

Sales and Outlet Analysis for Blinkit Retail Data

Data Import and Exploration

```
In [5]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

In [6]: df = pd.read_csv("C:/Users/sona3/Downloads/blinkit_data.csv")

In [4]: df.head(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 |
|---|------------------|-----------------|-----------------------|---------------------------|-------------------|----------------------|-------------|-------------------|-----------------|-------------|----------|--------|
| 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 | 12.15 | 126.5046 | 5.0 |
| 3 | Regular | FDL50 | Canned | 2000 | OUT013 | Tier 3 | High | Supermarket Type1 | 0.042278 | 12.15 | 126.5046 | 5.0 |
| 4 | Low Fat | DR125 | Soft Drinks | 2015 | OUT045 | Tier 2 | Small | Supermarket Type1 | 0.033970 | 19.60 | 55.1614 | 5.0 |
| 5 | Low Fat | FD552 | 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 | NCW30 | 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 | 0.0294129 | 20.75 | 124.1730 | 5.0 |
| 9 | Low Fat | FDX25 | Canned | 1998 | OUT027 | Tier 3 | Medium | Supermarket Type3 | 0.101562 | NaN | 181.8292 | 5.0 |

```
In [5]: df.shape
Out [5]: (8523, 12)

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')

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

In [8]: df['Item Fat Content'].unique()

Out [8]: array(['Regular', 'Low Fat', 'low fat', 'LF', 'reg'], dtype=object)

In [11]: df['Item Type'].unique()

Out [11]: array(['Fruits and Vegetables', 'Health and Hygiene', 'Frozen Foods',
        'Canned', 'Soft Drinks', 'Household', 'Snack Foods', 'Meat',
        'Breads', 'Hard Drinks', 'Others', 'Dairy', 'Breakfast',
        'Baking Goods', 'Seafood', 'Starchy Foods'], dtype=object)

In [13]: df['Outlet Location Type'].unique()

Out [13]: array(['Tier 1', 'Tier 3', 'Tier 2'], dtype=object)

In [14]: df['Outlet Size'].unique()

Out [14]: array(['Medium', 'Small', 'High'], dtype=object)

In [15]: df['Outlet Type'].unique()

Out [15]: array(['Supermarket Type1', 'Supermarket Type2', 'Grocery Store',
        'Supermarket Type3'], dtype=object)
```

Data Cleaning

```
In [38]: df['Item Fat Content'] = df['Item Fat Content'].replace(['Low Fat', 'low fat',
        'reg':'Regular'])

In [39]: df['Item Fat Content'].unique()

Out [39]: array(['Regular', 'Low Fat'], dtype=object)

In [70]: df.columns = df.columns.str.strip()

In [17]: print(df.isnull().sum())

Item Fat Content      0
Item Identifier      0
Item Type            0
Outlet Establishment Year  0
Outlet Identifier      0
Outlet Location Type    0
Outlet Size           0
Outlet Type           0
Item Visibility        0
Item Weight          1463
Sales                0
Rating               0
dtype: int64

In [19]: df['Item Weight'] = df.groupby('Item Type')['Item Weight'].transform(      # Keeps item type-level distribution more realistic
        lambda x: x.fillna(x.mean()))

In [20]: print(df.isnull().sum())

Item Fat Content      0
Item Identifier      0
Item Type            0
Outlet Establishment Year  0
Outlet Identifier      0
Outlet Location Type    0
Outlet Size           0
Outlet Type           0
Item Visibility        0
Item Weight          1463
Sales                0
Rating               0
dtype: int64
```

Check Outliers

```
In [10]: Q1 = df['Sales'].quantile(0.25)
Q3 = df['Sales'].quantile(0.75)
IQR = Q3 - Q1

# Define bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Find outliers
sales_outliers = df[(df['Sales'] < lower_bound) | (df['Sales'] > upper_bound)]

# Output results
print(f"Total outliers in 'Sales': {len(sales_outliers)}")
print(sales_outliers['Sales'])

Total outliers in 'Sales': 0
Empty DataFrame
Columns: [Sales]
Index: []
```

Business Requirements

```
In [17]: # Total Sales
Total_sales = df['Sales'].sum()

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

# No of item sold
No_of_items_sold = df['Sales'].count()

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

In [21]: print(f"Total Sales : ${Total_sales :,.0f} ")
print(f"Avg Sales : ${Avg_sales :,.0f} ")
print(f"No of items sold : {No_of_items_sold :,.0f} ")
print(f"Avg Ratings : {Avg_ratings :,.1f} ")

Total Sales : $1,201,681
Avg Sales : $141
No of items sold : 8,523
Avg Ratings : 4.0
```

```
In [43]: df['Outlet Identifier'].unique()

Out [43]: array(['OUT049', 'OUT018', 'OUT046', 'OUT013', 'OUT045', 'OUT017',
        'OUT010', 'OUT027', 'OUT035', 'OUT019'], dtype=object)
```

Total Sales by Outlets

```
In [54]: Sales_by_outlet = df.groupby('Outlet Identifier')['Sales'].sum().round(2)

In [55]: Sales_by_outlet
```

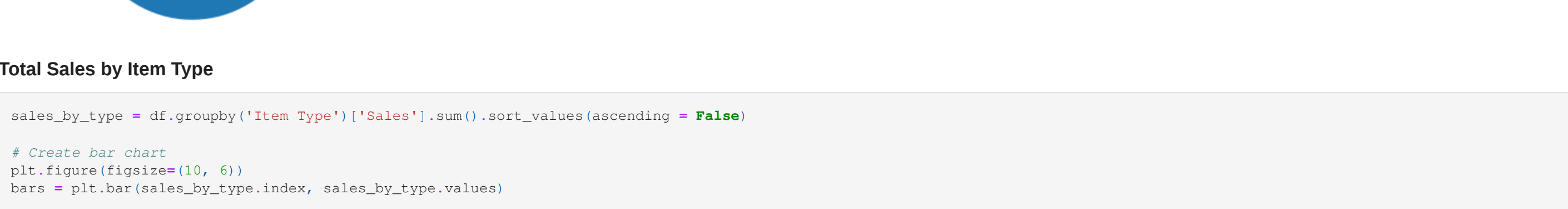
```
Out [55]: Outlet Identifier    Sales
OUT010      78131.56
OUT013     131809.02
OUT017     129105.98
OUT018     131497.77
OUT019      73807.58
OUT027     130714.67
OUT035     133103.91
OUT045     130942.78
OUT046     122113.97
OUT049     130476.86
Name: Sales, dtype: float64
```

Total Sales by Fat Content

```
In [26]: Sales_by_fat = df.groupby('Item Fat Content')['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()

Out [26]: <function matplotlib.pyplot.show(close=None, block=None)>
```



Total Sales by Item Type

```
In [29]: sales_by_type = df.groupby('Item Type')['Sales'].sum().sort_values(ascending = False)

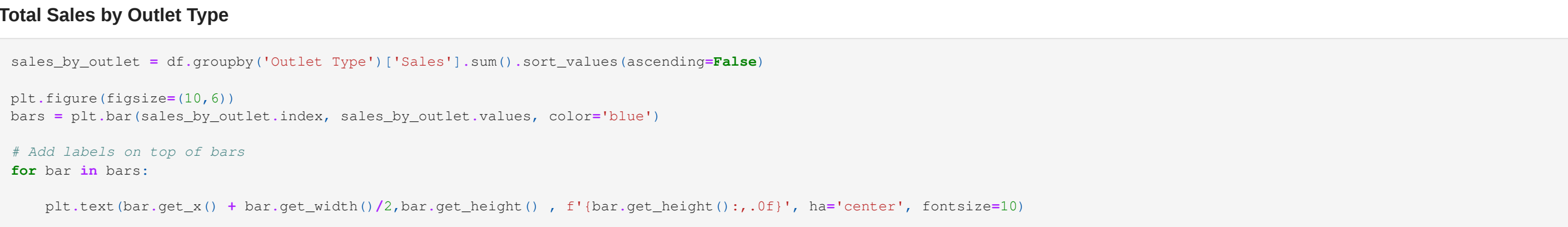
# Create bar chart
plt.figure(figsize=(10, 6))
bars = plt.bar(sales_by_type.index, sales_by_type.values)

plt.xticks(rotation = 75)
plt.xlabel('Item type')
plt.ylabel('Total Sales')
plt.title('Total Sales by Item Type')

# Add labels on top of each bar
for bar in bars:
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height(), # Position
             f'({bar.get_height():.0f})', ha='center', va='bottom', fontsize = 8) # Label & alignment

plt.tight_layout()
plt.show()

Total Sales by Item Type
```



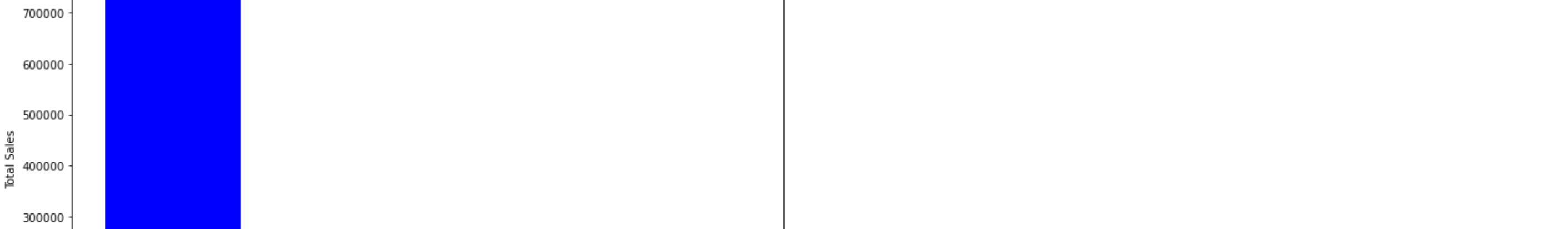
Total Sales by Outlet Type

```
In [27]: sales_by_outlet = df.groupby('Outlet Type')['Sales'].sum().sort_values(ascending=False)

plt.figure(figsize=(10,6))
bars = plt.bar(sales_by_outlet.index, sales_by_outlet.values, color='blue')

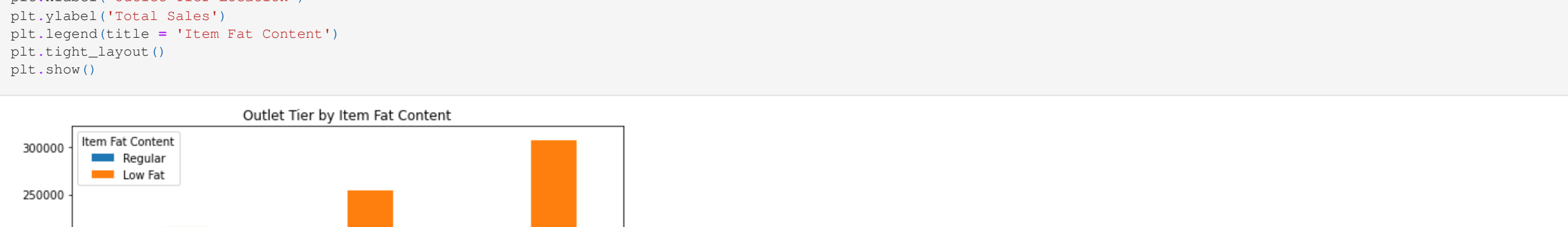
# Add labels on top of bars
for bar in bars:
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height(), # Position
             f'({bar.get_height():.0f})', ha='center', fontsize=10)

plt.title('Total Sales by Outlet Type')
plt.xlabel('Total Sales')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



Total Sales by Fat content and Outlet Location Type

```
In [31]: 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 Tier Location')
plt.ylabel('Total Sales')
plt.legend(title = 'Item Fat Content')
plt.tight_layout()
plt.show()
```

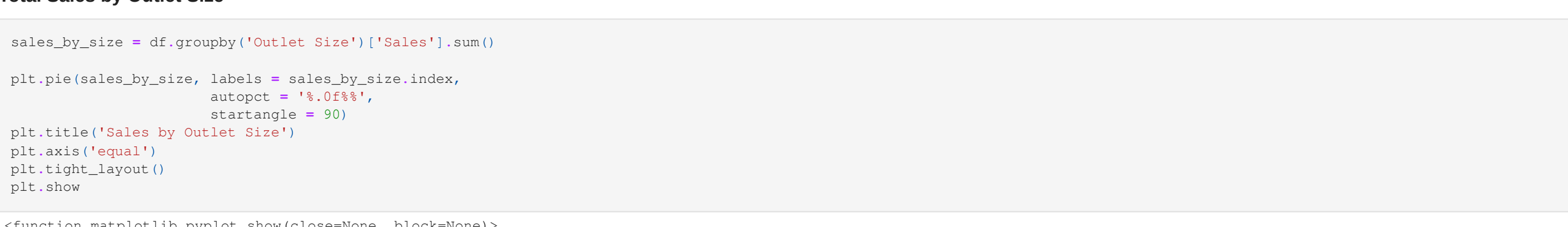


Total Sales by Outlet Size

```
In [32]: sales_by_size = df.groupby('Outlet Size')['Sales'].sum()

plt.pie(sales_by_size, labels = sales_by_size.index,
        autopct = '%.0f%',
        startangle = 90)
plt.title('Sales by Outlet Size')
plt.axis('equal')
plt.tight_layout()
plt.show()

Out [32]: <function matplotlib.pyplot.show(close=None, block=None)>
```



Total Sales by Outlet Location

```
In [33]: 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()
```



Average Rating by Item Type

```
In [56]: avg_rating_by_item = df.groupby('Item Type')['Rating'].mean().round(2).sort_values(ascending=False)
print(avg_rating_by_item)

Item Type      Rating
Meat           4.02
Household      4.00
Canned         3.99
Health and Hygiene  3.99
Baking Goods   3.98
Dairy          3.97
Frozen Foods    3.97
Fruits and Vegetables  3.96
Seafood        3.96
Others         3.95
Snack Foods    3.95
Breakfast     3.93
Soft Drinks    3.92
Starchy Foods  3.92
Hard Drinks    3.91
Breads        3.88
Name: Rating, dtype: float64
```

Which outlet has max sales?

```
In [52]: sales_by_outlet = df.groupby('Outlet Identifier')['Sales'].sum()

max_outlet = sales_by_outlet.idxmax()
max_sales = sales_by_outlet.max()

print(f"Outlet with max total sales: {max_outlet}")
print(f"Total Sales: {max_sales:,}")

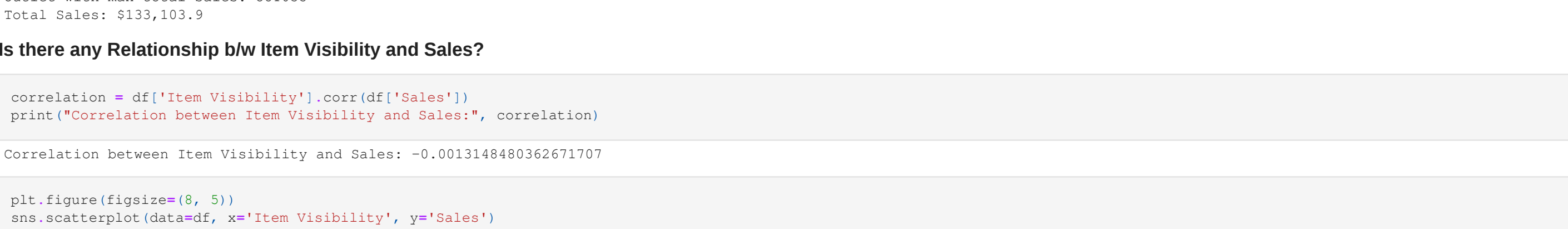
Outlet with max total sales: OUT035
Total Sales: $133,103.9
```

Is there any Relationship blw Item Visibility and Sales?

```
In [61]: correlation = df['Item Visibility'].corr(df['Sales'])
print("Correlation between Item Visibility and Sales:", correlation)

Correlation between Item Visibility and Sales: -0.0013148480362671707
```

```
In [60]: plt.figure(figsize=(8, 5))
sns.scatterplot(data=df, x='Item Visibility', y='Sales')
plt.title('Relationship Between Item Visibility and Sales')
plt.xlabel('Item Visibility')
plt.ylabel('Sales')
plt.show()
```



This scatter plot clearly shows that Item Visibility does not significantly affect Sales as well as Correlation is also close to 0.

Total Sales by Outlet establishment Year

```
In [75]: sales_by_year = df.groupby('Outlet Establishment Year')['Sales'].sum().reset_index()

plt.figure(figsize=(10,6))
sns.scatterplot(data=sf, x='Outlet Establishment Year', sales_by_year['Sales'], marker='o', linestyle='-', color='green')
plt.title('Total Sales by Outlet Establishment Year')
plt.xlabel('Establishment Year')
plt.ylabel('Total Sales')
plt.grid(True)
plt.tight_layout()
plt.show()
```

