# Business Problem

The Management team at Walmart Inc. wants to analyze the customer purchase behavior (specifically, purchase amount) against the customer's gender and the various other factors to help the business make better decisions. They want to understand if the spending habits differ between male and female customers: Do women spend more on Black Friday than men? (Assume 50 million customers are male and 50 million are female).

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

In [2]:
```
WM = pd.read_csv(r"H:\Scaler\Confidence Interval\walmart_data.csv")
```

In [3]:
```
#Checking the dataset
WM.head()
```

Out[3]:

| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1000001 | P00069042 | F | 0-17 | 10 | A | 2 | 0 | 3 | 8370 |
| **1** | 1000001 | P00248942 | F | 0-17 | 10 | A | 2 | 0 | 1 | 15200 |
| **2** | 1000001 | P00087842 | F | 0-17 | 10 | A | 2 | 0 | 12 | 1422 |
| **3** | 1000001 | P00085442 | F | 0-17 | 10 | A | 2 | 0 | 12 | 1057 |
| **4** | 1000002 | P00285442 | M | 55+ | 16 | C | 4+ | 0 | 8 | 7969 |

In [4]:
```
#Checking the info
WM.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  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64 
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

The data set contains 10 columns. The data type of columms is split between int64 and object. The columns User_ID,Occupation ,Marital_Status ,Product_Category & Purchase belong to int64 The columns Product_ID,Gender,Age,City_Category & Stay_In_Current_City_Years belong to the object category. There are no null values in all the columns

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

Out[5]:
```
(550068, 10)
```

The dataset has 550068 rows and 10 columns

In [6]:
```
WM.describe()
```

Out[6]:

| | 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 [7]:
```
WM.tail()
```

Out[7]:

| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|
| **550063** | 1006033 | P00372445 | M | 51-55 | 13 | B | 1 | 1 | 20 | 368 |
| **550064** | 1006035 | P00375436 | F | 26-35 | 1 | C | 3 | 0 | 20 | 371 |
| **550065** | 1006036 | P00375436 | F | 26-35 | 15 | B | 4+ | 1 | 20 | 137 |
| **550066** | 1006038 | P00375436 | F | 55+ | 1 | C | 2 | 0 | 20 | 365 |
| **550067** | 1006039 | P00371644 | F | 46-50 | 0 | B | 4+ | 1 | 20 | 490 |

In [8]:
```
WM.mode()
```

Out[8]:

| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1001680 | P00265242 | M | 26-35 | 4 | B | 1 | 0 | 5 | 7011 |

In [9]:
```
WM.median()
```

```
C:\Users\india\AppData\Local\Temp\ipykernel_17652\2619744334.py:1: FutureWarning: Dropping of nuisance columns in DataFrame red
uctions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns bef
ore calling the reduction.
  WM.median()
```

Out[9]:
```
User_ID             1003077.0
Occupation                7.0
Marital_Status            0.0
Product_Category          5.0
Purchase               8047.0
dtype: float64
```

In [10]:
```
#Creating a dataset for male customers
WMM = WM[WM['Gender']=='M']
```

In [11]:
```
WMM.head()
```

Out[11]:

| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|
| **4** | 1000002 | P00285442 | M | 55+ | 16 | C | 4+ | 0 | 8 | 7969 |
| **5** | 1000003 | P00193542 | M | 26-35 | 15 | A | 3 | 0 | 1 | 15227 |
| **6** | 1000004 | P00184942 | M | 46-50 | 7 | B | 2 | 1 | 1 | 19215 |
| **7** | 1000004 | P00346142 | M | 46-50 | 7 | B | 2 | 1 | 1 | 15854 |
| **8** | 1000004 | P0097242 | M | 46-50 | 7 | B | 2 | 1 | 1 | 15686 |

Creating a dataset for female customers

In [12]:
```
WMF = WM[WM['Gender']=='F']
```

In [13]:
```
WMF.head()
```

Out[13]:

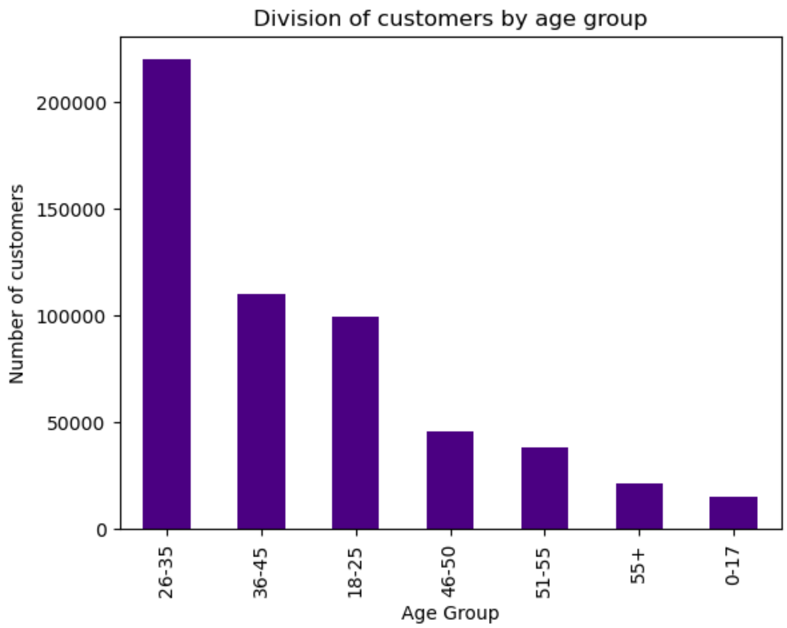
| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1000001 | P00069042 | F | 0-17 | 10 | A | 2 | 0 | 3 | 8370 |
| **1** | 1000001 | P00248942 | F | 0-17 | 10 | A | 2 | 0 | 1 | 15200 |
| **2** | 1000001 | P00087842 | F | 0-17 | 10 | A | 2 | 0 | 12 | 1422 |
| **3** | 1000001 | P00085442 | F | 0-17 | 10 | A | 2 | 0 | 12 | 1057 |
| **14** | 1000006 | P00231342 | F | 51-55 | 9 | A | 1 | 0 | 5 | 5378 |

In [52]:
```
def find_outliers_IQR(df):
    q1 = df.quantile(0.25)
    q3 = df.quantile(0.75)
    IQR = q3-q1
    outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]
    return outliers
```

In [15]:
```
WM['Age'].value_counts()
```

Out[15]:
```
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
```

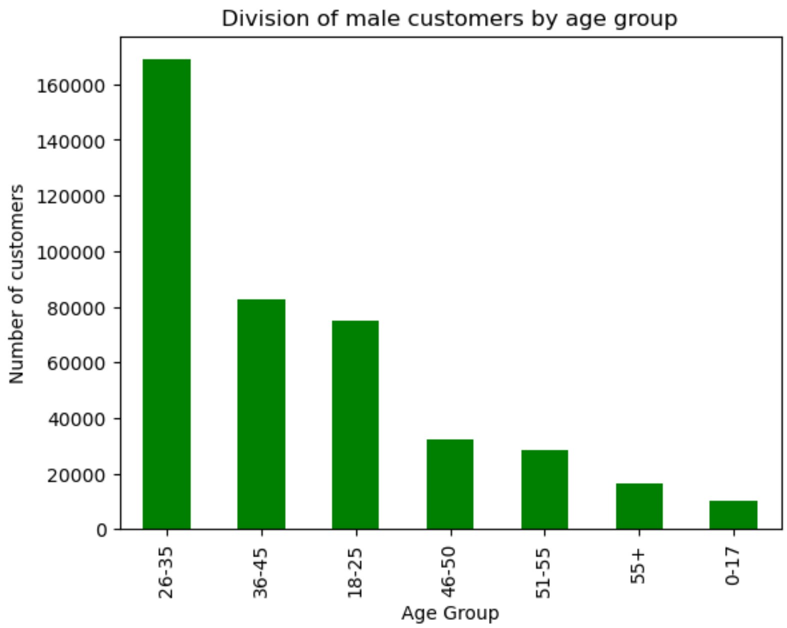
In [16]:
```
WM['Age'].value_counts().plot(kind='bar',color='indigo')
plt.xlabel('Age Group')
plt.ylabel('Number of customers')
plt.title('Division of customers by age group')
plt.show()
```




In [17]:
```
WMM['Age'].value_counts()
```

Out[17]:
```
26-35    168835
36-45     82843
18-25     75032
46-50     32502
51-55     28607
55+       16421
0-17      10019
Name: Age, dtype: int64
```

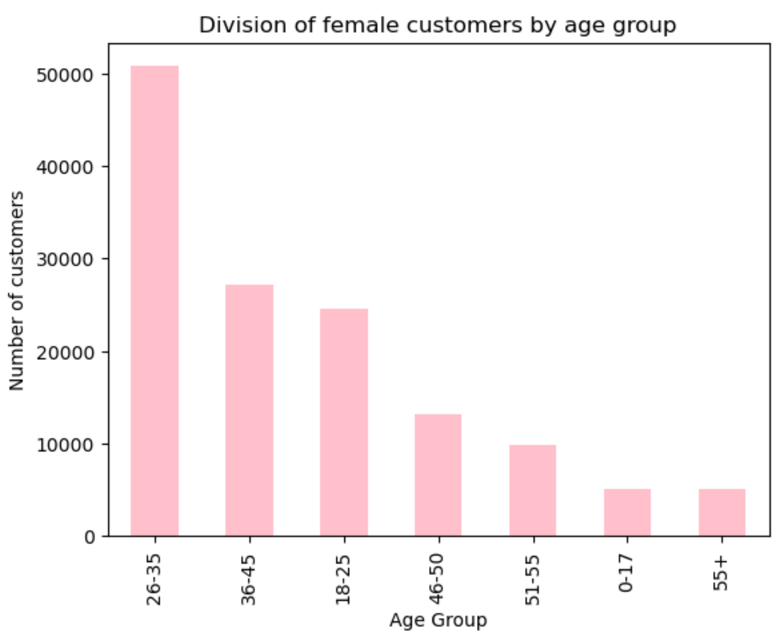
In [18]:
```
WMM['Age'].value_counts().plot(kind='bar',color='green')
plt.xlabel('Age Group')
plt.ylabel('Number of customers')
plt.title('Division of male customers by age group')
plt.show()
```




In [19]:
```
WMF['Age'].value_counts()
```

Out[19]:
```
26-35    50752
36-45    27170
18-25    24628
46-50    13199
51-55     9894
0-17      5083
55+       5083
Name: Age, dtype: int64
```

In [20]:
```
WMF['Age'].value_counts().plot(kind='bar',color='pink')
plt.xlabel('Age Group')
plt.ylabel('Number of customers')
plt.title('Division of female customers by age group')
plt.show()
```



In [21]:
```
import seaborn as sns
```

In [23]:
```
round(WM['Occupation'].mean(),2)
```

Out[23]:
```
8.08
```

In [27]:
```
WM['Occupation'].mode()
```

Out[27]:
```
0    4
Name: Occupation, dtype: int64
```

In [25]:
```
WM['Occupation'].median()
```

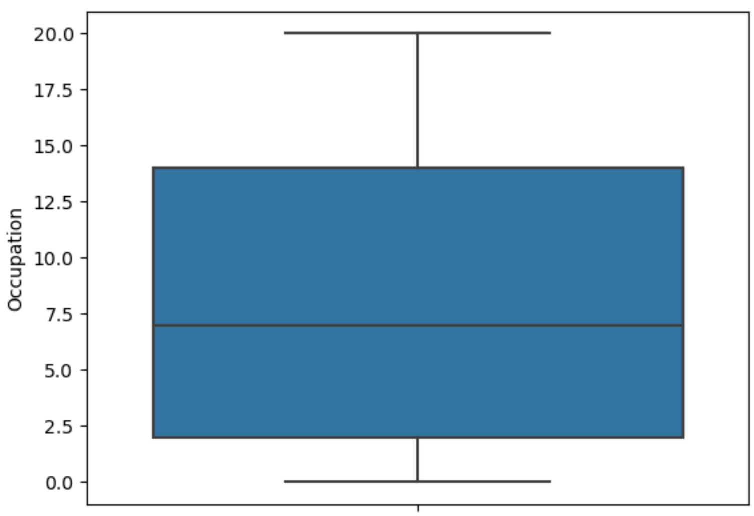
Out[25]:
```
7.0
```

In [29]:
```
WM.isnull().sum()
```

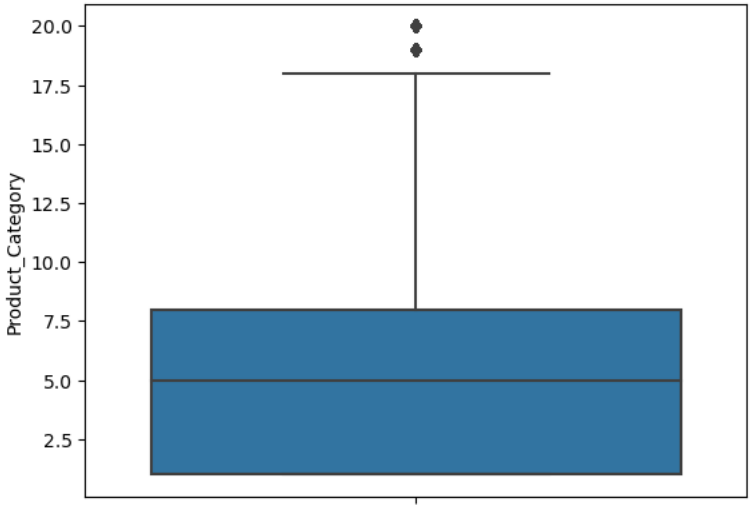
Out[29]:
```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category              0
Purchase                      0
dtype: int64
```

There are no null values in the dataset

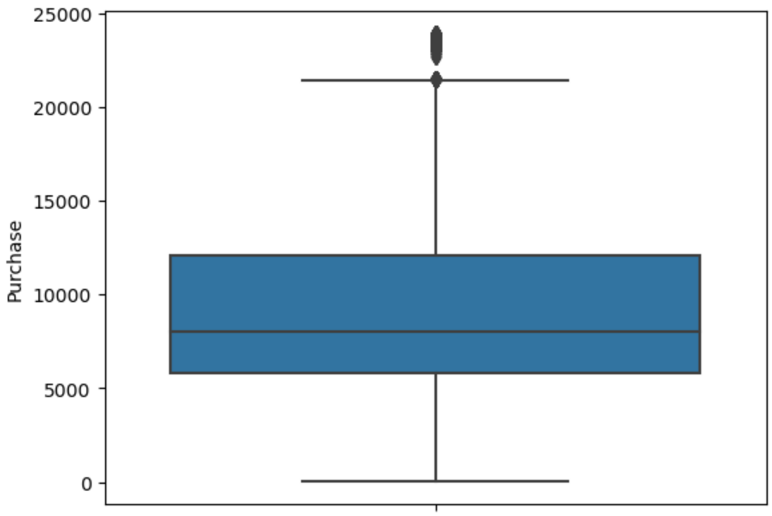
In [44]:
```
sns.boxplot(y=WM['Occupation'])
plt.show()
```



In [42]:
```
sns.boxplot(y=WM['Product_Category'])
plt.show()
```




In [43]:
```
sns.boxplot(y=WM['Purchase'])
plt.show()
```




In [54]:
```
PC_outliers = find_outliers_IQR(WM['Product_Category'])

print('number of outliers: '+ str(len(PC_outliers)))

print('max outlier value: '+ str(PC_outliers.max()))

print('min outlier values: '+ str(PC_outliers.min()))
```

```
number of outliers: 4153
max outlier value: 20
min outlier values: 19
```

In [55]:
```
Purchase_outliers = find_outliers_IQR(WM['Purchase'])

print('number of outliers: '+ str(len(Purchase_outliers)))

print('max outlier value: '+ str(Purchase_outliers.max()))

print('min outlier values: '+ str(Purchase_outliers.min()))
```

```
number of outliers: 2677
max outlier value: 23961
min outlier values: 21401
```

In [56]:
```
WMMD = WMM = WM[WM['Marital_Status']==1]
```

In [57]:
```
WMMD
```

Out[57]:

| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|
| **6** | 1000004 | P00184942 | M | 46-50 | 7 | B | 2 | 1 | 1 | 19215 |
| **7** | 1000004 | P00346142 | M | 46-50 | 7 | B | 2 | 1 | 1 | 15854 |
| **8** | 1000004 | P0097242 | M | 46-50 | 7 | B | 2 | 1 | 1 | 15686 |
| **9** | 1000005 | P00274942 | M | 26-35 | 20 | A | 1 | 1 | 8 | 7871 |
| **10** | 1000005 | P00251242 | M | 26-35 | 20 | A | 1 | 1 | 5 | 5254 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **550060** | 1006026 | P00371644 | M | 36-45 | 6 | C | 1 | 1 | 20 | 494 |
| **550061** | 1006029 | P00372445 | F | 26-35 | 1 | C | 1 | 1 | 20 | 599 |
| **550063** | 1006033 | P00372445 | M | 51-55 | 13 | B | 1 | 1 | 20 | 368 |
| **550065** | 1006036 | P00375436 | F | 26-35 | 15 | B | 4+ | 1 | 20 | 137 |
| **550067** | 1006039 | P00371644 | F | 46-50 | 0 | B | 4+ | 1 | 20 | 490 |

225337 rows × 10 columns

In [58]:
```
WMS = WMM = WM[WM['Marital_Status']==0]
```

In [59]:
```
WMS
```

Out[59]:

| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1000001 | P00069042 | F | 0-17 | 10 | A | 2 | 0 | 3 | 8370 |
| **1** | 1000001 | P00248942 | F | 0-17 | 10 | A | 2 | 0 | 1 | 15200 |
| **2** | 1000001 | P00087842 | F | 0-17 | 10 | A | 2 | 0 | 12 | 1422 |
| **3** | 1000001 | P00085442 | F | 0-17 | 10 | A | 2 | 0 | 12 | 1057 |
| **4** | 1000002 | P00285442 | M | 55+ | 16 | C | 4+ | 0 | 8 | 7969 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **550056** | 1006022 | P00375436 | M | 26-35 | 17 | C | 4+ | 0 | 20 | 254 |
| **550059** | 1006025 | P00370853 | F | 26-35 | 1 | B | 1 | 0 | 19 | 48 |
| **550062** | 1006032 | P00372445 | M | 46-50 | 7 | A | 3 | 0 | 20 | 473 |
| **550064** | 1006035 | P00375436 | F | 26-35 | 1 | C | 3 | 0 | 20 | 371 |
| **550066** | 1006038 | P00375436 | F | 55+ | 1 | C | 2 | 0 | 20 | 365 |

324731 rows × 10 columns

In [60]:
```
WM['Age'].value_counts()
```

Out[60]:
```
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 [61]:
```
WMF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 135809 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     135809 non-null  int64 
 1   Product_ID                  135809 non-null  object
 2   Gender                      135809 non-null  object
 3   Age                         135809 non-null  object
 4   Occupation                  135809 non-null  int64 
 5   City_Category               135809 non-null  object
 6   Stay_In_Current_City_Years  135809 non-null  object
 7   Marital_Status              135809 non-null  int64 
 8   Product_Category            135809 non-null  int64 
 9   Purchase                    135809 non-null  int64 
dtypes: int64(5), object(5)
memory usage: 11.4+ MB
```

In [62]:
```
WMF['Purchase'].mean()
```
Out[62]: 8734.565765155476

In [63]:
```
WMM['Purchase'].mean()
```
Out[63]: 9265.907618921507

# CALCULATING CONFIDENCE INTERVAL

In [64]:
```
#Upper limit of confidence interval
CIU = (pm+((1.96*std)/np.sqrt(n)))
#Lower limit of confidence interval
CIL = (pm-((1.96*std)/np.sqrt(n)))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [64], line 2
      1 #Upper limit of confidence interval
----> 2 CIU = (pm+((1.96*std)/np.sqrt(n)))
      3 #Lower limit of confidence interval
      4 CIL = (pm-((1.96*std)/np.sqrt(n)))

NameError: name 'pm' is not defined
```

In [65]:
```
WMM.head()
```
Out[65]:

| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1000001 | P00069042 | F | 0-17 | 10 | A | 2 | 0 | 3 | 8370 |
| **1** | 1000001 | P00248942 | F | 0-17 | 10 | A | 2 | 0 | 1 | 15200 |
| **2** | 1000001 | P00087842 | F | 0-17 | 10 | A | 2 | 0 | 12 | 1422 |
| **3** | 1000001 | P00085442 | F | 0-17 | 10 | A | 2 | 0 | 12 | 1057 |
| **4** | 1000002 | P00285442 | M | 55+ | 16 | C | 4+ | 0 | 8 | 7969 |

In [86]:
```
num_people = 10000
num_samples = 100

means = []
stds = []
for i in range(num_people):
    sample = WMM['Purchase'].sample(num_samples)
    means.append(round(sample.mean(),2))
    stds.append(round(sample.mean(),2))
```

In [87]:
```
means
```

```
 9754.24,
 9482.01,
 9317.79,
 9811.35,
 9356.86,
 9441.75,
 9087.72,
 10247.84,
 8994.89,
 9596.52,
 8583.8,
 9206.52,
 9172.4,
 8948.49,
 8813.83,
 9152.45,
 9894.45,
 9845.82,
 9114.87,
```

In [88]:
```
sum(means)/len(means)
```
Out[88]: 9259.461187999968

In [89]:
```
WMM.describe()
```

Out[89]:

| | User_ID | Occupation | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|
| **count** | 3.247310e+05 | 324731.000000 | 324731.0 | 324731.000000 | 324731.000000 |
| **mean** | 1.002999e+06 | 7.944782 | 0.0 | 5.339059 | 9265.907619 |
| **std** | 1.700466e+03 | 6.402753 | 0.0 | 3.912070 | 5027.347859 |
| **min** | 1.000001e+06 | 0.000000 | 0.0 | 1.000000 | 12.000000 |
| **25%** | 1.001524e+06 | 3.000000 | 0.0 | 1.000000 | 5605.000000 |
| **50%** | 1.003065e+06 | 7.000000 | 0.0 | 5.000000 | 8044.000000 |
| **75%** | 1.004386e+06 | 14.000000 | 0.0 | 8.000000 | 12061.000000 |
| **max** | 1.006040e+06 | 20.000000 | 0.0 | 20.000000 | 23961.000000 |

In [91]:
```
#Calculation upper confidence interval.
CIU = (9259.46+((1.96*5027.35)/np.sqrt(10000)))
CIU
```

Out[91]: 9357.99606

In [92]:
```
#Calculation upper confidence interval.
CIL = (9259.46-((1.96*5027.35)/np.sqrt(10000)))
```

Out[92]: 9160.923939999999

# CALCULATION CONFIDENCE INTERVAL FOR WOMEN

In [78]:
```
WMF.head()
```

Out[78]:

| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1000001 | P00069042 | F | 0-17 | 10 | A | 2 | 0 | 3 | 8370 |
| **1** | 1000001 | P00248942 | F | 0-17 | 10 | A | 2 | 0 | 1 | 15200 |
| **2** | 1000001 | P00087842 | F | 0-17 | 10 | A | 2 | 0 | 12 | 1422 |
| **3** | 1000001 | P00085442 | F | 0-17 | 10 | A | 2 | 0 | 12 | 1057 |
| **14** | 1000006 | P00231342 | F | 51-55 | 9 | A | 1 | 0 | 5 | 5378 |

In [80]:
```
WMF.describe()
```

Out[80]:

| | User_ID | Occupation | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|
| **count** | 1.358090e+05 | 135809.000000 | 135809.000000 | 135809.000000 | 135809.000000 |
| **mean** | 1.003130e+06 | 6.740540 | 0.419619 | 5.717714 | 8734.565765 |
| **std** | 1.786631e+03 | 6.239639 | 0.493498 | 3.696752 | 4767.233289 |
| **min** | 1.000001e+06 | 0.000000 | 0.000000 | 1.000000 | 12.000000 |
| **25%** | 1.001569e+06 | 1.000000 | 0.000000 | 3.000000 | 5433.000000 |
| **50%** | 1.003159e+06 | 4.000000 | 0.000000 | 5.000000 | 7914.000000 |
| **75%** | 1.004765e+06 | 11.000000 | 1.000000 | 8.000000 | 11400.000000 |
| **max** | 1.006039e+06 | 20.000000 | 1.000000 | 20.000000 | 23959.000000 |

In [93]:
```
num_people = 10000
num_samples = 100

means = []
stds = []
for i in range(num_people):
    sample = WMF['Purchase'].sample(num_samples)
    means.append(round(sample.mean(),2))
    stds.append(round(sample.mean(),2))
```

In [94]:
```
means
```

Out[94]:
```
[8721.82,
 8764.7,
 7969.66,
 8602.68,
 8840.88,
 8669.69,
 8379.92,
 9146.52,
 8758.54,
 9409.04,
 8264.77,
 8449.71,
 8232.15,
 9037.7,
 8891.59,
 8844.69,
 8895.26,
 8656.91,
 9611.71,
```

In [95]:
```
sum(means)/len(means)
```

Out[95]:
```
8731.888943999973
```

In [100]:
```
#Calculation upper confidence interval.
WCIU = (8731.88+((1.96*4767.23)/np.sqrt(10000)))
round(WCIU,2)
```

Out[100]:
```
8825.32
```

In [101]:
```
#Calculation lower confidence interval.
WCLU = (8731.88-((1.96*4767.23)/np.sqrt(10000)))
round(WCLU,2)
```

Out[101]:
```
8638.44
```

For men the 95% confidence interval is between 9160.92 & 9357.99 For women the confidence interval is between 8638.44 & 8825.32 This indicates that men are likely to spend more on Purchase than women

CREATING A DATASET FOR MARRIED AND UNMARRIED CUSTOMERS Married = 1 and dataset = WMMD Unmarried = 0 and dataset = WMS

In [103]:
```
WMMD.head()
```

Out[103]:

| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|
| **6** | 1000004 | P00184942 | M | 46-50 | 7 | B | 2 | 1 | 1 | 19215 |
| **7** | 1000004 | P00346142 | M | 46-50 | 7 | B | 2 | 1 | 1 | 15854 |
| **8** | 1000004 | P0097242 | M | 46-50 | 7 | B | 2 | 1 | 1 | 15686 |
| **9** | 1000005 | P00274942 | M | 26-35 | 20 | A | 1 | 1 | 8 | 7871 |
| **10** | 1000005 | P00251242 | M | 26-35 | 20 | A | 1 | 1 | 5 | 5254 |

In [104]:
```
WMS.head()
```

Out[104]:

| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1000001 | P00069042 | F | 0-17 | 10 | A | 2 | 0 | 3 | 8370 |
| **1** | 1000001 | P00248942 | F | 0-17 | 10 | A | 2 | 0 | 1 | 15200 |
| **2** | 1000001 | P00087842 | F | 0-17 | 10 | A | 2 | 0 | 12 | 1422 |
| **3** | 1000001 | P00085442 | F | 0-17 | 10 | A | 2 | 0 | 12 | 1057 |
| **4** | 1000002 | P00285442 | M | 55+ | 16 | C | 4+ | 0 | 8 | 7969 |

In [105]:
```
WMMD.describe()
```

Out[105]:

| | User_ID | Occupation | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|
| **count** | 2.253370e+05 | 225337.000000 | 225337.0 | 225337.000000 | 225337.000000 |
| **mean** | 1.003071e+06 | 8.266823 | 1.0 | 5.498245 | 9261.174574 |
| **std** | 1.765091e+03 | 6.687118 | 0.0 | 3.968868 | 5016.897378 |
| **min** | 1.000004e+06 | 0.000000 | 1.0 | 1.000000 | 12.000000 |
| **25%** | 1.001506e+06 | 2.000000 | 1.0 | 2.000000 | 5843.000000 |
| **50%** | 1.003093e+06 | 7.000000 | 1.0 | 5.000000 | 8051.000000 |
| **75%** | 1.004647e+06 | 14.000000 | 1.0 | 8.000000 | 12042.000000 |
| **max** | 1.006039e+06 | 20.000000 | 1.0 | 20.000000 | 23961.000000 |

In [106]:
```
num_people = 10000
num_samples = 100

means = []
stds = []
for i in range(num_people):
    sample = WMMD['Purchase'].sample(num_samples)
    means.append(round(sample.mean(),2))
    stds.append(round(sample.mean(),2))
```

In [107]:
```
means
```

```
 9155.79,
 8386.68,
 10085.47,
 9221.08,
 8806.89,
 9531.5,
 10334.29,
 9310.02,
 9835.62,
 9600.04,
 9267.64,
 8911.0,
 8846.98,
 10538.93,
 9212.75,
 8768.83,
 9688.71,
 8386.66,
 9778.48,
```

In [109]:
```
round(sum(means)/len(means),2)
```

Out[109]: 9264.05

In [111]:
```
#Calculation upper confidence interval.
MCIU = (9264.05+((1.96*5016.90)/np.sqrt(10000)))
round(WCIU,2)
```

Out[111]: 9362.38

In [113]:
```
#Calculation lower confidence interval.
MCIL = (9264.05-((1.96*5016.90)/np.sqrt(10000)))
round(MCIL,2)
```

Out[113]: 9165.72

For singles

In [114]:
```
num_people = 10000
num_samples = 100

means = []
stds = []
for i in range(num_people):
    sample = WMS['Purchase'].sample(num_samples)
    means.append(round(sample.mean(),2))
    stds.append(round(sample.mean(),2))
```

In [115]:
```
means
```
```
 9716.48,
 9087.42,
 9478.86,
 9612.14,
 9366.44,
 8725.42,
 9118.07,
 10078.16,
 9016.3,
 9455.98,
 10217.21,
 8892.04,
 8697.15,
 9130.1,
 10352.34,
 8860.88,
 9816.81,
 8881.55,
 8431.82,
 9711.58
```

In [116]:
```
sum(means)/len(means)
```
Out[116]: 9272.273515999941

WMS

In [117]:
```
WMS.describe()
```
Out[117]:

| | User_ID | Occupation | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|
| **count** | 3.247310e+05 | 324731.000000 | 324731.0 | 324731.000000 | 324731.000000 |
| **mean** | 1.002999e+06 | 7.944782 | 0.0 | 5.339059 | 9265.907619 |
| **std** | 1.700466e+03 | 6.402753 | 0.0 | 3.912070 | 5027.347859 |
| **min** | 1.000001e+06 | 0.000000 | 0.0 | 1.000000 | 12.000000 |
| **25%** | 1.001524e+06 | 3.000000 | 0.0 | 1.000000 | 5605.000000 |
| **50%** | 1.003065e+06 | 7.000000 | 0.0 | 5.000000 | 8044.000000 |
| **75%** | 1.004386e+06 | 14.000000 | 0.0 | 8.000000 | 12061.000000 |
| **max** | 1.006040e+06 | 20.000000 | 0.0 | 20.000000 | 23961.000000 |

In [118]:
```
#Calculation upper confidence interval.
SCIU = (9272.27+((1.96*5027.35)/np.sqrt(10000)))
round(SCIU,2)
```
Out[118]: 9370.81

In [119]:
```
#Calculation upper confidence interval.
SCIL = (9272.27-((1.96*5027.35)/np.sqrt(10000)))
round(SCIL,2)
```
Out[119]: 9173.73

For married customers the confidence interval is between 9165.72 & 9362.38 For single customers the confidence interval is between 9173.73 & 9370.81 The range is almost the same between married and single customers with single customers being slightly higher than the married customers

In [120]:
```
WM['Age'].value_counts()
```
Out[120]:
```
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 [122]:
```
WM0_17 = WM[WM['Age']=='0-17']
```

In [124]:
```
WM18_25 = WM[WM['Age']=='18-25']
```

In [125]:
```
WM26_35 = WM[WM['Age']=='26-35']
```

In [126]:
```
WM36_45 = WM[WM['Age']=='36-45']
```

In [127]:
```
WM46_50 = WM[WM['Age']=='46-50']
```

In [128]:
```
WM51_55 = WM[WM['Age']=='51-55']
```

In [129]:
```
WM55 = WM[WM['Age']=='55+']
```

Calculating confidence interval for 0-17

In [131]:
```
num_people = 10000
num_samples = 100

means = []
stds = []
for i in range(num_people):
    sample = WM0_17['Purchase'].sample(num_samples)
    means.append(round(sample.mean(),2))
    stds.append(round(sample.mean(),2))
```

In [132]:
```
means
```

```
 8234.17,
 9164.98,
 8551.08,
 9281.22,
 8624.8,
 8898.82,
 8560.64,
 9493.37,
 9033.54,
 8235.63,
 9042.51,
 8578.44,
 8482.57,
 9736.79,
 8545.48,
 9793.01,
 8728.93,
 8548.4,
 8165.24,
 8006.28,
```

In [136]:
```
round(sum(means)/len(means),2)
```

Out[136]: 8937.97

In [135]:
```
WM0_17.describe()
```

Out[135]:

| | User_ID | Occupation | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|
| **count** | 1.510200e+04 | 15102.000000 | 15102.0 | 15102.000000 | 15102.000000 |
| **mean** | 1.002722e+06 | 8.761025 | 0.0 | 5.083764 | 8933.464640 |
| **std** | 1.776555e+03 | 4.500672 | 0.0 | 3.800040 | 5111.114046 |
| **min** | 1.000001e+06 | 0.000000 | 0.0 | 1.000000 | 12.000000 |
| **25%** | 1.001263e+06 | 10.000000 | 0.0 | 2.000000 | 5328.000000 |
| **50%** | 1.002137e+06 | 10.000000 | 0.0 | 5.000000 | 7986.000000 |
| **75%** | 1.004493e+06 | 10.000000 | 0.0 | 8.000000 | 11874.000000 |
| **max** | 1.006006e+06 | 19.000000 | 0.0 | 20.000000 | 23955.000000 |

In [137]:
```
#Calculation upper confidence interval.
WM0_17CIU = (8937.97+((1.96*5111.11)/np.sqrt(10000)))
round(WM0_17CIU,2)
```

Out[137]: 9038.15

In [138]:
```
#Calculation upper confidence interval.
WM0_17CIL = (8937.97-((1.96*5111.11)/np.sqrt(10000)))
round(WM0_17CIL,2)
```

Out[138]: 8837.79

Calculating confidence interval for age range 18_25

In [140]:
```
means = []
stds = []
for i in range(num_people):
    sample = WM18_25['Purchase'].sample(num_samples)
    means.append(round(sample.mean(),2))
    stds.append(round(sample.mean(),2))
```

In [141]:
```
means
```

```
 8991.92,
 9311.67,
 9140.74,
 9277.66,
 9146.25,
 9145.16,
 9672.74,
 8601.46,
 9132.3,
 9902.19,
 9254.73,
 9181.89,
 7791.9,
 9293.62,
 9200.91,
 8996.68,
 9281.65,
 8707.71,
 9711.15,
 10129.75,
```

In [142]:
```
round(sum(means)/len(means),2)
```

Out[142]: 9174.37

In [143]:
```
WM18_25.describe()
```

Out[143]:

| | User_ID | Occupation | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|
| **count** | 9.966000e+04 | 99660.000000 | 99660.000000 | 99660.000000 | 99660.000000 |
| **mean** | 1.002801e+06 | 6.736384 | 0.211880 | 5.111088 | 9169.663606 |
| **std** | 1.732154e+03 | 5.947651 | 0.408643 | 3.810009 | 5034.321997 |
| **min** | 1.000018e+06 | 0.000000 | 0.000000 | 1.000000 | 12.000000 |
| **25%** | 1.001314e+06 | 4.000000 | 0.000000 | 1.000000 | 5415.000000 |
| **50%** | 1.002854e+06 | 4.000000 | 0.000000 | 5.000000 | 8027.000000 |
| **75%** | 1.004217e+06 | 11.000000 | 0.000000 | 8.000000 | 12028.000000 |
| **max** | 1.006031e+06 | 20.000000 | 1.000000 | 20.000000 | 23958.000000 |

In [144]:
```
#Calculation upper confidence interval.
WM18_25CIU = (9174.37+((1.96*5034.32)/np.sqrt(10000)))
round(WM18_25CIU,2)
```

Out[144]: 9273.04

In [145]:
```
#Calculation upper confidence interval.
WM18_25CIL = (9174.37-((1.96*5034.32)/np.sqrt(10000)))
round(WM18_25CIL,2)
```

Out[145]: 9075.7

Calculating confidence interval for age range 26_35

In [146]:
```
means = []
stds = []
for i in range(num_people):
    sample = WM26_35['Purchase'].sample(num_samples)
    means.append(round(sample.mean(),2))
    stds.append(round(sample.mean(),2))
```

In [147]:
```
means
```

Out[147]:
```
[8924.13,
 9096.95,
 9653.83,
 9405.86,
 10045.85,
 8893.15,
 9147.8,
 8946.44,
 8467.86,
 8898.61,
 9105.31,
 9913.21,
 8691.3,
 8764.1,
 10124.05,
 8051.61,
 8711.63,
 9578.55,
 10480.21,
```

In [148]:
```
round(sum(means)/len(means),2)
```

Out[148]: 9249.59

In [149]:
```
WM26_35.describe()
```

Out[149]:

| | User_ID | Occupation | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|
| **count** | 2.195870e+05 | 219587.000000 | 219587.000000 | 219587.000000 | 219587.000000 |
| **mean** | 1.003113e+06 | 7.896975 | 0.392970 | 5.314272 | 9252.690633 |
| **std** | 1.732500e+03 | 6.694999 | 0.488411 | 3.886768 | 5010.527303 |
| **min** | 1.000003e+06 | 0.000000 | 0.000000 | 1.000000 | 12.000000 |
| **25%** | 1.001599e+06 | 2.000000 | 0.000000 | 1.000000 | 5475.000000 |
| **50%** | 1.003243e+06 | 7.000000 | 0.000000 | 5.000000 | 8030.000000 |
| **75%** | 1.004524e+06 | 14.000000 | 1.000000 | 8.000000 | 12047.000000 |
| **max** | 1.006040e+06 | 20.000000 | 1.000000 | 20.000000 | 23961.000000 |

In [150]:
```
#Calculation upper confidence interval.
WM26_35CIU = (9249.59+((1.96*5010.53)/np.sqrt(10000)))
round(WM26_35CIU,2)
```

Out[150]: 9347.8

In [151]:
```
#Calculation lower confidence interval.
WM26_35CIL = (9249.59-((1.96*5010.53)/np.sqrt(10000)))
round(WM26_35CIL,2)
```

Out[151]: 9151.38

Calculating confidence interval for age range 36_45

In [152]:
```
means = []
stds = []
for i in range(num_people):
    sample =WM36_45['Purchase'].sample(num_samples)
    means.append(round(sample.mean(),2))
    stds.append(round(sample.mean(),2))
```

In [153]:
```
means
```

Out[153]:
```
[10050.21,
 9356.72,
 9176.14,
 9225.59,
 9532.19,
 9206.74,
 8636.56,
 8893.02,
 8953.68,
 9137.01,
 9393.3,
 9369.97,
 8383.05,
 9251.93,
 9061.0,
 9531.87,
 8561.32,
 8943.7,
 8994.92,
```

In [154]:
```
round(sum(means)/len(means),2)
```

Out[154]:
```
9326.69
```

In [155]:
```
WM36_45.describe()
```

Out[155]:

| | User_ID | Occupation | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|
| **count** | 1.100130e+05 | 110013.000000 | 110013.000000 | 110013.000000 | 110013.000000 |
| **mean** | 1.003066e+06 | 8.837365 | 0.396644 | 5.494242 | 9331.350695 |
| **std** | 1.689593e+03 | 6.589059 | 0.489203 | 3.988229 | 5022.923879 |
| **min** | 1.000007e+06 | 0.000000 | 0.000000 | 1.000000 | 12.000000 |
| **25%** | 1.001598e+06 | 2.000000 | 0.000000 | 1.000000 | 5876.000000 |
| **50%** | 1.003050e+06 | 7.000000 | 0.000000 | 5.000000 | 8061.000000 |
| **75%** | 1.004488e+06 | 16.000000 | 1.000000 | 8.000000 | 12107.000000 |
| **max** | 1.006026e+06 | 20.000000 | 1.000000 | 20.000000 | 23960.000000 |

In [156]:
```
#Calculation upper confidence interval.
WM36_45CIU = (9326.69+((1.96*5022.92)/np.sqrt(10000)))
round(WM36_45CIU,2)
```

Out[156]:
```
9425.14
```

In [157]:
```
#Calculation lower confidence interval.
WM36_45CIL = (9326.69-((1.96*5022.92)/np.sqrt(10000)))
round(WM36_45CIL,2)
```

Out[157]:
```
9228.24
```

Calculating confidence interval for age range 46_50

In [158]:
```
WM46_50.describe()
```

Out[158]:

| | User_ID | Occupation | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|
| **count** | 4.570100e+04 | 45701.000000 | 45701.000000 | 45701.000000 | 45701.000000 |
| **mean** | 1.003190e+06 | 8.517078 | 0.722326 | 5.742194 | 9208.625697 |
| **std** | 1.777321e+03 | 6.676416 | 0.447857 | 4.047325 | 4967.216367 |
| **min** | 1.000004e+06 | 0.000000 | 0.000000 | 1.000000 | 12.000000 |
| **25%** | 1.001798e+06 | 1.000000 | 0.000000 | 2.000000 | 5888.000000 |
| **50%** | 1.003430e+06 | 7.000000 | 1.000000 | 5.000000 | 8036.000000 |
| **75%** | 1.004661e+06 | 16.000000 | 1.000000 | 8.000000 | 11997.000000 |
| **max** | 1.006039e+06 | 20.000000 | 1.000000 | 20.000000 | 23960.000000 |

In [159]:
```
means = []
stds = []
for i in range(num_people):
    sample = WM46_50['Purchase'].sample(num_samples)
    means.append(round(sample.mean(),2))
    stds.append(round(sample.mean(),2))
```

In [161]:
```
means
```

Out[161]:
```
[8635.56,
 8911.07,
 9562.28,
 9317.37,
 9578.98,
 9060.79,
 10067.95,
 9075.03,
 8919.96,
 8392.75,
 8845.07,
 9230.85,
 8825.32,
 9457.54,
 9240.54,
 9120.83,
 8679.98,
 8692.49,
 8949.11,
```

In [162]:
```
round(sum(means)/len(means),2)
```

Out[162]:
```
9212.25
```

In [163]:
```
#Calculation upper confidence interval.
WM46_50CIU = (9212.25+((1.96*4967.22)/np.sqrt(10000)))
round(WM46_50CIU,2)
```

Out[163]:
```
9309.61
```

In [164]:
```
#Calculation upper confidence interval.
WM46_50CIL = (9212.25-((1.96*4967.22)/np.sqrt(10000)))
round(WM46_50CIL,2)
```

Out[164]:
```
9114.89
```

Calculating confidence interval for age range 51_55

In [165]:
```
WM51_55.describe()
```

Out[165]:

| | User_ID | Occupation | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|
| **count** | 3.850100e+04 | 38501.000000 | 38501.000000 | 38501.000000 | 38501.000000 |
| **mean** | 1.002985e+06 | 8.810109 | 0.718475 | 5.774214 | 9534.808031 |
| **std** | 1.680563e+03 | 6.669887 | 0.449749 | 4.107277 | 5087.368080 |
| **min** | 1.000006e+06 | 0.000000 | 0.000000 | 1.000000 | 12.000000 |
| **25%** | 1.001591e+06 | 2.000000 | 0.000000 | 2.000000 | 6017.000000 |
| **50%** | 1.002878e+06 | 7.000000 | 1.000000 | 5.000000 | 8130.000000 |
| **75%** | 1.004373e+06 | 16.000000 | 1.000000 | 8.000000 | 12462.000000 |
| **max** | 1.006033e+06 | 20.000000 | 1.000000 | 20.000000 | 23960.000000 |

In [166]:
```
means = []
stds = []
for i in range(num_people):
    sample = WM51_55['Purchase'].sample(num_samples)
    means.append(round(sample.mean(),2))
    stds.append(round(sample.mean(),2))
```

In [168]:
```
means
```

Out[168]:
```
[9227.55,
 10645.19,
 9557.9,
 9546.36,
 9778.59,
 10063.12,
 8779.27,
 10056.93,
 9411.52,
 9528.84,
 9599.78,
 9049.17,
 9509.69,
 9167.54,
 9011.37,
 9379.44,
 10106.38,
 9490.98,
 8956.26,
```

In [169]:
```
round(sum(means)/len(means),2)
```

Out[169]:
```
9538.48
```

In [170]:
```
#Calculation upper confidence interval.
WM51_55CIU = (9538.48+((1.96*5087.37)/np.sqrt(10000)))
round(WM51_55CIU,2)
```

Out[170]:
```
9638.19
```

In [171]:
```
#Calculation lower confidence interval.
WM51_55CIL = (9538.48-((1.96*5087.37)/np.sqrt(10000)))
round(WM51_55CIL,2)
```

Out[171]:
```
9438.77
```

Calculating confidence interval for age 55 and over

In [172]:
```
WM55.describe()
```

Out[172]:

| | User_ID | Occupation | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|
| **count** | 2.150400e+04 | 21504.000000 | 21504.000000 | 21504.000000 | 21504.000000 |
| **mean** | 1.002986e+06 | 9.502697 | 0.633417 | 6.066313 | 9336.280459 |
| **std** | 1.659541e+03 | 6.370448 | 0.481882 | 4.091461 | 5011.493996 |
| **min** | 1.000002e+06 | 0.000000 | 0.000000 | 1.000000 | 12.000000 |
| **25%** | 1.001739e+06 | 2.000000 | 0.000000 | 3.000000 | 6018.000000 |
| **50%** | 1.002661e+06 | 13.000000 | 1.000000 | 5.000000 | 8105.500000 |
| **75%** | 1.004193e+06 | 14.000000 | 1.000000 | 8.000000 | 11932.000000 |
| **max** | 1.006038e+06 | 20.000000 | 1.000000 | 20.000000 | 23960.000000 |

In [173]:
```
means = []
stds = []
for i in range(num_people):
    sample = WM55['Purchase'].sample(num_samples)
    means.append(round(sample.mean(),2))
    stds.append(round(sample.mean(),2))
```

In [174]:
```
means
```

```
 8724.95,
 9810.16,
 9260.61,
 8706.44,
 8774.53,
 10055.58,
 9215.07,
 9556.29,
 9179.33,
 9626.99,
 9455.44,
 9022.54,
 10033.74,
 9545.02,
 9113.14,
 9790.72,
 9868.93,
 8822.73,
 9550.31,
```

In [175]:
```
round(sum(means)/len(means),2)
```

Out[175]: 9329.97

In [176]:
```
#Calculation upper confidence interval.
WM55CIU = (9329.97+((1.96*5011.49)/np.sqrt(10000)))
round(WM55CIU,2)
```

Out[176]: 9428.2

In [177]:
```
#Calculation lower confidence interval.
WM55CIL = (9329.97-((1.96*5011.49)/np.sqrt(10000)))
round(WM55CIL,2)
```

Out[177]: 9231.74

The 95% confidence interval for age range of 0-17 is in between 8837.79 & 9038.15

The 95% confidence interval for age range of 18_25 is in between 9075.7 & 9273.04

The 95% confidence interval for age range of 26_35 is in between 9151.38 & 9347.8

The 95% confidence interval for age range of 36_45 is in between 9228.24 & 9425.14

The 95% confidence interval for age range of 46_50 is in between 9114.89 & 9309.61

The 95% confidence interval for age range of 51_55 is in between 9438.77 & 9638.19

The 95% confidence interval for age range of 55 and above is in between 9231.74 & 9428.2

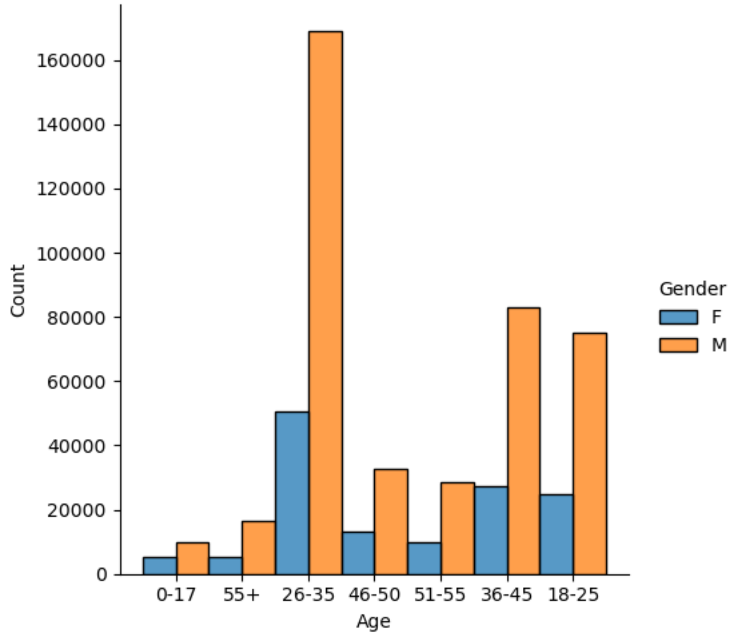
Confidence interval is highest in the age range of 51_55

In [180]:
```
WM.head()
```

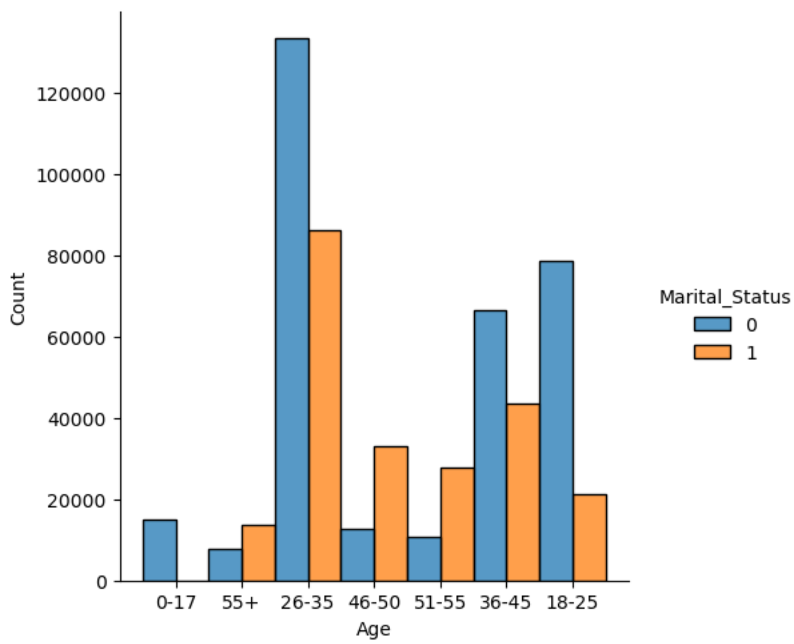
Out[180]:

| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years | Marital_Status | Product_Category | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1000001 | P00069042 | F | 0-17 | 10 | A | 2 | 0 | 3 | 8370 |
| **1** | 1000001 | P00248942 | F | 0-17 | 10 | A | 2 | 0 | 1 | 15200 |
| **2** | 1000001 | P00087842 | F | 0-17 | 10 | A | 2 | 0 | 12 | 1422 |
| **3** | 1000001 | P00085442 | F | 0-17 | 10 | A | 2 | 0 | 12 | 1057 |
| **4** | 1000002 | P00285442 | M | 55+ | 16 | C | 4+ | 0 | 8 | 7969 |

In [182]:
```
sns.displot(WM, x="Age", hue="Gender",multiple="dodge")
plt.show()
```



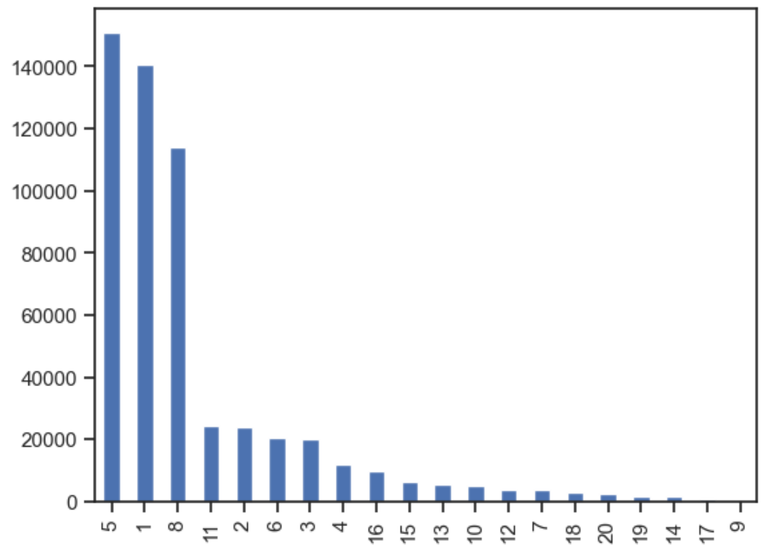

In [183]:
```
sns.displot(WM, x="Age", hue="Marital_Status",multiple="dodge")
plt.show()
```



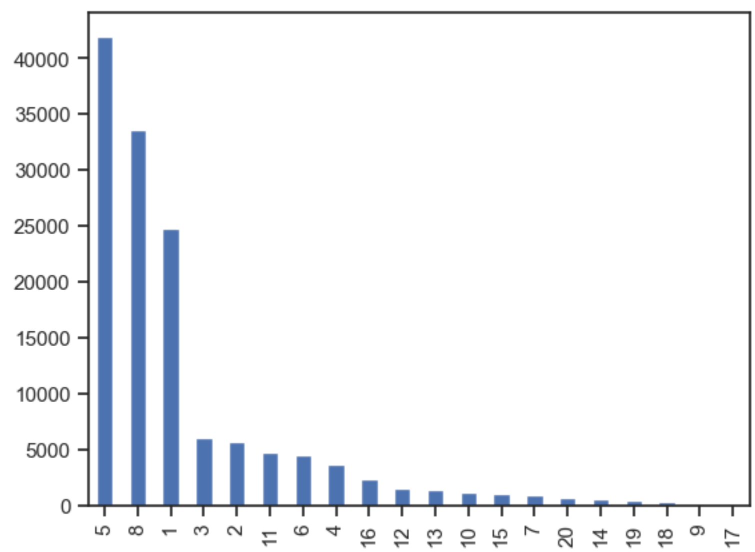
In [185]:
```
WM['Product_Category'].value_counts()
```

Out[185]:
```
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
```

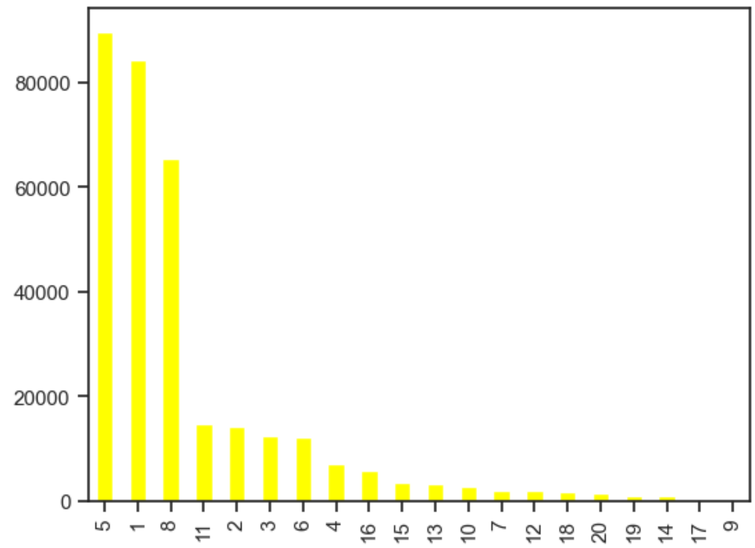
In [212]:
```
WM['Product_Category'].value_counts().plot(kind='bar')
plt.show()
```




In [213]:
```
WMF['Product_Category'].value_counts().plot(kind='bar')
plt.show()
```

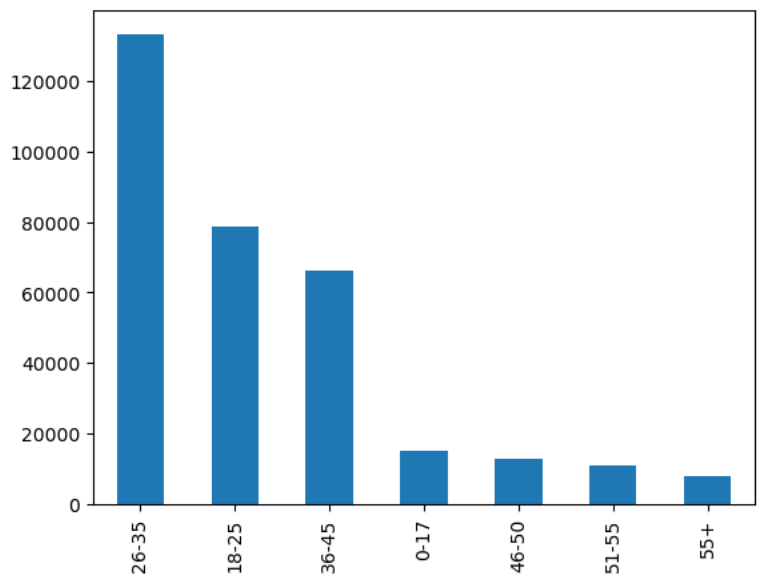


In [214]:
```
WMM['Product_Category'].value_counts().plot(kind='bar',color='yellow')
plt.show()
```



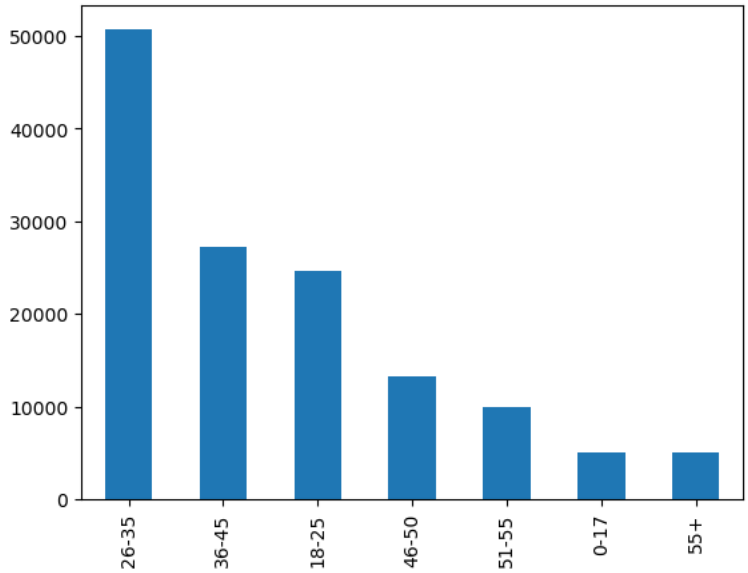

In [190]:
```
WMM['Age'].value_counts().plot(kind='bar')
```

Out[190]: <AxesSubplot:>

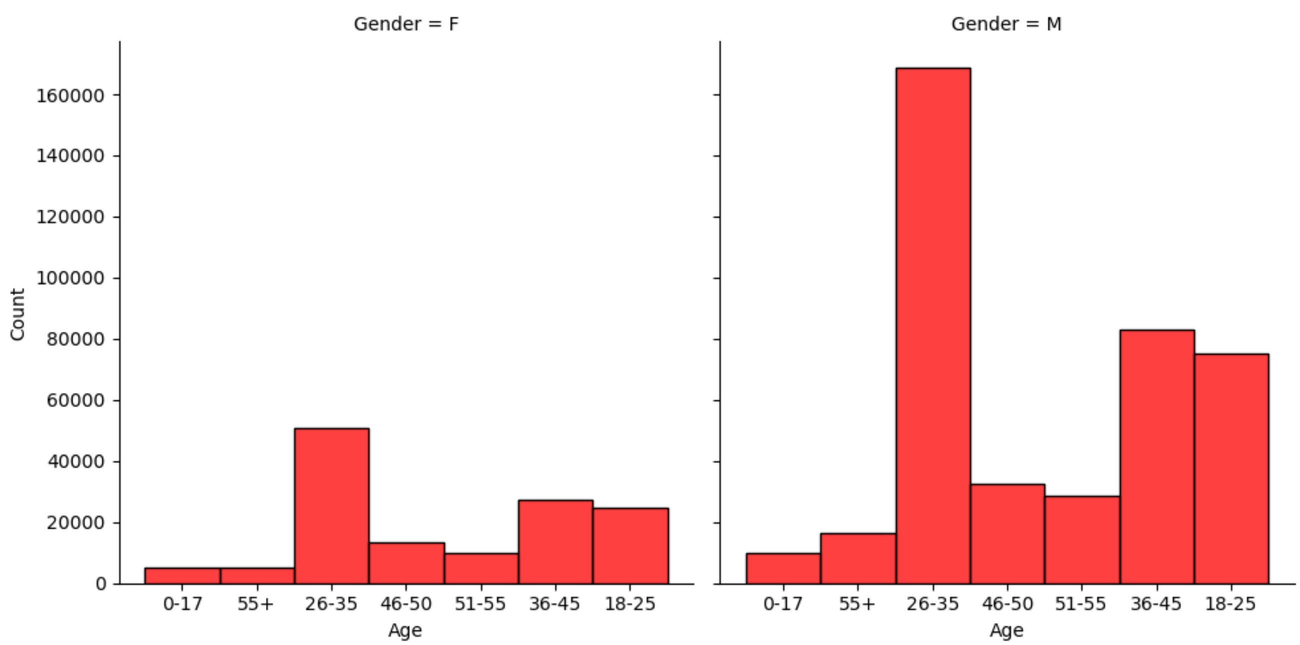


In [191]:
```
WMF['Age'].value_counts().plot(kind='bar')
```

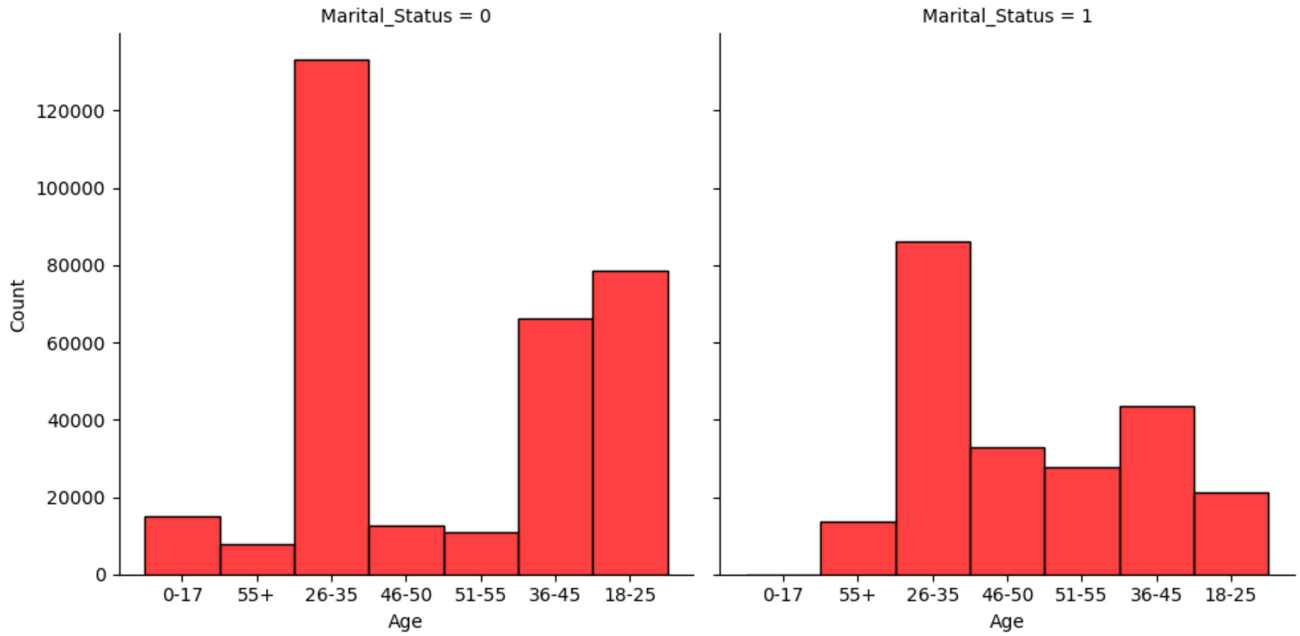
Out[191]: <AxesSubplot:>



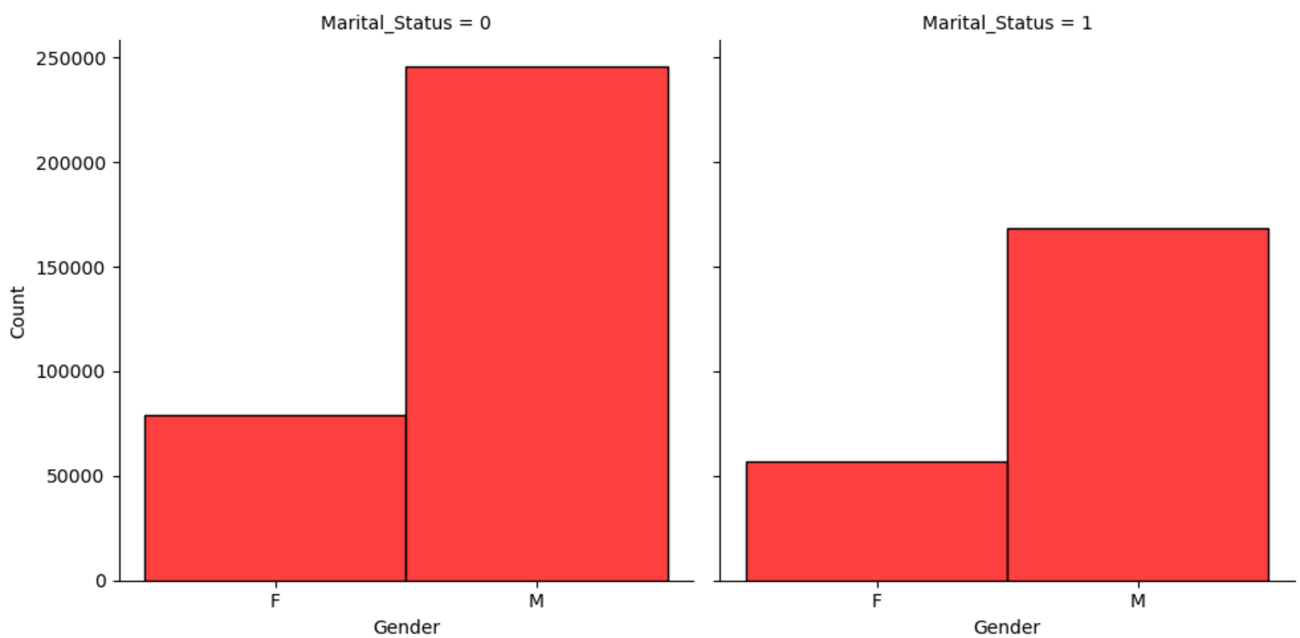

In [196]:
```
sns.displot(WM, x="Age", col="Gender",color='red')
#plt.title('Product purchased by Gender')
plt.show()
```



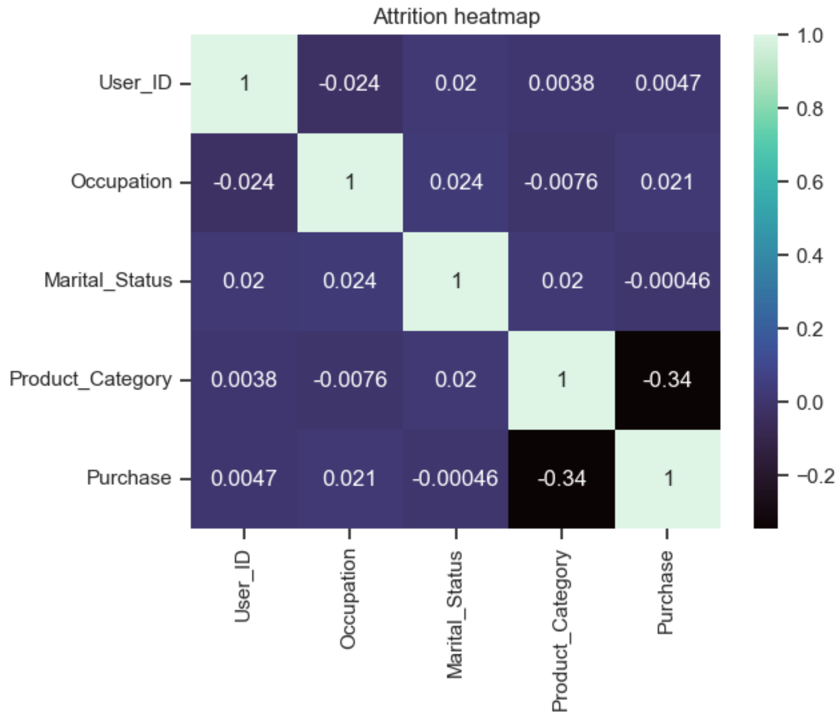
In [197]:
```
sns.displot(WM, x="Age", col="Marital_Status",color='red')
#plt.title('Product purchased by Gender')
plt.show()
```



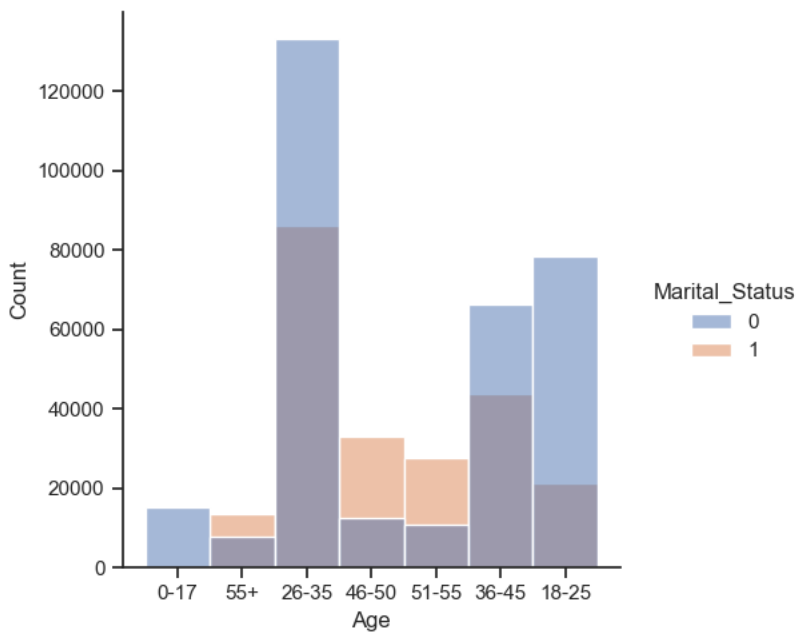

In [201]:
```
sns.displot(WM, x="Gender", col="Marital_Status",color='red')
#plt.title('Product purchased by Gender')
plt.show()
```



In [208]:
```
ax = sns.heatmap(WM.corr(),annot=True,cmap='mako')
plt.title('Attrition heatmap')
plt.show()
```




In [209]:
```
sns.displot(WM,x='Age',hue='Marital_Status')
plt.show()
```




# CONCLUSION

For men the 95% confidence interval is between 9160.92 & 9357.99

For women the confidence interval is between 8638.44 & 8825.32

The confidence interval for men and women are not overlapping and men have a higher confidence interval.

For married and unmarried customers the confidence interval is overlapping .

For married customers the confidence interval is between 9165.72 & 9362.38 For single customers the confidence interval is between 9173.73 & 9370.81 The range is almost the same between married and single customers with single customers being slightly higher than the married customers

Confidence interval based on different age ranges

The 95% confidence interval for age range of 0-17 is in between 8837.79 & 9038.15

The 95% confidence interval for age range of 18_25 is in between 9075.7 & 9273.04

The 95% confidence interval for age range of 26_35 is in between 9151.38 & 9347.8

The 95% confidence interval for age range of 36_45 is in between 9228.24 & 9425.14

The 95% confidence interval for age range of 46_50 is in between 9114.89 & 9309.61

The 95% confidence interval for age range of 51_55 is in between 9438.77 & 9638.19

The 95% confidence interval for age range of 55 and above is in between 9231.74 & 9428.2

Confidence interval is highest in the age range of 51_55

Amongst the age ranges the highest confidence interval is for age range of 51_55.

Overall the product category 5 is the highest amongst both male and female customers .

The highest count of customers is for the age group 26-35 .

Recommendations

Male customers have a higher confidence interval in purchase in comparison to women .So offers focused on bringing in more male customers would improve the business .

The age range of customers 26-35 has the highest count .Offers focused on this age range can improve the business as there is a hight count of visitors.

The age range of customers 51-55 has the highest confidence interval. Offers focused on this age range can improve the business as the confidence interval is high.

The product category 5 is the highest count amongst both male and female customers. Understanding its features and adding more of product 5 will improve business.

The product category 15,13,10,12,18,7,20,19,14,17,9 seem to have very poor performance .If possible they can be discontinued .

In [ ]:

```

```