Walmart Data Analysis

In [1]:
```
# importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Reading the csv file and storing it in dataframe df
df = pd.read_csv('Walmart_sales_analysis.csv')
```

In [3]:
```
# displaying first 5 rows
df
```

Out[3]:

	Store_Number	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price
0	1	2/5/2010	1,643,691	0	42.31	2.572
1	1	2/12/2010	1,641,957	1	38.51	2.548
2	1	2/19/2010	1,611,968	0	39.93	2.514
3	1	2/26/2010	1,409,728	0	46.63	2.561
4	1	3/5/2010	1,554,807	0	46.50	2.625
...	...	...	...	...	...	...
6430	45	9/28/2012	713,174	0	64.88	3.997
6431	45	10/5/2012	733,455	0	64.89	3.985
6432	45	10/12/2012	734,464	0	54.47	4.000
6433	45	10/19/2012	718,126	0	56.47	3.969
6434	45	10/26/2012	760,281	0	58.85	3.882

6435 rows × 8 columns

If we observe, Weekly Sales column has commas. We need to remove them

In [4]:
```
# displaying the shape(How many columns and how many rows are present in this
df.shape
```

Out[4]: (6435, 8)

In [5]:
```
# Displaying datatypes of each column and checking how many non null entries a
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_Number  6435 non-null   int64
 1   Date          6435 non-null   object
 2   Weekly_Sales  6435 non-null   object
 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   int64
 7   Unemployment  6435 non-null   float64
dtypes: float64(3), int64(3), object(2)
memory usage: 402.3+ KB
```

By looking at the information we can say that there are no null values present in the dataset. Date is not in Date type, we need to convert it.

1. Fixing column names. Removing extra spaces in the column names
2. We need to remove commas in weekly sales column. And convert to numeric
3. We need to convert data type of Date column to date type

In [6]:
```
# Removing any leading/trailing spaces in column names
df.columns = df.columns.str.strip()
```

In [7]:
```
# Removing commas and converting to float
df['Weekly_Sales'] = df['Weekly_Sales'].str.replace(',','')
```

In [8]:
```
df.head()
```

Out[8]:

	Store_Number	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	C
0	1	2/5/2010	1643691	0	42.31	2.572	2
1	1	2/12/2010	1641957	1	38.51	2.548	2
2	1	2/19/2010	1611968	0	39.93	2.514	2
3	1	2/26/2010	1409728	0	46.63	2.561	2
4	1	3/5/2010	1554807	0	46.50	2.625	2

In [9]:
```
df['Weekly_Sales'] = df['Weekly_Sales'].astype(float)
```

In [10]:
```
df['Weekly_Sales'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 6435 entries, 0 to 6434
Series name: Weekly_Sales
Non-Null Count  Dtype
--------------  -----
6435 non-null   float64
dtypes: float64(1)
memory usage: 50.4 KB
```

In [11]:
```
# Converting Date column to datetime
df['Date'] = pd.to_datetime(df['Date'])
```

In [12]:
```
df['Date'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 6435 entries, 0 to 6434
Series name: Date
Non-Null Count  Dtype
--------------  -----
6435 non-null   datetime64[ns]
dtypes: datetime64[ns](1)
memory usage: 50.4 KB
```

In [13]:
```
df.dtypes
```

Out[13]:

	0
Store_Number	int64
Date	datetime64[ns]
Weekly_Sales	float64
Holiday_Flag	int64
Temperature	float64
Fuel_Price	float64
CPI	int64
Unemployment	float64

dtype: object

In [14]:
```
df.isna().sum()
```

Out[14]:

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

dtype: int64

In [15]:
```
df.duplicated().sum()
```

Out[15]:
```
np.int64(0)
```

In [16]:
```
df.describe()
```

Out[16]:

	Store_Number	Date	Weekly_Sales	Holiday_Flag	Temperature	Fu
count	6435.000000	6435	6.435000e+03	6435.000000	6435.000000	643
mean	23.000000	2011-06-17 00:00:00	1.046965e+06	0.069930	60.663782	
min	1.000000	2010-02-05 00:00:00	2.099860e+05	0.000000	-2.060000	
25%	12.000000	2010-10-08 00:00:00	5.533500e+05	0.000000	47.460000	
50%	23.000000	2011-06-17 00:00:00	9.607460e+05	0.000000	62.670000	
75%	34.000000	2012-02-24 00:00:00	1.420158e+06	0.000000	74.940000	
max	45.000000	2012-10-26 00:00:00	3.818686e+06	1.000000	100.140000	
std	12.988182	NaN	5.643666e+05	0.255049	18.444933	

In [17]:
```
df.head()
```

Out[17]:

	Store_Number	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Pri
0	1	2010-02-05	1643691.0	0	42.31	2.5
1	1	2010-02-12	1641957.0	1	38.51	2.5
2	1	2010-02-19	1611968.0	0	39.93	2.5
3	1	2010-02-26	1409728.0	0	46.63	2.5
4	1	2010-03-05	1554807.0	0	46.50	2.6

EDA

Which store has the highest average weekly sales?

In [18]:
```
avg_sales_per_store = df.groupby('Store_Number')['Weekly_Sales'].apply('mean')
```

In [19]:
```
# Top 5 stores
top_5_stores = avg_sales_per_store.head()
```

In [20]:
```
top_5_stores
```

Out[20]:

	Store_Number	Weekly_Sales
0	20	2.107677e+06
1	4	2.094713e+06
2	14	2.020978e+06
3	13	2.003620e+06
4	2	1.925751e+06

In [21]:
```
top_5_stores.columns = ['Store_Number', 'Average_Weekly_Sales']
```

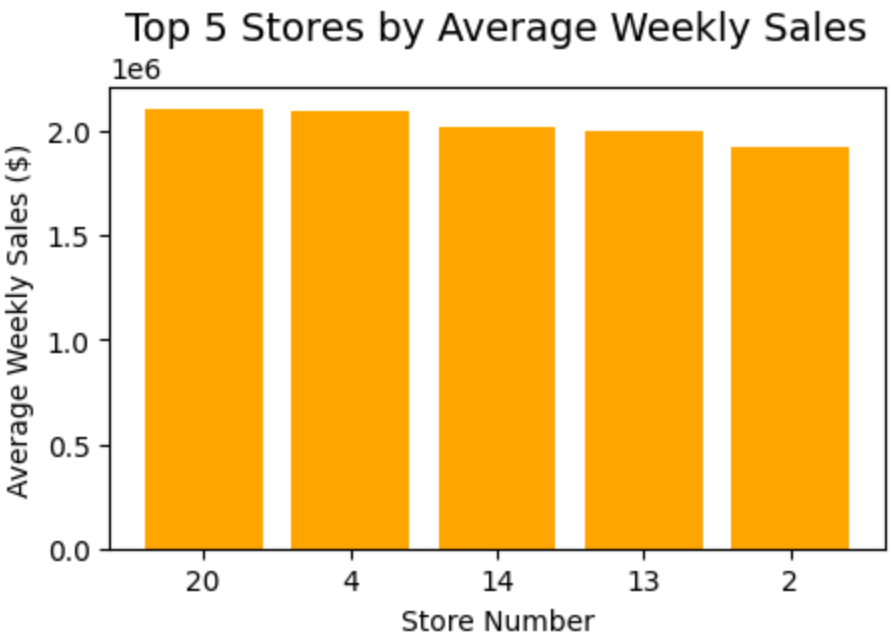
In [22]:
```
top_5_stores.loc[:,'Average_Weekly_Sales'] = top_5_stores['Average_Weekly_Sale
```

In [23]:
```
top_5_stores
```

Out[23]:

	Store_Number	Average_Weekly_Sales
0	20	2107677.0
1	4	2094713.0
2	14	2020978.0
3	13	2003620.0
4	2	1925751.0

In [24]:
```
plt.figure(figsize=(5,3))
plt.bar(top_5_stores['Store_Number'].astype(str),top_5_stores['Average_Weekly_
plt.title('Top 5 Stores by Average Weekly Sales', fontsize=14)
plt.xlabel('Store Number')
plt.ylabel('Average Weekly Sales ($)')
plt.show()
```



Total Weekly Sales Trend Over Time

In [25]:
```
total_sales_over_time = df.groupby('Date')['Weekly_Sales'].sum().reset_index()
```

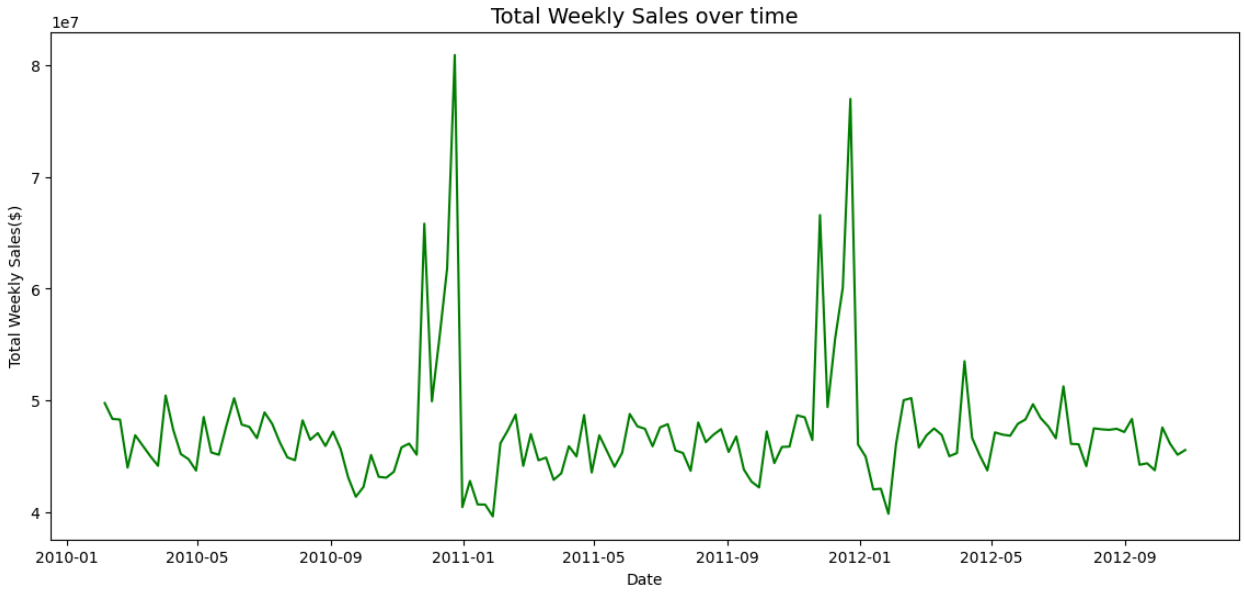
In [26]:
```
total_sales_over_time.columns = ['Date','Total_Weekly_Sales']
```

In [27]:
```
total_sales_over_time.head(2)
```

Out[27]:

	Date	Total_Weekly_Sales
0	2010-02-05	49750741.0
1	2010-02-12	48336676.0

In [28]:
```
plt.figure(figsize=(14,6))
plt.plot(total_sales_over_time['Date'],total_sales_over_time['Total_Weekly_Sal
plt.title('Total Weekly Sales over time', fontsize= 14)
plt.xlabel('Date')
plt.ylabel('Total Weekly Sales($)')
plt.show()
```



Do Sales Increase During Holidays?

In [29]:
```
holiday_sales = df.groupby('Holiday_Flag')['Weekly_Sales'].apply('mean').reset
holiday_sales
```

Out[29]:

	Holiday_Flag	Weekly_Sales
0	0	1.041256e+06
1	1	1.122888e+06

In [30]:
```
holiday_sales.columns = ['Holiday_Flag','Average_Weekly_Sales']
holiday_sales
```

Out[30]:

	Holiday_Flag	Average_Weekly_Sales
0	0	1.041256e+06
1	1	1.122888e+06

In [31]:
```
holiday_sales['Average_Weekly_Sales'] = holiday_sales['Average_Weekly_Sales'].
```

In [32]:
```
holiday_sales
```

Out[32]:

	Holiday_Flag	Average_Weekly_Sales
0	0	1041256
1	1	1122888

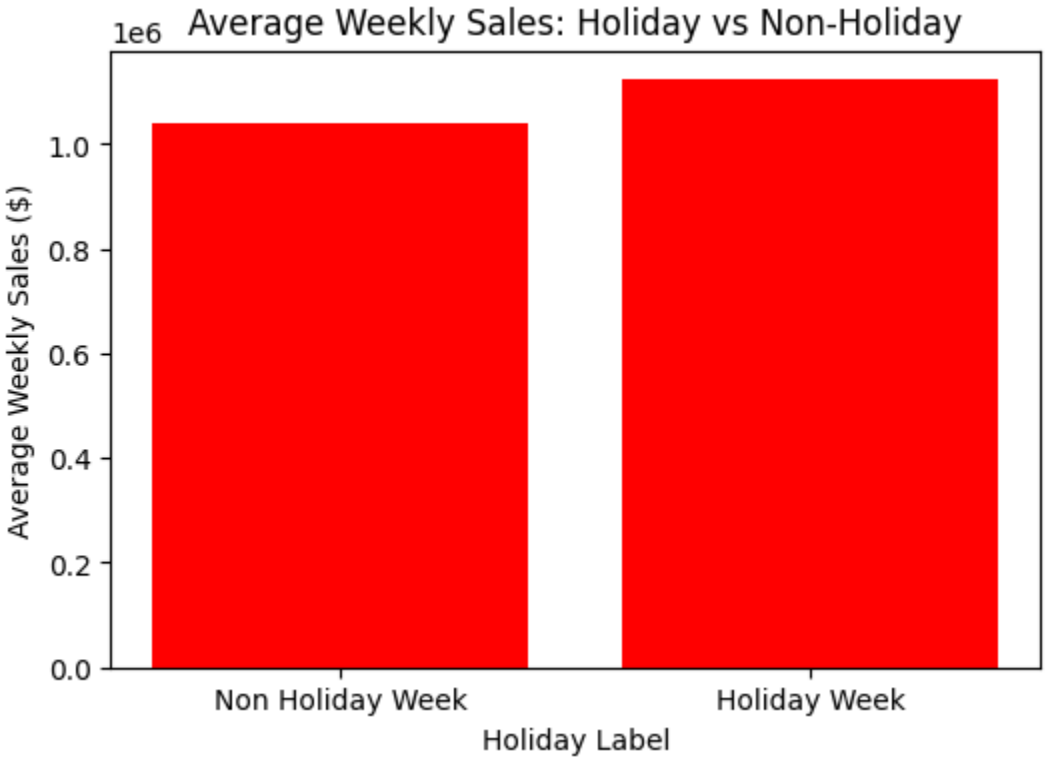
In [33]:
```
holiday_sales['Holiday_label'] = holiday_sales['Holiday_Flag'].map({0:'Non Hol
```

In [34]:
```
holiday_sales
```

Out[34]:

	Holiday_Flag	Average_Weekly_Sales	Holiday_label
0	0	1041256	Non Holiday Week
1	1	1122888	Holiday Week

In [35]:
```
plt.figure(figsize=(6,4))
plt.bar(holiday_sales['Holiday_label'], holiday_sales['Average_Weekly_Sales'],
plt.title('Average Weekly Sales: Holiday vs Non-Holiday')
plt.xlabel('Holiday Label')
plt.ylabel('Average Weekly Sales ($)')
plt.show()
```



Monthly or Seasonal Sales Patterns

In [36]:
```
# Creating new column called month
df['month'] = df['Date'].dt.month
```

In [37]:
```
df.head(2)
```

Out[37]:

	Store_Number	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Pri
0	1	2010-02-05	1643691.0	0	42.31	2.5
1	1	2010-02-12	1641957.0	1	38.51	2.5

In [38]:
```
# Calculating average sales per month

monthly_sales = df.groupby('month')['Weekly_Sales'].mean().reset_index()
monthly_sales['Weekly_Sales'] = monthly_sales['Weekly_Sales'].round(2)
monthly_sales
```

Out[38]:

	month	Weekly_Sales
0	1	923884.56
1	2	1053199.81
2	3	1013309.21
3	4	1026761.56
4	5	1031714.03
5	6	1064324.59
6	7	1031747.59
7	8	1048017.46
8	9	989335.36
9	10	999632.10
10	11	1147265.93
11	12	1281863.63

In [39]:
```
monthly_sales.head()
```

Out[39]:

	month	Weekly_Sales
0	1	923884.56
1	2	1053199.81
2	3	1013309.21
3	4	1026761.56
4	5	1031714.03

In [40]:
```
import calendar
monthly_sales['month'] = monthly_sales['month'].apply(lambda x: calendar.month
```

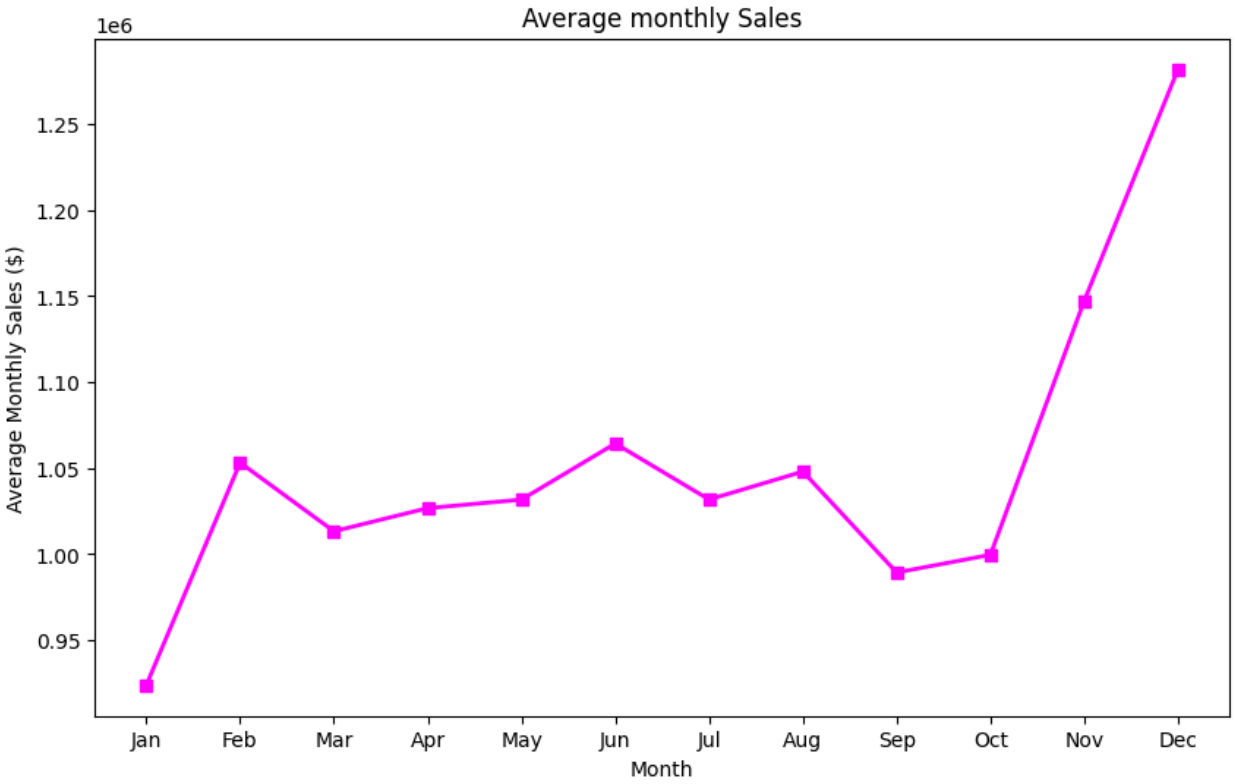
In [41]:
```
monthly_sales
```

Out[41]:

	month	Weekly_Sales
0	Jan	923884.56
1	Feb	1053199.81
2	Mar	1013309.21
3	Apr	1026761.56
4	May	1031714.03
5	Jun	1064324.59
6	Jul	1031747.59
7	Aug	1048017.46
8	Sep	989335.36
9	Oct	999632.10
10	Nov	1147265.93
11	Dec	1281863.63

In [42]:

```
plt.figure(figsize=(10,6))
plt.plot(monthly_sales['month'], monthly_sales['Weekly_Sales'],marker='s',colo
#plt.xticks(range('Jan', 'Dec'))
plt.title('Average monthly Sales')
plt.xlabel('Month')
plt.ylabel('Average Monthly Sales ($)')
plt.show()
```



Correlation Between Weekly Sales and External Factors

In [43]:
```
df.head(2)
```

Out[43]:

	Store_Number	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Pri
0	1	2010-02-05	1643691.0	0	42.31	2.5
1	1	2010-02-12	1641957.0	1	38.51	2.5

In [44]:
```
corr_df = df[['Weekly_Sales','Temperature','Fuel_Price','CPI','Unemployment']]
```

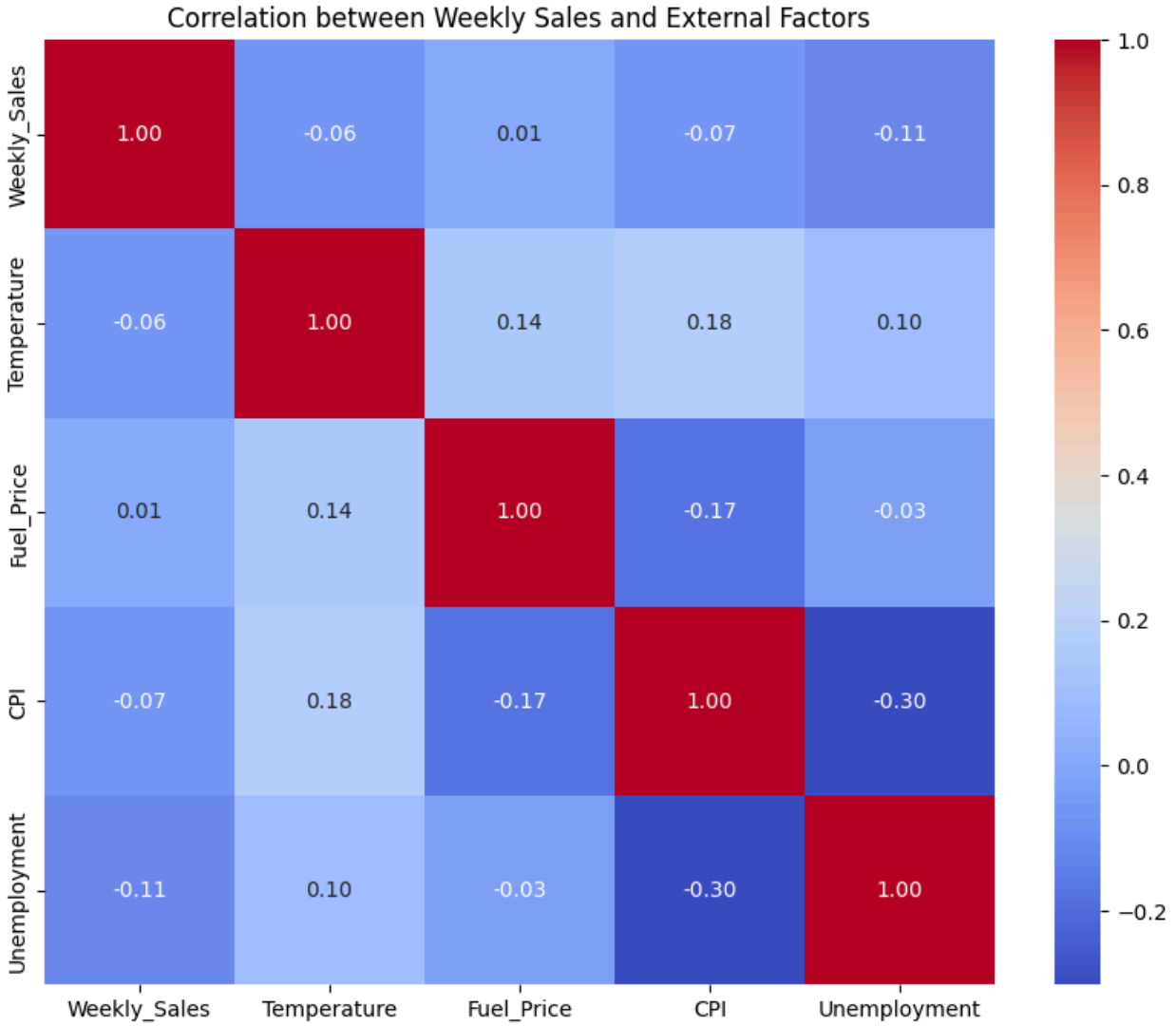
In [45]:
```
correlation_matrix = corr_df.corr()
correlation_matrix
```

Out[45]:

	Weekly_Sales	Temperature	Fuel_Price	CPI	Unemployme
Weekly_Sales	1.000000	-0.063810	0.009464	-0.072496	-0.10617
Temperature	-0.063810	1.000000	0.144982	0.176188	0.10115
Fuel_Price	0.009464	0.144982	1.000000	-0.170880	-0.03468
CPI	-0.072496	0.176188	-0.170880	1.000000	-0.30216
Unemployment	-0.106176	0.101158	-0.034684	-0.302162	1.00000

In [46]:

```
plt.figure(figsize=(10,8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation between Weekly Sales and External Factors')
plt.show()
```



Consistency vs. Volatility of Store Sales

In [47]:
```
df.head(2)
```

Out[47]:

	Store_Number	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Pri
0	1	2010-02-05	1643691.0	0	42.31	2.5
1	1	2010-02-12	1641957.0	1	38.51	2.5

In [48]:
```
store_sales_std = df.groupby('Store_Number')['Weekly_Sales'].std().reset_index
store_sales_std.head()
```

Out[48]:

	Store_Number	Weekly_Sales
0	1	155980.759881
1	2	237683.724553
2	3	46319.621759
3	4	266201.396823
4	5	37737.961155

In [49]:
```
store_sales_std.columns = ['Store_Number', 'Sales_Std_Dev']
```

In [50]:
```
store_sales_std.head(2)
```

Out[50]:

	Store_Number	Sales_Std_Dev
0	1	155980.759881
1	2	237683.724553

In [51]:
```
most_consistent_store = store_sales_std.sort_values(by='Sales_Std_Dev', ignore
print("Most Consistent Stores: ")
print()
print(most_consistent_store)
```

```
Most Consistent Stores:

   Store_Number  Sales_Std_Dev
0            37   21837.424358
1            30   22809.661016
2            33   24132.978647
3            44   24762.838370
4             5   37737.961155
```

In [52]:
```
most_volatile_stores = store_sales_std.sort_values(by='Sales_Std_Dev', ascendi
print("Most Volatile Stores: ")
```

```
print()
print(most_volatile_stores)
```

```
Most Volatile Stores:

   Store_Number  Sales_Std_Dev
0            14  317569.957816
1            10  302262.099050
2            20  275900.518187
3             4  266201.396823
4            13  265506.996140
```

Top-Selling Stores During Holidays Only

In [53]:
```
df.head(2)
```

Out[53]:

	Store_Number	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Pri
0	1	2010-02-05	1643691.0	0	42.31	2.5
1	1	2010-02-12	1641957.0	1	38.51	2.5

In [54]:
```
holiday_data = df.loc[df['Holiday_Flag'] == 1]
holiday_data
```

Out[54]:

	Store_Number	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel
1	1	2010-02-12	1641957.0	1	38.51	
31	1	2010-09-10	1507461.0	1	78.69	
42	1	2010-11-26	1955624.0	1	64.52	
47	1	2010-12-31	1367320.0	1	48.43	
53	1	2011-02-11	1649615.0	1	36.39	
...	...	...	...	...	...	
6375	45	2011-09-09	746130.0	1	71.48	
6386	45	2011-11-25	1170673.0	1	48.71	
6391	45	2011-12-30	869404.0	1	37.79	
6397	45	2012-02-10	803657.0	1	37.00	
6427	45	2012-09-07	766513.0	1	75.70	

450 rows × 9 columns

In [55]:
```
holiday_avg_store_sales = holiday_data.groupby('Store_Number')['Weekly_Sales']
holiday_avg_store_sales.columns = ['Store_Number','Holiday_Avg_Weekly_Sales']
holiday_avg_store_sales.head()
```

Out[55]:

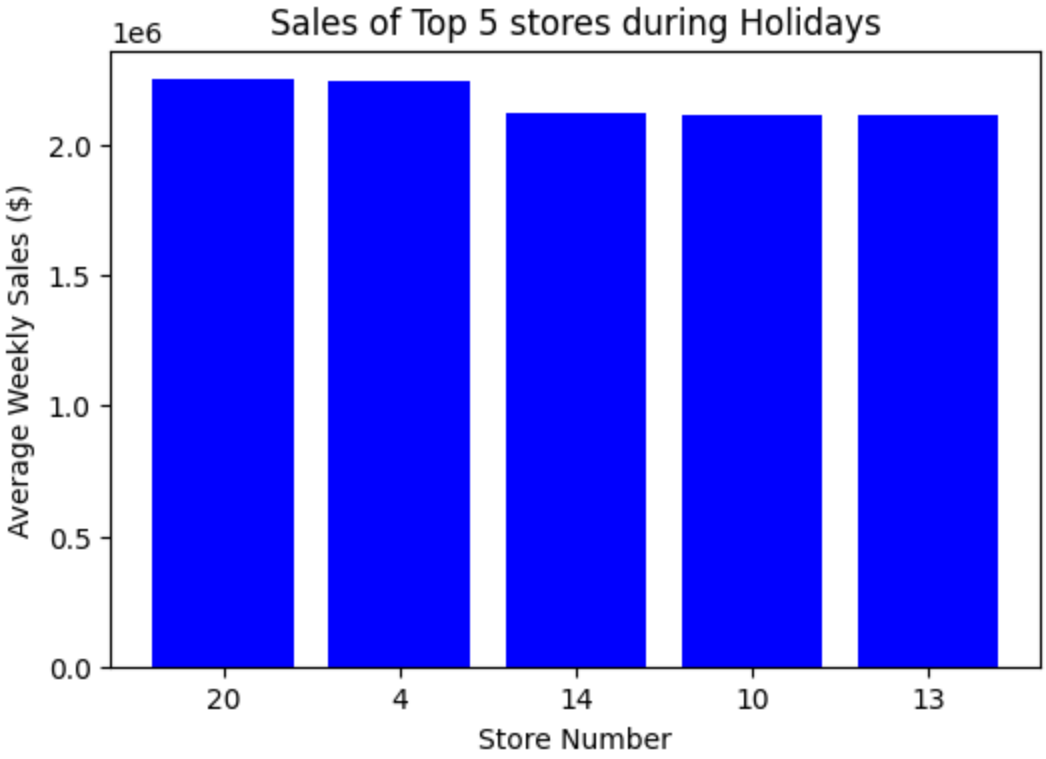
	Store_Number	Holiday_Avg_Weekly_Sales
0	1	1665747.6
1	2	2079267.1
2	3	437811.0
3	4	2243102.6
4	5	359501.7

In [56]:
```
top_holiday_stores =  holiday_avg_store_sales.sort_values(by='Holiday_Avg_Week
top_holiday_stores
```

Out[56]:

	Store_Number	Holiday_Avg_Weekly_Sales
19	20	2249035.1
3	4	2243102.6
13	14	2120583.0
9	10	2113755.9
12	13	2113043.8

In [57]:
```
plt.figure(figsize=(6,4))
plt.bar(top_holiday_stores['Store_Number'].astype(str), top_holiday_stores['Ho
plt.title('Sales of Top 5 stores during Holidays')
plt.xlabel('Store Number')
plt.ylabel('Average Weekly Sales ($)')
plt.show()
```



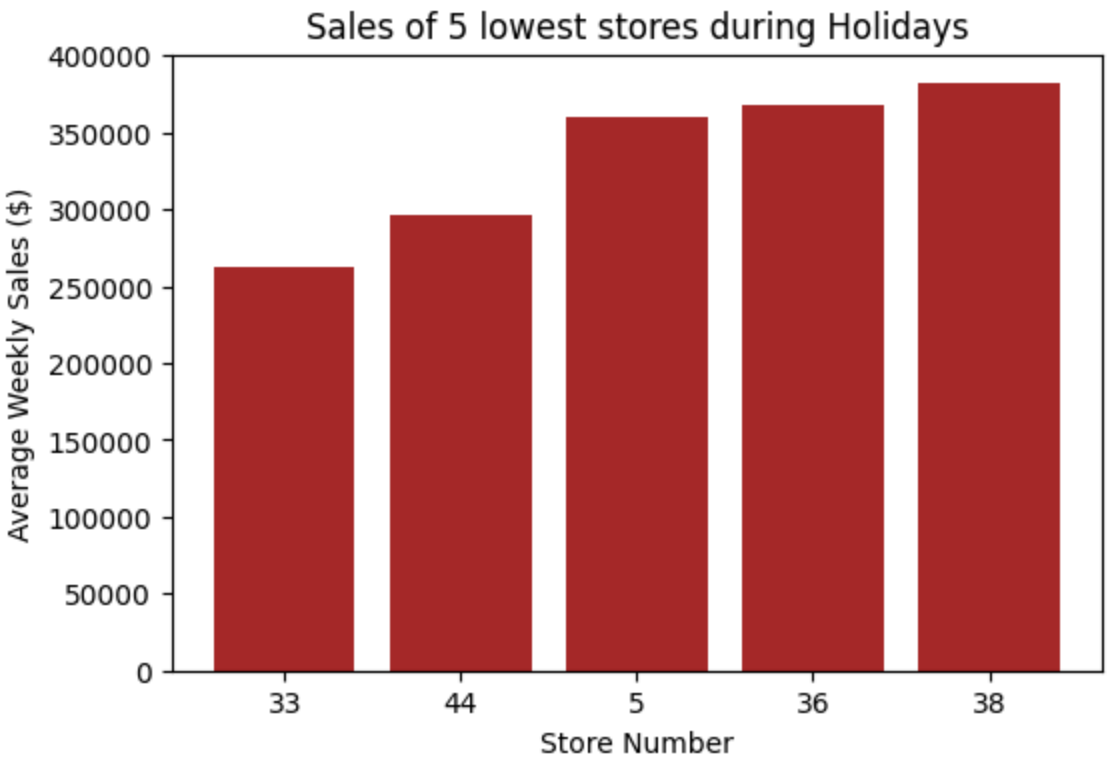
Stores with Lowest Average Weekly Sales

In [58]:
```
lowest_holiday_stores = holiday_avg_store_sales.sort_values(by='Holiday_Avg_We
lowest_holiday_stores
```

Out[58]:

	Store_Number	Holiday_Avg_Weekly_Sales
32	33	262594.6
43	44	296035.6
4	5	359501.7
35	36	367640.7
37	38	381509.9

In [59]:
```
plt.figure(figsize=(6,4))
plt.bar(lowest_holiday_stores['Store_Number'].astype(str), lowest_holiday_stor
plt.title('Sales of 5 lowest stores during Holidays')
plt.xlabel('Store Number')
plt.ylabel('Average Weekly Sales ($)')
plt.show()
```



In [60]:
```
df.head(2)
```

Out[60]:

	Store_Number	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Pri
0	1	2010-02-05	1643691.0	0	42.31	2.5
1	1	2010-02-12	1641957.0	1	38.51	2.5

Sales Patterns by Weekday

In [61]:
```
df['Day_of_Week'] = df['Date'].dt.day_name()
```

In [62]:
```
df['Day_of_Week'].value_counts()
```

Out[62]:

Day_of_Week	count
Friday	6435

dtype: int64

Dataset contains only Friday entries.

Why? Because Walmart typically reports weekly sales ending on Fridays.

We don’t need to do "Sales by Day of Week" analysis — there’s no variation

Weekly Sales by Quarter

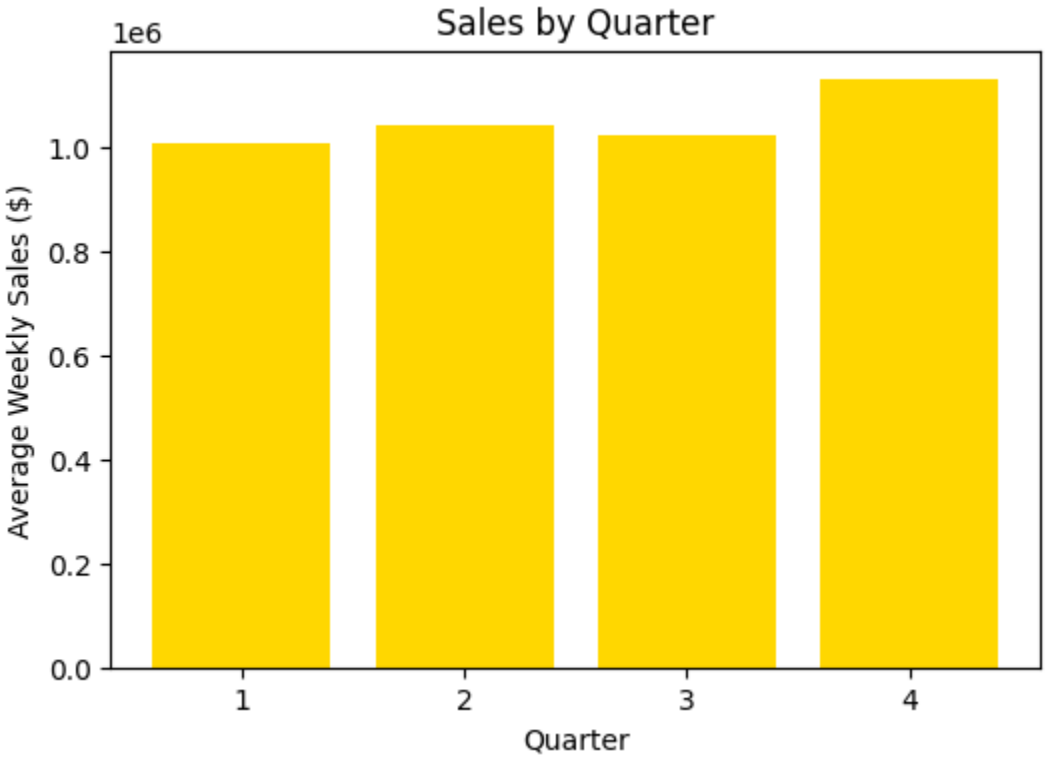
In [63]:
```
df['Quarter'] = df['Date'].dt.quarter
```

In [64]:
```
sales_by_quarter = df.groupby('Quarter')['Weekly_Sales'].mean().reset_index()
sales_by_quarter.head()
```

Out[64]:

	Quarter	Weekly_Sales
0	1	1.006136e+06
1	2	1.040806e+06
2	3	1.023251e+06
3	4	1.128774e+06

In [65]:
```
plt.figure(figsize=(6,4))
plt.bar(sales_by_quarter['Quarter'].astype(str), sales_by_quarter['Weekly_Sale
plt.title("Sales by Quarter")
plt.xlabel('Quarter')
plt.ylabel("Average Weekly Sales ($)")
plt.show()
```



In [ ]: