Blinkit Data Analysis Using Python EDA

Objective:

To conduct a comprehensive analysis of Blinkit's sales performance, customer satisfaction, and inventory distribution to identify key insights and opportunities for optimization using various KPIs and visualizations in Python (Matplotlib, Seaborn).

KPI’s Requirements

- Total Sales
- Average Sales
- Number Of Items
- Average Rating

Chart’s Analysis

- Total Sales by Fat Content
- Total Sales by Item Type
- Fat Content by Outlet for Total Sales
- Total Sales by Outlet Establishment
- Sales by Outlet Size
- Sales by Outlet Locations

Data Analysis Python Quick Commerce Project - Company: Blinkit

Import all the important libraries for this projects

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Import the dataset

In [2]:

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

In [3]:

```
df.head()
```

Out[3]:

	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
0	Regular	FDX32	Fruits and Vegetables	2012	OUT049	Tier 1	Medium	Supermarket Type1	0.100014	15.10	145.4786	5.0
1	Low Fat	NCB42	Health and Hygiene	2022	OUT018	Tier 3	Medium	Supermarket Type2	0.008596	11.80	115.3492	5.0
2	Regular	FDR28	Frozen Foods	2010	OUT046	Tier 1	Small	Supermarket Type1	0.025896	13.85	165.0210	5.0
3	Regular	FDL50	Canned	2000	OUT013	Tier 3	High	Supermarket Type1	0.042278	12.15	126.5046	5.0
4	Low Fat	DRI25	Soft Drinks	2015	OUT045	Tier 2	Small	Supermarket Type1	0.033970	19.60	55.1614	5.0

In [4]:

```
df.sample(10)
```

Out[4]:

	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
1579	Low Fat	FDP38	Canned	2020	OUT017	Tier 2	Small	Supermarket Type1	0.032284	10.100	52.2008	4.3
2817	Low Fat	FDM33	Snack Foods	2010	OUT046	Tier 1	Small	Supermarket Type1	0.087720	15.600	218.5798	4.1
8449	Regular	FDA51	Dairy	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.163882	NaN	113.2518	4.0
7021	Low Fat	FDD50	Canned	2020	OUT017	Tier 2	Small	Supermarket Type1	0.142443	18.850	170.4132	4.0
4797	Regular	FDL02	Canned	2010	OUT046	Tier 1	Small	Supermarket Type1	0.104083	20.000	107.4622	3.5
2868	Low Fat	NCM31	Others	2015	OUT045	Tier 2	Small	Supermarket Type1	0.081361	6.095	141.9154	4.1
3034	Low Fat	FDF05	Frozen Foods	2000	OUT013	Tier 3	High	Supermarket Type1	0.026849	17.500	264.8910	4.1
2848	Low Fat	FDX24	Baking Goods	2015	OUT045	Tier 2	Medium	Supermarket Type1	0.013957	8.355	94.0462	4.1
3785	Low Fat	NCJ42	Household	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.014232	NaN	100.9332	4.0
5955	Regular	FDP22	Snack Foods	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.000000	NaN	52.6666	4.0

In [5]:

```
df.tail()
```

Out[5]:

	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
8518	low fat	NCT53	Health and Hygiene	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.000000	NaN	164.5526	4.0
8519	low fat	FDN09	Snack Foods	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.034706	NaN	241.6828	4.0
8520	low fat	DRE13	Soft Drinks	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.027571	NaN	86.6198	4.0
8521	reg	FDT50	Dairy	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.107715	NaN	97.8752	4.0
8522	reg	FDM58	Snack Foods	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.000000	NaN	112.2544	4.0

In [6]:

```
# get the dimension of the dataframe
df.ndim
```

Out[6]:

```
2
```

In [7]:

```
# get the size of dataframe
df.size
```

Out[7]:

```
102276
```

In [8]:

```
# Get the shape of dataframe
df.shape
```

Out[8]:

```
(8523, 12)
```

In [9]:

```
print("Size of the DataFrame:", df.shape)
```

```
Size of the DataFrame: (8523, 12)
```

In [10]:

```
df.describe()
```

Out[10]:

	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

In [11]:

```
# get all the column name

df.columns
```

Out[11]:

```
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 [12]:

```
# get the type of columns present within the dataframe
df.dtypes
```

Out[12]:

```
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
```

In [18]:

```
print(df['Item Fat Content'].unique())
```

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

In [20]:

```
# Now clean have this column "LF: Low Fat, reg: Regular
```

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

In [ ]:

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

KPI's Requirements

In [23]:

```
# Total sales
total_sales = df['Sales'].sum()
```

In [24]:

```
total_sales
```

Out[24]:

```
1201681.4808
```

In [25]:

```
# Avg Sales

avg_sales = df['Sales'].mean()
```

In [26]:

```
avg_sales
```

Out[26]:

```
140.9927819781767
```

In [28]:

```
# No. of item sold

no_of_item_sold = df['Sales'].count()
```

In [29]:

```
no_of_item_sold
```

Out[29]:

```
8523
```

In [30]:

```
# Avg Rating

avg_ratings = df['Rating'].mean()
```

In [31]:

```
avg_ratings
```

Out[31]:

```
3.965857092573038
```

In [38]:

```
# Show all the KPI's Cummulative

print(f'Total Sales: ${total_sales:,.0f}')
print(f'Average Sales: ${avg_sales:,.0f}')
print(f'No of Items Sales: {no_of_item_sold:,.0f}')
print(f'Average Rating: {avg_ratings:,.0f}')
```

```
Total Sales: $1,201,681
Average Sales: $141
No of Items Sales: 8,523
Average Rating: 4
```

Gather the insight from the dataset

Total Sales by Fat Content

In [40]:

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

In [41]:

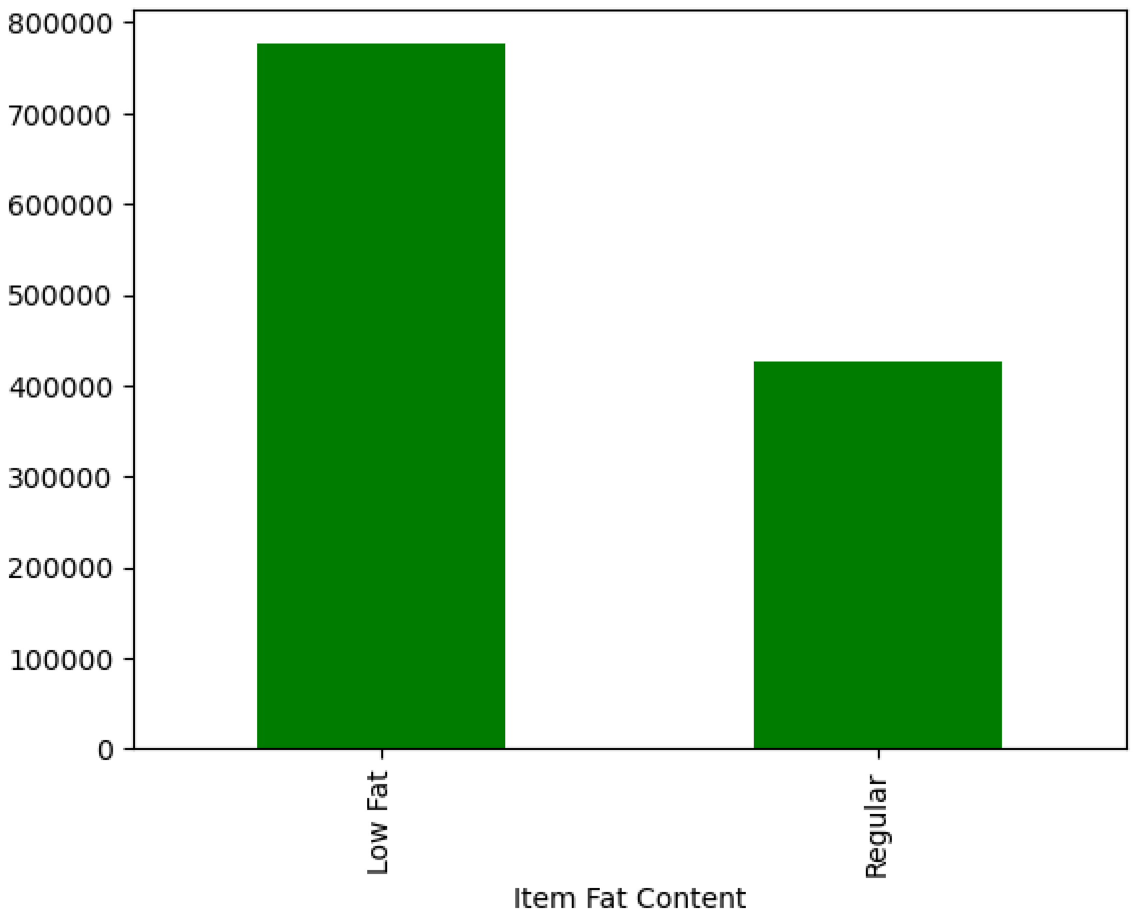
```
sales_by_fat
```

Out[41]:

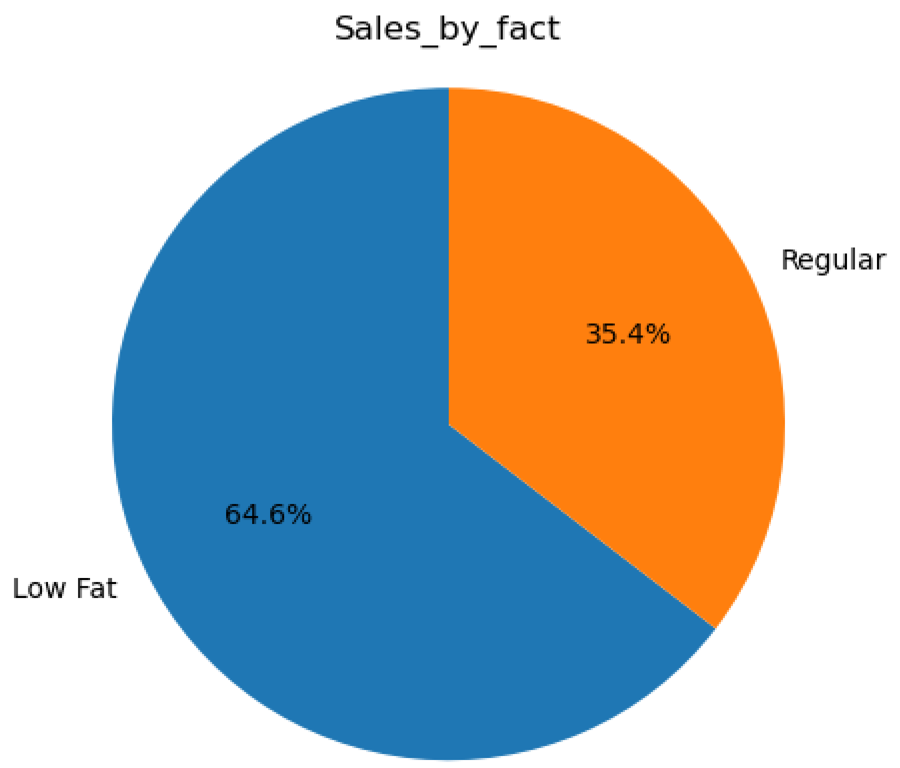
```
Item Fat Content
Low Fat     776319.6784
Regular     425361.8024
Name: Sales, dtype: float64
```

In [60]:

```
sales_by_fat = df.groupby('Item Fat Content')['Sales'].sum().plot(kind='bar', color ='green')
```



In [46]:
```
# Show the salebyfat using pie chart
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_fact')
plt.axis('equal')
plt.show()
```



Total Sales by item type

In [47]:
```
sales_by_type = df.groupby('Item Type')['Sales'].sum()
```

In [48]:
```
sales_by_type
```

Out[48]:
```
Item Type
Baking Goods              81894.7364
Breads                    35379.1198
Breakfast                 15596.6966
Canned                    90706.7270
Dairy                    101276.4596
Frozen Foods             118558.8814
Fruits and Vegetables    178124.0810
Hard Drinks               29334.6766
Health and Hygiene        68025.8388
Household                135976.5254
Meat                      59449.8638
Others                    22451.8916
Seafood                    9077.8700
Snack Foods              175433.9204
Soft Drinks               58514.1650
Starchy Foods             21880.0274
Name: Sales, dtype: float64
```

In [49]:
```
sales_by_type = df.groupby('Item Type')['Sales'].sum().sort_values(ascending = False)
```

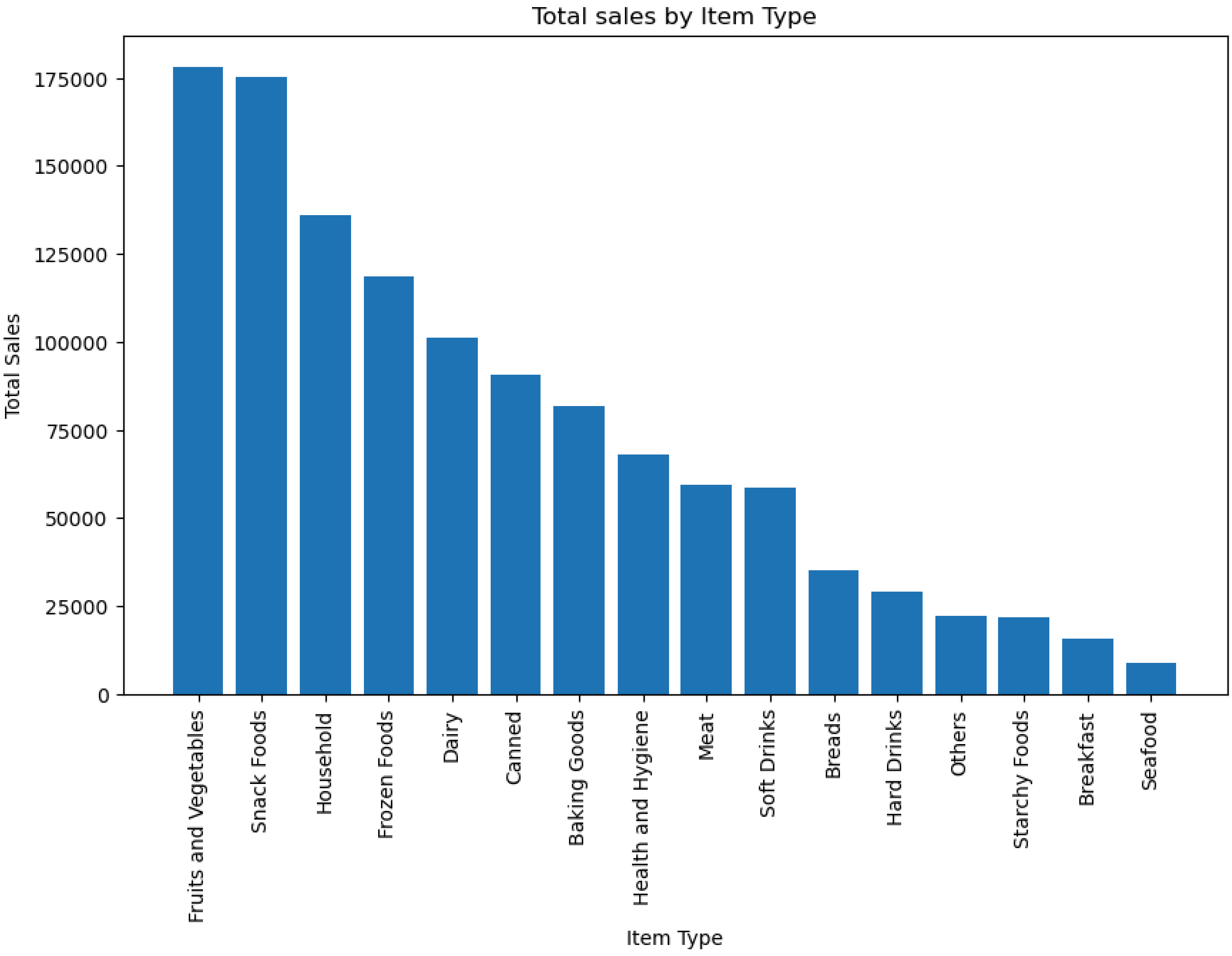
In [50]:
```
sales_by_type
```

Out[50]:
```
Item Type
Fruits and Vegetables    178124.0810
Snack Foods              175433.9204
Household                135976.5254
Frozen Foods             118558.8814
Dairy                    101276.4596
Canned                    90706.7270
Baking Goods              81894.7364
Health and Hygiene        68025.8388
Meat                      59449.8638
Soft Drinks               58514.1650
Breads                    35379.1198
Hard Drinks               29334.6766
Others                    22451.8916
Starchy Foods             21880.0274
Breakfast                 15596.6966
Seafood                    9077.8700
Name: Sales, dtype: float64
```

In [51]:
```
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')
```

Out[51]:
```
Text(0.5, 1.0, 'Total sales by Item Type')
```



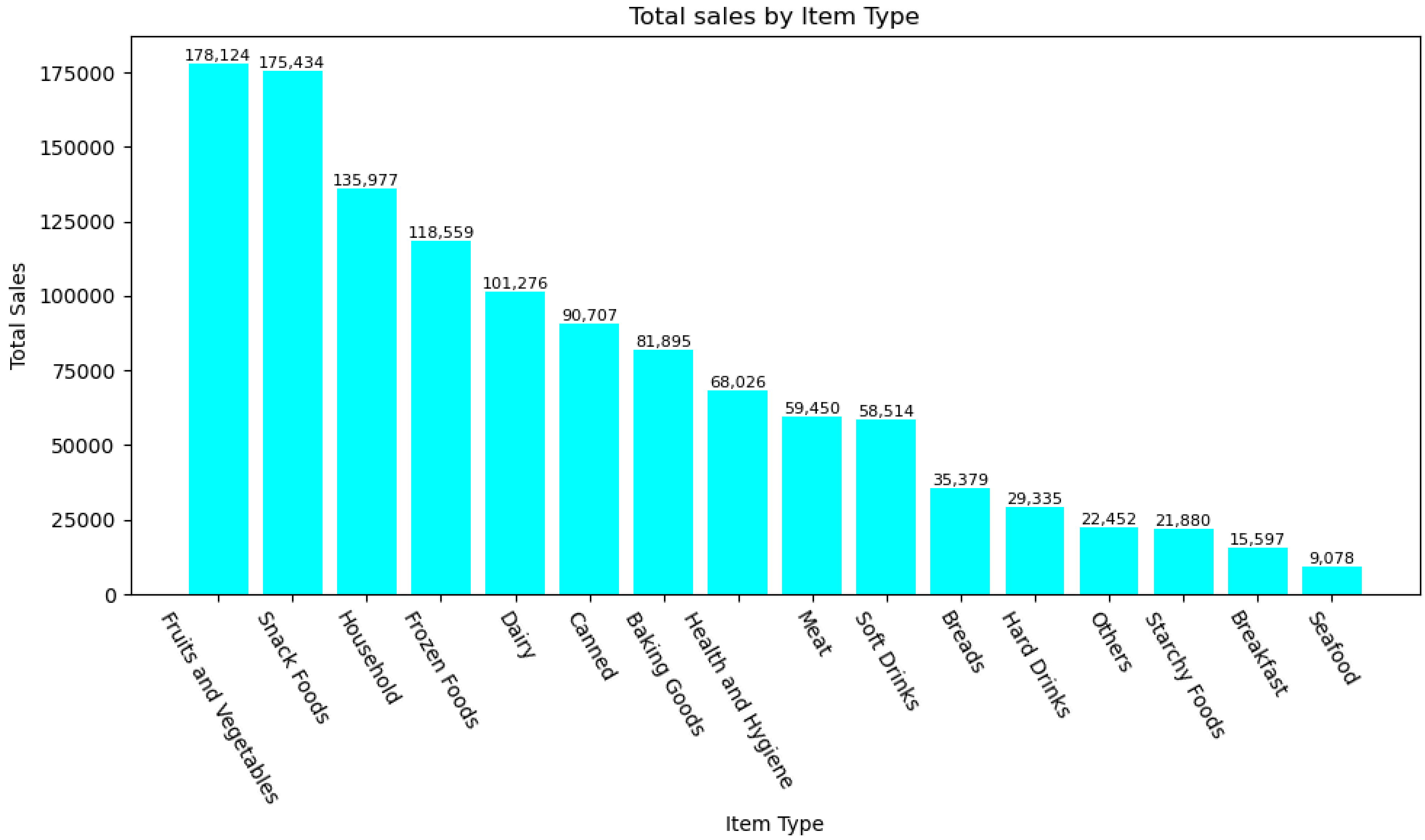
In [64]:
```
sales_by_type = df.groupby('Item Type')['Sales'].sum().sort_values(ascending = False)
plt.figure(figsize =(10,6))

bars = plt.bar(sales_by_type.index, sales_by_type.values, color = 'cyan')

plt.xticks(rotation = - 60)
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, color ='black' )

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



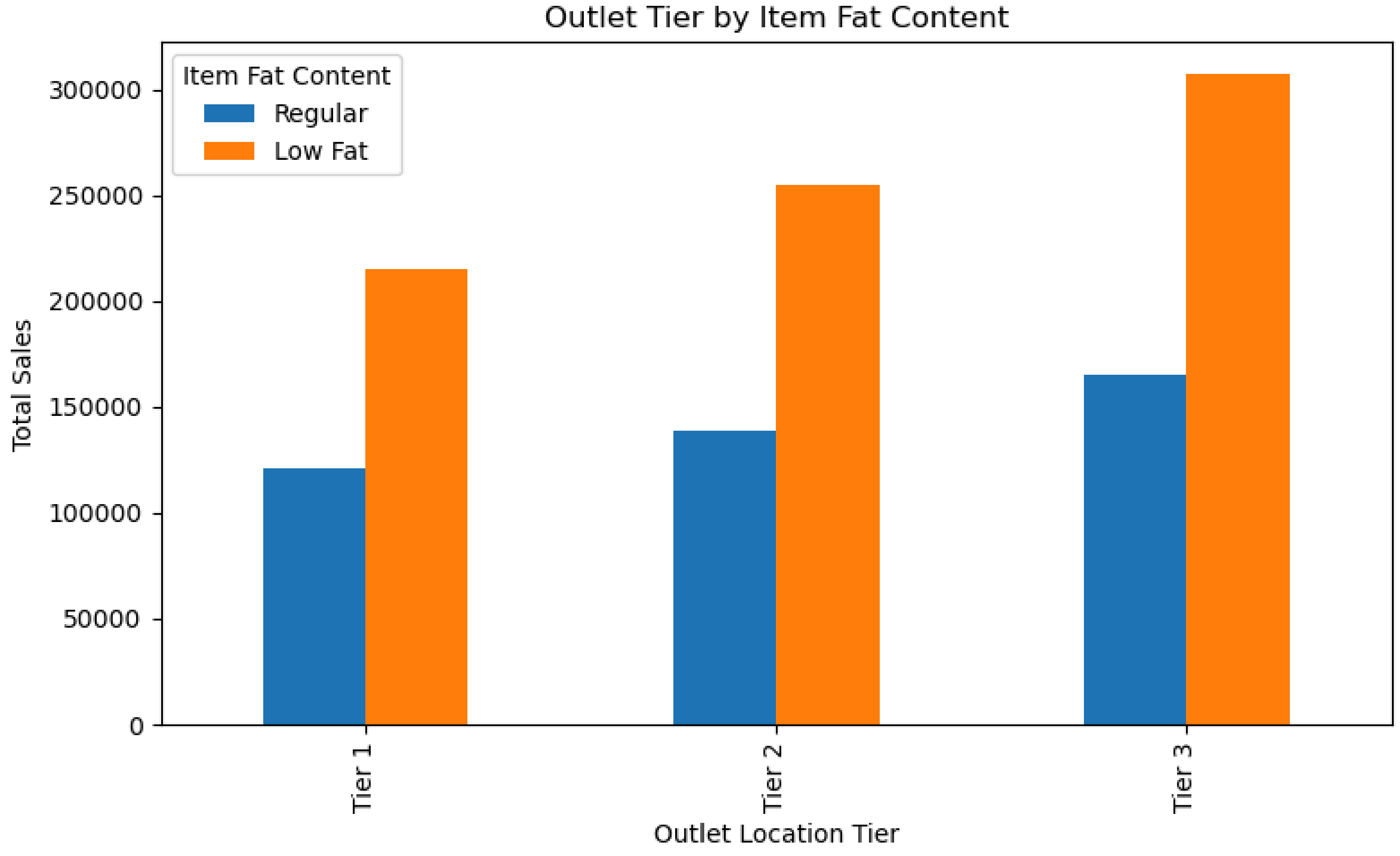
Fat Content by Outlet for Total Sales

In [66]:

```
grouped = df.groupby(['Outlet Location Type', 'Item Fat Content'])['Sales'].sum().unstack()

grouped = grouped [['Regular' , 'Low Fat']]

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



In [65]:

```
df.dtypes
```

Out[65]:

```
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
```

Total Sales By Outlet Establishment

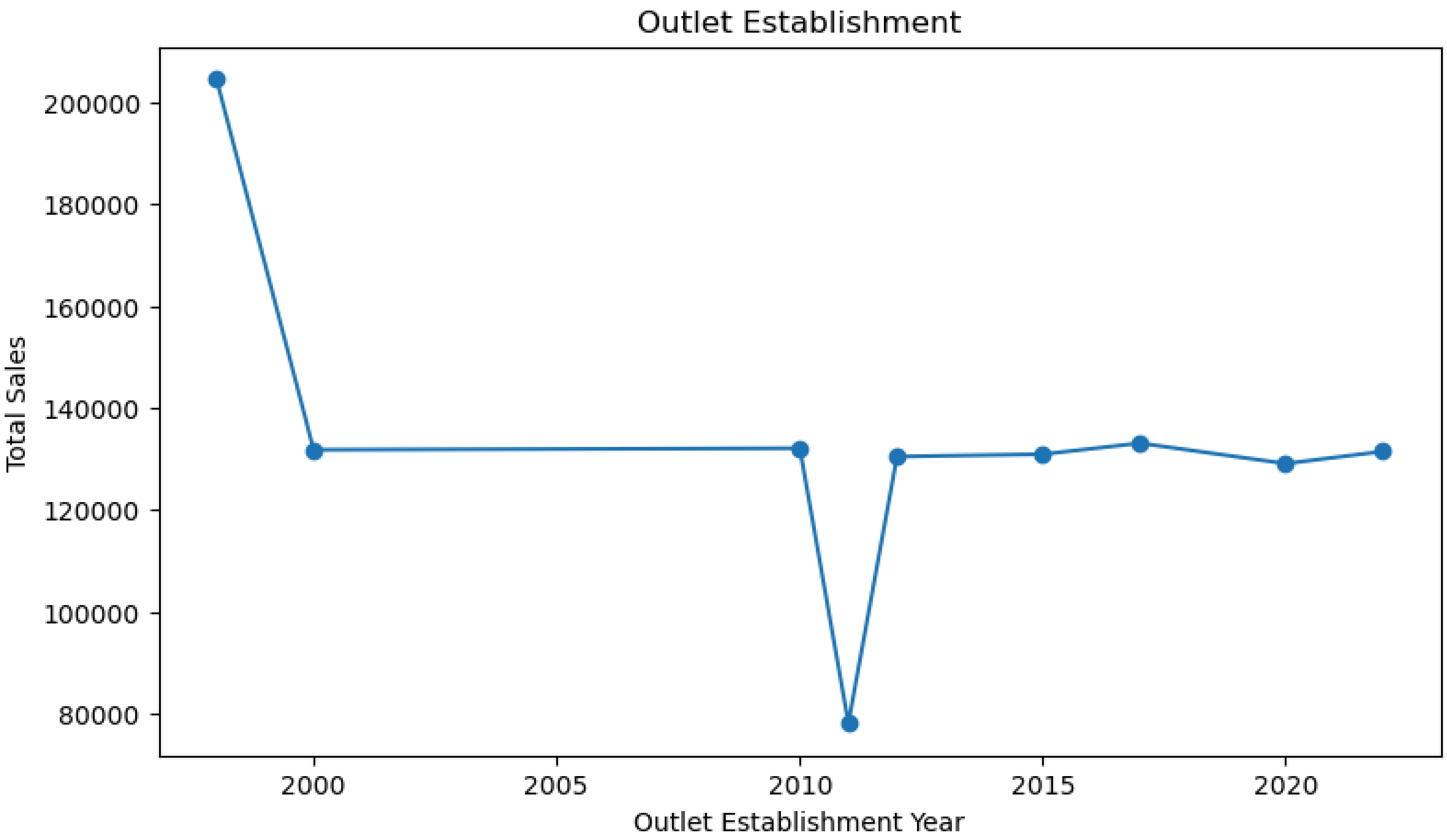
In [70]:

```
sales_by_year = df.groupby('Outlet Establishment Year')['Sales'].sum().sort_index()

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')

plt.show()
```



In [75]:

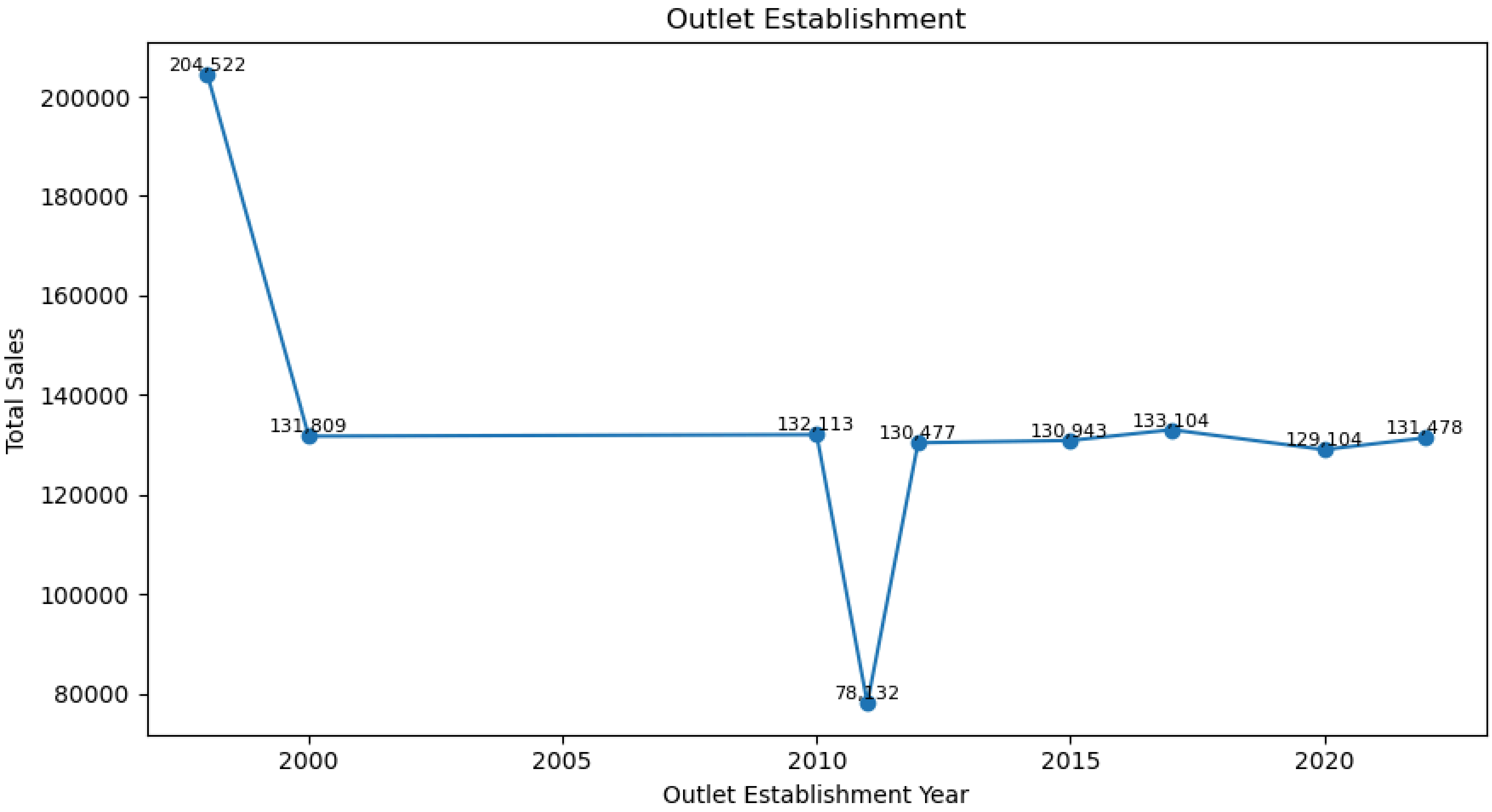
```
sales_by_year = df.groupby('Outlet Establishment Year')['Sales'].sum().sort_index()

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()
```



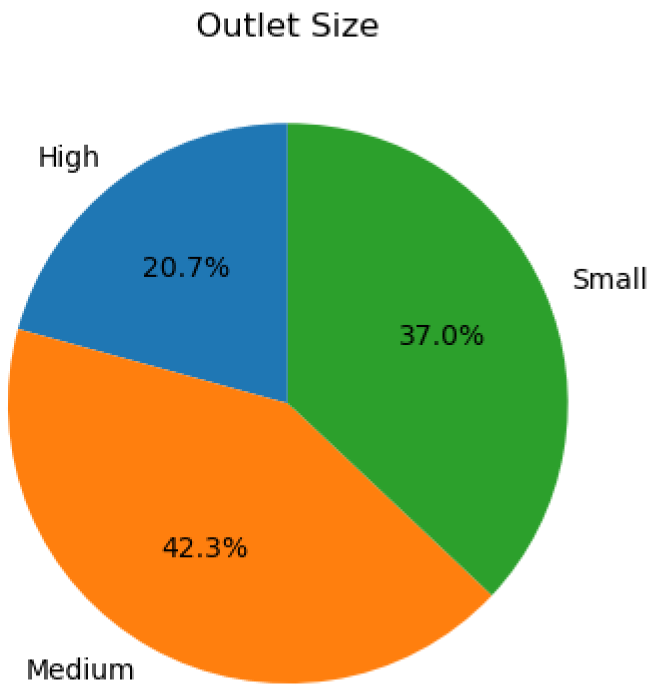
Sales by Outlet Size

In [77]:

```
sales_by_size = df.groupby("Outlet Size")["Sales"].sum()

plt.figure(figsize = (6,4))
plt.pie(sales_by_size, labels = sales_by_size.index,
        autopct = '%1.1f%%', startangle = 90)

plt.title("Outlet Size")
plt.tight_layout()
plt.show()
```



Sales by Outlet Locations

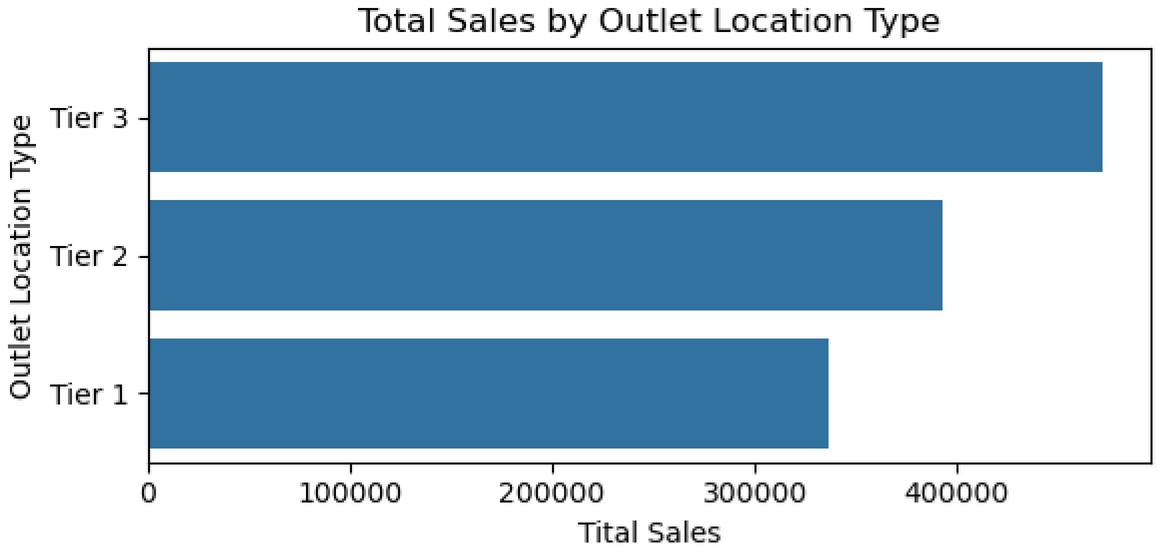
In [83]:

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

plt.figure(figsize=(6,3))
ax = sns.barplot(x='Sales', y='Outlet Location Type', data = sales_by_location)

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

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


Total Sales by Outlet Location Type
Outlet Location Type
Tier 3
Tier 2
Tier 1
0
100000
200000
300000
400000
Tital Sales

In [ ]: