DATA ANALYSIS PYTHON PROJECT-EVERYDAY ANALYSIS

IMPORT LIBRARIES

In [18]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

IMPORT RAW DATA

In [19]:
```
df = pd.read_excel("C:/Users/ACER/Downloads/everyday Grocery Data.xlsx")
```

SAMPLE DATA

In [22]:
```
df.head(20)
```

Out[22]:

	Item Fat Content	Item Identifier	Item Type	Outlet Establishment Year	Outlet Identifier	Outlet Location Type	Outlet Size	Outlet Typ
0	Regular	FDX32	Fruits and Vegetables	2012	OUT049	Tier 1	Medium	Supermark Type
1	Low Fat	NCB42	Health and Hygiene	2022	OUT018	Tier 3	Medium	Supermark Type
2	Regular	FDR28	Frozen Foods	2016	OUT046	Tier 1	Small	Supermark Type
3	Regular	FDL50	Canned	2014	OUT013	Tier 3	High	Supermark Type
4	Low Fat	DRI25	Soft Drinks	2015	OUT045	Tier 2	Small	Supermark Type
5	low fat	FDS52	Frozen Foods	2020	OUT017	Tier 2	Small	Supermark Type
6	Low Fat	NCU05	Health and Hygiene	2011	OUT010	Tier 3	Small	Groce Sto
7	Low Fat	NCD30	Household	2015	OUT045	Tier 2	Small	Supermark Type
8	Low Fat	FDW20	Fruits and Vegetables	2014	OUT013	Tier 3	High	Supermark Type
9	Low Fat	FDX25	Canned	2018	OUT027	Tier 3	Medium	Supermark Type
10	LF	FDX21	Snack Foods	2018	OUT027	Tier 3	Medium	Supermark Type
11	Low Fat	NCU41	Health and Hygiene	2017	OUT035	Tier 2	Small	Supermark Type
12	Low Fat	FDL20	Fruits and Vegetables	2022	OUT018	Tier 3	Medium	Supermark Type
13	Low Fat	NCR54	Household	2014	OUT013	Tier 3	High	Supermark Type
14	Low Fat	FDH19	Meat	2018	OUT027	Tier 3	Medium	Supermark Type
15	Regular	FDB57	Fruits and Vegetables	2017	OUT035	Tier 2	Small	Supermark Type
16	Low Fat	FDO23	Breads	2022	OUT018	Tier 3	Medium	Supermark Type

	Item Fat Content	Item Identifier	Item Type	Outlet Establishment Year	Outlet Identifier	Outlet Location Type	Outlet Size	Outlet Typ
17	Low Fat	NCB07	Household	2012	OUT049	Tier 1	Medium	Supermark Type
18	Low Fat	FDJ56	Fruits and Vegetables	2018	OUT027	Tier 3	Medium	Supermark Type
19	Low Fat	DRN47	Hard Drinks	2022	OUT018	Tier 3	Medium	Supermark Type

In [23]:
```
df.tail(10)
```

Out[23]:

	Item Fat Content	Item Identifier	Item Type	Outlet Establishment Year	Outlet Identifier	Outlet Location Type	Outlet Size	Outlet T
8513	Regular	DRY23	Soft Drinks	2018	OUT027	Tier 3	Medium	Superma T
8514	low fat	FDA11	Baking Goods	2018	OUT027	Tier 3	Medium	Superma T
8515	low fat	FDK38	Canned	2018	OUT027	Tier 3	Medium	Superma T
8516	low fat	FDO38	Canned	2018	OUT027	Tier 3	Medium	Superma T
8517	low fat	FDG32	Fruits and Vegetables	2018	OUT027	Tier 3	Medium	Superma T
8518	low fat	NCT53	Health and Hygiene	2018	OUT027	Tier 3	Medium	Superma T
8519	low fat	FDN09	Snack Foods	2018	OUT027	Tier 3	Medium	Superma T
8520	low fat	DRE13	Soft Drinks	2018	OUT027	Tier 3	Medium	Superma T
8521	reg	FDT50	Dairy	2018	OUT027	Tier 3	Medium	Superma T
8522	reg	FDM58	Snack Foods	2018	OUT027	Tier 3	Medium	Superma T

SIZE OF DATA

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

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

FIELD INFORMATION

In [24]:
```
df.columns
```

Out[24]:
```
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')
```

DATA TYPES

In [25]:
```
df.dtypes
```

Out[25]:
```
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 [26]:
```
print(df["Item Fat Content"].unique())
```

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

In [27]:
```
df['Item Fat Content']= df['Item Fat Content'].replace({'LF':'Low Fat','low fat':'L
```

In [28]:
```
print(df["Item Fat Content"].unique())
```

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

BUSINESS REQUIREMENTS

KPI'S REQUIREMENTS

In [29]:
```
#TOTAL SALES
total_sales=df['Sales'].sum()
#AVERAGE SALES
avg_sales=df['Sales'].mean()
#number of items sold
number_of_items_sold=df['Sales'].count()
#average ratings
```

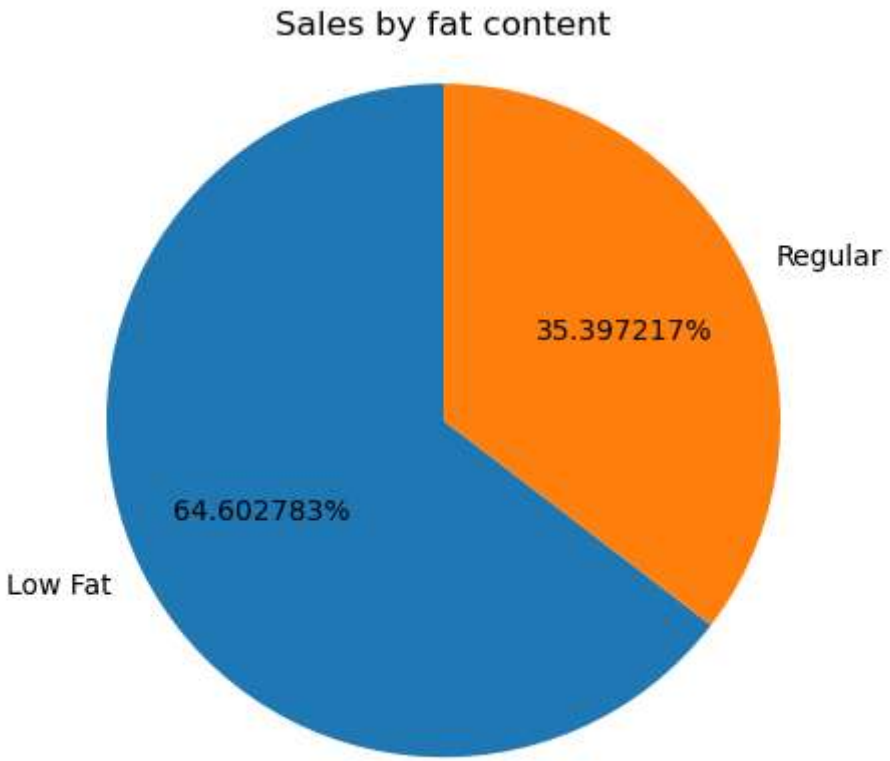
```
avg_ratings=df['Rating'].mean()
#display
print(f"Total sales:${total_sales:,.0f}")
print(f"Average  sales:${avg_sales:,.0f}")
print(f"Number of items sold:{number_of_items_sold:,.0f}")
print(f"Average rating:{avg_ratings:,.0f}")
```

```
Total sales:$1,201,681
Average  sales:$141
Number of items sold:8,523
Average rating:4
```

CHARTS REQUIREMENTS

TOTAL SALES BY FAT CONTENT

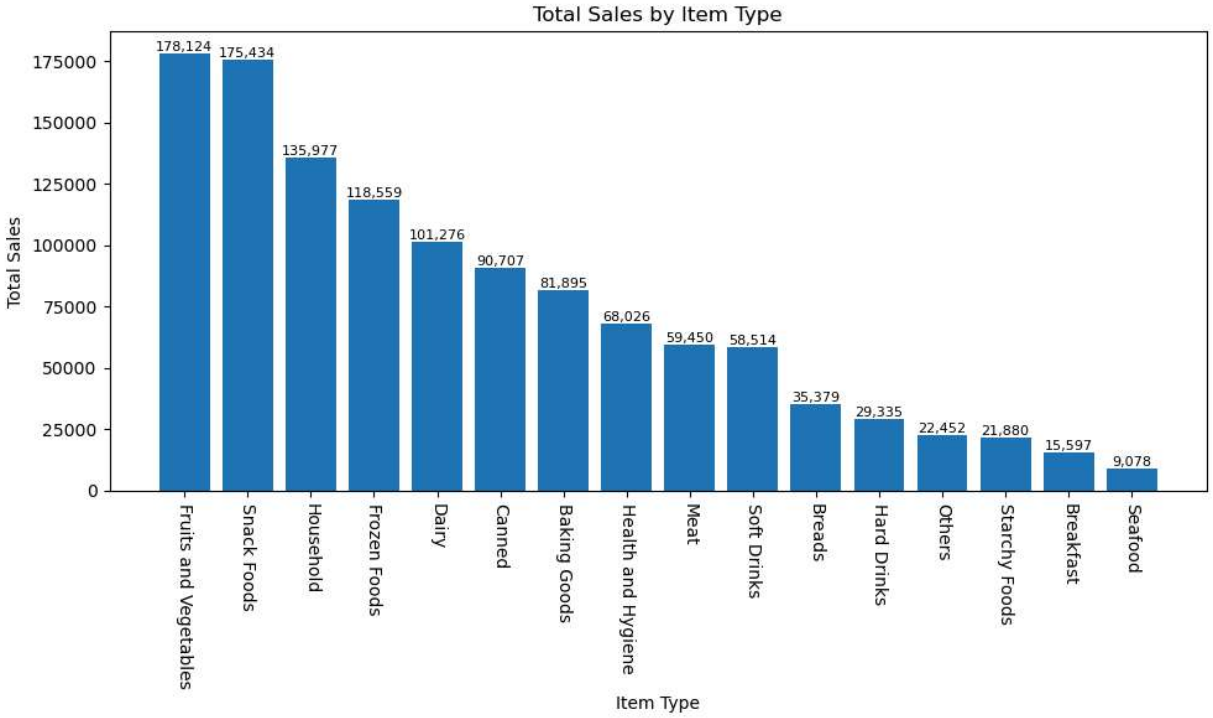
In [12]:
```
sales_by_fat=df.groupby('Item Fat Content')['Sales'].sum()
plt.pie(sales_by_fat,labels =sales_by_fat.index,
        autopct = '%1f%%',
        startangle=90)
plt.title('Sales by fat content')
plt.axis('equal')
plt.show()
```



TOTAL SALES BY ITEM TYPE

In [30]:
```
# Calculate total sales for each item type, sorted descending
sales_by_type = df.groupby('Item Type')['Sales'].sum().sort_values(ascending=False)

# Plot the bar chart
```

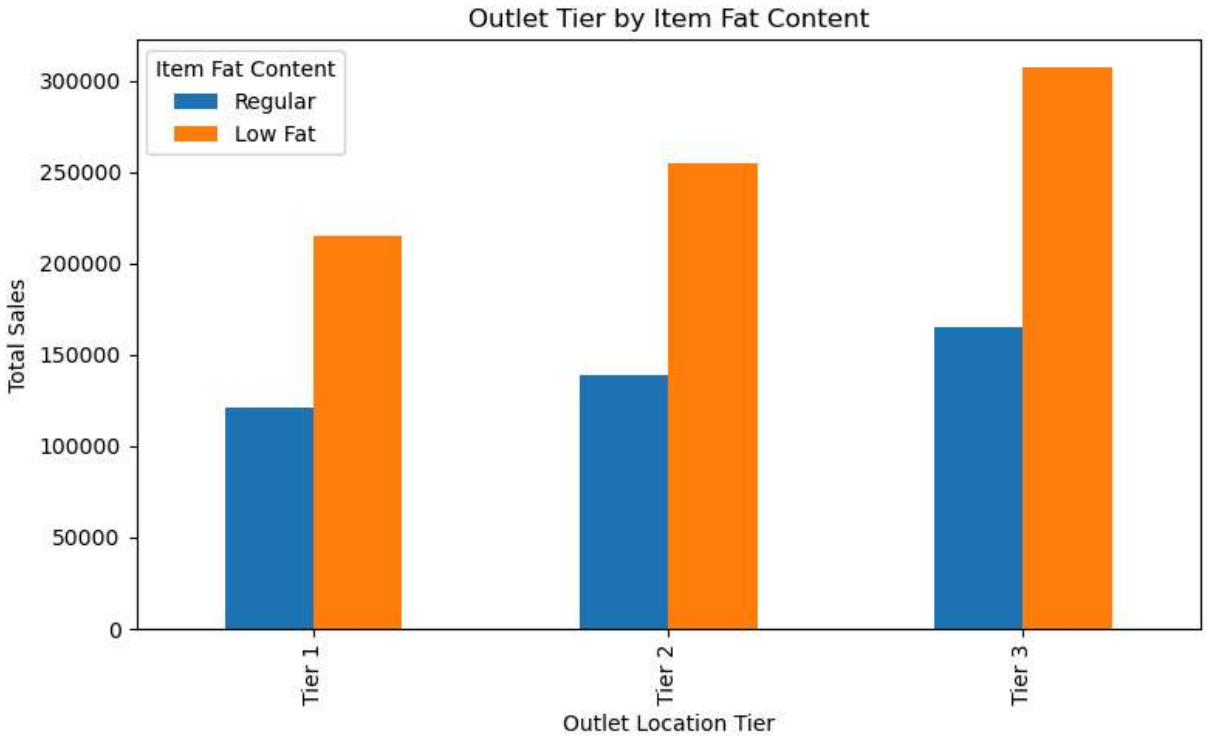
```
plt.figure(figsize=(10, 6))
bars = plt.bar(sales_by_type.index, sales_by_type.values)

# Rotate x-axis labels for better visibility
plt.xticks(rotation=-90)
plt.xlabel('Item Type')
plt.ylabel('Total Sales')
plt.title('Total Sales by Item Type')

# Annotate each bar with its sales value
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

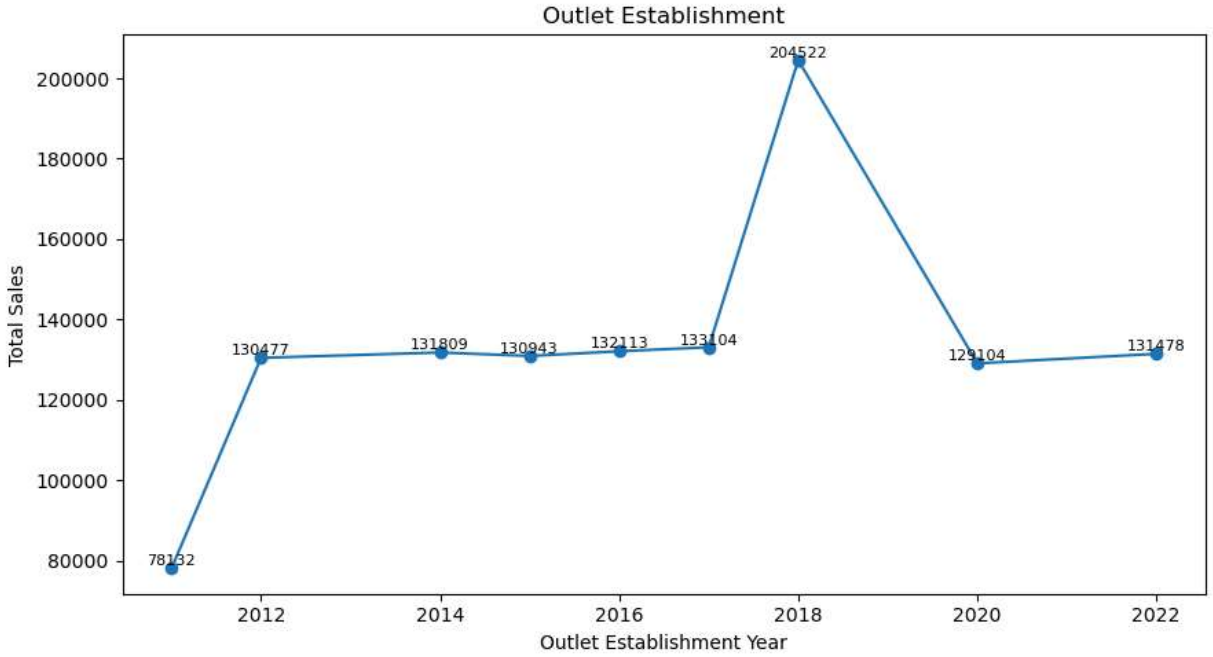
In [31]:
```
# Group and aggregate sales by Outlet Location Type and Item Fat Content
grouped = df.groupby(['Outlet Location Type', 'Item Fat Content'])['Sales'].sum().u

# Select only 'Regular' and 'Low Fat' for the fat content categories
grouped = grouped[['Regular', 'Low Fat']]

# Plot the grouped data as a bar chart
ax = grouped.plot(kind='bar', figsize=(8, 5), title='Outlet Tier by Item Fat Conten
plt.xlabel('Outlet Location Tier')
```

```
plt.ylabel('Total Sales')
plt.legend(title='Item Fat Content')
plt.tight_layout()
plt.show()
```



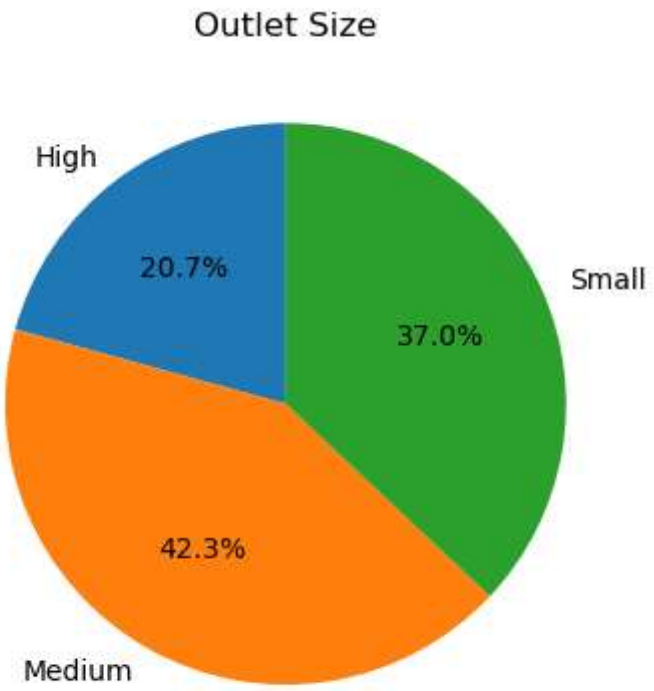
TOTAL SALES BY OUTLET ESTABLISHMENT

In [15]:
```
# Aggregate total sales by year of outlet establishment, sorted by year
sales_by_year = df.groupby('Outlet Establishment Year')['Sales'].sum().sort_index()

# Create the line plot
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')

# Annotate each point with its sales value
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()
```



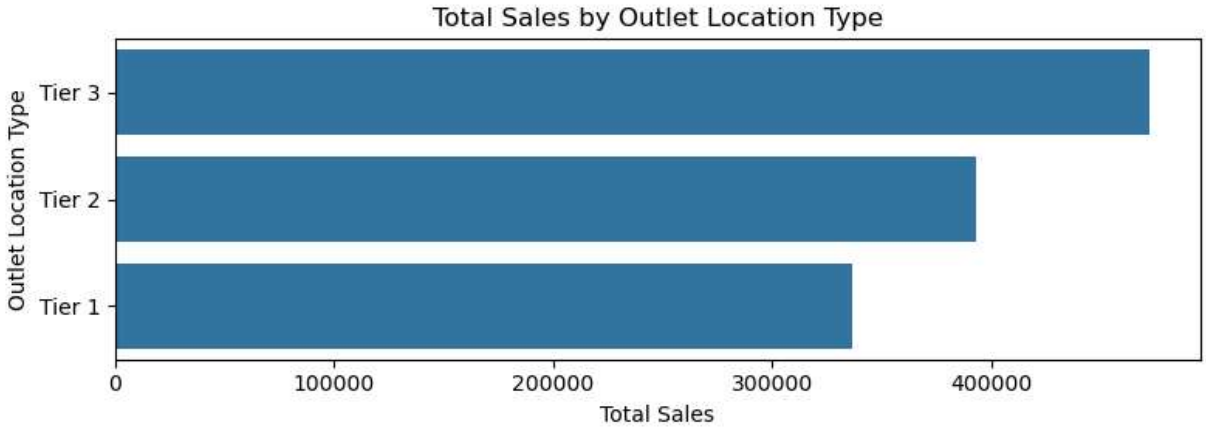
SALES BY OUTLET SIZE

In [16]:
```
# Aggregate total sales by outlet size
sales_by_size = df.groupby('Outlet Size')['Sales'].sum()

# Create the pie chart with custom formatting
plt.figure(figsize=(4, 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 LOCATION

In [17]:
```
# Aggregate total sales by outlet location type and reset index for plotting
sales_by_location = df.groupby('Outlet Location Type')['Sales'].sum().reset_index()

# Sort locations by sales in descending order
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 [ ]: