['Date'] == Thanksgiving2) | (df_walmart['Date'] ==␣
 ↪Thanksgiving3) | (df_walmart['Date'] == Thanksgiving4)]
LabourDay_mean_sales=df_walmart[(df_walmart['Date'] == LabourDay1) |␣
 ↪(df_walmart['Date'] == LabourDay2) | (df_walmart['Date'] == LabourDay3) |␣
 ↪(df_walmart['Date'] == LabourDay4)]
SuperBowl_mean_sales=df_walmart[(df_walmart['Date'] == SuperBowl1) |␣
 ↪(df_walmart['Date'] == SuperBowl2) | (df_walmart['Date'] == SuperBowl3) |␣
 ↪(df_walmart['Date'] == SuperBowl4)]
#
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' : df_walmart[df_walmart['Holiday_Flag'] == 0␣
 ↪]['Weekly_Sales'].mean()}
list_of_mean_sales
```

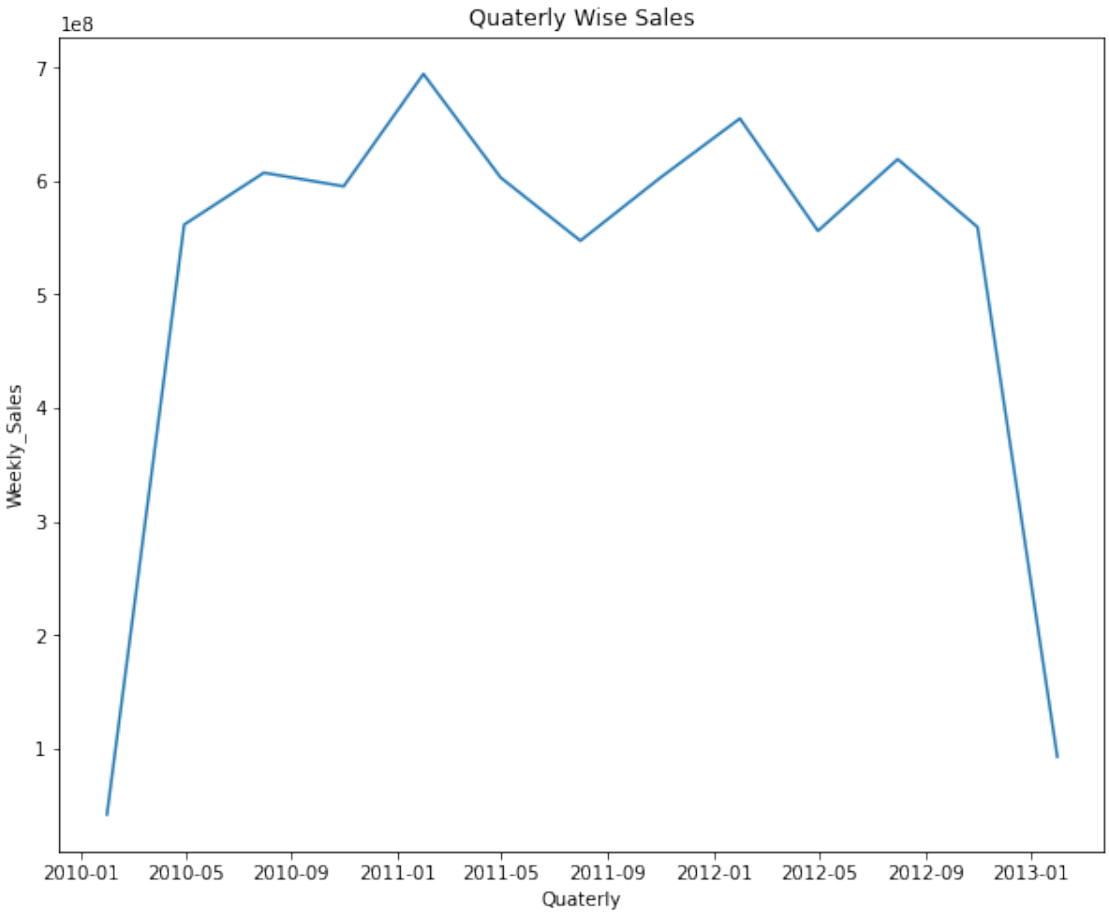
[13]:
```
{'Christmas_mean_sales': 960833.11,
 'Thanksgiving_mean_sales': 1471273.43,
 'LabourDay_mean_sales': 1008369.41,
 'SuperBowl_mean_sales': nan,
 'Non holiday weekly sales': 1041256.3802088564}
```

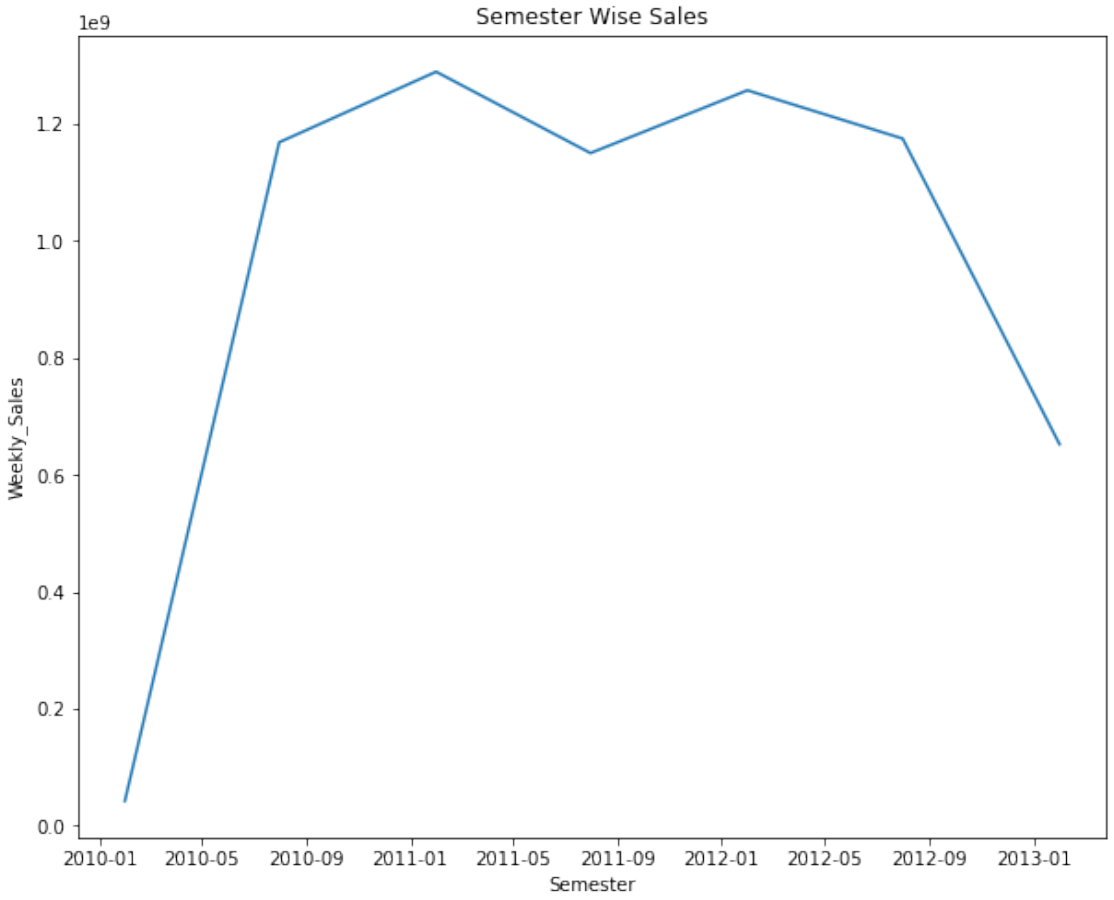
[14]:
```
#Provide a monthly and semester view of sales in units and give insights

#Monthly sales
monthly = df_walmart.groupby(pd.Grouper(key='Date', freq='1M')).sum()# groupby␣
 ↪each by month
monthly=monthly.reset_index()
fig, ax = plt.subplots(figsize=(10,8))
X = monthly['Date']
Y = monthly['Weekly_Sales']
plt.plot(X,Y)
plt.title('Month Wise Sales')
plt.xlabel('Monthly')
plt.ylabel('Weekly_Sales')
```

```
#Quaterly Sales
Quaterly = df_walmart.groupby(pd.Grouper(key='Date', freq='3M')).sum()
Quaterly = Quaterly.reset_index()
fig, ax = plt.subplots(figsize=(10,8))
X = Quaterly['Date']
Y = Quaterly['Weekly_Sales']
plt.plot(X,Y)
plt.title('Quaterly Wise Sales')
plt.xlabel('Quaterly')
plt.ylabel('Weekly_Sales')
#Semester Sales
Semester = df_walmart.groupby(pd.Grouper(key='Date', freq='6M')).sum()
Semester = Semester.reset_index()
fig, ax = plt.subplots(figsize=(10,8))
X = Semester['Date']
Y = Semester['Weekly_Sales']
plt.plot(X,Y)
plt.title('Semester Wise Sales')
plt.xlabel('Semester')
plt.ylabel('Weekly_Sales')
```

[14]: Text(0, 0.5, 'Weekly_Sales')


Month Wise Sales
1e8
3.0
2.5
2.0
1.5
1.0
0.5
Weekly_Sales
2010-01
2010-05
2010-09
2011-01
2011-05
2011-09
2012-01
2012-05
2012-09
2013-01
Monthly


Quaterly Wise Sales
1e8
7
6
5
4
3
2
1
Weekly_Sales
2010-01
2010-05
2010-09
2011-01
2011-05
2011-09
2012-01
2012-05
2012-09
2013-01
Quaterly


Semester Wise Sales
1e9
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Weekly_Sales
2010-01
2010-05
2010-09
2011-01
2011-05
2011-09
2012-01
2012-05
2012-09
2013-01
Semester

[ ]: