In [1]:
```python
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
from sklearn import linear_model
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
```

In [2]:
```python
train =pd.read_csv("bigdatamart_Train.csv")
test =pd.read_csv("bigdatamart_Test.csv")
```

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

Out[3]:

	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 [4]:
```python
train.shape
```

Out[4]:
```
(8523, 12)
```

In [5]:
```python
test.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
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999	Medium	Tier 1	Supermarket Type1
1	FDW14	8.300	reg	0.038428	Dairy	87.3198	OUT017	2007	NaN	Tier 2	Supermarket Type1
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	OUT010	1998	NaN	Tier 3	Grocery Store
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007	NaN	Tier 2	Supermarket Type1
4	FDY38	NaN	Regular	0.118599	Dairy	234.2300	OUT027	1985	Medium	Tier 3	Supermarket Type3

In [6]:
```python
test.shape
```

Out[6]:
```
(5681, 11)
```

In [7]:
```python
train.dtypes, test.dtypes
```

Out[7]:
```
(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,
 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
 dtype: object)
```

In [9]:
```python
train['source']='train'
test['source']='test'
both = pd.concat([train, test], ignore_index = True)
both.head()
```

Out[9]:

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

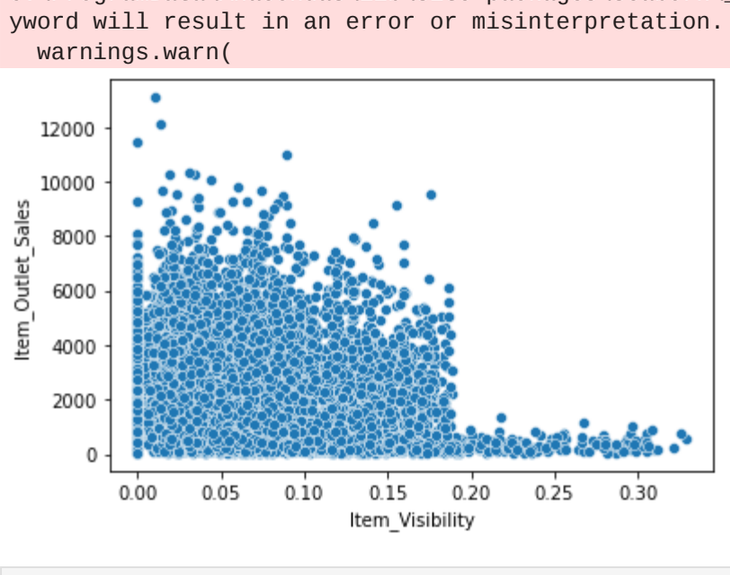
In [10]:
```python
both.describe()
```

Out[10]:

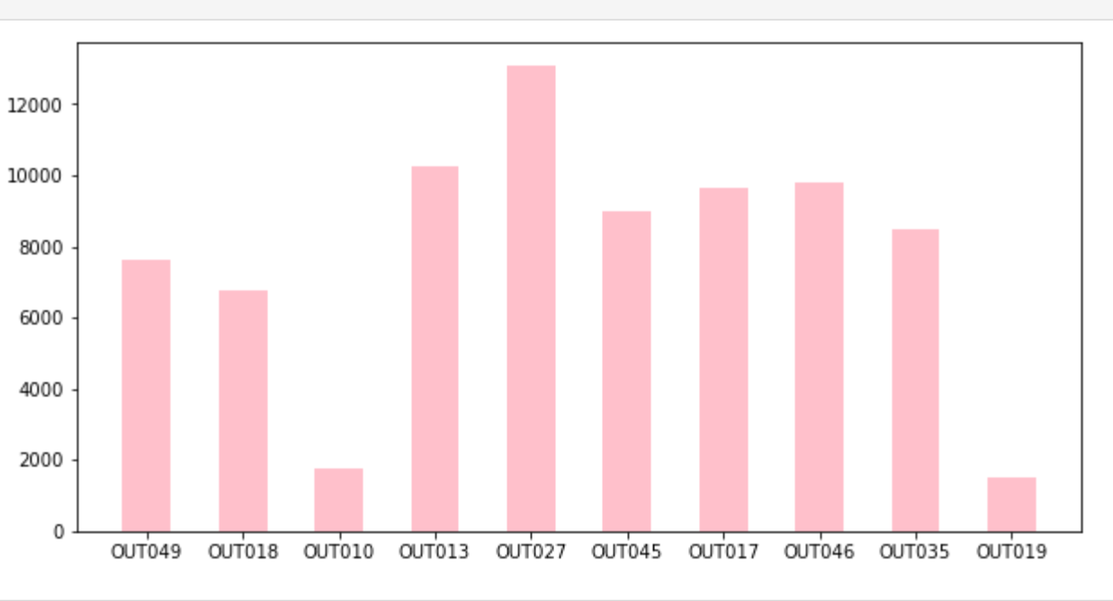
	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	11765.000000	14204.000000	14204.000000	14204.000000	8523.000000
mean	12.792854	0.065953	141.004977	1997.830681	2181.288914
std	4.652502	0.051459	62.086938	8.371664	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	8.710000	0.027036	94.012000	1987.000000	834.247400
50%	12.600000	0.054021	142.247000	1999.000000	1794.331000
75%	16.750000	0.094037	185.855600	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

In [13]:
```python
sns.scatterplot(train["Item_Visibility"], train["Item_Outlet_Sales"])
```

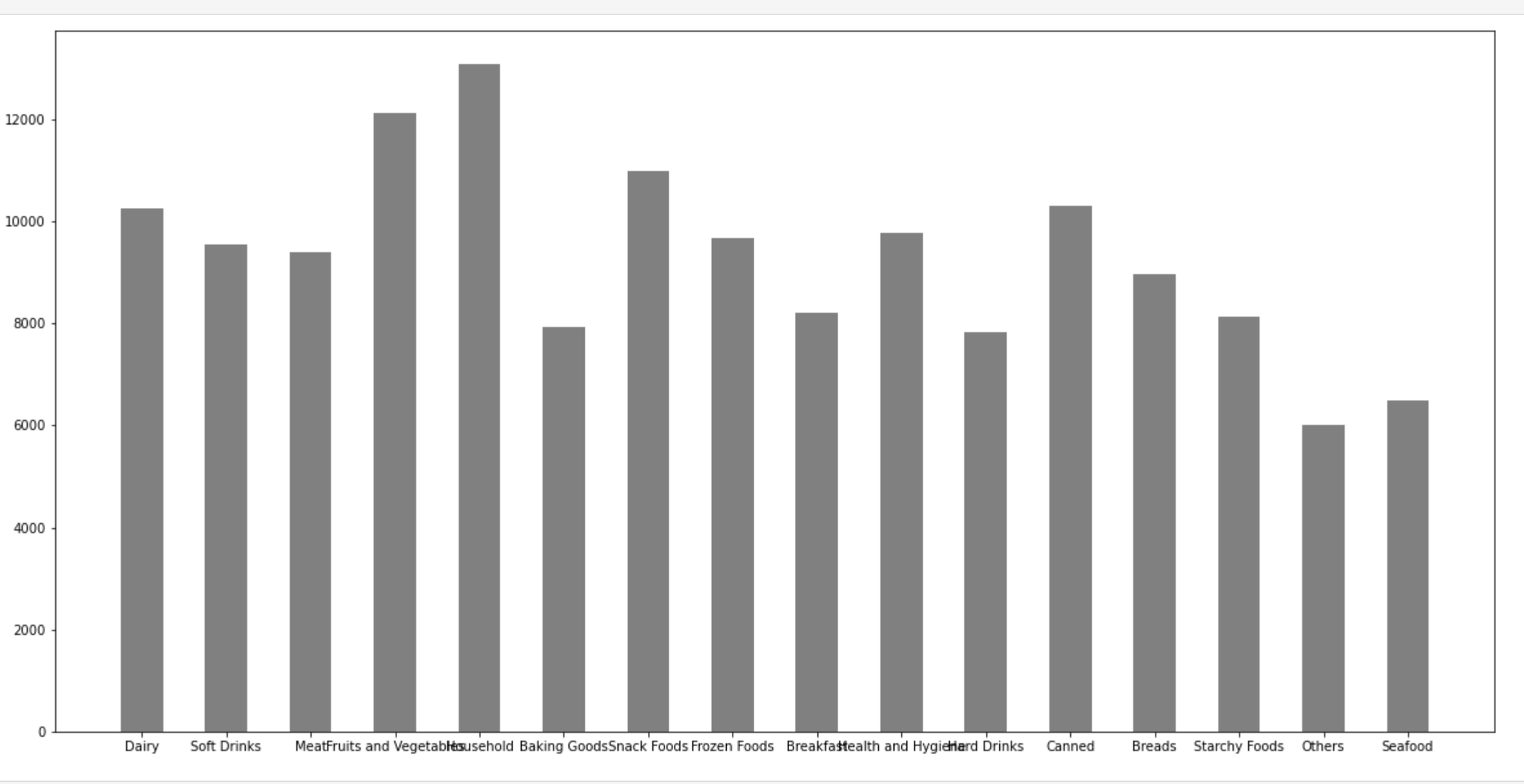
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



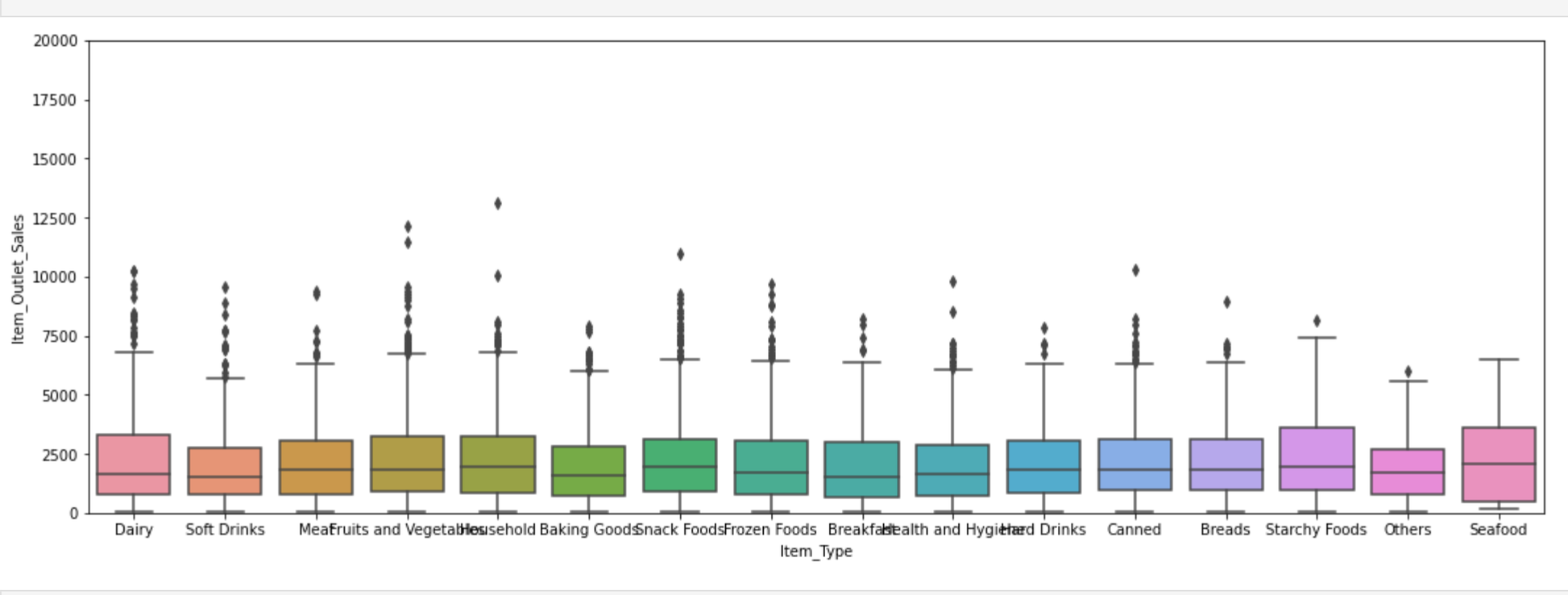
In [16]:
```python
#Inference- values having the most sales have visibillity less than 0.2
fig, ax = plt.subplots(figsize=(10,5))
plt.bar(train["Outlet_Identifier"],train["Item_Outlet_Sales"], width = 0.5 , color ='pink')
plt.show()
```



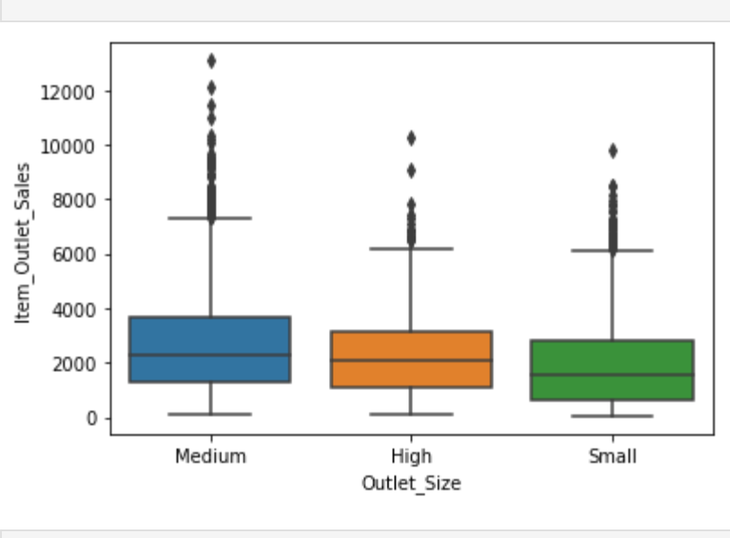
In [21]:
```python
#plot of sales vs type of product
fig, ax = plt.subplots(figsize=(20,10))
plt.bar(train["Item_Type"],train["Item_Outlet_Sales"], width = 0.5 , color ='grey')
plt.show()
```



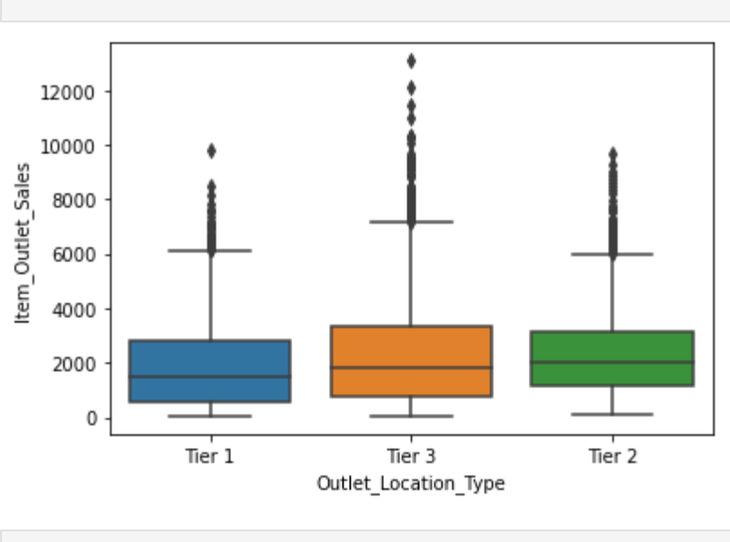
In [23]:
```python
data = pd.concat([train["Item_Type"], train["Item_Outlet_Sales"]], axis=1)
f, ax = plt.subplots(figsize=(18, 6))
fig = sns.boxplot(x="Item_Type", y="Item_Outlet_Sales", data=data)
fig.axis(ymin=0, ymax=20000)
plt.show()
```



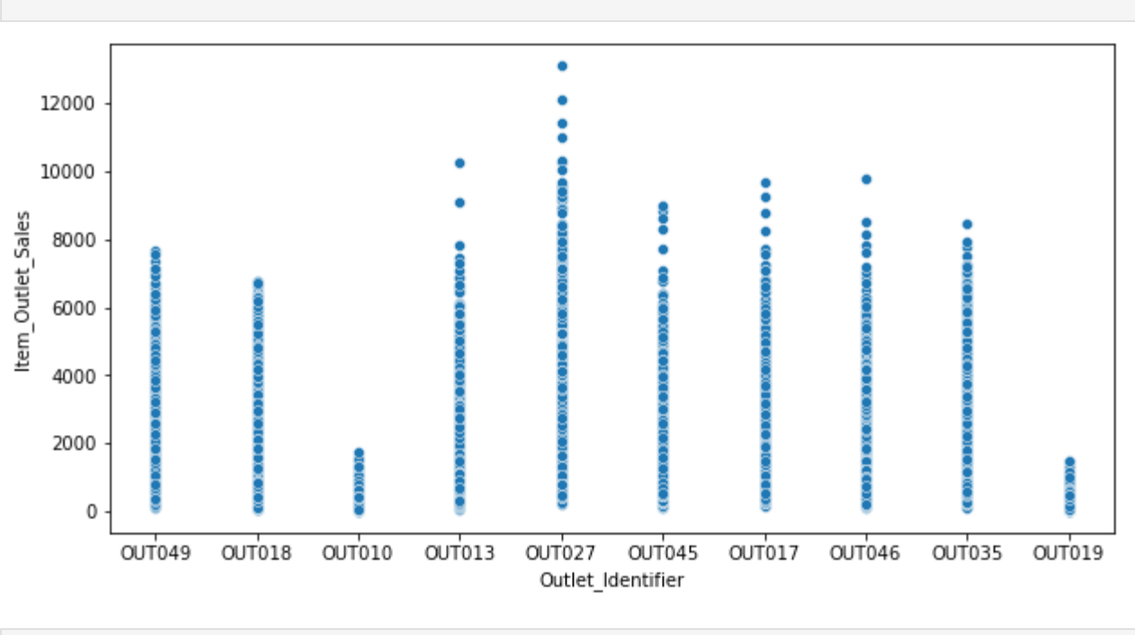
In [24]:
```python
#BOXPLOT ANALYSIS
sns.boxplot(x = 'Outlet_Size', y = 'Item_Outlet_Sales', data = combi)
plt.show()
```



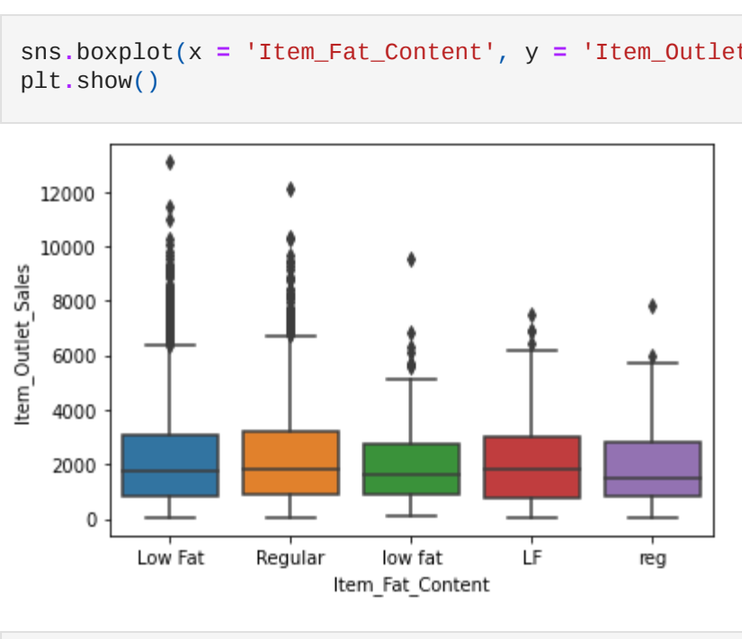
In [25]:
```python
sns.boxplot(x = 'Outlet_Location_Type', y = 'Item_Outlet_Sales', data = combi)
plt.show()
```



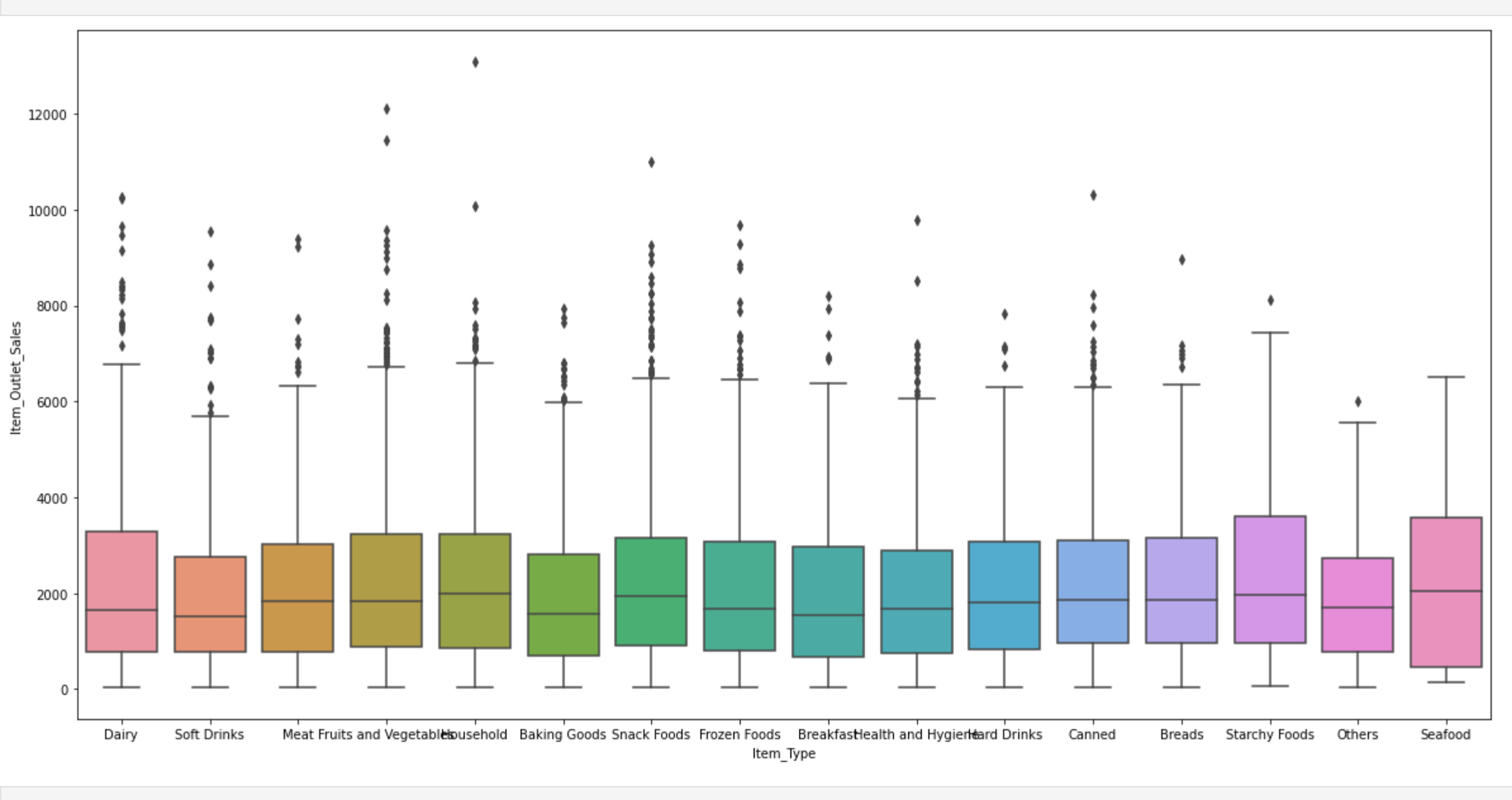
In [26]:
```python
plt.figure(figsize = (10,5))
sns.scatterplot(x = 'Outlet_Identifier', y = 'Item_Outlet_Sales', data = both)
plt.show()
```



In [27]:
```python
sns.boxplot(x = 'Item_Fat_Content', y = 'Item_Outlet_Sales', data = both)
plt.show()
```



In [28]:
```python
plt.figure(figsize = (20,10))
sns.boxplot(x = 'Item_Type', y = 'Item_Outlet_Sales', data =both)
plt.show()
```



In [29]:
```python
both.isnull().sum()
```

Out[29]:
```
Item_Identifier                 0
Item_Weight                  2439
Item_Fat_Content                0
Item_Visibility                 0
Item_Type                       0
Item_MRP                        0
Outlet_Identifier               0
Outlet_Establishment_Year       0
Outlet_Size                  4016
Outlet_Location_Type            0
Outlet_Type                     0
Item_Outlet_Sales            5681
source                          0
dtype: int64
```

In [30]:
```python
both["Item_Outlet_Sales"].fillna(both["Item_Outlet_Sales"].mean(), inplace=True)
both['Item_Weight'].fillna(both['Item_Weight'].mean(), inplace=True)
```

In [31]:
```python
both['Outlet_Size'].value_counts()
```

Out[31]:
```
Medium    4655
Small     3980
High      1553
Name: Outlet_Size, dtype: int64
```

In [32]:
```python
both['Outlet_Size'].fillna(('Medium'), inplace = True)
```

In [33]:
```python
both.isnull().sum()
```

Out[33]:
```
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
source                       0
dtype: int64
```

In [34]:
```python
#0 in item visibility isn't practical so treating it as a NaN value. Replacing 0 with NaN
both["Item_Visibility"].replace(0, np.nan, inplace=True)
```

In [35]:
```python
#Imputing missing values in item visibillity by median
both['Item_Visibility'].fillna(both['Item_Visibility'].median(), inplace=True)
```

In [36]:
```python
both.head()
```

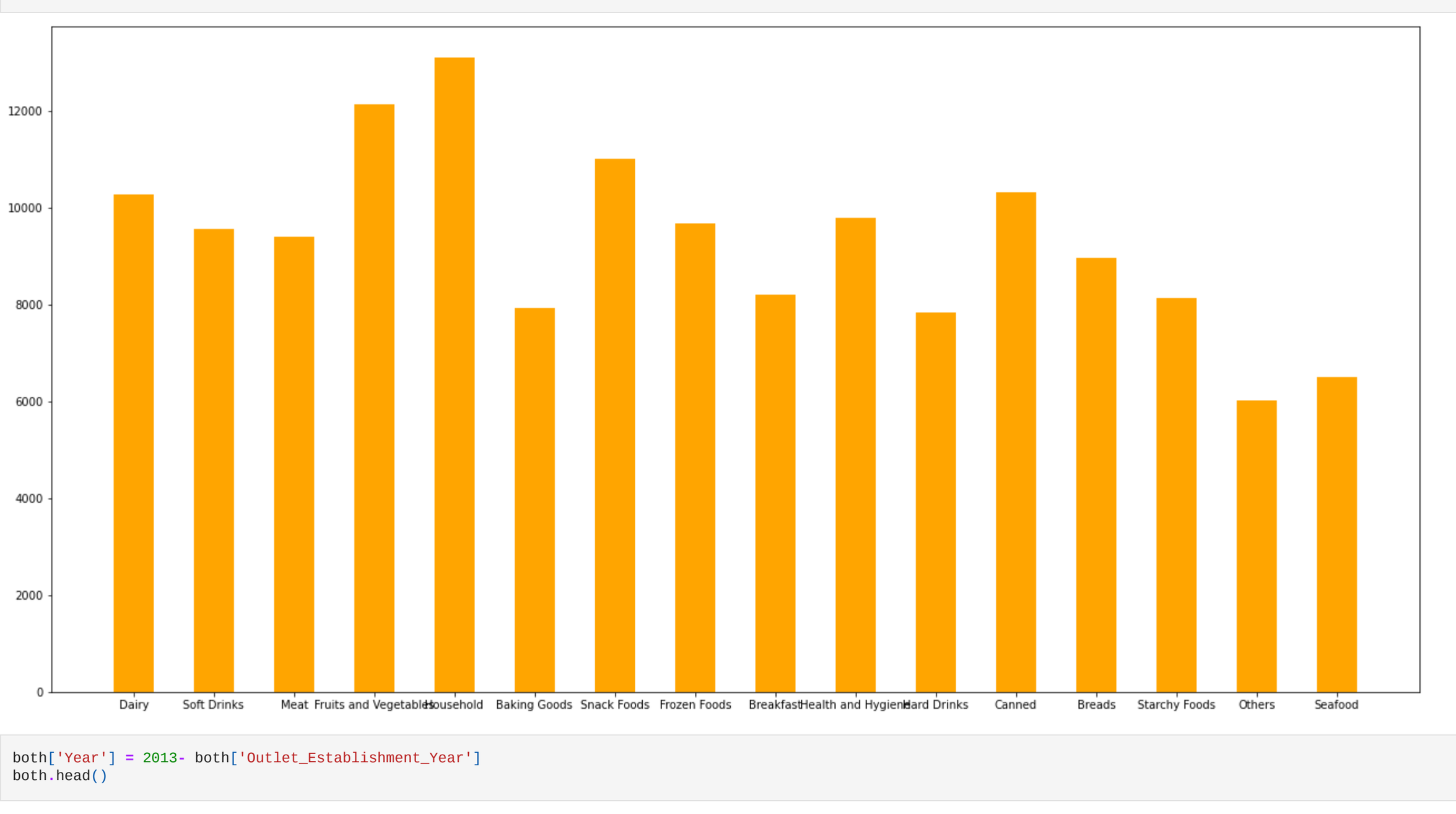
Out[36]:

	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	source
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380	train
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228	train
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700	train
3	FDX07	19.20	Regular	0.057851	Fruits and Vegetables	182.0950	OUT010	1998	Medium	Tier 3	Grocery Store	732.3800	train
4	NCD19	8.93	Low Fat	0.057851	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052	train

In [37]:
```python
both["Item_Fat_Content"].replace('LF', 'Low Fat', inplace=True)
both["Item_Fat_Content"].replace('low fat', 'Low Fat', inplace=True)
both["Item_Fat_Content"].replace('reg', 'Regular', inplace=True)
both["Item_Fat_Content"].value_counts()
```

Out[37]:
```
Low Fat    9185
Regular    5019
Name: Item_Fat_Content, dtype: int64
```

In [40]:
```python
fig, ax = plt.subplots(figsize=(22,11))
plt.bar(both["Item_Type"],both["Item_Outlet_Sales"], width = 0.5 , color ='orange')
plt.show()
```

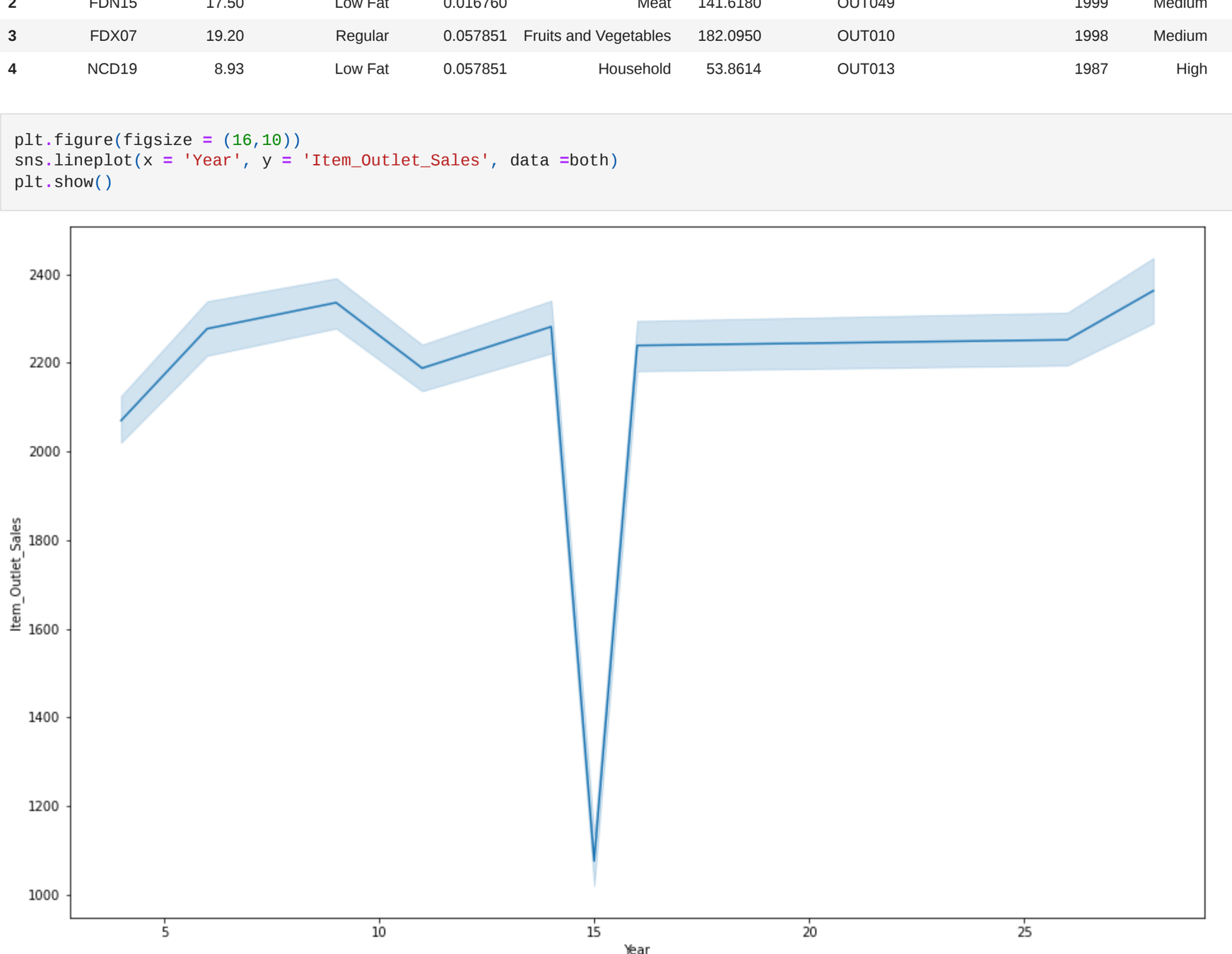


In [41]:
```python
both['Year'] = 2013- both['Outlet_Establishment_Year']
both.head()
```

Out[41]:

	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	source	Year
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380	train	14
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228	train	4
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700	train	14
3	FDX07	19.20	Regular	0.057851	Fruits and Vegetables	182.0950	OUT010	1998	Medium	Tier 3	Grocery Store	732.3800	train	15
4	NCD19	8.93	Low Fat	0.057851	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052	train	26

In [42]:
```python
plt.figure(figsize = (16,10))
sns.lineplot(x = 'Year', y = 'Item_Outlet_Sales', data =both)
plt.show()
```



In [43]:
```python
both['Item_Identifier'].value_counts()
both['Item_Type_Combined'] = both['Item_Identifier'].apply(lambda x: x[0:2])
both['Item_Type_Combined'] = both['Item_Type_Combined'].map({'FD':'Food',
                                                             'NC':'Non-Consumable',
                                                             'DR':'Drinks'})
both['Item_Type_Combined'].value_counts()
```

Out[43]:
```
Food              10201
Non-Consumable     2686
Drinks             1317
Name: Item_Type_Combined, dtype: int64
```

In [44]:
```python
#LABEL ENCODING
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
le = LabelEncoder()
both['Outlet'] = le.fit_transform(both['Outlet_Identifier'])
var_mod = ['Item_Fat_Content','Outlet_Location_Type','Outlet_Size','Item_Type_Combined','Outlet_Type','Outlet']
le = LabelEncoder()
for i in var_mod:
    both[i] = le.fit_transform(both[i])
```

In [46]:
```python
both = pd.get_dummies(both, columns=['Item_Fat_Content','Outlet_Location_Type','Outlet_Size','Outlet_Type','Item_Type_Combined'])
both.head()
```

Out[46]:

	Item_Identifier	Item_Weight	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Item_Outlet_Sales	source	Year	...	Outlet_Size_0	Outlet_Size_1	Outlet_Size_2	Outlet_Type_0	Outlet_Type_1	Outlet_Type_2	Outlet_Type_3	Item_Type_Combined_0	Item_Type_Combined_1	Item_Type_Combine
0	FDA15	9.30	0.016047	Dairy	249.8092	OUT049	1999	3735.1380	train	14	...	0	1	0	0	1	0	0	0	1	
1	DRC01	5.92	0.019278	Soft Drinks	48.2692	OUT018	2009	443.4228	train	4	...	0	1	0	0	0	1	0	1	0	
2	FDN15	17.50	0.016760	Meat	141.6180	OUT049	1999	2097.2700	train	14	...	0	1	0	0	1	0	0	0	1	
3	FDX07	19.20	0.057851	Fruits and Vegetables	182.0950	OUT010	1998	732.3800	train	15	...	0	1	0	1	0	0	0	0	1	
4	NCD19	8.93	0.057851	Household	53.8614	OUT013	1987	994.7052	train	26	...	1	0	0	0	1	0	0	0	0	

5 rows × 26 columns

In [47]:
```python
import warnings
warnings.filterwarnings('ignore')
#Dropping converted columns:
both.drop(['Item_Type','Outlet_Establishment_Year'],axis=1,inplace=True)

#Dividing into test and train:
train = both.loc[combi['source']=="train"]
test = both.loc[combi['source']=="test"]

#Dropping unnecessary columns:
test.drop(['Item_Outlet_Sales','source'],axis=1,inplace=True)
train.drop(['source'],axis=1,inplace=True)

#Exporting files:
train.to_csv("train_modified.csv",index=False)
test.to_csv("test_modified.csv",index=False)
train2 = pd.read_csv("train_modified.csv")
test2 = pd.read_csv("test_modified.csv")
```

In [48]:
```python
train2.head()
```

Out[48]:

	Item_Identifier	Item_Weight	Item_Visibility	Item_MRP	Outlet_Identifier	Item_Outlet_Sales	Year	Outlet	Item_Fat_Content_0	Item_Fat_Content_1	...	Outlet_Size_0	Outlet_Size_1	Outlet_Size_2	Outlet_Type_0	Outlet_Type_1	Outlet_Type_2	Outlet_Type_3	Item_Type_Combined_0	Item_Type_Combined_1	Item_Type_Combin
0	FDA15	9.30	0.016047	249.8092	OUT049	3735.1380	14	9	1	0	...	0	1	0	0	1	0	0	0	1	
1	DRC01	5.92	0.019278	48.2692	OUT018	443.4228	4	3	0	1	...	0	1	0	0	0	1	0	1	0	
2	FDN15	17.50	0.016760	141.6180	OUT049	2097.2700	14	9	1	0	...	0	1	0	0	1	0	0	0	1	
3	FDX07	19.20	0.057851	182.0950	OUT010	732.3800	15	0	0	1	...	0	1	0	1	0	0	0	0	1	
4	NCD19	8.93	0.057851	53.8614	OUT013	994.7052	26	1	1	0	...	1	0	0	0	1	0	0	0	0	

5 rows × 23 columns

In [56]:
```python
X_train = train2.drop(['Item_Outlet_Sales', 'Outlet_Identifier','Item_Identifier'], axis=1)
y_train = train2.Item_Outlet_Sales
```

In [50]:
```python
#BUILDING MODELS
#LINEAR REGRESSION
from sklearn.linear_model import  LinearRegression
from sklearn.metrics import r2_score
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[50]:
```
LinearRegression()
```

In [51]:
```python
lr_accuracy = round(regressor.score(X_train,y_train) * 100,2)
lr_accuracy
```

Out[51]:
```
56.35
```

In [52]:
```python
from sklearn.metrics import mean_squared_error
print("RMSE : ", np.sqrt(metrics.mean_squared_error(y_train, regressor.predict(X_train))))
```

```
RMSE :  1127.4042036173628
```

In [53]:
```python
#DECISION TREE
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(max_depth=15,min_samples_leaf=300)
regressor.fit(X_train, y_train)
```

Out[53]:
```
DecisionTreeRegressor(max_depth=15, min_samples_leaf=300)
```

In [55]:
```python
tree_accuracy = round(regressor.score(X_train,y_train),2)
tree_accuracy
```

Out[55]:
```
0.59
```

In [57]:
```python
r2_score(y_train, regressor.predict(X_train))
```

Out[57]:
```
0.588335434306813
```

In [58]:
```python
print("RMSE : " , np.sqrt(metrics.mean_squared_error(y_train, regressor.predict(X_train))))
```

```
RMSE :  1094.8445446843184
```

In [59]:
```python
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators=100,max_depth=6, min_samples_leaf=50,n_jobs=4)
regressor.fit(X_train, y_train)
```

Out[59]:
```
RandomForestRegressor(max_depth=6, min_samples_leaf=50, n_jobs=4)
```

In [60]:
```python
rf_accuracy = round(regressor.score(X_train,y_train),2)
rf_accuracy
```

Out[60]:
```
0.61
```

In [61]:
```python
print("RMSE : " ,np.sqrt(metrics.mean_squared_error(y_train, regressor.predict(X_train))))
```

```
RMSE :  1062.381088021117
```

In [ ]: