Library Import

In [ ]:
```python
import pandas as pd
import numpy as np
```

Load the Data

In [5]:
```python
import pandas as pd
df = pd.read_csv("https://raw.githubusercontent.com/dsrscientist/dataset1/master/train_datamart.csv")
df.head()
```

Out[5]:

	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
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	Tier 3	Grocery Store	732.3800
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052

Data Preparation and Cleaning

data preperation and cleaning is all about to prepare data for model building such as handle null values, missing values and impute suitable values at their place like mean, mode and median

In [6]:
```python
df.head(5)
```

Out[6]:

	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
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	Tier 3	Grocery Store	732.3800
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052

In [7]:
```python
df.shape
```

Out[7]:
```
(8523, 12)
```

in data we have 8523 rows and 12 column

In [8]:
```python
df.columns
```

Out[8]:
```
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]:
```python
df.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 [10]:
```python
df.describe()
```

Out[10]:

	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 [12]:
```python
#check for missing values
df.isna().sum()
```

Out[12]:
```
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
```

Only Item_Weight and Outlet_Size have missing values.

Item_Weight is a continuous variable. We can use either mean or median to impute the missing values, but here we will use mean.

Outlet_Size is a categorical variable so will use mode to impute the missing values in the column

In [13]:
```python
df.Item_Weight.fillna(df.Item_Weight.mean(), inplace=True)
```

In [14]:
```python
df.Item_Weight.isna().sum()
```

Out[14]:
```
0
```

now in Item_weight column there is no missing value

In [ ]:
```python
df.Outlet_Size.fillna(df.Outlet_Size.mode()[0],inplace=True)
```

In [15]:
```python
df.Outlet_Size.isnull().sum()
```

Out[15]:
```
2410
```

now in Outlet_Size column there is no missing value

In [ ]:
```python
import jovian
```

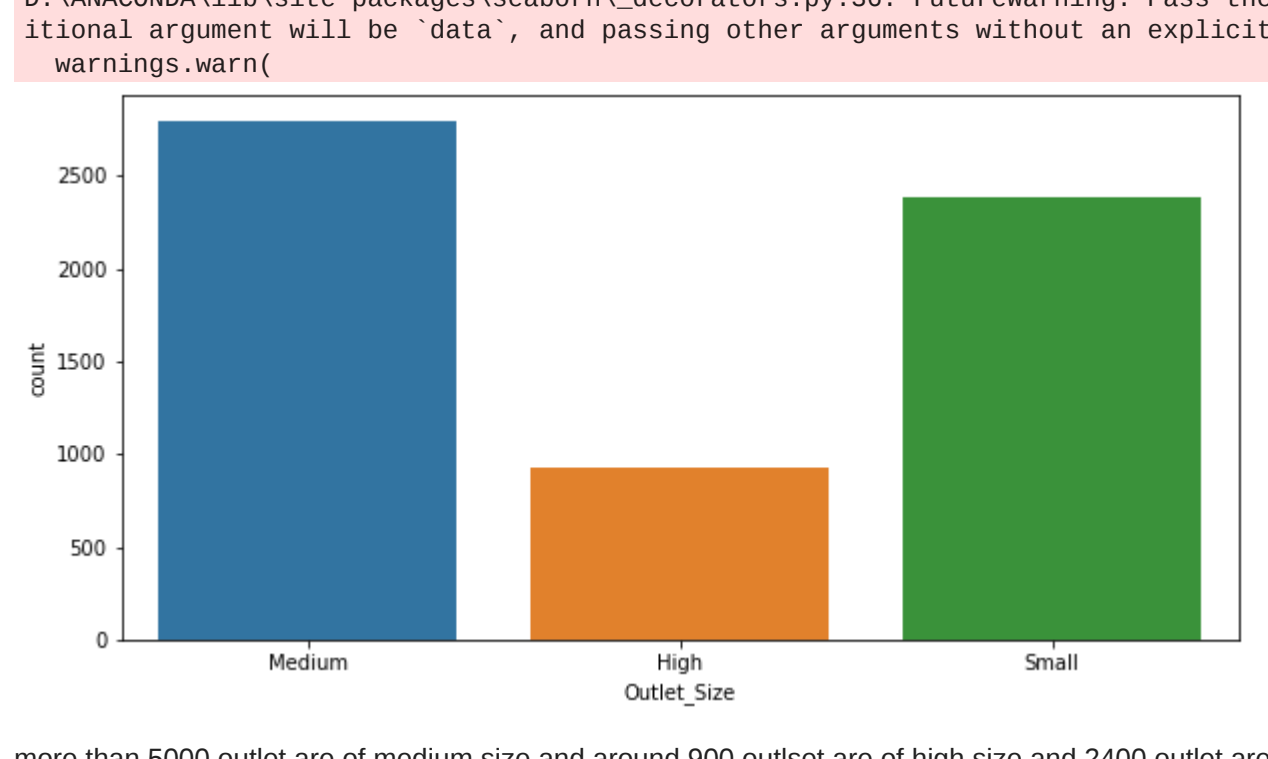
In [ ]:
```python
jovian.commit()
```

In [ ]:
```python
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (9, 5)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

Let's begin by importing matplotlib.pyplot and seaborn.

In [18]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(10,5))
sns.countplot(df.Outlet_Size);
```

```
D:\ANACONDA\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

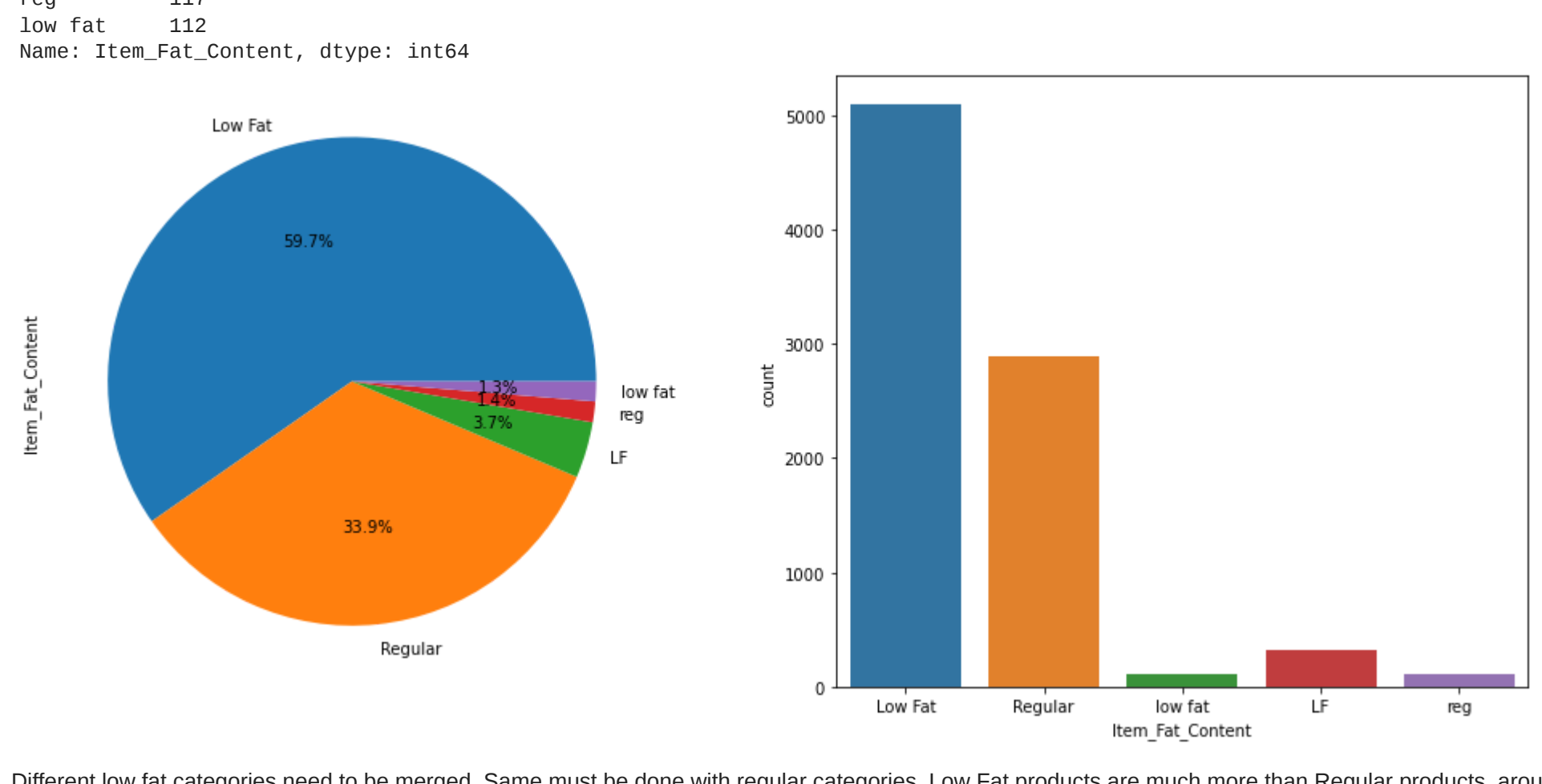


more than 5000 outlet are of medium size and around 900 outlset are of high size and 2400 outlet are of small size

In [19]:
```python
plt.figure(figsize=(17,7))
plt.subplot(1,2,1)
df['Item_Fat_Content'].value_counts().plot.pie(autopct='%1.1f%%')
plt.subplot(1,2,2)
sns.countplot(df['Item_Fat_Content'])
df['Item_Fat_Content'].value_counts()
```

```
D:\ANACONDA\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[19]:
```
Low Fat    5089
Regular    2889
LF          316
reg         117
low fat     112
Name: Item_Fat_Content, dtype: int64
```

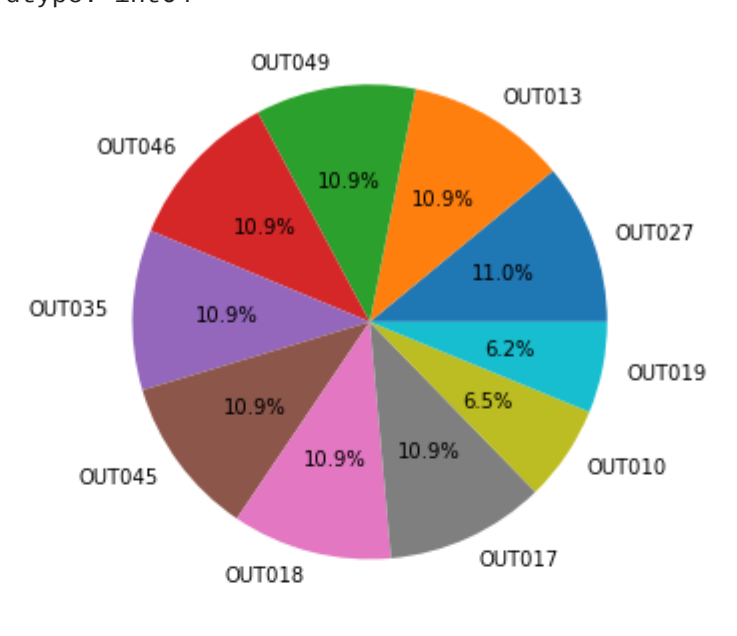


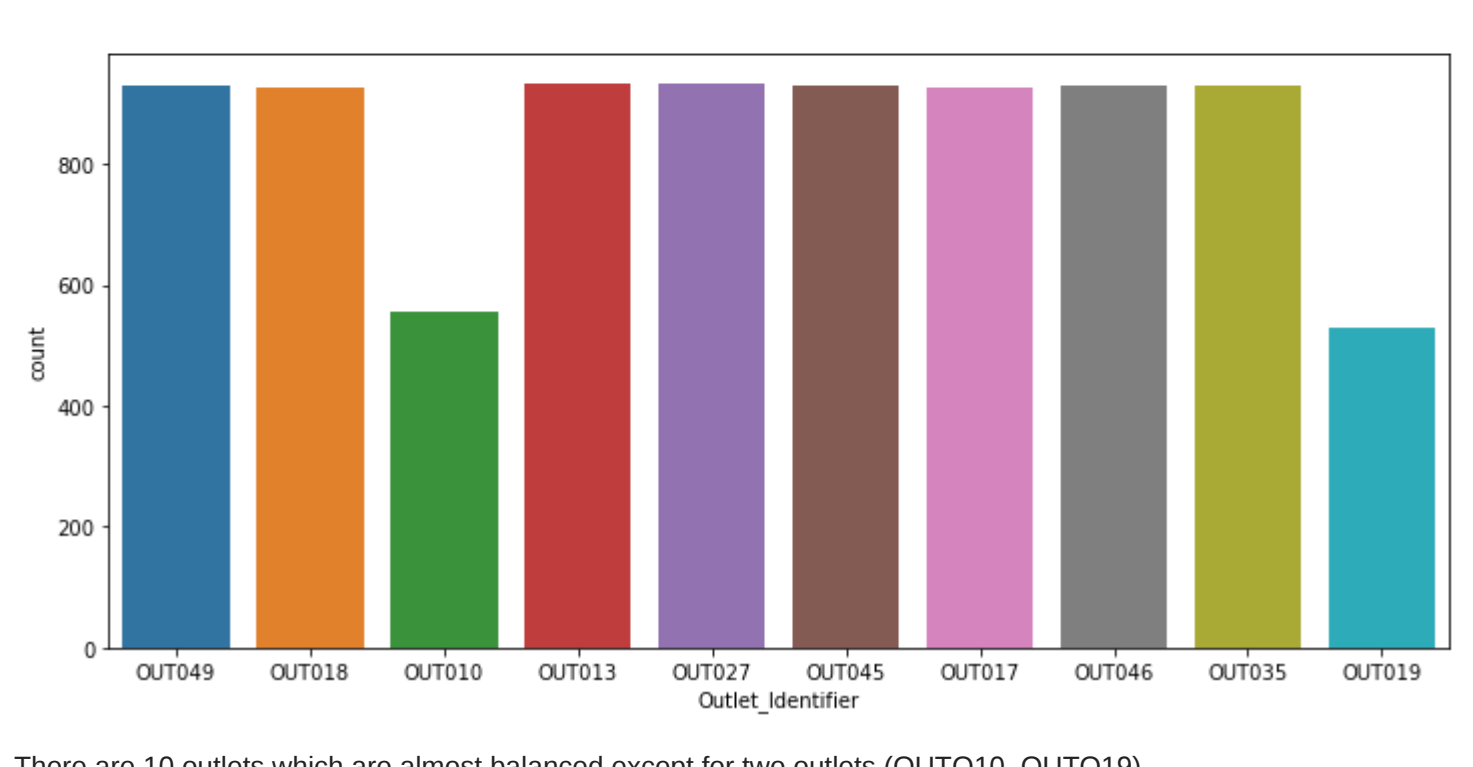
Different low fat categories need to be merged. Same must be done with regular categories. Low Fat products are much more than Regular products. around 60% food are low fat content and 34% food are regular fat content

In [20]:
```python
plt.figure(figsize=(12,12))
plt.subplot(2,1,1)
df['Outlet_Identifier'].value_counts().plot.pie(autopct='%1.1f%%')
plt.ylabel('')
plt.subplot(2,1,2)
sns.countplot(df['Outlet_Identifier'])
df['Outlet_Identifier'].value_counts()
```

```
D:\ANACONDA\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[20]:
```
OUT027    935
OUT013    932
OUT049    930
OUT046    930
OUT035    930
OUT045    929
OUT018    928
OUT017    926
OUT010    555
OUT019    528
Name: Outlet_Identifier, dtype: int64
```

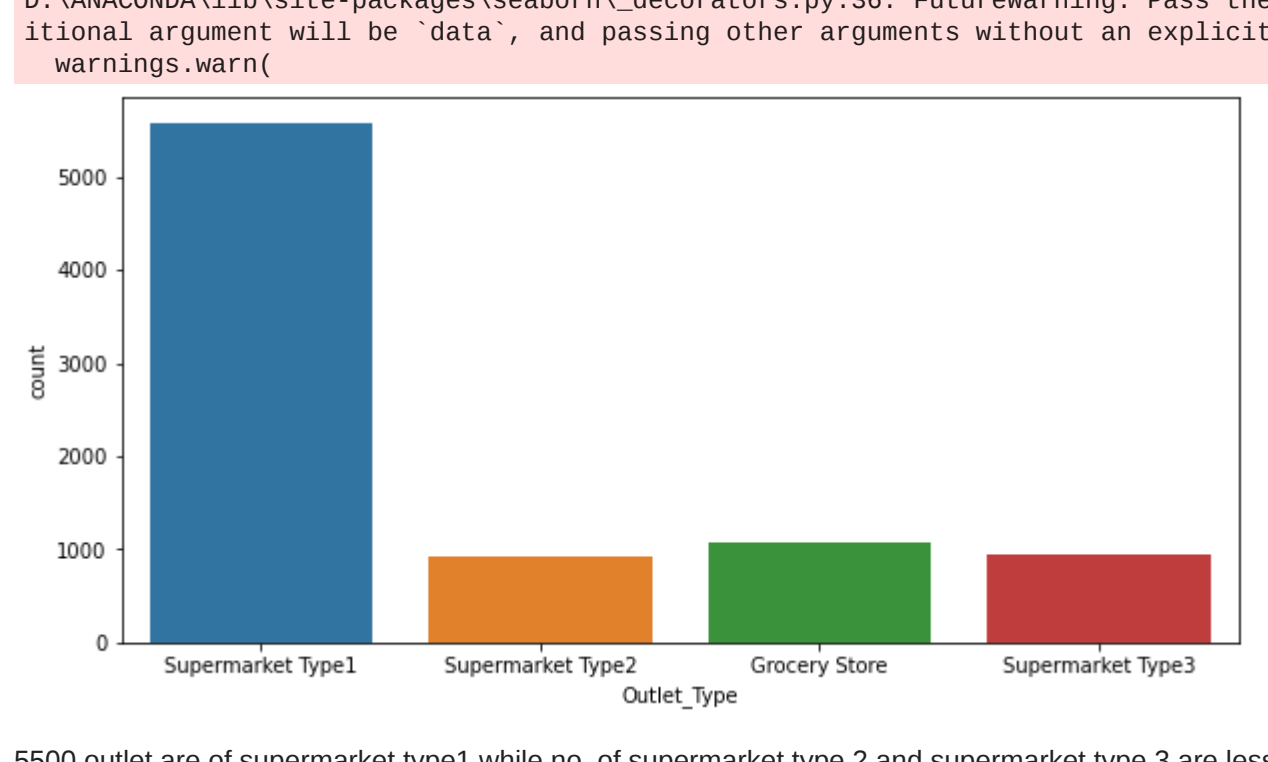




There are 10 outlets which are almost balanced except for two outlets (OUTO10, OUTO19)

In [21]:
```python
print(df.Outlet_Type.value_counts())
plt.figure(figsize=(10,5))

sns.countplot(df.Outlet_Type);
```

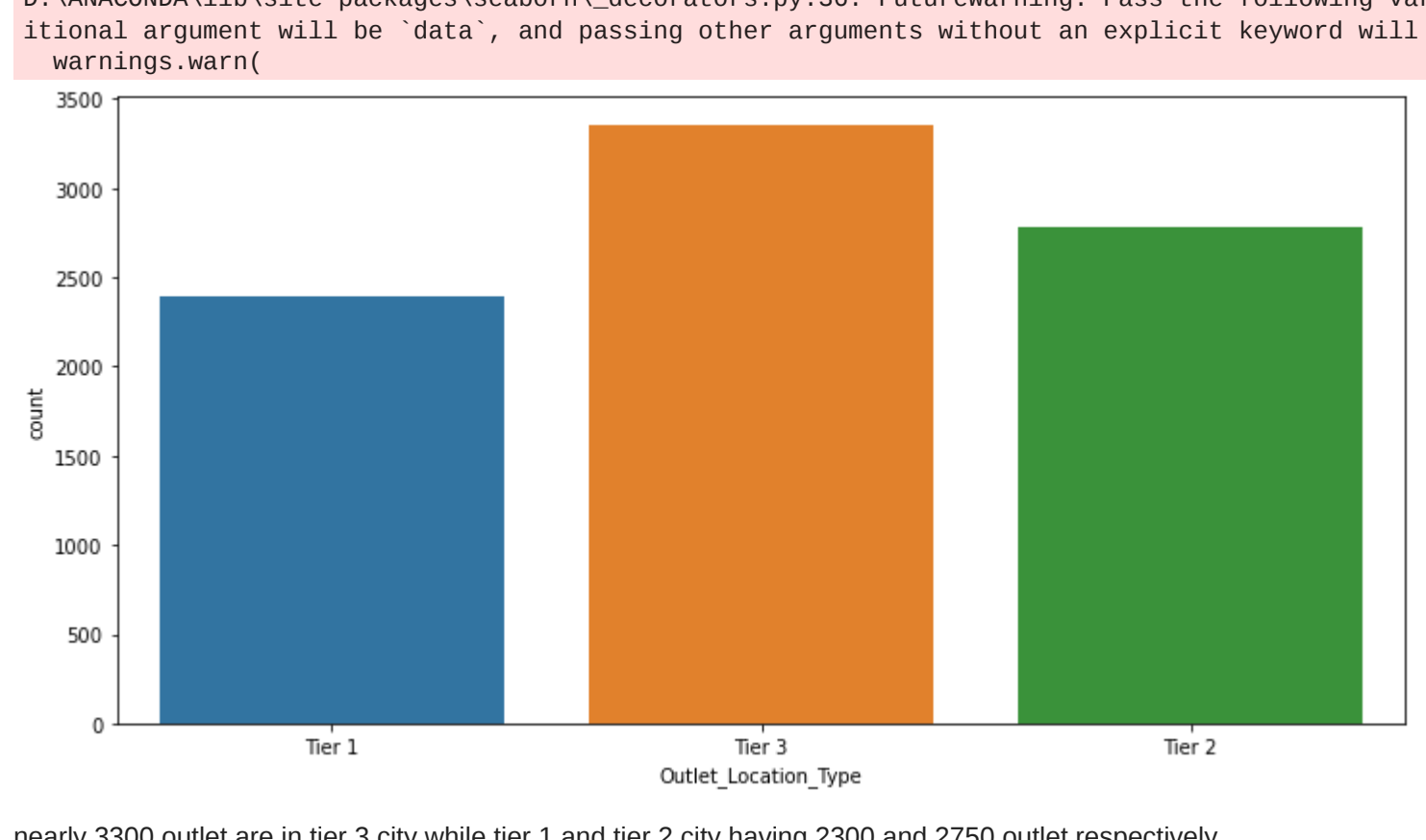
```
Supermarket Type1    5577
Grocery Store        1083
Supermarket Type3     935
Supermarket Type2     928
Name: Outlet_Type, dtype: int64
D:\ANACONDA\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



5500 outlet are of supermarket type1 while no. of supermarket type 2 and supermarket type 3 are less than 1000

In [22]:
```python
plt.figure(figsize=(12,6))
sns.countplot(df.Outlet_Location_Type);
```

```
D:\ANACONDA\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

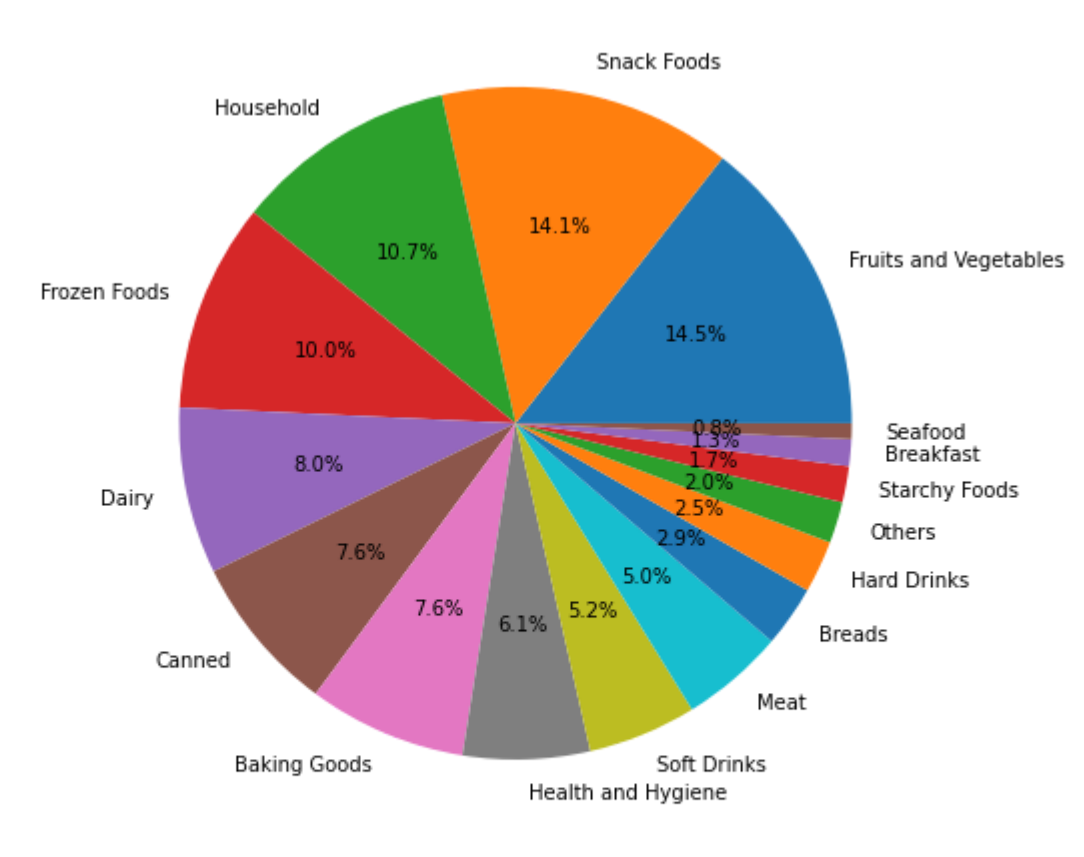


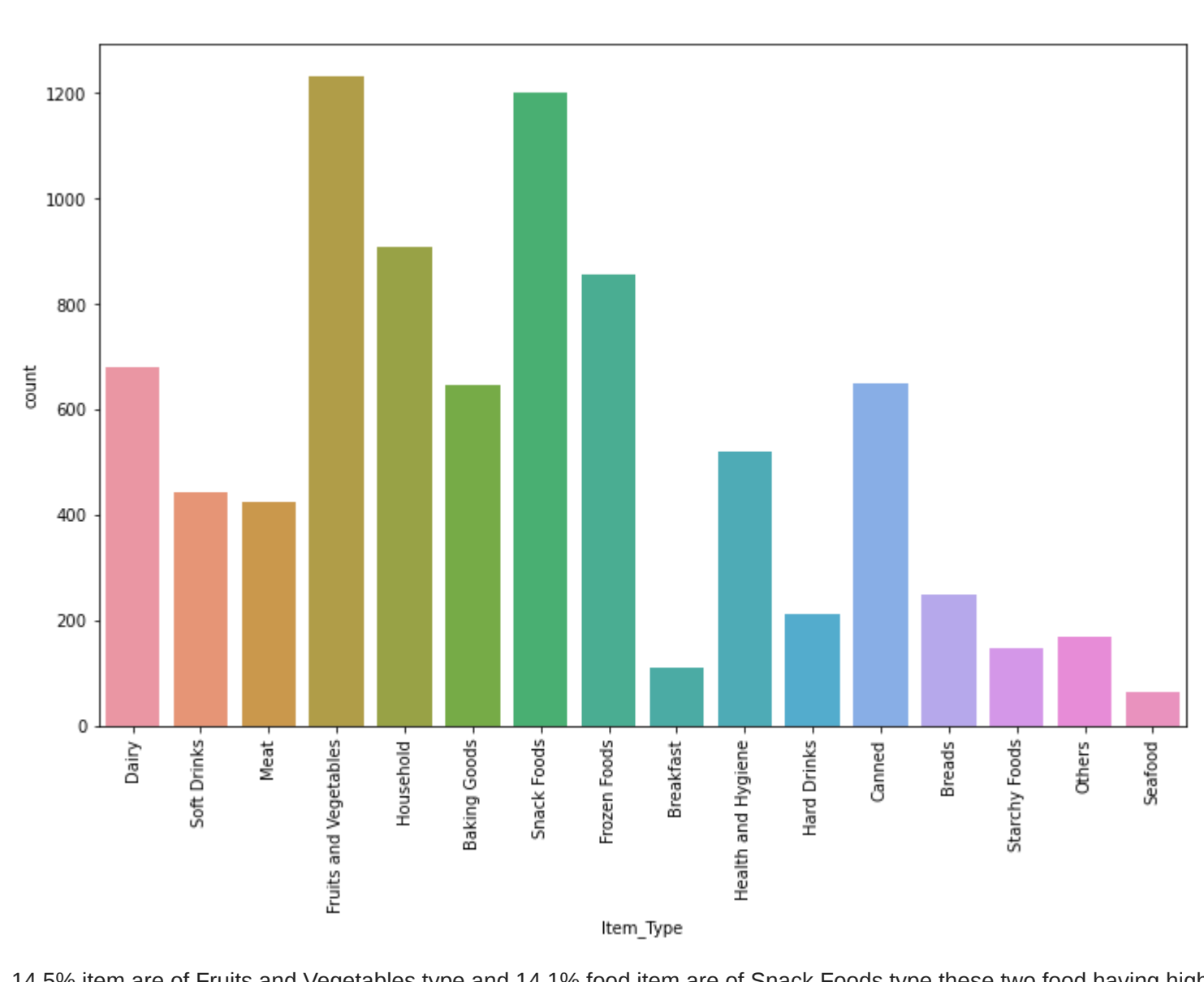
nearly 3300 outlet are in tier 3 city while tier 1 and tier 2 city having 2300 and 2750 outlet respectively

In [23]:
```python
plt.figure(figsize=(12,17))
plt.subplot(2,1,1)
df['Item_Type'].value_counts().plot.pie(autopct='%1.1f%%',textprops={'fontsize':10})
plt.ylabel('')
plt.subplot(2,1,2)
sns.countplot(df['Item_Type'])
plt.xticks(rotation = 90)
df['Item_Type'].value_counts()
```

```
D:\ANACONDA\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[23]:
```
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
```



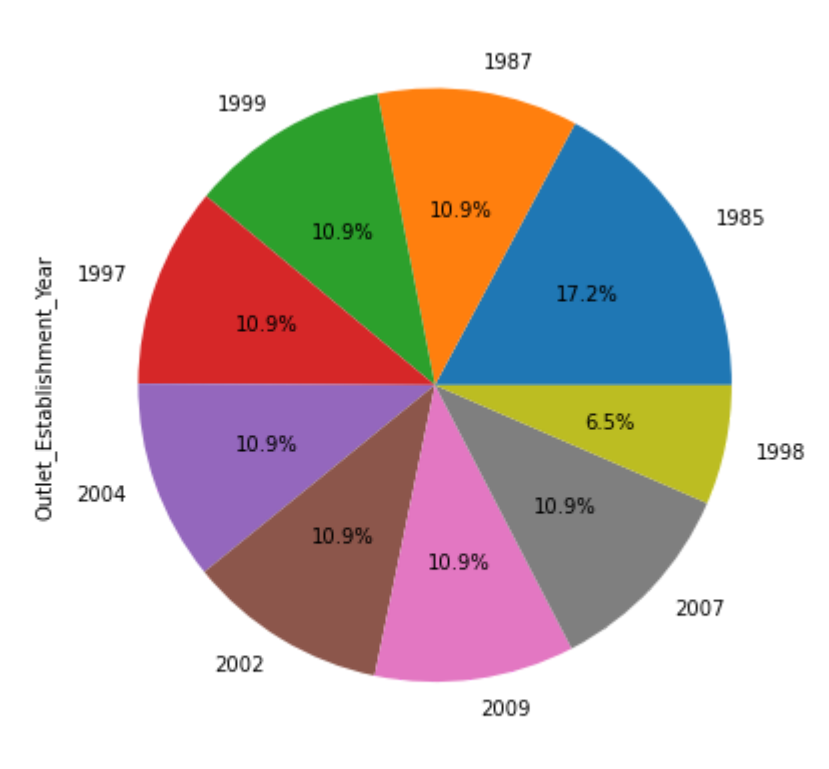


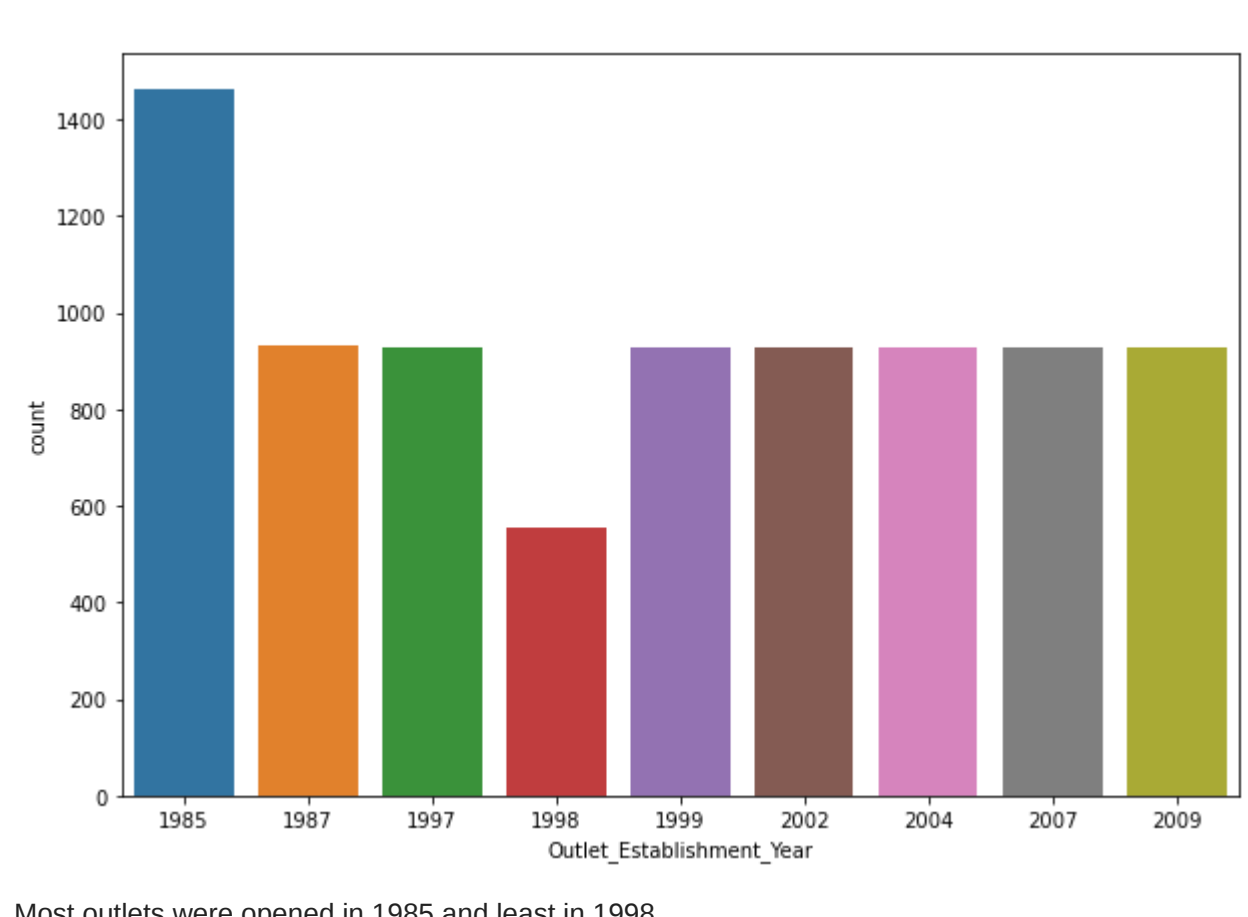
14.5% item are of Fruits and Vegetables type and 14.1% food item are of Snack Foods type these two food having highest share amond all food type

In [24]:
```python
plt.figure(figsize=(10,15))
plt.subplot(2,1,1)
df['Outlet_Establishment_Year'].value_counts().plot.pie(autopct='%1.1f%%')
plt.subplot(2,1,2)
sns.countplot(df['Outlet_Establishment_Year'])
df['Outlet_Establishment_Year'].value_counts()
```

```
D:\ANACONDA\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[24]:
```
1985    1463
1987     932
1999     930
1997     930
2004     930
2002     929
2009     928
2007     926
1998     555
Name: Outlet_Establishment_Year, dtype: int64
```



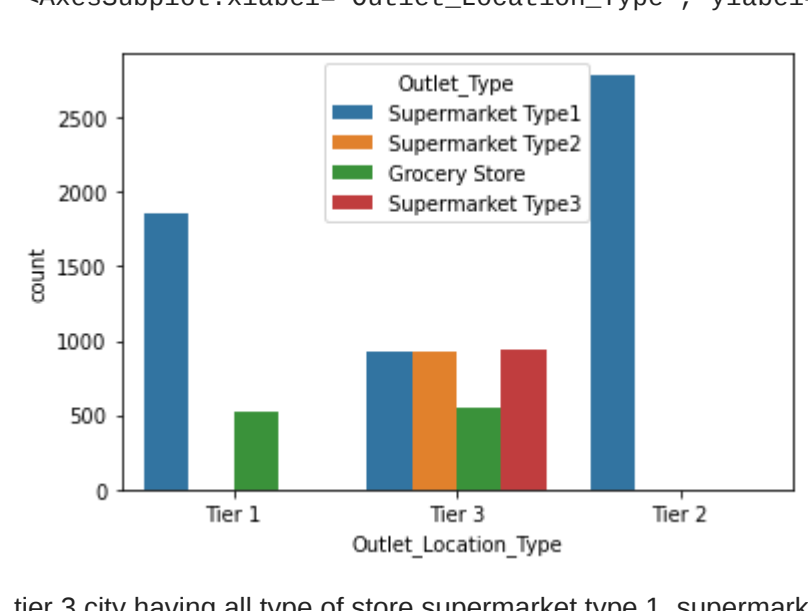


Most outlets were opened in 1985 and least in 1998

In [25]:
```python
sns.countplot(df['Outlet_Location_Type'],hue=df['Outlet_Type'])
```

```
D:\ANACONDA\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[25]:
```
<AxesSubplot:xlabel='Outlet_Location_Type', ylabel='count'>
```



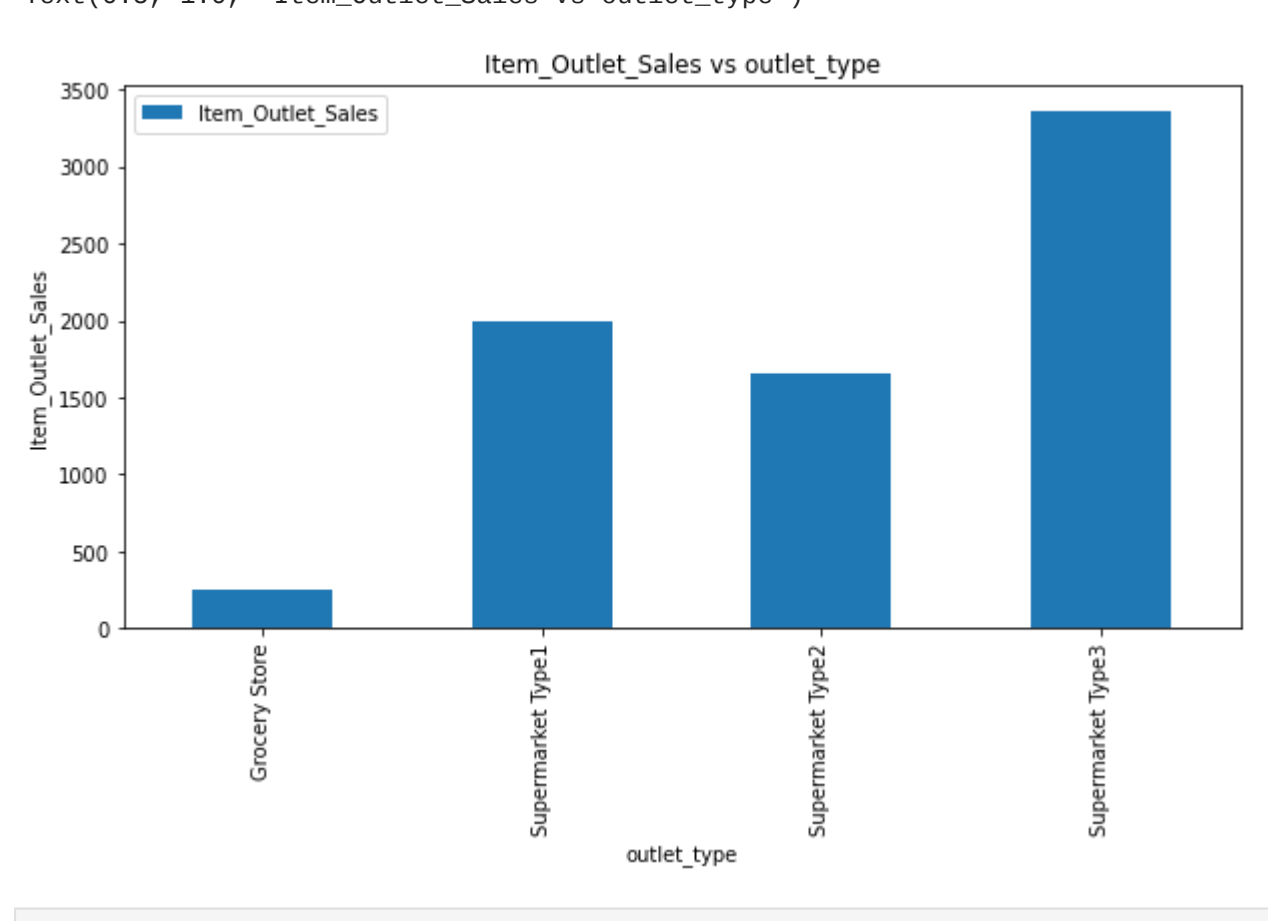
tier 3 city having all type of store supermarket type 1, supermarket type 2, super market 3 and grocery store while tier2 city having only supermarket type 1 store

In [ ]:
```python
import jovian
```

In [ ]:
```python
jovian.commit()
```

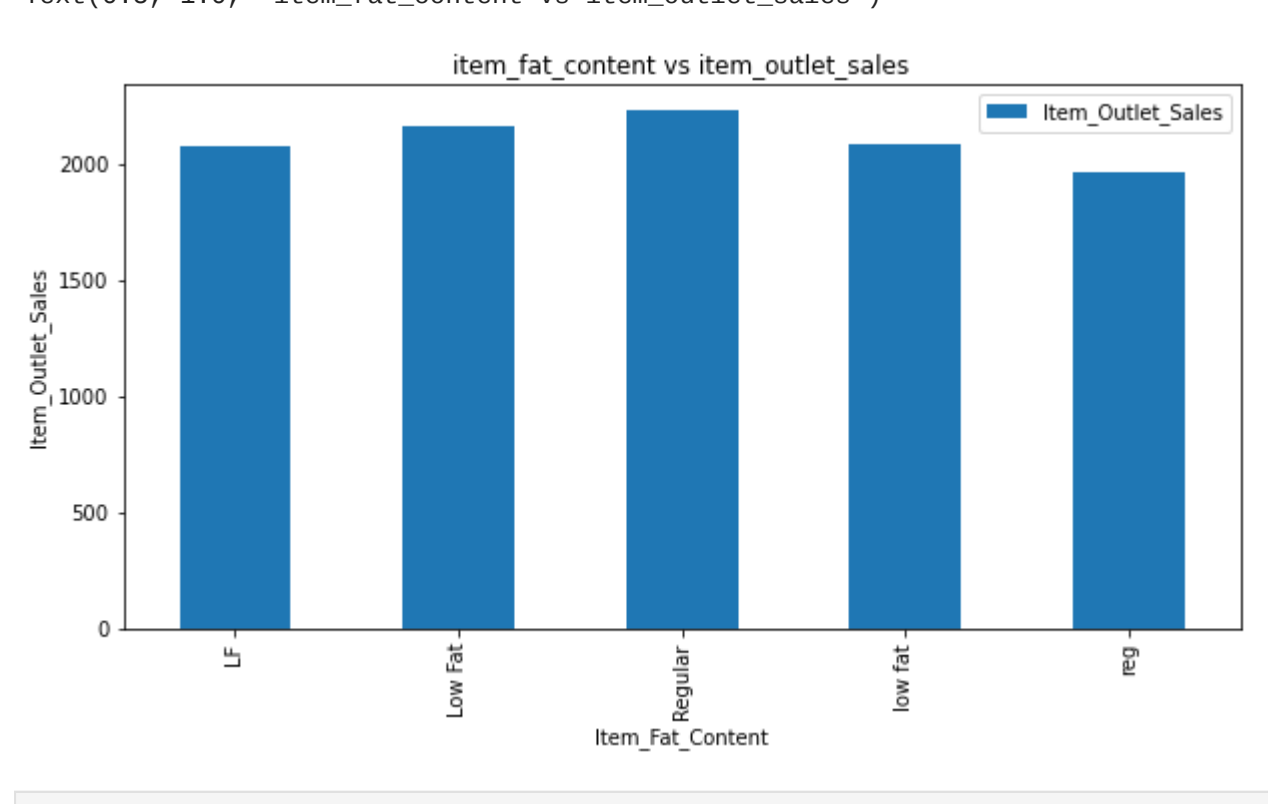
In [27]:
```python
#question which type of outlset haiving highest sales

#answers supermarket3 type of outlet having highest sales
import numpy as np
Outlet_Type_pivot=\
df.pivot_table(index='Outlet_Type',values='Item_Outlet_Sales',aggfunc=np.median)
Outlet_Type_pivot.plot(kind='bar',figsize=(10,5))
plt.xlabel('outlet_type')
plt.ylabel('Item_Outlet_Sales')
plt.title('Item_Outlet_Sales vs outlet_type')
```

Out[27]:
```
Text(0.5, 1.0, 'Item_Outlet_Sales vs outlet_type')
```



In [28]:
```python
Item_Fat_Content_pivot= \
df.pivot_table(index='Item_Fat_Content', values='Item_Outlet_Sales', aggfunc=np.mean)
Item_Fat_Content_pivot.plot(kind='bar',figsize=(10,5))
plt.xlabel('Item_Fat_Content')
plt.ylabel('Item_Outlet_Sales')
plt.title('item_fat_content vs item_outlet_sales')
```

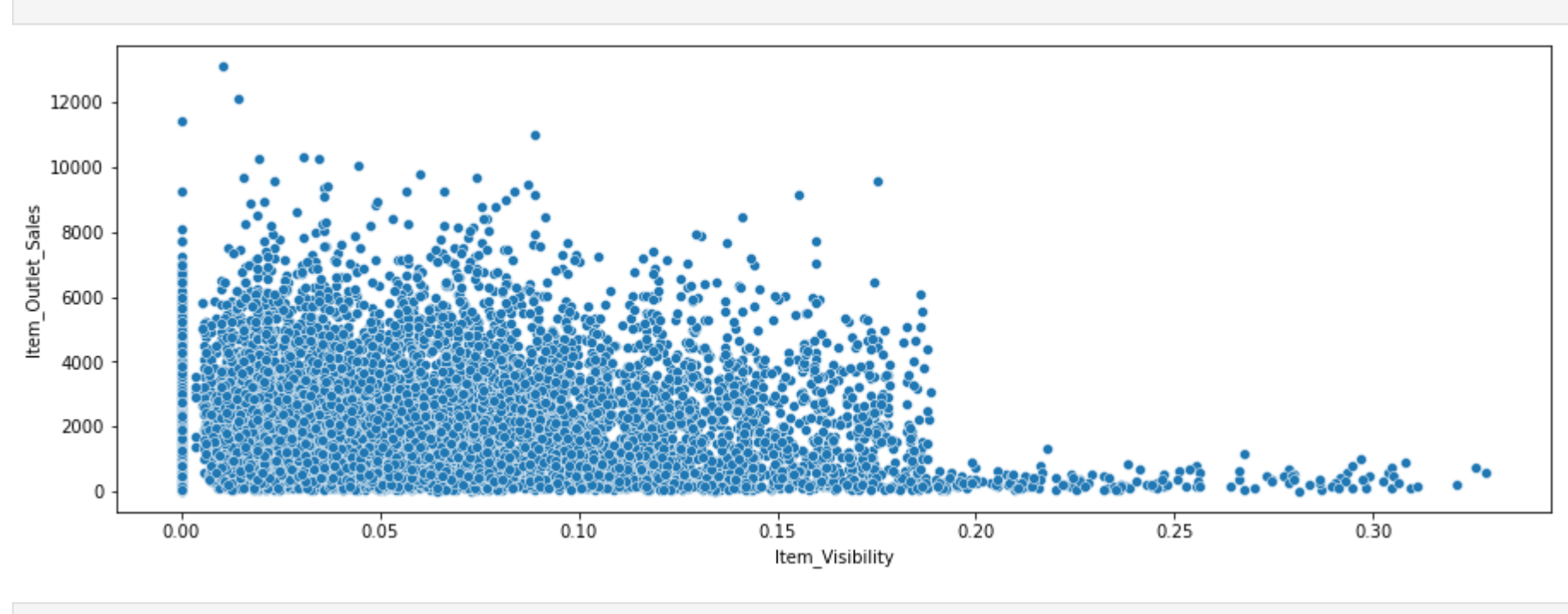
Out[28]:
```
Text(0.5, 1.0, 'item_fat_content vs item_outlet_sales')
```



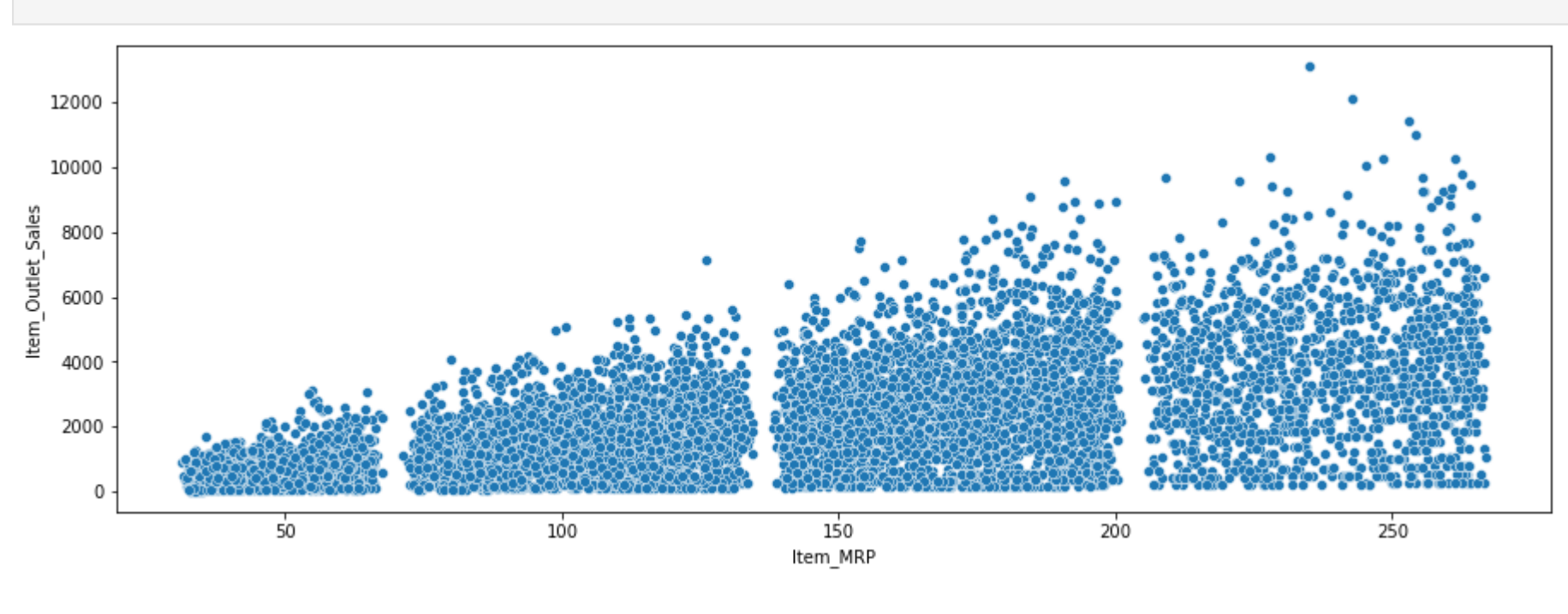
In [29]:
```python
df['Item_Fat_Content'].value_counts()
```

Out[29]:
```
Low Fat    5089
Regular    2889
LF          316
reg         117
low fat     112
Name: Item_Fat_Content, dtype: int64
```

In [30]:
```python
plt.figure(figsize=(15,5))
sns.scatterplot(x = df['Item_Visibility'], y = df['Item_Outlet_Sales'])
plt.show()
```



In [31]:
```python
plt.figure(figsize=(15,5))
sns.scatterplot(x = df['Item_MRP'], y = df['Item_Outlet_Sales'])
plt.show()
```



CONCLUSION:

It can be concluded that more locations should be switched or shifted to Supermarket Type3 to increase the sales of products at Big Mart. Any one-stop-shopping-center like Big Mart can benefit from this model by being able to predict its items' future sales at different locations.

In [ ]:
```
THANK YOU
```