|  |
| --- |
| *# Import necessary libraries* **import** pandas **as** pd **import** seaborn **as** sns **import** numpy **as** np **import** matplotlib.pyplot **as** plt **from** matplotlib **import** dates **from** datetime **import** datetime |

In [2]:

|  |
| --- |
| *##Business Understanding*  Walmart **is** an American retail corporation that operates a chain of hypermarkets, di  In this project, we focused to answer the following questions:  1.Which store has minimum **and** maximum sales**?**  2.Which store has maximum standard deviation i**.**e**.**, the sales vary a lot**.**  Also, find out the coefficient of mean to standard deviation  3.Which store**/**s has good quarterly growth rate **in** Q3’2012  4.Some holidays have a negative impact on sales**.** Find out holidays which have higher sales than the mean sales **in** non**-**holiday season **for** all stores together 5.Provide a monthly **and** semester view of sales **in** units **and** give insights 6.Build prediction to forecast demand**.**  *#Data Understanding*  There are sales data available **for** 45 stores of Walmart **in** Kaggle**.** This **is** the data sales **from** 2010**-**02**-**05 to 2012**-**11**-**01.  The data contains these features:   * Store **-** the store number **\*** Date **-** the week of sales * Weekly\_Sales **-** sales **for** the given store * Holiday\_Flag **-** whether the week **is** a special holiday week 1 – Holiday week 0 – * Temperature **-** Temperature on the day of sale * Fuel\_Price **-** Cost of fuel **in** the region * CPI – Prevailing consumer price index * Unemployment **-** Prevailing unemployment rate |

In [ ]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *## Data Preparation* | | |  |  |
|  | | |  |  |
| *# Load dataset*  data **=** pd**.**read\_csv('Walmart\_Store\_sales.csv') data | | |  |  |
|  | **Store** | **Date Weekly\_Sales Holiday\_Flag** | **Temperature** | **Fuel\_Price** |
| **0** | 1 | 05-  02- 1643690.90 0  2010 | 42.31 | 2.572 211.0 |
| **1** | 1 | 12-  02- 1641957.44 1  2010 | 38.51 | 2.548 211.2 |

In [ ]:

In [3]:

Out[3]:

19-

**2** 1 02- 1611968.17 0 39.93 2.514 211.2

2010

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **3** | 1 | 26-  022010 | 1409727.59 | 0 | 46.63 | 2.561 | 211.3 |
| **4** | 1 | 05-  032010 | 1554806.68 | 0 | 46.50 | 2.625 | 211.3 |
| **...** | ... | ... | ... | ... | ... | ... |  |
| **6430** | 45 | 28-  092012 | 713173.95 | 0 | 64.88 | 3.997 | 192.0 |
| **6431** | 45 | 05-  102012 | 733455.07 | 0 | 64.89 | 3.985 | 192.1 |
| **6432** | 45 | 12-  102012 | 734464.36 | 0 | 54.47 | 4.000 | 192.3 |
| **6433** | 45 | 19-  102012 | 718125.53 | 0 | 56.47 | 3.969 | 192.3 |
| **6434** | 45 | 26-  102012 | 760281.43 | 0 | 58.85 | 3.882 | 192.3 |

6435 rows × 8 columns

|  |
| --- |
| **import** pandas **as** pd  *# Assuming 'data' is your DataFrame*  data['Date'] **=** pd**.**to\_datetime(data['Date'], format**=**'%d-%m-%Y') data**.**info() |

In [50]:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 6435 entries, 0 to 6434 Data columns (total 11 columns):

# Column Non-Null Count Dtype --- ------ -------------- ----- 0 Store 6435 non-null int64

1. Date 6435 non-null datetime64[ns]
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
8. Day 6435 non-null int32
9. Month 6435 non-null int32 10 Year 6435 non-null int32 dtypes: datetime64[ns](1), float64(5), int32(3), int64(2) memory usage: 477.7 KB

In [48]: *# checking for missing values* data**.**isnull()**.**sum()

|  |  |
| --- | --- |
| Out[48]: | Store 0  Date 0  Weekly\_Sales 0  Holiday\_Flag 0  Temperature 0  Fuel\_Price 0  CPI 0  Unemployment 0  Day 0  Month 0 Year 0 dtype: int64 |

In [6]: *# Splitting Date and create new columns (Day, Month, and Year)* data["Day"]**=** pd**.**DatetimeIndex(data['Date'])**.**day data['Month'] **=** pd**.**DatetimeIndex(data['Date'])**.**month data['Year'] **=** pd**.**DatetimeIndex(data['Date'])**.**year data

# Out[6]: Store Date Weekly\_Sales Holiday\_Flag Temperature Fuel\_Price

05-

**0** 1 02- 1643690.90 0 42.31 2.572 211.0

2010

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 1 | 12-  022010 | 1641957.44 | 1 | 38.51 | 2.548 | 211.2 |
| **2** | 1 | 19-  022010 | 1611968.17 | 0 | 39.93 | 2.514 | 211.2 |
| **3** | 1 | 26-  022010 | 1409727.59 | 0 | 46.63 | 2.561 | 211.3 |
| **4** | 1 | 05-  032010 | 1554806.68 | 0 | 46.50 | 2.625 | 211.3 |
| **...** | ... | ... | ... | ... | ... | ... |  |
| **6430** | 45 | 28-  092012 | 713173.95 | 0 | 64.88 | 3.997 | 192.0 |
| **6431** | 45 | 05-  102012 | 733455.07 | 0 | 64.89 | 3.985 | 192.1 |
| **6432** | 45 | 12-  102012 | 734464.36 | 0 | 54.47 | 4.000 | 192.3 |
| **6433** | 45 | 19-  102012 | 718125.53 | 0 | 56.47 | 3.969 | 192.3 |

26-

**6434** 45 10- 760281.43 0 58.85 3.882 192.3

2012 6435 rows × 11 columns

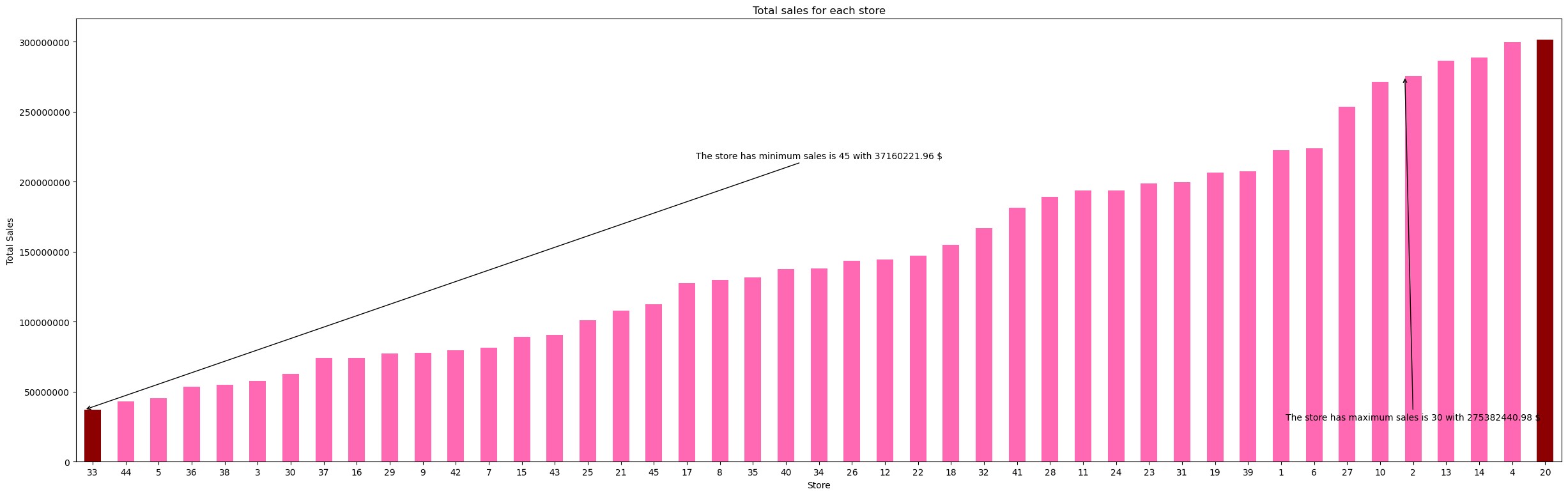
|  |
| --- |
| Q1: Which store has minimum **and** maximum sales**?** |

In [ ]:

|  |
| --- |
| plt**.**figure(figsize**=**(30,9))  *# Sum Weekly\_Sales for each store, then sortded by total sales*  total\_sales\_for\_each\_store **=** data**.**groupby('Store')['Weekly\_Sales']**.**sum()**.**sort\_value total\_sales\_for\_each\_store\_array **=** np**.**array(total\_sales\_for\_each\_store) *# convert t*  *# Assigning a specific color for the stores have the lowest and highest sales* clrs **=** ['hotpink' **if** ((x **<** max(total\_sales\_for\_each\_store\_array)) **and** (x **>** min(tota  ax **=** total\_sales\_for\_each\_store**.**plot(kind**=**'bar',color**=**clrs);  *# store have minimum sales* p **=** ax**.**patches[0] print(type(p**.**get\_height())) ax**.**annotate("The store has minimum sales is 45 with {0:.2f} $"**.**format((p**.**get\_height xytext**=**(0.50, 0.69), textcoords**=**'axes fraction', arrowprops**=**dict(arrowstyle**=**"->", connectionstyle**=**"arc3"), horizontalalignment**=**'center', verticalalignment**=**'center')  *# store have maximum sales* p **=** ax**.**patches[40] ax**.**annotate("The store has maximum sales is 30 with {0:.2f} $"**.**format((p**.**get\_height xytext**=**(0.90, 0.100), textcoords**=**'axes fraction', arrowprops**=**dict(arrowstyle**=**"->", connectionstyle**=**"arc3"), horizontalalignment**=**'center', verticalalignment**=**'center')  *# plot properties* plt**.**xticks(rotation**=**1) plt**.**ticklabel\_format(useOffset**=False**, style**=**'plain', axis**=**'y') plt**.**title('Total sales for each store') plt**.**xlabel('Store') plt**.**ylabel('Total Sales'); |

In [36]:

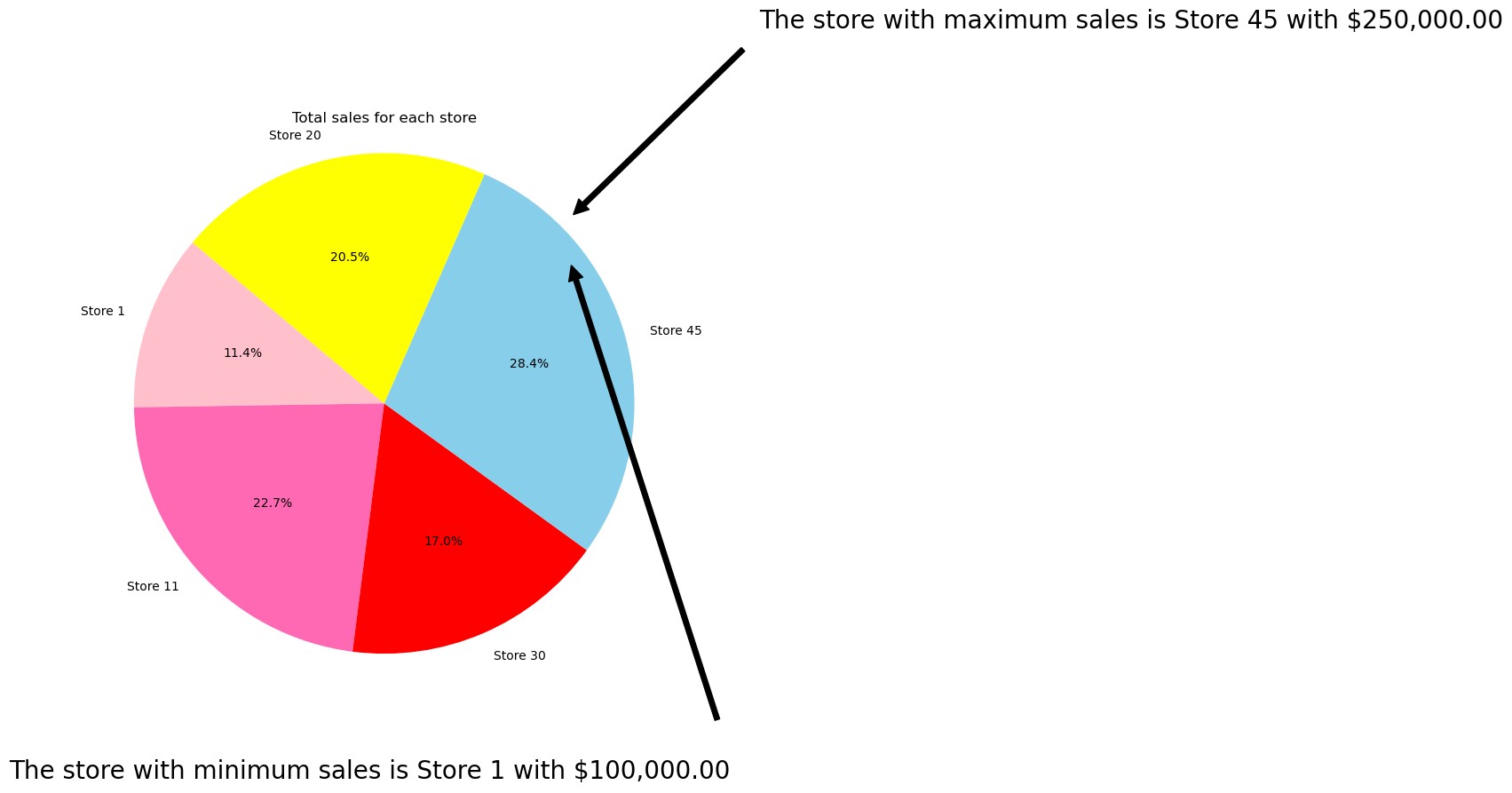
<class 'numpy.float64'>



|  |
| --- |
| Q2: Which store has maximum standard deviation i**.**e**.**, the sales vary a lot**.** Also, find out the coefficient of mean to standard deviation**?** |

|  |
| --- |
| **import** matplotlib.pyplot **as** plt **import** numpy **as** np  *# Sample data for demonstration*  stores **=** ['Store 1', 'Store 11', 'Store 30', 'Store 45', 'Store 20'] total\_sales **=** [100000, 200000, 150000, 250000, 180000]  *# Finding the store with the minimum and maximum sales* min\_sales\_store **=** stores[np**.**argmin(total\_sales)] max\_sales\_store **=** stores[np**.**argmax(total\_sales)]  *# Custom colors for the pie chart*  custom\_colors **=** ['pink', 'hotpink', 'red', 'skyblue', 'yellow']  *# Plotting the pie chart with custom colors* plt**.**figure(figsize**=**(8, 8)) plt**.**pie(total\_sales, labels**=**stores, autopct**=**'%1.1f%%', startangle**=**140, colors**=**custo plt**.**title('Total sales for each store')  *# Annotating the store with minimum sales*  plt**.**annotate(f"The store with minimum sales is {min\_sales\_store} with ${min(total\_s xy**=**(0.7, 0.7), xytext**=**(**-**1.5, **-**1.5), fontsize**=**20, color**=**'black', arrowprops**=**dict(facecolor**=**'black', shrink**=**0.07))  *# Annotating the store with maximum sales*  plt**.**annotate(f"The store with maximum sales is {max\_sales\_store} with ${max(total\_s xy**=**(0.7, 0.7), xytext**=**(1.5, 1.5), fontsize**=**20, color**=**'black', arrowprops**=**dict(facecolor**=**'black', shrink**=**0.07)) plt**.**axis('equal') *# Equal aspect ratio ensures that pie is drawn as a circle.*  *# Show the pie chart* plt**.**show() |

In [ ]: In [46]:



|  |
| --- |
| *# Which store has maximum standard deviation*  store\_max\_std **=** data**.**groupby('Store')['Weekly\_Sales']**.**std()**.**idxmax() max\_std\_value **=** data**.**groupby('Store')['Weekly\_Sales']**.**std()**.**max() print("The store with the maximum standard deviation is Store", store\_max\_std, "wit |

In [51]:

|  |
| --- |
| *# Distribution of store has maximum standard deviation* store\_std **=** data**.**groupby('Store')['Weekly\_Sales']**.**std()  *# Store with maximum standard deviation* store\_max\_std **=** store\_std**.**idxmax()  plt**.**figure(figsize**=**(20, 9)) sns**.**distplot(data[data['Store'] **==** store\_max\_std]['Weekly\_Sales'], color**=**'darkblue' plt**.**title('The Sales Distribution of Store #' **+** str(store\_max\_std)) plt**.**show() |

The store with the maximum standard deviation is Store 14 with $317570 In [56]:

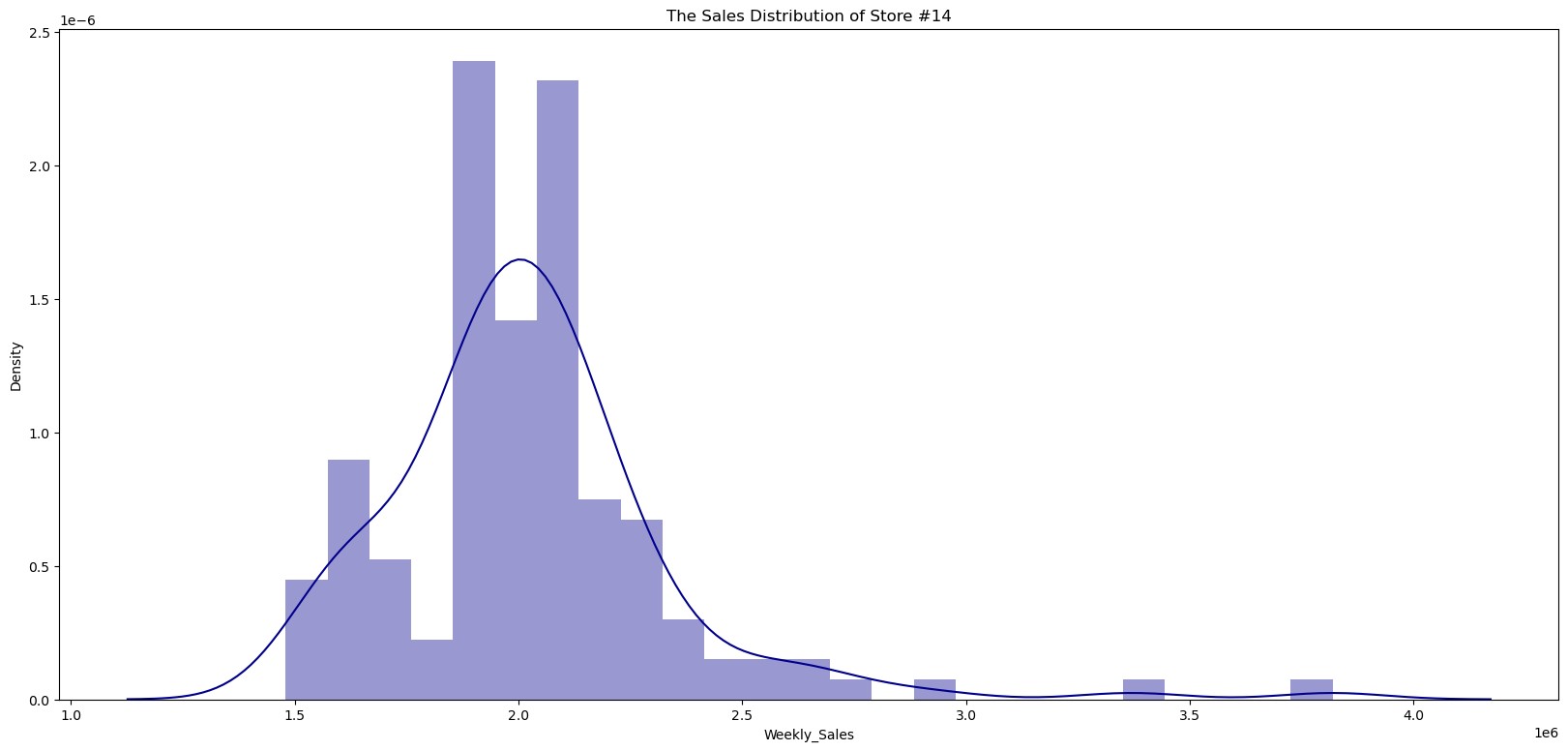
C:\Windows\Temp\ipykernel\_14340\1902952162.py:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[data['Store'] == store\_max\_std]['Weekly\_Sales'], color='darkblu e')



|  |
| --- |
| *# Coefficient of mean to standard deviation*  coef\_mean\_std **=** pd**.**DataFrame(data**.**groupby('Store')['Weekly\_Sales']**.**std() **/** data**.**gro coef\_mean\_std **=** coef\_mean\_std**.**rename(columns**=**{'Weekly\_Sales':'Coefficient of mean t coef\_mean\_std |

In [58]:

# Out[58]: Coefficient of mean to standard deviation Store

**1** 0.100292

|  |  |
| --- | --- |
| **2** | 0.123424 |
| **3** | 0.115021 |
| **4** | 0.127083 |
| **5** | 0.118668 |
| **6** | 0.135823 |
| **7** | 0.197305 |
| **8** | 0.116953 |
| **9** | 0.126895 |
| **10** | 0.159133 |
| **11** | 0.122262 |
| **12** | 0.137925 |
| **13** | 0.132514 |
| **14** | 0.157137 |
| **15** | 0.193384 |
| **16** | 0.165181 |
| **17** | 0.125521 |
| **18** | 0.162845 |

**19** 0.132680

|  |  |
| --- | --- |
| **20** | 0.130903 |
| **21** | 0.170292 |
| **22** | 0.156783 |
| **23** | 0.179721 |
| **24** | 0.123637 |
| **25** | 0.159860 |
| **26** | 0.110111 |
| **27** | 0.135155 |
| **28** | 0.137330 |
| **29** | 0.183742 |
| **30** | 0.052008 |
| **31** | 0.090161 |
| **32** | 0.118310 |
| **33** | 0.092868 |
| **34** | 0.108225 |
| **35** | 0.229681 |
| **36** | 0.162579 |
| **37** | 0.042084 |
| **38** | 0.110875 |
| **39** | 0.149908 |
| **40** | 0.123430 |
| **41** | 0.148177 |
| **42** | 0.090335 |
| **43** | 0.064104 |
| **44** | 0.081793 |
| **45** | 0.165613 |

|  |
| --- |
| *# Distribution of store has maximum coefficient of mean to standard deviation*  store\_max\_coef **=** coef\_mean\_std['Coefficient of mean to standard deviation']**.**idxmax( plt**.**figure(figsize**=**(15, 7)) sns**.**distplot(data[data['Store'] **==** store\_max\_coef]['Weekly\_Sales'], color**=**'darkgree plt**.**title('The Sales Distribution of Store #' **+** str(store\_max\_coef)) plt**.**show() |

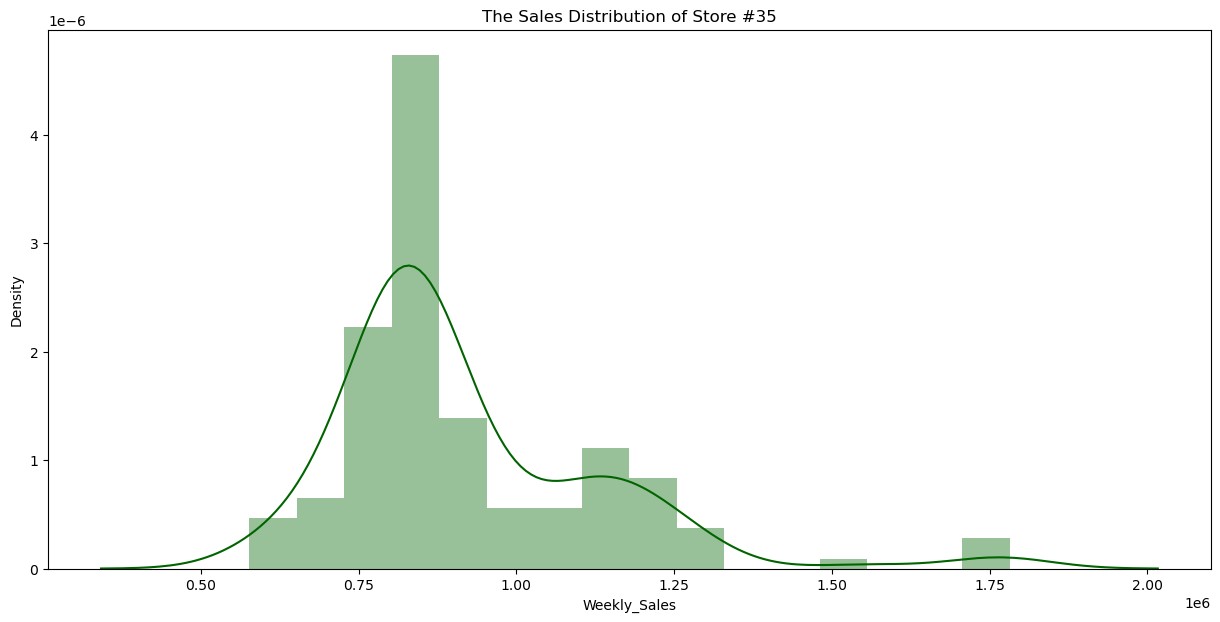
In [59]:

C:\Windows\Temp\ipykernel\_14340\2146623014.py:5: UserWarning: `distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(data[data['Store'] == store\_max\_coef]['Weekly\_Sales'], color='darkgre en')

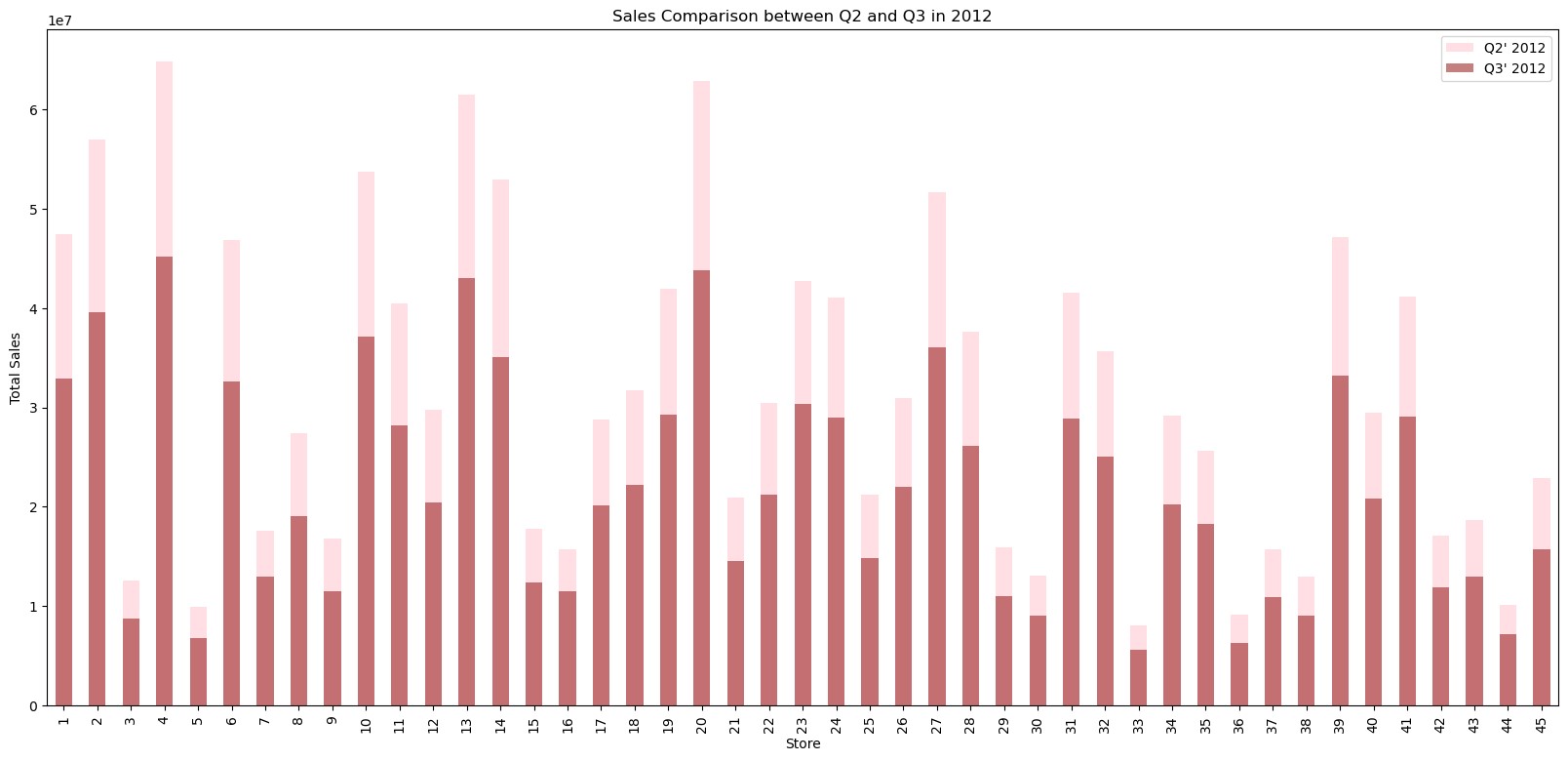


|  |
| --- |
| Q3: Which store**/**s has good quarterly growth rate **in** Q3’2012 |

In [ ]:

|  |
| --- |
| *# Sales for the third quarter in 2012*  Q3 **=** data[(data['Date'] **>** '2012-6-01') **&** (data['Date'] **<** '2012-12-30')]**.**groupby('St  *# Sales for the second quarter in 2012*  Q2 **=** data[(data['Date'] **>** '2012-04-01') **&** (data['Date'] **<** '2012-11-30')]**.**groupby('S  *# Plotting the difference between sales for the second and third quarters* plt**.**figure(figsize**=**(20, 9))  Q2**.**plot(kind**=**'bar', color**=**'pink', alpha**=**0.5) Q3**.**plot(kind**=**'bar', color**=**'darkred', alpha**=**0.5) plt**.**legend(["Q2' 2012", "Q3' 2012"]) plt**.**title('Sales Comparison between Q2 and Q3 in 2012') plt**.**xlabel('Store') plt**.**ylabel('Total Sales') plt**.**show() |

In [68]:



|  |
| --- |
| *# store/s has good quarterly growth rate in Q3’2012 - .sort\_values(by='Weekly\_Sale* print('Store have good quarterly growth rate in Q3’2012 is Store '**+**str(Q3**.**idxmax()) |

In [69]:

|  |
| --- |
| Q4: Some holidays have a negative impact on sales**.** Find out holidays which have hig non**-**holiday season **for** all stores together  **\*\***Holiday Events:**\*\***   * Super Bowl: 12**-**Feb**-**10, 11**-**Feb**-**11, 10**-**Feb**-**12, 8**-**Feb**-**13 * Labour Day: 10**-**Sep**-**10, 9**-**Sep**-**11, 7**-**Sep**-**12, 6**-**Sep**-**13 * Thanksgiving: 26**-**Nov**-**10, 25**-**Nov**-**11, 23**-**Nov**-**12, 29**-**Nov**-**13 * Christmas: 31**-**Dec**-**10, 30**-**Dec**-**11, 28**-**Dec**-**12, 27**-**Dec**-**13 |

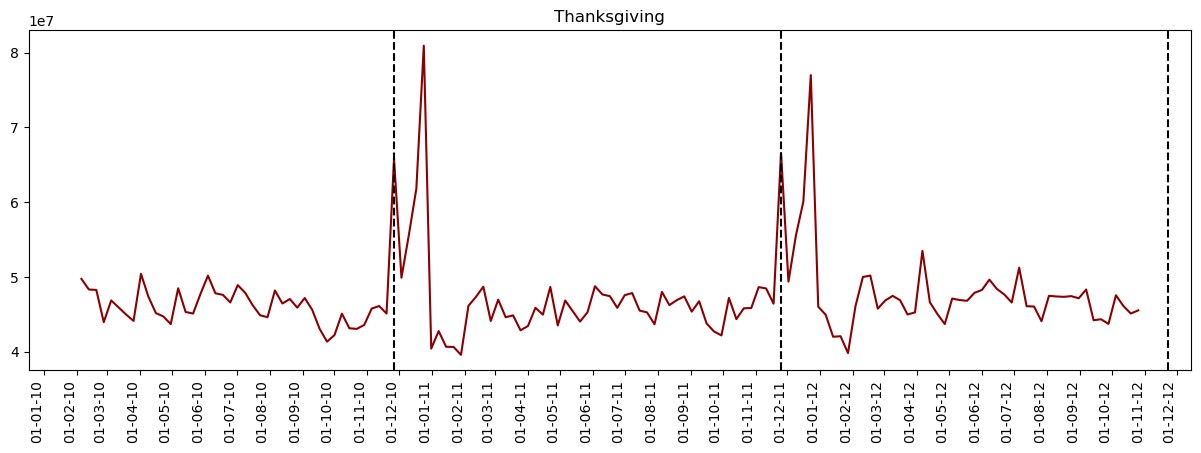
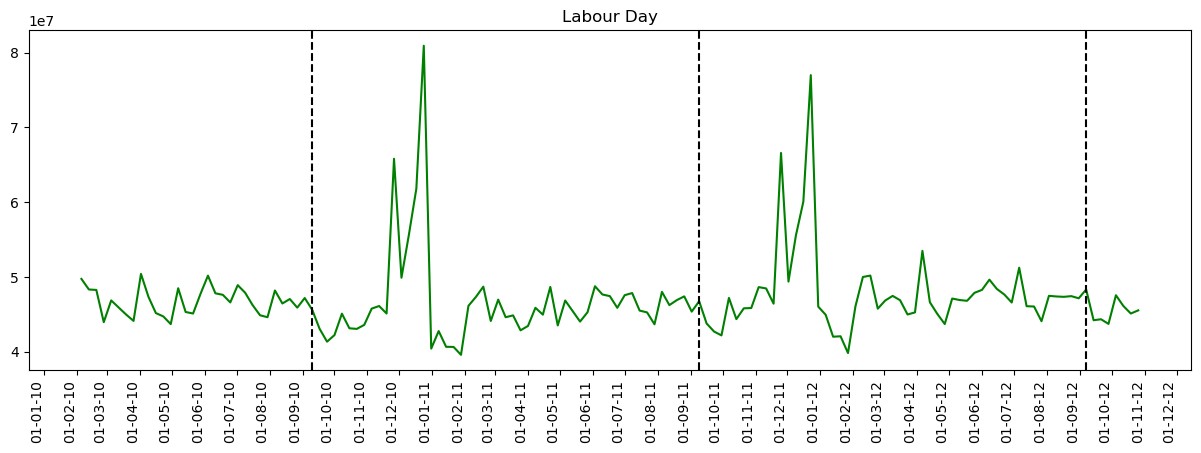
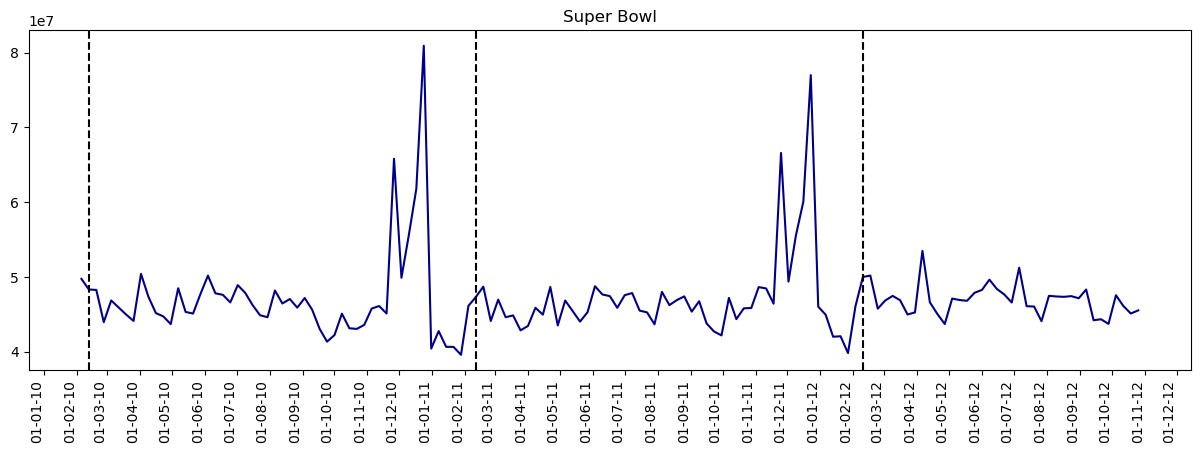
Store have good quarterly growth rate in Q3’2012 is Store 4 With 45191625.95 $ In [ ]:

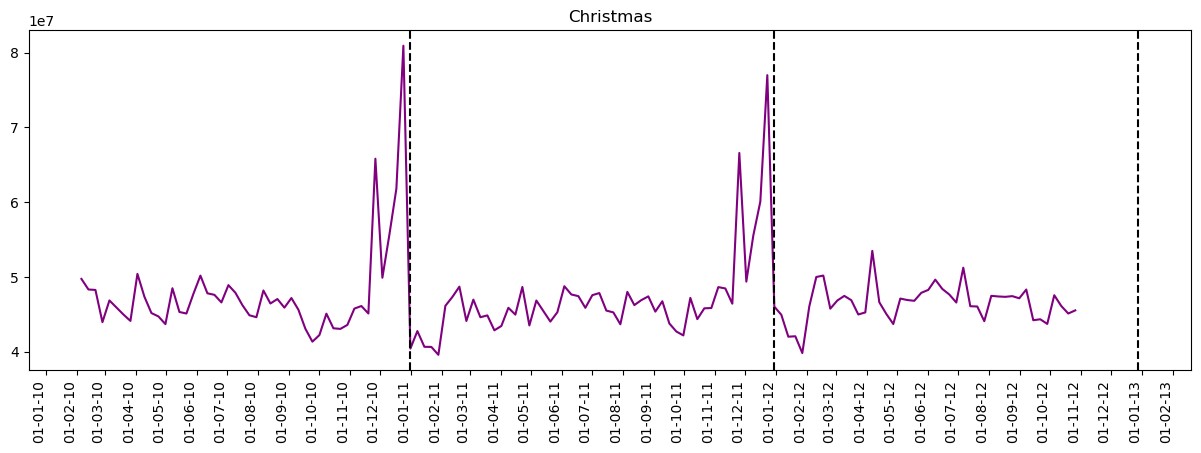
|  |
| --- |
| **def** plot\_line(df, holiday\_dates, holiday\_label, line\_color):  fig, ax **=** plt**.**subplots(figsize**=**(15, 5)) ax**.**plot(df['Date'], df['Weekly\_Sales'], label**=**holiday\_label, c**=**line\_color)  **for** day **in** holiday\_dates:  day **=** datetime**.**strptime(day, '%d-%m-%Y') plt**.**axvline(x**=**day, linestyle**=**'--', c**=**'black')    plt**.**title(holiday\_label) x\_dates **=** df['Date']**.**dt**.**strftime('%Y-%m-%d')**.**sort\_values()**.**unique() xfmt **=** dates**.**DateFormatter('%d-%m-%y') ax**.**xaxis**.**set\_major\_formatter(xfmt) ax**.**xaxis**.**set\_major\_locator(dates**.**DayLocator(1)) plt**.**gcf()**.**autofmt\_xdate(rotation**=**90) plt**.**show()  total\_sales **=** data**.**groupby('Date')['Weekly\_Sales']**.**sum()**.**reset\_index()  Super\_Bowl **=** ['12-2-2010', '11-2-2011', '10-2-2012']  Labour\_Day **=** ['10-9-2010', '9-9-2011', '7-9-2012'] |

In [82]:

Thanksgiving **=** ['26-11-2010', '25-11-2011', '23-11-2012'] Christmas **=** ['31-12-2010', '30-12-2011', '28-12-2012']

plot\_line(total\_sales, Super\_Bowl, 'Super Bowl', 'darkblue') plot\_line(total\_sales, Labour\_Day, 'Labour Day', 'green') plot\_line(total\_sales, Thanksgiving, 'Thanksgiving', 'darkred') plot\_line(total\_sales, Christmas, 'Christmas', 'purple')





|  |
| --- |
| The sales increased during thanksgiving**.** And the sales decreased during christmas**.** |

|  |
| --- |
| **from** datetime **import** datetime  Super\_Bowl\_dates **=** [datetime**.**strptime(date, '%d-%m-%Y')**.**date() **for** date **in** Super\_Bo super\_bowl\_data **=** data[data['Date']**.**isin(Super\_Bowl\_dates)] print(super\_bowl\_data) |

In [ ]: In [84]:

Store Date Weekly\_Sales Holiday\_Flag Temperature Fuel\_Price \ 1 1 2010-02-12 1641957.44 1 38.51 2.548

53 1 2011-02-11 1649614.93 1 36.39 3.022

105 1 2012-02-10 1802477.43 1 48.02 3.409

144 2 2010-02-12 2137809.50 1 38.49 2.548

196 2 2011-02-11 2168041.61 1 33.19 3.022 ... ... ... ... ... ... ...

6202 44 2011-02-11 307486.73 1 30.83 3.034

6254 44 2012-02-10 325377.97 1 33.73 3.116

6293 45 2010-02-12 656988.64 1 27.73 2.773

6345 45 2011-02-11 766456.00 1 30.30 3.239

6397 45 2012-02-10 803657.12 1 37.00 3.640

CPI Unemployment Day Month Year 1 211.242170 8.106 2 12 2010

53 212.936705 7.742 2 11 2011

105 220.265178 7.348 2 10 2012

144 210.897994 8.324 2 12 2010

196 212.592862 8.028 2 11 2011 ... ... ... ... ... ...

6202 127.859129 7.224 2 11 2011

6254 130.384903 5.774 2 10 2012

6293 181.982317 8.992 2 12 2010

6345 183.701613 8.549 2 11 2011

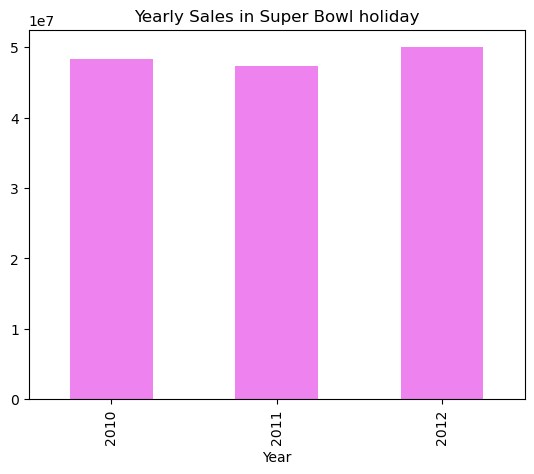
6397 189.707605 8.424 2 10 2012

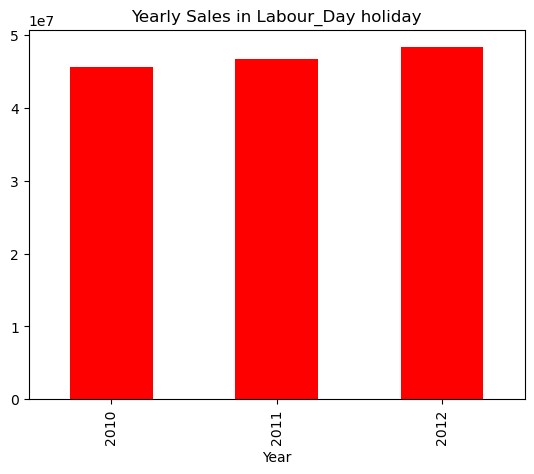
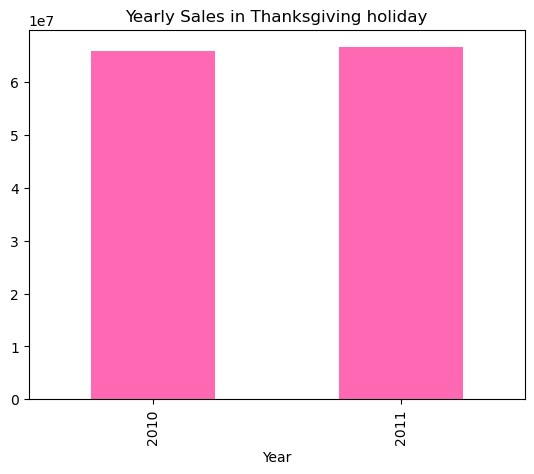
[135 rows x 11 columns]

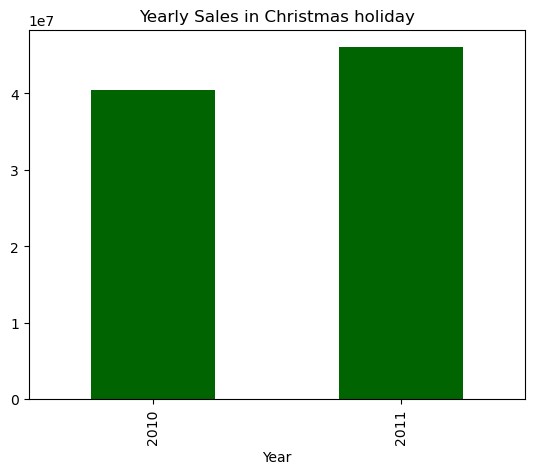
|  |
| --- |
| *# Yearly Sales in holidays*  Super\_Bowl\_df **=** pd**.**DataFrame(data**.**loc[data**.**Date**.**isin(Super\_Bowl)]**.**groupby('Year')[' Thanksgiving\_df **=** pd**.**DataFrame(data**.**loc[data**.**Date**.**isin(Thanksgiving)]**.**groupby('Year Labour\_Day\_df **=** pd**.**DataFrame(data**.**loc[data**.**Date**.**isin(Labour\_Day)]**.**groupby('Year')[' |

In [148…

|  |
| --- |
| Christmas\_df **=** pd**.**DataFrame(data**.**loc[data**.**Date**.**isin(Christmas)]**.**groupby('Year')['We  *# Plotting only if the DataFrame is not empty* **if** **not** Super\_Bowl\_df**.**empty:  Super\_Bowl\_df**.**loc[2010:2012]**.**plot(kind**=**'bar', legend**=False**, title**=**'Yearly Sales  **if** **not** Thanksgiving\_df**.**empty:  Thanksgiving\_df**.**loc[2010:2012]**.**plot(kind**=**'bar', legend**=False**, title**=**'Yearly Sal  **if** **not** Labour\_Day\_df**.**empty:  Labour\_Day\_df**.**loc[2010:2012]**.**plot(kind**=**'bar', legend**=False**, title**=**'Yearly Sales  **if** **not** Christmas\_df**.**empty:  Christmas\_df**.**loc[2010:2012]**.**plot(kind**=**'bar', legend**=False**, title**=**'Yearly Sales plt**.**show() |

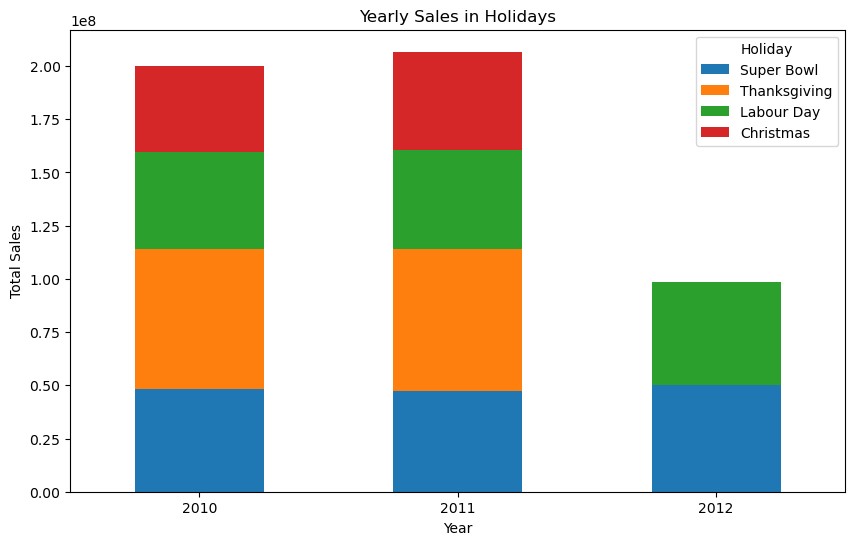






|  |
| --- |
| **import** pandas **as** pd  **import** matplotlib.pyplot **as** plt  *# Yearly Sales in holidays*  Super\_Bowl\_df **=** pd**.**DataFrame(data**.**loc[data**.**Date**.**isin(Super\_Bowl)]**.**groupby('Year')[' Thanksgiving\_df **=** pd**.**DataFrame(data**.**loc[data**.**Date**.**isin(Thanksgiving)]**.**groupby('Year Labour\_Day\_df **=** pd**.**DataFrame(data**.**loc[data**.**Date**.**isin(Labour\_Day)]**.**groupby('Year')[' Christmas\_df **=** pd**.**DataFrame(data**.**loc[data**.**Date**.**isin(Christmas)]**.**groupby('Year')['We  *# Concatenate the DataFrames*  holiday\_sales **=** pd**.**concat([Super\_Bowl\_df, Thanksgiving\_df, Labour\_Day\_df, Christmas holiday\_sales**.**columns **=** ['Super Bowl', 'Thanksgiving', 'Labour Day', 'Christmas']  *# Plotting a stacked bar chart*  holiday\_sales**.**plot(kind**=**'bar', stacked**=True**, figsize**=**(10, 6)) plt**.**title('Yearly Sales in Holidays') plt**.**xlabel('Year') plt**.**ylabel('Total Sales') plt**.**xticks(rotation**=**0) plt**.**legend(title**=**'Holiday') plt**.**show() |

In [96]:

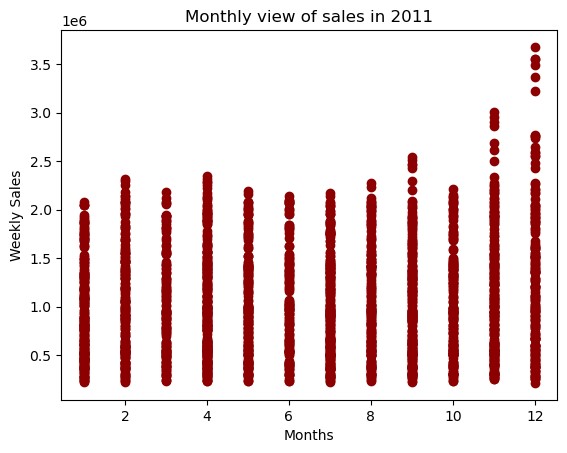
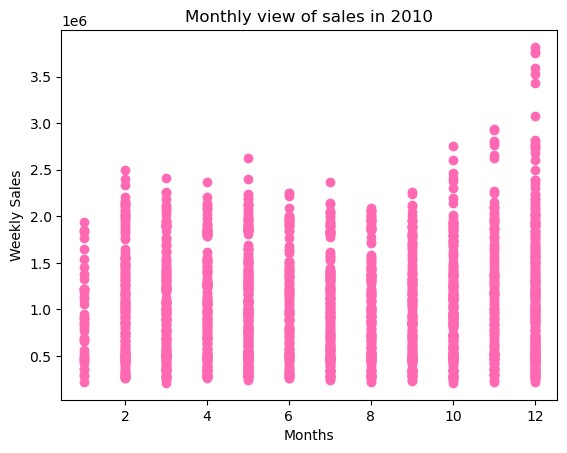


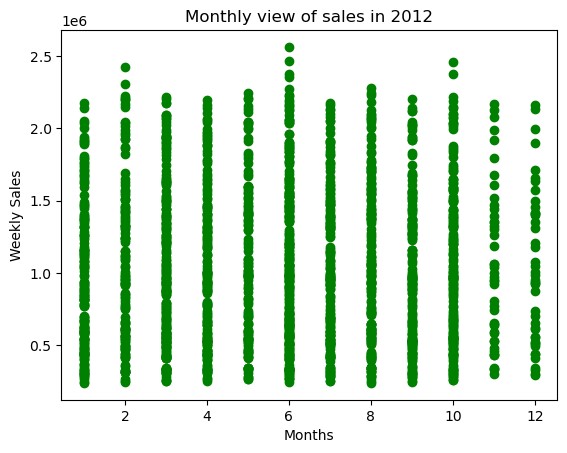
|  |
| --- |
| *### Q5: Provide a monthly and semester view of sales in units and give insights* |

In [ ]:

|  |
| --- |
| *# Monthly view of sales for each years*  plt**.**scatter(data[data**.**Year**==**2010]["Month"], data[data**.**Year**==**2010]["Weekly\_Sales"], plt**.**xlabel("Months") plt**.**ylabel("Weekly Sales") plt**.**title("Monthly view of sales in 2010") plt**.**show()  plt**.**scatter(data[data**.**Year**==**2011]["Month"], data[data**.**Year**==**2011]["Weekly\_Sales"], plt**.**xlabel("Months") plt**.**ylabel("Weekly Sales") plt**.**title("Monthly view of sales in 2011") plt**.**show()  plt**.**scatter(data[data**.**Year**==**2012]["Month"], data[data**.**Year**==**2012]["Weekly\_Sales"], plt**.**xlabel("Months") plt**.**ylabel("Weekly Sales") plt**.**title("Monthly view of sales in 2012") plt**.**show() |

In [100…

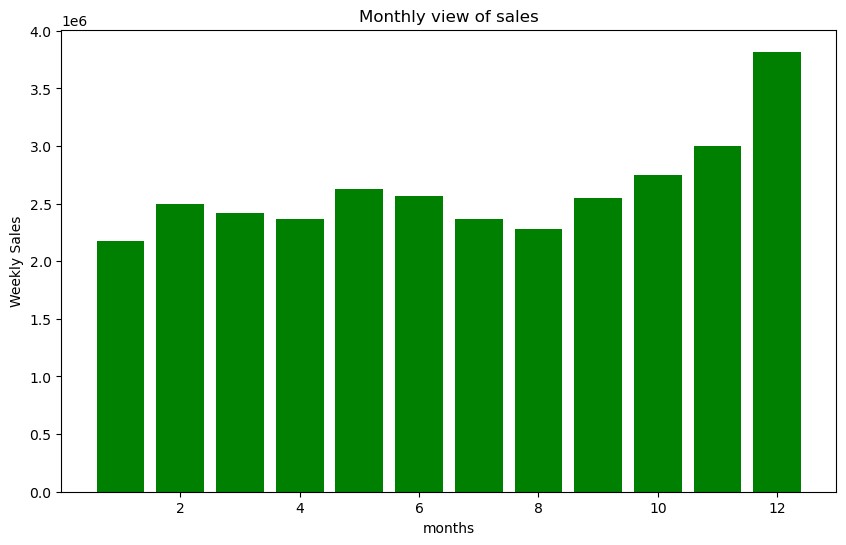




|  |
| --- |
| *# Monthly view of sales for all years* color **=** ['green'] plt**.**figure(figsize**=**(10,6)) plt**.**bar(data["Month"],data["Weekly\_Sales"],color**=**color) plt**.**xlabel("months") plt**.**ylabel("Weekly Sales") plt**.**title("Monthly view of sales") |

In [108…

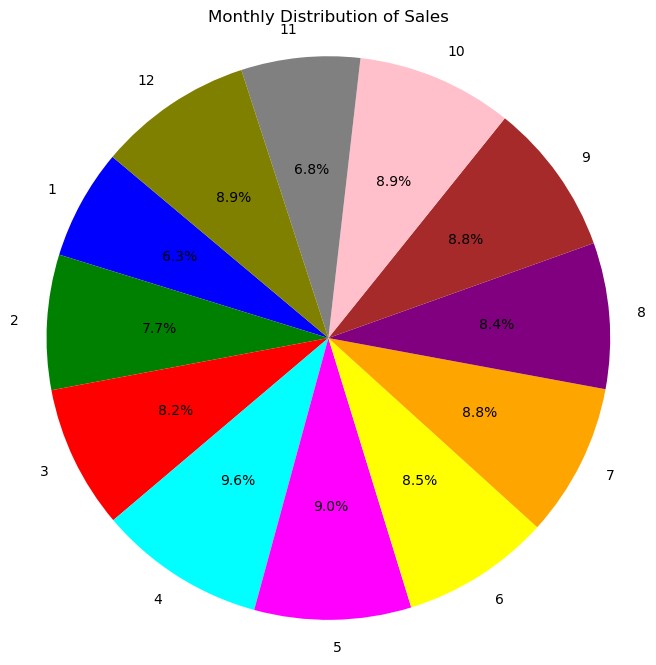
Out[108… Text(0.5, 1.0, 'Monthly view of sales')



|  |
| --- |
| *#Monthly Distribution of Sales in pie chart*  *# Monthly view of sales for all years* plt**.**figure(figsize**=**(15, 8)) colors **=** ['blue', 'green', 'red', 'cyan', 'magenta', 'yellow', 'orange', 'purple', plt**.**bar(data["Month"], data["Weekly\_Sales"], color**=**colors) plt**.**xlabel("Months") plt**.**ylabel("Weekly Sales") plt**.**title("Monthly view of sales")  plt**.**gcf()**.**clear() monthly\_total\_sales **=** data**.**groupby('Month')['Weekly\_Sales']**.**sum() plt**.**figure(figsize**=**(8, 8)) plt**.**pie(monthly\_total\_sales, labels**=**monthly\_total\_sales**.**index, autopct**=**'%1.1f%%', s plt**.**title('Monthly Distribution of Sales') plt**.**axis('equal') plt**.**show() |

In [109…

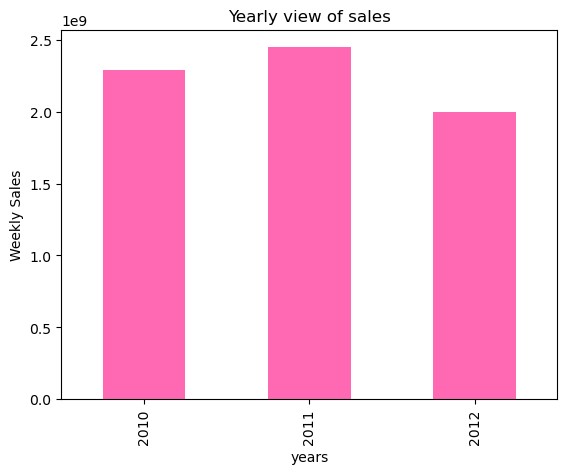
<Figure size 1500x800 with 0 Axes>



|  |
| --- |
| *# Yearly view of sales* color **=** ['hotpink'] plt**.**figure(figsize**=**(10,6)) data**.**groupby("Year")[["Weekly\_Sales"]]**.**sum()**.**plot(kind**=**'bar',legend**=False**,color**=**col plt**.**xlabel("years") plt**.**ylabel("Weekly Sales") plt**.**title("Yearly view of sales"); |

In [110…

<Figure size 1000x600 with 0 Axes>

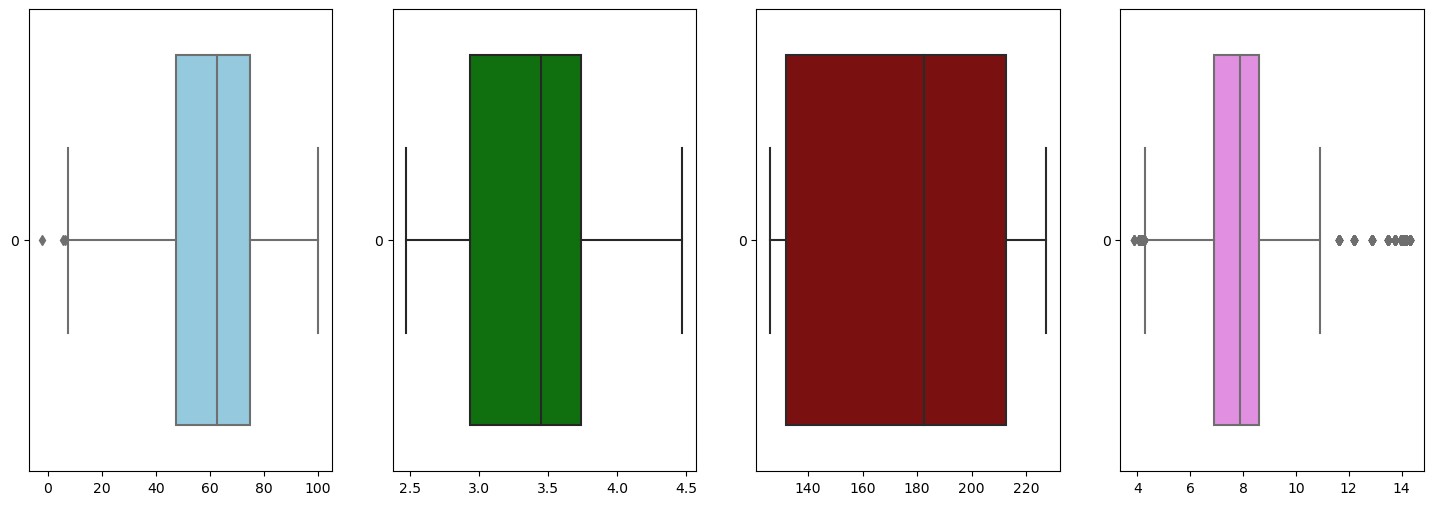


|  |
| --- |
| *## Build prediction models to forecast demand (Modeling)* |

In [ ]:

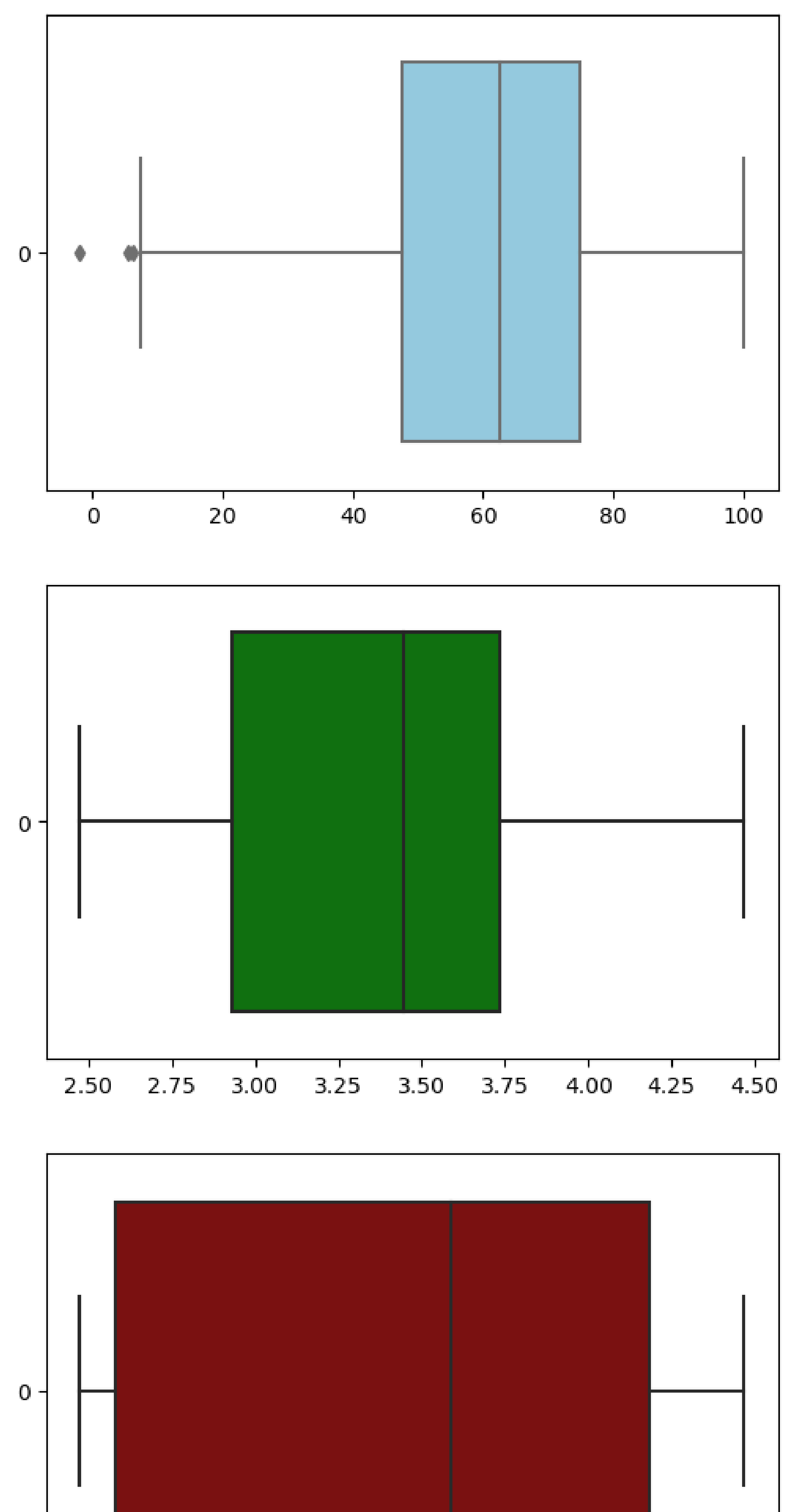
|  |
| --- |
| *# find outliers*  *# Define colors for the boxplots*  colors **=** ['skyblue', 'green', 'darkred', 'violet']  fig, axs **=** plt**.**subplots(1,4, figsize**=**(18, 6))  X **=** data[['Temperature', 'Fuel\_Price', 'CPI', 'Unemployment']] **for** i, column **in** enumerate(X):  sns**.**boxplot(data[column], ax**=**axs[i], color**=**colors[i],orient**=**'h') plt**.**show() |

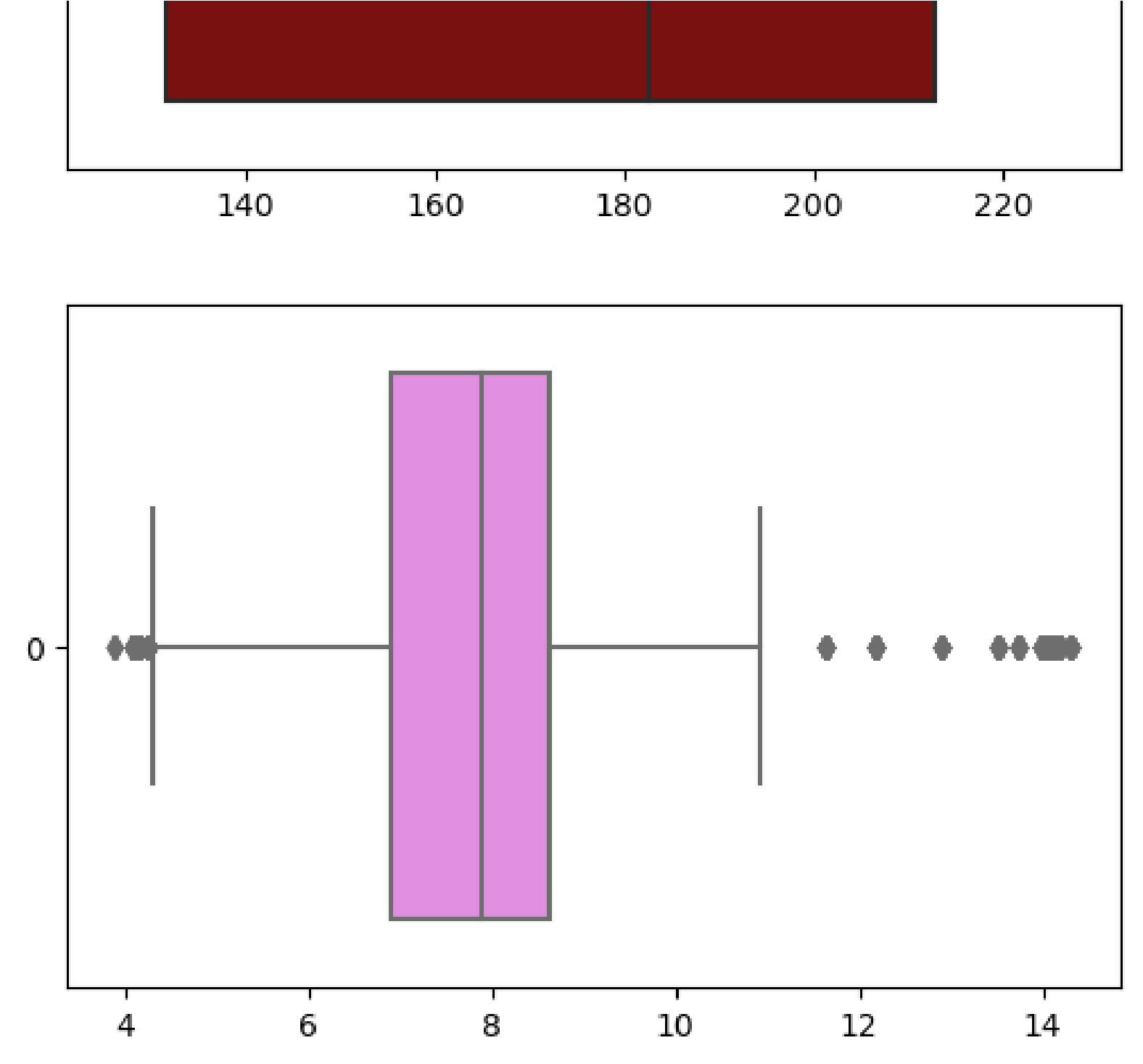
In [117…



|  |
| --- |
| *# find outliers*  colors **=** ['skyblue', 'green', 'darkred', 'violet']  fig, axs **=** plt**.**subplots(4,figsize**=**(6,18))  X **=** data[['Temperature','Fuel\_Price','CPI','Unemployment']] **for** i,column **in** enumerate(X):  sns**.**boxplot(data[column], ax**=**axs[i],color**=**colors[i],orient**=**'h') |

In [124…





|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 1 | 2010-  02-12 | 1641957.44 | 1 | 38.51 | 2.548 | 211. |
| **2** | 1 | 2010-  02-19 | 1611968.17 | 0 | 39.93 | 2.514 | 211. |
| **3** | 1 | 2010-  02-26 | 1409727.59 | 0 | 46.63 | 2.561 | 211. |
| **4** | 1 | 2010-  03-05 | 1554806.68 | 0 | 46.50 | 2.625 | 211. |
| **...** | ... | ... | ... | ... | ... | ... |  |
| **6430** | 45 | 2012-  09-28 | 713173.95 | 0 | 64.88 | 3.997 | 192. |
| **6431** | 45 | 2012-  10-05 | 733455.07 | 0 | 64.89 | 3.985 | 192. |
| **6432** | 45 | 2012-  10-12 | 734464.36 | 0 | 54.47 | 4.000 | 192. |

|  |
| --- |
| *# drop the outliers*  data\_new **=** data[(data['Unemployment']**<**10) **&** (data['Unemployment']**>**4.5) **&** (data['Tem data\_new |

In [120…

# Out[120… Store Date Weekly\_Sales Holiday\_Flag Temperature Fuel\_Price

2010-

**0** 1 1643690.90 0 42.31 2.572 211.

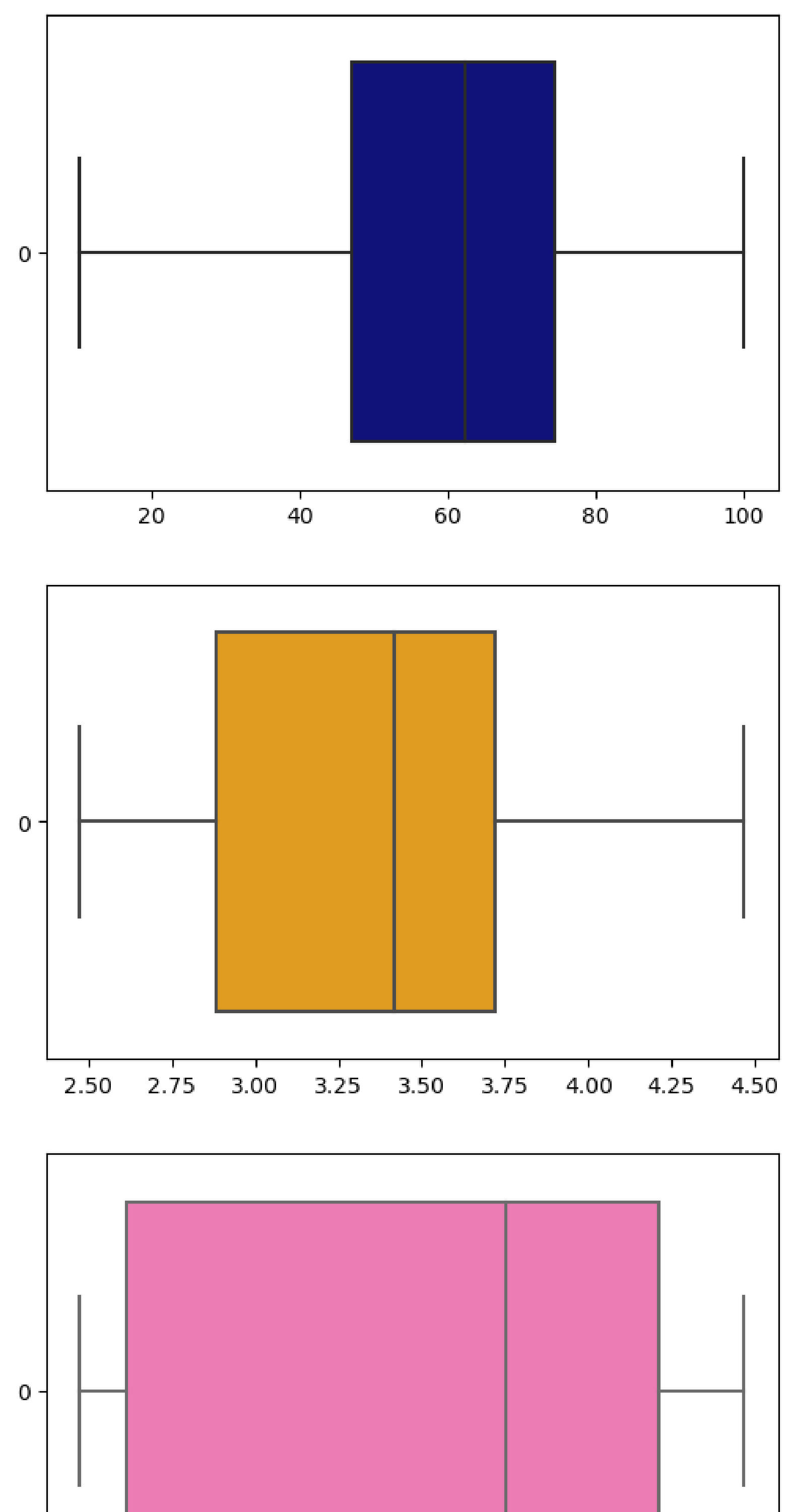
02-05

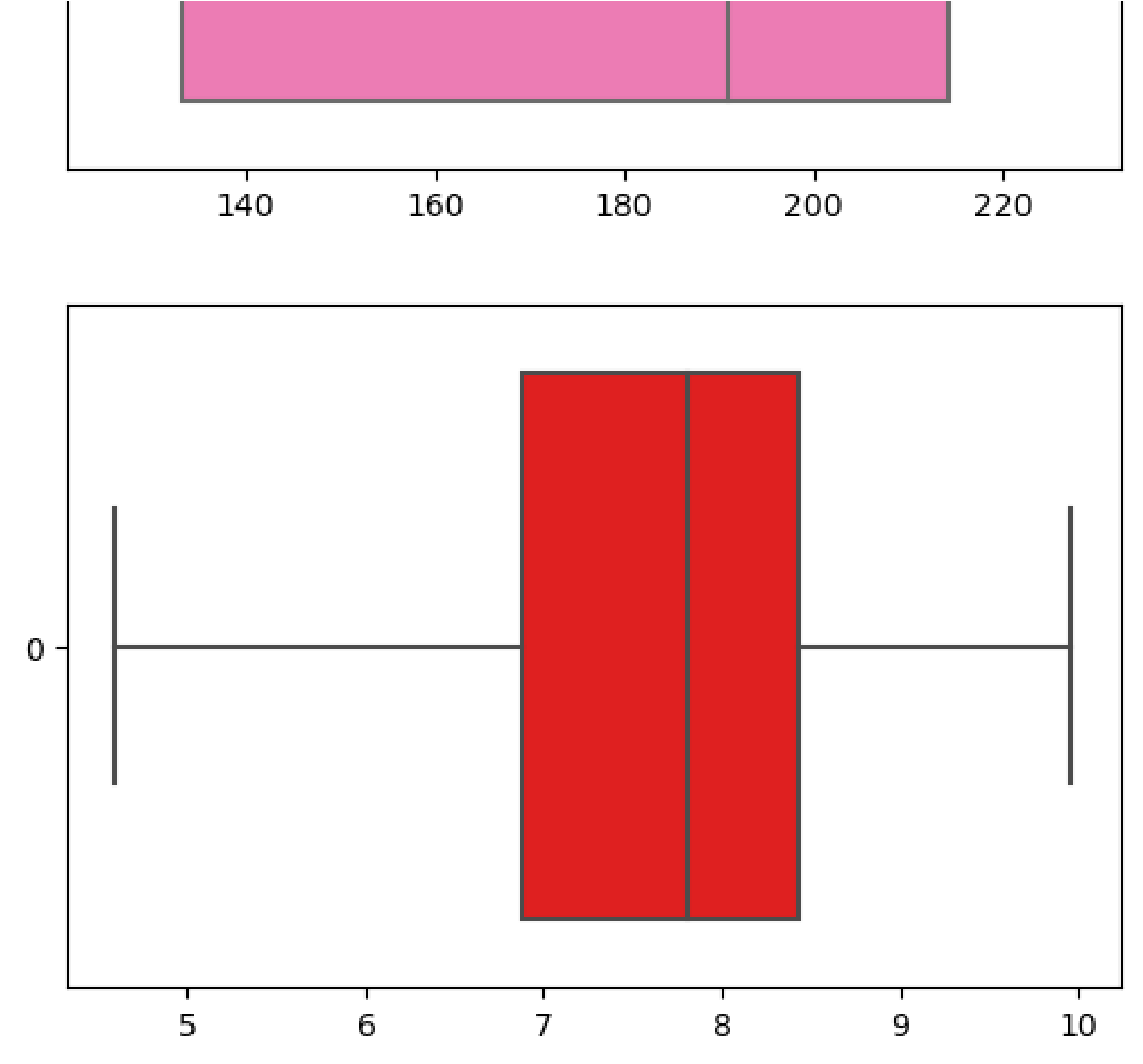
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **6433** | 45 | 2012-  10-19 | 718125.53 | 0 | 56.47 | 3.969 | 192. |
| **6434** | 45 | 2012-  10-26 | 760281.43 | 0 | 58.85 | 3.882 | 192. |

5658 rows × 11 columns

|  |
| --- |
| *# check outliers*  colors **=** ['darkblue', 'orange', 'hotpink', 'red']  fig, axs **=** plt**.**subplots(4,figsize**=**(6,18))  X **=** data\_new[['Temperature','Fuel\_Price','CPI','Unemployment']] **for** i,column **in** enumerate(X):  sns**.**boxplot(data\_new[column], ax**=**axs[i],color**=**colors[i],orient**=**'h') |

In [129…





|  |
| --- |
| *# Build Model* |

In [ ]:

|  |
| --- |
| *# Import sklearn*  **from** sklearn.ensemble **import** RandomForestRegressor **from** sklearn.model\_selection **import** train\_test\_split **from** sklearn **import** metrics  **from** sklearn.linear\_model **import** LinearRegression  *# Select features and target*  X **=** data\_new[['Store','Fuel\_Price','CPI','Unemployment','Day','Month','Year']] y **=** data\_new['Weekly\_Sales']  *# Split data to train and test (0.80:0.20)*  X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X,y,test\_size**=**0.2)  *# Linear Regression model* print('Linear Regression:') print() reg **=** LinearRegression() reg**.**fit(X\_train, y\_train) y\_pred **=** reg**.**predict(X\_test)  print('Accuracy:',reg**.**score(X\_train, y\_train)**\***100)  print('Mean Absolute Error:', metrics**.**mean\_absolute\_error(y\_test, y\_pred)) print('Mean Squared Error:', metrics**.**mean\_squared\_error(y\_test, y\_pred)) print('Root Mean Squared Error:', np**.**sqrt(metrics**.**mean\_squared\_error(y\_test, y\_pred |

In [144… plt**.**figure(figsize**=**(12, 9)) sns**.**scatterplot(x**=**y\_pred, y**=**y\_test, color**=**'darkred') plt**.**xlabel('Predicted Values') plt**.**ylabel('Actual Values') plt**.**title('Scatter Plot of Predicted vs Actual Values') plt**.**show()

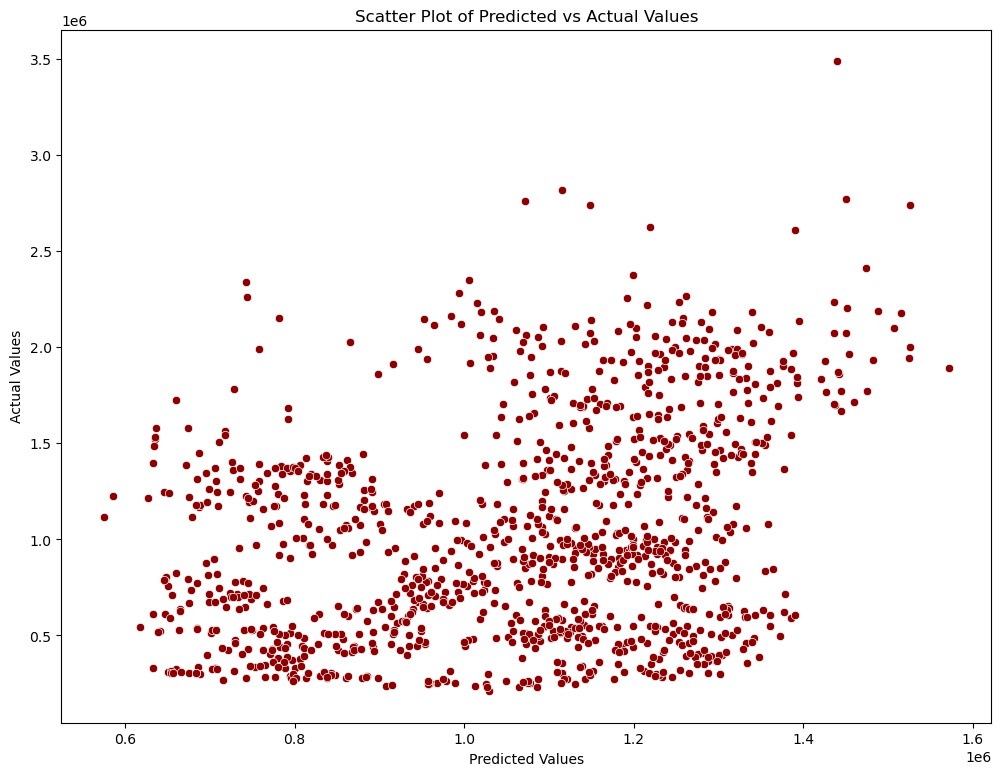
Linear Regression:

Accuracy: 13.405574820484645

Mean Absolute Error: 448698.28179194115

Mean Squared Error: 286286155292.5351

Root Mean Squared Error: 535057.1514264015



|  |
| --- |
| *# Random Forest Regressor* print('Random Forest Regressor:') print() rfr **=** RandomForestRegressor(n\_estimators**=**400, max\_depth**=**15, n\_jobs**=**5) rfr**.**fit(X\_train, y\_train) y\_pred **=** rfr**.**predict(X\_test)  print('Accuracy:', rfr**.**score(X\_test, y\_test) **\*** 100)  print('Mean Absolute Error:', metrics**.**mean\_absolute\_error(y\_test, y\_pred)) print('Mean Squared Error:', metrics**.**mean\_squared\_error(y\_test, y\_pred)) print('Root Mean Squared Error:', np**.**sqrt(metrics**.**mean\_squared\_error(y\_test, y\_pred |

In [147…

plt**.**figure(figsize**=**(10, 8)) sns**.**scatterplot(x**=**y\_pred, y**=**y\_test, color**=**'darkgreen') plt**.**xlabel('Predicted Values') plt**.**ylabel('Actual Values') plt**.**title('Scatter Plot of Predicted vs Actual Values') plt**.**show()

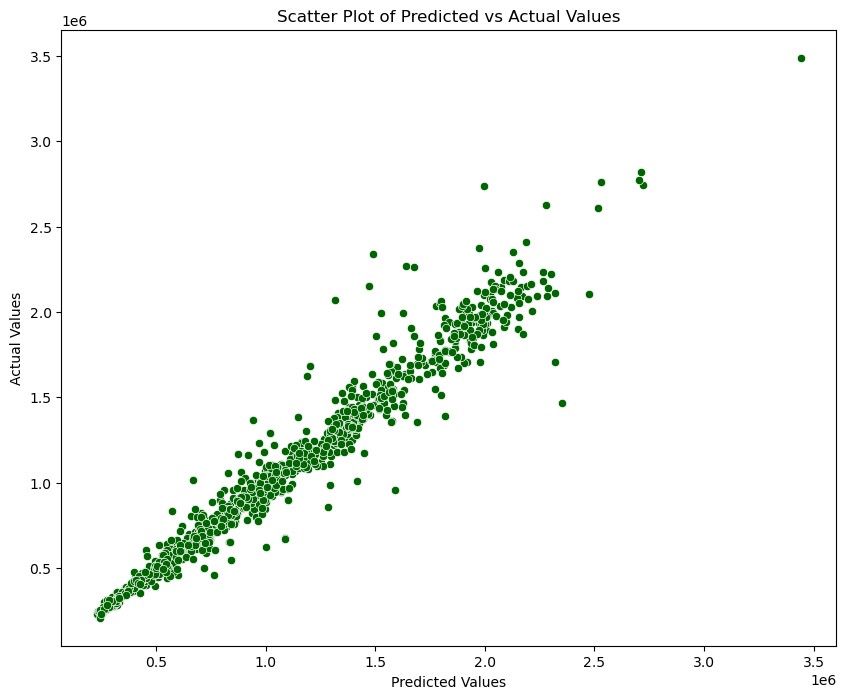
Random Forest Regressor:

Accuracy: 96.16637370882066

Mean Absolute Error: 63176.74643206486

Mean Squared Error: 12277892668.166634

Root Mean Squared Error: 110805.65269049515



|  |
| --- |
|  |

In [ ]: