In [1]:

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

```

In [2]:

```
#importing sales dataset
sales_train_df = pd.read_csv('D:/Data Science for Business Package/3. Sales Departme
sales_train_df.head()
```

```
C:\Users\Pradeep\AppData\Local\Temp\ipykernel_24732\3788187673.py:2: Dtype
Warning: Columns (7) have mixed types. Specify dtype option on import or s
et low_memory=False.
  sales_train_df = pd.read_csv('D:/Data Science for Business Package/3. Sa
les Department Data/train.csv')
```

Out[2]:

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	1	5	2015-07-31	5263	555	1	1	0	1
1	2	5	2015-07-31	6064	625	1	1	0	1
2	3	5	2015-07-31	8314	821	1	1	0	1
3	4	5	2015-07-31	13995	1498	1	1	0	1
4	5	5	2015-07-31	4822	559	1	1	0	1

In [3]:

```
# Id: transaction ID (combination of Store and date)
# Store: unique store Id
# 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
# SchoolHoliday: indicates if the (Store, Date) was affected by the closure of publi
```

In [4]:

```
1 sales_train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1017209 entries, 0 to 1017208
Data columns (total 9 columns):
 #   Column         Non-Null Count    Dtype 
---  ------         --------------    ----- 
 0   Store          1017209 non-null  int64 
 1   DayOfWeek      1017209 non-null  int64 
 2   Date           1017209 non-null  object
 3   Sales          1017209 non-null  int64 
 4   Customers      1017209 non-null  int64 
 5   Open           1017209 non-null  int64 
 6   Promo          1017209 non-null  int64 
 7   StateHoliday   1017209 non-null  object
 8   SchoolHoliday  1017209 non-null  int64 
dtypes: int64(7), object(2)
memory usage: 69.8+ MB
```

In [5]:

```
1 #checking for NULL values
2 sales_train_df.isnull().sum()
```

Out[5]:

```
Store            0
DayOfWeek        0
Date             0
Sales            0
Customers        0
Open             0
Promo            0
StateHoliday     0
SchoolHoliday    0
dtype: int64
```

In [6]:

```
1 sales_train_df.describe()
```

Out[6]:

	Store	DayOfWeek	Sales	Customers	Open	Promo
count	1.017209e+06	1.017209e+06	1.017209e+06	1.017209e+06	1.017209e+06	1.017209e+06
mean	5.584297e+02	3.998341e+00	5.773819e+03	6.331459e+02	8.301067e-01	3.815145e-01
std	3.219087e+02	1.997391e+00	3.849926e+03	4.644117e+02	3.755392e-01	4.857586e-01
min	1.000000e+00	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	2.800000e+02	2.000000e+00	3.727000e+03	4.050000e+02	1.000000e+00	0.000000e+00
50%	5.580000e+02	4.000000e+00	5.744000e+03	6.090000e+02	1.000000e+00	0.000000e+00
75%	8.380000e+02	6.000000e+00	7.856000e+03	8.370000e+02	1.000000e+00	1.000000e+00
max	1.115000e+03	7.000000e+00	4.155100e+04	7.388000e+03	1.000000e+00	1.000000e+00

In [7]:

```
1 #importing store info dateset
2 store_info_df = pd.read_csv('D:/Data Science for Business Package/3. Sales Departmen
3 store_info_df.head()
```

Out[7]:

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	Competi
0	1	c	a	1270.0	9.0	
1	2	a	a	570.0	11.0	
2	3	a	a	14130.0	12.0	
3	4	c	c	620.0	9.0	
4	5	a	a	29910.0	4.0	

In [8]:

```
1 # StoreType: categorical variable to indicate type of store (a, b, c, d)
2 # Assortment: describes an assortment level: a = basic, b = extra, c = extended
3 # CompetitionDistance (meters): distance to closest competitor store
4 # CompetitionOpenSince [Month/Year]: provides an estimate of the date when competiti
5 # Promo2: Promo2 is a continuing and consecutive promotion for some stores (0 = stor
6 # Promo2Since [Year/Week]: date when the store started participating in Promo2
7 # PromoInterval: describes the consecutive intervals Promo2 is started, naming the m
8
9
```

In [9]:

```
1 store_info_df.isnull().sum()
```

Out[9]:

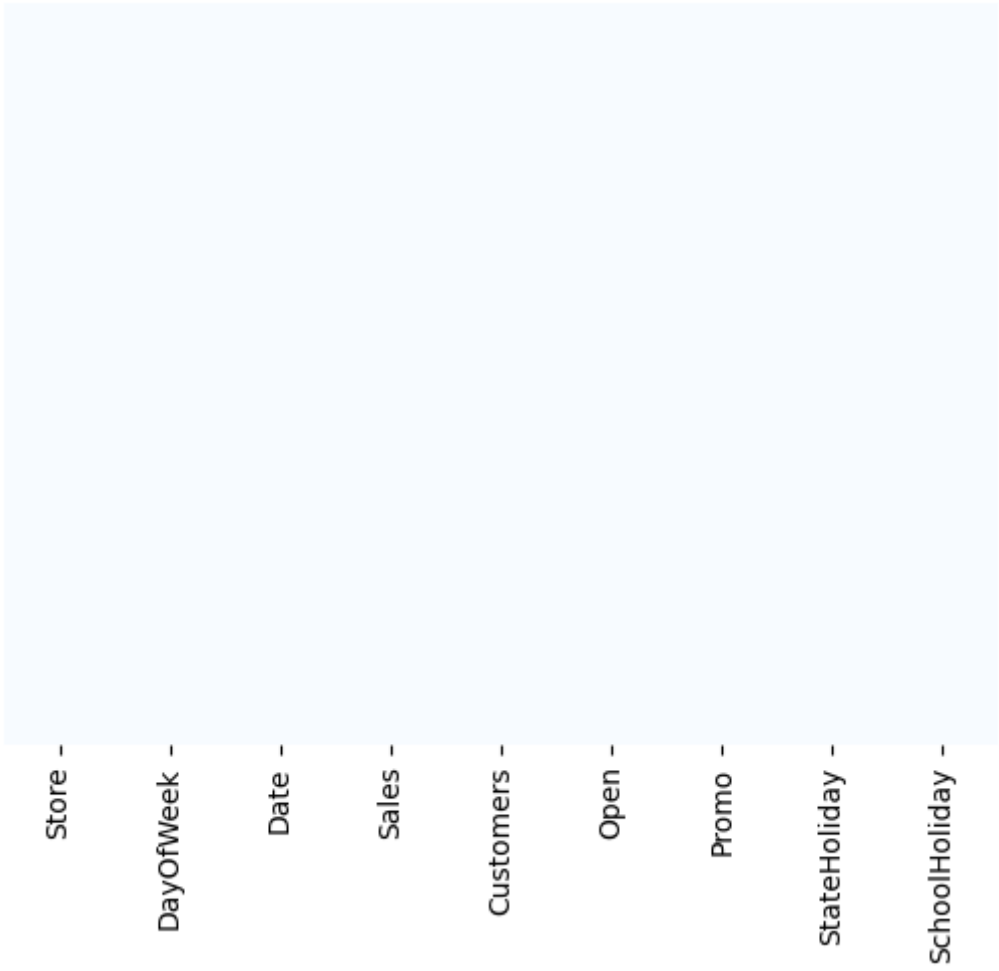
```
Store                        0
StoreType                    0
Assortment                   0
CompetitionDistance          3
CompetitionOpenSinceMonth  354
CompetitionOpenSinceYear   354
Promo2                       0
Promo2SinceWeek            544
Promo2SinceYear            544
PromoInterval              544
dtype: int64
```

In [10]:

```
1 #visualizing the null values if any
2 sns.heatmap(sales_train_df.isnull(), yticklabels = False, cbar = False, cmap="Blues"
3
```

Out[10]:

```
<AxesSubplot: >
```

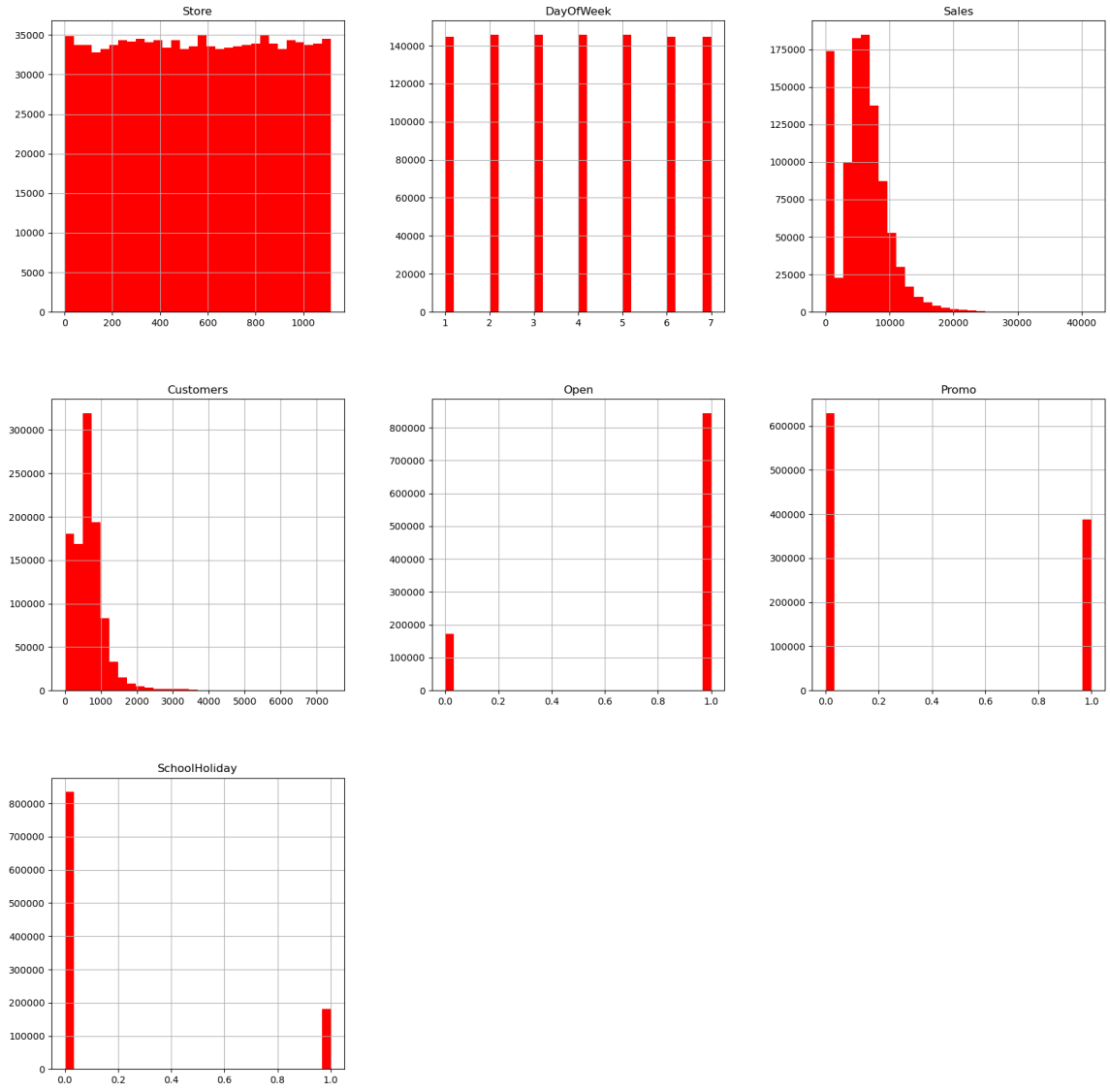


In [11]:

```
#visualizing numerical column
sales_train_df.hist(bins = 30, figsize = (20,20), color = 'r')
```

Out[11]:

```
array([[<AxesSubplot: title={'center': 'Store'}>,
        <AxesSubplot: title={'center': 'DayOfWeek'}>,
        <AxesSubplot: title={'center': 'Sales'}>],
       [<AxesSubplot: title={'center': 'Customers'}>,
        <AxesSubplot: title={'center': 'Open'}>,
        <AxesSubplot: title={'center': 'Promo'}>],
       [<AxesSubplot: title={'center': 'SchoolHoliday'}>,
        <AxesSubplot: >, <AxesSubplot: >]], dtype=object)
```



In [12]:

```
#stores that are open and closed
closed_train_df        = sales_train_df[sales_train_df['Open'] == 0]
open_train_df          = sales_train_df[sales_train_df['Open'] == 1]

print("Total =", len(sales_train_df))
print("Number of closed stores =", len(closed_train_df))
print("Number of open stores =", len(open_train_df))

```

```
Total = 1017209
Number of closed stores = 172817
Number of open stores = 844392
```

In [13]:

```
#removing closed stores
sales_train_df = sales_train_df[sales_train_df['Open'] == 1]
print(len(sales_train_df))
sales_train_df.head()
```

```
844392
```

Out[13]:

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	1	5	2015-07-31	5263	555	1	1	0	1
1	2	5	2015-07-31	6064	625	1	1	0	1
2	3	5	2015-07-31	8314	821	1	1	0	1
3	4	5	2015-07-31	13995	1498	1	1	0	1
4	5	5	2015-07-31	4822	559	1	1	0	1

In [14]:

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

Out[14]:

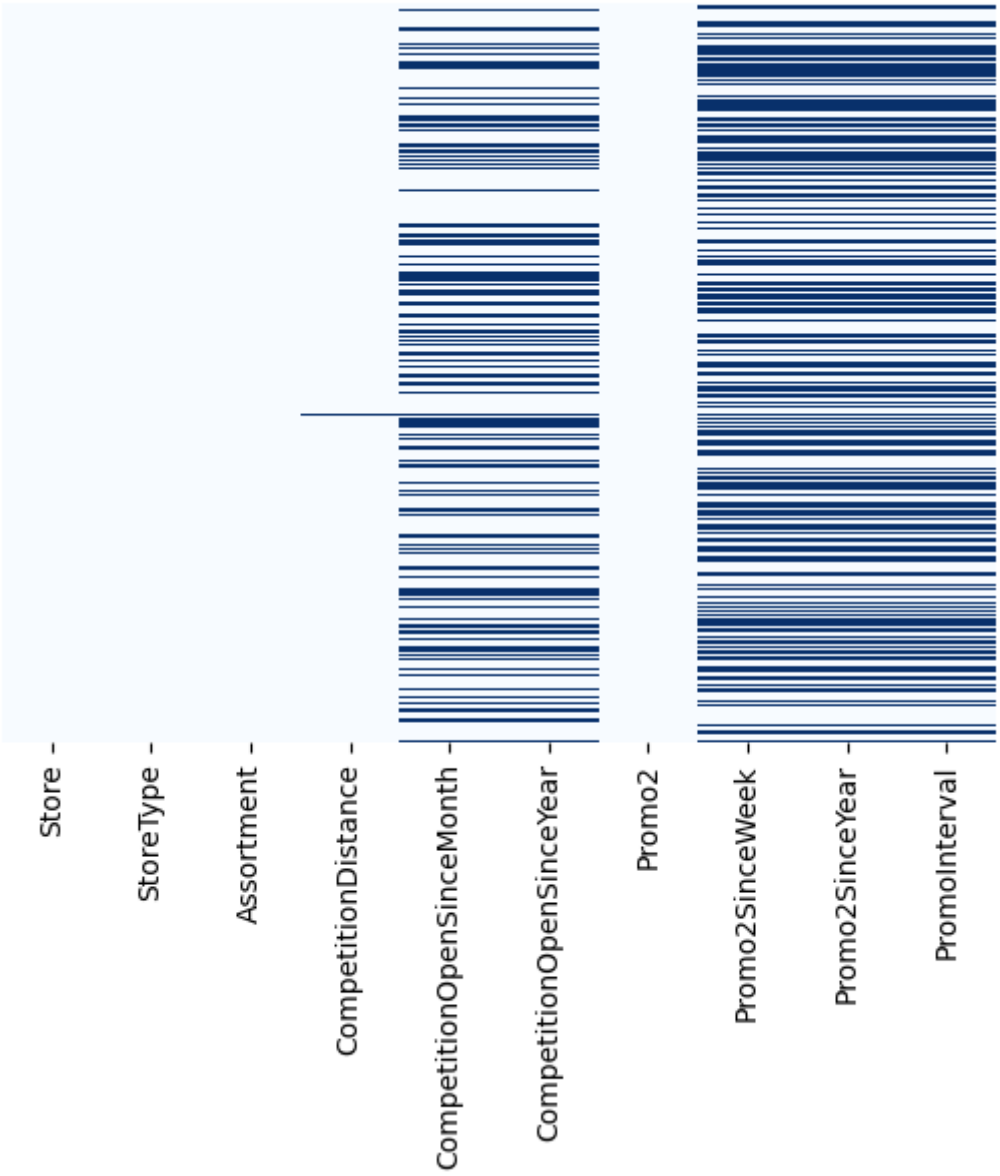
	Store	DayOfWeek	Date	Sales	Customers	Promo	StateHoliday	SchoolHoliday
0	1	5	2015-07-31	5263	555	1	0	1
1	2	5	2015-07-31	6064	625	1	0	1
2	3	5	2015-07-31	8314	821	1	0	1
3	4	5	2015-07-31	13995	1498	1	0	1
4	5	5	2015-07-31	4822	559	1	0	1

In [15]:

```
#visualizing NULL values for store dataset
sns.heatmap(store_info_df.isnull(), yticklabels = False, cbar = False, cmap="Blues")

```

Out[15]:

<AxesSubplot: >



In [16]:

```
#checkig NULL Values
store_info_df.isnull().sum()
```

Out[16]:

```
Store                         0
StoreType                     0
Assortment                    0
CompetitionDistance           3
CompetitionOpenSinceMonth   354
CompetitionOpenSinceYear    354
Promo2                        0
Promo2SinceWeek             544
Promo2SinceYear             544
PromoInterval               544
dtype: int64
```

In [17]:

```
#missing values in the 'CompetitionDistance'
store_info_df[store_info_df['CompetitionDistance'].isnull()]
```

Out[17]:

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	Comp
290	291	d	a	NaN	NaN	
621	622	a	c	NaN	NaN	
878	879	d	a	NaN	NaN	

In [18]:

```
#missing values in the 'CompetitionOpenSinceMonth'
store_info_df[store_info_df['CompetitionOpenSinceMonth'].isnull()]
```

Out[18]:

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	Com
11	12	a	c	1070.0	NaN	
12	13	d	a	310.0	NaN	
15	16	a	c	3270.0	NaN	
18	19	a	c	3240.0	NaN	
21	22	a	a	1040.0	NaN	
...	...	...	...	...	...	
1095	1096	a	c	1130.0	NaN	
1099	1100	a	a	540.0	NaN	
1112	1113	a	c	9260.0	NaN	
1113	1114	a	c	870.0	NaN	
1114	1115	d	c	5350.0	NaN	

354 rows × 10 columns

In [19]:

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

Out[19]:

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	Com
0	1	c	a	1270.0	9.0	
3	4	c	c	620.0	9.0	
4	5	a	a	29910.0	4.0	
5	6	a	a	310.0	12.0	
6	7	a	c	24000.0	4.0	
...	...	...	...	...	...	
1107	1108	a	a	540.0	4.0	
1109	1110	c	c	900.0	9.0	
1111	1112	c	c	1880.0	4.0	
1112	1113	a	c	9260.0	NaN	
1113	1114	a	c	870.0	NaN	

544 rows × 10 columns

In [20]:

```
#if 'promo2' is zero, 'promo2SinceWeek', 'Promo2SinceYear', and 'PromoInterval' info
str_cols = ['Promo2SinceWeek', 'Promo2SinceYear', 'PromoInterval', 'CompetitionOpenS

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

In [21]:

```
#checkig NULL Values
store_info_df.isnull().sum()
```

Out[21]:

```
Store                        0
StoreType                    0
Assortment                   0
CompetitionDistance          3
CompetitionOpenSinceMonth    0
CompetitionOpenSinceYear     0
Promo2                       0
Promo2SinceWeek              0
Promo2SinceYear              0
PromoInterval                0
dtype: int64
```

In [22]:

```
#average values of the 'CompetitionDistance' column
store_info_df['CompetitionDistance'].fillna(store_info_df['CompetitionDistance'].mea
```

In [23]:

```
#checkig NULL Values
store_info_df.isnull().sum()
```

Out[23]:

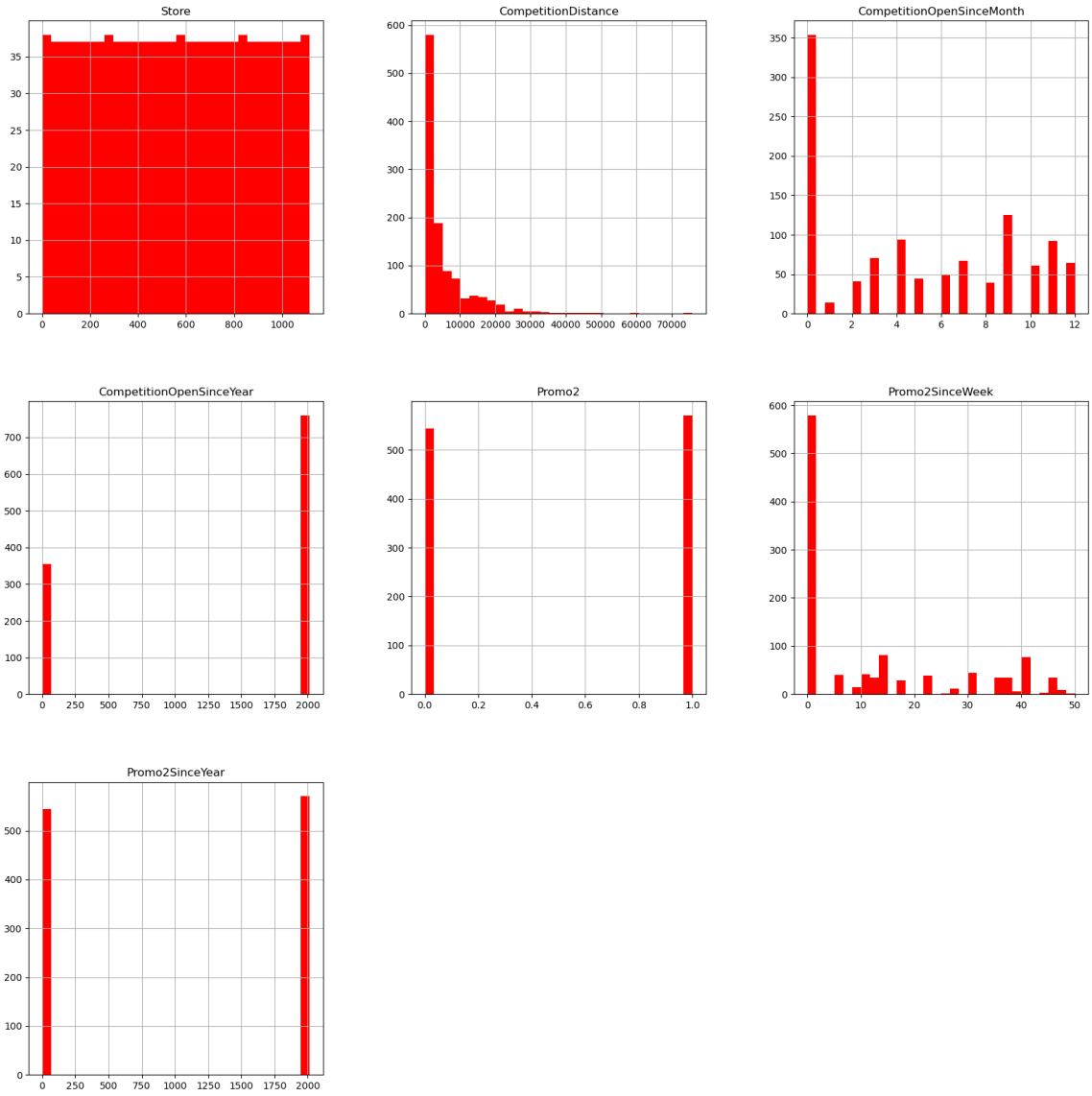
```
Store                        0
StoreType                    0
Assortment                   0
CompetitionDistance          0
CompetitionOpenSinceMonth    0
CompetitionOpenSinceYear     0
Promo2                       0
Promo2SinceWeek              0
Promo2SinceYear              0
PromoInterval                0
dtype: int64
```

In [24]:

```
1 store_info_df.hist(bins = 30, figsize = (20,20), color = 'r')
```

Out[24]:

```
array([[<AxesSubplot: title={'center': 'Store'}>,
        <AxesSubplot: title={'center': 'CompetitionDistance'}>,
        <AxesSubplot: title={'center': 'CompetitionOpenSinceMonth'}>],
       [<AxesSubplot: title={'center': 'CompetitionOpenSinceYear'}>,
        <AxesSubplot: title={'center': 'Promo2'}>,
        <AxesSubplot: title={'center': 'Promo2SinceWeek'}>],
       [<AxesSubplot: title={'center': 'Promo2SinceYear'}>,
        <AxesSubplot: >, <AxesSubplot: >]], dtype=object)
```



In [25]:

```
1 # merge both data frames together based on 'store'
2 sales_train_all_df = pd.merge(sales_train_df, store_info_df, how = 'inner', on = 'St
```

In [26]:

```
1 sales_train_all_df
```

Out[26]:

	Store	DayOfWeek	Date	Sales	Customers	Promo	StateHoliday	SchoolHoliday	S
0	1	5	2015-07-31	5263	555	1	0	1	
1	1	4	2015-07-30	5020	546	1	0	1	
2	1	3	2015-07-29	4782	523	1	0	1	
3	1	2	2015-07-28	5011	560	1	0	1	
4	1	1	2015-07-27	6102	612	1	0	1	
...	...	...	...	...	...	...	...	...	
844387	292	1	2013-01-07	9291	1002	1	0	0	
844388	292	6	2013-01-05	2748	340	0	0	0	
844389	292	5	2013-01-04	4202	560	0	0	1	
844390	292	4	2013-01-03	4580	662	0	0	1	
844391	292	3	2013-01-02	5076	672	0	0	1	

844392 rows × 17 columns

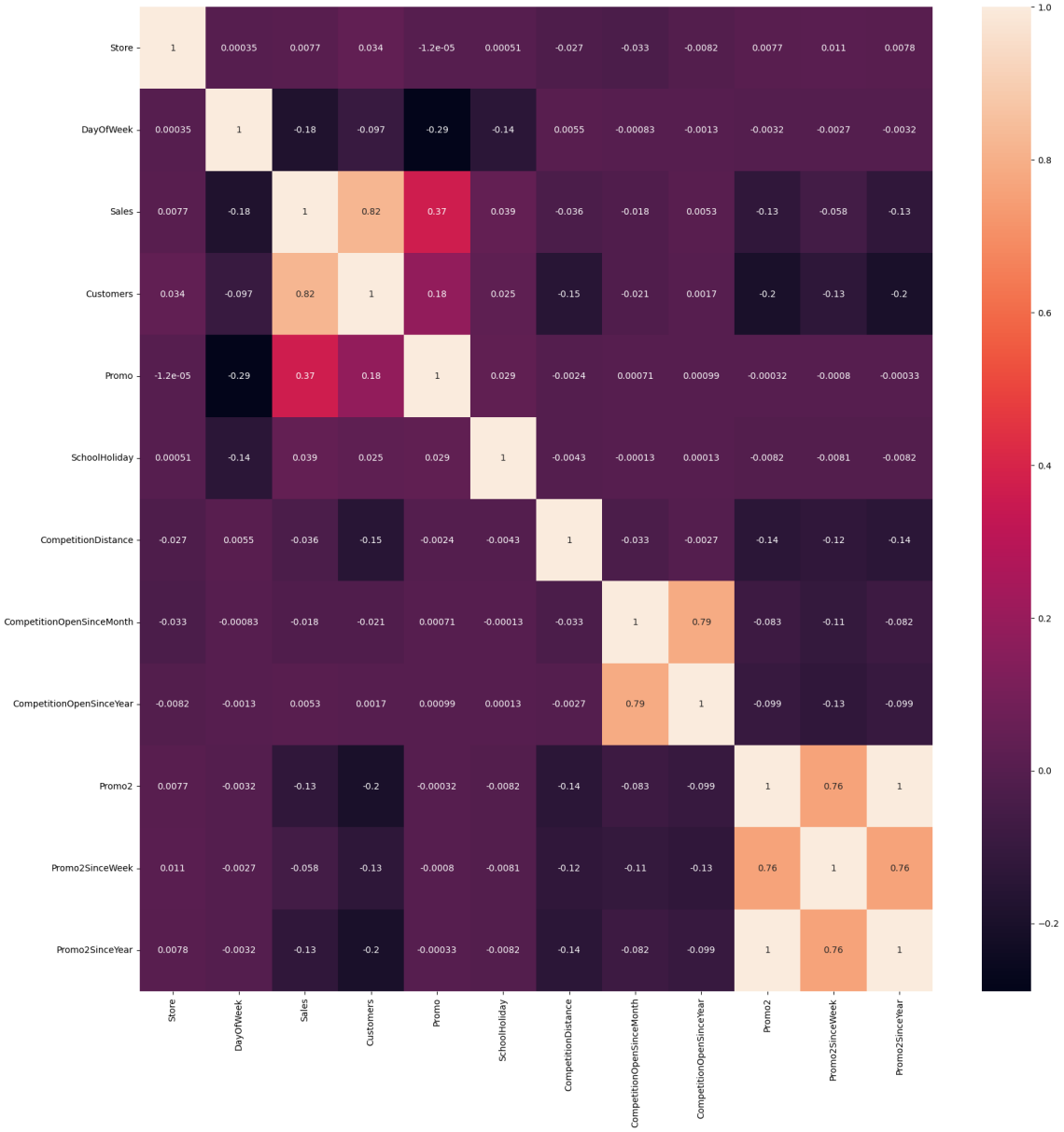
In [27]:

```
#correlation
correlations = sales_train_all_df.corr()
f, ax = plt.subplots(figsize = (20, 20))
sns.heatmap(correlations, annot = True)
# Customers/Prmo2 and sales are strongly correlated
```

```
C:\Users\Pradeep\AppData\Local\Temp\ipykernel_24732\1938977837.py:2: Futur
eWarning: The default value of numeric_only in DataFrame.corr is deprecate
d. In a future version, it will default to False. Select only valid column
s or specify the value of numeric_only to silence this warning.
  correlations = sales_train_all_df.corr()
```

Out[27]:

```
<AxesSubplot: >
```



In [28]:

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

```
C:\Users\Pradeep\AppData\Local\Temp\ipykernel_24732\1950390705.py:1: Futur
eWarning: The default value of numeric_only in DataFrame.corr is deprecate
d. In a future version, it will default to False. Select only valid column
s or specify the value of numeric_only to silence this warning.
  correlations = sales_train_all_df.corr()['Sales'].sort_values()
```

Out[28]:

```
DayOfWeek                  -0.178736
Promo2SinceYear            -0.127621
Promo2                     -0.127596
Promo2SinceWeek            -0.058476
CompetitionDistance        -0.036343
CompetitionOpenSinceMonth  -0.018370
CompetitionOpenSinceYear    0.005266
Store                       0.007710
SchoolHoliday               0.038617
Promo                       0.368145
Customers                   0.823597
Sales                       1.000000
Name: Sales, dtype: float64
```

In [29]:

```
# customers and promo are positively correlated with the sales
# Promo2 does not seem to be effective at all
```

In [30]:

```
#separating the year,month and day put it into a separate column
sales_train_all_df['Year'] = pd.DatetimeIndex(sales_train_all_df['Date']).year
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
```

Out[30]:

	Store	DayOfWeek	Date	Sales	Customers	Promo	StateHoliday	SchoolHoliday	S
0	1	5	2015-07-31	5263	555	1	0	1	
1	1	4	2015-07-30	5020	546	1	0	1	
2	1	3	2015-07-29	4782	523	1	0	1	
3	1	2	2015-07-28	5011	560	1	0	1	
4	1	1	2015-07-27	6102	612	1	0	1	
...	...	...	...	...	...	...	...	...	
844387	292	1	2013-01-07	9291	1002	1	0	0	
844388	292	6	2013-01-05	2748	340	0	0	0	
844389	292	5	2013-01-04	4202	560	0	0	1	
844390	292	4	2013-01-03	4580	662	0	0	1	
844391	292	3	2013-01-02	5076	672	0	0	1	

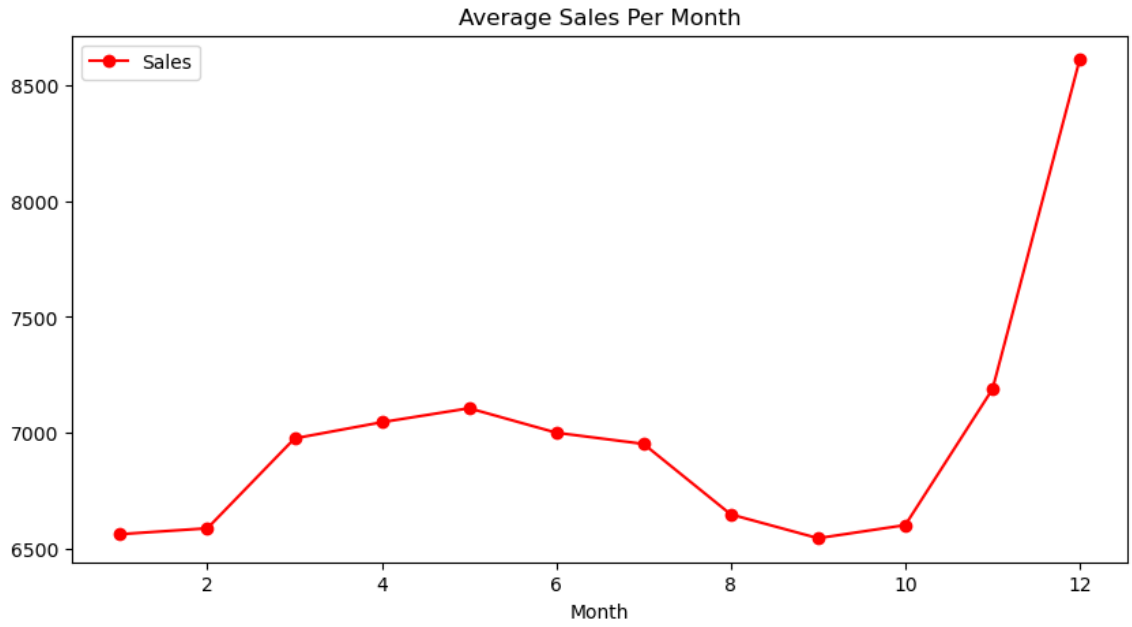
844392 rows × 20 columns

In [31]:

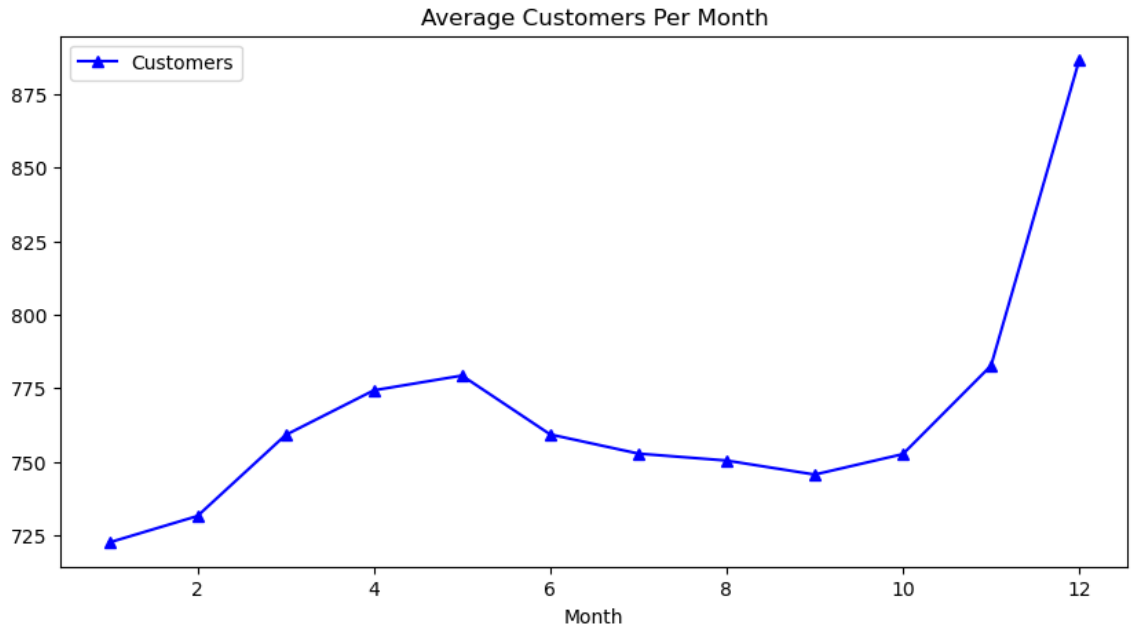
```
#average sales and customers per month
axis = sales_train_all_df.groupby('Month')[['Sales']].mean().plot(figsize = (10,5),
axis.set_title('Average Sales Per Month')

plt.figure()
axis = sales_train_all_df.groupby('Month')[['Customers']].mean().plot(figsize = (10,
axis.set_title('Average Customers Per Month')

```

Out[31]:

```
Text(0.5, 1.0, 'Average Customers Per Month')
```




```
<Figure size 640x480 with 0 Axes>
```

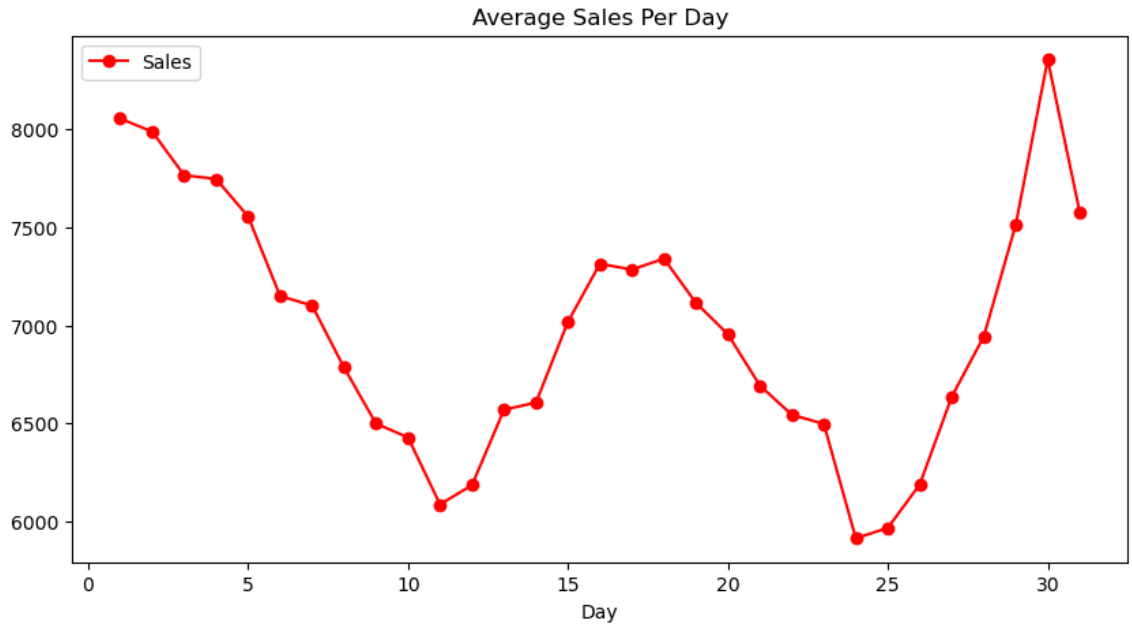


In [32]:

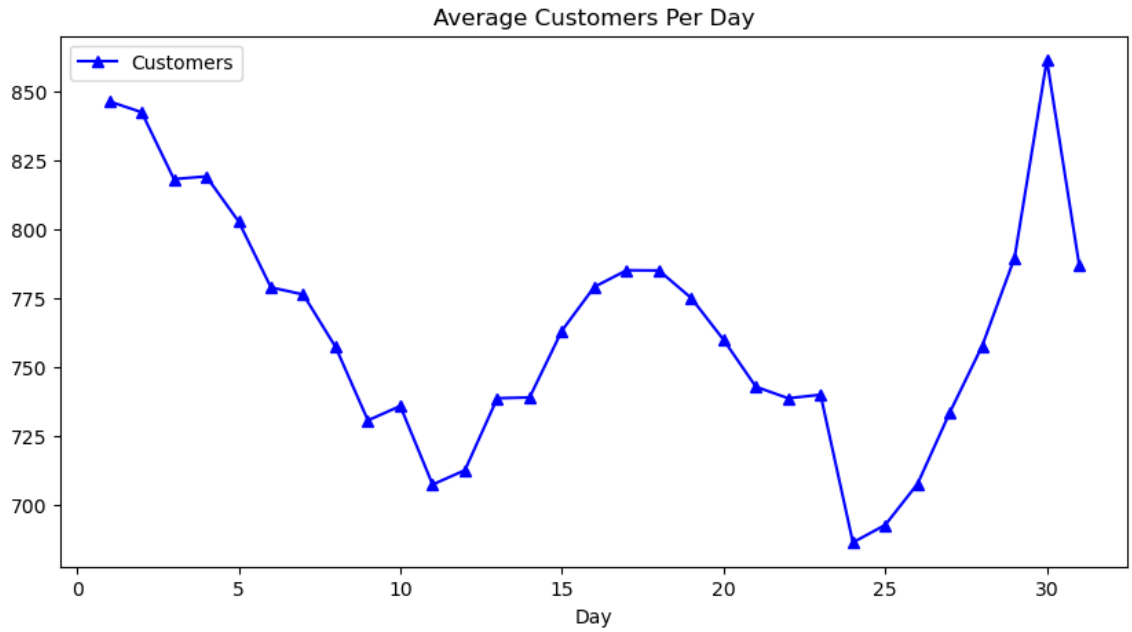
```
#average sales and customers per day
axis = sales_train_all_df.groupby('Day')[['Sales']].mean().plot(figsize = (10,5), ma
axis.set_title('Average Sales Per Day')

plt.figure()
axis = sales_train_all_df.groupby('Day')[['Customers']].mean().plot(figsize = (10,5)
axis.set_title('Average Customers Per Day')

```

Out[32]:

```
Text(0.5, 1.0, 'Average Customers Per Day')
```




```
<Figure size 640x480 with 0 Axes>
```

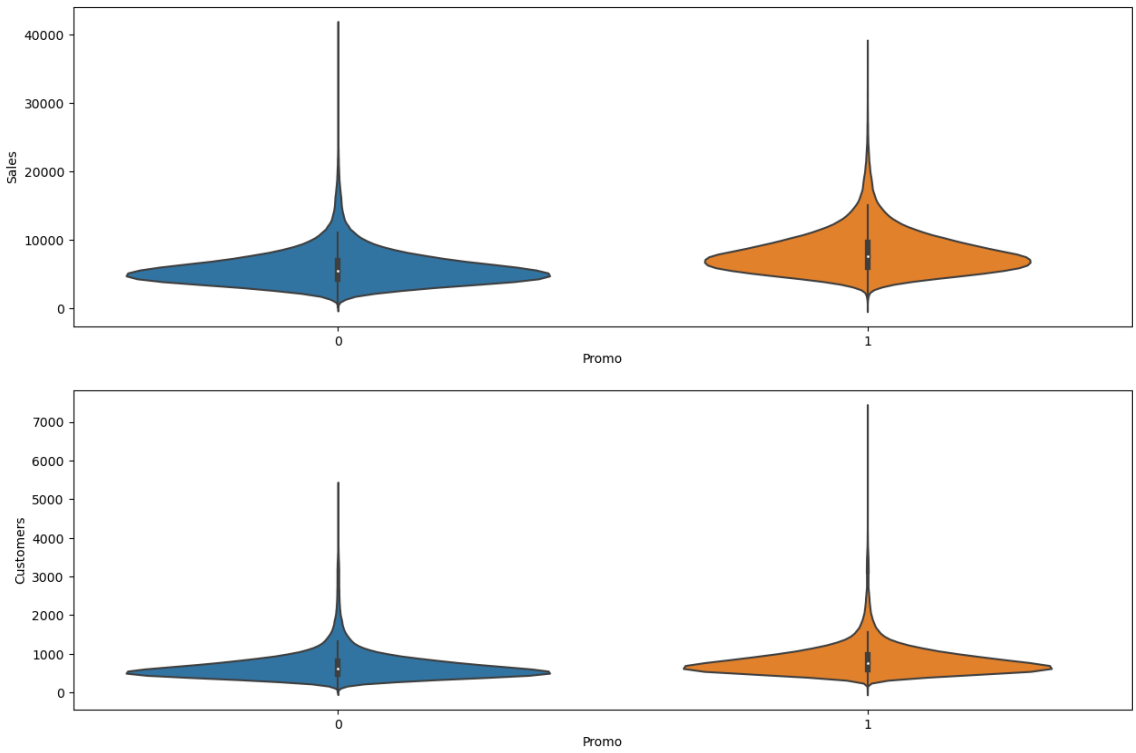


In [33]:

```
#violin plot
plt.figure(figsize=[15,10])

plt.subplot(211)
sns.violinplot(x = 'Promo', y = 'Sales', data = sales_train_all_df)

plt.subplot(212)
sns.violinplot(x = 'Promo', y = 'Customers', data = sales_train_all_df)
```

Out[33]:

```
<AxesSubplot: xlabel='Promo', ylabel='Customers'>
```



In [34]:

```
#training model
from prophet import Prophet
```

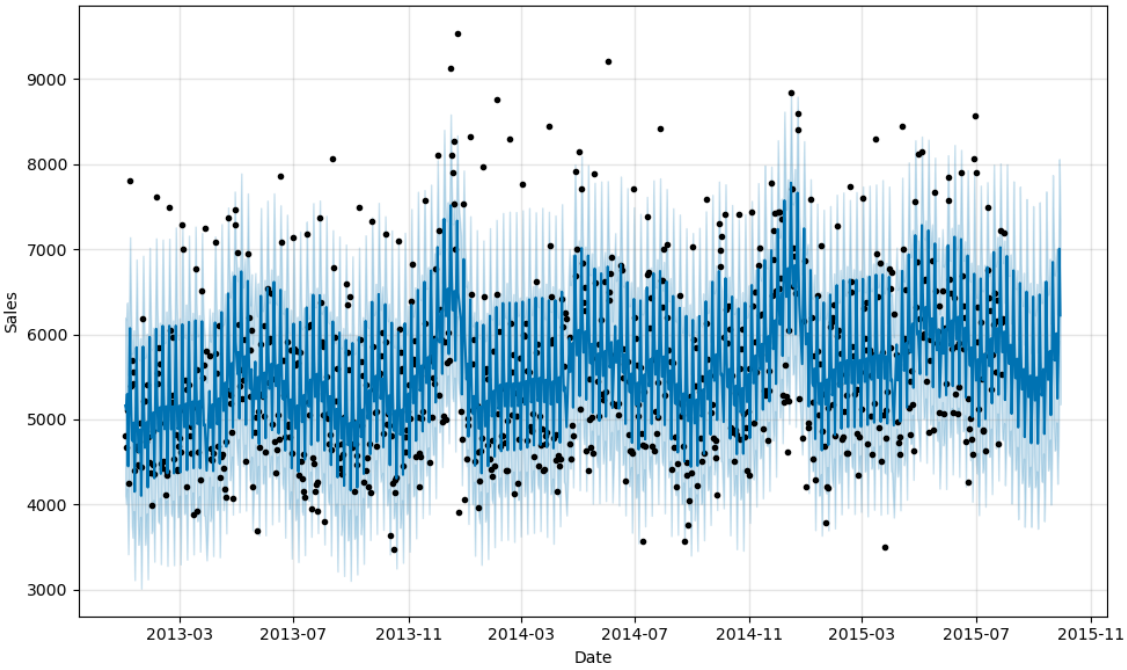
In [35]:

```
def sales_prediction(Store_ID, sales_df, periods):
  # Function that takes in the data frame, storeID, and number of future period fore
  # The function then generates date/sales columns in Prophet format
  # The function then makes time series predictions

  sales_df = sales_df[ sales_df['Store'] == Store_ID ]
  sales_df = sales_df[['Date', 'Sales']].rename(columns = {'Date': 'ds', 'Sales':'y'
  sales_df = sales_df.sort_values('ds')
  #print(sales_df)
  model    = Prophet()
  model.fit(sales_df)
  future   = model.make_future_dataframe(periods=periods)

  forecast = model.predict(future)
  figure   = model.plot(forecast, xlabel='Date', ylabel='Sales')
  figure2  = model.plot_components(forecast)
```

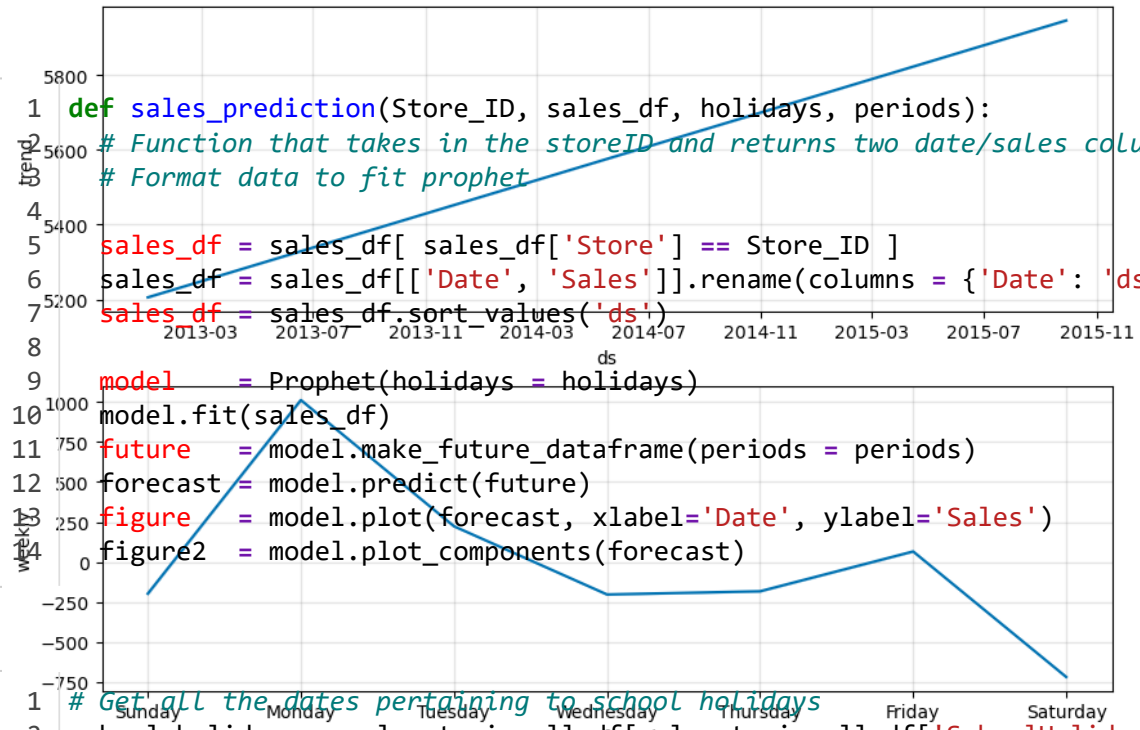
In [36]:

```
1 sales_prediction(10, sales_train_all_df, 60)
```

```
18:24:35 - cmdstanpy - INFO - Chain [1] start processing
18:24:35 - cmdstanpy - INFO - Chain [1] done processing
```



```
def sales_prediction(Store_ID, sales_df, holidays, periods):
# Function that takes in the storeID and returns two date/sales columns in Prophet
# Format data to fit prophet

sales_df = sales_df[ sales_df['Store'] == Store_ID ]
sales_df = sales_df[['Date', 'Sales']].rename(columns = {'Date': 'ds', 'Sales':'y'
sales_df = sales_df.sort_values('ds')

model    = Prophet(holidays = holidays)
model.fit(sales_df)
future   = model.make_future_dataframe(periods = periods)
forecast = model.predict(future)
figure   = model.plot(forecast, xlabel='Date', ylabel='Sales')
figure2  = model.plot_components(forecast)
```




```
# Get all the dates pertaining to school holidays
school_holidays = sales_train_all_df[sales_train_all_df['SchoolHoliday'] == 1].loc[:
school_holidays.shape
```

Out[38]:

```
(163457,)
```

```
# Get all the dates pertaining to state holidays
state_holidays = sales_train_all_df [ (sales_train_all_df['StateHoliday'] == 'a') |
state_holidays.shape
```

Out[39]:

```
(910,)
```

In [40]:

```
state_holidays = pd.DataFrame({'ds': pd.to_datetime(state_holidays),
                               'holiday': 'state_holiday'})
print(state_holidays)
```

```
           ds        holiday
0   2014-10-03  state_holiday
1   2013-10-03  state_holiday
2   2015-06-04  state_holiday
3   2014-06-19  state_holiday
4   2013-05-30  state_holiday
..         ...            ...
905 2013-04-01  state_holiday
906 2013-08-15  state_holiday
907 2015-06-04  state_holiday
908 2014-06-19  state_holiday
909 2013-05-30  state_holiday

[910 rows x 2 columns]
```

In [41]:

```
school_holidays = pd.DataFrame({'ds': pd.to_datetime(school_holidays),
                                'holiday': 'school_holiday'})
print(school_holidays)
```

```
              ds         holiday
0      2015-07-31  school_holiday
1      2015-07-30  school_holiday
2      2015-07-29  school_holiday
3      2015-07-28  school_holiday
4      2015-07-27  school_holiday
...           ...             ...
163452 2013-02-05  school_holiday
163453 2013-02-04  school_holiday
163454 2013-01-04  school_holiday
163455 2013-01-03  school_holiday
163456 2013-01-02  school_holiday

[163457 rows x 2 columns]
```

In [42]:

```
# concatenate both school and state holidays
school_state_holidays = pd.concat((state_holidays, school_holidays))

school_state_holidays.groupby('holiday').holiday.count()
```

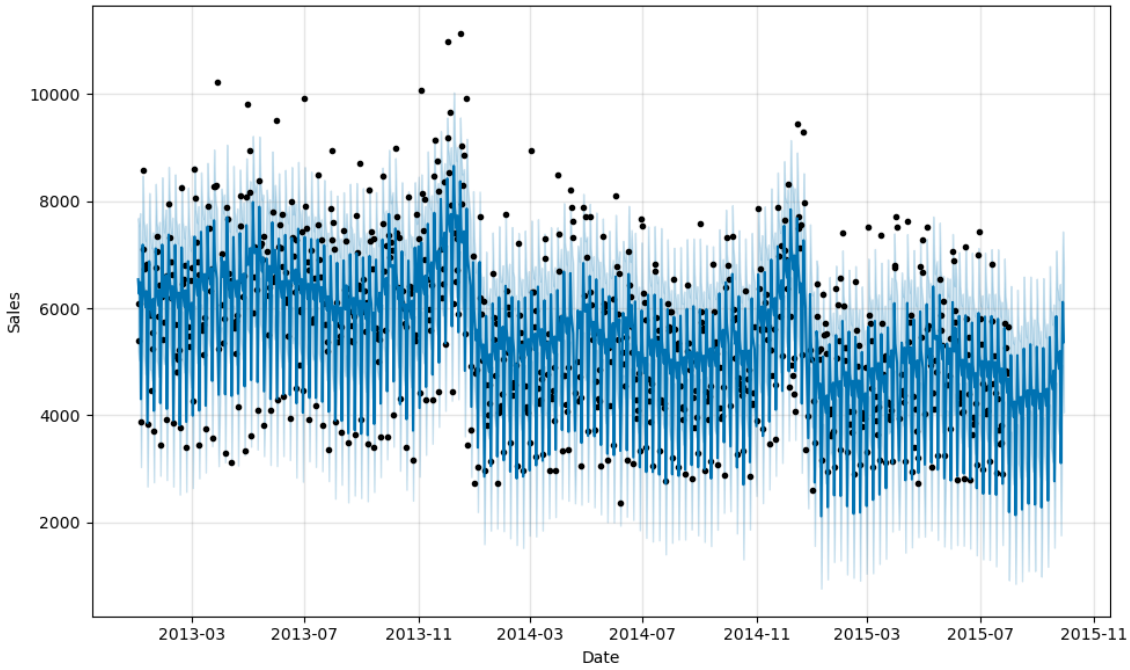
Out[42]:

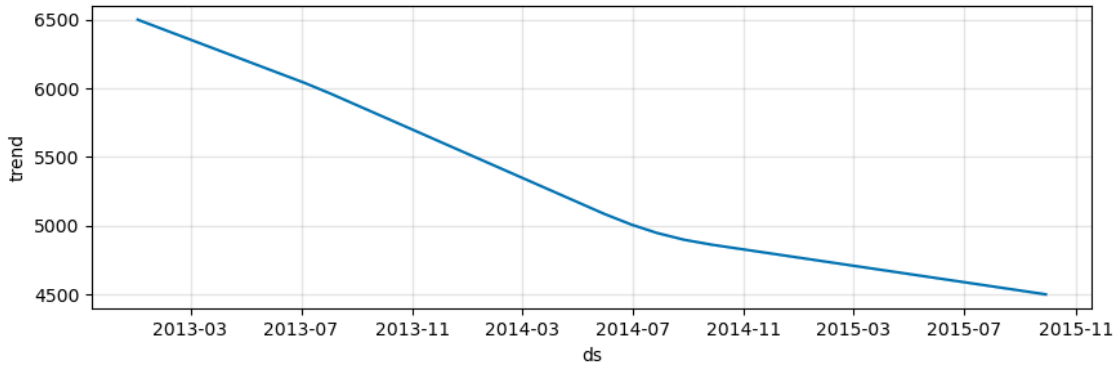
```
holiday
school_holiday    163457
state_holiday        910
Name: holiday, dtype: int64
```

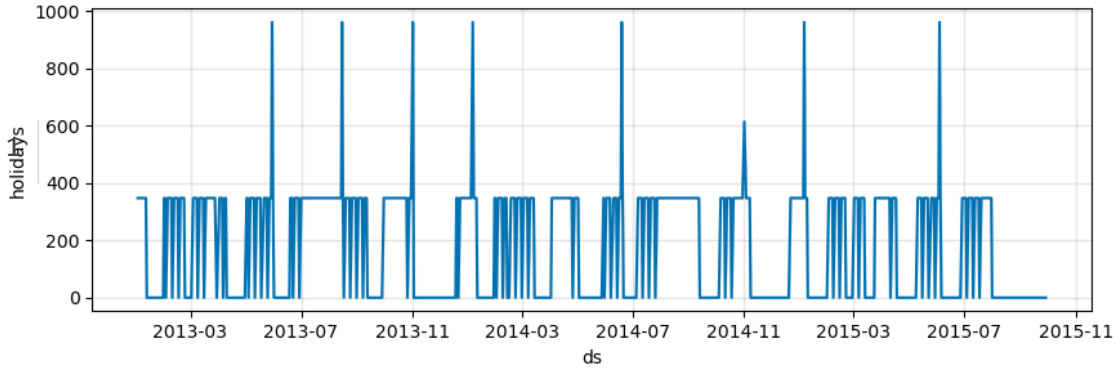
In [43]:

```
# Let's make predictions using holidays for a specific store
sales_prediction(6, sales_train_all_df, school_state_holidays, 60)
```

```
18:24:53 - cmdstanpy - INFO - Chain [1] start processing
18:24:53 - cmdstanpy - INFO - Chain [1] done processing
```




6500
6000
5500
5000
4500
trend
2013-03
2013-07
2013-11
2014-03
2014-07
2014-11
2015-03
2015-07
2015-11
ds


1000
800
600
400
200
0
holidays
2013-03
2013-07
2013-11
2014-03
2014-07
2014-11
2015-03
2015-07
2015-11
ds


1000