In [5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
gender_colors = {'Male': '#1f77b2',  # blue
                 'Female': '#ff69b2'}  # pink
```

In [3]:
```
df = pd.read_csv(r"D:\ppp\capstone\retail_sales_dataset.csv", encoding="unicode_
```

Data processing

In [7]:
```
df.shape
```

Out[7]:
```
(1000, 9)
```

In [9]:
```
df.head(10)
```

Out[9]:

	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
5	6	2023-04-25	CUST006	Female	45	Beauty	1	30	30
6	7	2023-03-13	CUST007	Male	46	Clothing	2	25	50
7	8	2023-02-22	CUST008	Male	30	Electronics	4	25	100
8	9	2023-12-13	CUST009	Male	63	Electronics	2	300	600
9	10	2023-10-07	CUST010	Female	52	Clothing	4	50	200

In [16]:
```
df.duplicated().sum()
```

Out[16]:
```
np.int64(0)
```

In [17]:
```
df.describe(include='all')
```

Out[17]:

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	
count	1000.000000	1000	1000	1000	1000.00000	1000	1000.000000	100
unique	NaN	345	1000	2	NaN	3	NaN	
top	NaN	2023-05-16	CUST1000	Female	NaN	Clothing	NaN	
freq	NaN	11	1	510	NaN	351	NaN	
mean	500.500000	NaN	NaN	NaN	41.39200	NaN	2.514000	17
std	288.819436	NaN	NaN	NaN	13.68143	NaN	1.132734	18
min	1.000000	NaN	NaN	NaN	18.00000	NaN	1.000000	2
25%	250.750000	NaN	NaN	NaN	29.00000	NaN	1.000000	3
50%	500.500000	NaN	NaN	NaN	42.00000	NaN	3.000000	5
75%	750.250000	NaN	NaN	NaN	53.00000	NaN	4.000000	30
max	1000.000000	NaN	NaN	NaN	64.00000	NaN	4.000000	50

In [10]:

```
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]:

```
pd.isnull(df)
```

Out[11]:

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
995	False	False	False	False	False	False	False	False	False
996	False	False	False	False	False	False	False	False	False
997	False	False	False	False	False	False	False	False	False
998	False	False	False	False	False	False	False	False	False
999	False	False	False	False	False	False	False	False	False

1000 rows × 9 columns

In [12]:
```
pd.isnull(df).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]:
```
df.columns
```

Out[13]:
```
Index(['Transaction ID', 'Date', 'Customer ID', 'Gender', 'Age',
       'Product Category', 'Quantity', 'Price per Unit', 'Total Amount'],
      dtype='object')
```

In [14]:
```
df.describe()
```

Out[14]:

	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 [15]:

```
df.dropna(inplace=True)
```

In [16]:

```
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 [108…

```
total_revenue = df['Total Amount'].sum()
unique_customers = df['Customer ID'].nunique()
average_order_value = df['Total Amount'].mean()

print("Total Revenue:", total_revenue)
print("Unique Customers:", unique_customers)
print("Average Order Value:", average_order_value)
```

```
Total Revenue: 456000
Unique Customers: 1000
Average Order Value: 456.0
```

Key Performance Indicators (KPIs)Total Revenue: 456000 Unique Customers: 1000 Average Order Value: 456.0

Exploratory Data Analysis

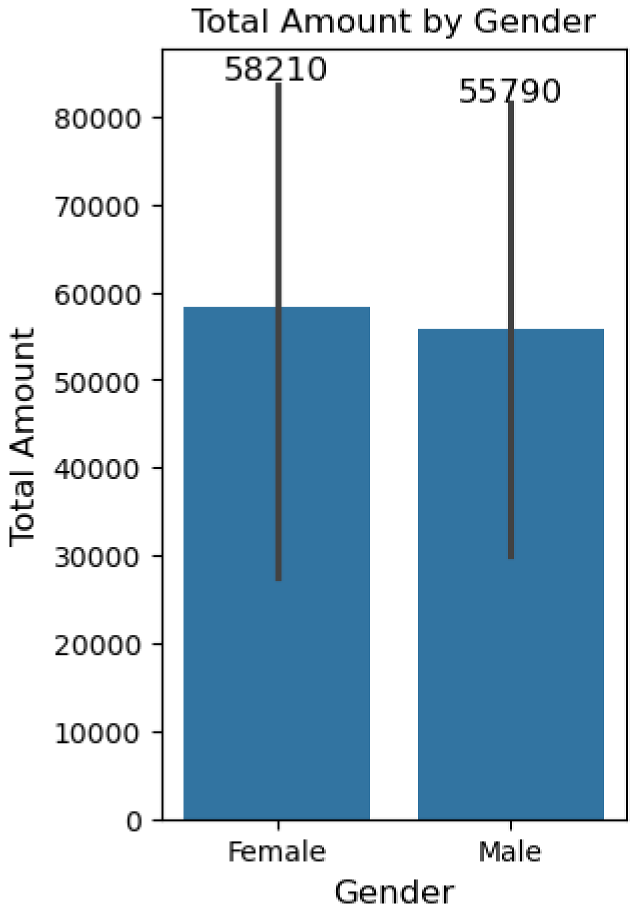
In [86]:

```
Amount_Gender = df.groupby(['Gender'], as_index=False)['Total Amount'].sum()
```

```
plt.figure(figsize=(3,5))
ax = sns.barplot(x='Gender', y='Total Amount', data=Amount_Quantity)

# Add value labels
for container in ax.containers:
    ax.bar_label(container, fmt='%.0f', label_type='edge', padding=80, fontsize=

plt.title('Total Amount by Gender', fontsize=12)
plt.xlabel('Gender', fontsize=12)
plt.ylabel('Total Amount', fontsize=12)
plt.show()
```



Total Amount by Gender.

In [21]:

```
df['Date'].describe()
```

Out[21]:

```
count                              1000
mean      2023-07-03 00:25:55.200000256
min                 2023-01-01 00:00:00
25%                 2023-04-08 00:00:00
50%                 2023-06-29 12:00:00
75%                 2023-10-04 00:00:00
max                 2024-01-01 00:00:00
Name: Date, dtype: object
```

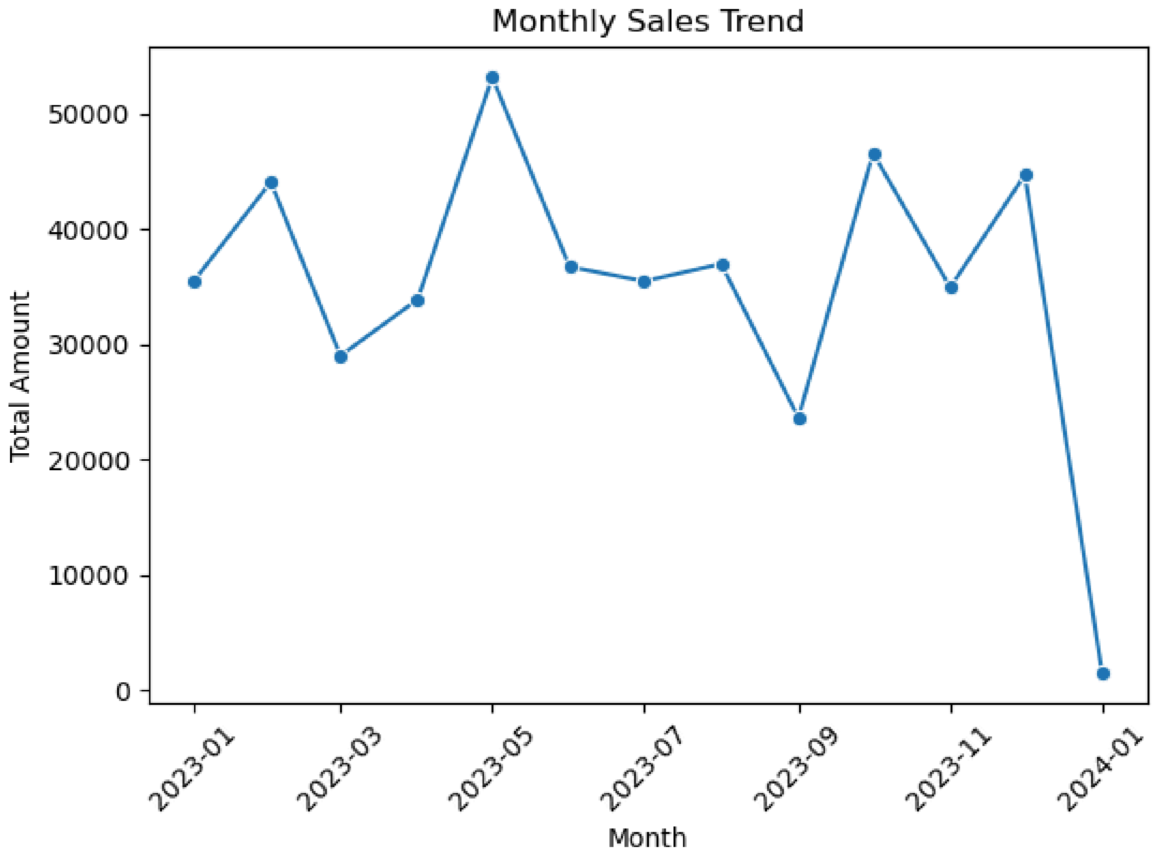
In [12]:

```
# Convert 'Date' to datetime and store in a new column
df['Date_converted'] = pd.to_datetime(df['Date'])

# Extract month from the converted date column
df['Month'] = df['Date_converted'].dt.to_period('M').dt.to_timestamp()
```

```
# Group by month
monthly_sales = df.groupby('Month')['Total Amount'].sum().reset_index()


ax=sns.lineplot(data=monthly_sales, x='Month', y='Total Amount', marker='o')
plt.xticks(rotation=45)
plt.title('Monthly Sales Trend')
plt.tight_layout()
plt.show()
```



Monthly sales trend drops at January 2024.

In [58]:
```
df['Age_Group'] = pd.cut(df['Age'], bins=[0, 18, 25, 35, 45, 60, 100], labels=['
```

In [22]:
```
df.Age_Group
```

Out[22]:
```
0      26-35
1      26-35
2      46-60
3      36-45
4      26-35
       ...  
995      60+
996    46-60
997    19-25
998    36-45
999    46-60
Name: Age_Group, Length: 1000, dtype: category
Categories (6, object): ['0-18' < '19-25' < '26-35' < '36-45' < '46-60' < '60
+']
```

In [20]:

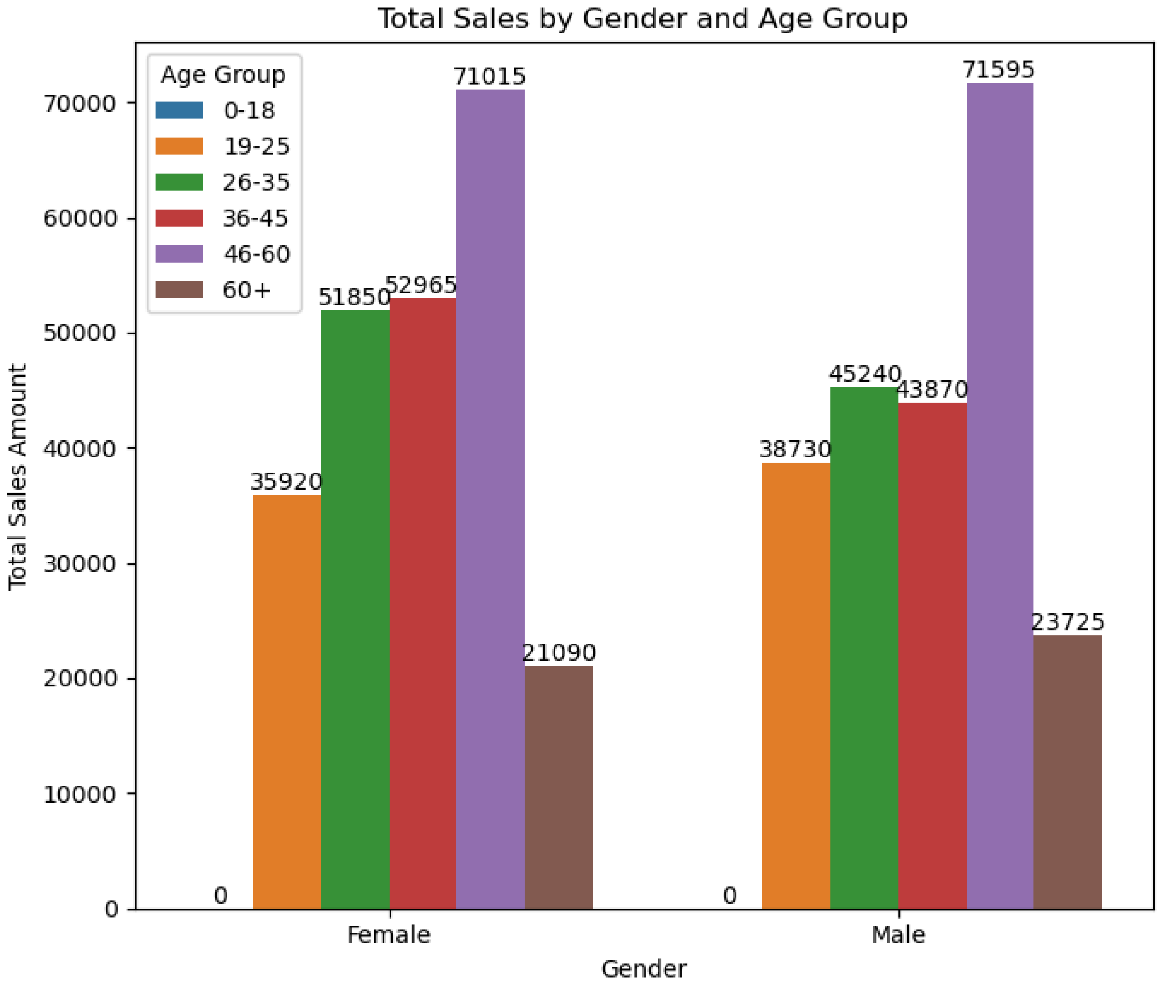
```
gender_age_sales = df.groupby(['Gender', 'Age_Group'], as_index=False)['Total Am

plt.figure(figsize=(7, 6))
ax = sns.barplot(data=gender_age_sales, x='Gender', y='Total Amount', hue='Age_G

# Add value labels on top of each bar
for container in ax.containers:
    ax.bar_label(container, fmt='%.0f')

plt.title('Total Sales by Gender and Age Group')
plt.xlabel('Gender')
plt.ylabel('Total Sales Amount')
plt.legend(title='Age Group')
plt.tight_layout()
plt.show()
```

```
C:\Users\SOURAV\AppData\Local\Temp\ipykernel_14512\3855600249.py:1: FutureWarnin
g: The default of observed=False is deprecated and will be changed to True in a f
uture version of pandas. Pass observed=False to retain current behavior or observ
ed=True to adopt the future default and silence this warning.
  gender_age_sales = df.groupby(['Gender', 'Age_Group'], as_index=False)['Total A
mount'].sum()
```



46-60 age group possesses the highest total amount in case of both genders.

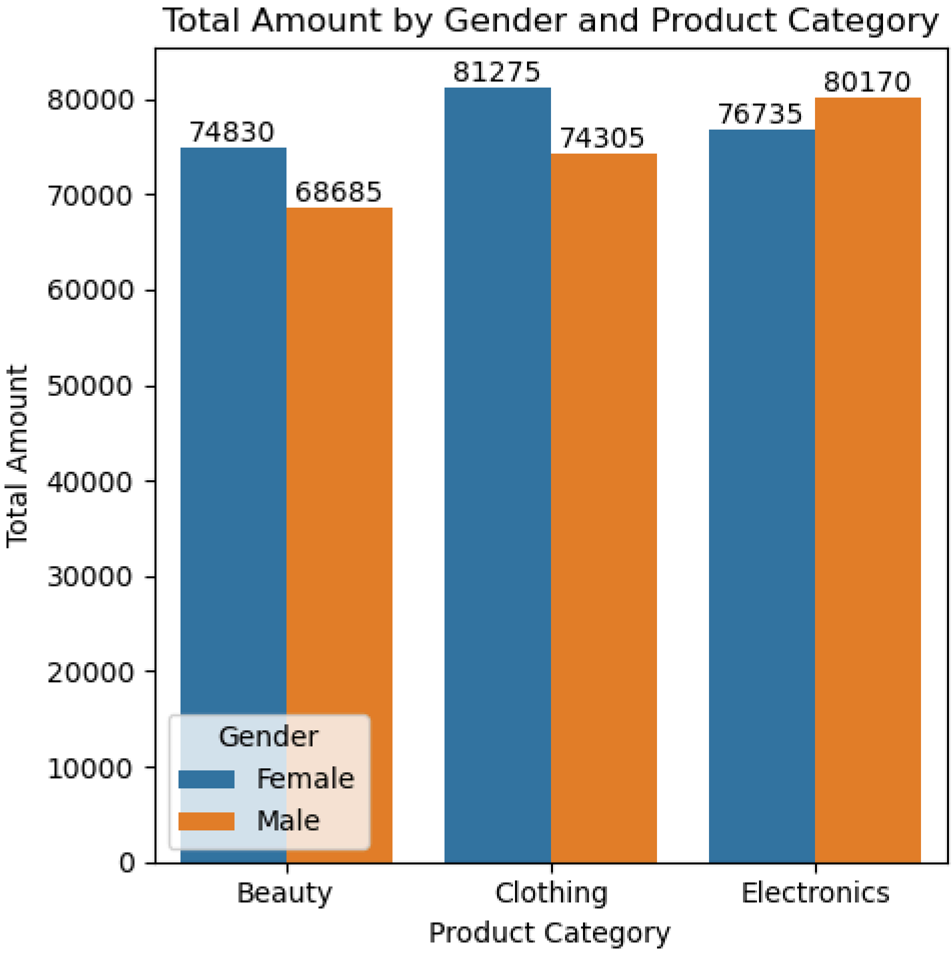
In [13]:

```
Gender_Product_Category = df.groupby(['Gender', 'Product Category'], as_index=Fa
plt.figure(figsize=(5,5))
```

```
# Create bar plot
ax = sns.barplot(x='Product Category', y='Total Amount', hue='Gender', data=Gend

# Add value labels on top of each bar
for container in ax.containers:
    ax.bar_label(container, fmt='%.0f')  # Display whole numbers


plt.title('Total Amount by Gender and Product Category')
plt.xlabel('Product Category')
plt.ylabel('Total Amount')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```

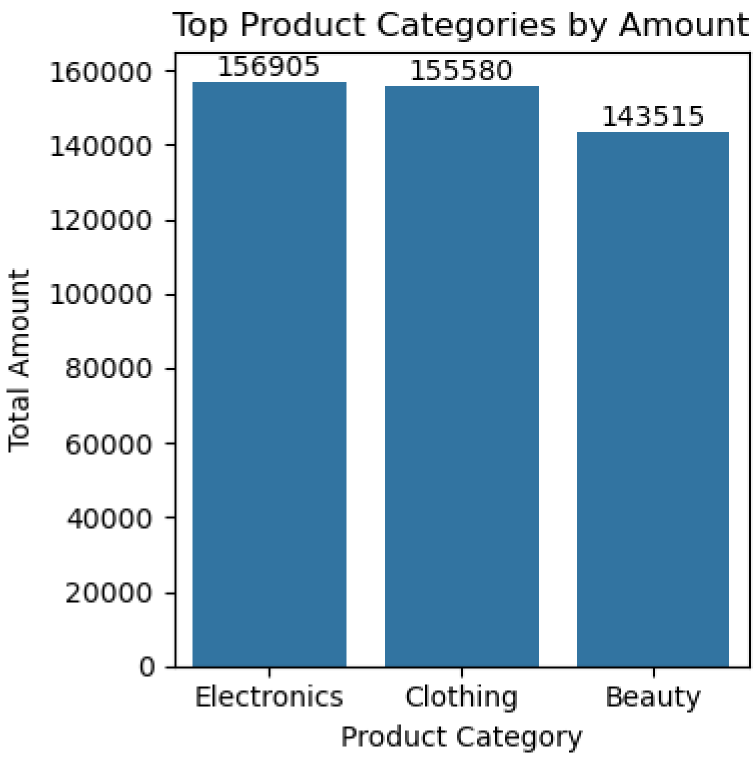


Male customer buys more electronics than female; on the other hand, beauty and clothing got purchased more by female.

In [26]:
```
top_products = df.groupby('Product Category', as_index=False)['Total Amount'].su
top_products = top_products.sort_values('Total Amount', ascending=False).head(10

plt.figure(figsize=(4,4))
ax=sns.barplot(data=top_products, y='Total Amount', x='Product Category')
for container in ax.containers:
    ax.bar_label(container, fmt='%.0f')
plt.title("Top Product Categories by Amount")
```

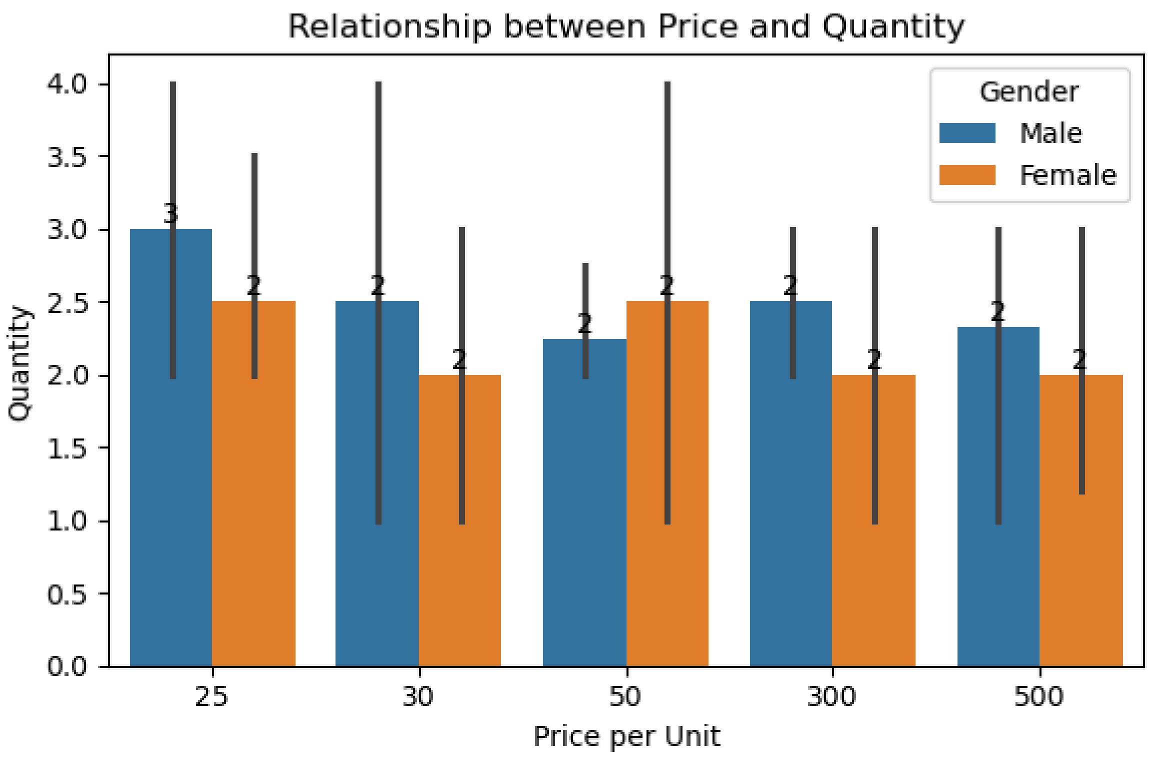
```python
plt.ylabel("Total Amount")
plt.xlabel("Product Category")
plt.tight_layout()
plt.show()
```



Electronics, Clothing, and Beauty are the top product categories.

In [56]:

```python
plt.figure(figsize=(6, 4))  # Slightly wider for clarity
ax=sns.barplot(x='Price per Unit', y='Quantity', hue='Gender', data=df.head(30))
for container in ax.containers:
    ax.bar_label(container, fmt='%.0f')
plt.title('Relationship between Price and Quantity')
plt.xlabel('Price per Unit')
plt.ylabel('Quantity')
plt.tight_layout()
plt.show()
```

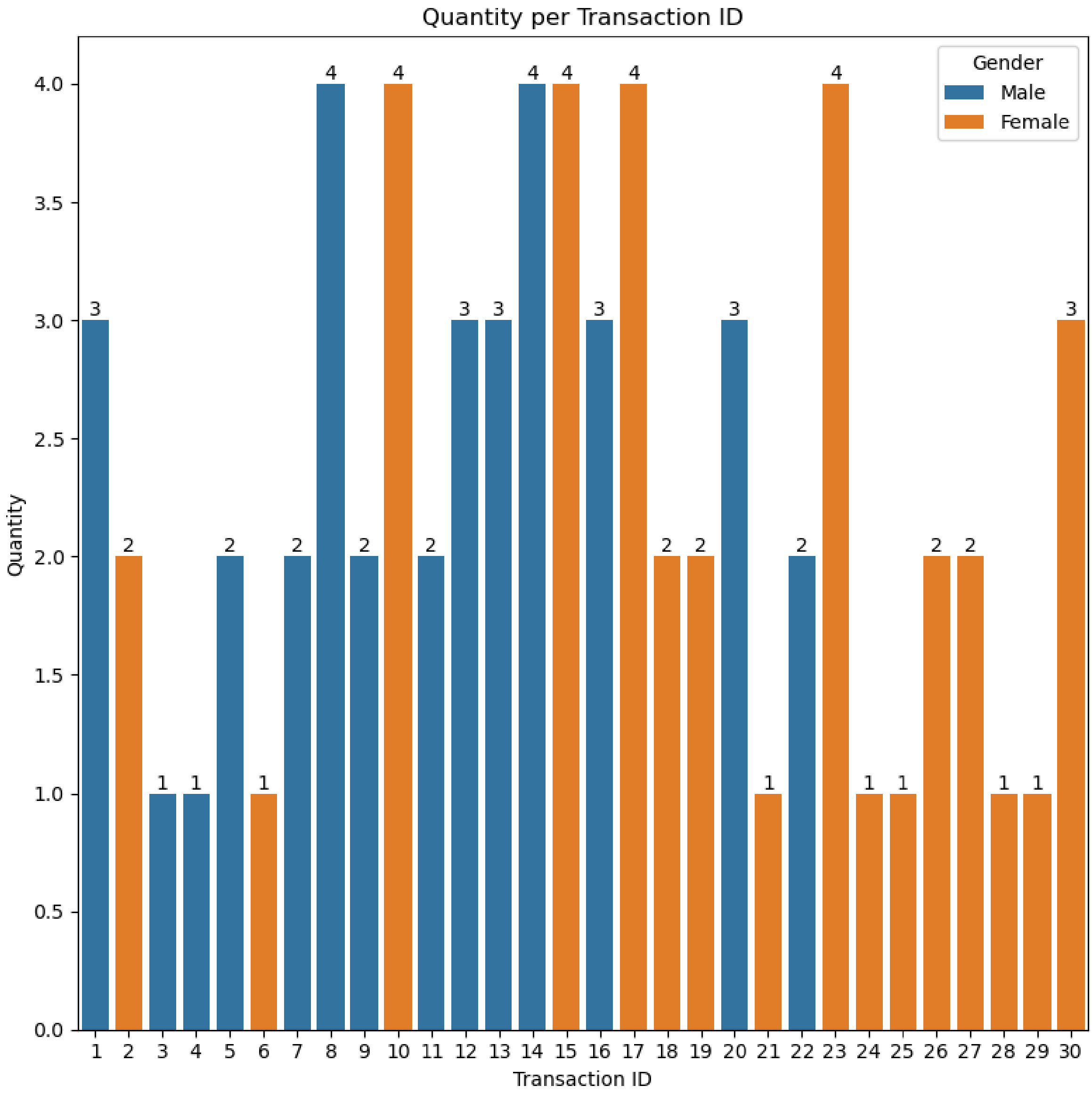


As the price per unit increases, both genders' purchase quantity reduces.

In [39]:

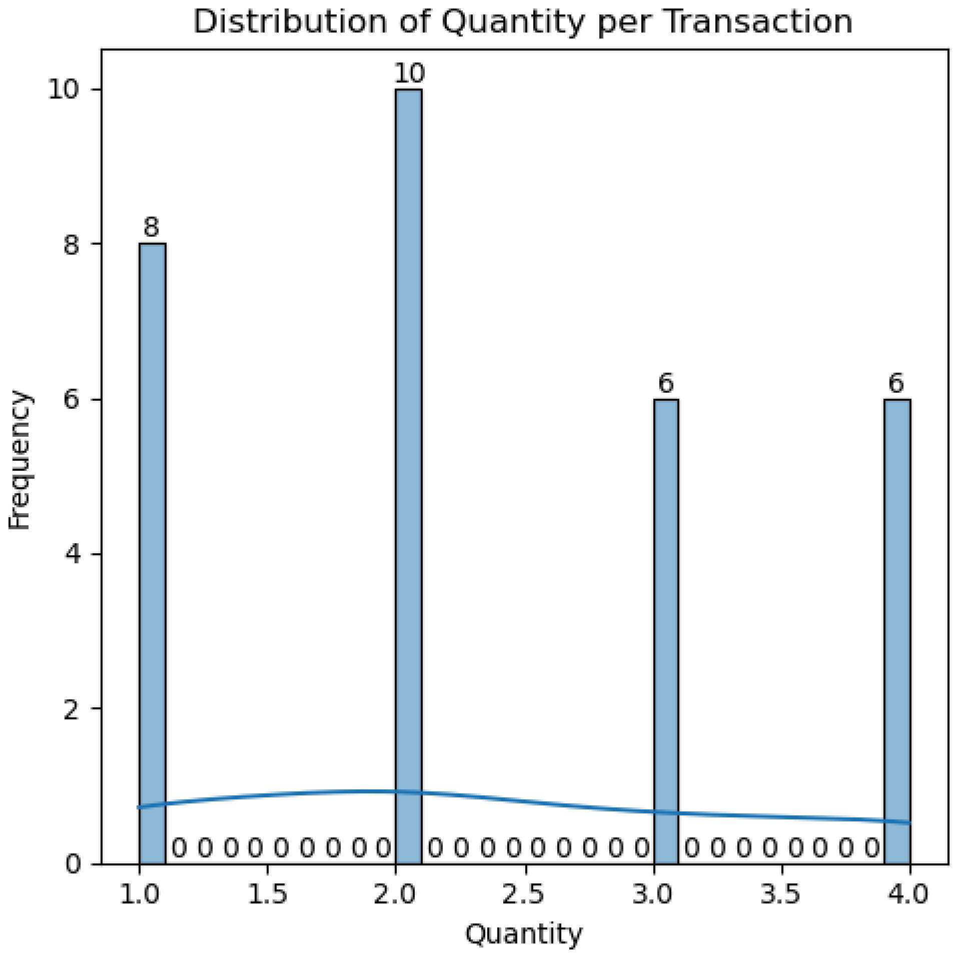
```
Transaction_Quantity = df.groupby(['Transaction ID','Gender'], as_index=False)['

plt.figure(figsize=(8,8))
ax = sns.barplot(x='Transaction ID', y='Quantity', hue='Gender', data=Transactic
for container in ax.containers:
    ax.bar_label(container, fmt='%.0f')
plt.title('Quantity per Transaction ID')
plt.xlabel('Transaction ID')
plt.ylabel('Quantity')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



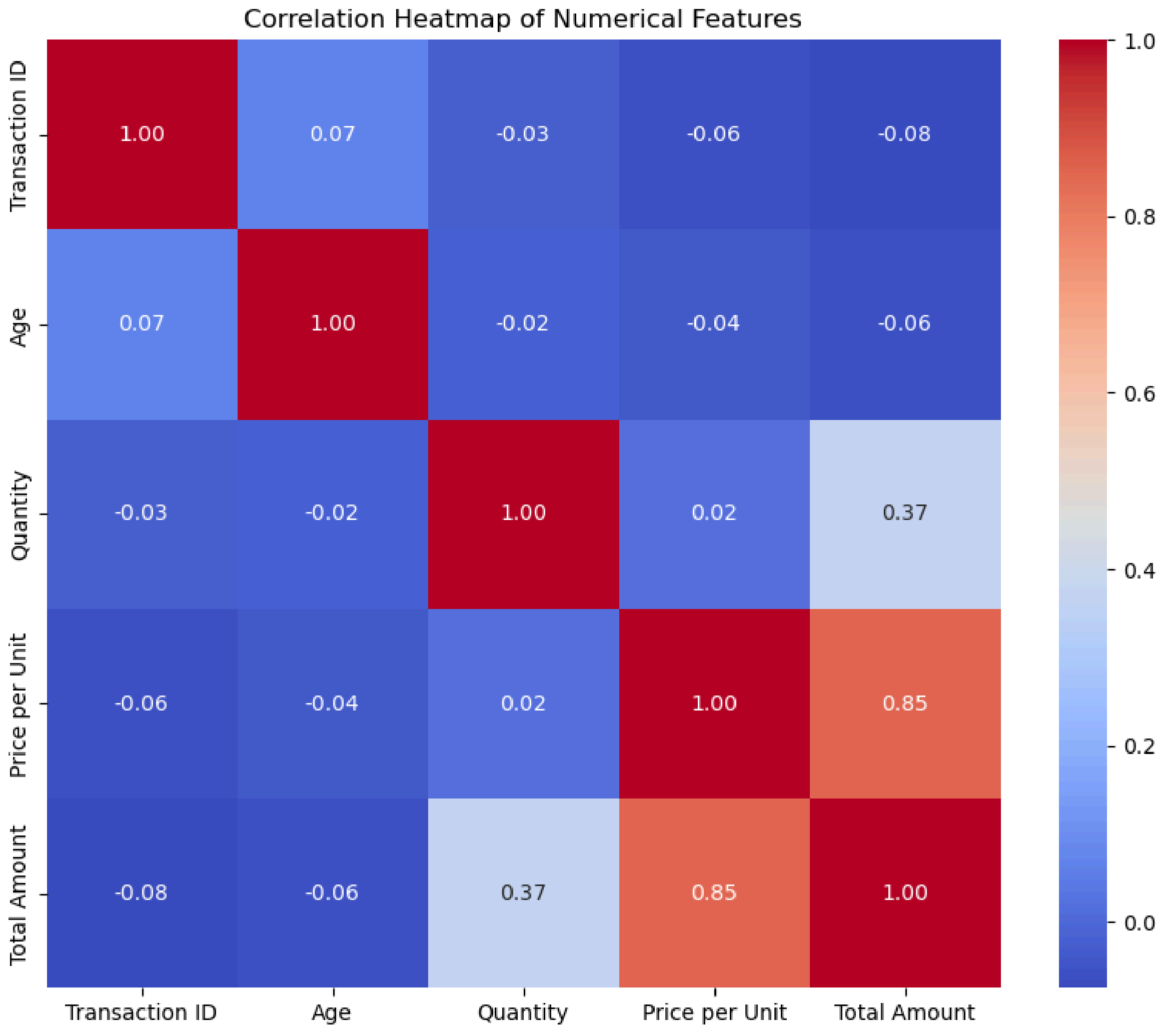
In [44]:

```
plt.figure(figsize=(5,5))
ax=sns.histplot(Transaction_Quantity['Quantity'], bins=30, kde=True)
for container in ax.containers:
    ax.bar_label(container, fmt='%.0f')
plt.title('Distribution of Quantity per Transaction')
plt.xlabel('Quantity')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```



In [48]:

```
plt.figure(figsize=(10, 8))
correlation_matrix = df.select_dtypes(include=[np.number]).corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of Numerical Features')
plt.show()
```



Strong Positive Correlation between `Price per Unit` and `Total Amount` (0.87) moderate correlation between 'Quantity' and 'Price per unit'(0.37).

In [ ]: