```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder
from sklearn import metrics
import math
```

```
sales_data = pd.read_csv("train.csv")
sales_data.head()
```

	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

```
sales_data.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
```

```
sales_data.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
```

```
# Imputation
sales_data['Item_Weight'] = sales_data['Item_Weight'].fillna(sales_data['Item_Weight'].mean())
sales_data['Outlet_Size'] = sales_data['Outlet_Size'].fillna(sales_data['Outlet_Size'].mode()[0])
sales_data.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
```

```
sales_data.head()
```

	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	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	Medium	Tier 3	Grocery Store

```
sales_data['Item_Fat_Content'].value_counts()
```

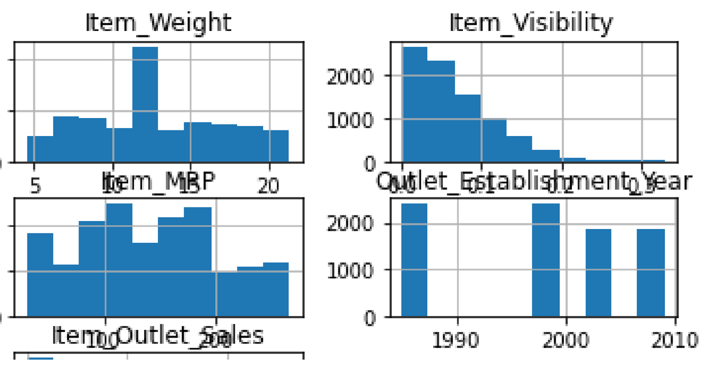
```
Low Fat    5089
Regular    2889
LF          316
reg         117
low fat     112
Name: Item_Fat_Content, dtype: int64
```

```
sales_data.describe()
```

	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

```
sales_data.hist()
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fee83d442d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fee83d19910>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fee83cd1f10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fee83c92550>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fee83c4ab50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fee83c0c190>]],
      dtype=object)
```




```
# Low Fat, LF and low fat are all same and Regular and reg are same so we need to combine them.
sales_data.replace({'Item_Fat_Content':{'low fat':'Low Fat','LF':'Low Fat','reg':'Regular'}},inplace=True)
sales_data['Item_Fat_Content'].value_counts()
```

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

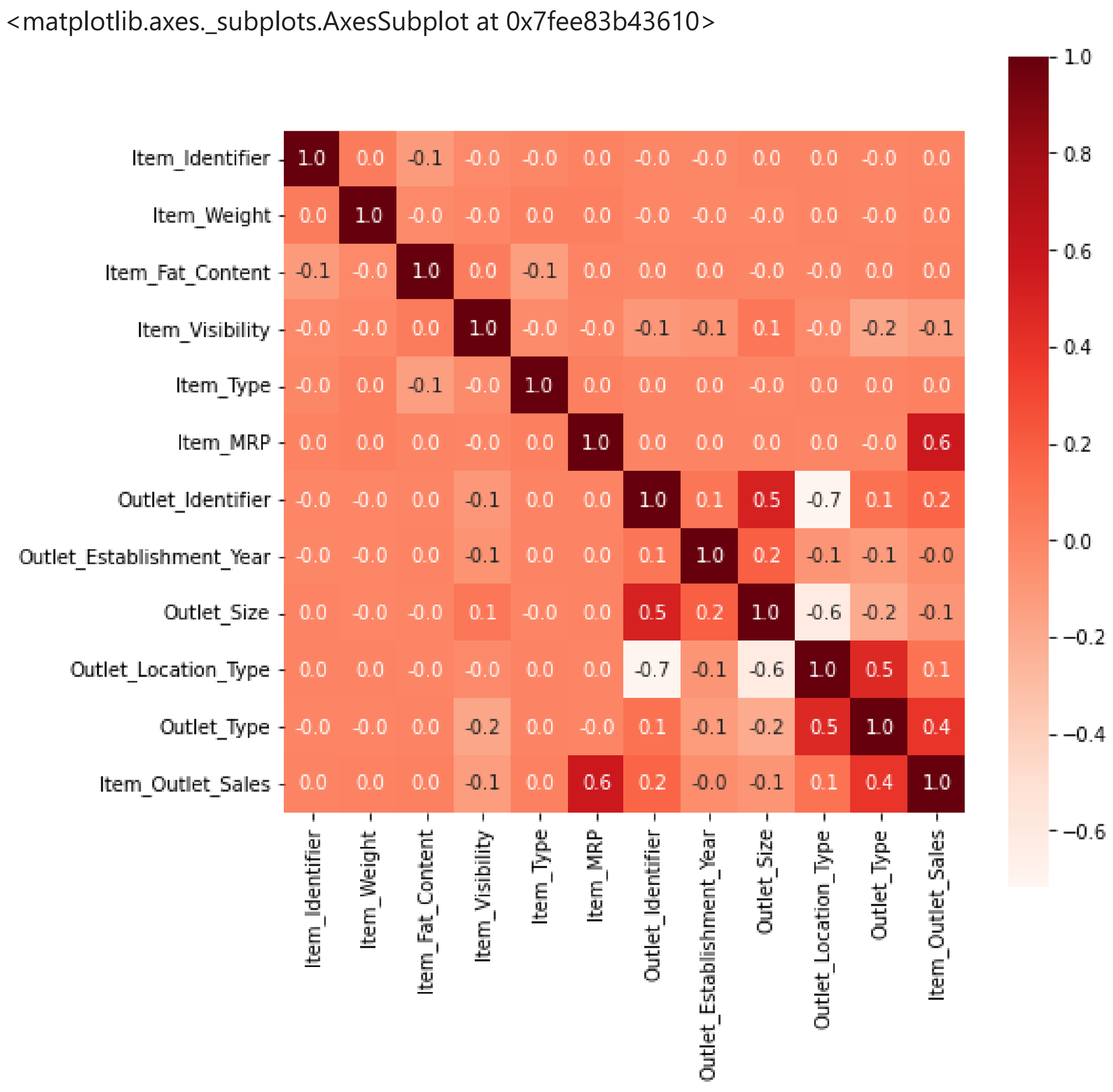
```
encoder = LabelEncoder()
sales_data['Item_Identifier'] = encoder.fit_transform(sales_data['Item_Identifier'])
sales_data['Item_Fat_Content'] = encoder.fit_transform(sales_data['Item_Fat_Content'])
sales_data['Item_Type'] = encoder.fit_transform(sales_data['Item_Type'])
# le_name_mapping = dict(zip(encoder.classes_, encoder.transform(encoder.classes_)))
# print(le_name_mapping)
sales_data['Outlet_Identifier'] = encoder.fit_transform(sales_data['Outlet_Identifier'])
sales_data['Outlet_Size'] = encoder.fit_transform(sales_data['Outlet_Size'])
sales_data['Outlet_Location_Type'] = encoder.fit_transform(sales_data['Outlet_Location_Type'])
sales_data['Outlet_Type'] = encoder.fit_transform(sales_data['Outlet_Type'])
```

```
sales_data.head()
```

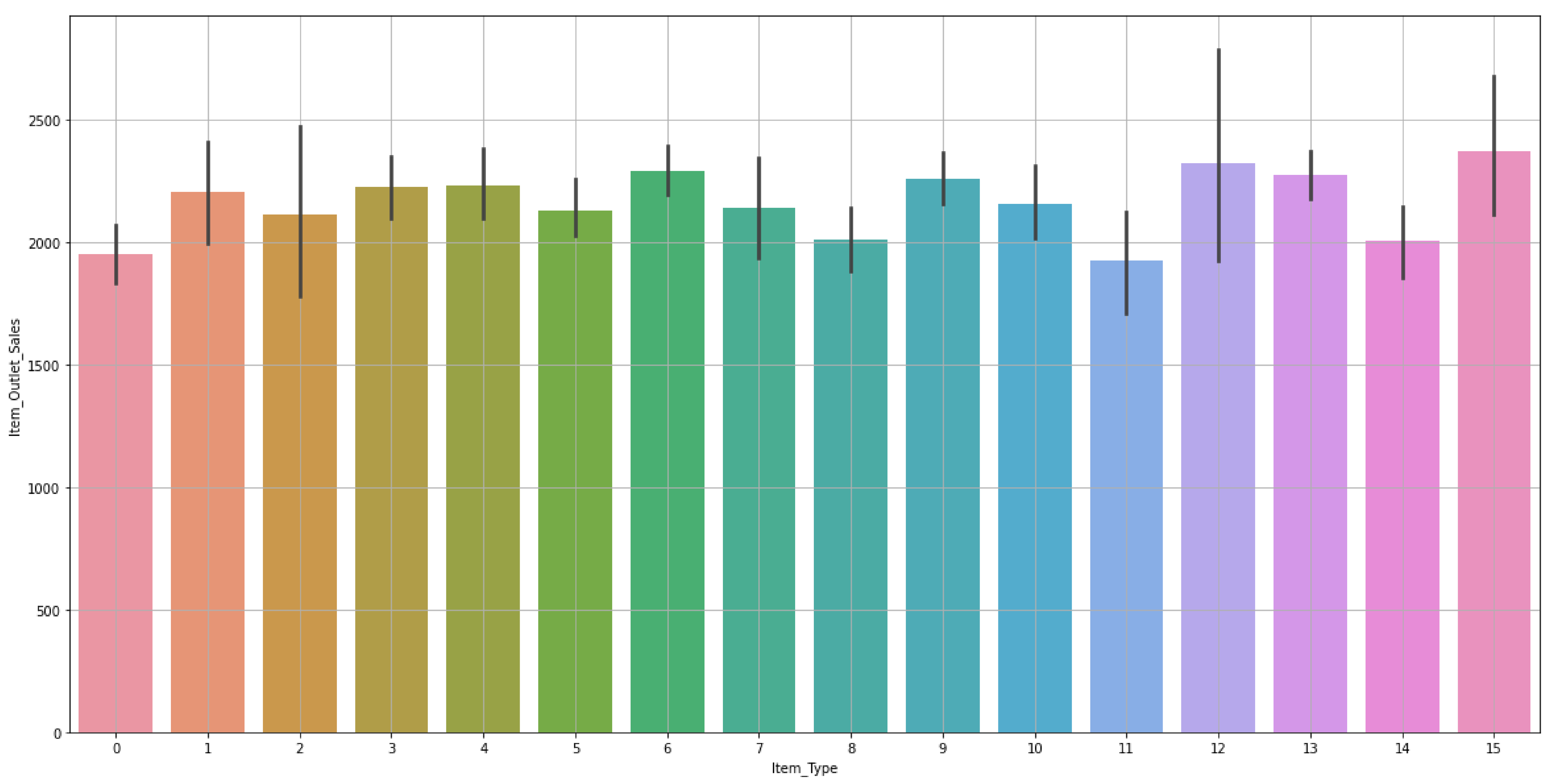
	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type
0	156	9.30	0	0.016047	4	249.8092	9	1999	1	0	1
1	8	5.92	1	0.019278	14	48.2692	3	2009	1	2	2
2	662	17.50	0	0.016760	10	141.6180	9	1999	1	0	1
3	1121	19.20	1	0.000000	6	182.0950	0	1998	1	2	0
4	1297	8.93	0	0.000000	9	53.8614	1	1987	0	2	1

```
corr = sales_data.corr()
plt.figure(figsize=(8,8))
sns.heatmap(corr,cbar=True,square=True,fmt='.1f',annot=True,cmap='Reds')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fee83b43610>
```

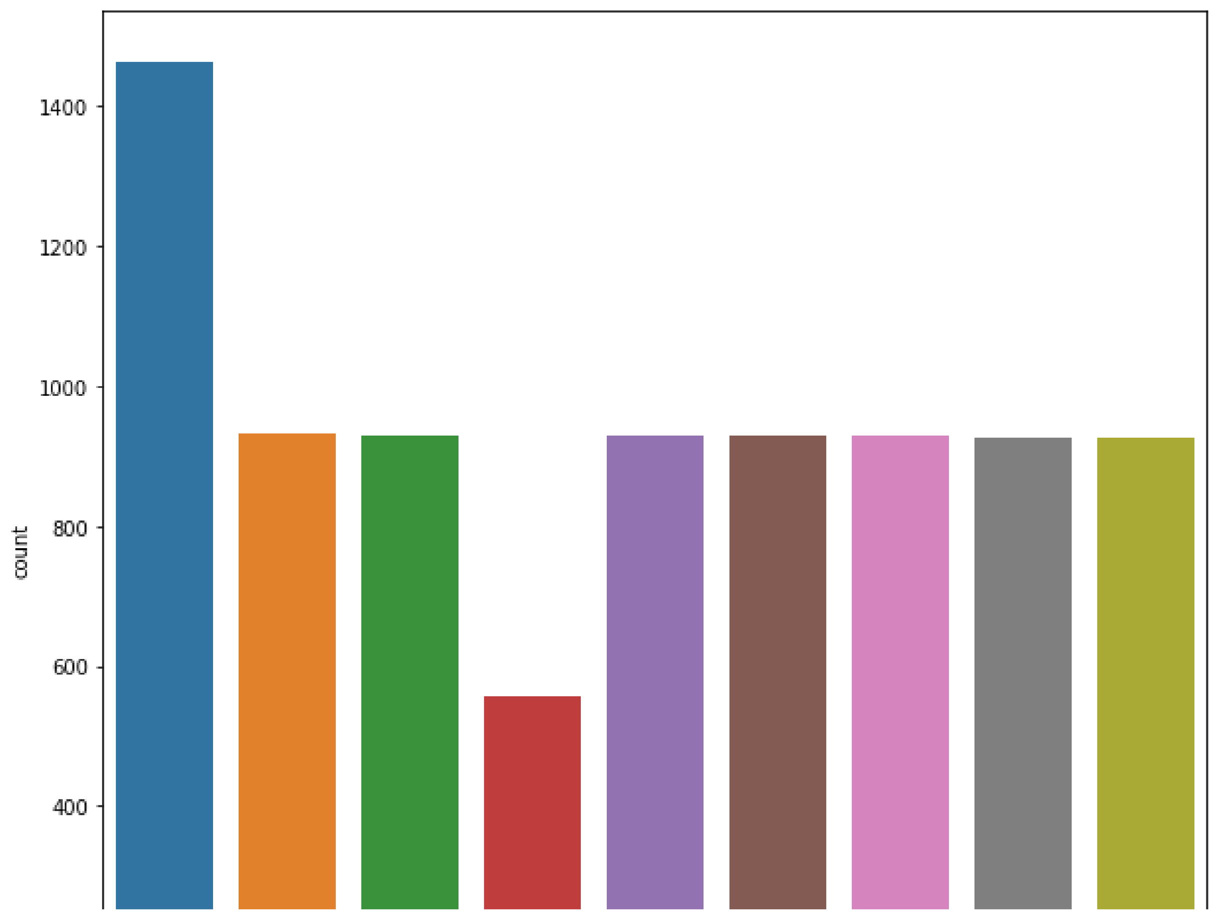



```
plt.figure(figsize=(20,10))
sns.barplot(x='Item_Type',y='Item_Outlet_Sales',data=sales_data)
plt.grid()
```



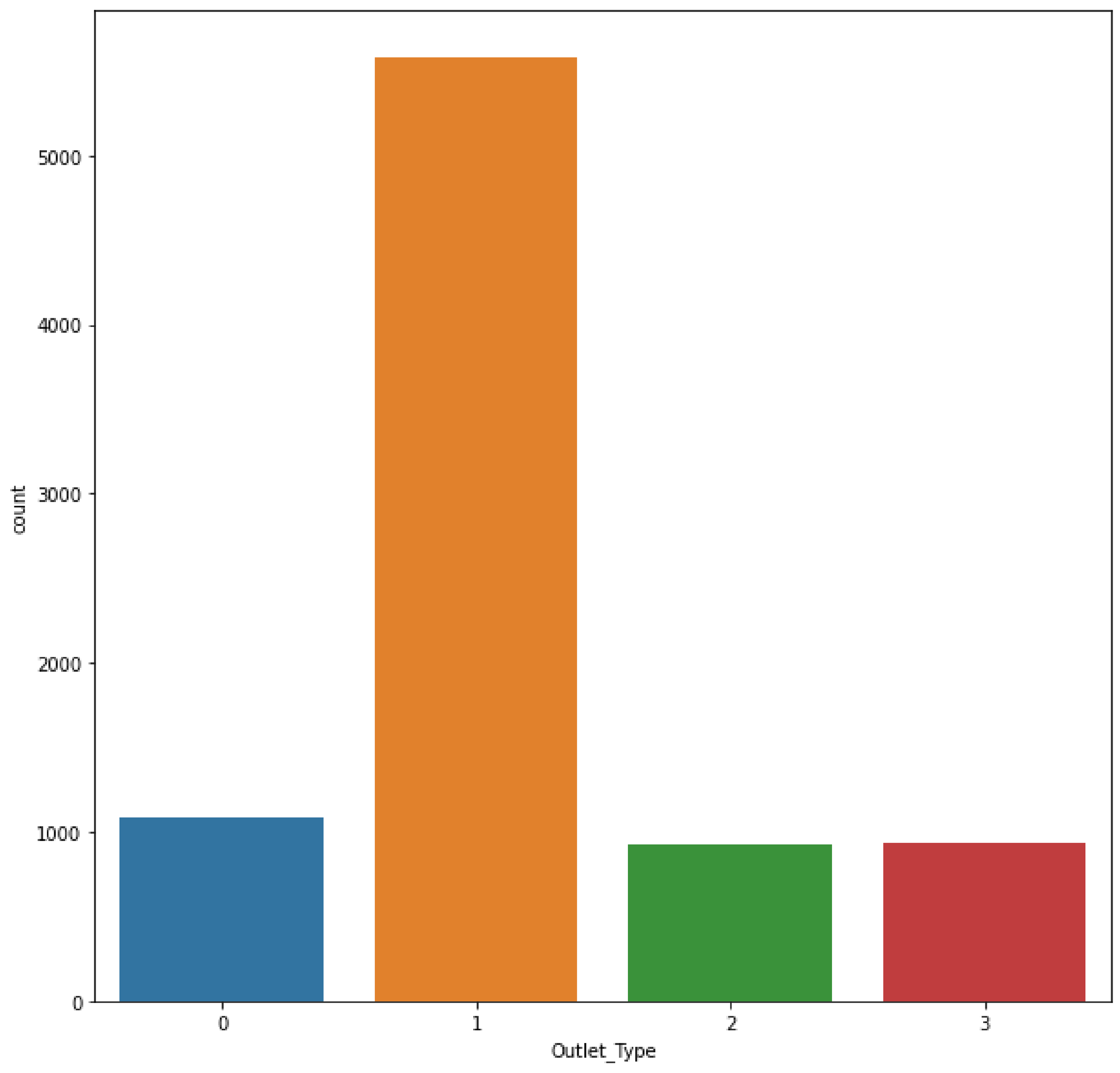

```
plt.figure(figsize=(10,10))
sns.countplot(x="Outlet_Establishment_Year", data=sales_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fee79ab4210>
```




```
plt.figure(figsize=(10,10))
sns.countplot(x="Outlet_Type", data=sales_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fee8786a8d0>
```




```
# We need to split the data
X = sales_data.drop(columns='Item_Outlet_Sales',axis=1) # We need all the variables (columns) as independent variables so we're just dropping the target column to make things easier.
y = sales_data['Item_Outlet_Sales'] # Target

# Then we split the data into training and testing data
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2, random_state = 2) # 80% data will be used for training the model and rest 20% for testing.
```

```
print(X.shape,X_train.shape)
```

```
(8523, 11) (6818, 11)
```

```
def RegressionAlgorithm(func, **kwargs):
    def innerFunction():
        model = func(**kwargs)
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        rmse = math.sqrt(metrics.mean_squared_error(y_test, y_pred))
        print("Root Mean squared error: %.2f" % rmse)
        print('R2 score: %.2f' % metrics.r2_score(y_test, y_pred))
    return innerFunction
```

```
RegressionAlgorithm(LinearRegression)()
```

```
Root Mean squared error: 1248.61
R2 score: 0.49
```

```
from sklearn.ensemble import RandomForestRegressor
RegressionAlgorithm(RandomForestRegressor,n_estimators=100)()
```

```
Root Mean squared error: 1176.34
R2 score: 0.55
```

```
from sklearn.ensemble import RandomForestRegressor
RegressionAlgorithm(RandomForestRegressor,n_estimators=100)()
```

```
Root Mean squared error: 1176.48
R2 score: 0.55
```

✓ 3s completed at 16:53