Technocolabs Data Analysis Internship

AIM:

The aim of this data science project is to build a predictive model and find out the sales of each product at a particular store.

Problem Statement:

The data scientists at BigMart have collected 2013 sales data for 1559 products across 10 stores in different cities. Also, certain attributes of each product and store have been defined. Using this model, BigMart will try to understand the properties of products and stores which play a key role in increasing sales. The data has missing values as some stores do not report all the data due to technical glitches. Hence, it will be required to treat them accordingly.

DATASET:

Train file: CSV containing the item outlet information with sales value. This dataset contains train (8523,12) rows and columns.

Test file: CSV containing item outlet combinations for which sales need to be forecasted. This Dataset contains test (5681,11) rows and columns that are needed to predict the sales for the test data set.

Importing Datasets

In [318]:

```
train = pd.read_csv(r"C:\Users\jaskeerat singh\Desktop\Data\Train.csv")
test = pd.read_csv(r"C:\Users\jaskeerat singh\Desktop\Data\Test.csv")
```

In [319]:

```
train["source"] = "training"
test["source"] = "testing"
join = pd.concat([train,test], ignore_index=True)
```

In [320]:

```
join.to_csv(r"C:\Users\jaskeerat singh\Desktop\Data\clean.csv")
```

Data Pre - Processing:

Now we try to impute missing values by different methods like by replacing nan values with either mean or mode. And dropping the unwanted columns.

In [321]:

```
df = pd.read_csv(r"C:\Users\jaskeerat singh\Desktop\Data\clean.csv")
```

In [322]:

```
df.drop("Unnamed: 0", axis = 1, inplace = True)
df['Item_Fat_Content'] = df['Item_Fat_Content'].replace({"LF":"Low Fat","reg":"Regular",
"low fat":"Low Fat"})
```

In [324]:

```
df["Item_Weight"].fillna(df["Item_Weight"].mean(), inplace = True)
df["Outlet_Size"].fillna("NO INFO", inplace = True)
df["Item_Outlet_Sales"].fillna(df["Item_Outlet_Sales"].mean(), inplace = True)
```

In [325]:

```
df.isnull().sum()
```

Out[325]:

```
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
```

Data Info:

In [329]:

```
df.shape
```

Out[329]:

```
(14204, 13)
```

In [330]:

```
df.columns
```

Out[330]:

```
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', 'source'],
      dtype='object')
```

In [331]:

```
df.dtypes
```

Out[331]:

```
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
source                        object
dtype: object
```

In [333]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14204 entries, 0 to 14203
Data columns (total 13 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            14204 non-null  object
 1   Item_Weight                14204 non-null  float64
 2   Item_Fat_Content           14204 non-null  object
 3   Item_Visibility            14204 non-null  float64
 4   Item_Type                  14204 non-null  object
 5   Item_MRP                   14204 non-null  float64
 6   Outlet_Identifier          14204 non-null  object
 7   Outlet_Establishment_Year  14204 non-null  int64
 8   Outlet_Size                14204 non-null  object
 9   Outlet_Location_Type       14204 non-null  object
 10  Outlet_Type                14204 non-null  object
 11  Item_Outlet_Sales          14204 non-null  float64
 12  source                     14204 non-null  object
dtypes: float64(4), int64(1), object(8)
memory usage: 1.4+ MB
```

In [334]:

```
df.describe()
```

Out[334]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	14204.000000	14204.000000	14204.000000	14204.000000	14204.000000
mean	12.792854	0.065953	141.004977	1997.830681	2181.288914
std	4.234226	0.051459	62.086938	8.371664	1321.864430
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	9.300000	0.027036	94.012000	1987.000000	1468.089000
50%	12.792854	0.054021	142.247000	1999.000000	2181.288914
75%	16.000000	0.094037	185.855600	2004.000000	2181.288914
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

Exploratory Data Analysis:

1) Univariate Analysis

In the univariate analysis, we try to understand how each variable/feature has the influence on the target variable and get to know whether the input variables really impact the output variable or not.

Distributions Of Items According to Fat Content

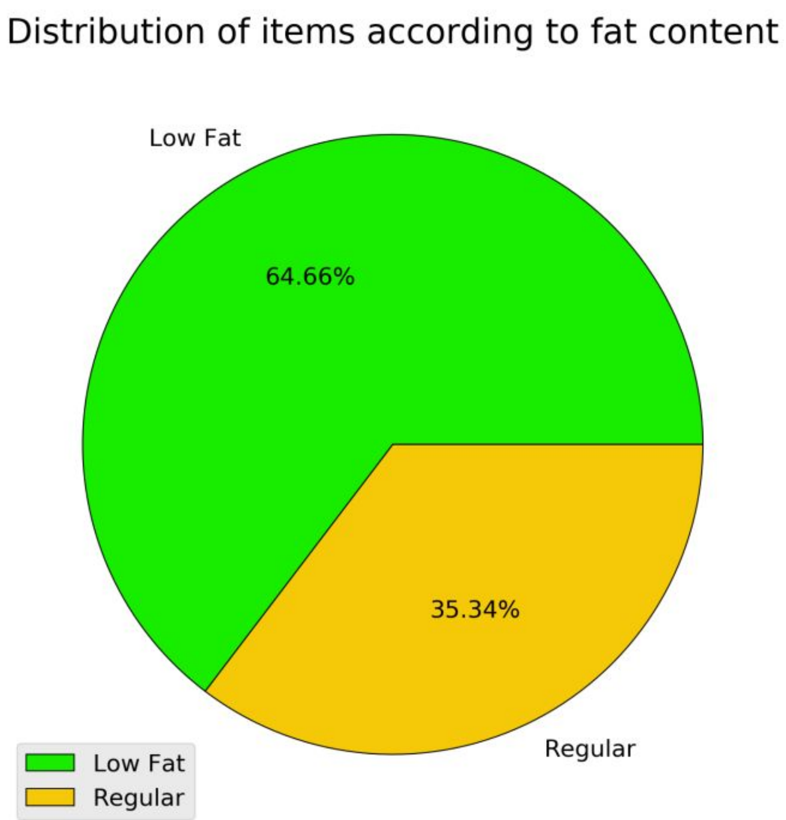
In [335]:

```
print("Count of Items with different Fat Content")
df1 = df.groupby("Item_Fat_Content")["Item_Identifier"].count().to_frame()
df1.rename(columns = {"Item_Identifier":"Total count"}, inplace = True)
df1
```

```
Count of Items with different Fat Content
```

Out[335]:

Item_Fat_Content	Total count
Low Fat	9185
Regular	5019



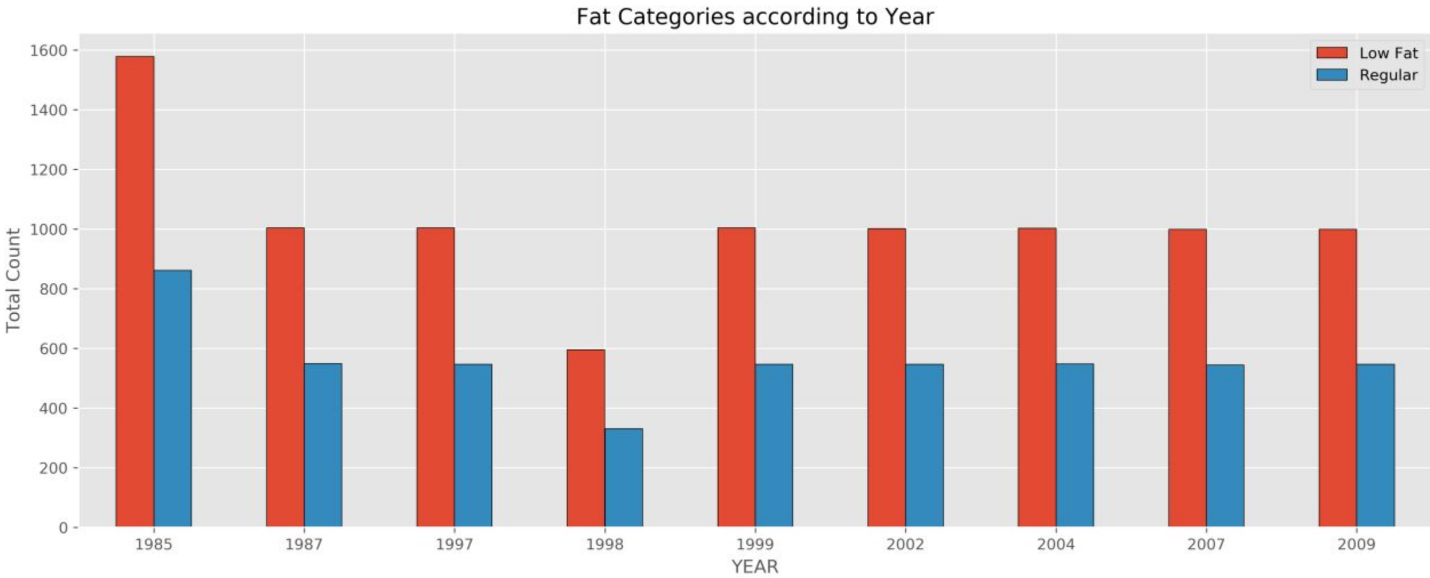
Items According to Year

In [337]:

```
print("Fat Categories according to Year")
df2 = df.groupby(["Outlet_Establishment_Year","Item_Fat_Content"])["Item_Identifier"].co
unt().to_frame()
df2 = df2.unstack()
df2
```

In [338]:

```
df2.plot.bar(stacked = False, edgecolor = "#000000")
plt.legend(["Low Fat","Regular"])
plt.ylabel("Total Count")
plt.xlabel("YEAR")
plt.xticks(rotation = "horizontal")
plt.title("Fat Categories according to Year")
plt.savefig(r"C:\Users\jaskeerat singh\Desktop\Data\figures\Fat Categories according to Y
ear.png", dpi=300, bbox_inches='tight')
plt.show()
```



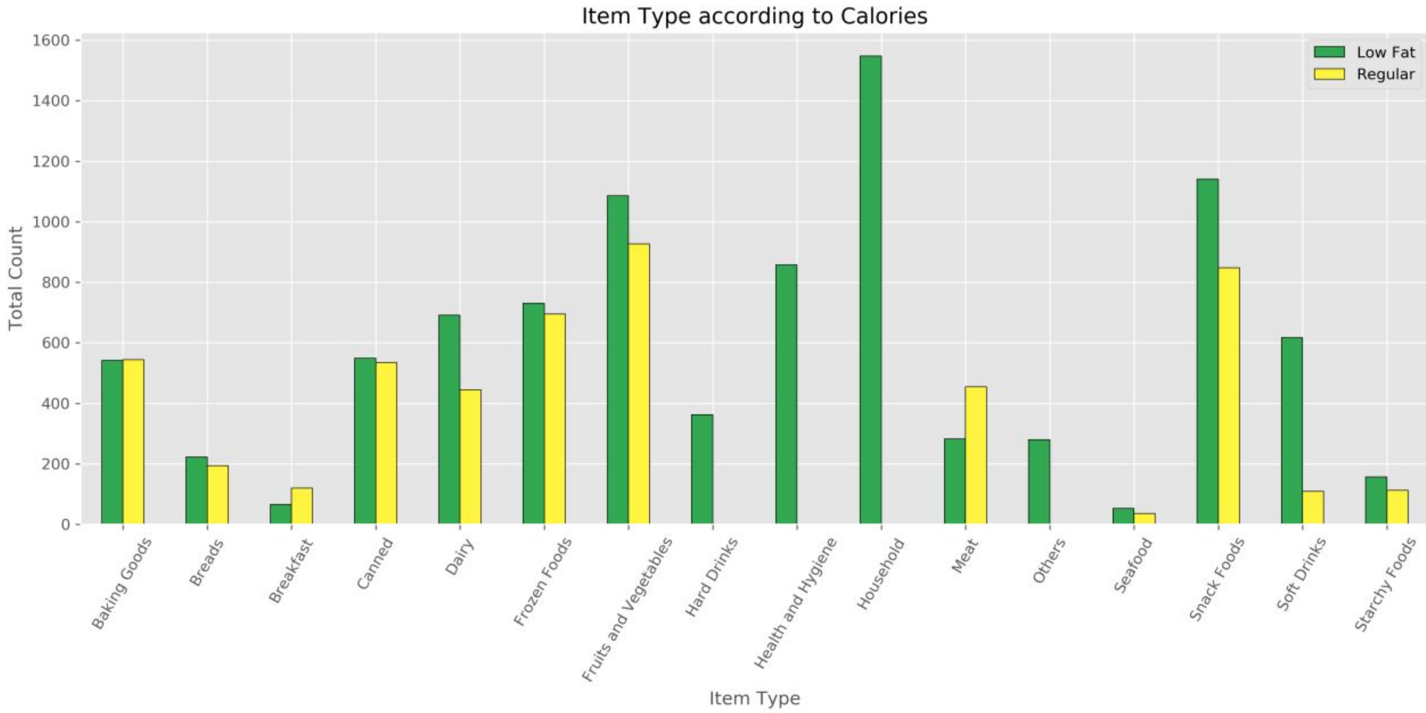
Item Type According To Fat Content

In [339]:

```
df3 = df.groupby(["Item_Type","Item_Fat_Content"])["Item_Identifier"].count().to_frame()
.unstack()
df3
```

In [340]:

```
df3.plot.bar(stacked = False, edgecolor = "#000000", color = ["#32a852","#fff540"])
plt.xlabel("Item Type")
plt.ylabel("Total Count")
plt.title("Item Type according to Calories")
plt.legend(["Low Fat","Regular"])
plt.xticks(rotation = 60)
plt.savefig(r"C:\Users\jaskeerat singh\Desktop\Data\figures\Item Type according to Calori
es.png", dpi=300, bbox_inches='tight')
plt.show()
```



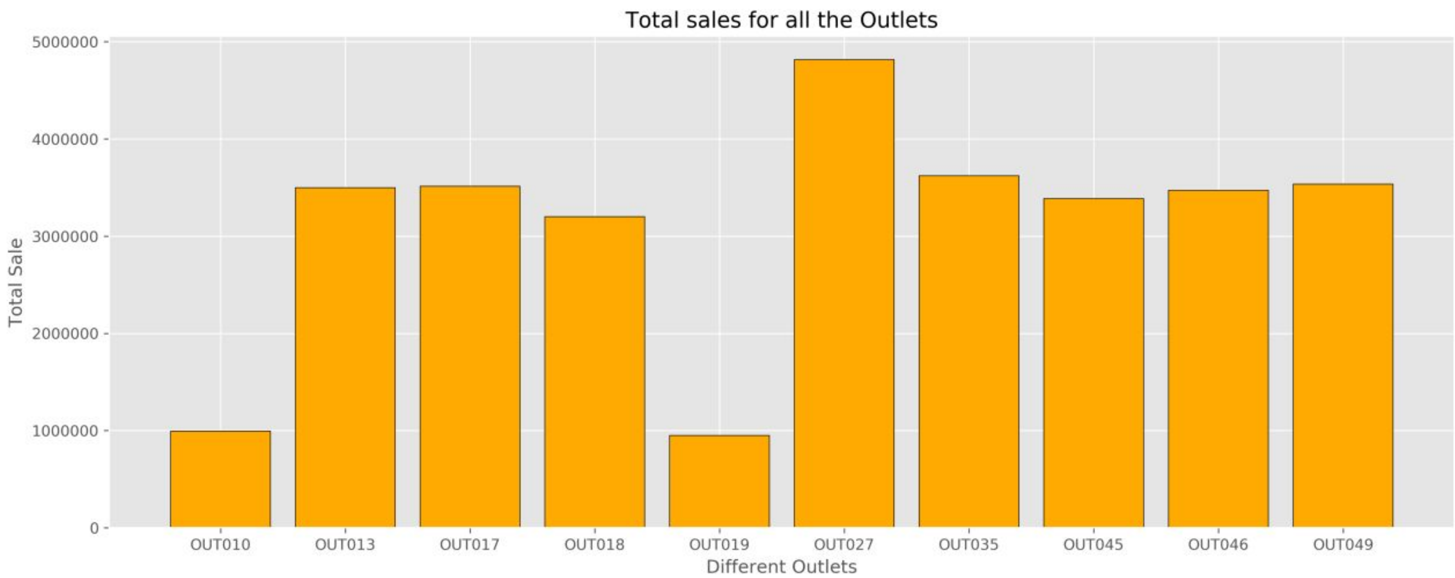
Total Sales For Different Outlets

In [341]:

```
print("Total sales for all the Outlets:- ")
df4 = df.groupby("Outlet_Identifier")["Item_Outlet_Sales"].sum().to_frame()
df4
```

In [342]:

```
plt.bar(df4.index, df4["Item_Outlet_Sales"], color = "#ffaa00", edgecolor = "#000000")
plt.xlabel("Different Outlets")
plt.ylabel("Total Sale")
plt.title("Total sales for all the Outlets")
plt.savefig(r"C:\Users\jaskeerat singh\Desktop\Data\figures\Total sales for all the Outle
ts.png", dpi=300, bbox_inches='tight')
plt.show()
```



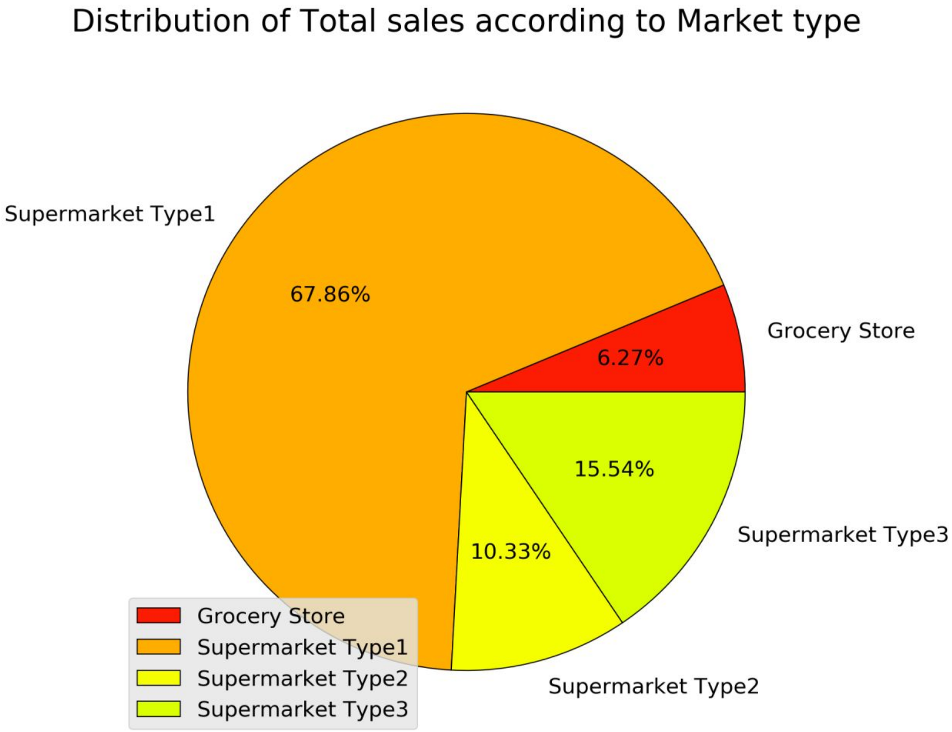
% Distribution According to Market Type

In [343]:

```
print("Sales for different Outlet Type:-")
df5 = df.groupby("Outlet_Type")["Item_Outlet_Sales"].sum().to_frame()
df5
```

In [344]:

```
plt.pie(df5, labels = df5.index, colors = ["#fc1c03","#ffae00","#f6ff00","#d9ff00"], auto
pct = "%1.2f%%", wedgeprops={"edgecolor":"Black"})
plt.title("Distribution of Total sales according to Market type")
plt.legend(loc = "lower left")
plt.savefig(r"C:\Users\jaskeerat singh\Desktop\Data\figures\Distribution of Total sales a
ccording to Market type.png", dpi=300, bbox_inches='tight')
plt.show()
```



Sales For Different Items According To Stores

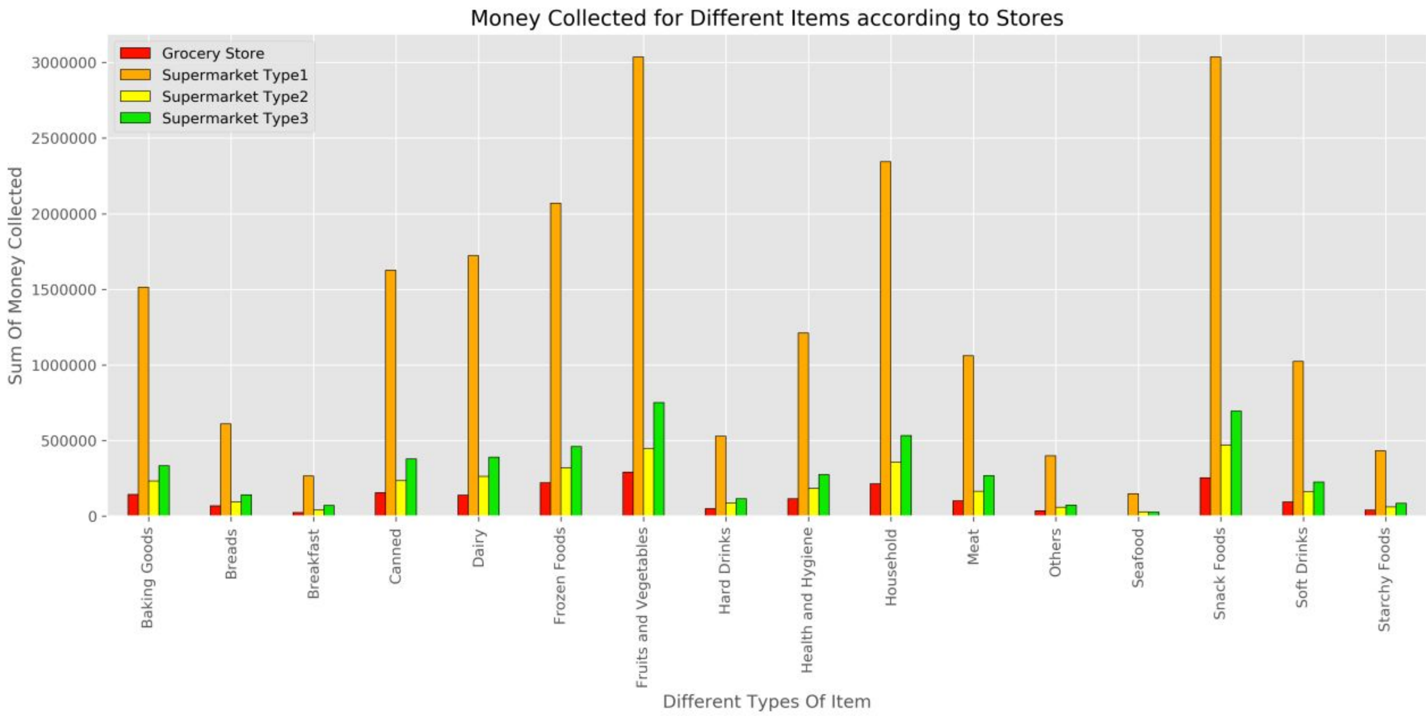
In [345]:

```
print("Money Collected for Different Items according to Stores:-")
df6 = df.groupby(["Item_Type","Outlet_Type"])["Item_Outlet_Sales"].sum().to_frame().unst
ack()
df6
```

In [346]:

```
df6.plot.bar(stacked = False, edgecolor = "#000000", color = ["#ff1100","#ffaa00","#ffff
```

```
00","#0fe800"])
plt.xlabel("Different Types Of Item")
plt.ylabel("Sum Of Money Collected")
plt.title("Money Collected for Different Items according to Stores")
plt.legend(["Grocery Store","Supermarket Type1","Supermarket Type2","Supermarket Type3"]
)
plt.savefig(r"C:\Users\jaskeerat singh\Desktop\Data\figures\Money Collected for Different
Items according to Stores.png", dpi=300, bbox_inches='tight')
plt.show()
```



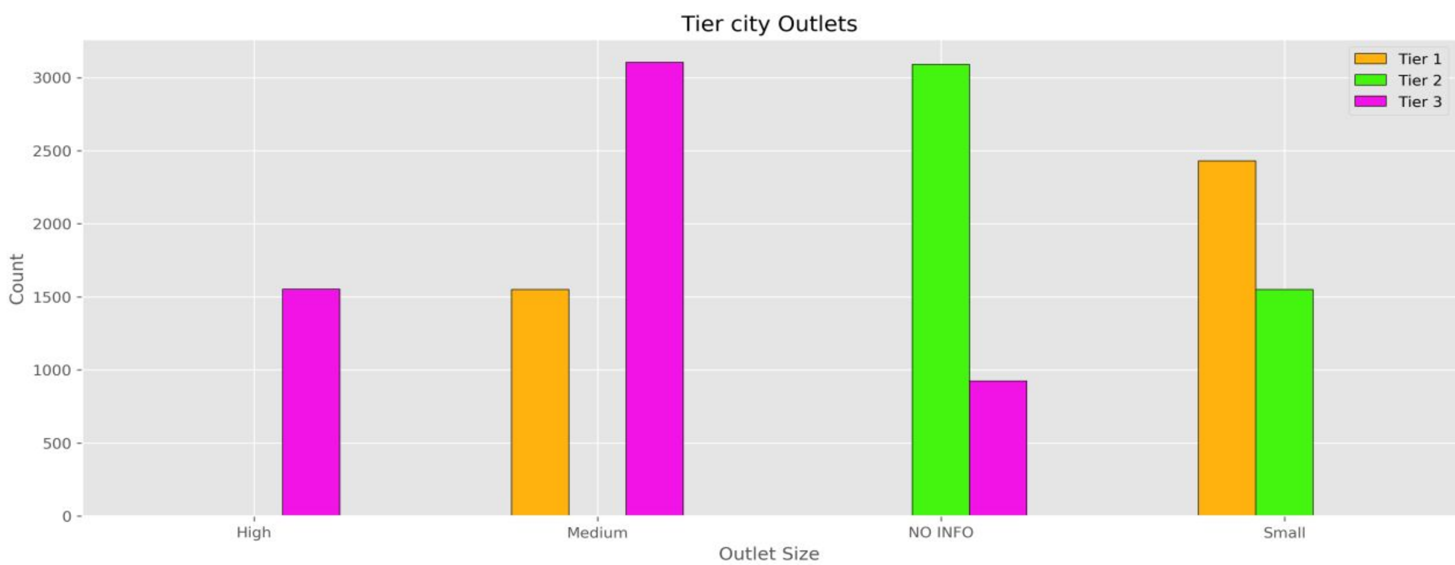
Total Items Sold According to Tier Size

In [351]:

```
df9 = df.groupby(["Outlet_Size","Outlet_Location_Type"])["Item_Identifier"].count().to_f
rame().unstack()
df9
```

In [352]:

```
df9.plot.bar(stacked = False, color = ["#ffb20d","#44f50f","#f213e7"], edgecolor = "#000
000")
plt.legend(["Tier 1","Tier 2","Tier 3"])
plt.xlabel("Outlet Size")
plt.ylabel("Count")
plt.title("Tier city Outlets")
plt.xticks(rotation = "horizontal")
plt.savefig(r"C:\Users\jaskeerat singh\Desktop\Data\figures\Tier city Outlets.png", dpi=3
00, bbox_inches='tight')
plt.show()
```

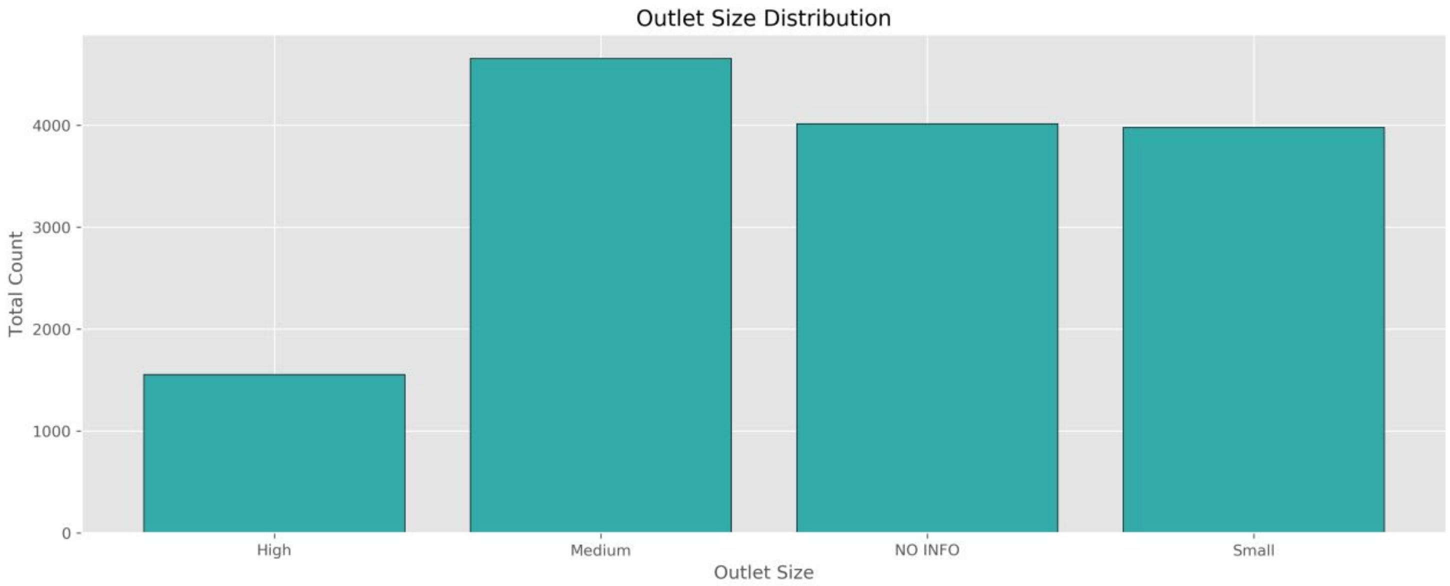


Total Items Sold According To Outlet Size

In [357]:

```
df13 = df.groupby("Outlet_Size").size().to_frame()
df13

plt.bar(df13.index, df13[0], color = "#33aba9", edgecolor = "#000000")
plt.xlabel("Outlet Size")
plt.ylabel("Total Count")
plt.title("Outlet Size Distribution")
plt.savefig(r"C:\Users\jaskeerat singh\Desktop\Data\figures\outlet size.png", dpi=300, bb
ox_inches='tight')
plt.show()
```

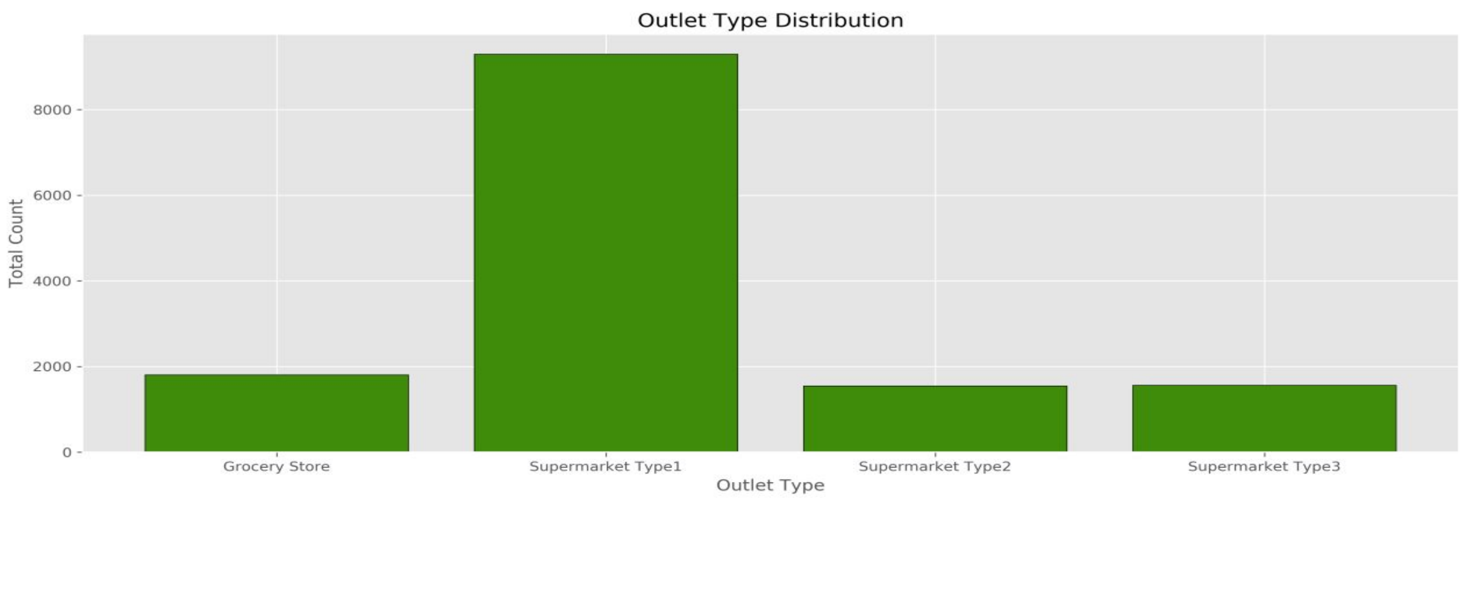


Total Items Sold According To Outlet Type

In [358]:

```
df14 = df.groupby("Outlet_Type").size().to_frame()
df14

plt.bar(df14.index, df14[0], color = "#3f8c0b", edgecolor = "#000000")
plt.xlabel("Outlet Type")
plt.ylabel("Total Count")
plt.title("Outlet Type Distribution")
plt.savefig(r"C:\Users\jaskeerat singh\Desktop\Data\figures\outlet type.png", dpi=300, bb
ox_inches='tight')
plt.show()
```

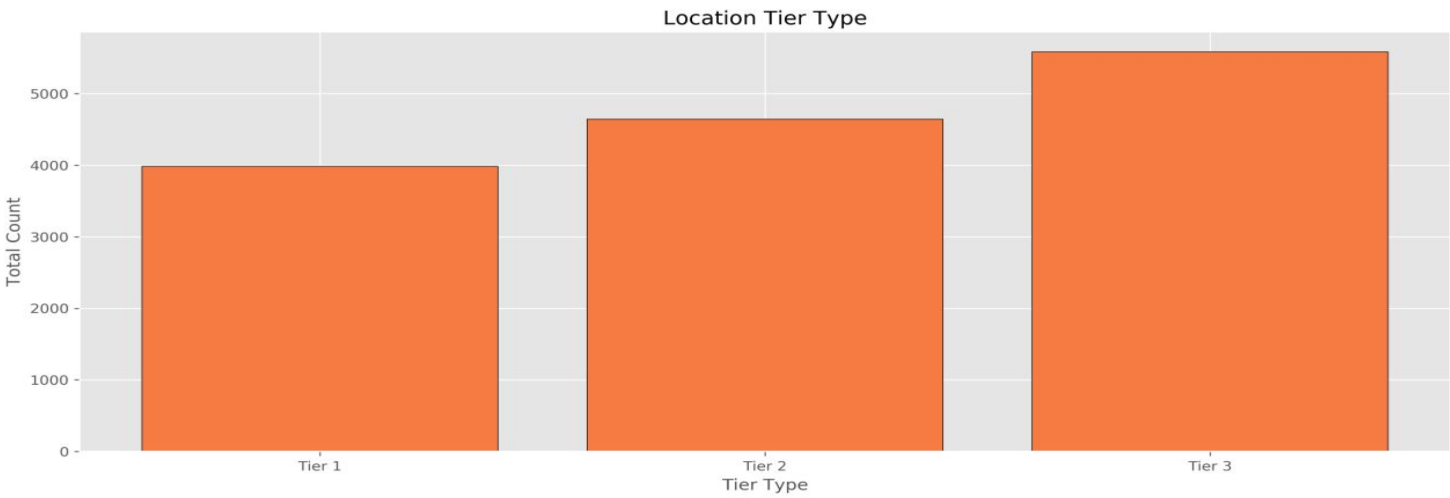


Total Items Sold According to Tier Type

In [359]:

```
df15 = df.groupby("Outlet_Location_Type").size().to_frame()
df15

plt.bar(df15.index, df15[0], color = "#f57b42", edgecolor = "#000000")
plt.xlabel("Tier Type")
plt.ylabel("Total Count")
plt.title("Location Tier Type")
plt.savefig(r"C:\Users\jaskeerat singh\Desktop\Data\figures\tier type.png", dpi=300, bbox
_inches='tight')
plt.show()
```



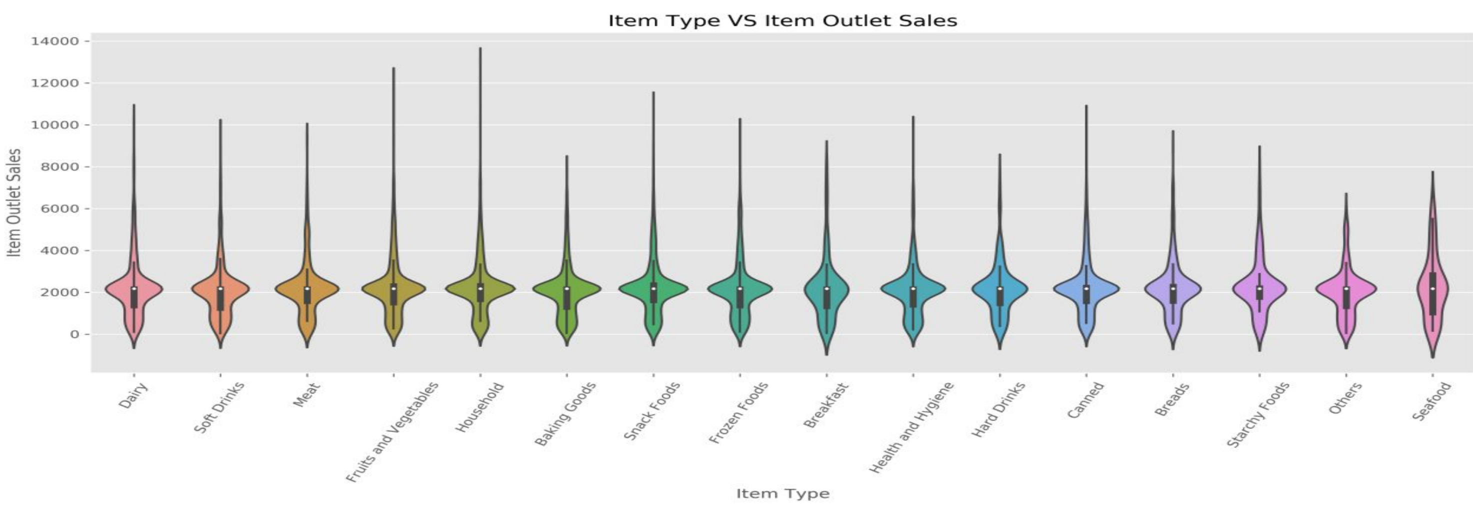
2) Bivariate Analysis

In the bivariate analysis, we will analyze how different features/variables are related to each other and how much impact they do have between them.

Violin Plot for Item Type VS Outlet Sales

In [360]:

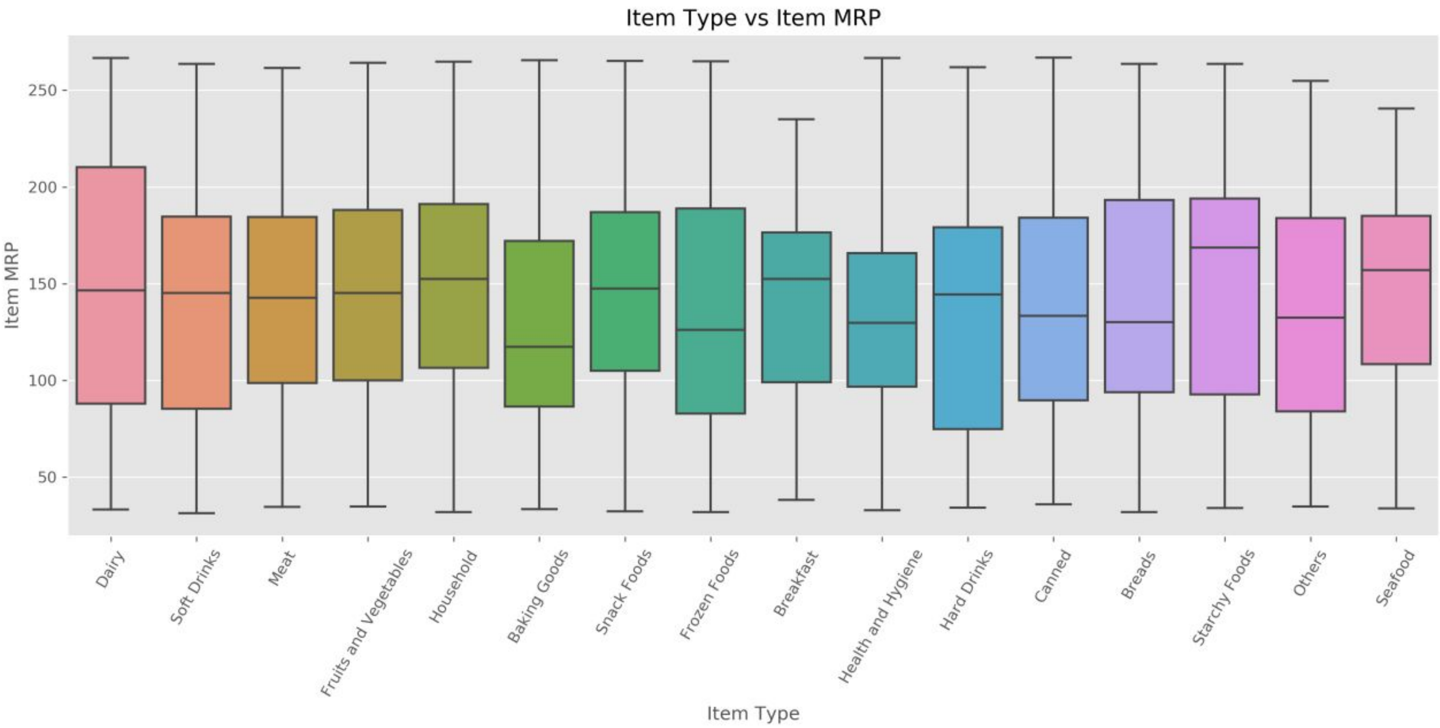
```
sns.violinplot(x = "Item_Type", y = "Item_Outlet_Sales", data = df)
plt.xlabel("Item Type")
plt.ylabel("Item Outlet Sales")
plt.title("Item Type VS Item Outlet Sales")
plt.xticks(rotation = 60)
plt.savefig(r"C:\Users\jaskeerat singh\Desktop\Data\figures\type vs outlet sales.png", dp
i=300, bbox_inches='tight')
plt.show()
```



Boxplot for Item Type VS Item MRP

In [361]:

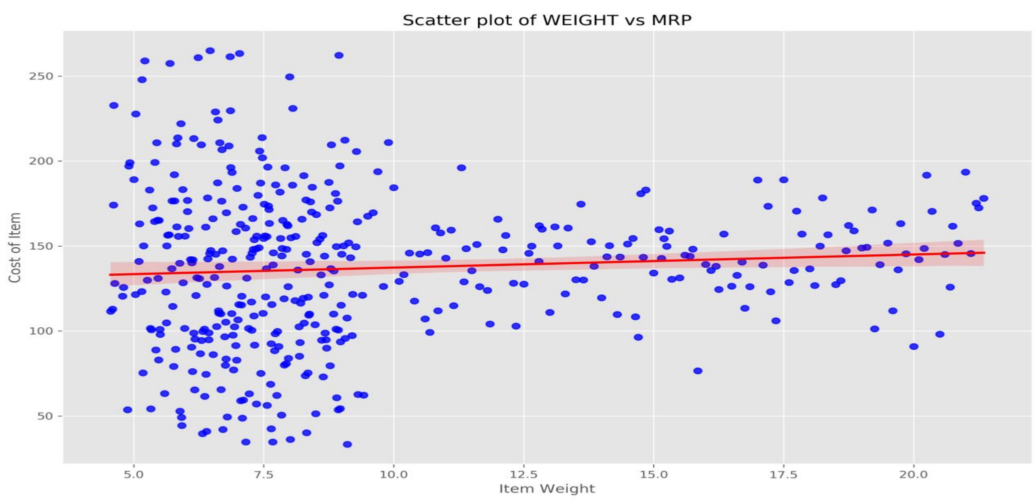
```
sns.boxplot(x = "Item_Type", y = "Item_MRP", data = df)
plt.xlabel("Item Type")
plt.ylabel("Item MRP")
plt.title("Item Type vs Item MRP")
plt.xticks(rotation = 60)
plt.savefig(r"C:\Users\jaskeerat singh\Desktop\Data\figures\type vs MRP.png", dpi=300, bb
ox_inches='tight')
plt.show()
```



Scatter Plot for Item Weight VS MRP

In [363]:

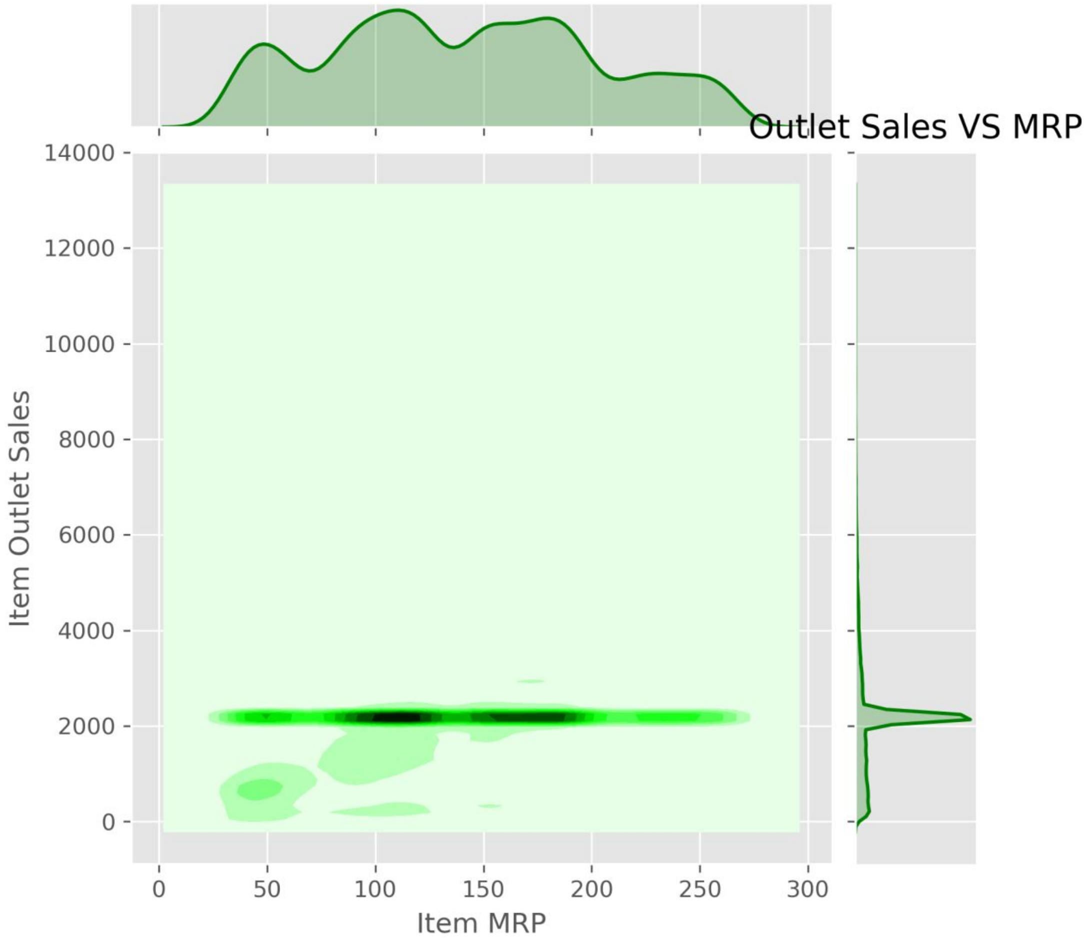
```
sns.lmplot("weight", "Item_MRP", data = df12, height=7, aspect=1.6,scatter_kws={"color":
"blue"}, line_kws={"color": "red"})
plt.xlabel("Item Weight")
plt.ylabel("Cost of Item")
plt.title("Scatter plot of WEIGHT vs MRP")
plt.savefig(r"C:\Users\jaskeerat singh\Desktop\Data\figures\Scatter plot of WEIGHT vs MRP
.png", dpi=300, bbox_inches='tight')
plt.show()
```



KDE Plot for Outlet Sales VS MRP

In [364]:

```
ax = sns.jointplot(x = "Item_MRP", y = "Item_Outlet_Sales", data = df, kind = "kde", col
or = "green")
ax.set_axis_labels(xlabel = "Item MRP", ylabel = "Item Outlet Sales")
plt.title("Outlet Sales VS MRP")
plt.savefig(r"C:\Users\jaskeerat singh\Desktop\Data\figures\outlet sales vs MRP.png", dpi
=300, bbox_inches='tight')
plt.show()
```



Label Encoding:

In [366]:

```
from sklearn.preprocessing import LabelEncoder

label = LabelEncoder()

df["outlet"] = label.fit_transform(df["Outlet_Identifier"])
df["outlet size"] = label.fit_transform(df["Outlet_Size"])
df["item identity"] = label.fit_transform(df["Item_Identifier"])
df["item type"] = label.fit_transform(df["Item_Type"])
```

One Hot Encoding:

In [367]:

```
from sklearn.preprocessing import OneHotEncoder

df = pd.get_dummies(df, columns = ["Item_Type","Item_Fat_Content","Outlet_Location_Type"
,"Outlet_Size","Outlet_Type"])
```

Data Modelling:

Now using the training data we will train our model using different machine learning algorithms to predict the sales price. First, we will split our data using the train_test_split method and will use Linear Regression, Regularized linear regression, Random Forest, and XGBoost algorithms, and let's see which model will give the lowest RMSE value which will become our preferable model to predict the sales price more accurately.

In [368]:

```
from sklearn.model_selection import KFold,train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn import metrics
```

In [369]:

```
training = df.loc[df["source"] == "training"]
testing = df.loc[df["source"] == "testing"]

training.to_csv(r"C:\Users\jaskeerat singh\Desktop\Data\training_mod.csv", index=False)
testing.to_csv(r"C:\Users\jaskeerat singh\Desktop\Data\testing_mod.csv", index=False)
```

In [370]:

```
training = pd.read_csv(r"C:\Users\jaskeerat singh\Desktop\Data\training_mod.csv")
training.drop("source", axis = 1, inplace = True)
testing.drop("source", axis = 1, inplace = True)
```

Linear Regression:

In [374]:

```
X = training.drop(["Item_Identifier","Outlet_Identifier","Item_Outlet_Sales"], axis=1)
y = training["Item_Outlet_Sales"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4, random_state
= 42)
```

In [382]:

```
linear = LinearRegression()
linear.fit(X_train,y_train)
```

```
y_pred = linear.predict(X_test)

np.sqrt(metrics.mean_squared_error(y_test,y_pred))
```

Out[382]:

```
1100.6333316017406
```

In [383]:

```
from sklearn.model_selection import cross_val_score

lr = LinearRegression()
c = 1111111111111111111
d = 0
for i in range(2,12):
    sc = cross_val_score(lr, X, y, cv=i, scoring='neg_root_mean_squared_error')
    sc = -sc
    print(i,") ",sc.mean())
    if sc.mean()<c:
        c=sc.mean()
        d=i
print('\nBest number of kfolds for cross validation is ',d,'\n')
```

```
2 )  1137.4154208323812
3 )  1133.7796242591005
4 )  1134.6969252714389
5 )  1132.7717901180756
6 )  1132.4110199818194
7 )  1131.554783209493
8 )  1131.72813395384
9 )  1131.7397025116445
10 )  1132.1532587972351
11 )  1131.5096373936842

Best number of kfolds for cross validation is  11
```

Regularized Linear regression:

In [384]:

```
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso

rlr = Ridge(alpha = 0.03)
rlr.fit(x_train, y_train)

pred_train_rlr = rlr.predict(x_train)
print(np.sqrt(metrics.mean_squared_error(y_train,pred_train_rlr)))

pred_test_rlr = rlr.predict(x_test)
print(np.sqrt(metrics.mean_squared_error(y_test,pred_test_rlr)))
```

```
1146.5585744641337
1100.6295269571458
```

Random Forest:

In [385]:

```
from sklearn.ensemble import RandomForestRegressor

forest = RandomForestRegressor()
forest.fit(x_train, y_train)

forest_train_predict = forest.predict(x_train)
print(np.sqrt(metrics.mean_squared_error(y_train, forest_train_predict)))
```

```
forest_test_predict = forest.predict(x_test)
print(np.sqrt(metrics.mean_squared_error(y_test, forest_test_predict)))
```

```
439.8789543689531
1111.23167581518
```

Predicting Values:

In [388]:

```
t = pd.read_csv(r"C:\Users\jaskeerat singh\Desktop\Data\testing_mod.csv")

use = t.drop(["Item_Identifier","Outlet_Identifier","Item_Outlet_Sales","source"], axis=1
)
value = linear.predict(use)
```

In [389]:

```
value
```

Out[389]:

```
array([1848.86375479, 1530.51482046, 1913.5217265 , ..., 1859.46895331,
       3643.78602979, 1290.89884878])
```

Task Summary:

All the using the algorithms required have been implemented and we are using a linear regression model which gives a root mean squared value of 1100. With this model, BigMart can understand that the properties of outlets Type, Tier, Size which play a major role in increasing the sales price and will help them in their company's growth.