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

In [2]:
```
csv_data = pd.read_csv("format1.csv")
excel_data = pd.read_excel("format2.xlsx")
json_data = pd.read_json("format3.json")
```

In [3]:
```
csv_data.head()
```

Out[3]:

	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gro incor
0	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	4.761905	26.14
1	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	4.761905	3.82
2	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	4.761905	16.21
3	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761905	23.28
4	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	4.761905	30.20

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

Out[4]:

	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gro incor
0	A	Yangon	Normal	Male	Electronic accessories	51.69	7	18.0915	379.9215	1/26/2019	18:22	Cash	361.83	4.761905	18.09
1	B	Mandalay	Member	Female	Fashion accessories	54.73	7	19.1555	402.2655	3/14/2019	19:02	Credit card	383.11	4.761905	19.15
2	B	Mandalay	Member	Male	Home and lifestyle	27.00	9	12.1500	255.1500	3/2/2019	14:16	Cash	243.00	4.761905	12.15
3	C	Naypyitaw	Normal	Female	Electronic accessories	30.24	1	1.5120	31.7520	3/4/2019	15:44	Cash	30.24	4.761905	1.51
4	B	Mandalay	Member	Female	Food and beverages	89.14	4	17.8280	374.3880	1/7/2019	12:20	Credit card	356.56	4.761905	17.82

In [5]:
```
json_data.head()
```

Out[5]:

	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income
701	B	Mandalay	Normal	Male	Food and beverages	32.32	3	4.8480	101.8080	2019-03-27	19:11	Credit card	96.96	4.761905	4.848(
702	B	Mandalay	Member	Female	Fashion accessories	19.77	10	9.8850	207.5850	2019-02-27	18:57	Credit card	197.70	4.761905	9.885(
703	B	Mandalay	Member	Male	Health and beauty	80.47	9	36.2115	760.4415	2019-01-06	11:18	Cash	724.23	4.761905	36.211
704	B	Mandalay	Member	Female	Home and lifestyle	88.39	9	39.7755	835.2855	2019-03-02	12:40	Cash	795.51	4.761905	39.775
705	B	Mandalay	Normal	Male	Health and beauty	71.77	7	25.1195	527.5095	2019-03-29	14:06	Cash	502.39	4.761905	25.119

In [6]:
```
def merge(dataframes):
    if dataframes:
        return pd.concat(dataframes)
```

In [7]:
```
df = merge([csv_data, excel_data, json_data])
```

In [8]:
```
df.head()
```

Out[8]:

	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gro inco
0	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	4.761905	26.14
1	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	4.761905	3.82
2	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	4.761905	16.21
3	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761905	23.28
4	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	4.761905	30.20

In [9]:
```
df.describe()
```

Out[9]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1000.000000	1000.000000	1000.00000
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.761905	15.379369	6.97270
std	26.494628	2.923431	11.708825	245.885335	234.17651	0.000000	11.708825	1.71858
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.761905	0.508500	4.00000
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.761905	5.924875	5.50000
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.761905	12.088000	7.00000
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.761905	22.445250	8.50000
max	99.960000	10.000000	49.650000	1042.650000	993.00000	4.761905	49.650000	10.00000

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 0 to 999
Data columns (total 16 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Branch                   1000 non-null   object
 1   City                     1000 non-null   object
 2   Customer type            1000 non-null   object
 3   Gender                   1000 non-null   object
 4   Product line             1000 non-null   object
 5   Unit price               1000 non-null   float64
 6   Quantity                 1000 non-null   int64
 7   Tax 5%                   1000 non-null   float64
 8   Total                    1000 non-null   float64
 9   Date                     1000 non-null   object
 10  Time                     1000 non-null   object
 11  Payment                  1000 non-null   object
 12  cogs                     1000 non-null   float64
 13  gross margin percentage  1000 non-null   float64
 14  gross income             1000 non-null   float64
 15  Rating                   1000 non-null   float64
dtypes: float64(7), int64(1), object(8)
memory usage: 132.8+ KB
```

In [11]:
```
df.Date = pd.to_datetime(df.Date)
df.Time = pd.to_datetime(df.Time)
```

```
In [12]: df.dtypes
```

```
Out[12]: Branch                          object
         City                            object
         Customer type                   object
         Gender                          object
         Product line                    object
         Unit price                     float64
         Quantity                         int64
         Tax 5%                         float64
         Total                          float64
         Date                    datetime64[ns]
         Time                    datetime64[ns]
         Payment                         object
         cogs                           float64
         gross margin percentage        float64
         gross income                   float64
         Rating                         float64
         dtype: object
```

```
In [13]: df.isna().sum()
```

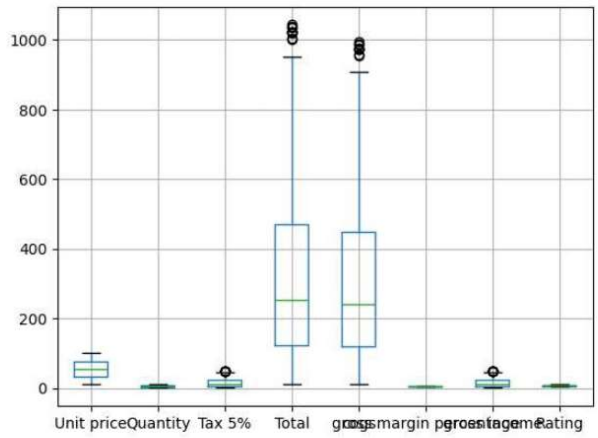
```
Out[13]: Branch                    0
         City                      0
         Customer type             0
         Gender                    0
         Product line              0
         Unit price                0
         Quantity                  0
         Tax 5%                    0
         Total                     0
         Date                      0
         Time                      0
         Payment                   0
         cogs                      0
         gross margin percentage   0
         gross income              0
         Rating                    0
         dtype: int64
```

```
In [14]: df.duplicated().sum()
```

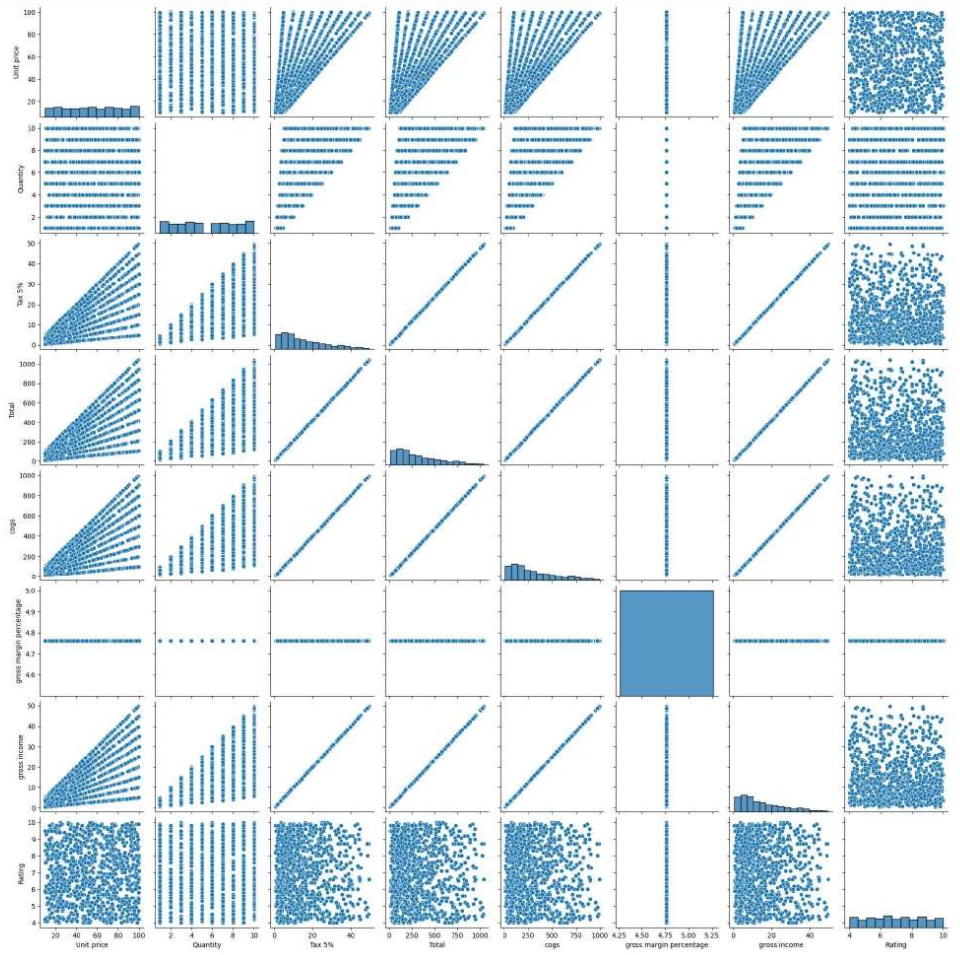
```
Out[14]: 0
```

```
In [15]: df.boxplot()
```

```
Out[15]: <Axes: >
```



```
In [16]: sns.pairplot(df)
         plt.show()
```



In [17]:
```
plt.figure(figsize=(12, 6))
plt.bar(df['Product line'], df['Total'], color='skyblue')
plt.xlabel('Product Category')
plt.ylabel('Total Sales Amount')
plt.title('Total Sales by Product Category')
plt.xticks(rotation=20)
plt.show()
```