PROJECT TITLE : **BIG MART SALES ANALYSIS**

by

N. Karthik

Objective :

The project aims to analyze sales data from Big Mart to uncover key insights, extract meaningful information, and interpret findings to guide stakeholders in making data-driven decisions.

Problem Statement :

Big Mart is a chain of retail outlets aiming to maximize its sales and profitability. The management has collected detailed data from 2013 on 1,559 products sold across 10 different stores in various cities. The dataset includes attributes related to the products, the outlets, and their sales performance.

The key objectives of this project are:

1. To **analyze sales trends and patterns** across different products and stores.
2. To **build a predictive model** that estimates the outlet sales based on its characteristics and store attributes.
3. To **Cluster the Outlet** based on 'Weight', 'ProductVisibility', 'MRP'.
4. To **identify key factors** influencing sales performance to assist in strategic decision-making.

Big Mart intends to leverage this analysis to:

- Better understand the properties of products and outlets that significantly drive sales.
- Optimize product placements, pricing strategies, and inventory management.
- Tailor marketing efforts to increase overall sales and enhance customer satisfaction.

Challenges:

The dataset contains missing values, possibly due to technical glitches in some stores. Effective handling of these missing data points is crucial to ensure the reliability of insights and predictions.

Dataset Overview:

The dataset is divided into two files: **Train** and **Test** datasets.

1. **Train Dataset:** Contains sales data for 8,523 records, including both input features and the target variable (*Item_Outlet_Sales*).
2. **Test Dataset:** Contains 5,681 records for which sales (*Item_Outlet_Sales*) need to be predicted.

Data Dictionary:

1. **Item_Identifier**: Unique product ID
2. **Item_Weight**: Weight of the product
3. **Item_Fat_Content**: Indicates whether the product is low fat or not
4. **Item_Visibility**: % of the total display area of all products in a store allocated to this product
5. **Item_Type**: Category of the product
6. **Item_MRP**: Maximum Retail Price (list price) of the product
7. **Outlet_Identifier**: Unique store ID
8. **Outlet_Establishment_Year**: The year in which the store was established
9. **Outlet_Size**: Size of the store (e.g., Small, Medium, Large)
10. **Outlet_Location_Type**: Type of city where the store is located (e.g., Tier 1, Tier 2, Tier 3)
11. **Outlet_Type**: Indicates the type of store (e.g., Grocery Store, Supermarket Type 1/2/3)
12. **Item_Outlet_Sales**: Total sales of the product in a particular store (Target variable in the train dataset)

Tools and Technologies:

- **Programming Language**: Python
- **Libraries Used**:
    - pandas for data manipulation and preprocessing

- matplotlib and seaborn for data visualization
- statsmodels for statistical analysis
- sklearn for predictive modeling

1. Libraries Used :

Libraries :

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
import statsmodels.api as sm
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
%matplotlib inline
```

2. Importing and Understanding the Data :

2.1: Import data

In [2]:
```
data = pd.read_csv("C://Users//KARTHIK//OneDrive//Desktop//INFOSYS//Final_Project//Train-Set.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	ProductID	Weight	FatContent	ProductVisibility	ProductType	MRP	OutletID	EstablishmentYear	OutletSize	LocationType	OutletType	OutletSales
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

2.2: Data shape

In [5]:
```
data.shape
```

Out[5]: (8523, 12)

2.3 Data Columns

```
In [6]: for columns in data.columns:
            print(columns)
```

```
ProductID
Weight
FatContent
ProductVisibility
ProductType
MRP
OutletID
EstablishmentYear
OutletSize
LocationType
OutletType
OutletSales
```

2.4 Data Information

```
In [7]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   ProductID          8523 non-null   object
 1   Weight             7060 non-null   float64
 2   FatContent         8523 non-null   object
 3   ProductVisibility  8523 non-null   float64
 4   ProductType        8523 non-null   object
 5   MRP                8523 non-null   float64
 6   OutletID           8523 non-null   object
 7   EstablishmentYear  8523 non-null   int64
 8   OutletSize         6113 non-null   object
 9   LocationType       8523 non-null   object
 10  OutletType         8523 non-null   object
 11  OutletSales        8523 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 799.2+ KB
```

2.5 Data Describe

```
In [8]: data.describe()
```

Out[8]:

	Weight	ProductVisibility	MRP	EstablishmentYear	OutletSales
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

3. Data Cleaning

3.1 Check for null values

```
In [9]: data.isnull().sum()
```

```
Out[9]: ProductID              0
        Weight              1463
        FatContent             0
        ProductVisibility      0
        ProductType            0
        MRP                    0
        OutletID               0
        EstablishmentYear      0
        OutletSize          2410
        LocationType           0
        OutletType             0
        OutletSales            0
        dtype: int64
```

3.2 Replacing the mean value for the column weight based on product type

```
In [10]: data['Weight'] = data.groupby('ProductType')['Weight'].transform(lambda x: x.fillna(x.mean()))
```

```
In [11]: data.isnull().sum()
```

```
Out[11]: ProductID              0
         Weight                 0
         FatContent             0
         ProductVisibility      0
         ProductType            0
         MRP                    0
         OutletID               0
         EstablishmentYear      0
         OutletSize          2410
         LocationType           0
         OutletType             0
         OutletSales            0
         dtype: int64
```

3.3 Replacing the mode values for the column OutletSize with missing values, based on the OutletType and LocationType

```
In [12]: # Mode Imputation or Group-Based Imputation for OutletSize
         # Fill OutletSize with the most common size within each OutletType and LocationType combination
         data['OutletSize'] = data.groupby(['OutletType', 'LocationType'])['OutletSize'].transform(lambda x: x.fillna(x.mode().iloc[0] if

         # If any missing values remain in OutletSize, fill them with the most frequent value
         imputer = SimpleImputer(strategy='most_frequent')
         data['OutletSize'] = imputer.fit_transform(data[['OutletSize']])
```

3.4 Check for null values

```
In [13]: data.isnull().sum()

Out[13]: ProductID              0
         Weight                 0
         FatContent             0
         ProductVisibility      0
         ProductType            0
         MRP                    0
         OutletID               0
         EstablishmentYear      0
         OutletSize             0
         LocationType           0
         OutletType             0
         OutletSales            0
         dtype: int64
```

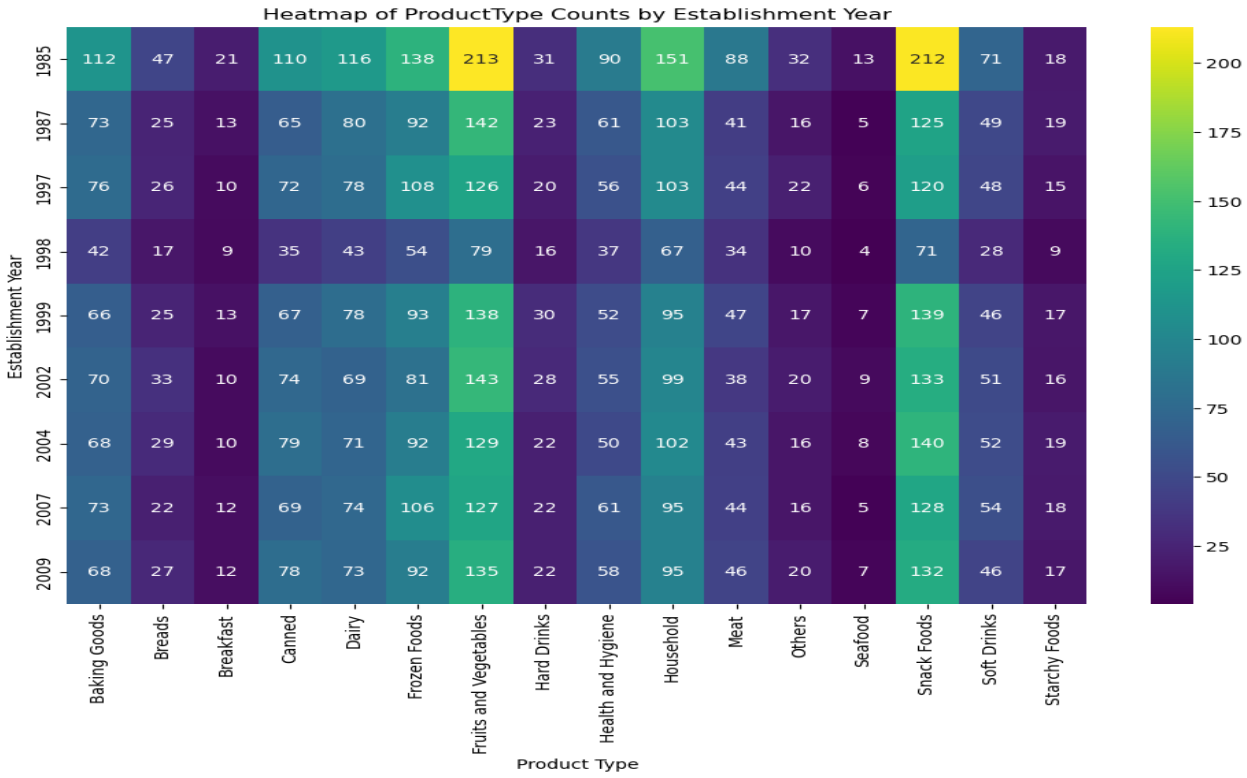
Missing values are handled using the appropriate methods.

Now the data doesn't contain any missing values. Everything is handled using appropriate methods.

4. Exploratory Data Analysis (EDA)

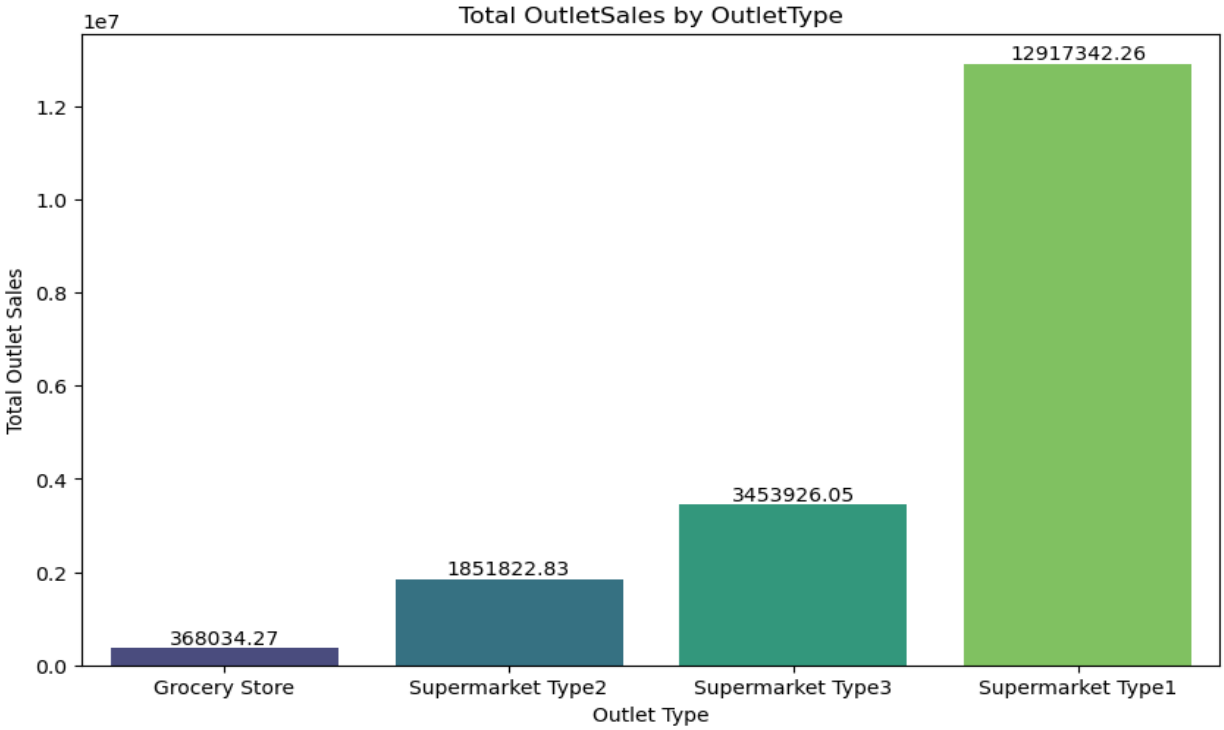
4.1 HeatMap

This heatmap shows the count of each product type according to the establishment year. ***Where Fruits and Vegetables have highest count***.



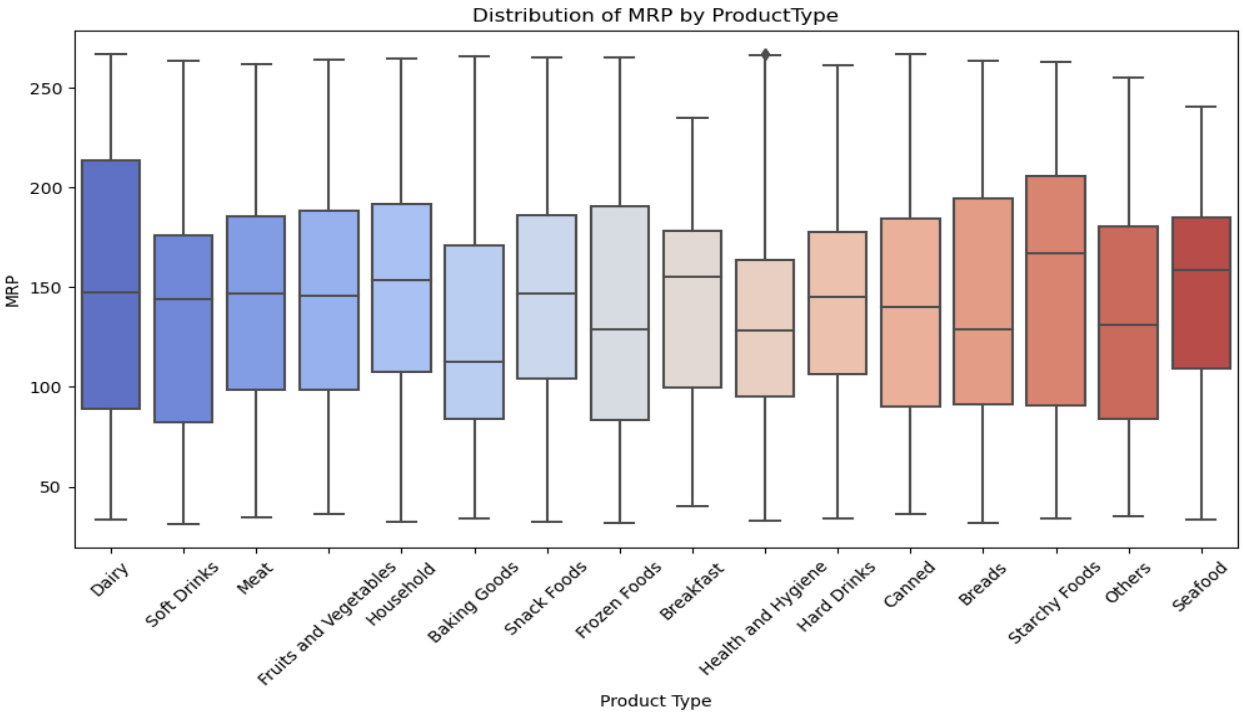
4.2 BarChart

This figure shows the Total OutletSales By each of the outletType, Where ***Supermarket Type1 has the Highest outlet sales***.



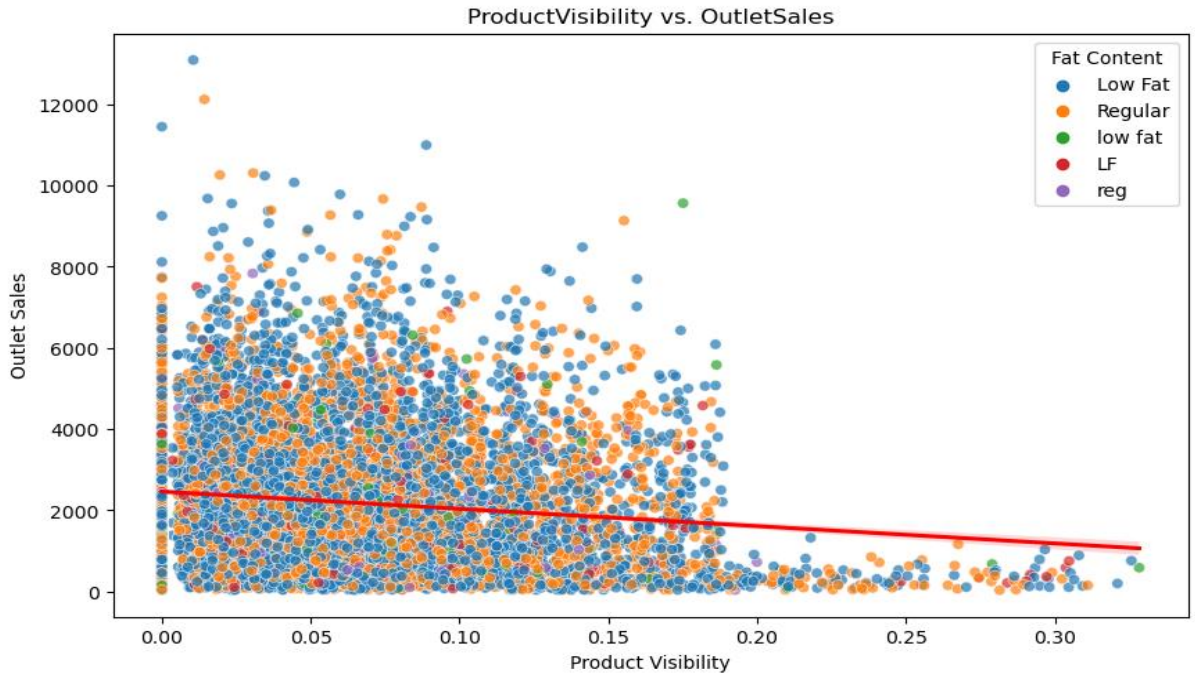
4.3 BoxPlot

This Box plot shows the Distribution of MRP by each productType.



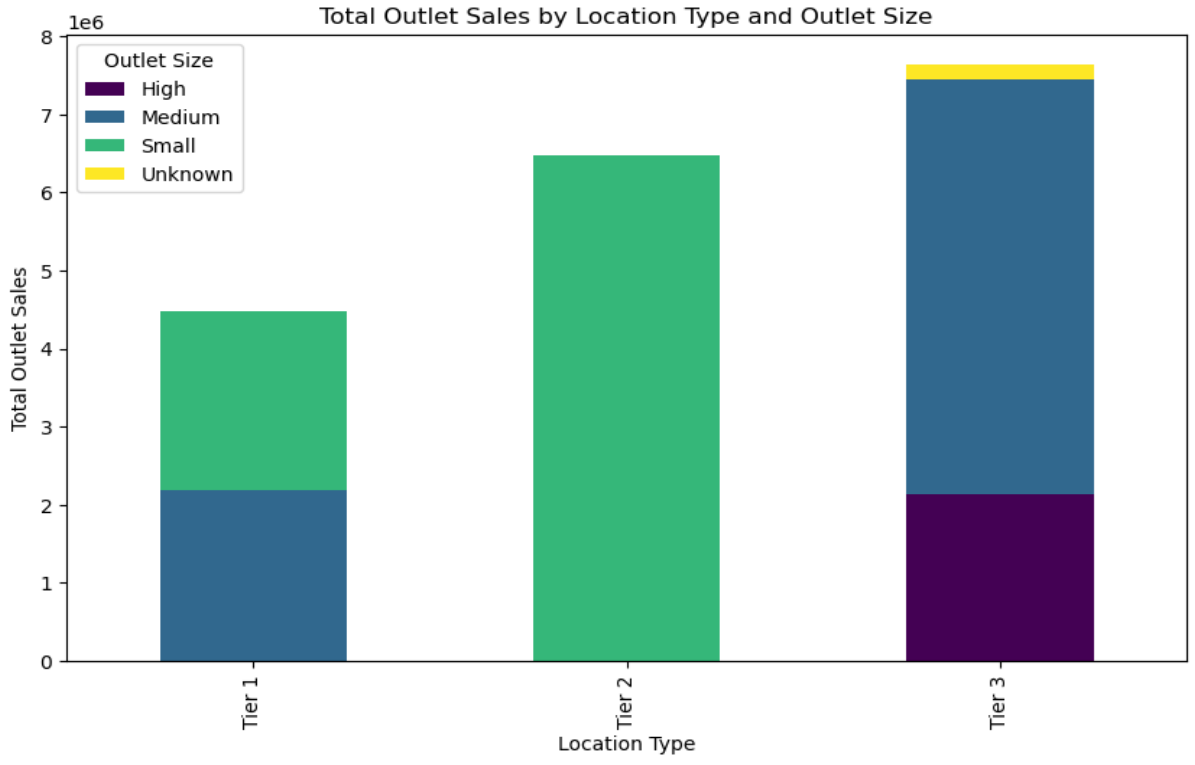
4.4 ScatterPlot

This ScatterPlot shows the Relationship between productVisibility and OutletSales. According to the dataset the lower product visibility has highest Sales.



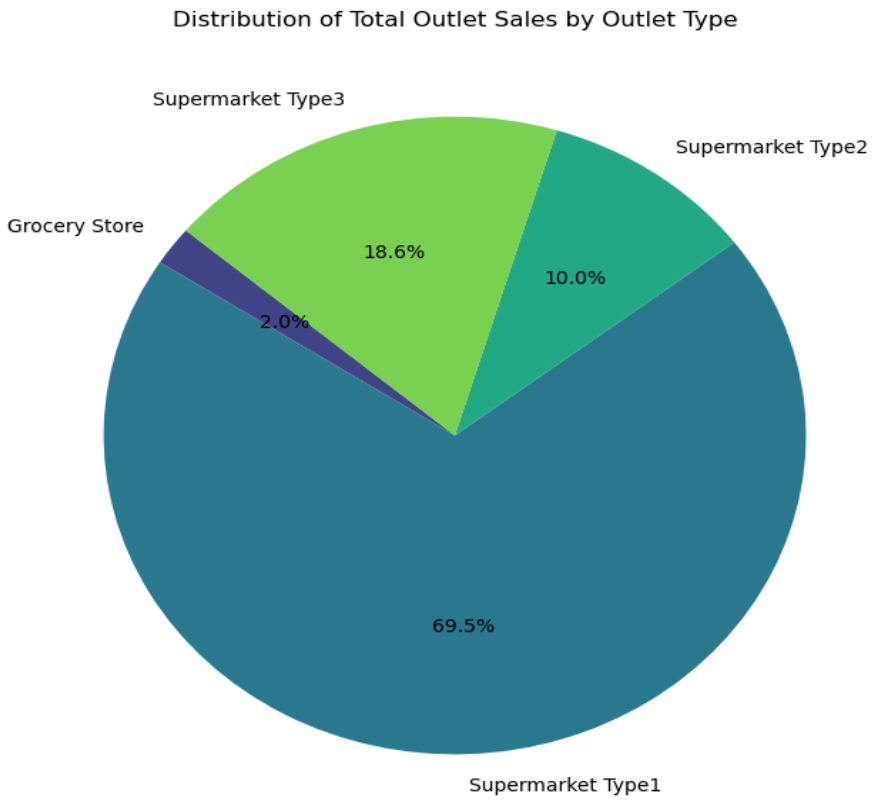
4.5 Stacked BarChart

This visual shows the Total OutletSales by each location type.



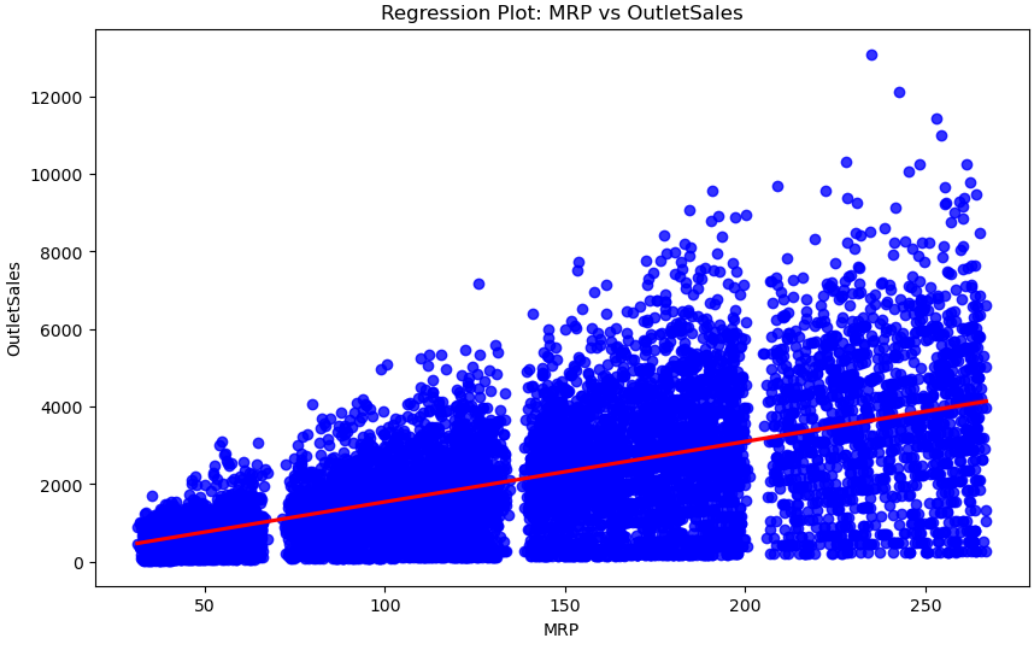
4.6 PieChart

This piechart shows the Distribution of Total OutletSales by OutletType.



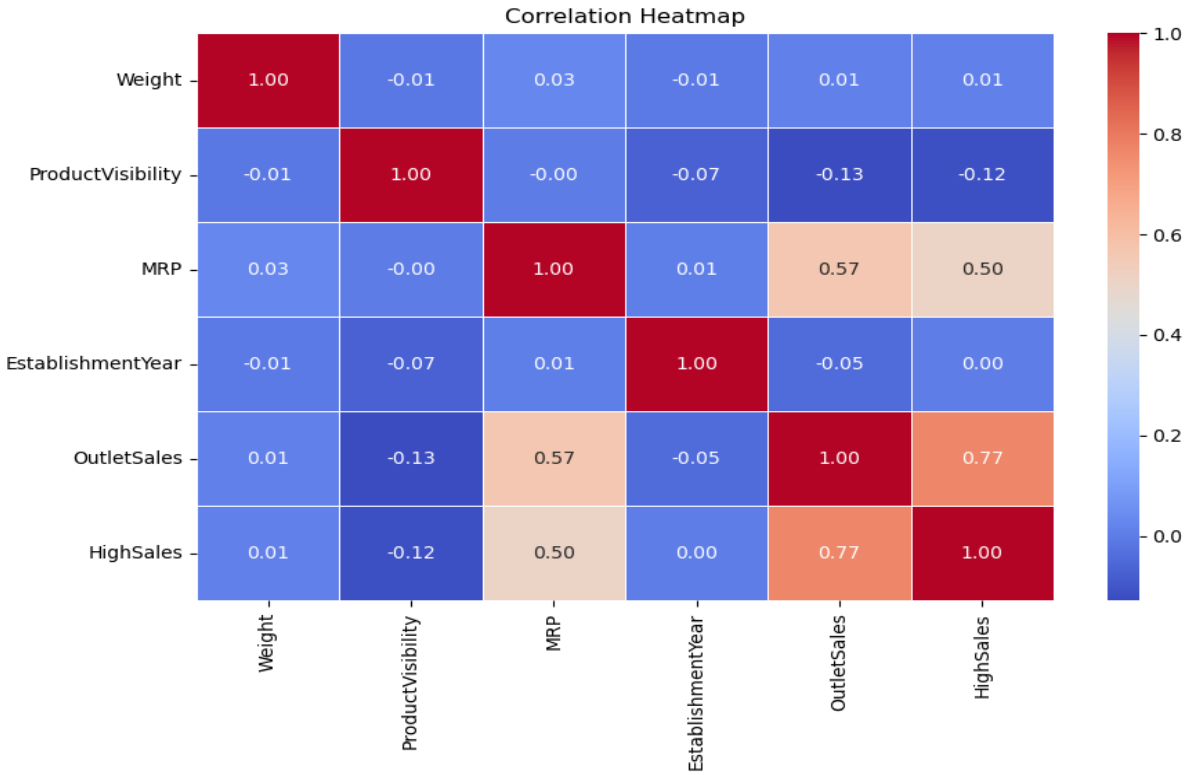
4.7 Regression Plot

This Regression plot shows that if MRP increases then the OutletSales also increasing.



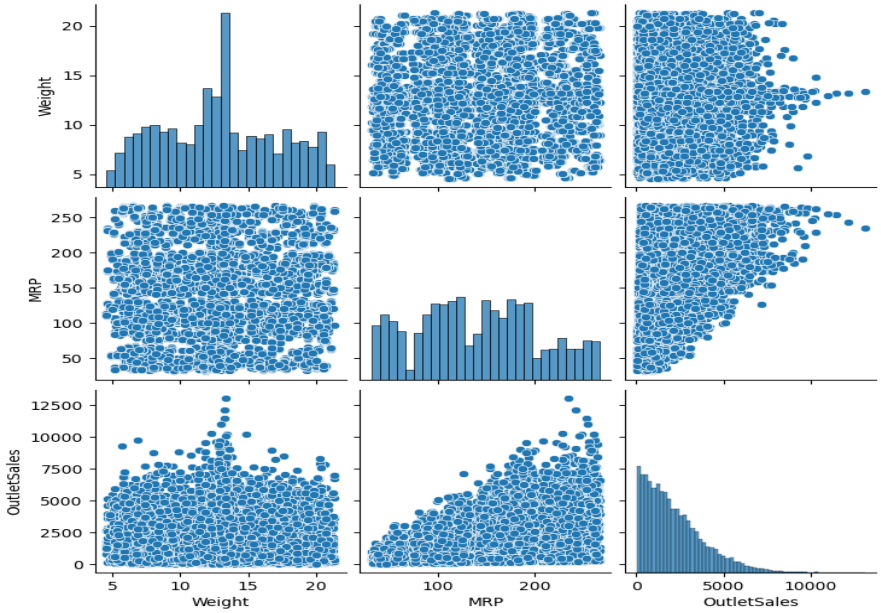
4.8 HeatMap

This Correlation Heatmap shows the correlation between the each variables. Where the MRP & OutletSales have (0.57), Which is the positive Correlation and MRP & HighSales have (0.50).



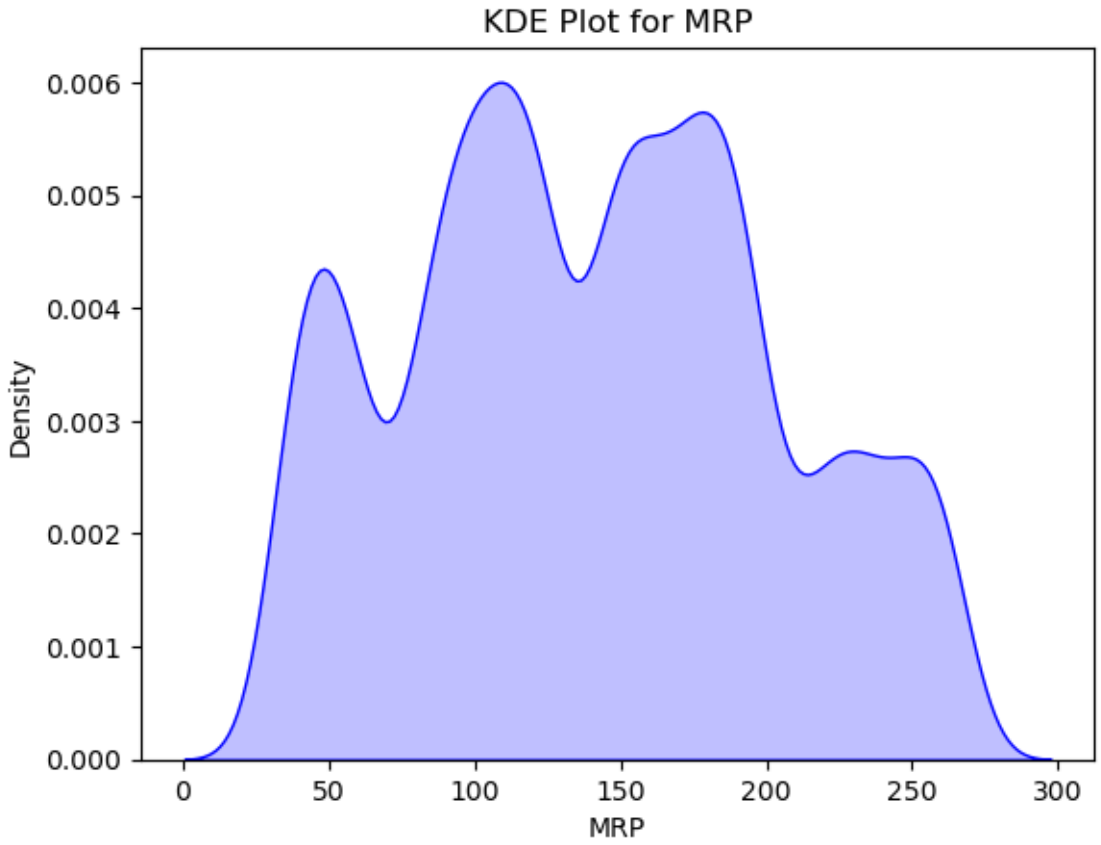
4.9 PairPlot

This PairPlot displays the scatter plots between pairs of numerical columns and histograms on the diagonal for each individual feature.



4.10 Kde Plot

This Visuals shows that Most of the MRP in the BIG MART DATA is range from 90-110 which is highest density and the second highest MRP ranges from 150 – 200.



5. Ordinary Least Square

5.1 Libraries

```
In [35]: import statsmodels.api as sm
         import pandas as pd
         import matplotlib.pyplot as plt
         from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

5.2 Features and Target

```
X = data[['Weight', 'ProductVisibility', 'MRP', 'EstablishmentYear', 'OutletSize', 'LocationType', 'OutletType']]
y = data['OutletSales']
```

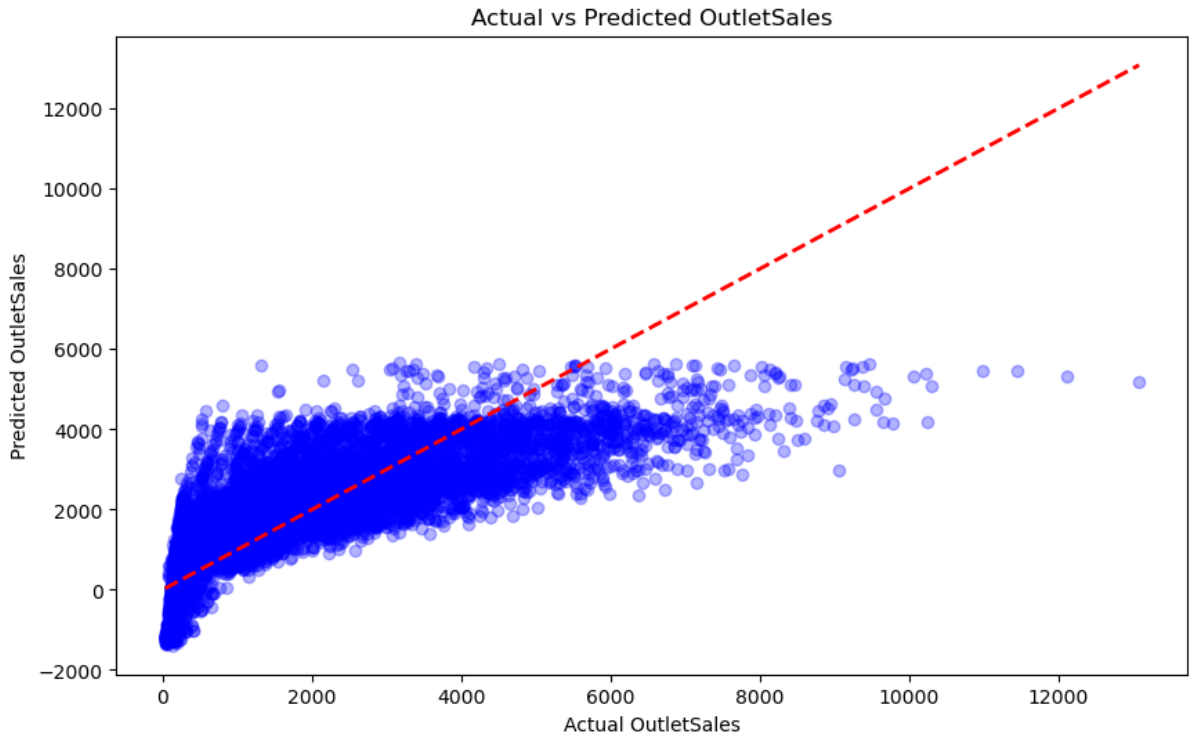
5.3 Fitting

```
In [37]: X = pd.get_dummies(X, drop_first=True)

         X = sm.add_constant(X)

         model = sm.OLS(y, X).fit()
```

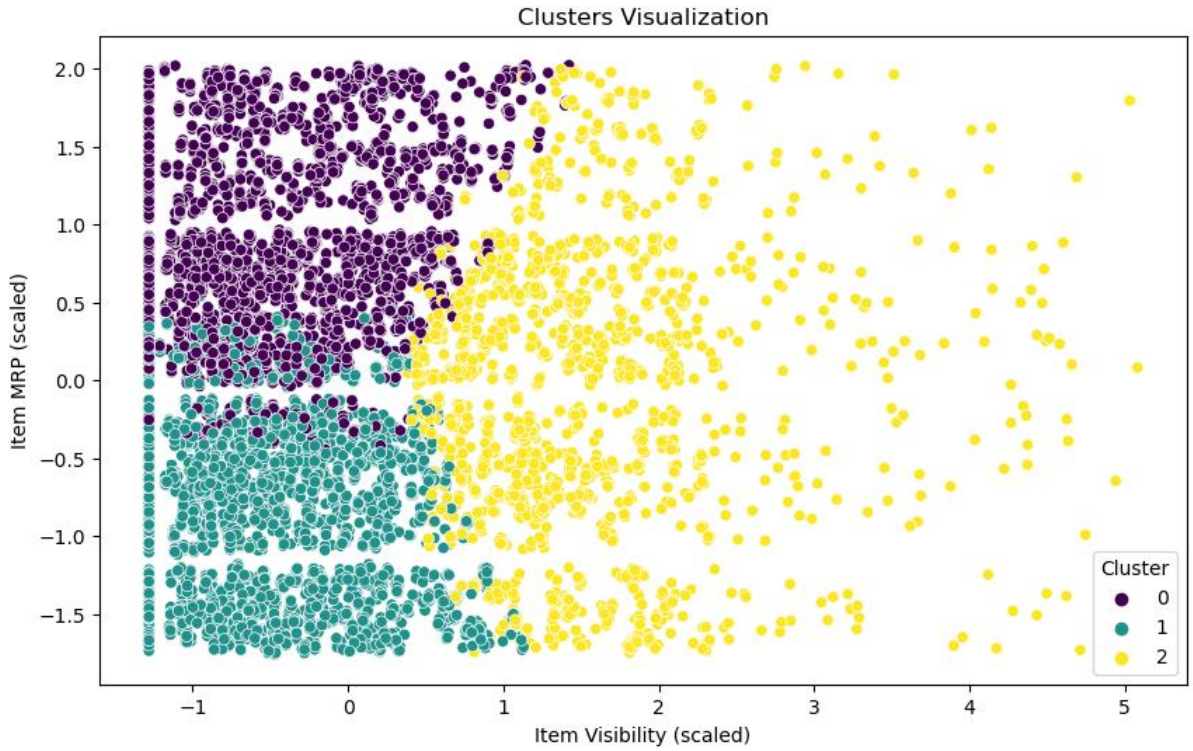
5.4 Fitted Line Equation

```
Fitted Line Equation:
y = -52833.03 + -0.53*Weight + -265.23*ProductVisibility + 15.57*MRP + 30.72*EstablishmentYear + -9911.63*OutletSize_Mediu
m + -9948.22*OutletSize_Small + -755.73*OutletSize_Unknown + -165.09*LocationType_Tier 2 + -9610.09*LocationType_Tier 3 +
1520.79*OutletType_Supermarket Type1 + 10449.34*OutletType_Supermarket Type2 + 12913.75*OutletType_Supermarket Type3
```

5.5 Actual vs Predicted Plot



6. K means Clustering



Cluster Centers:

	Weight	ProductVisibility	MRP
0	13.880908	0.045785	195.263207
1	11.823541	0.043246	89.819905
2	12.888935	0.142790	136.120049