About Walmart

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

Business Problem

Management at Walmart Inc decided to analyze customer purchase behavior based on gender, age, and marital status in order to increase sales on Black Friday. This will enable management to make more informed decisions.

The company collected the transactional data of customers who purchased products from the Walmart Stores during Black Friday. The dataset has the following features:
Dataset link: Walmart_data.csv

```
User_ID: User ID
Product_ID: Product ID
Gender: Sex of User
Age: Age in bins
Occupation: Occupation(Masked)
City_Category: Category of the City (A,B,C)
StayInCurrentCityYears: Number of years stay in current city
Marital_Status: Marital Status
ProductCategory: Product Category (Masked)
Purchase: Purchase Amount
```

Importing the required libraries or packages for EDA

In [263…

```
#Importing packages
import numpy as np
import pandas as pd

# Importing matplotlib and seaborn for graphs
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style='whitegrid')

import warnings
warnings.filterwarnings('ignore')

from scipy import stats
from scipy.stats import kstest
import statsmodels.api as sm

# Importing Date & Time util modules
from dateutil.parser import parse
```

```
import statistics
from scipy.stats import norm
```

Utility Functions - Used during Analysis

Missing Value - Calculator

In [264…

```
def missingValue(df):
    #Identifying Missing data. Already verified above. To be sure again checking
    total_null = df.isnull().sum().sort_values(ascending = False)
    percent = ((df.isnull().sum()/df.isnull().count())*100).sort_values(ascendir
    print("Total records = ", df.shape[0])

    md = pd.concat([total_null,percent.round(2)],axis=1,keys=['Total Missing','I
    return md
```

Categorical Variable Analysis

Bar plot - Frequency of feature in percentage
Pie Chart

In [265…

```
# Frequency of each feature in percentage.
def cat_analysis(df, colnames, nrows=2,mcols=2,width=20,height=30, sortbyindex=F
    fig , ax = plt.subplots(nrows,mcols,figsize=(width,height))
    fig.set_facecolor(color = 'lightgrey')
    string = "Frequency of "
    rows = 0
    for colname in colnames:
        count = (df[colname].value_counts(normalize=True)*100)
        string += colname + ' in (%)'
        if sortbyindex:
                count = count.sort_index()
        count.plot.bar(color=sns.color_palette("Paired"),ax=ax[rows][0])
        ax[rows][0].set_ylabel(string, fontsize=14,family = "Comic Sans MS")
        ax[rows][0].set_xlabel(colname, fontsize=14,family = "Comic Sans MS")
        count.plot.pie(colors = sns.color_palette("Paired"),autopct='%0.0f%%',
                       textprops={'fontsize': 14,'family':"Comic Sans MS"},ax=ax
        string = "Frequency of "
        rows += 1
```

Frequency Graph in percentage

In [266…

```
# Frequency of each feature in percentage.
def bar_plot_percentage(df, colnames, sortbyindex=False):
    fig = plt.figure(figsize=(32, 36))
    fig.set_facecolor("lightgrey")
    string = "Frequency of "
    for colname in colnames:
        plt.subplot(5,2,colnames.index(colname)+1)
        count = (df[colname].value_counts(normalize=True)*100)
        string += colname + ' in (%)'
```

```
        if sortbyindex:
                count = count.sort_index()
        count.plot.bar(color=sns.color_palette('Paired'))
        plt.xticks(rotation = 70,fontsize=14,family="Comic Sans MS")
        plt.yticks(fontsize=14,family="Comic Sans MS")
        plt.ylabel(string, fontsize=14,family = "Comic Sans MS")
        plt.xlabel(colname, fontsize=14,family = "Comic Sans MS")
        string = "Frequency of "
```

BiVariate Analysis for Numerical and Categorical variables

In [267…

```
def num_cat_bi(df,col_cat,col_num,nrows=1,mcols=2,width=15,height=6):
    fig , ax = plt.subplots(nrows,mcols,figsize=(width,height),squeeze=False)
    sns.set(style='white')
    fig.set_facecolor("lightgrey")
    rows = 0
    i = 0
    while rows < nrows:
        sns.boxplot(x = col_cat[i],y = col_num, data = df,ax=ax[rows][0],palette
        ax[rows][0].set_xlabel(col_cat[i], fontweight="bold",fontsize=14,family
        ax[rows][0].set_ylabel(col_num,fontweight="bold", fontsize=14,family = "
        i += 1
        sns.boxplot(x = col_cat[i],y = col_num, data = df,ax=ax[rows][1],palette
        ax[rows][1].set_xlabel(col_cat[i], fontweight="bold",fontsize=14,family
        ax[rows][1].set_ylabel(col_num,fontweight="bold", fontsize=14,family = "
        i += 1
        rows += 1
    plt.show()
```

Distribution plot based on the Male and Female

In [268…

```
def bar_M_vs_F(colname):
    fig = plt.figure(figsize=(16,6))

    male = retail_data_v1[retail_data_v1["Gender"]=='M'][colname].value_counts()
    male["percentage"]  = (male[colname]*100/male[colname].sum())
    male["legends"]        = "Male"


    female = retail_data_v1[retail_data_v1["Gender"]=='F'][colname].value_counts
    female["percentage"] = (female[colname]*100/female[colname].sum())
    female["legends"]    = "Female"

    m_f_status = pd.concat([female,male],axis=0)

    ax = sns.barplot("index","percentage",data=m_f_status,hue="legends",palette=
    plt.xlabel(colname)
    fig.set_facecolor("white")
    plt.title(colname + "percentage in data with respect to churn status")
```

MultiVariate Analysis for Numerical and Categorical variables

In [269…

```
def num_cat_bi_grpby(df,colname,category,groupby,nrows=1,mcols=2,width=18,height
    fig , ax = plt.subplots(nrows,mcols,figsize=(width,height),squeeze=False)
    sns.set(style='white')
    fig.set_facecolor("lightgrey")
    rows = 0
    for var in colname:
        sns.boxplot(x = category,y = var,hue=groupby, data = df,ax=ax[rows][0],p
        sns.lineplot(x=df[category],y=df[var],ax=ax[rows][1],hue=df[groupby],pal
        ax[rows][0].set_ylabel(var, fontweight="bold",fontsize=14,family = "Comi
        ax[rows][0].set_xlabel(category,fontweight="bold", fontsize=14,family =
        ax[rows][0].legend(loc='lower right')
        ax[rows][1].set_ylabel(var, fontweight="bold",fontsize=14,family = "Comi
        ax[rows][1].set_xlabel(category,fontweight="bold", fontsize=14,family =
        rows += 1
    plt.show()
```

Function for Booststrapping technique to calculate the CI

In [270…

```
def bootstrapping(sample1,sample2,smp_siz=500,itr_size=5000,confidence_level=0.9

    smp1_means_m = np.empty(itr_size)
    smp2_means_m = np.empty(itr_size)
    for i in range(itr_size):
        smp1_n = np.empty(smp_siz)
        smp2_n = np.empty(smp_siz)
        smp1_n = np.random.choice(sample1, size = smp_siz,replace=True)
        smp2_n = np.random.choice(sample2, size = smp_siz,replace=True)
        smp1_means_m[i] = np.mean(smp1_n)
        smp2_means_m[i] = np.mean(smp2_n)

    #Calcualte the Z-Critical value
    alpha = (1 - confidence_level)/no_of_tails
    z_critical = stats.norm.ppf(1 - alpha)

    # Calculate the mean, standard deviation & standard Error of sampling distri
    mean1  = np.mean(smp1_means_m)
    sigma1 = statistics.stdev(smp1_means_m)
    sem1   = stats.sem(smp1_means_m)

    lower_limit1 = mean1 - (z_critical * sigma1)
    upper_limit1 = mean1 + (z_critical * sigma1)

    # Calculate the mean, standard deviation & standard Error of sampling distri
    mean2  = np.mean(smp2_means_m)
    sigma2 = statistics.stdev(smp2_means_m)
    sem2   = stats.sem(smp2_means_m)

    lower_limit2 = mean2 - (z_critical * sigma2)
    upper_limit2 = mean2 + (z_critical * sigma2)
```

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

    sns.kdeplot(data=smp1_means_m,color="#467821",fill=True,linewidth=2)
    sns.kdeplot(data=smp2_means_m,color='#e5ae38',fill=True,linewidth=2)

    label_mean1=("μ (Males) :  {:.2f}".format(mean1))
    label_ult1=("Lower Limit(M):  {:.2f}\nUpper Limit(M):   {:.2f}".format(lower
    label_mean2=("μ (Females):  {:.2f}".format(mean2))
    label_ult2=("Lower Limit(F):  {:.2f}\nUpper Limit(F):   {:.2f}".format(lower

    plt.title(f"Sample Size: {smp_siz}, Male Avg: {np.round(mean1, 2)}, Male SME
              fontsize=14,family = "Comic Sans MS")
    plt.xlabel('Purchase')
    plt.axvline(mean1, color = 'y', linestyle = 'solid', linewidth = 2,label=lab
    plt.axvline(upper_limit1, color = 'r', linestyle = 'solid', linewidth = 2,la
    plt.axvline(lower_limit1, color = 'r', linestyle = 'solid', linewidth = 2)
    plt.axvline(mean2, color = 'b', linestyle = 'dashdot', linewidth = 2,label=l
    plt.axvline(upper_limit2, color = '#56B4E9', linestyle = 'dashdot', linewidt
    plt.axvline(lower_limit2, color = '#56B4E9', linestyle = 'dashdot', linewidt
    plt.legend(loc='upper right')

    plt.show()

    return smp1_means_m,smp2_means_m ,np.round(lower_limit1,2),np.round(upper_li
```

In [271…

```
def bootstrapping_m_vs_um(sample1,sample2,smp_siz=500,itr_size=5000,confidence_l

    smp1_means_m = np.empty(itr_size)
    smp2_means_m = np.empty(itr_size)
    for i in range(itr_size):
        smp1_n = np.empty(smp_siz)
        smp2_n = np.empty(smp_siz)
        smp1_n = np.random.choice(sample1, size = smp_siz,replace=True)
        smp2_n = np.random.choice(sample2, size = smp_siz,replace=True)
        smp1_means_m[i] = np.mean(smp1_n)
        smp2_means_m[i] = np.mean(smp2_n)

#     std_dev1 = np.std(sample1)
#     std_err1 = np.std(sample1,ddof=1)/np.sqrt(smp_siz)
#     std_dev2 = np.std(sample2)
#     std_err2 = np.std(sample2,ddof=1)/np.sqrt(smp_siz)

    #Calcualte the Z-Critical value
    alpha = (1 - confidence_level)/no_of_tails
    z_critical = stats.norm.ppf(1 - alpha)

    # Calculate the mean, standard deviation & standard Error of sampling distri
    mean1  = np.mean(smp1_means_m)
    sigma1 = statistics.stdev(smp1_means_m)
    sem1   = stats.sem(smp1_means_m)

    lower_limit1 = mean1 - (z_critical * sigma1)
    upper_limit1 = mean1 + (z_critical * sigma1)

    # Calculate the mean, standard deviation & standard Error of sampling distri
    mean2  = np.mean(smp2_means_m)
    sigma2 = statistics.stdev(smp2_means_m)
    sem2   = stats.sem(smp2_means_m)
```

```
#     print(smp_siz,std_dev1,std_err1,sem1)
#     print(smp_siz,std_dev2,std_err2,sem2)

    lower_limit2 = mean2 - (z_critical * sigma2)
    upper_limit2 = mean2 + (z_critical * sigma2)

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

    sns.kdeplot(data=smp1_means_m,color="#467821",fill=True,linewidth=2)
    sns.kdeplot(data=smp2_means_m,color='#e5ae38',fill=True,linewidth=2)

    label_mean1=("μ (Married) :  {:.2f}".format(mean1))
    label_ult1=("Lower Limit(M):  {:.2f}\nUpper Limit(M):   {:.2f}".format(lower
    label_mean2=("μ (Unmarried):  {:.2f}".format(mean2))
    label_ult2=("Lower Limit(F):  {:.2f}\nUpper Limit(F):   {:.2f}".format(lower

    plt.title(f"Sample Size: {smp_siz}, Married Avg: {np.round(mean1, 2)}, Marri
              fontsize=14,family = "Comic Sans MS")
    plt.xlabel('Purchase')
    plt.axvline(mean1, color = 'y', linestyle = 'solid', linewidth = 2,label=lab
    plt.axvline(upper_limit1, color = 'r', linestyle = 'solid', linewidth = 2,la
    plt.axvline(lower_limit1, color = 'r', linestyle = 'solid', linewidth = 2)
    plt.axvline(mean2, color = 'b', linestyle = 'dashdot', linewidth = 2,label=l
    plt.axvline(upper_limit2, color = '#56B4E9', linestyle = 'dashdot', linewidt
    plt.axvline(lower_limit2, color = '#56B4E9', linestyle = 'dashdot', linewidt
    plt.legend(loc='upper right')

    plt.show()

    return smp1_means_m,smp2_means_m ,np.round(lower_limit1,2),np.round(upper_li
```

In [272…

```
def bootstrapping_age(sample,smp_siz=500,itr_size=5000,confidence_level=0.95,no_

    smp_means_m = np.empty(itr_size)
    for i in range(itr_size):
        smp_n = np.empty(smp_siz)
        smp_n = np.random.choice(sample, size = smp_siz,replace=True)
        smp_means_m[i] = np.mean(smp_n)

    #Calcualte the Z-Critical value
    alpha = (1 - confidence_level)/no_of_tails
    z_critical = stats.norm.ppf(1 - alpha)

    # Calculate the mean, standard deviation & standard Error of sampling distri
    mean  = np.mean(smp_means_m)
    sigma = statistics.stdev(smp_means_m)
    sem   = stats.sem(smp_means_m)

    lower_limit = mean - (z_critical * sigma)
    upper_limit = mean + (z_critical * sigma)

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

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

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

```
    plt.title(f"Sample Size: {smp_siz},Mean:{np.round(mean,2)}, SME:{np.round(se
    plt.xlabel('Purchase')
    plt.axvline(mean, color = 'y', linestyle = 'solid', linewidth = 2,label=labe
    plt.axvline(upper_limit, color = 'r', linestyle = 'solid', linewidth = 2,lab
    plt.axvline(lower_limit, color = 'r', linestyle = 'solid', linewidth = 2)
    plt.legend(loc='upper right')

    plt.show()

    return smp_means_m ,np.round(lower_limit,2),np.round(upper_limit,2)
```

Exploratory Data Analysis

Loading and inspecting the Dataset

Loading the csv file

In [273…
```
retail_data = pd.read_csv("original_walmart_data.csv")
df = pd.read_csv("original_walmart_data.csv")
```

In [274…
```
retail_data.head()
```

Out[274]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_
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	

Checking Shape and Column names

In [275…
```
retail_data.shape
```

Out[275]:
```
(550068, 10)
```

In [276…
```
retail_data.columns
```

Out[276]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

Validating Duplicate Records

In [277…
```
retail_data.duplicated().sum()
```

Out[277]:
```
0
```

Inference

No dupicates records found.

Missing Data Analysis

In [278…
```
missingValue(retail_data).head(5)
```

Total records = 550068

Out[278]:

	Total Missing	In Percent
User_ID	0	0.0
Product_ID	0	0.0
Gender	0	0.0
Age	0	0.0
Occupation	0	0.0

Inference

No missing value found.

Unique values (counts) for each Feature

In [279…
```
retail_data.nunique()
```

Out[279]:
```
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
```

Inference

The total number of records exceeds five lacks but the UserIDs are only 5891, meaning customers have visited multiple times in order to buy products.

A deep dive into User ID

Based on 5891 user IDs, how many are married, male or female, or the age of the users.

In [280…
```
retail_data.groupby(['Gender'])['User_ID'].nunique()
```

Out[280]:
```
Gender
F    1666
M    4225
Name: User_ID, dtype: int64
```

In [281…
```
print("Females are ", 1666/5891)
print("Females are ", 4225/5891)
```

```
Females are  0.2828042777117637
Females are  0.7171957222882362
```

Inference

The percentage of women customers is only 28%

Around 72% of customers are male

In [282…
```
retail_data.groupby(['Age'])['User_ID'].nunique()
```

Out[282]:
```
Age
0-17      218
18-25    1069
26-35    2053
36-45    1167
46-50     531
51-55     481
55+       372
Name: User_ID, dtype: int64
```

In [283…
```
retail_data.groupby(['City_Category'])['User_ID'].nunique()
```

Out[283]:
```
City_Category
A    1045
B    1707
C    3139
Name: User_ID, dtype: int64
```

In [284…
```
retail_data.groupby(['Stay_In_Current_City_Years'])['User_ID'].nunique()
```

Out[284]:
```
Stay_In_Current_City_Years
0      772
1     2086
2     1145
3      979
4+     909
Name: User_ID, dtype: int64
```

In [285…
```
retail_data.groupby(['Marital_Status'])['User_ID'].nunique()
```

Out[285]:
```
Marital_Status
0    3417
1    2474
Name: User_ID, dtype: int64
```

Unique values for each Features

In [286…
```
colname = ['Gender','Age','City_Category','Stay_In_Current_City_Years','Marital_
for col in colname:
    print("\nUnique values of ",col," are : ",list(retail_data[col].unique()))
```

```
Unique values of  Gender  are :  ['F', 'M']

Unique values of  Age  are :  ['0-17', '55+', '26-35', '46-50', '51-55', '36-45',
'18-25']

Unique values of  City_Category  are :  ['A', 'C', 'B']

Unique values of  Stay_In_Current_City_Years  are :  ['2', '4+', '3', '1', '0']

Unique values of  Marital_Status  are :  [0, 1]

Unique values of  Product_Category  are :  [3, 1, 12, 8, 5, 4, 2, 6, 14, 11, 13,
15, 7, 16, 18, 10, 17, 9, 20, 19]

Unique values of  Occupation  are :  [10, 16, 15, 7, 20, 9, 1, 12, 17, 0, 3, 4, 1
1, 8, 19, 2, 18, 5, 14, 13, 6]
```

Inferences

All the values looks good.

DataType Validation

In [287…
```
retail_data.info()
```

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

Inference

'User_ID','Product_ID','Gender',
'Age','City_Category','Marital_Status' are categorical variables. As a result, we need to change the datatype to category.

In [288…
```
cols = ['User_ID','Product_ID','Gender', 'Age','City_Category','Marital_Status']
for col_name in cols:
    retail_data[col_name] = retail_data[col_name].astype("category")
```

In [289…
```
retail_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     550068 non-null  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  int64
 5   City_Category               550068 non-null  category
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  category
 8   Product_Category            550068 non-null  int64
 9   Purchase                    550068 non-null  int64
dtypes: category(6), int64(3), object(1)
memory usage: 21.3+ MB
```

Basic Statistics Analysis - count, min, max, and mean

In [290…
```
retail_data.describe().T
```

Out[290]:

	count	mean	std	min	25%	50%	75%	
Occupation	550068.0	8.076707	6.522660	0.0	2.0	7.0	14.0	
Product_Category	550068.0	5.404270	3.936211	1.0	1.0	5.0	8.0	
Purchase	550068.0	9263.968713	5023.065394	12.0	5823.0	8047.0	12054.0	23

In [291…
```
retail_data.describe(include=['object','category']).T
```

Out[291]:

	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
City_Category	550068	3	B	231173
Stay_In_Current_City_Years	550068	5	1	193821
Marital_Status	550068	2	0	324731

In [292…
```
retail_data.groupby(['Gender'])['Purchase'].describe()
```

Out[292]:

Gender	count	mean	std	min	25%	50%	75%	max
F	135809.0	8734.565765	4767.233289	12.0	5433.0	7914.0	11400.0	23959.0
M	414259.0	9437.526040	5092.186210	12.0	5863.0	8098.0	12454.0	23961.0

In [293…
```
retail_data.groupby(['Marital_Status'])['Purchase'].describe()
```

Out[293]:

Marital_Status	count	mean	std	min	25%	50%	75%	ma
0	324731.0	9265.907619	5027.347859	12.0	5605.0	8044.0	12061.0	23961
1	225337.0	9261.174574	5016.897378	12.0	5843.0	8051.0	12042.0	23961

In [294…
```
retail_data.groupby(['Age'])['Purchase'].describe()
```

Out[294]:

Age	count	mean	std	min	25%	50%	75%	max
0-17	15102.0	8933.464640	5111.114046	12.0	5328.0	7986.0	11874.0	23955.0
18-25	99660.0	9169.663606	5034.321997	12.0	5415.0	8027.0	12028.0	23958.0
26-35	219587.0	9252.690633	5010.527303	12.0	5475.0	8030.0	12047.0	23961.0
36-45	110013.0	9331.350695	5022.923879	12.0	5876.0	8061.0	12107.0	23960.0
46-50	45701.0	9208.625697	4967.216367	12.0	5888.0	8036.0	11997.0	23960.0
51-55	38501.0	9534.808031	5087.368080	12.0	6017.0	8130.0	12462.0	23960.0
55+	21504.0	9336.280459	5011.493996	12.0	6018.0	8105.5	11932.0	23960.0

In [295…
```
retail_data.groupby(['City_Category'])['Purchase'].describe()
```

Out[295]:

City_Category	count	mean	std	min	25%	50%	75%	ma
A	147720.0	8911.939216	4892.115238	12.0	5403.0	7931.0	11786.0	23961.
B	231173.0	9151.300563	4955.496566	12.0	5460.0	8005.0	11986.0	23960.
C	171175.0	9719.920993	5189.465121	12.0	6031.5	8585.0	13197.0	23961.

In [296…
```
retail_data.groupby(['City_Category'])['User_ID'].nunique()
```

Out[296]:
```
City_Category
A    1045
B    1707
C    3139
Name: User_ID, dtype: int64
```

Inferences

There are more single people than married people.

Most mall customers are between the ages of 26 and 35.

The majority of our customers come from city category B but customers come from City category C spent more as mean is 9719.

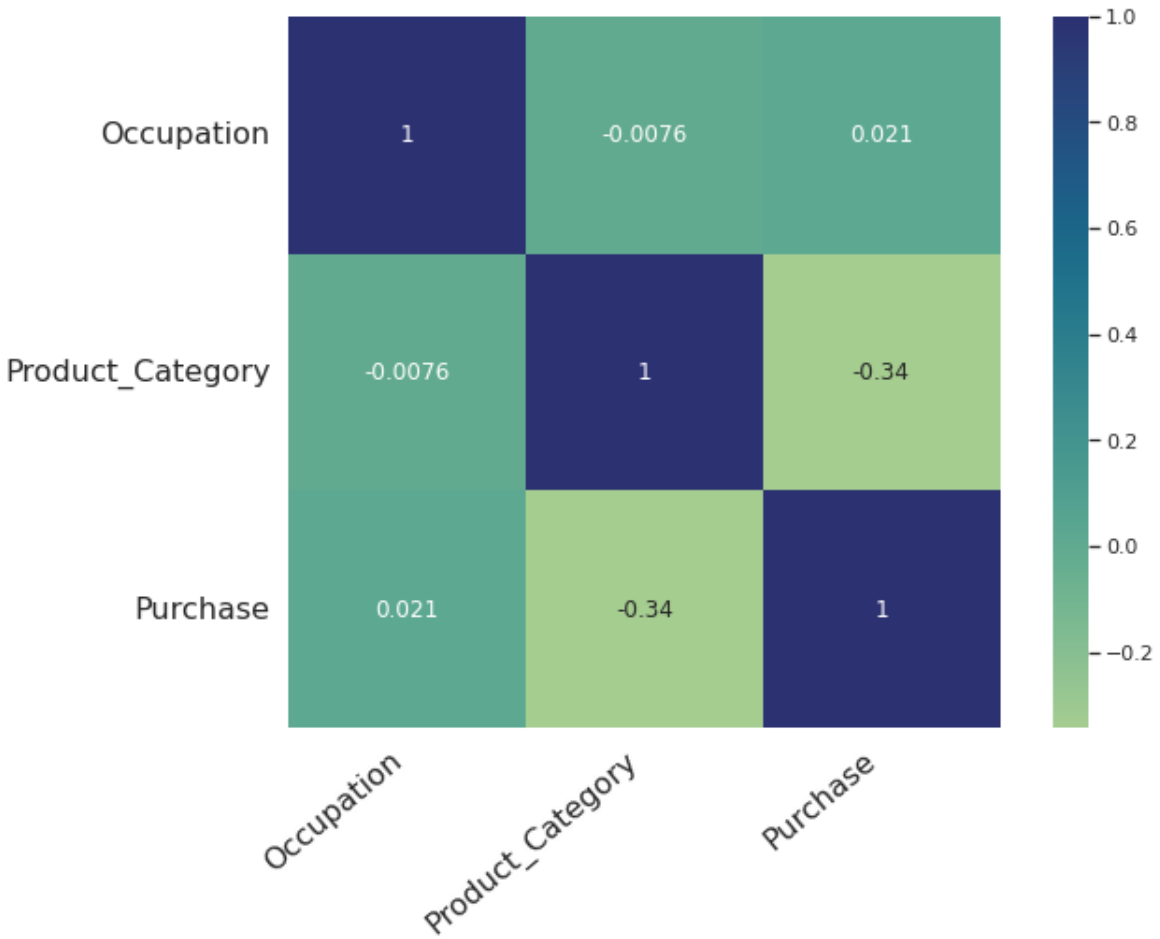
Male customers tend to spend more than female customers, as the mean is higher for male customers.

The majority of users come from City Category C, but more people from City Category B tend to purchase, which suggests the same users visit the mall multiple times in City Category B.

Correlation Analysis

In [297…

```
plt.figure(figsize = (10, 7))
ax = sns.heatmap(retail_data.corr(),
            annot=True,cmap='crest',square=True)

ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=40,fontsize=16,family = "Comic Sans MS",
    horizontalalignment='right')

ax.set_yticklabels(
    ax.get_yticklabels(),
    rotation=0,fontsize=16,family = "Comic Sans MS",
    horizontalalignment='right')

plt.show()
```

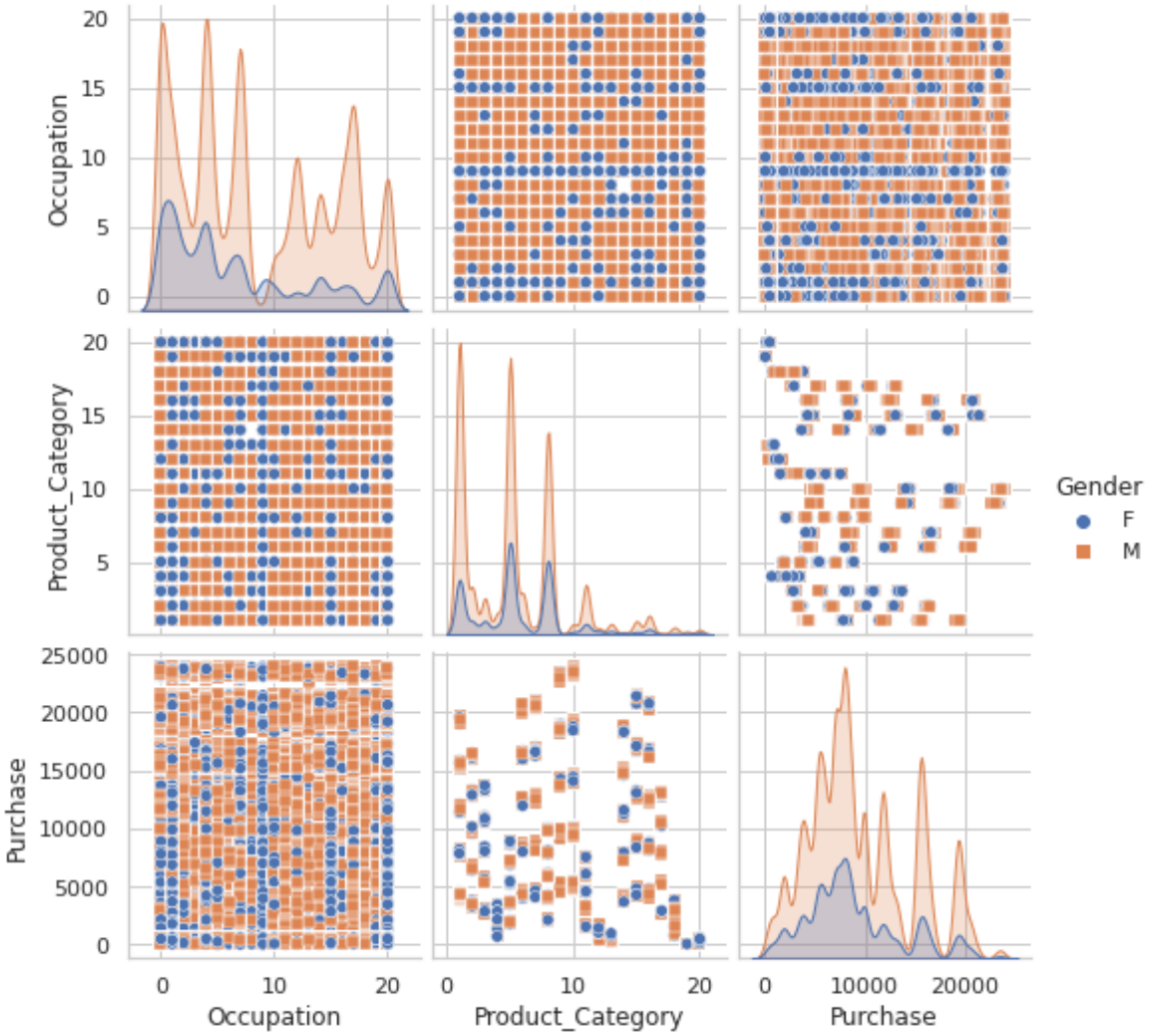


In [298…

```
sns.pairplot(retail_data, hue='Gender',markers=["o", "s"])
```

Out[298]:

```
<seaborn.axisgrid.PairGrid at 0x79cf222fc890>
```



Inference

Mostly features are categorical and not much correlation can be observed from above graphs.

UniVariate Analysis

Numerical Variables
Categorial variables

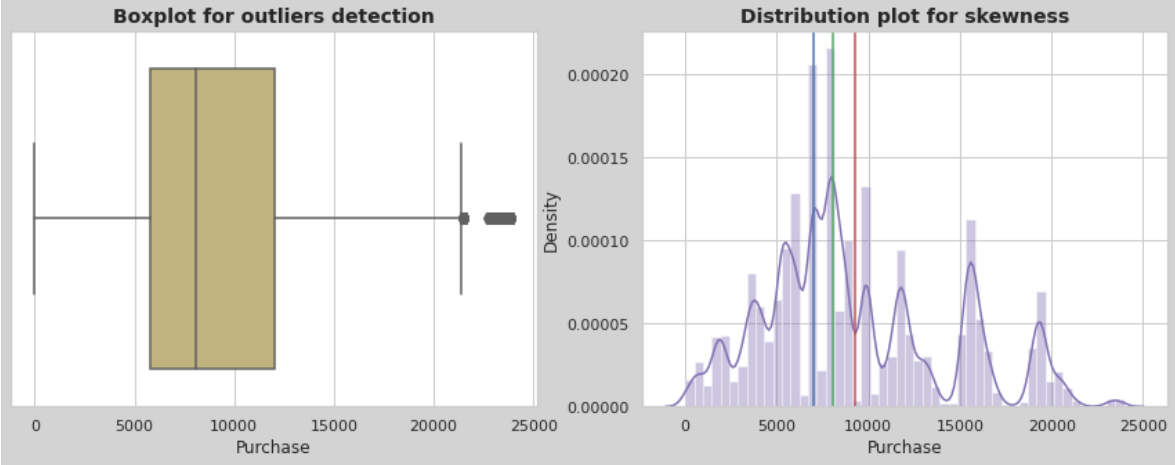
Numerical Variables - Outlier detection

In [299…
```
retail_data.columns
```

Out[299]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

In [300…
```
# Visualizing our dependent variable for Outliers and Skewness
fig = plt.figure(figsize=(15,5))
```

```
fig.set_facecolor("lightgrey")

plt.subplot(1,2,1)
sns.boxplot(retail_data["Purchase"],color='y')
plt.title("Boxplot for outliers detection", fontweight="bold",fontsize=14)
plt.xlabel('Purchase', fontsize=12,family = "Comic Sans MS")

plt.subplot(1,2,2)
sns.distplot(retail_data["Purchase"],color='m')

plt.title("Distribution plot for skewness", fontweight="bold",fontsize=14)
plt.ylabel('Density', fontsize=12,family = "Comic Sans MS")
plt.xlabel('Purchase', fontsize=12,family = "Comic Sans MS")
plt.axvline(retail_data["Purchase"].mean(),color="r")
plt.axvline(retail_data["Purchase"].median(),color="g")
plt.axvline(retail_data["Purchase"].mode()[0],color="b")

plt.show()
```



Inferences

Above graphs ;ooks like "right-skewed distribution" which means the mass of the distribution is concentrated on the left of the figure.

Majority of Customers purchase within the 5,000 - 20,000 range.

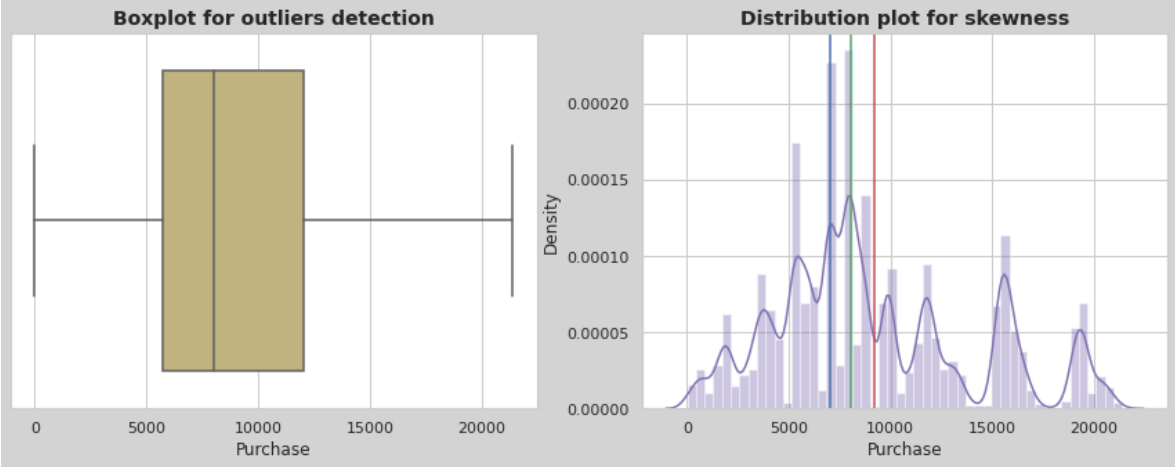
Handling outliers

In [301…
```
retail_data_v1 = retail_data.copy()
```

In [302…
```
#Outlier Treatment: Remove top 5% & bottom 1% of the Column Outlier values
Q3 = retail_data_v1['Purchase'].quantile(0.75)
Q1 = retail_data_v1['Purchase'].quantile(0.25)
IQR = Q3-Q1
retail_data_v1 = retail_data_v1[(retail_data_v1['Purchase'] > Q1 - 1.5*IQR) & (r
```

In [303…
```
# Visualizing our dependent variable for Outliers and Skewness
fig = plt.figure(figsize=(15,5))
fig.set_facecolor("lightgrey")
```

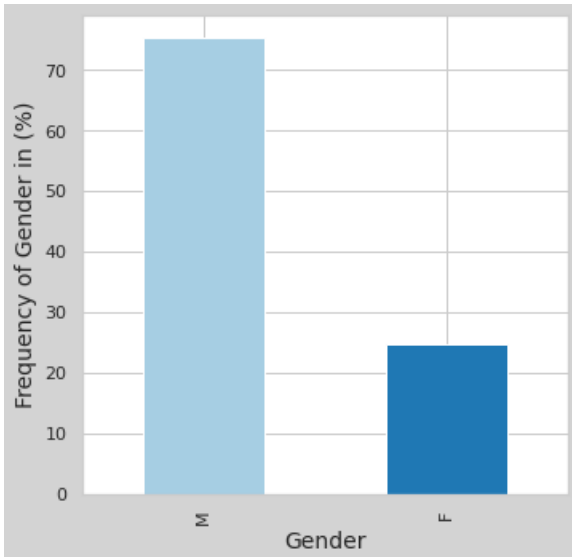
```
plt.subplot(1,2,1)
sns.boxplot(retail_data_v1["Purchase"],color='y')
plt.title("Boxplot for outliers detection", fontweight="bold",fontsize=14)
plt.xlabel('Purchase', fontsize=12,family = "Comic Sans MS")

plt.subplot(1,2,2)
sns.distplot(retail_data_v1["Purchase"],color='m')

plt.title("Distribution plot for skewness", fontweight="bold",fontsize=14)
plt.ylabel('Density', fontsize=12,family = "Comic Sans MS")
plt.xlabel('Purchase', fontsize=12,family = "Comic Sans MS")
plt.axvline(retail_data_v1["Purchase"].mean(),color="r")
plt.axvline(retail_data_v1["Purchase"].median(),color="g")
plt.axvline(retail_data_v1["Purchase"].mode()[0],color="b")

plt.show()
```

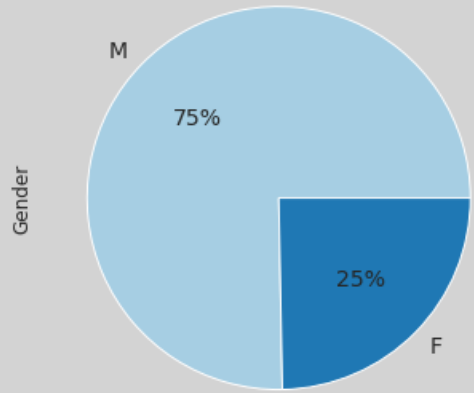


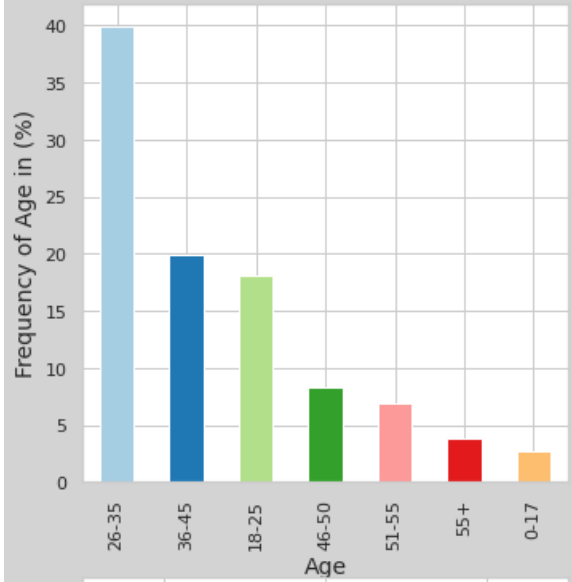
Categorical variable UniVariate Analysis

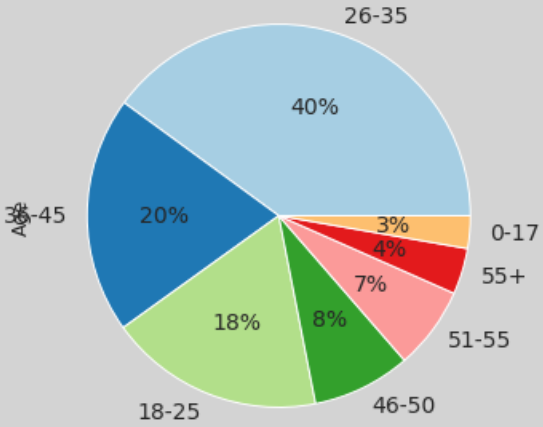
In [304…

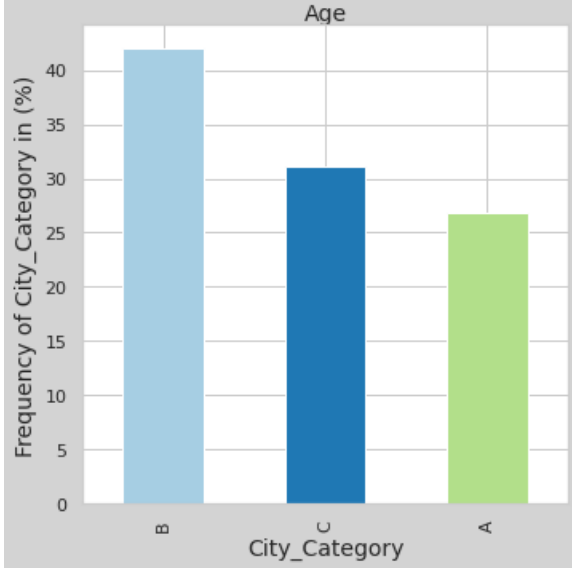
```
cat_colnames = ['Gender','Age','City_Category','Stay_In_Current_City_Years','Mar
cat_analysis(retail_data_v1,cat_colnames,5,2,12,32)
```

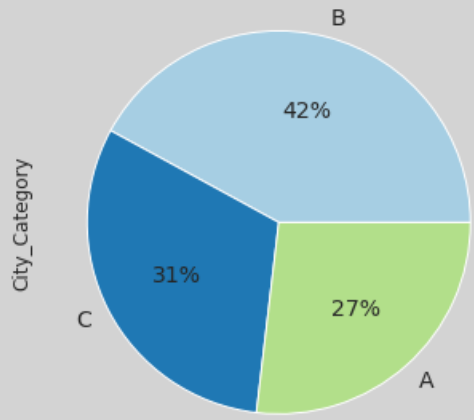

Frequency of Gender in (%)
70
60
50
40
30
20
10
0
M
F
Gender

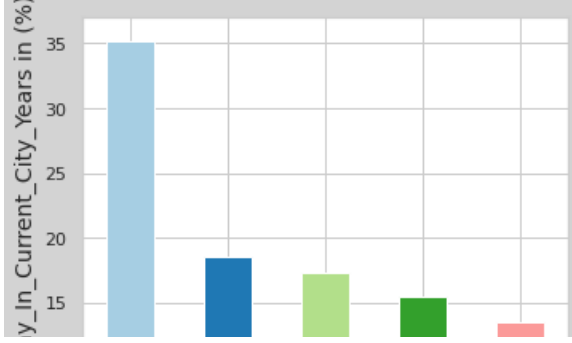

M
75%
25%
F
Gender

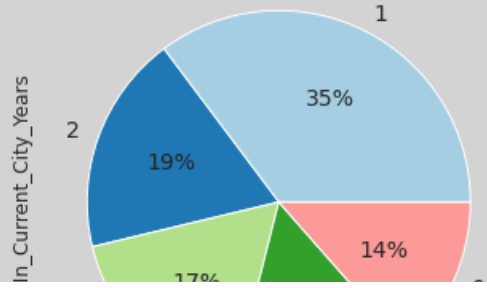

Frequency of Age in (%)
40
35
30
25
20
15
10
5
0
26-35
36-45
18-25
46-50
51-55
55+
0-17
Age

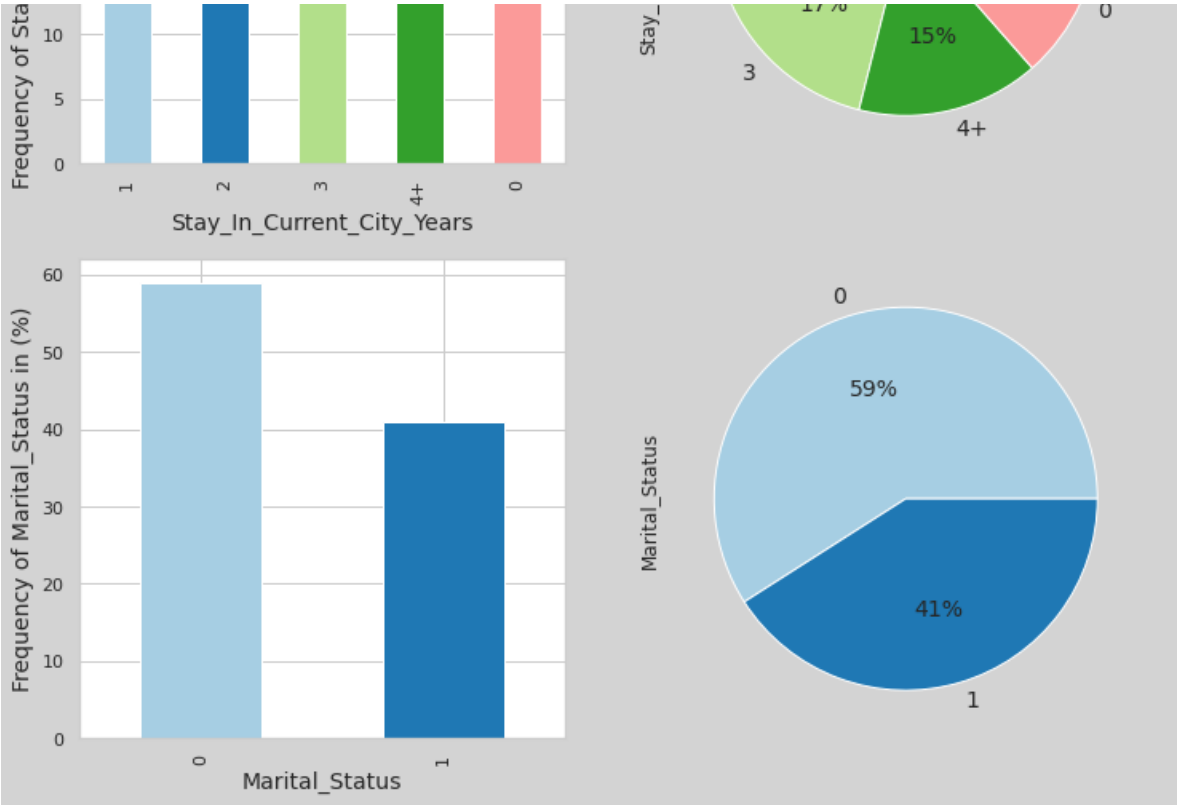

26-35
40%
36-45
20%
18-25
18%
46-50
8%
51-55
7%
55+
4%
0-17
3%


Frequency of City_Category in (%)
40
35
30
25
20
15
10
5
0
B
C
A
City_Category


B
42%
C
31%
A
27%
City_Category


35
30
25
20
15


1
35%
2
19%
14%



Inferences

Males clearly purchase more than females. 75% of men and only 25% of women purchase products.

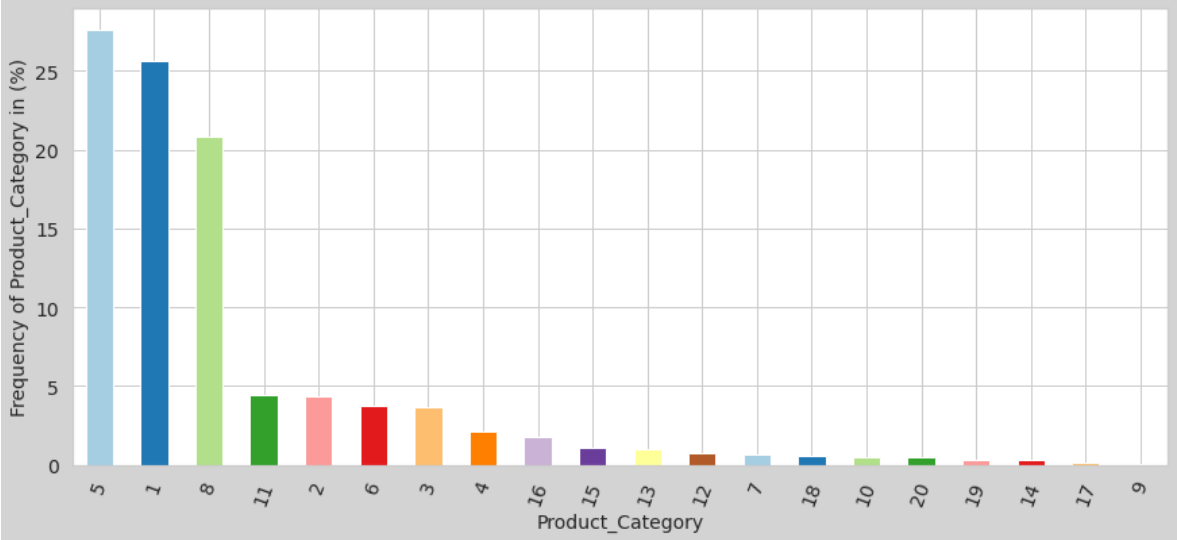
60% of purchases are made by people between the ages of 26 and 45

City Category B accounts for 42%, City Category C 31%, and City Category A represents 27% of all customer purchases.

In [305…
```
retail_data.columns
```

Out[305]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

In [306…
```
bar_plot_percentage(retail_data_v1,['Product_Category'])
```



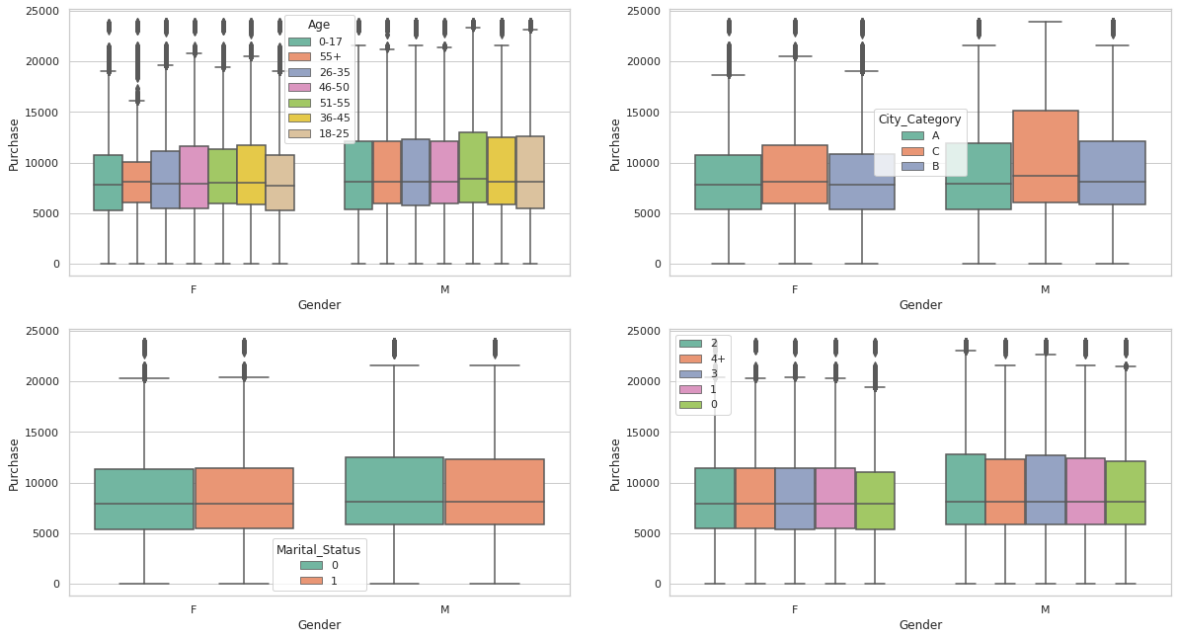
Inference

Product Category 5,1 & 8 are the products that customers buy the most.

MultiVariate Analysis

In [307…

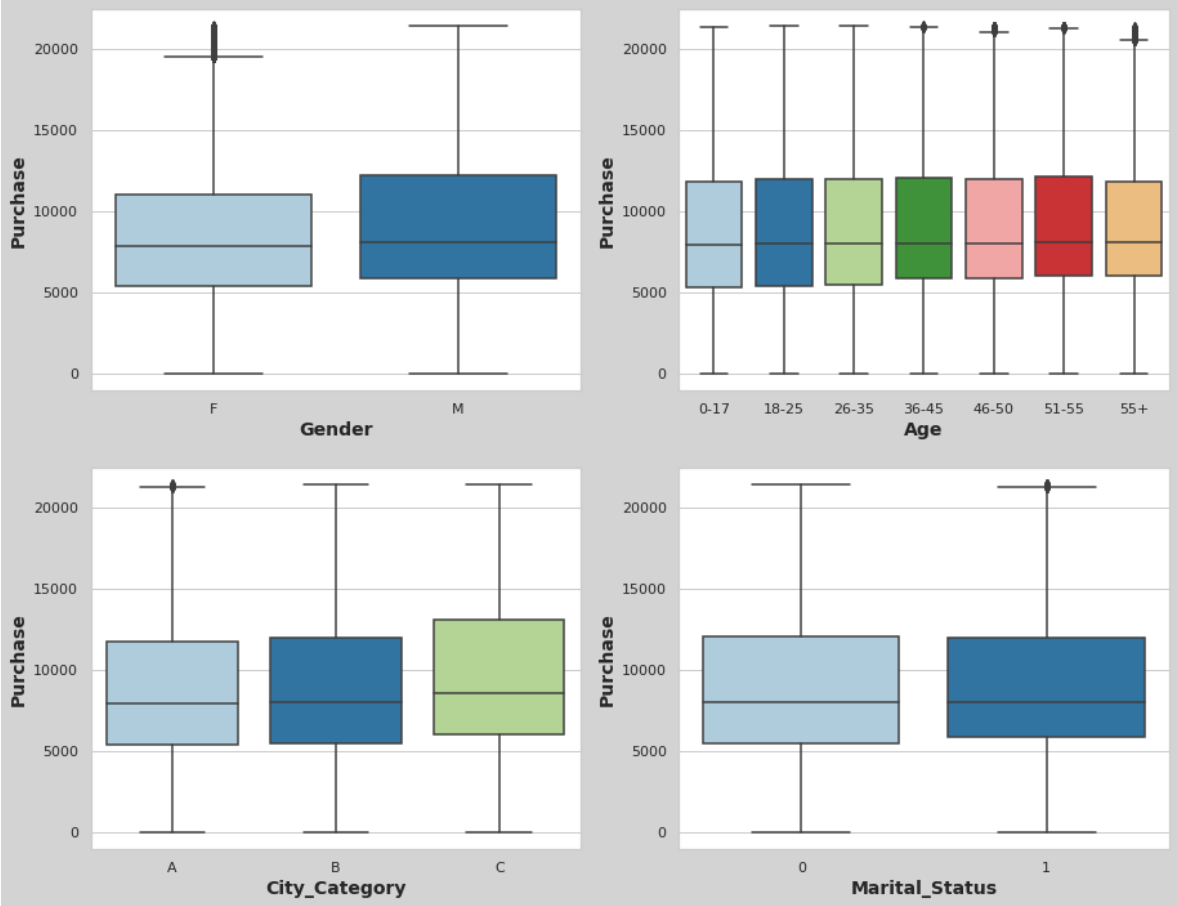
```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20, 6))
fig.subplots_adjust(top=1.5)
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Age', palette='Set2', ax=axs
sns.boxplot(data=df, y='Purchase', x='Gender', hue='City_Category', palette='Set

sns.boxplot(data=df, y='Purchase', x='Gender', hue='Marital_Status', palette='Se
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Stay_In_Current_City_Years',
axs[1,1].legend(loc='upper left')

plt.show()
```



In [308…

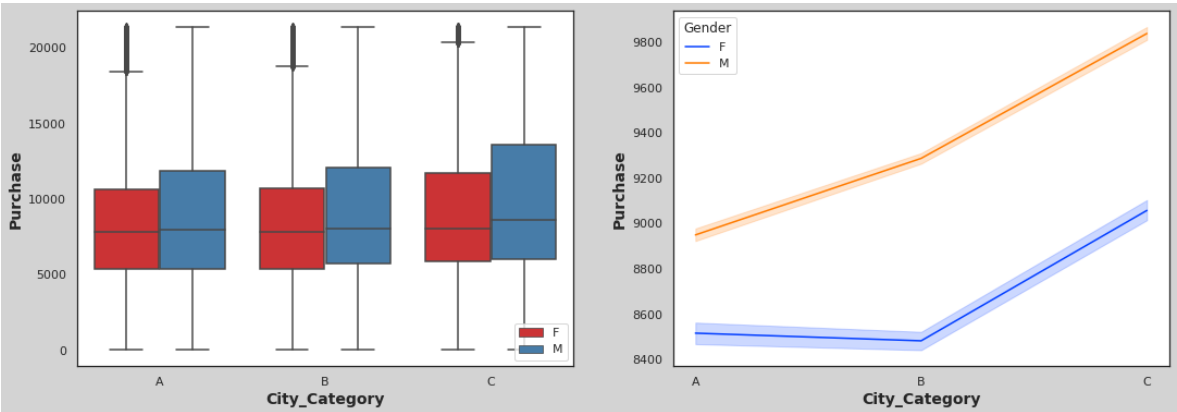
```
col_cat = ['Gender', 'Age','City_Category','Marital_Status']
```

```
num_cat_bi(retail_data_v1,col_cat,'Purchase',2,2,15,12)
```



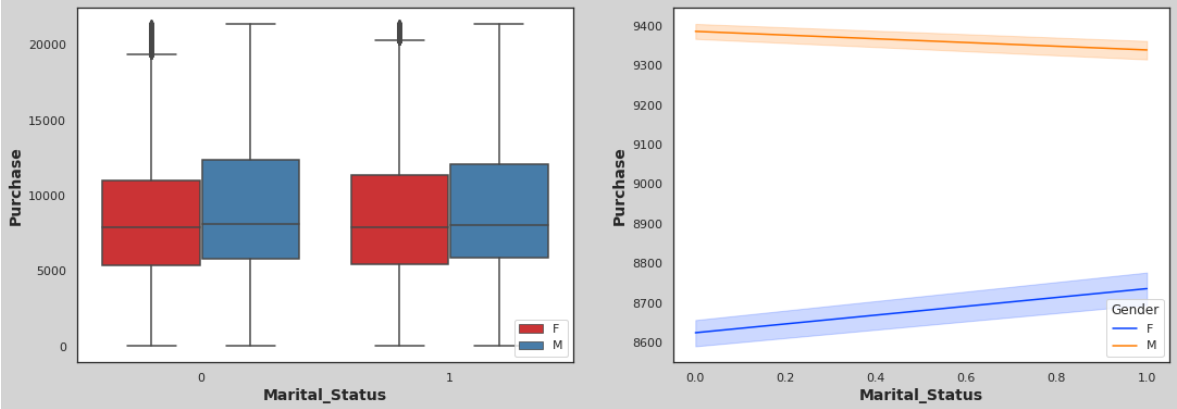
In [309…

```
col_num = [ 'Purchase']
num_cat_bi_grpby(retail_data_v1,col_num,"City_Category",'Gender')
```

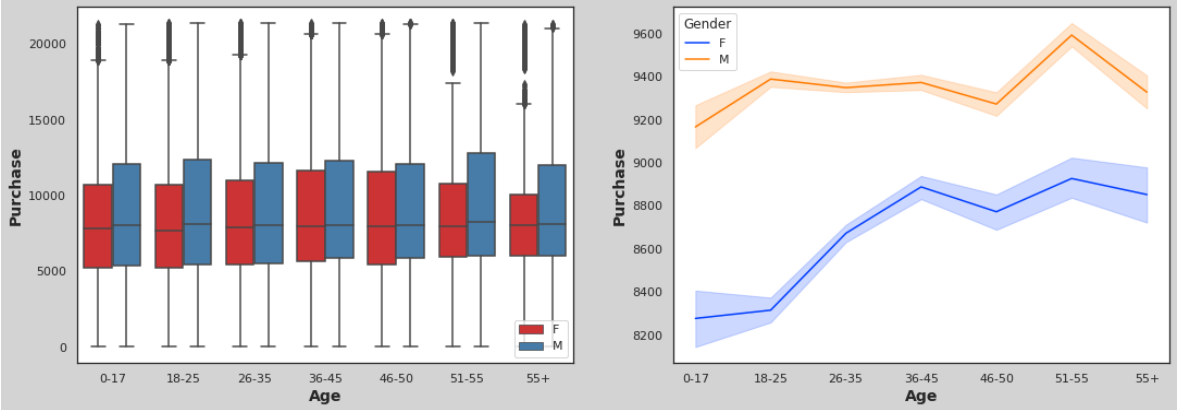


In [310…

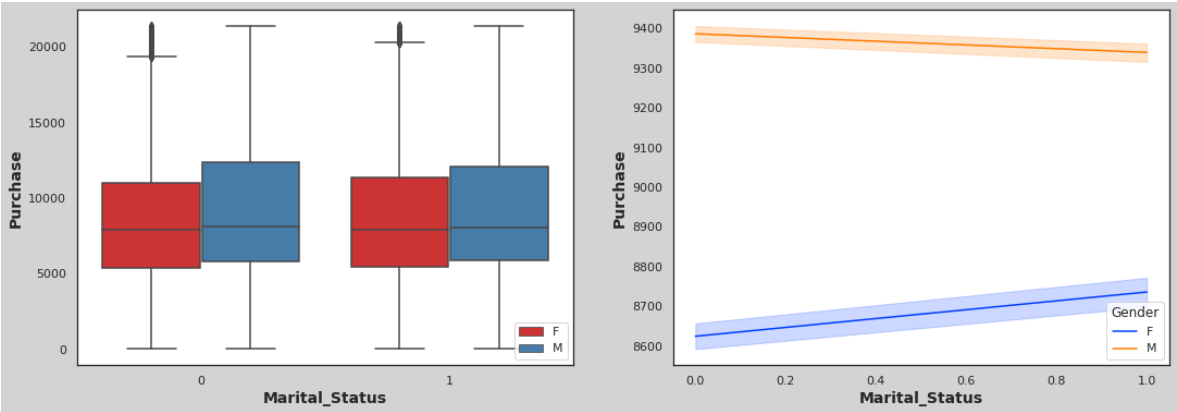
```
col_num = [ 'Purchase']
num_cat_bi_grpby(retail_data_v1,col_num,"Marital_Status",'Gender')
```



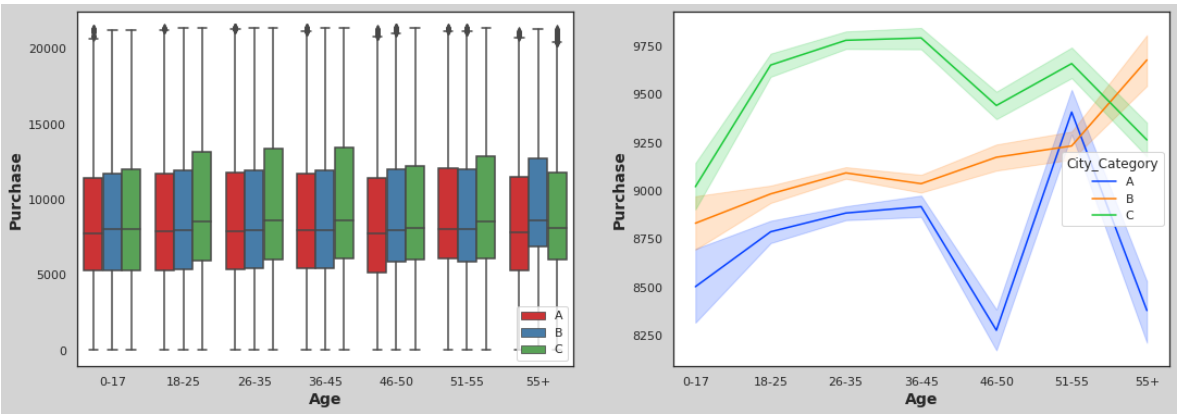
In [311…
```
col_num = [ 'Purchase']
num_cat_bi_grpby(retail_data_v1,col_num,"Age",'Gender')
```



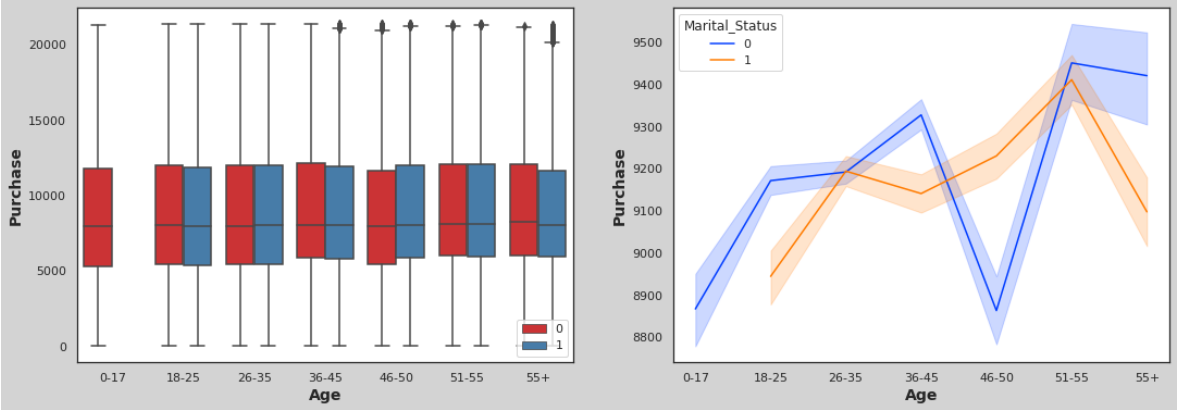
In [312…
```
col_num = [ 'Purchase']
num_cat_bi_grpby(retail_data_v1,col_num,"Marital_Status",'Gender')
```



In [313…
```
col_num = [ 'Purchase']
num_cat_bi_grpby(retail_data_v1,col_num,"Age",'City_Category')
```



In [314...
```
col_num = [ 'Purchase']
num_cat_bi_grpby(retail_data_v1,col_num,"Age",'Marital_Status')
```



Inferences

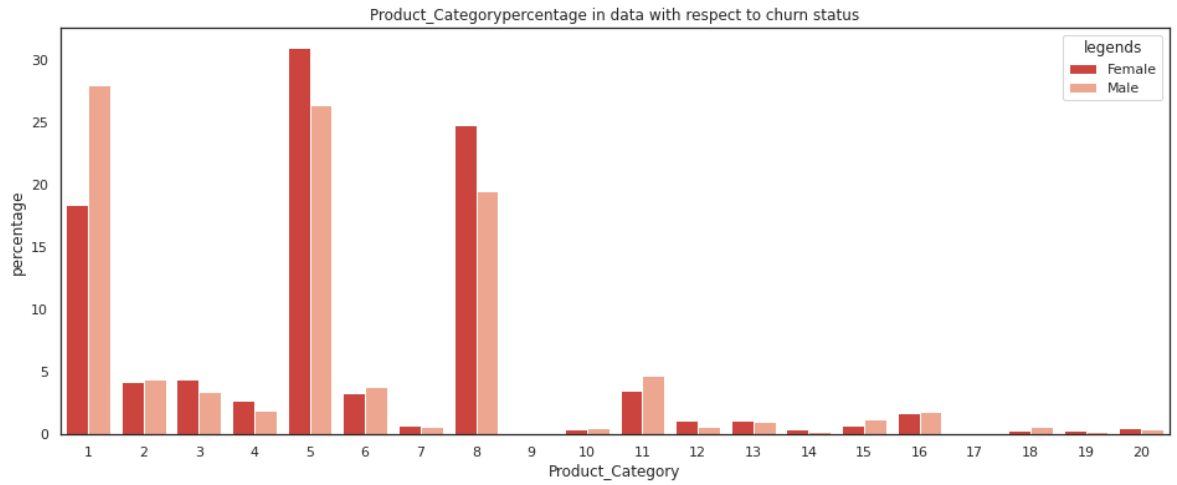
Purchases are high in city category C

Purchase is the same for all age groups

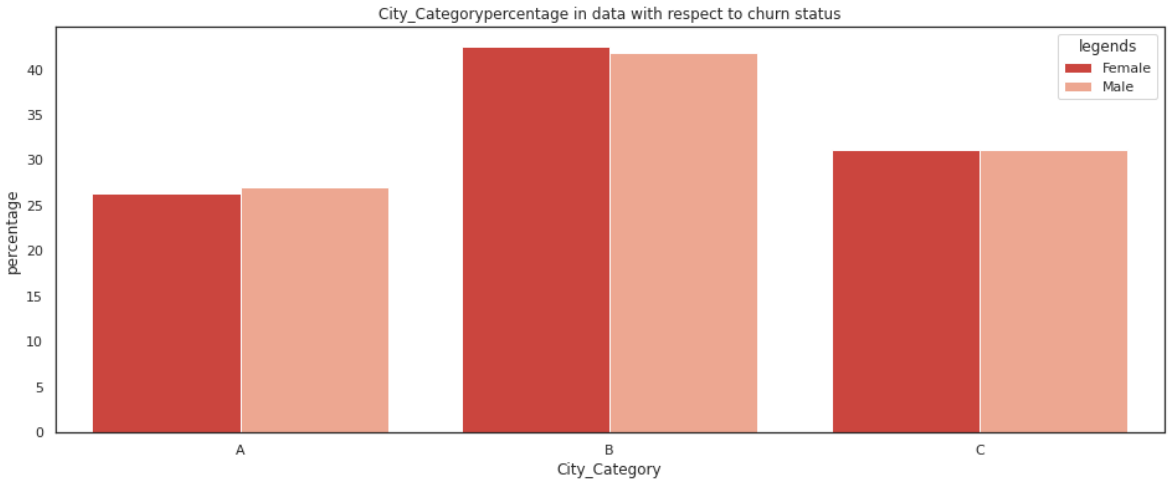
Most of the customers are 55+ and live in city category B

City category C has more customers between the ages of 18 and 45.

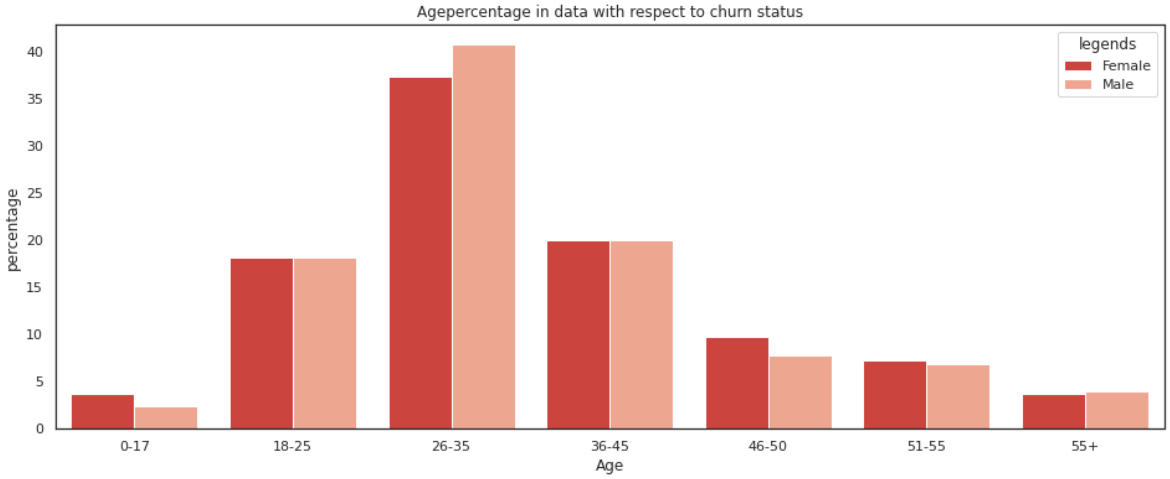
In [315...
```
bar_M_vs_F('Product_Category')
```



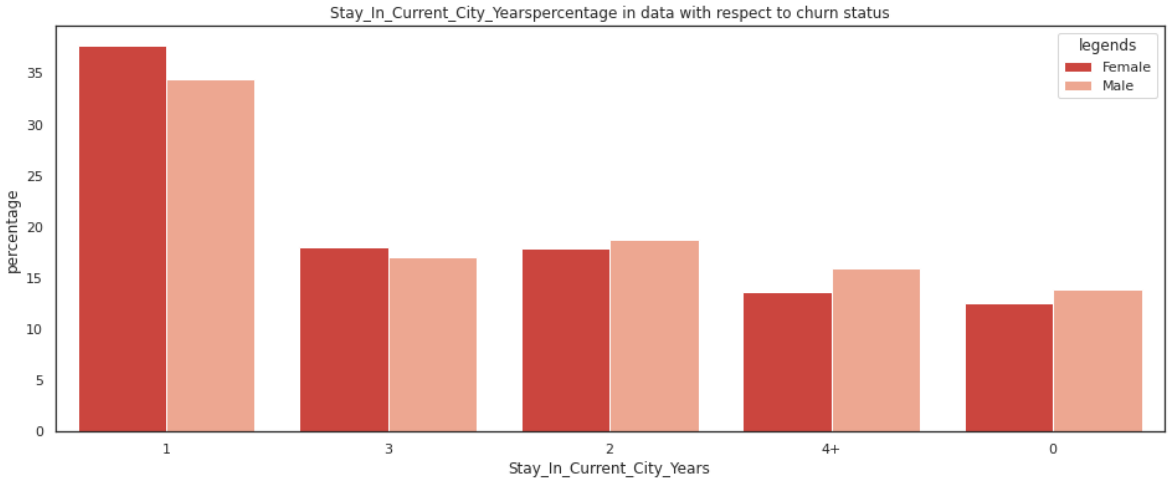
In [316...
```
bar_M_vs_F('City_Category')
```



In [317…
```
bar_M_vs_F('Age')
```



In [318…
```
bar_M_vs_F('Stay_In_Current_City_Years')
```



Inferences

Product 5 and 8 is common among females.

In City Category C, there are slightly more female customers.

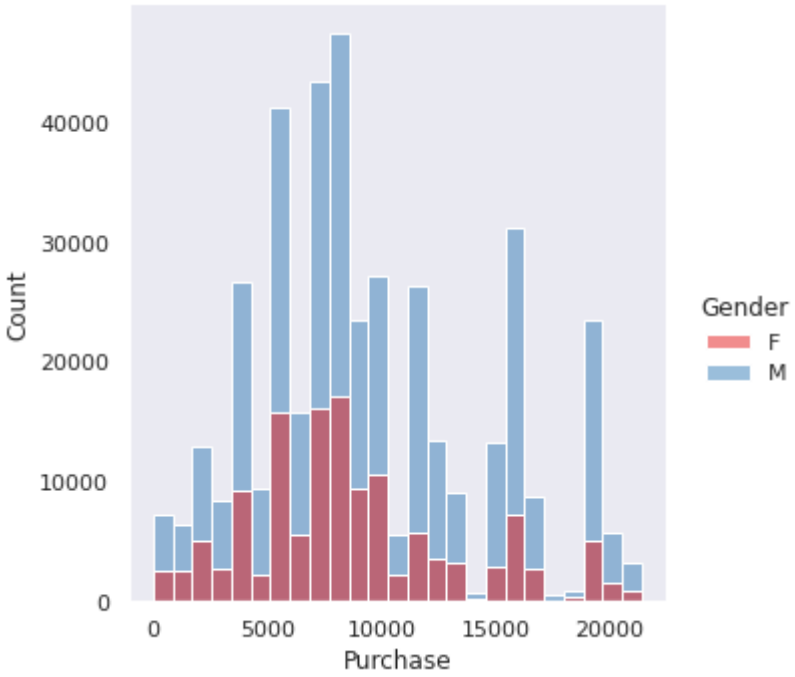
In [319…
```
print(retail_data_v1.groupby(['Gender','City_Category'])['User_ID'].count())
```

```
Gender  City_Category
F       A                35552
        B                57572
        C                42096
M       A               111484
        B               172542
        C               128145
Name: User_ID, dtype: int64
```

In [320…

```
fig = plt.figure(figsize=(25,10))
fig.set_facecolor("white")
sns.set(style='dark')
sns.displot(x= 'Purchase',data=retail_data_v1,hue='Gender',bins=25,palette="Set1
plt.show()
```

```
<Figure size 1800x720 with 0 Axes>
```



Inference

The amount of money spent by women is less than that spent by men

In [321…

```
retail_data_v1.sample(500,replace=True).groupby(['Gender'])['Purchase'].describe
```

Out[321]:

Gender	count	mean	std	min	25%	50%	75%	max
F	138.0	8501.202899	4547.516544	579.0	5399.5	7823.5	10601.25	20646.0
M	362.0	9177.461326	4961.104335	24.0	5467.5	8045.0	11954.75	20658.0

Inference

Even the sample mean shows that males spend more than females.

In [322…
```
retail_data_v1.groupby(['Gender'])['Purchase'].describe()
```

Out[322]:

Gender	count	mean	std	min	25%	50%	75%	max
F	135220.0	8671.049039	4679.058483	12.0	5429.0	7906.0	11064.0	21398.0
M	412171.0	9367.724355	5009.234088	12.0	5852.0	8089.0	12247.0	21399.0

In [323…
```
retail_data_v1.shape
```

Out[323]: (547391, 10)

Inference

Given the sample size of 5.4 Million data for customer purhase history with 1.3M Females and 4.1 Males

In [324…
```
retail_data_smp_male = retail_data_v1[retail_data_v1['Gender'] == 'M']['Purchase
retail_data_smp_female = retail_data_v1[retail_data_v1['Gender'] == 'F']['Purcha
```

In [325…
```
print("Male Customers : ",retail_data_smp_male.shape[0])
print("Female Customers : ",retail_data_smp_female.shape[0])
```

```
Male Customers :  412171
Female Customers :  135220
```

Central limit Theorem

The central limit theorem states that the sampling distribution of a sample mean is approximately normal if the sample size is large enough, even if the population distribution is not normal.

Calculate CI using Bootstrapping

We will be using Bootstrapping method to estimate the confidence interval of the population mean of the expenses by female and Male customers.

Bootstrapping

Bootstrapping is a method that can be used to estimate the standard error of any statistic and produce a confidence interval for the statistic.

The basic process for bootstrapping is as follows:

> Take k repeated samples with replacement from a given dataset.
> For each sample, calculate the statistic you're interested in.
> This results in k different estimates for a given statistic,
> which you can then use to calculate the standard error of the
> statistic and create a confidence interval for the statistic.

Calculate z-critical

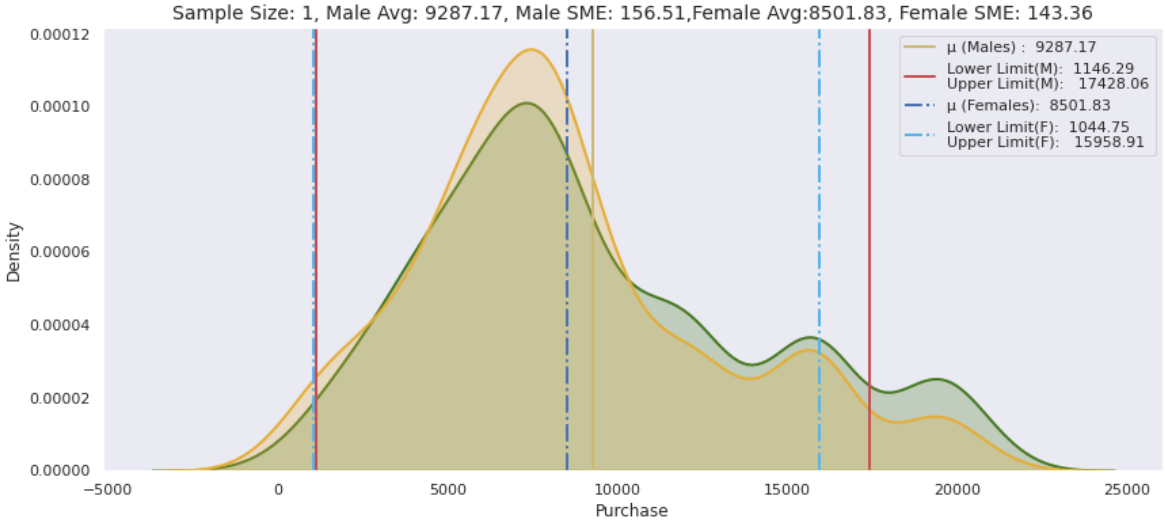
> Calculate the significance level : alpha = (1 - confidence
> interval)/ number of tails
>
> Z-Table lookup value is (1 - α)
>
> Find the y-axis value, then the x-axis value
>
> Critical value = y-axis value + x-axis value

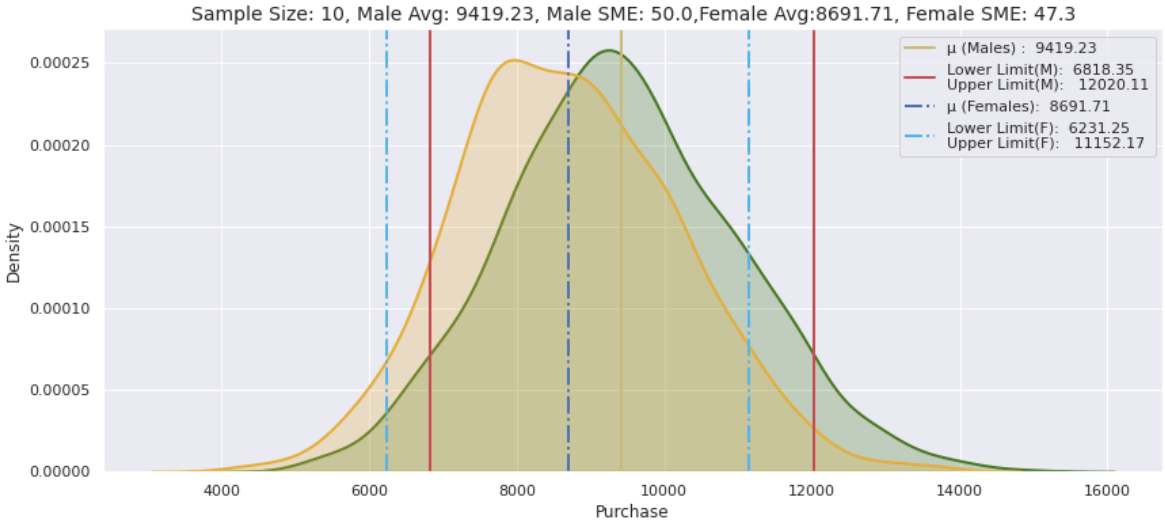
CLT Analysis for mean purchase with confidence 90% - Based on Gender

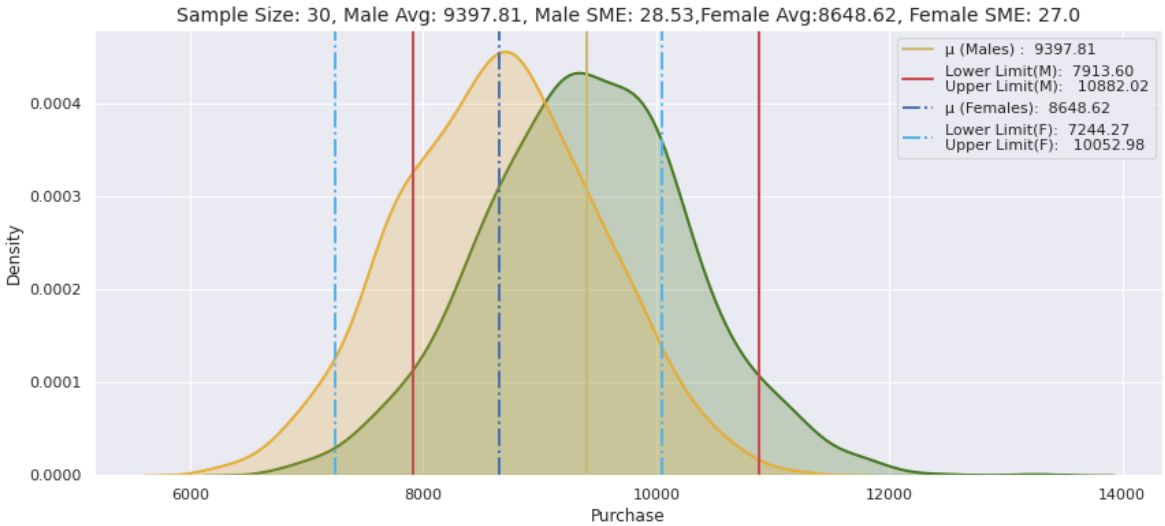
> Analysis of the true mean of purchase values by gender with a
> 90% confidence

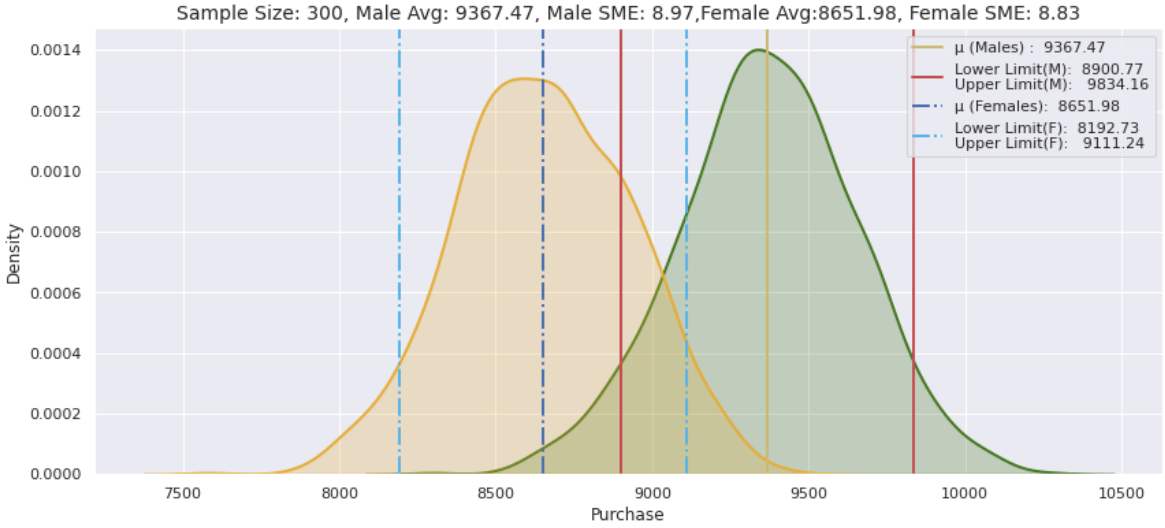
In [326…

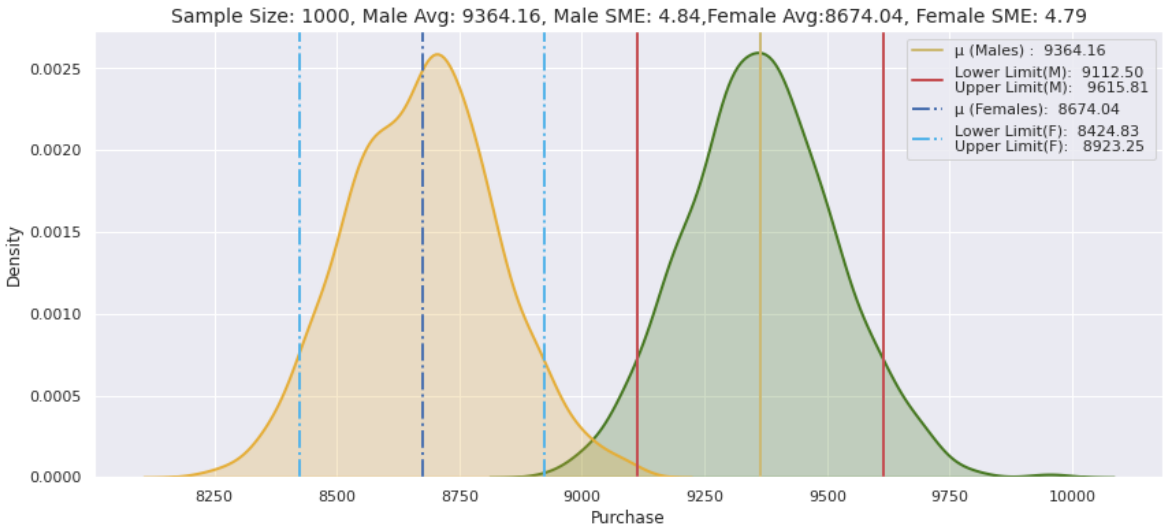
```
itr_size = 1000
size_list = [1, 10, 30, 300, 1000, 100000]
ci = 0.90

array = np.empty((0,7))

for smp_siz in size_list:
    m_avg, f_avg, ll_m, ul_m, ll_f, ul_f = bootstrapping(retail_data_smp_male,re

    array = np.append(array, np.array([['M', ll_m, ul_m, smp_siz, ([ll_m,ul_m])
    array = np.append(array, np.array([['F', ll_f, ul_f, smp_siz, ([ll_f,ul_f])

overlap = pd.DataFrame(array, columns = ['Gender','Lower_limit','Upper_limit','S
print()
```

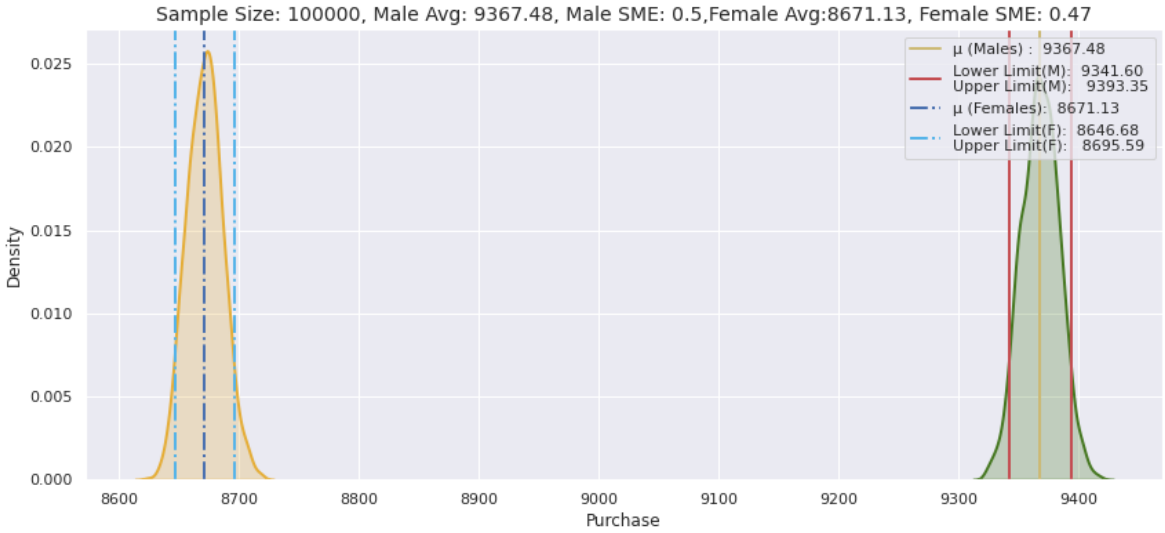

Sample Size: 1, Male Avg: 9287.17, Male SME: 156.51,Female Avg:8501.83, Female SME: 143.36
μ (Males) : 9287.17
Lower Limit(M): 1146.29
Upper Limit(M): 17428.06
μ (Females): 8501.83
Lower Limit(F): 1044.75
Upper Limit(F): 15958.91
Density
Purchase


Sample Size: 10, Male Avg: 9419.23, Male SME: 50.0,Female Avg:8691.71, Female SME: 47.3
μ (Males) : 9419.23
Lower Limit(M): 6818.35
Upper Limit(M): 12020.11
μ (Females): 8691.71
Lower Limit(F): 6231.25
Upper Limit(F): 11152.17
Density
Purchase


Sample Size: 30, Male Avg: 9397.81, Male SME: 28.53,Female Avg:8648.62, Female SME: 27.0
μ (Males) : 9397.81
Lower Limit(M): 7913.60
Upper Limit(M): 10882.02
μ (Females): 8648.62
Lower Limit(F): 7244.27
Upper Limit(F): 10052.98
Density
Purchase







In [327…

```
overlap.loc[(overlap['Gender'] == 'M') & (overlap['Sample_Size'] >= 300)]
```

Out[327]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
6	M	8900.77	9834.16	300	[8900.77, 9834.16]	933.39	90
8	M	9112.5	9615.81	1000	[9112.5, 9615.81]	503.31	90
10	M	9341.6	9393.35	100000	[9341.6, 9393.35]	51.75	90

In [328…

```
overlap.loc[(overlap['Gender'] == 'F') & (overlap['Sample_Size'] >= 300)]
```

Out[328]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
7	F	8192.73	9111.24	300	[8192.73, 9111.24]	918.51	90
9	F	8424.83	8923.25	1000	[8424.83, 8923.25]	498.42	90
11	F	8646.68	8695.59	100000	[8646.68, 8695.59]	48.91	90

Inferences

As the sample size increases, the two groups start to become distinct

With increasing sample size, Standard error of the mean in the samples decreases.

For sample size 100000 is 0.49

For Female (sample size 100000) range for mean purchase with confidence interval 90% is [8645.68, 8696.14]

For Male range for mean purchase with confidence interval 90% is [9341.03, 9393.94]

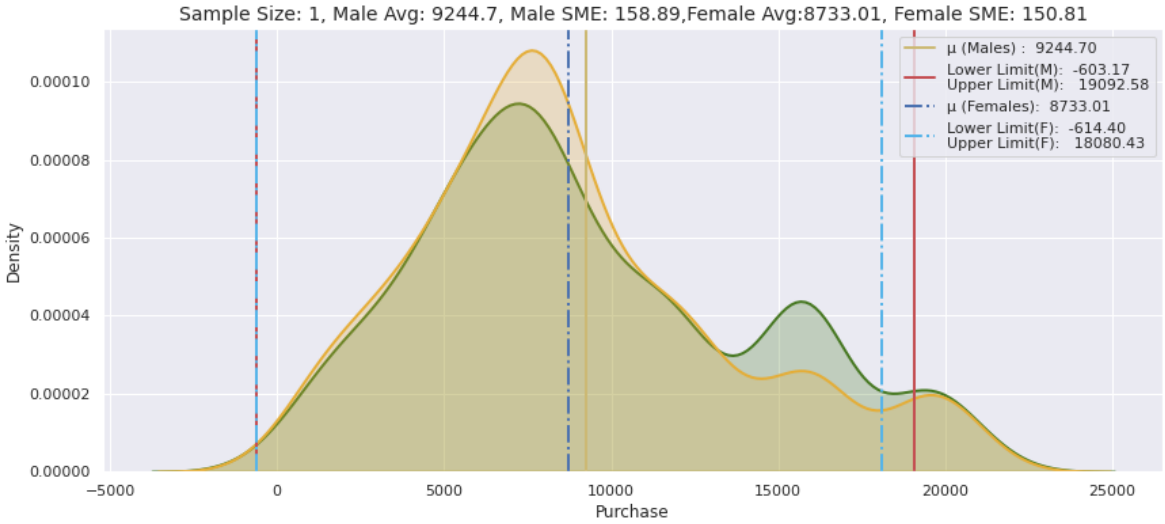
CLT Analysis for mean purchase with confidence 95% - Based on Gender

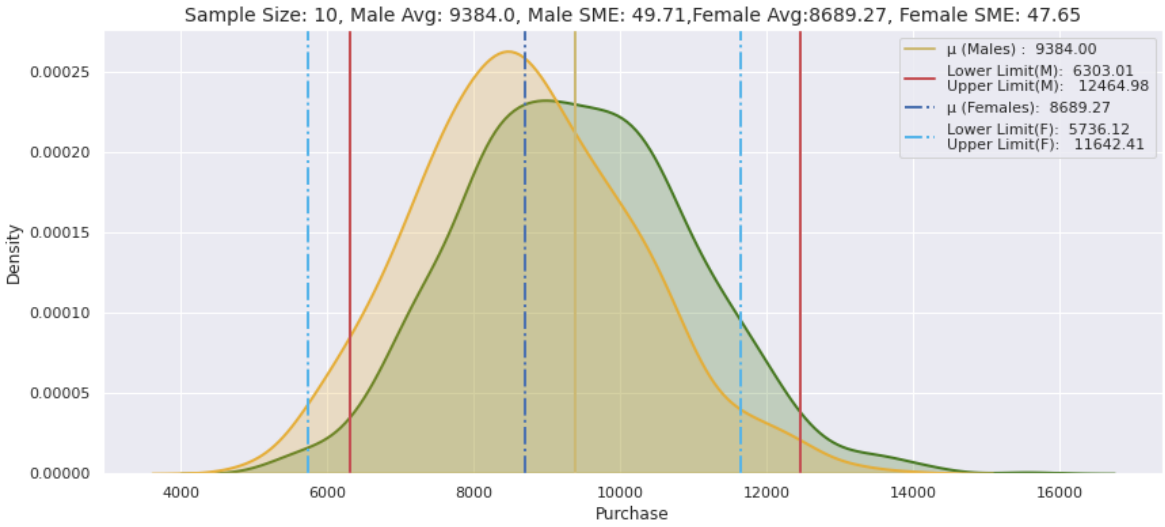
Analysis of the true mean of purchase values by gender with a 95% confidence

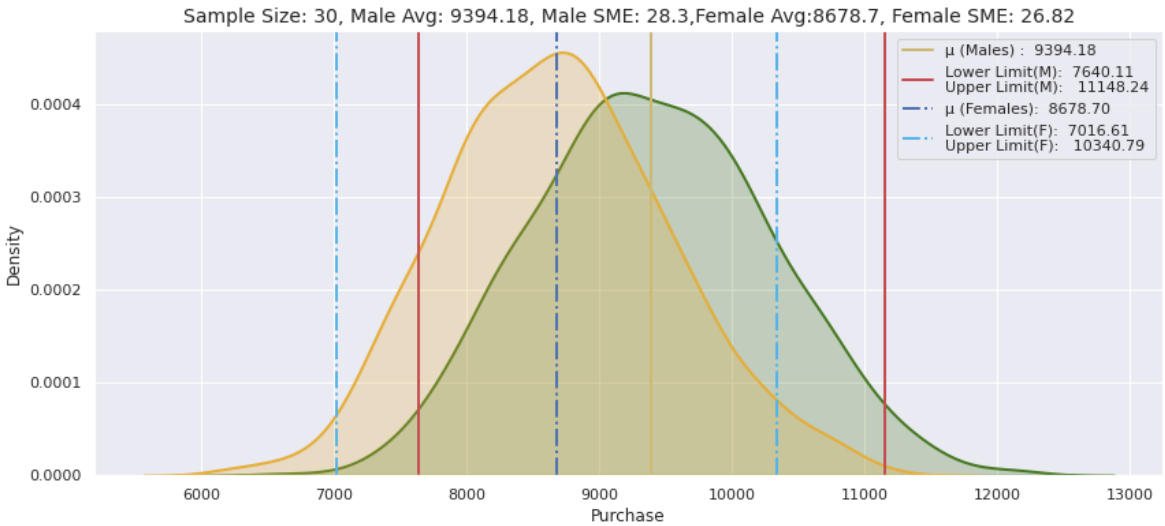
In [329…

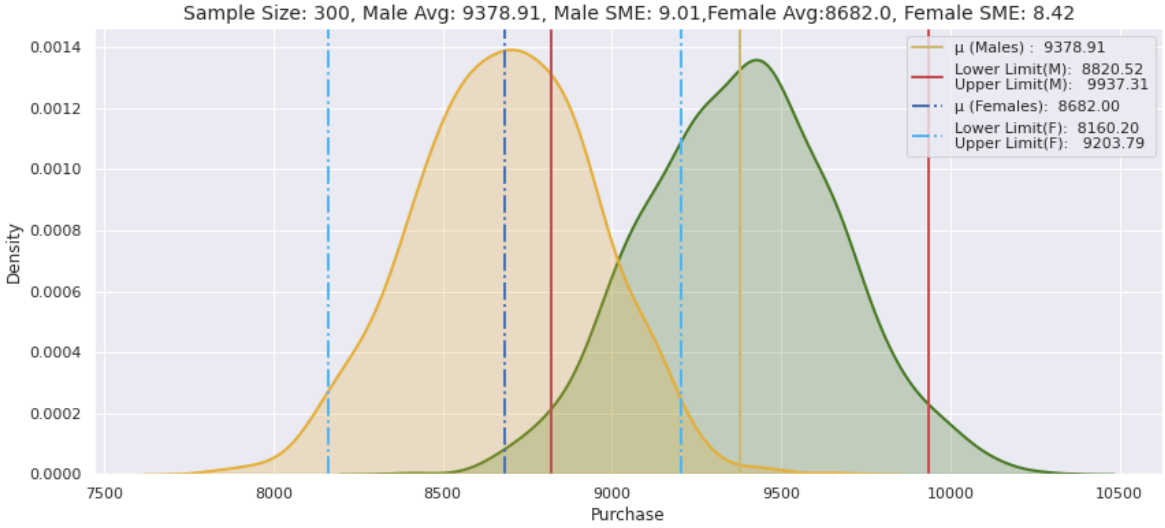
```
itr_size = 1000
size_list = [1, 10, 30, 300, 1000, 100000]
ci = 0.95
```

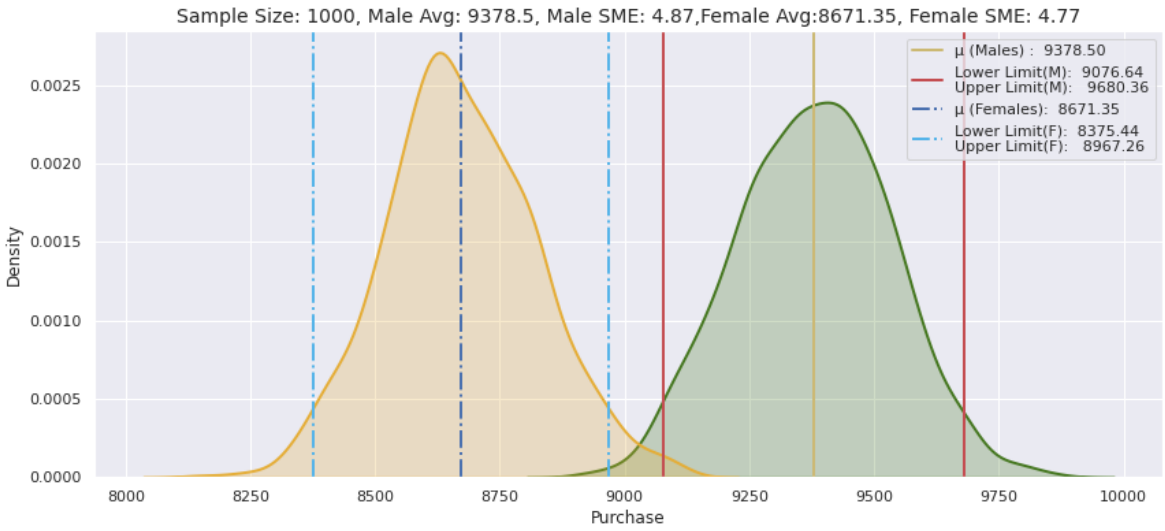
```
array = np.empty((0,7))

for smp_siz in size_list:
    m_avg, f_avg, ll_m, ul_m, ll_f, ul_f = bootstrapping(retail_data_smp_male,re

    array = np.append(array, np.array([['M', ll_m, ul_m, smp_siz, ([ll_m,ul_m])
    array = np.append(array, np.array([['F', ll_f, ul_f, smp_siz, ([ll_f,ul_f])

overlap_95 = pd.DataFrame(array, columns = ['Gender','Lower_limit','Upper_limit'
overlap = pd.concat([overlap, overlap_95], axis=0)
```

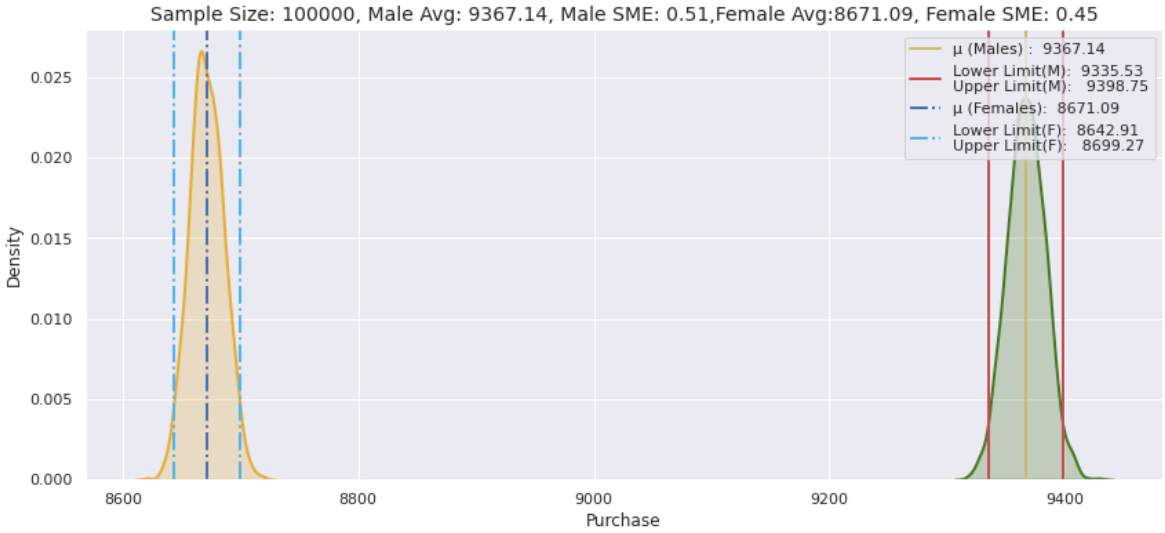












In [330…

```
overlap_95.loc[(overlap_95['Gender'] == 'M') & (overlap_95['Sample_Size'] >= 300
```

Out[330]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
6	M	8820.52	9937.31	300	[8820.52, 9937.31]	1116.79	95
8	M	9076.64	9680.36	1000	[9076.64, 9680.36]	603.72	95
10	M	9335.53	9398.75	100000	[9335.53, 9398.75]	63.22	95

In [331…

```
overlap_95.loc[(overlap_95['Gender'] == 'F') & (overlap_95['Sample_Size'] >= 300
```

Out[331]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
7	F	8160.2	9203.79	300	[8160.2, 9203.79]	1043.59	95
9	F	8375.44	8967.26	1000	[8375.44, 8967.26]	591.82	95
11	F	8642.91	8699.27	100000	[8642.91, 8699.27]	56.36	95

Inferences

Using confidence interval 95%, the mean purchase value by gender shows a similar pattern to that found with confidence interval 90%-

As the sample size increases, the Male and female groups start to become distinct

With increasing sample size, Standard error of the mean in the samples decreases. For sample size 100000 is 0.47

For Female (sample size 100000) range for mean purchase with confidence interval 90% is [8642.58, 8701.58]

For Male range for mean purchase with confidence interval 95% is [9336.23, 9397.53]

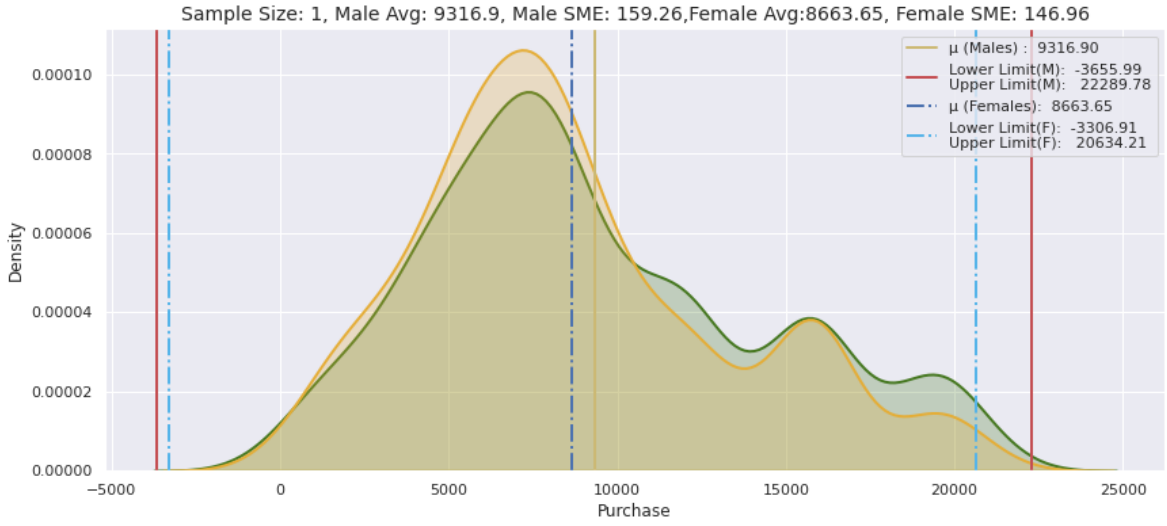
Overlappings are increasing with a confidence interval of 95%. Due to the increasing CI, we consider higher ranges within which the actual population might fall, so that both mean purchase are more likely to fall within the same range.

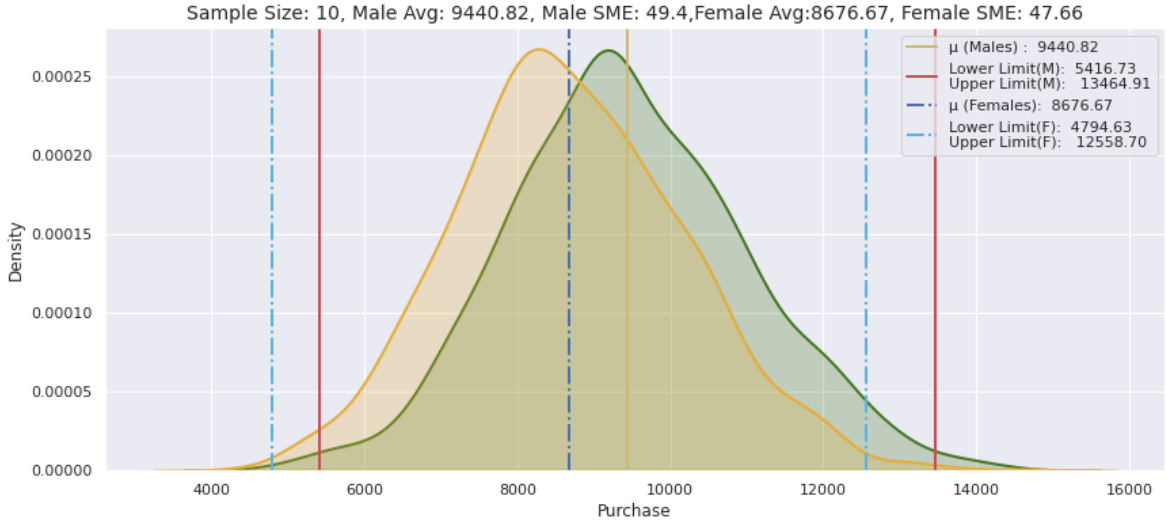
CLT Analysis for mean purchase with confidence 99% - Based on Gender

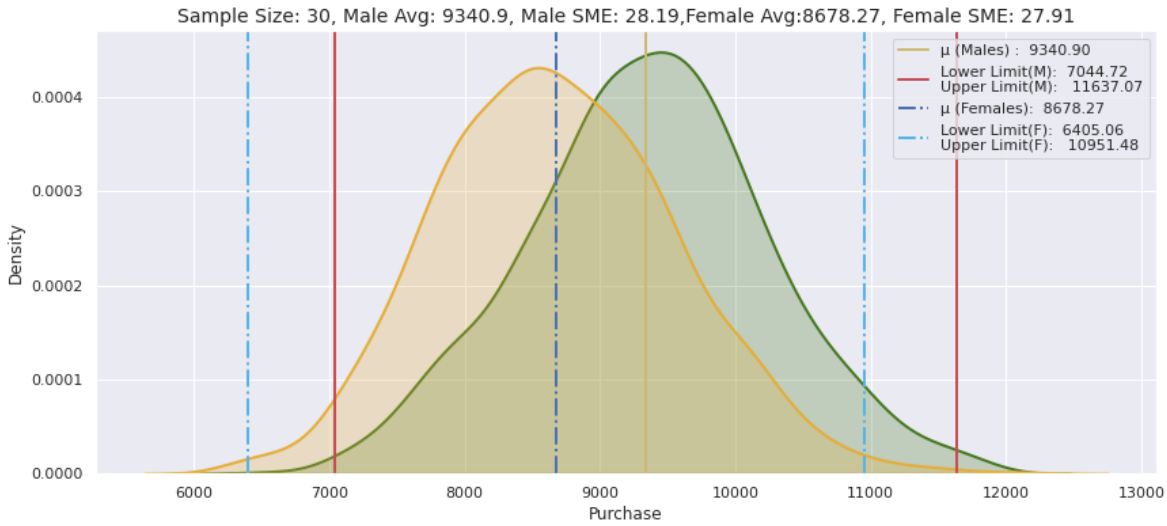
Analysis of the true mean of purchase values by gender with a 99% confidence.

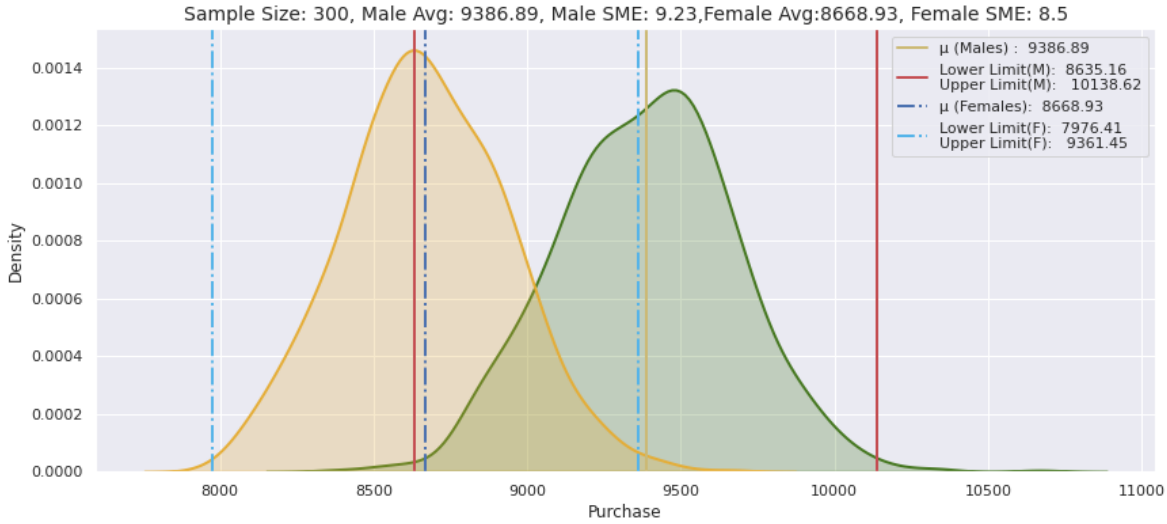
In [332…

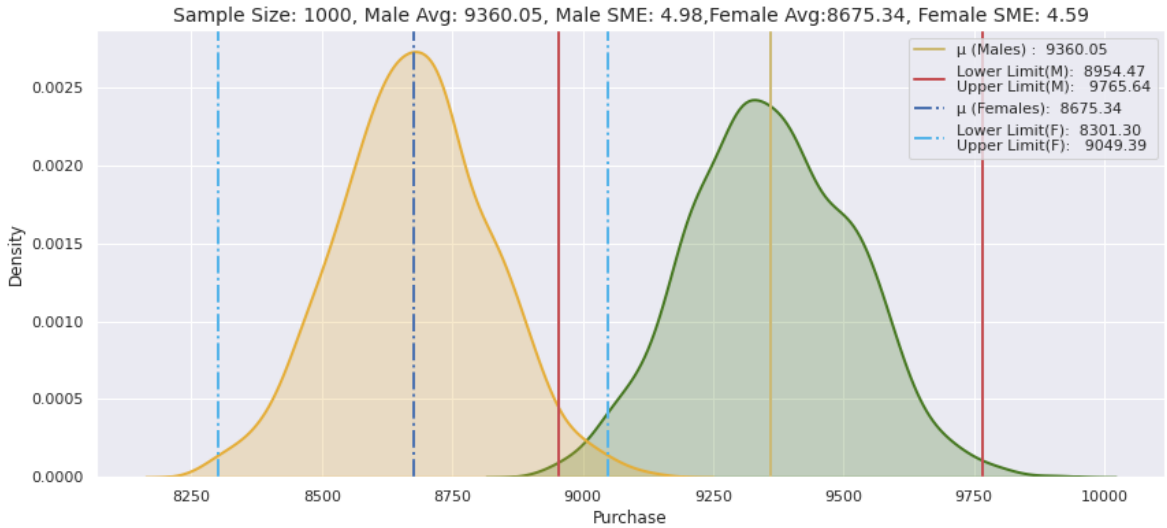
```
itr_size = 1000
size_list = [1, 10, 30, 300, 1000, 100000]
ci = 0.99

array = np.empty((0,7))

for smp_siz in size_list:
    m_avg, f_avg, ll_m, ul_m, ll_f, ul_f = bootstrapping(retail_data_smp_male,re

    array = np.append(array, np.array([['M', ll_m, ul_m, smp_siz, ([ll_m,ul_m])
    array = np.append(array, np.array([['F', ll_f, ul_f, smp_siz, ([ll_f,ul_f])

overlap_99 = pd.DataFrame(array, columns = ['Gender','Lower_limit','Upper_limit'
overlap = pd.concat([overlap, overlap_99], axis=0)
```

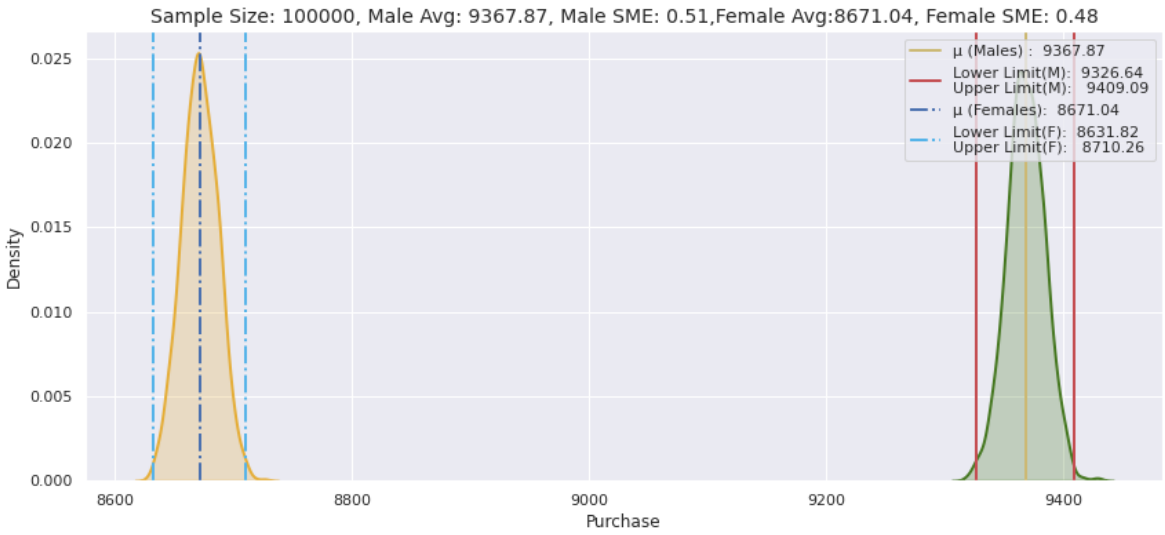





Sample Size: 30, Male Avg: 9340.9, Male SME: 28.19,Female Avg:8678.27, Female SME: 27.91
μ (Males) : 9340.90
Lower Limit(M): 7044.72
Upper Limit(M): 11637.07
μ (Females): 8678.27
Lower Limit(F): 6405.06
Upper Limit(F): 10951.48
Density
Purchase


Sample Size: 300, Male Avg: 9386.89, Male SME: 9.23,Female Avg:8668.93, Female SME: 8.5
μ (Males) : 9386.89
Lower Limit(M): 8635.16
Upper Limit(M): 10138.62
μ (Females): 8668.93
Lower Limit(F): 7976.41
Upper Limit(F): 9361.45
Density
Purchase


Sample Size: 1000, Male Avg: 9360.05, Male SME: 4.98,Female Avg:8675.34, Female SME: 4.59
μ (Males) : 9360.05
Lower Limit(M): 8954.47
Upper Limit(M): 9765.64
μ (Females): 8675.34
Lower Limit(F): 8301.30
Upper Limit(F): 9049.39
Density
Purchase



In [333…
```
overlap_99.loc[(overlap_99['Gender'] == 'M') & (overlap_99['Sample_Size'] >= 300
```

Out[333]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
6	M	8635.16	10138.62	300	[8635.16, 10138.62]	1503.46	99
8	M	8954.47	9765.64	1000	[8954.47, 9765.64]	811.17	99
10	M	9326.64	9409.09	100000	[9326.64, 9409.09]	82.45	99

In [334…
```
overlap_99.loc[(overlap_99['Gender'] == 'F') & (overlap_99['Sample_Size'] >= 300
```

Out[334]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
7	F	7976.41	9361.45	300	[7976.41, 9361.45]	1385.04	99
9	F	8301.3	9049.39	1000	[8301.3, 9049.39]	748.09	99
11	F	8631.82	8710.26	100000	[8631.82, 8710.26]	78.44	99

In [335…
```
overlap.loc[(overlap['Gender'] == 'M') & (overlap['Sample_Size'] >= 10000)]
```

Out[335]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
10	M	9341.6	9393.35	100000	[9341.6, 9393.35]	51.75	90
10	M	9335.53	9398.75	100000	[9335.53, 9398.75]	63.22	95
10	M	9326.64	9409.09	100000	[9326.64, 9409.09]	82.45	99

In [336…

```
overlap.loc[(overlap['Gender'] == 'F') & (overlap['Sample_Size'] >= 10000)]
```

Out[336]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
11	F	8646.68	8695.59	100000	[8646.68, 8695.59]	48.91	90
11	F	8642.91	8699.27	100000	[8642.91, 8699.27]	56.36	95
11	F	8631.82	8710.26	100000	[8631.82, 8710.26]	78.44	99

Inferences

Using confidence interval 99%, the mean purchase value by gender shows a similar pattern to that found with confidence interval 90% & 95%-

As the sample size increases, the Male and female groups start to become distinct

With increasing sample size, Standard error of the mean in the samples decreases. For sample size 100000 is 0.45

For Female (sample size 100000) range for mean purchase with confidence interval 99% is [8634.54, 8707.85]

For Male range for mean purchase with confidence interval 90% is [9328.03, 9409.07]

When the confidence percentage increases, the spread, that is the difference between the upper and lower limits, also increases. For Female Confidence percent as [90,95,99] have difference between the upper & lower limits as [50.46,59,73.31]

Recommendations

In light of the fact that females spend less than males on average, management needs to focus on their specific needs

differently. Adding some additional offers for women can increase their spending on Black Friday.

CLT Analysis for mean purchase with confidence 95% and 99% - Based on Marital Status

Analysis of the true mean of purchase values by marital Status with a 95% and 99% confidence.

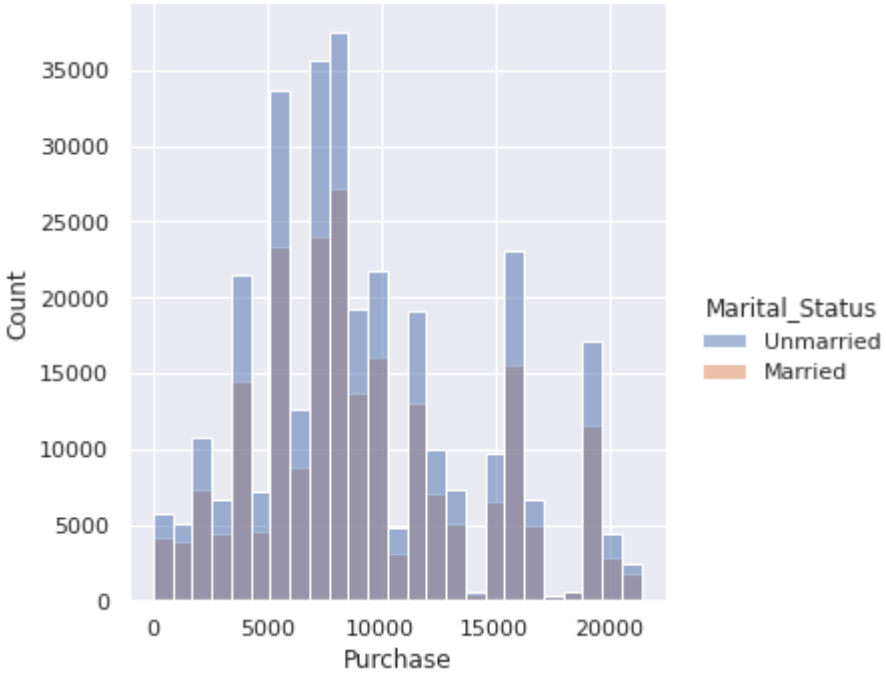
In [337…
```
retail_data_v1['Marital_Status'].replace(to_replace = 0, value = 'Unmarried', ir
retail_data_v1['Marital_Status'].replace(to_replace = 1, value = 'Married', inpl
```

In [338…
```
retail_data_v1.sample(500,replace=True).groupby(['Marital_Status'])['Purchase'].
```

Out[338]:

Marital_Status	count	mean	std	min	25%	50%	75%	ma
Unmarried	296.0	9489.851351	5196.748833	60.0	5749.5	8031.5	13441.75	20811.
Married	204.0	8673.053922	4952.532813	140.0	5384.0	7901.0	11565.50	21310.

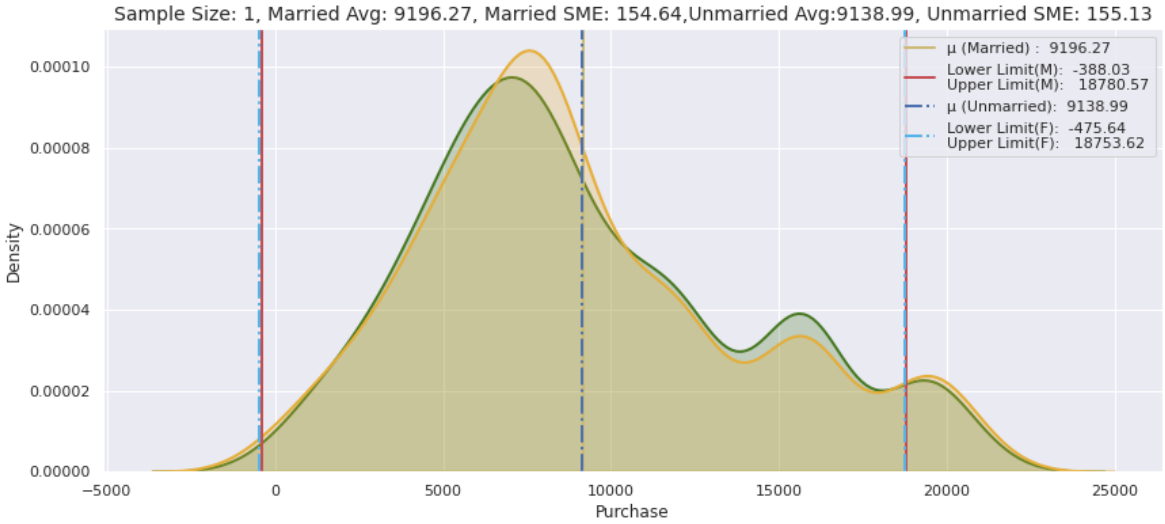
In [339…
```
sns.displot(data = retail_data_v1, x = 'Purchase', hue = 'Marital_Status',bins =
plt.show()
```

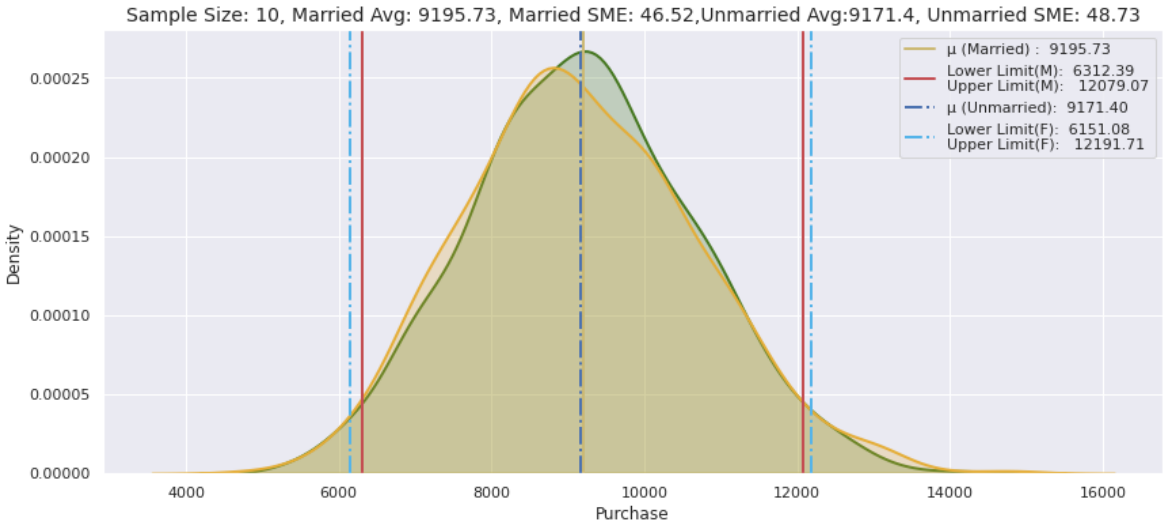


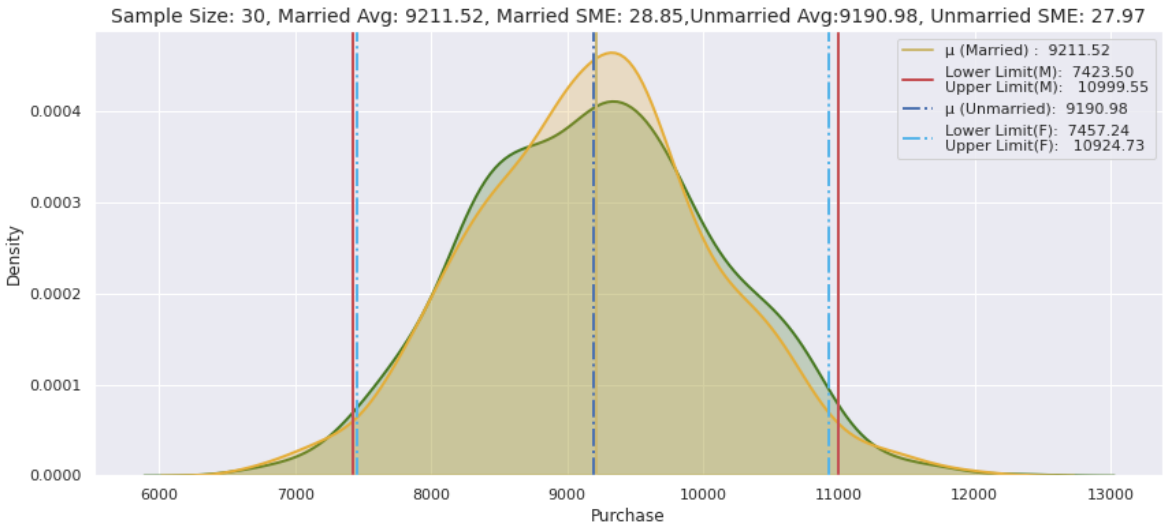
In [340…
```
retail_data_smp_married = retail_data_v1[retail_data_v1['Marital_Status'] == 'Ma
retail_data_smp_unmarried = retail_data_v1[retail_data_v1['Marital_Status'] == '
```

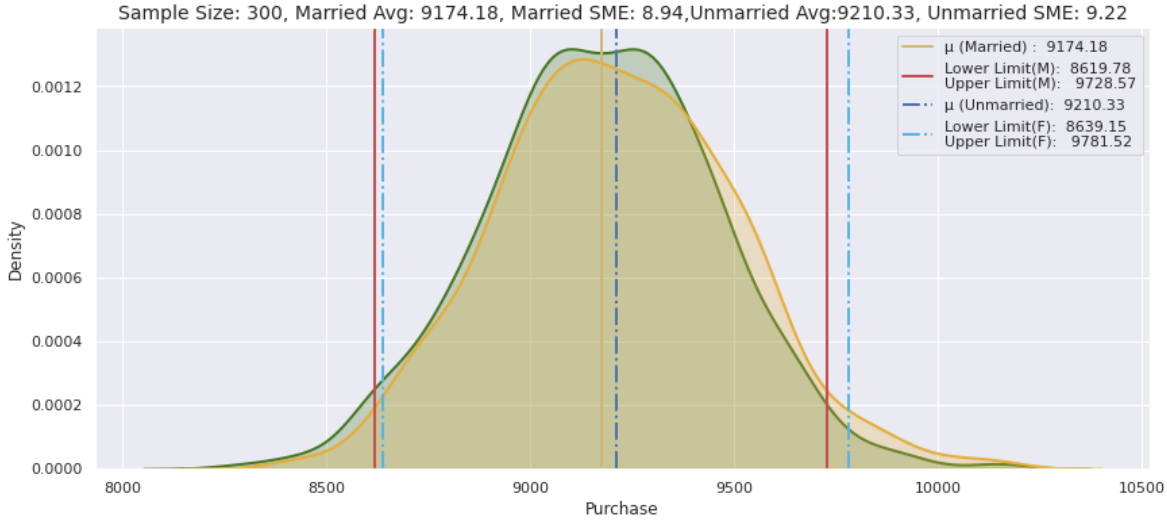
In [341…
```
itr_size = 1000
size_list = [1, 10, 30, 300, 1000, 100000]
ci = 0.95
```

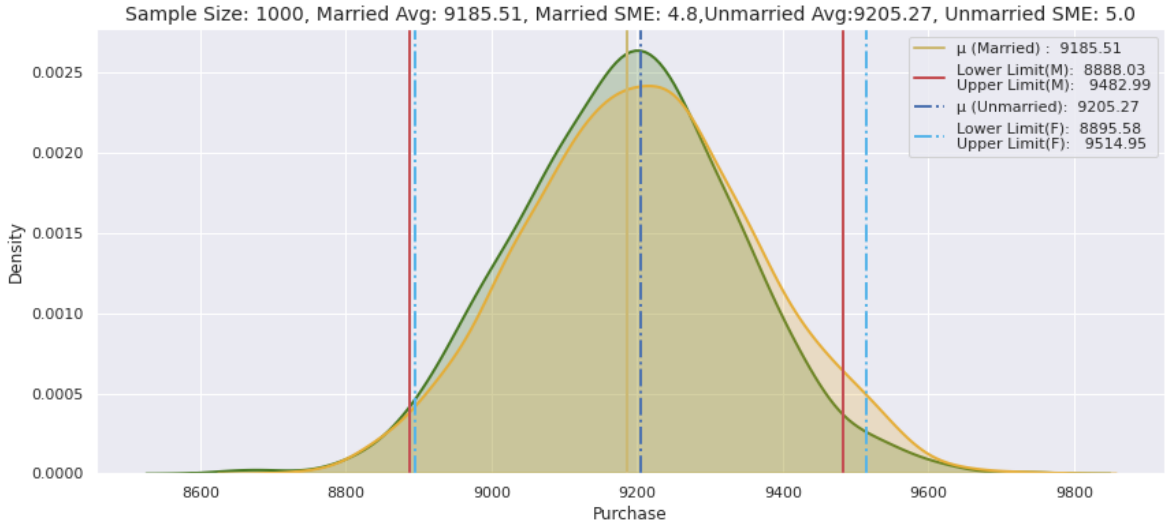
```
array = np.empty((0,7))

for smp_siz in size_list:
    m_avg, f_avg, ll_m, ul_m, ll_u, ul_u = bootstrapping_m_vs_um(retail_data_smp

    array = np.append(array, np.array([['Married', ll_m, ul_m, smp_siz, ([ll_m,u
    array = np.append(array, np.array([['Unmarried', ll_u, ul_u, smp_siz, ([ll_u

overlap = pd.DataFrame(array, columns = ['Marital_Status','Lower_limit','Upper_l
```

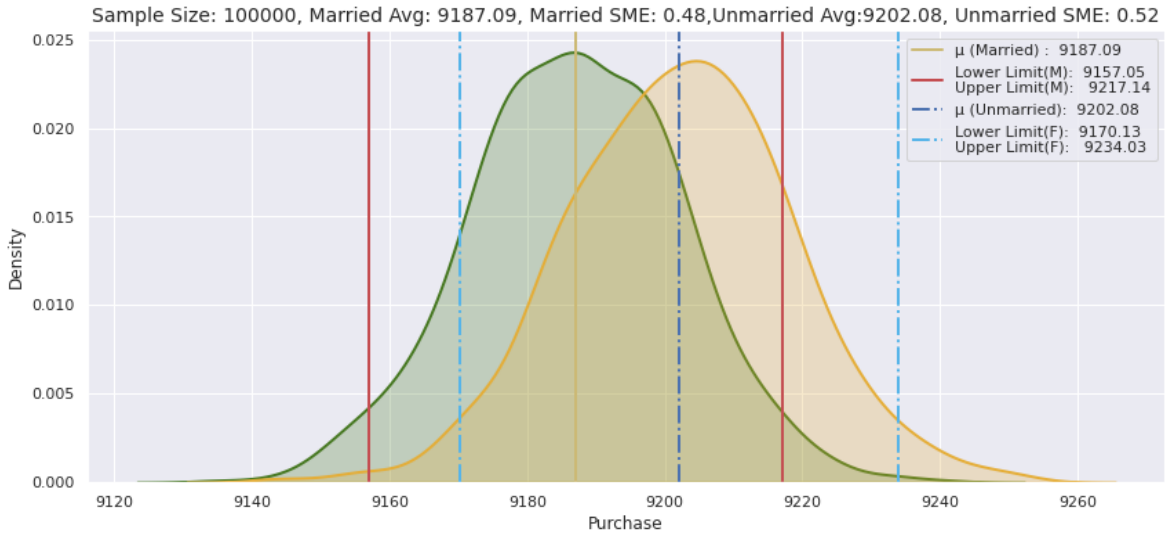








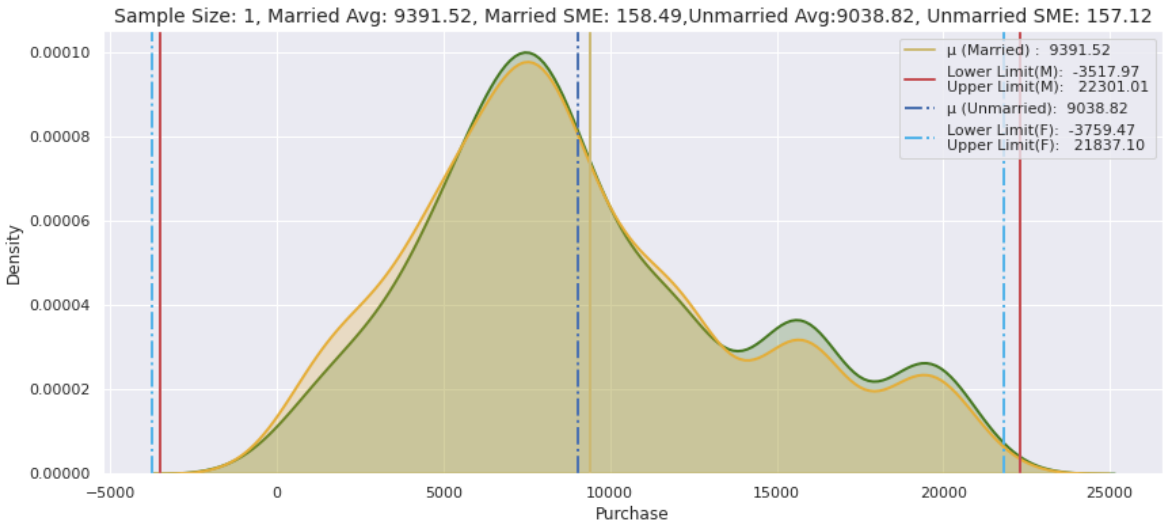


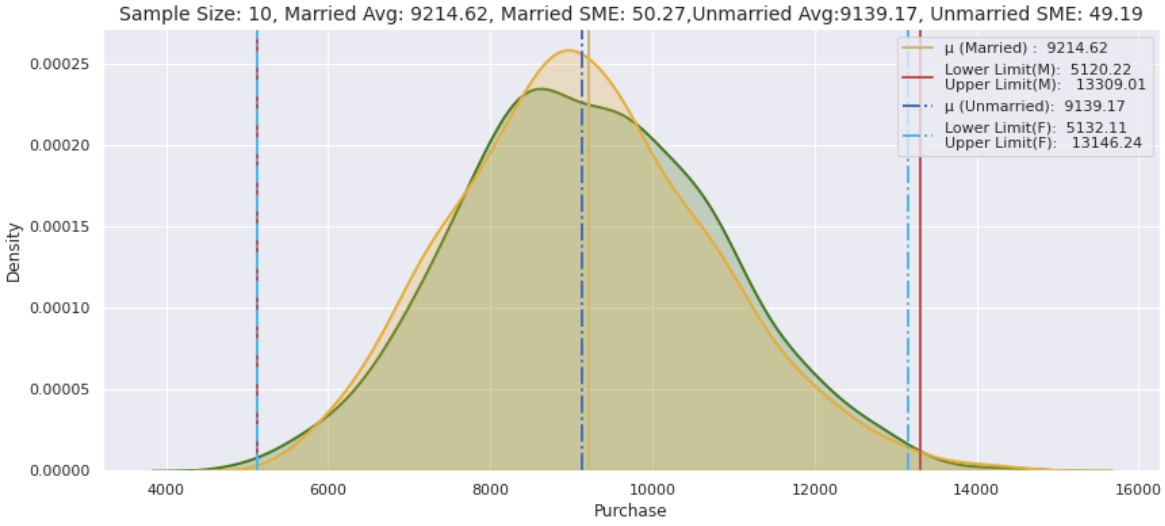


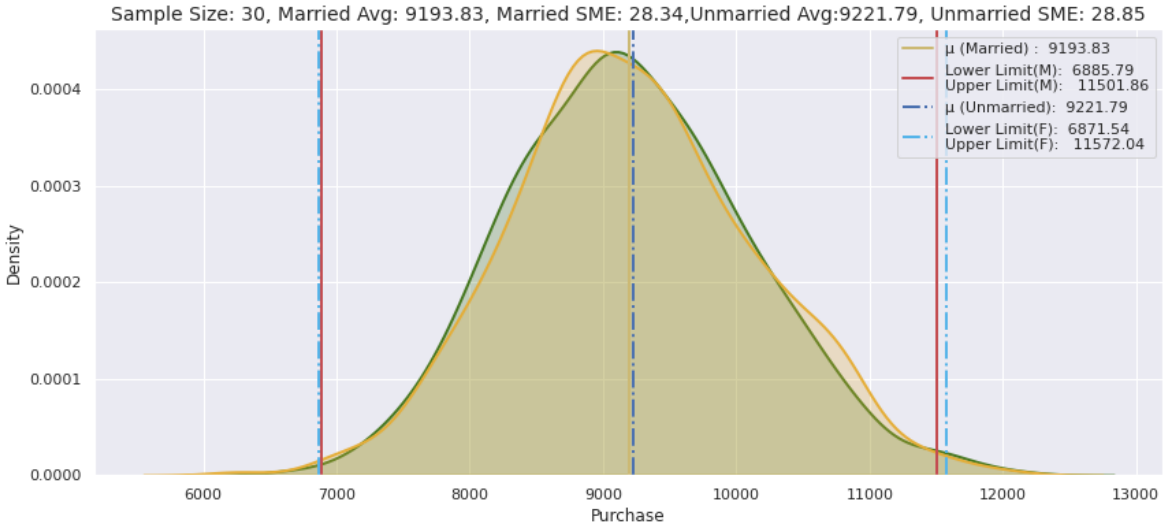
In [342…

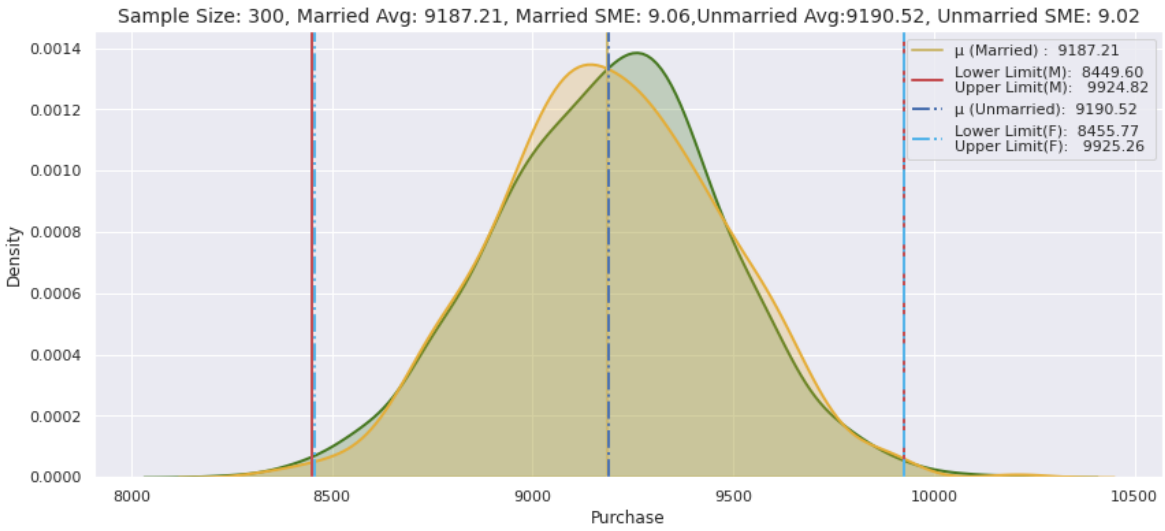
```
itr_size = 1000
size_list = [1, 10, 30, 300, 1000, 100000]
ci = 0.99

array = np.empty((0,7))

for smp_siz in size_list:
    m_avg, f_avg, ll_m, ul_m, ll_u, ul_u = bootstrapping_m_vs_um(retail_data_smp

    array = np.append(array, np.array([['Married', ll_m, ul_m, smp_siz, ([ll_m,u
    array = np.append(array, np.array([['Unmarried', ll_u, ul_u, smp_siz, ([ll_u
```

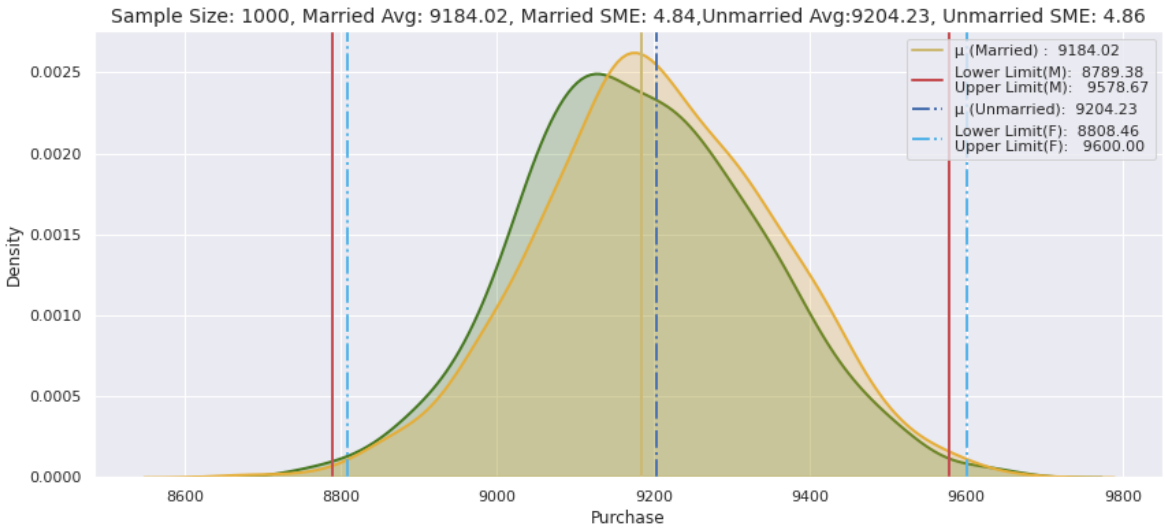
```
overlap = pd.DataFrame(array, columns = ['Marital_Status','Lower_limit','Upper_l
```

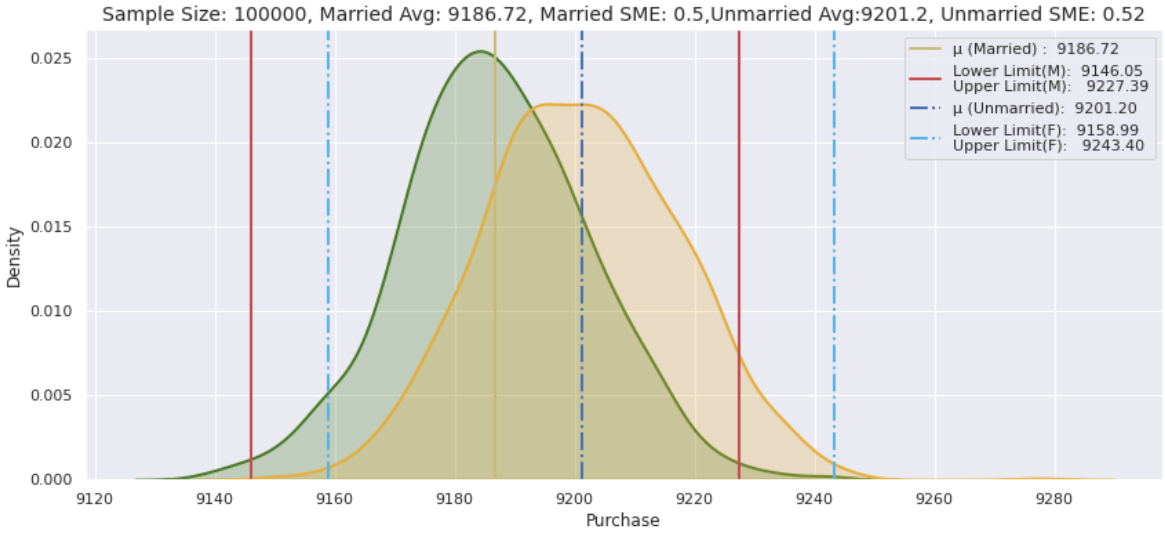












In [343…

```
overlap.head()
```

Out[343]:

	Marital_Status	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidenc
0	Married	-3517.97	22301.01	1	[-3517.97, 22301.01]	25818.98	
1	Unmarried	-3759.47	21837.1	1	[-3759.47, 21837.1]	25596.57	
2	Married	5120.22	13309.01	10	[5120.22, 13309.01]	8188.79	
3	Unmarried	5132.11	13146.24	10	[5132.11, 13146.24]	8014.13	
4	Married	6885.79	11501.86	30	[6885.79, 11501.86]	4616.07	

In [344…

```
overlap.loc[(overlap['Marital_Status'] == 'Married') & (overlap['Sample_Size'] >
```

Out[344]:

	Marital_Status	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidenc
6	Married	8449.6	9924.82	300	[8449.6, 9924.82]	1475.22	
8	Married	8789.38	9578.67	1000	[8789.38, 9578.67]	789.29	
10	Married	9146.05	9227.39	100000	[9146.05, 9227.39]	81.34	

In [345…

```
overlap.loc[(overlap['Marital_Status'] == 'Unmarried') & (overlap['Sample_Size']
```

Out[345]:

	Marital_Status	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidenc
7	Unmarried	8455.77	9925.26	300	[8455.77, 9925.26]	1469.49	
9	Unmarried	8808.46	9600.0	1000	[8808.46, 9600.0]	791.54	
11	Unmarried	9158.99	9243.4	100000	[9158.99, 9243.4]	84.41	

Inference

Overlapping is evident for married vs single customer spend even when more samples are analyzed, which indicates that customers spend the same regardless of whether they are single or married.

For Unmarried customer (sample size 100000) range for mean purchase with confidence interval 99% is [9162.0, 9241.98]

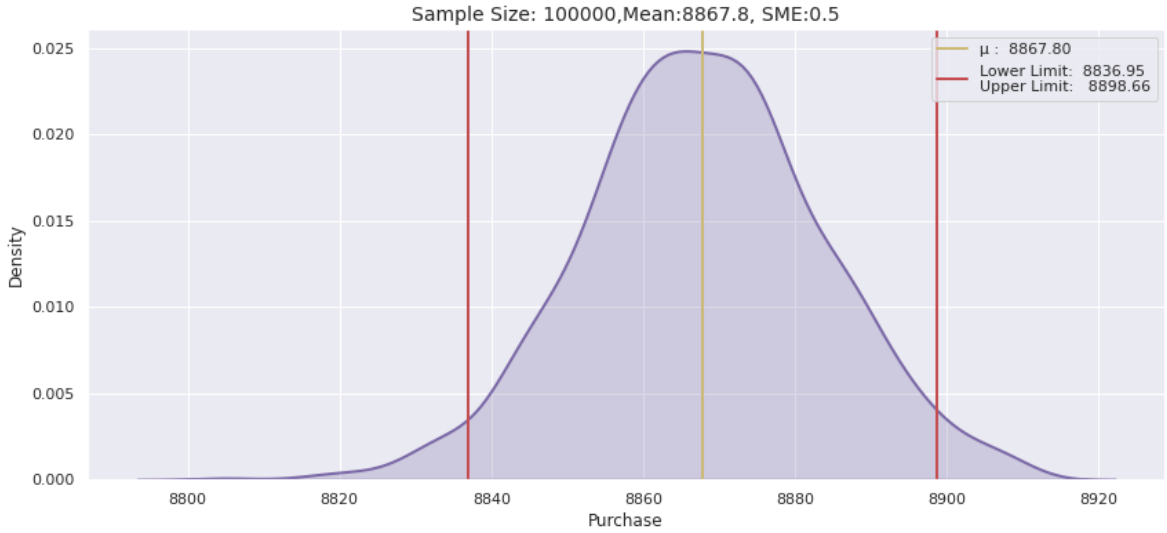
For married customer range for mean purchase with confidence interval 90% is [9148.09, 9227.05]

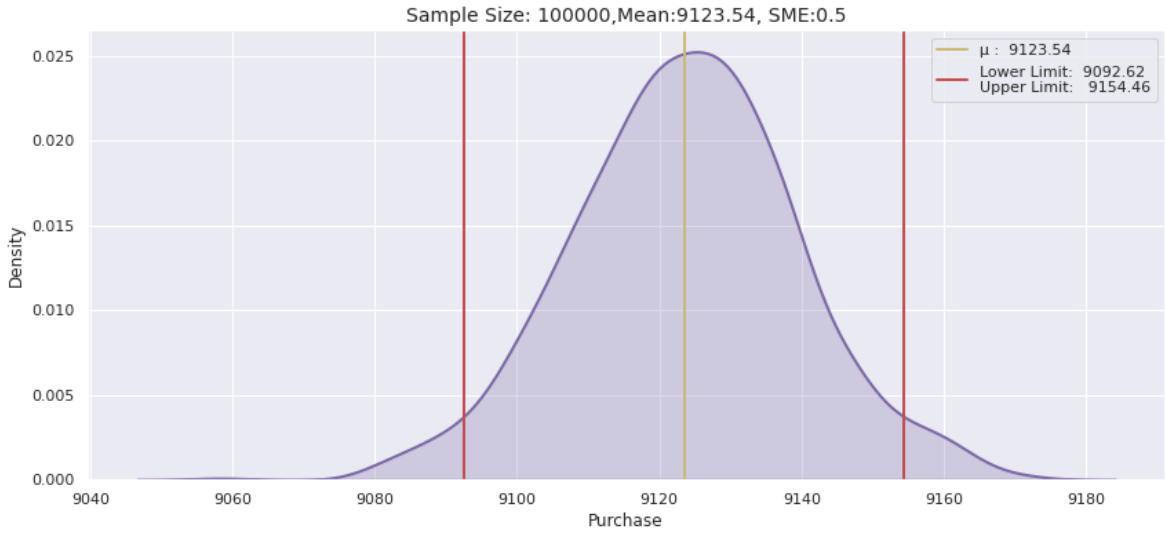
CLT Analysis for mean purchase with confidence 95%-99% - Based on Age Group

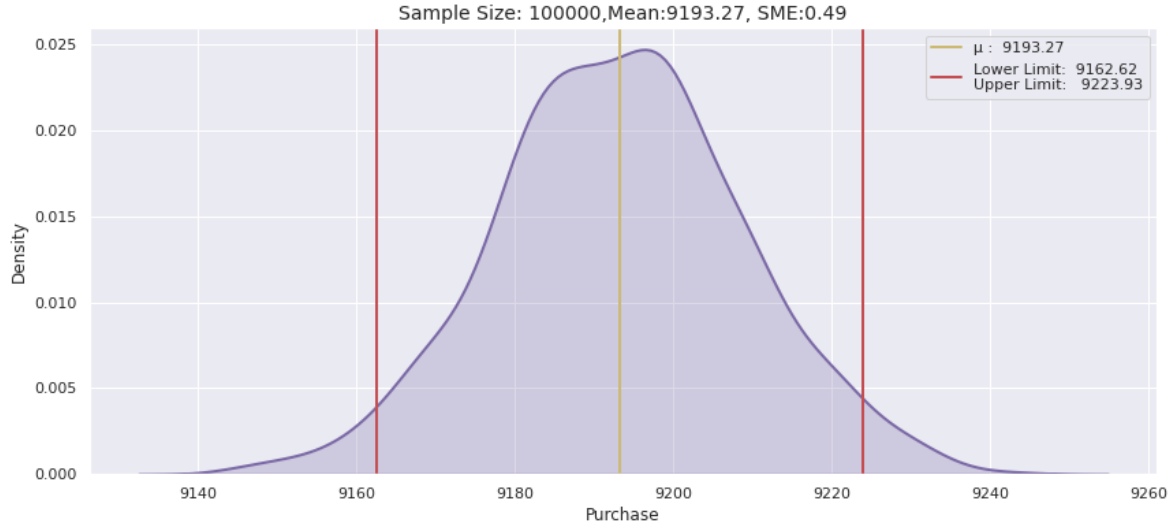
Analysis of the true mean of purchase values by Age Group with a 95% and 99% confidence.

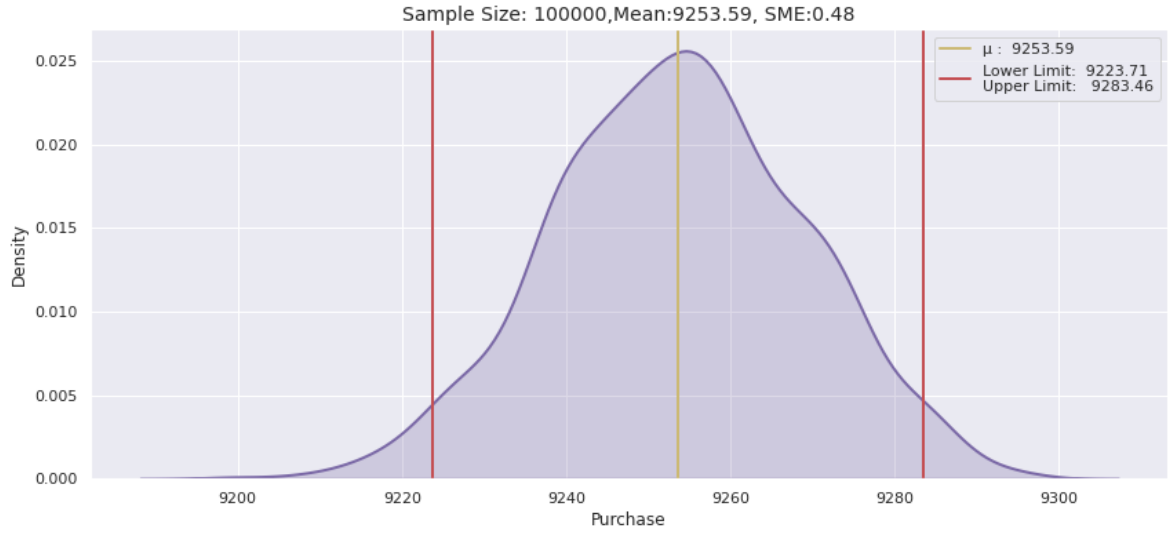
In [346…

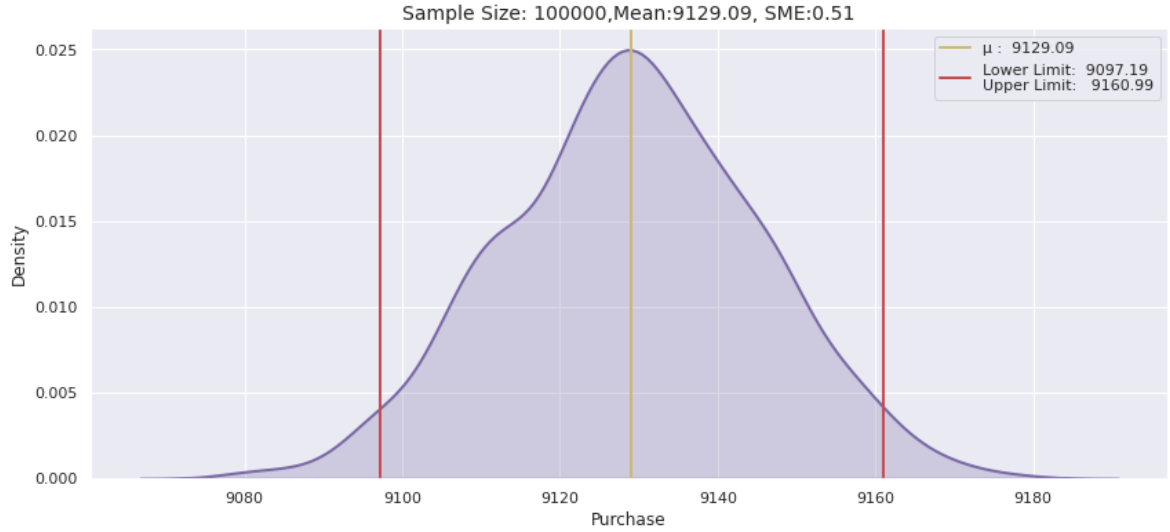
```
itr_size = 1000
smp_size = 1000
ci = 0.95
age_list =['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

array = np.empty((0,8))

for age in age_list:
    mean, ll_m, ul_m = bootstrapping_age(retail_data_v1[retail_data_v1['Age'] ==

    array = np.append(array, np.array([[age,np.round(mean,2), ll_m, ul_m, smp_si

age_data = pd.DataFrame(array, columns = ['Age_Group','Mean','Lower_limit','Uppe
```

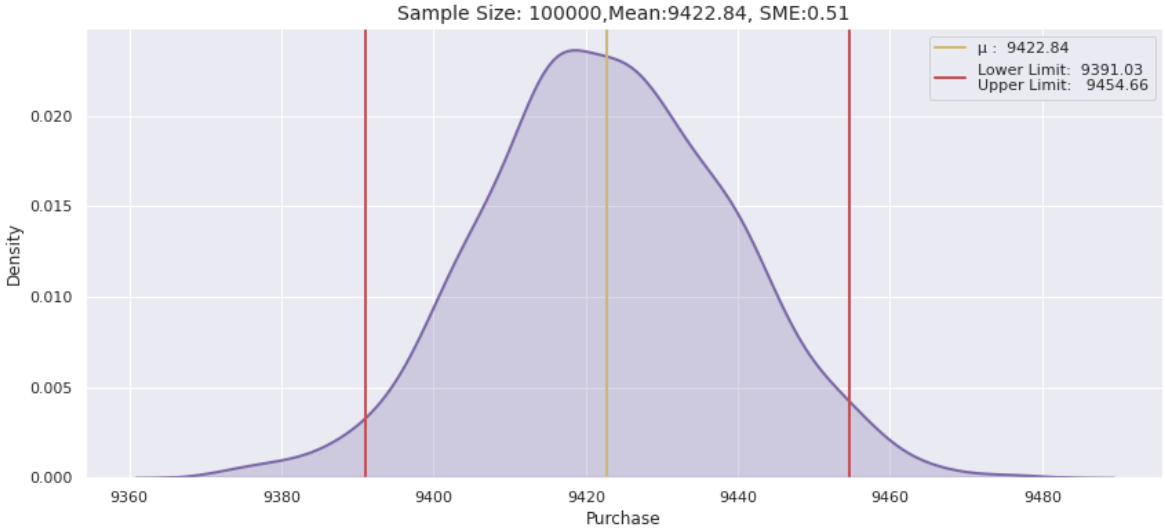


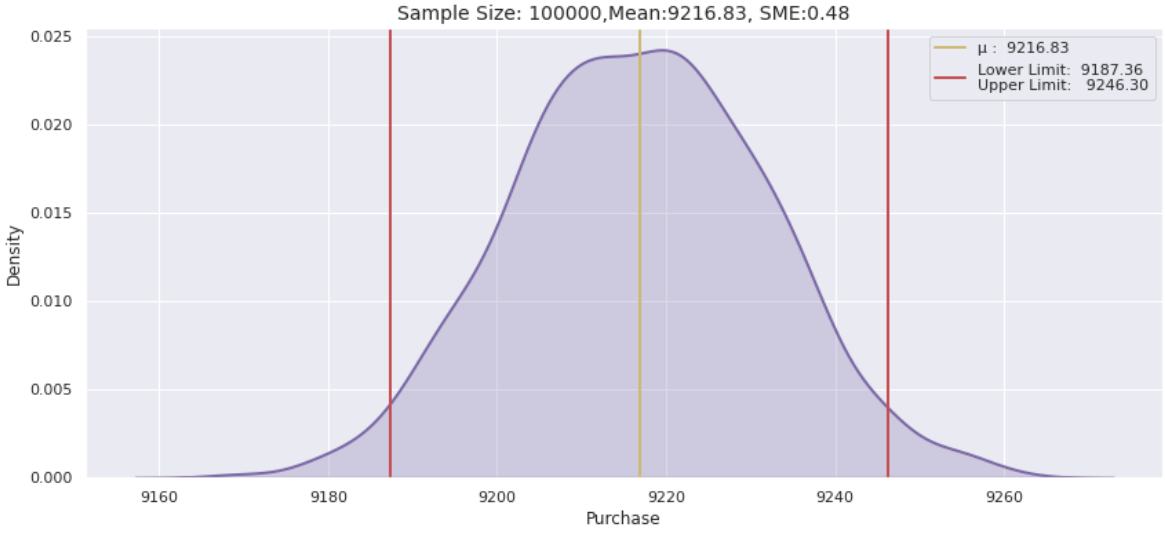



Sample Size: 100000,Mean:9193.27, SME:0.49
μ : 9193.27
Lower Limit: 9162.62
Upper Limit: 9223.93
Density
Purchase


Sample Size: 100000,Mean:9253.59, SME:0.48
μ : 9253.59
Lower Limit: 9223.71
Upper Limit: 9283.46
Density
Purchase

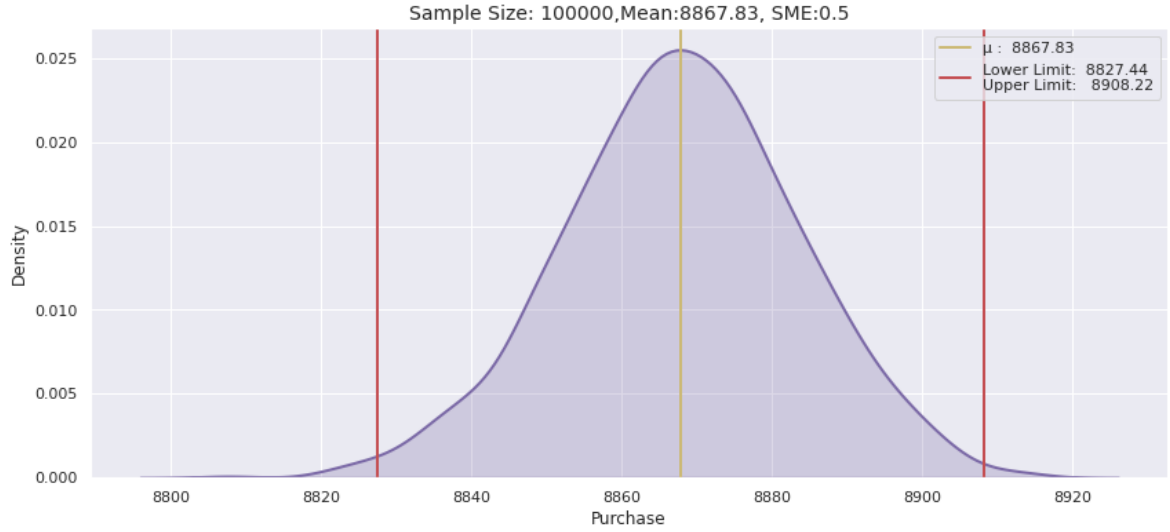

Sample Size: 100000,Mean:9129.09, SME:0.51
μ : 9129.09
Lower Limit: 9097.19
Upper Limit: 9160.99
Density
Purchase

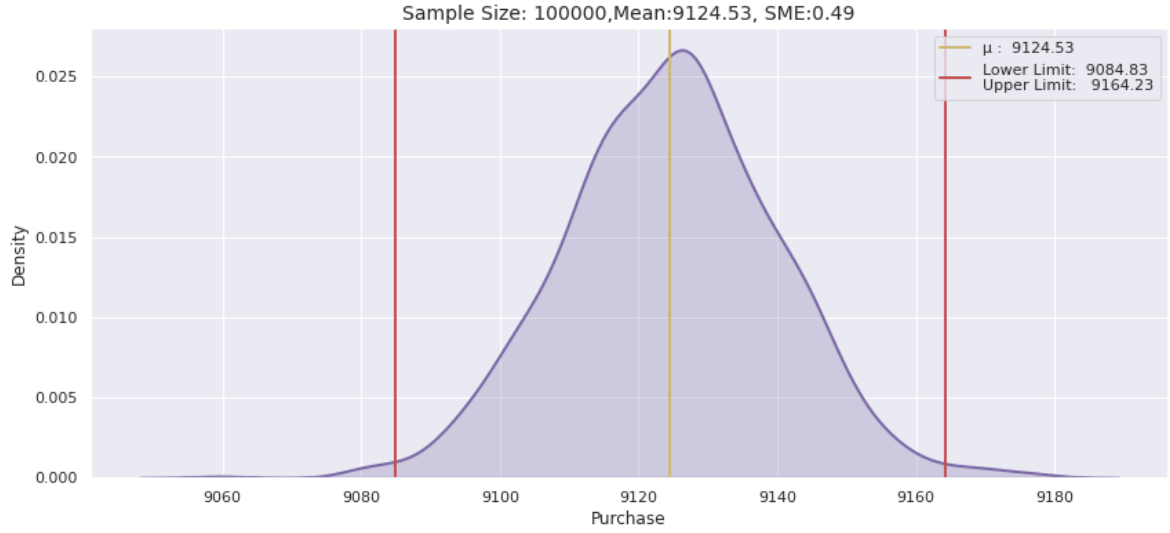


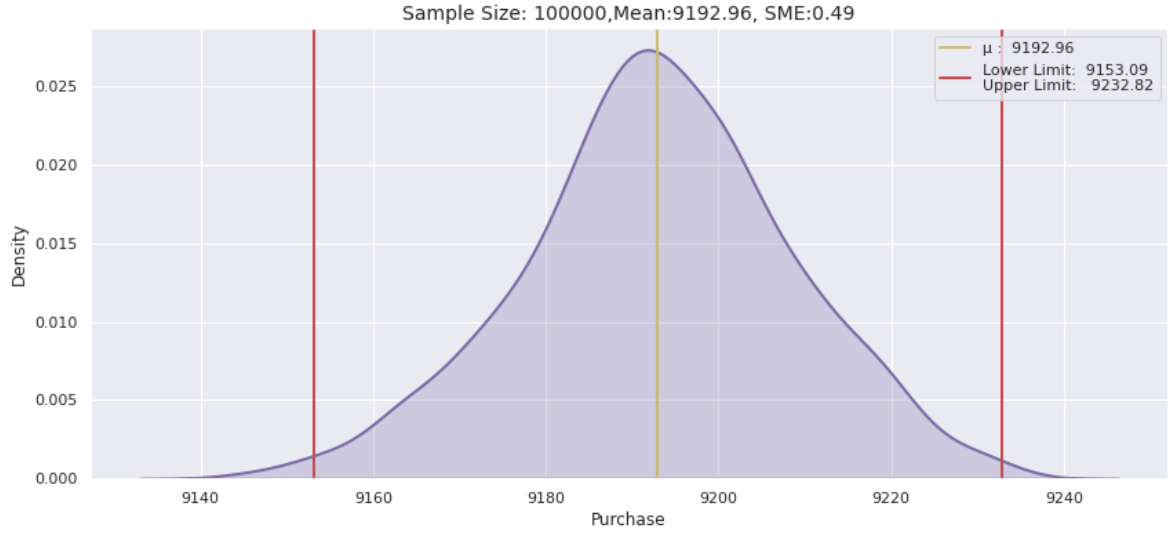


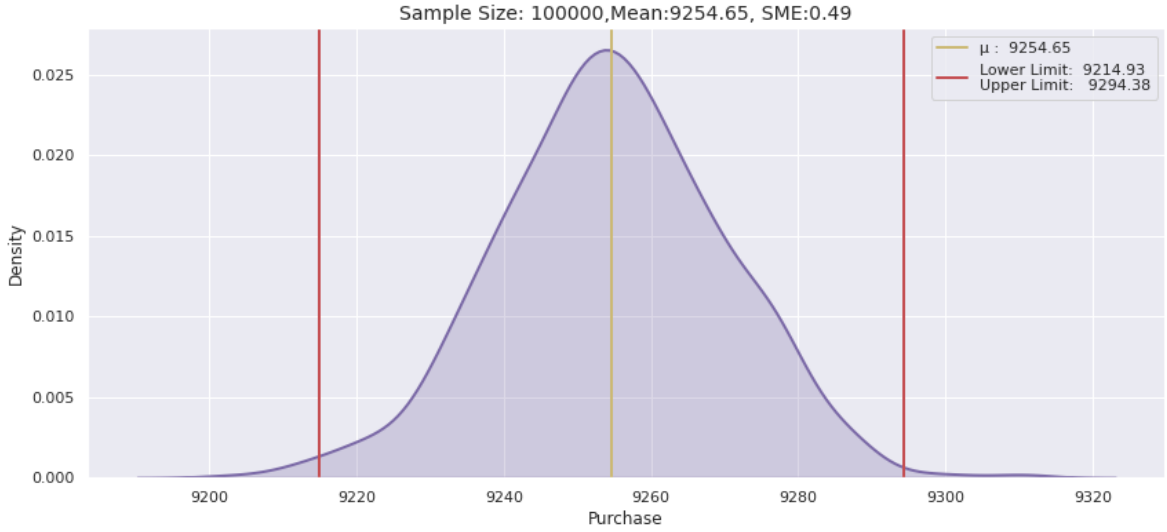
In [347…

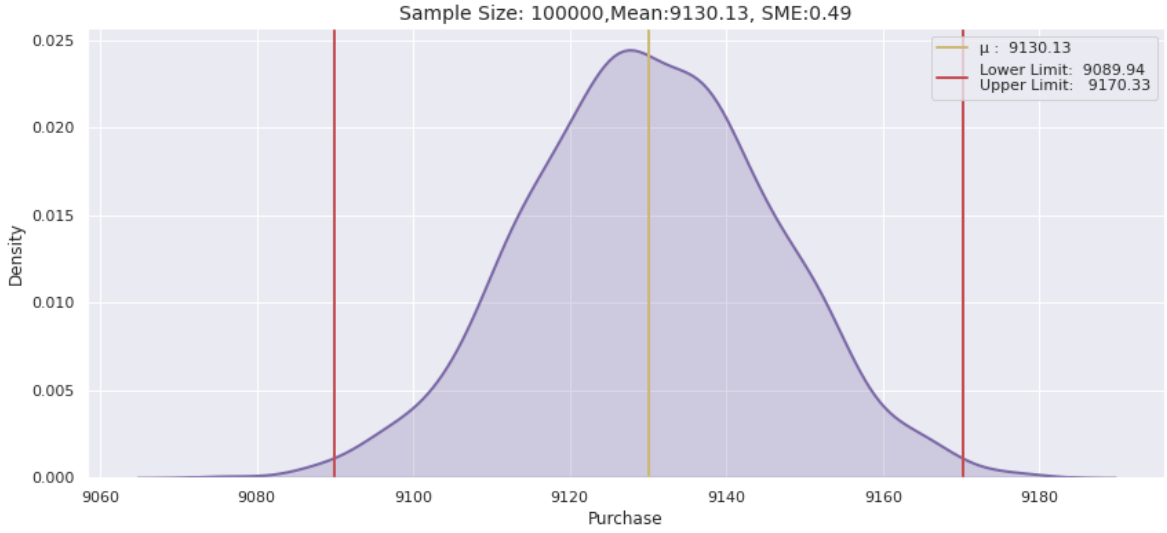
```
itr_size = 1000
smp_size = 1000
ci = 0.99
age_list =['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

array = np.empty((0,8))

for age in age_list:
    mean, ll_m, ul_m = bootstrapping_age(retail_data_v1[retail_data_v1['Age'] ==

    array = np.append(array, np.array([[age,np.round(mean,2), ll_m, ul_m, smp_si

age_data = pd.DataFrame(array, columns = ['Age_Group','Mean','Lower_limit','Uppe
```

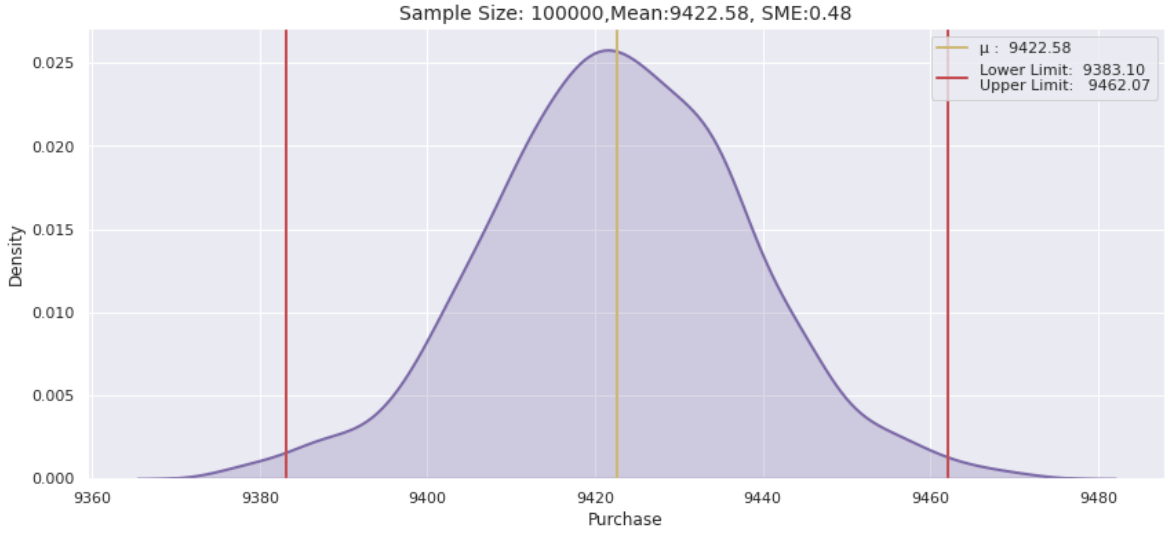

Sample Size: 100000,Mean:8867.83, SME:0.5
μ : 8867.83
Lower Limit: 8827.44
Upper Limit: 8908.22
Density
0.025
0.020
0.015
0.010
0.005
0.000
8800
8820
8840
8860
8880
8900
8920
Purchase

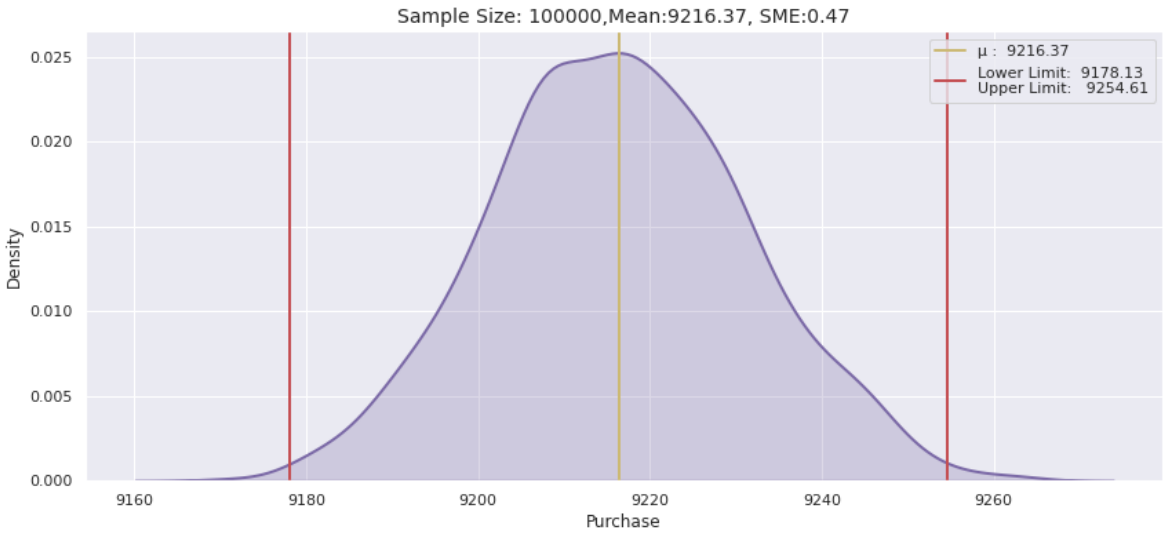

Sample Size: 100000,Mean:9124.53, SME:0.49
μ : 9124.53
Lower Limit: 9084.83
Upper Limit: 9164.23
Density
0.025
0.020
0.015
0.010
0.005
0.000
9060
9080
9100
9120
9140
9160
9180
Purchase


Sample Size: 100000,Mean:9192.96, SME:0.49
μ : 9192.96
Lower Limit: 9153.09
Upper Limit: 9232.82
Density
0.025
0.020
0.015
0.010
0.005
0.000
9140
9160
9180
9200
9220
9240
Purchase


Sample Size: 100000,Mean:9254.65, SME:0.49
μ : 9254.65
Lower Limit: 9214.93
Upper Limit: 9294.38
Density
Purchase


Sample Size: 100000,Mean:9130.13, SME:0.49
μ : 9130.13
Lower Limit: 9089.94
Upper Limit: 9170.33
Density
Purchase


Sample Size: 100000,Mean:9422.58, SME:0.48
μ : 9422.58
Lower Limit: 9383.10
Upper Limit: 9462.07
Density
Purchase



In [348…

```
age_data.head(7)
```

Out[348]:

	Age_Group	Mean	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confi
0	0-17	[8881.87, 8871.02, 8870.77, 8886.59, 8851.45, ...	8827.44	8908.22	100000	[8827.44, 8908.22]	80.78	
1	18-25	[9104.03, 9110.74, 9122.22, 9131.9, 9118.8, 91...	9084.83	9164.23	100000	[9084.83, 9164.23]	79.4	
2	26-35	[9193.53, 9186.68, 9192.08, 9181.67, 9202.61, ...	9153.09	9232.82	100000	[9153.09, 9232.82]	79.73	
3	36-45	[9213.47, 9235.72, 9256.42, 9268.28, 9250.08, ...	9214.93	9294.38	100000	[9214.93, 9294.38]	79.45	
4	46-50	[9145.36, 9137.58, 9133.21, 9142.84, 9124.32, ...	9089.94	9170.33	100000	[9089.94, 9170.33]	80.39	
5	51-55	[9399.12, 9403.97, 9441.47, 9400.14, 9407.91, ...	9383.1	9462.07	100000	[9383.1, 9462.07]	78.97	
6	55+	[9202.62, 9231.93, 9193.2, 9228.07, 9189.13, 9...	9178.13	9254.61	100000	[9178.13, 9254.61]	76.48	

Checking the Sampling distribution of a sample mean for each Age Group

In [349…

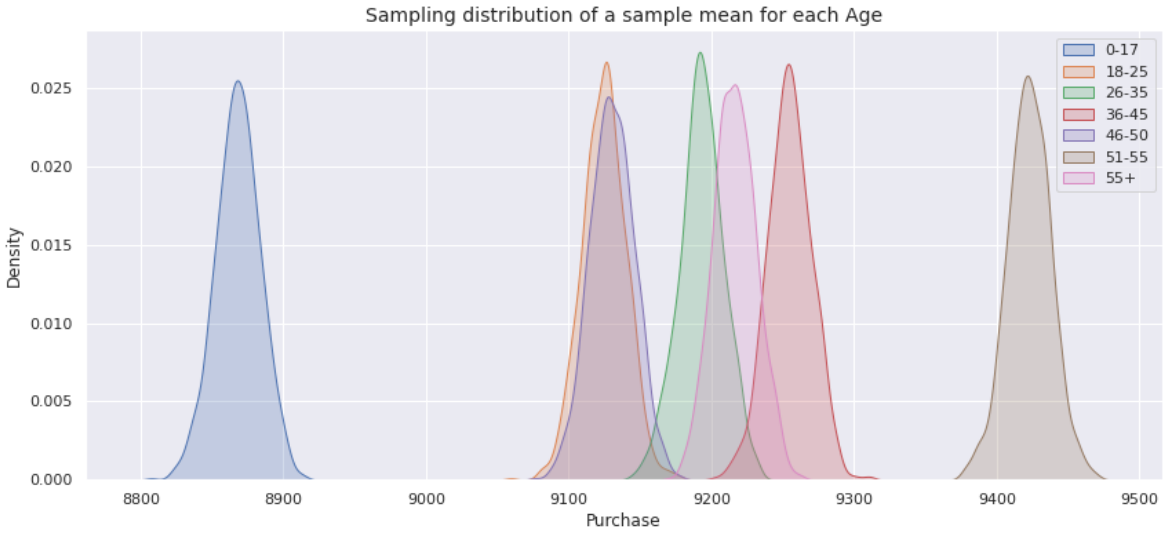
```
age_dict = {}
age_list = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
```

```
for i in range(len(age_data)):
    age_dict[age_list[i]] = age_data.loc[i, "Mean"]
```

In [350…

```
fig, ax = plt.subplots(figsize=(14,6))
sns.set_style("darkgrid")
for label_val in age_dict.keys():
    sns.kdeplot(age_dict[label_val], shade = True, label = label_val)

plt.title("Sampling distribution of a sample mean for each Age",fontsize=14,fami
plt.xlabel('Purchase')
plt.legend(loc='upper right')
```

Out[350]: <matplotlib.legend.Legend at 0x79cf223c7590>



Inferences

Spending by Age_group 0-17 is low compared to other age groups.

Customers in Age_group 51-55 spend the most between 9381.9 and 9463.7

Insights

~ 80% of the users are between the age 18-50 (40%: 26-35, 18%: 18-25, 20%: 36-45) 75% of the users are Male and 25% are Female 60% Single, 40% Married 35% Staying in the city from 1 year, 18% from 2 years, 17% from 3 years Total of 20 product categories are there

There are 20 differnent types of occupations in the city Most of the users are Male There are 20 different types of Occupation and Product_Category More users belong to B City_Category More users are Single as compare to Married Product_Category - 1, 5, 8, & 11 have highest purchasing frequency.

Average amount spend by Male customers: 925344.40 Average amount spend by Female customers: 712024.39

Confidence Interval by Gender Now using the Central Limit Theorem for the population: Average amount spend by male customers is 9,26,341.86 Average amount spend by female customers is 7,11,704.09

Now we can infer about the population that, 95% of the times: Average amount spend by male customer will lie in between: (895617.83, 955070.97) Average amount spend by female customer will lie in between: (673254.77, 750794.02)

Confidence Interval by Marital_Status Married confidence interval of means: (806668.83, 880384.76) Unmarried confidence interval of means: (848741.18, 912410.38)

Confidence Interval by Age For age 26-35 --> confidence interval of means: (945034.42, 1034284.21)

For age 36-45 --> confidence interval of means: (823347.80, 935983.62)

For age 18-25 --> confidence interval of means: (801632.78, 908093.46)

For age 46-50 --> confidence interval of means: (713505.63, 871591.93)

For age 51-55 --> confidence interval of means: (692392.43, 834009.42)

For age 55+ --> confidence interval of means: (476948.26, 602446.23)

For age 0-17 --> confidence interval of means: (527662.46, 710073.17)

Recommendations

```
Management should come-up with some games in the mall to attract
more younger generation will can help them to increase the sale.

The management should have some offers on kids (0-17 years) in
order to increase sales.

In order to attract more young shoppers, they can offer some
games for the younger generation.
```

Inferences & Recommendations

Based on EDA

```
The majority of our customers come from city category B but
customers come from City category C spent more as mean is 9719.

The majority of users come from City Category C, but more people
from City Category B tend to purchase, which suggests the same
users visit the mall multiple times in City Category B.

Majority of Customers purchase within the 5,000 - 20,000 range.
```

Males clearly purchase more than females. 75% of men and only 25% of women purchase products.

Most mall customers are between the ages of 26 and 35.60% of purchases are made by people between the ages of 26 and 45

City Category B accounts for 42%, City Category C 31%, and City Category A represents 27% of all customer purchases.Purchases are high in city category C

Most mall customers are between the ages of 26 and 35.City category C has more customers between the ages of 18 and 45.

In City Category C, there are slightly more female customers.

Product 5 and 8 is common among females.

Based on Statistical Analysis (using CLT & CI)

As the sample size increases, the two groups start to become distinct. With increasing sample size, Standard error of the mean in the samples decreases. For sample size 100000 is 0.49 with confidence is 90%.

Overlappings are increasing with a confidence interval of 95%. Due to the increasing CI, we consider higher ranges within which the actual population might fall, so that both mean purchase are more likely to fall within the same range.

Using confidence interval 99%, the mean purchase value by gender shows a similar pattern to that found with confidence interval 90% & 95%

For Female (sample size 100000) range for mean purchase with confidence interval 99% is [8634.54, 8707.85]

For Male range for mean purchase with confidence interval 99% is [9328.03, 9409.07]

When the confidence percentage increases, the spread, that is the difference between the upper and lower limits, also increases. For Female Confidence percent as [90,95,99] have difference between the upper & lower limits as [50.46,59,73.31]

Overlapping is evident for married vs single customer spend even when more samples are analyzed, which indicates that customers spend the same regardless of whether they are single or married.

Spending by Age_group 0-17 is low compared to other age groups.

```
Customers in Age_group 51-55 spend the most between 9381.9 and
9463.7
```

Recommendations

```
In light of the fact that females spend less than males on
average, management needs to focus on their specific needs
differently. Adding some additional offers for women can
increase their spending on Black Friday.

Management should come-up with some games in the mall to attract
more younger generation will can help them to increase the sale.

The management should have some offers on kids (0-17 years) in
order to increase sales.

In order to attract more young shoppers, they can offer some
games for the younger generation.
```

Men spent more money than women, So company should focus on retaining the male customers and getting more male customers.

Product_Category - 1, 5, 8, & 11 have highest purchasing frequency. it means these are the products in these categories are liked more by customers. Company can focus on selling more of these products or selling more of the products which are purchased less. Unmarried customers spend more money than married customers, So company should focus on acquisition of Unmarried customers.

Customers in the age 18-45 spend more money than the others, So company should focus on acquisition of customers who are in the age 18-45 Male customers living in City_Category C spend more money than other male customers living in B or C, Selling more products in the City_Category C will help the company increase the revenue.

In [ ]: