Walmart - Confidence Interval and CLT

[57]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

[2]:
```python
df= pd.read_csv('walmart_data.csv')
```

[3]:
```python
df
```

[3]:
```
        User_ID Product_ID Gender    Age  Occupation City_Category  \
0       1000001  P00069042      F   0-17          10             A
1       1000001  P00248942      F   0-17          10             A
2       1000001  P00087842      F   0-17          10             A
3       1000001  P00085442      F   0-17          10             A
4       1000002  P00285442      M    55+          16             C
…           …         …    …     …         …           …
550063  1006033  P00372445      M  51-55          13             B
550064  1006035  P00375436      F  26-35           1             C
550065  1006036  P00375436      F  26-35          15             B
550066  1006038  P00375436      F    55+           1             C
550067  1006039  P00371644      F  46-50           0             B

       Stay_In_Current_City_Years  Marital_Status  Product_Category  Purchase
0                               2               0                 3      8370
1                               2               0                 1     15200
2                               2               0                12      1422
3                               2               0                12      1057
4                              4+               0                 8      7969
…                             …               …                …       …
550063                          1               1                20       368
550064                          3               0                20       371
550065                         4+               1                20       137
550066                          2               0                20       365
550067                         4+               1                20       490
```

```
[550068 rows x 10 columns]
```

[4]:
```
print(f'No of Rows are : {df.shape[0]}')
print(f'No of Columns are : {df.shape[1]}')
```

```
No of Rows are : 550068
No of Columns are : 10
```

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     550068 non-null  int64
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64
 8   Product_Category            550068 non-null  int64
 9   Purchase                    550068 non-null  int64
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

[6]:
```
df.describe().T
```

[6]:
```
                     count          mean          std        min        25%  \
User_ID           550068.0  1.003029e+06  1727.591586  1000001.0  1001516.0
Occupation        550068.0  8.076707e+00     6.522660        0.0        2.0
Marital_Status    550068.0  4.096530e-01     0.491770        0.0        0.0
Product_Category  550068.0  5.404270e+00     3.936211        1.0        1.0
Purchase          550068.0  9.263969e+03  5023.065394       12.0     5823.0

                        50%        75%        max
User_ID           1003077.0  1004478.0  1006040.0
Occupation              7.0       14.0       20.0
Marital_Status          0.0        1.0        1.0
Product_Category        5.0        8.0       20.0
Purchase             8047.0    12054.0    23961.0
```

[6]:
```

```

[7]:
```
df['Age'].unique()
```

[7]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

[8]:
```
df['Age'].nunique()
```

[8]:
```
7
```

[9]:
```
df['Age'].value_counts()
```

[9]:
```
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
```

[10]:
```
df['Product_ID'].unique()
```

[10]:
```
array(['P00069042', 'P00248942', 'P00087842', …, 'P00370293',
       'P00371644', 'P00370853'], dtype=object)
```

[11]:
```
df['Product_ID'].nunique()
```

[11]:
```
3631
```

[12]:
```
df['Product_ID'].value_counts()
```

[12]:
```
Product_ID
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
             …
P00314842       1
P00298842       1
P00231642       1
P00204442       1
P00066342       1
Name: count, Length: 3631, dtype: int64
```

[13]:
```
df['Gender'].unique()
```

[13]: array(['F', 'M'], dtype=object)

[14]:
```
df['Gender'].value_counts()
```

[14]:
```
Gender
M    414259
F    135809
Name: count, dtype: int64
```

[15]:
```
df['Occupation'].unique()
```

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

[16]:
```
df['Occupation'].nunique()
```

[16]: 21

[17]:
```
df['Occupation'].value_counts()
```

[17]:
```
Occupation
4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
2     26588
16    25371
6     20355
3     17650
10    12930
5     12177
15    12165
11    11586
19     8461
13     7728
18     6622
9      6291
8      1546
Name: count, dtype: int64
```

[18]:
```
df['City_Category'].unique()
```

[18]: array(['A', 'C', 'B'], dtype=object)

[19]:
```python
df['City_Category'].nunique()
```

[19]:
```
3
```

[20]:
```python
df['City_Category'].value_counts()
```

[20]:
```
City_Category
B    231173
C    171175
A    147720
Name: count, dtype: int64
```

[21]:
```python
df['Stay_In_Current_City_Years'].unique()
```

[21]:
```
array(['2', '4+', '3', '1', '0'], dtype=object)
```

[22]:
```python
df['Stay_In_Current_City_Years'].value_counts()
```

[22]:
```
Stay_In_Current_City_Years
1     193821
2     101838
3      95285
4+     84726
0      74398
Name: count, dtype: int64
```

[23]:
```python
df['Marital_Status'].unique()
```

[23]:
```
array([0, 1])
```

[24]:
```python
df['Marital_Status'].value_counts()
```

[24]:
```
Marital_Status
0    324731
1    225337
Name: count, dtype: int64
```

[25]:
```python
df['Product_Category'].unique()
```

[25]:
```
array([ 3,  1, 12,  8,  5,  4,  2,  6, 14, 11, 13, 15,  7, 16, 18, 10, 17,
        9, 20, 19])
```

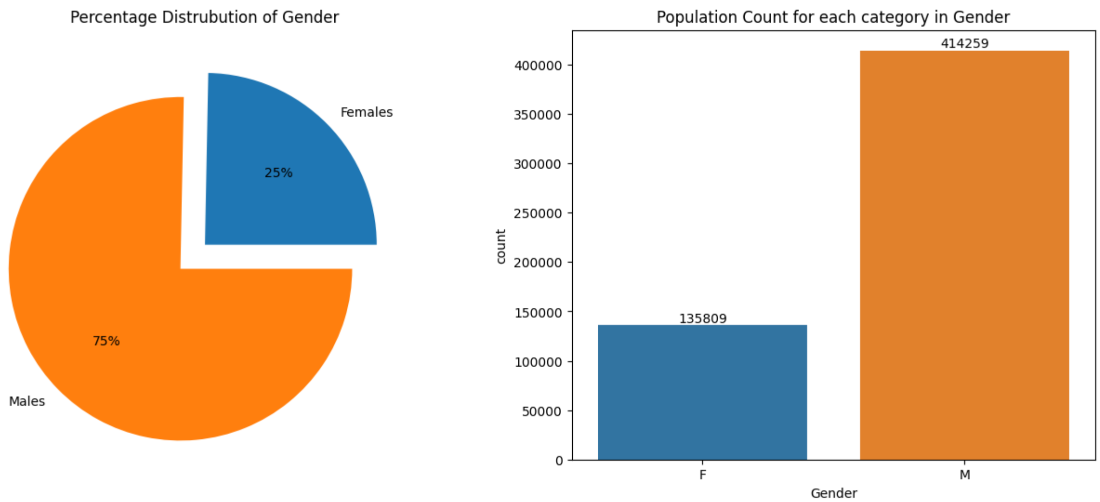
[26]:
```python
df['Product_Category'].nunique()
```

[26]:
```
20
```

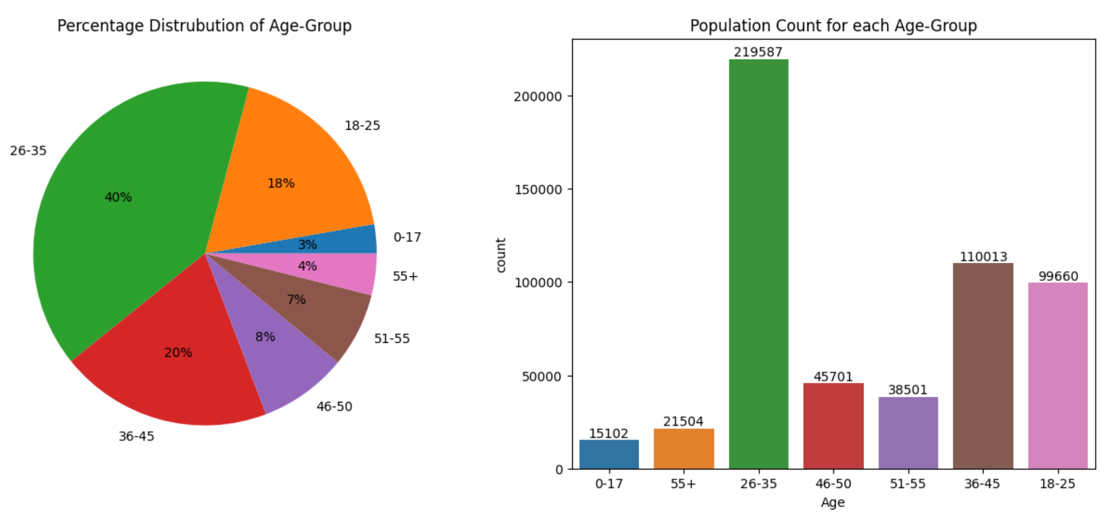
[27]:
```python
df['Product_Category'].value_counts()
```

[27]:
```
Product_Category
5     150933
1     140378
8     113925
11     24287
2      23864
6      20466
3      20213
4      11753
16      9828
15      6290
13      5549
10      5125
12      3947
7       3721
18      3125
20      2550
19      1603
14      1523
17       578
9        410
Name: count, dtype: int64
```

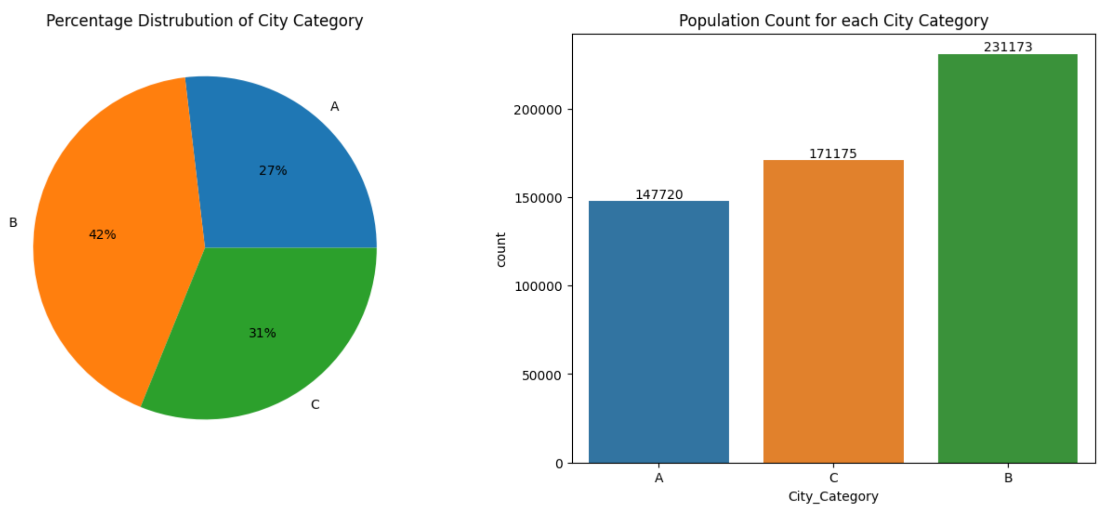
[28]:
```
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
labels=['Females','Males']
plt.pie(df.groupby('Gender')['Gender'].count(),labels=labels,autopct = '%0.
 ↪0f%%',explode=[0,0.2])
plt.title('Percentage Distrubution of Gender')
plt.subplot(1,2,2)
label= sns.countplot(x=df['Gender'] ,hue=df['Gender'])
for i in label.containers:
 label.bar_label(i)
plt.title('Population Count for each category in Gender')
plt.show()
```



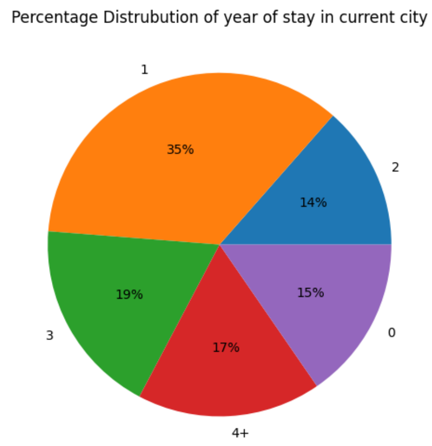
[29]:
```
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
labels = ['0-17','18-25','26-35','36-45','46-50','51-55','55+']
plt.pie(df.groupby('Age')['Age'].count(),labels=labels,autopct='%0.0f%%')
plt.title('Percentage Distrubution of Age-Group')
plt.subplot(1,2,2)
label=sns.countplot(x=df['Age'],hue=df['Age'])
for i in label.containers:
  label.bar_label(i)
plt.title('Population Count for each Age-Group')
plt.show()
```

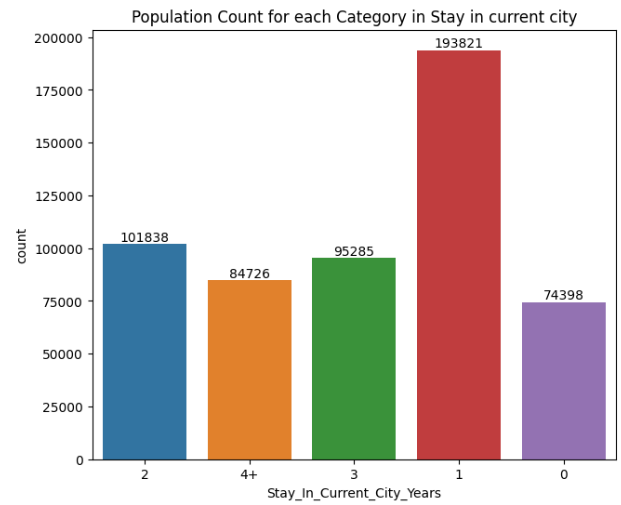


[30]:
```
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
labels = ['A','B','C']
plt.pie(df.groupby('City_Category')['City_Category'].
  ↪count(),labels=labels,autopct='%0.0f%%')
plt.title('Percentage Distrubution of City Category')
plt.subplot(1,2,2)
label=sns.countplot(x=df['City_Category'],hue=df['City_Category'])
for i in label.containers:
  label.bar_label(i)
plt.title('Population Count for each City Category')
plt.show()
```



[31]:
```
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
labels = ['2','1','3','4+','0']
plt.pie(df.groupby('Stay_In_Current_City_Years')['Stay_In_Current_City_Years'].
  ↪count(),labels=labels,autopct='%0.0f%%')
plt.title('Percentage Distrubution of year of stay in current city')
plt.subplot(1,2,2)
label=sns.
  ↪countplot(x=df['Stay_In_Current_City_Years'],hue=df['Stay_In_Current_City_Years'])
for i in label.containers:
  label.bar_label(i)
plt.title('Population Count for each Category in Stay in current city')
plt.show()
```

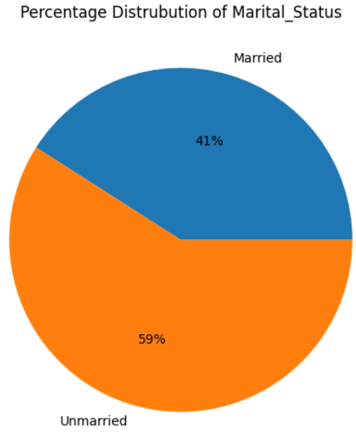


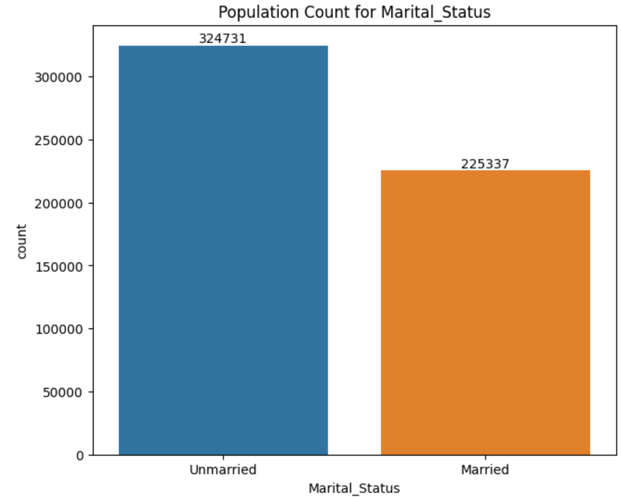


[32]:
```
df['Marital_Status'].value_counts()
```

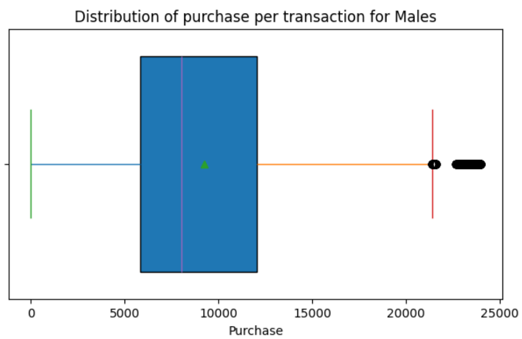
[32]:
```
Marital_Status
0    324731
1    225337
Name: count, dtype: int64
```

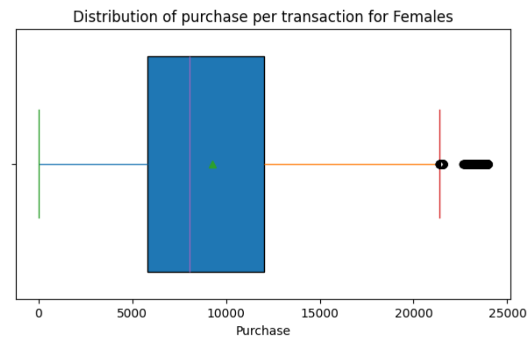
[33]:
```
df['Marital_Status'].replace(to_replace = 0, value = 'Unmarried',inplace = True)
df['Marital_Status'].replace(to_replace = 1, value = 'Married',inplace = True)

plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
labels = ['Married','Unmarried']
plt.pie(df.groupby('Marital_Status')['Marital_Status'].
 ↪count(),labels=labels,autopct='%0.0f%%')
plt.title('Percentage Distrubution of Marital_Status')
plt.subplot(1,2,2)
label=sns.countplot(x=df['Marital_Status'],hue=df['Marital_Status'])
for i in label.containers:
  label.bar_label(i)
plt.title('Population Count for Marital_Status')
plt.show()
```



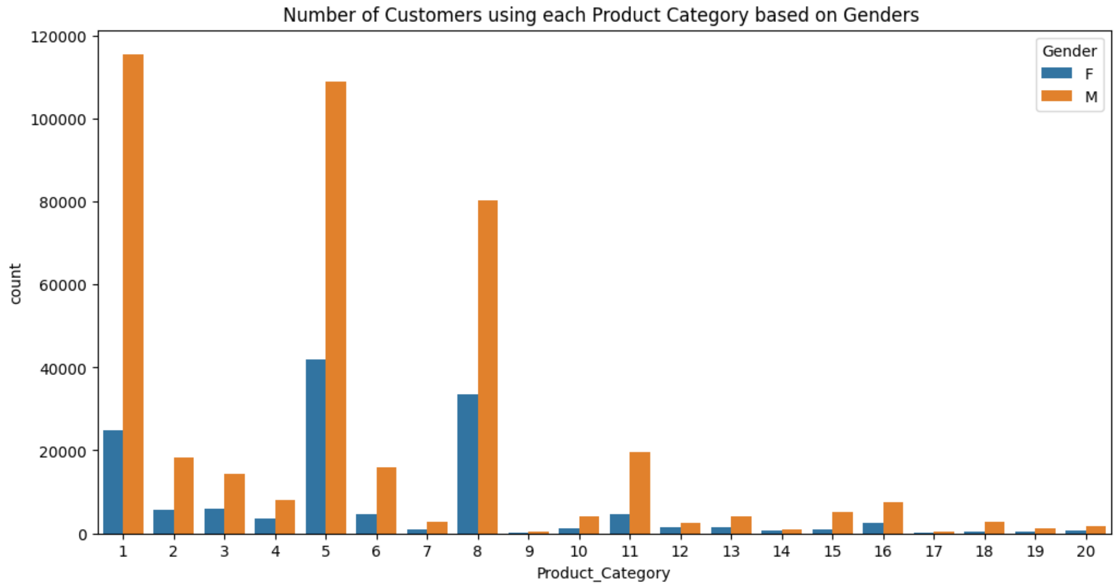


[34]:
```
plt.figure(figsize=(16,4))
plt.subplot(1,2,1)
plt.title('Distribution of purchase per transaction for Males')
sns.boxplot(x=df['Purchase'],hue=df[df['Gender']=='Male'],showmeans=True)
plt.subplot(1,2,2)
plt.title('Distribution of purchase per transaction for Females')
sns.boxplot(x=df['Purchase'],hue=df[df['Gender']=='Female'],showmeans=True)
plt.show()
```



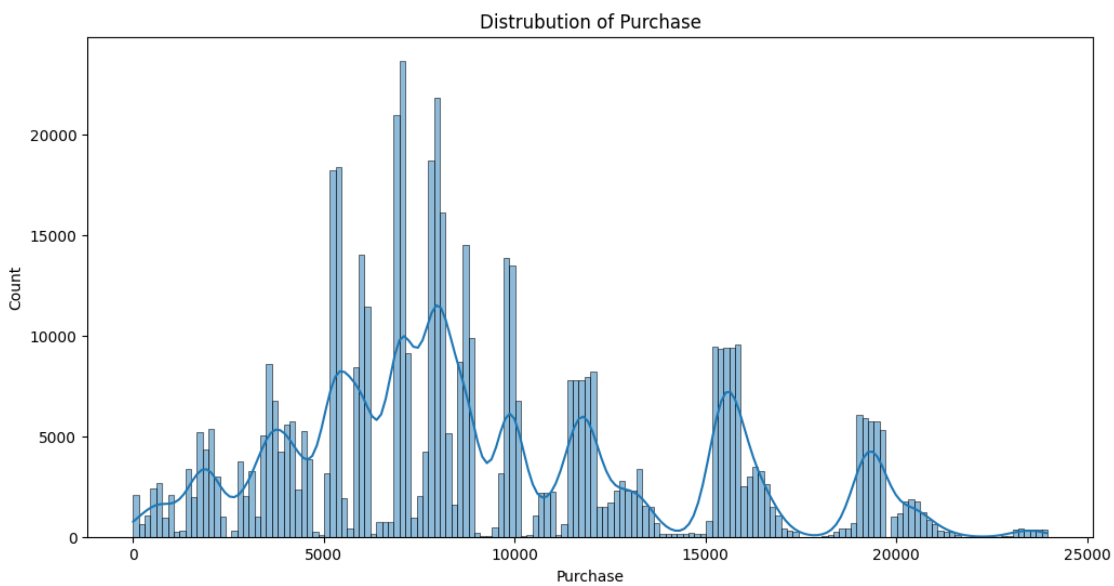


[35]:
```
plt.figure(figsize=(12,6))
label= sns.countplot(x=df['Product_Category'],hue=df['Gender'])
plt.title('Number of Customers using each Product Category based on Genders ')
plt.show()
```



- Product_Category 1,5,8 are more prefered by customers aprt from other categories.
- Product_Category 9,17 are least prefred by the customers.

[36]:
```
plt.figure(figsize=(12,6))
sns.histplot(x=df['Purchase'],kde=True)
plt.title('Distrubution of Purchase')
plt.show()
```



0.1 Bi-Variate Analysis

[37]:
```
df.head()
```

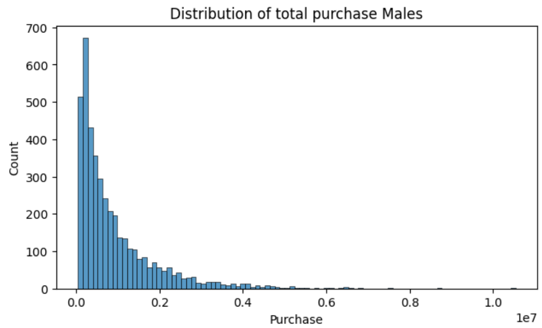
[37]:
```
   User_ID Product_ID Gender   Age  Occupation City_Category  \
0  1000001  P00069042      F  0-17          10             A
1  1000001  P00248942      F  0-17          10             A
2  1000001  P00087842      F  0-17          10             A
3  1000001  P00085442      F  0-17          10             A
4  1000002  P00285442      M   55+          16             C

  Stay_In_Current_City_Years Marital_Status  Product_Category  Purchase
0                          2      Unmarried                 3      8370
1                          2      Unmarried                 1     15200
2                          2      Unmarried                12      1422
3                          2      Unmarried                12      1057
4                         4+      Unmarried                 8      7969
```

[38]:
```
df1= pd.DataFrame(df.groupby(['User_ID','Gender'])['Purchase'].sum()).
 ↪reset_index()
df1
```

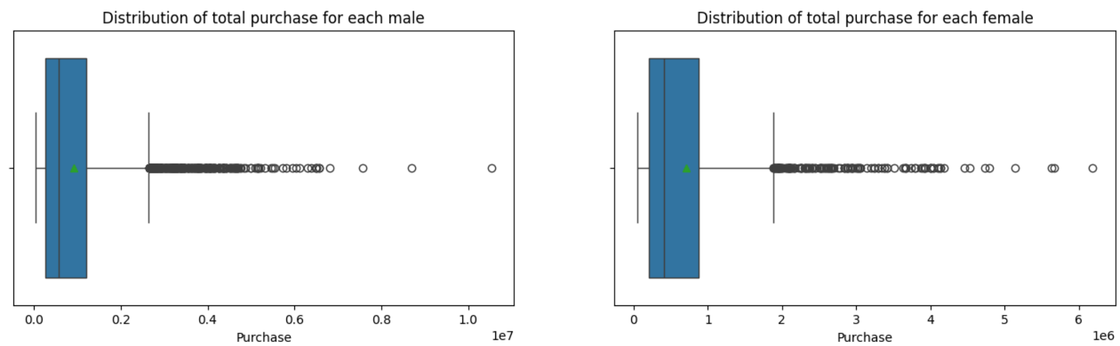
[38]:
```
      User_ID Gender  Purchase
0     1000001      F    334093
1     1000002      M    810472
2     1000003      M    341635
3     1000004      M    206468
4     1000005      M    821001
…         …    …         …
5886  1006036      F   4116058
5887  1006037      F   1119538
5888  1006038      F     90034
5889  1006039      F    590319
5890  1006040      M   1653299

[5891 rows x 3 columns]
```

[39]:
```
df1_male = df1[df1['Gender'] == 'M']
df1_female = df1.loc[df1['Gender'] == 'F']


plt.figure(figsize=(16,4))
plt.subplot(1,2,1)
plt.title('Distribution of total purchase Males')
sns.histplot(data=df1_male,x='Purchase')
plt.subplot(1,2,2)
plt.title('Distribution of total purchase Females')
```

```
sns.histplot(data=df1_female,x='Purchase')
plt.show()
```



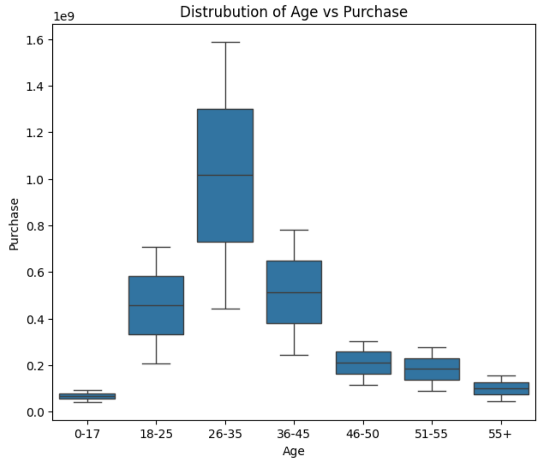
[40]:
```
plt.figure(figsize = (16, 4))
plt.subplot(1, 2, 1)
plt.title('Distribution of total purchase for each male')
sns.boxplot(data = df1_male, x = 'Purchase', showmeans = True)
plt.subplot(1, 2 ,2)
plt.title('Distribution of total purchase for each female')
sns.boxplot(data = df1_female, x = 'Purchase', showmeans = True)
plt.show()
```

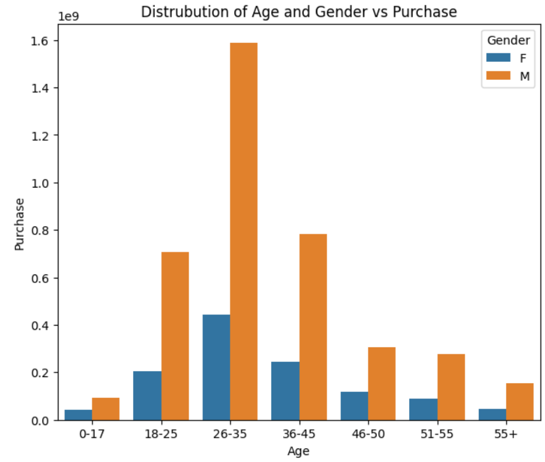


[40]:
```

```

[41]:
```
df2= pd.DataFrame(df.groupby(['Age','Gender'])['Purchase'].sum()).reset_index()

plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
sns.boxplot(x=df2['Age'],y=df2['Purchase'])
plt.title('Distrubution of Age vs Purchase')
plt.subplot(1,2,2)
sns.barplot(x=df2['Age'],y=df2['Purchase'],hue=df2['Gender'])
```

```
plt.title('Distrubution of Age and Gender vs Purchase')
plt.show()
```





- In Age group of 26-35 males purchased is siginficatly more and in age group of 0-17 and 55+ the purchase was very low.

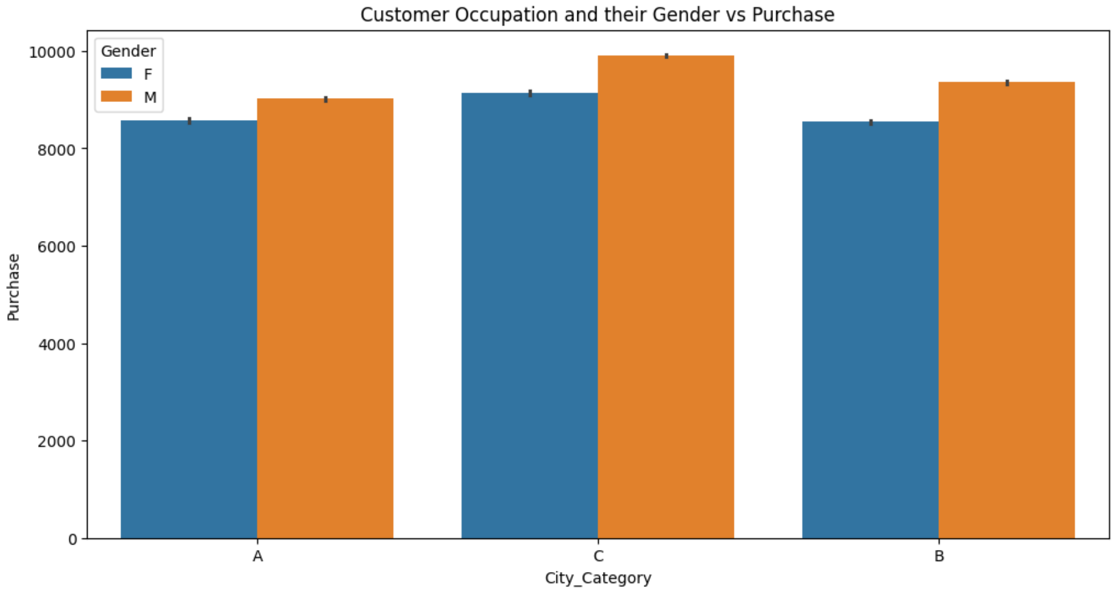
[42]:
```
df.groupby('City_Category')['Purchase'].mean()
```

[42]:
```
City_Category
A    8911.939216
B    9151.300563
C    9719.920993
Name: Purchase, dtype: float64
```

[43]:
```
df.groupby('Stay_In_Current_City_Years')['Purchase'].mean().round(2)
```

[43]:
```
Stay_In_Current_City_Years
0     9180.08
1     9250.15
2     9320.43
3     9286.90
4+    9275.60
Name: Purchase, dtype: float64
```

[44]:
```
plt.figure(figsize=(12,6))
sns.barplot(x=df['City_Category'],y=df['Purchase'],hue=df['Gender'])
plt.title('Customer Occupation and their Gender vs Purchase')
plt.show()
```

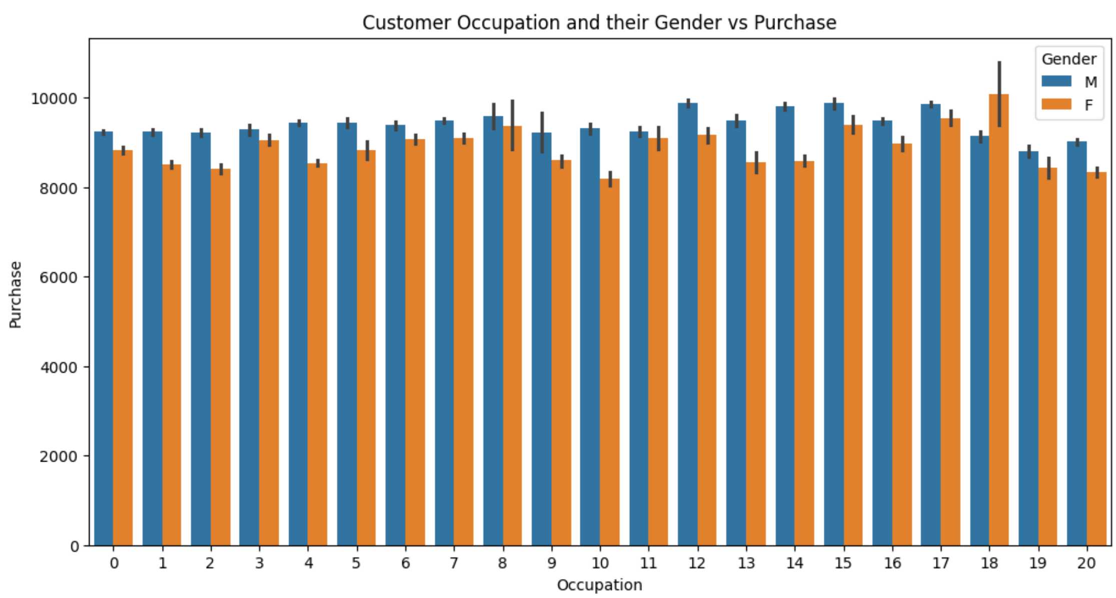


[45]:
```
plt.figure(figsize=(12,6))
sns.scatterplot(x=df['Occupation'],y=df['Product_Category'],hue=df['Gender'])
plt.title('Customer Occupation and their Gender vs Purchase')
plt.show()
```

[46]:
```
df.head()
```

[46]:
```
   User_ID Product_ID Gender   Age  Occupation City_Category  \
0  1000001  P00069042      F  0-17          10             A
1  1000001  P00248942      F  0-17          10             A
2  1000001  P00087842      F  0-17          10             A
3  1000001  P00085442      F  0-17          10             A
4  1000002  P00285442      M   55+          16             C

  Stay_In_Current_City_Years Marital_Status  Product_Category  Purchase
0                          2      Unmarried                 3      8370
1                          2      Unmarried                 1     15200
2                          2      Unmarried                12      1422
3                          2      Unmarried                12      1057
4                         4+      Unmarried                 8      7969
```

[47]:
```python
plt.figure(figsize=(12,6))
sns.barplot(x=df['Occupation'],y=df['Purchase'],hue=df['Gender'])
plt.title('Customer Occupation and their Gender vs Purchase')
plt.show()
```



0.2 Gender *Vs* Purchase Amount

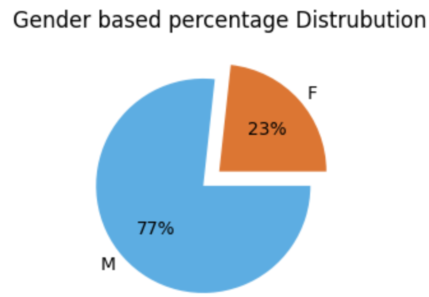
[48]:
```python
a= pd.DataFrame(df.groupby('Gender')['Purchase'].agg(['sum','mean','count']).
 ↪reset_index())
a['Precentage_distrubution']= np.round((a['sum']/a['sum'].sum())*100,2)
a
```

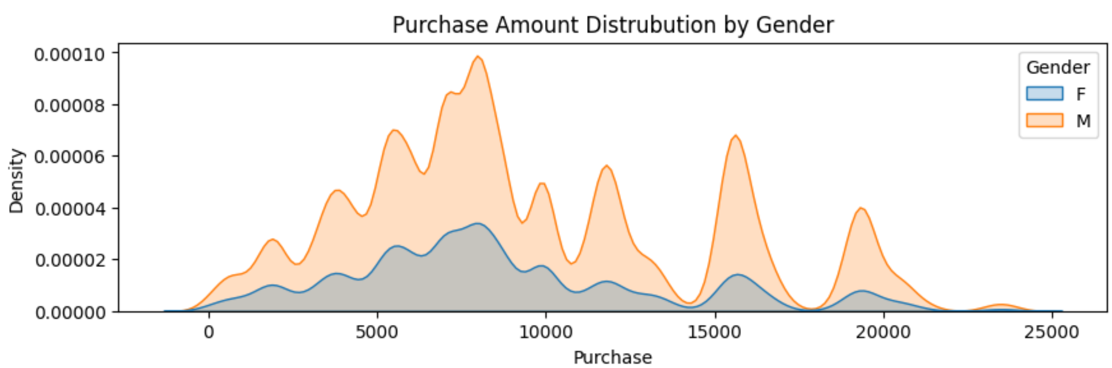
[48]:

	Gender	sum	mean	count	Precentage_distrubution
0	F	1186232642	8734.565765	135809	23.28
1	M	3909580100	9437.526040	414259	76.72

[49]:

```
plt.figure(figsize=(10,6))
plt.subplot(2,1,1)
colors= ['#DC7633','#5DADE2']
plt.pie(a['Precentage_distrubution'],labels= a['Gender'],autopct = '%0.
↪0f%%',explode=[0,0.2],colors=colors)
plt.title('Gender based percentage Distrubution')
plt.subplot(2,1,2)
sns.kdeplot(x=df['Purchase'],hue=df['Gender'],fill=True)
plt.title('Purchase Amount Distrubution by Gender')
plt.show()
```





- Total Purchase amount made by male is more than female.
- The Average transaction amount `male is $ 9437.52` and average transaction amount by `female is $ 8734.56`

0.3 Construction of confidence Interval for Males and females Purchases: CLT

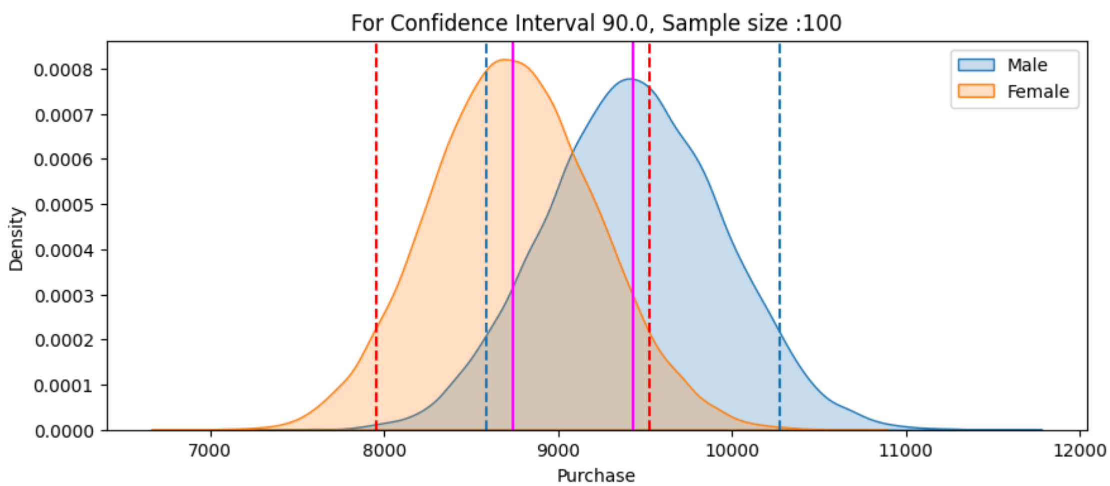
- From the plot we can see that the distrubution of purchase amount for males and females on black friday is not Normal.
- so we use `Central Limit Therom`.
- It states the distribution of sample means will approximate a normal distribution, regardless

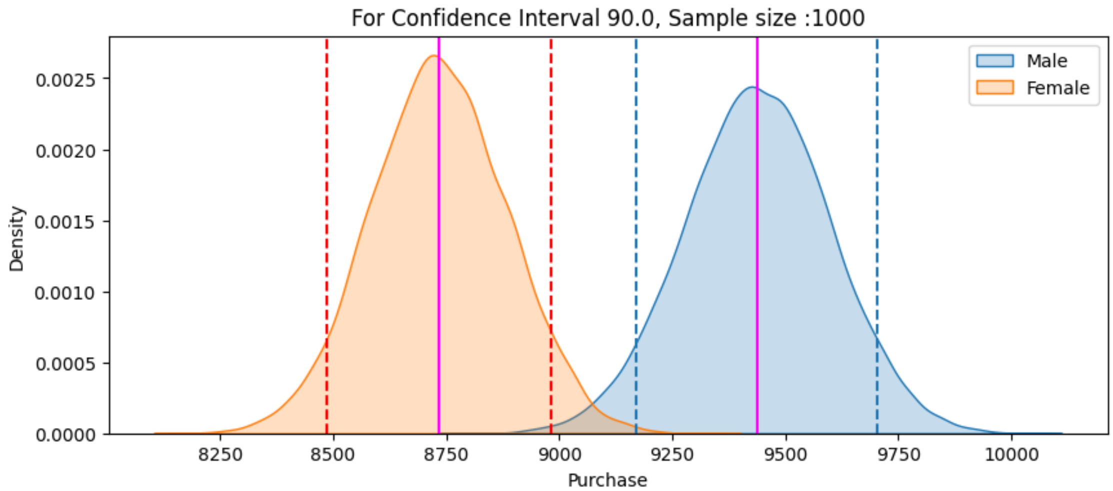
of the underlying population distribution

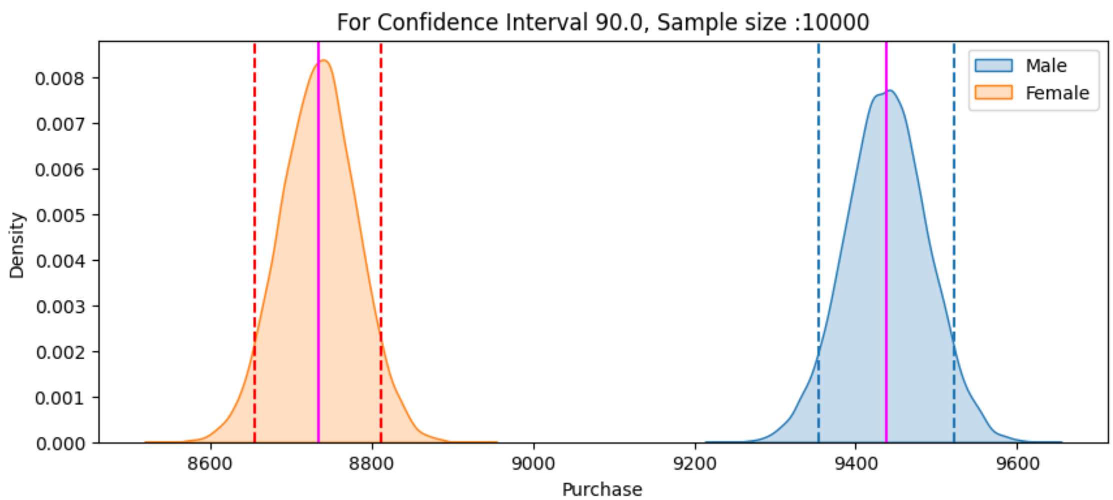
[50]:
```
from scipy.stats import norm
def gen_plot(sample1,sample2,sample_size,n_size,ci):
  plt.figure(figsize=(10,4))
  ci=ci/100
  sample1_means=[]
  sample2_means=[]
  for i in range(n_size):
    sample1_means.append(np.mean(sample1.sample(sample_size,replace=True)))
    sample2_means.append(np.mean(sample2.sample(sample_size,replace=True)))
  #for sample 1
  mean_1 = np.mean(sample1_means)
  std_1 = np.std(sample1_means)
  s_error_1 = std_1/ np.sqrt(len(sample1_means))

  lower_1 = norm.ppf((1-ci)/2)* std_1 + mean_1
  upper_1 = norm.ppf(1-(1-ci)/2)* std_1 + mean_1

  #for sample 2
  mean_2 = np.mean(sample2_means)
  std_2 = np.std(sample2_means)
  s_error_2 = std_2/ np.sqrt(len(sample2_means))

  lower_2 = norm.ppf((1-ci)/2)* std_2 + mean_2
  upper_2 = norm.ppf(1-(1-ci)/2)* std_2 + mean_2

  sns.kdeplot(data=sample1_means,fill=True,label='Male')
  plt.axvline(mean_1,color='#FF00FF')
  plt.axvline(lower_1,linestyle='--')
  plt.axvline(upper_1,linestyle='--')

  sns.kdeplot(data=sample2_means,fill=True,label='Female')
  plt.axvline(mean_2,color='#FF00FF')
  plt.axvline(lower_2,linestyle='--',color = 'red')
  plt.axvline(upper_2,linestyle='--',color = 'red')

  plt.title(f'For Confidence Interval {ci*100}, Sample size :{sample_size}')
  plt.legend()
  plt.xlabel('Purchase')
  plt.ylabel('Density')

  return round(mean_1,2), round(mean_2,2), round(lower_1,2),round(upper_1,2),␣
 ↪round(lower_2,2), round(upper_2,2)
```

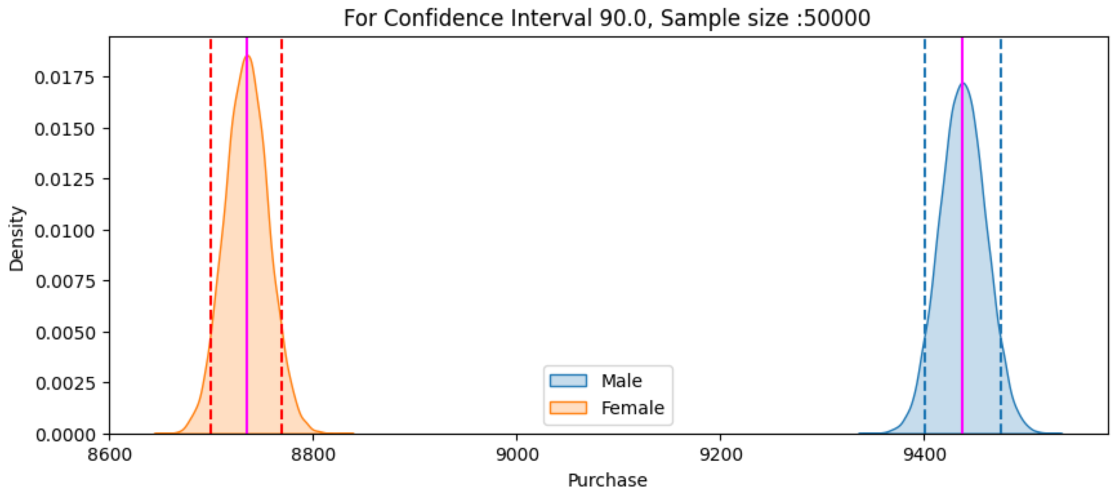
0.4 Confidence Interval 90%

[59]:
```
sample_sizes = [100,1000,10000,50000]
ci = 90
n_size = 20000

df_male = df[df['Gender']=='M']
df_female = df[df['Gender']=='F']

df_result = pd.DataFrame(columns = ['Gender','Sample Size','LowerLimit','Upper␣
↪Limit','Sample Mean','Interval Range','Confidence Interval'])

for i in sample_sizes:
  mean_1,mean_2,lower_1,upper_1,lower_2,upper_2 =␣
↪gen_plot(df_male['Purchase'],df_female['Purchase'],i,n_size,ci)
  df_result = pd.concat([df_result,pd.DataFrame({'Gender':'M','Sample Size':
↪i,'LowerLimit':lower_1,'Upper Limit':upper_1,'Sample Mean':mean_1,'Interval␣
↪Range':[(lower_1,upper_1)],'Confidence Interval':ci})],ignore_index = True)
  df_result = pd.concat([df_result,pd.DataFrame({'Gender':'F','Sample Size':
↪i,'LowerLimit':lower_2,'Upper Limit':upper_2,'Sample Mean':mean_2,'Interval␣
↪Range':[(lower_2,upper_2)],'Confidence Interval':ci})],ignore_index = True)
```



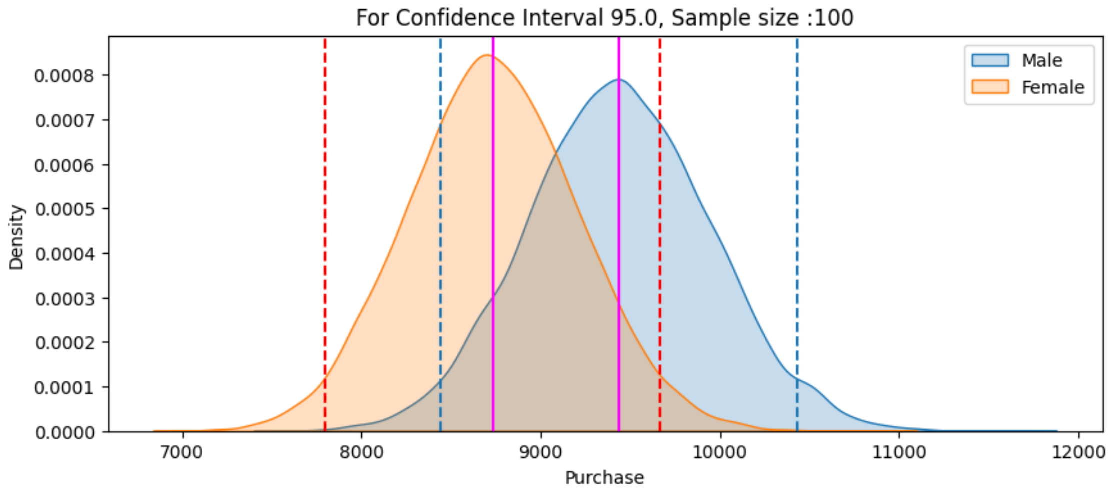

For Confidence Interval 90.0, Sample size :1000
Male
Female
Density
Purchase
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
8250
8500
8750
9000
9250
9500
9750
10000

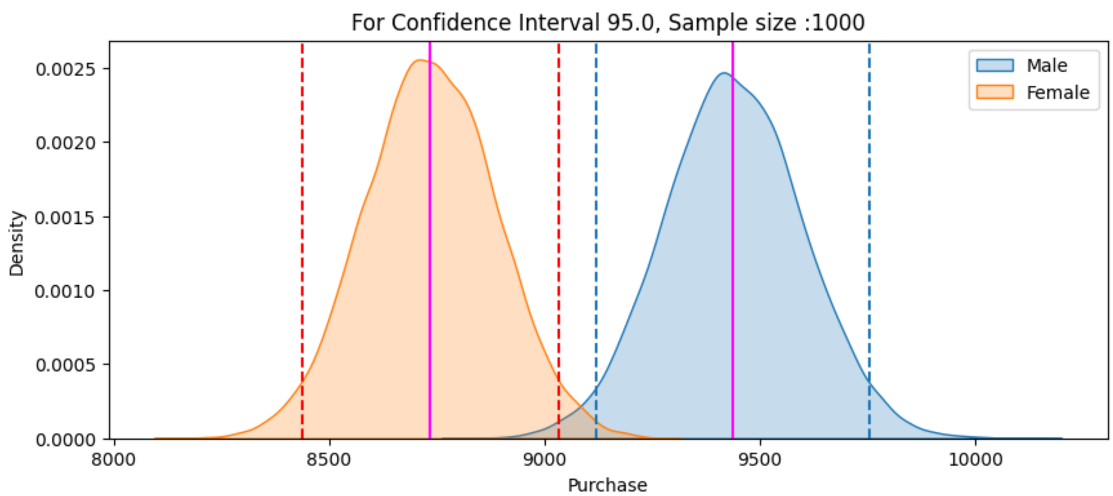

For Confidence Interval 90.0, Sample size :10000
Male
Female
Density
Purchase
0.000
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
8600
8800
9000
9200
9400
9600

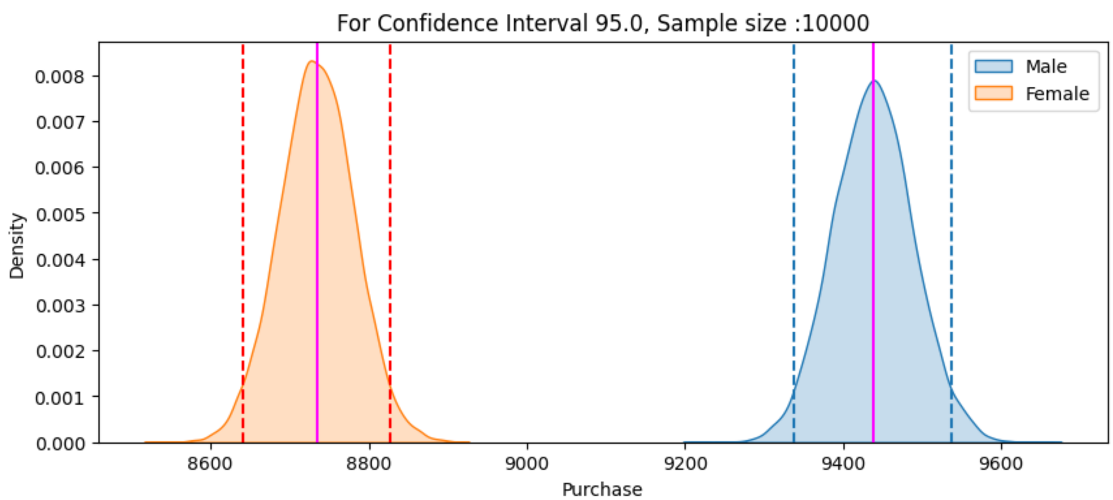


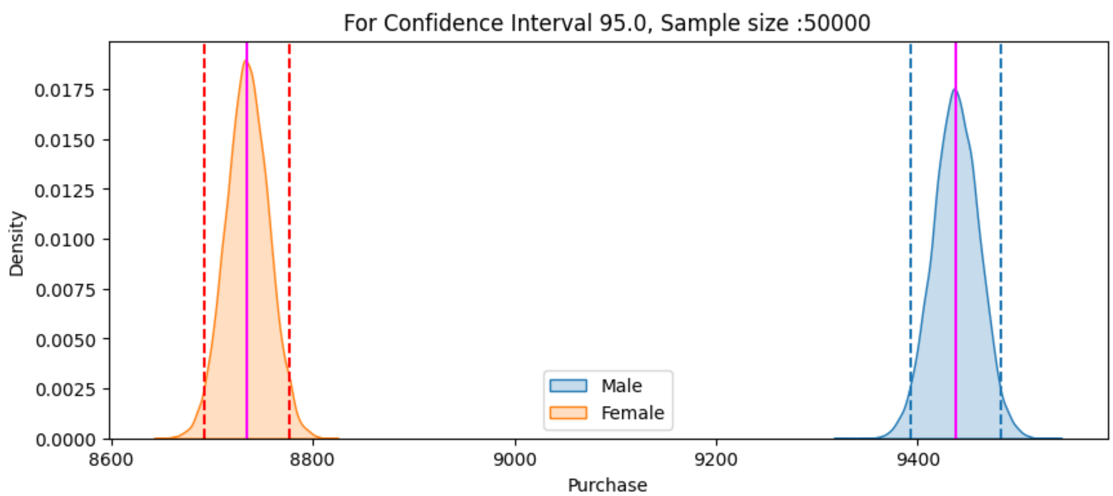
0.4.1 Confidence Interval of 95 %

[61]:
```
sample_sizes = [100,1000,10000,50000]
ci=95
n_size= 20000


for i in sample_sizes:
  mean_1,mean_2,lower_1,upper_1,lower_2,upper_2 =␣
 ↪gen_plot(df_male['Purchase'],df_female['Purchase'],i,n_size,ci)
  df_result = pd.concat([df_result,pd.DataFrame({'Gender':'M','Sample Size':
 ↪i,'LowerLimit':lower_1,'Upper Limit':upper_1,'Sample Mean':mean_1,
                                                 'Interval Range':
 ↪[(lower_1,upper_1)],'Confidence Interval':ci})],ignore_index = True)
  df_result = pd.concat([df_result,pd.DataFrame({'Gender':'F','Sample Size':
 ↪i,'LowerLimit':lower_2,'Upper Limit':upper_2,'Sample Mean':mean_2,
                                                  'Interval Range':
 ↪[(lower_2,upper_2)],'Confidence Interval':ci})],ignore_index = True)
```


For Confidence Interval 95.0, Sample size :100
Male
Female
Density
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0.0008
7000
8000
9000
10000
11000
12000
Purchase


For Confidence Interval 95.0, Sample size :1000
Male
Female
Density
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
8000
8500
9000
9500
10000
Purchase

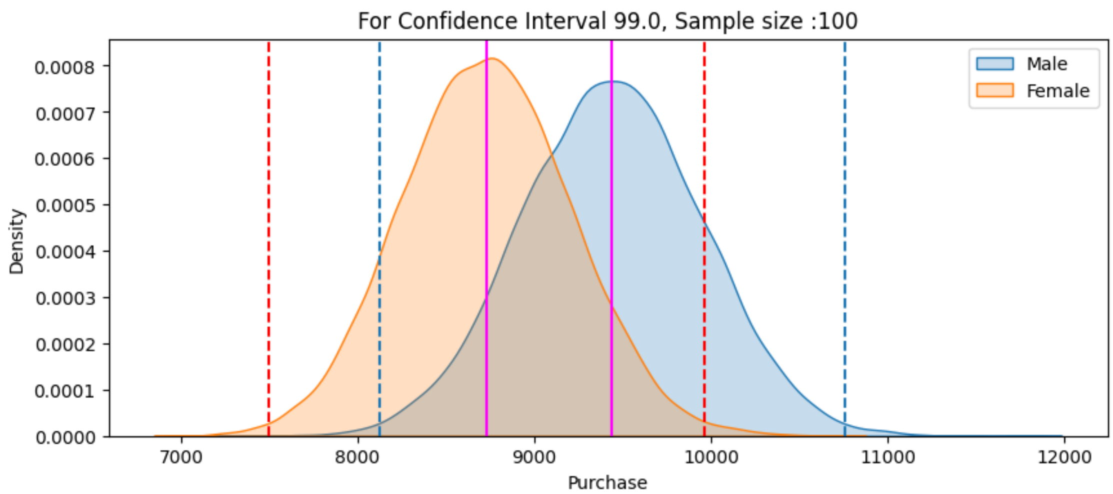


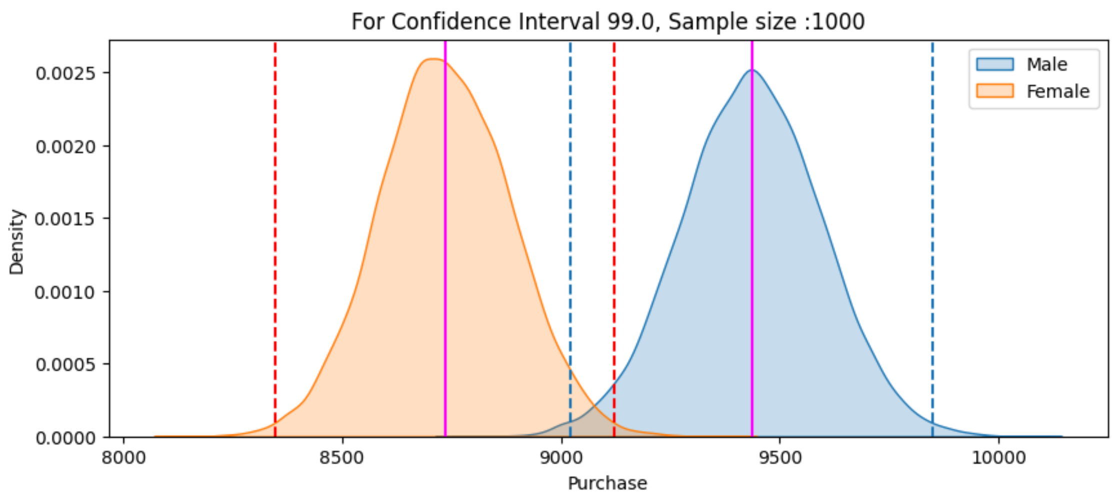


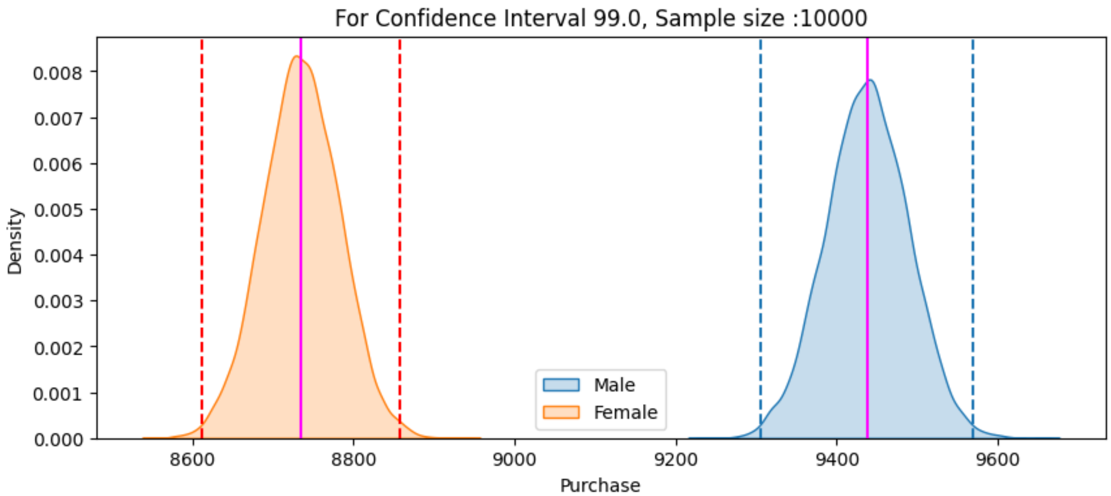
0.4.2 confidence Interval 99 %

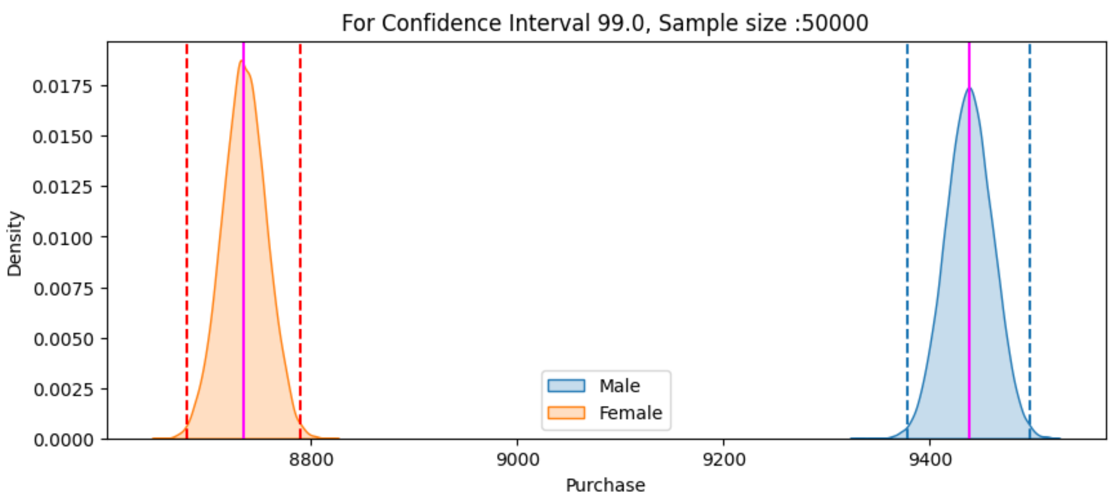
[62]:
```
sample_sizes = [100,1000,10000,50000]
ci=99
n_size= 20000


for i in sample_sizes:
  mean_1,mean_2,lower_1,upper_1,lower_2,upper_2 =␣
 ↪gen_plot(df_male['Purchase'],df_female['Purchase'],i,n_size,ci)
  df_result = pd.concat([df_result,pd.DataFrame({'Gender':'M','Sample Size':
 ↪i,'LowerLimit':lower_1,'Upper Limit':upper_1,'Sample Mean':mean_1,
```

```
                                                'Interval Range':
↪[(lower_1,upper_1)],'Confidence Interval':ci})],ignore_index = True)
 df_result = pd.concat([df_result,pd.DataFrame({'Gender':'F','Sample Size':
↪i,'LowerLimit':lower_2,'Upper Limit':upper_2,'Sample Mean':mean_2,
                                                 'Interval Range':
↪[(lower_2,upper_2)],'Confidence Interval':ci})],ignore_index = True)
```









[63]:

```
df_result
```

[63]:

	Gender	Sample Size	LowerLimit	Upper Limit	Sample Mean	\
0	M	100	8590.69	10274.99	9432.84	
1	F	100	7952.85	9525.88	8739.37	
2	M	1000	9171.20	9704.45	9437.83	
3	F	1000	8485.95	8981.57	8733.76	
4	M	10000	9353.65	9521.86	9437.76	
5	F	10000	8655.78	8812.32	8734.05	
6	M	50000	9400.12	9474.90	9437.51	
7	F	50000	8699.59	8769.69	8734.64	
8	M	100	8439.59	10433.79	9436.69	
9	F	100	7800.83	9666.53	8733.68	

10	M	1000	9120.89	9754.16	9437.52
11	F	1000	8436.82	9032.45	8734.63
12	M	10000	9338.44	9536.98	9437.71
13	F	10000	8641.17	8827.18	8734.18
14	M	50000	9392.60	9482.28	9437.44
15	F	50000	8693.07	8776.53	8734.80
16	M	100	8121.41	10757.07	9439.24
17	F	100	7494.82	9964.56	8729.69
18	M	1000	9022.73	9848.35	9435.54
19	F	1000	8348.02	9121.04	8734.53
20	M	10000	9306.23	9569.23	9437.73
21	F	10000	8612.05	8857.60	8734.82
22	M	50000	9378.74	9496.55	9437.65
23	F	50000	8680.00	8789.89	8734.95

	Interval Range	Confidence Interval
0	(8590.69, 10274.99)	90
1	(7952.85, 9525.88)	90
2	(9171.2, 9704.45)	90
3	(8485.95, 8981.57)	90
4	(9353.65, 9521.86)	90
5	(8655.78, 8812.32)	90
6	(9400.12, 9474.9)	90
7	(8699.59, 8769.69)	90
8	(8439.59, 10433.79)	95
9	(7800.83, 9666.53)	95
10	(9120.89, 9754.16)	95
11	(8436.82, 9032.45)	95
12	(9338.44, 9536.98)	95
13	(8641.17, 8827.18)	95
14	(9392.6, 9482.28)	95
15	(8693.07, 8776.53)	95
16	(8121.41, 10757.07)	99
17	(7494.82, 9964.56)	99
18	(9022.73, 9848.35)	99
19	(8348.02, 9121.04)	99
20	(9306.23, 9569.23)	99
21	(8612.05, 8857.6)	99
22	(9378.74, 9496.55)	99
23	(8680.0, 8789.89)	99

When Confidence Interval(CI) is `90`: - For sample size `100 for Males` the CI range is [8590.69, 10274.99] - For sample size `100 for Females` he CI range is [7952.85, 9525.88] - For sample size `50000 for Males` the CI range is [9400.12, 9474.9] - For sample size `50000 for Females` he CI range is [8699.59, 8769.69]

When Confidence Interval(CI) is `95`: - For sample size `100 for Males` the CI range is [8439.59, 10433.79] - For sample size `100 for Females` he CI range is [7800.83, 9666.53] - For sample size

50000 for Males the CI range is [9392.6, 9482.28] - For sample size 50000 for Females he CI range is [8693.07, 8776.53]

When Confidence Interval(CI) is 99: - For sample size 100 for Males the CI range is [8121.41, 10757.07] - For sample size 100 for Females he CI range is [7494.82, 9964.56] - For sample size 50000 for Males the CI range is [9378.74, 9496.55] - For sample size 50000 for Females he CI range is [8680.0, 8789.89]

- The analysis emphasizes how crucial sample size is for determining population parameters.
- It suggests that as the `sample size increases, the confidence intervals become narrower` and more precise.
- When at 90% confidence the average value for males falls approximately between `$ 9400.12and  $ 9474.9`. And for female it is `$8699.59and  $ 8769.69`.
- When at 95% confidence the average value for males falls approximately between `$ 9392.6 and $ 9482.28`. And for female it is `$8693.07 and $8776.53`.
- By this can say that Males spend more money than females.

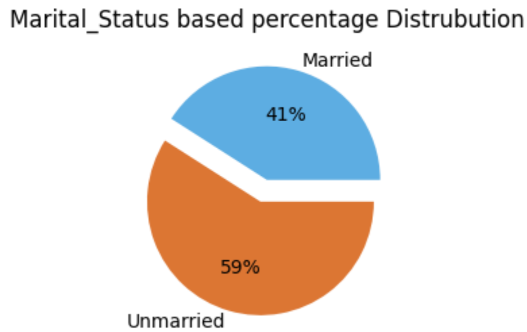
0.5 Marital Status *Vs* Purchase Amount

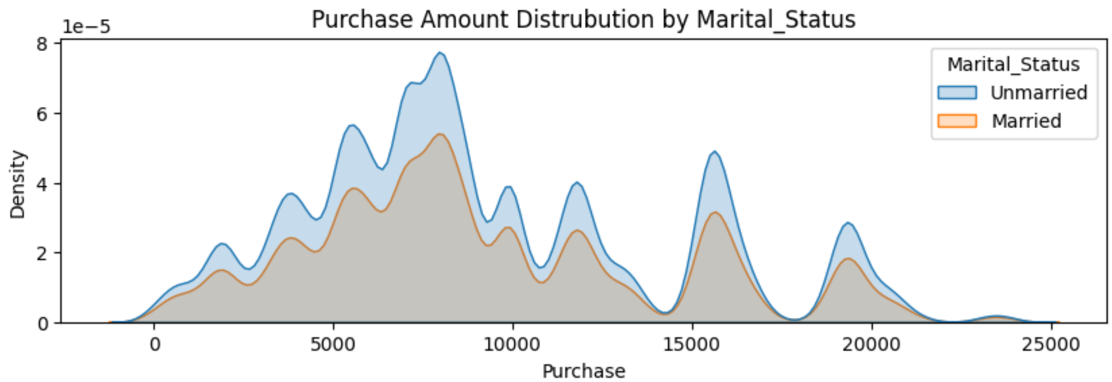
[64]:
```
temp= pd.DataFrame(df.groupby('Marital_Status')['Purchase'].
    ↪agg(['sum','mean','count']).reset_index())
temp['Precentage_distrubution']= np.round((temp['sum']/temp['sum'].sum())*100,2)
temp
```

[64]:

	Marital_Status	sum	mean	count	Precentage_distrubution
0	Married	2086885295	9261.174574	225337	40.95
1	Unmarried	3008927447	9265.907619	324731	59.05

[65]:
```
plt.figure(figsize=(10,6))
plt.subplot(2,1,1)
colors= ['#5DADE2','#DC7633']
plt.pie(temp['Precentage_distrubution'],labels= temp['Marital_Status'],autopct␣
    ↪= '%0.0f%%',explode=[0,0.2],colors=colors)
plt.title('Marital_Status based percentage Distrubution')
plt.subplot(2,1,2)
sns.kdeplot(x=df['Purchase'],hue=df['Marital_Status'],fill=True)
plt.title('Purchase Amount Distrubution by Marital_Status')
plt.show()
```





- The number of transcations made by Unmarried customers is more than the Married customers.
- but the average amount spent by both Unmarried and married customers are almost similar.

0.6 Construction of confidence Interval for Married and Unmarried Purchases: CLT

- From the plot we can see that the distrubution of purchase amount for Unmarried and Married on black friday is not Normal.
- so we use `Central Limit Therom`.
- It states the distribution of sample means will approximate a normal distribution, regardless of the underlying population distribution

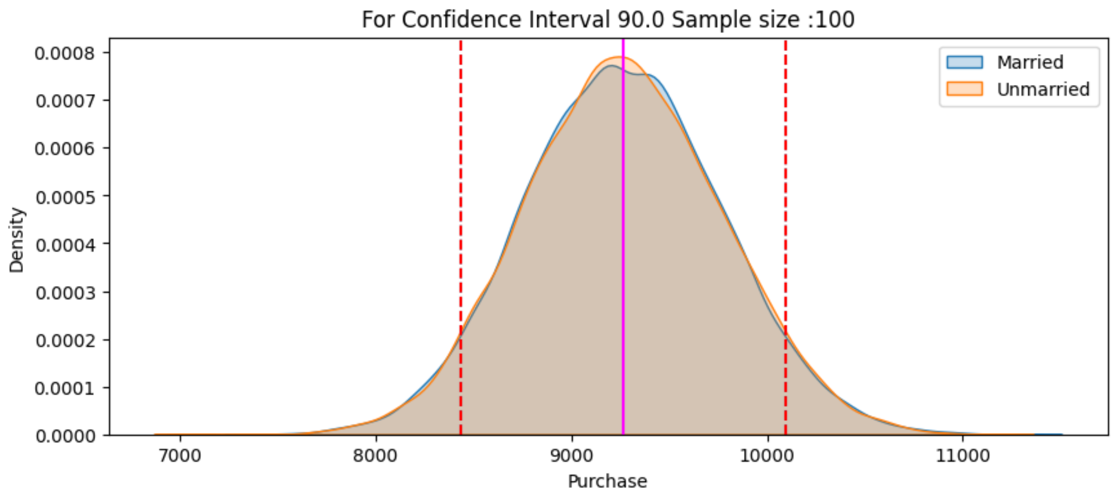
[66]:
```
from scipy.stats import norm
def gen_plot(sample1,sample2,sample_size,n_size,ci):
  plt.figure(figsize=(10,4))
  ci=ci/100
  sample1_means=[]
  sample2_means=[]
  for i in range(n_size):
    sample1_means.append(np.mean(sample1.sample(sample_size,replace=True)))
    sample2_means.append(np.mean(sample2.sample(sample_size,replace=True)))
  #for sample 1
  mean_1 = np.mean(sample1_means)
```

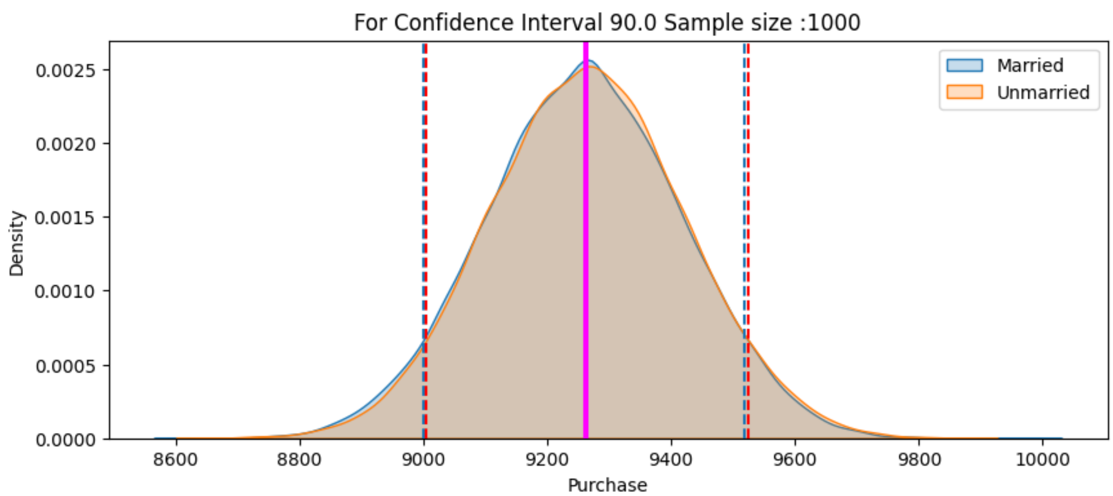
```
    std_1 = np.std(sample1_means)
    s_error_1 = std_1/ np.sqrt(len(sample1_means))

    lower_1 = norm.ppf((1-ci)/2)* std_1 + mean_1
    upper_1 = norm.ppf(1-(1-ci)/2)* std_1 + mean_1
    #for sample 2
    mean_2 = np.mean(sample2_means)
    std_2 = np.std(sample2_means)
    s_error_2 = std_2/ np.sqrt(len(sample2_means))

    lower_2 = norm.ppf((1-ci)/2)* std_2 + mean_2
    upper_2 = norm.ppf(1-(1-ci)/2)* std_2 + mean_2

    sns.kdeplot(data=sample1_means,fill=True,label='Married')
    plt.axvline(mean_1,color='#FF00FF')
    plt.axvline(lower_1,linestyle='--')
    plt.axvline(upper_1,linestyle='--')

    sns.kdeplot(data=sample2_means,fill=True,label='Unmarried')
    plt.axvline(mean_2,color='#FF00FF')
    plt.axvline(lower_2,linestyle='--',color = 'red')
    plt.axvline(upper_2,linestyle='--',color = 'red')

    plt.title(f'For Confidence Interval {ci*100} Sample size :{sample_size}')
    plt.legend()
    plt.xlabel('Purchase')
    plt.ylabel('Density')

    return round(mean_1,2), round(mean_2,2), round(lower_1,2),round(upper_1,2),␣
  ↪round(lower_2,2), round(upper_2,2)
```

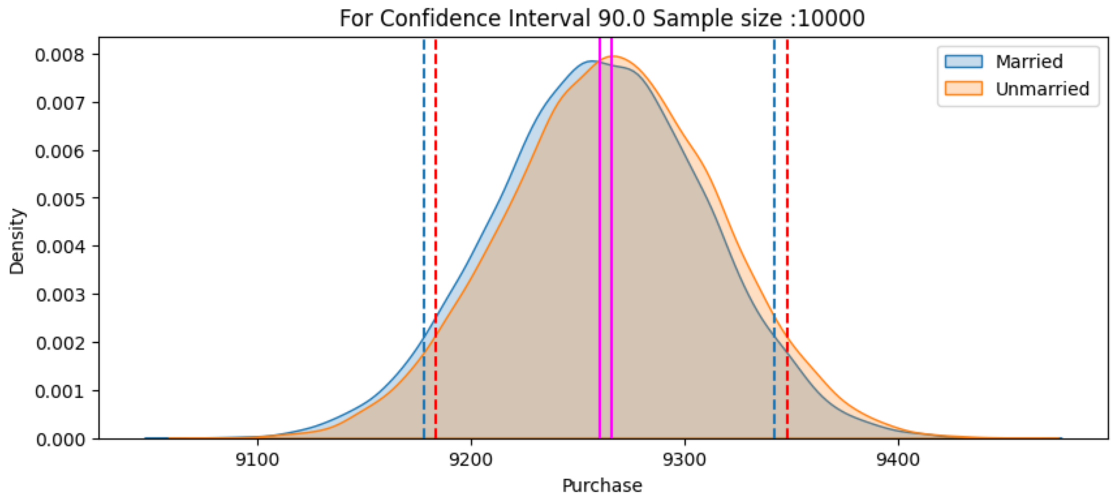
[72]:

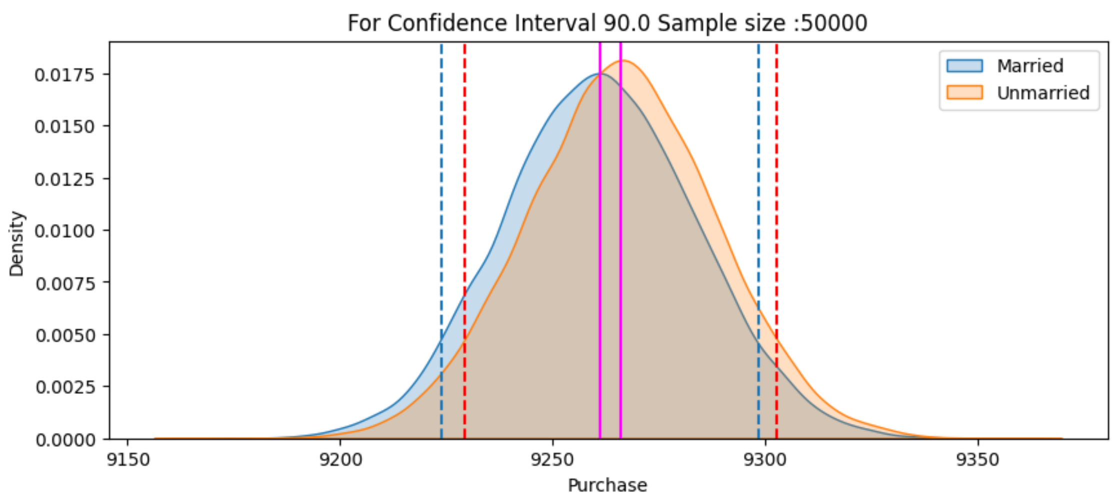
```
sample_sizes = [100,1000,10000,50000]
ci=90
n_size= 20000

df_married = df[df['Marital_Status'] == 'Married']
df_unmarried = df[df['Marital_Status'] == 'Unmarried']

df_result_2 = pd.DataFrame(columns = ['Marital_Status','Sample␣
  ↪Size','LowerLimit','Upper Limit','Sample Mean','Confidence Interval'])


for i in sample_sizes:
    mean_1,mean_2,lower_1,upper_1,lower_2,upper_2 =␣
  ↪gen_plot(df_married['Purchase'],df_unmarried['Purchase'],i,n_size,ci)
```

```python
df_result_2 = pd.concat([df_result_2,pd.DataFrame({'Marital_Status':
↪'Married','Sample Size':i,'LowerLimit':lower_1,'Upper Limit':upper_1,'Sample␣
↪Mean':mean_1,
                              'Interval Range':
↪[(lower_1,upper_1)],'Confidence Interval':ci})],ignore_index = True)
df_result_2 = pd.concat([df_result_2,pd.DataFrame({'Marital_Status':
↪'Unmarried','Sample Size':i,'LowerLimit':lower_2,'Upper Limit':
↪upper_2,'Sample Mean':mean_2,
                              'Interval Range':
↪[(lower_2,upper_2)],'Confidence Interval':ci})],ignore_index = True)
```





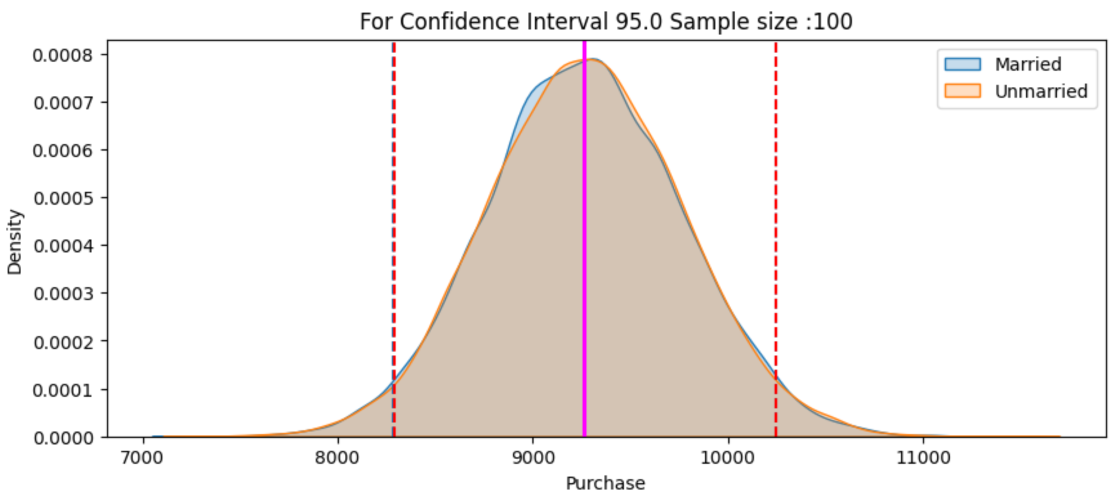


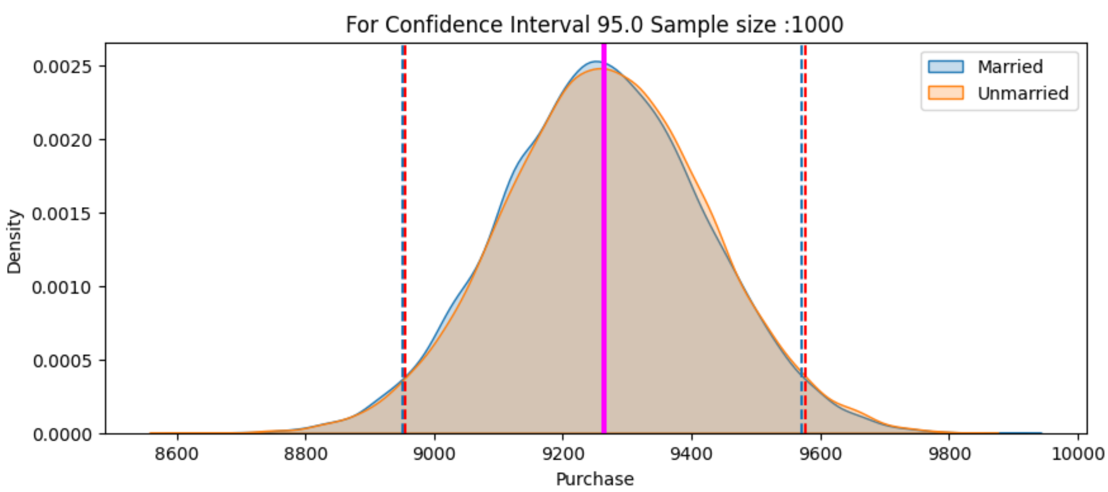


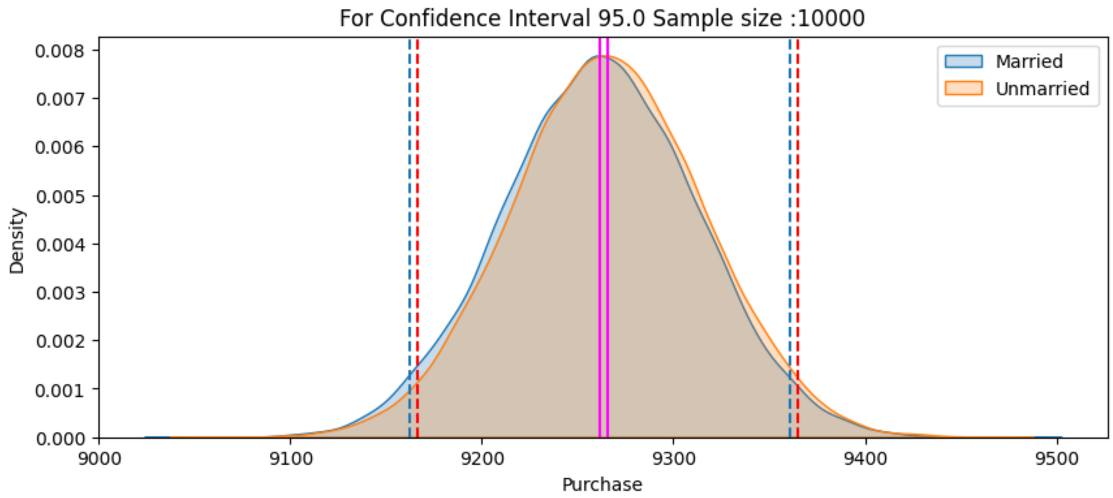
0.6.1 confidence Interval of 95 %

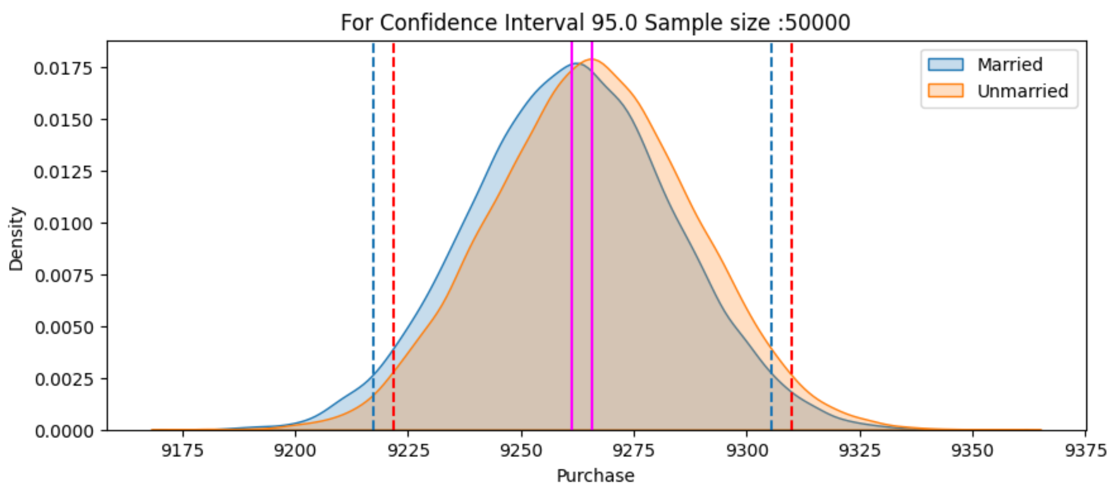
[74]:
```
sample_sizes = [100,1000,10000,50000]
ci=95
n_size= 20000

for i in sample_sizes:
  mean_1,mean_2,lower_1,upper_1,lower_2,upper_2 =␣
 ↪gen_plot(df_married['Purchase'],df_unmarried['Purchase'],i,n_size,ci)
  df_result_2 = pd.concat([df_result_2,pd.DataFrame({'Marital_Status':
 ↪'Married','Sample Size':i,'LowerLimit':lower_1,'Upper Limit':upper_1,'Sample␣
 ↪Mean':mean_1,
```

```
                                'Interval Range':
↪[(lower_1,upper_1)],'Confidence Interval':ci})],ignore_index = True)
 df_result_2 = pd.concat([df_result_2,pd.DataFrame({'Marital_Status':
↪'Unmarried','Sample Size':i,'LowerLimit':lower_2,'Upper Limit':
↪upper_2,'Sample Mean':mean_2,
                                'Interval Range':
↪[(lower_2,upper_2)],'Confidence Interval':ci})],ignore_index = True)
```





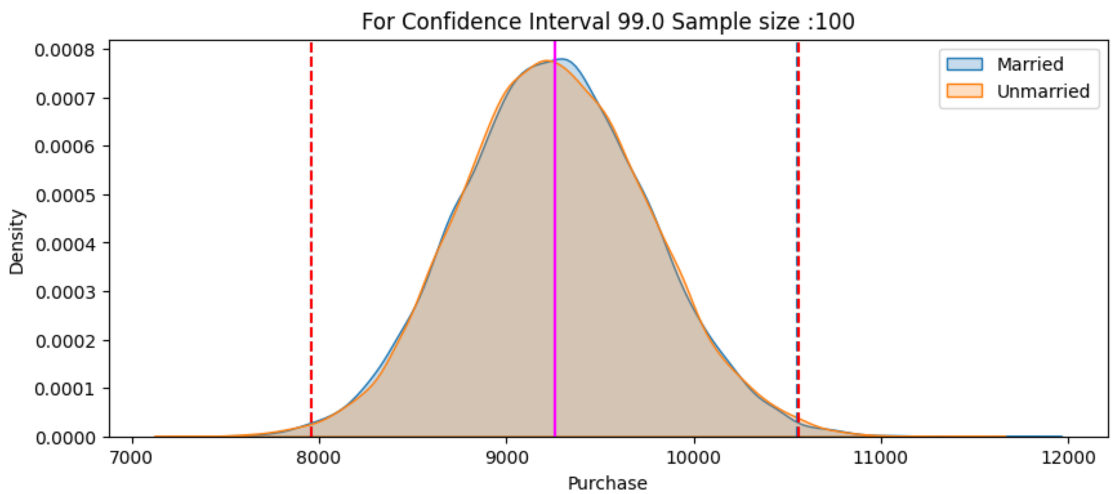


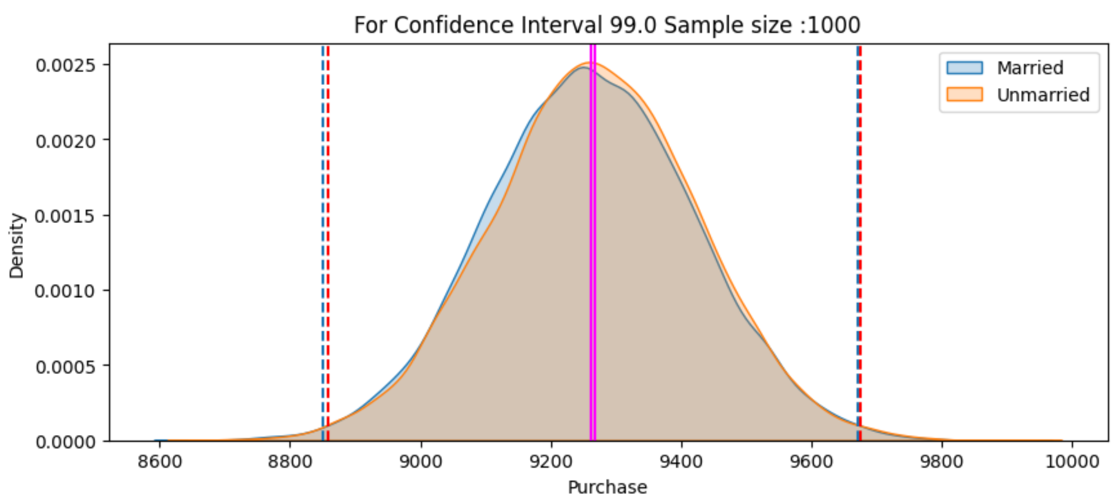


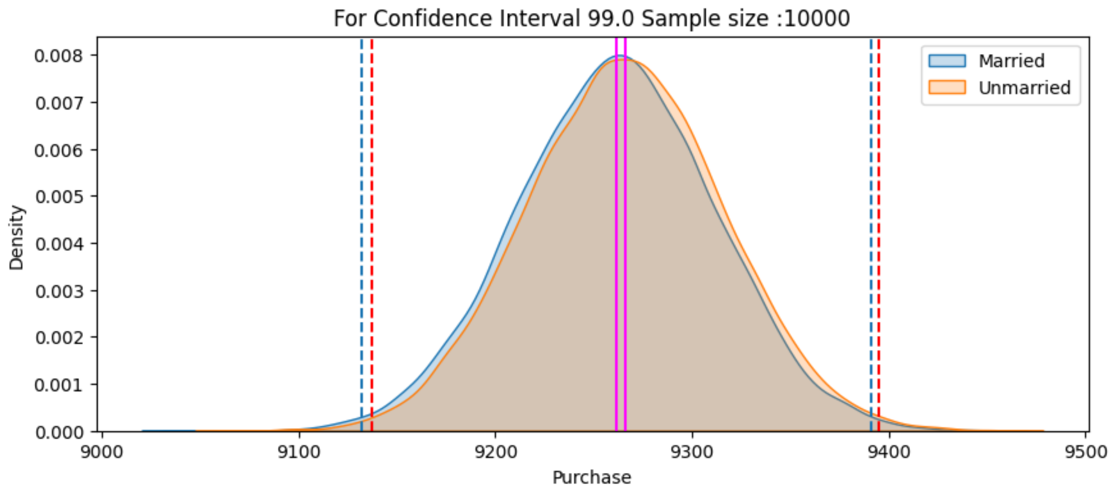
0.6.2 confidence Interval 99 %

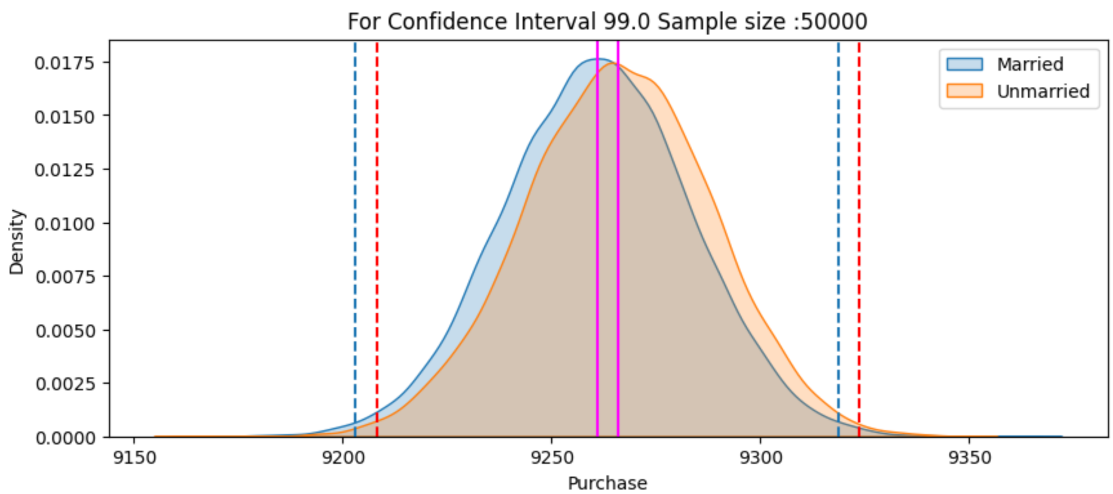
[75]:
```
sample_sizes = [100,1000,10000,50000]
ci=99
n_size= 20000


for i in sample_sizes:
    mean_1,mean_2,lower_1,upper_1,lower_2,upper_2 =␣
  ↪gen_plot(df_married['Purchase'],df_unmarried['Purchase'],i,n_size,ci)
    df_result_2 = pd.concat([df_result_2,pd.DataFrame({'Marital_Status':
  ↪'Married','Sample Size':i,'LowerLimit':lower_1,'Upper Limit':upper_1,'Sample␣
  ↪Mean':mean_1,
```

```
                                    'Interval Range':
↪[(lower_1,upper_1)],'Confidence Interval':ci})],ignore_index = True)
 df_result_2 = pd.concat([df_result_2,pd.DataFrame({'Marital_Status':
↪'Unmarried','Sample Size':i,'LowerLimit':lower_2,'Upper Limit':
↪upper_2,'Sample Mean':mean_2,
                                    'Interval Range':
↪[(lower_2,upper_2)],'Confidence Interval':ci})],ignore_index = True)
```









[76]:
```
df_result_2
```

[76]:

	Marital_Status	Sample Size	LowerLimit	Upper Limit	Sample Mean	\
0	Married	100	8433.73	10091.52	9262.63	
1	Unmarried	100	8436.80	10091.61	9264.20	
2	Married	1000	8999.64	9519.23	9259.44	
3	Unmarried	1000	9004.97	9524.04	9264.51	
4	Married	10000	9178.29	9342.15	9260.22	
5	Unmarried	10000	9183.85	9348.35	9266.10	
6	Married	50000	9223.88	9298.39	9261.13	
7	Unmarried	50000	9229.31	9302.90	9266.11	
8	Married	100	8265.13	10250.88	9258.01	
9	Unmarried	100	8275.64	10252.25	9263.94	

10	Married	1000	8949.82	9572.64	9261.23
11	Unmarried	1000	8956.09	9577.45	9266.77
12	Married	10000	9161.58	9359.89	9260.73
13	Unmarried	10000	9166.65	9364.05	9265.35
14	Married	100	8285.48	10245.27	9265.37
15	Unmarried	100	8288.15	10249.32	9268.74
16	Married	1000	8951.88	9571.68	9261.78
17	Unmarried	1000	8955.85	9577.38	9266.62
18	Married	10000	9162.10	9361.24	9261.67
19	Unmarried	10000	9166.69	9364.71	9265.70
20	Married	50000	9217.27	9305.42	9261.34
21	Unmarried	50000	9221.78	9309.86	9265.82
22	Married	100	7958.06	10555.62	9256.84
23	Unmarried	100	7960.49	10561.79	9261.14
24	Married	1000	8851.49	9671.05	9261.27
25	Unmarried	1000	8859.36	9675.02	9267.19
26	Married	10000	9131.89	9390.79	9261.34
27	Unmarried	10000	9137.03	9395.04	9266.04
28	Married	50000	9203.06	9318.73	9260.89
29	Unmarried	50000	9208.09	9323.66	9265.88

	Confidence Interval	Interval Range
0	90	(8433.73, 10091.52)
1	90	(8436.8, 10091.61)
2	90	(8999.64, 9519.23)
3	90	(9004.97, 9524.04)
4	90	(9178.29, 9342.15)
5	90	(9183.85, 9348.35)
6	90	(9223.88, 9298.39)
7	90	(9229.31, 9302.9)
8	95	(8265.13, 10250.88)
9	95	(8275.64, 10252.25)
10	95	(8949.82, 9572.64)
11	95	(8956.09, 9577.45)
12	95	(9161.58, 9359.89)
13	95	(9166.65, 9364.05)
14	95	(8285.48, 10245.27)
15	95	(8288.15, 10249.32)
16	95	(8951.88, 9571.68)
17	95	(8955.85, 9577.38)
18	95	(9162.1, 9361.24)
19	95	(9166.69, 9364.71)
20	95	(9217.27, 9305.42)
21	95	(9221.78, 9309.86)
22	99	(7958.06, 10555.62)
23	99	(7960.49, 10561.79)
24	99	(8851.49, 9671.05)

```
25                      99    (8859.36, 9675.02)
26                      99    (9131.89, 9390.79)
27                      99    (9137.03, 9395.04)
28                      99    (9203.06, 9318.73)
29                      99    (9208.09, 9323.66)
```

When Confidence Interval(CI) is `90`: - For sample size `100 for Married` the CI range is [8433.73, 10091.52] - For sample size `100 for Unmarried` he CI range is [8436.8, 10091.61] - For sample size `50000 for Married` the CI range is [9223.88, 9298.39] - For sample size `50000 for Unmarried` he CI range is [9229.31, 9302.9]

When Confidence Interval(CI) is `95`: - For sample size `100 for Married` the CI range is [8265.13, 10250.88] - For sample size `100 for Unmarried` he CI range is [8275.64, 10252.25] - For sample size `50000 for Married` the CI range is [9217.27, 9305.42] - For sample size `50000 for Unmarried` he CI range is [9221.78, 9309.86]

When Confidence Interval(CI) is `99`: - For sample size `100 for Married` the CI range is [7958.06, 10555.62] - For sample size `100 for Unmarried` he CI range is [7960.49, 10561.79] - For sample size `50000 for Married` the CI range is [9203.06, 9318.73] - For sample size `50000 for Unmarried` he CI range is [9208.09, 9323.66]

- The analysis emphasizes how crucial sample size is for determining population parameters for Marital_Status.
- It suggests that as the `sample size increases, the confidence intervals become narrower` and more precise.
- When at 95% confidence the average value for Married falls between `$ 9217.27 and $ 9305.42`. And for Unmarried it is `$9221.78 and $ 9309.86`.5.
- By this can say that Unmarried spend more money than Married customers.

0.7 Age Groups *Vs* Purchases

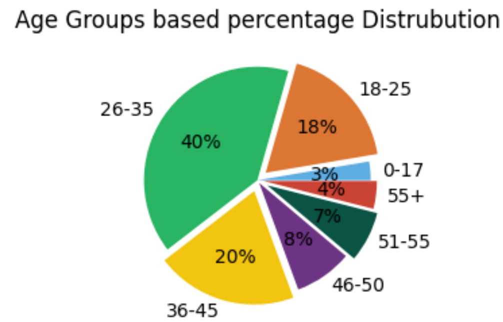
[77]:
```python
temp_1= pd.DataFrame(df.groupby('Age')['Purchase'].agg(['sum','mean','count']).
 ↪reset_index())
temp_1['Precentage_distrubution']= np.round((temp_1['sum']/temp_1['sum'].
 ↪sum())*100,2)
temp_1
```

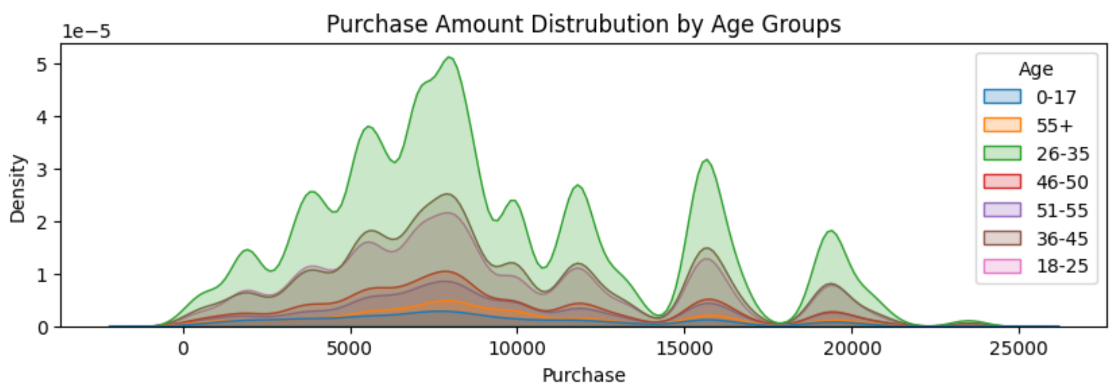
[77]:

	Age	sum	mean	count	Precentage_distrubution
0	0-17	134913183	8933.464640	15102	2.65
1	18-25	913848675	9169.663606	99660	17.93
2	26-35	2031770578	9252.690633	219587	39.87
3	36-45	1026569884	9331.350695	110013	20.15
4	46-50	420843403	9208.625697	45701	8.26
5	51-55	367099644	9534.808031	38501	7.20
6	55+	200767375	9336.280459	21504	3.94

[78]:
```python
plt.figure(figsize=(10,6))
plt.subplot(2,1,1)
```

```
colors= ['#5DADE2','#DC7633','#28B463','#F1C40F','#6C3483','#0B5345','#CB4335']
plt.pie(temp_1['Precentage_distrubution'],labels= temp_1['Age'],autopct = '%0.
 ↪0f%%',explode=[0.0,0.099,0.0,0.099,0.029,0.099,0.055],colors=colors)
plt.title('Age Groups based percentage Distrubution')
plt.subplot(2,1,2)
sns.kdeplot(x=df['Purchase'],hue=df['Age'],fill=True)
plt.title('Purchase Amount Distrubution by Age Groups')
plt.show()
```



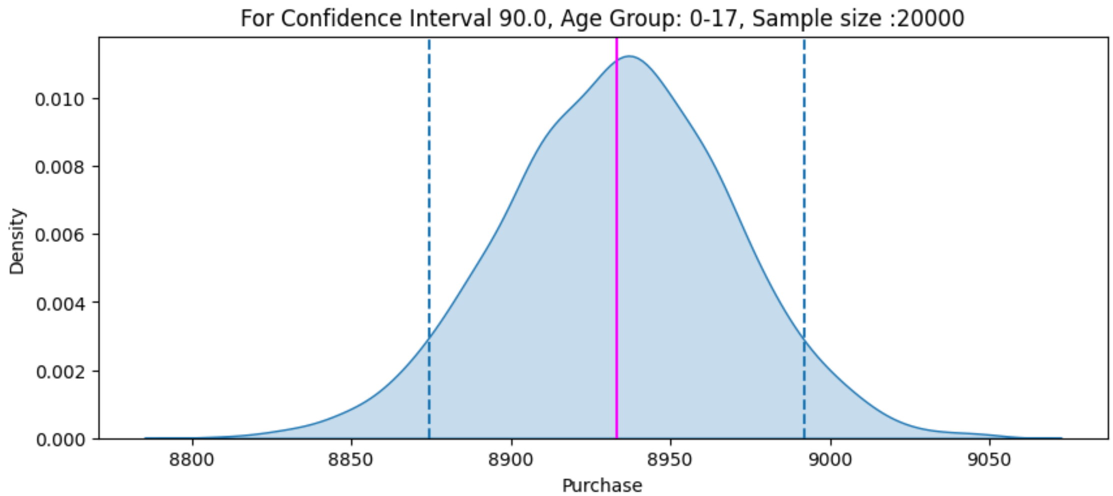


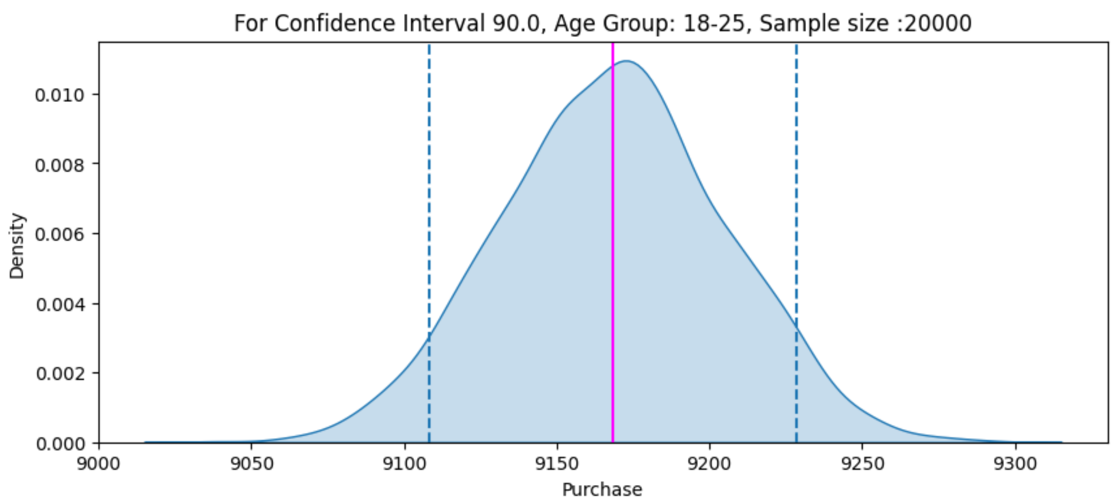
- the plot shows that balck friday sales are `more popular in the age groups 26-35` and `less popular in 0-17`.
- The number of transactions for `0-17` age group are less but there average purchase amount is 8933.

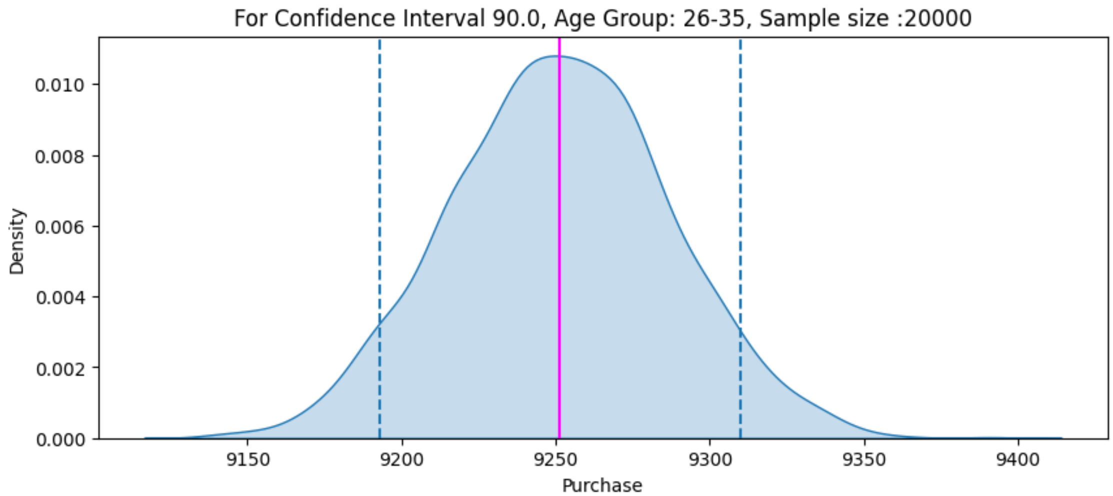
[79]:
```
from scipy.stats import norm
def gen_plot(sample,sample_size,n_size,ci):
  plt.figure(figsize=(10,4))
  ci=ci/100
  global flag
  sample1_means=[]
  for i in range(n_size):
    sample1_means.append(np.mean(sample.sample(sample_size,replace=True)))

  mean = np.mean(sample1_means)
```

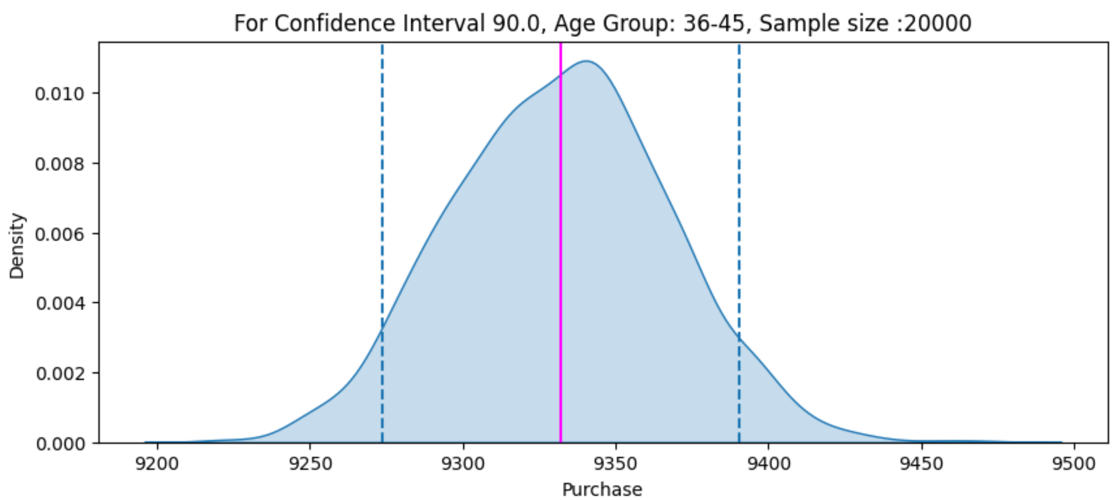
```
    std = np.std(sample1_means)
    s_error = std/ np.sqrt(len(sample1_means))

    lower = norm.ppf((1-ci)/2)* std + mean
    upper = norm.ppf(1-(1-ci)/2)* std + mean

    sns.kdeplot(data=sample1_means,fill=True)
    plt.axvline(mean,color='#FF00FF')
    plt.axvline(lower,linestyle='--')
    plt.axvline(upper,linestyle='--')


    plt.title(f'For Confidence Interval {ci*100}, Age Group: {age_group[flag]},␣
↪Sample size :{sample_size}')
    plt.xlabel('Purchase')
    plt.ylabel('Density')
    plt.show()
    flag+=1

    return round(mean,2), round(lower,2), round(upper,2)
```

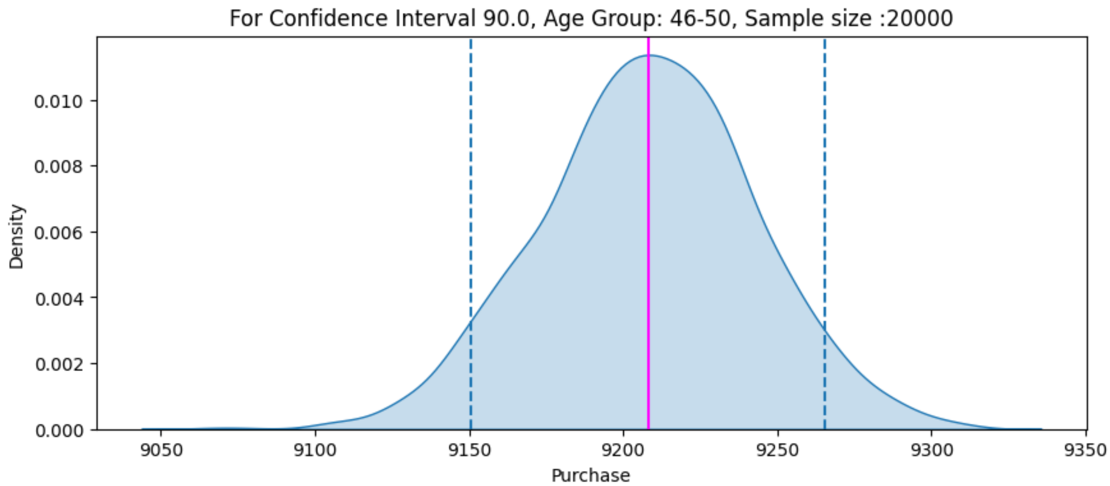
[80]:
```
sample_sizes = 20000
ci=90
n_size= 2000
flag=0
global age_group
age_group = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55','55+']

df_result_3 = pd.DataFrame(columns = ['Age_Group','Sample␣
↪Size','LowerLimit','Upper Limit','Sample Mean','Confidence␣
↪Interval','Interval Range'])

for i in age_group:
    mean,lower,upper =␣
↪gen_plot(df[df['Age']==i]['Purchase'],sample_sizes,n_size,ci)
    df_result_3= pd.concat([df_result_3,pd.DataFrame({'Age_Group':i,'Sample Size':
↪sample_sizes,'LowerLimit':lower,'Upper Limit':upper,'Sample Mean':
↪mean,'Confidence Interval':ci,
                                                                    'Interval Range':
↪[(lower,upper)]})], ignore_index =True)
```

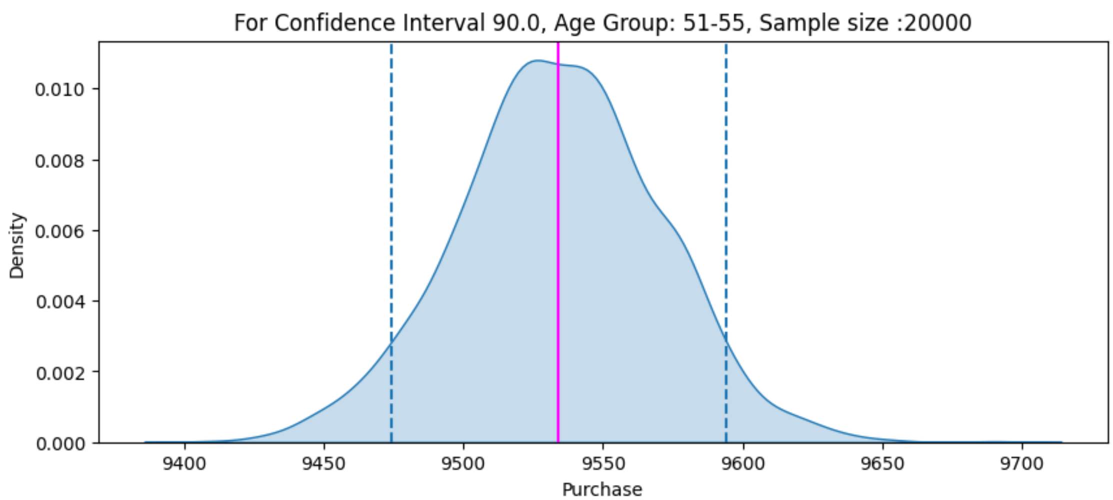

For Confidence Interval 90.0, Age Group: 0-17, Sample size :20000
Density
Purchase
0.000
0.002
0.004
0.006
0.008
0.010
8800
8850
8900
8950
9000
9050

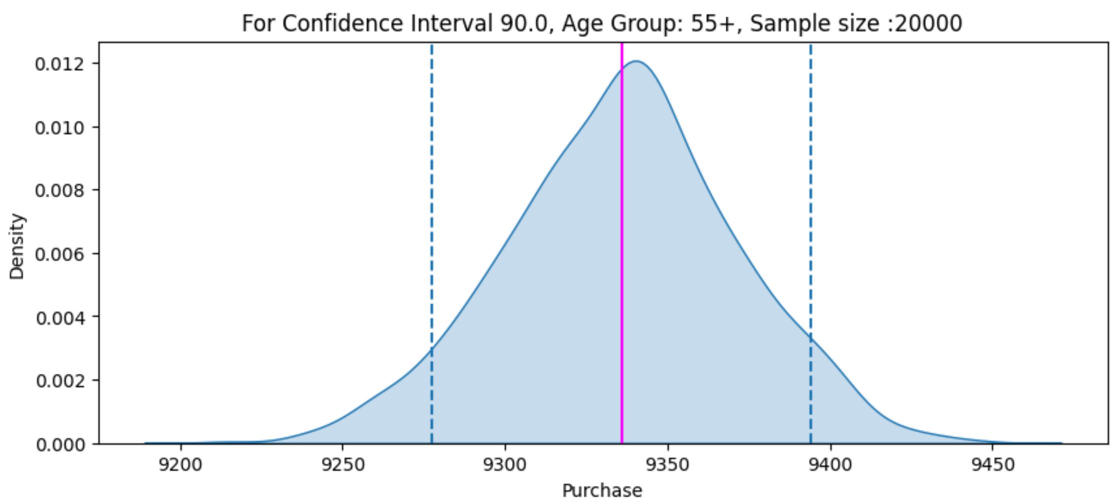

For Confidence Interval 90.0, Age Group: 18-25, Sample size :20000
Density
Purchase
0.000
0.002
0.004
0.006
0.008
0.010
9000
9050
9100
9150
9200
9250
9300


For Confidence Interval 90.0, Age Group: 26-35, Sample size :20000
Density
0.010
0.008
0.006
0.004
0.002
0.000
9150
9200
9250
9300
9350
9400
Purchase

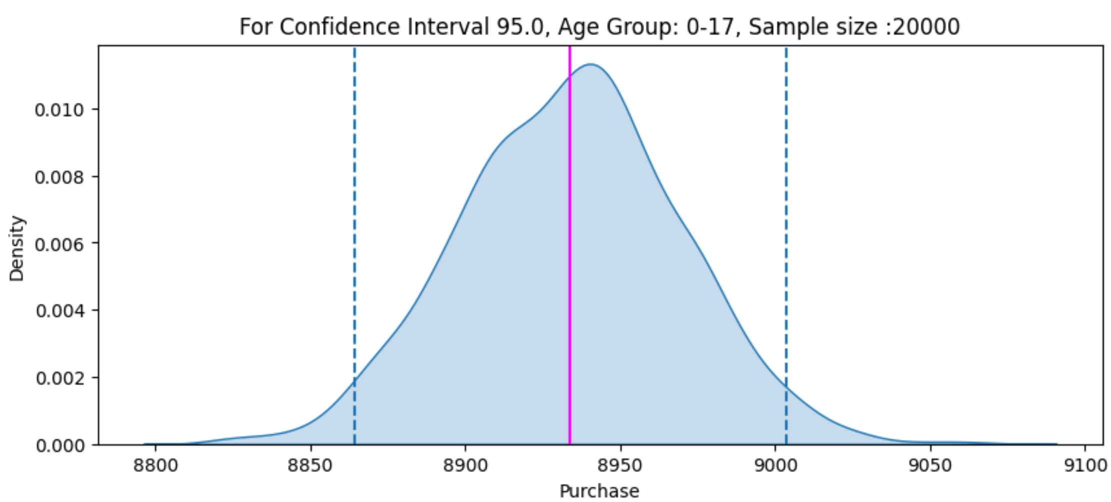

For Confidence Interval 90.0, Age Group: 36-45, Sample size :20000
Density
0.010
0.008
0.006
0.004
0.002
0.000
9200
9250
9300
9350
9400
9450
9500
Purchase

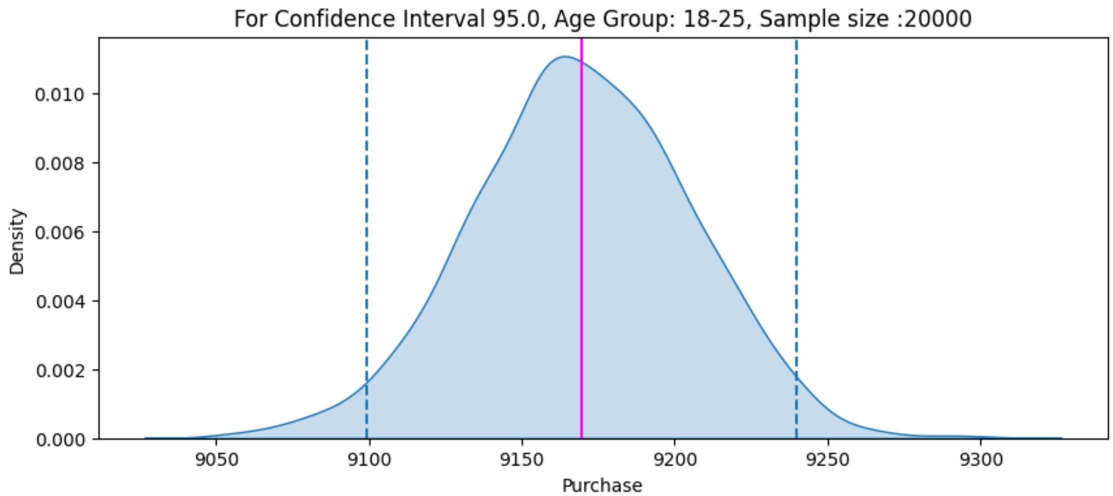

For Confidence Interval 90.0, Age Group: 46-50, Sample size :20000
Density
Purchase
0.000
0.002
0.004
0.006
0.008
0.010
9050
9100
9150
9200
9250
9300
9350

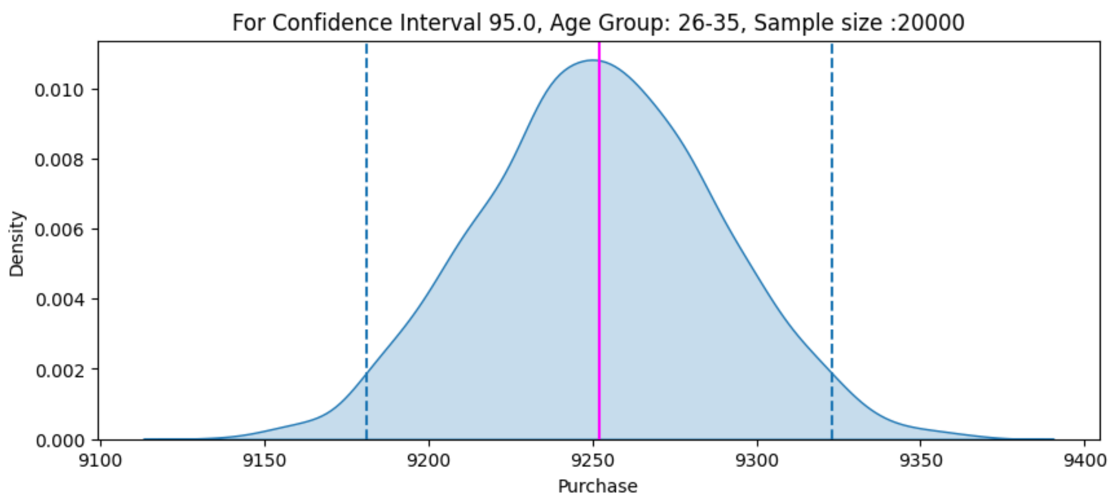

For Confidence Interval 90.0, Age Group: 51-55, Sample size :20000
Density
Purchase
0.000
0.002
0.004
0.006
0.008
0.010
9400
9450
9500
9550
9600
9650
9700

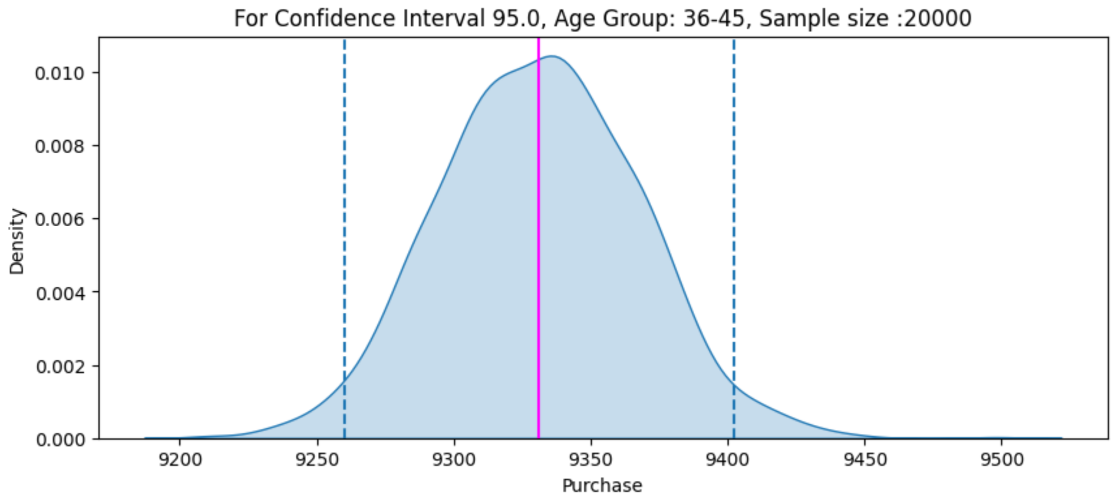


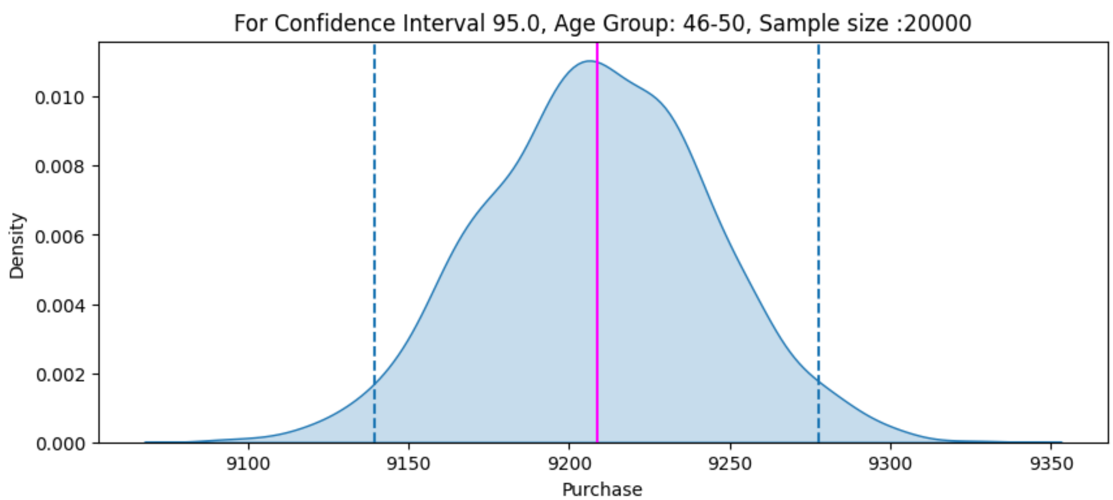
[81]:
```
sample_sizes = 20000
ci=95
n_size= 2000
flag=0


for i in age_group:
  mean,lower,upper =␣
 ↪gen_plot(df[df['Age']==i]['Purchase'],sample_sizes,n_size,ci)
  df_result_3= pd.concat([df_result_3,pd.DataFrame({'Age_Group':i,'Sample Size':
 ↪sample_sizes,'LowerLimit':lower,'Upper Limit':upper,'Sample Mean':
 ↪mean,'Confidence Interval':ci,
                                                'Interval Range':
 ↪[(lower,upper)]})], ignore_index =True)
```

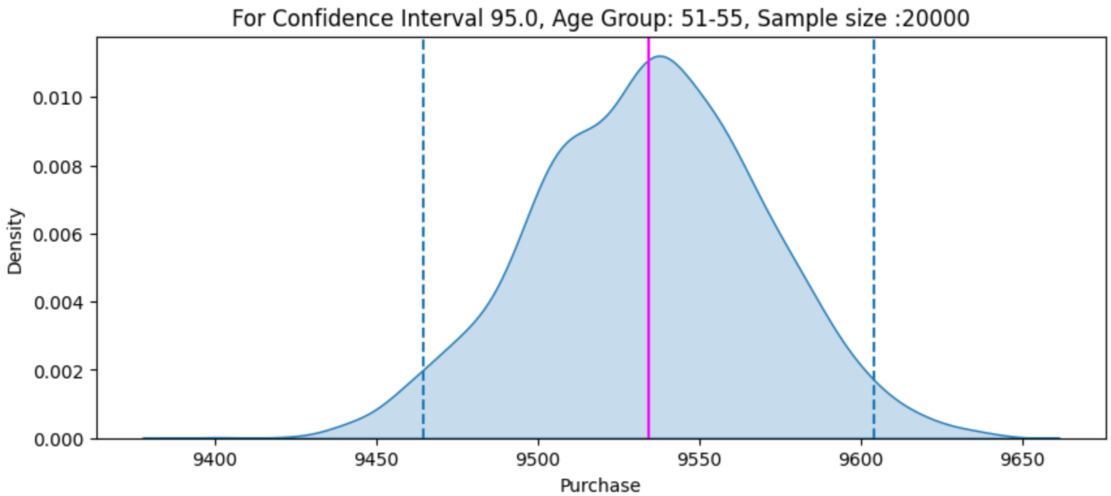


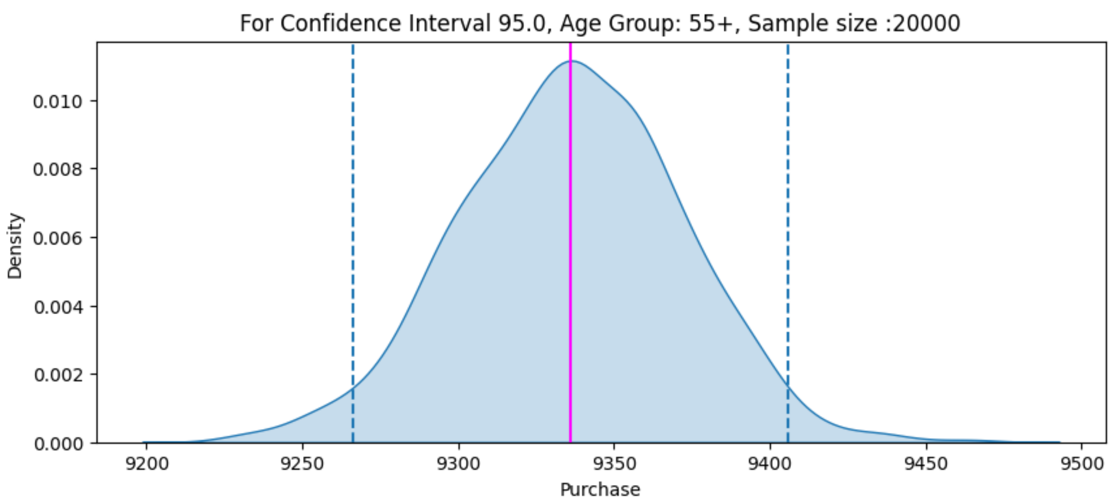

For Confidence Interval 95.0, Age Group: 18-25, Sample size :20000
Density
0.000
0.002
0.004
0.006
0.008
0.010
9050
9100
9150
9200
9250
9300
Purchase


For Confidence Interval 95.0, Age Group: 26-35, Sample size :20000
Density
0.000
0.002
0.004
0.006
0.008
0.010
9100
9150
9200
9250
9300
9350
9400
Purchase


For Confidence Interval 95.0, Age Group: 36-45, Sample size :20000
Density
0.000
0.002
0.004
0.006
0.008
0.010
9200
9250
9300
9350
9400
9450
9500
Purchase

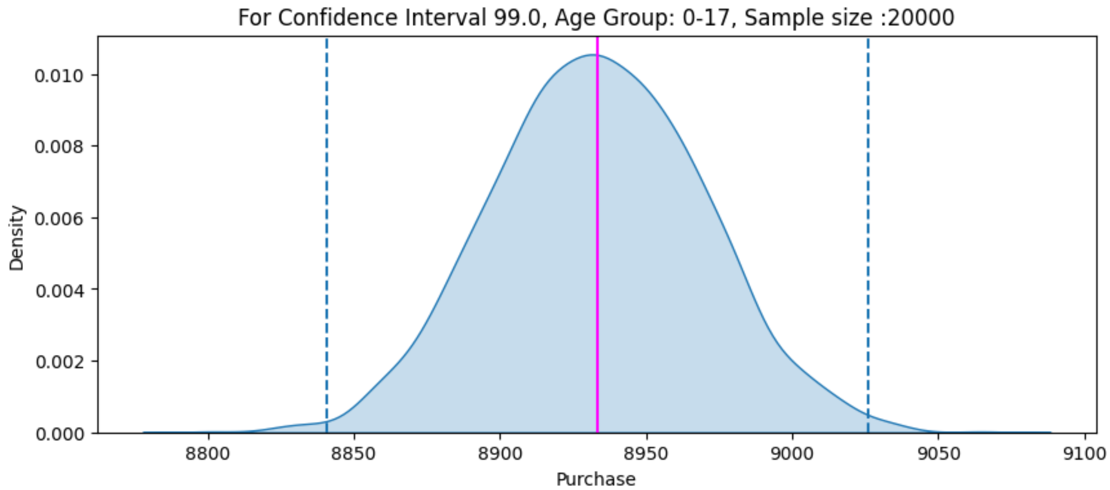

For Confidence Interval 95.0, Age Group: 46-50, Sample size :20000
Density
0.000
0.002
0.004
0.006
0.008
0.010
9100
9150
9200
9250
9300
9350
Purchase

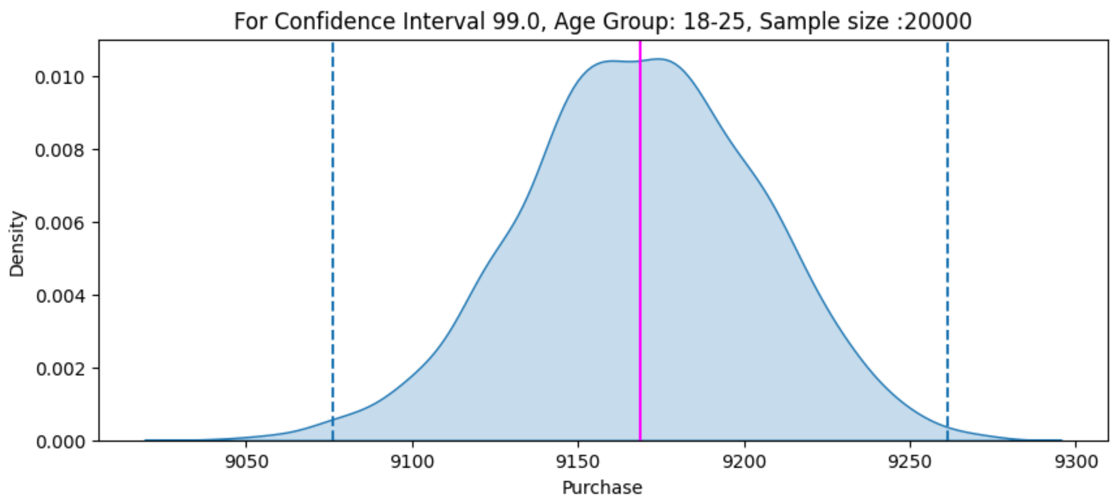


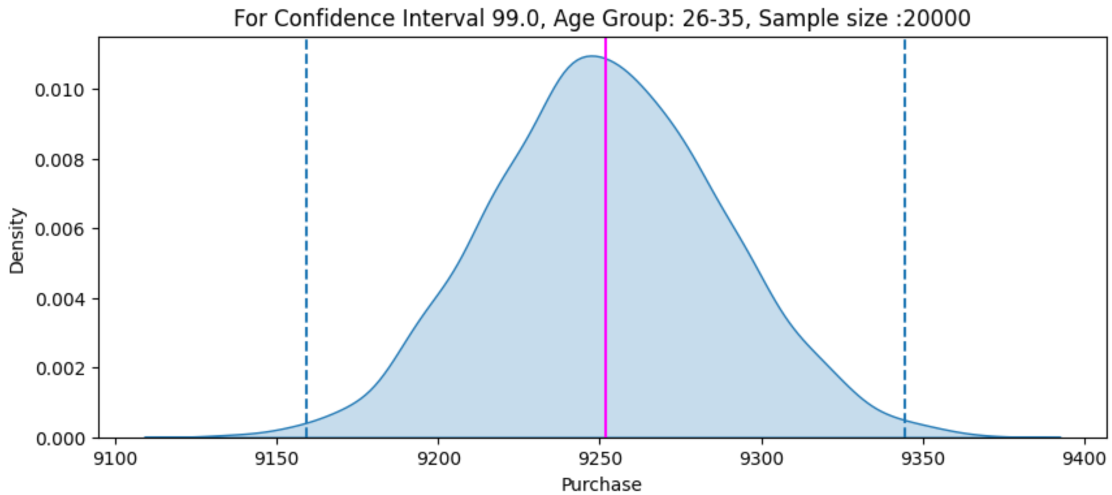


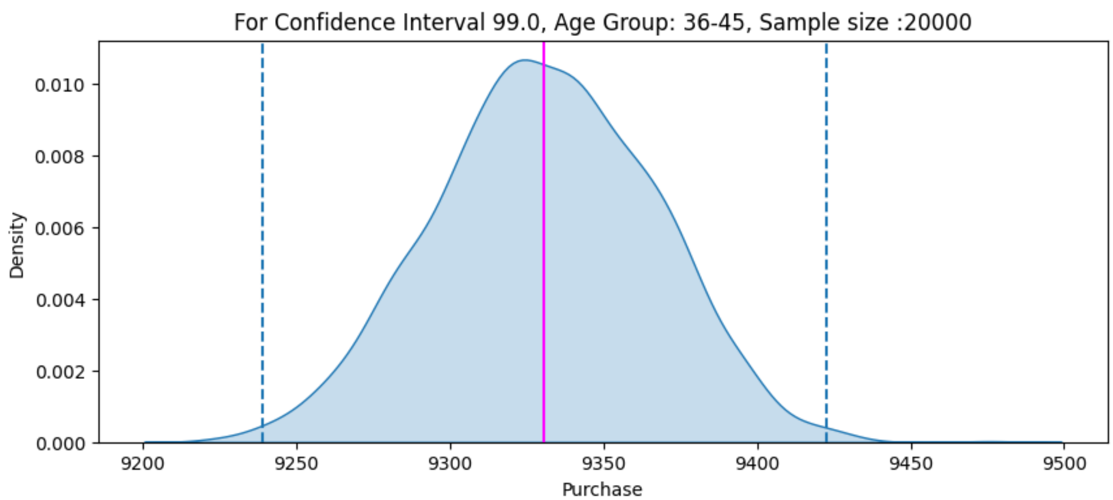
[82]:
```
sample_sizes = 20000
ci=99
n_size= 2000
flag=0


for i in age_group:
  mean,lower,upper =␣
↪gen_plot(df[df['Age']==i]['Purchase'],sample_sizes,n_size,ci)
  df_result_3= pd.concat([df_result_3,pd.DataFrame({'Age_Group':i,'Sample Size':
↪sample_sizes,'LowerLimit':lower,'Upper Limit':upper,'Sample Mean':
↪mean,'Confidence Interval':ci,
```

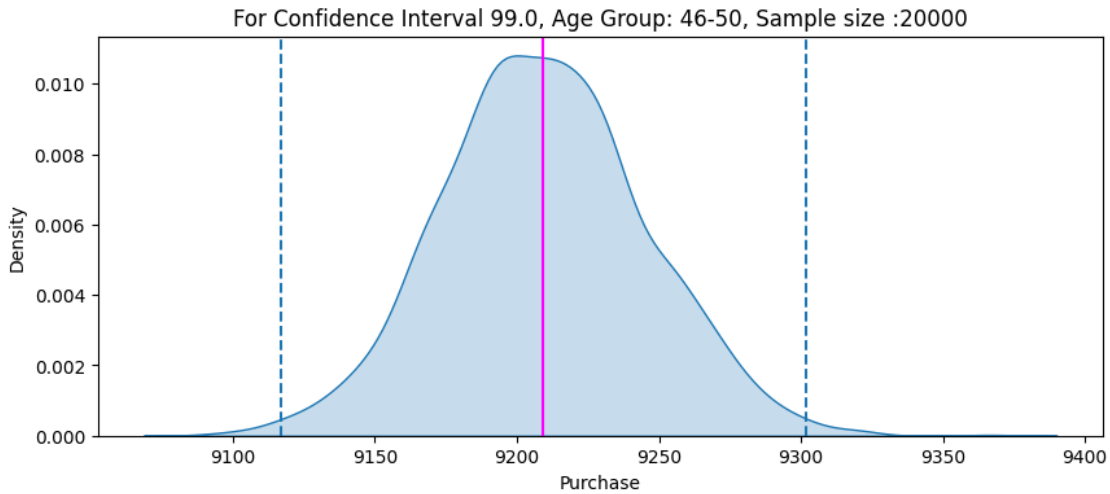
```
                                              'Interval Range':
↪[(lower,upper)]})], ignore_index =True)
```

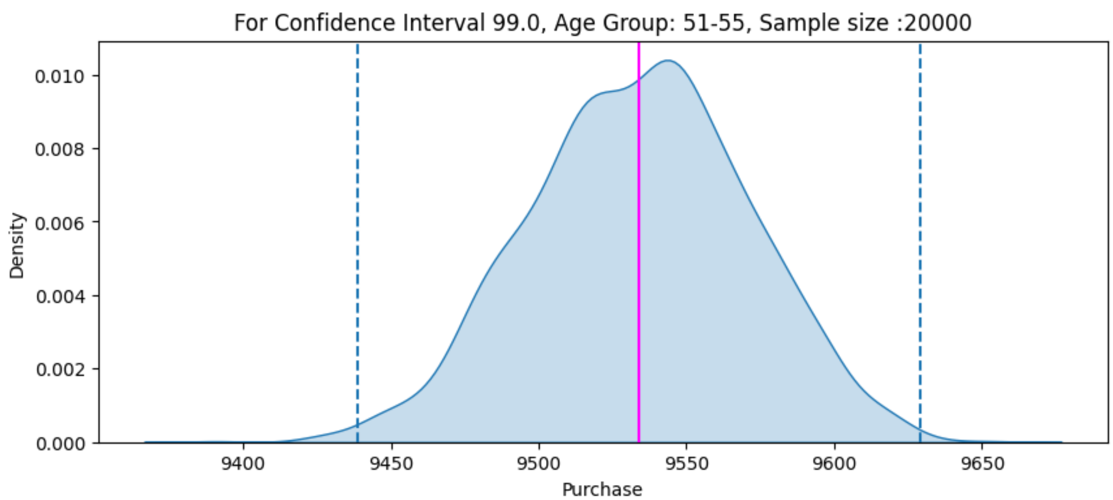


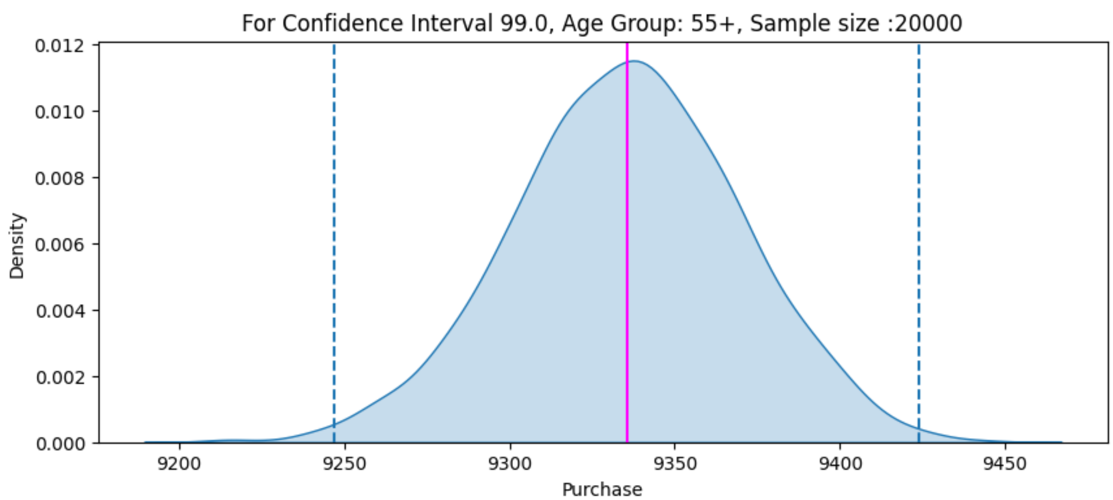



For Confidence Interval 99.0, Age Group: 26-35, Sample size :20000
Density
0.000
0.002
0.004
0.006
0.008
0.010
9100
9150
9200
9250
9300
9350
9400
Purchase


For Confidence Interval 99.0, Age Group: 36-45, Sample size :20000
Density
0.000
0.002
0.004
0.006
0.008
0.010
9200
9250
9300
9350
9400
9450
9500
Purchase


For Confidence Interval 99.0, Age Group: 46-50, Sample size :20000
Density
0.000
0.002
0.004
0.006
0.008
0.010
9100
9150
9200
9250
9300
9350
9400
Purchase


For Confidence Interval 99.0, Age Group: 51-55, Sample size :20000
Density
0.000
0.002
0.004
0.006
0.008
0.010
9400
9450
9500
9550
9600
9650
Purchase



[83]:
```
df_result_3
```

[83]:

	Age_Group	Sample Size	LowerLimit	Upper Limit	Sample Mean
0	0-17	20000	8874.50	8991.99	8933.24
1	18-25	20000	9108.29	9228.38	9168.33
2	26-35	20000	9192.83	9309.98	9251.40
3	36-45	20000	9273.62	9390.53	9332.07
4	46-50	20000	9150.77	9265.33	9208.05
5	51-55	20000	9474.38	9593.92	9534.15
6	55+	20000	9277.54	9394.35	9335.95
7	0-17	20000	8864.14	9003.51	8933.82
8	18-25	20000	9099.21	9239.84	9169.53
9	26-35	20000	9181.11	9322.94	9252.02
10	36-45	20000	9260.09	9402.37	9331.23
11	46-50	20000	9139.54	9277.70	9208.62
12	51-55	20000	9464.63	9603.88	9534.25
13	55+	20000	9266.36	9406.10	9336.23
14	0-17	20000	8840.82	9026.06	8933.44
15	18-25	20000	9076.38	9261.56	9168.97
16	26-35	20000	9159.35	9344.58	9251.97
17	36-45	20000	9238.97	9422.47	9330.72
18	46-50	20000	9116.75	9301.97	9209.36
19	51-55	20000	9438.70	9629.29	9534.00
20	55+	20000	9246.92	9424.01	9335.46

\

	Confidence Interval	Interval Range
0	90	(8874.5, 8991.99)
1	90	(9108.29, 9228.38)
2	90	(9192.83, 9309.98)
3	90	(9273.62, 9390.53)

```
4                          90  (9150.77, 9265.33)
5                          90  (9474.38, 9593.92)
6                          90  (9277.54, 9394.35)
7                          95  (8864.14, 9003.51)
8                          95  (9099.21, 9239.84)
9                          95  (9181.11, 9322.94)
10                         95  (9260.09, 9402.37)
11                         95   (9139.54, 9277.7)
12                         95  (9464.63, 9603.88)
13                         95   (9266.36, 9406.1)
14                         99  (8840.82, 9026.06)
15                         99  (9076.38, 9261.56)
16                         99  (9159.35, 9344.58)
17                         99  (9238.97, 9422.47)
18                         99  (9116.75, 9301.97)
19                         99   (9438.7, 9629.29)
20                         99  (9246.92, 9424.01)
```

When Confidence Interval(CI) is `90`: - For Age-Group `0-17` the CI range is [8874.5, 8991.99] - For Age-Group `18-25` the CI range is [9108.29, 9228.38] - For Age-Group `26-35` the CI range is [9192.83, 9309.98] - For Age-Group `36-45` the CI range is [9273.62, 9390.53] - For Age-Group `46-50` the CI range is [9150.77, 9265.33] - For Age-Group `51-55` the CI range is [9474.38, 9593.92] - For Age-Group `55+` the CI range is [9277.54, 9394.35]

When Confidence Interval(CI) is `95`: - For Age-Group `0-17` the CI range is [8864.14, 9003.51] - For Age-Group `18-25` the CI range is [9099.21, 9239.84] - For Age-Group `26-35` the CI range is [9181.11, 9322.94] - For Age-Group `36-45` the CI range is [9260.09, 9402.37] - For Age-Group `46-50` the CI range is [9139.54, 9277.7] - For Age-Group `51-55` the CI range is [9464.63, 9603.88] - For Age-Group `55+` the CI range is [9266.36, 9406.1]

When Confidence Interval(CI) is `99`: - For Age-Group `0-17` the CI range is [8840.82, 9026.06] - For Age-Group `18-25` the CI range is [9076.38, 9261.56] - For Age-Group `26-35` the CI range is [9159.35, 9344.58] - For Age-Group `36-45` the CI range is [9238.97, 9422.47] - For Age-Group `46-50` the CI range is [9116.75, 9301.97] - For Age-Group `51-55` the CI range is [9438.7, 9629.29] - For Age-Group `55+` the CI range is [9246.92, 9424.01]

- For Age-Group `0-17` has the least purchase range among the other age-groups.
- The number of transaction done by age group `55+`are less but when it comes to purchase range it is higher.It may be due to high value purchases made by this group.

1 Insights

- As the sample size of the data increases the confidence intervals become more narrow. Hence larger data more insights.
- Males make up 75% of users, while females make up 25%. Clearly, men buy more than women do.
- City_Category C has more nuber of customers for walmart.Butmore number of transactions are done by City_Category B.
- More number of customers prefer purchases that are in range of 20k- 50k dollors.

- Customers with Occupation of 17 and female has significant amount in purchases. while female whose occupation is 10 has least number of purchases.
- There are significant number of customers in the age group of 26-35.
- Product_Category 1,5,8 are more prefered by customers aprt from other categories.Product_Category 9,17 are least preffred by the customers.
- Majority of the transactions (53.75 % of total transactions) are made by the customers having 1 or 2 years of stay in the current city. -35.85% of all unique customers are between the ages of 26 and 35; 19.81% are between the ages of 36 and 45; 18.15% are between the ages of 18 and 25; and 9.00% are between the ages of 46 and 50.
- customers in the 51 - 55 age group have the highest spending per transaction.while customers in the 0 - 17 age group have the lowest spending per transaction.
- significant portion of transactions (53.75%) come from customers who have recently moved to the current city.It may be due to purchase of new products when arriving to a new city.
- At 95% confidence level,
  - When at 95% confidence the average value for males falls approximately between $ 9392.6 and $ 9482.28. And for female it is $ 8693.07 and $ 8776.53.
  - When at 95% confidence the average value for Married falls approximately between $ 9217.27 and $ 9305.42. And for Unmarried it is $ 9221.78 and $ 9309.86.5.
  - At 95% confidence interval age-group 0-17 has least range of purchases.And 51-55 has highest range of purchases.

2 Recommendations

Targeted Marketing

- Males spent more money than that of Females, So company should focus on retaining the male customers and getting more male customers.
- we know that in the age-group of 0-17 we have lowest spending we can increase this by giving coupons and rewards.

City segmentation marketing

- City_Category C has more nuber of customers for walmart.But more number of transactions are done by City_Category B.Increase the stores based on the category of the city customers.
- Male customers living in City_Category C spend more money than other malecustomers living in B or C, Selling more products in the City_Category C will help the company increase the revenue.

Top-selling product categories

- The top five product categories such as - 1, 5, 8, & 11 have highest purchasing frequency. it means these are the products in these categories are liked more by customers.so increasing this kind of product categories during the black friday sales can prevent the out of stock.

offers for high-spending/frequent customers

- give special offers or coupons for the customers who spend above the average purchases.And give some special discounts for the customers who vist the walmart store more frequently.

New comers/ new migrants

- Target the customers who are recently moved to current city.Provide them welcome offers. this can help the walmart to secure thier customers for long time.

Feedback and reviews from the customers

- feedback and reviews from the customers after the black friday sales should be given highest priority.
- Improve the quantity and quality of the product and services based on the customer's feedback.