# DATA ANALYSIS PYTHON PROJECT - BLINKIT ANALYSIS

## Import Libraries

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

## Import Raw Data

In [2]:
```
df = pd.read_csv("Blinkit Analysis/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 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 |
| **5** | low fat | FDS52 | Frozen Foods | 2020 | OUT017 | Tier 2 | Small | Supermarket Type1 | 0.005505 | 8.89 | 102.4016 | 5.0 |
| **6** | Low Fat | NCU05 | Health and Hygiene | 2011 | OUT010 | Tier 3 | Small | Grocery Store | 0.098312 | 11.80 | 81.4618 | 5.0 |
| **7** | Low Fat | NCD30 | Household | 2015 | OUT045 | Tier 2 | Small | Supermarket Type1 | 0.026904 | 19.70 | 96.0726 | 5.0 |
| **8** | Low Fat | FDW20 | Fruits and Vegetables | 2000 | OUT013 | Tier 3 | High | Supermarket Type1 | 0.024129 | 20.75 | 124.1730 | 5.0 |
| **9** | Low Fat | FDX25 | Canned | 1998 | OUT027 | Tier 3 | Medium | Supermarket Type3 | 0.101562 | NaN | 181.9292 | 5.0 |

In [4]:
```
df.tail(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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **8513** | Regular | DRY23 | Soft Drinks | 1998 | OUT027 | Tier 3 | Medium | Supermarket Type3 | 0.108568 | NaN | 42.9112 | 4.0 |
| **8514** | low fat | FDA11 | Baking Goods | 1998 | OUT027 | Tier 3 | Medium | Supermarket Type3 | 0.043029 | NaN | 94.7436 | 4.0 |
| **8515** | low fat | FDK38 | Canned | 1998 | OUT027 | Tier 3 | Medium | Supermarket Type3 | 0.053032 | NaN | 149.1734 | 4.0 |
| **8516** | low fat | FDO38 | Canned | 1998 | OUT027 | Tier 3 | Medium | Supermarket Type3 | 0.072486 | NaN | 78.9986 | 4.0 |
| **8517** | low fat | FDG32 | Fruits and Vegetables | 1998 | OUT027 | Tier 3 | Medium | Supermarket Type3 | 0.175143 | NaN | 222.3772 | 4.0 |
| **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 |

## Size of Data

In [5]:
```
print("Size of data is:", df.shape)
```

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

## Field (col name) of Data

In [6]:
```
df.columns
```

Out[6]:
```
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')
```

## Dataypes

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 - Remove inconsistancy

In [8]:
```
df['Item Fat Content'] = df['Item Fat Content'].replace({'low fat': 'Low Fat',
                                                         'LF':'Low Fat',
                                                         'reg': 'Regular'})
```

In [9]:
```
print(df['Item Fat Content'].unique())
```

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

In [10]:
```
df['Item Fat Content']
```

Out[10]:
```
0       Regular
1       Low Fat
2       Regular
3       Regular
4       Low Fat
         ...
8518    Low Fat
8519    Low Fat
8520    Low Fat
8521    Regular
8522    Regular
Name: Item Fat Content, Length: 8523, dtype: object
```

In [11]:
```
df['Item Fat Content'].dtype
```

Out[11]:
```
dtype('O')
```

In [12]:
```
df['Item Fat Content'].str.contains(' ')
```

Out[12]:
```
0       False
1        True
2       False
3       False
4        True
        ...
8518     True
8519     True
8520     True
8521    False
8522    False
Name: Item Fat Content, Length: 8523, dtype: bool
```

In [13]:
```
((df.map(lambda x: str(x).strip()) != df.map(str)).any().any())
```

Out[13]:
```
np.False_
```

In [14]:
```
(df.astype(str) != df.astype(str).apply(lambda x: x.str.strip())).any()
```

Out[14]:
```
Item Fat Content             False
Item Identifier              False
Item Type                    False
Outlet Establishment Year    False
Outlet Identifier            False
Outlet Location Type         False
Outlet Size                  False
Outlet Type                  False
Item Visibility              False
Item Weight                  False
Sales                        False
Rating                       False
dtype: bool
```

## BUSINESS REQUIREMENTS

## KIP's REQUIREMENTS

In [15]:
```
#total sales
# col name = Expression - action
Total_Sales = df['Sales'].sum()

#avg sales
Avg_sales = df['Sales'].mean()

#avg rating
Avg_ratings = df['Rating'].mean()

#no of item sold
Total_no_item_sold = df['Sales'].count()
```

In [16]:
```
#Display

print(f"Total Sales : ${Total_Sales:,.0f}")
print(f"Average Sales: ${Avg_sales:.0f}")
print(f"No of Item Sold: {Total_no_item_sold:,.0f}")
print(f"Average Rating: {Avg_ratings:.1f}")
```

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

# CHART's REQUIREMENTS

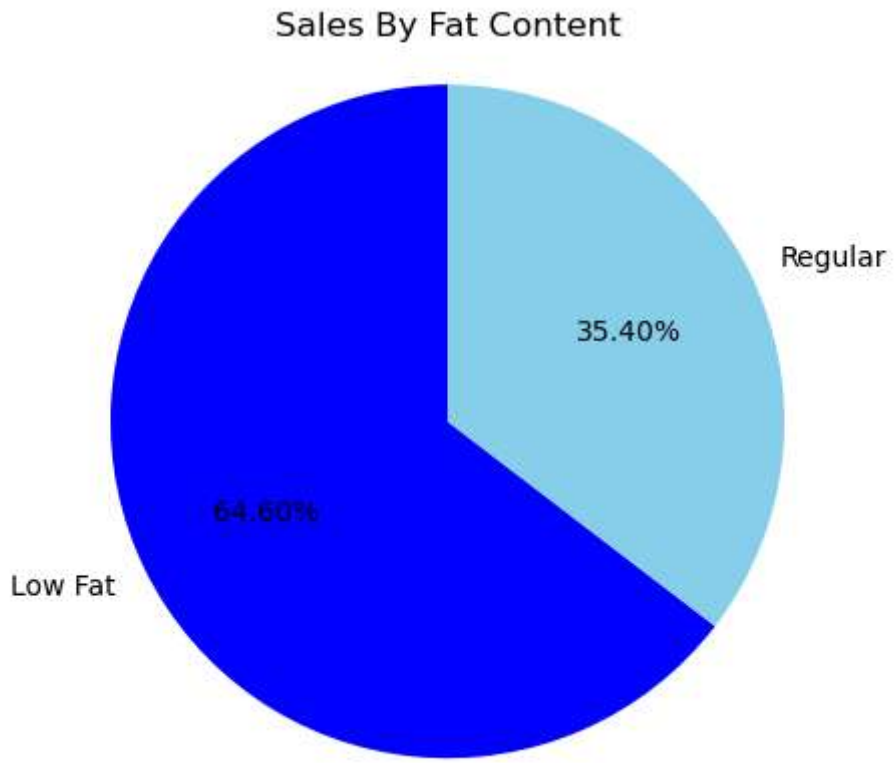
## Total sales by Fat Content

In [17]:
```
sales_by_fat = df.groupby('Item Fat Content')['Sales'].sum() #low fat, regular

colors = ['blue', 'skyblue'] #colors

plt.pie(sales_by_fat, labels = sales_by_fat.index,  #index- we are assigning the index that is name of that slice in pie chart
                                autopct = '%.2f%%',
                                startangle = 90,
                                colors = colors)

plt.title('Sales By Fat Content')
plt.axis('equal')
plt.show()
```




## Total sales by Item Type

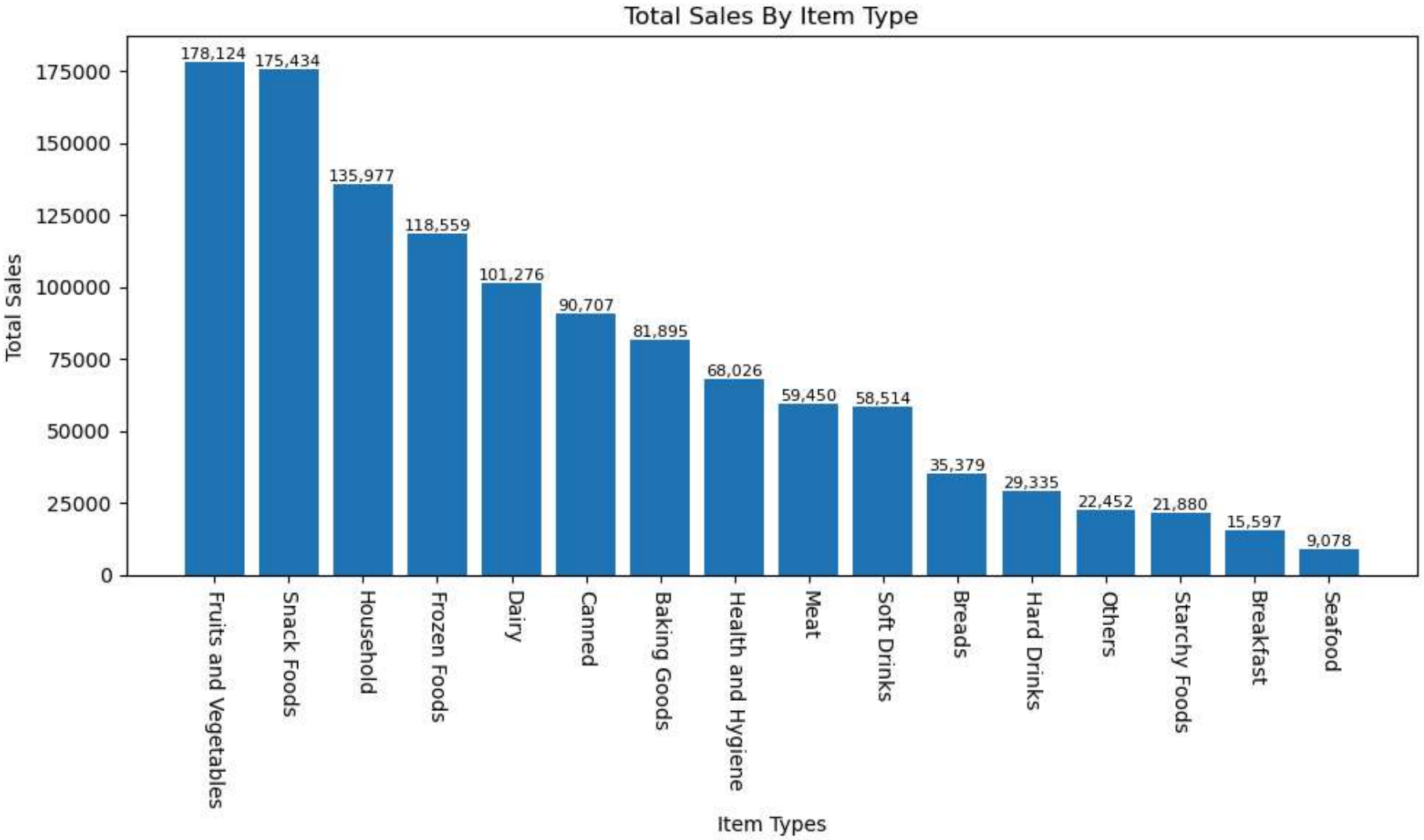
In [18]:
```
sales_by_type = df.groupby('Item Type')['Sales'].sum().sort_values(ascending = False)

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

plt.xticks(rotation = -90) #0- text overlapping, 45- it is not look that much good interms of looks and readbility
plt.xlabel('Item Types')
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(),  #optional - it used to give the numbers on the top of each b
                              #1st- it'll get x axis coodrinates and then width and add both divide by 2 to get center position
```

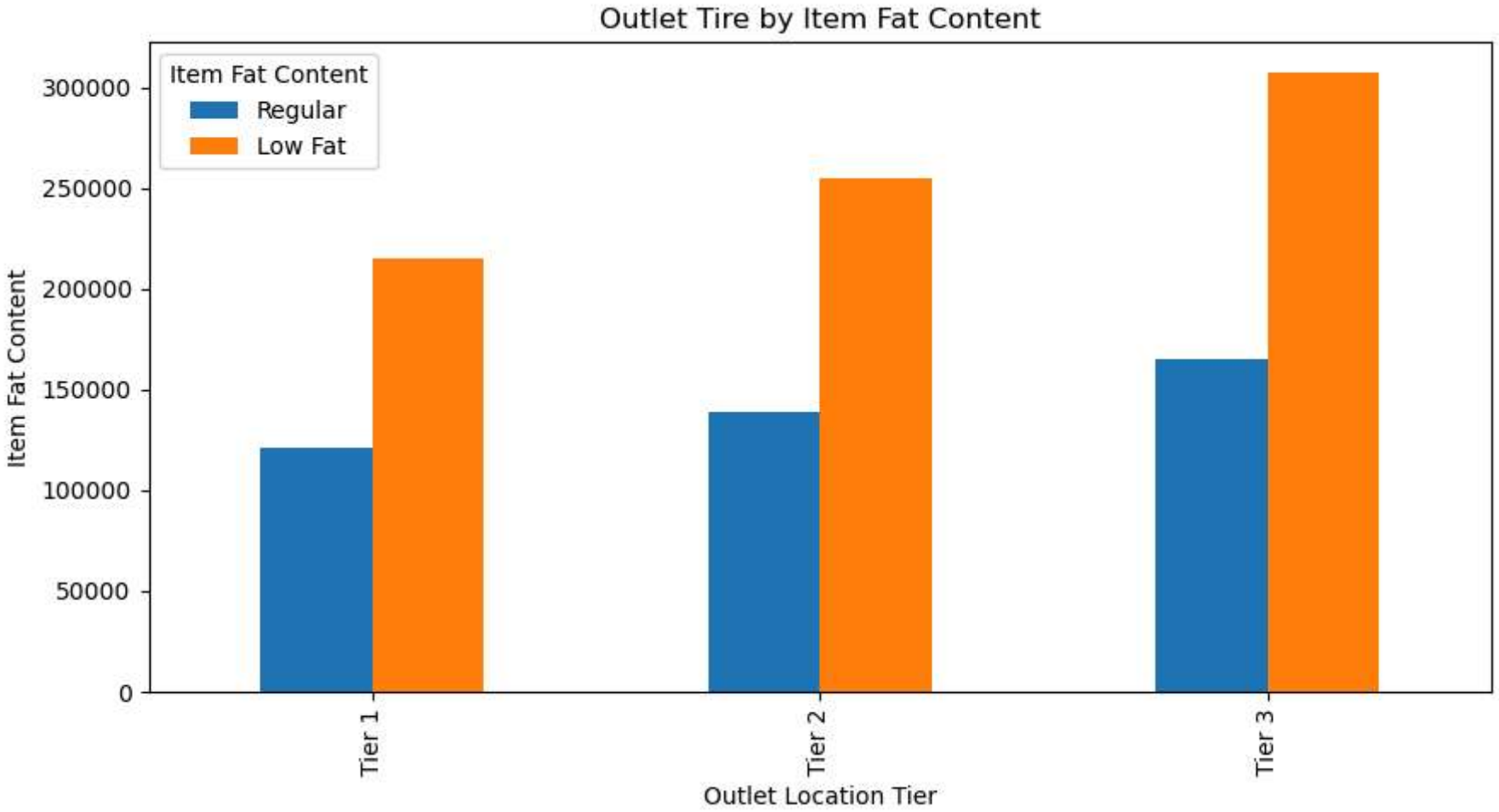
```
            f'{bar.get_height():,.0f}', ha = 'center', va = 'bottom', fontsize = 8)
plt.tight_layout()
plt.show()
```




## Fat Content b Outlet for Total Sales

In [19]:
```
Outlet_Content = df.groupby(['Outlet Location Type', 'Item Fat Content'])['Sales'].sum().unstack() #convert row into col regul
Outlet_Content = Outlet_Content[['Regular', 'Low Fat']]  # to ensure bar dont overlap on each others

ax = Outlet_Content.plot(kind = 'bar', figsize = (9, 5), title ='Outlet Tire by Item Fat Content')
plt.xlabel('Outlet Location Tier')
plt.ylabel('Item Fat Content')
plt.tight_layout()
plt.show()
```




## Total Sales by Outlet Establishment

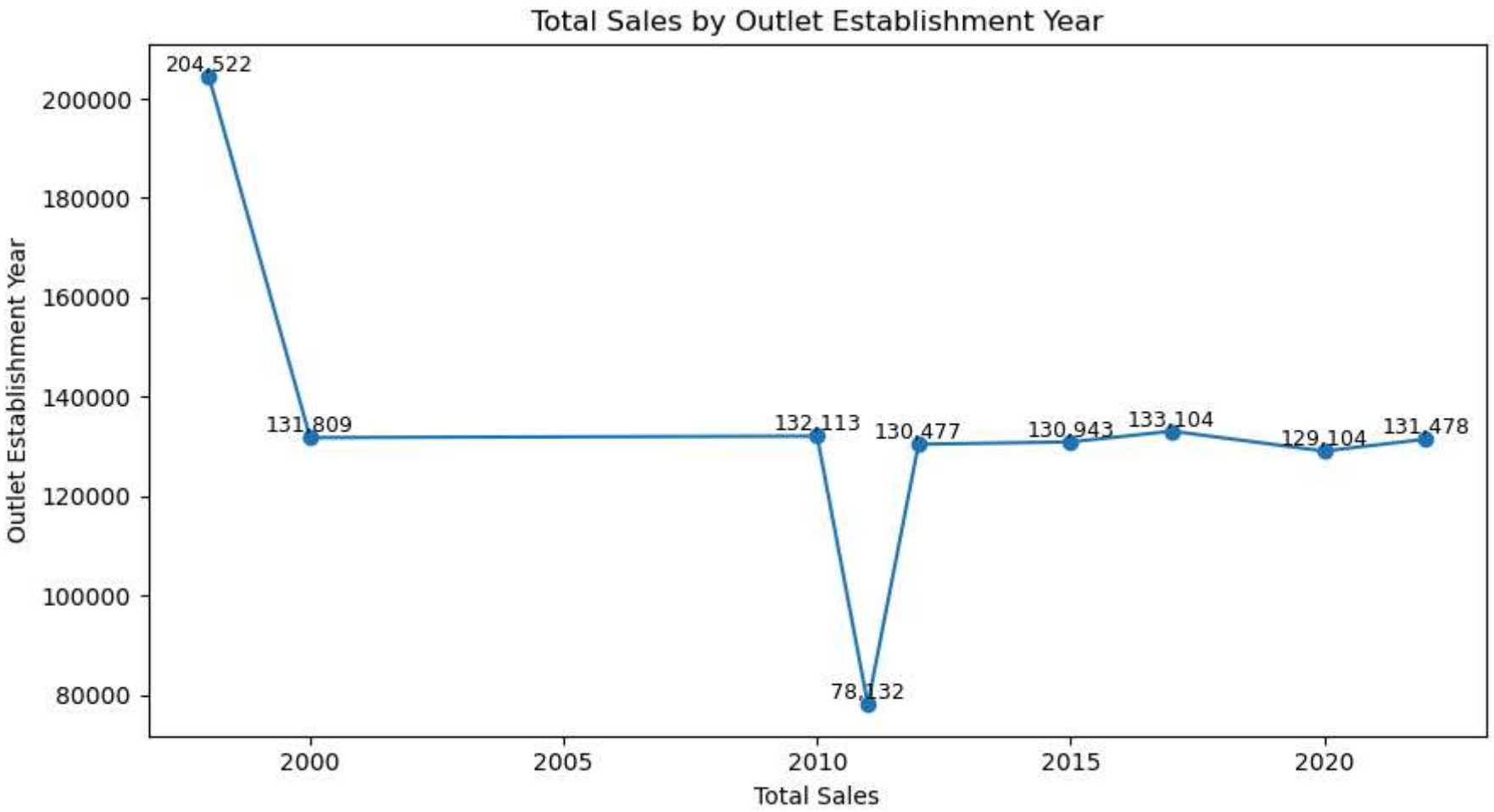
In [20]:
```
Total_sales_by_establishhment = df.groupby(['Outlet Establishment Year'])['Sales'].sum().sort_index() #to take indexes in sort

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

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

for x, y in zip(Total_sales_by_establishhment.index, Total_sales_by_establishhment.values):
    plt.text(x,y, f'{y:,.0f}', ha='center', va= 'bottom', fontsize = 9)

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



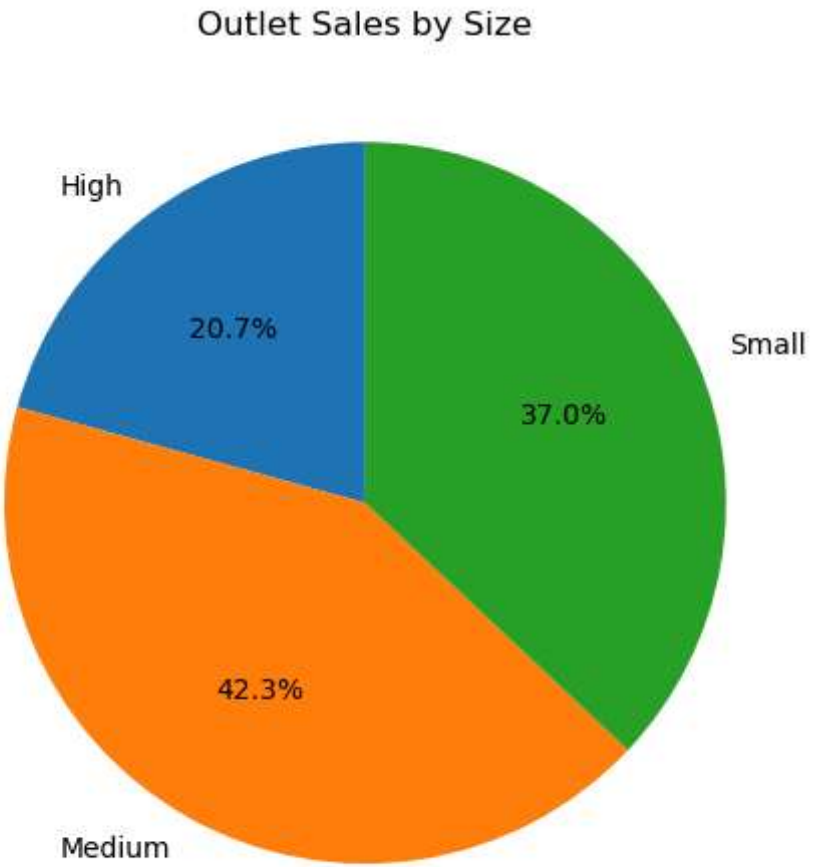

## Select By Outlet Size

In [21]:
```
sales_by_size = df.groupby('Outlet Size')['Sales'].sum()

plt.figure(figsize=(5,5)) # optional
plt.pie(sales_by_size, labels = sales_by_size.index, autopct = '%1.1f%%', startangle = 90)

plt.title('Outlet Sales by Size')
```

```
plt.tight_layout() # optional - but if we are applying any size to figure then it is good practice to tight_layout just to avo
plt.show()
```

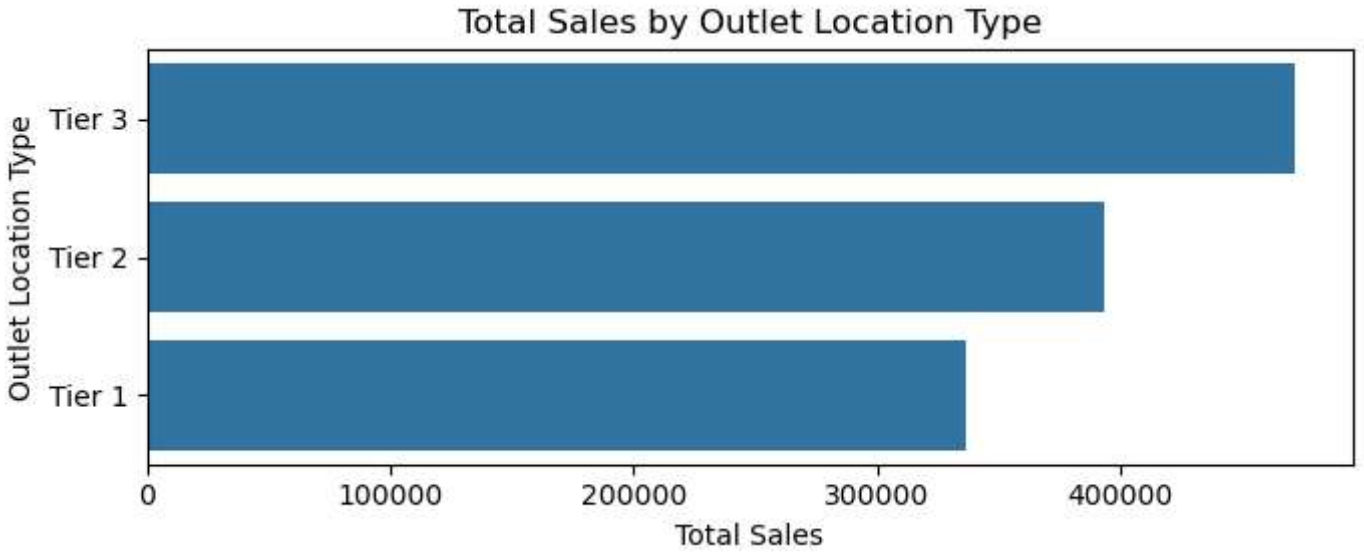



## Select By Outlet Size

In [25]:
```
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=(7, 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 [ ]: