walmartconfidenceintervalandclt

July 8, 2024

About Walmart:

Walmart is an American multinational retail corporation that operates a chain of supercenters, discount departmental stores, and grocery stores from the United States. Walmart has more than 100 million customers worldwide.

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).

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

[2]:
```
df = pd.read_csv('walmart_data.csv')
df.head()
```

[2]:
```
   User_ID Product_ID Gender   Age  Occupation City_Category  \
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

  Stay_In_Current_City_Years  Marital_Status  Product_Category  Purchase
0                          2               0                 3      8370
1                          2               0                 1     15200
2                          2               0                12      1422
3                          2               0                12      1057
4                         4+               0                 8      7969
```

[3]:
```
df.shape
```

[3]: (550068, 10)

[4]:
```
df.describe()
```

[4]:
```
            User_ID     Occupation  Marital_Status  Product_Category  \
count  5.500680e+05  550068.000000   550068.000000     550068.000000
mean   1.003029e+06       8.076707        0.409653          5.404270
std    1.727592e+03       6.522660        0.491770          3.936211
min    1.000001e+06       0.000000        0.000000          1.000000
25%    1.001516e+06       2.000000        0.000000          1.000000
50%    1.003077e+06       7.000000        0.000000          5.000000
75%    1.004478e+06      14.000000        1.000000          8.000000
max    1.006040e+06      20.000000        1.000000         20.000000

            Purchase
count  550068.000000
mean     9263.968713
std      5023.065394
min        12.000000
25%      5823.000000
50%      8047.000000
75%     12054.000000
max     23961.000000
```

[5]:
```
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  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
```

[6]:
```
#converting int to object

columns = ['User_ID','Occupation','Marital_Status','Product_Category']
```

```
df[columns] = df[columns].astype('object')
df
```

[6]:
```
        User_ID Product_ID Gender    Age Occupation City_Category  \
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
…           …         …      …     …          …
550063  1006033  P00372445      M  51-55         13             B
550064  1006035  P00375436      F  26-35          1             C
550065  1006036  P00375436      F  26-35         15             B
550066  1006038  P00375436      F    55+          1             C
550067  1006039  P00371644      F  46-50          0             B

       Stay_In_Current_City_Years Marital_Status Product_Category  Purchase
0                               2              0                3      8370
1                               2              0                1     15200
2                               2              0               12      1422
3                               2              0               12      1057
4                              4+              0                8      7969
…                             …              …                …        …
550063                          1              1               20       368
550064                          3              0               20       371
550065                         4+              1               20       137
550066                          2              0               20       365
550067                         4+              1               20       490

[550068 rows x 10 columns]
```

[7]:
```
df.isnull().sum()
```

[7]:
```
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
```

[8]:
```
df.duplicated().sum()
```

[8]: 0

[9]:
```python
#counting values by categories

Age = df['Age'].value_counts().sort_values(ascending = False)
Gender = df['Gender'].value_counts().sort_values(ascending = False)
Occupation = df['Occupation'].value_counts().sort_values(ascending = False)
Marital_Status = df['Marital_Status'].value_counts().sort_values(ascending =␣
 ↪False)
Product_Category = df['Product_Category'].value_counts().sort_values(ascending␣
 ↪= False)

print(Age)
print('=======')
print(Gender)
print('=======')
print(Occupation)
print('=======')
print(Marital_Status)
print('=======')
print(Product_Category)
```

```
Age
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: count, dtype: int64
=======
Gender
M    414259
F    135809
Name: count, dtype: int64
=======
Occupation
4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
2     26588
```

```
16    25371
6     20355
3     17650
10    12930
5     12177
15    12165
11    11586
19     8461
13     7728
18     6622
9      6291
8      1546
Name: count, dtype: int64
=======
Marital_Status
0    324731
1    225337
Name: count, dtype: int64
=======
Product_Category
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: count, dtype: int64
```

[10]:

```
#counting unique values

unique_age = df['Age'].unique()
unique_gender = df['Gender'].unique()
unique_occupation = df['Occupation'].unique()
```

```
unique_marital_status = df['Marital_Status'].unique()
unique_product_category = df['Product_Category'].unique()

print('unique_age = ', unique_age)
print('=======')
print('unique_gender = ', unique_gender)
print('=======')
print('unique_occupation = ', unique_occupation)
print('=======')
print('unique_marital_status = ', unique_marital_status)
print('=======')
print('unique_product_category = ', unique_product_category)
```

```
unique_age =  ['0-17' '55+' '26-35' '46-50' '51-55' '36-45' '18-25']
=======
unique_gender =  ['F' 'M']
=======
unique_occupation =  [10 16 15 7 20 9 1 12 17 0 3 4 11 8 19 2 18 5 14 13 6]
=======
unique_marital_status =  [0 1]
=======
unique_product_category =  [3 1 12 8 5 4 2 6 14 11 13 15 7 16 18 10 17 9 20 19]
```

[11]:
```
len(df)
```

[11]:
```
550068
```

Insights: * There are no missing values in the data.

- There are 3631 unique product IDs in the dataset. P00265242 is the most sold Product ID.
- There are 7 unique age groups.
- The most product sold is amongst the age group of 26-30.
- The spread between men and women are 414259 and 135809, the difference is significant in terms of buying the product.
- Unmarried customers tend to buy more products than married.
- The product category which is sold the most is P00265242.
- There are 20 unique product categories with 5 being the highest.

[12]:
```
pd.crosstab(df['Age'], df['Gender'], margins = True)
```

[12]:

Gender	F	M	All
Age			
0-17	5083	10019	15102
18-25	24628	75032	99660
26-35	50752	168835	219587

```
36-45     27170   82843  110013
46-50     13199   32502   45701
51-55      9894   28607   38501
55+        5083   16421   21504
All      135809  414259  550068
```

[13]:
```python
pd.crosstab(df['Age'], df['Gender'], margins = True, normalize = 'index')
```

[13]:
```
Gender          F         M
Age
0-17     0.336578  0.663422
18-25    0.247120  0.752880
26-35    0.231125  0.768875
36-45    0.246971  0.753029
46-50    0.288812  0.711188
51-55    0.256980  0.743020
55+      0.236375  0.763625
All      0.246895  0.753105
```

[14]:
```python
pd.crosstab(df['Marital_Status'], df['Gender'], margins = True)
```

[14]:
```
Gender                F       M     All
Marital_Status
0                 78821  245910  324731
1                 56988  168349  225337
All              135809  414259  550068
```

[15]:
```python
pd.crosstab(df['Marital_Status'], df['Gender'], margins = True, normalize=
↪'index')
```

[15]:
```
Gender                  F         M
Marital_Status
0                0.242727  0.757273
1                0.252901  0.747099
All              0.246895  0.753105
```

[16]:
```python
pd.crosstab(df['Occupation'], df['Gender'], margins = True)
```

[16]:
```
Gender          F      M    All
Occupation
0           18112  51526  69638
1           17984  29442  47426
2            8629  17959  26588
3            7919   9731  17650
4           17836  54472  72308
5            2220   9957  12177
6            8160  12195  20355
```

```
7             10028   49105   59133
8               361    1185    1546
9              5843     448    6291
10             4003    8927   12930
11             1500   10086   11586
12             3469   27710   31179
13             1498    6230    7728
14             6763   20546   27309
15             2390    9775   12165
16             4107   21264   25371
17             3929   36114   40043
18              230    6392    6622
19             2017    6444    8461
20             8811   24751   33562
All          135809  414259  550068
```

[17]:
```python
pd.crosstab(df['Occupation'], df['Gender'], margins = True, normalize = 'index')
```

[17]:
```
Gender             F         M
Occupation
0           0.260088  0.739912
1           0.379201  0.620799
2           0.324545  0.675455
3           0.448669  0.551331
4           0.246667  0.753333
5           0.182311  0.817689
6           0.400884  0.599116
7           0.169584  0.830416
8           0.233506  0.766494
9           0.928787  0.071213
10          0.309590  0.690410
11          0.129467  0.870533
12          0.111261  0.888739
13          0.193841  0.806159
14          0.247647  0.752353
15          0.196465  0.803535
16          0.161878  0.838122
17          0.098120  0.901880
18          0.034733  0.965267
19          0.238388  0.761612
20          0.262529  0.737471
All         0.246895  0.753105
```

[18]:
```python
pd.crosstab(df['Product_Category'], df['Gender'], margins = True)
```

[18]:
```
Gender                 F       M     All
Product_Category
```

1	24831	115547	140378
2	5658	18206	23864
3	6006	14207	20213
4	3639	8114	11753
5	41961	108972	150933
6	4559	15907	20466
7	943	2778	3721
8	33558	80367	113925
9	70	340	410
10	1162	3963	5125
11	4739	19548	24287
12	1532	2415	3947
13	1462	4087	5549
14	623	900	1523
15	1046	5244	6290
16	2402	7426	9828
17	62	516	578
18	382	2743	3125
19	451	1152	1603
20	723	1827	2550
All	135809	414259	550068

[19]:
```
pd.crosstab(df['Product_Category'], df['Gender'], margins = True, normalize =
 ↪'index')
```

[19]:

Gender	F	M
Product_Category		
1	0.176887	0.823113
2	0.237094	0.762906
3	0.297136	0.702864
4	0.309623	0.690377
5	0.278011	0.721989
6	0.222760	0.777240
7	0.253426	0.746574
8	0.294562	0.705438
9	0.170732	0.829268
10	0.226732	0.773268
11	0.195125	0.804875
12	0.388143	0.611857
13	0.263471	0.736529
14	0.409061	0.590939
15	0.166296	0.833704
16	0.244404	0.755596
17	0.107266	0.892734
18	0.122240	0.877760
19	0.281347	0.718653
20	0.283529	0.716471

```
All                 0.246895  0.753105
```

Insights:

- Only 25% women purchase the product, remaining being men.
- Amongst married and unmarried, the distribution of gender is same as overall.
- In the overall occupation, men are dominating at around 75%, the most number of women is in occupation 9 which is around 92%.

1 Checking how the data is spread basis distinct users

[20]:
```
df2 = df.groupby(['User_ID'])['Age'].unique()
df2.value_counts()/len(df2)*100
```

[20]:
```
Age
[26-35]    34.849771
[36-45]    19.809879
[18-25]    18.146325
[46-50]     9.013750
[51-55]     8.164997
[55+]       6.314717
[0-17]      3.700560
Name: count, dtype: float64
```

Insight:

- Customers of age [26-35] covers around 35% of market.

[21]:
```
df1 = df.groupby(['User_ID'])['Occupation'].unique()
df1.value_counts()/len(df1)*100
```

[21]:
```
Occupation
[4]     12.561535
[0]     11.678832
[7]     11.356306
[1]      8.776099
[17]     8.334748
[12]     6.382618
[14]     4.990664
[20]     4.634188
[2]      4.345612
[16]     3.989136
[6]      3.870311
[10]     3.259209
[3]      2.885758
[15]     2.376507
[13]     2.376507
```

```
[11]      2.172806
[5]       1.884230
[9]       1.493804
[19]      1.205228
[18]      1.137328
[8]       0.288576
Name: count, dtype: float64
```

Insight:

- Occupation 4 covers 12% of customers.

[22]:
```
df3 = df.groupby(['User_ID'])['Marital_Status'].unique()
df3.value_counts()/len(df3)*100
```

[22]:
```
Marital_Status
[0]    58.003735
[1]    41.996265
Name: count, dtype: float64
```

Insight:

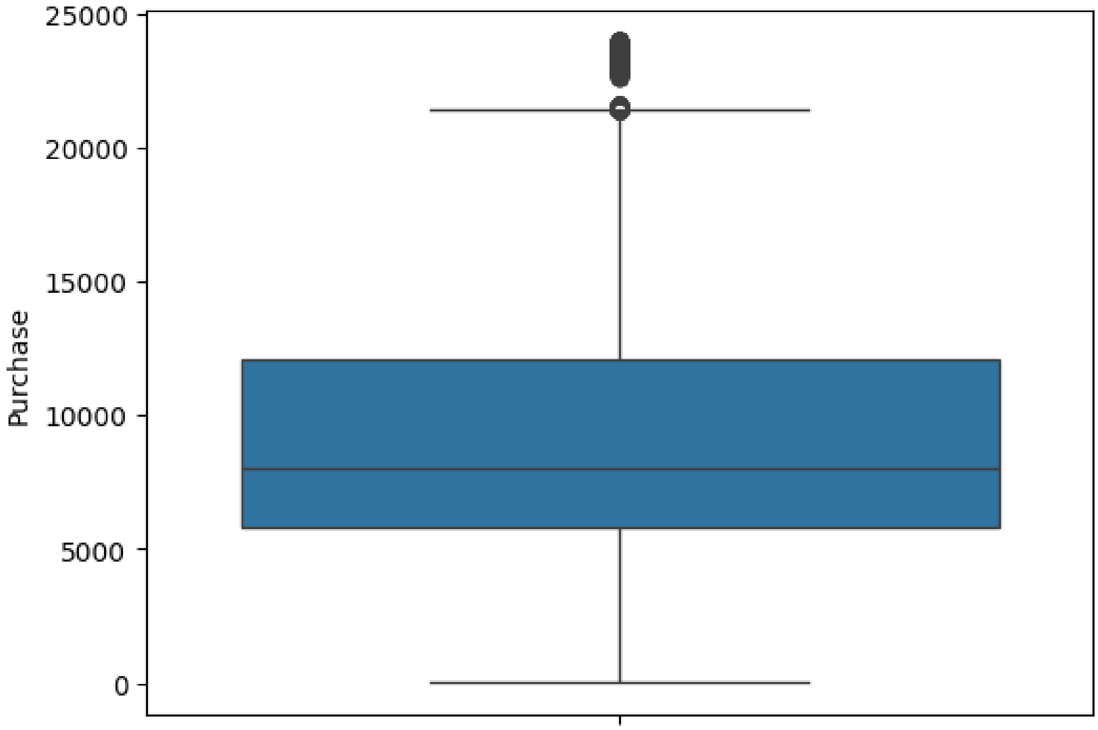
- We have 58% of the single users and 42% of married users. Combining with previous observation, single users contributes more as 58% of the single contributes to the 60% of the purchase count.

[23]:
```
df4 = df.groupby(['User_ID'])['Product_Category'].unique()
df4.value_counts()/len(df4)*100
```

[23]:
```
Product_Category
[1]                                                      0.050925
[3, 1, 12, 2, 4, 5, 8, 6, 14, 16, 20]                    0.016975
[5, 3, 1, 4, 8, 2, 11, 13, 6, 10, 7, 16, 20]             0.016975
[1, 11, 3, 5, 8, 4, 16, 2, 19]                           0.016975
[5, 3, 11, 1, 20]                                        0.016975
                                                           …
[1, 8, 6, 2, 11, 18, 5, 15, 4, 10, 13, 12, 16, 7, 17, 3, 19]    0.016975
[1, 8, 16, 5, 6, 13]                                     0.016975
[5, 8, 1, 4, 14, 3, 2, 13, 11, 12, 15, 10, 6, 18, 20]    0.016975
[11, 1, 5, 8, 15, 4, 3, 2, 12, 6, 14, 7, 18, 16, 10, 20]    0.016975
[1, 5, 8, 6, 3, 15, 10, 16, 11, 7, 2, 9, 13, 4]          0.016975
Name: count, Length: 5889, dtype: float64
```

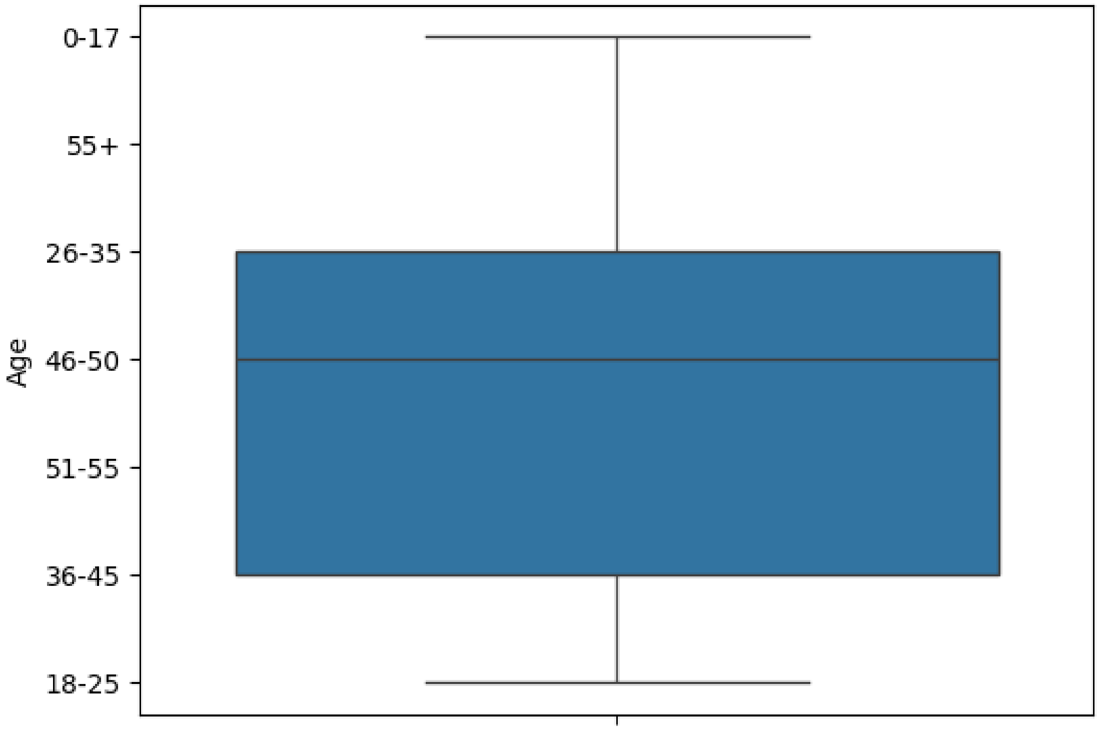
[24]:
```
#checking out for outliers

sns.boxplot(df['Purchase'])
```

[24]:
```
<Axes: ylabel='Purchase'>
```



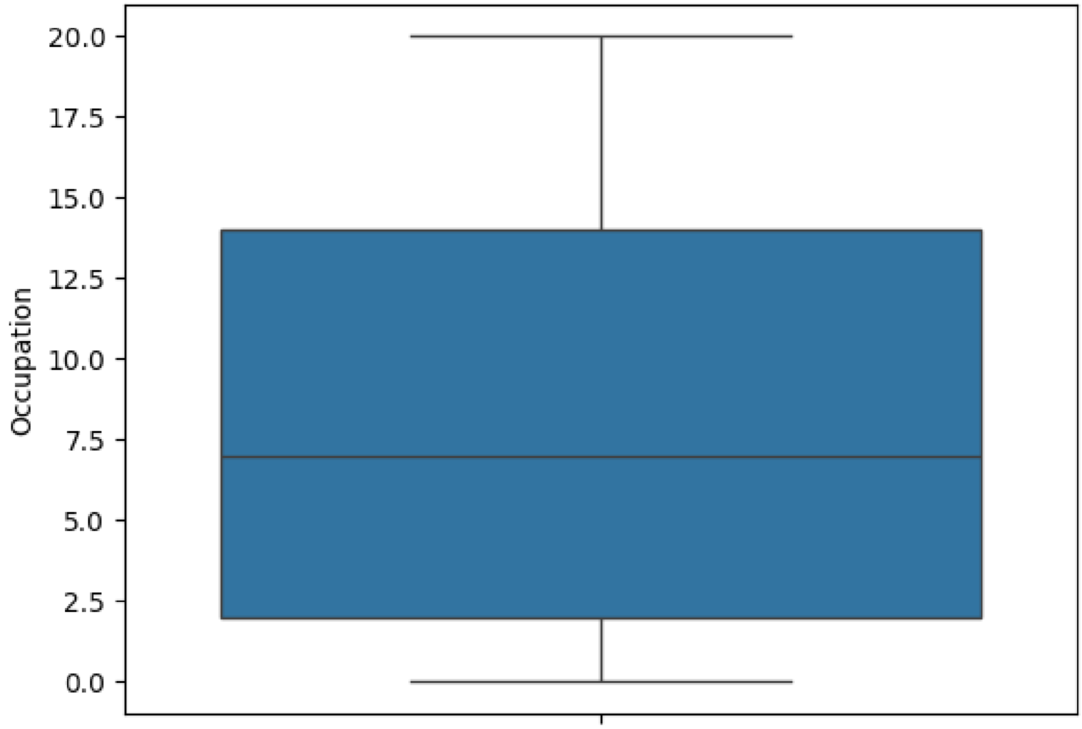
[25]:
```
sns.boxplot(df['Age'])
```

[25]: <Axes: ylabel='Age'>



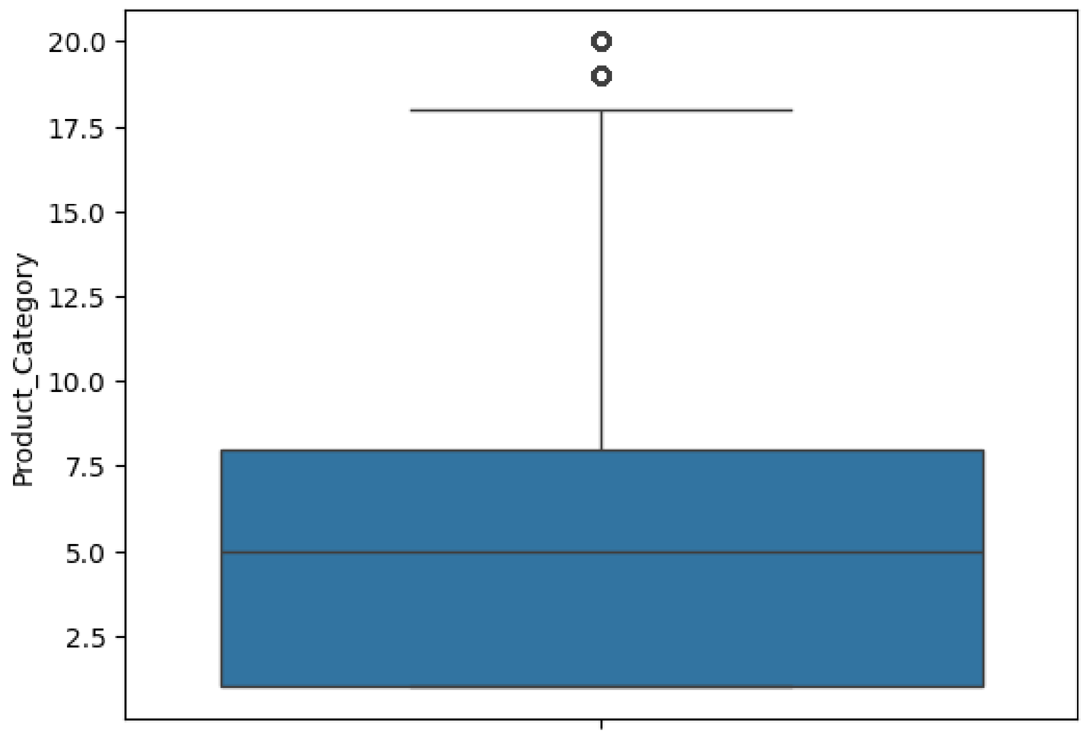
[26]:
```
sns.boxplot(df['Occupation'])
```

[26]: <Axes: ylabel='Occupation'>



[27]:
```
sns.boxplot(df['Product_Category'])
```

[27]: <Axes: ylabel='Product_Category'>



[28]:
```
#clipping the purchase data

df['remove_purchase'] = np.clip(df['Purchase'], np.
 ↪percentile(df['Purchase'],5), np.percentile(df['Purchase'],95))
df
```

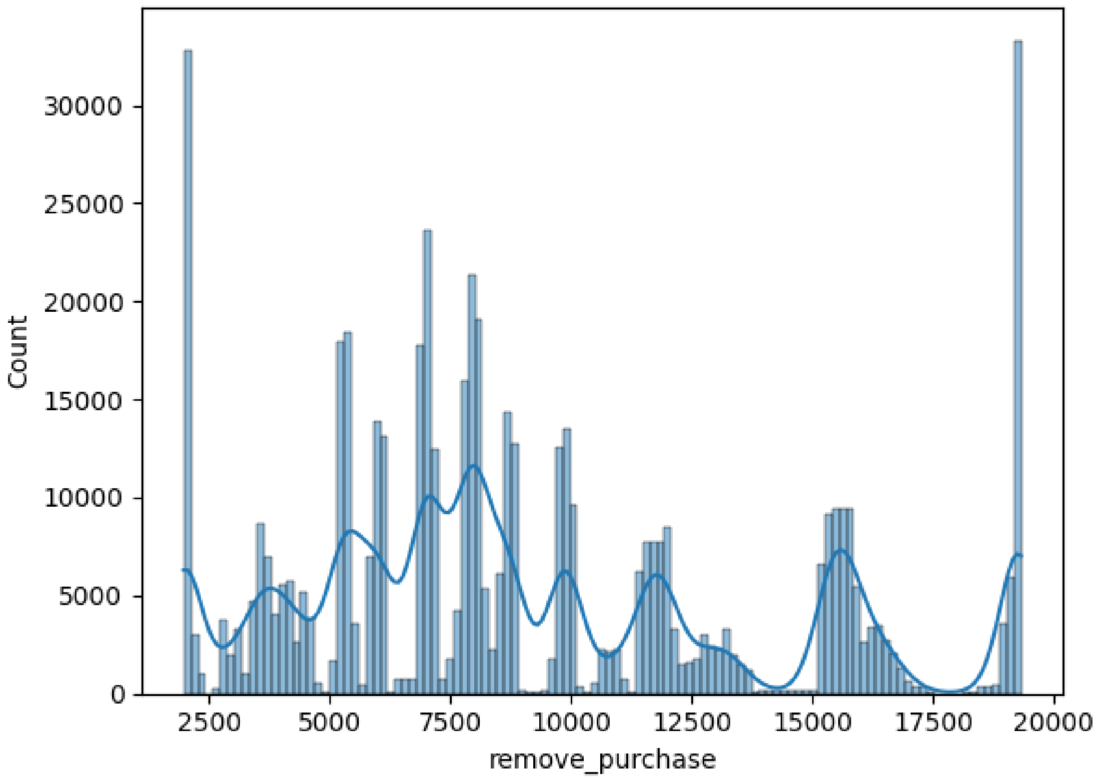
[28]:
```
        User_ID Product_ID Gender   Age Occupation City_Category  \
0       1000001  P00069042      F  0-17         10             A
1       1000001  P00248942      F  0-17         10             A
2       1000001  P00087842      F  0-17         10             A
3       1000001  P00085442      F  0-17         10             A
4       1000002  P00285442      M   55+         16             C
…           …         …   …     …     …          …
550063  1006033  P00372445      M 51-55         13             B
550064  1006035  P00375436      F 26-35          1             C
550065  1006036  P00375436      F 26-35         15             B
550066  1006038  P00375436      F   55+          1             C
550067  1006039  P00371644      F 46-50          0             B

       Stay_In_Current_City_Years Marital_Status Product_Category  Purchase  \
0                               2              0                3      8370
1                               2              0                1     15200
2                               2              0               12      1422
```

```
3                              2          0            12      1057
4                             4+          0             8      7969
…                             …          …           …       …
550063                         1          1            20       368
550064                         3          0            20       371
550065                        4+          1            20       137
550066                         2          0            20       365
550067                        4+          1            20       490

        remove_purchase
0                  8370
1                 15200
2                  1984
3                  1984
4                  7969
…                  …
550063             1984
550064             1984
550065             1984
550066             1984
550067             1984

[550068 rows x 11 columns]
```

2 Univariate Analysis

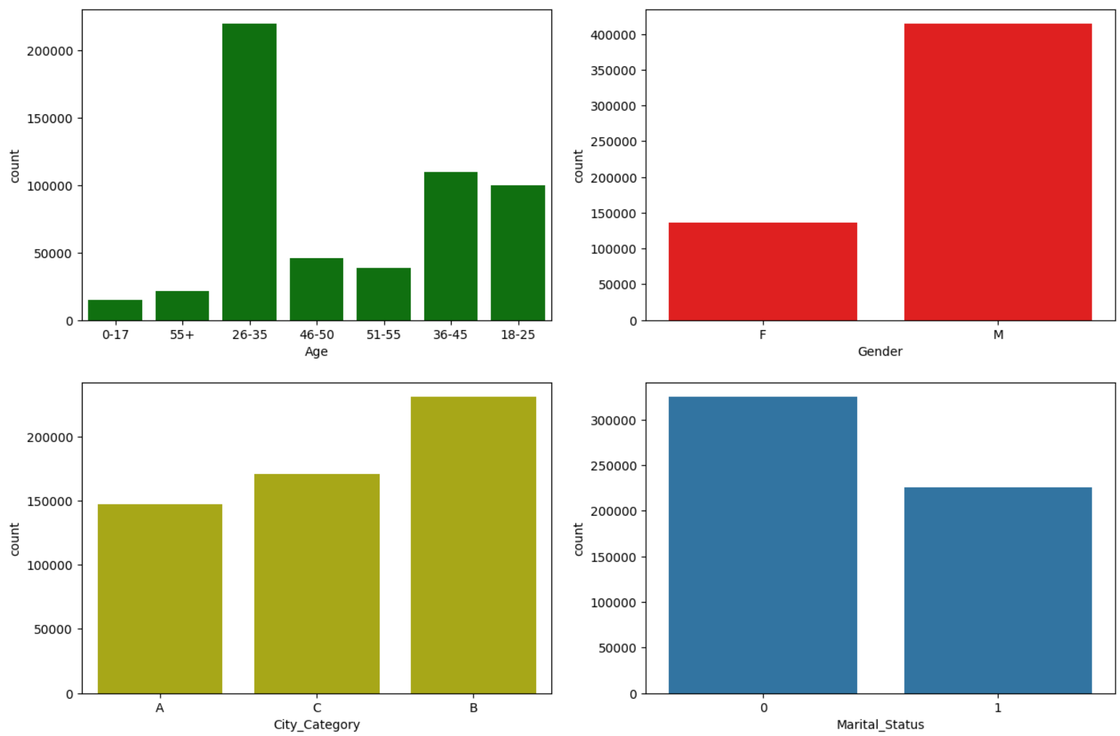
[29]:
```
sns.histplot(x = 'remove_purchase', data = df, kde = True)
```

[29]: <Axes: xlabel='remove_purchase', ylabel='Count'>



[120]:
```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))
sns.countplot(x='Age',data=df,ax=axs[0,0],color='g')
sns.countplot(x='Gender',data=df,ax=axs[0,1],color='r')
sns.countplot(x='City_Category',data=df,ax=axs[1,0],color='y')
sns.countplot(x='Marital_Status',data=df,ax=axs[1,1])
```

[120]: <Axes: xlabel='Marital_Status', ylabel='count'>

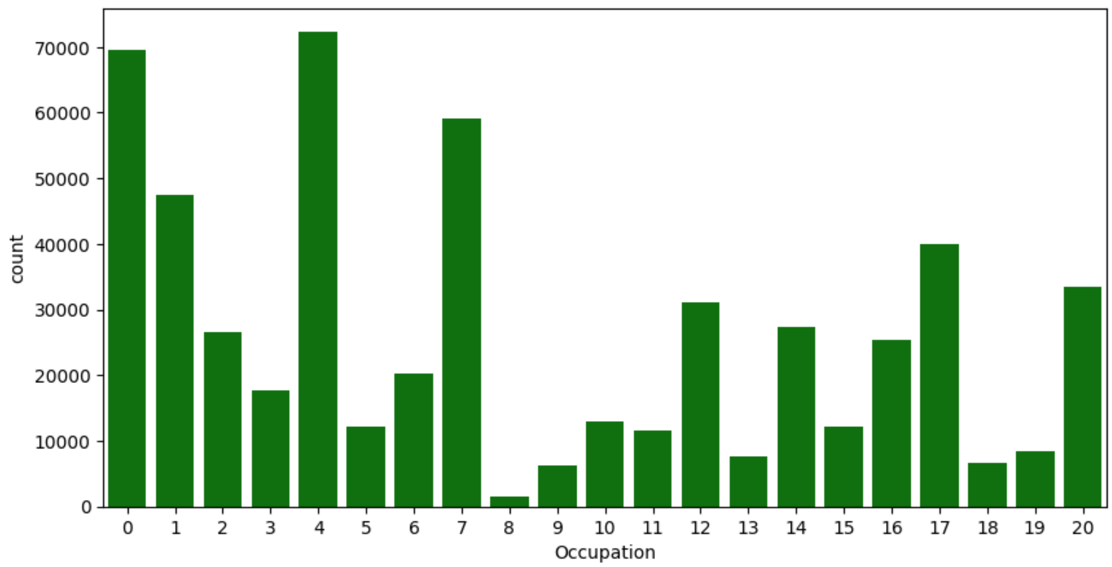


Insights:

- We can clearly see from the graphs above the purchases done by males are much higher than females.
- People of age 26-35 are the target customers.
- The purchases are highest from City category B.
- Single customer purchases are higher than married users.

[32]:
```
plt.figure(figsize=(10,5))
sns.countplot(x='Occupation',data=df,color='g')
```

[32]: <Axes: xlabel='Occupation', ylabel='count'>

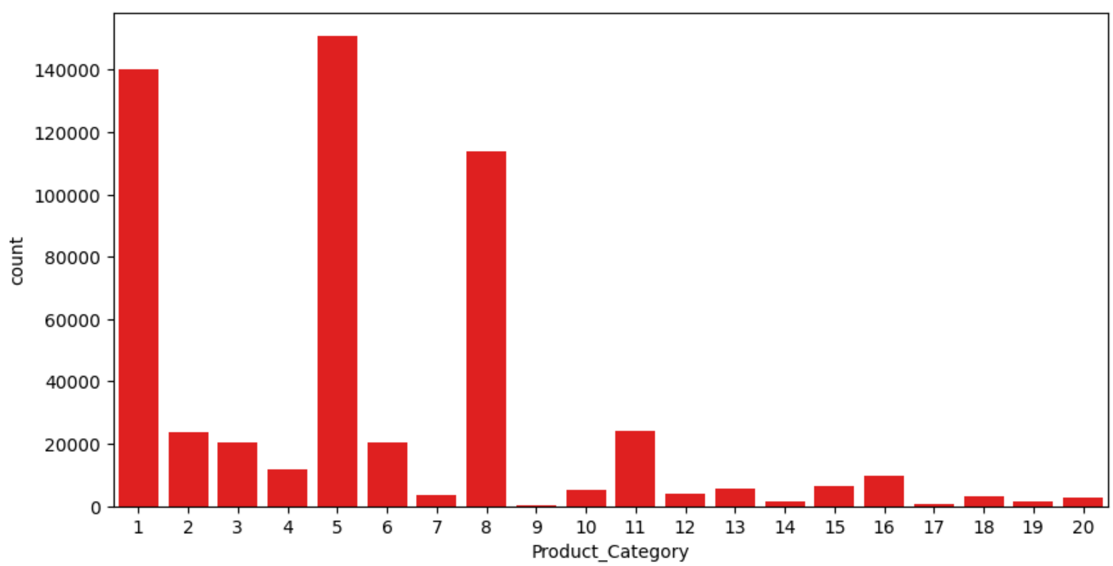


Insight:

- We have 21 occupations categories. Occupation category 4, 0, and 7 are with higher number of purchases and category 8 with the lowest number of purchaes.

[34]:
```
plt.figure(figsize=(10,5))
sns.countplot(x='Product_Category',data=df,color='r')
```

[34]: <Axes: xlabel='Product_Category', ylabel='count'>



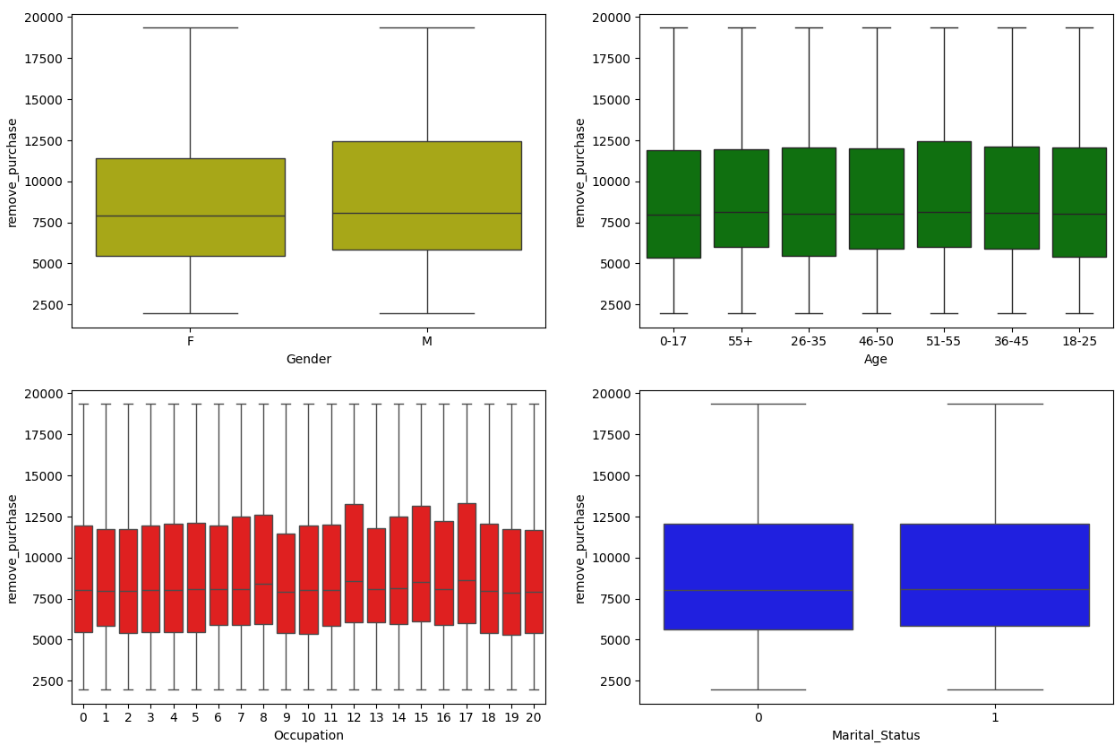
Insight:

- There are 20 product categories with product category 1, 5 and 8 having higher purchasing frequency.

3 Bivariate Analysis

[41]:
```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))
sns.boxplot(x = 'Gender', y = 'remove_purchase', data = df, ax = axs[0,0],
 ↪color= 'y')
sns.boxplot(x = 'Age', y = 'remove_purchase', data = df, ax = axs[0,1], color=
 ↪'g')
sns.boxplot(x = 'Occupation', y = 'remove_purchase', data = df, ax = axs[1,0],
 ↪color= 'r')
sns.boxplot(x = 'Marital_Status', y = 'remove_purchase', data = df, ax =
 ↪axs[1,1], color= 'b')
```

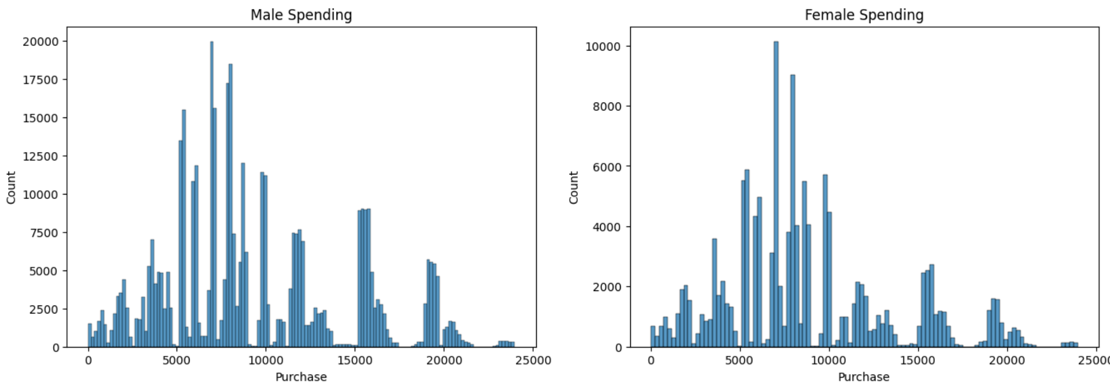
[41]:
```
<Axes: xlabel='Marital_Status', ylabel='remove_purchase'>
```



[42]:
```
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(16, 5))
sns.histplot(data = df[df['Gender'] == 'M']['Purchase'], ax = axs[0]).
 ↪set_title('Male Spending')
```

```
sns.histplot(data = df[df['Gender'] == 'F']['Purchase'], ax = axs[1]).set_title('Female Spending')
```

[42]: Text(0.5, 1.0, 'Female Spending')

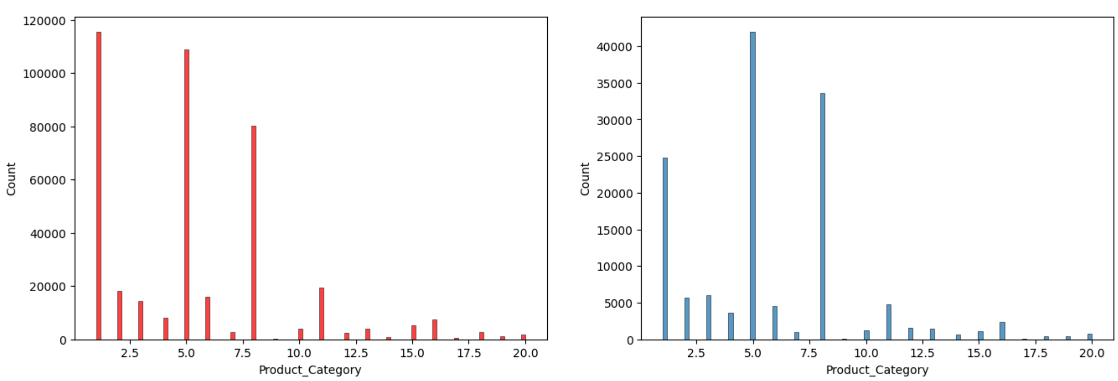


Insight:

- From the above histplot, we can clearly see spending behaviour is very much similar in nature for both males and females as the maximum purchase count are between the purchase value range of 5000 -10000 for both. But, the purchase count are more in case of males.

[43]:
```
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(16, 5))
sns.histplot(data = df[df['Gender'] == 'M']['Product_Category'], ax = axs[0], color = 'r')
sns.histplot(data = df[df['Gender'] == 'F']['Product_Category'], ax = axs[1])
```

[43]: <Axes: xlabel='Product_Category', ylabel='Count'>



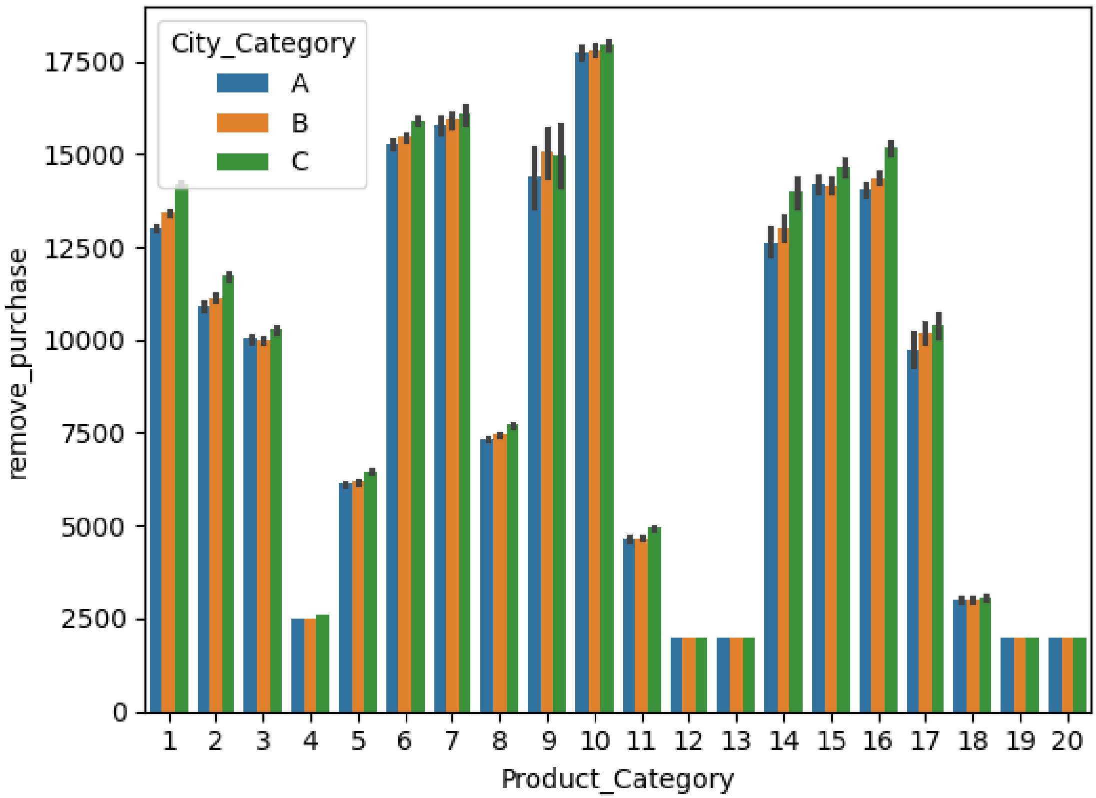
Insight:

- The product catergory purchased more by male and female by 5.0.

4 Multivariate Analysis

[45]:
```
sns.barplot(x = 'Product_Category', y = 'remove_purchase', hue =␣
↪'City_Category', data = df)
```

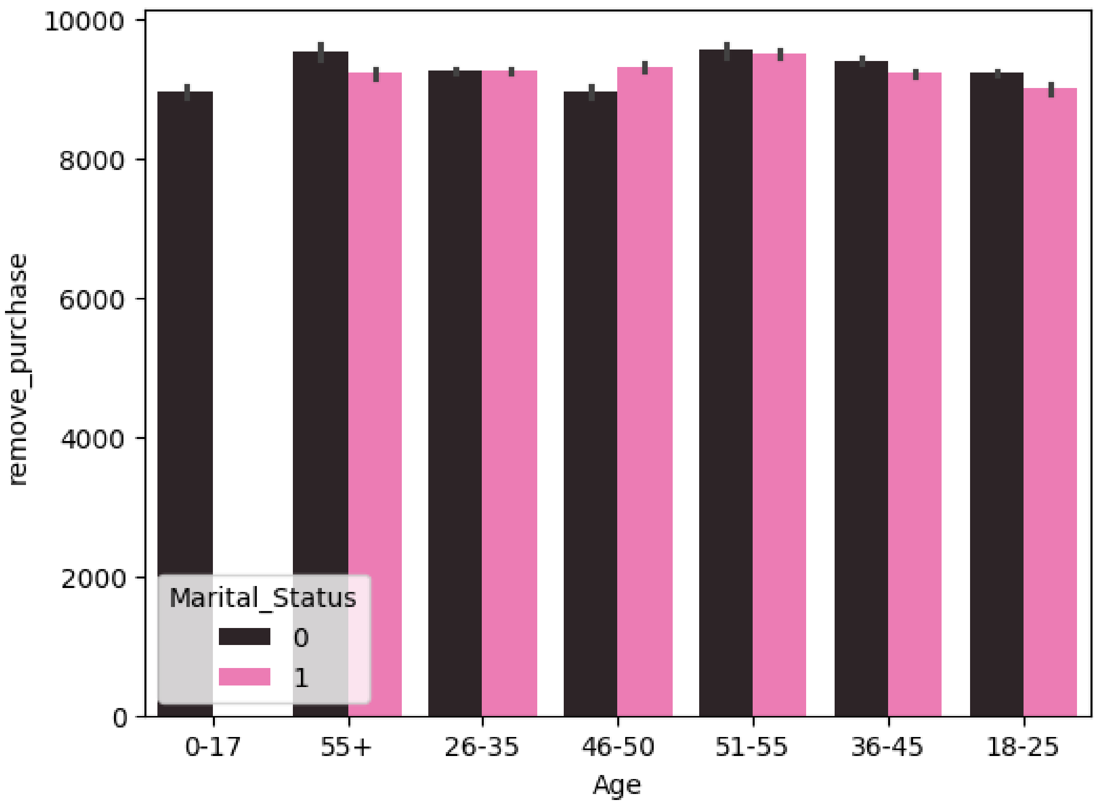
[45]: <Axes: xlabel='Product_Category', ylabel='remove_purchase'>



[47]:
```
sns.barplot(x = 'Age', y = 'remove_purchase', hue = 'Marital_Status', data =␣
↪df, color = 'hotpink')
```

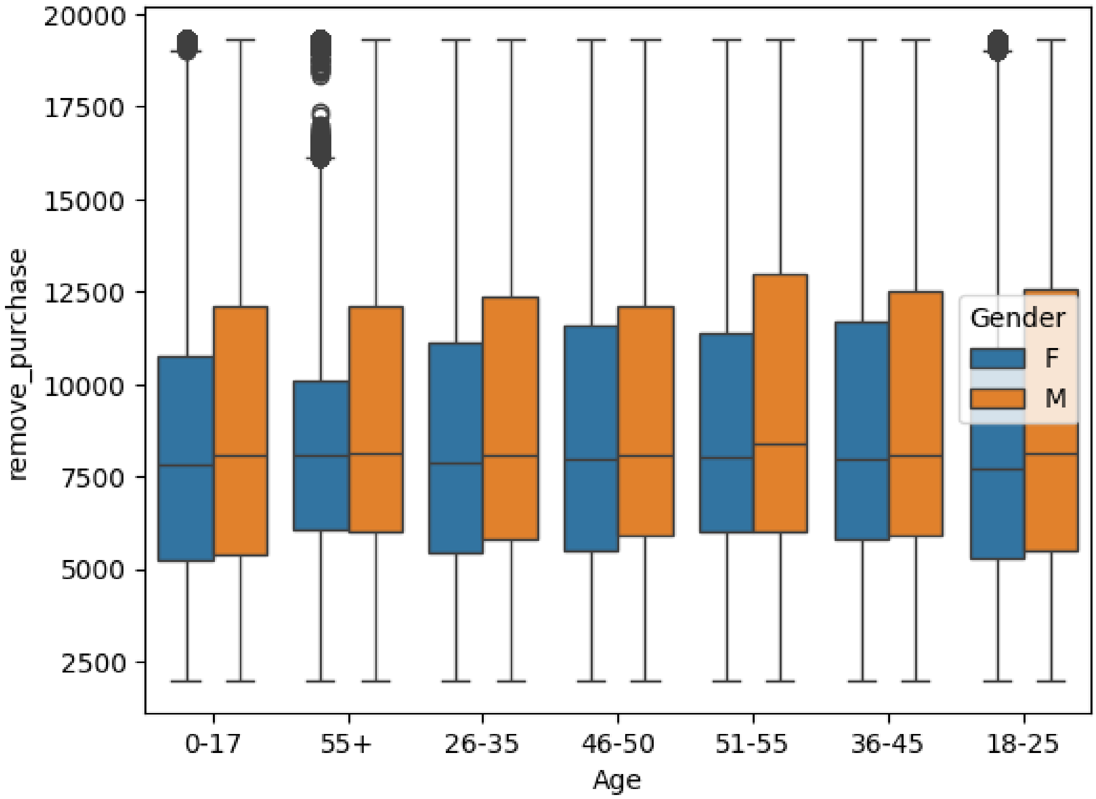
```
<ipython-input-47-ee5babe841f1>:1: FutureWarning:

Setting a gradient palette using color= is deprecated and will be removed in
v0.14.0. Set `palette='dark:hotpink'` for the same effect.

  sns.barplot(x = 'Age', y = 'remove_purchase', hue = 'Marital_Status', data =
df, color = 'hotpink')
```

[47]: <Axes: xlabel='Age', ylabel='remove_purchase'>




```
[48]: sns.boxplot(x = 'Age', y = 'remove_purchase', hue = 'Gender', data = df)
```

```
[48]: <Axes: xlabel='Age', ylabel='remove_purchase'>
```



5 Average amount spent per gender

[52]:
```
avg_spend = df.groupby(['User_ID','Gender'])['Purchase'].mean()
avg_spend.reset_index()
```

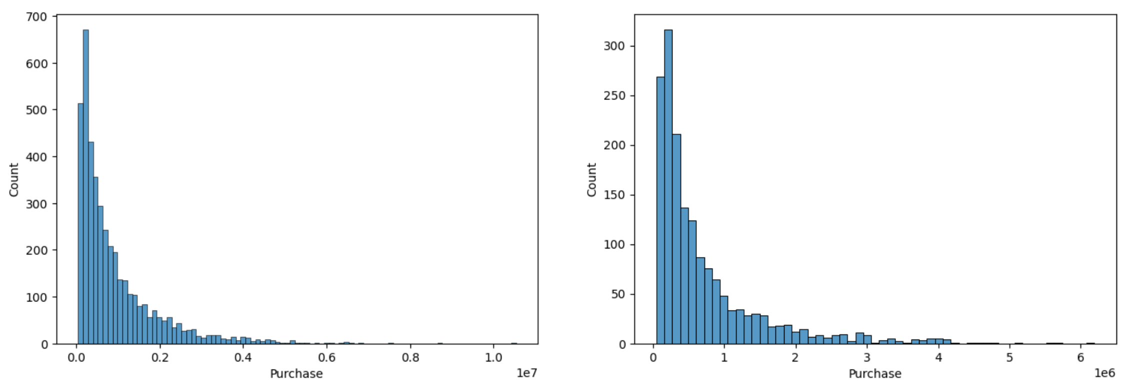
[52]:
```
      User_ID Gender      Purchase
0     1000001      F   9545.514286
1     1000002      M  10525.610390
2     1000003      M  11780.517241
3     1000004      M  14747.714286
4     1000005      M   7745.292453
…         …     …            …
5886  1006036      F   8007.894942
5887  1006037      F   9176.540984
5888  1006038      F   7502.833333
5889  1006039      F   7977.283784
5890  1006040      M   9184.994444

[5891 rows x 3 columns]
```

[61]:
```
avgamt_gender = df.groupby(['User_ID','Gender'])['Purchase'].sum()
avgamt_gender = avgamt_gender.reset_index()
avgamt_gender
```

[61]:
```
      User_ID Gender  Purchase
0     1000001      F    334093
1     1000002      M    810472
2     1000003      M    341635
3     1000004      M    206468
4     1000005      M    821001
…         …     …         …
5886  1006036      F   4116058
5887  1006037      F   1119538
5888  1006038      F     90034
5889  1006039      F    590319
5890  1006040      M   1653299

[5891 rows x 3 columns]
```

[62]:
```
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(16, 5))
sns.histplot(data = avgamt_gender[avgamt_gender['Gender'] == 'M']['Purchase'],␣
 ↪ax = axs[0])
sns.histplot(data = avgamt_gender[avgamt_gender['Gender'] == 'F']['Purchase'],␣
 ↪ax = axs[1])
```

[62]: <Axes: xlabel='Purchase', ylabel='Count'>



Insight:

- Average amount spend by males are higher than females.

[66]:
```
avgamt_gender.groupby(['Gender'])[['Purchase']].mean()
```

[66]:
```
            Purchase
Gender
F      712024.394958
M      925344.402367
```

[67]:
```python
avgamt_gender.groupby(['Gender'])[['Purchase']].sum()
```

[67]:
```
         Purchase
Gender
F      1186232642
M      3909580100
```

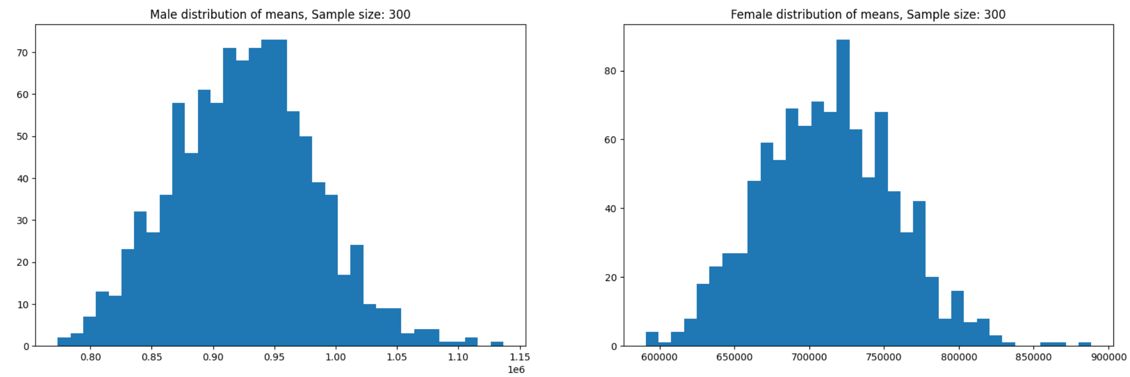
Insight:

- The average purchase for male is greater than female.

[68]:
```python
avgamt_male = avgamt_gender[avgamt_gender['Gender'] == 'M']
avgamt_female = avgamt_gender[avgamt_gender['Gender'] == 'F']
```

[71]:
```python
##Finding the sample(sample size=300) for avg purchase amount for males and␣
↪females

genders = ['M', 'F']
sample_size = 300
num_repitions = 1000

male_means = []
female_means = []

for i in range(num_repitions):
    males_sample = avgamt_male.sample(sample_size, replace = True)['Purchase'].
↪mean()
    females_sample = avgamt_female.sample(sample_size, replace =␣
↪True)['Purchase'].mean()
    male_means.append(males_sample)
    female_means.append(females_sample)
```

[75]:
```python
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))
axis[0].hist(male_means, bins = 35)
axis[1].hist(female_means, bins = 35)
axis[0].set_title("Male distribution of means, Sample size: 300")
axis[1].set_title("Female distribution of means, Sample size: 300")

plt.show()
```



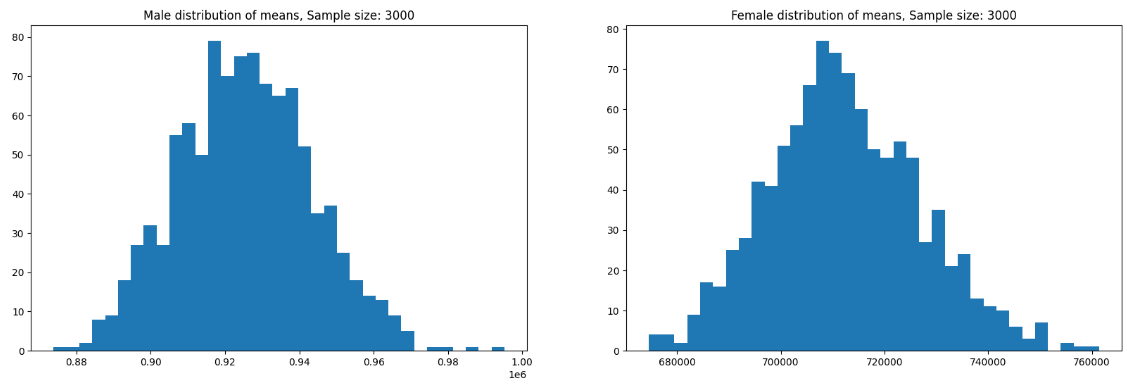
[76]:
```
z95=1.960
#95% Confidence Interval

sample_mean_male = np.mean(male_means)
sample_mean_female = np.mean(female_means)

sample_std_male=pd.Series(male_means).std()
sample_std_female=pd.Series(female_means).std()

sample_std_error_male=sample_std_male/np.sqrt(300)
sample_std_error_female=sample_std_female/np.sqrt(300)


Upper_Limit_male=z95*sample_std_error_male + sample_mean_male
Lower_Limit_male=sample_mean_male - z95*sample_std_error_male

Upper_Limit_female=z95*sample_std_error_female + sample_mean_female
Lower_Limit_female=sample_mean_female - z95*sample_std_error_female

male_mean_confidence_interval = (Lower_Limit_male, Upper_Limit_male)
female_mean_confidence_interval = (Lower_Limit_female, Upper_Limit_female)

print(f"Male mean confidence interval: {male_mean_confidence_interval}")
print(f"Female mean confidence interval: {female_mean_confidence_interval}")
```

```
Male mean confidence interval: (919210.7021406022, 932265.4160260643)
Female mean confidence interval: (707579.6571021017, 717704.2482512316)
```

[77]:
```
##Finding the sample(sample size=3000) for avg purchase amount for males and␣
↪females

genders = ["M", "F"]
sample_size = 3000
```

```
num_repitions = 1000

male_means = []
female_means = []

for i in range(num_repitions):
    male_mean = avgamt_male.sample(sample_size, replace=True)['Purchase'].mean()
    female_mean = avgamt_female.sample(sample_size, replace=True)['Purchase'].
 ↪mean()

    male_means.append(male_mean)
    female_means.append(female_mean)
```

[79]:
```
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))
axis[0].hist(male_means, bins=35)
axis[1].hist(female_means, bins=35)
axis[0].set_title("Male distribution of means, Sample size: 3000")
axis[1].set_title("Female distribution of means, Sample size: 3000")

plt.show()
```



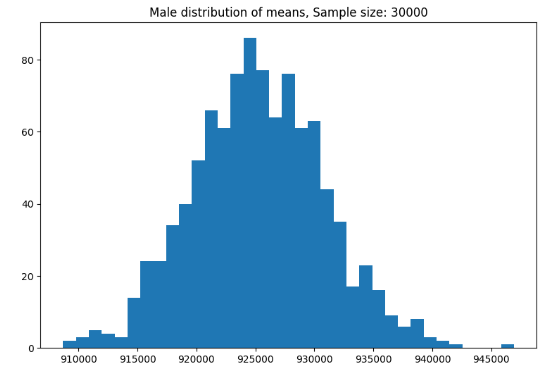
[80]:
```
z95=1.960
#95% Confidence Interval

sample_mean_male = np.mean(male_means)
sample_mean_female = np.mean(female_means)

sample_std_male=pd.Series(male_means).std()
sample_std_female=pd.Series(female_means).std()

sample_std_error_male=sample_std_male/np.sqrt(3000)
sample_std_error_female=sample_std_female/np.sqrt(3000)
```

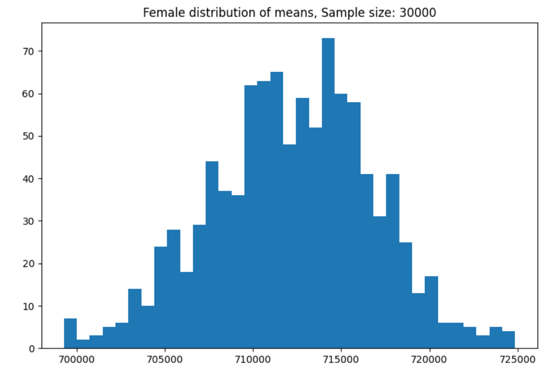
```
Upper_Limit_male=z95*sample_std_error_male + sample_mean_male
Lower_Limit_male=sample_mean_male - z95*sample_std_error_male

Upper_Limit_female=z95*sample_std_error_female + sample_mean_female
Lower_Limit_female=sample_mean_female - z95*sample_std_error_female

male_mean_confidence_interval = (Lower_Limit_male, Upper_Limit_male)
female_mean_confidence_interval = (Lower_Limit_female, Upper_Limit_female)

print(f"Male mean confidence interval: {male_mean_confidence_interval}")
print(f"Female mean confidence interval: {female_mean_confidence_interval}")
```

```
Male mean confidence interval: (924939.9812161535, 926228.01209718)
Female mean confidence interval: (711836.5439900327, 712881.1520279672)
```

[82]:
```
##Finding the sample(sample size=30000) for avg purchase amount for males and␣
↪females


genders = ["M", "F"]
sample_size = 30000
num_repitions = 1000

male_means = []
female_means = []

for i in range(num_repitions):
    male_mean = avgamt_male.sample(sample_size, replace=True)['Purchase'].mean()
    female_mean = avgamt_female.sample(sample_size, replace=True)['Purchase'].
↪mean()

    male_means.append(male_mean)
    female_means.append(female_mean)
```

[84]:
```
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))
axis[0].hist(male_means, bins=35)
axis[1].hist(female_means, bins=35)
axis[0].set_title("Male distribution of means, Sample size: 30000")
axis[1].set_title("Female distribution of means, Sample size: 30000")

plt.show()
```





[85]:
```
z95 = 1.960
# 95% Confidence Interval

sample_male_mean = np.mean(male_means)
sample_female_mean = np.mean(female_means)

sample_male_std = pd.Series(male_means).std()
sample_female_std = pd.Series(female_means).std()

sample_male_std_error = sample_male_std / np.sqrt(30000)
sample_female_std_error = sample_female_std / np.sqrt(30000)

Upper_Limit_male = z95 * sample_male_std_error + sample_male_mean
Lower_Limit_male = sample_male_mean - z95 * sample_male_std_error

Upper_Limit_female = z95 * sample_female_std_error + sample_female_mean
Lower_Limit_female = sample_female_mean - z95 * sample_female_std_error

male_mean_confidence_interval = (Lower_Limit_male, Upper_Limit_male)
female_mean_confidence_interval = (Lower_Limit_female, Upper_Limit_female)

print(f"Male mean confidence interval: {male_mean_confidence_interval}")
print(f"Female mean confidence interval: {female_mean_confidence_interval}")
```

```
Male mean confidence interval: (925204.438070709, 925331.0980862911)
Female mean confidence interval: (712187.4227626218, 712291.7653653115)
```

Insights:

- By increasing the sample size we can see confidence interval is more closer to the population mean.
- Sample size affect the shape of the distributions of the mean, as the deviation becomes lesser.

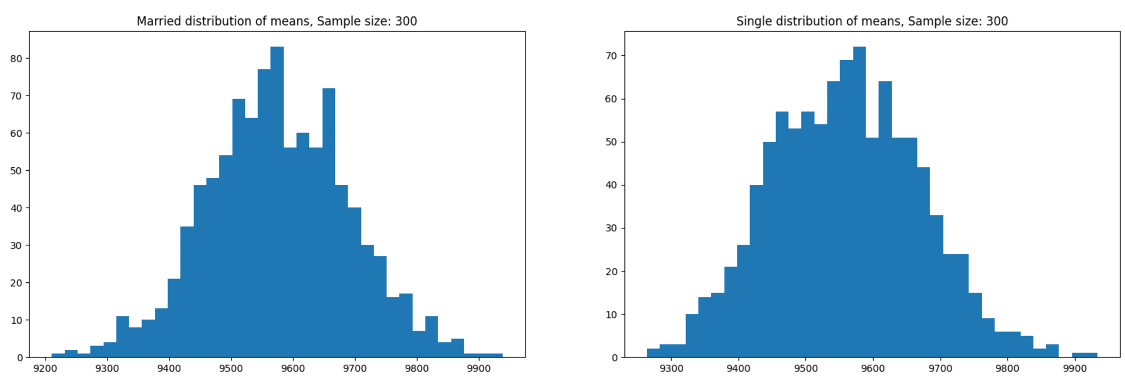
6 How does Marital_Status affect the amount spent?

[86]:

```
avg_marital = df.groupby(['User_ID','Marital_Status'])['Purchase'].mean()
avg_marital = avg_marital.reset_index()
avg_marital

avg_single = avg_marital[avg_marital['Marital_Status'] == 0]
avg_married = avg_marital[avg_marital['Marital_Status'] == 1]

sample_size = 300
num_repitions = 1000
married_means = []
single_means = []

for i in range(num_repitions):
    avg_married = avg_marital[avg_marital['Marital_Status'] == 1].
 ↪sample(sample_size, replace = True)['Purchase'].mean()
    avg_single = avg_marital[avg_marital['Marital_Status'] == 0].
 ↪sample(sample_size, replace = True)['Purchase'].mean()

    married_means.append(avg_married)
    single_means.append(avg_single)
```

[87]:

```
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))
axis[0].hist(married_means, bins = 35)
axis[1].hist(single_means, bins = 35)
axis[0].set_title("Married distribution of means, Sample size: 300")
axis[1].set_title("Single distribution of means, Sample size: 300")

plt.show()
```



[91]:

```
z95=1.960
#95% Confidence Interval
```

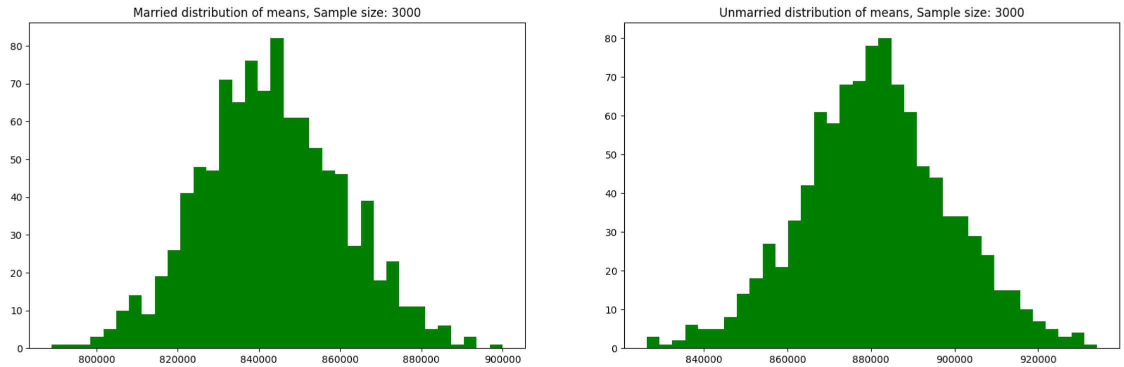
```
sample_mean_married = np.mean(married_means)
sample_mean_single = np.mean(single_means)

sample_std_married=pd.Series(married_means).std()
sample_std_single=pd.Series(single_means).std()

sample_std_error_married=sample_std_married/np.sqrt(300)
sample_std_error_single=sample_std_single/np.sqrt(300)


Upper_Limit_married=z95*sample_std_error_married + sample_mean_married
Lower_Limit_maried=sample_std_error_married - z95*sample_mean_married

Upper_Limit_single=z95*sample_std_error_single + sample_mean_single
Lower_Limit_single=sample_mean_single-z95*sample_std_error_single

married_mean_confidence_interval = (Upper_Limit_married,Lower_Limit_maried)
single_mean_confidence_interval = (Upper_Limit_single,Lower_Limit_single)

print(f"Married mean confidence interval: {married_mean_confidence_interval}")
print(f"Single mean confidence interval: {single_mean_confidence_interval}")
```

```
Married mean confidence interval: (845175.2522226233, -1651674.248839387)
Single mean confidence interval: (883286.2020662723, 879268.4414457278)
```

[89]:
```
avg_Marital = df.groupby(['User_ID', 'Marital_Status'])[['Purchase']].sum()
avg_Marital = avg_Marital.reset_index()
avg_Marital

avg_Single = avg_Marital[avg_Marital['Marital_Status']==0]
avg_Married = avg_Marital[avg_Marital['Marital_Status']==1]

sample_size = 3000
num_repitions = 1000
married_means = []
single_means = []


for i in range(num_repitions):
    avg_married = avg_Marital[avg_Marital['Marital_Status']==1].
 ↪sample(sample_size, replace=True)['Purchase'].mean()
    avg_single = avg_Marital[avg_Marital['Marital_Status']==0].
 ↪sample(sample_size, replace=True)['Purchase'].mean()

    married_means.append(avg_married)
    single_means.append(avg_single)
```

[92]:

```
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))

axis[0].hist(married_means, bins=35,color='g')
axis[1].hist(single_means, bins=35,color='g')
axis[0].set_title("Married distribution of means, Sample size: 3000")
axis[1].set_title("Unmarried distribution of means, Sample size: 3000")

plt.show()
```



[93]:

```
z95=1.960
#95% Confidence Interval

sample_mean_married = np.mean(married_means)
sample_mean_single = np.mean(single_means)

sample_std_married=pd.Series(married_means).std()
sample_std_single=pd.Series(single_means).std()

sample_std_error_married=sample_std_married/np.sqrt(3000)
sample_std_error_single=sample_std_single/np.sqrt(3000)


Upper_Limit_married=z95*sample_std_error_married + sample_mean_married
Lower_Limit_maried=sample_std_error_married - z95*sample_mean_married

Upper_Limit_single=z95*sample_std_error_single + sample_mean_single
Lower_Limit_single=sample_std_error_single - z95*sample_mean_single

married_mean_confidence_interval = (Upper_Limit_married,Lower_Limit_maried)
single_mean_confidence_interval = (Upper_Limit_single,Lower_Limit_single)

print(f"Married confidence interval: {married_mean_confidence_interval}")
print(f"Single confidence interval: {single_mean_confidence_interval}")
```

```
Married confidence interval: (843827.4061439544, -1652361.9254101363)
Single confidence interval: (881912.5854887126, -1726979.4364924168)
```

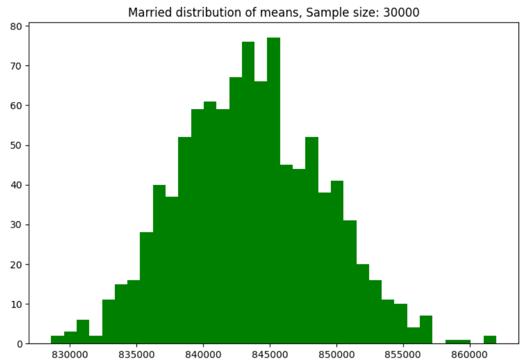
CI's of male and female do not overlap and upper limits of female purchase CI are lesser than lower limits of male purchase CI. This proves that men usually spend more than women (NOTE: as per data 77% contibutions are from men and only 23% purchases are from women).

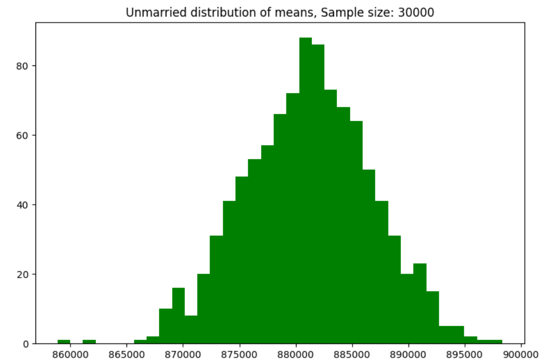
The reason for less purchase by women could have several factors:

Males might be doing the purchase for females. Salary can be a factor in less purchase. We also need to see whether male-based products were sold more than women-based products to clearly identify difference in spending pattern. If the female based products quality/quantity needs to be improved for women purchasing.

[94]:
```
avg_Marital = df.groupby(['User_ID', 'Marital_Status'])[['Purchase']].sum()
avg_Marital = avg_Marital.reset_index()
avg_Marital

avg_Single = avg_Marital[avg_Marital['Marital_Status']==0]
avg_Married = avg_Marital[avg_Marital['Marital_Status']==1]

sample_size = 30000
num_repitions = 1000
married_means = []
single_means = []


for i in range(num_repitions):
    avg_married = avg_Marital[avg_Marital['Marital_Status']==1].
 ↪sample(sample_size, replace=True)['Purchase'].mean()
    avg_single = avg_Marital[avg_Marital['Marital_Status']==0].
 ↪sample(sample_size, replace=True)['Purchase'].mean()

    married_means.append(avg_married)
    single_means.append(avg_single)
```

[121]:
```
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))

axis[0].hist(married_means, bins=35,color='g')
axis[1].hist(single_means, bins=35,color='g')
axis[0].set_title("Married distribution of means, Sample size: 30000")
axis[1].set_title("Unmarried distribution of means, Sample size: 30000")

plt.show()
```





Insight:

- The means sample seems to be normally distributed for both married and singles. Also, we can see the mean of the sample means are closer to the population mean as per central limit theorem.

[96]:
```
z95=1.960
#95% Confidence Interval

sample_mean_married = np.mean(married_means)
sample_mean_single = np.mean(single_means)

sample_std_married=pd.Series(married_means).std()
sample_std_single=pd.Series(single_means).std()

sample_std_error_married=sample_std_married/np.sqrt(30000)
sample_std_error_single=sample_std_single/np.sqrt(30000)


Upper_Limit_married=z95*sample_std_error_married + sample_mean_married
Lower_Limit_maried=sample_std_error_married - z95*sample_mean_married

Upper_Limit_single=z95*sample_std_error_single + sample_mean_single
Lower_Limit_single=sample_std_error_single - z95*sample_mean_single

married_mean_confidence_interval = (Upper_Limit_married,Lower_Limit_maried)
single_mean_confidence_interval = (Upper_Limit_single,Lower_Limit_single)

print(f"Married mean confidence interval: {married_mean_confidence_interval}")
print(f"Single mean confidence interval: {single_mean_confidence_interval}")
```

```
Married mean confidence interval: (843622.0992677433, -1653348.239324587)
Single mean confidence interval: (881191.7282535823, -1726978.893278055)
```

Insights:

The average spending of married and unmarried do not overlap as unmarried tend to spend more than married.

Reasons related for it could be * Being married comes with more responsibilties so they tend to save more, and spend less on the purchases of non-necessity items. * Products related to family and kids could be added on for longer term, which would be necessary for a family person.

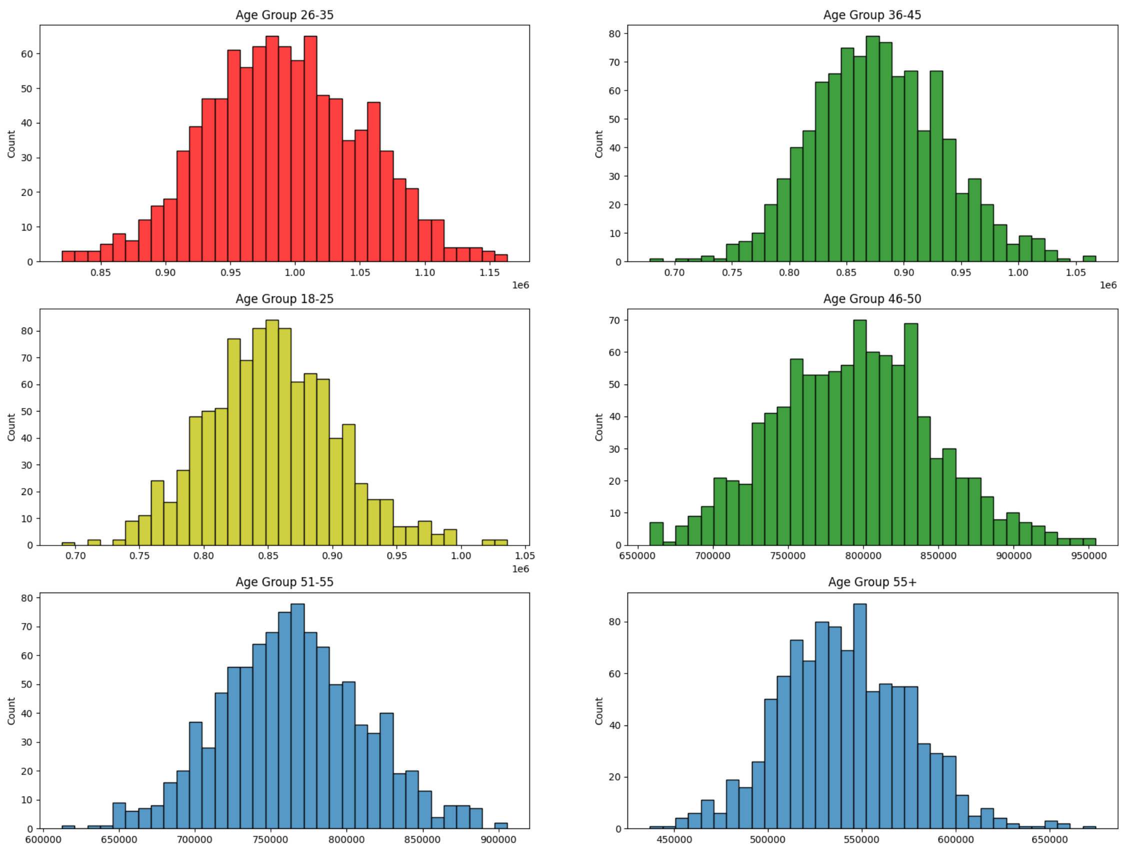
[100]:
```
avgamt_age = df.groupby(['User_ID', 'Age'])[['Purchase']].sum()
avgamt_age = avgamt_age.reset_index()
avgamt_age['Age'].value_counts()
```

[100]:
```
Age
26-35    2053
36-45    1167
18-25    1069
46-50     531
51-55     481
55+       372
0-17      218
Name: count, dtype: int64
```

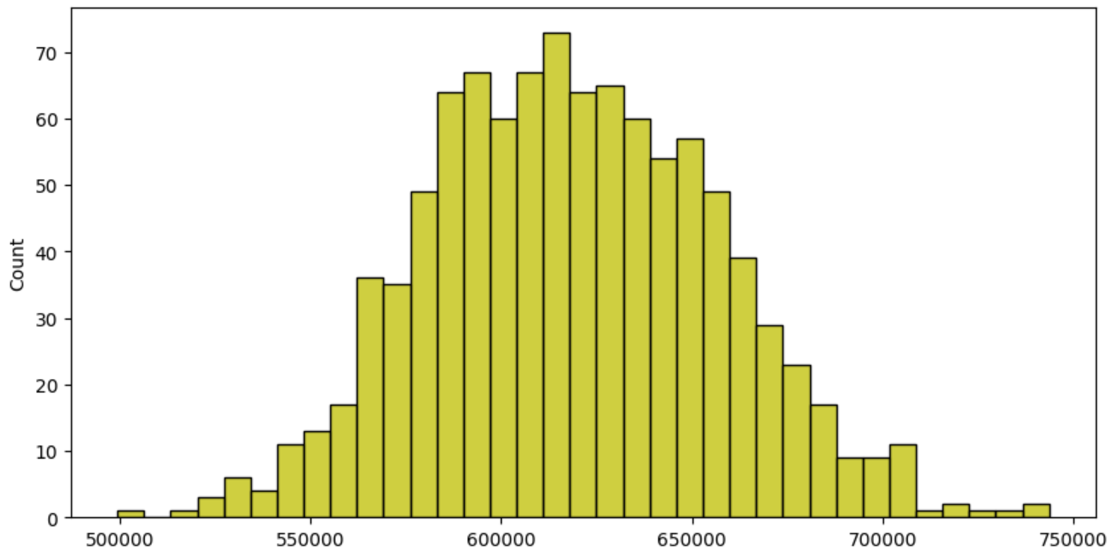
7 How does Age affect the amount spent?

[122]:
```
sample_size = 300
num_repetition = 1000

all_sample_means = {}

age_intervals = ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']
for i in age_intervals:
  all_sample_means[i] = []

for i in age_intervals:
  for j in range(num_repetition):
    mean = avgamt_age[avgamt_age['Age'] == i].sample(sample_size, replace =␣
↪True)['Purchase'].mean()
    all_sample_means[i].append(mean)


fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(20, 15))

sns.histplot(data = all_sample_means['26-35'], bins = 35, ax = axis[0, 0],␣
↪color = 'r').set_title('Age Group 26-35')
sns.histplot(data = all_sample_means['36-45'], bins = 35, ax = axis[0, 1],␣
↪color = 'g').set_title('Age Group 36-45')
sns.histplot(data = all_sample_means['18-25'], bins = 35, ax = axis[1, 0],␣
↪color = 'y').set_title('Age Group 18-25')
```

```python
sns.histplot(data = all_sample_means['46-50'], bins = 35, ax = axis[1, 1], color = 'g').set_title('Age Group 46-50')
sns.histplot(data = all_sample_means['51-55'], bins = 35, ax = axis[2, 0]).set_title('Age Group 51-55')
sns.histplot(data = all_sample_means['55+'], bins = 35, ax = axis[2, 1]).set_title('Age Group 55+')

plt.show()
```

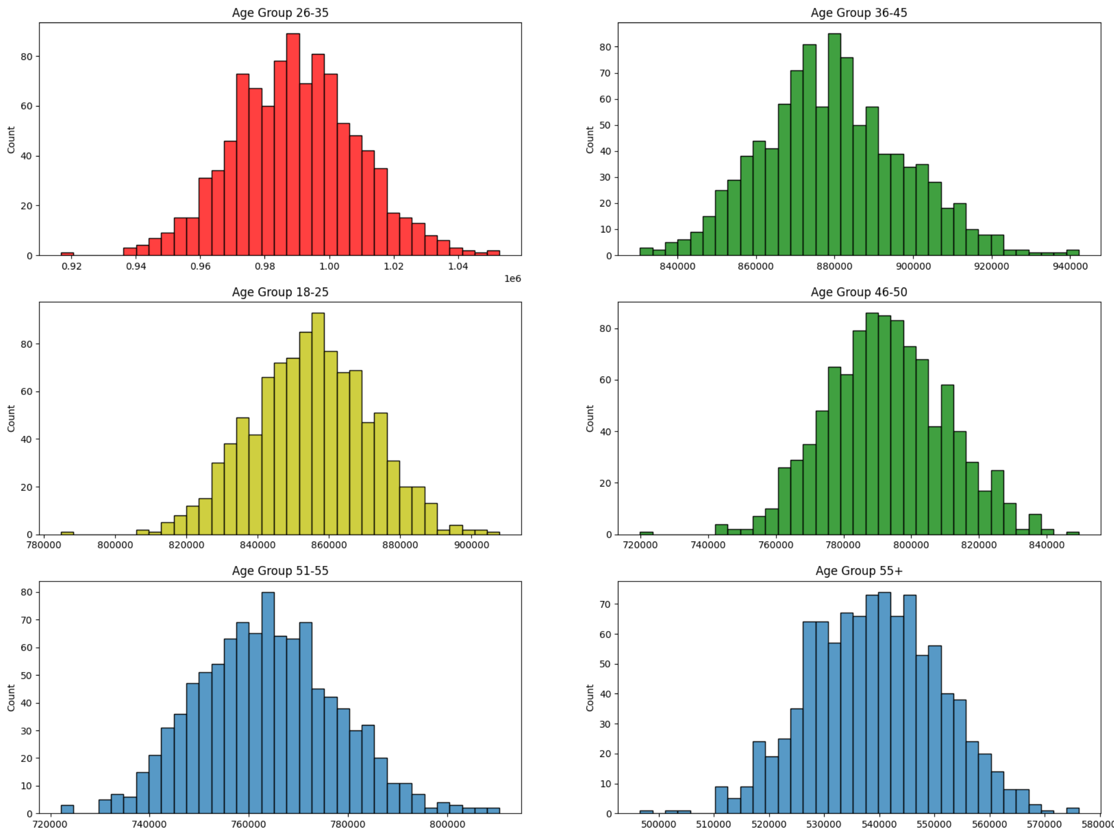


[124]:
```python
plt.figure(figsize=(10, 5))
sns.histplot(all_sample_means['0-17'], bins = 35, color = 'y')
plt.show()
```

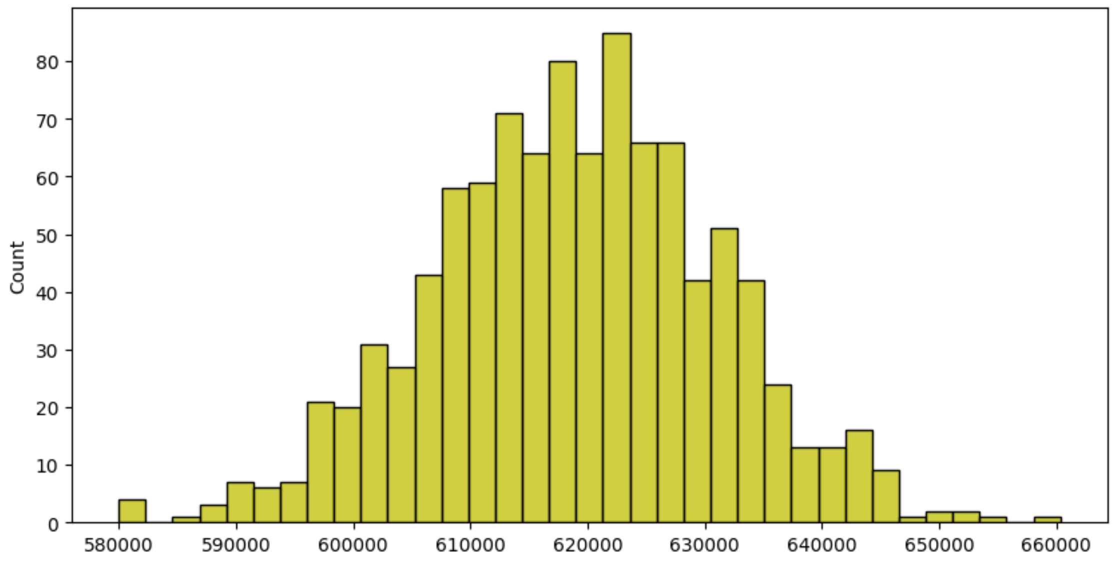


[125]:
```
sample_size = 3000
num_repitions = 1000

all_sample_means = {}

age_intervals = ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']
for i in age_intervals:
    all_sample_means[i] = []

for i in age_intervals:
    for j in range(num_repitions):

        mean = avgamt_age[avgamt_age['Age']==i].sample(sample_size,␣
 ↪replace=True)['Purchase'].mean()
        all_sample_means[i].append(mean)



fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(20, 15))

sns.histplot(data = all_sample_means['26-35'], bins = 35, ax = axis[0, 0],␣
 ↪color = 'r').set_title('Age Group 26-35')
sns.histplot(data = all_sample_means['36-45'], bins = 35, ax = axis[0, 1],␣
 ↪color = 'g').set_title('Age Group 36-45')
sns.histplot(data = all_sample_means['18-25'], bins = 35, ax = axis[1, 0],␣
 ↪color = 'y').set_title('Age Group 18-25')
sns.histplot(data = all_sample_means['46-50'], bins = 35, ax = axis[1, 1],␣
 ↪color = 'g').set_title('Age Group 46-50')
```

```
sns.histplot(data = all_sample_means['51-55'], bins = 35, ax = axis[2, 0]).
  ↪set_title('Age Group 51-55')
sns.histplot(data = all_sample_means['55+'], bins = 35, ax = axis[2, 1]).
  ↪set_title('Age Group 55+')

plt.show()
```

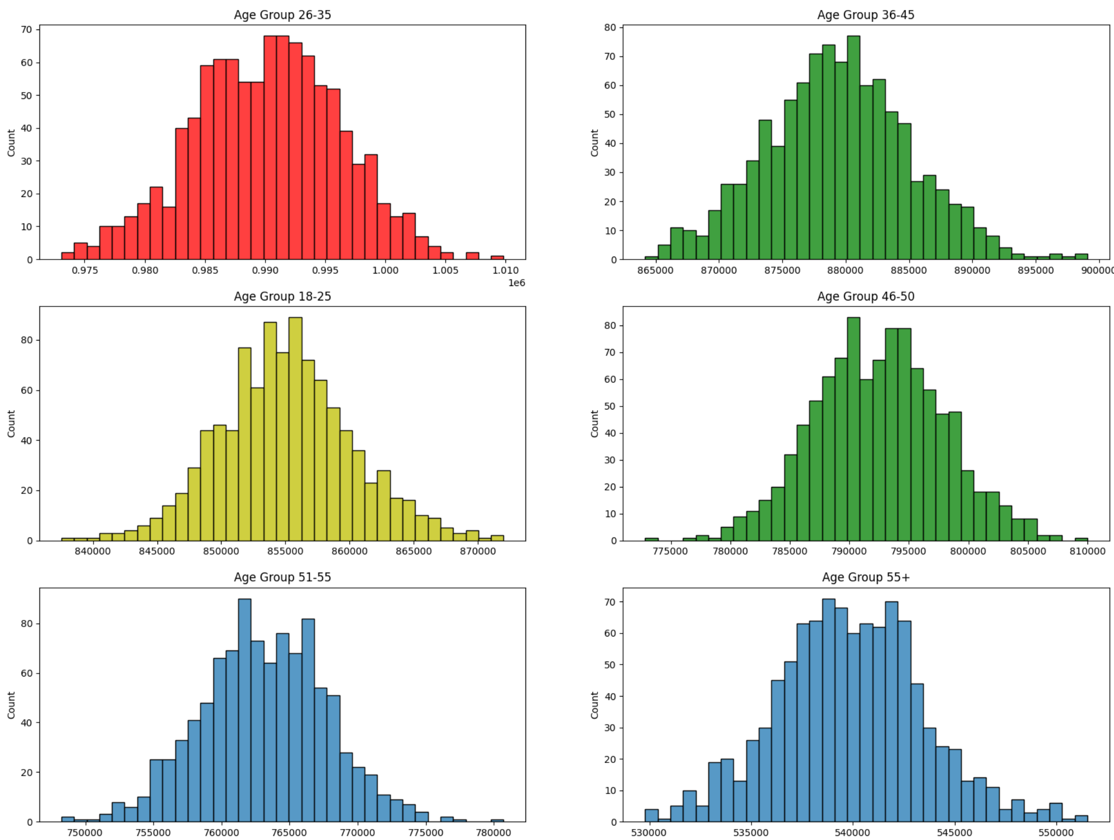


[126]:
```
plt.figure(figsize=(10, 5))
sns.histplot(all_sample_means['0-17'],bins=35,color='y')
plt.show()
```

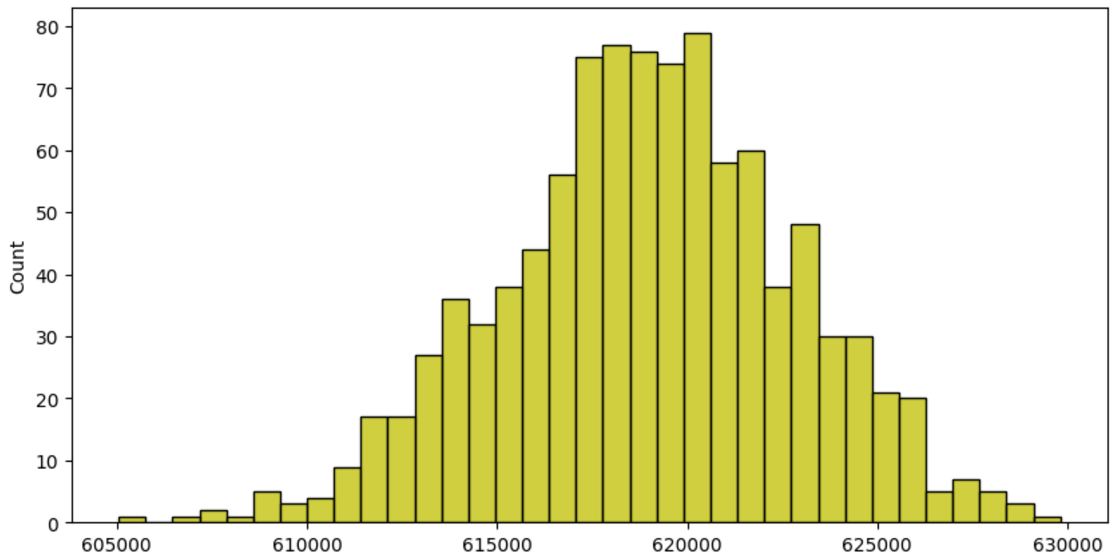


[127]:
```
sample_size = 30000
num_repitions = 1000

all_sample_means = {}

age_intervals = ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']
for i in age_intervals:
    all_sample_means[i] = []

for i in age_intervals:
    for j in range(num_repitions):

        mean = avgamt_age[avgamt_age['Age']==i].sample(sample_size,␣
 ↪replace=True)['Purchase'].mean()
        all_sample_means[i].append(mean)



fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(20, 15))

sns.histplot(data = all_sample_means['26-35'], bins = 35, ax = axis[0, 0],␣
 ↪color = 'r').set_title('Age Group 26-35')
sns.histplot(data = all_sample_means['36-45'], bins = 35, ax = axis[0, 1],␣
 ↪color = 'g').set_title('Age Group 36-45')
sns.histplot(data = all_sample_means['18-25'], bins = 35, ax = axis[1, 0],␣
 ↪color = 'y').set_title('Age Group 18-25')
sns.histplot(data = all_sample_means['46-50'], bins = 35, ax = axis[1, 1],␣
 ↪color = 'g').set_title('Age Group 46-50')
```

```
sns.histplot(data = all_sample_means['51-55'], bins = 35, ax = axis[2, 0]).
↪set_title('Age Group 51-55')
sns.histplot(data = all_sample_means['55+'], bins = 35, ax = axis[2, 1]).
↪set_title('Age Group 55+')

plt.show()
```



[128]:
```
plt.figure(figsize=(10, 5))
sns.histplot(all_sample_means['0-17'],bins=35,color='y')
plt.show()
```



[110]:
```
z95=1.960 #95% Confidence Interval

sample_size = 300
num_repitions = 1000

all_means = {}

age_intervals = ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']
for i in age_intervals:
    all_means[i] = []

for i in age_intervals:
    for j in range(num_repitions):
        mean = avgamt_age[avgamt_age['Age']==i].sample(sample_size,␣
 ↪replace=True)['Purchase'].mean()
        all_means[i].append(mean)
for val in ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']:

    new_df = avgamt_age[avgamt_age['Age']==val]

    std_error = z95*new_df['Purchase'].std()/np.sqrt(len(new_df))
    sample_mean = new_df['Purchase'].mean()
    lower_lim = sample_mean - std_error
    upper_lim = sample_mean + std_error


    print("For age {} confidence interval of means: ({:.2f}, {:.2f})".
 ↪format(val, lower_lim, upper_lim))
```

```
For age 26-35 confidence interval of means: (945034.42, 1034284.21)
For age 36-45 confidence interval of means: (823347.80, 935983.62)
For age 18-25 confidence interval of means: (801632.78, 908093.46)
For age 46-50 confidence interval of means: (713505.63, 871591.93)
For age 51-55 confidence interval of means: (692392.43, 834009.42)
For age 55+ confidence interval of means: (476948.26, 602446.23)
For age 0-17 confidence interval of means: (527662.46, 710073.17)
```

[111]:

```
z95=1.960 #95% Confidence Interval

sample_size = 3000
num_repitions = 1000

all_means = {}

age_intervals = ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']
for i in age_intervals:
    all_means[i] = []

for i in age_intervals:
    for j in range(num_repitions):
        mean = avgamt_age[avgamt_age['Age']==i].sample(sample_size,␣
 ↪replace=True)['Purchase'].mean()
        all_means[i].append(mean)
for val in ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']:

    new_df = avgamt_age[avgamt_age['Age']==val]

    std_error = z95*new_df['Purchase'].std()/np.sqrt(len(new_df))
    sample_mean = new_df['Purchase'].mean()
    lower_lim = sample_mean - std_error
    upper_lim = sample_mean + std_error


    print("For age {} confidence interval of means: ({:.2f}, {:.2f})".
 ↪format(val, lower_lim, upper_lim))
```


```
For age 26-35 confidence interval of means: (945034.42, 1034284.21)
For age 36-45 confidence interval of means: (823347.80, 935983.62)
For age 18-25 confidence interval of means: (801632.78, 908093.46)
For age 46-50 confidence interval of means: (713505.63, 871591.93)
For age 51-55 confidence interval of means: (692392.43, 834009.42)
For age 55+ confidence interval of means: (476948.26, 602446.23)
For age 0-17 confidence interval of means: (527662.46, 710073.17)
```


[112]:

```
z95=1.960 #95% Confidence Interval
```

```
sample_size = 30000
num_repitions = 1000

all_means = {}

age_intervals = ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']
for i in age_intervals:
    all_means[i] = []

for i in age_intervals:
    for j in range(num_repitions):
        mean = avgamt_age[avgamt_age['Age']==i].sample(sample_size,␣
↪replace=True)['Purchase'].mean()
        all_means[i].append(mean)
for val in ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']:

    new_df = avgamt_age[avgamt_age['Age']==val]

    std_error = z95*new_df['Purchase'].std()/np.sqrt(len(new_df))
    sample_mean = new_df['Purchase'].mean()
    lower_lim = sample_mean - std_error
    upper_lim = sample_mean + std_error


    print("For age {} confidence interval of means: ({:.2f}, {:.2f})".
↪format(val, lower_lim, upper_lim))
```

```
For age 26-35 confidence interval of means: (945034.42, 1034284.21)
For age 36-45 confidence interval of means: (823347.80, 935983.62)
For age 18-25 confidence interval of means: (801632.78, 908093.46)
For age 46-50 confidence interval of means: (713505.63, 871591.93)
For age 51-55 confidence interval of means: (692392.43, 834009.42)
For age 55+ confidence interval of means: (476948.26, 602446.23)
For age 0-17 confidence interval of means: (527662.46, 710073.17)
```

Insights:

- The age group 26-35 has the highest confidence interval range, suggesting that this age group might have the highest average of the measured variable.
- Lowest Mean Range: The age group 55+ has the lowest confidence interval range, indicating that this age group has the lowest average of the measured variable.
- Variability: The width of the confidence interval can give insight into variability. For instance, the 0-17 and 55+ age groups have relatively wide intervals, suggesting more variability in the data within these groups.

Recommendations : * Men spent more money than women, company can focus on retaining the male customers and getting more male customers.

- Product_Category - 1, 5, 8 have highest purchasing frequency. it means these are the products in these categories are in more demand. Company can focus on selling more of these products.
- Unmarried customers spend more money than married customers, So company should focus on acquisition of Unmarried customers.
- Customers in the age 26-35 spend more money than the others, So company should focus on acquisition of customers who are in the age 26-35.
- We have more customers aged 26-35 in the city category B and A, company can focus more on these customers for these cities to increase the business.
- Male customers living in City_Category C spend more money than other male customers living in B or C, Selling more products in the City_Category C will help the company increase the revenue.
- Some of the Product category like 19,20,13 have very less purchase. Company can think of dropping it.
- The top 10 users who have purchased more company should give more offers and discounts so that they can be retained and can be helpful for companies business.
- The occupation which are contributing more company can think of offering credit cards or other benefits to those customers by liasing with some financial partners to increase the sales.
- The top products should be given focus in order to maintain the quality in order to further increase the sales of those products.
- People who are staying in city for an year have contributed to 35% of the total purchase amount. Company can focus on such customer base who are neither too old nor too new residents in the city.
- We have highest frequency of purchase order between 5k and 10k, company can focus more on these mid range products to increase the sales.

[ ]:

```

```