In [424]:

```
#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
```

Business Problem

The Management team at Walmart Inc. wants to analyze the customer purchase behavior (specifically, 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? (Assume 50 million customers are male and 50 million are female).

These are steps , I followed in this case study

1) Defining Problem Statement and Analyzing basic metrics.

2) Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), statistical summary.

3) Non-Graphical Analysis: Value counts and unique attributes .

4) Visual Analysis - Univariate & Bivariate with Observations.

```
## For continuous variable(s): Distplot, countplot, histogram for   univariate analysis
## For categorical variable(s): Boxplot
## For correlation: Heatmaps, Pairplots
```

5) Missing Value & Outlier Detection .

6) Business Insights based on Non- Graphical and Visual Analysis .

```
## Comments on the range of attributes
## Comments on the distribution of the variables and relationship between them
## Comments for each univariate and bivariate plot
```

7) Distributions , CLT (Central limit theorem).

8) Answering questions

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

Confidence intervals and distribution of the mean of the expenses by female and male customers.

Are confidence intervals of average male and female spending. overlapping? How can Walmart leverage this conclusion to make changes or improvements?

Results when the same activity is performed for Married vs Unmarried.

Results when the same activity is performed for Age.

9) Final Insights - Illustrate the insights based on exploration and CLT.

Comments on the distribution of the variables and relationship between them

Comments for each univariate and bivariate plots

Comments on different variables when generalizing it for Population

10) Recommendations

Actionable items for business. No technical jargon. No complications. Simple action items that everyone can understand

In [425]:

```
#importing libraries for our purpose
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import statsmodels.api as sm
df=pd.read_csv('Walmart_data.csv')
df.head(20)
```

Out[425]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Yea
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	26-35	20	A	
14	1000006	P00231342	F	51-55	9	A	
15	1000006	P00190242	F	51-55	9	A	
16	1000006	P0096642	F	51-55	9	A	
17	1000006	P00058442	F	51-55	9	A	
18	1000007	P00036842	M	36-45	1	B	

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Yea
19	1000008	P00249542	M	26-35	12	C	

In [426]:

```python
# Checking the length of data
len(df)
```

Out[426]:

```
550068
```

In [427]:

```python
# Shape of data
# Number of rows and columns
print("Number of columns in the dataset: {}".format(df.shape[1]))
print("Number of rows in the data set: {}".format(df.shape[0]))

# Rows and Columns
```

```
Number of columns in the dataset: 10
Number of rows in the data set: 550068
```

In [428]:

```python
df.columns  # there are 10 different columns
```

Out[428]:

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

In [429]:

```python
# Complete information of our dataset
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
```

In [430]:

```
# can check datatypes this way also
df.dtypes
```

Out[430]:

```
User_ID                       int64
Product_ID                   object
Gender                       object
Age                          object
Occupation                    int64
City_Category                object
Stay_In_Current_City_Years   object
Marital_Status                int64
Product_Category              int64
Purchase                      int64
dtype: object
```

In [431]:

```
# Are there any Duplicate valuse?
df.duplicated().sum()
```

Out[431]:

```
0
```

In [ ]:

```
# Great !
#there is no Duplicate data
```

In [432]:

```
# Number of Unique values in our data
for i in df.columns:
    print(i, ":" , df.nunique())
```

```
User_ID : 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
Product_ID : 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
Gender : 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
Age : 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
Occupation : 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
```

```
City_Category : 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
Stay_In_Current_City_Years : 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
Marital_Status : 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
Product_Category : 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
Purchase : 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
```

In [ ]:

```
# Here are some unique values by product id and user id
```

In [433]:

```
# Checking for null values in every column of our data
df.isnull().sum()
```

Out[433]:

```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category              0
Purchase                      0
dtype: int64
```

In [11]:

```
# Great!
# there is no null value
```

In [299]:

```
df["User_ID"].nunique()
```

Out[299]:

5891

In [300]:

```
df["Product_ID"].nunique()
```

Out[300]:

3631

In [301]:

```
df["Gender"].nunique()
```

Out[301]:

2

In [302]:

```
df["Age"].nunique()
```

Out[302]:

7

In [303]:

```
df["Occupation"].nunique()
```

Out[303]:

21

In [304]:

```
df["City_Category"].nunique()
```

Out[304]:

3

In [305]:

```
df["Stay_In_Current_City_Years"].nunique()
```

Out[305]:

5

In [306]:

```
df["Marital_Status"].nunique()
```

Out[306]:

2

In [307]:

```
df["Product_Category"].nunique()
```

Out[307]:

20

In [308]:

```
df["Purchase"].nunique()
```

Out[308]:

18105

Unique values (names) are checked for each Features

In [309]:

```
colname = ['Gender','Age','City_Category','Stay_In_Current_City_Years','Marital_Status
for col in colname:
    print("\nUnique values of ",col," are : ",list(df[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  Occupation  are :  [10, 16, 15, 7, 20, 9, 1, 12, 17, 0, 3,
4, 11, 8, 19, 2, 18, 5, 14, 13, 6]
```

A deep dive into User ID

In [310]:

```
df.groupby(['Gender'])['User_ID'].nunique()
```

Out[310]:

```
Gender
F    1666
M    4225
Name: User_ID, dtype: int64
```

In [311]:

```
print("Females are ", 1666/5891)
print("Males are ", 4225/5891)
```

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

Observation

The percentage of women customers is only 28%

Around 72% of customers are male

In [312]:

```
df.groupby(['Age'])['User_ID'].nunique()
```

Out[312]:

```
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 [313]:

```
df.groupby(['City_Category'])['User_ID'].nunique()
```

Out[313]:

```
City_Category
A    1045
B    1707
C    3139
Name: User_ID, dtype: int64
```

In [314]:

```
df.groupby(['Stay_In_Current_City_Years'])['User_ID'].nunique()
```

Out[314]:

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

In [315]:

```
df.groupby(['Marital_Status'])['User_ID'].nunique()
```

Out[315]:

```
Marital_Status
0    3417
1    2474
Name: User_ID, dtype: int64
```

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

In [316]:

```
df.describe().T
```

Out[316]:

	count	mean	std	min	25%	50%	
User_ID	550068.0	1.003029e+06	1727.591586	1000001.0	1001516.0	1003077.0	10044
Occupation	550068.0	8.076707e+00	6.522660	0.0	2.0	7.0	
Marital_Status	550068.0	4.096530e-01	0.491770	0.0	0.0	0.0	
Product_Category	550068.0	5.404270e+00	3.936211	1.0	1.0	5.0	
Purchase	550068.0	9.263969e+03	5023.065394	12.0	5823.0	8047.0	120

In [317]:

```
df.describe(include=['object','category']).T
```

Out[317]:

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

In [318]:

```
df.groupby(['Gender'])['Purchase'].describe()
```

Out[318]:

	count	mean	std	min	25%	50%	75%	max
Gender								
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 [319]:

```
df.groupby(['Marital_Status'])['Purchase'].describe()
```

Out[319]:

Marital_Status	count	mean	std	min	25%	50%	75%	max
0	324731.0	9265.907619	5027.347859	12.0	5605.0	8044.0	12061.0	23961.0
1	225337.0	9261.174574	5016.897378	12.0	5843.0	8051.0	12042.0	23961.0

In [320]:

```
df.groupby(['Age'])['Purchase'].describe()
```

Out[320]:

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 [321]:

```
df.groupby(['City_Category'])['Purchase'].describe()
```

Out[321]:

City_Category	count	mean	std	min	25%	50%	75%	max
A	147720.0	8911.939216	4892.115238	12.0	5403.0	7931.0	11786.0	23961.0
B	231173.0	9151.300563	4955.496566	12.0	5460.0	8005.0	11986.0	23960.0
C	171175.0	9719.920993	5189.465121	12.0	6031.5	8585.0	13197.0	23961.0

Observation

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.

In [ ]:

```

```

In [ ]:

```

```

Dervied Columns¶

Added 2 new feature from Age

"AgeCategory" - Teens, 20s, 30s and Above 40s

"AgeGroup" - '0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'

removed abnormality from "Age column"

In [465]:

```
# 55+ to 55
df["Age"].replace(['55+'], '55')
```

Out[465]:

```
0          0-17
1          0-17
2          0-17
3          0-17
4            55
          ...
550063    51-55
550064    26-35
550065    26-35
550066       55
550067    46-50
Name: Age, Length: 550068, dtype: object
```

In [466]:

```
# removing the intervals and using bins and labals instead.
df.loc[df["Age"] == "0-17","Age"] = 17
df.loc[df["Age"] == "18-25","Age"] = 20
df.loc[df["Age"] == "26-35","Age"] = 30
df.loc[df["Age"] == "36-45","Age"] = 40
df.loc[df["Age"] == "46-50","Age"] = 40
df.loc[df["Age"] == "51-55","Age"] = 50
df.loc[df["Age"] == "55+","Age"] = 55
```

In [467]:

```
bins = [0,17, 20, 30, 40, 50, 55]
labels =["Kids","Teens","20s","30s", '40s','50s']
df['AgeGroup'] = pd.cut(df['Age'], bins)
df['AgeCategory'] = pd.cut(df['Age'], bins,labels=labels)
```

change Marital_Status into "Single" and "Partnered"

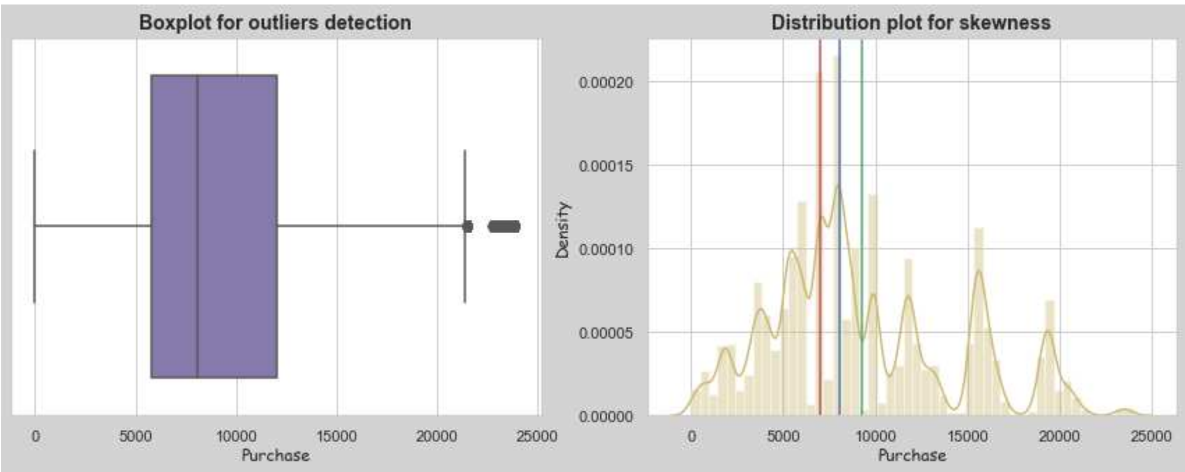
In [36]:

```
# Now Data looking good
```

Outliers Detection and removal(where needed)

In [434]:

```python
# 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(df["Purchase"],color='m')
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(df["Purchase"],color='y')

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(df["Purchase"].mean(),color="g")
plt.axvline(df["Purchase"].median(),color="b")
plt.axvline(df["Purchase"].mode()[0],color="r")

plt.show()
```



Observations

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 [435]:

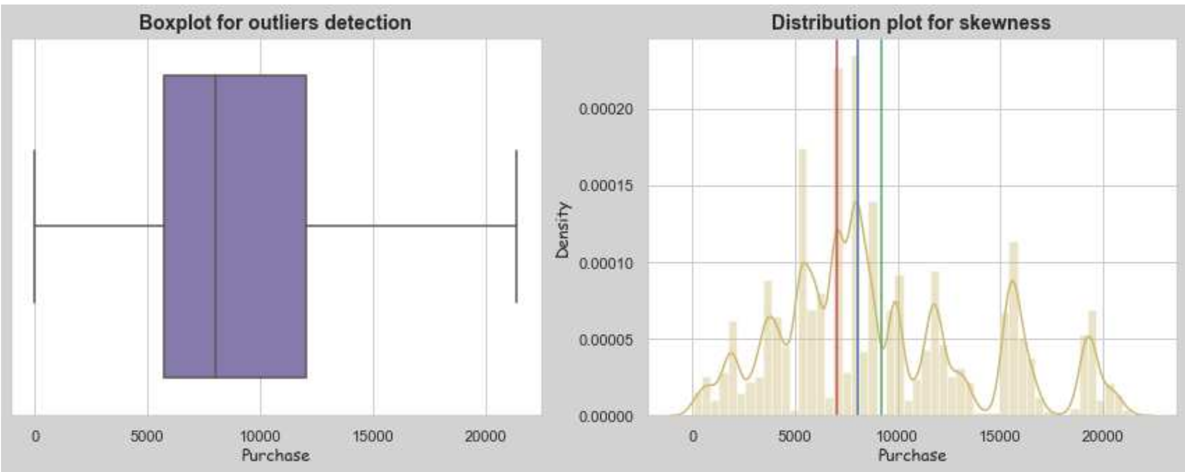
```python
df1 = df.copy()
```

In [436]:

```
#Outlier Treatment: Remove top 5% & bottom 1% of the Column Outlier values
Q3 = df1['Purchase'].quantile(0.75)
Q1 = df1['Purchase'].quantile(0.25)
IQR = Q3-Q1
df1 = df1[(df1['Purchase'] > Q1 - 1.5*IQR) & (df1['Purchase'] < Q3 + 1.5*IQR)]
```

In [437]:

```
# 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(df1["Purchase"],color='m')
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(df1["Purchase"],color='y')

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(df1["Purchase"].mean(),color="g")
plt.axvline(df1["Purchase"].median(),color="b")
plt.axvline(df1["Purchase"].mode()[0],color="r")

plt.show()
```



In [330]:

```
new_data = df.copy()
```

using IQR

In [331]:

```
new_data = df.copy()
new_data['Purchase'].describe()
```

Out[331]:

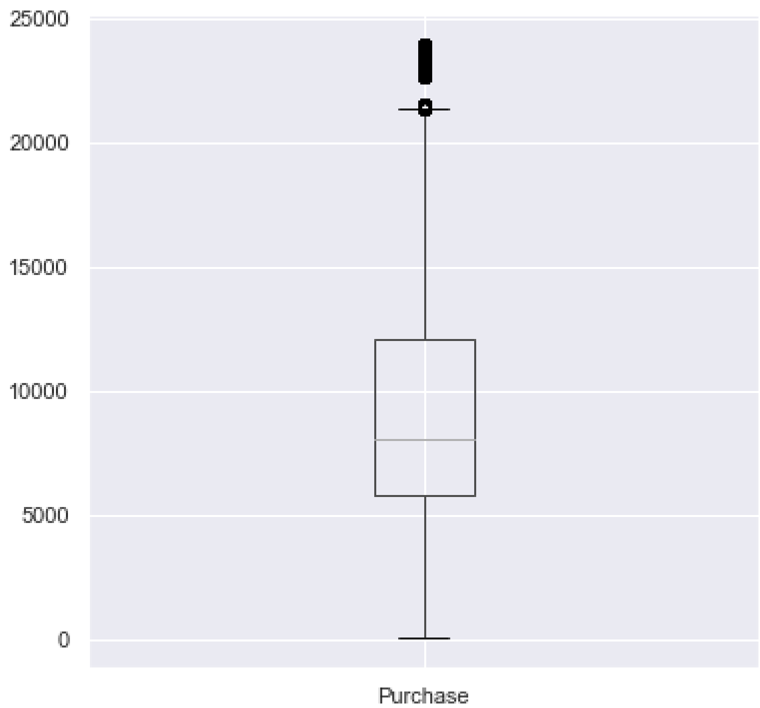
```
count    550068.000000
mean       9263.968713
std        5023.065394
min          12.000000
25%        5823.000000
50%        8047.000000
75%       12054.000000
max       23961.000000
Name: Purchase, dtype: float64
```

In [332]:

```
new_data[["Purchase"]].boxplot()
```

Out[332]:

```
<AxesSubplot:>
```



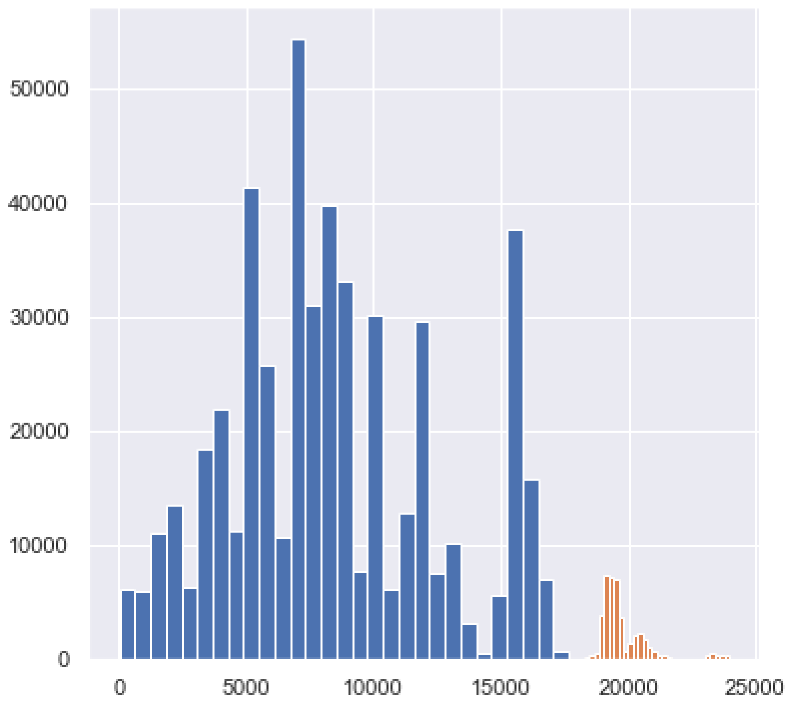
Detection

In [333]:

```
def detect_outliers(d):
    iqr = d.quantile(0.75) - d.quantile(0.25)
    upper = d.quantile(0.75) + 1*iqr
    lower = d.quantile(0.25) - 1*iqr
    return d.loc[(d < lower) | (d > upper)]


ol = detect_outliers(new_data['Purchase'])
new_data.loc[~new_data.index.isin(ol.index)]['Purchase'].hist(bins=30)
ol.hist(bins=30)
```

Out[333]:

<AxesSubplot:>



Removal

In [334]:

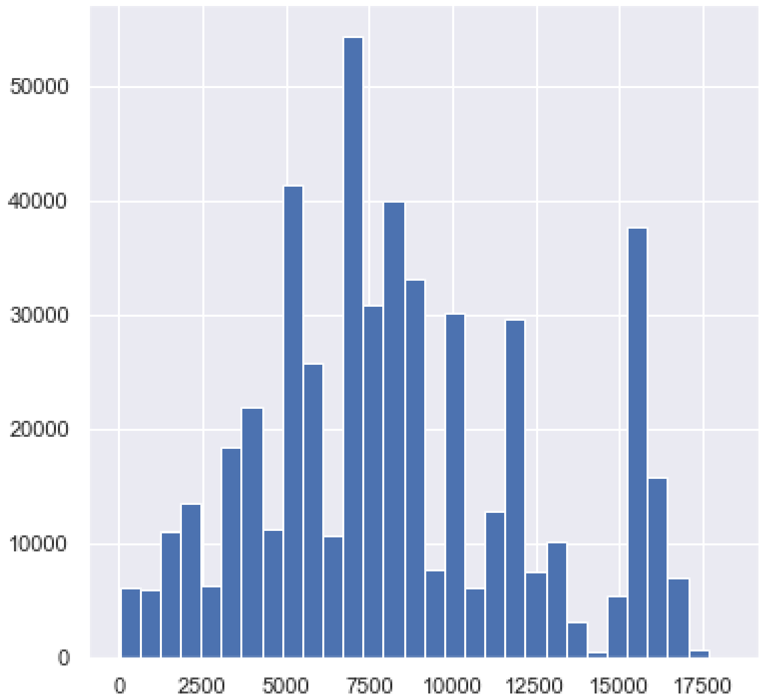
```
# outlier removal

def remove_outliers(d):
    iqr = d.quantile(0.75) - d.quantile(0.25)
    upper = d.quantile(0.75) + 1*iqr
    lower = d.quantile(0.25) - 1*iqr
    return d.loc[(d > lower) & (d < upper)]

x = remove_outliers(new_data['Purchase'])

```

In [335]:

```
x.hist(bins=30)
```

Out[335]:

<AxesSubplot:>



Instead of removing , putting median value so that shape of data will not effact

Examine Data

In [336]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 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  object
 8   Product_Category            550068 non-null  int64
 9   Purchase                    550068 non-null  int64
 10  AgeGroup                    550068 non-null  category
 11  AgeCategory                 550068 non-null  category
dtypes: category(2), int64(4), object(6)
memory usage: 43.0+ MB
```

In [443]:

```
new =  {col : {"4+": "4"} for col in ["Stay_In_Current_City_Years"]}
df.replace(new, inplace=True)
```

In [338]:

```
df.head()
```

Out[338]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	1000001	P00069042	F	17	10	A	2
1	1000001	P00248942	F	17	10	A	2
2	1000001	P00087842	F	17	10	A	2
3	1000001	P00085442	F	17	10	A	2
4	1000002	P00285442	M	55	16	C	4

In [339]:

```
# convert into int
df["Age"] = df["Age"].astype("int64")

```

In [340]:

```
df["Occupation"] = df["Occupation"].astype("int64")
```

In [341]:

```
df["Stay_In_Current_City_Years"] = df["Stay_In_Current_City_Years"].astype("int64")
```

In [342]:

```
df["Product_Category"] = df["Product_Category"].astype("int64")
```

In [343]:

```
#This is to look at what all unique values have . Just trying to use python
list_col=['Gender','Age','Occupation','City_Category','Stay_In_Current_City_Years','Mar
for col in list_col:
    print('{} :{} ' . format(col.upper(),df[col].unique()))
```

```
GENDER :['F' 'M']
AGE :[17 55 30 40 50 20]
OCCUPATION :[10 16 15  7 20  9  1 12 17  0  3  4 11  8 19  2 18  5 14 13  6]
CITY_CATEGORY :['A' 'C' 'B']
STAY_IN_CURRENT_CITY_YEARS :[2 4 3 1 0]
MARITAL_STATUS :['Single' 'Partnered']
PRODUCT_CATEGORY :[ 3  1 12  8  5  4  2  6 14 11 13 15  7 16 18 10 17  9 20
19]
```

In [ ]:

```

```

Observartion

¶

There are male and Female both customers.

There are both Partnered(1) and single(0) customers

Age of customers ranges from 0 to 55

City_category is "A" "B" "C"

Customers are living in the current city in year 0 to 4+.

There are different type of Product category.

Value count for each column

In [344]:

```
for i in df.columns:
    print(i,":", df[i].value_counts())
```

```
User_ID : 1001680    1026
1004277     979
1001941     898
1001181     862
1000889     823
           ...
1002690       7
1002111       7
1005810       7
1004991       7
1000708       6
Name: User_ID, Length: 5891, dtype: int64
Product_ID : P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
             ...
P00314842       1
P00298842       1
P00231642       1
P00204442       1
P00066342       1
Name: Product_ID, Length: 3631, dtype: int64
Gender : M    414259
F    135809
Name: Gender, dtype: int64
Age : 30    219587
40    155714
20     99660
50     38501
55     21504
17     15102
Name: Age, dtype: int64
Occupation : 4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
2     26588
16    25371
6     20355
3     17650
10    12930
5     12177
15    12165
11    11586
19     8461
13     7728
18     6622
9      6291
8      1546
Name: Occupation, dtype: int64
```

```
City_Category : B    231173
C    171175
A    147720
Name: City_Category, dtype: int64
Stay_In_Current_City_Years : 1    193821
2    101838
3     95285
4     84726
0     74398
Name: Stay_In_Current_City_Years, dtype: int64
Marital_Status : Single       324731
Partnered    225337
Name: Marital_Status, dtype: int64
Product_Category : 5     150933
1     140378
8     113925
11     24287
2      23864
6      20466
3      20213
4      11753
16      9828
15      6290
13      5549
10      5125
12      3947
7       3721
18      3125
20      2550
19      1603
14      1523
17       578
9        410
Name: Product_Category, dtype: int64
Purchase : 7011     191
7193     188
6855     187
6891     184
7012     183
        ...
23491      1
18345      1
3372       1
855        1
21489      1
Name: Purchase, Length: 18105, dtype: int64
AgeGroup : (20, 30]    219587
(30, 40]    155714
(17, 20]     99660
(40, 50]     38501
(50, 55]     21504
(0, 17]      15102
Name: AgeGroup, dtype: int64
AgeCategory : 20s      219587
30s      155714
Teens     99660
40s       38501
50s       21504
Kids      15102
Name: AgeCategory, dtype: int64
```

save memory

In [61]:

```
# Observations on shape of data, data types of all the attributes, conversion of catego
# changing it to  object dtype to category  to save memory
df.Product=df["City_Category"].astype("category")
df.Gender=df["Gender"].astype("category")
df.MaritalStatus=df["Marital_Status"].astype("category")
```

```
C:\Users\SHELEN~1\AppData\Local\Temp/ipykernel_7472/3706345722.py:3: UserWar
ning: Pandas doesn't allow columns to be created via a new attribute name -
see https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-acc
ess (https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-ac
cess)
  df.Product=df["City_Category"].astype("category")
C:\Users\SHELEN~1\AppData\Local\Temp/ipykernel_7472/3706345722.py:5: UserWar
ning: Pandas doesn't allow columns to be created via a new attribute name -
 see https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-ac
cess (https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-a
ccess)
  df.MaritalStatus=df["Marital_Status"].astype("category")
```

In [ ]:

```

```

Univariate Analysis

In [438]:

```
def analysis(data):
 # function plots a combined graph for univariate analysis of continous variable
 #to check spread, central tendency , dispersion and outliers
    Name=data.name.upper()
    fig, axes =plt.subplots(1,3,figsize=(17, 7))
    fig.suptitle("SPREAD OF DATA FOR "+ Name  , fontsize=18, fontweight='bold')
    sns.distplot(data,kde=False,color='Blue',ax=axes[0])
    axes[0].axvline(data.mean(), color='y', linestyle='--',linewidth=2)
    axes[0].axvline(data.median(), color='r', linestyle='dashed', linewidth=2)
    axes[0].axvline(data.mode()[0],color='g',linestyle='solid',linewidth=2)
    axes[0].legend({'Mean':data.mean(),'Median':data.median(),'Mode':data.mode()})
    sns.boxplot(x=data,showmeans=True, orient='h',color="purple",ax=axes[1])
    #just exploring violin plot
    sns.violinplot(data,ax=axes[2],showmeans=True)
```

In [80]:

```
df.columns
```

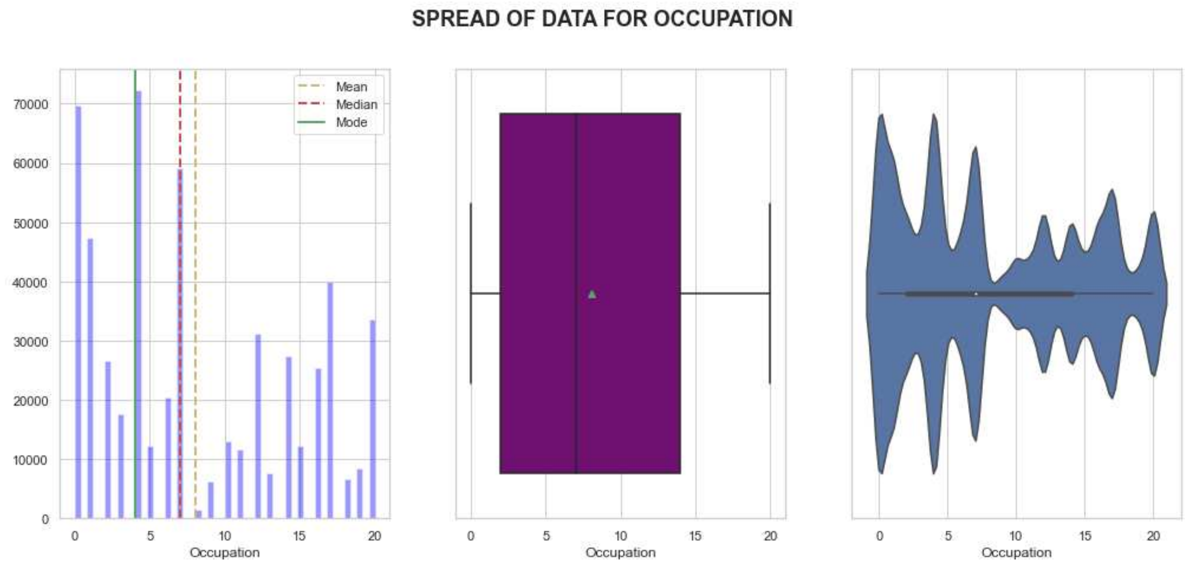
Out[80]:

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

Continuous/Numerical variable (univariate analysis)

In [441]:

```
analysis(df.Occupation)
```



Observation

Occupation has no Outliers.

Occupation is skewed toword right little bit , median is 7.5 ,mean 7.8 around and mode 4.

```
analysis(df.Stay_In_Current_City_Years)
```

Observation

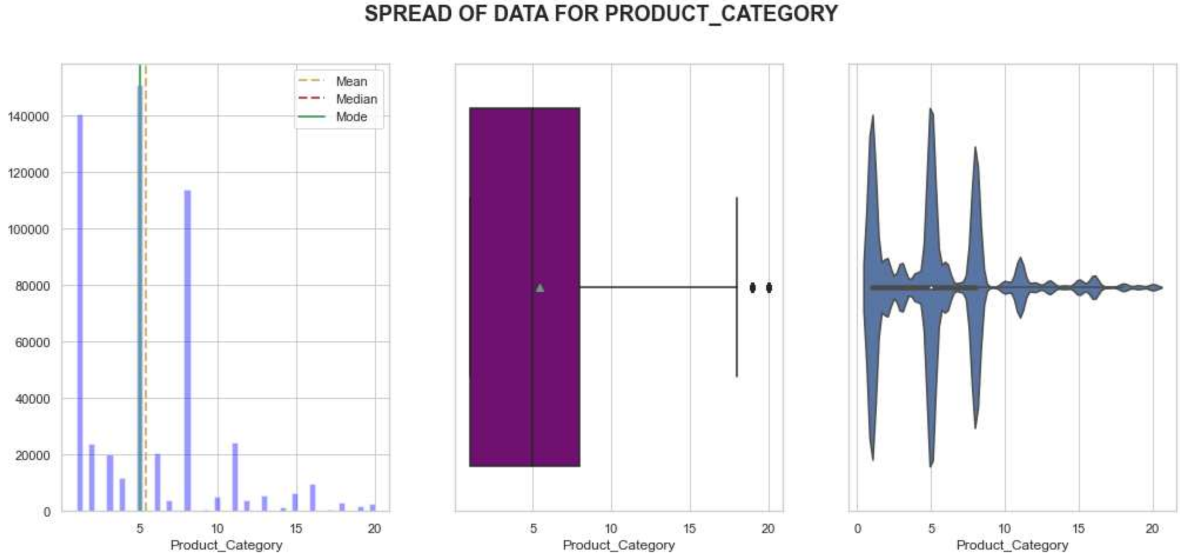
Stay_In_Current_City_Years is skewed towards right .

Customers Purchasing Product are Staying in city 1 to 3 years are more . and Median is 2 , Mean is 1.9 and mode is 1.

Customers Purchasing the Product more is living since 1 year in the city.

In [445]:

```
analysis(df.Product_Category)
```



Observation

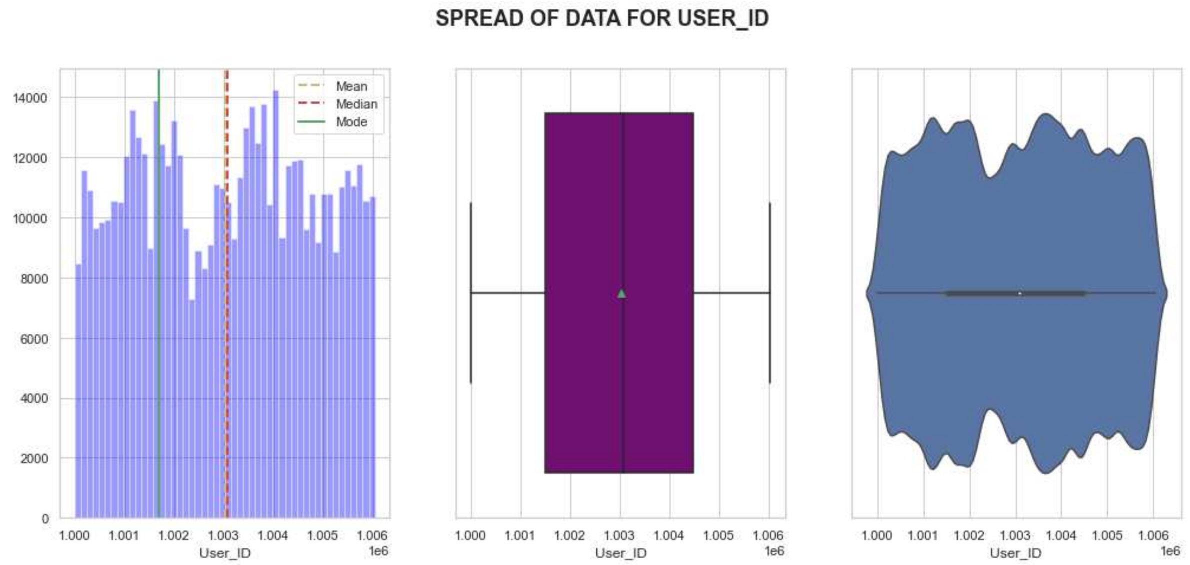
Product_Category is skewed towards right , Median, Mean and Mode are on same almost.

There are some outliers in Product_Category .Some Costumers are purchasing beyond 17.5.

Most of the Product are Costumers Purchasing are 2.5, 5.0 and 7.5.

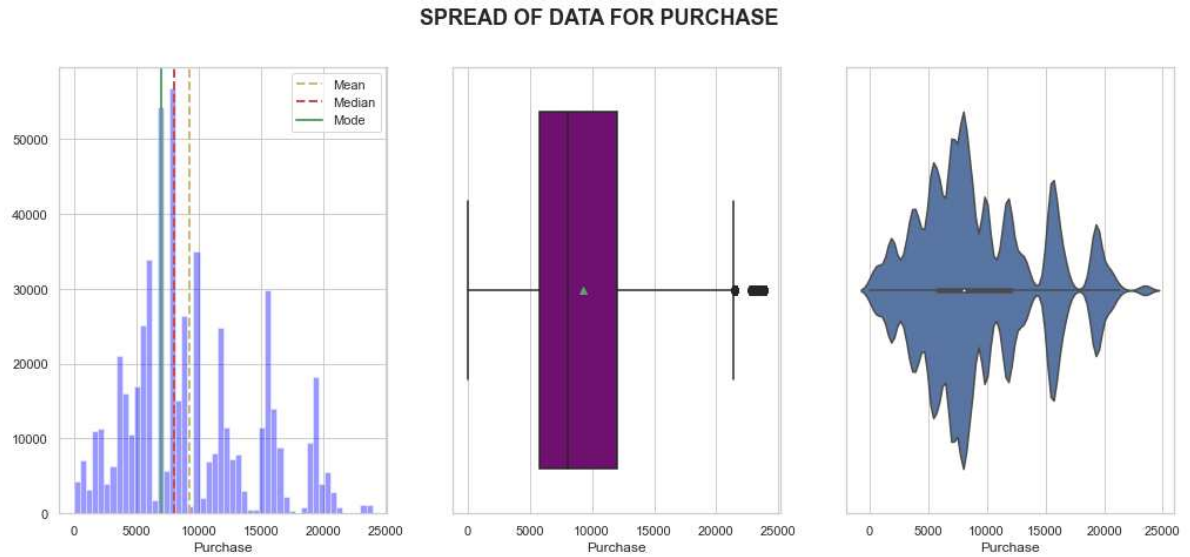
In [446]:

```
analysis(df.User_ID)
```



In [447]:

```
analysis(df.Purchase)
```



Observation

Purchase is skewed towards right , Median is 5500 , Mean is 10000 and Mode is 9000.

Most of the customers are in lower purchasing range and expending less than 20K.

Purchase has some Outliers. few costumers Purchasing beyond 20K.

In [ ]:

```

```

Distplot

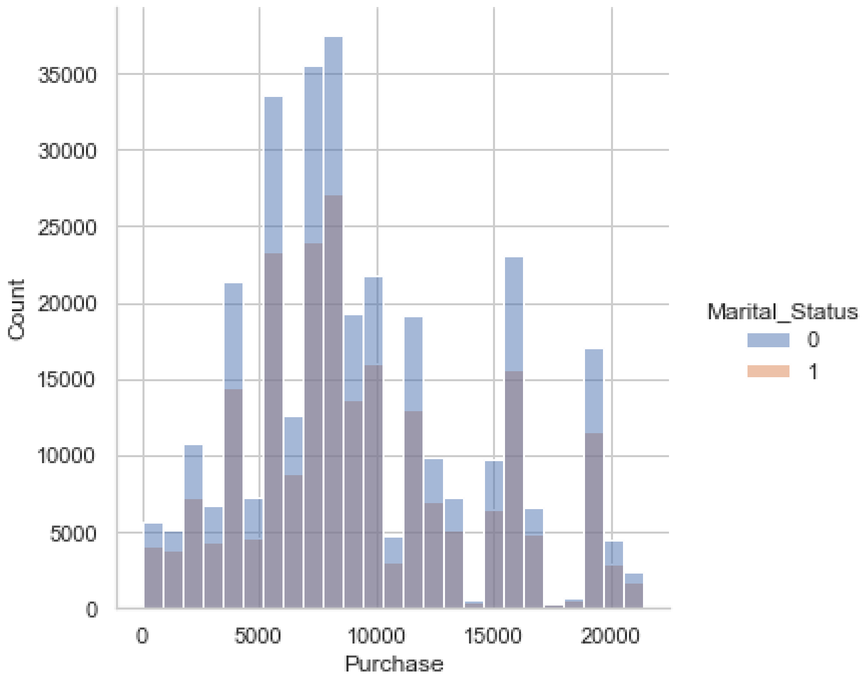
Kernel Density Estimation (KDE) :

is a way to estimate the probability density function of a continuous variable.

The peaks of a Density Plot help display where values are concentrated over the interval.

In [449]:

```
sns.displot(data = df1, x = 'Purchase', hue = 'Marital_Status',bins = 25)
plt.show()
```

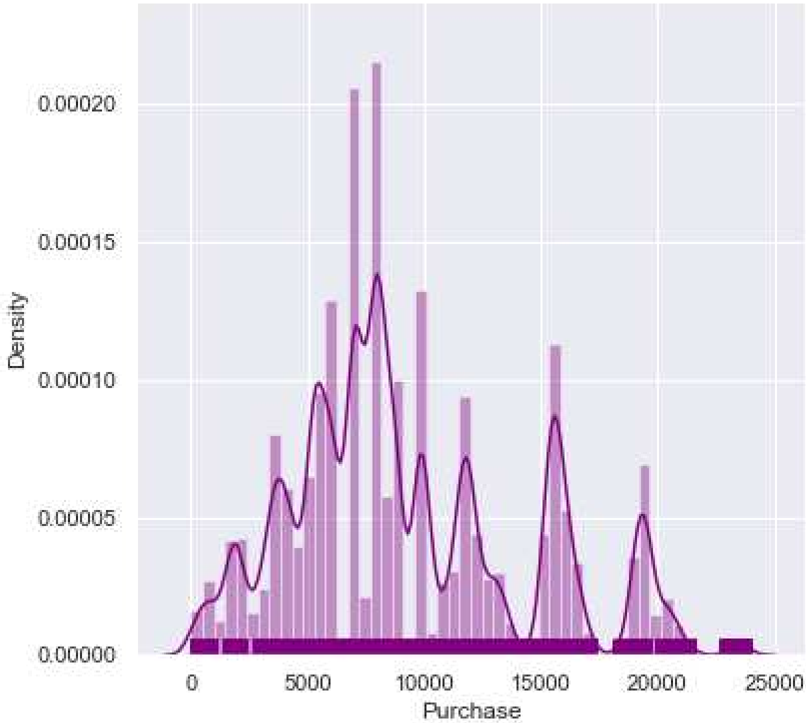


In [451]:

```
import seaborn as sn
sn.set(rc = {'figure.figsize' : (6,6)})
sn.distplot(df["Purchase"],color = "purple",rug = True)
```

Out[451]:

```
<AxesSubplot:xlabel='Purchase', ylabel='Density'>
```



Observation

In this plot as you can see the Vaslues are more concentrated from 0 to 20K.

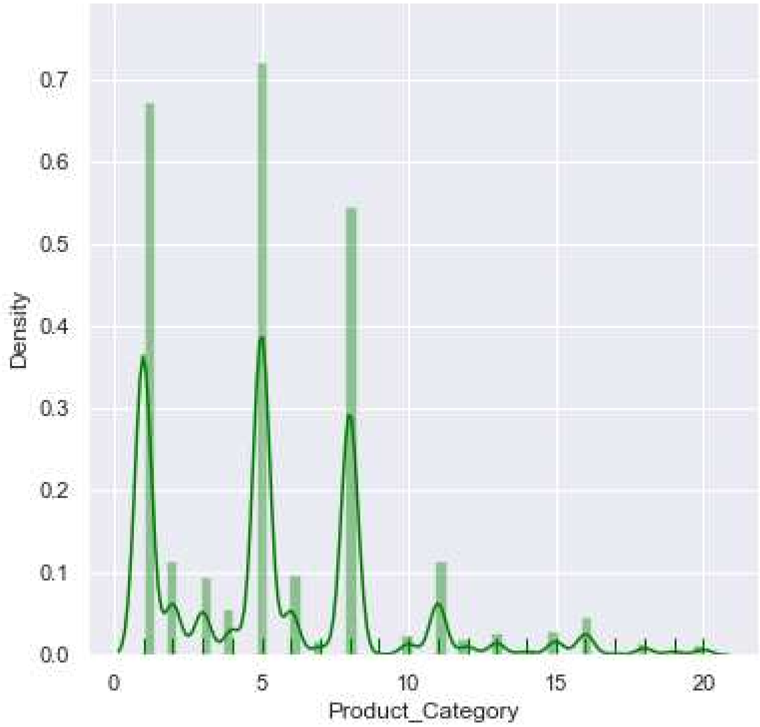
Purchase is highest at 5K to 10K.

In [452]:

```
sn.set(rc = {'figure.figsize' : (6,6)})
sn.distplot(df["Product_Category"],color = "green",rug = True)
```

Out[452]:

<AxesSubplot:xlabel='Product_Category', ylabel='Density'>



Observation

Products which are in Range of 0 to 10 ,having more density

.

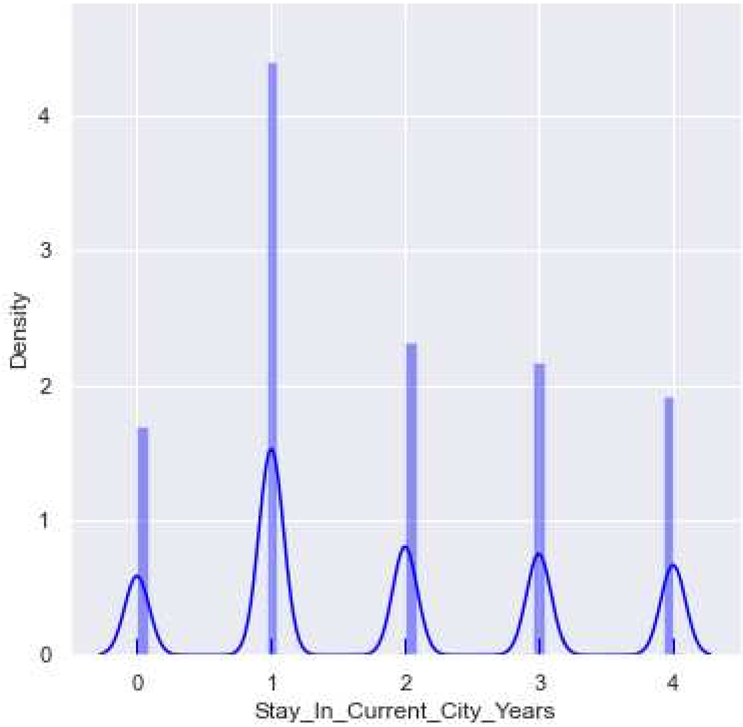
0 to 10 category products are more Purchased by Population

In [453]:

```
sn.set(rc = {'figure.figsize' : (6,6)})
sn.distplot(df["Stay_In_Current_City_Years"],color = "blue",rug = True)
```

Out[453]:

```
<AxesSubplot:xlabel='Stay_In_Current_City_Years', ylabel='Density'>
```



Observation

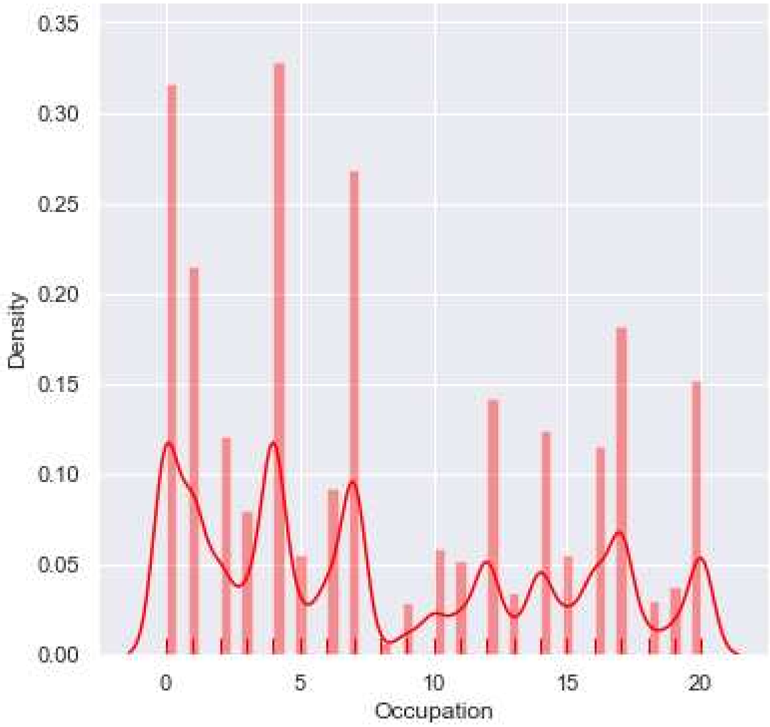
People who are living in The current city since 1 year are having more density

In [454]:

```
sn.set(rc = {'figure.figsize' : (6,6)})
sn.distplot(df["Occupation"],color = "red",rug = True)
```

Out[454]:

```
<AxesSubplot:xlabel='Occupation', ylabel='Density'>
```



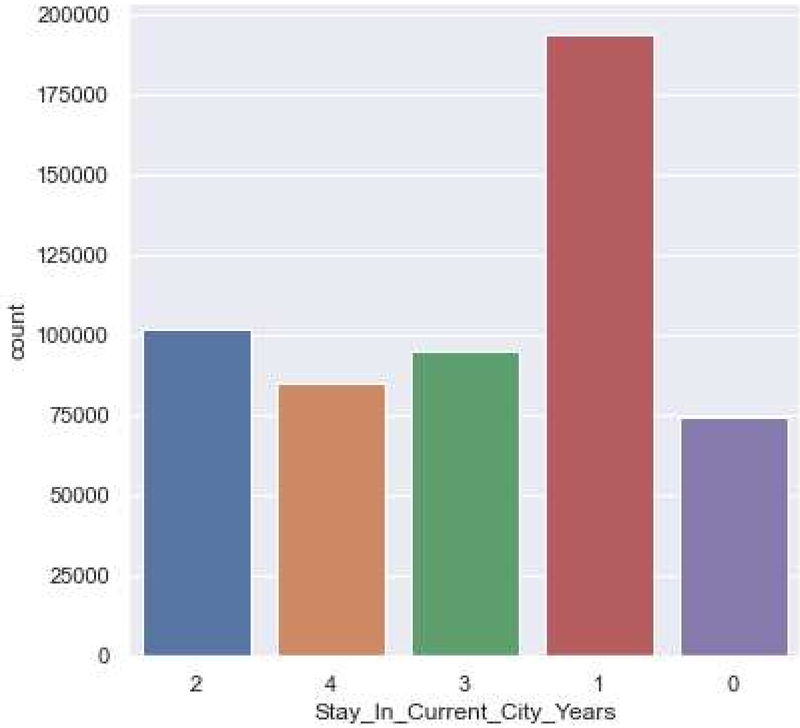
Observation

Occupation of Costumers are having probability density approximately 0.05 to 0.10 .

countplot

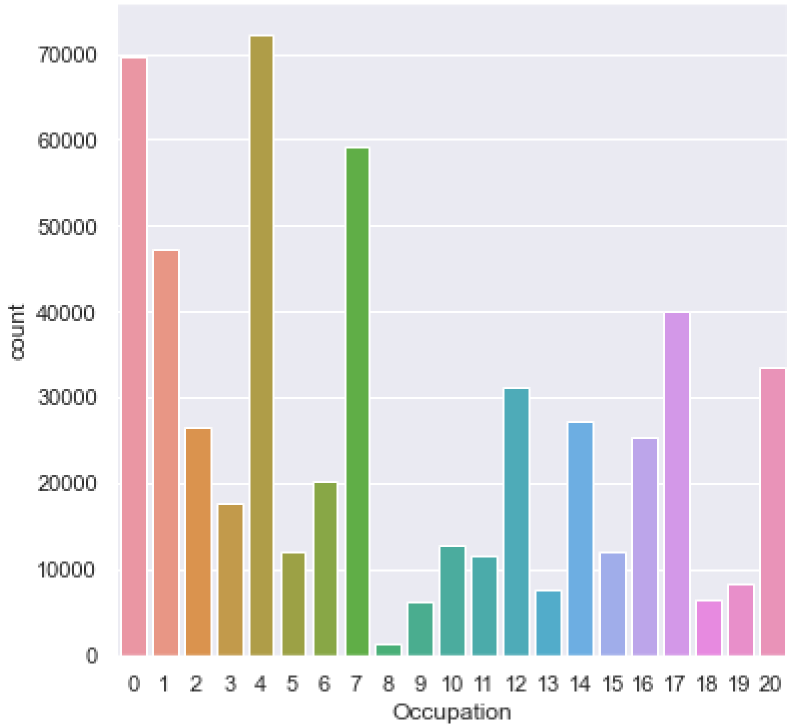
In [455]:

```
ax = sns.countplot(x="Stay_In_Current_City_Years", data=df)
```



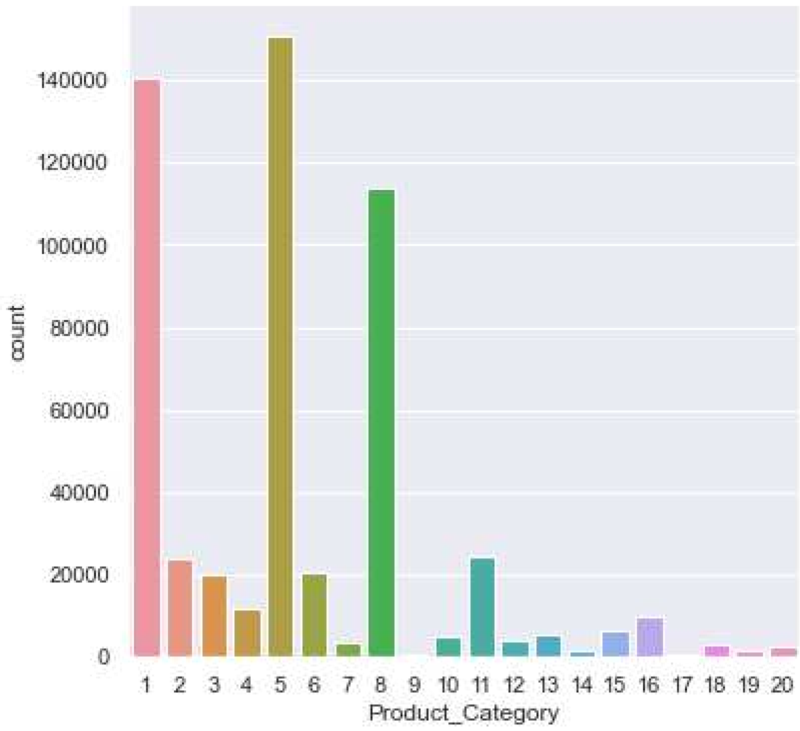
In [456]:

```
ax = sns.countplot(x="Occupation", data=df)
```



In [457]:

```
ax = sns.countplot(x="Product_Category", data=df)
```



Observation

People who are living in The current city since 1 year are having more count approx 20K.

In this plot as you can see the Vaslues are more concentrated from 0 to 20K.

Purchase is highest at 5K to 10K.

Products which are in Range of 0 to 10 ,having more count .

Histogram

In [71]:

```
df.columns
```

Out[71]:

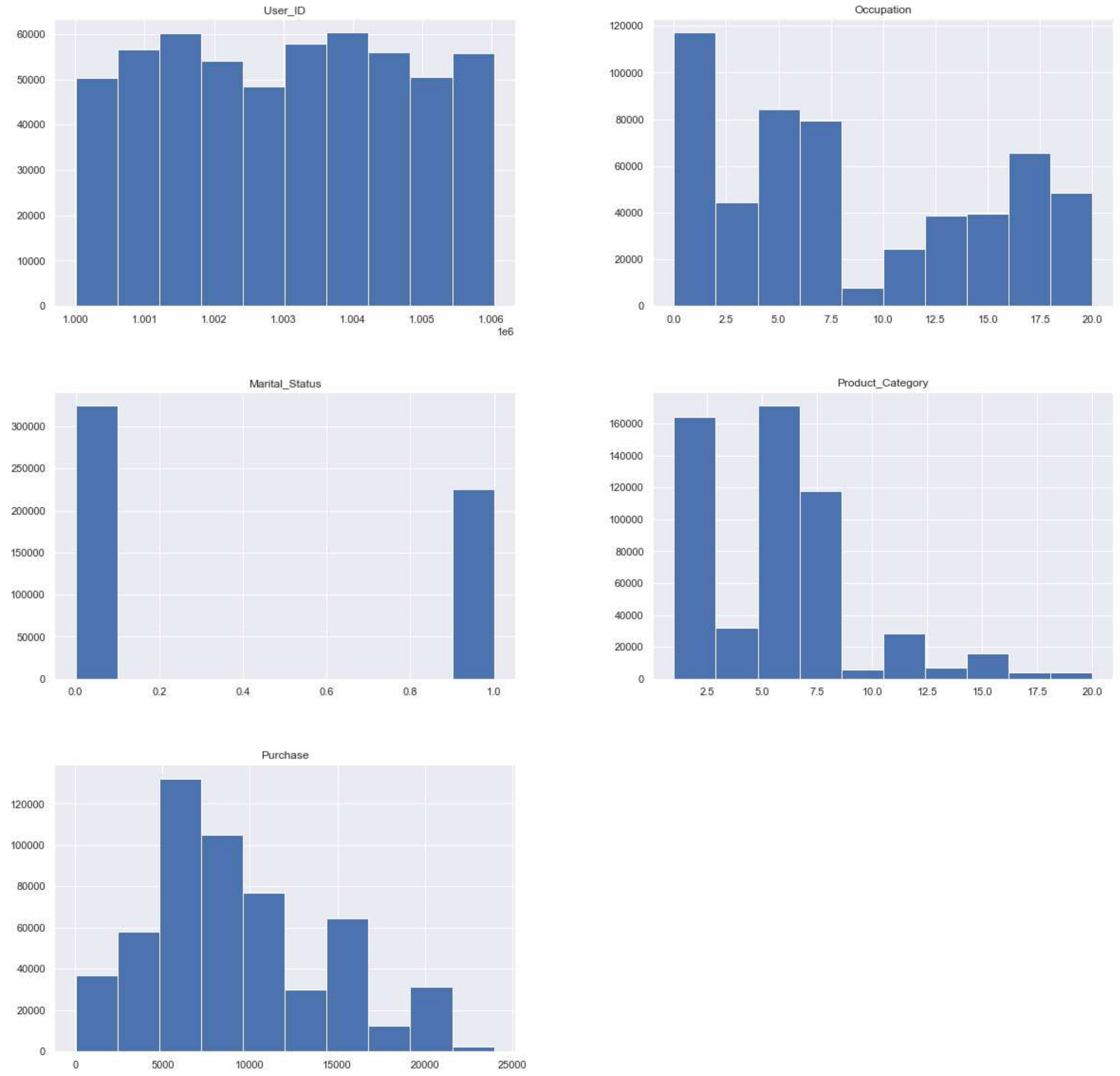
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Categor
y',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase', 'AgeGroup', 'AgeCategory'],
      dtype='object')
```

In [458]:

```python
df.hist(figsize=(20,20))
```

Out[458]:

```
array([[<AxesSubplot:title={'center':'User_ID'}>,
        <AxesSubplot:title={'center':'Occupation'}>],
       [<AxesSubplot:title={'center':'Marital_Status'}>,
        <AxesSubplot:title={'center':'Product_Category'}>],
       [<AxesSubplot:title={'center':'Purchase'}>, <AxesSubplot:>]],
      dtype=object)
```



Categorical variable (univariate analysis)

In [459]:

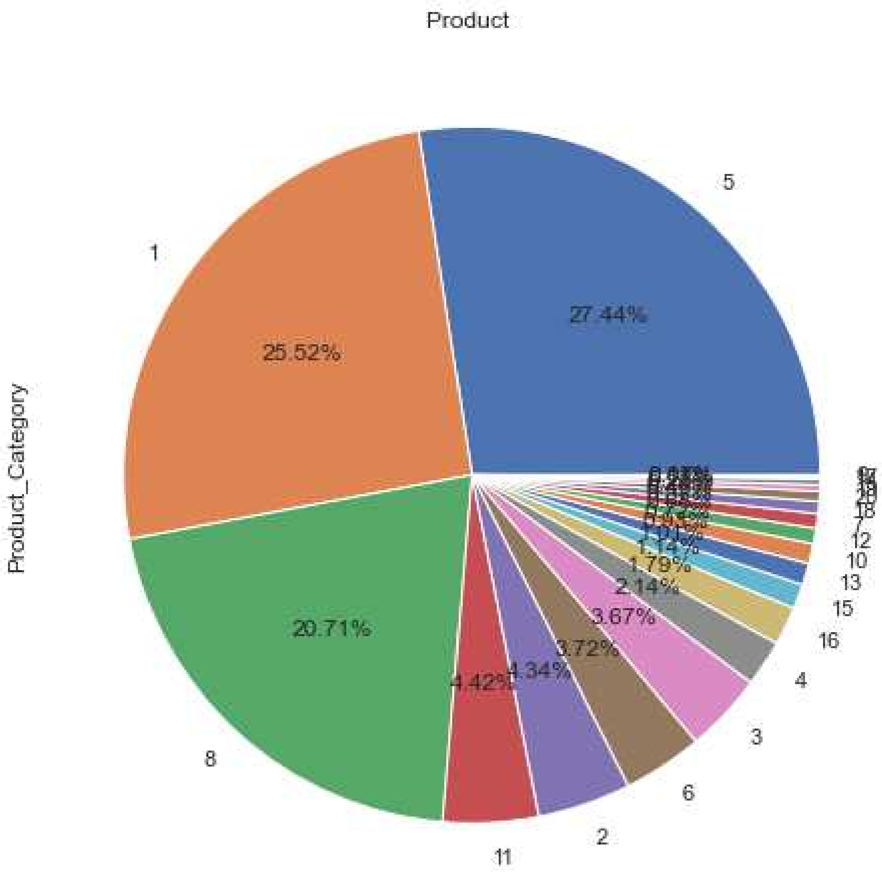
```
df.head()
```

Out[459]:

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

In [460]:

```python
plt.figure(figsize=(10,5))
df['Product_Category'].value_counts().plot.pie(autopct='%1.2f%%',figsize=(8,8))
plt.title("Product ")
plt.show()
```

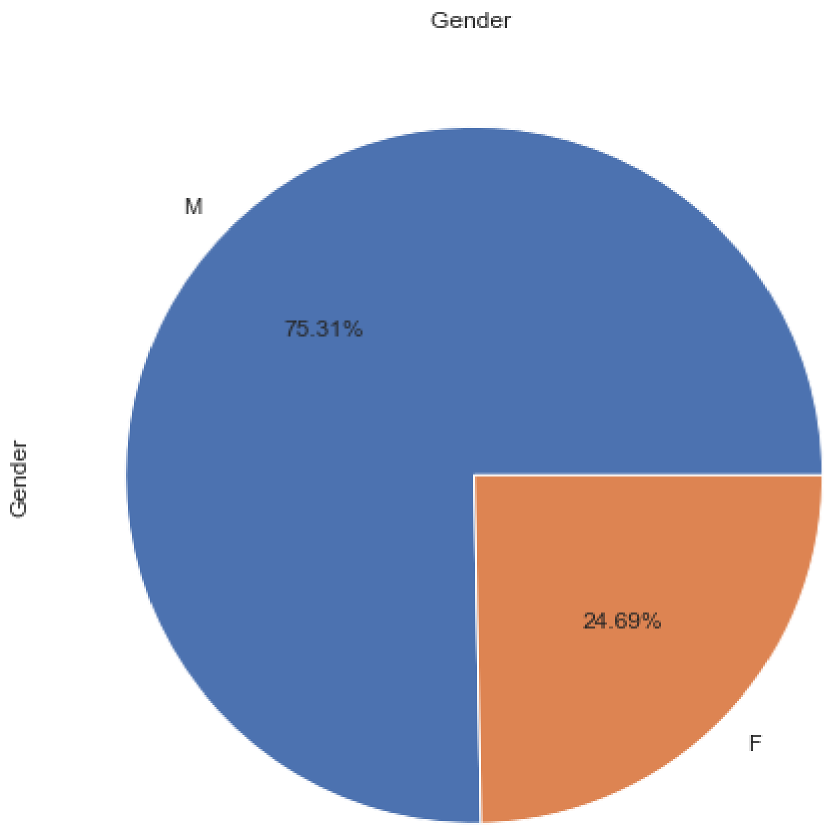


Observation

From this pie chart we can see or observe clearly that Product_Category "1" , "5" , "8" having high count

In [461]:

```
plt.figure(figsize=(10,5))
df['Gender'].value_counts().plot.pie(autopct='%1.2f%%',figsize=(8,8))
plt.title("Gender ")
plt.show()
```



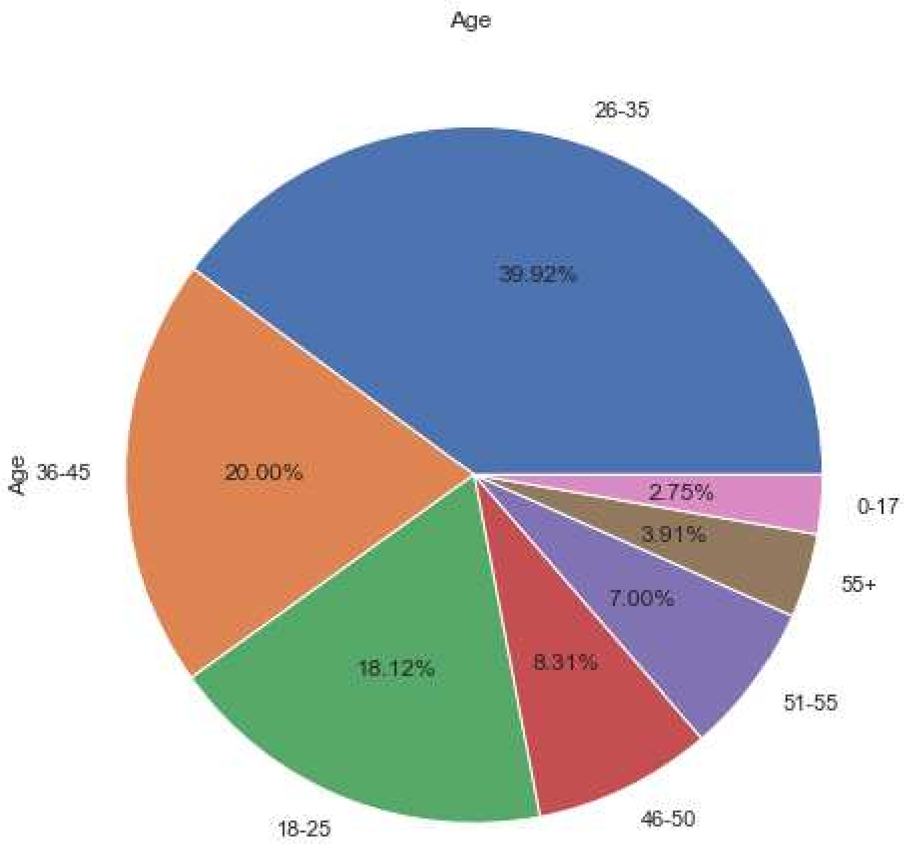
Observation

Female are less as compaired to males

males are 75% and females are 25%

In [462]:

```
plt.figure(figsize=(10,5))
df['Age'].value_counts().plot.pie(autopct='%1.2f%%',figsize=(8,8))
plt.title("Age ")
plt.show()
```



Observation

Value counts of Age 20 to 40 are highest .

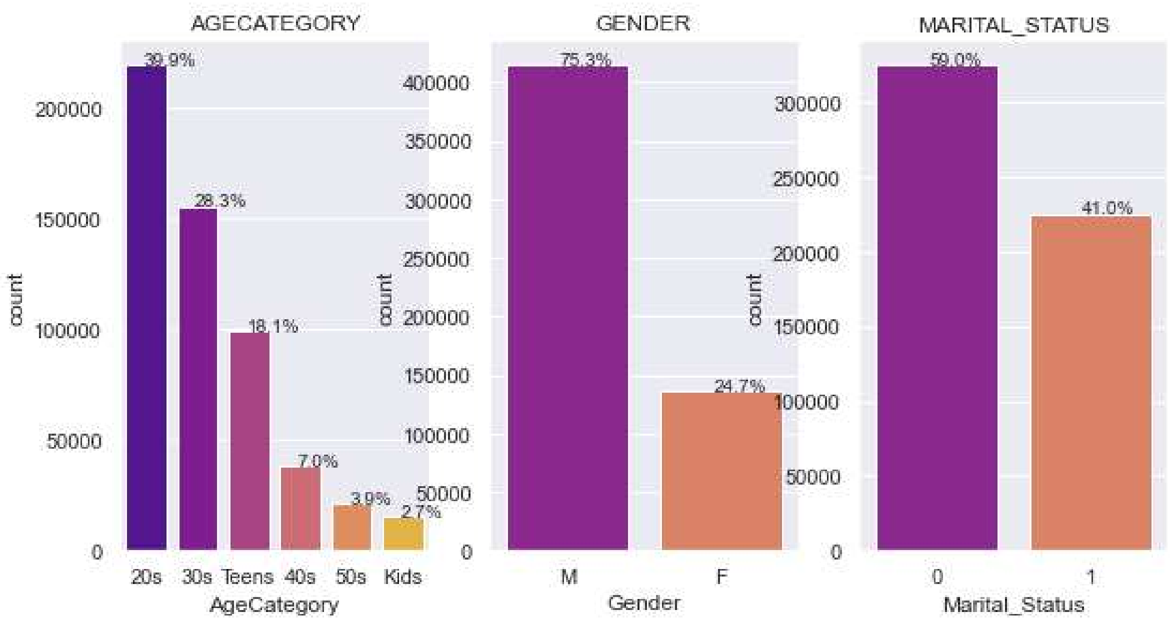
less than 17 and greater than 40 Age people come on Black friday in very less count

In [463]:

```
# Function to create barplots that indicate percentage for each category.
def bar(plot, feature):
    '''
    plot
    feature: 1-d categorical feature array
    '''
    total = len(feature) # length of the column
    for p in plot.patches:
        percentage = '{:.1f}%'.format(100 * p.get_height()/total) # percentage of each
        x = p.get_x() + p.get_width() / 2 - 0.05 # width of the plot
        y = p.get_y() + p.get_height()           # hieght of the plot
        plot.annotate(percentage, (x, y), size = 10) # annotate the percentage
```

In [468]:

```
fig1, axes1 =plt.subplots(1,3,figsize=(10, 5))
list_col=['AgeCategory','Gender','Marital_Status']
j=0
for i in range(len(list_col)):
    order = df[list_col[i]].value_counts(ascending=False).index # to display bar in asc
    axis=sns.countplot(x=list_col[i], data=df , order=order,ax=axes1[i],palette='plasma
    bar(axes1[i],df[list_col[i]])
```



In this bar plot More info added like Percentage

In [469]:

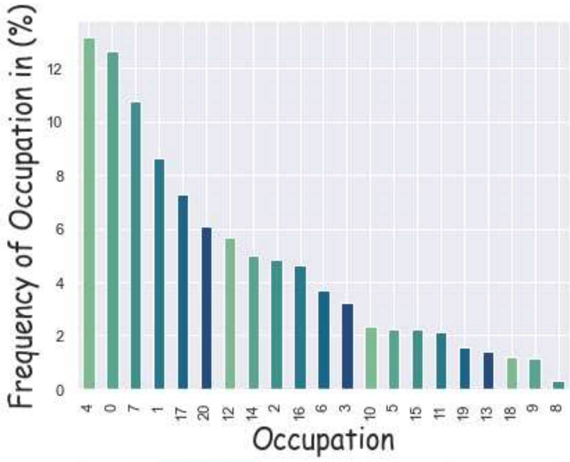
```
# Frequency of each feature in percentage.
def cat_analysis(df, colnames, nrows=2,mcols=2,width=20,height=15, sortbyindex=False):
    fig , ax = plt.subplots(nrows,mcols,figsize=(width,height))
    fig.set_facecolor(color = 'white')
    string = "Frequency of "
    rows = 0
    for colname in colnames:
        count = (df[colname].value_counts(normalize=True)*100)
        string += colname + ' in (%)'
        if sortbyindex:
                count = count.sort_index()
        count.plot.bar(color=sns.color_palette("crest"),ax=ax[rows][0])
        ax[rows][0].set_ylabel(string, fontsize=20,family = "Comic Sans MS")
        ax[rows][0].set_xlabel(colname, fontsize=20,family = "Comic Sans MS")
        count.plot.pie(colors = sns.color_palette("bright"),autopct='%0.0f%%',
                       textprops={'fontsize': 20,'family':"Comic Sans MS"},ax=ax[rows]
        string = "Frequency of "
        rows += 1
```

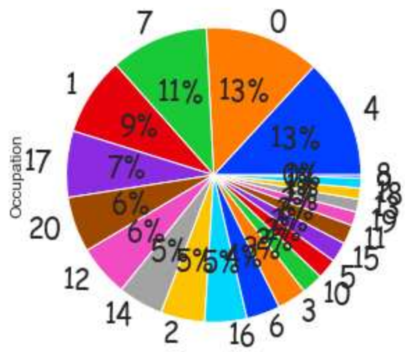
In [ ]:

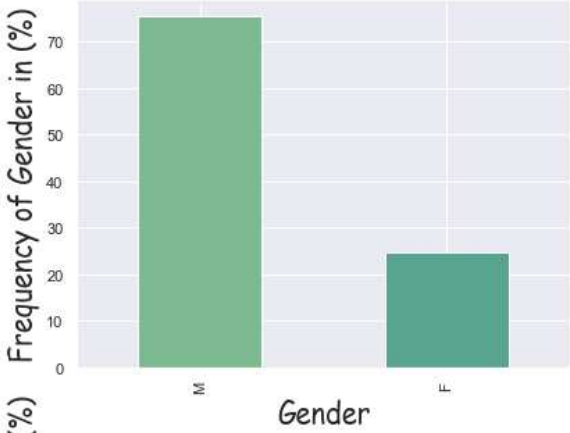
```
df.columns
```

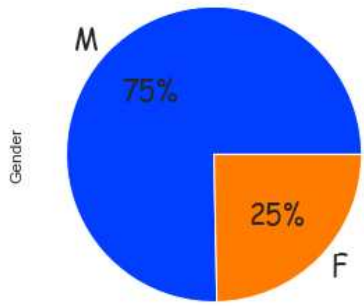
In [470]:

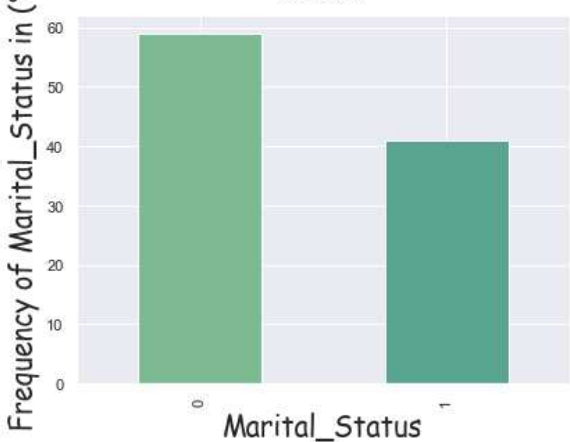
```
cat_colnames = ['Occupation', 'Gender', 'Marital_Status', 'AgeGroup', 'AgeCategory','C
cat_analysis(df,cat_colnames,7,2,14,40)
```

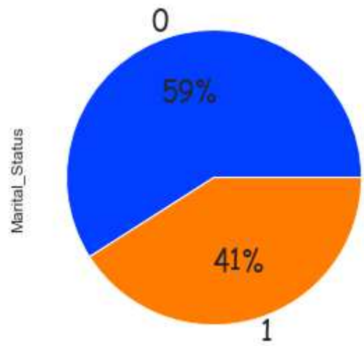


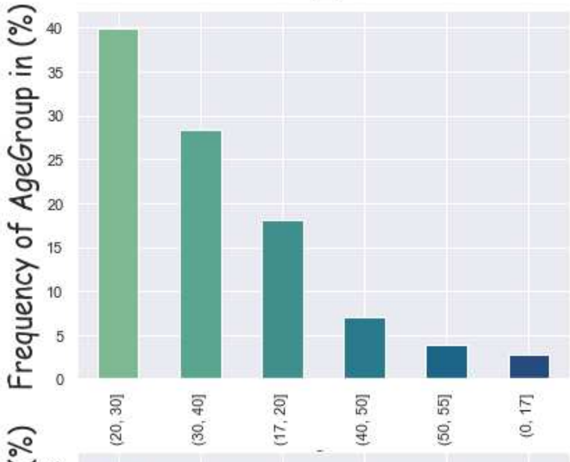


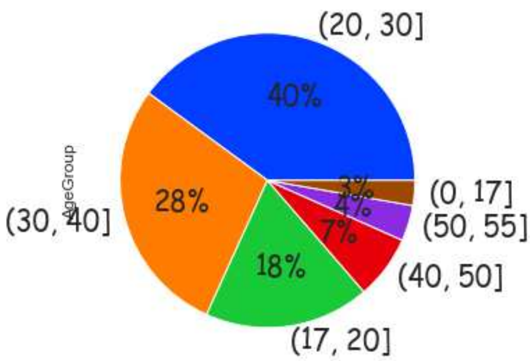


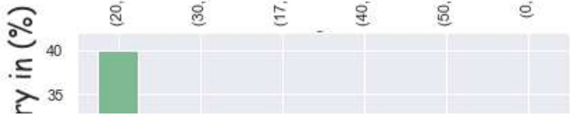


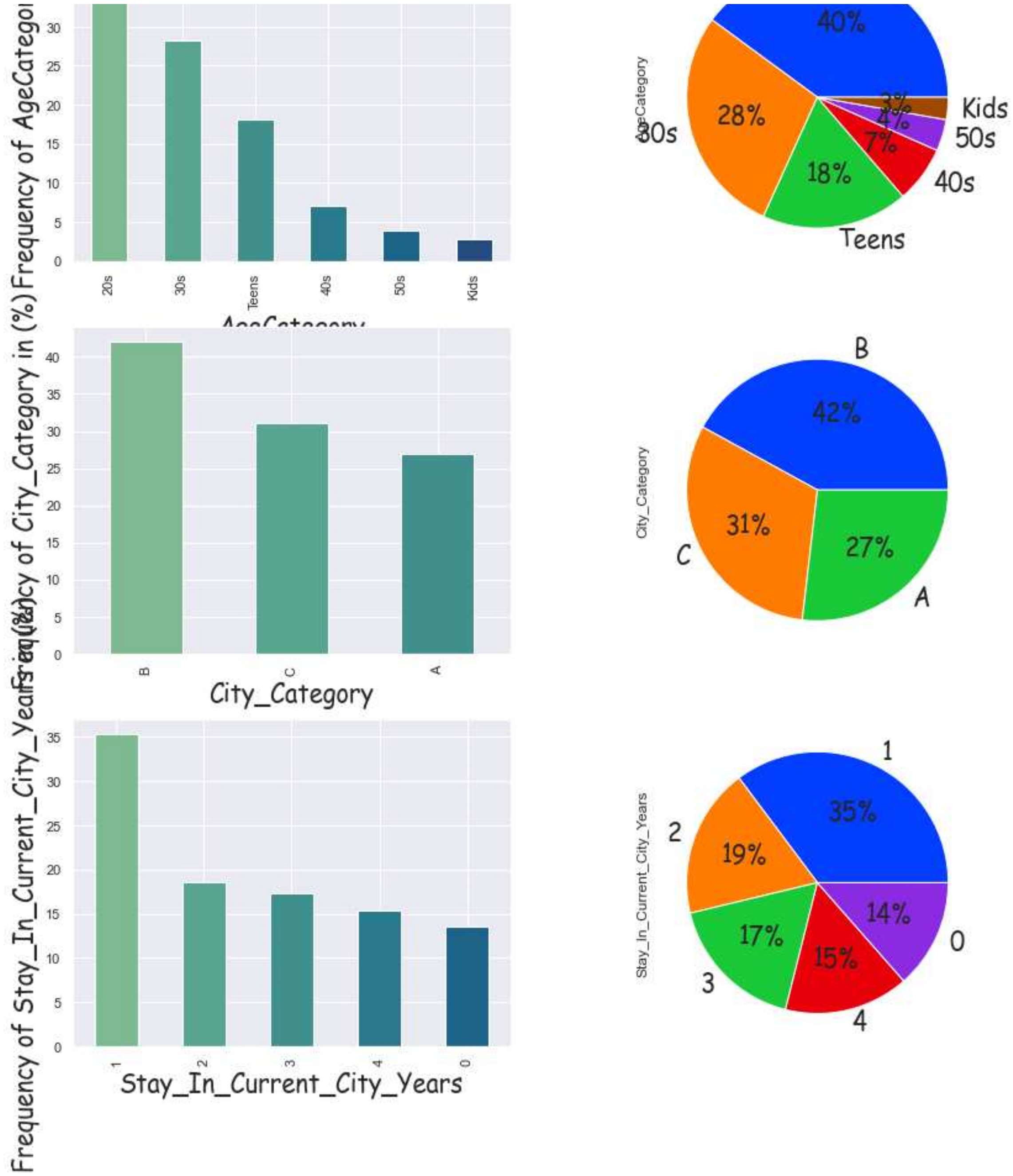












Observation

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.

"B" City_Category People more likely to come

"Single" Costumers are more likely to come on Black Friday

Males are more likely to come

In [ ]:

```
df.info()
```

Bivariate Analysis

In [471]:

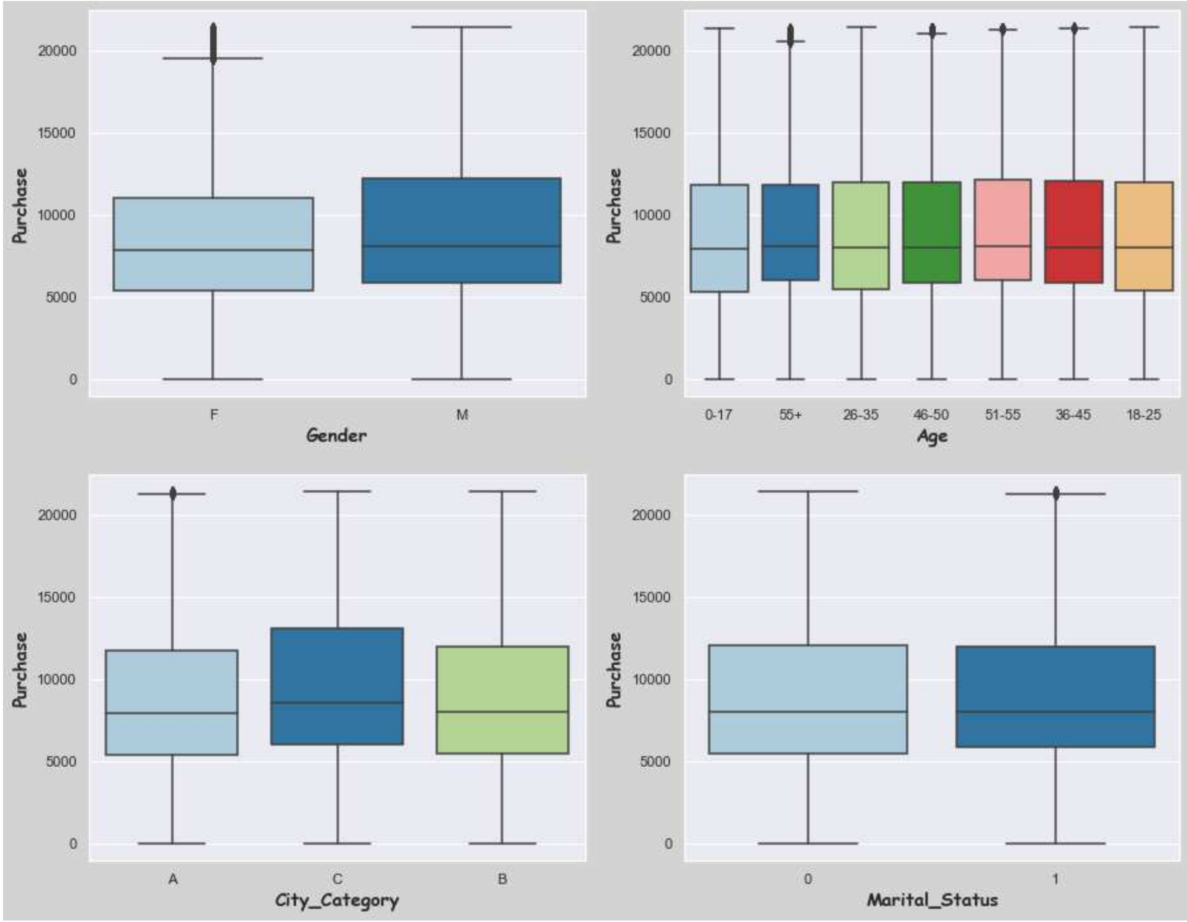
```
def num_cat_bi(df,col_cat,col_num,nrows=1,mcols=2,width=15,height=6):
    fig , ax = plt.subplots(nrows,mcols,figsize=(width,height),squeeze=False)
    sns.set(style='white')
    fig.set_facecolor("lightgrey")
    rows = 0
    i = 0
    while rows < nrows:
        sns.boxplot(x = col_cat[i],y = col_num,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

In [472]:

```
def num_cat_bi_grpby(df,colname,category,groupby,nrows=1,mcols=2,width=18,height=6):
    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],palette=
        sns.lineplot(x=df[category],y=df[var],ax=ax[rows][1],hue=df[groupby],palette="b
        ax[rows][0].set_ylabel(var, fontweight="bold",fontsize=14,family = "Comic Sans
        ax[rows][0].set_xlabel(category,fontweight="bold", fontsize=14,family = "Comic
        ax[rows][0].legend(loc='lower right')
        ax[rows][1].set_ylabel(var, fontweight="bold",fontsize=14,family = "Comic Sans
        ax[rows][1].set_xlabel(category,fontweight="bold", fontsize=14,family = "Comic
        rows += 1
    plt.show()
```

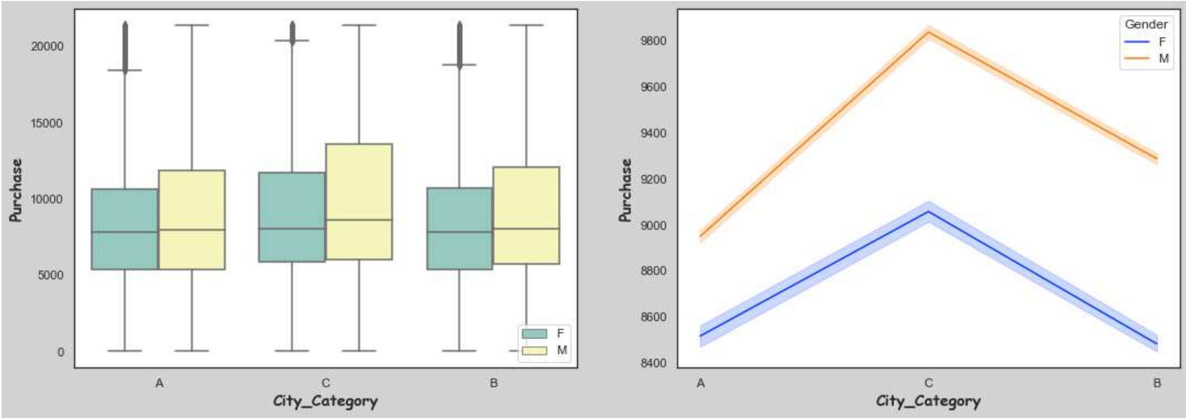
In [473]:

```
col_cat = ['Gender', 'Age','City_Category','Marital_Status']
num_cat_bi(df1,col_cat,'Purchase',2,2,15,12)
```



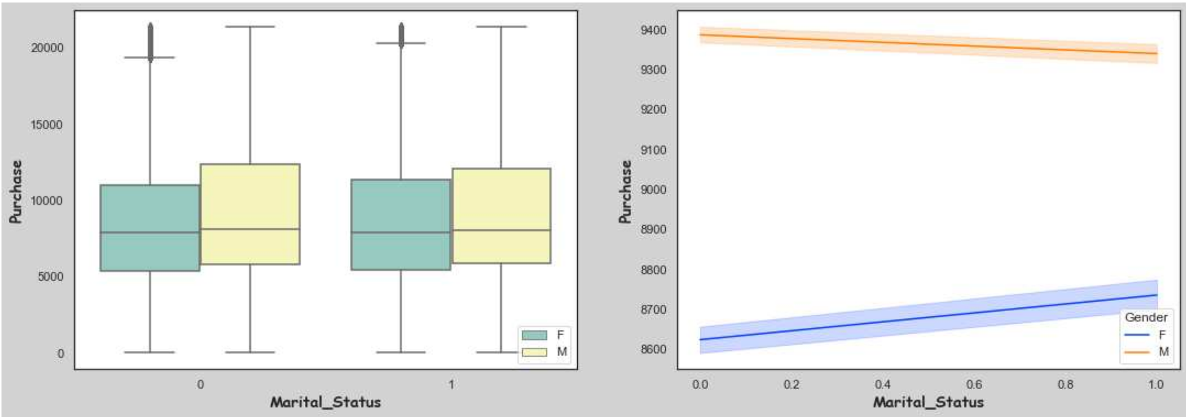
In [474]:

```
col_num = [ 'Purchase']
num_cat_bi_grpby(df1,col_num,"City_Category",'Gender')
```



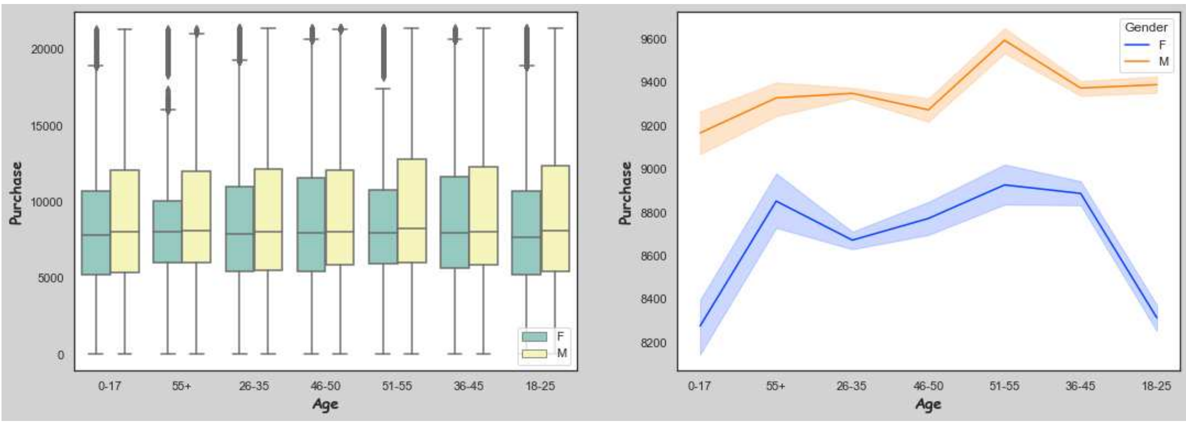
In [475]:

```
col_num = [ 'Purchase']
num_cat_bi_grpby(df1,col_num,"Marital_Status",'Gender')
```



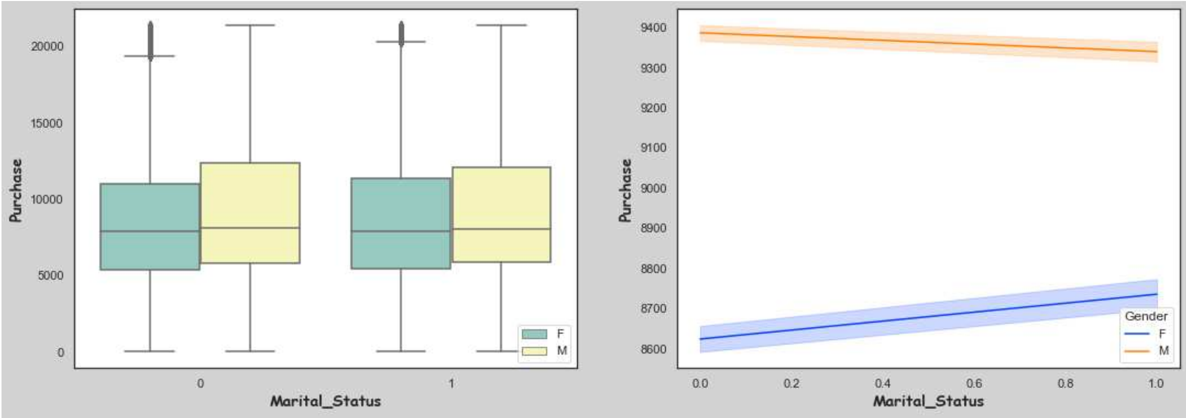
In [476]:

```
col_num = [ 'Purchase']
num_cat_bi_grpby(df1,col_num,"Age",'Gender')
```



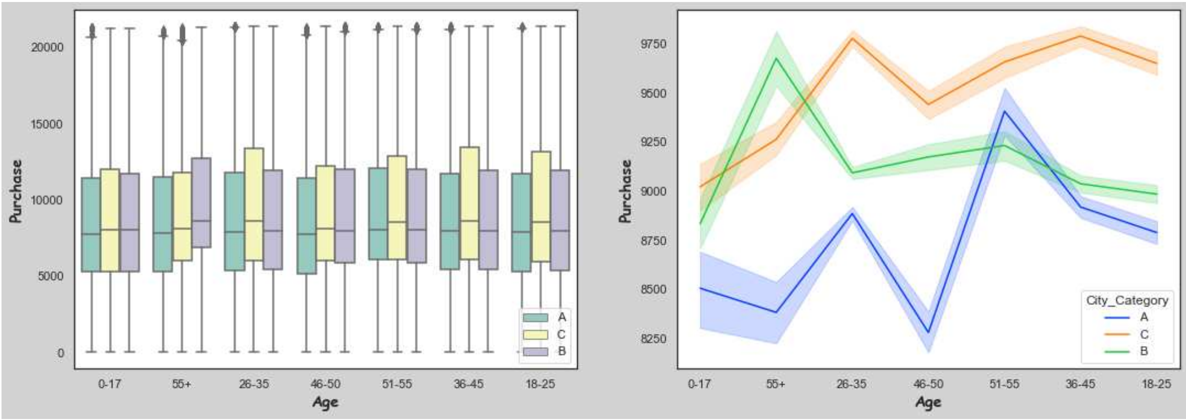
In [477]:

```
col_num = [ 'Purchase']
num_cat_bi_grpby(df1,col_num,"Marital_Status",'Gender')
```



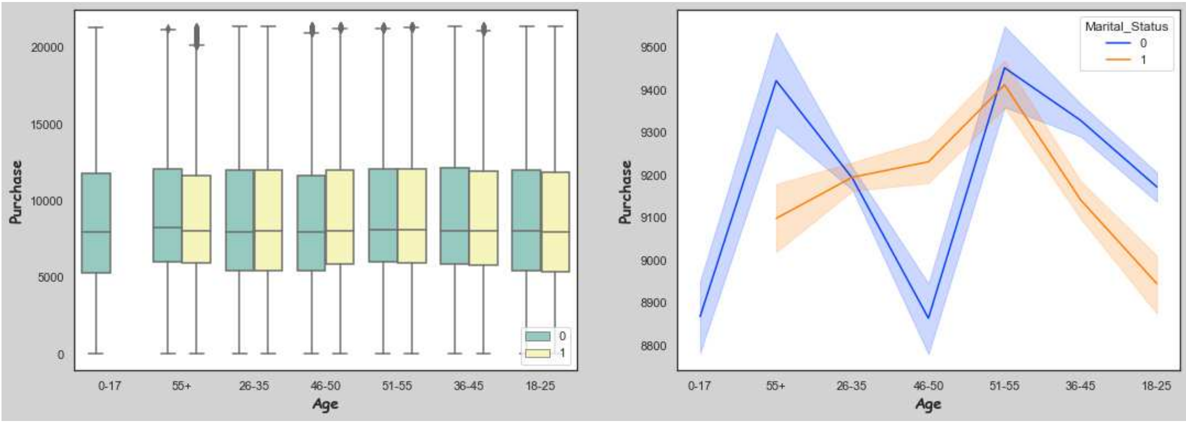
In [478]:

```
col_num = [ 'Purchase']
num_cat_bi_grpby(df1,col_num,"Age",'City_Category')
```



In [479]:

```
col_num = [ 'Purchase']
num_cat_bi_grpby(df1,col_num,"Age",'Marital_Status')
```



Observation

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

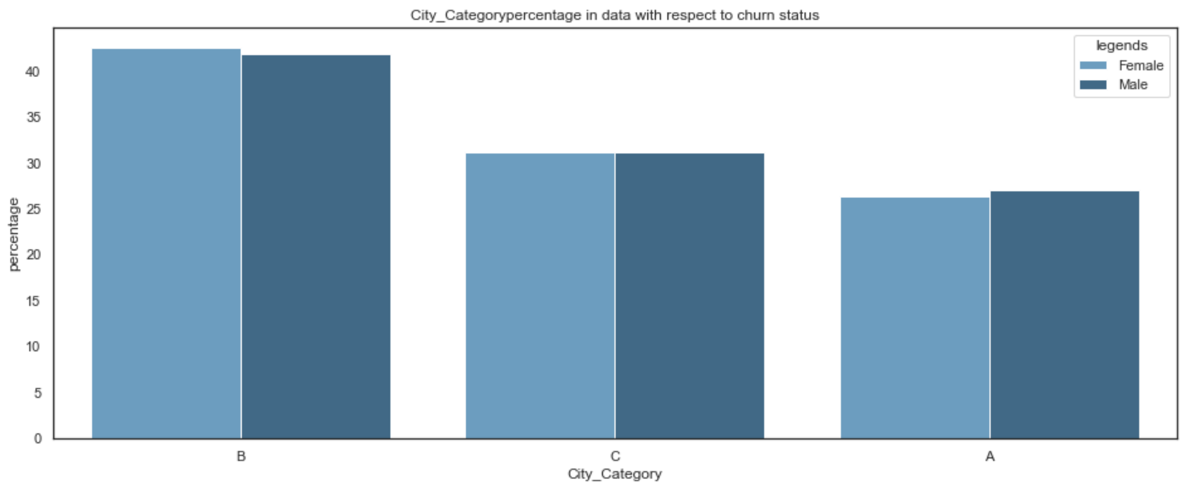
City category C has more customers between the ages of 18 and 45

In [480]:

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

    male = df1[df1["Gender"]=='M'][colname].value_counts().reset_index()
    male["percentage"]  = (male[colname]*100/male[colname].sum())
    male["legends"]        = "Male"


    female = df1[df1["Gender"]=='F'][colname].value_counts().reset_index()
    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="Blues_
    plt.xlabel(colname)
    fig.set_facecolor("white")
    plt.title(colname + "percentage in data with respect to churn status")
    plt.show()
```

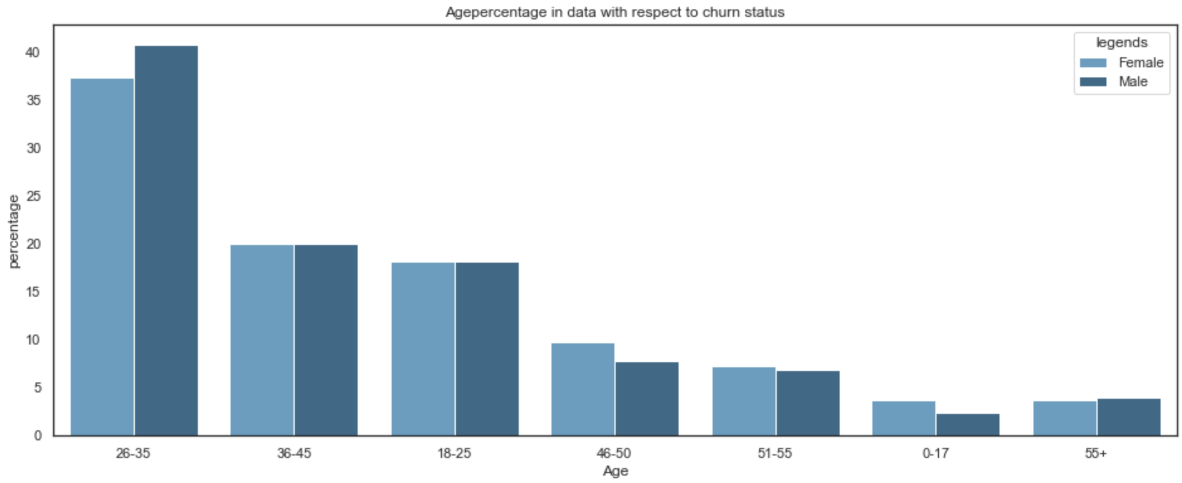
In [481]:

```
bar_M_vs_F('City_Category')
```



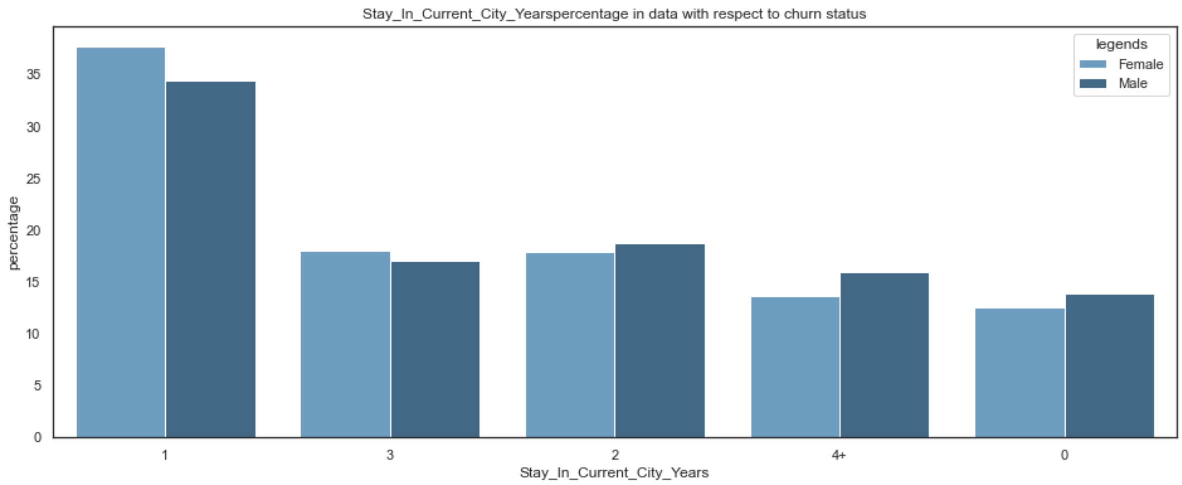
In [482]:

```
bar_M_vs_F('Age')
```



In [483]:

```
bar_M_vs_F('Stay_In_Current_City_Years')
```



Observation

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

In [484]:

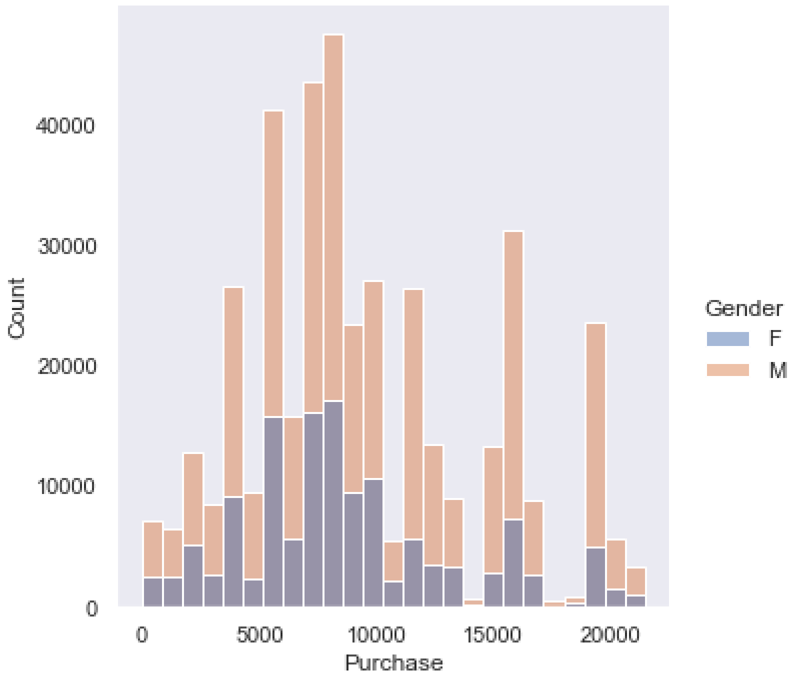
```
print(df1.groupby(['Gender','City_Category'])['User_ID'].count())
```

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

In [485]:

```
fig = plt.figure(figsize=(25,10))
fig.set_facecolor("lightgrey")
sns.set(style='dark')
sns.displot(x= 'Purchase',data=df1,hue='Gender',bins=25)
plt.show()
```

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



Observation

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

In [486]:

```
df1.sample(500,replace=True).groupby(['Gender'])['Purchase'].describe()
```

Out[486]:

Gender	count	mean	std	min	25%	50%	75%	max
F	138.0	9148.76087	5006.290049	14.0	5450.00	8283.5	11899.75	21270.0
M	362.0	9507.59116	4889.926512	243.0	5872.75	8637.0	12783.00	20437.0

In [ ]:

```

```

Observation

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

In [487]:

```
df1.groupby(['Gender'])['Purchase'].describe()
```

Out[487]:

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

Observation

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

In [489]:

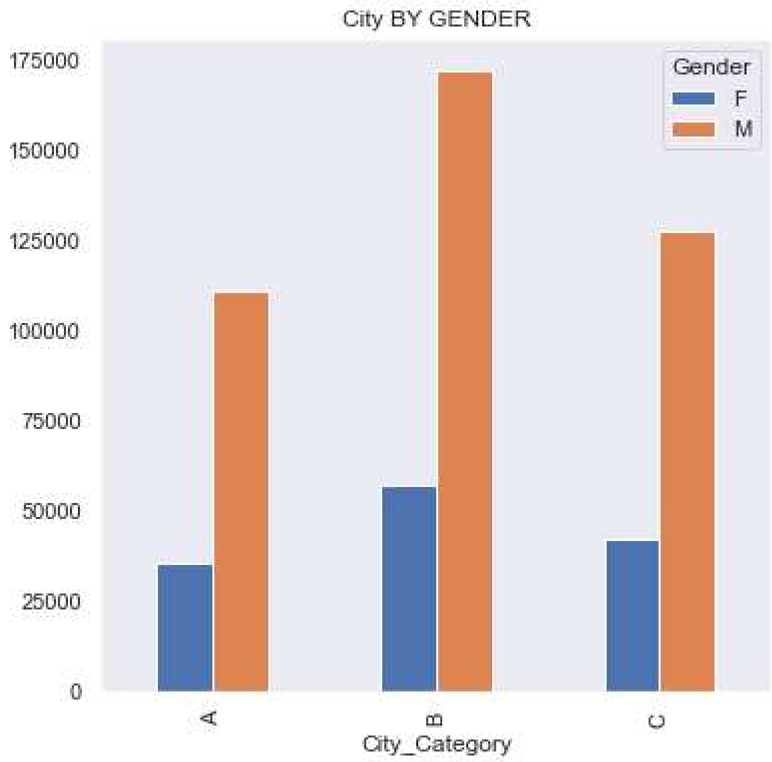
```
plt.figure(figsize=(10,10))
prd_gender=pd.crosstab(df1['City_Category'],df['Gender'] )
print(prd_gender)

ax=prd_gender.plot(kind='bar')

plt.title("City BY GENDER")
```

```
Gender              F       M
City_Category
A               35552  111484
B               57572  172542
C               42096  128145
```

Out[489]:

```
Text(0.5, 1.0, 'City BY GENDER')
```

```
<Figure size 720x720 with 0 Axes>
```



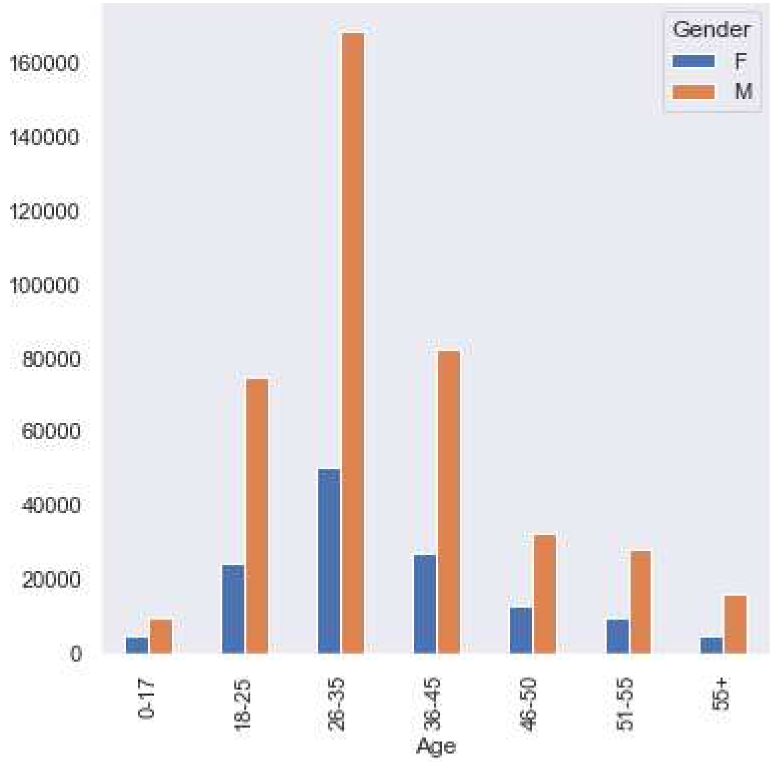
In [490]:

```
prd_mar_status=pd.crosstab(df1['Age'],df1['Gender'] )
print(prd_mar_status)
prd_mar_status.plot(kind='bar')
```

```
Gender      F       M
Age
0-17     5062    9970
18-25   24582   74752
26-35   50560  168101
36-45   27036   82373
46-50   13136   32306
51-55    9815   28376
55+      5029   16293
```

Out[490]:

<AxesSubplot:xlabel='Age'>



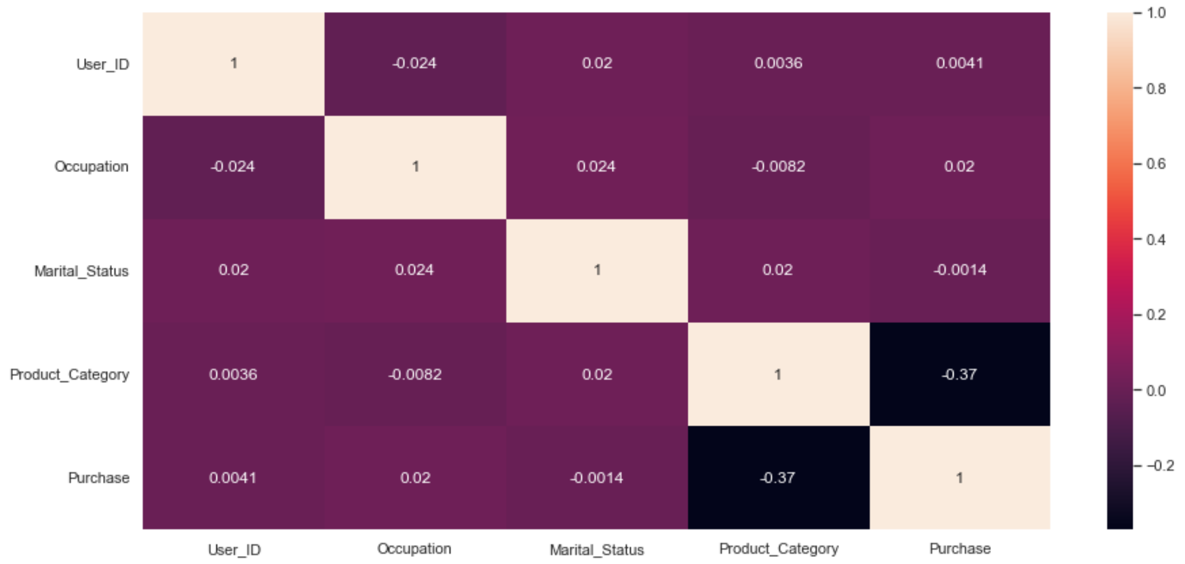
Heatmap

In [491]:

```
plt.figure(figsize=(15,7))
sns.heatmap(df1.corr(), annot=True)
```

Out[491]:

```
<AxesSubplot:>
```



In [492]:

```
corr_pairs = df1.corr().unstack() # give pairs of correlation
print( corr_pairs[abs(corr_pairs)>0.5]) # Gives us correlated data
```

```
User_ID           User_ID             1.0
Occupation        Occupation          1.0
Marital_Status    Marital_Status      1.0
Product_Category  Product_Category    1.0
Purchase          Purchase            1.0
dtype: float64
```

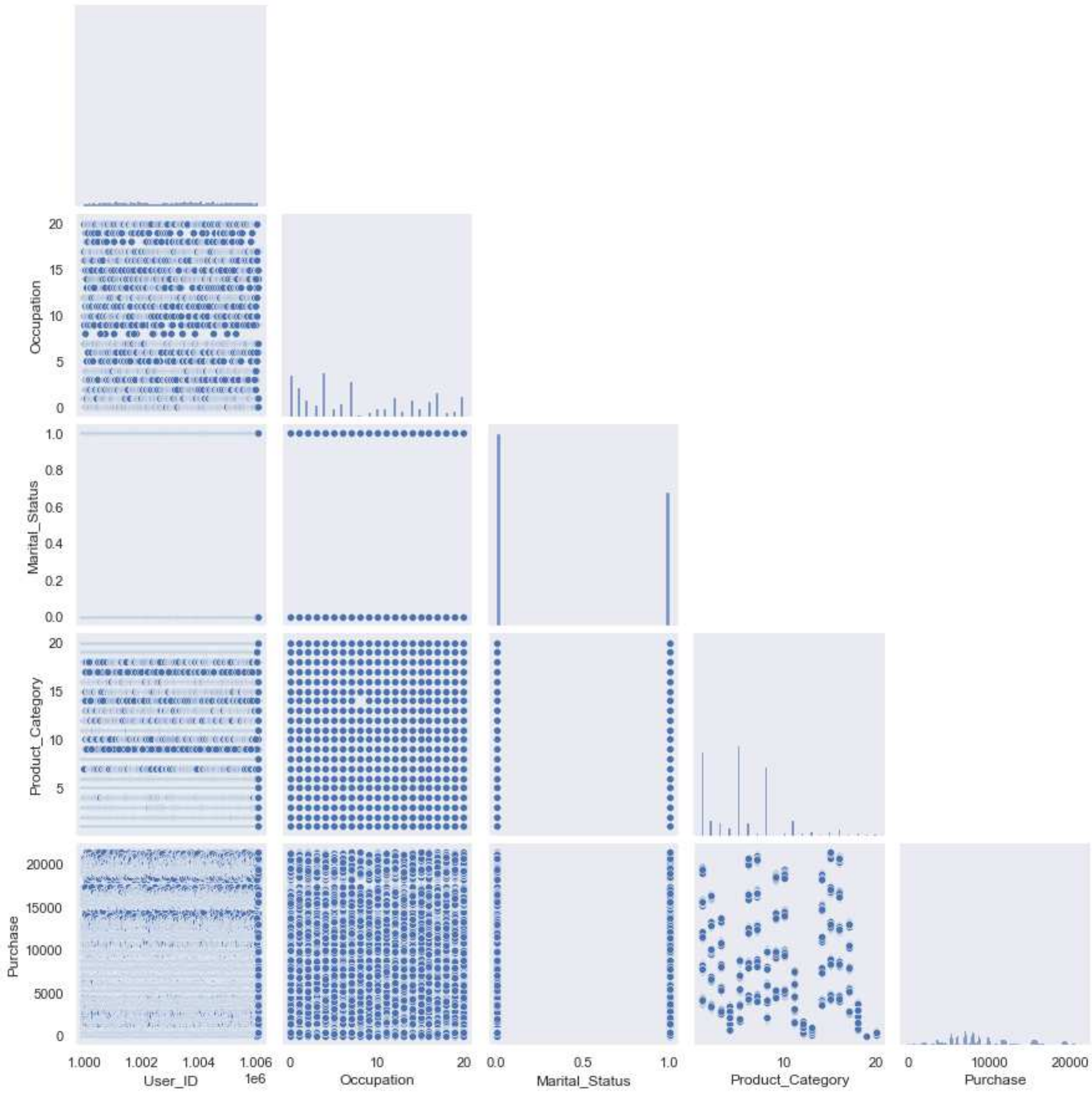
In [493]:

```
plt.figure(figsize=(15,7))
sns.pairplot(data=df1,corner=True)
```

Out[493]:

<seaborn.axisgrid.PairGrid at 0x24050a95fd0>

<Figure size 1080x504 with 0 Axes>

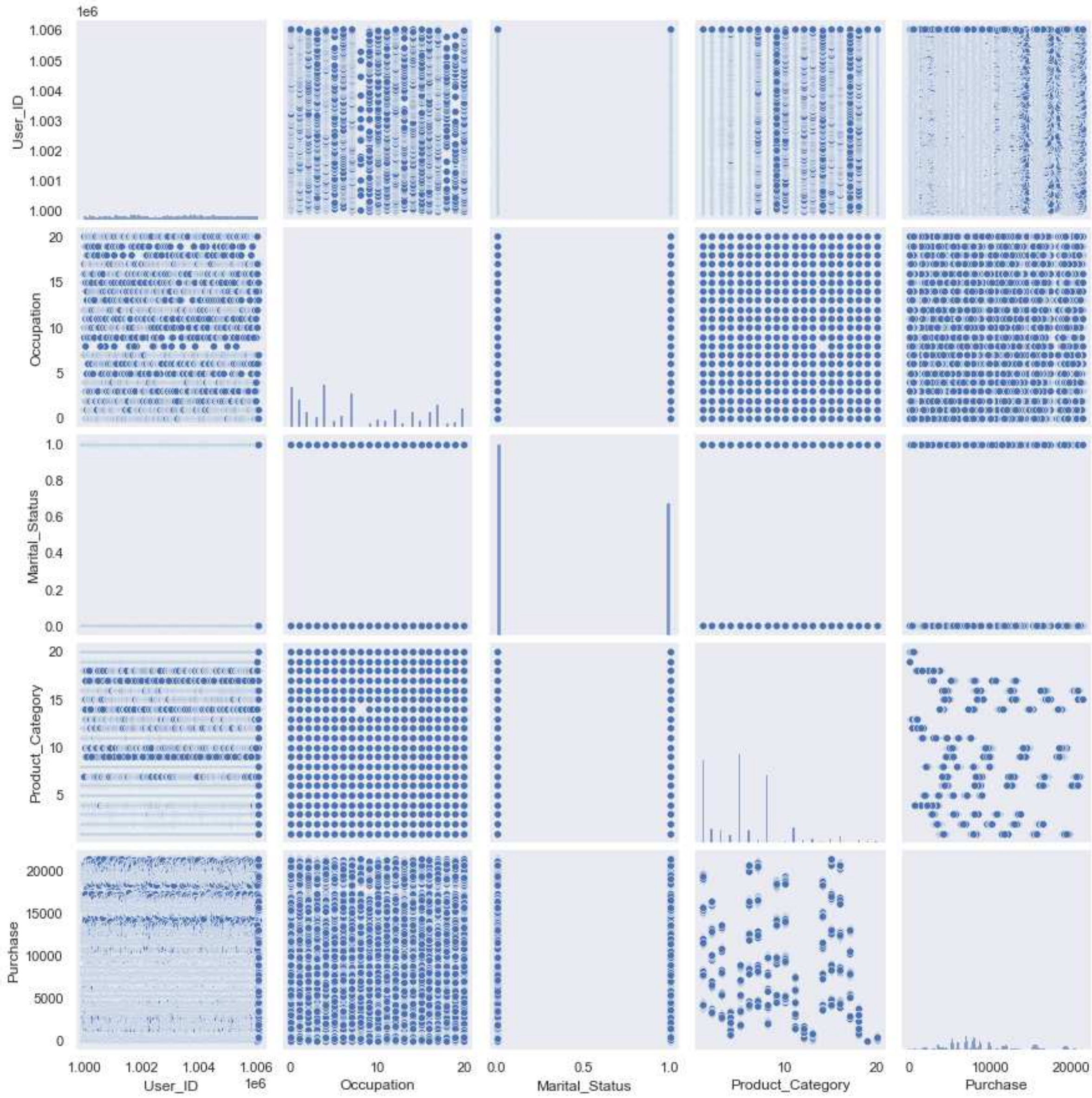


In [494]:

```
1 sns.pairplot(df1)
```

Out[494]:

<seaborn.axisgrid.PairGrid at 0x24051512640>



Observation

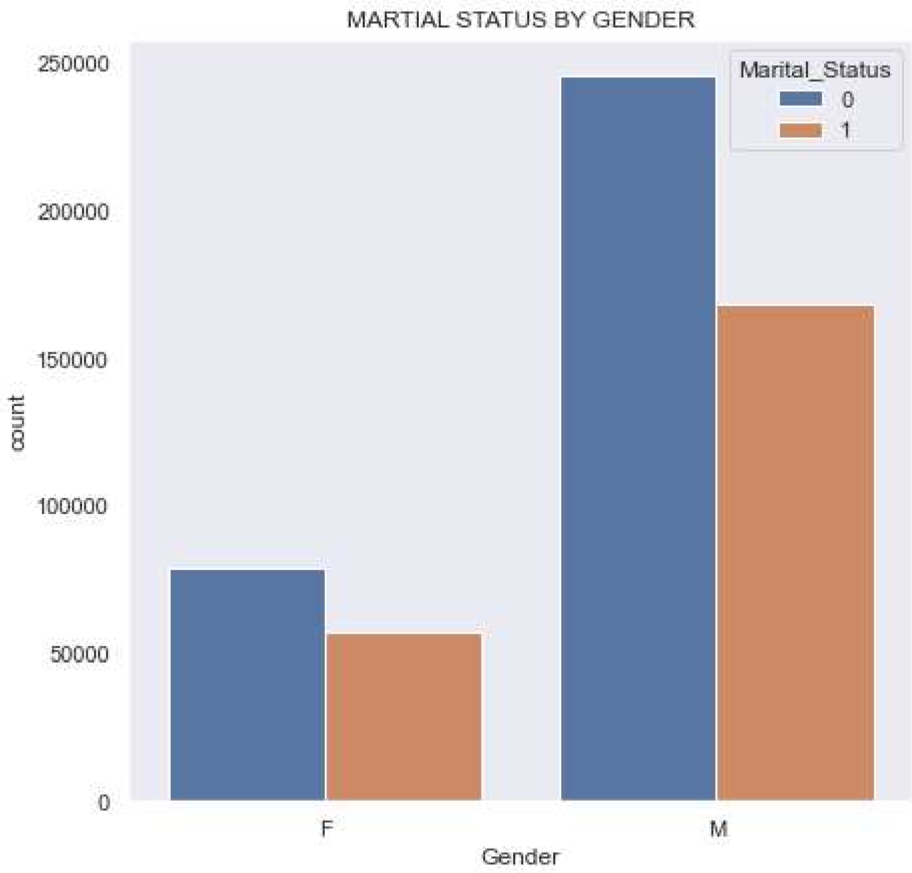
Mostly features are categorical and not much correlation can be observed from above graphs.

In [495]:

```
plt.figure(figsize=(7,7))
sns.countplot(df['Gender'],hue=df["Marital_Status"]).set(title='MARTIAL STATUS BY GENDE
```

Out[495]:

```
[Text(0.5, 1.0, 'MARTIAL STATUS BY GENDER')]
```



In [125]:

```
df.columns
```

Out[125]:

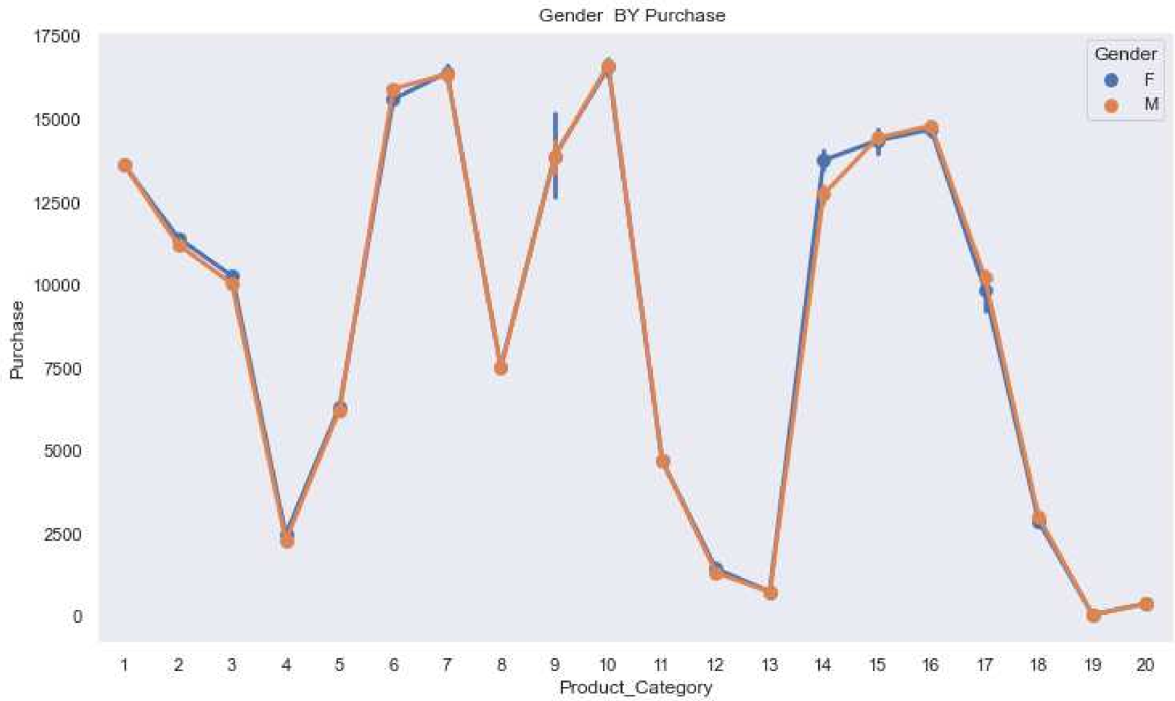
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Categor
y',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase', 'AgeGroup', 'AgeCategory'],
      dtype='object')
```

In [496]:

```
plt.figure(figsize=(12,7))
sns.pointplot(x=df1["Product_Category"],y=df1["Purchase"],hue=df1['Gender']).set(title=
```

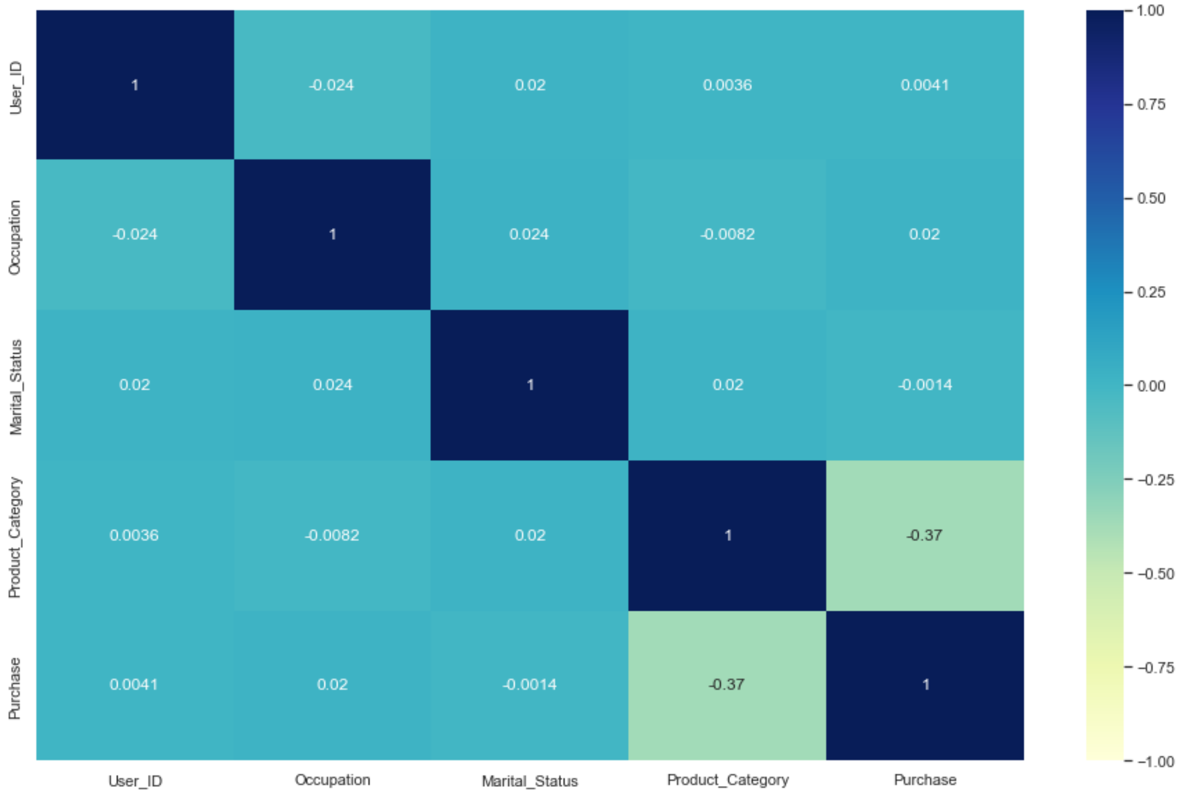
Out[496]:

```
[Text(0.5, 1.0, 'Gender  BY Purchase ')]
```



In [497]:

```
plt.figure(figsize = (16, 10))
sns.heatmap(df1.corr(), annot=True, vmin=-1, vmax = 1,cmap="YlGnBu")
plt.show()
```



Confidence Interval and Central limit theorem(CLT)

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.

Assumptions¶

Randomization: The data must be sampled randomly such that every member in a population has an equal probability of being selected to be in the sample.

Independence: The sample values must be independent of each other.

The 10% Condition: When the sample is drawn without replacement, the sample size should be no larger than 10% of the population.

Large Sample Condition: The sample size needs to be sufficiently large.

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.

In [498]:

```
def bootstrapping(sample1,sample2,smp_siz=500,itr_size=5000,confidence_level=0.95,no_of

    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 distribution
    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 distribution
    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_limit1
    label_mean2=("μ (Females):  {:.2f}".format(mean2))
    label_ult2=("Lower Limit(F):  {:.2f}\nUpper Limit(F):   {:.2f}".format(lower_limit2

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

    plt.show()

    return smp1_means_m,smp2_means_m ,np.round(lower_limit1,2),np.round(upper_limit1,2)
```

In [499]:

```
def bootstrapping_m_vs_um(sample1,sample2,smp_siz=500,itr_size=5000,confidence_level=0.

    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 distribution
    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 distribution
    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 (Married) :  {:.2f}".format(mean1))
    label_ult1=("Lower Limit(M):  {:.2f}\nUpper Limit(M):   {:.2f}".format(lower_limit1
    label_mean2=("μ (Unmarried):  {:.2f}".format(mean2))
    label_ult2=("Lower Limit(F):  {:.2f}\nUpper Limit(F):   {:.2f}".format(lower_limit2

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

    plt.show()

    return smp1_means_m,smp2_means_m ,np.round(lower_limit1,2),np.round(upper_limit1,2)
```

In [500]:

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

    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 distribution
    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,upper_l

    plt.title(f"Sample Size: {smp_siz},Mean:{np.round(mean,2)}, SME:{np.round(sem,2)}",
    plt.xlabel('Purchase')
    plt.axvline(mean, color = 'y', linestyle = 'solid', linewidth = 2,label=label_mean)
    plt.axvline(upper_limit, color = 'r', linestyle = 'solid', linewidth = 2,label=labe
    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)
```

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 [501]:

```
retail_data_smp_male = df1[df1['Gender'] == 'M']['Purchase']
retail_data_smp_female = df1[df1['Gender'] == 'F']['Purchase']
```

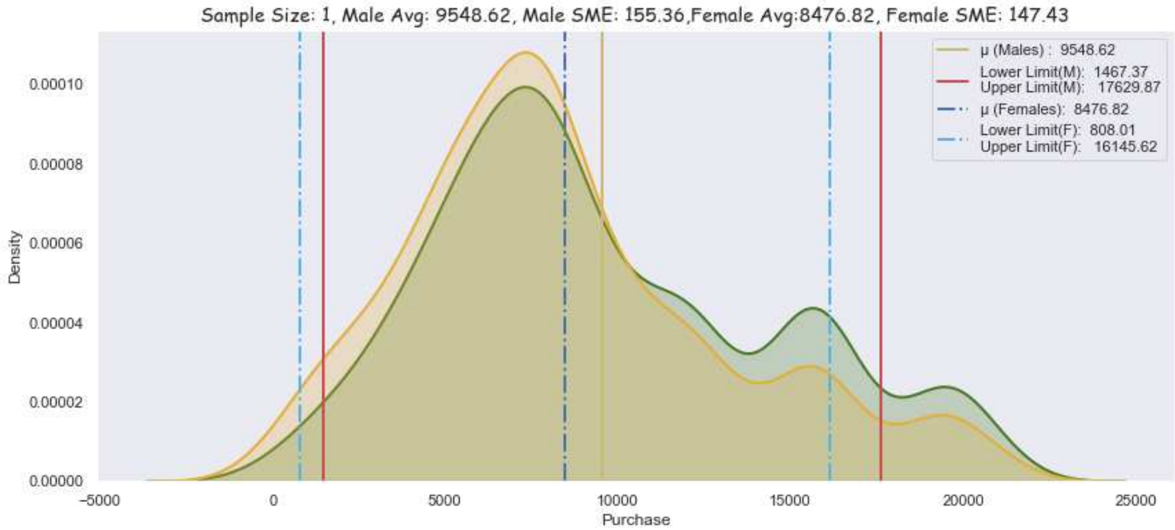
In [502]:

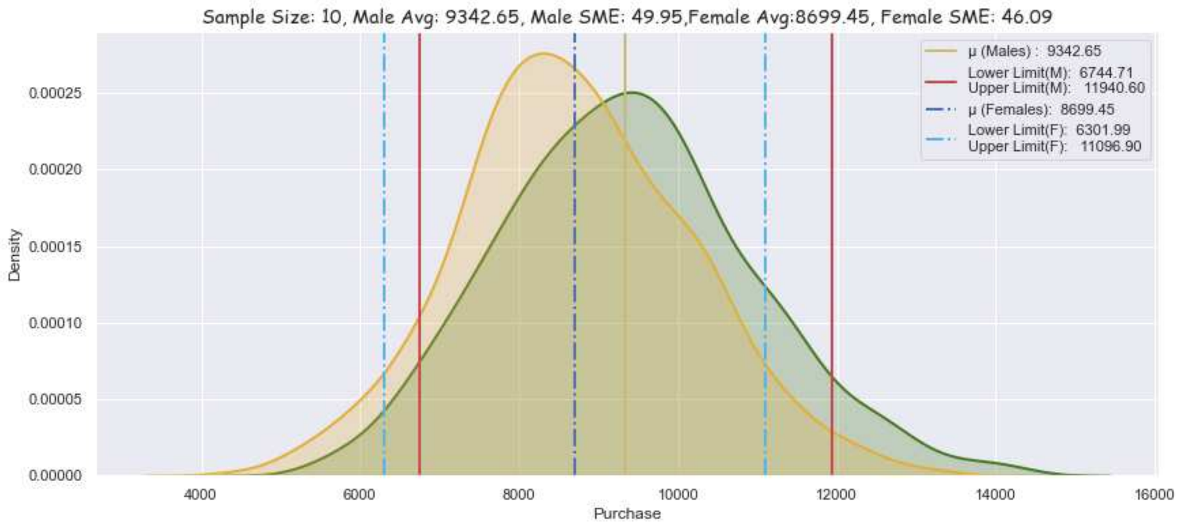
```
print("Male Customers : ",retail_data_smp_male.shape[0])
print("Female Customers : ",retail_data_smp_female.shape[0])
```

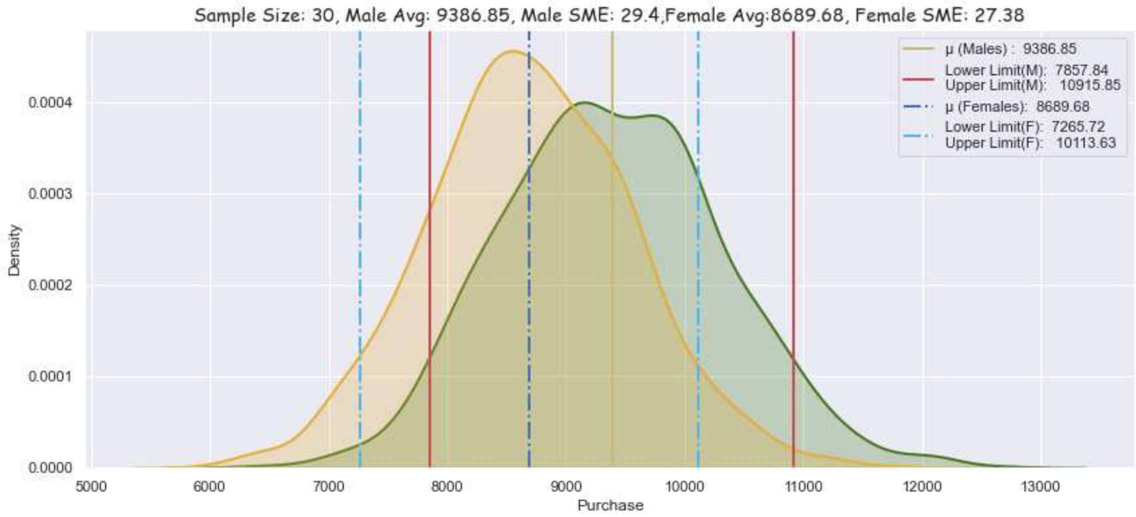
```
Male Customers :  412171
Female Customers :  135220
```

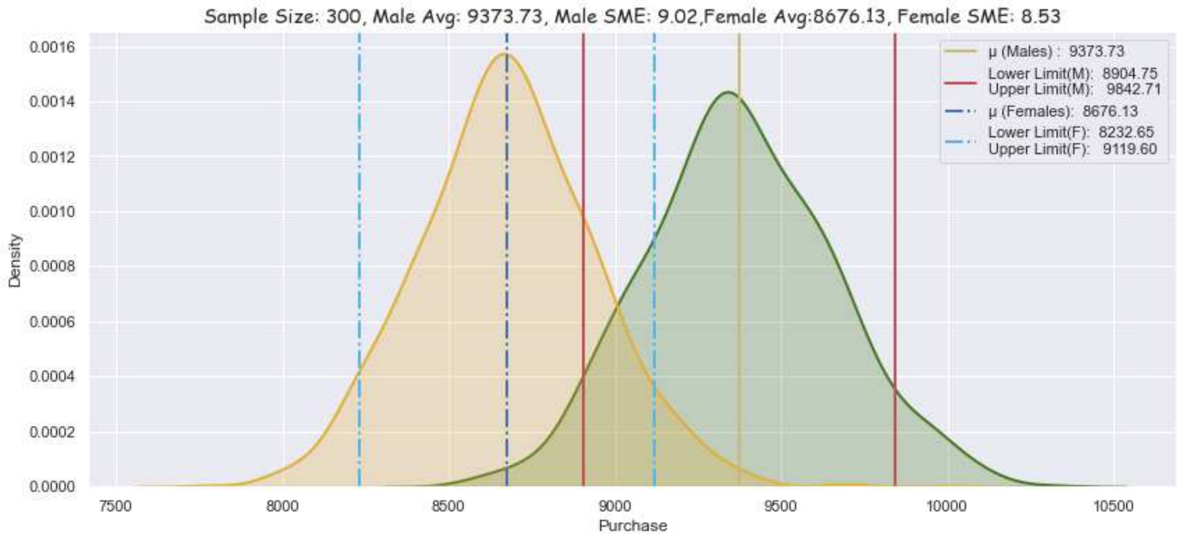
In [503]:

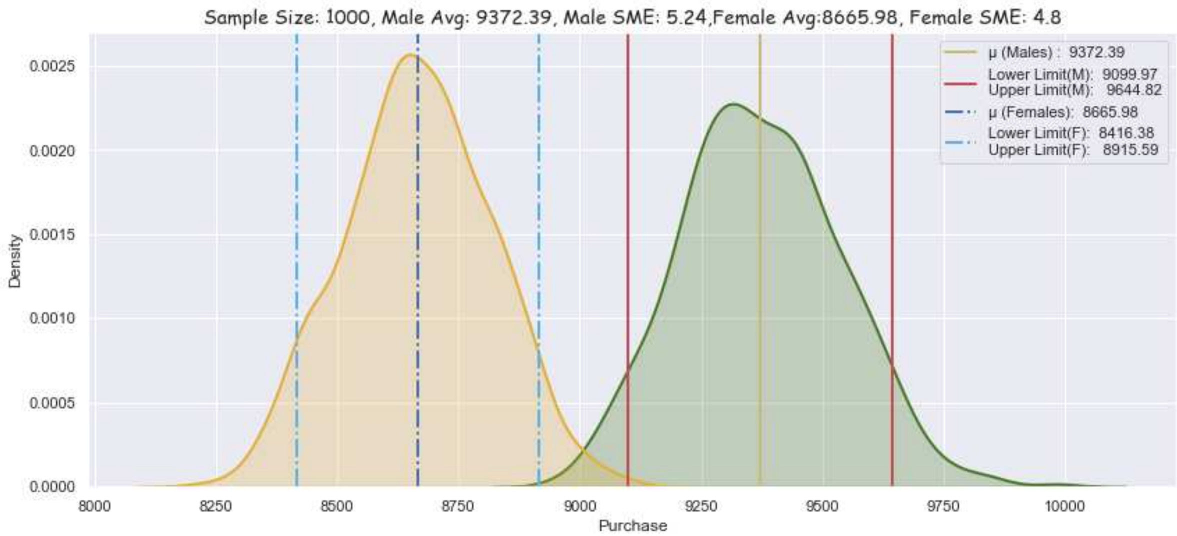
```
itr_size = 1000
size_list = [1, 10, 30, 300, 1000, 100000]
ci = 0.90

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,retail_da

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

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

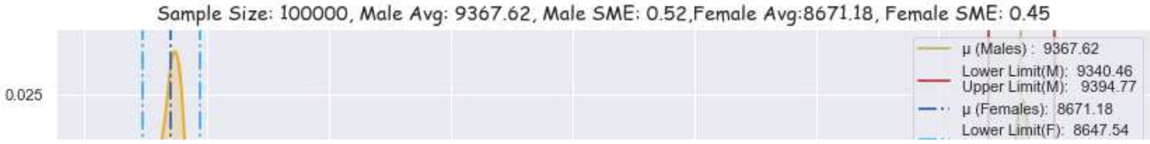





Sample Size: 30, Male Avg: 9386.85, Male SME: 29.4,Female Avg:8689.68, Female SME: 27.38
μ (Males) : 9386.85
Lower Limit(M): 7857.84
Upper Limit(M): 10915.85
μ (Females): 8689.68
Lower Limit(F): 7265.72
Upper Limit(F): 10113.63
Density
Purchase


Sample Size: 300, Male Avg: 9373.73, Male SME: 9.02,Female Avg:8676.13, Female SME: 8.53
μ (Males) : 9373.73
Lower Limit(M): 8904.75
Upper Limit(M): 9842.71
μ (Females): 8676.13
Lower Limit(F): 8232.65
Upper Limit(F): 9119.60
Density
Purchase


Sample Size: 1000, Male Avg: 9372.39, Male SME: 5.24,Female Avg:8665.98, Female SME: 4.8
μ (Males) : 9372.39
Lower Limit(M): 9099.97
Upper Limit(M): 9644.82
μ (Females): 8665.98
Lower Limit(F): 8416.38
Upper Limit(F): 8915.59
Density
Purchase



In [504]:

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

Out[504]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
6	M	8904.75	9842.71	300	[8904.75, 9842.71]	937.96	90
8	M	9099.97	9644.82	1000	[9099.97, 9644.82]	544.85	90
10	M	9340.46	9394.77	100000	[9340.46, 9394.77]	54.31	90

In [505]:

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

Out[505]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
7	F	8232.65	9119.6	300	[8232.65, 9119.6]	886.95	90
9	F	8416.38	8915.59	1000	[8416.38, 8915.59]	499.21	90
11	F	8647.54	8694.81	100000	[8647.54, 8694.81]	47.27	90

Observation

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]

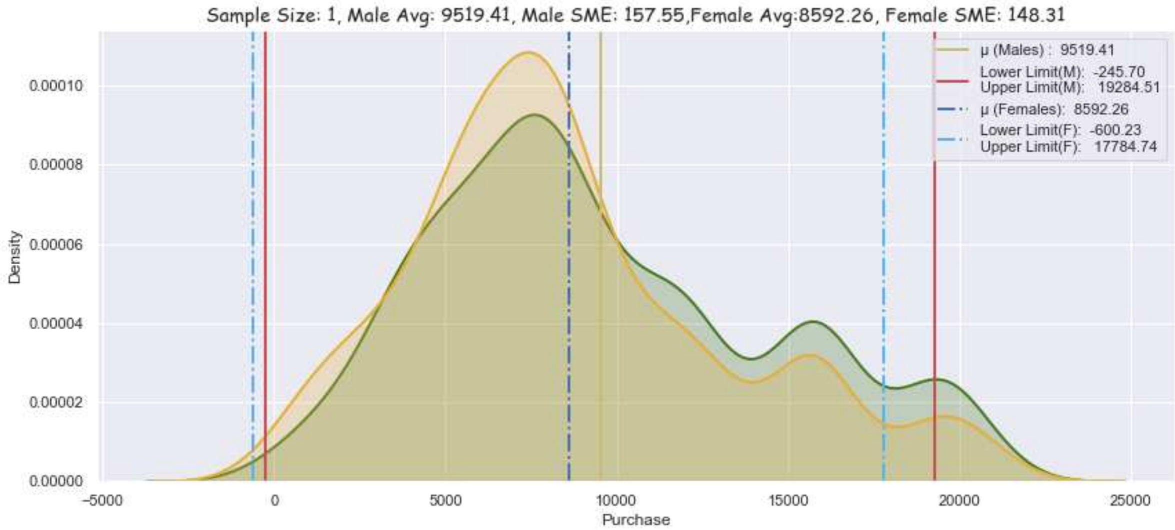
For Male range for mean purchase with confidence interval 90% is [9341.03, 9393.94]

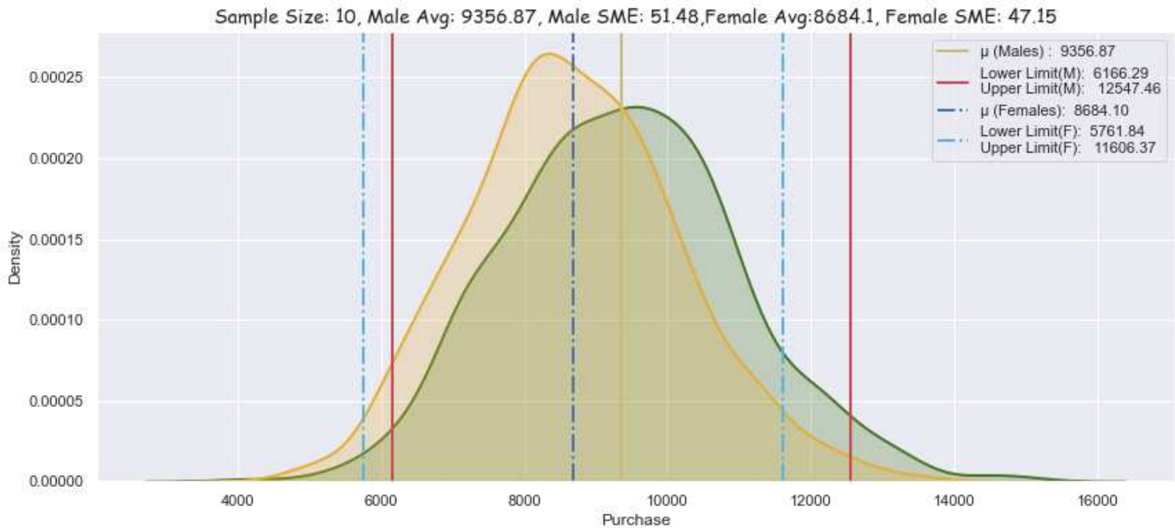
CLT Analysis for mean purchase with confidence 95% - Based on Gender

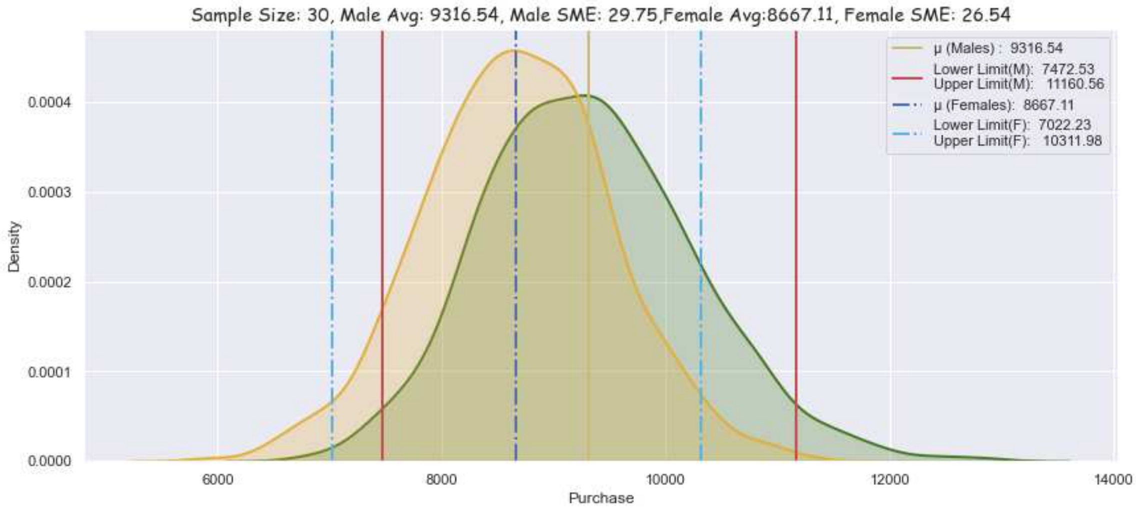
Analysis of the true mean of purchase values by gender with a 95% confidence

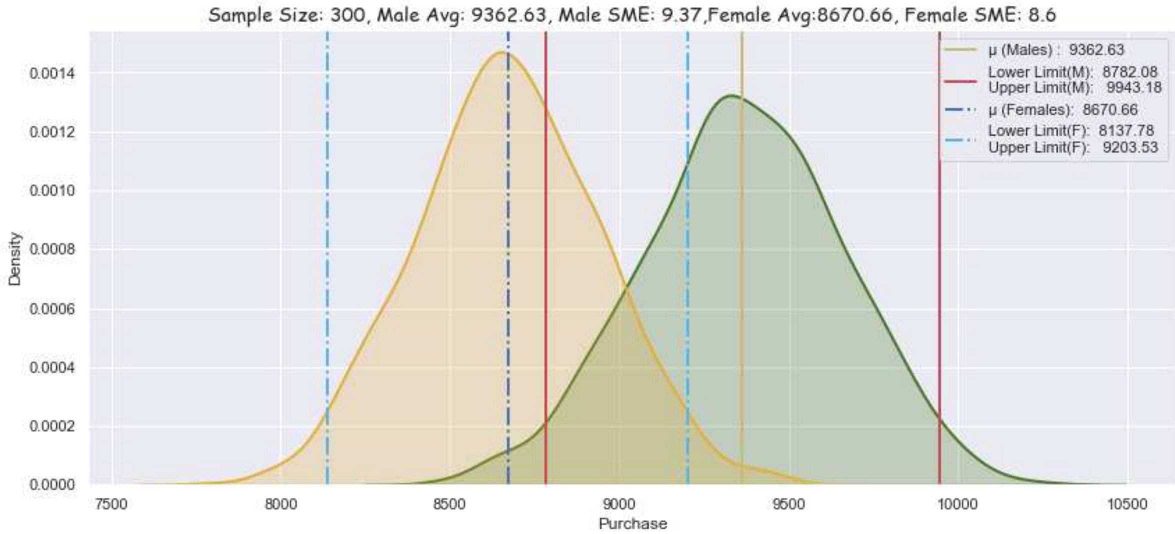
In [506]:

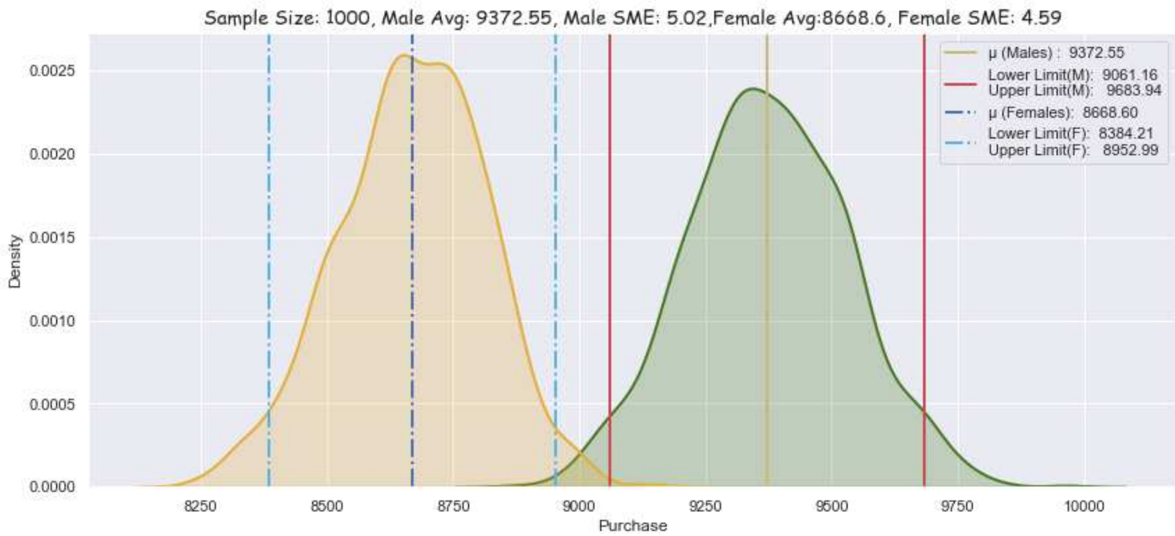
```
itr_size = 1000
size_list = [1, 10, 30, 300, 1000, 100000]
ci = 0.95

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,retail_da

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

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

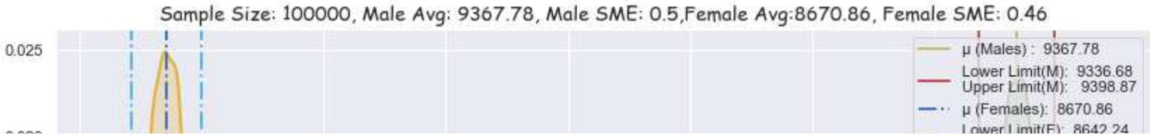





Sample Size: 30, Male Avg: 9316.54, Male SME: 29.75,Female Avg:8667.11, Female SME: 26.54
μ (Males) : 9316.54
Lower Limit(M): 7472.53
Upper Limit(M): 11160.56
μ (Females): 8667.11
Lower Limit(F): 7022.23
Upper Limit(F): 10311.98
Density
Purchase


Sample Size: 300, Male Avg: 9362.63, Male SME: 9.37,Female Avg:8670.66, Female SME: 8.6
μ (Males) : 9362.63
Lower Limit(M): 8782.08
Upper Limit(M): 9943.18
μ (Females): 8670.66
Lower Limit(F): 8137.78
Upper Limit(F): 9203.53
Density
Purchase


Sample Size: 1000, Male Avg: 9372.55, Male SME: 5.02,Female Avg:8668.6, Female SME: 4.59
μ (Males) : 9372.55
Lower Limit(M): 9061.16
Upper Limit(M): 9683.94
μ (Females): 8668.60
Lower Limit(F): 8384.21
Upper Limit(F): 8952.99
Density
Purchase



In [507]:

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

Out[507]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
6	M	8782.08	9943.18	300	[8782.08, 9943.18]	1161.1	95
8	M	9061.16	9683.94	1000	[9061.16, 9683.94]	622.78	95
10	M	9336.68	9398.87	100000	[9336.68, 9398.87]	62.19	95

In [508]:

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

Out[508]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
7	F	8137.78	9203.53	300	[8137.78, 9203.53]	1065.75	95
9	F	8384.21	8952.99	1000	[8384.21, 8952.99]	568.78	95
11	F	8642.24	8699.48	100000	[8642.24, 8699.48]	57.24	95

Observation

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

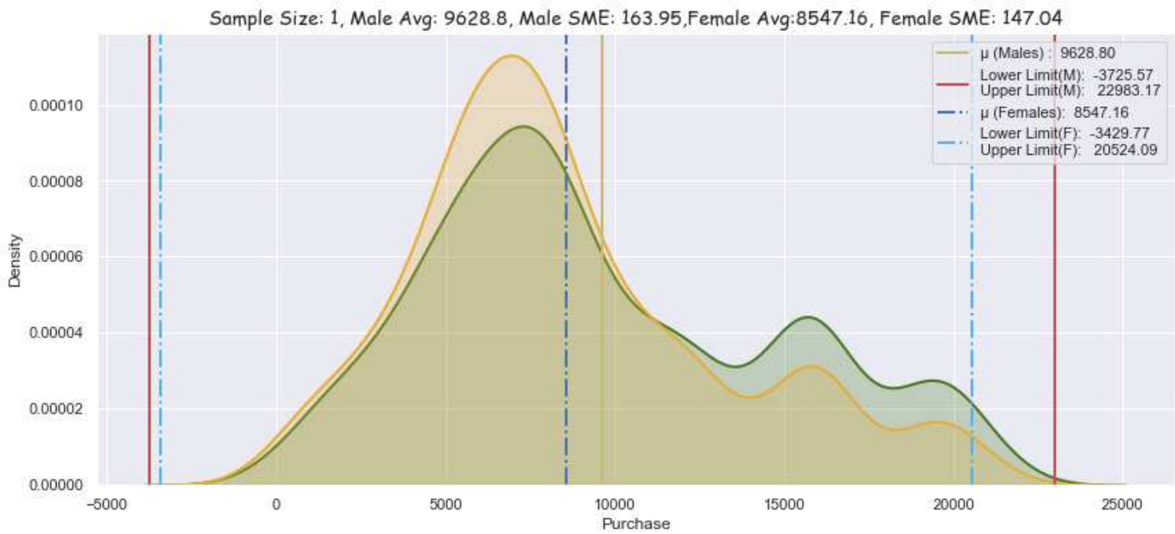
range.

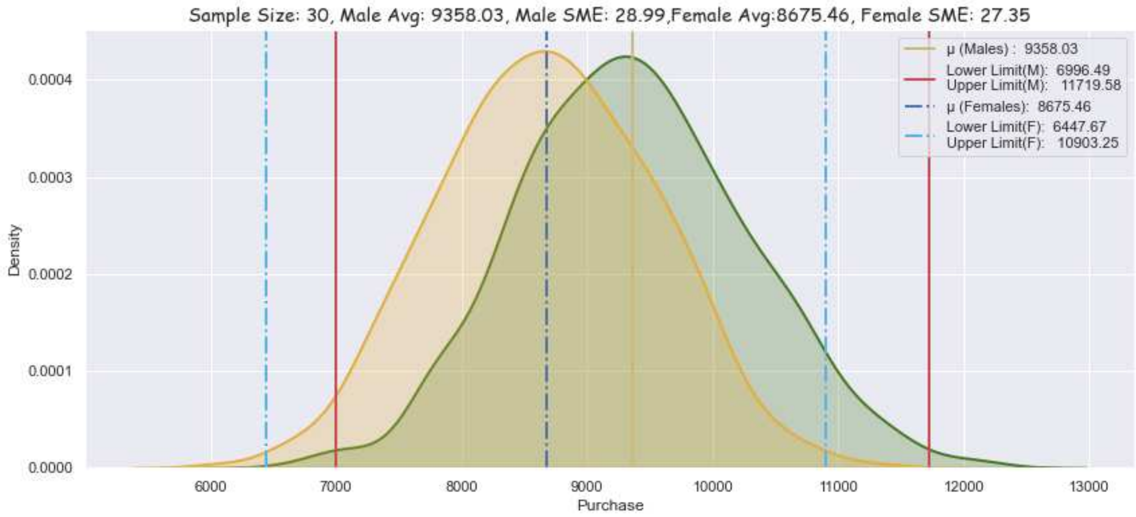
CLT Analysis for mean purchase with confidence 99% - Based on Gender

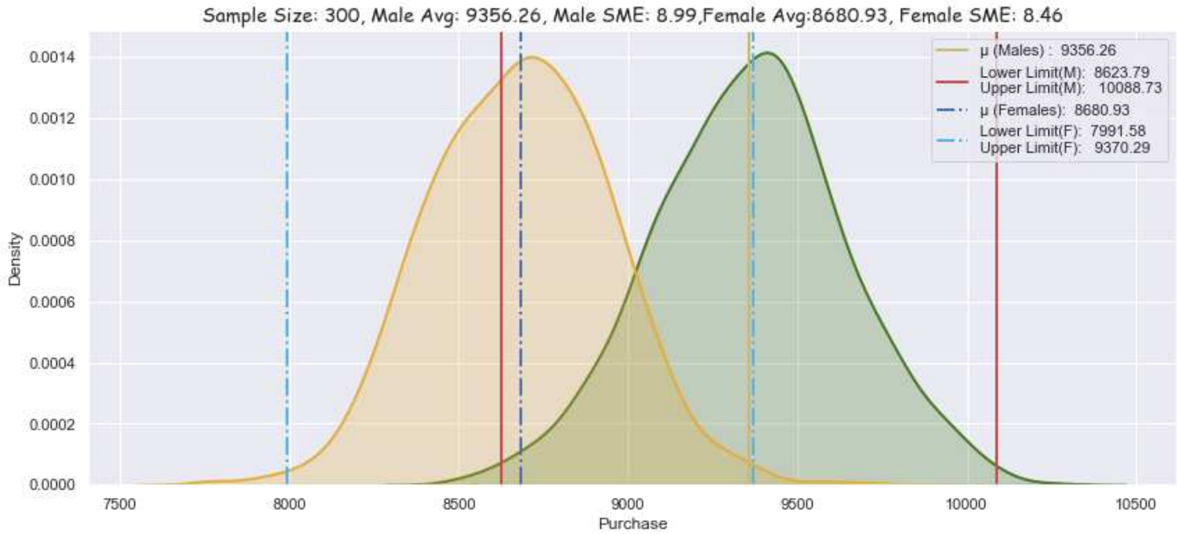
Analysis of the true mean of purchase values by gender with a 99% confidence.

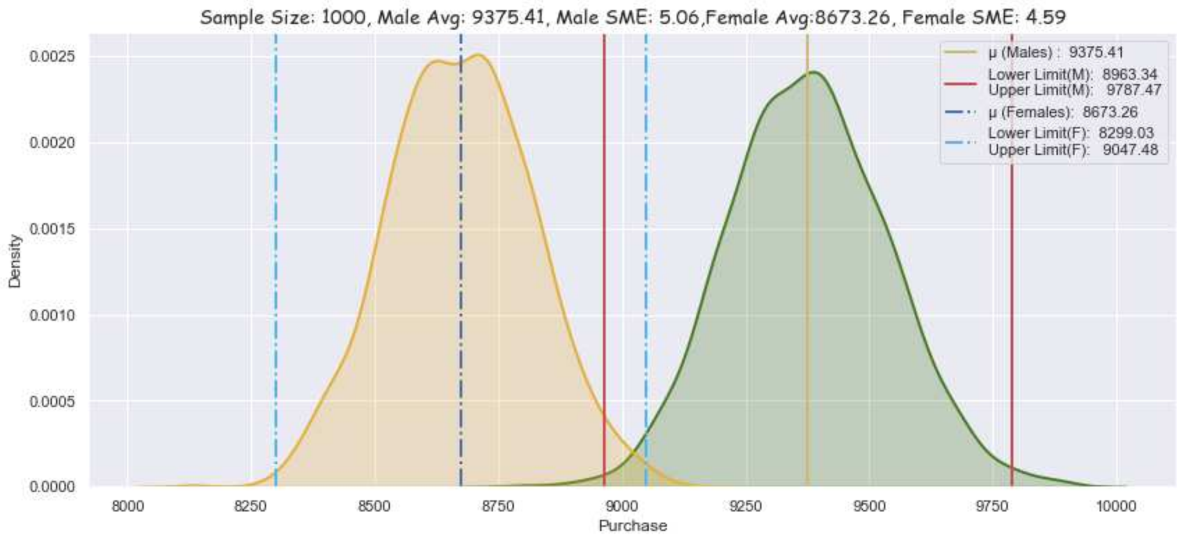
In [509]:

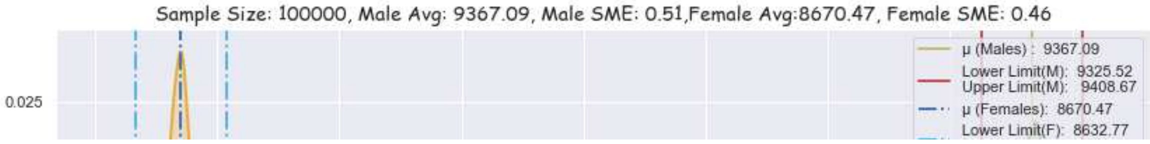
```
itr_size = 1000
size_list = [1, 10, 30, 300, 1000, 100000]
ci = 0.99

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,retail_da

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

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




Sample Size: 30, Male Avg: 9358.03, Male SME: 28.99,Female Avg:8675.46, Female SME: 27.35
μ (Males) : 9358.03
Lower Limit(M): 6996.49
Upper Limit(M): 11719.58
μ (Females): 8675.46
Lower Limit(F): 6447.67
Upper Limit(F): 10903.25
Density
Purchase


Sample Size: 300, Male Avg: 9356.26, Male SME: 8.99,Female Avg:8680.93, Female SME: 8.46
μ (Males) : 9356.26
Lower Limit(M): 8623.79
Upper Limit(M): 10088.73
μ (Females): 8680.93
Lower Limit(F): 7991.58
Upper Limit(F): 9370.29
Density
Purchase


Sample Size: 1000, Male Avg: 9375.41, Male SME: 5.06,Female Avg:8673.26, Female SME: 4.59
μ (Males) : 9375.41
Lower Limit(M): 8963.34
Upper Limit(M): 9787.47
μ (Females): 8673.26
Lower Limit(F): 8299.03
Upper Limit(F): 9047.48
Density
Purchase



In [510]:

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

Out[510]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
6	M	8623.79	10088.73	300	[8623.79, 10088.73]	1464.94	99
8	M	8963.34	9787.47	1000	[8963.34, 9787.47]	824.13	99
10	M	9325.52	9408.67	100000	[9325.52, 9408.67]	83.15	99

In [511]:

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

Out[511]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
7	F	7991.58	9370.29	300	[7991.58, 9370.29]	1378.71	99
9	F	8299.03	9047.48	1000	[8299.03, 9047.48]	748.45	99
11	F	8632.77	8708.17	100000	[8632.77, 8708.17]	75.4	99

In [512]:

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

Out[512]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
10	M	9340.46	9394.77	100000	[9340.46, 9394.77]	54.31	90
10	M	9336.68	9398.87	100000	[9336.68, 9398.87]	62.19	95
10	M	9325.52	9408.67	100000	[9325.52, 9408.67]	83.15	99

In [513]:

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

Out[513]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
11	F	8647.54	8694.81	100000	[8647.54, 8694.81]	47.27	90
11	F	8642.24	8699.48	100000	[8642.24, 8699.48]	57.24	95
11	F	8632.77	8708.17	100000	[8632.77, 8708.17]	75.4	99

Observation

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.

In [ ]:

```

```

Calculate Confidence Interval (CI) - to estimate the

mean weight of the expenses by married and unmarried customers.¶

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

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

In [514]:

```
df1['Marital_Status'].replace(to_replace = 0, value = 'Unmarried', inplace = True)
df1['Marital_Status'].replace(to_replace = 1, value = 'Married', inplace = True)
```

In [515]:

```
df1.sample(500,replace=True).groupby(['Marital_Status'])['Purchase'].describe()
```

Out[515]:

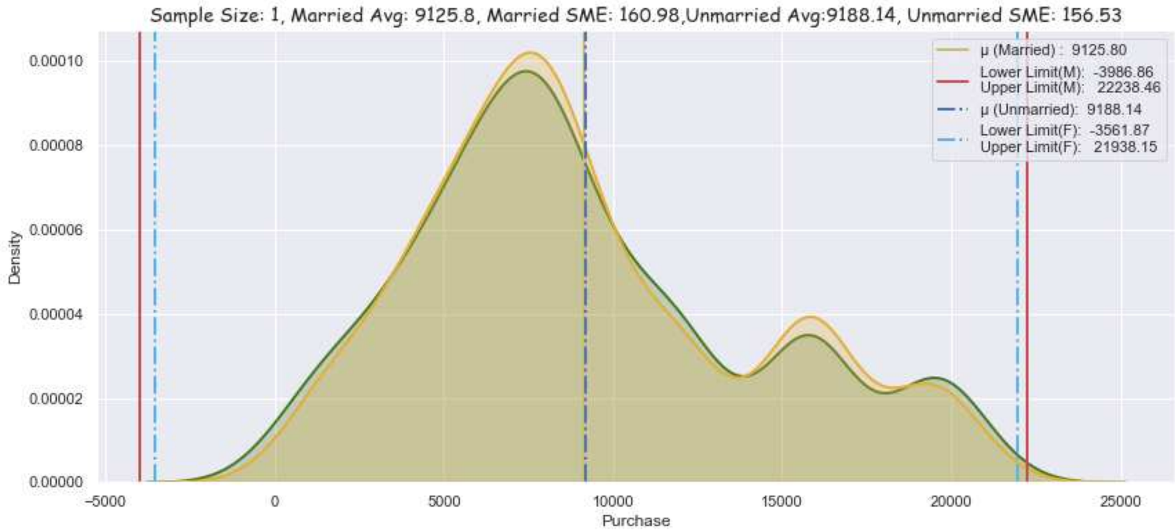
	count	mean	std	min	25%	50%	75%	max
Marital_Status								
Married	203.0	9064.788177	5174.084806	48.0	5345.0	7936.0	11858.0	21314.0
Unmarried	297.0	9274.734007	4956.374473	25.0	5421.0	8028.0	11870.0	20774.0

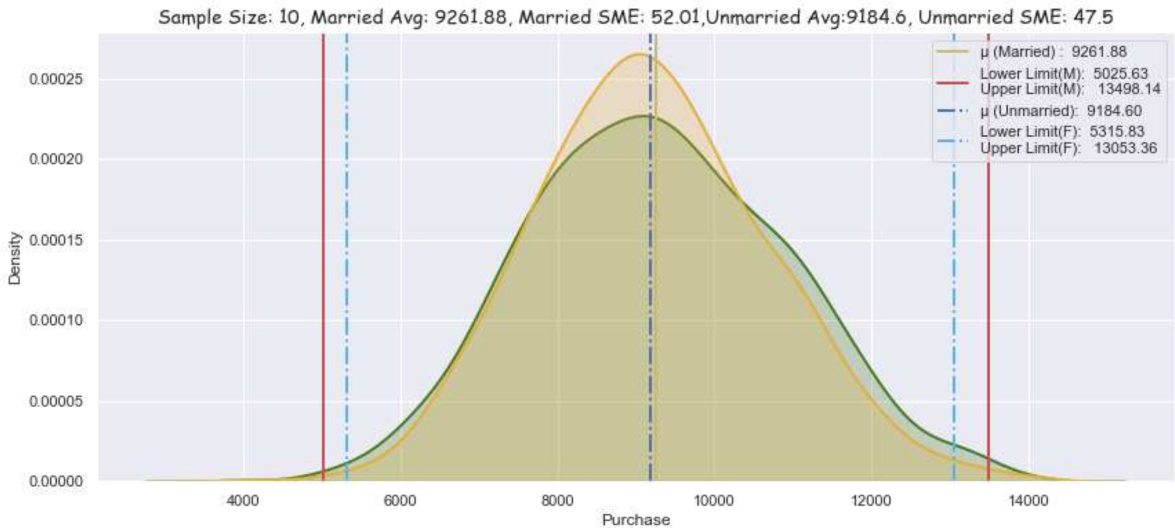
In [516]:

```
retail_data_smp_married = df1[df1['Marital_Status'] == 'Married']['Purchase']
retail_data_smp_unmarried = df1[df1['Marital_Status'] == 'Unmarried']['Purchase']
```

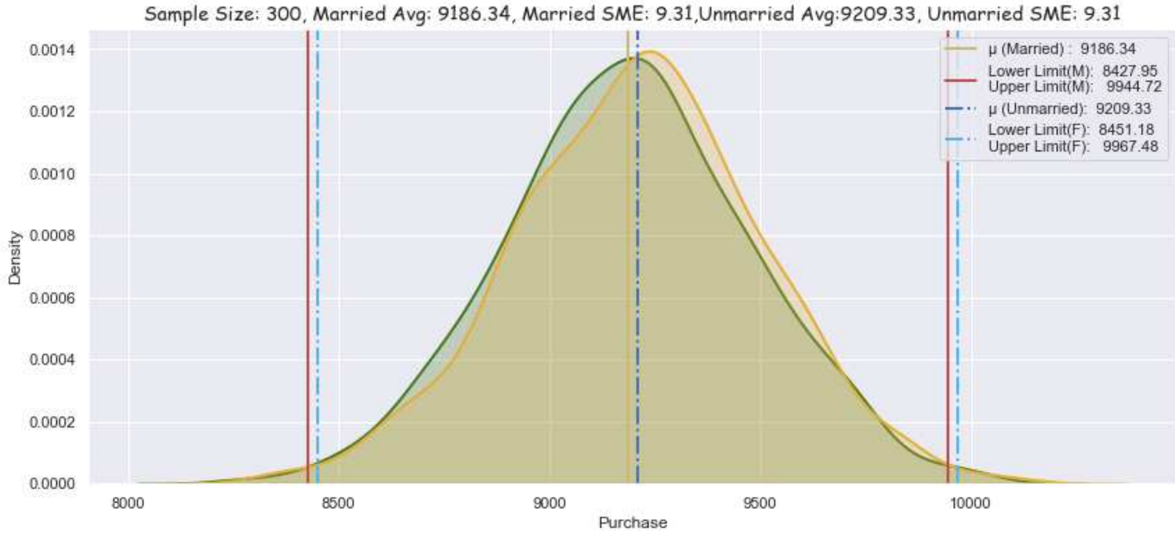
In [517]:

```
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_marrie

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

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






Sample Size: 30, Married Avg: 9209.55, Married SME: 27.58,Unmarried Avg:9195.26, Unmarried SME: 28.0
μ (Married) : 9209.55
Lower Limit(M): 6963.18
Upper Limit(M): 11455.91
μ (Unmarried): 9195.26
Lower Limit(F): 6914.18
Upper Limit(F): 11476.33
Density
Purchase


Sample Size: 300, Married Avg: 9186.34, Married SME: 9.31,Unmarried Avg:9209.33, Unmarried SME: 9.31
μ (Married) : 9186.34
Lower Limit(M): 8427.95
Upper Limit(M): 9944.72
μ (Unmarried): 9209.33
Lower Limit(F): 8451.18
Upper Limit(F): 9967.48
Density
Purchase


Sample Size: 1000, Married Avg: 9180.84, Married SME: 5.01,Unmarried Avg:9206.33, Unmarried SME: 4.91
μ (Married) : 9180.84
Lower Limit(M): 8772.71
Upper Limit(M): 9588.97
μ (Unmarried): 9206.33
Lower Limit(F): 8806.59
Upper Limit(F): 9606.08
Density
Purchase

In [519]:

```
overlap.head()
```

Out[519]:

	Marital_Status	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
0	Married	-3986.86	22238.46	1	[-3986.86, 22238.46]	26225.32	99
1	Unmarried	-3561.87	21938.15	1	[-3561.87, 21938.15]	25500.02	99
2	Married	5025.63	13498.14	10	[5025.63, 13498.14]	8472.51	99
3	Unmarried	5315.83	13053.36	10	[5315.83, 13053.36]	7737.53	99
4	Married	6963.18	11455.91	30	[6963.18, 11455.91]	4492.73	99

In [520]:

```
overlap.loc[(overlap['Marital_Status'] == 'Married') & (overlap['Sample_Size'] >= 300)]
```

Out[520]:

	Marital_Status	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
6	Married	8427.95	9944.72	300	[8427.95, 9944.72]	1516.77	99
8	Married	8772.71	9588.97	1000	[8772.71, 9588.97]	816.26	99
10	Married	9146.89	9227.12	100000	[9146.89, 9227.12]	80.23	99

In [521]:

```
overlap.loc[(overlap['Marital_Status'] == 'Unmarried') & (overlap['Sample_Size'] >= 30
```

Out[521]:

	Marital_Status	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
7	Unmarried	8451.18	9967.48	300	[8451.18, 9967.48]	1516.3	99
9	Unmarried	8806.59	9606.08	1000	[8806.59, 9606.08]	799.49	99
11	Unmarried	9161.91	9240.89	100000	[9161.91, 9240.89]	78.98	99

Observation

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]

In [ ]:

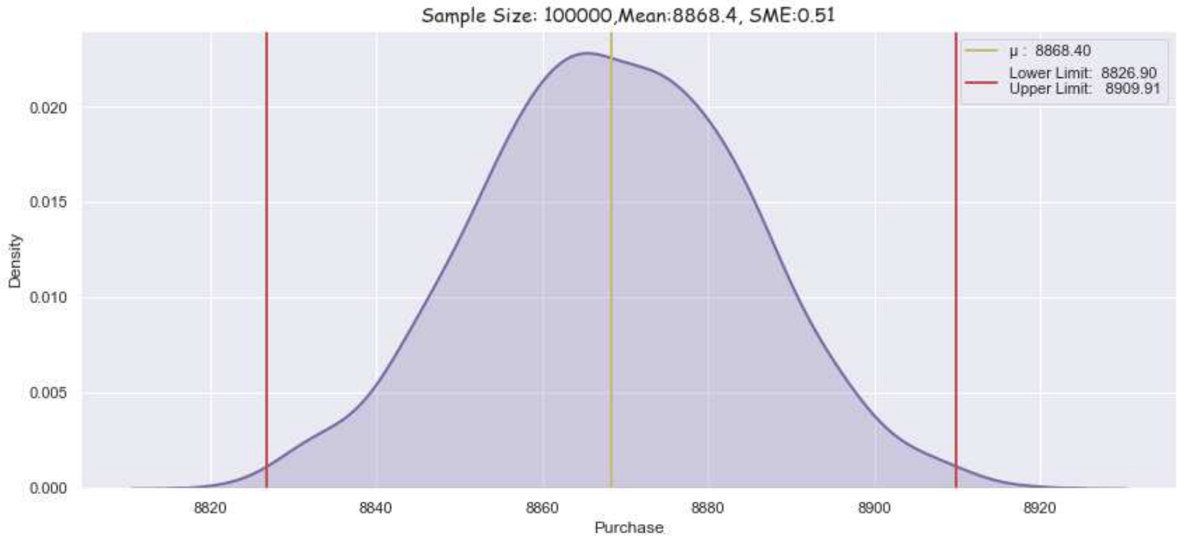
```
 
```

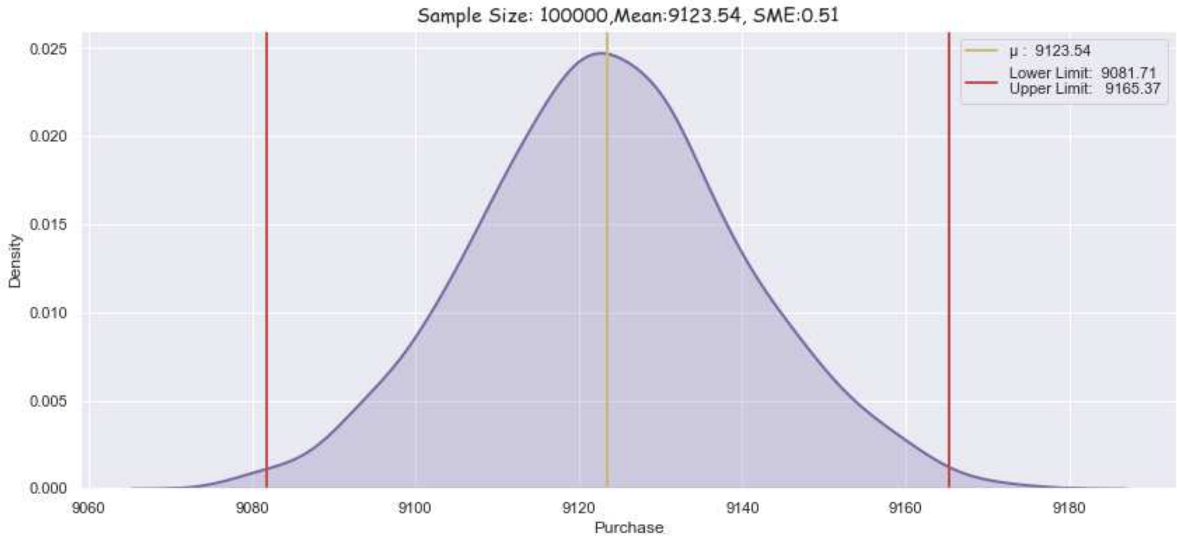
CLT Analysis for mean purchase with confidence 99% - Based on Age Group

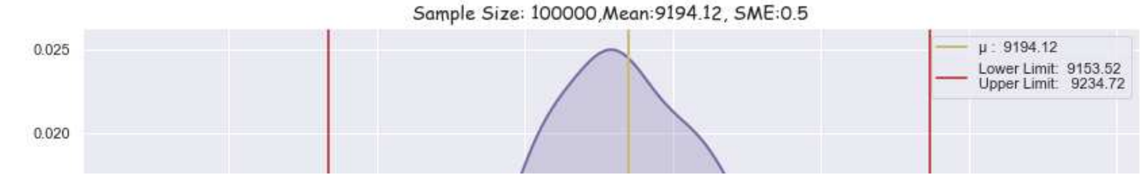
Analysis of the true mean of purchase values by Age Group with a 99% confidence.

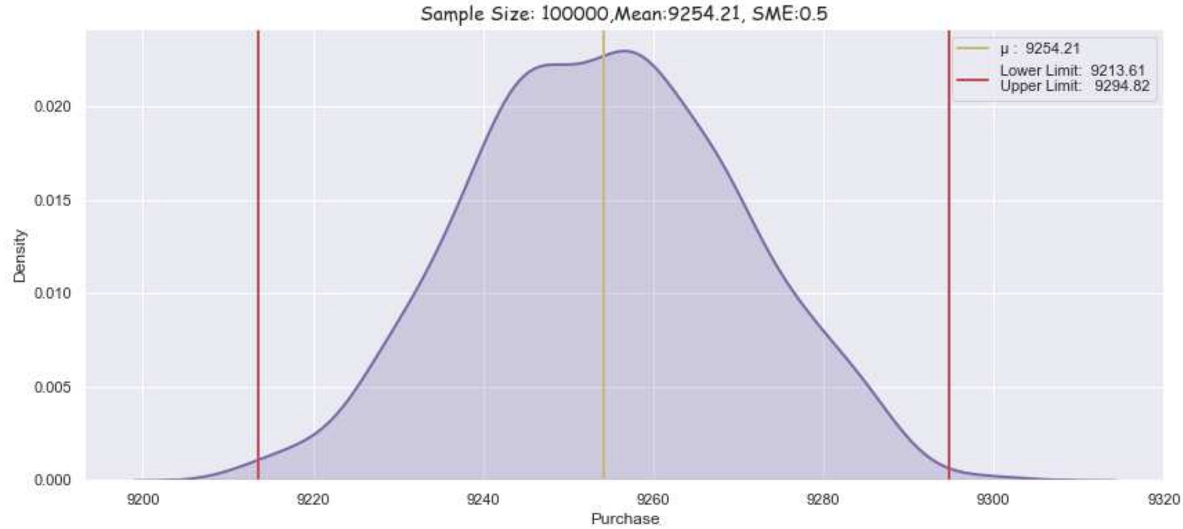
In [522]:

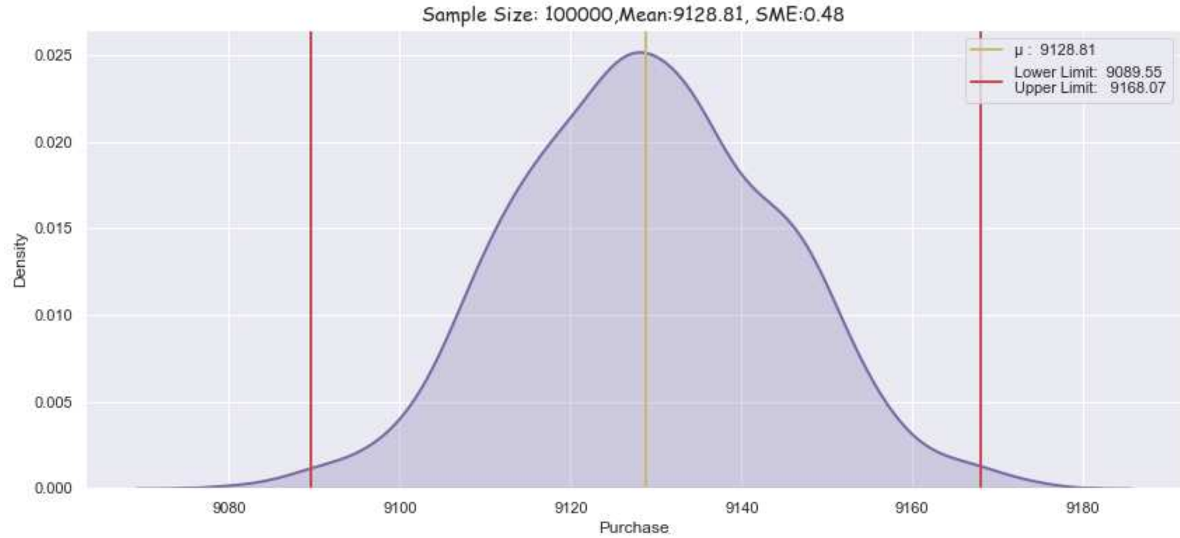
```
itr_size = 1000
smp_size = 1000
ci = 0.99
age_list =['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

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

for age in age_list:
    mean, ll_m, ul_m = bootstrapping_age(df1[df1['Age'] == age]['Purchase'],smp_siz,itr

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

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

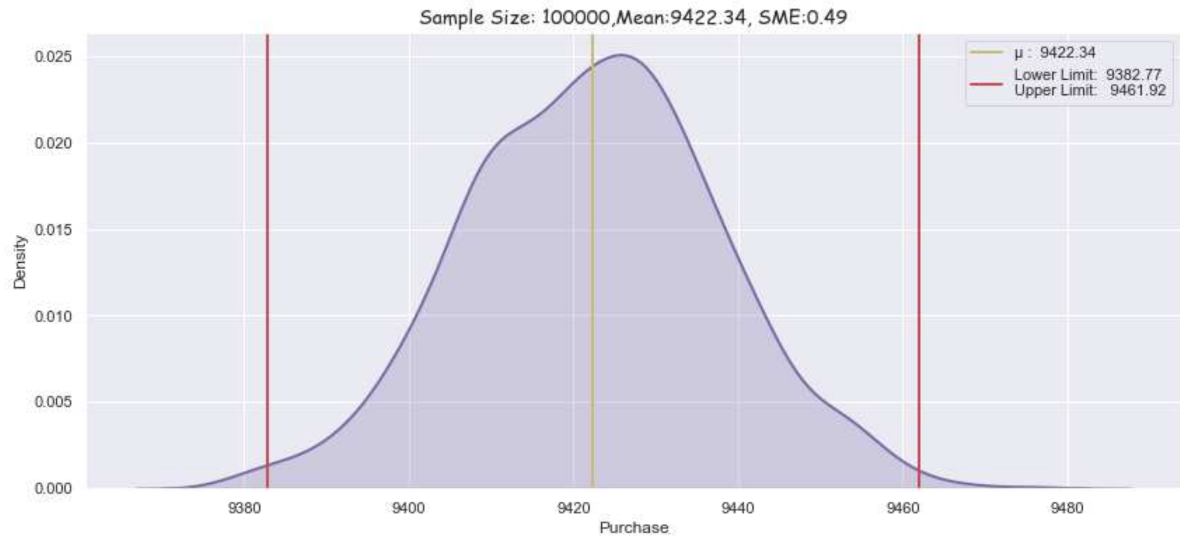


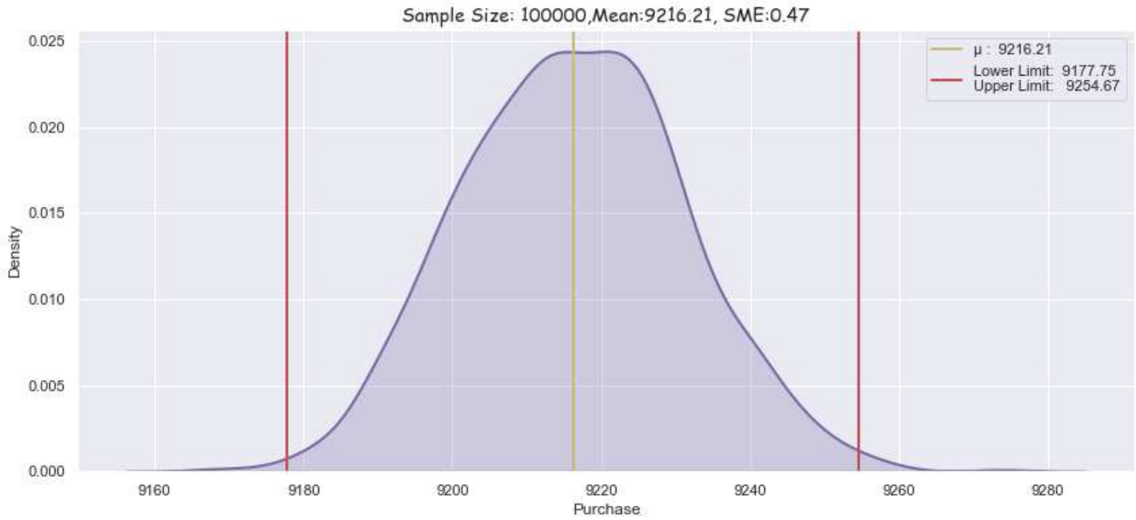



Sample Size: 100000,Mean:9194.12, SME:0.5
μ : 9194.12
Lower Limit: 9153.52
Upper Limit: 9234.72


Sample Size: 100000,Mean:9254.21, SME:0.5
μ : 9254.21
Lower Limit: 9213.61
Upper Limit: 9294.82
Density
Purchase


Sample Size: 100000,Mean:9128.81, SME:0.48
μ : 9128.81
Lower Limit: 9089.55
Upper Limit: 9168.07
Density
Purchase


Sample Size: 100000,Mean:9422.34, SME:0.49
μ : 9422.34
Lower Limit: 9382.77
Upper Limit: 9461.92
Density
Purchase


Sample Size: 100000,Mean:9216.21, SME:0.47
μ : 9216.21
Lower Limit: 9177.75
Upper Limit: 9254.67
Density
0.025
0.020
0.015
0.010
0.005
0.000
9160
9180
9200
9220
9240
9260
9280
Purchase

In [523]:

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

Out[523]:

	Age_Group	Mean	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_ı
0	0-17	[8875.0, 8842.47, 8872.33, 8847.31, 8879.01, 8...	8826.9	8909.91	100000	[8826.9, 8909.91]	83.01	
1	18-25	[9125.27, 9101.05, 9120.14, 9127.74, 9138.5, 9...	9081.71	9165.37	100000	[9081.71, 9165.37]	83.66	
2	26-35	[9201.32, 9199.47, 9187.59, 9178.37, 9205.67, ...	9153.52	9234.72	100000	[9153.52, 9234.72]	81.2	
3	36-45	[9274.63, 9259.09, 9249.27, 9259.07, 9225.51, ...	9213.61	9294.82	100000	[9213.61, 9294.82]	81.21	
4	46-50	[9148.03, 9118.48, 9140.93, 9116.92, 9137.74, ...	9089.55	9168.07	100000	[9089.55, 9168.07]	78.52	
5	51-55	[9430.06, 9436.56, 9425.79, 9434.78, 9436.53, ...	9382.77	9461.92	100000	[9382.77, 9461.92]	79.15	
6	55+	[9219.69, 9210.84, 9208.5, 9212.06, 9218.21, 9...	9177.75	9254.67	100000	[9177.75, 9254.67]	76.92	

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

In [524]:

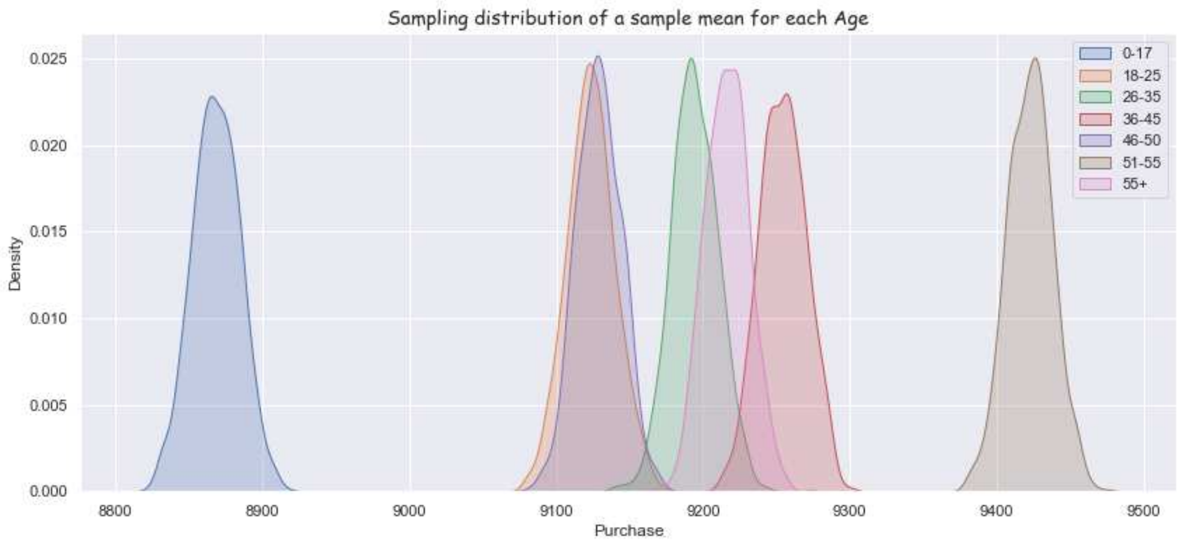
```
age_dict = {}
age_list = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
for i in range(len(age_data)):
    age_dict[age_list[i]] = age_data.loc[i, "Mean"]
```

In [525]:

```
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,family="Cor
plt.xlabel('Purchase')
plt.legend(loc='upper right')
```

Out[525]:

```
<matplotlib.legend.Legend at 0x240769d7f70>
```



Observation

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

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
1
```

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.

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

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.

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..