IMPORTAR LAS LIBRERÍAS Y LOS DATASETS

In [40]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import datetime
import warnings
warnings.filterwarnings("ignore")
```

IMPORTAR EL DATASET DE VENTAS

In [41]:
```python
sales_train_df = pd.read_csv("train.csv")
```

In [42]:
```python
sales_train_df.head(5)
# En millón de observaciones
# 1115 tiendas únicas
# Las ventas es la variable objetivo (la que se intenta predecir)

# Id: ID de transacción (combinación de la tienda y la fecha)
# Store: identificador único de la tienda
# Sales: ventas diarias, esta es la variable objetivo
# Customers: número de clientes de un día dado
# Open: Booleano para indicar si la tienda estaba abierta o cerrada (0 = cerrada, 1 = abierta)
# Promo: describe si la tienda tenía algún tipo de promoción ese día o no
# StateHoliday: indica si el día era festivo o no (a = vacaciones públicas, b = vacaciones de Pascua, c = Navid
# SchoolHoliday: indica si (Store, Date) se vio afectado por el cierre de las escuelas públicas
```

Out[42]:

	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 [43]:
```python
sales_train_df.tail(10)
```

Out[43]:

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
1017199	1106	2	2013-01-01	0	0	0	0	a	1
1017200	1107	2	2013-01-01	0	0	0	0	a	1
1017201	1108	2	2013-01-01	0	0	0	0	a	1
1017202	1109	2	2013-01-01	0	0	0	0	a	1
1017203	1110	2	2013-01-01	0	0	0	0	a	1
1017204	1111	2	2013-01-01	0	0	0	0	a	1
1017205	1112	2	2013-01-01	0	0	0	0	a	1
1017206	1113	2	2013-01-01	0	0	0	0	a	1
1017207	1114	2	2013-01-01	0	0	0	0	a	1
1017208	1115	2	2013-01-01	0	0	0	0	a	1

In [44]:
```python
sales_train_df.info()
# 9 columnas en total
# 8 características, cada una con 1017209 puntos de datos
# 1 variable objetivo (ventas)
```

```
<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 [45]:
```python
sales_train_df.describe()
# Cantidad de ventas promedio por día = 5773 Euros, ventas mínimas por día = 0, ventas máximas por día = 41551
# Número medio de clientes = 633, número mínimo de clientes = 0, número máximo de clientes = 7388
```

Out[45]:

	Store	DayOfWeek	Sales	Customers	Open	Promo	SchoolHoliday
count	1.017209e+06	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	1.786467e-01
std	3.219087e+02	1.997391e+00	3.849926e+03	4.644117e+02	3.755392e-01	4.857586e-01	3.830564e-01
min	1.000000e+00	1.000000e+00	0.000000e+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	0.000000e+00
50%	5.580000e+02	4.000000e+00	5.744000e+03	6.090000e+02	1.000000e+00	0.000000e+00	0.000000e+00
75%	8.380000e+02	6.000000e+00	7.856000e+03	8.370000e+02	1.000000e+00	1.000000e+00	0.000000e+00
max	1.115000e+03	7.000000e+00	4.155100e+04	7.388000e+03	1.000000e+00	1.000000e+00	1.000000e+00

IMPORTAR LA INFORMACIÓN SOBRE LAS TIENDAS

In [46]:
```python
store_info_df = pd.read_csv("store.csv")
# StoreType: categoría que indica el tipo de tienda (a, b, c, d)
# Assortment: a = básico, b = extra, c = extendido
# CompetitionDistance (en metros): distancia a la tienda de la competencia más cercana
# CompetitionOpenSince [Mes/Año]: fecha en que abrió la competencia
# Promo2: Promo2 es una promoción continuada y consecutiva en algunas tiendas (0 = la tienda no participa, 1 =
# Promo2Since [Año/Semana]: fecha en la que la tienda empieza a participar en la Promo2
# PromoInterval: describe los intervalos consecutivos donde la Promo2 empieza, indicando los meses en los que
```

In [47]:
```python
store_info_df.head(5)
```

Out[47]:

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek
0	1	c	a	1270.0	9.0	2008.0	0	NaN
1	2	a	a	570.0	11.0	2007.0	1	13.0
2	3	a	a	14130.0	12.0	2006.0	1	14.0
3	4	c	c	620.0	9.0	2009.0	0	NaN
4	5	a	a	29910.0	4.0	2015.0	0	NaN

In [48]:
```python
# Hay que tener en cuenta que el data frame anterior incluye las transacciones registradas por día (en millones)
# Este data frame solo incluye información sobre las 1115 tiendas exclusivas que forman parte de este estudio
store_info_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1115 entries, 0 to 1114
Data columns (total 10 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Store                      1115 non-null   int64
 1   StoreType                  1115 non-null   object
 2   Assortment                 1115 non-null   object
 3   CompetitionDistance        1112 non-null   float64
 4   CompetitionOpenSinceMonth  761 non-null    float64
 5   CompetitionOpenSinceYear   761 non-null    float64
 6   Promo2                     1115 non-null   int64
 7   Promo2SinceWeek            571 non-null    float64
 8   Promo2SinceYear            571 non-null    float64
 9   PromoInterval              571 non-null    object
dtypes: float64(5), int64(2), object(3)
memory usage: 87.2+ KB
```

In [49]:
```python
store_info_df.describe()
# De media, la distancia de la competencia es de 5404 metros (5,4 kms)
```

Out[49]:

	Store	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2Since
count	1115.00000	1112.000000	761.000000	761.000000	1115.000000	571.000000	571.00
mean	558.00000	5404.901079	7.224704	2008.668857	0.512108	23.595447	2011.76
std	322.01708	7663.174720	3.212348	6.195983	0.500078	14.141984	1.67
min	1.00000	20.000000	1.000000	1900.000000	0.000000	1.000000	2009.00
25%	279.50000	717.500000	4.000000	2006.000000	0.000000	13.000000	2011.00
50%	558.00000	2325.000000	8.000000	2010.000000	1.000000	22.000000	2012.00
75%	836.50000	6882.500000	10.000000	2013.000000	1.000000	37.000000	2013.00
max	1115.00000	75860.000000	12.000000	2015.000000	1.000000	50.000000	2015.00

EXPLORAR EL DATASET

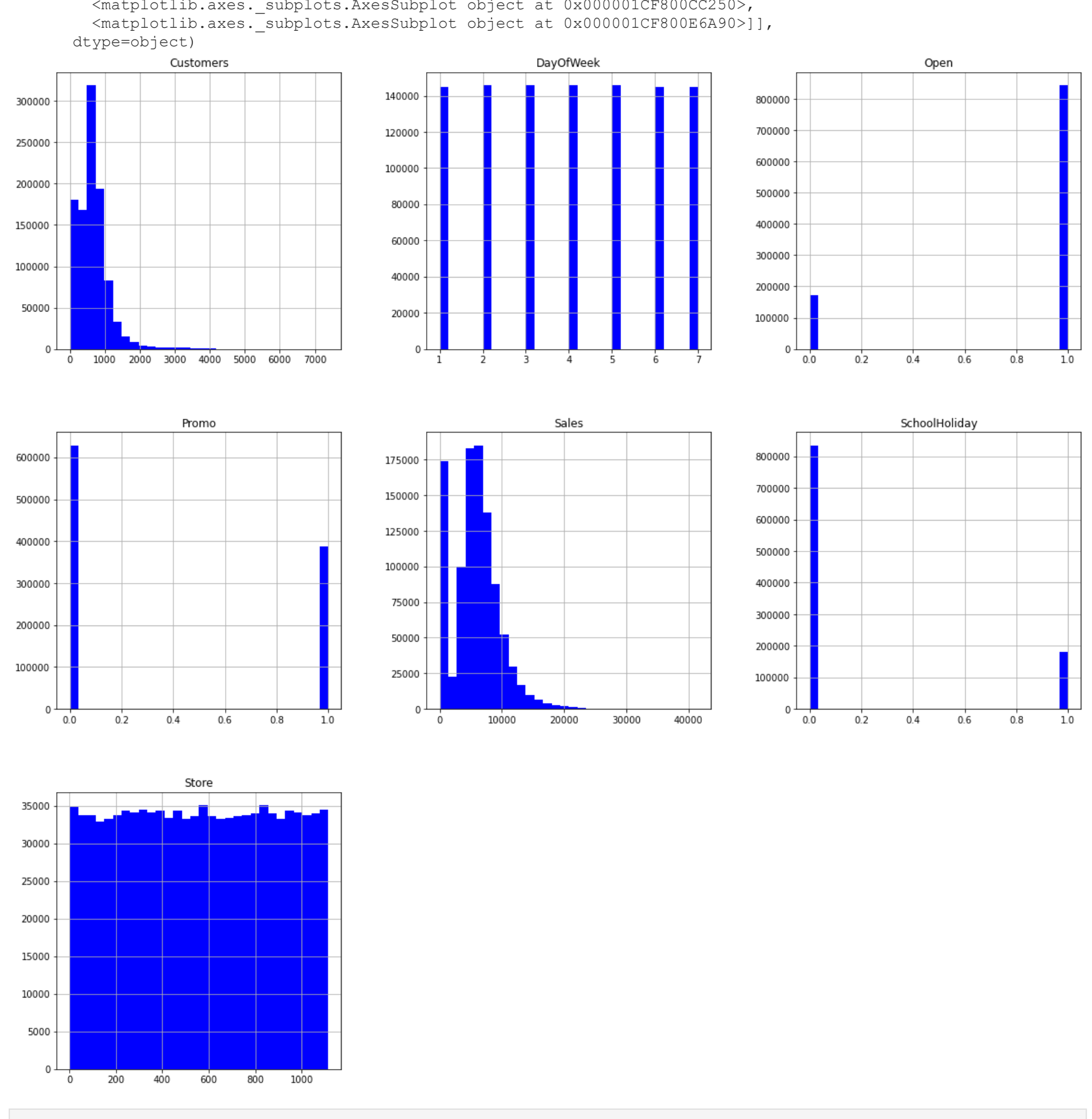
EXPLORAR EL DATASET DE VENTAS

In [70]:
```python
# Verificamos si nos faltan datos
sns.heatmap(sales_train_df.isnull(), yticklabels=False, cbar = False, cmap = "Blues")
# Comprobamos que no tenemos datos nulos
```

Out[70]: `<matplotlib.axes._subplots.AxesSubplot at 0x1cfaff17b20>`

In [71]:
```python
sales_train_df.hist(bins = 30, figsize=(20,20), color = 'b')
# Promedio de 600 clientes por día, el máximo es 4500 (¡tenga en cuenta que no podemos ver el valor atípico en
# Los datos se distribuyen por igual en varios días de la semana (~ 150000 observaciones x 7 días = ~ 1,1 millo
# Las tiendas están abiertas ~ 80% del tiempo
# Los datos se distribuyen por igual entre todas las tiendas (sin sesgo)
# La promoción # 1 se ejecutó aproximadamente el 40% del tiempo
# Ventas promedio alrededor de 5000-6000 Euros
# Las vacaciones escolares duran alrededor del 18% del tiempo
```

Out[71]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001CFC1134A00>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CFC11654F0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CFC17E2BE0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001CF800153A0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CF8003AB20>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CF8006D220>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001CF8006D310>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CF800CC250>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CF800E6A90>]],
      dtype=object)
```



In [72]:
```python
sales_train_df["Customers"].max()
```

Out[72]: 7388

In [73]:
```python
# Comprobamos cuántas tiendas están abiertas y cerradas!
closed_train_df = sales_train_df[sales_train_df['Open'] == 0]
open_train_df   = sales_train_df[sales_train_df['Open'] == 1]
```

In [74]:
```python
# Se contabilizan el número de tiendas que están abiertas y cerradas

print("Total = {} ".format(len(sales_train_df)))
print("Número de tiendas abiertas = {}".format(len(open_train_df)))
print("Número de tiendas cerradas = {}".format(len(closed_train_df)))
print("Porcentaje de tiendas cerradas = {}%".format(100.0*len(closed_train_df)/len(sales_train_df)))
```

```
Total = 1017209
Número de tiendas abiertas = 844392
Número de tiendas cerradas = 172817
Porcentaje de tiendas cerradas = 16.98933060954042%
```

In [75]:
```python
# Nos quedamos solo con las tiendas abiertas y eliminamos las tiendas cerradas
sales_train_df = sales_train_df[sales_train_df['Open'] == 1]
```

In [76]:
```python
sales_train_df
```

Out[76]:

	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
...	...	...	...	...	...	...	...	...	...
1016776	682	2	2013-01-01	3375	566	1	0	a	1
1016827	733	2	2013-01-01	10765	2377	1	0	a	1
1016863	769	2	2013-01-01	5035	1248	1	0	a	1
1017042	948	2	2013-01-01	4491	1039	1	0	a	1
1017190	1097	2	2013-01-01	5961	1405	1	0	a	1

844392 rows × 9 columns

In [77]:
```python
# Eliminamos la columna open ya que ahora no tiene sentido
sales_train_df.drop(['Open'], axis = 1, inplace = True)
```

In [78]:
```python
sales_train_df
```

Out[78]:

	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
...	...	...	...	...	...	...	...	...
1016776	682	2	2013-01-01	3375	566	0	a	1
1016827	733	2	2013-01-01	10765	2377	0	a	1
1016863	769	2	2013-01-01	5035	1248	0	a	1
1017042	948	2	2013-01-01	4491	1039	0	a	1
1017190	1097	2	2013-01-01	5961	1405	0	a	1

844392 rows × 8 columns

In [79]:
```python
# Ventas promedio = 6955 Euros, número promedio de clientes = 762 (ha subido)
sales_train_df.describe()
```

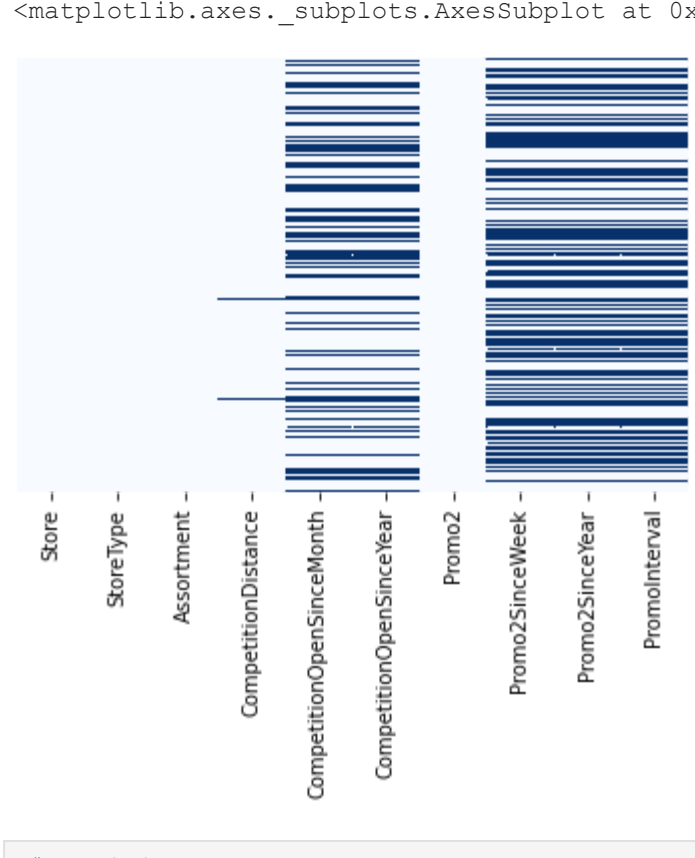
Out[79]:

	Store	DayOfWeek	Sales	Customers	Promo	SchoolHoliday
count	844392.000000	844392.000000	844392.000000	844392.000000	844392.000000	844392.000000
mean	558.422920	3.520361	6955.514291	762.728395	0.446352	0.193580
std	321.731914	1.723689	3104.214680	401.227674	0.497114	0.395103
min	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	280.000000	2.000000	4859.000000	519.000000	0.000000	0.000000
50%	558.000000	3.000000	6369.000000	676.000000	0.000000	0.000000
75%	837.000000	5.000000	8360.000000	893.000000	1.000000	0.000000
max	1115.000000	7.000000	41551.000000	7388.000000	1.000000	1.000000

EXPLORAR LOS DATOS DE LA INFORMACIÓN DE LAS TIENDAS

In [80]:
```python
# Comprobamos si falta algún dato en el data frame de información de la tienda
sns.heatmap(store_info_df.isnull(), yticklabels=False, cbar=False, cmap = "Blues")
```

Out[80]: `<matplotlib.axes._subplots.AxesSubplot at 0x1cf80b1c670>`



In [81]:
```python
# Verificamos los valores faltantes en la 'CompetitionDistance'
# Solo faltan 3 filas
store_info_df[store_info_df['CompetitionDistance'].isnull()]
```

Out[81]:

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek
290	291	d	a	NaN	NaN	NaN	0	NaN
621	622	a	c	NaN	NaN	NaN	0	NaN
878	879	d	a	NaN	NaN	NaN	1	5.0

In [82]:
```python
store_info_df[store_info_df['CompetitionOpenSinceMonth'].isnull()]
# Verificamos los valores faltantes en el 'CompetitionOpenSinceMonth'
# Faltan muchas filas = 354 (casi un tercio de las 1115 tiendas)
```

Out[82]:

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek
11	12	a	c	1070.0	NaN	NaN	1	13.0
12	13	d	a	310.0	NaN	NaN	1	45.0
15	16	a	c	3270.0	NaN	NaN	0	NaN
18	19	a	c	3240.0	NaN	NaN	1	22.0
21	22	a	a	1040.0	NaN	NaN	1	22.0
...	...	...	...	...	...	...	...	...
1095	1096	a	c	1130.0	NaN	NaN	1	10.0
1099	1100	a	a	540.0	NaN	NaN	1	14.0
1112	1113	a	c	9260.0	NaN	NaN	0	NaN
1113	1114	a	c	870.0	NaN	NaN	0	NaN
1114	1115	d	c	5350.0	NaN	NaN	1	22.0

354 rows × 10 columns

In [83]:
```python
store_info_df[store_info_df['Promo2'] == 0]
```

Out[83]:

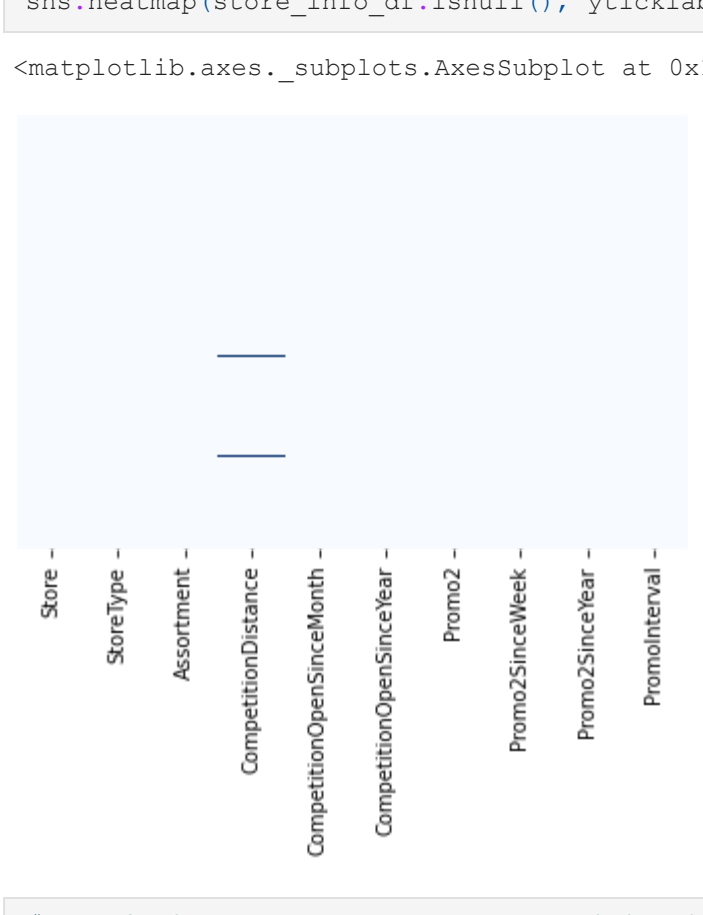
	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek
0	1	c	a	1270.0	9.0	2008.0	0	NaN
3	4	c	c	620.0	9.0	2009.0	0	NaN
4	5	a	a	29910.0	4.0	2015.0	0	NaN
5	6	a	a	310.0	12.0	2013.0	0	NaN
6	7	a	c	24000.0	4.0	2013.0	0	NaN
...	...	...	...	...	...	...	...	...
1107	1108	a	a	540.0	4.0	2004.0	0	NaN
1109	1110	c	c	900.0	9.0	2010.0	0	NaN
1111	1112	c	c	1880.0	4.0	2006.0	0	NaN
1112	1113	a	c	9260.0	NaN	NaN	0	NaN
1113	1114	a	c	870.0	NaN	NaN	0	NaN

544 rows × 10 columns

In [84]:
```python
# Parece que si 'promo2' es cero, 'promo2SinceWeek', 'Promo2SinceYear' y la información de 'PromoInterval' se e
# Hay 354 filas donde 'CompetitionOpenSinceYear' y 'CompetitionOpenSinceMonth' faltan valores
# Establecemos estos valores en ceros
str_cols = ['Promo2SinceWeek', 'Promo2SinceYear', 'PromoInterval', 'CompetitionOpenSinceYear', 'CompetitionOpen

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

In [85]:
```python
sns.heatmap(store_info_df.isnull(), yticklabels=False, cbar=False, cmap = "Blues")
```

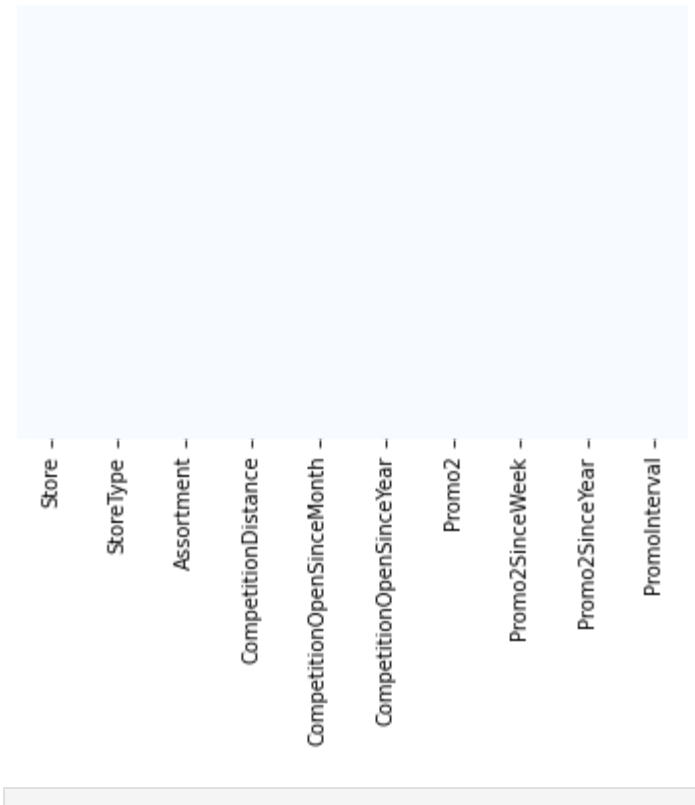
Out[85]: `<matplotlib.axes._subplots.AxesSubplot at 0x1cfaeb46c70>`



In [86]:
```python
# Hay 3 filas con valores de 'CompetitionDistance' que faltan, llenémoslas con valores promedio de la columna
store_info_df['CompetitionDistance'].fillna(store_info_df['CompetitionDistance'].mean(), inplace=True)
```

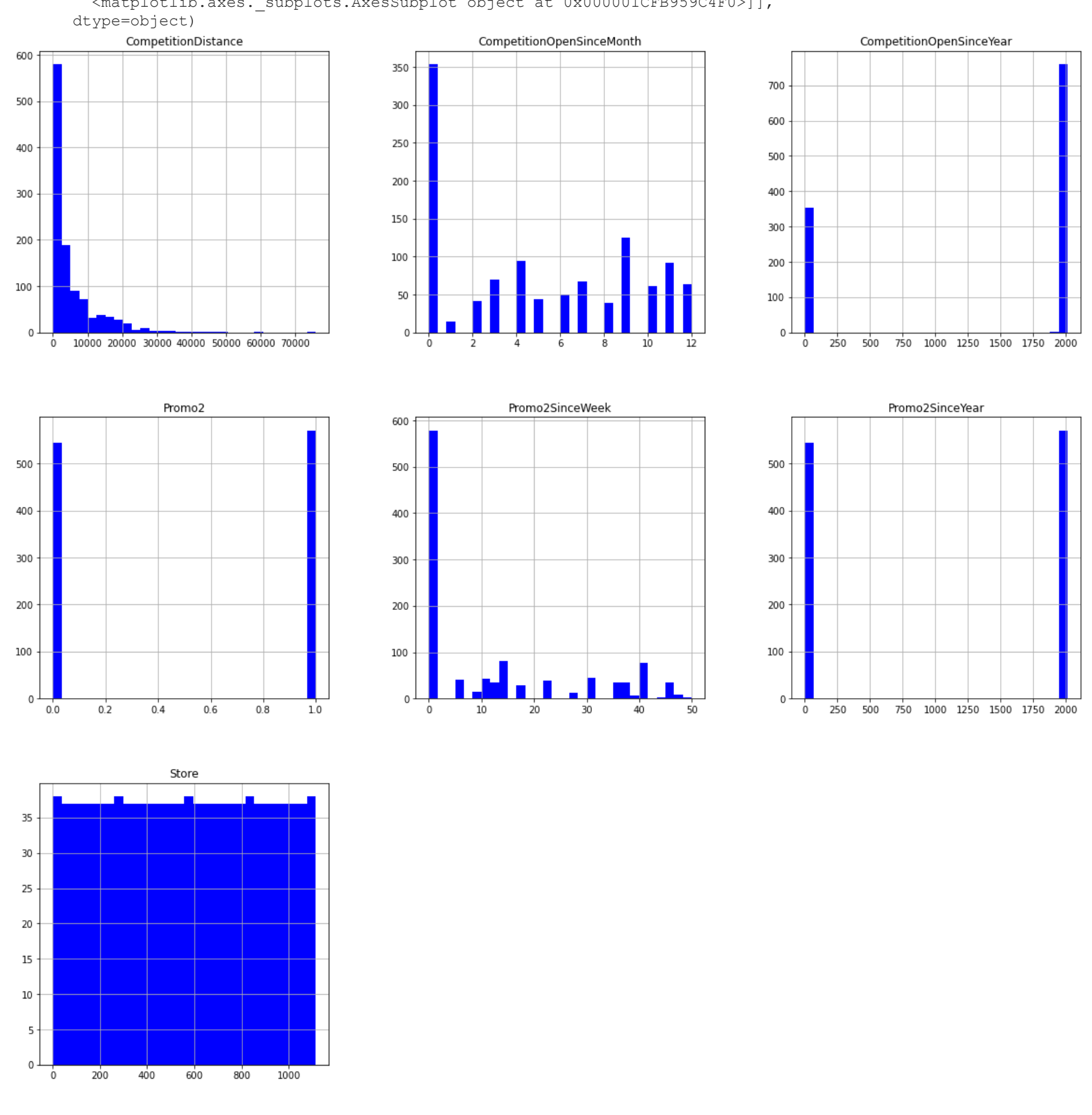
In [87]:
```python
sns.heatmap(store_info_df.isnull(), yticklabels=False, cbar=False, cmap = "Blues")
```

Out[87]: `<matplotlib.axes._subplots.AxesSubplot at 0x1cfafac7d00>`



In [88]:
```python
# la mitad de las tiendas participan en la promoción 2
# la mitad de las tiendas tienen su competencia a una distancia de 0-3000 m (3 kms de distancia)
store_info_df.hist(bins = 30, figsize=(20,20), color = 'b')
```

Out[88]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001CFAFAC7A60>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CFB9559B50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CFAFAB40A0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001CFB957AFA0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CFAFABB3A0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CFAF5EE5B0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001CFAFAC64C0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CFB94A71C0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001CFB959C4F0>]],
      dtype=object)
```



EXPLOREMOS EL DATASET COMBINADO

In [89]:
```python
# Combinemos ambos data frames en función de 'store'
sales_train_all_df = pd.merge(sales_train_df, store_info_df, how = 'inner', on = 'Store')
```

In [90]:
```python
sales_train_all_df.to_csv('test.csv', index = False)
```

In [91]:
```python
sales_train_all_df
```

Out[91]:

	Store	DayOfWeek	Date	Sales	Customers	Promo	StateHoliday	SchoolHoliday	StoreType	Assortment	CompetitionDistance
0	1	5	2015-07-31	5263	555	1	0	1	c	a	1270.0
1	1	4	2015-07-30	5020	546	1	0	1	c	a	1270.0
2	1	3	2015-07-29	4782	523	1	0	1	c	a	1270.0
3	1	2	2015-07-28	5011	560	1	0	1	c	a	1270.0
4	1	1	2015-07-27	6102	612	1	0	1	c	a	1270.0
...	...	...	...	...	...	...	...	...	...	...	...
844387	292	1	2013-01-07	9291	1002	1	0	0	a	a	1100.0
844388	292	6	2013-01-05	2748	340	0	0	0	a	a	1100.0
844389	292	5	2013-01-04	4202	560	0	0	1	a	a	1100.0
844390	292	4	2013-01-03	4580	662	0	0	1	a	a	1100.0
844391	292	3	2013-01-02	5076	672	0	0	1	a	a	1100.0

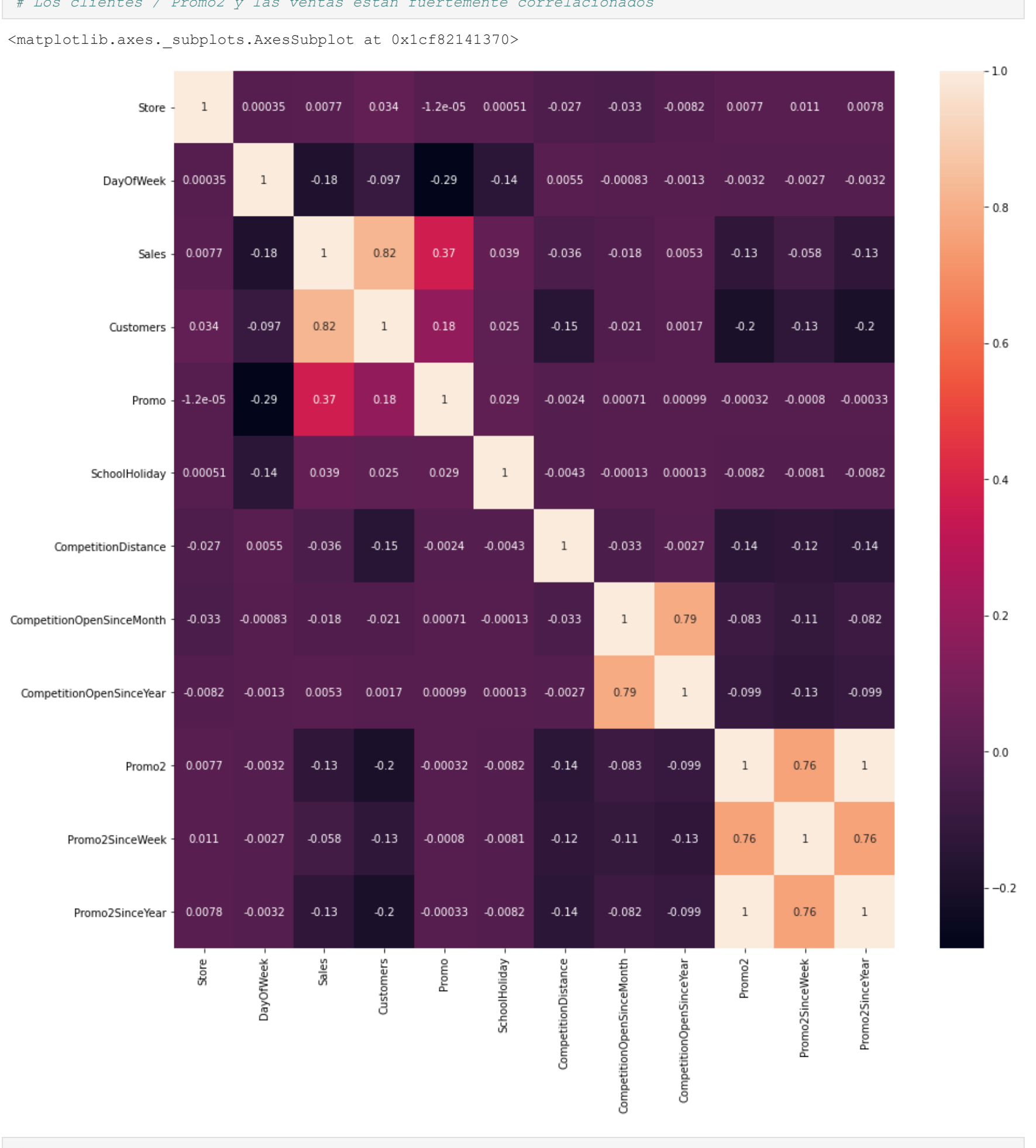
844392 rows × 17 columns

In [92]:
```python
correlations = sales_train_all_df.corr()['Sales'].sort_values()
correlations
# los clientes y la promoción se correlacionan positivamente con las ventas
# Promo2 no parece ser efectivo en absoluto
```

Out[92]:
```
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 [93]:
```python
correlations = sales_train_all_df.corr()
f, ax = plt.subplots(figsize = (15,15))
sns.heatmap(correlations, annot=True)
# Los clientes / Promo2 y las ventas están fuertemente correlacionados
```

Out[93]: `<matplotlib.axes._subplots.AxesSubplot at 0x1cf82141370>`



In [94]:
```python
# Separamos el año y lo ponemos en una columna separada
sales_train_all_df['Year'] = pd.DatetimeIndex(sales_train_all_df['Date']).year
```

In [95]:
```python
sales_train_all_df
```

Out[95]:

	Store	DayOfWeek	Date	Sales	Customers	Promo	StateHoliday	SchoolHoliday	StoreType	Assortment	CompetitionDistance
0	1	5	2015-07-31	5263	555	1	0	1	c	a	1270.0
1	1	4	2015-07-30	5020	546	1	0	1	c	a	1270.0
2	1	3	2015-07-29	4782	523	1	0	1	c	a	1270.0
3	1	2	2015-07-28	5011	560	1	0	1	c	a	1270.0
4	1	1	2015-07-27	6102	612	1	0	1	c	a	1270.0
...	...	...	...	...	...	...	...	...	...	...	...
844387	292	1	2013-01-07	9291	1002	1	0	0	a	a	1100.0
844388	292	6	2013-01-05	2748	340	0	0	0	a	a	1100.0
844389	292	5	2013-01-04	4202	560	0	0	1	a	a	1100.0
844390	292	4	2013-01-03	4580	662	0	0	1	a	a	1100.0
844391	292	3	2013-01-02	5076	672	0	0	1	a	a	1100.0

844392 rows × 18 columns

In [96]:
```python
# Hacemos lo mismo para el día y el mes
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
```

In [97]:
```python
sales_train_all_df
```

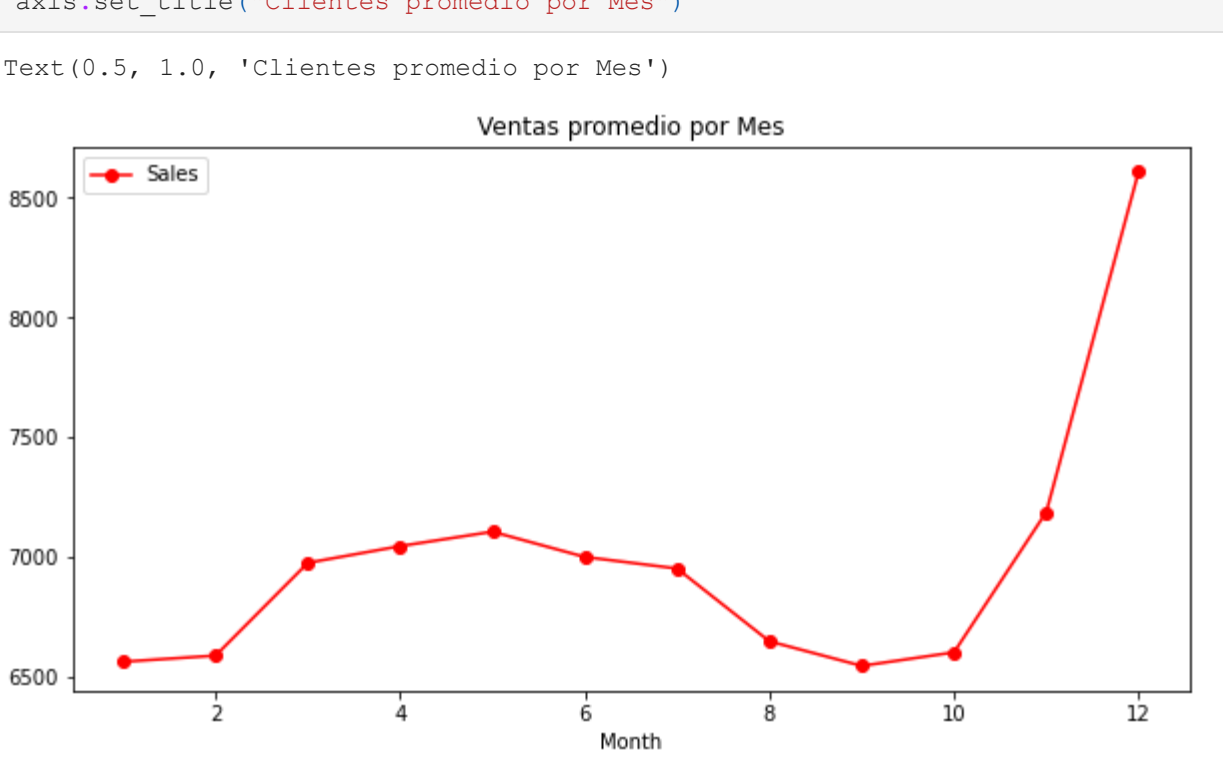
Out[97]:

	Store	DayOfWeek	Date	Sales	Customers	Promo	StateHoliday	SchoolHoliday	StoreType	Assortment	CompetitionDistance
0	1	5	2015-07-31	5263	555	1	0	1	c	a	1270.0
1	1	4	2015-07-30	5020	546	1	0	1	c	a	1270.0
2	1	3	2015-07-29	4782	523	1	0	1	c	a	1270.0
3	1	2	2015-07-28	5011	560	1	0	1	c	a	1270.0
4	1	1	2015-07-27	6102	612	1	0	1	c	a	1270.0
...	...	...	...	...	...	...	...	...	...	...	...
844387	292	1	2013-01-07	9291	1002	1	0	0	a	a	1100.0
844388	292	6	2013-01-05	2748	340	0	0	0	a	a	1100.0
844389	292	5	2013-01-04	4202	560	0	0	1	a	a	1100.0
844390	292	4	2013-01-03	4580	662	0	0	1	a	a	1100.0
844391	292	3	2013-01-02	5076	672	0	0	1	a	a	1100.0

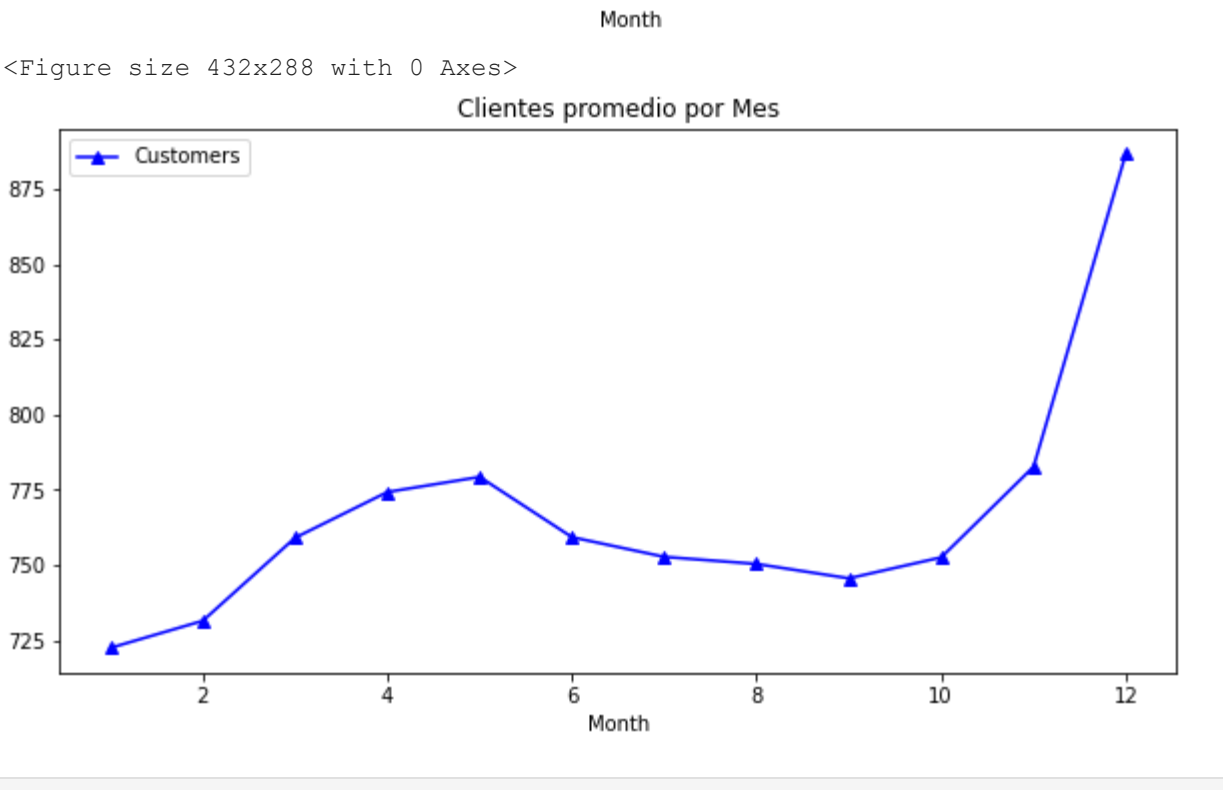
844392 rows × 20 columns

In [98]:
```python
# Echamos un vistazo a las ventas promedio y la cantidad de clientes por mes.
# Parece que las ventas y el número de clientes alcanzan su punto máximo en el período de Navidad

axis = sales_train_all_df.groupby('Month')[['Sales']].mean().plot(figsize = (10, 5), marker = 'o', color = 'r')
axis.set_title("Ventas promedio por Mes")


plt.figure()
axis = sales_train_all_df.groupby('Month')[['Customers']].mean().plot(figsize = (10, 5), marker = '^', color =
axis.set_title("Clientes promedio por Mes")
```

Out[98]: Text(0.5, 1.0, 'Clientes promedio por Mes')



`<Figure size 432x288 with 0 Axes>`



In [99]:
```python
# Echamos un vistazo a las ventas y a los clientes por día del mes.
# El número mínimo de clientes suele rondar el día 24 del mes.
# La mayoría de los clientes y las ventas son alrededor del 30 y el 1 del mes
axis = sales_train_all_df.groupby('Day')[['Sales']].mean().plot(figsize = (10, 5), marker = 'o', color = 'r')
axis.set_title("Ventas promedio por Día")


plt.figure()
axis = sales_train_all_df.groupby('Day')[['Customers']].mean().plot(figsize = (10, 5), marker = '^', color = 'b
axis.set_title("Clientes promedio por Día")
```