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

In [2]:
```
df = pd.read_csv('customer_shopping_data.csv')
```

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

Out[3]:

	invoice_no	customer_id	gender	age	category	quantity	price	payment_method	invoice_date	shopping_mall
0	I138884	C241288	Female	28	Clothing	5	1500.40	Credit Card	5/8/2022	Kanyon
1	I317333	C111565	Male	21	Shoes	3	1800.51	Debit Card	12/12/2021	Forum Istanbul
2	I127801	C266599	Male	20	Clothing	1	300.08	Cash	9/11/2021	Metrocity
3	I173702	C988172	Female	66	Shoes	5	3000.85	Credit Card	16/05/2021	Metropol AVM
4	I337046	C189076	Female	53	Books	4	60.60	Cash	24/10/2021	Kanyon

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

Out[4]:

	invoice_no	customer_id	gender	age	category	quantity	price	payment_method	invoice_date	shopping_mall
99452	I219422	C441542	Female	45	Souvenir	5	58.65	Credit Card	21/09/2022	Kanyon
99453	I325143	C569580	Male	27	Food & Beverage	2	10.46	Cash	22/09/2021	Forum Istanbul
99454	I824010	C103292	Male	63	Food & Beverage	2	10.46	Debit Card	28/03/2021	Metrocity
99455	I702964	C800631	Male	56	Technology	4	4200.00	Cash	16/03/2021	Istinye Park
99456	I232867	C273973	Female	36	Souvenir	3	35.19	Credit Card	15/10/2022	Mall of Istanbul

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
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 [6]:
```
df.describe()
```

Out[6]:

	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 [7]:
```
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 [8]:
```
df.shape
```

Out[8]:
```
(99457, 10)
```

In [9]:
```
df.columns
```

Out[9]:
```
Index(['invoice_no', 'customer_id', 'gender', 'age', 'category', 'quantity',
       'price', 'payment_method', 'invoice_date', 'shopping_mall'],
      dtype='object')
```

In [10]:
```
df['customer_id'].value_counts()
```

Out[10]:
```
C241288    1
C116138    1
C382765    1
C285074    1
C405356    1
          ..
C220083    1
C286933    1
C301304    1
C214184    1
C273973    1
Name: customer_id, Length: 99457, dtype: int64
```

In [11]:
```
df.duplicated().sum()
```

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

In [12]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [13]:
```
df.nunique()
```

Out[13]:
```
invoice_no        99457
customer_id       99457
gender                2
age                  52
category              8
quantity              5
price                40
payment_method        3
invoice_date        797
shopping_mall        10
dtype: int64
```

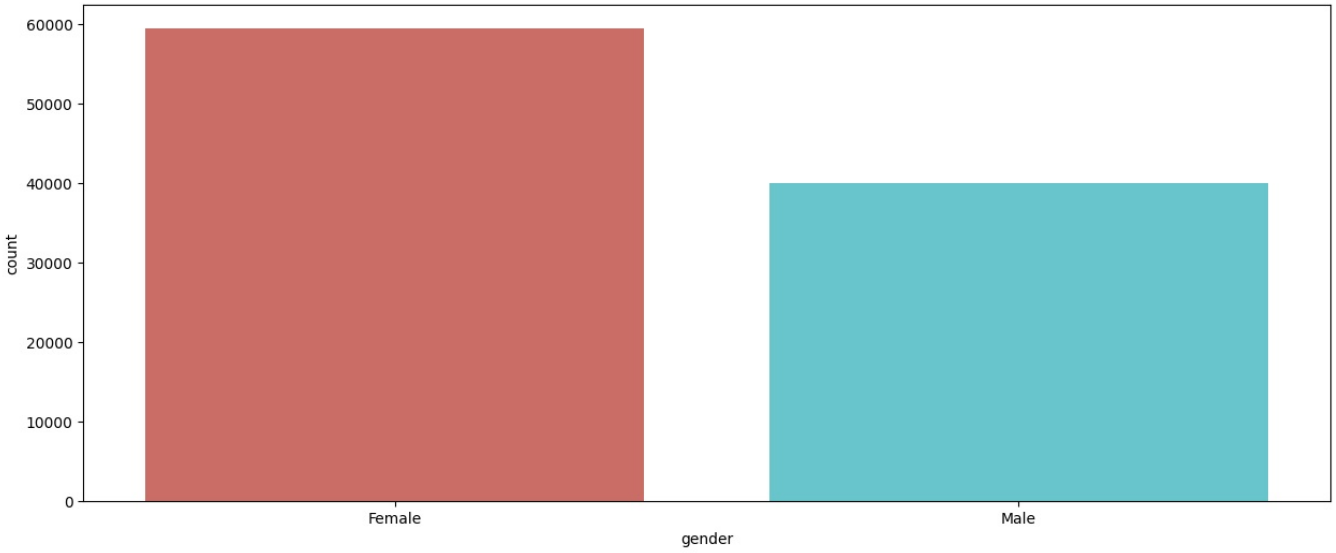
In [16]:
```
df['gender'].value_counts()
```

Out[16]:
```
Female    59482
Male      39975
Name: gender, dtype: int64
```

In [18]:
```
df['gender'].unique()
```

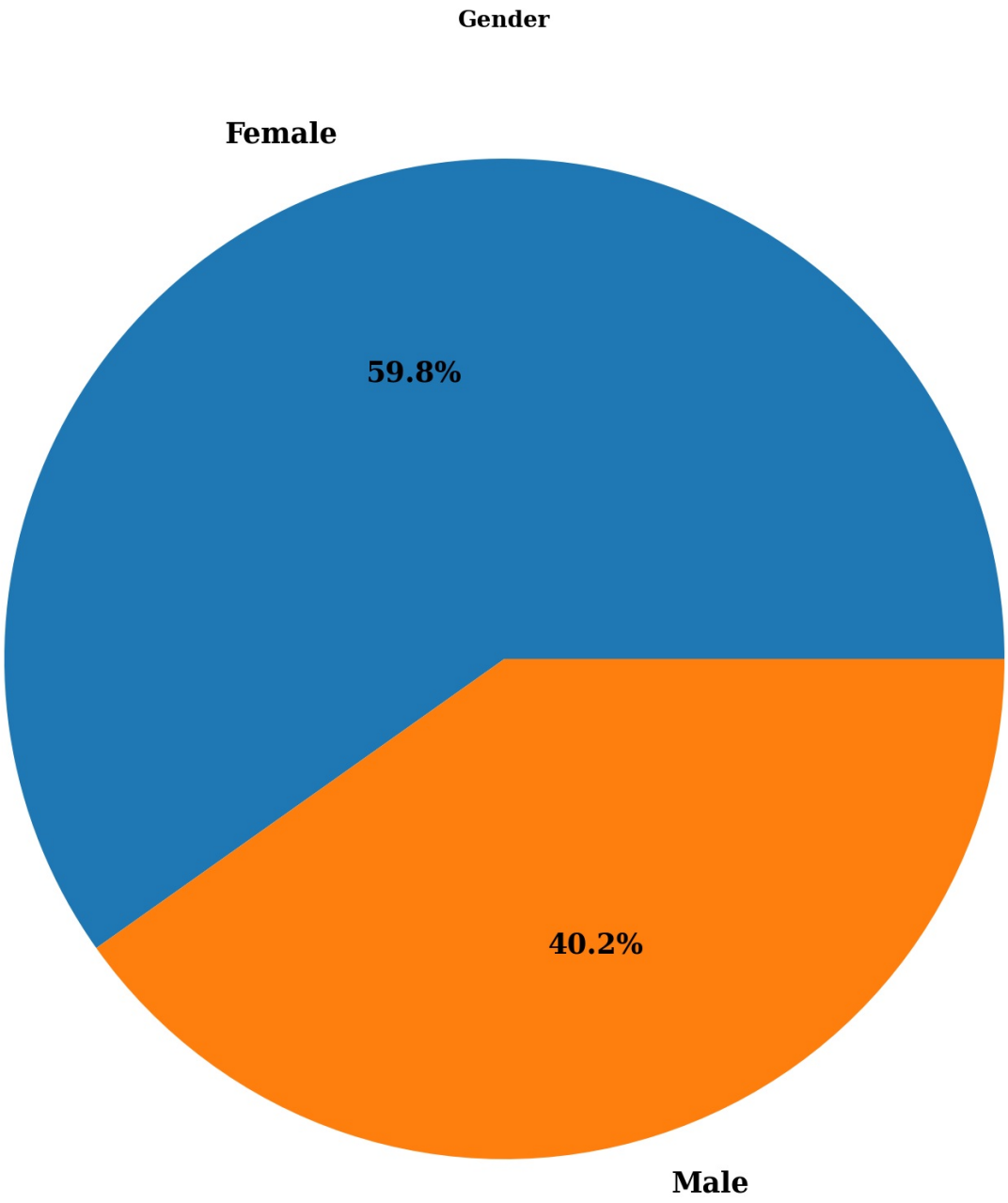
Out[18]:
```
array(['Female', 'Male'], dtype=object)
```

In [22]:
```
plt.figure(figsize=(15,6))
sns.countplot(x ='gender',data=df,palette = 'hls')
plt.show()
```



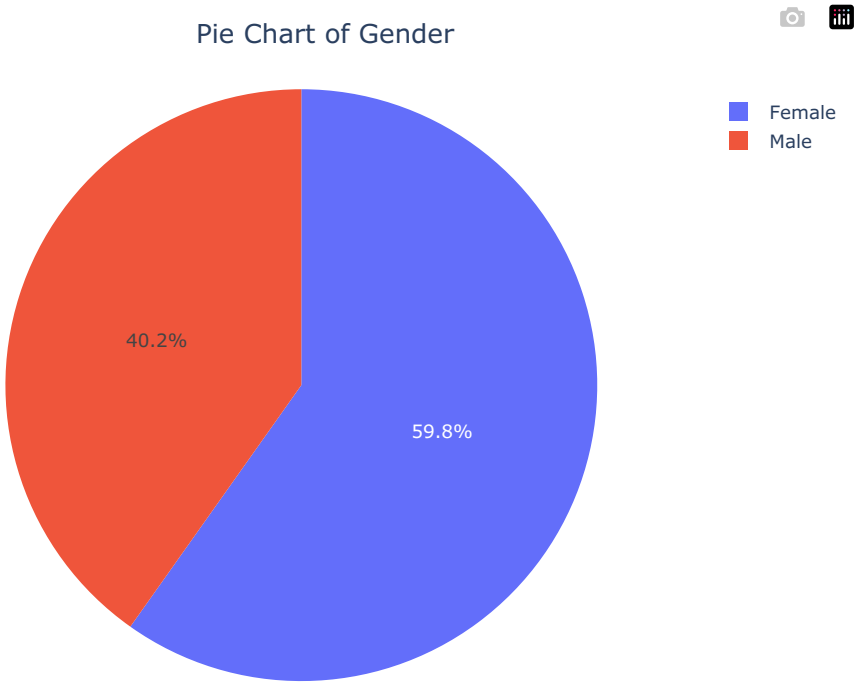
In [23]:
```
plt.figure(figsize=(30,20))
plt.pie(df['gender'].value_counts(), labels=df['gender'].value_counts().index, autopct='%1.1f%%', textprops={ '
                                        'color': 'black',
                                        'weight': 'bold',
                                        'family': 'serif' })
```

```
hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('Gender', size=20, **hfont)
plt.show()
```



In [24]:
```
import plotly.express as px
```

In [25]:
```
value_counts = df['gender'].value_counts()
fig = px.pie(names=value_counts.index, values=value_counts.values)
fig.update_layout(
    title='Pie Chart of Gender',
    title_x=0.5
)
fig.show()
```



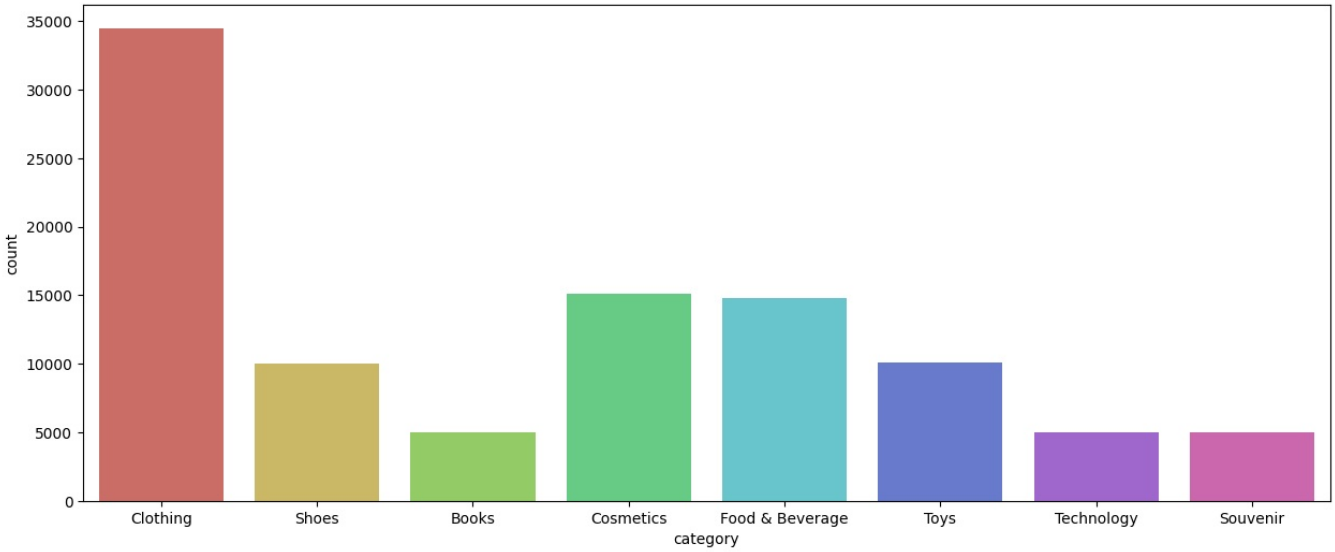
In [26]:
```
df['category'].unique()
```

Out[26]:
```
array(['Clothing', 'Shoes', 'Books', 'Cosmetics', 'Food & Beverage',
       'Toys', 'Technology', 'Souvenir'], dtype=object)
```

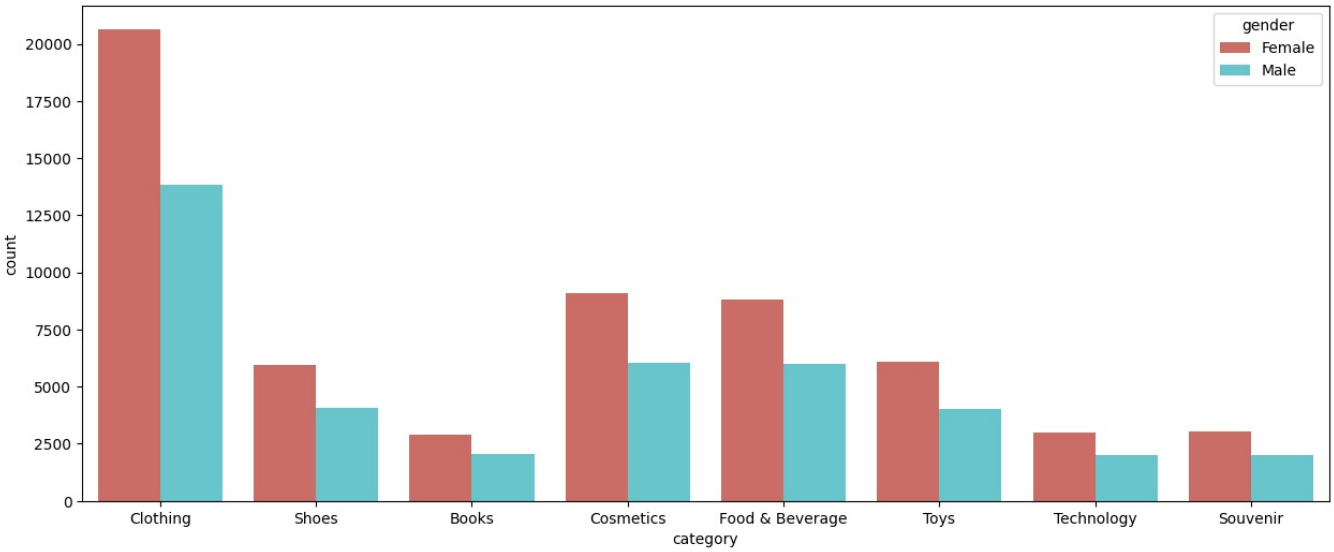
In [27]:
```
df['category'].value_counts()
```

Out[27]:
```
Clothing           34487
Cosmetics          15097
Food & Beverage    14776
Toys               10087
Shoes              10034
Souvenir            4999
Technology          4996
Books               4981
Name: category, dtype: int64
```

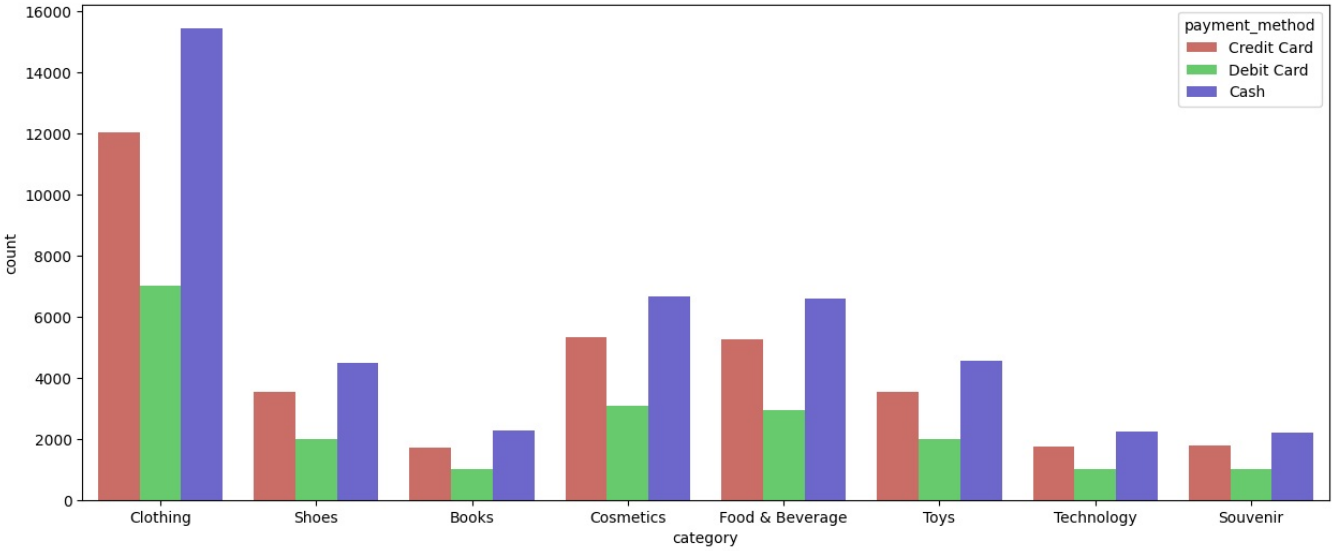
In [30]:
```
plt.figure(figsize=(15,6))
sns.countplot(x='category', data = df, palette = 'hls')
plt.show()
```



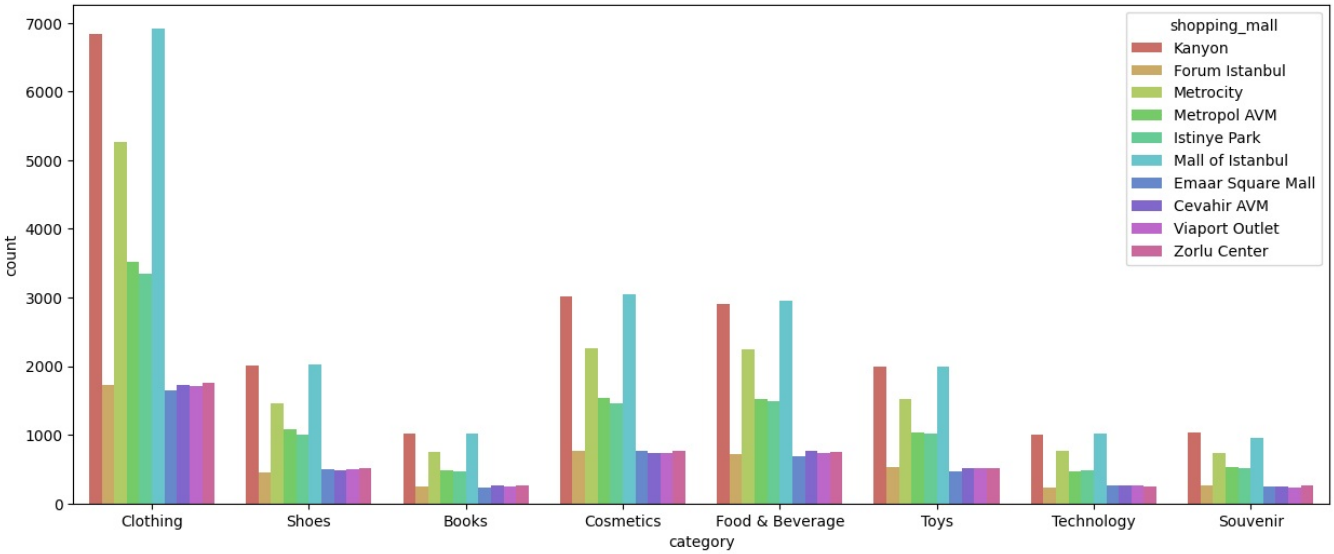
In [32]:
```
plt.figure(figsize=(15,6))
sns.countplot(x='category', hue = 'gender', data = df, palette = 'hls')
plt.show()
```



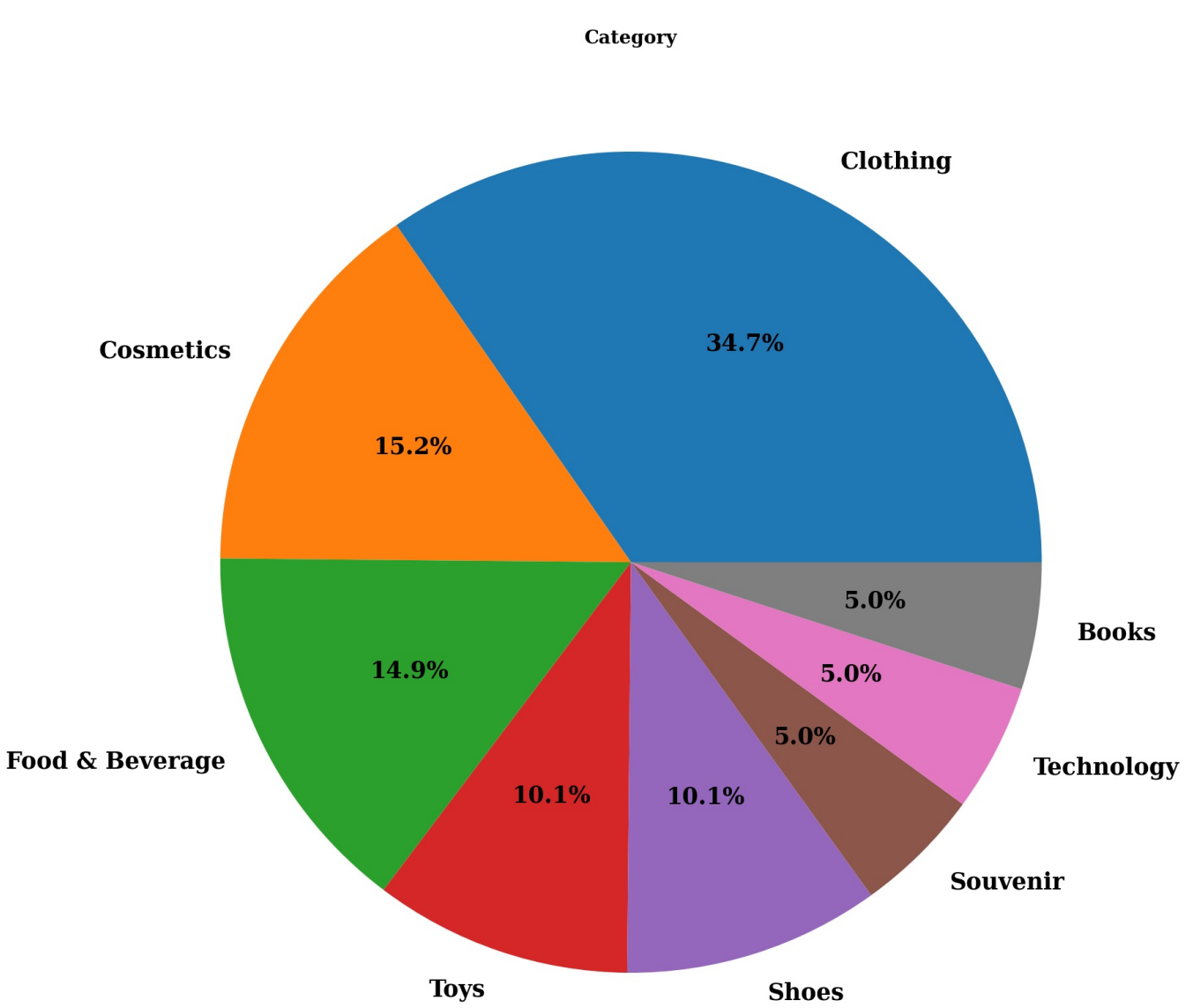
In [33]:
```
plt.figure(figsize=(15,6))
sns.countplot(x='category', hue = 'payment_method', data = df, palette = 'hls')
plt.show()
```



In [34]:
```
plt.figure(figsize=(15,6))
sns.countplot(x='category', hue = 'shopping_mall', data = df, palette = 'hls')
plt.show()
```



In [35]:
```
plt.figure(figsize=(30,20))
plt.pie(df['category'].value_counts(), labels=df['category'].value_counts().index, autopct='%1.1f%%', textprops
                                        'color': 'black',
                                        'weight': 'bold',
                                        'family': 'serif' })
hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('Category', size=20, **hfont)
plt.show()
```



In [36]:
```
value_counts = df['category'].value_counts()
fig = px.pie(names=value_counts.index, values=value_counts.values)
fig.update_layout(
    title='Pie Chart of Category',
    title_x=0.5
)
fig.show()
```

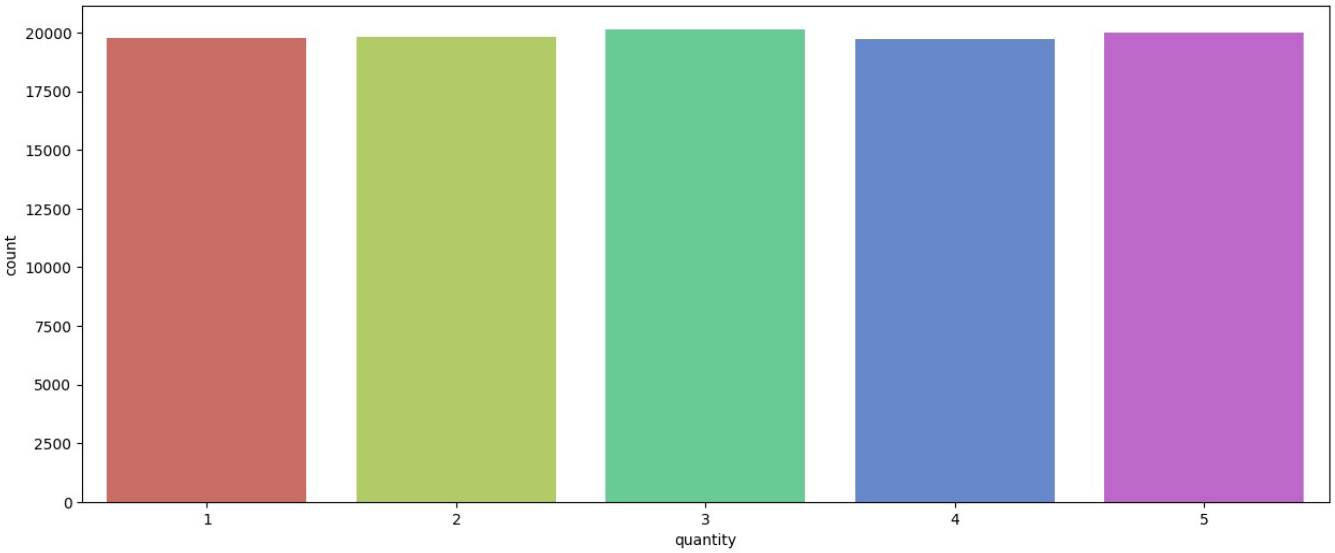
In [37]:
```
df['quantity'].unique()
```

Out[37]:
```
array([5, 3, 1, 4, 2], dtype=int64)
```

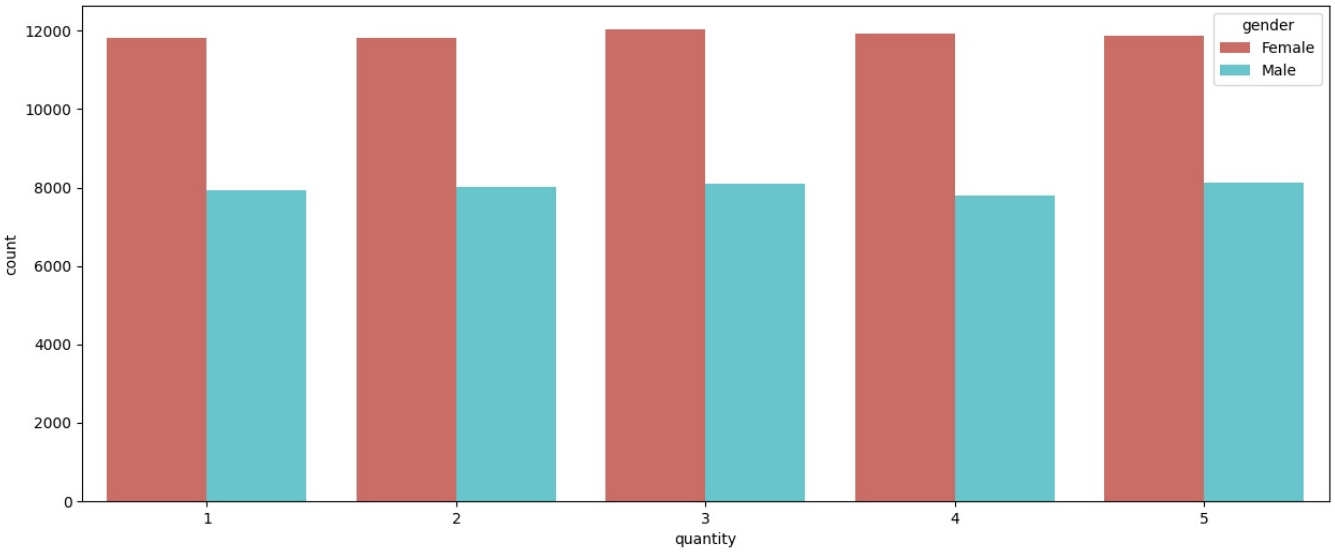
In [38]:
```
df['quantity'].value_counts()
```

Out[38]:
```
3    20149
5    19990
2    19828
1    19767
4    19723
Name: quantity, dtype: int64
```

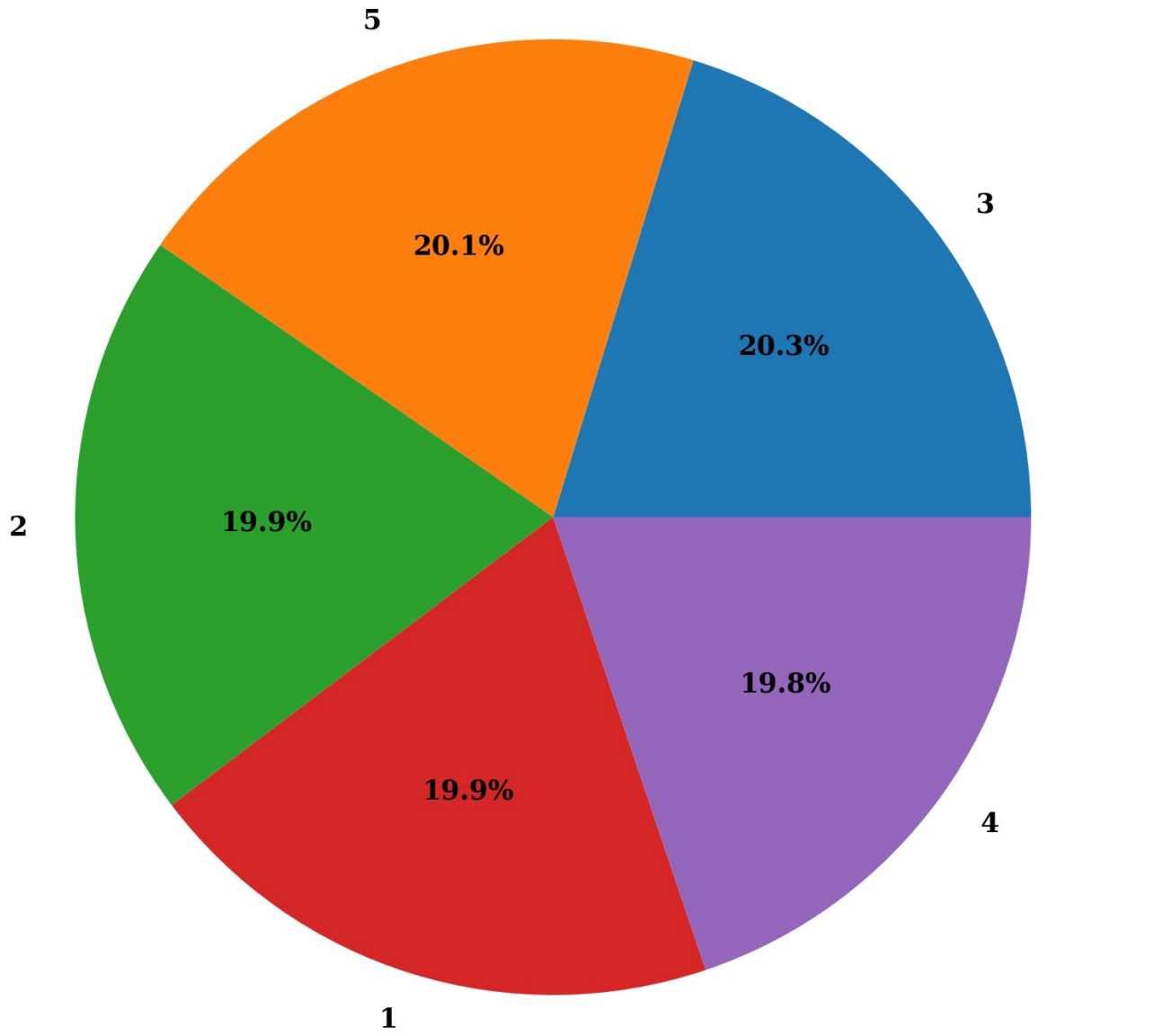
In [40]:
```
plt.figure(figsize=(15,6))
sns.countplot(x='quantity', data = df, palette = 'hls')
plt.show()
```



In [41]:
```
plt.figure(figsize=(15,6))
sns.countplot(x='quantity', hue = 'gender', data = df, palette = 'hls')
plt.show()
```



In [42]:
```
plt.figure(figsize=(30,20))
plt.pie(df['quantity'].value_counts(), labels=df['quantity'].value_counts().index, autopct='%1.1f%%', textprops
                                        'color': 'black',
                                        'weight': 'bold',
                                        'family': 'serif' })
hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('Quantity', size=20, **hfont)
plt.show()
```



In [43]:
```
value_counts = df['quantity'].value_counts()
fig = px.pie(names=value_counts.index, values=value_counts.values)
fig.update_layout(
    title='Pie Chart of Quantity',
    title_x=0.5
)
fig.show()
```

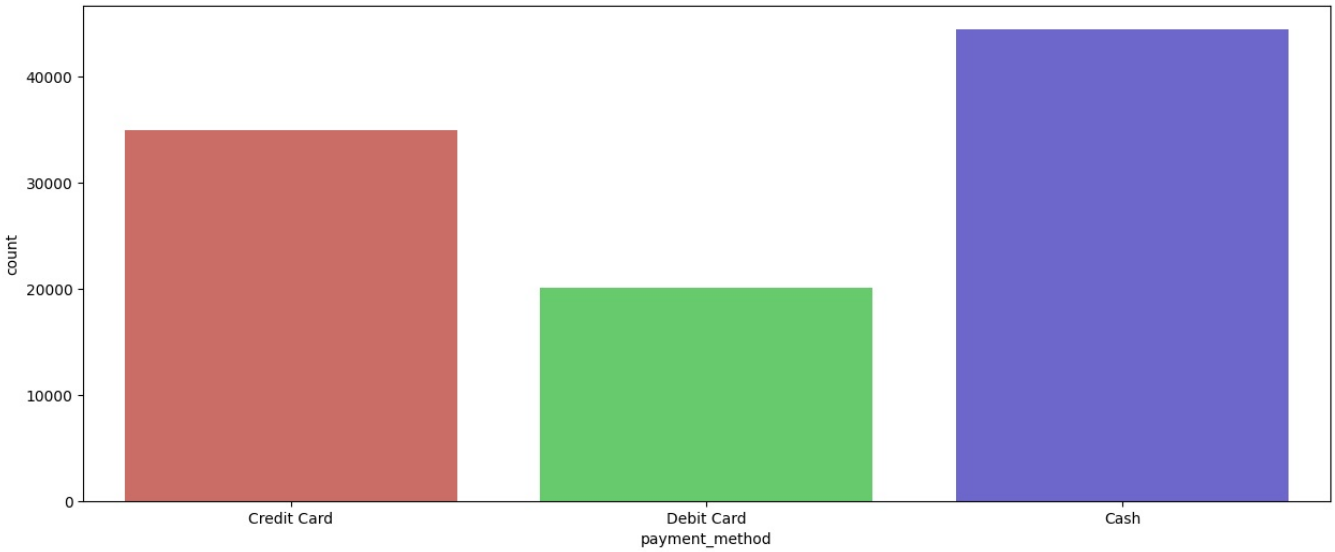
In [44]:
```
df['payment_method'].unique()
```

Out[44]:
```
array(['Credit Card', 'Debit Card', 'Cash'], dtype=object)
```

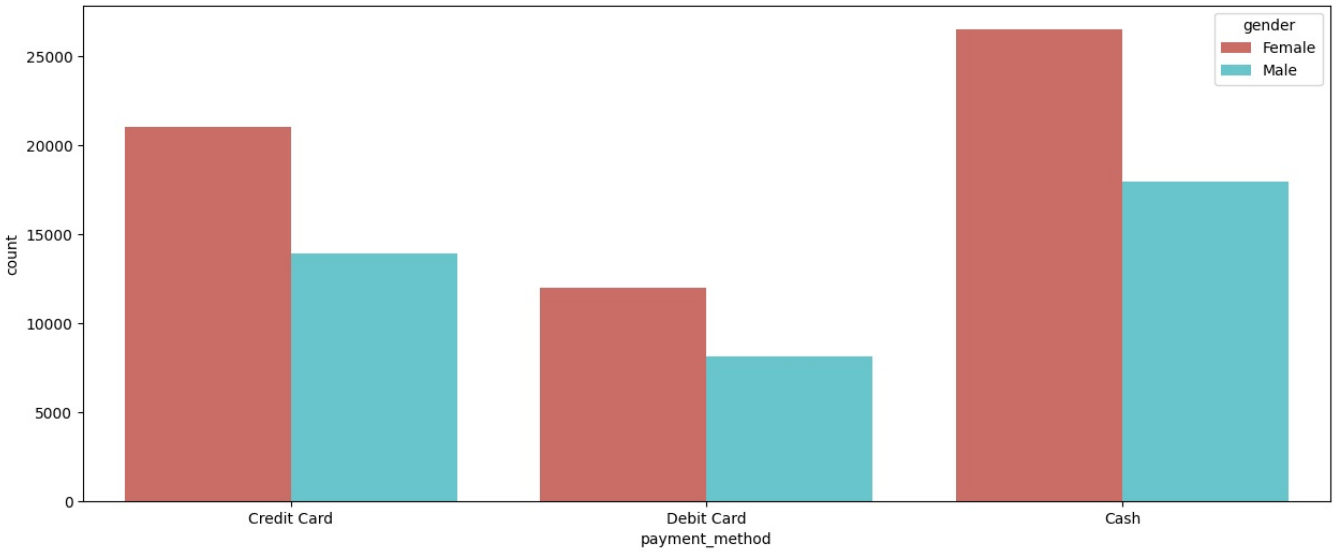
In [45]:
```
df['payment_method'].value_counts()
```

Out[45]:
```
Cash           44447
Credit Card    34931
Debit Card     20079
Name: payment_method, dtype: int64
```

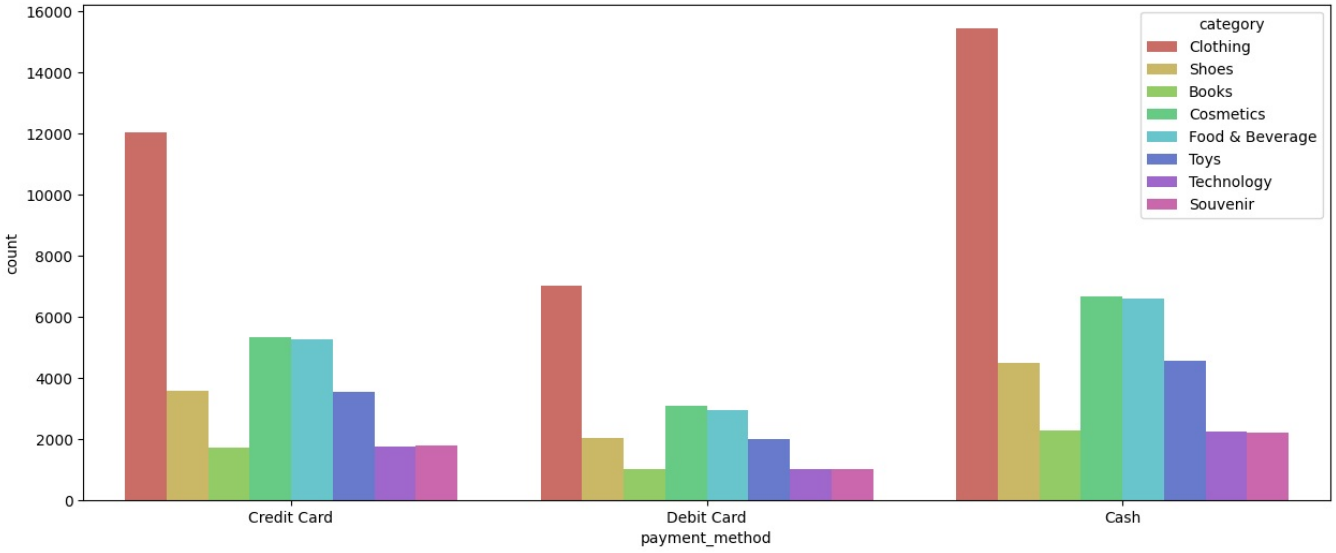
In [46]:
```
plt.figure(figsize=(15,6))
sns.countplot(x='payment_method',data=df,palette ='hls')
plt.show()
```



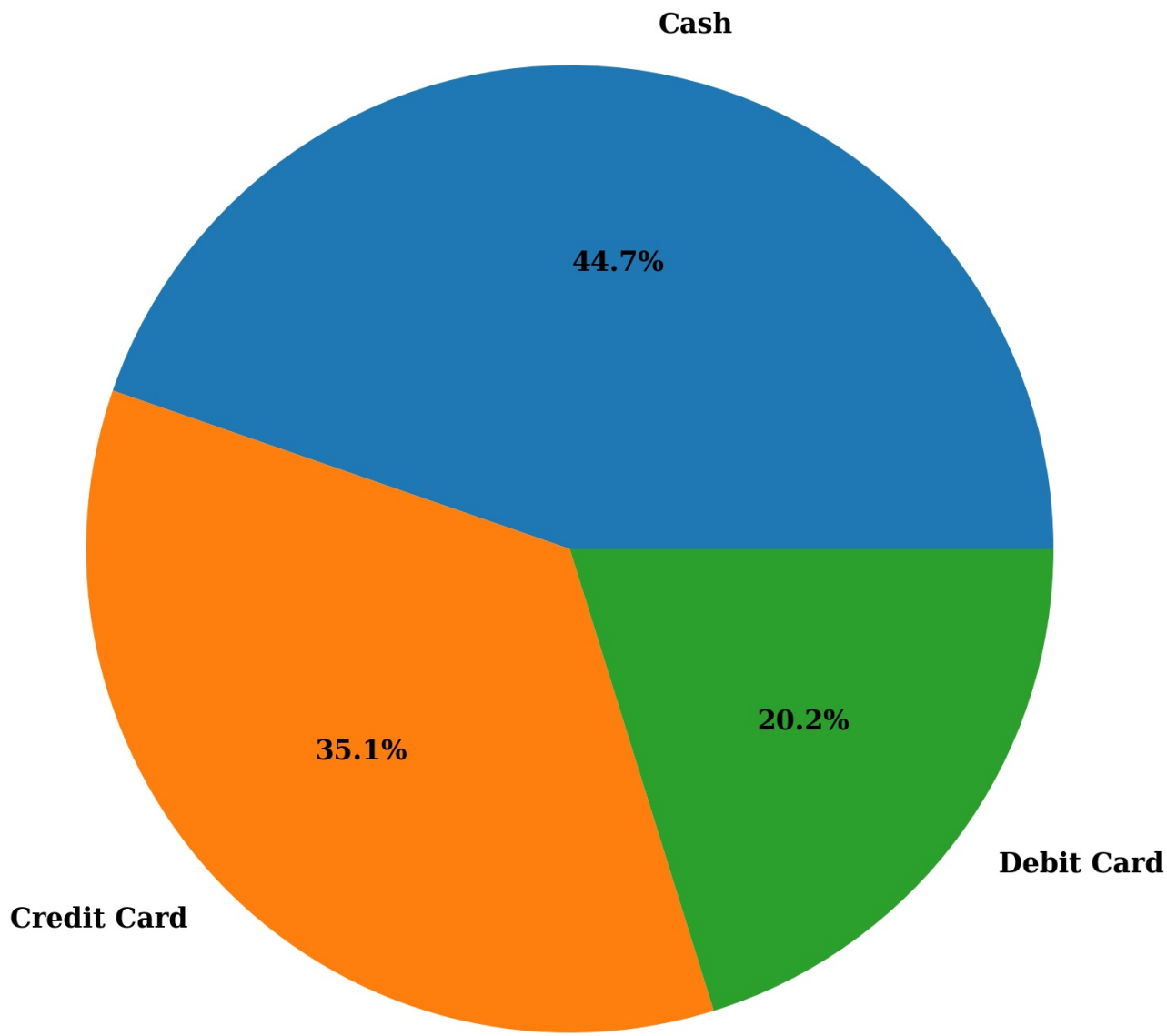
In [47]:
```
plt.figure(figsize=(15,6))
sns.countplot(x='payment_method', hue = 'gender', data = df, palette = 'hls')
plt.show()
```



In [48]:
```
plt.figure(figsize=(15,6))
sns.countplot(x='payment_method', hue = 'category', data = df, palette = 'hls')
plt.show()
```



In [49]:
```
plt.figure(figsize=(30,20))
plt.pie(df['payment_method'].value_counts(), labels=df['payment_method'].value_counts().index, autopct='%1.1f%%
                                        'color': 'black',
                                        'weight': 'bold',
                                        'family': 'serif' })
hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('Payment Method', size=20, **hfont)
plt.show()
```



In [50]:
```
value_counts = df['payment_method'].value_counts()
fig = px.pie(names=value_counts.index, values=value_counts.values)
fig.update_layout(
    title='Pie Chart of Payment Method',
    title_x=0.5
)
fig.show()
```

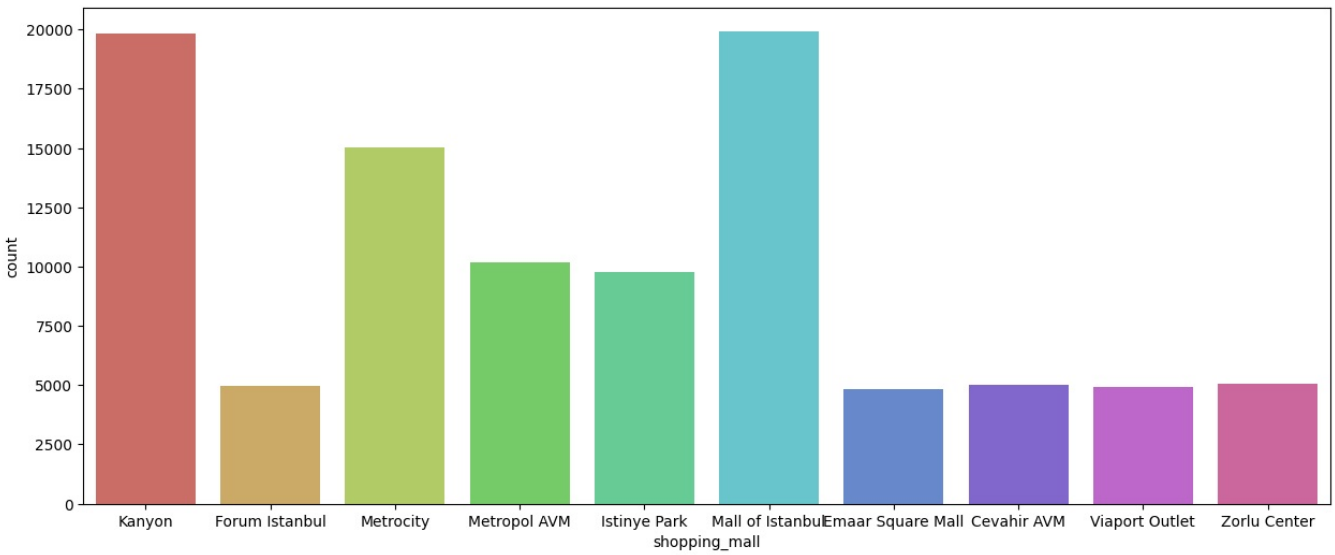
In [51]:
```
df['shopping_mall'].unique()
```

Out[51]:
```
array(['Kanyon', 'Forum Istanbul', 'Metrocity', 'Metropol AVM',
       'Istinye Park', 'Mall of Istanbul', 'Emaar Square Mall',
       'Cevahir AVM', 'Viaport Outlet', 'Zorlu Center'], dtype=object)
```

In [52]:
```
df['shopping_mall'].value_counts()
```

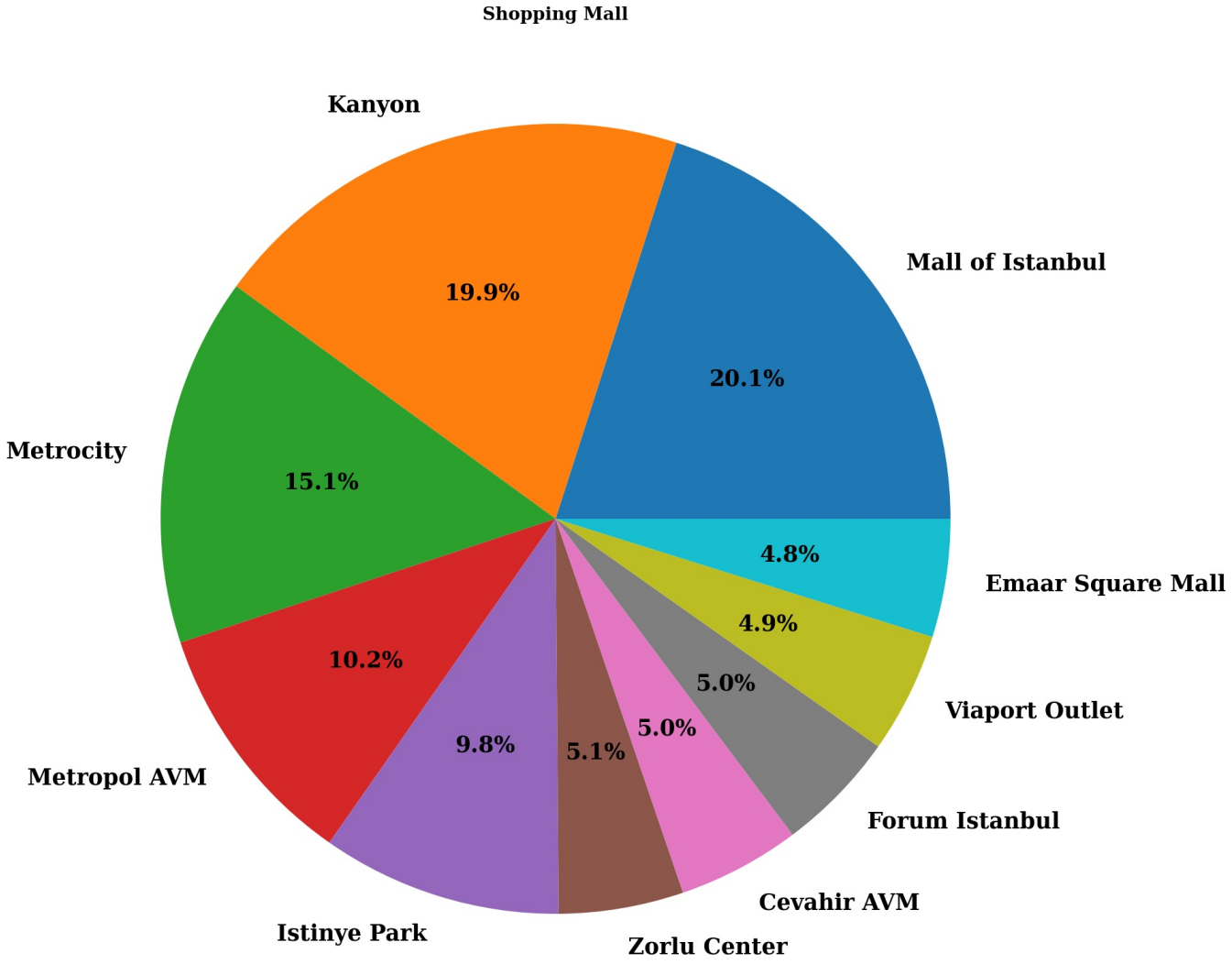
Out[52]:
```
Mall of Istanbul     19943
Kanyon               19823
Metrocity            15011
Metropol AVM         10161
Istinye Park          9781
Zorlu Center          5075
Cevahir AVM           4991
Forum Istanbul        4947
Viaport Outlet        4914
Emaar Square Mall     4811
Name: shopping_mall, dtype: int64
```

In [53]:
```
plt.figure(figsize=(15,6))
sns.countplot(x ='shopping_mall', data = df, palette = 'hls')
plt.show()
```



In [54]:
```
plt.figure(figsize=(30,20))
plt.pie(df['shopping_mall'].value_counts(), labels=df['shopping_mall'].value_counts().index, autopct='%1.1f%%',
                                          'color': 'black',
                                          'weight': 'bold',
                                          'family': 'serif' })
hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('Shopping Mall', size=20, **hfont)
```

```
plt.show()
```

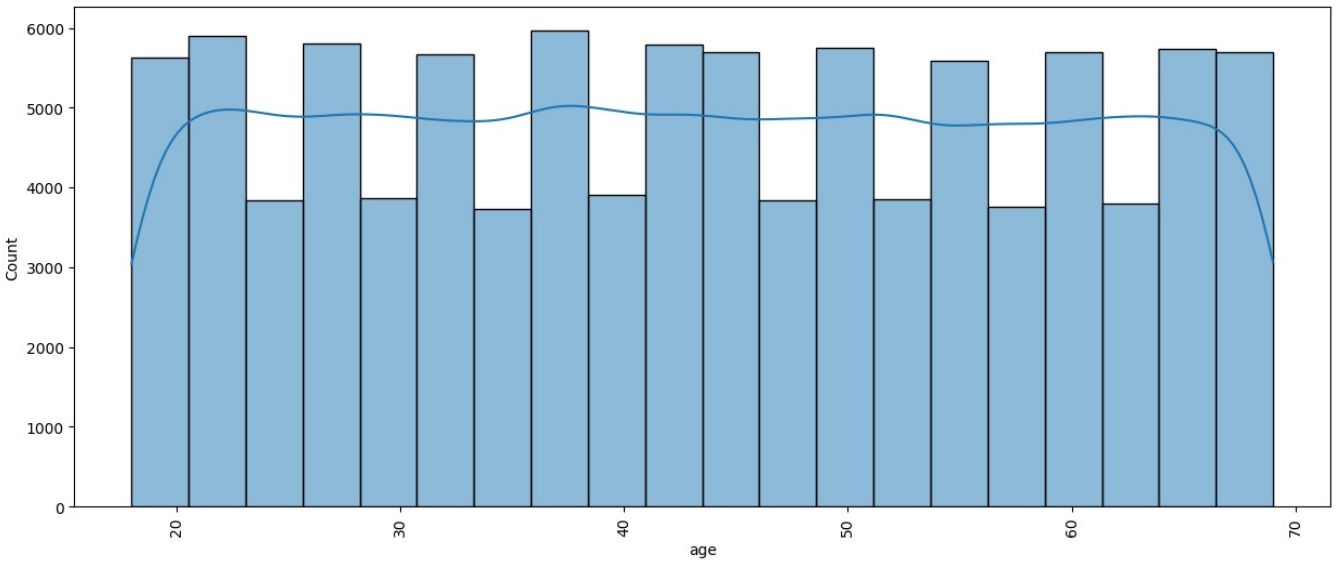


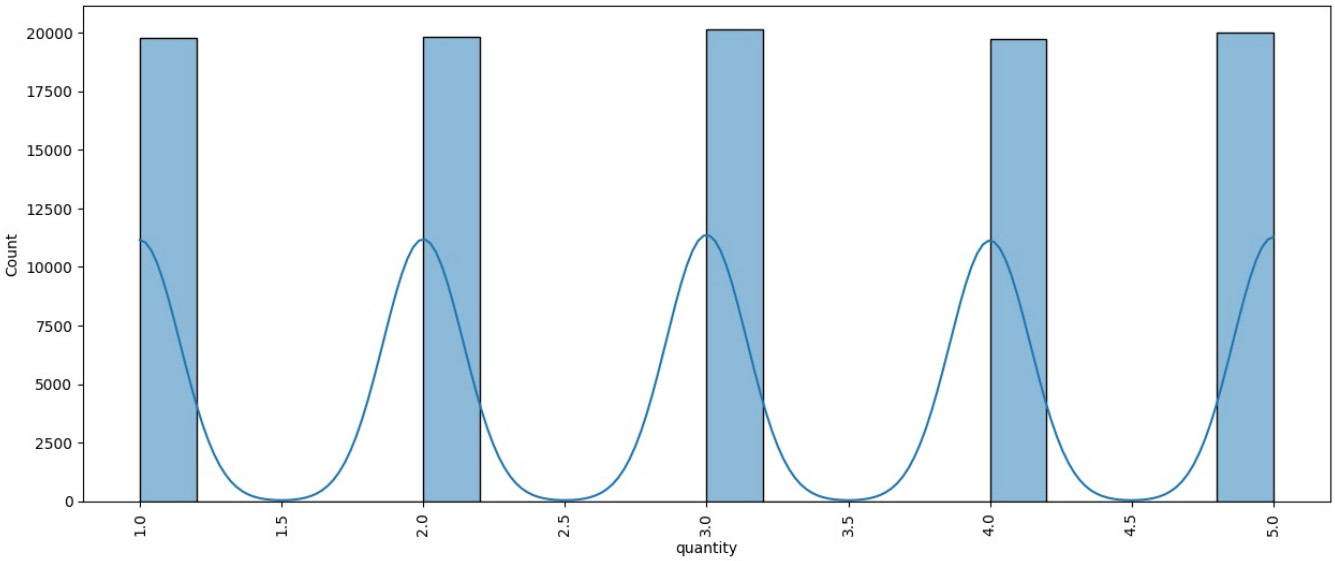
In [55]:
```
value_counts = df['shopping_mall'].value_counts()
fig = px.pie(names=value_counts.index, values=value_counts.values)
fig.update_layout(
    title='Pie Chart of Shopping Mall',
    title_x=0.5
)
fig.show()
```

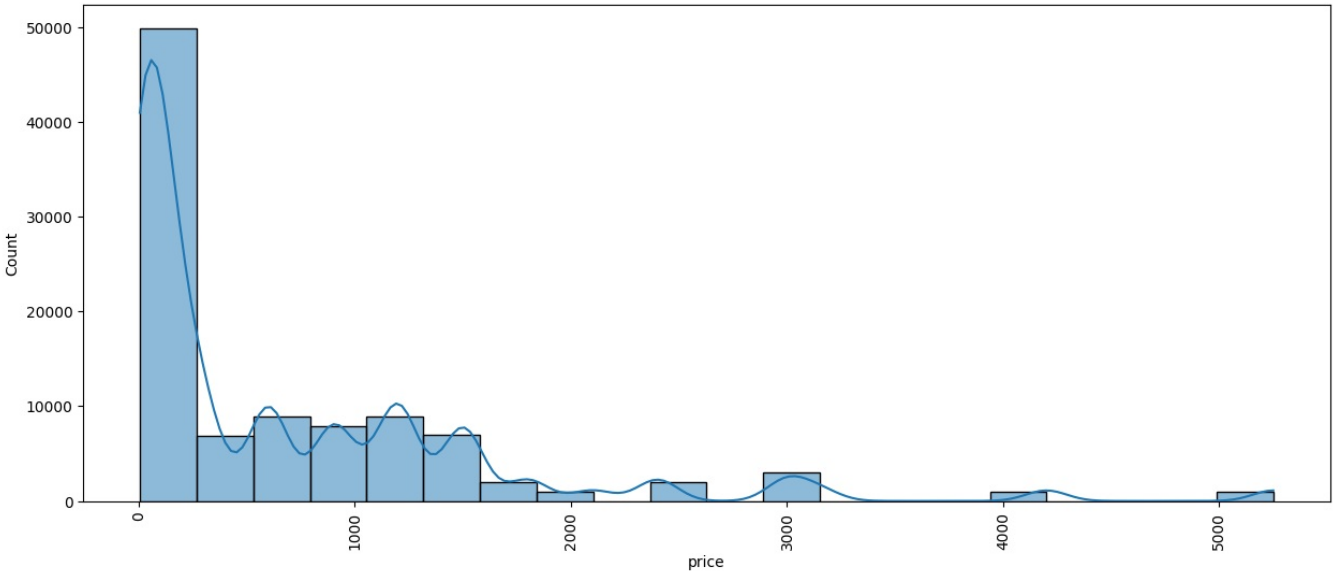
In [56]:
```
numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
object_cols = df.select_dtypes(include=['object']).columns.tolist()

print("Numerical columns: ", numerical_cols)
print("Object columns: ", object_cols)
```

```
Numerical columns:  ['age', 'quantity', 'price']
Object columns:  ['invoice_no', 'customer_id', 'gender', 'category', 'payment_method', 'invoice_date', 'shoppin
g_mall']
```

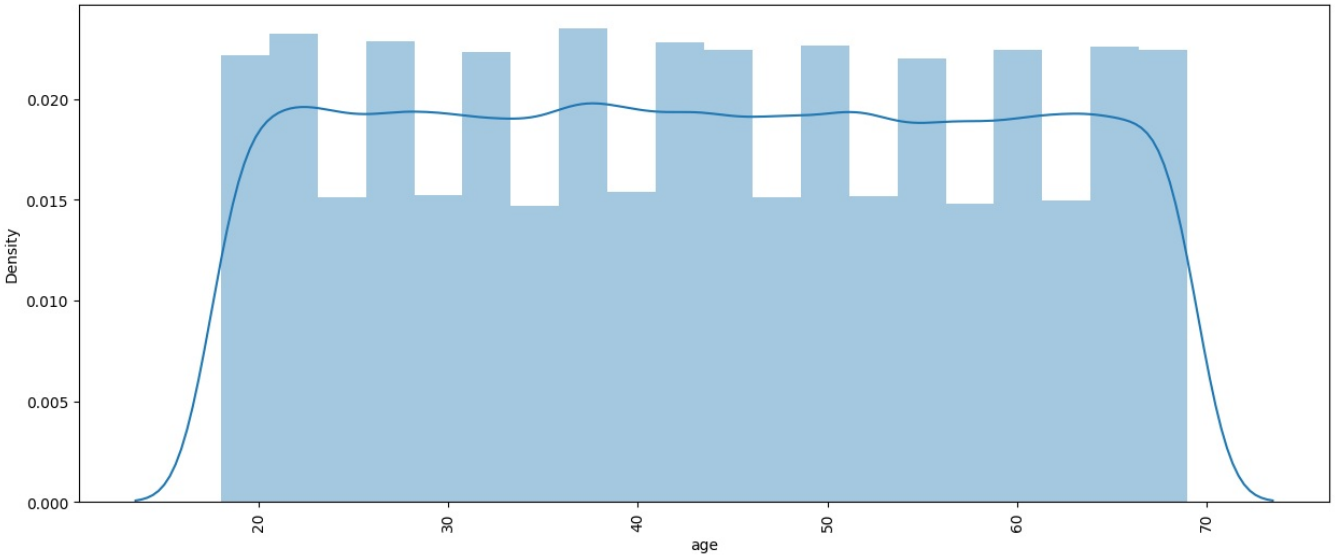
In [57]:
```
for i in numerical_cols:
    plt.figure(figsize=(15,6))
    sns.histplot(df[i], kde = True, bins = 20, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```

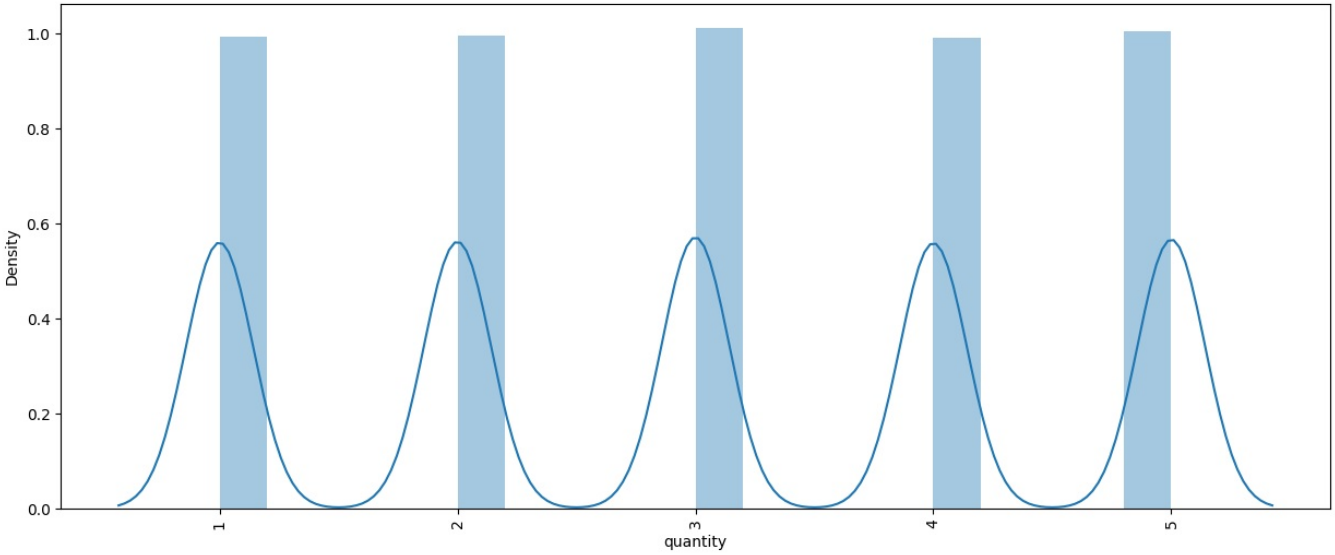


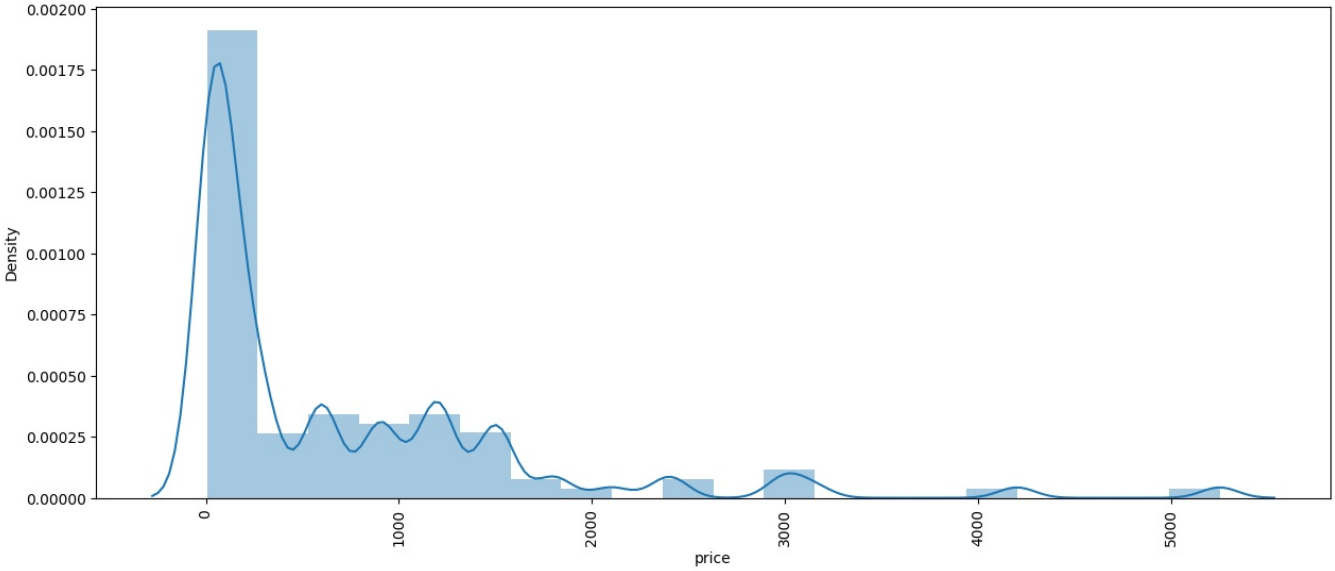




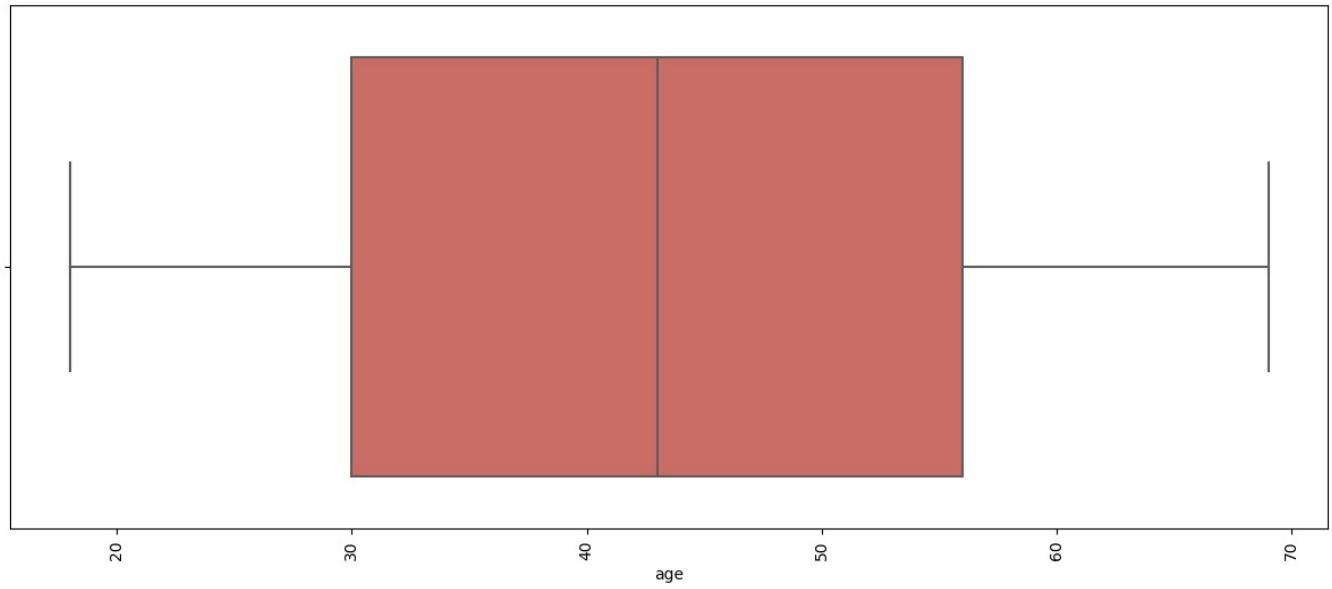
In [59]:
```
for i in numerical_cols:
    plt.figure(figsize=(15,6))
    sns.distplot(df[i], kde = True, bins = 20)
    plt.xticks(rotation = 90)
    plt.show()
```

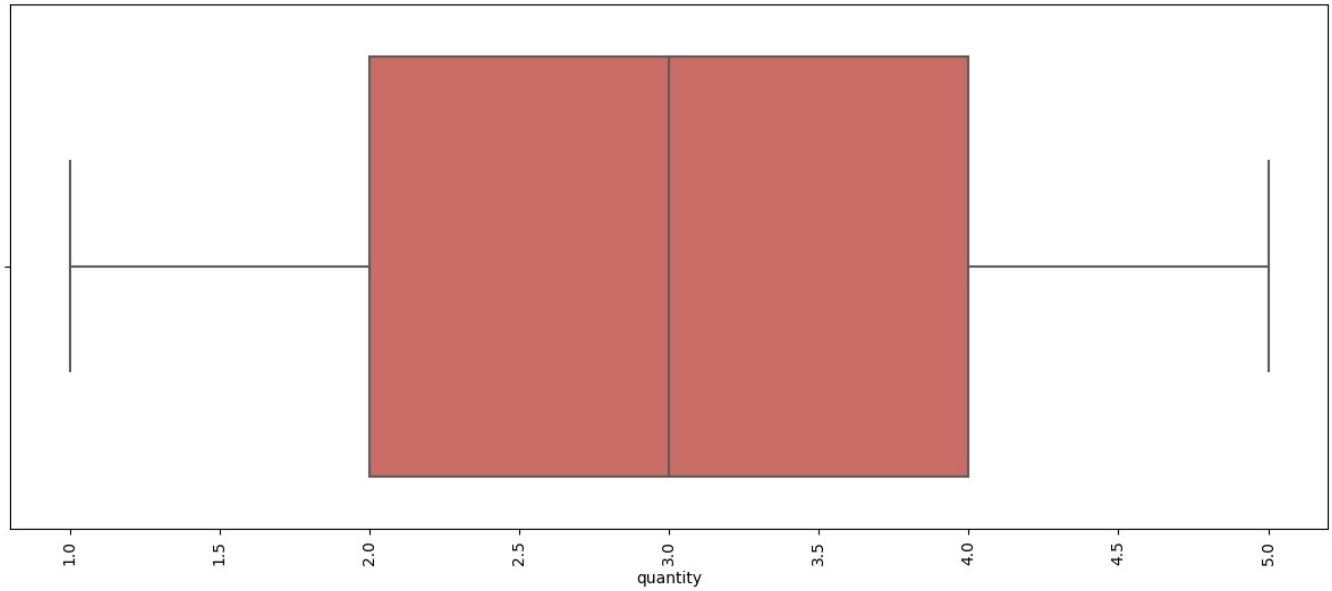


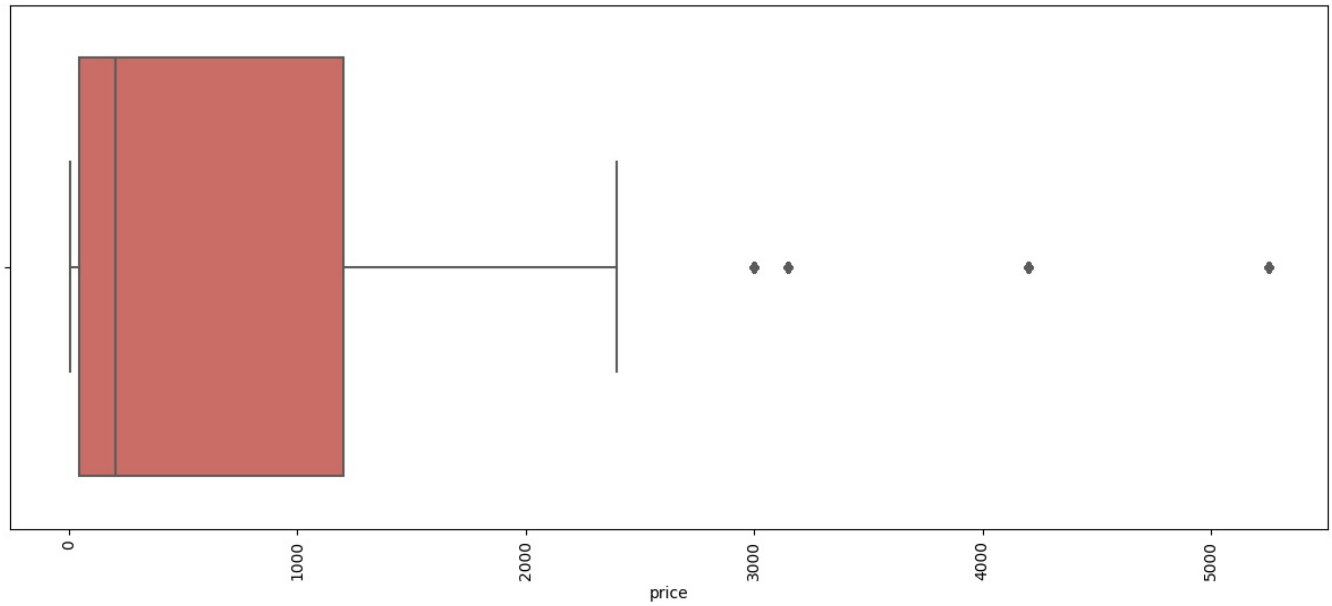




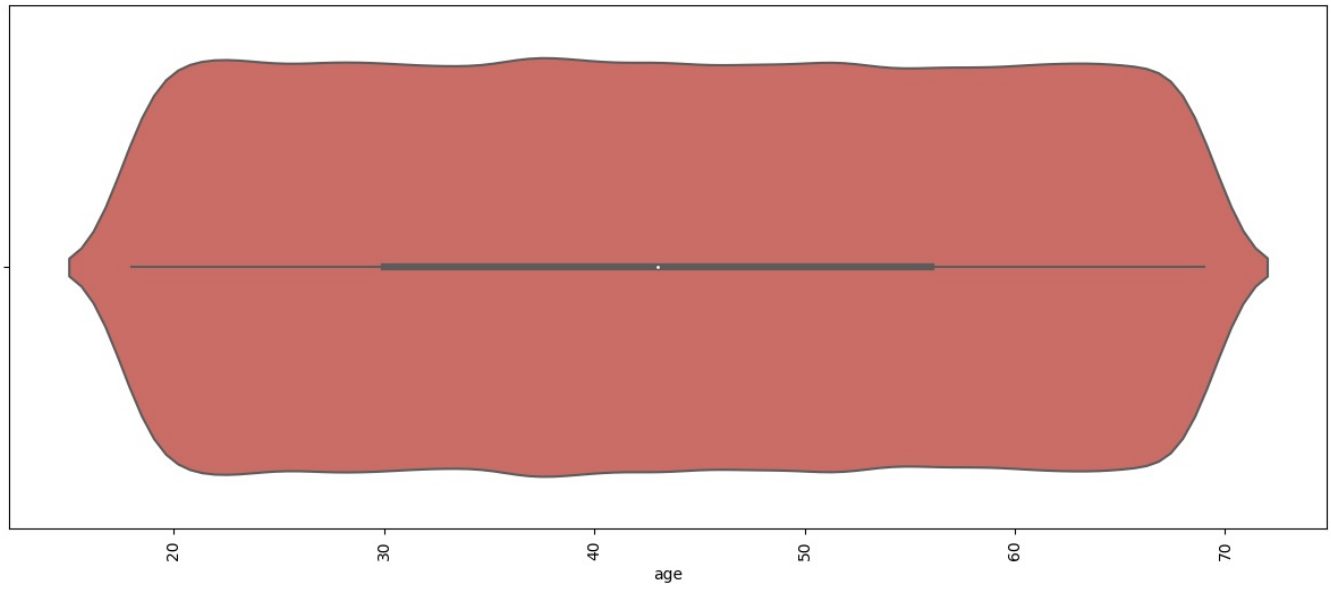
In [61]:
```
for i in numerical_cols:
    plt.figure(figsize=(15,6))
    sns.boxplot(x=i, data=df, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```

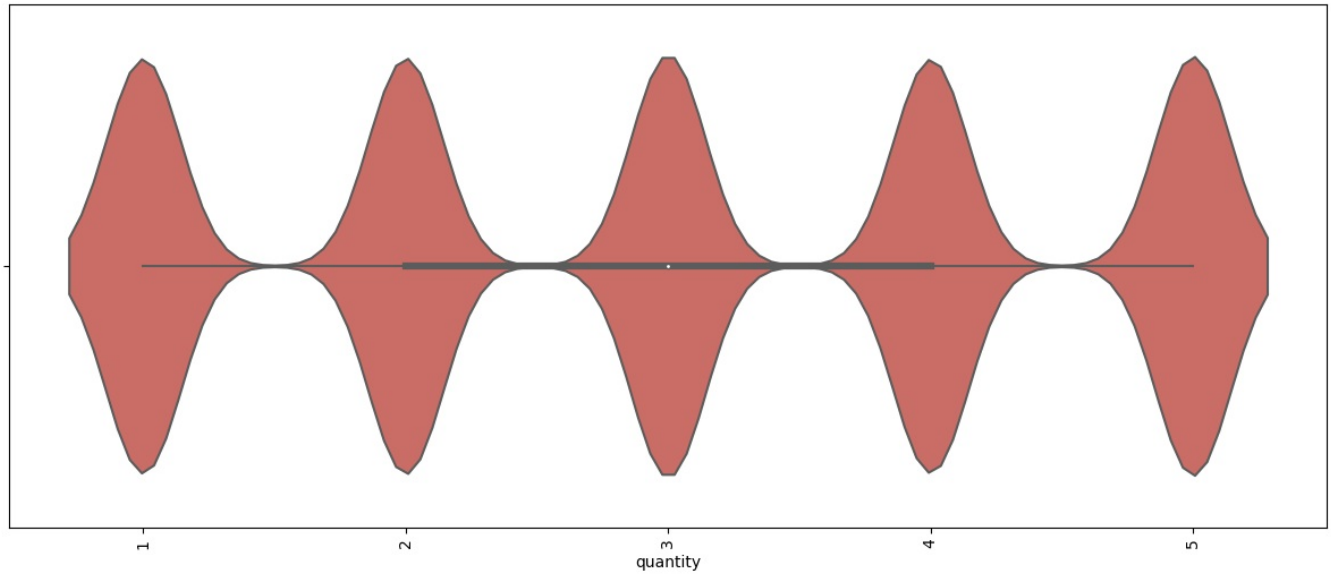


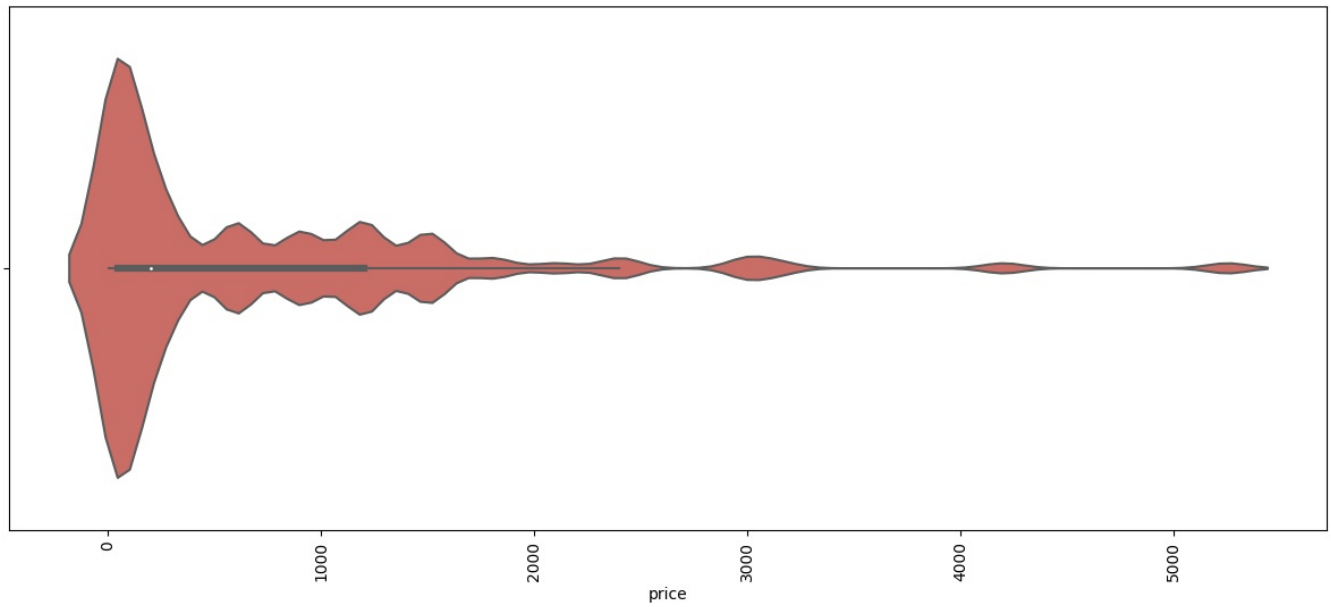




In [62]:
```
for i in numerical_cols:
    plt.figure(figsize=(15,6))
    sns.violinplot(x=i, data=df, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```



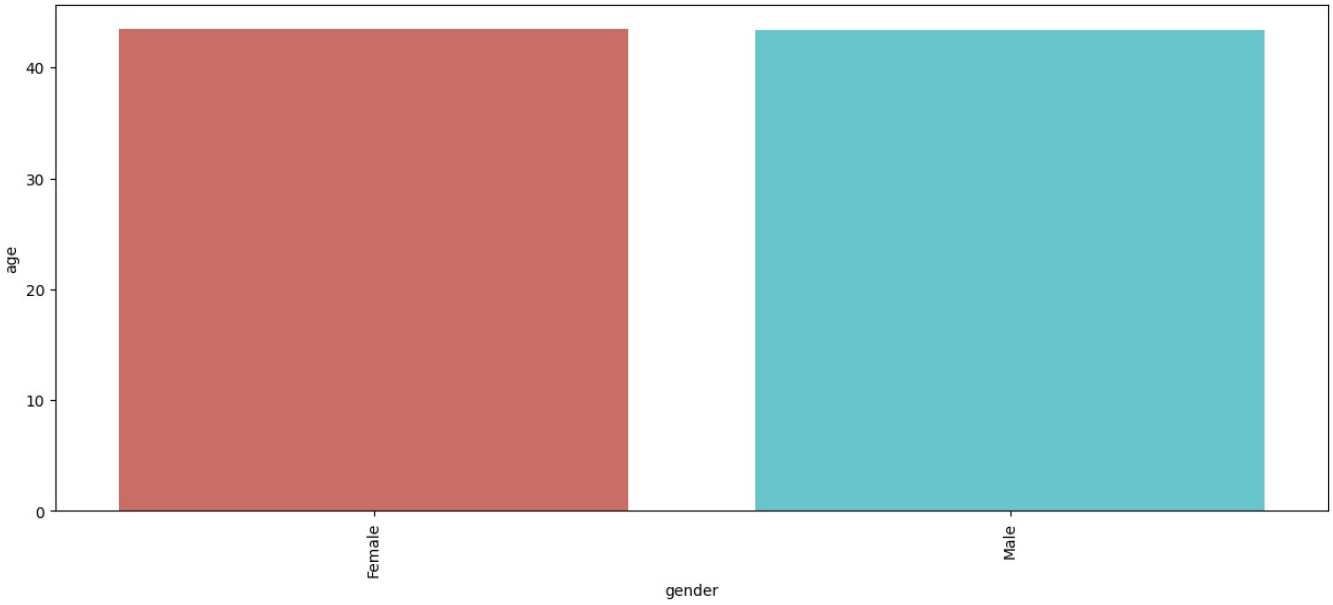


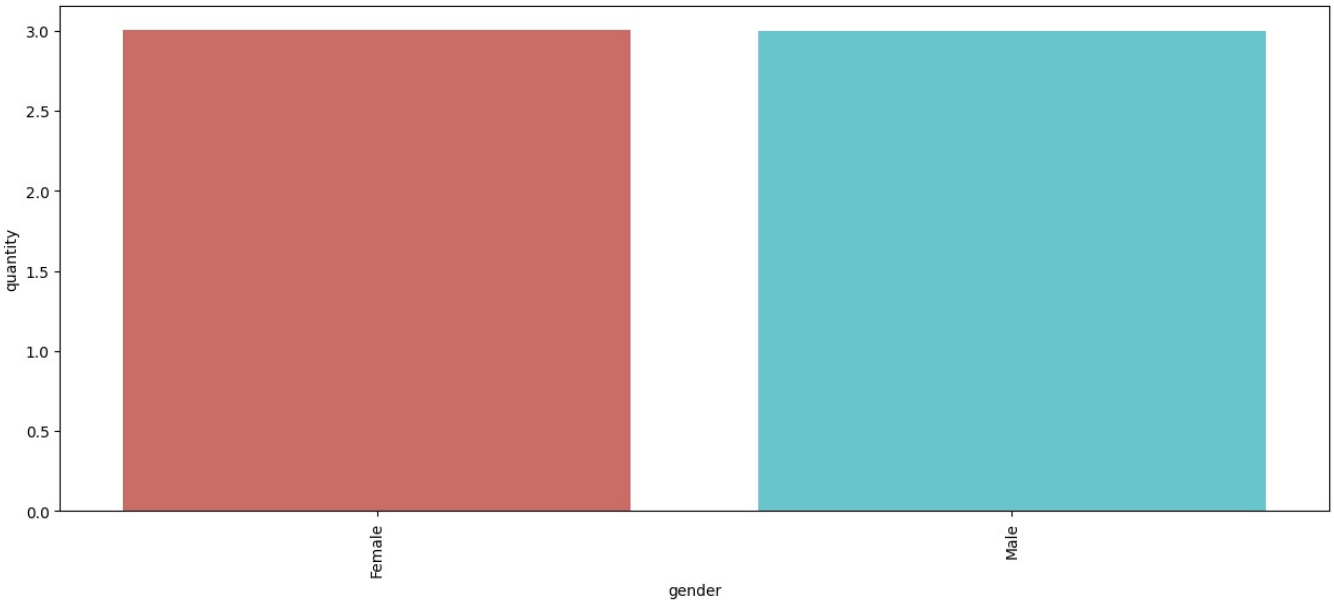


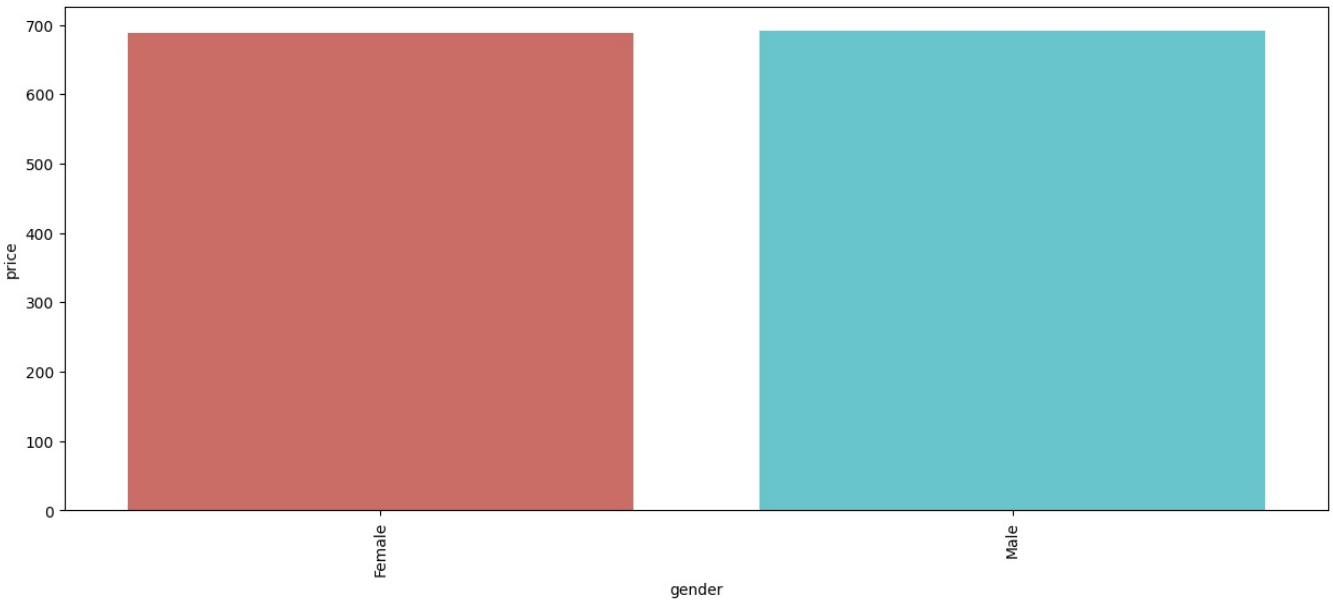
In [63]:
```
print("Numerical columns: ", numerical_cols)
print("Object columns: ", object_cols)
```

```
Numerical columns:  ['age', 'quantity', 'price']
Object columns:  ['invoice_no', 'customer_id', 'gender', 'category', 'payment_method', 'invoice_date', 'shoppin
g_mall']
```

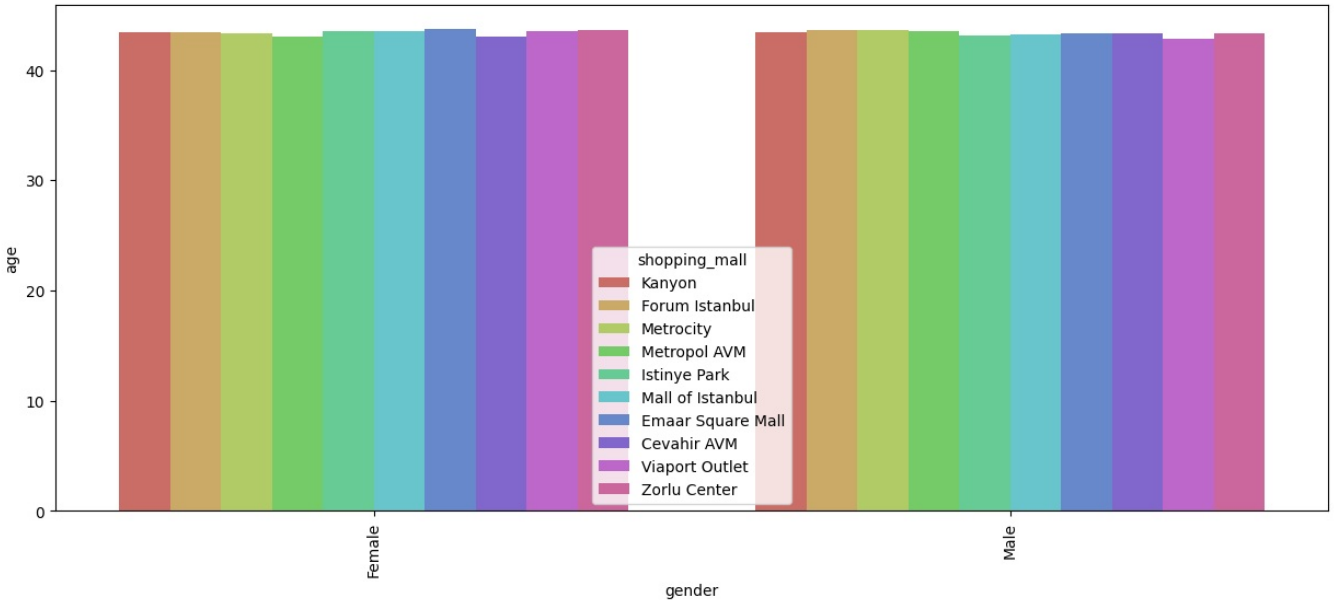
In [64]:
```
for i in numerical_cols:
    plt.figure(figsize=(15,6))
    sns.barplot(x = df['gender'], y = df[i], data = df, ci = None, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```

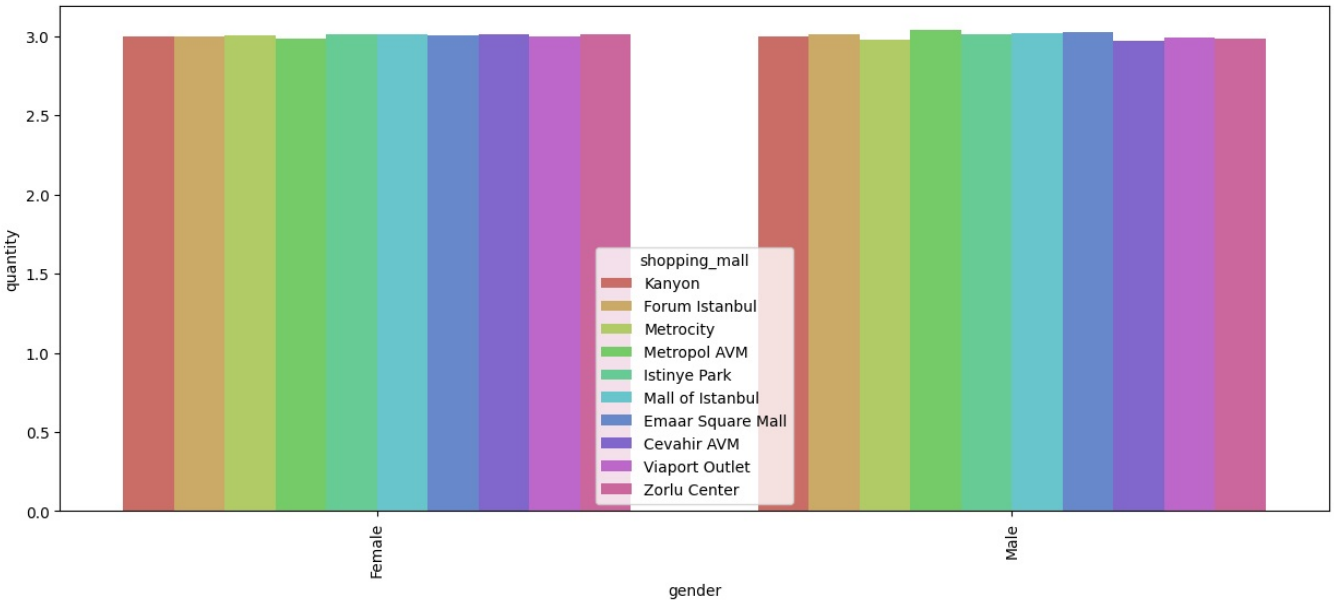


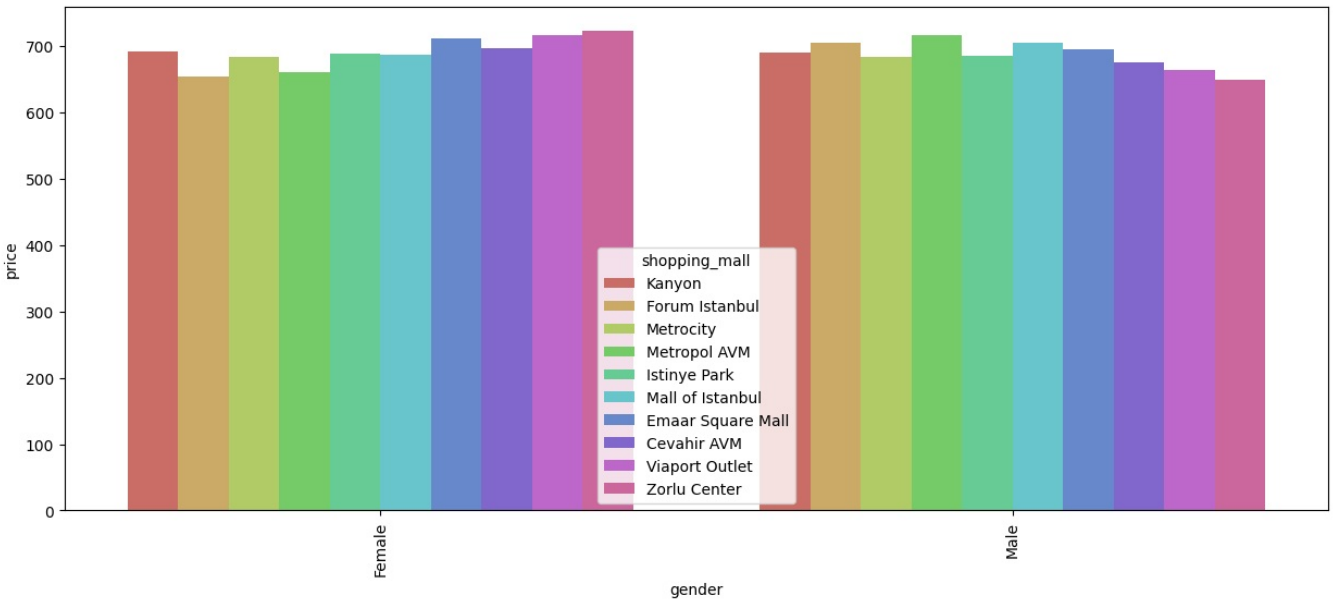




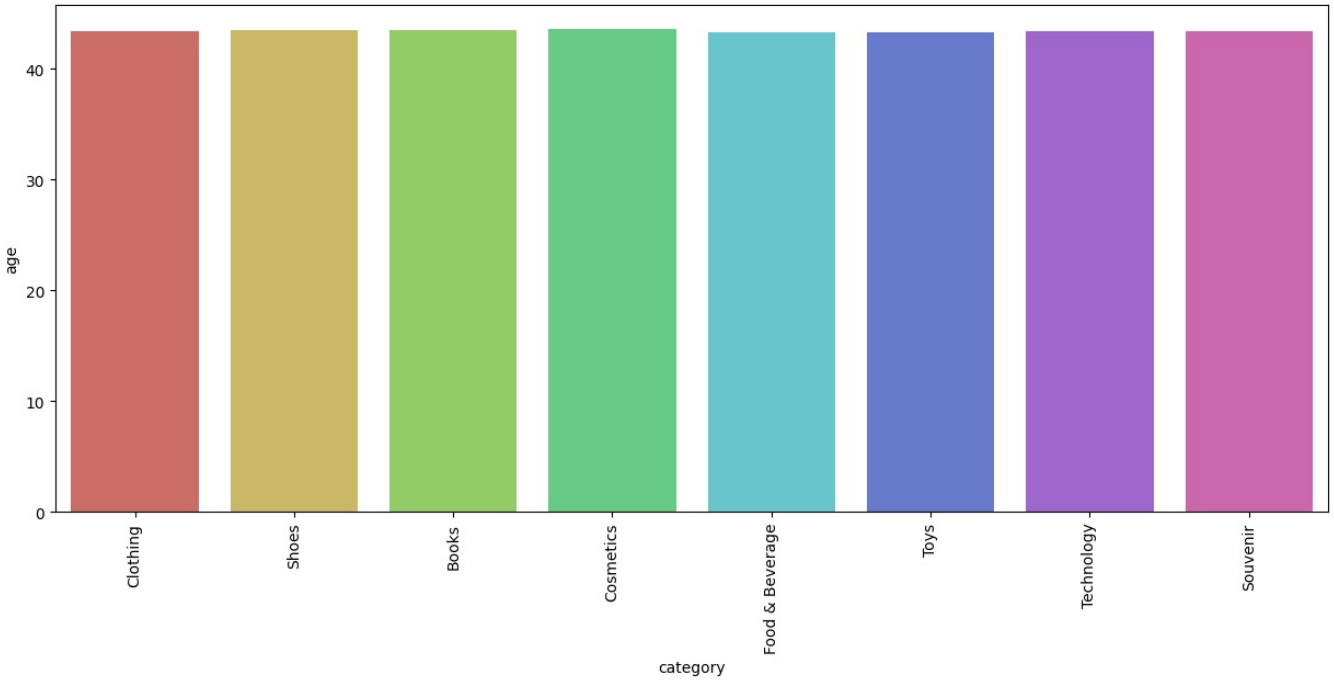
In [65]:
```
for i in numerical_cols:
    plt.figure(figsize=(15,6))
    sns.barplot(x = df['gender'], y = df[i], hue = 'shopping_mall', data = df, ci = None, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```

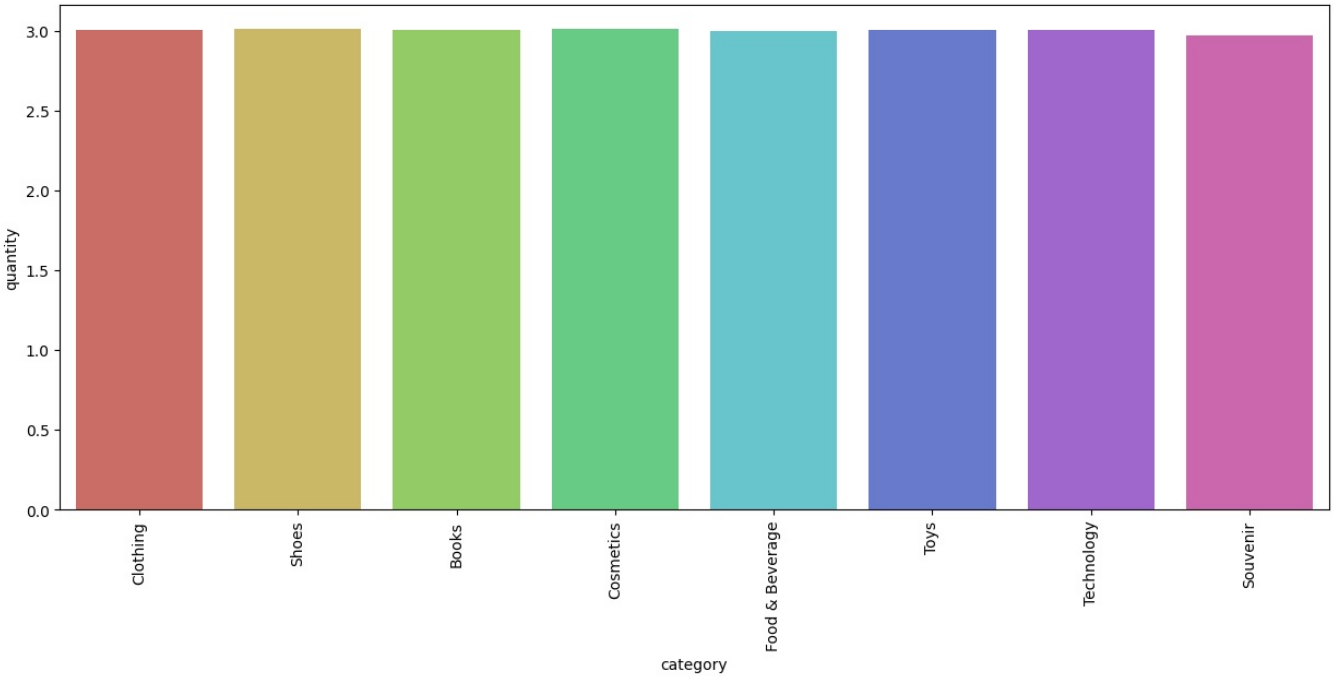


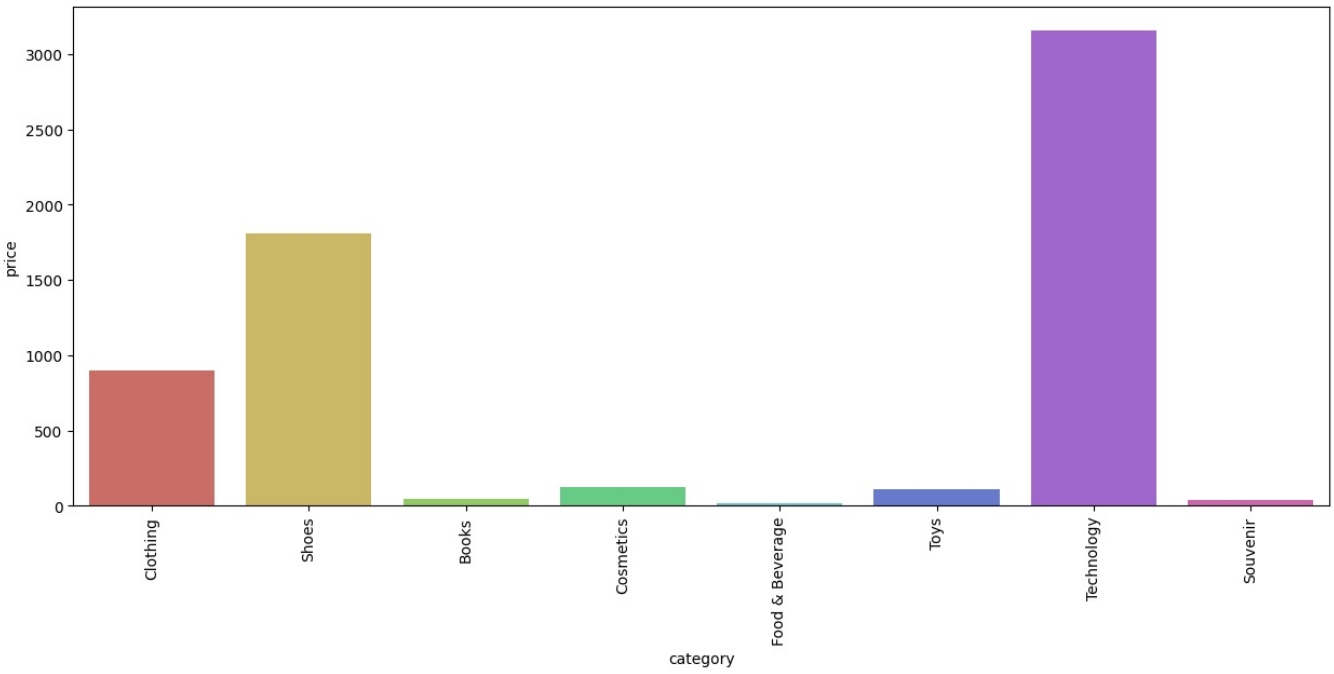




In [66]:
```
for i in numerical_cols:
    plt.figure(figsize=(15,6))
    sns.barplot(x = df['category'], y = df[i], data = df, ci = None, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```

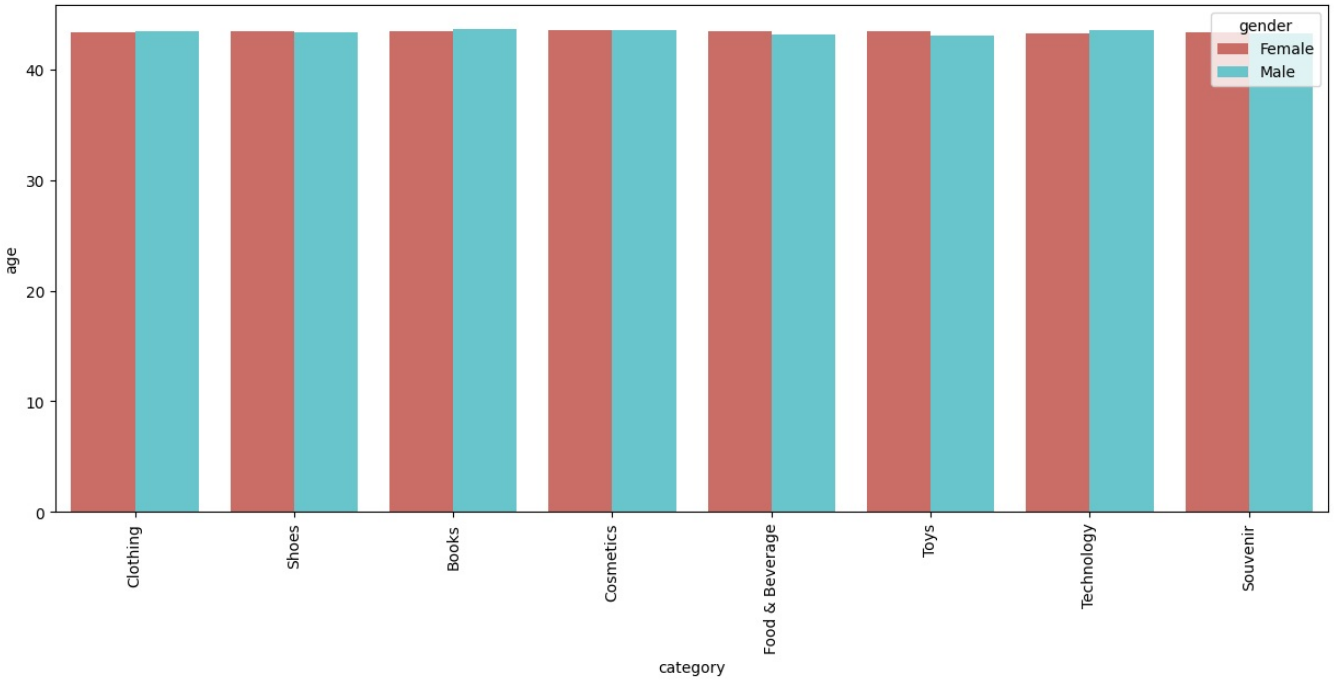


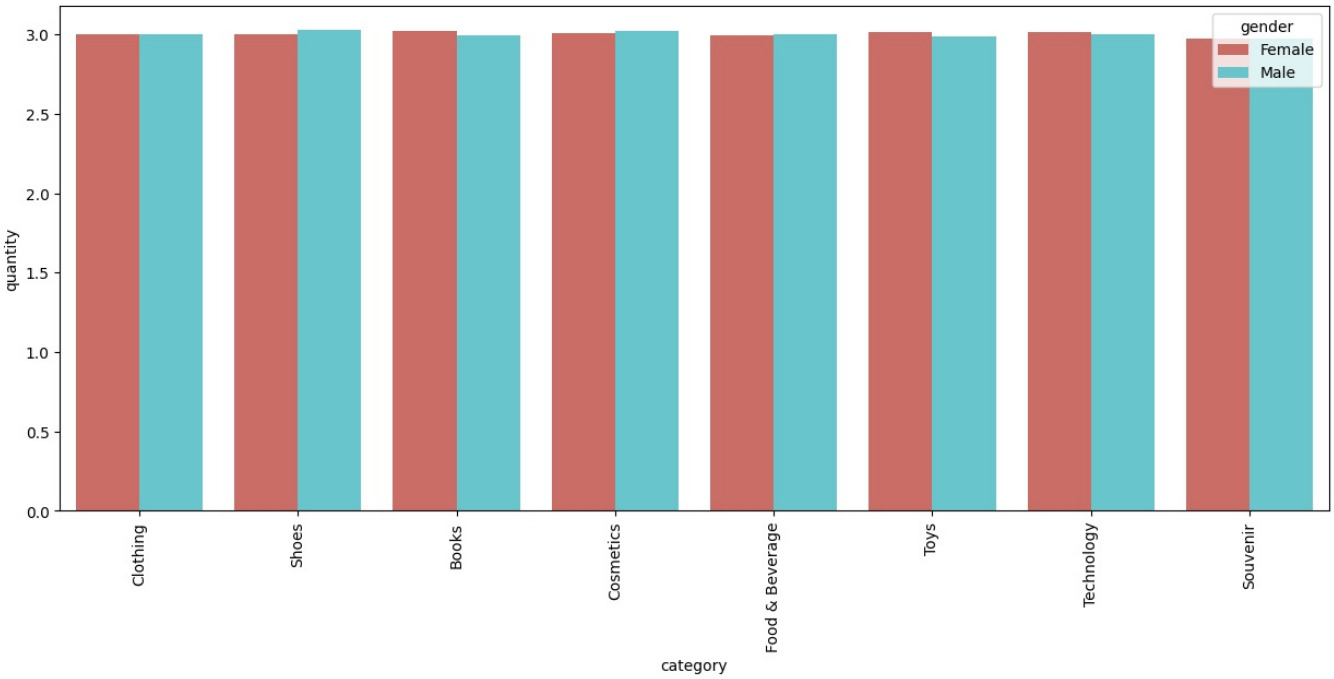


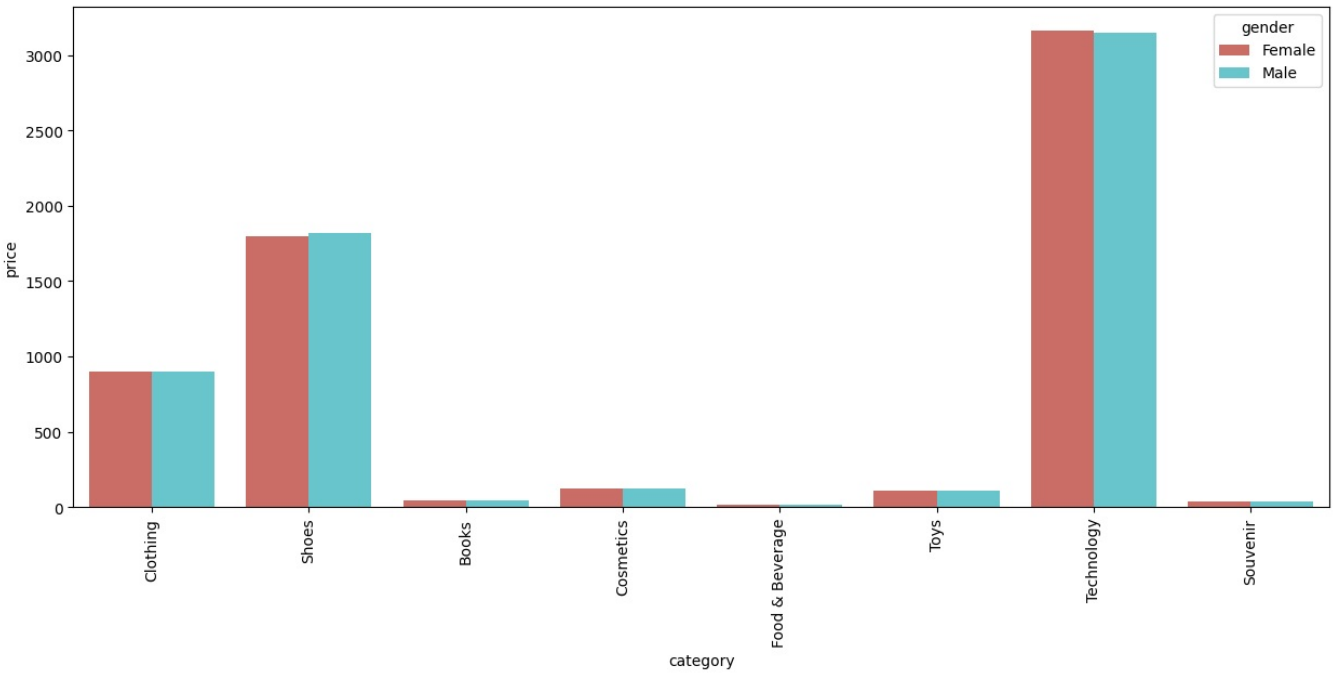


In [67]:
```
for i in numerical_cols:
    plt.figure(figsize=(15,6))
    sns.barplot(x = df['category'], y = df[i], hue = 'gender', data = df, ci = None, palette = 'hls')
```

```
    plt.xticks(rotation = 90)
    plt.show()
```

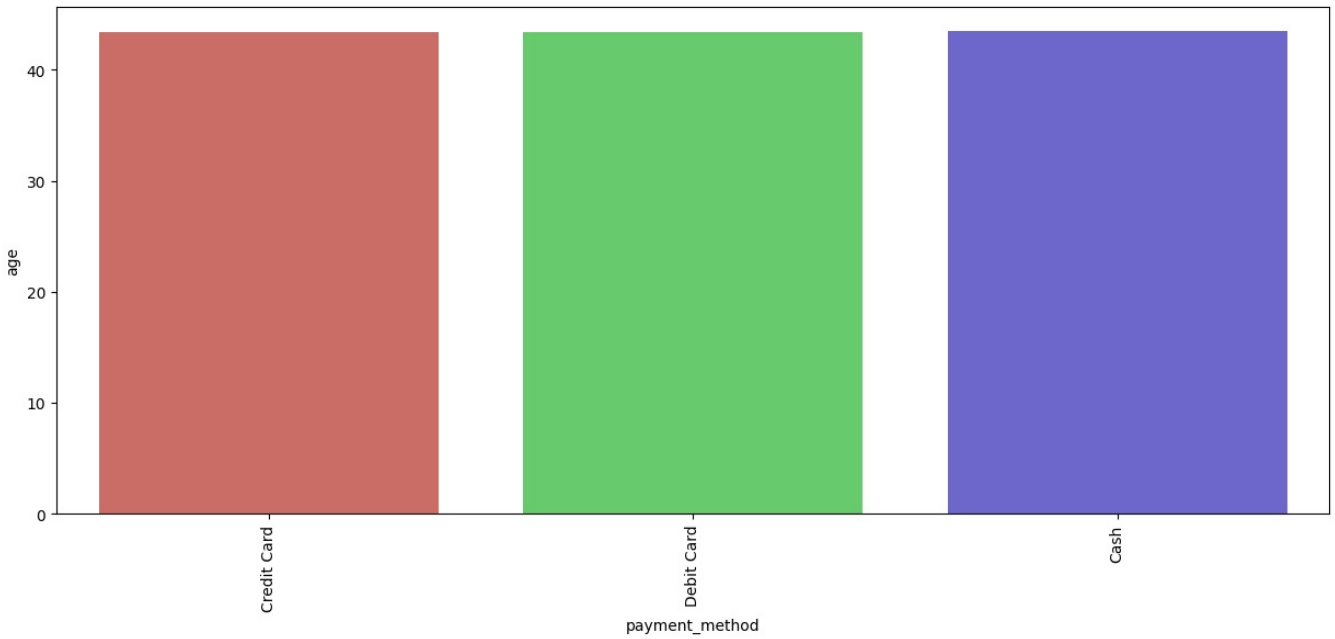


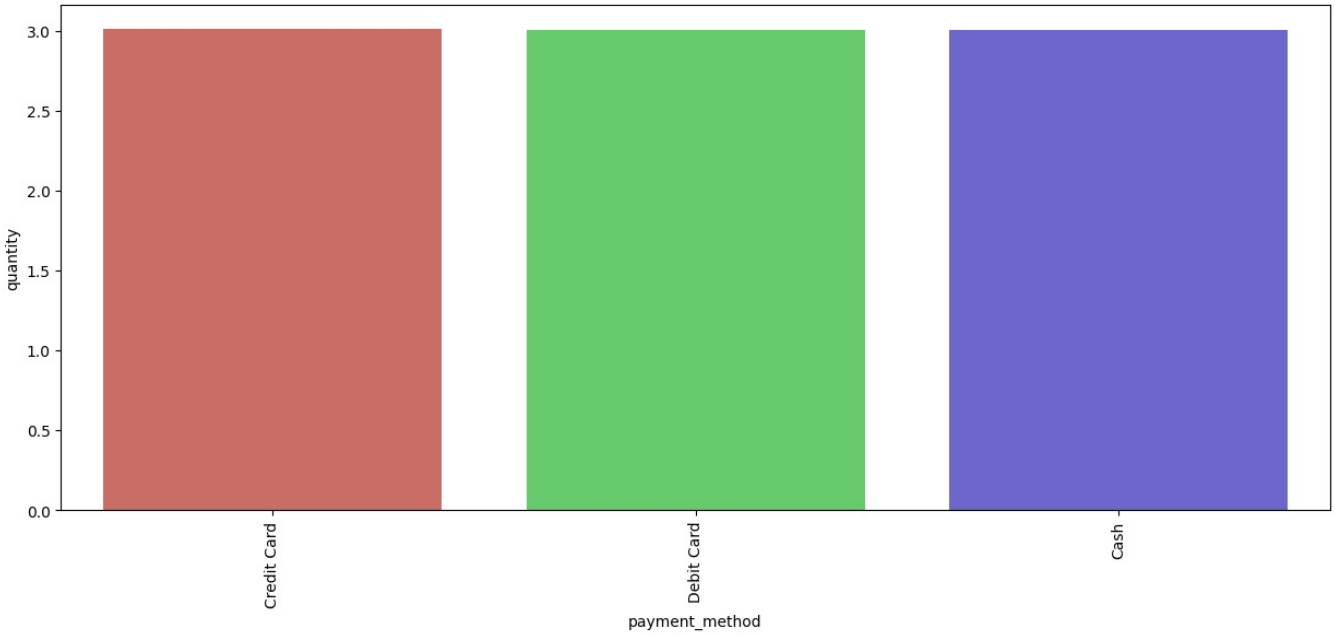


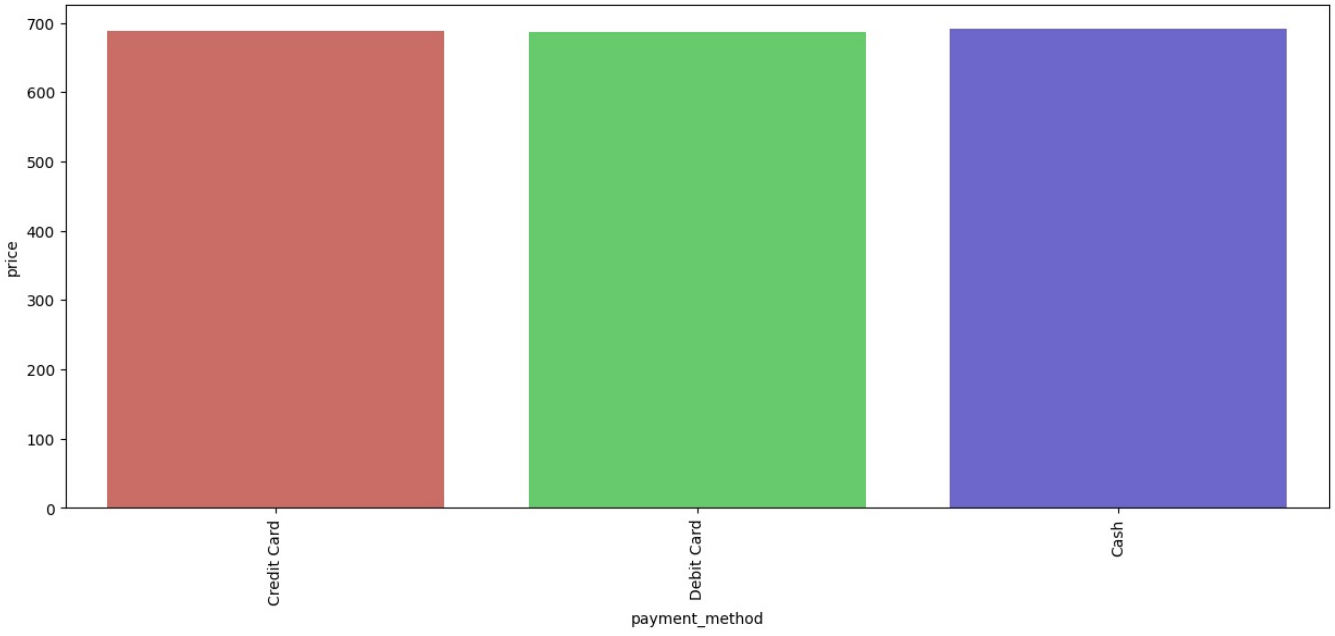


In [68]:
```
for i in numerical_cols:
```

```
    plt.figure(figsize=(15,6))
    sns.barplot(x = df['payment_method'], y = df[i], data = df, ci = None, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```

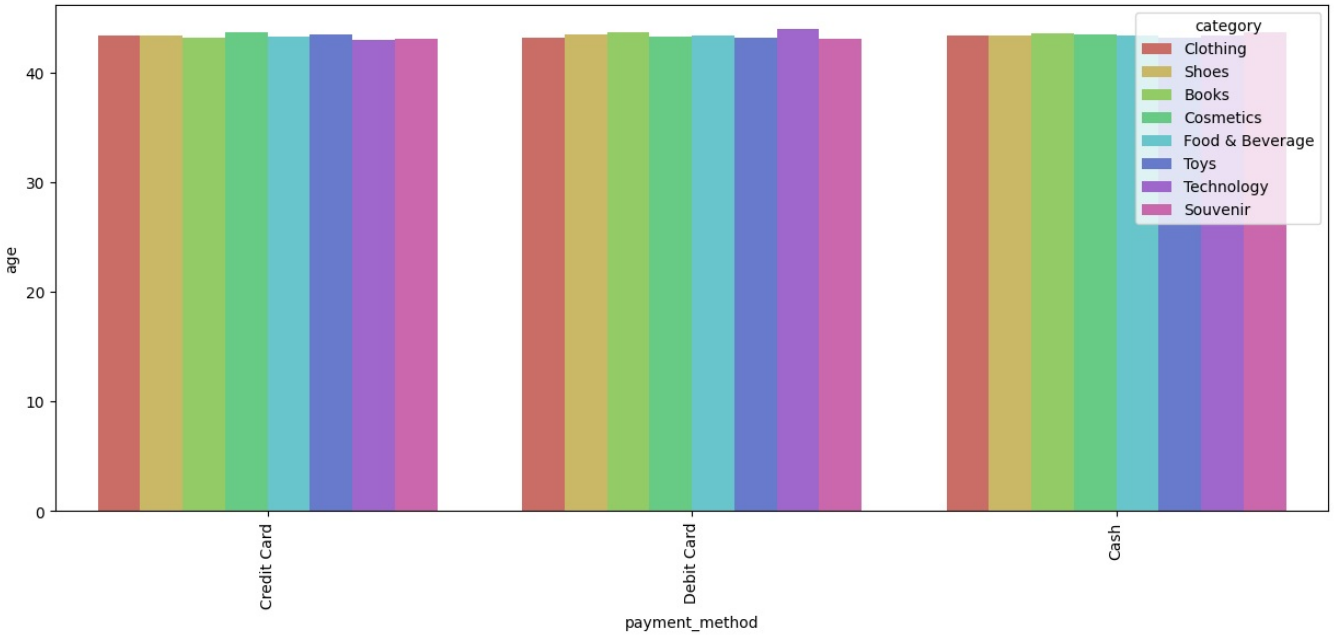


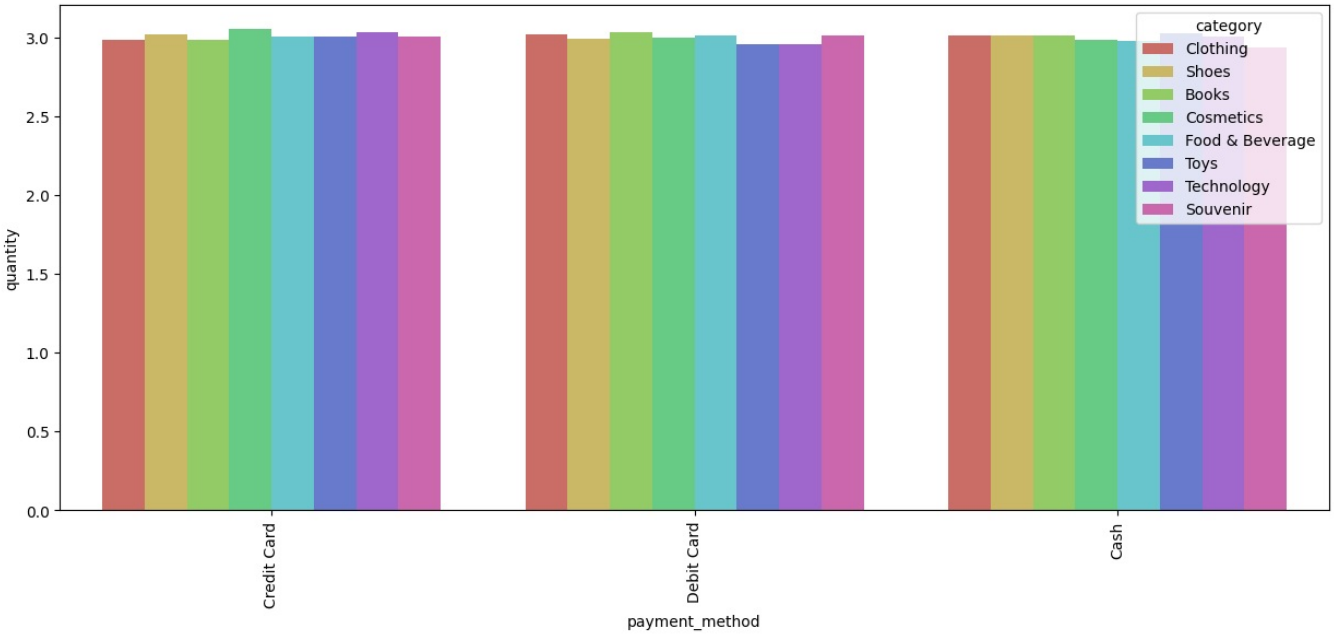


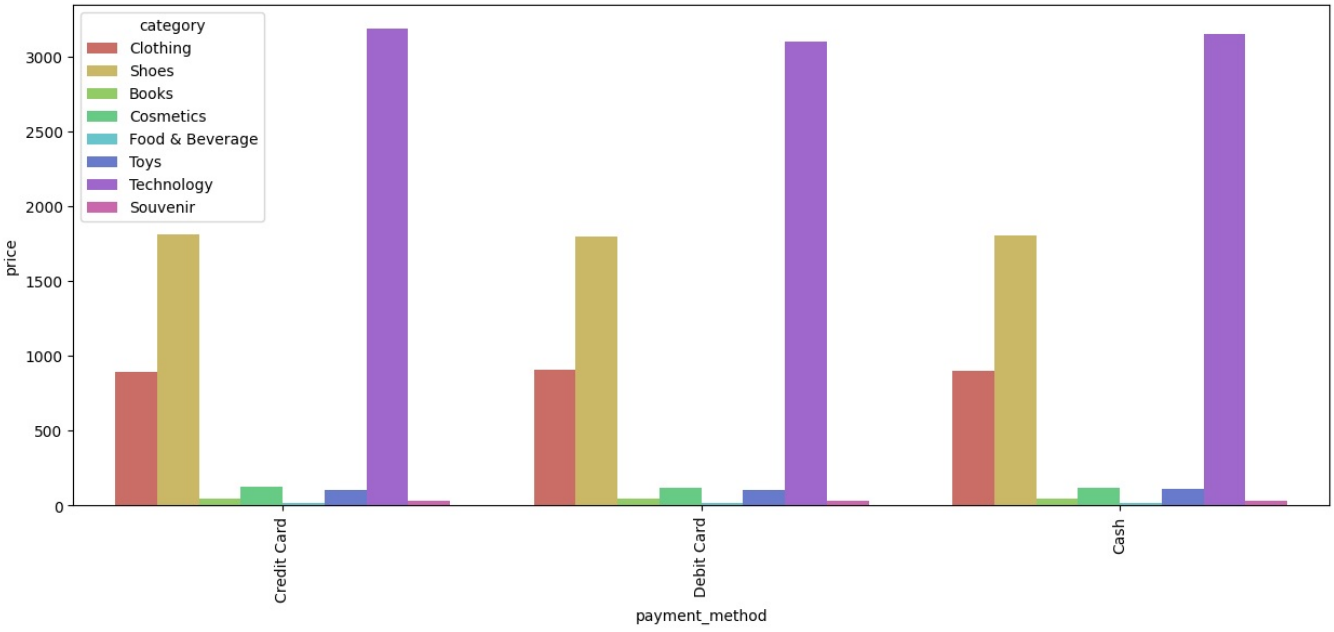


In [69]:
```
for i in numerical_cols:
    plt.figure(figsize=(15,6))
    sns.barplot(x = df['payment_method'], y = df[i], hue = 'category', data = df, ci = None, palette = 'hls')
    plt.xticks(rotation = 90)
```

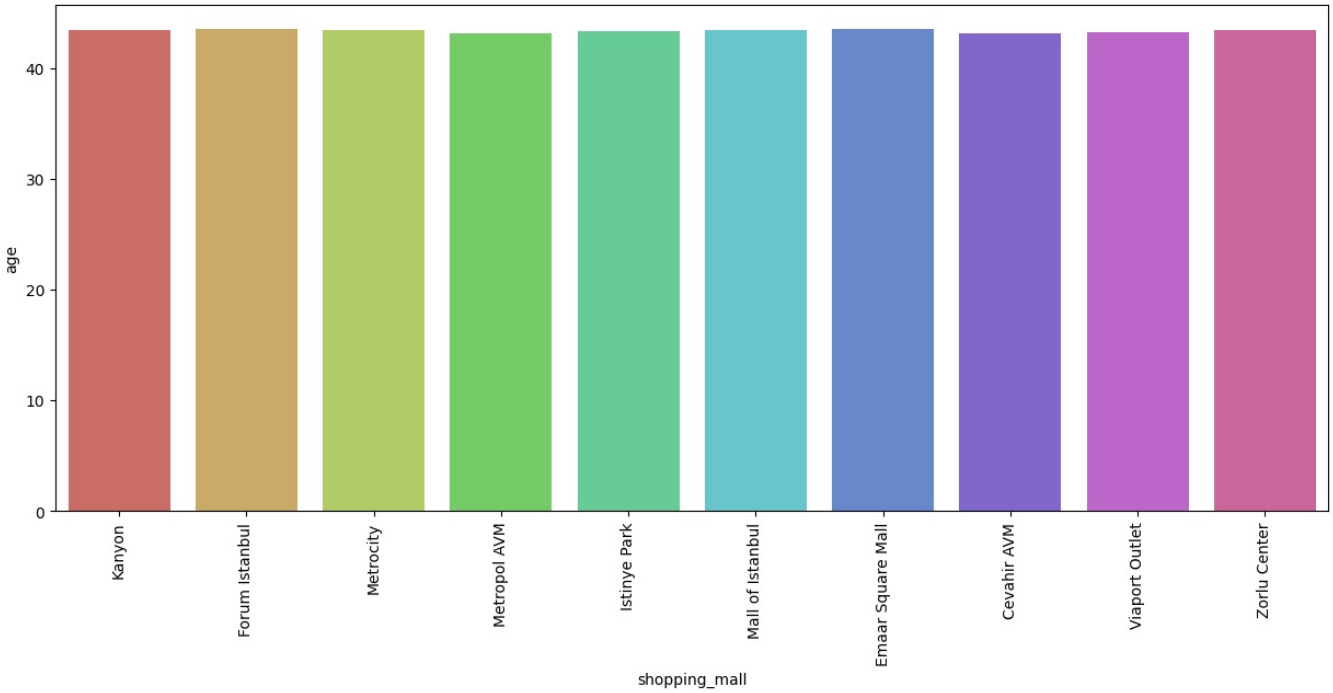
```
    plt.show()
```

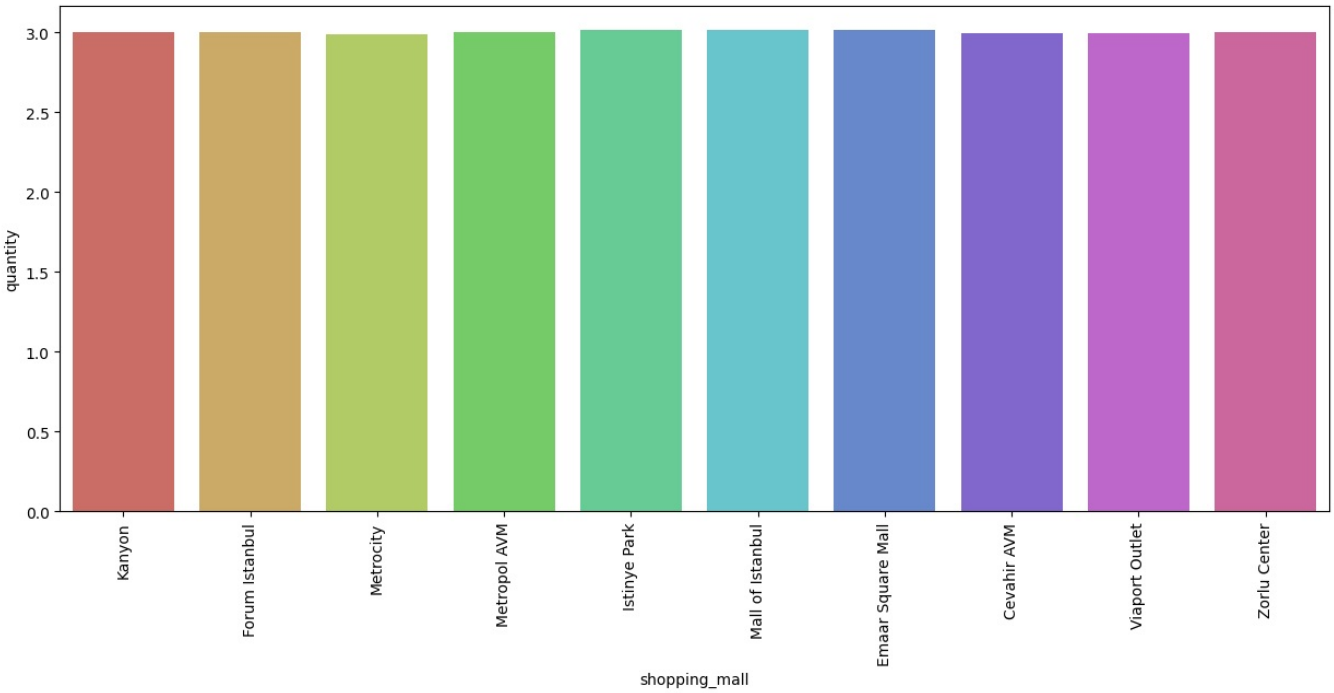


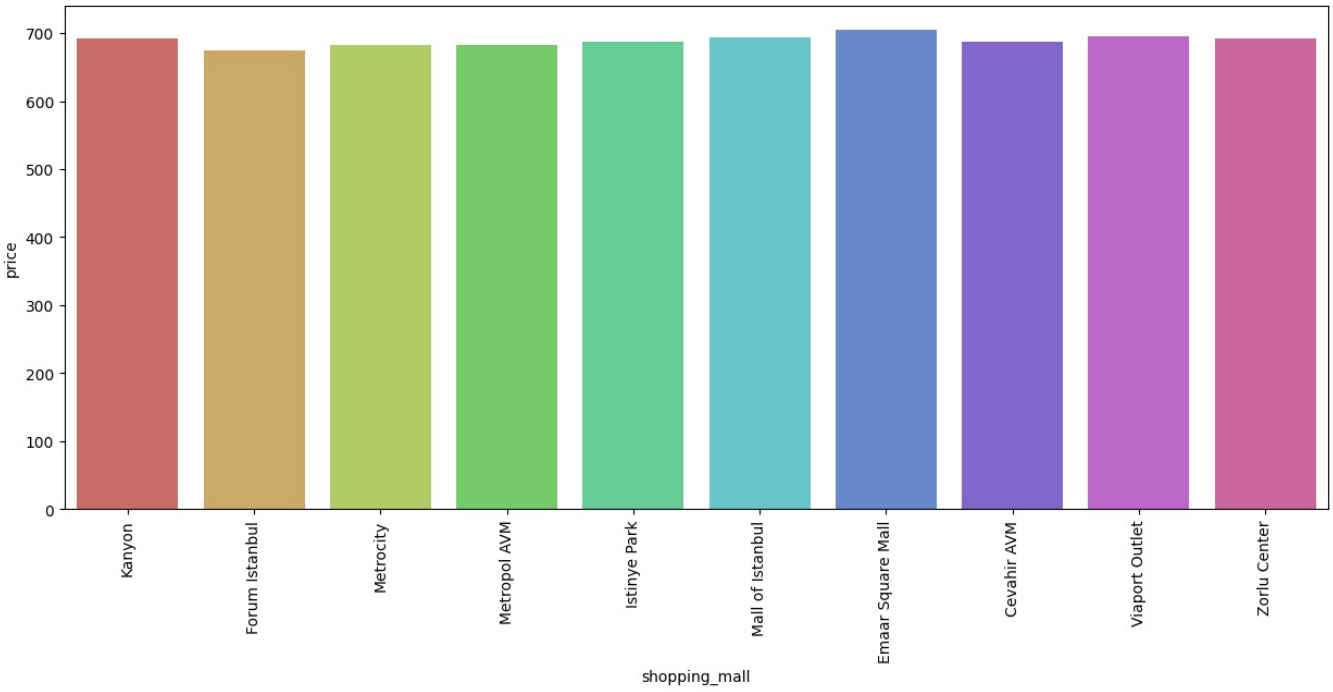




In [70]:
```
for i in numerical_cols:
    plt.figure(figsize=(15,6))
    sns.barplot(x = df['shopping_mall'], y = df[i], data = df, ci = None, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```

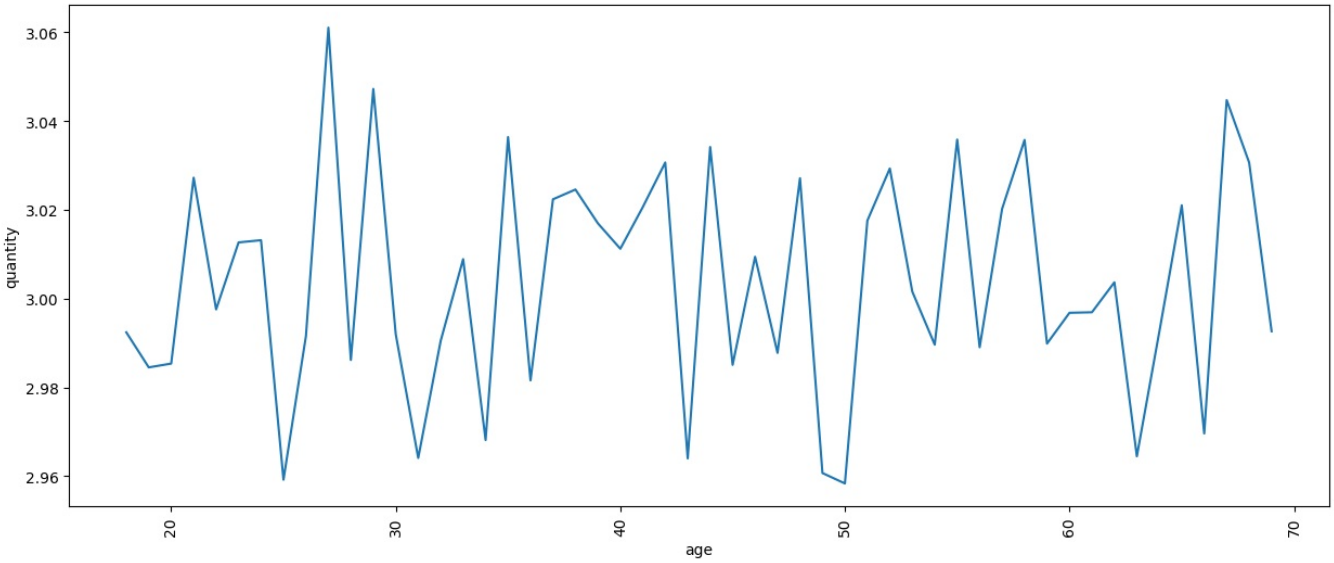


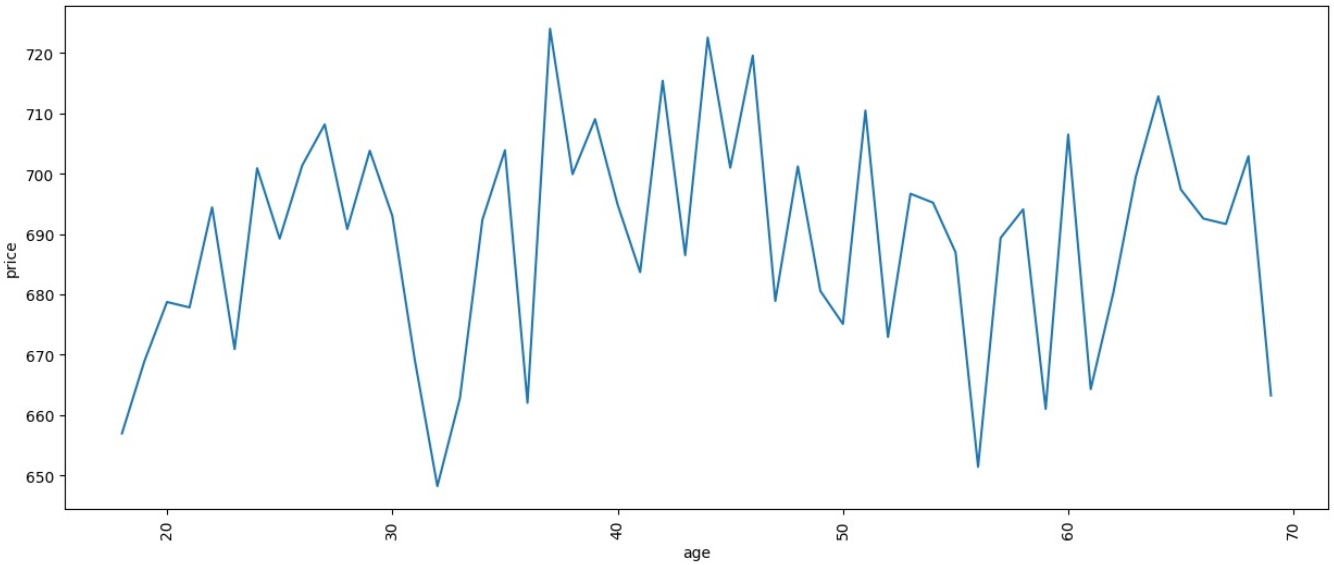


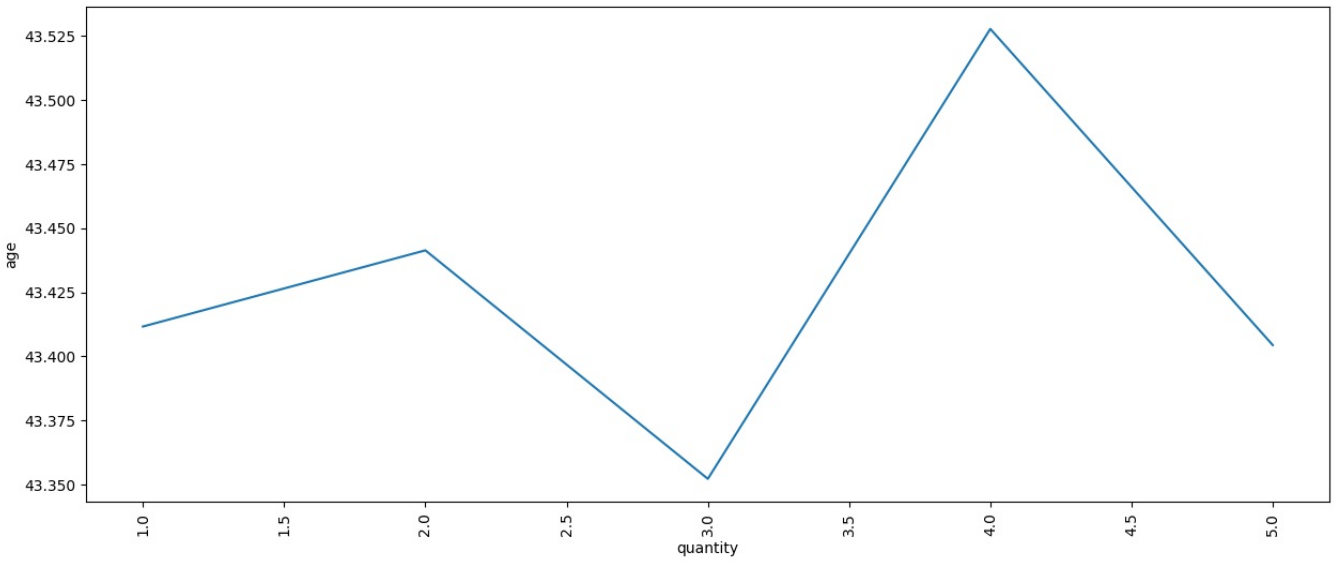


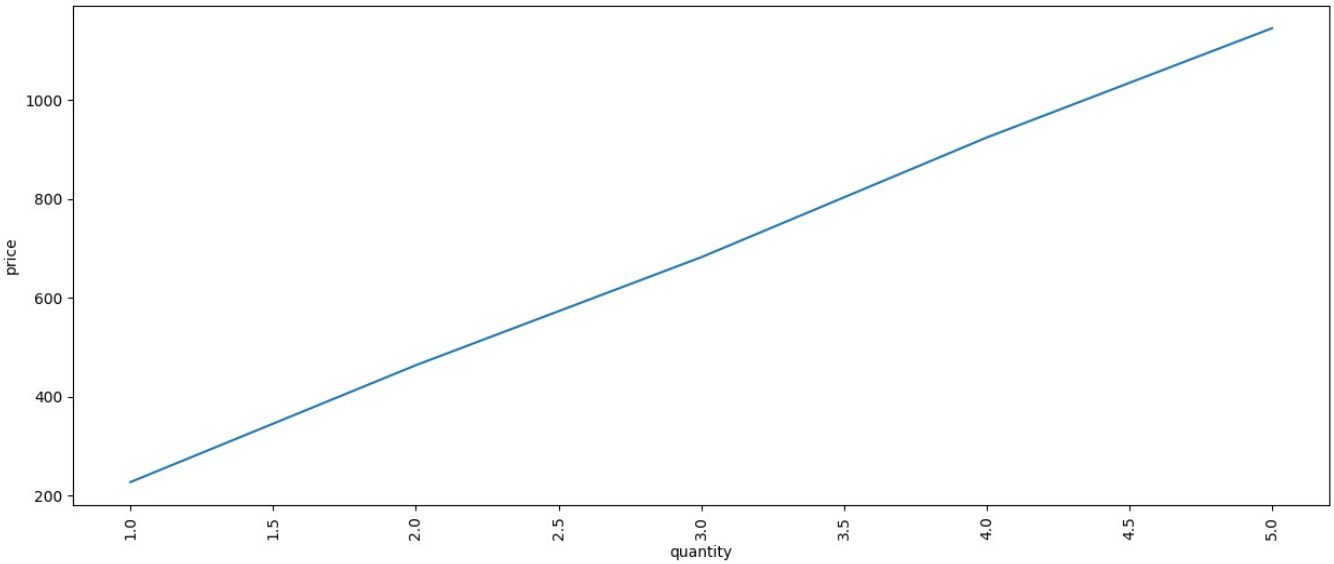
In [71]:
```
for i in numerical_cols:
    for j in numerical_cols:
```

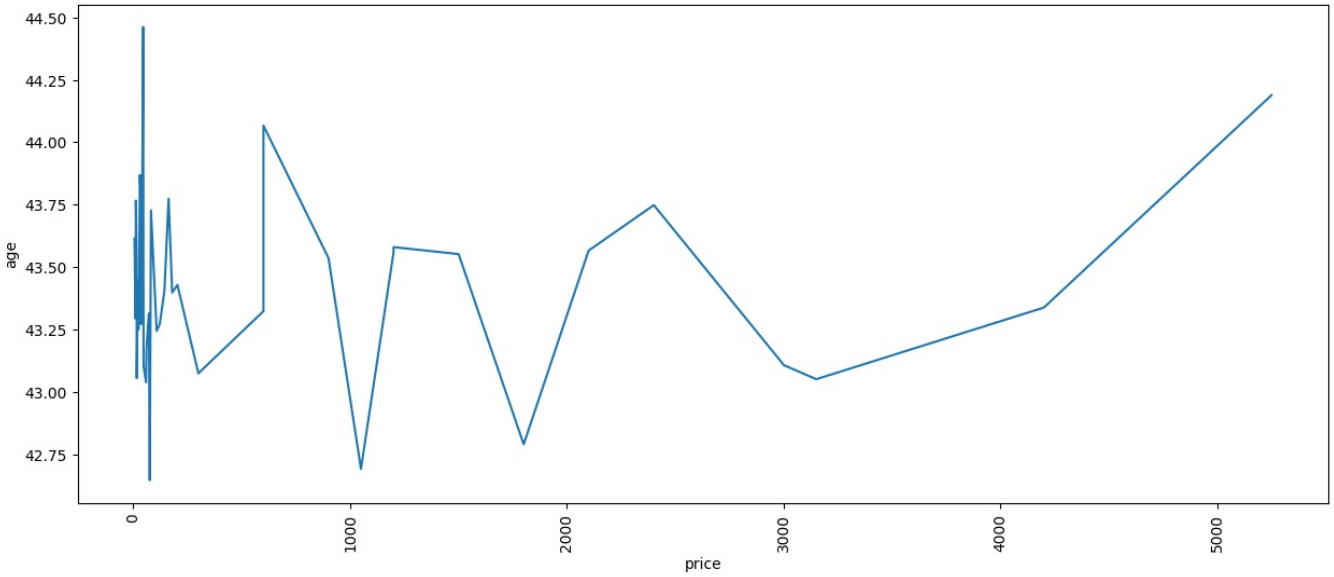
```
        if i != j:
            plt.figure(figsize=(15,6))
            sns.lineplot(x = df[i], y = df[j], data = df, ci = None, palette = 'hls')
            plt.xticks(rotation = 90)
            plt.show()
```

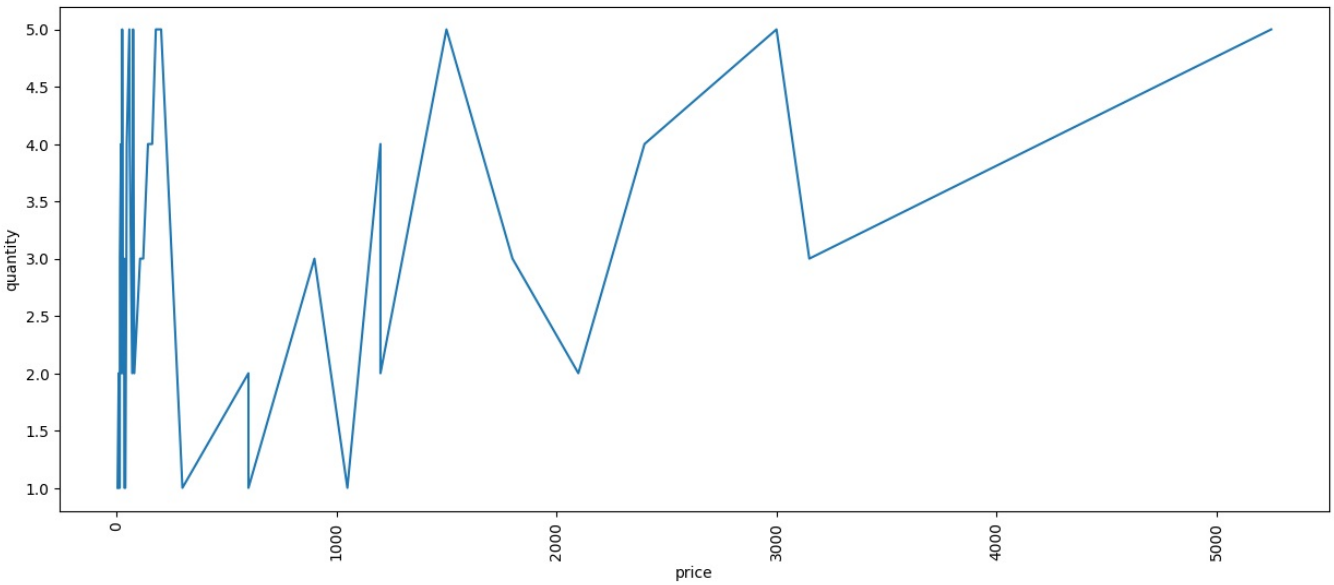






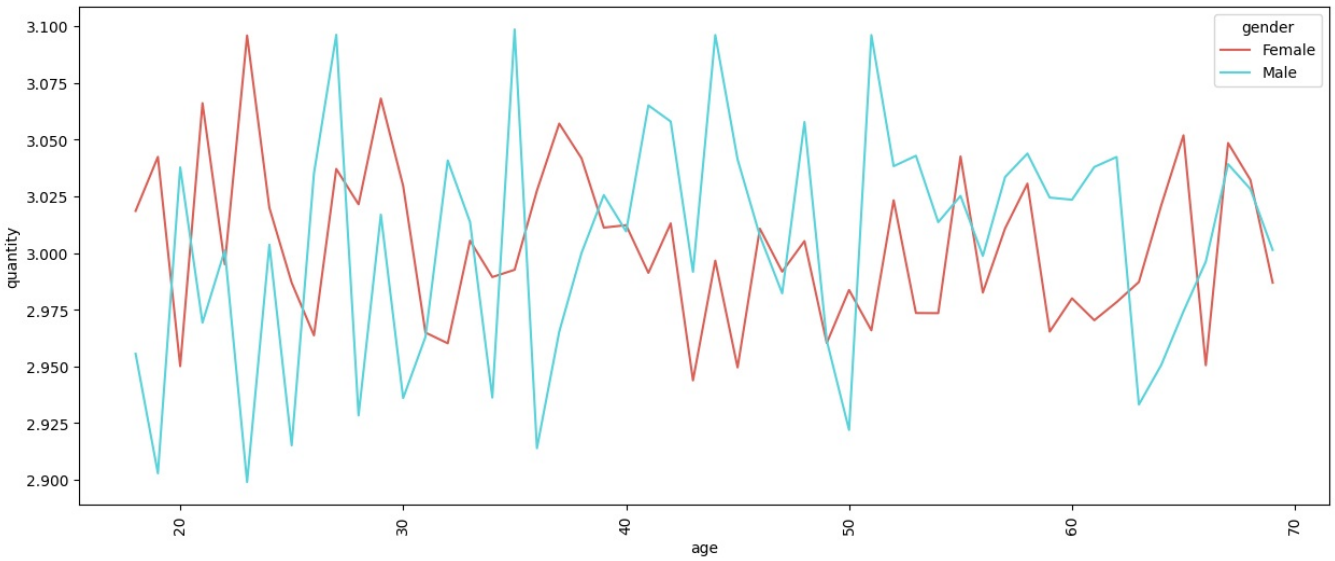


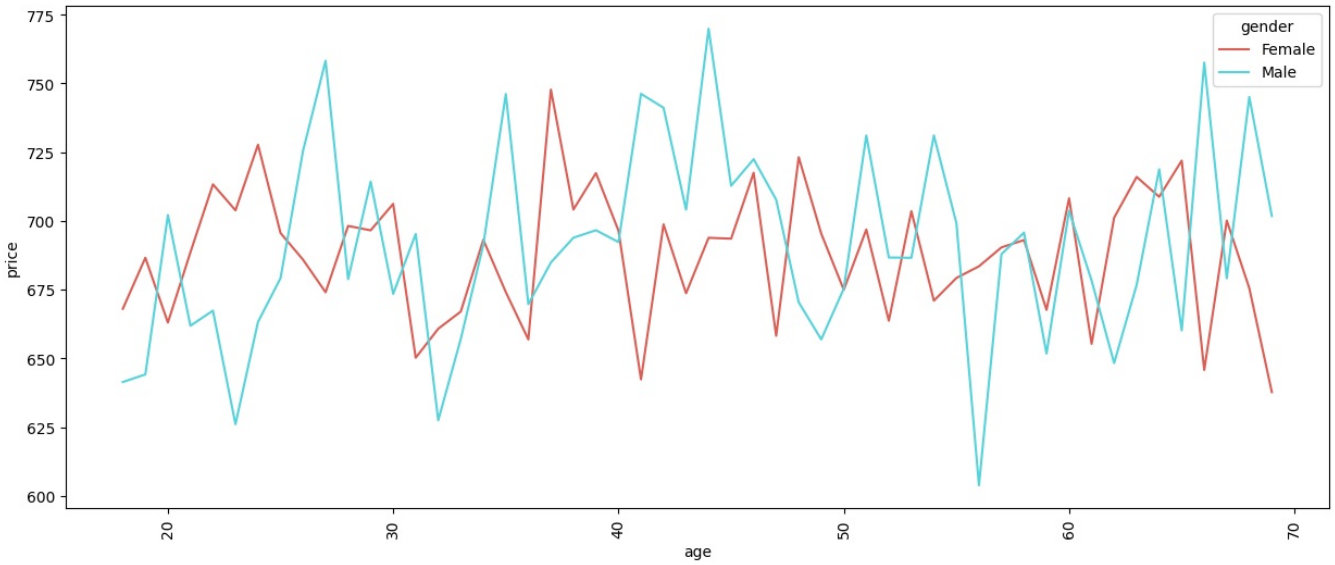


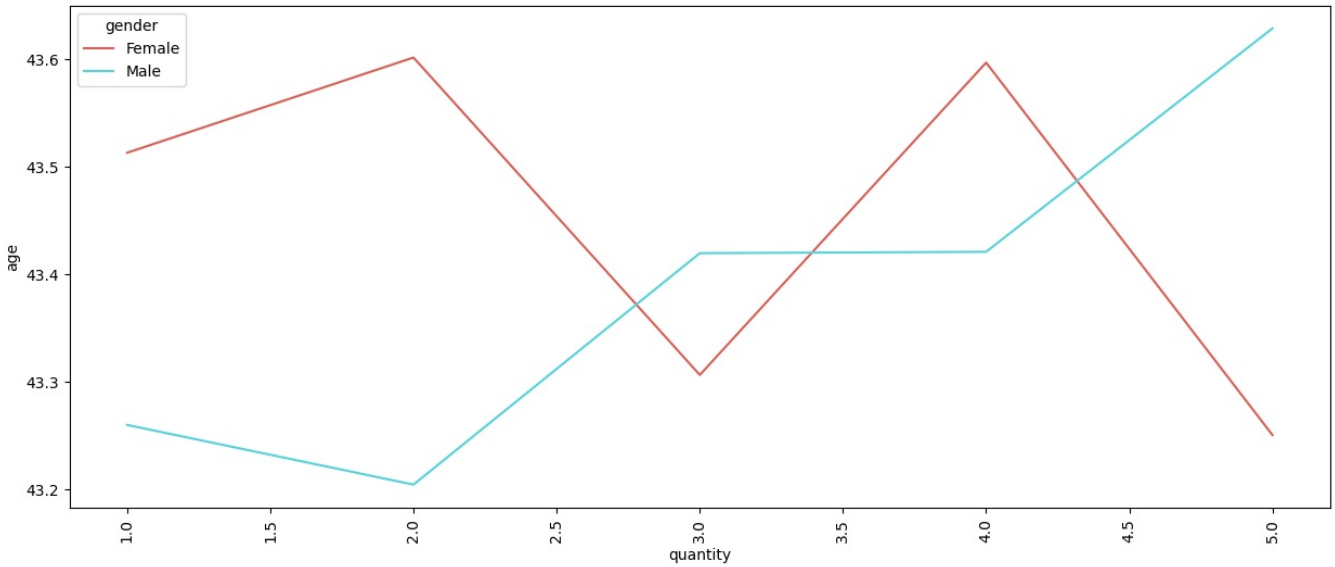


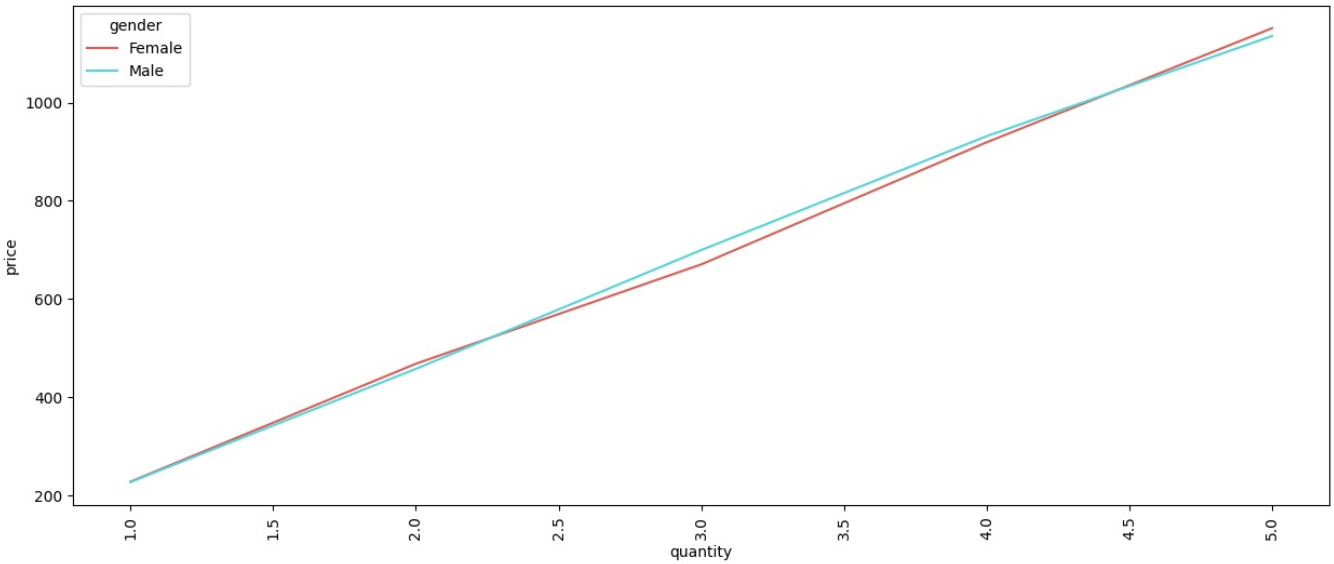
In [72]:
```
for i in numerical_cols:
    for j in numerical_cols:
        if i != j:
```

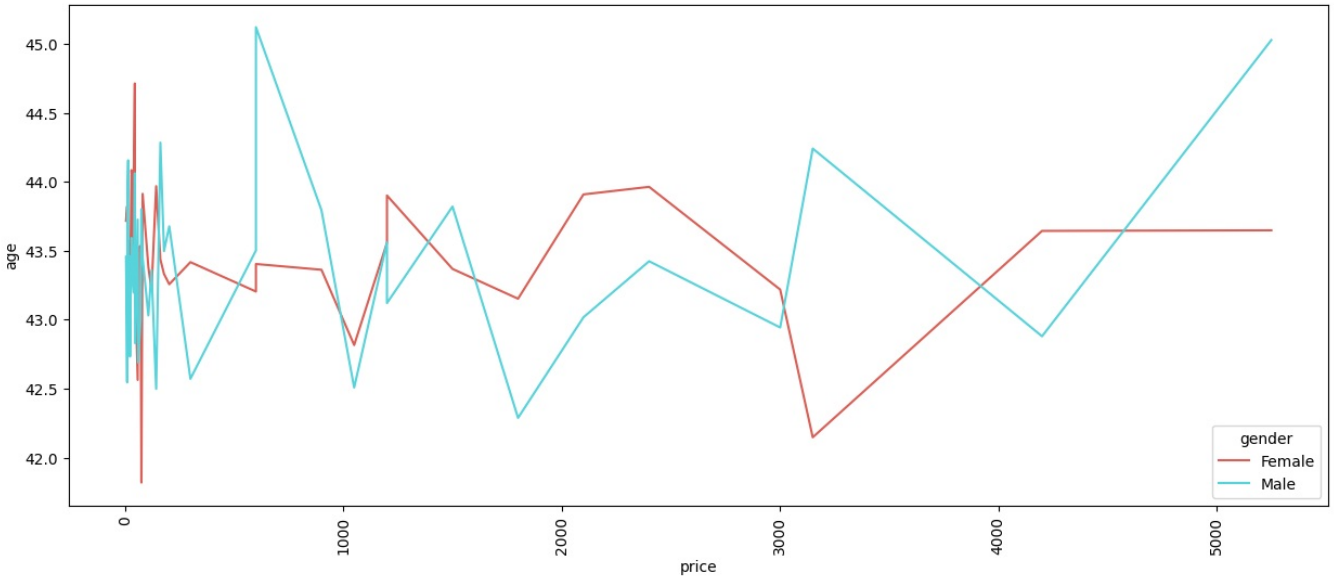
```
            plt.figure(figsize=(15,6))
            sns.lineplot(x = df[i], y = df[j], hue = 'gender', data = df, ci = None, palette = 'hls')
            plt.xticks(rotation = 90)
            plt.show()
```

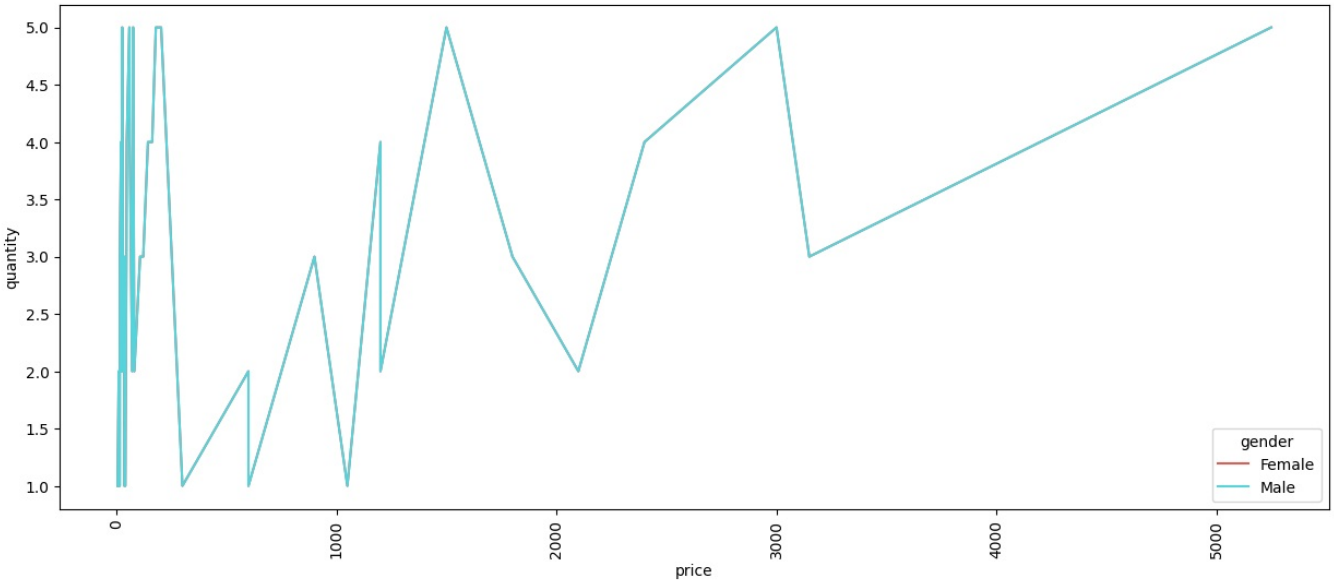










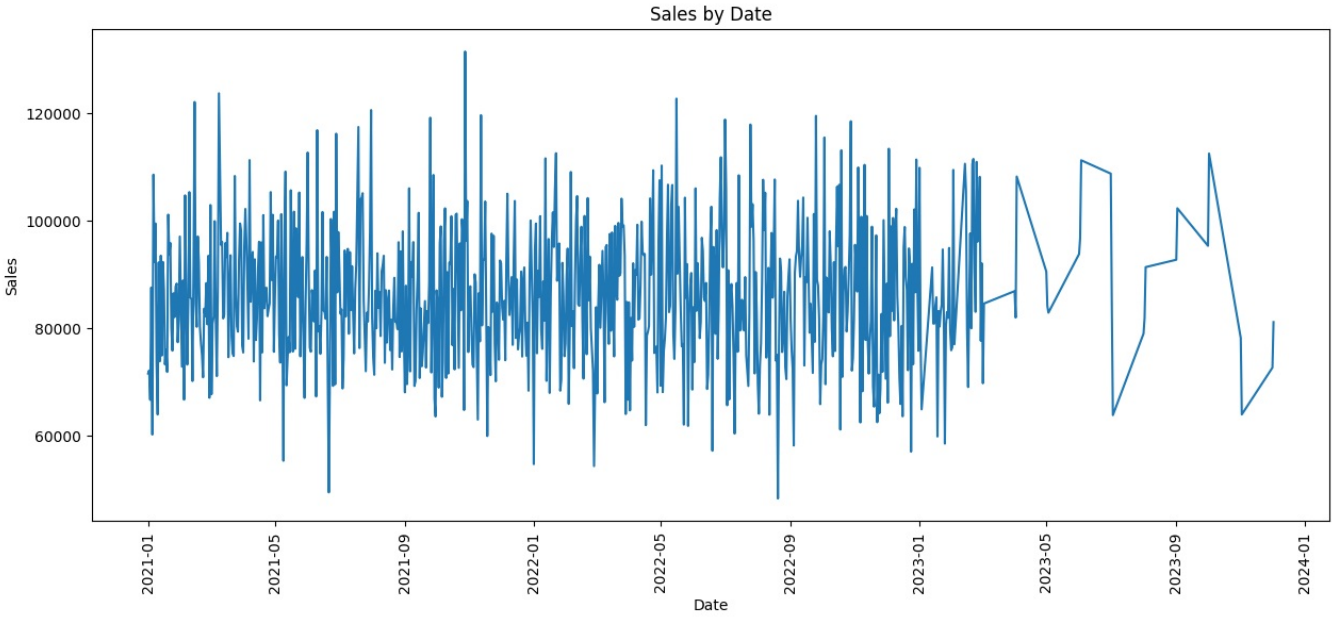


In [73]:
```
df['invoice_date'] = pd.to_datetime(df['invoice_date'])
```

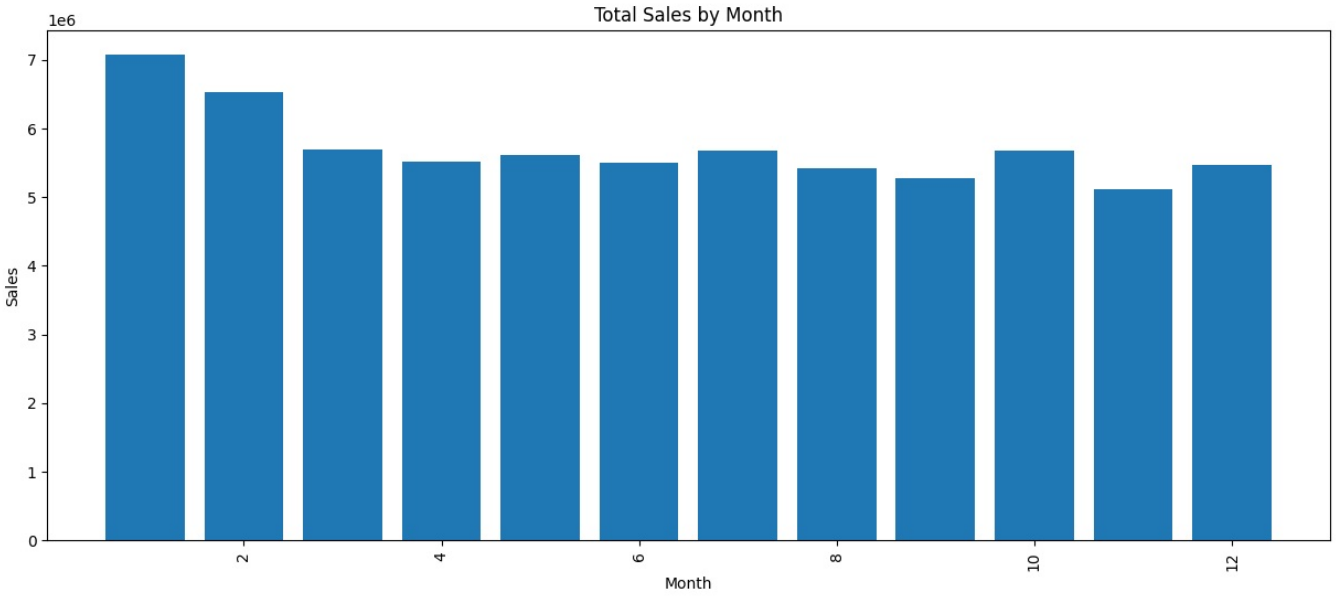
In [74]:
```
df.info()
```

```
<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  datetime64[ns]
 9   shopping_mall   99457 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(2), object(6)
memory usage: 7.6+ MB
```

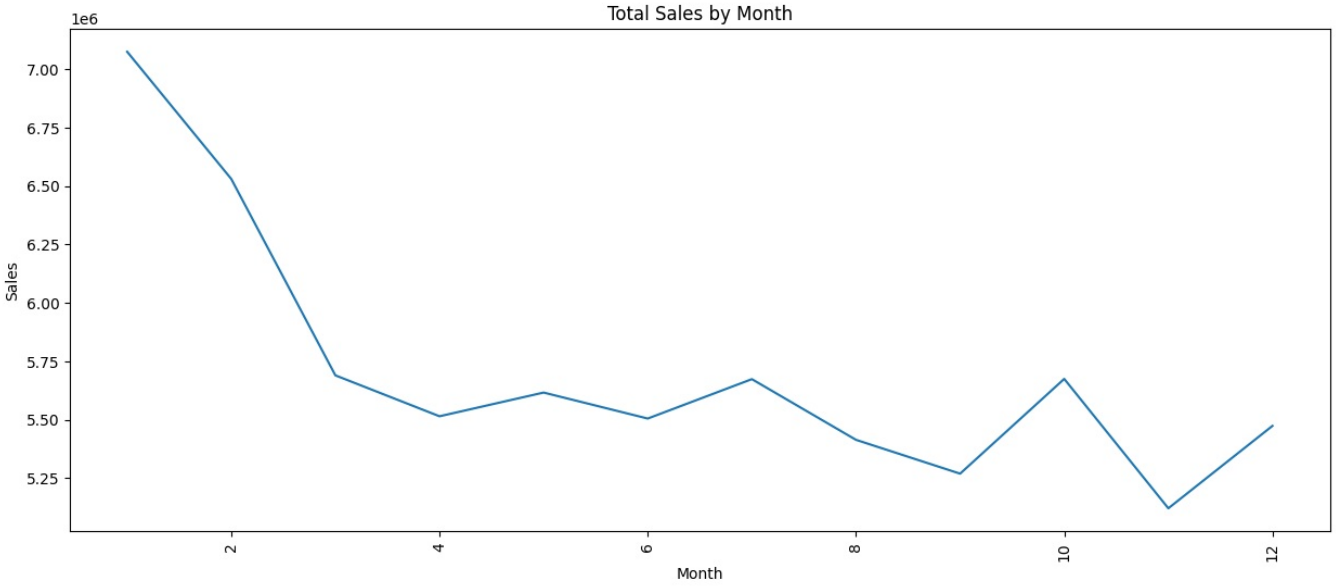
In [75]:
```
sales_by_date = df.groupby('invoice_date')['price'].sum()
plt.figure(figsize=(15,6))
plt.plot(sales_by_date.index, sales_by_date.values)
plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales by Date')
plt.xticks(rotation = 90)
plt.show()
```



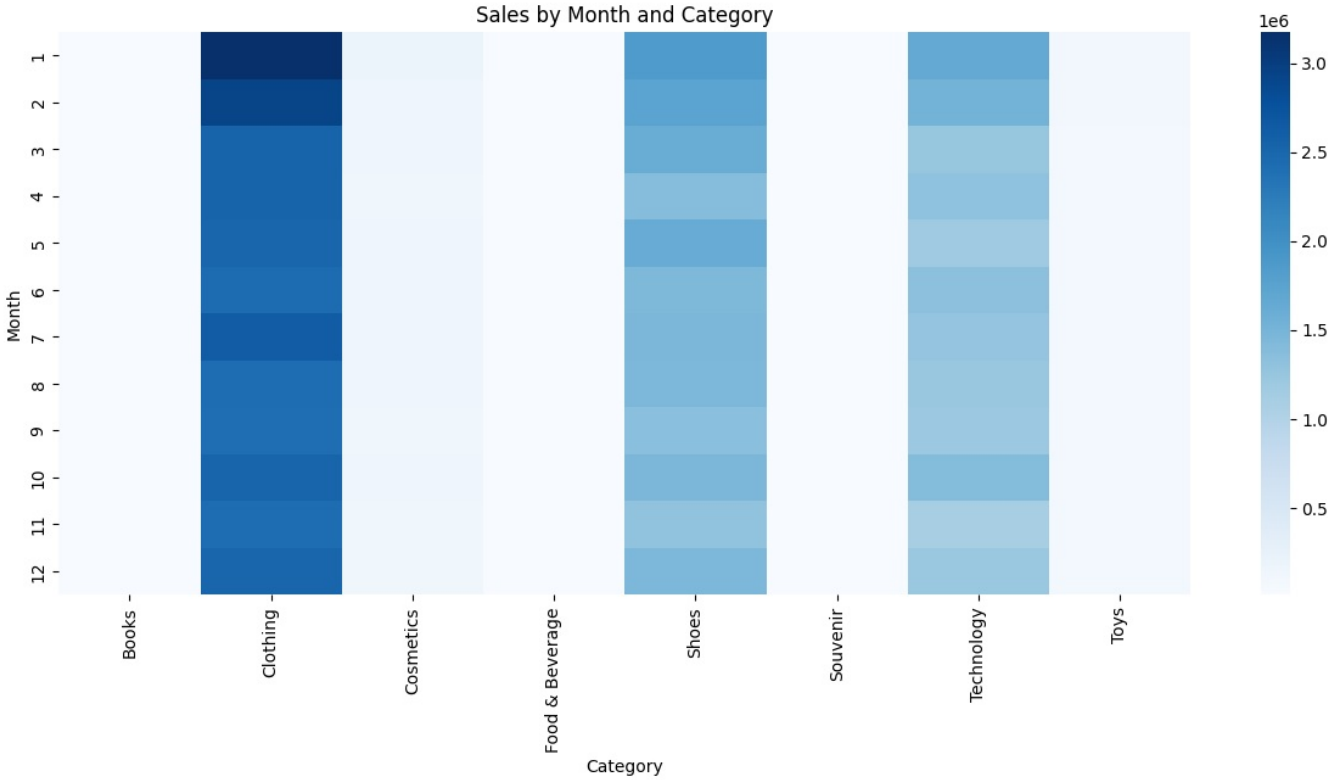
In [76]:
```
df['month'] = pd.DatetimeIndex(df['invoice_date']).month
sales_by_month = df.groupby('month')['price'].sum()

plt.figure(figsize=(15,6))
plt.bar(sales_by_month.index, sales_by_month.values)
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Total Sales by Month')
plt.xticks(rotation = 90)
plt.show()
```



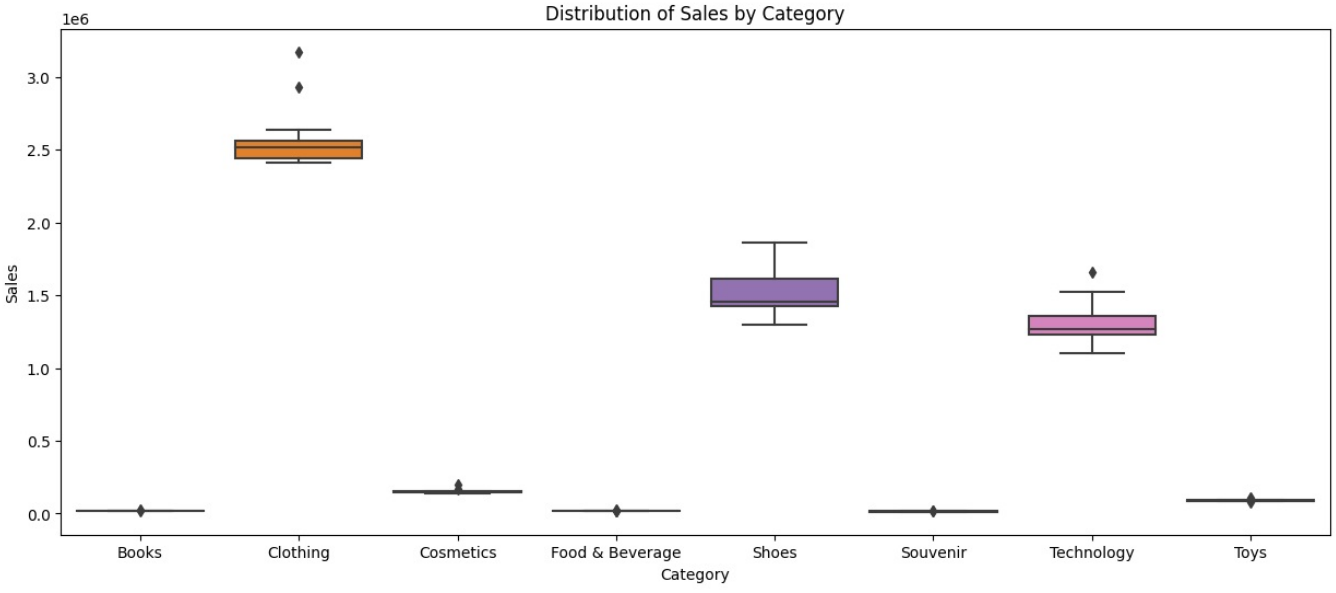
In [77]:
```
df['month'] = pd.DatetimeIndex(df['invoice_date']).month
sales_by_month = df.groupby('month')['price'].sum()

plt.figure(figsize=(15,6))
plt.plot(sales_by_month.index, sales_by_month.values)
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Total Sales by Month')
plt.xticks(rotation = 90)
plt.show()
```



In [78]:
```
df['month'] = pd.DatetimeIndex(df['invoice_date']).month
sales_by_month_category = df.groupby(['month', 'category'])['price'].sum().unstack()

plt.figure(figsize=(15,6))
sns.heatmap(sales_by_month_category, cmap='Blues')
plt.xlabel('Category')
plt.ylabel('Month')
plt.title('Sales by Month and Category')
plt.xticks(rotation = 90)
plt.show()
```



In [79]:
```
sales_by_month_category = df.groupby(['month', 'category'])['price'].sum().reset_index()

plt.figure(figsize=(15,6))
sns.boxplot(x='category', y='price', data=sales_by_month_category)
plt.xlabel('Category')
plt.ylabel('Sales')
plt.title('Distribution of Sales by Category')
plt.show()
```



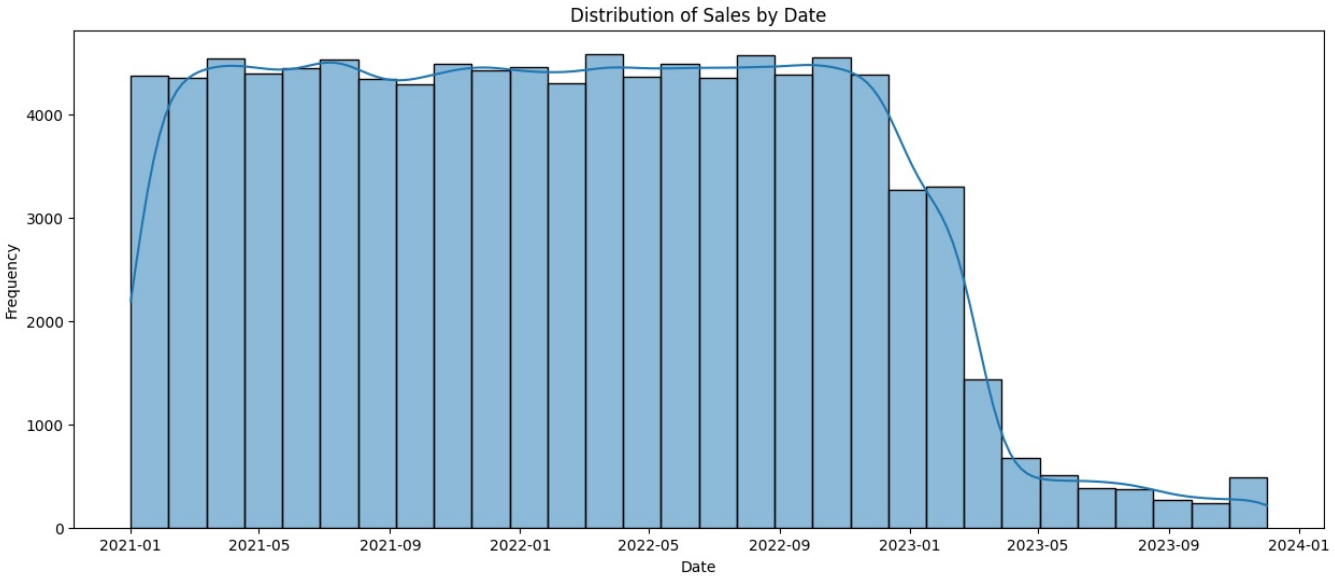
In [80]:
```
sales_by_month_category
```

Out[80]:

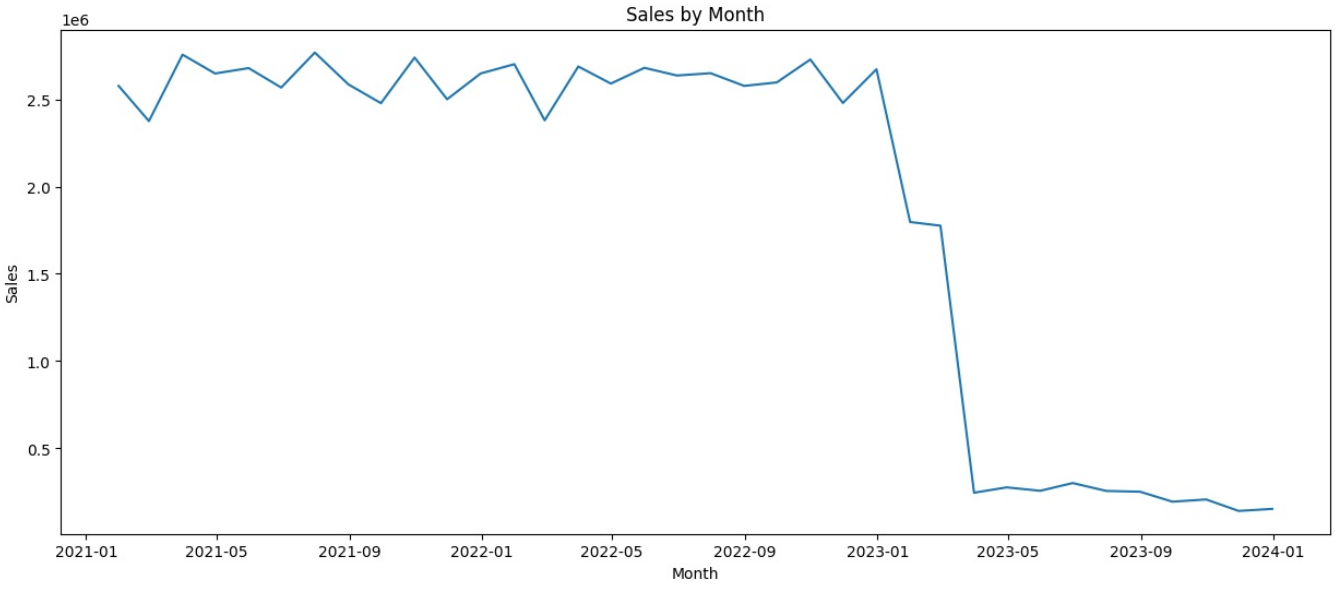
	month	category	price
0	1	Books	23603.70
1	1	Clothing	3173646.08
2	1	Cosmetics	199030.70
3	1	Food & Beverage	23508.85
4	1	Shoes	1864728.19
...	...	...	...
91	12	Food & Beverage	18984.90
92	12	Shoes	1447009.87
93	12	Souvenir	13407.39
94	12	Technology	1228500.00
95	12	Toys	93578.24

96 rows × 3 columns

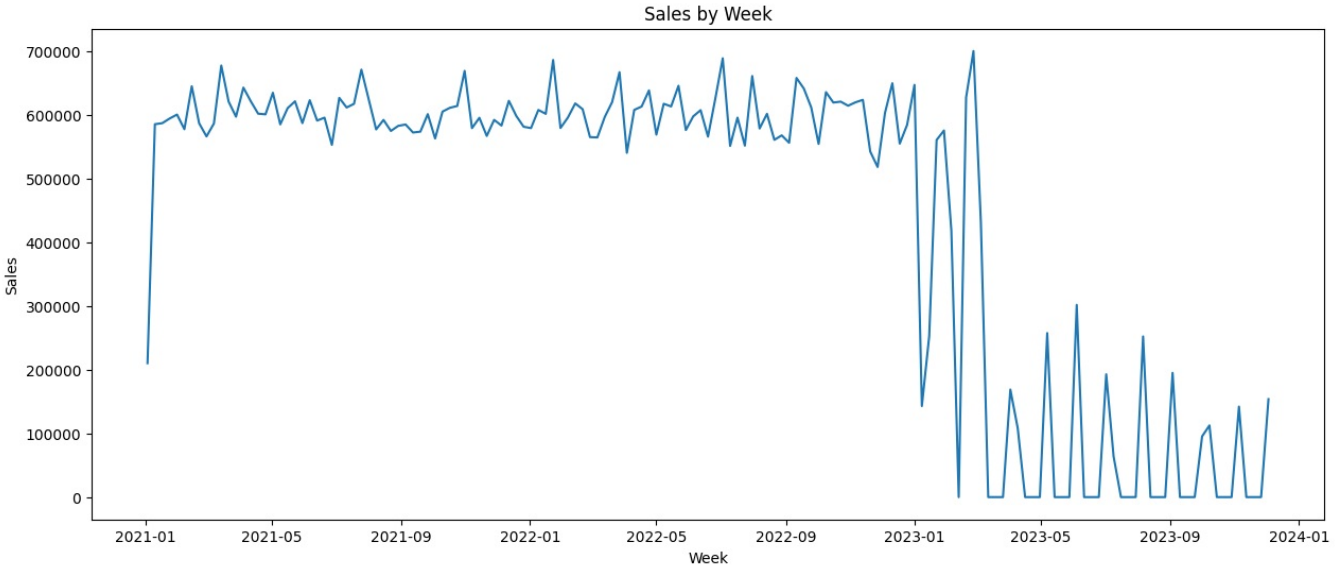
In [81]:
```
plt.figure(figsize=(15,6))
sns.histplot(df['invoice_date'], bins=30, kde=True)
plt.xlabel('Date')
plt.ylabel('Frequency')
plt.title('Distribution of Sales by Date')
plt.show()
```



In [82]:
```
sales_by_month = df.resample('M', on='invoice_date')['price'].sum()

plt.figure(figsize=(15,6))
plt.plot(sales_by_month.index, sales_by_month.values)
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales by Month')
plt.show()
```



In [83]:
```
sales_by_month = df.resample('W', on='invoice_date')['price'].sum()

plt.figure(figsize=(15,6))
plt.plot(sales_by_month.index, sales_by_month.values)
plt.xlabel('Week')
plt.ylabel('Sales')
plt.title('Sales by Week')
plt.show()
```



In [84]:
```
df.head()
```

Out[84]:

	invoice_no	customer_id	gender	age	category	quantity	price	payment_method	invoice_date	shopping_mall	month
0	I138884	C241288	Female	28	Clothing	5	1500.40	Credit Card	2022-05-08	Kanyon	5
1	I317333	C111565	Male	21	Shoes	3	1800.51	Debit Card	2021-12-12	Forum Istanbul	12
2	I127801	C266599	Male	20	Clothing	1	300.08	Cash	2021-09-11	Metrocity	9
3	I173702	C988172	Female	66	Shoes	5	3000.85	Credit Card	2021-05-16	Metropol AVM	5
4	I337046	C189076	Female	53	Books	4	60.60	Cash	2021-10-24	Kanyon	10

In [85]:
```
data = pd.get_dummies(df, columns=['gender', 'category', 'payment_method', 'shopping_mall'])
```

In [86]:
```
data.head()
```

Out[86]:

	invoice_no	customer_id	age	quantity	price	invoice_date	month	gender_Female	gender_Male	category_Books	...	shopping_mall_C
0	I138884	C241288	28	5	1500.40	2022-05-08	5	1	0	0	...	
1	I317333	C111565	21	3	1800.51	2021-12-12	12	0	1	0	...	
2	I127801	C266599	20	1	300.08	2021-09-11	9	0	1	0	...	
3	I173702	C988172	66	5	3000.85	2021-05-16	5	1	0	0	...	
4	I337046	C189076	53	4	60.60	2021-10-24	10	1	0	1	...	

5 rows × 30 columns

In [87]:
```
data.tail()
```

Out[87]:

	invoice_no	customer_id	age	quantity	price	invoice_date	month	gender_Female	gender_Male	category_Books	...	shopping_ma
99452	I219422	C441542	45	5	58.65	2022-09-21	9	1	0	0	...	
99453	I325143	C569580	27	2	10.46	2021-09-22	9	0	1	0	...	
99454	I824010	C103292	63	2	10.46	2021-03-28	3	0	1	0	...	
99455	I702964	C800631	56	4	4200.00	2021-03-16	3	0	1	0	...	
99456	I232867	C273973	36	3	35.19	2022-10-15	10	1	0	0	...	

5 rows × 30 columns

In [88]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99457 entries, 0 to 99456
Data columns (total 30 columns):
 #   Column                          Non-Null Count  Dtype         
---  ------                          --------------  -----         
 0   invoice_no                      99457 non-null  object        
 1   customer_id                     99457 non-null  object        
 2   age                             99457 non-null  int64         
 3   quantity                        99457 non-null  int64         
 4   price                           99457 non-null  float64       
 5   invoice_date                    99457 non-null  datetime64[ns]
 6   month                           99457 non-null  int64         
 7   gender_Female                   99457 non-null  uint8         
 8   gender_Male                     99457 non-null  uint8         
 9   category_Books                  99457 non-null  uint8         
 10  category_Clothing               99457 non-null  uint8         
 11  category_Cosmetics              99457 non-null  uint8         
 12  category_Food & Beverage        99457 non-null  uint8         
 13  category_Shoes                  99457 non-null  uint8         
 14  category_Souvenir               99457 non-null  uint8         
 15  category_Technology             99457 non-null  uint8         
 16  category_Toys                   99457 non-null  uint8         
 17  payment_method_Cash             99457 non-null  uint8         
 18  payment_method_Credit Card      99457 non-null  uint8         
 19  payment_method_Debit Card       99457 non-null  uint8         
 20  shopping_mall_Cevahir AVM       99457 non-null  uint8         
 21  shopping_mall_Emaar Square Mall 99457 non-null  uint8         
 22  shopping_mall_Forum Istanbul    99457 non-null  uint8         
 23  shopping_mall_Istinye Park      99457 non-null  uint8         
 24  shopping_mall_Kanyon            99457 non-null  uint8         
 25  shopping_mall_Mall of Istanbul  99457 non-null  uint8         
 26  shopping_mall_Metrocity         99457 non-null  uint8         
 27  shopping_mall_Metropol AVM      99457 non-null  uint8         
 28  shopping_mall_Viaport Outlet    99457 non-null  uint8         
 29  shopping_mall_Zorlu Center      99457 non-null  uint8         
dtypes: datetime64[ns](1), float64(1), int64(3), object(2), uint8(23)
memory usage: 7.5+ MB
```

In [89]:
```
df_clean = data.drop(['invoice_no', 'customer_id', 'invoice_date'], axis=1)
```

In [90]:
```
df_clean.columns
```

Out[90]:
```
Index(['age', 'quantity', 'price', 'month', 'gender_Female', 'gender_Male',
       'category_Books', 'category_Clothing', 'category_Cosmetics',
       'category_Food & Beverage', 'category_Shoes', 'category_Souvenir',
       'category_Technology', 'category_Toys', 'payment_method_Cash',
       'payment_method_Credit Card', 'payment_method_Debit Card',
       'shopping_mall_Cevahir AVM', 'shopping_mall_Emaar Square Mall',
       'shopping_mall_Forum Istanbul', 'shopping_mall_Istinye Park',
       'shopping_mall_Kanyon', 'shopping_mall_Mall of Istanbul',
       'shopping_mall_Metrocity', 'shopping_mall_Metropol AVM',
       'shopping_mall_Viaport Outlet', 'shopping_mall_Zorlu Center'],
      dtype='object')
```

In [91]:
```
df_corr = df_clean.corr()
```

In [92]:
```
df_corr
```

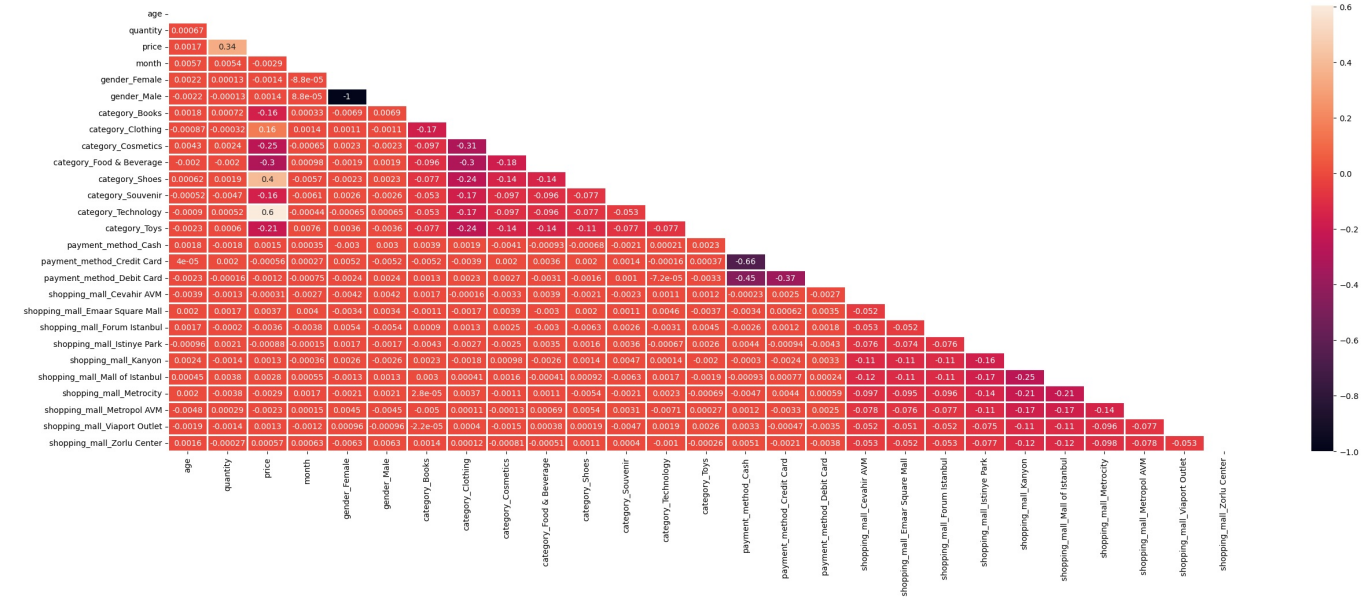
Out[92]:

	age	quantity	price	month	gender_Female	gender_Male	category_Books	category_Clothing	categor
age	1.000000	0.000667	0.001694	0.005692	0.002150	-0.002150	0.001758	-0.000868	
quantity	0.000667	1.000000	0.344880	0.005353	0.000131	-0.000131	0.000715	-0.000318	
price	0.001694	0.344880	1.000000	-0.002933	-0.001450	0.001450	-0.157036	0.163976	
month	0.005692	0.005353	-0.002933	1.000000	-0.000088	0.000088	0.000334	0.001398	
gender_Female	0.002150	0.000131	-0.001450	-0.000088	1.000000	-1.000000	-0.006861	0.001140	
gender_Male	-0.002150	-0.000131	0.001450	0.000088	-1.000000	1.000000	0.006861	-0.001140	
category_Books	0.001758	0.000715	-0.157036	0.000334	-0.006861	0.006861	1.000000	-0.167290	
category_Clothing	-0.000868	-0.000318	0.163976	0.001398	0.001140	-0.001140	-0.167290	1.000000	
category_Cosmetics	0.004334	0.002424	-0.254765	-0.000651	0.002342	-0.002342	-0.097135	-0.308211	
category_Food & Beverage	-0.002025	-0.002034	-0.298954	0.000982	-0.001905	0.001905	-0.095914	-0.304338	
category_Shoes	0.000623	0.001904	0.397954	-0.005687	-0.002316	0.002316	-0.076915	-0.244053	
category_Souvenir	-0.000522	-0.004662	-0.159944	-0.006091	0.002559	-0.002559	-0.052823	-0.167608	
category_Technology	-0.000899	0.000517	0.602977	-0.000441	-0.000652	0.000652	-0.052806	-0.167555	
category_Toys	-0.002260	0.000599	-0.207577	0.007573	0.003552	-0.003552	-0.077141	-0.244769	
payment_method_Cash	0.001819	-0.001766	0.001497	0.000346	-0.003024	0.003024	0.003895	0.001864	
payment_method_Credit Card	0.000040	0.001971	-0.000558	0.000271	0.005151	-0.005151	-0.005158	-0.003869	
payment_method_Debit Card	-0.002300	-0.000157	-0.001190	-0.000752	-0.002381	0.002381	0.001310	0.002292	
shopping_mall_Cevahir AVM	-0.003904	-0.001340	-0.000313	-0.002688	-0.004223	0.004223	0.001698	-0.000159	
shopping_mall_Emaar Square Mall	0.002024	0.001708	0.003704	0.003971	-0.003374	0.003374	-0.001062	-0.001696	
shopping_mall_Forum Istanbul	0.001685	-0.000195	-0.003620	-0.003813	0.005411	-0.005411	0.000900	0.001323	
shopping_mall_Istinye Park	-0.000958	0.002114	-0.000880	-0.000147	0.001674	-0.001674	-0.004312	-0.002738	
shopping_mall_Kanyon	0.002392	-0.001424	0.001274	-0.000357	0.002593	-0.002593	0.002334	-0.001781	
shopping_mall_Mall of Istanbul	0.000447	0.003850	0.002826	0.000546	-0.001294	0.001294	0.003018	0.000407	
shopping_mall_Metrocity	0.002031	-0.003786	-0.002878	0.001671	-0.002096	0.002096	0.000028	0.003712	
shopping_mall_Metropol AVM	-0.004821	0.000286	-0.002312	0.000153	0.004539	-0.004539	-0.005005	0.000115	
shopping_mall_Viaport Outlet	-0.001949	-0.001407	0.001332	-0.001157	0.000955	-0.000955	-0.000022	0.000395	
shopping_mall_Zorlu Center	0.001626	-0.000272	0.000567	0.000625	-0.006262	0.006262	0.001432	0.000118	

27 rows × 27 columns

In [93]:

```
plt.figure(figsize=(30, 10))
matrix = np.triu(df_corr)
sns.heatmap(df_corr, annot=True, linewidth=.8, mask=matrix, cmap="rocket");
plt.show()
```



In [94]:
```
corr = df_corr
```

In [95]:
```
high_corr_features = set()
for i in range(len(corr.columns)):
    for j in range(i):
        if abs(corr.iloc[i, j]) >= 0.1:
            high_corr_features.add((corr.columns[i], corr.columns[j], corr.iloc[i, j]))

# Print highly correlated features
for feature_pair in high_corr_features:
    print(feature_pair)
```

```
('category_Cosmetics', 'price', -0.2547652359031358)
('payment_method_Debit Card', 'payment_method_Credit Card', -0.37004888604589137)
('shopping_mall_Metropol AVM', 'shopping_mall_Istinye Park', -0.11140527073761701)
('shopping_mall_Metropol AVM', 'shopping_mall_Metrocity', -0.14222223332394512)
('shopping_mall_Viaport Outlet', 'shopping_mall_Kanyon', -0.11374657056401054)
('category_Souvenir', 'price', -0.15994388350871744)
('category_Clothing', 'category_Books', -0.16728954396122633)
('shopping_mall_Kanyon', 'shopping_mall_Istinye Park', -0.16477409703833323)
('shopping_mall_Metrocity', 'shopping_mall_Istinye Park', -0.1392415298731586)
('category_Cosmetics', 'category_Clothing', -0.3082111295659363)
('payment_method_Debit Card', 'payment_method_Cash', -0.45208632228464174)
('shopping_mall_Metrocity', 'shopping_mall_Kanyon', -0.21035396188678884)
('category_Shoes', 'category_Cosmetics', -0.14170640242153562)
('shopping_mall_Mall of Istanbul', 'shopping_mall_Kanyon', -0.2498669807009949)
('category_Technology', 'category_Clothing', -0.1675545479967624)
('shopping_mall_Mall of Istanbul', 'shopping_mall_Cevahir AVM', -0.11511435003055782)
('payment_method_Credit Card', 'payment_method_Cash', -0.6613609394784303)
('gender_Male', 'gender_Female', -1.0)
('category_Shoes', 'category_Food & Beverage', -0.1399258271147214)
('shopping_mall_Zorlu Center', 'shopping_mall_Mall of Istanbul', -0.11613065597136397)
('category_Toys', 'category_Food & Beverage', -0.14033648147692132)
('shopping_mall_Kanyon', 'shopping_mall_Cevahir AVM', -0.11468099355722823)
('category_Shoes', 'category_Clothing', -0.24405274998490123)
('category_Food & Beverage', 'price', -0.29895432512912895)
('shopping_mall_Kanyon', 'shopping_mall_Forum Istanbul', -0.11414778765900538)
('category_Clothing', 'price', 0.16397642452323746)
('category_Books', 'price', -0.1570363233350594)
('category_Toys', 'category_Clothing', -0.2447689960736651)
('shopping_mall_Kanyon', 'shopping_mall_Emaar Square Mall', -0.11248690598525163)
('category_Food & Beverage', 'category_Clothing', -0.30433838198918334)
('category_Technology', 'price', 0.6029774026353889)
('category_Toys', 'category_Shoes', -0.11253757742875245)
('shopping_mall_Metropol AVM', 'shopping_mall_Kanyon', -0.168301368823485)
('category_Souvenir', 'category_Clothing', -0.167607508612874)
('shopping_mall_Metrocity', 'shopping_mall_Mall of Istanbul', -0.21114884731851244)
('shopping_mall_Metropol AVM', 'shopping_mall_Mall of Istanbul', -0.1689373459404173)
('category_Toys', 'category_Cosmetics', -0.14212228241671207)
('shopping_mall_Mall of Istanbul', 'shopping_mall_Emaar Square Mall', -0.11291197144170896)
('category_Food & Beverage', 'category_Cosmetics', -0.176710556357773)
('category_Shoes', 'price', 0.39795410526458325)
('shopping_mall_Viaport Outlet', 'shopping_mall_Mall of Istanbul', -0.11417639604026371)
('shopping_mall_Zorlu Center', 'shopping_mall_Kanyon', -0.11569347353925402)
('shopping_mall_Mall of Istanbul', 'shopping_mall_Forum Istanbul', -0.11457912925419063)
('price', 'quantity', 0.34487984279119643)
('shopping_mall_Mall of Istanbul', 'shopping_mall_Istinye Park', -0.1653967453026517)
('category_Toys', 'price', -0.2075771357500389)
```

In [96]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

In [97]:
```
X = df_clean[['age', 'quantity', 'price', 'month', 'gender_Female', 'gender_Male',
```

```
       'category_Books', 'category_Clothing', 'category_Cosmetics',
       'category_Food & Beverage', 'category_Shoes', 'category_Souvenir',
       'category_Technology', 'category_Toys', 'payment_method_Cash',
       'payment_method_Credit Card', 'payment_method_Debit Card',
       'shopping_mall_Cevahir AVM', 'shopping_mall_Emaar Square Mall',
       'shopping_mall_Forum Istanbul', 'shopping_mall_Istinye Park',
       'shopping_mall_Kanyon', 'shopping_mall_Mall of Istanbul',
       'shopping_mall_Metrocity', 'shopping_mall_Metropol AVM',
       'shopping_mall_Viaport Outlet', 'shopping_mall_Zorlu Center']]

# Standardize data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply PCA
pca = PCA(n_components=10)
X_pca = pca.fit_transform(X_scaled)
```

In [98]:
```
pca
```

Out[98]:
```
PCA(n_components=10)
```

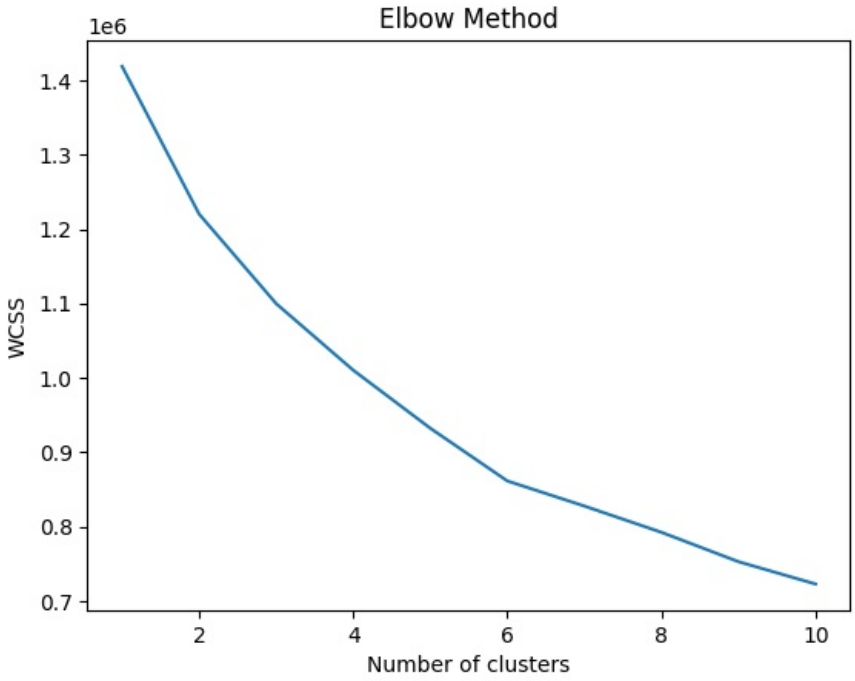
In [99]:
```
X_pca
```

Out[99]:
```
array([[ 0.90410668,  1.48338836,  1.46050333, ..., -0.19205227,
        -0.4437191 , -0.00647939],
       [ 2.14846798, -1.24800423,  0.54459106, ..., -0.01390102,
         1.15600331, -0.08629919],
       [ 0.28672153, -1.75948768, -1.34415613, ...,  0.47128544,
        -1.36926011,  0.01055116],
       ...,
       [-0.88341471, -2.02202801,  0.52520975, ..., -1.49541185,
        -2.08867679, -0.67026422],
       [ 4.92828777, -0.37186644, -1.33248362, ..., -0.31206584,
         0.8138678 ,  0.74095777],
       [-1.34635676,  0.81532148,  1.49511138, ..., -0.0816399 ,
        -0.15349793,  0.40821504]])
```

In [100…
```
from sklearn.cluster import KMeans
```

In [101…
```
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X_pca)
    wcss.append(kmeans.inertia_)
    print(f"n_clusters: {i}, WCSS: {kmeans.inertia_}")
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

```
n_clusters: 1, WCSS: 1418779.4279590878
n_clusters: 2, WCSS: 1220109.5862052466
n_clusters: 3, WCSS: 1099684.4103859991
n_clusters: 4, WCSS: 1010147.1733832029
n_clusters: 5, WCSS: 932113.7781117349
n_clusters: 6, WCSS: 861197.1181592819
n_clusters: 7, WCSS: 827444.8582237436
n_clusters: 8, WCSS: 792128.922317006
n_clusters: 9, WCSS: 752610.3378023453
n_clusters: 10, WCSS: 722587.1864294253
```



In [102…
```
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=42)
kmeans.fit(X_pca)
```

Out[102]:
```
KMeans(n_clusters=4, random_state=42)
```

In [103…
```
labels = kmeans.labels_
centers = kmeans.cluster_centers_
```

In [104…
```
labels
```

Out[104]:
```
array([1, 3, 2, ..., 2, 3, 0])
```

In [105…
```
centers
```

Out[105]:
```
array([[-1.30417363e+00,  8.45782841e-01, -7.03161215e-03,
         6.34169572e-01,  5.41346828e-02,  2.81810761e-02,
         1.49681948e-02,  2.21542053e-02, -3.26130190e-02,
         6.19550377e-02],
       [ 2.15988105e-01,  1.26293844e+00, -5.50447101e-02,
        -1.53046867e+00, -7.38577566e-02, -1.04280766e-02,
        -5.53331946e-02,  3.41356537e-04,  8.58568481e-03,
        -7.55056920e-02],
       [ 9.29772852e-02, -1.76051975e+00,  2.65748911e-02,
        -2.03960391e-01, -2.06401759e-02, -1.58786690e-03,
        -7.93303823e-03,  2.00519797e-02,  7.77832052e-03,
        -1.31982103e-03],
       [ 2.67629977e+00,  8.25407621e-01,  3.36726332e-02,
         1.56299574e+00,  4.69975196e-02, -4.93342885e-02,
         7.83317034e-02, -1.14918187e-01,  4.57495028e-02,
        -2.60647949e-02]])
```

In [106…
```
data_new = df_clean.copy()
```

In [107…
```
data_new.columns
```

Out[107]:
```
Index(['age', 'quantity', 'price', 'month', 'gender_Female', 'gender_Male',
       'category_Books', 'category_Clothing', 'category_Cosmetics',
       'category_Food & Beverage', 'category_Shoes', 'category_Souvenir',
       'category_Technology', 'category_Toys', 'payment_method_Cash',
       'payment_method_Credit Card', 'payment_method_Debit Card',
       'shopping_mall_Cevahir AVM', 'shopping_mall_Emaar Square Mall',
       'shopping_mall_Forum Istanbul', 'shopping_mall_Istinye Park',
       'shopping_mall_Kanyon', 'shopping_mall_Mall of Istanbul',
       'shopping_mall_Metrocity', 'shopping_mall_Metropol AVM',
       'shopping_mall_Viaport Outlet', 'shopping_mall_Zorlu Center'],
      dtype='object')
```

In [108…
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Splitting the data into training and test sets
X = data_new.drop('price', axis=1)
y = data_new['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)

# Creating a list of regression models to test
models = [LinearRegression(), DecisionTreeRegressor(), RandomForestRegressor()]

# Creating a list of model names for display purposes
model_names = ['Linear Regression', 'Decision Tree', 'Random Forest']

# Looping through the models and printing the performance metrics for each
for i, model in enumerate(models):
    print(model_names[i] + ':')
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print('MSE: ', mean_squared_error(y_test, y_pred))
    print('R-squared: ', r2_score(y_test, y_pred))
    print('-------------------------------------')
```

```
Linear Regression:
MSE:  1.66614137844201e+26
R-squared:  -1.8476769465760363e+20
-------------------------------------
Decision Tree:
MSE:  3.1632169718420855e-21
R-squared:  1.0
-------------------------------------
Random Forest:
MSE:  3.732837180415106e-22
R-squared:  1.0
-------------------------------------
```

In [109…
```
from sklearn.linear_model import LogisticRegression
```

In [110…
```
X_train, X_test, y_train, y_test = train_test_split(data_new.drop('payment_method_Cash', axis=1),
                                                    data_new['payment_method_Cash'], test_size=0.2,
                                                    random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)

# train a logistic regression model
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

from sklearn.metrics import accuracy_score

# predict on test set and calculate accuracy
y_pred = logreg.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 1.0
```

In [111…
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

# Create Decision Tree classifier with max depth of 3
dt = DecisionTreeClassifier(max_depth=3)
```

```
# Train the classifier
dt.fit(X_train, y_train)

# Make predictions on the test set
y_pred = dt.predict(X_test)

# Evaluate accuracy
acc = accuracy_score(y_test, y_pred)
print("Decision Tree Accuracy:", acc)

# Create Random Forest classifier with 100 trees
rf = RandomForestClassifier(n_estimators=100)

# Train the classifier
rf.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf.predict(X_test)

# Evaluate accuracy
acc = accuracy_score(y_test, y_pred)
print("Random Forest Accuracy:", acc)
```

```
Decision Tree Accuracy: 1.0
Random Forest Accuracy: 1.0
```

In [ ]: