Walmart 2025 (2)

June 24, 2025

1 Walmart Project

- **We have weekly sales data available for various Walmart outlets. We will use statistical analysis and EDA to come up with various insights and trends that can give the stakeholders a clear perspective on the key business metrics**
- **The stakeholders have also asked to perform predictive modelling to forecast the sales for the next 12 weeks**

1.1 Importing preliminary libraries

[71]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

1.2 Loading the dataset

[73]:
```
df = pd.read_csv("Walmart DataSet.csv")
```

[74]:
```
df.head()
```

[74]:
```
   Store        Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
0      1  05-02-2010    1643690.90             0        42.31       2.572
1      1  12-02-2010    1641957.44             1        38.51       2.548
2      1  19-02-2010    1611968.17             0        39.93       2.514
3      1  26-02-2010    1409727.59             0        46.63       2.561
4      1  05-03-2010    1554806.68             0        46.50       2.625

          CPI  Unemployment
0  211.096358         8.106
1  211.242170         8.106
2  211.289143         8.106
3  211.319643         8.106
4  211.350143         8.106
```

1.3 General Observation

[76]:
```
df.info()
```

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

[77]:
```
#Number of stores
df["Store"].nunique()
```

[77]:
```
45
```

[78]:
```
for column in df:
    print(f"{column} - ({len(df[column].unique())}) : {df[column].unique()} \n")
```

```
Store - (45) : [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
22 23 24
 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45]

Date - (143) : ['05-02-2010' '12-02-2010' '19-02-2010' '26-02-2010' '05-03-2010'
 '12-03-2010' '19-03-2010' '26-03-2010' '02-04-2010' '09-04-2010'
 '16-04-2010' '23-04-2010' '30-04-2010' '07-05-2010' '14-05-2010'
 '21-05-2010' '28-05-2010' '04-06-2010' '11-06-2010' '18-06-2010'
 '25-06-2010' '02-07-2010' '09-07-2010' '16-07-2010' '23-07-2010'
 '30-07-2010' '06-08-2010' '13-08-2010' '20-08-2010' '27-08-2010'
 '03-09-2010' '10-09-2010' '17-09-2010' '24-09-2010' '01-10-2010'
 '08-10-2010' '15-10-2010' '22-10-2010' '29-10-2010' '05-11-2010'
 '12-11-2010' '19-11-2010' '26-11-2010' '03-12-2010' '10-12-2010'
 '17-12-2010' '24-12-2010' '31-12-2010' '07-01-2011' '14-01-2011'
 '21-01-2011' '28-01-2011' '04-02-2011' '11-02-2011' '18-02-2011'
 '25-02-2011' '04-03-2011' '11-03-2011' '18-03-2011' '25-03-2011'
 '01-04-2011' '08-04-2011' '15-04-2011' '22-04-2011' '29-04-2011'
 '06-05-2011' '13-05-2011' '20-05-2011' '27-05-2011' '03-06-2011'
 '10-06-2011' '17-06-2011' '24-06-2011' '01-07-2011' '08-07-2011'
 '15-07-2011' '22-07-2011' '29-07-2011' '05-08-2011' '12-08-2011'
 '19-08-2011' '26-08-2011' '02-09-2011' '09-09-2011' '16-09-2011'
```

```
 '23-09-2011' '30-09-2011' '07-10-2011' '14-10-2011' '21-10-2011'
 '28-10-2011' '04-11-2011' '11-11-2011' '18-11-2011' '25-11-2011'
 '02-12-2011' '09-12-2011' '16-12-2011' '23-12-2011' '30-12-2011'
 '06-01-2012' '13-01-2012' '20-01-2012' '27-01-2012' '03-02-2012'
 '10-02-2012' '17-02-2012' '24-02-2012' '02-03-2012' '09-03-2012'
 '16-03-2012' '23-03-2012' '30-03-2012' '06-04-2012' '13-04-2012'
 '20-04-2012' '27-04-2012' '04-05-2012' '11-05-2012' '18-05-2012'
 '25-05-2012' '01-06-2012' '08-06-2012' '15-06-2012' '22-06-2012'
 '29-06-2012' '06-07-2012' '13-07-2012' '20-07-2012' '27-07-2012'
 '03-08-2012' '10-08-2012' '17-08-2012' '24-08-2012' '31-08-2012'
 '07-09-2012' '14-09-2012' '21-09-2012' '28-09-2012' '05-10-2012'
 '12-10-2012' '19-10-2012' '26-10-2012']

Weekly_Sales - (6435) : [1643690.9  1641957.44 1611968.17 …  734464.36
718125.53  760281.43]

Holiday_Flag - (2) : [0 1]

Temperature - (3528) : [42.31 38.51 39.93 … 75.87 77.55 74.09]

Fuel_Price - (892) : [2.572 2.548 2.514 2.561 2.625 2.667 2.72  2.732 2.719 2.77
2.808 2.795
 2.78  2.835 2.854 2.826 2.759 2.705 2.668 2.637 2.653 2.669 2.642 2.623
 2.608 2.64  2.627 2.692 2.664 2.619 2.577 2.565 2.582 2.624 2.603 2.633
 2.725 2.716 2.689 2.728 2.771 2.735 2.708 2.843 2.869 2.886 2.943 2.976
 2.983 3.016 3.01  2.989 3.022 3.045 3.065 3.288 3.459 3.488 3.473 3.524
 3.622 3.743 3.807 3.81  3.906 3.899 3.907 3.786 3.699 3.648 3.637 3.594
 3.48  3.575 3.651 3.682 3.684 3.638 3.554 3.523 3.533 3.546 3.526 3.467
 3.355 3.285 3.274 3.353 3.372 3.332 3.297 3.308 3.236 3.172 3.158 3.159
 3.112 3.129 3.157 3.261 3.268 3.29  3.36  3.409 3.51  3.555 3.63  3.669
 3.734 3.787 3.845 3.891 3.877 3.814 3.749 3.688 3.561 3.501 3.452 3.393
 3.346 3.286 3.227 3.256 3.311 3.407 3.417 3.494 3.571 3.62  3.73  3.717
 3.721 3.666 3.617 3.601 3.506 2.598 2.573 2.54  2.59  2.654 2.704 2.743
 2.752 2.74  2.773 2.81  2.805 2.787 2.836 2.845 2.82  2.756 2.701 2.635
 2.621 2.612 2.65  2.698 2.671 2.584 2.574 2.594 2.645 2.736 2.718 2.699
 2.741 2.727 2.86  2.884 2.887 2.955 2.98  2.992 3.017 2.996 3.033 3.058
 3.087 3.305 3.461 3.495 3.521 3.605 3.724 3.781 3.866 3.872 3.881 3.771
 3.683 3.64  3.618 3.57  3.504 3.469 3.563 3.627 3.659 3.662 3.55  3.532
 3.371 3.299 3.283 3.361 3.362 3.322 3.294 3.225 3.176 3.153 3.149 3.103
 3.119 3.263 3.273 3.354 3.411 3.493 3.541 3.619 3.667 3.707 3.759 3.82
 3.864 3.747 3.685 3.551 3.483 3.433 3.329 3.257 3.187 3.224 3.356 3.374
 3.476 3.552 3.61  3.646 3.709 3.706 3.603 3.514 2.58  2.55  2.586 2.62
 2.684 2.717 2.75  2.765 2.776 2.766 2.788 2.737 2.7   2.674 2.715 2.711
 2.691 2.69  2.723 2.731 2.8   2.793 2.745 2.762 2.748 2.729 2.758 2.742
 2.712 2.778 2.781 2.829 2.882 2.911 2.973 3.008 3.011 3.037 3.051 3.101
 3.232 3.406 3.414 3.611 3.636 3.663 3.735 3.767 3.828 3.795 3.763 3.697
 3.661 3.597 3.54  3.545 3.547 3.542 3.499 3.485 3.511 3.566 3.596 3.581
 3.538 3.498 3.491 3.548 3.527 3.505 3.479 3.424 3.378 3.331 3.266 3.173
```

```
3.095 3.077 3.055 3.038 3.031 3.113 3.191 3.486 3.664 3.75  3.854 3.901
3.936 3.927 3.903 3.87  3.837 3.804 3.764 3.741 3.723 3.693 3.613 3.585
3.528 3.509 3.558 3.556 3.765 3.789 3.779 3.76  3.686 2.962 2.828 2.915
2.825 2.877 3.034 3.054 3.086 3.004 3.109 3.05  3.105 3.127 3.145 3.12
2.941 3.057 2.935 3.084 2.978 3.1   2.971 3.123 3.049 3.041 2.961 3.028
2.939 3.001 2.924 3.08  3.014 3.13  3.009 3.047 3.162 3.091 3.125 3.148
3.287 3.312 3.336 3.231 3.348 3.381 3.43  3.398 3.674 3.892 3.716 3.772
3.818 4.089 3.917 4.151 4.193 4.202 3.99  3.933 3.893 3.981 3.935 3.842
3.793 3.694 3.803 3.794 3.798 3.784 3.827 3.698 3.843 3.677 3.701 3.644
3.489 3.428 3.443 3.477 3.66  3.675 3.543 3.722 3.95  3.882 3.963 4.273
4.288 4.294 4.282 4.254 4.111 4.088 4.058 4.186 4.308 4.127 4.277 4.103
4.144 4.014 3.875 3.589 3.769 3.595 3.811 4.002 4.055 3.886 4.124 3.966
4.125 4.132 4.468 4.449 4.301 2.946 2.987 2.925 3.083 3.09  2.949 3.043
3.094 3.044 3.013 3.161 3.203 3.223 3.342 3.53  3.692 3.909 4.003 3.868
4.134 4.169 4.087 4.031 3.898 3.705 3.805 3.74  3.913 3.918 3.727 3.824
3.813 3.6   3.599 3.657 3.702 4.178 4.25  4.038 4.121 4.222 4.171 4.11
4.293 3.726 4.093 4.133 2.666 2.681 2.733 2.782 2.819 2.842 2.936 2.948
2.95  2.908 2.871 2.841 2.814 2.802 2.791 2.797 2.837 2.85  2.868 2.87
2.875 2.872 2.853 2.849 2.831 2.83  2.812 2.817 2.846 2.891 2.903 2.934
2.96  2.974 3.062 3.23  3.435 3.487 3.616 3.655 3.744 3.77  3.802 3.778
3.752 3.732 3.704 3.668 3.553 3.574 3.606 3.578 3.58  3.641 3.623 3.592
3.567 3.579 3.513 3.445 3.389 3.341 3.282 3.186 3.056 3.116 3.242 3.38
3.529 3.671 3.833 3.831 3.809 3.808 3.801 3.788 3.776 3.756 3.737 3.681
3.537 3.512 3.582 3.624 3.689 3.821 3.815 3.797 3.755 2.784 2.754 2.777
2.818 2.844 2.899 2.902 2.921 2.966 2.982 2.958 2.847 2.809 2.815 2.783
2.779 2.755 2.706 2.713 2.707 2.764 2.917 2.931 3.    3.039 3.046 3.14
3.141 3.179 3.193 3.205 3.229 3.237 3.239 3.245 3.631 3.625 3.72  3.962
4.046 4.066 4.062 3.985 3.922 3.748 3.711 3.829 3.812 3.703 3.738 3.742
3.645 3.583 3.569 3.492 3.415 3.413 3.422 3.695 3.739 3.816 3.848 3.862
3.9   3.953 3.996 4.044 4.027 4.004 3.951 3.889 3.564 3.475 3.647 3.654
3.834 3.867 3.911 3.948 3.997 4.    3.969 2.954 2.94  2.909 2.91  2.919
2.938 2.963 2.957 3.021 3.042 3.096 3.006 2.972 2.942 2.933 2.932 2.923
2.913 2.885 2.84  2.999 3.138 3.2   3.255 3.301 3.309 3.351 3.367 3.391
3.402 3.4   3.416 3.42  3.796 3.895 4.061 4.117 4.192 4.211 4.069 4.025
3.989 3.964 3.916 3.915 3.972 4.02  3.995 3.942 3.879 3.93  3.937 3.858
3.775 3.757 3.719 3.587 3.826 3.874 3.983 4.021 4.054 4.098 4.143 4.187
4.17  4.163 4.029 3.979 3.871 3.819 3.863 4.026 4.076 4.203 4.158 4.153
4.071 2.747 2.753 2.834 2.895 2.981 2.906 2.857 2.806 2.796 2.792 2.878
3.03  3.07  3.132 3.139 3.15  3.177 3.215 3.243 3.24  3.281 3.437 3.634
3.823 3.919 3.988 4.078 4.095 4.101 4.034 3.973 3.924 3.873 3.851 3.88
3.758 3.633 3.604 3.586 3.536 3.47  3.439 3.568 3.751 3.876 3.921 3.957
4.023 3.991 3.947 3.85  3.746 3.629 3.577 3.84  3.884 4.056 4.018 2.545
2.539 2.472 2.52  2.769 2.786 2.767 2.615 2.601 2.606 2.596 2.542 2.602
2.644 2.604 2.562 2.533 2.513 2.578 2.567 2.595 2.68  2.655 2.694 2.813
2.852 2.863 2.995 3.053 3.448 3.92  3.925 3.69  3.502 3.44  3.652 3.608
3.534 3.481 3.441 3.328 3.262 3.234 3.306 3.254 3.26  3.181 3.164 3.147
3.133 3.098 3.275 3.313 3.421 3.462 3.503 3.934 3.888 3.835 3.713 3.358
3.392 3.404 3.49  3.576]
```

```
CPI - (2145) : [211.0963582 211.2421698 211.2891429 … 214.6772833 214.7212488
 214.7415392]

Unemployment - (349) : [ 8.106  7.808  7.787  7.838  7.742  7.682  7.962  7.866
7.348  7.143
  6.908  6.573  8.324  8.2    8.099  8.163  8.028  7.931  7.852  7.441
  7.057  6.891  6.565  6.17   7.368  7.343  7.346  7.564  7.551  7.574
  7.567  7.197  6.833  6.664  6.334  6.034  8.623  7.896  7.372  7.127
  6.51   5.946  5.644  5.143  4.607  4.308  4.077  3.879  6.566  6.465
  6.496  6.768  6.634  6.489  6.529  6.3    5.943  5.801  5.603  5.422
  7.259  7.092  6.973  7.007  6.858  6.855  6.925  6.551  6.132  5.964
  5.668  5.329  9.014  8.963  9.017  9.137  8.818  8.595  8.622  8.513
  8.256  8.09   7.872  7.557  6.299  6.29   6.315  6.433  6.262  6.297
  6.425  6.123  5.825  5.679  5.401  5.124  6.415  6.384  6.442  6.56
  6.416  6.38   6.404  6.054  5.667  5.539  5.277  4.954  9.765  9.524
  9.199  9.003  8.744  8.494  8.257  7.874  7.545  7.382  7.17   6.943
 13.975 14.099 14.18  14.313 14.021 13.736 13.503 12.89  12.187 11.627
 10.926 10.199  8.316  8.107  7.951  7.795  7.47   7.193  6.877  6.392
  6.104  5.965  5.765  5.621  8.992  8.899  8.743  8.724  8.549  8.521
  8.625  8.523  8.424  8.567  8.684  8.667  8.35   8.185  8.067  7.771
  7.658  7.806  7.943  8.15   8.193  7.992  7.039  6.842  6.868  6.986
  6.614  6.339  6.338  6.232  6.162  6.169  6.061  5.847  6.548  6.635
  6.697  6.885  6.866  6.774  6.745  6.617  6.403  6.235  5.936  5.527
  9.202  9.269  9.342  9.331  9.131  8.975  8.89   8.471  8.075  8.304
  8.535  8.243  8.187  7.856  7.527  7.484  7.287  7.274  7.082  6.961
  7.139  7.28   7.293  8.283  8.348  8.433  8.572  8.458  8.252  8.023
  7.706  7.503  7.671  7.753  7.543  5.892  5.435  5.326  5.287  5.114
  4.781  4.584  4.42   4.261  4.125  4.156  4.145  8.326  8.211  8.117
  8.275  8.212  8.358  8.454  8.659  8.983  8.953  8.693  8.488  8.512
  8.445  8.149  7.907  7.818  7.767  7.598  7.467  7.489  7.405  7.138
  8.237  8.058  7.982  8.021  7.827  7.725  7.85   7.906  8.009  8.253
  8.239  8.    10.064 10.16  10.409 10.524 10.256  9.966  9.863  9.357
  8.988  9.14   9.419  9.151 10.115  9.849  9.495  9.265  8.951  8.687
  8.442  8.01   7.603  7.396  7.147  6.895  9.521  9.593  9.816 10.21
 10.398 10.581 10.641 10.148  9.653  9.575  9.285  8.839  9.262  9.051
  8.861  8.763  8.745  8.876  8.665  8.554  8.464  8.36   8.476  8.395
  8.3    8.177  7.716  7.244  6.989  6.623  6.228  7.541  7.363  7.335
  7.508  7.241  6.934  6.901  6.759  6.589  6.547  6.432  6.195  8.119
  7.972  7.804  7.61   7.224  6.906  6.078  5.774  5.407  5.217]
```

1.4 Exploarotry Data Analysis

[80]:
```
df.columns
```

[80]:
```
Index(['Store', 'Date', 'Weekly_Sales', 'Holiday_Flag', 'Temperature',
       'Fuel_Price', 'CPI', 'Unemployment'],
      dtype='object')
```

Renaming the columns: Converting all column names into small letters for convenience

[82]:
```
df.rename(columns = {"Store" : "store",
                     "Date" : "date",
                     "Weekly_Sales" : "weekly_sales",
                     "Holiday_Flag" : "holiday_flag",
                     "Temperature" : "temperature",
                     "Fuel_Price" : "fuel_price",
                     "CPI" : "cpi",
                     "Unemployment" : "unemployment"}, inplace=True)
```

[83]:
```
df.columns
```

[83]:
```
Index(['store', 'date', 'weekly_sales', 'holiday_flag', 'temperature',
       'fuel_price', 'cpi', 'unemployment'],
      dtype='object')
```

[84]:
```
# Changing the data type of date column
df["date"] = pd.to_datetime(df["date"], dayfirst=True, errors="coerce")
```

Let us verify the number of weeks of sales data available for each store

[86]:
```
df["store"].value_counts().unique()
```

[86]:
```
array([143], dtype=int64)
```

We have 143 weeks of data available for each store

1.4.1 Statistical Summary

[89]:
```
df.describe().T
```

[89]:
```
               count                 mean                  min  \
store         6435.0                 23.0                  1.0
date            6435  2011-06-17 00:00:00  2010-02-05 00:00:00
weekly_sales  6435.0       1046964.877562            209986.25
holiday_flag  6435.0              0.06993                  0.0
temperature   6435.0            60.663782                -2.06
fuel_price    6435.0             3.358607                2.472
cpi           6435.0           171.578394              126.064
unemployment  6435.0             7.999151                3.879

                              25%                  50%                  75%  \
store                        12.0                 23.0                 34.0
date          2010-10-08 00:00:00  2011-06-17 00:00:00  2012-02-24 00:00:00
```

```
weekly_sales            553350.105            960746.04            1420158.66
holiday_flag                   0.0                  0.0                   0.0
temperature                  47.46                62.67                 74.94
fuel_price                   2.933                3.445                 3.735
cpi                        131.735           182.616521            212.743293
unemployment                 6.891                7.874                 8.622

                              max            std
store                        45.0      12.988182
date          2012-10-26 00:00:00            NaN
weekly_sales           3818686.45  564366.622054
holiday_flag                  1.0       0.255049
temperature                100.14      18.444933
fuel_price                  4.468        0.45902
cpi                    227.232807      39.356712
unemployment               14.313       1.875885
```

Analysis of statistical summary **weekly_Sales**

- Mean sales 1.046M
- Standard deviation is 564K
- Median is 960K
- Median<mean, hence the data is slightly skewed towards the right
- Min sales is 209.9K and Max sales is 3.81M
- The range from min to first quartile is smaller compared to the range from third quartile to max
- The minimum value is within 2 standard deviations from the mean
- While the maximum value is outside of 3 standard deviations from the mean
- This means there are outliers present outside upper bound

holiday_flag

- It's a boolean column
- 0 - no holiday in a given week
- 1 - there is a holiday in a given week
- Clearly, most of the values will be 0 as there are very few holidays compared to working days

temperature

- Mean temperature is 60.66
- Standard deviation is 18.44
- Min temperatrue is -2.06 while Max temperature is 100.14
- The range from first quartile to min is bigger compared to the range from third quartile to max
- And median>mean, this suggests the data is left skewed
- Max temperatrue is within 3 standard deviations from the mean while min temperature is beyond 3 standard deviations from the mean, indicating outliers in the lower bound

fuel_price

- Mean price is 3.35

- Standard deviation is 0.45
- Min price is 2.47 and max price is 4.46
- Min price is within 2 standard deviations and Max price is within 3 standard deviations from the mean
- Mean is slightly less than the median, hence data is slightly skewed towards the left

cpi

- Mean cpi is 171.5
- Standard deviation is 39.35
- Min cpi is 126 and max cpi is 227
- Both Min and Max cpi are within 2 standard deviations from the mean

unemployment

- Mean unemployment rate is 7.99~8
- Standard deviation is 1.875
- Min rate is 3.879 and max rate is 14.313
- The Min rate is within 3 standard deviations while max rate is beyond 3 standard deviations from the mean
- Hence, some outliers present outside the upper bound

[103]:
```
# Group the data by stores and evaluate mean, min, max, sum and standard␣
↪deviation of each store
store_groups = df.groupby("store")
store_stats = store_groups.agg({
    'weekly_sales': ['mean', 'std', 'min', 'max', 'sum']
}).round(2)
store_stats
```

[103]:

	weekly_sales				
	mean	std	min	max	sum
store					
1	1555264.40	155980.77	1316899.31	2387950.20	2.224028e+08
2	1925751.34	237683.69	1650394.44	3436007.68	2.753824e+08
3	402704.44	46319.63	339597.38	605990.41	5.758674e+07
4	2094712.96	266201.44	1762539.30	3676388.98	2.995440e+08
5	318011.81	37737.97	260636.71	507900.07	4.547569e+07
6	1564728.19	212525.86	1261253.18	2727575.18	2.237561e+08
7	570617.31	112585.47	372673.61	1059715.27	8.159828e+07
8	908749.52	106280.83	772539.12	1511641.09	1.299512e+08
9	543980.55	69028.67	452905.22	905324.68	7.778922e+07
10	1899424.57	302262.06	1627707.31	3749057.69	2.716177e+08
11	1356383.12	165833.89	1100418.69	2306265.36	1.939628e+08
12	1009001.61	139166.87	802105.50	1768249.89	1.442872e+08
13	2003620.31	265507.00	1633663.12	3595903.20	2.865177e+08
14	2020978.40	317569.95	1479514.66	3818686.45	2.889999e+08
15	623312.47	120538.65	454183.42	1368318.17	8.913368e+07
16	519247.73	85769.68	368600.00	1004730.69	7.425243e+07

```
17       893581.39  112162.94   635862.55  1309226.79  1.277821e+08
18      1084718.42  176641.51   540922.94  2027507.15  1.551147e+08
19      1444999.04  191722.64  1181204.53  2678206.42  2.066349e+08
20      2107676.87  275900.56  1761016.51  3766687.43  3.013978e+08
21       756069.08  128752.81   596218.24  1587257.78  1.081179e+08
22      1028501.04  161251.35   774262.28  1962445.04  1.470756e+08
23      1389864.46  249788.04  1016756.10  2734277.10  1.987506e+08
24      1356755.39  167745.68  1057290.41  2386015.75  1.940160e+08
25       706721.53  112976.79   558794.63  1295391.19  1.010612e+08
26      1002911.84  110431.29   809833.21  1573982.47  1.434164e+08
27      1775216.20  239930.14  1263534.86  3078162.08  2.538559e+08
28      1323522.24  181758.97  1079669.11  2026026.39  1.892637e+08
29       539451.43   99120.14   395987.24  1130926.79  7.714155e+07
30       438579.62   22809.67   369722.32   519354.88  6.271689e+07
31      1395901.44  125855.94  1198071.60  2068942.97  1.996139e+08
32      1166568.15  138017.25   955463.84  1959526.96  1.668192e+08
33       259861.69   24132.93   209986.25   331173.51  3.716022e+07
34       966781.56  104630.16   836717.75  1620748.25  1.382498e+08
35       919724.98  211243.46   576332.05  1781866.98  1.315207e+08
36       373511.99   60725.17   270677.98   489372.02  5.341221e+07
37       518900.28   21837.46   451327.61   605791.46  7.420274e+07
38       385731.65   42768.17   303908.81   499267.66  5.515963e+07
39      1450668.13  217466.45  1158698.44  2554482.84  2.074455e+08
40       964128.04  119002.11   764014.75  1648829.18  1.378703e+08
41      1268125.42  187907.16   991941.73  2263722.68  1.813419e+08
42       556403.86   50262.93   428953.60   674919.45  7.956575e+07
43       633324.72   40598.41   505405.85   725043.04  9.056544e+07
44       302748.87   24762.83   241937.11   376233.89  4.329309e+07
45       785981.41  130168.53   617207.58  1682862.03  1.123953e+08
```

[117]:
```python
store_stats.columns        #column names
```

[117]:
```
MultiIndex([('weekly_sales', 'mean'),
            ('weekly_sales',  'std'),
            ('weekly_sales',  'min'),
            ('weekly_sales',  'max'),
            ('weekly_sales',  'sum')],
           )
```

[119]:
```python
# Renaming the columns in store_stats
store_stats.columns = ["_".join(col).strip() for col in store_stats.columns]
```

[121]:
```python
store_stats.head()
```

[121]:
```
       weekly_sales_mean  weekly_sales_std  weekly_sales_min  \
store
1             1555264.40         155980.77        1316899.31
```

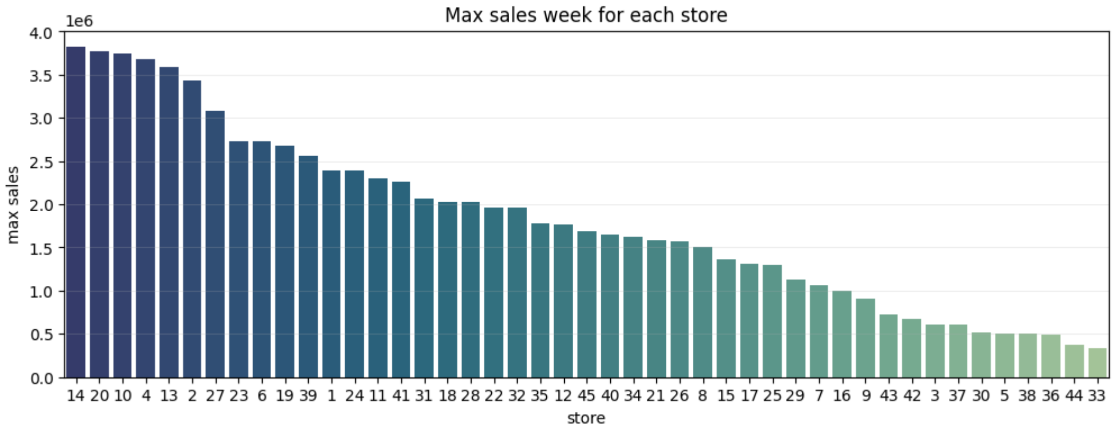
```
2             1925751.34         237683.69        1650394.44
3              402704.44          46319.63         339597.38
4             2094712.96         266201.44        1762539.30
5              318011.81          37737.97         260636.71

       weekly_sales_max  weekly_sales_sum
store
1            2387950.20      2.224028e+08
2            3436007.68      2.753824e+08
3             605990.41      5.758674e+07
4            3676388.98      2.995440e+08
5             507900.07      4.547569e+07
```

[123]:
```
store_stats = store_stats.reset_index()    #Resetting the index for␣
↪store_stats, so the index starts from 0
```

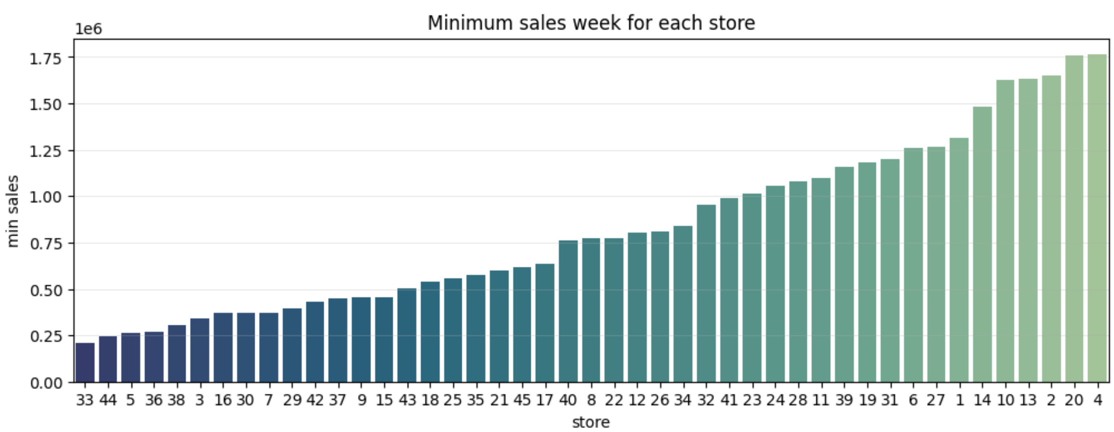
[125]:
```
store_stats.head()
```

[125]:
```
   store  weekly_sales_mean  weekly_sales_std  weekly_sales_min  \
0      1         1555264.40         155980.77        1316899.31
1      2         1925751.34         237683.69        1650394.44
2      3          402704.44          46319.63         339597.38
3      4         2094712.96         266201.44        1762539.30
4      5          318011.81          37737.97         260636.71

   weekly_sales_max  weekly_sales_sum
0        2387950.20      2.224028e+08
1        3436007.68      2.753824e+08
2         605990.41      5.758674e+07
3        3676388.98      2.995440e+08
4         507900.07      4.547569e+07
```

[127]:
```
# Visualizing the maximum sales week of each store
plt.figure(figsize=(12,4))
sns.barplot(
    x="store",
    y="weekly_sales_max",
    data=store_stats,
    order=store_stats.sort_values("weekly_sales_max", ascending=False)["store"],
    palette="crest_r"
)
plt.xlabel("store")
plt.ylabel("max sales")
plt.title("Max sales week for each store")
plt.grid(axis="y", alpha=0.4, linewidth=0.4)
plt.show()
```

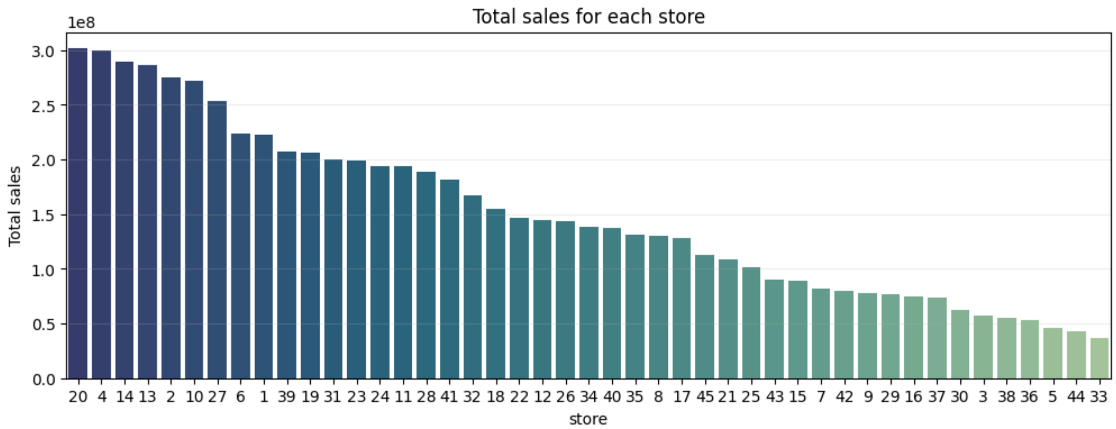


[129]:
```
#Visualizing the minimum sales week of each store
plt.figure(figsize=(12,4))
sns.barplot(x="store",
            y="weekly_sales_min",
            data=store_stats,
            order=store_stats.sort_values("weekly_sales_min")["store"],
            palette="crest_r"
           )
plt.xlabel("store")
plt.ylabel("min sales")
plt.title("Minimum sales week for each store")
plt.grid(axis="y", alpha=0.5, linewidth=0.4)
plt.show()
```



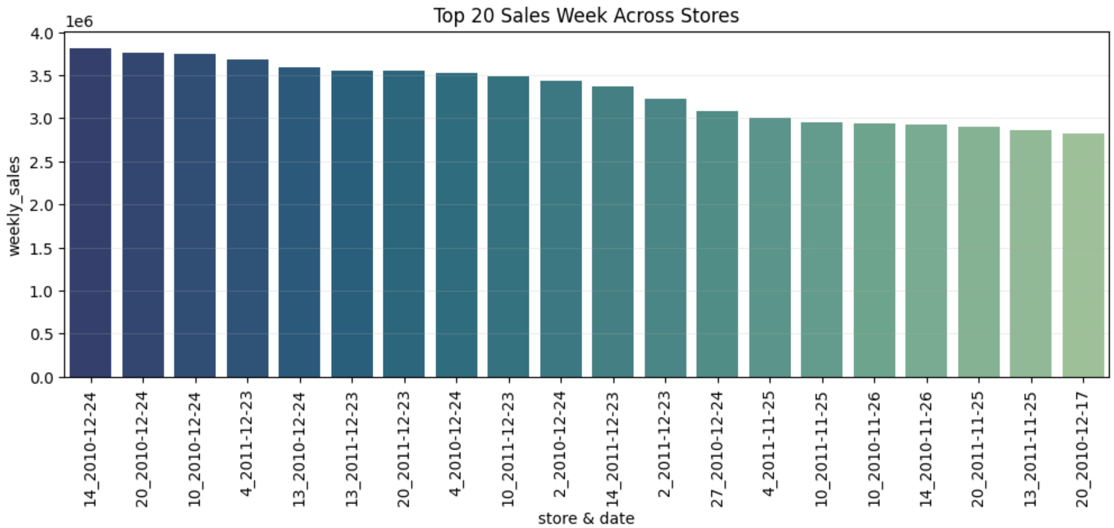
Store 33 and store 44 have the lowest max sales and the lowest min sales

[143]:
```
# Visualizing total sales for each store
plt.figure(figsize=(12,4))
sns.barplot(
    x="store",
    y="weekly_sales_sum",
    data=store_stats,
    order=store_stats.sort_values("weekly_sales_sum", ascending=False)["store"],
    palette="crest_r"
)
plt.xlabel("store")
plt.ylabel("Total sales")
plt.title("Total sales for each store")
plt.grid(axis="y",alpha=0.4, linewidth=0.4)
plt.show()
```



[148]:
```
# Visualizing top 20 weekly sales across stores

#Creating a dataframme of top 20 weekly sales
top_20_sales = df[["date", "store","weekly_sales"]].sort_values(
    "weekly_sales", ascending=False
).head(20).reset_index()

# We need the X-axis on the chart to show the store name and the corresponding␣
↪week
top_20_sales["store_date"] = top_20_sales["store"].
↪astype(str)+"_"+top_20_sales["date"].astype(str)

plt.figure(figsize=(12,4))
sns.barplot(
    x="store_date",
    y="weekly_sales",
```

```
    data=top_20_sales,
    palette="crest_r"
)
plt.xlabel("store & date")
plt.ylabel("weekly_sales")
plt.xticks(rotation=90)
plt.grid(axis="y", alpha=0.4, linewidth=0.4)
plt.title("Top 20 Sales Week Across Stores")
plt.show()
```

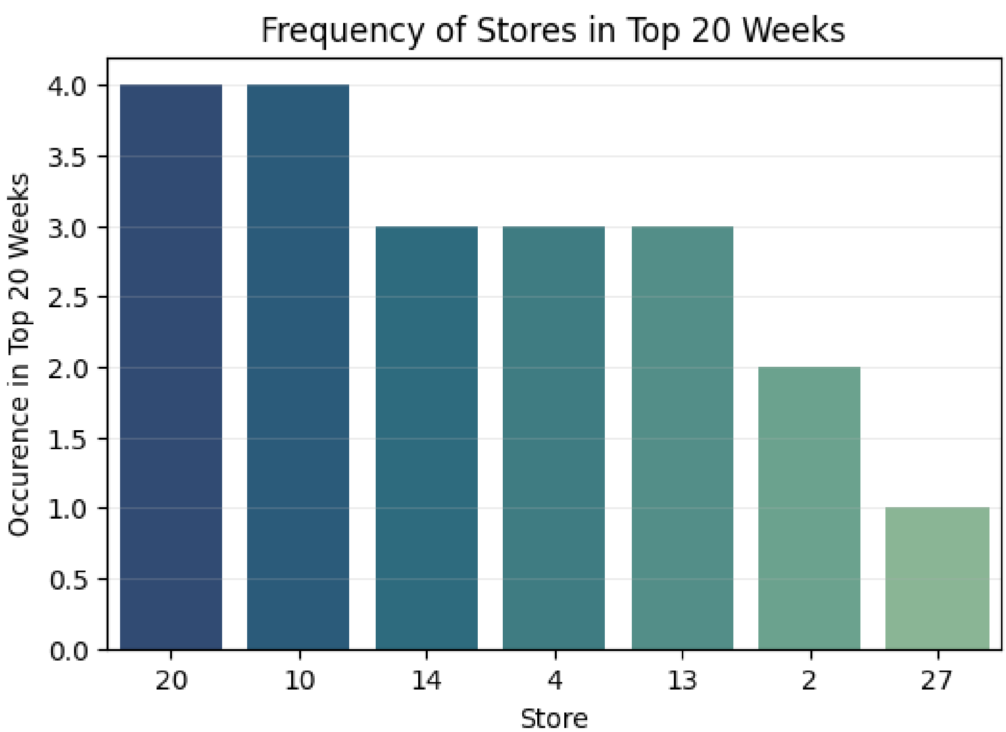


- **We can observe that the 20 highest sales recorded are in the month of December and November**
- **In fact, the 13 highest sales are from the month of December and the date is 23rd and 24th**
- **This does not come as a surprise as the sales are going to be at the highest during Christmas holidays**
- **Store 14 recorded the highest sales ever in the week of "2010-12-24"**

[135]:
```
top_20_sales["store"].value_counts()
```

[135]:
```
store
20    4
10    4
14    3
4     3
13    3
2     2
27    1
Name: count, dtype: int64
```

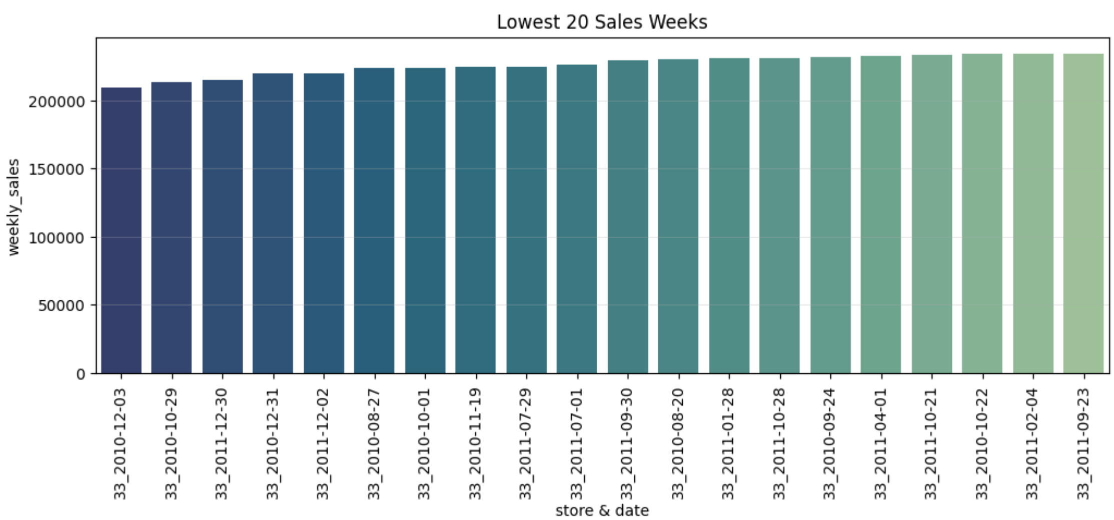
[137]:
```python
plt.figure(figsize=(6,4))

sns.countplot(
    x=top_20_sales["store"],
    order=top_20_sales["store"].value_counts().index,
    palette="crest_r"
)
plt.xlabel("Store")
plt.ylabel("Occurence in Top 20 Weeks")
plt.title("Frequency of Stores in Top 20 Weeks")
plt.grid(axis="y", alpha=0.4, linewidth=0.4)
plt.show()
```



- stores 20 and store 10 are repeated 4 times in the top 20 weeks
- Stores 20 also has the highest total sales and store 10 is ranked 6th in the highest total sales
- stores 4 and 14 are repeated 3 times in the top 20 weeks
- These stores and also ranked 2nd and 3rd respectively in the highest total sales

[146]:
```python
# Visualizing 20 lowest sales week

# Creating a dataframe of lowest 20 sales
lowest_20_sales = df[["date", "store","weekly_sales"]].sort_values(
```

```
    "weekly_sales"
).head(20).reset_index()

# We need store and the corresponding week on the x-axis
lowest_20_sales["store_date"] = lowest_20_sales["store"].
 ↪astype(str)+"_"+lowest_20_sales["date"].astype(str)

plt.figure(figsize=(12,4))
sns.barplot(
    x="store_date",
    y="weekly_sales",
    data=lowest_20_sales,
    palette="crest_r"
)
plt.xlabel("store & date")
plt.ylabel("weekly_sales")
plt.title("Lowest 20 Sales Weeks")
plt.xticks(rotation=90)
plt.grid(axis="y", alpha=0.5, linewidth=0.4)
plt.show()
```



- **Store 33 consistently records lowest sales**
- **All 20 lowest sales week recorded are from store 33**
- **Store 33 also has the lowest total sum of sales**
- **This is the worst performing store**

[141]:
```
store_stats.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45 entries, 0 to 44
```

```
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   store              45 non-null     int64
 1   weekly_sales_mean  45 non-null     float64
 2   weekly_sales_std   45 non-null     float64
 3   weekly_sales_min   45 non-null     float64
 4   weekly_sales_max   45 non-null     float64
 5   weekly_sales_sum   45 non-null     float64
dtypes: float64(5), int64(1)
memory usage: 2.2 KB
```

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

[56]:
```python
# Determining correlation between weekly sales and unemployment
correlations = []
for store in df["store"].unique():
    store_data = df[df["store"] == store]
    store_corr = store_data["weekly_sales"].corr(store_data["unemployment"])
    correlations.append({"store": store, "correlation": store_corr})
```

[58]:
```python
corr_df = pd.DataFrame(correlations)
```

[60]:
```python
corr_df = corr_df.sort_values(by="correlation", ascending=True)
```

[62]:
```python
# Top 5 stores that are negatively affected by unemployment
corr_df.head()
```

[62]:

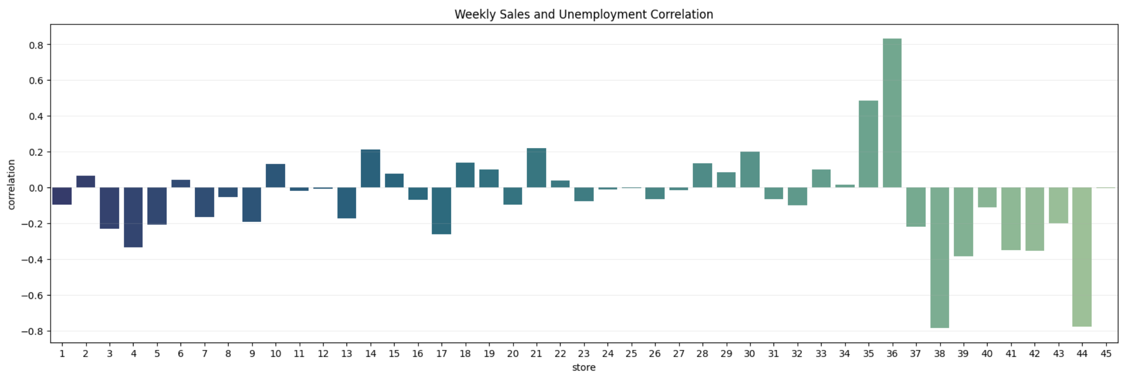
	store	correlation
37	38	-0.785290
43	44	-0.780076
38	39	-0.384681
41	42	-0.356355
40	41	-0.350630

[64]:
```python
# Top 5 stores that are positively affected by unemployment
corr_df.tail()
```

[64]:

	store	correlation
29	30	0.201862
13	14	0.210786
20	21	0.218367
34	35	0.483865
35	36	0.833734

[66]:
```python
# Visualizing the stores sales and unemployment correlation
plt.figure(figsize=(20,6))
sns.barplot(
    x="store",
    y="correlation",
    data=corr_df,
    palette="crest_r"
)
plt.xlabel("store")
plt.ylabel("correlation")
plt.title("Weekly Sales and Unemployment Correlation")
plt.grid(axis="y", linewidth=0.4, alpha=0.4)
plt.show()
```



[68]:
```python
print("Stores affected most by unemployment rate \n")

print(corr_df.head(2))
print("\n Store 37 & 43 have the highest negative correlation")
print("\n That means as the unemployment rate increases, weekly sales decrease␣
 ↪\n")

print(corr_df.tail(2))
print("\n Store 34 & 35 have the highest positive correlation")
print("\n That means as the unemployment rate increases, weekly sales also␣
 ↪increase")
```

```
Stores affected most by unemployment rate

    store  correlation
37     38    -0.785290
43     44    -0.780076

 Store 37 & 43 have the highest negative correlation
```

```
That means as the unemployment rate increases, weekly sales decrease
```

```
    store  correlation
34     35     0.483865
35     36     0.833734
```

```
Store 34 & 35 have the highest positive correlation

That means as the unemployment rate increases, weekly sales also increase
```

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

[78]:
```python
# Aggregating the data for all the stores per week
weekly_sales_grp = df.groupby([df['date']]).agg(
    avg_weekly_sales=("weekly_sales", "mean"))
```

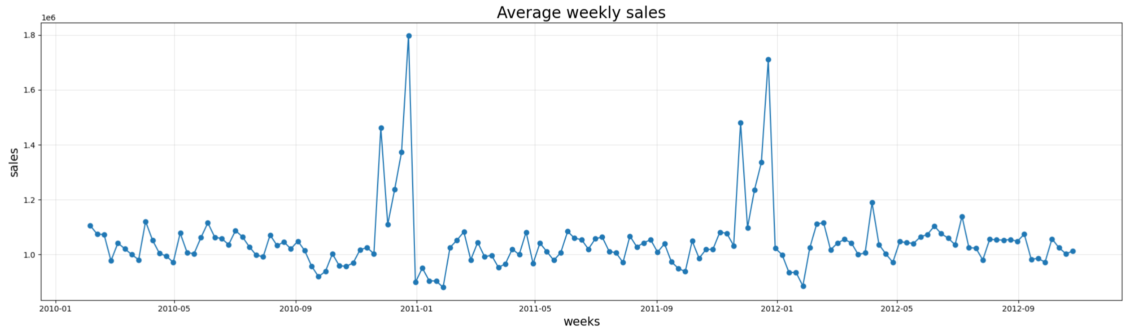
[80]:
```python
weekly_sales_grp.head()
```

[80]:
```
            avg_weekly_sales
date
2010-02-05      1.105572e+06
2010-02-12      1.074148e+06
2010-02-19      1.072822e+06
2010-02-26      9.770794e+05
2010-03-05      1.041588e+06
```

[76]:
```python
weekly_sales_grp.shape
```

[76]:
```
(143, 1)
```

[94]:
```python
#Visualizing Average weekly sales over time
plt.figure(figsize=(20,6))
plt.plot(weekly_sales_grp, marker="o")
plt.xlabel("weeks", fontsize=15)
plt.ylabel("sales", fontsize=15)
plt.title("Average weekly sales", fontsize=20)
plt.grid(linewidth=0.5, alpha=0.5)
plt.tight_layout()
plt.show()
```



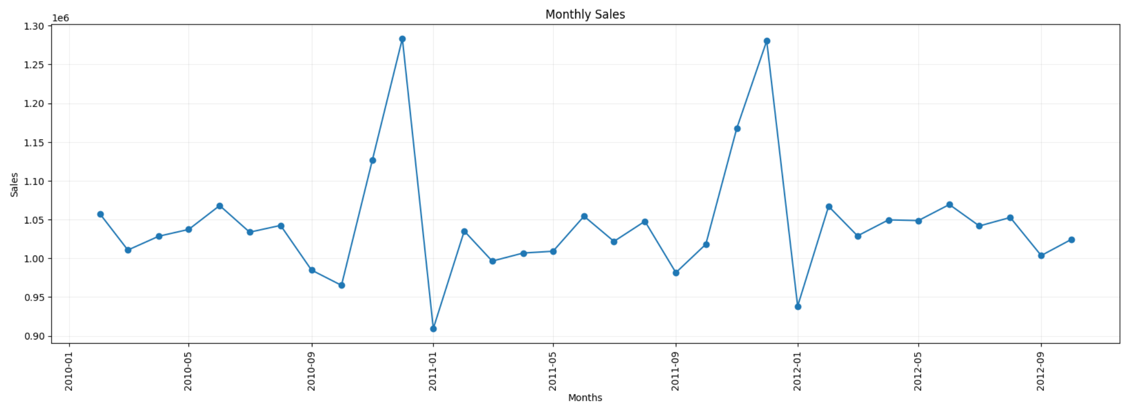
[102]:
```
#Creating year and month column
df["year"] = df["date"].dt.year
df["month"] = df["date"].dt.month

#Aggregating the data by year and month
ym_sales = df.groupby(["year", "month"]).agg(
    monthly_sales = ("weekly_sales", "mean")
).reset_index()

#Creating a date column from year and month
ym_sales["date"] = pd.to_datetime(ym_sales[["year", "month"]].assign(day=1))

#Drop the year and month column as they are no longer needed
ym_sales.drop(columns=["year", "month"], inplace=True)

#Assigning date as index column and dropping the date column
ym_sales.index = ym_sales["date"]
ym_sales.drop(columns=["date"], inplace=True)

ym_sales.head()
```

[102]:

	monthly_sales
date	
2010-02-01	1.057405e+06
2010-03-01	1.010666e+06
2010-04-01	1.028499e+06
2010-05-01	1.037283e+06
2010-06-01	1.068034e+06

[104]:
```
#Visualizing Monthly sales over time
plt.figure(figsize=(20,6))
plt.plot(ym_sales, marker="o")
plt.xlabel("Months")
plt.ylabel("Sales")
plt.title("Monthly Sales")
```

```
plt.grid(linewidth=0.4, alpha=0.4)
plt.xticks(rotation=90)
plt.show()
```



From the monthly sales data, we can observe that the sales increase significantly in November and peak in December. This is the holiday season. With sales on offer during Black Friday, Thanksgiving, and Christmas, this comes as no surprise

1.7 c. Does temperature affect the weekly sales in any manner?

[108]:
```
df.head()
```

[108]:

	store	date	weekly_sales	holiday_flag	temperature	fuel_price	\
0	1	2010-02-05	1643690.90	0	42.31	2.572	
1	1	2010-02-12	1641957.44	1	38.51	2.548	
2	1	2010-02-19	1611968.17	0	39.93	2.514	
3	1	2010-02-26	1409727.59	0	46.63	2.561	
4	1	2010-03-05	1554806.68	0	46.50	2.625	

	cpi	unemployment	year	month
0	211.096358	8.106	2010	2
1	211.242170	8.106	2010	2
2	211.289143	8.106	2010	2
3	211.319643	8.106	2010	2
4	211.350143	8.106	2010	3

[110]:
```
# Aggregating data based on average temperature and average sales
date_grpby = df.groupby(["date"]).agg(
    avg_temperature=("temperature", "mean"), avg_sales=("weekly_sales", "mean")
).reset_index()
```

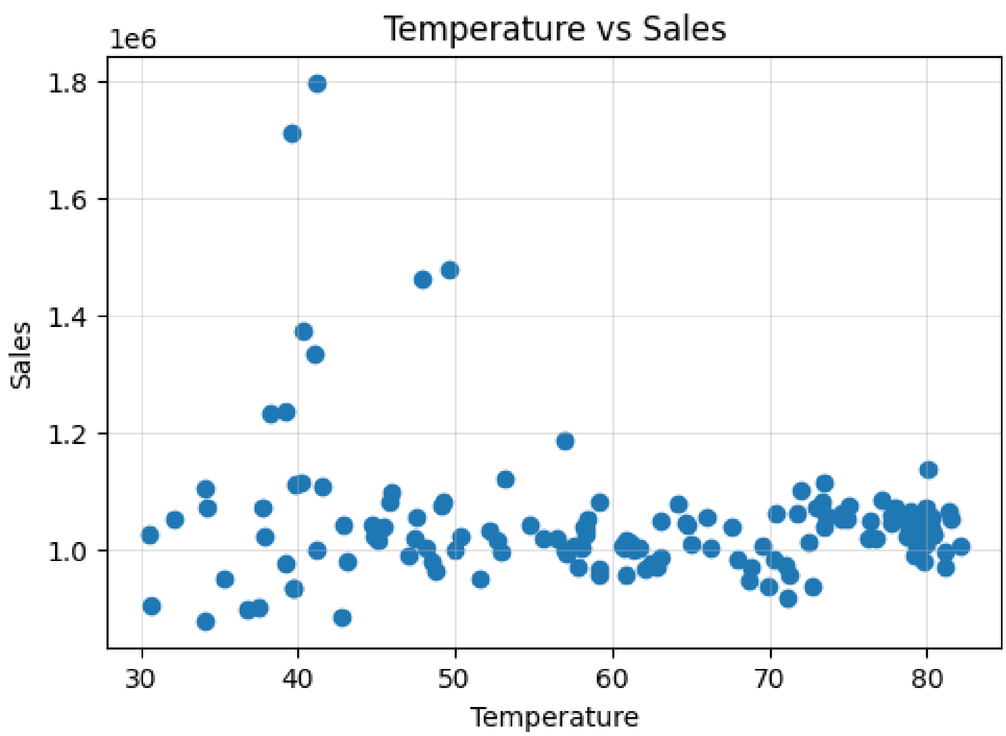
[112]:
```
date_grpby.head()
```

[112]:

```
         date  avg_temperature     avg_sales
0  2010-02-05        34.037333  1.105572e+06
1  2010-02-12        34.151333  1.074148e+06
2  2010-02-19        37.719778  1.072822e+06
3  2010-02-26        39.243556  9.770794e+05
4  2010-03-05        42.917333  1.041588e+06
```

[114]:

```
#Visualizing Temperature and Sales relationship
plt.figure(figsize=(6,4))
plt.scatter(x=date_grpby["avg_temperature"], y=date_grpby["avg_sales"])
plt.xlabel("Temperature")
plt.ylabel("Sales")
plt.title("Temperature vs Sales")
plt.grid(linewidth=0.5, alpha=0.5)
plt.show()
```



From the above scatter plot, we can observe that sales remain constant more or less as the temperature increases. However, there are few exceptions around 40 and 50 degrees, as we observe the peak sales during these temperatures

[117]:

```
# Verify the correlation
date_grpby["avg_temperature"].corr(date_grpby["avg_sales"])
```

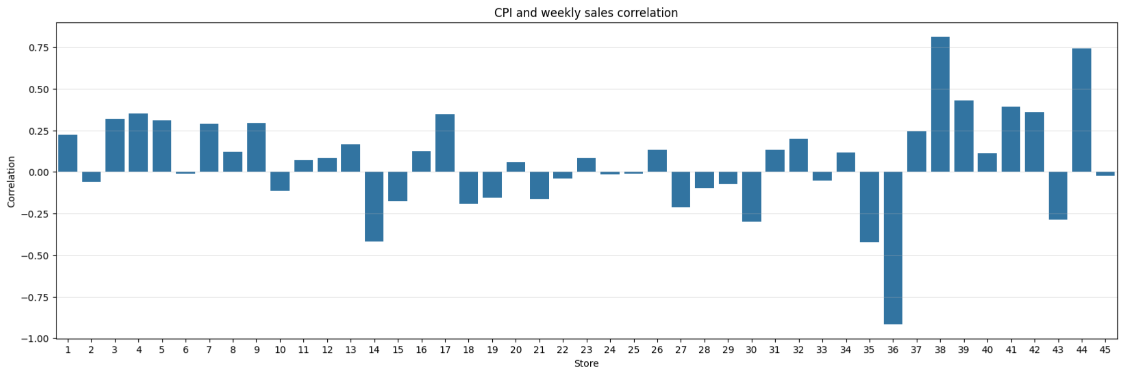
[117]: -0.15915988004722792

The correlation is extremely weak(almost non-existent) as expected

1.8 D. How is the Consumer Price index affecting the weekly sales of various stores?

[123]:
```
# Correlation of cpi and sales for each store
correlation = []
for store in df["store"].unique():
    df_store = df[df["store"]==store]
    cpi_corr = df_store["cpi"].corr(df_store["weekly_sales"])
    correlation.append({"store": store, "correlation": cpi_corr})

cpi_corr_df = pd.DataFrame(correlation).sort_values("correlation",␣
 ↪ascending=False)

print(cpi_corr_df.head())
print(cpi_corr_df.tail())

#Visualizing correlation
plt.figure(figsize=(20,6))
sns.barplot(x="store", y="correlation", data=cpi_corr_df)
plt.xlabel("Store")
plt.ylabel("Correlation")
plt.title("CPI and weekly sales correlation")
plt.grid(axis="y", linewidth=0.5, alpha=0.5)
plt.show()
```

```
    store  correlation
37     38     0.812837
43     44     0.740150
38     39     0.428043
40     41     0.392293
41     42     0.360859
    store  correlation
42     43    -0.285686
29     30    -0.298188
13     14    -0.419755
34     35    -0.424107
35     36    -0.915095
```



There is a strong positive correlation between CPI and weekly sales for store 38 and a moderate positive correlation for store 44. This means as the cpi increases, sales also increase

Whereas store 36 strongly correlates negatively with CPI and weekly sales. That means as the cpi increases, sales decrease

For the remaining stores, there is not so strong correlation between cpi and weekly sales

1.9 E. Top performing stores according to the historica data.

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

[126]:
```
# Top perfroming stores
top_stores = df.groupby(["store"]).agg(
    total_sales=("weekly_sales", "sum")
).reset_index().sort_values(
    "total_sales", ascending=False
)
```

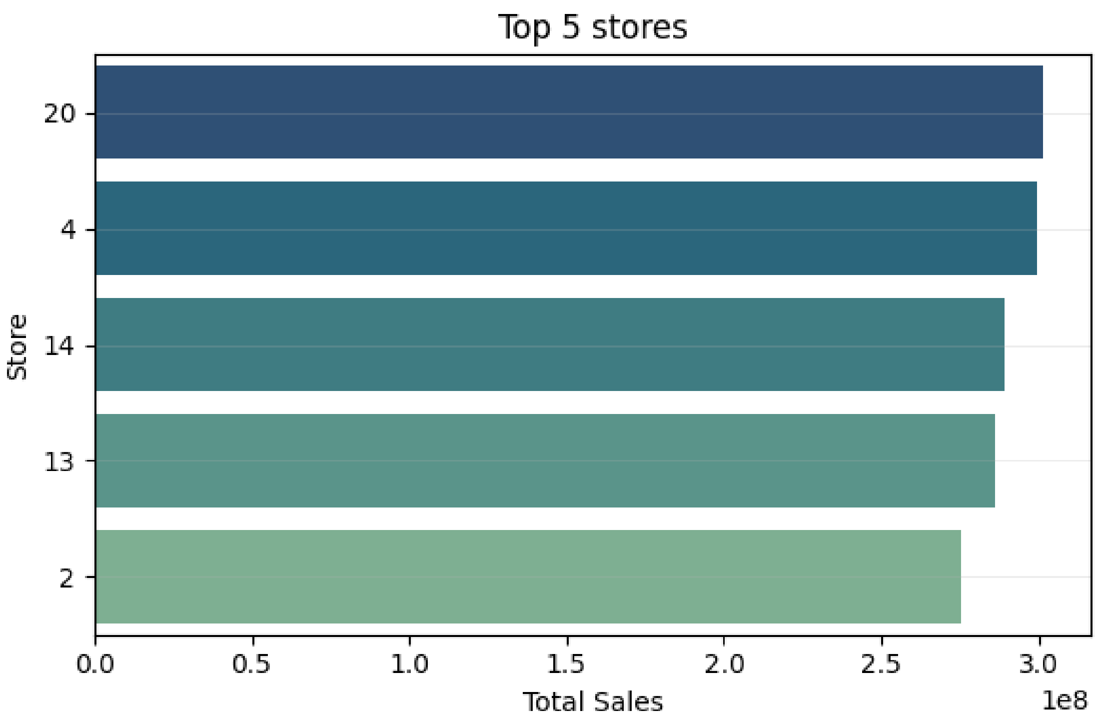
[128]:
```
# Top 5 best perfroming stores
best_stores = top_stores.head()
best_stores
```

[128]:

	store	total_sales
19	20	3.013978e+08
3	4	2.995440e+08
13	14	2.889999e+08
12	13	2.865177e+08
1	2	2.753824e+08

[132]:
```
#Visualizing Top 5 stores and their sales
plt.figure(figsize=(6,4))
```

```
best_stores["store"] = best_stores["store"].astype("category")
sns.barplot(x=best_stores["total_sales"], y=best_stores["store"],␣
 ↪order=best_stores["store"], palette="crest_r")
plt.xlabel("Total Sales")
plt.ylabel("Store")
plt.title("Top 5 stores")
plt.grid(axis="y", linewidth=0.4, alpha=0.4)
plt.tight_layout()
plt.show()
```



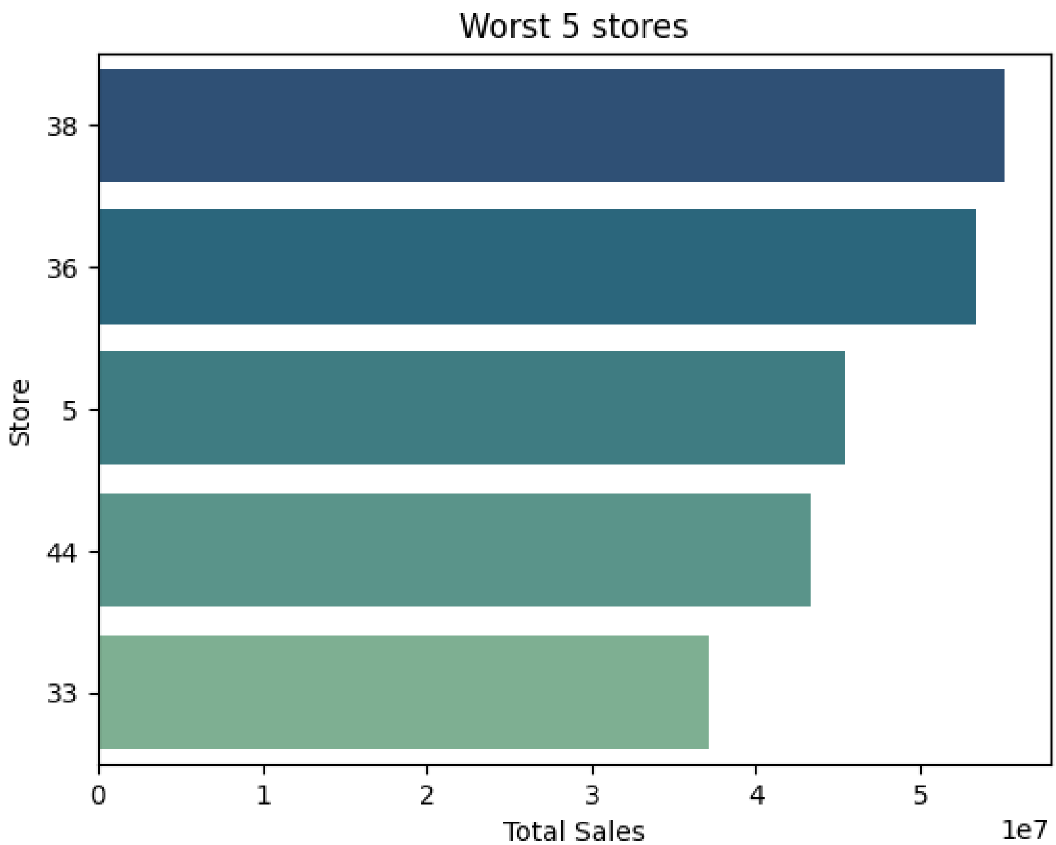
[134]:
```
# The worst 5 performing stores
worst_stores = top_stores.tail()
worst_stores
```

[134]:

	store	total_sales
37	38	55159626.42
35	36	53412214.97
4	5	45475688.90
43	44	43293087.84
32	33	37160221.96

[138]:
```
#Visualizing the 5 lowest performing stores
worst_stores["store"] = worst_stores["store"].astype("category")
```

```
sns.barplot(data=worst_stores, x="total_sales", y="store",␣
    ↪order=worst_stores["store"], palette="crest_r")
plt.xlabel("Total Sales")
plt.ylabel("Store")
plt.title("Worst 5 stores")
plt.show()
```



[140]:
```
# Difference in sales of the top 5 and the worst 5 stores
diff_sales = best_stores["total_sales"].sum() - worst_stores["total_sales"].
    ↪sum()
print(f"Differnece in sales performance of the top 5 and the worst 5 stores is:␣
    ↪{diff_sales}")
```

```
Differnece in sales performance of the top 5 and the worst 5 stores is:
1217340961.87
```

1.11 Model Buliding Preprocess

For model building we will need the date and the weekly_sales column. So we will store these columns in a new data frame

[144]:
```python
# We have 45 stores. We will take input from the user on the store number
a = int(input("Enter store number:"))
store = df[df["store"]==a]
```

```
Enter store number: 1
```

[146]:
```python
store.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 143 entries, 0 to 142
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   store         143 non-null    int64
 1   date          143 non-null    datetime64[ns]
 2   weekly_sales  143 non-null    float64
 3   holiday_flag  143 non-null    int64
 4   temperature   143 non-null    float64
 5   fuel_price    143 non-null    float64
 6   cpi           143 non-null    float64
 7   unemployment  143 non-null    float64
 8   year          143 non-null    int32
 9   month         143 non-null    int32
dtypes: datetime64[ns](1), float64(5), int32(2), int64(2)
memory usage: 11.2 KB
```

For time series modeling, we need only the date and weekly_sales column and date column as index

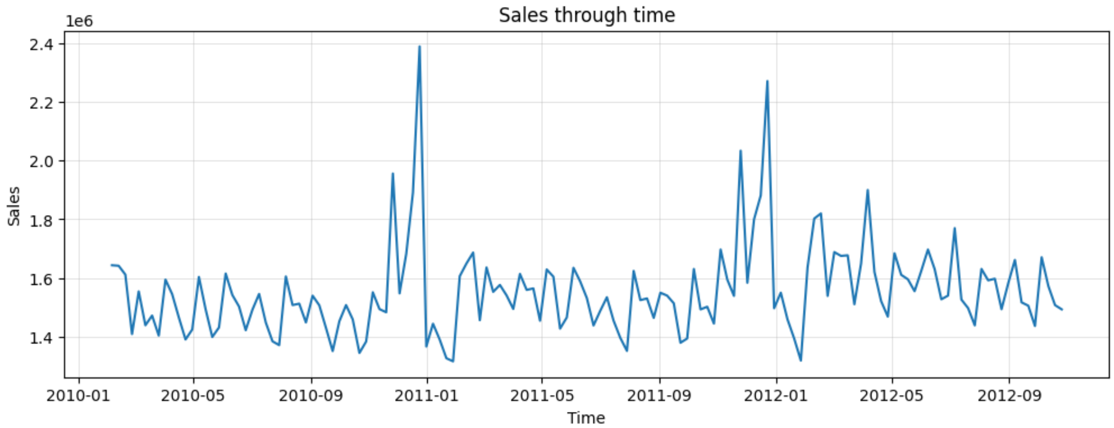
[149]:
```python
store = store.loc[:, ["date", "weekly_sales"]].set_index("date")
```

[151]:
```python
store.head()
```

[151]:

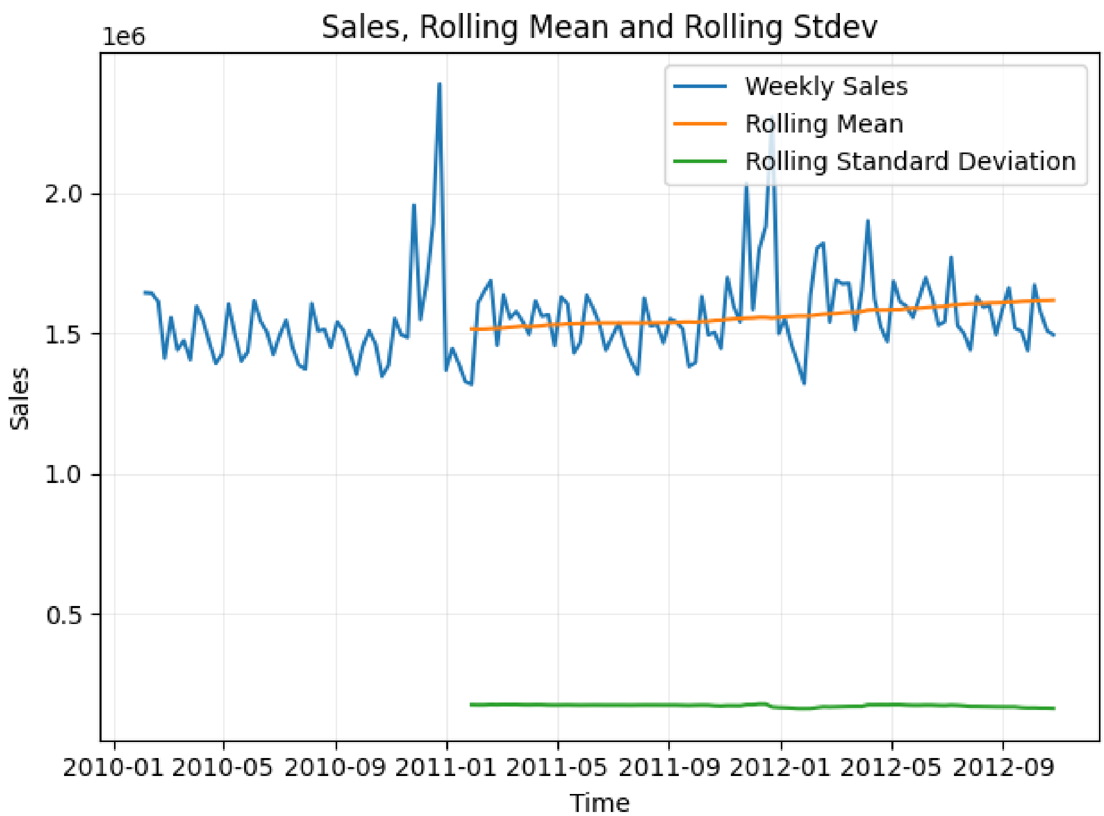
date	weekly_sales
2010-02-05	1643690.90
2010-02-12	1641957.44
2010-02-19	1611968.17
2010-02-26	1409727.59
2010-03-05	1554806.68

[153]:
```python
#Visualizing Sales through time
plt.figure(figsize=(12,4))
plt.plot(store["weekly_sales"])
plt.xlabel("Time")
plt.ylabel("Sales")
plt.title("Sales through time")
plt.grid(linewidth=0.5, alpha=0.5)
plt.show()
```



[161]:

```
# Let us calculate the rolling mean and rolling standard deviation
rol_mean = store.rolling(window=52).mean().dropna()
rol_std = store.rolling(window=52).std().dropna()

plt.plot(store["weekly_sales"], label="Weekly Sales")
plt.plot(rol_mean, label="Rolling Mean")
plt.plot(rol_std, label="Rolling Standard Deviation")
plt.xlabel("Time")
plt.ylabel("Sales")
plt.legend(loc="upper right")
plt.title("Sales, Rolling Mean and Rolling Stdev")
plt.grid(linewidth=0.4, alpha=0.4)
plt.tight_layout()
plt.show()
```



For time series modeling we need mean as constant and standard deviation as 0. The mean just shows a gradual increase We can also verify this with AdFuller Test

[163]:
```
from statsmodels.tsa.stattools import adfuller
```

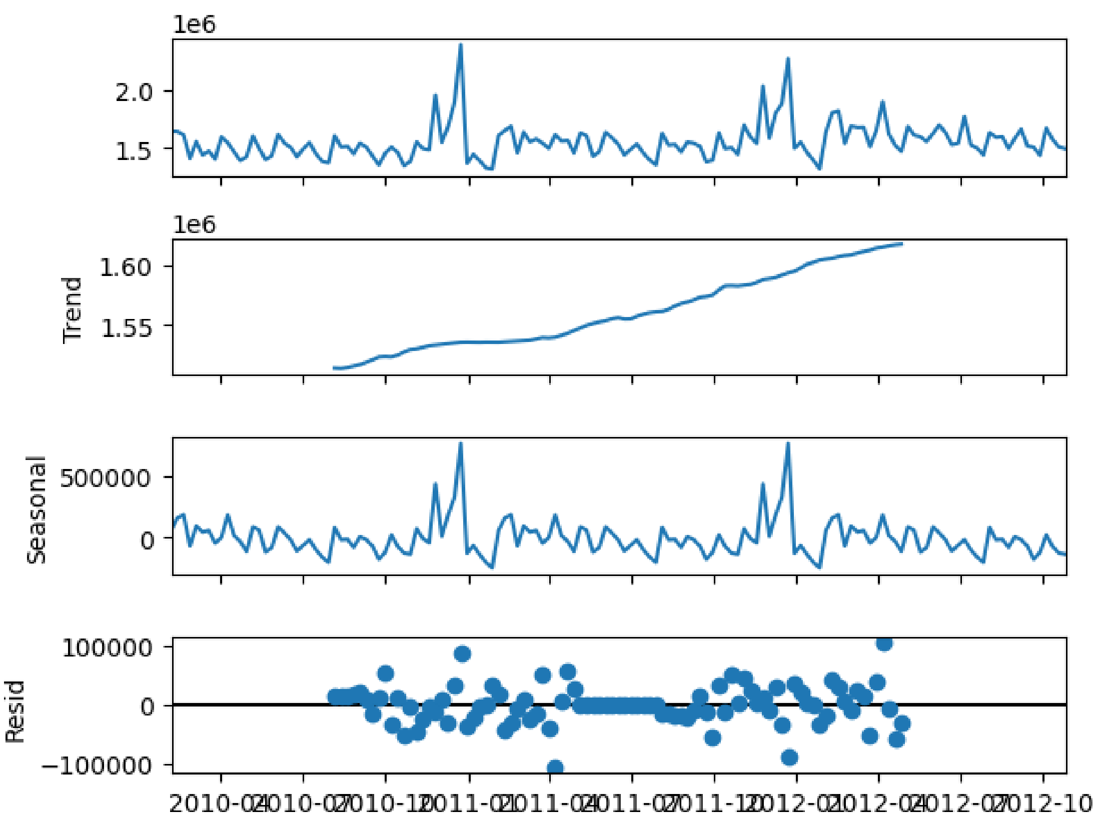
[165]:
```
def check_stationarity(timeseries):
    stationarity = adfuller(timeseries, autolag="AIC")
    print(f"ADF statistic: {stationarity[0]}")
    print(f"Pvalue: {stationarity[1]}")
    for key, value in stationarity[4].items():
        print("Critical values")
        print(f"{key}, {value}")
    print(f"lag value: {stationarity[2]}")
    print(f"nobs: {stationarity[3]}")

    if stationarity[1]<0.05:
        print("Series is stationary")
    else:
        print("series is not stationary")
```

[167]:
```
check_stationarity(store)
```

```
ADF statistic: -5.102186145192288
Pvalue: 1.3877788330759434e-05
Critical values
1%, -3.47864788917503
Critical values
5%, -2.882721765644168
Critical values
10%, -2.578065326612056
lag value: 4
nobs: 138
Series is stationary
```

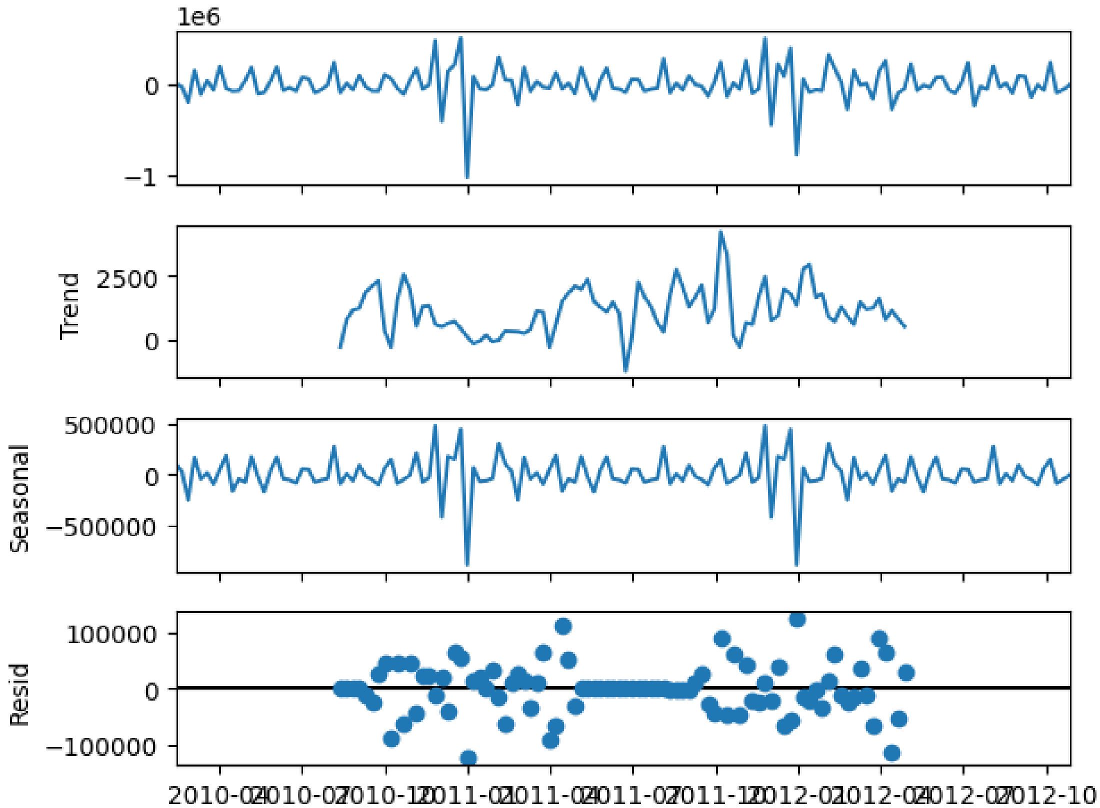
[169]:
```
# Let's decompose the time series into trend, seasonality and residuals
from statsmodels.tsa.seasonal import seasonal_decompose
```

[171]:
```
# Decomposition visualization
decomposition = seasonal_decompose(store, period=52)
decomposition.plot();
```

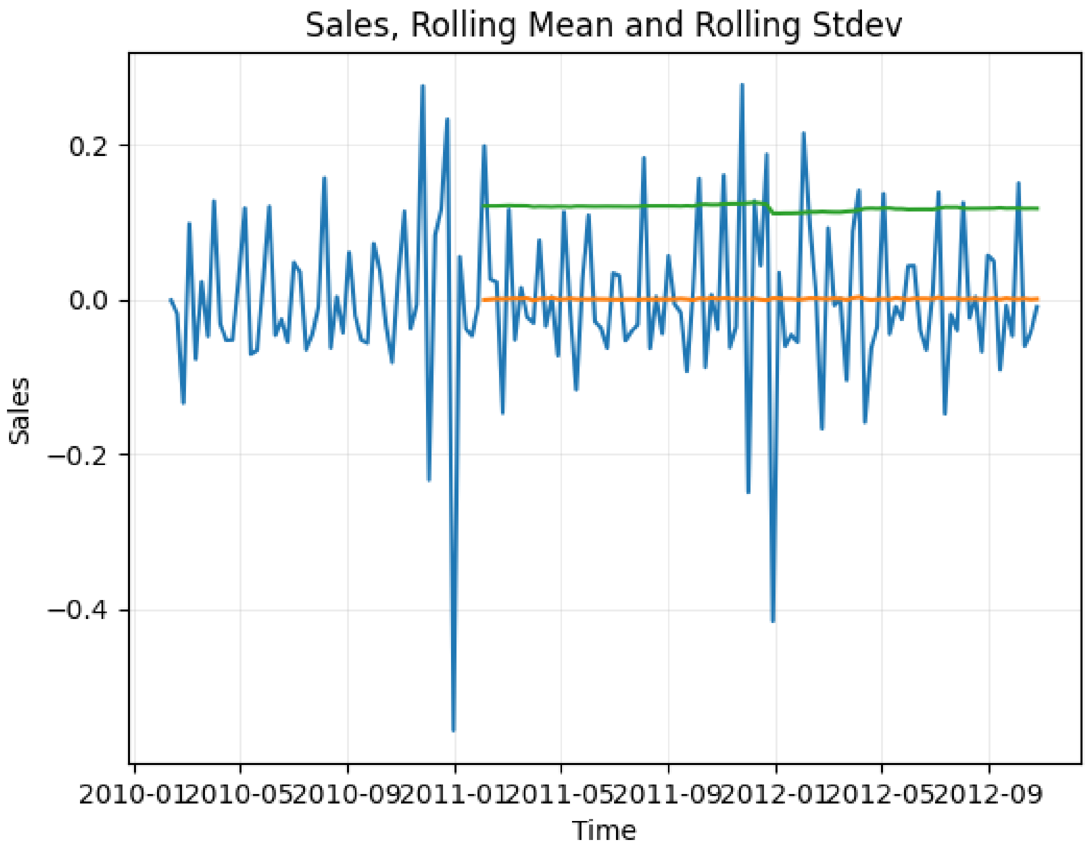


Adfuller test shows the series is stationarity, however, our decomposition plot shows a trend. This is counter-intuitive. If the series is stationary, no trends can be observed but in this case, we observe the trend inspite of stationarity

[183]:
```python
# We will do one differencing and observe the trend
store_diff = store.diff().dropna()

decompose = seasonal_decompose(store_diff, period=52)
decompose.plot();
```



[181]:
```python
# Checking the rolling mean and standard deviaiton of differenced series
rol_mean = store_diff.rolling(window=52).mean().dropna()
rol_std = store_diff.rolling(window=52).std().dropna()

plt.plot(store_diff["weekly_sales"])
plt.plot(rol_mean)
plt.plot(rol_std)
plt.xlabel("Time")
plt.ylabel("Sales")
plt.title("Sales, Rolling Mean and Rolling Stdev")
plt.grid(linewidth=0.4, alpha=0.4)
plt.show()
```

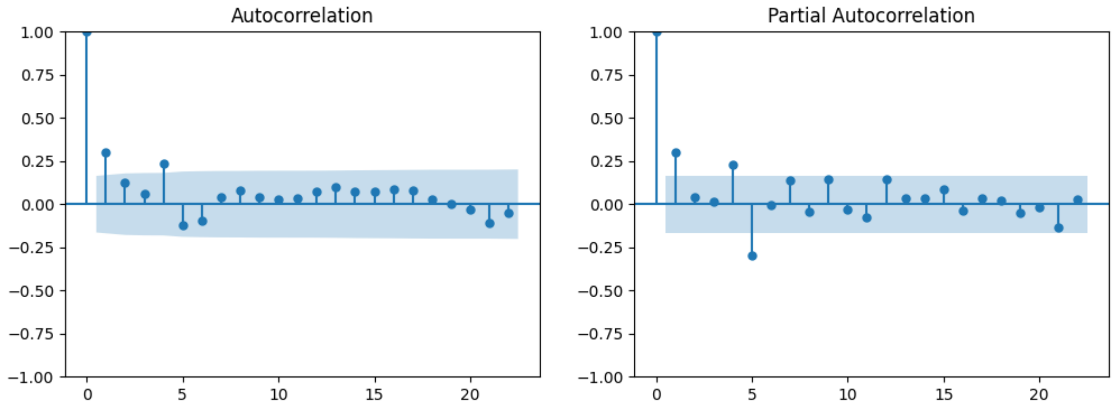


The decomposition plot after one differencing does not show any trend

However, our Adfuller test shows our original data is stationarity and no differencing is required to achieve the stationarity. We will use auto arima to determine the p,d, and q values. This will also provide us with the differencing required, if any. For the moment, we will take the original data as stationary, hence d=0 and proceed to determine the values of p and q

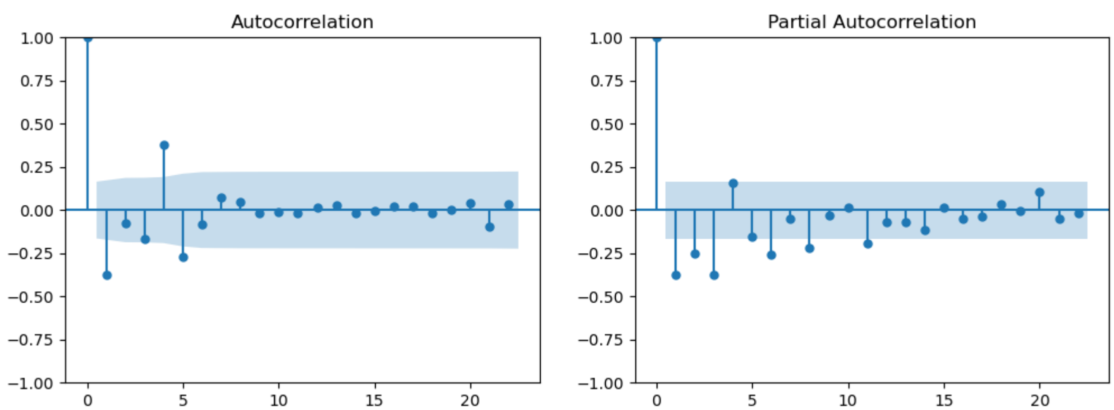
[189]:
```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

[191]:
```
# Plotting acf and pacf plots to determine the values of p and q
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12,4))

plot_acf(store, ax1)
plot_pacf(store, ax2)
plt.show()
```



- **For determing P, we use pacf plot. The pacf vlaue is significant at lag 1 and then dies down and it is again significant at lag 4**
- **For determining Q, we use acf plot. The acf value is significant at lag 1 and at lag 4**

[396]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12,4))

plot_acf(store_diff_2, ax1)
plot_pacf(store_diff_2, ax2)
plt.show()
```



[194]:
```
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
import itertools
```

[196]:
```
#Determining p and q values via iteration
# From acf and pacf plots, we know that p and q values will lie generally in
 ↪the range of (0,5)
```

```python
# And we also know that differencing the series once will eliminate the trend,␣
↪so we know d=1
d=1
p=q=range(0,5)
pq = list(itertools.product(p,q))
model_list=[]
for x in pq:
    comb = list(x)
    comb.insert(1,d)
    comb=tuple(comb)
    model_list.append(comb)
model_list
```

[196]:
```
[(0, 1, 0),
 (0, 1, 1),
 (0, 1, 2),
 (0, 1, 3),
 (0, 1, 4),
 (1, 1, 0),
 (1, 1, 1),
 (1, 1, 2),
 (1, 1, 3),
 (1, 1, 4),
 (2, 1, 0),
 (2, 1, 1),
 (2, 1, 2),
 (2, 1, 3),
 (2, 1, 4),
 (3, 1, 0),
 (3, 1, 1),
 (3, 1, 2),
 (3, 1, 3),
 (3, 1, 4),
 (4, 1, 0),
 (4, 1, 1),
 (4, 1, 2),
 (4, 1, 3),
 (4, 1, 4)]
```

[198]:
```python
def arima_optimizer(data, pdq_range):
    best_aic = float('inf')
    best_order = None
    for order in pdq_range:
        try:
            model=ARIMA(data, order=order)
            result=model.fit()
            if result.aic<best_aic:
```

```
                best_aic, best_order = result.aic, order
        except:
            continue
    return best_order
```

[200]:
```
best_arima = arima_optimizer(store, model_list)
print(f"Best arima model: {best_arima}")
```

```
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
```

```
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
```

```
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
```

```
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
```

```
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
```

```
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
```

```
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
```

```
Best arima model: (2, 1, 3)
```

Best arima model predicted is (2,1,3) i.e p=2, d=1 and q=3

We will also use auto_arima to verify this

[204]:
```python
from pmdarima import auto_arima
```

[206]:
```python
arima_model = auto_arima(store["weekly_sales"], seasonal=True, trace=True)
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=3819.024, Time=0.18 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=3850.005, Time=0.02 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=3838.724, Time=0.06 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=3831.432, Time=0.04 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=3848.013, Time=0.02 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=3819.410, Time=0.22 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=3821.284, Time=0.11 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=3817.475, Time=0.15 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=3818.117, Time=0.13 sec
 ARIMA(4,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=0.38 sec
 ARIMA(3,1,3)(0,0,0)[0] intercept   : AIC=3810.990, Time=0.16 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=3809.486, Time=0.19 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=3810.431, Time=0.10 sec
 ARIMA(2,1,4)(0,0,0)[0] intercept   : AIC=3810.637, Time=0.24 sec
 ARIMA(1,1,4)(0,0,0)[0] intercept   : AIC=inf, Time=0.26 sec
 ARIMA(3,1,4)(0,0,0)[0] intercept   : AIC=inf, Time=0.49 sec
 ARIMA(2,1,3)(0,0,0)[0]             : AIC=3807.686, Time=0.14 sec
 ARIMA(1,1,3)(0,0,0)[0]             : AIC=3808.792, Time=0.08 sec
```

```
ARIMA(2,1,2)(0,0,0)[0]             : AIC=3817.717, Time=0.10 sec
ARIMA(3,1,3)(0,0,0)[0]             : AIC=3809.251, Time=0.17 sec
ARIMA(2,1,4)(0,0,0)[0]             : AIC=3809.475, Time=0.15 sec
ARIMA(1,1,2)(0,0,0)[0]             : AIC=3818.746, Time=0.12 sec
ARIMA(1,1,4)(0,0,0)[0]             : AIC=3809.305, Time=0.10 sec
ARIMA(3,1,2)(0,0,0)[0]             : AIC=3815.496, Time=0.13 sec
ARIMA(3,1,4)(0,0,0)[0]             : AIC=3810.246, Time=0.38 sec

Best model:  ARIMA(2,1,3)(0,0,0)[0]
Total fit time: 4.145 seconds
```

By reconiling the acf-pacf plots, best order through itertools and auto_arima, all generate the same p,d,q values of (2,1,3).

Hence, we will take the order (2,1,3) for model building

[210]:
```python
from statsmodels.tsa.arima.model import ARIMA
```

[234]:
```python
store.shape
```

[234]: (143, 1)

[236]:
```python
# Splitting the data into train and test
train = store[:120]
test = store[120:]
```

[238]:
```python
#Implementing ARIMA model
model = ARIMA(store, order=(2,1,3))
result = model.fit()
print(result.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:           weekly_sales   No. Observations:                  143
Model:                 ARIMA(2, 1, 3)   Log Likelihood               -1897.843
Date:                Tue, 24 Jun 2025   AIC                           3807.686
Time:                        14:33:00   BIC                           3825.421
Sample:                    02-05-2010   HQIC                          3814.893
                         - 10-26-2012
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.7500      0.127     -5.919      0.000      -0.998      -0.502
ar.L2         -0.3101      0.128     -2.420      0.016      -0.561      -0.059
ma.L1          0.2867      0.118      2.427      0.015       0.055       0.518
ma.L2         -0.3357      0.109     -3.090      0.002      -0.549      -0.123
ma.L3         -0.6403      0.067     -9.574      0.000      -0.771      -0.509
sigma2      2.408e+10   4.97e-12   4.85e+21      0.000    2.41e+10    2.41e+10
```

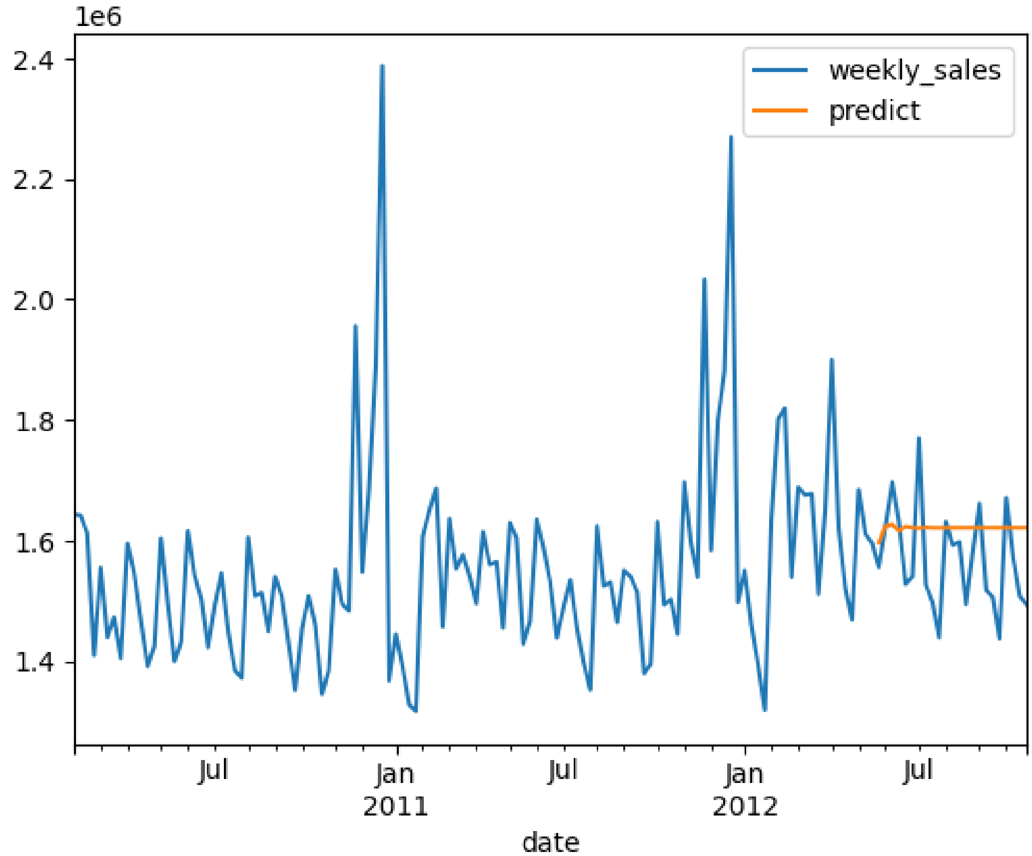
```
================================================================================
===
Ljung-Box (L1) (Q):                   0.07   Jarque-Bera (JB):
44.55
Prob(Q):                              0.80   Prob(JB):
0.00
Heteroskedasticity (H):               0.49   Skew:
0.77
Prob(H) (two-sided):                  0.02   Kurtosis:
5.27
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
[2] Covariance matrix is singular or near-singular, with condition number
3.27e+38. Standard errors may be unstable.
```

```
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
```

[240]:
```
# Predicting the sales for test set
store["predict"] = result.predict(start=len(train), end=len(train)+len(test)-1,␣
 ↪dynamic=True)
```

[242]:
```
store[["weekly_sales", "predict"]].plot()
```

[242]:
```
<Axes: xlabel='date'>
```



ARIMA's predictions have been way off. We will implement SARIMAX now

[247]:
```
from statsmodels.tsa.statespace.sarimax import SARIMAX, SARIMAXResults
```

[249]:
```
# Implement Sarimax
model_sarimax = SARIMAX(store["weekly_sales"], order=(2,1,3),␣
↪seasonal_order=(2,1,3,52))
result_sarimax = model_sarimax.fit()
print(result_sarimax.summary())
```

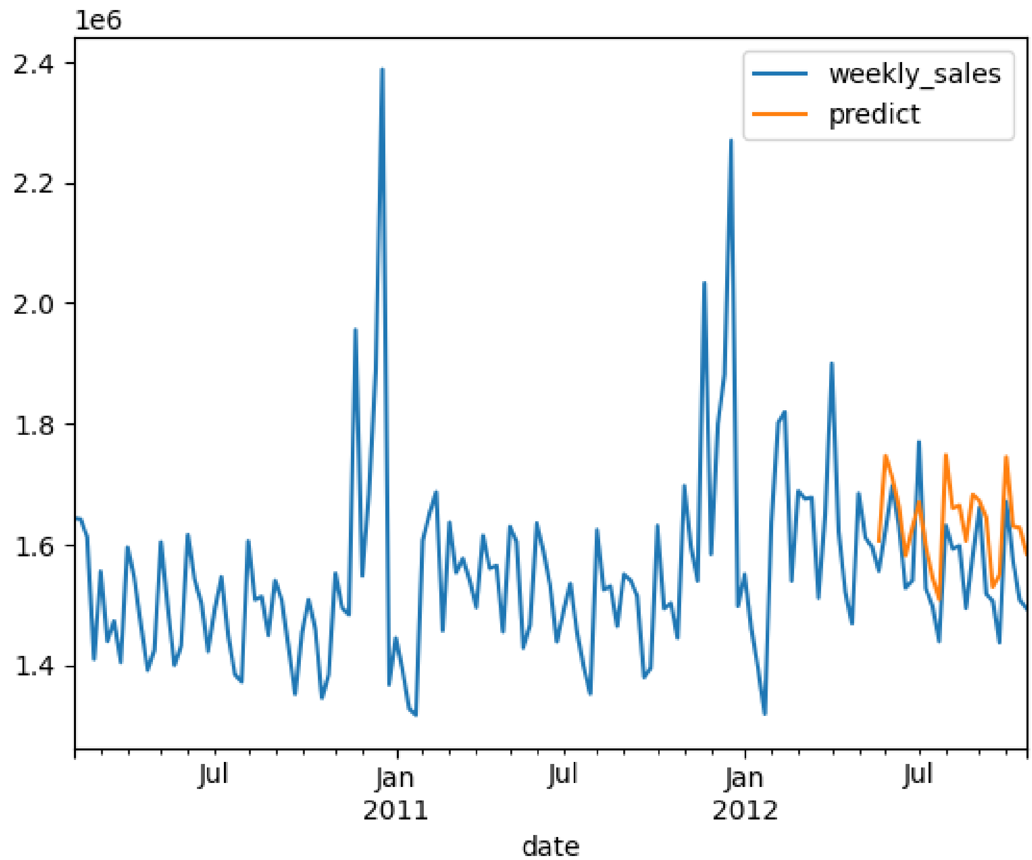
```
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
C:\ProgramData\anaconda3\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency W-FRI will be used.
  self._init_dates(dates, freq)
                                     SARIMAX Results
================================================================================
```

```
==========
Dep. Variable:                        weekly_sales   No. Observations:
143
Model:             SARIMAX(2, 1, 3)x(2, 1, 3, 52)   Log Likelihood
-1140.461
Date:                             Tue, 24 Jun 2025   AIC
2302.922
Time:                                     14:36:20   BIC
2330.420
Sample:                                 02-05-2010   HQIC
2314.011
                                      - 10-26-2012
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.3348      1.090      0.307      0.759      -1.802       2.471
ar.L2          0.1530      0.788      0.194      0.846      -1.392       1.698
ma.L1         -0.5267      1.093     -0.482      0.630      -2.669       1.616
ma.L2         -0.1571      0.864     -0.182      0.856      -1.850       1.536
ma.L3          0.1060      0.183      0.578      0.563      -0.253       0.465
ar.S.L52      -0.1972   4959.570  -3.98e-05      1.000   -9720.776    9720.381
ar.S.L104     -0.0465   3868.412   -1.2e-05      1.000   -7581.994    7581.901
ma.S.L52       0.0116   1.07e+04   1.08e-06      1.000    -2.1e+04     2.1e+04
ma.S.L104      0.0307   1.02e+04   3.02e-06      1.000   -1.99e+04    1.99e+04
ma.S.L156      0.6706    946.220      0.001      0.999   -1853.887    1855.228
sigma2      4.086e+09   3.59e-06   1.14e+15      0.000    4.09e+09    4.09e+09
================================================================================
===
Ljung-Box (L1) (Q):                  11.59   Jarque-Bera (JB):
0.15
Prob(Q):                              0.00   Prob(JB):
0.93
Heteroskedasticity (H):               1.66   Skew:
0.10
Prob(H) (two-sided):                  0.17   Kurtosis:
3.03
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
[2] Covariance matrix is singular or near-singular, with condition number
4.68e+35. Standard errors may be unstable.
```

[251]:
```
# Predicting the sales for test set
store["predict"] = result_sarimax.predict(start=len(train),␣
↪end=len(train)+len(test)-1, dynamic=True)
```

[253]:
```
#Visualiing the predicted sales against actual sets
store[["weekly_sales", "predict"]].plot()
```

[253]: <Axes: xlabel='date'>



[255]:
```
# Isolating the test set
prediction = store.iloc[120:]
```

[257]:
```
prediction
```

[257]:

```
            weekly_sales       predict
date
2012-05-25    1555444.55  1.606381e+06
2012-06-01    1624477.58  1.747055e+06
2012-06-08    1697230.96  1.710746e+06
```

```
2012-06-15    1630607.00  1.662248e+06
2012-06-22    1527845.81  1.581477e+06
2012-06-29    1540421.49  1.628080e+06
2012-07-06    1769854.16  1.670398e+06
2012-07-13    1527014.04  1.598001e+06
2012-07-20    1497954.76  1.545744e+06
2012-07-27    1439123.71  1.508691e+06
2012-08-03    1631135.79  1.748095e+06
2012-08-10    1592409.97  1.659798e+06
2012-08-17    1597868.05  1.664754e+06
2012-08-24    1494122.38  1.606031e+06
2012-08-31    1582083.40  1.682790e+06
2012-09-07    1661767.33  1.672105e+06
2012-09-14    1517428.87  1.644327e+06
2012-09-21    1506126.06  1.529442e+06
2012-09-28    1437059.26  1.549953e+06
2012-10-05    1670785.97  1.745078e+06
2012-10-12    1573072.81  1.630188e+06
2012-10-19    1508068.77  1.627620e+06
2012-10-26    1493659.74  1.584881e+06
```

[259]:

```
#Calculating the mean square error of predicted sales
mse = (np.sum((prediction["predict"] - prediction["weekly_sales"])**2))/
 ↪len(prediction)
print("Mean squared error is:", mse)

rmse = mse**0.5
print("Root Mean Squared error is:", rmse)
```

```
Mean squared error is: 6855290855.330803
Root Mean Squared error is: 82796.68384259603
```

[275]:

```
avg_error = rmse*100/prediction["weekly_sales"].mean()
avg_error
```

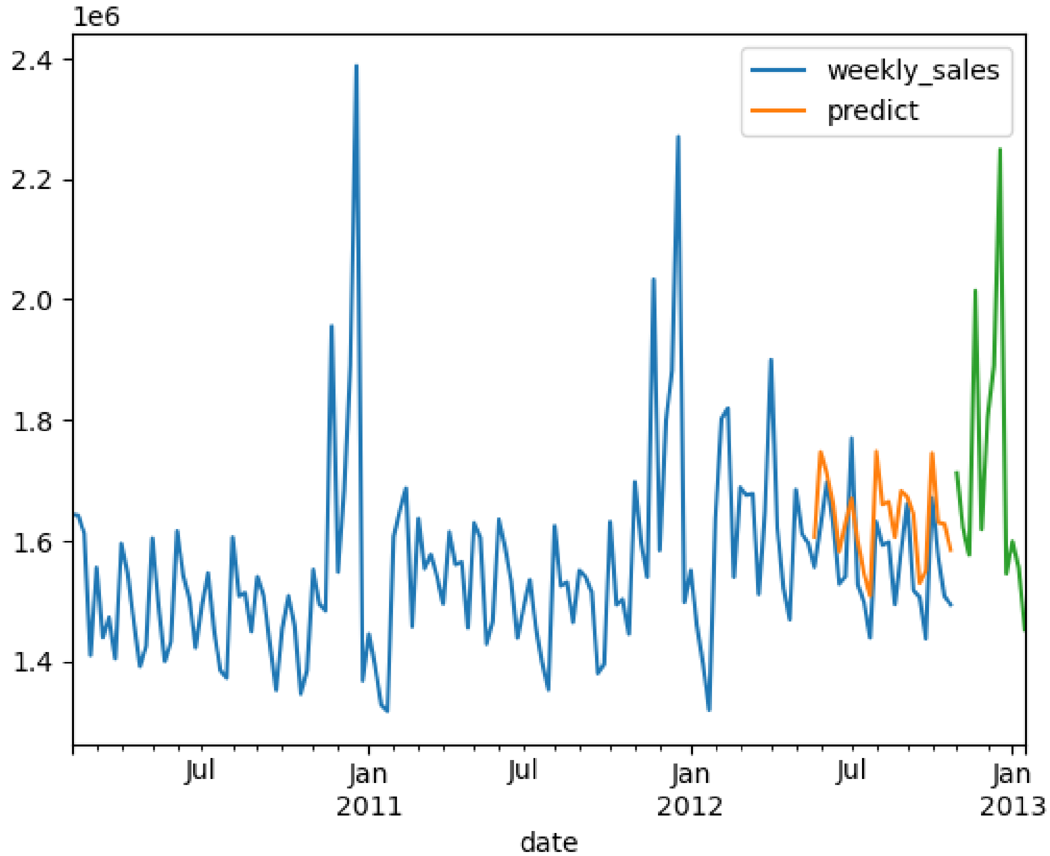
[275]: 5.278708351369967

Average error of prediction is 5.28%

[243]:

```
# Forecasting for the next 12 weeks
forecast = result_sarimax.forecast(steps=12)
store.plot()
forecast.plot()
```

[243]: <Axes: xlabel='date'>



[244]:
```
forecast
```

[244]:
```
2012-11-02    1.711799e+06
2012-11-09    1.620868e+06
2012-11-16    1.576180e+06
2012-11-23    2.014698e+06
2012-11-30    1.618342e+06
2012-12-07    1.804626e+06
2012-12-14    1.890240e+06
2012-12-21    2.249352e+06
2012-12-28    1.545341e+06
2013-01-04    1.599037e+06
2013-01-11    1.554590e+06
2013-01-18    1.452489e+06
Freq: W-FRI, Name: predicted_mean, dtype: float64
```

Summary of insights

- **Store 14 registered the highest single week sales ever while store 33 has the lowest single week sales ever**

- **Store 20 has the highest toal sales just eclipsing 30 Million USD followed closely by store 4. Store 33 has the lowest total sales at under 5 Million USD. Stores 44 and 5 are also languishing at the bottom with total sales less than 5 Million USD**
- **We can observe that the 20 highest sales recorded are in the month of December and November**
- **In fact, the 13 highest sales are from the month of December and the date is 23rd and 24th**
- **This does not come as a surprise as the sales are going to be at the highest during Christmas holidays**
- **Store 14 recorded the highest sales ever in the week of "2010-12-24"**
- **Store 33 consistently records lowest sales**
- **All 20 lowest sales week recorded are from store 33**
- **Store 33 also has the lowest total sum of sales**
- **This is the worst performing store**
- **From the monthly sales data, we can observe that the sales increase significantly in November and peak in December. This is the holiday season. With sales on offer during Black Friday, Thanksgiving, and Christmas, this comes as no surprise**