****DATA ANALYSIS PYTHON PROJECT- BLINKIT ANALYSIS****

IMPORT LIBRARYS

In [27]:

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

IMPORT SAMPLE DATA

In [28]:

```
df=pd.read_csv("C:/Users/BHAGYASHREE/Desktop/data/BlinkIT Grocery Data.csv")
```

In [29]:

```
df.head(20)
```

Out[29]:

	Item Fat Content	Item Identifier	Item Type	Outlet Establishment Year	Outlet Identifier	Outlet Location Type	Outlet Size	Outlet Type	Item Visibility	Item Weight	Total 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
10	LF	FDX21	Snack Foods	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.084555	NaN	109.8912	5.0
11	Low Fat	NCU41	Health and Hygiene	2017	OUT035	Tier 2	Small	Supermarket Type1	0.052045	18.85	192.1846	5.0
12	Low Fat	FDL20	Fruits and Vegetables	2022	OUT018	Tier 3	Medium	Supermarket Type2	0.128938	17.10	112.3886	5.0
13	Low Fat	NCR54	Household	2000	OUT013	Tier 3	High	Supermarket Type1	0.090487	16.35	195.2110	5.0
14	Low Fat	FDH19	Meat	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.032928	NaN	173.1738	5.0
15	Regular	FDB57	Fruits and Vegetables	2017	OUT035	Tier 2	Small	Supermarket Type1	0.018802	20.25	222.1772	5.0
16	Low Fat	FDO23	Breads	2022	OUT018	Tier 3	Medium	Supermarket Type2	0.147024	17.85	93.7436	5.0
17	Low Fat	NCB07	Household	2012	OUT049	Tier 1	Medium	Supermarket Type1	0.077628	19.20	197.6110	5.0
18	Low Fat	FDJ56	Fruits and Vegetables	1998	OUT027	Tier 3	Medium	Supermarket Type3	0.182515	NaN	98.7700	5.0
19	Low Fat	DRN47	Hard Drinks	2022	OUT018	Tier 3	Medium	Supermarket Type2	0.016895	12.10	178.5660	5.0

In [30]:

```
df.tail(10)
```

Out[30]:

	Item Fat Content	Item Identifier	Item Type	Outlet Establishment Year	Outlet Identifier	Outlet Location Type	Outlet Size	Outlet Type	Item Visibility	Item Weight	Total 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 [31]:

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

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

FIELD INFO

In [32]:

```
df.columns
```

Out[32]:

```
Index(['Item Fat Content', 'Item Identifier', 'Item Type',
       'Outlet Establishment Year', 'Outlet Identifier',
       'Outlet Location Type', 'Outlet Size', 'Outlet Type', 'Item Visibility',
       'Item Weight', 'Total Sales', 'Rating'],
      dtype='object')
```

DATA TYPE

In [33]:

```
df.dtypes
```

Out[33]:

```
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
Total Sales                  float64
Rating                       float64
dtype: object
```

DATA CLEANING

In [ ]:

In [34]:

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

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

In [ ]:

In [35]:

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

In [36]:

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

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

BUSINESS REQUIREMENTS

KPI's REQUIREMENTS

In [42]:

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

#Avg Sales
avg_sales = df['Total Sales'].mean()

#No of Item Sold
no_of_item_sold = df['Total Sales'].count()

#Avg Ratings
avg_ratings = df['Rating'].mean()

#Display

print(f"Total Sales: ${total_sales:,.0f}")
print(f"Averag Sales: ${avg_sales:,.0f}")
print(f"No of Items Sold: {no_of_item_sold:,.0f}")
print(f"Average Ratings: {avg_ratings:,.1f}")
```

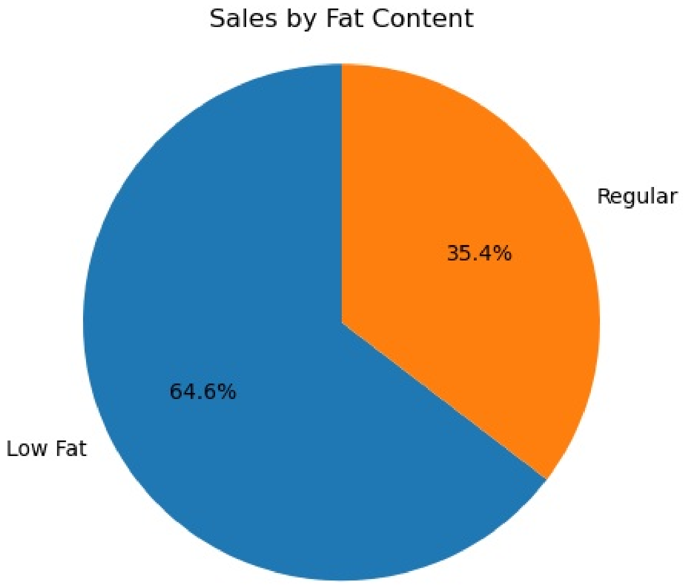
```
Total Sales: $1,201,681
Averag Sales: $141
No of Items Sold: 8,523
Average Ratings: 4.0
```

CHARTS REQUIREMENTS

Total sales by Fat Content

In [48]:

```
sales_by_fat = df.groupby('Item Fat Content')['Total 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()
```

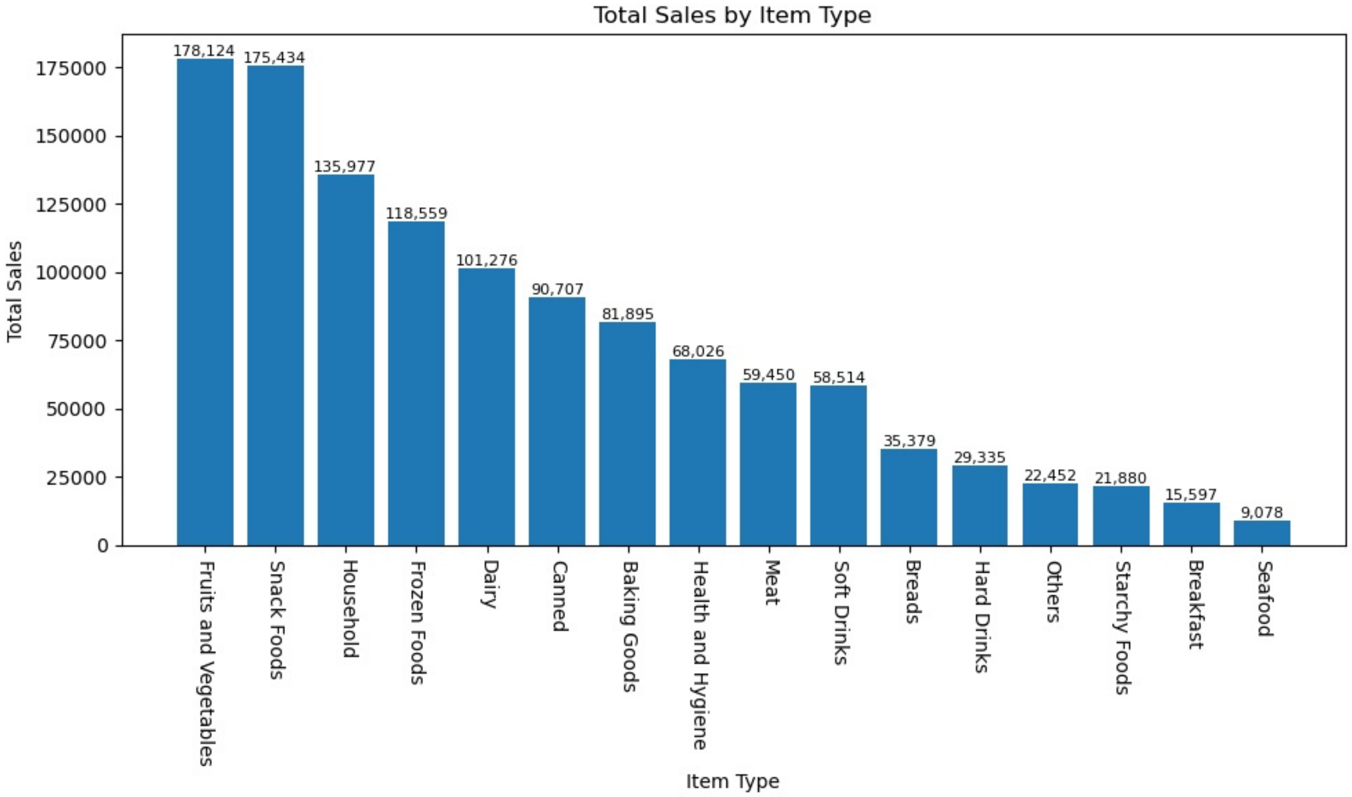


Total sales by Item Type

In [56]:

```
Sales_by_type = df.groupby('Item Type')['Total Sales'].sum().sort_values(ascending=False)
```

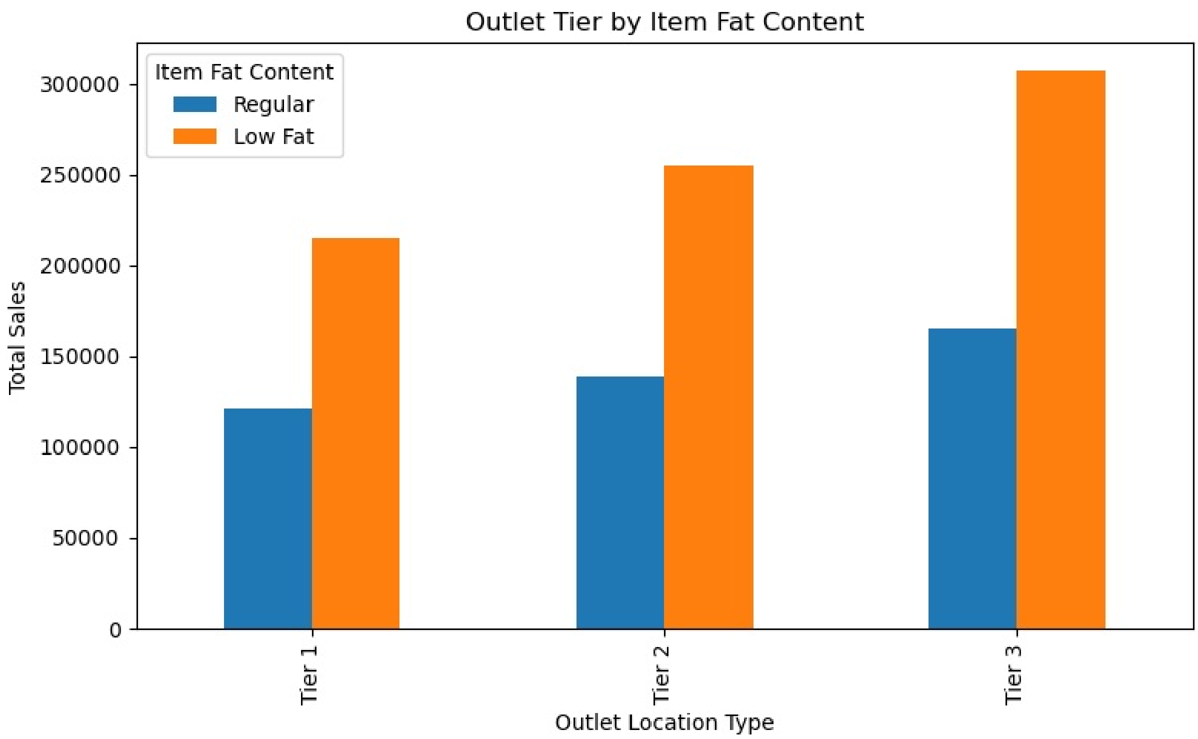
```
plt.figure(figsize=(10,6))
bars = plt.bar(Sales_by_type.index, Sales_by_type.values)

plt.xticks(rotation=-90)
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

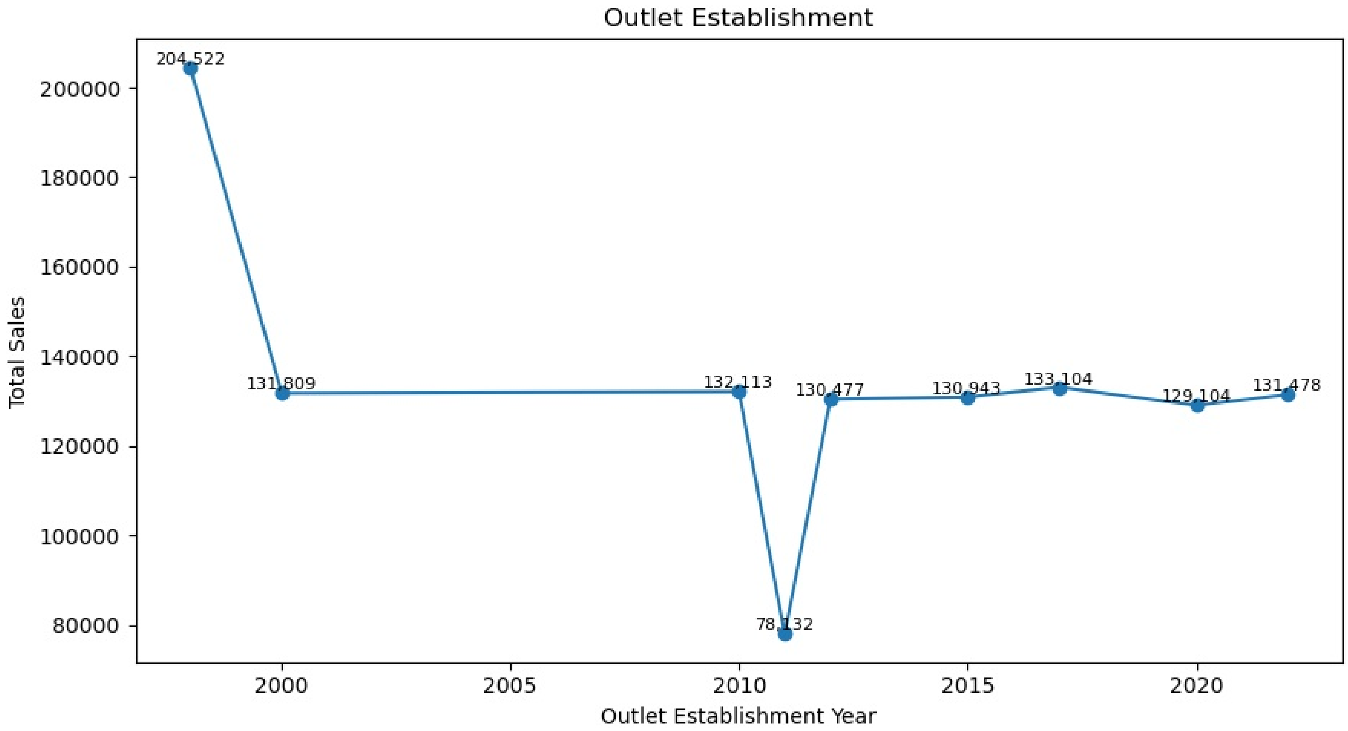
In [61]:

```
grouped = df.groupby(['Outlet Location Type','Item Fat Content'])['Total 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 Type')
plt.ylabel('Total Sales')
plt.legend(title='Item Fat Content')
plt.tight_layout()
plt.show()
```

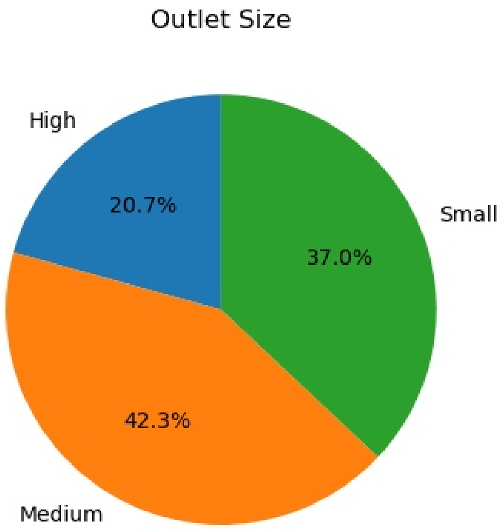


Total Sales by Outlet Establishment

In [71]:
```
sales_by_year= df.groupby('Outlet Establishment Year')['Total 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()
```



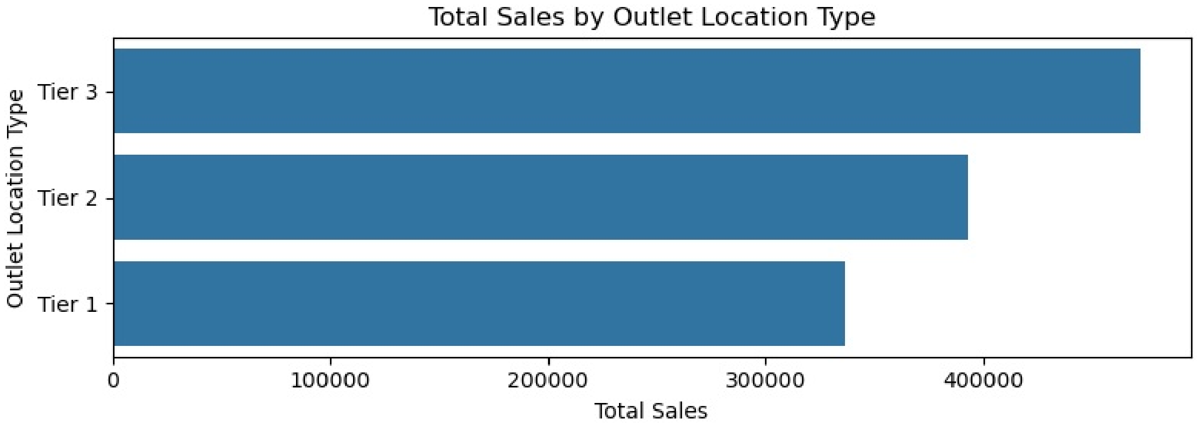
In [75]:
```
sales_by_size = df.groupby('Outlet Size')['Total Sales'].sum()

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 [80]:

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

plt.figure(figsize=(8,3)) #Smaller height , enough width
ax = sns.barplot(x='Total 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()  #Ensure Layout fits without scroll
plt.show()
```



In [ ]: