-------------------Capstone Project---------------

Project Walmart

Table of Contents

1.Problem Statement

2.Project Objective

3.Data Description

4.Data Pre-processing Steps and Inspiration

5.Choosing the Algorithm for the Project

6.Motivation and Reasons for Choosing the Algorithm

7.Assumptions

8.Model Evaluation and Techniques

9.Inferences from the Same

10.Future Possibilities of the Project

11.Conclusion

12.References

1.Problem Statement :
A retail store that has multiple outlets across the country are facing issues in managing the inventory - to match the demand with respect to supply.

2. Project Objective:

> The objective of this project is to predict the weekly sales for different outlets of Walmart based on available historical data.

> We are provided with the weekly sales data for their various outlets. Use statistical analysis, EDA, outlier analysis, and handle the missing values to come up with various insights.

> There is additional information in the dataset about the factors that might influence the sales of a particular week. Factors like Holidays flag, Consumer Price Index (CPI), temperature, fuel price, and unemployment rate have been recorded for each week to try and understand if there is a correlation between the sales of each week and their determinant factors.

> Correlation testing has been performed to understand if there is a correlation between the individual factors and weekly sales and whether such factors have any impact on sales made by Walmart.

>> This study also includes an extensive exploratory data analysis on the provided dataset to understand the following:

>>> If the weekly sales are affected by the unemployment rate, if yes - which stores are suffering the most?

>>> If the weekly sales show a seasonal trend, when and what could be the reason?

>>> Does temperature affect the weekly sales in any manner?

>>> How is the Consumer Price index affecting the weekly sales of various stores?

>>> Top performing stores according to the historical data.

>>> The worst performing store, and how significant is the difference between the highest and lowest performing stores.

> After extensive exploratory data analysis, forecasting the sales for each store for the next 12 weeks using predictive modeling techniques.

3.Data Set:

> The dataset for this study has been acquired in csv format. It contains historic weekly sales information about 45 Walmart stores across different regions in the country.

> The dataset contains weekly sales information from 2010-02-05 to 2012-10-26 about the stores.

> It also contains a column that suggests whether a particular date falls on a holiday or not. In total, there are 6435 rows and 8 columns.

> Another big aspect of this study is to determine whether there is change in the weekly store sales because of changes in temperature, fuel prices, holidays, unemployment, and fluctuations in consumer price indexes, The dataset contains all necessary information about these factors and is used in the analysis to study their impact on sale.

4.Data Preprocessing Steps And Inspiration:

> The Numpy, Pandas, Matplotlib, and seaborn library used with python inspects crucial components of a dataframe under study. For this study, it was essential to get a sense of the datasets before they were used to create complex models. This package checks, compares, and visualizes the different components associated with the above-mentioned datasets. It gives a brief visualized column-wise summary about missing or out of range values, distribution of values, types of columns, the correlation between numeric columns, etc.

The preprocessing of the data included the following steps:

1. First step to import the labriery
2. Second step to read the data file Walmart DataSet.csv.
3. Now check the over all information of Data set.
4. Check data inforamation & shape, duplicated, isnull etc.

```
# Import necessary libraries:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Read the data file Walmart DataSet.csv
df_walmart = pd.read_csv('Walmart DataSet.csv')
df_walmart
```

	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

```
# Checking Data set info.
df_walmart.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
```

```
# Here its observed that, the data type of Date column is Object...
# We need to change it to Date time format.
df_walmart['Date']=pd.to_datetime(df_walmart['Date'])
```

```
<ipython-input-5-4f92bb719be1>:3: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This
  df_walmart['Date']=pd.to_datetime(df_walmart['Date'])
```

```
df_walmart
```

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

6435 rows × 8 columns

```
df_walmart.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   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
dtypes: datetime64[ns](1), float64(5), int64(2)
memory usage: 402.3 KB
```

```
df_walmart.index=df_walmart['Date']
```

```
df_walmart
```

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

6435 rows × 8 columns

```
del df_walmart['Date']
```

```
df_walmart
```

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

6435 rows × 7 columns

```
df_walmart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 6435 entries, 2010-05-02 to 2012-10-26
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Store         6435 non-null   int64  
 1   Weekly_Sales  6435 non-null   float64
 2   Holiday_Flag  6435 non-null   int64  
 3   Temperature   6435 non-null   float64
 4   Fuel_Price    6435 non-null   float64
 5   CPI           6435 non-null   float64
 6   Unemployment  6435 non-null   float64
dtypes: float64(5), int64(2)
memory usage: 402.2 KB
```

```
df_walmart.head()
```

Date	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
2010-05-02	1	1643690.90	0	42.31	2.572	211.096358	8.106
2010-12-02	1	1641957.44	1	38.51	2.548	211.242170	8.106
2010-02-19	1	1611968.17	0	39.93	2.514	211.289143	8.106
2010-02-26	1	1409727.59	0	46.63	2.561	211.319643	8.106
2010-05-03	1	1554806.68	0	46.50	2.625	211.350143	8.106

```
df_walmart.tail()
```

Date	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
2012-09-28	45	713173.95	0	64.88	3.997	192.013558	8.684
2012-05-10	45	733455.07	0	64.89	3.985	192.170412	8.667
2012-12-10	45	734464.36	0	54.47	4.000	192.327265	8.667
2012-10-19	45	718125.53	0	56.47	3.969	192.330854	8.667
2012-10-26	45	760281.43	0	58.85	3.882	192.308899	8.667

```
# Now lets check the null value if any.
df_walmart.isnull().sum()
```

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

```
df_walmart.dtypes
```

```
Store             int64
Weekly_Sales    float64
Holiday_Flag      int64
Temperature     float64
Fuel_Price      float64
CPI             float64
Unemployment    float64
dtype: object
```

```
# Checking duplicate value if any.
print(df_walmart.duplicated().sum())
```

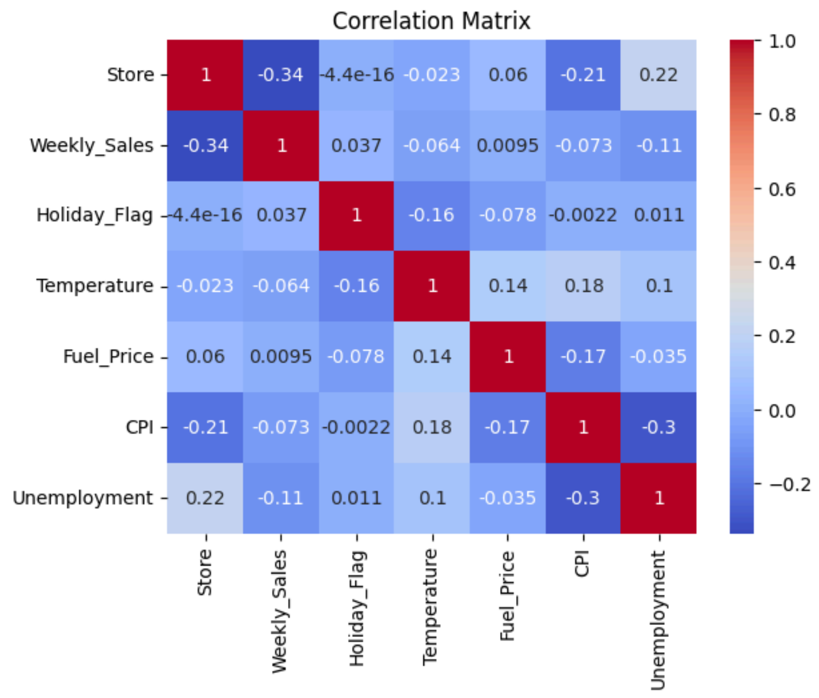
```
0
```

The Temperature, Fuel_Price, CPI, Holiday_Flag, and Unemployment columns in the dataset provide information about external factors that may impact sales.

Analyzing the relationship between these factors and sales can help stores to better understand their customer base and adjust their inventory accordingly.

Now lets calculate the correlation coefficients between each external factor and weekly sales to quantify the strength of the relationship.

```
corr = df_walmart.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
Store	1	-0.34	-4.4e-16	-0.023	0.06	-0.21	0.22
Weekly_Sales	-0.34	1	0.037	-0.064	0.0095	-0.073	-0.11
Holiday_Flag	-4.4e-16	0.037	1	-0.16	-0.078	-0.0022	0.011
Temperature	-0.023	-0.064	-0.16	1	0.14	0.18	0.1
Fuel_Price	0.06	0.0095	-0.078	0.14	1	-0.17	-0.035
CPI	-0.21	-0.073	-0.0022	0.18	-0.17	1	-0.3
Unemployment	0.22	-0.11	0.011	0.1	-0.035	-0.3	1

```
# Calculate the correlation coefficients between each external factor and weekly sales
corr_sales = df_walmart[['Weekly_Sales', 'Temperature', 'Fuel_Price', 'CPI', 'Unemployment']].corr()['Weekly_Sales']
print(corr_sales)
```

```
Weekly_Sales    1.000000
Temperature    -0.063810
Fuel_Price      0.009464
CPI            -0.072634
Unemployment   -0.106176
Name: Weekly_Sales, dtype: float64
```

> the coorelation coefficient derived, gives inference:
>
> > > A plot of the correlation matrix of the dataset. This plot shows the strength and direction of the relationships between variables.
> > >
> > > There seems to be a negative correlation between weekly sales and temperature, unemployment, CPI, and fuel price. This could suggest that sales are not impacted by these external factors.
> > >
> > > There seems to be positive correlation between fuel price and sales, but 0.009464 suggest that very negligible impact by the fuel price.

```
Total_Weekly_sales = df_walmart.Weekly_Sales.groupby(df_walmart.index).sum()
Total_Weekly_sales
```

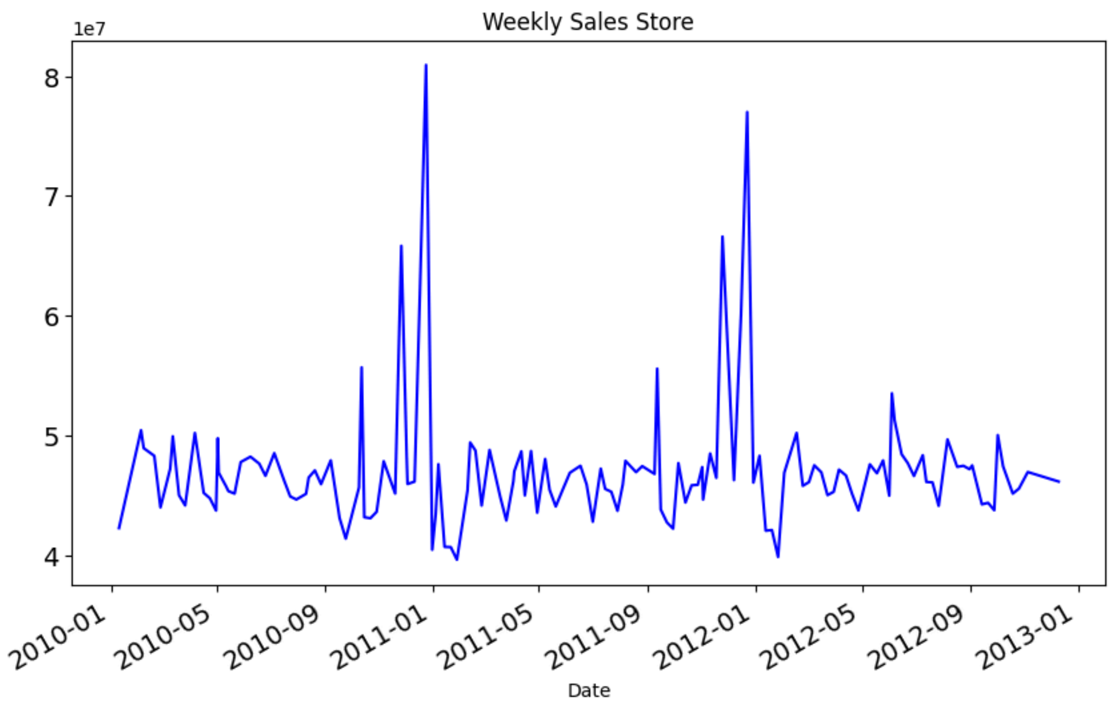
```
Date
2010-01-10    42239875.87
2010-02-04    50423831.26
2010-02-07    48917484.50
2010-02-19    48276993.78
2010-02-26    43968571.13
                 ...
2012-10-08    47403451.04
2012-10-19    45122410.57
2012-10-26    45544116.29
2012-11-05    46925878.99
2012-12-10    46128514.25
Name: Weekly_Sales, Length: 143, dtype: float64
```

```
Total_Weekly_sales.nlargest(5)
```

```
Date
2010-12-24    80931415.60
2011-12-23    76998241.31
```

```
2011-11-25    66593605.26
2010-11-26    65821003.24
2010-12-17    61820799.85
Name: Weekly_Sales, dtype: float64
```

```
# if we plot total weekly sales visualization, we can see some spikes in total sales in some weeaks.
#Week{2010-12-24,2011-12-23,2011-11-25,2010-11-26,2010-12-17} have some spikes in sales.

df=pd.DataFrame(Total_Weekly_sales)
df.Weekly_Sales.plot(figsize=(10,6), title= 'Weekly Sales Store', fontsize=14, color = 'blue')
plt.show()
```



```
Stores_total_sales = df_walmart.groupby('Store')['Weekly_Sales'].sum()
Stores_total_sales
```

```
Store
1     2.224028e+08
2     2.753824e+08
3     5.758674e+07
4     2.995440e+08
5     4.547569e+07
6     2.237561e+08
7     8.159828e+07
8     1.299512e+08
9     7.778922e+07
10    2.716177e+08
11    1.939628e+08
12    1.442872e+08
13    2.865177e+08
14    2.889999e+08
15    8.913368e+07
16    7.425243e+07
17    1.277821e+08
18    1.551147e+08
19    2.066349e+08
20    3.013978e+08
21    1.081179e+08
22    1.470756e+08
23    1.987506e+08
24    1.940160e+08
25    1.010612e+08
26    1.434164e+08
27    2.538559e+08
28    1.892637e+08
29    7.714155e+07
30    6.271689e+07
31    1.996139e+08
32    1.668192e+08
33    3.716022e+07
```

```
34    1.382498e+08
35    1.315207e+08
36    5.341221e+07
37    7.420274e+07
38    5.515963e+07
39    2.074455e+08
40    1.378703e+08
41    1.813419e+08
42    7.956575e+07
43    9.056544e+07
44    4.329309e+07
45    1.123953e+08
Name: Weekly_Sales, dtype: float64
```

```
Stores_total_sales.nlargest(5)
```

```
Store
20    3.013978e+08
4     2.995440e+08
14    2.889999e+08
13    2.865177e+08
2     2.753824e+08
Name: Weekly_Sales, dtype: float64
```

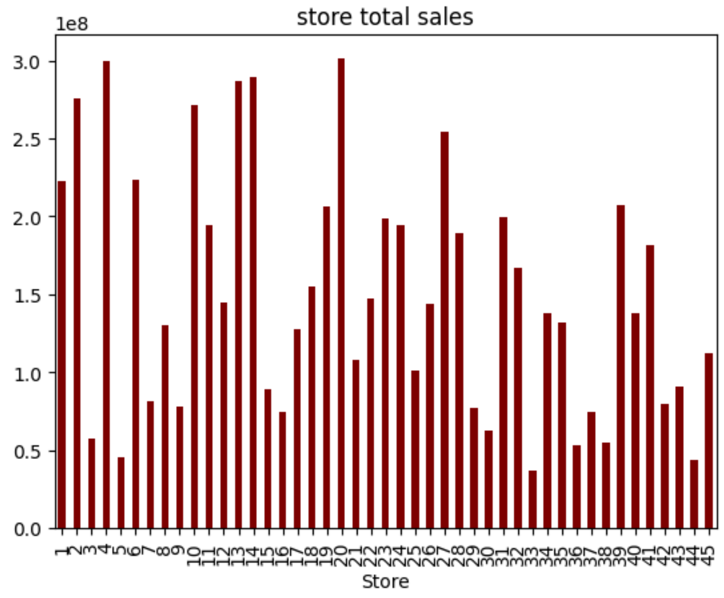
```
Stores_total_sales.nsmallest(5)
```

```
Store
33    37160221.96
44    43293087.84
5     45475688.90
36    53412214.97
38    55159626.42
Name: Weekly_Sales, dtype: float64
```

```
df2=pd.DataFrame(Stores_total_sales)
df2
```

```
       Weekly_Sales
Store
1      2.224028e+08
2      2.753824e+08
3      5.758674e+07
4      2.995440e+08
5      4.547569e+07
6      2.237561e+08
7      8.159828e+07
8      1.299512e+08
9      7.778922e+07
10     2.716177e+08
11     1.939628e+08
12     1.442872e+08
13     2.865177e+08
14     2.889999e+08
15     8.913368e+07
16     7.425243e+07
17     1.277821e+08
18     1.551147e+08
19     2.066349e+08
20     3.013978e+08
21     1.081179e+08
22     1.470756e+08
23     1.987506e+08
24     1.940160e+08
25     1.010612e+08
26     1.434164e+08
27     2.538559e+08
28     1.892637e+08
29     7.714155e+07
30     6.271689e+07
31     1.996139e+08
32     1.668192e+08
33     3.716022e+07
34     1.382498e+08
35     1.315207e+08
36     5.341221e+07
37     7.420274e+07
38     5.515963e+07
39     2.074455e+08
40     1.378703e+08
41     1.813419e+08
42     7.956575e+07
43     9.056544e+07
44     4.329309e+07
45     1.123953e+08
```

```
Stores_total_sales.plot.bar(y='Weekly_Sales', title= 'store total sales',fontsize=10,color= 'maroon')
plt.figure(figsize=(20,6))
plt.ylabel= 'Weekly sales'
plt.xlabel= 'Store'
plt.show()
```



```
<Figure size 2000x600 with 0 Axes>
```

```
# Its is found that, out of 45 stores, top 5 performing store in terms of sales volume is store no-20,4,14,13,2.
# worst performing store is store no-33,44,5,36,38.
# here in bar chart visualization also confirmed.
```

```
#Avg sales storewise:
Avg_sales_stores = df_walmart.groupby('Store')['Weekly_Sales'].mean()
Avg_sales_stores.nlargest()
```

```
Store
20    2.107677e+06
4     2.094713e+06
14    2.020978e+06
13    2.003620e+06
2     1.925751e+06
Name: Weekly_Sales, dtype: float64
```

```
Avg_sales_stores.nsmallest()
```

```
Store
33    259861.692028
44    302748.866014
5     318011.810490
36    373511.992797
38    385731.653287
Name: Weekly_Sales, dtype: float64
```

> > Store 20 is best performing.&
>
> > Store 33 is worst performing, considering both Total sales & Avg Sales parameter.
>
> > There huge & significant difference of sales between store 20 and store 33.
>
> > we can see this by visualization.

```
store_20 = df_walmart[df_walmart.Store == 20]
store_20
```

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
Date							
2010-05-02	20	2401395.47	0	25.92	2.784	204.247194	8.187
2010-12-02	20	2109107.90	1	22.12	2.773	204.385747	8.187
2010-02-19	20	2161549.76	0	25.43	2.745	204.432100	8.187
2010-02-26	20	1898193.95	0	32.32	2.754	204.463087	8.187
2010-05-03	20	2119213.72	0	31.75	2.777	204.494073	8.187
...	...	...	...	...	...	...	...
2012-09-28	20	2008350.58	0	58.65	3.997	215.736716	7.280
2012-05-10	20	2246411.89	0	60.77	3.985	215.925886	7.293
2012-12-10	20	2162951.36	0	47.20	4.000	216.115057	7.293
2012-10-19	20	1999363.49	0	56.26	3.969	216.146470	7.293
2012-10-26	20	2031650.55	0	60.04	3.882	216.151590	7.293

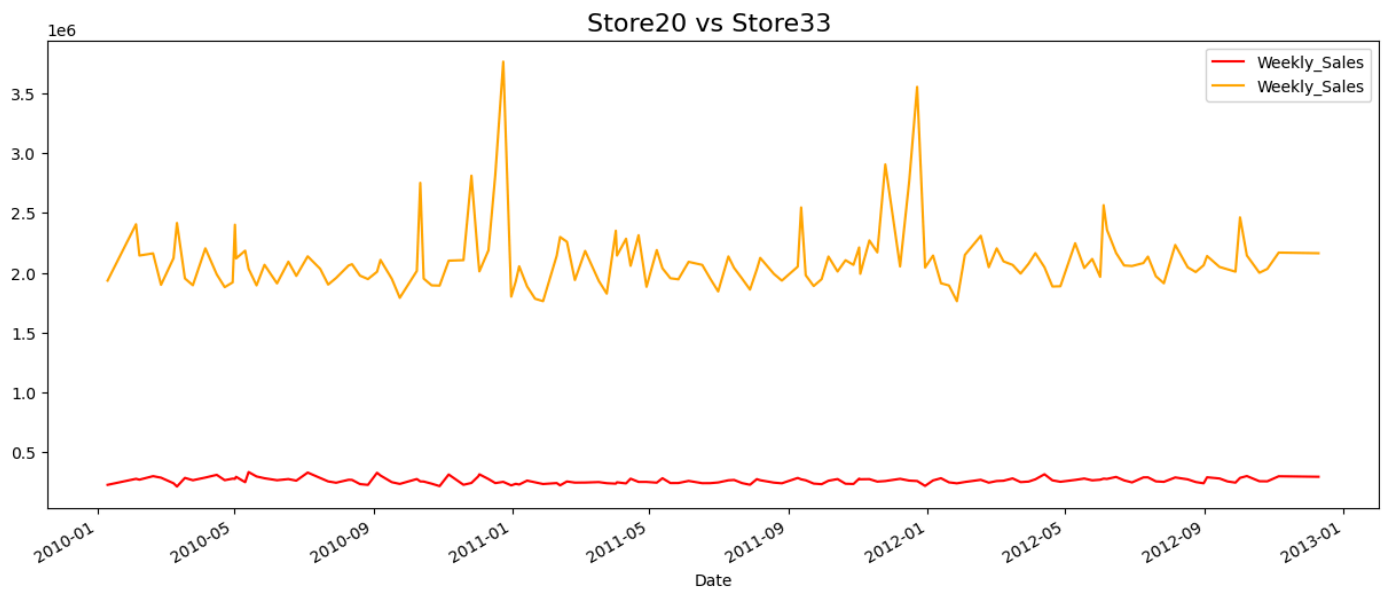
143 rows × 7 columns

```
store_33 = df_walmart[df_walmart.Store == 33]
store_33
```

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
Date							
2010-05-02	33	274593.43	0	58.40	2.962	126.442065	10.115
2010-12-02	33	294882.83	1	55.47	2.828	126.496258	10.115
2010-02-19	33	296850.83	0	62.16	2.915	126.526286	10.115
2010-02-26	33	284052.77	0	56.50	2.825	126.552286	10.115
2010-05-03	33	291484.89	0	59.17	2.877	126.578286	10.115
...	...	...	...	...	...	...	...
2012-09-28	33	242813.51	0	86.42	3.966	131.043000	7.147
2012-05-10	33	265444.90	0	85.18	4.132	131.075667	6.895
2012-12-10	33	291781.15	0	79.64	4.468	131.108333	6.895
2012-10-19	33	254412.34	0	75.55	4.449	131.149968	6.895
2012-10-26	33	253731.13	0	73.70	4.301	131.193097	6.895

143 rows × 7 columns

```
y1=store_33.Weekly_Sales
y2=store_20.Weekly_Sales
y1.plot(figsize=(15, 6),legend=True, color = 'Red')
y2.plot(figsize=(15, 6), legend=True, color = 'Orange')
plt.title('Store20 vs Store33 ', fontsize = '16')
plt.show()
```



> If the weekly sales are affected by the unemployment rate, if yes - which stores are suffering the most?

> Does temperature affect the weekly sales in any manner?

> How is the Consumer Price index affecting the weekly sales of various stores?

```
# To identify the impact of external factors on sales using the Walmart dataset, we can follow these steps:
##Create a multiple regression model to analyze the impact of multiple external factors on weekly sales.
##Plot the correlation matrix of the dataset to visualize the relationships between variables.
##Create scatter plots of the external factors against weekly sales to visualize the relationship.
##Calculate correlation coefficients between external factor and weekly sales to quantify the strength of impact.
```

```
# Calculate the correlation coefficients between each external factor and weekly sales
corr_sales = df_walmart[['Weekly_Sales', 'Temperature', 'Fuel_Price', 'CPI', 'Unemployment']].corr()['Weekly_Sales']
print(corr_sales)
```

```
Weekly_Sales    1.000000
Temperature    -0.063810
Fuel_Price      0.009464
CPI            -0.072634
Unemployment   -0.106176
Name: Weekly_Sales, dtype: float64
```

```
# Create a multiple regression model to analyze the impact of multiple external factors on weekly
from sklearn.linear_model import LinearRegression
a = df_walmart[['Temperature','Fuel_Price','CPI','Unemployment']]
b = df_walmart['Weekly_Sales']
model = LinearRegression().fit(a,b)
r_sq = model.score(a,b)
coefficients = model.coef_
intercept = model.intercept_
print(f"R-squared: {r_sq}")
print(f"Coefficients: {coefficients}")
print(f"Intercept: {intercept}")
```

```
R-squared: 0.024330716534334385
Coefficients: [  -885.66992595 -12248.42446553  -1585.81799199 -41214.98725744]
Intercept: 1743607.6199776107
```

> The regression coefficients of a multiple regression model that analyzes the impact of multiple external factors on weekly sales.

> The R-squared value indicates the proportion of variance in weekly sales that can be explained by the external factors, and the coefficients indicate the strength and direction of the relationship between each factor and sales.

> The multiple regression model that was built to analyze the impact of external factors on weekly sales has an R-squared value of 0.0243, which indicates that only 2.43% of the variance in weekly sales can be explained by the external factors in the model.

> This means that there are other factors that are not included in the model that also have an impact on sales.

> The coefficients of the model represent the strength and direction of the relationship between each external factor and weekly sales.

> The coefficients are as follows:

> > Temperature: -885.67

> > Fuel_Price: -12,248.42

> > CPI: -1,585.82

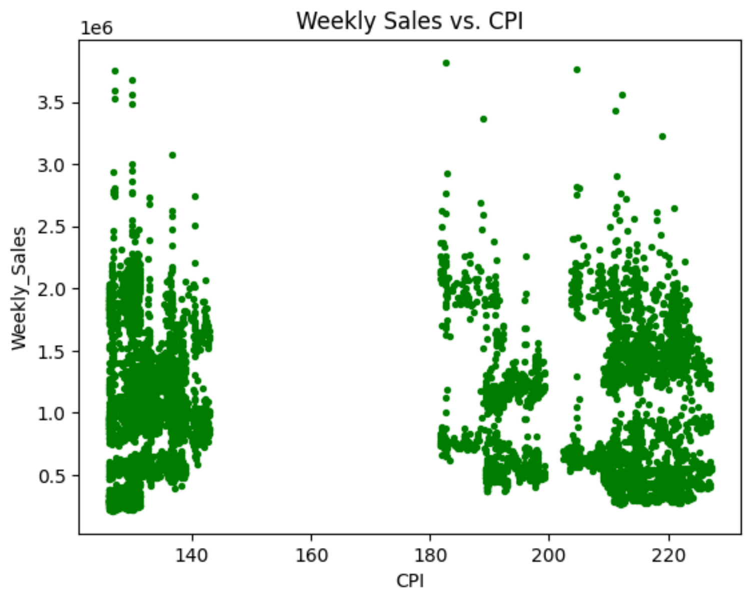
> > Unemployment: -41,214.99

> > > These coefficients indicate that an increase in temperature, fuel price, CPI, and unemployment is associated with a decrease in weekly sales

> > The intercept of the model is 1,743,607.62, which represents the estimated weekly sales when all external factors are at 0.

> > This value is not particularly meaningful in this context because all external factors are unlikely to be 0 in real-world scenarios. However, it is still included in the model to account for the baseline level of weekly sales

```
# Create a scatter plot to visualize the relationship between weekly sales and CPI

df_walmart.plot.scatter(x = 'CPI', y = 'Weekly_Sales', s = 8, color='green');
plt.title('Weekly Sales vs. CPI')
plt.show()
```

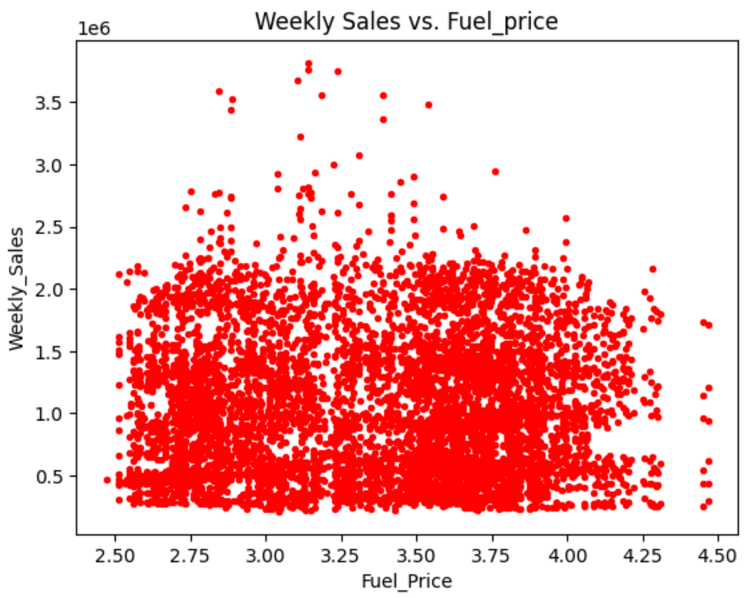


> > According to the US Bureau of Labor Statistics, CPI (Consumer Price Index) is defined as the measure of the average change over time in the prices paid by urban consumers for a market basket of consumer goods and services.

> > In layman's terms, CPI is a measure that assesses the price changes that are associated with the cost of living for each individual. CPI is a great measure for the government when studying inflation and is often used to evaluate and adjust government assistance needs based on income levels and provide wage adjustments with respect to changing cost of living expenses.

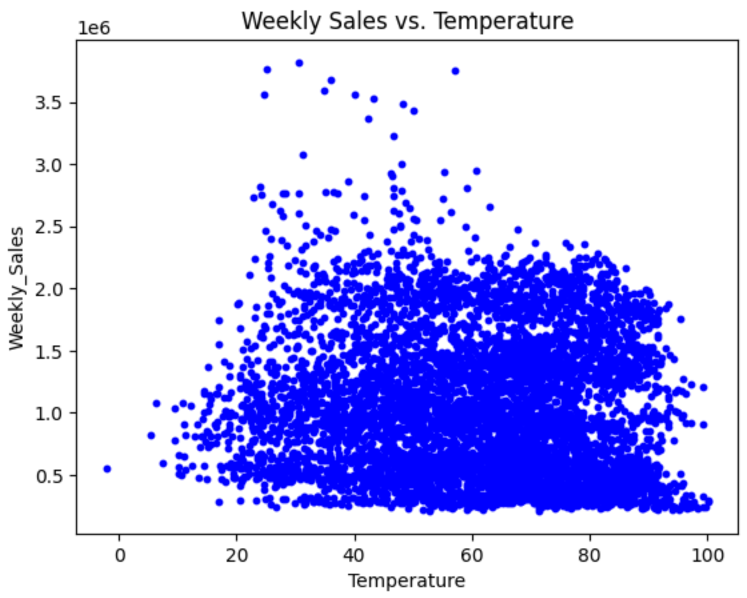
> > A higher CPI generally means that the price of goods has increased and that an individual needs to spend more money to maintain the same standard of living.
> >
> > In our scatter plot above, we can identify three different clusters around different ranges of CPI; while there seems to be no visible relationship between the change in CPI and weekly sales for Walmart stores (sales still occur at high CPI rates).
> >
> > The only negligible observation that can be made is the high amount of sales for some stores when CPI is at a low rate of 140.

```
# Create a scatter plot to visualize the relationship between weekly sales and Fuel_Price
df_walmart.plot.scatter(x = 'Fuel_Price', y = 'Weekly_Sales', s = 8, color='red');
plt.title('Weekly Sales vs. Fuel_price')
plt.show()
```



> > > An economist from Brown University used gasoline price data to study if consumers change their buying behavior based on changes in fuel prices.
> > >
> > > The economist assumes that even a slight increase in fuel prices significantly incresaes the annual expenditure and thus discourages consumers from actively spending.
> > >
> > > This can also slightly be observed in the visualization; while there seems to be a decrease in sales when fuel price is higher than 4.25 dollars, sales are higher when fuel price ranges between 2.75 to 3.75 dollars. While there is no definite pattern that proves this, some observations do support the theory that lower fuel prices encourage higher sales.

```
# Create a scatter plot to visualize the relationship between weekly sales and Temprature
df_walmart.plot.scatter(x = 'Temperature', y = 'Weekly_Sales', s = 10, color='blue');
plt.title('Weekly Sales vs. Temperature')
plt.show()
```



It has widely been known in the retail sector that weather has a profound effect on sales.

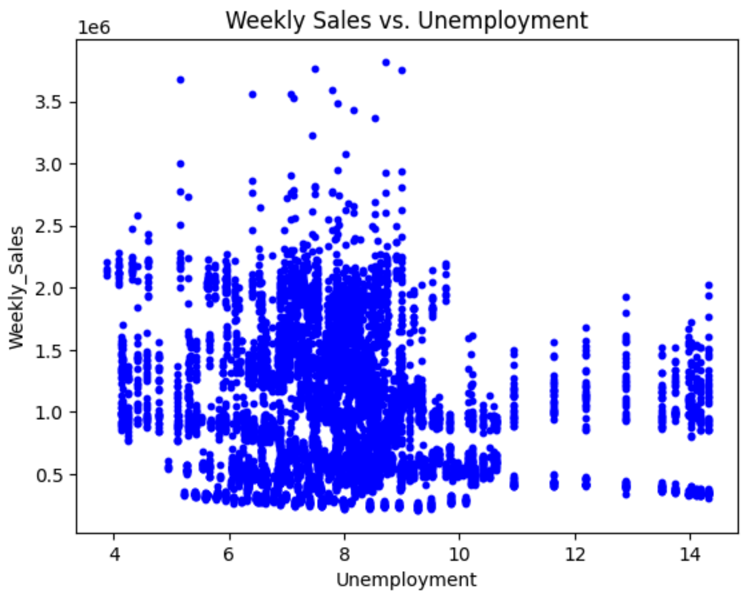
While warmer weather promotes sales, cold/harsh or extremely hot weather is generally not a great encouragement for shoppers to get outdoors and spend money.

Generally speaking, temperatures between 40 to 70 degrees Fahrenheit are considered as favorable for humans to live in considering they are not as hot or cold.

As seen above, the highest sales occur for most store types between the range of 40 to 80 degrees Fahrenheit, thus proving the idea that pleasant weather encourages higher sales.

Sales are relatively lower for very low and very high temperatures but seem to be adequately high for favorable climate conditions.

```
# Create a scatter plot to visualize the relationship between weekly sales and Unemployment
df_walmart.plot.scatter(x = 'Unemployment', y = 'Weekly_Sales', s = 10, color='blue');
plt.title('Weekly Sales vs. Unemployment')
plt.show()
```



Spending sharply drops on the onset of unemployment; a higher unemployment index would generally result in a dip in sales as individuals tend to decrease overall spending.

> > > In our dataset, unemployment is presented through an index of the unemployment rate during that week in the region of the store. From our scatter plot, it is easier to gather the following information:
> > >
> > > > For the given store types, there seems to be a visible decrease in sales when the unemployment index is higher than 11.

FORCASTING:

```
walmart_data=pd.read_csv('Walmart DataSet.csv')
```

```
walmart_data
```

	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

```
walmart_data.head()
```

	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

```
walmart_data.tail()
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
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

```
walmart_data.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
```

```
walmart_data['Date']=pd.to_datetime(walmart_data['Date'], format='%d-%m-%Y')
```

```
walmart_data
```

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

6435 rows × 8 columns

```
walmart_data.index=walmart_data['Date']
```

```
walmart_data
```

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

6435 rows × 8 columns

```
del walmart_data['Date']
```

```
walmart_data
```

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

6435 rows × 7 columns

```
df=walmart_data.drop(['Holiday_Flag','Temperature','Fuel_Price','CPI','Unemployment'], axis=1)
```

```
df
```

Date	Store	Weekly_Sales
2010-02-05	1	1643690.90
2010-02-12	1	1641957.44
2010-02-19	1	1611968.17
2010-02-26	1	1409727.59
2010-03-05	1	1554806.68
...	...	...
2012-09-28	45	713173.95
2012-10-05	45	733455.07
2012-10-12	45	734464.36
2012-10-19	45	718125.53
2012-10-26	45	760281.43

6435 rows × 2 columns

*** Now lets pick the top store-20 and store-14 and forcast its sales:-

```
## Now lets pick the top store-20 and store-14 and forcast its sales.

df_20=df[df['Store']==20]
df_14=df[df['Store']==14]
```

```
df_20.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 143 entries, 2010-02-05 to 2012-10-26
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Store         143 non-null    int64  
 1   Weekly_Sales  143 non-null    float64
dtypes: float64(1), int64(1)
memory usage: 3.4 KB
```

```
df_14.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 143 entries, 2010-02-05 to 2012-10-26
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Store         143 non-null    int64  
 1   Weekly_Sales  143 non-null    float64
dtypes: float64(1), int64(1)
memory usage: 3.4 KB
```

```
df_20
```

	Store	Weekly_Sales
Date		
2010-02-05	20	2401395.47
2010-02-12	20	2109107.90
2010-02-19	20	2161549.76
2010-02-26	20	1898193.95
2010-03-05	20	2119213.72
...	...	...
2012-09-28	20	2008350.58
2012-10-05	20	2246411.89
2012-10-12	20	2162951.36
2012-10-19	20	1999363.49
2012-10-26	20	2031650.55

143 rows × 2 columns

```
#Removing store column from the dataframe.
del df_20['Store']
```

```
df_20.columns
```

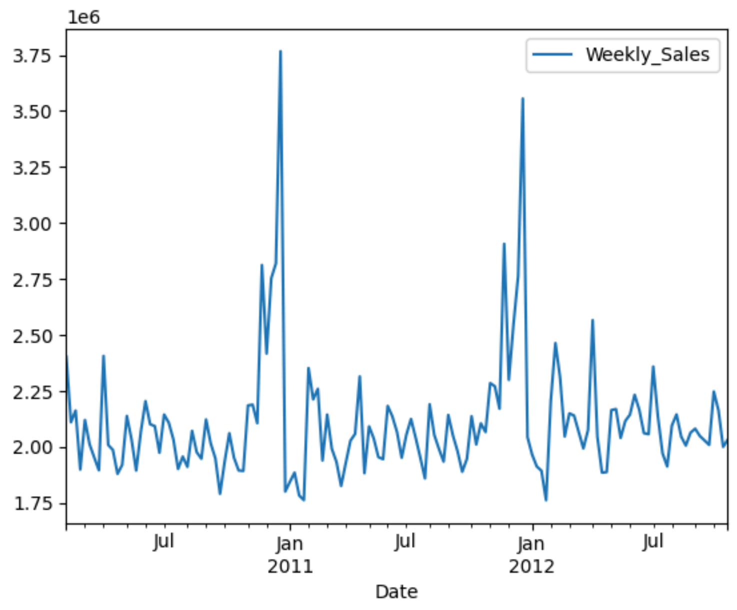
```
Index(['Weekly_Sales'], dtype='object')
```

```
df_20.index
```

```
DatetimeIndex(['2010-02-05', '2010-02-12', '2010-02-19', '2010-02-26',
               '2010-03-05', '2010-03-12', '2010-03-19', '2010-03-26',
               '2010-04-02', '2010-04-09',
               ...
               '2012-08-24', '2012-08-31', '2012-09-07', '2012-09-14',
               '2012-09-21', '2012-09-28', '2012-10-05', '2012-10-12',
               '2012-10-19', '2012-10-26'],
              dtype='datetime64[ns]', name='Date', length=143, freq=None)
```

```
df_20.plot()
```

```
<Axes: xlabel='Date'>
```



```
# Now lets check the data is stationary or not stationary.
# adfuller / kpss
# null hypothesis = data is stationary
# alternative hypo= data not stationary
# adfuller p-value<0.05 --> accept null hypo
```

```
from statsmodels.tsa.stattools import adfuller
```

```
result=adfuller(df_20['Weekly_Sales'])
pvalue=result[1]
# second value is p-value
# the index for second value will be 1 as index start from 0
# checking if data is stationary --- ADF
```

```
pvalue
```

```
3.4912952838128377e-06
```

```
if pvalue<0.05:
  print('we accept null hypothesis- data is stationary')

else:
  print('we reject null hypothesis - data is not stationary')
```

```
we accept null hypothesis- data is stationary
```

```
r=pd.Series(result[0:3],index=['Test Stats','p-value','number of observations used'])
```

```
r
```

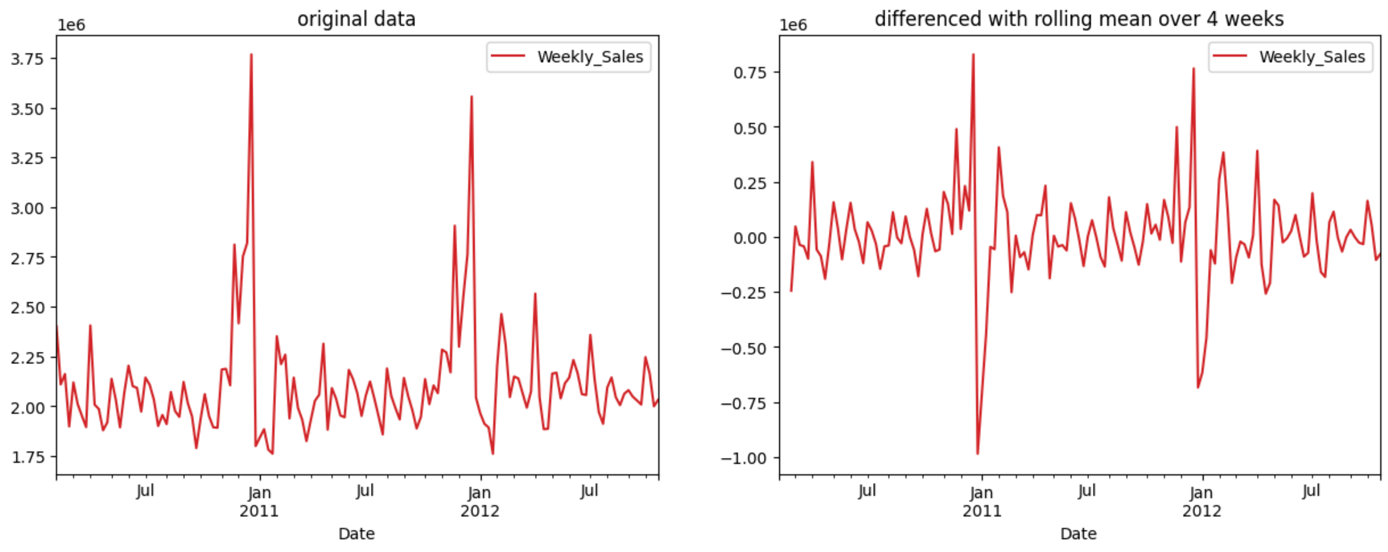
```
Test Stats                     -5.393739
p-value                         0.000003
number of observations used     4.000000
dtype: float64
```

```
rolling_mean=df_20.rolling(window=4).mean()
rolling_mean_detrended=df_20-rolling_mean
```

```
rolling_mean_detrended[:5]
```

	Weekly_Sales
Date	
2010-02-05	NaN
2010-02-12	NaN
2010-02-19	NaN
2010-02-26	-244367.8200
2010-03-05	47197.3875

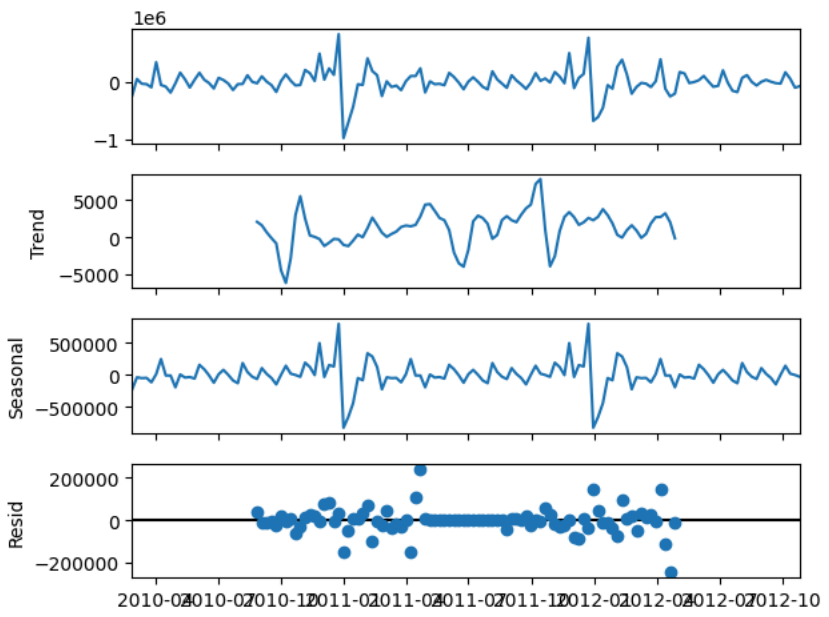
```
ax1=plt.subplot(1,2,2)
rolling_mean_detrended.plot(figsize=(15,5),color='tab:red',
                            title='differenced with rolling mean over 4 weeks',
                            ax=ax1)

ax2=plt.subplot(1,2,1)
df_20.plot(figsize=(15,5),
          color='tab:red',
          title="original data",
          ax=ax2)

plt.show()
```

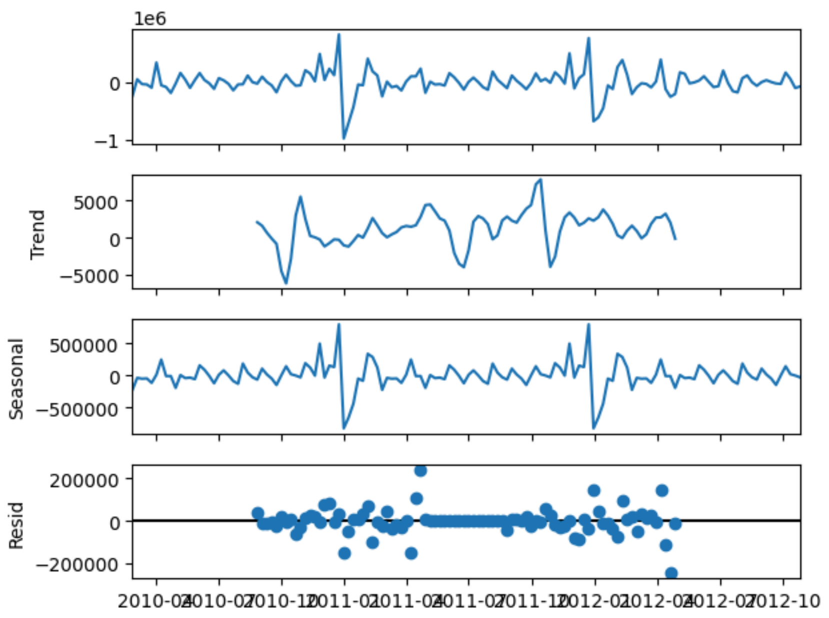


```
##As the data is stationary, we would straight move to seasonal decompose.
from statsmodels.tsa.seasonal import seasonal_decompose
```

```
decompose_result=seasonal_decompose(rolling_mean_detrended.dropna())
```

```
decompose_result.plot()
```





```
rolling_mean_detrended[:5]
```

Date	Weekly_Sales
2010-02-05	NaN
2010-02-12	NaN
2010-02-19	NaN
2010-02-26	-244367.8200
2010-03-05	47197.3875

```
rolling_mean_detrended.shift()[:5]
```

Date	Weekly_Sales
2010-02-05	NaN
2010-02-12	NaN
2010-02-19	NaN
2010-02-26	NaN
2010-03-05	-244367.82

```
rolling_mean_detrended
```

Date	Weekly_Sales
2010-02-05	NaN
2010-02-12	NaN
2010-02-19	NaN
2010-02-26	-244367.8200
2010-03-05	47197.3875
...	...
2012-09-28	-33003.6350
2012-10-05	163586.9675
2012-10-12	51376.0925
2012-10-19	-104905.8400
2012-10-26	-78443.7725

143 rows × 1 columns

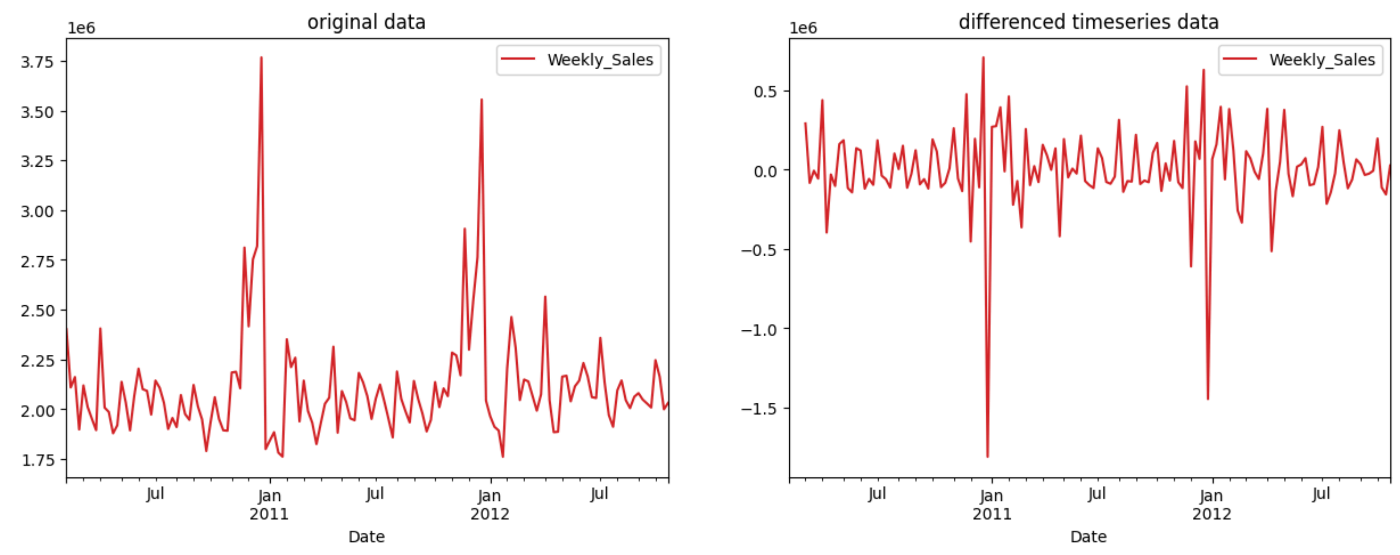
```
rolling_mean_detrended.shift()
```

Date	Weekly_Sales
2010-02-05	NaN
2010-02-12	NaN
2010-02-19	NaN
2010-02-26	NaN
2010-03-05	-244367.8200
...	...
2012-09-28	-26299.7200
2012-10-05	-33003.6350
2012-10-12	163586.9675
2012-10-19	51376.0925
2012-10-26	-104905.8400

143 rows × 1 columns

```
rolling_mean_detrended_diff=rolling_mean_detrended-rolling_mean_detrended.shift()
```

```
ax1=plt.subplot(1,2,2)
rolling_mean_detrended_diff.plot(figsize=(15,5),color='tab:red',
                            title='differenced timeseries data',
                            ax=ax1)

ax2=plt.subplot(1,2,1)
df_20.plot(figsize=(15,5),
          color='tab:red',
          title="original data",
          ax=ax2)

plt.show()
```



```
rolling_mean_detrended_diff=rolling_mean_detrended_diff.dropna()
```

```
rolling_mean_detrended_diff
```

Date	Weekly_Sales
2010-03-05	291565.2075
2010-03-12	-83705.6150
2010-03-19	-6701.0775
2010-03-26	-56242.7300
2010-04-02	439106.8950
...	...
2012-09-28	-6703.9150
2012-10-05	196590.6025
2012-10-12	-112210.8750
2012-10-19	-156281.9325
2012-10-26	26462.0675

139 rows × 1 columns

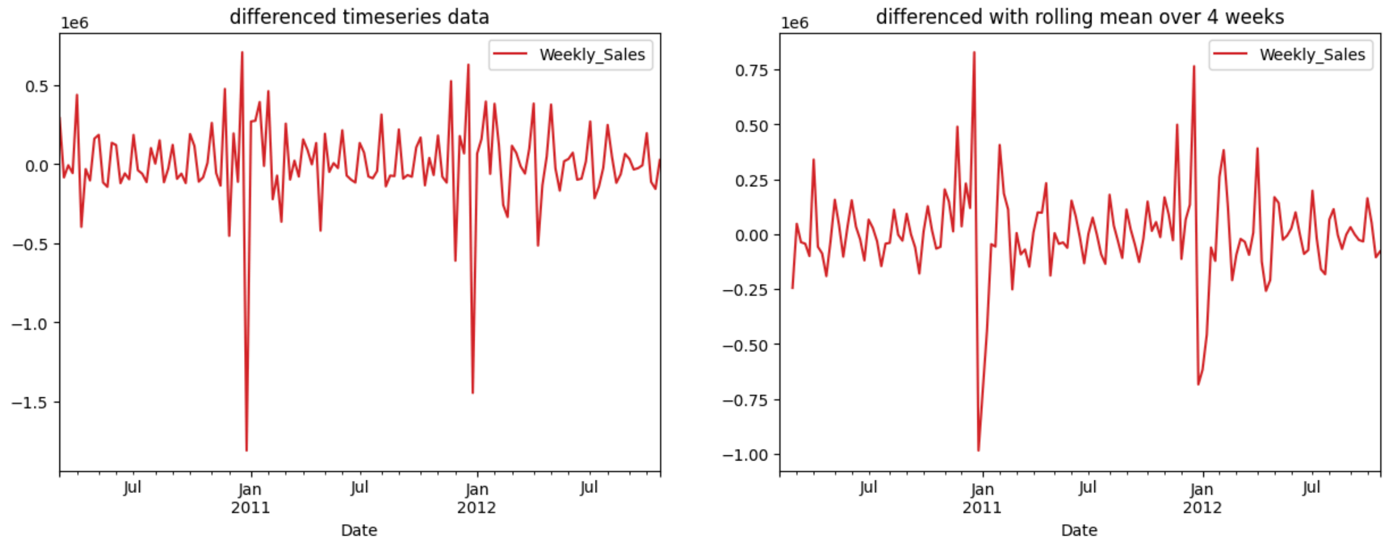
```
result=adfuller(rolling_mean_detrended_diff['Weekly_Sales'])
pvalue=result[1]
# second value is p-value
# the index for second value will be 1 as index start from 0
# checking if data is stationary --- ADF
```

```
if pvalue<0.05:
  print('we accept null hypothesis- data is stationary')

else:
  print('we reject null hypothesis - data is not stationary')
```

```
we accept null hypothesis- data is stationary
```

```
ax1=plt.subplot(1,2,2)
rolling_mean_detrended.plot(figsize=(15,5),color='tab:red',
                            title='differenced with rolling mean over 4 weeks',
                            ax=ax1)

ax2=plt.subplot(1,2,1)
rolling_mean_detrended_diff.plot(figsize=(15,5),color='tab:red',
                            title='differenced timeseries data',
                            ax=ax2)
```

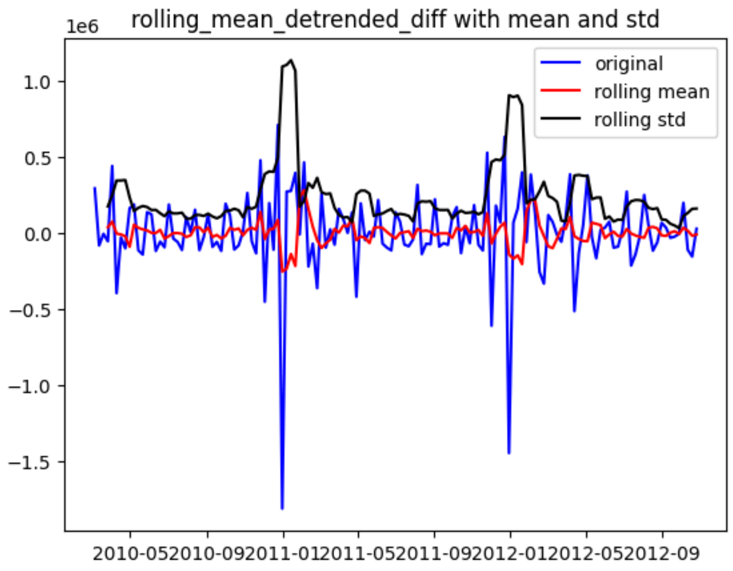
```
<Axes: title={'center': 'differenced timeseries data'}, xlabel='Date'>
```



```
m=rolling_mean_detrended_diff.rolling(window=4).mean()
s=rolling_mean_detrended_diff.rolling(window=4).std()
```

```
plt.plot(rolling_mean_detrended_diff,color='blue',label='original')
plt.plot(m,color='red',label='rolling mean')
plt.plot(s,color='black',label='rolling std')
plt.legend(loc='best')
plt.title('rolling_mean_detrended_diff with mean and std')
```

```
Text(0.5, 1.0, 'rolling_mean_detrended_diff with mean and std')
```



```
!pip install pmdarima
```

```
Requirement already satisfied: pmdarima in /usr/local/lib/python3.10/dist-packages (2.0.4)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.3.2)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (3.0.8)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.23.5)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.5.3)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.2.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.11.4)
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (0.14.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.0.7)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (67.7.2)
Requirement already satisfied: packaging>=17.1 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (23.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2023.3.post1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22->pmdarima) (3.2.
Requirement already satisfied: patsy>=0.5.4 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2->pmdarima) (0.5.6)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.4->statsmodels>=0.13.2->pmdarima) (1.16.0
```

```
from pmdarima import auto_arima
```

```
order=auto_arima(rolling_mean_detrended_diff['Weekly_Sales'])
```

```
order.summary()
```

```
                         SARIMAX Results
  Dep. Variable:   y                 No. Observations: 139
        Model:     SARIMAX(3, 0, 1)  Log Likelihood    -1905.219
         Date:     Sun, 28 Jan 2024        AIC         3822.438
         Time:     07:56:46                BIC         3840.045
       Sample:     03-05-2010             HQIC         3829.593
                   - 10-26-2012
Covariance Type: opg
             coef      std err   z        P>|z|  [0.025    0.975]
intercept 771.0376  890.193  0.866    0.386 -973.708  2515.783
  ar.L1   0.1275    0.066    1.938    0.053 -0.001    0.256
  ar.L2   -0.0624   0.089    -0.702   0.483 -0.237    0.112
  ar.L3   -0.2268   0.098    -2.325   0.020 -0.418    -0.036
  ma.L1   -0.9779   0.052    -18.828  0.000 -1.080    -0.876
 sigma2   5.643e+10 1.11e-05 5.09e+15 0.000 5.64e+10  5.64e+10
   Ljung-Box (L1) (Q):      0.13 Jarque-Bera (JB): 320.92
          Prob(Q):          0.72     Prob(JB):     0.00
Heteroskedasticity (H):     0.77       Skew:       -0.59
  Prob(H) (two-sided):      0.37     Kurtosis:     10.35


Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 2.18e+31. Standard errors may be unstable.
```

```python
from statsmodels.tsa.arima.model import ARIMA
```

```python
train=rolling_mean_detrended_diff.iloc[:120]['Weekly_Sales']
test=rolling_mean_detrended_diff.iloc[120:]['Weekly_Sales']
```

```python
train
```

```
Date
2010-03-05    291565.2075
2010-03-12    -83705.6150
2010-03-19     -6701.0775
2010-03-26    -56242.7300
2010-04-02    439106.8950
                 ...
2012-05-18   -167573.4550
2012-05-25     18645.4725
2012-06-01     33233.6650
2012-06-08     72869.2850
2012-06-15    -98286.3475
Name: Weekly_Sales, Length: 120, dtype: float64
```

```python
test
```

```
Date
2012-06-22    -90971.5225
2012-06-29     17157.4150
2012-07-06    270579.3975
2012-07-13   -215755.1375
2012-07-20   -141905.2300
2012-07-27    -22512.8125
2012-08-03    248841.5075
2012-08-10     47338.3625
2012-08-17   -117906.9025
2012-08-24    -63165.3725
2012-08-31     65148.6675
2012-09-07     33976.5825
2012-09-14    -33301.2700
2012-09-21    -25174.1925
2012-09-28     -6703.9150
2012-10-05    196590.6025
2012-10-12   -112210.8750
2012-10-19   -156281.9325
2012-10-26     26462.0675
Name: Weekly_Sales, dtype: float64
```

```python
model=ARIMA(train,order=(3,0,1))
model_fit=model.fit()
```

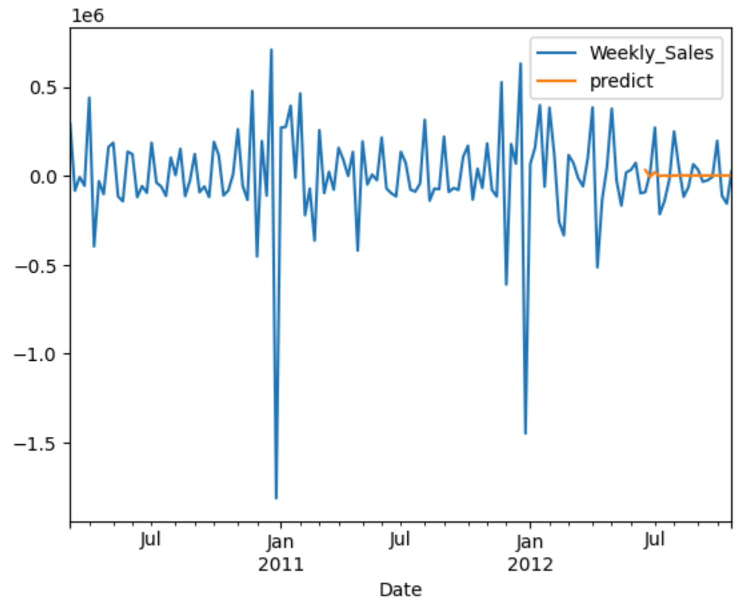
```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so i
  self._init_dates(dates, freq)
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so i
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so i
  self._init_dates(dates, freq)
```

```
rolling_mean_detrended_diff['predict']=model_fit.predict(start=len(train),
                                                         end=len(train)+len(test)-1,dynamic=True)
```

```
rolling_mean_detrended_diff[['Weekly_Sales','predict']].plot()
```

```
<Axes: xlabel='Date'>
```



```
from statsmodels.tsa.statespace.sarimax import SARIMAX,SARIMAXResults
```

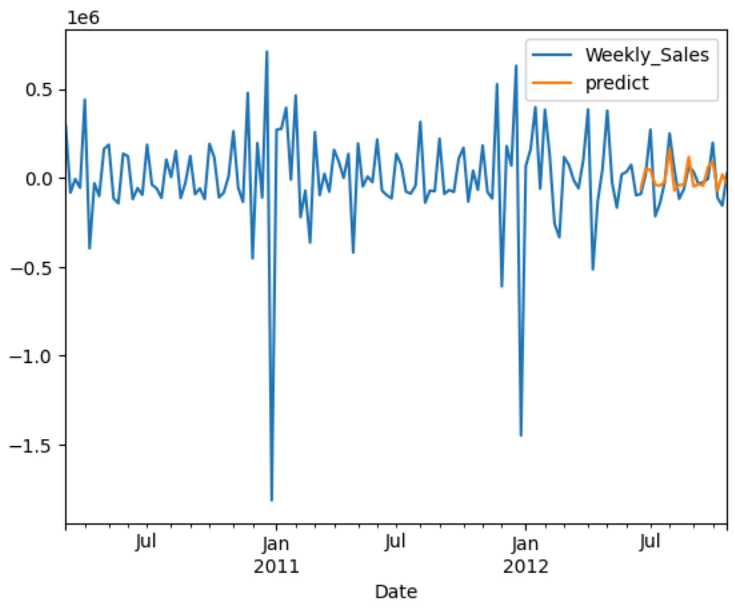
```
model=SARIMAX(train,order=(3,0,1),seasonal_order=(3,0,1,52))
model=model.fit()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so i
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so i
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/statespace/sarimax.py:866: UserWarning: Too few observations to estimate startin
  warn('Too few observations to estimate starting parameters%s.'
```

```
rolling_mean_detrended_diff['predict']=model.predict(start=len(train),
                                                     end=len(train)+len(test)-1,dynamic=True)
```

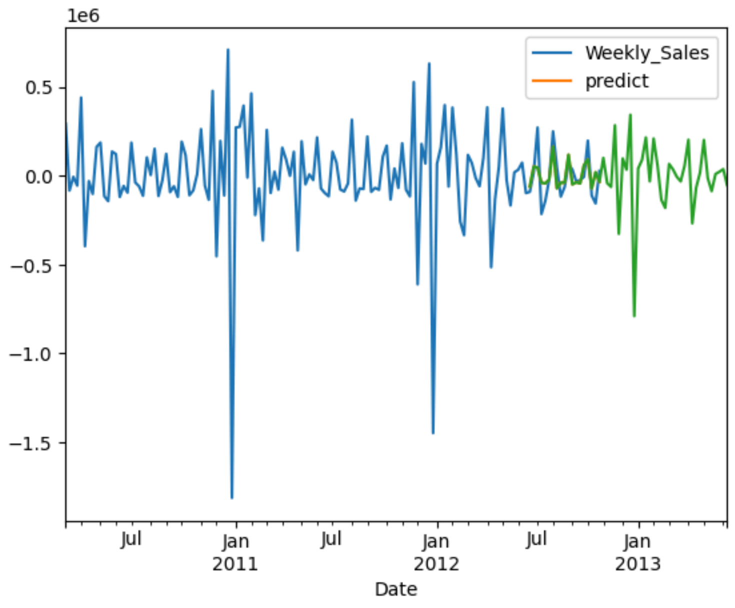
```
rolling_mean_detrended_diff[['Weekly_Sales','predict']].plot()
```

```
<Axes: xlabel='Date'>
```



```
forecast = model.forecast(steps=52)
rolling_mean_detrended_diff.plot()
forecast.plot()
```

```
<Axes: xlabel='Date'>
```



```
print(forecast)
```

```
2012-06-22    -61087.258439
2012-06-29     50838.903188
2012-07-06     45347.952010
2012-07-13    -39528.373193
2012-07-20    -43538.402549
2012-07-27    -22143.335720
2012-08-03    163014.566528
2012-08-10    -71299.264058
2012-08-17    -39535.972704
2012-08-24    -39399.994348
2012-08-31    117377.990232
2012-09-07    -49159.901904
2012-09-14    -37108.200380
2012-09-21    -43855.243929
2012-09-28     59328.432153
2012-10-05     90251.140979
2012-10-12    -71779.875941
2012-10-19     19279.241825
```

```
2012-10-26    -35656.642162
2012-11-02     99766.717050
2012-11-09    -42309.299036
2012-11-16    -63336.372511
2012-11-23    283468.012208
2012-11-30   -327322.441632
2012-12-07     96728.487508
2012-12-14     33501.328811
2012-12-21    342288.772972
2012-12-28   -790119.620328
2013-01-04     40105.506201
2013-01-11     88562.529280
2013-01-18    214527.978893
2013-01-25    -32102.573184
2013-02-01    208521.310691
2013-02-08     59029.265759
2013-02-15   -135858.710376
2013-02-22   -181372.893185
2013-03-01     65809.737186
2013-03-08     35287.888894
2013-03-15     -6473.087345
2013-03-22    -32385.699311
2013-03-29     55601.396841
2013-04-05    201886.382017
2013-04-12   -268369.097328
2013-04-19    -67185.389939
2013-04-26     17377.998357
2013-05-03    200460.569259
2013-05-10    -14550.781591
2013-05-17    -87188.853826
2013-05-24      9233.245710
2013-05-31     21520.391101
2013-06-07     36569.857719
2013-06-14    -53129.172886
Freq: W-FRI, Name: predicted_mean, dtype: float64
```

Start coding or generate with AI.

* Now we are forcasting for store-14 *

```
#Removing store column from the dataframe.
#del df_14['Store']
```

```
df_14
```

Date	Weekly_Sales
2010-02-05	2623469.95
2010-02-12	1704218.84
2010-02-19	2204556.70
2010-02-26	2095591.63
2010-03-05	2237544.75
...	...
2012-09-28	1522512.20
2012-10-05	1687592.16
2012-10-12	1639585.61
2012-10-19	1590274.72
2012-10-26	1704357.62

143 rows × 1 columns

```
df_14.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 143 entries, 2010-02-05 to 2012-10-26
Data columns (total 1 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
```

```
 0   Weekly_Sales  143 non-null    float64
dtypes: float64(1)
memory usage: 2.2 KB
```

```
df_14.columns
```

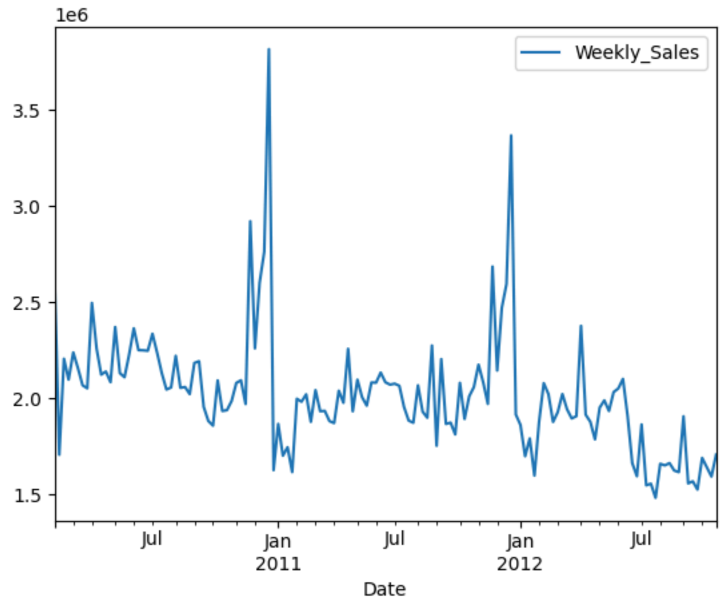
```
Index(['Weekly_Sales'], dtype='object')
```

```
df_14.index
```

```
DatetimeIndex(['2010-02-05', '2010-02-12', '2010-02-19', '2010-02-26',
               '2010-03-05', '2010-03-12', '2010-03-19', '2010-03-26',
               '2010-04-02', '2010-04-09',
               ...
               '2012-08-24', '2012-08-31', '2012-09-07', '2012-09-14',
               '2012-09-21', '2012-09-28', '2012-10-05', '2012-10-12',
               '2012-10-19', '2012-10-26'],
              dtype='datetime64[ns]', name='Date', length=143, freq=None)
```

```
df_14.plot()
```

```
<Axes: xlabel='Date'>
```



Start coding or generate with AI.

```
result14=adfuller(df_14['Weekly_Sales'])
pvalue=result14[1]
```

```
pvalue
```

```
0.06786986708375098
```

```
if pvalue<0.05:
  print('w accept null hypothesis- data is stationary')

else:
  print('we reject null hypothesis - data is not stationary')
```

```
we reject null hypothesis - data is not stationary
```

```
r1=pd.Series(result14[0:3],index=['Test Stats','p-value','number of observations used'])
```

```
r1
```

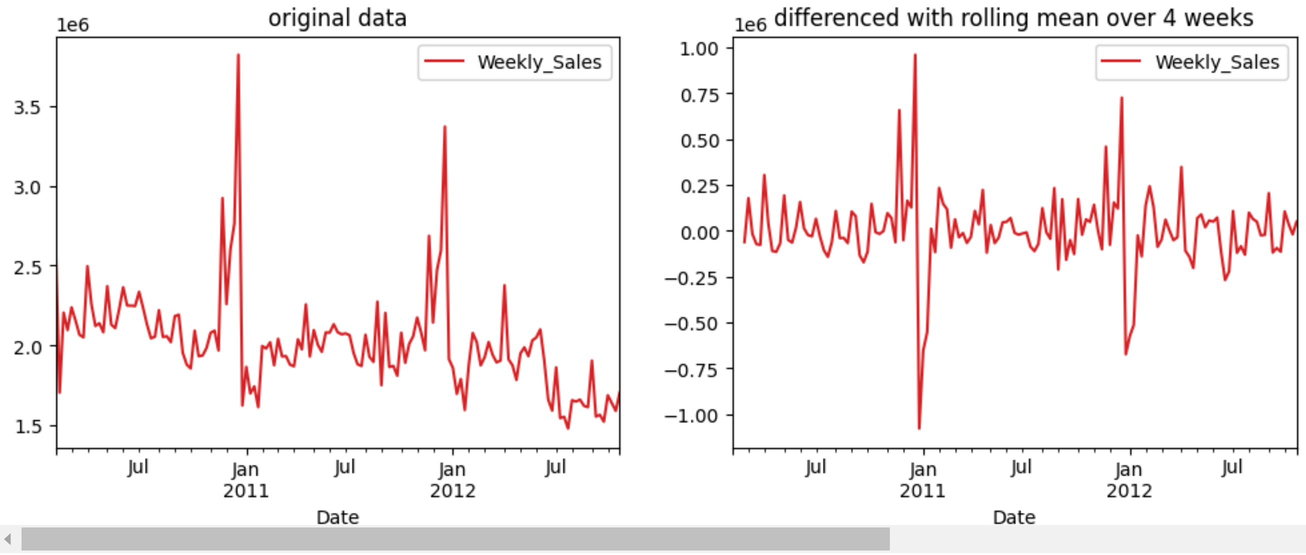
```
Test Stats                     -2.736887
p-value                         0.067870
number of observations used     6.000000
dtype: float64
```

```
rolling_mean=df_14.rolling(window=4).mean()
rolling_mean_detrended=df_14-rolling_mean
```

```
rolling_mean_detrended[:5]
```

Date	Weekly_Sales
2010-02-05	NaN
2010-02-12	NaN
2010-02-19	NaN
2010-02-26	-61367.65
2010-03-05	177066.77

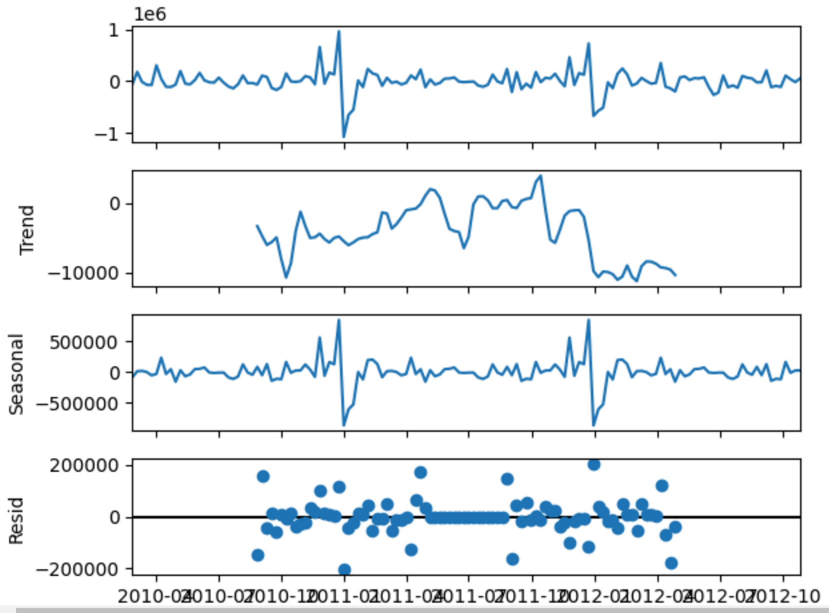
```
ax1=plt.subplot(1,2,2)
rolling_mean_detrended.plot(figsize=(12,4),color='tab:red',
                            title='differenced with rolling mean over 4 weeks',
                            ax=ax1)

ax2=plt.subplot(1,2,1)
df_14.plot(figsize=(12,4),
          color='tab:red',
          title="original data",
          ax=ax2)

plt.show()
```



```
from statsmodels.tsa.seasonal import seasonal_decompose
```

```
decompose_result14=seasonal_decompose(rolling_mean_detrended.dropna())
```

```
decompose_result14.plot();
```



```
rolling_mean_detrended[:5]
```

Date	Weekly_Sales
2010-02-05	NaN
2010-02-12	NaN
2010-02-19	NaN
2010-02-26	-61367.65
2010-03-05	177066.77

```
rolling_mean_detrended.shift()[:5]
```

Date	Weekly_Sales
2010-02-05	NaN
2010-02-12	NaN
2010-02-19	NaN
2010-02-26	NaN
2010-03-05	-61367.65

```
rolling_mean_detrended
```

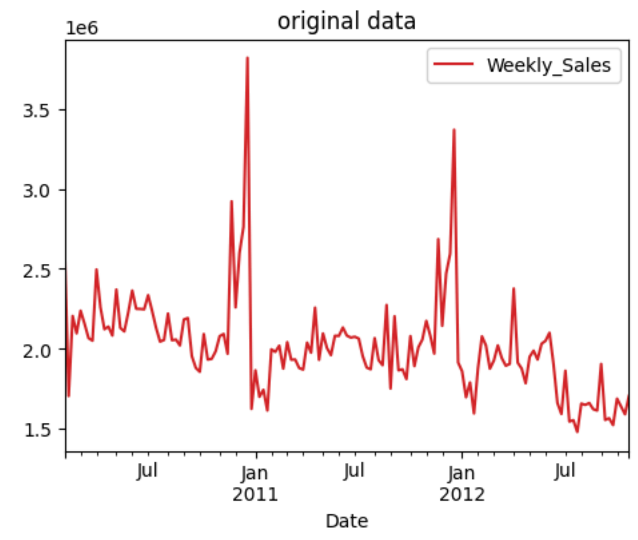
```
            Weekly_Sales
      Date
2010-02-05           NaN
2010-02-12           NaN
2010-02-19           NaN
2010-02-26   -61367.6500
2010-03-05   177066.7700
       ...           ...
2012-09-28  -114280.6050
2012-10-05   105029.4000
2012-10-12    35825.0025
2012-10-19   -19716.4525
2012-10-26    48905.0925

143 rows × 1 columns
```

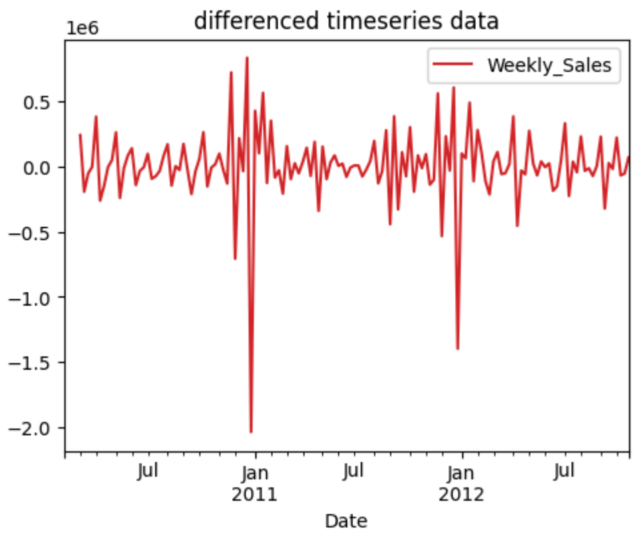
```
rolling_mean_detrended.shift()
```

```
            Weekly_Sales
      Date
2010-02-05           NaN
2010-02-12           NaN
2010-02-19           NaN
2010-02-26           NaN
2010-03-05   -61367.6500
       ...           ...
2012-09-28   -94147.8425
2012-10-05  -114280.6050
2012-10-12   105029.4000
2012-10-19    35825.0025
2012-10-26   -19716.4525

143 rows × 1 columns
```

```
rolling_mean_detrended_diff=rolling_mean_detrended-rolling_mean_detrended.shift()
```

```
  ax1=plt.subplot(1,2,2)
  rolling_mean_detrended_diff.plot(figsize=(12,4),color='tab:red',
                              title='differenced timeseries data',
                              ax=ax1)

  ax2=plt.subplot(1,2,1)
  df_14.plot(figsize=(12,4),
            color='tab:red',
            title="original data",
            ax=ax2)

  plt.show()
```





```
rolling_mean_detrended_diff=rolling_mean_detrended_diff.dropna()
```

```
rolling_mean_detrended_diff
```

Date	Weekly_Sales
2010-03-05	238434.4200
2010-03-12	-194463.7450
2010-03-19	-55231.4100
2010-03-26	-4524.1900
2010-04-02	380712.8000
...	...
2012-09-28	-20132.7625
2012-10-05	219310.0050
2012-10-12	-69204.3975
2012-10-19	-55541.4550
2012-10-26	68621.5450

139 rows × 1 columns

```
result14=adfuller(rolling_mean_detrended_diff['Weekly_Sales'])
pvalue=result14[1]
```

```
pvalue
```

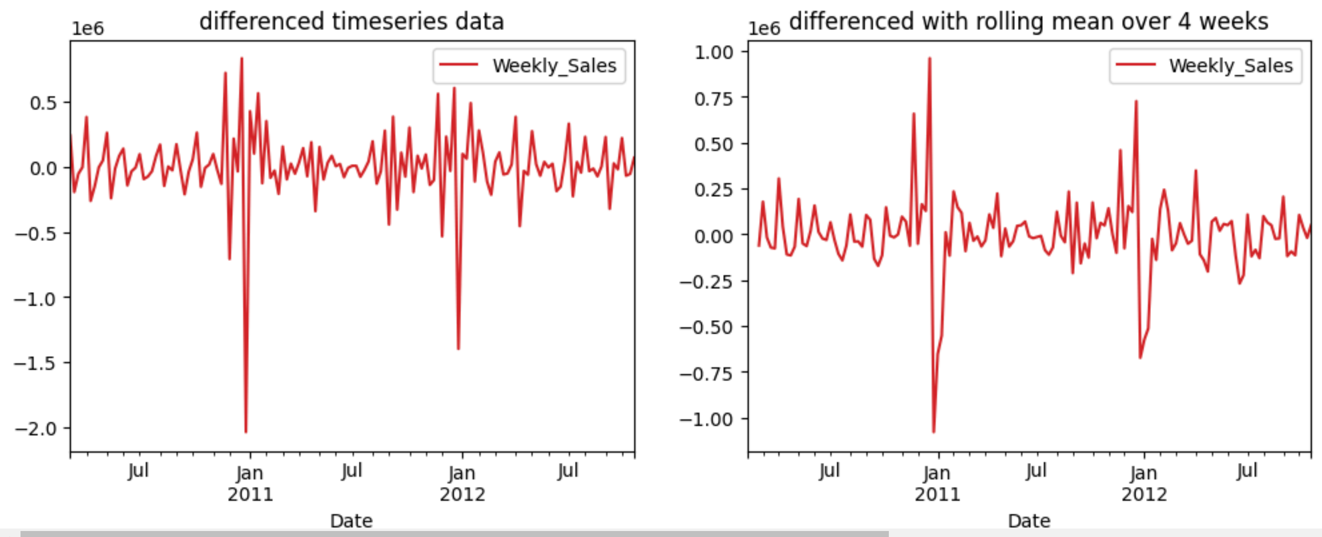
```
3.303266782084958e-12
```

```
if pvalue<0.05:
  print('we accept null hypothesis- data is stationary')

else:
  print('we reject null hypothesis - data is not stationary')
```

```
we accept null hypothesis- data is stationary
```

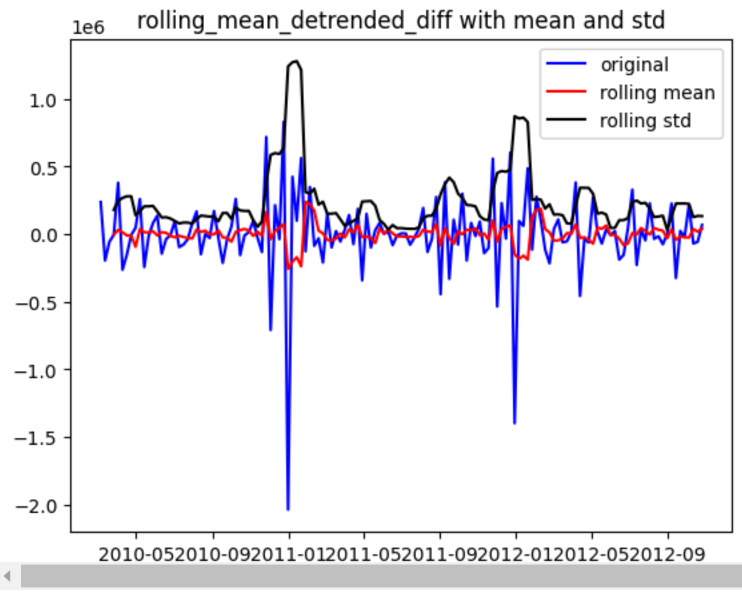
```
ax1=plt.subplot(1,2,2)
rolling_mean_detrended.plot(figsize=(12,4),color='tab:red',
                            title='differenced with rolling mean over 4 weeks',
                            ax=ax1)

ax2=plt.subplot(1,2,1)
rolling_mean_detrended_diff.plot(figsize=(12,4),color='tab:red',
                            title='differenced timeseries data',
                            ax=ax2)
```

```
<Axes: title={'center': 'differenced timeseries data'}, xlabel='Date'>
```



```
m=rolling_mean_detrended_diff.rolling(window=4).mean()
s=rolling_mean_detrended_diff.rolling(window=4).std()


plt.plot(rolling_mean_detrended_diff,color='blue',label='original')
plt.plot(m,color='red',label='rolling mean')
plt.plot(s,color='black',label='rolling std')
plt.legend(loc='best')
plt.title('rolling_mean_detrended_diff with mean and std')
```

```
Text(0.5, 1.0, 'rolling_mean_detrended_diff with mean and std')
```



```
!pip install pmdarima
```

```
Requirement already satisfied: pmdarima in /usr/local/lib/python3.10/dist-packages (2.0.4)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.3.2)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (3.0.8)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.23.5)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.5.3)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.2.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.11.4)
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (0.14.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.0.7)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (67.7.2)
Requirement already satisfied: packaging>=17.1 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (23.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2023.3.post1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22->pmdarima) (3.2.
```