grocery-sales

October 20, 2024

[1]:
```
import pandas as pd
df = pd.read_csv('C:\\Users\\ANUSHA\\Downloads\\Sales_Data_Cleaned.csv')
df.head()
```

[1]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	Profit
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8	OUT049	1999	Medium	Tier 2	Supermarket Type1	3735.1380	11.5
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.3	OUT018	2009	Medium	Tier 2	Supermarket Type2	443.4228	14.3
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6	OUT049	1999	Medium	Tier 2	Supermarket Type1	2097.2700	14.5
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.1	OUT010	1998	Medium	Tier 2	Grocery Store	732.3800	13.6
4	NCD19	8.93	Low Fat	0.000000	Household	53.9	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052	14.1

[2]:
```
df.isnull().any()
```

[2]:
```
Item_Identifier             False
Item_Weight                 False
Item_Fat_Content            False
```

```
Item_Visibility              False
Item_Type                    False
Item_MRP                     False
Outlet_Identifier            False
Outlet_Establishment_Year    False
Outlet_Size                  False
Outlet_Location_Type         False
Outlet_Type                  False
Item_Outlet_Sales            False
Profit                       False
dtype: bool
```

[3]:
```
# Products in dataset
unique_products_count = df['Item_Identifier'].nunique()
print(f"Number of unique products: {unique_products_count}")
```

```
Number of unique products: 1559
```

[4]:
```
# Product weight
filtered_weights = df['Item_Weight'].replace(0, pd.NA).dropna()
average_weight = filtered_weights.mean()
min_weight = filtered_weights.min()
max_weight = filtered_weights.max()
print("Product Weight Statistics (excluding 0 values):")
print(f"Average product weight: {average_weight:.2f} kg")
print(f"Smallest product weight: {min_weight:.2f} kg")
print(f"Largest product weight: {max_weight:.2f} kg")
```

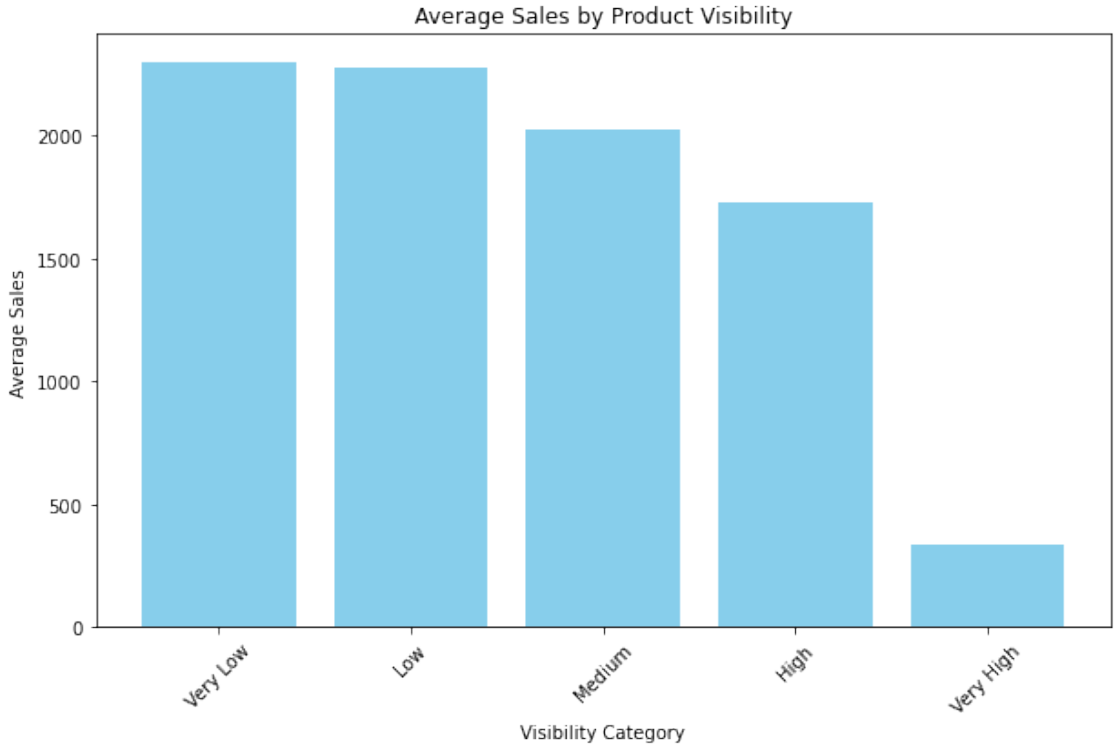
```
Product Weight Statistics (excluding 0 values):
Average product weight: 12.76 kg
Smallest product weight: 4.55 kg
Largest product weight: 100.00 kg
```

[5]:
```
# Item_Fat_Content
df['Item_Fat_Content'] = df['Item_Fat_Content'].str.lower().replace({'lf': 'low␣
 ↪fat', 'low fat': 'low fat','reg': 'regular','regular':␣
 ↪'regular','regularular':'regular'})
fat_content_count = df['Item_Fat_Content'].value_counts()
print("Count of products by fat content:")
print(fat_content_count)
```

```
Count of products by fat content:
low fat    5517
regular    3006
Name: Item_Fat_Content, dtype: int64
```

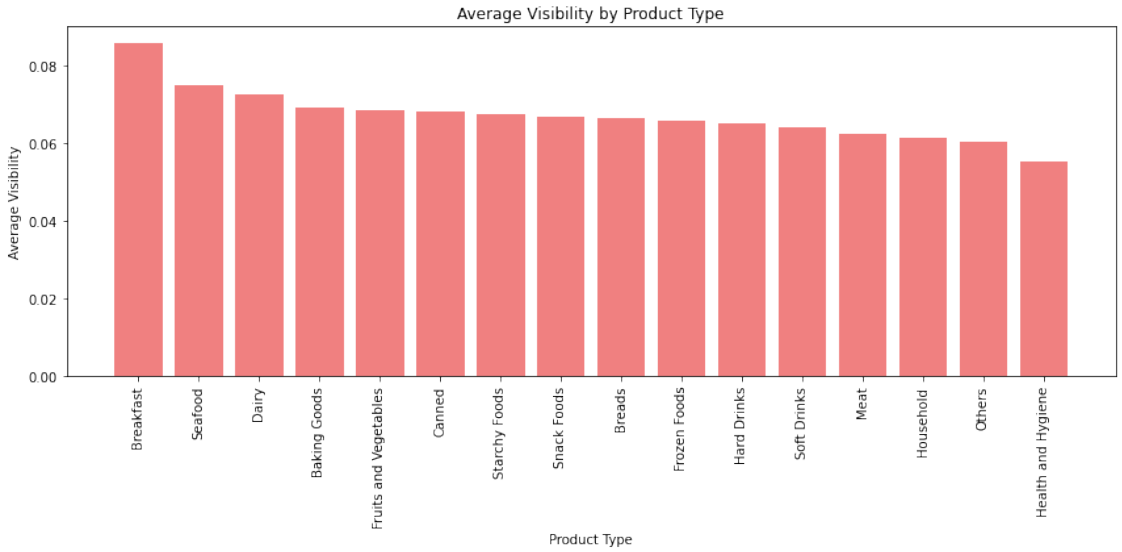
[6]:
```
# sales by fat content
import plotly.express as px
fat_content_sales = df.groupby('Item_Fat_Content')['Item_Outlet_Sales'].sum().
 ↪reset_index()
fig = px.
 ↪bar(fat_content_sales,x='Item_Fat_Content',y='Item_Outlet_Sales',title='Total␣
 ↪Sales by Item Fat Content',labels={'Item_Outlet_Sales': 'Total Sales',␣
 ↪'Item_Fat_Content': 'Fat Content'},text='Item_Outlet_Sales')
fig.update_traces(texttemplate='%{text:.2f}', textposition='outside')
fig.update_layout(xaxis_title='Item Fat Content',yaxis_title='Total␣
 ↪Sales',hovermode="closest")
fig.show()
```

[7]:
```
import matplotlib.pyplot as plt
visibility_bins = [0, 0.05, 0.1, 0.15, 0.2, 1]
visibility_labels = ['Very Low', 'Low', 'Medium', 'High', 'Very High']

# products based on visibility
df['Visibility_Category'] = pd.cut(df['Item_Visibility'], bins=visibility_bins,␣
 ↪labels=visibility_labels)
sales_by_visibility = df.groupby('Visibility_Category')['Item_Outlet_Sales'].
 ↪mean().reset_index()
plt.figure(figsize=(10, 6))
plt.bar(sales_by_visibility['Visibility_Category'],␣
 ↪sales_by_visibility['Item_Outlet_Sales'], color='skyblue')
plt.title('Average Sales by Product Visibility')
plt.xlabel('Visibility Category')
plt.ylabel('Average Sales')
plt.xticks(rotation=45)
plt.show()
print(sales_by_visibility)
```



	Visibility_Category	Item_Outlet_Sales
0	Very Low	2299.537585
1	Low	2279.229840
2	Medium	2020.919905
3	High	1725.698117
4	Very High	336.999142

[8]:
```
#Visibility by product type
visibility_by_item_type = df.groupby('Item_Type')['Item_Visibility'].mean().
 ↪sort_values(ascending=False).reset_index()
plt.figure(figsize=(12, 6))
plt.bar(visibility_by_item_type['Item_Type'],␣
 ↪visibility_by_item_type['Item_Visibility'], color='lightcoral')
plt.title('Average Visibility by Product Type')
plt.xlabel('Product Type')
plt.ylabel('Average Visibility')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
print(visibility_by_item_type)
```

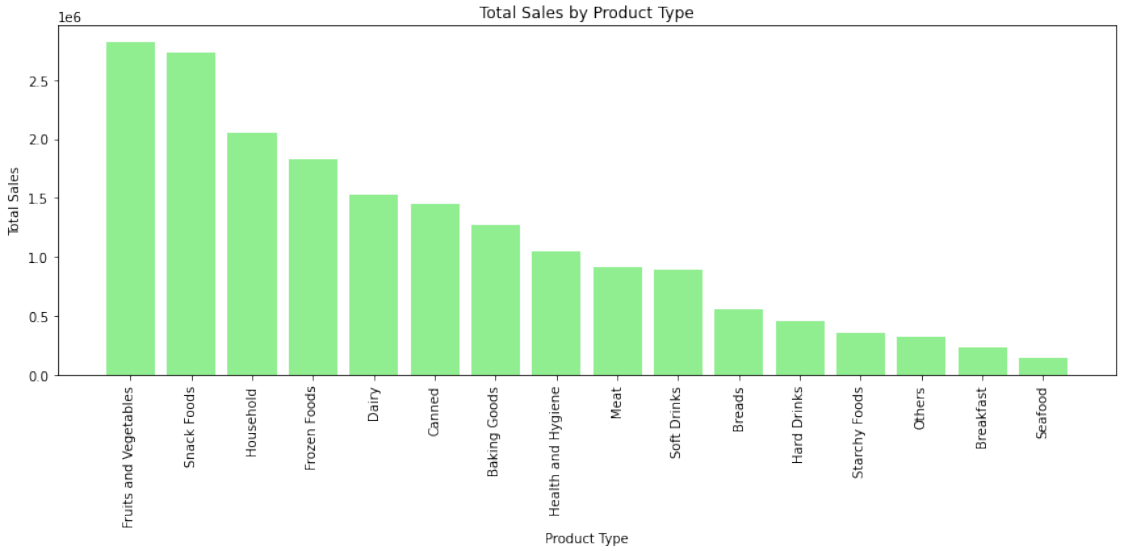


	Item_Type	Item_Visibility
0	Breakfast	0.085723
1	Seafood	0.074976
2	Dairy	0.072427
3	Baking Goods	0.069169
4	Fruits and Vegetables	0.068513
5	Canned	0.068129
6	Starchy Foods	0.067564
7	Snack Foods	0.066850
8	Breads	0.066255
9	Frozen Foods	0.065645
10	Hard Drinks	0.064943
11	Soft Drinks	0.063972
12	Meat	0.062284
13	Household	0.061322
14	Others	0.060241
15	Health and Hygiene	0.055216

[9]:
```
#Total sales by product type
sales_by_item_type = df.groupby('Item_Type')['Item_Outlet_Sales'].sum().
 ↪sort_values(ascending=False).reset_index()
sales_by_item_type['Item_Outlet_Sales'] =␣
 ↪sales_by_item_type['Item_Outlet_Sales'].apply(lambda x: '{:,.0f}'.format(x))
print(sales_by_item_type)
plt.figure(figsize=(12, 6))
plt.bar(sales_by_item_type['Item_Type'],␣
 ↪sales_by_item_type['Item_Outlet_Sales'].str.replace(',', '').astype(float),␣
 ↪color='lightgreen')
```

```
plt.title('Total Sales by Product Type')
plt.xlabel('Product Type')
plt.ylabel('Total Sales')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

	Item_Type	Item_Outlet_Sales
0	Fruits and Vegetables	2,820,060
1	Snack Foods	2,732,786
2	Household	2,055,494
3	Frozen Foods	1,825,735
4	Dairy	1,522,594
5	Canned	1,444,151
6	Baking Goods	1,265,525
7	Health and Hygiene	1,045,200
8	Meat	917,566
9	Soft Drinks	892,898
10	Breads	553,237
11	Hard Drinks	457,793
12	Starchy Foods	351,401
13	Others	325,518
14	Breakfast	232,299
15	Seafood	148,868



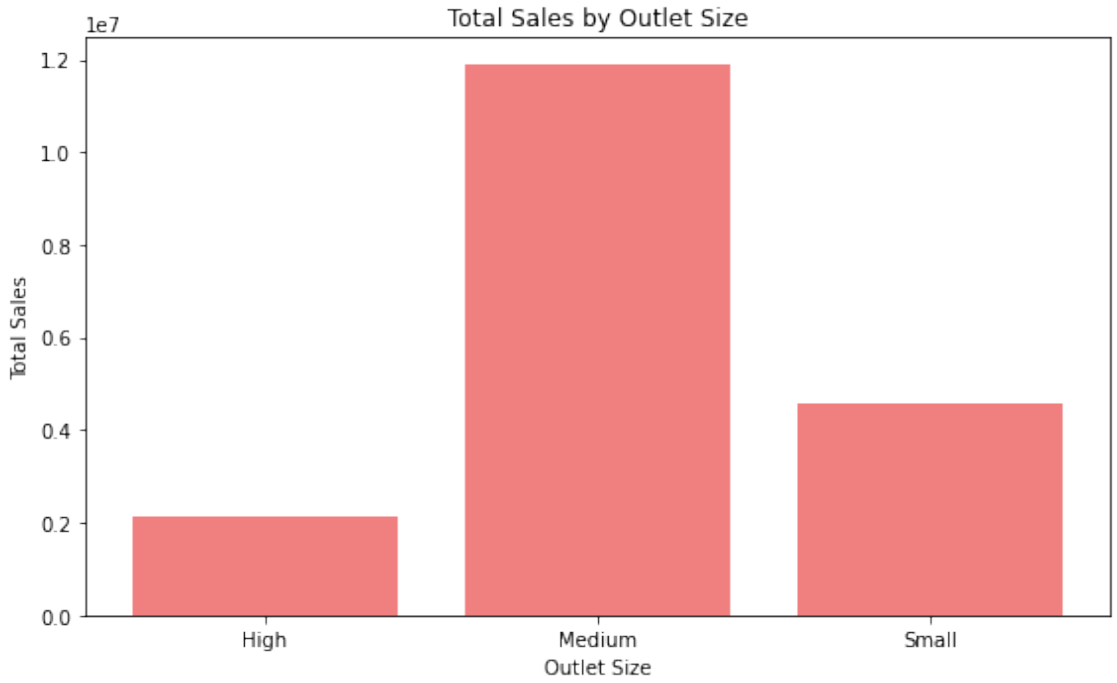
[10]:
```
# Price of product in Item_MRP
average_mrp = df['Item_MRP'].mean()
min_mrp = df['Item_MRP'].min()
max_mrp = df['Item_MRP'].max()
```

```
print(f"Average product price (MRP): {average_mrp:.2f}")
print(f"Lowest product price (MRP): {min_mrp:.2f}")
print(f"Highest product price (MRP): {max_mrp:.2f}")
```

```
Average product price (MRP): 141.00
Lowest product price (MRP): 31.30
Highest product price (MRP): 266.90
```

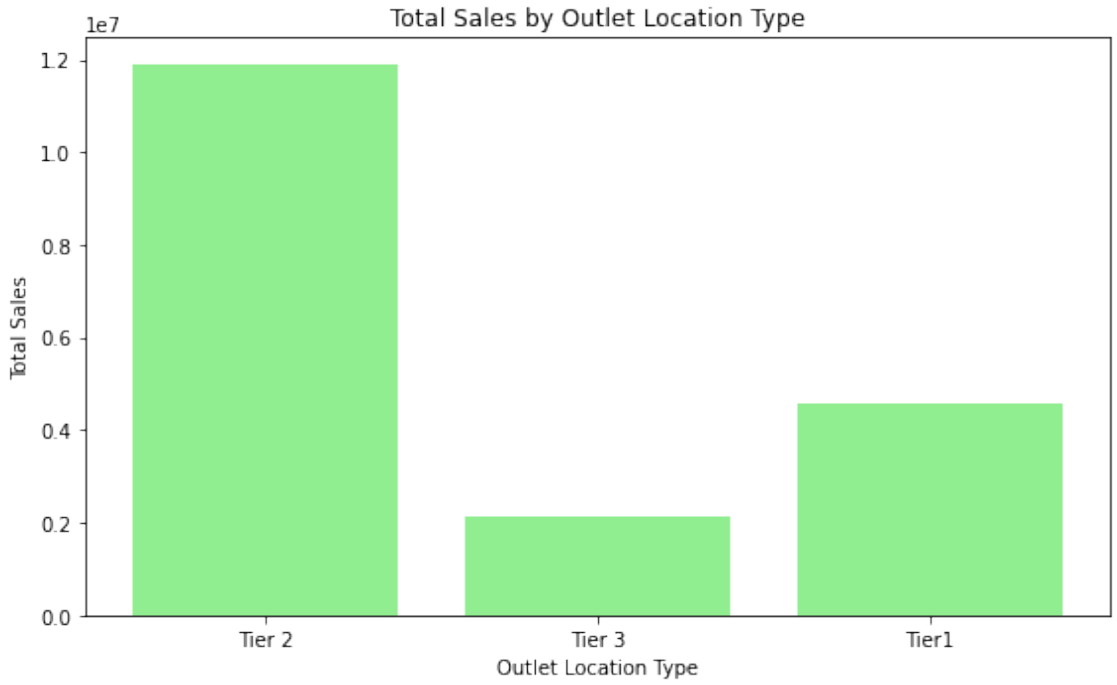
[11]:
```
# Sales by outlet size
sales_by_outlet_size = df.groupby('Outlet_Size')['Item_Outlet_Sales'].sum().
 ↪reset_index()
sales_by_outlet_size['Item_Outlet_Sales'] =␣
 ↪sales_by_outlet_size['Item_Outlet_Sales'].apply(lambda x: '{:,.0f}'.
 ↪format(x))
print(sales_by_outlet_size)
plt.figure(figsize=(8, 5))
plt.bar(sales_by_outlet_size['Outlet_Size'],␣
 ↪sales_by_outlet_size['Item_Outlet_Sales'].str.replace(',', '').
 ↪astype(float), color='lightcoral')
plt.title('Total Sales by Outlet Size')
plt.xlabel('Outlet Size')
plt.ylabel('Total Sales')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```

```
  Outlet_Size Item_Outlet_Sales
0        High         2,142,664
1      Medium        11,882,250
2       Small         4,566,212
```



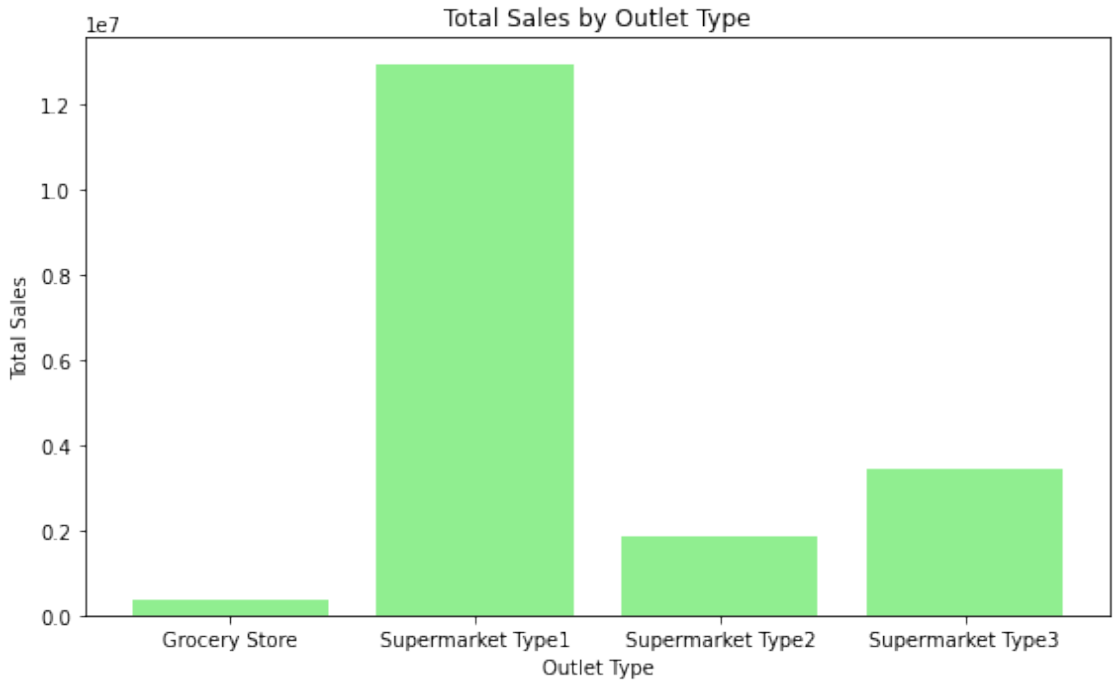
[12]:
```
# sales by outlet location type
sales_by_location_type = df.
 ↪groupby('Outlet_Location_Type')['Item_Outlet_Sales'].sum().reset_index()
sales_by_location_type['Item_Outlet_Sales'] =␣
 ↪sales_by_location_type['Item_Outlet_Sales'].apply(lambda x: '{:,.0f}'.
 ↪format(x))
print(sales_by_location_type)
plt.figure(figsize=(8, 5))
plt.bar(sales_by_location_type['Outlet_Location_Type'],␣
 ↪sales_by_location_type['Item_Outlet_Sales'].str.replace(',', '').
 ↪astype(float), color='lightgreen')
plt.title('Total Sales by Outlet Location Type')
plt.xlabel('Outlet Location Type')
plt.ylabel('Total Sales')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```

```
  Outlet_Location_Type Item_Outlet_Sales
0               Tier 2        11,882,250
1               Tier 3         2,142,664
2                Tier1         4,566,212
```

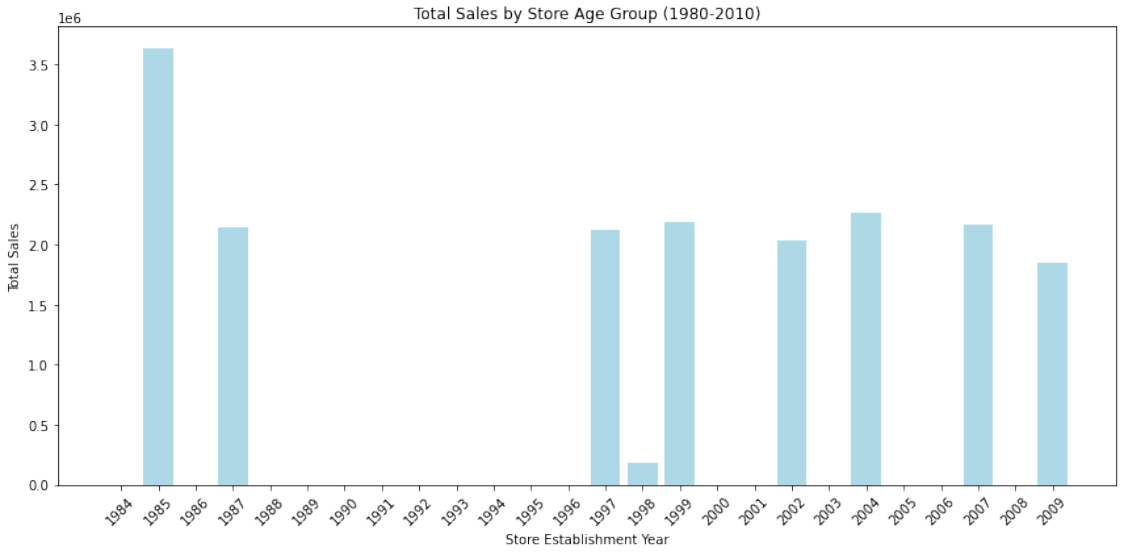


[13]:
```
# sales by outlet type
sales_by_outlet_type = df.groupby('Outlet_Type')['Item_Outlet_Sales'].sum().
 ↪reset_index()
sales_by_outlet_type['Item_Outlet_Sales'] =␣
 ↪sales_by_outlet_type['Item_Outlet_Sales'].apply(lambda x: '{:,.0f}'.
 ↪format(x))
print(sales_by_outlet_type)
plt.figure(figsize=(8, 5))
plt.bar(sales_by_outlet_type['Outlet_Type'],␣
 ↪sales_by_outlet_type['Item_Outlet_Sales'].str.replace(',', '').
 ↪astype(float), color='lightgreen')
plt.title('Total Sales by Outlet Type')
plt.xlabel('Outlet Type')
plt.ylabel('Total Sales')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```

```
         Outlet_Type Item_Outlet_Sales
0      Grocery Store           368,034
1  Supermarket Type1        12,917,342
2  Supermarket Type2         1,851,823
3  Supermarket Type3         3,453,926
```



[14]:
```
# sales by outlet age group
from datetime import datetime
current_year = datetime.now().year
df['Store_Age'] = current_year - df['Outlet_Establishment_Year']
bins = range(1984, 2011)
labels = [str(year) for year in range(1984, 2010)]
df['Store_Age_Group'] = pd.cut(df['Outlet_Establishment_Year'], bins=bins,␣
 ↪right=False, labels=labels)

sales_by_age_group = df.groupby('Store_Age_Group')['Item_Outlet_Sales'].sum().
 ↪reset_index()
sales_by_age_group['Item_Outlet_Sales'] =␣
 ↪sales_by_age_group['Item_Outlet_Sales'].apply(lambda x: '{:,.0f}'.format(x))
plt.figure(figsize=(12, 6))
plt.bar(sales_by_age_group['Store_Age_Group'],␣
 ↪sales_by_age_group['Item_Outlet_Sales'].str.replace(',', '').astype(float),␣
 ↪color='lightblue')
plt.title('Total Sales by Store Age Group (1980-2010)')
plt.xlabel('Store Establishment Year')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
print(sales_by_age_group)
```



	Store_Age_Group	Item_Outlet_Sales
0	1984	0
1	1985	3,633,620
2	1986	0
3	1987	2,142,664
4	1988	0
5	1989	0
6	1990	0
7	1991	0
8	1992	0
9	1993	0
10	1994	0
11	1995	0
12	1996	0
13	1997	2,118,395
14	1998	188,340
15	1999	2,183,970
16	2000	0
17	2001	0
18	2002	2,036,725
19	2003	0
20	2004	2,268,123
21	2005	0
22	2006	0
23	2007	2,167,465
24	2008	0
25	2009	1,851,823

[15]:
```
pip install plotly
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: plotly in c:\programdata\anaconda3\lib\site-
packages (5.6.0)
Requirement already satisfied: tenacity>=6.2.0 in
c:\programdata\anaconda3\lib\site-packages (from plotly) (8.0.1)
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packages
(from plotly) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

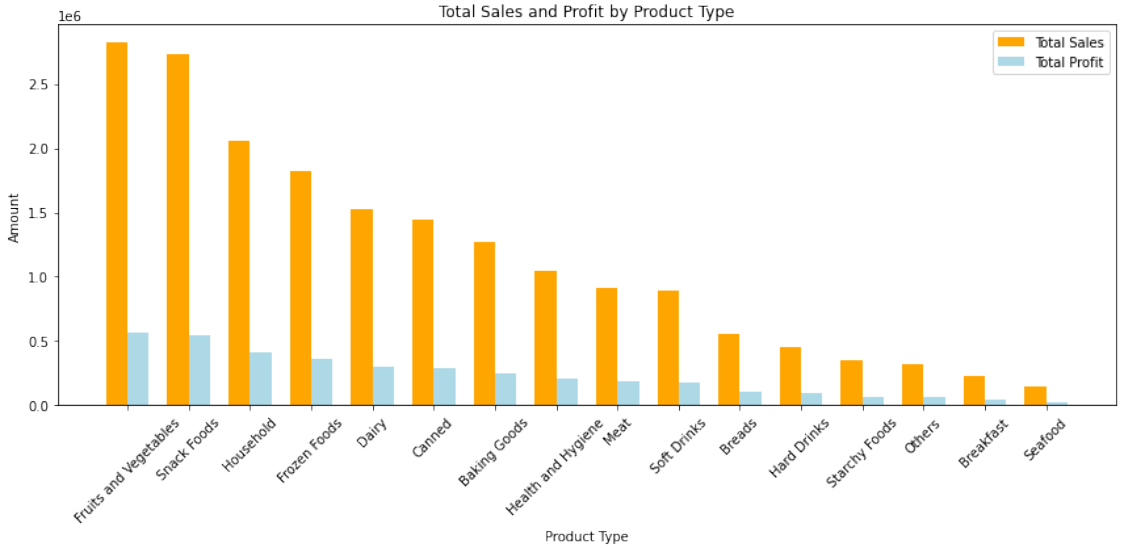
[16]:
```
#profit by product and outlet type
import plotly.express as px
profit_by_item_type = df.groupby('Item_Type')['Item_Outlet_Sales'].sum().
 ↪reset_index()
profit_by_item_type = profit_by_item_type.sort_values(by='Item_Outlet_Sales',␣
 ↪ascending=False)

fig_item = px.bar(profit_by_item_type, x='Item_Type', y='Item_Outlet_Sales',␣
 ↪title='Total Profit by Product Type (Item_Outlet_Sales as proxy)',␣
 ↪labels={'Item_Outlet_Sales': 'Total Sales (Proxy for Profit)', 'Item_Type':␣
 ↪'Product Type'})
fig_item.update_traces(hovertemplate='Product Type: %{x}<br>Total Sales: $%{y:,.
 ↪2f}<extra></extra>')
fig_item.show()

profit_by_outlet_type = df.groupby('Outlet_Type')['Item_Outlet_Sales'].sum().
 ↪reset_index()
profit_by_outlet_type = profit_by_outlet_type.
 ↪sort_values(by='Item_Outlet_Sales', ascending=False)

fig_outlet = px.bar(profit_by_outlet_type, x='Outlet_Type',␣
 ↪y='Item_Outlet_Sales', title='Total Profit by Outlet Type (Item_Outlet_Sales␣
 ↪as proxy)', labels={'Item_Outlet_Sales': 'Total Sales (Proxy for Profit)',␣
 ↪'Outlet_Type': 'Outlet Type'})
fig_outlet.update_traces(hovertemplate='Outlet Type: %{x}<br>Total Sales: $%{y:
 ↪,.2f}<extra></extra>')
fig_outlet.show()
```

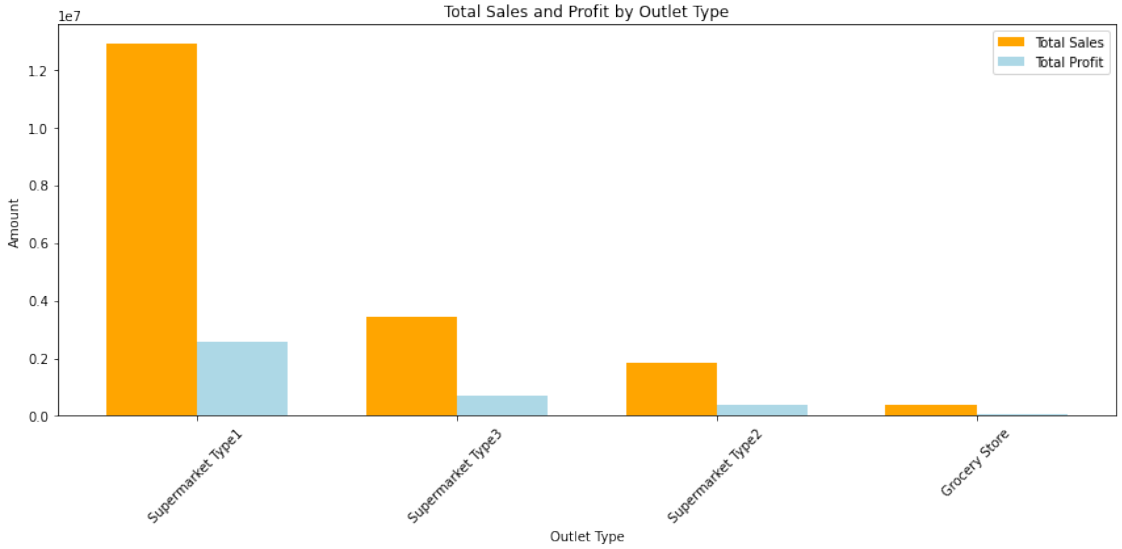
[17]:
```
# sales and profit by product type
df['Profit'] = df['Item_Outlet_Sales'] * 0.2  # Example profit calculation;␣
 ↪adjust if you have specific logic

sales_profit_by_item_type = df.groupby('Item_Type').
 ↪agg(Total_Sales=('Item_Outlet_Sales', 'sum'),Total_Profit=('Profit', 'sum')).
 ↪reset_index()
sales_profit_by_item_type = sales_profit_by_item_type.
 ↪sort_values(by='Total_Sales', ascending=False)
```

```
sales_profit_by_item_type['Total_Sales'] =␣
 ↪sales_profit_by_item_type['Total_Sales'].apply(lambda x: f"{x:,.2f}")
sales_profit_by_item_type['Total_Profit'] =␣
 ↪sales_profit_by_item_type['Total_Profit'].apply(lambda x: f"{x:,.2f}")
print("Total Sales and Profit by Product Type:")
print(sales_profit_by_item_type)

sales_profit_by_outlet_type = df.groupby('Outlet_Type').
 ↪agg(Total_Sales=('Item_Outlet_Sales', 'sum'),Total_Profit=('Profit', 'sum')).
 ↪reset_index()
sales_profit_by_outlet_type = sales_profit_by_outlet_type.
 ↪sort_values(by='Total_Sales', ascending=False)
sales_profit_by_outlet_type['Total_Sales'] =␣
 ↪sales_profit_by_outlet_type['Total_Sales'].apply(lambda x: f"{x:,.2f}")
sales_profit_by_outlet_type['Total_Profit'] =␣
 ↪sales_profit_by_outlet_type['Total_Profit'].apply(lambda x: f"{x:,.2f}")
print("Total Sales and Profit by Outlet Type:")
print(sales_profit_by_outlet_type)

plt.figure(figsize=(12, 6))
bar_width = 0.35
index = range(len(sales_profit_by_item_type))
plt.bar(index, sales_profit_by_item_type['Total_Sales'].str.replace(',', '').
 ↪astype(float), bar_width, label='Total Sales', color='orange')
plt.bar([i + bar_width for i in index],␣
 ↪sales_profit_by_item_type['Total_Profit'].str.replace(',', '').
 ↪astype(float), bar_width, label='Total Profit', color='lightblue')
plt.title('Total Sales and Profit by Product Type')
plt.xlabel('Product Type')
plt.ylabel('Amount')
plt.xticks([i + bar_width / 2 for i in index],␣
 ↪sales_profit_by_item_type['Item_Type'], rotation=45)
plt.legend()
plt.tight_layout()
plt.show()

plt.figure(figsize=(12, 6))
index = range(len(sales_profit_by_outlet_type))
plt.bar(index, sales_profit_by_outlet_type['Total_Sales'].str.replace(',', '').
 ↪astype(float), bar_width, label='Total Sales', color='orange')
plt.bar([i + bar_width for i in index],␣
 ↪sales_profit_by_outlet_type['Total_Profit'].str.replace(',', '').
 ↪astype(float), bar_width, label='Total Profit', color='lightblue')
plt.title('Total Sales and Profit by Outlet Type')
plt.xlabel('Outlet Type')
plt.ylabel('Amount')
```

```
plt.xticks([i + bar_width / 2 for i in index],␣
  ↪sales_profit_by_outlet_type['Outlet_Type'], rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```

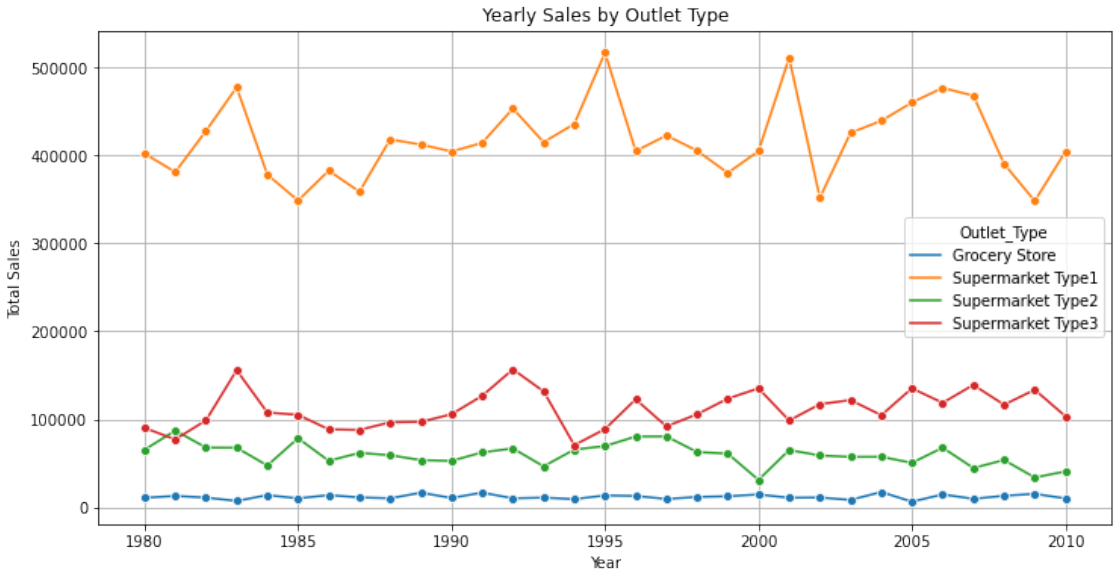
```
Total Sales and Profit by Product Type:
               Item_Type   Total_Sales Total_Profit
6   Fruits and Vegetables  2,820,059.82   564,011.96
13            Snack Foods  2,732,786.09   546,557.22
9               Household  2,055,493.71   411,098.74
5             Frozen Foods  1,825,734.79   365,146.96
4                   Dairy  1,522,594.05   304,518.81
3                 Canned  1,444,151.49   288,830.30
0             Baking Goods  1,265,525.34   253,105.07
8      Health and Hygiene  1,045,200.14   209,040.03
10                   Meat    917,565.61   183,513.12
14            Soft Drinks    892,897.72   178,579.54
1                  Breads    553,237.19   110,647.44
7             Hard Drinks    457,793.43    91,558.69
15          Starchy Foods    351,401.25    70,280.25
11                 Others    325,517.61    65,103.52
2               Breakfast    232,298.95    46,459.79
12                Seafood    148,868.22    29,773.64
Total Sales and Profit by Outlet Type:
         Outlet_Type    Total_Sales  Total_Profit
1  Supermarket Type1  12,917,342.26  2,583,468.45
3  Supermarket Type3   3,453,926.05    690,785.21
2  Supermarket Type2   1,851,822.83    370,364.57
0      Grocery Store     368,034.27     73,606.85
```

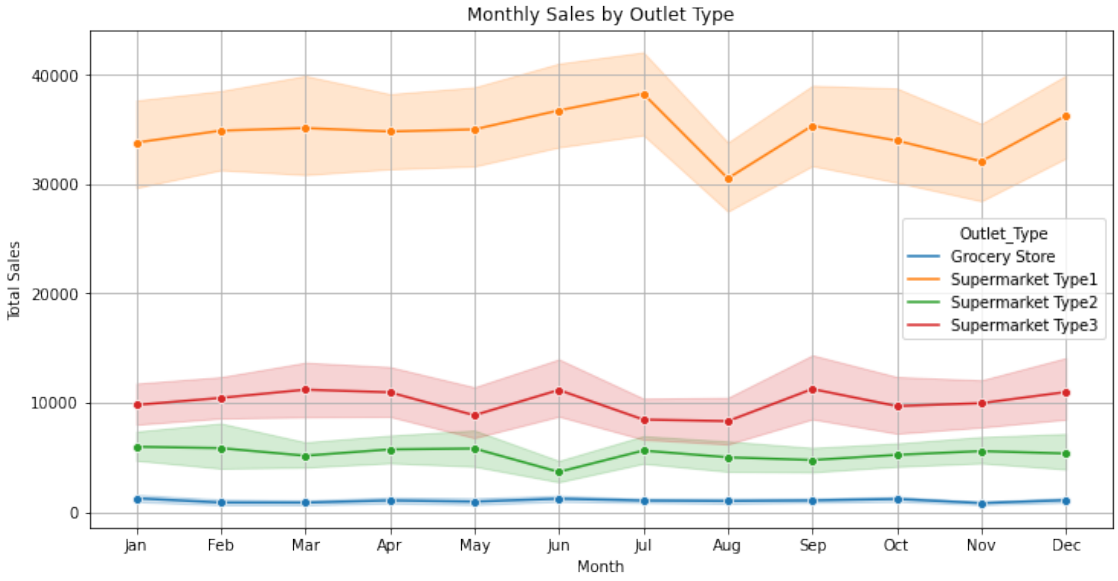




[18]:
```
# yearly and monthly sales by outlet type
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
df['Year'] = np.random.randint(1980, 2011, df.shape[0])
df['Month'] = np.random.randint(1, 13, df.shape[0])
yearly_sales = df.groupby(['Year', 'Outlet_Type'])['Item_Outlet_Sales'].sum().
 ↪reset_index()
monthly_sales = df.groupby(['Year', 'Month',␣
 ↪'Outlet_Type'])['Item_Outlet_Sales'].sum().reset_index()

plt.figure(figsize=(12, 6))
sns.lineplot(data=yearly_sales, x='Year', y='Item_Outlet_Sales',␣
 ↪hue='Outlet_Type', marker='o')
plt.title('Yearly Sales by Outlet Type')
plt.xlabel('Year')
plt.ylabel('Total Sales')
plt.grid()
plt.show()

plt.figure(figsize=(12, 6))
sns.lineplot(data=monthly_sales, x='Month', y='Item_Outlet_Sales',␣
 ↪hue='Outlet_Type', marker='o')
plt.title('Monthly Sales by Outlet Type')
plt.xlabel('Month')
plt.ylabel('Total Sales')
```

```
plt.xticks(ticks=np.arange(1, 13), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May',␣
 ↪'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])
plt.grid()
plt.show()
```



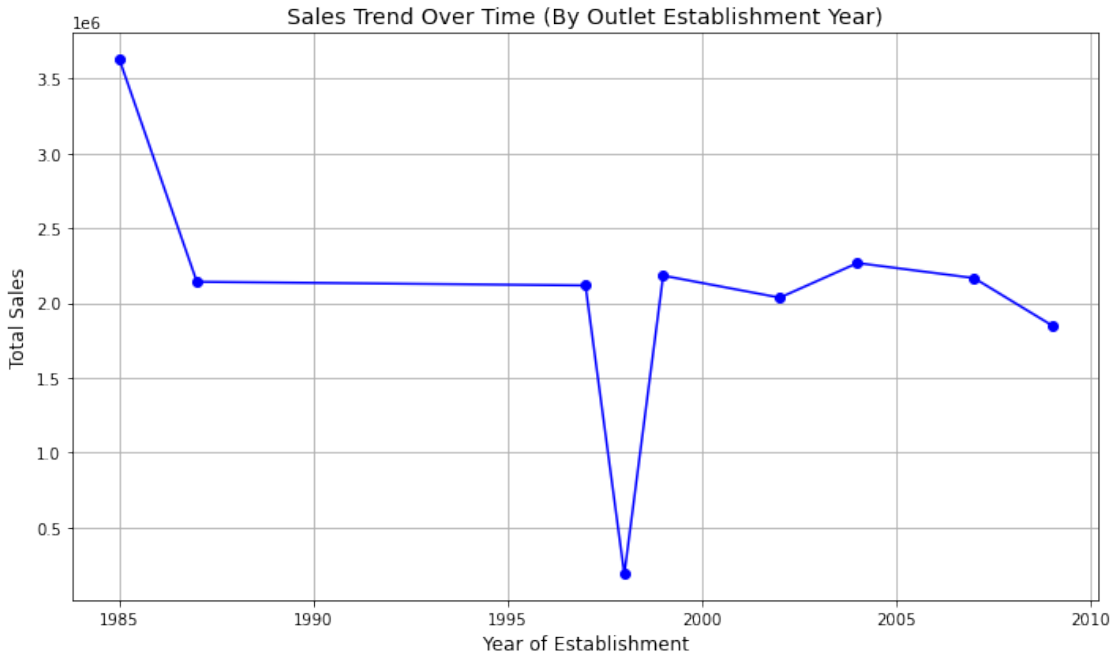


[19]:
```
# total sales by item and outlet type
import plotly.express as px
```

```
sales_by_item_and_outlet = df.groupby(['Item_Type',␣
 ↪'Outlet_Type'])['Item_Outlet_Sales'].sum().reset_index()
fig = px.bar(sales_by_item_and_outlet, x='Item_Type', y='Item_Outlet_Sales',␣
 ↪color='Outlet_Type', text='Item_Outlet_Sales',labels={'Item_Outlet_Sales':␣
 ↪'Total Sales', 'Item_Type': 'Item Type'},title='Total Item Outlet Sales by␣
 ↪Item Type and Outlet Type')

fig.update_traces(texttemplate='%{text:,.0f}', textposition='outside')
fig.update_layout(barmode='group', xaxis_title='Item Type', yaxis_title='Total␣
 ↪Sales')
fig.show()
```

[20]:
```
# total sales by outlet type and fat content
import pandas as pd
import plotly.express as px
sales_by_outlet_fat = df.groupby(['Outlet_Type',␣
 ↪'Item_Fat_Content'])['Item_Outlet_Sales'].sum().reset_index()
fig = px.bar(sales_by_outlet_fat, x='Outlet_Type', y='Item_Outlet_Sales',␣
 ↪color='Item_Fat_Content', labels={'Item_Outlet_Sales': 'Total Sales',␣
 ↪'Outlet_Type': 'Outlet Type'},title='Total Item Outlet Sales by Outlet Type␣
 ↪and Item Fat Content')
fig.update_layout(barmode='group', xaxis_title='Outlet Type',␣
 ↪yaxis_title='Total Sales')
fig.show()
```

[21]:
```
import pandas as pd
import matplotlib.pyplot as plt

# sales trend over time
sales_trend = df.groupby('Outlet_Establishment_Year')['Item_Outlet_Sales'].
 ↪sum().reset_index()
sales_trend = sales_trend.sort_values('Outlet_Establishment_Year')
plt.figure(figsize=(10, 6))
plt.plot(sales_trend['Outlet_Establishment_Year'],␣
 ↪sales_trend['Item_Outlet_Sales'], marker='o', linestyle='-', color='b')
plt.title('Sales Trend Over Time (By Outlet Establishment Year)', fontsize=14)
plt.xlabel('Year of Establishment', fontsize=12)
plt.ylabel('Total Sales', fontsize=12)
plt.grid(True)
plt.tight_layout()
plt.show()
```



[22]:
```
# PREDICTIVE ANALYSIS
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

df['Item_Weight'].fillna(df['Item_Weight'].mean(), inplace=True)

label_encoder = LabelEncoder()
df['Item_Fat_Content'] = label_encoder.fit_transform(df['Item_Fat_Content'])
df['Outlet_Type'] = label_encoder.fit_transform(df['Outlet_Type'])
df['Outlet_Location_Type'] = label_encoder.
 ↪fit_transform(df['Outlet_Location_Type'])
df['Outlet_Size'] = label_encoder.fit_transform(df['Outlet_Size'].astype(str))


X = df[['Item_Weight', 'Item_Fat_Content', 'Item_Visibility', 'Item_MRP',␣
 ↪'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type']]
y = df['Item_Outlet_Sales']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
```

```
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')

# Predicting for a new product
new_data = pd.DataFrame({
    'Item_Weight': [12.5],
    'Item_Fat_Content': [1],  # (0 = Low Fat, 1 = Regular)
    'Item_Visibility': [0.05],
    'Item_MRP': [250.0],
    'Outlet_Size': [1],  # (Small = 0, Medium = 1, Large = 2)
    'Outlet_Location_Type': [2],
    'Outlet_Type': [1]})
future_sales = model.predict(new_data)
print(f'Predicted Future Sales for the product: {future_sales[0]:.2f}')
```

```
Mean Squared Error: 1408350.4888513514
R-squared: 0.49719609306700785
Predicted Future Sales for the product: 4132.76
```

[23]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

df['Item_Visibility'].replace(0, np.nan, inplace=True)  # Replace 0 visibility␣
 ↪(if invalid) with NaN
df['Item_Visibility'].fillna(df['Item_Visibility'].mean(), inplace=True)
visibility_bins = [0, 0.05, 0.10, 0.15, 0.2, df['Item_Visibility'].max()]
visibility_labels = ['Very Low', 'Low', 'Medium', 'High', 'Very High']

df['Visibility_Category'] = pd.cut(df['Item_Visibility'], bins=visibility_bins,␣
 ↪labels=visibility_labels, include_lowest=True)
sales_by_visibility = df.groupby('Visibility_Category')['Item_Outlet_Sales'].
 ↪mean().reset_index()

fig = px.bar(sales_by_visibility, x='Visibility_Category',␣
 ↪y='Item_Outlet_Sales', title="Average Sales by Promotional Visibility",␣
 ↪labels={'Item_Outlet_Sales': 'Average Sales', 'Visibility_Category':␣
 ↪'Visibility␣
 ↪Level'},text='Item_Outlet_Sales',color='Item_Outlet_Sales',color_continuous_scale='Blues')
fig.update_traces(texttemplate='%{text:.2f}', textposition='outside')
fig.update_layout(uniformtext_minsize=8, uniformtext_mode='hide',␣
 ↪xaxis_title="Visibility Category", yaxis_title="Average Sales",␣
 ↪showlegend=False)
```

```
fig.show()
```

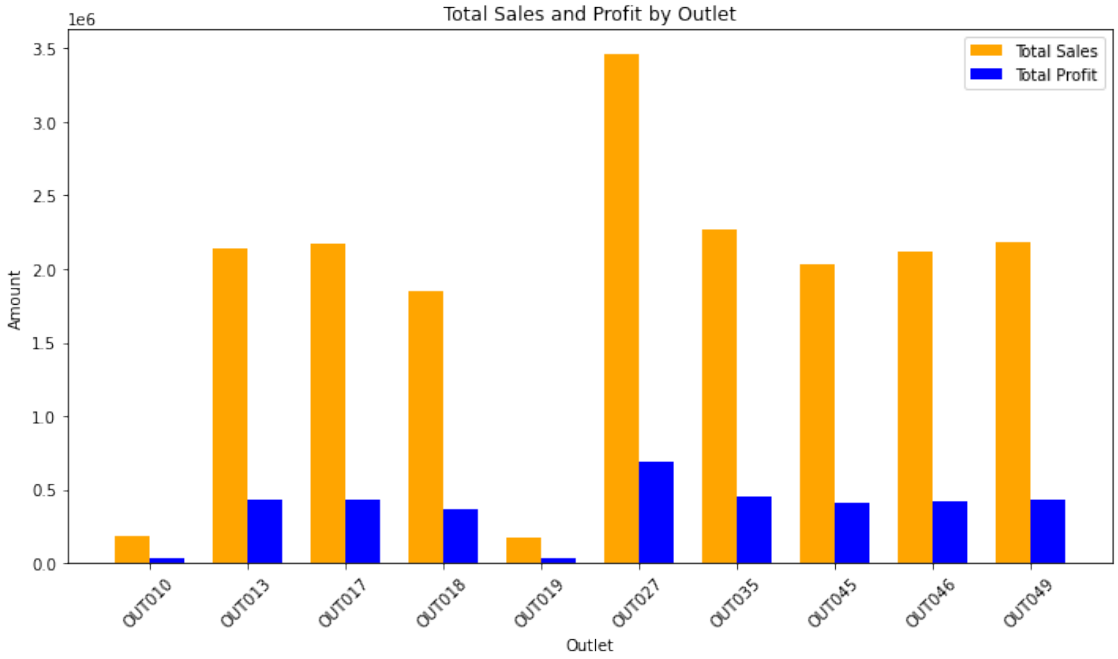
[24]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px

df['Item_Visibility'].replace(0, np.nan, inplace=True)
df['Item_Visibility'].fillna(df['Item_Visibility'].mean(), inplace=True)

visibility_bins = [0, 0.05, 0.1, 0.15, 0.2, df['Item_Visibility'].max()]
visibility_labels = ['Very Low', 'Low', 'Medium', 'High', 'Very High']
df['Visibility_Category'] = pd.cut(df['Item_Visibility'], bins=visibility_bins,␣
 ↪labels=visibility_labels, include_lowest=True)

sales_by_visibility = df.groupby('Visibility_Category')['Item_Outlet_Sales'].
 ↪sum().reset_index()
sales_by_visibility['Item_Outlet_Sales'] =␣
 ↪sales_by_visibility['Item_Outlet_Sales'].apply(lambda x: '{:,.0f}'.format(x))
fig = px.bar(
    sales_by_visibility,
    x='Visibility_Category',
    y=sales_by_visibility['Item_Outlet_Sales'].str.replace(',', '').
 ↪astype(float),
    title="Effect of Product Visibility on Sales",
    labels={'Item_Outlet_Sales': 'Total Sales', 'Visibility_Category':␣
 ↪'Promotional Visibility'},
    text=sales_by_visibility['Item_Outlet_Sales'],
    color='Item_Outlet_Sales',
    color_continuous_scale='Greens')

fig.update_traces(texttemplate='%{text}', textposition='outside')
fig.update_layout(xaxis_title="Visibility Level (Promotion Intensity)",␣
 ↪yaxis_title="Total Sales",showlegend=False)
fig.show()
```

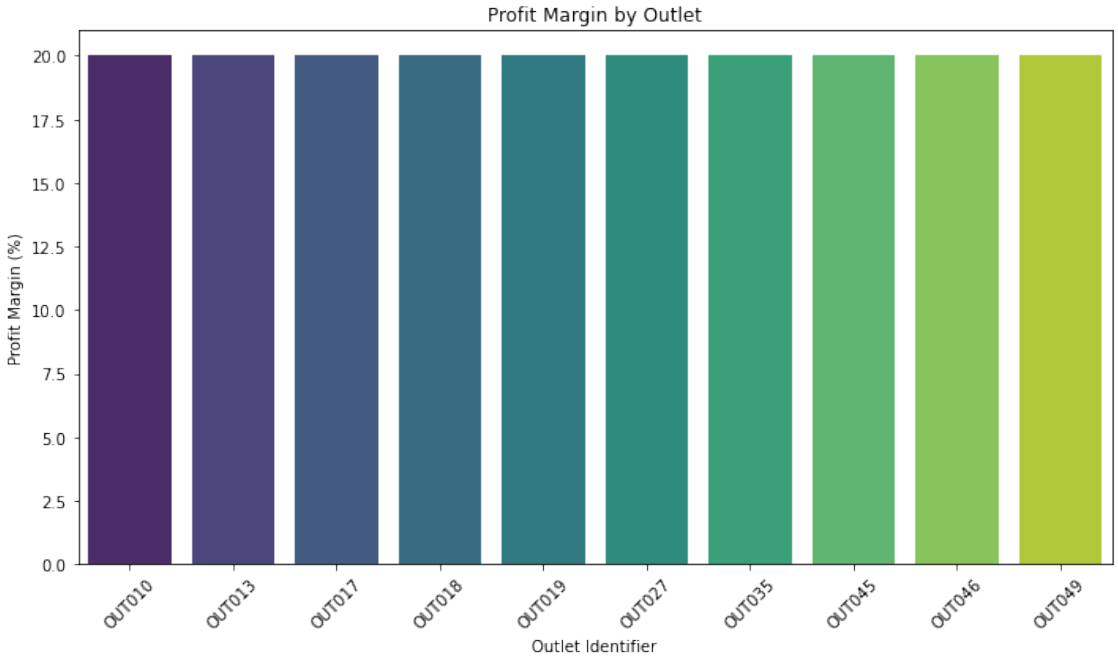
[25]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df['Profit'] = df['Item_Outlet_Sales'] * 0.2

outlet_performance = df.groupby('Outlet_Identifier').
 ↪agg(Total_Sales=('Item_Outlet_Sales', 'sum'),Total_Profit=('Profit', 'sum')).
 ↪reset_index()
```

```
outlet_performance['Profit_Margin'] = (outlet_performance['Total_Profit'] /␣
 ↪outlet_performance['Total_Sales']) * 100
outlet_performance['Total_Sales'] = outlet_performance['Total_Sales'].map('{:,.
 ↪2f}'.format)
outlet_performance['Total_Profit'] = outlet_performance['Total_Profit'].map('{:
 ↪,.2f}'.format)
print("Outlet Performance Analysis (Total Sales, Profit, and Profit Margin):")
print(outlet_performance)

outlet_performance['Total_Sales'] = outlet_performance['Total_Sales'].str.
 ↪replace(',', '').astype(float)
outlet_performance['Total_Profit'] = outlet_performance['Total_Profit'].str.
 ↪replace(',', '').astype(float)
plt.figure(figsize=(10, 6))
bar_width = 0.35
index = range(len(outlet_performance))

plt.bar(index, outlet_performance['Total_Sales'], bar_width, label='Total␣
 ↪Sales', color='orange')
plt.bar([i + bar_width for i in index], outlet_performance['Total_Profit'],␣
 ↪bar_width, label='Total Profit', color='blue')

plt.title('Total Sales and Profit by Outlet')
plt.xlabel('Outlet')
plt.ylabel('Amount')
plt.xticks([i + bar_width / 2 for i in index],␣
 ↪outlet_performance['Outlet_Identifier'], rotation=45)
plt.legend()
plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 6))
sns.barplot(x='Outlet_Identifier', y='Profit_Margin', data=outlet_performance,␣
 ↪palette='viridis')
plt.title('Profit Margin by Outlet')
plt.xlabel('Outlet Identifier')
plt.ylabel('Profit Margin (%)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Outlet Performance Analysis (Total Sales, Profit, and Profit Margin):
  Outlet_Identifier   Total_Sales Total_Profit  Profit_Margin
0            OUT010    188,340.17    37,668.03           20.0
1            OUT013  2,142,663.58   428,532.72           20.0
2            OUT017  2,167,465.29   433,493.06           20.0
3            OUT018  1,851,822.83   370,364.57           20.0
```

4	OUT019	179,694.09	35,938.82	20.0
5	OUT027	3,453,926.05	690,785.21	20.0
6	OUT035	2,268,122.94	453,624.59	20.0
7	OUT045	2,036,725.48	407,345.10	20.0
8	OUT046	2,118,395.17	423,679.03	20.0
9	OUT049	2,183,969.81	436,793.96	20.0





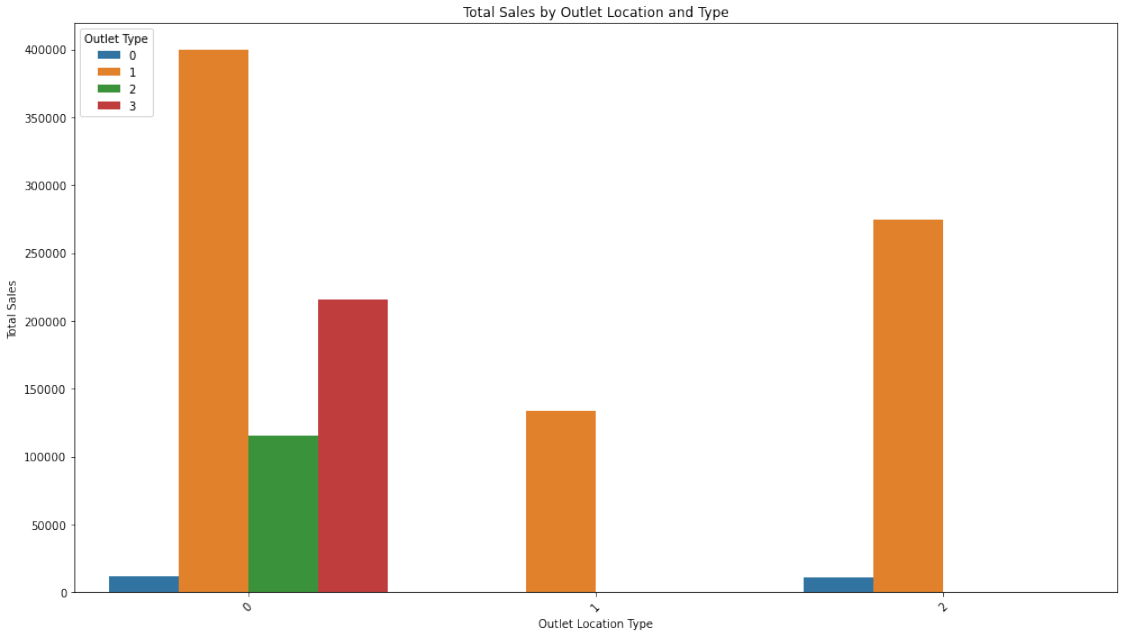
[27]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

sales_by_location_and_type = df.groupby(['Outlet_Location_Type', 'Outlet_Type',␣
 ↪'Item_Type'])['Item_Outlet_Sales'].sum().reset_index()
top_5_products_by_segment = sales_by_location_and_type.
 ↪groupby(['Outlet_Location_Type', 'Outlet_Type']).apply(lambda x: x.
 ↪nlargest(5, 'Item_Outlet_Sales')).reset_index(drop=True)
print("Top 5 selling products in each Outlet_Location_Type and Outlet_Type␣
 ↪combination:")
print(top_5_products_by_segment)
plt.figure(figsize=(14, 8))
sns.barplot(x='Outlet_Location_Type', y='Item_Outlet_Sales', hue='Outlet_Type',␣
 ↪data=sales_by_location_and_type, ci=None)
plt.title('Total Sales by Outlet Location and Type')
plt.xlabel('Outlet Location Type')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.legend(title='Outlet Type')
plt.tight_layout()
plt.show()

plt.figure(figsize=(16, 8))
sns.barplot(x='Item_Type', y='Item_Outlet_Sales', hue='Outlet_Type',␣
 ↪data=top_5_products_by_segment, ci=None)
plt.title('Top 5 Selling Products by Outlet Location and Type')
plt.xlabel('Product Type')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.legend(title='Outlet Type')
plt.tight_layout()
plt.show()
```

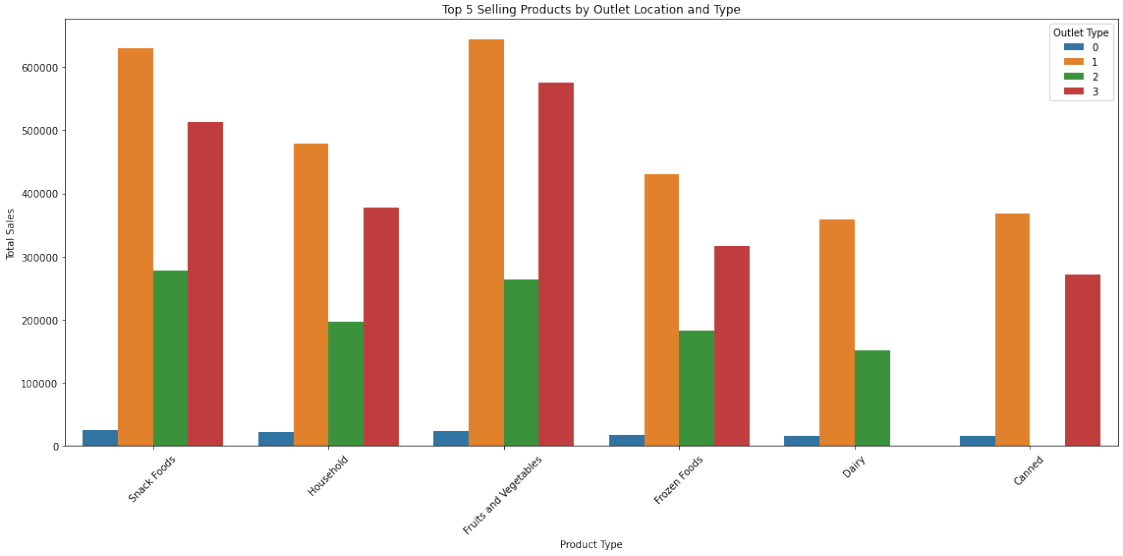
```
Top 5 selling products in each Outlet_Location_Type and Outlet_Type combination:
    Outlet_Location_Type  Outlet_Type              Item_Type  \
0                      0            0            Snack Foods
1                      0            0              Household
2                      0            0  Fruits and Vegetables
3                      0            0           Frozen Foods
4                      0            0                  Dairy
5                      0            1  Fruits and Vegetables
6                      0            1            Snack Foods
7                      0            1              Household
8                      0            1           Frozen Foods
```

9	0	1	Dairy
10	0	2	Snack Foods
11	0	2	Fruits and Vegetables
12	0	2	Household
13	0	2	Frozen Foods
14	0	2	Dairy
15	0	3	Fruits and Vegetables
16	0	3	Snack Foods
17	0	3	Household
18	0	3	Frozen Foods
19	0	3	Canned
20	1	1	Fruits and Vegetables
21	1	1	Snack Foods
22	1	1	Household
23	1	1	Frozen Foods
24	1	1	Dairy
25	2	0	Snack Foods
26	2	0	Fruits and Vegetables
27	2	0	Household
28	2	0	Canned
29	2	0	Dairy
30	2	1	Snack Foods
31	2	1	Fruits and Vegetables
32	2	1	Household
33	2	1	Frozen Foods
34	2	1	Canned

	Item_Outlet_Sales
0	25942.8970
1	25550.0750
2	24548.0460
3	17942.6442
4	15307.4078
5	981032.3312
6	949753.7130
7	699800.4086
8	632247.0090
9	520286.0810
10	278714.5328
11	263471.7076
12	196267.1872
13	183599.0106
14	152130.6394
15	576028.1886
16	513088.1172
17	378299.5704
18	316272.3108
19	272150.4106

20	341526.7706
21	309246.1234
22	248046.4532
23	203696.8494
24	196254.5370
25	25653.2740
26	24054.0224
27	18157.0318
28	16739.5436
29	16144.3184
30	630387.4296
31	609398.7504
32	489372.9870
33	456724.1524
34	368528.9554




Top 5 Selling Products by Outlet Location and Type
Outlet Type
0
1
2
3
Total Sales
0
100000
200000
300000
400000
500000
600000
Snack Foods
Household
Fruits and Vegetables
Frozen Foods
Dairy
Canned
Product Type

[ ]: