In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
data = np.loadtxt(r"C:\Users\akash.bana\Desktop\Akash_backup\Akash\Scaler\Prob & Stats\P
data
```

Out[2]:

```
array(['User_ID,Product_ID,Gender,Age,Occupation,City_Category,Stay_In_Cu
rrent_City_Years,Marital_Status,Product_Category,Purchase',
       '1000001,P00069042,F,0-17,10,A,2,0,3,8370',
       '1000001,P00248942,F,0-17,10,A,2,0,1,15200', ...,
       '1006036,P00375436,F,26-35,15,B,4+,1,20,137',
       '1006038,P00375436,F,55+,1,C,2,0,20,365',
       '1006039,P00371644,F,46-50,0,B,4+,1,20,490'], dtype='<U122')
```

In [3]:

```
df = pd.DataFrame(data)
df = df[0].str.split(',',expand=True)
df.columns = df.loc[0]
df = df.drop(index=0)
df.head()
```

Out[3]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Year
1	1000001	P00069042	F	0-17	10	A	
2	1000001	P00248942	F	0-17	10	A	
3	1000001	P00087842	F	0-17	10	A	
4	1000001	P00085442	F	0-17	10	A	
5	1000002	P00285442	M	55+	16	C	4

In [4]:

```
df = df.reset_index(drop=True)
df.head()
```

Out[4]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Year
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	4

Changing the datatype for required columns

In [5]:

```
df = df.astype({'User_ID':'int','Occupation':'int','Marital_Status':'int','Product_Categ
df.head()
```

Out[5]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Year
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	4

In [89]:

```
df.shape
```

Out[89]:

```
(550068, 10)
```

In [86]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     550068 non-null  int32
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int32
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int32
 8   Product_Category            550068 non-null  int32
 9   Purchase                    550068 non-null  int32
dtypes: int32(5), object(5)
memory usage: 31.5+ MB
```

In [90]:

```
df.describe()
```

Out[90]:

	User_ID	Occupation	Marital_Status	Product_Category	Purchase
count	5.500680e+05	550068.000000	550068.000000	550068.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9263.968713
std	1.727592e+03	6.522660	0.491770	3.936211	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	12.000000
25%	1.001516e+06	2.000000	0.000000	1.000000	5823.000000
50%	1.003077e+06	7.000000	0.000000	5.000000	8047.000000
75%	1.004478e+06	14.000000	1.000000	8.000000	12054.000000
max	1.006040e+06	20.000000	1.000000	20.000000	23961.000000

In [91]:

```
df.describe(include='object')
```

Out[91]:

	Product_ID	Gender	Age	City_Category	Stay_In_Current_City_Years
count	550068	550068	550068	550068	550068
unique	3631	2	7	3	5
top	P00265242	M	26-35	B	1
freq	1880	414259	219587	231173	193821

In [93]:

```
df.nunique()
```

Out[93]:

```
0
User_ID                       5891
Product_ID                    3631
Gender                           2
Age                              7
Occupation                      21
City_Category                    3
Stay_In_Current_City_Years       5
Marital_Status                   2
Product_Category                20
Purchase                     18105
dtype: int64
```

In [94]:

```
df['Gender'].value_counts()
```

Out[94]:

```
M    414259
F    135809
Name: Gender, dtype: int64
```

In [95]:

```
df['Age'].value_counts()
```

Out[95]:

```
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
```

In [96]:

```
df['City_Category'].value_counts()
```

Out[96]:

```
B    231173
C    171175
A    147720
Name: City_Category, dtype: int64
```

In [97]:

```
df['Stay_In_Current_City_Years'].value_counts()
```

Out[97]:

```
1     193821
2     101838
3      95285
4+     84726
0      74398
Name: Stay_In_Current_City_Years, dtype: int64
```

In [98]:

```
df['Marital_Status'].value_counts()
```

Out[98]:

```
0    324731
1    225337
Name: Marital_Status, dtype: int64
```

In [99]:

```
df['Product_Category'].value_counts()
```

Out[99]:

```
5     150933
1     140378
8     113925
11     24287
2      23864
6      20466
3      20213
4      11753
16      9828
15      6290
13      5549
10      5125
12      3947
7       3721
18      3125
20      2550
19      1603
14      1523
17       578
9        410
Name: Product_Category, dtype: int64
```

In [6]:

```
def M_status(val):
    if val == 0:
        return "unmarried"
    else:
        return 'married'
df['M_Status'] = df['Marital_Status'].apply(M_status)
df = df.drop(columns=['Marital_Status'])
df.head()
```

Out[6]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Year
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	4

Categorization based on age

In [7]:

```
def Age_status(val):
    if val == '0-17':
        return "children"
    elif val =='18-25':
        return 'young adult'
    elif val == '26-35':
        return 'adult'
    elif val == '36-45':
        return 'mid-age adult'
    elif val == '46-50':
        return 'older mid-age adult'
    elif val == '51-55':
        return 'old adult'
    elif val == '55+':
        return 'older adult'
df['age_category'] = df['Age'].apply(Age_status)
df.head()
```

Out[7]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Year
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	4

Insights:

1. Changed the data type of columns: {'User_ID':'int','Occupation':'int','Marital_Status':'int','Product_Category':'int','Purchase':'int'}
2. Categorized the age column based on age
3. Two genders: Male & Female
4. Marital status: 0 --> Not married and 1 --> married
5. City categories: A, B & C

Uni-variate analysis

In [183]:

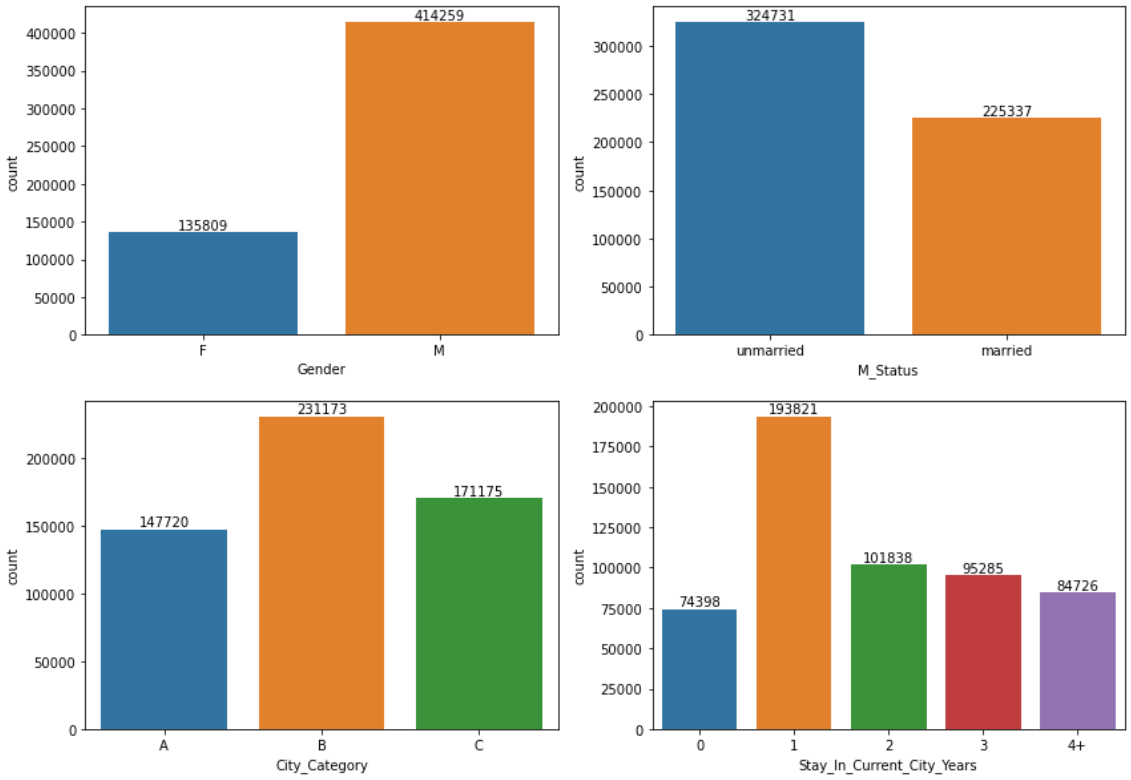
```
df.head()
```

Out[183]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Year
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	4

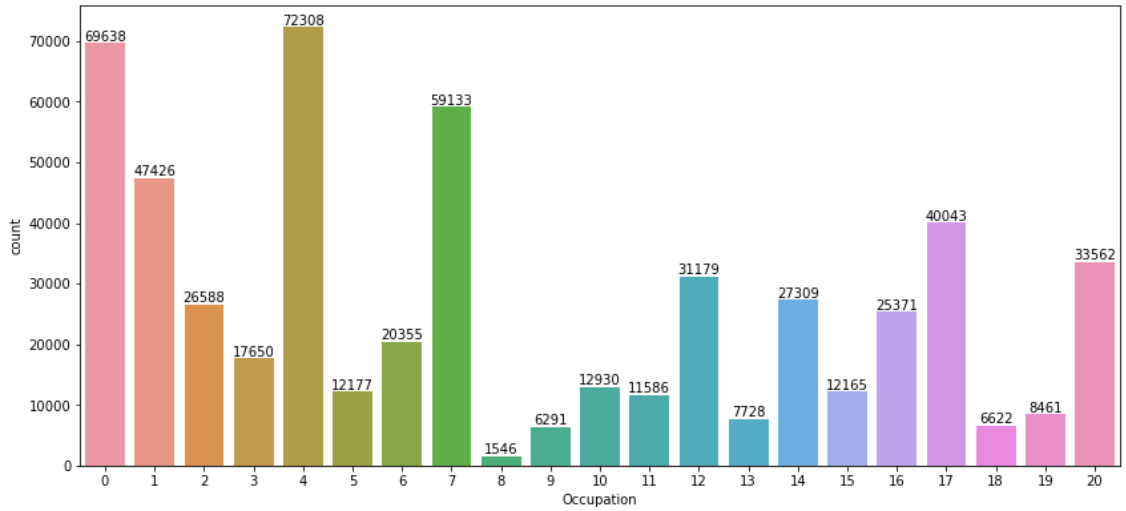
In [216]:

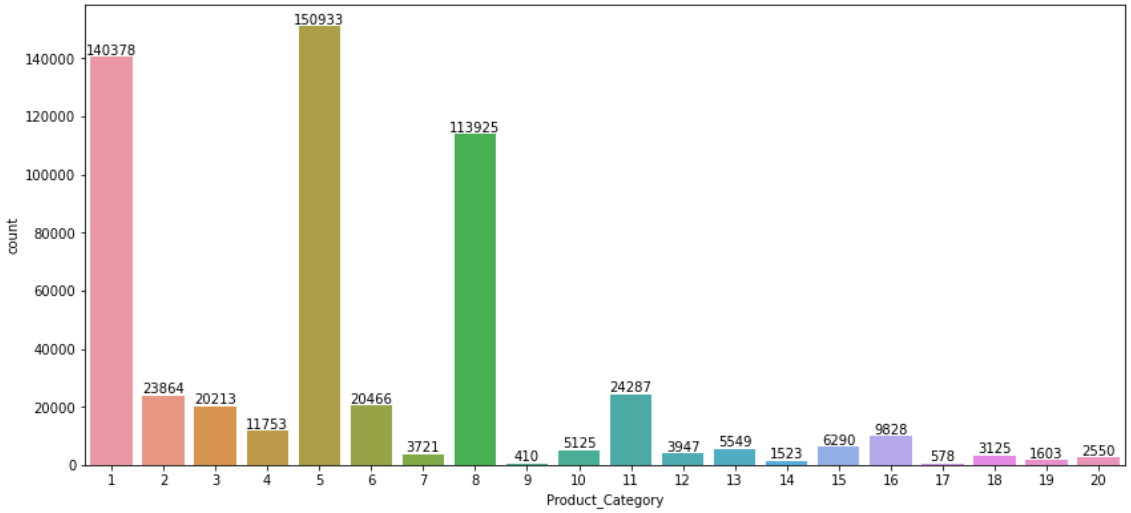
```
plt.figure(figsize=(14,10))
plt.subplot(2,2,1)
x = sns.countplot(data=df,x='Gender')
for i in x.containers:
    x.bar_label(i)
plt.subplot(2,2,2)
y = sns.countplot(data=df,x='M_Status')
for i in y.containers:
    y.bar_label(i)
plt.subplot(2,2,3)
df1 = df.sort_values(by=['City_Category'],ascending=True)
z = sns.countplot(data=df1,x='City_Category')
for i in z.containers:
    z.bar_label(i)
plt.subplot(2,2,4)
df2 = df.sort_values(by=['Stay_In_Current_City_Years'],ascending=True)
a = sns.countplot(data=df2,x='Stay_In_Current_City_Years')
for i in a.containers:
    a.bar_label(i)
plt.show()
```



In [190]:

```
plt.figure(figsize=(14,14))
plt.subplot(2,1,1)
y = sns.countplot(data=df,x='Occupation')
for i in y.containers:
    y.bar_label(i)
plt.subplot(2,1,2)
z = sns.countplot(data=df,x='Product_Category')
for i in z.containers:
    z.bar_label(i)
plt.show()
```





In [155]:

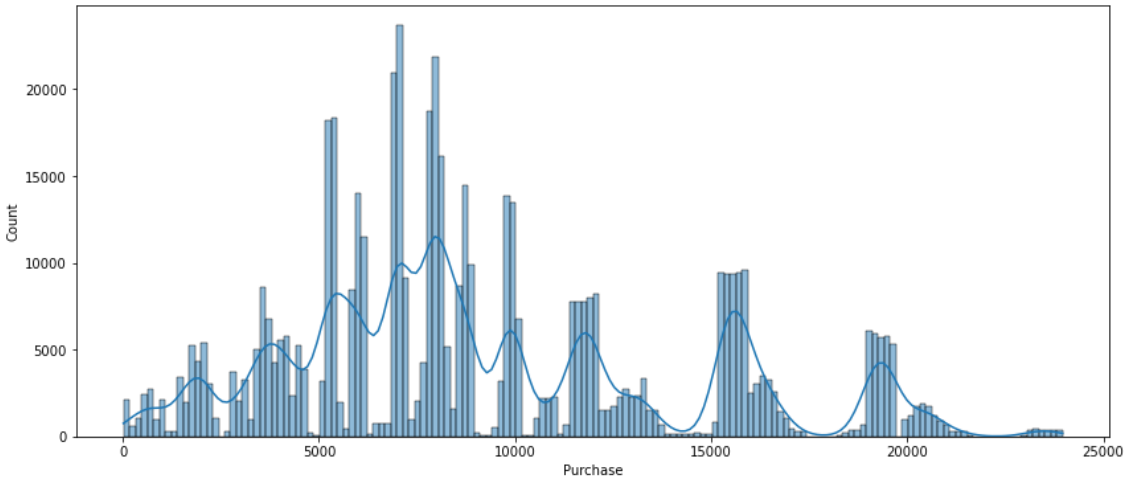
```
x = df.loc[(df['Product_Category']==1) | (df['Product_Category']==5) | (df['Product_Cate
perc = int(x/df.shape[0]*100)
perc
```

Out[155]:

73

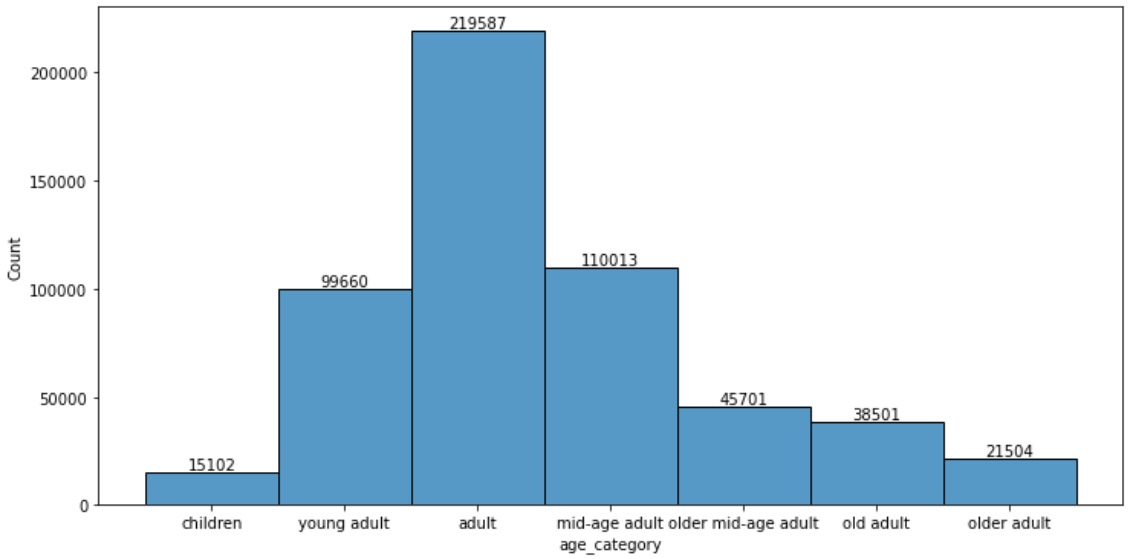
In [210]:

```
plt.figure(figsize=(14,6))
sns.histplot(data=df,x='Purchase',kde=True)
plt.show()
```



In [304]:

```
df1 = df
def age_rank(val):
     if val == '0-17':
        return 1
     elif val =='18-25':
        return 2
     elif val == '26-35':
        return 3
     elif val == '36-45':
        return 4
     elif val == '46-50':
        return 5
     elif val == '51-55':
        return 6
     elif val == '55+':
        return 7
df1['age_rank'] = df1['Age'].apply(age_rank)
df1 = df1.sort_values(by=['age_rank'],ascending=True)
plt.figure(figsize=(12,6))
x = sns.histplot(data=df1,x='age_category')
for i in x.containers:
    x.bar_label(i,)
plt.show()
```



In [156]:

```
x = df.loc[(df['age_category']=='young adult') | (df['age_category']=='adult') | (df['ag
perc = int(x/df.shape[0]*100)
perc
```

Out[156]:

78

Insights:

Purchasing details:

1. Gender: Male customers > Female customers
2. Marital status: Unmarried > married
3. City: B > C > A
4. Most sold product categories: 1, 5 & 8
5. Purchase / product: between 5000 to 10,000
6. Age: between 18 and 45

Bi-variate analysis

In [217]:

```
df.head()
```

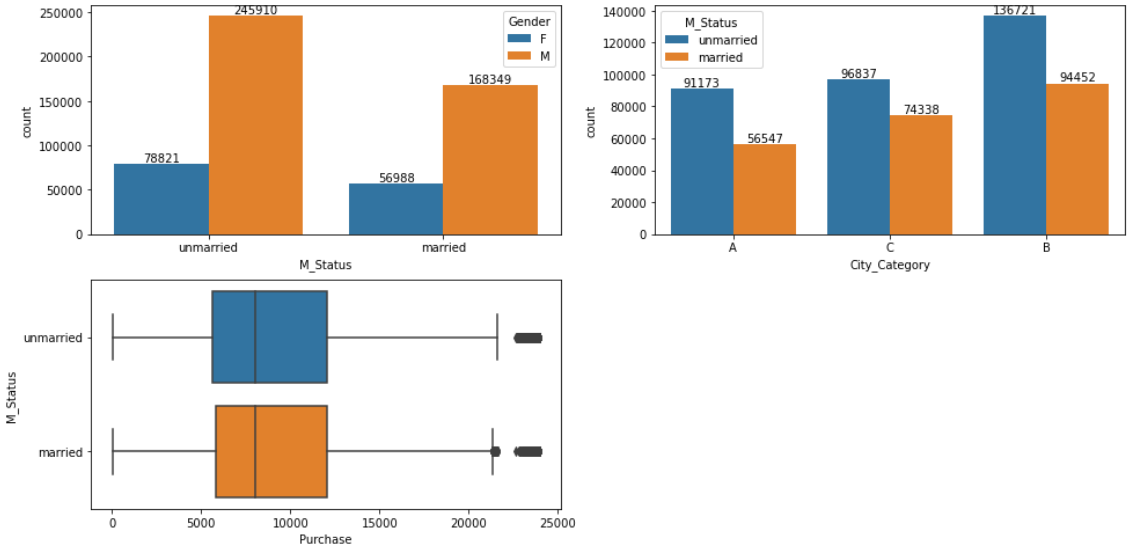
Out[217]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Year
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	4

Impact of marital status

In [236]:

```
plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
x = sns.countplot(data=df,x='M_Status',hue='Gender')
for i in x.containers:
    x.bar_label(i,)
plt.subplot(2,2,2)
y = sns.countplot(data=df,x='City_Category',hue='M_Status')
for i in y.containers:
    y.bar_label(i,)
plt.subplot(2,2,3)
sns.boxplot(data=df,y='M_Status',x='Purchase')
plt.show()
```



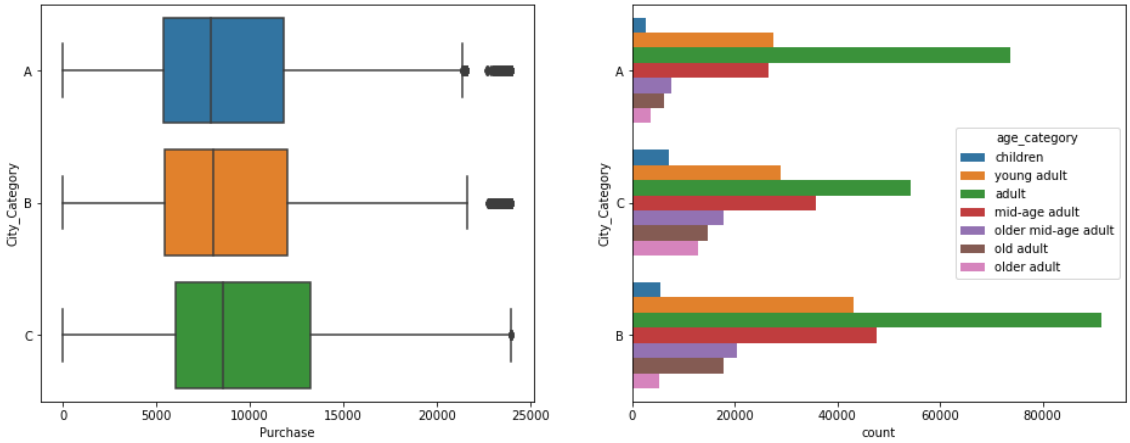
Insights:

1. Be it married or unmarried, men tend to purchase more than women
2. Irrespective of the cities, unmarried people tend to purchase more
3. Average amount spent per product is similar irrespective of marital status

Impact of city

In [261]:

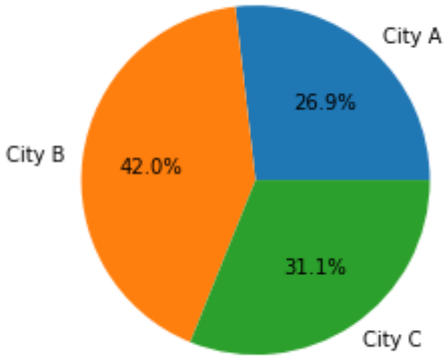
```
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
df1 = df.sort_values(by=['City_Category'],ascending=True)
sns.boxplot(data=df1,x='Purchase',y='City_Category')
plt.subplot(1,2,2)
df2 = df.sort_values(by=['age_rank'],ascending=True)
sns.countplot(data=df2,y='City_Category',hue='age_category')
plt.show()
```



In [22]:

```
city_A = df.groupby('City_Category')['City_Category'].count()[0]
city_B = df.groupby('City_Category')['City_Category'].count()[1]
city_C = df.groupby('City_Category')['City_Category'].count()[2]
sizes = [city_A,city_B,city_C]
labels = ['City A', 'City B', 'City C']
print(f"Number of transactions in city A: {city_A}")
print(f"Number of transactions in city B: {city_B}")
print(f"Number of transactions in city C: {city_C}")
plt.pie(sizes, labels = labels,autopct='%1.1f%%')
plt.show()
```

```
Number of transactions in city A: 147720
Number of transactions in city B: 231173
Number of transactions in city C: 171175
```



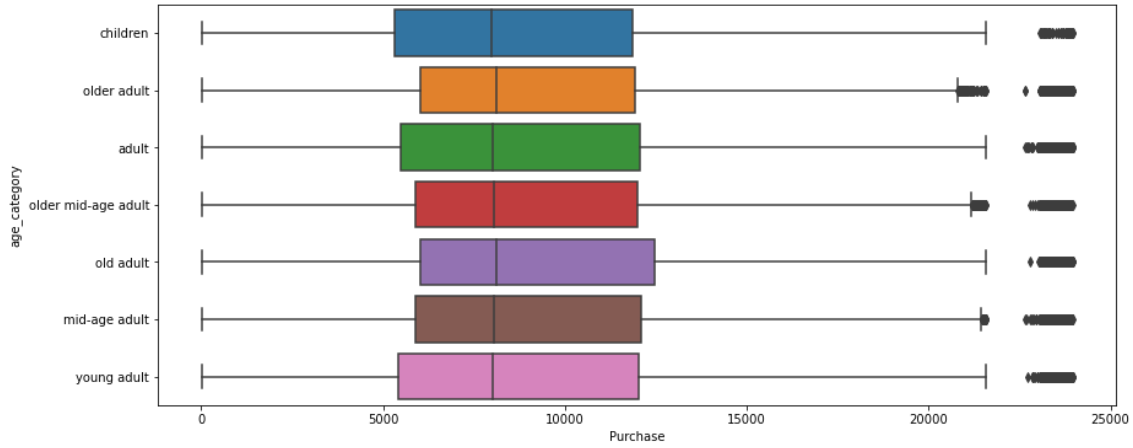
Insights:

1. Average amount spent per product is similar for every city
2. Irrespective of the cities, spending among different age groups behave are similar

Impact of Age

In [263]:

```
plt.figure(figsize=(14,6))
sns.boxplot(data=df,x='Purchase',y='age_category')
plt.show()
```



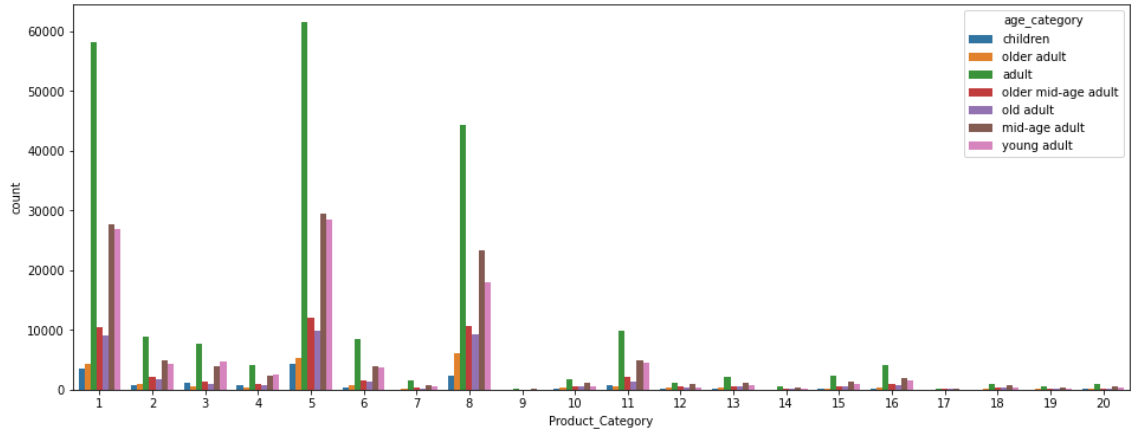
Insights:

1. Average amount spent per product is similar for all age groups

Product category vs Age

In [307]:

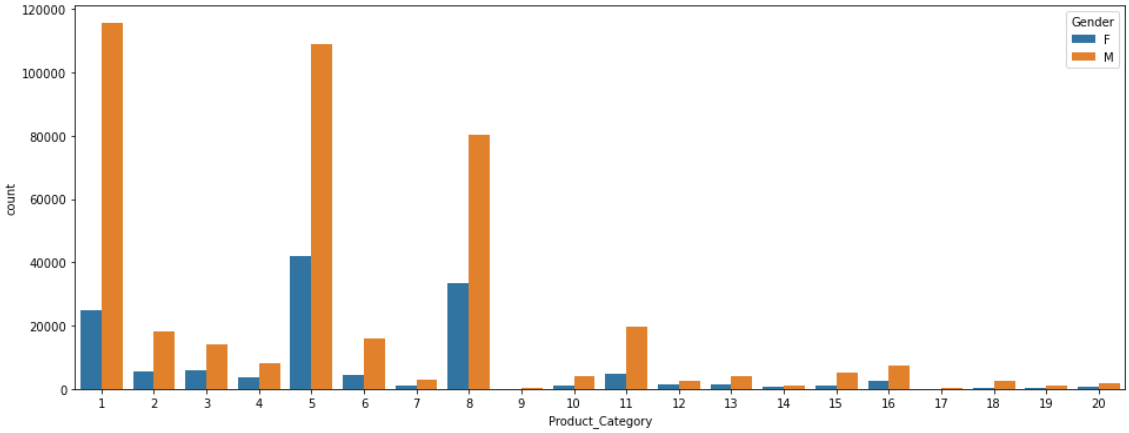
```
plt.figure(figsize=(16,6))
sns.countplot(data=df,x='Product_Category',hue='age_category')
plt.show()
```



Product category vs Gender

In [28]:

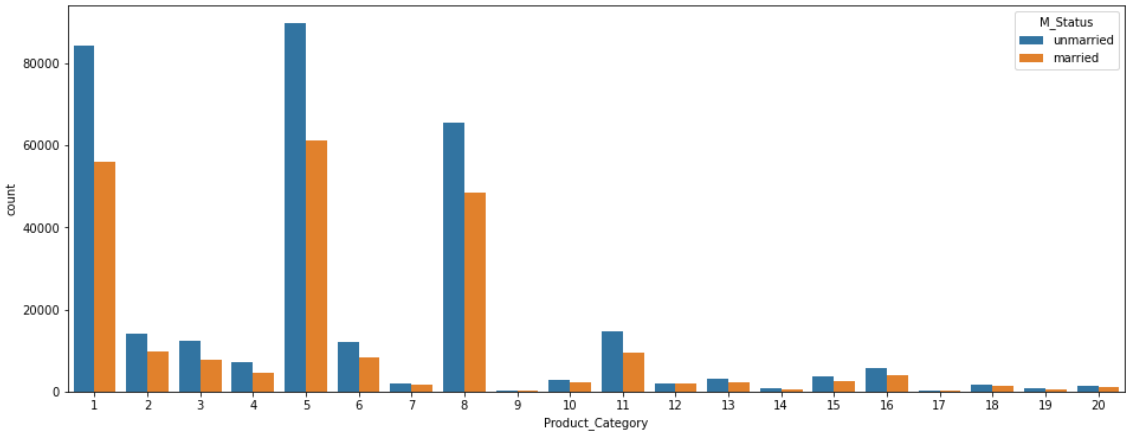
```
plt.figure(figsize=(16,6))
sns.countplot(data=df,x='Product_Category',hue='Gender')
plt.show()
```



Product category vs Marital status

In [30]:

```
plt.figure(figsize=(16,6))
sns.countplot(data=df,x='Product_Category',hue='M_Status')
plt.show()
```



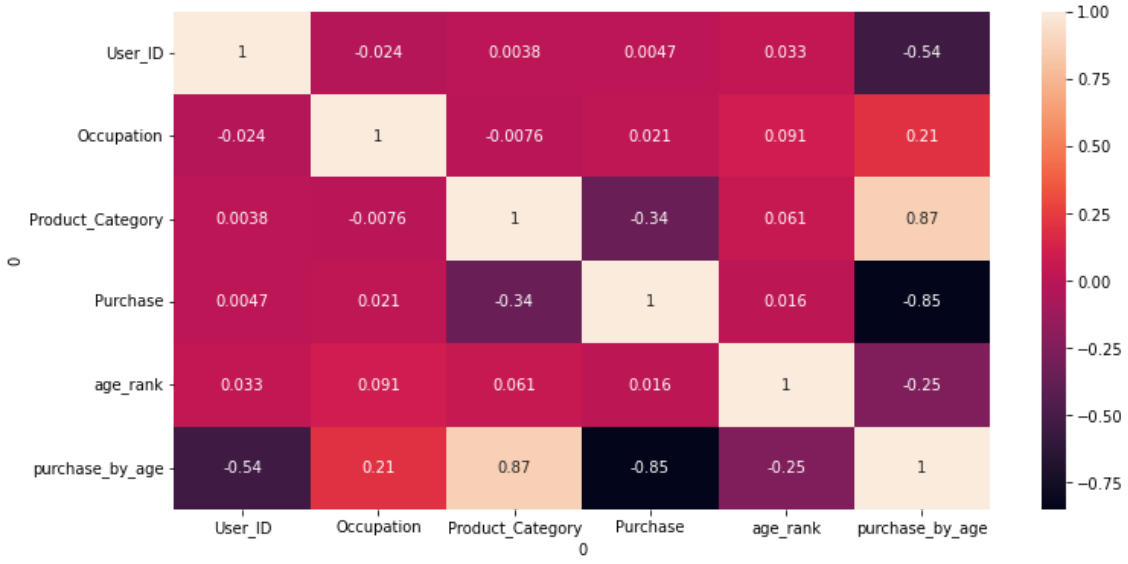
Correlation

In [265]:

```
plt.figure(figsize=(12,6))
sns.heatmap(df.corr(),annot=True)
```

Out[265]:

```
<AxesSubplot:xlabel='0', ylabel='0'>
```



Missing values

In [7]:

```
df.isna().sum()
```

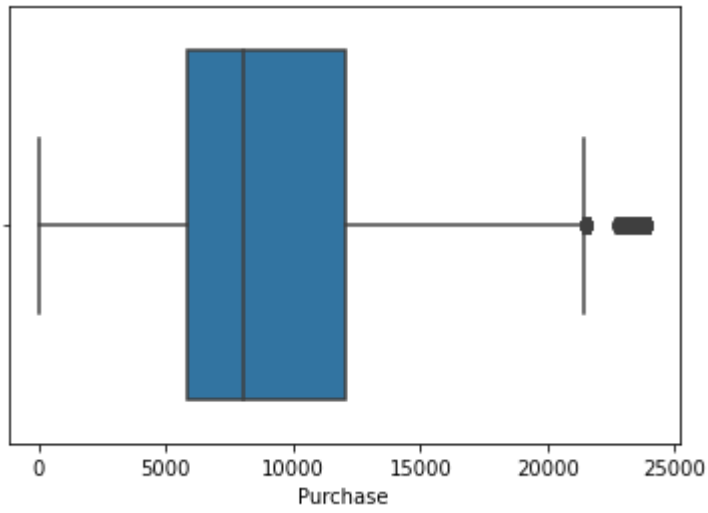
Out[7]:

```
0
User_ID                     0
Product_ID                  0
Gender                      0
Age                         0
Occupation                  0
City_Category               0
Stay_In_Current_City_Years  0
Marital_Status              0
Product_Category            0
Purchase                    0
dtype: int64
```

Outliers

In [13]:

```
sns.boxplot(data=df,x='Purchase')
plt.show()
```



In [8]:

```
df_outlr = df['Purchase'].sort_values()
q1 = np.percentile(df_outlr,25)
q2 = np.percentile(df_outlr,50)
q3 = np.percentile(df_outlr,75)
q1,q2,q3
```

Out[8]:

```
(5823.0, 8047.0, 12054.0)
```

In [9]:

```
IQR = q3 - q1
IQR
```

Out[9]:

```
6231.0
```

In [10]:

```
lower = q1 - (1.5*IQR)
upper = q3 + (1.5*IQR)
lower,upper
```

Out[10]:

```
(-3523.5, 21400.5)
```

In [11]:

```
round((df.loc[df['Purchase']>upper]['User_ID'].count()/df.shape[0])*100,2)
```

Out[11]:

```
0.49
```

In [12]:

```
df_new_outlr = df.loc[np.any(df['Purchase'] < lower) or (df['Purchase'] > upper )]
outlr = df_new_outlr.index.to_list()
len(outlr)
```

Out[12]:

```
2677
```

In [13]:

```
df1 = df
df_without_outlr = df1.drop(index=outlr)
df_without_outlr.head()
```

Out[13]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Year
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	4

Insights:

1. No missing values found
2. Removed the outliers using IQR

Spending habits between men and women

In [115]:

```
dfx = df_without_outlr
dfx.head()
```

Out[115]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Year
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	4

In [117]:

```
male_mean_purchase = int(dfx.groupby('Gender')['Purchase'].mean()[1])
female_mean_purchase = int(dfx.groupby('Gender')['Purchase'].mean()[0])
print(f"Average spending by men: {male_mean_purchase}")
print(f"Average spending by women: {female_mean_purchase}")
```

```
Average spending by men: 9367
Average spending by women: 8671
```

In [118]:

```
male = dfx.groupby('Gender')['Gender'].count()[1]
female = dfx.groupby('Gender')['Gender'].count()[0]
ratio = round(male/female,2)
print(f"Male customers: {male}")
print(f"Female customers: {female}")
print(f"Male to Female customers: {ratio}")
```

```
Male customers: 412171
Female customers: 135220
Male to Female customers: 3.05
```

In [119]:

```
df_female = dfx.loc[dfx['Gender']=='F']
df_male = dfx.loc[dfx['Gender']=='M']
```

Female vs Male: CLT with sample size = 30

In [121]:

```
bootstrap = []
n = 30
for i in range(1000):
    sample = np.random.choice(df_female['Purchase'],size=n,replace=True)
    bootstrap.append(np.mean(sample))

bs_m = []
n = 30
for i in range(1000):
    sample = np.random.choice(df_male['Purchase'],size=n,replace=True)
    bs_m.append(np.mean(sample))
```

In [122]:

```
CI_90 = (int(np.percentile(bootstrap,5)),int(np.percentile(bootstrap,95)))
CI_95 = (int(np.percentile(bootstrap,2.5)),int(np.percentile(bootstrap,97.5)))
CI_99 = (int(np.percentile(bootstrap,0.5)),int(np.percentile(bootstrap,99.5)))
print("Female CI with n=30:\n")
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")
print("Male CI with n=30:\n")
CI_90 = (int(np.percentile(bs_m,5)),int(np.percentile(bs_m,95)))
CI_95 = (int(np.percentile(bs_m,2.5)),int(np.percentile(bs_m,97.5)))
CI_99 = (int(np.percentile(bs_m,0.5)),int(np.percentile(bs_m,99.5)))
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}")
```

```
Female CI with n=30:

90% CI ---> (7222, 10049)
95% CI ---> (6924, 10301)
99% CI ---> (6538, 10718)

Male CI with n=30:

90% CI ---> (7920, 10943)
95% CI ---> (7675, 11307)
99% CI ---> (7287, 11638)
```

Female vs Male: CLT with sample size = 100

In [123]:

```
bsf = []
n = 100
for i in range(1000):
    sample = np.random.choice(df_female['Purchase'],size=n,replace=False)
    bsf.append(np.mean(sample))

bsm = []
n = 100
for i in range(1000):
    sample = np.random.choice(df_male['Purchase'],size=n,replace=False)
    bsm.append(np.mean(sample))
```

In [124]:

```
CI_90 = (int(np.percentile(bsf,5)),int(np.percentile(bsf,95)))
CI_95 = (int(np.percentile(bsf,2.5)),int(np.percentile(bsf,97.5)))
CI_99 = (int(np.percentile(bsf,0.5)),int(np.percentile(bsf,99.5)))
print("Female CI with n=100:\n")
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")
print("Male CI with n=100:\n")
CI_90 = (int(np.percentile(bsm,5)),int(np.percentile(bsm,95)))
CI_95 = (int(np.percentile(bsm,2.5)),int(np.percentile(bsm,97.5)))
CI_99 = (int(np.percentile(bsm,0.5)),int(np.percentile(bsm,99.5)))
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}")
```

```
Female CI with n=100:

90% CI ---> (7949, 9462)
95% CI ---> (7860, 9574)
99% CI ---> (7583, 9920)

Male CI with n=100:

90% CI ---> (8579, 10179)
95% CI ---> (8426, 10331)
99% CI ---> (8066, 10740)
```

Female vs Male: CLT with sample size = 1000

In [125]:

```
bsf = []
n = 1000
for i in range(1000):
    sample = np.random.choice(df_female['Purchase'],size=n,replace=False)
    bsf.append(np.mean(sample))

bsm = []
n = 1000
for i in range(1000):
    sample = np.random.choice(df_male['Purchase'],size=n,replace=False)
    bsm.append(np.mean(sample))
```

In [126]:

```
CI_90 = (int(np.percentile(bsf,5)),int(np.percentile(bsf,95)))
CI_95 = (int(np.percentile(bsf,2.5)),int(np.percentile(bsf,97.5)))
CI_99 = (int(np.percentile(bsf,0.5)),int(np.percentile(bsf,99.5)))
print("Female CI with n=1000:\n")
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")
print("Male CI with n=1000:\n")
CI_90 = (int(np.percentile(bsm,5)),int(np.percentile(bsm,95)))
CI_95 = (int(np.percentile(bsm,2.5)),int(np.percentile(bsm,97.5)))
CI_99 = (int(np.percentile(bsm,0.5)),int(np.percentile(bsm,99.5)))
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}")
```

```
Female CI with n=1000:

90% CI ---> (8425, 8913)
95% CI ---> (8390, 8957)
99% CI ---> (8260, 9021)

Male CI with n=1000:

90% CI ---> (9094, 9633)
95% CI ---> (9045, 9677)
99% CI ---> (8973, 9777)
```

Insights:

1. Male to Female customers: 3.05
2. Sample: Average spending by men: 9367, Average spending by women: 8671
3. Population: With a minimum sample size of 1000, Average male spending --> (8973, 9777), Average female spending --> (8260, 9021) at 99% CI

Spending habits between Married vs Unmarried

In [127]:

```
dfy = df_without_outlr
dfy.head()
```

Out[127]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Year
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	4

In [128]:

```
married_mean_purchase = int(dfy.groupby('M_Status')['Purchase'].mean()[1])
unmarried_mean_purchase = int(dfy.groupby('M_Status')['Purchase'].mean()[0])
print(f"Average spending by a married person: {married_mean_purchase}")
print(f"Average spending by an unmarried person: {unmarried_mean_purchase}")
```

```
Average spending by a married person: 9201
Average spending by an unmarried person: 9187
```

In [129]:

```
married = dfy.groupby('M_Status')['M_Status'].count()[1]
unmarried = dfy.groupby('M_Status')['M_Status'].count()[0]
ratio = round(married/unmarried,2)
print(f"Married customers: {married}")
print(f"Unmarried customers: {unmarried}")
print(f"Married to unmarried customers: {ratio}")
```

```
Married customers: 323242
Unmarried customers: 224149
Married to unmarried customers: 1.44
```

In [130]:

```
df_unmarried = dfy.loc[dfy['M_Status']=='unmarried']
df_married = dfy.loc[dfy['M_Status']=='married']
```

Married vs Unmarried: CLT with sample size = 30

In [132]:

```
bs_married = []
n = 30
for i in range(1000):
    sample = np.random.choice(df_married['Purchase'],size=n,replace=True)
    bs_married.append(np.mean(sample))

bs_unmarried = []
n = 30
for i in range(1000):
    sample = np.random.choice(df_unmarried['Purchase'],size=n,replace=True)
    bs_unmarried.append(np.mean(sample))
```

In [133]:

```
CI_90 = (int(np.percentile(bs_married,5)),int(np.percentile(bs_married,95)))
CI_95 = (int(np.percentile(bs_married,2.5)),int(np.percentile(bs_married,97.5)))
CI_99 = (int(np.percentile(bs_married,0.5)),int(np.percentile(bs_married,99.5)))
print("Married CI with n=30:\n")
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")
print("Unmarried CI with n=30:\n")
CI_90 = (int(np.percentile(bs_unmarried,5)),int(np.percentile(bs_unmarried,95)))
CI_95 = (int(np.percentile(bs_unmarried,2.5)),int(np.percentile(bs_unmarried,97.5)))
CI_99 = (int(np.percentile(bs_unmarried,0.5)),int(np.percentile(bs_unmarried,99.5)))
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}")
```

```
Married CI with n=30:

90% CI ---> (7796, 10628)
95% CI ---> (7510, 10923)
99% CI ---> (7129, 11596)

Unmarried CI with n=30:

90% CI ---> (7778, 10740)
95% CI ---> (7457, 11016)
99% CI ---> (6803, 11783)
```

Married vs Unmarried: CLT with sample size = 100

In [134]:

```
bs_married = []
n = 100
for i in range(1000):
    sample = np.random.choice(df_married['Purchase'],size=n,replace=True)
    bs_married.append(np.mean(sample))

bs_unmarried = []
n = 100
for i in range(1000):
    sample = np.random.choice(df_unmarried['Purchase'],size=n,replace=True)
    bs_unmarried.append(np.mean(sample))
```

In [135]:

```
CI_90 = (int(np.percentile(bs_married,5)),int(np.percentile(bs_married,95)))
CI_95 = (int(np.percentile(bs_married,2.5)),int(np.percentile(bs_married,97.5)))
CI_99 = (int(np.percentile(bs_married,0.5)),int(np.percentile(bs_married,99.5)))
print("Married CI with n=100:\n")
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")
print("Unmarried CI with n=100:\n")
CI_90 = (int(np.percentile(bs_unmarried,5)),int(np.percentile(bs_unmarried,95)))
CI_95 = (int(np.percentile(bs_unmarried,2.5)),int(np.percentile(bs_unmarried,97.5)))
CI_99 = (int(np.percentile(bs_unmarried,0.5)),int(np.percentile(bs_unmarried,99.5)))
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}")
```

```
Married CI with n=100:

90% CI ---> (8339, 10021)
95% CI ---> (8159, 10191)
99% CI ---> (7954, 10533)

Unmarried CI with n=100:

90% CI ---> (8380, 10035)
95% CI ---> (8272, 10180)
99% CI ---> (7915, 10552)
```

Married vs Unmarried: CLT with sample size = 1000

In [136]:

```
bs_married = []
n = 1000
for i in range(1000):
    sample = np.random.choice(df_married['Purchase'],size=n,replace=True)
    bs_married.append(np.mean(sample))

bs_unmarried = []
n = 1000
for i in range(1000):
    sample = np.random.choice(df_unmarried['Purchase'],size=n,replace=True)
    bs_unmarried.append(np.mean(sample))
```

In [137]:

```
CI_90 = (int(np.percentile(bs_married,5)),int(np.percentile(bs_married,95)))
CI_95 = (int(np.percentile(bs_married,2.5)),int(np.percentile(bs_married,97.5)))
CI_99 = (int(np.percentile(bs_married,0.5)),int(np.percentile(bs_married,99.5)))
print("Married CI with n=1000:\n")
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")
print("Unmarried CI with n=1000:\n")
CI_90 = (int(np.percentile(bs_unmarried,5)),int(np.percentile(bs_unmarried,95)))
CI_95 = (int(np.percentile(bs_unmarried,2.5)),int(np.percentile(bs_unmarried,97.5)))
CI_99 = (int(np.percentile(bs_unmarried,0.5)),int(np.percentile(bs_unmarried,99.5)))
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}")
```

```
Married CI with n=1000:

90% CI ---> (8930, 9449)
95% CI ---> (8877, 9488)
99% CI ---> (8796, 9573)

Unmarried CI with n=1000:

90% CI ---> (8940, 9457)
95% CI ---> (8905, 9491)
99% CI ---> (8818, 9575)
```

Insights:

1. Married to unmarried customers: 1.44
2. Sample: Average spending by a married person: 9201, Average spending by an unmarried person: 9187
3. Population: With a minimum sample size of 1000, Average spending by a married person --> (8796, 9573), Average spending by an unmarried person --> (8818, 9575) at 99% CI

Spending habits by age

In [138]:

```
dfz = df_without_outlr
dfz.head()
```

Out[138]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Year
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	4

In [139]:

```
df_children = dfz.loc[dfz['age_category']=='children']
df_young_adult = dfz.loc[dfz['age_category']=='young adult']
df_adult = dfz.loc[dfz['age_category']=='adult']
df_mid_age_adult = dfz.loc[dfz['age_category']=='mid-age adult']
df_older_mid_age_adult = dfz.loc[dfz['age_category']=='older mid-age adult']
df_old_adult = dfz.loc[dfz['age_category']=='old adult']
df_older_adult = dfz.loc[dfz['age_category']=='older adult']
```

In [141]:

```
print(f"Average spending by children: {int(df_children['Purchase'].mean())}")
print(f"Average spending by young adults: {int(df_young_adult['Purchase'].mean())}")
print(f"Average spending by adults: {int(df_adult['Purchase'].mean())}")
print(f"Average spending by mid age adults: {int(df_mid_age_adult['Purchase'].mean())}")
print(f"Average spending by older mid age adults: {int(df_older_mid_age_adult['Purchase'
print(f"Average spending by old adults: {int(df_old_adult['Purchase'].mean())}")
print(f"Average spending by older adults: {int(df_older_adult['Purchase'].mean())}")
```

```
Average spending by children: 8867
Average spending by young adults: 9124
Average spending by adults: 9193
Average spending by mid age adults: 9254
Average spending by older mid age adults: 9128
Average spending by old adults: 9423
Average spending by older adults: 9216
```

Age: CLT with sample size = 100

In [142]:

```
bs_children = []
n = 100
for i in range(1000):
    sample = np.random.choice(df_children['Purchase'],size=n,replace=True)
    bs_children.append(np.mean(sample))

bs_young_adult = []
n = 100
for i in range(1000):
    sample = np.random.choice(df_young_adult['Purchase'],size=n,replace=True)
    bs_young_adult.append(np.mean(sample))

bs_adult = []
n = 100
for i in range(1000):
    sample = np.random.choice(df_adult['Purchase'],size=n,replace=True)
    bs_adult.append(np.mean(sample))

bs_mid_age_adult = []
n = 100
for i in range(1000):
    sample = np.random.choice(df_mid_age_adult['Purchase'],size=n,replace=True)
    bs_mid_age_adult.append(np.mean(sample))

bs_older_mid_age_adult = []
n = 100
for i in range(1000):
    sample = np.random.choice(df_older_mid_age_adult['Purchase'],size=n,replace=True)
    bs_older_mid_age_adult.append(np.mean(sample))

bs_old_adult = []
n = 100
for i in range(1000):
    sample = np.random.choice(df_old_adult['Purchase'],size=n,replace=True)
    bs_old_adult.append(np.mean(sample))

bs_older_adult = []
n = 100
for i in range(1000):
    sample = np.random.choice(df_older_adult['Purchase'],size=n,replace=True)
    bs_older_adult.append(np.mean(sample))
```

In [143]:

```
CI_90 = (int(np.percentile(bs_children,5)),int(np.percentile(bs_children,95)))
CI_95 = (int(np.percentile(bs_children,2.5)),int(np.percentile(bs_children,97.5)))
CI_99 = (int(np.percentile(bs_children,0.5)),int(np.percentile(bs_children,99.5)))
print("Children CI with n=100:\n")
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")

print("Young adult CI with n=100:\n")
CI_90 = (int(np.percentile(bs_young_adult,5)),int(np.percentile(bs_young_adult,95)))
CI_95 = (int(np.percentile(bs_young_adult,2.5)),int(np.percentile(bs_young_adult,97.5)))
CI_99 = (int(np.percentile(bs_young_adult,0.5)),int(np.percentile(bs_young_adult,99.5)))
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")

print("Adult CI with n=100:\n")
CI_90 = (int(np.percentile(bs_adult,5)),int(np.percentile(bs_adult,95)))
CI_95 = (int(np.percentile(bs_adult,2.5)),int(np.percentile(bs_adult,97.5)))
CI_99 = (int(np.percentile(bs_adult,0.5)),int(np.percentile(bs_adult,99.5)))
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")

print("Mid-age adult CI with n=100:\n")
CI_90 = (int(np.percentile(bs_mid_age_adult,5)),int(np.percentile(bs_mid_age_adult,95)))
CI_95 = (int(np.percentile(bs_mid_age_adult,2.5)),int(np.percentile(bs_mid_age_adult,97.
CI_99 = (int(np.percentile(bs_mid_age_adult,0.5)),int(np.percentile(bs_mid_age_adult,99.
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")

print("Older Mid-age adult CI with n=100:\n")
CI_90 = (int(np.percentile(bs_older_mid_age_adult,5)),int(np.percentile(bs_older_mid_age
CI_95 = (int(np.percentile(bs_older_mid_age_adult,2.5)),int(np.percentile(bs_older_mid_a
CI_99 = (int(np.percentile(bs_older_mid_age_adult,0.5)),int(np.percentile(bs_older_mid_a
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")

print("Old adult CI with n=100:\n")
CI_90 = (int(np.percentile(bs_old_adult,5)),int(np.percentile(bs_old_adult,95)))
CI_95 = (int(np.percentile(bs_old_adult,2.5)),int(np.percentile(bs_old_adult,97.5)))
CI_99 = (int(np.percentile(bs_old_adult,0.5)),int(np.percentile(bs_old_adult,99.5)))
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")

print("Older adult CI with n=100:\n")
CI_90 = (int(np.percentile(bs_older_adult,5)),int(np.percentile(bs_older_adult,95)))
CI_95 = (int(np.percentile(bs_older_adult,2.5)),int(np.percentile(bs_older_adult,97.5)))
CI_99 = (int(np.percentile(bs_older_adult,0.5)),int(np.percentile(bs_older_adult,99.5)))
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")
```

```
Children CI with n=100:

90% CI ---> (8041, 9706)
95% CI ---> (7836, 9895)
99% CI ---> (7541, 10146)

Young adult CI with n=100:

90% CI ---> (8311, 9955)
95% CI ---> (8143, 10098)
99% CI ---> (7926, 10406)

Adult CI with n=100:

90% CI ---> (8412, 10062)
95% CI ---> (8305, 10229)
99% CI ---> (8106, 10594)

Mid-age adult CI with n=100:

90% CI ---> (8459, 10094)
95% CI ---> (8333, 10278)
99% CI ---> (8107, 10645)

Older Mid-age adult CI with n=100:

90% CI ---> (8341, 9938)
95% CI ---> (8196, 10100)
99% CI ---> (8009, 10396)

Old adult CI with n=100:

90% CI ---> (8665, 10243)
95% CI ---> (8509, 10388)
99% CI ---> (8248, 10706)

Older adult CI with n=100:

90% CI ---> (8455, 9994)
95% CI ---> (8336, 10137)
99% CI ---> (8122, 10553)
```

Age: CLT with sample size = 1000

In [144]:

```
bs_children = []
n = 1000
for i in range(1000):
    sample = np.random.choice(df_children['Purchase'],size=n,replace=True)
    bs_children.append(np.mean(sample))

bs_young_adult = []
n = 1000
for i in range(1000):
    sample = np.random.choice(df_young_adult['Purchase'],size=n,replace=True)
    bs_young_adult.append(np.mean(sample))

bs_adult = []
n = 1000
for i in range(1000):
    sample = np.random.choice(df_adult['Purchase'],size=n,replace=True)
    bs_adult.append(np.mean(sample))

bs_mid_age_adult = []
n = 1000
for i in range(1000):
    sample = np.random.choice(df_mid_age_adult['Purchase'],size=n,replace=True)
    bs_mid_age_adult.append(np.mean(sample))

bs_older_mid_age_adult = []
n = 1000
for i in range(1000):
    sample = np.random.choice(df_older_mid_age_adult['Purchase'],size=n,replace=True)
    bs_older_mid_age_adult.append(np.mean(sample))

bs_old_adult = []
n = 1000
for i in range(1000):
    sample = np.random.choice(df_old_adult['Purchase'],size=n,replace=True)
    bs_old_adult.append(np.mean(sample))

bs_older_adult = []
n = 1000
for i in range(1000):
    sample = np.random.choice(df_older_adult['Purchase'],size=n,replace=True)
    bs_older_adult.append(np.mean(sample))
```

In [145]:

```
CI_90 = (int(np.percentile(bs_children,5)),int(np.percentile(bs_children,95)))
CI_95 = (int(np.percentile(bs_children,2.5)),int(np.percentile(bs_children,97.5)))
CI_99 = (int(np.percentile(bs_children,0.5)),int(np.percentile(bs_children,99.5)))
print("Children CI with n=1000:\n")
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")

print("Young adult CI with n=1000:\n")
CI_90 = (int(np.percentile(bs_young_adult,5)),int(np.percentile(bs_young_adult,95)))
CI_95 = (int(np.percentile(bs_young_adult,2.5)),int(np.percentile(bs_young_adult,97.5)))
CI_99 = (int(np.percentile(bs_young_adult,0.5)),int(np.percentile(bs_young_adult,99.5)))
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")

print("Adult CI with n=1000:\n")
CI_90 = (int(np.percentile(bs_adult,5)),int(np.percentile(bs_adult,95)))
CI_95 = (int(np.percentile(bs_adult,2.5)),int(np.percentile(bs_adult,97.5)))
CI_99 = (int(np.percentile(bs_adult,0.5)),int(np.percentile(bs_adult,99.5)))
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")

print("Mid-age adult CI with n=1000:\n")
CI_90 = (int(np.percentile(bs_mid_age_adult,5)),int(np.percentile(bs_mid_age_adult,95)))
CI_95 = (int(np.percentile(bs_mid_age_adult,2.5)),int(np.percentile(bs_mid_age_adult,97.
CI_99 = (int(np.percentile(bs_mid_age_adult,0.5)),int(np.percentile(bs_mid_age_adult,99.
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")

print("Older Mid-age adult CI with n=1000:\n")
CI_90 = (int(np.percentile(bs_older_mid_age_adult,5)),int(np.percentile(bs_older_mid_age
CI_95 = (int(np.percentile(bs_older_mid_age_adult,2.5)),int(np.percentile(bs_older_mid_a
CI_99 = (int(np.percentile(bs_older_mid_age_adult,0.5)),int(np.percentile(bs_older_mid_a
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")

print("Old adult CI with n=1000:\n")
CI_90 = (int(np.percentile(bs_old_adult,5)),int(np.percentile(bs_old_adult,95)))
CI_95 = (int(np.percentile(bs_old_adult,2.5)),int(np.percentile(bs_old_adult,97.5)))
CI_99 = (int(np.percentile(bs_old_adult,0.5)),int(np.percentile(bs_old_adult,99.5)))
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")

print("Older adult CI with n=1000:\n")
CI_90 = (int(np.percentile(bs_older_adult,5)),int(np.percentile(bs_older_adult,95)))
CI_95 = (int(np.percentile(bs_older_adult,2.5)),int(np.percentile(bs_older_adult,97.5)))
CI_99 = (int(np.percentile(bs_older_adult,0.5)),int(np.percentile(bs_older_adult,99.5)))
print(f"90% CI ---> {CI_90}")
print(f"95% CI ---> {CI_95}")
print(f"99% CI ---> {CI_99}\n")
```

```
Children CI with n=1000:

90% CI ---> (8603, 9136)
95% CI ---> (8544, 9190)
99% CI ---> (8461, 9298)

Young adult CI with n=1000:

90% CI ---> (8887, 9376)
95% CI ---> (8824, 9440)
99% CI ---> (8734, 9540)

Adult CI with n=1000:

90% CI ---> (8934, 9459)
95% CI ---> (8894, 9510)
99% CI ---> (8809, 9583)

Mid-age adult CI with n=1000:

90% CI ---> (8988, 9522)
95% CI ---> (8926, 9572)
99% CI ---> (8833, 9660)

Older Mid-age adult CI with n=1000:

90% CI ---> (8886, 9367)
95% CI ---> (8838, 9413)
99% CI ---> (8741, 9488)

Old adult CI with n=1000:

90% CI ---> (9181, 9693)
95% CI ---> (9132, 9738)
99% CI ---> (8982, 9853)

Older adult CI with n=1000:

90% CI ---> (8963, 9459)
95% CI ---> (8919, 9494)
99% CI ---> (8820, 9618)
```

Insights:

1. Sample: Average spending by children: 8867 Average spending by young adults: 9124 Average spending by adults: 9193 Average spending by mid age adults: 9254 Average spending by older mid age adults: 9128 Average spending by old adults: 9423 Average spending by older adults: 9216
2. Population: with sample size of 1000, at 99% CI Average spending by children: (8461, 9298) Average spending by young adults: (8734, 9540) Average spending by adults: (8809, 9583) Average spending by mid age adults: (8833, 9660) Average spending by older mid age adults: (8741, 9488) Average spending by old adults: (8982, 9853) Average spending by older adults: (8820, 9618)

Recommendations:

1. Ratio of male to female customers for all product categories --> 3.05. Males are the major contributer to sales of each product category
2. Ratio of married to unmarried customers for all product categories --> 1.44
3. Contribution of product category 1, 5 & 8 --> 73% of total sales. Replace Product categories of least sales: 9,14,17,18,19 & 20 with 1,5 & 8 to increase sales & use inventory efficiently
4. Contribution to total sales by each city type: City A --> 27%, City B --> 42%, City C --> 31%. Irrespective of the city, purchasing habits like amount and product category between different age groups remain similar
5. Age range of customers between 18 to 45 contributed --> 78% of total sales. They are the major contributers for every product category
6. Sample: Average spending by men: 9367, Average spending by women: 8671
7. Population:Average male spending --> (8973, 9777), Average female spending --> (8260, 9021) at 99% CI
8. Sample: Average spending by a married person: 9201, Average spending by an unmarried person: 9187
9. Population: Average spending by a married person --> (8796, 9573), Average spending by an unmarried person --> (8818, 9575) at 99% CI