In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import binom
from scipy.stats import norm
from scipy.stats import poisson
from scipy.stats import geom
from scipy.stats import expon
from scipy.stats import t
```

In [2]:
```
#Importing Walmart dataset and reading the first 5 records
df=pd.read_csv('walmart_data.csv')
df.head()
```

Out[2]:

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

In [3]:
```
#Checking basics
df.shape
#We have a dataframe consisting of half a million observations and 10 features
```

Out[3]:
```
(550068, 10)
```

In [4]:
```
#Getting mean,count,std_dev etc.
df.describe()
```

Out[4]:

	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 [5]:
```
df.isnull().sum()
#there are no null values in any columns so data is relatively clean
```

Out[5]:
```
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
```

In [6]:
```
#User_ID,Product_ID,Gender,Age,Occupation,City_Category,Stay_In_Current_City_Years,
print('Unique Ages --> ',df.Age.unique())
```

```
Unique Ages -->  ['0-17' '55+' '26-35' '46-50' '51-55' '36-45' '18-25']
```

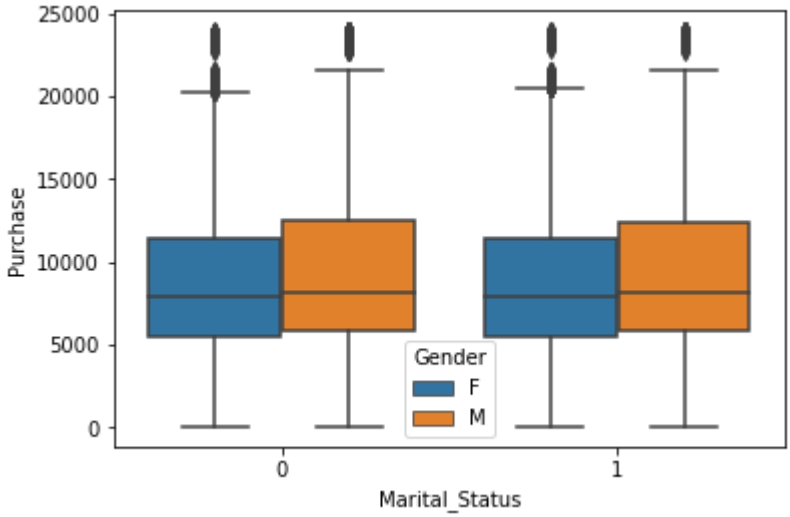
In [7]:
```
print('Stays in current city -->',df.Stay_In_Current_City_Years.unique())
```

```
Stays in current city --> ['2' '4+' '3' '1' '0']
```

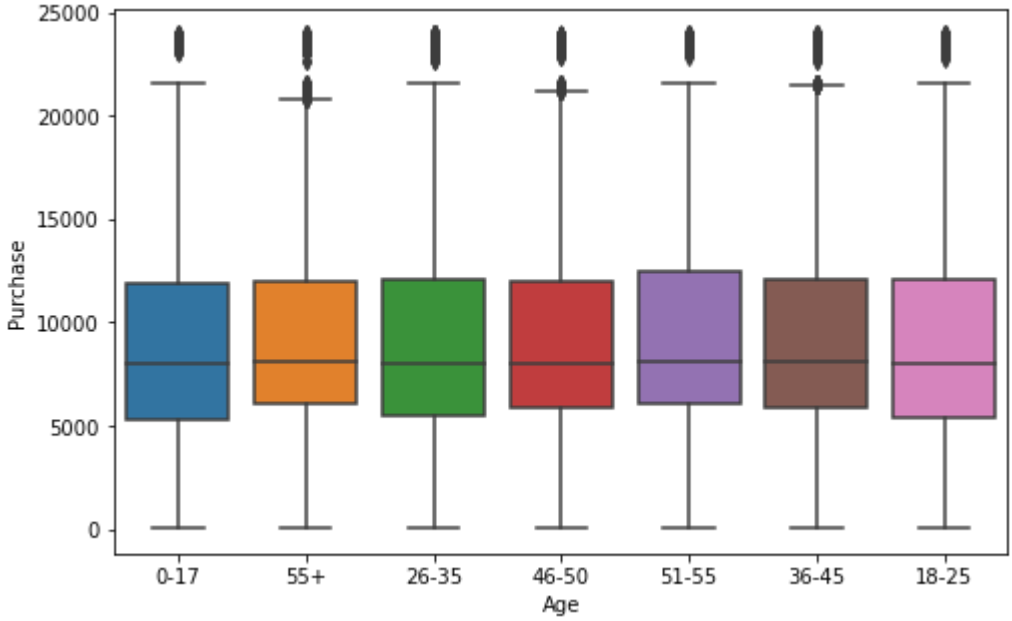
In [8]:
```
df.Occupation.unique() #masked
```

Out[8]:
```
array([10, 16, 15,  7, 20,  9,  1, 12, 17,  0,  3,  4, 11,  8, 19,  2, 18,
        5, 14, 13,  6], dtype=int64)
```

In [9]:
```
#User_ID,Product_ID,Gender,Age,Occupation,City_Category,Stay_In_Current_City_Years,
sns.boxplot(x='Marital_Status', y='Purchase', data=df, hue='Gender')
plt.show()
```



In [10]:
```
fig, ax = plt.subplots(figsize=(8, 5))
sns.boxplot(x='Age', y='Purchase', data=df, ax=ax)
plt.show()
```



In [11]:
```
#User_ID,Product_ID,Gender,Age,Occupation,City_Category,Stay_In_Current_City_Years,
print('No. of unique people who shopped --> ',len(df['User_ID'].unique()),'--> M:',
```

```
No. of unique people who shopped -->  5891 --> M: 4225 F: 1666
```

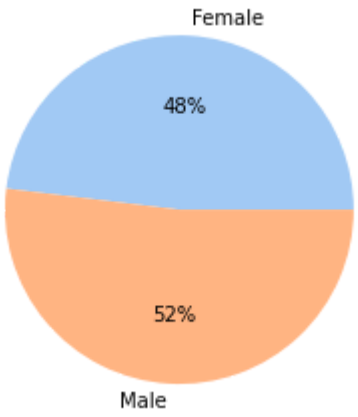
In [12]:
```
df.groupby(by=['Gender', 'User_ID']).sum()['Purchase']
```

Out[12]:
```
Gender  User_ID
F       1000001     334093
        1000006     379930
        1000010    2169510
        1000011     557023
        1000016     150490
                       ...
M       1006030     737361
        1006032     517261
        1006033     501843
        1006034     197086
        1006040    1653299
Name: Purchase, Length: 5891, dtype: int64
```

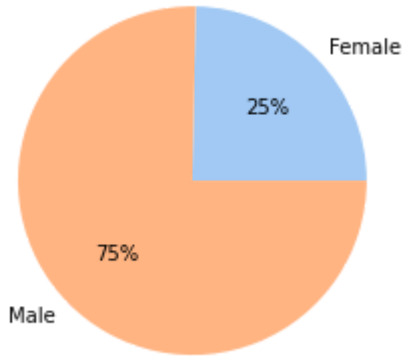
In [13]:
```
df.groupby(by=['Gender']).mean()['Purchase']
```

Out[13]:
```
Gender
F    8734.565765
M    9437.526040
Name: Purchase, dtype: float64
```

In [14]:
```
data = df.groupby(by=['Gender']).mean()['Purchase']
labels = ['Female', 'Male']
colors = sns.color_palette('pastel')[0:2]
plt.pie(data, labels = labels, colors = colors, autopct='%.0f%%')
plt.show()
```



In [15]:
```
data = df.groupby(by=['Gender']).count()['User_ID']
labels = ['Female', 'Male']
colors = sns.color_palette('pastel')[0:2]
plt.pie(data, labels = labels, colors = colors, autopct='%.0f%%')
plt.show()
```



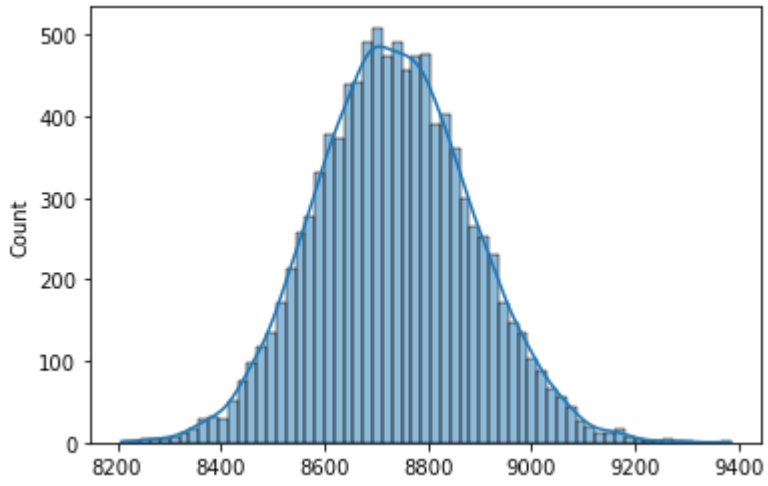
In [29]:
```
#Bootstrapping and CLT

data = df.loc[:,['Gender','Purchase']]
data_f = data[data['Gender']=='F']
data_m = data[data['Gender']=='M']
#print(data_f.shape[0],data_m.shape[0])
print('actual mean of female purchases -->',np.mean(data_f['Purchase']))
print('actual mean of male purchases -->',np.mean(data_m['Purchase']))

bootstrapped_samples_f = np.random.choice(data_f['Purchase'], size=1000)
bootstrapped_samples_f
print('mean of 1000 sample(would keep on changing as sample would change) -->', np.
```

```
actual mean of female purchases --> 8734.565765155476
actual mean of male purchases --> 9437.526040472265
mean of 1000 sample(would keep on changing as sample would change) --> 8878.536
```

In [30]:
```
bootstrapped_samples_f_1 = []
for x in range(10000):
    bootstrapped_samples_f = np.random.choice(data_f['Purchase'], size=1000)
    bootstrapped_samples_f_mean = np.mean(bootstrapped_samples_f)
    bootstrapped_samples_f_1.append(bootstrapped_samples_f_mean)
```

In [31]:
```
sns.histplot(bootstrapped_samples_f_1, kde=True)
```

Out[31]:
```
<AxesSubplot:ylabel='Count'>
```



In [32]:
```
print(np.mean(bootstrapped_samples_f_1))
print(np.std(bootstrapped_samples_f_1))
```

```
8734.4982734
150.89333037497732
```

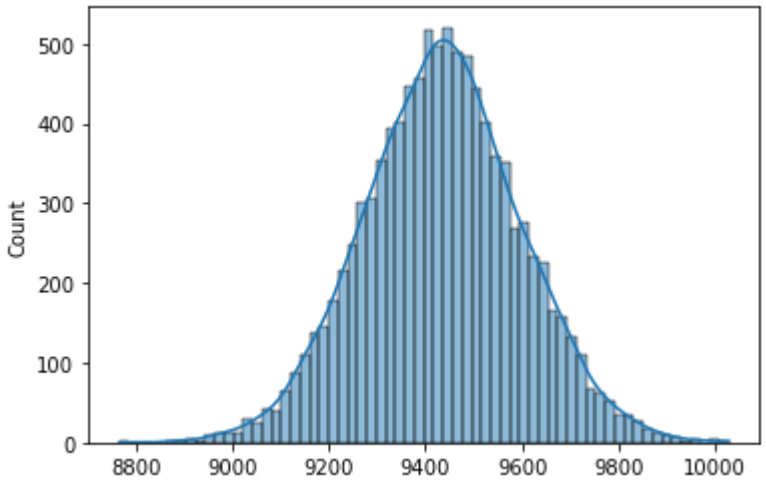
In [33]:
```
left = np.percentile(bootstrapped_samples_f_1,2.5)
right = np.percentile(bootstrapped_samples_f_1,97.5)
print(f'With 95% confidence we can say that the avg female purchase lies in between
```

```
With 95% confidence we can say that the avg female purchase lies in between 8446.8
07550000001 and 9037.289025
```

In [34]:
```
bootstrapped_samples_m_1 = []
for x in range(10000):
    bootstrapped_samples_m = np.random.choice(data_m['Purchase'], size=1000)
    bootstrapped_samples_m_mean = np.mean(bootstrapped_samples_m)
    bootstrapped_samples_m_1.append(bootstrapped_samples_m_mean)
```

In [35]:
```
sns.histplot(bootstrapped_samples_m_1, kde=True)
```

Out[35]:
```
<AxesSubplot:ylabel='Count'>
```



In [36]:
```
print(np.mean(bootstrapped_samples_m_1))
print(np.std(bootstrapped_samples_m_1))
```

```
9434.5686802
162.57381773326273
```

In [37]:
```
left = np.percentile(bootstrapped_samples_m_1,2.5)
right = np.percentile(bootstrapped_samples_m_1,97.5)
print(f'With 95% confidence we can say that the avg male purchase lies in between {
```

```
With 95% confidence we can say that the avg male purchase lies in between 9117.807
45 and 9756.59055
```

In [38]:
```
#As we can clearly see these are both non-overlapping intervals for male and female
```

In [40]:
```
#Bootstrapping and CLT

data = df.loc[:,['Marital_Status','Purchase']]
data_m0 = data[data['Marital_Status']==0]
```

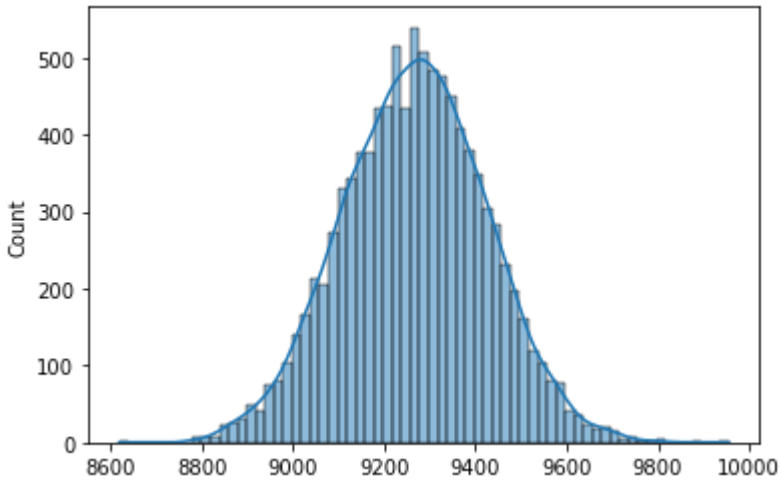
```
data_m1 = data[data['Marital_Status']==1]
#print(data_f.shape[0],data_m.shape[0])
print('actual mean of unmarried purchases -->',np.mean(data_m0['Purchase']))
print('actual mean of married couple purchases -->',np.mean(data_m1['Purchase']))

bootstrapped_samples_m0 = np.random.choice(data_m0['Purchase'], size=1000)
bootstrapped_samples_m0
print('mean of 1000 sample(would keep on changing as sample would change) -->', np.
```

```
actual mean of unmarried purchases --> 9265.907618921507
actual mean of married couple purchases --> 9261.174574082374
mean of 1000 sample(would keep on changing as sample would change) --> 9265.34
```

In [41]:
```
bootstrapped_samples_m0_1 = []
for x in range(10000):
    bootstrapped_samples_m0 = np.random.choice(data_m0['Purchase'], size=1000)
    bootstrapped_samples_m0_mean = np.mean(bootstrapped_samples_m0)
    bootstrapped_samples_m0_1.append(bootstrapped_samples_m0_mean)
```

In [42]:
```
sns.histplot(bootstrapped_samples_m0_1, kde=True)
```

Out[42]:
```
<AxesSubplot:ylabel='Count'>
```



In [43]:
```
print(np.mean(bootstrapped_samples_m0_1))
print(np.std(bootstrapped_samples_m0_1))
```

```
9266.4118685
158.2635486836606
```

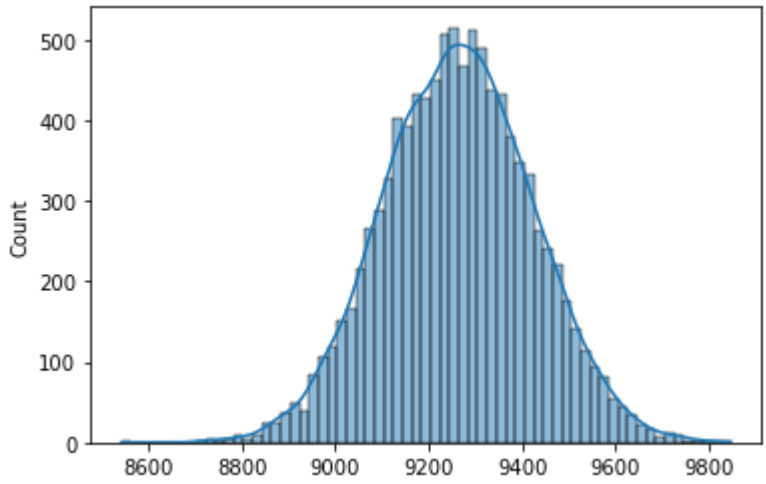
In [44]:
```
left = np.percentile(bootstrapped_samples_m0_1,2.5)
right = np.percentile(bootstrapped_samples_m0_1,97.5)
print(f'With 95% confidence we can say that the avg unmarried people purchase lies
```

```
With 95% confidence we can say that the avg unmarried people purchase lies in betw
een 8953.926475 and 9575.759325
```

In [46]:
```
bootstrapped_samples_m1_1 = []
for x in range(10000):
    bootstrapped_samples_m1 = np.random.choice(data_m1['Purchase'], size=1000)
    bootstrapped_samples_m1_mean = np.mean(bootstrapped_samples_m1)
    bootstrapped_samples_m1_1.append(bootstrapped_samples_m1_mean)
```

In [47]:
```
sns.histplot(bootstrapped_samples_m1_1, kde=True)
```

Out[47]:
```
<AxesSubplot:ylabel='Count'>
```



In [48]:
```
print(np.mean(bootstrapped_samples_m1_1))
print(np.std(bootstrapped_samples_m1_1))
```

```
9262.595206699998
159.93820206930417
```

In [49]:
```
left = np.percentile(bootstrapped_samples_m1_1,2.5)
right = np.percentile(bootstrapped_samples_m1_1,97.5)
print(f'With 95% confidence we can say that the avg married couple purchase lies in
```

```
With 95% confidence we can say that the avg married couple purchase lies in betwee
n 8953.6237 and 9576.389425000001
```

In [50]:
```
#So we can see, here the confidence intervals are overlapping which means there is
# demarcation that people whether married or unmarried would spend more
```