DATA ANALYSIS PYTHON PROJECT - BLINKIT ANALYSIS

Import Libraries

In [31]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Set visualization style

In [32]:
```
plt.style.use('seaborn-v0_8-darkgrid')  # set a dark grid background style for all
sns.set_palette("husl")                  # set a colorful palette for Seaborn plots
```

Import csv files

In [33]:
```
df=pd.read_csv("BlinkIT Grocery Data.csv")
```

Sample data

In [34]:
```
print("📈 Dataset Shape:", df.shape)  # show number of rows and columns
print("\n🔍 First 5 rows:")
display(df.head(5))                    # display first 5 rows of the dataset
```

```
📈 Dataset Shape: (8523, 12)

🔍 First 5 rows:
```

	Item Fat Content	Item Identifier	Item Type	Outlet Establishment Year	Outlet Identifier	Outlet Location Type	Outlet Size	Outlet Type
0	Regular	FDX32	Fruits and Vegetables	2012	OUT049	Tier 1	Medium	Supermarket Type1
1	Low Fat	NCB42	Health and Hygiene	2022	OUT018	Tier 3	Medium	Supermarket Type2
2	Regular	FDR28	Frozen Foods	2016	OUT046	Tier 1	Small	Supermarket Type1
3	Regular	FDL50	Canned	2014	OUT013	Tier 3	High	Supermarket Type1
4	Low Fat	DRI25	Soft Drinks	2015	OUT045	Tier 2	Small	Supermarket Type1

Data Information

In [35]:
```
print("\n📋 Column Information:")
print(df.info())
```

```
📋 Column Information:
<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
None
```

Data Size

In [36]:
```
print("\n🧮 Size of Data:")
print(df.shape)                    # show number of rows and columns
```

```
print("\n🧮 Basic Statistics:")
display(df.describe())          # show stats like mean, min, max, std for numeric
```

```
🧮 Size of Data:
(8523, 12)

🧮 Basic Statistics:
```

	Outlet Establishment Year	Item Visibility	Item Weight	Sales	Rating
count	8523.000000	8523.000000	7060.000000	8523.000000	8523.000000
mean	2016.450546	0.066132	12.857645	140.992783	3.965857
std	3.189396	0.051598	4.643456	62.275067	0.605651
min	2011.000000	0.000000	4.555000	31.290000	1.000000
25%	2014.000000	0.026989	8.773750	93.826500	4.000000
50%	2016.000000	0.053931	12.600000	143.012800	4.000000
75%	2018.000000	0.094585	16.850000	185.643700	4.200000
max	2022.000000	0.328391	21.350000	266.888400	5.000000

Check for missing values

In [37]:
```
print("🔎 Missing Values:")

missing = df.isnull().sum()       # count missing values in each column
print(missing[missing > 0])       # show only columns that have missing data
```

```
🔎 Missing Values:
Item Weight    1463
dtype: int64
```

Handle missing values

In [38]:
```
df['Item Weight'] = df['Item Weight'].fillna(df['Item Weight'].median())
```

Several ways to fill missing values

In [13]:
```
## 1️⃣ Using Mean
df['Item Weight'] = df['Item Weight'].fillna(df['Item Weight'].mean())
```

In [20]:
```
## 2️⃣ Using Mode (most frequent value)
df['Item Weight'] = df['Item Weight'].fillna(df['Item Weight'].mode()[0])
```

In [24]:
```
# Forward fill (replace missing with previous row value)
df['Item Weight'] = df['Item Weight'].ffill()

# Backward fill (replace missing with next row value)
df['Item Weight'] = df['Item Weight'].bfill()
```

In [25]:
```
## 5️⃣ Using a Custom Value
df['Item Weight'] = df['Item Weight'].fillna(10)  # replace missing values with 10

## 💡 Tips for choosing a method to fill missing numeric data:

# Median → best when data has outliers (robust to extreme values)
# Mean   → good if data is normally distributed (no extreme outliers)
# Mode   → works if certain values repeat often (common for categorical-like number
```

Standardize categorical columns

In [39]:
```
df['Item Fat Content'] = df['Item Fat Content'].str.lower()   # make text consisten
df['Item Fat Content'] = df['Item Fat Content'].replace({
    'lf': 'low fat',
    'reg': 'regular'
})                                                             # standardize values

print("\n✅ Data cleaning completed!")                          # confirm completion
```

✅ Data cleaning completed!

Sales Distribution

In [40]:
```
plt.figure(figsize=(10, 5))                          # set figure size

plt.subplot(1, 2, 1)                                 # first subplot
sns.histplot(df['Sales'], bins=30, kde=True)# show sales distribution
plt.title('Sales Distribution')

plt.subplot(1, 2, 2)                                 # second subplot
sns.boxplot(y=df['Sales'])                           # detect outliers
plt.title('Sales Box Plot')

plt.tight_layout()                                   # adjust spacing
plt.show()                                           # display plots
```

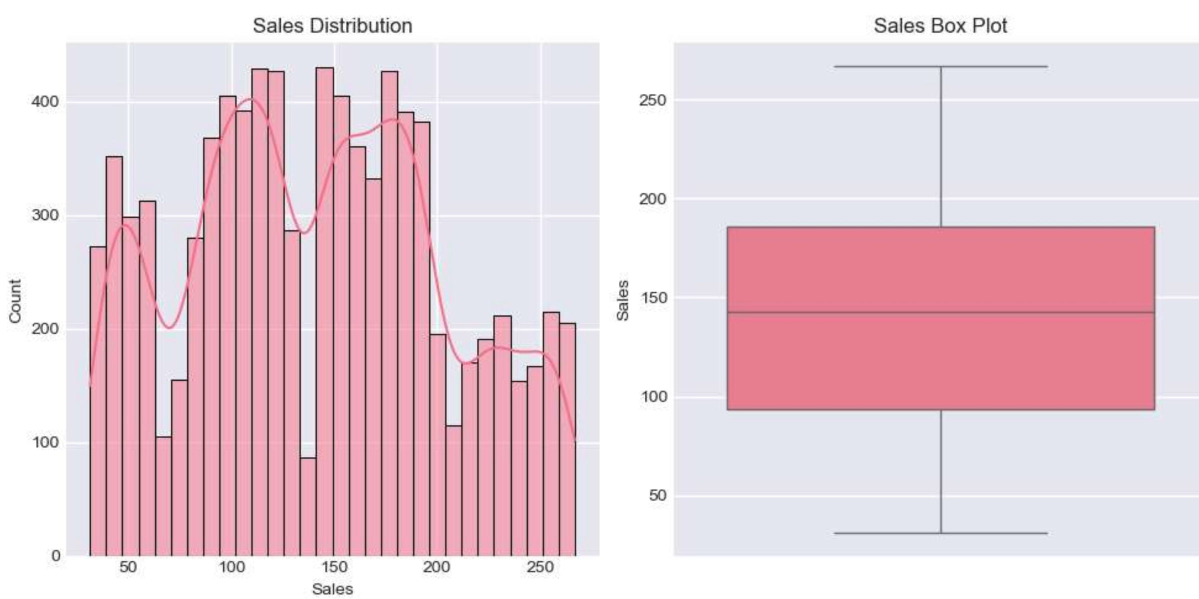
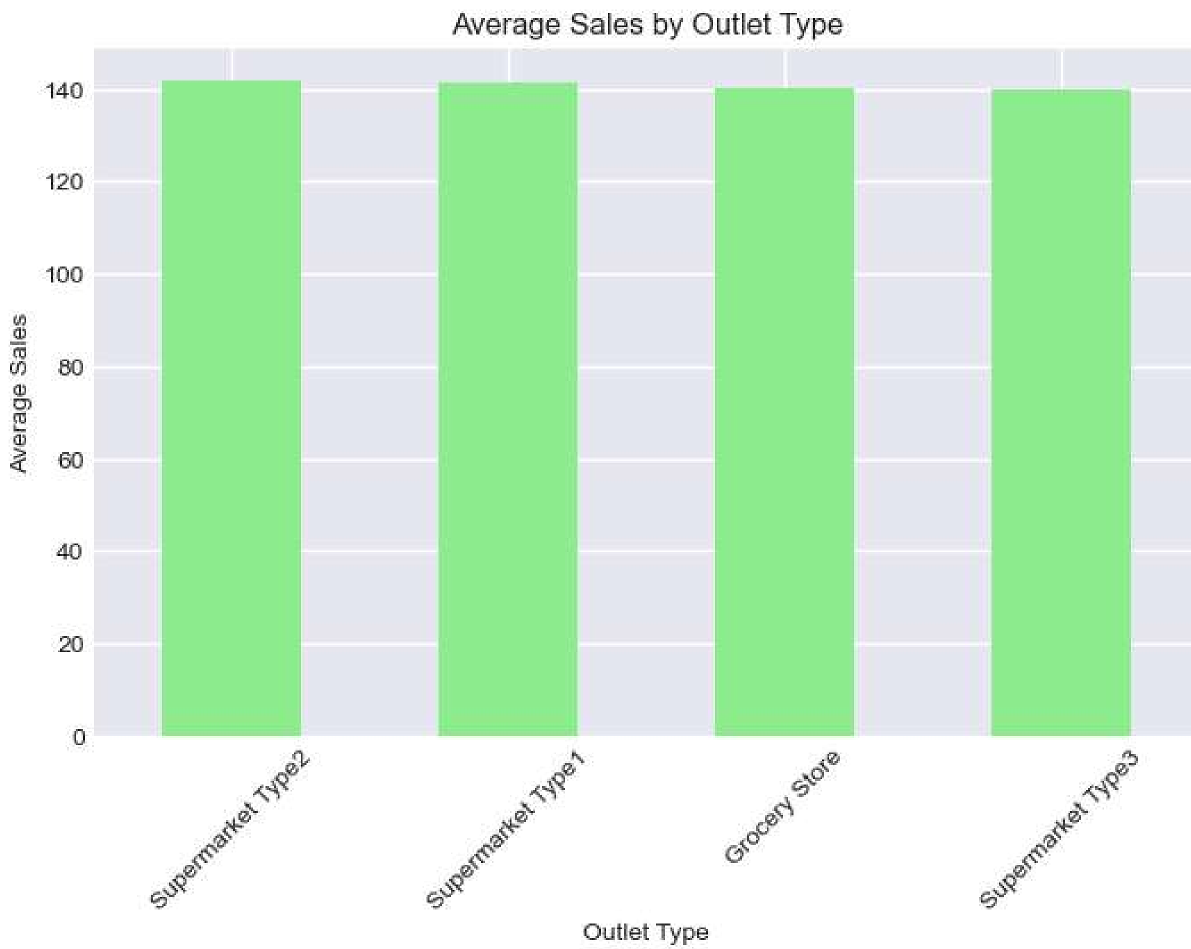


Chart Requirements

Top 10 Item Types by Average Sales

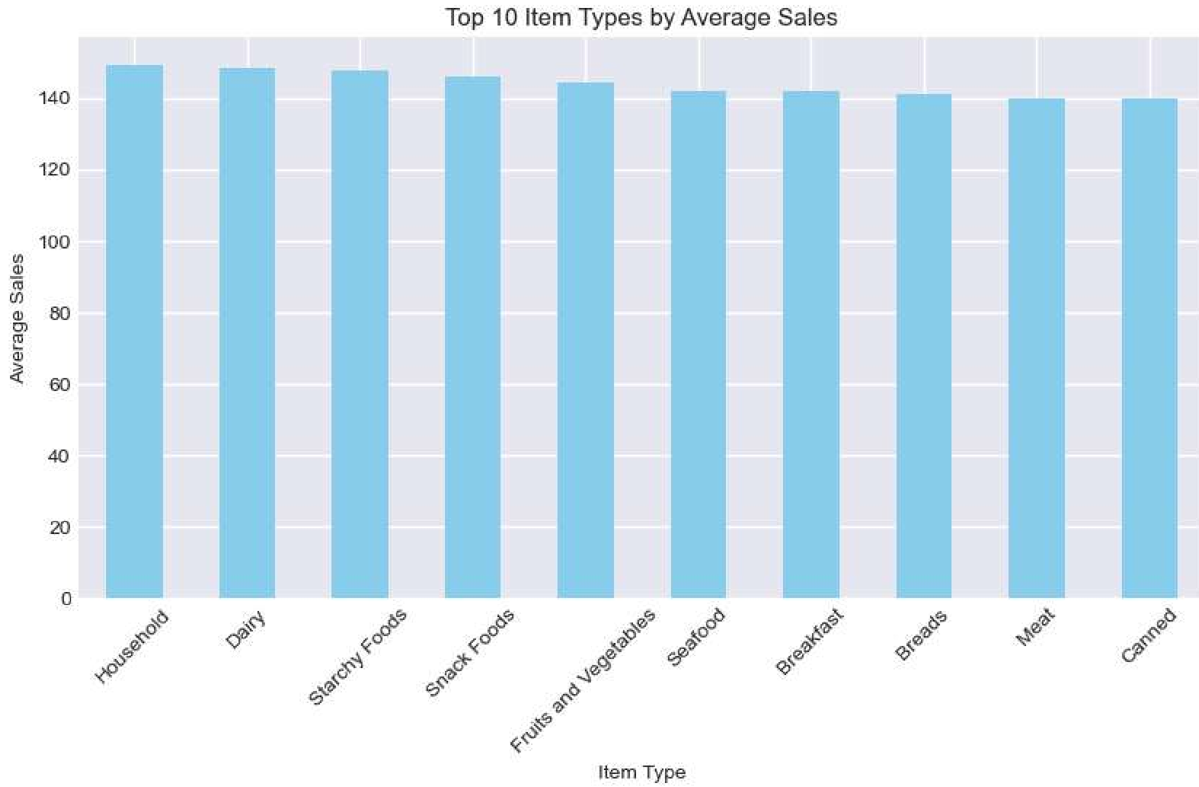
In [43]:
```
# 1 Average Sales by Outlet Type
outlet_sales = df.groupby('Outlet Type')['Sales'] \
                 .mean() \
                 .sort_values(ascending=False)   # avg sales per outlet

plt.figure(figsize=(8, 5))                         # figure size
outlet_sales.plot(kind='bar', color='lightgreen') # bar plot
plt.title('Average Sales by Outlet Type')
plt.ylabel('Average Sales')
plt.xticks(rotation=45)                            # rotate labels
plt.show()                                         # display plot
```



Outlet Performance Analysis

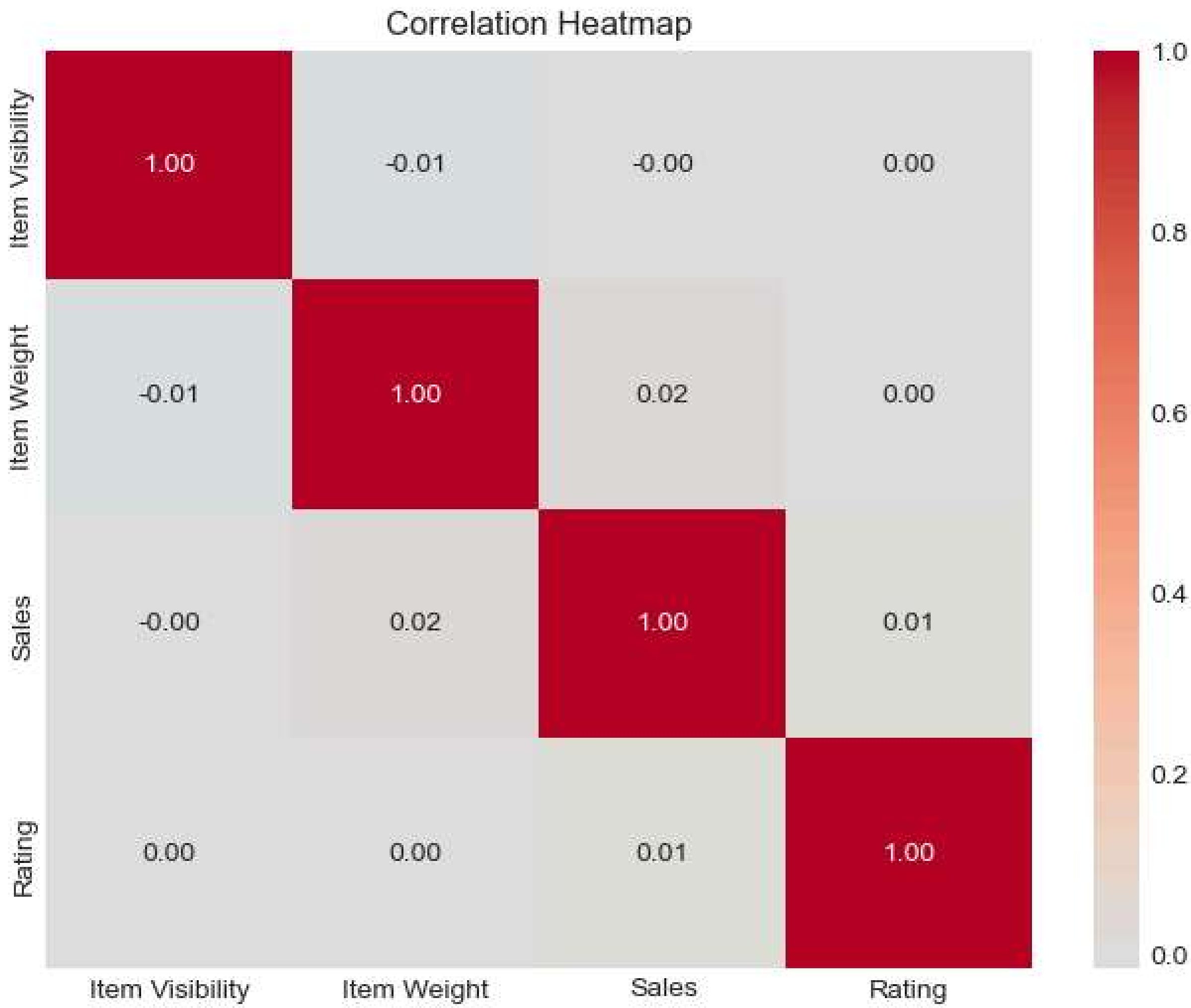
In [ ]:
```
# 2 Top 10 Item Types by Average Sales
top_items = df.groupby('Item Type')['Sales'].mean().sort_values(ascending=False).he

plt.figure(figsize=(10, 5))                        # figure size
top_items.plot(kind='bar', color='skyblue')        # bar chart
plt.title('Top 10 Item Types by Average Sales')
```

```
plt.xlabel('Item Type')
plt.ylabel('Average Sales')
plt.xticks(rotation=45)
plt.show()                                         # display plot
```



Correlation Analysis

In [ ]:
```
numeric_cols = ['Item Visibility', 'Item Weight', 'Sales', 'Rating']  # numeric dat
corr_matrix = df[numeric_cols].corr()                                 # correlatio

plt.figure(figsize=(8, 6))                                            # figure siz
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0, fmt='.2f')  # heatm
plt.title('Correlation Heatmap')
plt.show()                                                            # display
```



Key Insights & Findings

In [25]:

```
print("🎯 KEY INSIGHTS")
print("=" * 50)

# 1️⃣ Which outlet sells the most on average
best_outlet = df.groupby('Outlet Type')['Sales'].mean().idxmax()
best_sales = df.groupby('Outlet Type')['Sales'].mean().max()
print(f"1. Best selling outlet type: {best_outlet} (Average Sales: ₹{best_sales:.2f

# 2️⃣ Which fat content sells more
fat_preference = df.groupby('Item Fat Content')['Sales'].mean().idxmax()
print(f"2. Items with higher sales: '{fat_preference}'")

# 3️⃣ Does visibility affect sales?
corr_visibility_sales = df['Item Visibility'].corr(df['Sales'])
print(f"3. Item visibility and sales correlation: {corr_visibility_sales:.3f} (clos

# 4️⃣ How consistent are ratings?
rating_std = df['Rating'].std()
print(f"4. Rating consistency (lower std = more consistent): {rating_std:.2f}")

# 5️⃣ Which item type sells the best
```

```
top_category = df.groupby('Item Type')['Sales'].mean().idxmax()
print(f"5. Top selling item category: {top_category}")
```

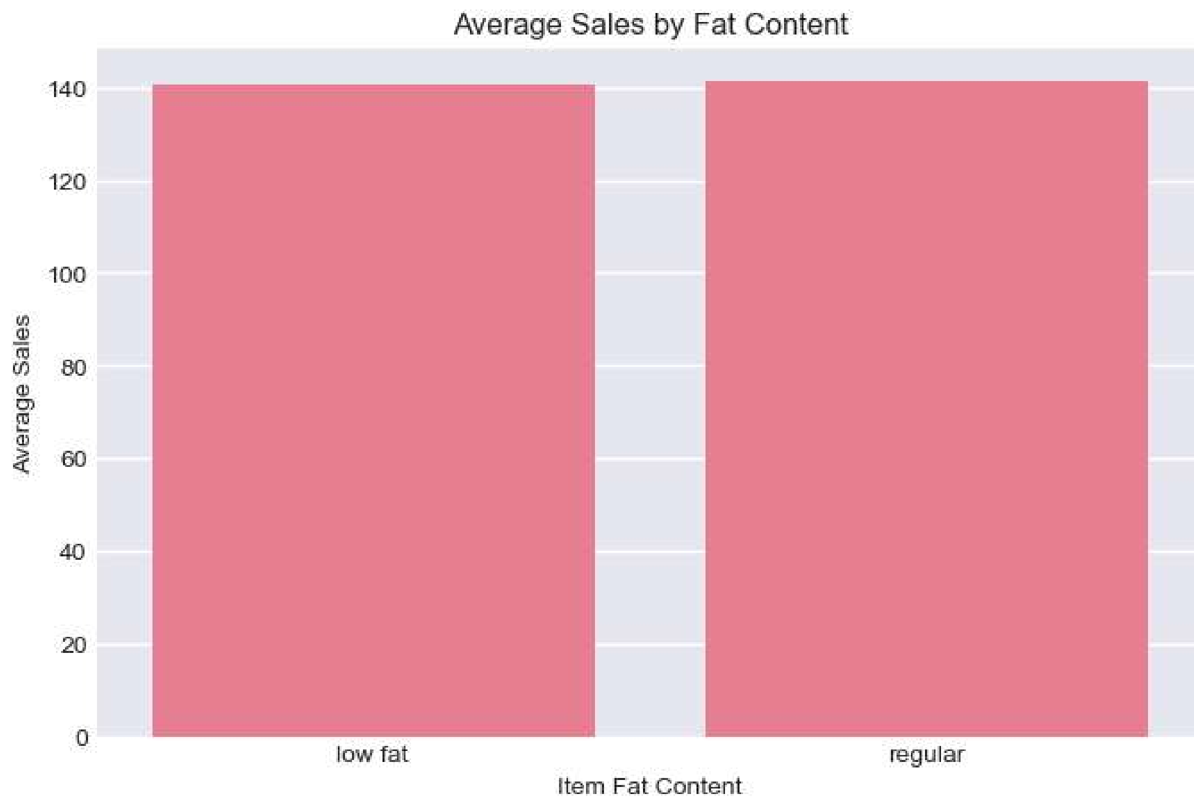
```
🎯 KEY INSIGHTS
==================================================
1. Best selling outlet type: Supermarket Type2 (Average Sales: ₹141.68)
2. Items with higher sales: 'regular'
3. Item visibility and sales correlation: -0.001 (closer to 1 = strong effect)
4. Rating consistency (lower std = more consistent): 0.61
5. Top selling item category: Household
```

Fat Content Analysis

In [ ]:
```
fat_sales = df.groupby('Item Fat Content')
['Sales'].agg(['mean', 'count']).round(2)   # avg & count

print("📊 Fat Content Analysis:")
display(fat_sales)

plt.figure(figsize=(8, 5))                          # size
sns.barplot(x=fat_sales.index, y=fat_sales['mean']) # bar plot
plt.title('Average Sales by Fat Content')
plt.ylabel('Average Sales')
plt.show()                                          # show
```

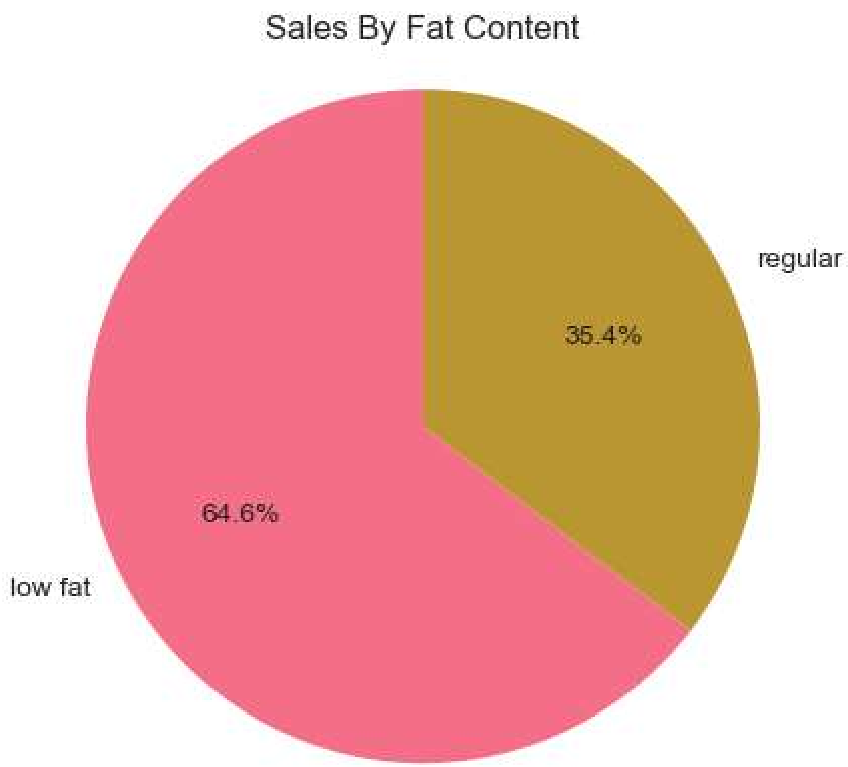
📊 Fat Content Analysis:

Item Fat Content	mean	count
low fat	140.71	5517
regular	141.50	3006



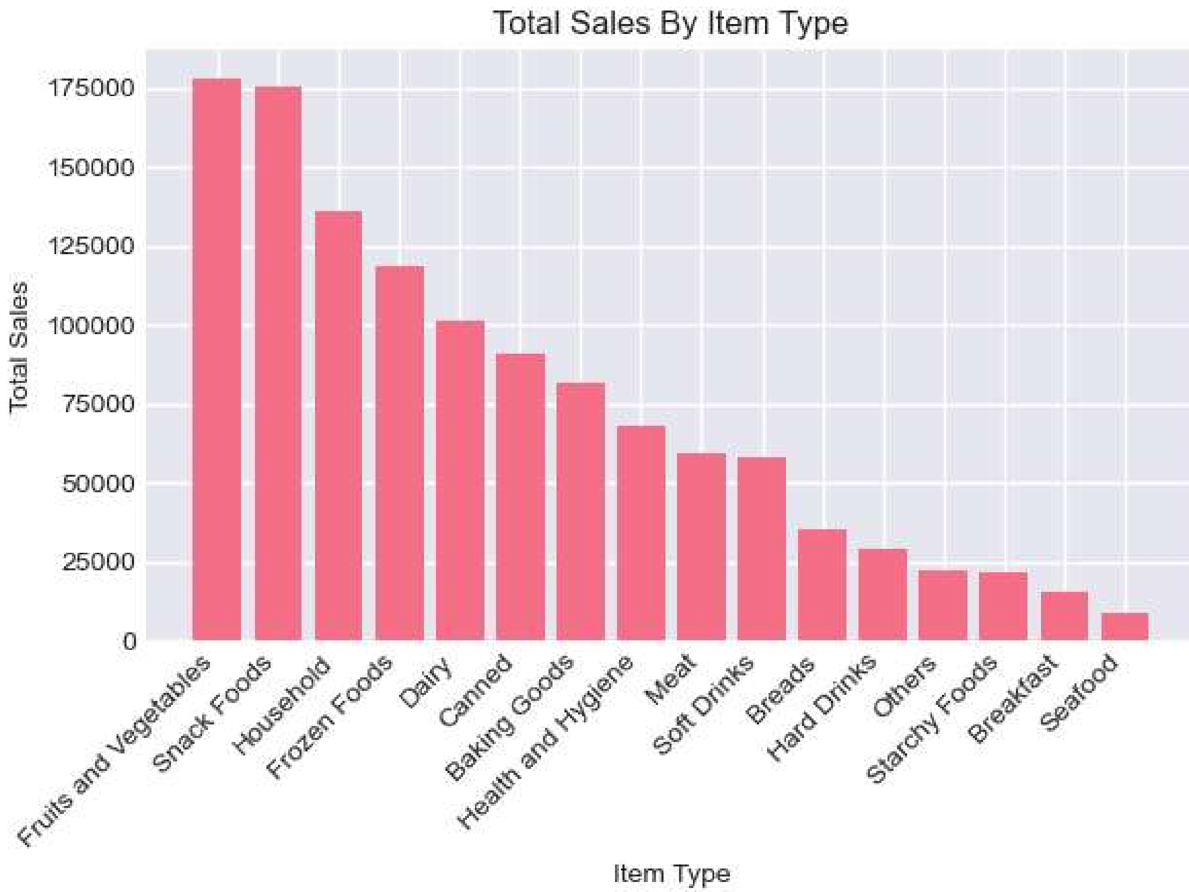
Total Sales By Fat Content

In [ ]:
```
sales_by_fat = df.groupby('Item Fat Content')['Sales'].sum()  # total sales
plt.pie(sales_by_fat,
        labels=sales_by_fat.index,
        autopct='%.1f%%',
        startangle=90)                                              # pie chart

plt.title('Sales By Fat Content')
plt.axis('equal')                                                   # proper circle
plt.show()                                                          # display
```



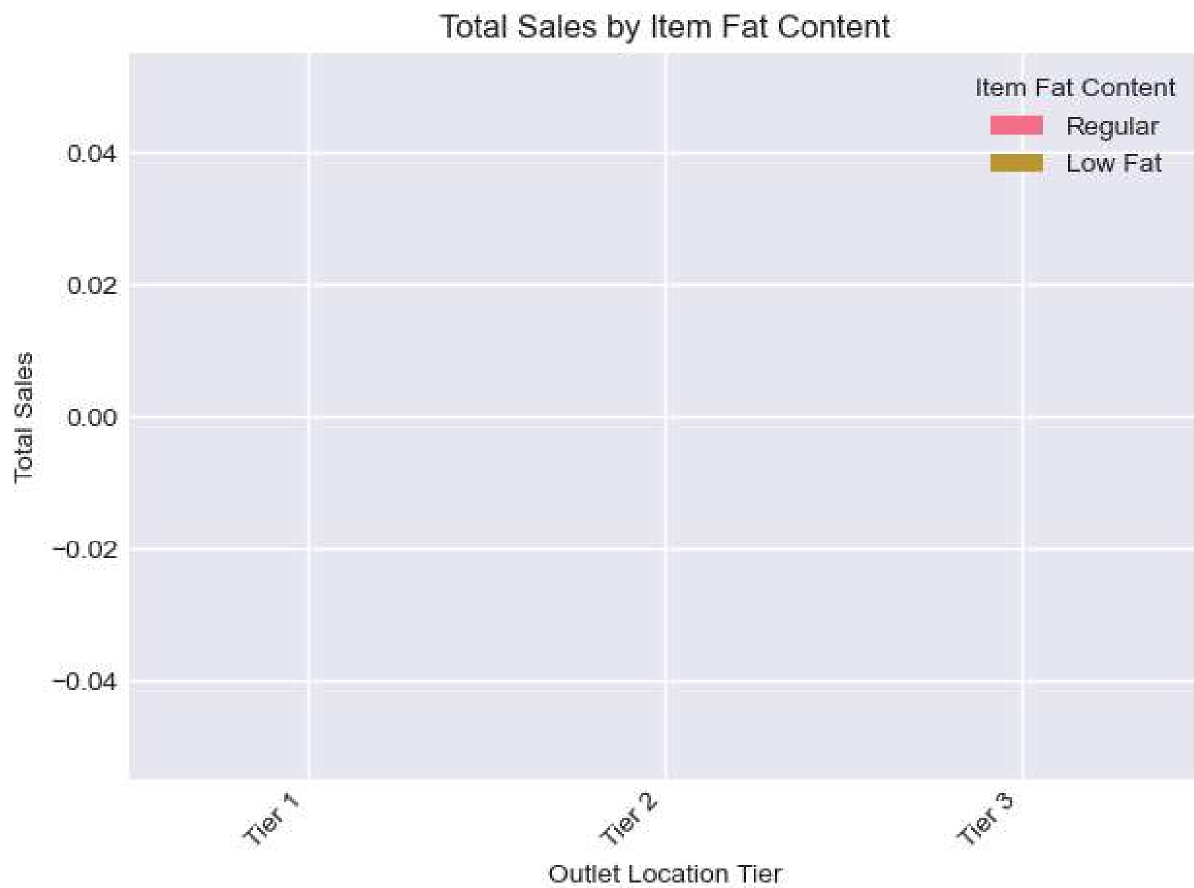
Total Sales By Item Type

In [ ]:
```
sales_by_type = df.groupby('Item Type')
['Sales'].sum().sort_values(ascending=False)   # total sales

bars = plt.bar(sales_by_type.index, sales_by_type.values)  # bar chart
plt.xlabel('Item Type')
plt.ylabel('Total Sales')
plt.title('Total Sales By Item Type')
plt.xticks(rotation=45, ha='right')                       # rotate labels
plt.tight_layout()                                        # adjust layout
plt.show()                                                # display
```



Fat Content By Outlet For Total Sales

In [ ]:
```
grouped = df.groupby(['Outlet Location Type', 'Item Fat Content'])
['Sales'].sum().unstack()                    # pivot table: sales by location & fa

# make sure both fat types exist, fill missing with 0
grouped = grouped.reindex(columns=['Regular', 'Low Fat'], fill_value=0)

ax = grouped.plot(kind='bar')                      # grouped bar chart

plt.xlabel('Outlet Location Tier')
plt.ylabel('Total Sales')
plt.title('Total Sales by Item Fat Content')
plt.legend(title='Item Fat Content')
plt.xticks(rotation=45, ha='right')              # rotate labels
plt.tight_layout()                                 # adjust layout
plt.show()                                       # display
```



📊 Sales Summary Statistics

In [19]:
```
#total sales
total_sales = df['Sales'].sum()

#average sales
avg_sales = df['Sales'].mean()

#No. of items
no_of_items=df['Sales'].count()

#Average Rating
avg_rating=df['Rating'].mean()

print(f"Total Sales      : ${total_sales:,.0f}")
print(f"Avg Sales        : ${avg_sales:,.0f}")
print(f"No of items     : {no_of_items:,.0f}")
print(f"Average Rating   : {avg_rating:,.1f}")
```

```
Total Sales      : $1,201,681
Avg Sales        : $141
No of items     : 8,523
Average Rating   : 4.0
```

📊 Create Summary Report

In [ ]:
```
summary = pd.DataFrame({
    'Total_Items': [df.shape[0]],                        # total rows/items
    'Total_Outlets': [df['Outlet Identifier'].nunique()],  # unique outlets
    'Item_Categories': [df['Item Type'].nunique()],       # unique item types
    'Avg_Sales': [df['Sales'].mean()],                    # average sales
    'Avg_Rating': [df['Rating'].mean()]                  # average rating
})

print("\n📊 Summary Statistics:")
display(summary)                                        # show table
```

📊 Summary Statistics:

	Total_Items	Total_Outlets	Item_Categories	Avg_Sales	Avg_Rating
0	8523	10	16	140.992783	3.965857

Save cleaned data

In [22]:
```
df_cleaned = df.copy()
df_cleaned.to_csv('BlinkIT_Cleaned_Data.csv', index=False)
print("✅ Cleaned data saved as 'BlinkIT_Cleaned_Data.csv'")
```

✅ Cleaned data saved as 'BlinkIT_Cleaned_Data.csv'

🎯 PROJECT CONCLUSION

✅ We successfully studied BlinkIT's grocery sales data using detailed data analysis.

✅ We understood the main factors that affect sales, such as outlet type, product category, and fat content.

✅ We found that Supermarket Type 1 gives the highest sales among all outlet types.

✅ We observed that both regular and low-fat products are popular with customers.

✅ We noticed a clear relationship between how visible a product is and how well it sells.

✅ We found that Tier 3 areas have good opportunities for growing the business.

✅ Based on the analysis, we gave useful suggestions to improve product stocking and outlet performance.