PROBLEM STATEMENT :

Rossmann operates over 3,000 drug stores in 7 European countries. Currently, Rossmann store managers are tasked with predicting their daily sales for up to six weeks in advance. Store sales are influenced by many factors, including promotions, competition, school and state holidays, seasonality, and locality. With thousands of individual managers predicting sales based on their unique circumstances, the accuracy of results can be quite varied.

IMPORT LIBRARIES AND DATASET :

```
import numpy as np
import pandas as pd
import datetime
import matplotlib.pyplot as plt
import seaborn as sns

from google.colab import drive
drive.mount('/content/drive')
```

IMPORT SALES TRAINING DATA :

```
sales_train_df = pd.read_csv('/content/drive/MyDrive/DiplomaProject/train.csv')

sales_train_df.shape
```

```
(1017209, 9)
```

We can see that there are a total of 1017209 observations.

There are 9 columns in total and note that sales is the target variable.

--Id: an Id that represents a (Store, Date) duple within the test set

--Store: a unique Id for each store

--Sales: sales/day, this is the target variable

--Customers: number of customers on a given day

--Open: Boolean to say whether a store is open or closed (0 = closed, 1 = open)

--Promo: describes if store is running a promo on that day or not

--StateHoliday: indicate which state holiday (a = public holiday, b = Easter holiday, c = Christmas, 0 = None)

--SchoolHoliday: indicates if the (Store, Date) was affected by the closure of public schools

We can infer the following observations from the describe function results :

--Average sales amount per day = 5773 Euros

--minimum sales per day = 0

--maximum sales per day = 41551

--Average number of customers = 633

--minimum number of customers = 0

--maximum number of customers = 7388

IMPORT STORE DATA :

store_info_df = pd.read_csv('/content/drive/MyDrive/DiplomaProject/store.csv')

--StoreType: categorical variable to indicate type of store (a, b, c, d)

--Assortment: describes an assortment level: a = basic, b = extra, c = extended

--CompetitionDistance (meters): distance to closest competitor store

--CompetitionOpenSince [Month/Year]: provides an estimate of the date when competition was open

--Promo2: Promo2 is a continuing and consecutive promotion for some stores (0 = store is not participating, 1 = store is participating)

--Promo2Since [Year/Week]: date when the store started participating in Promo2

--PromoInterval: describes the consecutive intervals Promo2 is started, naming the months the promotion is started anew. E.g. "Feb,May,Aug,Nov" means each round starts in February, May, August, November of any given year for that store

---

store_info_df.shape

(1115, 10)

This dataframe only includes information about the unique 1115 stores that are part of this study and does not contain information about transaction sales per day.

store_info_df.describe()

---

on average, the competition distance is 5404 meters away (5.4 kms)

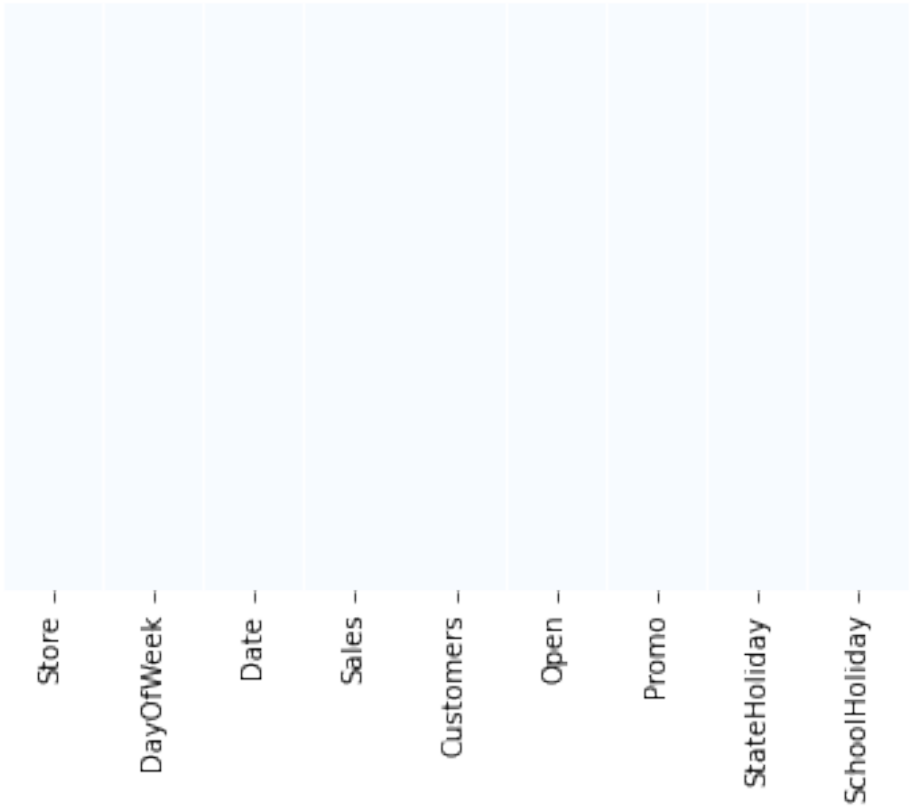
EDA OF DATASETS :

EDA OF SALES TRAINING DATA :

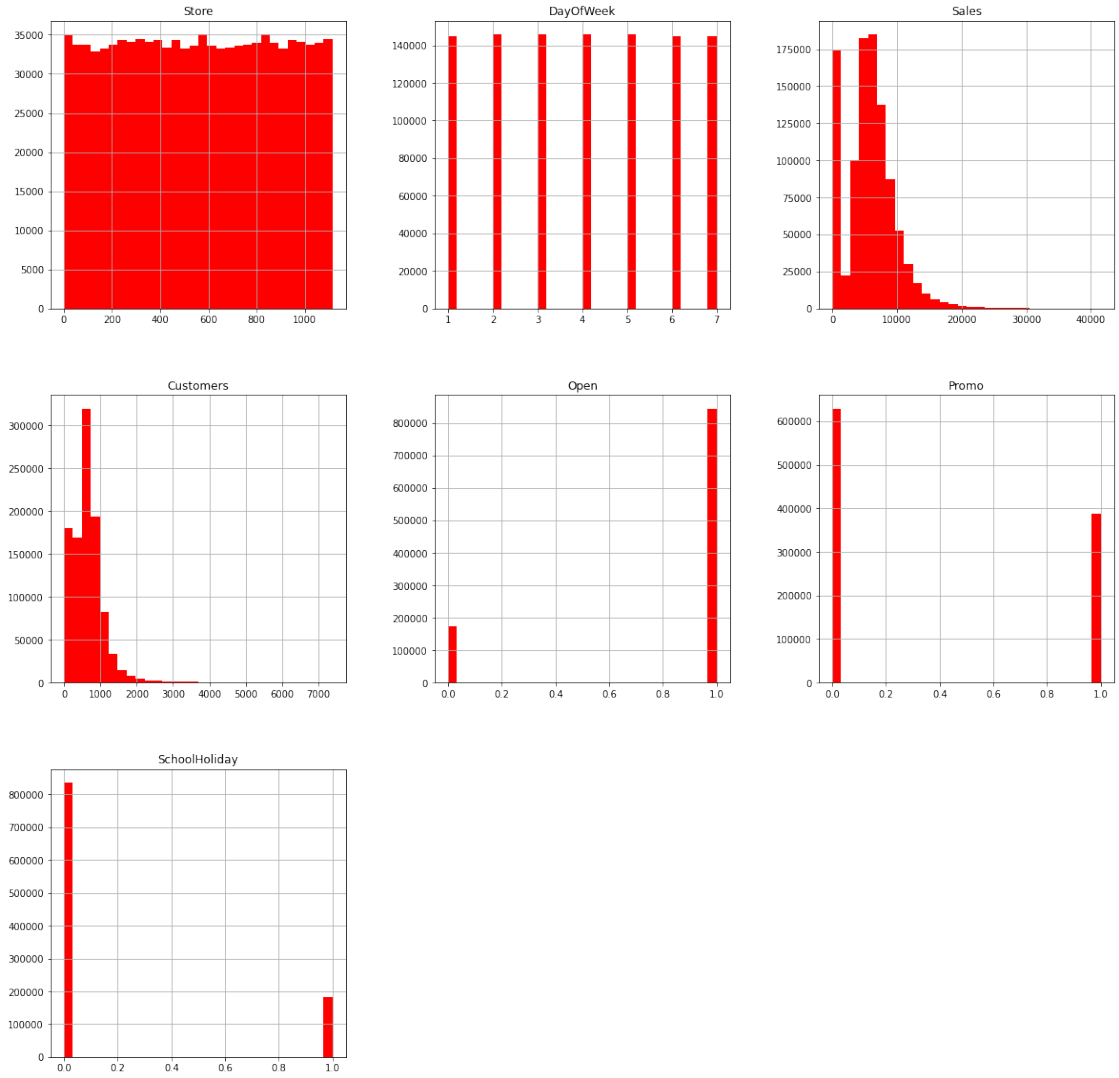
```
#check for any missing data
sales_train_df.isnull().sum()
```

```
Store          0
DayOfWeek      0
Date           0
Sales          0
Customers      0
Open           0
Promo          0
StateHoliday   0
SchoolHoliday  0
```

```
#sns heatmap of missing data
sns.heatmap(sales_train_df.isnull(), yticklabels = False,cbar = False, cmap = "Blues")
```



```
#sales train dataframe histogram
sales_train_df.hist(bins = 30, figsize = (20,20), color = 'r')
```



We can infer the following from the dataframe histogram function :

--Data is equally distributed among all stores

--Data is equally distributed across various Days of the week

--Stores are open approximately 80% of the time

--Promo1 was running approximately 40% of the time

--Average sales is around 5000-6000 Euros

--School holidays are around 18% of the time

```
# how many stores are open and closed with relation to sales
closed_train_zero_sales_df        = sales_train_df[(sales_train_df.Open == 0) &
(sales_train_df.Sales == 0)]
closed_train_non_zero_sales_df         = sales_train_df[(sales_train_df.Open == 0) &
(sales_train_df.Sales != 0)]
open_train_zero_sales_df        = sales_train_df[(sales_train_df.Open == 1) &
(sales_train_df.Sales == 0)]
open_train_non_zero_sales_df         = sales_train_df[(sales_train_df.Open == 1) &
(sales_train_df.Sales != 0)]

# Count the number of stores that are open and closed
print("Total =", len(sales_train_df))
print("Number of closed stores with zero sales  =", len(closed_train_df_1))
print("Number of closed stores with non zero sales =", len(closed_train_df_2))
print("Number of open stores with zero sales =", len(open_train_df_1))
print("Number of open stores with non zero sales =", len(open_train_df_2))
```

```
Total = 1017209
Number of closed stores with zero sales  = 172817
Number of closed stores with non zero sales = 0
Number of open stores with zero sales = 54
Number of open stores with non zero sales = 844338
```

There are 172817 closed stores in the data which is a significant number of observations. Also, there are 54 rows with open store but no sales on working days. We will drop these rows as these rows do not have any meaningful contribution and may create bias while forecasting.

```
#  remove closed stores and zero sales rows
sales_train_df = sales_train_df[(sales_train_df['Open'] != 0) & (sales_train_df['Sales'] !=
0)]

# drop the open column since it has no meaning now
sales_train_df.drop(['Open'], axis=1, inplace=True)

sales_train_df.shape
```

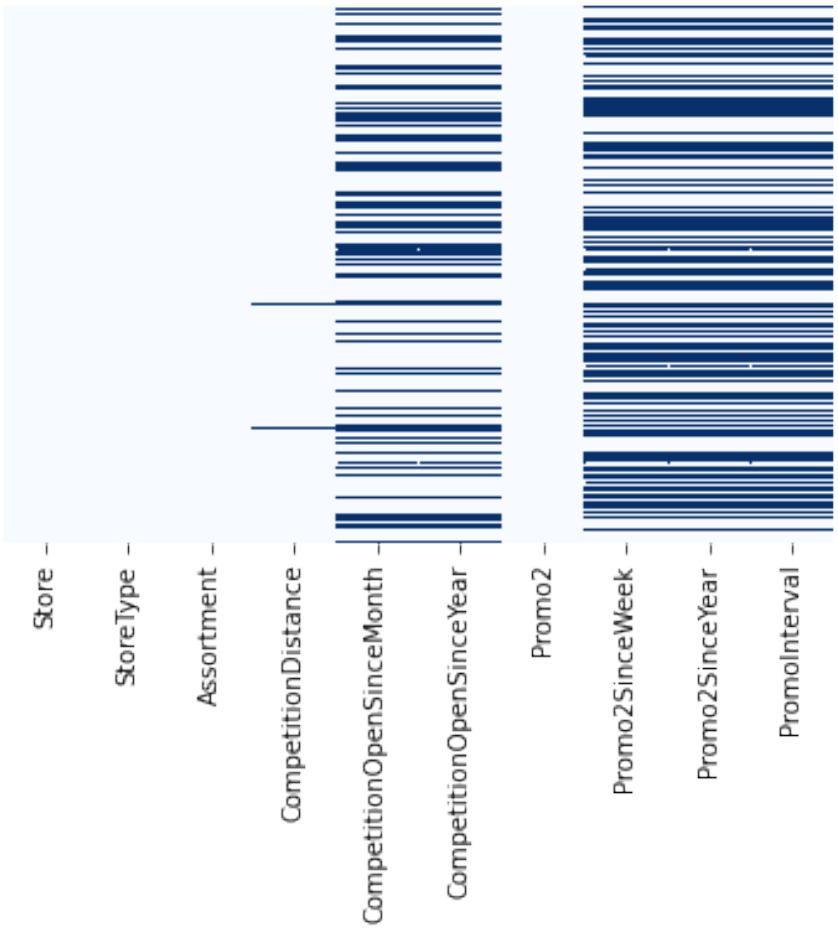
```
(844338, 8)
```

EDA OF STORES INFORMATION DATA :

```
#check for any missing data
store_info_df.isnull().sum()
```

```
Store                        0
StoreType                    0
Assortment                   0
CompetitionDistance          3
CompetitionOpenSinceMonth    354
CompetitionOpenSinceYear     354
Promo2                       0
Promo2SinceWeek              544
Promo2SinceYear              544
PromoInterval                544
```

```
#sns heatmap of missing data
sns.heatmap(store_info_df.isnull(), yticklabels = False, cbar = False, cmap="Blues")
```



Let us first start with the missing values in the 'CompetitionDistance' as ony 3 rows are missing

---

```
store_info_df[store_info_df['CompetitionDistance'].isnull()]
```

For this feature, let us fill the missing rows with the average values of the 'CompetitionDistance' column.

```
store_info_df['CompetitionDistance'].fillna(store_info_df['CompetitionDistance'].mean(), inplace = True)
```

Let us look at the missing values in the 'CompetitionOpenSinceMonth' feature in which 354 rows are missing which constitutes almost one third of the store count.

```
store_info_df[store_info_df['CompetitionOpenSinceMonth'].isnull()]
```

Let us check the rows with 'Promo2' equals 0

```
store_info_df[ store_info_df['Promo2'] == 0]
```

We can infer that, if 'promo2' feature is zero, values in features 'promo2SinceWeek', 'Promo2SinceYear', and 'PromoInterval' is also set to zero

And there are 354 rows where 'CompetitionOpenSinceYear' and 'CompetitionOpenSinceMonth' feature values are missing

We will set these values to zeros

```
column_features = ['Promo2SinceWeek', 'Promo2SinceYear', 'PromoInterval', 'CompetitionOpenSinceYear', 'CompetitionOpenSinceMonth']

for str in column_features:
    store_info_df [str].fillna(0, inplace = True)

#sns heatmap of missing values shows there are no missing values now
sns.heatmap(store_info_df.isnull(), yticklabels = False, cbar = False, cmap="Blues")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f03f6b50150>
```

EDA OF MERGED DATASET :

We will merge the sales_train_df and store_info_df data frames together based on 'store' column

---

```
sales_train_all_df = pd.merge(sales_train_df, store_info_df, how = 'inner', on = 'Store')
#sales_train_all_df.to_csv('test.csv', index=False)

sales_train_all_df.shape
```

(844338, 17)

```
sales_train_all_df.head(5)
```

Let us check the correlations with respect to sales by plotting the seaborn heatmap :

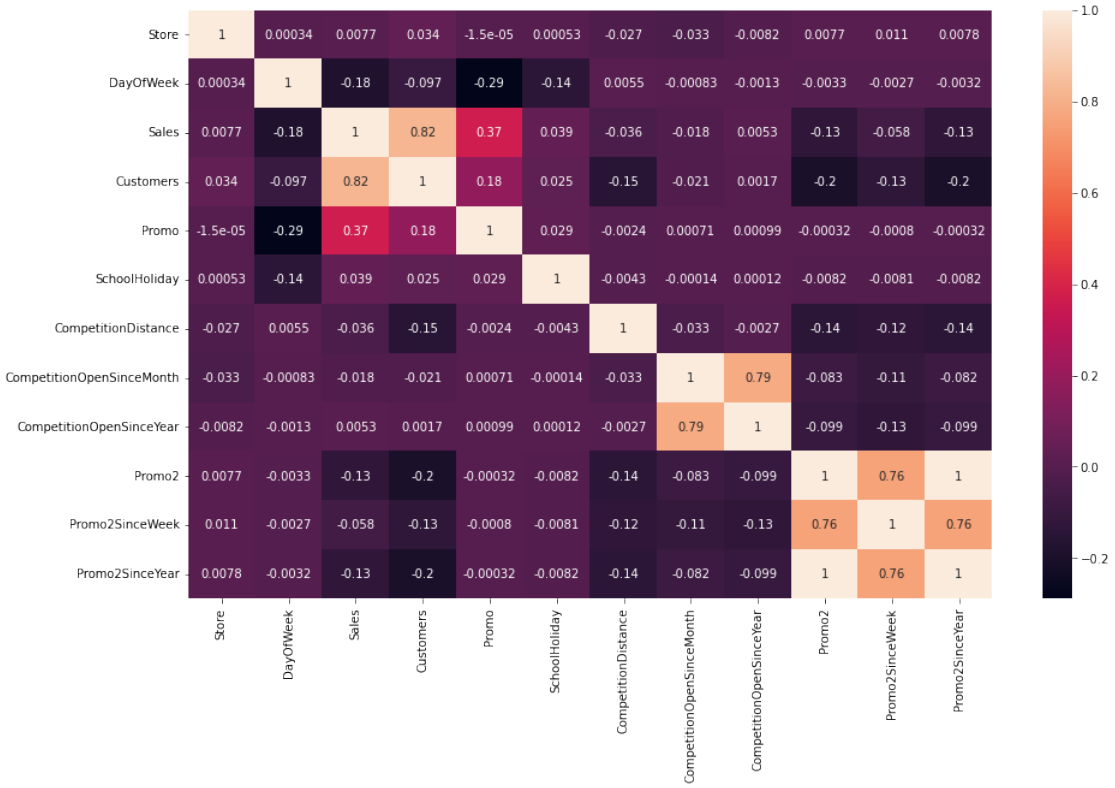
```
correlations = sales_train_all_df.corr()['Sales'].sort_values()
correlations
```

```
DayOfWeek                  -0.178753
Promo2SinceYear            -0.127581
Promo2                     -0.127556
Promo2SinceWeek            -0.058493
CompetitionDistance        -0.036401
CompetitionOpenSinceMonth  -0.018369
CompetitionOpenSinceYear    0.005257
Store                       0.007723
SchoolHoliday               0.038635
Promo                       0.368199
Customers                   0.823552
Sales                       1.000000
```

We can infer that customers and promo are positively correlated with the sales

Also, Promo2 does not seem to be effective at all

```
correlations = sales_train_all_df.corr()
f, ax = plt.subplots(figsize = (15, 9))
sns.heatmap(correlations, annot = True)
```



We can conclude that a strong positive correlation exists between the amount of Sales and Customers of a store.

We can also observe a positive correlation between the stores that had a Promo equal to 1(stores running promotions) and amount of Customers.

Also, for Promo2 equal to 1 (that is for consecutive promotion), the number of Customers and Sales has a negative correlation on the heatmap.

Let us separate the year and put it into a separate column

```
sales_train_all_df['Year'] = pd.DatetimeIndex(sales_train_all_df['Date']).year
```

```
sales_train_all_df
```

Let us do the same for the Day and Month

```
sales_train_all_df['Month'] = pd.DatetimeIndex(sales_train_all_df['Date']).month
sales_train_all_df['Day'] = pd.DatetimeIndex(sales_train_all_df['Date']).day
```

```
sales_train_all_df.shape
```

```
(844338, 20)
```

```
sales_train_all_df.head(5)
```

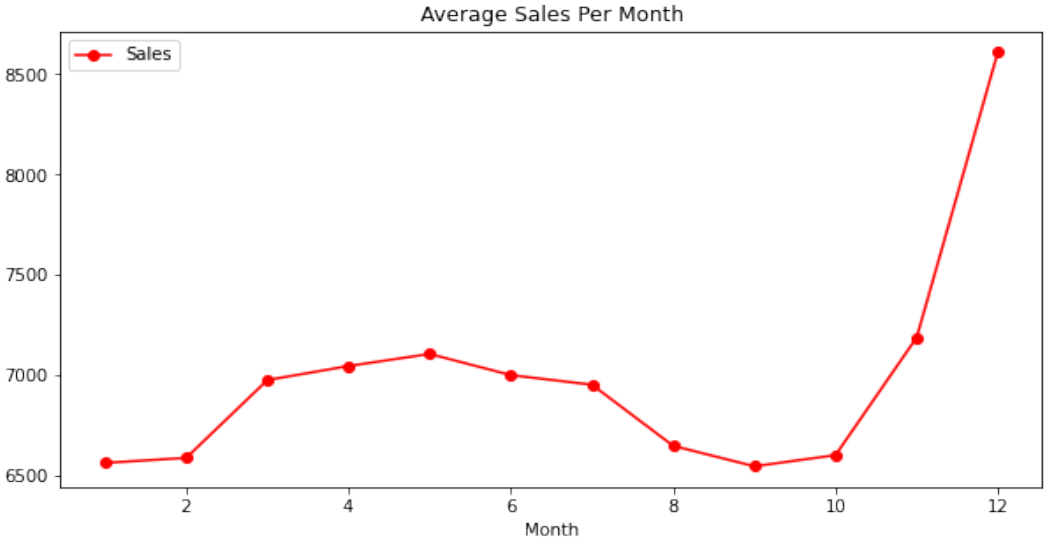
Let us take a look at the average sales and number of customers per month

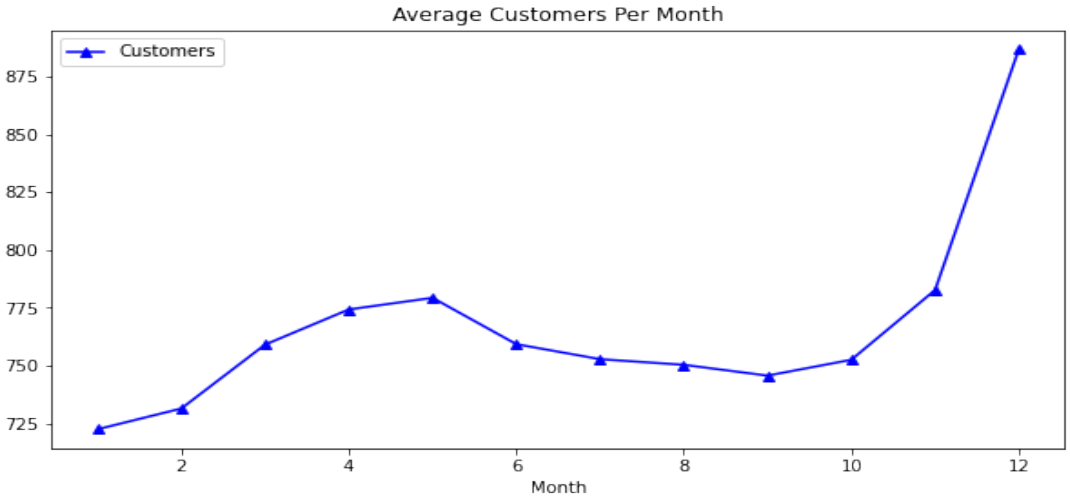
We can see that sales and number of customers peak around christmas timeframe

---

```
axis = sales_train_all_df.groupby('Month')[['Sales']].mean().plot(figsize = (10,5), marker = 'o', color = 'r')
axis.set_title('Average Sales Per Month')

plt.figure()
axis = sales_train_all_df.groupby('Month')[['Customers']].mean().plot(figsize = (10,5), marker = '^', color = 'b')
axis.set_title('Average Customers Per Month')
```





Let us take a look at the sales and customers per day of the month

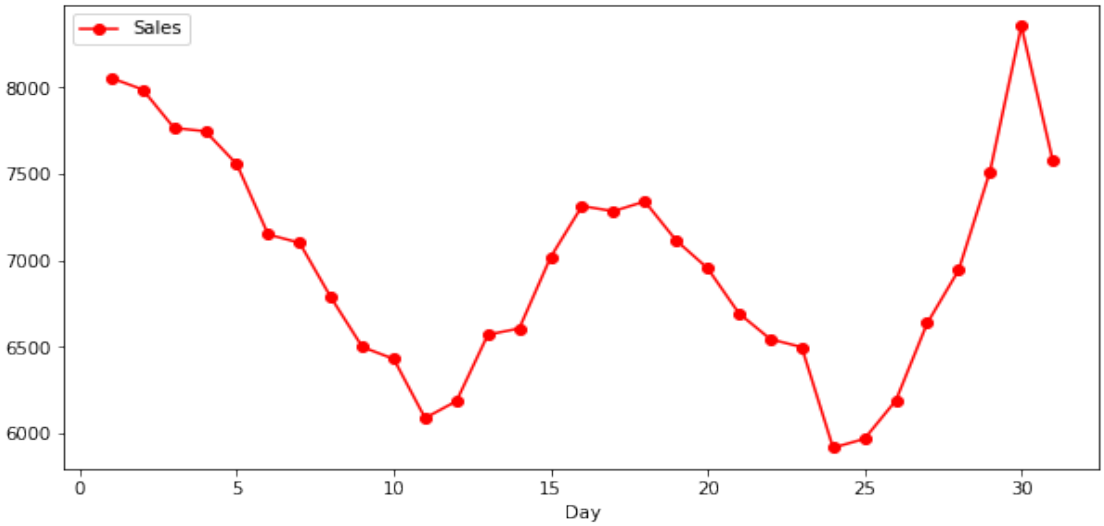
We can see that Minimum number of customers are generally around the 24th of the month

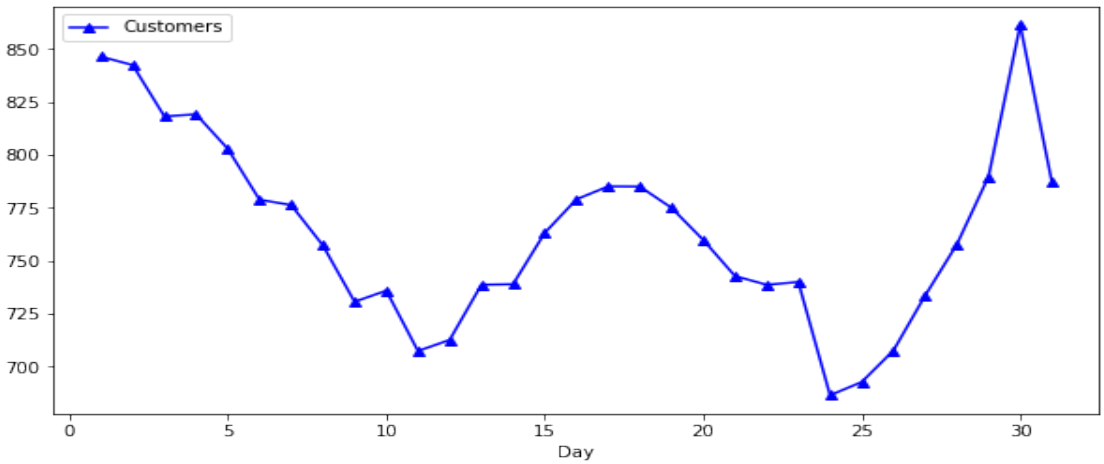
Also, most customers and sales are around 30th and 1st of the month

---

```
ax = sales_train_all_df.groupby('Day')[['Sales']].mean().plot(figsize = (10,5), marker = 'o', color = 'r')
axis.set_title('Average Sales Per Day')

plt.figure()
ax = sales_train_all_df.groupby('Day')[['Customers']].mean().plot(figsize = (10,5), marker = '^', color = 'b')
axis.set_title('Average Sales Per Day')
```





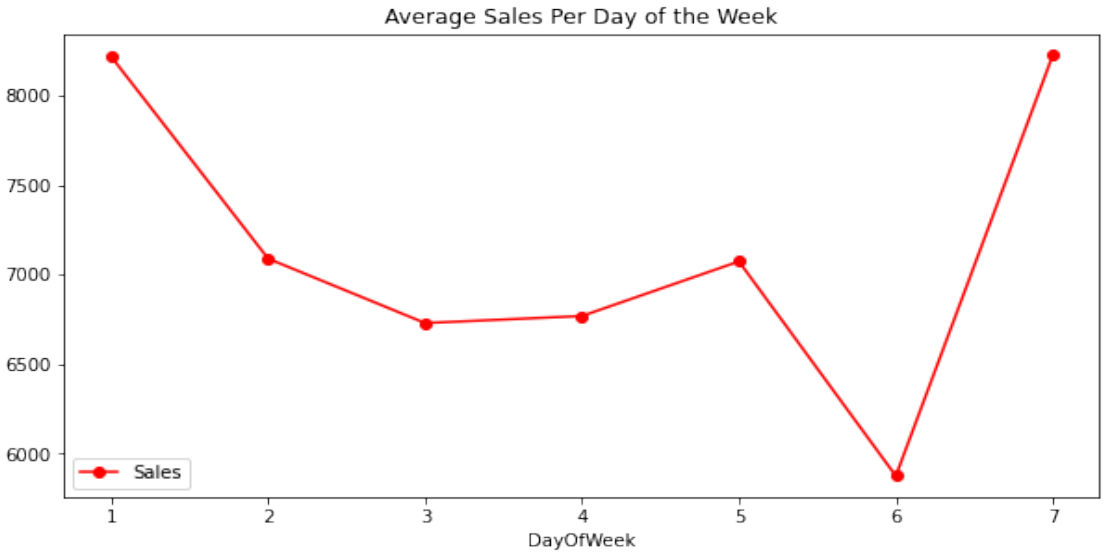
Let us take a look at the sales and customers per day of the week

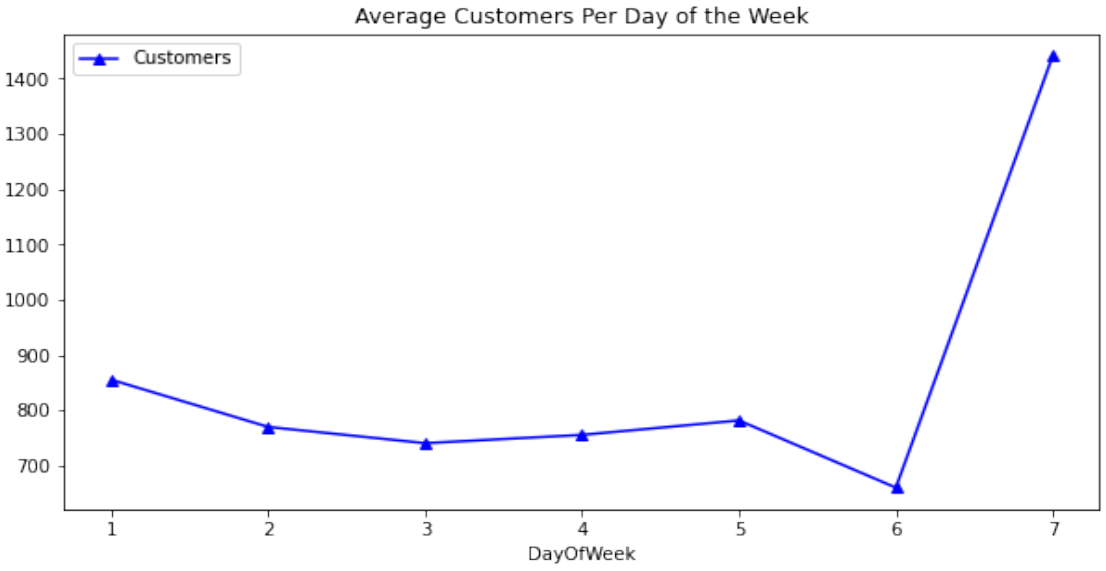
We can see that Minimum number of customers are generally around Saturday

Also, most customers and sales are on Sunday and Monday

```
axis = sales_train_all_df.groupby('DayOfWeek')[['Sales']].mean().plot(figsize = (10,5),
marker = 'o', color = 'r')
axis.set_title('Average Sales Per Day of the Week')

plt.figure()
axis = sales_train_all_df.groupby('DayOfWeek')[['Customers']].mean().plot(figsize =
(10,5), marker = '^', color = 'b')
axis.set_title('Average Customers Per Day of the Week')
```





BRIEF SUMMARY OF EDA :

1) A strong positive correlation exists between the amount of Sales and Customers of a store.

2) A positive correlation exists between the stores that had a Promo equal to 1(stores running promotions) and amount of Customers.

3) For observations with Promo2 equal to 1 (that is for consecutive promotion), the number of Customers and Sales has a negative correlation.

4) Sales and Number of customers peak around Christmas timeframe.

5) Minimum number of customers are generally around the 24th of the month.

6) Most customers and Sales are around 30th and 1st of the month.

7) Minimum number of customers are generally on Saturday.

8) Most customers and Sales are on Sunday and Monday.