walmart-busscase

April 4, 2024

Problem Statment : Walmart is an American multinational retail corporation that operates a chain of supercenters, discount departmental stores, and grocery stores in the United States. Walmart has more than 100 million customers worldwide. Wants to analyze the customer purchase behavior against the customer's gender and the various other factors to help the business make better decisions. Want to understand if the spending habits differ between male and female customers

[185]:
```
#Import the dataset and do usual data analysis
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv("F:\\buss_cass\\data\\walmart_data.txt")
```

Checking the structure & characteristics of the dataset

[186]:
```
df.head()
```

[186]:
```
   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
```

[187]:
```
df.shape
```

[187]:
```
(550068, 10)
```

[188]:
```
df.isna().sum()
```

[188]:
```
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
```

[189]:
```
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
```

[190]:
```
#there are some int datatypes coverting into object
dt = df.copy()
```

[191]:
```
columns=['User_ID','Occupation', 'Marital_Status', 'Product_Category']
dt[columns]=dt[columns].astype('object')
```

[192]:
```
dt.describe(include = 'all')
```

[192]:
```
           User_ID Product_ID  Gender     Age  Occupation City_Category  \
count     550068.0     550068  550068  550068    550068.0        550068
unique      5891.0       3631       2       7        21.0             3
top      1001680.0  P00265242       M   26-35         4.0             B
freq        1026.0       1880  414259  219587     72308.0        231173
mean           NaN        NaN     NaN     NaN         NaN           NaN
std            NaN        NaN     NaN     NaN         NaN           NaN
```

```
min             NaN         NaN     NaN     NaN          NaN           NaN
25%             NaN         NaN     NaN     NaN          NaN           NaN
50%             NaN         NaN     NaN     NaN          NaN           NaN
75%             NaN         NaN     NaN     NaN          NaN           NaN
max             NaN         NaN     NaN     NaN          NaN           NaN

       Stay_In_Current_City_Years  Marital_Status  Product_Category  \
count                      550068        550068.0          550068.0
unique                          5             2.0              20.0
top                             1             0.0               5.0
freq                       193821        324731.0          150933.0
mean                          NaN             NaN               NaN
std                           NaN             NaN               NaN
min                           NaN             NaN               NaN
25%                           NaN             NaN               NaN
50%                           NaN             NaN               NaN
75%                           NaN             NaN               NaN
max                           NaN             NaN               NaN

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

[ ]:

[193]:
```
df.nunique()
```

[193]:
```
User_ID                        5891
Product_ID                     3631
Gender                            2
Age                               7
Occupation                       21
City_Category                     3
Stay_In_Current_City_Years        5
Marital_Status                    2
Product_Category                 20
Purchase                      18105
dtype: int64
```

0.0.1 Observation:

```
1.In the get dataset there is no null values
2.There are total 5891 unique user and 3631 unique product_id
3.Most purchased product_id P00265242
4.UserId 1001680 has most visted customer
5.There are 7 unique age groups and most of the purchase belongs to age 26-35 group
```

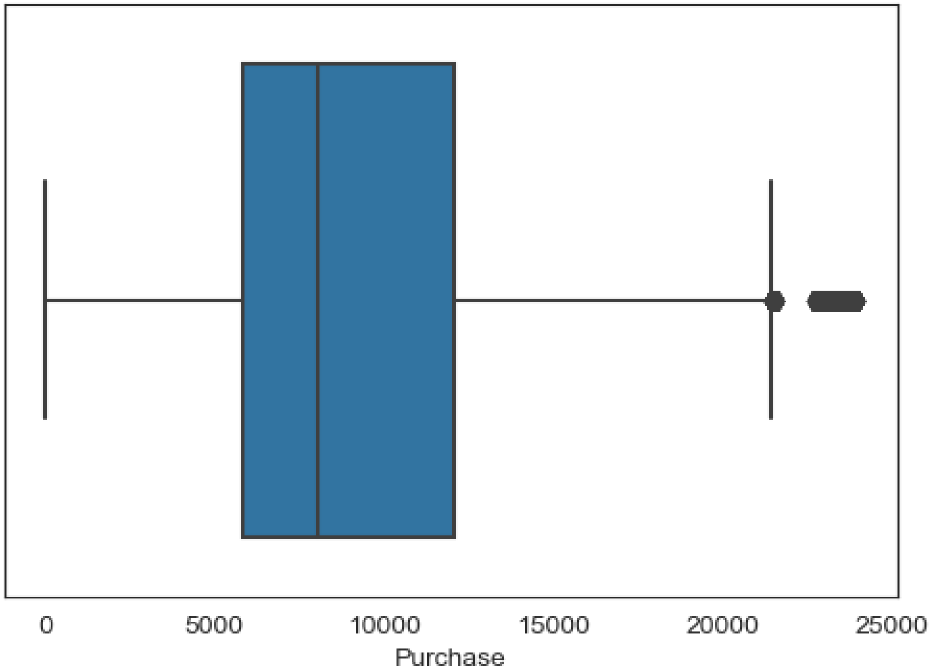
Missing Value & Outlier Detection

[194]:
```
df['Purchase'].describe()
```

[194]:
```
count    550068.000000
mean       9263.968713
std        5023.065394
min          12.000000
25%        5823.000000
50%        8047.000000
75%       12054.000000
max       23961.000000
Name: Purchase, dtype: float64
```

The mean value 9263.96 and min and max values are 12.00 and 23961 as we know outlier will impact the mean value here mean value is very less compare to the max value

[195]:
```
#Boxplot to find outliers
plt.figure(figsize=(6, 4))
sns.boxplot(data=df, x='Purchase', orient='h')
plt.show()
```



[196]:

```
#Removing the Outliers
def remove_outliers(data):
    # Calculate the 5% & 75%
    q5 = data.quantile(0.25)
    q95 = data.quantile(0.75)


    # Calculate the interquartile range
    iqr = q95 - q5

    # Define the lower and upper bounds for outliers
    lower_bound = q5 - 1.5 * iqr
    upper_bound = q95 + 1.5 * iqr

    clipped_data = np.clip(data, lower_bound, upper_bound)

    return clipped_data

# Example usage:
data = df['Purchase']
clipped_data = remove_outliers(data)
print("number of outliers: "+ str(len(clipped_data)))
print("max outlier value:"+ str(clipped_data.max()))
print("min outlier value: "+ str(clipped_data.min()))
```

```
number of outliers: 550068
max outlier value:21400.5
min outlier value: 12.0
```

[ ]:

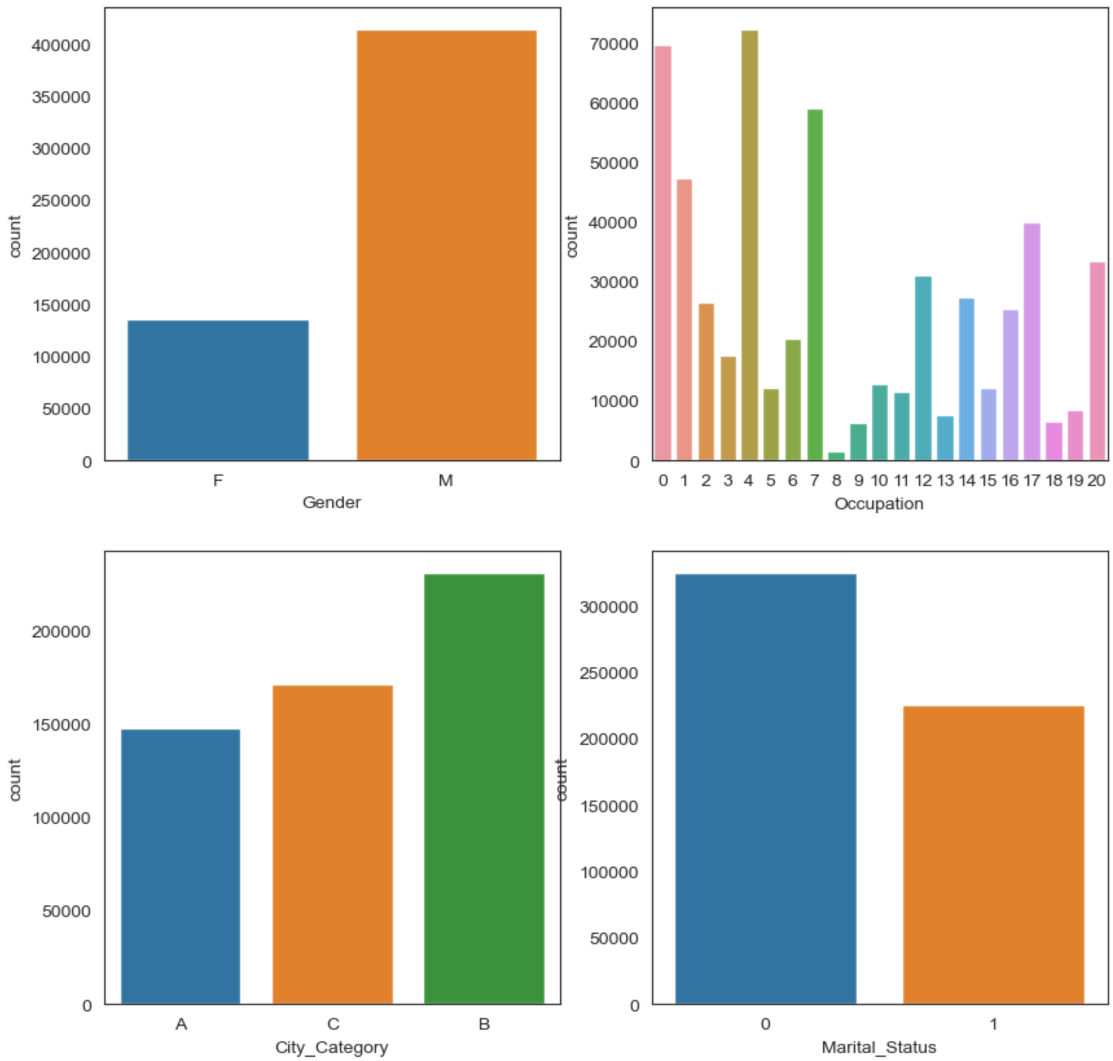
Data exploration

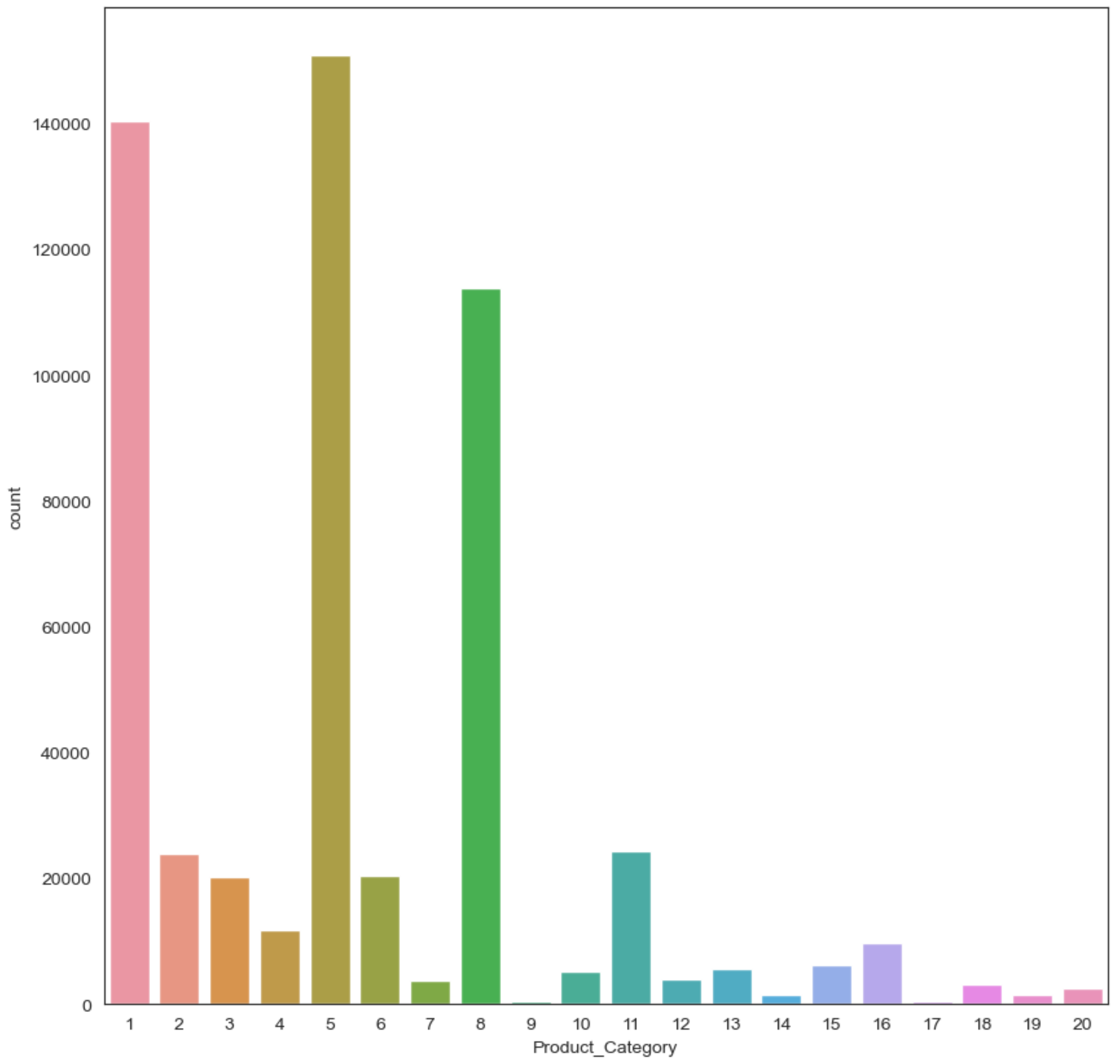
[ ]:

[ ]:

[197]:

```
categorical_cols = ['Gender',␣
 ↪'Occupation','City_Category','Marital_Status','Product_Category']
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(10, 10))
sns.countplot(data=df, x='Gender', ax=axs[0,0])
sns.countplot(data=df, x='Occupation', ax=axs[0,1])
sns.countplot(data=df, x='City_Category', ax=axs[1,0])
sns.countplot(data=df, x='Marital_Status', ax=axs[1,1])
plt.show()
plt.figure(figsize=(10, 10))
sns.countplot(data=df, x='Product_Category')
plt.show()
```


count
Gender
F
M
Occupation
City_Category
A
C
B
Marital_Status
0
1



[198]:
```python
df_prod = df[['Product_Category' , 'Age']].value_counts().reset_index()
df_prod.head()
```

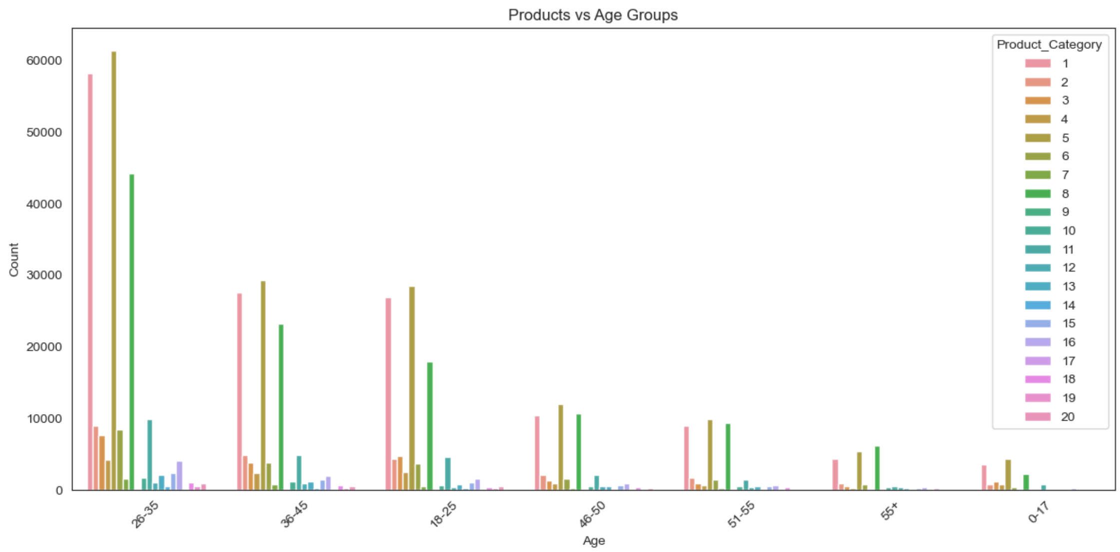
[198]:

	Product_Category	Age	count
0	5	26-35	61473
1	1	26-35	58249
2	8	26-35	44256
3	5	36-45	29377
4	5	18-25	28522

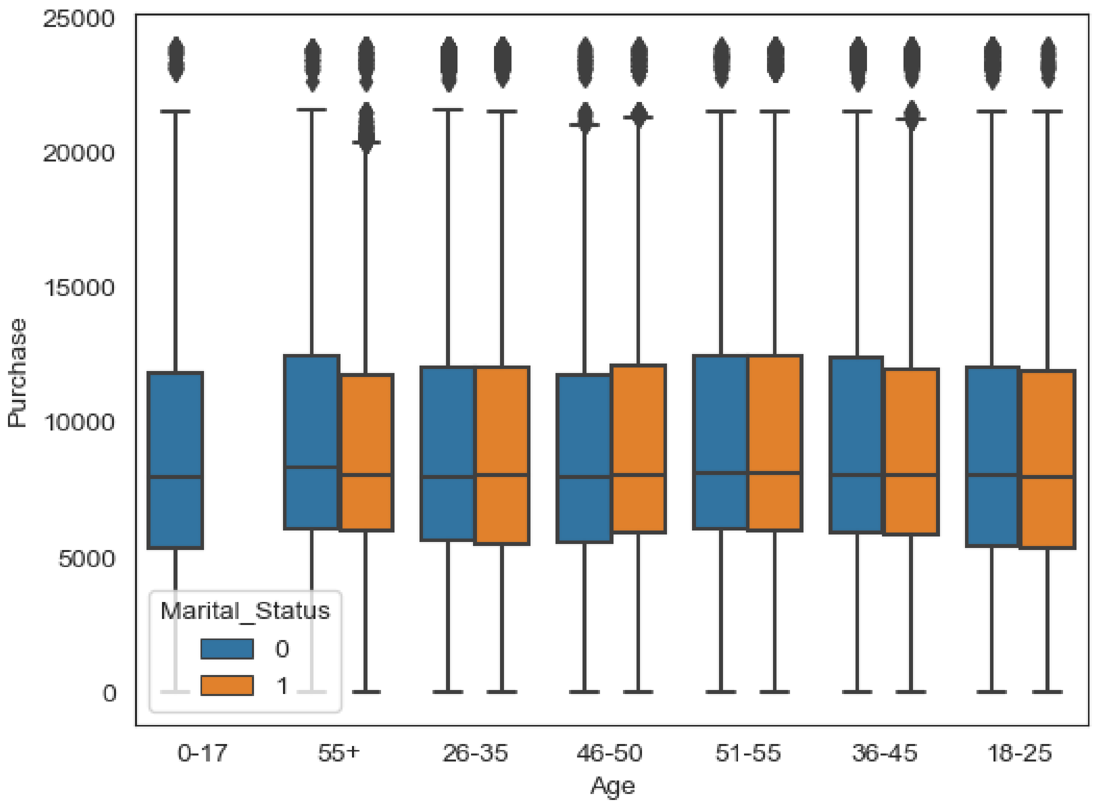
[ ]:

[199]:
```python
#Visual Analysis-products and age groups
plt.figure(figsize=(12, 6))
```

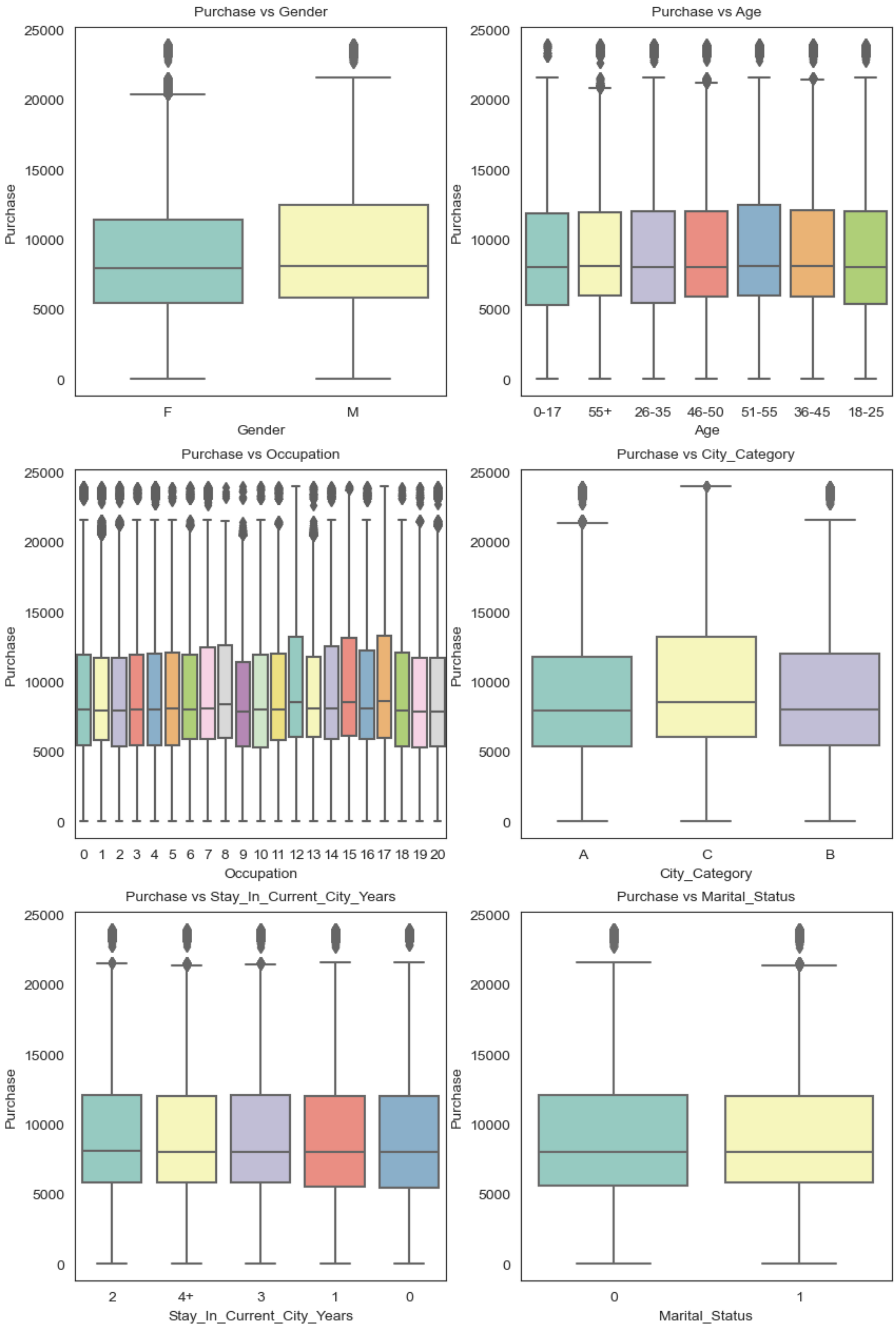
```
sns.barplot(data=df_prod, x='Age', y='count', hue='Product_Category')
plt.title('Products vs Age Groups')
plt.xlabel('Age')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



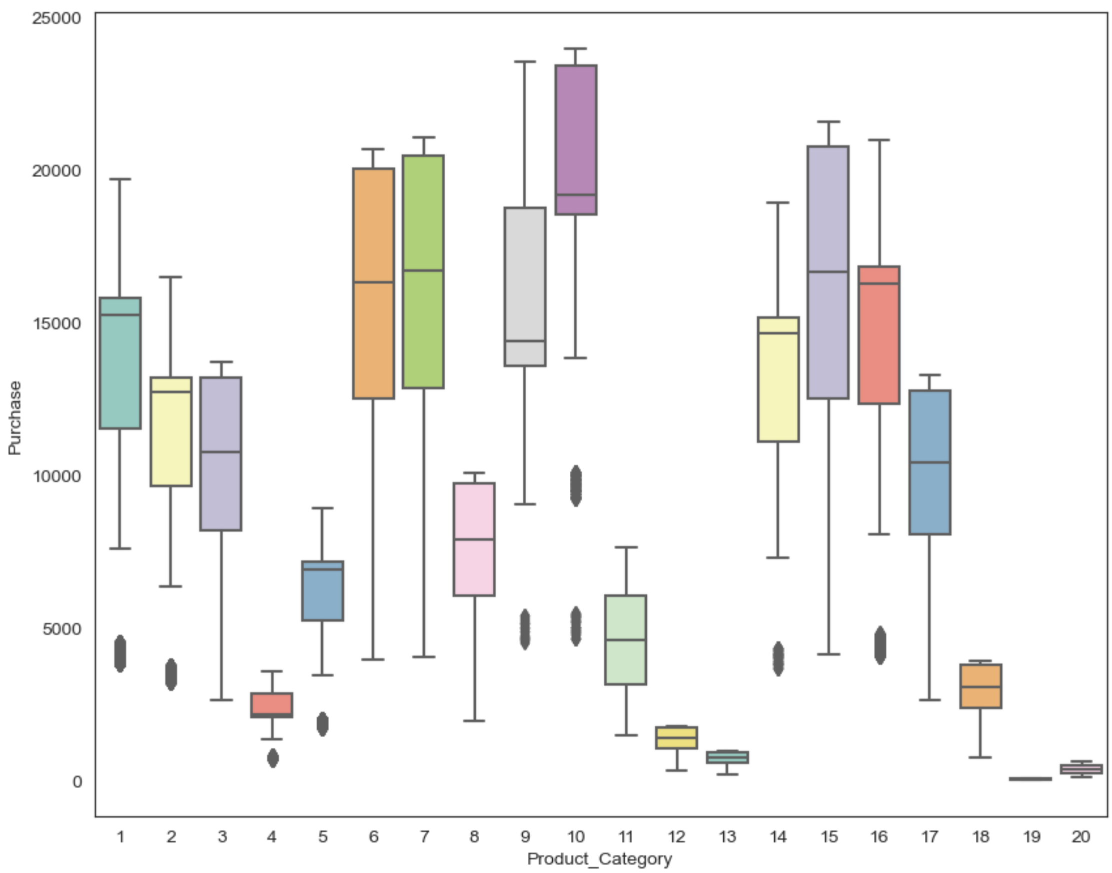
[200]:
```
#multivariate analysis between age,marital status, and the amount spent
sns.boxplot(data=df, y='Purchase', x='Age', hue='Marital_Status')
plt.show()
```



[201]:
```
#Visual Analysis
li = ['Gender', 'Age', 'Occupation', 'City_Category',
'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category']
sns.set_style("white")
fig, axs = plt.subplots(nrows=3, ncols=2, figsize=(10,10))
fig.subplots_adjust(top=1.3)
count = 0
for row in range(3):
    for col in range(2):
        sns.boxplot(data=df, y='Purchase', x=li[count], ax=axs[row,col],␣
 ↪palette='Set3')
        axs[row,col].set_title(f"Purchase vs {li[count]}", pad=8,fontsize=10)
        count += 1
plt.show()
```


Purchase vs Gender
Purchase
25000
20000
15000
10000
5000
0
F
M
Gender
Purchase vs Age
0-17
55+
26-35
46-50
51-55
36-45
18-25
Age
Purchase vs Occupation
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Occupation
Purchase vs City_Category
A
C
B
City_Category
Purchase vs Stay_In_Current_City_Years
2
4+
3
1
0
Stay_In_Current_City_Years
Purchase vs Marital_Status
0
1
Marital_Status

[202]:
```
plt.figure(figsize=(10, 8))
sns.boxplot(data=df, y='Purchase', x=attrs[-1], palette='Set3')
plt.show()
```



Observations

```
1.Most of the users are Male
2.There are 20 different types of Occupation and Product_Category
3.More users belong to B City_Category
4.More users are Single as compare to Married
5.Product_Category - 1, 5, 8, & 11 have highest purchasing frequency.
```

Average amount spends per customer for Male and Female

[204]:
```
cus_df = df.groupby(['User_ID', 'Gender'])[['Purchase']].sum().reset_index()
cus_df.head()
```

[204]:
```
   User_ID Gender  Purchase
0  1000001      F    334093
```

```
1  1000002       M    810472
2  1000003       M    341635
3  1000004       M    206468
4  1000005       M    821001
```

[205]:
```
# Gender wise value counts in avg_amt_df
cus_df['Gender'].value_counts()
```

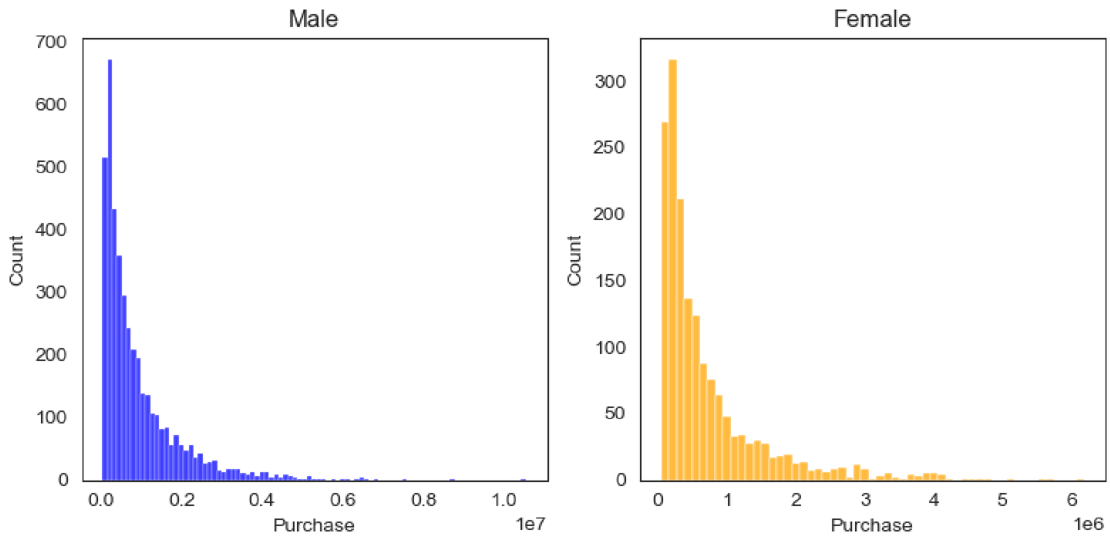
[205]:
```
Gender
M    4225
F    1666
Name: count, dtype: int64
```

[207]:
```
male_avg = cus_df[cus_df['Gender']=='M']['Purchase'].mean()
female_avg = cus_df[cus_df['Gender']=='F']['Purchase'].mean()
print(male_avg)
print(female_avg)
```

```
925344.4023668639
712024.3949579832
```

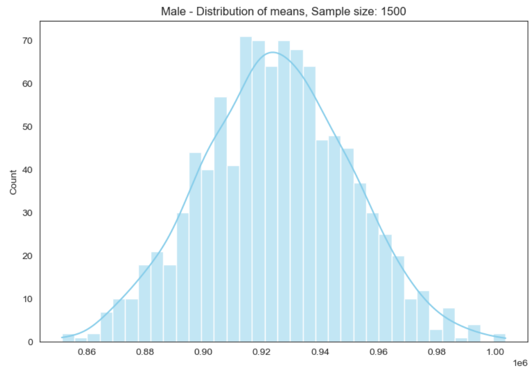
mean purchase of Male = 925344.40, female = 712024.39, male spend more money than female

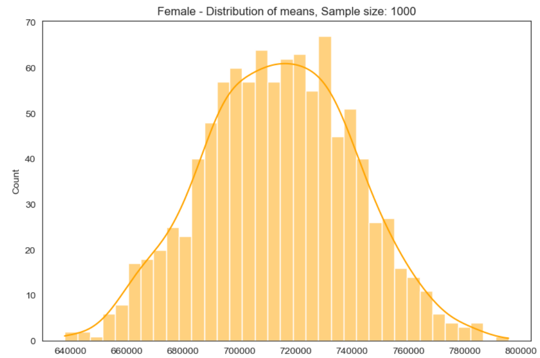
[227]:
```
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(8,4))

# Plot histogram for 'Male'
sns.histplot(data=cus_df[cus_df['Gender'] == 'M'], x='Purchase', ax=axs[0],␣
↪color='blue')
axs[0].set_title('Male')

# Plot histogram for 'Female'
sns.histplot(data=cus_df[cus_df['Gender'] == 'F'], x='Purchase', ax=axs[1],␣
↪color='orange')
axs[1].set_title('Female')
plt.tight_layout()
plt.show()
```




```
[265]: male_df = cus_df[cus_df['Gender']=='M']
female_df = cus_df[cus_df['Gender']=='F']
genders = ["M", "F"]
male_sample_size = 1500
female_sample_size = 1000
male_means = []
female_means = []
for _ in range(1000):
    male_mean = male_df.sample(male_sample_size,replace=True)['Purchase'].mean()
    female_mean = female_df.sample(female_sample_size,replace=True)['Purchase'].
↪mean()

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

```
[266]: fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))
sns.histplot(male_means, kde=True, bins=35, ax=axis[0], color='skyblue')
sns.histplot(female_means, kde=True, bins=35, ax=axis[1], color='orange')
axis[0].set_title("Male - Distribution of means, Sample size: 1500")
axis[1].set_title("Female - Distribution of means, Sample size: 1000")
plt.show()
```





[ ]:

[276]:
```
#Taking the values for z at 90%, 95% confidence interval
from scipy.stats import norm
z90 = norm.ppf(0.90)
z95 = norm.ppf(0.95)
```

Calculating 90% confidence interval for sample size male = 1500 & female = 1000:

[279]:
```
#male
sample_mean_M = np.mean(male_means)
sample_std_M = pd.Series(male_means).std()

se_M = (z90*sample_std_M)/(np.sqrt(1500))

lower_limit = sample_mean_M - se_M
upper_limit = sample_mean_M + se_M
print("Male_CI_90:", [lower_limit, upper_limit] )
```

```
Male_CI_90: [924020.0866301863, 925703.0202684805]
```

[280]:
```
#Female
sample_mean_F = np.mean(female_means)
sample_std_F = pd.Series(female_means).std()

se_F = (z90*sample_std_F)/(np.sqrt(1000))

lower_limit = sample_mean_F - se_F
upper_limit = sample_mean_F + se_F

print("Female_CI_90:", [lower_limit, upper_limit])
```

```
Female_CI_90: [712801.6347753559, 714966.6117366441]
```

Calculating 95% confidence interval for sample size male = 1500 & female = 1000:

[282]:
```
#male
sample_mean_M = np.mean(male_means)
sample_std_M = pd.Series(male_means).std()

se_M = (z95*sample_std_M)/(np.sqrt(1500))

lower_limit = sample_mean_M - se_M
upper_limit = sample_mean_M + se_M
print("Male_CI_95:", [lower_limit, upper_limit] )
```

```
Male_CI_95: [923781.5424793141, 925941.5644193527]
```

[283]:
```
#Female
sample_mean_F = np.mean(female_means)
sample_std_F = pd.Series(female_means).std()

se_F = (z95*sample_std_F)/(np.sqrt(1000))

lower_limit = sample_mean_F - se_F
upper_limit = sample_mean_F + se_F

print("Female_CI_95:", [lower_limit, upper_limit])
```

```
Female_CI_95: [712494.7643324954, 715273.4821795046]
```

CLT and Confidence interval considering marital status:

[288]:
```
avg_Marital = df.groupby(['User_ID', 'Marital_Status'])[['Purchase']].sum().
↪reset_index()
avg_Marital.head()
```

[288]:

	User_ID	Marital_Status	Purchase
0	1000001	0	334093
1	1000002	0	810472
2	1000003	0	341635
3	1000004	1	206468
4	1000005	1	821001

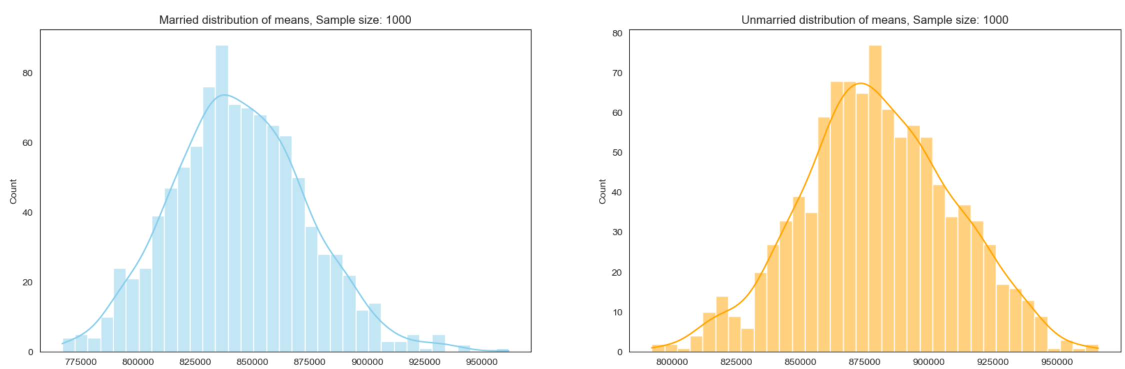
[289]:
```
avg_Marital['Marital_Status'].value_counts()
```

[289]:
```
Marital_Status
0    3417
1    2474
Name: count, dtype: int64
```

[287]:
```
avgamt_married = avg_Marital[avg_Marital['Marital_Status']==1]
avgamt_single = avg_Marital[avg_Marital['Marital_Status']==0]
```

```
sample_size = 1000
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)
```

[285]:
```
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))
sns.histplot(married_means, kde=True, bins=35, ax=axis[0], color='skyblue')
sns.histplot(single_means, kde=True, bins=35, ax=axis[1], color='orange')
axis[0].set_title("Married distribution of means, Sample size: 1000")
axis[1].set_title("Unmarried distribution of means, Sample size: 1000")

plt.show()
```



Calculating 90% confidence interval for sample size married = 1000 & singles = 1000:

[291]:
```
#married
sample_mean_M = np.mean(married_means)
sample_std_M = pd.Series(married_means).std()

se_M = (z90*sample_std_M)/(np.sqrt(1000))

lower_limit = sample_mean_M - se_M
upper_limit = sample_mean_M + se_M
print("Married_CI_90:", [lower_limit, upper_limit] )
```

```
Married_CI_90: [842975.4897113338, 845376.5821146662]
```

[292]:
```
#single
sample_mean_s = np.mean(single_means)
sample_std_s = pd.Series(single_means).std()

se_s = (z90*sample_std_s)/(np.sqrt(1000))

lower_limit = sample_mean_s - se_s
upper_limit = sample_mean_s + se_s
print("Single_CI_90:", [lower_limit, upper_limit] )
```

```
Single_CI_90: [877405.7785519884, 879883.8127680115]
```

Calculating 95% confidence interval for sample size married = 1000 & singles = 1000:

[294]:
```
#married
sample_mean_M = np.mean(married_means)
sample_std_M = pd.Series(married_means).std()

se_M = (z95*sample_std_M)/(np.sqrt(1000))

lower_limit = sample_mean_M - se_M
upper_limit = sample_mean_M + se_M
print("Married_CI_95:", [lower_limit, upper_limit] )
```

```
Married_CI_95: [842635.151545, 845716.920281]
```

[295]:
```
#single
sample_mean_s = np.mean(single_means)
sample_std_s = pd.Series(single_means).std()

se_s = (z95*sample_std_s)/(np.sqrt(1000))

lower_limit = sample_mean_s - se_s
upper_limit = sample_mean_s + se_s
print("Single_CI_95:", [lower_limit, upper_limit] )
```

```
Single_CI_95: [877054.534418267, 880235.0569017329]
```

CLT and Confidence interval considering Age:

[296]:
```
avgamt_age = df.groupby(['User_ID', 'Age'])[['Purchase']].sum().reset_index()
avgamt_age.head()
```

[296]:

	User_ID	Age	Purchase
0	1000001	0-17	334093
1	1000002	55+	810472
2	1000003	26-35	341635

```
3  1000004  46-50    206468
4  1000005  26-35    821001
```

[298]:

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

[298]:

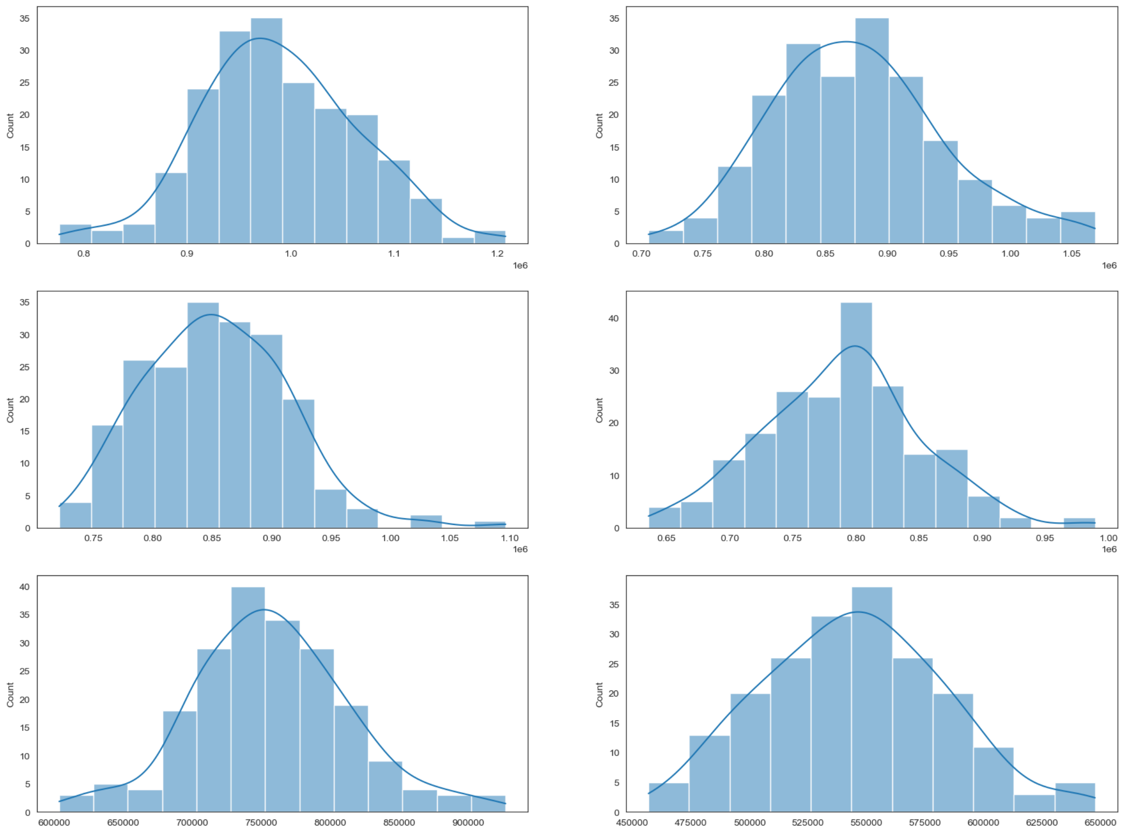
```
Age
26-35    2053
36-45    1167
18-25    1069
46-50     531
51-55     481
55+       372
0-17      218
Name: count, dtype: int64
```

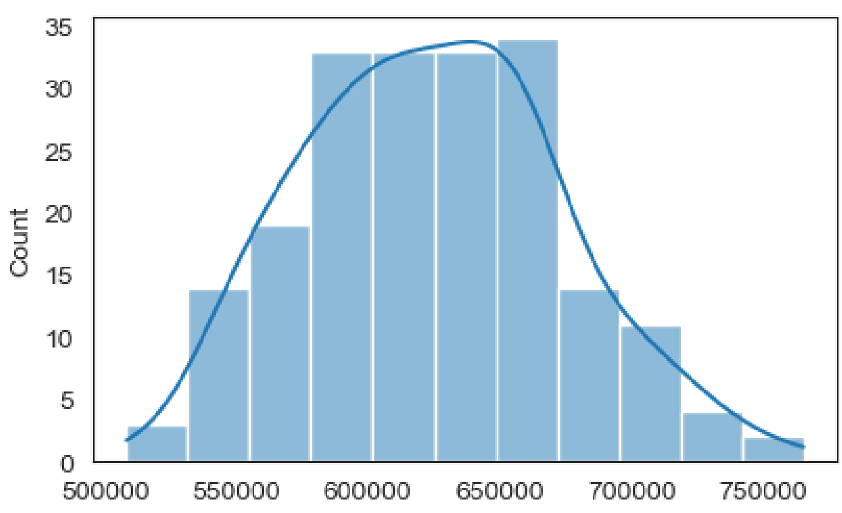
[319]:

```
sample_dict = {}

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

for i in age_intervals:
    for j in range(200):

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

[320]:

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

sns.histplot(sample_dict['26-35'],kde = True,ax=axis[0,0])
sns.histplot(sample_dict['36-45'],kde = True,ax=axis[0,1])
sns.histplot(sample_dict['18-25'],kde = True,ax=axis[1,0])
sns.histplot(sample_dict['46-50'],kde = True,ax=axis[1,1])
sns.histplot(sample_dict['51-55'],kde = True,ax=axis[2,0])
sns.histplot(sample_dict['55+'],kde = True,ax=axis[2,1])

plt.show()

plt.figure(figsize=(5, 3))
sns.histplot(sample_dict['0-17'],kde = True)
plt.show()
```





Calculating 90% confidence interval

[324]:
```
all_population_means={}
all_sample_means = {}

sample_size = 200
num_repitions = 1000

age_intervals = ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']
for i in age_intervals:
    all_sample_means[i] = []
    all_population_means[i]=[]
    population_mean=avgamt_age[avgamt_age['Age']==i]['Purchase'].mean()
    all_population_means[i].append(population_mean)

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)


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 = z90*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: (960481.20, 1018837.43)
For age 36-45 confidence interval of means: (842842.09, 916489.33)
For age 18-25 confidence interval of means: (820058.31, 889667.93)
For age 46-50 confidence interval of means: (740866.20, 844231.37)
For age 51-55 confidence interval of means: (716902.59, 809499.26)
For age 55+ confidence interval of means: (498668.64, 580725.85)
For age 0-17 confidence interval of means: (559232.93, 678502.69)
```

Calculating 95% confidence interval

[326]:
```
all_population_means={}
all_sample_means = {}

sample_size = 200
num_repitions = 1000
```

```python
age_intervals = ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']
for i in age_intervals:
    all_sample_means[i] = []
    all_population_means[i]=[]
    population_mean=avgamt_age[avgamt_age['Age']==i]['Purchase'].mean()
    all_population_means[i].append(population_mean)

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)


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: (952209.61, 1027109.02)
For age 36-45 confidence interval of means: (832403.10, 926928.32)
For age 18-25 confidence interval of means: (810191.63, 899534.61)
For age 46-50 confidence interval of means: (726214.90, 858882.66)
For age 51-55 confidence interval of means: (703777.65, 822624.20)
For age 55+ confidence interval of means: (487037.60, 592356.89)
For age 0-17 confidence interval of means: (542327.27, 695408.35)
```

Recommendations:

```
1.Men spent more money than women, company can focus on retaining the male customers and gettin

2. Product_Category - 1, 5, 8 have highest purchasing frequency. it means these are the product

3. Unmarried customers spend more money than married customers, So company should focus on mar

4. Customers in the age 26-35 spend more money than the others, So company should focus on acqu

5. We have more customers aged 26-35 in the city category B and A, company can focus more on th
```

6. Some of the Product category like 19,20,13 have very less purchase. Company can think of dr

7. The top 10 users who have purchased more company should give more offers and discounts so t

8. The occupation which are contributing more company can think of offering credit cards or ot

9. The top products should be given focus in order to maintain the quality in order to further

10. People who are staying in city for an year have contributed to 35% of the total purchase a

[ ]:

[ ]: