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

In [1]:
```
```

In [2]:
```
data = pd.read_csv("walmart_data.csv")
```

In [4]:
```
#data types of all the columns inthe table
data.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
```

In [5]:
```
#null value check for all columns
data.isnull().sum()
```

Out[5]:

	0
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]:

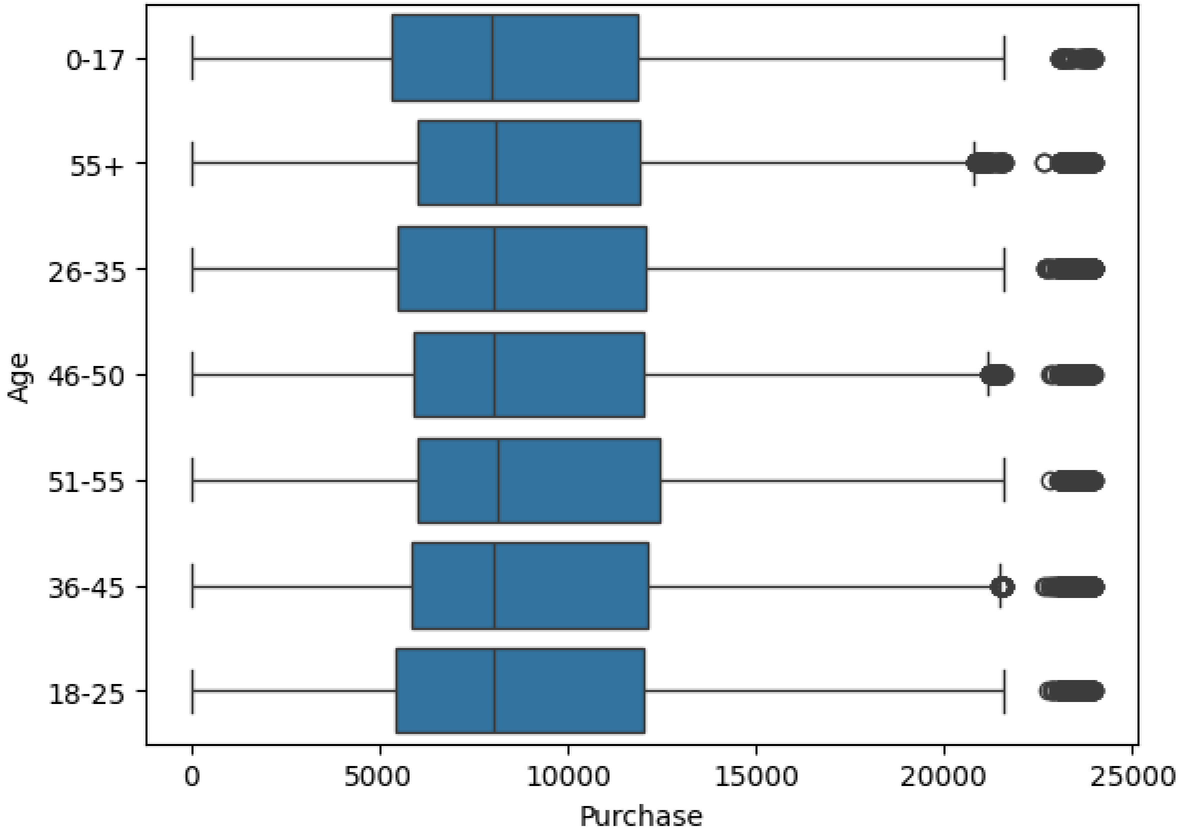
```
#nmber of rows and columns in the table
data.shape
```

Out[6]: (550068, 10)

In [7]:

```
#Outliers for continous variables -- age, Stay_In_Current_City_Years and Purchase
sns.boxplot(x='Purchase',y="Age", data=data)
```

Out[7]: <Axes: xlabel='Purchase', ylabel='Age'>



In [8]:
```
# Descrbing the continous Variables
data['Age'].describe()
```

Out[8]:

	Age
count	550068
unique	7
top	26-35
freq	219587

dtype: object

In [9]:
```
data['Stay_In_Current_City_Years'].describe()
```

Out[9]:

	Stay_In_Current_City_Years
count	550068
unique	5
top	1
freq	193821

dtype: object

In [10]:
```
data['Stay_In_Current_City_Years'].unique()
```

Out[10]:
```
array(['2', '4+', '3', '1', '0'], dtype=object)
```

In [11]:
```
data['Purchase'].describe()
```

Out[11]:

	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

dtype: float64

In [12]:
```
#data categorisation for male and female customers
data_female = data[data['Gender'] == 'F']
data_male = data[data['Gender'] == 'M']
```

In [13]:
```
#Avg amount spent per transaction of all the 50 million female customers
avg_female = data_female['Purchase'].mean()
avg_female
std_female = data_female['Purchase'].std()
se_female = std_female/np.sqrt(len(data_female))
se_female
```

Out[13]:
```
12.936063220950688
```

In [14]:
```
#intervel where population avg will lie
norm.interval(0.95,avg_female,se_female)
```

Out[14]: (8709.21154714068, 8759.919983170272)

In [15]:
```
#Avg amount spent per transaction of all the 50 million male customers
avg_male = data_male['Purchase'].mean()
std_male = data_male['Purchase'].std()
se_male = std_male/np.sqrt(len(data_male))
se_male
```

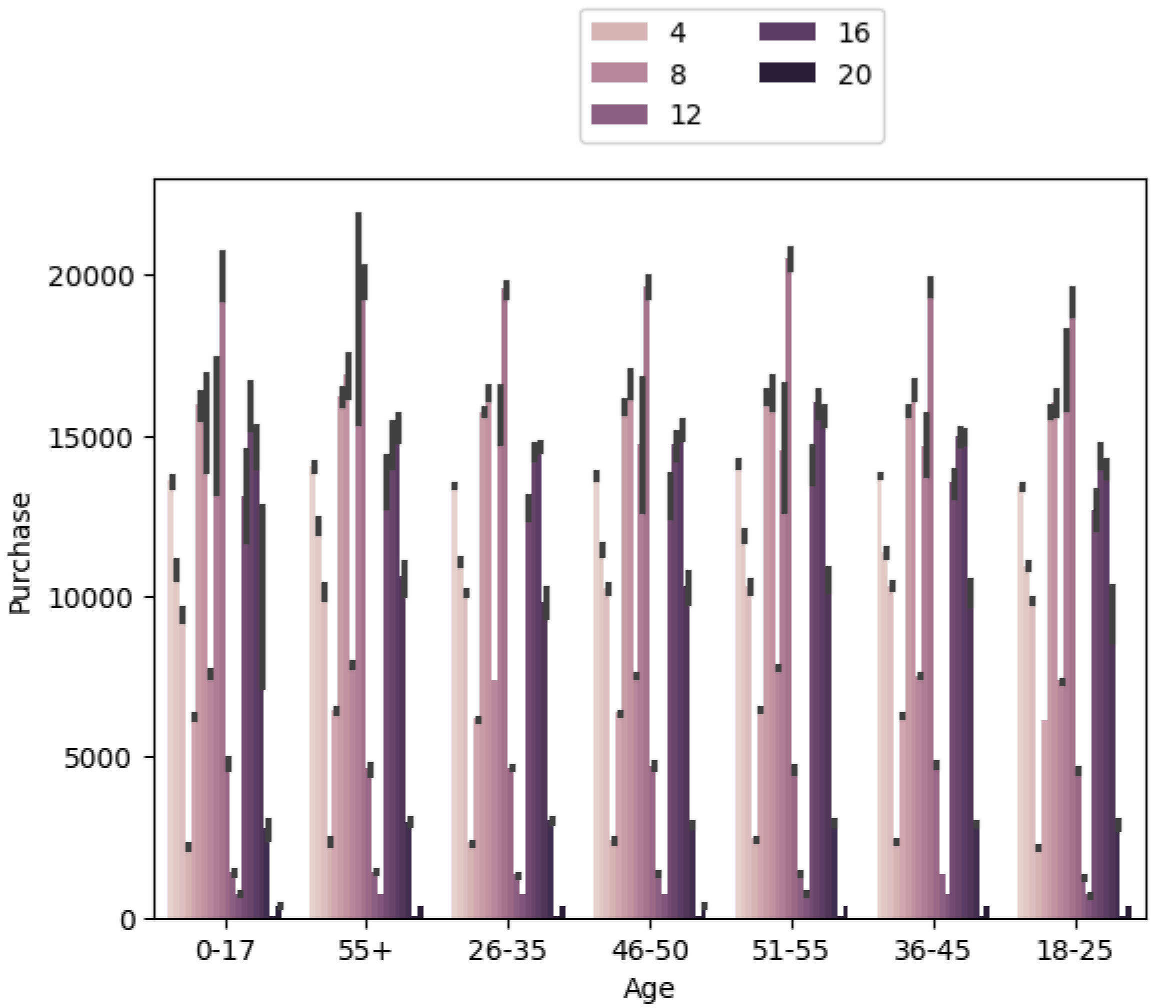
Out[15]: 7.91167247562093

In [16]:
```
#intervel where population avg will lie
norm.interval(0.95,avg_male,se_male)
```

Out[16]: (9422.01944736257, 9453.032633581959)

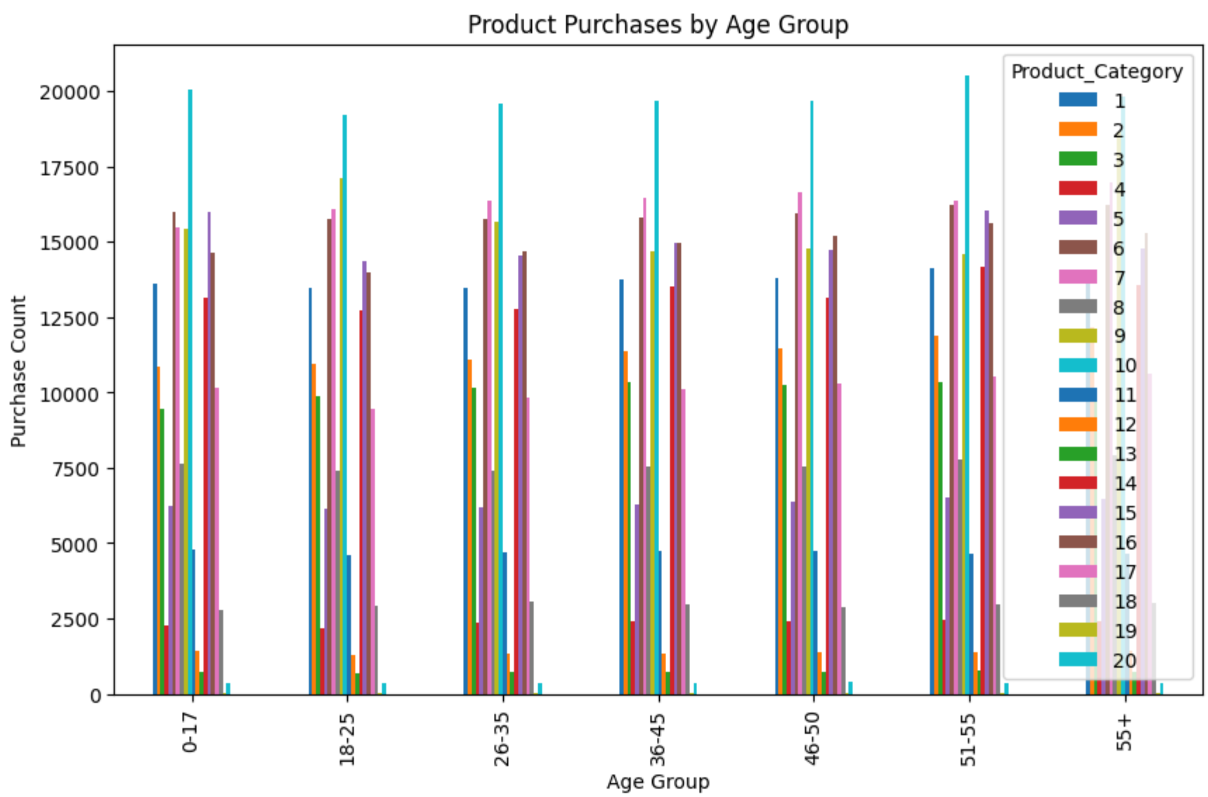
In [17]:
```
#what product groups are bought by various age groups
sns.barplot(y="Purchase", x="Age", data=data, hue="Product_Category")
plt.legend(bbox_to_anchor=(0.75, 1.25), ncol=2)
```

Out[17]: <matplotlib.legend.Legend at 0x7fa75bff7910>



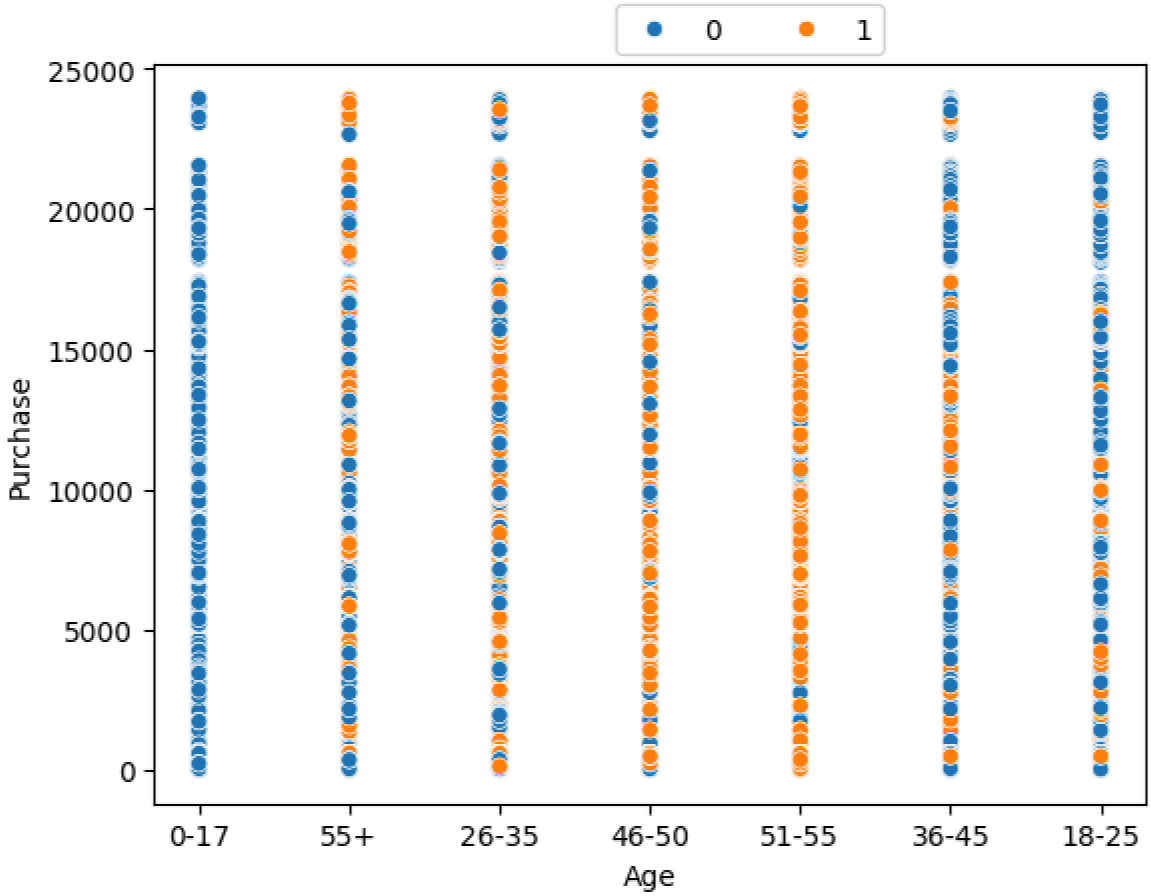
In [18]:
```
data.pivot_table(index='Age', columns='Product_Category', values='Purchase').plot(k
plt.title('Product Purchases by Age Group')
plt.xlabel('Age Group')
```

```
plt.ylabel('Purchase Count')
plt.show()
```



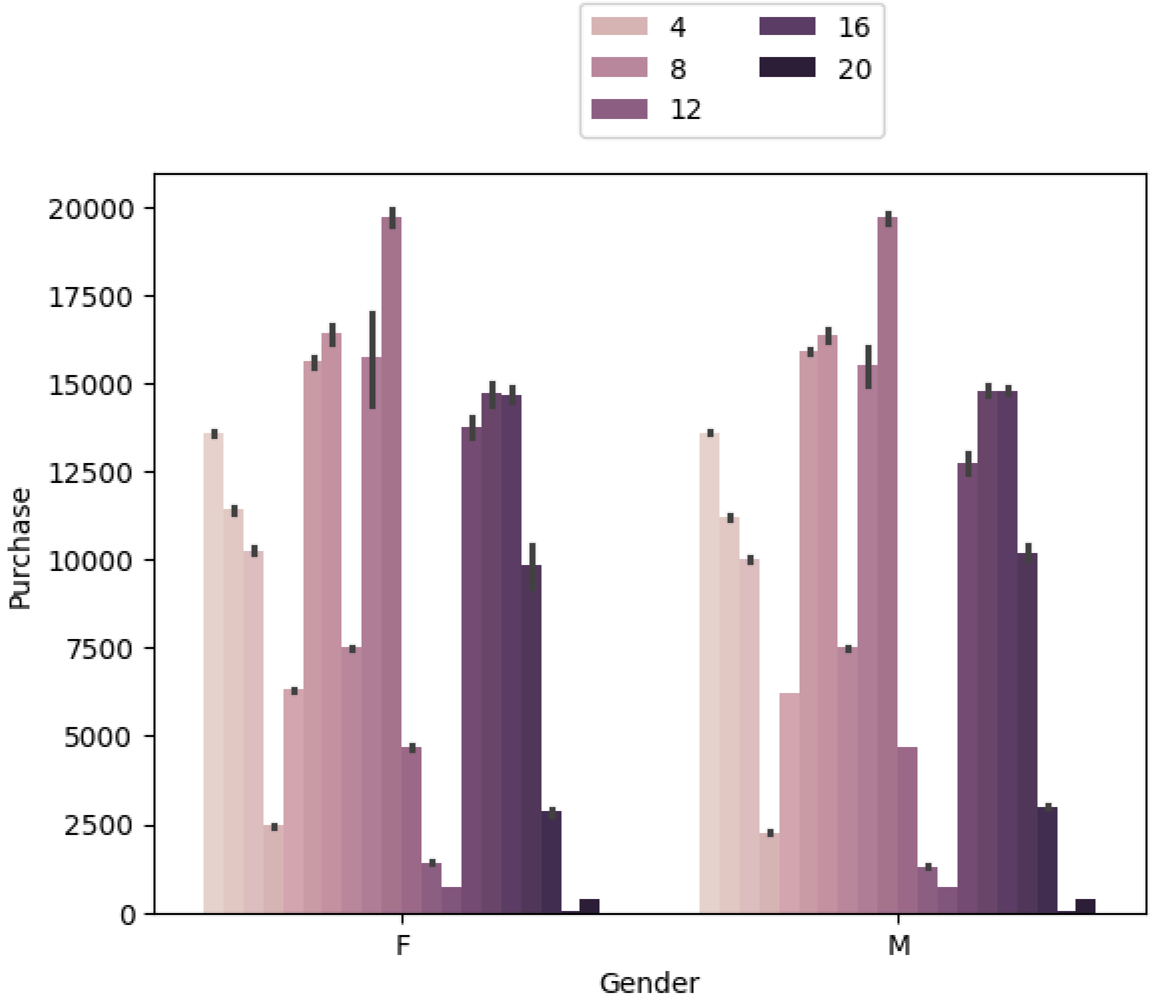
In [19]:
```
#2) is there a relationship between age, marital status and purchase amount -- mult
sns.scatterplot(x="Age", y="Purchase", hue="Marital_Status", data=data)
plt.legend(bbox_to_anchor=(0.75, 1.10), ncol=2)
```

Out[19]: `<matplotlib.legend.Legend at 0x7fa75b40afb0>`



In [20]:
```
#3) Are there preferred product categories for different genders
sns.barplot(hue="Product_Category", x="Gender", data=data, y="Purchase")
plt.legend(bbox_to_anchor=(0.75, 1.25), ncol=2)
```

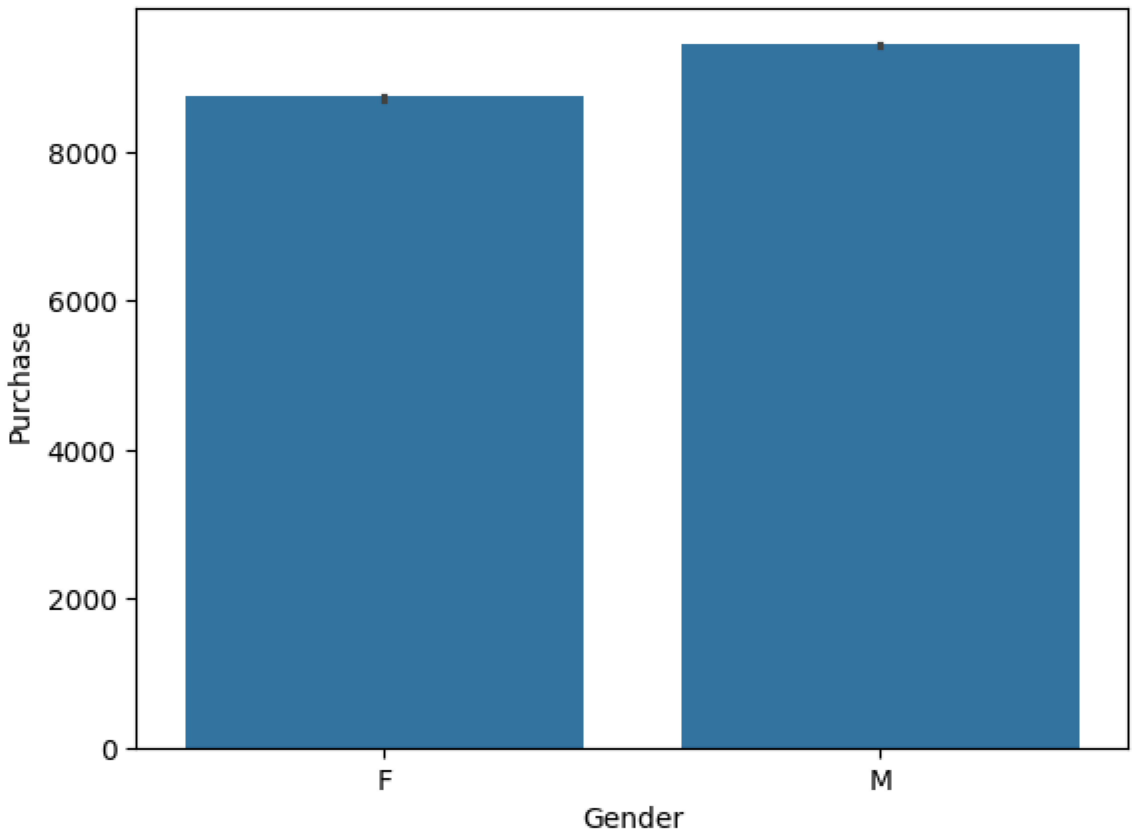
Out[20]: `<matplotlib.legend.Legend at 0x7fa758506050>`



In [21]:
```
#how does gender affect amount spent -- Female
sns.barplot(x="Gender", y="Purchase", data=data)
plt.legend(bbox_to_anchor=(0.75, 1.25), ncol=2)
```

```
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
```

Out[21]: <matplotlib.legend.Legend at 0x7fa7585055a0>



In [22]:
```
#various sampling for female - complete data
avg_female_masterData = data_female['Purchase'].mean()
std_female_masterData = data_female['Purchase'].std()
se_female_masterData = std_female_masterData/np.sqrt(len(data_female))

norm.interval(0.95,avg_female_masterData,se_female_masterData)
```

Out[22]: (8709.21154714068, 8759.919983170272)

In [23]:
```
#various sampling for female - 300
sample_female_300 = data_female.sample(n=300)
avg_female_300 = sample_female_300['Purchase'].mean()
std_female_300 = sample_female_300['Purchase'].std()
se_female_300 = std_female_300/np.sqrt(len(sample_female_300))

norm.interval(0.95,avg_female_300,se_female_300)
```

Out[23]: (7949.009375386708, 9123.483957946624)

In [24]:
```
#various sampling for female - 3000
sample_female_3000 = data_female.sample(n=3000)
avg_female_3000 = sample_female_3000['Purchase'].mean()
std_female_3000 = sample_female_3000['Purchase'].std()
se_female_3000 = std_female_3000/np.sqrt(len(sample_female_3000))
```

```
norm.interval(0.95,avg_female_3000,se_female_3000)
```

Out[24]: (8687.342622383601, 9040.594044283067)

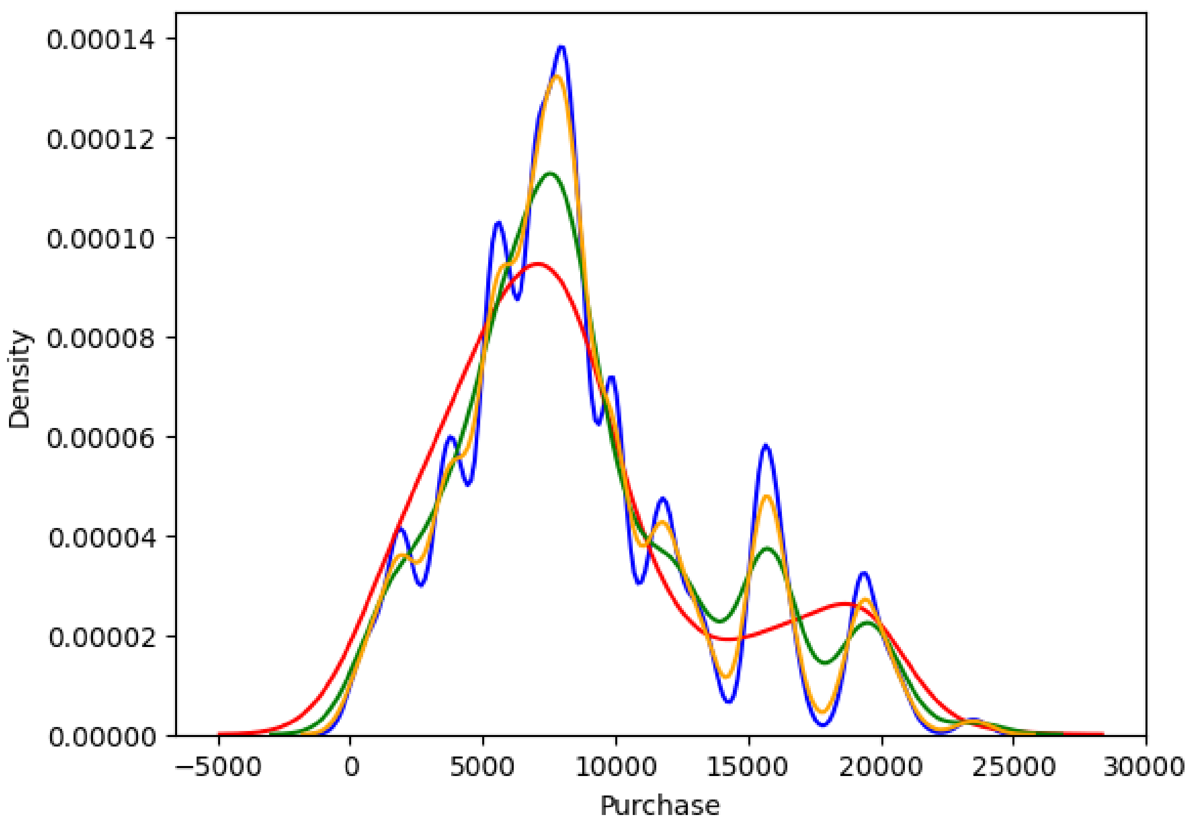
In [25]:
```
#various sampling for female - 30000
sample_female_30000 = data_female.sample(n=30000)
avg_female_30000 = sample_female_30000['Purchase'].mean()
std_female_30000 = sample_female_30000['Purchase'].std()
se_female_30000 = std_female_30000/np.sqrt(len(sample_female_30000))

norm.interval(0.95,avg_female_30000,se_female_30000)
```

Out[25]: (8671.055412444131, 8778.390187555868)

In [26]:
```
#Sample size impact on shape of distributions - Female spending
sns.kdeplot(data_female['Purchase'],color="b")
sns.kdeplot(sample_female_300['Purchase'],color='r')
sns.kdeplot(sample_female_3000['Purchase'],color='g')
sns.kdeplot(sample_female_30000['Purchase'],color="orange")
```

Out[26]: <Axes: xlabel='Purchase', ylabel='Density'>



In [27]:
```
#various sampling for male - complete data
avg_male_masterData = data_male['Purchase'].mean()
std_male_masterData = data_male['Purchase'].std()
se_male_masterData = std_male_masterData/np.sqrt(len(data_male))

norm.interval(0.95,avg_male_masterData,se_male_masterData)
```

Out[27]: (9422.01944736257, 9453.032633581959)

In [28]:
```
#various sampling for male - 300
sample_male_300 = data_male.sample(n=300)
avg_male_300 = sample_male_300['Purchase'].mean()
std_male_300 = sample_male_300['Purchase'].std()
se_male_300 = std_male_300/np.sqrt(len(sample_male_300))

norm.interval(0.95,avg_male_300,se_male_300)
```

Out[28]: (8548.863960771681, 9600.402705894985)

In [29]:
```
#various sampling for male - 3000
sample_male_3000 = data_male.sample(n=3000)
avg_male_3000 = sample_male_3000['Purchase'].mean()
std_male_3000 = sample_male_3000['Purchase'].std()
se_male_3000 = std_male_3000/np.sqrt(len(sample_male_3000))

norm.interval(0.95,avg_male_3000,se_male_3000)
```

Out[29]: (9328.508186545703, 9696.319813454298)

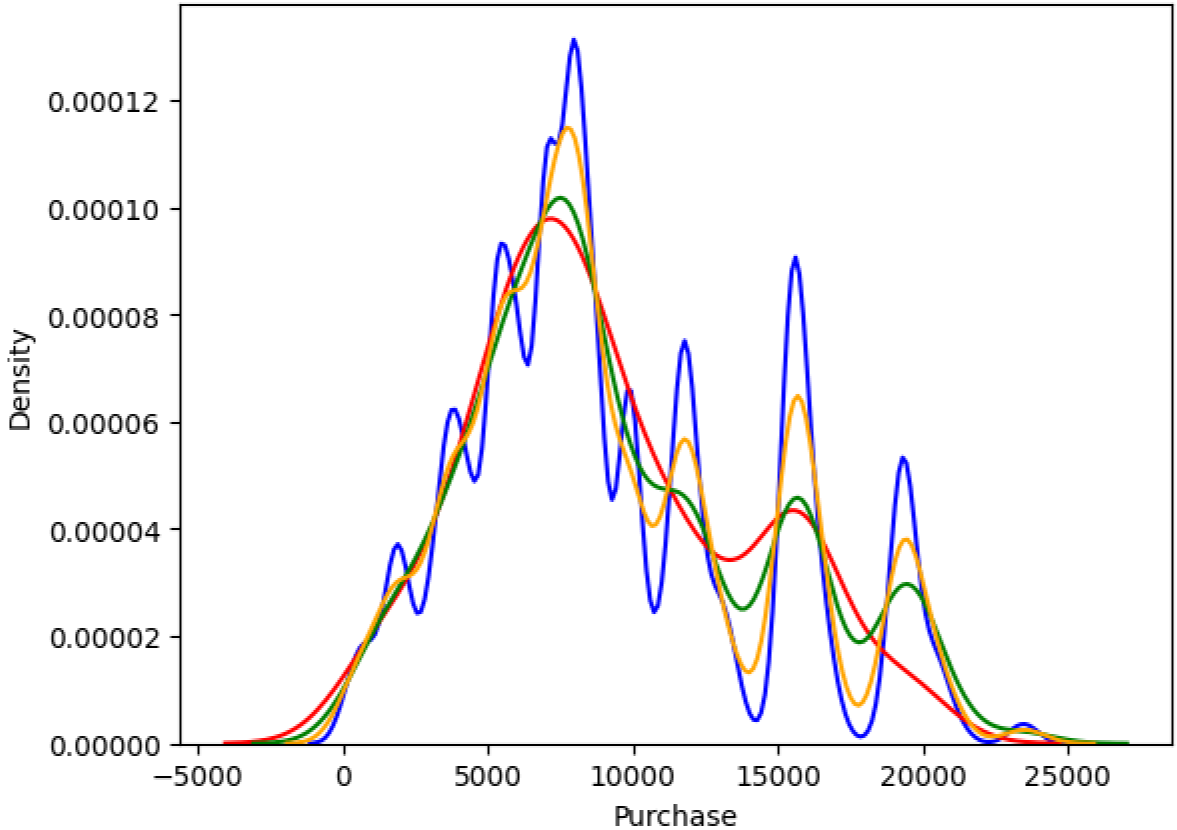
In [30]:
```
#various sampling for male - 30000
sample_male_30000 = data_male.sample(n=30000)
avg_male_30000 = sample_male_30000['Purchase'].mean()
std_male_30000 = sample_male_30000['Purchase'].std()
se_male_30000 = std_male_30000/np.sqrt(len(sample_male_30000))

norm.interval(0.95,avg_male_30000,se_male_30000)
```

Out[30]: (9368.811999610445, 9484.313000389555)

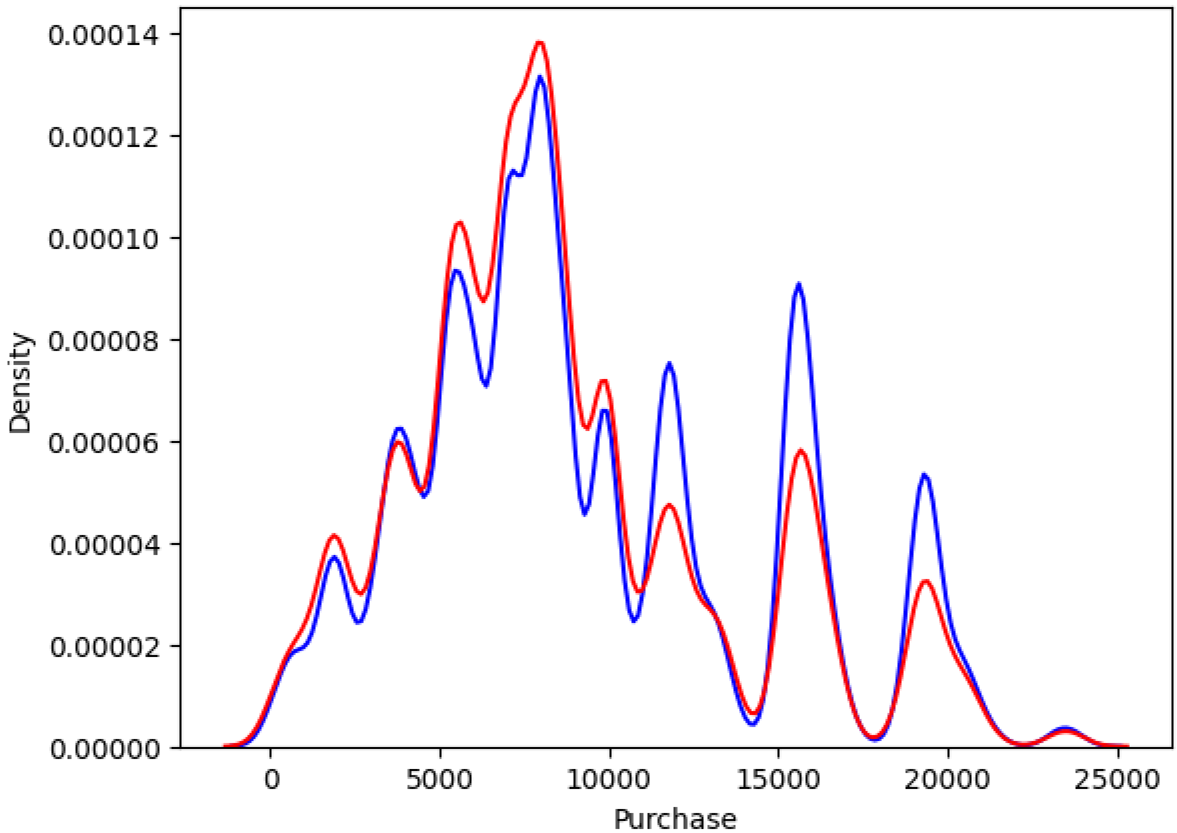
In [31]:
```
#Sample size impact on shape of distributions - male spending
sns.kdeplot(data_male['Purchase'],color="b")
sns.kdeplot(sample_male_300['Purchase'],color='r')
sns.kdeplot(sample_male_3000['Purchase'],color='g')
sns.kdeplot(sample_male_30000['Purchase'],color="orange")
```

Out[31]: <Axes: xlabel='Purchase', ylabel='Density'>



In [32]:
```
#Purchase amount distribution for male and female
sns.kdeplot(data_male['Purchase'],color="b")
sns.kdeplot(data_female['Purchase'],color="r")
```

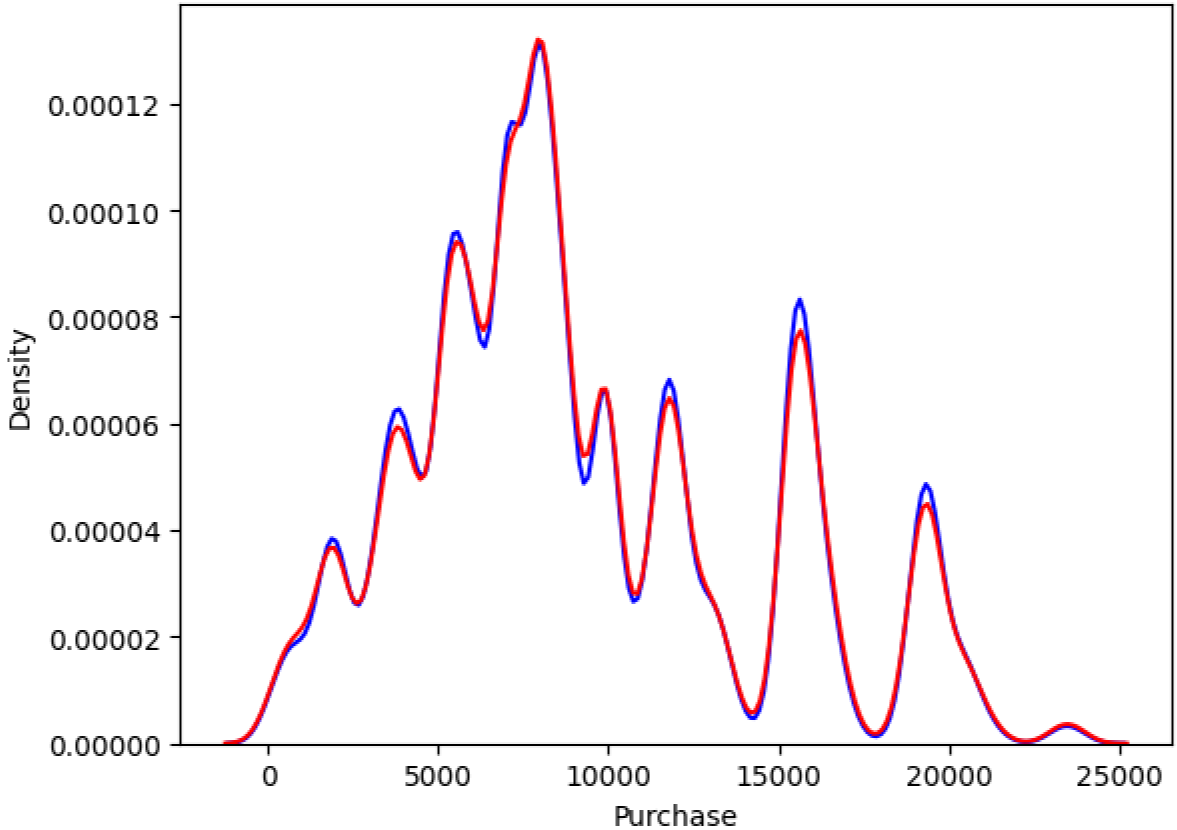
Out[32]: <Axes: xlabel='Purchase', ylabel='Density'>



In [33]:
```python
#How does marital Status impact the purchase amount
data_single = data[data['Marital_Status'] == 0]
data_married = data[data['Marital_Status'] == 1]

sns.kdeplot(data_single['Purchase'],color="b")
sns.kdeplot(data_married['Purchase'],color="r")
```

Out[33]: <Axes: xlabel='Purchase', ylabel='Density'>

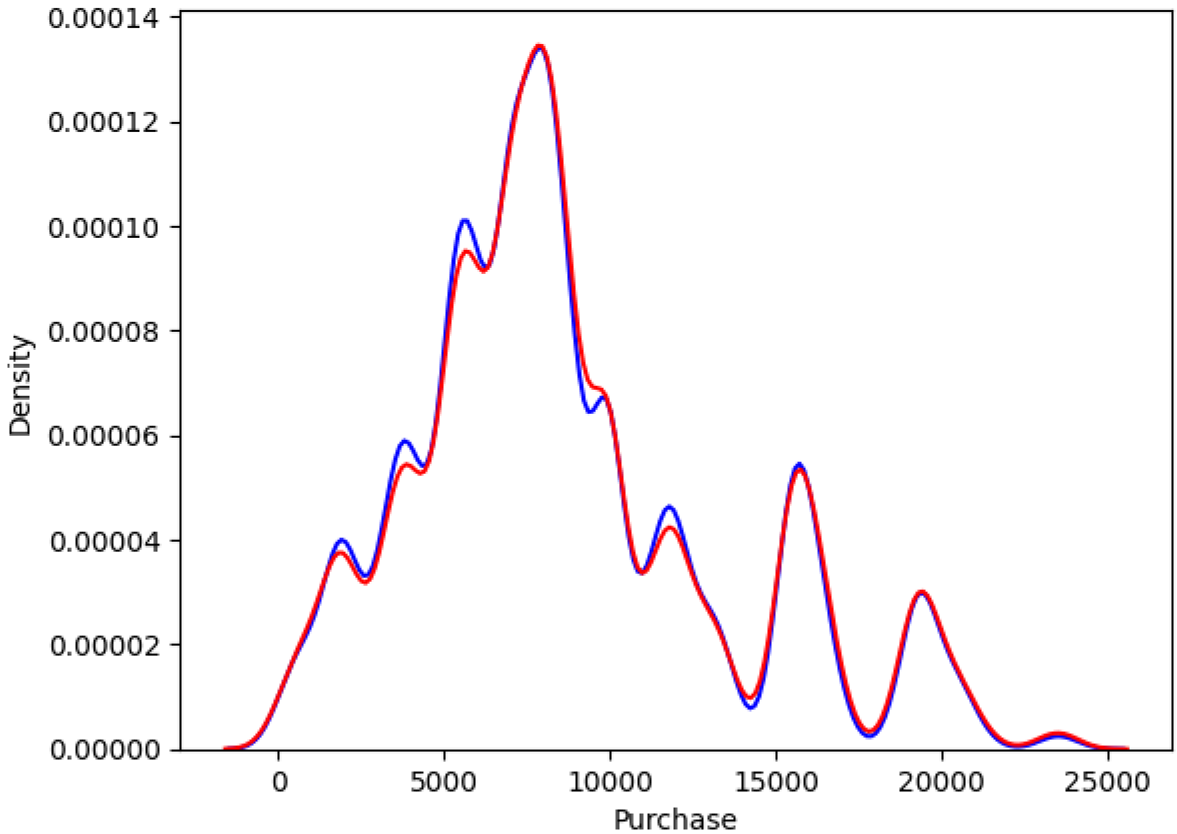


In [34]:
```
#Single Females : Married Females - Purchase amount
data_single_female = data_single[data['Gender'] == 'F']
data_married_female = data_married[data['Gender'] == 'F']


sns.kdeplot(data_single_female['Purchase'],color="b")
sns.kdeplot(data_married_female['Purchase'],color="r")
```

```
<ipython-input-34-6a0f1d6e0cec>:2: UserWarning: Boolean Series key will be reindexed
to match DataFrame index.
  data_single_female = data_single[data['Gender'] == 'F']
<ipython-input-34-6a0f1d6e0cec>:3: UserWarning: Boolean Series key will be reindexed
to match DataFrame index.
  data_married_female = data_married[data['Gender'] == 'F']
```

Out[34]: `<Axes: xlabel='Purchase', ylabel='Density'>`

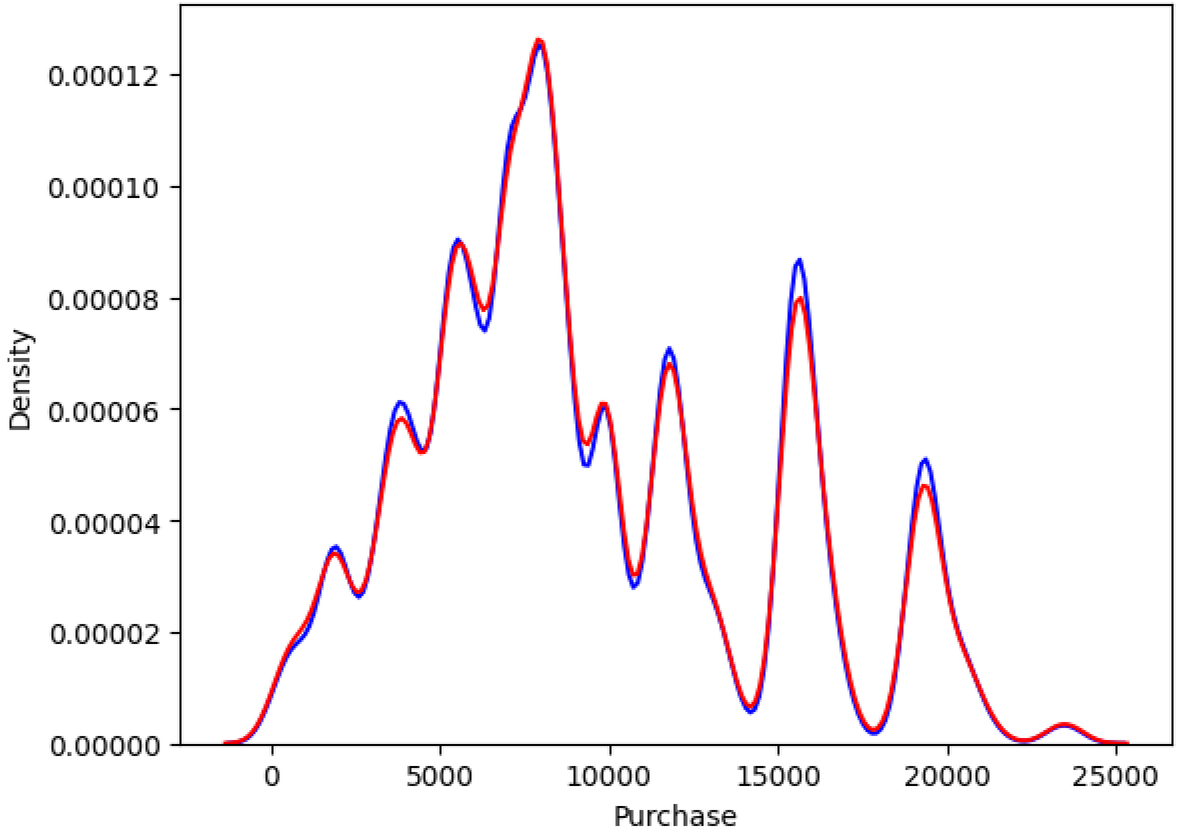


In [35]:
```
#Single males : Married males - Purchase amount
data_single_male = data_single[data['Gender'] == 'M']
data_married_male = data_married[data['Gender'] == 'M']


sns.kdeplot(data_single_male['Purchase'],color="b")
sns.kdeplot(data_married_male['Purchase'],color="r")
```

```
<ipython-input-35-b5a9306cbfdf>:2: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  data_single_male = data_single[data['Gender'] == 'M']
<ipython-input-35-b5a9306cbfdf>:3: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  data_married_male = data_married[data['Gender'] == 'M']
```

Out[35]: <Axes: xlabel='Purchase', ylabel='Density'>



In [36]:

```
#various sampling for Single - complete data
avg_dataSingle_masterData = data_single['Purchase'].mean()
std_dataSingle_masterData = data_single['Purchase'].std()
se_dataSingle_masterData = std_dataSingle_masterData/np.sqrt(len(data_single))
print(avg_dataSingle_masterData)
norm.interval(0.95,avg_dataSingle_masterData,se_dataSingle_masterData)
```

```
9265.907618921507
```

Out[36]: (9248.61641818668, 9283.198819656332)

In [37]:

```
#various sampling for Single - 300
data_single_sample300 = data_single.sample(300)
avg_dataSingle_sample300 = data_single_sample300['Purchase'].mean()
std_dataSingle_sample300 = data_single_sample300['Purchase'].std()
se_dataSingle_sample300 = std_dataSingle_sample300/np.sqrt(len(data_single_sample30
norm.interval(0.95,avg_dataSingle_sample300,se_dataSingle_sample300)
```

Out[37]: (8362.409637003915, 9396.950362996085)

In [38]:

```
#various sampling for Single - 3000
data_single_sample3000 = data_single.sample(3000)
avg_dataSingle_sample3000 = data_single_sample3000['Purchase'].mean()
std_dataSingle_sample3000 = data_single_sample3000['Purchase'].std()
se_dataSingle_sample3000 = std_dataSingle_sample3000/np.sqrt(len(data_single_sample
norm.interval(0.95,avg_dataSingle_sample3000,se_dataSingle_sample3000)
```

Out[38]: (9209.444384173748, 9573.15028249292)

In [39]:
```
#various sampling for Single - 30000
data_single_sample30000 = data_single.sample(30000)
avg_dataSingle_sample30000 = data_single_sample30000['Purchase'].mean()
std_dataSingle_sample30000 = data_single_sample30000['Purchase'].std()
se_dataSingle_sample30000 = std_dataSingle_sample3000/np.sqrt(len(data_single_sampl
norm.interval(0.95,avg_dataSingle_sample30000,se_dataSingle_sample30000)
```

Out[39]: (9245.364248143691, 9360.378151856308)

In [40]:
```
#various sampling for married - complete data
avg_datamarried_masterData = data_married['Purchase'].mean()
std_datamarried_masterData = data_married['Purchase'].std()
se_datamarried_masterData = std_datamarried_masterData/np.sqrt(len(data_married))
print(avg_datamarried_masterData)
norm.interval(0.95,avg_datamarried_masterData,se_datamarried_masterData)
```

9261.174574082374

Out[40]: (9240.460427057078, 9281.888721107669)

In [41]:
```
#various sampling for married - 300
data_married_sample300 = data_married.sample(300)
avg_datamarried_sample300 = data_married_sample300['Purchase'].mean()
std_datamarried_sample300 = data_married_sample300['Purchase'].std()
se_datamarried_sample300 = std_datamarried_sample300/np.sqrt(len(data_married_sampl
norm.interval(0.95,avg_datamarried_sample300,se_datamarried_sample300)
```

Out[41]: (8805.276141444549, 9978.190525222119)

In [42]:
```
#various sampling for married - 3000
data_married_sample3000 = data_married.sample(3000)
avg_datamarried_sample3000 = data_married_sample3000['Purchase'].mean()
std_datamarried_sample3000 = data_married_sample3000['Purchase'].std()
se_datamarried_sample3000 = std_datamarried_sample3000/np.sqrt(len(data_married_sam
norm.interval(0.95,avg_datamarried_sample3000,se_datamarried_sample3000)
```

Out[42]: (8955.913132498783, 9319.274867501215)

In [43]:
```
#various sampling for married - 30000
data_married_sample30000 = data_married.sample(30000)
avg_datamarried_sample30000 = data_married_sample30000['Purchase'].mean()
std_datamarried_sample30000 = data_married_sample30000['Purchase'].std()
se_datamarried_sample30000 = std_datamarried_sample3000/np.sqrt(len(data_married_sa
norm.interval(0.95,avg_datamarried_sample30000,se_datamarried_sample30000)
```

Out[43]: (9204.14276514209, 9319.04783485791)

In [44]:
```
data_age_017 = data[data['Age'] == '0-17']
data_age_1825 = data[data['Age'] == '18-25']
data_age_2635 = data[data['Age'] == '26-35']
data_age_3645 = data[data['Age'] == '36-45']
data_age_4650 = data[data['Age'] == '46-50']
fiftyone_plus = ['51-55','55+']
data_age_51plus = data[data['Age'].isin(fiftyone_plus)]
```

In [45]:
```
#various sampling for  age group 0-17 - complete data
avg_age017_masterData = data_age_017['Purchase'].mean()
std_age017_masterData = data_age_017['Purchase'].std()
se_age017_masterData = std_age017_masterData/np.sqrt(len(data_age_017))
print(avg_age017_masterData)
norm.interval(0.95,avg_age017_masterData,se_age017_masterData)
```

```
8933.464640444974
```

Out[45]:
```
(8851.947970542686, 9014.981310347262)
```

In [46]:
```
#various sampling for  age group 18-25 - complete data
avg_age1825 = data_age_1825['Purchase'].mean()
std_age1825 = data_age_1825['Purchase'].std()
se_age1825 = std_age017_masterData/np.sqrt(len(data_age_1825))
print(avg_age1825)
norm.interval(0.95,avg_age1825,se_age1825)
```

```
9169.663606261289
```

Out[46]:
```
(9137.931184259874, 9201.396028262703)
```

In [47]:
```
#various sampling for  age group 26-35 - complete data
avg_age2635 = data_age_2635['Purchase'].mean()
std_age2635 = data_age_2635['Purchase'].std()
se_age2635 = std_age2635/np.sqrt(len(data_age_2635))
print(avg_age2635)
norm.interval(0.95,avg_age2635,se_age2635)
```

```
9252.690632869888
```

Out[47]:
```
(9231.73367640003, 9273.647589339746)
```

In [48]:
```
#various sampling for  age group 36-45 - complete data
avg_age3645 = data_age_3645['Purchase'].mean()
std_age3645 = data_age_3645['Purchase'].std()
se_age3645 = std_age017_masterData/np.sqrt(len(data_age_3645))
print(avg_age3645)
norm.interval(0.95,avg_age3645,se_age3645)
```

```
9331.350694917874
```

Out[48]:
```
(9301.148280754005, 9361.553109081742)
```

In [49]:
```
#various sampling for  age group 46-50 - complete data
avg_age4650 = data_age_4650['Purchase'].mean()
std_age4650 = data_age_4650['Purchase'].std()
se_age4650 = std_age4650/np.sqrt(len(data_age_4650))
print(avg_age4650)
norm.interval(0.95,avg_age4650,se_age4650)
```

```
9208.625697468327
```

Out[49]:
```
(9163.085142648752, 9254.166252287903)
```

In [50]:
```
#various sampling for  age group 51+ - complete data
avg_age51plus = data_age_51plus['Purchase'].mean()
std_age51plus = data_age_51plus['Purchase'].std()
se_age51plus = std_age51plus/np.sqrt(len(data_age_51plus))
```

```
print(avg_age51plus)
norm.interval(0.95,avg_age51plus,se_age51plus)
```

```
9463.661678193484
```

Out[50]:
```
(9423.166383417869, 9504.1569729691)
```

In [51]:
```
#changes to width of the interval - Gender Scenario - Male
print(avg_male_masterData)
print(norm.interval(0.90,avg_male_masterData,se_male_masterData))
print(norm.interval(0.95,avg_male_masterData,se_male_masterData))
print(norm.interval(0.99,avg_male_masterData,se_male_masterData))
```

```
9437.526040472265
(9424.512497305488, 9450.539583639042)
(9422.01944736257, 9453.032633581959)
(9417.146922669479, 9457.90515827505)
```

In [52]:
```
#changes to width of interval - gender Scenario - Female
print(avg_female_masterData)
print(norm.interval(0.90,avg_female_masterData,se_female_masterData))
print(norm.interval(0.95,avg_female_masterData,se_female_masterData))
print(norm.interval(0.99,avg_female_masterData,se_female_masterData))
```

```
8734.565765155476
(8713.287834648021, 8755.84369566293)
(8709.21154714068, 8759.919983170272)
(8701.24467443839, 8767.88685587256)
```

In [54]:
```
#changes to width interval - Married Customers
print(avg_datamarried_masterData)
print(norm.interval(0.90,avg_datamarried_masterData,se_datamarried_masterData))
print(norm.interval(0.95,avg_datamarried_masterData,se_datamarried_masterData))
print(norm.interval(0.99,avg_datamarried_masterData,se_datamarried_masterData))
```

```
9261.174574082374
(9243.790713903045, 9278.558434261702)
(9240.460427057078, 9281.888721107669)
(9233.951570329937, 9288.39757783481)
```

In [55]:
```
#changes to width interval - UnMarried Customers
print(avg_dataSingle_masterData)
print(norm.interval(0.90,avg_dataSingle_masterData,se_dataSingle_masterData))
print(norm.interval(0.95,avg_dataSingle_masterData,se_dataSingle_masterData))
print(norm.interval(0.99,avg_dataSingle_masterData,se_dataSingle_masterData))
```

```
9265.907618921507
(9251.39638582367, 9280.418852019344)
(9248.61641818668, 9283.198819656332)
(9243.183129136169, 9288.632108706845)
```

In [ ]: