Retail Analysis with Walmart Data_Darshana_N

May 20, 2020

[7]:
```
import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
import sklearn
import seaborn as sns
sns.set_style("whitegrid")
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import mean_squared_error, mean_absolute_error

#For date time functions
from datetime import datetime
from datetime import timedelta
import math

# Importing the most popular regression libraries.
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression, LogisticRegression,␣
↪ridge_regression, Lasso, SGDRegressor, Ridge
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, ExtraTreesRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
```

[8]:
```
#Loading the data from csv files.
train=pd.read_csv('train.csv')
features=pd.read_csv('features.csv')
stores = pd.read_csv('stores.csv')
```

[11]:
```
train.head()
```

[11]:

	Store	Dept	Date	Weekly_Sales	IsHoliday
0	1	1	2010-02-05	24924.50	False
1	1	1	2010-02-12	46039.49	True
2	1	1	2010-02-19	41595.55	False
3	1	1	2010-02-26	19403.54	False
4	1	1	2010-03-05	21827.90	False

[12]:
```
features.head()
```

[12]:

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	\
0	1	2010-02-05	42.31	2.572	NaN	NaN	
1	1	2010-02-12	38.51	2.548	NaN	NaN	
2	1	2010-02-19	39.93	2.514	NaN	NaN	
3	1	2010-02-26	46.63	2.561	NaN	NaN	
4	1	2010-03-05	46.50	2.625	NaN	NaN	

	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsHoliday
0	NaN	NaN	NaN	211.096358	8.106	False
1	NaN	NaN	NaN	211.242170	8.106	True
2	NaN	NaN	NaN	211.289143	8.106	False
3	NaN	NaN	NaN	211.319643	8.106	False
4	NaN	NaN	NaN	211.350143	8.106	False

[14]:
```
stores.head()
```

[14]:

	Store	Type	Size
0	1	A	151315
1	2	A	202307
2	3	B	37392
3	4	A	205863
4	5	B	34875

[15]:
```
#Check for null values in train
train.isnull().any()
```

[15]:
```
Store           False
Dept            False
Date            False
Weekly_Sales    False
IsHoliday       False
dtype: bool
```

[16]:
```
#Check for null values in train
features.isnull().any()
```

[16]:
```
Store           False
Date            False
```

```
Temperature     False
Fuel_Price      False
MarkDown1        True
MarkDown2        True
MarkDown3        True
MarkDown4        True
MarkDown5        True
CPI              True
Unemployment     True
IsHoliday       False
dtype: bool
```

[17]:
```
#Check for null values in train
stores.isnull().any()
```

[17]:
```
Store    False
Type     False
Size     False
dtype: bool
```

There are null values present in below columns.

MarkDown1, MarkDown2, MarkDown3, MarkDown4, MarkDown5, CPI and Unemployment

[18]:
```
#Count of null values present in respective columns of features.
features.isnull().sum()
```

[18]:
```
Store             0
Date              0
Temperature       0
Fuel_Price        0
MarkDown1      4158
MarkDown2      5269
MarkDown3      4577
MarkDown4      4726
MarkDown5      4140
CPI             585
Unemployment    585
IsHoliday         0
dtype: int64
```

[19]:
```
#Original shape of train, features and stores data.
print('train: ', train.shape)
print('features: ', features.shape)
print('stores ', stores.shape)
```

```
train:  (421570, 5)
features:  (8190, 12)
```

```
stores  (45, 3)
```

[20]:
```
#Merging the three csv files using inner join.
data = train.merge(features, on=['Store','Date'],how='inner').
 ↪merge(stores,on=['Store'],how='inner')
data.shape
```

[20]:
```
(421570, 17)
```

[21]:
```
data.head()
```

[21]:

	Store	Dept	Date	Weekly_Sales	IsHoliday_x	Temperature	\
0	1	1	2010-02-05	24924.50	False	42.31	
1	1	2	2010-02-05	50605.27	False	42.31	
2	1	3	2010-02-05	13740.12	False	42.31	
3	1	4	2010-02-05	39954.04	False	42.31	
4	1	5	2010-02-05	32229.38	False	42.31	

	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	\
0	2.572	NaN	NaN	NaN	NaN	NaN	
1	2.572	NaN	NaN	NaN	NaN	NaN	
2	2.572	NaN	NaN	NaN	NaN	NaN	
3	2.572	NaN	NaN	NaN	NaN	NaN	
4	2.572	NaN	NaN	NaN	NaN	NaN	

	CPI	Unemployment	IsHoliday_y	Type	Size
0	211.096358	8.106	False	A	151315
1	211.096358	8.106	False	A	151315
2	211.096358	8.106	False	A	151315
3	211.096358	8.106	False	A	151315
4	211.096358	8.106	False	A	151315

[22]:
```
#Removing additional IsHoliday column (IsHoliday_y) and renaming original␣
 ↪IsHoliday_x column to IsHoliday.
data=data.drop(['IsHoliday_y'],axis=1)
data=data.rename(columns={'IsHoliday_x':'IsHoliday'})
data.columns
```

[22]:
```
Index(['Store', 'Dept', 'Date', 'Weekly_Sales', 'IsHoliday', 'Temperature',
       'Fuel_Price', 'MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4',
       'MarkDown5', 'CPI', 'Unemployment', 'Type', 'Size'],
      dtype='object')
```

[23]:
```
data.head()
```

[23]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	\
0	1	1	2010-02-05	24924.50	False	42.31	2.572	

```
1      1     2  2010-02-05      50605.27      False        42.31       2.572
2      1     3  2010-02-05      13740.12      False        42.31       2.572
3      1     4  2010-02-05      39954.04      False        42.31       2.572
4      1     5  2010-02-05      32229.38      False        42.31       2.572

   MarkDown1  MarkDown2  MarkDown3  MarkDown4  MarkDown5         CPI  \
0        NaN        NaN        NaN        NaN        NaN  211.096358
1        NaN        NaN        NaN        NaN        NaN  211.096358
2        NaN        NaN        NaN        NaN        NaN  211.096358
3        NaN        NaN        NaN        NaN        NaN  211.096358
4        NaN        NaN        NaN        NaN        NaN  211.096358

   Unemployment Type    Size
0         8.106    A  151315
1         8.106    A  151315
2         8.106    A  151315
3         8.106    A  151315
4         8.106    A  151315
```

[24]:
```
data.shape
```

[24]:
```
(421570, 16)
```

Data Preprocessing:

NaN for markdown means that there was no markdown event for that date. So we can replace that with 0 indicating no mark down

[25]:
```
# First we check what happens when we replace NaN's with 0.
data.fillna(0).head()
```

[25]:
```
   Store  Dept        Date  Weekly_Sales  IsHoliday  Temperature  Fuel_Price  \
0      1     1  2010-02-05      24924.50      False        42.31       2.572
1      1     2  2010-02-05      50605.27      False        42.31       2.572
2      1     3  2010-02-05      13740.12      False        42.31       2.572
3      1     4  2010-02-05      39954.04      False        42.31       2.572
4      1     5  2010-02-05      32229.38      False        42.31       2.572

   MarkDown1  MarkDown2  MarkDown3  MarkDown4  MarkDown5         CPI  \
0        0.0        0.0        0.0        0.0        0.0  211.096358
1        0.0        0.0        0.0        0.0        0.0  211.096358
2        0.0        0.0        0.0        0.0        0.0  211.096358
3        0.0        0.0        0.0        0.0        0.0  211.096358
4        0.0        0.0        0.0        0.0        0.0  211.096358

   Unemployment Type    Size
0         8.106    A  151315
1         8.106    A  151315
```

```
2         8.106    A  151315
3         8.106    A  151315
4         8.106    A  151315
```

[26]:
```
##There are null values present in the dataset. Let's remove those
data.isnull().head()
```

[26]:
```
   Store   Dept   Date  Weekly_Sales  IsHoliday  Temperature  Fuel_Price  \
0  False  False  False         False      False        False       False
1  False  False  False         False      False        False       False
2  False  False  False         False      False        False       False
3  False  False  False         False      False        False       False
4  False  False  False         False      False        False       False

   MarkDown1  MarkDown2  MarkDown3  MarkDown4  MarkDown5    CPI  Unemployment  \
0       True       True       True       True       True  False         False
1       True       True       True       True       True  False         False
2       True       True       True       True       True  False         False
3       True       True       True       True       True  False         False
4       True       True       True       True       True  False         False

    Type   Size
0  False  False
1  False  False
2  False  False
3  False  False
4  False  False
```

[27]:
```
# Removing rows with null values in all columns
data.dropna(axis=0,how="all",inplace=True)
# Removing rows with null values in all rows
data.dropna(axis=1,how='all',inplace=True)
```

[28]:
```
# Fill missing values with 0
data=data.fillna(0)
```

[29]:
```
data.isna().sum()
```

[29]:
```
Store           0
Dept            0
Date            0
Weekly_Sales    0
IsHoliday       0
Temperature     0
Fuel_Price      0
MarkDown1       0
MarkDown2       0
```

```
MarkDown3       0
MarkDown4       0
MarkDown5       0
CPI             0
Unemployment    0
Type            0
Size            0
dtype: int64
```

[30]:
```
data.describe()
```

[30]:
```
                Store           Dept   Weekly_Sales    Temperature  \
count  421570.000000  421570.000000  421570.000000  421570.000000
mean       22.200546      44.260317   15981.258123      60.090059
std        12.785297      30.492054   22711.183519      18.447931
min         1.000000       1.000000   -4988.940000      -2.060000
25%        11.000000      18.000000    2079.650000      46.680000
50%        22.000000      37.000000    7612.030000      62.090000
75%        33.000000      74.000000   20205.852500      74.280000
max        45.000000      99.000000  693099.360000     100.140000

          Fuel_Price      MarkDown1      MarkDown2      MarkDown3  \
count  421570.000000  421570.000000  421570.000000  421570.000000
mean        3.361027    2590.074819     879.974298     468.087665
std         0.458515    6052.385934    5084.538801    5528.873453
min         2.472000       0.000000    -265.760000     -29.100000
25%         2.933000       0.000000       0.000000       0.000000
50%         3.452000       0.000000       0.000000       0.000000
75%         3.738000    2809.050000       2.200000       4.540000
max         4.468000   88646.760000  104519.540000  141630.610000

           MarkDown4      MarkDown5            CPI   Unemployment  \
count  421570.000000  421570.000000  421570.000000  421570.000000
mean     1083.132268    1662.772385     171.201947       7.960289
std      3894.529945    4207.629321      39.159276       1.863296
min         0.000000       0.000000     126.064000       3.879000
25%         0.000000       0.000000     132.022667       6.891000
50%         0.000000       0.000000     182.318780       7.866000
75%       425.290000    2168.040000     212.416993       8.572000
max     67474.850000  108519.280000     227.232807      14.313000

                Size
count  421570.000000
mean   136727.915739
std     60980.583328
min     34875.000000
25%     93638.000000
```

```
50%    140167.000000
75%    202505.000000
max    219622.000000
```

[31]:
```
print("Final Data shape",data.shape)
```

```
Final Data shape (421570, 16)
```

[32]:
```
# Taking Mean of Temparature, Weekly Sales and Unemployment columns and forming␣
 ↪additional columns.
temp_mean = data.Temperature.fillna(data['Temperature'].mean())
unemployment_mean = data.Unemployment.fillna(data['Unemployment'].mean())
weekly_sales_mean = data.Weekly_Sales.fillna(data['Weekly_Sales'].mean())

data['Weekly Sales Mean'] = weekly_sales_mean
data['Unemployment Mean'] = unemployment_mean
data['Temperature Mean']  = temp_mean
```

[33]:
```
data.describe()
```

[33]:

	Store	Dept	Weekly_Sales	Temperature
count	421570.000000	421570.000000	421570.000000	421570.000000
mean	22.200546	44.260317	15981.258123	60.090059
std	12.785297	30.492054	22711.183519	18.447931
min	1.000000	1.000000	-4988.940000	-2.060000
25%	11.000000	18.000000	2079.650000	46.680000
50%	22.000000	37.000000	7612.030000	62.090000
75%	33.000000	74.000000	20205.852500	74.280000
max	45.000000	99.000000	693099.360000	100.140000

	Fuel_Price	MarkDown1	MarkDown2	MarkDown3
count	421570.000000	421570.000000	421570.000000	421570.000000
mean	3.361027	2590.074819	879.974298	468.087665
std	0.458515	6052.385934	5084.538801	5528.873453
min	2.472000	0.000000	-265.760000	-29.100000
25%	2.933000	0.000000	0.000000	0.000000
50%	3.452000	0.000000	0.000000	0.000000
75%	3.738000	2809.050000	2.200000	4.540000
max	4.468000	88646.760000	104519.540000	141630.610000

	MarkDown4	MarkDown5	CPI	Unemployment
count	421570.000000	421570.000000	421570.000000	421570.000000
mean	1083.132268	1662.772385	171.201947	7.960289
std	3894.529945	4207.629321	39.159276	1.863296
min	0.000000	0.000000	126.064000	3.879000
25%	0.000000	0.000000	132.022667	6.891000
50%	0.000000	0.000000	182.318780	7.866000

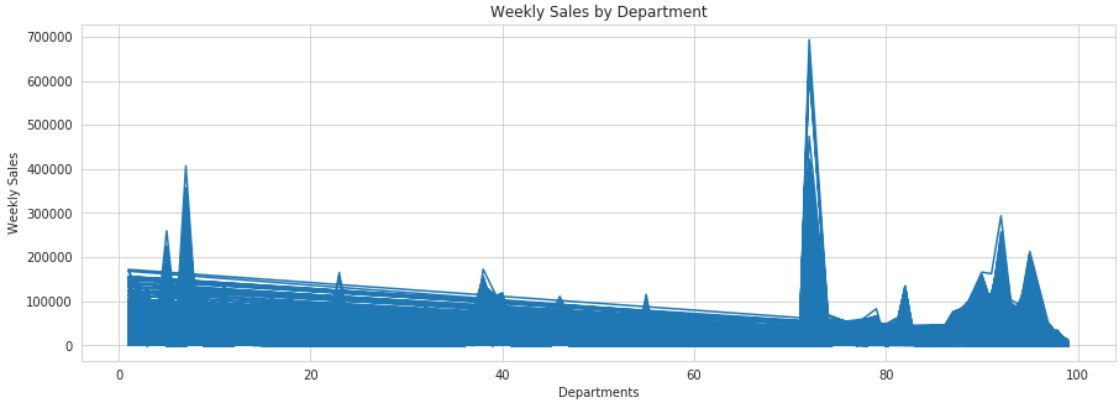
```
75%       425.290000    2168.040000      212.416993       8.572000
max     67474.850000  108519.280000      227.232807      14.313000

                Size  Weekly Sales Mean  Unemployment Mean  Temperature Mean
count  421570.000000      421570.000000      421570.000000     421570.000000
mean   136727.915739       15981.258123           7.960289         60.090059
std     60980.583328       22711.183519           1.863296         18.447931
min     34875.000000       -4988.940000           3.879000         -2.060000
25%     93638.000000        2079.650000           6.891000         46.680000
50%    140167.000000        7612.030000           7.866000         62.090000
75%    202505.000000       20205.852500           8.572000         74.280000
max    219622.000000      693099.360000          14.313000        100.140000
```

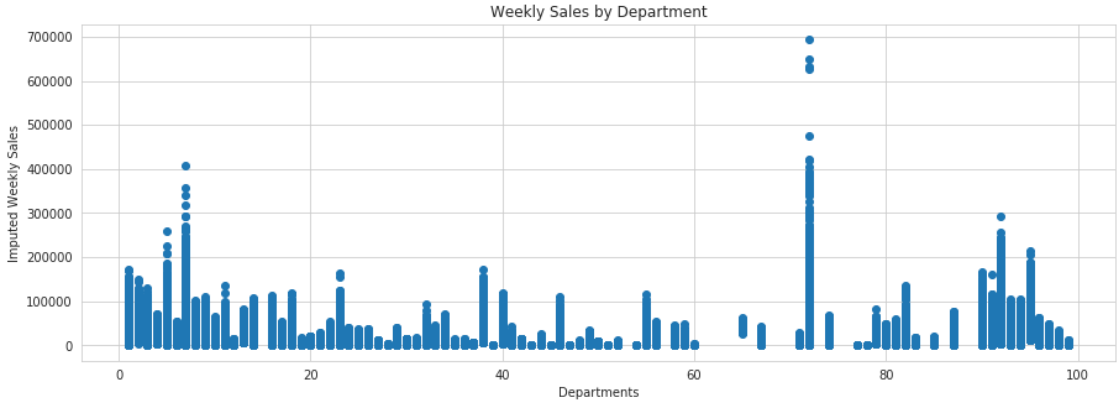
[34]:
```
# Remove negative values as sales cannot be negative values.
data= data[data['Weekly_Sales'] >= 0]
data.shape
```

[34]:
```
(420285, 19)
```

Exploratory Data Analysis: 1. Weekly Sales by Department:

[35]:
```
#Plot of Weekly Sales and Department.
x = data['Dept']
y = data['Weekly_Sales']
plt.figure(figsize=(15,5))
plt.title('Weekly Sales by Department')
plt.xlabel('Departments')
plt.ylabel('Weekly Sales')
plt.plot(x,y)
plt.show()

#Plot of Mean Weekly Sales and Department.
x = data['Dept']
y = data['Weekly Sales Mean']
plt.figure(figsize=(15,5))
plt.title('Weekly Sales by Department')
plt.xlabel('Departments')
plt.ylabel('Imputed Weekly Sales')
plt.scatter(x,y)
plt.show()
```



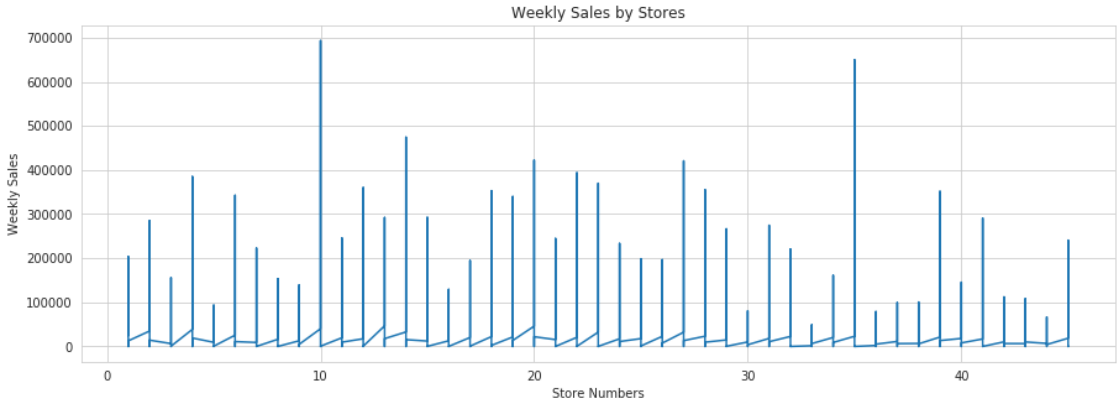


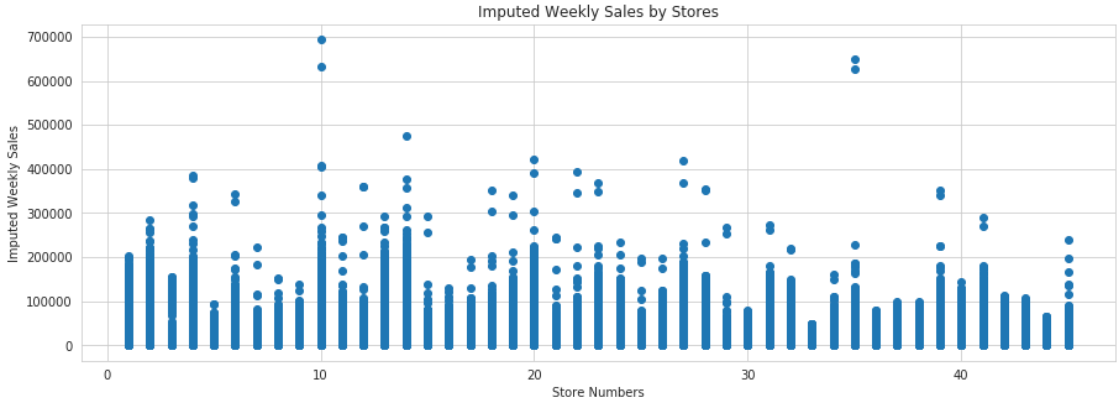
Observation: Most of the Departments have weekly sales below 200000.

2. Weekly Sales by Stores:

[36]:
```
#Plot of Weekly Sales and Stores.
x = data['Store']
y = data['Weekly_Sales']
plt.figure(figsize=(15,5))
plt.title('Weekly Sales by Stores')
plt.xlabel('Store Numbers')
plt.ylabel('Weekly Sales')
plt.plot(x,y)
plt.show()

#Plot of Mean Weekly Sales and Stores.
x = data['Store']
y = data['Weekly Sales Mean']
plt.figure(figsize=(15,5))
plt.title('Imputed Weekly Sales by Stores')
```

```
plt.xlabel('Store Numbers')
plt.ylabel('Imputed Weekly Sales')
plt.scatter(x,y)
plt.show()
```





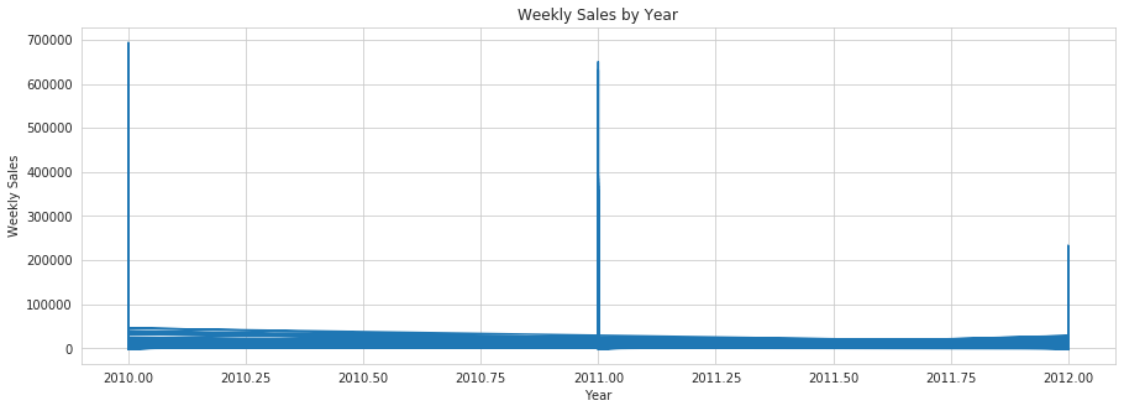
Observation: Weekly sales is highest for Store number 10 and lowest for Store number 44.

3. Weekly Sales by Year, Month and Week:

[39]:
```
# First we need to get Year, Month and Week columns from Date column.
import pandas as pd
data.Date=pd.to_datetime(data.Date)
data['Year']  = data.Date.dt.year
data['Month'] = data.Date.dt.month
data['Week']  = data.Date.dt.week

data.head(2)
```

[39]:
```
   Store  Dept       Date  Weekly_Sales  IsHoliday  Temperature  Fuel_Price  \
0      1     1 2010-02-05      24924.50      False        42.31       2.572
1      1     2 2010-02-05      50605.27      False        42.31       2.572

   MarkDown1  MarkDown2  MarkDown3  …         CPI  Unemployment  Type  \
0        0.0        0.0        0.0  …  211.096358         8.106     A
1        0.0        0.0        0.0  …  211.096358         8.106     A

     Size Weekly Sales Mean  Unemployment Mean  Temperature Mean  Year  Month  \
0  151315           24924.50              8.106             42.31  2010      2
1  151315           50605.27              8.106             42.31  2010      2

   Week
0     5
1     5

[2 rows x 22 columns]
```

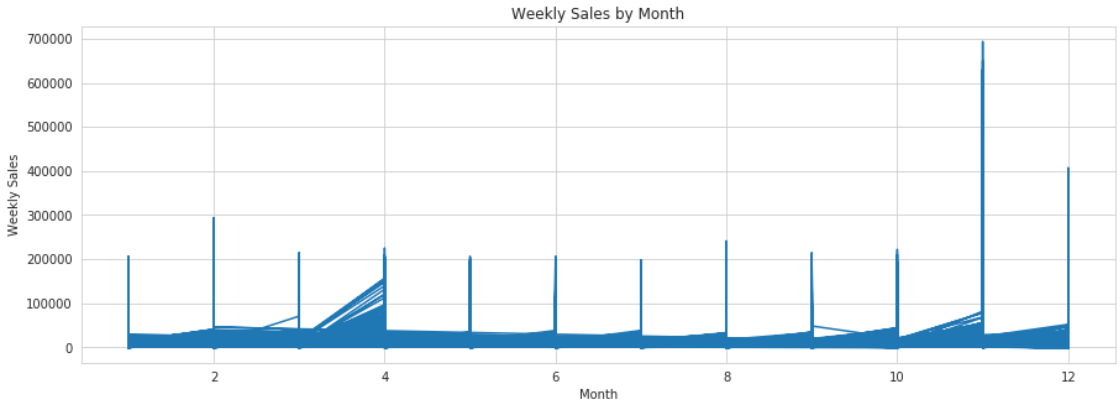
[40]:
```
#Plot of Weekly Sales and Year.
x = data['Year']
y = data['Weekly_Sales']
plt.figure(figsize=(15,5))
plt.title('Weekly Sales by Year')
plt.xlabel('Year')
plt.ylabel('Weekly Sales')
plt.plot(x,y)
plt.show()
```



Observation: Year 2011 had the highest weekly sales.

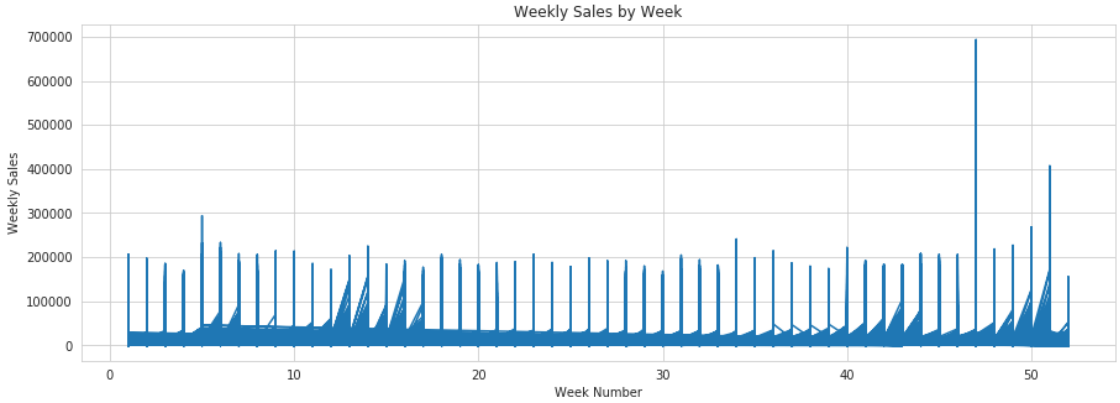
[41]:
```
#Plot of Weekly Sales and Month.
x = data['Month']
y = data['Weekly_Sales']
```

```
plt.figure(figsize=(15,5))
plt.title('Weekly Sales by Month')
plt.xlabel('Month')
plt.ylabel('Weekly Sales')
plt.plot(x,y)
plt.show()
```



Observation: November month witnessed the maximum weekly sales.

[42]:
```
#Plot of Weekly Sales and Week.
x = data['Week']
y = data['Weekly_Sales']
plt.figure(figsize=(15,5))
plt.title('Weekly Sales by Week')
plt.xlabel('Week Number')
plt.ylabel('Weekly Sales')
plt.plot(x,y)
plt.show()
```



Observation: Week 47 of November month had the highest weekly sales.

4. Store Size by Store Type:

Let's make a pie chart to show the ratio of A, B, and C types of total 45 Walmart stores.

First, let's group data by type of stores and see the descriptive figures.

Later we will plot box plot for Store Type and Store Size.

[43]:
```
print("the shape of stores data set is", stores.shape)
print('='*50)
print("the unique value of store is", stores['Store'].unique())
print('='*110)
print("the unique value of Type is", stores['Type'].unique())
```

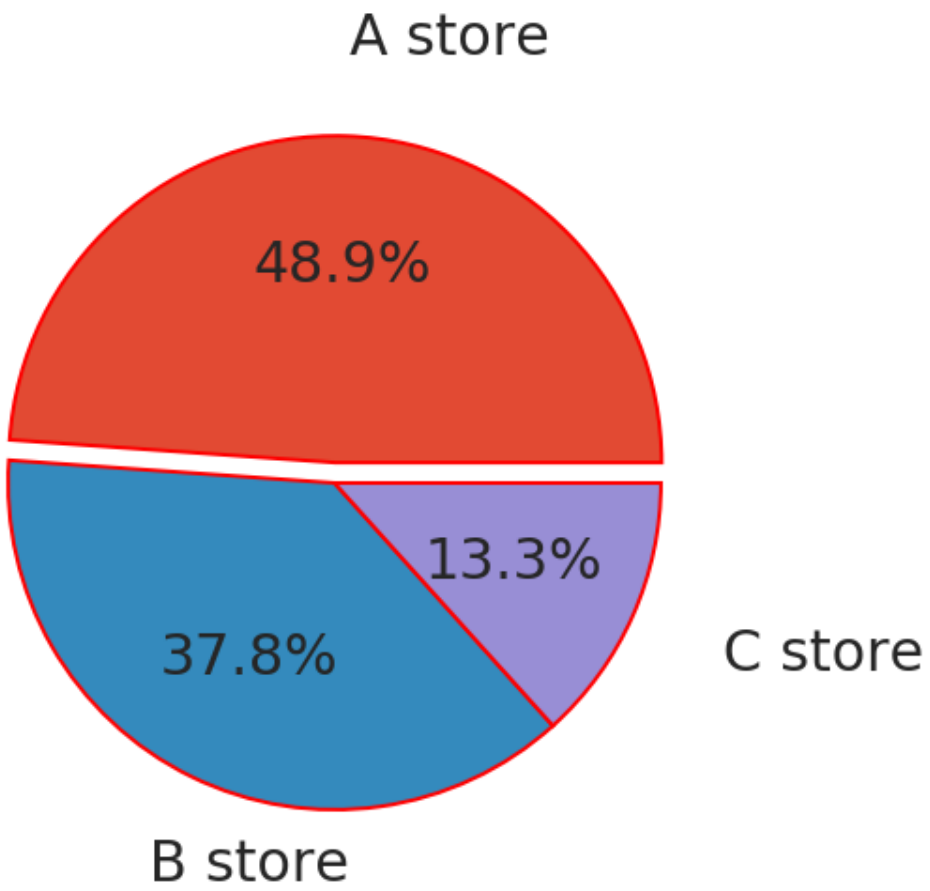
```
the shape of stores data set is (45, 3)
==================================================
the unique value of store is [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
18 19 20 21 22 23 24
 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45]
================================================================================
==============================
the unique value of Type is ['A' 'B' 'C']
```

[44]:
```
sorted_type = stores.groupby('Type')
print(sorted_type.describe()['Size'].round(2))
```

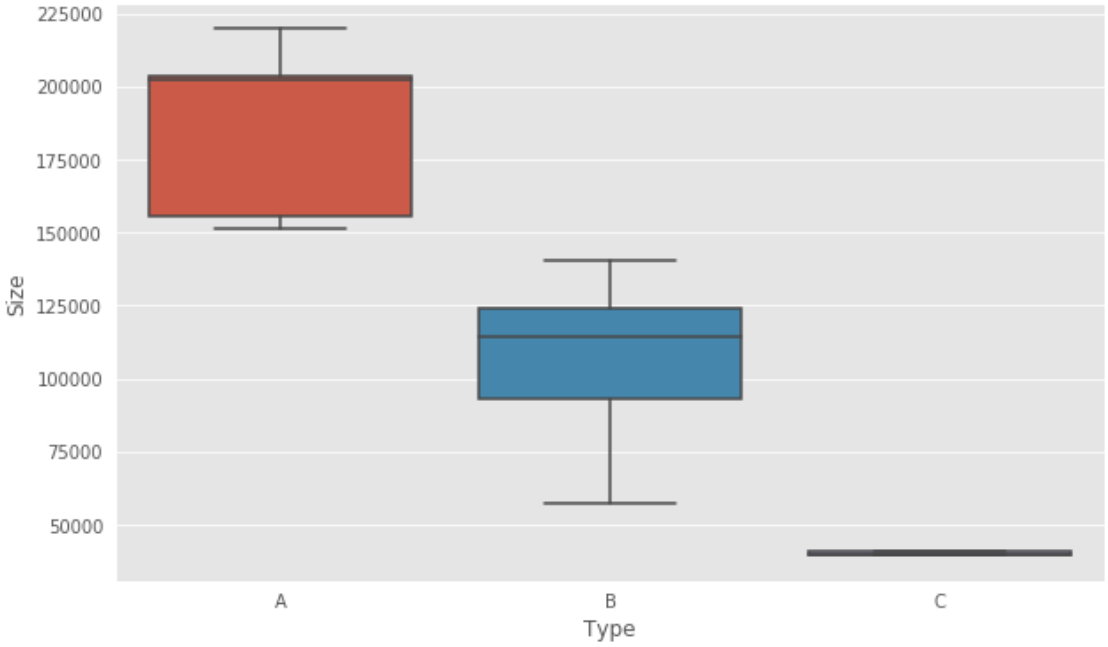
```
      count       mean       std      min        25%       50%       75%  \
Type
A      22.0  177247.73  49392.62  39690.0  155840.75  202406.0  203819.0
B      17.0  101190.71  32371.14  34875.0   93188.00  114533.0  123737.0
C       6.0   40541.67   1304.15  39690.0   39745.00   39910.0   40774.0

           max
Type
A     219622.0
B     140167.0
C      42988.0
```

[50]:
```
#Make Pie chart for Stores including Weekly Sales.
plt.style.use('ggplot')
labels=['A store','B store','C store']
sizes=sorted_type.describe()['Size'].round(1)
sizes=[(22/(17+6+22))*100,(17/(17+6+22))*100,(6/(17+6+22))*100] # convert to␣
 ↪the proportion

fig, axes = plt.subplots(1,1, figsize=(10,10))

wprops={'edgecolor':'Red',
```

```
      'linewidth':2}

tprops = {'fontsize':30}

axes.pie(sizes,
        labels=labels,
        explode=(0.05,0,0),
        autopct='%1.1f%%',
        pctdistance=0.6,
        labeldistance=1.3,
        wedgeprops=wprops,
        textprops=tprops,
        radius=0.8,
        center=(0.5,0.5))
plt.show()
```



[52]:
```
#Box Plot of Store Type and Store Size.
type_size = pd.concat([stores['Type'], stores['Size']], axis=1)
plt.figure(figsize=(10,6))
fig = sns.boxplot(x='Type', y='Size', data=type_size, showfliers=False)
```



Observations:

Type A store is the largest store and C being the smallest. There is considerable separation among the store types, hence store type is best predictor for store size.

5. Train-Stores table analysis:

[54]:
```
train_stores = train.merge(stores, on='Store', how='inner')
train_stores.head()
```

[54]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Type	Size
0	1	1	2010-02-05	24924.50	False	A	151315
1	1	1	2010-02-12	46039.49	True	A	151315
2	1	1	2010-02-19	41595.55	False	A	151315
3	1	1	2010-02-26	19403.54	False	A	151315
4	1	1	2010-03-05	21827.90	False	A	151315

[55]:
```
# Form Date, Year, Month, Week, Day and No. of days.
train_stores['Date'] = pd.to_datetime(train_stores['Date'])
train_stores['Year'] = train_stores['Date'].dt.year
train_stores['Month']= train_stores['Date'].dt.month
train_stores['Week'] = train_stores['Date'].dt.week
```

```
train_stores['Day']   = train_stores['Date'].dt.day
train_stores['No. of days'] = (train_stores['Date'].dt.date -␣
 ↪train_stores['Date'].dt.date.min()).apply(lambda x:x.days)
```

[56]:
```
train_stores.head()
```

[56]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Type	Size	Year	Month	\
0	1	1	2010-02-05	24924.50	False	A	151315	2010	2	
1	1	1	2010-02-12	46039.49	True	A	151315	2010	2	
2	1	1	2010-02-19	41595.55	False	A	151315	2010	2	
3	1	1	2010-02-26	19403.54	False	A	151315	2010	2	
4	1	1	2010-03-05	21827.90	False	A	151315	2010	3	

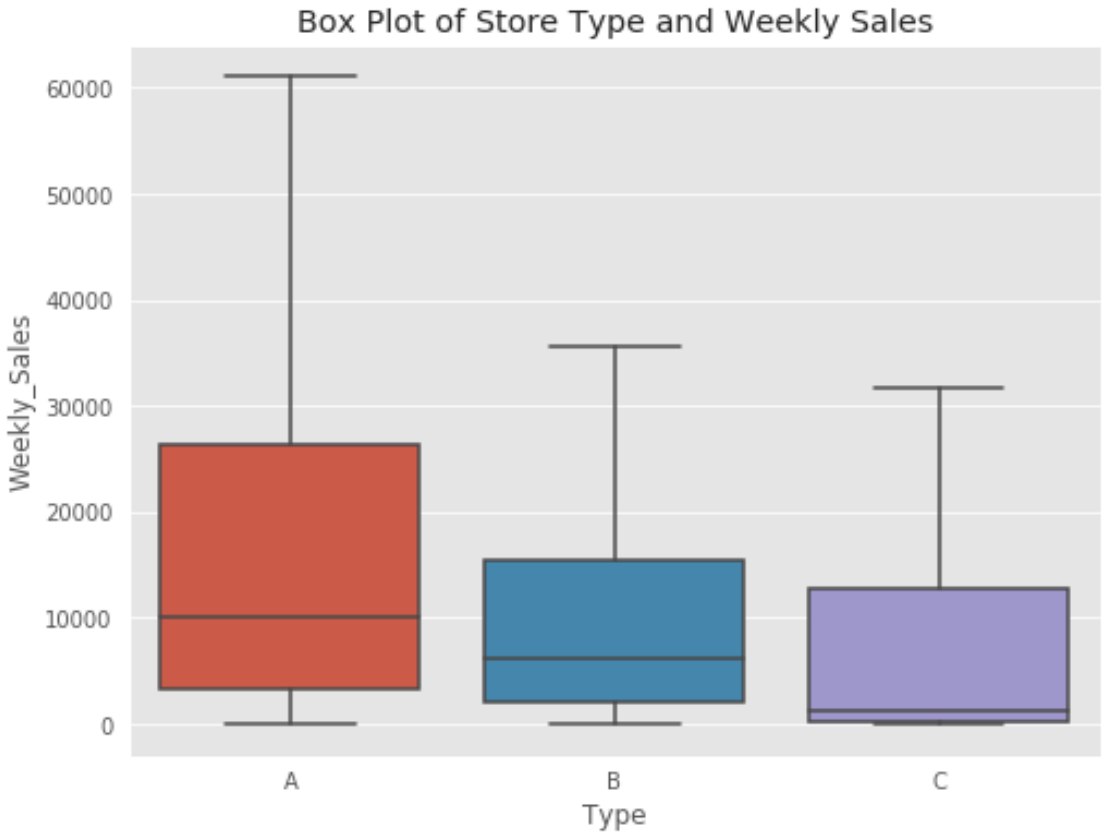
	Week	Day	No. of days
0	5	5	0
1	6	12	7
2	7	19	14
3	8	26	21
4	9	5	28

[57]:
```
Year        = pd.Series(train_stores['Year'].unique())
Week        = pd.Series(train_stores['Week'].unique())
Month       = pd.Series(train_stores['Month'].unique())
Day         = pd.Series(train_stores['Day'].unique())
No_of_days  = pd.Series(train_stores['No. of days'].unique())
```

6. Weekly Sales for Store Type:

[58]:
```
# There are negative values present in Weekly sales which are absurd because␣
 ↪sales cannot be negative.
train_stores= train_stores[train_stores['Weekly_Sales'] > 0]
```

[59]:
```
#Plot of Store Type and Weekly Sales
type_sales = pd.concat([train_stores['Type'], train_stores['Weekly_Sales']],␣
 ↪axis=1)
plt.figure(figsize=(8,6))
plt.title('Box Plot of Store Type and Weekly Sales')
fig = sns.boxplot(x='Type', y='Weekly_Sales', data=type_sales, showfliers=False)
```



Observation: Type A stores have their medians higher than any other medians in other store types, so the weekly sales for store type A is more than other store types.

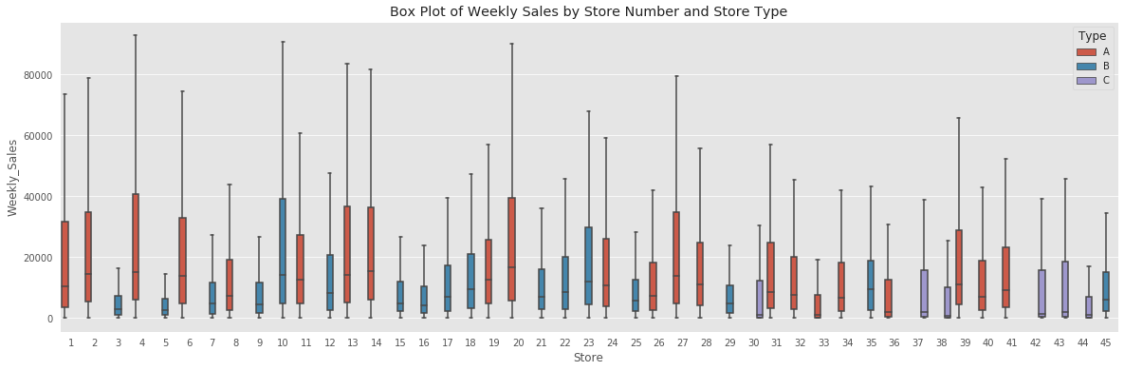
7. Weekly Sales for Store Size:

[60]:
```
#Plot of Store Size and Weekly Sales.
size_sales = pd.concat([train_stores['Size'], train_stores['Weekly_Sales']],␣
 ↪axis=1)
plt.figure(figsize=(20,6))
plt.title('Box Plot of Store Size and Weekly Sales')
fig = sns.boxplot(x='Size', y='Weekly_Sales', data=size_sales, showfliers=False)
```

Observation: There is no clear distinction that can be drawn from this plot.

8. Weekly Sales by Store Number and Store Type

[62]:
```
#Plot of Weekly Sales by Store Number and Store Type.
data_8 = pd.concat([train_stores['Store'], train_stores['Weekly_Sales'],␣
 ↪train_stores['Type']], axis=1)
plt.figure(figsize=(20,6))
plt.title('Box Plot of Weekly Sales by Store Number and Store Type')
fig = sns.boxplot(x='Store', y='Weekly_Sales', data=data_8, showfliers=False,␣
 ↪hue='Type')
```

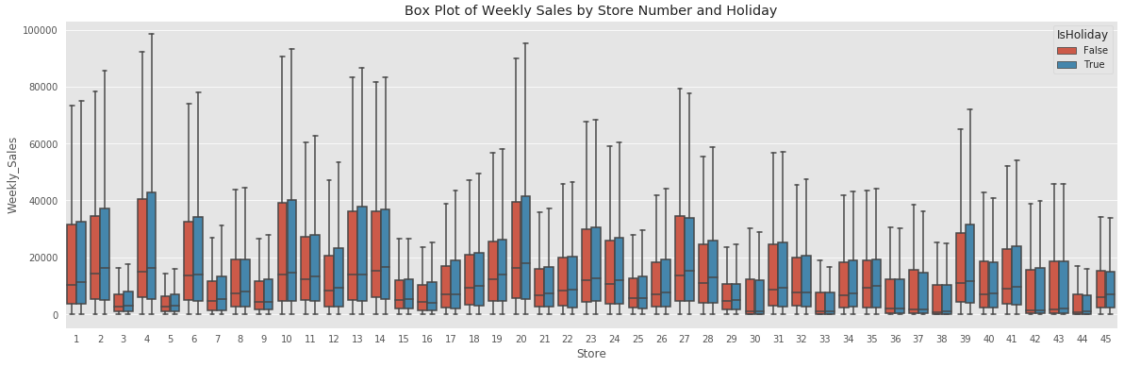


Observation: We can observe that store number 4 followed by 20 have the highest sales, so are their median sales values.

9. Weekly Sales by Store Number and Holiday:

[63]:
```
#Plot of Weekly Sales by Store Number and Holiday.
data_9 = pd.concat([train_stores['Store'], train_stores['Weekly_Sales'],␣
 ↪train_stores['IsHoliday']], axis=1)
plt.figure(figsize=(20,6))
plt.title('Box Plot of Weekly Sales by Store Number and Holiday')
```

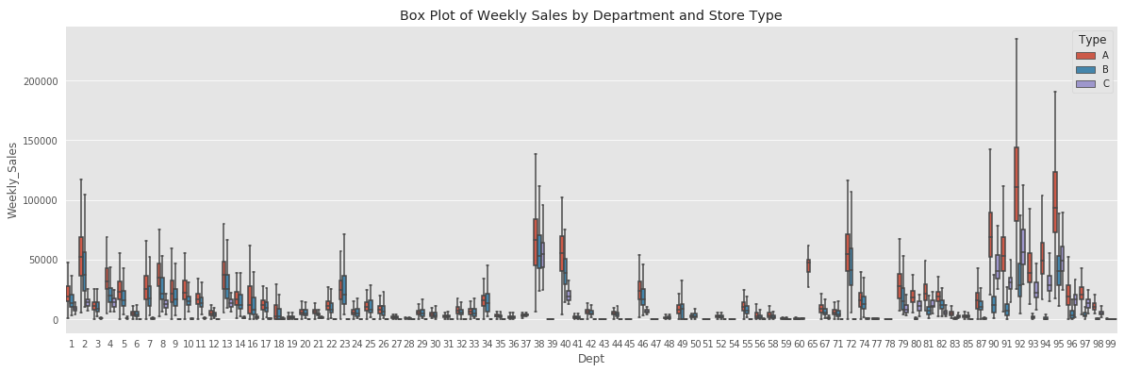
```
fig = sns.boxplot(x='Store', y='Weekly_Sales', data=data_9, showfliers=False,␣
↪hue='IsHoliday')
```



Observation: We can't interpret the reation here, but we can observe that sales are more on holidays.

10. Weekly Sales by Department and Store Type:

[64]:
```
data_10 = pd.concat([train_stores['Dept'], train_stores['Weekly_Sales'],␣
↪train_stores['Type']], axis=1)
plt.figure(figsize=(20,6))
plt.title('Box Plot of Weekly Sales by Department and Store Type')
fig = sns.boxplot(x='Dept', y='Weekly_Sales', data=data_10, showfliers=False,␣
↪hue='Type')
```

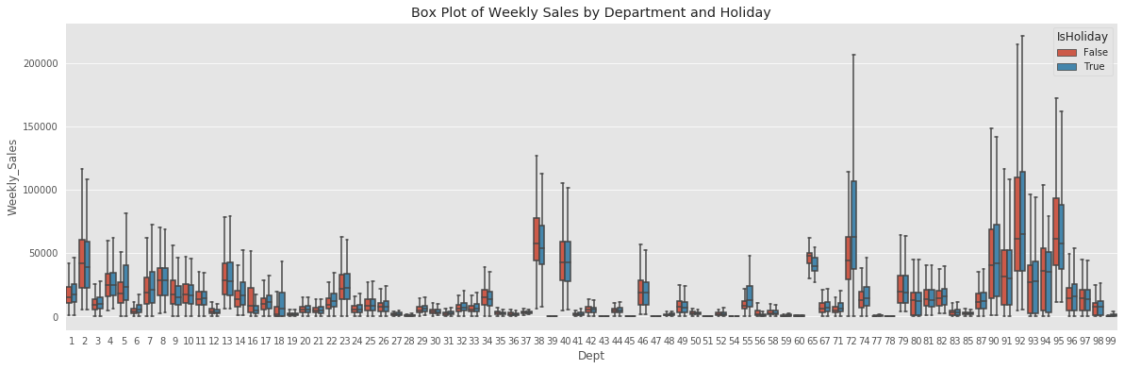


Observation: Here we can infer that weekly sales for store type A is more than any other store types.

11. Weekly Sales by Department and Holiday:

[65]:
```
data_11= pd.concat([train_stores['Dept'], train_stores['Weekly_Sales'],␣
↪train_stores['IsHoliday']], axis=1)
```

```
plt.figure(figsize=(20,6))
plt.title('Box Plot of Weekly Sales by Department and Holiday')
fig = sns.boxplot(x='Dept', y='Weekly_Sales', data=data_11, showfliers=False,␣
↪hue="IsHoliday")
```



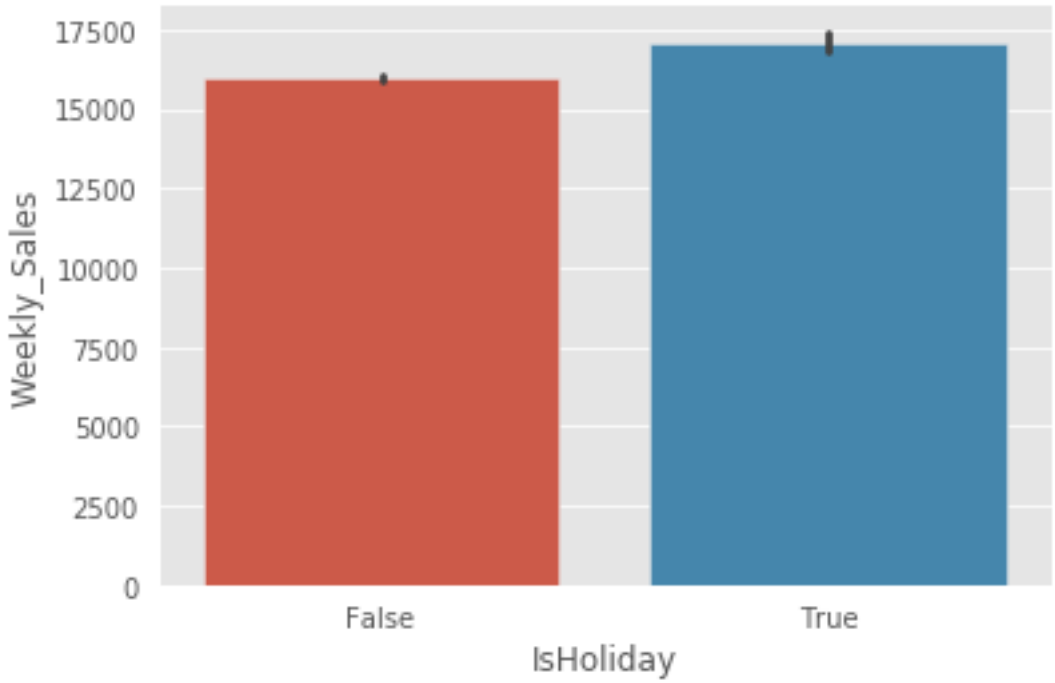
Observations:

Department number 92 has highest sales that too happened on holiday. There is no explicit relation between Department and Weekly Sales.

12. Weekly Sales on Holidays and Non-Holidays:

[66]:
```
holiday = train_stores['Weekly_Sales'].loc[train_stores['IsHoliday']== True] #␣
↪Weekly Sales in Holidays
non_holiday = train_stores['Weekly_Sales'].loc[train_stores['IsHoliday']==␣
↪False] #Weekly Sales in Non-holidays.

sns.barplot(x='IsHoliday', y='Weekly_Sales', data=train_stores)
```

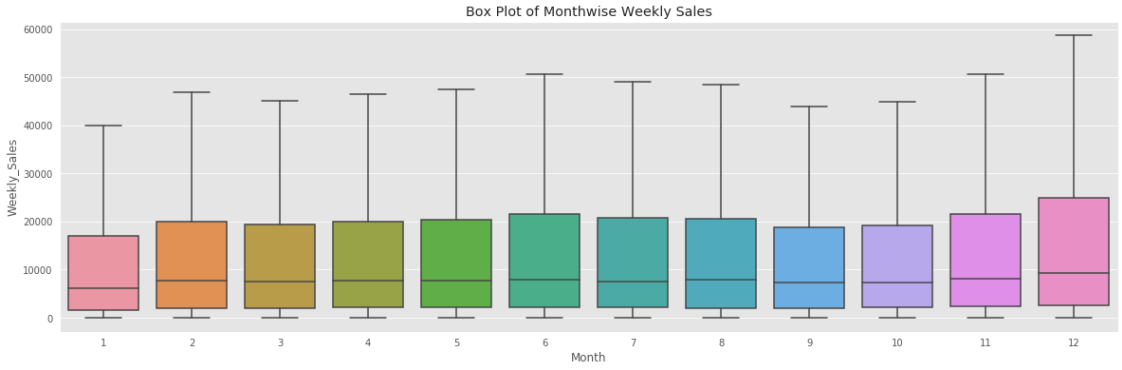
[66]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd9fc4b610>
```



Observation: We see the sales are higher on holidays than on non holidays.

13. Monthwise Weekly Sales:

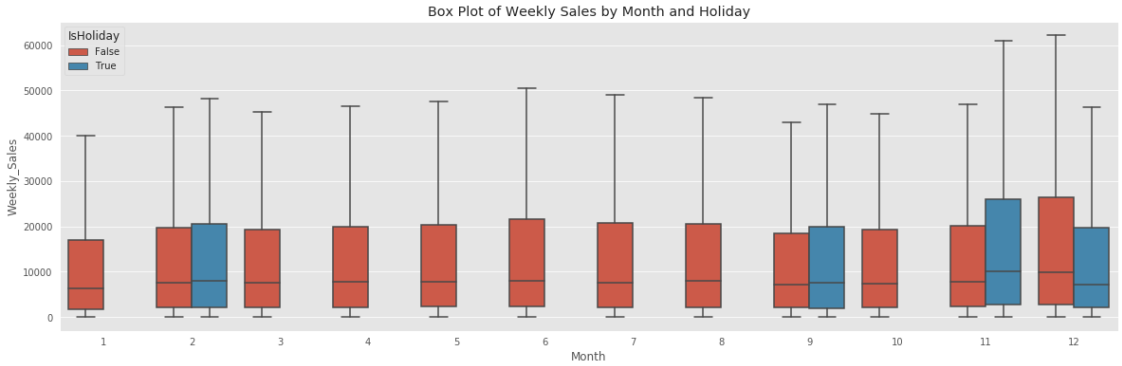
[67]:
```
data_13 = pd.concat([train_stores['Month'], train_stores['Weekly_Sales']],␣
↪axis=1)
plt.figure(figsize=(20,6))
plt.title('Box Plot of Monthwise Weekly Sales')
fig = sns.boxplot(x='Month', y='Weekly_Sales', data=data_13, showfliers=False)
```



Observation: December month had maximum sales.

14. Weekly Sales by Month and Holiday:

[69]:
```
data_14 = pd.concat([train_stores['Month'], train_stores['Weekly_Sales'],␣
 ↪train_stores['IsHoliday']], axis=1)
plt.figure(figsize=(20,6))
plt.title('Box Plot of Weekly Sales by Month and Holiday')
fig = sns.boxplot(x='Month', y='Weekly_Sales', data=data_14, showfliers=False,␣
 ↪hue='IsHoliday')
```



Observation: Non holidays have sales in each of the monhs and in December the sales, which is the highest among all sales, is even more than sales in holidays.

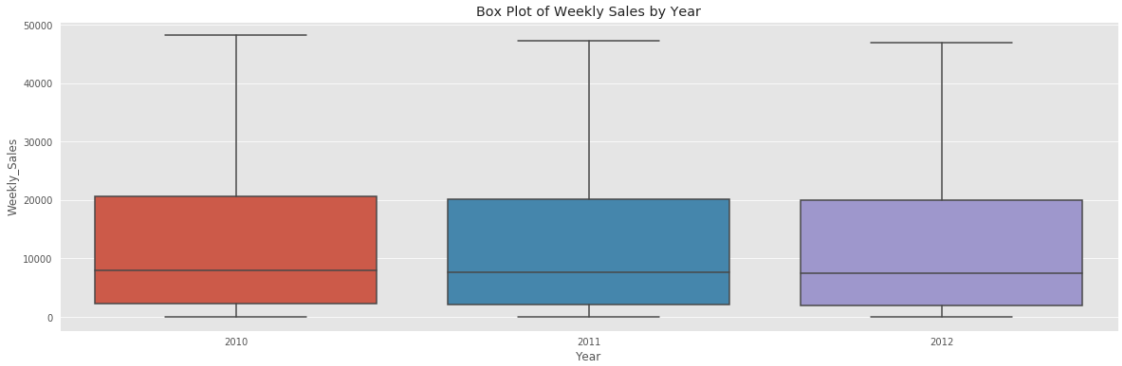
15. Weekly Sales by Month and Store Type:

[70]:
```
data_15 = pd.concat([train_stores['Month'], train_stores['Weekly_Sales'],␣
 ↪train_stores['Type']], axis=1)
plt.figure(figsize=(20,6))
plt.title('Box Plot of Weekly Sales by Month and Store Type')
fig = sns.boxplot(x='Month', y='Weekly_Sales', data=data_15, showfliers=False,␣
 ↪hue='Type')
```

Observation: In every month Store Type A has the maximum sales.

16. Weekly Sales by Year:

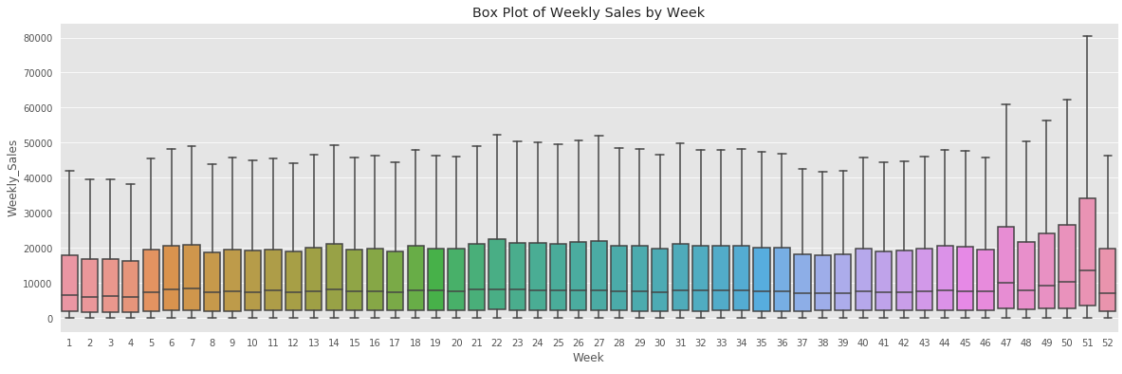
[72]:
```
data_16 = pd.concat([train_stores['Year'], train_stores['Weekly_Sales']],␣
↪axis=1)
plt.figure(figsize=(20,6))
plt.title('Box Plot of Weekly Sales by Year')
fig = sns.boxplot(x='Year', y='Weekly_Sales', data=data_16, showfliers=False)
```



Observation: There seems no clear distinction from this plot.

17. Weekly Sales by Week:

[73]:
```
data_17 = pd.concat([train_stores['Week'], train_stores['Weekly_Sales']],␣
↪axis=1)
plt.figure(figsize=(20,6))
plt.title('Box Plot of Weekly Sales by Week')
fig = sns.boxplot(x='Week', y='Weekly_Sales', data=data_17, showfliers=False)
```



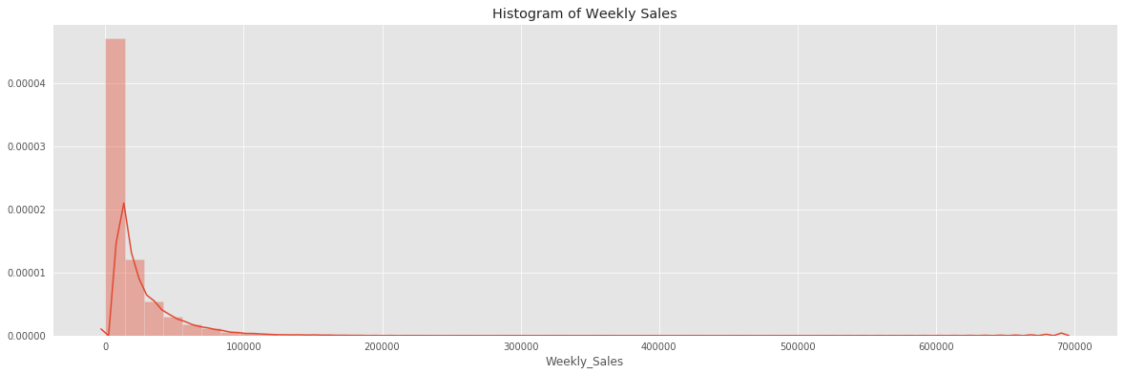
Observation: The 51st week i.e. third week of December has the highest sales.

18. Histogram of Weekly Sales:

[74]:
```
plt.figure(figsize=(20,6))
plt.title('Histogram of Weekly Sales')
```

```
fig = sns.distplot(train_stores['Weekly_Sales'].dropna()) #Taking only valid␣
↪weekly sales values.
print('Minimum sales:', train_stores['Weekly_Sales'].min())
```

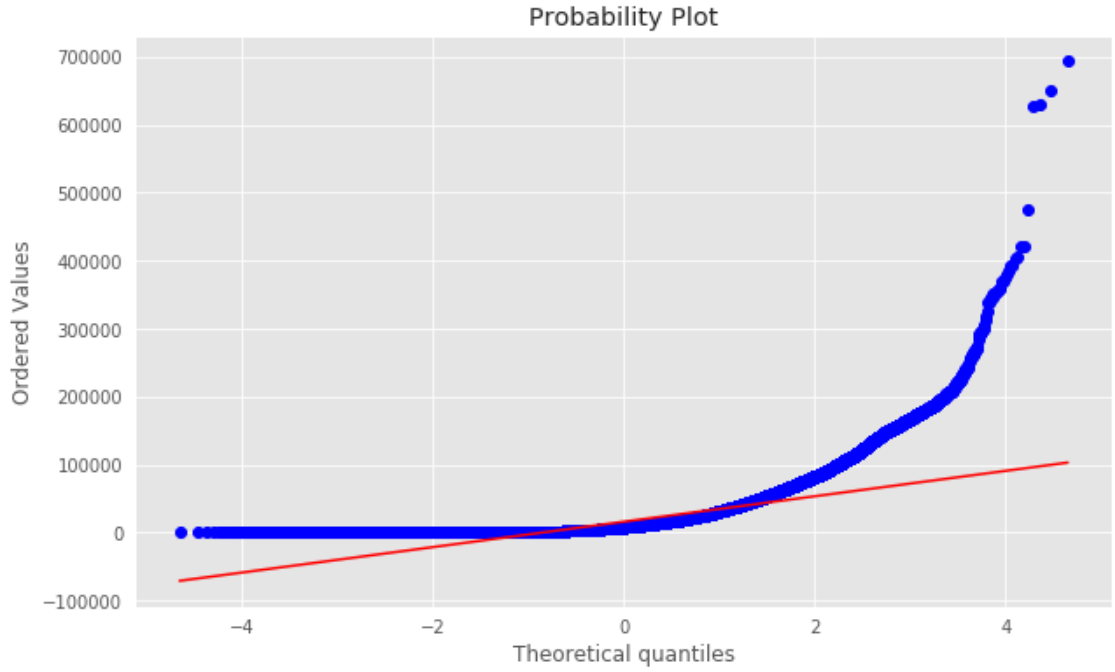
```
Minimum sales: 0.01
```



Observation: About all sales have happened below 100000. Maximum sales are done at sales value of 1000.

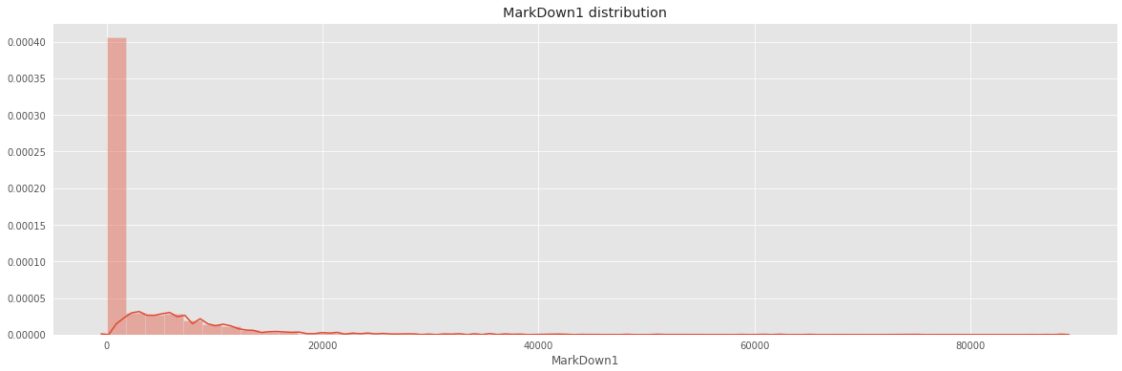
19. Probability Plot of Weekly Sales:

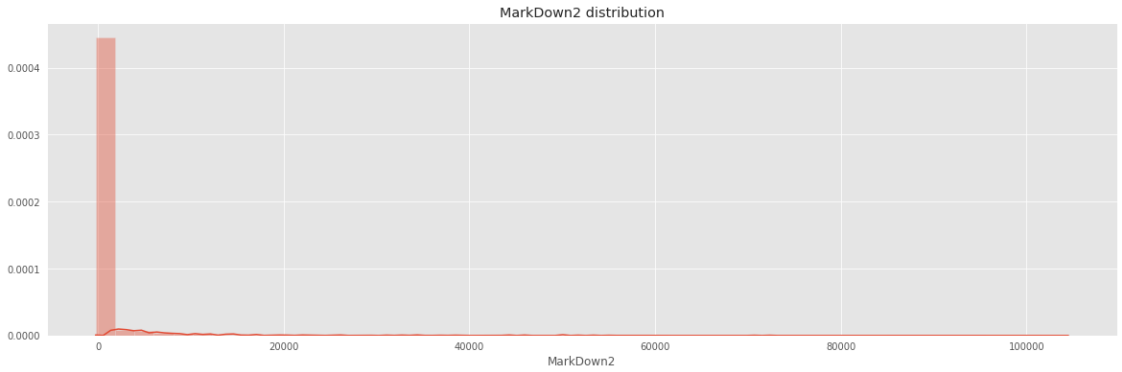
[75]:
```
plt.figure(figsize=(10,6))
fig = stats.probplot(train_stores['Weekly_Sales'], plot=plt)
```

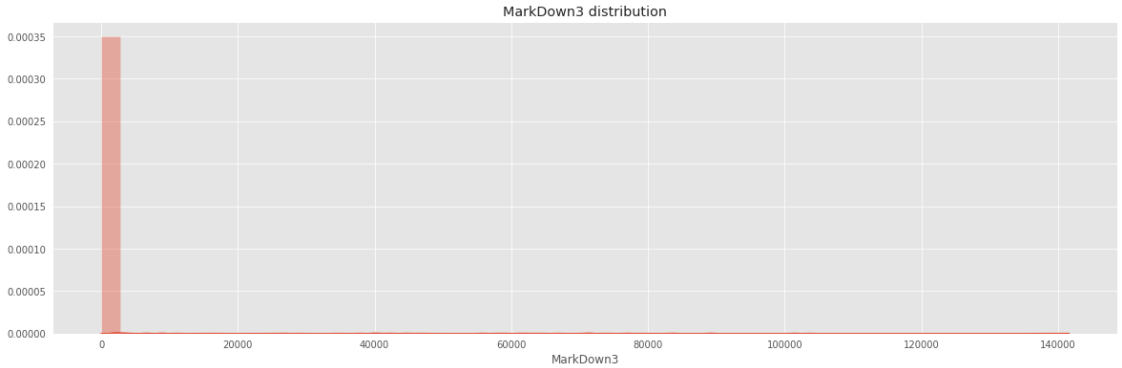


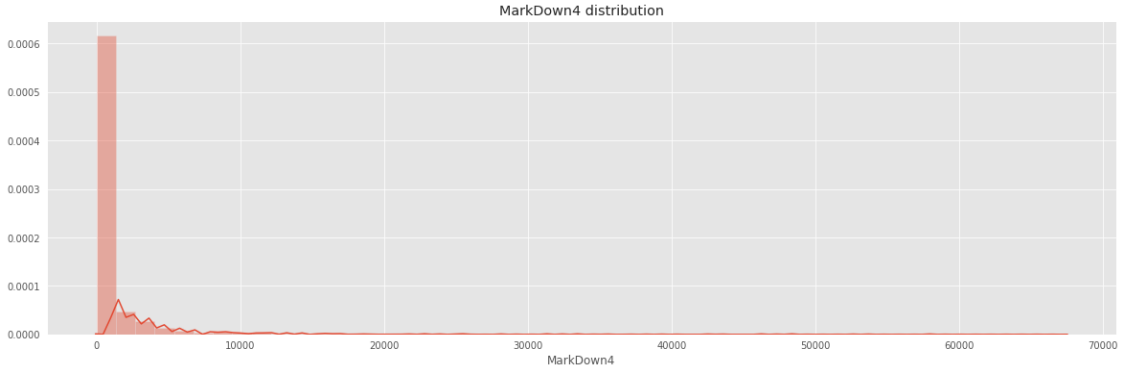
20. Distributions of MarkDown 1, MarkDown 2, MarkDown 3, MarkDown 4, MarkDown 5:

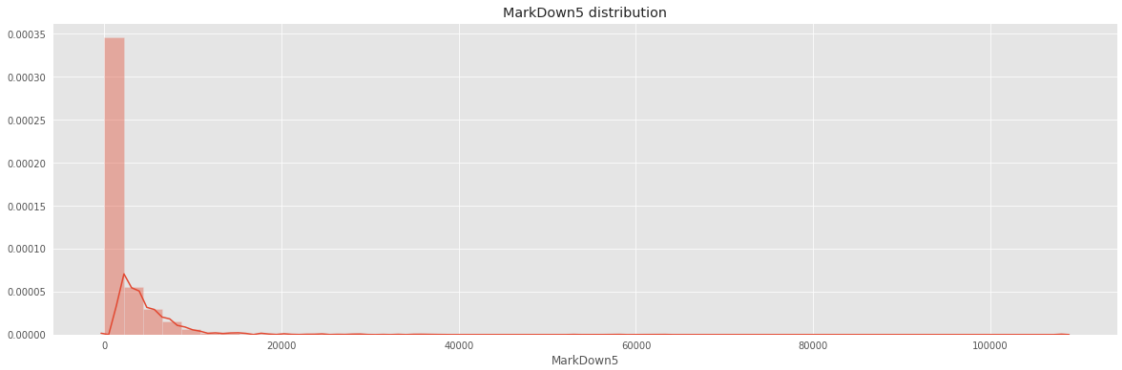
[76]:
```
# Histograms of MarkDowns.
def markdowns(data, column):
  plt.figure(figsize=(20,6))
  sns.distplot(data[column], kde=True)
  plt.title(str(column)+' distribution')
  plt.xlabel(column)

markdowns(data, 'MarkDown1')
markdowns(data, 'MarkDown2')
markdowns(data, 'MarkDown3')
markdowns(data, 'MarkDown4')
markdowns(data, 'MarkDown5')
```





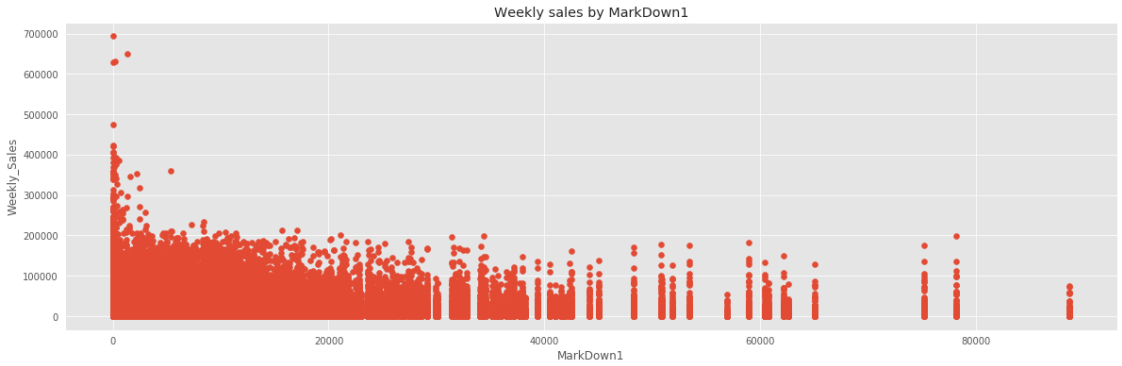


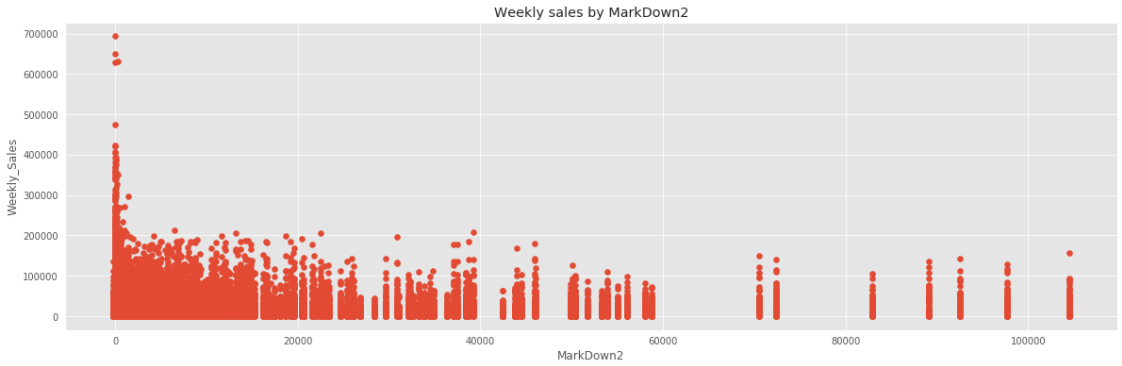


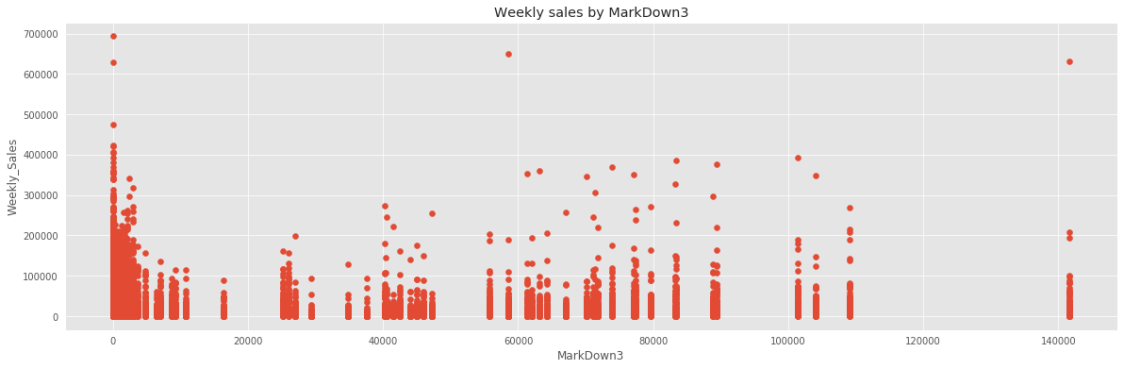


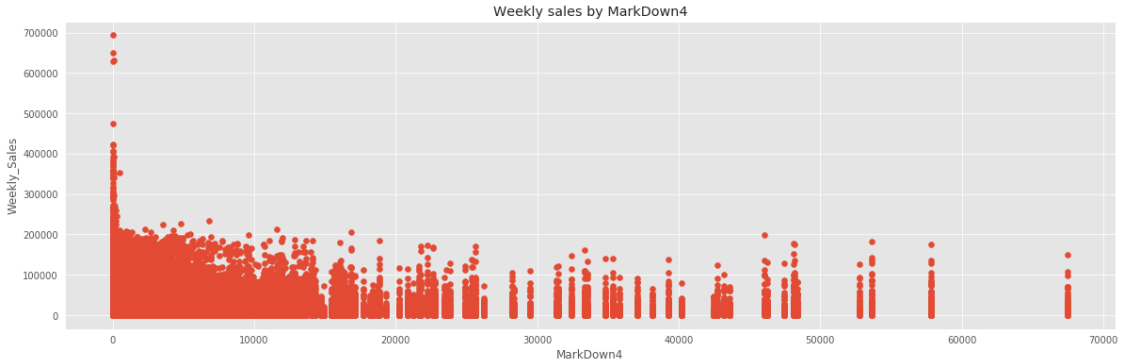
[77]:
```python
# Scatter plots of MarkDowns.
def scatter(data, column):
    plt.figure(figsize=(20,6))
    plt.scatter(data[column] , data['Weekly_Sales'])
    plt.title('Weekly sales by '+str(column))
    plt.ylabel('Weekly_Sales')
    plt.xlabel(column)
```

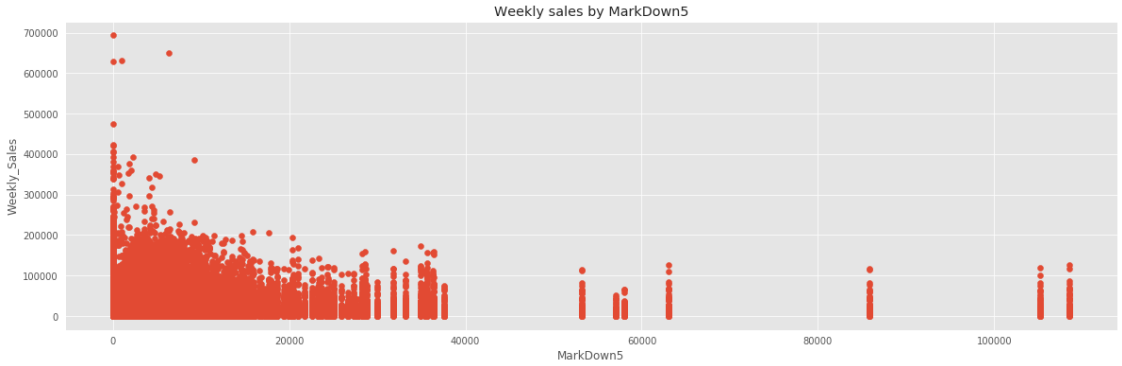
[78]:
```
scatter(data, 'MarkDown1')
scatter(data, 'MarkDown2')
scatter(data, 'MarkDown3')
scatter(data, 'MarkDown4')
scatter(data, 'MarkDown5')
```







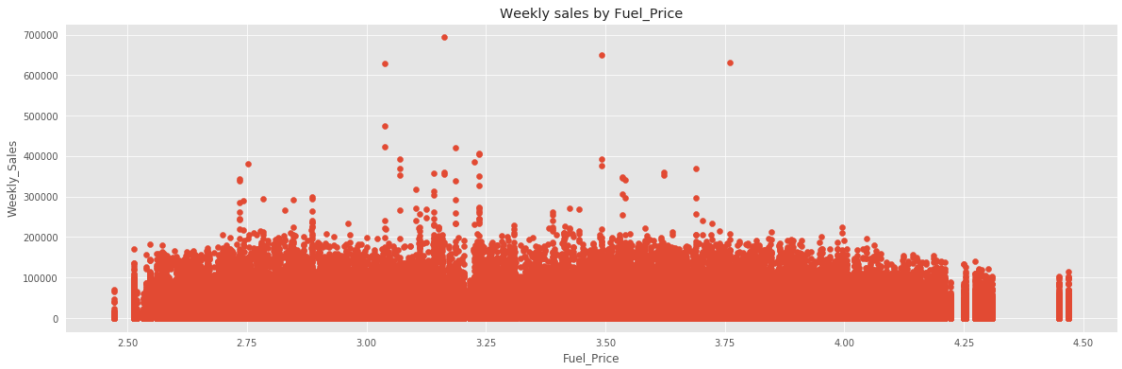


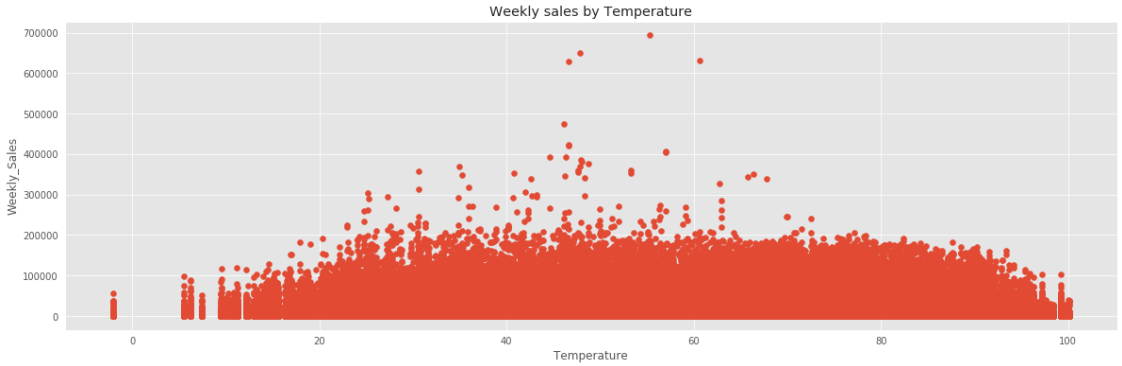


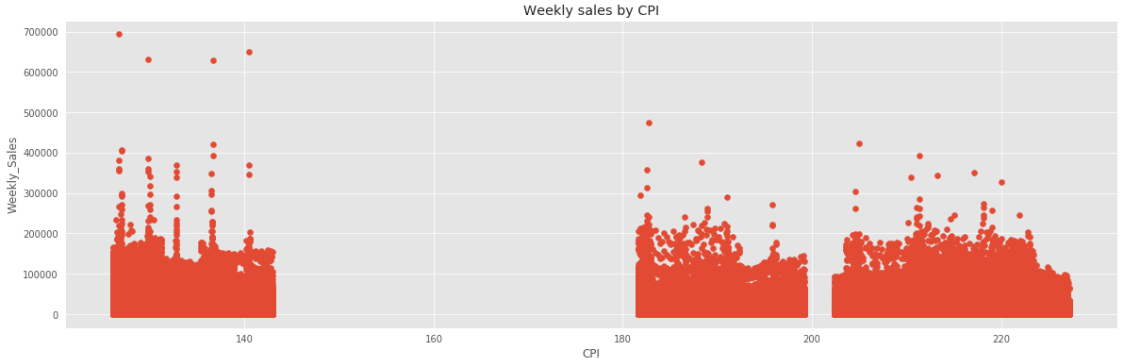
Observation: Most of the Markdowns have been given for Weekly sales of less than 200000

21. Fuel, Temparature and CPI effects:

[79]:
```
scatter(data, 'Fuel_Price') # Fuel
scatter(data, 'Temperature') #Temparature
scatter(data, 'CPI') #CPI
```







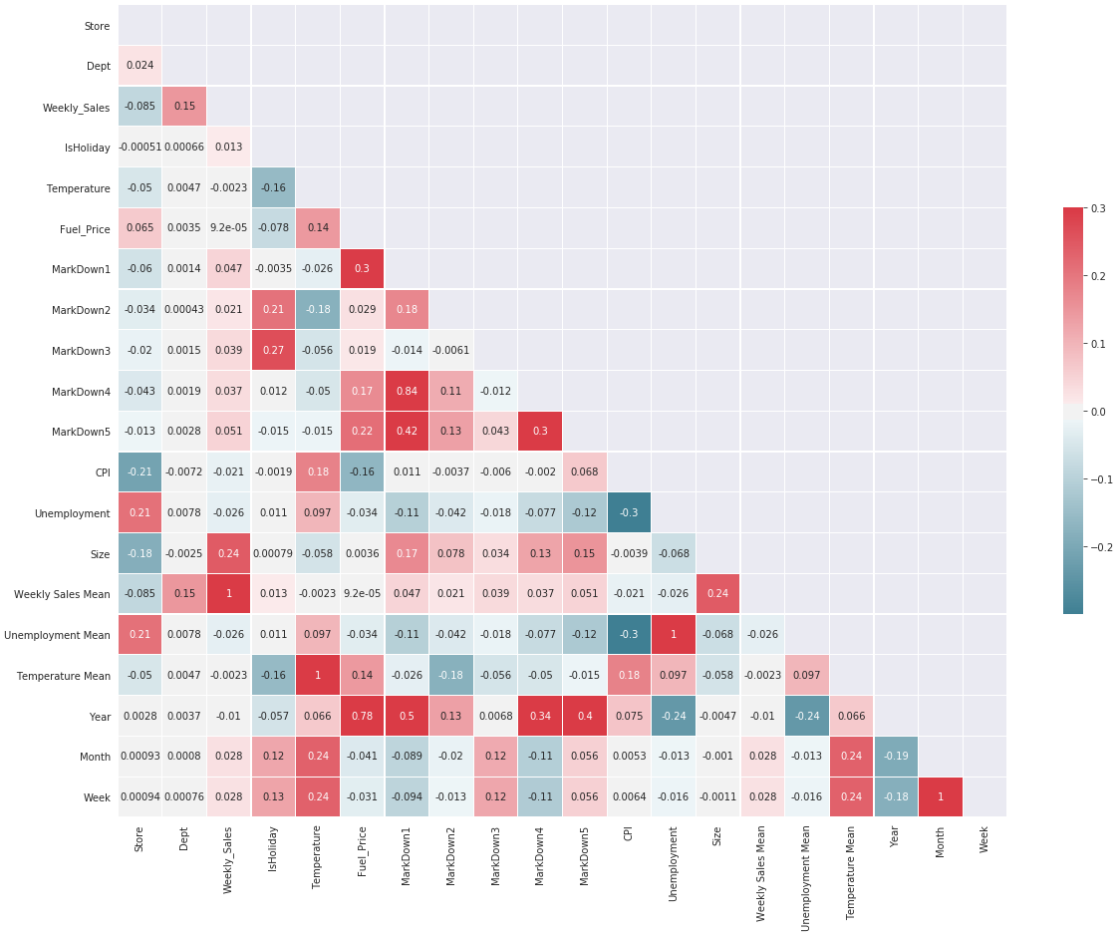
Observations:

Most of the Fuel price value lies in range of 2.5 to 4 for weekly sales of less than 200000.

Sales of less than 200000 happened for average temparature range of 30 to 80.

No weekly sales for CPI range of 145 to 180. CPI range (200 - 230) is more when the weekly sales is less than 200000.

[80]:
```
sns.set_style('dark')
corr = data.corr()

mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)]=True

f,ax = plt.subplots(figsize=(20,15))
cmap = sns.diverging_palette(220, 10, as_cmap=True)

fig = sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0, annot=True,
            linewidths=.2, cbar_kws={"shrink": .5})
```



Logical deductions:

If the Fuel Price in the area is less then more customers would travel to the store for shopping purpose, however that has a very less correlation.

If there is an inflation, Cost of goods will rise which causes CPI to increase. This will cause a reduction in sales which is obvious. But again, CPI doesn't have much correlation.

For a lesser store numbers, the weekly sales are higher. Unemployment, Temperature is less, Sales are higher. (Not much correlation)

[ ]: