In [5]:

```python
# import pandas library
import pandas as pd
```

In [7]:

```python
#imported retail sales dataset
df = pd.read_csv(r"C:\Users\ho\Desktop\retail_sales_dataset.csv")
df
```

Out[7]:

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
0	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
1	2	2023-02-27	CUST002	Female	26	Clothing	2	500	1000
2	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
3	4	2023-05-21	CUST004	Male	37	Clothing	1	500	500
4	5	2023-05-06	CUST005	Male	30	Beauty	2	50	100
...	...	...	...	...	...	...	...	...	...
995	996	2023-05-16	CUST996	Male	62	Clothing	1	50	50
996	997	2023-11-17	CUST997	Male	52	Beauty	3	30	90
997	998	2023-10-29	CUST998	Female	23	Beauty	4	25	100
998	999	2023-12-05	CUST999	Female	36	Electronics	3	50	150
999	1000	2023-04-12	CUST1000	Male	47	Electronics	4	30	120

1000 rows × 9 columns

In [8]:

```python
#To count head entries
df.head()
```

Out[8]:

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
0	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
1	2	2023-02-27	CUST002	Female	26	Clothing	2	500	1000
2	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
3	4	2023-05-21	CUST004	Male	37	Clothing	1	500	500
4	5	2023-05-06	CUST005	Male	30	Beauty	2	50	100

In [9]:

```python
df.tail()
```

Out[9]:

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
995	996	2023-05-16	CUST996	Male	62	Clothing	1	50	50
996	997	2023-11-17	CUST997	Male	52	Beauty	3	30	90
997	998	2023-10-29	CUST998	Female	23	Beauty	4	25	100
998	999	2023-12-05	CUST999	Female	36	Electronics	3	50	150
999	1000	2023-04-12	CUST1000	Male	47	Electronics	4	30	120

In [10]:

```python
#TO get basic information about dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Transaction ID    1000 non-null   int64 
 1   Date              1000 non-null   object
 2   Customer ID       1000 non-null   object
 3   Gender            1000 non-null   object
 4   Age               1000 non-null   int64 
 5   Product Category  1000 non-null   object
 6   Quantity          1000 non-null   int64 
 7   Price per Unit    1000 non-null   int64 
 8   Total Amount      1000 non-null   int64 
dtypes: int64(5), object(4)
memory usage: 70.4+ KB
```

In [11]:

```python
df.drop_duplicates()
#There are no duplicate values
```

Out[11]:

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
0	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
1	2	2023-02-27	CUST002	Female	26	Clothing	2	500	1000
2	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
3	4	2023-05-21	CUST004	Male	37	Clothing	1	500	500
4	5	2023-05-06	CUST005	Male	30	Beauty	2	50	100
...	...	...	...	...	...	...	...	...	...
995	996	2023-05-16	CUST996	Male	62	Clothing	1	50	50
996	997	2023-11-17	CUST997	Male	52	Beauty	3	30	90
997	998	2023-10-29	CUST998	Female	23	Beauty	4	25	100
998	999	2023-12-05	CUST999	Female	36	Electronics	3	50	150
999	1000	2023-04-12	CUST1000	Male	47	Electronics	4	30	120

1000 rows × 9 columns

In [12]:

```python
#Checking null values in all columns
df.isnull().sum()
```

Out[12]:

```
Transaction ID      0
Date                0
Customer ID         0
Gender              0
Age                 0
Product Category    0
Quantity            0
Price per Unit      0
Total Amount        0
dtype: int64
```

In [13]:

```python
# Descipitve Statistics to get mean, median,std and max values
df.describe()
```

Out[13]:

	Transaction ID	Age	Quantity	Price per Unit	Total Amount
count	1000.000000	1000.00000	1000.000000	1000.000000	1000.000000
mean	500.500000	41.39200	2.514000	179.890000	456.000000
std	288.819436	13.68143	1.132734	189.681356	559.997632
min	1.000000	18.00000	1.000000	25.000000	25.000000
25%	250.750000	29.00000	1.000000	30.000000	60.000000
50%	500.500000	42.00000	3.000000	50.000000	135.000000
75%	750.250000	53.00000	4.000000	300.000000	900.000000
max	1000.000000	64.00000	4.000000	500.000000	2000.000000

In [14]:

```python
#Modal values for Each Column
df.mode().dropna()
```

Out[14]:

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
0	1	2023-05-16	CUST001	Female	43.0	Clothing	4.0	50.0	50.0

In [15]:

```python
# Ordering Data by Date so as to do TIME SERIES ANALYSIS
df = df.sort_values(by="Date")
df.head(8)
```

Out[15]:

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
521	522	2023-01-01	CUST522	Male	46	Beauty	3	500	1500
179	180	2023-01-01	CUST180	Male	41	Clothing	3	300	900
558	559	2023-01-01	CUST559	Female	40	Clothing	4	300	1200
302	303	2023-01-02	CUST303	Male	19	Electronics	3	30	90
978	979	2023-01-02	CUST979	Female	19	Beauty	1	25	25
162	163	2023-01-02	CUST163	Female	64	Clothing	3	50	150
420	421	2023-01-02	CUST421	Female	37	Clothing	3	500	1500
609	610	2023-01-03	CUST610	Female	26	Beauty	2	300	600

In [16]:

```python
# IMPORITNG "MATPLOTLIB" AND "SEABORN"
import matplotlib.pyplot as plt
import seaborn as sns
```

In [17]:

```python
#CONVERTING DATE TO Date_Time FORMAT
df ["Date"] = pd.to_datetime(df['Date'])
```

In [19]:

```python
#OBTAINING A PLOT OF TOTAL AMOUNT OF PARTICULAR DATES USING SEABORN
plt.figure(figsize=(15,5))
sns.lineplot(x="Date", y="Total Amount", data=df)
plt.show()
```

In [21]:

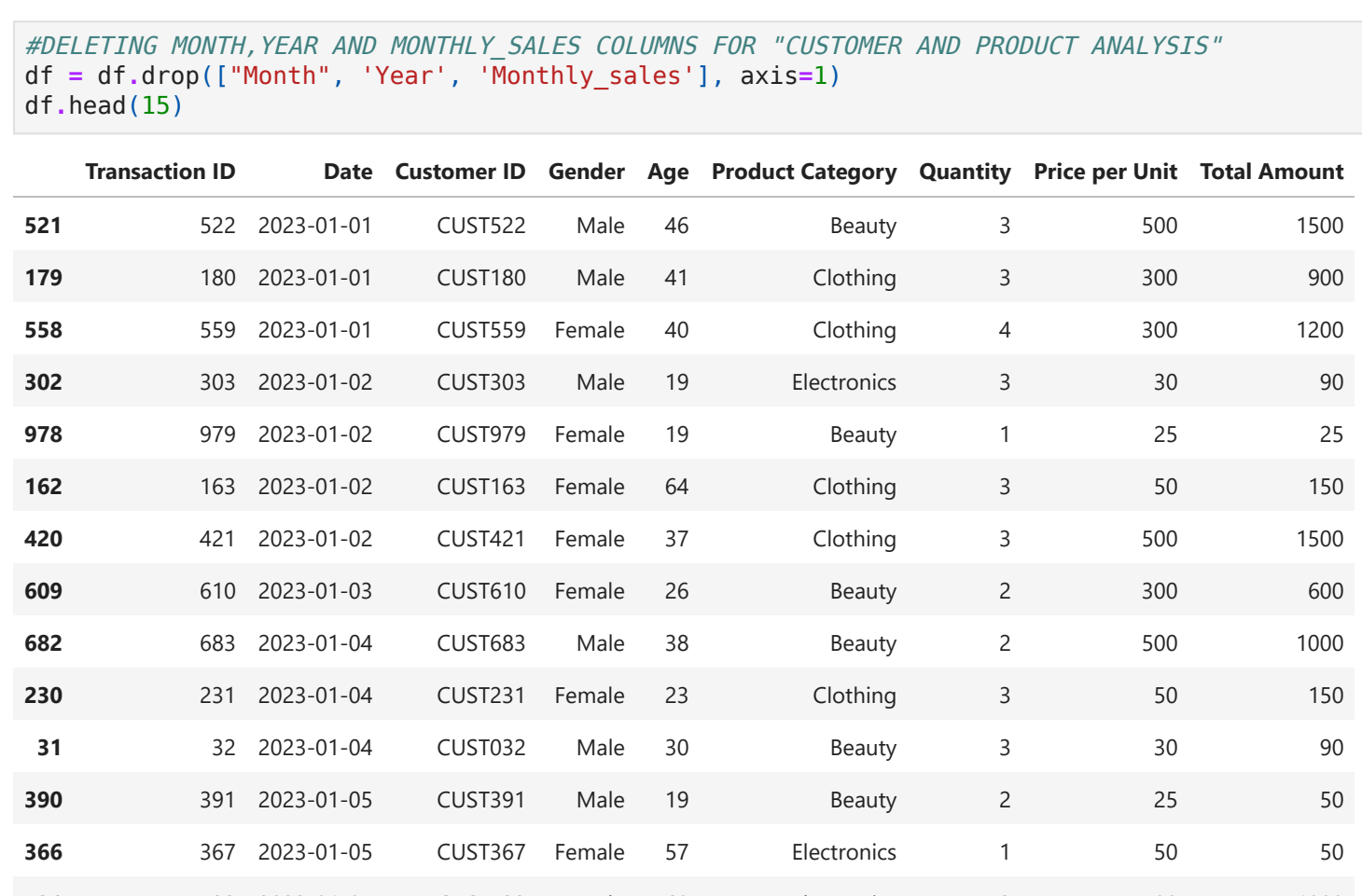
```python
#CREATING DATE AND MONTH COLUMNS
df["Month"]=df["Date"].dt.month
df["Year"]=df["Date"].dt.year
```

In [22]:

```python
#CALCULATING TOTAL MONTHLY SALES
Monthly_sales = df.groupby(["Year", "Month"], as_index = False)["Total Amount"].sum()
df['Monthly_sales']= Monthly_sales['Total Amount']
```

In [23]:

```python
#A PLOT OF TOTAL MONTHLY SALES THROUGHOUT THE YEAR
plt.figure(figsize=(8,5))
sns.lineplot(x='Date', y ='Monthly_sales', data=df)
plt.show()
```



In [24]:

```python
#DELETING MONTH,YEAR AND MONTHLY_SALES COLUMNS FOR "CUSTOMER AND PRODUCT ANALYSIS"
df = df.drop(["Month", 'Year', 'Monthly_sales'], axis=1)
df.head(15)
```

Out[24]:

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
521	522	2023-01-01	CUST522	Male	46	Beauty	3	500	1500
179	180	2023-01-01	CUST180	Male	41	Clothing	3	300	900
558	559	2023-01-01	CUST559	Female	40	Clothing	4	300	1200
302	303	2023-01-02	CUST303	Male	19	Electronics	3	30	90
978	979	2023-01-02	CUST979	Female	19	Beauty	1	25	25
162	163	2023-01-02	CUST163	Female	64	Clothing	3	50	150
420	421	2023-01-02	CUST421	Female	37	Clothing	3	500	1500
609	610	2023-01-03	CUST610	Female	26	Beauty	2	300	600
682	683	2023-01-04	CUST683	Male	38	Beauty	2	500	1000
230	231	2023-01-04	CUST231	Female	23	Clothing	3	50	150
31	32	2023-01-04	CUST032	Male	30	Beauty	3	30	90
390	391	2023-01-05	CUST391	Male	19	Beauty	2	25	50
366	367	2023-01-05	CUST367	Female	57	Electronics	1	50	50
431	432	2023-01-05	CUST432	Female	60	Electronics	2	500	1000
149	150	2023-01-06	CUST150	Female	58	Electronics	4	30	120

CUSTOMER AND PRODUCT ANALYSIS

In [48]:

```python
#CREATING AGE BRACKETS
age_bins = [10,20,30,40,50,60]
age_labels = ['10-19','20-29','30-39','40-49','50+']
df['Age group']=pd.cut(df['Age'],bins=age_bins,labels=age_labels,right=True)
```

In [28]:

```python
#CALCULATING TOTAL AMOUNT SPENT BY EACH AGE GROUP
Total_amount_by_age = df.groupby('Age group')['Total Amount'].sum()
Total_amount_by_age
```

Out[28]:

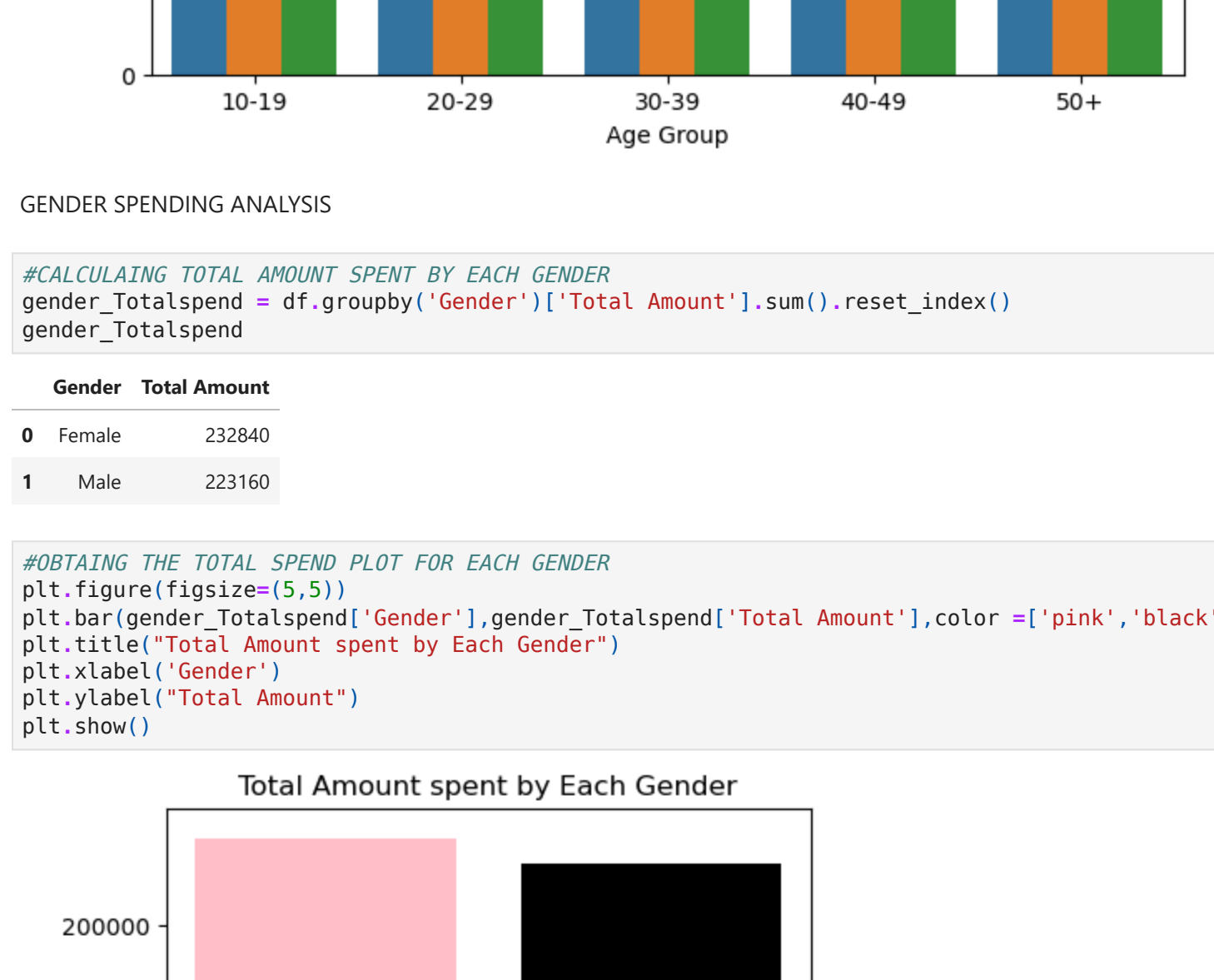
```
Age group
10-19     34730
20-29     98215
30-39     95950
40-49     93795
50+      100085
Name: Total Amount, dtype: int64
```

In [31]:

```python
#GROUPING BY AGE GROUP AND PRODUCT CATEGORY, AND CALCULATING THE SUM OF SPENDING
grouped_data = df.groupby(["Age group","Product Category"])["Total Amount"].sum().reset_index()
```

In [32]:

```python
#CREATING A BARPLOT TO VISUALIZE SPENDING PATTERNS BY AGE GROUP AND PRODUCT CATEGORY
plt.figure(figsize=(8,6))
sns.barplot(x='Age group', y='Total Amount', hue='Product Category',data=grouped_data)
plt.title("spending Patterns by Age Group and Product Category")
plt.xlabel('Age Group')
plt.ylabel("Total Amount")
plt.show()
```



GENDER SPENDING ANALYSIS

In [33]:

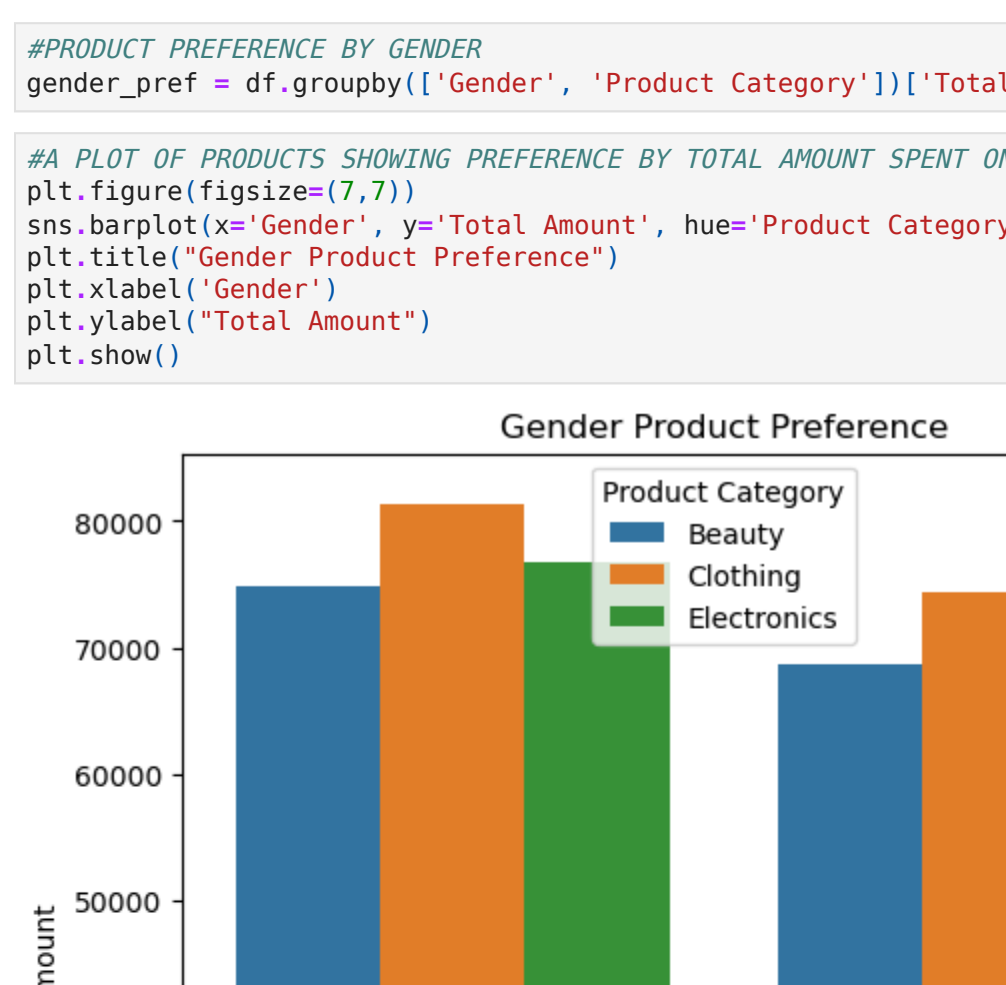
```python
#CALCULAING TOTAL AMOUNT SPENT BY EACH GENDER
gender_Totalspend = df.groupby('Gender')['Total Amount'].sum().reset_index()
gender_Totalspend
```

Out[33]:

	Gender	Total Amount
0	Female	232840
1	Male	223160

In [36]:

```python
#OBTAING THE TOTAL SPEND PLOT FOR EACH GENDER
plt.figure(figsize=(5,5))
plt.bar(gender_Totalspend['Gender'],gender_Totalspend['Total Amount'],color =['pink','black'])
plt.title("Total Amount spent by Each Gender")
plt.xlabel('Gender')
plt.ylabel("Total Amount")
plt.show()
```



In [37]:

```python
#PRODUCT PREFERENCE BY GENDER
gender_pref = df.groupby(['Gender', 'Product Category'])['Total Amount'].sum().reset_index()
```

In [38]:

```python
#A PLOT OF PRODUCTS SHOWING PREFERENCE BY TOTAL AMOUNT SPENT ON EACH BY EACH GENDER
plt.figure(figsize=(7,7))
sns.barplot(x='Gender', y='Total Amount', hue='Product Category',data=gender_pref)
plt.title("Gender Product Preference")
plt.xlabel('Gender')
plt.ylabel("Total Amount")
plt.show()
```

In [40]:

```python
#TOTAL AMOUNT SPENT BY AVERAGE MALE VS FEMALE
mean_amnt_by_gender = df.groupby('Gender')['Total Amount'].mean()
mean_amnt_by_gender
```

Out[40]:

```
Gender
Female    456.549020
Male      455.428571
Name: Total Amount, dtype: float64
```

PRODUCT PERFORMANCE ANALYSIS

In [42]:

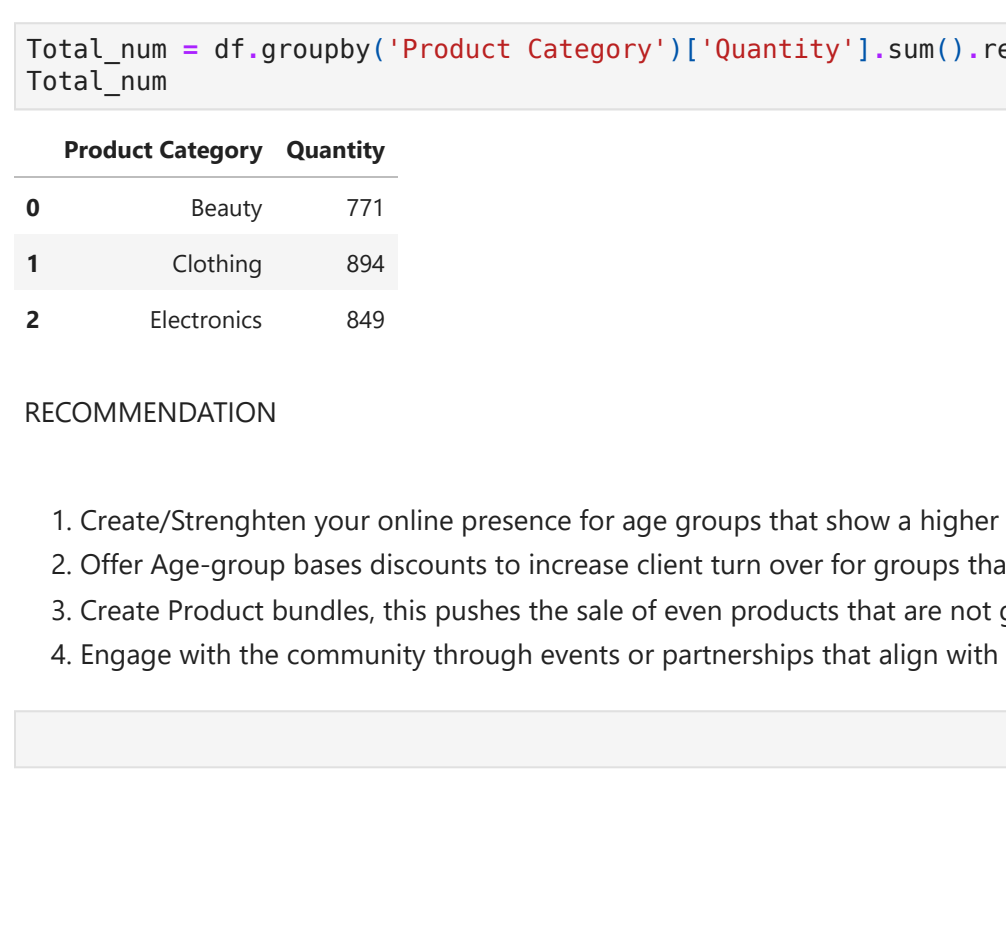
```python
#TOTAL REVENUE FROM EACH PRODUCT CATEGORY
Total_revenue = df.groupby('Product Category')['Total Amount'].sum().reset_index()
Total_revenue
```

Out[42]:

	Product Category	Total Amount
0	Beauty	143515
1	Clothing	155580
2	Electronics	156905

In [45]:

```python
plt.figure(figsize=(5,5))
plt.bar(Total_revenue['Product Category'],Total_revenue['Total Amount'],color =['red','yellow','green'])
plt.title("Total Revenue from Each Product")
plt.xlabel('Product Category')
plt.ylabel("Total Amount")
plt.show()
```



In [47]:

```python
Total_num = df.groupby('Product Category')['Quantity'].sum().reset_index()
Total_num
```

Out[47]:

	Product Category	Quantity
0	Beauty	771
1	Clothing	894
2	Electronics	849

RECOMMENDATION

1. Create/Strenghten your online presence for age groups that show a higher inclination towards digital channel to improve sales.
2. Offer Age-group bases discounts to increase client turn over for groups that are not performing well.
3. Create Product bundles, this pushes the sale of even products that are not getting alot of sales.
4. Engage with the community through events or partnerships that align with the interesets of each group.

In [ ]: