In [1]:

```
!gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv (https://d2b
eiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv)
To: C:\Users\anusha\Desktop\DAV prob & stats\walmart_data.csv

  0%|          | 0.00/23.0M [00:00<?, ?B/s]
  2%|2         | 524k/23.0M [00:00<00:08, 2.60MB/s]
  5%|4         | 1.05M/23.0M [00:00<00:06, 3.15MB/s]
  7%|6         | 1.57M/23.0M [00:00<00:06, 3.39MB/s]
  9%|9         | 2.10M/23.0M [00:00<00:05, 3.84MB/s]
 11%|#1        | 2.62M/23.0M [00:00<00:05, 3.59MB/s]
 16%|#5        | 3.67M/23.0M [00:01<00:04, 3.89MB/s]
 18%|#8        | 4.19M/23.0M [00:01<00:06, 2.93MB/s]
 20%|##        | 4.72M/23.0M [00:01<00:05, 3.20MB/s]
 23%|##2       | 5.24M/23.0M [00:01<00:05, 3.33MB/s]
 25%|##5       | 5.77M/23.0M [00:01<00:04, 3.53MB/s]
 27%|##7       | 6.29M/23.0M [00:01<00:05, 3.03MB/s]
 30%|##9       | 6.82M/23.0M [00:02<00:04, 3.28MB/s]
 32%|###1      | 7.34M/23.0M [00:02<00:04, 3.20MB/s]
 34%|###4      | 7.86M/23.0M [00:02<00:05, 2.62MB/s]
 36%|###6      | 8.39M/23.0M [00:02<00:06, 2.35MB/s]
 39%|###8      | 8.91M/23.0M [00:02<00:05, 2.41MB/s]
 41%|####      | 9.44M/23.0M [00:03<00:06, 2.22MB/s]
 43%|####3     | 9.96M/23.0M [00:03<00:05, 2.27MB/s]
 46%|####5     | 10.5M/23.0M [00:03<00:06, 2.09MB/s]
 48%|####7     | 11.0M/23.0M [00:03<00:05, 2.22MB/s]
 50%|#####     | 11.5M/23.0M [00:04<00:04, 2.31MB/s]
 52%|#####2    | 12.1M/23.0M [00:04<00:06, 1.73MB/s]
 55%|#####4    | 12.6M/23.0M [00:04<00:05, 2.06MB/s]
 57%|#####6    | 13.1M/23.0M [00:05<00:04, 2.23MB/s]
 59%|#####9    | 13.6M/23.0M [00:05<00:03, 2.43MB/s]
 61%|######1   | 14.2M/23.0M [00:05<00:03, 2.42MB/s]
 64%|######3   | 14.7M/23.0M [00:05<00:03, 2.58MB/s]
 66%|######6   | 15.2M/23.0M [00:05<00:02, 2.77MB/s]
 68%|######8   | 15.7M/23.0M [00:05<00:02, 2.83MB/s]
 71%|#######   | 16.3M/23.0M [00:06<00:02, 2.89MB/s]
 73%|#######2  | 16.8M/23.0M [00:06<00:02, 2.79MB/s]
 75%|#######5  | 17.3M/23.0M [00:06<00:02, 2.60MB/s]
 77%|#######7  | 17.8M/23.0M [00:06<00:02, 2.53MB/s]
 80%|#######9  | 18.4M/23.0M [00:06<00:01, 2.55MB/s]
 82%|########1 | 18.9M/23.0M [00:07<00:01, 2.62MB/s]
 84%|########4 | 19.4M/23.0M [00:07<00:01, 2.55MB/s]
 87%|########6 | 19.9M/23.0M [00:07<00:01, 2.20MB/s]
 89%|########8 | 20.4M/23.0M [00:07<00:01, 2.13MB/s]
 91%|#########1| 21.0M/23.0M [00:08<00:00, 2.34MB/s]
 93%|#########3| 21.5M/23.0M [00:08<00:00, 2.26MB/s]
 96%|#########5| 22.0M/23.0M [00:08<00:00, 2.37MB/s]
 98%|#########7| 22.5M/23.0M [00:08<00:00, 2.28MB/s]
100%|##########| 23.0M/23.0M [00:08<00:00, 2.36MB/s]
100%|##########| 23.0M/23.0M [00:08<00:00, 2.56MB/s]
```

In [2]:

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

In [3]:

```
df = pd.read_csv("/Users/anusha/Desktop/DAV prob & stats/walmart_data.csv")
```

In [4]:

```
df
```

Out[4]:

	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
...	...	...	...	...	...	...	...	...	...	...
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

550068 rows × 10 columns

observations of data type and shape of the data

In [18]:

```
df.shape
```

Out[18]:

```
(550068, 10)
```

In [37]:

```
df.describe()
```

Out[37]:

	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 [26]:

```
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  object
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  object
 8   Product_Category            550068 non-null  object
 9   Purchase                    550068 non-null  int64 
dtypes: int64(2), object(8)
memory usage: 42.0+ MB
```

In [5]:

```
cols = ['Occupation', 'Marital_Status', 'Product_Category']
df[cols] = df[cols].astype('object')
```

In [27]:

```
df.groupby('Gender')['User_ID'].nunique()
```

Out[27]:

```
Gender
F    1666
M    4225
Name: User_ID, dtype: int64
```

In [28]:

```
df.groupby('Gender')['User_ID'].describe(include=all)
```

Out[28]:

Gender	count	mean	std	min	25%	50%	75%	max
F	135809.0	1.003130e+06	1786.630589	1000001.0	1001569.0	1003159.0	1004765.0	1006039.0
M	414259.0	1.002996e+06	1706.493873	1000002.0	1001505.0	1003041.0	1004411.0	1006040.0

In [29]:

```
df.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
```

value counts and unique attributes

In [9]:

```
df['User_ID'].nunique()
```

Out[9]:

```
5891
```

In [11]:

```
df['Product_ID'].nunique()
```

Out[11]:

```
3631
```

In [14]:

```
gender_count = df.groupby('Gender')['User_ID'].nunique()
gender_count
```

Out[14]:

```
Gender
F    1666
M    4225
Name: User_ID, dtype: int64
```

In [14]:

```
Age_count = df['Age'].value_counts().reset_index()
Age_count.columns = ['Age', 'Count']
Age_count
```

Out[14]:

	Age	Count
0	26-35	219587
1	36-45	110013
2	18-25	99660
3	46-50	45701
4	51-55	38501
5	55+	21504
6	0-17	15102

In [13]:

```
Occupation_count = df.groupby('Occupation')['User_ID'].nunique()
Occupation_count
```

Out[13]:

```
Occupation
0     688
1     517
2     256
3     170
4     740
5     111
6     228
7     669
8      17
9      88
10    192
11    128
12    376
13    140
14    294
15    140
16    235
17    491
18     67
19     71
20    273
Name: User_ID, dtype: int64
```

In [15]:

```
City_Category_count = df.groupby('City_Category')['User_ID'].nunique()

City_Category_count
```

Out[15]:

```
City_Category
A    1045
B    1707
C    3139
Name: User_ID, dtype: int64
```

In [16]:

```
Stay_In_Current_City_Years_count = df.groupby('Stay_In_Current_City_Years')['User_ID'].nunique()

Stay_In_Current_City_Years_count
```

Out[16]:

```
Stay_In_Current_City_Years
0      772
1     2086
2     1145
3      979
4+     909
Name: User_ID, dtype: int64
```

In [18]:

```
Product_Category_count = df['Product_Category'].value_counts().reset_index()
Product_Category_count.columns = ['Product_Category', 'Count']
Product_Category_count
```

Out[18]:

	Product_Category	Count
0	5	150933
1	1	140378
2	8	113925
3	11	24287
4	2	23864
5	6	20466
6	3	20213
7	4	11753
8	16	9828
9	15	6290
10	13	5549
11	10	5125
12	12	3947
13	7	3721
14	18	3125
15	20	2550
16	19	1603
17	14	1523
18	17	578
19	9	410

In [17]:

```
Marital_Status_count = df.groupby('Marital_Status')['User_ID'].nunique()
Marital_Status_count.columns = ['Martial_Status','count']
Marital_Status_count
```

Out[17]:

```
Marital_Status
0    3417
1    2474
Name: User_ID, dtype: int64
```

In [5]:

```
#Basic data exploration using contigency tab
```

In [11]:

```
df.groupby('Gender')['Purchase'].describe()
```

Out[11]:

Gender	count	mean	std	min	25%	50%	75%	max
F	135809.0	8734.565765	4767.233289	12.0	5433.0	7914.0	11400.0	23959.0
M	414259.0	9437.526040	5092.186210	12.0	5863.0	8098.0	12454.0	23961.0

In [6]:

```
pd.crosstab(index=df["City_Category"],columns=df["Age"],margins=True,normalize="index")
```

Out[6]:

Age / City_Category	0-17	18-25	26-35	36-45	46-50	51-55	55+
A	0.017222	0.186400	0.499222	0.180185	0.051496	0.041288	0.024188
B	0.023511	0.187076	0.396171	0.205898	0.088272	0.076743	0.022330
C	0.041612	0.168705	0.316974	0.209131	0.103333	0.085649	0.074596
All	0.027455	0.181178	0.399200	0.199999	0.083082	0.069993	0.039093

In [15]:

```
df.groupby(['Gender','Marital_Status'])['Purchase'].count().unstack()
```

Out[15]:

Marital_Status / Gender	0	1
F	78821	56988
M	245910	168349

In [21]:

```
pd.crosstab(df['Marital_Status'],[df['Gender']],normalize = True,margins = True,margins_name = 'Total')*100
```

Out[21]:

Gender / Marital_Status	F	M	Total
0	14.329319	44.705382	59.034701
1	10.360174	30.605125	40.965299
Total	24.689493	75.310507	100.000000

In [5]:

```
pd.crosstab(df['Age'],[df['Gender']],normalize = True,margins = True,margins_name = 'Total')*100
```

Out[5]:

Gender / Age	F	M	Total
0-17	0.924068	1.821411	2.745479
18-25	4.477265	13.640495	18.117760
26-35	9.226496	30.693478	39.919974
36-45	4.939389	15.060502	19.999891
46-50	2.399522	5.908724	8.308246
51-55	1.798687	5.200630	6.999316
55+	0.924068	2.985267	3.909335
Total	24.689493	75.310507	100.000000

In [6]:

```
pd.crosstab(df['Stay_In_Current_City_Years'],[df['Gender']],normalize = True,margins = True,margins_name = 'Total')*100
```

Out[6]:

Gender Stay_In_Current_City_Years	F	M	Total
0	3.101980	10.423257	13.525237
1	9.325756	25.910069	35.235825
2	4.423453	14.090258	18.513711
3	4.457631	12.864773	17.322404
4+	3.380673	12.022150	15.402823
Total	24.689493	75.310507	100.000000

In [9]:

```
pd.crosstab(df['Stay_In_Current_City_Years'],[df['Marital_Status']],normalize = True,margins = True,margins_name = 'Total')*100
```

Out[9]:

Marital_Status Stay_In_Current_City_Years	0	1	Total
0	8.164082	5.361155	13.525237
1	20.124057	15.111768	35.235825
2	11.053179	7.460532	18.513711
3	10.479977	6.842427	17.322404
4+	9.213406	6.189417	15.402823
Total	59.034701	40.965299	100.000000

In [8]:

```
pd.crosstab(df['Marital_Status'],[df['Age']],normalize = True,margins = True,margins_name = 'Total')*100
```

Out[8]:

Age Marital_Status	0-17	18-25	26-35	36-45	46-50	51-55	55+	Total
0	2.745479	14.278962	24.232640	12.067054	2.306987	1.970484	1.433096	59.034701
1	0.000000	3.838798	15.687333	7.932837	6.001258	5.028833	2.476239	40.965299
Total	2.745479	18.117760	39.919974	19.999891	8.308246	6.999316	3.909335	100.000000

In [11]:

```
df.groupby('City_Category')['Purchase'].describe()
```

Out[11]:

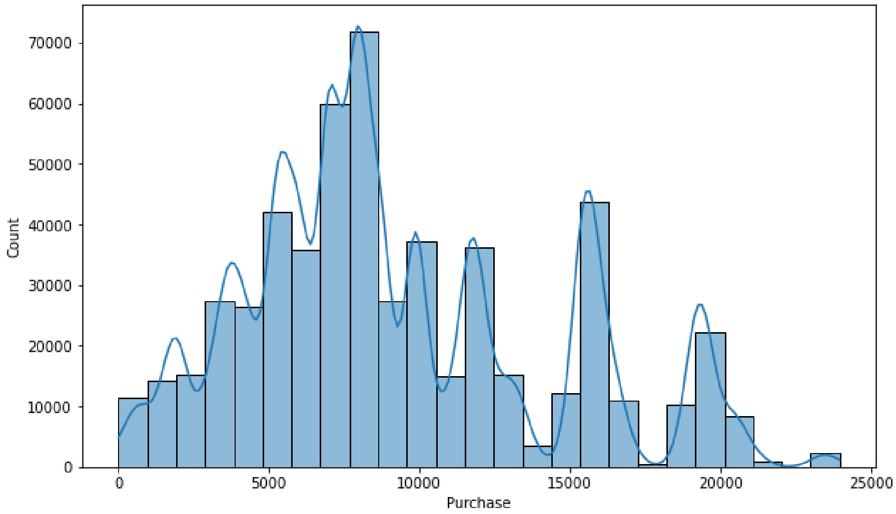
City_Category	count	mean	std	min	25%	50%	75%	max
A	147720.0	8911.939216	4892.115238	12.0	5403.0	7931.0	11786.0	23961.0
B	231173.0	9151.300563	4955.496566	12.0	5460.0	8005.0	11986.0	23960.0
C	171175.0	9719.920993	5189.465121	12.0	6031.5	8585.0	13197.0	23961.0

In [ ]:

```
#univariate Analysis
```

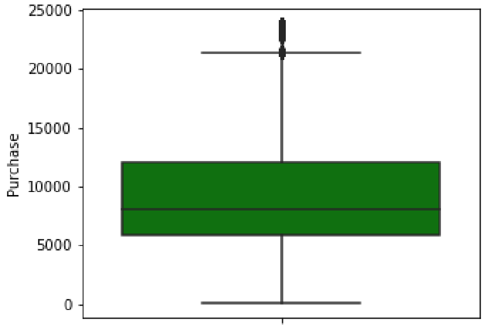
In [6]:

```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x="Purchase", kde=True,bins = 25)
plt.show()
```



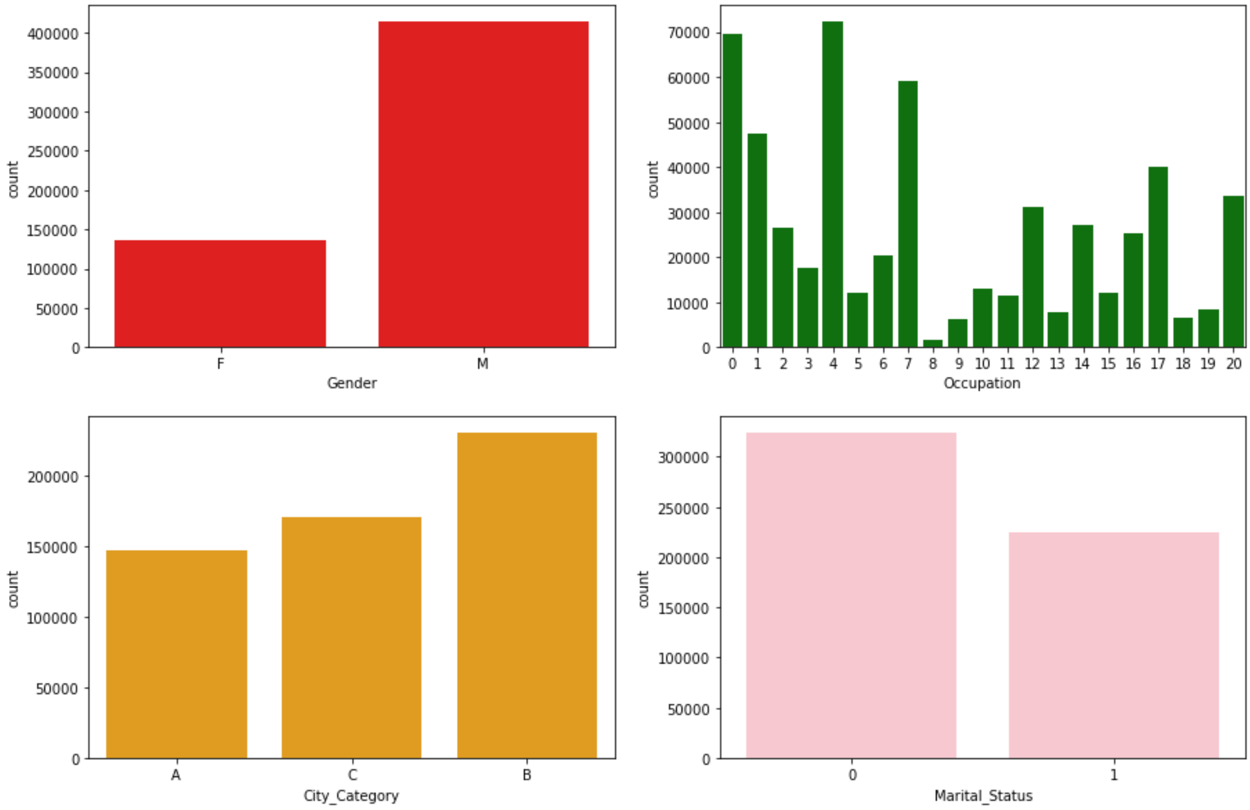
In [9]:

```
plt.figure(figsize=(5, 4))
sns.boxplot(data=df, y='Purchase',color='Green')
plt.show()
```



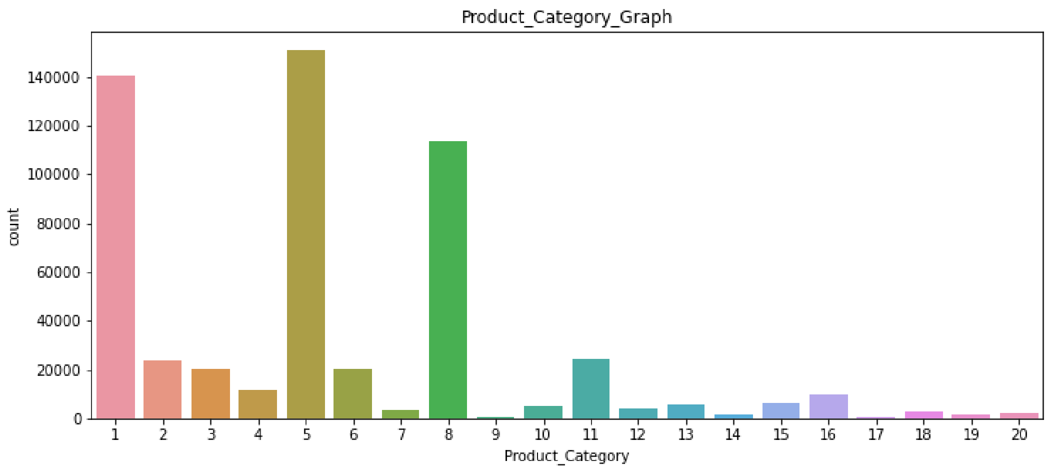
In [11]:

```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))
sns.countplot(data=df, x='Gender', ax=axs[0,0],color = 'Red')
sns.countplot(data=df, x='Occupation', ax=axs[0,1],color = 'Green')
sns.countplot(data=df, x='City_Category', ax=axs[1,0],color = 'Orange')
sns.countplot(data=df, x='Marital_Status', ax=axs[1,1],color = 'Pink' )
plt.show("Gender_Graph")
plt.show("Occupation_Graph")
plt.show("City_Category_Graph")
plt.show("Marital_Status_Graph")
plt.show()
```



In [13]:

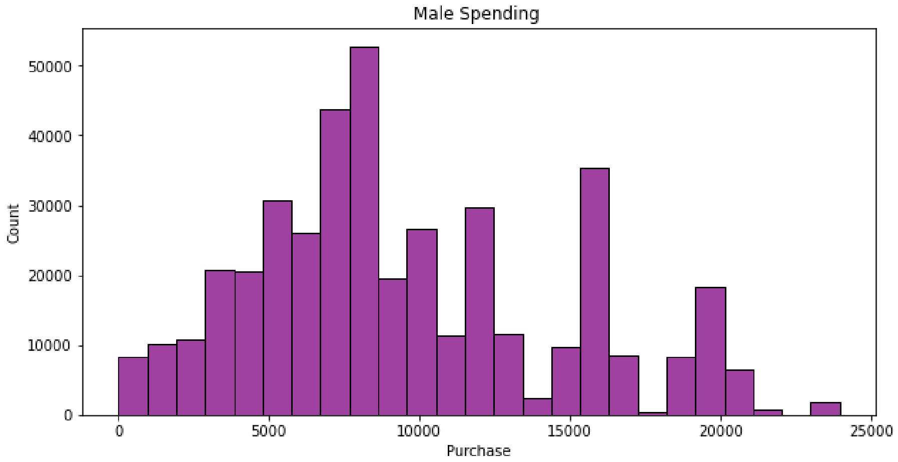
```
plt.figure(figsize=(12, 5))
sns.countplot(data=df, x='Product_Category')
plt.title("Product_Category_Graph")
plt.show()
```



Bivariate Analysis:

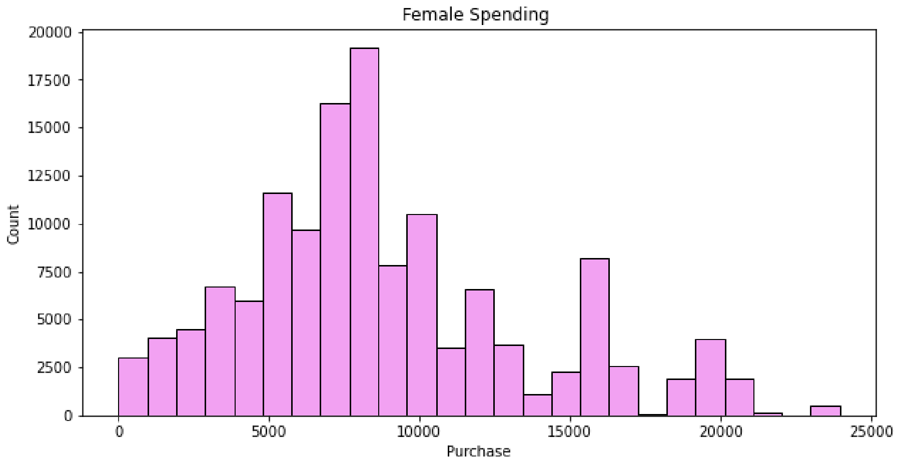
In [20]:

```
plt.figure(figsize=(10,5))
sns.histplot(data=df[df['Gender']=='M']['Purchase'],bins = 25,color='Purple')
plt.title("Male Spending ")
plt.show()
```



In [22]:

```
plt.figure(figsize=(10,5))
sns.histplot(data=df[df['Gender']=='F']['Purchase'],bins = 25,color='Violet')
plt.title("Female Spending ")
plt.show()
```

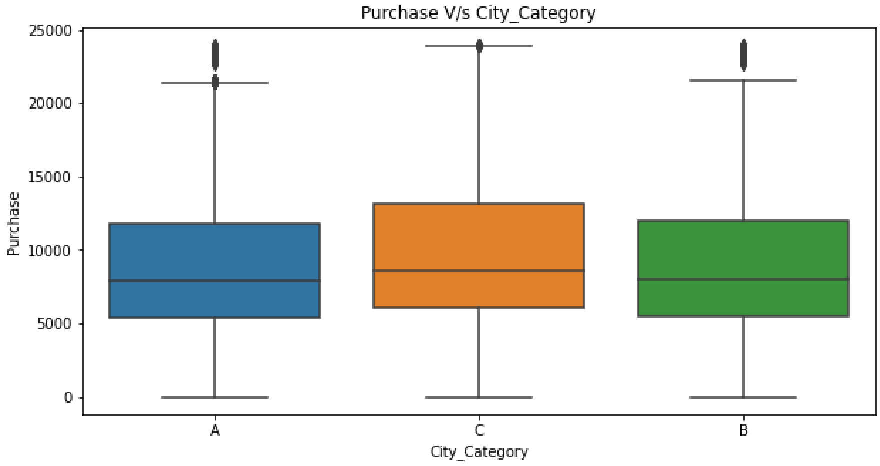


In [7]:

```
plt.figure(figsize=(10, 5))
sns.boxplot(data=df,y = 'Purchase',x='Occupation')
plt.title("Purchase V/s occupation")
plt.show()
```

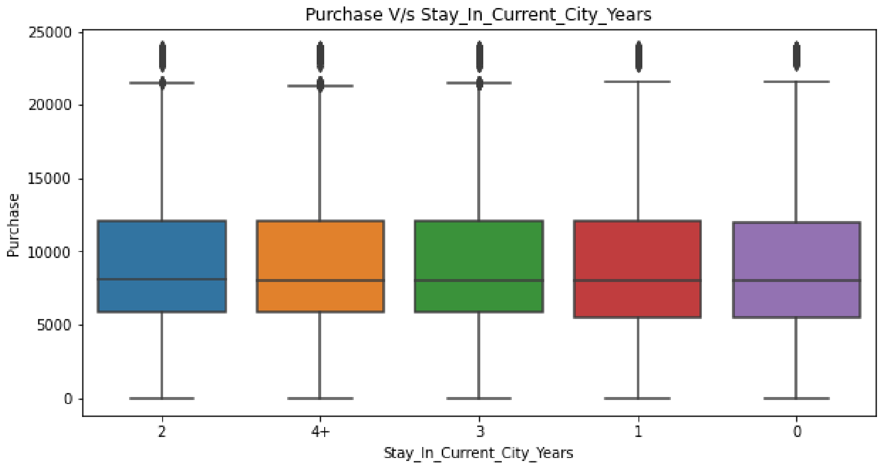
In [8]:

```
plt.figure(figsize=(10, 5))
sns.boxplot(data=df,y = 'Purchase',x='City_Category')
plt.title("Purchase V/s City_Category")
plt.show()
```



In [9]:

```
plt.figure(figsize=(10, 5))
sns.boxplot(data=df,y = 'Purchase',x='Stay_In_Current_City_Years')
plt.title("Purchase V/s Stay_In_Current_City_Years")
plt.show()
```

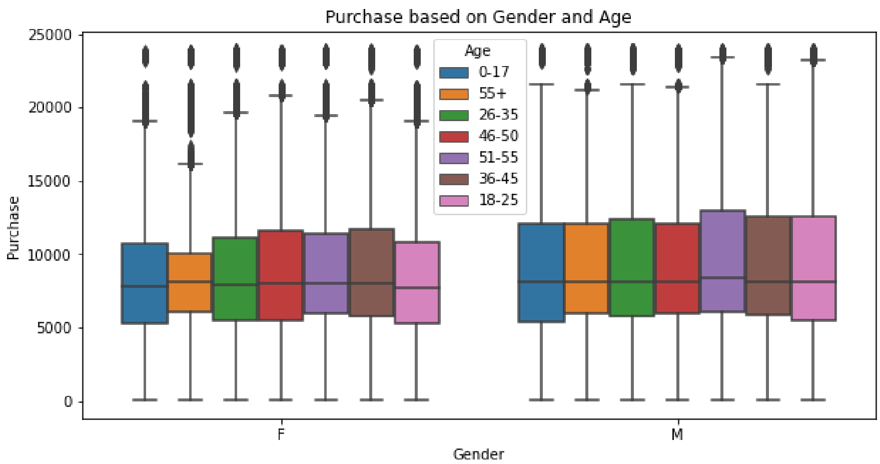


In [ ]:

```
#Multivariant Analysis
```

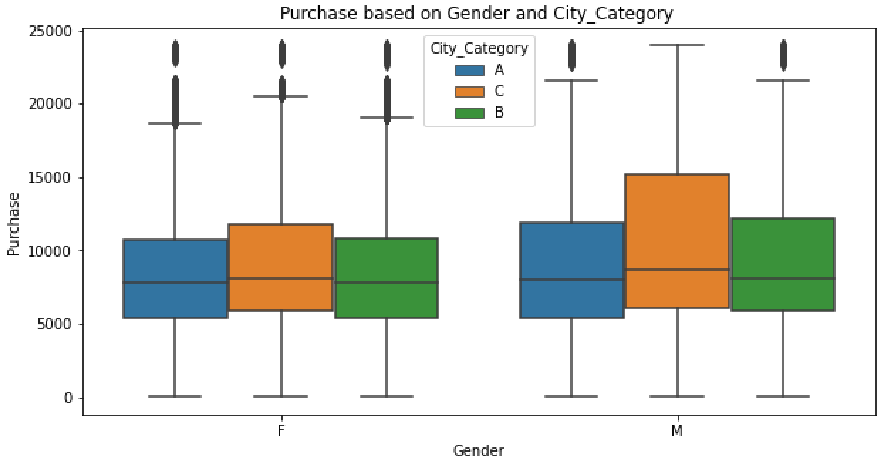
In [10]:

```
plt.figure(figsize=(10,5))
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Age')
plt.title("Purchase based on Gender and Age")
plt.show()
```



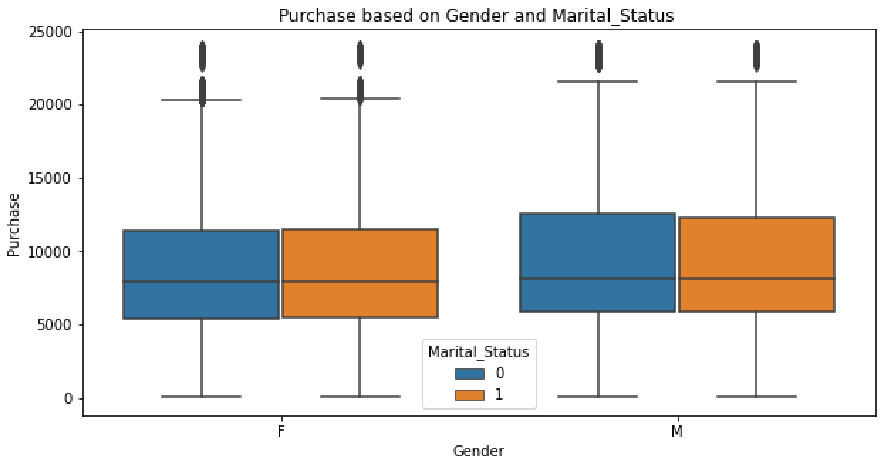
In [11]:

```
plt.figure(figsize=(10,5))
sns.boxplot(data=df, y='Purchase', x='Gender', hue='City_Category')
plt.title("Purchase based on Gender and City_Category ")
plt.show()
```



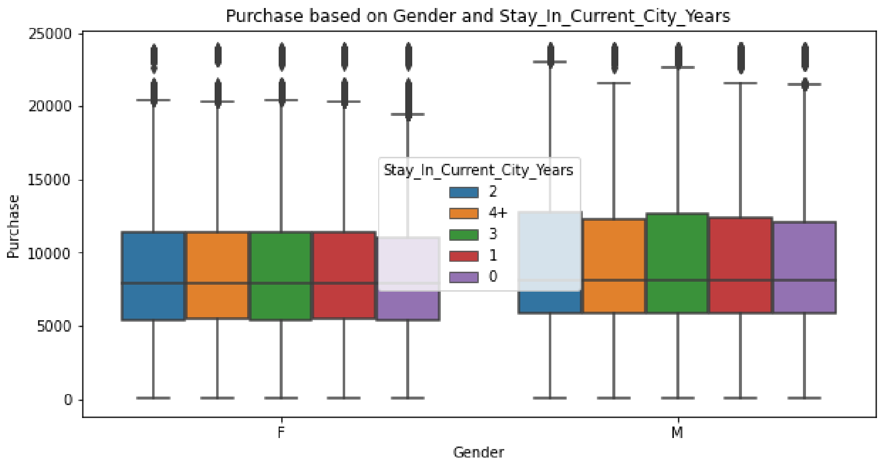
In [12]:

```
plt.figure(figsize=(10,5))
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Marital_Status')
plt.title("Purchase based on Gender and Marital_Status ")
plt.show()
```



In [13]:

```
plt.figure(figsize=(10,5))
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Stay_In_Current_City_Years')
plt.title("Purchase based on Gender and Stay_In_Current_City_Years")
plt.show()
```



In [ ]:

```
#For correlation: Heatmaps, Pairplots
```

In [38]:

```
df.corr()
```

Out[38]:

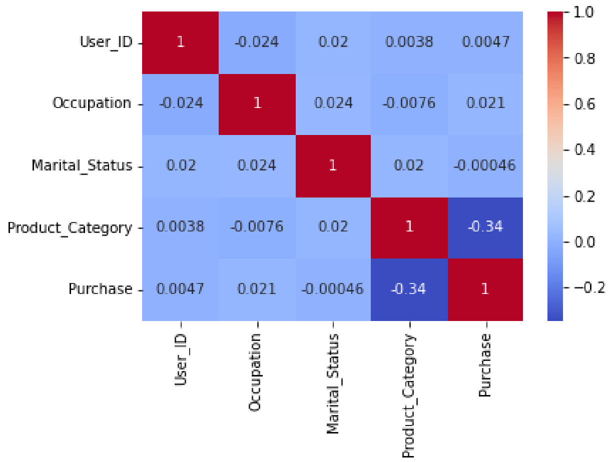
	User_ID	Occupation	Marital_Status	Product_Category	Purchase
User_ID	1.000000	-0.023971	0.020443	0.003825	0.004716
Occupation	-0.023971	1.000000	0.024280	-0.007618	0.020833
Marital_Status	0.020443	0.024280	1.000000	0.019888	-0.000463
Product_Category	0.003825	-0.007618	0.019888	1.000000	-0.343703
Purchase	0.004716	0.020833	-0.000463	-0.343703	1.000000

In [39]:

```
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
```

Out[39]:

<AxesSubplot:>



In [ ]:

```
plt.figure(figsize=(8, 5))
sns.histplot(data=df[df['Gender']=='M']['Purchase'],bins = 25,color='Blue')
plt.title("Purchase V/s Product_category")
plt.show()
```

Average money spend by per customer of male and Female

In [6]:

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

Out[6]:

	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 × 3 columns

In [7]:

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

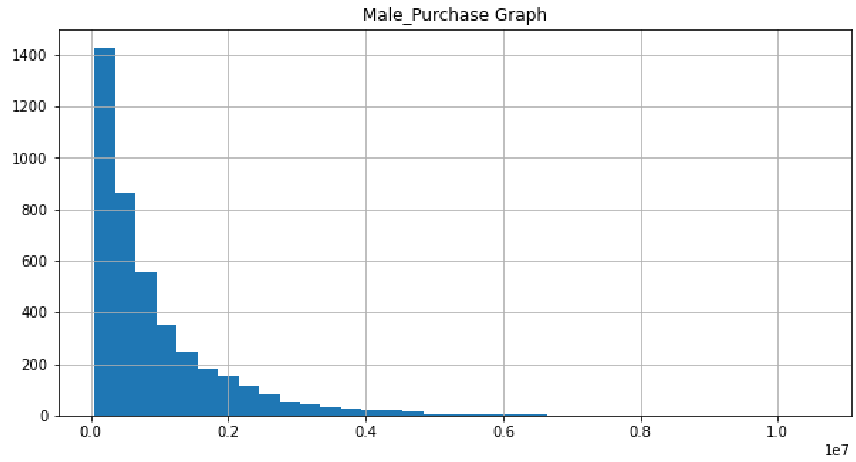
Out[7]:

```
M    4225
F    1666
Name: Gender, dtype: int64
```

In [8]:

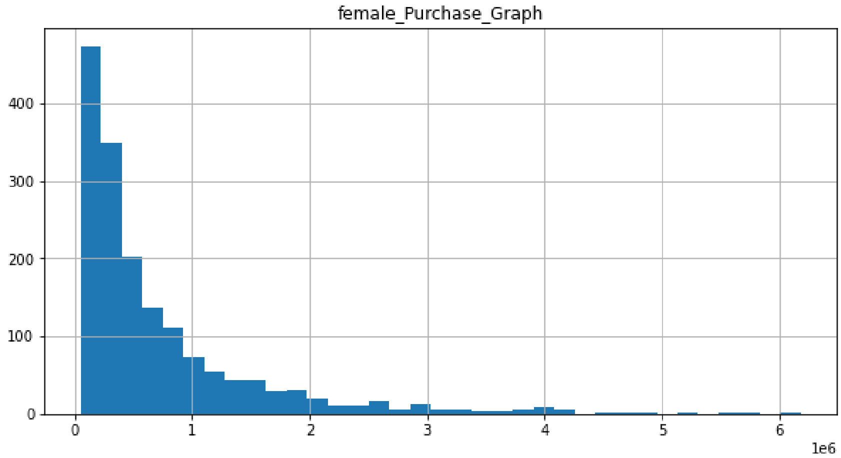
```

plt.figure(figsize=(10,5))
avg_df[avg_df['Gender']=='M']['Purchase'].hist(bins=35)
plt.title("Male_Purchase Graph")
plt.show()
```



In [9]:

```
plt.figure(figsize=(10,5))
avg_df[avg_df['Gender']=='F']['Purchase'].hist(bins=35)
plt.title("female_Purchase_Graph")
plt.show()
```



In [17]:

```
male_avg = avg_df[avg_df['Gender']=='M']['Purchase'].mean()
female_avg = avg_df[avg_df['Gender']=='F']['Purchase'].mean()
```

In [18]:

```
male_avg
```

Out[18]:

```
925344.4023668639
```

In [20]:

```
female_avg
```

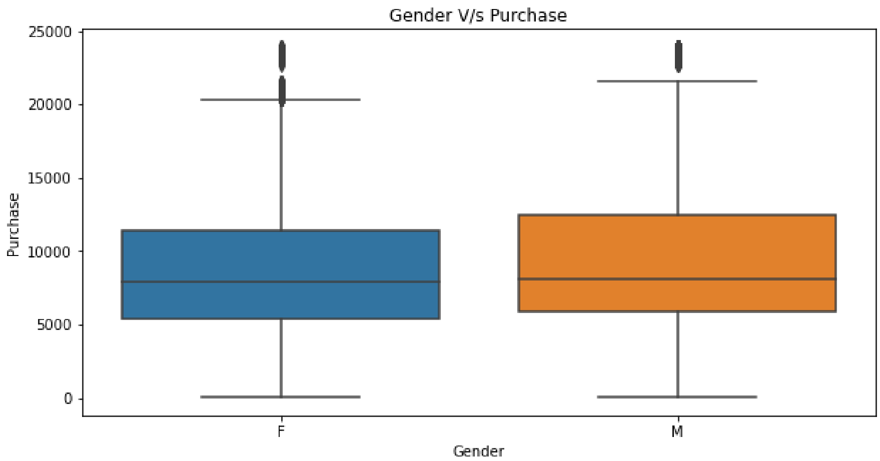
Out[20]:

```
712024.3949579832
```

Using Central limit theorem,confidence interval and percentile method For Gender V/s purchase

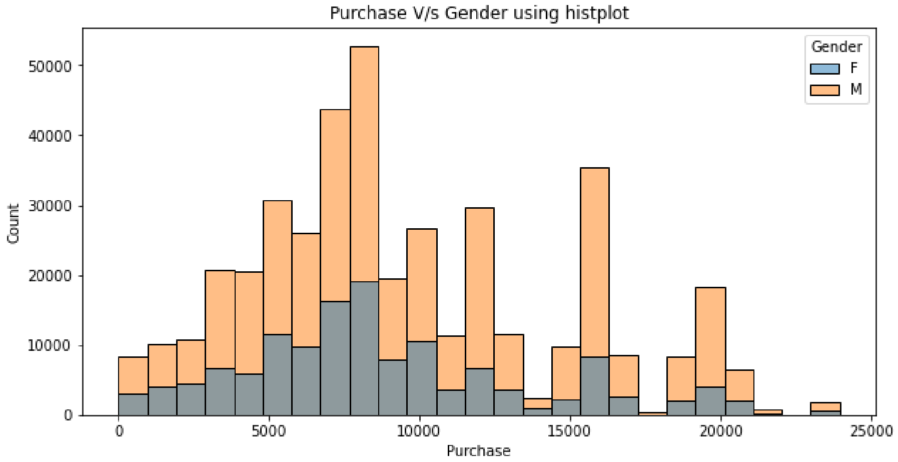
In [11]:

```
plt.figure(figsize=(10,5))
sns.boxplot(x="Gender", y="Purchase", data=df)
plt.title("Gender V/s Purchase")
plt.show()
```



In [12]:

```
plt.figure(figsize=(10,5))
sns.histplot(x="Purchase", hue="Gender", data=df, bins=25)
plt.title("Purchase V/s Gender using histplot")
plt.show()
```



In [6]:

```
# Central limit theorum
# 1: randoms sample
# df.sample(500)
# 2: mean
df.sample(500).groupby('Gender')['Purchase'].describe()
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
Gender								
F	114.0	8380.333333	4892.282529	771.0	5175.75	7766.0	10422.25	20869.0
M	386.0	9174.248705	5181.518070	13.0	5315.00	7982.5	12582.50	20682.0

In [16]:

```
df.sample(500).groupby('Gender')['Purchase'].describe()
```

Out[16]:

	count	mean	std	min	25%	50%	75%	max
Gender								
F	114.0	8924.219298	4654.131119	597.0	6004.00	7955.0	11785.25	22816.0
M	386.0	8848.142487	5192.158797	13.0	5232.25	7919.5	11957.25	23799.0

In [17]:

```
df.sample(500).groupby('Gender')['Purchase'].describe()
```

Out[17]:

	count	mean	std	min	25%	50%	75%	max
Gender								
F	126.0	9148.095238	4802.162255	761.0	6729.75	7962.5	11580.5	21079.0
M	374.0	9490.949198	5062.835497	37.0	5596.25	8131.0	13273.0	23518.0

In [7]:

```
sample_size = 500
iterations = 2000
```

In [8]:

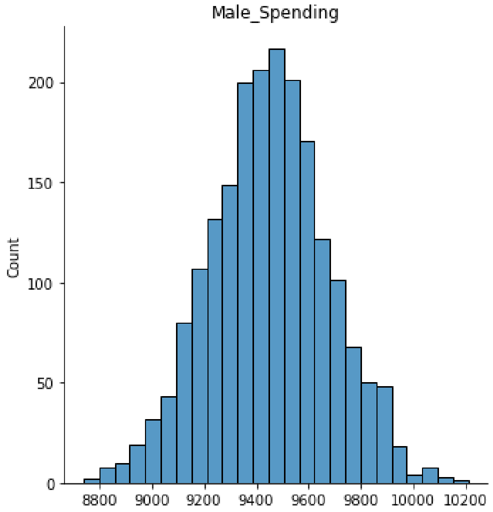
```
df_filtered = df[df.Gender=='M']
male_spends = []
for iter in range(iterations):
    male_spends.append(
        df_filtered.sample(sample_size)['Purchase'].mean()
    )
```

In [9]:

```
print(np.mean(male_spends))
plt.figure(figsize=(15,5))
sns.displot(x=male_spends, bins=25)
plt.title("Male_Spending")
plt.show()
```

9447.341247

<Figure size 1080x360 with 0 Axes>



In [10]:

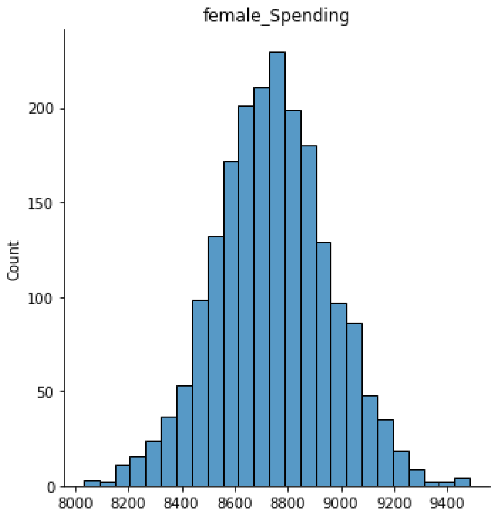
```
df_filtered = df[df.Gender=='F']
female_spends = []
for iter in range(iterations):
    female_spends.append(
        df_filtered.sample(sample_size)['Purchase'].mean()
    )
```

In [11]:

```
print(np.mean(female_spends))
plt.figure(figsize=(15,5))
sns.displot(x=female_spends, bins=25)
plt.title("female_Spending")
plt.show()
```

8741.050929

<Figure size 1080x360 with 0 Axes>



calculating 95%,90%,99% confidence level using z score method

In [18]:

```
# males with 95% confidence
min_male95 = np.mean(male_spends)-1.96*np.std(male_spends)
max_male95 = np.mean(male_spends)+1.96*np.std(male_spends)
print(min_male95, max_male95)
```

```
8991.606541124718 9882.603298875281
```

In [12]:

```
# males with 90% confidence
min_male90 = np.mean(male_spends)-1.64*np.std(male_spends)
max_male90 = np.mean(male_spends)+1.64*np.std(male_spends)
print(min_male90, max_male90)
```

```
9077.009387720389 9817.673106279612
```

In [13]:

```
# males with 99% confidence
min_male99 = np.mean(male_spends)-2.57*np.std(male_spends)
max_male99 = np.mean(male_spends)+2.57*np.std(male_spends)
print(min_male99, max_male99)
```

```
8867.00412605573 10027.67836794427
```

In [19]:

```
# females with 95% confidence
min_female95 = np.mean(female_spends)-1.96*np.std(female_spends)
max_female95 = np.mean(female_spends)+1.96*np.std(female_spends)
print(min_female95, max_female95)
```

```
8318.825830520895 9147.636075479106
```

In [23]:

```
# females with 90% confidence
min_female90 = np.mean(female_spends)-1.64*np.std(female_spends)
max_female90 = np.mean(female_spends)+1.64*np.std(female_spends)
print(min_female90, max_female90)
```

```
8389.368528685947 9092.733329314055
```

In [24]:

```
# females with 99% confidence
min_female99 = np.mean(female_spends)-2.57*np.std(female_spends)
max_female99 = np.mean(female_spends)+2.57*np.std(female_spends)
print(min_female99, max_female99)
```

```
8189.93887484932 9292.162983150682
```

Using percentile method

In [20]:

```
print(np.percentile(male_spends, [2.5, 97.5]))
print(np.percentile(female_spends, [2.5, 97.5]))
```

```
[8991.862  9899.2212]
[8311.52975 9138.17315]
```

Using Central limit theorem,confidence interval and percentile method For Marital_Status V/s purchase

In [22]:

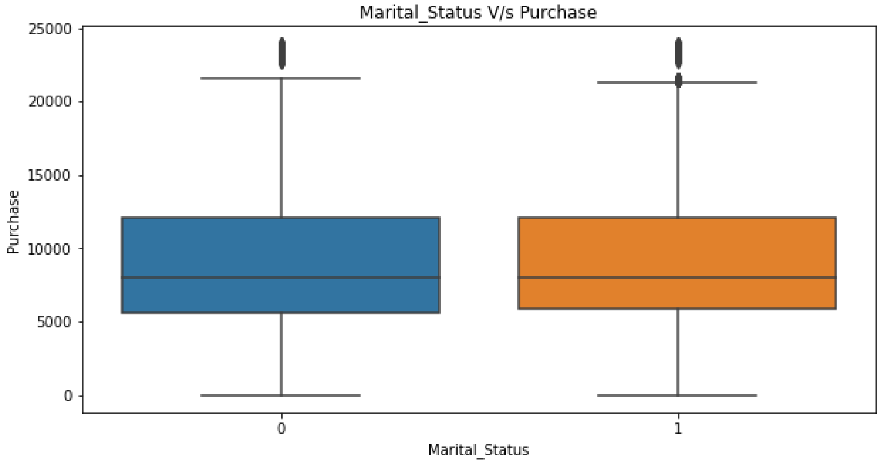
```
df.groupby('Marital_Status')['Purchase'].describe()
```

Out[22]:

Marital_Status	count	mean	std	min	25%	50%	75%	max
0	324731.0	9265.907619	5027.347859	12.0	5605.0	8044.0	12061.0	23961.0
1	225337.0	9261.174574	5016.897378	12.0	5843.0	8051.0	12042.0	23961.0

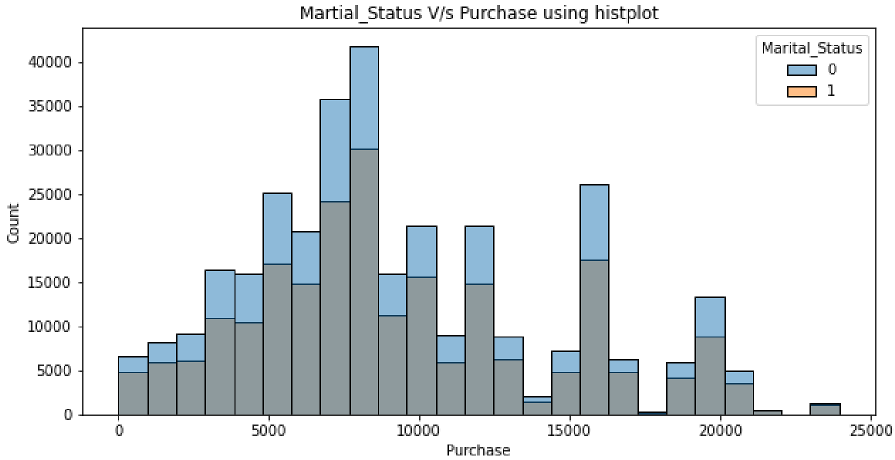
In [24]:

```
plt.figure(figsize=(10,5))
sns.boxplot(x="Marital_Status", y="Purchase", data=df)
plt.title("Marital_Status V/s Purchase")
plt.show()
```



In [6]:

```
plt.figure(figsize=(10,5))
sns.histplot(x="Purchase", hue="Marital_Status", data=df, bins=25)
plt.title("Martial_Status V/s Purchase using histplot")
plt.show()
```



Using CLT

In [27]:

```
df.sample(300).groupby('Marital_Status')['Purchase'].describe()
```

Out[27]:

Marital_Status	count	mean	std	min	25%	50%	75%	max
0	176.0	9119.301136	4817.005489	12.0	5843.25	8038.5	11909.25	20848.0
1	124.0	9409.411290	5174.153880	26.0	6027.75	7945.0	12064.00	23454.0

In [28]:

```
df.sample(300).groupby('Marital_Status')['Purchase'].describe()
```

Out[28]:

Marital_Status	count	mean	std	min	25%	50%	75%	max
0	170.0	9068.470588	4688.778471	494.0	5901.0	8019.5	11634.25	23652.0
1	130.0	9578.023077	5154.299337	758.0	5936.0	8613.0	12452.75	21063.0

In [29]:

```
df.sample(300).groupby('Marital_Status')['Purchase'].describe()
```

Out[29]:

Marital_Status	count	mean	std	min	25%	50%	75%	max
0	181.0	9541.729282	5241.760655	478.0	5826.0	8124.0	12705.0	23897.0
1	119.0	8848.109244	4622.453464	126.0	5645.5	8058.0	10769.0	23842.0

In [14]:

```
sample_size = 300
iterations = 1000
```

In [15]:

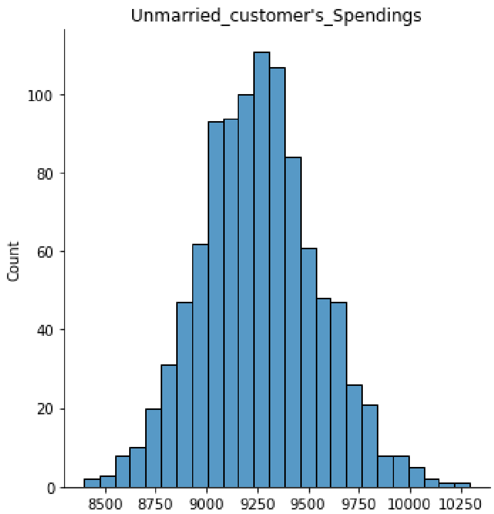
```
df_unmarried = df[df.Marital_Status==0]
unmarried_spends = []
for iter in range(iterations):
    unmarried_spends.append(
        df_unmarried.sample(sample_size)['Purchase'].mean()
    )
```

In [16]:

```
print(np.mean(unmarried_spends))
plt.figure(figsize=(10,5))
sns.displot(x=unmarried_spends, bins=25)
plt.title("Unmarried_customer's_Spendings")
plt.show()
```

9259.012903333332

<Figure size 720x360 with 0 Axes>



In [17]:

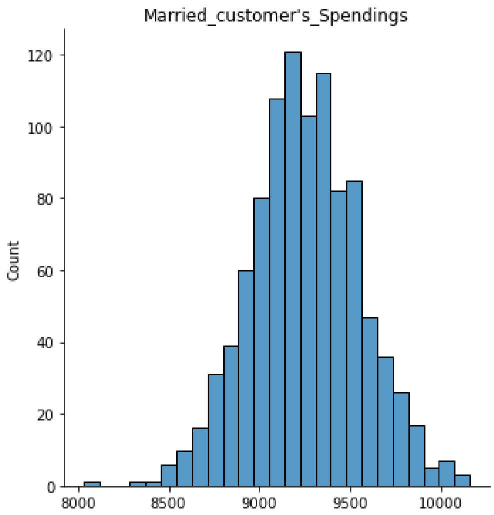
```
df_married = df[df.Marital_Status==1]
married_spends = []
for iter in range(iterations):
    married_spends.append(
        df_married.sample(sample_size)['Purchase'].mean()
    )
```

In [18]:

```
print(np.mean(married_spends))
plt.figure(figsize=(10,5))
sns.displot(x=married_spends, bins=25)
plt.title("Married_customer's_Spendings")
plt.show()
```

```
9251.073886666667

<Figure size 720x360 with 0 Axes>
```



Using Confidence Interval of 90%,95%,99%

In [48]:

```
# Unmarried 90% CI
min_unmarried90 = np.mean(unmarried_spends)-1.64*np.std(unmarried_spends)
max_unmarried90 = np.mean(unmarried_spends)+1.64*np.std(unmarried_spends)
print(min_unmarried90, max_unmarried90)
```

```
8898.507726661444 9623.949763338556
```

In [19]:

```
# Unmarried 95% CI
min_unmarried95 = np.mean(unmarried_spends)-1.96*np.std(unmarried_spends)
max_unmarried95 = np.mean(unmarried_spends)+1.96*np.std(unmarried_spends)
print(min_unmarried95, max_unmarried95)
```

```
8692.461829242036 9825.563977424628
```

In [20]:

```
# Unmarried 99% CI
min_unmarried99 = np.mean(unmarried_spends)-2.57*np.std(unmarried_spends)
max_unmarried99 = np.mean(unmarried_spends)+2.57*np.std(unmarried_spends)
print(min_unmarried99, max_unmarried99)
```

```
8516.137260264644 10001.88854640202
```

In [49]:

```
# married 90% CI
min_married90 = np.mean(married_spends)-1.64*np.std(married_spends)
max_married90 = np.mean(married_spends)+1.64*np.std(married_spends)
print(min_married90, max_married90)
```

```
8790.237789848248 9748.80057681842
```

In [21]:

```
# married 95% CI
min_married95 = np.mean(married_spends)-1.96*np.std(married_spends)
max_married95 = np.mean(married_spends)+1.96*np.std(married_spends)
print(min_married95, max_married95)
```

```
8664.488109143509 9837.659664189825
```

In [22]:

```
# married 99% CI
min_married99 = np.mean(married_spends)-2.57*np.std(married_spends)
max_married99 = np.mean(married_spends)+2.57*np.std(married_spends)
print(min_married99, max_married99)
```

```
8481.928249812321 10020.219523521013
```

using percentile method

In [50]:

```
print(np.percentile(unmarried_spends, [5, 95]))
print(np.percentile(married_spends, [5, 95]))
```

```
[8893.6095 9629.2085]
[8788.45733333 9759.5765    ]
```

Using Central limit theorem,confidence interval and percentile method For Age V/s purchase

In [51]:

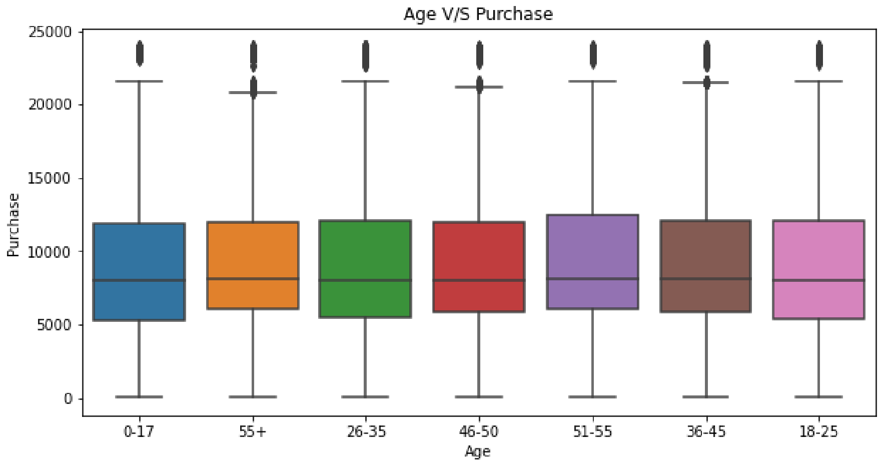
```
df.groupby('Age')['Purchase'].describe()
```

Out[51]:

Age	count	mean	std	min	25%	50%	75%	max
0-17	15102.0	8933.464640	5111.114046	12.0	5328.0	7986.0	11874.0	23955.0
18-25	99660.0	9169.663606	5034.321997	12.0	5415.0	8027.0	12028.0	23958.0
26-35	219587.0	9252.690633	5010.527303	12.0	5475.0	8030.0	12047.0	23961.0
36-45	110013.0	9331.350695	5022.923879	12.0	5876.0	8061.0	12107.0	23960.0
46-50	45701.0	9208.625697	4967.216367	12.0	5888.0	8036.0	11997.0	23960.0
51-55	38501.0	9534.808031	5087.368080	12.0	6017.0	8130.0	12462.0	23960.0
55+	21504.0	9336.280459	5011.493996	12.0	6018.0	8105.5	11932.0	23960.0

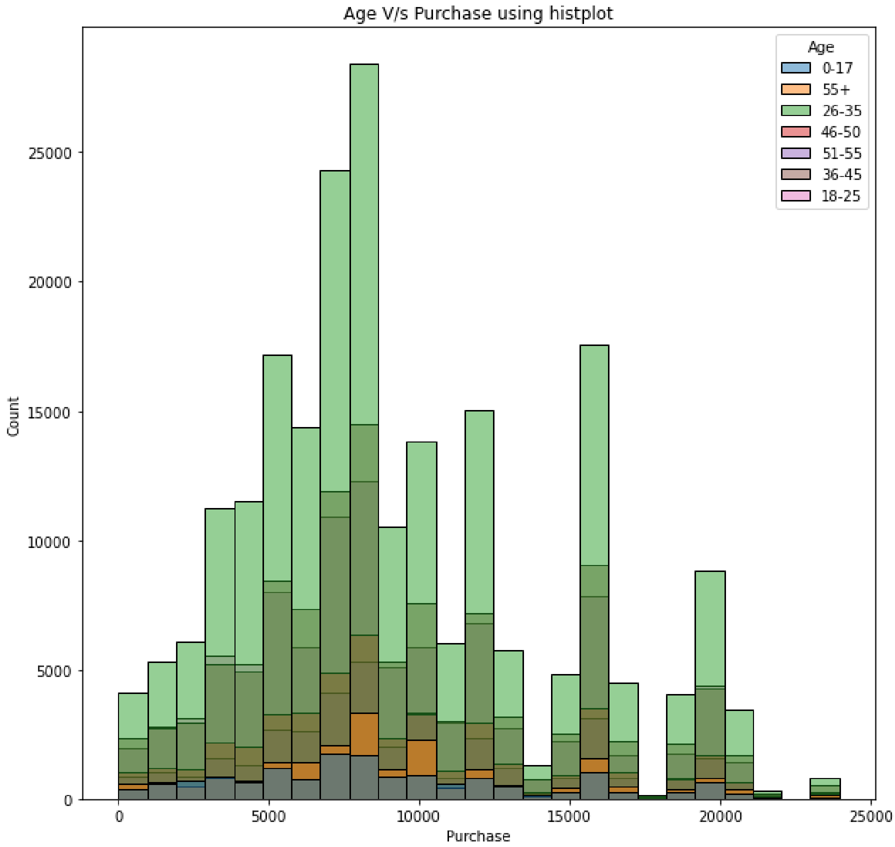
In [35]:

```
plt.figure(figsize=(10,5))
sns.boxplot(x='Age',y ='Purchase',data=df)
plt.title("Age V/S Purchase")
plt.show()
```



In [9]:

```
plt.figure(figsize=(10,10))
sns.histplot(x="Purchase", hue="Age", data=df, bins=25)
plt.title("Age V/s Purchase using histplot")
plt.show()
```



In [6]:

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

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

Out[6]:

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

In [25]:

```
sample= 200
iterations = 1000
sample_size = 200
num_repitions = 1000
```

In [26]:

```
Age_spends1 = [df[df.Age=='0-17'].sample(sample_size)['Purchase'].mean() for iter in range(iterations)]
```

In [27]:

```
print(np.mean(Age_spends1))
```

```
8919.74509
```

In [36]:

```
Age_spends2 = [df[df.Age=='18-25'].sample(sample_size)['Purchase'].mean() for iter in range(iterations)]
print(np.mean(Age_spends2))
```

9146.15592

In [28]:

```

Age_spends3 = [df[df.Age=='26-35'].sample(sample_size)['Purchase'].mean() for iter in range(iterations)]
print(np.mean(Age_spends3))
```

9275.487395

In [29]:

```
Age_spends4 = [df[df.Age=='36-45'].sample(sample_size)['Purchase'].mean() for iter in range(iterations)]
print(np.mean(Age_spends4))

```

9322.28648

In [30]:

```
Age_spends5 = [df[df.Age=='46-50'].sample(sample_size)['Purchase'].mean() for iter in range(iterations)]
print(np.mean(Age_spends5))
```

9213.220765

In [31]:

```
Age_spends6 = [df[df.Age=='51-55'].sample(sample_size)['Purchase'].mean() for iter in range(iterations)]
print(np.mean(Age_spends6))
```

9533.1371

In [32]:

```
Age_spends7 = [df[df.Age=='55+'].sample(sample_size)['Purchase'].mean() for iter in range(iterations)]
print(np.mean(Age_spends7))
```

9339.12391

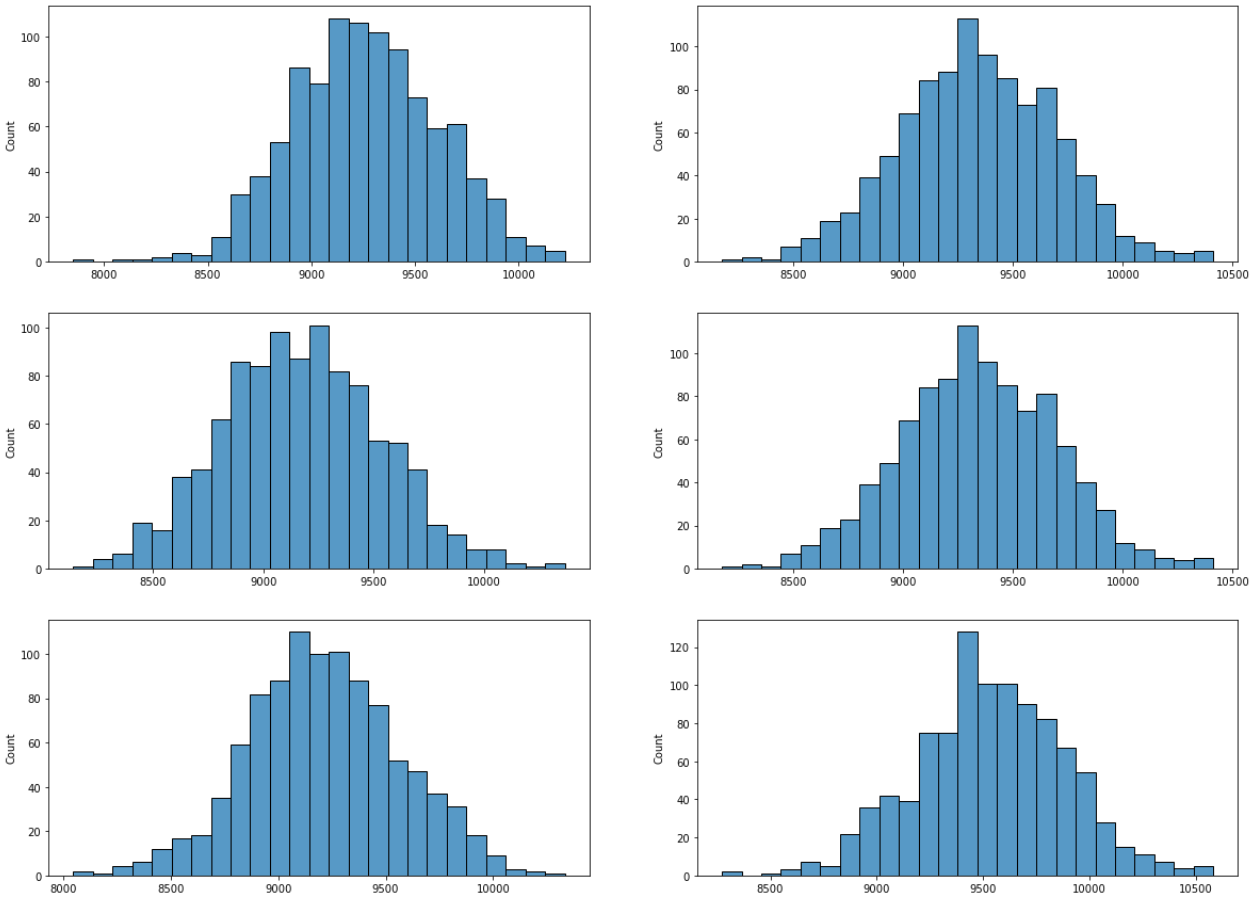
In [31]:

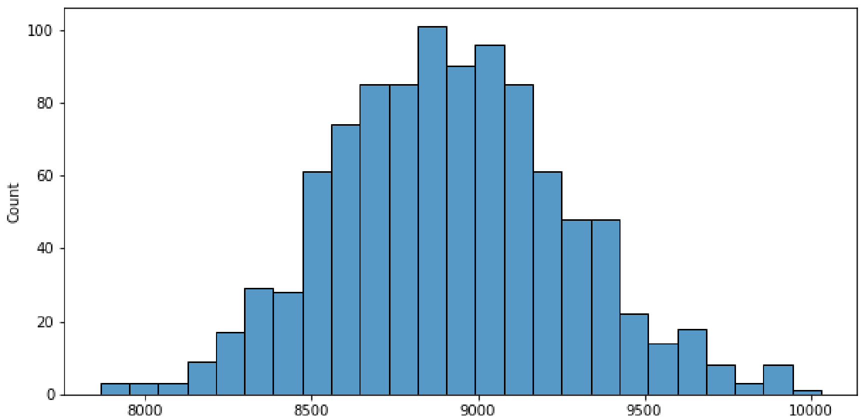
```
fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(20, 15))

sns.histplot(Age_spends3,bins=25,ax=axis[0,0])
sns.histplot(Age_spends4,bins=25,ax=axis[0,1])
sns.histplot(Age_spends2,bins=25,ax=axis[1,0])
sns.histplot(Age_spends4,bins=25,ax=axis[1,1])
sns.histplot(Age_spends5,bins=25,ax=axis[2,0])
sns.histplot(Age_spends6,bins=25,ax=axis[2,1])

plt.show()

plt.figure(figsize=(10, 5))
sns.histplot(Age_spends1,bins=25)
plt.show()
```





In [ ]:

```
#Calculating 90%,95%,99% confidence interval for age_expenses for different age groups for sample size 200:
```

In [32]:

```
#Age_range[0-17] 90% CI
min_Age_spends1_90 = np.mean(Age_spends1)-1.64*np.std(Age_spends1)
max_Age_spends1_90 = np.mean(Age_spends1)+1.64*np.std(Age_spends1)
print(min_Age_spends1_90, max_Age_spends1_90)
```

8336.430256640017 9495.894873359985

In [33]:

```
#Age_range[18-25] 90% CI
min_Age_spends2_90 = np.mean(Age_spends2)-1.64*np.std(Age_spends2)
max_Age_spends2_90 = np.mean(Age_spends2)+1.64*np.std(Age_spends2)
print(min_Age_spends2_90, max_Age_spends2_90)
```

8585.555028848848 9759.16752115115

In [34]:

```
#Age_range[26-35] 90% CI
min_Age_spends3_90 = np.mean(Age_spends3)-1.64*np.std(Age_spends3)
max_Age_spends3_90 = np.mean(Age_spends3)+1.64*np.std(Age_spends3)
print(min_Age_spends3_90, max_Age_spends3_90)
```

8683.868115567524 9838.044564432477

In [35]:

```
#Age_range[36-45] 90% CI
min_Age_spends4_90 = np.mean(Age_spends4)-1.64*np.std(Age_spends4)
max_Age_spends4_90 = np.mean(Age_spends4)+1.64*np.std(Age_spends4)
print(min_Age_spends4_90, max_Age_spends4_90)
```

8757.907794061615 9923.144805938387

In [36]:

```
#Age_range[46-50] 90% CI
min_Age_spends5_90 = np.mean(Age_spends5)-1.64*np.std(Age_spends5)
max_Age_spends5_90 = np.mean(Age_spends5)+1.64*np.std(Age_spends5)
print(min_Age_spends5_90, max_Age_spends5_90)
```

8634.011188693168 9795.179741306833

In [37]:

```
#Age_range[51-55] 90% CI
min_Age_spends6_90 = np.mean(Age_spends6)-1.64*np.std(Age_spends6)
max_Age_spends6_90 = np.mean(Age_spends6)+1.64*np.std(Age_spends6)
print(min_Age_spends6_90, max_Age_spends6_90)
```

8959.603319484982 10112.890460515018

In [38]:

```
#Age_range[55+] 90% CI
min_Age_spends7_90 = np.mean(Age_spends7)-1.64*np.std(Age_spends7)
max_Age_spends7_90 = np.mean(Age_spends7)+1.64*np.std(Age_spends7)
print(min_Age_spends7_90, max_Age_spends7_90)
```

8746.522202041026 9916.895107958975

In [33]:

```
#Age_range[0-17] 95% CI
min_Age_spends1_95 = np.mean(Age_spends1)-1.96*np.std(Age_spends1)
max_Age_spends1_95 = np.mean(Age_spends1)+1.96*np.std(Age_spends1)
print(min_Age_spends1_95, max_Age_spends1_95)
```

8223.177602722288 9616.312577277713

In [37]:

```
#Age_range[18-25] 95% CI
min_Age_spends2_95 = np.mean(Age_spends2)-1.96*np.std(Age_spends2)
max_Age_spends2_95 = np.mean(Age_spends2)+1.96*np.std(Age_spends2)
print(min_Age_spends2_95, max_Age_spends2_95)
```

8456.151802338614 9836.160037661384

In [35]:

```
#Age_range[26-35] 95% CI
min_Age_spends3_95 = np.mean(Age_spends3)-1.96*np.std(Age_spends3)
max_Age_spends3_95 = np.mean(Age_spends3)+1.96*np.std(Age_spends3)
print(min_Age_spends3_95, max_Age_spends3_95)
```

8589.407922559634 9961.566867440366

In [38]:

```
#Age_range[36-45] 95% CI
min_Age_spends4_95 = np.mean(Age_spends4)-1.96*np.std(Age_spends4)
max_Age_spends4_95 = np.mean(Age_spends4)+1.96*np.std(Age_spends4)
print(min_Age_spends4_95, max_Age_spends4_95)
```

8634.263502678414 10010.309457321588

In [39]:

```
#Age_range[46-50] 95% CI
min_Age_spends5_95 = np.mean(Age_spends5)-1.96*np.std(Age_spends5)
max_Age_spends5_95 = np.mean(Age_spends5)+1.96*np.std(Age_spends5)
print(min_Age_spends5_95, max_Age_spends5_95)
```

8513.967582468344 9912.473947531656

In [40]:

```
#Age_range[51-55] 95% CI
min_Age_spends6_95 = np.mean(Age_spends6)-1.96*np.std(Age_spends6)
max_Age_spends6_95 = np.mean(Age_spends6)+1.96*np.std(Age_spends6)
print(min_Age_spends6_95, max_Age_spends6_95)
```

8807.584433300079 10258.689766699921

In [41]:

```
#Age_range[55+] 95% CI
min_Age_spends7_95 = np.mean(Age_spends7)-1.96*np.std(Age_spends7)
max_Age_spends7_95 = np.mean(Age_spends7)+1.96*np.std(Age_spends7)
print(min_Age_spends7_95, max_Age_spends7_95)
```

8647.068020448763 10031.179799551237

In [42]:

```
#Age_range[0-17] 99% CI
min_Age_spends1_99 = np.mean(Age_spends1)-2.57*np.std(Age_spends1)
max_Age_spends1_99 = np.mean(Age_spends1)+2.57*np.std(Age_spends1)
print(min_Age_spends1_99, max_Age_spends1_99)
```

8006.388741885856 9833.101438114145

In [43]:

```
#Age_range[18-25] 99% CI
min_Age_spends2_99 = np.mean(Age_spends2)-2.57*np.std(Age_spends2)
max_Age_spends2_99 = np.mean(Age_spends2)+2.57*np.std(Age_spends2)
print(min_Age_spends2_99, max_Age_spends2_99)
```

8241.405622862367 10050.906217137632

In [44]:

```
#Age_range[26-35] 99% CI
min_Age_spends3_99 = np.mean(Age_spends3)-2.57*np.std(Age_spends3)
max_Age_spends3_99 = np.mean(Age_spends3)+2.57*np.std(Age_spends3)
print(min_Age_spends3_99, max_Age_spends3_99)
```

8375.883188789929 10175.091601210072

In [45]:

```
#Age_range[36-45] 99% CI
min_Age_spends4_99 = np.mean(Age_spends4)-2.57*np.std(Age_spends4)
max_Age_spends4_99 = np.mean(Age_spends4)+2.57*np.std(Age_spends4)
print(min_Age_spends4_99, max_Age_spends4_99)
```

8420.133902593634 10224.439057406367

In [46]:

```
#Age_range[46-50] 99% CI
min_Age_spends5_99 = np.mean(Age_spends5)-2.57*np.std(Age_spends5)
max_Age_spends5_99 = np.mean(Age_spends5)+2.57*np.std(Age_spends5)
print(min_Age_spends5_99, max_Age_spends5_99)
```

```
8296.342867496756 10130.098662503244
```

In [47]:

```
#Age_range[51-55] 99% CI
min_Age_spends6_99 = np.mean(Age_spends6)-2.57*np.std(Age_spends6)
max_Age_spends6_99 = np.mean(Age_spends6)+2.57*np.std(Age_spends6)
print(min_Age_spends6_99, max_Age_spends6_99)
```

```
8581.774674786328 10484.499525213672
```

In [48]:

```
#Age_range[55+] 99% CI
min_Age_spends7_99 = np.mean(Age_spends7)-2.57*np.std(Age_spends7)
max_Age_spends7_99 = np.mean(Age_spends7)+2.57*np.std(Age_spends7)
print(min_Age_spends7_99, max_Age_spends7_99)
```

```
8431.68327931292 10246.56454068708
```

In [ ]:

```
# using percentile method
```

In [41]:

```
print(np.percentile(Age_spends1, [5, 95]))
print(np.percentile(Age_spends2, [5, 95]))
print(np.percentile(Age_spends3, [5, 95]))
print(np.percentile(Age_spends4, [5, 95]))
print(np.percentile(Age_spends5, [5, 95]))
print(np.percentile(Age_spends6, [5, 95]))
print(np.percentile(Age_spends7, [5, 95]))
```

```
[8346.773  9528.2275]
[8610.04725 9753.50725]
[8696.19425 9845.408  ]
[8754.0075  9911.70475]
[8638.7675 9816.343 ]
[ 8949.63675 10100.34   ]
[8730.78275 9918.04075]
```

In [ ]:

```

```