Exploratory Data Analysis- Blinkit Analysis

import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Import Raw Data

In [2]:
```
df= pd.read_csv("blinkit_data.csv")
```

Sample Data

In [3]:
```
df.head(10)
```

Out[3]:

	Item Fat Content	Item Identifier	Item Type	Outlet Establishment Year	Outlet Identifier	Outlet Location Type	Outlet Size	Outlet T
0	Regular	FDX32	Fruits and Vegetables	2012	OUT049	Tier 1	Medium	Superma Ty
1	Low Fat	NCB42	Health and Hygiene	2022	OUT018	Tier 3	Medium	Superma Ty
2	Regular	FDR28	Frozen Foods	2010	OUT046	Tier 1	Small	Superma Ty
3	Regular	FDL50	Canned	2000	OUT013	Tier 3	High	Superma Ty
4	Low Fat	DRI25	Soft Drinks	2015	OUT045	Tier 2	Small	Superma Ty
5	low fat	FDS52	Frozen Foods	2020	OUT017	Tier 2	Small	Superma Ty
6	Low Fat	NCU05	Health and Hygiene	2011	OUT010	Tier 3	Small	Gro S
7	Low Fat	NCD30	Household	2015	OUT045	Tier 2	Small	Superma Ty
8	Low Fat	FDW20	Fruits and Vegetables	2000	OUT013	Tier 3	High	Superma Ty
9	Low Fat	FDX25	Canned	1998	OUT027	Tier 3	Medium	Superma Ty

Size Of Data

In [4]:
```
print("Size of data :",df.shape)
```

```
Size of data : (8523, 12)
```

Field Info

In [5]:
```
df.columns
```

Out[5]:
```
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')
```

In [6]:
```
df.describe()
```

Out[6]:

	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

Field Data Types

In [7]:
```
df.dtypes
```

Out[7]:
```
Item Fat Content              object
Item Identifier               object
Item Type                     object
Outlet Establishment Year      int64
Outlet Identifier             object
Outlet Location Type          object
Outlet Size                   object
Outlet Type                   object
Item Visibility              float64
Item Weight                  float64
Sales                        float64
Rating                       float64
dtype: object
```

Data Cleaning

In [8]:
```
print(df["Item Fat Content"].unique())
```

```
['Regular' 'Low Fat' 'low fat' 'LF' 'reg']
```

In [9]:
```
df["Item Fat Content"]=df["Item Fat Content"].replace({'low fat':'Low Fat',
                                                       'LF':'Low Fat',
                                                       'reg':'Regular'})
```

In [10]:
```
print(df["Item Fat Content"].unique())
```

```
['Regular' 'Low Fat']
```

Bussiness Requirements

KPI's Requirements

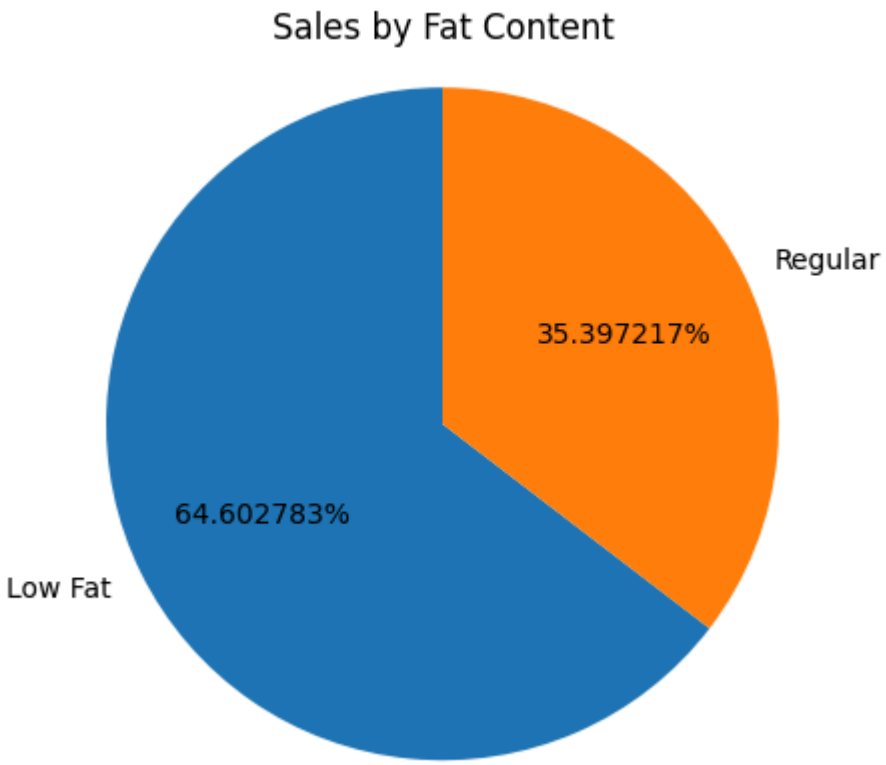
In [11]:
```
#Total Sales
total_sales=df['Sales'].sum()

#Average Sales
average_sales=df['Sales'].mean()

#No Of Items Sold
no_of_items_sold=df['Sales'].count()

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

#Results
print(f"Total Sales: ${total_sales:,.0f}")
print(f"Average Sales: ${average_sales:,.1f}")
print(f"No Of Items Sold: {no_of_items_sold:,.0f}")
print(f"Average Ratings: {average_ratings:,.1f}")
```

```
Total Sales: $1,201,681
Average Sales: $141.0
No Of Items Sold: 8,523
Average Ratings: 4.0
```

Charts Requirements

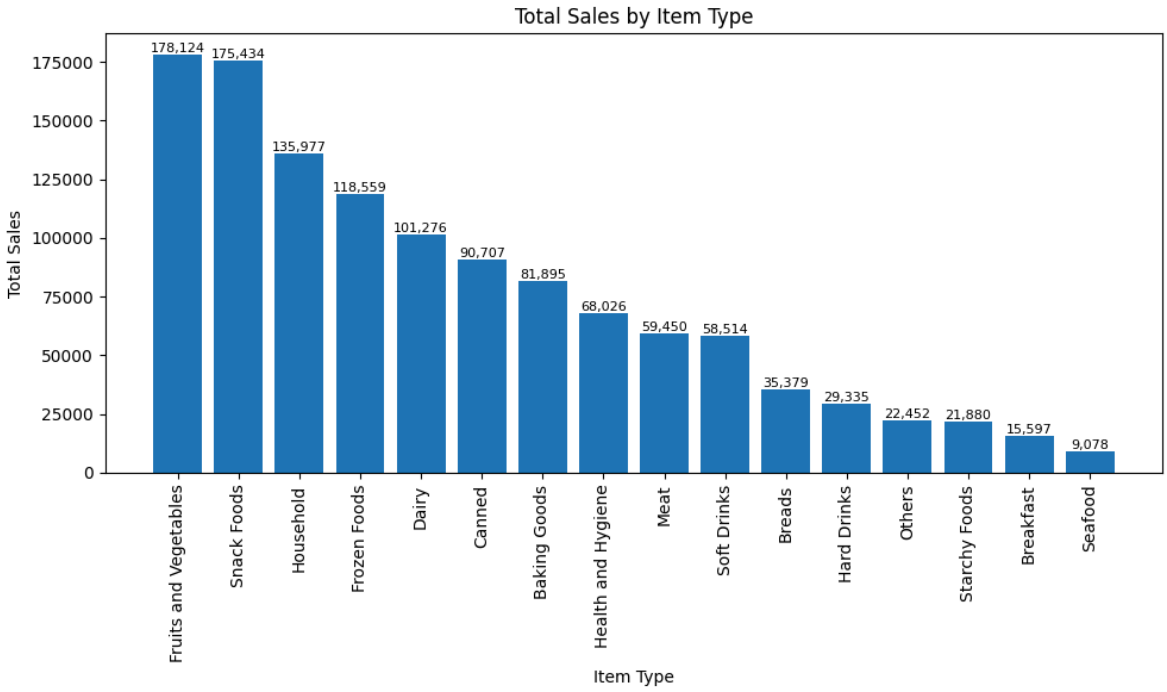
Total Sales by Fat Content

In [12]:
```
sales_by_fat= df.groupby('Item Fat Content')['Sales'].sum()

plt.pie(sales_by_fat, labels= sales_by_fat.index,
                                autopct= '%1f%%',
                                startangle= 90)
plt.title('Sales by Fat Content')
plt.axis('equal')
plt.show()
```



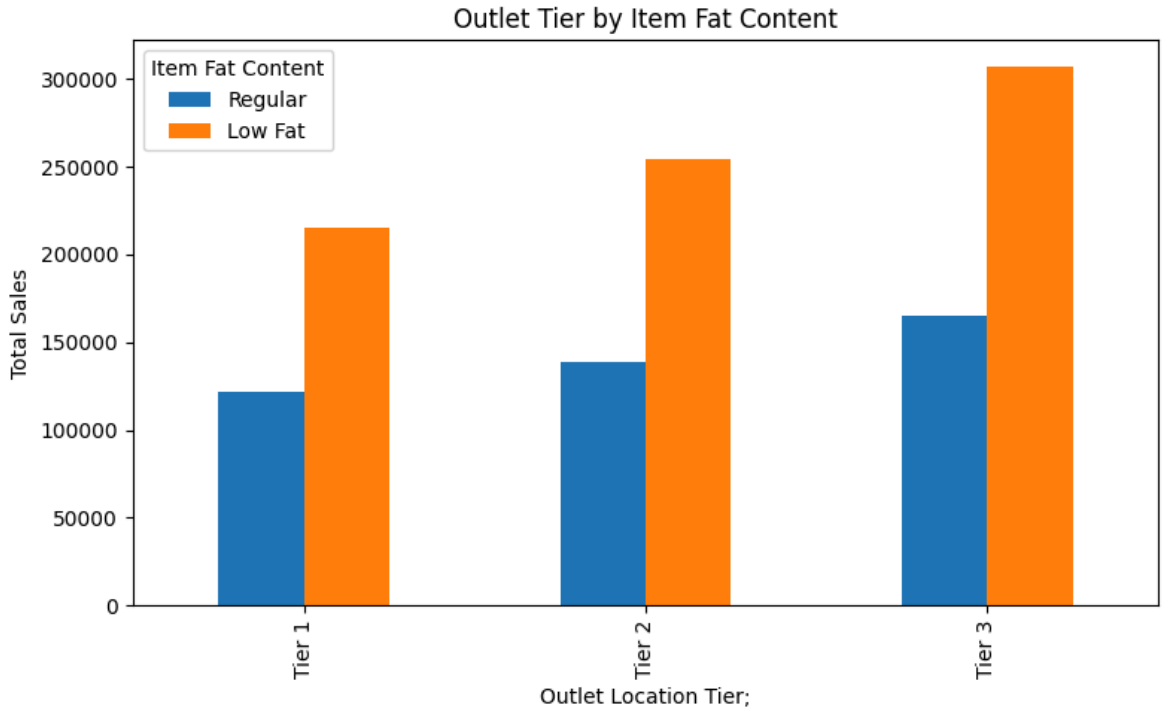
Total Sales by Item Type

In [13]:
```
sales_by_type =df.groupby('Item Type')['Sales'].sum().sort_values(ascending=Fals

plt.figure(figsize=(10,6))
bars = plt.bar(sales_by_type.index, sales_by_type.values)

plt.xticks(rotation=90)
plt.xlabel('Item Type')
plt.ylabel('Total Sales')
plt.title('Total Sales by Item Type')

for bar in bars:
    plt.text(bar.get_x() + bar.get_width()/2 , bar.get_height(),
             f'{bar.get_height():,.0f}', ha='center', va='bottom', fontsize=8)

plt.tight_layout()
plt.show()
```



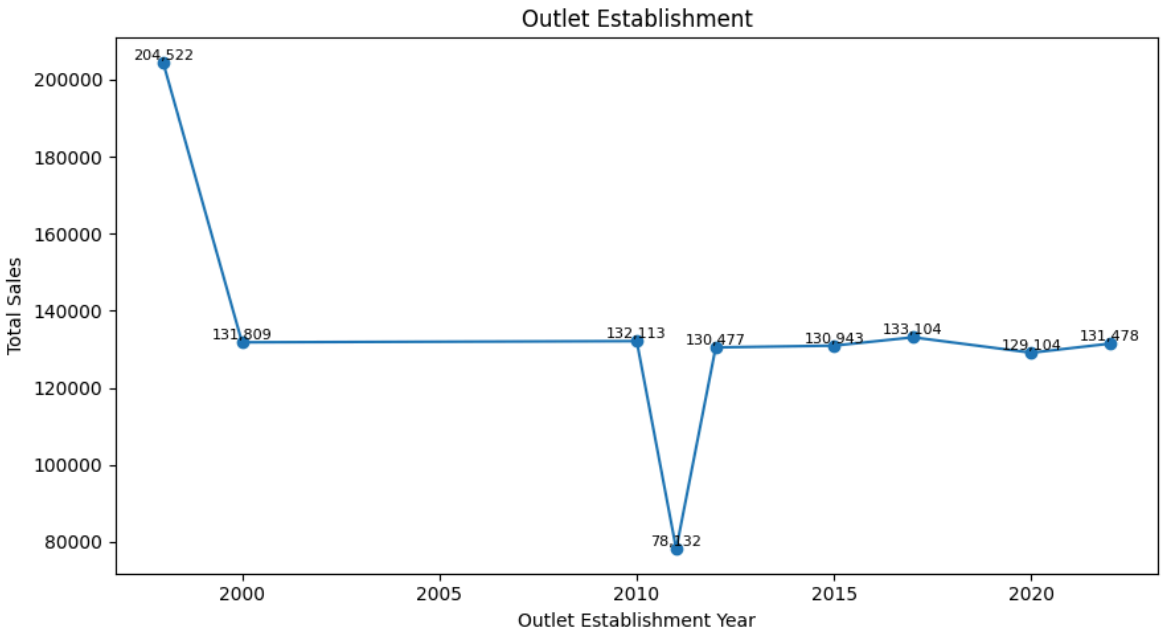
Fat Content by Outlet for Total Sales

In [14]:
```
grouped = df.groupby(['Outlet Location Type', 'Item Fat Content'])['Sales'].sum(
grouped = grouped[['Regular','Low Fat']]

ax= grouped.plot(kind='bar', figsize=(8,5), title='Outlet Tier by Item Fat Conte
plt.xlabel('Outlet Location Tier;')
plt.ylabel('Total Sales')
plt.legend(title="Item Fat Content")
plt.tight_layout()
plt.show()
```



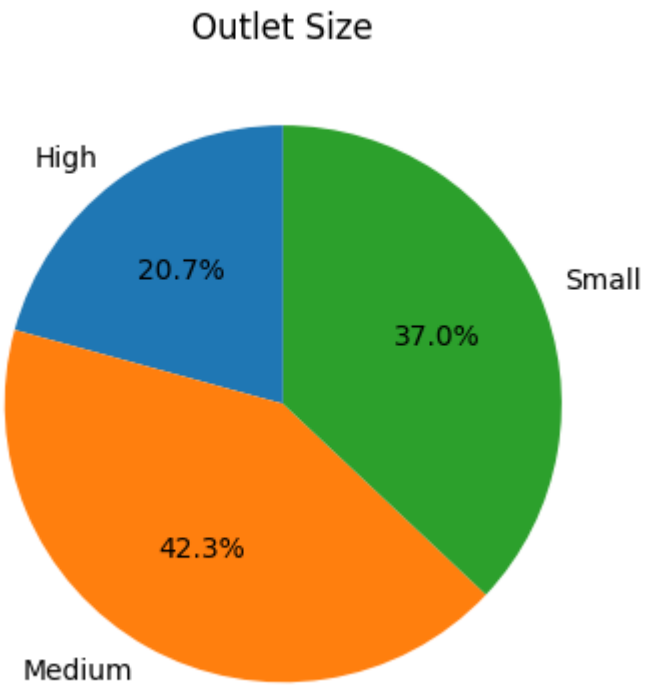
Total Sales by Outlet Establishment

In [15]:
```
sales_by_year = df.groupby('Outlet Establishment Year')['Sales'].sum().sort_inde

plt.figure(figsize=(9,5))
plt.plot(sales_by_year.index, sales_by_year.values, marker='o', linestyle='-')

plt.xlabel('Outlet Establishment Year')
plt.ylabel('Total Sales')
plt.title('Outlet Establishment')

for x, y in zip(sales_by_year.index, sales_by_year.values):
    plt.text(x,y,
             f'{y:,.0f}', ha='center', va='bottom', fontsize=8)

plt.tight_layout()
plt.show()
```



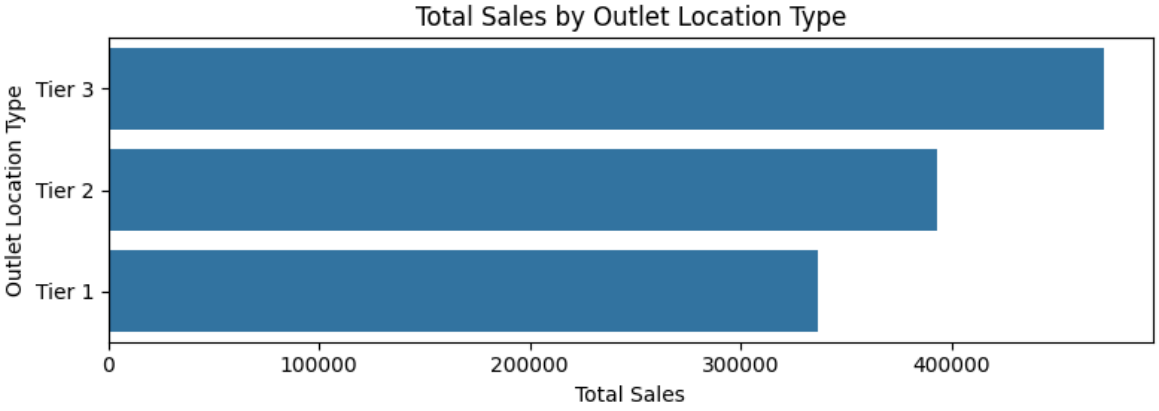
Sales by Outlet Size

In [16]:
```
sales_by_size = df.groupby('Outlet Size')['Sales'].sum()

plt.figure(figsize=(4,4))
plt.pie(sales_by_size, labels=sales_by_size.index,autopct='%1.1f%%', startangle=
plt.title('Outlet Size')
plt.tight_layout()
plt.show()
```



Sales by Outlet Location

In [17]:
```
sales_by_location = df.groupby('Outlet Location Type')['Sales'].sum().reset_inde
sales_by_location = sales_by_location.sort_values('Sales', ascending=False)

plt.figure(figsize=(8, 3))  # Smaller height, enough width
ax = sns.barplot(x='Sales', y='Outlet Location Type', data=sales_by_location)

plt.title('Total Sales by Outlet Location Type')
plt.xlabel('Total Sales')
plt.ylabel('Outlet Location Type')

plt.tight_layout()  # Ensures layout fits without scroll
plt.show()
```



In [ ]: