Data Analysis Project on - Blinkit Sales Data

- Blinkit is an online grocery delivery platform that connects customers with local stores, providing groceries and essentials quickly and efficiently,this dataset contains information that helps us to make quick decisions make insights.
- This data can help analyze patterns such as which product types sell best, how fat content affects sales, outlet performance across regions,and more.
- Blinkit was founded to provide on-demand grocery delivery services.
- It connects customers to nearby stores for instant grocery needs.

Import Libraries

In [6]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Data Exploration and Understanding

Import Raw Data

In [7]:

```
df=pd.read_csv("blinkit_data.csv")
df
```

Out[7]:

	Item Fat Content	Item Identifier	Item Type	Outlet Establishment Year	Outlet Identifier	Outlet Location Type	Outlet Size	Outlet Type	Item Visibility
0	Regular	FDX32	Fruits and Vegetables	2012	OUT049	Tier 1	Medium	Supermarket Type1	0.100014
1	Low Fat	NCB42	Health and Hygiene	2022	OUT018	Tier 3	Medium	Supermarket Type2	0.008596
2	Regular	FDR28	Frozen Foods	2010	OUT046	Tier 1	Small	Supermarket Type1	0.025896
3	Regular	FDL50	Canned	2000	OUT013	Tier 3	High	Supermarket Type1	0.042278
4	Low Fat	DRI25	Soft Drinks	2015	OUT045	Tier 2	Small	Supermarket Type1	0.033970
...	...	...	...	...	...	...	...	...	...
8518	low fat	NCT53	Health and Hygiene	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.000000

	Item Fat Content	Item Identifier	Item Type	Outlet Establishment Year	Outlet Identifier	Outlet Location Type	Outlet Size	Outlet Type	Item Visibility
8519	low fat	FDN09	Snack Foods	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.034706
8520	low fat	DRE13	Soft Drinks	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.027571
8521	reg	FDT50	Dairy	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.107715
8522	reg	FDM58	Snack Foods	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.000000

8523 rows × 12 columns

Observation

- it will show the first 5 rows from the dataset

Sample from the Blinkit dataset

In [8]:
```
df.sample(5)
```

Out[8]:

	Item Fat Content	Item Identifier	Item Type	Outlet Establishment Year	Outlet Identifier	Outlet Location Type	Outlet Size	Outlet Type	Item Visibility	W
8254	LF	FDS51	Meat	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.032025	
7372	Low Fat	FDC39	Dairy	2017	OUT035	Tier 2	Small	Supermarket Type1	0.159165	
387	Regular	FDH04	Frozen Foods	2022	OUT018	Tier 3	Medium	Supermarket Type2	0.011419	
3602	Low Fat	FDT36	Baking Goods	2011	OUT010	Tier 3	Medium	Grocery Store	0.186251	1
8107	Low Fat	FDX15	Meat	2022	OUT018	Tier 3	Medium	Supermarket Type2	0.156936	1

Observation

- The sample shows the few dataset rows,if the dataset is loaded successfully or not.

In [9]:
```
df.head(5)
```

Out[9]:

	Item Fat Content	Item Identifier	Item Type	Outlet Establishment Year	Outlet Identifier	Outlet Location Type	Outlet Size	Outlet Type	Item Visibility	W
0	Regular	FDX32	Fruits and Vegetables	2012	OUT049	Tier 1	Medium	Supermarket Type1	0.100014	
1	Low Fat	NCB42	Health and Hygiene	2022	OUT018	Tier 3	Medium	Supermarket Type2	0.008596	
2	Regular	FDR28	Frozen Foods	2010	OUT046	Tier 1	Small	Supermarket Type1	0.025896	
3	Regular	FDL50	Canned	2000	OUT013	Tier 3	High	Supermarket Type1	0.042278	
4	Low Fat	DRI25	Soft Drinks	2015	OUT045	Tier 2	Small	Supermarket Type1	0.033970	

Observation

- it will show the first 5 rows from the dataset

Shape of the dataset

In [10]:
```
df.shape
```

Out[10]:
```
(8523, 12)
```

Observation

- this shows the shape of the dataset contains 8523 rows and 12 columns with blinkit and there item type and outlet, sales rating.

Columns in the Dataset

In [11]:
```
df.columns.tolist()
```

Out[11]:

```
['Item Fat Content',
 'Item Identifier',
 'Item Type',
 'Outlet Establishment Year',
 'Outlet Identifier',
 'Outlet Location Type',
 'Outlet Size',
 'Outlet Type',
 'Item Visibility',
 'Item Weight',
 'Sales',
 'Rating']
```

Summary of the Dataset and Data types

In [12]:

```
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 Fat Content           8523 non-null   object 
 1   Item Identifier            8523 non-null   object 
 2   Item Type                  8523 non-null   object 
 3   Outlet Establishment Year  8523 non-null   int64  
 4   Outlet Identifier          8523 non-null   object 
 5   Outlet Location Type       8523 non-null   object 
 6   Outlet Size                8523 non-null   object 
 7   Outlet Type                8523 non-null   object 
 8   Item Visibility            8523 non-null   float64
 9   Item Weight                7060 non-null   float64
 10  Sales                      8523 non-null   float64
 11  Rating                     8523 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 799.2+ KB
```

Observation

- Rows:8523
- columns:12
- Non-Null count:9 columns have the no non null values
- numerical columns: 5 columns are numerical(outlet establishment year,item visibility,item weight,sales,rating)
- categorical columns: 7 columns are categorical that includes item fat content,item type...etc
- memory usage: 799.2 KB

Summary of the Dataset and Data types

In [13]:

```
df.describe()
```

Out[13]:

	Outlet Establishment Year	Item Visibility	Item Weight	Sales	Rating
count	8523.000000	8523.000000	7060.000000	8523.000000	8523.000000
mean	2010.831867	0.066132	12.857645	140.992782	3.965857
std	8.371760	0.051598	4.643456	62.275067	0.605651
min	1998.000000	0.000000	4.555000	31.290000	1.000000
25%	2000.000000	0.026989	8.773750	93.826500	4.000000
50%	2012.000000	0.053931	12.600000	143.012800	4.000000
75%	2017.000000	0.094585	16.850000	185.643700	4.200000
max	2022.000000	0.328391	21.350000	266.888400	5.000000

Cleaning the Dataset(checking Null values, duplicate values)

Checking Null Values

In [14]:

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

Out[14]:

```
Item Fat Content               0
Item Identifier                0
Item Type                      0
Outlet Establishment Year      0
Outlet Identifier              0
Outlet Location Type           0
Outlet Size                    0
Outlet Type                    0
Item Visibility                0
Item Weight                 1463
Sales                          0
Rating                         0
dtype: int64
```

Observation

- columns(Item visibility)containts the null values

In [15]:

```
df['Item Weight'] = df['Item Weight'].fillna(df['Item Weight'].mean())
```

In [16]:

```
df['Item Weight'].isnull().sum()
```

Out[16]:

```
0
```

After cleaning null values

In [17]:
```
df.isnull().sum()
```

Out[17]:
```
Item Fat Content             0
Item Identifier              0
Item Type                    0
Outlet Establishment Year    0
Outlet Identifier            0
Outlet Location Type         0
Outlet Size                  0
Outlet Type                  0
Item Visibility              0
Item Weight                  0
Sales                        0
Rating                       0
dtype: int64
```

Checking for duplicate values

In [18]:
```
df.duplicated().sum()
```

Out[18]:
```
0
```

Observation

- we don't have any duplicate values in the dataset(if there is duplicate values present in the dataset we have to remove them)

Checking inconsistent columns

In [19]:
```
df['Item Fat Content'].unique()
```

Out[19]:
```
array(['Regular', 'Low Fat', 'low fat', 'LF', 'reg'], dtype=object)
```

In [20]:
```
df['Item Fat Content'] = df['Item Fat Content'].replace({
    'LF': 'Low Fat',
    'low fat': 'Low Fat',
    'reg': 'Regular'
})
```

In [21]:
```
df['Item Fat Content'].unique()
```

Out[21]:
```
array(['Regular', 'Low Fat'], dtype=object)
```

Observation

- we found inconsistent columns in the dataset first column(Item Fat content) and converted into regular column

In [22]:
```
df.columns
```

Out[22]:
```
Index(['Item Fat Content', 'Item Identifier', 'Item Type',
       'Outlet Establishment Year', 'Outlet Identifier',
       'Outlet Location Type', 'Outlet Size', 'Outlet Type', 'Item Visibility',
       'Item Weight', 'Sales', 'Rating'],
      dtype='object')
```

Understanding the Columns

In [23]:
```
df['Item Fat Content'].value_counts()
```

Out[23]:
```
Item Fat Content
Low Fat    5517
Regular    3006
Name: count, dtype: int64
```

Observation

- it shows us to how many types Fat present in Item fat content and how many times they appear.

In [24]:
```
df['Item Identifier'].value_counts()
```

Out[24]:
```
Item Identifier
FDW13    10
FDG33    10
FDF56     9
FDF52     9
FDV38     9
         ..
FDK57     1
FDO33     1
DRF48     1
FDC23     1
FDE52     1
Name: count, Length: 1559, dtype: int64
```

Observation

- it shows how many types of identifiers in them

In [25]:
```
df['Item Type'].value_counts()
```

Out[25]:
```
Item Type
Fruits and Vegetables    1232
Snack Foods              1200
Household                 910
Frozen Foods              856
Dairy                     682
Canned                    649
Baking Goods              648
Health and Hygiene        520
Soft Drinks               445
Meat                      425
Breads                    251
Hard Drinks               214
Others                    169
Starchy Foods             148
Breakfast                 110
Seafood                    64
Name: count, dtype: int64
```

Observation

- it shows how many types of food items present in them and their counts.

In [26]:
```
df['Outlet Establishment Year'].value_counts()
```

Out[26]:
```
Outlet Establishment Year
1998    1463
2000     932
2012     930
2010     930
2017     930
2015     929
2022     928
2020     926
2011     555
Name: count, dtype: int64
```

Observation

- this shows that outlet establismet year and their number of counts to understand clearly.

In [27]:
```
df['Outlet Identifier'].value_counts()
```

Out[27]:

```
Outlet Identifier
OUT027    935
OUT013    932
OUT049    930
OUT046    930
OUT035    930
OUT045    929
OUT018    928
OUT017    926
OUT010    555
OUT019    528
Name: count, dtype: int64
```

Observation

- it shows that number of outlets identified according to the region and its counts.

In [28]:

```
df['Outlet Location Type'].value_counts()
```

Out[28]:

```
Outlet Location Type
Tier 3    3350
Tier 2    2785
Tier 1    2388
Name: count, dtype: int64
```

Observation

- it shows how many outlet located in different types of tiers and their count in tiers.

In [29]:

```
df['Outlet Size'].value_counts()
```

Out[29]:

```
Outlet Size
Medium    3631
Small     3139
High      1753
Name: count, dtype: int64
```

Observation

- it shows that the size of the outlet and their count.

In [25]:

```
df['Outlet Type'].value_counts()
```

Out[25]:

```
Outlet Type
Supermarket Type1    5577
Grocery Store        1083
Supermarket Type3     935
```

```
Supermarket Type2     928
Name: count, dtype: int64
```

Observation

- it shows the type of outlet and their differnet types of outlet and their counts.

In [26]:

```
df['Item Visibility'].value_counts()
```

Out[26]:

```
Item Visibility
0.000000    526
0.076975      3
0.037794      2
0.127930      2
0.121521      2
           ...
0.027344      1
0.022059      1
0.150853      1
0.018930      1
0.107715      1
Name: count, Length: 7880, dtype: int64
```

Observation

- it shows which of the items are visibling and their size.

In [27]:

```
df['Item Weight'].value_counts()
```

Out[27]:

```
Item Weight
12.857645    1463
12.150000      86
17.600000      82
13.650000      77
11.800000      76
             ...
6.895000        2
6.520000        1
9.420000        1
7.685000        1
5.400000        1
Name: count, Length: 416, dtype: int64
```

Observation

- it specifies the item weight and their number of counts.

In [28]:

```
df['Sales'].value_counts()
```

Out[28]:

```
Sales
172.0422    7
196.5768    6
109.5228    6
196.5084    6
142.0154    6
           ..
50.2666     1
178.3686    1
165.4868    1
36.2848     1
112.2544    1
Name: count, Length: 5938, dtype: int64
```

Observation

- it shows that how many number of sales happened in the stores in online and their counts.

In [29]:

```
df['Rating'].value_counts()
```

Out[29]:

```
Rating
4.0    3339
4.3     694
4.2     673
4.1     524
5.0     427
4.4     340
3.9     302
4.5     256
3.8     253
3.7     211
3.0     193
3.5     171
3.6     150
3.3     121
1.0     114
4.6     102
3.4      83
4.7      79
2.0      74
4.8      62
3.2      55
2.8      49
3.1      49
2.5      48
2.3      26
2.9      24
2.7      21
4.9      15
2.6      14
2.4      11
1.5      11
2.2       9
1.7       8
```

```
1.8       5
2.1       4
1.9       2
1.3       2
1.4       1
1.2       1
Name: count, dtype: int64
```

observation

- it shows that how many persons are given ratings and how they satisfied with our service and their number of counts.

Decting How many No of outliers

- IQR (interquartile range) is used to detect the outlier in all numerical columns.

Steps:

- Calculate the Q1(25th percentile) and Q3(75th percentile)
- Find IQR=Q3-Q1
- define Lower and Upper Bound
- Lower Bound: Q1-1.5 * IQR
- Upper Bound: Q3+1.5 * IQR
- Any value outside this range is considered as outlier

In [30]:
```
# Method 1
```

In [31]:
```
# Define numerical columns
Numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns

# Detect outliers
outliers = {}
for col in Numerical_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    outlier_values = df[(df[col] < lower) | (df[col] > upper)]
    if not outlier_values.empty:
        outliers[col] = outlier_values[col].values.tolist()
print("Outliers found in each column:")
for col, values in outliers.items():
    print(f"{col}: {len(values)} outliers")
```

```
Outliers found in each column:
Item Visibility: 144 outliers
Rating: 1931 outliers
```

In [32]:

```
# Method 2
```

In [33]:

```
Numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns
for col in Numerical_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    # Count outliers
    outliers_count = df[(df[col] < lower) | (df[col] > upper)][col].count()

    print(f"{col}: {outliers_count} outliers")
```

```
Outlet Establishment Year: 0 outliers
Item Visibility: 144 outliers
Item Weight: 0 outliers
Sales: 0 outliers
Rating: 1931 outliers
```

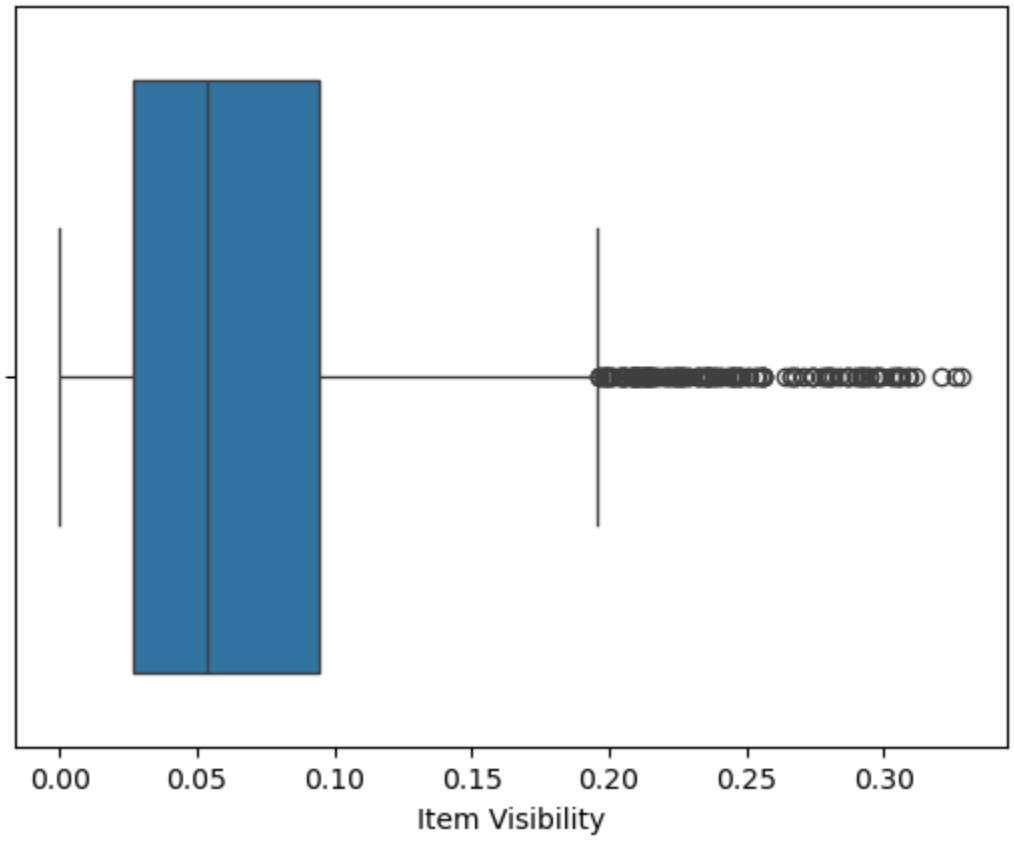
Outliers in Item Visibility & Rating

In [34]:

```
sns.boxplot(data=df, x='Item Visibility')
```

Out[34]:

```
<Axes: xlabel='Item Visibility'>
```

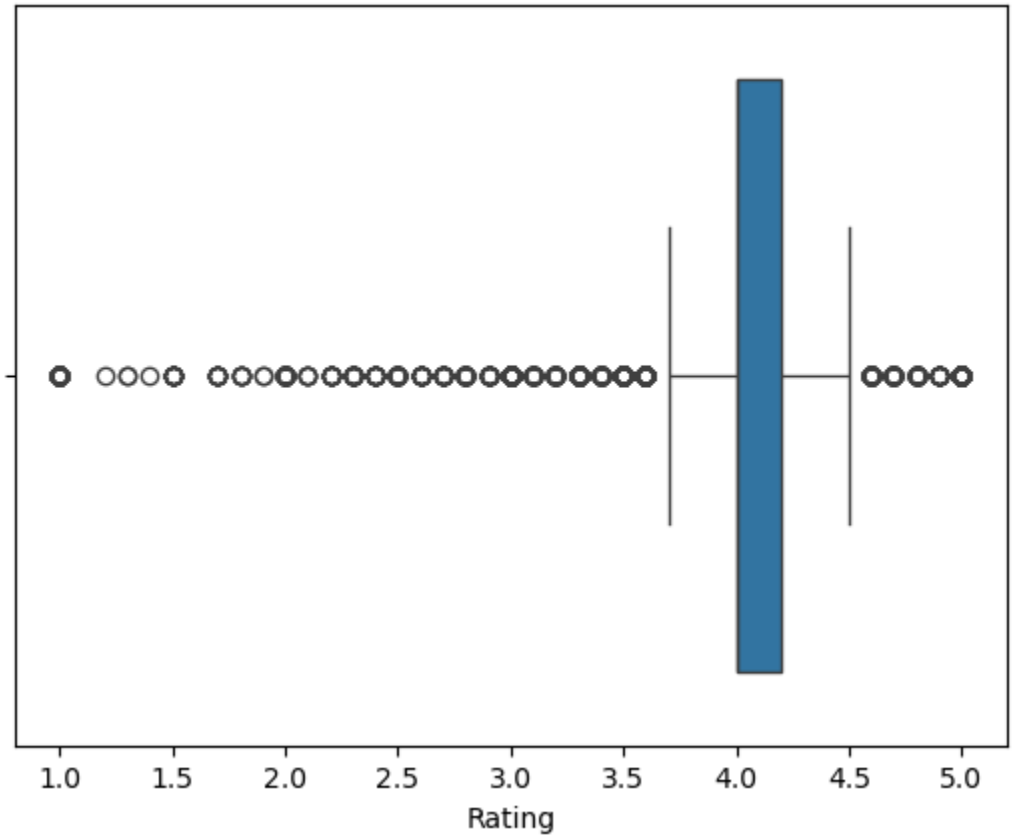


In [35]:

```
sns.boxplot(data=df, x='Rating')
```

Out[35]:

```
<Axes: xlabel='Rating'>
```



Detecting outliers

In [36]:

```
# Detecting outliers at the same time by usung for loop
```

In [37]:

```
for col in ['Item Visibility', 'Rating']:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    df = df[(df[col] >= lower) & (df[col] <= upper)]

print(" Outliers removed successfully!")
print("Shape after removing outliers:", df.shape)
```

```
 Outliers removed successfully!
Shape after removing outliers: (6480, 12)
```

In [38]:

```
# Detecting them Seperately
```

Outliers of Item visibility

In [39]:

```
Q1 = df['Item Visibility'].quantile(0.25)
Q3 = df['Item Visibility'].quantile(0.75)
IQR = Q3 - Q1
lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR

df = df[(df['Item Visibility'] >= lower) & (df['Item Visibility'] <= upper)]

print("Outliers removed successfully for Item Visibility")
print("Shape after removing Item Visibility outliers:", df.shape)
```

```
Outliers removed successfully for Item Visibility
Shape after removing Item Visibility outliers: (6472, 12)
```

Outliers of Rating

In [40]:

```
# Remove outliers for Rating
Q1 = df['Rating'].quantile(0.25)
Q3 = df['Rating'].quantile(0.75)
IQR = Q3 - Q1
lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR

df = df[(df['Rating'] >= lower) & (df['Rating'] <= upper)]

print("Outliers removed successfully for Rating")
print("Shape after removing Rating outliers:", df.shape)
```

```
Outliers removed successfully for Rating
Shape after removing Rating outliers: (6472, 12)
```

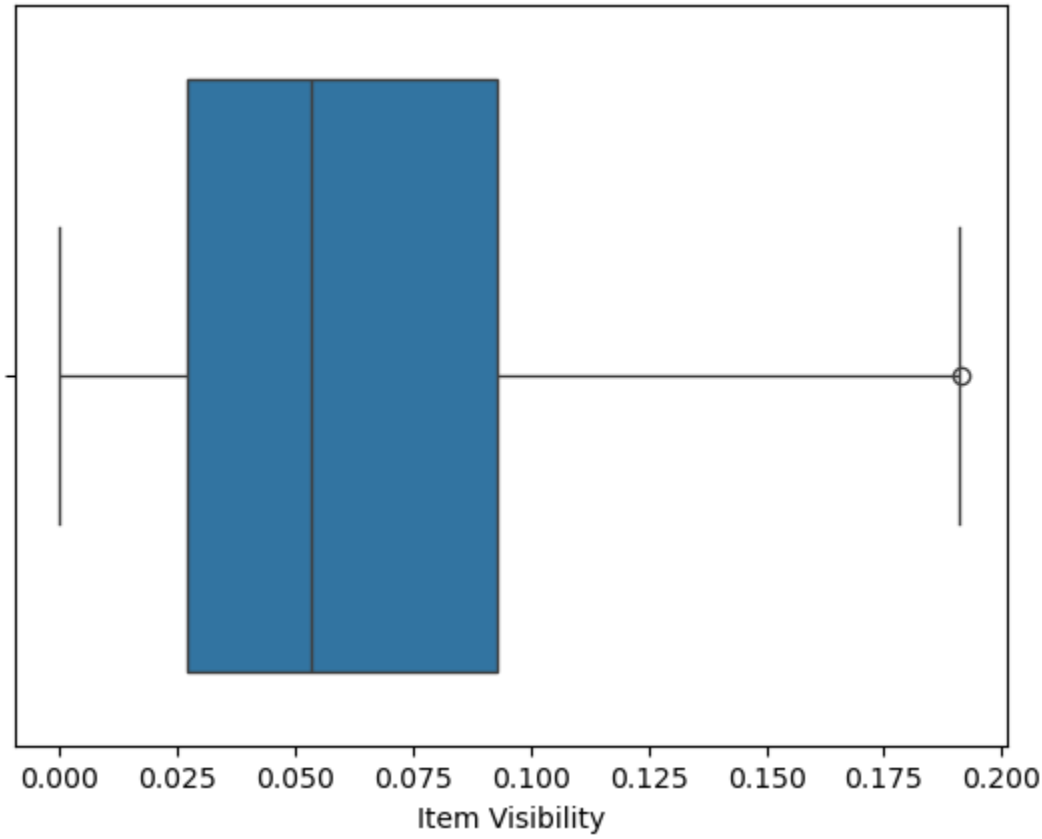
After Removing Outliers from both Columns

In [41]:

```
sns.boxplot(data=df, x='Item Visibility')
```

Out[41]:

```
<Axes: xlabel='Item Visibility'>
```

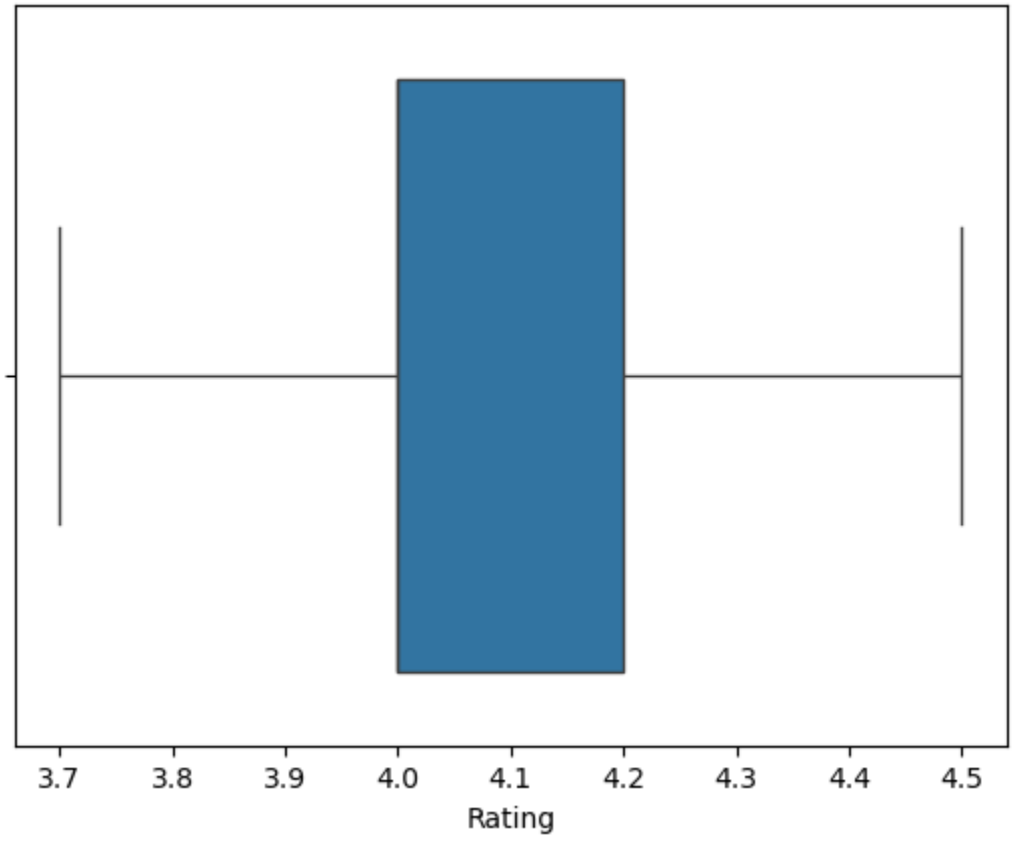


In [42]:

```
sns.boxplot(data=df, x='Rating')
```

Out[42]:

```
<Axes: xlabel='Rating'>
```



- Filterd the outliers using the IQR method in both item visibility and rating columns.
- By Caluculating upper and lower and filtered using & operator to keep the values b/w the range.

Final Summary after Data understanding & Data Exploration

- Dataset Structure: 6472rows x 12 columns
- Column Types: 5 Nuerical columns + 7 Categorical
- Missing values: There are some missing values in item visibility,rating columns
- Outliers Found in item visibility & rating,not much extreme outliers but closer to upper bound in both columns,removed them.
- Duplicated values: There is no duplicated values in them.

Ensure the dataset is consistent and ready for analysis

Explore the dataset

Go through each column carefully to understand its data type and meaning.

Use Pandas functions(basic checks) (.info(), .describe(), .value_counts()) to summarize the dataset

In [43]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6472 entries, 685 to 8522
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Item Fat Content           6472 non-null   object 
 1   Item Identifier            6472 non-null   object 
 2   Item Type                  6472 non-null   object 
 3   Outlet Establishment Year  6472 non-null   int64  
 4   Outlet Identifier          6472 non-null   object 
 5   Outlet Location Type       6472 non-null   object 
 6   Outlet Size                6472 non-null   object 
 7   Outlet Type                6472 non-null   object 
 8   Item Visibility            6472 non-null   float64
 9   Item Weight                6472 non-null   float64
 10  Sales                      6472 non-null   float64
 11  Rating                     6472 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 657.3+ KB
```

In [44]:

```
df.describe()
```

Out[44]:

	Outlet Establishment Year	Item Visibility	Item Weight	Sales	Rating
count	6472.000000	6472.000000	6472.000000	6472.000000	6472.000000
mean	2010.893387	0.063177	12.861526	141.298838	4.077905
std	8.374026	0.045736	4.226266	62.388188	0.176462
min	1998.000000	0.000000	4.555000	31.290000	3.700000
25%	2000.000000	0.027031	9.373750	94.045550	4.000000
50%	2012.000000	0.053376	12.857645	142.915400	4.000000
75%	2017.000000	0.092707	16.000000	186.755600	4.200000
max	2022.000000	0.191501	21.350000	266.888400	4.500000

In [45]:

```
df.value_counts()
```

Out[45]:

```
Item Fat Content  Item Identifier  Item Type     Outlet Establishment Year  Outlet Ident
ifier  Outlet Location Type  Outlet Size  Outlet Type        Item Visibility  Item Weigh
t  Sales     Rating
Low Fat           DRA12            Soft Drinks   2000                       OUT013
Tier 3                High         Supermarket Type1  0.040912         11.600000    142.
3154  4.0       1
Regular           FDA51            Dairy         2000                       OUT013
Tier 3                High         Supermarket Type1  0.164543         8.050000     112.
2518  4.0       1
                  FDB02            Canned        2015                       OUT045
Tier 2                High         Supermarket Type1  0.029223         9.695000     175.
4370  4.0       1
                                                 2000                       OUT013
Tier 3                High         Supermarket Type1  0.029140         9.695000     176.
3370  4.0       1
                                                 1998                       OUT027
Tier 3                Medium       Supermarket Type3  0.029023         12.857645    177.
8370  4.4       1

..
Low Fat           FDS10            Snack Foods   2020                       OUT017
Tier 2                High         Supermarket Type1  0.035385         19.200000    180.
3318  4.2       1
                                                 2011                       OUT010
Tier 3                High         Grocery Store      0.058893         19.200000    178.
9318  4.0       1
                                                 2010                       OUT046
Tier 1                Small        Supermarket Type1  0.035186         19.200000    181.
3318  4.0       1
                                                 1998                       OUT027
Tier 3                Medium       Supermarket Type3  0.035015         12.857645    182.
0318  4.4       1
Regular           FDZ59            Baking Goods  2012                       OUT049
Tier 1                Medium       Supermarket Type1  0.104183         6.630000     166.
```

```
1500  4.0       1
Name: count, Length: 6472, dtype: int64
```

In [46]:

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

Out[46]:

```
Item Fat Content             0
Item Identifier              0
Item Type                    0
Outlet Establishment Year    0
Outlet Identifier            0
Outlet Location Type         0
Outlet Size                  0
Outlet Type                  0
Item Visibility              0
Item Weight                  0
Sales                        0
Rating                       0
dtype: int64
```

In [47]:

```
df.head()
```

Out[47]:

	Item Fat Content	Item Identifier	Item Type	Outlet Establishment Year	Outlet Identifier	Outlet Location Type	Outlet Size	Outlet Type	Item Visibility
685	Low Fat	NCP18	Household	2020	OUT017	Tier 2	Small	Supermarket Type1	0.028760
686	Low Fat	FDY58	Snack Foods	2022	OUT018	Tier 3	Medium	Supermarket Type2	0.040081
687	Low Fat	NCB06	Health and Hygiene	2017	OUT035	Tier 2	Small	Supermarket Type1	0.082317
688	Regular	FDB36	Baking Goods	2022	OUT018	Tier 3	Medium	Supermarket Type2	0.000000
689	Low Fat	FDR47	Breads	2011	OUT010	Tier 3	Small	Grocery Store	0.000000

observation

- there is no need of fixing the inconsistecies in the categorical columns like spellings formatting and cases(lower/upper)
- Removed unnceesary white spacesor special charscters.
- Standardize categorical to avoid fragmentation of groups.

Non-visual bivariate analysis

- Categorical vs Categorical: Check which of items are sold in which outlet most frequently.(e.g., Item type vsOutlet Type).
- Categorical vs numerical: Compare average salesacross outlet types.(e.g., Outlet Type vs Sales)
- Numerical vs Numerical: Explore whether heavier items sell more or less.

Categorical-Categorical

1. Which item types are sold the most in each outlet type?

In [48]:
```
pd.crosstab(df['Item Type'], df['Outlet Type'])
```

Out[48]:

Outlet Type Item Type	Grocery Store	Supermarket Type1	Supermarket Type2	Supermarket Type3
Baking Goods	59	326	57	56
Breads	25	126	15	23
Breakfast	10	51	10	9
Canned	49	333	59	58
Dairy	57	354	58	52
Frozen Foods	72	437	69	73
Fruits and Vegetables	108	608	112	109
Hard Drinks	17	106	17	14
Health and Hygiene	47	259	42	47
Household	89	464	72	78
Meat	46	200	36	41
Others	20	79	18	13
Seafood	6	33	7	6
Snack Foods	99	596	99	105
Soft Drinks	35	224	34	38
Starchy Foods	4	82	12	12

Observation

- Supermarkets sell a wider range of items like snacks, dairy, and frozen foods.
- Grocery stores handle fewer item types.
- Larger outlets have more diverse product categories.

2. Which outlet type has the highest average sales?

In [49]:
```
df.groupby('Outlet Type')['Sales'].mean()
```

Out[49]:
```
Outlet Type
Grocery Store        140.440469
Supermarket Type1    141.420106
Supermarket Type2    141.166148
Supermarket Type3    141.590557
Name: Sales, dtype: float64
```

Observation

- Supermarket Type1 shows the highest average sales.
- Grocery stores have relatively lower sales.
- Bigger outlet types attract more customers and revenue.

3. Which item type has higher visibility in stores?

In [50]:
```
df.groupby('Item Type')['Item Visibility'].median()
```

Out[50]:
```
Item Type
Baking Goods             0.056293
Breads                   0.054221
Breakfast                0.068893
Canned                   0.047783
Dairy                    0.062039
Frozen Foods             0.056281
Fruits and Vegetables    0.054080
Hard Drinks              0.064218
Health and Hygiene       0.046521
Household                0.042297
Meat                     0.046995
Others                   0.048005
Seafood                  0.054068
Snack Foods              0.054277
Soft Drinks              0.045239
Starchy Foods            0.060078
Name: Item Visibility, dtype: float64
```

Observation

- Snacks and soft drinks have higher median visibility.
- Health and household items have lower visibility.
- Frequently purchased items are displayed more prominently.

4. Does item weight affect sales?

In [51]:
```
df[['Item Weight', 'Sales']].corr()
```

Out[51]:

	Item Weight	Sales
Item Weight	1.000000	0.024499
Sales	0.024499	1.000000

Observation

- Correlation between item weight and sales is weak or close to zero.
- Product weight has little to no impact on total sales.

5. Does more visibility lead to better ratings?

In [52]:
```
df[['Item Visibility', 'Rating']].corr()
```

Out[52]:

	Item Visibility	Rating
Item Visibility	1.000000	-0.001671
Rating	-0.001671	1.000000

Observation

- Very low or no correlation between visibility and ratings.
- Product visibility does not strongly affect customer satisfaction.
- Ratings may depend more on product quality than display.

6. Which outlet type has better customer ratings?

In [53]:
```
df.groupby('Outlet Type')['Rating'].mean()
```

Out[53]:
```
Outlet Type
Grocery Store        4.080754
Supermarket Type1    4.079243
Supermarket Type2    4.073222
Supermarket Type3    4.071798
Name: Rating, dtype: float64
```

Observation

- Supermarkets generally have higher average ratings compared to grocery stores.
- This shows that customers are more satisfied with the shopping experience in larger outlets.

7. Which item type has the highest average sales?

In [54]:
```
df.groupby('Item Type')['Sales'].mean()
```

Out[54]:
```
Item Type
Baking Goods             127.425053
Breads                   142.555526
Breakfast                143.462925
Canned                   139.886154
Dairy                    148.819719
Frozen Foods             140.112833
Fruits and Vegetables    146.224147
Hard Drinks              135.982308
Health and Hygiene       128.271234
Household                150.272069
Meat                     140.534233
Others                   135.769048
Seafood                  141.480427
Snack Foods              145.946710
Soft Drinks              128.825825
Starchy Foods            141.345416
Name: Sales, dtype: float64
```

Observation

- Items like snacks, dairy, and soft drinks show higher average sales.
- Health and household products have relatively lower sales.
- This suggests that fast-moving consumer goods contribute more to total revenue.

8. Does outlet size affect total sales?

In [55]:
```
df.groupby('Outlet Size')['Sales'].mean()
```

Out[55]:
```
Outlet Size
High      142.962855
Medium    140.114251
Small     141.794086
Name: Sales, dtype: float64
```

Observation

- Larger outlets have higher average sales than small outlets.
- This indicates that bigger outlets can handle more customers and product variety
- Store size positively influences revenue generation.

9. Do sales differ by outlet location type (Urban, Tier 1, Tier 2, etc.)?

In [56]:
```
df.groupby('Outlet Location Type')['Sales'].mean()
```

Out[56]:
```
Outlet Location Type
Tier 1    141.024616
Tier 2    141.440201
Tier 3    141.375494
Name: Sales, dtype: float64
```

Observation

- Urban and Tier 1 outlets show higher average sales.
- Rural or Tier 3 outlets record comparatively lower sales.
- Sales performance depends partly on location and customer base.

10. Do low-fat or regular-fat items sell more?

In [57]:
```
df.groupby('Item Fat Content')['Sales'].mean()
```

Out[57]:
```
Item Fat Content
Low Fat    140.106943
Regular    143.491737
Name: Sales, dtype: float64
```

Observation

- Regular-fat items generally have higher average sales than low-fat ones.
- Customers seem to prefer regular products despite health-focused options.
- Fat content slightly influences customer buying behavior.

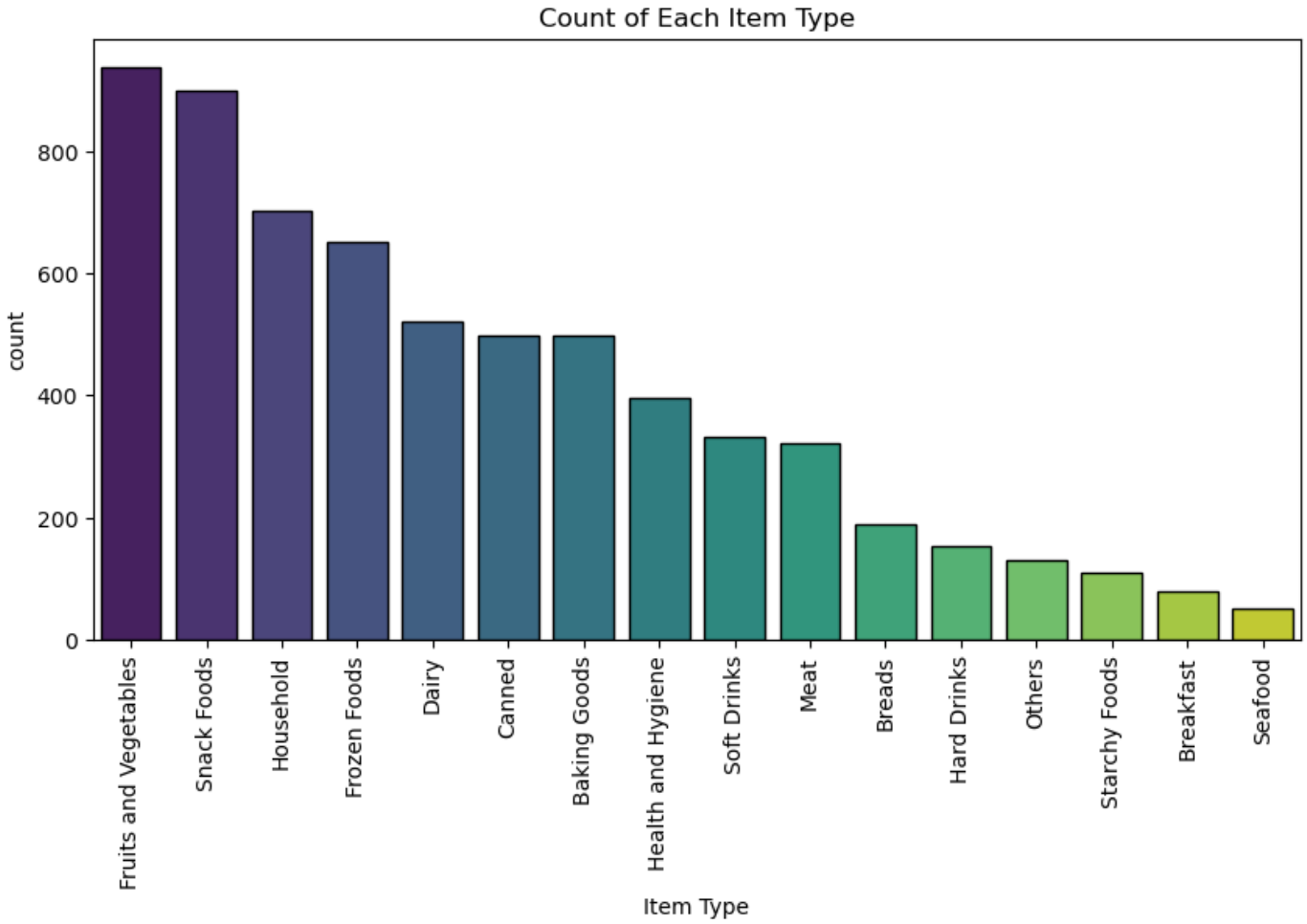
Univariante Analysis

- Univariate analysis focuses on studying a single variable to understand its distribution, pattern, and summary statistics. It helps identify trends like which values are most frequent or how data is spread.

1. Which item type is most sold in the Blinkit dataset?

In [74]:

```
plt.figure(figsize=(10,5))
sns.countplot(x='Item Type', data=df, order=df['Item Type'].value_counts().index, edgeco
plt.title('Count of Each Item Type')
plt.xticks(rotation=90)
plt.show()
```



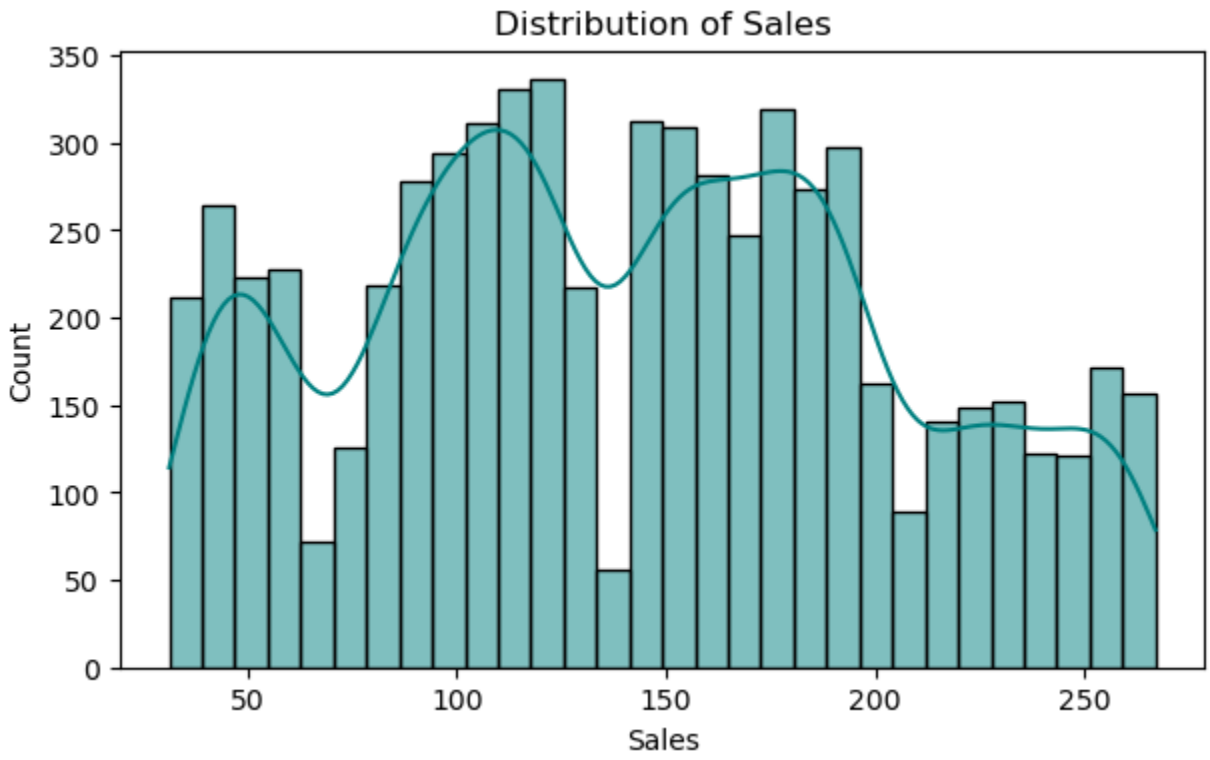
Observation

- Fruits and vegetables and snack foods are the most common item types in the dataset
- Fruits and vegetables and snack foods contribute to most of the sales.
- Household and frozen foods also show good sales performance.
- Seafood, breakfast, and starchy foods have very low sales.
- Overall, sales are mainly driven by everyday essential and frequently purchased items.

2. What is the distribution of total sales values?

In [59]:

```
plt.figure(figsize=(7,4))
sns.histplot(df['Sales'], kde=True, bins=30, color='teal')
plt.title('Distribution of Sales')
plt.show()
```



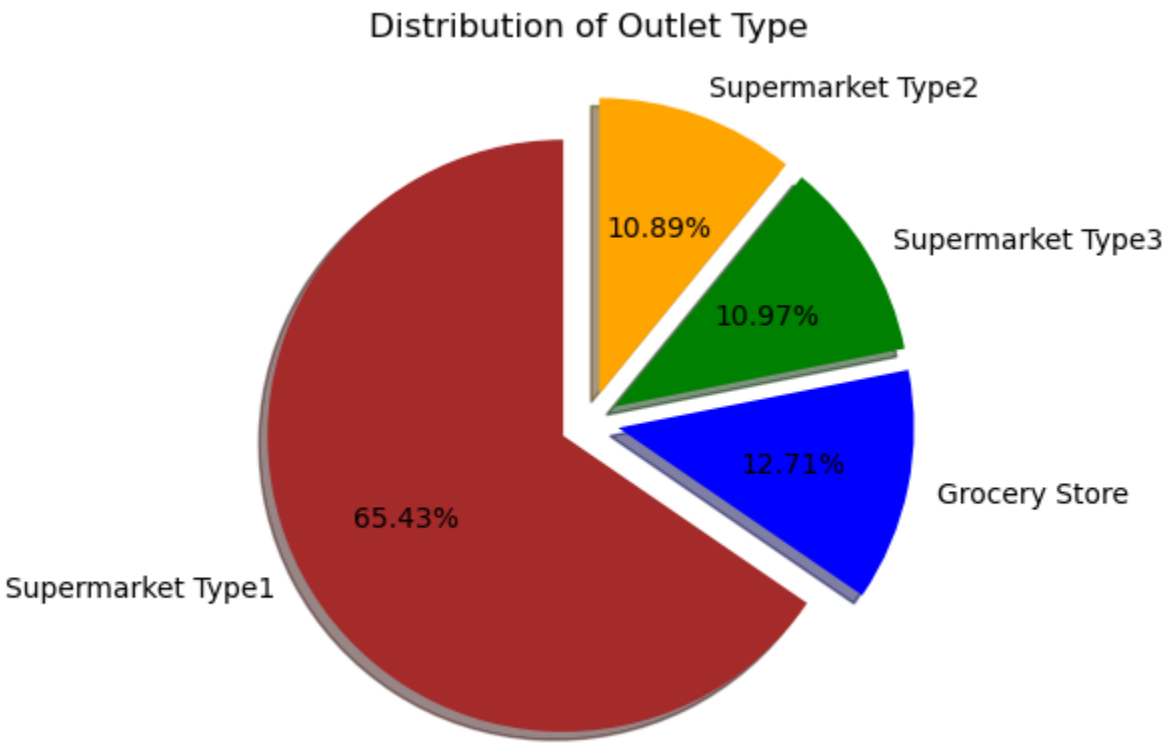
Observation

- Sales are spread across a wide range, showing variation in how different products perform.
- Most products achieve moderate sales between 100 and 200, indicating steady performance.
- Only a few products record very high or very low sales, showing limited extremes in the dataset.

3. What is the distribution of different outlet types in the Blinkit dataset?

In [3]:

```
plt.pie(
    df['Outlet Type'].value_counts(),
    labels=df['Outlet Type'].value_counts().index,
    colors=['brown', 'blue', 'green', 'orange'],
    startangle=90,
    autopct='%0.2f%%',
    shadow=True,
    explode=(0.1, 0.1, 0.1, 0.1)
)
plt.title("Distribution of Outlet Type")
plt.show()
```



Observation

- Supermarket Type1 outlets make up the largest portion of all outlets in the dataset.
- Grocery Store, Supermarket Type2, and Supermarket Type3 have smaller shares.
- Most sales and data entries come from Supermarket Type1, showing its major presence in the Blinkit dataset.

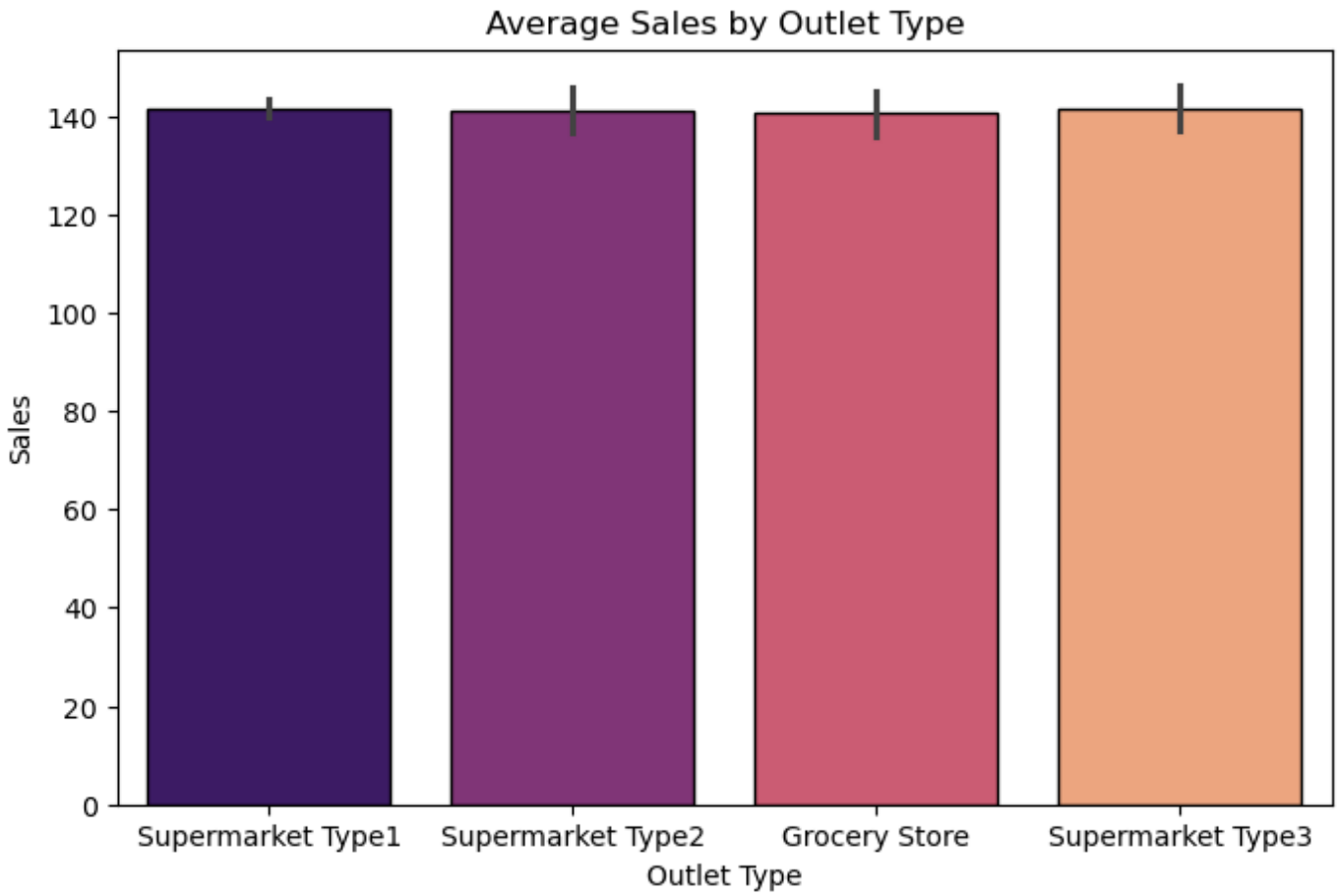
Bivariate Analysis

- Bivariate analysis is used to explore the relationship or association between two different variables. It helps identify how one variable influences or correlates with another through comparison or correlation techniques.

1. Which outlet type has the highest total sales?

In [62]:

```
plt.figure(figsize=(8,5))
sns.barplot(x='Outlet Type', y='Sales', data=df, edgecolor='Black',estimator='mean', pal
plt.title('Average Sales by Outlet Type')
plt.show()
```



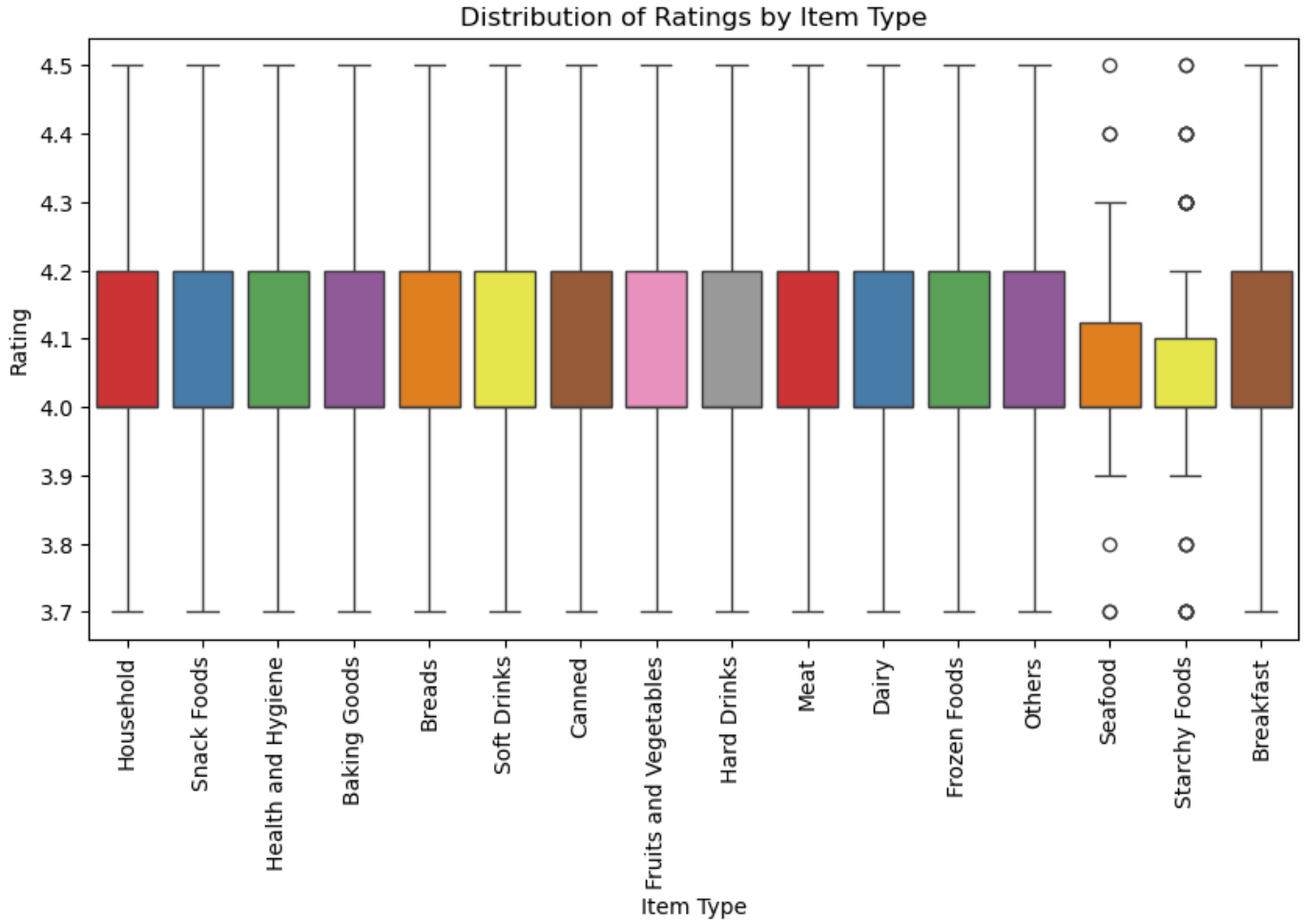
Observation

- The average sales across all outlet types appear to be quite similar.
- Supermarket Type1 shows a slightly higher average sales compared to other outlet types.
- Grocery Store and Supermarket Type2 have nearly equal average sales values.
- Supermarket Type3 also performs well, maintaining sales close to the overall average.
- Overall, all outlet types contribute evenly, with Supermarket Type1 having a small edge

2. Which item types receive better ratings?

In [63]:

```
plt.figure(figsize=(10,5))
sns.boxplot(x='Item Type', y='Rating', data=df, palette='Set1')
plt.xticks(rotation=90)
plt.title('Distribution of Ratings by Item Type')
plt.show()
```



Observation

- Most item types have ratings centered around 4.1 to 4.2, showing consistent quality.
- Categories like Fruits and Vegetables and Breakfast have slightly higher ratings.
- Seafood and Starchy Foods show more variation and a few lower-rated outliers

3. Is there any relationship between item weight and sales?

In [64]:

```
plt.figure(figsize=(6,4))
sns.scatterplot(x='Item Weight', y='Sales', data=df, color='coral')
plt.title('Item Weight vs Sales')
plt.show()
```

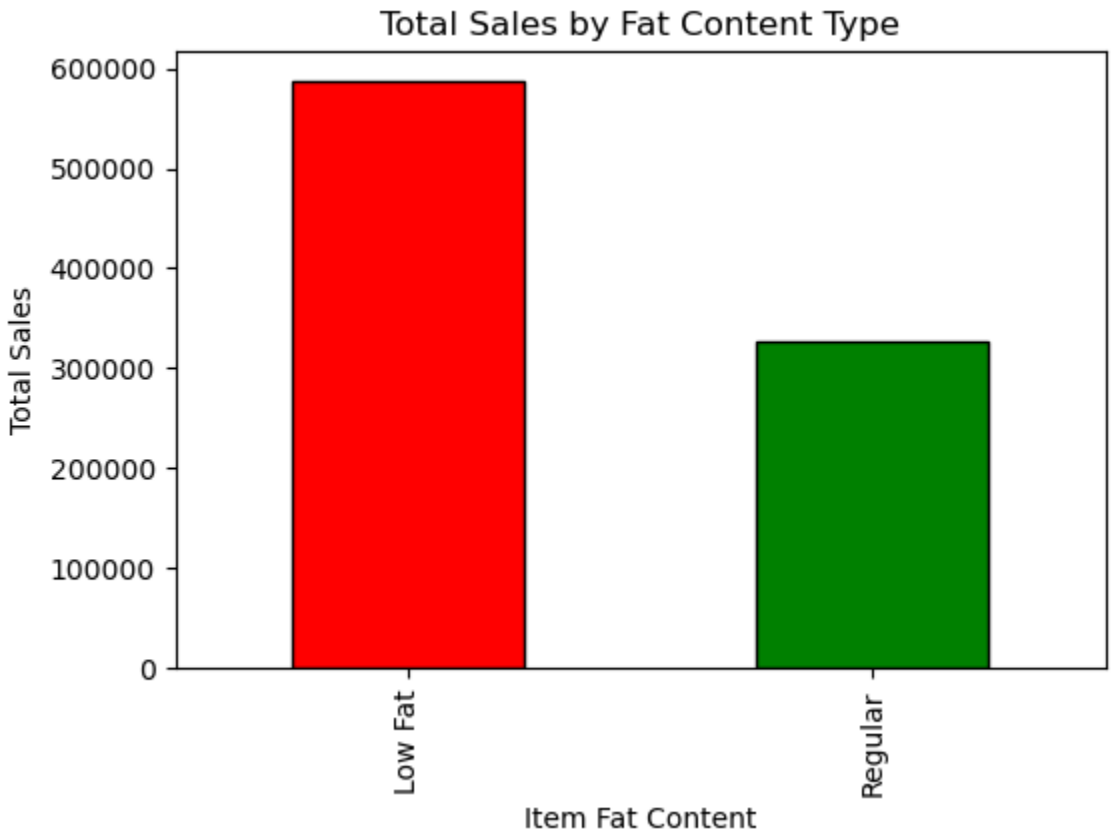
Observation

- There is no clear relationship between item weight and sales — the points are widely scattered.
- This suggests that item weight does not significantly impact sales in the Blinkit dataset.

4. How does the total sales vary between Low Fat and Regular items in the Blinkit dataset?

In [73]:

```
df.groupby('Item Fat Content')['Sales'].sum().plot(
    kind='bar', color=['red', 'green'], edgecolor='Black', figsize=(6,4)
)
plt.title('Total Sales by Fat Content Type')
plt.xlabel('Item Fat Content')
plt.ylabel('Total Sales')
plt.show()
```



Observation

- Regular items show higher total sales compared to Low Fat items.
- Customers may prefer Regular (full-fat) products, or these items might be priced higher.
- Low Fat items still contribute significantly to sales, showing interest in healthier options.
- The sales gap between the two categories helps identify product preference and pricing impact.

Multivariate Analysis

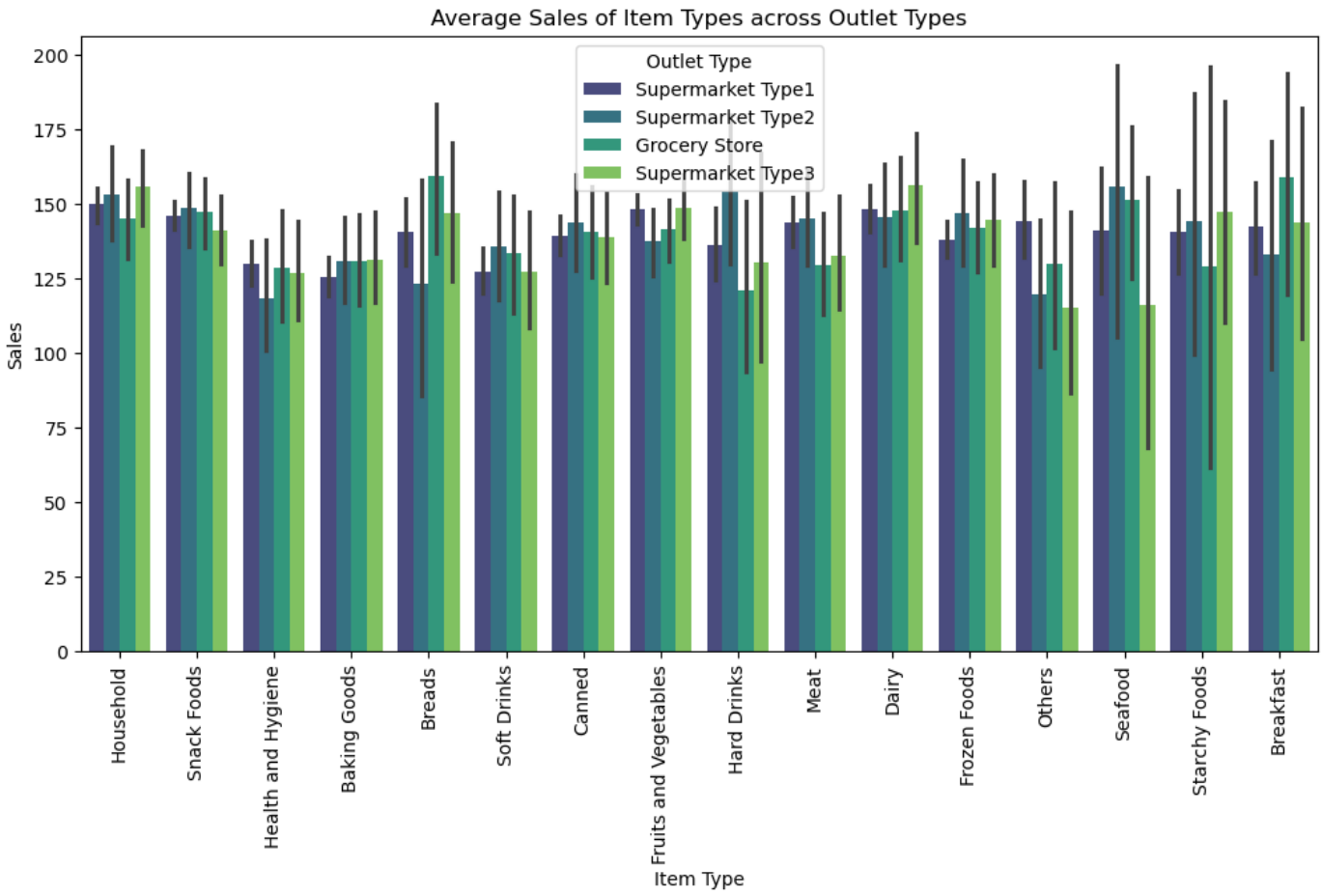
- Multivariate analysis studies three or more variables at the same time to understand complex relationships and combined effects. It gives a broader view of how multiple factors interact with each other.

1. How do different item types perform in sales across outlet types?

In [65]:

```
plt.figure(figsize=(12,6))
sns.barplot(
    x='Item Type',
    y='Sales',
    hue='Outlet Type',
    data=df,
    estimator='mean',
    palette='viridis'
)
```

```
plt.xticks(rotation=90)
plt.title('Average Sales of Item Types across Outlet Types')
plt.show()
```



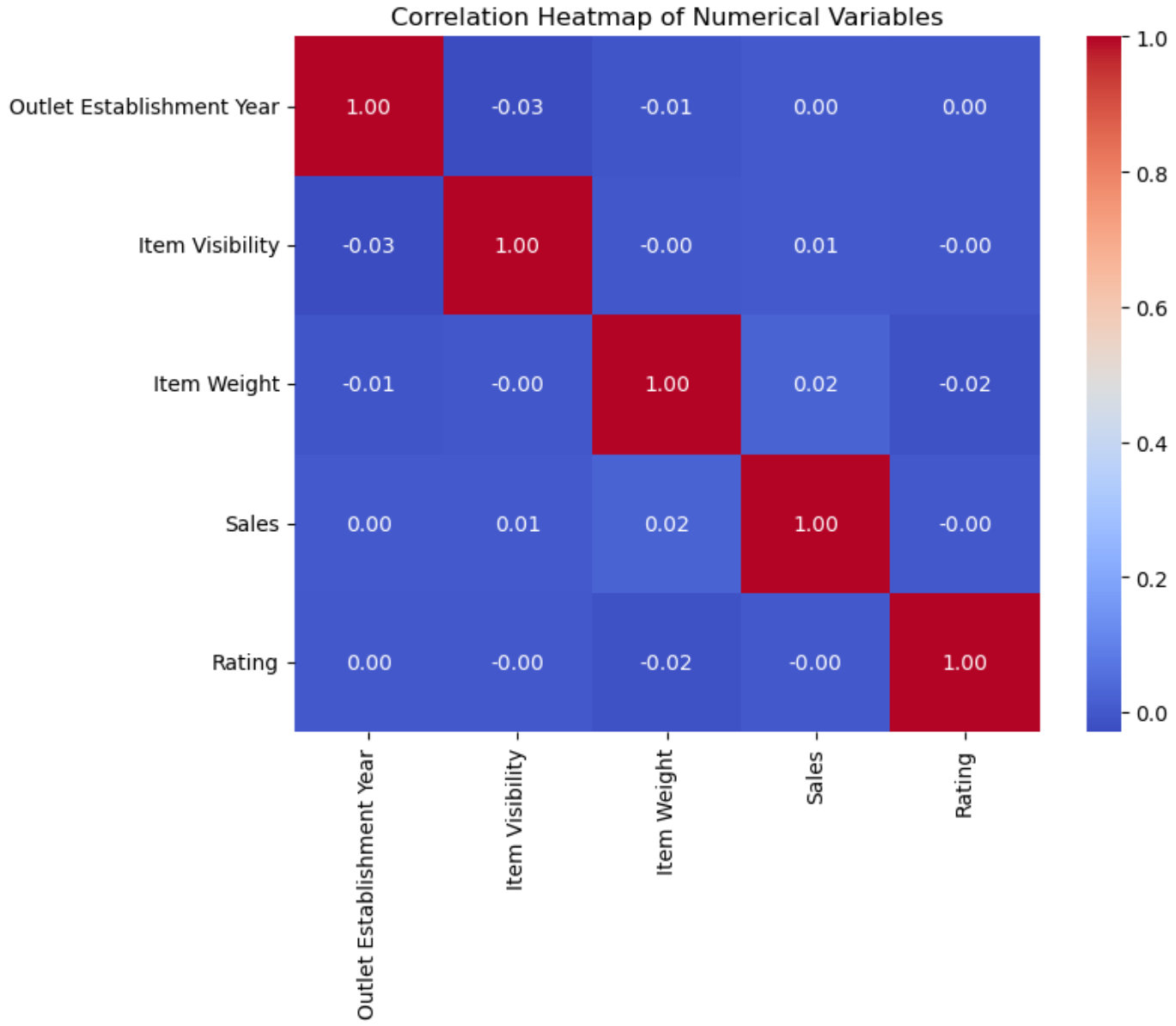
Observation

- Supermarket Type1 outlets show the highest average sales across most item types, indicating their strong performance.
- Supermarket Type2 outlets also perform well but slightly lower than Type1.
- Grocery Store and Supermarket Type3 outlets have lower sales, suggesting smaller or less popular outlets contribute less to overall sales.

2. What is the relationship among numerical features?

In [66]:

```
# Select only numerical columns
num_df = df.select_dtypes(include=['int64', 'float64'])
# Plot correlation heatmap
plt.figure(figsize=(8,6))
sns.heatmap(num_df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap of Numerical Variables')
plt.show()
```



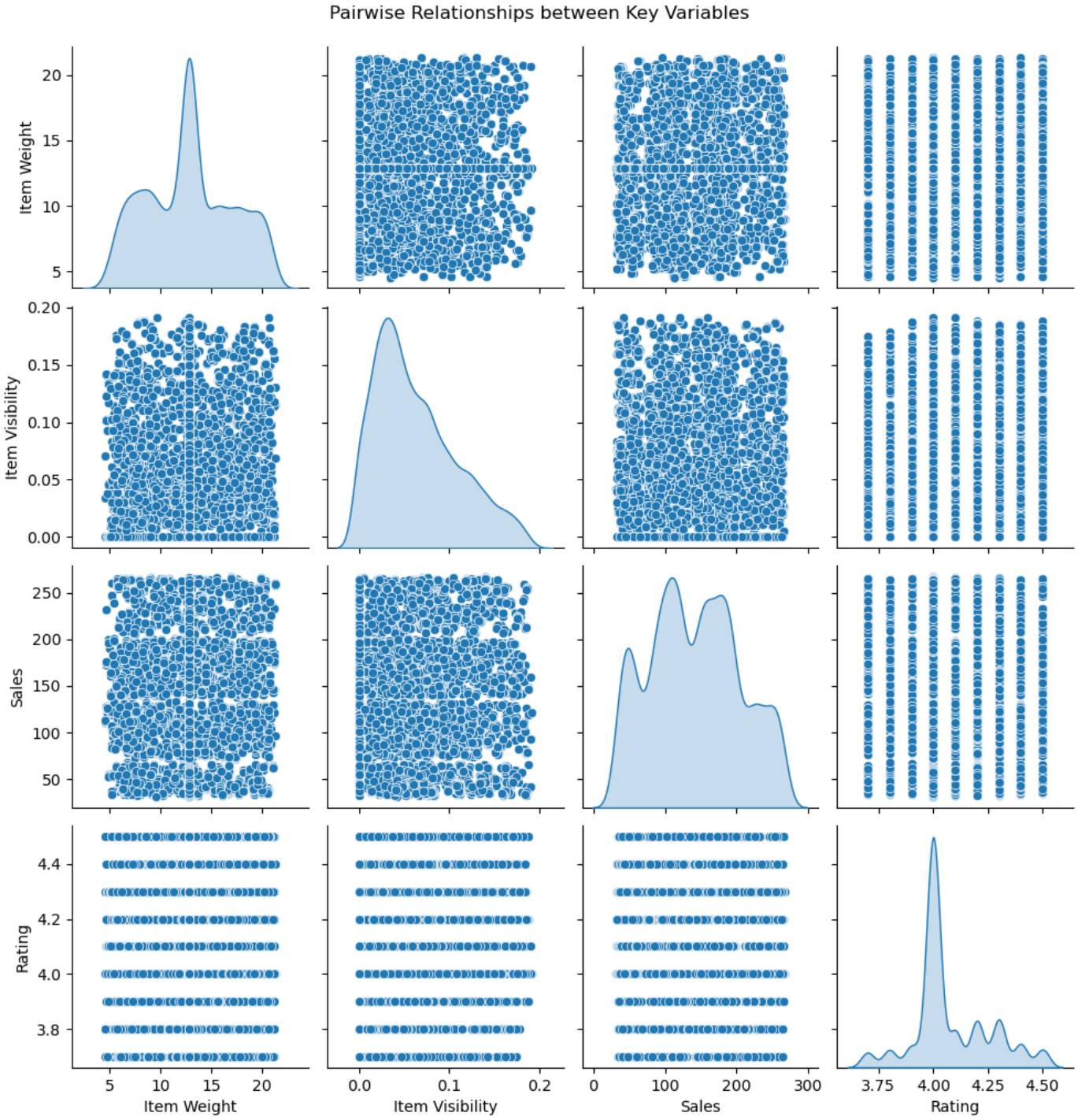
Observation

- The heatmap shows that most numerical variables have weak to moderate correlations with each other.
- Sales has a positive correlation with Item Weight and Rating, suggesting that heavier or better-rated items may sell slightly more.
- Item Visibility shows a weak or negative correlation with other variables, indicating it does not strongly influence sales.

3. Does outlet size affect the sales of low-fat vs regular-fat items?

In [67]:

```
sns.pairplot(df[['Item Weight', 'Item Visibility', 'Sales', 'Rating']], diag_kind='kde')
plt.suptitle('Pairwise Relationships between Key Variables', y=1.02)
plt.show()
```



4.How does Item Weight affect Sales across different Outlet Types, and what role does Rating (shown by point size) play in this relationship?

In [68]:

```
sns.scatterplot(x='Item Weight', y='Sales', hue='Outlet Type', size='Rating', data=df, p
plt.title('Sales vs Item Weight across Outlet Types')
plt.show()
```

Observation

- There is no strong relationship between item weight and sales — sales remain scattered across all weights.
- Outlet Type doesn't greatly affect the pattern, as all outlet types show similar spread in sales.
- Rating (point size) also doesn't show a major influence, indicating that neither heavier items nor higher ratings lead to noticeably higher sales.

In [ ]:

In [ ]:

In [ ]:

In [ ]: