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

from scipy.stats import binom,norm,uniform,t
```

```
Start coding or generate with AI.
```

Introduction

Walmart is an American multinational retail corporation that operates a chain of supercenters, discount departmental stores, and grocery stores in the United States. Walmart has more than 100 million customers worldwide.

Objective The Management team at Walmart Inc. wants to analyze the customer purchase behavior (precisely, purchase amount) against the customer's gender and the various other factors to help the business make better decisions. They want to understand if the spending habits differ between male and female customers: Do women spend more on Black Friday than men?

About Data ● User_ID: User ID

● Product_ID: Product ID

● Gender: Sex of User

● Age: Age in bins

● Occupation: Occupation

● City_Category: Category of the City (A,B,C)

● StayInCurrentCityYears: Number of years stay in current city

● Marital_Status: Marital Status

● ProductCategory: Product Category

● Purchase: Purchase Amount

Loading Dataset

```
!wget 'https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094' -O 'walm
```

```
--2024-06-05 06:57:59--  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.cs
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 3.162.130.189, 3.162.130.111, 3.162.130.97, .
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|3.162.130.189|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 23027994 (22M) [text/plain]
Saving to: 'walmart.csv'

walmart.csv         100%[===================>]  21.96M  --.-KB/s    in 0.09s

2024-06-05 06:58:00 (235 MB/s) - 'walmart.csv' saved [23027994/23027994]
```

```
df = pd.read_csv('walmart.csv')
data=df.copy()
df.head(20)
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Curren
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	
5	1000003	P00193542	M	26-35	15	A	
6	1000004	P00184942	M	46-50	7	B	
7	1000004	P00346142	M	46-50	7	B	
8	1000004	P0097242	M	46-50	7	B	
9	1000005	P00274942	M	26-35	20	A	
10	1000005	P00251242	M	26-35	20	A	
11	1000005	P00014542	M	26-35	20	A	
12	1000005	P00031342	M	26-35	20	A	

Exploratory Data Analysis

```
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
```

```
df.size
```

```
5500680
```

```
df.shape
```

```
(550068, 10)
```

```
df.isna().sum()
```

```
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
```

```
df.duplicated().value_counts()
```

```
False    550068
Name: count, dtype: int64
```

```
df.nunique()
```

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

Observations

Walmart dataset has 10 features with almost 5L+ plus rows. There are no duplicate rows and no null values. Total 5891 customers have made purchases during the period of observations and 3631 different products were sold.

Convert all columns (except Purchase) to categorical type in the DataFrame

```
for _ in df.columns[:-1]:
  df[_]=df[_].astype('category')
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  category
 1   Product_ID                  550068 non-null  category
 2   Gender                      550068 non-null  category
 3   Age                         550068 non-null  category
 4   Occupation                  550068 non-null  category
 5   City_Category               550068 non-null  category
 6   Stay_In_Current_City_Years  550068 non-null  category
 7   Marital_Status              550068 non-null  category
 8   Product_Category            550068 non-null  category
 9   Purchase                    550068 non-null  int64
dtypes: category(9), int64(1)
memory usage: 10.3 MB
```

```
df.describe()
```

	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

Observation

There is significant difference between mean and std .. indicating outliers.

```
df.describe(include='category').T
```

	count	unique	top	freq
User_ID	550068	5891	1001680	1026
Product_ID	550068	3631	P00265242	1880
Gender	550068	2	M	414259
Age	550068	7	26-35	219587
Occupation	550068	21	4	72308
City_Category	550068	3	B	231173
Stay_In_Current_City_Years	550068	5	1	193821
Marital_Status	550068	2	0	324731
Product_Category	550068	20	5	150933

Observations:

Customer (1001680) has purchased more than others

Product (P00265242) is most bought item

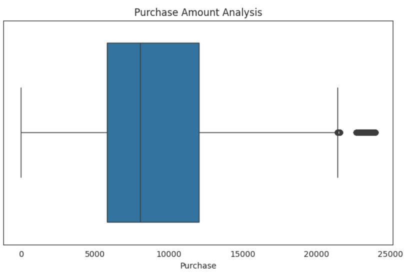
Most of the customers are Male

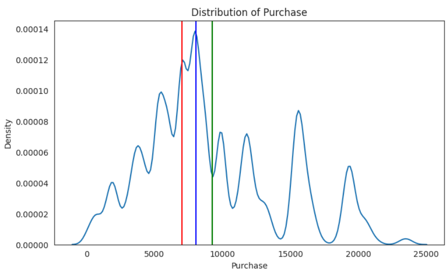
Most of customers lies in [26-35] Age bracket

Majority of the customers are Unmarried

Outlier detection

```
fig=plt.figure(figsize=(19,5))
sns.set_style('white')
plt.subplot(1,2,1)
plt.title('Purchase Amount Analysis')

sns.boxplot(data=df,x='Purchase',orient='h')

plt.subplot(1,2,2)
plt.title("Distribution of Purchase")
sns.kdeplot(x=df['Purchase'])
plt.axvline(df["Purchase"].mean(),color="g")
plt.axvline(df["Purchase"].median(),color="b")
plt.axvline(df["Purchase"].mode()[0],color="r")

plt.show()
```





Observations

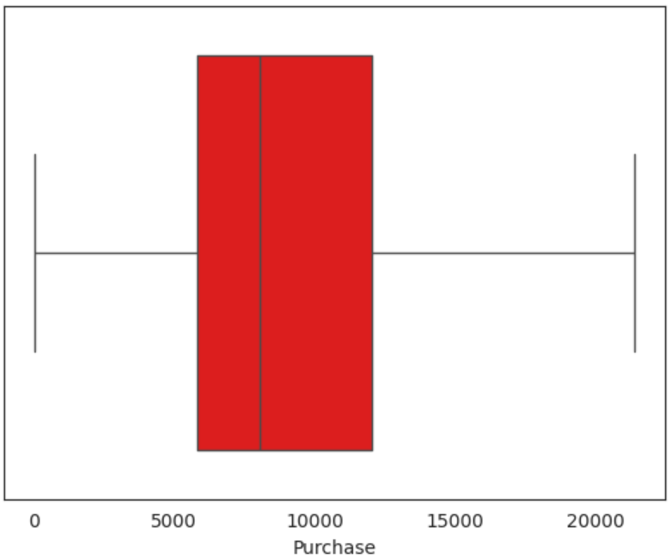
There are outliers in purchase amount. While observing the distribution of purchase amount from density plot. It is quite obvious that the distribution is right skewed means majority of data concentrated on left side. Majority of customer purchase within 5,000 - 20,000 range.

Handling Outliers

```
# Calculating Q3,Q1 and IQR

Q3=np.percentile(df['Purchase'],75)
Q1=np.percentile(df['Purchase'],25)
IQR=Q3-Q1

#Calculating upper and lower bound values

upper_bound= Q3+ 1.5*IQR
lower_bound= Q1-1.5*IQR

#Outlier count

upper_count_values= len(df[df['Purchase']>upper_bound])
lower_count_values= len(df[df['Purchase']<lower_bound])
total_values= upper_count_values+lower_count_values

print(" Upper count values ", upper_count_values)
print("Lower count values ", lower_count_values)
print("Total outlier values", total_values)
```

```
 Upper count values  2677
Lower count values  0
Total outlier values 2677
```

```
clipped_data=np.clip(df['Purchase'],lower_bound,upper_bound)

sns.boxplot(data=clipped_data,orient='h',color='r')
plt.show()
```

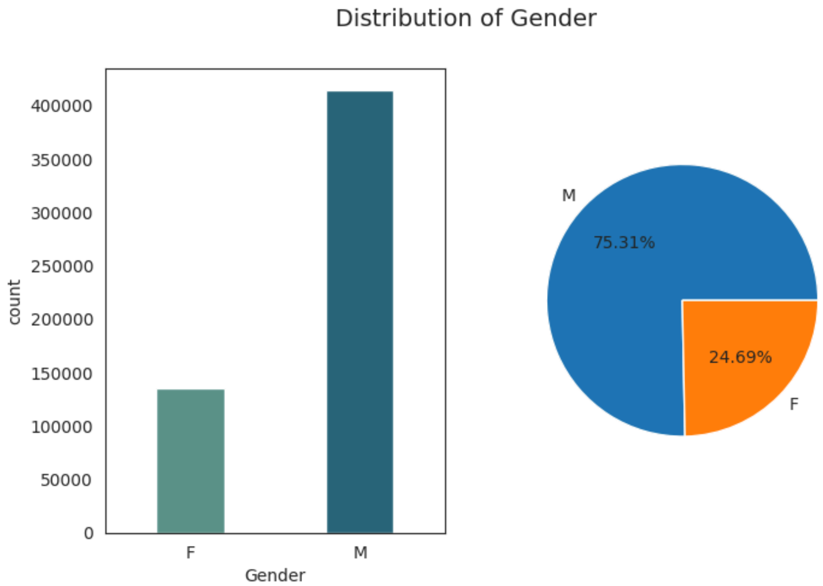



```
#
# Map numerical values in 'Marital_Status' to categorical labels

df['Marital_Status']=df['Marital_Status'].apply(lambda x:'Married' if x==1 else 'Single')
df.head(20)
```

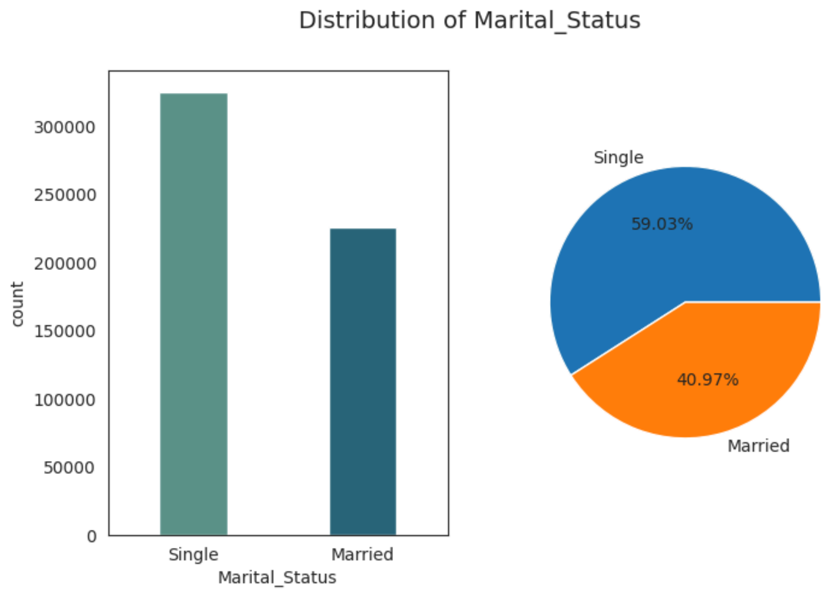
	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Curren
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	
5	1000003	P00193542	M	26-35	15	A	
6	1000004	P00184942	M	46-50	7	B	
7	1000004	P00346142	M	46-50	7	B	
8	1000004	P0097242	M	46-50	7	B	
9	1000005	P00274942	M	26-35	20	A	
10	1000005	P00251242	M	26-35	20	A	
11	1000005	P00014542	M	26-35	20	A	
12	1000005	P00031342	M	26-35	20	A	
13	1000005	P00145042	M	26-	20	A	

Univariate Analysis

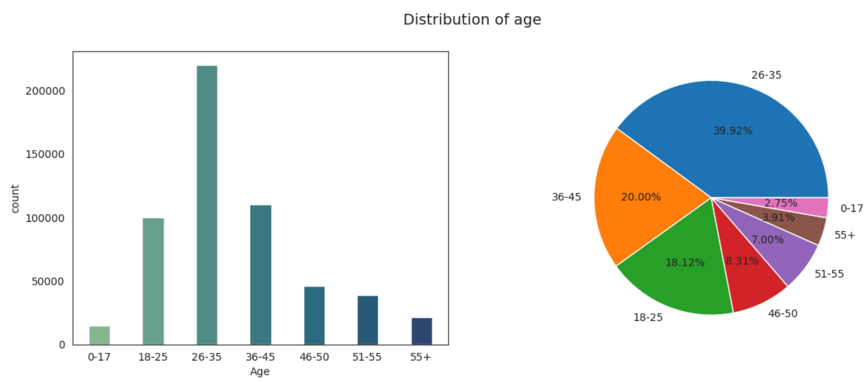
```
fig=plt.figure(figsize=(8,5))
sns.set_style(style='white')
plt.subplot(1,2,1)
sns.countplot(data=df, x="Gender", palette="crest", hue='Gender', legend=False,width=0.4)

#first row sec column
plt.subplot(1,2,2)
plt.pie(df["Gender"].value_counts(),
        labels = df["Gender"].value_counts().index,
        autopct = '%1.2f%%')
plt.suptitle('Distribution of Gender', fontsize = 14)

plt.show()
```



```
fig=plt.figure(figsize=(8,5))
sns.set_style(style='white')
plt.subplot(1,2,1)
sns.countplot(data=df, x="Marital_Status", palette="crest", hue='Marital_Status', legend=False,width=0.4)

#first row sec column
plt.subplot(1,2,2)
plt.pie(df["Marital_Status"].value_counts(),
        labels = df["Marital_Status"].value_counts().index,
        autopct = '%1.2f%%')
plt.suptitle('Distribution of Marital_Status', fontsize = 14)

plt.show()
```



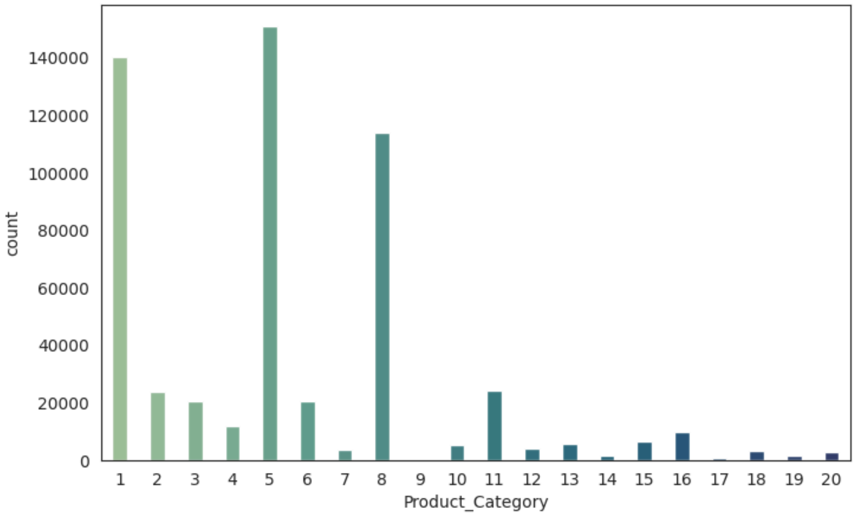

```
fig=plt.figure(figsize=(14,5))
sns.set_style(style='white')
plt.subplot(1,2,1)
sns.countplot(data=df, x="Age", palette="crest", hue='Age', legend=False,width=0.4)

#first row sec column
plt.subplot(1,2,2)
plt.pie(df["Age"].value_counts(),
        labels = df["Age"].value_counts().index,
        autopct = '%1.2f%%')
plt.suptitle('Distribution of age', fontsize = 14)

plt.show()
```



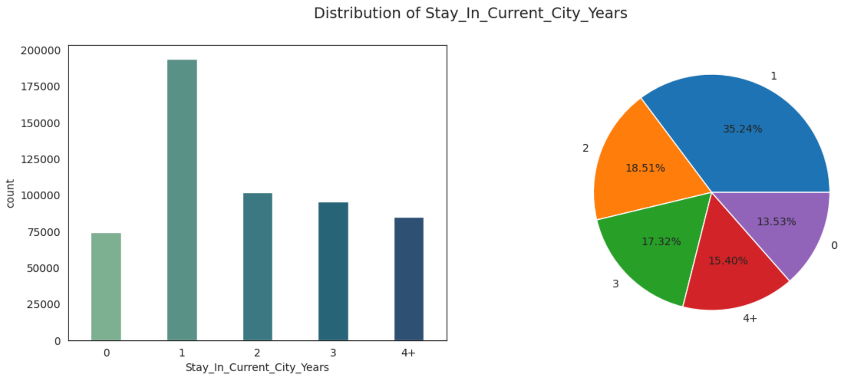
```
df.columns
```

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

```
fig=plt.figure(figsize=(18,5))
sns.set_style(style='white')
plt.subplot(1,2,1)
sns.countplot(data=df, x="Product_Category", palette="crest", hue='Product_Category', legend=False,width=0.4)

plt.show()
```




```
fig=plt.figure(figsize=(14,5))
sns.set_style(style='white')
plt.subplot(1,2,1)
sns.countplot(data=df, x="Stay_In_Current_City_Years", palette="crest", hue='Stay_In_Current_City_Years', legend=False,widtl

#first row sec column
plt.subplot(1,2,2)
plt.pie(df['Stay_In_Current_City_Years'].value_counts(),
        labels = df['Stay_In_Current_City_Years'].value_counts().index,
        autopct = '%1.2f%%')
plt.suptitle('Distribution of Stay_In_Current_City_Years', fontsize = 14)

plt.show()
```



Observations: Age Group Distribution:

The age group '26-35' has the highest count, indicating that customers in this age range make the most purchases. It is followed by the age groups '36-45' and '18-25'.

City Category Distribution:

City_Category 'B' has the highest count, indicating that customers from City_Category 'B' have made the most purchases. City_Category 'C' and 'A' follow in terms of count.

Marital Status Impact:

Customers with a marital status of 'Single' have a higher count compared to those who are 'Married', suggesting that single individuals make more purchases in the dataset.

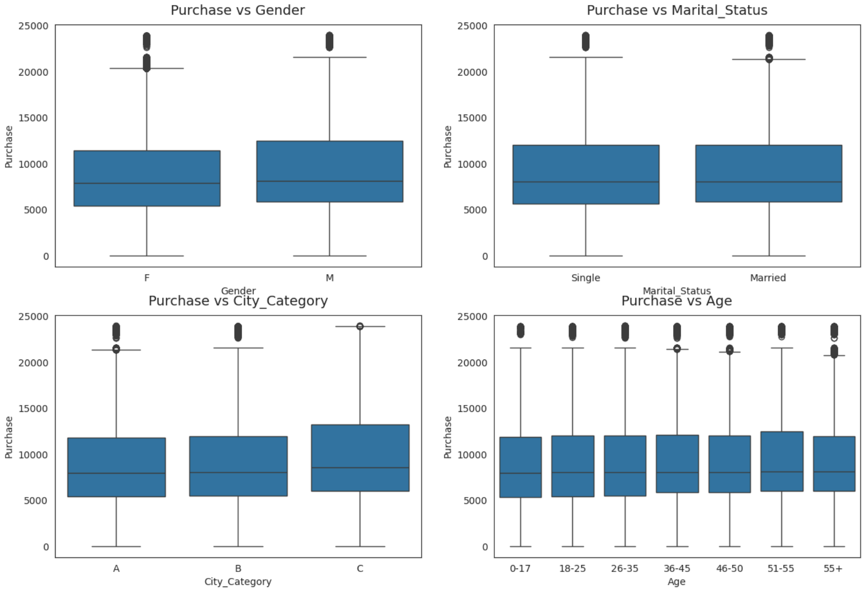
City Residence Duration Impact:

Customers who have stayed in their current city for more than 1 year show a higher purchase tendency, suggesting a positive correlation between the duration of stay and purchasing behavior. Product Category Purchase Analysis:

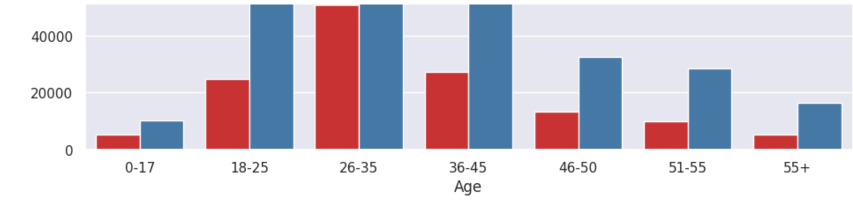
Product categories '1' and '5' exhibit higher purchase amounts, indicating that these categories contribute significantly to the overall sales revenue.

Bivariate Analysis

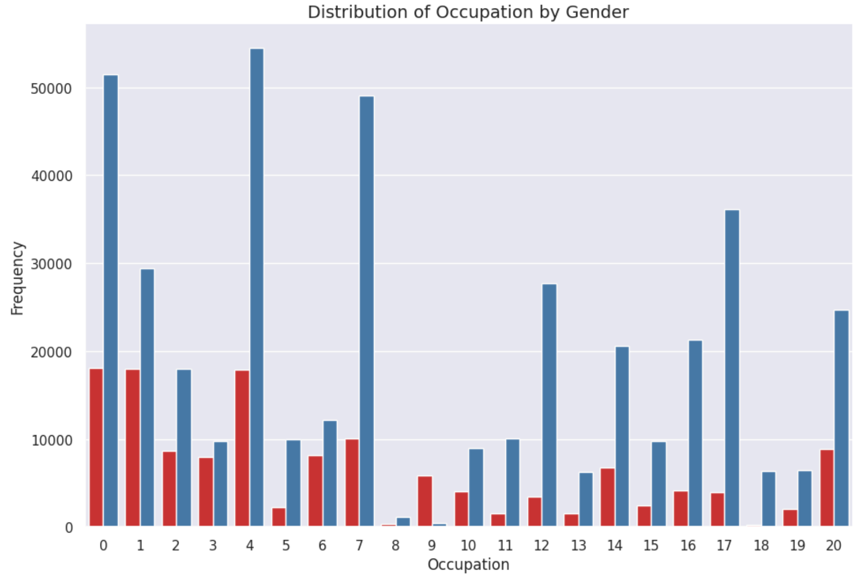
```
cat_col = ["Gender", "Marital_Status", "City_Category", "Age"]

fig, axs = plt.subplots(nrows=2, ncols = 2, figsize=(15,10))
k = 0
sns.set_style("dark")
for i in range(2):
    for j in range(2):
        sns.boxplot(data=df, x=cat_col[k], y="Purchase",  ax=axs[i, j])
        axs[i, j].set_title("Purchase vs " + cat_col[k], pad = 10, fontsize = 14)
        k += 1
plt.show()
```



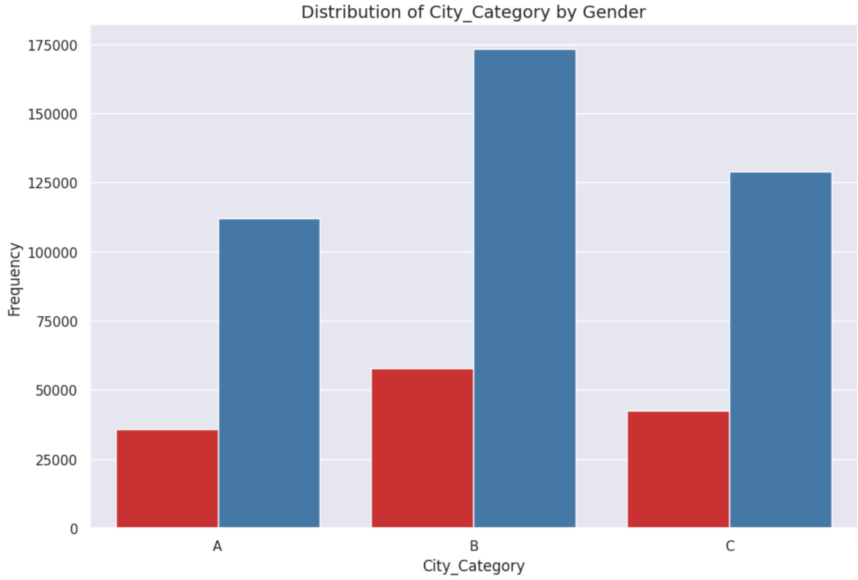

```
category = ['Age', 'Occupation', 'City_Category', 'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category']


plt.figure(figsize=(10, 40))
sns.set(style='darkgrid')

# Plot each categorical column
for i, col in enumerate(category, 1):
    plt.subplot(6, 1, i)
    sns.countplot(data=df, x=col, hue='Gender', palette='Set1', legend=False, dodge=True)
    sns.despine()

    # Set labels and title
    plt.xlabel(f'{col}', fontsize=12)
    plt.ylabel('Frequency', fontsize=12)
    plt.title(f'Distribution of {col} by Gender', fontsize=14, fontfamily='sans-serif')

    plt.tight_layout()

plt.show()
```



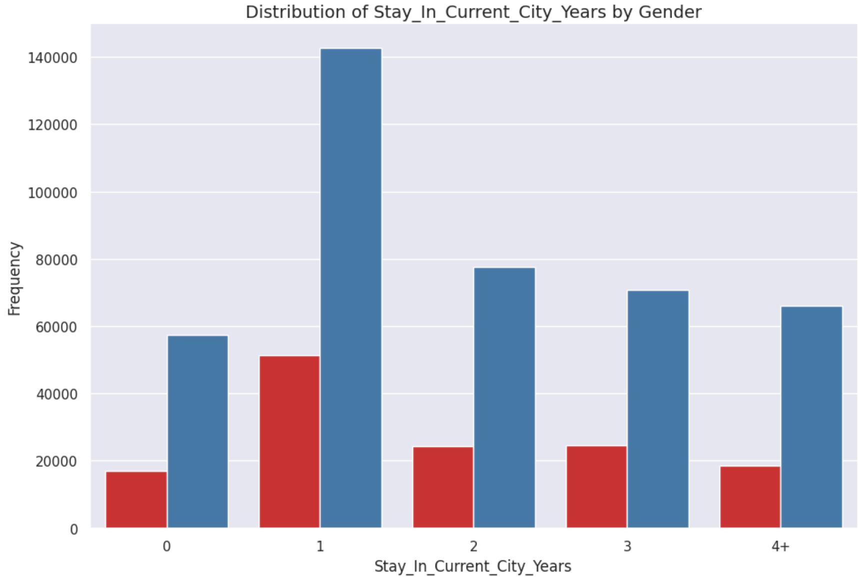
Distribution of Occupation by Gender



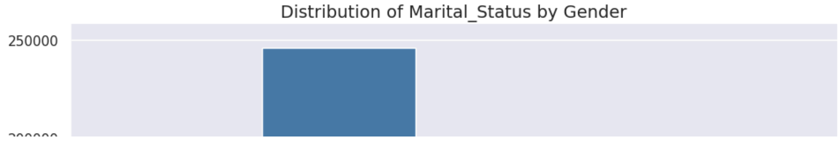
Distribution of City_Category by Gender

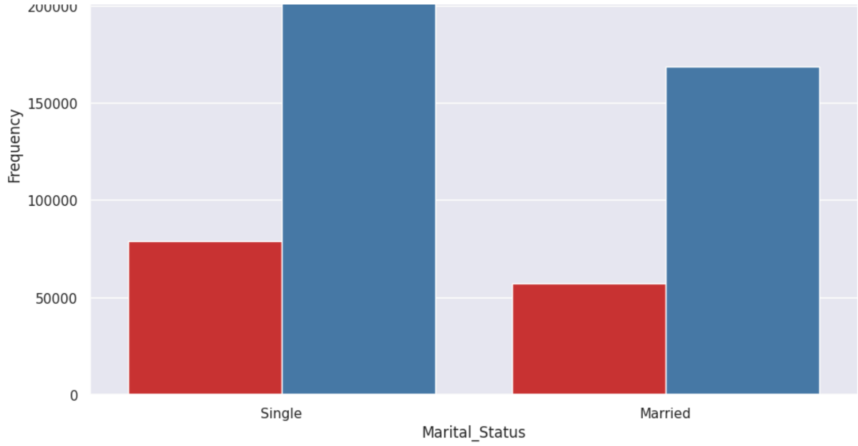


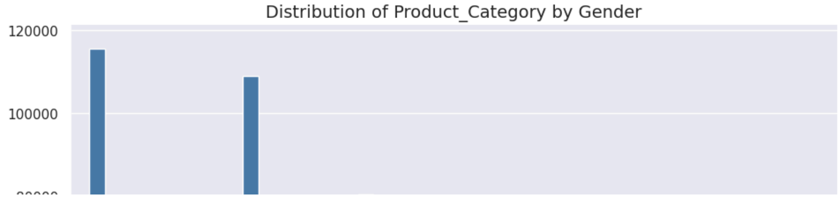
Distribution of Stay_In_Current_City_Years by Gender



Distribution of Marital_Status by Gender




200000
150000
100000
50000
0
Frequency
Single
Married
Marital_Status


Distribution of Product_Category by Gender
120000
100000

Insights:

Gender-Related Purchase Analysis:

Across various age groups, males tend to have higher purchase counts compared to females, with the age group '26-35' showing the most significant difference.

Occupation-Related Purchase Analysis:

Occupations '0' and '4' show the highest purchase counts, suggesting that individuals in these occupations contribute significantly to overall sales, with '4' having notably higher purchases than others. City Category-Related Purchase Analysis:

City_Category 'B' has the highest purchase counts for both genders, indicating that customers residing in City_Category 'B' contribute significantly to overall sales compared to 'A' and 'C'. Stay in Current City Duration Impact:

Customers who have stayed in their current city for 1 year exhibit the highest purchase counts, suggesting that individuals with a 1-year residence duration have a higher tendency to make purchases compared to other durations. Marital Status-Related Purchase Analysis:

Individuals with a marital status of 'Single' have higher purchase counts compared to those who are 'Married', indicating that single individuals contribute more to overall sales. Product Category-Related Purchase Analysis:

Product Category '1' has the highest purchase counts, indicating that it significantly contributes to overall sales. Product Categories '5' and '8' also show notable purchase counts.

Double-click (or enter) to edit

```
df.columns
```

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

```
df_new=data[['Occupation','Product_Category','Purchase']]
df_new.corr()
```

	Occupation	Product_Category	Purchase
Occupation	1.000000	-0.007618	0.020833
Product_Category	-0.007618	1.000000	-0.343703
Purchase	0.020833	-0.343703	1.000000

```
sns.heatmap(data=df_new.corr(),annot=True, cmap='Blues')
```

<Axes: >

Balck friday Sales analysis on gender

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

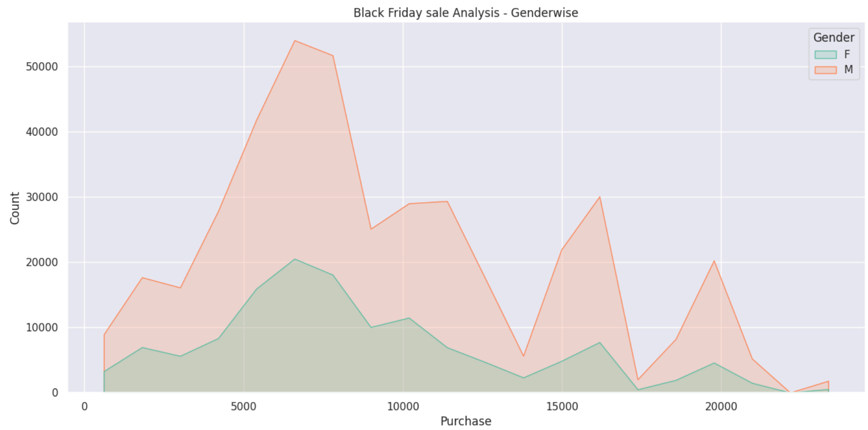
Start coding or generate with AI.

```
gender_data=df.groupby('Gender').agg({'Purchase':'mean'}).reset_index()
gender_data
```

	Gender	Purchase
0	F	8734.565765
1	M	9437.526040

Next steps: Generate code with gender_data View recommended plots

```
plt.figure(figsize=(15,7))
sns.set(style='darkgrid')
sns.histplot(data=df, x = "Purchase", bins=20, hue = "Gender",element='poly',palette= 'Set2')
sns.despine()
plt.title('Black Friday sale Analysis - Genderwise')
plt.show()
```



Insights:

Men spent more money than women during the Black Friday sale.

The total number of male customers (4225) exceeds the total number of female customers (1666).

The average amount spent by male customers (9437) is higher than the average amount spent by female customers (8734).

With a larger male customer base, it is likely that men will make more purchases compared to females.

The higher sales among male customers could be attributed to a product range better suited to their preferences, leading to increased sales of products targeted towards men.

```
gender_data['Purchase Percentage']=gender_data['Purchase']/df['Purchase'].sum()*100
gender_data
```

	Gender	Purchase	Purchase Percentage
0	F	8734.565765	0.000171
1	M	9437.526040	0.000185

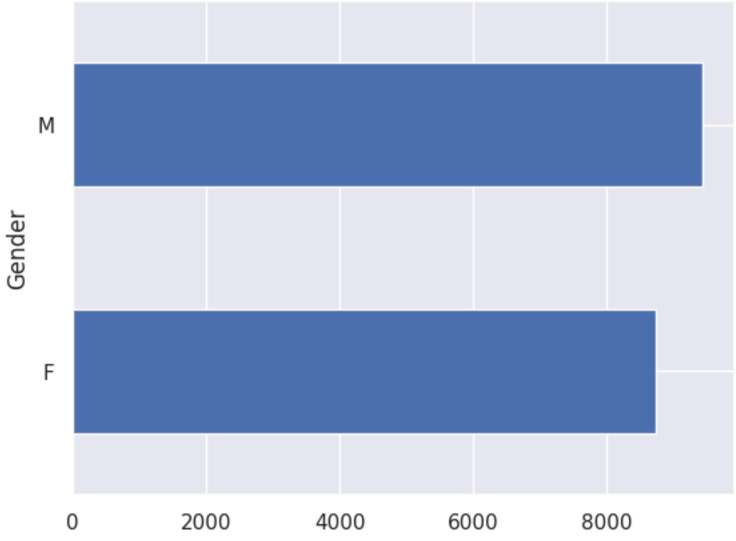
Next steps: Generate code with `gender_data` View recommended plots

Start coding or generate with AI.

Average Purchase by Gender

```
# @title Average Purchase by Gender

gender_data.groupby('Gender')['Purchase'].mean().plot(kind='barh')
```

<Axes: ylabel='Gender'>

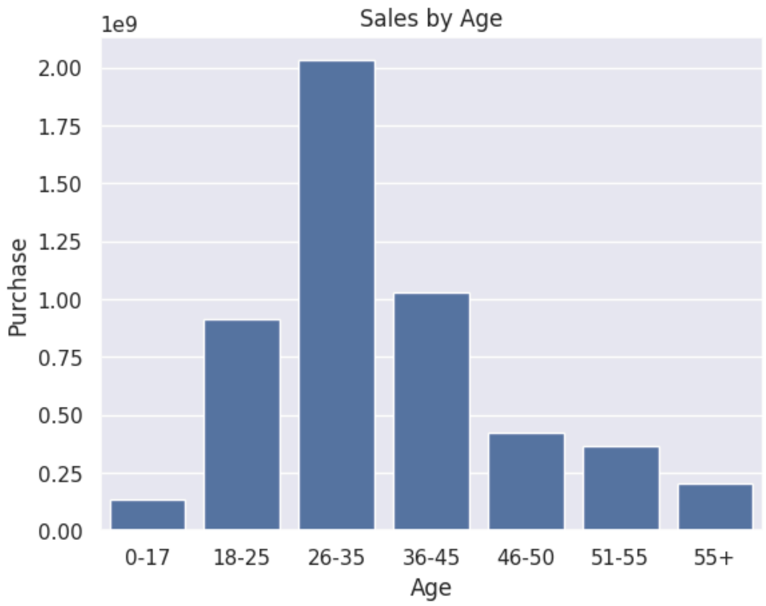



```
age_data=df.groupby(['Age']).agg({'Purchase':'sum'}).reset_index()
age_data
```

	Age	Purchase
0	0-17	134913183
1	18-25	913848675
2	26-35	2031770578
3	36-45	1026569884
4	46-50	420843403
5	51-55	367099644
6	55+	200767375

Next steps: Generate code with age_data View recommended plots

```
sns.barplot(data=age_data,x='Age',y='Purchase')
plt.title("Sales by Age")
plt.show()
```



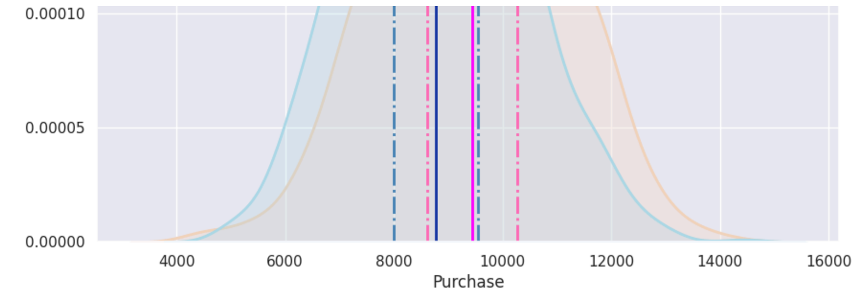
CLT and Confidence Intervals

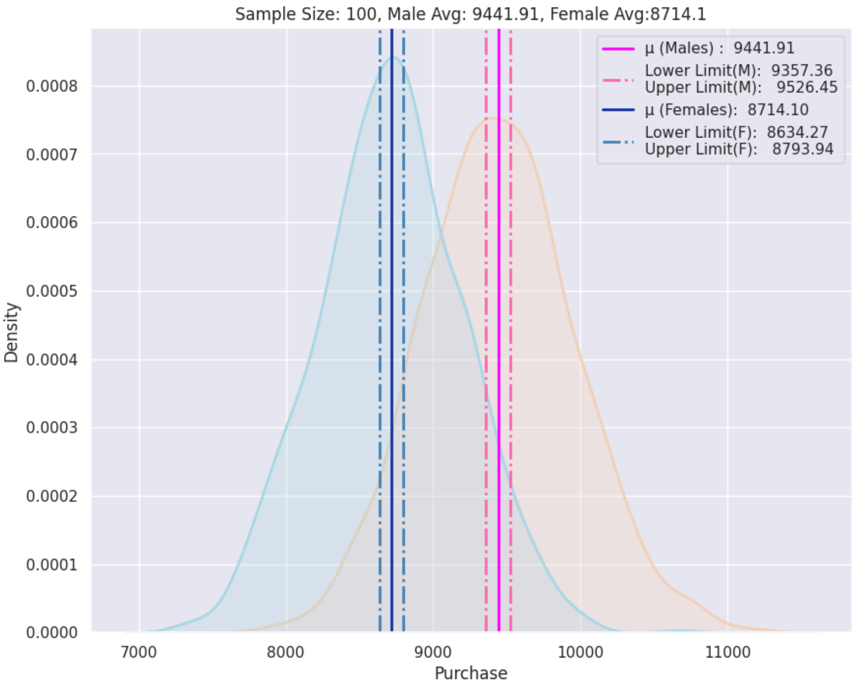
Male Vs Female Purchase Values

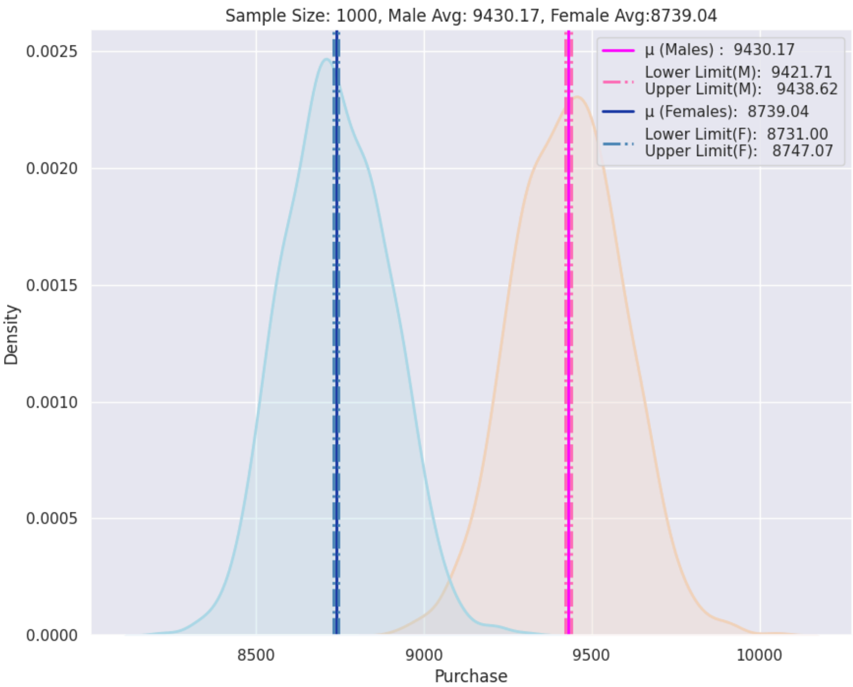
```
df_male = df[df['Gender']=='M']
df_female = df[df['Gender']=='F']


def sampling(sample1,sample2,sample_size,itr_size,ci):
    ci = ci/100

    plt.figure(figsize=(10,8))
    sample1_n = [np.mean(sample1.sample(sample_size)) for i in range(itr_size)]
    sample2_n = [np.mean(sample2.sample(sample_size)) for i in range(itr_size)]

    # For Sample1's means
    mean1 = np.mean(sample1_n)
    sigma1 = np.std(sample1_n)
    # sem1 = sem(sample1_n)

    # lower_limit_1 = norm.ppf((1-ci)/2) * sigma1 + mean1
    # upper_limit_1 = norm.ppf(ci+(1-ci)/2) * sigma1 + mean1

    ci_arr1= norm.interval(confidence=ci,loc=np.mean(sample1_n),scale=np.std(sample1_n)/np.sqrt(sample_size))
    lower_limit_1 = ci_arr1[0]
    upper_limit_1 = ci_arr1[1]

    # For Sample2's means
    mean2 = np.mean(sample2_n)
    sigma2 = np.std(sample2_n)
    ci_arr2= norm.interval(confidence=ci,loc=np.mean(sample2_n),scale=np.std(sample2_n)/np.sqrt(sample_size))
    lower_limit_2 = ci_arr2[0]
    upper_limit_2 = ci_arr2[1]

    sns.kdeplot(data = sample1_n, color="#F2D2BD", fill = True, linewidth = 2)
    label_mean1=("μ (Males) :  {:.2f}".format(mean1))
    plt.axvline(mean1, color = '#FF00FF', linestyle = 'solid', linewidth = 2, label=label_mean1)
    label_limits1=("Lower Limit(M):  {:.2f}\nUpper Limit(M):   {:.2f}".format(lower_limit_1,upper_limit_1))
    plt.axvline(lower_limit_1, color = '#FF69B4', linestyle = 'dashdot', linewidth = 2, label=label_limits1)
    plt.axvline(upper_limit_1, color = '#FF69B4', linestyle = 'dashdot', linewidth = 2)

    sns.kdeplot(data = sample2_n ,color='#ADD8E6', fill = True, linewidth = 2)
    label_mean2=("μ (Females):  {:.2f}".format(mean2))
    plt.axvline(mean2, color = '#1434A4', linestyle = 'solid', linewidth = 2, label=label_mean2)
    label_limits2=("Lower Limit(F):  {:.2f}\nUpper Limit(F):   {:.2f}".format(lower_limit_2,upper_limit_2))
    plt.axvline(lower_limit_2, color = '#4682B4', linestyle = 'dashdot', linewidth = 2, label=label_limits2)
    plt.axvline(upper_limit_2, color = '#4682B4', linestyle = 'dashdot', linewidth = 2)

    plt.title(f"Sample Size: {sample_size}, Male Avg: {np.round(mean1, 2)}, Female Avg:{np.round(mean2, 2)}")
    plt.legend(loc = 'upper right')
    plt.xlabel('Purchase')
    plt.ylabel('Density')


    return round(mean1,2), round(mean2,2), round(lower_limit_1,2), round(upper_limit_1,2), round(lower_limit_2,2), round(upp
```

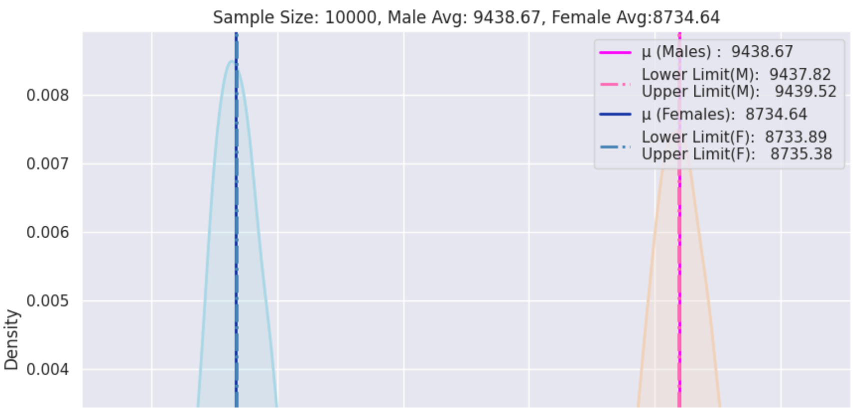
Lets plot the mean of 1000 Random Samples of sizes 10,100,1000 and 10000 with 90% Confidence Interval

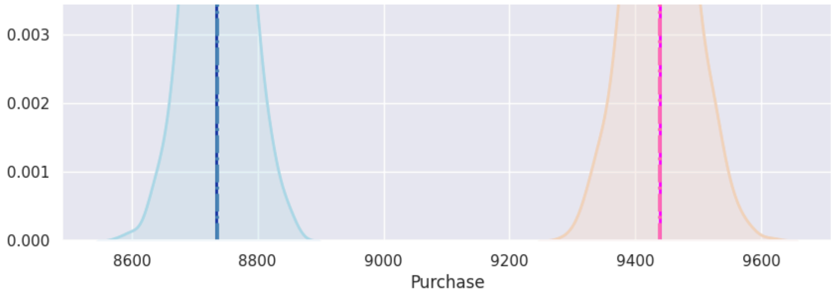
```
sample_sizes = sample_sizes = [10,100,1000,10000,100000]
ci = 90
itr_size = 1000

res = pd.DataFrame(columns = ['Gender','Sample Size','Lower Limit','Upper Limit','Sample Mean','Confidence Interval','Interv

for i in sample_sizes:
    m_avg, f_avg, ll_m, ul_m, ll_f, ul_f = sampling(df_male['Purchase'],df_female['Purchase'],i,itr_size,ci)

    res.loc[len(res.index)] = {'Gender':'M','Sample Size':i,'Lower Limit':ll_m,'Upper Limit':ul_m,'Sample Mean':m_avg,'Confi
    res.loc[len(res.index)] = {'Gender':'F','Sample Size':i,'Lower Limit':ll_f,'Upper Limit':ul_f,'Sample Mean':f_avg,'Confi
```

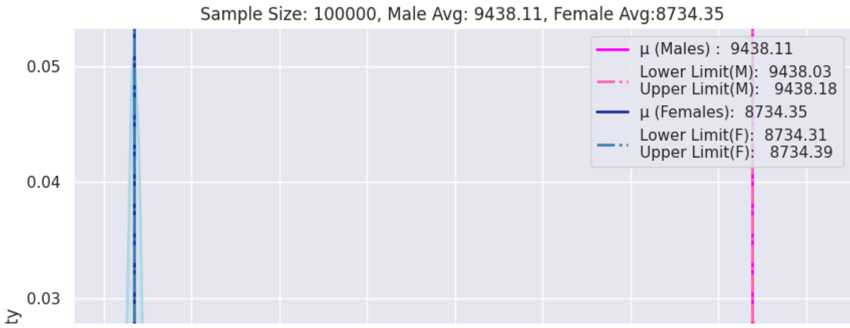

0.00010
0.00005
0.00000
4000
6000
8000
10000
12000
14000
16000
Purchase


Sample Size: 100, Male Avg: 9441.91, Female Avg:8714.1
μ (Males) : 9441.91
Lower Limit(M): 9357.36
Upper Limit(M): 9526.45
μ (Females): 8714.10
Lower Limit(F): 8634.27
Upper Limit(F): 8793.94
Density
Purchase


Sample Size: 1000, Male Avg: 9430.17, Female Avg:8739.04
μ (Males) : 9430.17
Lower Limit(M): 9421.71
Upper Limit(M): 9438.62
μ (Females): 8739.04
Lower Limit(F): 8731.00
Upper Limit(F): 8747.07
Density
Purchase

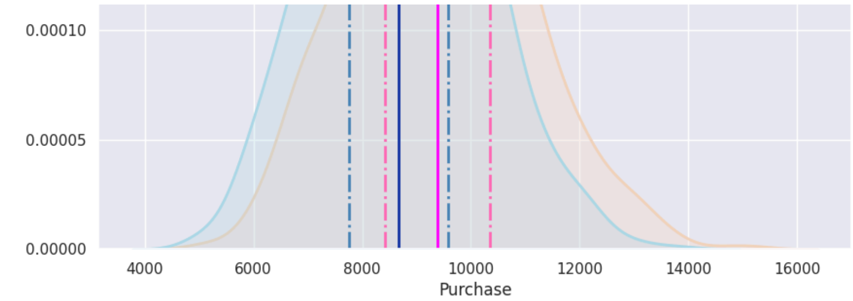

Sample Size: 10000, Male Avg: 9438.67, Female Avg:8734.64
μ (Males) : 9438.67
Lower Limit(M): 9437.82
Upper Limit(M): 9439.52
μ (Females): 8734.64
Lower Limit(F): 8733.89
Upper Limit(F): 8735.38
Density

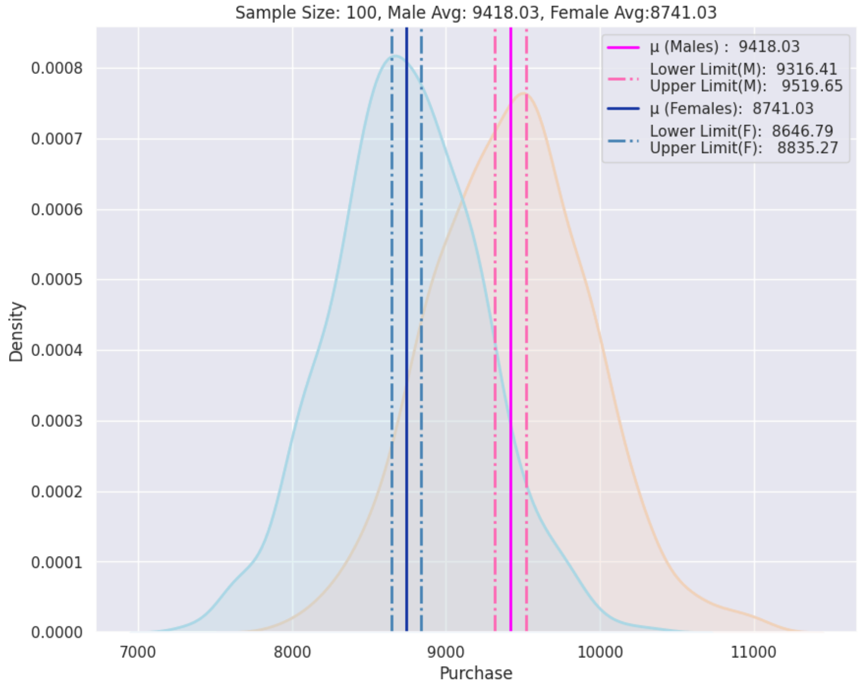

0.003
0.002
0.001
0.000
8600
8800
9000
9200
9400
9600
Purchase

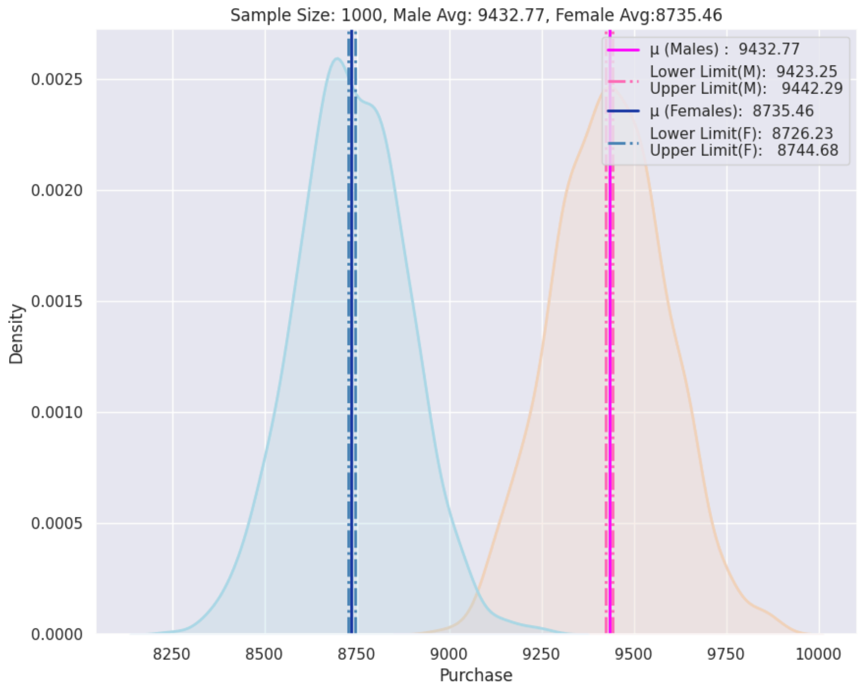

Sample Size: 100000, Male Avg: 9438.11, Female Avg:8734.35
μ (Males) : 9438.11
Lower Limit(M): 9438.03
Upper Limit(M): 9438.18
μ (Females): 8734.35
Lower Limit(F): 8734.31
Upper Limit(F): 8734.39
0.05
0.04
0.03
ty

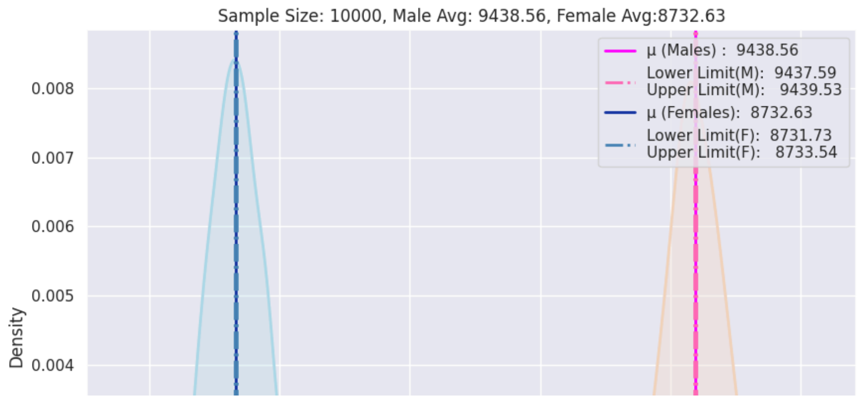
Lets plot the mean of 1000 Random Samples of sizes 10,100,1000,10000 and 100000 with 95% Confidence Interval

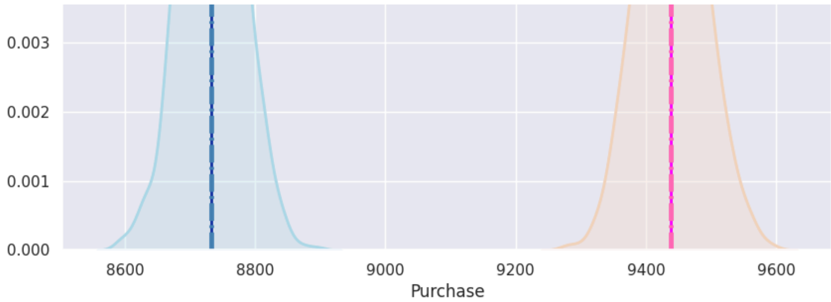
```
sample_sizes = sample_sizes = [10,100,1000,10000,100000]
ci = 95
itr_size = 1000

# res = pd.DataFrame(columns = ['Gender','Sample Size','Lower Limit','Upper Limit','Sample Mean','Confidence Interval','Inte

for i in sample_sizes:
    m_avg, f_avg, ll_m, ul_m, ll_f, ul_f = sampling(df_male['Purchase'],df_female['Purchase'],i,itr_size,ci)

    res.loc[len(res.index)] = {'Gender':'M','Sample Size':i,'Lower Limit':ll_m,'Upper Limit':ul_m,'Sample Mean':m_avg,'Confi
    res.loc[len(res.index)] = {'Gender':'F','Sample Size':i,'Lower Limit':ll_f,'Upper Limit':ul_f,'Sample Mean':f_avg,'Confi
```

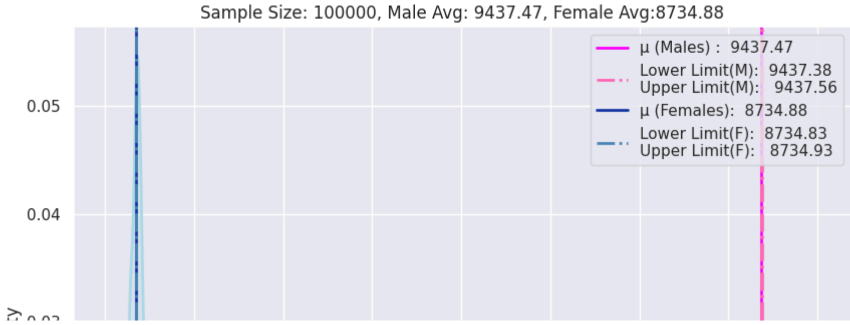

0.00010
0.00005
0.00000
4000 6000 8000 10000 12000 14000 16000
Purchase


Sample Size: 100, Male Avg: 9418.03, Female Avg:8741.03
μ (Males) : 9418.03
Lower Limit(M): 9316.41
Upper Limit(M): 9519.65
μ (Females): 8741.03
Lower Limit(F): 8646.79
Upper Limit(F): 8835.27
Density
0.0000 0.0001 0.0002 0.0003 0.0004 0.0005 0.0006 0.0007 0.0008
7000 8000 9000 10000 11000
Purchase


Sample Size: 1000, Male Avg: 9432.77, Female Avg:8735.46
μ (Males) : 9432.77
Lower Limit(M): 9423.25
Upper Limit(M): 9442.29
μ (Females): 8735.46
Lower Limit(F): 8726.23
Upper Limit(F): 8744.68
Density
0.0000 0.0005 0.0010 0.0015 0.0020 0.0025
8250 8500 8750 9000 9250 9500 9750 10000
Purchase


Sample Size: 10000, Male Avg: 9438.56, Female Avg:8732.63
μ (Males) : 9438.56
Lower Limit(M): 9437.59
Upper Limit(M): 9439.53
μ (Females): 8732.63
Lower Limit(F): 8731.73
Upper Limit(F): 8733.54
Density
0.004 0.005 0.006 0.007 0.008


0.003
0.002
0.001
0.000
8600
8800
9000
9200
9400
9600
Purchase


Sample Size: 100000, Male Avg: 9437.47, Female Avg:8734.88
μ (Males) : 9437.47
Lower Limit(M): 9437.38
Upper Limit(M): 9437.56
μ (Females): 8734.88
Lower Limit(F): 8734.83
Upper Limit(F): 8734.93
0.05
0.04

```
res
```

	Gender	Sample Size	Lower Limit	Upper Limit	Sample Mean	Confidence Interval	Interval Range	Range
0	M	10	8615.36	10273.07	9444.21	90	[8615.36, 10273.07]	1657.71
1	F	10	7994.06	9551.67	8772.87	90	[7994.06, 9551.67]	1557.61
2	M	100	9357.36	9526.45	9441.91	90	[9357.36, 9526.45]	169.09
3	F	100	8634.27	8793.94	8714.10	90	[8634.27, 8793.94]	159.67
4	M	1000	9421.71	9438.62	9430.17	90	[9421.71, 9438.62]	16.91
5	F	1000	8731.00	8747.07	8739.04	90	[8731.0, 8747.07]	16.07
6	M	10000	9437.82	9439.52	9438.67	90	[9437.82, 9439.52]	1.70
7	F	10000	8733.89	8735.38	8734.64	90	[8733.89, 8735.38]	1.49
8	M	100000	9438.03	9438.18	9438.11	90	[9438.03, 9438.18]	0.15
9	F	100000	8734.31	8734.39	8734.35	90	[8734.31, 8734.39]	0.08
10	M	10	8425.92	10349.67	9387.79	95	[8425.92, 10349.67]	1923.75
11	F	10	7759.39	9588.01	8673.70	95	[7759.39, 9588.01]	1828.62
12	M	100	9316.41	9519.65	9418.03	95	[9316.41, 9519.65]	203.24

Next steps: Generate code with `res` View recommended plots

Observations: We can observe that

The CI with 90% confidence for sample size 10 for Males is [6653.41, 12210.87]

The CI with 90% confidence for sample size 10 for Females is [6245.08, 11265.77]

For Sample size 10 The confidence interval for both Male and Female is overlapping

and as the sample size increases, we can see the interval ranges seperating and then finally they both dont overalap.

The CI with 90% confidence for sample size 100000 for Males is [9415.08, 9460.27]

The CI with 90% confidence for sample size 100000 for Females is [8721.97, 8747.07]

For Sample size 100000 The confidence interval for both Male and Female is now not overlapping.

We can also observe the same with 95% Confidence.

The CI with 95% confidence for sample size 10 for Males is [6335.11, 12484.27]

The CI with 95% confidence for sample size 10 for Females is [5728.62, 11778.12]

For Sample size 10 The confidence interval for both Male and Female is overlapping

and as the sample size increases, we can see the interval ranges seperating and then finally they both dont overalap.

The CI with 95% confidence for sample size 100000 for Males is [9410.99, 9465.95]

The CI with 95% confidence for sample size 100000 for Females is [8719.59, 8750.12]

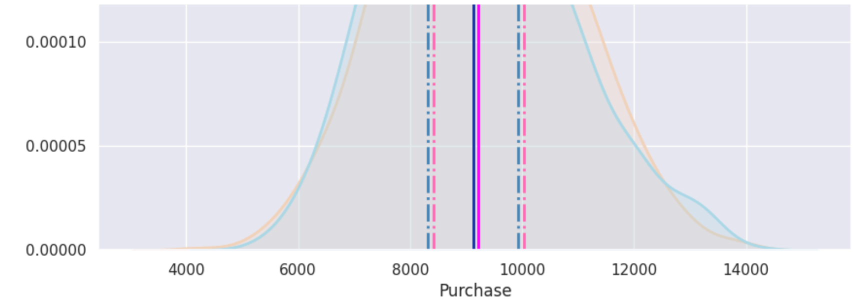
For Sample size 100000 The confidence interval for both Male and Female is now not overlapping.

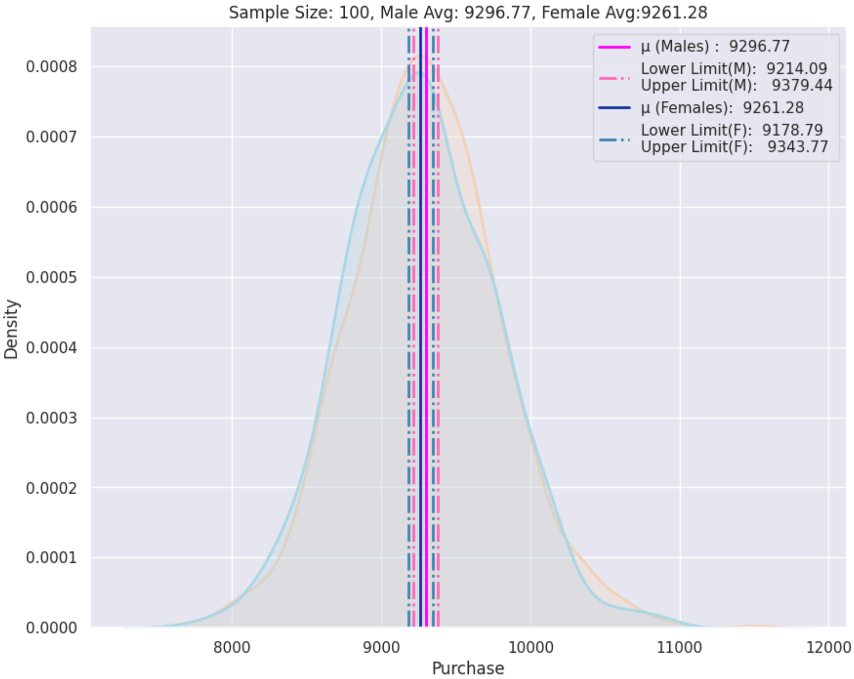
```
df_married = df[df['Marital_Status'] == 'Married']
df_unmarried = df[df['Marital_Status'] == 'Single']
df_unmarried
```

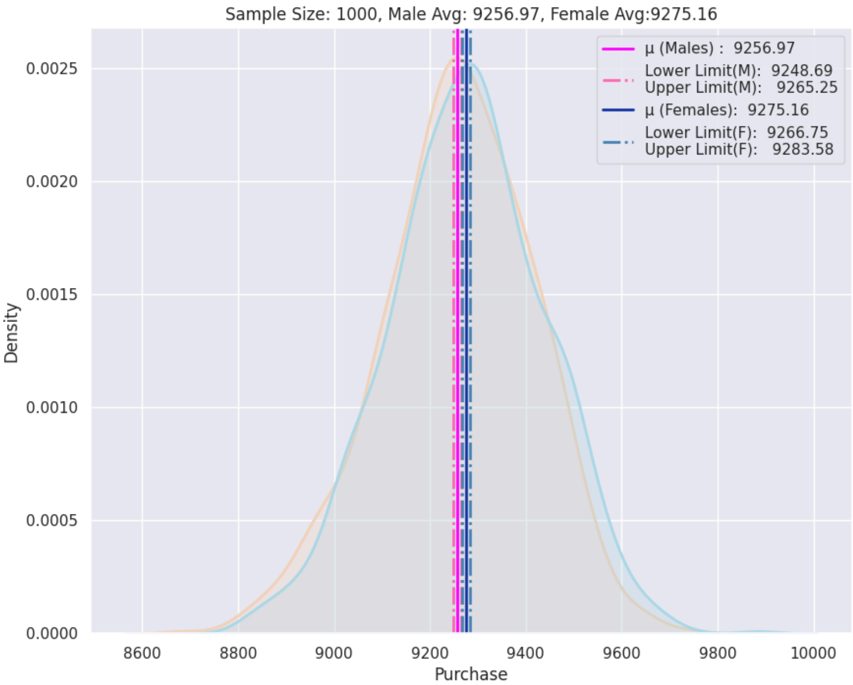
	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Cu
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	
...	...	...	...	...	...	...	
550056	1006022	P00375436	M	26-35	17	C	
550059	1006025	P00370853	F	26-35	1	B	
550062	1006032	P00372445	M	46-	7	A	

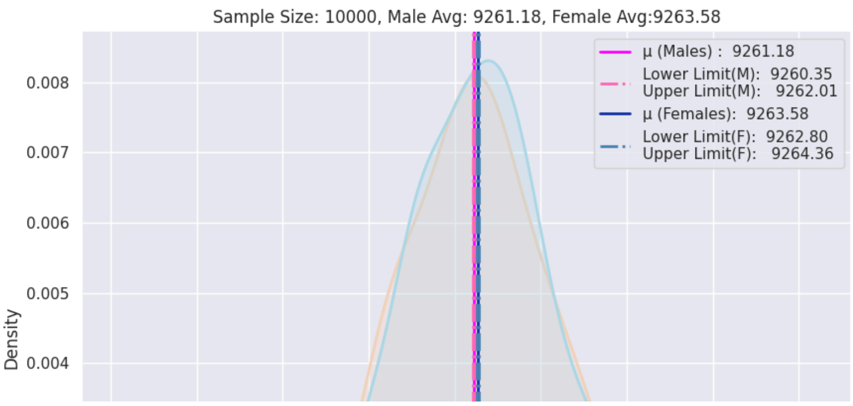
Lets plot the mean of 1000 Random Samples of sizes 10,100,1000,10000 and 100000 with 90% Confidence Interval

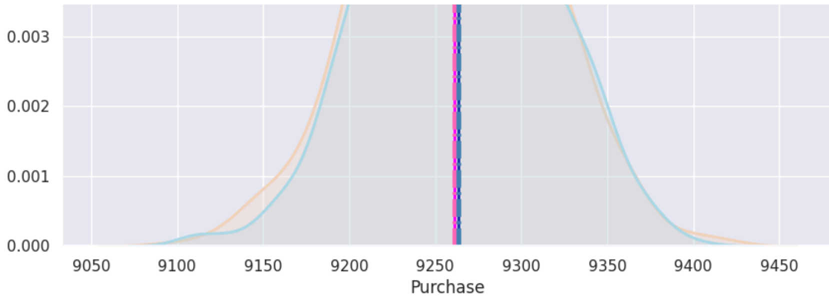
```
sample_sizes = sample_sizes = [10,100,1000,10000,100000]
ci = 90
itr_size = 1000

res1 = pd.DataFrame(columns = ['Marital Status','Sample Size','Lower Limit','Upper Limit','Sample Mean','Confidence Interval

for i in sample_sizes:
    m_avg, f_avg, ll_m, ul_m, ll_f, ul_f = sampling(df_married['Purchase'],df_unmarried['Purchase'],i,itr_size,ci)

    res1.loc[len(res1.index)] = {'Marital Status':'Married','Sample Size':i,'Lower Limit':ll_m,'Upper Limit':ul_m,'Sample Me
    res1.loc[len(res1.index)] = {'Marital Status':'Single','Sample Size':i,'Lower Limit':ll_f,'Upper Limit':ul_f,'Sample Mea
```

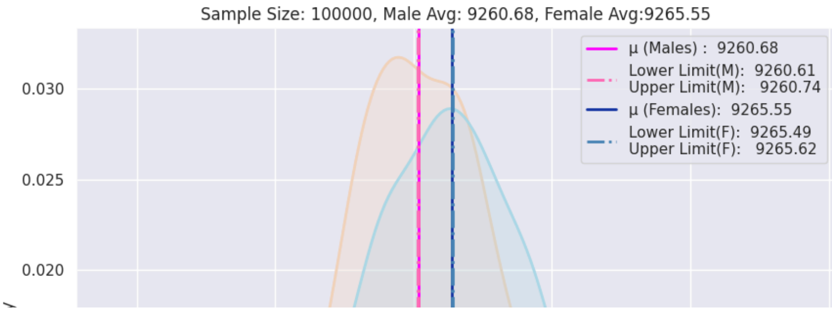

0.00010
0.00005
0.00000
4000 6000 8000 10000 12000 14000
Purchase


Sample Size: 100, Male Avg: 9296.77, Female Avg:9261.28
μ (Males) : 9296.77
Lower Limit(M): 9214.09
Upper Limit(M): 9379.44
μ (Females): 9261.28
Lower Limit(F): 9178.79
Upper Limit(F): 9343.77
Density
0.0000 0.0001 0.0002 0.0003 0.0004 0.0005 0.0006 0.0007 0.0008
8000 9000 10000 11000 12000
Purchase


Sample Size: 1000, Male Avg: 9256.97, Female Avg:9275.16
μ (Males) : 9256.97
Lower Limit(M): 9248.69
Upper Limit(M): 9265.25
μ (Females): 9275.16
Lower Limit(F): 9266.75
Upper Limit(F): 9283.58
Density
0.0000 0.0005 0.0010 0.0015 0.0020 0.0025
8600 8800 9000 9200 9400 9600 9800 10000
Purchase


Sample Size: 10000, Male Avg: 9261.18, Female Avg:9263.58
μ (Males) : 9261.18
Lower Limit(M): 9260.35
Upper Limit(M): 9262.01
μ (Females): 9263.58
Lower Limit(F): 9262.80
Upper Limit(F): 9264.36
Density
0.004 0.005 0.006 0.007 0.008

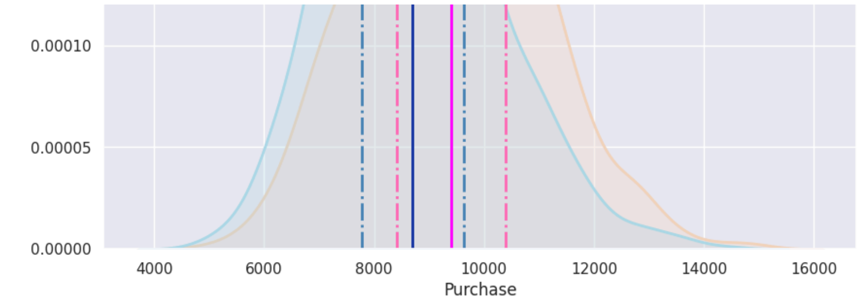

0.003
0.002
0.001
0.000
9050
9100
9150
9200
9250
9300
9350
9400
9450
Purchase


Sample Size: 100000, Male Avg: 9260.68, Female Avg:9265.55
μ (Males) : 9260.68
Lower Limit(M): 9260.61
Upper Limit(M): 9260.74
μ (Females): 9265.55
Lower Limit(F): 9265.49
Upper Limit(F): 9265.62
0.030
0.025
0.020

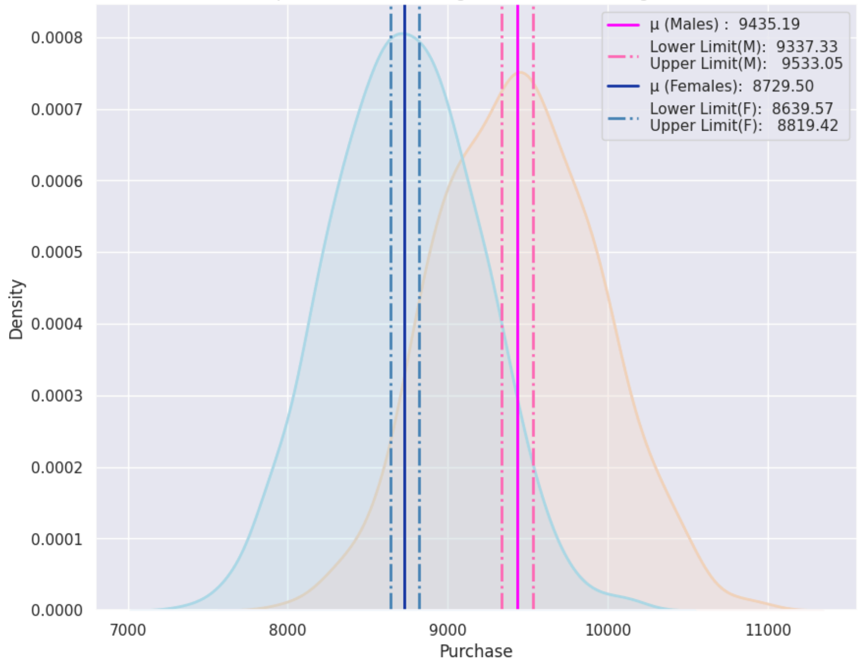
Lets plot the mean of 1000 Random Samples of sizes 10,100,1000,10000 and 100000 with 95% Confidence Interval

Double-click (or enter) to edit

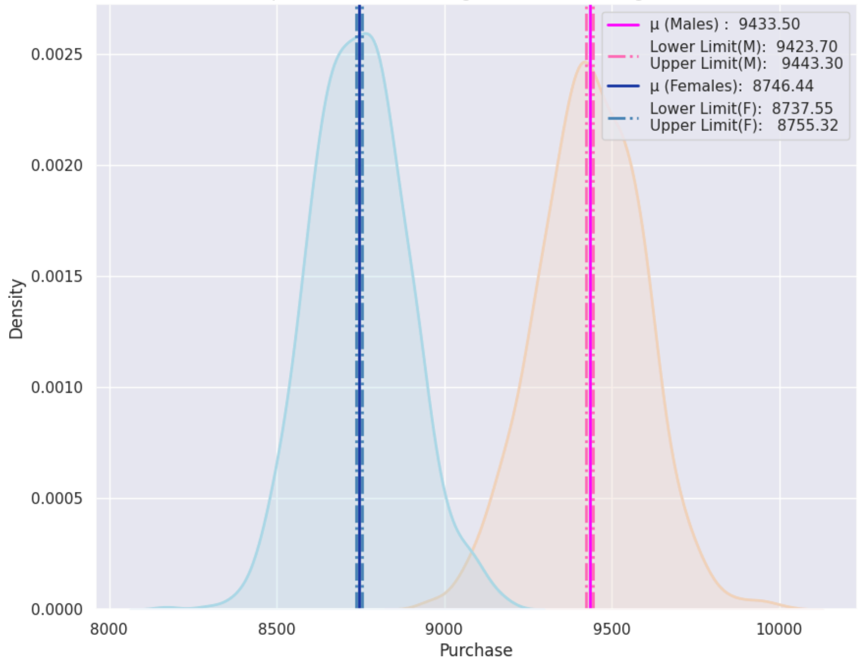
```
sample_sizes = sample_sizes = [10,100,1000,10000,100000]
ci = 95
itr_size = 1000

# res1 = pd.DataFrame(columns = ['Marital Status','Sample Size','Lower Limit','Upper Limit','Sample Mean','Confidence Interv

for i in sample_sizes:
    m_avg, f_avg, ll_m, ul_m, ll_f, ul_f = sampling(df_married['Purchase'],df_unmarried['Purchase'],i,itr_size,ci)

    res1.loc[len(res1.index)] = {'Marital Status':'Married','Sample Size':i,'Lower Limit':ll_m,'Upper Limit':ul_m,'Sample Me
    res1.loc[len(res1.index)] = {'Marital Status':'Single','Sample Size':i,'Lower Limit':ll_f,'Upper Limit':ul_f,'Sample Mea
```



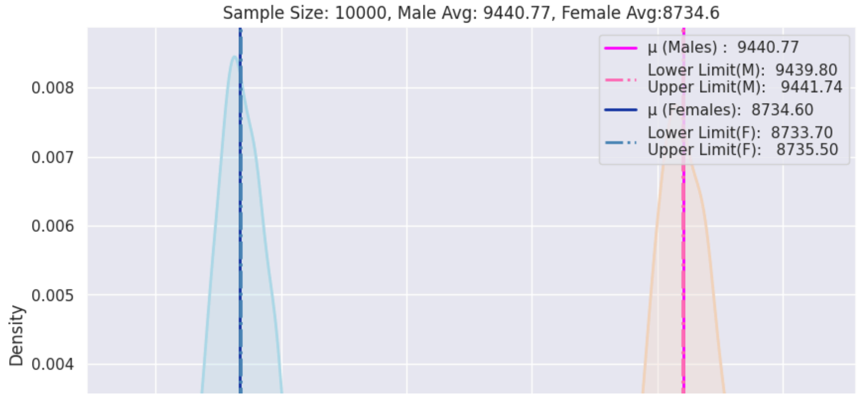
Sample Size: 100, Male Avg: 9435.19, Female Avg:8729.5

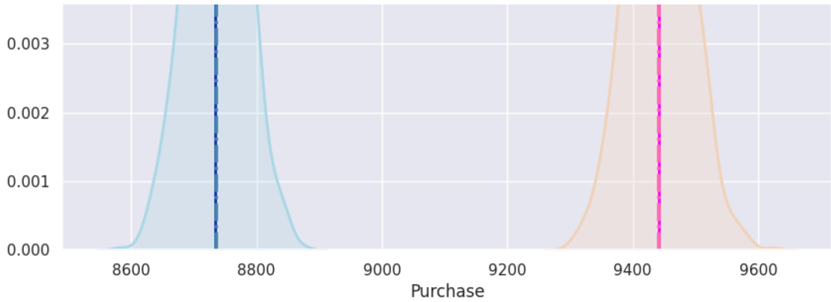


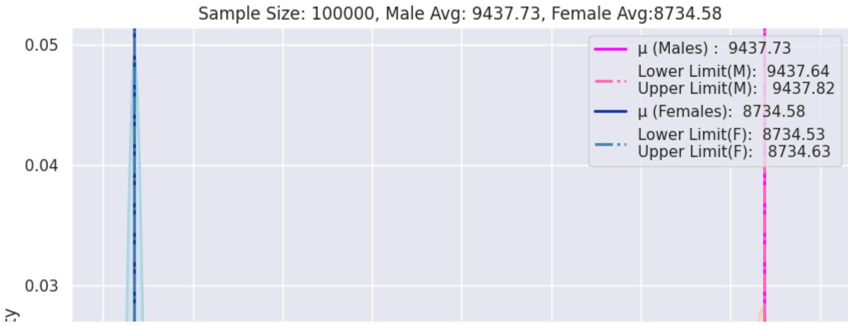
Sample Size: 1000, Male Avg: 9433.5, Female Avg:8746.44



Sample Size: 10000, Male Avg: 9440.77, Female Avg:8734.6




0.003
0.002
0.001
0.000
8600
8800
9000
9200
9400
9600
Purchase


Sample Size: 100000, Male Avg: 9437.73, Female Avg:8734.58
0.05
0.04
0.03
μ (Males) : 9437.73
Lower Limit(M): 9437.64
Upper Limit(M): 9437.82
μ (Females): 8734.58
Lower Limit(F): 8734.53
Upper Limit(F): 8734.63

Deep Dive into the confidence intervals of Married vs UnMarried

```
res1
```

For married and unmarried customers, sample size 10, confidence interval 90 we can observe that the interval range is overlapping

For married and unmarried customers, sample size 100000, confidence interval 90 we can observe that the interval range is still overlapping

This means there is no effect of marital status on purchase habits of customers

```
def sampling_age(sample, sample_size, itr_size, ci):
    ci = ci/100

    global flag

    sample_n = [np.mean(sample.sample(sample_size)) for i in range(itr_size)]

    mean  = np.mean(sample_n)
    sigma = np.std(sample_n)
    ci_arr= norm.interval(confidence=ci,loc=np.mean(sample_n),scale=np.std(sample_n)/np.sqrt(sample_size))
    lower_limit = ci_arr[0]
    upper_limit = ci_arr[1]

    fig, ax = plt.subplots(figsize=(14,6))
    sns.set_style("darkgrid")

    sns.kdeplot(data=sample_n,color="#7A68A6",fill=True,linewidth=2)

    label_mean=("μ :  {:.2f}".format(mean))
    label_ult=("Lower Limit:  {:.2f}\nUpper Limit:   {:.2f}".format(lower_limit,upper_limit))

    plt.title(f"Age Group: {age_group[flag]}, Sample Size: {sample_size}, Mean:{np.round(mean,2)}",fontsize=14)
    plt.xlabel('Purchase')
    plt.axvline(mean, color = 'y', linestyle = 'solid', linewidth = 2,label=label_mean)
    plt.axvline(upper_limit, color = 'r', linestyle = 'dotted', linewidth = 2,label=label_ult)
    plt.axvline(lower_limit, color = 'r', linestyle = 'dotted', linewidth = 2)
    plt.legend(loc='upper right')

    plt.show()
    flag += 1

    return sample_n ,np.round(lower_limit,2),np.round(upper_limit,2), round(mean,2)
```

Lets visualise the graphs of 1000 mean values of purchase samples for sample size of 1000 for all the age groups with 90% confidence interval.

Generate a slider using jupyter widgets Close

Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro

```
df.columns
```

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
```

```
           'Purchase'],
          dtype='object')
```

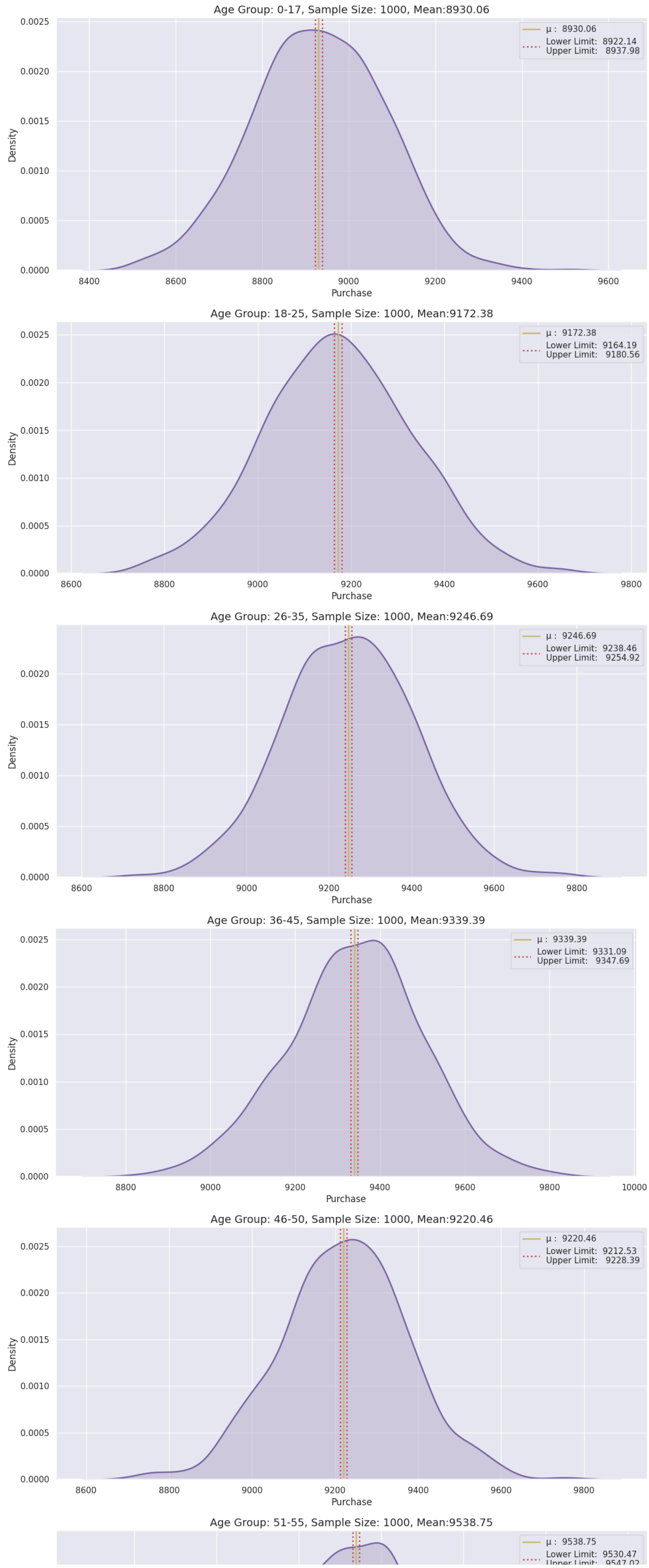
```
df['Age'].unique()
```

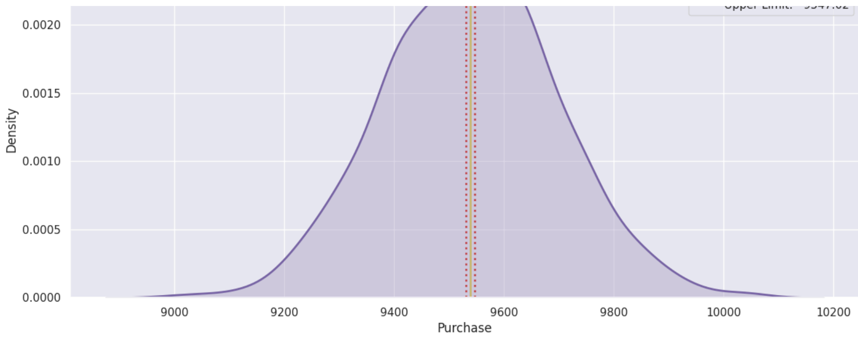
```
['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25']
Categories (7, object): ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
```

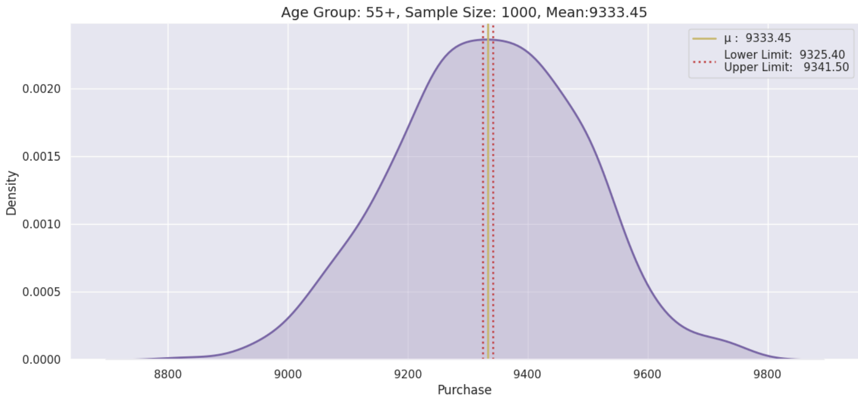
```
np.mean(df[df['Age']=='0-17']['Purchase'])
```

```
8933.464640444974
```

```
ci = 90
itr_size = 1000
sample_size = 1000
flag = 0
# global age_group
age_group = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

res2 = pd.DataFrame(columns = ['Age_Group','Sample Size','Lower Limit','Upper Limit','Sample Mean','Confidence Interval','Ir

for i in age_group:
    m_avg, ll, ul, mean = sampling_age(df[df['Age']==i]['Purchase'],sample_size,itr_size,ci)

    res2.loc[len(res2.index)] = {'Age_Group':i,'Sample Size':sample_size,'Lower Limit':ll,'Upper Limit':ul,'Sample Mean':mea
```


Age Group: 0-17, Sample Size: 1000, Mean:8930.06
μ : 8930.06
Lower Limit: 8922.14
Upper Limit: 8937.98
Density
Purchase
Age Group: 18-25, Sample Size: 1000, Mean:9172.38
μ : 9172.38
Lower Limit: 9164.19
Upper Limit: 9180.56
Density
Purchase
Age Group: 26-35, Sample Size: 1000, Mean:9246.69
μ : 9246.69
Lower Limit: 9238.46
Upper Limit: 9254.92
Density
Purchase
Age Group: 36-45, Sample Size: 1000, Mean:9339.39
μ : 9339.39
Lower Limit: 9331.09
Upper Limit: 9347.69
Density
Purchase
Age Group: 46-50, Sample Size: 1000, Mean:9220.46
μ : 9220.46
Lower Limit: 9212.53
Upper Limit: 9228.39
Density
Purchase
Age Group: 51-55, Sample Size: 1000, Mean:9538.75
μ : 9538.75
Lower Limit: 9530.47


Density
Purchase


Age Group: 55+, Sample Size: 1000, Mean:9333.45
μ : 9333.45
Lower Limit: 9325.40
Upper Limit: 9341.50
Density
Purchase

Lets visualise the graphs of 1000 mean values of purchase samples for sample size of 1000 for all the age groups with 95% confidence interval.

```
ci = 95
itr_size = 1000
sample_size = 1000
flag = 0
# global age_group
age_group = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

# res2 = pd.DataFrame(columns = ['Age_Group','Sample Size','Lower Limit','Upper Limit','Sample Mean','Confidence Interval','

for i in age_group:
    m_avg, ll, ul, mean = sampling_age(df[df['Age']==i]['Purchase'],sample_size,itr_size,ci)

    res2.loc[len(res2.index)] = {'Age_Group':i,'Sample Size':sample_size,'Lower Limit':ll,'Upper Limit':ul,'Sample Mean':mea
```

```
ci = 99
itr_size = 1000
sample_size = 1000
flag = 0
# global age_group
age_group = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

# res2 = pd.DataFrame(columns = ['Age_Group','Sample Size','Lower Limit','Upper Limit','Sample Mean','Confidence Interval','

for i in age_group:
    m_avg, ll, ul, mean = sampling_age(df[df['Age']==i]['Purchase'],sample_size,itr_size,ci)

    res2.loc[len(res2.index)] = {'Age_Group':i,'Sample Size':sample_size,'Lower Limit':ll,'Upper Limit':ul,'Sample Mean':mea
```