walmart

January 23, 2022

1 Name: Sunil Pradhan

1.1 Project: 4

1.2 Project Name: Retail Analysis with Walmart Data

[1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[2]:

```
#importing and displaying the dataset
wal=pd.read_csv("Walmart_Store_sales.csv")
wal.head()
```

[2]:

```
   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
```

[3]:

```
#checing info details
wal.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
```

[4]:

```
#converting the data type of date column
wal.Date=pd.to_datetime(wal['Date'])
```

[5]:

```
wal.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
```

[6]:

```
#checking the shape of the dataset
wal.shape
```

[6]:

```
(6435, 8)
```

[7]:

```
#checking for mising values
wal.isnull().sum()
```

[7]:

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

```
dtype: int64
```

[8]:
```
#finding the store has maximum sale

total_sales=pd.DataFrame(wal.groupby('Store')['Weekly_Sales'].sum().
 ↪sort_values())
total_sales
```

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

```
6     2.237561e+08
27    2.538559e+08
10    2.716177e+08
2     2.753824e+08
13    2.865177e+08
14    2.889999e+08
4     2.995440e+08
20    3.013978e+08
```

[9]:

```
print("Store",total_sales.head(1).index[0], "has minimum sales value of: {0:.
↪2f}",float(total_sales.head(1)['Weekly_Sales']))
```

```
Store 33 has minimum sales value of: {0:.2f} 37160221.96
```

[10]:

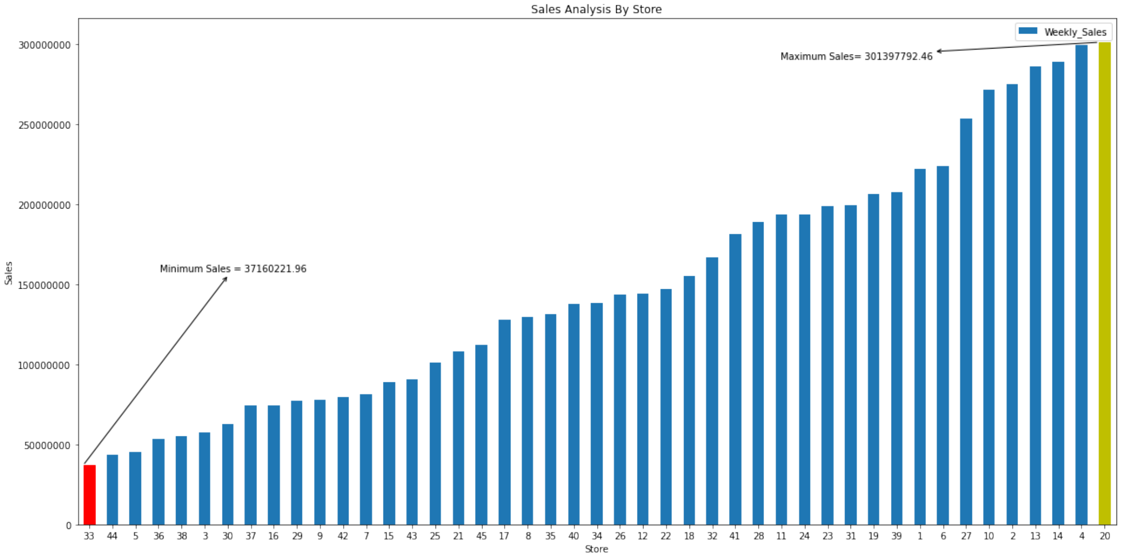
```
print("Store", total_sales.tail(1).index[0], "has maximum sales values of: ",␣
↪float(total_sales.tail(1)['Weekly_Sales']))
```

```
Store 20 has maximum sales values of:  301397792.46
```

[11]:

```
#plot series sales
ax=total_sales.plot(kind='bar',
                   figsize=(20,10))

plt.xticks(rotation=0)
plt.ticklabel_format(useOffset=False, style='plain', axis='y')
plt.title("Sales Analysis By Store")
plt.xlabel('Store')
plt.ylabel('Sales')



#label store with minimum sales
minimum=ax.patches[0]
minimum.set_color('r')
ax.annotate("Minimum Sales = {0:.2f}".format((minimum.get_height())),
           xy=(minimum.get_x(), minimum.get_height()),
           xytext=(0.15,0.5), textcoords='axes fraction',
           arrowprops=dict(arrowstyle="<-", connectionstyle="arc3"),
           horizontalalignment='center')

#label with maximum sales
maximum=ax.patches[len(ax.patches)-1]
maximum.set_color('y')
ax.annotate("Maximum Sales= {0:.2f}".format((maximum.get_height())),
           xy=(maximum.get_x(), maximum.get_height()),
           xytext=(0.75,0.92), textcoords='axes fraction',
           arrowprops=dict(arrowstyle="<-", connectionstyle="arc3"),
           horizontalalignment='center')
```

```
[11]: Text(0.75, 0.92, 'Maximum Sales= 301397792.46')
```



```
[12]: #finding the store has maximum standard deviation
      std_sales=pd.DataFrame(wal.groupby('Store')['Weekly_Sales'].std().sort_values())
      std_sales
```

```
[12]:         Weekly_Sales
      Store
      37      21837.461190
      30      22809.665590
      33      24132.927322
      44      24762.832015
      5       37737.965745
      43      40598.413260
      38      42768.169450
      3       46319.631557
      42      50262.925530
      36      60725.173579
      9       69028.666585
      16      85769.680133
      29      99120.136596
      34     104630.164676
      8      106280.829881
      26     110431.288141
      17     112162.936087
      7      112585.469220
      25     112976.788600
      40     119002.112858
```

```
15      120538.652043
31      125855.942933
21      128752.812853
45      130168.526635
32      138017.252087
12      139166.871880
1       155980.767761
22      161251.350631
11      165833.887863
24      167745.677567
18      176641.510839
28      181758.967539
41      187907.162766
19      191722.638730
35      211243.457791
6       212525.855862
39      217466.454833
2       237683.694682
27      239930.135688
23      249788.038068
13      265506.995776
4       266201.442297
20      275900.562742
10      302262.062504
14      317569.949476
```

[13]:

```
print("Store", std_sales.head(1).index[0], "has minimum standard deviation␣
 ↪value of: ", float(std_sales.head(1)['Weekly_Sales']))
```

```
Store 37 has minimum standard deviation value of:  21837.46119004889
```

[14]:

```
print("Store", std_sales.tail(1).index[0], "has maximum standard deviation␣
 ↪value of: ", float(std_sales.tail(1)['Weekly_Sales']))
```

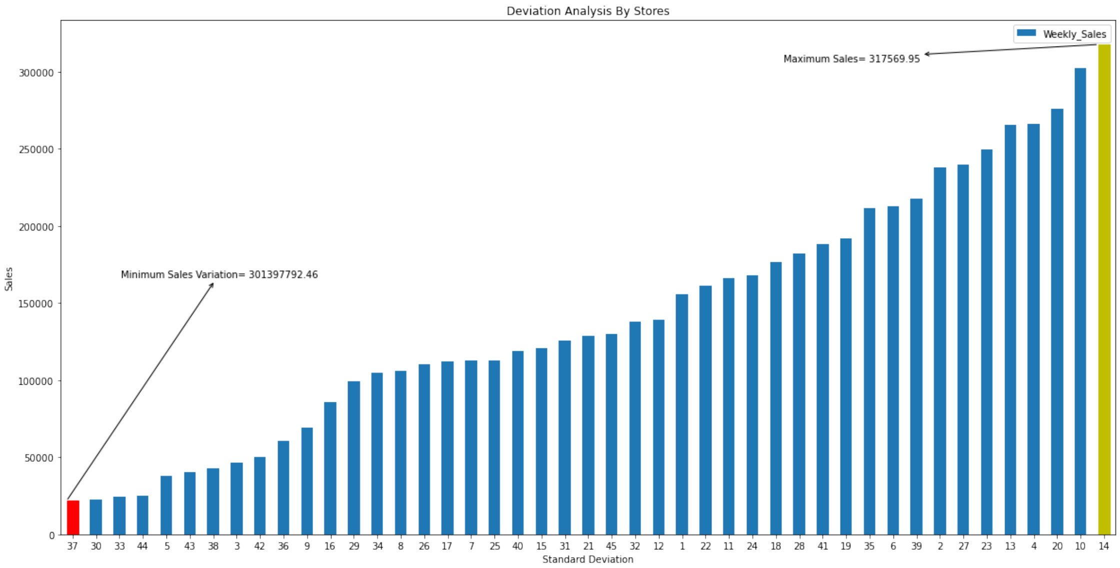
```
Store 14 has maximum standard deviation value of:  317569.9494755081
```

[15]:

```
#plot variation in std_sales
ax=std_sales.plot(kind='bar',
                 figsize=(20,10))

plt.xticks(rotation=0)
plt.ticklabel_format(useOffset=False, style='plain', axis='y')
plt.xlabel('Standard Deviation')
plt.ylabel('Sales')
plt.title('Deviation Analysis By Stores')
```

```
#labeling store with minimum variation in sales
minimum=ax.patches[0]
minimum.set_color('r')
ax.annotate("Minimum Sales Variation= {0:.2f}".format((maximum.get_height())),
           xy=(minimum.get_x(), minimum.get_height()),
           xytext=(0.15,0.5), textcoords='axes fraction',
           arrowprops=dict(arrowstyle="<-", connectionstyle="arc3"),
           horizontalalignment='center')


#labeling store with maximum variation in sales
maximum=ax.patches[len(ax.patches)-1]
maximum.set_color('y')
ax.annotate("Maximum Sales= {0:.2f}".format((maximum.get_height())),
           xy=(maximum.get_x(), maximum.get_height()),
           xytext=(0.75,0.92), textcoords='axes fraction',
           arrowprops=dict(arrowstyle="<-", connectionstyle="arc3"),
           horizontalalignment='center')
```

```
[15]: Text(0.75, 0.92, 'Maximum Sales= 317569.95')
```



```
[16]: #finding the coefficient of mean to standard deviation
mean_sales=pd.DataFrame(wal.groupby('Store')['Weekly_Sales'].std()/wal.
 ↪groupby('Store')['Weekly_Sales'].mean()).rename(columns={'Weekly_Sales':
 ↪'Coef_Mean'}).sort_values('Coef_Mean')
mean_sales
```

```
[16]:        Coef_Mean
      Store
      37      0.042084
      30      0.052008
      43      0.064104
      44      0.081793
      31      0.090161
      42      0.090335
      33      0.092868
      1       0.100292
      34      0.108225
      26      0.110111
      38      0.110875
      3       0.115021
      8       0.116953
      32      0.118310
      5       0.118668
      11      0.122262
      2       0.123424
      40      0.123430
      24      0.123637
      17      0.125521
      9       0.126895
      4       0.127083
      20      0.130903
      13      0.132514
      19      0.132680
      27      0.135155
      6       0.135823
      28      0.137330
      12      0.137925
      41      0.148177
      39      0.149908
      22      0.156783
      14      0.157137
      10      0.159133
      25      0.159860
      36      0.162579
      18      0.162845
      16      0.165181
      45      0.165613
      21      0.170292
      23      0.179721
      29      0.183742
      15      0.193384
      7       0.197305
      35      0.229681
```

[17]:

```
print("Store", mean_sales.head(1).index[0], "has the minimum coefficient of␣
↪mean w.r.t standard devaition of: ", float(mean_sales.head(1)['Coef_Mean']))
```

```
Store 37 has the minimum coefficient of mean w.r.t standard devaition of:
0.04208411895180789
```

[18]:

```
print("Store", mean_sales.tail(1).index[0], "has the maximum coefficient of␣
↪mean w.r.t standard devaition of: ", float(mean_sales.tail(1)['Coef_Mean']))
```

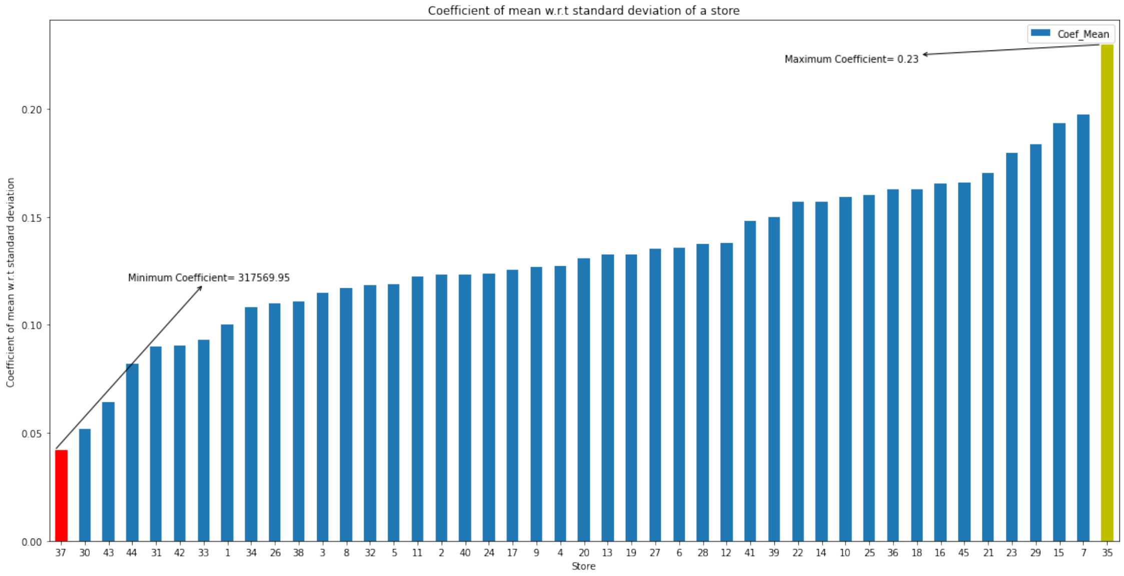
```
Store 35 has the maximum coefficient of mean w.r.t standard devaition of:
0.2296811138997643
```

[19]:

```
#plot variation in std_sales
ax=mean_sales.plot(kind='bar',
                 figsize=(20,10))

plt.xticks(rotation=0)
plt.ticklabel_format(useOffset=False, style='plain', axis='y')
plt.title('Coefficient of mean w.r.t standard deviation of a store')
plt.xlabel('Store')
plt.ylabel('Coefficient of mean w.r.t standard deviation')


#labeling store with minimum variation in sales
minimum=ax.patches[0]
minimum.set_color('r')
ax.annotate("Minimum Coefficient= {0:.2f}".format((maximum.get_height())),
           xy=(minimum.get_x(), minimum.get_height()),
           xytext=(0.15,0.5), textcoords='axes fraction',
           arrowprops=dict(arrowstyle="<-", connectionstyle="arc3"),
           horizontalalignment='center')


#labeling store with maximum variation in sales
maximum=ax.patches[len(ax.patches)-1]
maximum.set_color('y')
ax.annotate("Maximum Coefficient= {0:.2f}".format((maximum.get_height())),
           xy=(maximum.get_x(), maximum.get_height()),
           xytext=(0.75,0.92), textcoords='axes fraction',
           arrowprops=dict(arrowstyle="<-", connectionstyle="arc3"),
           horizontalalignment='center')
```

[19]:

```
Text(0.75, 0.92, 'Maximum Coefficient= 0.23')
```



[20]:
```
#breaking the dataset to a new dataset of data having date of 2012 year
fy12=wal[wal['Date'].dt.year==2012]
fy12.head()
```

[20]:
```
     Store       Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
100      1 2012-06-01    1550369.92             0        49.01       3.157
101      1 2012-01-13    1459601.17             0        48.53       3.261
102      1 2012-01-20    1394393.84             0        54.11       3.268
103      1 2012-01-27    1319325.59             0        54.26       3.290
104      1 2012-03-02    1636339.65             0        56.55       3.360

            CPI  Unemployment
100  219.714258         7.348
101  219.892526         7.348
102  219.985689         7.348
103  220.078852         7.348
104  220.172015         7.348
```

[21]:
```
#2nd and 3rd quarter sales of stores in year 2012

quartely_sales=pd.DataFrame()
quartely_sales['q2']=fy12[fy12.Date.dt.month.isin([4,5,6])].
 ↪groupby('Store')['Weekly_Sales'].sum()
quartely_sales['q3']=fy12[fy12.Date.dt.month.isin([7,8,9])].
 ↪groupby('Store')['Weekly_Sales'].sum()

#growth in 3rd qaurter
```

```
quartely_sales['growth']=(quartely_sales['q3']-quartely_sales['q2'])*100/
 ↪quartely_sales['q3']

#sorting the stores by growth
quartely_sales.sort_values('growth', ascending=False)
```

```
[21]:                q2           q3     growth
      Store
      16      6626133.44   6441311.11  -2.869328
      7       7613593.92   7322393.92  -3.976841
      35     10753570.97  10252122.68  -4.891166
      26     13218289.66  12417575.35  -6.448234
      39     20191585.63  18899955.17  -6.834040
      23     18283424.90  17103654.36  -6.897769
      41     17560035.88  16373588.44  -7.246105
      44      4322555.33   4020486.01  -7.513254
      32     15415236.21  14142164.84  -9.001955
      37      6859777.96   6250524.08  -9.747245
      8      11934275.61  10873860.34  -9.751967
      21      9226279.62   8403507.99  -9.790812
      19     18315278.56  16644341.31 -10.039071
      24     17768191.98  16125999.86 -10.183506
      13     26803225.55  24319994.35 -10.210657
      40     12849747.45  11647661.37 -10.320407
      33      3512138.05   3177072.43 -10.546364
      18     13834706.08  12507521.72 -10.611090
      4      28384185.16  25652119.35 -10.650449
      31     18249155.35  16454328.46 -10.907932
      11     17879095.77  16094363.07 -11.089179
      25      9247467.19   8309440.44 -11.288687
      42      7608247.31   6830839.86 -11.380847
      10     23598433.93  21169356.45 -11.474498
      30      5786335.45   5181974.44 -11.662756
      20     27550180.62  24665938.11 -11.693220
      43      8239792.67   7376726.03 -11.699860
      38      5732362.70   5129297.64 -11.757264
      27     22593640.73  20191238.11 -11.898243
      2      25085123.61  22396867.61 -12.002821
      3       5562668.16   4966495.93 -12.003880
      17     12918892.02  11533998.38 -12.007056
      12     13193365.04  11777508.50 -12.021698
      34     12858027.98  11476258.98 -12.040239
      22     13329065.39  11818544.33 -12.780940
      28     16985999.95  15055659.67 -12.821360
      1      21036965.58  18633209.98 -12.900384
      6      20728970.16  18341221.11 -13.018485
      9       7431320.13   6528239.56 -13.833447
```

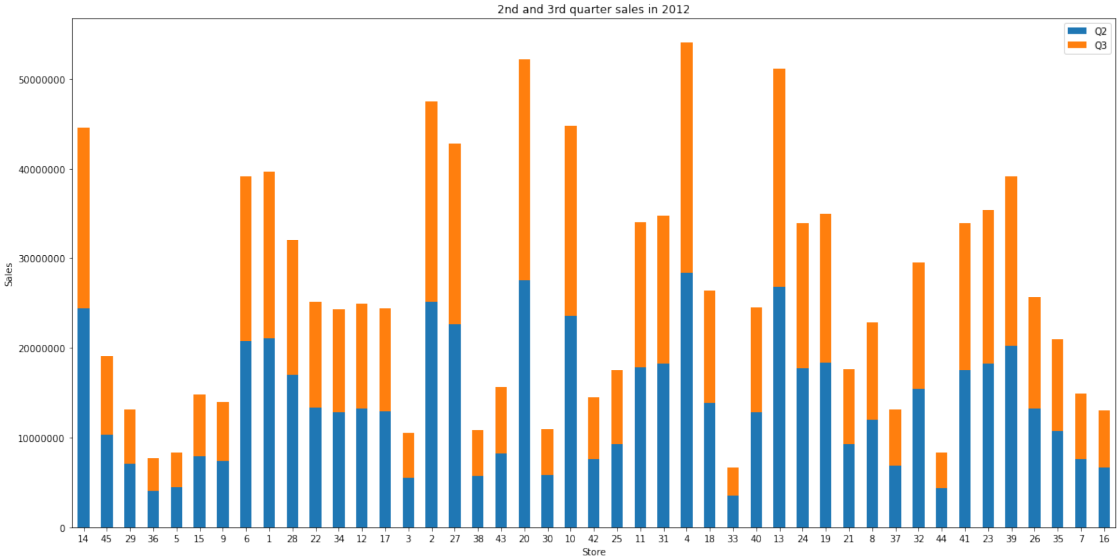
```
15      7867952.23   6909374.37 -13.873584
5       4427262.21   3880621.88 -14.086411
36      4090378.90   3578123.58 -14.316312
29      7034493.19   6127862.07 -14.795227
45     10278900.05   8851242.32 -16.129462
14     24427769.06  20140430.40 -21.287225
```

[22]:

```
#plotting the growth

ax=quartely_sales.sort_values('growth').plot(kind='bar',
                                             figsize=(20,10),
                                             stacked=True)


plt.xticks(rotation=0)
plt.ticklabel_format(useOffset=False, style='plain', axis='y')
plt.xlabel('Store')
plt.ylabel("Sales")
plt.title('2nd and 3rd quarter sales in 2012')
plt.legend(['Q2','Q3'])
```

[22]:

```
<matplotlib.legend.Legend at 0x181ff438550>
```



from the above table and visualization, we can clearly see that the sales are falling in 3rd quarter w.r.t 2nd quarter. Store 16 has good performance than others

[23]:

```
#sales during holiday events

from datetime import datetime
sales_total=wal.groupby('Date')['Weekly_Sales'].sum().reset_index()
```

```
sales_total
```

[23]:

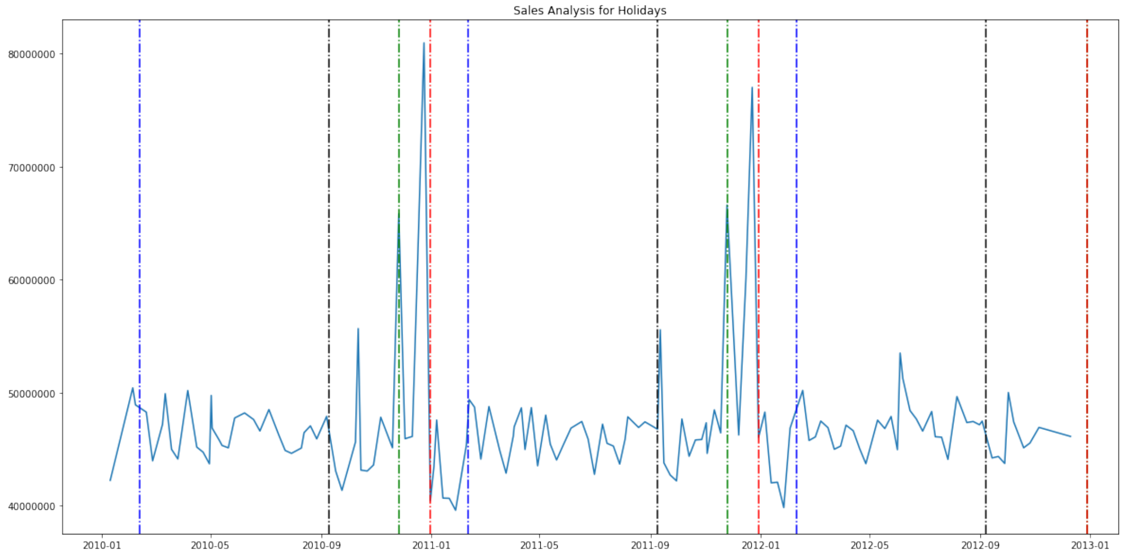
```
           Date  Weekly_Sales
0   2010-01-10   42239875.87
1   2010-02-04   50423831.26
2   2010-02-07   48917484.50
3   2010-02-19   48276993.78
4   2010-02-26   43968571.13
..         …            …
138 2012-10-08   47403451.04
139 2012-10-19   45122410.57
140 2012-10-26   45544116.29
141 2012-11-05   46925878.99
142 2012-12-10   46128514.25

[143 rows x 2 columns]
```

[24]:

```
super_bowl=['12-02-2010', '11-02-2011', '10-02-2012']
labour_day=['10-09-2010', '09-09-2011', '07-09-2012']
thanks_giving=['26-11-2010', '25-11-2011', '28-12-2012']
christmas=['31-12-2010', '30-12-2011', '28-12-2012']
```

[25]:

```
fig,ax=plt.subplots(figsize=(20,10))
ax.plot(sales_total['Date'], sales_total['Weekly_Sales'])

for day in super_bowl:
    day=datetime.strptime(day, '%d-%m-%Y')
    plt.axvline(x=day, linestyle='-.', c='b')

for day in labour_day:
    day=datetime.strptime(day, '%d-%m-%Y')
    plt.axvline(x=day, linestyle='-.', c='black')

for day in thanks_giving:
    day=datetime.strptime(day, '%d-%m-%Y')
    plt.axvline(x=day, linestyle='-.', c='g')

for day in christmas:
    day=datetime.strptime(day, '%d-%m-%Y')
    plt.axvline(x=day, linestyle='-.', c='r')

plt.title("Sales Analysis for Holidays")
plt.ticklabel_format(useOffset=False, style='plain', axis='y')
plt.show()
```



[26]:

```
#holiday sales
holiday_sales=pd.DataFrame()
holiday_sales['super_bowl']=wal[wal['Date'].isin(super_bowl)].
 ↪groupby(wal['Date'].dt.year)['Weekly_Sales'].sum()
holiday_sales['labour_day']=wal[wal['Date'].isin(labour_day)].
 ↪groupby(wal['Date'].dt.year)['Weekly_Sales'].sum()
holiday_sales['thanks_giving']=wal[wal['Date'].isin(thanks_giving)].
 ↪groupby(wal['Date'].dt.year)['Weekly_Sales'].sum()
holiday_sales['christmas']=wal[wal['Date'].isin(christmas)].groupby(wal['Date'].
 ↪dt.year)['Weekly_Sales'].sum()

holiday_sales.fillna(0, inplace=True)
holiday_sales
```

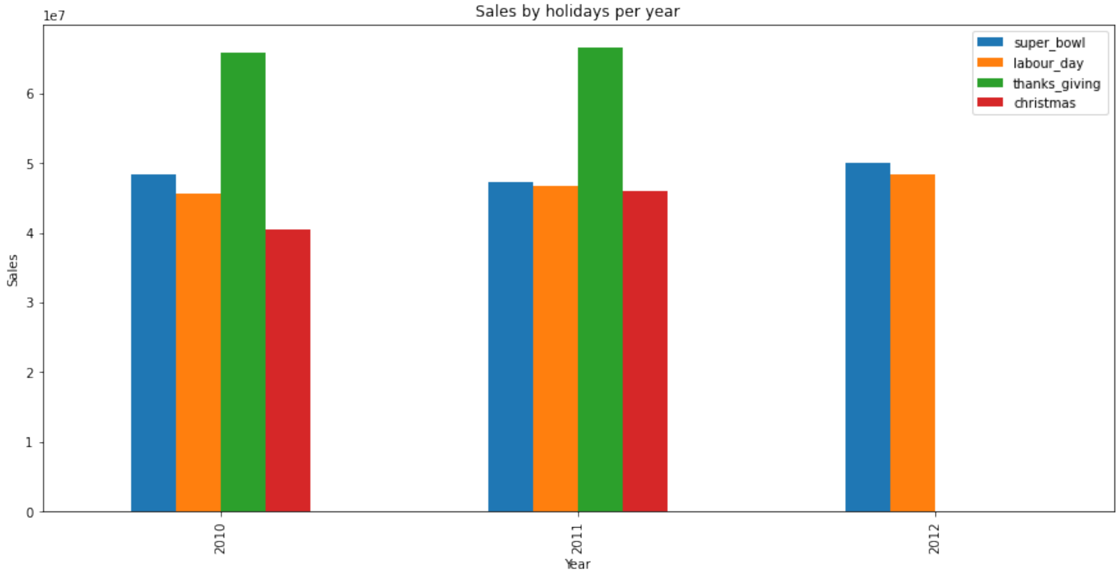
[26]:

```
        super_bowl   labour_day  thanks_giving    christmas
Date
2010   48336677.63  45634397.84    65821003.24  40432519.00
2011   47336192.79  46763227.53    66593605.26  46042461.04
2012   50009407.92  48330059.31           0.00         0.00
```

[27]:

```
#visualizing the sales in holidays
holiday_sales.plot(kind='bar',
                   figsize=(15,7),
                   xlabel='Year',
                   ylabel='Sales',
                   title='Sales by holidays per year')
```

```
[27]: <AxesSubplot:title={'center':'Sales by holidays per year'}, xlabel='Year',
      ylabel='Sales'>
```



From the above graphs, we can say sales incraese in thanks_giving holidays.

For all the holidays in a particular year, the sales are the much higher than the average non-holiday sales.

```
[28]: #copying original dataset to wal1 and creating a column having name quarter
      quarter=lambda x: 1 if x.month in [1,2,3] else 2 if x.month in [4,5,6] else 3␣
       ↪if x.month in [7,8,9] else 4
      wal1=wal
      wal1['Quarter']=wal1.Date.apply(quarter)
```

```
[29]: wal1
```

```
[29]:       Store       Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
      0         1 2010-05-02    1643690.90             0        42.31       2.572
      1         1 2010-12-02    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 2010-05-03    1554806.68             0        46.50       2.625
      …       …        …             …             …          …           …
      6430     45 2012-09-28     713173.95             0        64.88       3.997
      6431     45 2012-05-10     733455.07             0        64.89       3.985
      6432     45 2012-12-10     734464.36             0        54.47       4.000
      6433     45 2012-10-19     718125.53             0        56.47       3.969
      6434     45 2012-10-26     760281.43             0        58.85       3.882
```

```
             CPI  Unemployment  Quarter
0     211.096358         8.106        2
1     211.242170         8.106        4
2     211.289143         8.106        1
3     211.319643         8.106        1
4     211.350143         8.106        2
…            …           …       …
6430  192.013558         8.684        3
6431  192.170412         8.667        2
6432  192.327265         8.667        4
6433  192.330854         8.667        4
6434  192.308899         8.667        4

[6435 rows x 9 columns]
```

[30]:
```
#breaking wal1 to 3 dataset having year 2010, 2011 & 2012
fy_2010=wal1[wal1['Date'].dt.year==2010]
fy_2011=wal1[wal1['Date'].dt.year==2011]
fy_2012=wal1[wal1['Date'].dt.year==2012]
```

[31]:
```
fy_2010.head()
```

[31]:
```
   Store       Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
0      1 2010-05-02    1643690.90             0        42.31       2.572
1      1 2010-12-02    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 2010-05-03    1554806.68             0        46.50       2.625

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

[32]:
```
fy_2011.head()
```

[32]:
```
    Store       Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
48      1 2011-07-01    1444732.28             0        48.27       2.976
49      1 2011-01-14    1391013.96             0        35.40       2.983
50      1 2011-01-21    1327405.42             0        44.04       3.016
51      1 2011-01-28    1316899.31             0        43.83       3.010
52      1 2011-04-02    1606629.58             0        42.27       2.989

           CPI  Unemployment  Quarter
48  211.404742         7.742        3
```

```
49   211.457411         7.742        1
50   211.827234         7.742        1
51   212.197058         7.742        1
52   212.566881         7.742        2
```

[33]:

```
fy_2012.head()
```

[33]:

```
     Store        Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
100      1  2012-06-01    1550369.92             0        49.01       3.157
101      1  2012-01-13    1459601.17             0        48.53       3.261
102      1  2012-01-20    1394393.84             0        54.11       3.268
103      1  2012-01-27    1319325.59             0        54.26       3.290
104      1  2012-03-02    1636339.65             0        56.55       3.360

            CPI  Unemployment  Quarter
100  219.714258         7.348        2
101  219.892526         7.348        1
102  219.985689         7.348        1
103  220.078852         7.348        1
104  220.172015         7.348        1
```

[34]:

```
#monthly view of sales
monthly_sales=pd.DataFrame()
monthly_sales['2010']=fy_2010.groupby(fy_2010.Date.dt.month)['Weekly_Sales'].
 ↪sum()
monthly_sales['2011']=fy_2011.groupby(fy_2011.Date.dt.month)['Weekly_Sales'].
 ↪sum()
monthly_sales['2012']=fy_2012.groupby(fy_2012.Date.dt.month)['Weekly_Sales'].
 ↪sum()

monthly_sales
```

[34]:

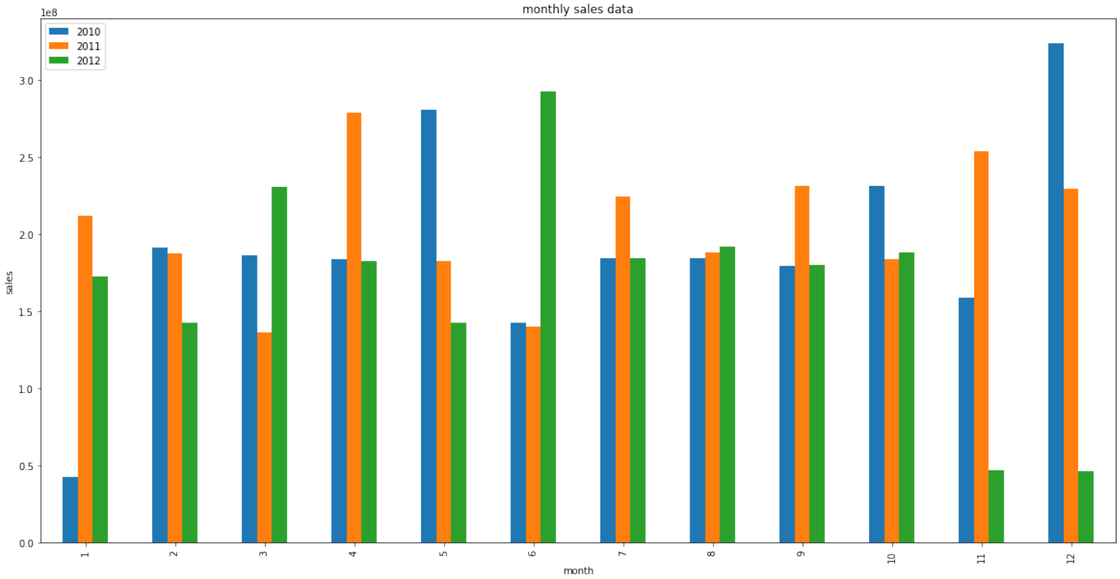
```
              2010          2011          2012
Date
1     4.223988e+07  2.119657e+08  1.722207e+08
2     1.915869e+08  1.876092e+08  1.428296e+08
3     1.862262e+08  1.365205e+08  2.307397e+08
4     1.838118e+08  2.789693e+08  1.825428e+08
5     2.806119e+08  1.828017e+08  1.422830e+08
6     1.424361e+08  1.401936e+08  2.923883e+08
7     1.842664e+08  2.244611e+08  1.845865e+08
8     1.845381e+08  1.880810e+08  1.916126e+08
9     1.797041e+08  2.310323e+08  1.797959e+08
10    2.311201e+08  1.837193e+08  1.880794e+08
11    1.587731e+08  2.534703e+08  4.692588e+07
12    3.235716e+08  2.293760e+08  4.612851e+07
```

[35]:

```
#visualizing the above
monthly_sales.plot(kind='bar',
                  figsize=(20,10),
                  xlabel='month',
                  ylabel='sales',
                  title='monthly sales data')
```

[35]:

```
<AxesSubplot:title={'center':'monthly sales data'}, xlabel='month',
ylabel='sales'>
```



[36]:

```
#yearly view of sales
yearly_sales=pd.DataFrame(wal.groupby(wal['Date'].dt.year)['Weekly_Sales'].
 ↪sum()).reset_index().rename(columns={'Date':'Year'})
```

[37]:

```
yearly_sales
```

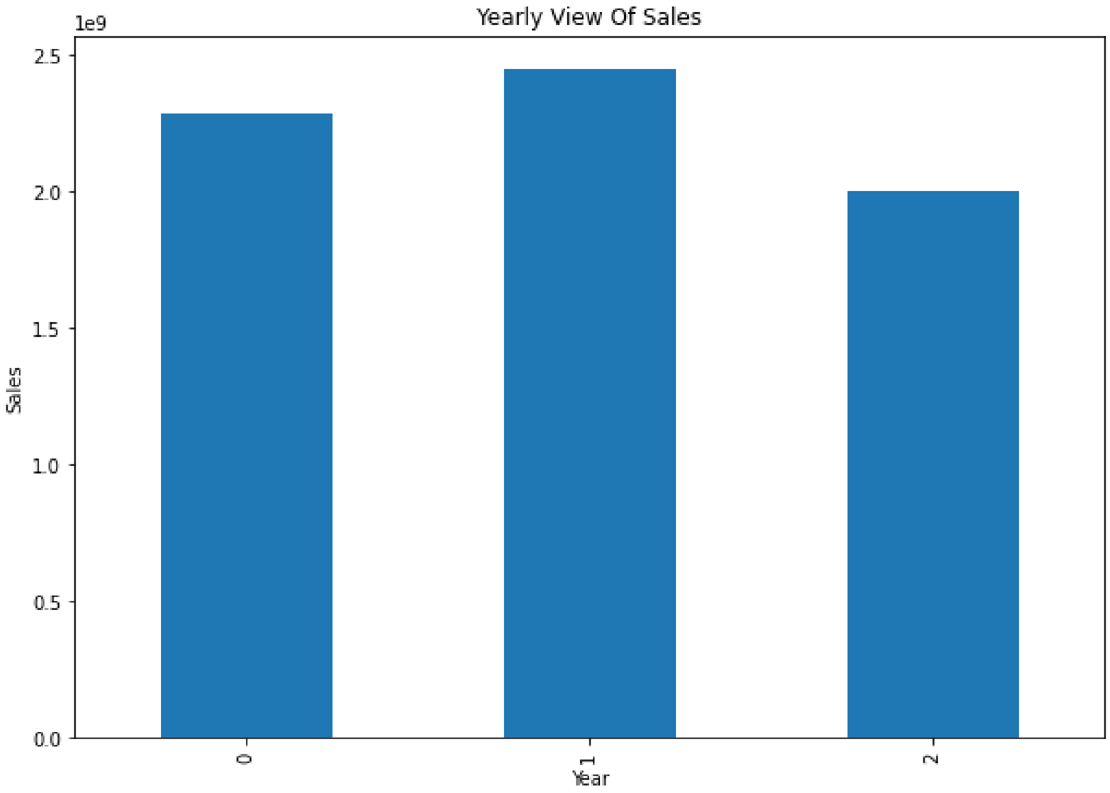
[37]:

```
   Year  Weekly_Sales
0  2010  2.288886e+09
1  2011  2.448200e+09
2  2012  2.000133e+09
```

[38]:

```
#visualizing the above
yearly_sales['Weekly_Sales'].plot(kind='bar',
                 figsize=(10,7),
                 xlabel='Year',
                 ylabel='Sales',
                 title='Yearly View Of Sales')
```

```
[38]: <AxesSubplot:title={'center':'Yearly View Of Sales'}, xlabel='Year',
      ylabel='Sales'>
```



```
[39]: #comparing sales during holidays and non-holidays
      hol_sales=pd.DataFrame()
      hol_sales['holiday_sales']=wal[wal['Holiday_Flag']==1].
       ↪groupby('Store')['Weekly_Sales'].sum()
      hol_sales['non_holiday_sales']=wal[wal['Holiday_Flag']==0].
       ↪groupby('Store')['Weekly_Sales'].sum()

      hol_sales
```

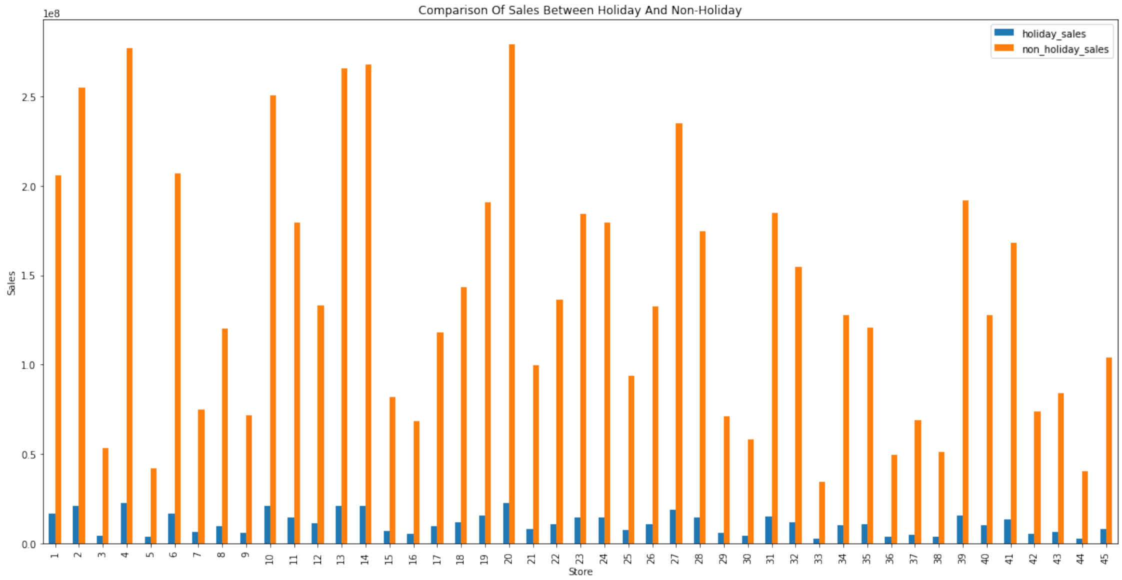
```
[39]:        holiday_sales  non_holiday_sales
      Store
      1        16657476.56       2.057453e+08
      2        20792669.00       2.545898e+08
      3         4378110.50       5.320862e+07
      4        22431026.24       2.771129e+08
      5         3595016.07       4.188067e+07
      6        16809079.27       2.069471e+08
      7         6724002.65       7.487427e+07
```

```
8          9753308.60       1.201979e+08
9          5889508.21       7.189971e+07
10        21137559.49       2.504802e+08
11        14483944.85       1.794788e+08
12        11381404.20       1.329058e+08
13        21130438.06       2.653873e+08
14        21205829.98       2.677941e+08
15         7064060.18       8.206962e+07
16         5667336.46       6.858509e+07
17         9797969.71       1.179842e+08
18        11694221.61       1.434205e+08
19        15770467.34       1.908644e+08
20        22490350.81       2.789074e+08
21         8264913.09       9.985297e+07
22        10848746.56       1.362269e+08
23        14625422.94       1.841252e+08
24        14750982.51       1.792650e+08
25         7396768.42       9.366441e+07
26        10720468.49       1.326959e+08
27        18922992.78       2.349329e+08
28        14782446.05       1.744812e+08
29         6069578.89       7.107198e+07
30         4368593.07       5.834829e+07
31        15000260.30       1.846136e+08
32        12037840.83       1.547814e+08
33         2625945.19       3.453428e+07
34        10419780.89       1.278300e+08
35        10743484.57       1.207772e+08
36         3676406.30       4.973581e+07
37         5075250.50       6.912749e+07
38         3815098.78       5.134453e+07
39        15511274.80       1.919343e+08
40        10080340.75       1.277900e+08
41        13349478.56       1.679925e+08
42         5676941.58       7.388881e+07
43         6359462.78       8.420597e+07
44         2960356.01       4.033273e+07
45         8362937.13       1.040324e+08
```

[40]:

```
#visualizing the above comparison
hol_sales.plot(kind='bar',
              figsize=(20,10),
              xlabel='Store',
              ylabel='Sales',
              title='Comparison Of Sales Between Holiday And Non-Holiday')
```

[40]:
```
<AxesSubplot:title={'center':'Comparison Of Sales Between Holiday And Non-
Holiday'}, xlabel='Store', ylabel='Sales'>
```



From the above chart we can clearly see that the store 20 has highest non-holiday sales

[41]:
```
#For Store 1 - Building  prediction models to forecast demand

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import mean_squared_error
```

[42]:
```
#getting store-1 data
store1=wal[wal['Store']==1]
store1
```

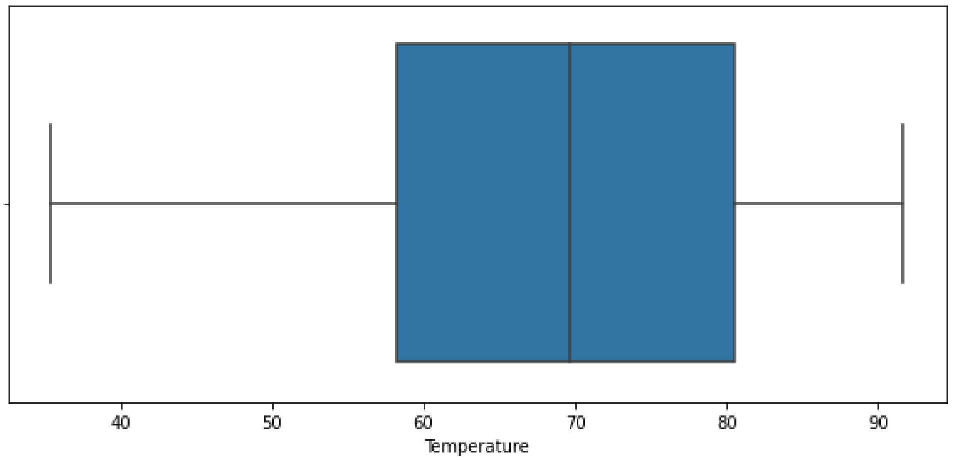
[42]:
```
     Store        Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
0        1  2010-05-02    1643690.90             0        42.31       2.572
1        1  2010-12-02    1641957.44             1        38.51       2.548
2        1  2010-02-19    1611968.17             0        39.93       2.514
3        1  2010-02-26    1409727.59             0        46.63       2.561
4        1  2010-05-03    1554806.68             0        46.50       2.625
..     …           …           …           …           …           …
138      1  2012-09-28    1437059.26             0        76.08       3.666
139      1  2012-05-10    1670785.97             0        68.55       3.617
140      1  2012-12-10    1573072.81             0        62.99       3.601
141      1  2012-10-19    1508068.77             0        67.97       3.594
142      1  2012-10-26    1493659.74             0        69.16       3.506
```

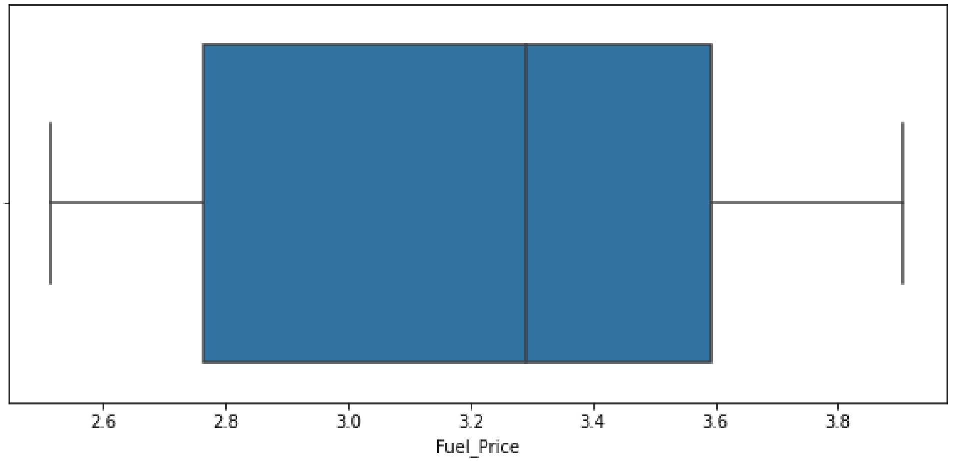
```
            CPI  Unemployment  Quarter
0    211.096358         8.106        2
1    211.242170         8.106        4
2    211.289143         8.106        1
3    211.319643         8.106        1
4    211.350143         8.106        2
..          …             …      …
138  222.981658         6.908        3
139  223.181477         6.573        2
140  223.381296         6.573        4
141  223.425723         6.573        4
142  223.444251         6.573        4

[143 rows x 9 columns]
```

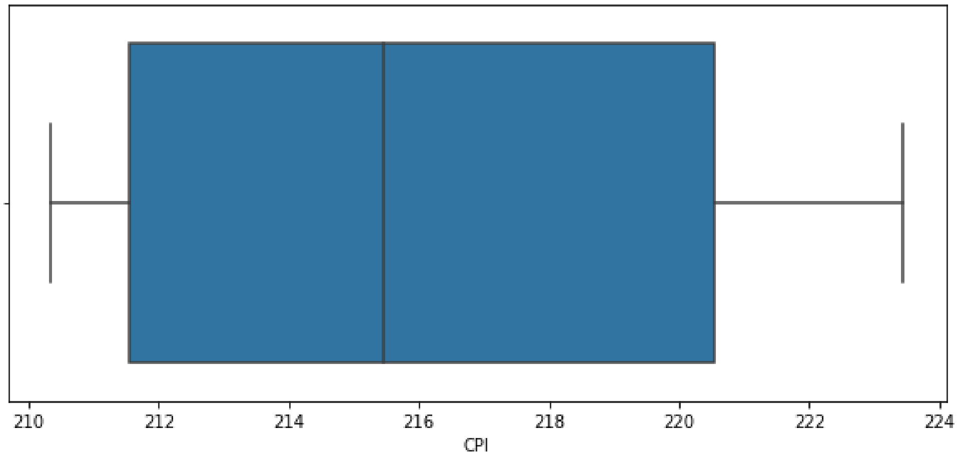
[43]:

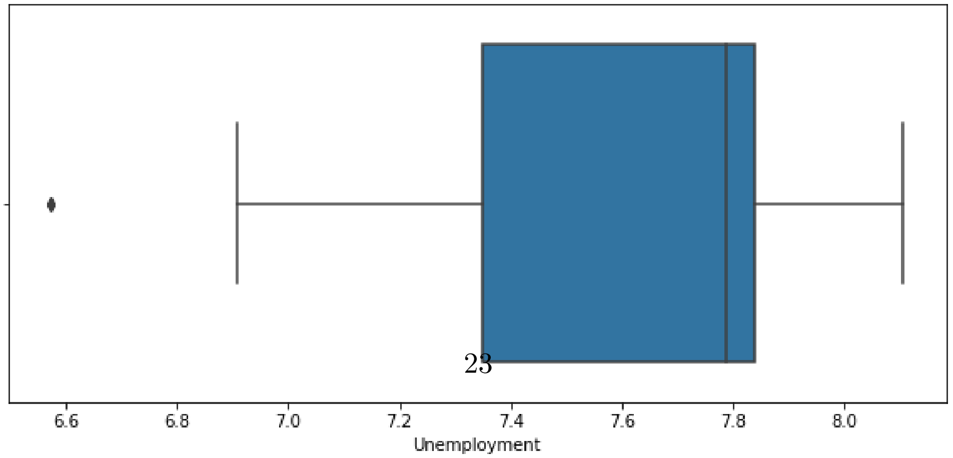
```python
#finding outliers
fig,axs=plt.subplots(4,figsize=(10,20))
a=store1[['Temperature','Fuel_Price','CPI','Unemployment']]
for i,column in enumerate(a):
    sns.boxplot(store1[column],ax=axs[i])
```

```
C:\Users\sssun\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\sssun\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\sssun\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\sssun\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```


40
50
60
70
80
90
Temperature


2.6
2.8
3.0
3.2
3.4
3.6
3.8
Fuel_Price


210
212
214
216
218
220
222
224
CPI


6.6
6.8
7.0
7.2
7.4
7.6
7.8
8.0
Unemployment

[44]:

```
#dropping outliers
store1_new=store1[(store1['Unemployment']<10) & (store1['Unemployment']>6.8)]
```

[45]:

```
store1_new
```

[45]:

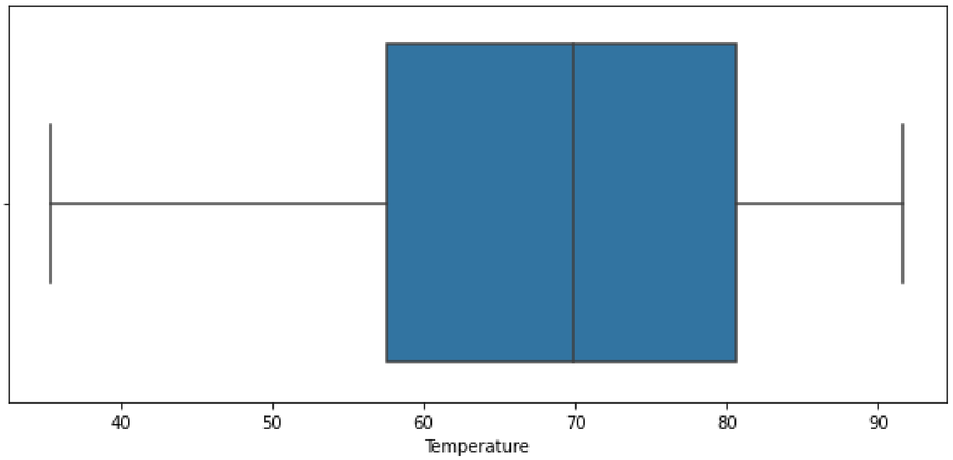
```
     Store       Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
0        1 2010-05-02    1643690.90             0        42.31       2.572
1        1 2010-12-02    1641957.44             1        38.51       2.548
2        1 2010-02-19    1611968.17             0        39.93       2.514
3        1 2010-02-26    1409727.59             0        46.63       2.561
4        1 2010-05-03    1554806.68             0        46.50       2.625
..     …        …           …           …          …         …
134      1 2012-08-31    1582083.40             0        80.49       3.638
135      1 2012-07-09    1661767.33             1        83.96       3.730
136      1 2012-09-14    1517428.87             0        74.97       3.717
137      1 2012-09-21    1506126.06             0        69.87       3.721
138      1 2012-09-28    1437059.26             0        76.08       3.666

            CPI  Unemployment  Quarter
0    211.096358         8.106        2
1    211.242170         8.106        4
2    211.289143         8.106        1
3    211.319643         8.106        1
4    211.350143         8.106        2
..          …           …      …
134  222.305480         6.908        3
135  222.439015         6.908        3
136  222.582019         6.908        3
137  222.781839         6.908        3
138  222.981658         6.908        3

[139 rows x 9 columns]
```

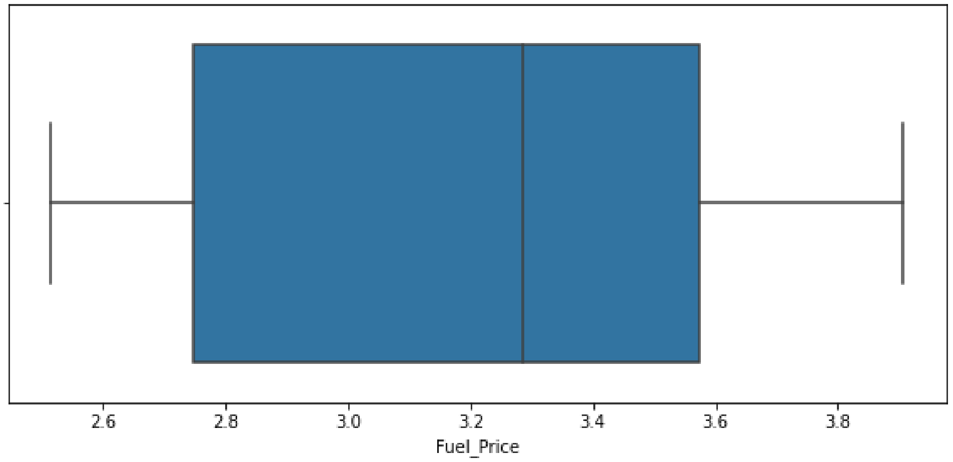
[46]:

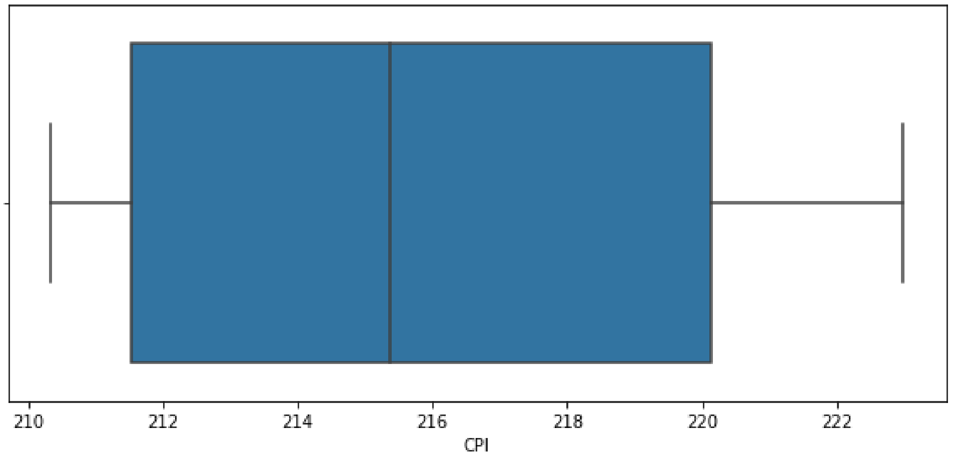
```
fig,axs=plt.subplots(4,figsize=(10,20))
a=store1_new[['Temperature','Fuel_Price','CPI','Unemployment']]
for i,column in enumerate(a):
    sns.boxplot(store1_new[column],ax=axs[i])
```

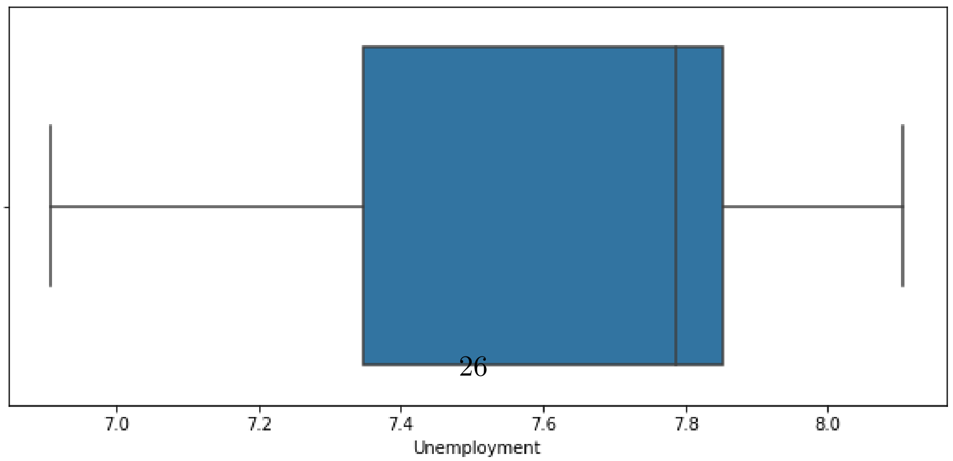
```
C:\Users\sssun\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\sssun\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
```

```
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\sssun\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\sssun\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```


40
50
60
70
80
90
Temperature


2.6
2.8
3.0
3.2
3.4
3.6
3.8
Fuel_Price


210
212
214
216
218
220
222
CPI


7.0
7.2
7.4
7.6
7.8
8.0
Unemployment

[47]:

```
#dropping the column store and quarter as they are not required
store1_new.drop(columns=['Store','Quarter'], inplace=True, axis=1)
```

```
C:\Users\sssun\anaconda3\lib\site-packages\pandas\core\frame.py:4906:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

[48]:

```
store1_new
```

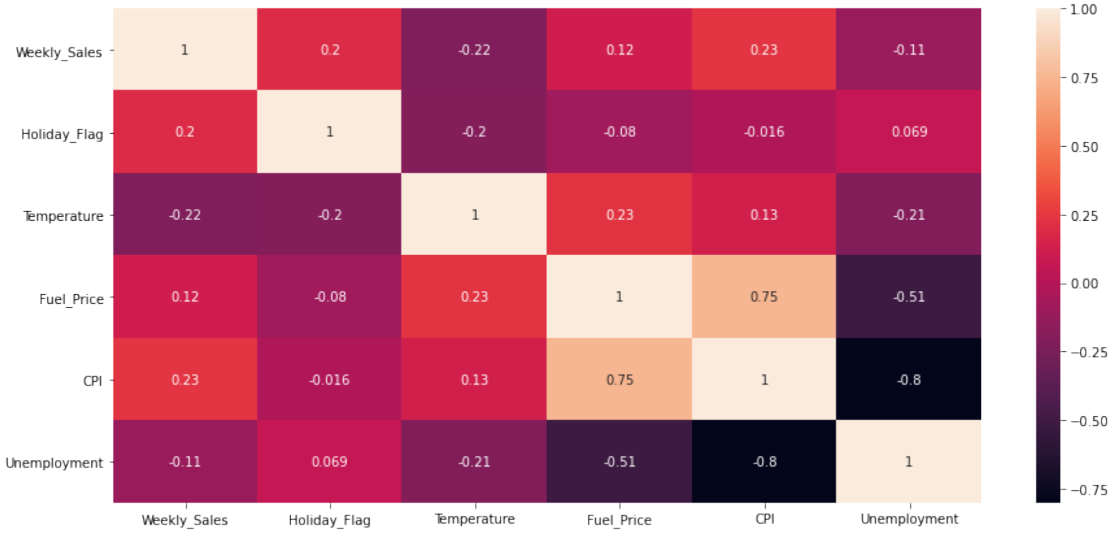
[48]:

```
          Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
0   2010-05-02    1643690.90             0        42.31       2.572
1   2010-12-02    1641957.44             1        38.51       2.548
2   2010-02-19    1611968.17             0        39.93       2.514
3   2010-02-26    1409727.59             0        46.63       2.561
4   2010-05-03    1554806.68             0        46.50       2.625
..         …           …           …          …         …
134 2012-08-31    1582083.40             0        80.49       3.638
135 2012-07-09    1661767.33             1        83.96       3.730
136 2012-09-14    1517428.87             0        74.97       3.717
137 2012-09-21    1506126.06             0        69.87       3.721
138 2012-09-28    1437059.26             0        76.08       3.666

            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
..          …           …
134  222.305480         6.908
135  222.439015         6.908
136  222.582019         6.908
137  222.781839         6.908
138  222.981658         6.908

[139 rows x 7 columns]
```

[49]:

```
#plotting the correlation
corr=store1_new.corr()
plt.figure(figsize=(15,7))
sns.heatmap(corr, annot=True)
```

[49]: <AxesSubplot:>



[50]:
```
#select feature and target
x=store1_new[['Fuel_Price','CPI','Unemployment']]
y=store1_new['Weekly_Sales']
```

[51]:
```
#splitting the dataset to train and set in 70:30
x_train, x_test, y_train, y_test= train_test_split(x,y,test_size=0.
 ↪2,random_state=7)
```

[52]:
```
#applying linear regression model

print("Linear Regression\n\n")
reg=LinearRegression()
reg.fit(x_train,y_train)
y_pred=reg.predict(x_test)


print('Accuracy:',reg.score(x_train, y_train)*100)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,␣
 ↪y_pred)))
```

```
Linear Regression


Accuracy: 8.039548342683144
Mean Absolute Error: 124239.64040070007
Mean Squared Error: 22799142240.032238
Root Mean Squared Error: 150993.84835162072
```

[53]:

```
# Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
print('Random Forest Regressor:\n\n')
rfr = RandomForestRegressor(n_estimators = 400,max_depth=15,n_jobs=5)
rfr.fit(x_train,y_train)
y_pred=rfr.predict(x_test)
print('Accuracy:',rfr.score(x_test, y_test)*100)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,␣
 ↪y_pred)))
```

```
Random Forest Regressor:



Accuracy: 41.10073490551326
Mean Absolute Error: 99724.18998582901
Mean Squared Error: 15011272634.127914
Root Mean Squared Error: 122520.49883235015
```

[54]:

```
#Change dates into days by creating new variable
wal['Day']=pd.to_datetime(wal['Date']).dt.day_name()
wal
```

[54]:

```
      Store        Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
0         1  2010-05-02    1643690.90             0        42.31       2.572
1         1  2010-12-02    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  2010-05-03    1554806.68             0        46.50       2.625
…       …         …           …             …           …           …
6430     45  2012-09-28     713173.95             0        64.88       3.997
6431     45  2012-05-10     733455.07             0        64.89       3.985
6432     45  2012-12-10     734464.36             0        54.47       4.000
6433     45  2012-10-19     718125.53             0        56.47       3.969
6434     45  2012-10-26     760281.43             0        58.85       3.882

             CPI  Unemployment  Quarter       Day
0     211.096358         8.106        2    Sunday
1     211.242170         8.106        4  Thursday
2     211.289143         8.106        1    Friday
3     211.319643         8.106        1    Friday
4     211.350143         8.106        2    Monday
…          …            …         …         …
```

```
6430  192.013558         8.684        3    Friday
6431  192.170412         8.667        2  Thursday
6432  192.327265         8.667        4    Monday
6433  192.330854         8.667        4    Friday
6434  192.308899         8.667        4    Friday

[6435 rows x 10 columns]
```