بسم الله الرحمن الرحيم

ALEXANDRIA UNIVERSITY
كلية الحاسبات وعلوم البيانات

Data Science Methodology

Dr.Tamar Emara

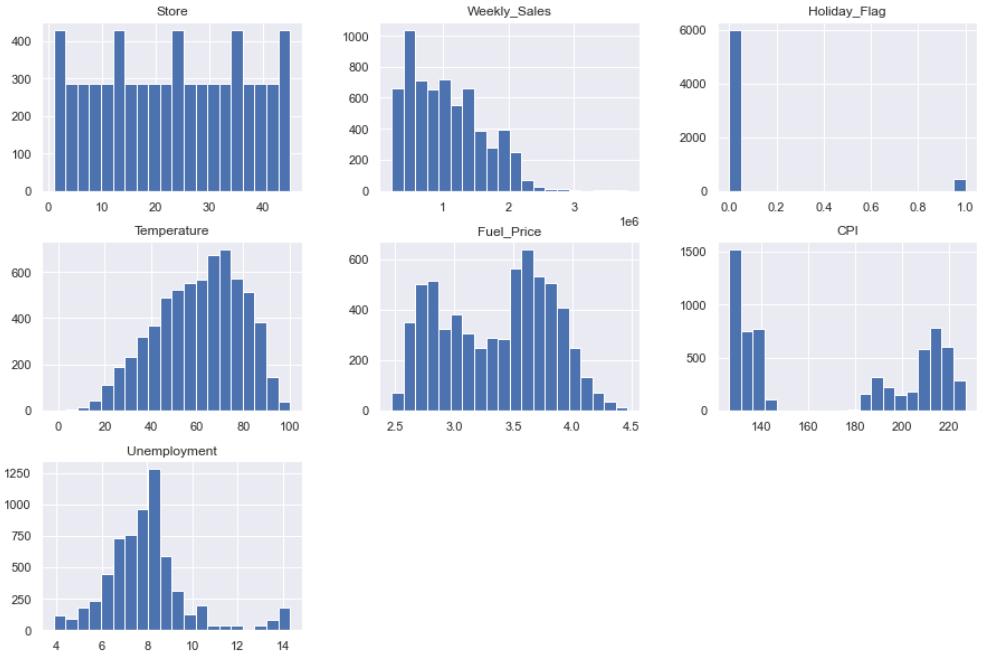
Team Members:	(ID)s:
Aabed Waleed Mohamed	20221374477
Sohile Adel Abdelghafoor	20221375835
Gehad Mohamed Mohsen	20221370605

What is the dataset?

This is the Walmart Data set that could be found on kaggle at https://www.kaggle.com/datasets/varsharam/walmart-sales-dataset-of45stores/download?data which features the Date from 2010-02-05 up to 2012-11-01 across 45 stores with the weekly sales of each week for each store, the Temperature, the Fuel price on each week, Consumer Price Indexing(CPI) and the weeks that had holidays within them.

First Phase: cleaning the data

Part 1 looking at the distributions



So we started by visualizing the data to show the distribution of each column. If we are building a Machine Learning model and we found two similar distributions we have to remove one of them, in that case we wouldn't lose that much value from our data.
Also we can detect outliers with these histograms.

Data Cleaning

```
df.describe() # No errors (i.e. negative Fuel_Price or Temperature of 1000)
```

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
count	6435.000000	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000
mean	23.000000	1.046965e+06	0.069930	60.663782	3.358607	171.578394	7.999151
std	12.988182	5.643666e+05	0.255049	18.444933	0.459020	39.356712	1.875885
min	1.000000	2.099862e+05	0.000000	-2.060000	2.472000	126.064000	3.879000
25%	12.000000	5.533501e+05	0.000000	47.460000	2.933000	131.735000	6.891000
50%	23.000000	9.607460e+05	0.000000	62.670000	3.445000	182.616521	7.874000
75%	34.000000	1.420159e+06	0.000000	74.940000	3.735000	212.743293	8.622000
max	45.000000	3.818686e+06	1.000000	100.140000	4.468000	227.232807	14.313000

we also check for duplicated rows and we find non

```
In [6]: df.duplicated().sum() # No duplicated rows
Out[6]: 0
```

We need to change the Dtype of Date feature to datetime64[ns].

```
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
```

Also, we noticed that all Non-Null Count values are equal, therefore no null values.

```
: df.info()
```

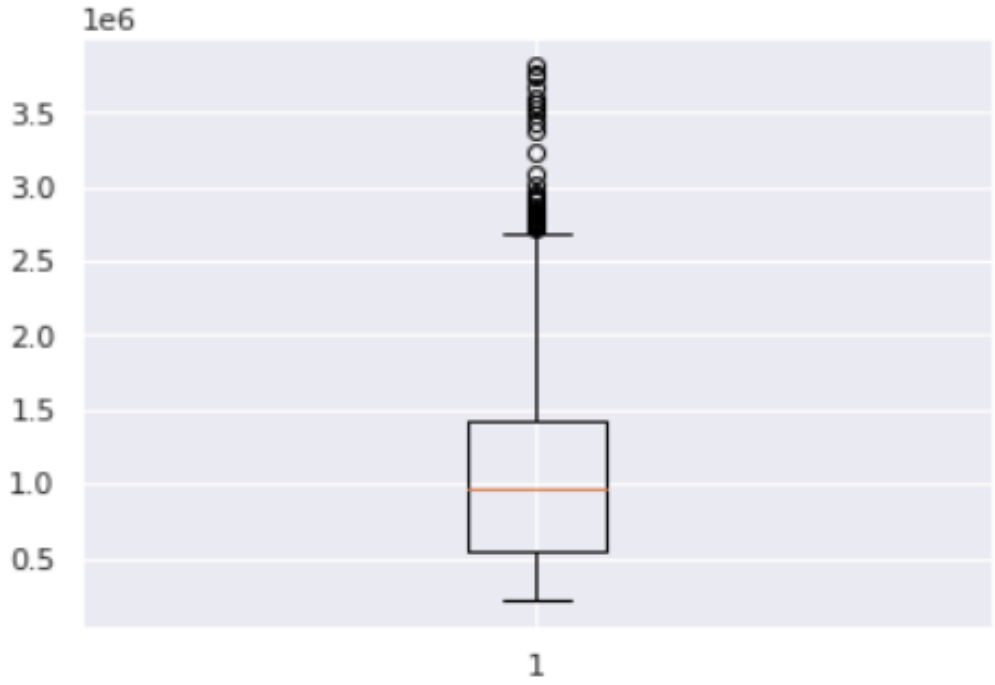
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Store         6435 non-null   int64  
 1   Date          6435 non-null   datetime64[ns]
 2   Weekly_Sales  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
```

```python
def drop_outliers(data, column, lower_limit, upper_limit):
    data = data[((data[column] < upper_limit) & (data[column] > lower_limit)) | data[column].isnull()]
    return data
```

```python
def handle_outliers(data, column):
    quartile1 = full_df[column].quantile(.25)
    quartile3 = full_df[column].quantile(.75)
    iqr = quartile3 - quartile1
    upper_limit = quartile3 + 1.5 * iqr
    lower_limit = quartile1 - 1.5 * iqr

    outliers = full_df[(full_df[column] >= upper_limit) | (full_df[column] <= lower_limit)]

    outliers_proportion = len(outliers)/len(full_df)

    print(f'{column} outlier limits: {lower_limit}, {upper_limit}')
    print(f'Outliers proportion: {outliers_proportion}')

    if outliers_proportion == 0:
        print(f'No outliers found in {column}.\n')
        plt.boxplot(data[column])
        plt.title(f'{column} boxplot');

    elif outliers_proportion <= 0.05:
        data_before = full_df
        data_after = drop_outliers(data, column, lower_limit, upper_limit)
        print(f'{data_before.shape[0] - data_after.shape[0]} rows dropped\n')

        fig = plt.figure(figsize =(6, 4))
        ax = fig.add_axes([0, 0, 1, 1])
        ax.boxplot([data_before[column], data_after[column]])
        plt.xticks([1, 2], [f'{column} before', f'{column} after'])
        plt.title(f'Dropping {column} outliers');

        data = data_after

    else:  # In case we have many outliers, we won't drop them.
        print(f"We have too many outliers in {column} feature. We won't drop them.\n")
        plt.boxplot(data[column])
        plt.title(f'{column} boxplot');

    return data
```

Even though the Weekly Sales feature has outliers, we won't drop them because it's completely normal to see the weekly sales insanely rise during holidays.

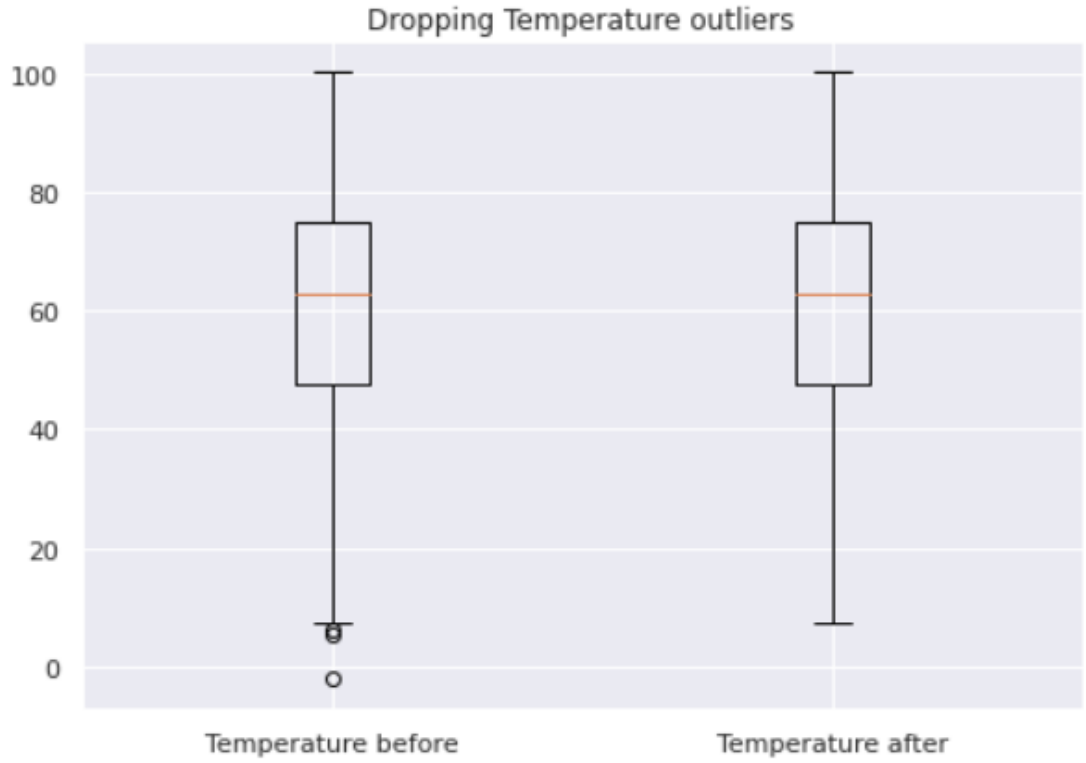
```
plt.boxplot(df.Weekly_Sales);
```



As we can see in the Temperature we found 3 outliers so we are going to drop them out of our data

```
df = handle_outliers(df, 'Temperature');
```

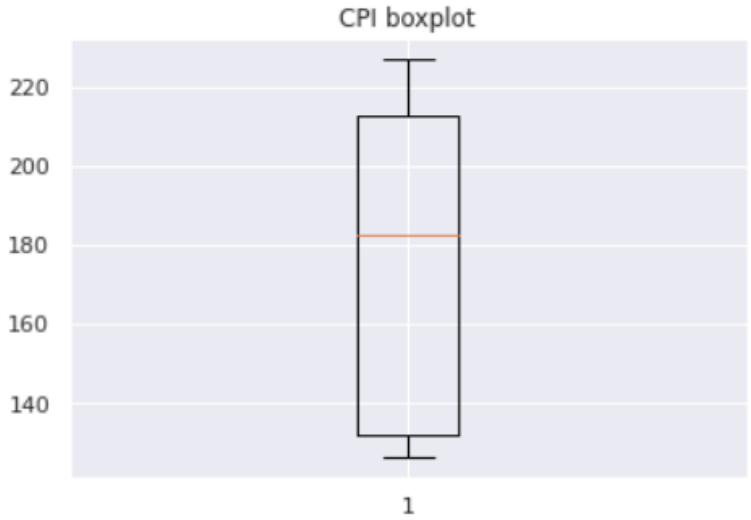
```
Temperature outlier limits: 6.240000000000002, 116.16
Outliers proportion: 0.0004662004662004662
3 rows dropped
```



We find no outliers in the Fuel Prices and the CPI data

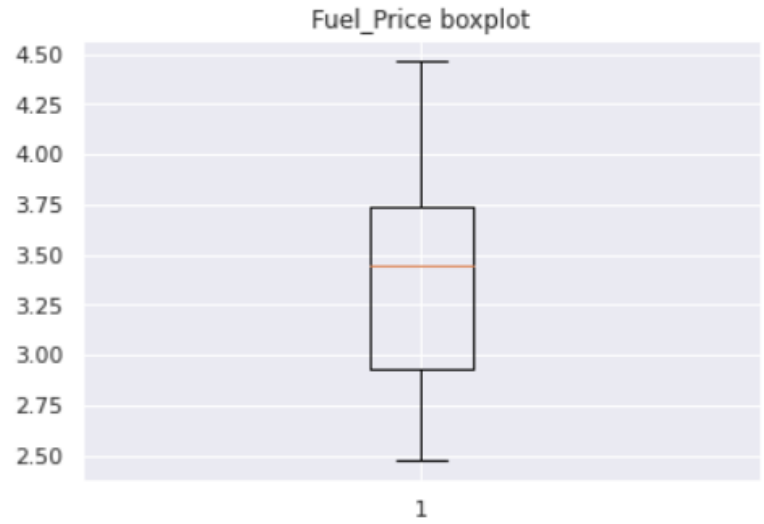
```
df = handle_outliers(df, 'CPI');
```

```
CPI outlier limits: 10.222559825000019, 334.255733625
Outliers proportion: 0.0
No outliers found in CPI.
```




```
df = handle_outliers(df, 'Fuel_Price');
```

```
Fuel_Price outlier limits: 1.7299999999999998, 4.938
Outliers proportion: 0.0
No outliers found in Fuel_Price.
```



```
df = df = handle_outliers(df, 'Unemployment');
```

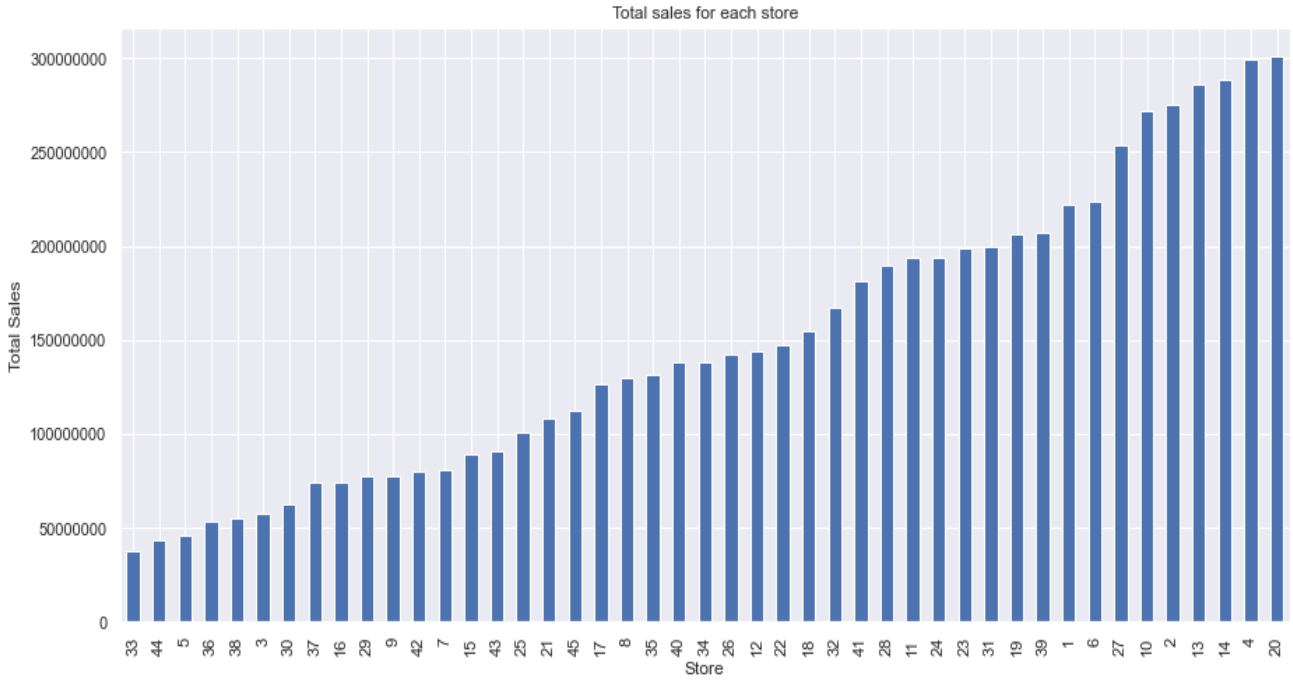
```
Unemployment outlier limits: 4.2945, 11.218499999999999
Outliers proportion: 0.07474747474747474
We have too many outliers in Unemployment feature. We won't drop them.
```

Second phase: asking questions

Which Store has the maximum sales?

As we can see that store no 20 has the highest sales with 301397792.4
(we used attributes to determine which store has the highest profit)

```
total_sales= df.groupby('Store')['Weekly_Sales'].sum().sort_values()
plt.figure(figsize=(15,8))
plt.ticklabel_format(style='plain', axis='y')
plt.title('Total sales for each store')
plt.xlabel('Store')
plt.ylabel('Total Sales')
total_sales.plot(kind='bar');
```




```
total_sales = total_sales.reset_index()
```

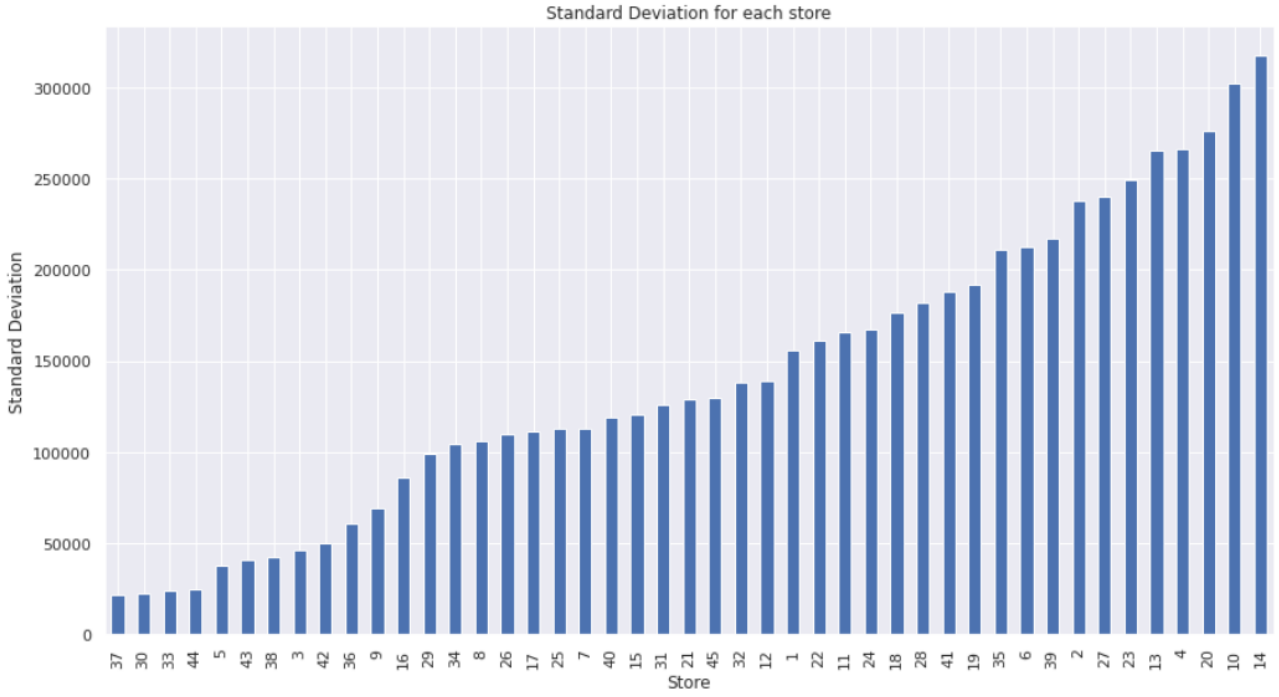
```
print(f"As we can see that store no {int(total_sales.iloc[-1][0])} has the highest sales with {total_sales.iloc[-1][1]}")
```

As we can see that store no 20 has the highest sales with 301397792.46

Which store has maximum standard deviation?

(we used attributes to determine which store has the most variation in total sales)

```
walmart_data_std = df.groupby('Store')['Weekly_Sales'].std().sort_values()
plt.figure(figsize=(15,8))
plt.ticklabel_format(style='plain', axis='y')
plt.title('Standard Deviation for each store')
plt.xlabel('Store')
plt.ylabel('Standard Deviation')
walmart_data_std.plot(kind='bar');
```




```
walmart_data_std = walmart_data_std.reset_index()
```

```
print(f"Thus, the store which has maximum standard deviation is store number {int(walmart_data_std.iloc[-1][0])} with {walm
```

Thus, the store which has maximum standard deviation is store number 14 with 317569.9494755081.

Which holiday has sales higher than the mean of the non holiday season for all the stores together?

We first identified the holidays that we have, so we went on the internet and searched the dates that were offered to have a holiday during the week.

And to answer our questions we compared the sales to the mean of the data and we found that all the holidays have higher sales than the mean of the non_holiday season for all the stores together.

```
month_numbers_list = [2, 9, 11, 12]
holidays_list = ['Super Bowl', 'Labor Day', 'Thanksgiving', 'Christmas']
holidays_sales_mean_dict = {}
for i in range(len(month_numbers_list)):
    sales_mean = df.query('Holiday_Flag == 1 & Date.dt.month == @month_numbers_list[@i]')['Weekly_Sales'].mean()
    holidays_sales_mean_dict[holidays_list[i]] = sales_mean

non_holiday_sales_mean = df.query('Holiday_Flag == 0')['Weekly_Sales'].mean()
print(f'Mean sales for each holiday:\n{holidays_sales_mean_dict}\n')
print(f'Mean sales for non holidays:\n{non_holiday_sales_mean}\n\n')

print('Therefore holidays that have higher sales than the mean sales in the non-holiday season:')
holidays_higher_dict = {}
for holiday in list(holidays_sales_mean_dict.keys()) :
    if holidays_sales_mean_dict[holiday] > non_holiday_sales_mean:
        print(f'  - {holiday}.')
        holidays_higher_dict[holiday] = holidays_sales_mean_dict[holiday]
```

Mean sales for each holiday:
{'Super Bowl': 1079127.9877037038, 'Labor Day': 1042427.293925926, 'Thanksgiving': 1471273.427777778, 'Christmas': 960833.1115555555}

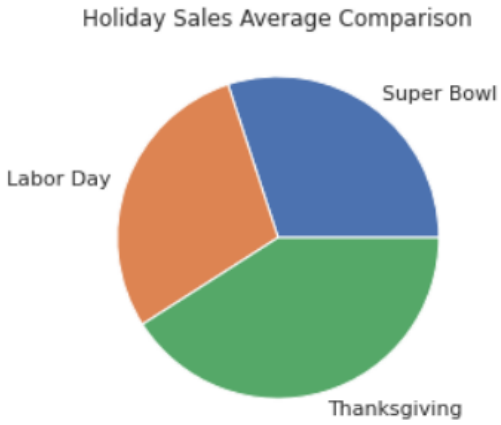
Mean sales for non holidays:
1041367.5779839519

Therefore holidays that have higher sales than the mean sales in the non-holiday season:

- Super Bowl.
- Labor Day.
- Thanksgiving.

(we used attributes to determine what holiday is more profitable)

```
plt.pie(list(holidays_higher_dict.values()) ,labels = list(holidays_higher_dict.keys()));
plt.title('Holiday Sales Average Comparison');
```



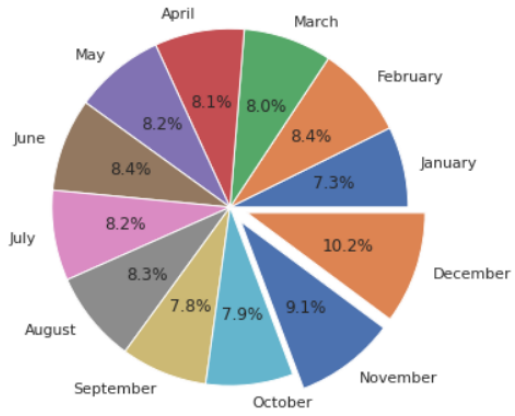
Thanksgiving have the highest mean.

And we found that April and July has the highest amount of sales while January and November has the lowest amount of sales.

(we used attributes to determine which months are most likely to have high sales.)

We find that December and November has the highest Average Sales while January and September has the lowest Average Sales.

```
plt.figure(figsize=(6,6))
Date_df = df[['Date','Weekly_Sales','Store']].sort_values('Date')
Date_df['Month'] = Date_df['Date'].dt.month
Month_mean = Date_df.groupby(['Month'])['Weekly_Sales'].mean()
Month_Names_arr = ['January','February','March','April','May','June','July','August','September', 'October', 'November','De
plt.pie(Month_mean, explode=[0,0,0,0,0,0,0,0,0,.1,.1],labels=Month_Names_arr,autopct='%1.1f%%');
```

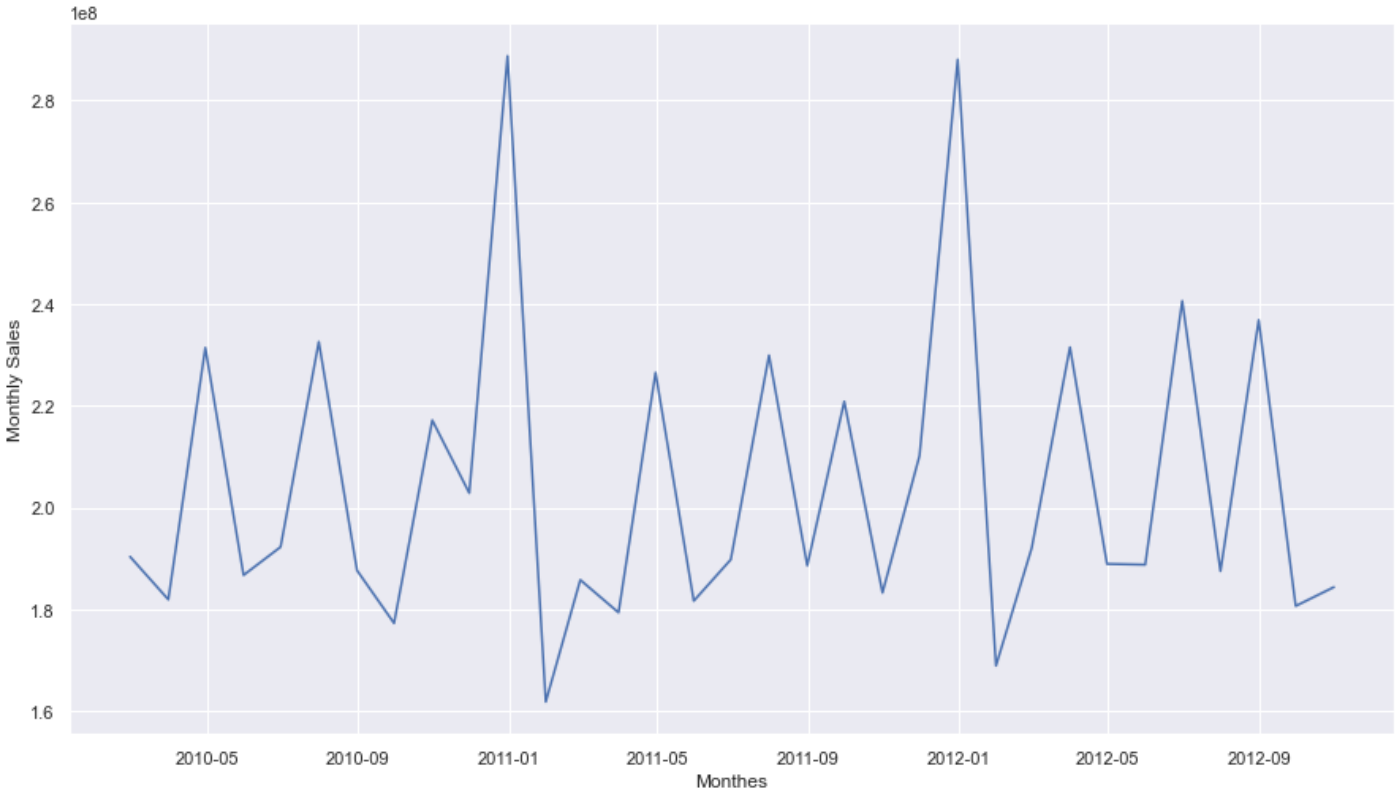


What are the monthly sales and semester sales?

Here is the monthly sales overview

(We used these attributes to figure out what months saw an increase or decrease in sales.)

```
Monthly_df = df[['Weekly_Sales','Date']].groupby(pd.Grouper(key='Date', freq='1M')).sum().reset_index()
plt.figure(figsize=(15,8))
plt.plot(Monthly_df['Date'], Monthly_df['Weekly_Sales'])
plt.title('Monthly veiw of Sales')
plt.xlabel('Monthes')
plt.ylabel('Monthly Sales');
```

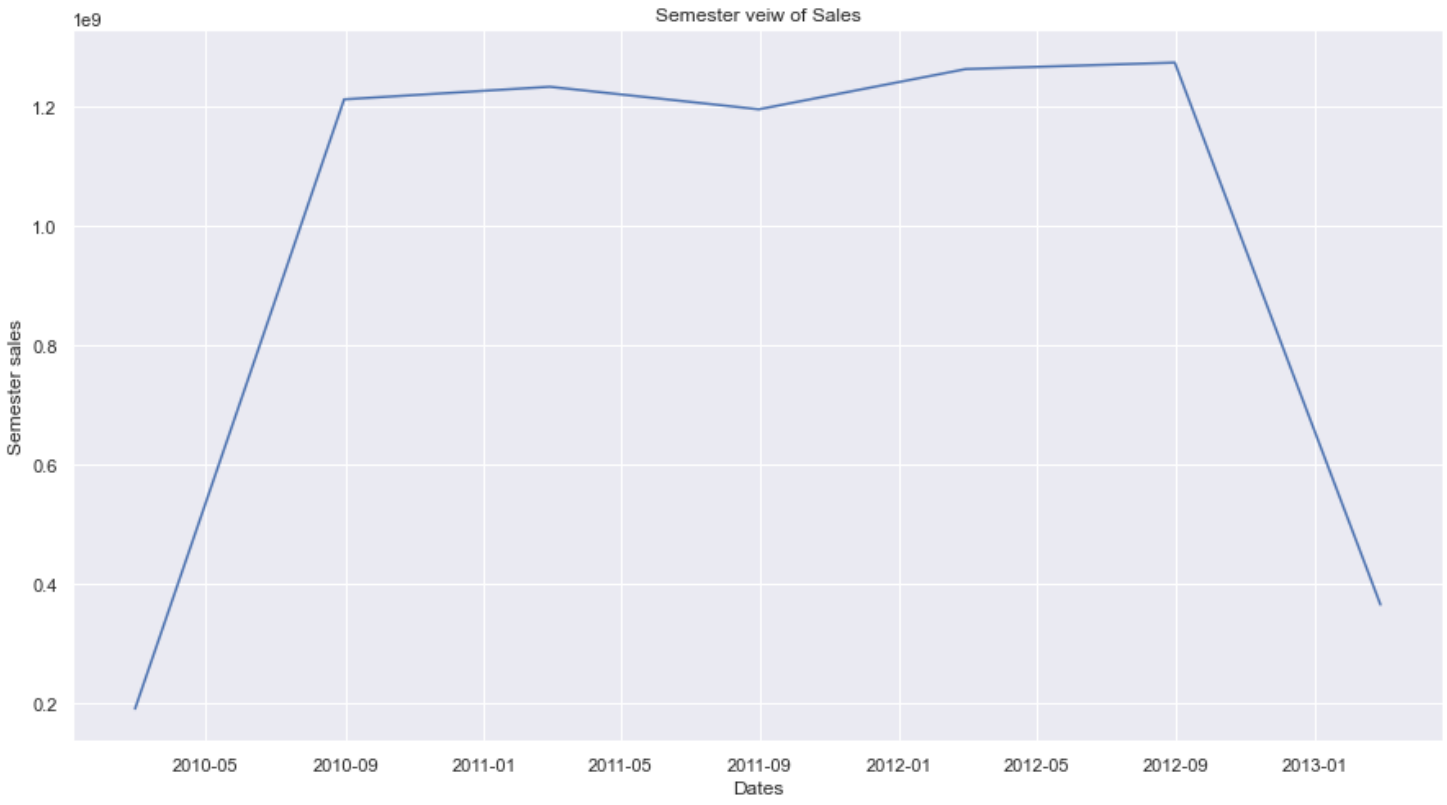


The Monthly Sales graph shows that the highest amount of sales is recorded between December 2010 and February 2011.

And here is the semester overview

(We used these attributes to figure out what semester saw an increase or decrease in sales.)

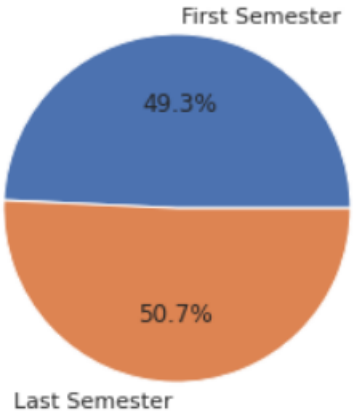
```
Semester = df[['Date','Weekly_Sales']].groupby(pd.Grouper(key='Date', freq='6M')).sum().reset_index()
plt.figure(figsize=(15,8))
plt.plot(Semester['Date'], Semester['Weekly_Sales'])
plt.title('Semester veiw of Sales')
plt.xlabel('Dates')
plt.ylabel('Semester sales');
```



According to the semester sales graph, the lowest amount of sales is recorded in 1st semester of 2010 and 2nd semester of 2012.

(We used these attributes to figure out what semester is more profitable.)

```
Semester = df[['Date','Weekly_Sales']]
Semester['Month'] = Semester['Date'].dt.month
First_Semester = Semester.query("Month <= 6")
Last_Semester = Semester.query("Month > 6")
Comparison_Array = []
Comparison_Array.append(First_Semester['Weekly_Sales'].sum())
Comparison_Array.append(Last_Semester['Weekly_Sales'].sum())
label_Array = ['First Semester','Last Semester']
plt.pie(Comparison_Array,labels = label_Array ,autopct='%1.1f%%');
```



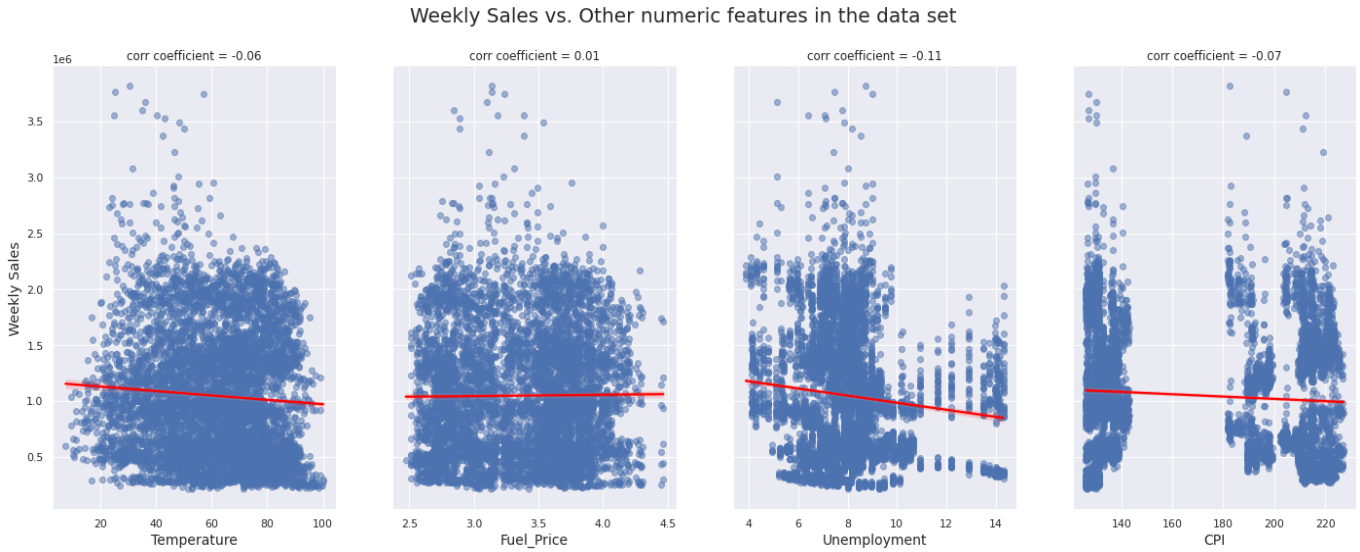
We see that Sales are a bit better in the second half of the year than the second

What is the relation between the weekly sales and the other numerical features?

So here is the plot between weekly Sales and All the other numerical features, let's tackle them down one by one

(We used these attributes to figure out what features have a correlation with the weekly sales.)

```
features_list = ['Temperature', 'Fuel_Price', 'Unemployment', 'CPI']

corr_list = list(df[['Weekly_Sales', 'Temperature', 'Fuel_Price', 'Unemployment', 'CPI']].corr().iloc[0, 1:])

fig, ax = plt.subplots(1,4, figsize=(24, 8), sharey=True)


for i in range(len(features_list)):
    sns.regplot(x=features_list[i], y='Weekly_Sales', data=df, ax=ax[i], line_kws={'color':'red'}, scatter_kws={'alpha':0.5
    ax[i].set_xlabel(features_list[i], size = 14)
    ax[i].set_ylabel(None)
    ax[i].set_title(f'corr coefficient = {round(corr_list[i], 2)}')

ax[0].set_ylabel('Weekly Sales', size = 14)
fig.suptitle("Weekly Sales vs. Other numeric features in the data set", size = 20);
```



- From the Temperature plot, there is a very weak negative correlation and almost no correlation between Weekly Sales and Temperature.

- From the Fuel Price plot, there is a very weak positive correlation and almost no correlation between Weekly Sales and Fuel Price.

- From the Unemployment plot, there is a very weak negative correlation between Weekly Sales and Unemployment.

- From the CPI, there is a very weak negative correlation and almost no correlation between Weekly Sales and CPI. Also we notice that there are 2 cluster, one with a CPI around 150 or lower and the other with a CPI around 180 or greater.