Supermarket EDA

November 2, 2019

1 Data Gathering

1.0.1 Loading Modules

```
In [1]: import pandas as pd
        from pandas import DataFrame as df
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline
        import warnings # Ignores any warning
        warnings.filterwarnings("ignore")
```

1.0.2 Loading Dataset

```
In [2]: train = pd.read_csv("Train.csv")
        test = pd.read_csv("Test.csv")
```

2 Exploratory Data Analysis

```
In [3]: #Five point summary of numeric attributes
        train.describe()
```

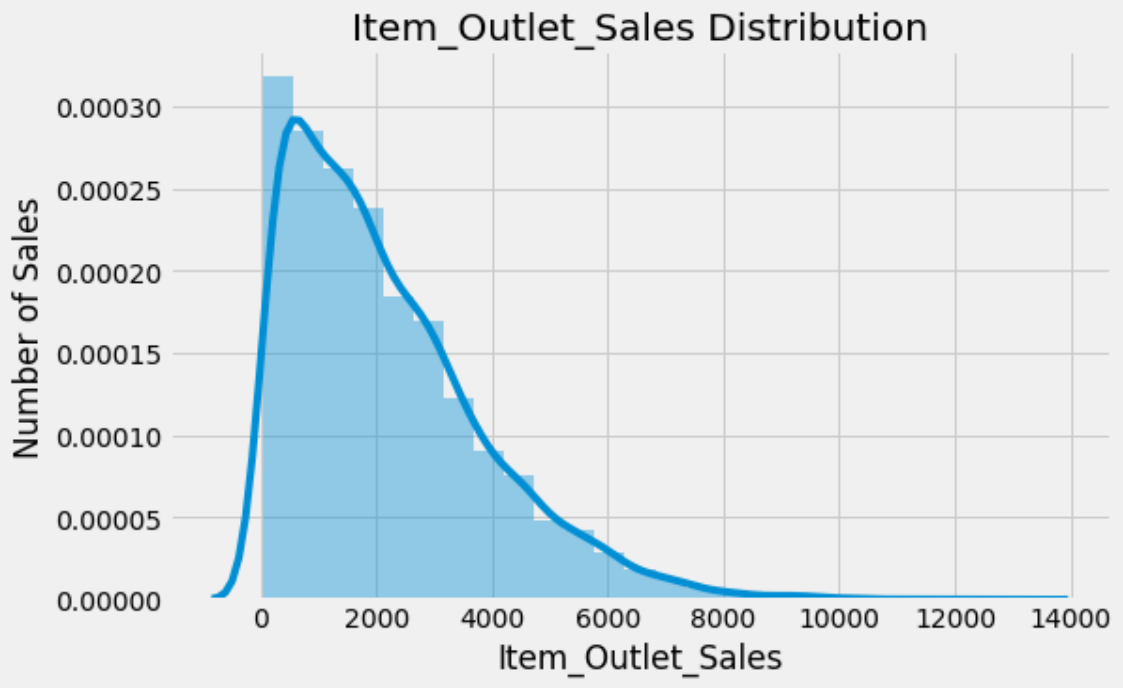
```
Out[3]:         Item_Weight  Item_Visibility     Item_MRP  Outlet_Establishment_Year  \
        count  7060.000000      8523.000000  8523.000000                8523.000000
        mean     12.857645         0.066132   140.992782                1997.831867
        std       4.643456         0.051598    62.275067                   8.371760
        min       4.555000         0.000000    31.290000                1985.000000
        25%       8.773750         0.026989    93.826500                1987.000000
        50%      12.600000         0.053931   143.012800                1999.000000
        75%      16.850000         0.094585   185.643700                2004.000000
        max      21.350000         0.328391   266.888400                2009.000000

               Item_Outlet_Sales
        count        8523.000000
        mean         2181.288914
        std          1706.499616
```

```
min            33.290000
25%           834.247400
50%          1794.331000
75%          3101.296400
max         13086.964800
```

2.0.1 Univariate Analysis

```
In [4]: # Visualizing the distributions of Item_Outlet_Sales
        plt.style.use('fivethirtyeight')
        plt.figure(figsize=(8,5))
        sns.distplot(train.Item_Outlet_Sales, bins = 25)
        plt.ticklabel_format(style='plain', axis='x', scilimits=(0,1))
        plt.xlabel("Item_Outlet_Sales")
        plt.ylabel("Number of Sales")
        plt.title("Item_Outlet_Sales Distribution")
```

```
Out[4]: Text(0.5, 1.0, 'Item_Outlet_Sales Distribution')
```




```
In [5]: #Skewness and Kurtosis
        print ("Skew is:", train.Item_Outlet_Sales.skew())
        print("Kurtosis: %f" % train.Item_Outlet_Sales.kurt())
```

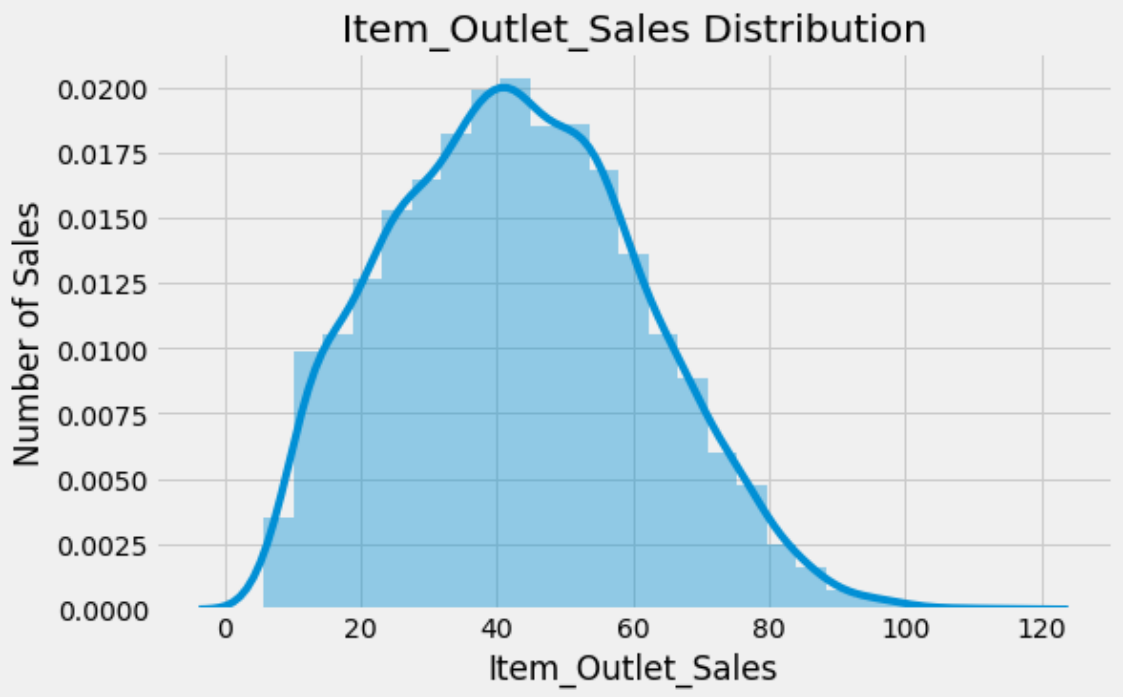
```
Skew is: 1.1775306028542798
Kurtosis: 1.615877
```

Skewness is more than 1, meaning the distribution is right skewed and Kurtosis is also more than 1, meaning it is too peaked

```
In [6]: #To make data to follow normality assumption, sqrt transformation is applied
        train['Item_Outlet_Sales']=train['Item_Outlet_Sales'].transform(func='sqrt')
        #train.head()
```

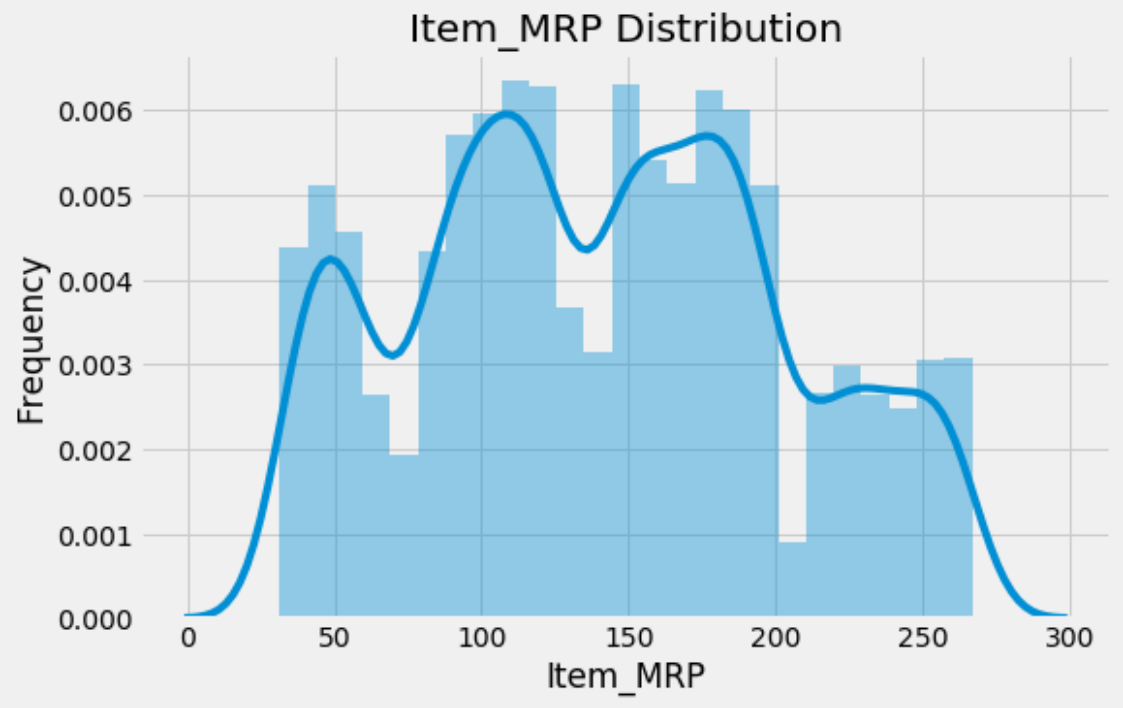
```
In [7]: plt.style.use('fivethirtyeight')
        plt.figure(figsize=(8,5))
        sns.distplot(train.Item_Outlet_Sales, bins = 25)
        plt.ticklabel_format(style='plain', axis='x', scilimits=(0,1))
        plt.xlabel("Item_Outlet_Sales")
        plt.ylabel("Number of Sales")
        plt.title("Item_Outlet_Sales Distribution")
```

```
Out[7]: Text(0.5, 1.0, 'Item_Outlet_Sales Distribution')
```



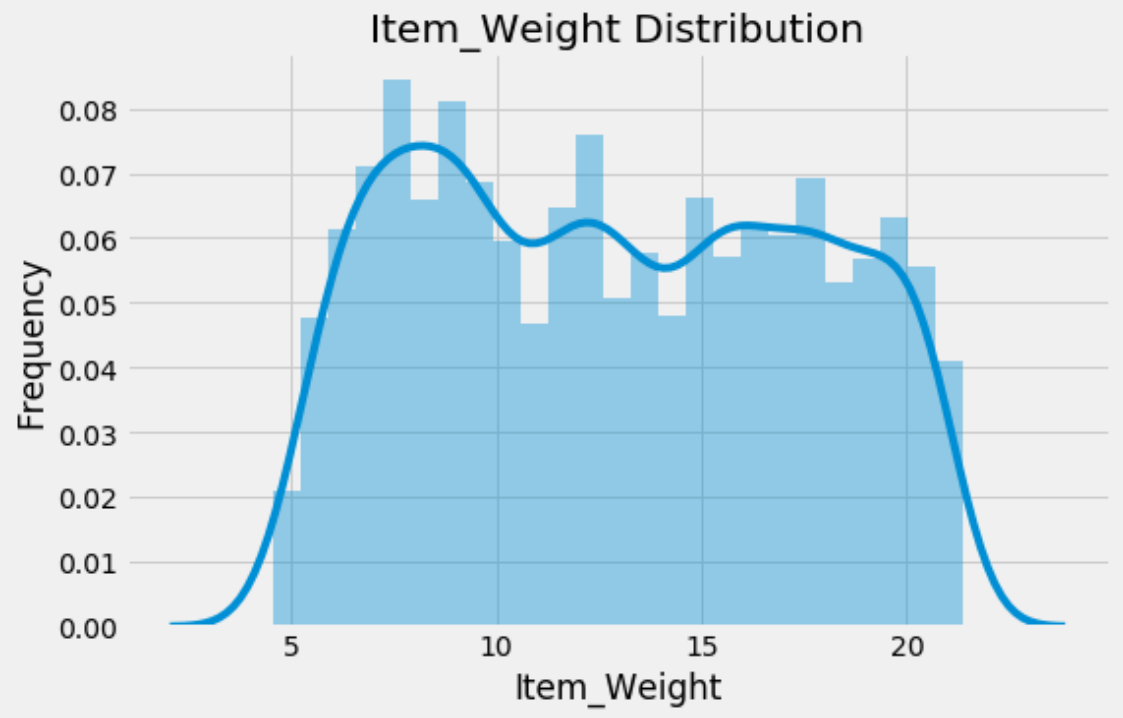

```
In [8]: plt.style.use('fivethirtyeight')
        plt.figure(figsize=(8,5))
        sns.distplot(train.Item_MRP, bins = 25)
        plt.ticklabel_format(style='plain', axis='x', scilimits=(0,1))
        plt.xlabel("Item_MRP")
        plt.ylabel("Frequency")
        plt.title("Item_MRP Distribution")
```

```
Out[8]: Text(0.5, 1.0, 'Item_MRP Distribution')
```

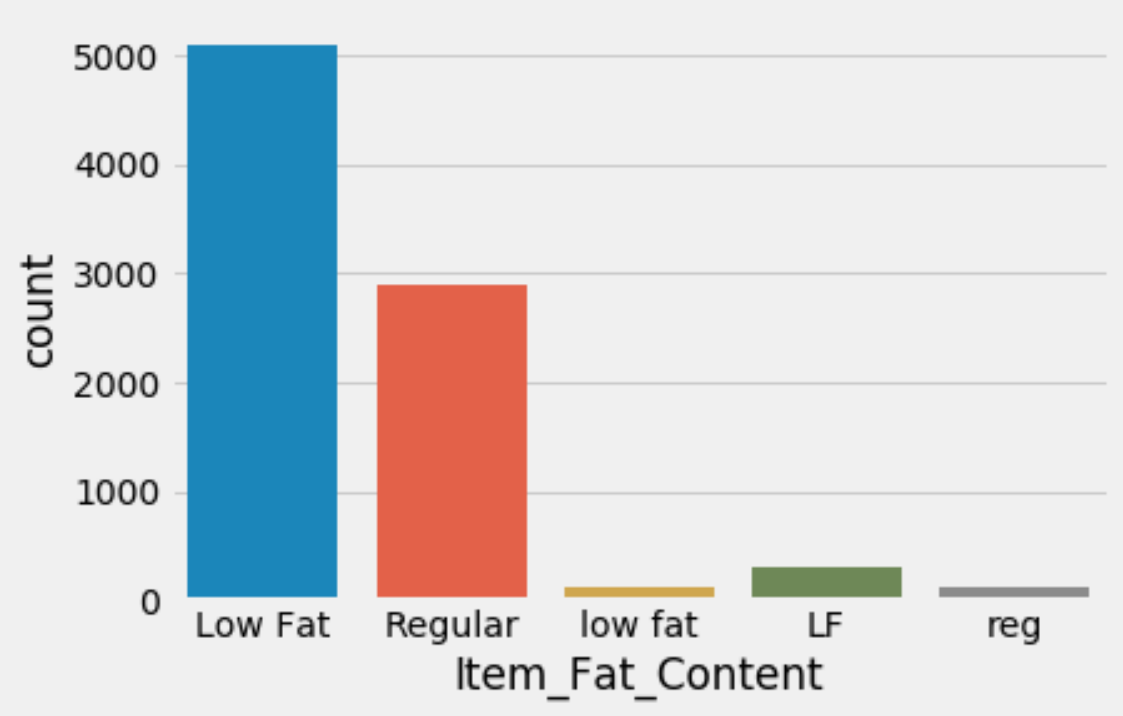



```
In [9]: print ("Skew is:", train.Item_Weight.skew())
        print("Kurtosis: %f" % train.Item_Weight.kurt())

Skew is: 0.0824262091221237
Kurtosis: -1.227766
```

```
In [10]: plt.style.use('fivethirtyeight')
         plt.figure(figsize=(8,5))
         sns.distplot(train.Item_Weight, bins = 25)
         plt.ticklabel_format(style='plain', axis='x', scilimits=(0,1))
         plt.xlabel("Item_Weight")
         plt.ylabel("Frequency")
         plt.title("Item_Weight Distribution")

Out[10]: Text(0.5, 1.0, 'Item_Weight Distribution')
```

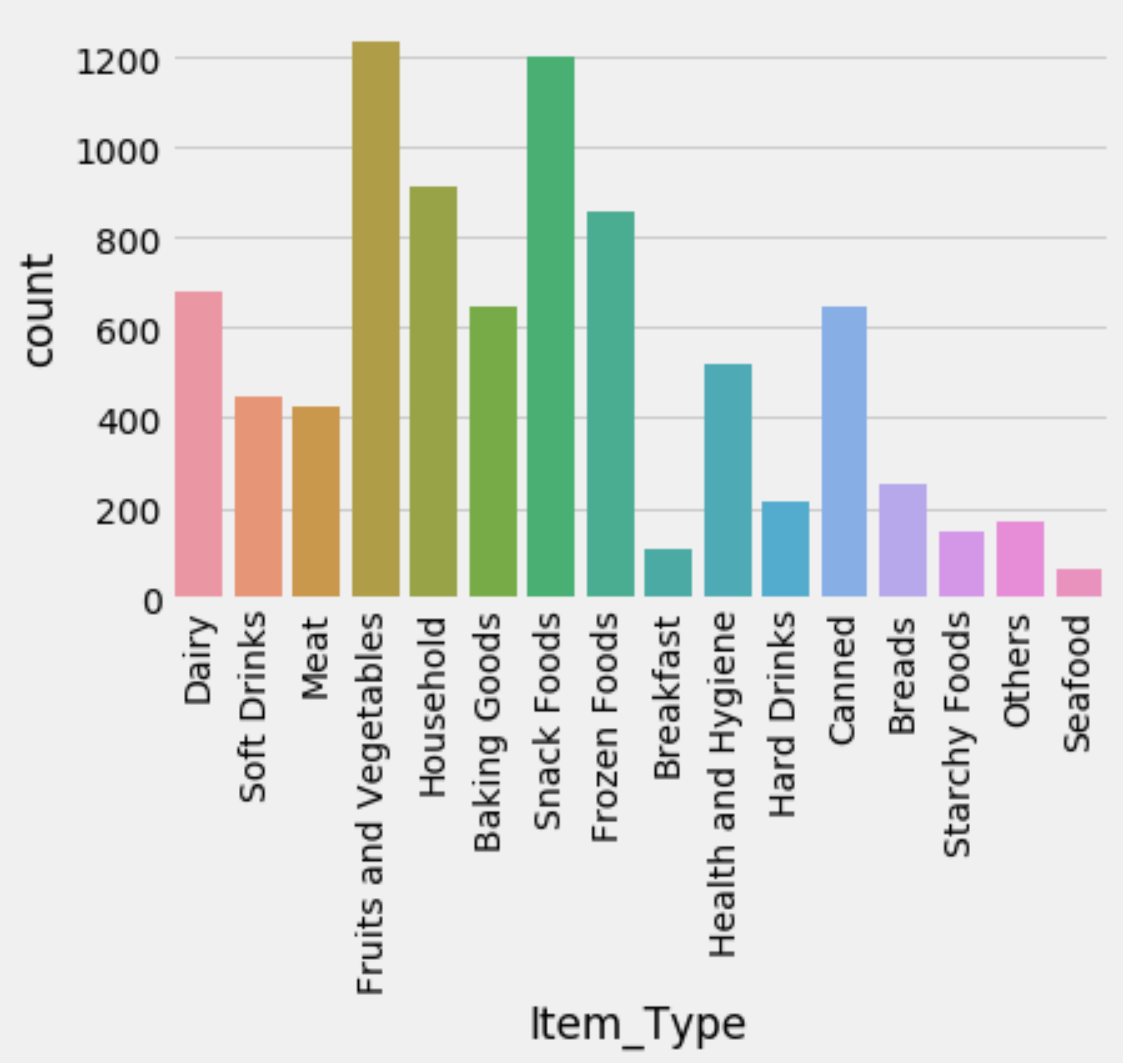



```
In [11]: print ("Skew is:", train.Item_MRP.skew())
         print("Kurtosis: %f" % train.Item_MRP.kurt())

Skew is: 0.1272022683110526
Kurtosis: -0.889769


In [12]: # Histogram of Item_Fat_Content
         sns.countplot(train.Item_Fat_Content)

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7f722e21f2e8>
```



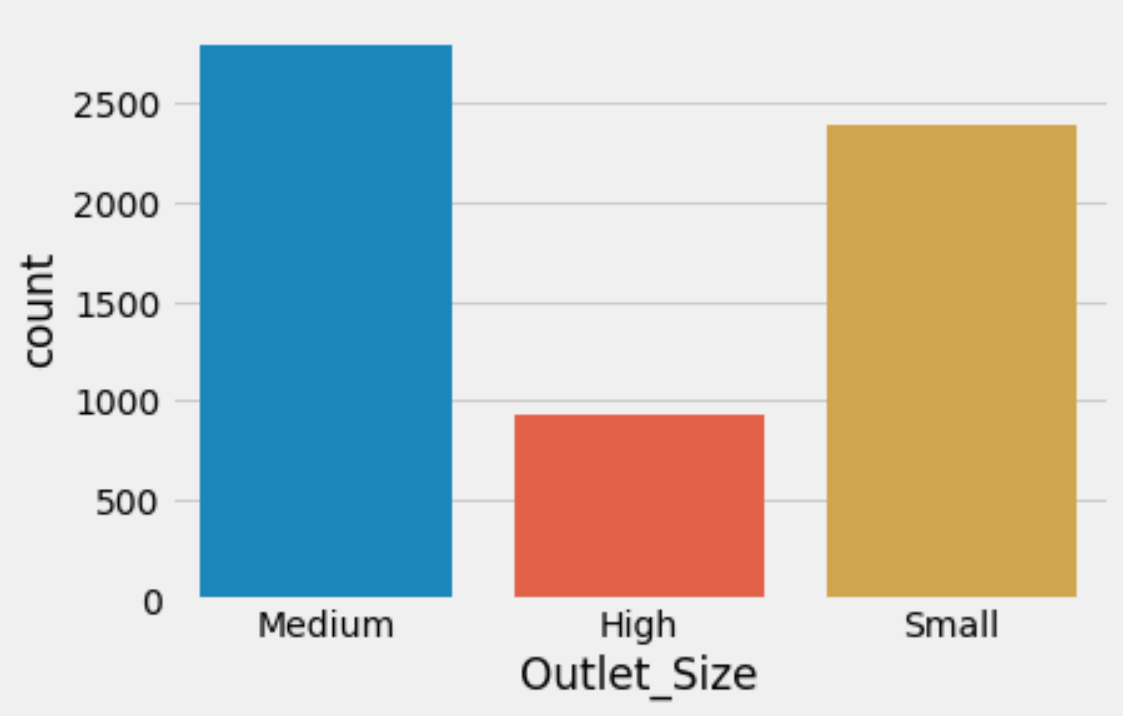
There is no consistent format maintained during integration of data. 'Low Fat', 'low fat' and 'lf' all mean the same.

```
In [13]: #Histogram of Item_type available at Mart
         sns.countplot(train.Item_Type)
         plt.xticks(rotation=90)

Out[13]: (array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15]),
          <a list of 16 Text xticklabel objects>)
```



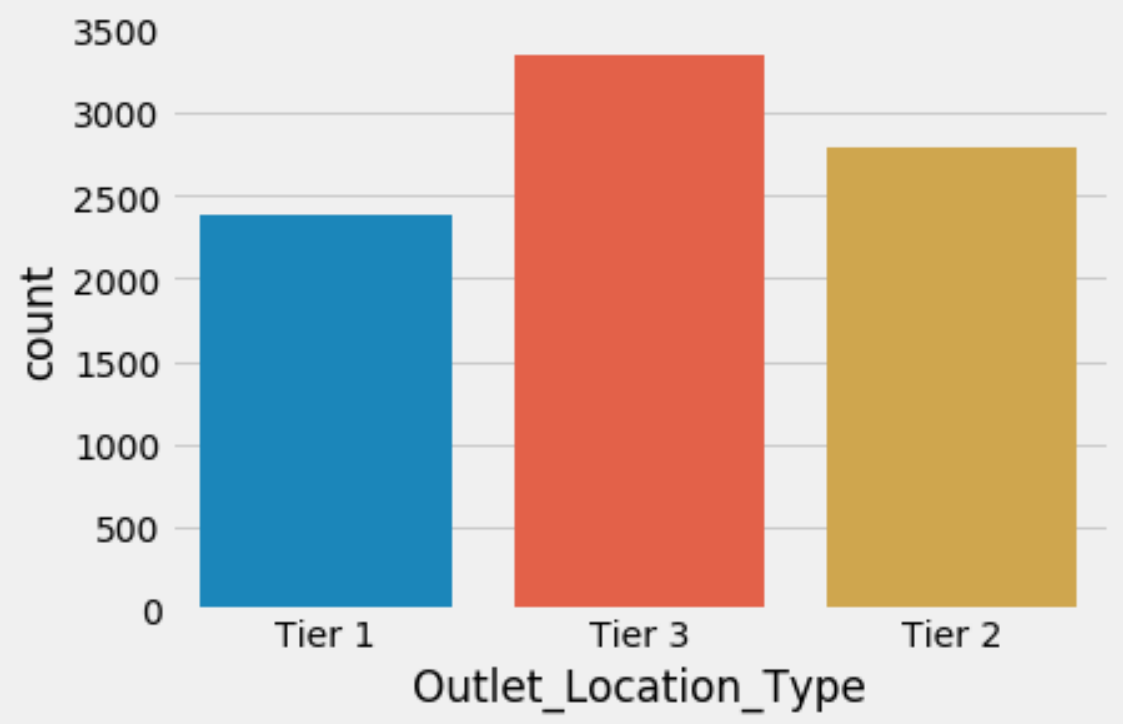
In [14]:
```
#Histogram of Outlet_Size, Outlet_Location_Type and Outlet_Type
sns.countplot(train.Outlet_Size)
```

Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f722e119320>
```



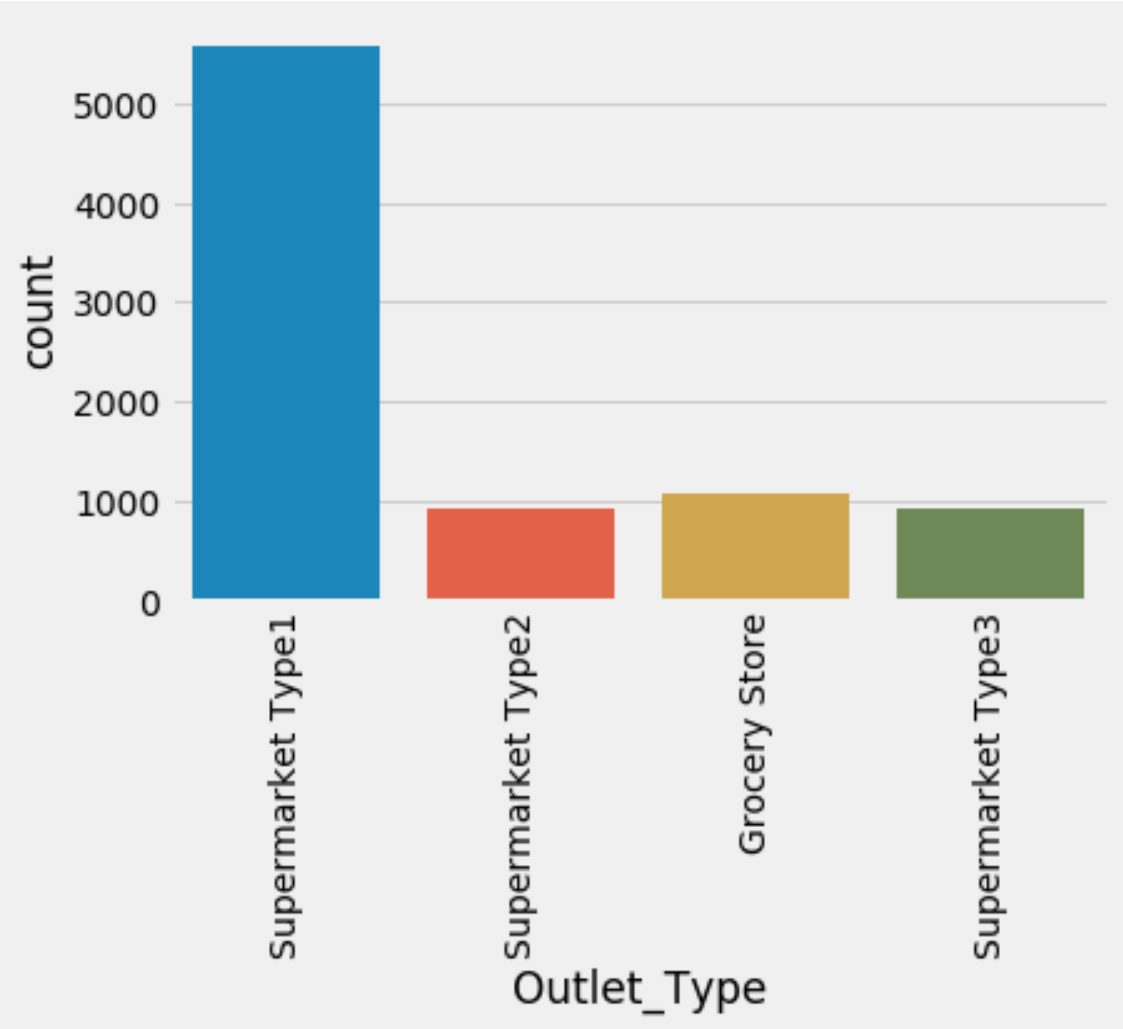

```
In [15]: sns.countplot(train.Outlet_Location_Type)
```

```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f722e114470>
```



In [16]:
```
sns.countplot(train.Outlet_Type)
plt.xticks(rotation=90)
```

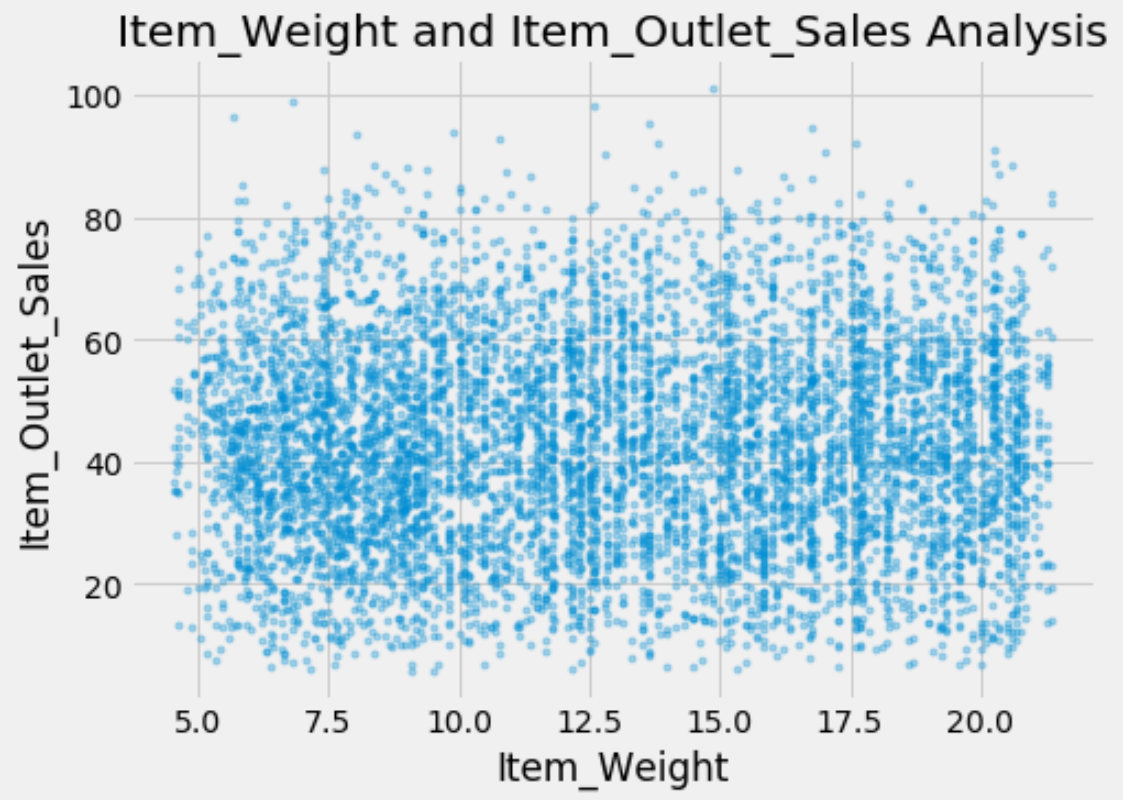
Out[16]: (array([0, 1, 2, 3]), <a list of 4 Text xticklabel objects>)



2.0.2 Bivariate Analysis

In [17]:
```
# Scatter plot of Outlet_Sales vs Item_Weight
plt.figure(figsize=(7,5))
plt.xlabel("Item_Weight")
plt.ylabel("Item_Outlet_Sales")
plt.title("Item_Weight and Item_Outlet_Sales Analysis")
plt.plot(train.Item_Weight, train["Item_Outlet_Sales"],'.', alpha = 0.3)
```

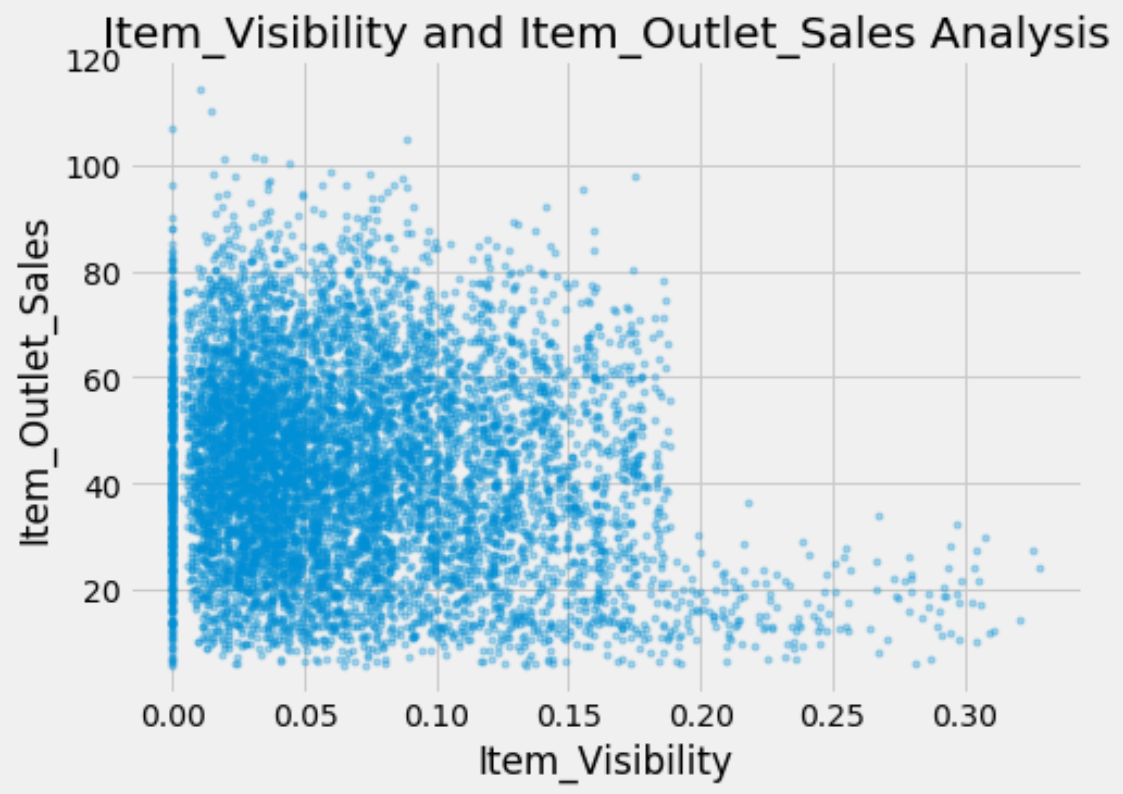
Out[17]: [<matplotlib.lines.Line2D at 0x7f722e033470>]



No significant pattern evident

In [18]:
```
#Scatter plot of Outlet_Sales vs Item_Visibility
plt.figure(figsize=(7,5))
plt.xlabel('Item_Visibility')
plt.ylabel('Item_Outlet_Sales')
plt.title('Item_Visibility and Item_Outlet_Sales Analysis')
plt.plot(train.Item_Visibility, train['Item_Outlet_Sales'],'.', alpha = 0.3)
```

Out[18]: [<matplotlib.lines.Line2D at 0x7f722dfc8d30>]



There is a negative correlation between Item_Visibility and Outlet_Sales. This is verified below.

```
In [19]: numeric_features = train.select_dtypes(include=[np.number])
         corr =numeric_features.corr()
         print(corr['Item_Outlet_Sales'].sort_values(ascending=False))
```

```
Item_Outlet_Sales            1.000000
Item_MRP                     0.563434
Item_Weight                  0.013037
Outlet_Establishment_Year    0.007511
Item_Visibility             -0.161541
Name: Item_Outlet_Sales, dtype: float64
```