Walmart Case study

November 22, 2023

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
```

1 Defining Problem Statement and Analyzing basic metrics

[3]:
```
df = pd.read_csv("/Users/senth/Desktop/walmart_data.csv")
df.head()
```

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

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

[4]:
```
print(f"Number of rows: {df.shape[0]:,} \nNumber of columns: {df.shape[1]}")
```

```
Number of rows: 550,068
Number of columns: 10
```

[5]:
```
df.isna().sum()
```

[5]:
```
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
```

[6]:
```python
df.nunique().sort_values(ascending=False)
```

[6]:
```
Purchase                      18105
User_ID                        5891
Product_ID                     3631
Occupation                       21
Product_Category                 20
Age                               7
Stay_In_Current_City_Years        5
City_Category                     3
Gender                            2
Marital_Status                    2
dtype: int64
```

[7]:
```python
df.duplicated().sum()
```

[7]:
```
0
```

[8]:
```python
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
```

[9]:
```
col = ['User_ID','Product_ID','Gender', 'Age','City_Category','Marital_Status']
df[col] = df[col].astype('category')
```

[10]:
```
df.dtypes
```

[10]:
```
User_ID                       category
Product_ID                    category
Gender                        category
Age                           category
Occupation                       int64
City_Category                 category
Stay_In_Current_City_Years      object
Marital_Status                category
Product_Category                 int64
Purchase                         int64
dtype: object
```

[11]:
```
df.describe().T
```

[11]:
```
                     count         mean          std   min     25%     50%  \
Occupation        550068.0     8.076707     6.522660   0.0     2.0     7.0
Product_Category  550068.0     5.404270     3.936211   1.0     1.0     5.0
Purchase          550068.0  9263.968713  5023.065394  12.0  5823.0  8047.0

                      75%      max
Occupation           14.0     20.0
Product_Category      8.0     20.0
Purchase          12054.0  23961.0
```

[13]:
```
df.describe(include=['object','category']).T
```

[13]:
```
                             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
```

1.1 Univariate Analysis

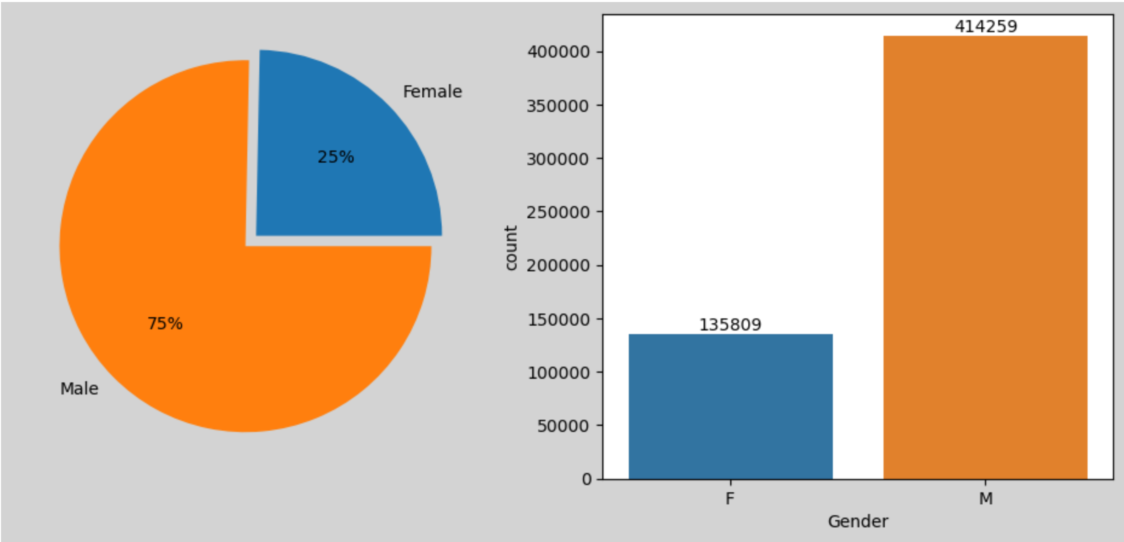
[14]:
```
df['User_ID'].nunique()
```

[14]: 5891

[15]:
```
df['Product_ID'].nunique()
```

[15]: 3631

[16]:
```
plt.figure(figsize = (12,5)).set_facecolor("lightgrey")

plt.subplot(1,2,1)
labels = ['Female','Male']
plt.pie(df.groupby('Gender')['Gender'].count(), labels = labels, explode = (0.
 ↪08,0), autopct = '%0.0f%%')

plt.subplot(1,2,2)
label = sns.countplot(data = df, x='Gender')
for i in label.containers:
    label.bar_label(i)

plt.show()
```



Out of 0.54 million entries, 75% records are of men and 25% of women.

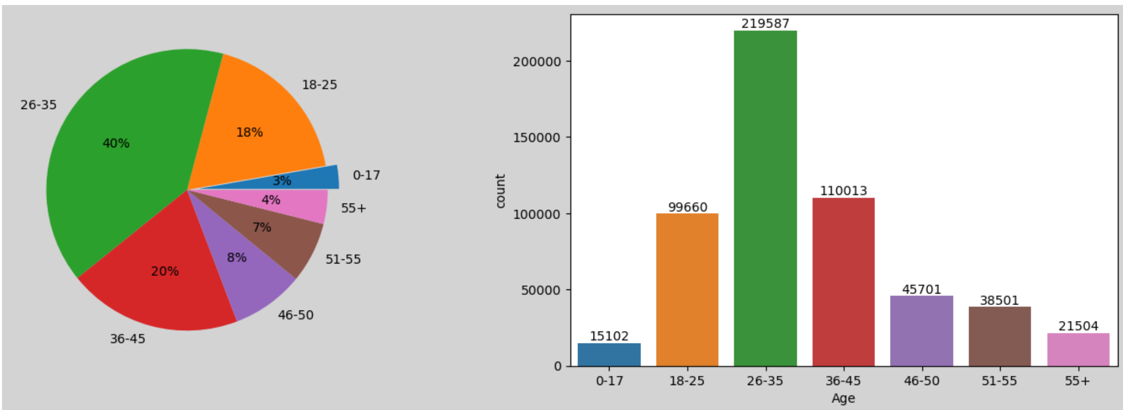
Approximately there are 0.41 million records for men and 0.13 for Females.

[17]:
```
df['Age'].unique()
```

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

[18]:
```
plt.figure(figsize = (17,5)).set_facecolor("lightgrey")

plt.subplot(1,2,1)
```

```
labels = ['0-17','18-25','26-35','36-45','46-50','51-55','55+']
plt.pie(df.groupby('Age')['Age'].count(), labels = labels, explode = (0.
 ↪08,0,0,0,0,0,0), autopct = '%0.0f%%')

plt.subplot(1,2,2)
label = sns.countplot(data = df, x='Age')
for i in label.containers:
    label.bar_label(i)

plt.show()
```



0% of the buyers fall under the age group of 26-35 which is the highest amongst all age groups.

Approximately 0.21 million records are present for age group 26-35 followed by 0.11 million records for group 36-45.

Age group 0-17 and 55+ are the least frequent buyers which is only 3% and 4% of the data respectively.

Approximately only 15k and 21k records are there for age group 0-17 and group 55+.

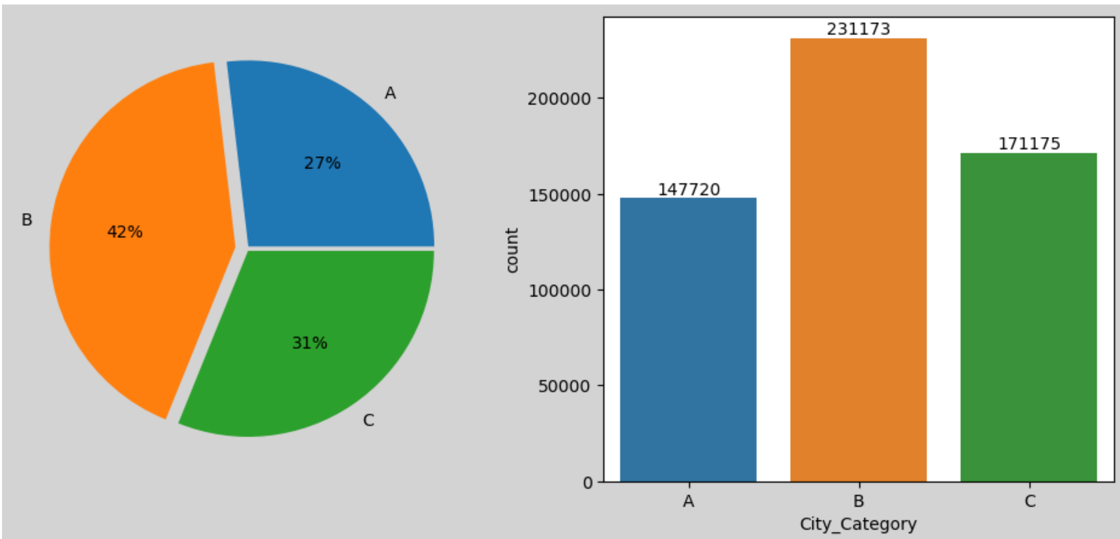
We can observe that most buyers are in within the age of 18-45 before and after this range we can see less buyers.

[19]:
```
df['City_Category'].unique()
```

[19]:
```
['A', 'C', 'B']
Categories (3, object): ['A', 'B', 'C']
```

[20]:
```
plt.figure(figsize = (12,5)).set_facecolor("lightgrey")

plt.subplot(1,2,1)
labels = ['A','B','C']
plt.pie(df.groupby('City_Category')['City_Category'].count(), labels = labels,␣
 ↪explode = (0.015,0.06,0.015), autopct = '%0.0f%%')
```

```
plt.subplot(1,2,2)
label = sns.countplot(data = df, x='City_Category')
for i in label.containers:
    label.bar_label(i)

plt.show()
```



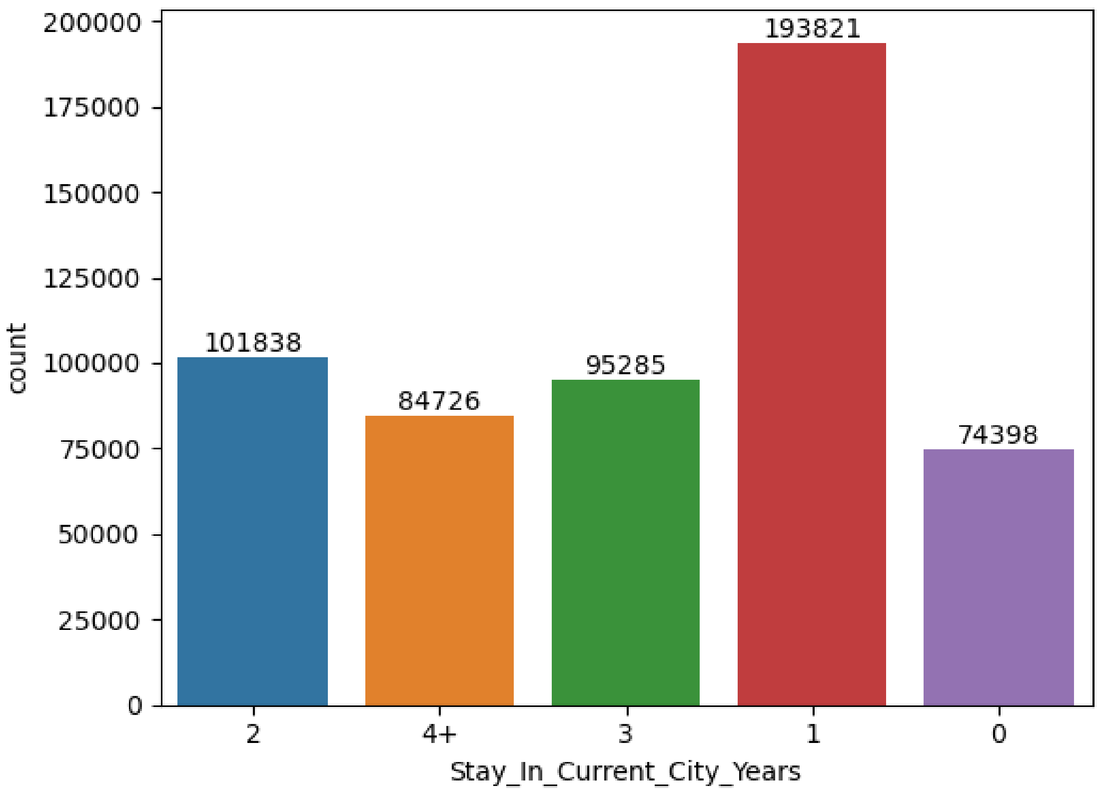
There are 42% buyers from City Category B, 31% from Category C and 27% from Category A

Approximately 0.23 million records are present for Category B, 0.17 million for Category C and

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

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

[22]:
```
label = sns.countplot(data = df, x='Stay_In_Current_City_Years')
for i in label.containers:
    label.bar_label(i)
```



Most buyers are in their current cities since 1 year followed by 2 years and 3 years.

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

[23]:
```
[0, 1]
Categories (2, int64): [0, 1]
```

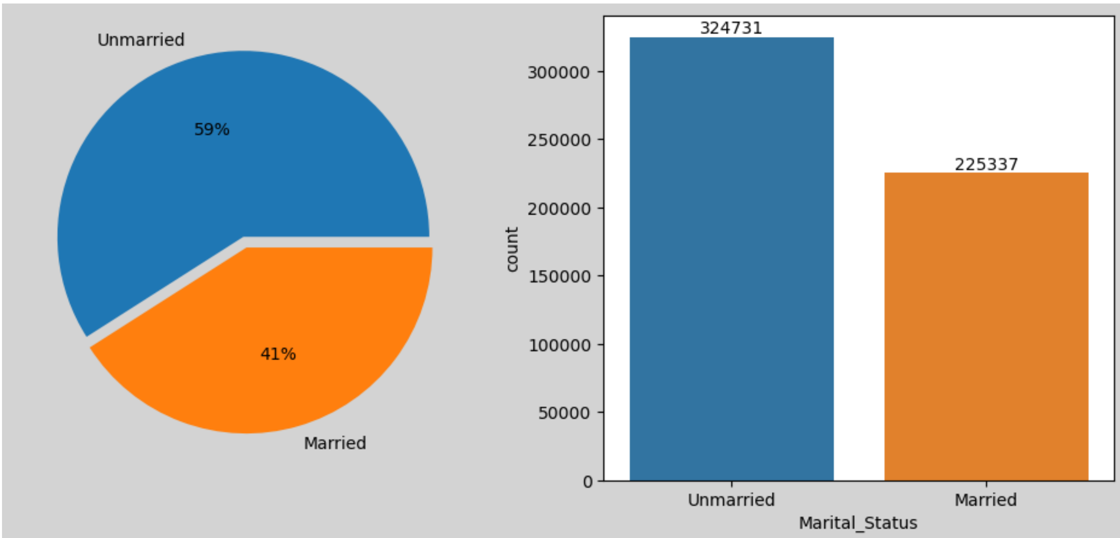
We can observe that in dataset for marital_status column there values 0 and 1.

0 means Unmarried and 1 means Married. So lets replace these values in the dataset.

[24]:
```
df['Marital_Status'].replace(to_replace = 0, value = 'Unmarried', inplace =␣
 ↪True)
df['Marital_Status'].replace(to_replace = 1, value = 'Married', inplace = True)
```

[25]:
```
plt.figure(figsize = (12,5)).set_facecolor("lightgrey")

plt.subplot(1,2,1)
labels = ['Unmarried','Married']
plt.pie(df.groupby('Marital_Status')['Marital_Status'].count(), labels =␣
 ↪labels, explode = (0.06,0), autopct = '%0.0f%%')

plt.subplot(1,2,2)
```

```python
label = sns.countplot(data = df, x='Marital_Status')
for i in label.containers:
    label.bar_label(i)

plt.show()
```



We can observe that 59% of the frequent buyers are of unmarried people, while 41% of married.

There are an approximate of 0.32 million entries for unmarried people and 0.22 million for married people.

[26]:
```python
round(df['Purchase'].describe(),2)
```

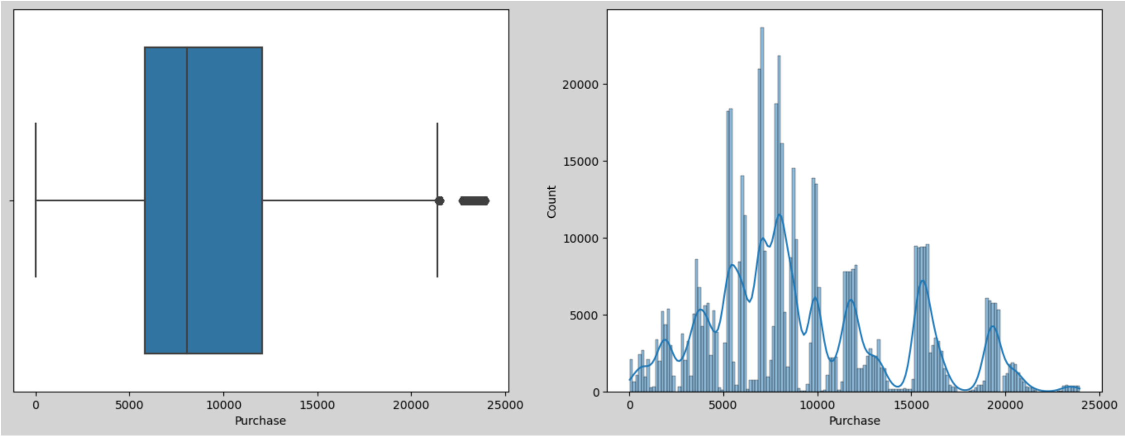
[26]:
```
count    550068.00
mean       9263.97
std        5023.07
min          12.00
25%        5823.00
50%        8047.00
75%       12054.00
max       23961.00
Name: Purchase, dtype: float64
```

While observing their spending habits of all buyers

```
The average order value is 9263.97

While 50% of the buyers spend an approximate of 8047.

The lowest order value is as low as 12.
```

While, the highest order value is of 23961.

[27]:
```
plt.figure(figsize=(17, 6)).set_facecolor("lightgrey")

plt.subplot(1,2,1)
sns.boxplot(data=df, x='Purchase', orient='h')

plt.subplot(1,2,2)
sns.histplot(data=df, x='Purchase', kde=True)
plt.show()
```



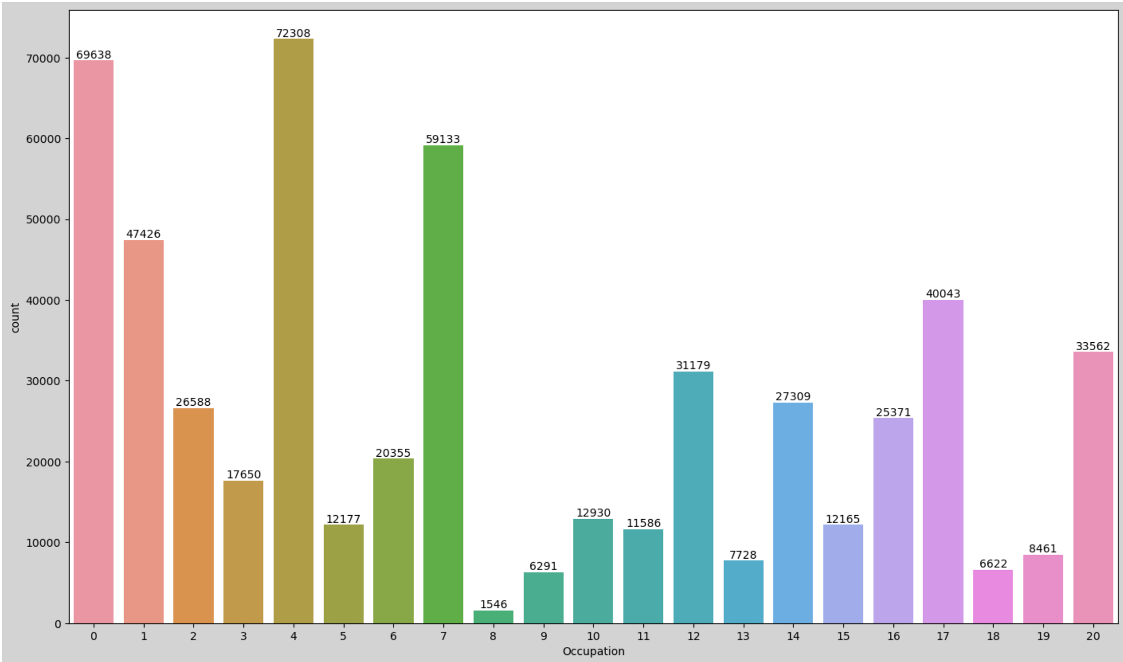
While observing the purchase values of the orders we can infer that

Most of the values lies between 6000 and 12000.

Most order values lies in the range of 5000 - 10000

There are more orders in the range 15000 - 16000 followed by 11000 - 11500 range and a few also

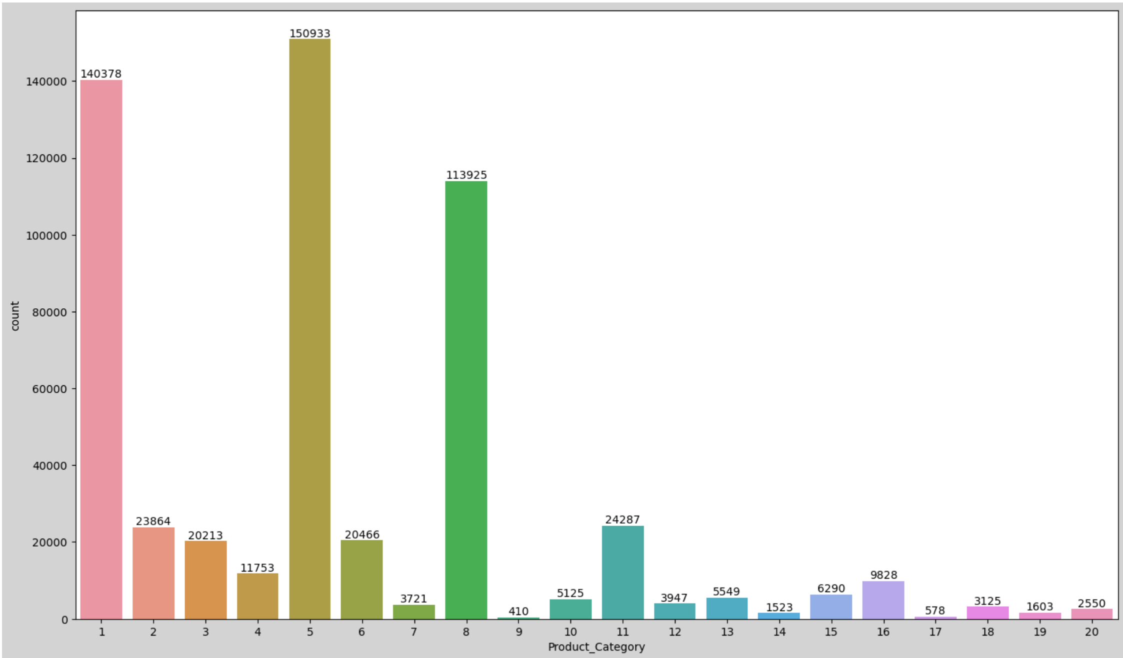
[28]:
```
plt.figure(figsize=(17, 10)).set_facecolor("lightgrey")
label = sns.countplot(data = df, x='Occupation')
for i in label.containers:
    label.bar_label(i)
```



People having occupation 4 are the most frequent buyers followed by occupation 0 and 7.

People having occupation 8 are the least frequent buyers followed by occupation 9 and 18.

[29]:
```
plt.figure(figsize=(17, 10)).set_facecolor("lightgrey")
label = sns.countplot(data = df, x='Product_Category')
for i in label.containers:
    label.bar_label(i)
```



The most frequent bought product category is 5 followed by 1 and 8.

All the other categories are not much touched.

The least frequent bought are category 9 followed by 17 and 14.

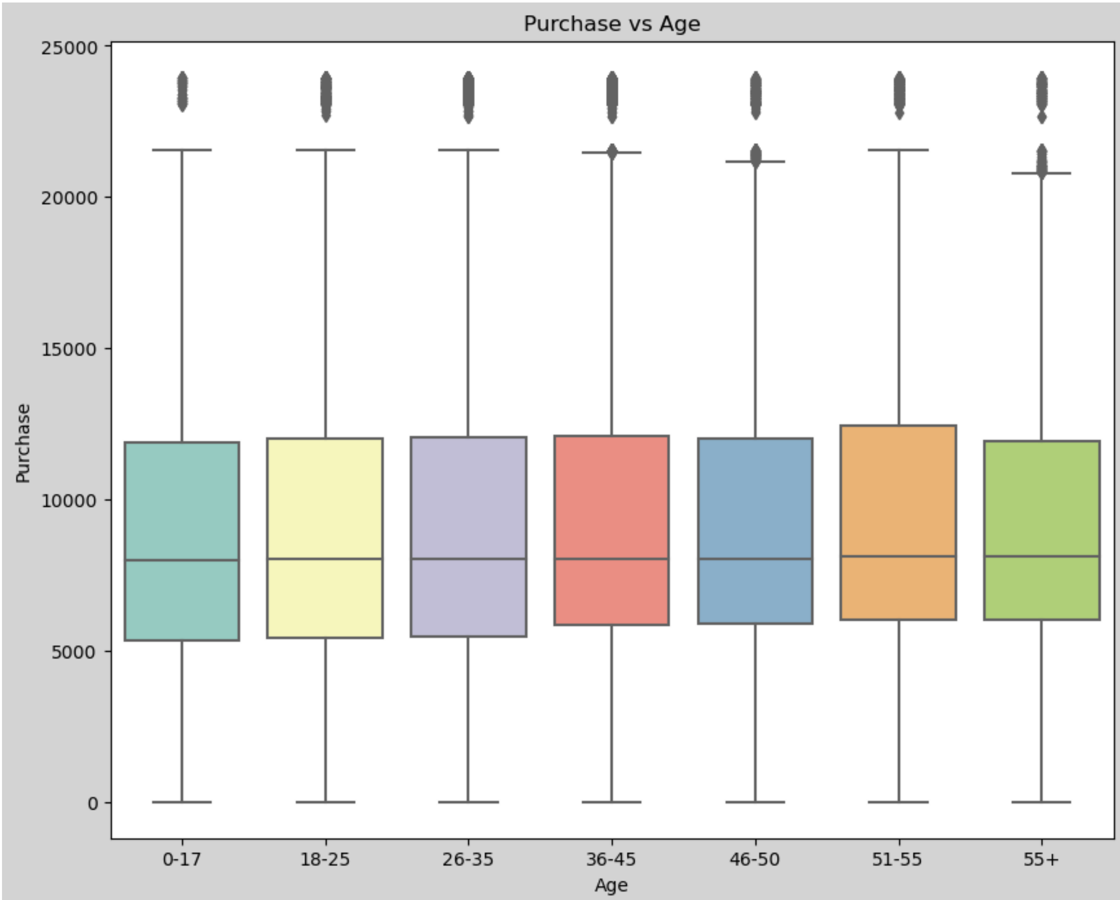
1.2 Bi-variate Analysis

[30]:
```
plt.figure(figsize = (10,6)).set_facecolor("lightgrey")
sns.boxplot(data = df, y ='Purchase', x = 'Gender', palette = 'Set3')
plt.title('Purchase vs Gender')
plt.show()
```

[31]:
```
df.groupby(['Gender'])['Purchase'].describe()
```

[31]:
```
           count         mean          std   min     25%     50%      75%  \
Gender
F       135809.0  8734.565765  4767.233289  12.0  5433.0  7914.0  11400.0
M       414259.0  9437.526040  5092.186210  12.0  5863.0  8098.0  12454.0

            max
Gender
F       23959.0
M       23961.0
```

[32]:
```
plt.figure(figsize = (10,8)).set_facecolor("lightgrey")
sns.boxplot(data = df, y ='Purchase', x = 'Age', palette = 'Set3')
plt.title('Purchase vs Age')
plt.show()
```



[33]:
```
df.groupby(['Age'])['Purchase'].describe()
```

[33]:

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

           max
Age
0-17   23955.0
18-25  23958.0
26-35  23961.0
36-45  23960.0
46-50  23960.0
```

```
51-55   23960.0
55+     23960.0
```

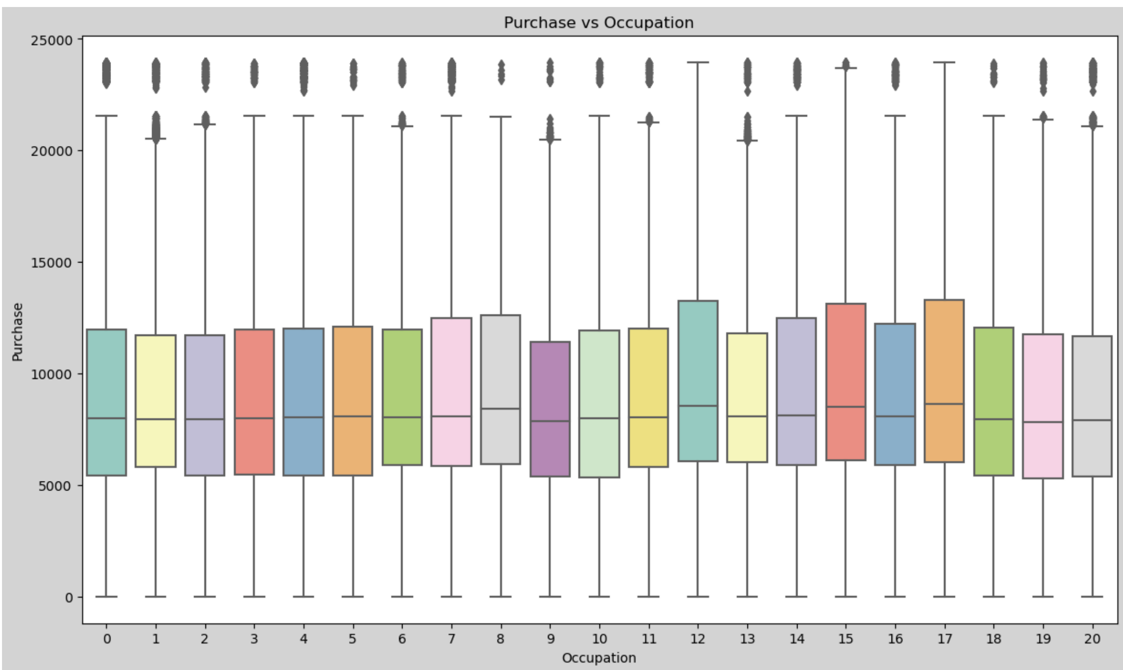
The average order value is highest for age group 51-55 which is around 9534.

While, the average amount is lowest for age group 0-17 which is arouns 8933.

The highest order value for all the groups is around 23960.

The losest order value is 12 for all the groups.

[34]:
```
plt.figure(figsize = (14,8)).set_facecolor("lightgrey")
sns.boxplot(data = df, y ='Purchase', x = 'Occupation', palette = 'Set3')
plt.title('Purchase vs Occupation')
plt.show()
```



[35]:
```
df.groupby(['Occupation'])['Purchase'].describe()
```

[35]:

```
              count         mean          std   min       25%     50%  \
Occupation
0           69638.0  9124.428588  4971.757402  12.0  5445.00  8001.0
1           47426.0  8953.193270  4838.482159  12.0  5825.00  7966.0
2           26588.0  8952.481683  4939.418663  12.0  5419.00  7952.0
3           17650.0  9178.593088  5000.942719  12.0  5478.00  8008.0
4           72308.0  9213.980251  5043.674855  12.0  5441.75  8043.0
5           12177.0  9333.149298  5025.616603  12.0  5452.00  8080.0
```

6	20355.0	9256.535691	4989.216005	12.0	5888.00	8050.0
7	59133.0	9425.728223	5086.097089	12.0	5878.00	8069.0
8	1546.0	9532.592497	4916.641374	14.0	5961.75	8419.5
9	6291.0	8637.743761	4653.290986	13.0	5403.00	7886.0
10	12930.0	8959.355375	5124.339999	12.0	5326.25	8012.5
11	11586.0	9213.845848	5103.802992	12.0	5835.75	8041.5
12	31179.0	9796.640239	5140.437446	12.0	6054.00	8569.0
13	7728.0	9306.351061	4940.156591	12.0	6038.00	8090.5
14	27309.0	9500.702772	5069.600234	12.0	5922.00	8122.0
15	12165.0	9778.891163	5088.424301	12.0	6109.00	8513.0
16	25371.0	9394.464349	4995.918117	12.0	5917.00	8070.0
17	40043.0	9821.478236	5137.024383	12.0	6012.00	8635.0
18	6622.0	9169.655844	4987.697451	12.0	5420.00	7955.0
19	8461.0	8710.627231	5024.181000	12.0	5292.00	7840.0
20	33562.0	8836.494905	4919.662409	12.0	5389.00	7903.5

	75%	max
Occupation		
0	11957.00	23961.0
1	11702.75	23960.0
2	11718.00	23955.0
3	11961.00	23914.0
4	12034.00	23961.0
5	12091.00	23924.0
6	11971.50	23951.0
7	12486.00	23948.0
8	12607.00	23869.0
9	11436.00	23943.0
10	11931.75	23955.0
11	12010.00	23946.0
12	13239.00	23960.0
13	11798.50	23959.0
14	12508.00	23941.0
15	13150.00	23949.0
16	12218.50	23947.0
17	13292.50	23961.0
18	12062.75	23894.0
19	11745.00	23939.0
20	11677.00	23960.0

But, here we can observe that the highest median value is for occupation 17

The lowest median value is for occupation 19.

Occupation 17 have the high average order values compared to other occupations which is 9821.

Occupation 9 have the lowest average order value which is 8637.

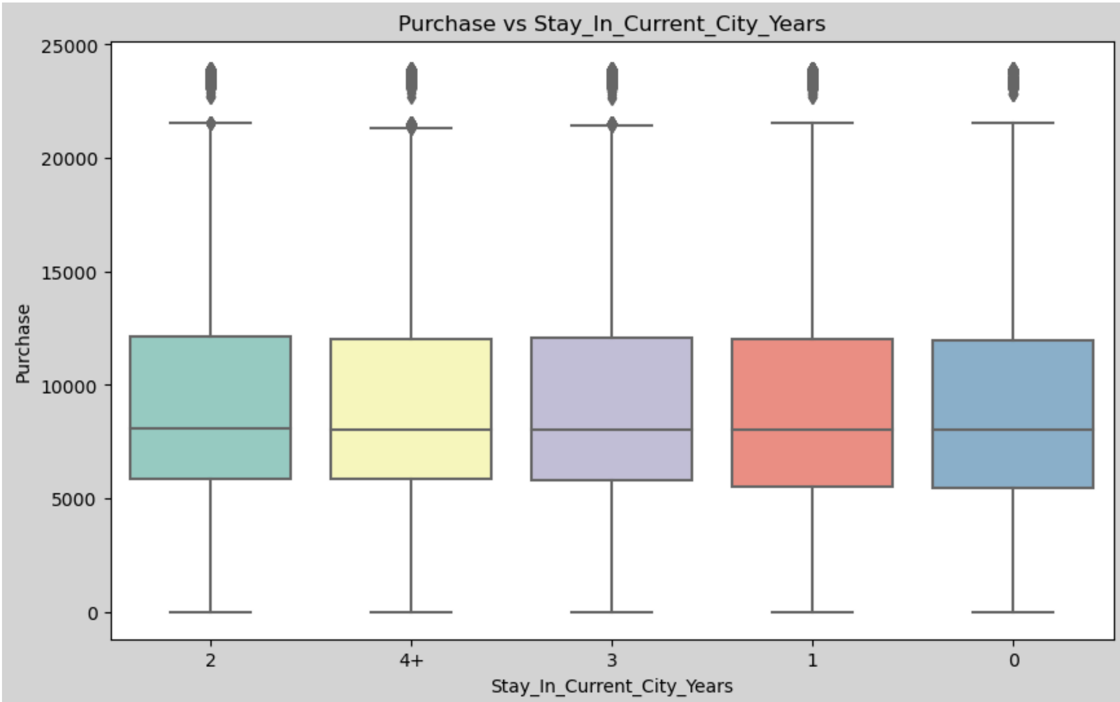
[36]:
```
plt.figure(figsize = (10,6)).set_facecolor("lightgrey")
sns.boxplot(data = df, y ='Purchase', x = 'City_Category', palette = 'Set3')
plt.title('Purchase vs City_Category')
plt.show()
```

[37]:
```
df.groupby(['City_Category'])['Purchase'].describe()
```

[37]:

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

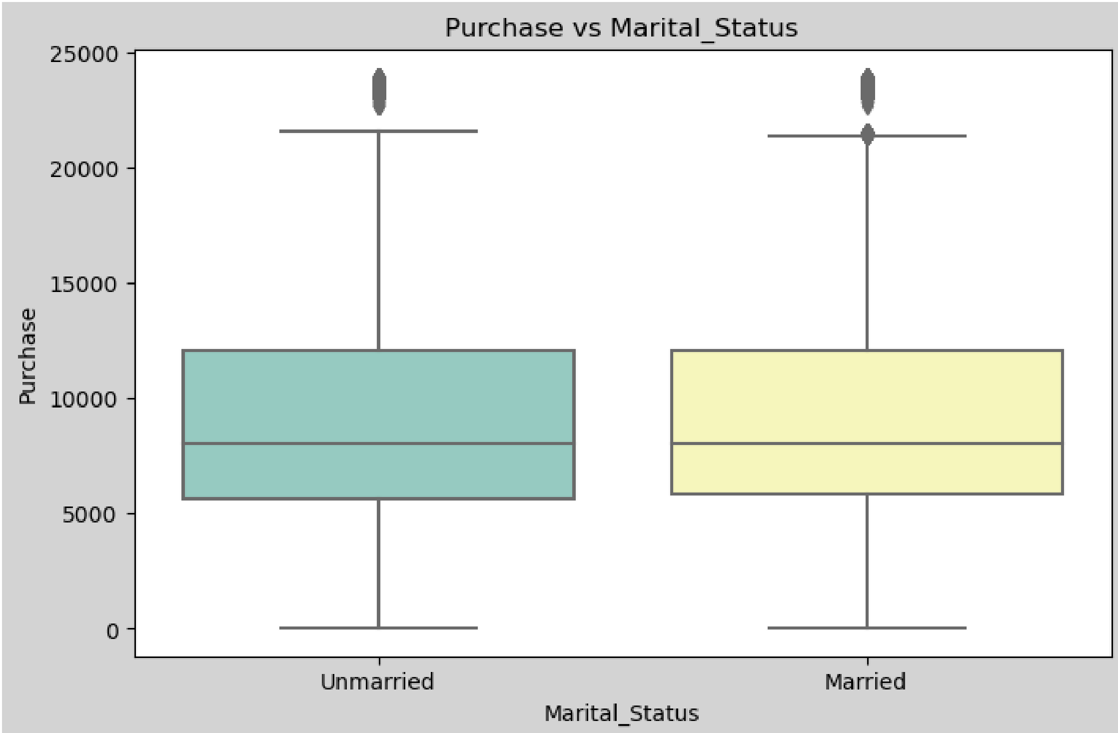
[38]:
```
plt.figure(figsize = (10,6)).set_facecolor("lightgrey")
sns.boxplot(data = df, y ='Purchase', x = 'Stay_In_Current_City_Years', palette
↪= 'Set3')
plt.title('Purchase vs Stay_In_Current_City_Years')
plt.show()
```



[39]:
```
df.groupby(['Stay_In_Current_City_Years'])['Purchase'].describe()
```

[39]:
```
                               count         mean          std   min     25%  \
Stay_In_Current_City_Years
0                            74398.0  9180.075123  4990.479940  12.0  5480.0
1                           193821.0  9250.145923  5027.476933  12.0  5500.0
2                           101838.0  9320.429810  5044.588224  12.0  5846.0
3                            95285.0  9286.904119  5020.343541  12.0  5832.0
4+                           84726.0  9275.598872  5017.627594  12.0  5844.0

                               50%      75%      max
Stay_In_Current_City_Years
0                           8025.0  11990.0  23960.0
1                           8041.0  12042.0  23961.0
2                           8072.0  12117.0  23961.0
3                           8047.0  12075.0  23961.0
4+                          8052.0  12038.0  23958.0
```

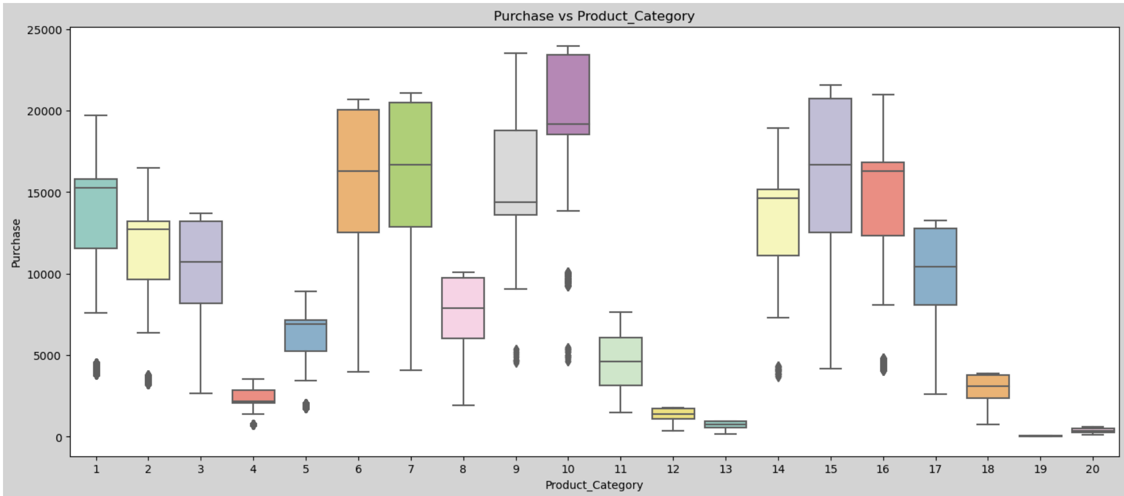
[40]:
```
plt.figure(figsize = (8,5)).set_facecolor("lightgrey")
sns.boxplot(data = df, y ='Purchase', x = 'Marital_Status', palette = 'Set3')
plt.title('Purchase vs Marital_Status')
plt.show()
```



[41]:
```
df.groupby(['Marital_Status'])['Purchase'].describe()
```

[41]:
```
                   count         mean          std  min     25%     50%  \
Marital_Status
Unmarried       324731.0  9265.907619  5027.347859  12.0  5605.0  8044.0
Married         225337.0  9261.174574  5016.897378  12.0  5843.0  8051.0

                    75%      max
Marital_Status
Unmarried       12061.0  23961.0
Married         12042.0  23961.0
```

[42]:
```
plt.figure(figsize = (17,7)).set_facecolor("lightgrey")
sns.boxplot(data = df, y ='Purchase', x = 'Product_Category', palette = 'Set3')
plt.title('Purchase vs Product_Category')
plt.show()
```



[43]:
```
df.groupby(['Product_Category'])['Purchase'].describe()
```

[43]:

Product_Category	count	mean	std	min	25%
1	140378.0	13606.218596	4298.834894	3790.0	11546.00
2	23864.0	11251.935384	3570.642713	3176.0	9645.75
3	20213.0	10096.705734	2824.626957	2638.0	8198.00
4	11753.0	2329.659491	812.540292	684.0	2058.00
5	150933.0	6240.088178	1909.091687	1713.0	5242.00
6	20466.0	15838.478550	4011.233690	3981.0	12505.00
7	3721.0	16365.689600	4174.554105	4061.0	12848.00
8	113925.0	7498.958078	2013.015062	1939.0	6036.00
9	410.0	15537.375610	5330.847116	4528.0	13583.50
10	5125.0	19675.570927	4225.721898	4624.0	18546.00
11	24287.0	4685.268456	1834.901184	1472.0	3131.00
12	3947.0	1350.859894	362.510258	342.0	1071.00
13	5549.0	722.400613	183.493126	185.0	578.00
14	1523.0	13141.625739	4069.009293	3657.0	11097.00
15	6290.0	14780.451828	5175.465852	4148.0	12523.25
16	9828.0	14766.037037	4360.213198	4036.0	12354.00
17	578.0	10170.759516	2333.993073	2616.0	8063.50
18	3125.0	2972.864320	727.051652	754.0	2359.00
19	1603.0	37.041797	16.869148	12.0	24.00
20	2550.0	370.481176	167.116975	118.0	242.00

Product_Category	50%	75%	max
1	15245.0	15812.00	19708.0
2	12728.5	13212.00	16504.0
3	10742.0	13211.00	13717.0

```
4                    2175.0   2837.00   3556.0
5                    6912.0   7156.00   8907.0
6                   16312.0  20051.00  20690.0
7                   16700.0  20486.00  21080.0
8                    7905.0   9722.00  10082.0
9                   14388.5  18764.00  23531.0
10                  19197.0  23438.00  23961.0
11                   4611.0   6058.00   7654.0
12                   1401.0   1723.00   1778.0
13                    755.0    927.00    962.0
14                  14654.0  15176.50  18931.0
15                  16660.0  20745.75  21569.0
16                  16292.5  16831.00  20971.0
17                  10435.5  12776.75  13264.0
18                   3071.0   3769.00   3900.0
19                     37.0     50.00     62.0
20                    368.0    490.00    613.0
```

The median value for product category 10 is the highest which is 19197.

The median value for product category 19 is the lowest which is only 37.

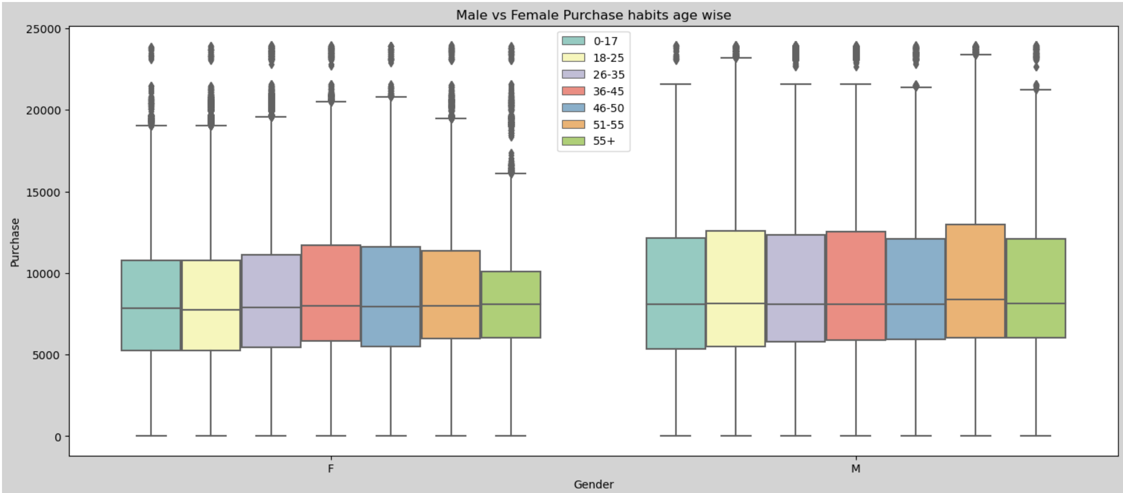
The average order value for category 10 is the highest which is 19675.

The average order value for category 19 is also the lowest which is 37.

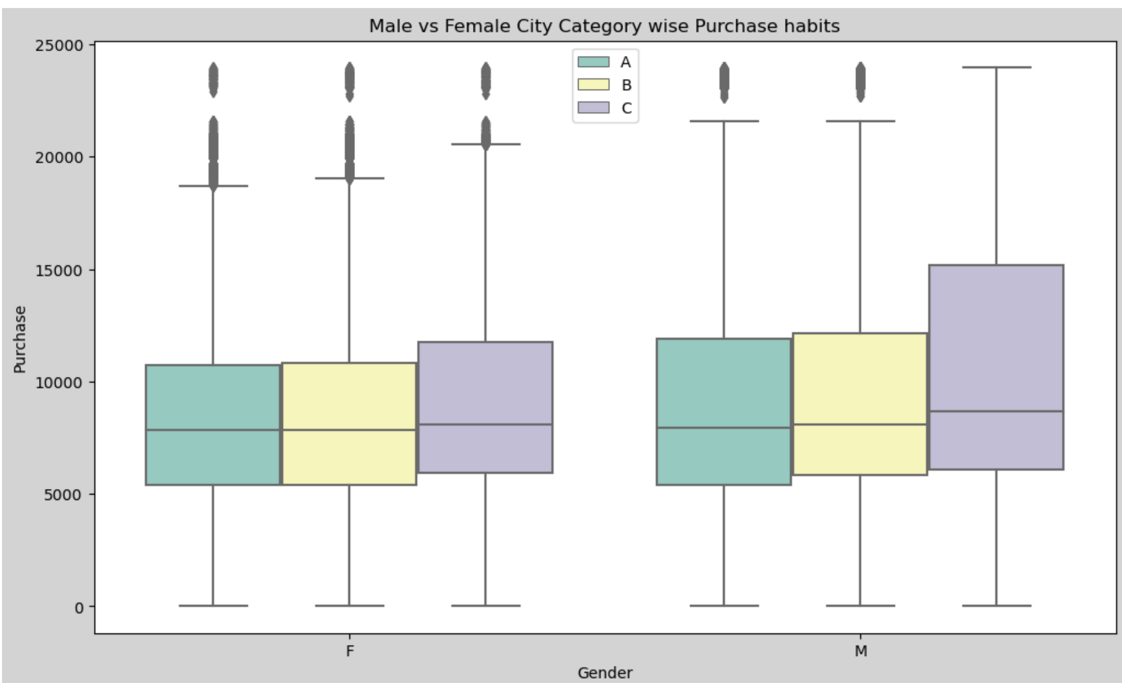
Clearly, category 19 is the least preferred or least frequent bought product category.

1.3 Multi-variate Analysis

[44]:
```python
plt.figure(figsize = (17,7)).set_facecolor("lightgrey")
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Age', palette='Set3')
plt.legend(loc=9)
plt.title('Male vs Female Purchase habits age wise')

plt.show()
```

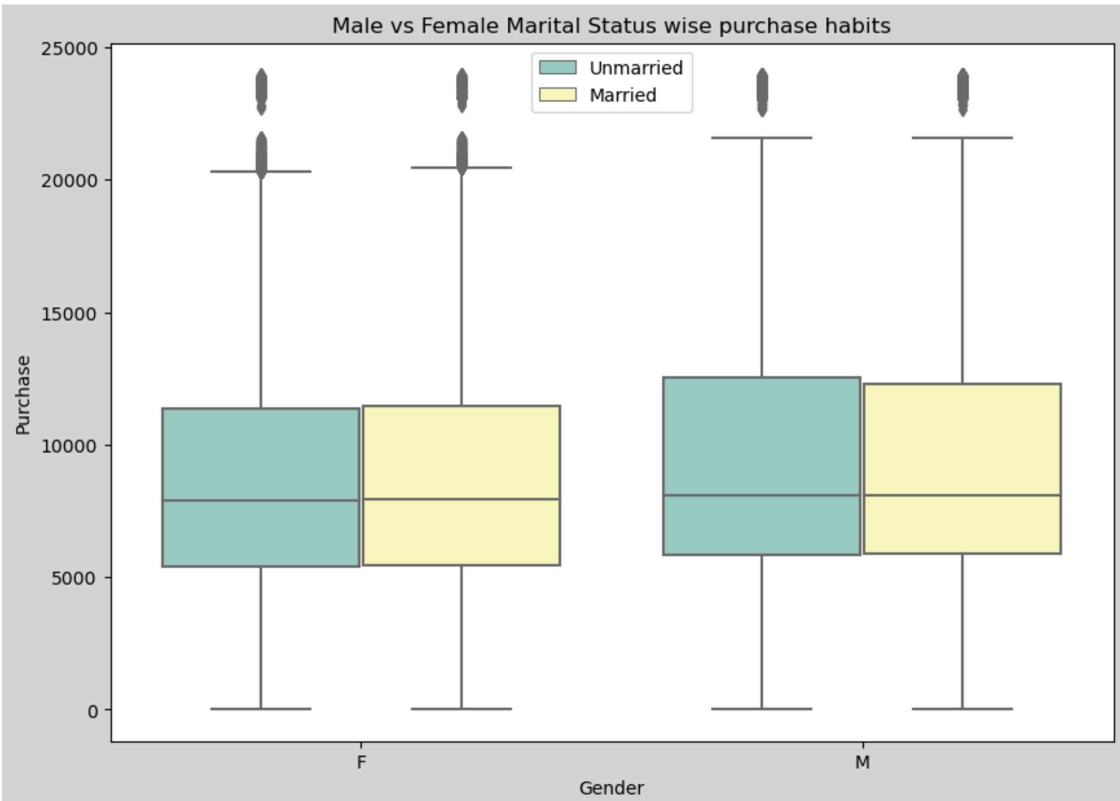


[45]:
```
plt.figure(figsize = (12,7)).set_facecolor("lightgrey")
sns.boxplot(data=df, y='Purchase', x='Gender', hue='City_Category', palette='Set3')
plt.legend(loc=9)
plt.title("Male vs Female City Category wise Purchase habits")
plt.show()
```



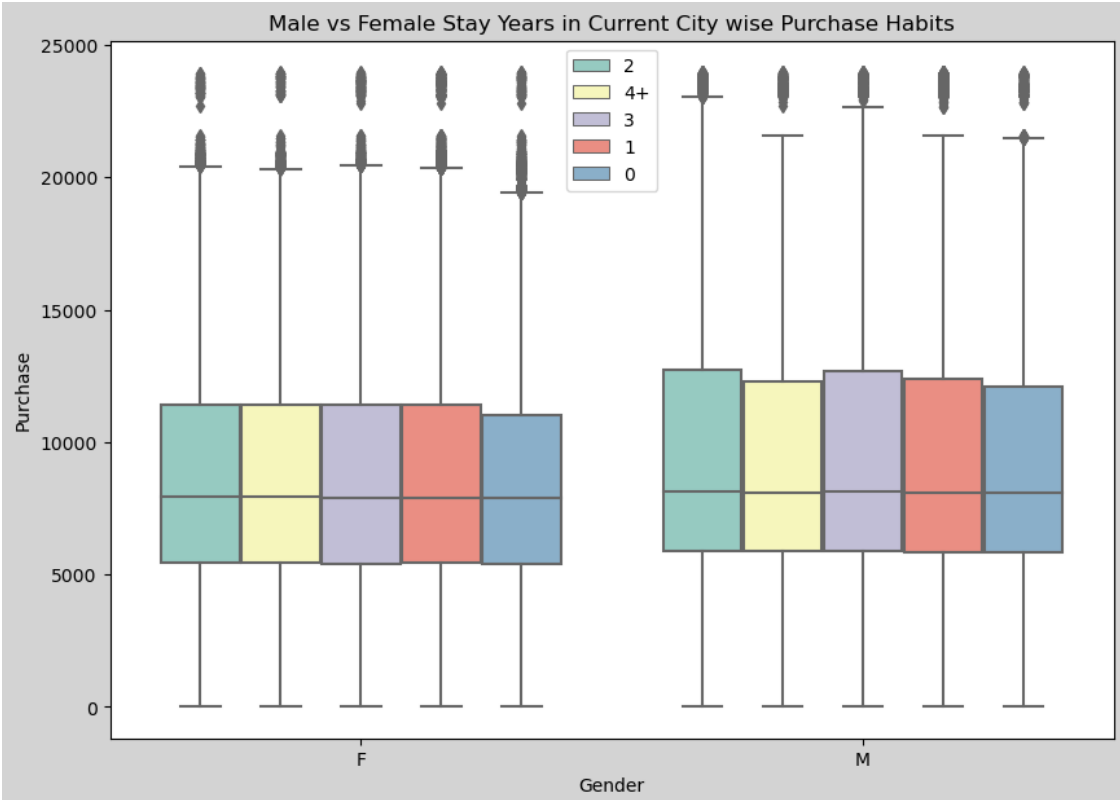
[46]:

```
plt.figure(figsize = (10,7)).set_facecolor("lightgrey")
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Marital_Status',␣
 ↪palette='Set3')
plt.legend(loc=9)
plt.title('Male vs Female Marital Status wise purchase habits')
plt.show()
```



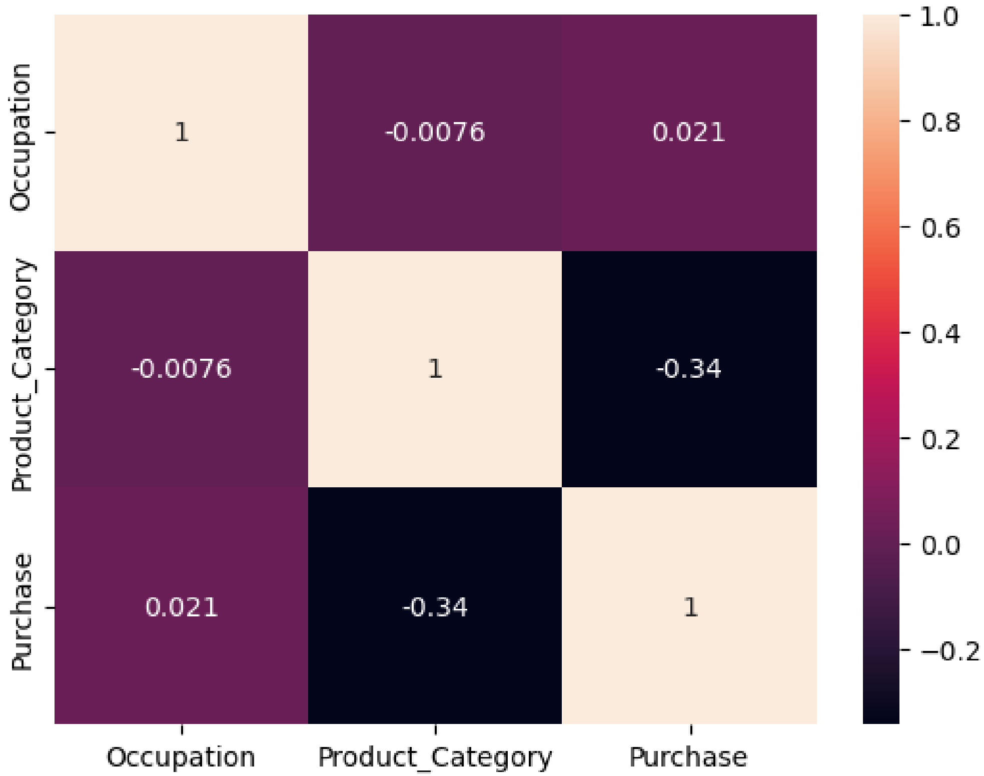
[47]:

```
plt.figure(figsize = (10,7)).set_facecolor("lightgrey")
sns.boxplot(data=df, y='Purchase', x='Gender',␣
 ↪hue='Stay_In_Current_City_Years', palette='Set3')
plt.legend(loc=9)
plt.title('Male vs Female Stay Years in Current City wise Purchase Habits')
plt.show()
```



Lets check the Correlation in the numerical values of the dataset.

[48]:
```
sns.heatmap(df.corr(), annot = True)
plt.show()
```



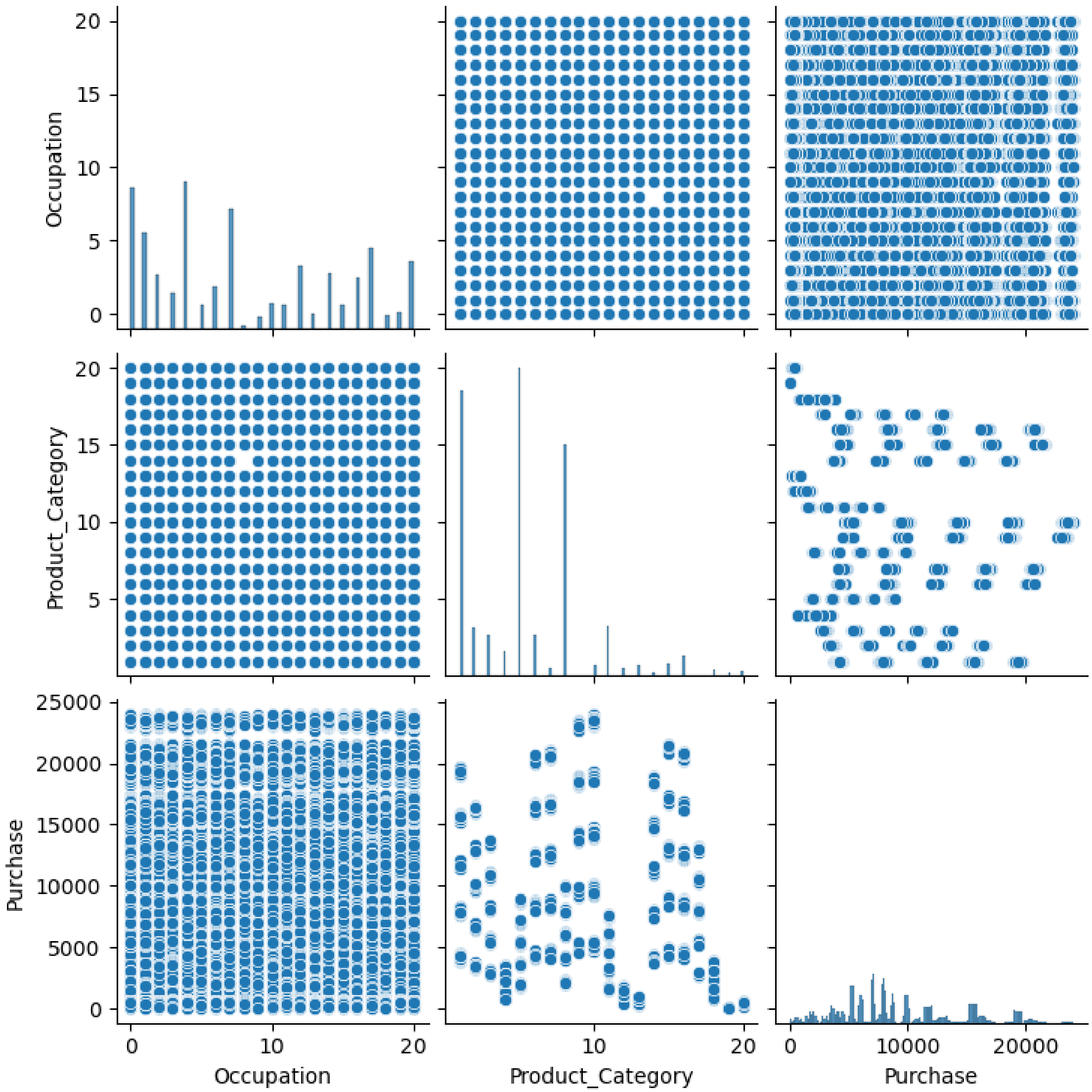
We can observe that there is

`High Negative Correlation(-0.0076) between Product Category and Occupation.`

`Slight Positive Correlation(0.021) between Purchase and Occupation.`

`Negative Correlation(-0.34) between Product Category and Purchase.`

[49]:
```
sns.pairplot(df)
plt.show()
```



2 Central Limit Theorom

[50]:
```
def bootstrap(sample1,sample2,sample_size,itr_size=1000,ci=90):
    ci = ci/100

    plt.figure(figsize=(16,8))
    sample1_n = [np.mean(sample1.sample(sample_size)) for i in range(itr_size)]
    sample2_n = [np.mean(sample2.sample(sample_size)) for i in range(itr_size)]

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

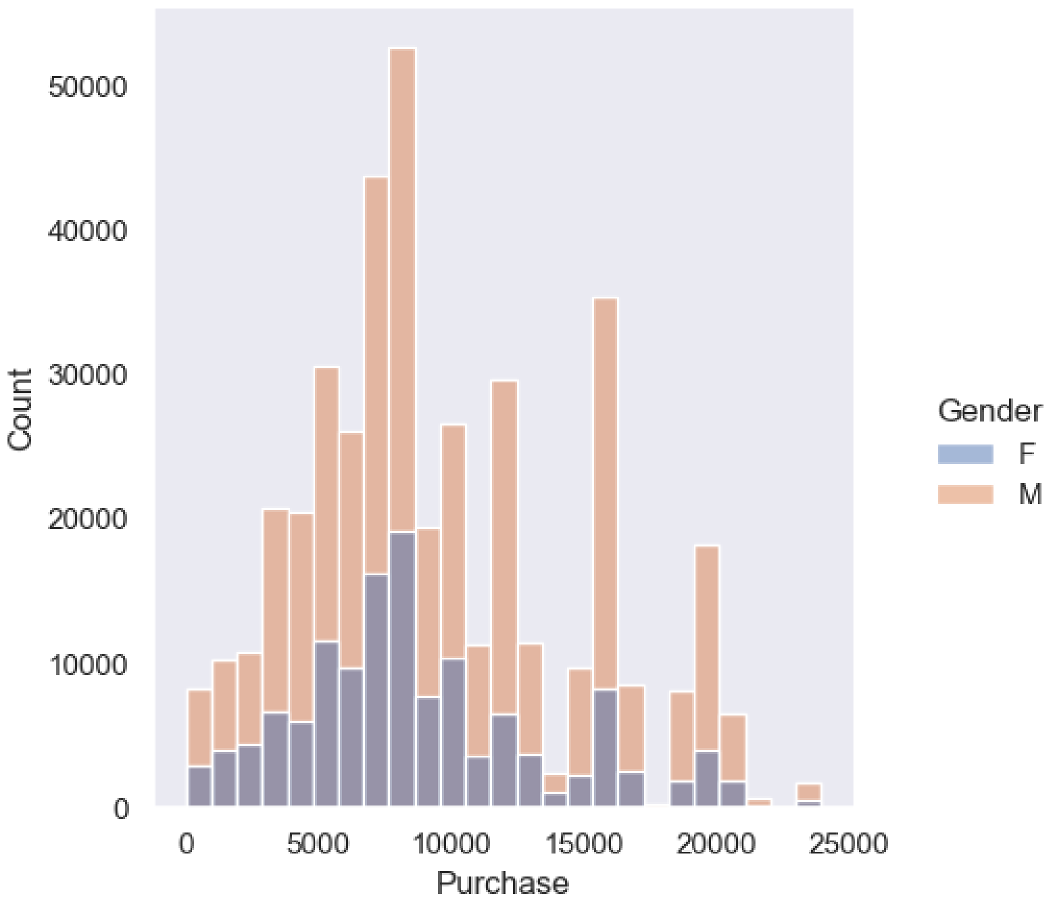
```python
    sem1 = stats.sem(sample1_n)

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

    # For Sample2's means
    mean2 = np.mean(sample2_n)
    sigma2 = np.std(sample2_n)
    sem2 = stats.sem(sample2_n)

    lower_limit_2 = norm.ppf((1-ci)/2) * sigma2 + mean2
    upper_limit_2 = norm.ppf(ci + (1-ci)/2) * sigma2 + mean2

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

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

    plt.title(f"Sample Size: {sample_size}, Male Avg: {np.round(mean1, 2)}, Male SME: {np.round(sem1,2)}, Female Avg:{np.round(mean2, 2)}, Female SME: {np.round(sem2,2)}")
    plt.legend(loc = 'upper right')
    plt.xlabel('Purchase')
    plt.ylabel('Density')


    return round(mean1,2), round(mean2,2), round(lower_limit_1,2), round(upper_limit_1,2), round(lower_limit_2,2), round(upper_limit_2,2)
```

[51]:
```
df_male = df[df['Gender']=='M']
df_female = df[df['Gender']=='F']
```

2.1 Male Vs Female Purchase Values

[52]:
```
plt.figure(figsize=(12,8))
sns.set(style='dark')
sns.displot(x= 'Purchase',data=df,hue='Gender',bins=25)
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



[53]:
```
df.groupby(['Gender'])['Purchase'].describe()
```

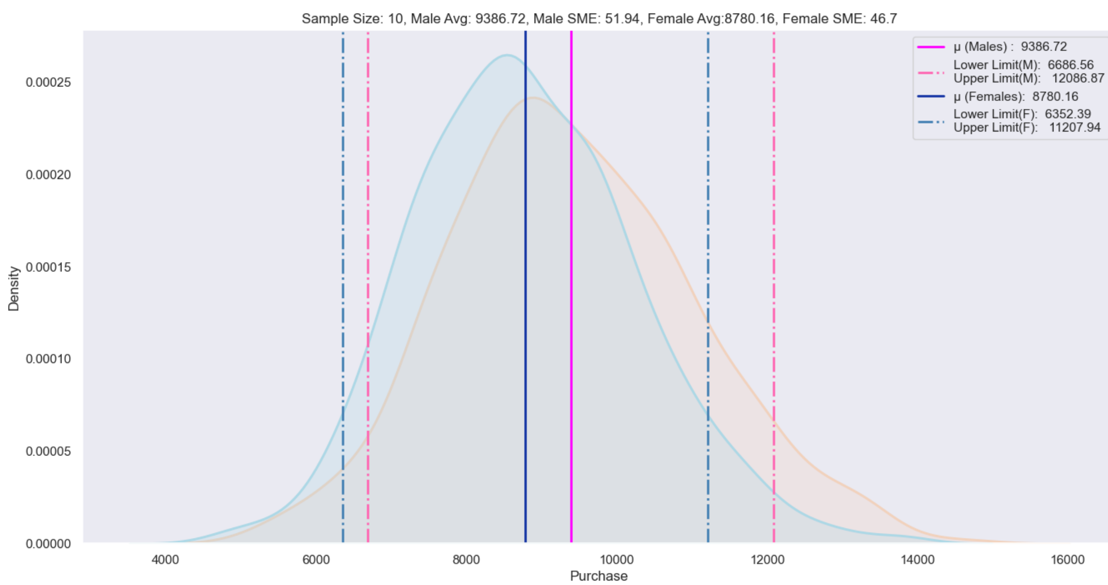
[53]:

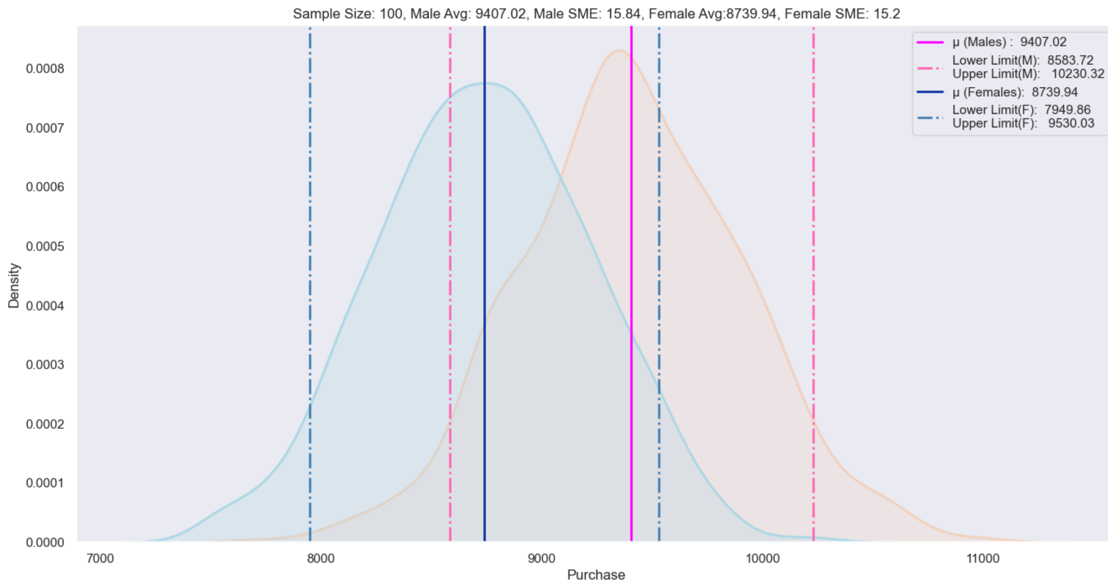
Gender	count	mean	std	min	25%	50%	75%	\
F	135809.0	8734.565765	4767.233289	12.0	5433.0	7914.0	11400.0	
M	414259.0	9437.526040	5092.186210	12.0	5863.0	8098.0	12454.0	

```
           max
Gender
F      23959.0
M      23961.0
```

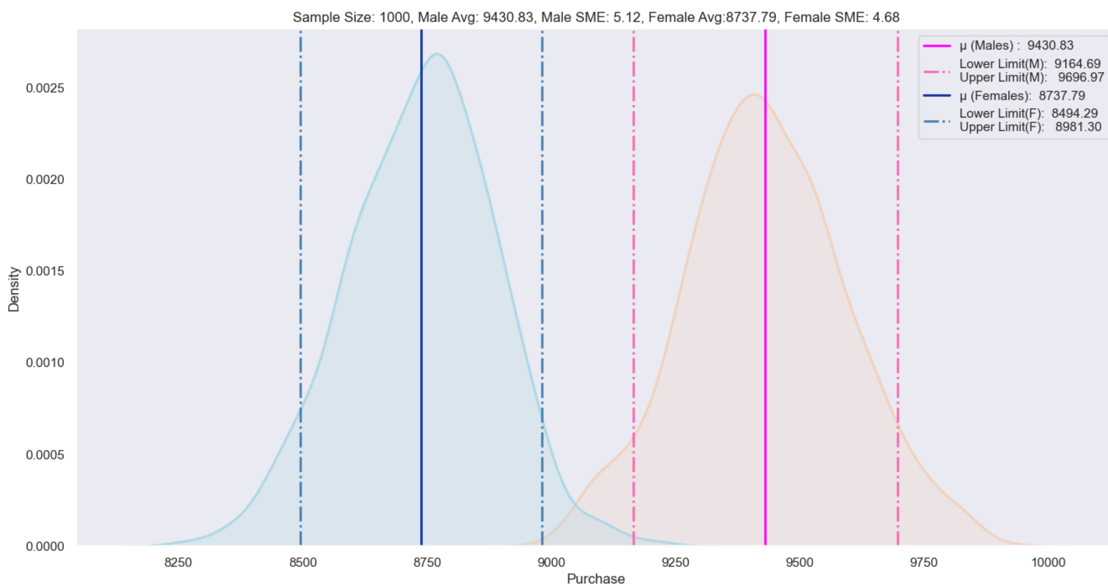
[54]:
```python
sample_sizes = [10,100,1000,10000,100000]
ci = 90
itr_size = 1000
```

[55]:
```python
res = pd.DataFrame(columns = ['Gender','Sample Size','Lower Limit','Upper␣
↪Limit','Sample Mean','Confidence Interval','Interval Range','Range'])

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

    res = res.append({'Gender':'M','Sample Size':i,'Lower Limit':ll_m,'Upper␣
↪Limit':ul_m,'Sample Mean':m_avg,'Confidence Interval':ci,'Interval Range':
↪[ll_m,ul_m],'Range': ul_m-ll_m}, ignore_index = True)
    res = res.append({'Gender':'F','Sample Size':i,'Lower Limit':ll_f,'Upper␣
↪Limit':ul_f,'Sample Mean':f_avg,'Confidence Interval':ci,'Interval Range':
↪[ll_f,ul_f],'Range': ul_f-ll_f}, ignore_index = True)
```

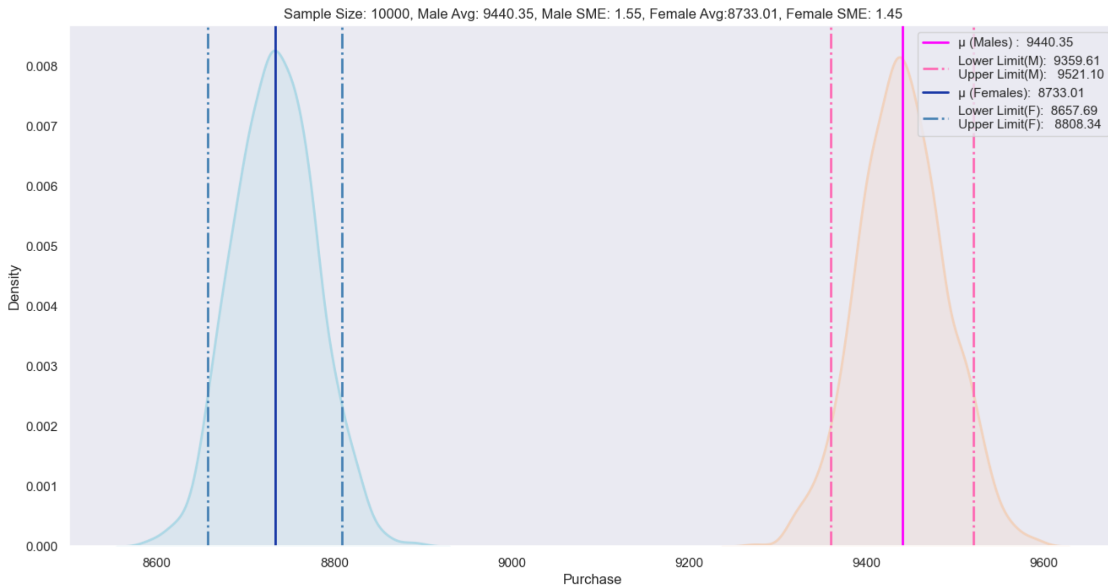




Sample Size: 100, Male Avg: 9407.02, Male SME: 15.84, Female Avg:8739.94, Female SME: 15.2
μ (Males) : 9407.02
Lower Limit(M): 8583.72
Upper Limit(M): 10230.32
μ (Females): 8739.94
Lower Limit(F): 7949.86
Upper Limit(F): 9530.03
Density
Purchase



Sample Size: 1000, Male Avg: 9430.83, Male SME: 5.12, Female Avg:8737.79, Female SME: 4.68
μ (Males) : 9430.83
Lower Limit(M): 9164.69
Upper Limit(M): 9696.97
μ (Females): 8737.79
Lower Limit(F): 8494.29
Upper Limit(F): 8981.30
Density
Purchase

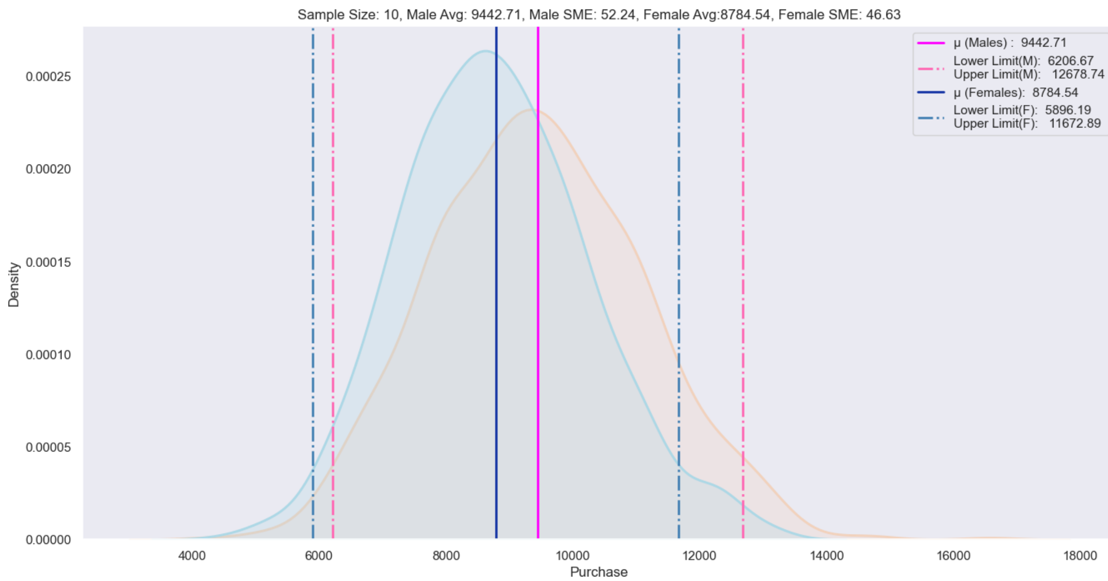


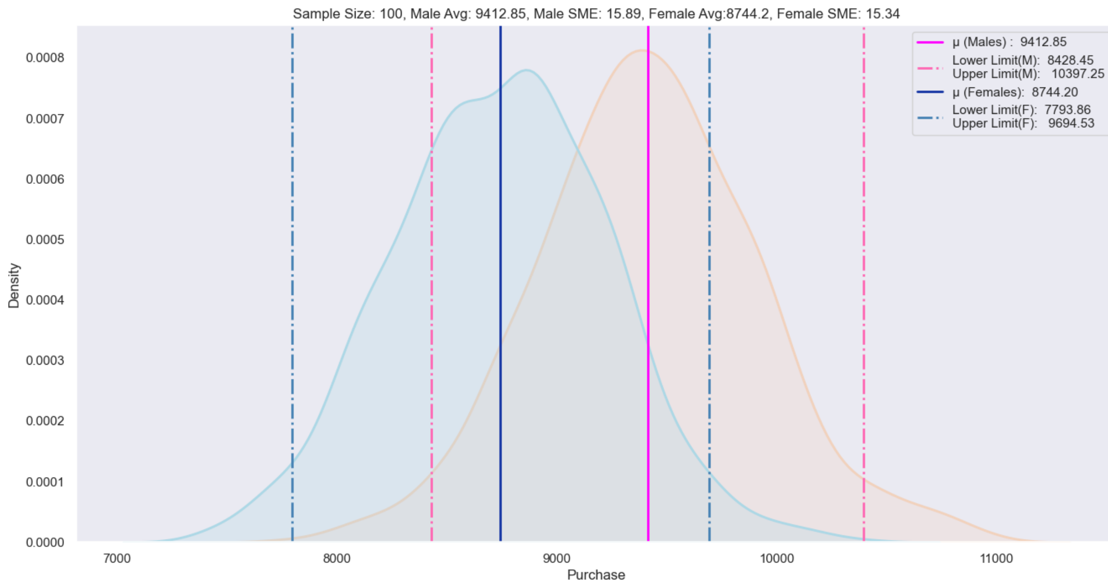
We can observe that as the sample size increases,

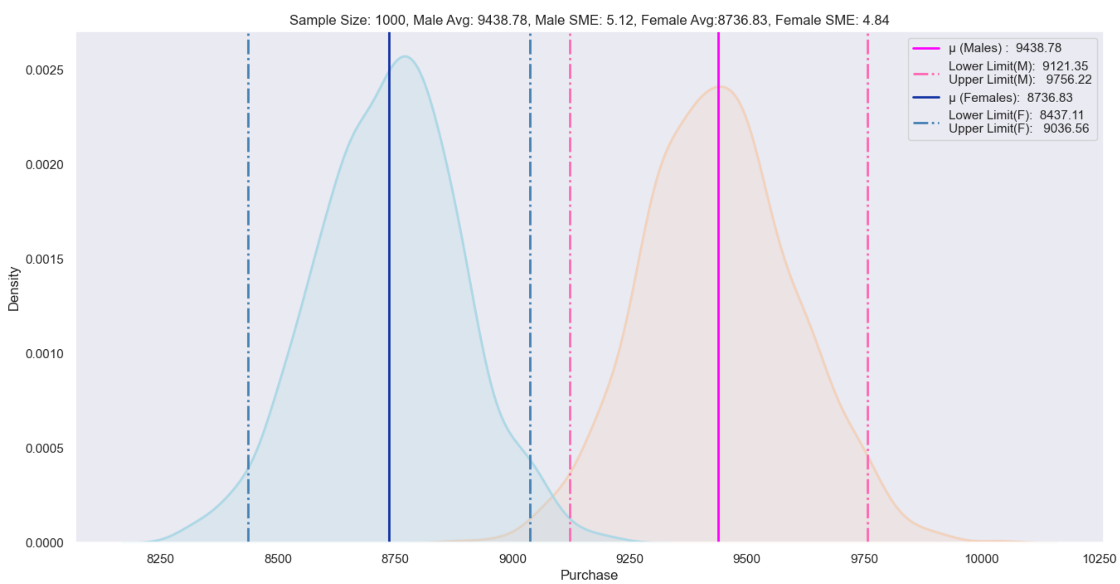
```
The average for both of them change significantly.

Both the plots start to seperate and become distinct.
```

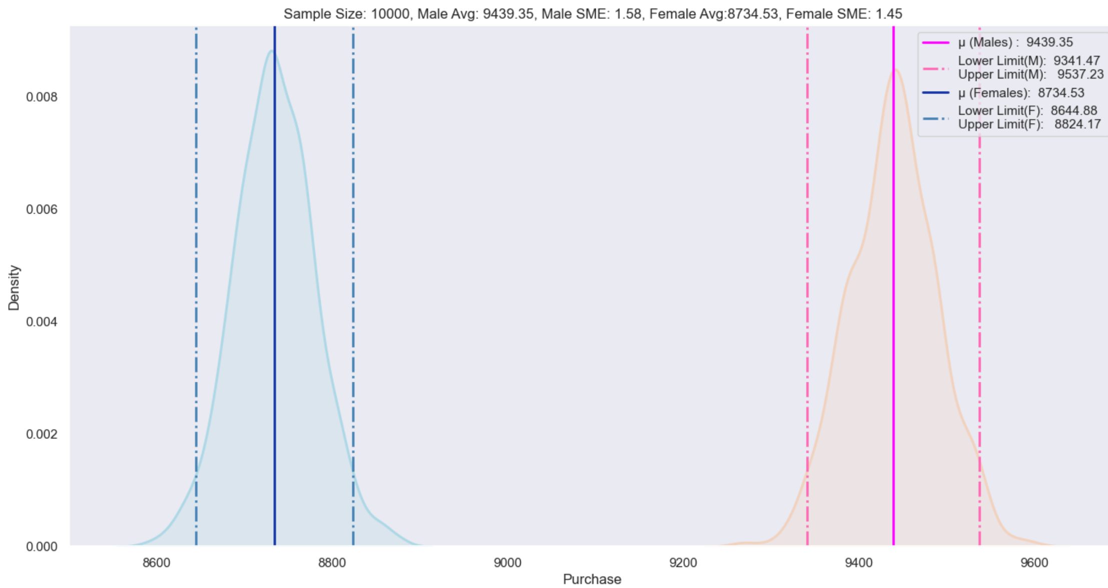
[56]:
```python
sample_sizes = [10,100,1000,10000,100000]
ci = 95
itr_size = 1000
```

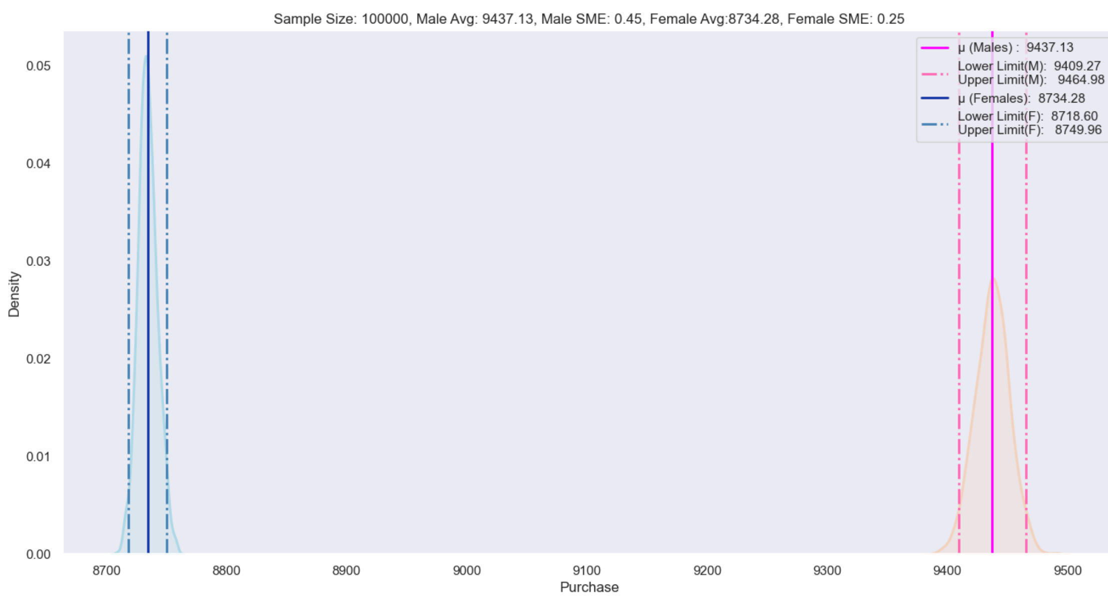
```
for i in sample_sizes:
    m_avg, f_avg, ll_m, ul_m, ll_f, ul_f =␣
 ↪bootstrap(df_male['Purchase'],df_female['Purchase'],i,itr_size,ci)

    res = res.append({'Gender':'M','Sample Size':i,'Lower Limit':ll_m,'Upper␣
 ↪Limit':ul_m,'Sample Mean':m_avg,'Confidence Interval':ci,'Interval Range':
 ↪[ll_m,ul_m],'Range': ul_m-ll_m}, ignore_index = True)
    res = res.append({'Gender':'F','Sample Size':i,'Lower Limit':ll_f,'Upper␣
 ↪Limit':ul_f,'Sample Mean':f_avg,'Confidence Interval':ci,'Interval Range':
 ↪[ll_f,ul_f],'Range': ul_f-ll_f}, ignore_index = True)
```




Sample Size: 100, Male Avg: 9412.85, Male SME: 15.89, Female Avg:8744.2, Female SME: 15.34
μ (Males) : 9412.85
Lower Limit(M): 8428.45
Upper Limit(M): 10397.25
μ (Females): 8744.20
Lower Limit(F): 7793.86
Upper Limit(F): 9694.53
Density
Purchase


Sample Size: 1000, Male Avg: 9438.78, Male SME: 5.12, Female Avg:8736.83, Female SME: 4.84
μ (Males) : 9438.78
Lower Limit(M): 9121.35
Upper Limit(M): 9756.22
μ (Females): 8736.83
Lower Limit(F): 8437.11
Upper Limit(F): 9036.56
Density
Purchase





[57]:
```
res
```

[57]:

	Gender	Sample Size	Lower Limit	Upper Limit	Sample Mean	\
0	M	10	6686.56	12086.87	9386.72	
1	F	10	6352.39	11207.94	8780.16	
2	M	100	8583.72	10230.32	9407.02	
3	F	100	7949.86	9530.03	8739.94	
4	M	1000	9164.69	9696.97	9430.83	

5	F	1000	8494.29	8981.30	8737.79
6	M	10000	9359.61	9521.10	9440.35
7	F	10000	8657.69	8808.34	8733.01
8	M	100000	9413.66	9460.72	9437.19
9	F	100000	8721.37	8747.74	8734.56
10	M	10	6206.67	12678.74	9442.71
11	F	10	5896.19	11672.89	8784.54
12	M	100	8428.45	10397.25	9412.85
13	F	100	7793.86	9694.53	8744.20
14	M	1000	9121.35	9756.22	9438.78
15	F	1000	8437.11	9036.56	8736.83
16	M	10000	9341.47	9537.23	9439.35
17	F	10000	8644.88	8824.17	8734.53
18	M	100000	9409.27	9464.98	9437.13
19	F	100000	8718.60	8749.96	8734.28

	Confidence Interval	Interval Range	Range
0	90	[6686.56, 12086.87]	5400.31
1	90	[6352.39, 11207.94]	4855.55
2	90	[8583.72, 10230.32]	1646.60
3	90	[7949.86, 9530.03]	1580.17
4	90	[9164.69, 9696.97]	532.28
5	90	[8494.29, 8981.3]	487.01
6	90	[9359.61, 9521.1]	161.49
7	90	[8657.69, 8808.34]	150.65
8	90	[9413.66, 9460.72]	47.06
9	90	[8721.37, 8747.74]	26.37
10	95	[6206.67, 12678.74]	6472.07
11	95	[5896.19, 11672.89]	5776.70
12	95	[8428.45, 10397.25]	1968.80
13	95	[7793.86, 9694.53]	1900.67
14	95	[9121.35, 9756.22]	634.87
15	95	[8437.11, 9036.56]	599.45
16	95	[9341.47, 9537.23]	195.76
17	95	[8644.88, 8824.17]	179.29
18	95	[9409.27, 9464.98]	55.71
19	95	[8718.6, 8749.96]	31.36

We can observe that

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

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

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

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

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

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

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

We can also observe the same with 95% Confidence.

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

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

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

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

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

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

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

2.2 Married Vs Unmarried Purchase Values

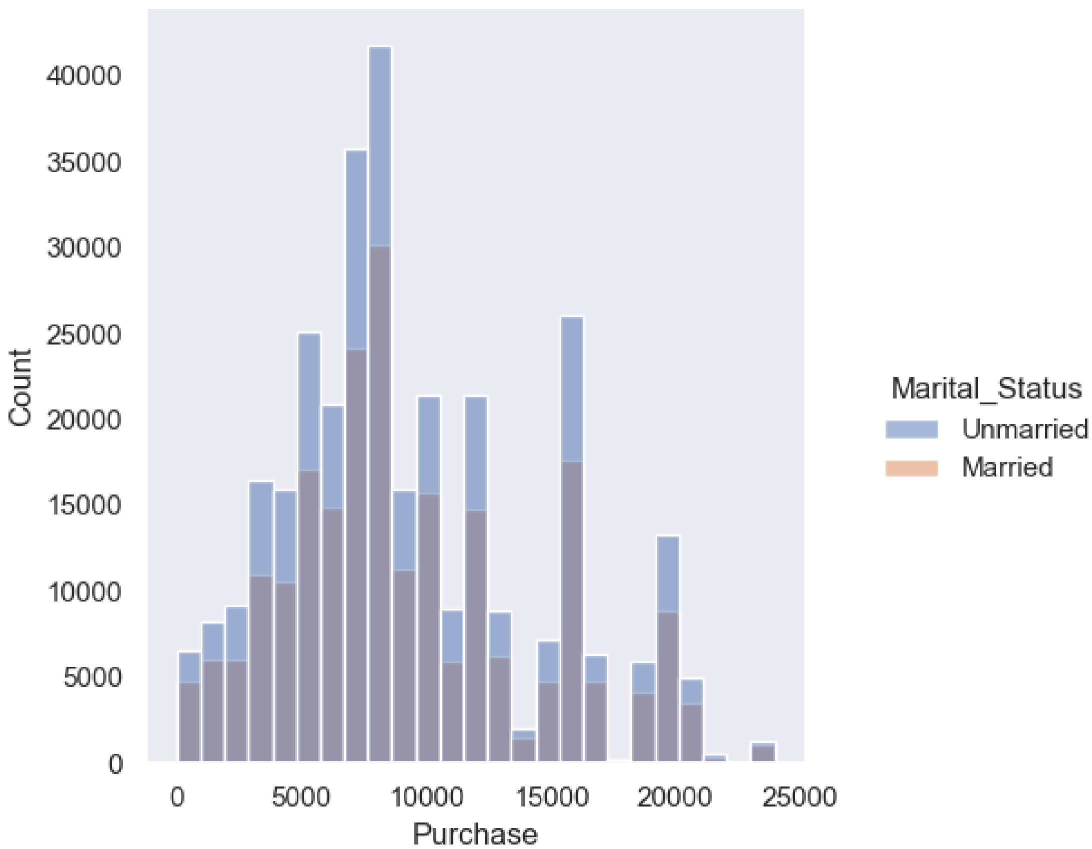
[58]:
```python
def bootstrap_m_vs_um(sample1,sample2,sample_size,itr_size=1000,ci=90):
    ci = ci/100

    plt.figure(figsize=(16,8))
    sample1_n = [np.mean(sample1.sample(sample_size)) for i in range(itr_size)]
    sample2_n = [np.mean(sample2.sample(sample_size)) for i in range(itr_size)]

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

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

    # For Sample2's means
    mean2 = np.mean(sample2_n)
    sigma2 = np.std(sample2_n)
    sem2 = stats.sem(sample2_n)

    lower_limit_2 = norm.ppf((1-ci)/2) * sigma2 + mean2
    upper_limit_2 = norm.ppf(ci + (1-ci)/2) * sigma2 + mean2

    sns.kdeplot(data = sample1_n, color="#F2D2BD", fill = True, linewidth = 2)
```

```
    label_mean1=("  (Married) :  {:.2f}".format(mean1))
    plt.axvline(mean1, color = '#FF00FF', linestyle = 'solid', linewidth = 2,␣
↪label=label_mean1)
    label_limits1=("Lower Limit(M):  {:.2f}\nUpper Limit(M):   {:.2f}".
↪format(lower_limit_1,upper_limit_1))
    plt.axvline(lower_limit_1, color = '#FF69B4', linestyle = 'dashdot',␣
↪linewidth = 2, label=label_limits1)
    plt.axvline(upper_limit_1, color = '#FF69B4', linestyle = 'dashdot',␣
↪linewidth = 2)

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

    plt.title(f"Sample Size: {sample_size}, Married Avg: {np.round(mean1, 2)},␣
↪Married SME: {np.round(sem1,2)}, Unmarried Avg:{np.round(mean2, 2)},␣
↪Unmarried SME: {np.round(sem2,2)}")
    plt.legend(loc = 'upper right')
    plt.xlabel('Purchase')
    plt.ylabel('Density')

    return round(mean1,2), round(mean2,2), round(lower_limit_1,2),␣
↪round(upper_limit_1,2), round(lower_limit_2,2), round(upper_limit_2,2)
```

[59]:
```
df_married = df[df['Marital_Status'] == 'Married']
df_unmarried = df[df['Marital_Status'] == 'Unmarried']
```

[60]:
```
plt.figure(figsize = (16,8))
sns.displot(data = df, x = 'Purchase', hue = 'Marital_Status',bins = 25)
plt.show()
```

```
<Figure size 1600x800 with 0 Axes>
```



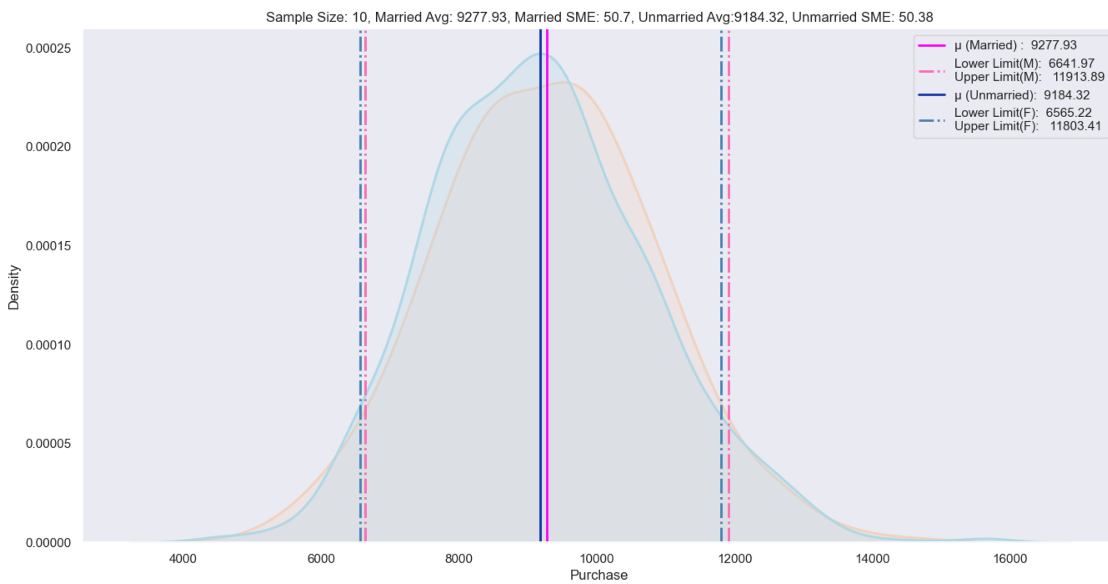
[61]:
```
df.groupby(['Marital_Status'])['Purchase'].describe()
```

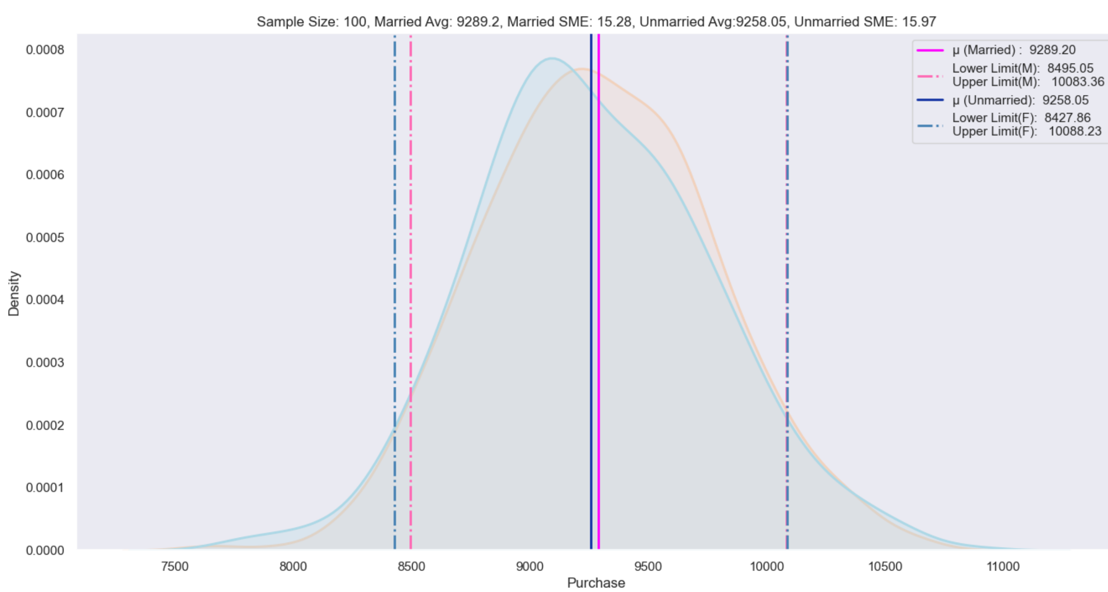
[61]:
```
                   count         mean          std   min     25%     50%  \
Marital_Status
Unmarried       324731.0  9265.907619  5027.347859  12.0  5605.0  8044.0
Married         225337.0  9261.174574  5016.897378  12.0  5843.0  8051.0

                    75%      max
Marital_Status
Unmarried       12061.0  23961.0
Married         12042.0  23961.0
```

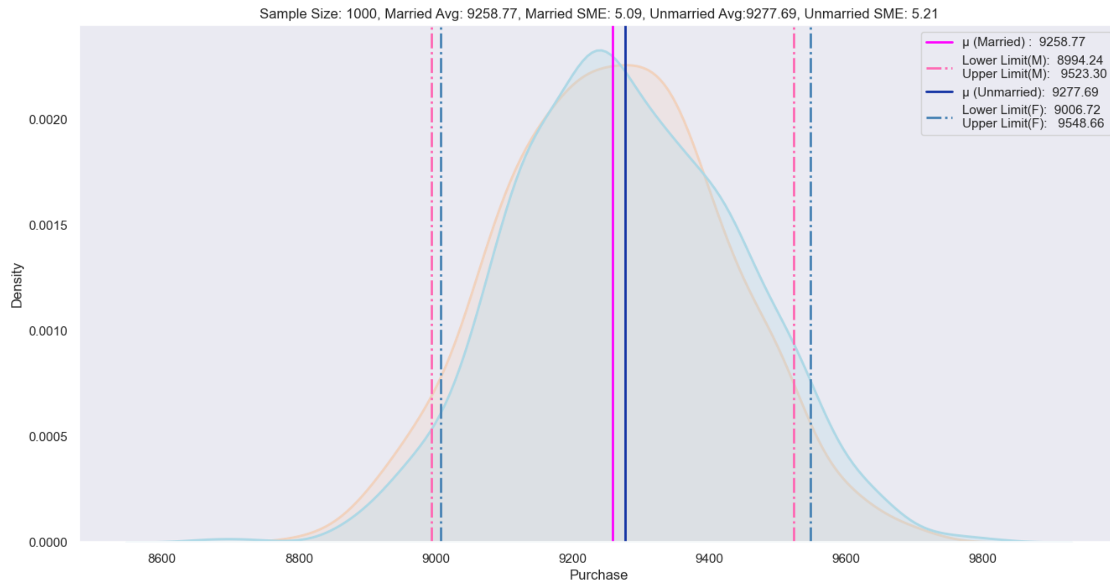
[62]:
```
sample_sizes = [10,100,1000,10000,100000]
ci = 90
itr_size = 1000
```

[63]:
```
res = pd.DataFrame(columns = ['Marital_Status','Sample Size','Lower␣
 ↪Limit','Upper Limit','Sample Mean','Confidence Interval','Interval␣
 ↪Range','Range'])
```

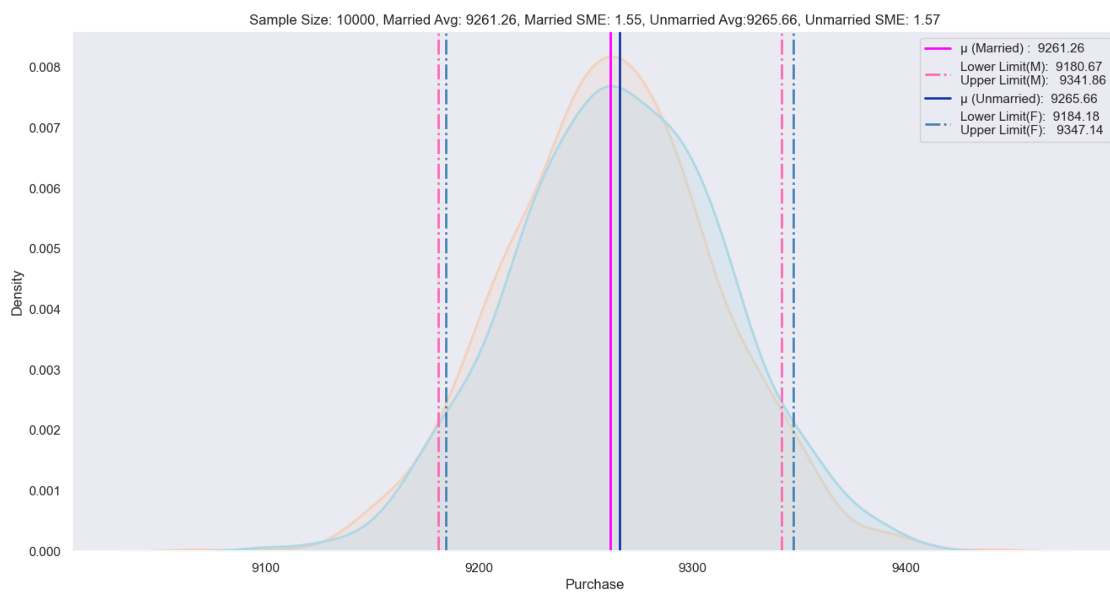
```
for i in sample_sizes:
    m_avg, un_avg, ll_m, ul_m, ll_un, ul_un = bootstrap_m_vs_um(df_married['Purchase'],df_unmarried['Purchase'],i,itr_size,ci)

    res = res.append({'Marital_Status':'Married','Sample Size':i,'Lower Limit':ll_m,'Upper Limit':ul_m,'Sample Mean':m_avg,'Confidence Interval':ci,'Interval Range':[ll_m,ul_m],'Range': ul_m-ll_m}, ignore_index = True)
    res = res.append({'Marital_Status':'Unmarried','Sample Size':i,'Lower Limit':ll_un,'Upper Limit':ul_un,'Sample Mean':un_avg,'Confidence Interval':ci,'Interval Range':[ll_un,ul_un],'Range': ul_un-ll_un}, ignore_index = True)
```



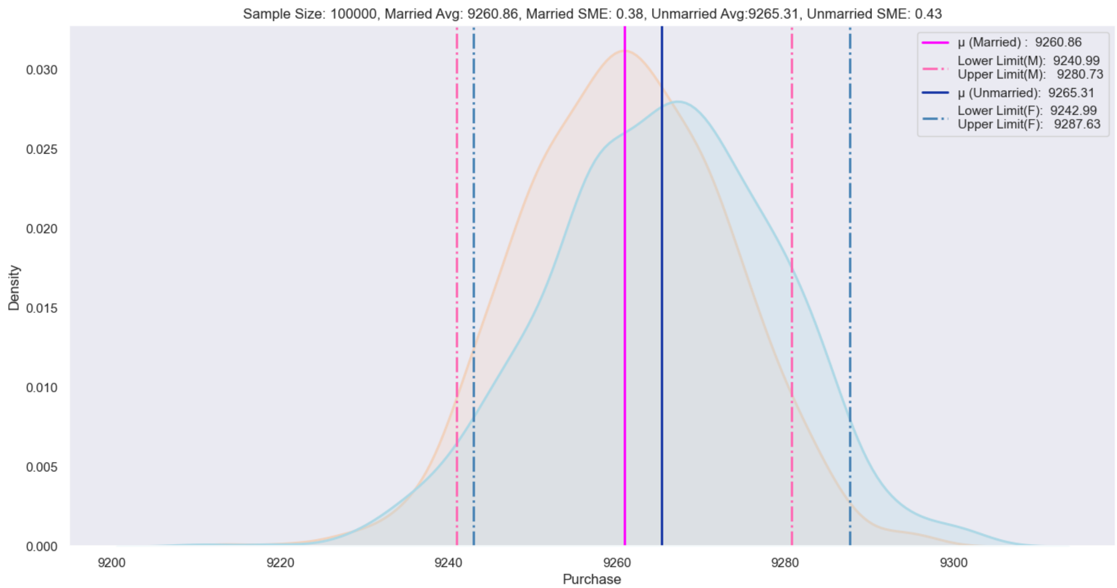




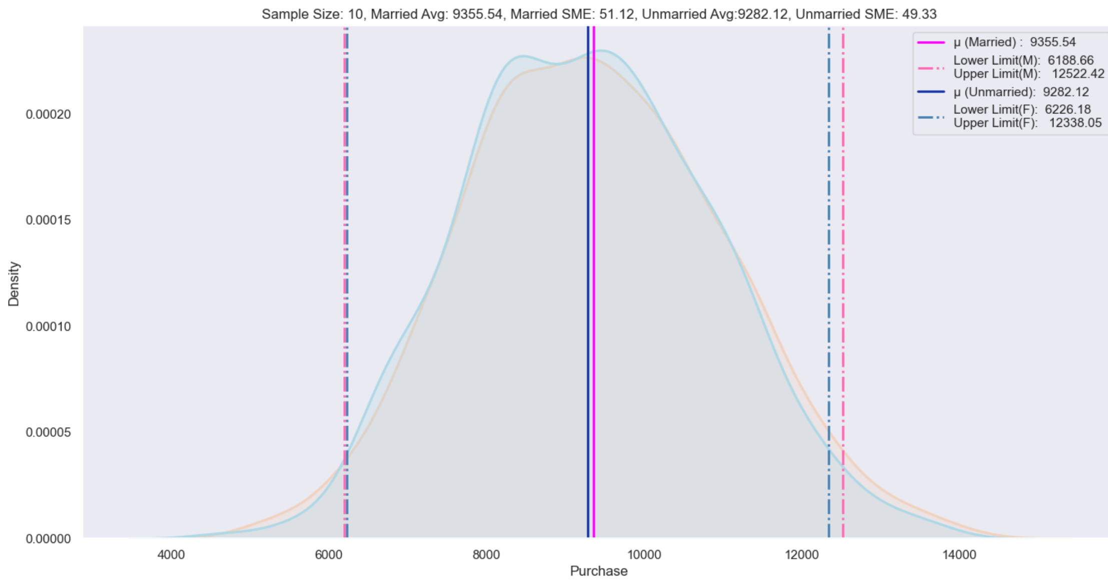
Sample Size: 1000, Married Avg: 9258.77, Married SME: 5.09, Unmarried Avg:9277.69, Unmarried SME: 5.21
μ (Married) : 9258.77
Lower Limit(M): 8994.24
Upper Limit(M): 9523.30
μ (Unmarried): 9277.69
Lower Limit(F): 9006.72
Upper Limit(F): 9548.66
Density
Purchase
0.0000
0.0005
0.0010
0.0015
0.0020
8600
8800
9000
9200
9400
9600
9800

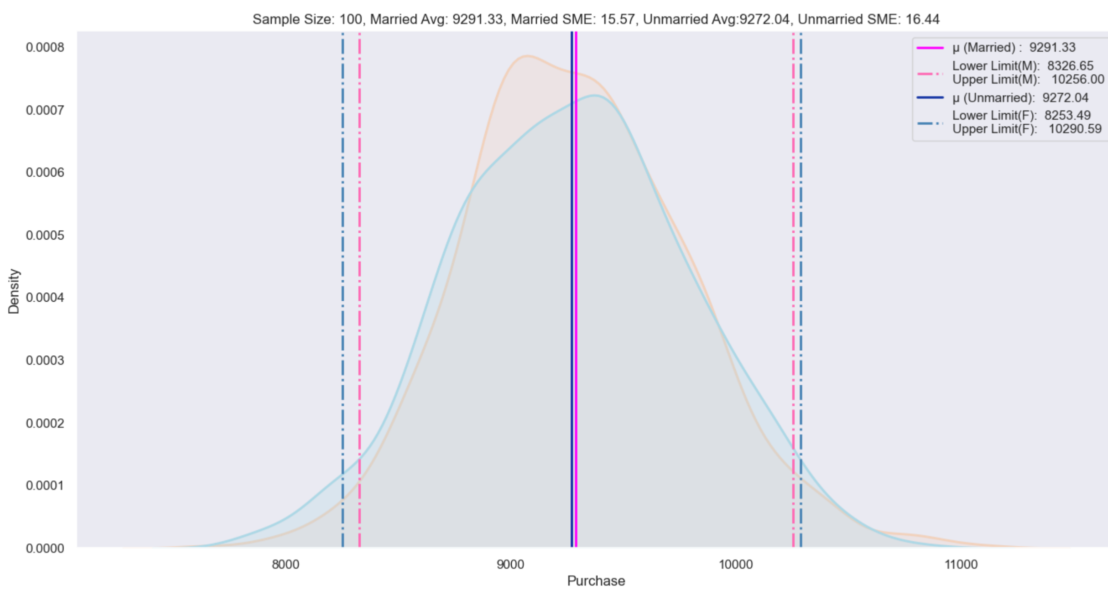


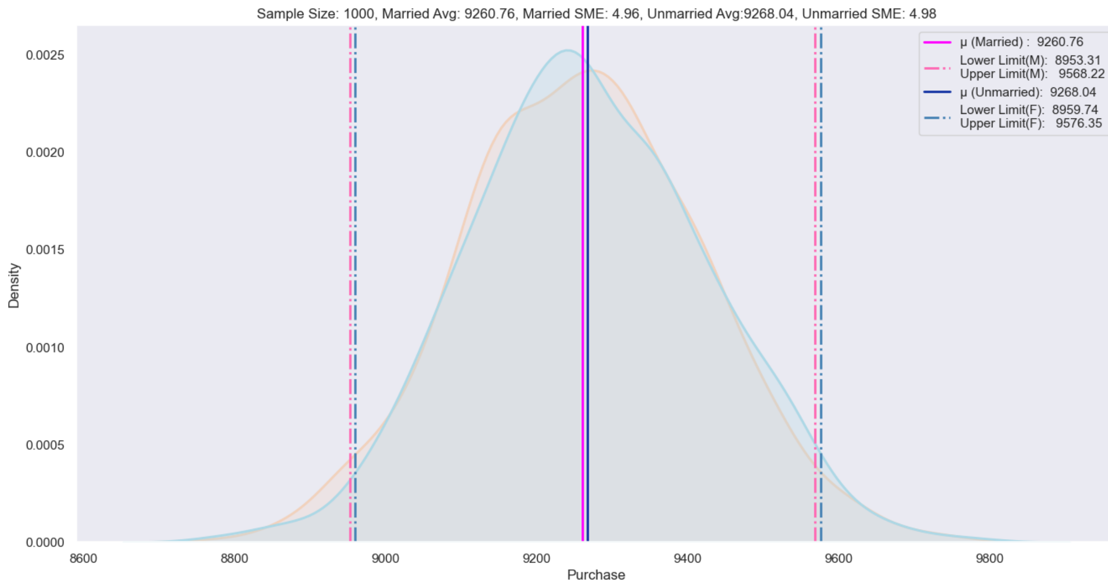
Sample Size: 10000, Married Avg: 9261.26, Married SME: 1.55, Unmarried Avg:9265.66, Unmarried SME: 1.57
μ (Married) : 9261.26
Lower Limit(M): 9180.67
Upper Limit(M): 9341.86
μ (Unmarried): 9265.66
Lower Limit(F): 9184.18
Upper Limit(F): 9347.14
Density
Purchase
0.000
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
9100
9200
9300
9400

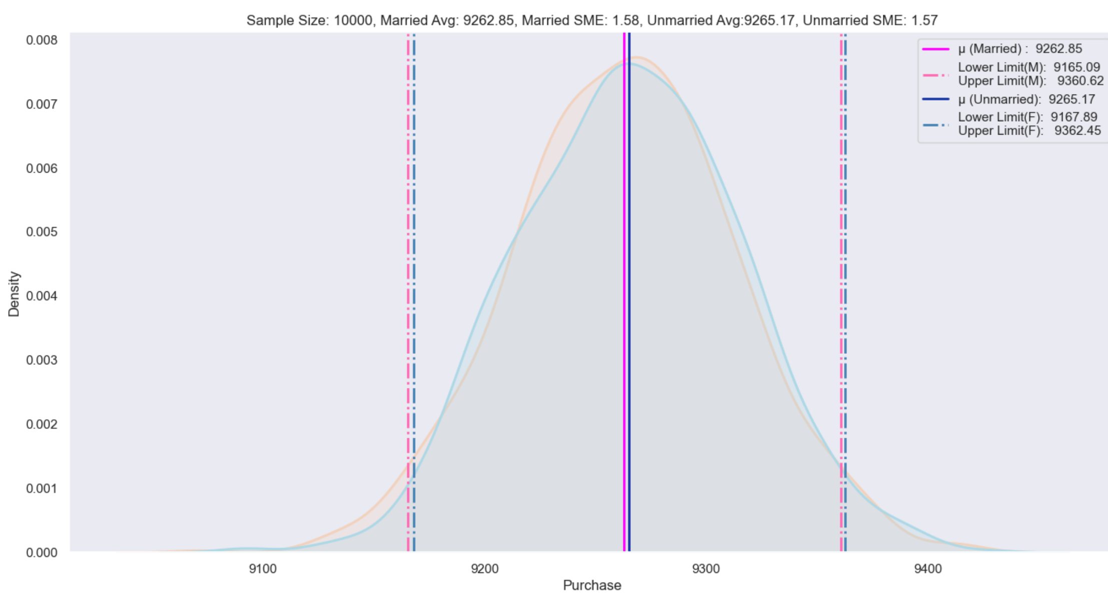


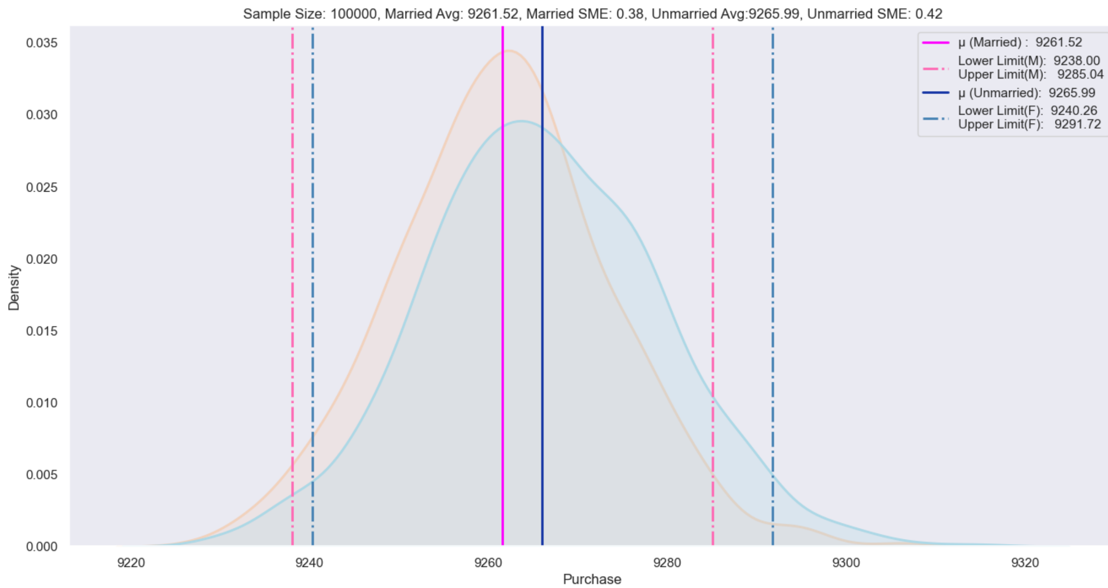
[65]:
```python
sample_sizes = [10,100,1000,10000,100000]
ci = 95
itr_size = 1000

for i in sample_sizes:
    m_avg, un_avg, ll_m, ul_m, ll_un, ul_un = bootstrap_m_vs_um(df_married['Purchase'],df_unmarried['Purchase'],i,itr_size,ci)

    res = res.append({'Marital_Status':'Married','Sample Size':i,'Lower Limit':ll_m,'Upper Limit':ul_m,'Sample Mean':m_avg,'Confidence Interval':ci,'Interval Range':[ll_m,ul_m],'Range': ul_m-ll_m}, ignore_index = True)
    res = res.append({'Marital_Status':'Unmarried','Sample Size':i,'Lower Limit':ll_un,'Upper Limit':ul_un,'Sample Mean':un_avg,'Confidence Interval':ci,'Interval Range':[ll_un,ul_un],'Range': ul_un-ll_un}, ignore_index = True)
```


Sample Size: 10, Married Avg: 9355.54, Married SME: 51.12, Unmarried Avg:9282.12, Unmarried SME: 49.33
μ (Married) : 9355.54
Lower Limit(M): 6188.66
Upper Limit(M): 12522.42
μ (Unmarried): 9282.12
Lower Limit(F): 6226.18
Upper Limit(F): 12338.05
Density
Purchase
0.00000
0.00005
0.00010
0.00015
0.00020
4000
6000
8000
10000
12000
14000


Sample Size: 100, Married Avg: 9291.33, Married SME: 15.57, Unmarried Avg:9272.04, Unmarried SME: 16.44
μ (Married) : 9291.33
Lower Limit(M): 8326.65
Upper Limit(M): 10256.00
μ (Unmarried): 9272.04
Lower Limit(F): 8253.49
Upper Limit(F): 10290.59
Density
Purchase
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0.0008
8000
9000
10000
11000


Sample Size: 1000, Married Avg: 9260.76, Married SME: 4.96, Unmarried Avg:9268.04, Unmarried SME: 4.98
μ (Married) : 9260.76
Lower Limit(M): 8953.31
Upper Limit(M): 9568.22
μ (Unmarried): 9268.04
Lower Limit(F): 8959.74
Upper Limit(F): 9576.35
Density
Purchase


Sample Size: 10000, Married Avg: 9262.85, Married SME: 1.58, Unmarried Avg:9265.17, Unmarried SME: 1.57
μ (Married) : 9262.85
Lower Limit(M): 9165.09
Upper Limit(M): 9360.62
μ (Unmarried): 9265.17
Lower Limit(F): 9167.89
Upper Limit(F): 9362.45
Density
Purchase



We can observe that

There is overlapping even if we increase the sample size.

There is no effect of their marital status on their purchases.

[66]:
```
res
```

[66]:

	Marital_Status	Sample Size	Lower Limit	Upper Limit	Sample Mean	\
0	Married	10	6641.97	11913.89	9277.93	
1	Unmarried	10	6565.22	11803.41	9184.32	
2	Married	100	8495.05	10083.36	9289.20	
3	Unmarried	100	8427.86	10088.23	9258.05	
4	Married	1000	8994.24	9523.30	9258.77	
5	Unmarried	1000	9006.72	9548.66	9277.69	
6	Married	10000	9180.67	9341.86	9261.26	
7	Unmarried	10000	9184.18	9347.14	9265.66	
8	Married	100000	9240.99	9280.73	9260.86	
9	Unmarried	100000	9242.99	9287.63	9265.31	
10	Married	10	6183.16	12386.30	9284.73	
11	Unmarried	10	6182.73	12398.80	9290.77	
12	Married	10	6188.66	12522.42	9355.54	
13	Unmarried	10	6226.18	12338.05	9282.12	
14	Married	100	8326.65	10256.00	9291.33	
15	Unmarried	100	8253.49	10290.59	9272.04	
16	Married	1000	8953.31	9568.22	9260.76	
17	Unmarried	1000	8959.74	9576.35	9268.04	
18	Married	10000	9165.09	9360.62	9262.85	

```
19      Unmarried       10000       9167.89       9362.45       9265.17
20        Married      100000       9238.00       9285.04       9261.52
21      Unmarried      100000       9240.26       9291.72       9265.99
```

```
   Confidence Interval        Interval Range    Range
0                   90   [6641.97, 11913.89]  5271.92
1                   90   [6565.22, 11803.41]  5238.19
2                   90   [8495.05, 10083.36]  1588.31
3                   90   [8427.86, 10088.23]  1660.37
4                   90     [8994.24, 9523.3]   529.06
5                   90    [9006.72, 9548.66]   541.94
6                   90    [9180.67, 9341.86]   161.19
7                   90    [9184.18, 9347.14]   162.96
8                   90    [9240.99, 9280.73]    39.74
9                   90    [9242.99, 9287.63]    44.64
10                  95    [6183.16, 12386.3]  6203.14
11                  95    [6182.73, 12398.8]  6216.07
12                  95   [6188.66, 12522.42]  6333.76
13                  95   [6226.18, 12338.05]  6111.87
14                  95    [8326.65, 10256.0]  1929.35
15                  95   [8253.49, 10290.59]  2037.10
16                  95    [8953.31, 9568.22]   614.91
17                  95    [8959.74, 9576.35]   616.61
18                  95    [9165.09, 9360.62]   195.53
19                  95    [9167.89, 9362.45]   194.56
20                  95     [9238.0, 9285.04]    47.04
21                  95    [9240.26, 9291.72]    51.46
```

For married and unmarried customers, sample size 10, confidence interval 90 we can observe tha

For married and unmarried customers, sample size 100000, confidence interval 90 we can observe

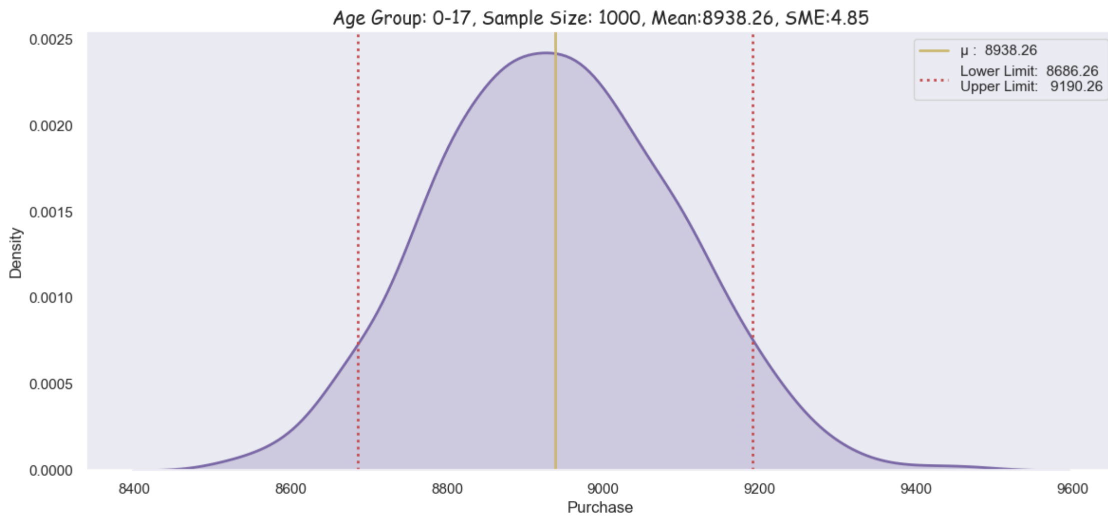
This means there is no effect of marital status on purchase habits of customers

2.3 Age groups wise purchase habits

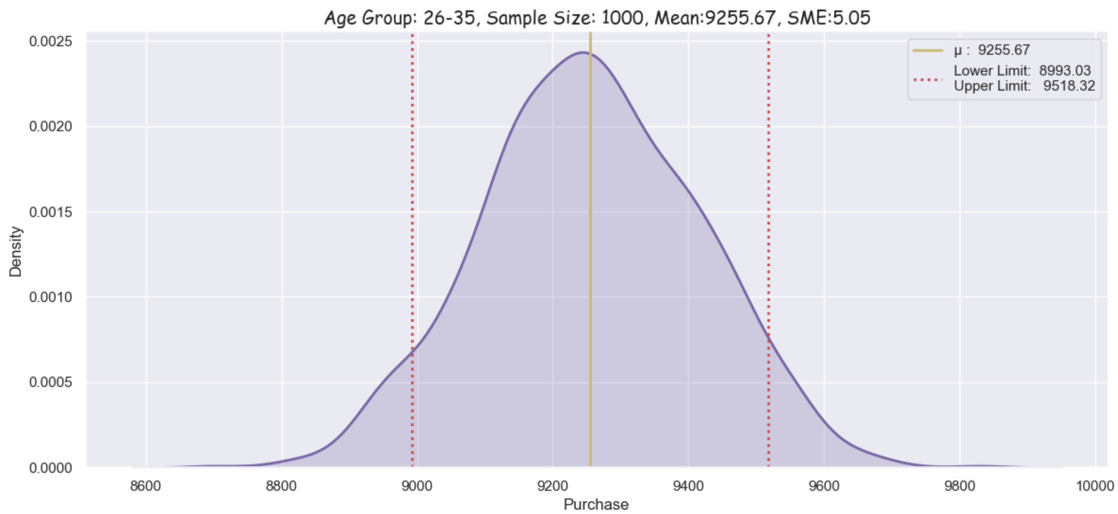
[67]:
```
def bootstrap_age(sample, sample_size, itr_size=1000, ci = 90):
    ci = ci/100

    global flag

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

    mean  = np.mean(sample_n)
    sigma = np.std(sample_n)
    sem   = stats.sem(sample_n)
```

```
    lower_limit = norm.ppf((1-ci)/2) * sigma + mean
    upper_limit = norm.ppf(ci + (1-ci)/2) * sigma + mean

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

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

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

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

    plt.show()
    flag += 1

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

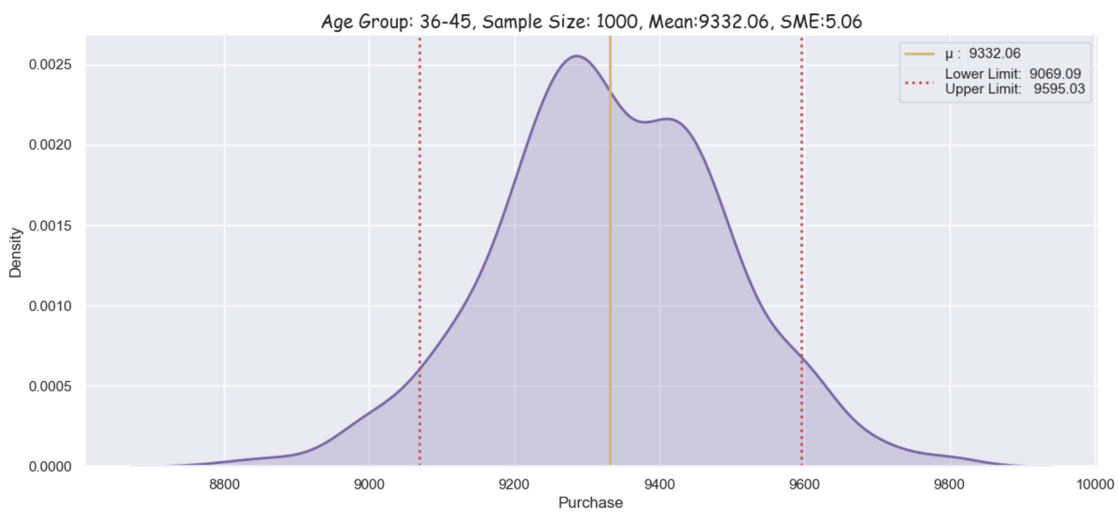
[68]:
```
ci = 90
itr_size = 1000
sample_size = 1000
flag = 0
global age_group
age_group = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

res = pd.DataFrame(columns = ['Age_Group','Sample Size','Lower Limit','Upper Limit','Sample Mean','Confidence Interval','Interval Range','Range'])

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

    res = res.append({'Age_Group':i,'Sample Size':sample_size,'Lower Limit':ll,'Upper Limit':ul,'Sample Mean':mean,'Confidence Interval':ci,'Interval Range':[ll,ul],'Range': ul-ll}, ignore_index = True)
```


Age Group: 0-17, Sample Size: 1000, Mean:8938.26, SME:4.85
μ : 8938.26
Lower Limit: 8686.26
Upper Limit: 9190.26
Density
Purchase
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
8400
8600
8800
9000
9200
9400
9600

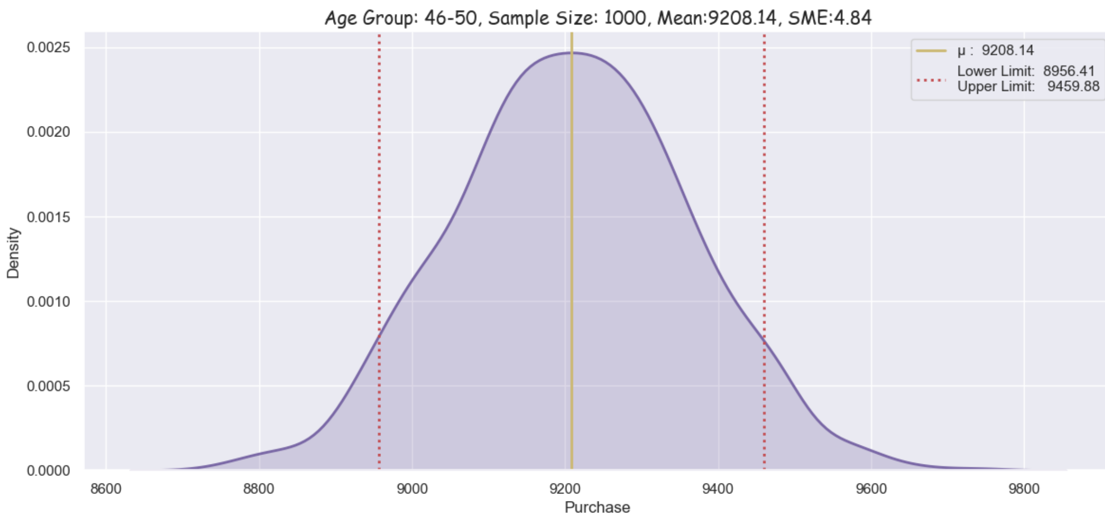

Age Group: 18-25, Sample Size: 1000, Mean:9175.85, SME:5.12
μ : 9175.85
Lower Limit: 8909.53
Upper Limit: 9442.17
Density
Purchase
0.0000
0.0005
0.0010
0.0015
0.0020
8600
8800
9000
9200
9400
9600
9800

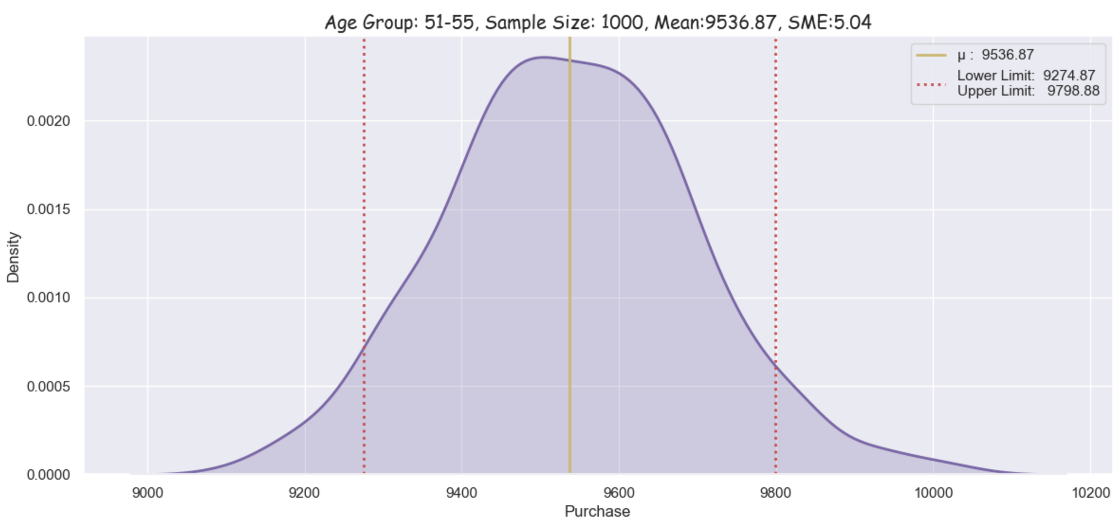


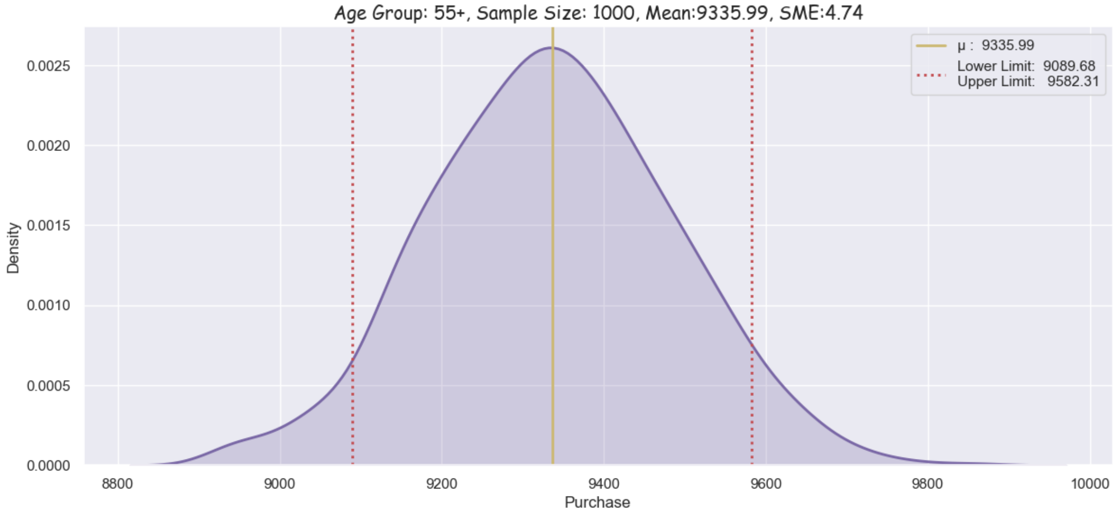
Age Group: 26-35, Sample Size: 1000, Mean:9255.67, SME:5.05
μ : 9255.67
Lower Limit: 8993.03
Upper Limit: 9518.32
Density
Purchase
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
8600
8800
9000
9200
9400
9600
9800
10000



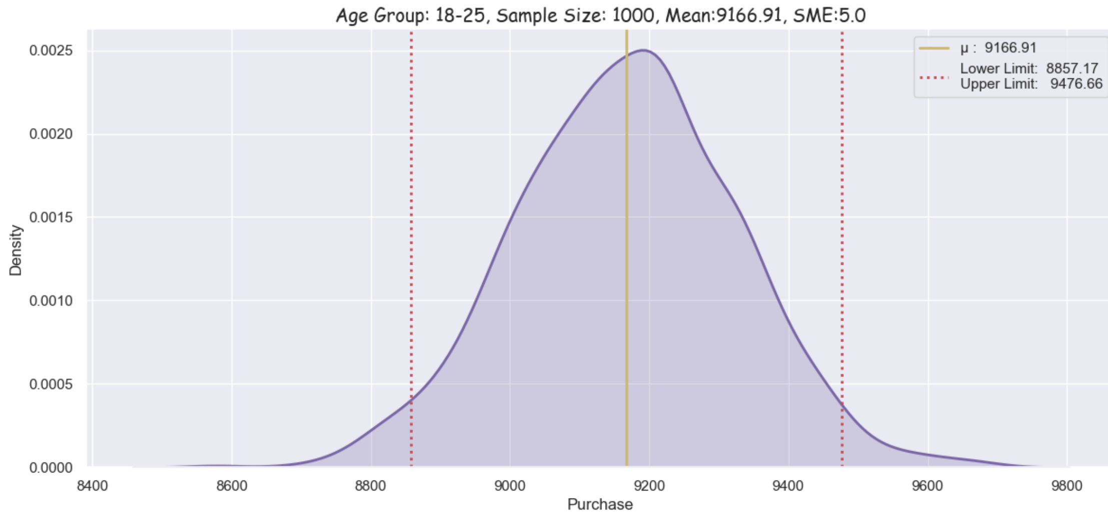
Age Group: 36-45, Sample Size: 1000, Mean:9332.06, SME:5.06
μ : 9332.06
Lower Limit: 9069.09
Upper Limit: 9595.03
Density
Purchase
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
8800
9000
9200
9400
9600
9800
10000


Age Group: 46-50, Sample Size: 1000, Mean:9208.14, SME:4.84
μ : 9208.14
Lower Limit: 8956.41
Upper Limit: 9459.88
Density
Purchase


Age Group: 51-55, Sample Size: 1000, Mean:9536.87, SME:5.04
μ : 9536.87
Lower Limit: 9274.87
Upper Limit: 9798.88
Density
Purchase

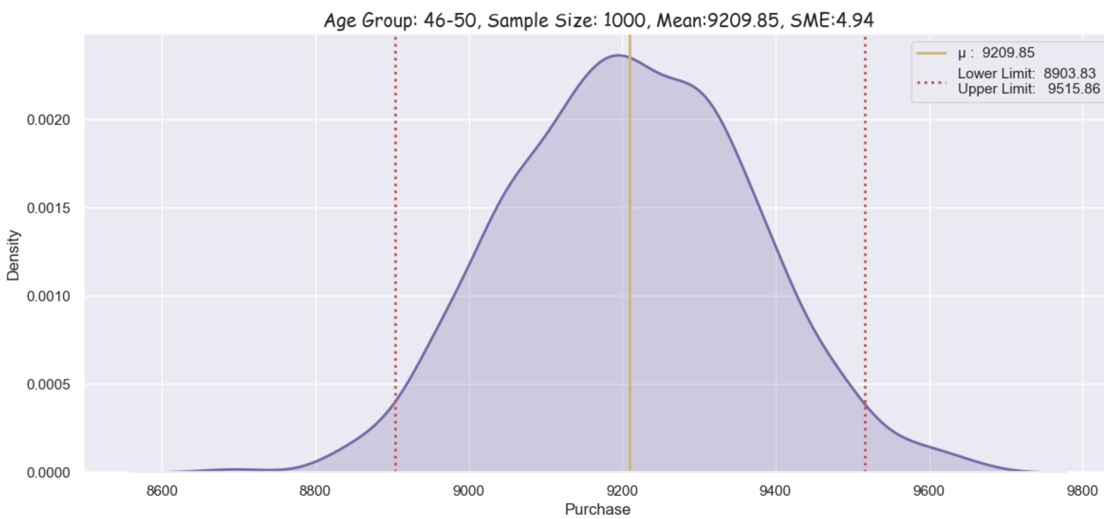


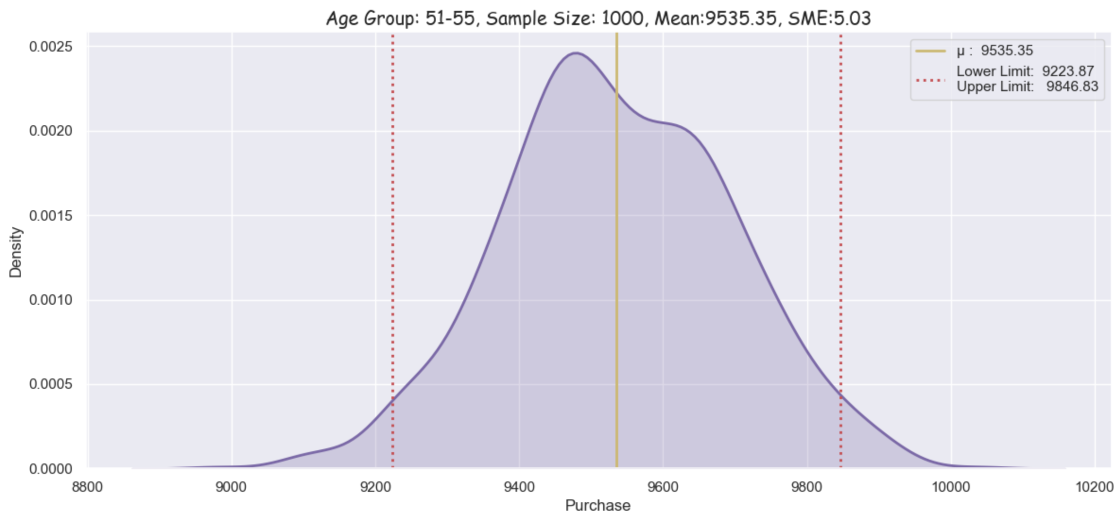

```
[69]: ci = 95
itr_size = 1000
sample_size = 1000
flag = 0

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

    res = res.append({'Age_Group':i,'Sample Size':sample_size,'Lower Limit':
 ↪ll,'Upper Limit':ul,'Sample Mean':mean,'Confidence Interval':ci,'Interval␣
 ↪Range':[ll,ul],'Range': ul-ll}, ignore_index = True)
```


Age Group: 18-25, Sample Size: 1000, Mean:9166.91, SME:5.0
μ : 9166.91
Lower Limit: 8857.17
Upper Limit: 9476.66
Density
Purchase


Age Group: 26-35, Sample Size: 1000, Mean:9254.59, SME:4.89
μ : 9254.59
Lower Limit: 8951.61
Upper Limit: 9557.57
Density
Purchase


Age Group: 36-45, Sample Size: 1000, Mean:9336.02, SME:4.88
μ : 9336.02
Lower Limit: 9033.57
Upper Limit: 9638.46
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
Density
8800
9000
9200
9400
9600
9800
10000
Purchase


Age Group: 46-50, Sample Size: 1000, Mean:9209.85, SME:4.94
μ : 9209.85
Lower Limit: 8903.83
Upper Limit: 9515.86
0.0020
0.0015
0.0010
0.0005
0.0000
Density
8600
8800
9000
9200
9400
9600
9800
Purchase



[70]:
```
res
```

[70]:

	Age_Group	Sample Size	Lower Limit	Upper Limit	Sample Mean \
0	0-17	1000	8686.26	9190.26	8938.26
1	18-25	1000	8909.53	9442.17	9175.85
2	26-35	1000	8993.03	9518.32	9255.67
3	36-45	1000	9069.09	9595.03	9332.06
4	46-50	1000	8956.41	9459.88	9208.14
5	51-55	1000	9274.87	9798.88	9536.87
6	55+	1000	9089.68	9582.31	9335.99
7	0-17	1000	8618.50	9230.77	8924.64
8	18-25	1000	8857.17	9476.66	9166.91

9	26-35	1000	8951.61	9557.57	9254.59
10	36-45	1000	9033.57	9638.46	9336.02
11	46-50	1000	8903.83	9515.86	9209.85
12	51-55	1000	9223.87	9846.83	9535.35
13	55+	1000	9026.26	9645.83	9336.04

	Confidence Interval	Interval Range	Range
0	90	[8686.26, 9190.26]	504.00
1	90	[8909.53, 9442.17]	532.64
2	90	[8993.03, 9518.32]	525.29
3	90	[9069.09, 9595.03]	525.94
4	90	[8956.41, 9459.88]	503.47
5	90	[9274.87, 9798.88]	524.01
6	90	[9089.68, 9582.31]	492.63
7	95	[8618.5, 9230.77]	612.27
8	95	[8857.17, 9476.66]	619.49
9	95	[8951.61, 9557.57]	605.96
10	95	[9033.57, 9638.46]	604.89
11	95	[8903.83, 9515.86]	612.03
12	95	[9223.87, 9846.83]	622.96
13	95	[9026.26, 9645.83]	619.57

We can observe with 90% confidence that

`Age group 0-17 has the least purchase value range of [8719.59, 8750.12].`

`Age group 51-55 has highest purchase value range of [9288.27, 9802.69].`

We can observe with 95% confidence that

`Age group 0-17 has the least purchase value range of [9288.27, 9802.69].`

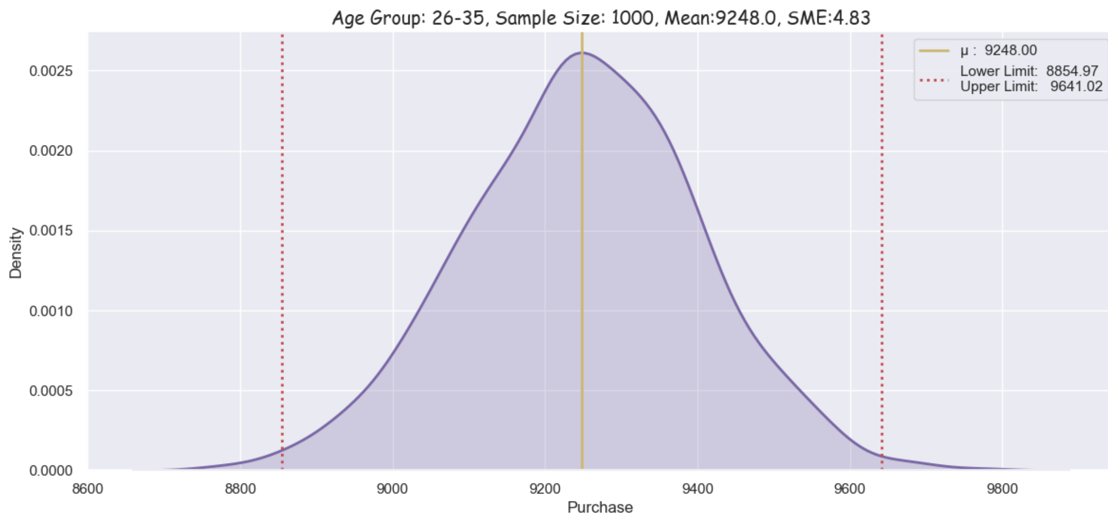
`Age group 51-55 has highest purchase value range of [9218.76, 9861.45].`

All the age groups still have overlap which makes it difficult to interpret the ranges.

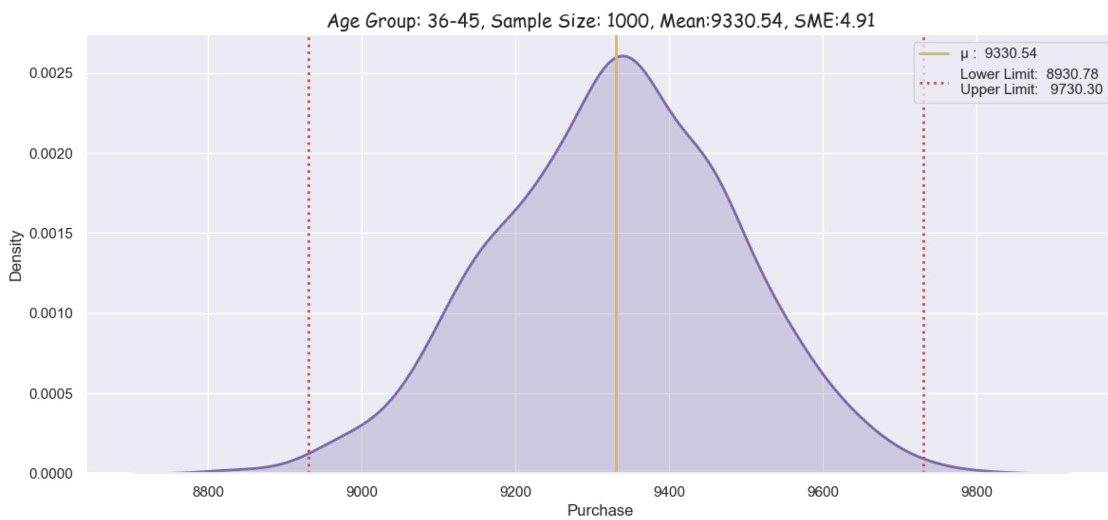
So now, Lets visualise the graphs of 1000 mean values of purchase samples for sample size of 1000 for all the age groups with 99% confidence interval.

[71]:
```
ci = 99
itr_size = 1000
sample_size = 1000
flag = 0

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

```
    res = res.append({'Age_Group':i,'Sample Size':sample_size,'Lower Limit':
↪ll,'Upper Limit':ul,'Sample Mean':mean,'Confidence Interval':ci,'Interval␣
↪Range':[ll,ul],'Range': ul-ll}, ignore_index = True)
```

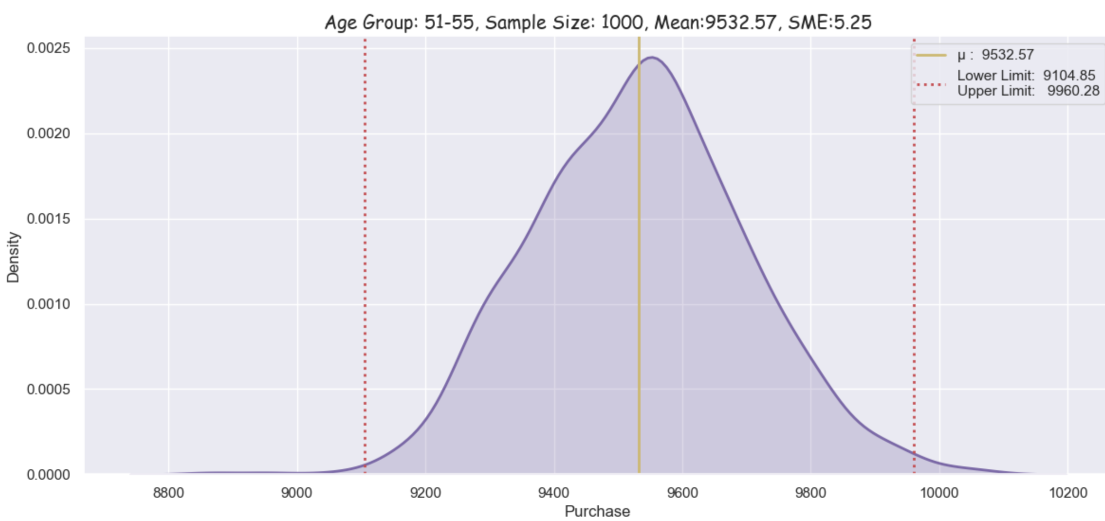


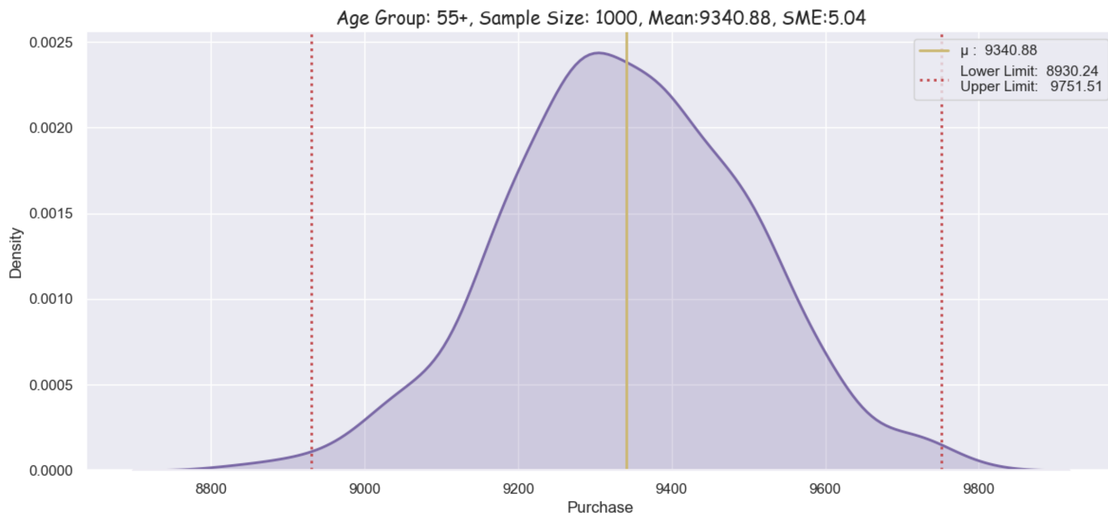
Age Group: 26-35, Sample Size: 1000, Mean:9248.0, SME:4.83
μ : 9248.00
Lower Limit: 8854.97
Upper Limit: 9641.02
Density
Purchase



Age Group: 36-45, Sample Size: 1000, Mean:9330.54, SME:4.91
μ : 9330.54
Lower Limit: 8930.78
Upper Limit: 9730.30
Density
Purchase


Age Group: 46-50, Sample Size: 1000, Mean:9207.73, SME:4.79
μ : 9207.73
Lower Limit: 8818.16
Upper Limit: 9597.30
Density
Purchase


Age Group: 51-55, Sample Size: 1000, Mean:9532.57, SME:5.25
μ : 9532.57
Lower Limit: 9104.85
Upper Limit: 9960.28
Density
Purchase



[72]:
```
res
```

[72]:
```
   Age_Group  Sample Size  Lower Limit  Upper Limit  Sample Mean  \
0       0-17         1000      8686.26      9190.26      8938.26
1      18-25         1000      8909.53      9442.17      9175.85
2      26-35         1000      8993.03      9518.32      9255.67
3      36-45         1000      9069.09      9595.03      9332.06
4      46-50         1000      8956.41      9459.88      9208.14
5      51-55         1000      9274.87      9798.88      9536.87
6        55+         1000      9089.68      9582.31      9335.99
7       0-17         1000      8618.50      9230.77      8924.64
8      18-25         1000      8857.17      9476.66      9166.91
9      26-35         1000      8951.61      9557.57      9254.59
10     36-45         1000      9033.57      9638.46      9336.02
11     46-50         1000      8903.83      9515.86      9209.85
12     51-55         1000      9223.87      9846.83      9535.35
13       55+         1000      9026.26      9645.83      9336.04
14      0-17         1000      8518.98      9337.42      8928.20
15     18-25         1000      8767.66      9582.31      9174.98
16     26-35         1000      8854.97      9641.02      9248.00
17     36-45         1000      8930.78      9730.30      9330.54
18     46-50         1000      8818.16      9597.30      9207.73
19     51-55         1000      9104.85      9960.28      9532.57
20       55+         1000      8930.24      9751.51      9340.88

   Confidence Interval        Interval Range   Range
0                   90  [8686.26, 9190.26]  504.00
1                   90  [8909.53, 9442.17]  532.64
2                   90  [8993.03, 9518.32]  525.29
```

```
3                        90  [9069.09, 9595.03]  525.94
4                        90  [8956.41, 9459.88]  503.47
5                        90  [9274.87, 9798.88]  524.01
6                        90  [9089.68, 9582.31]  492.63
7                        95   [8618.5, 9230.77]  612.27
8                        95  [8857.17, 9476.66]  619.49
9                        95  [8951.61, 9557.57]  605.96
10                       95  [9033.57, 9638.46]  604.89
11                       95  [8903.83, 9515.86]  612.03
12                       95  [9223.87, 9846.83]  622.96
13                       95  [9026.26, 9645.83]  619.57
14                       99  [8518.98, 9337.42]  818.44
15                       99  [8767.66, 9582.31]  814.65
16                       99  [8854.97, 9641.02]  786.05
17                       99   [8930.78, 9730.3]  799.52
18                       99   [8818.16, 9597.3]  779.14
19                       99  [9104.85, 9960.28]  855.43
20                       99  [8930.24, 9751.51]  821.27
```

We can observe with 99% confidence that

```
Age group 0-17 has the least purchase value range of [8543.18, 9341.05].

Age group 51-55 has highest purchase value range of [9134.55, 9943.98].

We can say that age group does not have much effect on the spending of customers as their inte
```

3 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

3.0.1 Confidence Interval by Gender

Now using the Central Limit Theorem for the population:

1. Average amount spend by male customers is 9,26,341.86
2. Average amount spend by female customers is 7,11,704.09

Now we can infer about the population that, 95% of the times:

1. Average amount spend by male customer will lie in between: (895617.83, 955070.97)

2. Average amount spend by female customer will lie in between: (673254.77, 750794.02)

3.0.2 Confidence Interval by Marital Status

1. Married confidence interval of means: (806668.83, 880384.76)
2. Unmarried confidence interval of means: (848741.18, 912410.38)

3.0.3 Confidence Interval by Age

1. For age 26-35 –> confidence interval of means: (945034.42, 1034284.21)
2. For age 36-45 –> confidence interval of means: (823347.80, 935983.62)
3. For age 18-25 –> confidence interval of means: (801632.78, 908093.46)
4. For age 46-50 –> confidence interval of means: (713505.63, 871591.93)
5. For age 51-55 –> confidence interval of means: (692392.43, 834009.42)
6. For age 55+ –> confidence interval of means: (476948.26, 602446.23)
7. For age 0-17 –> confidence interval of means: (527662.46, 710073.17)

4 Recommendations

1. Men spent more money than women, So company should focus on retaining the male customers and getting more male customers.
2. 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.
3. Unmarried customers spend more money than married customers, So company should focus on acquisition of Unmarried customers.
4. 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
5. 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.

[ ]:

```
```