Analyzing Walmart's Promotional Impact During Holidays

This study delves into the promotional strategies employed by Walmart during key holiday periods, investigating the impact of markdowns on sales performance. Through thorough analysis of historical data and promotional activities, the study aims to uncover insights into how these markdown events influence consumer behavior and overall sales trends during holiday seasons. By examining the effectiveness of promotional strategies, this research provides valuable insights for retailers seeking to optimize their marketing efforts and capitalize on holiday shopping trends.

Exploratory data analysis and preprocessing

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from datetime import datetime
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv('D:\\walmart-sales-dataset-of-45stores.csv')
```

In [3]:
```
df.head(10)
```

Out[3]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	5/2/2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12/2/2010	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	5/3/2010	1554806.68	0	46.50	2.625	211.350143	8.106
5	1	12/3/2010	1439541.59	0	57.79	2.667	211.380643	8.106
6	1	19-03-2010	1472515.79	0	54.58	2.720	211.215635	8.106
7	1	26-03-2010	1404429.92	0	51.45	2.732	211.018042	8.106
8	1	2/4/2010	1594968.28	0	62.27	2.719	210.820450	7.808
9	1	9/4/2010	1545418.53	0	65.86	2.770	210.622857	7.808

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Store         6435 non-null   int64  
 1   Date          6435 non-null   object 
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64  
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
dtypes: float64(5), int64(2), object(1)
memory usage: 402.3+ KB
```

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

Out[5]:
```
(6435, 8)
```

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

Out[6]:

	count	mean	std	min	25%	50%	75%	max
Store	6435.0	2.300000e+01	12.988182	1.000	12.000	23.000000	3.400000e+01	4.500000e+01
Weekly_Sales	6435.0	1.046965e+06	564366.622054	209986.250	553350.105	960746.040000	1.420159e+06	3.818686e+06
Holiday_Flag	6435.0	6.993007e-02	0.255049	0.000	0.000	0.000000	0.000000e+00	1.000000e+00
Temperature	6435.0	6.066378e+01	18.444933	-2.060	47.460	62.670000	7.494000e+01	1.001400e+02
Fuel_Price	6435.0	3.358607e+00	0.459020	2.472	2.933	3.445000	3.735000e+00	4.468000e+00
CPI	6435.0	1.715784e+02	39.356712	126.064	131.735	182.616521	2.127433e+02	2.272328e+02
Unemployment	6435.0	7.999151e+00	1.875885	3.879	6.891	7.874000	8.622000e+00	1.431300e+01

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Store', 'Date', 'Weekly_Sales', 'Holiday_Flag', 'Temperature',
       'Fuel_Price', 'CPI', 'Unemployment'],
      dtype='object')
```

In [8]:
```
df.nunique().to_frame()
```

Out[8]:

	0
Store	45
Date	143
Weekly_Sales	6435
Holiday_Flag	2
Temperature	3528
Fuel_Price	892
CPI	2145
Unemployment	349

In [9]:
```
df.isna().sum().to_frame()
```

Out[9]:

	0
Store	0
Date	0
Weekly_Sales	0
Holiday_Flag	0
Temperature	0
Fuel_Price	0
CPI	0
Unemployment	0

In [10]:
```
df.isna().sum().sum()
```

Out[10]:
```
0
```

In [11]:
```
df.duplicated().sum()
```

Out[11]:
```
0
```

In [12]:
```
def convert_dates(date_str):
    formats = ['%d/%m/%Y', '%d-%m-%Y']
    for fmt in formats:
        try:
            return pd.to_datetime(date_str, format=fmt)
        except ValueError:
            pass
    return pd.NaT

df['Date'] = df['Date'].apply(convert_dates)

df['Date']=pd.to_datetime(df['Date'],format='%d-%m-%Y')
df['Date'].to_frame().sample(5)
```

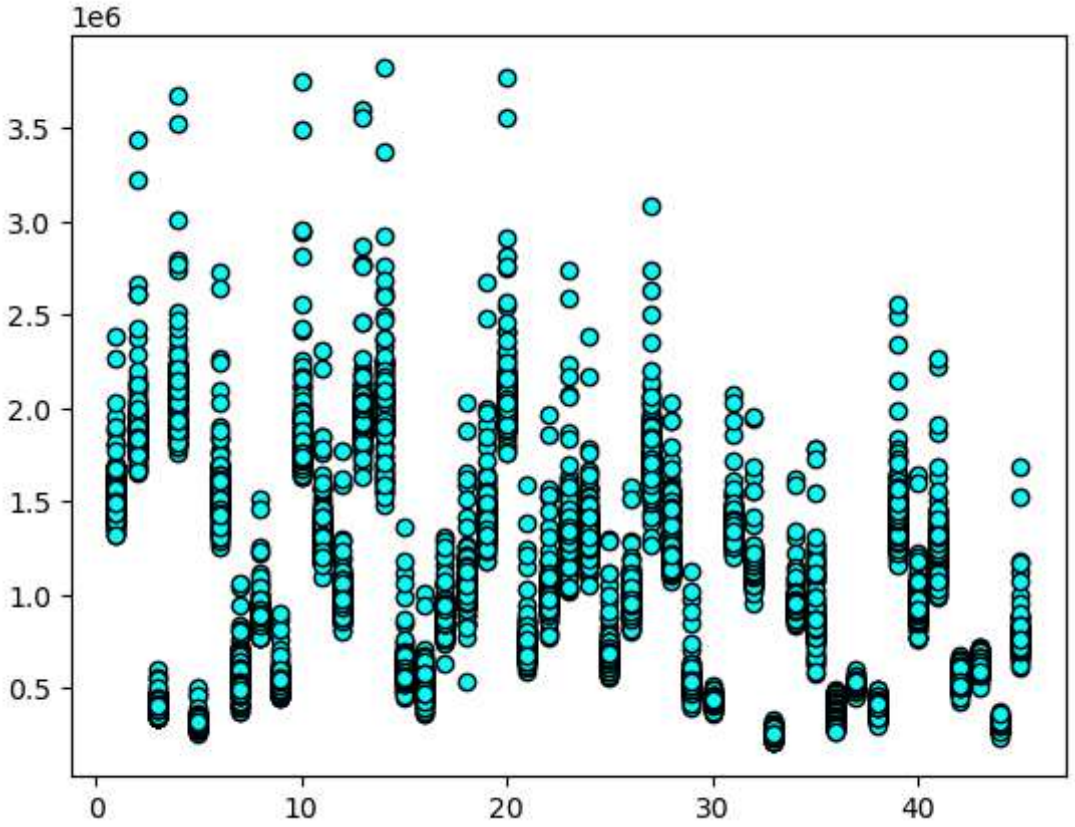
Out[12]:

	Date
6291	2012-10-26
1186	2010-11-26
6107	2012-01-13
384	2011-12-23
5871	2010-04-02

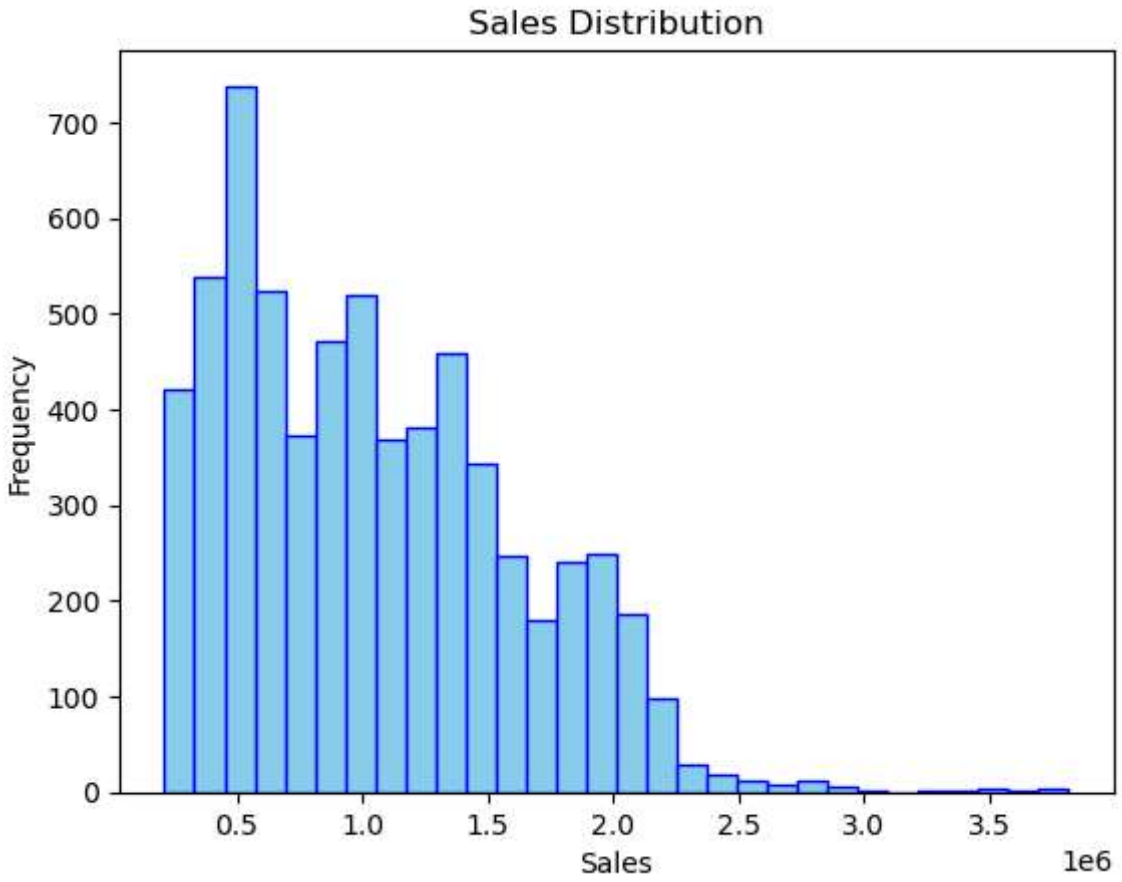
visualizations

In [13]:

```
plt.scatter(df["Store"],df["Weekly_Sales"],edgecolor='black',color='cyan')
plt.show()
```



In [14]:
```
plt.hist(df['Weekly_Sales'],bins=30,color='skyblue', edgecolor='blue')
plt.title('Sales Distribution')
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.show()
```



as we see the range of (weekly_sales) from 500000 to 2000000

Store with maximum sales

In [15]:
```
max_sales = df.groupby('Store')['Weekly_Sales'].sum().reset_index()
pd.options.display.float_format = '{:.0f}'.format
max_sales.sample(10)
```

Out[15]:

	Store	Weekly_Sales
1	2	275382441
33	34	138249763
44	45	112395341
13	14	288999911
6	7	81598275
39	40	137870310
0	1	222402809
40	41	181341935
38	39	207445542
32	33	37160222

In [16]:
```
ordered_max_sales = max_sales.sort_values(by='Weekly_Sales', ascending=False)
```

In [17]:
```
ordered_max_sales.sample(5)
```

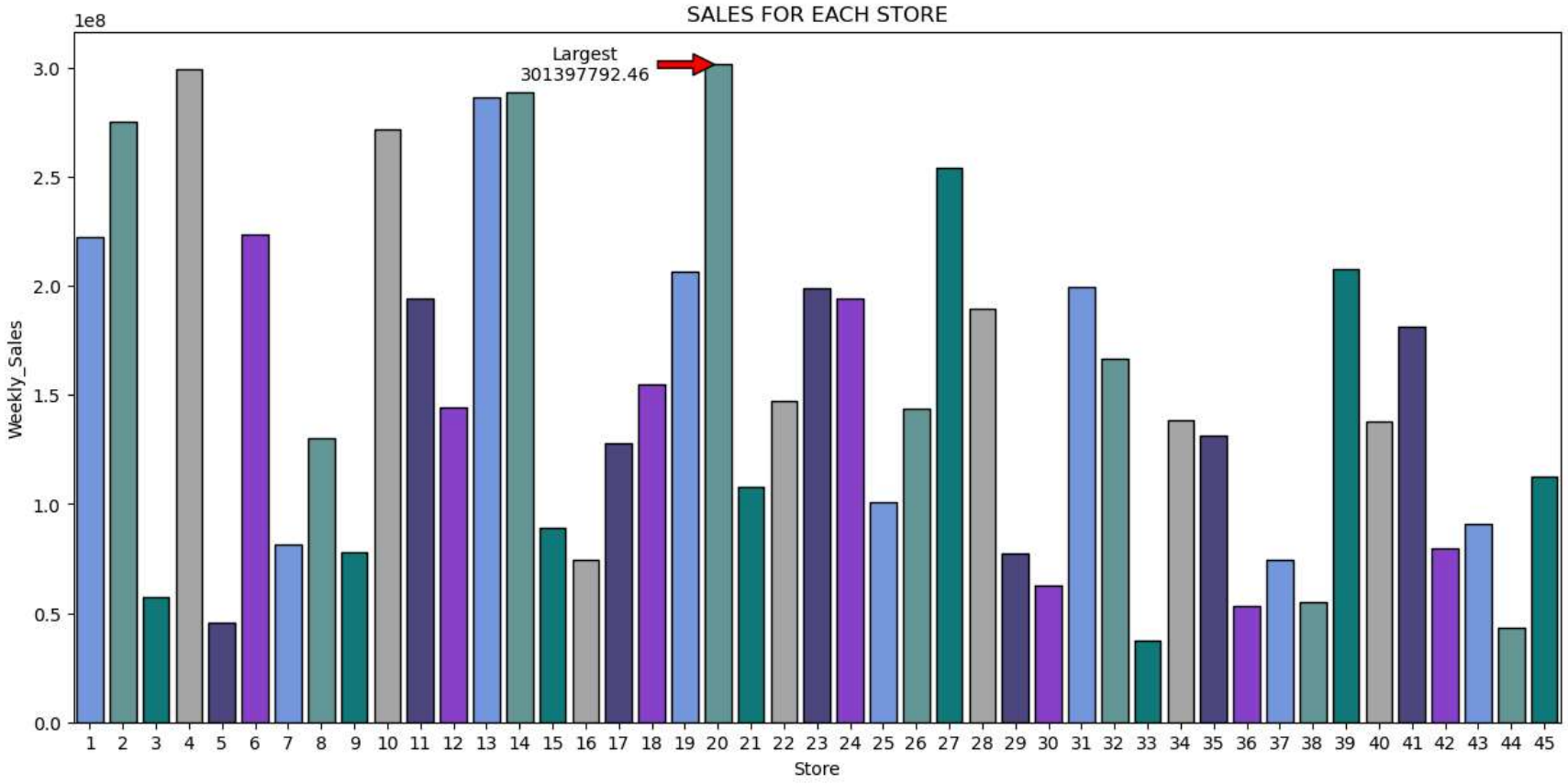
Out[17]:

	Store	Weekly_Sales
6	7	81598275
5	6	223756131
14	15	89133684
37	38	55159626
9	10	271617714

In [18]:
```
maximum_store = ordered_max_sales.iloc[0,0]
maximum_sales = ordered_max_sales.iloc[0,1]
print("STORE : ",maximum_store)
print("with maximum sales that equals : ",maximum_sales)
```

```
STORE :  20
with maximum sales that equals :  301397792.46
```

In [96]:
```
plt.figure(figsize=(15, 7))
colors = ["#6495ED", "#5F9EA0", "#008B8B", "#A9A9A9", "#483D8B", "#8A2BE2"]

bar = sns.barplot(x='Store',y='Weekly_Sales',data=ordered_max_sales,palette=colors ,edgecolor='black')

plt.annotate(f'Largest\n{maximum_sales}', xy=(maximum_store  - 1, maximum_sales ), xytext=(maximum_store - 5, maximum_sales  + 200),
             ha='center', va='center', color='black', arrowprops=dict(facecolor='red', shrink=0.05))

bar.set_title("SALES FOR EACH STORE")
plt.show()
```



we note that Store : *20* has the max sales which is : 301397792.46

Which store has maximum standard deviation i.e., the sales vary a lot

In [20]:
```
std = df.groupby('Store')['Weekly_Sales'].std().reset_index()
ordered_std = std.sort_values(by='Weekly_Sales',ascending=False)
ordered_std.rename(columns={'Weekly_Sales':'STD'},inplace = True)
ordered_std.sample(5)
```

Out[20]:

	Store	STD
28	29	99120
33	34	104630
36	37	21837
44	45	130169
43	44	24763

In [21]:
```
ordered_std['STD'].max()
```

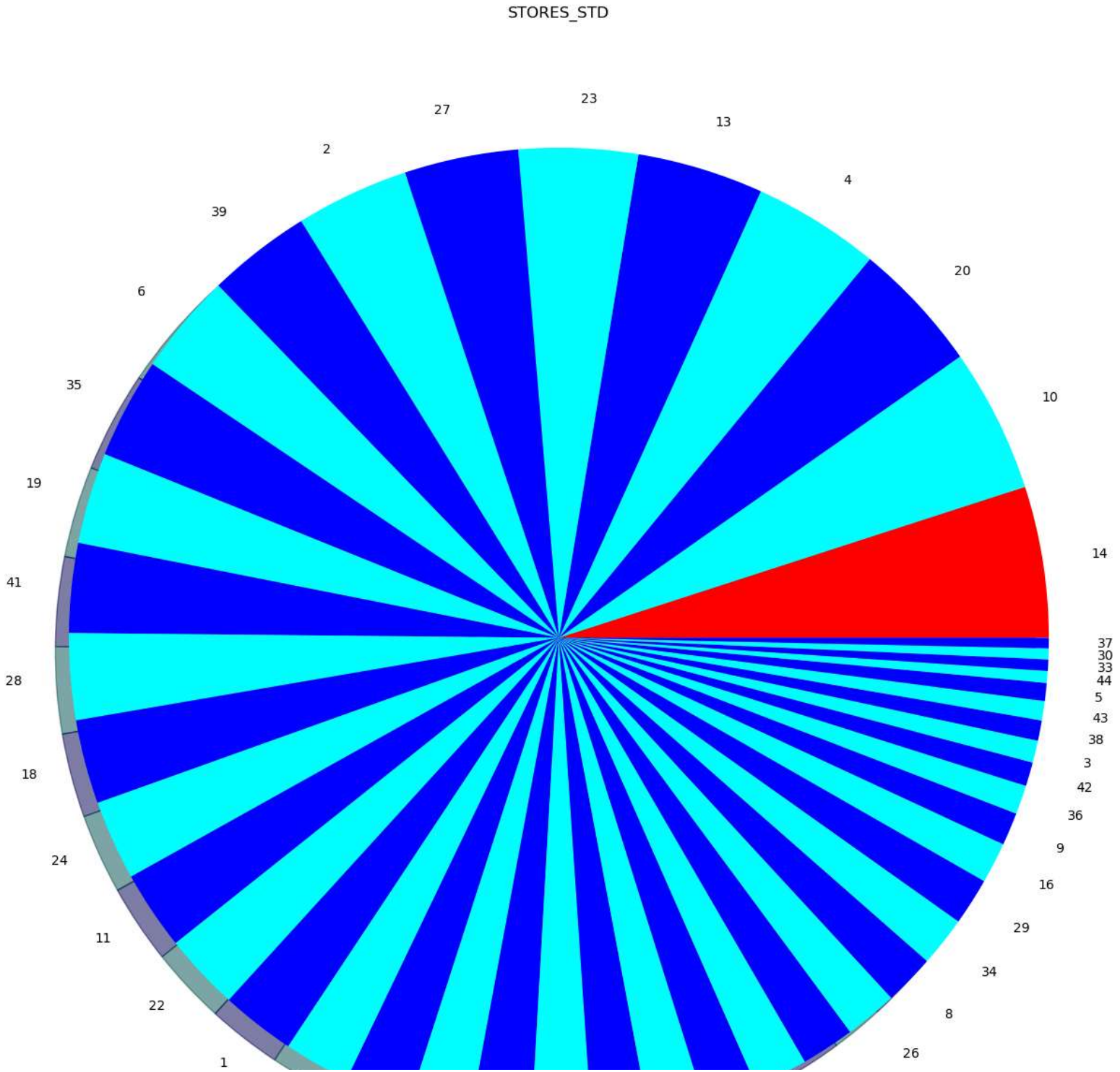
Out[21]:
```
317569.9494755081
```

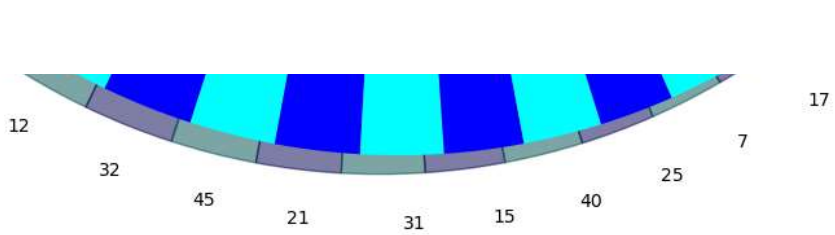
In [22]:
```
maximum_store_std = ordered_std.iloc[0,0]
maximum_sales_std = ordered_std.iloc[0,1]
print("STORE : ",maximum_store_std)
print("with maximum sales std that equals : ",maximum_sales_std)
```

```
STORE :  14
with maximum sales std that equals :  317569.9494755081
```

In [23]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(17, 17))

std_values = ordered_std['STD']
stores = ordered_std['Store']
max_index = std_values.argmax()

colors = ["blue",'cyan'] * len(std_values)
```

```
colors[max_index] = "red"

plt.pie(std_values, labels=stores, colors=colors, shadow=True)

plt.title('STORES_STD')
plt.show()
```

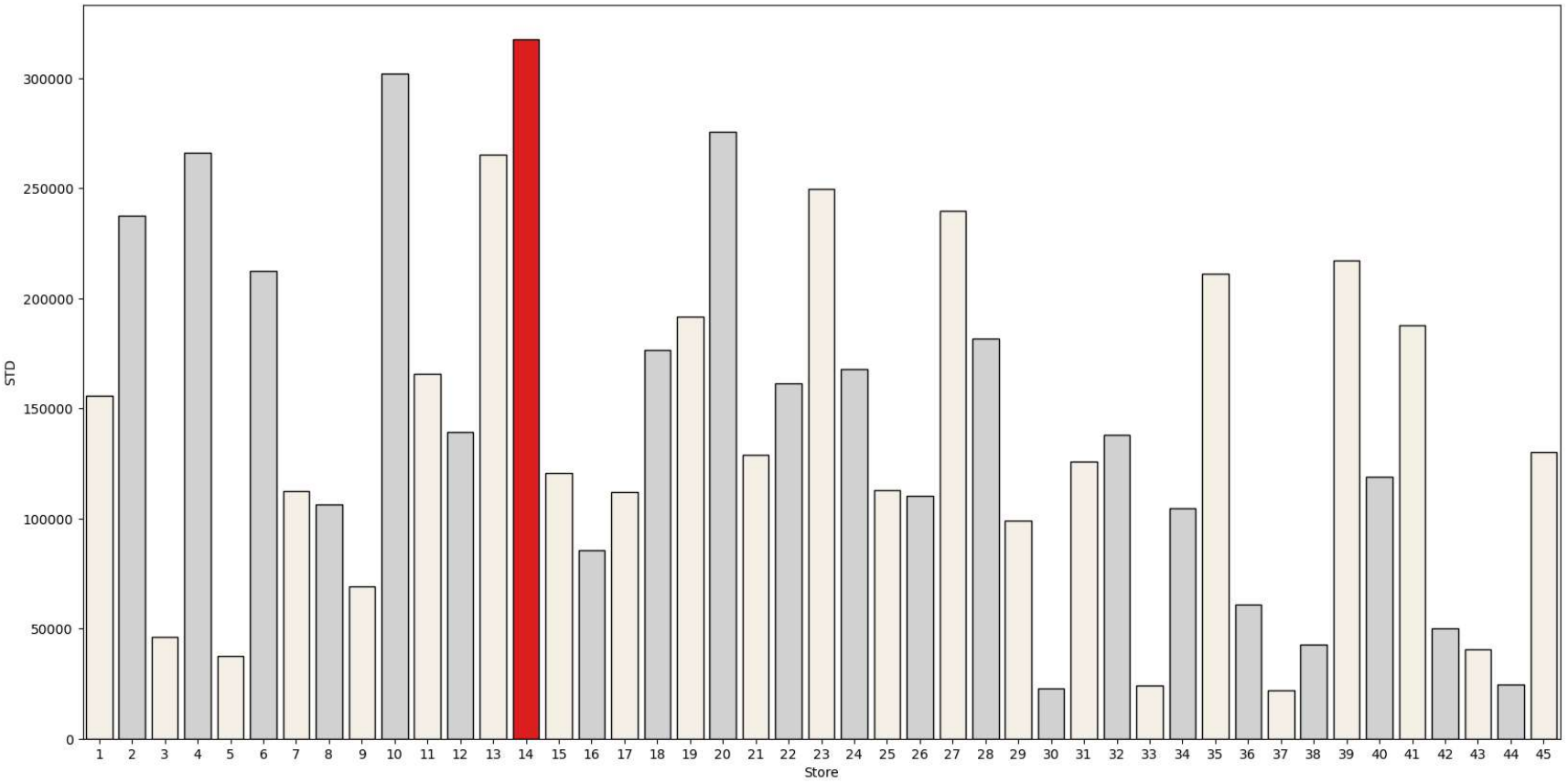

STORES_STD
23
13
4
20
10
14
37
30
33
44
5
43
38
3
42
36
9
16
29
34
8
26
1
22
11
24
18
28
41
19
35
6
39
2
27



here we notice that store 14 has the maximum standard deviation that equals 317569.9494755081

In [24]:

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(20, 10))

store_data = ordered_std

max_index = store_data['STD'].idxmax()

colors = ['#FAF0E6','#D3D3D3'] * len(store_data)
colors[max_index] = 'red'

sns.barplot(x='Store', y='STD', data=store_data, edgecolor='black', palette=colors)

plt.show()
```



Some holidays have a negative impact on sales. there are holidays that have higher sales than the mean sales in the non-holiday season

In [25]:
```
super_bowl_dates = ['2010-02-12', '2011-02-11', '2012-02-10']
labour_day_dates = ['2010-09-10', '2011-09-09', '2012-09-07']
thanksgiving_dates = ['2010-11-26', '2011-11-25', '2012-11-23']
christmas_dates = ['2010-12-31', '2011-12-30', '2012-12-28']
```

In [26]:
```
super_bowl = df[df['Date'].isin(super_bowl_dates)]
labour_day = df[df['Date'].isin(labour_day_dates)]
thanksgiving = df[df['Date'].isin(thanksgiving_dates)]
```

```
christmas = df[df['Date'].isin(christmas_dates)]
NoHoliday = df[df['Holiday_Flag'] == 0]
```

In [27]:
```
thanksgiving.sample(5)
```

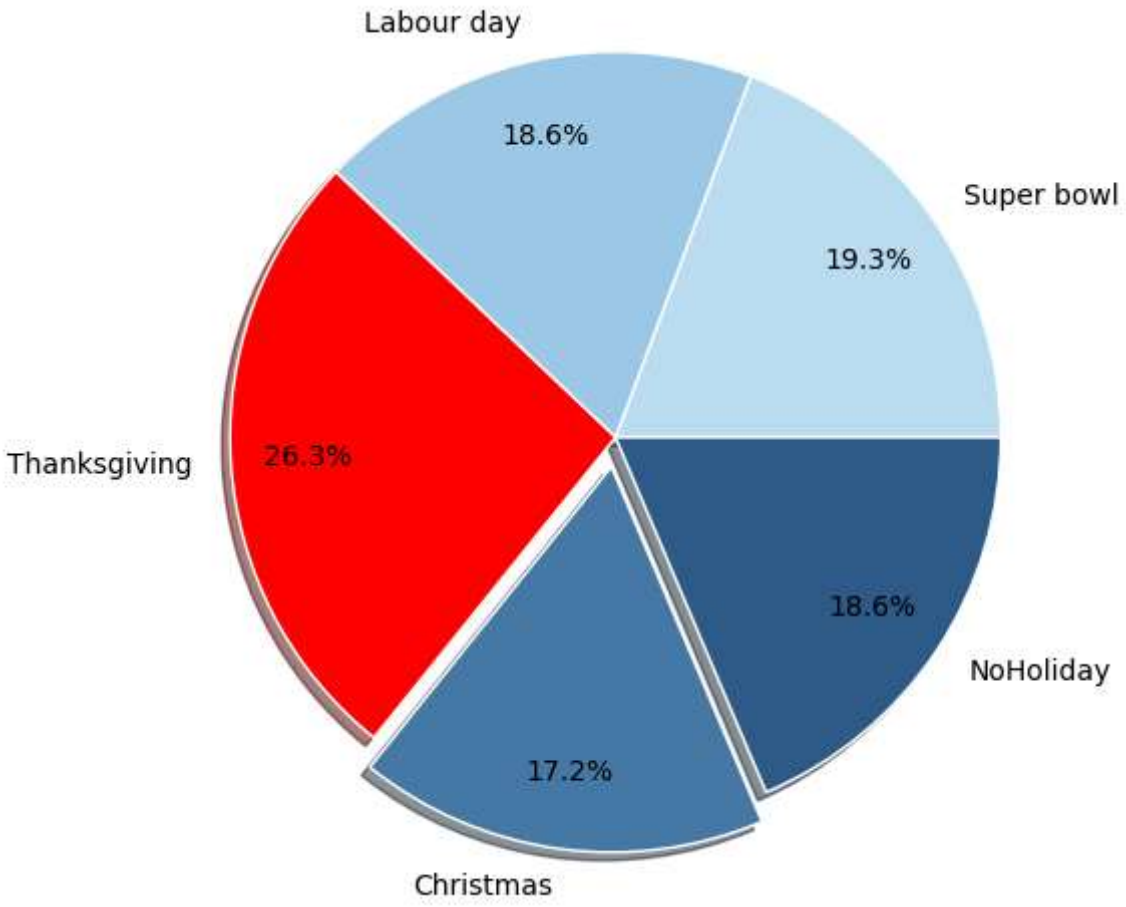
Out[27]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
94	1	2011-11-25	2033321	1	60	3	218	8
185	2	2010-11-26	2658725	1	63	3	211	8
3669	26	2011-11-25	1282320	1	31	4	136	8
1810	13	2011-11-25	2864171	1	39	3	130	6
1901	14	2010-11-26	2921710	1	46	3	183	9

In [28]:
```
mean_sales = [super_bowl['Weekly_Sales'].mean(),
 labour_day['Weekly_Sales'].mean(),
 thanksgiving['Weekly_Sales'].mean(),
 christmas['Weekly_Sales'].mean(),
 NoHoliday['Weekly_Sales'].mean()]
print(mean_sales)
```

```
[1079127.9877037038, 1042427.293925926, 1471273.427777778, 960833.1115555555, 1041256.3802088555]
```

In [29]:
```
holiday_Labels = ['Super bowl','Labour day','Thanksgiving','Christmas','NoHoliday']
Color = ["#B9DDF1", "#9FCAE6", "red", "#497AA7", "#2E5B88"]
fig ,ax = plt.subplots()
ax.pie(mean_sales ,labels = holiday_Labels ,
 radius = 1.3 ,colors = Color , shadow = True ,
 autopct = '%1.1f%%' , pctdistance = 0.8 ,
 explode = [0,0,0,0.1,0] ,wedgeprops ={"linewidth": 1, "edgecolor": "white"})
plt.show()
```



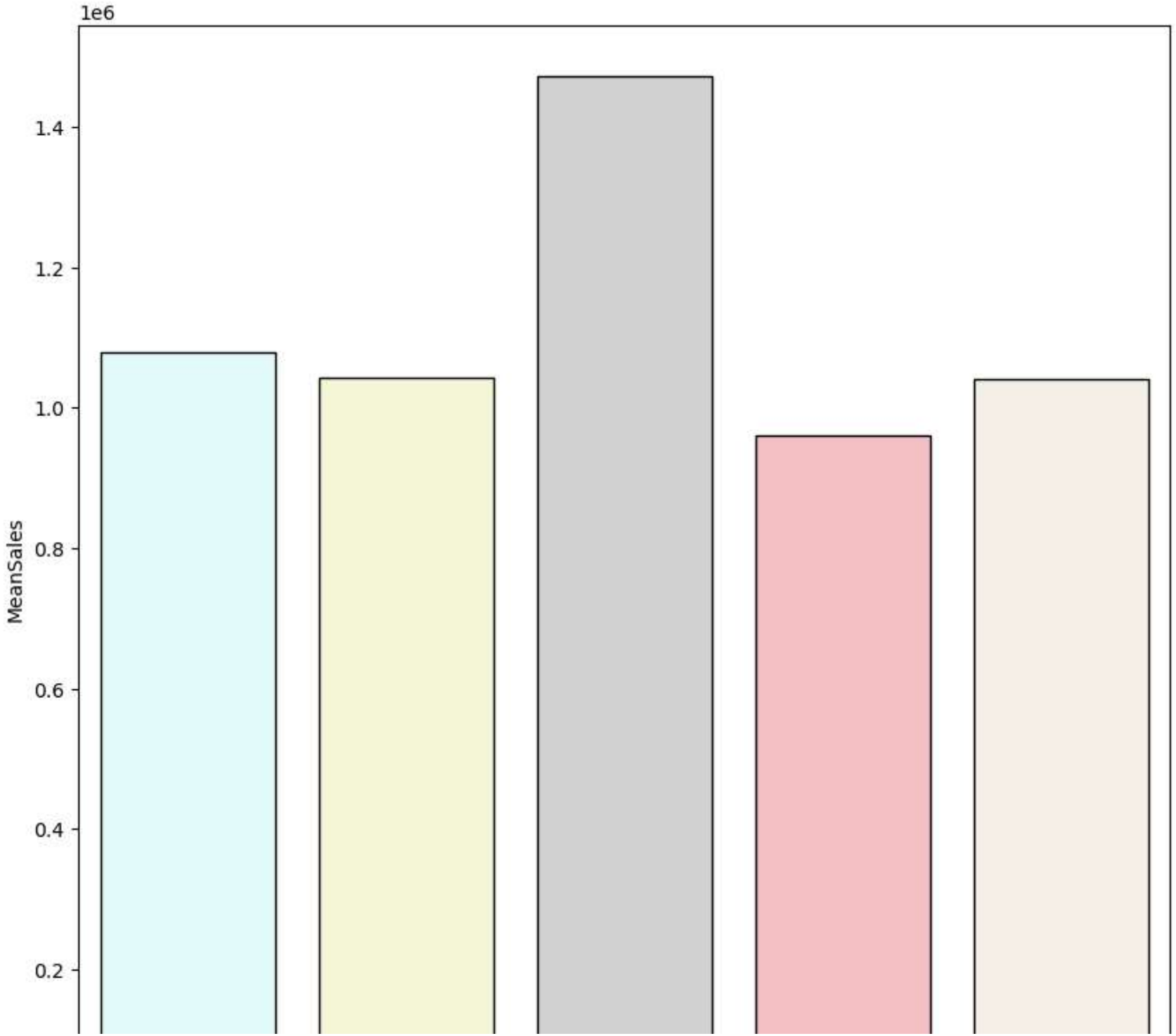
In [30]:
```
Holidays_meanSales = pd.DataFrame({'Holiday': holiday_Labels, 'MeanSales': mean_sales})
pd.options.display.float_format = '{:.0f}'.format
Holidays_meanSales
```

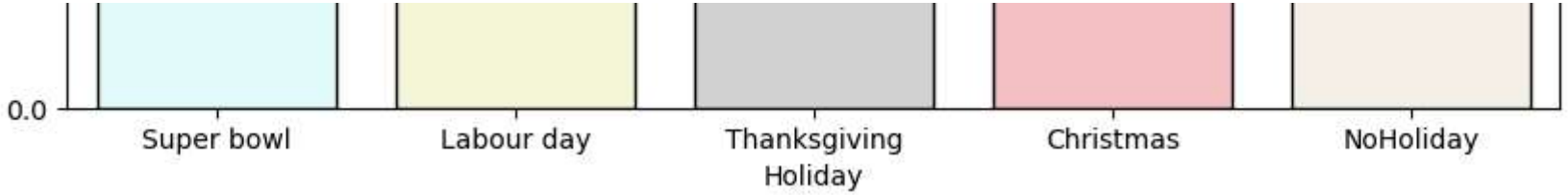
Out[30]:

	Holiday	MeanSales
0	Super bowl	1079128
1	Labour day	1042427
2	Thanksgiving	1471273
3	Christmas	960833
4	NoHoliday	1041256

In [31]:

```
plt.figure(figsize=(10,10))
sns.barplot(x='Holiday',y='MeanSales',data=Holidays_meanSales,palette=['#E0FFFF','#FAFAD2','#D3D3D3','#FFB6C1','#FAF0E6'],edgecolor='black')
plt.show()
```


1e6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
MeanSales



"Our analysis of Weekly Sales data shows that holidays like Christmas, Labour Day, and Super Bowl positively affect mean weekly sales. During holidays, customer spending increases due to related offers and events, such as Thanksgiving Day, which has the highest mean weekly sales, indicating a surge in shopping activities, especially for White Friday sales and holiday offer preparations.

Interestingly, weeks with no holidays also have high mean weekly sales, indicating continuous customer spending patterns outside of holidays. These insights underscore the importance of marketing strategies and inventory management tailored to specific holiday trends. Understanding these patterns helps stakeholders and retailers optimize their approach to maximize sales and enhance business performance, facilitating business expansion."

monthly and semester view of sales

In [48]:
```
df['year'] = df['Date'].dt.year
df['semester'] = df['Date'].apply(lambda x: 1 if x.month <= 6 else 2)
df['month'] = df['Date'].dt.month
```

In [49]:
```
df.sample(10)
```

Out[49]:

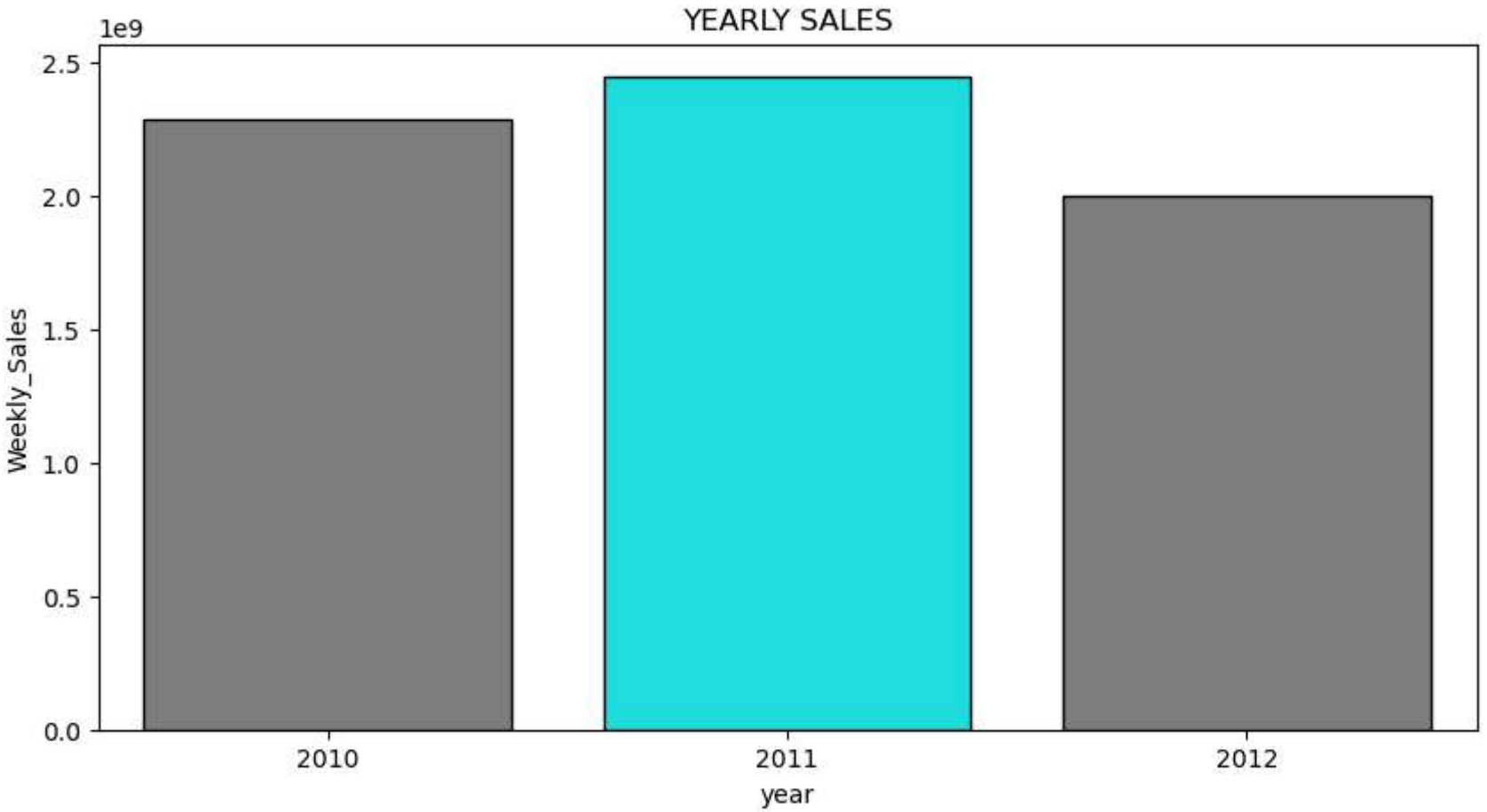
	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	year	semester	month
2624	19	2011-01-21	1212968	0	23	3	133	8	2011	1	1
6305	45	2010-05-07	812191	0	71	3	182	9	2010	1	5
988	7	2012-08-03	680955	0	62	4	198	8	2012	2	8
3550	25	2012-05-11	739866	0	59	4	215	7	2012	1	5
2112	15	2012-03-16	570611	0	47	4	138	8	2012	1	3
3028	22	2010-07-30	970774	0	76	3	136	8	2010	2	7
1907	14	2011-01-07	1864746	0	34	3	183	9	2011	1	1
1208	9	2011-04-29	532226	0	71	4	219	6	2011	1	4
4075	29	2011-06-17	536914	0	64	4	135	10	2011	1	6
3807	27	2011-10-21	1689591	0	60	4	140	8	2011	2	10

In [57]:
```
year_sales = df.groupby('year')['Weekly_Sales'].sum().reset_index()
```

In [79]:
```
month_sales = df.groupby('month')['Weekly_Sales'].sum().reset_index()
```

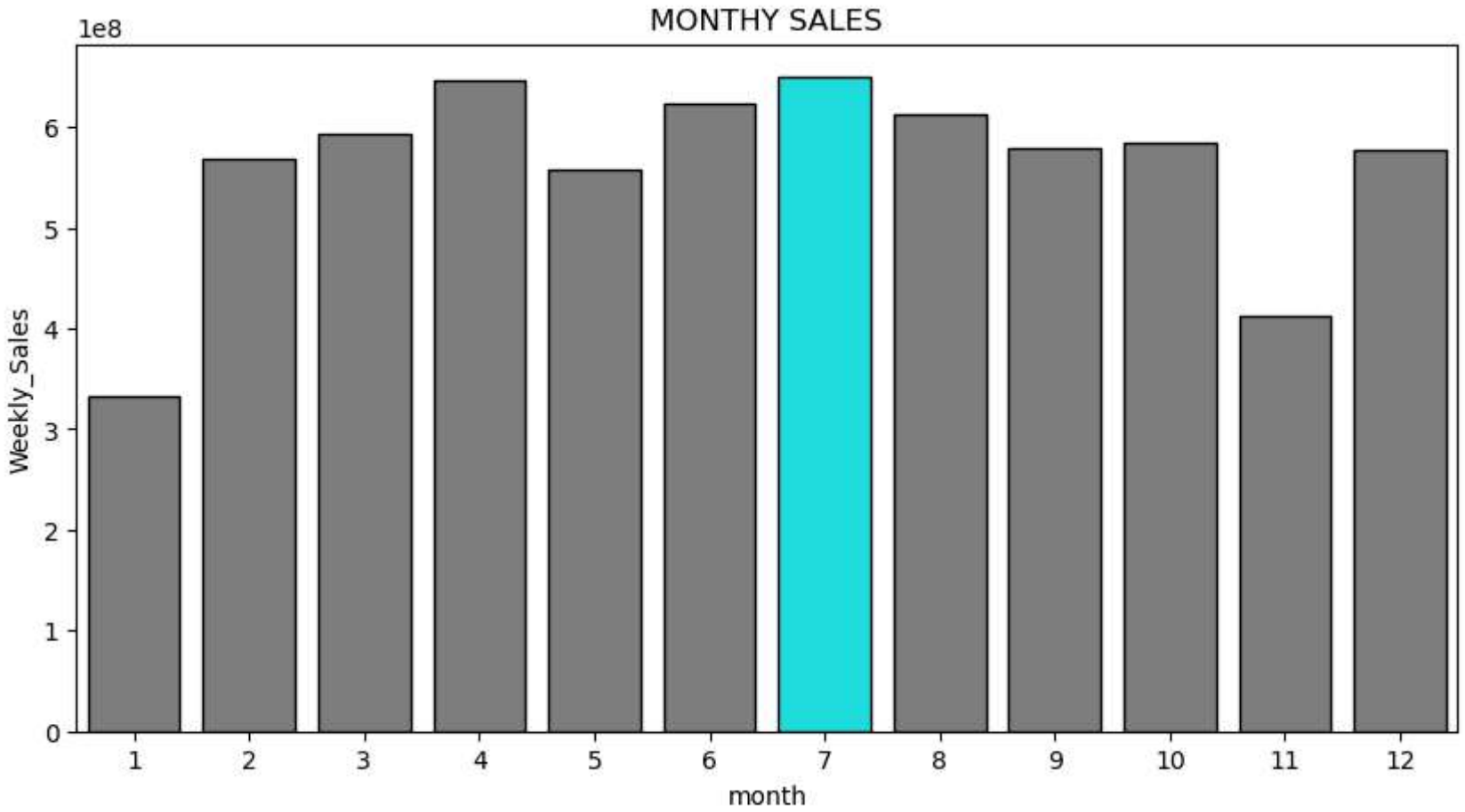
In [82]:
```
semester_sales = df.groupby('semester')['Weekly_Sales'].sum().reset_index()
```

In [90]:
```
max_year = year_sales['Weekly_Sales'].idxmax()
clr = ['grey']*len(year_sales)
clr[max_year]='cyan'
plt.figure(figsize=(10,5))
sns.barplot(x='year',y='Weekly_Sales',data=year_sales,palette=clr,edgecolor='black')
plt.title('YEARLY SALES')
plt.show()
print('yearly avg sales : ',year_sales['Weekly_Sales'].mean() )
```



yearly avg sales : 2245739662.3700004

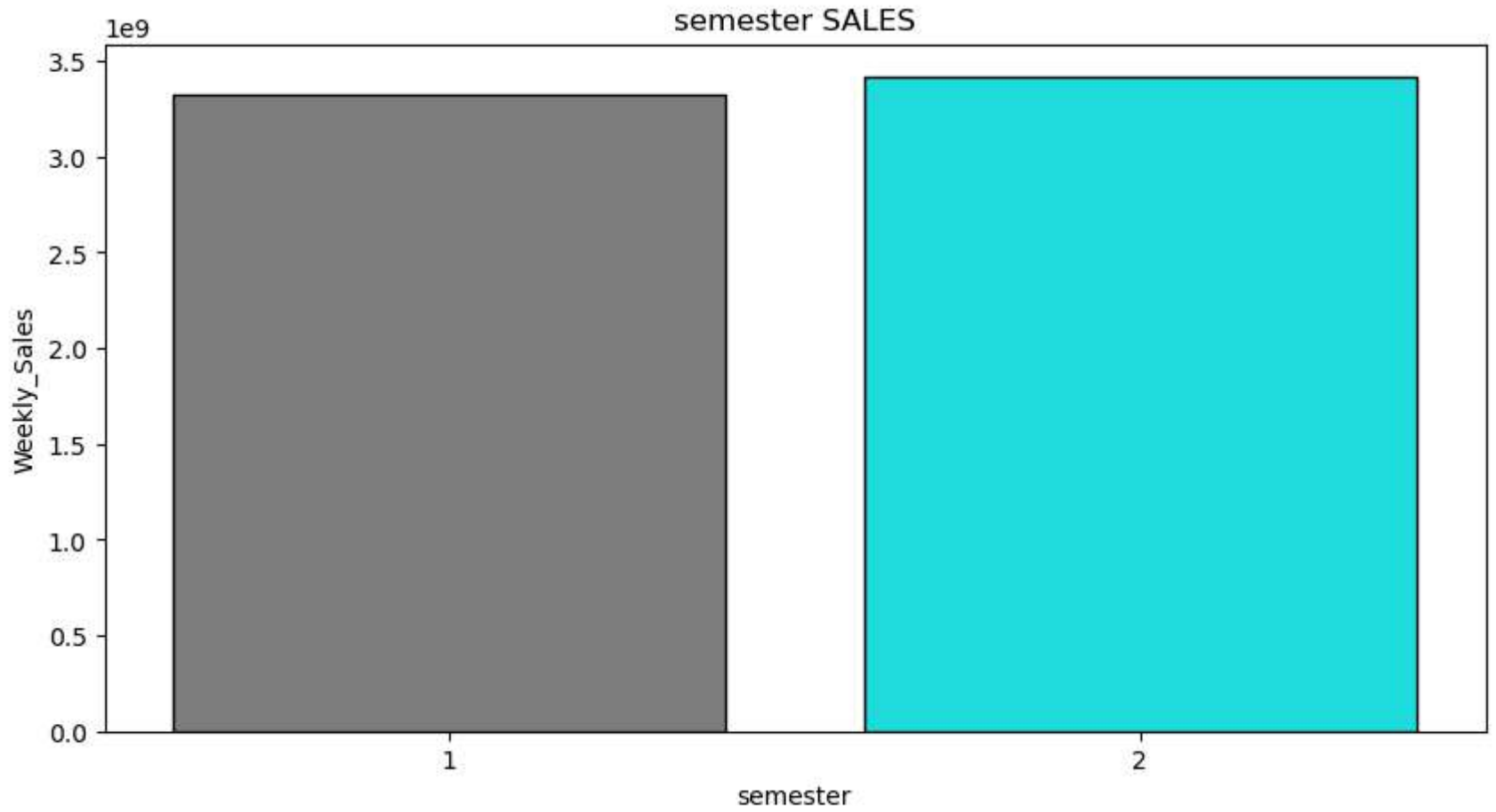
In [91]:
```
max_month =month_sales['Weekly_Sales'].idxmax()
clr = ['grey']*len(month_sales)
clr[max_month]='cyan'
plt.figure(figsize=(10,5))
sns.barplot(x='month',y='Weekly_Sales',data=month_sales,palette=clr,edgecolor='black')
plt.title('MONTHY SALES')
plt.show()
print('monthly avg sales : ',month_sales['Weekly_Sales'].mean() )
```



monthly avg sales : 561434915.5925001

In [92]:

```
max_semester =semester_sales['Weekly_Sales'].idxmax()
clr = ['grey']*len(semester_sales)
clr[max_semester]='cyan'
plt.figure(figsize=(10,5))
sns.barplot(x='semester',y='Weekly_Sales',data=semester_sales,palette=clr,edgecolor='black')
plt.title('semester SALES')
plt.show()
print('semester avg sales : ',semester_sales['Weekly_Sales'].mean() )
```



semester avg sales : 3368609493.5550003

The sales data from 2010 to 2012 reveals important trends about the business's performance. In 2010, weekly sales were around 2.29 billion, setting a base level. 2011 showed strong growth, with sales hitting about 2.45 billion. However, in 2012, sales dropped to around 2.00 billion. These ups and downs suggest changing market conditions and external factors influencing consumer behavior.

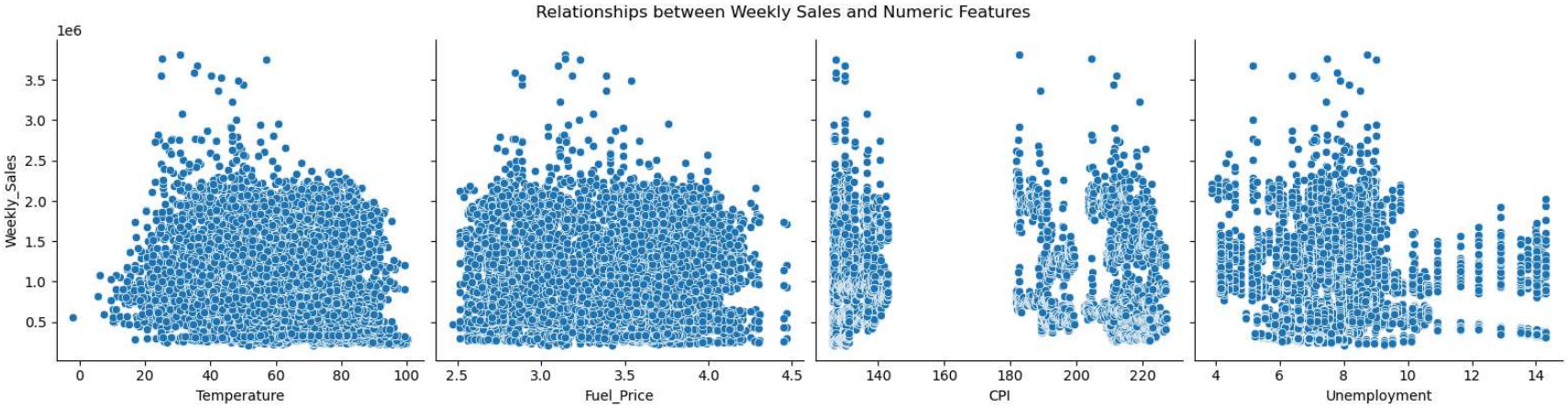
On average, yearly sales were about 2,245,739,662.37. But looking at semesters, the average was higher, totaling 3,368,609,493.56. This indicates some seasonal patterns or impactful periods each year. Monthly sales averaged around 561,434,915.59, highlighting the importance of understanding monthly trends for marketing, inventory, and resource planning.

In summary, it's essential to consider both yearly trends and shorter-term fluctuations. Investigating factors like economic conditions, marketing efforts, and external events could provide valuable insights for making strategic decisions and improving sales strategies.

relations between weekly sales vs. other numeric features

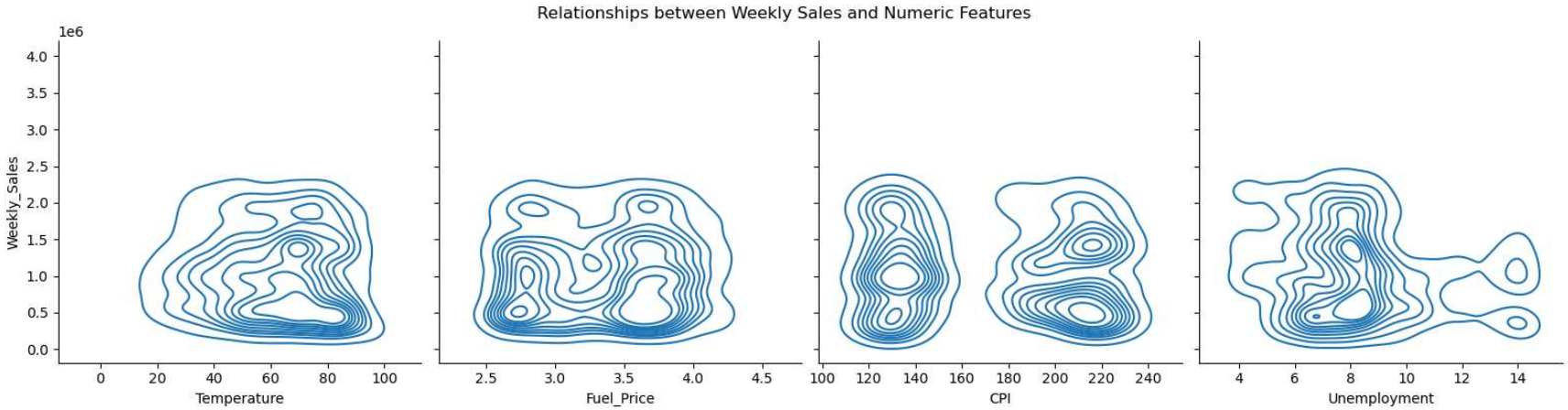
In [93]:
```
numeric_features = ['Temperature', 'Fuel_Price', 'CPI', 'Unemployment']
plt.figure(figsize=(20, 6))
sns.pairplot(data=df, x_vars=numeric_features, y_vars='Weekly_Sales', kind='scatter', height=4)
plt.suptitle('Relationships between Weekly Sales and Numeric Features', y=1.02)
plt.show()
```

<Figure size 2000x600 with 0 Axes>



In [94]:
```
numeric_features = ['Temperature', 'Fuel_Price', 'CPI', 'Unemployment']
plt.figure(figsize=(20, 6))
sns.pairplot(data=df, x_vars=numeric_features, y_vars='Weekly_Sales', kind='kde', height=4)
plt.suptitle('Relationships between Weekly Sales and Numeric Features', y=1.02)
plt.show()
```

<Figure size 2000x600 with 0 Axes>



In [95]:
```
temp_sales = df.groupby('Temperature')['Weekly_Sales'].mean().reset_index()
fuel_sales = df.groupby('Fuel_Price')['Weekly_Sales'].mean().reset_index()
cpi_sales = df.groupby('CPI')['Weekly_Sales'].mean().reset_index()
unemployment_sales = df.groupby('Unemployment')['Weekly_Sales'].mean().reset_index()
```