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

In [ ]:
```
df=pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/
```

In [ ]:
```
#Checking for Null or missing values
df.isna().sum()/df.shape[0]
```

Out[ ]:
```
User_ID                       0.0
Product_ID                    0.0
Gender                        0.0
Age                           0.0
Occupation                    0.0
City_Category                 0.0
Stay_In_Current_City_Years    0.0
Marital_Status                0.0
Product_Category              0.0
Purchase                      0.0
dtype: float64
```

In [ ]:
```
#Checking the count of unique values in each column in the dataset:
df.nunique().sort_values(ascending=False)
```

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

In [ ]:
```
df["Gender"]=df["Gender"].astype("category")
df["City_Category"]=df["City_Category"].astype("category")
```

In [ ]:
```
df.shape
```

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

In [ ]:
```
df.describe(include=["object","category"]).T
```

Out[ ]:

	count	unique	top	freq
Product_ID	550068	3631	P00265242	1880
Gender	550068	2	M	414259
Age	550068	7	26-35	219587
City_Category	550068	3	B	231173
Stay_In_Current_City_Years	550068	5	1	193821

In [ ]:
```
df["Gender"].value_counts()
```

Out[ ]:
```
Gender
M    414259
F    135809
Name: count, dtype: int64
```

In [ ]:
```
df["Age"].value_counts()
```

Out[ ]:
```
Age
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: count, dtype: int64
```

In [ ]:
```
df["City_Category"].value_counts()
```

Out[ ]:
```
City_Category
B    231173
C    171175
A    147720
Name: count, dtype: int64
```

In [ ]:
```
df["Marital_Status"].value_counts()
```

Out[ ]:
```
Marital_Status
0    324731
1    225337
Name: count, dtype: int64
```

In [ ]:
```
df["Stay_In_Current_City_Years"].value_counts()
```

Out[ ]:
```
Stay_In_Current_City_Years
1     193821
2     101838
3      95285
4+     84726
0      74398
Name: count, dtype: int64
```

In [ ]:
```
df["Product_Category"].unique()
```

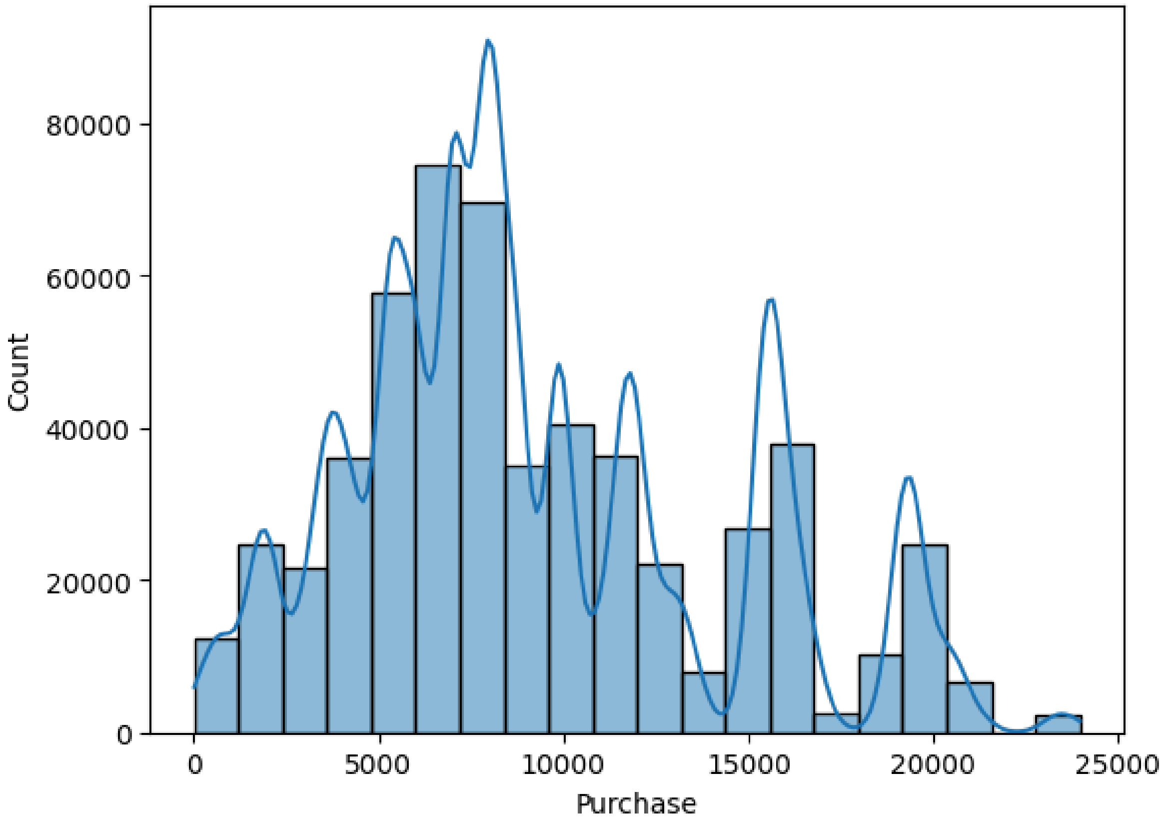
Out[ ]:
```
array([ 3,  1, 12,  8,  5,  4,  2,  6, 14, 11, 13, 15,  7, 16, 18, 10, 17,
        9, 20, 19])
```

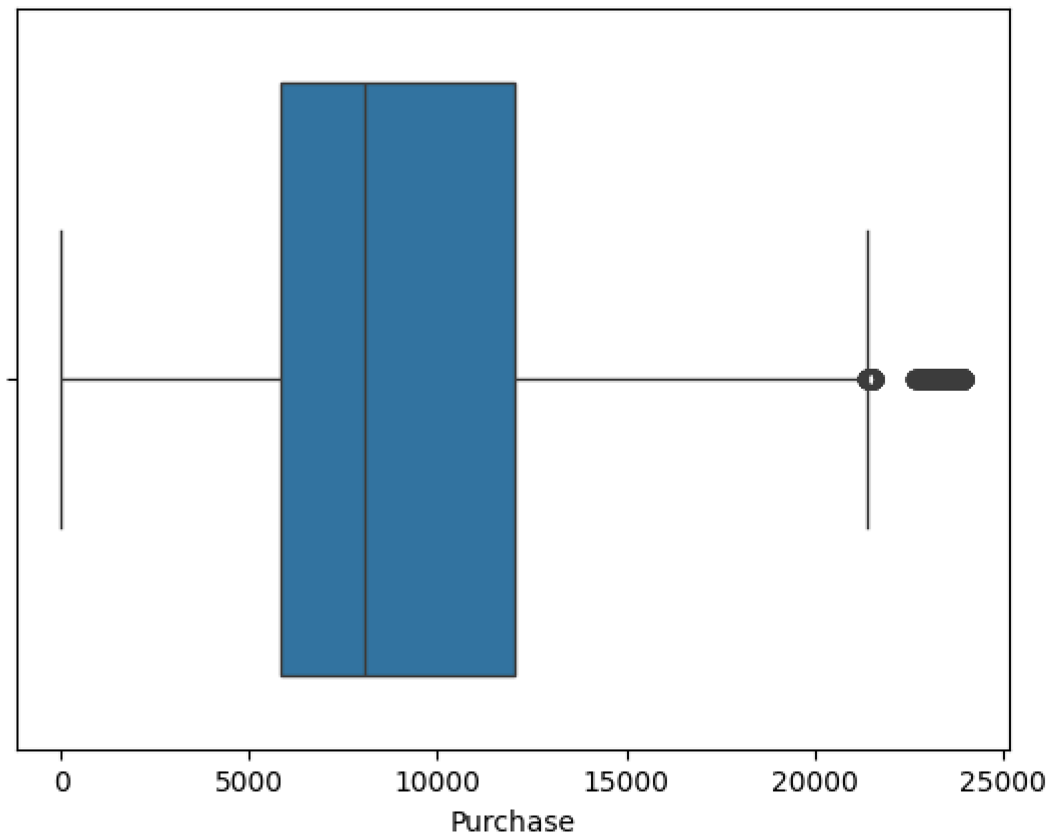
In [ ]:
```
df["Occupation"].unique()
```

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

In [ ]:
```
def encode(data):
  if data==1:
    return "Married"
  else:
    return "Single"
df["Marital_Status"]=df["Marital_Status"].apply(encode)
```

In [ ]:
```
def encode(data):
  if data=="M":
    return "Male"
  else:
    return "Female"
df["Gender"]=df["Gender"].apply(encode)
```

In [ ]:
```
sns.histplot(data=df['Purchase'],bins=20,kde=True)
```
Out[ ]:
```
<Axes: xlabel='Purchase', ylabel='Count'>
```


In [ ]:
```
sns.boxplot(x = df['Purchase'],vert = False,patch_artist = True)
```
Out[ ]:
```
<Axes: xlabel='Purchase'>
```


In [ ]:
```
iqr1=df['Purchase'].quantile(0.75)-df['Purchase'].quantile(0.25)
upper_limit=df['Purchase'].quantile(0.75) + 1.5*iqr1
upper_limit
```
Out[ ]:
```
21400.5
```

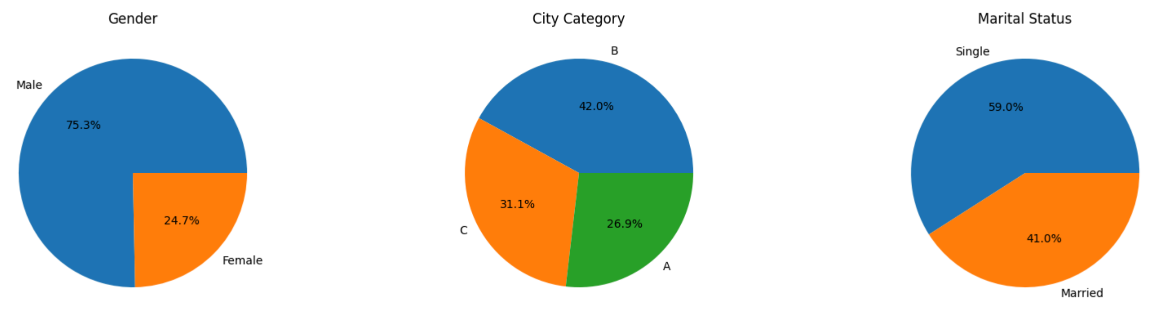
In [ ]:
```
len(df.loc[df['Purchase'] > 21400.5,'Purchase'])
```
Out[ ]:
```
2677
```

In [ ]:
```
len(df.loc[df['Purchase'] > 21400.5,'Purchase'])/ len(df['Purchase'])
```
Out[ ]:
```
0.004866671029763593
```

-- There are total of 2677 outliers which is roughly 0.48% of the total data present in purchase amount.

Univariate Analysis

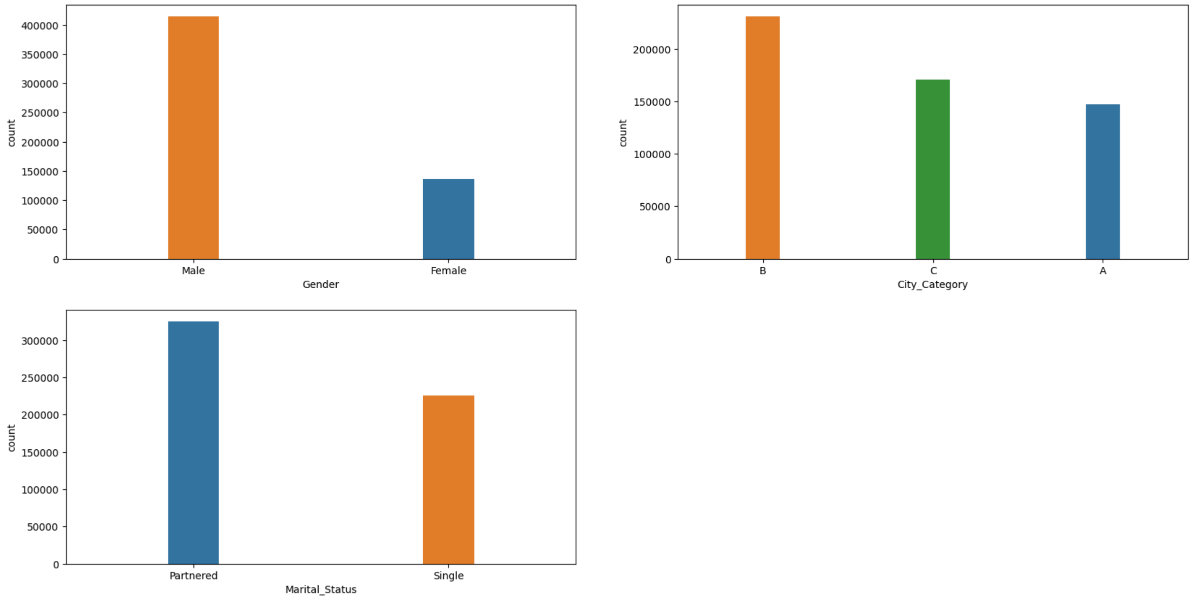
In [ ]:
```
plt.figure(figsize=(20,10))
plt.subplot(2,3,1)
mylabel=df["Gender"].value_counts()
size=mylabel.values
label=mylabel.index
plt.pie(size,labels=label,autopct="%1.1f%%")
plt.title("Gender")
plt.subplot(2,3,2)
mylabel=df["City_Category"].value_counts()
size=mylabel.values
label=mylabel.index
plt.pie(size,labels=label,autopct="%1.1f%%")
plt.title("City Category")
plt.subplot(2,3,3)
mylabel=df["Marital_Status"].value_counts()
size=mylabel.values
label=mylabel.index
plt.pie(size,labels=label,autopct="%1.1f%%")
plt.title("Marital Status")
plt.show()
```



Insights

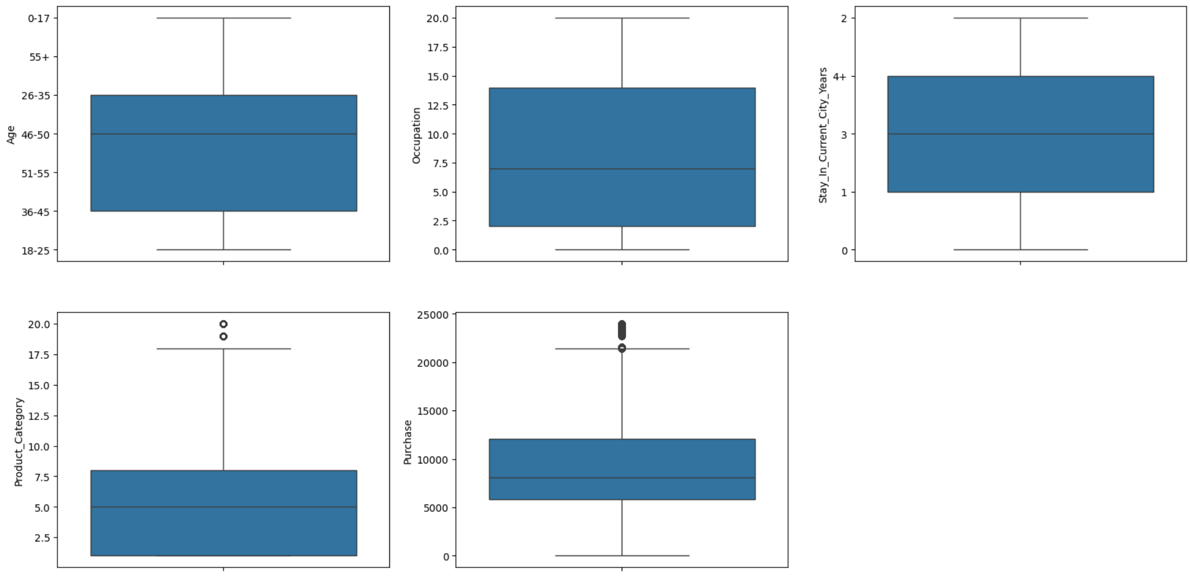
- There is a significant difference in purchase behavior of men and women during the Black Friday event at Walmart
- From City Category City Category-B made most transactions followed by City Category-C and City Category-B.
- Unmarried Customers made higher shopping and Purchases.

In [ ]:
```
plt.figure(figsize=(20,10))
plt.subplot(2,2,1)
sns.countplot(data=df,x="Gender",order=df["Gender"].value_counts().index,hue="Gende
plt.subplot(2,2,2)
sns.countplot(data=df,x="City_Category",order=df["City_Category"].value_counts().ir
plt.subplot(2,2,3)
sns.countplot(data=df,x="Marital_Status",order=df["Marital_Status"].value_counts().
plt.show()
```



In [ ]:
```
plt.figure(figsize=(20,10))
plt.subplot(2,3,1)
sns.boxplot(y="Age",data=df)
plt.subplot(2,3,2)
sns.boxplot(y="Occupation",data=df)
plt.subplot(2,3,3)
sns.boxplot(y="Stay_In_Current_City_Years",data=df)
plt.subplot(2,3,4)
sns.boxplot(y="Product_Category",data=df)
plt.subplot(2,3,5)
sns.boxplot(y="Purchase",data=df)
```

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

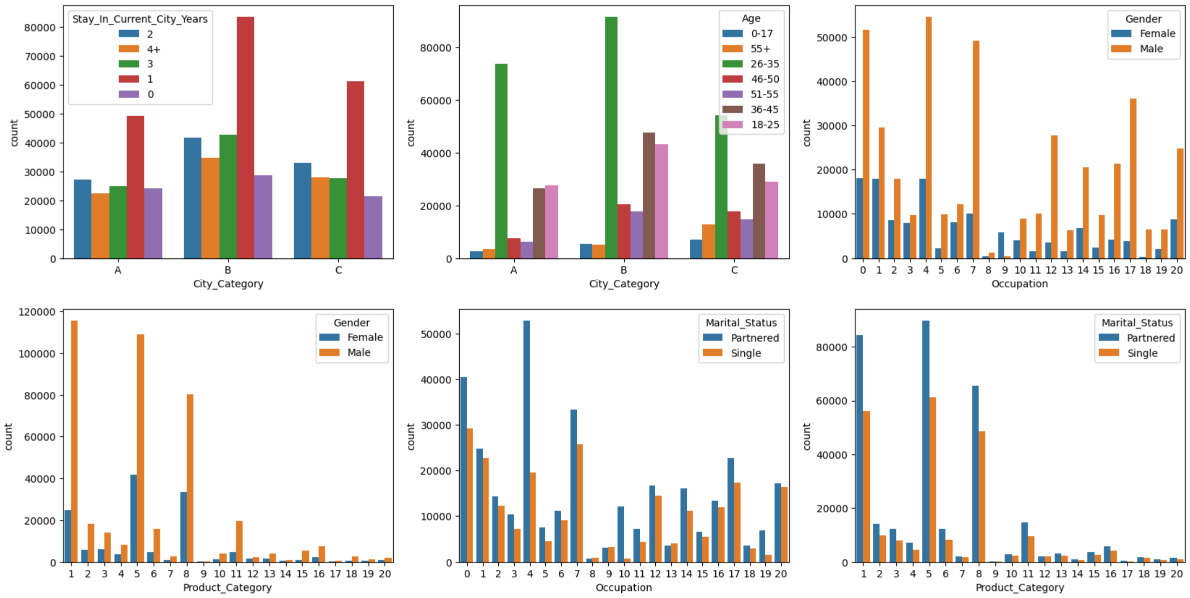


Insights

- There are many outliers in the Purchase column Indicating that most customers made higher purchases.

Bivariate Analysis

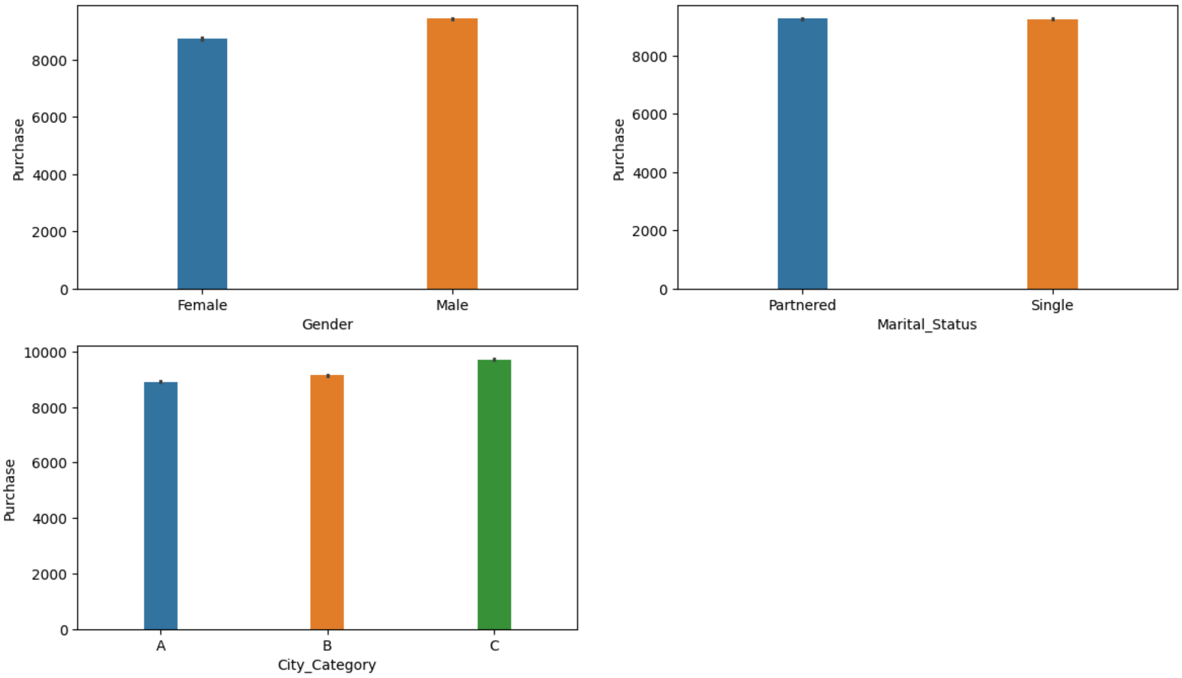
In [ ]:
```
plt.figure(figsize=(20,10))
plt.subplot(2,3,1)
sns.countplot(x="City_Category",hue="Stay_In_Current_City_Years",data=df)
plt.subplot(2,3,2)
sns.countplot(x="City_Category",hue="Age",data=df)
plt.subplot(2,3,3)
sns.countplot(x="Occupation",hue="Gender",data=df)
plt.subplot(2,3,4)
sns.countplot(x="Product_Category",hue="Gender",data=df)
plt.subplot(2,3,5)
sns.countplot(x="Occupation",hue="Marital_Status",data=df)
plt.subplot(2,3,6)
sns.countplot(x="Product_Category",hue="Marital_Status",data=df)
plt.show()
```



Insights

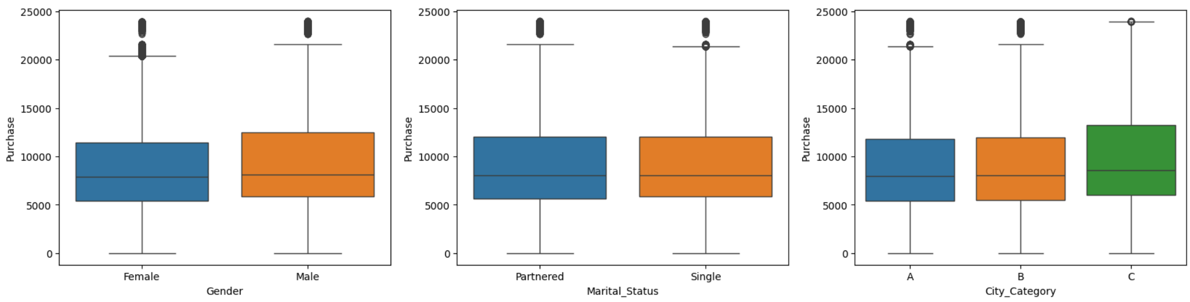
- From the above plot we can say that most of the walmart customers are from city category A,B and C who stayed in the city for a year.
- Most of the walmart customers are within a age range of 26-35 from the 3 Cities.
- From genders most male customers prefered product 1,5 and 8.
- Married and Unmarried customers of walmart mostly purchased product 1,5 and 8.
- From genders most male customers are having occupations 0,4 and 7.
- Married and Unmarried customers are having occupations 0,4 and 7.

In [ ]:
```
plt.figure(figsize=(14,8))
plt.subplot(2,2,1)
sns.barplot(x="Gender",y="Purchase",hue="Gender",data=df,width=0.2)
plt.subplot(2,2,2)
sns.barplot(x="Marital_Status",y="Purchase",hue="Marital_Status",data=df,width=0.2)
plt.subplot(2,2,3)
sns.barplot(x="City_Category",y="Purchase",hue="City_Category",data=df,width=0.2)
plt.show()
```



In [ ]:
```
plt.figure(figsize=(20,10))
plt.subplot(2,3,1)
sns.boxplot(x="Gender",y="Purchase",hue="Gender",data=df)
plt.subplot(2,3,2)
sns.boxplot(x="Marital_Status",y="Purchase",hue="Marital_Status",data=df)
plt.subplot(2,3,3)
sns.boxplot(x="City_Category",y="Purchase",hue="City_Category",data=df)
```

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



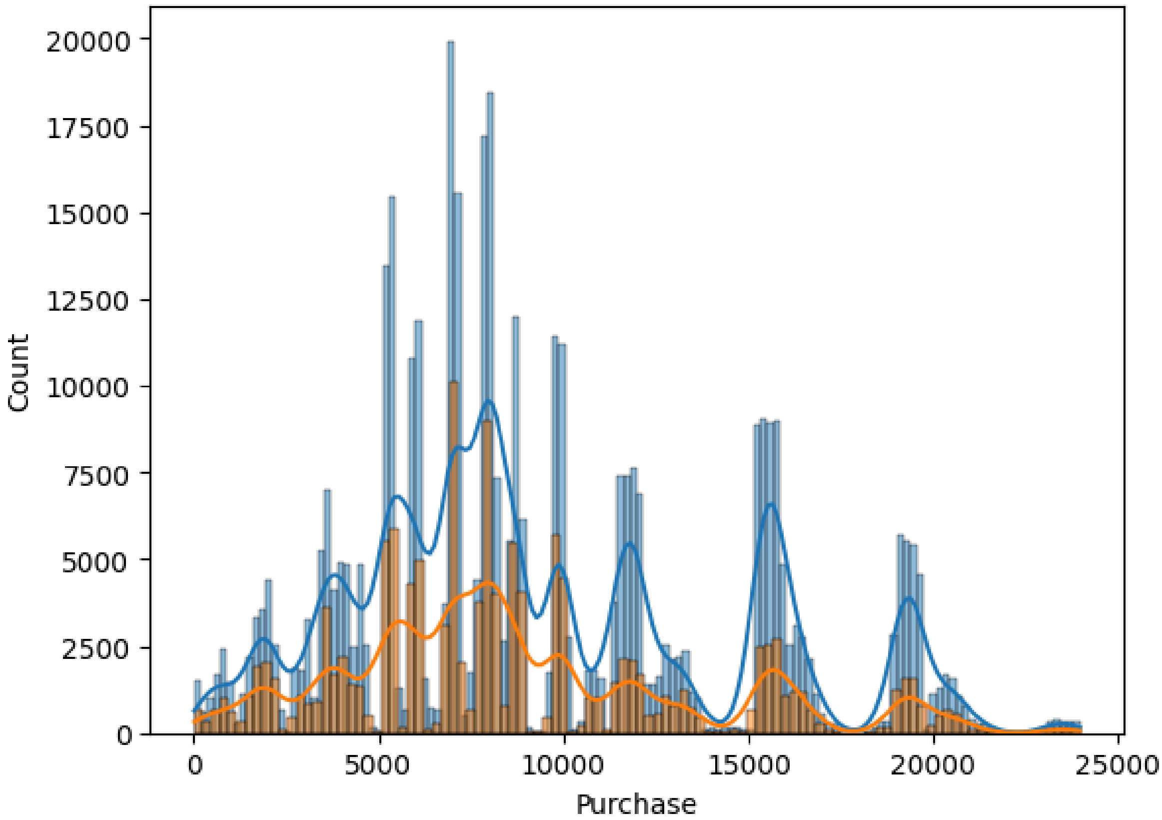
Insights

- There are many outliers in purchase column based on gender.Indicating that most male and female walmart customers shopped above the average amount.
- There are many outliers in purchase column based on Marital Status.Indicating that most unmarried and married walmart customers shopped above the average amount

In [ ]:
```
df_male=df[df["Gender"]=="Male"]["Purchase"]
df_female=df[df["Gender"]=="Female"]["Purchase"]
```

In [ ]:
```
sns.histplot(df_male,kde=True)
sns.histplot(df_female,kde=True)
```

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



In [ ]:
```
#calculating mean of purchase amounts for Males and Females:
mu_M=df_male.mean()
mu_F=df_female.mean()
print(mu_M,mu_F)
```

```
9437.526040472265 8734.565765155476
```

In [ ]:
```
#calculating the standard deviation for purchase amounts for males and females:
sigma_M=df_male.std()
sigma_F=df_female.std()
print(sigma_M,sigma_F)
```

```
5092.18620977797 4767.233289291458
```

In [ ]:
```
#Calculating the standard error:
df_male.shape, df_female.shape
se_M = round(sigma_M/(np.sqrt(df_male.shape[0])),3)
se_F = round(sigma_F/(np.sqrt(df_female.shape[0])),3)
se_M,se_F
```

Out[ ]:
```
(7.912, 12.936)
```

In [ ]:
```
#95% confidence interval --> 5% significance level --> alpha = 0.05
#Since the test will be 2-tailed, alpha=0.025 on each side.
z=norm.ppf(0.025)
z
```

Out[ ]:
```
-1.9599639845400545
```

In [ ]:
```
#The upper and lower limits of the confidence interval with 95% confidence -->
#Males:
(mu_M+(se_M*z),mu_M-(se_M*z))
```

Out[ ]:
```
(9422.018805426584, 9453.033275517946)
```

In [ ]:
```
#Females:
(mu_F+(se_F*z),mu_F-(se_F*z))
```

Out[ ]:
```
(8709.211671051466, 8759.919859259486)
```

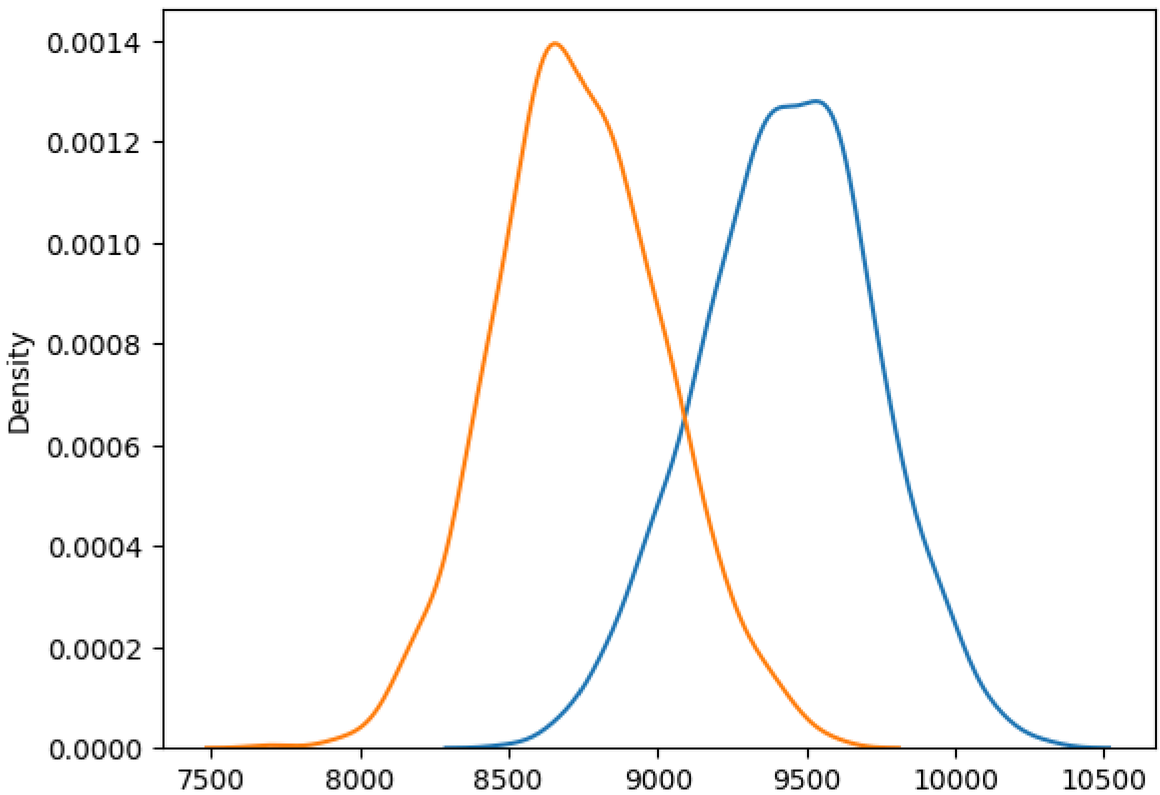
Taking samples of 300 entries for Genders

In [ ]:
```
sample_size=300
iterations=1000
df_samp300_male=[df_male.sample(sample_size,replace=True).mean() for i in range(ite
df_samp300_female=[df_female.sample(sample_size,replace=True).mean() for i in range
```

Creating kde plots to check if it there is any Overlapping

In [ ]:
```
sns.kdeplot(df_samp300_male)
sns.kdeplot(df_samp300_female)
```

Out[ ]:
```
<Axes: ylabel='Density'>
```



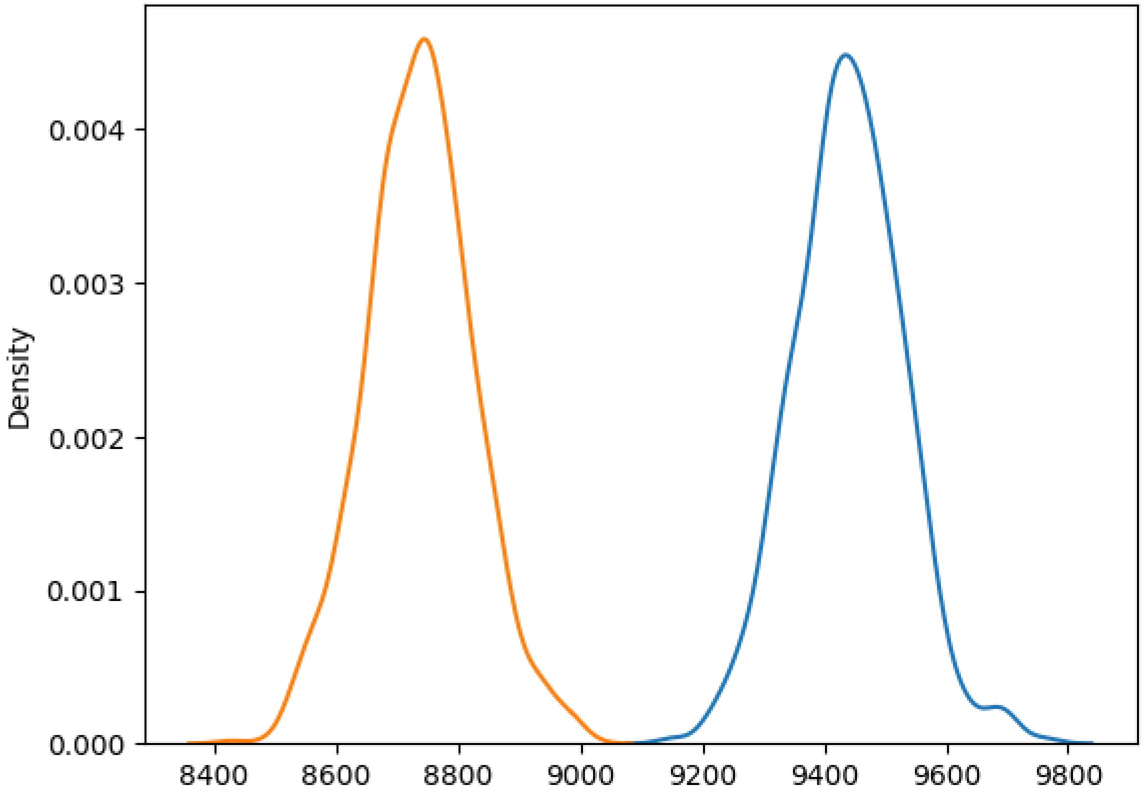
Taking samples of 3000 entries for Genders

In [ ]:
```
sample_size=3000
iterations=1000
df_samp3000_male=[df_male.sample(sample_size,replace=True).mean() for i in range(it
df_samp3000_female=[df_female.sample(sample_size,replace=True).mean() for i in rang
```

Creating kde plots to check if it there is any Overlapping

In [ ]:
```
sns.kdeplot(df_samp3000_male)
sns.kdeplot(df_samp3000_female)
```

Out[ ]:
```
<Axes: ylabel='Density'>
```



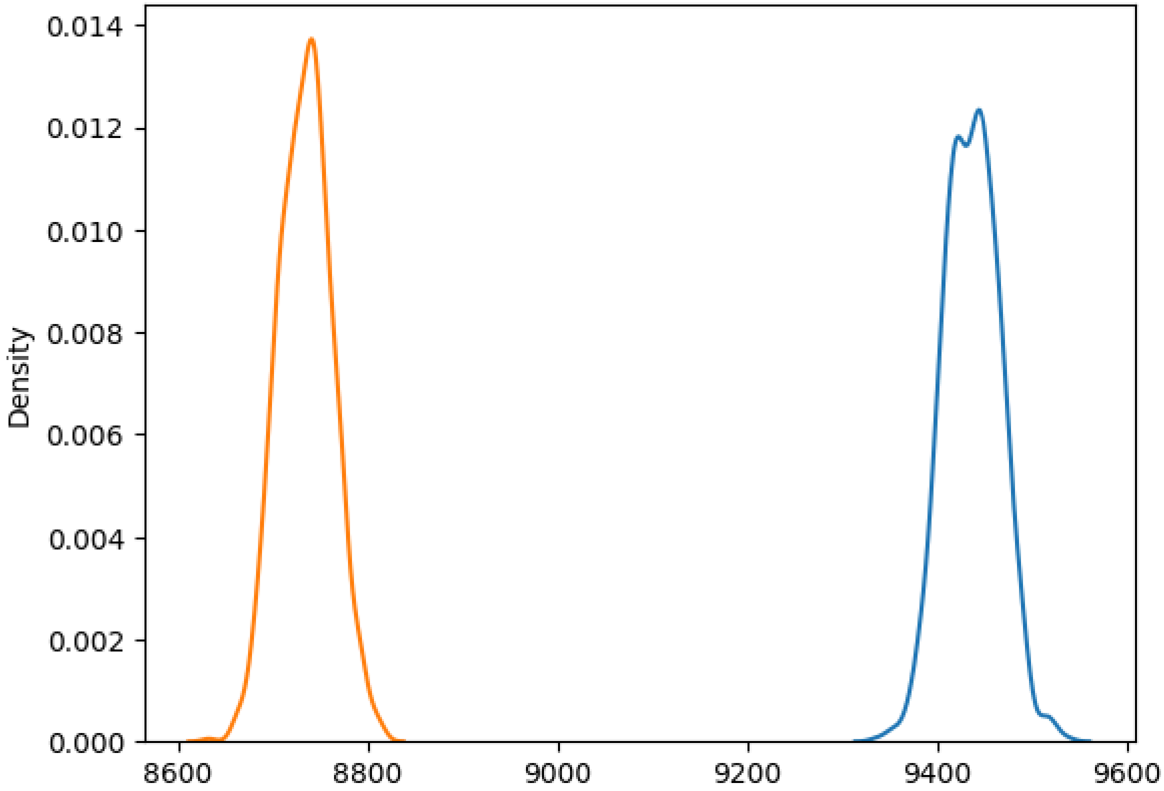
Taking samples of 30000 entries for Genders

In [ ]:
```
sample_size=30000
iterations=1000
df_samp30000_male=[df_male.sample(sample_size,replace=True).mean() for i in range(i
df_samp30000_female=[df_female.sample(sample_size,replace=True).mean() for i in rar
```

Creating kde plots to check if it there is any Overlapping

In [ ]:
```
sns.kdeplot(df_samp30000_male)
sns.kdeplot(df_samp30000_female)
```

Out[ ]:
```
<Axes: ylabel='Density'>
```



Finding different confidence intervals for males and females

In [ ]:
```
for i in ['males', 'females']:
  print('For {g}-'.format(g = i))
  if i == 'males':
    means = df_samp30000_male
    gen = 'Male'
  else:
    means = df_samp30000_female
    gen = 'Female'
  print('Mean of sample means =',np.mean(means))
  print('Population mean =', np.mean(df.loc[df['Gender']==gen, 'Purchase']))
  print('Standard deviation of means (Standard Error) =', np.std(means))
  print('Standard deviation of population =',df.loc[df['Gender']==gen,'Purchase'].s
  print('99% CONFIDENCE INTERVAL for mean expense by {g} users-'.format(g =i))
  print((np.percentile(means, 0.5).round(2), np.percentile(means, 99.5).round(2)))
  print('95% CONFIDENCE INTERVAL for mean expense by {g} users-'.format(g =i))
  print((np.percentile(means, 2.5).round(2), np.percentile(means, 97.5).round(2)))
  print('90% CONFIDENCE INTERVAL for mean expense by {g} users-'.format(g =i))
  print((np.percentile(means, 5).round(2), np.percentile(means, 95).round(2)))
  print('-'*50)
```

```
For males-
Mean of sample means = 9436.606184466667
Population mean = 9437.526040472265
Standard deviation of means (Standard Error) = 29.350865377085803
Standard deviation of population = 5092.18620977797
99% CONFIDENCE INTERVAL for mean expense by males users-
(9361.73, 9517.4)
95% CONFIDENCE INTERVAL for mean expense by males users-
(9381.76, 9491.04)
90% CONFIDENCE INTERVAL for mean expense by males users-
(9390.99, 9484.34)
--------------------------------------------------
For females-
Mean of sample means = 8734.873394333334
Population mean = 8734.565765155476
Standard deviation of means (Standard Error) = 28.04266267193141
Standard deviation of population = 4767.233289291458
99% CONFIDENCE INTERVAL for mean expense by females users-
(8664.34, 8808.23)
95% CONFIDENCE INTERVAL for mean expense by females users-
(8682.22, 8791.27)
90% CONFIDENCE INTERVAL for mean expense by females users-
(8689.67, 8780.1)
--------------------------------------------------
```

Insights

Are women spending more money per transaction than men? Why or Why not

1.Sample Size

- As the sample size increases the confidence intervals become narrower and precise.This suggests that the larger sample sizes can provide more reliable insights. 2.Confidence Intervals
- For smaller samples the confidence interval is overlapping as we are increasing the sample size the overlapping is not seen.This means that there is a statistically difference between average spending for men and women. 3.Population Average
- We are 95% confident that the true population average for males falls between $9,382 and 9$,492 ,and for females , it falls between $8,683 and 8$,792 .
- From the above confidence interval comparing males and females average spending we have concluding that the males are tending to spend more than females per transaction on average

In [ ]:
```
df_single=df[df["Marital_Status"]=="Single"]
df_married=df[df["Marital_Status"]=="Married"]
```

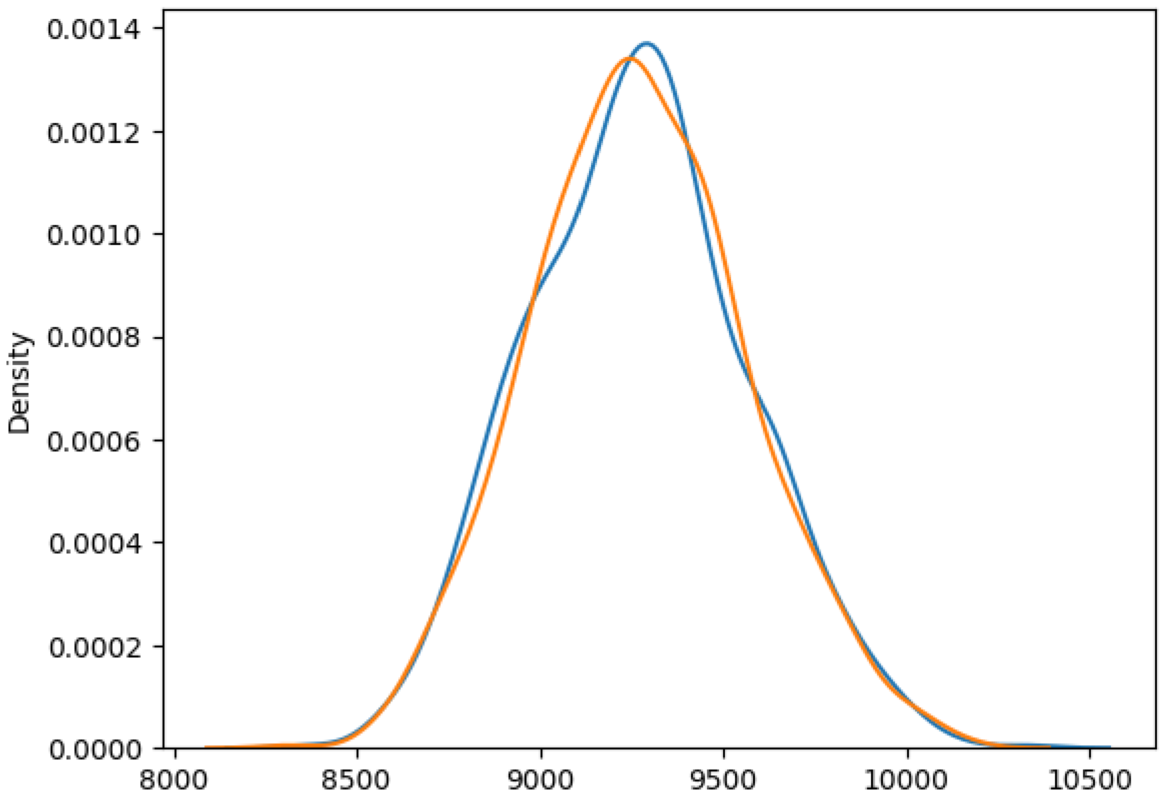
Taking samples of 300 entries for married and unmarried people

In [ ]:
```
sample_size=300
iterations=1000
df_samp300_single=[df_single.sample(sample_size,replace=True)["Purchase"].mean() fo
df_samp300_married=[df_married.sample(sample_size,replace=True)["Purchase"].mean()
```

Creating kde plots to check if it there is any Overlapping

In [ ]:
```
sns.kdeplot(df_samp300_single)
sns.kdeplot(df_samp300_married)
```

Out[ ]: <Axes: ylabel='Density'>



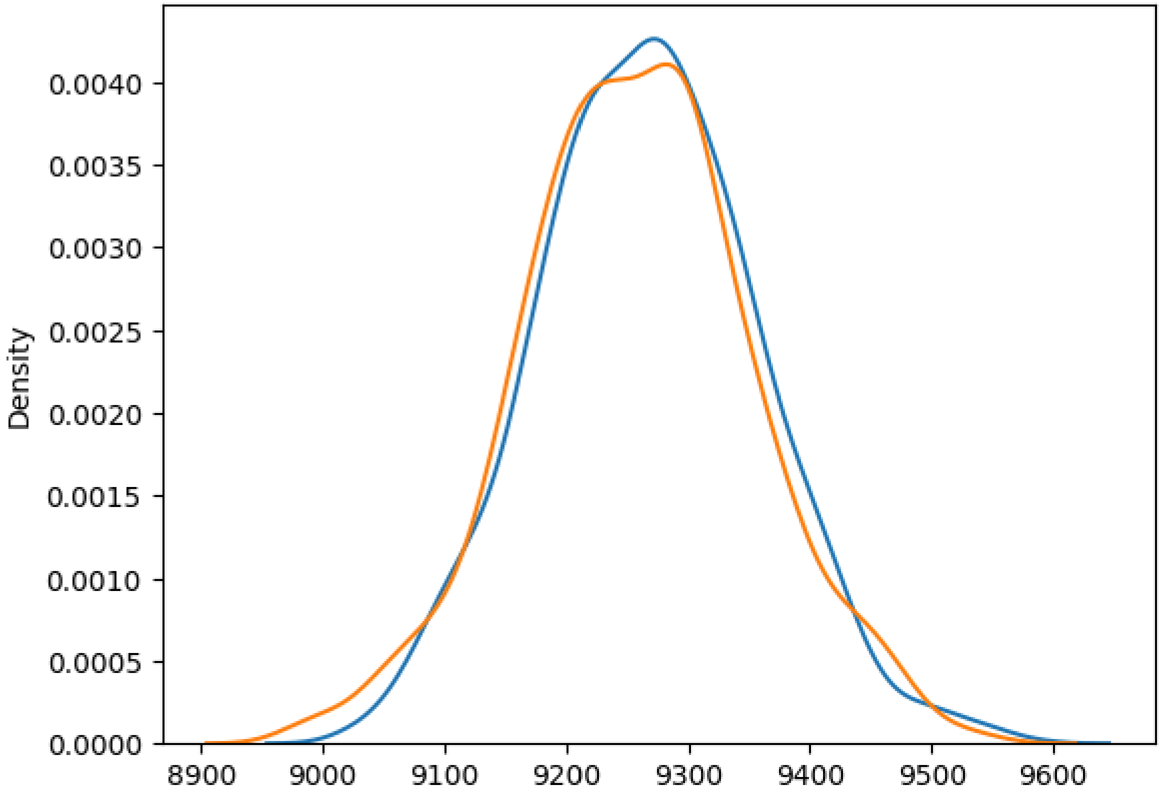
Taking samples of 3000 entries for married and unmarried people

In [ ]:
```
sample_size=3000
iterations=1000
df_samp3000_single=[df_single.sample(sample_size,replace=True)["Purchase"].mean() f
df_samp3000_married=[df_married.sample(sample_size,replace=True)["Purchase"].mean()
```

Creating kde plots to check if it there is any Overlapping

In [ ]:
```
sns.kdeplot(df_samp3000_single)
sns.kdeplot(df_samp3000_married)
```

Out[ ]: <Axes: ylabel='Density'>



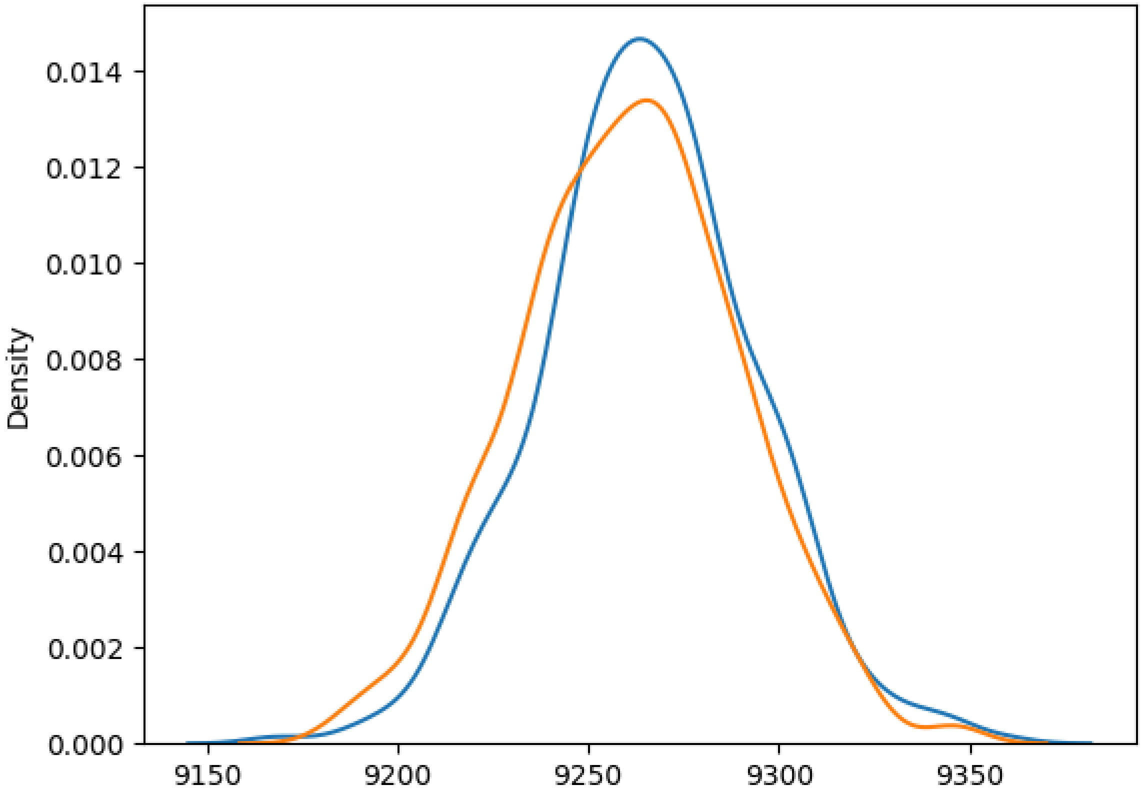
Taking samples of 30000 entries for married and unmarried people

In [ ]:
```
sample_size=30000
iterations=1000
df_samp30000_single=[df_single.sample(sample_size,replace=True)["Purchase"].mean()
df_samp30000_married=[df_married.sample(sample_size,replace=True)["Purchase"].mean(
```

Creating kde plots to check if it there is any Overlapping

In [ ]:
```
sns.kdeplot(df_samp30000_single)
sns.kdeplot(df_samp30000_married)
```

Out[ ]:
```
<Axes: ylabel='Density'>
```



Finding different confidence intervals for mean expense by married and unmarried customers

In [ ]:
```
for i in ['married', 'unmarried']:
  print('For {m}-'.format(m = i))
  if i == 'married':
    means = df_samp30000_single
    ms = "Single"
  else:
    means = df_samp30000_married
    ms = "Partnered"
  print('Mean of sample means =',np.mean(means))
  print('Population mean =', np.mean(df.loc[df['Marital_Status']==ms,'Purchase']))
  print('Standard deviation of means (Standard Error) =', np.std(means))
  print('Standard deviation of population =',df.loc[df['Marital_Status']==ms,'Purch
  print('99% CONFIDENCE INTERVAL for mean expense by {m} users-'.format(m =i))
  print((np.percentile(means, 0.5).round(2), np.percentile(means, 99.5).round(2)))
  print('95% CONFIDENCE INTERVAL for mean expense by {m} users-'.format(m =i))
  print((np.percentile(means, 2.5).round(2), np.percentile(means, 97.5).round(2)))
  print('90% CONFIDENCE INTERVAL for mean expense by {m} users-'.format(m =i))
  print((np.percentile(means, 5).round(2), np.percentile(means, 95).round(2)))
  print('-'*50)
```

```
For married-
Mean of sample means = 9265.6018606
Population mean = 9265.907618921507
Standard deviation of means (Standard Error) = 28.343141187901068
Standard deviation of population = 5027.347858674449
99% CONFIDENCE INTERVAL for mean expense by married users-
(9189.44, 9346.4)
95% CONFIDENCE INTERVAL for mean expense by married users-
(9212.39, 9321.68)
90% CONFIDENCE INTERVAL for mean expense by married users-
(9218.71, 9310.69)
--------------------------------------------------
For unmarried-
Mean of sample means = 9260.689213133333
Population mean = nan
Standard deviation of means (Standard Error) = 29.003489546745747
Standard deviation of population = nan
99% CONFIDENCE INTERVAL for mean expense by unmarried users-
(9188.78, 9343.47)
95% CONFIDENCE INTERVAL for mean expense by unmarried users-
(9202.96, 9317.49)
90% CONFIDENCE INTERVAL for mean expense by unmarried users-
(9214.35, 9308.59)
--------------------------------------------------
```

Insights

1.Sample Size

- As the sample size increases the confidence intervals become narrower and precise.This suggests that the larger sample sizes can provide more reliable insights 2.Confidence Intervals

- We can see that the confidence intervals for all the sample sizes are overlapping.can conclude that there is no significant difference between the average spending per transaction for married and unmarried customers. 3.Population Average

- We are 95% confident that the confidence intervals of married customers 9,213 and 9,322,and for unmarried customers 9,203 and 9,318 are equal in spending.

In [ ]:
```
df_age1=df[df["Age"]=="0-17"]
df_age2=df[df["Age"]=="18-25"]
df_age3=df[df["Age"]=="26-35"]
df_age4=df[df["Age"]=="36-45"]
df_age5=df[df["Age"]=="46-50"]
df_age6=df[df["Age"]=="51-55"]
df_age7=df[df["Age"]=="55+"]
```

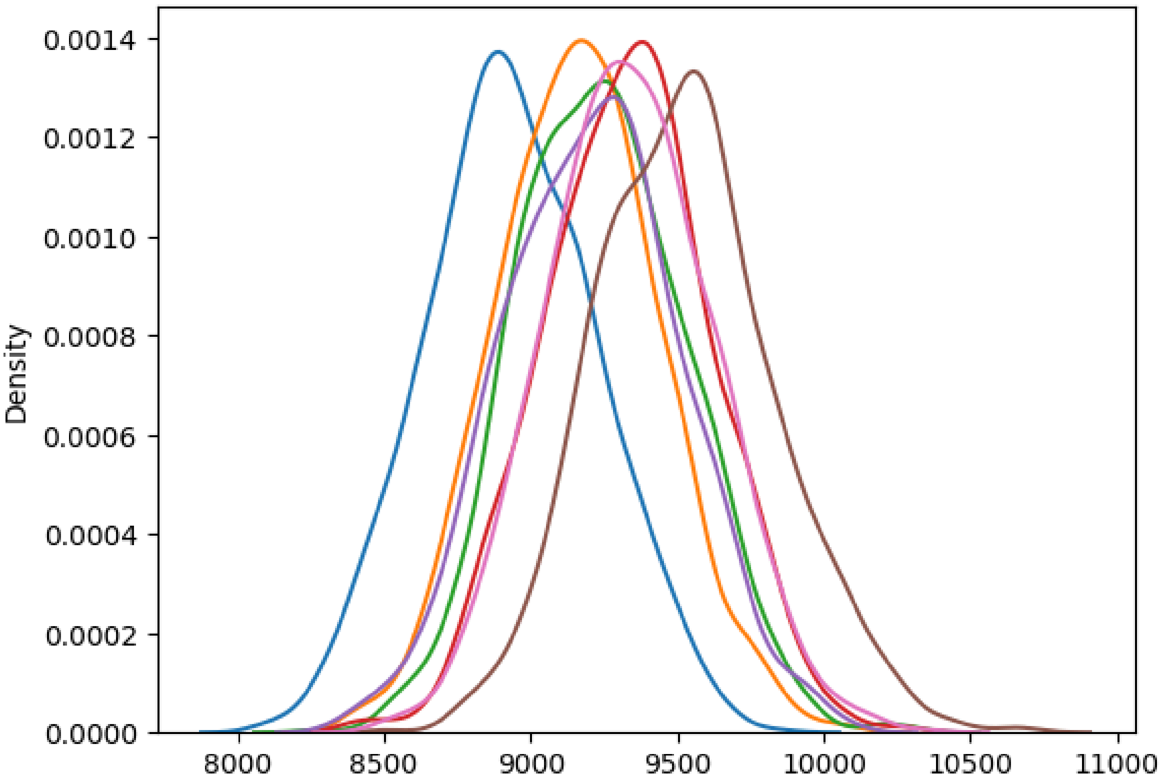
Taking samples of 300 entries for each age group

In [ ]:
```
sample_size=300
iterations=1000
df_samp300_age1=[df_age1.sample(sample_size,replace=True)["Purchase"].mean() for i
df_samp300_age2=[df_age2.sample(sample_size,replace=True)["Purchase"].mean() for i
df_samp300_age3=[df_age3.sample(sample_size,replace=True)["Purchase"].mean() for i
df_samp300_age4=[df_age4.sample(sample_size,replace=True)["Purchase"].mean() for i
df_samp300_age5=[df_age5.sample(sample_size,replace=True)["Purchase"].mean() for i
df_samp300_age6=[df_age6.sample(sample_size,replace=True)["Purchase"].mean() for i
df_samp300_age7=[df_age7.sample(sample_size,replace=True)["Purchase"].mean() for i
```

Creating kde plots to check if it there is any Overlapping

In [ ]:
```
sns.kdeplot(df_samp300_age1)
sns.kdeplot(df_samp300_age2)
sns.kdeplot(df_samp300_age3)
sns.kdeplot(df_samp300_age4)
sns.kdeplot(df_samp300_age5)
sns.kdeplot(df_samp300_age6)
sns.kdeplot(df_samp300_age7)
```

Out[ ]:
```
<Axes: ylabel='Density'>
```



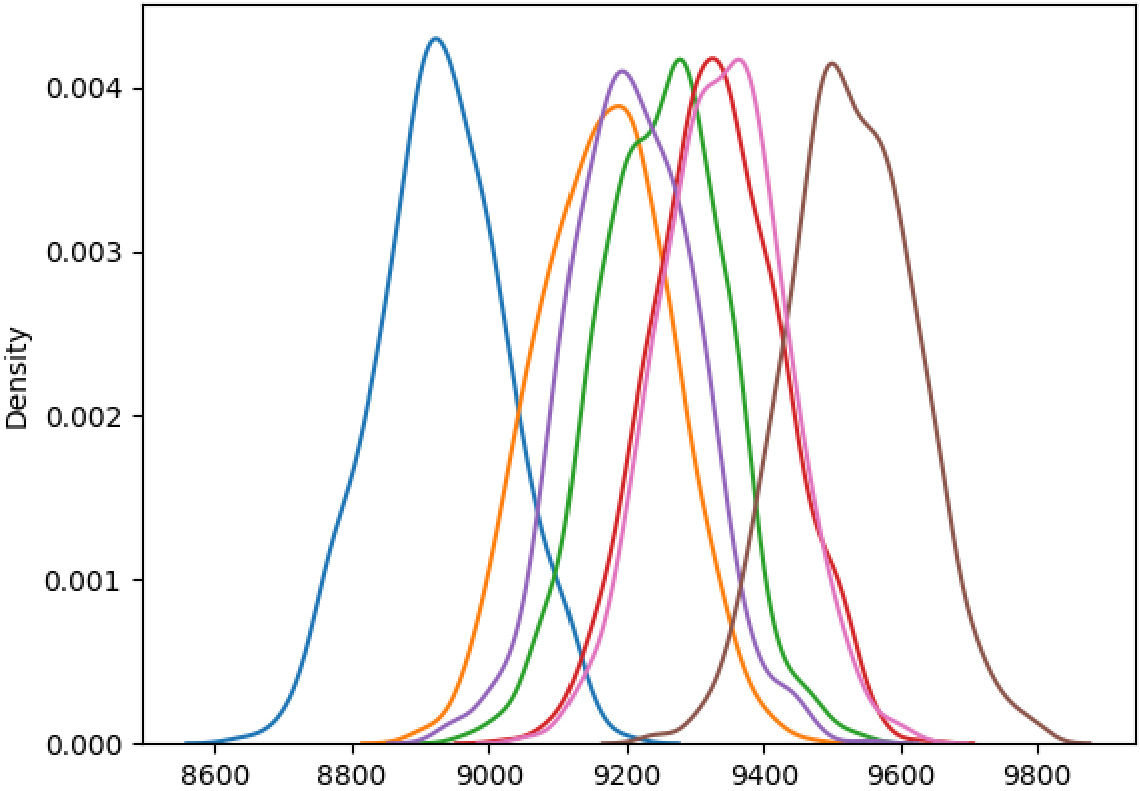
Taking samples of 3000 entries for each age group

In [ ]:
```
sample_size=3000
iterations=1000
df_samp3000_age1=[df_age1.sample(sample_size,replace=True)["Purchase"].mean() for i
df_samp3000_age2=[df_age2.sample(sample_size,replace=True)["Purchase"].mean() for i
df_samp3000_age3=[df_age3.sample(sample_size,replace=True)["Purchase"].mean() for i
df_samp3000_age4=[df_age4.sample(sample_size,replace=True)["Purchase"].mean() for i
df_samp3000_age5=[df_age5.sample(sample_size,replace=True)["Purchase"].mean() for i
df_samp3000_age6=[df_age6.sample(sample_size,replace=True)["Purchase"].mean() for i
df_samp3000_age7=[df_age7.sample(sample_size,replace=True)["Purchase"].mean() for i
```

Creating kde plots to check if it there is any Overlapping

In [ ]:
```
sns.kdeplot(df_samp3000_age1)
sns.kdeplot(df_samp3000_age2)
sns.kdeplot(df_samp3000_age3)
sns.kdeplot(df_samp3000_age4)
sns.kdeplot(df_samp3000_age5)
sns.kdeplot(df_samp3000_age6)
sns.kdeplot(df_samp3000_age7)
```

Out[ ]:
```
<Axes: ylabel='Density'>
```



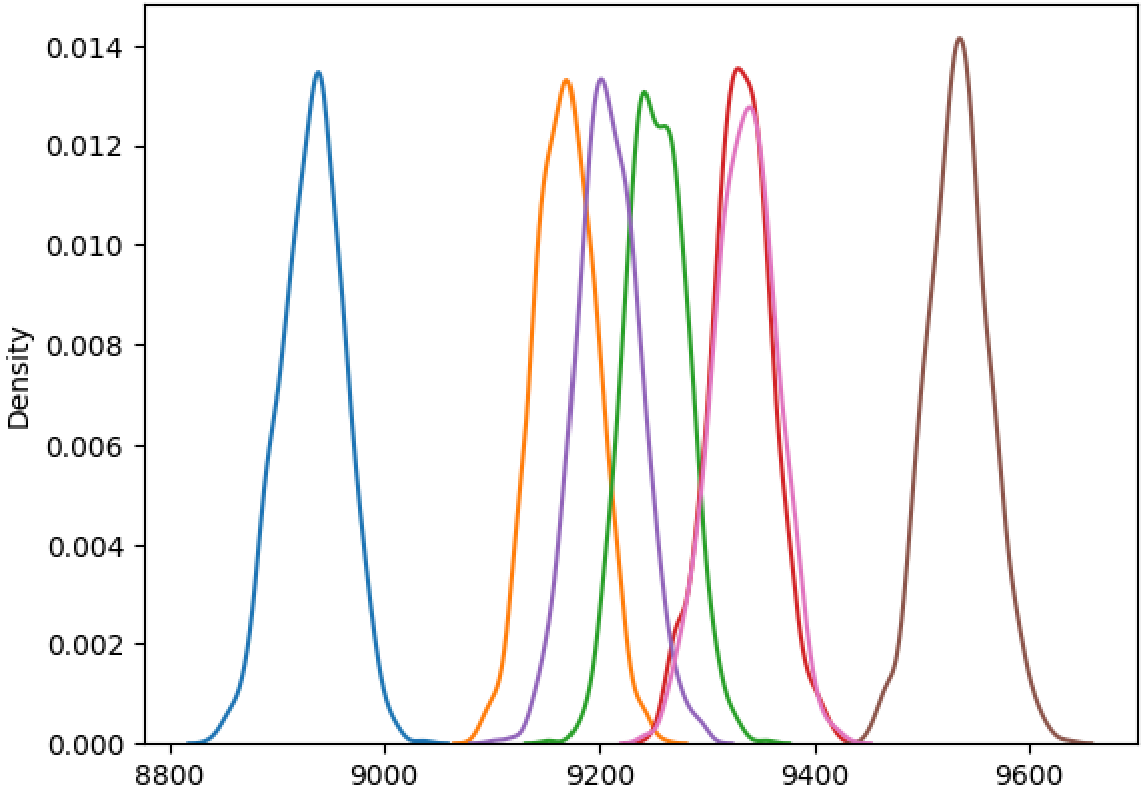
Taking samples of 30000 entries for each age group

In [ ]:
```
sample_size=30000
iterations=1000
df_samp30000_age1=[df_age1.sample(sample_size,replace=True)["Purchase"].mean() for
df_samp30000_age2=[df_age2.sample(sample_size,replace=True)["Purchase"].mean() for
df_samp30000_age3=[df_age3.sample(sample_size,replace=True)["Purchase"].mean() for
df_samp30000_age4=[df_age4.sample(sample_size,replace=True)["Purchase"].mean() for
df_samp30000_age5=[df_age5.sample(sample_size,replace=True)["Purchase"].mean() for
df_samp30000_age6=[df_age6.sample(sample_size,replace=True)["Purchase"].mean() for
df_samp30000_age7=[df_age7.sample(sample_size,replace=True)["Purchase"].mean() for
```

Creating kde plots to check if it there is any Overlapping

In [ ]:
```
sns.kdeplot(df_samp30000_age1)
sns.kdeplot(df_samp30000_age2)
sns.kdeplot(df_samp30000_age3)
sns.kdeplot(df_samp30000_age4)
sns.kdeplot(df_samp30000_age5)
sns.kdeplot(df_samp30000_age6)
sns.kdeplot(df_samp30000_age7)
```

Out[ ]: <Axes: ylabel='Density'>



Finding confidence intervals for mean purchase for each age group

In [ ]:
```
for i in ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']:
  print('For {m}-'.format(m = i))
  if i == '0-17':
    means = df_samp30000_age1
  elif i == '18-25':
    means = df_samp30000_age2
  elif i == '26-35':
    means = df_samp30000_age3
  elif i == '36-45':
    means = df_samp30000_age4
  elif i == '46-50':
    means = df_samp30000_age5
  elif i == '51-55':
    means = df_samp30000_age6
  else:
    means = df_samp30000_age7
  print('Mean of sample means =',np.mean(means))
  print('Population mean =', np.mean(df.loc[df['Age']==i, 'Purchase']))
  print('Standard deviation of means (Standard Error) =', np.std(means))
  print('Standard deviation of population =',df.loc[df['Age']==i, 'Purchase'].std()
  print('99% CONFIDENCE INTERVAL for mean expense by users of age group {a}-'.forma
  print((np.percentile(means, 0.5).round(2), np.percentile(means, 99.5).round(2)))
  print('95% CONFIDENCE INTERVAL for mean expense by users of age group {a}-'.forma
  print((np.percentile(means, 2.5).round(2), np.percentile(means, 97.5).round(2)))
  print('90% CONFIDENCE INTERVAL for mean expense by users of age group {a}-'.forma
  print((np.percentile(means, 5).round(2), np.percentile(means, 95).round(2)))
  print('-'*50)
```

```
For 0-17-
Mean of sample means = 8933.11144
Population mean = 8933.464640444974
Standard deviation of means (Standard Error) = 29.948209745506333
Standard deviation of population = 5111.11404600277
99% CONFIDENCE INTERVAL for mean expense by users of age group 0-17-
(8855.07, 9006.47)
95% CONFIDENCE INTERVAL for mean expense by users of age group 0-17-
(8873.76, 8988.92)
90% CONFIDENCE INTERVAL for mean expense by users of age group 0-17-
(8883.82, 8980.9)
--------------------------------------------------
For 18-25-
Mean of sample means = 9168.8991614
Population mean = 9169.663606261289
Standard deviation of means (Standard Error) = 28.438341196098012
Standard deviation of population = 5034.32199717658
99% CONFIDENCE INTERVAL for mean expense by users of age group 18-25-
(9095.98, 9240.85)
95% CONFIDENCE INTERVAL for mean expense by users of age group 18-25-
(9116.57, 9223.66)
90% CONFIDENCE INTERVAL for mean expense by users of age group 18-25-
(9123.17, 9215.95)
--------------------------------------------------
For 26-35-
Mean of sample means = 9252.912666299999
Population mean = 9252.690632869888
Standard deviation of means (Standard Error) = 27.901943389139415
Standard deviation of population = 5010.527303002956
99% CONFIDENCE INTERVAL for mean expense by users of age group 26-35-
(9188.79, 9325.09)
95% CONFIDENCE INTERVAL for mean expense by users of age group 26-35-
(9201.19, 9307.42)
90% CONFIDENCE INTERVAL for mean expense by users of age group 26-35-
(9207.63, 9298.71)
--------------------------------------------------
For 36-45-
Mean of sample means = 9332.042720833333
Population mean = 9331.350694917874
Standard deviation of means (Standard Error) = 29.266955913388
Standard deviation of population = 5022.923879204662
99% CONFIDENCE INTERVAL for mean expense by users of age group 36-45-
(9260.39, 9407.74)
95% CONFIDENCE INTERVAL for mean expense by users of age group 36-45-
(9271.79, 9391.35)
90% CONFIDENCE INTERVAL for mean expense by users of age group 36-45-
(9279.47, 9379.95)
--------------------------------------------------
For 46-50-
Mean of sample means = 9208.697521633334
Population mean = 9208.625697468327
Standard deviation of means (Standard Error) = 29.52197981177885
Standard deviation of population = 4967.216367142941
99% CONFIDENCE INTERVAL for mean expense by users of age group 46-50-
(9138.49, 9290.65)
95% CONFIDENCE INTERVAL for mean expense by users of age group 46-50-
(9152.82, 9268.0)
90% CONFIDENCE INTERVAL for mean expense by users of age group 46-50-
(9161.53, 9257.58)
--------------------------------------------------
For 51-55-
Mean of sample means = 9534.982672266668
Population mean = 9534.808030960236
Standard deviation of means (Standard Error) = 29.25803124301832
```

```
Standard deviation of population = 5087.368079602135
99% CONFIDENCE INTERVAL for mean expense by users of age group 51-55-
(9461.95, 9612.17)
95% CONFIDENCE INTERVAL for mean expense by users of age group 51-55-
(9477.98, 9592.78)
90% CONFIDENCE INTERVAL for mean expense by users of age group 51-55-
(9488.24, 9584.64)
--------------------------------------------------
For 55+-
Mean of sample means = 9335.694198833333
Population mean = 9336.280459449405
Standard deviation of means (Standard Error) = 30.206371691101346
Standard deviation of population = 5011.4939956034605
99% CONFIDENCE INTERVAL for mean expense by users of age group 55+-
(9261.04, 9414.23)
95% CONFIDENCE INTERVAL for mean expense by users of age group 55+-
(9275.73, 9392.21)
90% CONFIDENCE INTERVAL for mean expense by users of age group 55+-
(9284.8, 9384.45)
--------------------------------------------------
```

Insights

1.Sample size

- As the sample size increases the confidence intervals become narrower and precise.This suggests that the larger sample sizes can provide more reliable insights 2.Confidence Intervals
- 0 - 17 - Customers in this age group have the lowest spending per transaction 18 - 25, 26 - 35, 46 - 50 - Customers in these age groups have overlapping confidence intervals indicating similar buying characteristics.
- 36 - 45, 55+ - Customers in these age groups have overlapping confidence intervals indicating and similar spending patterns

3.Population Average

- We are 95% confident that the true population average for following age groups falls between the below range
- 0 - 17 = $ 8,874 to 8,989
- 18 - 25 = $ 9,117 to 9,224
- 26 - 35 = $ 9,202 to 9,306
- 36 - 45 = $ 9,272 to 9,392
- 46 - 50 = $ 9,153 to 9,269
- 51 - 55 = $ 9,478 to 9,593
- 55+ = $ 9,276 to 9,393

Recommendations -

- Walmart can keep the most purchased product categories which selling a lot, rather than focusing on average products.
- Make advertisements which should be targeted to the people of age group 26-35.
- Products of categories 1, 5 and 8 can be kept in inventory as well as made easily visible in the stores.
- More products popular among people with occupations 0, 4 and 7 can be kept in store.
- Advertisements can be targeted towards people who have spent between 1 to 2 years in their cities
- Can focus on unmarried people and married people.
- Can start offers and rewards on purchases above higher products.