In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import os
import warnings
warnings.filterwarnings('ignore')
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 15, 6
```

In [3]:
```
display (os.getcwd())
```

```
'C:\\Users\\spars'
```

In [4]:
```
os.chdir ('C:\\Users\\spars\\Desktop\\Acmegrade Data Science August Files\\DS Day21 4
display (os.getcwd())
```

```
'C:\\Users\\spars\\Desktop\\Acmegrade Data Science August Files\\DS Day21 4\\PRJ Sal
es Forecasting'
```

In [5]:
```
dt = pd.read_csv('Train.csv')
```

In [6]:
```
display (dt.head(10))
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Id
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	
5	FDP36	10.395	Regular	0.000000	Baking Goods	51.4008	
6	FDO10	13.650	Regular	0.012741	Snack Foods	57.6588	
7	FDP10	NaN	Low Fat	0.127470	Snack Foods	107.7622	
8	FDH17	16.200	Regular	0.016687	Frozen Foods	96.9726	
9	FDU28	19.200	Regular	0.094450	Frozen Foods	187.8214	

In [7]:
```
print(dt.shape)
```

```
(8523, 12)
```

In [8]:
```
display(dt.columns)
```

```
Index(['Item_Identifier', 'Item_Weight', 'Item_Fat_Content', 'Item_Visibility',
       'Item_Type', 'Item_MRP', 'Outlet_Identifier',
       'Outlet_Establishment_Year', 'Outlet_Size', 'Outlet_Location_Type',
       'Outlet_Type', 'Item_Outlet_Sales'],
      dtype='object')
```

In [9]:
```
display(dt.describe())
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	7060.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.857645	0.066132	140.992782	1997.831867	2181.288914
std	4.643456	0.051598	62.275067	8.371760	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	8.773750	0.026989	93.826500	1987.000000	834.247400
50%	12.600000	0.053931	143.012800	1999.000000	1794.331000
75%	16.850000	0.094585	185.643700	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

In [10]:
```
display(dt.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
None
```

In [11]:
```
display(dt.apply(lambda x: len(x.unique())))
```

```
Item_Identifier              1559
Item_Weight                   416
Item_Fat_Content                5
Item_Visibility              7880
Item_Type                      16
Item_MRP                     5938
Outlet_Identifier              10
Outlet_Establishment_Year       9
Outlet_Size                     4
Outlet_Location_Type            3
Outlet_Type                     4
Item_Outlet_Sales            3493
dtype: int64
```

In [12]:
```
display (dt.isnull().sum())
```

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

In [13]:
```
cat_col = []
for x in dt.dtypes.index:
    if dt.dtypes[x] == 'object':
        cat_col.append(x)
display (cat_col)
```

```
['Item_Identifier',
 'Item_Fat_Content',
 'Item_Type',
 'Outlet_Identifier',
 'Outlet_Size',
 'Outlet_Location_Type',
 'Outlet_Type']
```

In [14]:
```
cat_col.remove('Item_Identifier')
cat_col.remove('Outlet_Identifier')
display (cat_col)
```

```
['Item_Fat_Content',
 'Item_Type',
 'Outlet_Size',
 'Outlet_Location_Type',
 'Outlet_Type']
```

In [15]:
```
for col in cat_col:
    print(col , len(dt[col].unique()))
```

```
Item_Fat_Content 5
Item_Type 16
Outlet_Size 4
Outlet_Location_Type 3
Outlet_Type 4
```

In [16]:
```
for col in cat_col:
    print(col)
    print(dt[col].value_counts())
    print()
    print ('*' *50)
```

```
Item_Fat_Content
Low Fat    5089
Regular    2889
LF          316
reg         117
low fat     112
Name: Item_Fat_Content, dtype: int64

**************************************************
Item_Type
Fruits and Vegetables    1232
Snack Foods              1200
Household                 910
Frozen Foods              856
Dairy                     682
Canned                    649
Baking Goods              648
Health and Hygiene        520
Soft Drinks               445
Meat                      425
Breads                    251
Hard Drinks               214
Others                    169
Starchy Foods             148
Breakfast                 110
Seafood                    64
Name: Item_Type, dtype: int64

**************************************************
Outlet_Size
Medium    2793
Small     2388
High       932
Name: Outlet_Size, dtype: int64

**************************************************
Outlet_Location_Type
Tier 3    3350
Tier 2    2785
Tier 1    2388
Name: Outlet_Location_Type, dtype: int64

**************************************************
Outlet_Type
Supermarket Type1    5577
Grocery Store        1083
Supermarket Type3     935
Supermarket Type2     928
Name: Outlet_Type, dtype: int64

**************************************************
```

In [17]:
```
miss_bool = dt['Item_Weight'].isnull()
display (miss_bool)
```

```
0       False
1       False
2       False
3       False
4       False
        ...
8518    False
8519    False
8520    False
8521    False
8522    False
Name: Item_Weight, Length: 8523, dtype: bool
```

In [18]:
```
miss_bool.head(20)
```

Out[18]:
```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7      True
8     False
9     False
10    False
11    False
12    False
13    False
14    False
15    False
16    False
17    False
18     True
19    False
Name: Item_Weight, dtype: bool
```

In [19]:
```
display (dt['Item_Weight'].isnull().sum())
```

```
1463
```

In [20]:
```
Item_Weight_null = dt[dt['Item_Weight'].isna()]
display (Item_Weight_null)
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outle
7	FDP10	NaN	Low Fat	0.127470	Snack Foods	107.7622	
18	DRI11	NaN	Low Fat	0.034238	Hard Drinks	113.2834	
21	FDW12	NaN	Regular	0.035400	Baking Goods	144.5444	
23	FDC37	NaN	Low Fat	0.057557	Baking Goods	107.6938	
29	FDC14	NaN	Regular	0.072222	Canned	43.6454	
...	...	...	...	...	...	...	
8485	DRK37	NaN	Low Fat	0.043792	Soft Drinks	189.0530	
8487	DRG13	NaN	Low Fat	0.037006	Soft Drinks	164.7526	
8488	NCN14	NaN	Low Fat	0.091473	Others	184.6608	
8490	FDU44	NaN	Regular	0.102296	Fruits and Vegetables	162.3552	
8504	NCN18	NaN	Low Fat	0.124111	Household	111.7544	

1463 rows × 12 columns

In [21]:
```
Item_Weight_null['Item_Identifier'].value_counts()
```

Out[21]:
```
FDK08    2
FDA08    2
FDV23    2
FDY56    2
FDI04    2
        ..
FDM44    1
FDZ48    1
FDK41    1
FDD57    1
NCN18    1
Name: Item_Identifier, Length: 1142, dtype: int64
```

In [22]:
```
item_weight_mean = dt.pivot_table(values = "Item_Weight", index = 'Item_Identifier')
display (item_weight_mean)
```

Item_Identifier	Item_Weight
DRA12	11.600
DRA24	19.350
DRA59	8.270
DRB01	7.390
DRB13	6.115
...	...
NCZ30	6.590
NCZ41	19.850
NCZ42	10.500
NCZ53	9.600
NCZ54	14.650

1555 rows × 1 columns

In [23]:
```
display (dt['Item_Identifier'])
```

```
0       FDA15
1       DRC01
2       FDN15
3       FDX07
4       NCD19
        ...
8518    FDF22
8519    FDS36
8520    NCJ29
8521    FDN46
8522    DRG01
Name: Item_Identifier, Length: 8523, dtype: object
```

In [24]:
```
for i, item in enumerate(dt['Item_Identifier']):
    if miss_bool[i]:
        if item in item_weight_mean:
            dt['Item_Weight'][i] = item_weight_mean.loc[item]['Item_Weight']
        else:
            dt['Item_Weight'][i] = np.mean(dt['Item_Weight'])
```

In [25]:
```
display (dt['Item_Weight'].isnull().sum())
```

```
0
```

In [26]:
```
dt.groupby('Outlet_Size').agg({'Outlet_Size': np.size})
```

Out[26]:

Outlet_Size	Outlet_Size
High	932
Medium	2793
Small	2388

In [27]:
```
display (dt['Outlet_Size'].isnull().sum())
```

```
2410
```

In [28]:
```
Outlet_Size_null= dt[dt['Outlet_Size'].isna()]
display (Outlet_Size_null)
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outle
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	
8	FDH17	16.200	Regular	0.016687	Frozen Foods	96.9726	
9	FDU28	19.200	Regular	0.094450	Frozen Foods	187.8214	
25	NCD06	13.000	Low Fat	0.099887	Household	45.9060	
28	FDE51	5.925	Regular	0.161467	Dairy	45.5086	
...	...	...	...	...	...	...	
8502	NCH43	8.420	Low Fat	0.070712	Household	216.4192	
8508	FDW31	11.350	Regular	0.043246	Fruits and Vegetables	199.4742	
8509	FDG45	8.100	Low Fat	0.214306	Fruits and Vegetables	213.9902	
8514	FDA01	15.000	Regular	0.054489	Canned	57.5904	
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570	

2410 rows × 12 columns

In [29]:
```
Outlet_Size_null['Outlet_Type'].value_counts()
```

Out[29]:
```
Supermarket Type1    1855
Grocery Store         555
Name: Outlet_Type, dtype: int64
```

In [30]:
```
dt.groupby (['Outlet_Type','Outlet_Size'] ).agg({'Outlet_Type':[np.size]})
```

Out[30]:

		Outlet_Type
		size
Outlet_Type	**Outlet_Size**	
Grocery Store	**Small**	528
Supermarket Type1	**High**	932
	Medium	930
	Small	1860
Supermarket Type2	**Medium**	928
Supermarket Type3	**Medium**	935

In [31]:
```
outlet_size_mode = dt.pivot_table(values='Outlet_Size', columns='Outlet_Type', aggfu
display (outlet_size_mode)
```

Outlet_Type	Grocery Store	Supermarket Type1	Supermarket Type2	Supermarket Type3
Outlet_Size	Small	Small	Medium	Medium

In [32]:
```
miss_bool = dt['Outlet_Size'].isnull()
dt.loc[miss_bool, 'Outlet_Size'] = dt.loc[miss_bool, 'Outlet_Type'].apply(lambda x: 
```

In [33]:
```
display (dt['Outlet_Size'].isnull().sum())
```

```
0
```

In [34]:
```
dt.groupby (['Outlet_Type','Outlet_Size'] ).agg({'Outlet_Type':[np.size]})
```

Out[34]:

		Outlet_Type
		size
Outlet_Type	**Outlet_Size**	
Grocery Store	**Small**	1083
Supermarket Type1	**High**	932
	Medium	930
	Small	3715
Supermarket Type2	**Medium**	928
Supermarket Type3	**Medium**	935

In [35]:
```
display (sum(dt['Item_Visibility']==0))
```

```
526
```

In [36]:
```
dt.loc[:, 'Item_Visibility'].replace([0], [dt['Item_Visibility'].mean()], inplace=Tr
```

In [37]:
```
sum(dt['Item_Visibility']==0)
```

Out[37]:
```
0
```

In [38]:
```
dt['Item_Fat_Content'].value_counts()
```

Out[38]:
```
Low Fat    5089
Regular    2889
LF          316
reg         117
low fat     112
Name: Item_Fat_Content, dtype: int64
```

In [39]:
```
dt['Item_Fat_Content'] = dt['Item_Fat_Content'].replace({'LF':'Low Fat', 'reg':'Regu
display (dt['Item_Fat_Content'].value_counts())
```

```
Low Fat    5517
Regular    3006
Name: Item_Fat_Content, dtype: int64
```

In [40]:
```
display (dt.isnull().sum())
```

```
Item_Identifier              0
Item_Weight                  0
Item_Fat_Content             0
Item_Visibility              0
Item_Type                    0
Item_MRP                     0
Outlet_Identifier            0
Outlet_Establishment_Year    0
Outlet_Size                  0
Outlet_Location_Type         0
Outlet_Type                  0
Item_Outlet_Sales            0
dtype: int64
```

In [41]:
```
display(dt.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                8523 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
None
```

In [42]:
```
dt['New_Item_Type'] = dt['Item_Identifier'].apply(lambda x: x[:2])
display (dt['New_Item_Type'])
```

```
0       FD
1       DR
2       FD
3       FD
4       NC
        ..
8518    FD
8519    FD
8520    NC
8521    FD
8522    DR
Name: New_Item_Type, Length: 8523, dtype: object
```

In [43]:
```
display (dt['New_Item_Type'].value_counts())
```

```
FD    6125
NC    1599
DR     799
Name: New_Item_Type, dtype: int64
```

In [44]:
```
dt['New_Item_Type'] = dt['New_Item_Type'].map({'FD':'Food', 'NC':'Non-Consumable', 'I
display (dt['New_Item_Type'].value_counts())
```

```
Food              6125
Non-Consumable    1599
Drinks             799
Name: New_Item_Type, dtype: int64
```

In [45]:
```
display (dt['Item_Fat_Content'].value_counts())
```

```
Low Fat    5517
Regular    3006
Name: Item_Fat_Content, dtype: int64
```

In [46]:
```
dt.groupby (['New_Item_Type','Item_Fat_Content'] ).agg({'Outlet_Type':[np.size]})
```

Out[46]:

		Outlet_Type
		size
New_Item_Type	Item_Fat_Content	
Drinks	Low Fat	728
	Regular	71
Food	Low Fat	3190
	Regular	2935
Non-Consumable	Low Fat	1599

In [47]:
```
dt.loc[dt['New_Item_Type']=='Non-Consumable', 'Item_Fat_Content'] = 'Non-Edible'
display (dt['Item_Fat_Content'].value_counts())
```

```
Low Fat       3918
Regular       3006
Non-Edible    1599
Name: Item_Fat_Content, dtype: int64
```

In [48]:
```
dt.groupby (['New_Item_Type','Item_Fat_Content'] ).agg({'Outlet_Type':[np.size]})
```

Out[48]:

		Outlet_Type
		size
New_Item_Type	**Item_Fat_Content**	
Drinks	**Low Fat**	728
	Regular	71
Food	**Low Fat**	3190
	Regular	2935
Non-Consumable	**Non-Edible**	1599

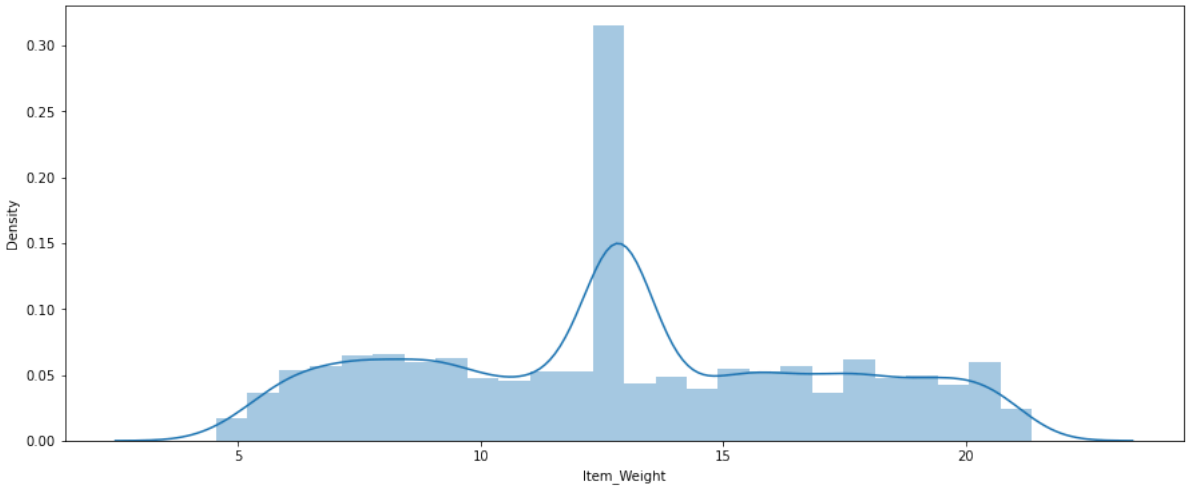
In [49]:
```
dt['Outlet_Years'] = 2022 - dt['Outlet_Establishment_Year']
print (dt['Outlet_Years'])
```

```
0       23
1       13
2       23
3       24
4       35
        ..
8518    35
8519    20
8520    18
8521    13
8522    25
Name: Outlet_Years, Length: 8523, dtype: int64
```

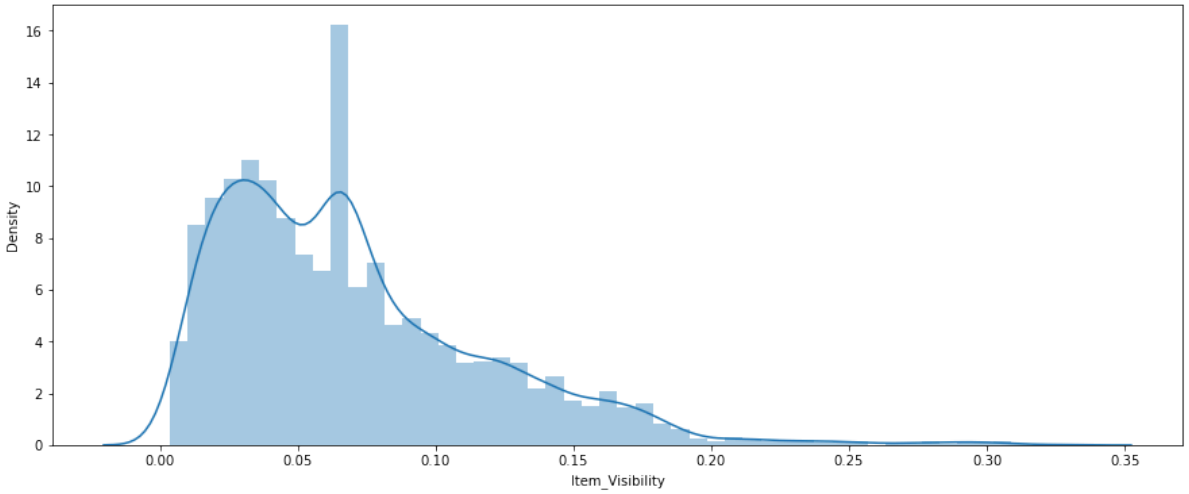
In [50]:
```
display (dt.head(10))
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Ic
0	FDA15	9.300000	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.920000	Regular	0.019278	Soft Drinks	48.2692	
2	FDN15	17.500000	Low Fat	0.016760	Meat	141.6180	
3	FDX07	19.200000	Regular	0.066132	Fruits and Vegetables	182.0950	
4	NCD19	8.930000	Non-Edible	0.066132	Household	53.8614	
5	FDP36	10.395000	Regular	0.066132	Baking Goods	51.4008	
6	FDO10	13.650000	Regular	0.012741	Snack Foods	57.6588	
7	FDP10	12.857645	Low Fat	0.127470	Snack Foods	107.7622	
8	FDH17	16.200000	Regular	0.016687	Frozen Foods	96.9726	
9	FDU28	19.200000	Regular	0.094450	Frozen Foods	187.8214	

In [51]:
```
sns.distplot(dt['Item_Weight'])
plt.show()
```



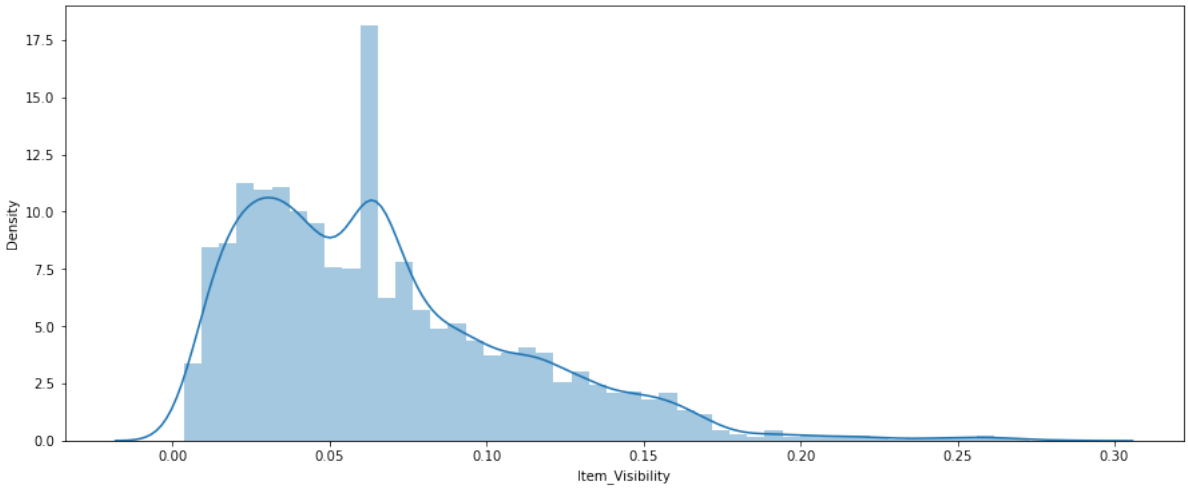
In [52]:
```
sns.distplot(dt['Item_Visibility'])
plt.show()
```



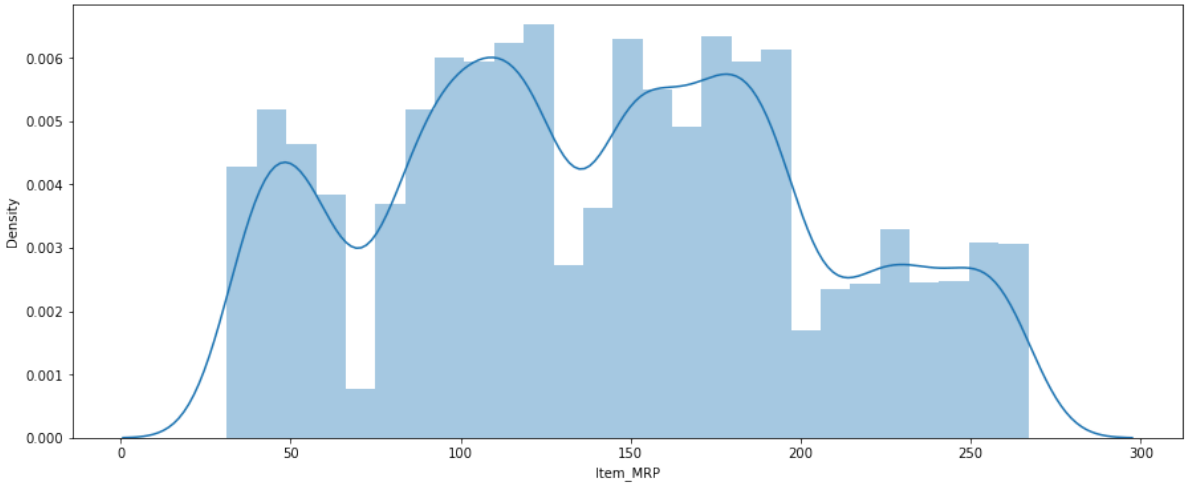
In [53]:
```
dt['Item_Visibility'] = np.log(1+dt['Item_Visibility'])
display (dt['Item_Visibility'])
```

```
0       0.015920
1       0.019095
2       0.016621
3       0.064037
4       0.064037
          ...
8518    0.055230
8519    0.045912
8520    0.034581
8521    0.135597
8522    0.043900
Name: Item_Visibility, Length: 8523, dtype: float64
```

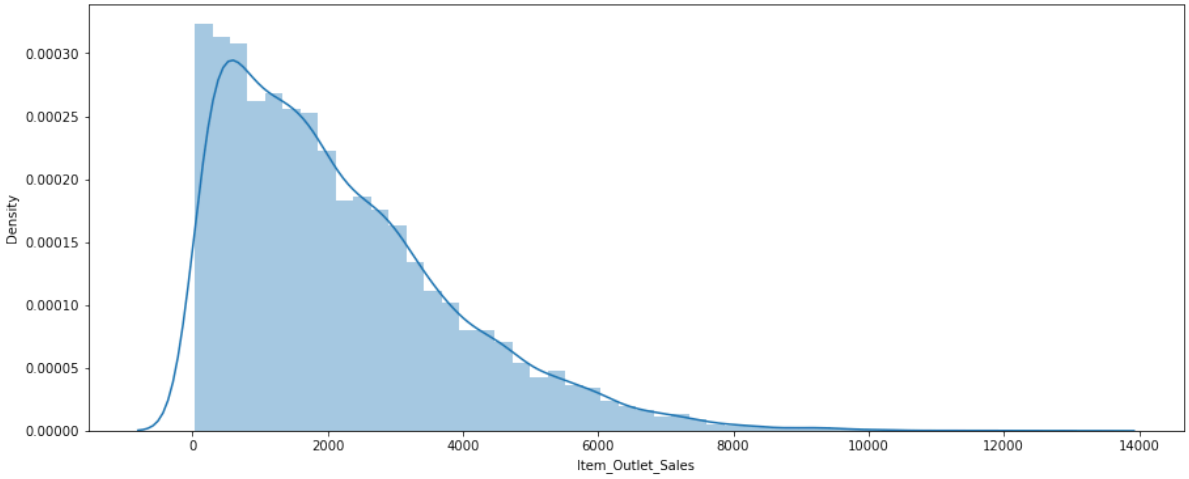
In [54]:
```
sns.distplot(dt['Item_Visibility'])
plt.show()
```



In [55]:
```
sns.distplot(dt['Item_MRP'])
plt.show()
```



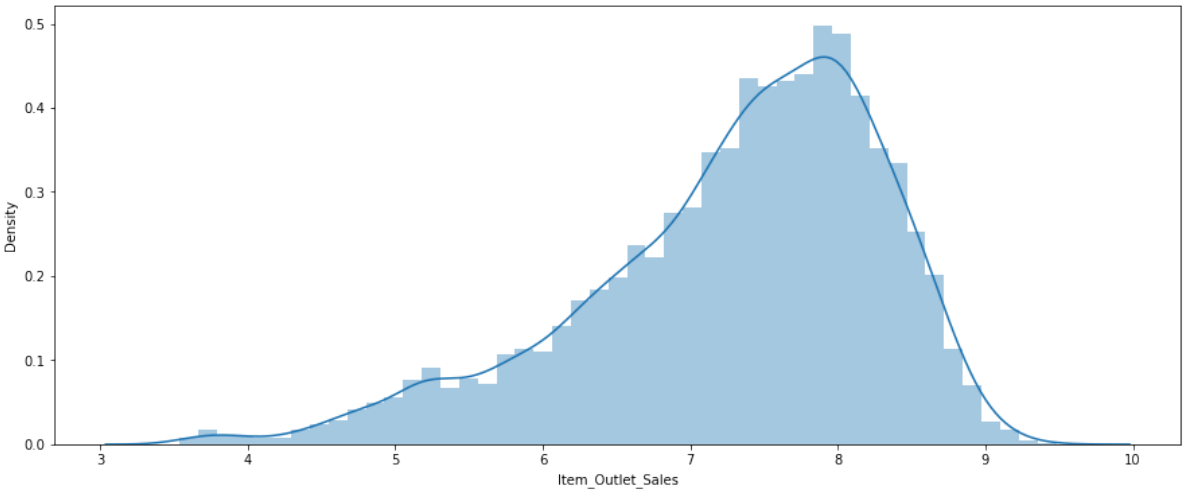
In [56]:
```
sns.distplot(dt['Item_Outlet_Sales'])
plt.show()
```



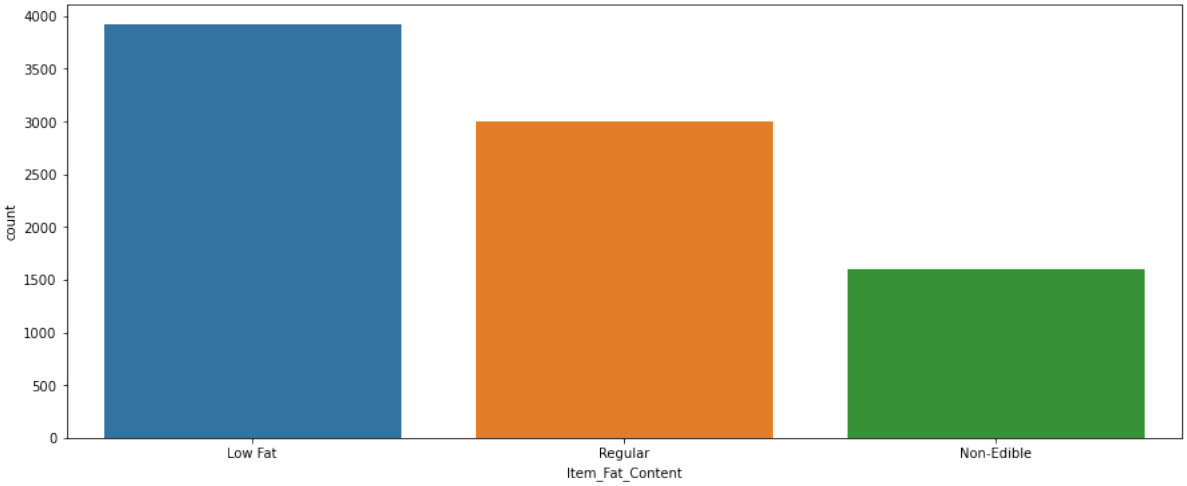
In [57]:
```
dt['Item_Outlet_Sales'] = np.log(1+dt['Item_Outlet_Sales'])
display (dt['Item_Outlet_Sales'])
```

```
0       8.225808
1       6.096776
2       7.648868
3       6.597664
4       6.903451
          ...
8518    7.929984
8519    6.310436
8520    7.085159
8521    7.521100
8522    6.642056
Name: Item_Outlet_Sales, Length: 8523, dtype: float64
```

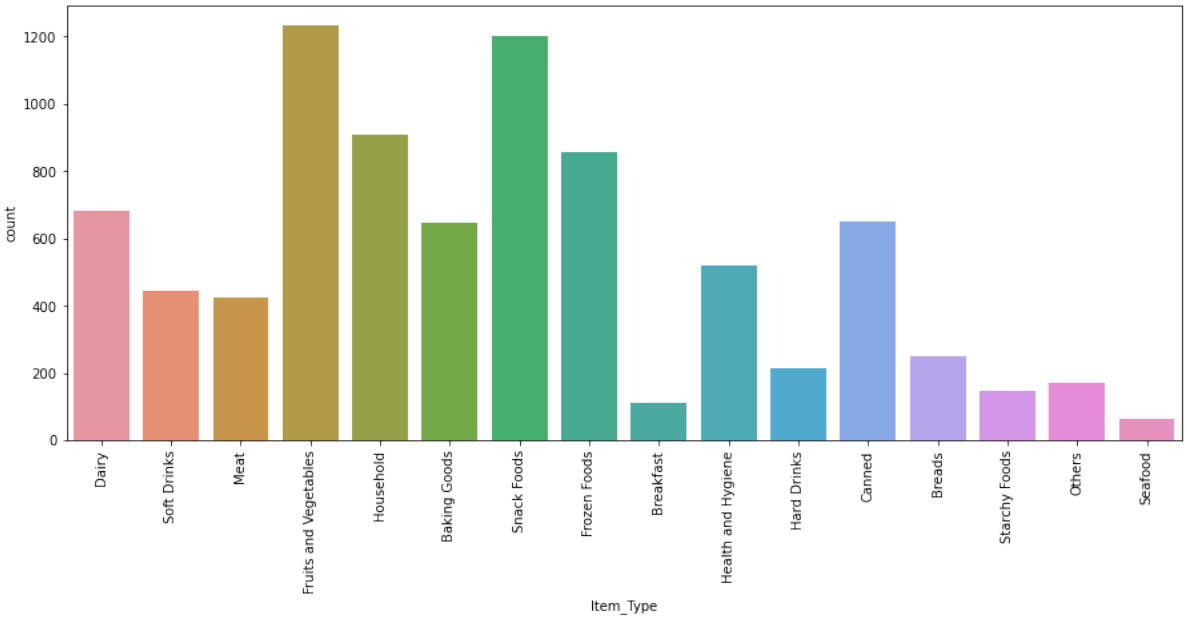
In [58]:
```
sns.distplot(dt['Item_Outlet_Sales'])
plt.show()
```



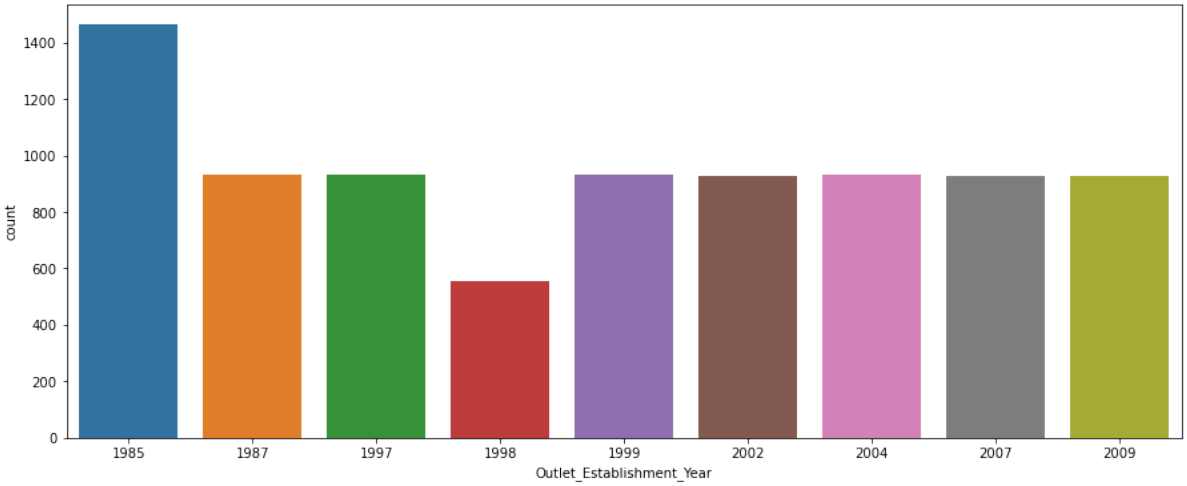
In [59]:
```
sns.countplot(dt["Item_Fat_Content"])
plt.show()
```



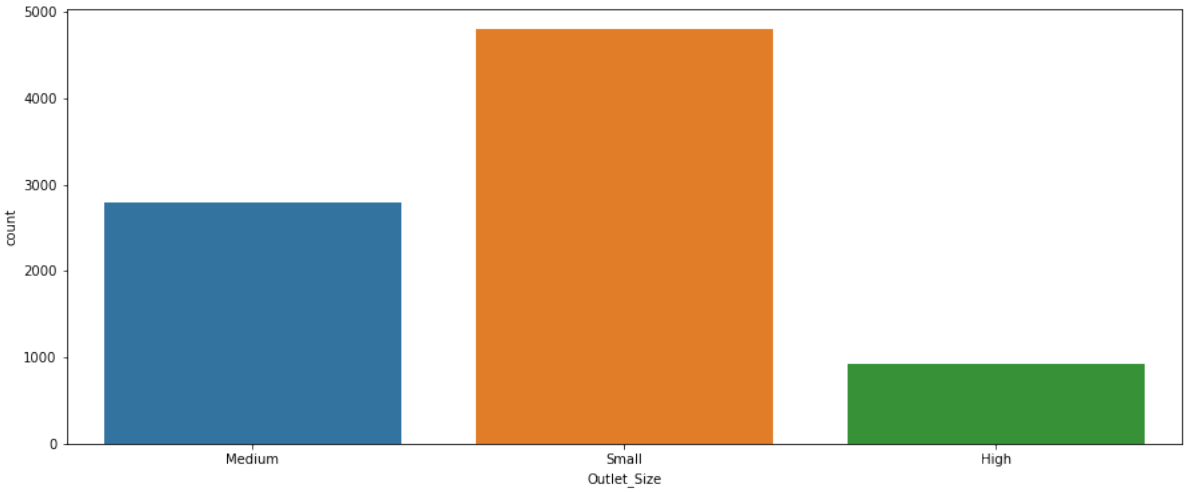
In [60]:
```
l = list(dt['Item_Type'].unique())
chart = sns.countplot(dt["Item_Type"])
chart.set_xticklabels(labels=l, rotation=90)
plt.show()
```



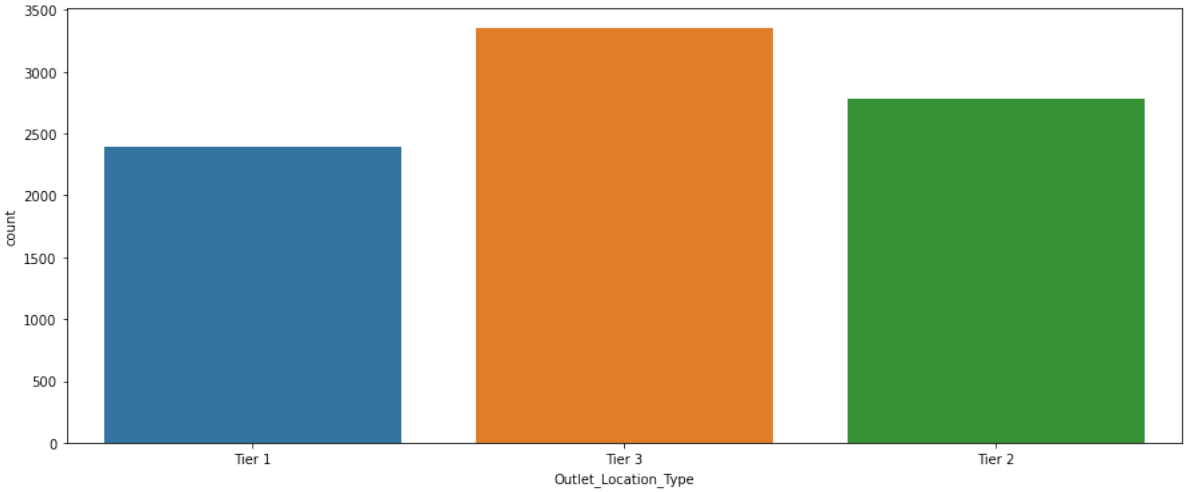
In [61]:
```
sns.countplot(dt['Outlet_Establishment_Year'])
plt.show()
```



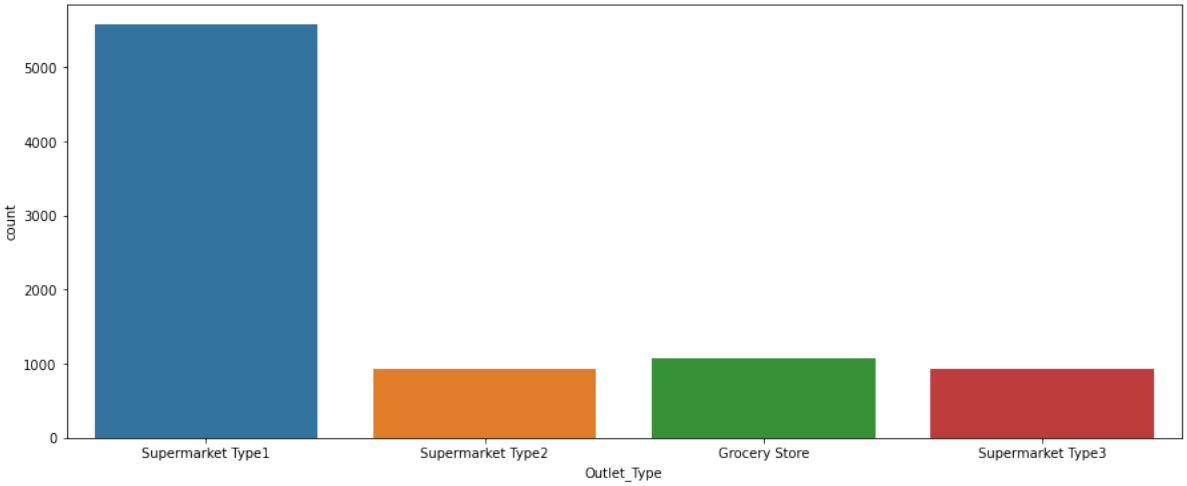
In [62]:
```
sns.countplot(dt['Outlet_Size'])
plt.show()
```



In [63]:
```
sns.countplot(dt['Outlet_Location_Type'])
plt.show()
```



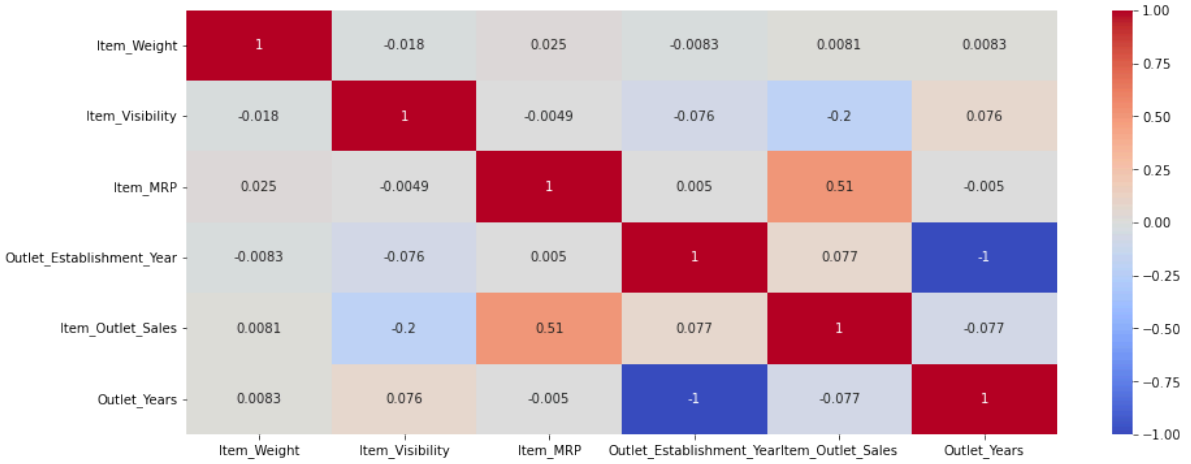
In [64]:
```
sns.countplot(dt['Outlet_Type'])
plt.show()
```



In [65]:
```
corr = dt.corr()
display (corr)
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_
Item_Weight	1.000000	-0.017807	0.024756	-0.008301	
Item_Visibility	-0.017807	1.000000	-0.004858	-0.076053	
Item_MRP	0.024756	-0.004858	1.000000	0.005020	
Outlet_Establishment_Year	-0.008301	-0.076053	0.005020	1.000000	
Item_Outlet_Sales	0.008059	-0.198589	0.509886	0.077032	
Outlet_Years	0.008301	0.076053	-0.005020	-1.000000	

In [66]:
```
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.show()
```



In [67]:
```
dt.head(10)
```

Out[67]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Ic
0	FDA15	9.300000	Low Fat	0.015920	Dairy	249.8092	
1	DRC01	5.920000	Regular	0.019095	Soft Drinks	48.2692	
2	FDN15	17.500000	Low Fat	0.016621	Meat	141.6180	
3	FDX07	19.200000	Regular	0.064037	Fruits and Vegetables	182.0950	
4	NCD19	8.930000	Non-Edible	0.064037	Household	53.8614	
5	FDP36	10.395000	Regular	0.064037	Baking Goods	51.4008	
6	FDO10	13.650000	Regular	0.012661	Snack Foods	57.6588	
7	FDP10	12.857645	Low Fat	0.119976	Snack Foods	107.7622	
8	FDH17	16.200000	Regular	0.016549	Frozen Foods	96.9726	
9	FDU28	19.200000	Regular	0.090252	Frozen Foods	187.8214	

In [68]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dt['Outlet'] = le.fit_transform(dt['Outlet_Identifier'])
display (dt['Outlet'])
```

```
0       9
1       3
2       9
3       0
4       1
       ..
8518    1
8519    7
8520    6
8521    3
8522    8
Name: Outlet, Length: 8523, dtype: int32
```

In [69]:
```
cat_col = ['Item_Fat_Content', 'Item_Type', 'Outlet_Size', 'Outlet_Location_Type', '
for col in cat_col:
    dt[col] = le.fit_transform(dt[col])
display (dt.head())
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Id
0	FDA15	9.30	0	0.015920	4	249.8092	
1	DRC01	5.92	2	0.019095	14	48.2692	
2	FDN15	17.50	0	0.016621	10	141.6180	
3	FDX07	19.20	2	0.064037	6	182.0950	
4	NCD19	8.93	1	0.064037	9	53.8614	

In [70]:
```
dt = pd.get_dummies(dt, columns=['Item_Fat_Content', 'Outlet_Size', 'Outlet_Location_
display (dt.head())
```

	Item_Identifier	Item_Weight	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Est
0	FDA15	9.30	0.015920	4	249.8092	OUT049	
1	DRC01	5.92	0.019095	14	48.2692	OUT018	
2	FDN15	17.50	0.016621	10	141.6180	OUT049	
3	FDX07	19.20	0.064037	6	182.0950	OUT010	
4	NCD19	8.93	0.064037	9	53.8614	OUT013	

5 rows × 26 columns

In [71]:
```
X = dt.drop(columns=['Outlet_Establishment_Year', 'Item_Identifier', 'Outlet_Identif
X.head(10)
```

Out[71]:

	Item_Weight	Item_Visibility	Item_Type	Item_MRP	Outlet_Years	Outlet	Item_Fat_Content_0	I
0	9.300000	0.015920	4	249.8092	23	9	1	
1	5.920000	0.019095	14	48.2692	13	3	0	
2	17.500000	0.016621	10	141.6180	23	9	1	
3	19.200000	0.064037	6	182.0950	24	0	0	
4	8.930000	0.064037	9	53.8614	35	1	0	
5	10.395000	0.064037	0	51.4008	13	3	0	
6	13.650000	0.012661	13	57.6588	35	1	0	
7	12.857645	0.119976	13	107.7622	37	5	1	
8	16.200000	0.016549	5	96.9726	20	7	0	
9	19.200000	0.090252	5	187.8214	15	2	0	

10 rows × 22 columns

In [72]:
```
y = dt['Item_Outlet_Sales']
y.head(10)
```

Out[72]:
```
0    8.225808
1    6.096776
2    7.648868
3    6.597664
4    6.903451
5    6.323658
6    5.842247
7    8.299973
8    6.982490
9    8.457769
Name: Item_Outlet_Sales, dtype: float64
```

In [73]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_stat
print (X_train.shape, X_test.shape ,  y_train.shape, y_test.shape)
```

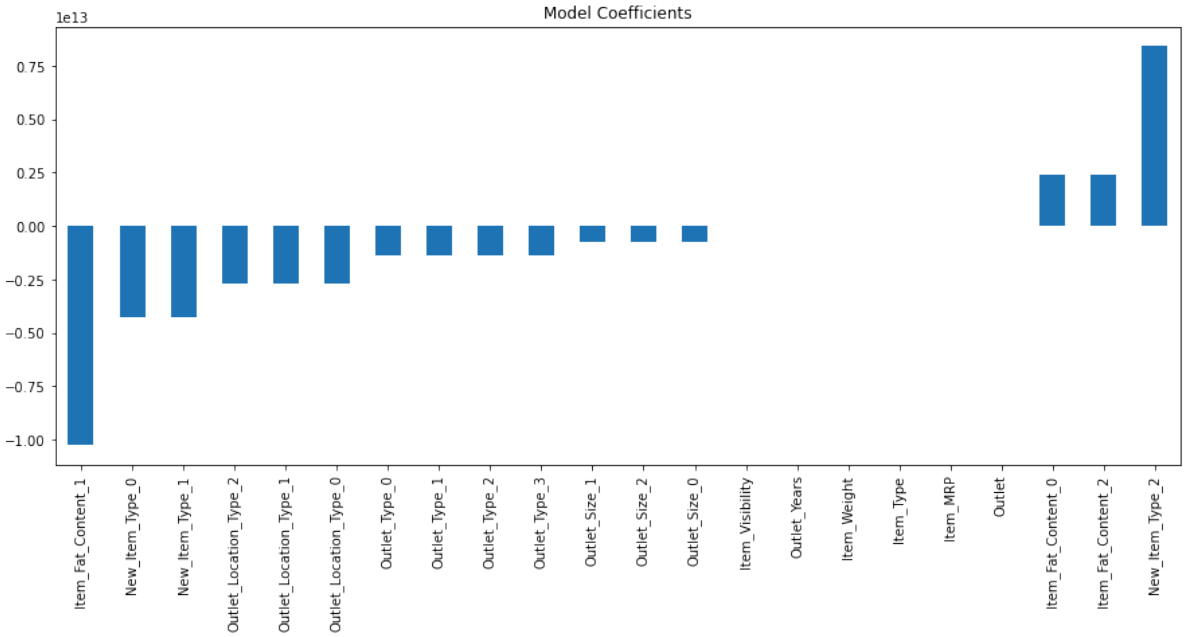
```
(5966, 22) (2557, 22) (5966,) (2557,)
```

In [74]:
```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
def train(model, X, y):
    # training the model
    model.fit(X, y)

    pred = model.predict(X)
    # perform cross-validation
    cv_score = cross_val_score(model, X, y, scoring='neg_mean_squared_error', cv=5)
    cv_score = np.abs(np.mean(cv_score))

    print("Model Report")
    print("CV Score:", cv_score)
    print("R2_Score:", r2_score(y,pred))
```

In [75]:
```
from sklearn.linear_model import LinearRegression, Ridge, Lasso
model = LinearRegression(normalize=True)
train(model, X_train, y_train)
```

```
coef = pd.Series(model.coef_, X.columns).sort_values()
print (coef)
coef.plot(kind='bar', title="Model Coefficients")
plt.show()
```

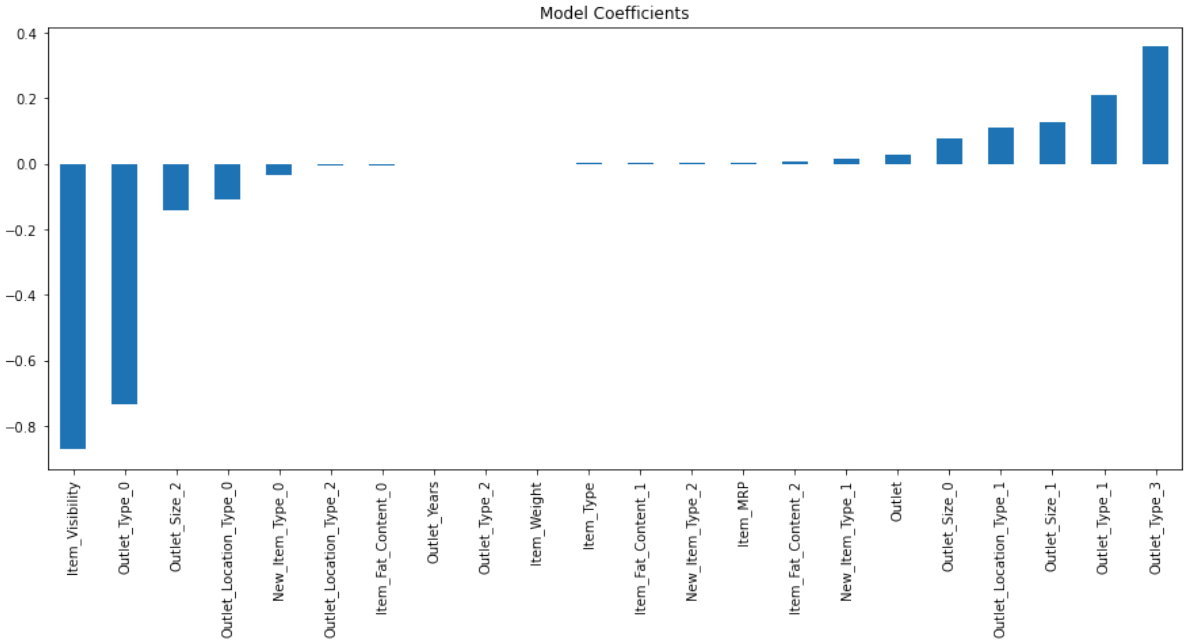
```
Model Report
CV Score: 0.2842367408376636
R2_Score: 0.7232417685607002
Item_Fat_Content_1       -1.023976e+13
New_Item_Type_0          -4.242821e+12
New_Item_Type_1          -4.242821e+12
Outlet_Location_Type_2   -2.669911e+12
Outlet_Location_Type_1   -2.669911e+12
Outlet_Location_Type_0   -2.669911e+12
Outlet_Type_0            -1.392581e+12
Outlet_Type_1            -1.392581e+12
Outlet_Type_2            -1.392581e+12
Outlet_Type_3            -1.392581e+12
Outlet_Size_1            -7.164604e+11
Outlet_Size_2            -7.164604e+11
Outlet_Size_0            -7.164604e+11
Item_Visibility          -7.635822e-02
Outlet_Years             -7.376485e-02
Item_Weight              -1.765038e-03
Item_Type                 1.723789e-03
Item_MRP                  8.166404e-03
Outlet                    6.239844e-02
Item_Fat_Content_0        2.425977e+12
Item_Fat_Content_2        2.425977e+12
New_Item_Type_2           8.422915e+12
dtype: float64
```



In [76]:

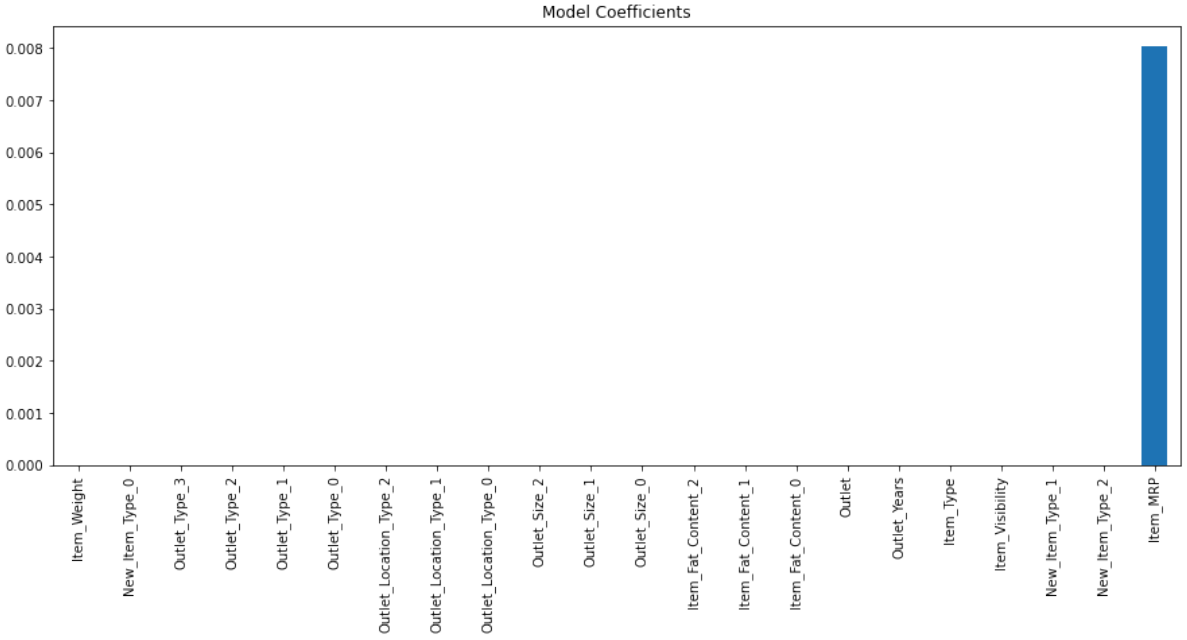
```
model = Ridge(normalize=True)
train(model, X_train, y_train)
coef = pd.Series(model.coef_, X.columns).sort_values()
coef.plot(kind='bar', title="Model Coefficients")
plt.show()
```

```
Model Report
CV Score: 0.42110063227787486
R2_Score: 0.5885164591166643
```



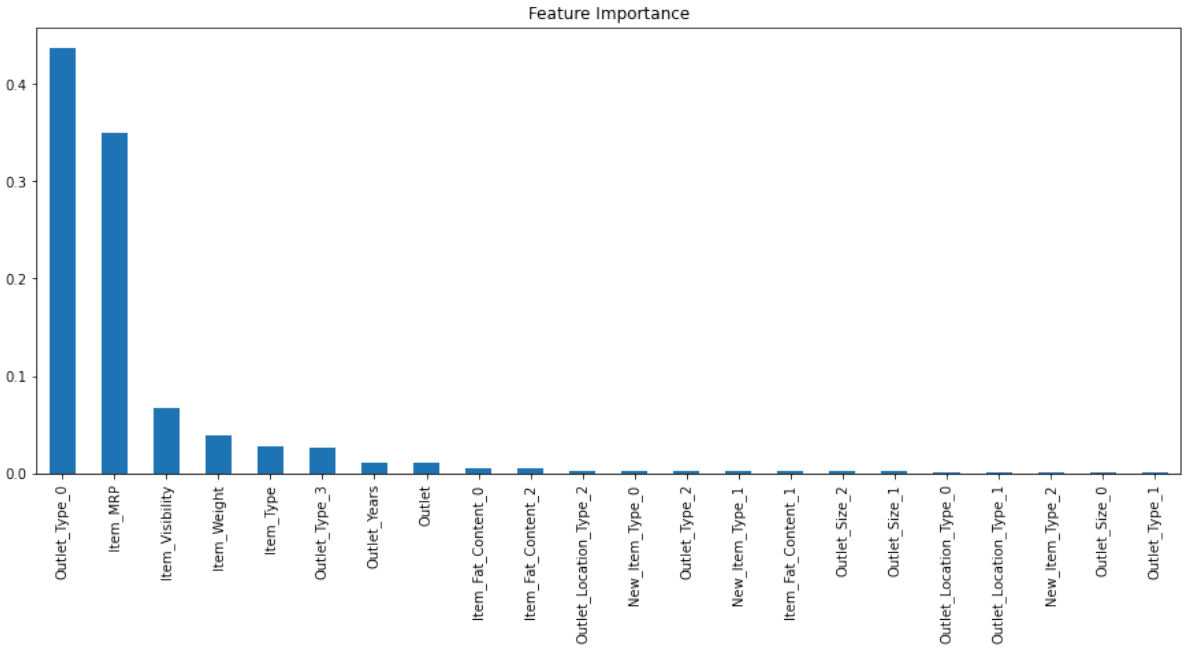
In [77]:
```
model = Lasso()
train(model, X_train, y_train)
coef = pd.Series(model.coef_, X.columns).sort_values()
coef.plot(kind='bar', title="Model Coefficients")
plt.show()
```

```
Model Report
CV Score: 0.7534899315973709
R2_Score: 0.26227683517122646
```



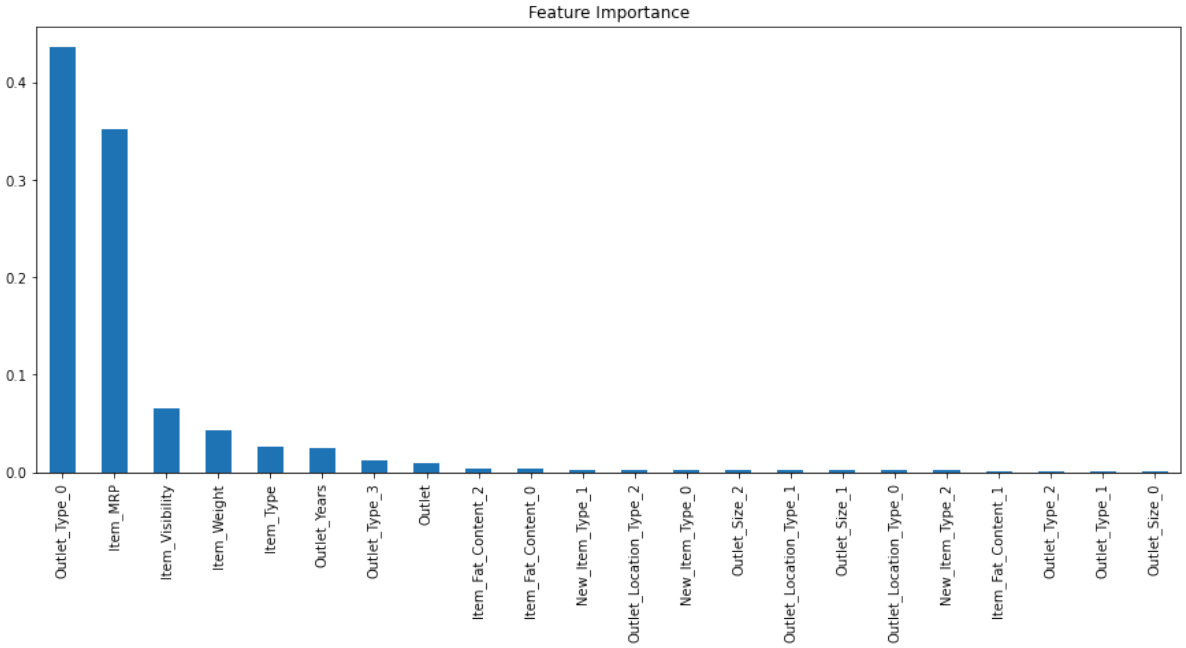
In [78]:
```
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor()
train(model,X_train, y_train)
coef = pd.Series(model.feature_importances_, X.columns).sort_values(ascending=False)
coef.plot(kind='bar', title="Feature Importance")
plt.show()
```

```
Model Report
CV Score: 0.5502580039455862
R2_Score: 1.0
```



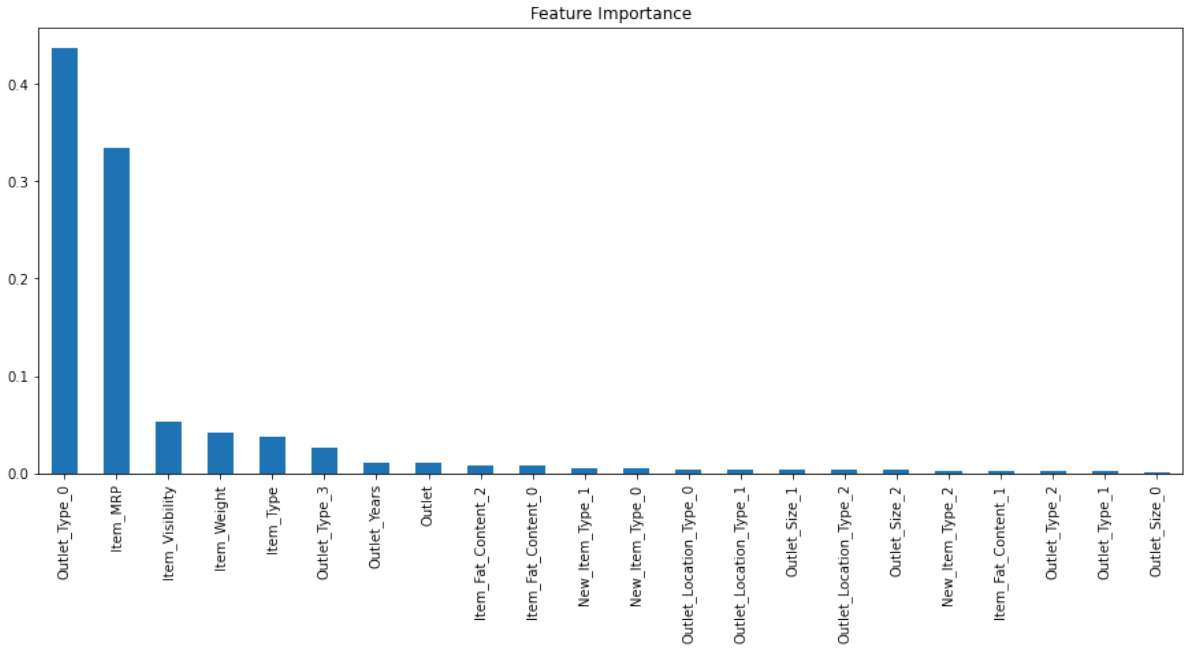
In [79]:
```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()
train(model, X_train, y_train)
coef = pd.Series(model.feature_importances_, X.columns).sort_values(ascending=False)
coef.plot(kind='bar', title="Feature Importance")
plt.show()
```

```
Model Report
CV Score: 0.29932434139277353
R2_Score: 0.9594031911222669
```



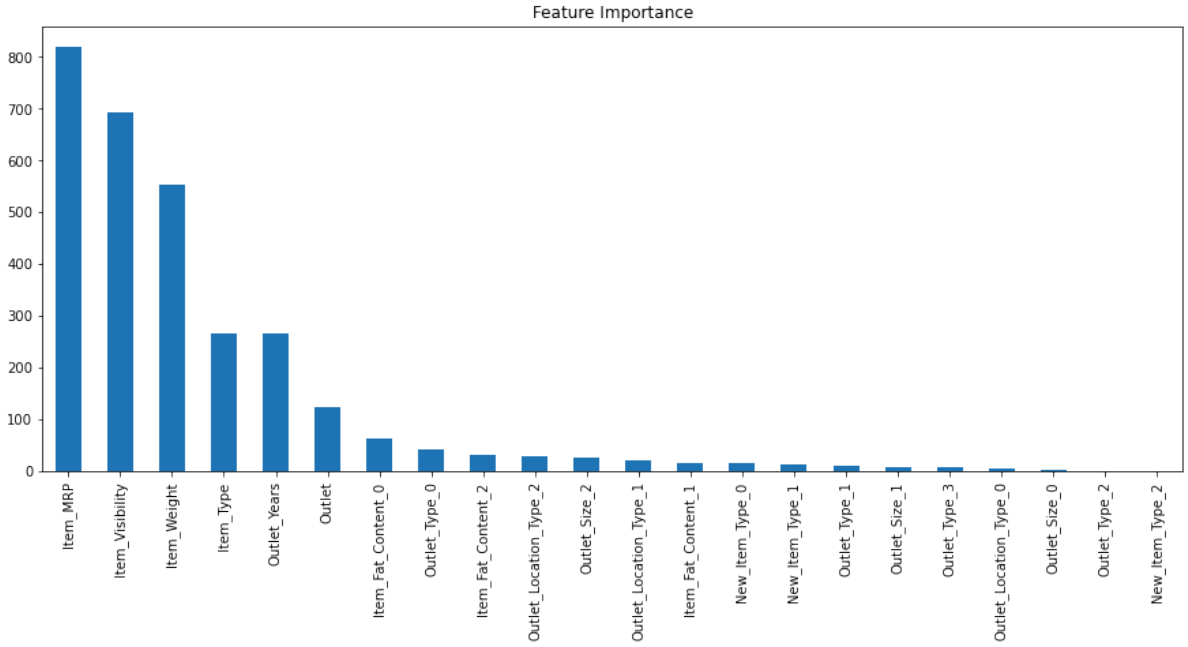
In [80]:
```
from sklearn.ensemble import ExtraTreesRegressor
model = ExtraTreesRegressor()
train(model, X_train, y_train)
coef = pd.Series(model.feature_importances_, X.columns).sort_values(ascending=False)
coef.plot(kind='bar', title="Feature Importance")
plt.show()
```

```
Model Report
CV Score: 0.32472233069111256
R2_Score: 1.0
```



In [81]:
```
from lightgbm import LGBMRegressor
model = LGBMRegressor()
train(model, X_train, y_train)
coef = pd.Series(model.feature_importances_, X.columns).sort_values(ascending=False)
coef.plot(kind='bar', title="Feature Importance")
plt.show()
```

```
Model Report
CV Score: 0.28146321091831317
R2_Score: 0.8145374278416446
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: