In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv(r"C:\Users\mani ganesh\Desktop\big_mart_sales_data.csv")
df.head()
```

Out[2]:

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

In [3]:
```
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 [4]:
```
df.select_dtypes(include=['int64','float']).head()
```

Out[4]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
0	9.30	0.016047	249.8092	1999	3735.1380
1	5.92	0.019278	48.2692	2009	443.4228
2	17.50	0.016760	141.6180	1999	2097.2700
3	19.20	0.000000	182.0950	1998	732.3800
4	8.93	0.000000	53.8614	1987	994.7052

In [5]:
```
df.describe()
```

Out[5]:

	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 [6]:
```
df.shape
```

Out[6]:
```
(8523, 12)
```

In [7]:
```
df.isna().sum()
```

Out[7]:
```
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 [8]:
```
df['Item_Weight'].describe()
```

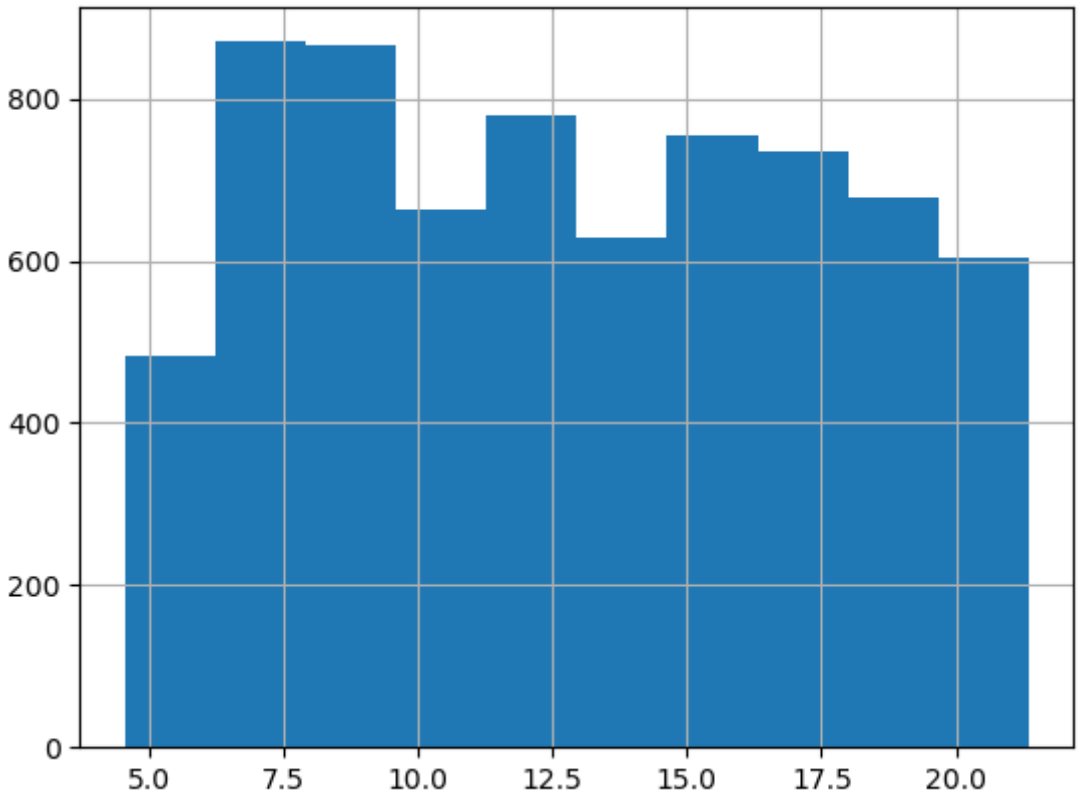
Out[8]:
```
count    7060.000000
mean       12.857645
std         4.643456
min         4.555000
25%         8.773750
50%        12.600000
75%        16.850000
max        21.350000
Name: Item_Weight, dtype: float64
```

In [9]:
```
df.Item_Weight.median()
```

Out[9]:
```
12.6
```

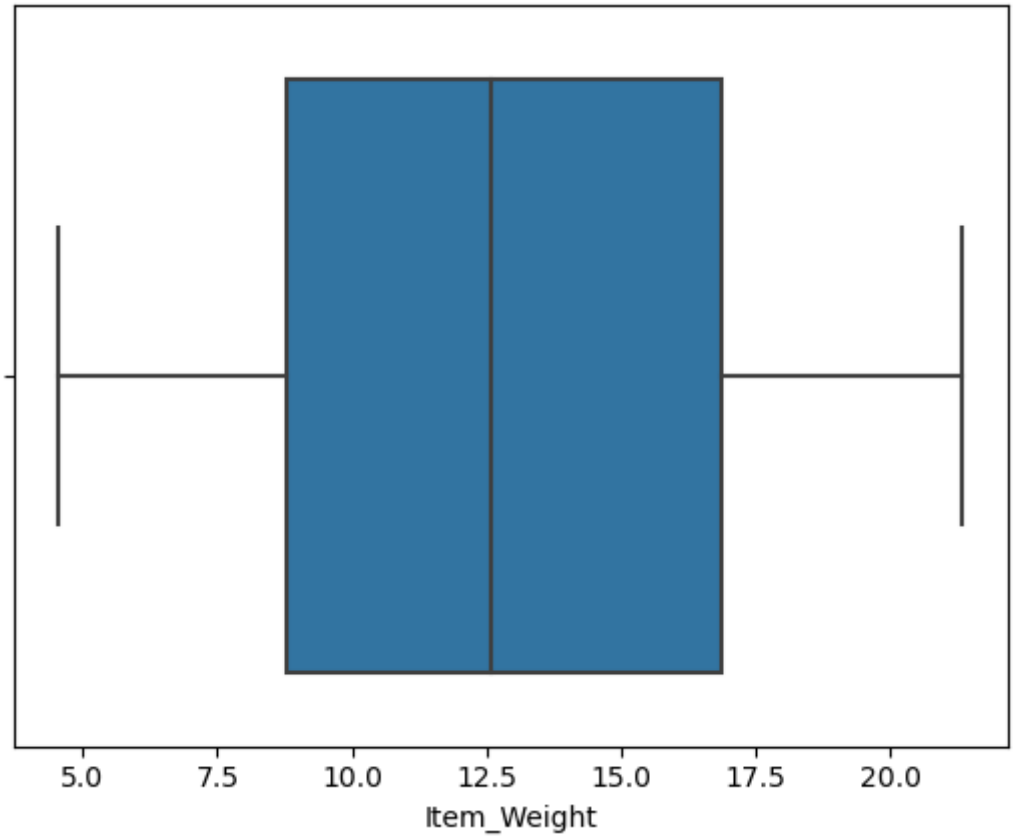
In [10]:
```
df.Item_Weight.hist()
```

Out[10]:
```
<Axes: >
```



Mean, Median, Mode are so close to each other, which can mean that the dataset is of symmetric distribution. So it is safe to impute the missing values with mean. For further check we can use boxplot.

In [11]:
```
sns.boxplot(x = df['Item_Weight'])
plt.show()
```



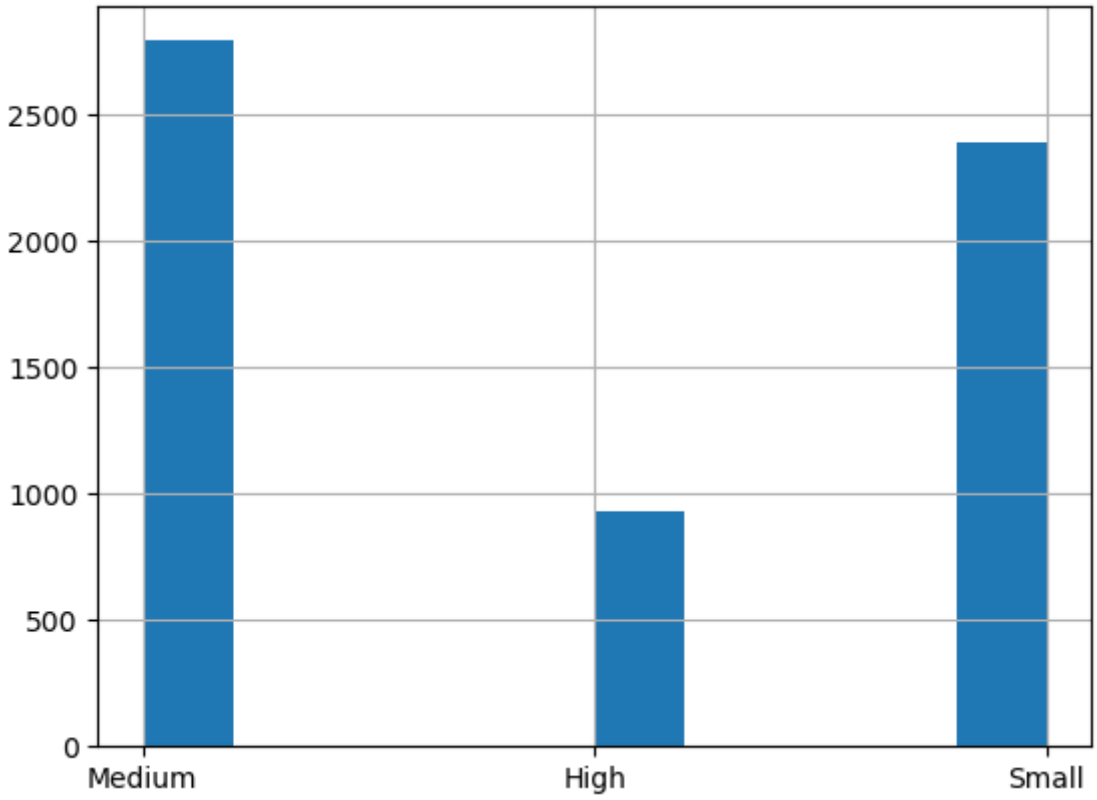
In [12]:
```
df.Item_Weight.fillna(df.Item_Weight.mean(),inplace=True)
```

In [13]:
```
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                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
```

In [14]:
```
df.Outlet_Size.hist()
```

Out[14]: <Axes: >



Hence it categorical values, It is safe to impute them with mode

In [15]:
```
df.Outlet_Size.fillna(df.Outlet_Size.mode()[0],inplace=True)
```

In [16]:
```
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                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
```

Data Cleaning

In [17]:
```
df.Item_Fat_Content.unique()
```

Out[17]:
```
array(['Low Fat', 'Regular', 'low fat', 'LF', 'reg'], dtype=object)
```

In [18]:
```
df.Item_Fat_Content.replace(['low fat','LF'],'Low Fat',inplace=True)
```

In [19]:
```
df.Item_Fat_Content.replace('reg','Regular',inplace=True)
```

In [20]:
```
df.Item_Fat_Content.unique()
```

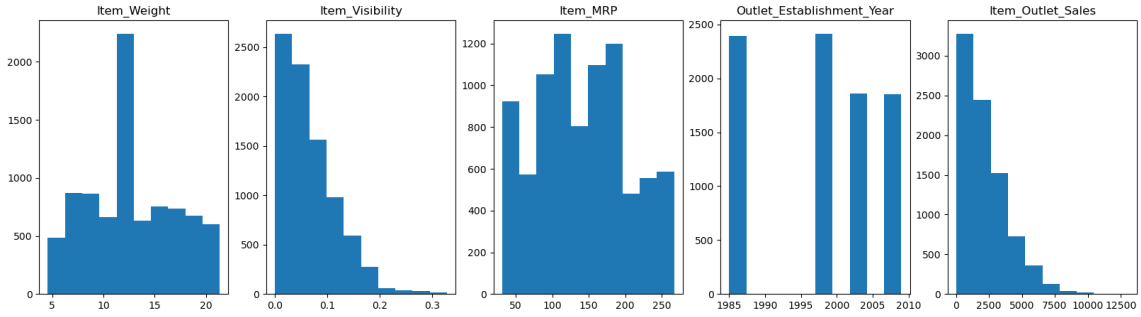
Out[20]:
```
array(['Low Fat', 'Regular'], dtype=object)
```

Data Visualization of Numeric Columns

In [21]:
```
numerics = df.select_dtypes(include=['float64', 'int64']).columns.tolist()
numerics
```

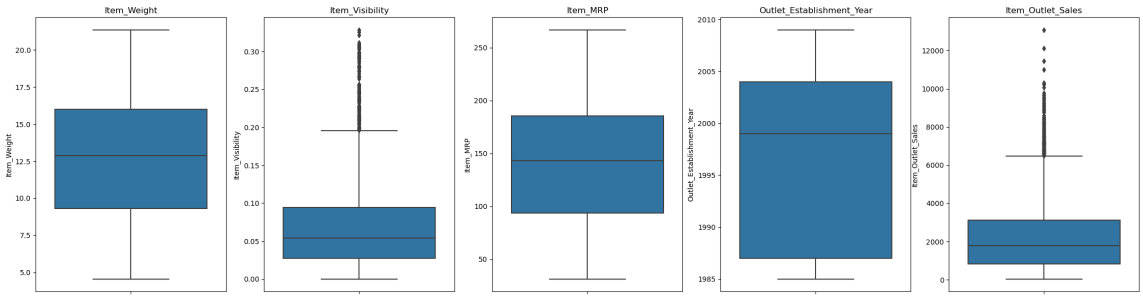
Out[21]:
```
['Item_Weight',
 'Item_Visibility',
 'Item_MRP',
 'Outlet_Establishment_Year',
 'Item_Outlet_Sales']
```

In [22]:
```
fig, ax = plt.subplots(1, 5, figsize=(20, 5))

for i, col in enumerate(numerics):
    ax[i].hist(df[col])
    ax[i].set_title(col)
```



It seems that the Item Visibility and Item outlet sales columns are right skewed!

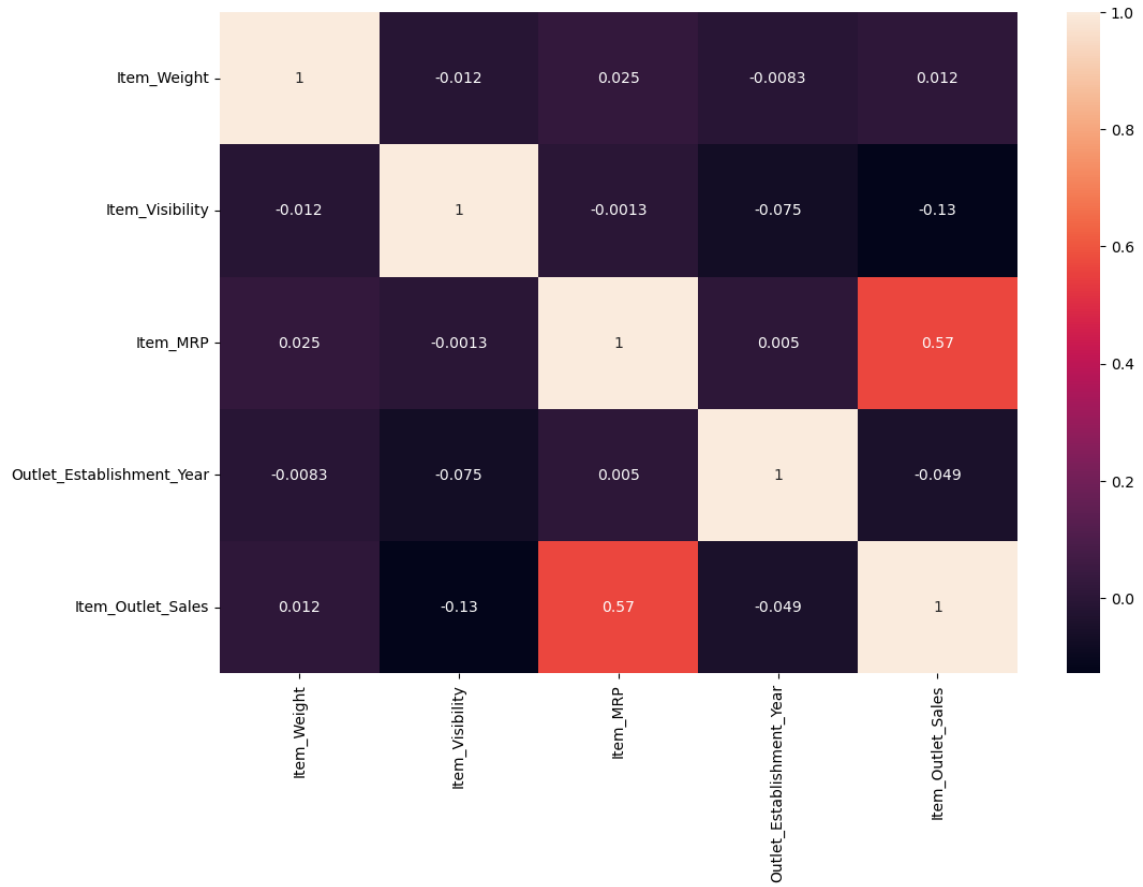
In [23]:
```
fig, ax = plt.subplots(1, 5, figsize=(28, 7))

for i, col in enumerate(numerics):
    sns.boxplot(data=df, y=col, ax=ax[i])
    ax[i].set_title(col)
```



It seems that the Item Visibility and Item outlet sales columns have some outliers which need to be handled!

In [24]:
```
plt.figure(figsize=(12,8))
sns.heatmap(df[numerics].corr(),annot=True)
```

Out[24]: <Axes: >



Data VIsualization of Categorical Columns

Encoding the categorical variables

In [25]:
```
df.select_dtypes(include='object').head()
```

Out[25]:

	Item_Identifier	Item_Fat_Content	Item_Type	Outlet_Identifier	Outlet_Size	Outlet_Location_
0	FDA15	Low Fat	Dairy	OUT049	Medium	
1	DRC01	Regular	Soft Drinks	OUT018	Medium	
2	FDN15	Low Fat	Meat	OUT049	Medium	
3	FDX07	Regular	Fruits and Vegetables	OUT010	Medium	
4	NCD19	Low Fat	Household	OUT013	High	

In [26]:
```
df.select_dtypes(include='object')['Item_Type'].unique()
```

Out[26]:
```
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 [27]:
```
from sklearn.preprocessing import OneHotEncoder
```

In [28]:
```
# Initialize the OneHotEncoder
oh = OneHotEncoder(sparse=False, handle_unknown='ignore')
columns_to_encode = ['Item_Fat_Content','Item_Type','Outlet_Identifier','Ou
edata = oh.fit_transform(df[columns_to_encode])
```

```
C:\Users\mani ganesh\anaconda3\lib\site-packages\sklearn\preprocessing\_en
coders.py:828: FutureWarning: `sparse` was renamed to `sparse_output` in v
ersion 1.2 and will be removed in 1.4. `sparse_output` is ignored unless y
ou leave `sparse` to its default value.
  warnings.warn(
```

In [29]:
```
edata
```

Out[29]:
```
array([[1., 0., 0., ..., 1., 0., 0.],
       [0., 1., 0., ..., 0., 1., 0.],
       [1., 0., 0., ..., 1., 0., 0.],
       ...,
       [1., 0., 0., ..., 1., 0., 0.],
       [0., 1., 0., ..., 0., 1., 0.],
       [1., 0., 0., ..., 1., 0., 0.]])
```

In [30]:
```
d2 = pd.DataFrame(edata, columns=oh.get_feature_names_out(columns_to_encode
d2
```

0	1.0	0.0	0.0
1	0.0	1.0	0.0
2	1.0	0.0	0.0
3	0.0	1.0	0.0
4	1.0	0.0	0.0
...	...	...	...
8518	1.0	0.0	0.0
8519	0.0	1.0	1.0
8520	1.0	0.0	0.0
8521	0.0	1.0	0.0
8522	1.0	0.0	0.0

8523 rows × 38 columns

In [31]:
```
df = pd.concat([df,d2],axis=1)
df
```

3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614
...	...	...	...	...	...	...
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5218
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1224
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1332
8522	DRG01	14.800	Low Fat	0.044878	Soft Drinks	75.4670

8523 rows × 50 columns

In [32]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 50 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
 12  Item_Fat_Content_Low Fat           8523 non-null   float64
 13  Item_Fat_Content_Regular           8523 non-null   float64
 14  Item_Type_Baking Goods             8523 non-null   float64
 15  Item_Type_Breads                   8523 non-null   float64
 16  Item_Type_Breakfast                8523 non-null   float64
 17  Item_Type_Canned                   8523 non-null   float64
 18  Item_Type_Dairy                    8523 non-null   float64
 19  Item_Type_Frozen Foods             8523 non-null   float64
 20  Item_Type_Fruits and Vegetables    8523 non-null   float64
 21  Item_Type_Hard Drinks              8523 non-null   float64
 22  Item_Type_Health and Hygiene       8523 non-null   float64
 23  Item_Type_Household                8523 non-null   float64
 24  Item_Type_Meat                     8523 non-null   float64
 25  Item_Type_Others                   8523 non-null   float64
 26  Item_Type_Seafood                  8523 non-null   float64
 27  Item_Type_Snack Foods              8523 non-null   float64
 28  Item_Type_Soft Drinks              8523 non-null   float64
 29  Item_Type_Starchy Foods            8523 non-null   float64
 30  Outlet_Identifier_OUT010           8523 non-null   float64
 31  Outlet_Identifier_OUT013           8523 non-null   float64
 32  Outlet_Identifier_OUT017           8523 non-null   float64
 33  Outlet_Identifier_OUT018           8523 non-null   float64
 34  Outlet_Identifier_OUT019           8523 non-null   float64
 35  Outlet_Identifier_OUT027           8523 non-null   float64
 36  Outlet_Identifier_OUT035           8523 non-null   float64
 37  Outlet_Identifier_OUT045           8523 non-null   float64
 38  Outlet_Identifier_OUT046           8523 non-null   float64
 39  Outlet_Identifier_OUT049           8523 non-null   float64
 40  Outlet_Size_High                   8523 non-null   float64
 41  Outlet_Size_Medium                 8523 non-null   float64
 42  Outlet_Size_Small                  8523 non-null   float64
 43  Outlet_Location_Type_Tier 1        8523 non-null   float64
 44  Outlet_Location_Type_Tier 2        8523 non-null   float64
 45  Outlet_Location_Type_Tier 3        8523 non-null   float64
 46  Outlet_Type_Grocery Store          8523 non-null   float64
 47  Outlet_Type_Supermarket Type1      8523 non-null   float64
 48  Outlet_Type_Supermarket Type2      8523 non-null   float64
 49  Outlet_Type_Supermarket Type3      8523 non-null   float64
dtypes: float64(42), int64(1), object(7)
memory usage: 3.3+ MB
```

In [33]:
```
df.drop(columns_to_encode,axis=1,inplace=True)
```

In [34]:
```
df.info()
```

```
 27  Outlet_Identifier_OUT018          8523 non-null   float64
 28  Outlet_Identifier_OUT019          8523 non-null   float64
 29  Outlet_Identifier_OUT027          8523 non-null   float64
 30  Outlet_Identifier_OUT035          8523 non-null   float64
 31  Outlet_Identifier_OUT045          8523 non-null   float64
 32  Outlet_Identifier_OUT046          8523 non-null   float64
 33  Outlet_Identifier_OUT049          8523 non-null   float64
 34  Outlet_Size_High                  8523 non-null   float64
 35  Outlet_Size_Medium                8523 non-null   float64
 36  Outlet_Size_Small                 8523 non-null   float64
 37  Outlet_Location_Type_Tier 1       8523 non-null   float64
 38  Outlet_Location_Type_Tier 2       8523 non-null   float64
 39  Outlet_Location_Type_Tier 3       8523 non-null   float64
 40  Outlet_Type_Grocery Store         8523 non-null   float64
 41  Outlet_Type_Supermarket Type1     8523 non-null   float64
 42  Outlet_Type_Supermarket Type2     8523 non-null   float64
 43  Outlet_Type_Supermarket Type3     8523 non-null   float64
dtypes: float64(42), int64(1), object(1)
memory usage: 2.9+ MB
```

In [35]:
```
df.drop(['Item_Identifier'],axis=1,inplace=True)
df.head()
```

Out[35]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales	Item
0	9.30	0.016047	249.8092	1999	3735.1380	
1	5.92	0.019278	48.2692	2009	443.4228	
2	17.50	0.016760	141.6180	1999	2097.2700	
3	19.20	0.000000	182.0950	1998	732.3800	
4	8.93	0.000000	53.8614	1987	994.7052	

5 rows × 43 columns

In [36]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 43 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Item_Weight                     8523 non-null   float64
 1   Item_Visibility                 8523 non-null   float64
 2   Item_MRP                        8523 non-null   float64
 3   Outlet_Establishment_Year       8523 non-null   int64  
 4   Item_Outlet_Sales               8523 non-null   float64
 5   Item_Fat_Content_Low Fat        8523 non-null   float64
 6   Item_Fat_Content_Regular        8523 non-null   float64
 7   Item_Type_Baking Goods          8523 non-null   float64
 8   Item_Type_Breads                8523 non-null   float64
 9   Item_Type_Breakfast             8523 non-null   float64
 10  Item_Type_Canned                8523 non-null   float64
 11  Item_Type_Dairy                 8523 non-null   float64
 12  Item_Type_Frozen Foods          8523 non-null   float64
 13  Item_Type_Fruits and Vegetables 8523 non-null   float64
```

In [37]:
```
df.describe()
```

Out[37]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.857645	0.066132	140.992782	1997.831867	2181.288914
std	4.226124	0.051598	62.275067	8.371760	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	9.310000	0.026989	93.826500	1987.000000	834.247400
50%	12.857645	0.053931	143.012800	1999.000000	1794.331000
75%	16.000000	0.094585	185.643700	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

8 rows × 43 columns

Model Building

In [38]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
```

In [39]:
```
X = df.drop('Item_Outlet_Sales', axis=1)
y = df['Item_Outlet_Sales']
```

In [40]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y)
```

In [41]:
```
def scores(model):
    train_pred = model.predict(X_train)
    print(f'Score on Training dataset = {r2_score(y_train,train_pred)}')
    test_pred = model.predict(X_test)
    print(f'Score on Testing dataset = {r2_score(y_test,test_pred)}')
```

In [42]:
```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
scores(lin_reg)
```

```
Score on Training dataset = 0.5694936243911511
Score on Testing dataset = 0.5432408882832404
```