Import Libraries

In [ ]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
from scipy import stats
import os,sys
from scipy.stats import norm, skew
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
import warnings
warnings.filterwarnings('ignore')
```

Import data

In [ ]:

```
df_train=pd.read_csv('Train.csv')
df_test=pd.read_csv('Test.csv')
```

Data Understanding

In [ ]:

```
df_train.head()
```

Out[3]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identif
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT0
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT0
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT0
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT0
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT0

In [ ]:

```
df_train.dtypes
```

Out[4]:

```
Item_Identifier               object
Item_Weight                  float64
Item_Fat_Content              object
Item_Visibility              float64
Item_Type                     object
Item_MRP                     float64
Outlet_Identifier             object
Outlet_Establishment_Year      int64
Outlet_Size                   object
Outlet_Location_Type          object
Outlet_Type                   object
Item_Outlet_Sales            float64
dtype: object
```

In [ ]:

```
df_train.shape
```

Out[5]:

```
(8523, 12)
```

In [ ]:

```
df_train.isna().any()
```

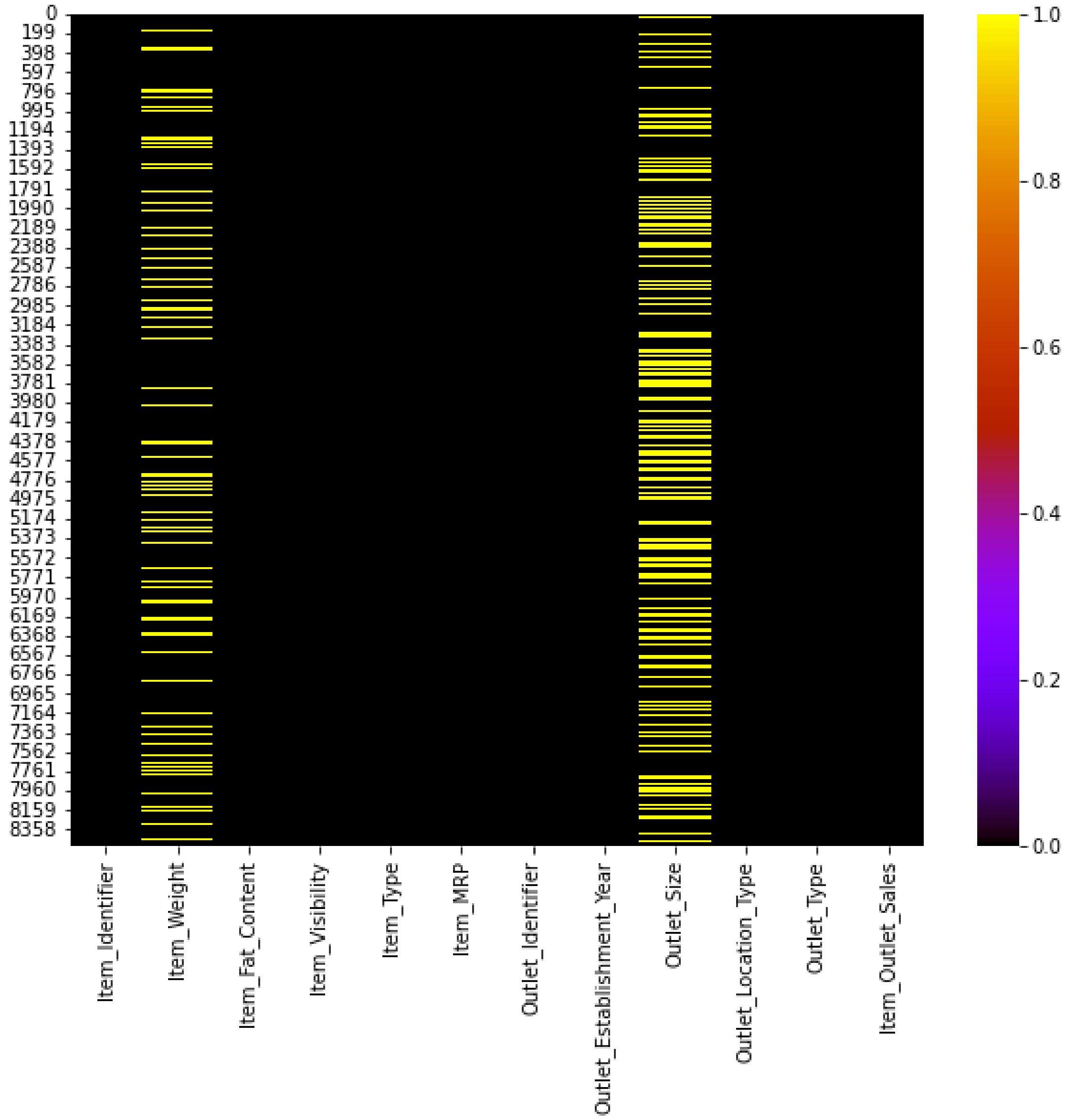
Out[6]:

```
Item_Identifier              False
Item_Weight                   True
Item_Fat_Content             False
Item_Visibility              False
Item_Type                    False
Item_MRP                     False
Outlet_Identifier            False
Outlet_Establishment_Year    False
Outlet_Size                   True
Outlet_Location_Type         False
Outlet_Type                  False
Item_Outlet_Sales            False
dtype: bool
```

Data Visualization

In [ ]:

```
plt.figure(figsize=(10,8))
sns.heatmap(df_train.isna(),cmap='gnuplot')
plt.show()
```



yellow mark show null values

1. Item Weight

In [ ]:

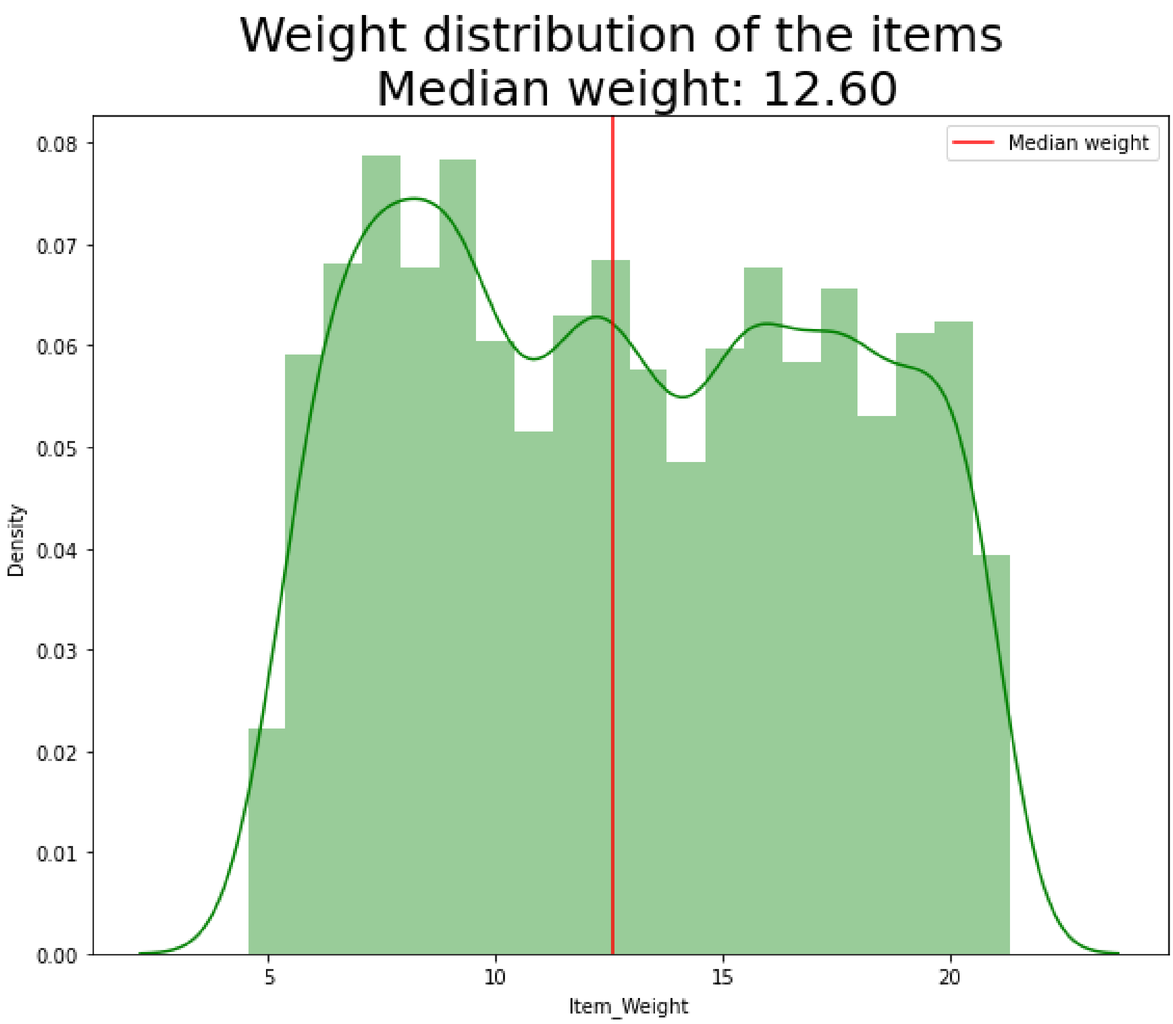
```
df_train['Item_Weight'].describe()
```

Out[8]:

```
count    7060.000000
mean       12.857645
std         4.643456
min         4.555000
25%         8.773750
50%        12.600000
75%        16.850000
max        21.350000
Name: Item_Weight, dtype: float64
```

In [ ]:

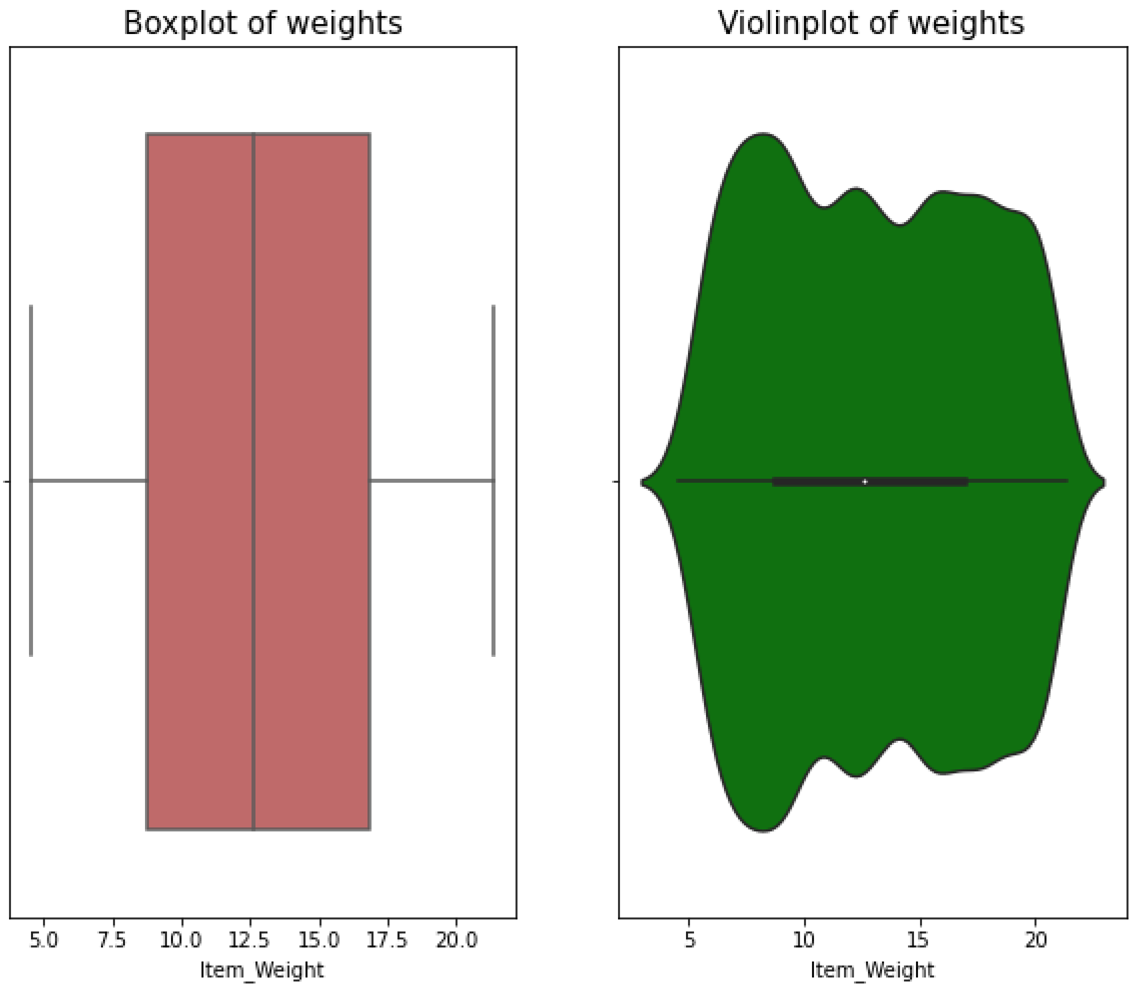
```
plt.figure(figsize=(10,8))
sns.distplot(df_train['Item_Weight'].dropna(),color='green')
plt.title('Weight distribution of the items \n Median weight: {0:.2f}'.format(df_train['Ite
plt.axvline(df_train['Item_Weight'].dropna().median(),color='red',label='Median weight')
plt.legend()
plt.show()
```



In [ ]:

```
fig1=plt.figure(figsize=(10,8))
ax1=fig1.add_subplot(121)
sns.boxplot(df_train['Item_Weight'],ax=ax1,orient='v',color='indianred')
ax1.set_title('Boxplot of weights',size=15)

ax2=fig1.add_subplot(122)
sns.violinplot(df_train['Item_Weight'],ax=ax2,orient='v',color='green')
ax2.set_title('Violinplot of weights',size=15)
plt.show()
```



As we can see from the above violin and distplot, the curve platueus over a large range of weights. Hence, it is simply not possible for us to assume a weight for the null values. We shall leave them as it is or drop them if it is later deemed to not be too important in our analysis.

2. Item Fat Content

In [ ]:

```
df_train['Item_Fat_Content'].unique()
```

Out[11]:

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

In [ ]:

```
df_train['Item_Fat_Content']=df_train['Item_Fat_Content'].replace('low fat','Low Fat')
df_train['Item_Fat_Content']=df_train['Item_Fat_Content'].replace('LF','Low Fat')
df_train['Item_Fat_Content']=df_train['Item_Fat_Content'].replace('reg','Regular')
df_train['Item_Fat_Content'].unique()
```

Out[12]:

```
array(['Low Fat', 'Regular'], dtype=object)
```

In [ ]:

```
df_train['Count']=1
df_fat=df_train.groupby('Item_Fat_Content')['Count'].sum().reset_index()

fig2=px.pie(df_fat,values='Count',names='Item_Fat_Content',hole=0.4)

fig2.update_layout(title='Fat content',title_x=0.48,
                  annotations=[dict(text='Fat',font_size=15, showarrow=False,height=800,wid
fig2.update_traces(textfont_size=15,textinfo='percent+label')
fig2.show()
```

3. Item Visibility

In [ ]:

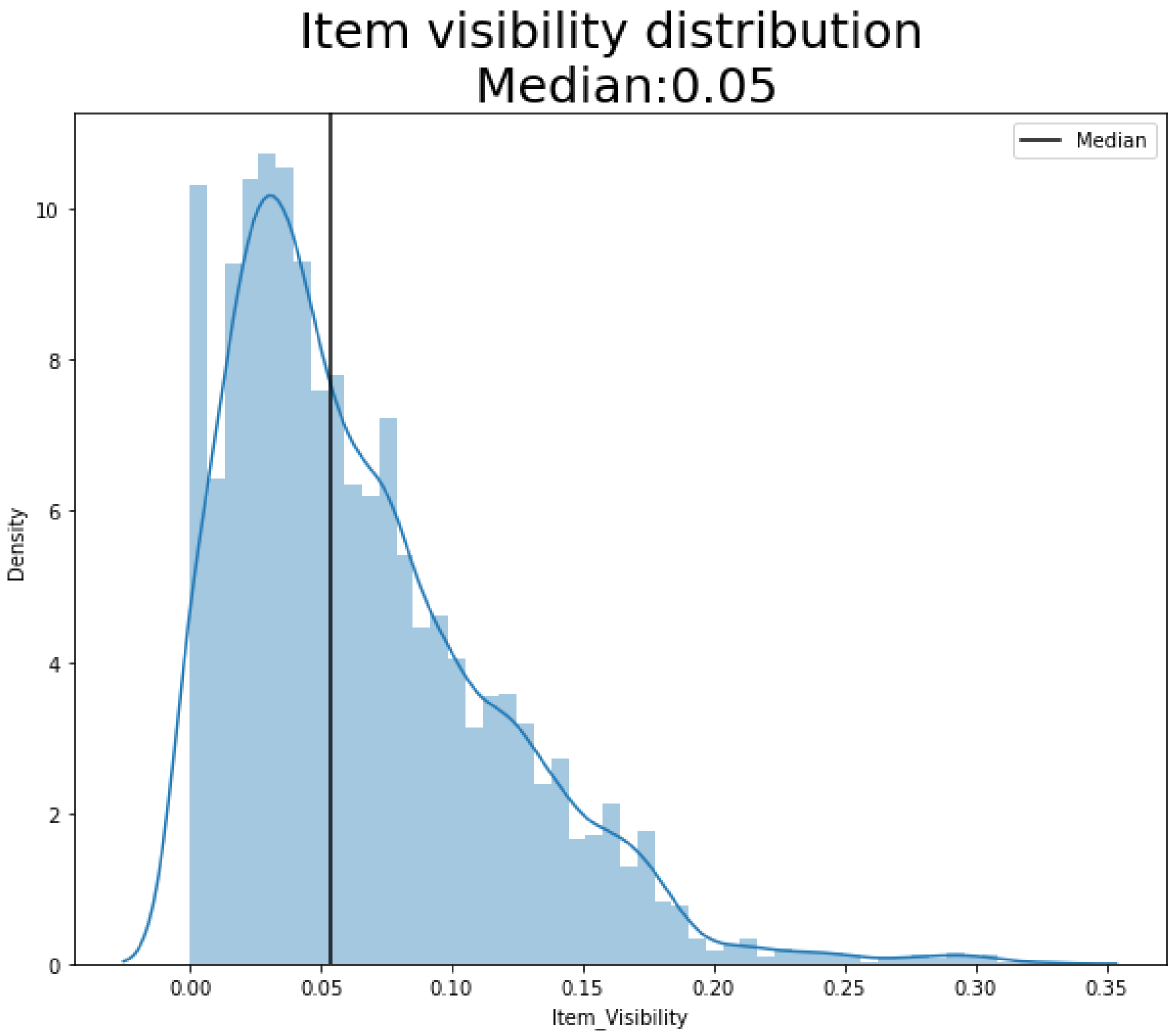
```
df_train['Item_Visibility'].describe()
```

Out[14]:

```
count    8523.000000
mean        0.066132
std         0.051598
min         0.000000
25%         0.026989
50%         0.053931
75%         0.094585
max         0.328391
Name: Item_Visibility, dtype: float64
```

In [ ]:

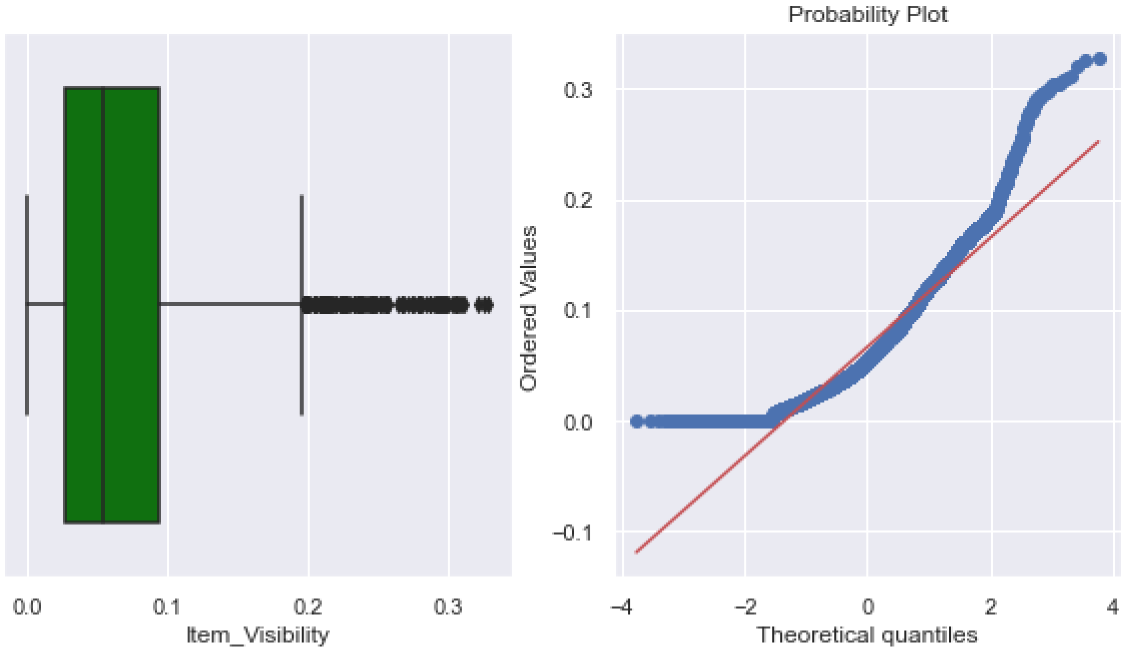
```
plt.figure(figsize=(10,8))
sns.distplot(df_train['Item_Visibility'])
plt.title('Item visibility distribution \n Median:{0:.2f}'.format(df_train['Item_Visibility
plt.axvline(df_train['Item_Visibility'].median(),color='black',label='Median')
plt.legend()
plt.show()
```



we can see that it is right skew curve. Hence, a median would give us better indication than a mean value.

In [ ]:

```
sns.set()
fig3=plt.figure(figsize=(10,5))
ax1=fig3.add_subplot(121)
sns.boxplot(df_train['Item_Visibility'],orient='v',ax=ax1,color='green')
ax2=fig3.add_subplot(122)
stats.probplot(df_train['Item_Visibility'],plot=ax2)
plt.show()
```



As we can see, values above 0.2 visibility are outliers. Presence of outliers don't bode well with machine learning algos. Hence, we need to remove the outliers and try to form a normal distribution.

The probplot also seems to suggest that the values are deviating from the normal values after 0.2

In [ ]:

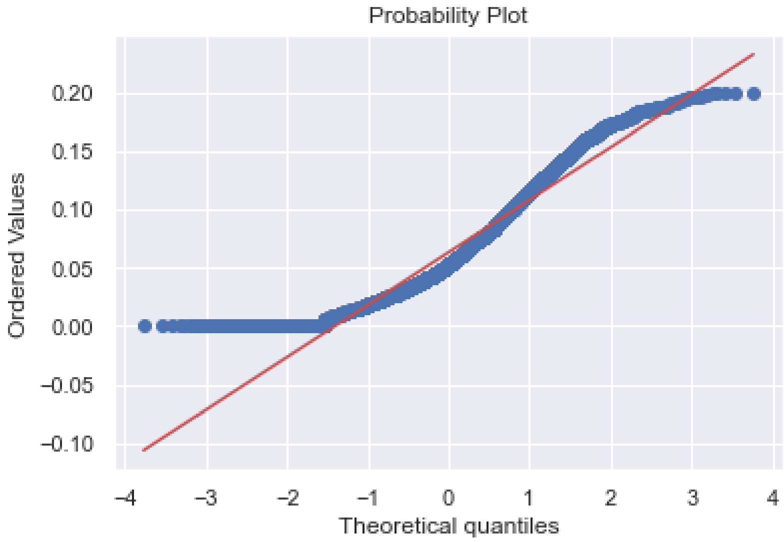
```
df_train[df_train['Item_Visibility']>0.2].shape[0]
```

Out[17]:

```
134
```

In [ ]:

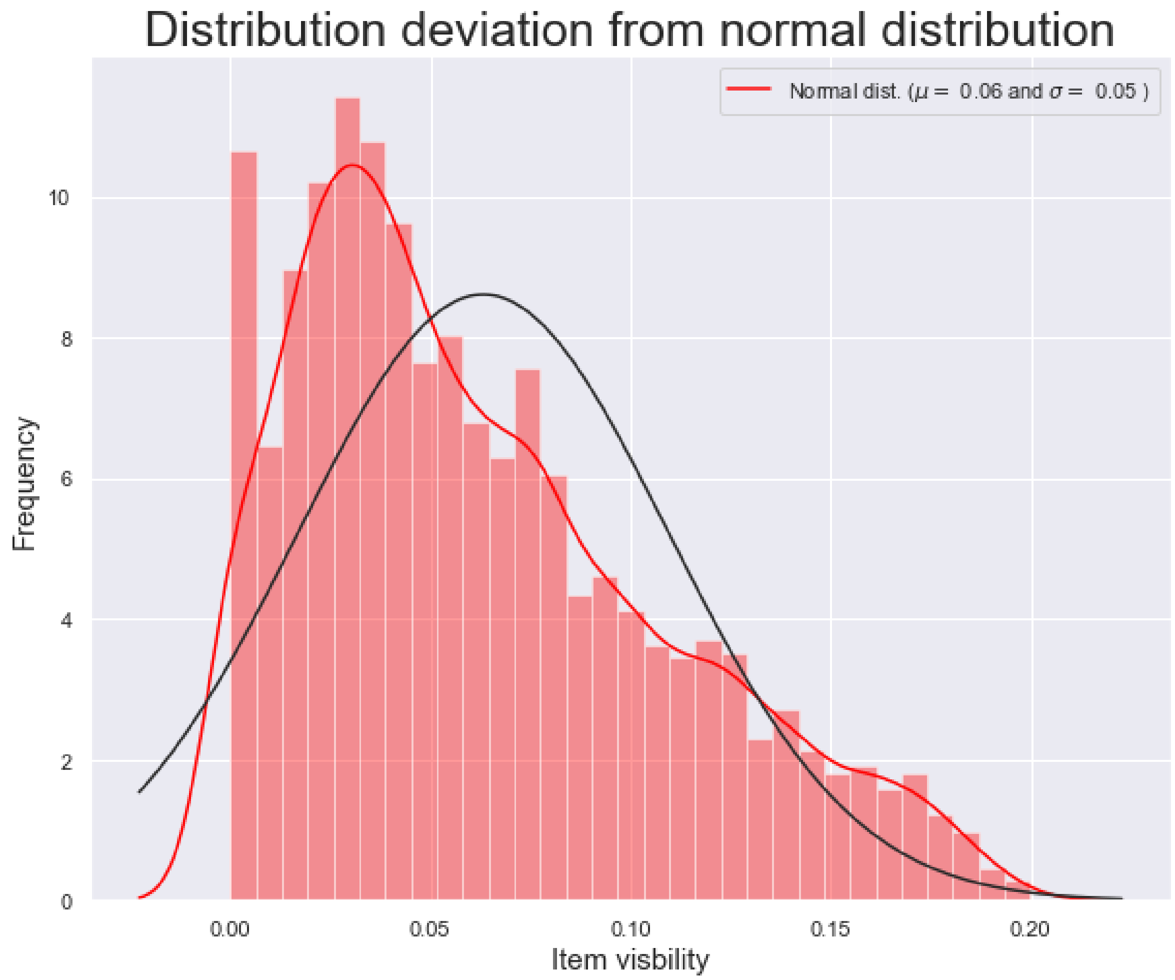
```
df_train=df_train[df_train['Item_Visibility']<0.2]
stats.probplot(df_train['Item_Visibility'],plot=plt)
plt.show()
```



Now, we see that the values above 0 are following a normal distribution to some extent.

In [ ]:

```
plt.figure(figsize=(10,8))
sns.distplot(df_train['Item_Visibility'],fit=norm,color='red')
plt.title('Distribution deviation from normal distribution',size=25)
plt.ylabel('Frequency',size=15)
plt.xlabel('Item visbility',size=15)
mu=df_train['Item_Visibility'].mean()
sigma=df_train['Item_Visibility'].std()
plt.legend(['Normal dist. ($\mu=$ {0:.2f} and $\sigma=$ {1:.2f} )'.format(mu, sigma)])
plt.show()
```



This is as close to a normal distribution I could get for our model

4. Item Type

In [ ]:

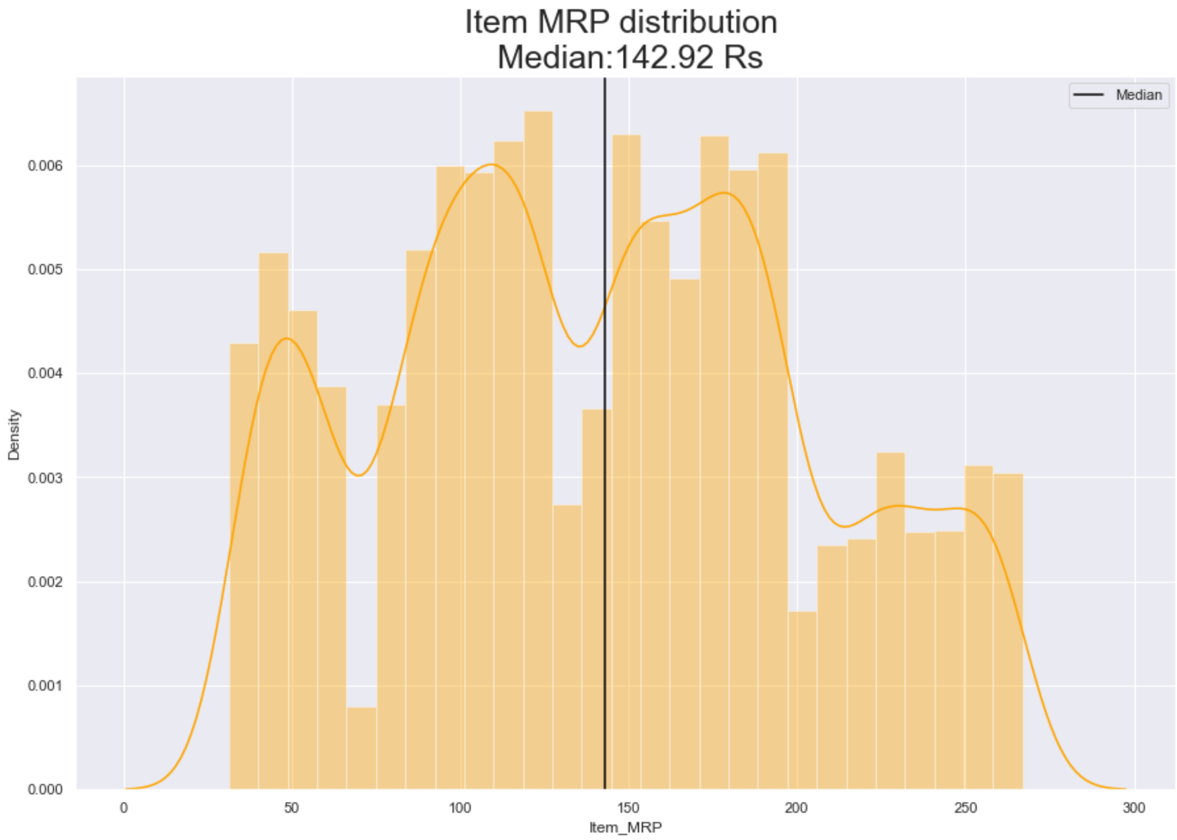
```
df_type=df_train.groupby('Item_Type')['Count'].sum().reset_index()
fig4=px.sunburst(df_train,path=['Item_Type','Item_Fat_Content'],names='Item_Type',color_con
fig4.update_layout(title='Item types',title_x=0.2,title_y=0.8,
                  annotations=[dict(showarrow=True,height=1000,width=900)],margin=dict(l=20
fig4.show()

fig5=px.pie(df_type,values='Count',names='Item_Type')
fig5.update_layout(title='Item distribution',title_x=0.1,title_y=0.8)
fig5.update_traces(textfont_size=15,textinfo='percent')
fig5.show()
```

5. Item Type

In [ ]:

```
plt.figure(figsize=(15,10))
sns.distplot(df_train['Item_MRP'],color='orange')
plt.title('Item MRP distribution \n Median:{0:.2f} Rs'.format(df_train['Item_MRP'].median()
plt.axvline(df_train['Item_MRP'].median(),color='black',label='Median')
plt.legend()
plt.show()
```



As we can see, we don't have any clear distribution of the prices here. The distribution is multi modal in nature with mulitple peaks.

The graph basically:

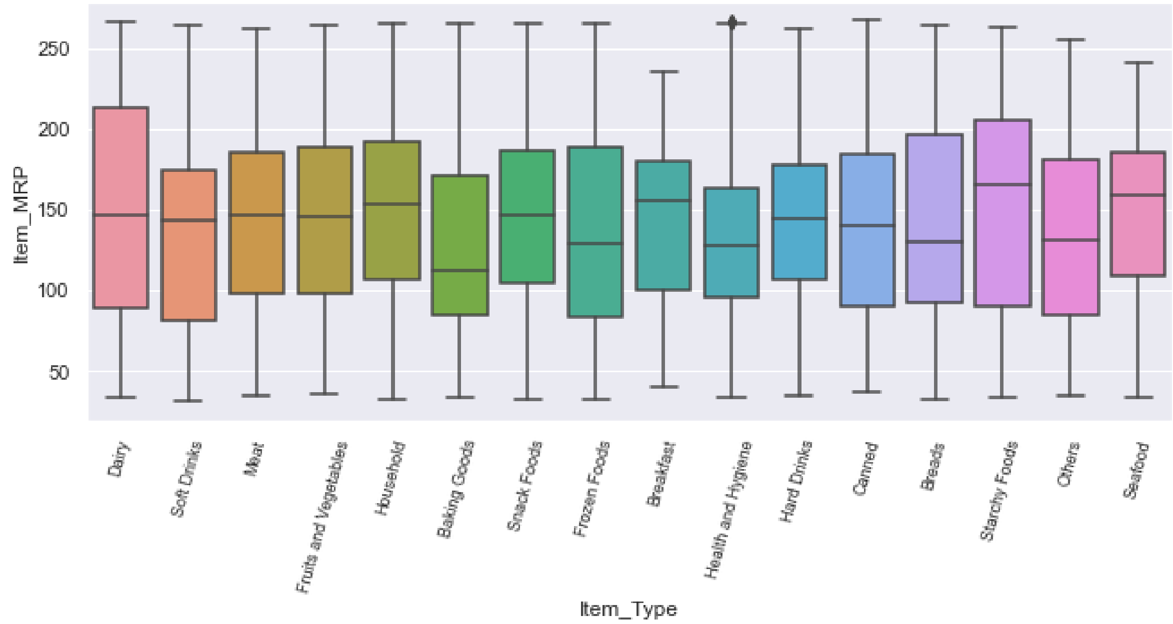
we have fair number of products whose prices range from 25-75 Rs. we have fair number of products in the 80-120 Rs range. Infact, it is the highest. the products increase again from 150-200 Rs range. There are fair number of products from 220-240 Rs range aswell.

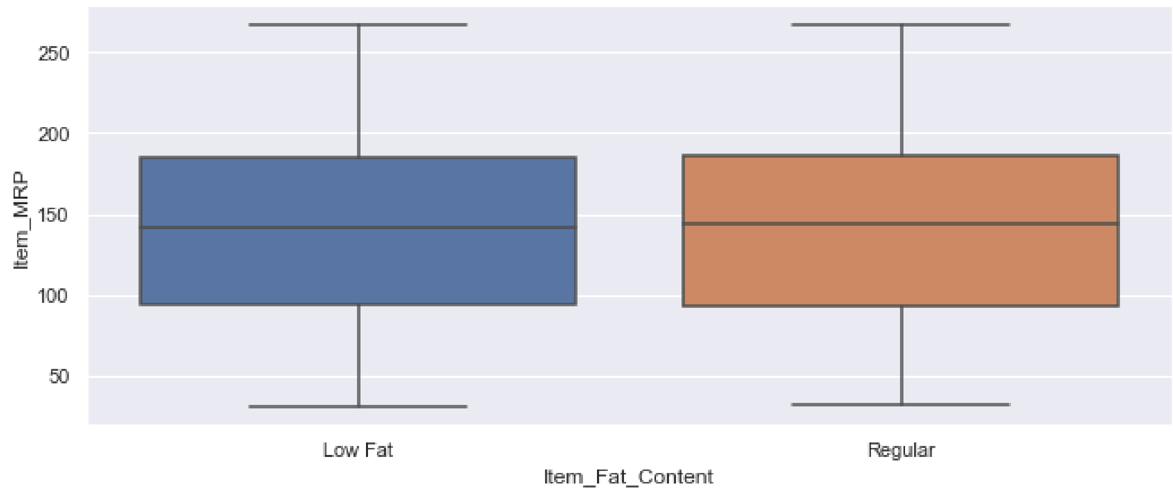
In [ ]:

```
labels=df_train['Item_Type'].unique()
fig6=plt.figure(figsize=(10,10))
ax1=fig6.add_subplot(211)
sns.boxplot(x='Item_Type',y='Item_MRP',data=df_train,ax=ax1)
ax1.set_xticklabels(labels, rotation=75,size=9)

ax2=fig6.add_subplot(212)
sns.boxplot(x='Item_Fat_Content',y='Item_MRP',data=df_train,ax=ax2)

fig6.tight_layout(pad=3)
plt.show()
```





From the above plot, we see which item types have high MRPs. Dairy product and Starchy foods have a higher median price than the rest.

Both low and regular food have almost identical median price.

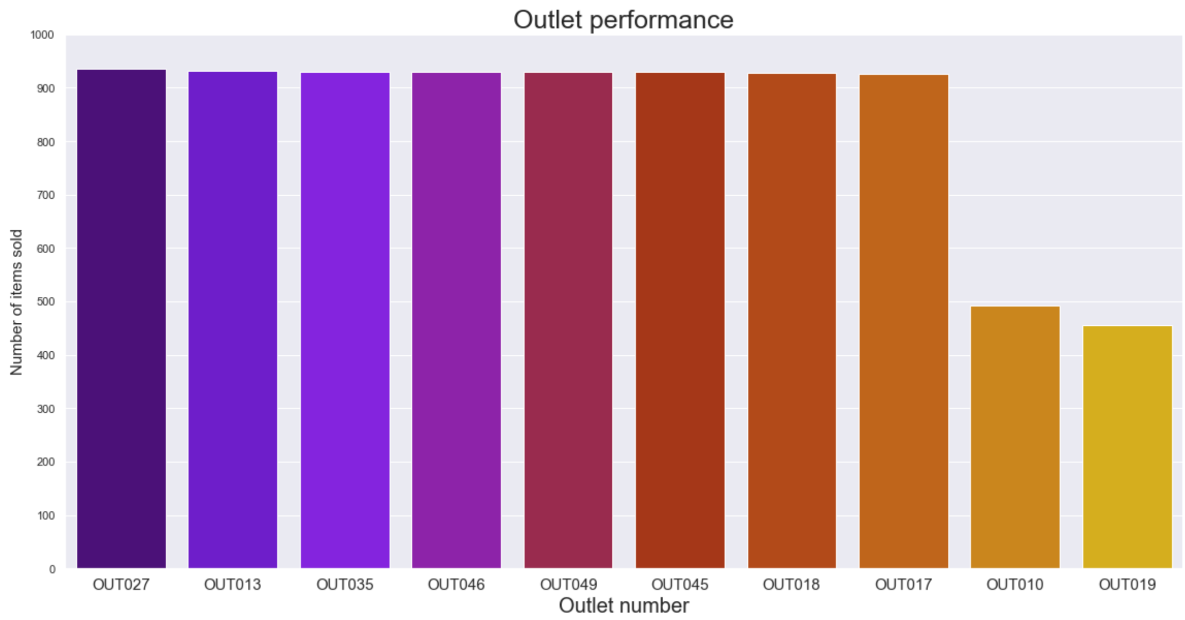
6. Outlet Identifier

In [ ]:

```
df_outlets=df_train.groupby('Outlet_Identifier')['Count'].sum().reset_index().sort_values(b
```

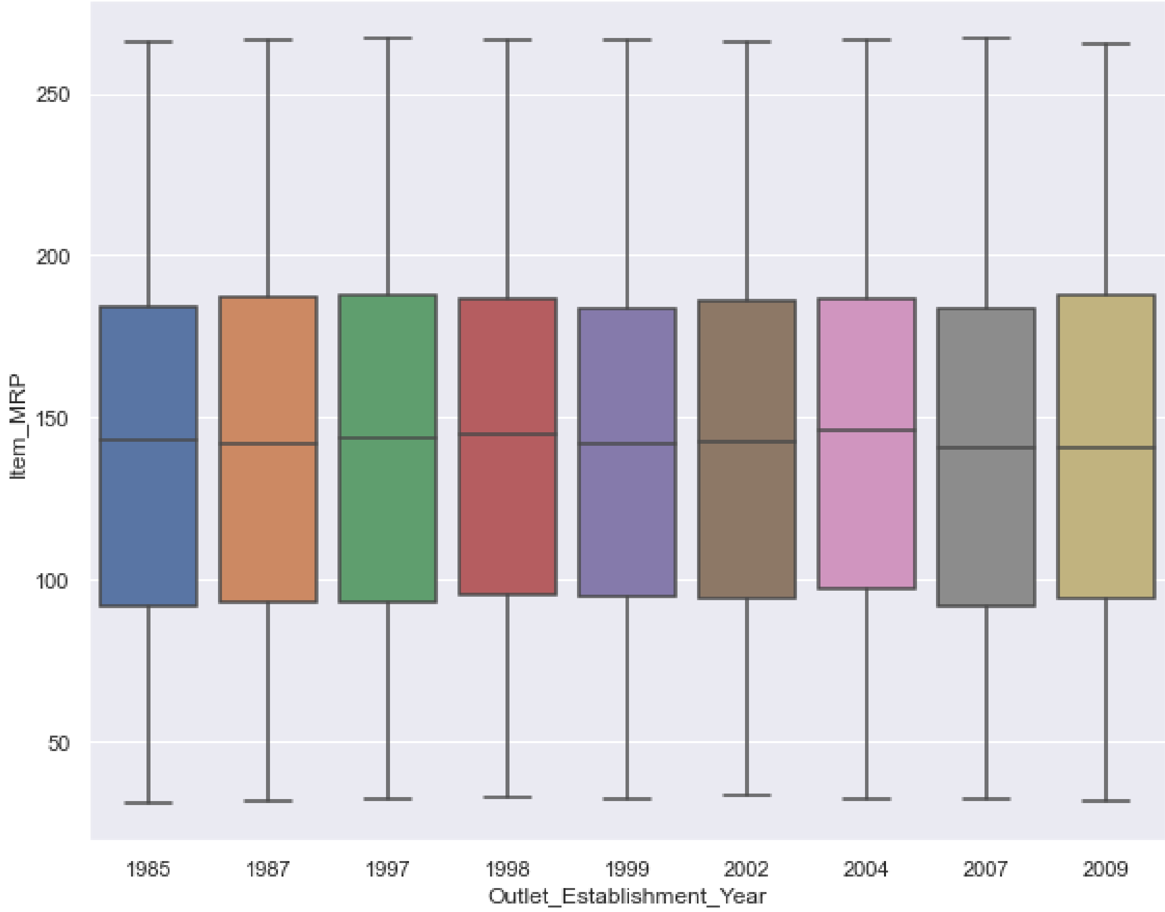
In [ ]:

```
sns.catplot('Outlet_Identifier','Count',data=df_outlets,aspect=2,height=8,kind='bar',palett
plt.xticks(size=15)
plt.ylabel('Number of items sold',size=15)
plt.xlabel('Outlet number',size=20)
plt.title('Outlet performance',size=25)
plt.yticks(np.arange(0,1100,100))
plt.show()
```



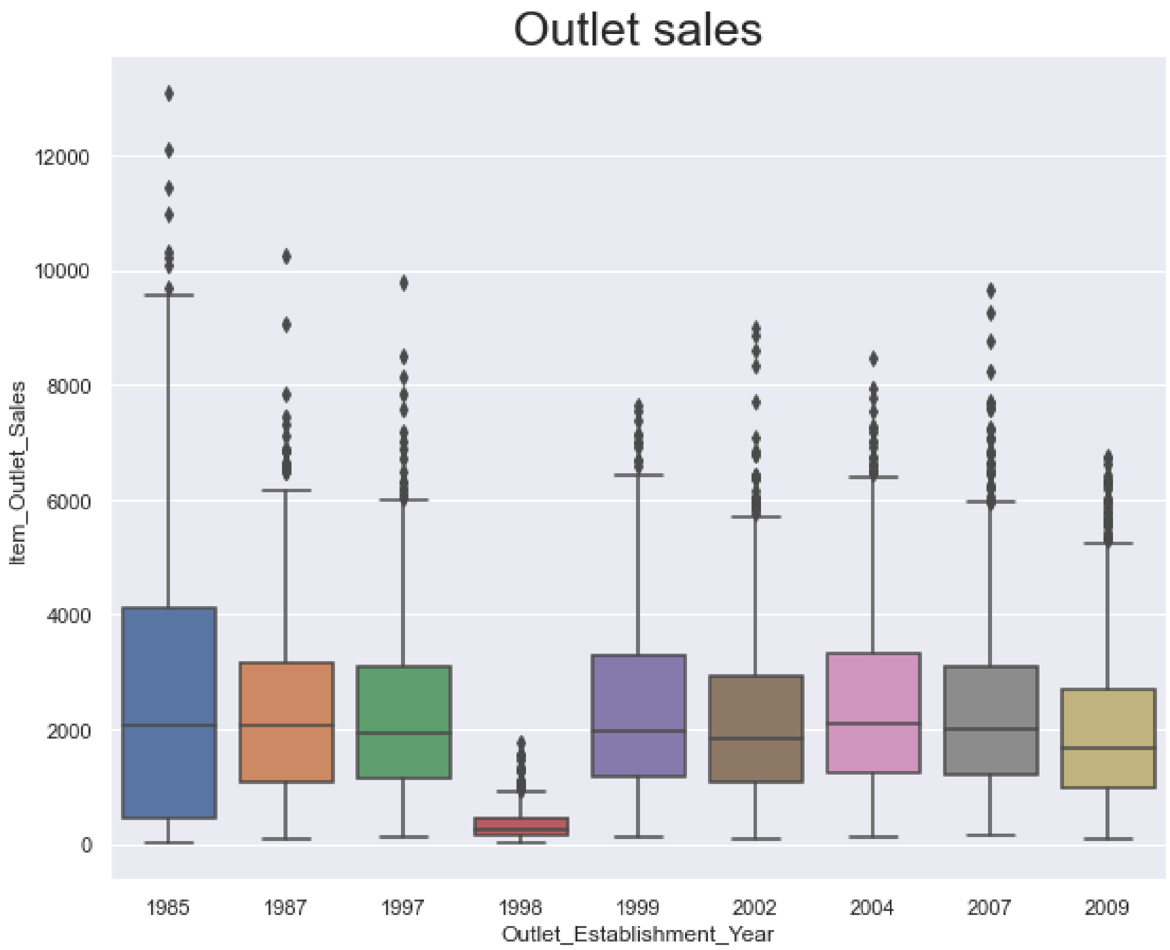
In [ ]:

```
plt.figure(figsize=(10,8))
sns.boxplot('Outlet_Establishment_Year','Item_MRP',data=df_train)
plt.show()
```



In [ ]:

```
plt.figure(figsize=(10,8))
sns.boxplot('Outlet_Establishment_Year','Item_Outlet_Sales',data=df_train)
plt.title('Outlet sales',size=25)
plt.show()
```



7. Outlet size

In [ ]:

```
df_train['Outlet_Size'].isna().value_counts()
```

Out[27]:

```
False    6041
True     2348
Name: Outlet_Size, dtype: int64
```

In [ ]:

```
df_size=df_train.groupby('Outlet_Size')['Count'].sum().reset_index()
fig7=px.pie(df_size,values='Count',names='Outlet_Size',hole=0.4)
fig7.update_layout(title='Store sizes',title_x=0.5,annotations=[dict(text='Fat',font_size=1
fig7.update_traces(textfont_size=15,textinfo='percent+label')
fig7.show()
```

In [ ]:

```
df_size_sales=df_train.groupby('Outlet_Size')[['Item_MRP','Item_Outlet_Sales']].mean().rese
df_size_sales
```

Out[29]:

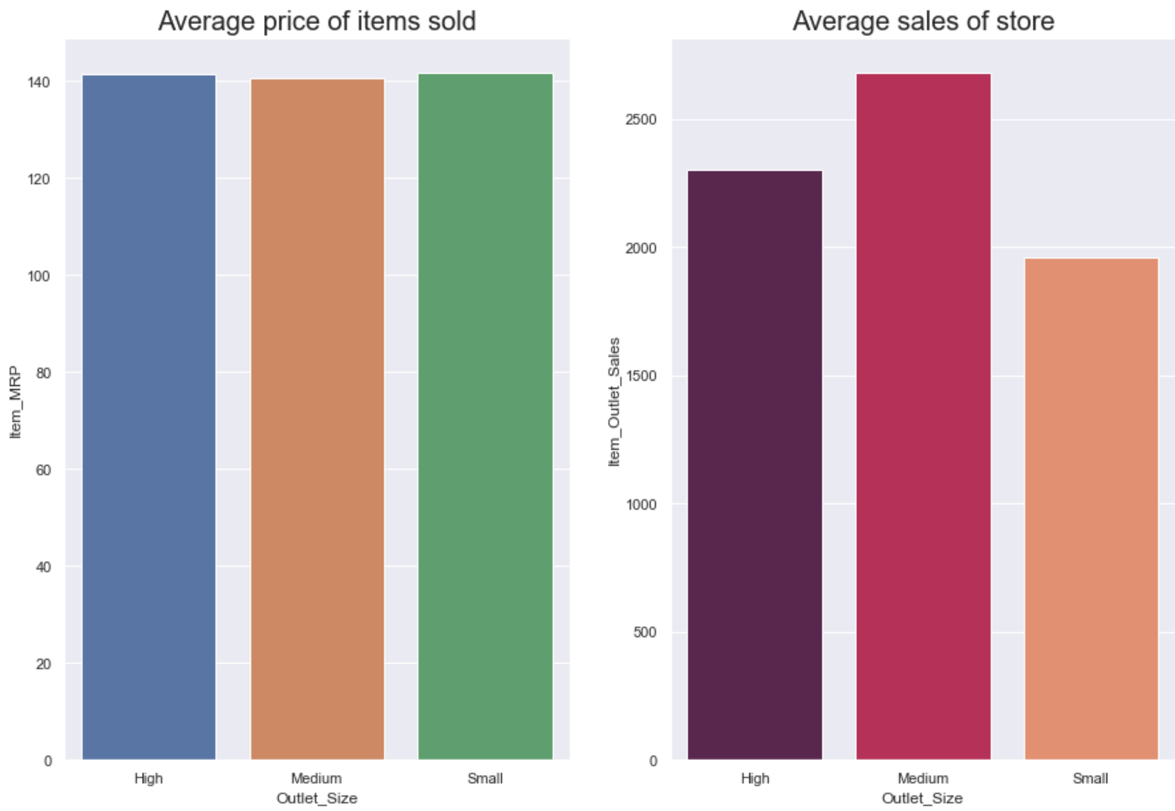
	Outlet_Size	Item_MRP	Item_Outlet_Sales
0	High	141.425982	2298.995256
1	Medium	140.590514	2681.603542
2	Small	141.756737	1960.412740

In [ ]:

```
fig8=plt.figure(figsize=(15,10))
ax1=fig8.add_subplot(121)
sns.barplot('Outlet_Size','Item_MRP',data=df_size_sales,ax=ax1)

ax2=fig8.add_subplot(122)
sns.barplot('Outlet_Size','Item_Outlet_Sales',data=df_size_sales,ax=ax2,palette='rocket')

ax1.set_title('Average price of items sold',size=20)
ax2.set_title('Average sales of store',size=20)
plt.show()
```



The average price of items sold in each outlet store size is nearly the same which is Rs 140. However, The medium stores seem to sell better followed by high sized and then small sized stores.

8. Outlet and Outlet location

In [ ]:

```
df_train.head()
```

Out[31]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identif
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT0
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT0
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT0
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT0
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT0

In [ ]:

```
fig9=px.sunburst(df_train,path=['Outlet_Type','Outlet_Location_Type'],color_continuous_scal
fig9.update_layout(title='Store type with location type',title_x=0.5)
fig9.show()
```

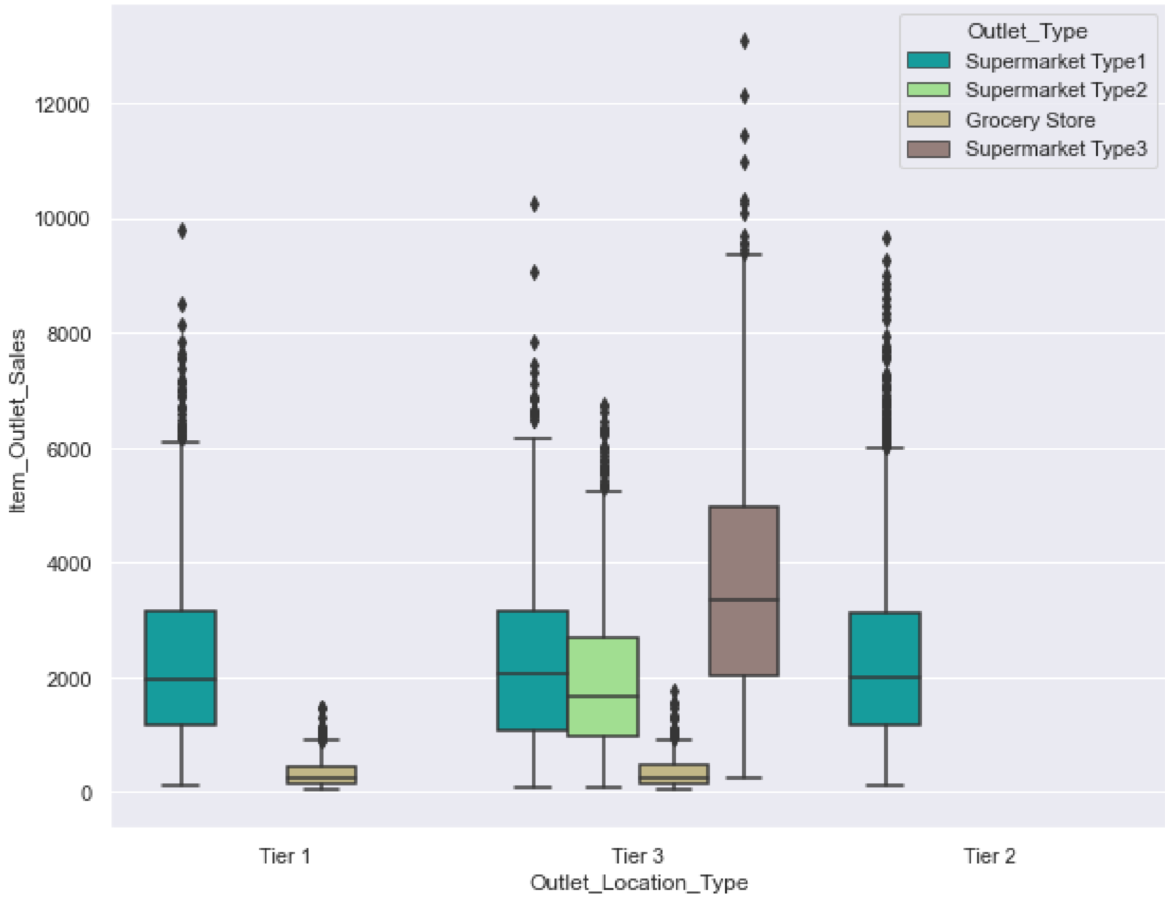
As we can see, majoirty of the stores are of type 1 supermarket distributed over various location tiers.

Supermarket type 2 and 3 are confined to only tier 3 locations. Very small section of the stores are actually grocery stores.

Let us check how do these stores sell based on location tier using a boxplot.

In [ ]:

```
plt.figure(figsize=(10,8))
sns.boxplot(y='Item_Outlet_Sales',hue='Outlet_Type',x='Outlet_Location_Type',data=df_train,
plt.show()
```



As we can see, tier 3 locations seem to be selling better than both tier 2 and tier 1. It is also to be noted that tier 3 has more number of stores in it. Hence, the sales are better too.

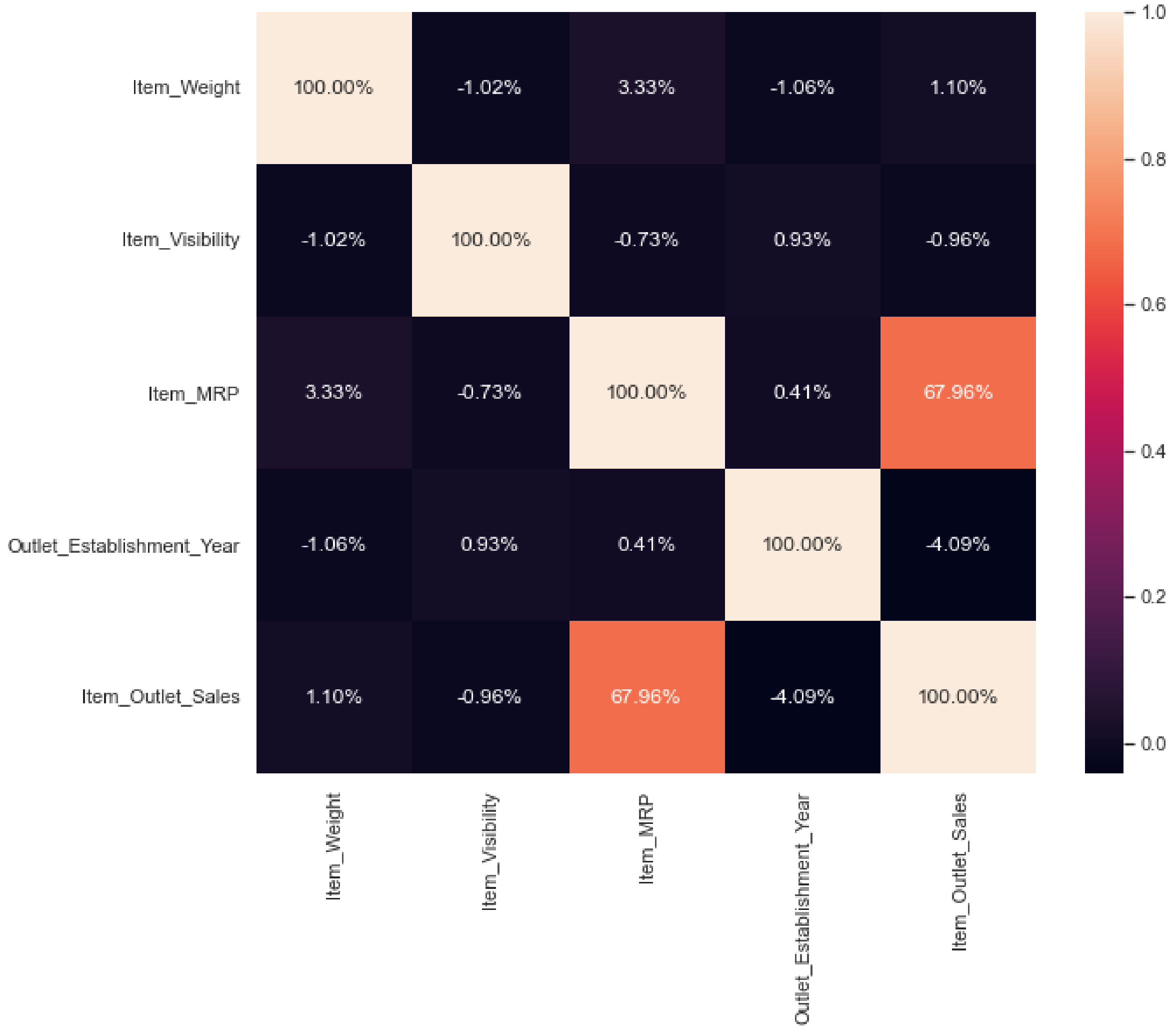
Correlation heatmap

In [ ]:

```
df_train.drop('Count',axis=1,inplace=True)
```

In [ ]:

```
corrs=df_train.dropna().corr()
plt.figure(figsize=(10,8))
sns.heatmap(corrs,annot=True,fmt='.2%')
plt.show()
```



From the above, we can see that correlation of Item_Weight is extremely low. Hence, we can simply drop this column and get done with the issues of null values. We shall similarly remove the order_size as there is no way to deal with the null values here aswell. We would also get rid of the item_identifier and outlet_indetifier since it is of no consequence to us.

In [ ]:

```
unn_cols=['Item_Weight','Outlet_Size','Item_Identifier','Outlet_Identifier']

for cols in unn_cols:
    df_train.drop(cols,axis=1,inplace=True)
```

Data Wrangling

In [ ]:

```
df_train['Item_Fat_Content'].replace('Low Fat',1,inplace=True)
df_train['Item_Fat_Content'].replace('Regular',0,inplace=True)
```

In [ ]:

```
df_dummies_type=pd.get_dummies(df_train['Item_Type'])
```

In [ ]:

```
df_train=df_train.merge(df_dummies_type,on=df_train.index)
```

In [ ]:

```
df_train.drop('key_0',axis=1,inplace=True)
df_train.drop('Item_Type',axis=1,inplace=True)
```

In [ ]:

```
df_train['Outlet_Location_Type'].replace('Tier 1',1,inplace=True)
df_train['Outlet_Location_Type'].replace('Tier 2',2,inplace=True)
df_train['Outlet_Location_Type'].replace('Tier 3',3,inplace=True)
```

In [ ]:

```
df_dummies_outlet=pd.get_dummies(df_train['Outlet_Type'])
df_train=df_train.merge(df_dummies_outlet,on=df_train.index)
```

Item Outlet Sales

In [ ]:

```
targets=df_train['Item_Outlet_Sales']
df_train.drop('Item_Outlet_Sales',axis=1,inplace=True)
df_train.head()
```

Out[43]:

	key_0	Item_Fat_Content	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Outlet_Location_Typ
0	0	1	0.016047	249.8092	1999	
1	1	0	0.019278	48.2692	2009	
2	2	1	0.016760	141.6180	1999	
3	3	0	0.000000	182.0950	1998	
4	4	1	0.000000	53.8614	1987	

5 rows × 27 columns

In [ ]:

```
```