In [84]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [85]:
```
train=pd.read_csv('Train.csv')
test=pd.read_csv('Test.csv')
```

In [86]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Item_Identifier            8523 non-null   object 
 1   Item_Weight                7060 non-null   float64
 2   Item_Fat_Content           8523 non-null   object 
 3   Item_Visibility            8523 non-null   float64
 4   Item_Type                  8523 non-null   object 
 5   Item_MRP                   8523 non-null   float64
 6   Outlet_Identifier          8523 non-null   object 
 7   Outlet_Establishment_Year  8523 non-null   int64  
 8   Outlet_Size                6113 non-null   object 
 9   Outlet_Location_Type       8523 non-null   object 
 10  Outlet_Type                8523 non-null   object 
 11  Item_Outlet_Sales          8523 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 799.2+ KB
```

In [87]:
```
train.shape
```

Out[87]:
```
(8523, 12)
```

In [88]:
```
train.head()
```

Out[88]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outle
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	M
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	M
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	M
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	

Missing Value

In [89]:
```
train.isna().sum().sort_values(ascending=False)
```

Out[89]:
```
Outlet_Size                  2410
Item_Weight                  1463
Item_Outlet_Sales               0
Outlet_Type                     0
Outlet_Location_Type            0
Outlet_Establishment_Year       0
Outlet_Identifier               0
Item_MRP                        0
Item_Type                       0
Item_Visibility                 0
Item_Fat_Content                0
Item_Identifier                 0
dtype: int64
```

Item_Weight

In [90]:
```
train.loc[:,('Item_Weight','Item_Fat_Content','Item_Type')]
```

Out[90]:

	Item_Weight	Item_Fat_Content	Item_Type
0	9.300	Low Fat	Dairy
1	5.920	Regular	Soft Drinks
2	17.500	Low Fat	Meat
3	19.200	Regular	Fruits and Vegetables
4	8.930	Low Fat	Household
...	...	...	...
8518	6.865	Low Fat	Snack Foods
8519	8.380	Regular	Baking Goods
8520	10.600	Low Fat	Health and Hygiene
8521	7.210	Regular	Snack Foods
8522	14.800	Low Fat	Soft Drinks

8523 rows × 3 columns

In [91]:
```
train.Item_Type.unique()
```

Out[91]:
```
array(['Dairy', 'Soft Drinks', 'Meat', 'Fruits and Vegetables',
       'Household', 'Baking Goods', 'Snack Foods', 'Frozen Foods',
       'Breakfast', 'Health and Hygiene', 'Hard Drinks', 'Canned',
       'Breads', 'Starchy Foods', 'Others', 'Seafood'], dtype=object)
```

In [92]:
```
med=train.groupby(['Item_Fat_Content','Item_Type'])['Item_Weight'].transform('median')
medt=test.groupby(['Item_Fat_Content','Item_Type'])['Item_Weight'].transform('median')
```

In [93]:
```
train.Item_Weight=np.where(train.Item_Weight.isnull(), med, train.Item_Weight)
```

In [94]:
```
test.Item_Weight=np.where(test.Item_Weight.isnull(), medt, test.Item_Weight)
```

In [95]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Item_Identifier            8523 non-null   object 
 1   Item_Weight                8523 non-null   float64
 2   Item_Fat_Content           8523 non-null   object 
 3   Item_Visibility            8523 non-null   float64
 4   Item_Type                  8523 non-null   object 
 5   Item_MRP                   8523 non-null   float64
 6   Outlet_Identifier          8523 non-null   object 
 7   Outlet_Establishment_Year  8523 non-null   int64  
 8   Outlet_Size                6113 non-null   object 
 9   Outlet_Location_Type       8523 non-null   object 
 10  Outlet_Type                8523 non-null   object 
 11  Item_Outlet_Sales          8523 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 799.2+ KB
```

Outlet_Size

In [96]:
```
Na=train.loc[:,('Outlet_Size','Outlet_Location_Type','Outlet_Type')][10:]
Na
```

Out[96]:

	Outlet_Size	Outlet_Location_Type	Outlet_Type
10	Medium	Tier 1	Supermarket Type1
11	Small	Tier 1	Supermarket Type1
12	Medium	Tier 1	Supermarket Type1
13	Small	Tier 1	Supermarket Type1
14	High	Tier 3	Supermarket Type1
...	...	...	...
8518	High	Tier 3	Supermarket Type1
8519	NaN	Tier 2	Supermarket Type1
8520	Small	Tier 2	Supermarket Type1
8521	Medium	Tier 3	Supermarket Type2
8522	Small	Tier 1	Supermarket Type1

8513 rows × 3 columns

In [97]:
```
Na[Na.Outlet_Size.isna()]
```

Out[97]:

	Outlet_Size	Outlet_Location_Type	Outlet_Type
25	NaN	Tier 2	Supermarket Type1
28	NaN	Tier 3	Grocery Store
30	NaN	Tier 3	Grocery Store
33	NaN	Tier 2	Supermarket Type1
45	NaN	Tier 3	Grocery Store
...	...	...	...
8502	NaN	Tier 2	Supermarket Type1
8508	NaN	Tier 2	Supermarket Type1
8509	NaN	Tier 3	Grocery Store
8514	NaN	Tier 2	Supermarket Type1
8519	NaN	Tier 2	Supermarket Type1

2407 rows × 3 columns

In [98]:
```
train.Outlet_Type.unique()
```

Out[98]:
```
array(['Supermarket Type1', 'Supermarket Type2', 'Grocery Store',
       'Supermarket Type3'], dtype=object)
```

In [99]:
```
filter1=train['Outlet_Type']=='Grocery Store'
filter2=train['Outlet_Location_Type']=='Tier 1'

train.where(filter1& filter2).dropna()# we cab clearly see that there is some relationship between Outl
et
```

Out[99]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	O
23	FDC37	12.300	Low Fat	0.057557	Baking Goods	107.6938	OUT019	1985.0	
29	FDC14	13.350	Regular	0.072222	Canned	43.6454	OUT019	1985.0	
49	FDS02	12.700	Regular	0.255395	Dairy	196.8794	OUT019	1985.0	
59	FDI26	10.195	Low Fat	0.061082	Canned	180.0344	OUT019	1985.0	
63	FDY40	13.350	Regular	0.150286	Frozen Foods	51.0692	OUT019	1985.0	
...	...	...	...	...	...	...	...	...	
8454	NCH54	13.150	Low Fat	0.127234	Household	158.3920	OUT019	1985.0	
8458	FDX20	12.500	Low Fat	0.074518	Fruits and Vegetables	227.3720	OUT019	1985.0	
8469	FDQ45	11.500	Regular	0.019114	Snack Foods	182.1608	OUT019	1985.0	
8480	FDQ58	14.100	Low Fat	0.000000	Snack Foods	154.5340	OUT019	1985.0	
8490	FDU44	13.800	Regular	0.102296	Fruits and Vegetables	162.3552	OUT019	1985.0	

528 rows × 12 columns

In [100]:
```
train.dropna(inplace=True)
```

In [101]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6113 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Item_Identifier            6113 non-null   object 
 1   Item_Weight                6113 non-null   float64
 2   Item_Fat_Content           6113 non-null   object 
 3   Item_Visibility            6113 non-null   float64
 4   Item_Type                  6113 non-null   object 
 5   Item_MRP                   6113 non-null   float64
 6   Outlet_Identifier          6113 non-null   object 
 7   Outlet_Establishment_Year  6113 non-null   int64  
 8   Outlet_Size                6113 non-null   object 
 9   Outlet_Location_Type       6113 non-null   object 
 10  Outlet_Type                6113 non-null   object 
 11  Item_Outlet_Sales          6113 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 620.9+ KB
```

EDA

Outlet size

In [102]:
```
len(train)
```

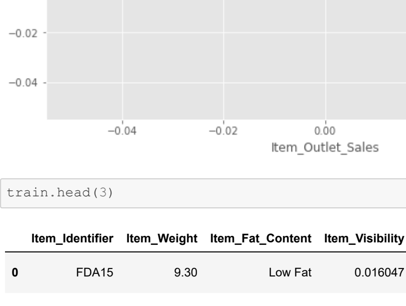
Out[102]:
```
6113
```

In [103]:
```
f,ax=plt.subplots(1,2,figsize=(18,5))
train.groupby('Outlet_Size')['Item_Outlet_Sales'].sum().plot.bar(ax=ax[0],color='red')
ax[0].set_title('Sum')# medium out the
train.groupby('Outlet_Size')['Item_Outlet_Sales'].mean().plot.bar(ax=ax[1])
ax[1].set_title('Mean')
```

Out[103]:
```
Text(0.5, 1.0, 'Mean')
```

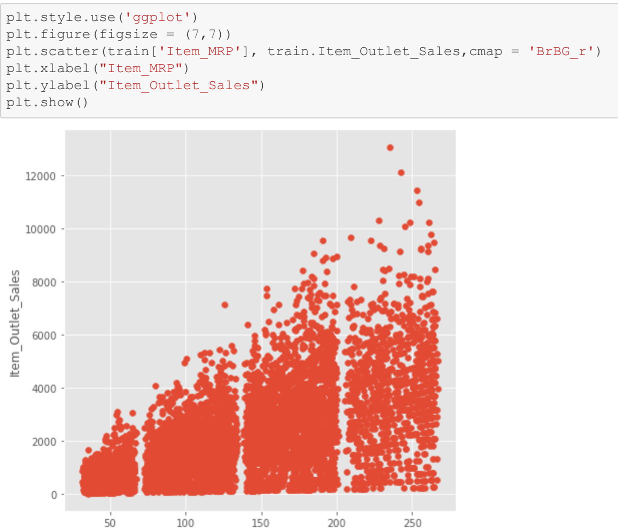
In [104]:
```
sns.countplot(train['Outlet_Size'])
```

Out[104]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e84bf278e0>
```



In [105]:
```
plt.figure (figsize = (10,5))
sns.distplot (train [train.Outlet_Size == "1"].Item_Outlet_Sales, hist = False, kde_kws = {'shade' : T
rue}, label = 'Small')
sns.distplot (train[train.Outlet_Size == "2"].Item_Outlet_Sales, hist = False, kde_kws = {'shade' : Tr
ue}, label = 'Medium')
sns.distplot (train [train.Outlet_Size == "3"].Item_Outlet_Sales, hist = False, kde_kws = {'shade' : T
rue}, label = 'High');
```

```
C:\Users\Rahul\anaconda3\lib\site-packages\seaborn\distributions.py:198: RuntimeWarning: Mean of empt
y slice.
  line, = ax.plot(a.mean(), 0)
C:\Users\Rahul\anaconda3\lib\site-packages\numpy\core\_methods.py:161: RuntimeWarning: invalid value
encountered in double_scalars
  ret = ret.dtype.type(ret / rcount)
```



In [106]:
```
train.head(3)
```

Out[106]:

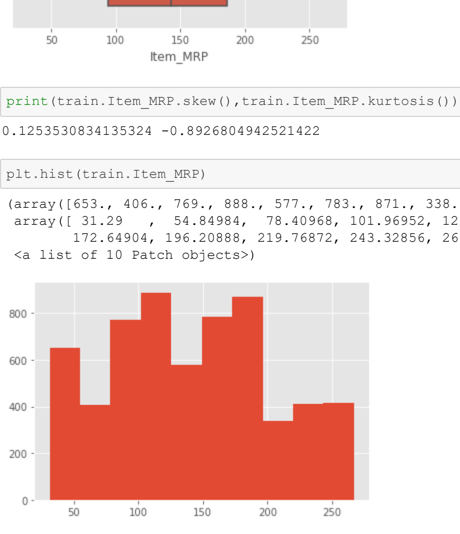
	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outle
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	M
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	M
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	M

In [107]:
```
train['Outlet_Size']=train.Outlet_Size.replace(['Medium','Small','High'],['2','1','3'])# we replaced t
hem with 1,2,3 as small<medium<high
```

In [108]:
```
train['Outlet_Size'].unique()
```

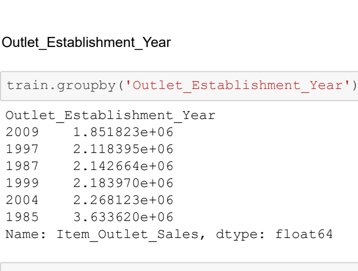
Out[108]:
```
array(['2', '3', '1'], dtype=object)
```

In [109]:
```
plt.style.use('ggplot')
plt.figure(figsize = (7,7))
plt.scatter(train['Item_MRP'], train.Item_Outlet_Sales,cmap = 'BrBG_r')
plt.xlabel("Item_MRP")
plt.ylabel("Item_Outlet_Sales")
plt.show()
```



In [110]:
```
sns.boxplot(train.Item_MRP)# there are no oultiers
```

Out[110]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e84d4c2be0>
```

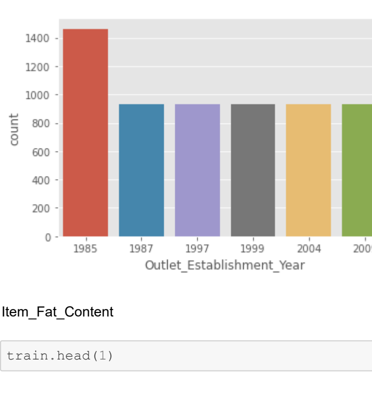


In [111]:
```
print(train.Item_MRP.skew(),train.Item_MRP.kurtosis())
```

```
0.1253530834135324 -0.8926804942521422
```

In [112]:
```
plt.hist(train.Item_MRP)
```

Out[112]:
```
(array([653., 406., 769., 888., 577., 783., 871., 338., 411., 417.]),
 array([ 31.29    ,  54.84984,  78.40968, 101.96952, 125.52936, 149.0892 ,
        172.64904, 196.20888, 219.76872, 243.32856, 266.8884 ]),
 <a list of 10 Patch objects>)
```



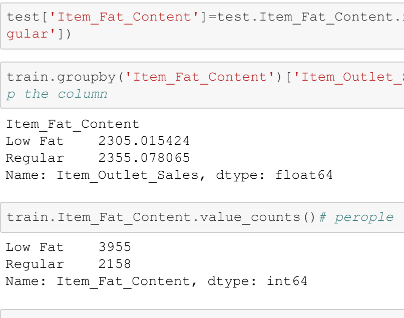
Outlet_Establishment_Year

In [113]:
```
train.groupby('Outlet_Establishment_Year')['Item_Outlet_Sales'].sum().sort_values(ascending=True)
```

Out[113]:
```
Outlet_Establishment_Year
2009    1.851823e+06
1997    2.118395e+06
1987    2.142664e+06
1999    2.183970e+06
2004    2.268123e+06
1985    3.633620e+06
Name: Item_Outlet_Sales, dtype: float64
```

In [114]:
```
sns.countplot(train['Outlet_Establishment_Year'])# most outlet were opened in 1985
```

Out[114]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e84d4c5550>
```



Item_Fat_Content

In [115]:
```
train.head(1)
```

Out[115]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outle
0	FDA15	9.3	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	

In [116]:
```
train.Item_Fat_Content.unique()
```

Out[116]:
```
array(['Low Fat', 'Regular', 'low fat', 'reg', 'LF'], dtype=object)
```

In [117]:
```
train['Item_Fat_Content']=train.Item_Fat_Content.replace(['LF','low fat','reg'],['Low Fat','Low Fat',
'Regular'])
```

In [118]:
```
train.Item_Fat_Content.nunique()
```

Out[118]:
```
2
```

In [119]:
```
test['Item_Fat_Content']=test.Item_Fat_Content.replace(['LF','low fat','reg'],['Low Fat','Low Fat','Re
gular'])
```

In [120]:
```
train.groupby('Item_Fat_Content')['Item_Outlet_Sales'].mean()# so there's no difference  so we can dro
p the column
```

Out[120]:
```
Item_Fat_Content
Low Fat    2305.015424
Regular    2355.078065
Name: Item_Outlet_Sales, dtype: float64
```

In [121]:
```
train.Item_Fat_Content.value_counts()# perople tend to prefer low fat things over regular ones
```

Out[121]:
```
Low Fat    3955
Regular    2158
Name: Item_Fat_Content, dtype: int64
```

In [122]:
```
train.drop('Item_Fat_Content',axis=1,inplace=True)
test.drop('Item_Fat_Content',axis=1,inplace=True)
```

In [123]:
```
train.head(1)
```

Out[123]:

	Item_Identifier	Item_Weight	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Loca
0	FDA15	9.3	0.016047	Dairy	249.8092	OUT049	1999	2	

Chi Sq

Outlet_Location_Type&Outlet_Type

In [124]:
```
import scipy.stats as stats
```

In [125]:
```
train.Outlet_Location_Type.value_counts()
```

Out[125]:
```
Tier 3    2795
Tier 1    2388
Tier 2     930
Name: Outlet_Location_Type, dtype: int64
```

In [126]:
```
train.Outlet_Type.value_counts()
```

Out[126]:
```
Supermarket Type1    3722
Supermarket Type3     935
Supermarket Type2     928
Grocery Store         528
Name: Outlet_Type, dtype: int64
```

In [127]:
```
contigency_table = pd.crosstab(train.Outlet_Type,train.Outlet_Location_Type, margins=True)
contigency_table
```

Out[127]:

Outlet_Location_Type	Tier 1	Tier 2	Tier 3	All
Outlet_Type				
Grocery Store	528	0	0	528
Supermarket Type1	1860	930	932	3722
Supermarket Type2	0	0	928	928
Supermarket Type3	0	0	935	935
All	2388	930	2795	6113

In [128]:
```
chi_square , p_value, degrees_of_freedom, expected_frequencies=stats.chi2_contingency(contigency_table
)
print(expected_frequencies)
```

```
[[ 206.25944708   80.3271716   241.41338132  528.        ]
 [1453.97284476  566.24570587 1701.78144937 3722.        ]
 [ 362.51660396  141.18108948  424.30230656  928.        ]
 [ 365.2511042   142.24603304  427.50286275  935.        ]
 [2388.          930.         2795.         6113.        ]]
```

In [129]:
```
print(chi_square, p_value)# hence we reject the hypothesis there is some relationship between the vari
ables
```

```
3730.483305468889 0.0
```

In [130]:
```
train.drop(['Item_Identifier','Outlet_Identifier'],inplace=True,axis=1)
test.drop(['Item_Identifier','Outlet_Identifier'],inplace=True,axis=1)
```

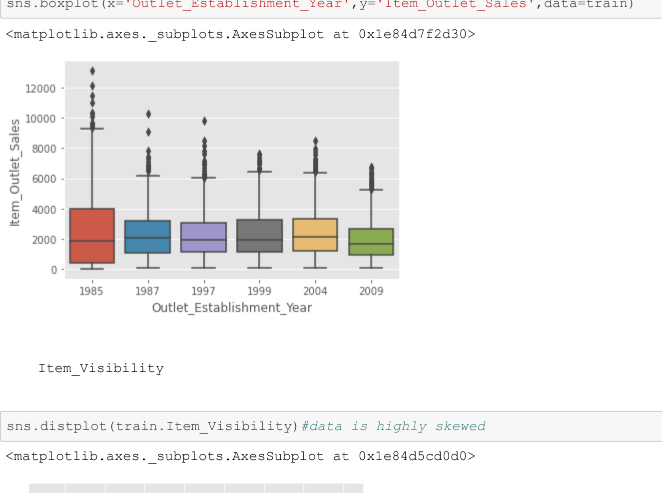
In [131]:
```
train.head()
```

Out[131]:

	Item_Weight	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Ou
0	9.300	0.016047	Dairy	249.8092	1999	2	Tier 1	Supermarket Type1	
1	5.920	0.019278	Soft Drinks	48.2692	2009	2	Tier 3	Supermarket Type2	
2	17.500	0.016760	Meat	141.6180	1999	2	Tier 1	Supermarket Type1	
4	8.930	0.000000	Household	53.8614	1987	3	Tier 3	Supermarket Type1	
5	10.395	0.000000	Baking Goods	51.4008	2009	2	Tier 3	Supermarket Type2	

In [132]:
```
plt.figure (figsize = (10,5))
sns.heatmap(train.corr(),annot=True)# there are no variables that are correlated
```

Out[132]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e84d5730d0>
```



Outlet_Establishment_Year

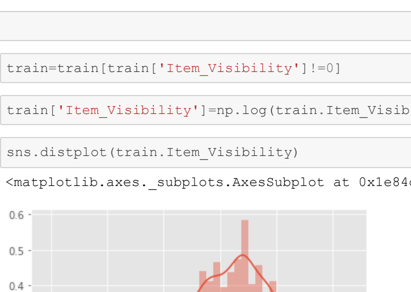
In [133]:
```
train.head(2)
```

Out[133]:

	Item_Weight	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Ou
0	9.30	0.016047	Dairy	249.8092	1999	2	Tier 1	Supermarket Type1	
1	5.92	0.019278	Soft Drinks	48.2692	2009	2	Tier 3	Supermarket Type2	

In [134]:
```
sns.boxplot(x='Outlet_Establishment_Year',y='Item_Outlet_Sales',data=train)
```

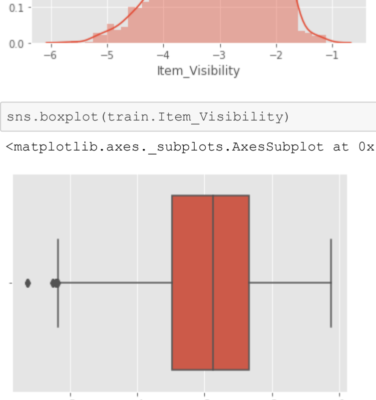
Out[134]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e84d7f2d30>
```



Item_Visibility

In [135]:
```
sns.distplot(train.Item_Visibility)#data is highly skewed
```

Out[135]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e84d5cd0d0>
```



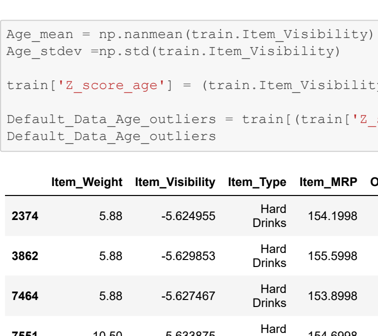
In [ ]:

In [136]:
```
train=train[train['Item_Visibility']!=0]
```

In [137]:
```
train['Item_Visibility']=np.log(train.Item_Visibility)
```

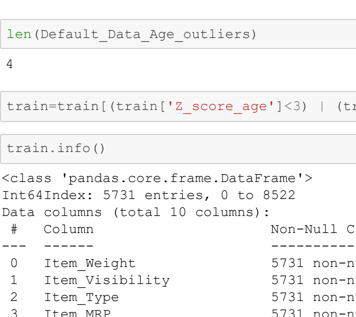
In [138]:
```
sns.distplot(train.Item_Visibility)
```

Out[138]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e84dad1610>
```



In [139]:
```
sns.boxplot(train.Item_Visibility)
```

Out[139]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e84db77f70>
```



Outlier detection with z value

In [140]:
```
Age_mean = np.nanmean(train.Item_Visibility)
Age_stdev =np.std(train.Item_Visibility)

train['Z_score_age'] = (train.Item_Visibility - Age_mean)/Age_stdev

Default_Data_Age_outliers = train[(train['Z_score_age']>3) | (train['Z_score_age'] < -3)]
Default_Data_Age_outliers
```

Out[140]:

	Item_Weight	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_
2374	5.88	-5.624955	Hard Drinks	154.1998	2009	2	Tier 3	Supermarket Type2	
3862	5.88	-5.629853	Hard Drinks	155.5998	1987	3	Tier 3	Supermarket Type1	
7464	5.88	-5.627467	Hard Drinks	153.8998	1999	2	Tier 1	Supermarket Type1	
7551	10.50	-5.633875	Hard Drinks	154.6998	1985	2	Tier 3	Supermarket Type3	

In [141]:
```
len(Default_Data_Age_outliers)
```

Out[141]:
```
4
```

In [142]:
```
train=train[(train['Z_score_age']<3) | (train['Z_score_age'] > -3)]
```

In [143]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5731 entries, 0 to 8522
Data columns (total 10 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Item_Weight                5731 non-null   float64
 1   Item_Visibility            5731 non-null   float64
 2   Item_Type                  5731 non-null   object 
 3   Item_MRP                   5731 non-null   float64
 4   Outlet_Establishment_Year  5731 non-null   int64  
 5   Outlet_Size                5731 non-null   object 
 6   Outlet_Location_Type       5731 non-null   object 
 7   Outlet_Type                5731 non-null   object 
 8   Item_Outlet_Sales          5731 non-null   float64
 9   Z_score_age                5731 non-null   float64
dtypes: float64(5), int64(1), object(4)
memory usage: 492.5+ KB
```

Annova

In [144]:
```
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

In [145]:
```
mod = ols('Item_Outlet_Sales~Item_Type', data=train) #y has to be continuous
mod = mod.fit() #ols ordinary least squares - trying to fit a line on the scatterplot b/w y and x whic
h is minimizing sse
aov = sm.stats.anova_lm(mod) #fitted ols model to be passed
print(aov)
```

```
               df        sum_sq       mean_sq         F    PR(>F)
Item_Type    15.0  9.328569e+07  6.219046e+06  2.059276  0.009225
Residual   5715.0  1.725939e+10  3.020016e+06       NaN       NaN
```

In [146]:
```
#There is a significant difference in the average prices according to the Item_type.
```

Feature Engineering

Label Encoding

In [147]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [148]:
```
train['Item_Type']=le.fit_transform(train['Item_Type'])
```

In [149]:
```
test['Item_Type']=le.transform(test['Item_Type'])
```

One Hot Encoding

In [150]:
```python
train=pd.get_dummies(data=train, columns=['Outlet_Location_Type', 'Outlet_Type'],drop_first=True)
test=pd.get_dummies(data=test, columns=['Outlet_Location_Type', 'Outlet_Type'],drop_first=True)
```

In [151]:
```python
train.head()
```

Out[151]:

	Item_Weight	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Item_Outlet_Sales	Z_score_age	Outlet_Loca
0	9.30	-4.132215	4	249.8092	1999	2	3735.1380	-1.509749	
1	5.92	-3.948780	14	48.2692	2009	2	443.4228	-1.275974	
2	17.50	-4.088756	10	141.6180	1999	2	2097.2700	-1.454364	
6	13.65	-4.362923	13	57.6588	1987	3	343.5528	-1.803770	
7	14.10	-2.059875	13	107.7622	1985	2	4022.7636	1.131299	

In [152]:
```python
train.drop(['Outlet_Establishment_Year','Z_score_age',],inplace=True,axis=1)
```

In [153]:
```python
train.head()
```

Out[153]:

	Item_Weight	Item_Visibility	Item_Type	Item_MRP	Outlet_Size	Item_Outlet_Sales	Outlet_Location_Type_Tier 2	Outlet_Location_Type_Tier 3
0	9.30	-4.132215	4	249.8092	2	3735.1380	0	0
1	5.92	-3.948780	14	48.2692	2	443.4228	0	1
2	17.50	-4.088756	10	141.6180	2	2097.2700	0	0
6	13.65	-4.362923	13	57.6588	3	343.5528	0	1
7	14.10	-2.059875	13	107.7622	2	4022.7636	0	1

SCALIING

In [154]:
```python
from sklearn.preprocessing import StandardScaler
```

In [155]:
```python
Scale=StandardScaler()
```

In [156]:
```python
train[['Item_Weight','Item_Visibility','Item_MRP']]=Scale.fit_transform(train[['Item_Weight','Item_Vis
ibility','Item_MRP']])
```

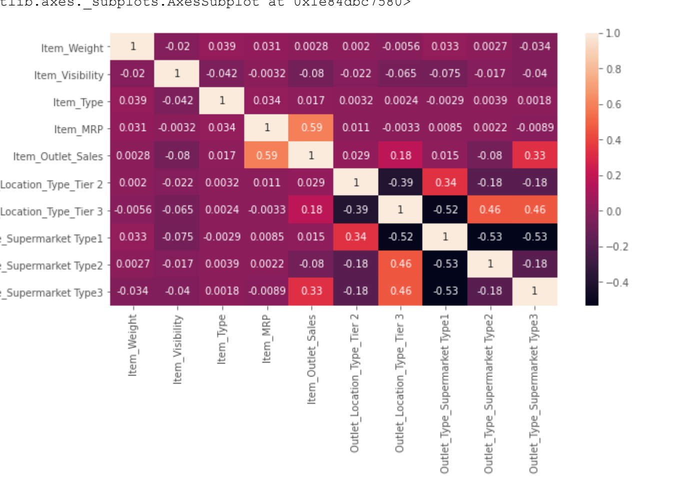
In [157]:
```python
train.head()
```

Out[157]:

	Item_Weight	Item_Visibility	Item_Type	Item_MRP	Outlet_Size	Item_Outlet_Sales	Outlet_Location_Type_Tier 2	Outlet_Location_Type_Tier 3
0	-0.853122	-1.509749	4	1.742629	2	3735.1380	0	0
1	-1.670407	-1.275974	14	-1.499642	2	443.4228	0	1
2	1.129638	-1.454364	10	0.002105	2	2097.2700	0	0
6	0.198708	-1.803770	13	-1.348587	3	343.5528	0	1
7	0.307518	1.131299	13	-0.542550	2	4022.7636	0	1

In [158]:
```python
plt.figure(figsize=(10,5))
sns.heatmap(train.corr(),annot=True)
```

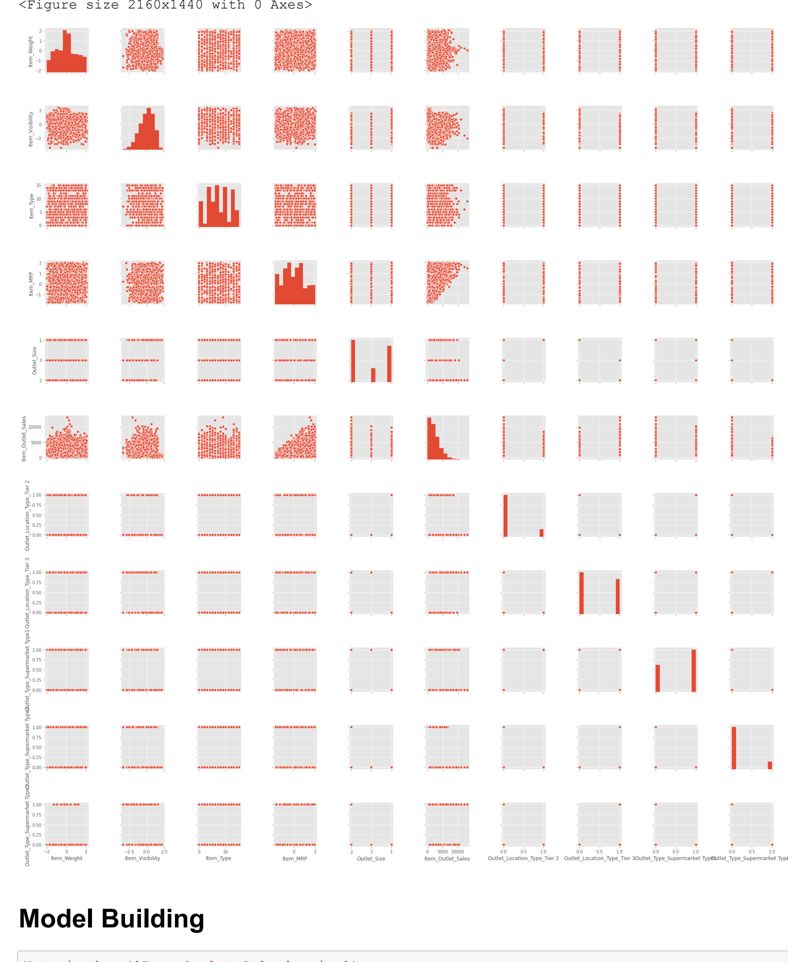
Out[158]: <matplotlib.axes._subplots.AxesSubplot at 0x1e84dbc7580>



In [159]:
```python
plt.figure(figsize=(30,20))
sns.pairplot(train)
```

Out[159]: <seaborn.axisgrid.PairGrid at 0x1e84dc41400>

<Figure size 2160x1440 with 0 Axes>



Model Building

In [160]:
```python
X=train.drop('Item_Outlet_Sales',axis=1)
X.shape
```

Out[160]: (5731, 10)

In [161]:
```python
X_scaled=Scale.fit_transform(X)
```

In [162]:
```python
#X_scaled=pd.DataFrame(X_scaled,columns=X.columns)
#X_scaled.head()
```

In [163]:
```python
import statsmodels
from statsmodels.stats.outliers_influence import variance_inflation_factor
variables = X_scaled

vif = pd.DataFrame()

vif["VIF"] = [variance_inflation_factor(variables, i) for i in range(variables.shape[1])]

vif["Features"] = X.columns
vif
```

Out[163]:

	VIF	Features
0	1.004546	Item_Weight
1	1.045574	Item_Visibility
2	1.004702	Item_Type
3	1.002355	Item_MRP
4	6.391509	Outlet_Size
5	1.701447	Outlet_Location_Type_Tier 2
6	9.750995	Outlet_Location_Type_Tier 3
7	4.435198	Outlet_Type_Supermarket Type1
8	4.059739	Outlet_Type_Supermarket Type2
9	4.105065	Outlet_Type_Supermarket Type3

In [164]:
```python
X_scaled=pd.DataFrame(X_scaled,columns=X.columns)
X_scaled.head()
X_scaled.drop(["Item_Weight",'Item_Visibility','Item_Type','Outlet_Size','Outlet_Location_Type_Tier 2'
,'Outlet_Location_Type_Tier 3'],axis=1,inplace=True)
```

In [165]:
```python
Y=train.loc[:,"Item_Outlet_Sales"]
Y
```

Out[165]:
```
0       3735.1380
1        443.4228
2       2097.2700
6        343.5528
7       4022.7636
          ...
8517    3608.6360
8518    2778.3834
8520    1193.1136
8521    1845.5976
8522     765.6700
Name: Item_Outlet_Sales, Length: 5731, dtype: float64
```

In [166]:
```python
from sklearn.model_selection import train_test_split
```

In [167]:
```python
X_train, X_test, y_train,y_test = train_test_split(X_scaled,Y, test_size = .3, random_state = 3)
```

In [212]:
```python
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import accuracy_score
from sklearn.svm import SVR
from sklearn.ensemble import AdaBoostRegressor
from sklearn.neighbors import KNeighborsRegressor
```

In [169]:
```python
model=LinearRegression()
model1=Ridge()
model2=RidgeCV()
model3=Lasso()
model4=LassoCV()
Sv=SVR()
```

In [217]:
```python
model.fit(X_train,y_train)
Sv.fit(X_train,y_train)
print(model.intercept_)
print(model.coef_)
print("R2 score of the model on test data SVR")
print(Sv.score(X_test,y_test))
print("R2 score of the model on test data")
print(model.score(X_test,y_test))
```

```
2316.3422104849938
[1024.4685246   973.4793457   578.60475165 1213.79075137]
R2 score of the model on test data SVR
0.11901720116178194
R2 score of the model on test data
0.5518203802646215
```

In [171]:
```python
pd.DataFrame(model.coef_,index=X_scaled.columns)
```

Out[171]:

	0
Item_MRP	1024.468525
Outlet_Type_Supermarket Type1	973.479346
Outlet_Type_Supermarket Type2	578.604752
Outlet_Type_Supermarket Type3	1213.790751

AdaBoost

In [201]:
```python
Ab=AdaBoostRegressor(n_estimators=1000,loss='linear')
```

In [202]:
```python
Ab.fit(X_train,y_train)
```

Out[202]: AdaBoostRegressor(n_estimators=1000)

In [203]:
```python
print(Ab.score(X_test,y_test))
print('===================')
print(Ab.score(X_train,y_train))
```

```
0.47646286731213217
===================
0.49775495077907395
```

In [204]:
```python
Ab.get_params()
```

Out[204]:
```
{'base_estimator': None,
 'learning_rate': 1.0,
 'loss': 'linear',
 'n_estimators': 1000,
 'random_state': None}
```

In [191]:
```python
Features=pd.DataFrame(Ab.feature_importances_,index=X_scaled.columns)
Features
```

Out[191]:

	0
Item_MRP	0.778474
Outlet_Type_Supermarket Type1	0.054359
Outlet_Type_Supermarket Type2	0.009638
Outlet_Type_Supermarket Type3	0.157529

In [192]:
```python
from sklearn.model_selection import GridSearchCV
```

In [206]:
```python
params = {'learning_rate':np.linspace(0.1,1,10)}
```

In [208]:
```python
grid=GridSearchCV(estimator=Ab,cv=5,param_grid=params,n_jobs=-1)
```

In [209]:
```python
grid.fit(X_train,y_train)
```

Out[209]:
```
GridSearchCV(cv=5, estimator=AdaBoostRegressor(n_estimators=1000), n_jobs=-1,
             param_grid={'learning_rate': array([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ])})
```

In [210]:
```python
grid.best_params_
```

Out[210]: {'learning_rate': 0.1}

In [211]:
```python
grid.best_score_
```

Out[211]: 0.4776963754777038

Polynomial

In [174]:
```python
polynomial_features= PolynomialFeatures(degree=2)

x_poly2 = polynomial_features.fit_transform(X_train)
```

In [175]:
```python
x_poly2.shape
```

Out[175]: (4011, 15)

In [176]:
```python
model.fit(x_poly2 ,y_train)
```

Out[176]: LinearRegression()

In [177]:
```python
model.score(x_poly2 ,y_train)
```

Out[177]: 0.5981235933727165

Assumption check

In [178]:
```python
import scipy.stats as stats
import statsmodels.formula.api as sm
```

In [ ]:

In [179]:
```python
model_m = sm.ols(formula='Item_Outlet_Sales~ Item_MRP', data = train)

# fitting the model
model_fit_m = model_m.fit()

# summary of the model
model_fit_m.summary()
```

Out[179]:

OLS Regression Results

Dep. Variable:	Item_Outlet_Sales	R-squared:	0.346
Model:	OLS	Adj. R-squared:	0.346
Method:	Least Squares	F-statistic:	3038.
Date:	Mon, 28 Sep 2020	Prob (F-statistic):	0.00
Time:	16:30:34	Log-Likelihood:	-49676.
No. Observations:	5731	AIC:	9.936e+04
Df Residuals:	5729	BIC:	9.937e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2323.9633	18.585	125.047	0.000	2287.530	2360.396
Item_MRP	1024.2676	18.585	55.114	0.000	987.835	1060.701

Omnibus:	524.572	Durbin-Watson:	1.991
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1174.416
Skew:	0.570	Prob(JB):	9.52e-256
Kurtosis:	4.902	Cond. No.	1.00

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.