In [ ]:
```
NAME: MANE SHIVRAJ PANDURANG
COURSE: CL I
CLASS: BE AI&DS.
```

In [1]:
```
# Step 1 - Load the Sales dataset in CSV, JSON and Excel Format.
```

In [2]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
csv_data = pd.read_csv(r"C:\Users\saira\Downloads\supermarket_sales - Sheet1.csv")
excel_data = pd.read_excel(r"C:\Users\saira\Downloads\supermarket_sales-Sheet1.xlsx
json_data = pd.read_json(r"C:\Users\saira\Downloads\myData (1).json")
```

In [4]:
```
csv_data.head()
excel_data.head()
json_data.head()
```

Out[4]:

	Invoice_ID	Branch	City	Customer_type	Gender	Product_line	Unit_price	Quanti
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	

In [5]:
```
csv_data.tail()
excel_data.tail()
json_data.tail()
```

Out[5]:

	Invoice_ID	Branch	City	Customer_type	Gender	Product_line	Unit_price	Qua
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	

In [6]:
```
# Step 2 - Explore the Structure and Content.
```

In [7]:
```
csv_data.describe()
excel_data.describe()
json_data.describe()
```

Out[7]:

	Unit_price	Quantity	Tax_5%	Total	Date	cogs	gr
count	1000.000000	1000.000000	1000.000000	1000.000000	1000	1000.00000	
mean	55.672130	5.510000	15.379369	322.966749	2019-02-14 00:05:45.600000	307.58738	
min	10.080000	1.000000	0.508500	10.678500	2019-01-01 00:00:00	10.17000	
25%	32.875000	3.000000	5.924875	124.422375	2019-01-24 00:00:00	118.49750	
50%	55.230000	5.000000	12.088000	253.848000	2019-02-13 00:00:00	241.76000	
75%	77.935000	8.000000	22.445250	471.350250	2019-03-08 00:00:00	448.90500	
max	99.960000	10.000000	49.650000	1042.650000	2019-03-30 00:00:00	993.00000	
std	26.494628	2.923431	11.708825	245.885335	NaN	234.17651	

In [8]:
```
csv_data.info()
excel_data.info()
json_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Invoice ID               1000 non-null   object 
 1   Branch                   1000 non-null   object 
 2   City                     1000 non-null   object 
 3   Customer type            1000 non-null   object 
 4   Gender                   1000 non-null   object 
 5   Product line             1000 non-null   object 
 6   Unit price               1000 non-null   float64
 7   Quantity                 1000 non-null   int64  
 8   Tax 5%                   1000 non-null   float64
 9   Total                    1000 non-null   float64
 10  Date                     1000 non-null   object 
 11  Time                     1000 non-null   object 
 12  Payment                  1000 non-null   object 
 13  cogs                     1000 non-null   float64
 14  gross margin percentage  1000 non-null   float64
 15  gross income             1000 non-null   float64
 16  Rating                   1000 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Invoice ID               1000 non-null   object 
 1   Branch                   1000 non-null   object 
 2   City                     1000 non-null   object 
 3   Customer type            1000 non-null   object 
 4   Gender                   1000 non-null   object 
 5   Product line             1000 non-null   object 
 6   Unit price               1000 non-null   float64
 7   Quantity                 1000 non-null   int64  
 8   Tax 5%                   1000 non-null   float64
 9   Total                    1000 non-null   float64
 10  Date                     1000 non-null   object 
 11  Time                     1000 non-null   object 
 12  Payment                  1000 non-null   object 
 13  cogs                     1000 non-null   float64
 14  gross margin percentage  1000 non-null   float64
 15  gross income             1000 non-null   float64
 16  Rating                   1000 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Invoice_ID               1000 non-null   object 
 1   Branch                   1000 non-null   object 
 2   City                     1000 non-null   object 
```

```
 3   Customer_type            1000 non-null   object
 4   Gender                   1000 non-null   object
 5   Product_line             1000 non-null   object
 6   Unit_price               1000 non-null   float64
 7   Quantity                 1000 non-null   int64
 8   Tax_5%                   1000 non-null   float64
 9   Total                    1000 non-null   float64
 10  Date                     1000 non-null   datetime64[ns]
 11  Time                     1000 non-null   object
 12  Payment                  1000 non-null   object
 13  cogs                     1000 non-null   float64
 14  gross_margin_percentage  1000 non-null   float64
 15  gross_income             1000 non-null   float64
 16  Rating                   1000 non-null   float64
dtypes: datetime64[ns](1), float64(7), int64(1), object(8)
memory usage: 132.9+ KB
```

In [9]:
```
csv_data.isnull().sum()
excel_data.isnull().sum()
json_data.isnull().sum()
```

Out[9]:
```
Invoice_ID                0
Branch                    0
City                      0
Customer_type             0
Gender                    0
Product_line              0
Unit_price                0
Quantity                  0
Tax_5%                    0
Total                     0
Date                      0
Time                      0
Payment                   0
cogs                      0
gross_margin_percentage   0
gross_income              0
Rating                    0
dtype: int64
```

In [11]:
```
csv_data.duplicated().sum()
excel_data.duplicated().sum()
json_data.duplicated().sum()
```

Out[11]:
```
0
```

In [12]:
```
# Step 4 - Convert the Data into a Unified Format.
```

In [21]:
```
csv_data.columns = [col.lower() for col in csv_data.columns]
csv_data['date'] = pd.to_datetime(csv_data['date'])
csv_data['unit price'] = csv_data['unit price'].astype(float)
csv_data['quantity'] = csv_data['quantity'].astype(int)
csv_data['total'] = csv_data['total'].astype(float)

excel_data.columns = [col.lower() for col in excel_data.columns]
excel_data['date'] = pd.to_datetime(excel_data['date'])
```

```
excel_data['unit price'] = excel_data['unit price'].astype(float)
excel_data['quantity'] = excel_data['quantity'].astype(int)
excel_data['total'] = excel_data['total'].astype(float)

json_data.columns = [col.lower() for col in json_data.columns]
json_data['date'] = pd.to_datetime(json_data['date'])
json_data['unit price'] = json_data['unit_price'].astype(float)
json_data['quantity'] = json_data['quantity'].astype(int)
json_data['total'] = json_data['total'].astype(float)
```

In [20]:
```
print(json_data.columns)
```

```
Index(['invoice_id', 'branch', 'city', 'customer_type', 'gender',
       'product_line', 'unit_price', 'quantity', 'tax_5%', 'total', 'date',
       'time', 'payment', 'cogs', 'gross_margin_percentage', 'gross_income',
       'rating'],
      dtype='object')
```

In [22]:
```
csv_data.info()
excel_data.info()
json_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   invoice id               1000 non-null   object
 1   branch                   1000 non-null   object
 2   city                     1000 non-null   object
 3   customer type            1000 non-null   object
 4   gender                   1000 non-null   object
 5   product line             1000 non-null   object
 6   unit price               1000 non-null   float64
 7   quantity                 1000 non-null   int32
 8   tax 5%                   1000 non-null   float64
 9   total                    1000 non-null   float64
 10  date                     1000 non-null   datetime64[ns]
 11  time                     1000 non-null   object
 12  payment                  1000 non-null   object
 13  cogs                     1000 non-null   float64
 14  gross margin percentage  1000 non-null   float64
 15  gross income             1000 non-null   float64
 16  rating                   1000 non-null   float64
dtypes: datetime64[ns](1), float64(7), int32(1), object(8)
memory usage: 129.0+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   invoice id               1000 non-null   object
 1   branch                   1000 non-null   object
 2   city                     1000 non-null   object
 3   customer type            1000 non-null   object
 4   gender                   1000 non-null   object
 5   product line             1000 non-null   object
 6   unit price               1000 non-null   float64
 7   quantity                 1000 non-null   int32
 8   tax 5%                   1000 non-null   float64
 9   total                    1000 non-null   float64
 10  date                     1000 non-null   datetime64[ns]
 11  time                     1000 non-null   object
 12  payment                  1000 non-null   object
 13  cogs                     1000 non-null   float64
 14  gross margin percentage  1000 non-null   float64
 15  gross income             1000 non-null   float64
 16  rating                   1000 non-null   float64
dtypes: datetime64[ns](1), float64(7), int32(1), object(8)
memory usage: 129.0+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   invoice_id               1000 non-null   object
 1   branch                   1000 non-null   object
 2   city                     1000 non-null   object
```

```
 3   customer_type            1000 non-null   object
 4   gender                   1000 non-null   object
 5   product_line             1000 non-null   object
 6   unit_price               1000 non-null   float64
 7   quantity                 1000 non-null   int32
 8   tax_5%                   1000 non-null   float64
 9   total                    1000 non-null   float64
 10  date                     1000 non-null   datetime64[ns]
 11  time                     1000 non-null   object
 12  payment                  1000 non-null   object
 13  cogs                     1000 non-null   float64
 14  gross_margin_percentage  1000 non-null   float64
 15  gross_income             1000 non-null   float64
 16  rating                   1000 non-null   float64
 17  unit price               1000 non-null   float64
dtypes: datetime64[ns](1), float64(8), int32(1), object(8)
memory usage: 136.8+ KB
```

In [37]:
```
csv_total_sales = csv_data['total sales'].sum()
excel_total_sales = excel_data['total sales'].sum()
json_total_sales = json_data['total sales'].sum()
```

In [46]:
```
csv_total_sales
excel_total_sales
json_total_sales
```

Out[46]:
```
307587.38
```

In [48]:
```
csv_average_order_value = csv_data['total sales'].mean()
excel_average_order_value = excel_data['total sales'].mean()
json_average_order_value = json_data['total sales'].mean()
```

In [49]:
```
csv_average_order_value
excel_average_order_value
json_average_order_value
```

Out[49]:
```
307.58738
```

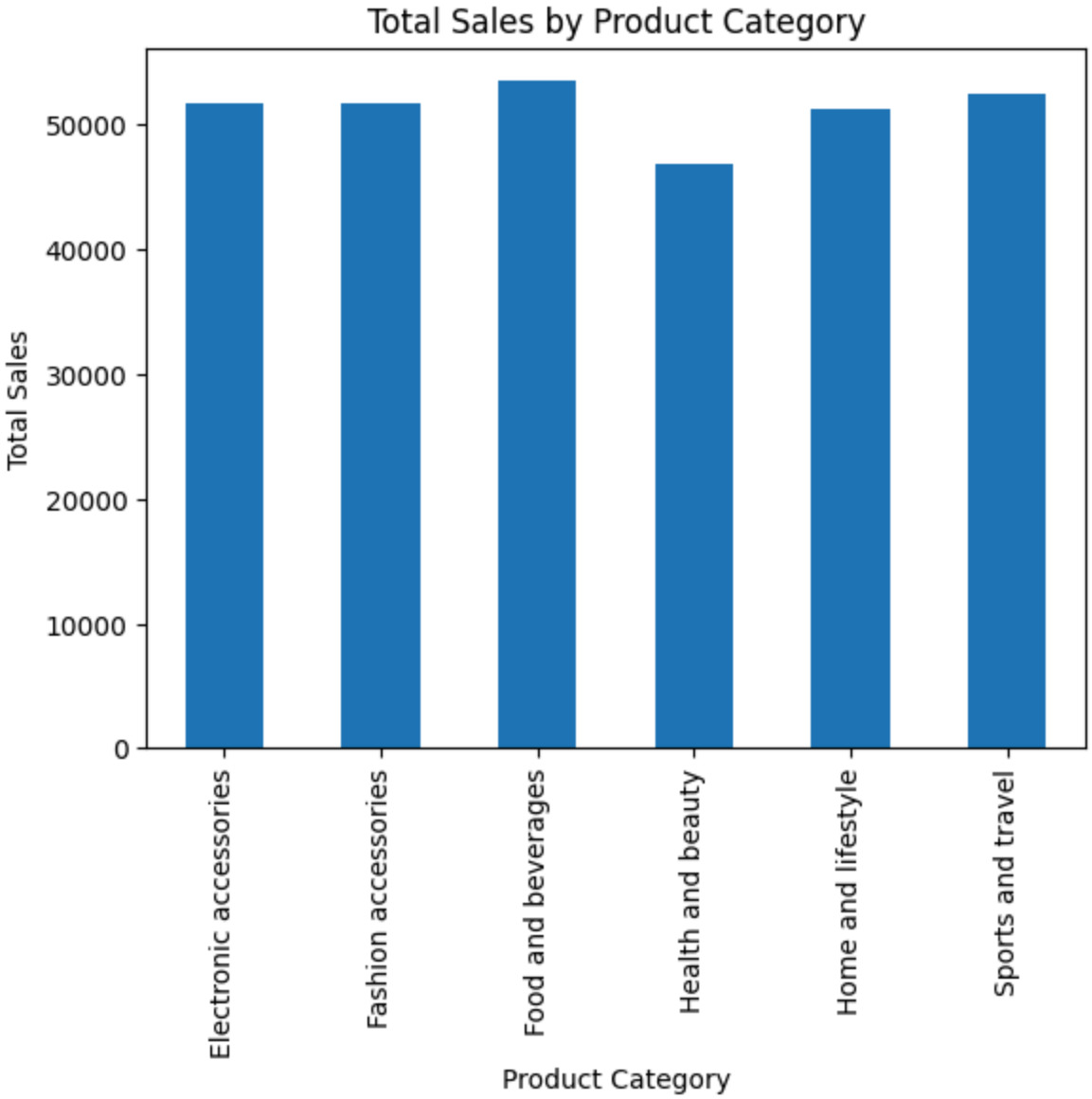
In [53]:
```
csv_category_sales = csv_data.groupby('product line')['total sales'].sum()
excel_category_sales = excel_data.groupby('product line')['total sales'].sum()
json_category_sales = json_data.groupby('product_line')['total sales'].sum()
```

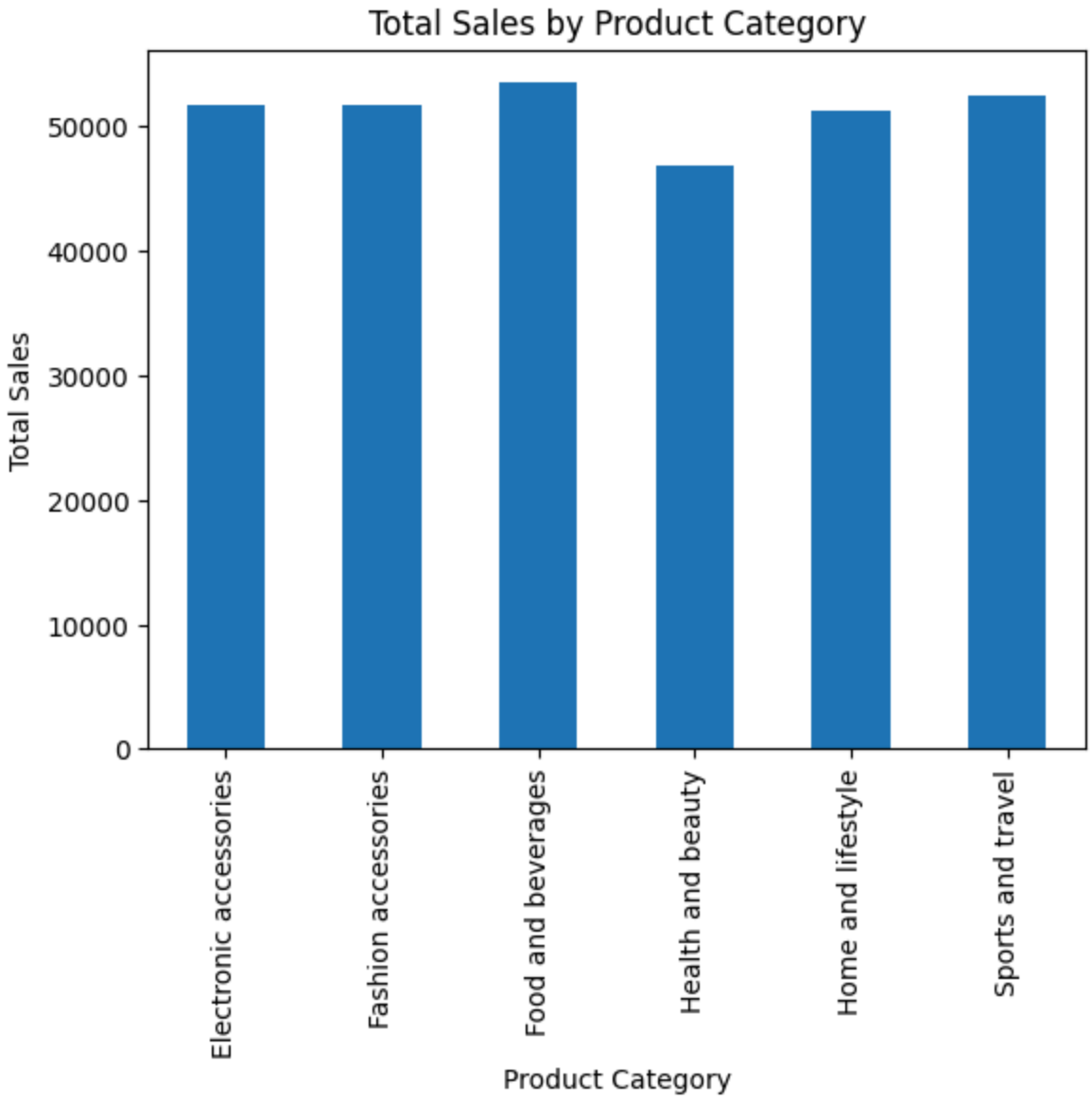
In [54]:
```
csv_category_sales
excel_category_sales
json_category_sales
```

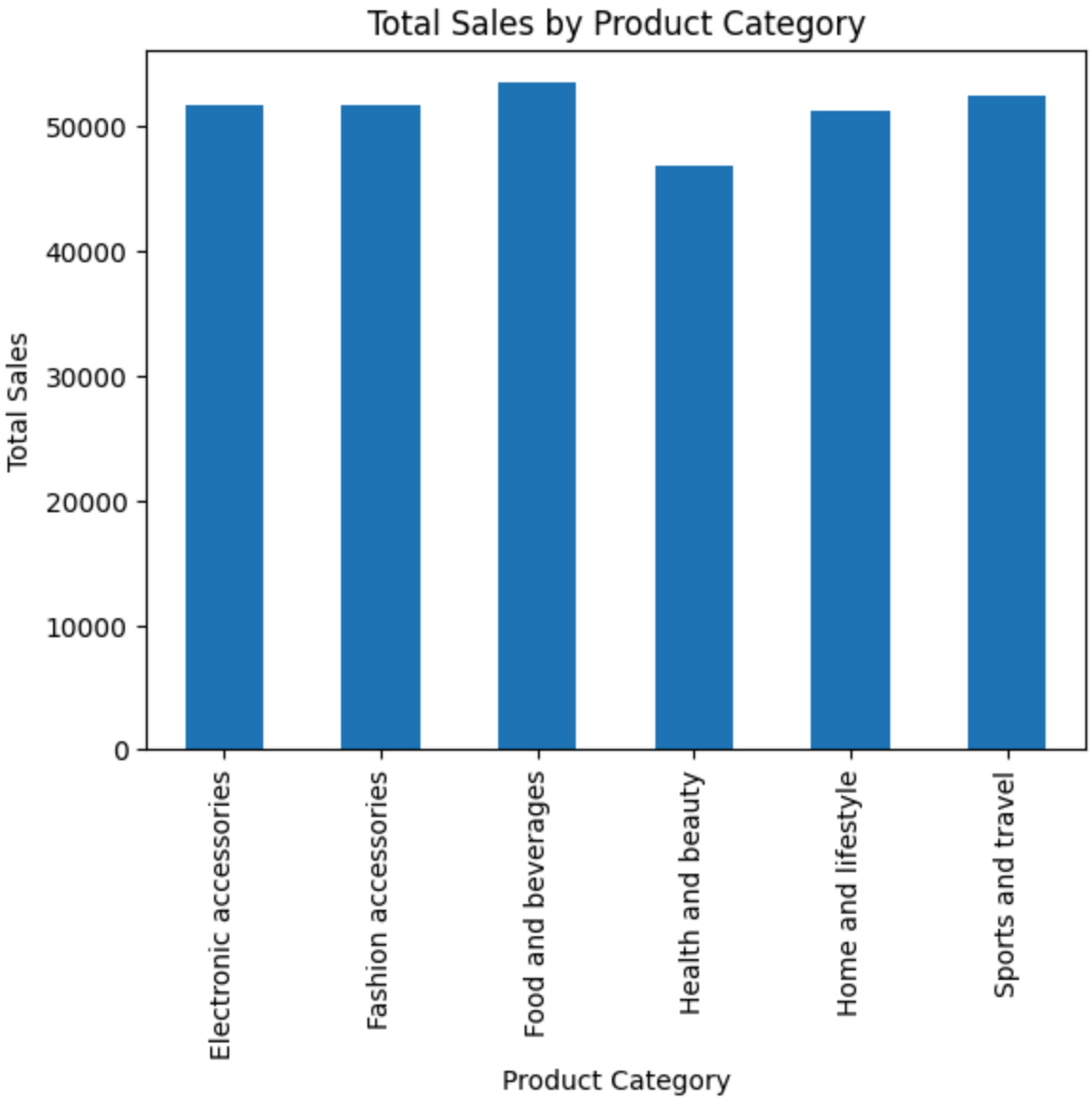
Out[54]:
```
product_line
Electronic accessories    51750.03
Fashion accessories       51719.90
Food and beverages        53471.28
Health and beauty         46851.18
Home and lifestyle        51297.06
Sports and travel         52497.93
Name: total sales, dtype: float64
```

In [55]:
```
# Step 6 -Create Visualizations.
```

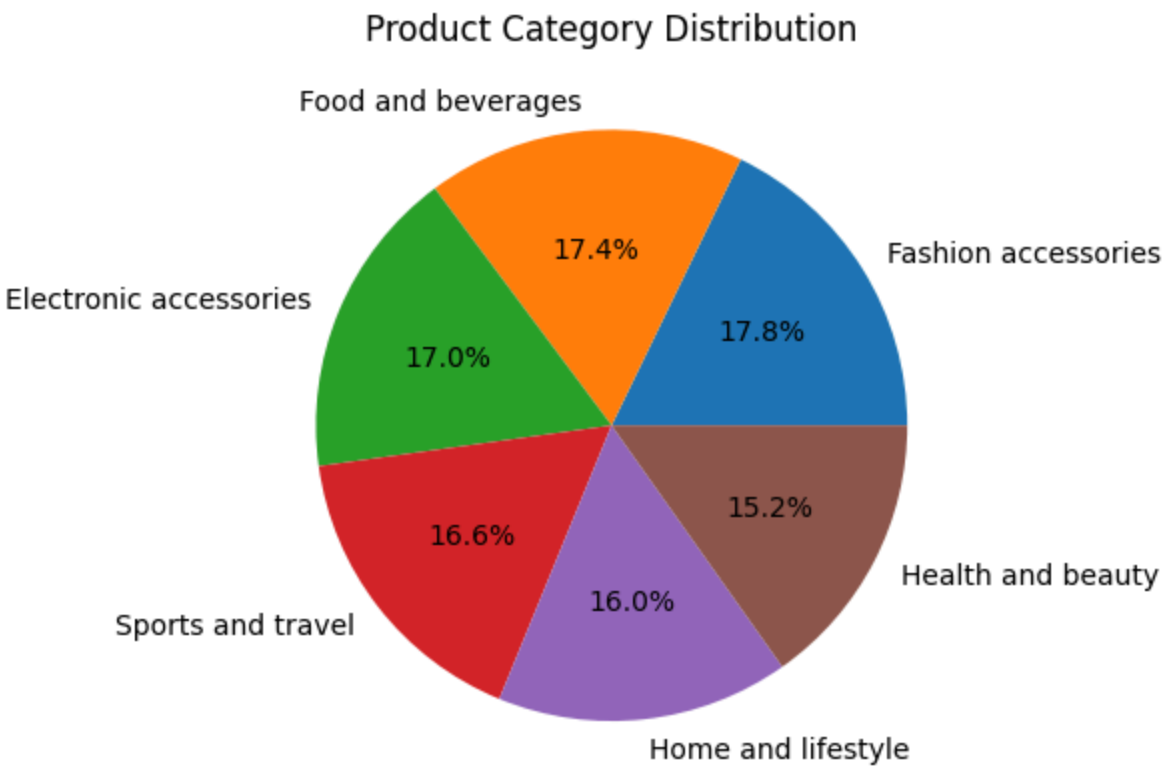
In [58]:
```
import matplotlib.pyplot as plt

csv_category_sales.plot(kind='bar', title='Total Sales by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Total Sales')
plt.show()

excel_category_sales.plot(kind='bar', title='Total Sales by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Total Sales')
plt.show()

json_category_sales.plot(kind='bar', title='Total Sales by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Total Sales')
plt.show()
```

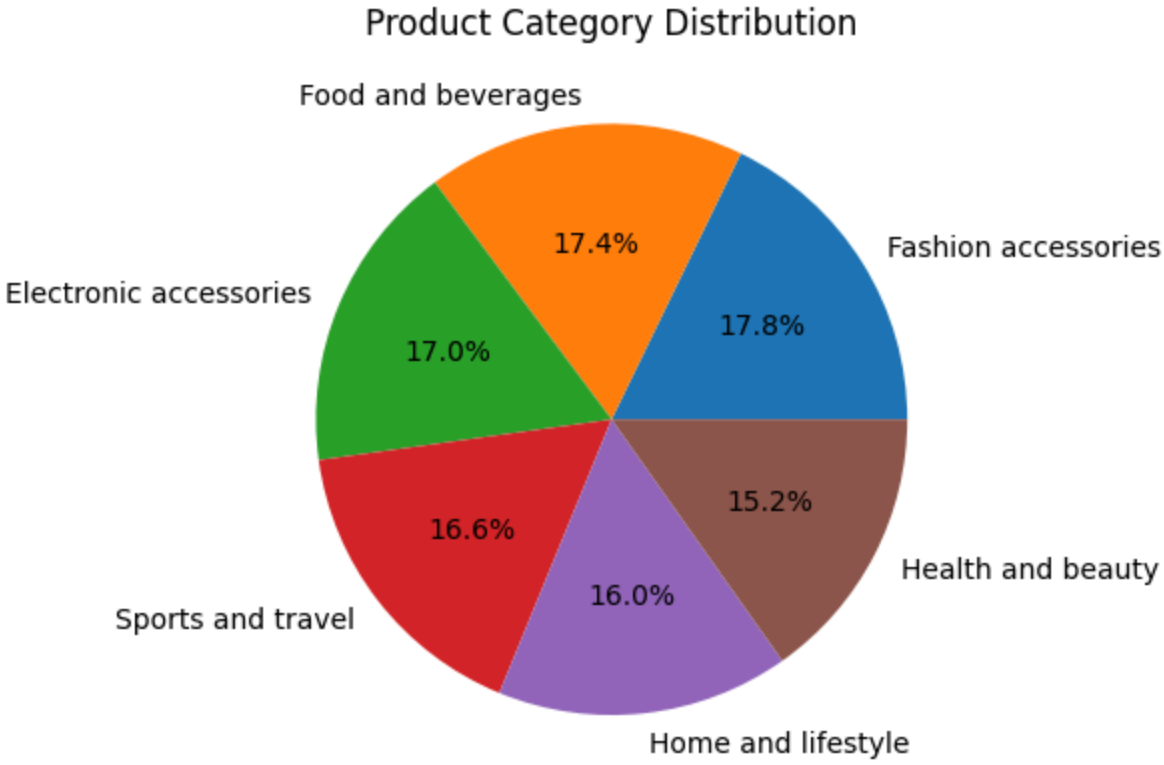


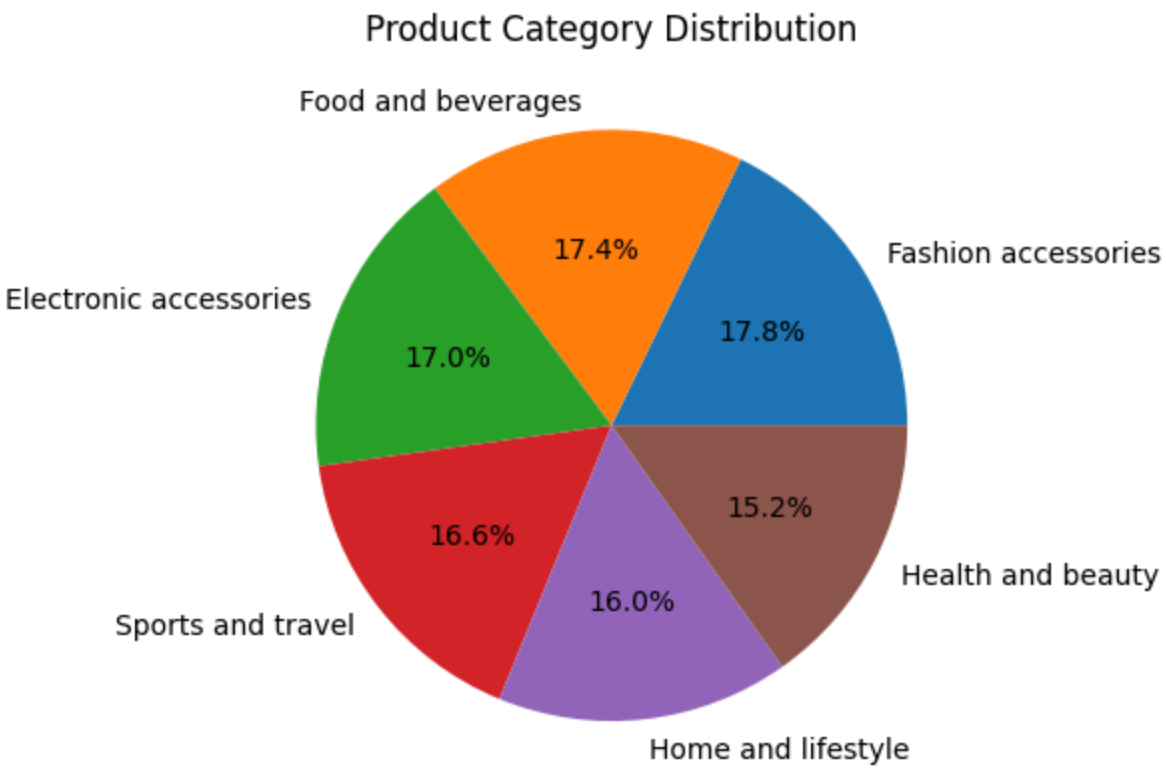

Total Sales by Product Category
Total Sales
0
10000
20000
30000
40000
50000
Electronic accessories
Fashion accessories
Food and beverages
Health and beauty
Home and lifestyle
Sports and travel
Product Category



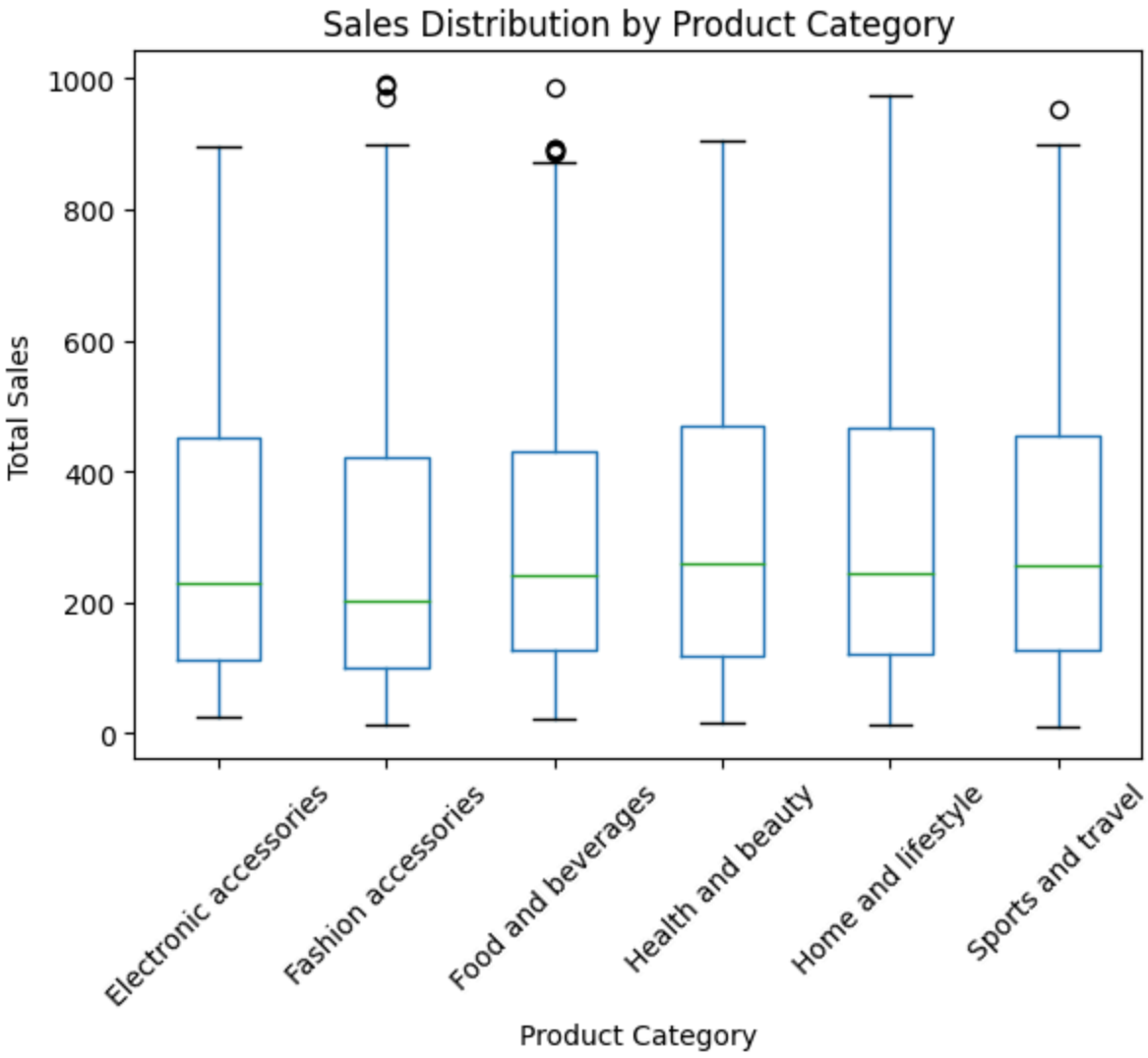
In [61]:
```
category_distribution = csv_data['product line'].value_counts()
category_distribution.plot(kind='pie', title='Product Category Distribution', autop
plt.ylabel('')
plt.show()

category_distribution = excel_data['product line'].value_counts()
category_distribution.plot(kind='pie', title='Product Category Distribution', autop
plt.ylabel('')
plt.show()

category_distribution = json_data['product_line'].value_counts()
category_distribution.plot(kind='pie', title='Product Category Distribution', autop
plt.ylabel('')
plt.show()
```

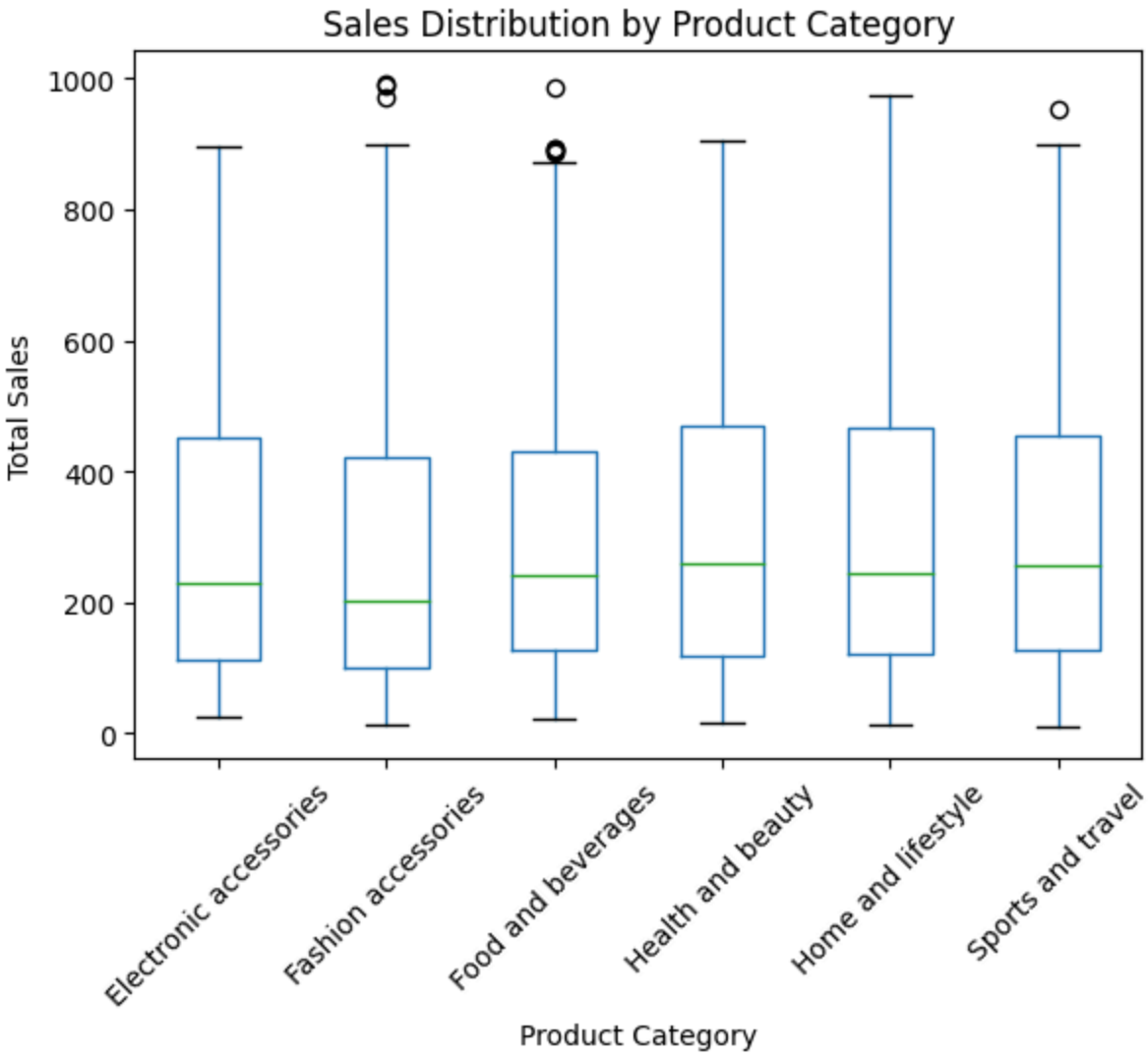

Product Category Distribution
Food and beverages
Fashion accessories
Electronic accessories
17.4%
17.8%
17.0%
15.2%
16.6%
Health and beauty
16.0%
Sports and travel
Home and lifestyle

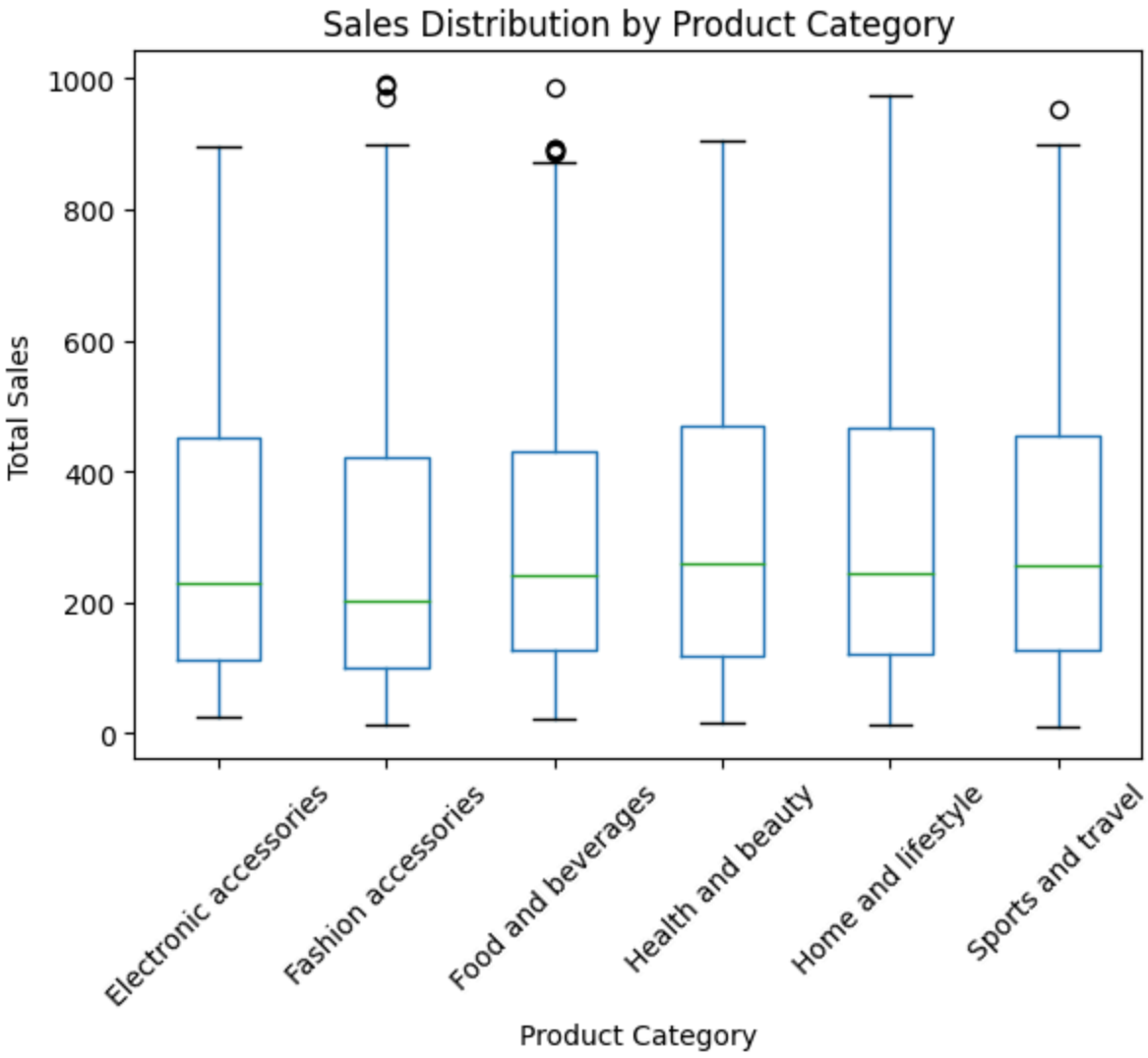

Product Category Distribution
Food and beverages
Fashion accessories
Electronic accessories
17.4%
17.8%
17.0%
15.2%
16.6%
Health and beauty
16.0%
Sports and travel
Home and lifestyle



In [64]:
```
csv_data.boxplot(column='total sales', by='product line', grid=False, rot=45)
plt.title('Sales Distribution by Product Category')
plt.suptitle('')
plt.xlabel('Product Category')
plt.ylabel('Total Sales')
plt.show()

excel_data.boxplot(column='total sales', by='product line', grid=False, rot=45)
plt.title('Sales Distribution by Product Category')
plt.suptitle('')
plt.xlabel('Product Category')
plt.ylabel('Total Sales')
plt.show()

json_data.boxplot(column='total sales', by='product_line', grid=False, rot=45)
plt.title('Sales Distribution by Product Category')
plt.suptitle('')
plt.xlabel('Product Category')
plt.ylabel('Total Sales')
plt.show()
```


Sales Distribution by Product Category
Total Sales
1000
800
600
400
200
0
Electronic accessories
Fashion accessories
Food and beverages
Health and beauty
Home and lifestyle
Sports and travel
Product Category


Sales Distribution by Product Category
Total Sales
1000
800
600
400
200
0
Electronic accessories
Fashion accessories
Food and beverages
Health and beauty
Home and lifestyle
Sports and travel
Product Category


Sales Distribution by Product Category
Total Sales
1000
800
600
400
200
0
Electronic accessories
Fashion accessories
Food and beverages
Health and beauty
Home and lifestyle
Sports and travel
Product Category

In [ ]: