Blinkit Analysis

July 19, 2025

1 DATA ANALYSIS PYTHON PROJECT-BLINKIT ANALYSIS

Import Libraries

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

Import Raw Data

[2]:
```
df=pd.read_csv('BlinkIT Grocery Data.csv')
```

Sample Data

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

[3]:
```
  Item_FatContent Item_Identifier              Item_Type  \
0         Regular           FDX32  Fruits and Vegetables
1         Low Fat           NCB42     Health and Hygiene
2         Regular           FDR28           Frozen Foods
3         Regular           FDL50                 Canned
4         Low Fat           DRI25            Soft Drinks
5         low fat           FDS52           Frozen Foods
6         Low Fat           NCU05     Health and Hygiene
7         Low Fat           NCD30              Household
8         Low Fat           FDW20  Fruits and Vegetables
9         Low Fat           FDX25                 Canned

   Outlet_Establishment_Year Outlet_Identifier Outlet_Location_Type  \
0                        2012           OUT049               Tier 1
1                        2022           OUT018               Tier 3
2                        2016           OUT046               Tier 1
3                        2014           OUT013               Tier 3
4                        2015           OUT045               Tier 2
5                        2020           OUT017               Tier 2
6                        2011           OUT010               Tier 3
7                        2015           OUT045               Tier 2
```

```
8                       2014            OUT013               Tier 3
9                       2018            OUT027               Tier 3

  Outlet_Size        Outlet_Type  Item_Visibility  Item_Weight     Sales  \
0      Medium  Supermarket Type1         0.100014        15.10  145.4786
1      Medium  Supermarket Type2         0.008596        11.80  115.3492
2       Small  Supermarket Type1         0.025896        13.85  165.0210
3        High  Supermarket Type1         0.042278        12.15  126.5046
4       Small  Supermarket Type1         0.033970        19.60   55.1614
5       Small  Supermarket Type1         0.005505         8.89  102.4016
6       Small      Grocery Store         0.098312        11.80   81.4618
7       Small  Supermarket Type1         0.026904        19.70   96.0726
8        High  Supermarket Type1         0.024129        20.75  124.1730
9      Medium  Supermarket Type3         0.101562          NaN  181.9292

   Rating
0     5.0
1     5.0
2     5.0
3     5.0
4     5.0
5     5.0
6     5.0
7     5.0
8     5.0
9     5.0
```

[4]:
```
df.tail(10)
```

[4]:
```
     Item_FatContent Item_Identifier              Item_Type  \
8513         Regular           DRY23            Soft Drinks
8514         low fat           FDA11          Baking Goods
8515         low fat           FDK38                 Canned
8516         low fat           FDO38                 Canned
8517         low fat           FDG32  Fruits and Vegetables
8518         low fat           NCT53     Health and Hygiene
8519         low fat           FDN09            Snack Foods
8520         low fat           DRE13            Soft Drinks
8521             reg           FDT50                  Dairy
8522             reg           FDM58            Snack Foods

      Outlet_Establishment_Year Outlet_Identifier Outlet_Location_Type  \
8513                       2018            OUT027               Tier 3
8514                       2018            OUT027               Tier 3
8515                       2018            OUT027               Tier 3
8516                       2018            OUT027               Tier 3
8517                       2018            OUT027               Tier 3
```

```
8518                       2018           OUT027                Tier 3
8519                       2018           OUT027                Tier 3
8520                       2018           OUT027                Tier 3
8521                       2018           OUT027                Tier 3
8522                       2018           OUT027                Tier 3

     Outlet_Size        Outlet_Type  Item_Visibility  Item_Weight      Sales  \
8513      Medium  Supermarket Type3         0.108568          NaN    42.9112
8514      Medium  Supermarket Type3         0.043029          NaN    94.7436
8515      Medium  Supermarket Type3         0.053032          NaN   149.1734
8516      Medium  Supermarket Type3         0.072486          NaN    78.9986
8517      Medium  Supermarket Type3         0.175143          NaN   222.3772
8518      Medium  Supermarket Type3         0.000000          NaN   164.5526
8519      Medium  Supermarket Type3         0.034706          NaN   241.6828
8520      Medium  Supermarket Type3         0.027571          NaN    86.6198
8521      Medium  Supermarket Type3         0.107715          NaN    97.8752
8522      Medium  Supermarket Type3         0.000000          NaN   112.2544

      Rating
8513     4.0
8514     4.0
8515     4.0
8516     4.0
8517     4.0
8518     4.0
8519     4.0
8520     4.0
8521     4.0
8522     4.0
```

Size of Data

```
[5]: print('Size of Data:',df.shape)
```

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

Field Info

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

```
[6]: Index(['Item_FatContent', '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

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

[7]:

```
Item_FatContent               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

[8]:

```
print(df['Item_FatContent'].unique())
```

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

[9]:

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

[10]:

```
print(df['Item_FatContent'].unique())
```

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

1.1 Business Requirements

1.1.1 1. KPI's REQUIREMENTS

[11]:

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

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

# No of Items Sold
No_of_items=df['Sales'].count()

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

### Displays

print(f"Total Sales: ${Total_Sales:,.0f}")
print(f"Average Sales: ${Avg_Sales:,.1f}")
print(f"No of Items Sold: {No_of_items:,.0f}")
print(f"Average Rating: {Avg_Rating:,.1f}")
```

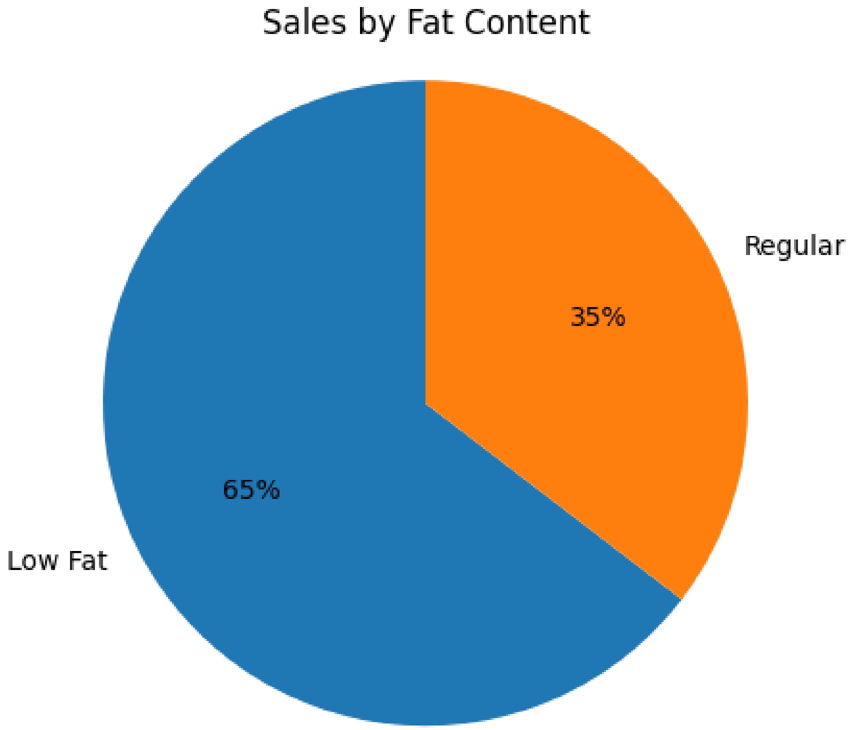
```
Total Sales: $1,201,681
Average Sales: $141.0
No of Items Sold: 8,523
Average Rating: 4.0
```

1.1.2 1. CHARTS REQUIREMENTS

Total Sales by Fat Content

[12]:

```
sales_by_fat=df.groupby('Item_FatContent')['Sales'].sum()
plt.pie(sales_by_fat,labels=sales_by_fat.index,
         autopct='%.0f%%',
        startangle=90  )
plt.title('Sales by Fat Content')
plt.axis('equal')
plt.show()
```



Explanation df.groupby(‘Item_FatContent’)[‘Sales’].sum():

Groups your data by the Item_FatContent column (like Low Fat, Regular, etc.).

For each fat content type, it sums up the sales.

Result: A summary table with fat content types as the index and their total sales as values.

plt.pie(): Creates a pie chart.

sales_by_fat: Values (total sales) used to define the size of each slice.

labels=sales_by_fat.index: Sets the labels (e.g. Low Fat, Regular) based on the fat content types.

autopct=‘%.0f%%’: Adds percentage labels on the slices (formatted to 0 decimal place).

startangle=90: Rotates the chart to start at 90°, so the pie chart looks properly oriented.

plt.axis(‘equal’) : Ensures the pie chart is drawn as a circle (not an ellipse).

Total Sales by Item Type

[13]:

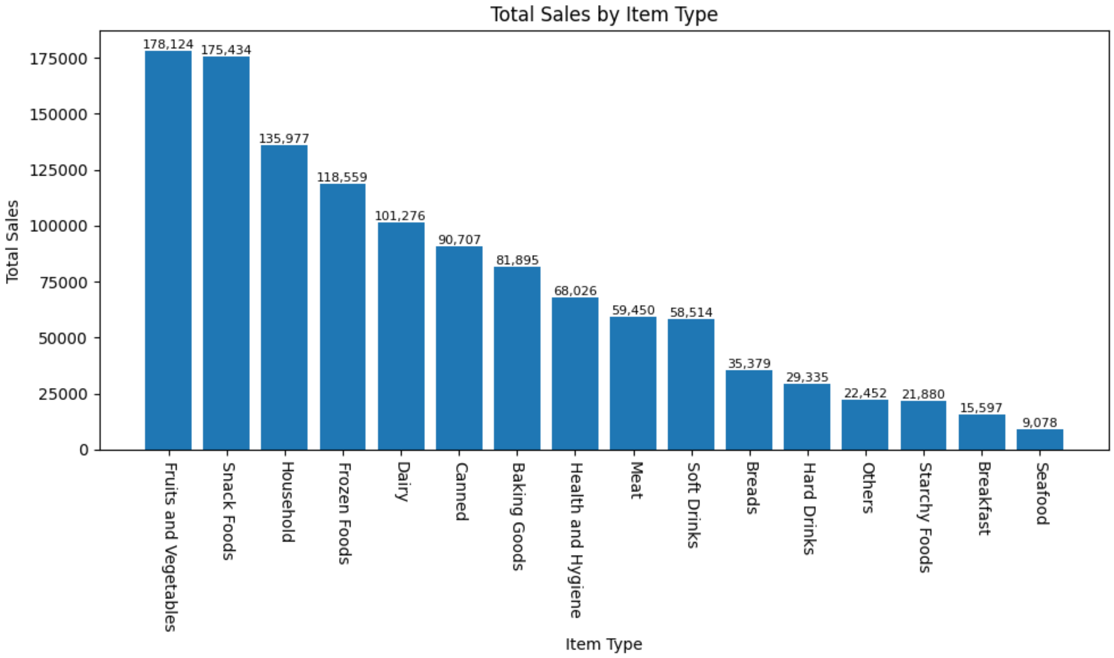
```
sales_by_type=df.groupby('Item_Type')['Sales'].sum().
 ↪sort_values(ascending=False)

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

plt.xticks(rotation=-90)
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()
```



Explanation bars = plt.bar(sales_by_type.index, sales_by_type.values)
plt.bar(): Creates a vertical bar chart.

X-axis: sales_by_type.index (item types).

Bar heights (Y-axis): sales_by_type.values (total sales).

bars variable stores the bar objects so you can later annotate them.

plt.xticks(rotation=-90)
Rotates X-axis labels (item types) vertically to avoid label overlap, improving readability.

bar.get_x() + bar.get_width()/2: Finds the horizontal center of each bar.

bar.get_height(): Gets the height (sales total) to position the text right above the bar.

f'{bar.get_height():,.0f}': Formats the sales value as a number with commas (like 250,000).

ha='center', va='bottom': Aligns text horizontally centered, and positioned just above the bar.

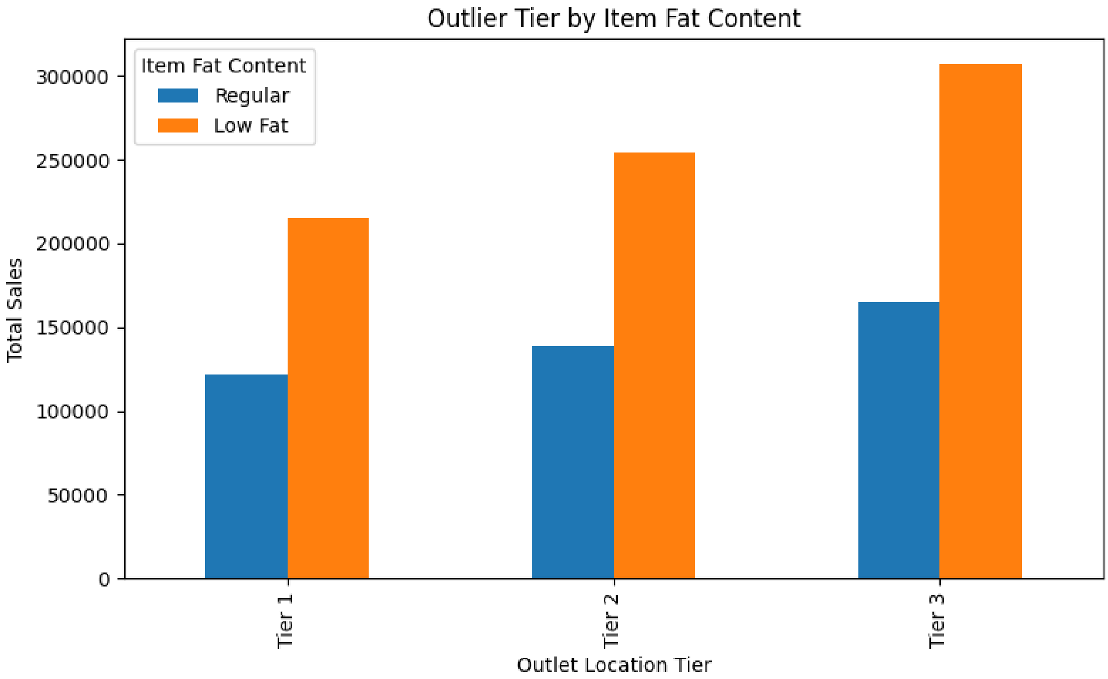
fontsize=8: Sets the font size smaller for clarity.

plt.tight_layout() Automatically adjusts spacing to prevent label cutoffs or overlapping.

Fat Content by Outlet for Total Sales

[14]:
```
grouped=df.groupby(['Outlet_Location_Type','Item_FatContent'])['Sales'].sum().
  ↪unstack()
grouped=grouped[['Regular','Low Fat']]
```

```
ax=grouped.plot(kind='bar',figsize=(8,5),title='Outlier 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()
```



Explanation .unstack():
Converts the second grouping level (Item_FatContent) into separate columns.

This reshapes your data so that each Outlet_Location_Type appears as a row, and each Item_FatContent (Regular, Low Fat) becomes a separate column.

grouped = grouped[[‘Regular’, ‘Low Fat’]]
Selects only the Regular and Low Fat columns.

Excludes any other fat content types (like “Low Fat (Type 2)”, etc.) for clean plotting.

grouped.plot():

kind=‘bar’: Creates a vertical grouped bar chart.

figsize=(8,5): Sets figure size.

title=‘Outlier Tier by Item Fat Content’: Adds a chart title (though your title has a typo: should probably be “Outlet Tier by Item Fat Content”).

Each outlet location tier will have grouped bars: one for Regular, one for Low Fat.

Total Sales by Outlet Establishment

[15]:

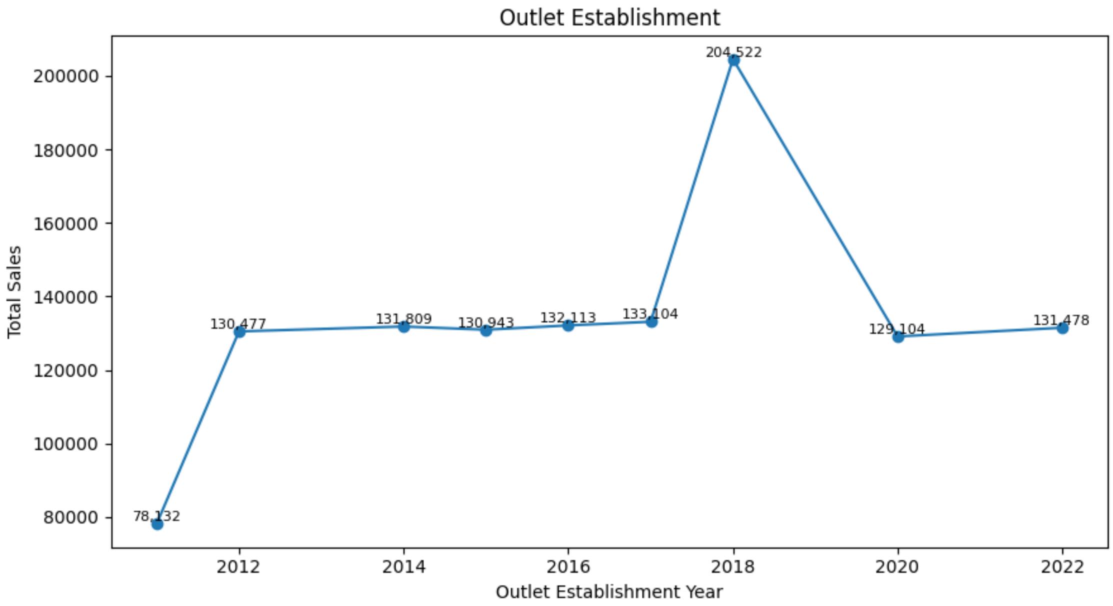
```
Sales_by_year=df.groupby('Outlet_Establishment_Year')['Sales'].sum().
  ↪sort_index()

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()
```



Explanation .sort_index():
Sorts the years in ascending order (makes the X-axis chronological).

plt.figure(figsize=(9,5)): Sets the size of the chart (9 inches wide, 5 inches tall).

plt.plot():

X-axis: Years (outlet establishment years).

Y-axis: Total sales for each year.

marker=‘o’: Places a dot at each data point.

linestyle=‘-’: Connects the points with a solid line.

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)

Loops through each point (x=year, y=sales).

Places the sales value above each point:

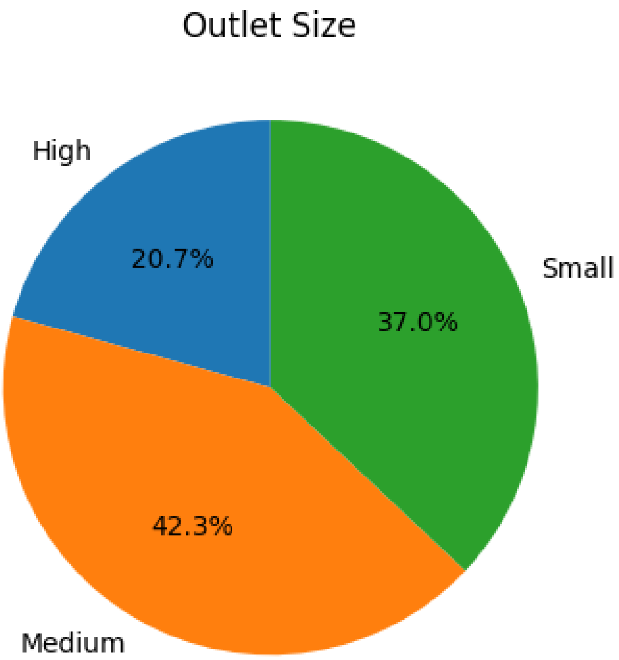
f’{y:,.0f}’: Formats the sales value with commas (like 120,000).

ha=‘center’, va=‘bottom’: Centers the text horizontally and positions it just above the marker.

fontsize=8: Small, readable font size.

Sales by Outlet Size

[19]:
```
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=90)
plt.title('Outlet Size')
plt.tight_layout()
plt.show()
```



Explanation plt.figure(figsize=(4,4)):

Sets the figure size to a small, square chart (4 inches by 4 inches).

plt.pie():

sales_by_size: The sales totals determine the size of each pie slice.

labels=sales_by_size.index: Each slice is labeled by the outlet size category.

autopct='%1.1f%%': Displays the percentage share of each slice (formatted to 1 decimal place).

startangle=90: Rotates the starting position of the first slice to the top for visual consistency.

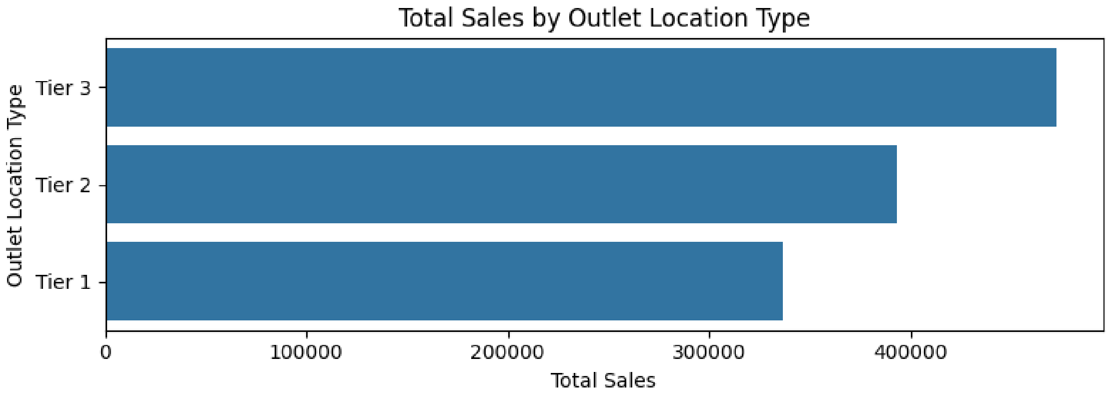
Sales by Outlet Location

[20]:

```
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=(8,3))

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()
plt.show()
```



Explanation sns.barplot():

x='Sales': Bar lengths represent total sales.

y='Outlet_Location_Type': Bars are arranged horizontally by location type.

data=sales_by_location: Uses your grouped and sorted DataFrame.

Result: Horizontal bar chart, where:

Longer bars = higher sales.

Locations are listed along the Y-axis.

```
[ ]:
```