In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries instal
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/d
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will lis

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that ge
# You can also write temporary files to /kaggle/temp/, but they won't be saved
```

```
/kaggle/input/customer-shopping-dataset/customer_shopping_data.csv
```

In [2]:
```
df = pd.read_csv("/kaggle/input/customer-shopping-dataset/customer_shopping_da
```

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

Out[3]:

	invoice_no	customer_id	gender	age	category	quantity	price	payment_
0	I138884	C241288	Female	28	Clothing	5	1500.40	Cre
1	I317333	C111565	Male	21	Shoes	3	1800.51	De
2	I127801	C266599	Male	20	Clothing	1	300.08	
3	I173702	C988172	Female	66	Shoes	5	3000.85	Cre
4	I337046	C189076	Female	53	Books	4	60.60	

In [4]:
```
df.tail()
```

Out[4]:

	invoice_no	customer_id	gender	age	category	quantity	price	pay
99452	I219422	C441542	Female	45	Souvenir	5	58.65	
99453	I325143	C569580	Male	27	Food & Beverage	2	10.46	
99454	I824010	C103292	Male	63	Food & Beverage	2	10.46	
99455	I702964	C800631	Male	56	Technology	4	4200.00	
99456	I232867	C273973	Female	36	Souvenir	3	35.19	

In [5]:
```
df.shape
```

Out[5]: (99457, 10)

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99457 entries, 0 to 99456
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   invoice_no      99457 non-null  object 
 1   customer_id     99457 non-null  object 
 2   gender          99457 non-null  object 
 3   age             99457 non-null  int64  
 4   category        99457 non-null  object 
 5   quantity        99457 non-null  int64  
 6   price           99457 non-null  float64
 7   payment_method  99457 non-null  object 
 8   invoice_date    99457 non-null  object 
 9   shopping_mall   99457 non-null  object 
dtypes: float64(1), int64(2), object(7)
memory usage: 7.6+ MB
```

In [7]:
```
df.describe()
```

Out[7]:

	age	quantity	price
count	99457.000000	99457.000000	99457.000000
mean	43.427089	3.003429	689.256321
std	14.990054	1.413025	941.184567
min	18.000000	1.000000	5.230000
25%	30.000000	2.000000	45.450000
50%	43.000000	3.000000	203.300000
75%	56.000000	4.000000	1200.320000
max	69.000000	5.000000	5250.000000

In [8]:
```
df.dtypes
```

Out[8]:
```
invoice_no         object
customer_id        object
gender             object
age                 int64
category           object
quantity            int64
price             float64
payment_method     object
invoice_date       object
shopping_mall      object
dtype: object
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
invoice_no        0
customer_id       0
gender            0
age               0
category          0
quantity          0
price             0
payment_method    0
invoice_date      0
shopping_mall     0
dtype: int64
```

In [10]:
```
df.duplicated().sum()
```

Out[10]:
```
0
```

In [11]:
```
df.columns
```

Out[11]:
```
Index(['invoice_no', 'customer_id', 'gender', 'age', 'category', 'quantity',
       'price', 'payment_method', 'invoice_date', 'shopping_mall'],
      dtype='object')
```

In [12]:
```
df.drop(columns=['customer_id', 'invoice_no'], inplace=True)
```

In [13]:
```
df['invoice_date'] = pd.to_datetime(df['invoice_date'], dayfirst=True)
```

In [14]:
```
df['year'] = df['invoice_date'].dt.year
df['month'] = df['invoice_date'].dt.month
df['day'] = df['invoice_date'].dt.day
```

In [15]:
```
df['total_amount'] = df['quantity'] * df['price']
```

In [16]:
```
df.head()
```

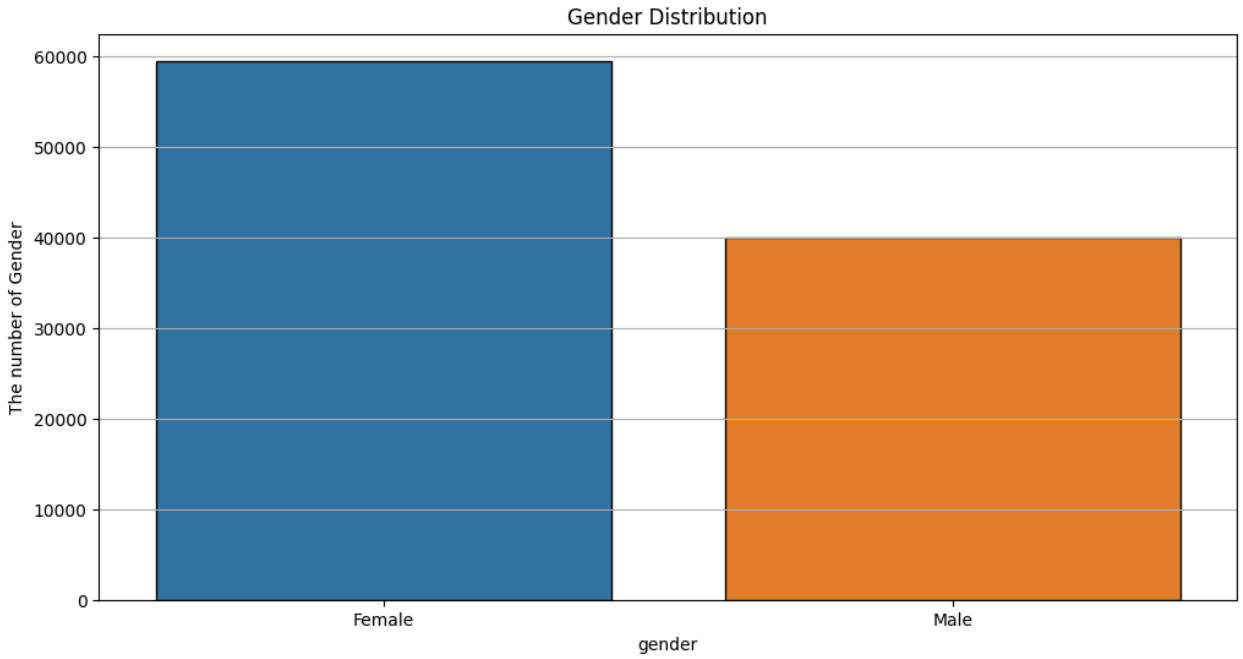
Out[16]:

	gender	age	category	quantity	price	payment_method	invoice_date	sh
0	Female	28	Clothing	5	1500.40	Credit Card	2022-08-05	
1	Male	21	Shoes	3	1800.51	Debit Card	2021-12-12	F
2	Male	20	Clothing	1	300.08	Cash	2021-11-09	
3	Female	66	Shoes	5	3000.85	Credit Card	2021-05-16	
4	Female	53	Books	4	60.60	Cash	2021-10-24	

In [17]:
```
df["gender"].value_counts()
```

Out[17]:
```
gender
Female    59482
Male      39975
Name: count, dtype: int64
```

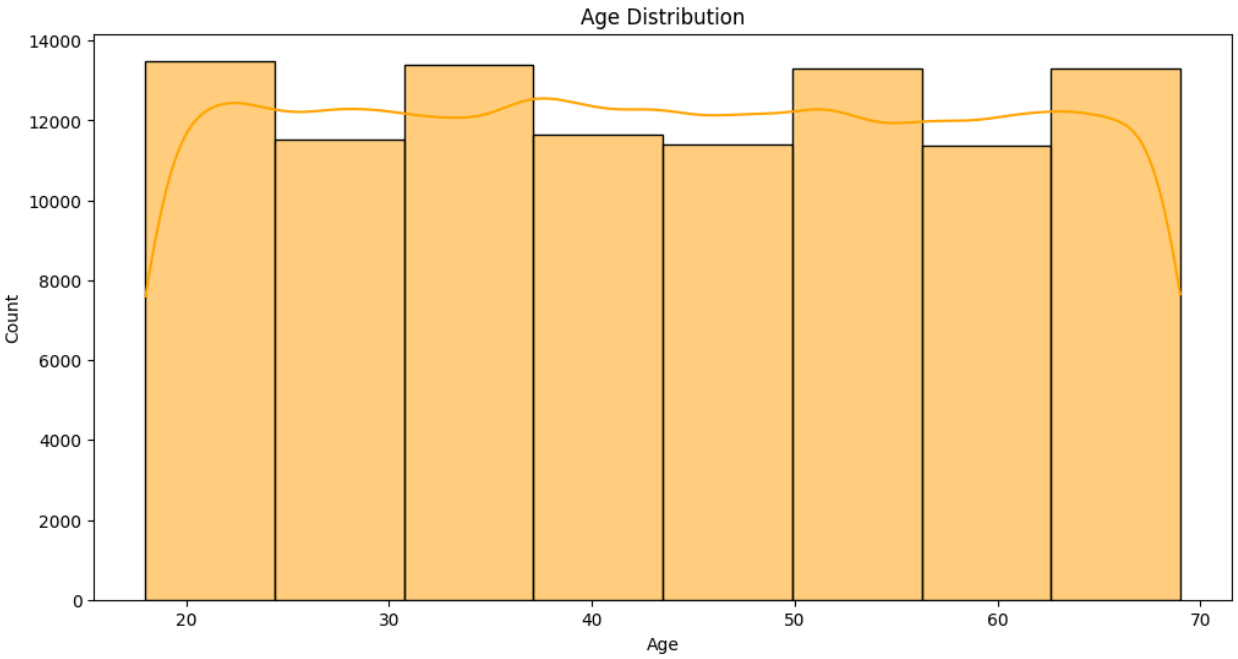
In [18]:
```
plt.figure(figsize = (12,6))
sns.countplot(data = df, x = "gender", edgecolor = "black")
plt.title("Gender Distribution")
plt.ylabel("The number of Gender")
plt.grid(axis = "y")
plt.show()
```



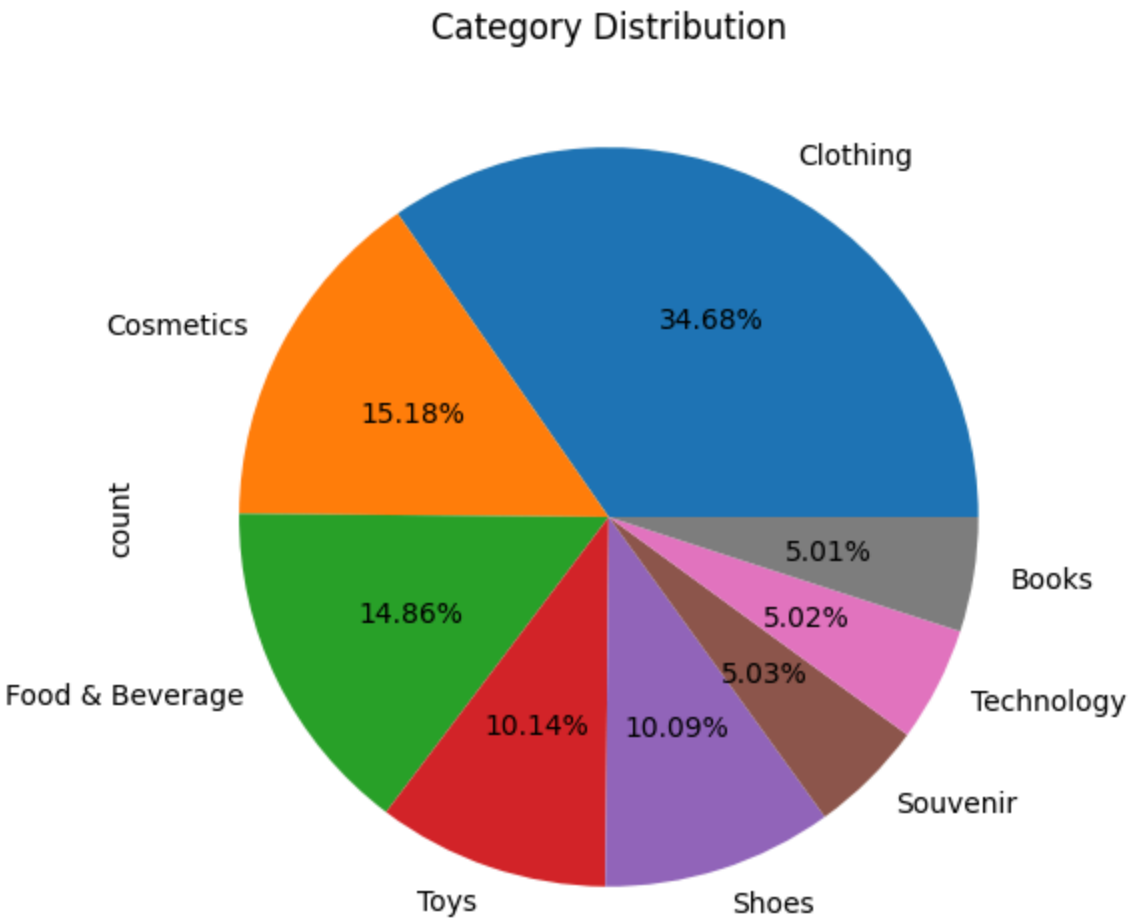
In [19]:
```
plt.figure(figsize = (12,6))
sns.histplot(data=df, x='age', bins=8, kde=True, color='orange', edgecolor='bl

plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



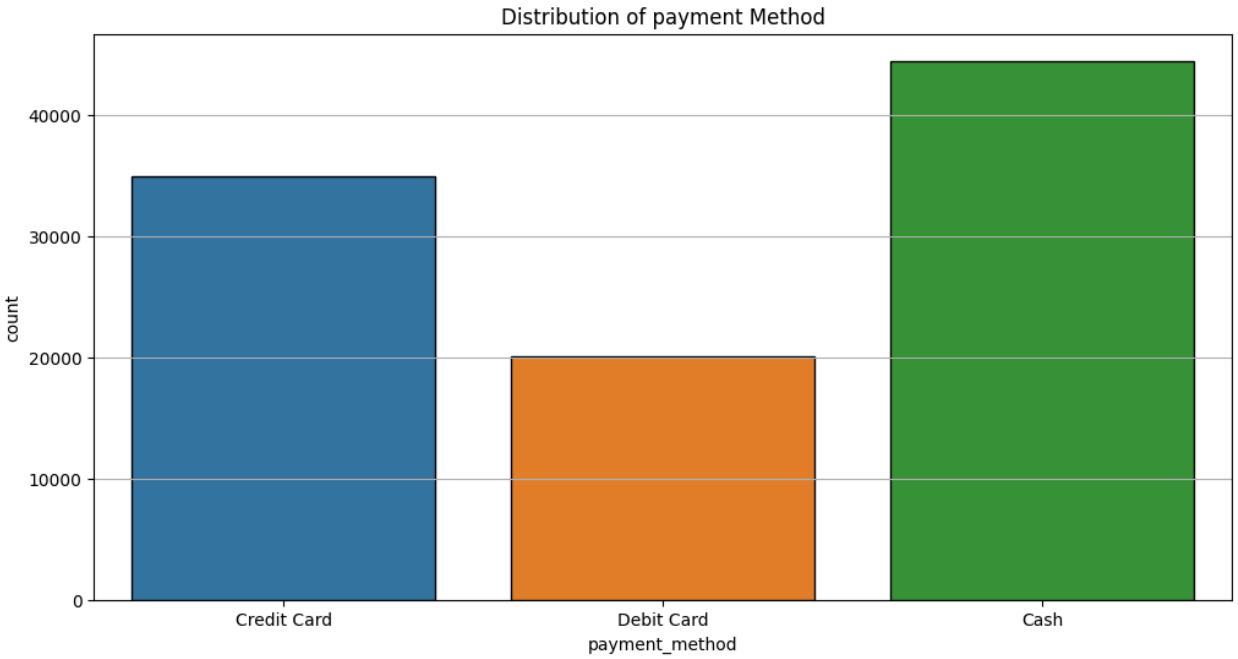
In [20]:
```
plt.figure(figsize = (12,6))
df["category"].value_counts().plot(kind = "pie", autopct = "%1.2f%%")
plt.title("Category Distribution")
plt.show()
```



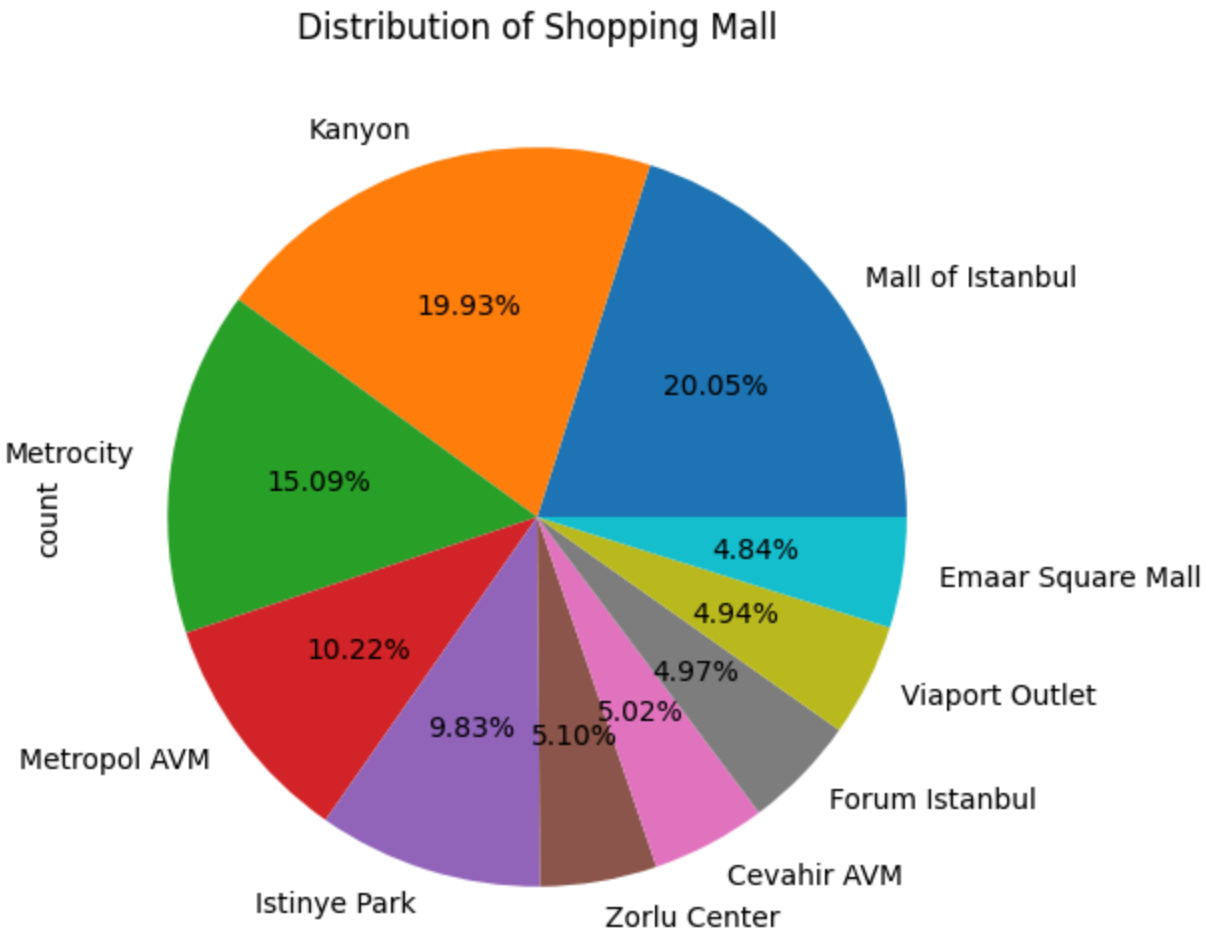
In [21]:
```
df["payment_method"].value_counts()
```

Out[21]:
```
payment_method
Cash           44447
Credit Card    34931
Debit Card     20079
Name: count, dtype: int64
```

In [22]:
```
plt.figure(figsize = (12,6))
sns.countplot(data = df, x = "payment_method", edgecolor = "black")
plt.title("Distribution of payment Method")
plt.grid(axis = "y")
plt.show()
```



In [23]:
```
plt.figure(figsize = (12,6))
df["shopping_mall"].value_counts().plot(kind = "pie", autopct = "%1.2f%%")
plt.title("Distribution of Shopping Mall")
plt.show()
```



In [24]:
```
df.columns
```

Out[24]:
```
Index(['gender', 'age', 'category', 'quantity', 'price', 'payment_method',
       'invoice_date', 'shopping_mall', 'year', 'month', 'day',
       'total_amount'],
      dtype='object')
```

In [25]:
```
df.groupby("age")["total_amount"].sum().sort_values(ascending = False).head(15
```

Out[25]:
```
age
37    5578539.57
64    5272475.10
51    5238724.74
22    5208841.41
27    5171859.06
39    5135672.77
48    5131747.86
44    5131686.80
24    5082409.90
43    5050323.72
40    5034207.14
42    5021505.35
46    5004837.20
38    4994225.53
26    4981414.82
Name: total_amount, dtype: float64
```

In [26]:
```
df.groupby("category")["quantity"].sum().sort_values(ascending = False)
```

Out[26]:
```
category
Clothing           103558
Cosmetics           45465
Food & Beverage     44277
Toys                30321
Shoes               30217
Technology          15021
Books               14982
Souvenir            14871
Name: quantity, dtype: int64
```

In [27]:
```
df.groupby("category")["total_amount"].sum().sort_values(ascending = False)
```

Out[27]:
```
category
Clothing           1.139968e+08
Shoes              6.655345e+07
Technology         5.786235e+07
Cosmetics          6.792863e+06
Toys               3.980426e+06
Food & Beverage    8.495351e+05
Books              8.345529e+05
Souvenir           6.358247e+05
Name: total_amount, dtype: float64
```

In [28]:
```
df.groupby("category")["price"].mean()
```

Out[28]:
```
category
Books                45.568621
Clothing            901.084021
Cosmetics           122.448626
Food & Beverage      15.671948
Shoes              1807.388568
Souvenir             34.894345
Technology         3156.935548
Toys                107.733185
Name: price, dtype: float64
```

In [29]:
```
df.groupby('payment_method')['total_amount'].sum().sort_values(ascending=False
```

Out[29]:
```
payment_method
Cash           1.128322e+08
Credit Card    8.807712e+07
Debit Card     5.059643e+07
Name: total_amount, dtype: float64
```

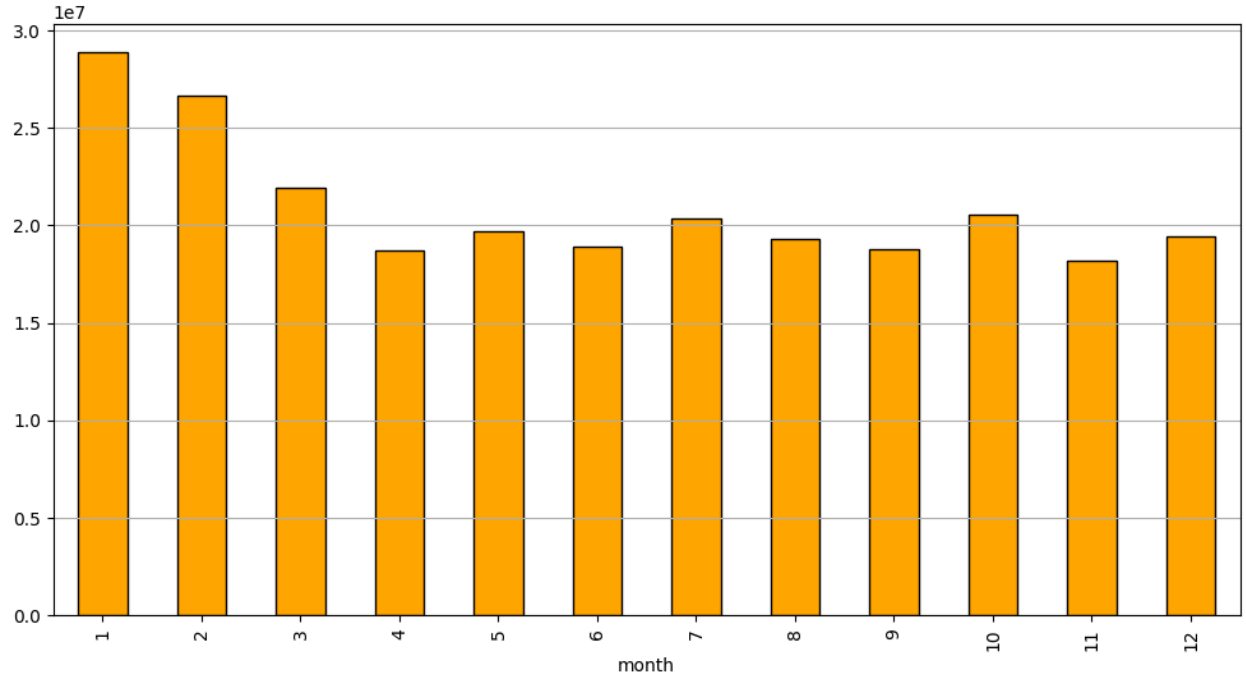
In [30]:
```
df.groupby('shopping_mall')['total_amount'].sum().sort_values(ascending=False)
```

Out[30]:
```
shopping_mall
Mall of Istanbul     50872481.68
Kanyon               50554231.10
Metrocity            37302787.33
Metropol AVM         25379913.19
Istinye Park         24618827.68
Zorlu Center         12901053.82
Cevahir AVM          12645138.20
Viaport Outlet       12521339.72
Emaar Square Mall    12406100.29
Forum Istanbul       12303921.24
Name: total_amount, dtype: float64
```

In [31]:
```
df.groupby('shopping_mall')['total_amount'].mean()
```

Out[31]:
```
shopping_mall
Cevahir AVM          2533.588099
Emaar Square Mall    2578.694718
Forum Istanbul       2487.148017
Istinye Park         2517.005181
Kanyon               2550.281547
Mall of Istanbul     2550.894132
Metrocity            2485.030133
Metropol AVM         2497.777108
Viaport Outlet       2548.095181
Zorlu Center         2542.079570
Name: total_amount, dtype: float64
```

In [32]:
```
plt.figure(figsize = (12,6))
df.groupby('month')['total_amount'].sum().plot(kind='bar', color = "orange", e
plt.grid(axis = "y")
plt.show()
```



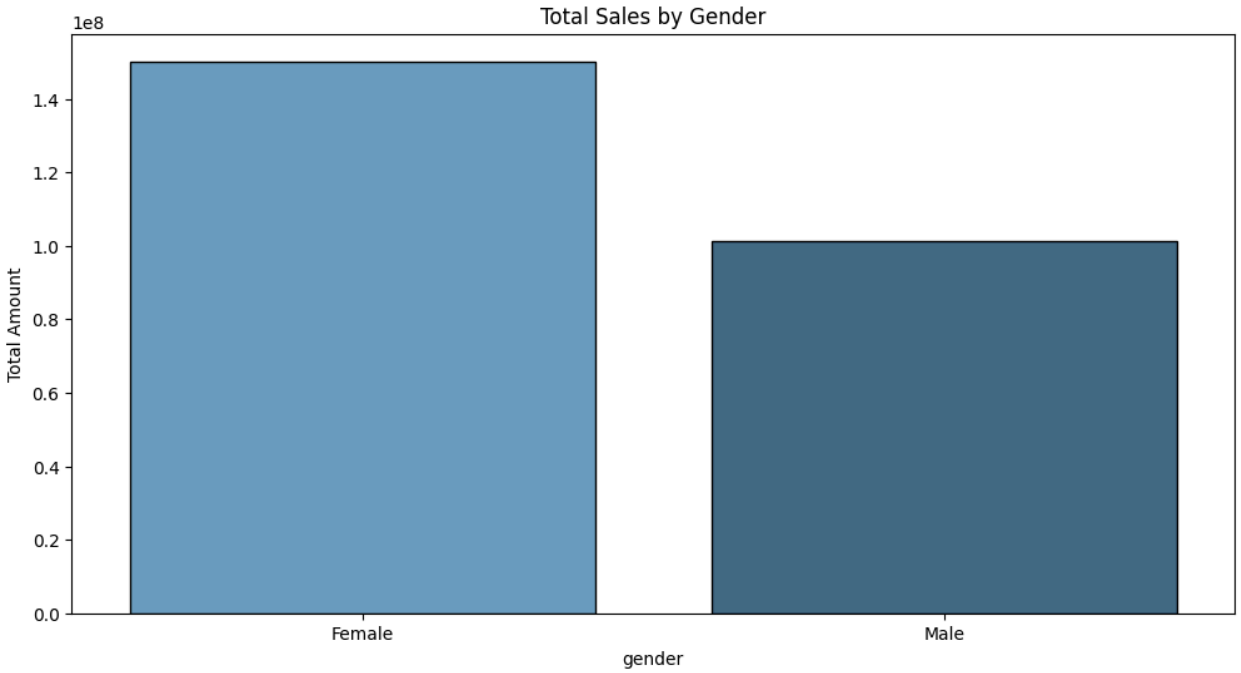
In [33]:
```
df['total_amount'].mean()
```

Out[33]:
```
2528.7892682264696
```

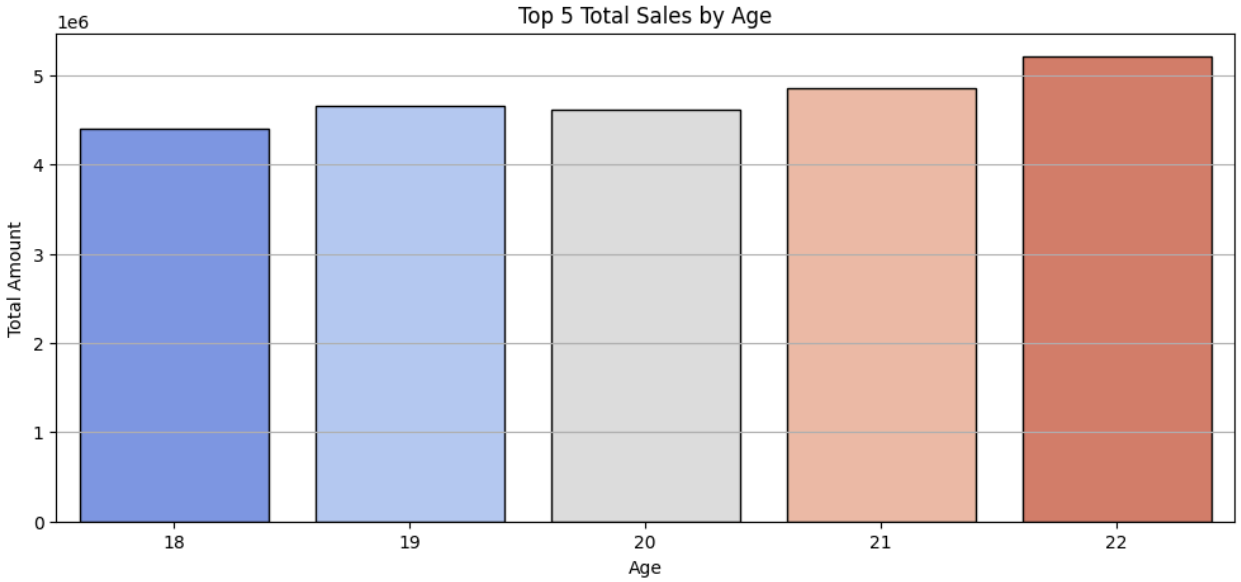
In [34]:
```
df.groupby(['gender', 'age'])['total_amount'].sum().sort_values(ascending=Fals
```

Out[34]:
```
gender  age
Female  37     3651099.74
        22     3184208.80
        40     3149142.77
        64     3142076.40
        24     3117351.54
               ...
Male    18     1775859.70
        23     1768891.40
        49     1755094.26
        56     1680684.36
        32     1650170.58
Name: total_amount, Length: 104, dtype: float64
```

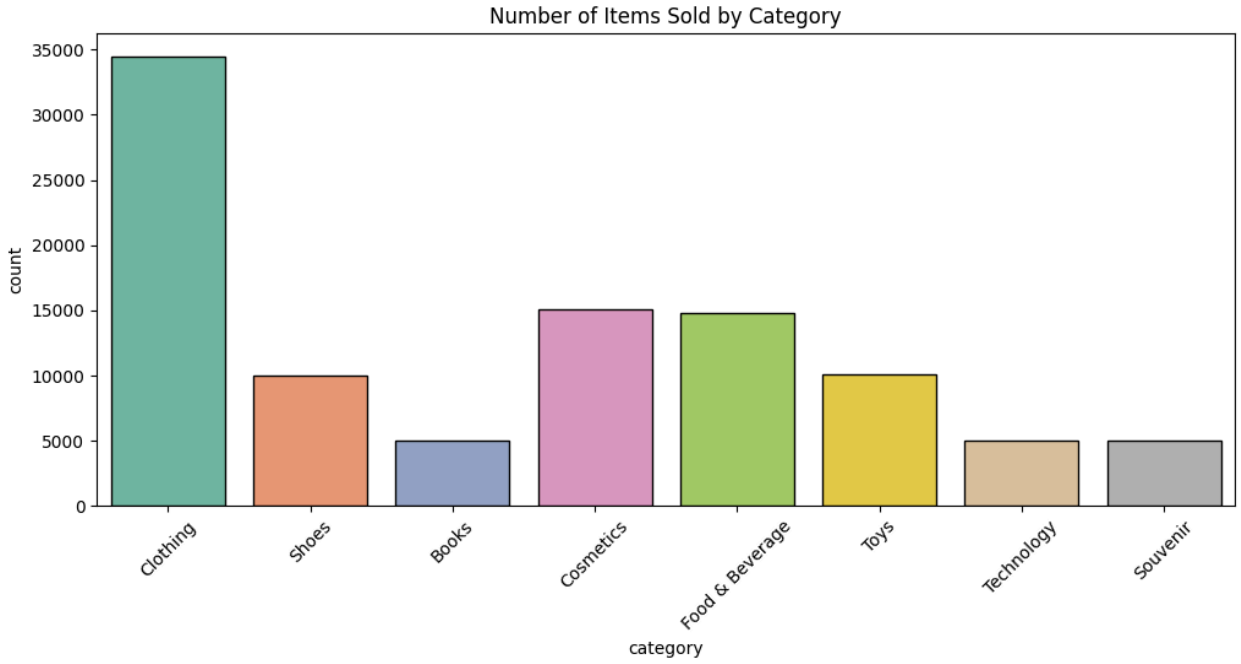
In [35]:
```
gender_sales = df.groupby('gender')['total_amount'].sum()
plt.figure(figsize=(12,6))
sns.barplot(x=gender_sales.index, y=gender_sales.values, palette='Blues_d', ec
plt.title('Total Sales by Gender')
plt.ylabel('Total Amount')
plt.show()
```



In [36]:
```
age_sales = df.groupby('age')['total_amount'].sum().sort_index().head()
plt.figure(figsize=(12,5))
sns.barplot(x=age_sales.index, y=age_sales.values, palette='coolwarm', edgecol
plt.title('Top 5 Total Sales by Age')
plt.xlabel('Age')
plt.ylabel('Total Amount')
plt.grid(axis = "y")
plt.show()
```

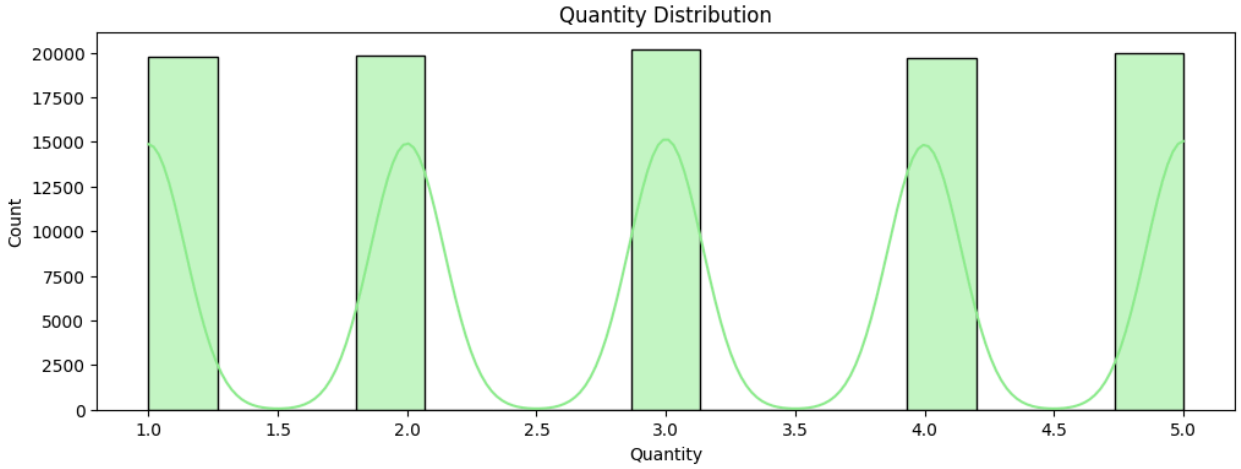


In [37]:
```
plt.figure(figsize=(12,5))
sns.countplot(data=df, x='category', palette='Set2', edgecolor='black')
plt.title('Number of Items Sold by Category')
plt.xticks(rotation=45)
plt.show()
```



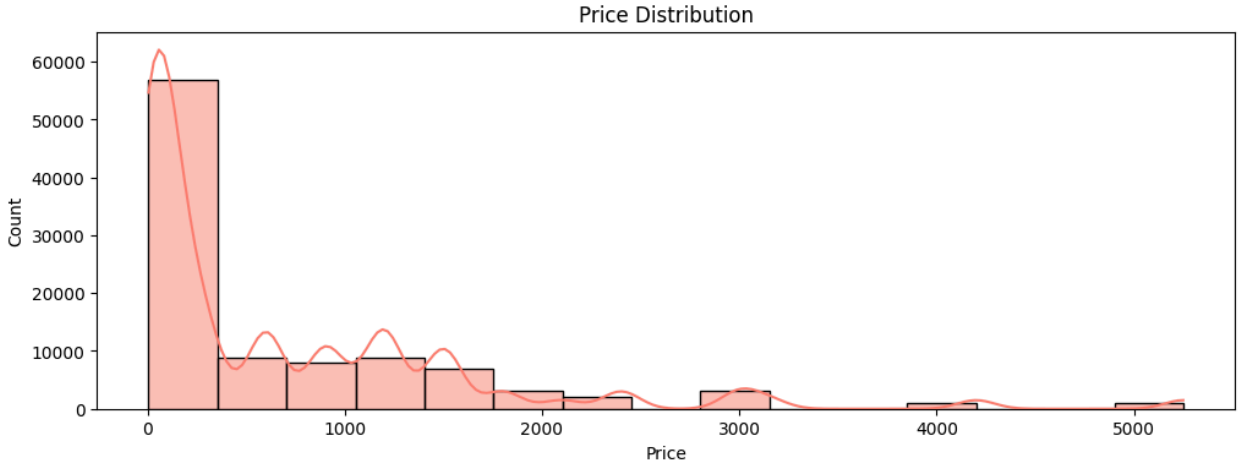
In [38]:
```
plt.figure(figsize=(12,4))
sns.histplot(df['quantity'], bins=15, kde=True, color='lightgreen', edgecolor=
plt.title('Quantity Distribution')
plt.xlabel('Quantity')
plt.ylabel('Count')
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

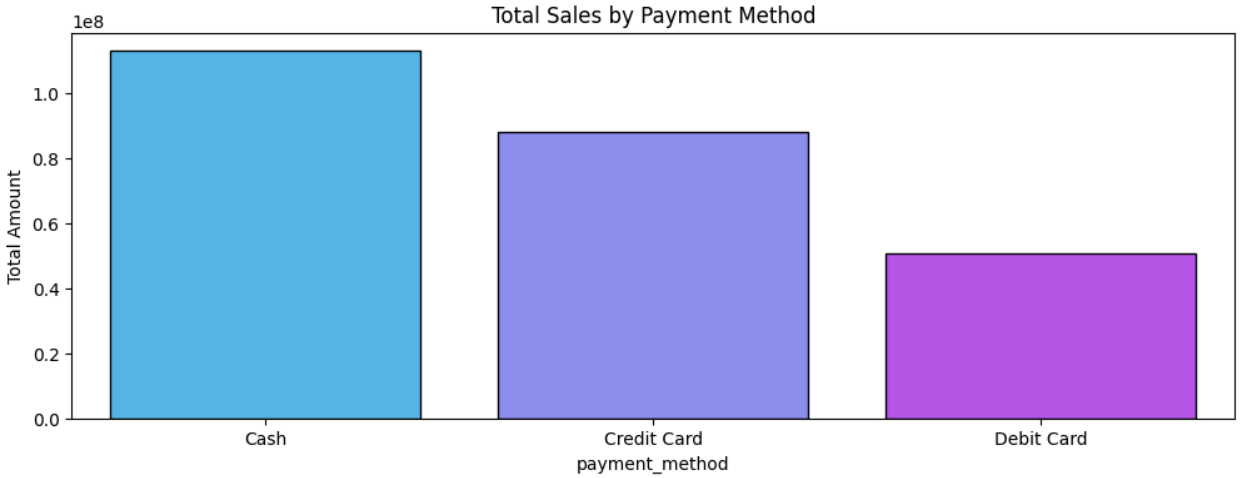


In [39]:
```
plt.figure(figsize=(12,4))
sns.histplot(df['price'], bins=15, kde=True, color='salmon', edgecolor='black'
plt.title('Price Distribution')
plt.xlabel('Price')
plt.ylabel('Count')
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

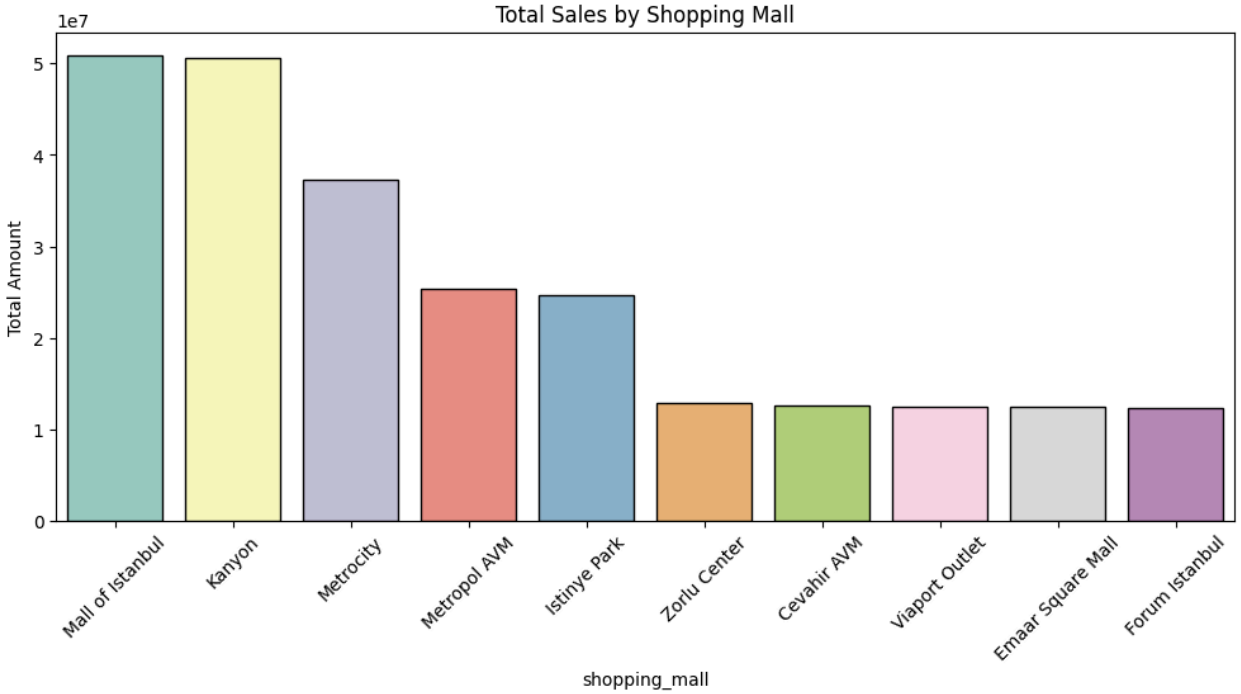


In [40]:
```
payment_sales = df.groupby('payment_method')['total_amount'].sum()
plt.figure(figsize=(12,4))
sns.barplot(x=payment_sales.index, y=payment_sales.values, palette='cool', edg
plt.title('Total Sales by Payment Method')
plt.ylabel('Total Amount')
plt.show()
```



In [41]:
```
plt.figure(figsize=(12,4))
sns.countplot(data=df, x='shopping_mall', palette='Set2', edgecolor='black')
plt.title('Transactions by Shopping Mall')
plt.xticks(rotation=45)
plt.show()
```

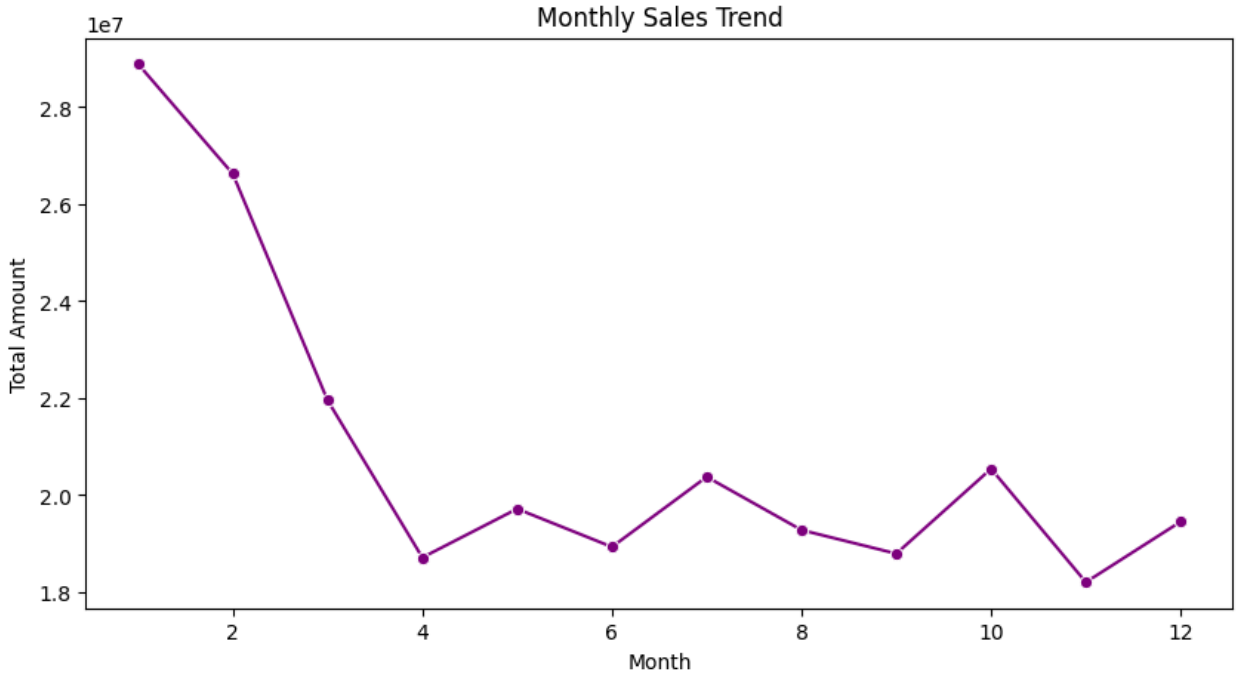
In [42]:
```
mall_sales = df.groupby('shopping_mall')['total_amount'].sum().sort_values(asc
plt.figure(figsize=(12,5))
sns.barplot(x=mall_sales.index, y=mall_sales.values, palette='Set3', edgecolor
plt.title('Total Sales by Shopping Mall')
plt.ylabel('Total Amount')
plt.xticks(rotation=45)
plt.show()
```



In [43]:
```
monthly_sales = df.groupby('month')['total_amount'].sum()
plt.figure(figsize=(10,5))
sns.lineplot(x=monthly_sales.index, y=monthly_sales.values, marker='o', color=
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
```

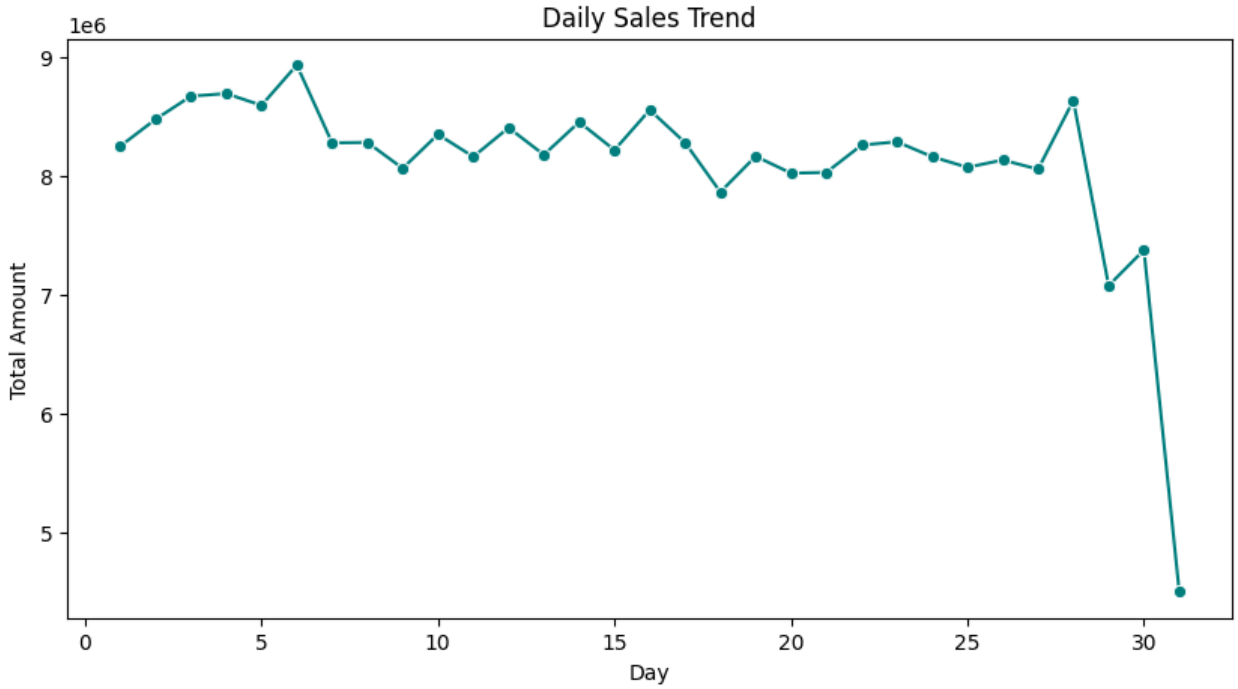
```
plt.ylabel('Total Amount')
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [44]:
```
daily_sales = df.groupby('day')['total_amount'].sum()
plt.figure(figsize=(10,5))
sns.lineplot(x=daily_sales.index, y=daily_sales.values, marker='o', color='tea
plt.title('Daily Sales Trend')
plt.xlabel('Day')
plt.ylabel('Total Amount')
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.11/dist-packages/seaborn/_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [ ]:

Name - Shashank Tiwari

Linkedin- https://www.linkedin.com/in/sshankt/

Date - 05-10-2025

In [ ]: