In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
import statsmodels.stats.api as sms
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.compat import lzip
```

In [4]:
```
df=pd.read_csv("Walmart(1).csv")
df
```

Out[4]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106
...	...	...	...	...	...	...	...	...
6430	45	28-09-2012	713173.95	0	64.88	3.997	192.013558	8.684
6431	45	05-10-2012	733455.07	0	64.89	3.985	192.170412	8.667
6432	45	12-10-2012	734464.36	0	54.47	4.000	192.327265	8.667
6433	45	19-10-2012	718125.53	0	56.47	3.969	192.330854	8.667
6434	45	26-10-2012	760281.43	0	58.85	3.882	192.308899	8.667

6435 rows × 8 columns

In [5]:
```
df.dtypes
```

Out[5]:
```
Store             int64
Date             object
Weekly_Sales    float64
Holiday_Flag      int64
Temperature     float64
Fuel_Price      float64
CPI             float64
Unemployment    float64
dtype: object
```

In [6]:
```
df.describe()
```

Out[6]:

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployme
count	6435.000000	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.000000	6435.0000
mean	23.000000	1.046965e+06	0.069930	60.663782	3.358607	171.578394	7.9991
std	12.988182	5.643666e+05	0.255049	18.444933	0.459020	39.356712	1.8758
min	1.000000	2.099862e+05	0.000000	-2.060000	2.472000	126.064000	3.8790
25%	12.000000	5.533501e+05	0.000000	47.460000	2.933000	131.735000	6.8910
50%	23.000000	9.607460e+05	0.000000	62.670000	3.445000	182.616521	7.8740
75%	34.000000	1.420159e+06	0.000000	74.940000	3.735000	212.743293	8.6220
max	45.000000	3.818686e+06	1.000000	100.140000	4.468000	227.232807	14.3130

In [7]:
```
df.isna().sum()
```

Out[7]:
```
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
dtype: int64
```

In [8]:
```
df.describe(include = 'object')
```

Out[8]:

	Date
count	6435
unique	143
top	05-02-2010
freq	45

In [9]:
```
var_cuantitativas = df.select_dtypes('number').columns
var_cualitativas  =df.select_dtypes('object').columns
```

In [10]:
```
df[var_cualitativas]
```

Out[10]:

	Date
0	05-02-2010
1	12-02-2010
2	19-02-2010
3	26-02-2010
4	05-03-2010
...	...
6430	28-09-2012
6431	05-10-2012
6432	12-10-2012
6433	19-10-2012
6434	26-10-2012

6435 rows × 1 columns

In [11]:
```
Q1_Weekly_Sales = df.Weekly_Sales.quantile(0.25)
Q3_Weekly_Sales = df.Weekly_Sales.quantile(0.75)
IQR_Weekly_Sales = Q3_Weekly_Sales - Q1_Weekly_Sales  #rango intercuartil
print(IQR_Weekly_Sales)
```

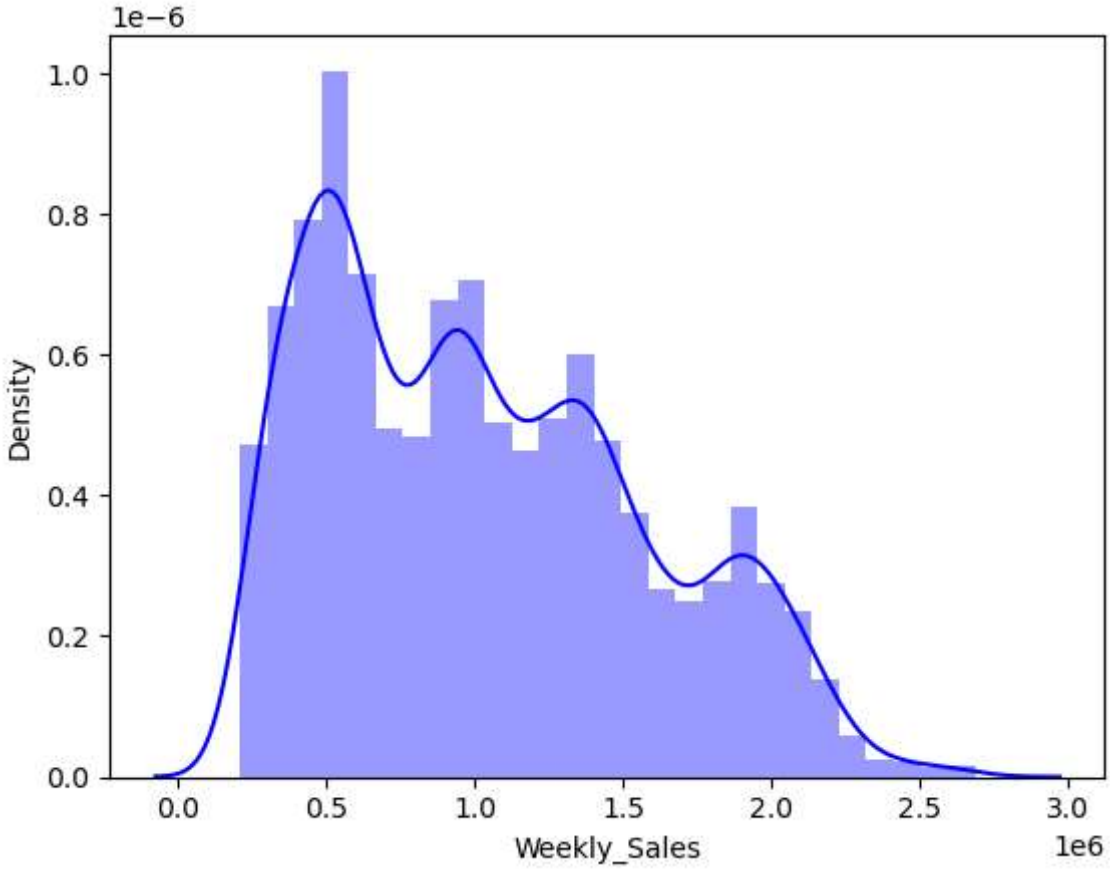
```
866808.5549999999
```

In [12]:
```
# Ahora removemos aquellas observaciones que se encuentran por fuera del rango: 1.5 x
df = df[~((df['Weekly_Sales'] < (Q1_Weekly_Sales - 1.5 * IQR_Weekly_Sales)) |(df['Week
df.shape
```

Out[12]:
```
(6401, 8)
```

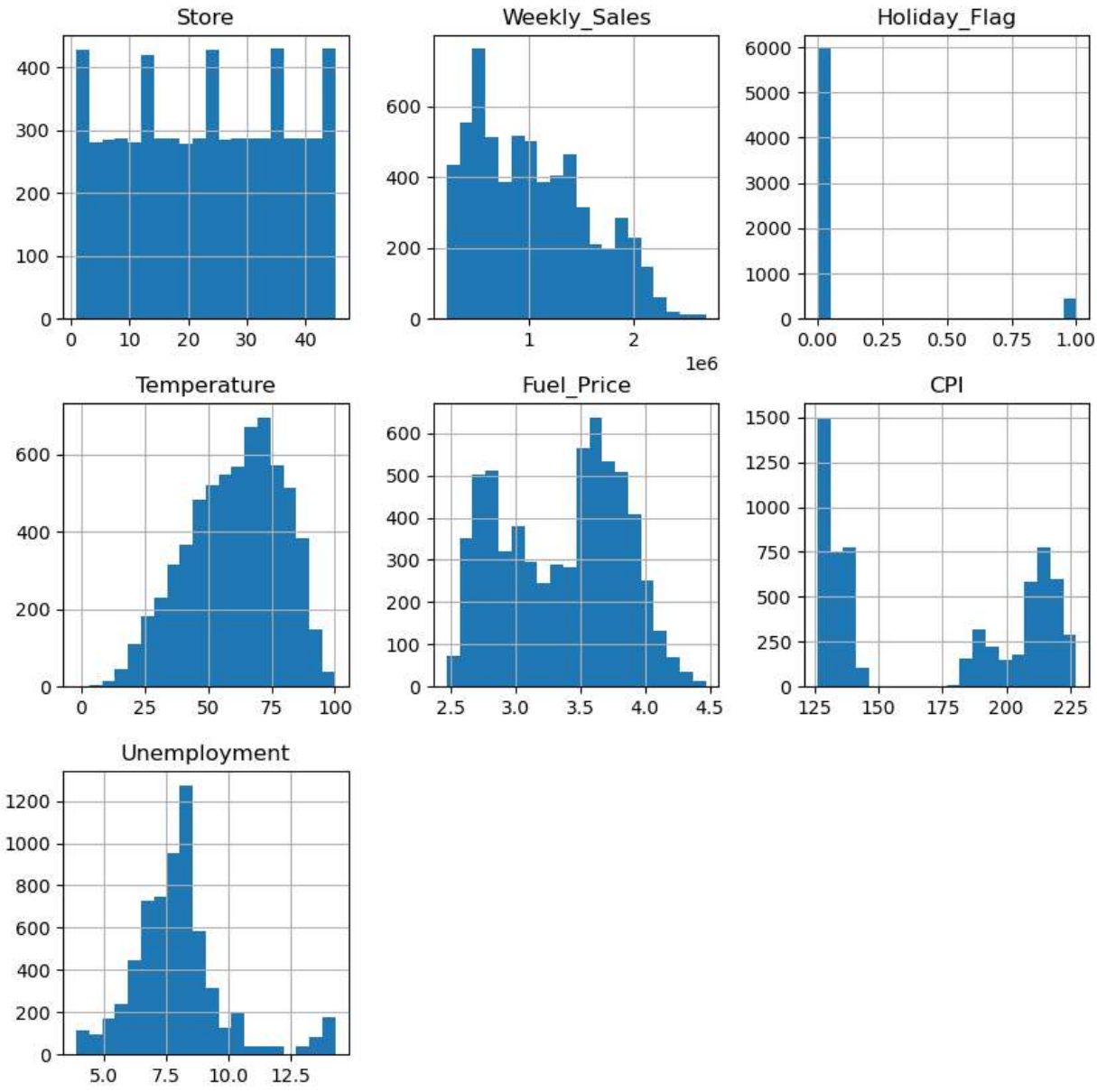
In [13]:
```
sns.distplot(df['Weekly_Sales'],color="blue")
```

Out[13]:
```
<Axes: xlabel='Weekly_Sales', ylabel='Density'>
```



In [14]:
```
df[var_cuantitativas].hist(bins = 20, figsize = (10,10))
;
```

Out[14]:
```
''
```



In [15]:
```
Q1_Store = df.Store.quantile(0.25)
Q3_Store = df.Store.quantile(0.75)
IQR_Store= Q3_Store- Q1_Store #rango intercuartil
print(IQR_Store)
```

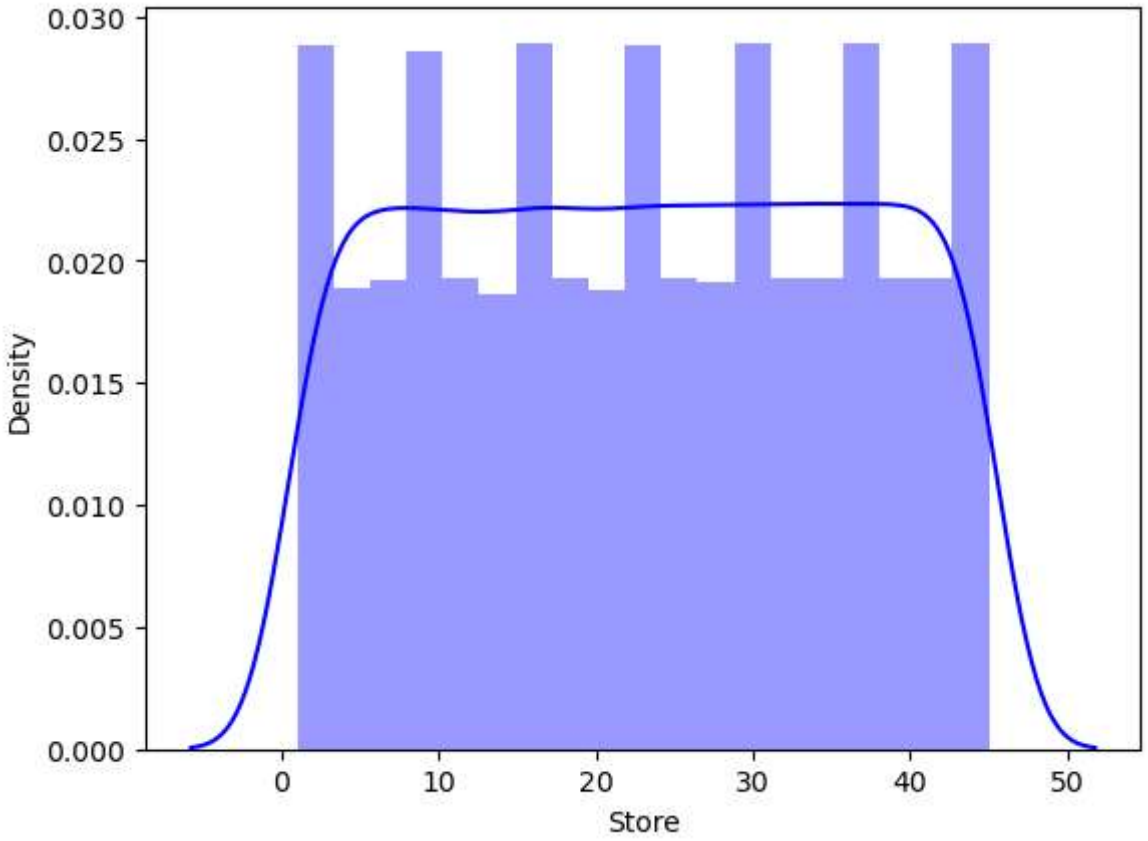
```
22.0
```

In [16]:
```
df = df[~((df['Store'] < (Q1_Store - 1.5 * IQR_Store)) |(df['Store'] > (Q3_Store + 1.5
df.shape
```

Out[16]:
```
(6401, 8)
```

In [17]:
```
sns.distplot(df['Store'],color="blue")
```

Out[17]:
```
<Axes: xlabel='Store', ylabel='Density'>
```



In [18]:
```
df.corr().style.background_gradient(cmap='coolwarm')
```

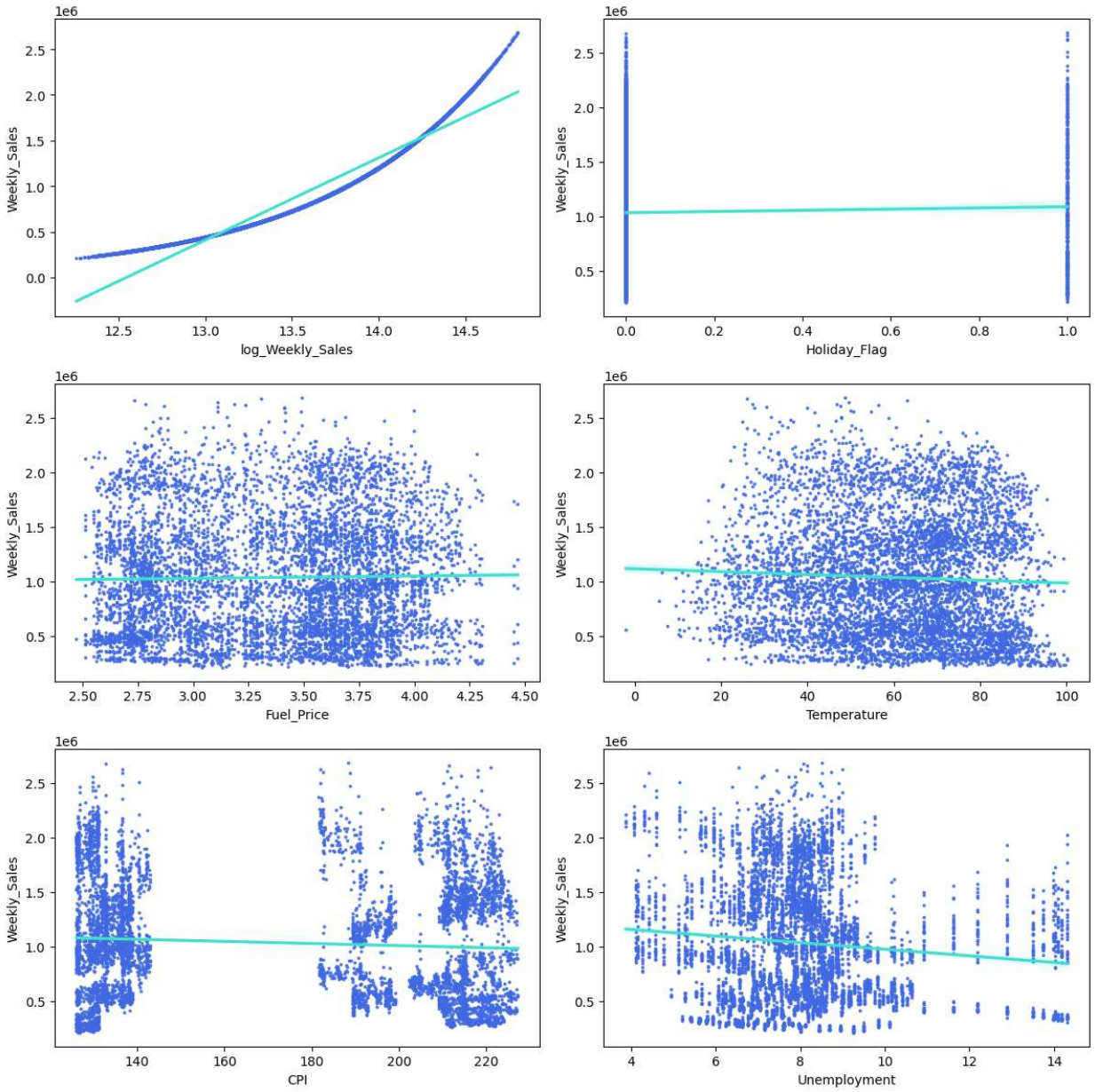
Out[18]:

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployı
Store	1.000000	-0.332881	0.003566	-0.026652	0.057863	-0.212481	0.22
Weekly_Sales	-0.332881	1.000000	0.025358	-0.044340	0.018189	-0.069617	-0.10
Holiday_Flag	0.003566	0.025358	1.000000	-0.154556	-0.077808	0.000121	0.01
Temperature	-0.026652	-0.044340	-0.154556	1.000000	0.143080	0.176510	0.09
Fuel_Price	0.057863	0.018189	-0.077808	0.143080	1.000000	-0.172078	-0.03
CPI	-0.212481	-0.069617	0.000121	0.176510	-0.172078	1.000000	-0.30
Unemployment	0.222746	-0.104298	0.012385	0.099266	-0.035469	-0.304158	1.00

In [19]:
```
log_Weekly_Sales=np.log(df.Weekly_Sales)
df['log_Weekly_Sales']=log_Weekly_Sales
```

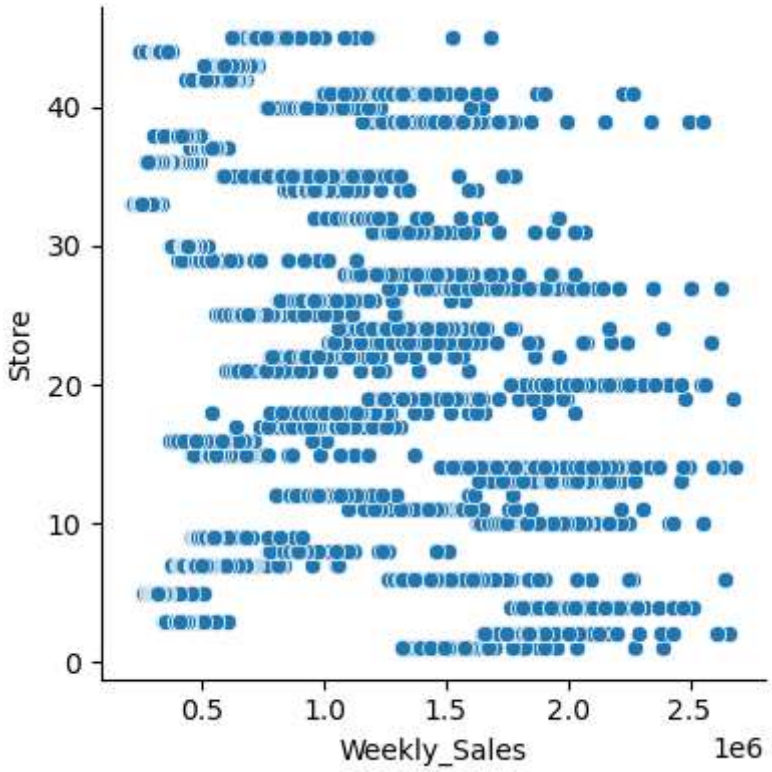
In [20]:
```
n = 7
fig = plt.figure(figsize=(12,12))
# Correlaciones en pares
corr = df.corr()
#
cols = corr.nlargest(7, "Weekly_Sales")["Weekly_Sales"].index
# Calculate correlation
for i in np.arange(1,7):
    regline = df[cols[i]]
    ax = fig.add_subplot(3,2,i)
    sns.regplot(x=regline, y=df['Weekly_Sales'], scatter_kws={"color": "royalblue", "s
```

```
                line_kws={"color": "turquoise"})
plt.tight_layout()
plt.show()
```



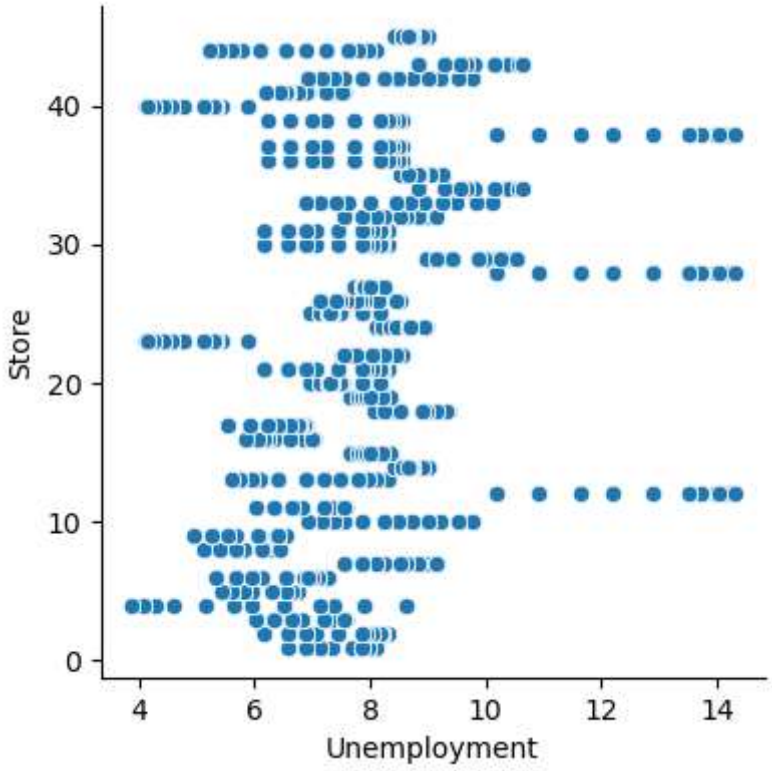
In [34]:
```
sns.pairplot(x_vars='Weekly_Sales', y_vars='Store' , data=df, size=4)
```

Out[34]:
```
<seaborn.axisgrid.PairGrid at 0x14d757e9300>
```



In [35]:
```
sns.pairplot(x_vars='Unemployment', y_vars='Store' , data=df, size=4)
```

Out[35]:
```
<seaborn.axisgrid.PairGrid at 0x14d71f35510>
```



In [55]:
```
from sklearn.linear_model import LinearRegression
```

In [56]:
```
var_cuantitativas=df.select_dtypes("number").columns
```

In [57]:
```
var_cualitativas=df.select_dtypes("object").columns
```

In [58]:
```
from sklearn.preprocessing import LabelEncoder
```

In [59]:
```
df[var_cualitativas]=df[var_cualitativas].apply(LabelEncoder().fit_transform)
df
```

Out[59]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	19	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	52	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	85	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	118	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	20	1554806.68	0	46.50	2.625	211.350143	8.106
...	...	...	...	...	...	...	...	...
6430	45	130	713173.95	0	64.88	3.997	192.013558	8.684
6431	45	22	733455.07	0	64.89	3.985	192.170412	8.667
6432	45	55	734464.36	0	54.47	4.000	192.327265	8.667
6433	45	88	718125.53	0	56.47	3.969	192.330854	8.667
6434	45	121	760281.43	0	58.85	3.882	192.308899	8.667

6401 rows × 9 columns

In [60]:
```
valores_perdidos = df.isnull().sum()
```

In [61]:
```
print(valores_perdidos)
```

```
Store               0
Date                0
Weekly_Sales        0
Holiday_Flag        0
Temperature         0
Fuel_Price          0
CPI                 0
Unemployment        0
log_Weekly_Sales    0
dtype: int64
```

In [62]:
```
x=df[df.columns.difference(["Weekly_Sales"])]
y=df.Weekly_Sales
```

In [63]:
```
from sklearn.model_selection import train_test_split
```

In [64]:
```
x_train, x_test,y_train, y_test = train_test_split(x,y, test_size=0.20, random_state=1
```

In [65]:
```
x_train.shape
```

Out[65]:
```
(5120, 8)
```

In [66]:
```
x_test.shape
```

Out[66]:
```
(1281, 8)
```

In [67]:
```
modelo_rl=LinearRegression()
```

In [68]:
```
modelo_rl.fit(x_train, y_train)
```

Out[68]:
```
▾ LinearRegression
LinearRegression()
```

In [69]:
```
pred_train = modelo_rl.predict(x_train)
pred_test = modelo_rl.predict(x_test)
```

In [70]:
```
len(pred_train)
```

Out[70]:
```
5120
```

In [71]:
```
len(pred_test)
```

Out[71]:
```
1281
```

In [72]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

In [73]:
```
mse_train = mean_squared_error(y_train, pred_train)
mse_test = mean_squared_error(y_test, pred_test)
```

In [74]:
```
print(mse_train, mse_test)
```

```
20460696480.133076 21018852382.353256
```

In [75]:
```
rmse_train = np.sqrt(mse_train)
rmse_test = np.sqrt(mse_test)
rmse_train
```

Out[75]:
```
143040.89093728786
```

In [76]:
```
mae_train = mean_absolute_error(y_train, pred_train)
mae_test = mean_absolute_error(y_test, pred_test)
mae_train
```

Out[76]:
```
113420.29196208843
```

In [77]:
```
from sklearn.metrics import r2_score
```

In [78]:
```
r2_train = r2_score(y_train, pred_train)
r2_test = r2_score(y_test, pred_test)
print(r2_train, r2_test)
```

```
0.9310061838120013 0.929872699173956
```

In [79]:
```
modelo_rl.intercept_
```

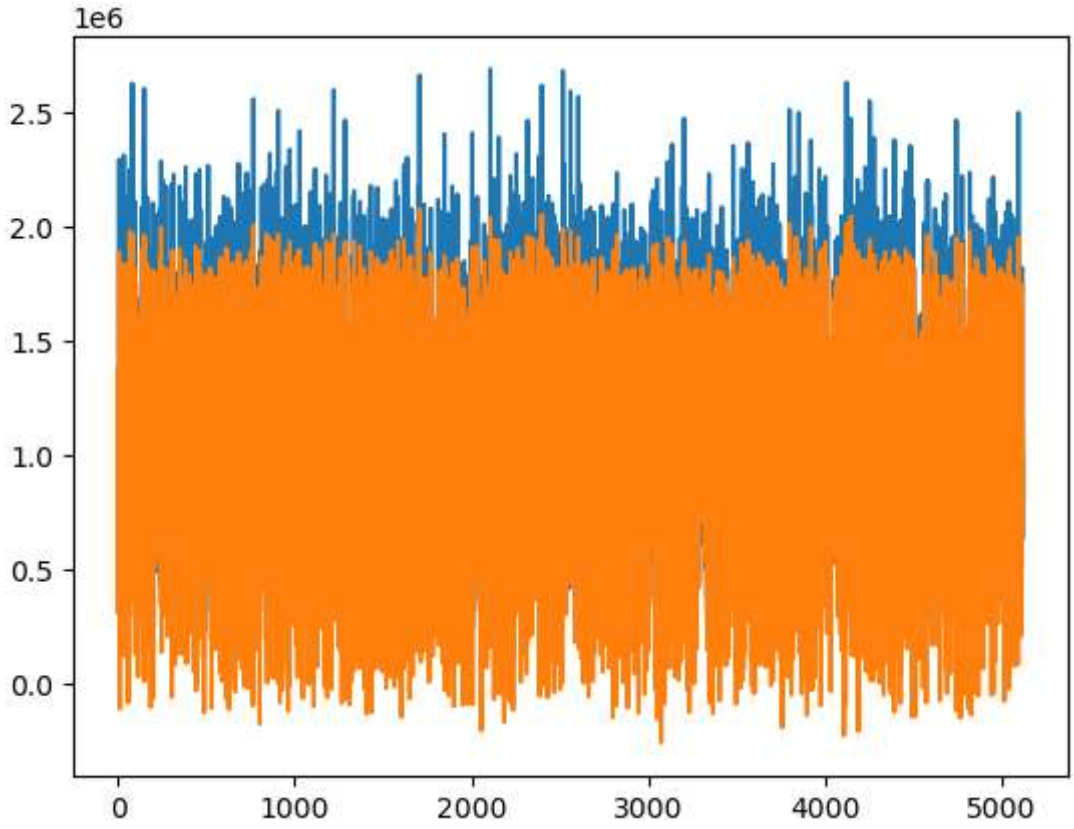
Out[79]:
```
-10888504.561193127
```

In [80]:
```
len(modelo_rl.coef_)
```

Out[80]:
```
8
```

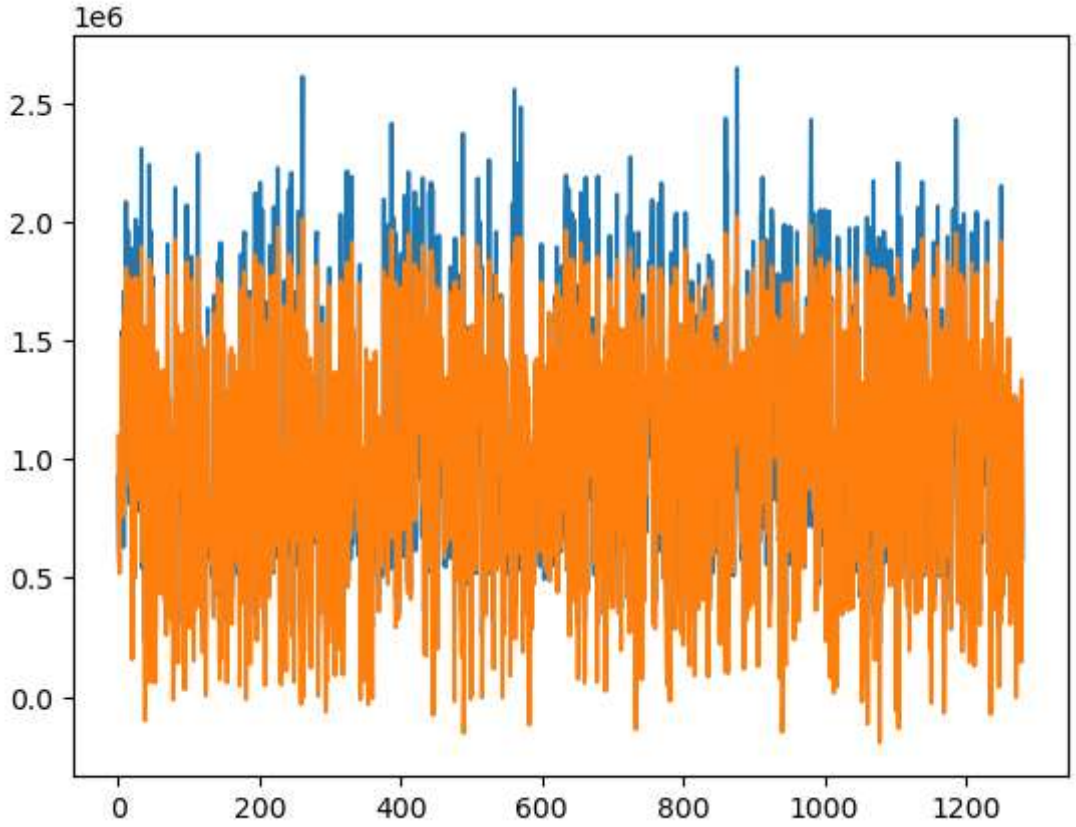
In [81]:
```
fig, ax =plt.subplots()
ax.plot(y_train.values)
ax.plot(pred_train)
```

Out[81]:
```
[<matplotlib.lines.Line2D at 0x14d76e0fbe0>]
```



In [82]:
```
fig, ax =plt.subplots()
ax.plot(y_test.values)
ax.plot(pred_test)
```

Out[82]:
```
[<matplotlib.lines.Line2D at 0x14d772e8c70>]
```



In [83]:
```
modelo_rl.coef_
```

Out[83]:
```
array([-4.35775683e+02, -6.75486914e+01, -1.12095662e+04,  3.27620050e+04,
       -2.46821790e+03,  1.32499874e+03, -9.06346224e+03,  8.82685401e+05])
```

In [84]:
```
# Feature importance
from sklearn.preprocessing import StandardScaler
```

In [85]:
```
sc=StandardScaler()
```

In [86]:
```
x_train_std= sc.fit_transform(x_train)
x_test_std = sc.transform(x_test)
```

In [87]:
```
x_train_std
```

Out[87]:
```
array([[ 0.50424771, -1.01320874, -1.03760735, ..., -2.73519483,
         0.43275132, -0.50535333],
       [ 1.29259603, -1.68818216,  0.313248  , ...,  1.03092491,
        -1.17184413, -1.55005023],
       [-1.06202421, -1.5194388 , -0.39600562, ..., -1.13095655,
         1.14011359,  0.18582415],
       ...,
       [-1.08246149,  0.11978236,  1.26037436, ...,  1.47660675,
         0.26043149,  1.23182257],
       [ 0.88615379,  0.21620713, -0.24542562, ..., -1.20289306,
        -0.35172938, -0.57568266],
       [-0.98364714, -1.1578459 , -0.9153975 , ..., -1.63072599,
         0.07694279,  0.15335303]])
```

In [88]:
```
modelo_rl_std=LinearRegression()
modelo_rl_std.fit(x_train_std, y_train)
```

Out[88]:
```
▾ LinearRegression
LinearRegression()
```

In [89]:
```
pred_train_std=modelo_rl_std.predict(x_train_std)
pred_test_std = modelo_rl_std.predict(x_test_std)
```

In [90]:
```
r2_train = r2_score(y_train, pred_train_std)
r2_test = r2_score (y_test, pred_test_std)
print(r2_train, r2_test)
```

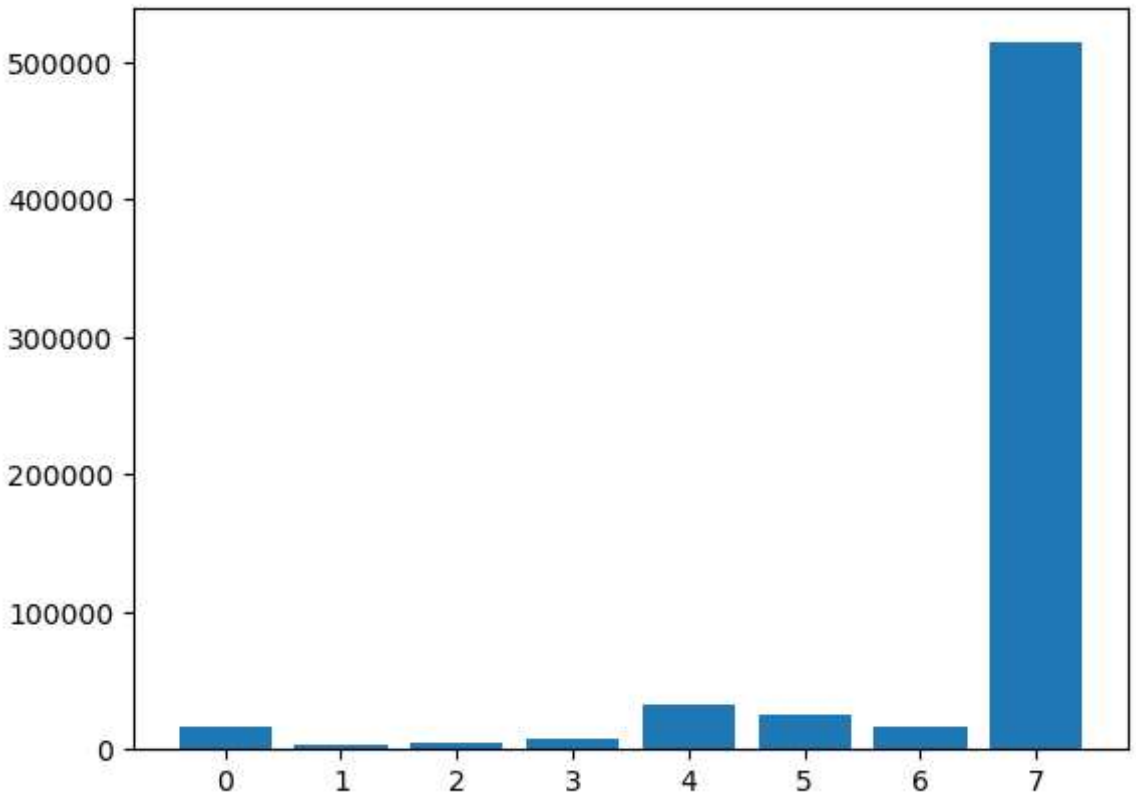
```
0.9310061838120013 0.929872699173956
```

In [91]:
```
# Importnacia de coeficientes

importancia=modelo_rl_std.coef_
```

In [92]:
```
# Graficar
plt.bar([x for x in range(len(importancia))], abs(importancia))
```

Out[92]:
```
<BarContainer object of 8 artists>
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: