Walmart - Business CaseStudy [CLT and Confidence Interval]

Analysed by : KASI

WALMART :

`Walmart` , founded in `1962 by Sam Walton` , is a retail giant and one of the world's largest and most influential companies. Headquartered in `Bentonville, Arkansas` , this American multinational corporation has established itself as a global powerhouse in the retail industry. Walmart operates a vast network of hypermarkets, discount department stores, and grocery stores under various brand names across the United States and in numerous countries around the world.

Known for its `"Everyday Low Prices"` strategy, Walmart has redefined the retail landscape with its commitment to offering a wide range of products at affordable prices. With its extensive supply chain and efficient distribution systems, the company has played a pivotal role in shaping consumer expectations and shopping habits. Beyond retail, Walmart has also ventured into e-commerce, technology innovation, and sustainability initiatives, further solidifying its position as a key player in the modern retail ecosystem.

- Walmart: Where Shopping Becomes a Global Phenomenon

`Walmart` , the retail titan, stretches its tentacles across 19 countries, boasting over 10,500 stores and serving more than 100 million customers worldwide. It's not just a shopping haven; it's a data goldmine waiting to be unearthed.

- A Retail Colossus with a Human Touch

Despite its vast size, Walmart remains dedicated to its core values of customer service and community involvement. The company's philanthropic efforts focus on areas like hunger relief and children's health, and its commitment to employee development has earned it recognition as a top employer.

Walmart's story is far from over. As the retail landscape continues to evolve, this retail giant is sure to adapt and innovate, remaining a dominant force in the world of shopping.

Business Problem:

Objective

- 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).

About Data

The company collected the transactional data of customers who purchased products from the Walmart Stores during Black Friday.

Features of the dataset:

The dataset has the following features:

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

In [284…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from scipy.stats import norm,boxcox
```

```python
from statsmodels.stats.weightstats import ztest
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [2]:
```python
# walmart colour palette
cp = ['#003087','#007dc6','#79b9e7','#ffb81c','#ffc120','#f47421','#6cace4','#76c143','#222222','#444444']
cp1 = ['#0071ce','#ffc220','#003087','#ffb81c','#6cace4']
cp2 = ['#0071CE','#FCB61A']
cp3 = ['#0071CE','#ffc220','#003087','#ffc120','#ffb81c','#f47421','#6cace4','#76c143','#222222','#444444','#79b9e7','#ffb81c']
```

In [3]:
```python
walmart = pd.read_csv('walmart_data.csv')
```

In [4]:
```python
wm = walmart.copy()
```

Exploration of data :

In [4]:
```python
wm.head()
```

Out[4]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	7969

In [5]:
```python
wm.tail(3)
```

Out[5]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
550065	1006036	P00375436	F	26-35	15	B	4+	1	20	137
550066	1006038	P00375436	F	55+	1	C	2	0	20	365
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	490

In [6]:
```python
wm.shape
```

Out[6]:
```
(550068, 10)
```

🕎 Changing the Datatype of Columns

In [7]:
```python
wm.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 [5]:
```python
for _ in wm.columns[:-1]:
    wm[_] = wm[_].astype('category')
wm.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype   
---  ------                      --------------   -----   
 0   User_ID                     550068 non-null  category
 1   Product_ID                  550068 non-null  category
 2   Gender                      550068 non-null  category
 3   Age                         550068 non-null  category
 4   Occupation                  550068 non-null  category
 5   City_Category               550068 non-null  category
 6   Stay_In_Current_City_Years  550068 non-null  category
 7   Marital_Status              550068 non-null  category
 8   Product_Category            550068 non-null  category
 9   Purchase                    550068 non-null  int64   
dtypes: category(9), int64(1)
memory usage: 10.3 MB
```

Insights

- Except Purchase Column, all the other data types are of categorical type. We have changed the datatypes of all such columns to category.

Statistical Summary

In [9]:
```
wm.describe()
```

Out[9]:

	Purchase
count	550068.000000
mean	9263.968713
std	5023.065394
min	12.000000
25%	5823.000000
50%	8047.000000
75%	12054.000000
max	23961.000000

In [10]:
```
wm.describe(include = 'category').T
```

Out[10]:

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

Insights

- `The purchase amounts` vary widely, with the minimum recorded purchase being `$12` and the maximum reaching `$23961`. The `median` purchase amount of `$8047` is notably lower than the `mean` purchase amount of `$9264`, indicating a **Right-Skewed Distribution** where a few high-value purchases pull up the mean value.

```
 - **User_ID** - Among *5,50,068* transactions there are `5891` unique user_id, indicating same customers buying multiple products.

 - **Product_ID** - There are `3631` unique products,with the product having the code `P00265242` being the `highest seller`, with a maximum of `1,880 units` sold.

- **Gender** - Out of *5,50,068* transactions, *4,14,259* `(nearly 75%)` were done by **Male** gender indicating a significant disparity in purchase behavior between males and females during the Black Friday event.

- **Age** - We have `7` unique age groups in the dataset. `26 - 35` Age group has maximum of *2,19,587* transactions. We will analyse this feature in detail in future

- **Stay_In_Current_City_Years** - Customers with `1` year of stay in current city accounted to maximum of `1,93,821` transactions among all the other customers with (0,2,3,4+) years of stay in current city
```

- **Marital_Status** - `59%` of the total transactions were done by `Unmarried Customers` and `41%` by `Married Customers`.

📜 📜 Duplicate Detection

In [11]:
```
wm.duplicated().sum()
```

Out[11]:
```
0
```

🏷️ Insights

- The dataset does not contain any duplicates.

🔸 ❓ Null Detection

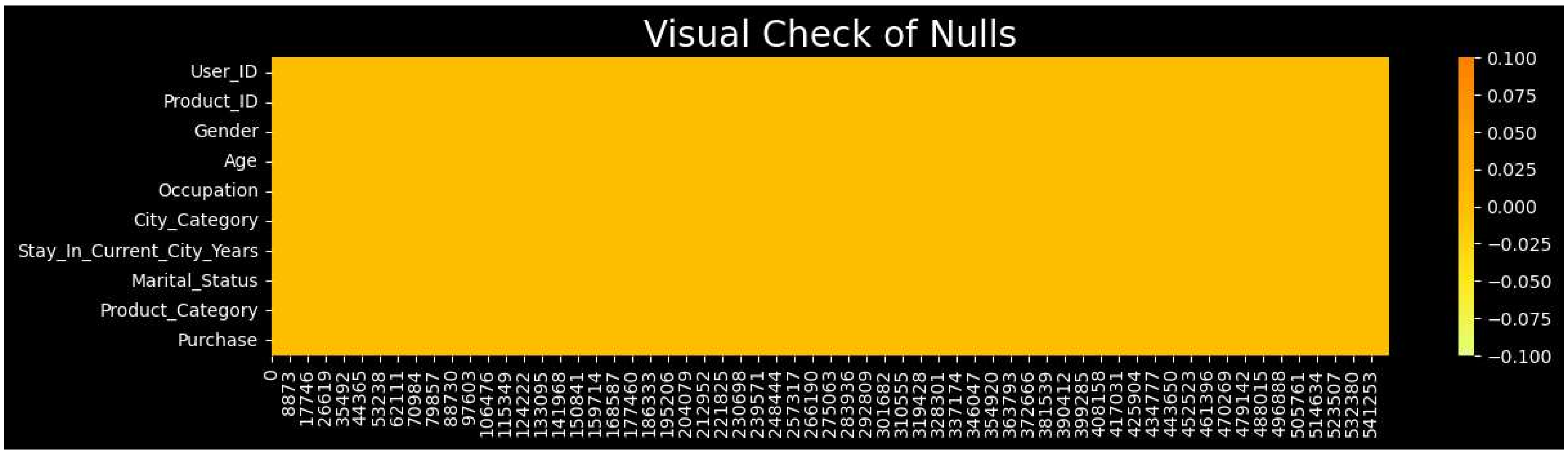
In [12]:
```
wm.isna().any()
```

Out[12]:
```
User_ID                       False
Product_ID                    False
Gender                        False
Age                           False
Occupation                    False
City_Category                 False
Stay_In_Current_City_Years    False
Marital_Status                False
Product_Category              False
Purchase                      False
dtype: bool
```

In [13]:
```
wm.isnull().sum()
```

Out[13]:
```
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 [32]:
```
plt.figure(figsize=(14,3))
plt.style.use('dark_background')
sns.heatmap(wm.isnull().T,cmap='Wistia')
plt.title('Visual Check of Nulls',fontsize=20)
plt.show()
```



🏷️ Insights

- The dataset does not contain any missing values.

🕵️ Exploratory Data Analysis

🔰 Non-Graphical Analysis

In [15]:
```
#checking the unique values for columns
for col in wm.columns:
    print()
    print('Total Unique Values in',col,'column are :-',wm[col].nunique())
    print('Unique Values in',col,'column are :-\n',wm[col].unique())
    print()
    print('-'*140)
```

```
Total Unique Values in User_ID column are :- 5891
Unique Values in User_ID column are :-
 [1000001, 1000002, 1000003, 1000004, 1000005, ..., 1004588, 1004871, 1004113, 1005391, 1001529]
Length: 5891
Categories (5891, int64): [1000001, 1000002, 1000003, 1000004, ..., 1006037, 1006038, 1006039, 1006040]

--------------------------------------------------------------------------------------------------------------------------------
------------

Total Unique Values in Product_ID column are :- 3631
Unique Values in Product_ID column are :-
 ['P00069042', 'P00248942', 'P00087842', 'P00085442', 'P00285442', ..., 'P00375436', 'P00372445', 'P00370293', 'P00371644', 'P00
370853']
Length: 3631
Categories (3631, object): ['P00000142', 'P00000242', 'P00000342', 'P00000442', ..., 'P0099642', 'P0099742', 'P0099842', 'P00999
42']

--------------------------------------------------------------------------------------------------------------------------------
------------

Total Unique Values in Gender column are :- 2
Unique Values in Gender column are :-
 ['F', 'M']
Categories (2, object): ['F', 'M']

--------------------------------------------------------------------------------------------------------------------------------
------------

Total Unique Values in Age column are :- 7
Unique Values in Age column are :-
 ['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+']

--------------------------------------------------------------------------------------------------------------------------------
------------

Total Unique Values in Occupation column are :- 21
Unique Values in Occupation column are :-
 [10, 16, 15, 7, 20, ..., 18, 5, 14, 13, 6]
Length: 21
Categories (21, int64): [0, 1, 2, 3, ..., 17, 18, 19, 20]

--------------------------------------------------------------------------------------------------------------------------------
------------

Total Unique Values in City_Category column are :- 3
Unique Values in City_Category column are :-
 ['A', 'C', 'B']
Categories (3, object): ['A', 'B', 'C']

--------------------------------------------------------------------------------------------------------------------------------
------------

Total Unique Values in Stay_In_Current_City_Years column are :- 5
Unique Values in Stay_In_Current_City_Years column are :-
 ['2', '4+', '3', '1', '0']
Categories (5, object): ['0', '1', '2', '3', '4+']

--------------------------------------------------------------------------------------------------------------------------------
------------

Total Unique Values in Marital_Status column are :- 2
Unique Values in Marital_Status column are :-
 [0, 1]
Categories (2, int64): [0, 1]

--------------------------------------------------------------------------------------------------------------------------------
------------

Total Unique Values in Product_Category column are :- 20
Unique Values in Product_Category column are :-
 [3, 1, 12, 8, 5, ..., 10, 17, 9, 20, 19]
Length: 20
Categories (20, int64): [1, 2, 3, 4, ..., 17, 18, 19, 20]

--------------------------------------------------------------------------------------------------------------------------------
------------

Total Unique Values in Purchase column are :- 18105
Unique Values in Purchase column are :-
 [ 8370 15200  1422 ...   135   123   613]

--------------------------------------------------------------------------------------------------------------------------------
------------
```


In [16]:
```
for _ in wm.columns:
    if wm[_].dtype != 'category':
        print(f'Value_counts of the column {_} are :- \n{wm[_].value_counts().to_frame().reset_index()}')
        print()
        print('-'*140)
        print()
```

```
Value_counts of the column Purchase are :-
       Purchase  count
0          7011    191
1          7193    188
2          6855    187
3          6891    184
4          7012    183
...         ...    ...
18100     23491      1
18101     18345      1
18102      3372      1
18103       855      1
18104     21489      1

[18105 rows x 2 columns]

--------------------------------------------------------------------------------------------------------------------------------
------------
```

In [6]:

```
#replacing the values in marital_status column

wm['Marital_Status'] = wm['Marital_Status'].replace({0:'Single',1:'Married'})
wm['Marital_Status'].unique()
```

Out[6]:

```
['Single', 'Married']
Categories (2, object): ['Single', 'Married']
```

In [18]:

```
wm.sample(2)
```

Out[18]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
303394	1004705	P00051842	F	18-25	0	C	0	Married	4	2761
253603	1003155	P00024342	M	26-35	7	A	1	Single	8	9965

In [19]:

```
wm.Product_ID.nunique() , wm.Product_Category.nunique()
```

Out[19]:

```
(3631, 20)
```

In [20]:

```
cols = ['Gender','Age','Product_Category','City_Category','Occupation','Stay_In_Current_City_Years','Marital_Status']
for _ in cols:
    print(_,'percentage proportions')
    print('*'*(len(_)+23))
    print()
    print((wm[_].value_counts(normalize=True)*100).to_frame().reset_index().round(2))
    print()
    print("-"*140)
    print()
```

```
Gender percentage proportions
*****************************

  Gender  proportion
0      M       75.31
1      F       24.69

------------------------------------------------------------------------------------------------------------------------------------------------

Age percentage proportions
**************************

     Age  proportion
0  26-35       39.92
1  36-45       20.00
2  18-25       18.12
3  46-50        8.31
4  51-55        7.00
5    55+        3.91
6   0-17        2.75

------------------------------------------------------------------------------------------------------------------------------------------------

Product_Category percentage proportions
***************************************

   Product_Category  proportion
0                 5       27.44
1                 1       25.52
2                 8       20.71
3                11        4.42
4                 2        4.34
5                 6        3.72
6                 3        3.67
7                 4        2.14
8                16        1.79
9                15        1.14
10               13        1.01
11               10        0.93
12               12        0.72
13                7        0.68
14               18        0.57
15               20        0.46
16               19        0.29
17               14        0.28
18               17        0.11
19                9        0.07

------------------------------------------------------------------------------------------------------------------------------------------------

City_Category percentage proportions
************************************

  City_Category  proportion
0             B       42.03
1             C       31.12
2             A       26.85

------------------------------------------------------------------------------------------------------------------------------------------------

Occupation percentage proportions
*********************************

   Occupation  proportion
0           4       13.15
1           0       12.66
2           7       10.75
3           1        8.62
4          17        7.28
5          20        6.10
6          12        5.67
7          14        4.96
8           2        4.83
9          16        4.61
10          6        3.70
11          3        3.21
12         10        2.35
13          5        2.21
14         15        2.21
15         11        2.11
16         19        1.54
17         13        1.40
18         18        1.20
19          9        1.14
20          8        0.28

------------------------------------------------------------------------------------------------------------------------------------------------
```

```
Stay_In_Current_City_Years percentage proportions
**************************************************

  Stay_In_Current_City_Years  proportion
0                          1       35.24
1                          2       18.51
2                          3       17.32
3                         4+       15.40
4                          0       13.53

-----------------------------------------------------------------------------------------------------------------------------------
------------

Marital_Status percentage proportions
**************************************

  Marital_Status  proportion
0         Single       59.03
1        Married       40.97

-----------------------------------------------------------------------------------------------------------------------------------
------------
```

In [21]:
```
wm.groupby(['Gender'])[['User_ID']].nunique()
```

Out[21]:

	User_ID
Gender	
F	1666
M	4225

In [22]:
```
wm.groupby(['Gender'])[['User_ID','Purchase']].agg({'User_ID' : 'nunique','Purchase' : 'sum'})
```

Out[22]:

	User_ID	Purchase
Gender		
F	1666	1186232642
M	4225	3909580100

In [23]:
```
cat_cols = ['Gender','Age','Occupation','City_Category','Stay_In_Current_City_Years','Marital_Status', 'Product_Category']
cat_cols_melt = wm[cat_cols].melt()
cat_cols_melt
```

Out[23]:

	variable	value
0	Gender	F
1	Gender	F
2	Gender	F
3	Gender	F
4	Gender	M
...	...	...
3850471	Product_Category	20
3850472	Product_Category	20
3850473	Product_Category	20
3850474	Product_Category	20
3850475	Product_Category	20

3850476 rows × 2 columns

In [24]:
```
(cat_cols_melt.groupby(['variable','value'])[['value']].count() / len(wm['User_ID'])*100).round(2)
```

Out[24]:

variable	value	value
Age	0-17	2.75
	18-25	18.12
	26-35	39.92
	36-45	20.00
	46-50	8.31
	51-55	7.00
	55+	3.91
City_Category	A	26.85
	B	42.03
	C	31.12
Gender	F	24.69
	M	75.31
Marital_Status	Married	40.97
	Single	59.03
Occupation	0	12.66
	1	8.62
	2	4.83
	3	3.21
	4	13.15
	5	2.21
	6	3.70
	7	10.75
	8	0.28
	9	1.14
	10	2.35
	11	2.11
	12	5.67
	13	1.40
	14	4.96
	15	2.21
	16	4.61
	17	7.28
	18	1.20
	19	1.54
	20	6.10
Product_Category	1	25.52
	2	4.34
	3	3.67
	4	2.14
	5	27.44
	6	3.72
	7	0.68
	8	20.71
	9	0.07
	10	0.93
	11	4.42
	12	0.72
	13	1.01
	14	0.28
	15	1.14
	16	1.79

variable	value	value
	17	0.11
	18	0.57
	19	0.29
	20	0.46
Stay_In_Current_City_Years	0	13.53
	1	35.24
	2	18.51
	3	17.32
	4+	15.40

😳 Observations:

- We can see that there are `75%` of the **Male customers** in the data and `25%` of the **Female customers** have purchased.
- We can also see that the **Male customers** have `purchased more products` from Walmart than the *Female customers*.
- We can also observe that the 40% of the customers are from the Age range of `26-35`. The second highest is 20% for *36-45 Age* range.
- We can see that `40%` of the customers are from the **city category of B** and `30%` of the customers are from *category is C*.
- From the data we can see that `59%` of the customers are **single**.
- Highest product sold is of **Product Category 5** with `55.2% sales`. Second highest product sold is of Product Category 1 with 51.3%

In [25]:

```
print(f"Genderwise distribution".upper())
print()

for _ in wm.columns[:-1]:
    if _ not in ["User_ID","Purchase","Gender"]:
        print("_" * len(f"For {_}"))
        print(f"For {_} :".upper())
        print("_" * len(f"For {_}"))
        print()
        print(f"{'---> '}Total Count of Users based on {_}")
        print()
        print(pd.crosstab(index = wm["Gender"], columns = wm[_],
                          values = wm["User_ID"], aggfunc = "count", margins = True))
        ax = sns.heatmap(wmcorr,annot=True,fmt='.3f',linewidths=.5,cmap=cp2)
        print("")
        print(f"{'---> '}Total Purchase amount per user based on {_}")
        print()
        print(pd.crosstab(index = wm["Gender"], columns = wm[_],
                          values = wm["Purchase"], aggfunc = "sum", margins = True))
        print()
        print("")
        print(f"{'---> '}Total Purchase amount per user in percentage based on {_}")
        print()
        print((pd.crosstab(index = wm["Gender"], columns = wm[_], values = wm["Purchase"],
                           aggfunc = "sum", margins = True, normalize = True)*100).round(2))
        print("_" * 140)
```

```
GENDERWISE DISTRIBUTION

______________
FOR PRODUCT_ID :
______________

---> Total Count of Users based on Product_ID

Product_ID  P00000142  P00000242  P00000342  P00000442  P00000542  P00000642  \
Gender
F                 347         91         69         46         50         71
M                 805        285        175         46         99        441
All              1152        376        244         92        149        512

Product_ID  P00000742  P00000842  P00000942  P00001042  ...  P0099042  \
Gender                                                  ...
F                 117         14          7         76  ...        51
M                 124         22         48        427  ...        93
All               241         36         55        503  ...       144

Product_ID  P0099142  P0099242  P0099342  P0099442  P0099642  P0099742  \
Gender
F                  0        91        89        53         4        44
M                  7       166       351       147         9        82
All                7       257       440       200        13       126

Product_ID  P0099842  P0099942     All
Gender
F                 51         7  135809
M                 51         7  414259
All              102        14  550068

[3 rows x 3632 columns]

---> Total Purchase amount per user based on Product_ID

Product_ID  P00000142  P00000242  P00000342  P00000442  P00000542  P00000642  \
Gender
F             3915603     953005     382566     236177     283740    1016899
M             8921873    3014491     913909     204996     523472    6618679
All          12837476    3967496    1296475     441173     807212    7635578

Product_ID  P00000742  P00000842  P00000942  P00001042  ...  P0099042  \
Gender                                                  ...
F              734394     138434      51613    1038258  ...    302002
M              719171     221880     529512    5884122  ...    596457
All           1453565     360314     581125    6922380  ...    898459

Product_ID  P0099142  P0099242  P0099342  P0099442  P0099642  P0099742  \
Gender
F                  0    656169    609919    748812     27925    333487
M              45914   1093576   2444179   2121571     55785    658461
All            45914   1749745   3054098   2870383     83710    991948

Product_ID  P0099842  P0099942         All
Gender
F             383831     42447  1186232642
M             353481     35572  3909580100
All           737312     78019  5095812742

[3 rows x 3632 columns]


---> Total Purchase amount per user in percentage based on Product_ID

Product_ID  P00000142  P00000242  P00000342  P00000442  P00000542  P00000642  \
Gender
F                0.08       0.02       0.01       0.00       0.01       0.02
M                0.18       0.06       0.02       0.00       0.01       0.13
All              0.25       0.08       0.03       0.01       0.02       0.15

Product_ID  P00000742  P00000842  P00000942  P00001042  ...  P0099042  \
Gender                                                  ...
F                0.01       0.00       0.00       0.02  ...      0.01
M                0.01       0.00       0.01       0.12  ...      0.01
All              0.03       0.01       0.01       0.14  ...      0.02

Product_ID  P0099142  P0099242  P0099342  P0099442  P0099642  P0099742  \
Gender
F                0.0      0.01      0.01      0.01       0.0      0.01
M                0.0      0.02      0.05      0.04       0.0      0.01
All              0.0      0.03      0.06      0.06       0.0      0.02

Product_ID  P0099842  P0099942     All
Gender
F               0.01       0.0   23.28
M               0.01       0.0   76.72
All             0.01       0.0  100.00

[3 rows x 3632 columns]
```

```
FOR AGE :
_______

---> Total Count of Users based on Age

Age      0-17  18-25   26-35   36-45  46-50  51-55    55+     All
Gender
F        5083  24628   50752   27170  13199   9894   5083  135809
M       10019  75032  168835   82843  32502  28607  16421  414259
All     15102  99660  219587  110013  45701  38501  21504  550068

---> Total Purchase amount per user based on Age

Age          0-17      18-25       26-35       36-45      46-50      51-55  \
Gender
F        42385978  205475842   442976233   243438963  116706864   89465997
M        92527205  708372833  1588794345   783130921  304136539  277633647
All     134913183  913848675  2031770578  1026569884  420843403  367099644

Age           55+         All
Gender
F        45782765  1186232642
M       154984610  3909580100
All     200767375  5095812742


---> Total Purchase amount per user in percentage based on Age

Age     0-17  18-25  26-35  36-45  46-50  51-55   55+     All
Gender
F       0.83   4.03   8.69   4.78   2.29   1.76  0.90   23.28
M       1.82  13.90  31.18  15.37   5.97   5.45  3.04   76.72
All     2.65  17.93  39.87  20.15   8.26   7.20  3.94  100.00
____________________________________________________________________________________________________________
____________

______________
FOR OCCUPATION :
______________

---> Total Count of Users based on Occupation

Occupation      0      1      2      3      4      5      6      7     8  \
Gender
F           18112  17984   8629   7919  17836   2220   8160  10028   361
M           51526  29442  17959   9731  54472   9957  12195  49105  1185
All         69638  47426  26588  17650  72308  12177  20355  59133  1546

Occupation     9  ...      12    13     14     15     16     17    18    19  \
Gender            ...
F           5843  ...    3469  1498   6763   2390   4107   3929   230  2017
M            448  ...   27710  6230  20546   9775  21264  36114  6392  6444
All         6291  ...   31179  7728  27309  12165  25371  40043  6622  8461

Occupation     20     All
Gender
F            8811  135809
M           24751  414259
All         33562  550068

[3 rows x 22 columns]

---> Total Purchase amount per user based on Occupation

Occupation          0          1          2          3          4          5  \
Gender
F           159883833  152806726   72569470   71707639  152264321   19595050
M           475523125  271807418  165459113   90294529  513980163   94054709
All         635406958  424614144  238028583  162002168  666244484  113649759

Occupation          6          7         8         9  ...         12  \
Gender                                              ...
F            74079792   91177610   3379484  50206487  ...   31762002
M           114336992  466193977  11357904   4133559  ...  273687444
All         188416784  557371587  14737388  54340046  ...  305449446

Occupation        13         14         15         16         17        18  \
Gender
F           12827008   58010060   22453799   36820127   37496159   2317160
M           59092473  201444632   96506412  201526828  355785294  58404301
All         71919481  259454692  118960211  238346955  393281453  60721461

Occupation        19         20          All
Gender
F           17007150   73428976   1186232642
M           56693467  223141466   3909580100
All         73700617  296570442   5095812742

[3 rows x 22 columns]


---> Total Purchase amount per user in percentage based on Occupation

Occupation      0     1     2     3      4     5     6      7     8     9  \
```

```
Gender
F            3.14  3.00  1.42  1.41   2.99  0.38  1.45   1.79  0.07  0.99
M            9.33  5.33  3.25  1.77  10.09  1.85  2.24   9.15  0.22  0.08
All         12.47  8.33  4.67  3.18  13.07  2.23  3.70  10.94  0.29  1.07

Occupation  ...    12    13    14    15    16    17    18    19    20     All
Gender      ...
F           ...  0.62  0.25  1.14  0.44  0.72  0.74  0.05  0.33  1.44   23.28
M           ...  5.37  1.16  3.95  1.89  3.95  6.98  1.15  1.11  4.38   76.72
All         ...  5.99  1.41  5.09  2.33  4.68  7.72  1.19  1.45  5.82  100.00

[3 rows x 22 columns]
____________________________________________________________________________________________________________
____________
_________________
FOR CITY_CATEGORY :
_________________

---> Total Count of Users based on City_Category

City_Category       A       B       C     All
Gender
F               35704   57796   42309  135809
M              112016  173377  128866  414259
All            147720  231173  171175  550068

---> Total Purchase amount per user based on City_Category

City_Category           A           B           C         All
Gender
F               306329915   493617008   386285719  1186232642
M              1010141746  1621916597  1277521757  3909580100
All            1316471661  2115533605  1663807476  5095812742


---> Total Purchase amount per user in percentage based on City_Category

City_Category      A      B      C     All
Gender
F               6.01   9.69   7.58   23.28
M              19.82  31.83  25.07   76.72
All            25.83  41.52  32.65  100.00
____________________________________________________________________________________________________________
____________
______________________________
FOR STAY_IN_CURRENT_CITY_YEARS :
______________________________

---> Total Count of Users based on Stay_In_Current_City_Years

Stay_In_Current_City_Years      0       1       2      3     4+     All
Gender
F                           17063   51298   24332  24520  18596  135809
M                           57335  142523   77506  70765  66130  414259
All                         74398  193821  101838  95285  84726  550068

---> Total Purchase amount per user based on Stay_In_Current_City_Years

Stay_In_Current_City_Years          0           1          2          3  \
Gender
F                           146844869   450142630  212674244  213207201
M                           536134360  1342729903  736499687  671695458
All                         682979229  1792872533  949173931  884902659

Stay_In_Current_City_Years         4+         All
Gender
F                           163363698  1186232642
M                           622520692  3909580100
All                         785884390  5095812742


---> Total Purchase amount per user in percentage based on Stay_In_Current_City_Years

Stay_In_Current_City_Years      0      1      2      3     4+     All
Gender
F                            2.88   8.83   4.17   4.18   3.21   23.28
M                           10.52  26.35  14.45  13.18  12.22   76.72
All                         13.40  35.18  18.63  17.37  15.42  100.00
____________________________________________________________________________________________________________
____________
__________________
FOR MARITAL_STATUS :
__________________

---> Total Count of Users based on Marital_Status

Marital_Status  Single  Married     All
Gender
F                78821    56988  135809
M               245910   168349  414259
All             324731   225337  550068

---> Total Purchase amount per user based on Marital_Status
```

```
Marital_Status      Single     Married         All
Gender
F                684154127   502078515  1186232642
M               2324773320  1584806780  3909580100
All             3008927447  2086885295  5095812742


---> Total Purchase amount per user in percentage based on Marital_Status

Marital_Status  Single  Married     All
Gender
F                13.43     9.85   23.28
M                45.62    31.10   76.72
All              59.05    40.95  100.00
```

```
____________________
FOR PRODUCT_CATEGORY :
____________________

---> Total Count of Users based on Product_Category

Product_Category       1      2      3      4       5      6     7       8  \
Gender
F                  24831   5658   6006   3639   41961   4559   943   33558
M                 115547  18206  14207   8114  108972  15907  2778   80367
All               140378  23864  20213  11753  150933  20466  3721  113925

Product_Category    9    10  ...    12    13    14    15    16   17    18  \
Gender                       ...
F                  70  1162  ...  1532  1462   623  1046  2402   62   382
M                 340  3963  ...  2415  4087   900  5244  7426  516  2743
All               410  5125  ...  3947  5549  1523  6290  9828  578  3125

Product_Category    19    20     All
Gender
F                  451   723  135809
M                 1152  1827  414259
All               1603  2550  550068

[3 rows x 21 columns]

---> Total Purchase amount per user based on Product_Category

Product_Category           1          2          3         4          5  \
Gender
F                  337631145   64543617   61637516   8933206  264658078
M                 1572382609  203972569  142447197  18447282  677177151
All               1910013754  268516186  204084713  27380488  941835229

Product_Category          6          7          8        9         10  ...  \
Gender                                                                 ...
F                  71104116  15460347  251682476  1100702   22882193  ...
M                 253046186  45436384  602636323  5269622   77955108  ...
All               324150302  60896731  854318799  6370324  100837301  ...

Product_Category       12       13        14        15         16       17  \
Gender
F                 2179897  1072884   8564607  15371312   35264942   610477
M                 3151947  2935717  11450089  77597730  109855670  5268222
All               5331844  4008601  20014696  92969042  145120612  5878699

Product_Category       18     19      20         All
Gender
F                 1088168  16992  268641  1186232642
M                 8202033  42386  676086  3909580100
All               9290201  59378  944727  5095812742

[3 rows x 21 columns]


---> Total Purchase amount per user in percentage based on Product_Category

Product_Category      1     2     3     4      5     6     7      8     9  \
Gender
F                  6.63  1.27  1.21  0.18   5.19  1.40  0.30   4.94  0.02
M                 30.86  4.00  2.80  0.36  13.29  4.97  0.89  11.83  0.10
All               37.48  5.27  4.00  0.54  18.48  6.36  1.20  16.77  0.13

Product_Category    10  ...    12    13    14    15    16    17    18   19  \
Gender                  ...
F                 0.45  ...  0.04  0.02  0.17  0.30  0.69  0.01  0.02  0.0
M                 1.53  ...  0.06  0.06  0.22  1.52  2.16  0.10  0.16  0.0
All               1.98  ...  0.10  0.08  0.39  1.82  2.85  0.12  0.18  0.0

Product_Category    20     All
Gender
F                 0.01   23.28
M                 0.01   76.72
All               0.02  100.00

[3 rows x 21 columns]
```

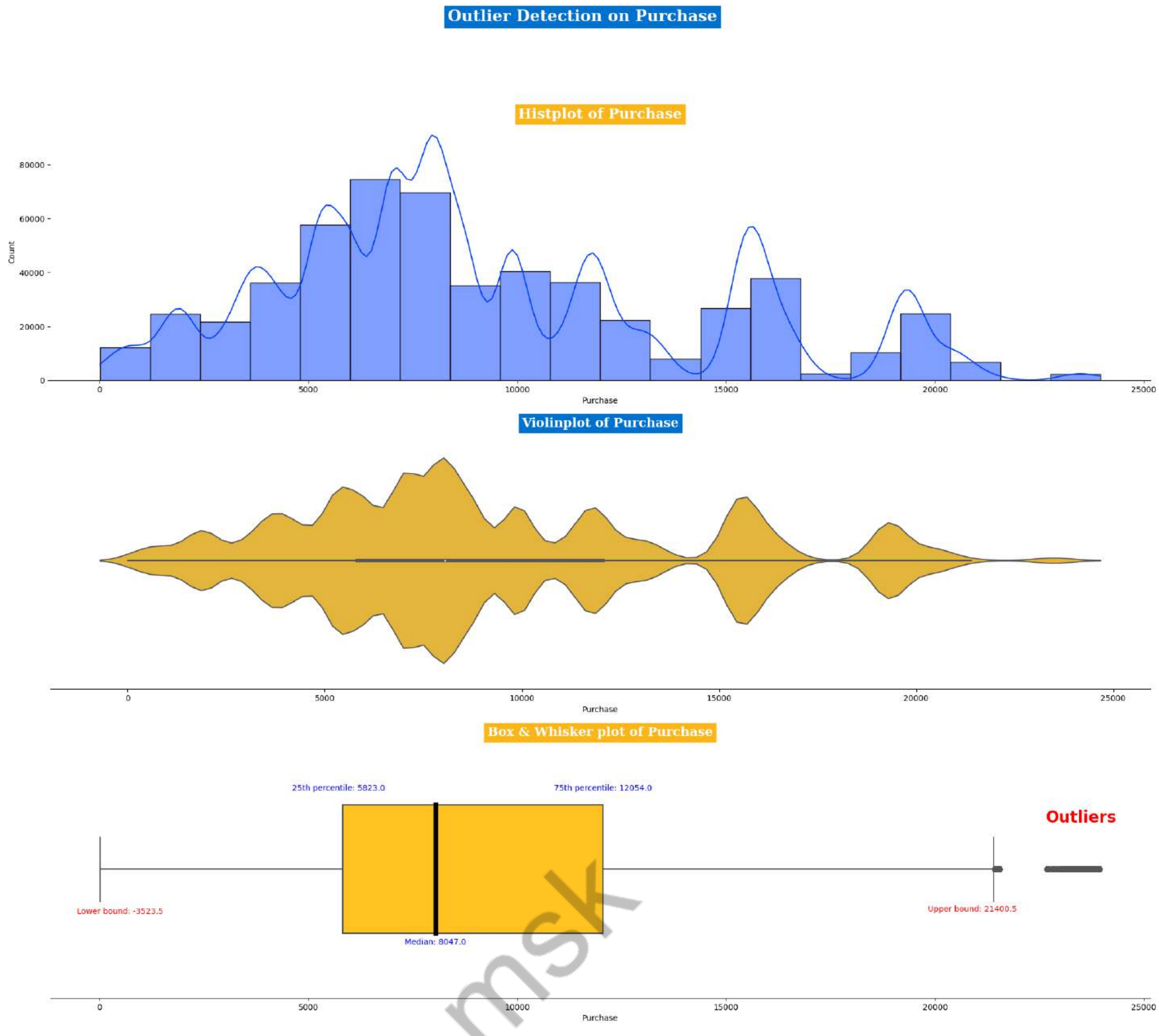
In [59]:
```
wm.columns
```

Out[59]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

In [168…
```
num_cols = ['Purchase']

for i in range(len(num_cols)):
    data = wm[num_cols[i]].tolist()
    mini = np.min(data)
    Q1 = np.percentile(data, 25)
    Q2 = np.median(data)  #percentile(data, 50)
    Q3 = np.percentile(data, 75)
    maxi = np.max(data)
    IQR = Q3 - Q1

    lo = Q1 - (1.5 * IQR)
    ho = Q3 + (1.5 * IQR)

    lower_outliers=[]
    upper_outliers=[]
    for k in data:
        if k < lo:
            lower_outliers.append(k)

        elif k > ho:
            upper_outliers.append(k)

    uo_pct = round((len(upper_outliers)*100/wm.shape[0]),2)
    lo_pct = round((len(lower_outliers)*100/wm.shape[0]),2)

    print()
    print(f"Outlier detection of {num_cols[i]}")
    print('.'*30)
    print("Minimum:", mini)
    print("Maximum:", maxi)
    print()
    print(f'Initial Range (with outlier) : {(maxi-mini)}')
    print("Q1:", Q1)
    print("Q2:", Q2)
    print("Q3:", Q3)
    print("IQR:", IQR)
    print("Lower bound:",lo)
    print("upper bound:",ho)
    print(f'Final Range (without outlier) : {(ho-lo)}')
    print()
    # print("Lower outliers are:", lower_outliers)
    # print("Upper outliers are:", upper_outliers)
    print(f'Lower Outlier Percentage is {lo_pct}%')
    print(f'Upper Outlier Percentage is {uo_pct}%')
    print(f'Overall Outlier Percentage is {(lo_pct+uo_pct)}%')
    print()

    if len(set(lower_outliers)):
        print(f'Outlier points towards left of boxplot : {len(set(lower_outliers))} and they are {(set(lower_outliers))}')
    else:
        print(f'Outlier points towards left of boxplot : {len(set(lower_outliers))}')
    if len(set(upper_outliers)):
        print(f'Outlier points towards right of boxplot : {len(set(upper_outliers))} and they are {(set(upper_outliers))}')
    else:
        print(f'Outlier points towards right of boxplot : {len(set(upper_outliers))}')
    print()

    plt.figure(figsize=(25,20))
    plt.style.use('default')
    plt.style.use('seaborn-v0_8-bright')
    plt.suptitle(f'Outlier Detection on {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=20,
              backgroundcolor=cp2[i],color='w')
    print()
    plt.subplot(3,1,1)
    plt.title(f'Histplot of {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=18,
              loc='center',backgroundcolor=cp2[i+1],color='w')
    sns.histplot(wm[num_cols[i]],bins=20,kde=True)

    plt.subplot(3,1,2)
    sns.violinplot(wm,x=num_cols[i],color=cp1[i+1])
    plt.title(f'Violinplot of {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=16,
              loc='center',backgroundcolor=cp1[i],color='w')
    plt.yticks([])
    print()
    plt.subplot(3,1,3)
    bxp = sns.boxplot(wm,x=num_cols[i],color=cp1[i+1],width=0.5,saturation=97,flierprops={"marker":"d"},
                      medianprops={"color": "k", "linewidth": 6})#,boxprops={"facecolor": (.3, .5, .7, .5)})
    plt.title(f'Box & Whisker plot of {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=16,
              loc='center',backgroundcolor=cp2[i+1],color='w')
    sns.despine(left=True)
    plt.yticks([])
    plt.text(Q2, 0.3 , f'Median: {Q2:.1f}',ha='center', va='bottom',fontsize=10,color='b')
```

```
    plt.text(Q1-100, -0.3 , f'25th percentile: {Q1:.1f}',ha='center', va='bottom',fontsize=10,color='b')
    plt.text(Q3, -0.3 , f'75th percentile: {Q3:.1f}',ha='center', va='bottom',fontsize=10,color='b')
    plt.text(500, 0.18 , f'Lower bound: {lo}',ha='center', va='bottom',fontsize=10,color='r')
    plt.text(ho-500, 0.17 , f'Upper bound: {ho}',ha='center', va='bottom',fontsize=10,color='r')
    plt.text(23500, -0.17 , 'Outliers',ha='center', va='bottom',fontsize=20,color='r',fontweight='bold')

    #bxp.annotate(f'text={Q2:.2f}',xy=(Q2,1),xytext=(Q2+1,1.5),textcoords='offset points',
    #             ha='center',fontsize=10,color='red',arrowprops= dict(arrowstyle="<-", lw=1, connectionstyle="arc,rad=0"))
# takes longer time
    # plt.subplot(4,4,1)
    # sns.swarmplot(wm,x=num_cols[i],color=cp2[i])
    # plt.title(f'Swarmplot of {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=12,
    #            loc='center',backgroundcolor=cp2[i+1],color='w')
    # sns.despine(left=True)
    # plt.yticks([])
print('-'*144)
```

```
Outlier detection of Purchase
..............................
Minimum: 12
Maximum: 23961

Initial Range (with outlier) : 23949
Q1: 5823.0
Q2: 8047.0
Q3: 12054.0
IQR: 6231.0
Lower bound: -3523.5
upper bound: 21400.5
Final Range (without outlier) : 24924.0

Lower Outlier Percentage is 0.0%
Upper Outlier Percentage is 0.49%
Overall Outlier Percentage is 0.49%

Outlier points towards left of boxplot : 0
Outlier points towards right of boxplot : 1027 and they are {23610, 23612, 23624, 23856, 23630, 22651, 22656, 22666, 22668, 2267
8, 22684, 22710, 22719, 22730, 22743, 23724, 22791, 22795, 22803, 23485, 22814, 22816, 22823, 22846, 22848, 22852, 22855, 22858,
22864, 22871, 21425, 22902, 22913, 22919, 22942, 22946, 22963, 22976, 22984, 22989, 22990, 22994, 21429, 23006, 23482, 23029, 23
040, 23041, 23042, 23043, 23044, 23045, 23046, 23047, 23048, 23049, 23050, 23051, 23052, 23053, 23054, 23055, 23056, 23057, 2305
8, 23059, 23060, 23061, 23062, 23063, 23064, 23065, 23066, 23067, 23068, 23069, 23070, 23071, 23072, 23073, 23074, 23075, 23076,
23077, 23466, 23080, 23081, 23082, 23083, 23084, 23085, 23086, 23087, 23088, 23089, 23090, 23091, 23092, 23094, 23095, 23096, 23
097, 23098, 23099, 23101, 23102, 23103, 23104, 23105, 23106, 23107, 23108, 23109, 23110, 23111, 23112, 23113, 23114, 23115, 2311
6, 23117, 23118, 23119, 23120, 23121, 23122, 23123, 23124, 23125, 23126, 23127, 23128, 23129, 23130, 23131, 23132, 23133, 23136,
23137, 23138, 23139, 23140, 23141, 23142, 23143, 23144, 23145, 23146, 23147, 23148, 23149, 23150, 23151, 23152, 23153, 23154, 23
155, 23156, 23157, 23158, 23159, 23160, 23161, 23162, 23163, 23164, 23165, 23166, 23167, 23168, 23169, 23170, 23171, 23172, 2317
4, 23175, 23176, 23177, 23178, 23179, 23180, 23181, 23182, 23183, 23184, 23185, 23186, 23187, 23188, 23189, 23190, 23192, 23193,
23195, 23196, 23197, 23199, 23200, 23201, 23202, 23203, 23204, 23205, 23207, 23208, 23209, 23210, 23211, 23212, 23214, 23215, 23
216, 23217, 23218, 23219, 23220, 23221, 23222, 23223, 23224, 23225, 23226, 23227, 23228, 23229, 23230, 21438, 23231, 23233, 2323
4, 23235, 23236, 23237, 23238, 23239, 23240, 23241, 23242, 23243, 23244, 23246, 23247, 23248, 23249, 23250, 23251, 23252, 23253,
23254, 23255, 23256, 23257, 23258, 23259, 23260, 23261, 23262, 23263, 23265, 23266, 23267, 23268, 23269, 23270, 23271, 23272, 23
273, 23274, 23275, 23276, 23277, 23278, 23279, 23280, 23281, 23282, 23283, 23284, 23285, 23891, 23287, 23288, 23289, 23290, 2329
1, 23292, 23293, 23295, 23296, 23297, 23300, 23301, 23302, 23303, 23304, 23305, 23306, 23307, 23308, 23309, 23310, 21441, 23312,
23313, 23314, 23315, 23316, 23317, 23318, 23319, 23320, 23321, 23322, 23323, 23325, 23326, 23327, 23328, 23329, 23330, 23331, 23
332, 23333, 23334, 23335, 23336, 23337, 23338, 23339, 23340, 23341, 23342, 23343, 23344, 23345, 23346, 23347, 23348, 23349, 2335
0, 23351, 23352, 23353, 23354, 23355, 23356, 23357, 23358, 23359, 23360, 23361, 23362, 23363, 23364, 23365, 23366, 23367, 23368,
23369, 23370, 23371, 23372, 23373, 23374, 23376, 23378, 23380, 23381, 23383, 23384, 23385, 23386, 23387, 23388, 23389, 23390, 23
391, 23392, 23393, 23394, 23395, 23396, 23397, 23398, 23399, 23400, 23401, 23402, 23403, 23404, 23405, 23406, 23407, 23408, 2340
9, 23410, 21445, 23412, 23411, 23414, 23415, 23416, 23417, 23418, 23419, 23420, 23421, 23422, 23423, 23424, 23425, 23426, 23427,
23428, 23429, 23481, 23431, 23432, 23433, 23789, 23435, 23434, 23437, 23438, 23439, 23441, 23442, 23443, 23445, 23446, 23447, 23
448, 23449, 21402, 23451, 21401, 23452, 23454, 23455, 23456, 23457, 21410, 21411, 23460, 21412, 23462, 21415, 23463, 21417, 2141
8, 21419, 21416, 21421, 23470, 21423, 21424, 23472, 21422, 23475, 23476, 21427, 21430, 23479, 23478, 21433, 21431, 21435, 23484,
21437, 21436, 23487, 21440, 23489, 23490, 23491, 23492, 23488, 23486, 21447, 23496, 23497, 21449, 23499, 21452, 21453, 23502, 23
501, 21455, 21451, 23506, 23507, 23508, 23509, 23510, 23511, 23512, 21462, 21464, 23515, 21468, 21469, 23518, 23517, 21472, 2352
1, 21471, 21475, 23524, 21476, 23523, 23525, 23528, 21481, 21482, 23531, 21477, 23533, 23527, 21487, 23535, 21489, 21488, 23538,
23539, 23541, 21494, 21495, 23544, 23545, 23546, 23547, 21500, 23542, 23550, 21503, 21504, 21505, 23551, 21507, 23556, 23557, 23
558, 21510, 23555, 21508, 23562, 21512, 23564, 23565, 21518, 21516, 23567, 21519, 21522, 21523, 21524, 21525, 23574, 21526, 2152
8, 21529, 21530, 23578, 23579, 23581, 23582, 21533, 21536, 23585, 23586, 23587, 21540, 23589, 21539, 21543, 21544, 23592, 21546,
23595, 23596, 23594, 23591, 23598, 23599, 23601, 21554, 23603, 23604, 21553, 21555, 23607, 23608, 21561, 23609, 21560, 21563, 21
565, 23611, 21567, 23615, 21569, 23616, 21568, 23620, 23614, 23621, 23619, 23622, 23625, 23626, 23627, 23628, 23629, 23623, 2363
1, 23632, 23633, 23634, 23635, 23636, 23637, 23638, 23639, 23640, 23641, 23642, 23643, 23644, 23645, 23646, 23647, 23648, 23649,
23650, 23651, 23652, 23653, 23654, 23655, 23657, 23659, 23660, 23663, 23664, 23665, 23666, 23667, 23668, 23669, 23670, 23671, 23
672, 23674, 23675, 23676, 23677, 23678, 23679, 23680, 23681, 23682, 23683, 23684, 23685, 23686, 23687, 23689, 23690, 23691, 2369
2, 23693, 23694, 23695, 23696, 23697, 23698, 23699, 23700, 23701, 23702, 23703, 23704, 23705, 23706, 23708, 23709, 23710, 23711,
23713, 23714, 23715, 23716, 23717, 23718, 23719, 23720, 21404, 23722, 23723, 23721, 23725, 21405, 23727, 23728, 23729, 23730, 23
731, 23732, 23726, 23733, 23735, 21406, 23736, 23738, 23739, 23740, 23741, 23742, 23743, 21408, 23737, 23744, 23747, 23748, 2374
9, 23750, 23751, 23752, 23753, 23754, 23755, 23756, 23757, 23758, 23759, 23760, 23761, 23762, 23763, 23764, 23765, 23766, 23767,
23768, 23769, 23770, 23771, 23772, 23773, 23774, 23450, 23775, 23777, 23778, 23779, 23780, 23781, 23776, 23783, 23784, 23785, 23
786, 23787, 23788, 23782, 23790, 23453, 23792, 23793, 23794, 23795, 23796, 23797, 23798, 23799, 23800, 23802, 23804, 23863, 2380
6, 23807, 23808, 23809, 23810, 23811, 23812, 23813, 23805, 23815, 23816, 23817, 23819, 23820, 23821, 23822, 23823, 23824, 23825,
23826, 23459, 23827, 23829, 23830, 23831, 23461, 23833, 23834, 23835, 23836, 23837, 23838, 23839, 23840, 23841, 23842, 23843, 23
844, 21428, 23464, 23847, 23848, 23849, 23850, 23845, 23852, 23853, 23854, 23851, 23465, 23855, 23858, 23859, 23860, 23861, 2346
7, 23857, 23864, 23865, 23866, 23867, 23868, 23468, 23869, 23469, 23871, 23870, 23874, 23875, 23876, 23877, 23878, 23879, 23881,
23471, 23883, 23884, 23885, 23887, 23888, 23889, 23890, 23473, 23892, 23893, 23894, 23895, 23896, 23897, 21439, 23899, 23900, 23
474, 23902, 23903, 23904, 23905, 23906, 23907, 23908, 23909, 23910, 23477, 23912, 23913, 23914, 23915, 23916, 21442, 23918, 2391
9, 23920, 23921, 21443, 23923, 23924, 21444, 23480, 23927, 23928, 23929, 23930, 23931, 23932, 23933, 23934, 23935, 23936, 21446,
23938, 23939, 23940, 23941, 23942, 23483, 23944, 21448, 23946, 23947, 23948, 23949, 23943, 23950, 23952, 23953, 23954, 21450, 23
956, 23951, 23958, 23959, 23960, 23961, 23955, 23828, 21454, 23898, 21456, 23494, 21459, 23495, 21461, 23734, 23498, 21463, 2146
6, 21467, 23504, 23911, 23505, 21470, 21474, 23513, 21478, 23514, 21479, 23925, 23519, 23926, 21484, 23520, 21485, 21486, 23522,
21490, 21491, 21493, 23529, 23530, 23937, 23532, 21497, 23534, 21499, 21501, 23537, 21502, 23945, 23540, 21506, 23543, 21509, 21
511, 23548, 21513, 23549, 21514, 21517, 23553, 23554, 21520, 21521, 23561, 23563, 23566, 21532, 23568, 23569, 21535, 23572, 2153
7, 21538, 23575, 23576, 21541, 23577, 21542, 23846, 23580, 21547, 21548, 23584, 21550, 21551, 21552, 23588, 23590, 21557, 23593,
21558, 21559, 21562, 21564, 23600, 21566, 23602, 21409, 23605}


-----------------------------------------------------------------------------------------------------------------------------
----------------
```



🏷️ Insights

- **Outliers**
  - There are total of `1027 outliers` which is roughly `0.49%` of the total data present in purchase amount. We will not remove them as it indicates a broad range of spending behaviors during the sale, highlighting the importance of tailoring marketing strategies to both regular and high-value customers to maximize revenue.
- **Distribution**
  - Data suggests that the majority of customers spent between `5,823 USD` and `12,054 USD`, with the `median` purchase amount being `8,047 USD`.
  - The lower limit of `12 USD` while the upper limit of `21,400 USD` reveal significant variability in customer spending

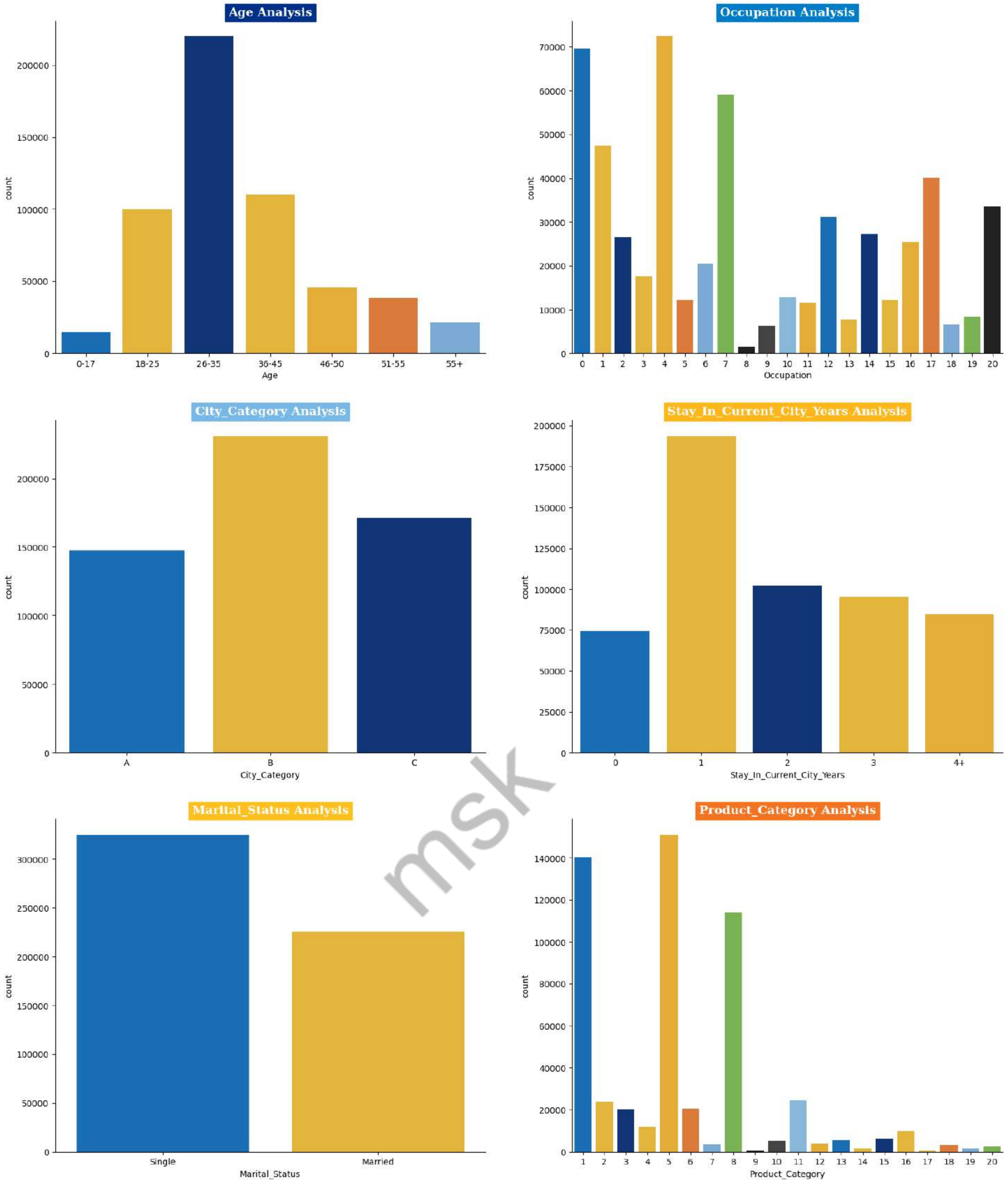
---

📈 Graphical Analysis

📊 Univariate & Bivariate

In [232…

```
cat_cols = ['Age','Occupation','City_Category','Stay_In_Current_City_Years','Marital_Status', 'Product_Category']
plt.figure(figsize = (20,24))
plt.style.use('seaborn-v0_8-bright')

for i in range(len(cat_cols)):
    plt.subplot(3,2,i+1)
    sns.countplot(data = wm, x = cat_cols[i], palette = cp3)
    sns.despine()
    plt.title(f'{cat_cols[i]} Analysis',fontsize=14,fontfamily='serif',fontweight='bold',backgroundcolor=cp[i],color='w')
```



Insights

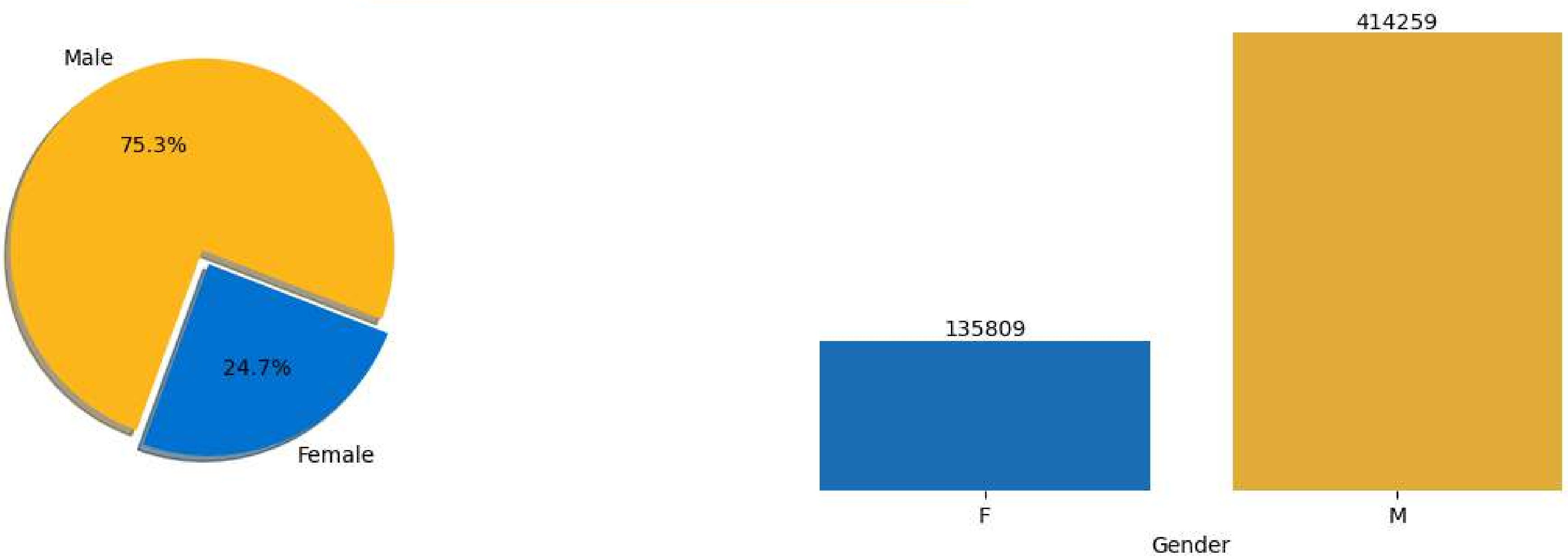
- **Observation**
  - Customers in the age group of `26-35` have purchased **more** than other age groups.
  - Customers from `City_Category B` have purchased **more** than other city customers.
  - Customers who stay more than `1 year` in a **city** has the tendency to visit and purchase more.
  - Customers who are `Single` has purchased more than the customers who are married.

In [224…

```
plt.figure(figsize=(15,4))
plt.suptitle("Genderwise Purchase Distribution",fontsize=14,fontfamily='serif',fontweight='bold'
             ,backgroundcolor=cp1[0],color='w')
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')

plt.subplot(1,2,1)
plt.pie(wm.groupby('Gender')['Gender'].count(), labels = ['Female','Male'], explode = (0.08,0),
        colors=cp2,shadow=True, startangle=250, autopct='%1.1f%%',textprops={'color':"k"})
```

```
plt.subplot(1,2,2)
label = sns.countplot(data = wm, x='Gender',palette=cp2)
sns.despine(left=True,bottom=True)
plt.yticks([])
plt.ylabel('')
for i in label.containers:
    label.bar_label(i)

plt.show()
```

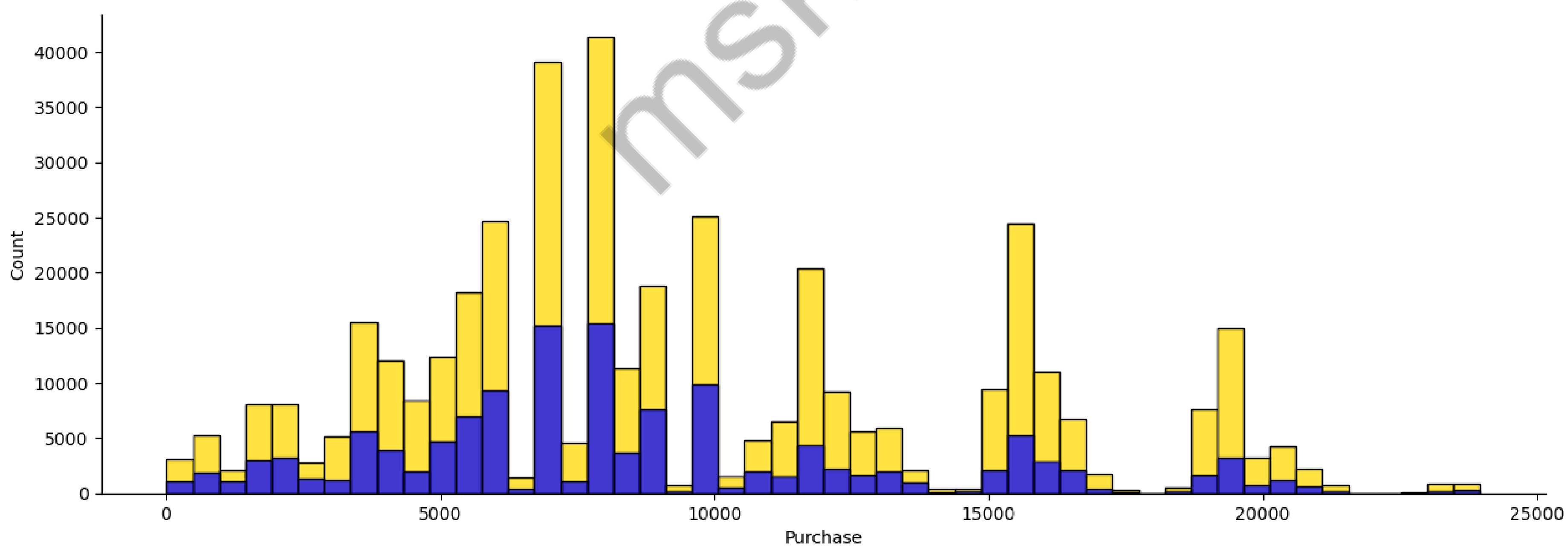
Genderwise Purchase Distribution



In [220…

```
plt.figure(figsize=(15,5))
plt.suptitle("Genderwise Purchase Distribution - Histplot",fontsize=14,fontfamily='serif',fontweight='bold'
             ,backgroundcolor='b',color='w')
plt.style.use('seaborn-v0_8-bright')
sns.histplot(data=wm[wm['Gender']=='M']['Purchase'], bins = 50,color='gold')
sns.histplot(data=wm[wm['Gender']=='F']['Purchase'], bins = 50,color='b')
sns.despine()
plt.show()
```

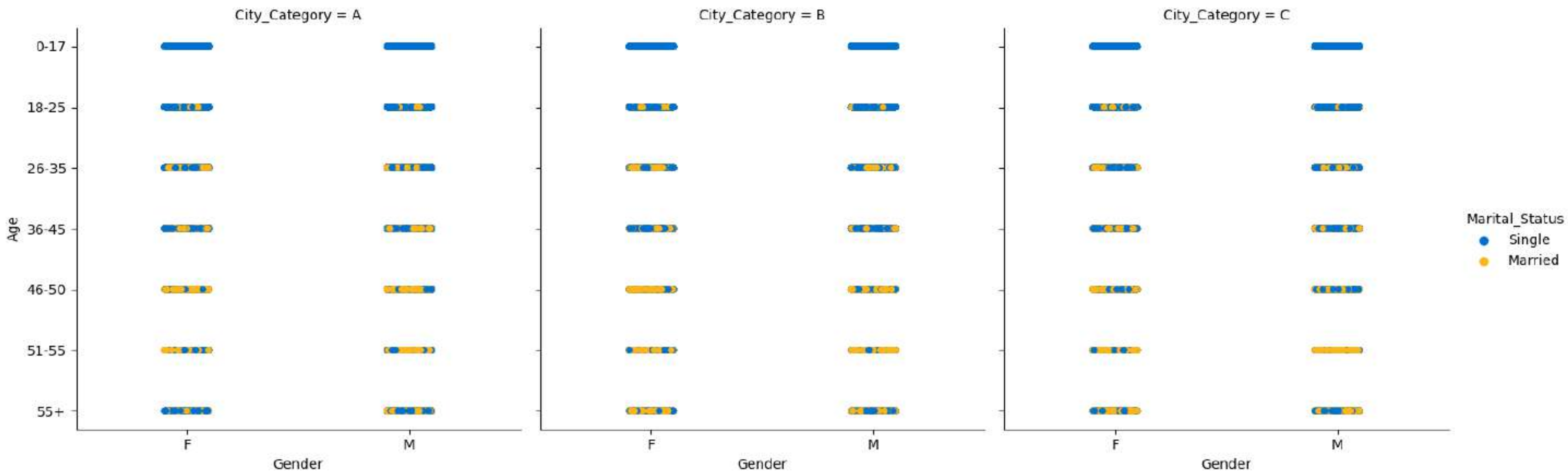
Genderwise Purchase Distribution - Histplot



In [211…

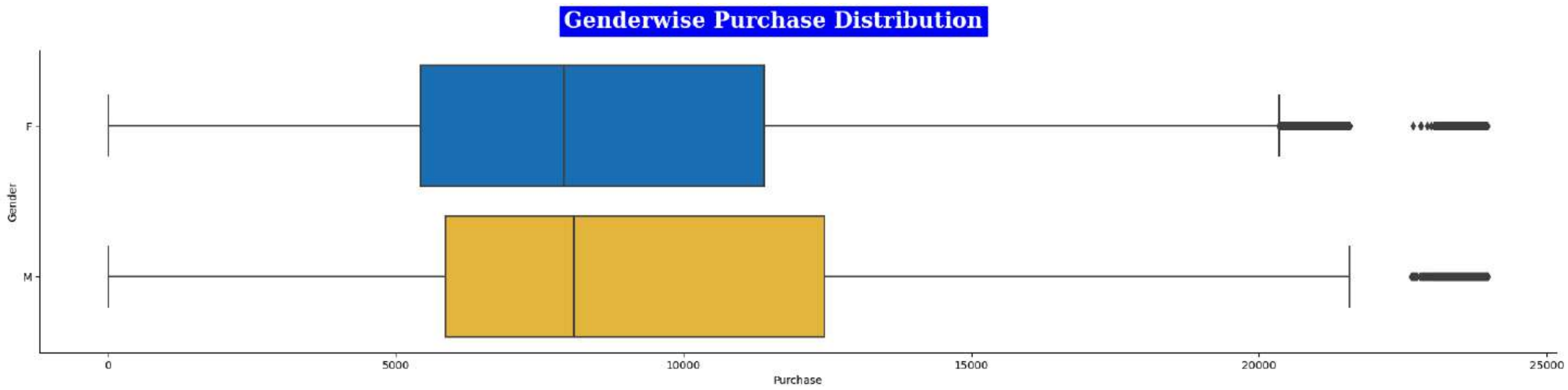
```
plt.style.use('seaborn-v0_8-bright')
plt.figure(figsize=(14,0.05))
plt.axis('off')
plt.title('Genderwise Purchase Distribution according to City_Category',fontfamily='serif',fontweight='bold',
          fontsize=14,backgroundcolor='b',color='w')
sns.catplot(x='Gender',y='Age',col='City_Category',hue='Marital_Status',data=wm,kind='strip',palette=cp2,marker='o')
plt.show()
```

Genderwise Purchase Distribution according to City_Category



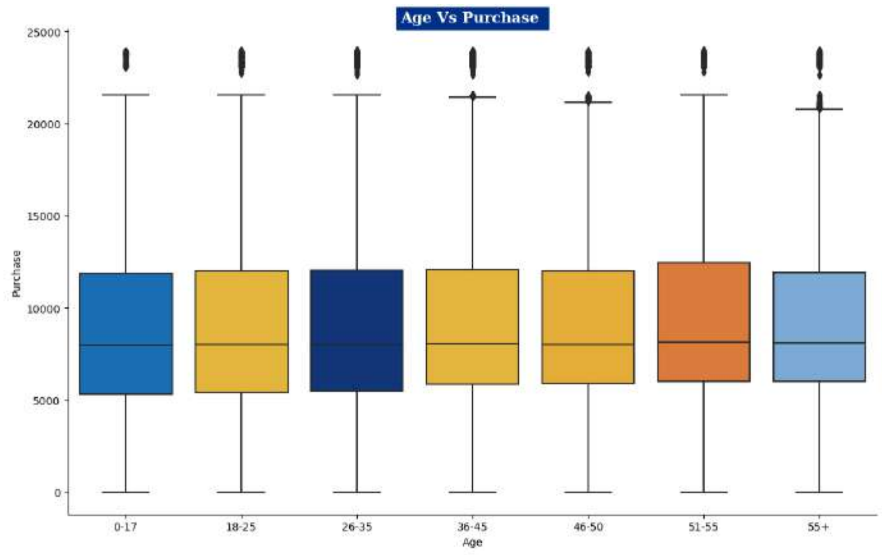
In [223…

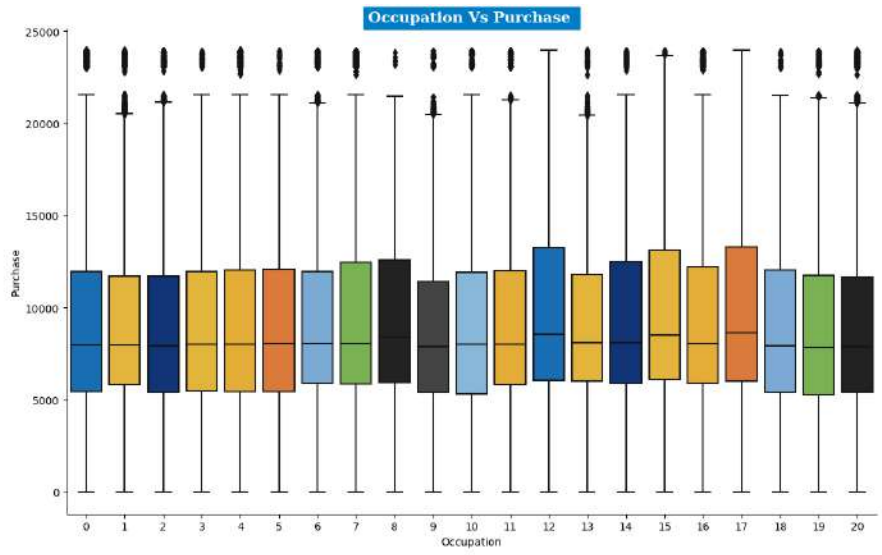
```
plt.figure(figsize=(25,5))
plt.suptitle("Genderwise Purchase Distribution",fontsize=20,fontfamily='serif',fontweight='bold'
             ,backgroundcolor='b',color='w')
plt.style.use('seaborn-v0_8-bright')
sns.boxplot(data = wm, y = "Gender", x = "Purchase" , palette=cp1)
sns.despine()
plt.show()
```

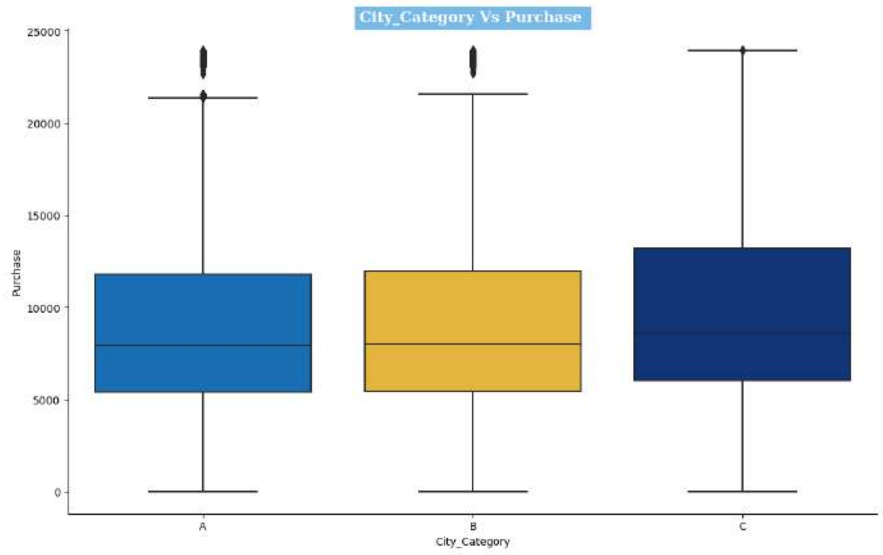


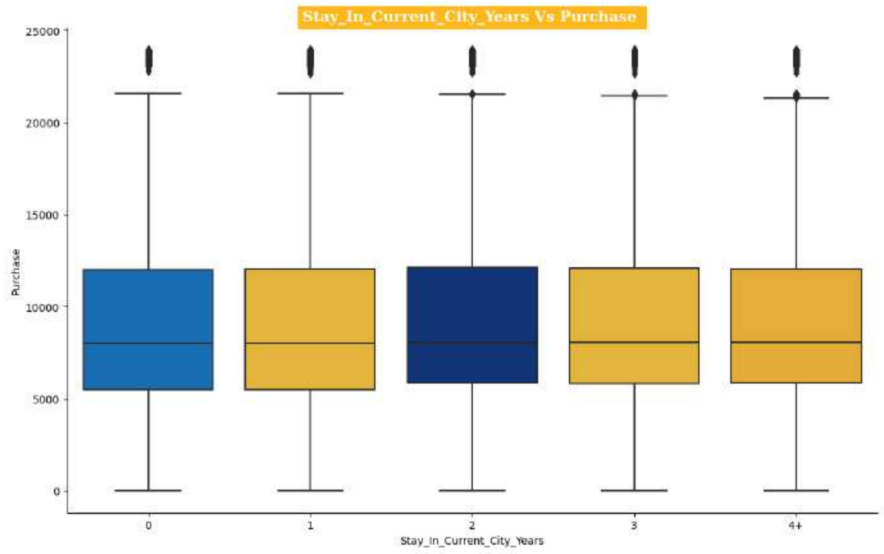
In [235…

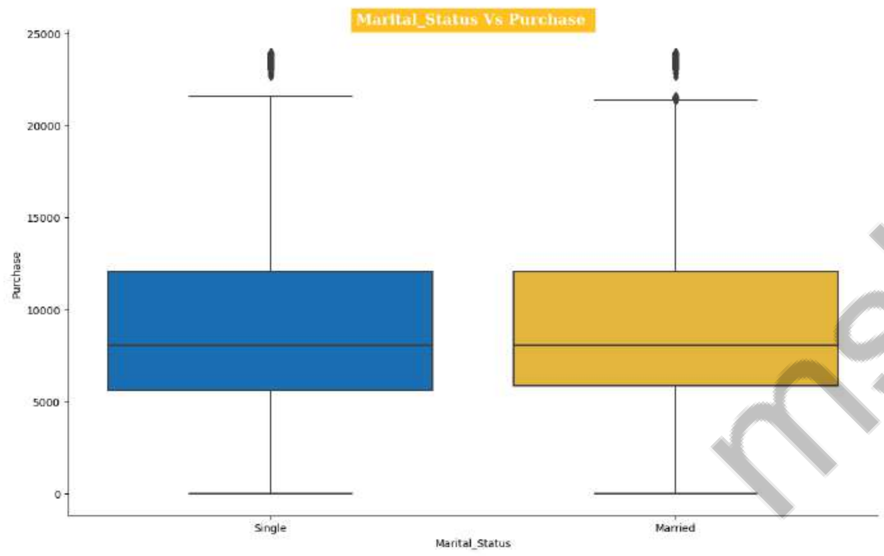
```
cat_cols = ['Age','Occupation','City_Category','Stay_In_Current_City_Years','Marital_Status', 'Product_Category']
plt.figure(figsize = (30,28))
plt.style.use('seaborn-v0_8-bright')

for i in range(len(cat_cols)):
    plt.subplot(3,2,i+1)
    sns.boxplot(data = wm, x = cat_cols[i], y='Purchase', palette = cp3)
    sns.despine()
    plt.title(f'{cat_cols[i]} Vs Purchase ',fontsize=14,fontfamily='serif',fontweight='bold',backgroundcolor=cp[i],color='w')
```

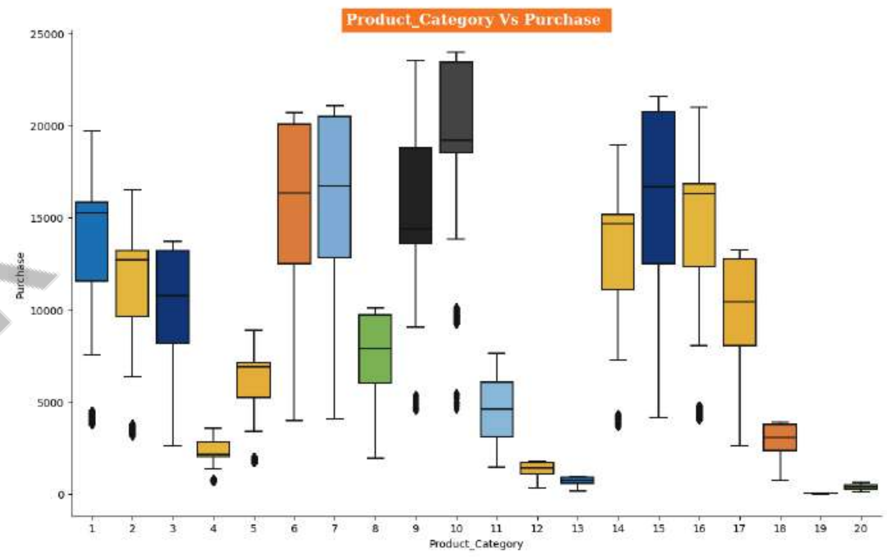












🧐 Observation

- Out of all the variables analysed above, it's noteworthy that the purchase amount remains `relatively stable` regardless of the variable under consideration. As indicated in the data, the `median purchase amount` consistently hovers around `8,000 USD`, regardless of the specific variable being examined.

🚀 Q. Top 10 products and product_category based on Black Friday sales :

In [104…

```
tspc = wm.groupby(['Product_Category']).agg(cnt=('User_ID','count'))[:10]
tspc = tspc.sort_values(by='cnt',ascending=False).reset_index()
tspc
```

Out[104]:

	Product_Category	cnt
0	5	150933
1	1	140378
2	8	113925
3	2	23864
4	6	20466
5	3	20213
6	4	11753
7	10	5125
8	7	3721
9	9	410

In [120…

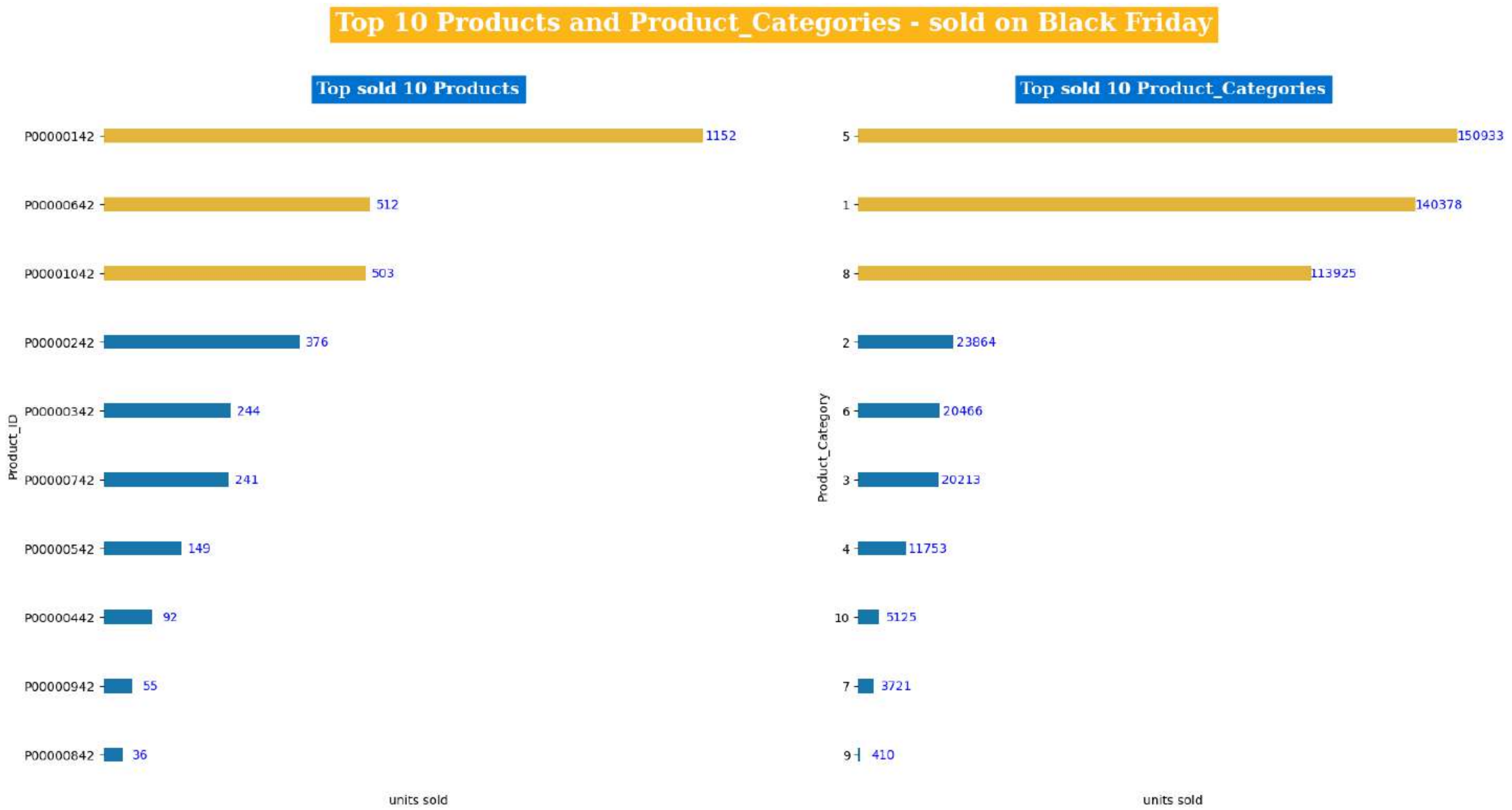
```
tsp = wm.groupby(['Product_ID']).agg(cnt=('User_ID','count'))[:10]
tsp = tsp.sort_values(by='cnt',ascending=False).reset_index()
tsp
```

Out[120]:

	Product_ID	cnt
0	P00000142	1152
1	P00000642	512
2	P00001042	503
3	P00000242	376
4	P00000342	244
5	P00000742	241
6	P00000542	149
7	P00000442	92
8	P00000942	55
9	P00000842	36

In [171…

```
plt.figure(figsize=(20,10))
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')
plt.suptitle('Top 10 Products and Product_Categories - sold on Black Friday',fontfamily='serif',fontweight='bold',fontsize=20,
             backgroundcolor=cp2[1],color='w')

color_map =  ['#ffc120' for i in range(3)] + ["#007dc6" for i in range(7)]

plt.subplot(1,2,2)
sns.barplot(y=tspc['Product_Category'] , x=tspc['cnt'],order=tspc['Product_Category'],palette=color_map,width=0.2)
#sns.scatterplot(y=tspc['Product_Category'],x=tspc['cnt'],palette=color_map,s=120)
sns.despine(left=True,bottom=True,trim=True)
plt.title('Top sold 10 Product_Categories',fontfamily='serif',fontweight='bold',fontsize=14,backgroundcolor=cp2[0],color='w')
plt.xlabel('units sold')
plt.xticks([])
n=10
for i in range(n):
    plt.annotate(tspc.cnt[i], (tspc.cnt[i]+5700,i+0.1),ha='center' , va='bottom' , color='b')

plt.subplot(1,2,1)
sns.barplot(y=tsp['Product_ID'] , x=tsp['cnt'],order=tsp['Product_ID'],palette=color_map,width=0.2)
#sns.scatterplot(y=tsp['Product_ID'] , x=tsp['cnt'],palette=color_map,s=120)
sns.despine(left=True,bottom=True,trim=True)
plt.title('Top sold 10 Products',fontfamily='serif',fontweight='bold',fontsize=14,backgroundcolor=cp2[0],color='w')
plt.xlabel('units sold')
plt.xticks([])

n=10
for i in range(n):
    plt.annotate(tsp.cnt[i], (tsp.cnt[i]+35,i+0.1),ha='center' , va='bottom' , color='b')

plt.show()
```

🏷️ Insights

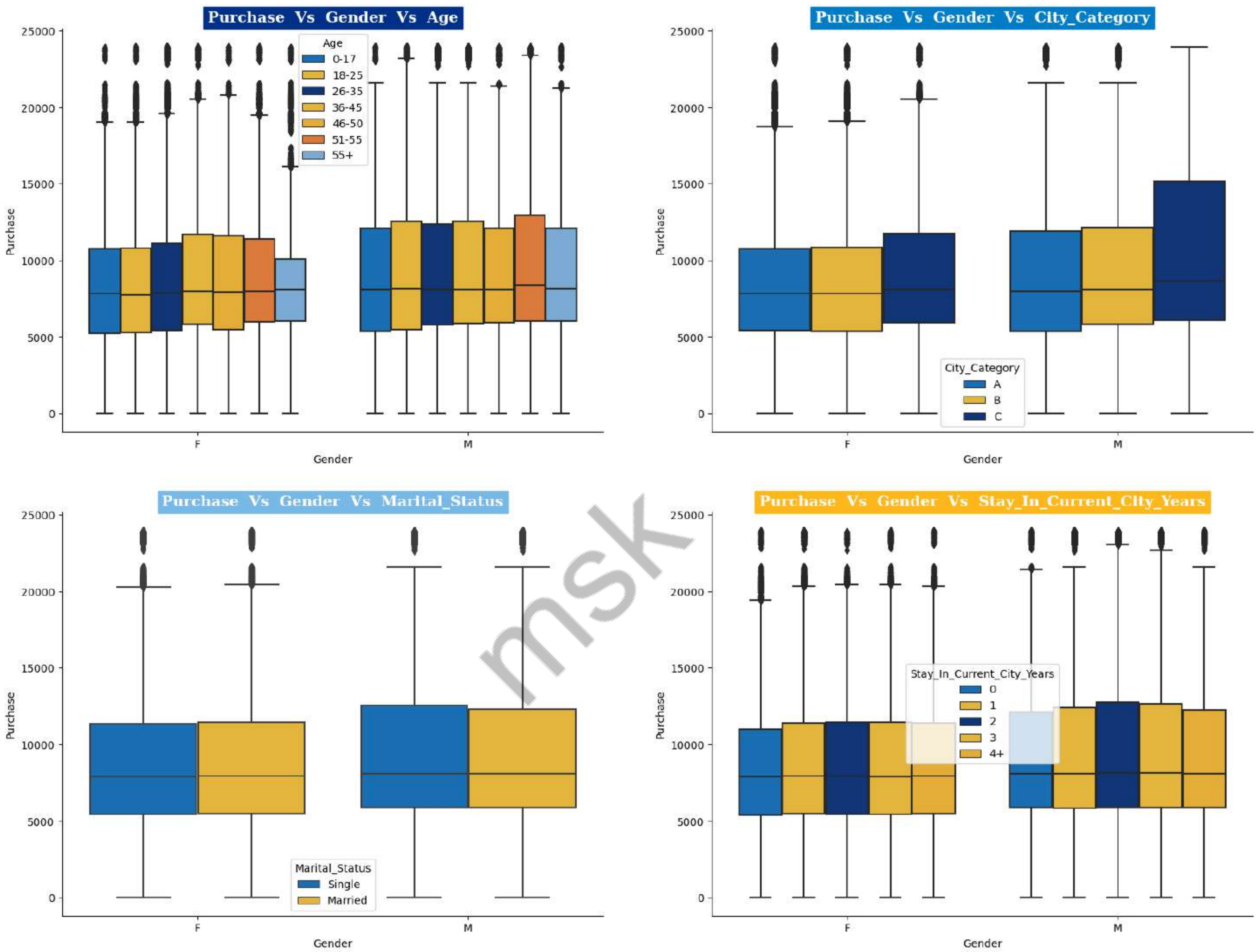
- **Top 10 Products Sold**
    - The top-selling products during Walmart's Black Friday sales are characterized by a `relatively small variation` in sales numbers, suggesting that Walmart offers a variety of products that many different customers like to buy.

- **Top 10 Product Categories**
  - Categories `5,1 and 8` have significantly outperformed other categories with combined Sales of nearly `75%` of the total sales suggesting a strong preference for these products among customers.

Multivariate Anlaysis

In [180…

```
cat_cols = ['Age','City_Category','Marital_Status','Stay_In_Current_City_Years']

plt.figure(figsize = (20,15))
plt.style.use('seaborn-v0_8-bright')
for i in range(len(cat_cols)):
    plt.subplot(2,2,i+1)
    sns.boxplot(data = wm, x = 'Gender', y= 'Purchase',hue = cat_cols[i], palette = cp3)
    sns.despine()
    plt.title(f'Purchase  Vs  Gender  Vs  {cat_cols[i]}', fontsize = 14,fontfamily='serif',fontweight='bold'
              ,backgroundcolor=cp[i],color='w')
```



In [182…

```
wm.columns
```

Out[182]:

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

In [187…

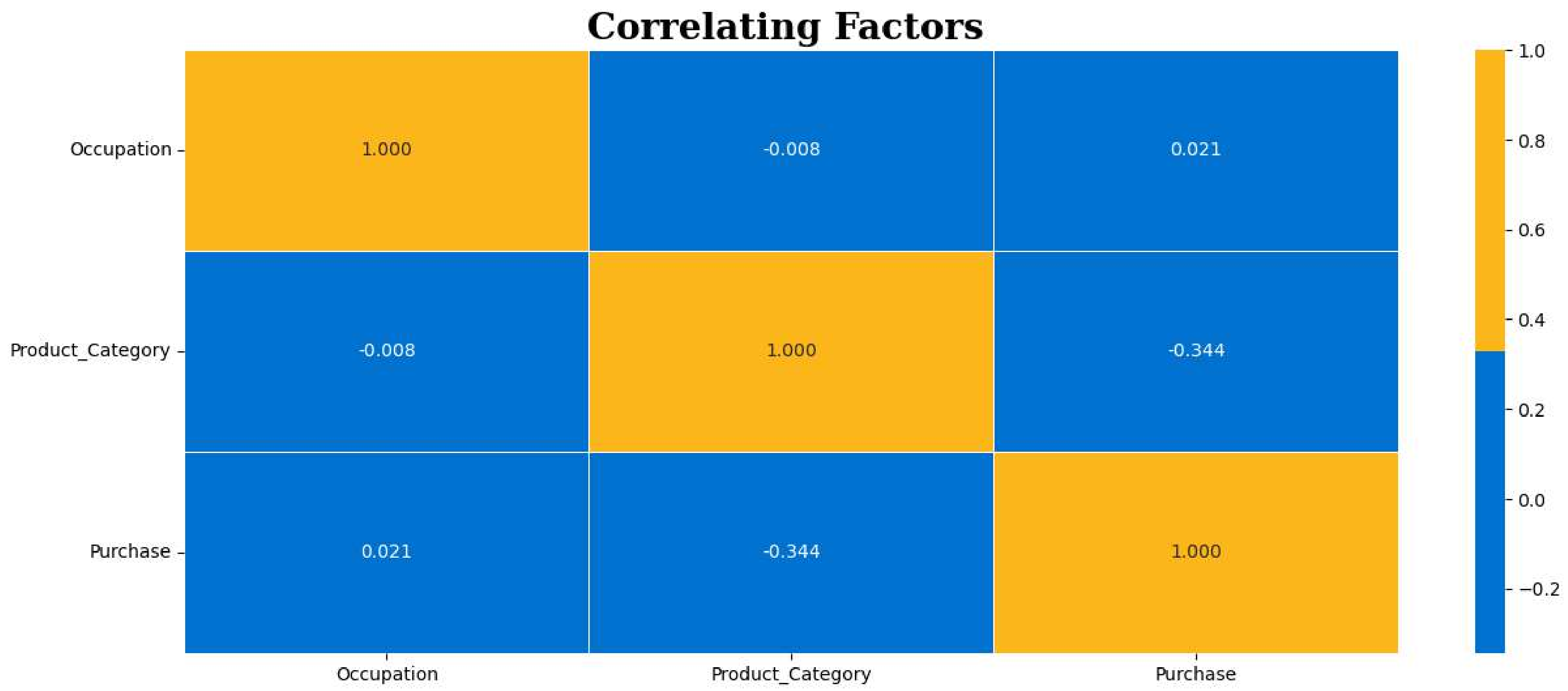
```
wmcorr = wm[['Occupation', 'Product_Category','Purchase']].corr()
wmcorr
```

Out[187]:

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

In [192…

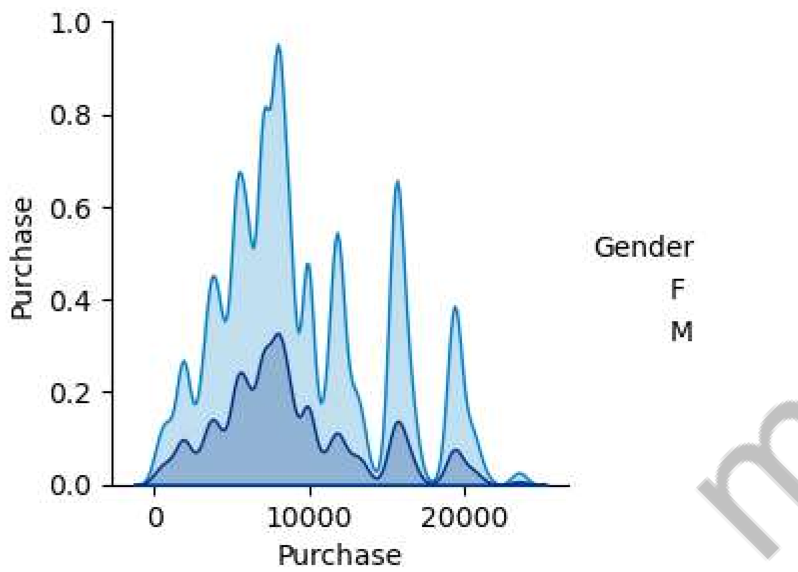
```
#Correlation HeatMap
plt.figure(figsize=(15,6))
ax = sns.heatmap(wmcorr,annot=True,fmt='.3f',linewidths=.5,cmap=cp2)
plt.title('Correlating Factors ',fontfamily='serif',fontweight='bold',fontsize=20)
plt.yticks(rotation=0)
plt.show()
```

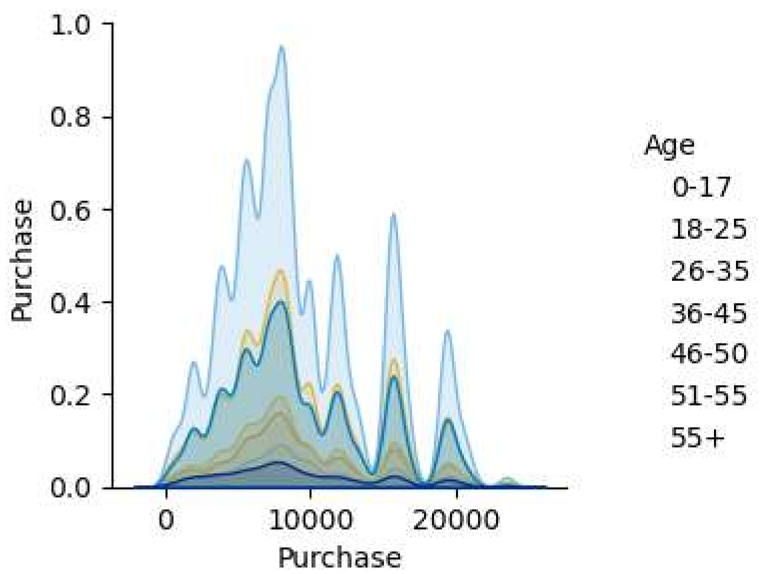


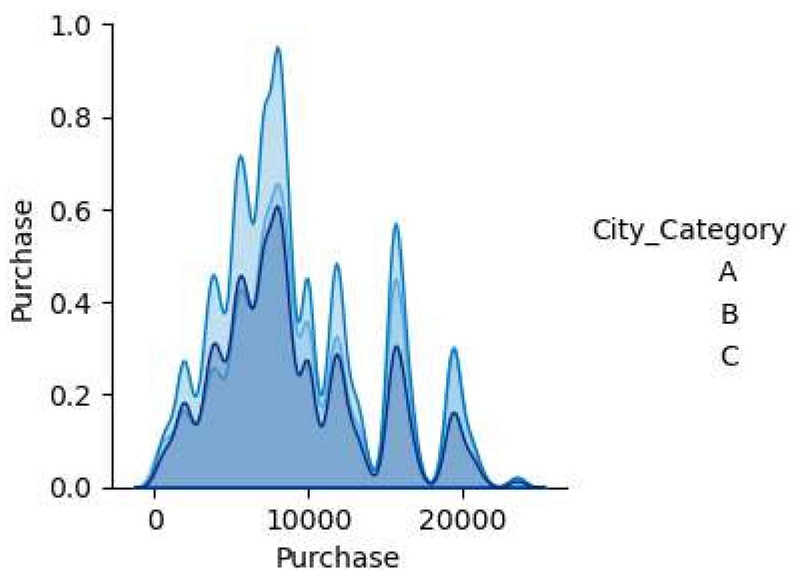
In [213…

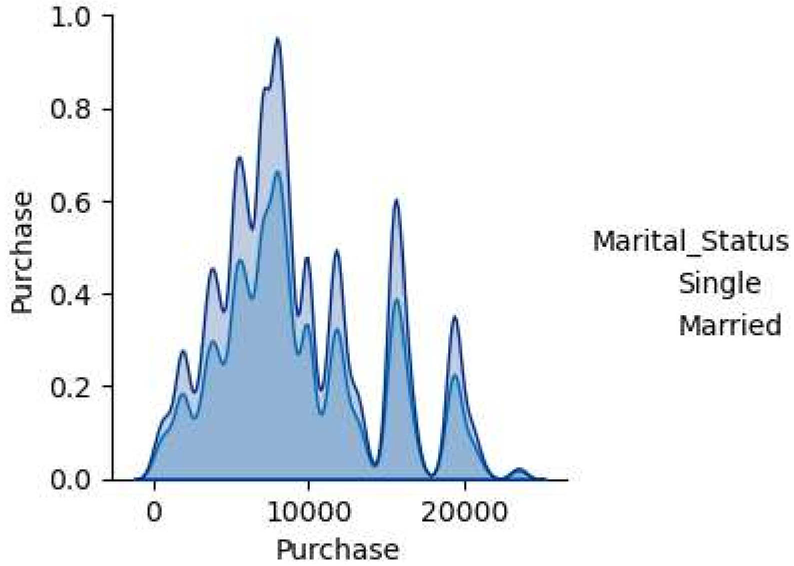
```
# Pairplot Analysis
cat_cols = ['Gender','Age','City_Category','Marital_Status','Stay_In_Current_City_Years']
plt.suptitle(f'Pairplot based on Purchase',fontfamily='serif',fontweight='bold',fontsize=12,backgroundcolor=cp[0],color='w')
for i in range(len(cat_cols)):
    sns.pairplot(wm,y_vars = "Purchase",hue=cat_cols[i],kind='reg',palette=cp,height=3)
    plt.show()
```

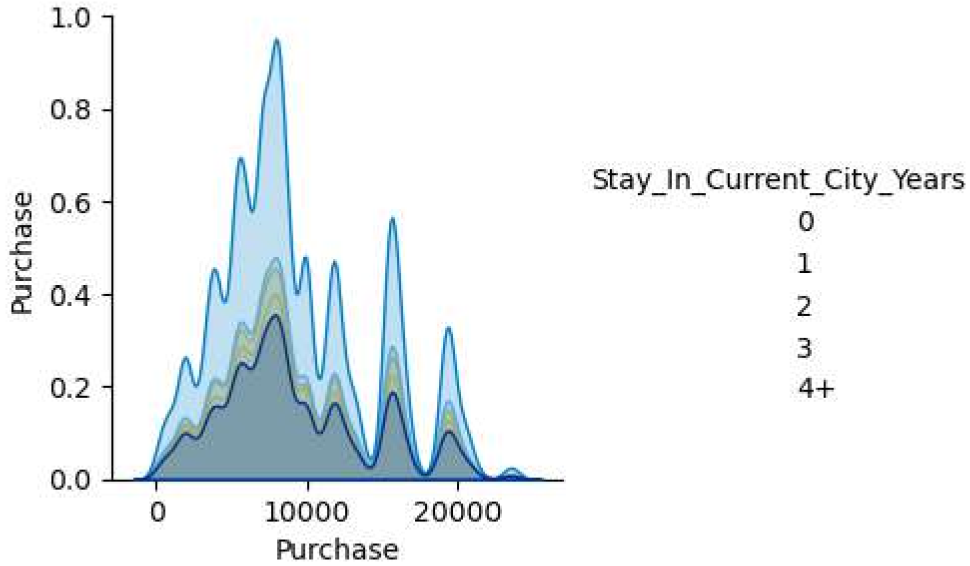
```
<Figure size 640x480 with 0 Axes>
```







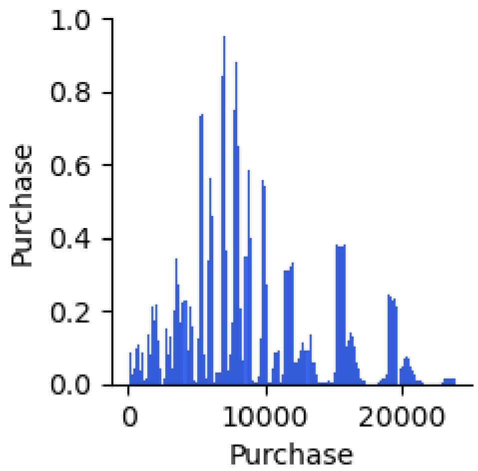




In [214…

```
sns.pairplot(wm)
```

Out[214]: <seaborn.axisgrid.PairGrid at 0x1c8cd36fe50>



🧨 Q.Difference between the mean and median value of the purchase amount:

In [215…

```
avg_purchase = wm.Purchase.mean()
avg_purchase
```

Out[215]: 9263.968712959126

In [216…

```
median_purchase = wm.Purchase.median()
median_purchase
```

Out[216]: 8047.0

In [217…

```
Difference = avg_purchase - median_purchase
Difference
```

Out[217]: 1216.9687129591257

🧨 Q. Tracking the amount spent per transaction of all the 50 million female customers,and all the 50 million male customers,calculate the average, and conclude the results.

In [218…

```
avg_purchase_amt = wm.groupby('Gender')[['Purchase']].mean().reset_index().round(2)
avg_purchase_amt
```

Out[218]:

	Gender	Purchase
0	F	8734.57
1	M	9437.53

Insights

- The Difference between the Mean and Median value of the numerical column `Purchase_amt` is found to be **1216.97**.
- The Average amount spent by **Male Customer** is `9437.53` which is substancially higher than **Female Customers** is `8734.57`.

In [219…
```
wm_male = wm[wm['Gender']=='M']
wm_female = wm[wm['Gender']=='F']
```

In [220…
```
len(wm_male)
```

Out[220]: 414259

In [222…
```
len(wm_female)
```

Out[222]: 135809

In [229…
```
# total purchase split by gender
psg = wm.groupby(['User_ID','Gender'])[['Purchase']].sum()
tpsg = psg[psg['Purchase']!=0]
tpsg.reset_index(inplace=True)
tpsg
```

Out[229]:

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

5891 rows × 3 columns

In [232…
```
tpsg.Gender.value_counts().to_frame()
```

Out[232]:

Gender	count
M	4225
F	1666

In [236…
```
tpsg.groupby('Gender').agg(avg_purchase_amt=('Purchase','mean')).round(2)
```

Out[236]:

Gender	avg_purchase_amt
F	712024.39
M	925344.40

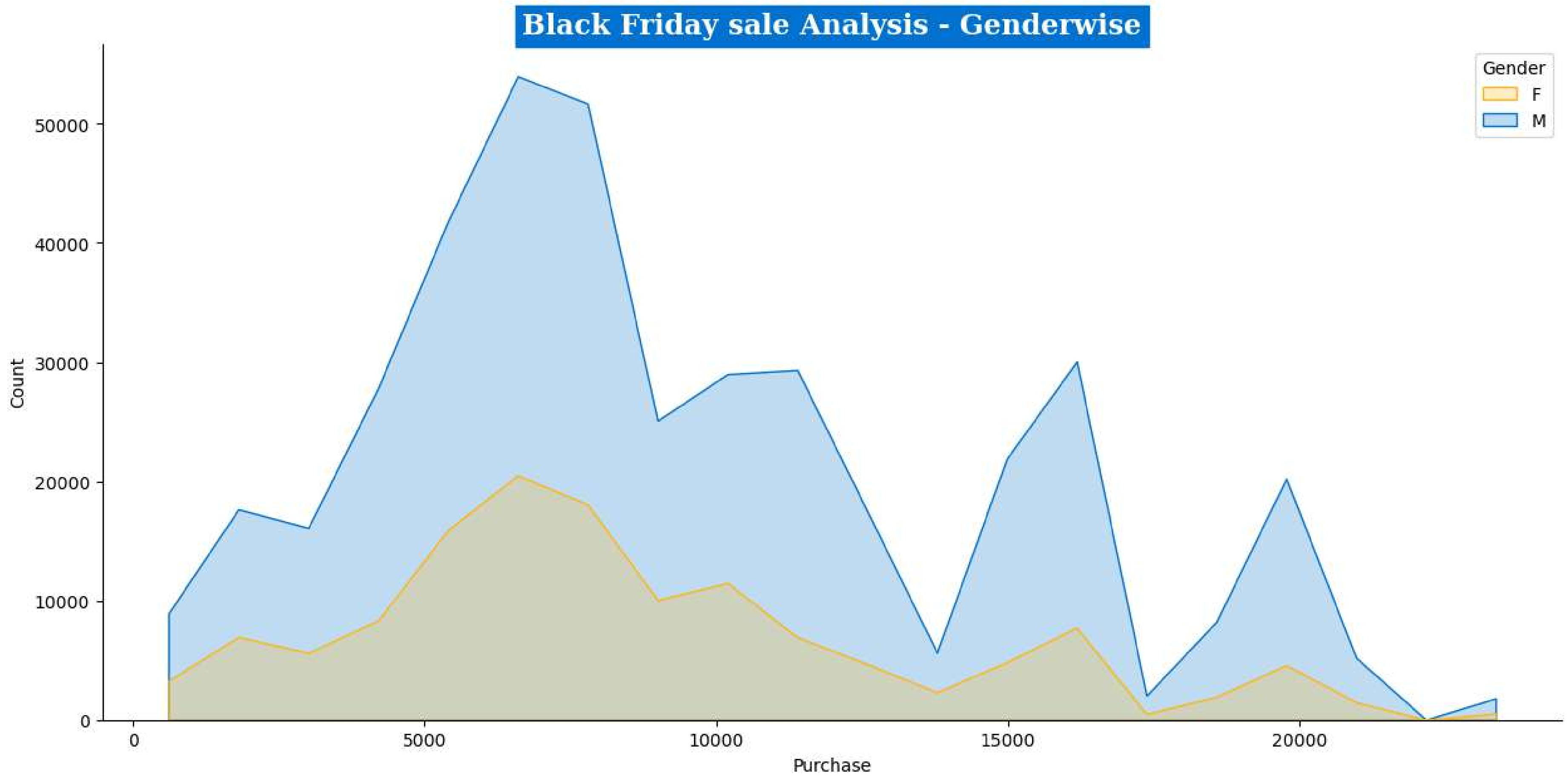
In [275…
```
wm.groupby('Gender')['Purchase'].describe().T
```

Out[275]:

Gender	F	M
count	135809.000000	414259.00000
mean	8734.565765	9437.52604
std	4767.233289	5092.18621
min	12.000000	12.00000
25%	5433.000000	5863.00000
50%	7914.000000	8098.00000
75%	11400.000000	12454.00000
max	23959.000000	23961.00000

In [392…

```
plt.figure(figsize=(15,7))
sns.histplot(data=wm, x = "Purchase", bins=20, hue = "Gender",element='poly',palette=cp2[::-1])
sns.despine()
plt.title('Black Friday sale Analysis - Genderwise',fontfamily='serif',fontweight='bold',fontsize=16,backgroundcolor=cp2[0],color
plt.show()
```



In [252…

```
male_purchase = tpsg[tpsg['Gender']=='M']['Purchase']
male_purchase
```

Out[252]:

```
1        810472
2        341635
3        206468
4        821001
6        234668
          ...
5880     737361
5882     517261
5883     501843
5884     197086
5890    1653299
Name: Purchase, Length: 4225, dtype: int64
```


In [253…

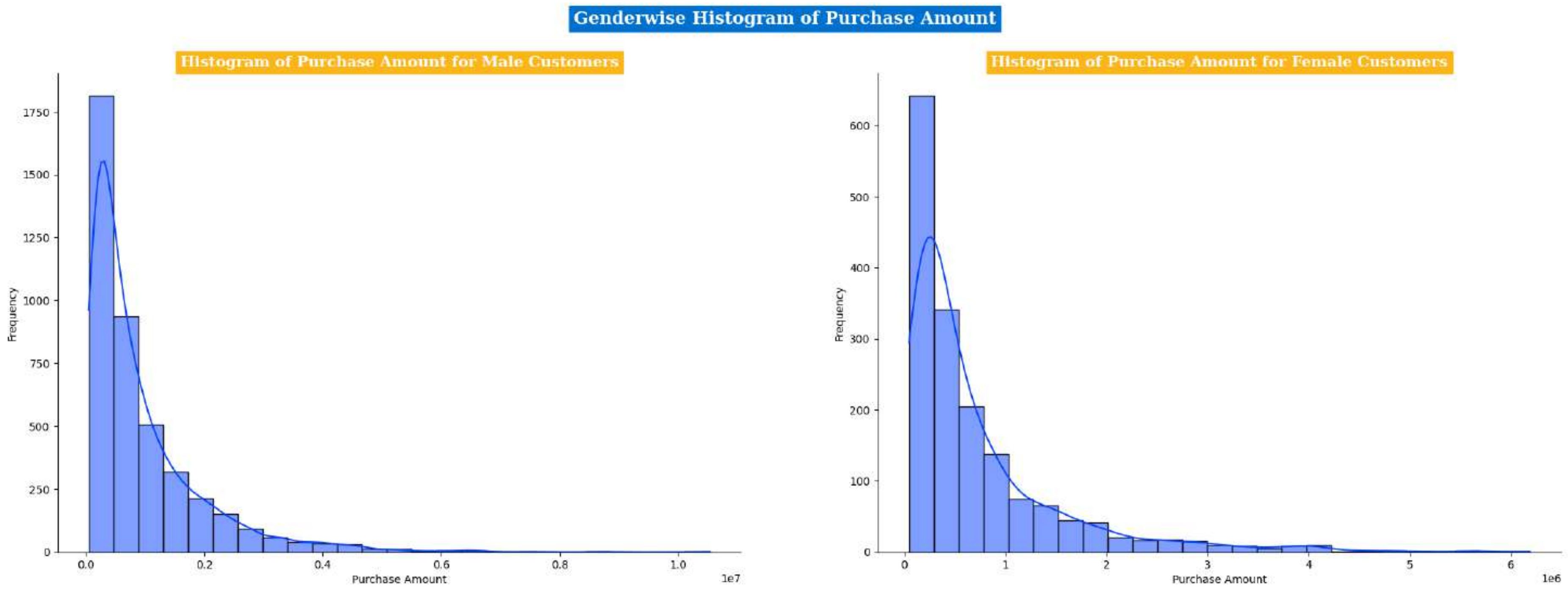
```
female_purchase = tpsg[tpsg['Gender']=='F']['Purchase']
female_purchase
```

Out[253]:

```
0        334093
5        379930
9       2169510
10       557023
15       150490
          ...
5885     956645
5886    4116058
5887    1119538
5888      90034
5889     590319
Name: Purchase, Length: 1666, dtype: int64
```

In [254…

```
plt.figure(figsize=(25,8))
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')
plt.suptitle('Genderwise Histogram of Purchase Amount',fontfamily='serif',fontweight='bold',fontsize=16,
              backgroundcolor=cp2[0],color='w')

plt.subplot(1,2,1)
sns.histplot(male_purchase, bins=25,kde=True)
plt.title("Histogram of Purchase Amount for Male Customers",fontfamily='serif',fontweight='bold',fontsize=14,
              backgroundcolor=cp2[1],color='w')
plt.xlabel("Purchase Amount")
plt.ylabel("Frequency")

plt.subplot(1,2,2)
sns.histplot(female_purchase, bins=25,kde=True)
plt.title("Histogram of Purchase Amount for Female Customers",fontfamily='serif',fontweight='bold',fontsize=14,
              backgroundcolor=cp2[1],color='w')
sns.despine()
plt.xlabel("Purchase Amount")
plt.ylabel("Frequency")
plt.show()
```



🏷️ Insights:

- **Male customers spent more money than female customers** during the `Black Friday` sale.
    - The number of total **Males** (`4225`) is ***greater*** than number of total **Females** (`1666`).
    - Average amount spend by **Male customers** (`925344.40`) is ***greater*** than Average amount spend by **Female customers**(`712024.39`).
    - There are more male customers than female customers, hence the male customers will tend to buy more than female customers.
    - There could be more products suited to males than the female products which could lead to increase in sales of the products bought by men.

🔀 This data seems like lognormal distributed curve and also Right skewed. lets try boxcox to get the normal distribution.

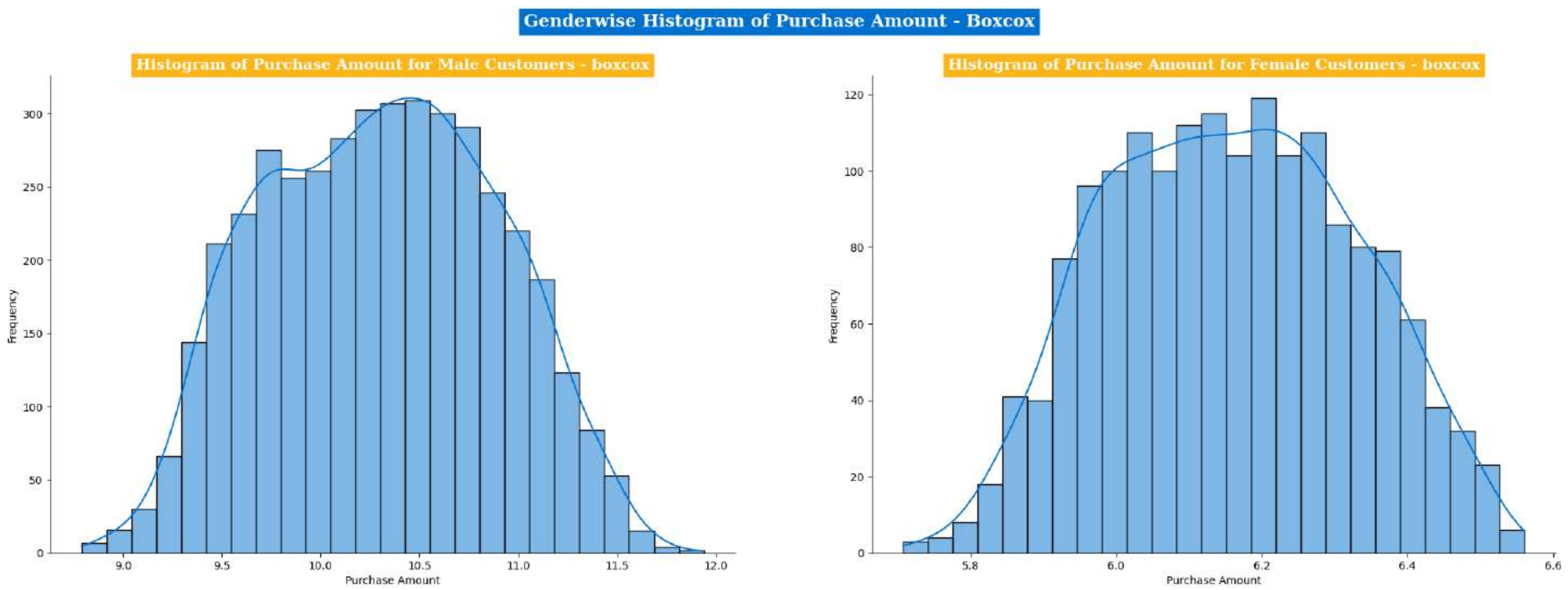
In [256…

```
transformed_data_male_purchase , m_best_lambda = boxcox(male_purchase)
print(f"Best lambda for male purchase = {m_best_lambda}")
transformed_data_female_purchase , f_best_lambda = boxcox(female_purchase)
print(f"Best lambda for female purchase = {f_best_lambda}")
```

```
Best lambda for male purchase = -0.0396131636344579
Best lambda for female purchase = -0.13352643386543317
```


In [260…

```
plt.figure(figsize=(25,8))
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')
plt.suptitle('Genderwise Histogram of Purchase Amount - Boxcox',fontfamily='serif',fontweight='bold',fontsize=16,
              backgroundcolor=cp2[0],color='w')

plt.subplot(1,2,1)
sns.histplot(transformed_data_male_purchase, bins=25,kde=True,color=cp2[0],cbar=True)
plt.title("Histogram of Purchase Amount for Male Customers - boxcox",fontfamily='serif',fontweight='bold',fontsize=14,
              backgroundcolor=cp2[1],color='w')
plt.xlabel("Purchase Amount")
plt.ylabel("Frequency")

plt.subplot(1,2,2)
sns.histplot(transformed_data_female_purchase, bins=25,kde=True,color=cp2[0])
plt.title("Histogram of Purchase Amount for Female Customers - boxcox",fontfamily='serif',fontweight='bold',fontsize=14,
              backgroundcolor=cp2[1],color='w')
sns.despine()
plt.xlabel("Purchase Amount")
plt.ylabel("Frequency")plt.show()
```



In [273…

```
n_female = len(female_purchase)
mean_female = female_purchase.mean()

population_std_dev = female_purchase.std()

#confidence level (95%)
confidence_level = 0.95

z_stat, p_value = ztest(female_purchase, value=population_std_dev)


lower_bound = mean_female - (z_stat * (population_std_dev / (n_female**0.5)))
upper_bound = mean_female + (z_stat * (population_std_dev / (n_female**0.5)))

print(f"Female Sample Mean: {mean_female}")
print(f"Z-statistic: {z_stat}")
print(f"P-value: {p_value}")
print(f"Confidence Interval: ({lower_bound}, {upper_bound})")
```

```
Female Sample Mean: 712024.3949579832
Z-statistic: -4.820238048690314
P-value: 1.4338702083630263e-06
Confidence Interval: (807370.7261464577, 616678.0637695086)
```

In [272…

```
n_male = len(male_purchase)
mean_male = male_purchase.mean()

population_std_dev = male_purchase.std()

#confidence level (95%)
confidence_level = 0.95

z_stat, p_value = ztest(male_purchase, value=population_std_dev)

lower_bound = mean_male - (z_stat * (population_std_dev / (n_female**0.5)))
upper_bound = mean_male + (z_stat * (population_std_dev / (n_female**0.5)))

print(f"Male Sample Mean: {mean_male}")
print(f"Z-statistic: {z_stat}")
print(f"P-value: {p_value}")
print(f"Confidence Interval: ({lower_bound}, {upper_bound})")
```

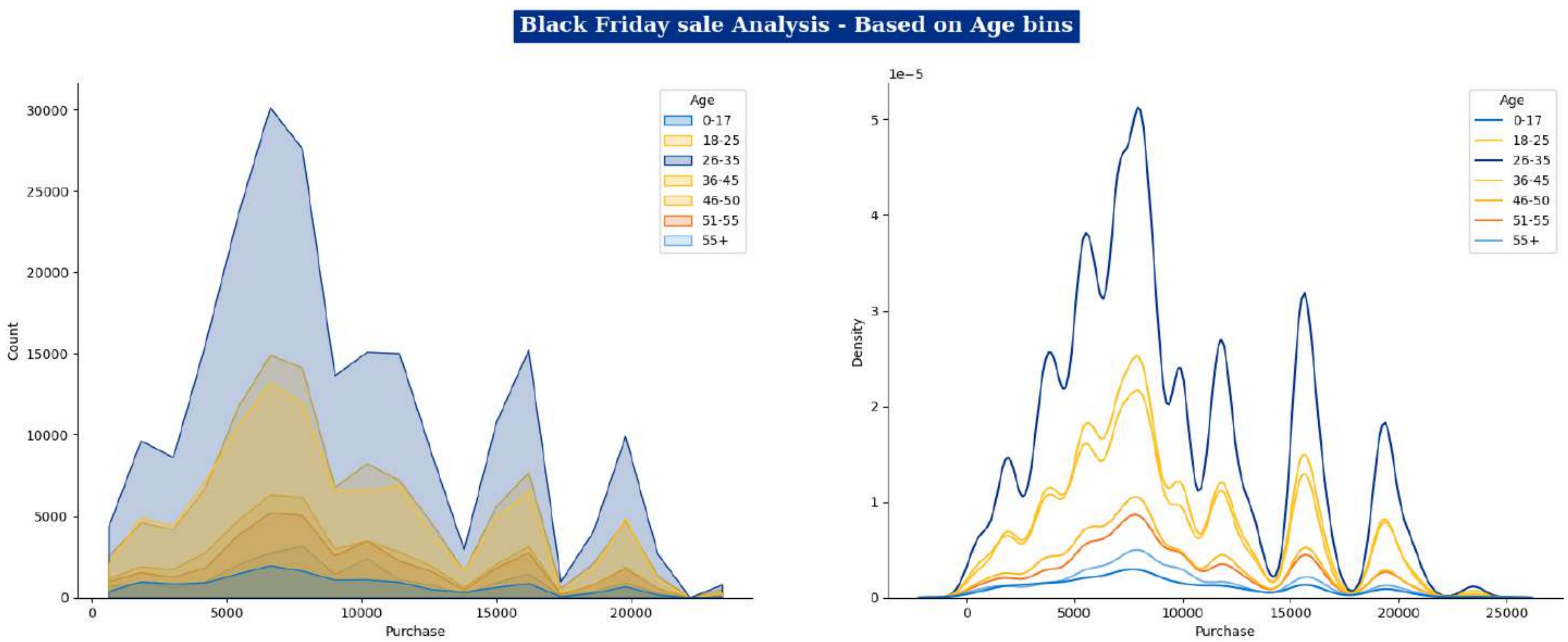
```
Male Sample Mean: 925344.4023668639
Z-statistic: -3.9880811051336282
P-value: 6.660989296222788e-05
Confidence Interval: (1021667.0824554558, 829021.722278272)
```


👢 Bootstrapping & Central Limit Theorem

▶️ 〰️ ◀️ Confidence intervals and distribution of the mean of the expenses based on customers Age

In [419…

```
plt.figure(figsize=(20,7))
plt.suptitle('Black Friday sale Analysis - Based on Age bins',fontfamily='serif',fontweight='bold',
          fontsize=16,backgroundcolor=cp[0],color='w')

plt.subplot(1,2,1)
sns.histplot(data=wm, x = "Purchase", bins=20, hue = "Age",element='poly',palette=cp3)

plt.subplot(1,2,2)
sns.kdeplot(data=wm, x = "Purchase", hue = "Age",palette=cp3)
sns.despine()
plt.show()
```



In [505…

```
def bootstrapping_age(age_grp,data,sample_size,ntimes,ci):

    plt.figure(figsize=(25,6.5))
```

```
    plt.style.use('seaborn-v0_8-bright')
    plt.suptitle(f'Classfification of Customers based on Age',
             fontfamily='serif', fontweight='bold', fontsize=16, backgroundcolor='#ffb81c', color='w')

    data_sample_means = []
    for i in range(ntimes):
        dsm = np.mean(np.random.choice(data,sample_size))
        data_sample_means.append(dsm)

    ci = ci/100

    # data_sample_means parameters
    mean = np.mean(data_sample_means)
    sigma = np.std(data_sample_means)
    std_err_of_mean = stats.sem(data_sample_means)   # sem auto calculates the std.err for mean

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

    # plot1 # for mu = alt+230
    sns.kdeplot(data = data_sample_means, color=cp2[1], fill = True, linewidth = 2)
    label_mean = (f" µ :  {mean:.2f}")
    plt.axvline(mean, color = 'darkblue', linestyle = 'solid', linewidth = 2, label=label_mean)
    label_limits=(f"Lower Limit : {lower_limit:.2f}\nUpper Limit :  {upper_limit:.2f}")
    plt.axvline(lower_limit, color = 'goldenrod', linestyle = 'dashdot', linewidth = 2, label=label_limits)
    plt.axvline(upper_limit, color = 'goldenrod', linestyle = 'dashdot', linewidth = 2)
    sns.despine()

    plt.title(f"Sample Size: {sample_size} with Confidence Interval: {ci*100}%  for age_group- {age_grp}",
          fontfamily='serif', fontweight='bold', fontsize=14, backgroundcolor='darkblue', color='w')

    plt.legend()
    plt.xlabel('Purchase Amount')
    plt.ylabel('Probability Density')

    return data_sample_means , round(lower_limit,2), round(upper_limit,2), round(mean,2)
```

In [420…

```
wm.Age.nunique()
```

Out[420]:

```
7
```

In [421…

```
wm.Age.unique()
```

Out[421]:

```
['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+']
```


In [422…

```
wm.Age.value_counts()
```

Out[422]:

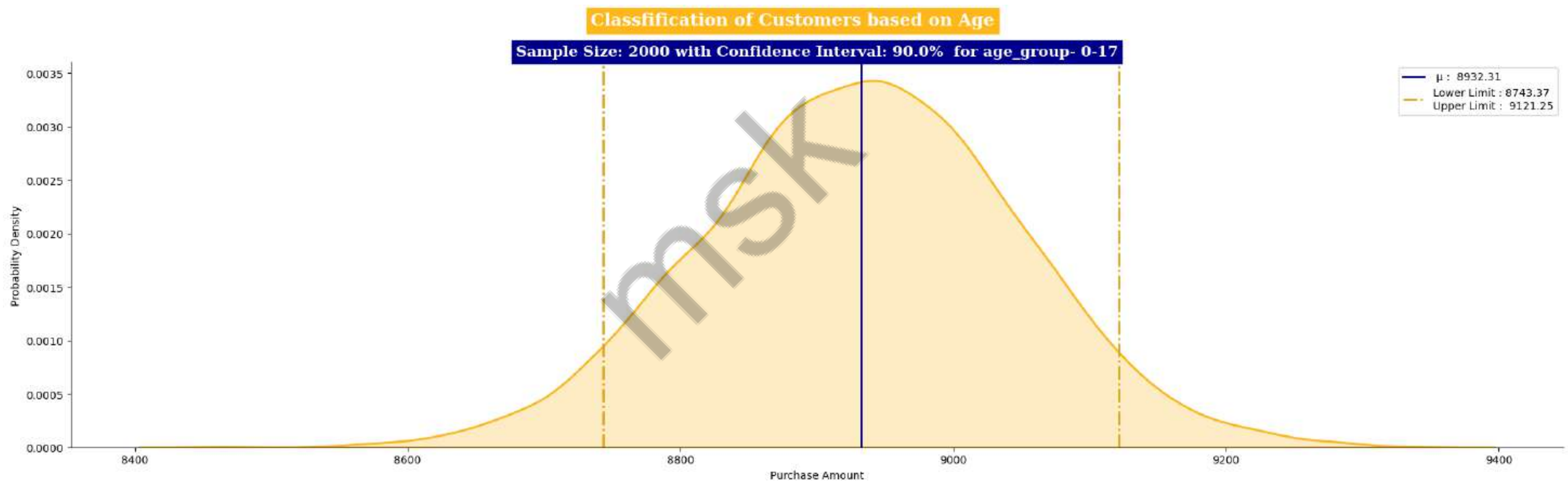
```
Age
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: count, dtype: int64
```

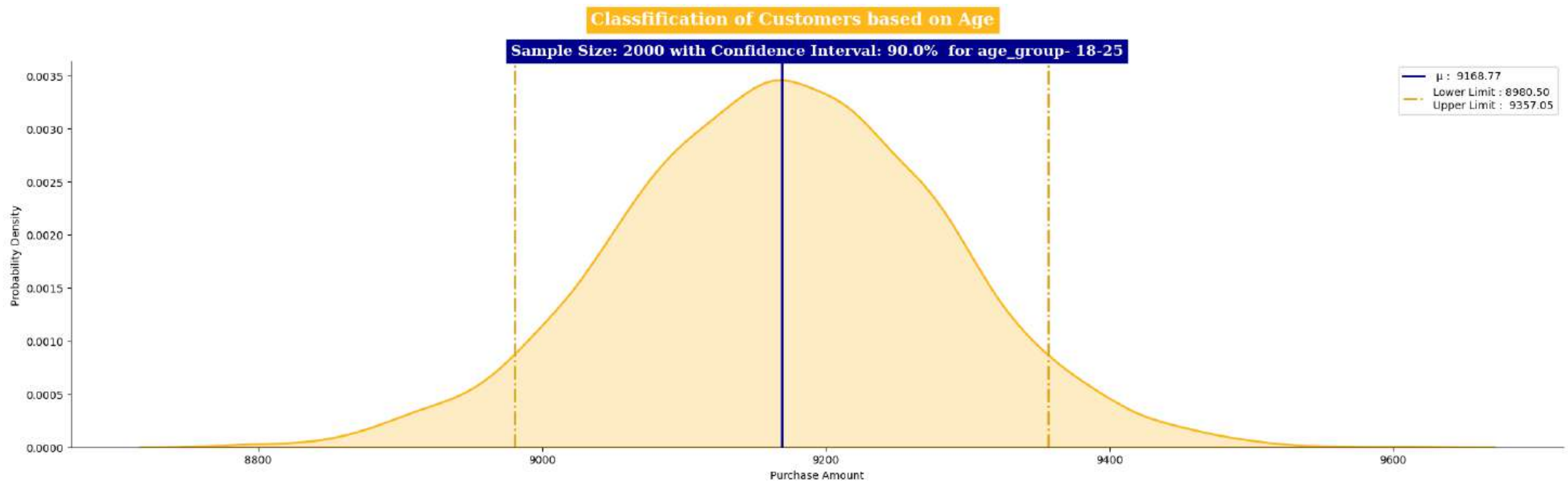
In [507…

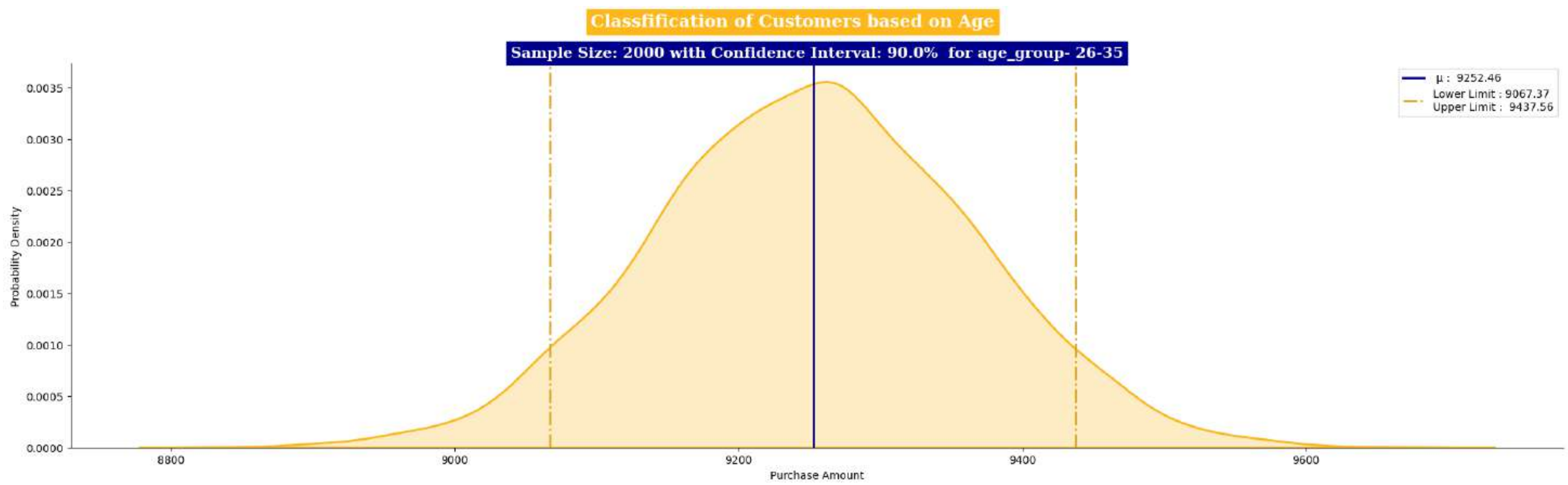
```
sample_size = 2000
ntimes = 5000
ci = [90,95,99]
age_group = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

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

for i in ci:
    for _ in age_group:
        age_data = wm[wm['Age']==_]['Purchase']
        avg, ll, ul, mean = bootstrapping_age( _ , age_data , sample_size , ntimes , i )
        df.loc[len(df.index)]=[ _ , sample_size , ll , ul , mean , i , (ul-ll) ]

df
```

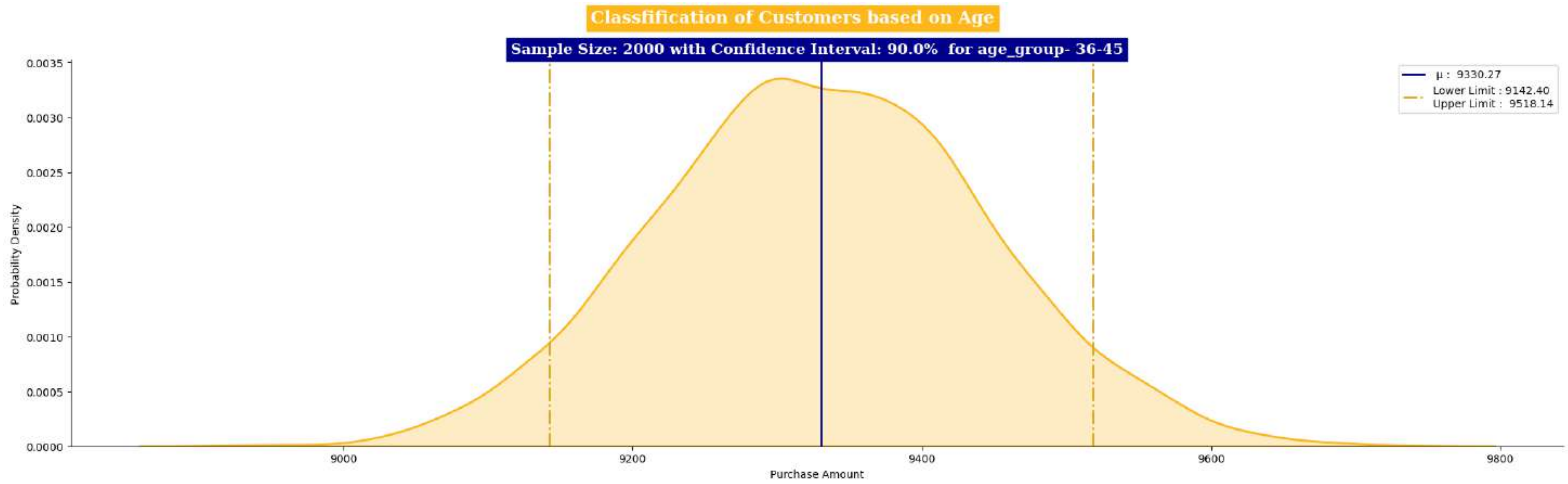
Out[507]:

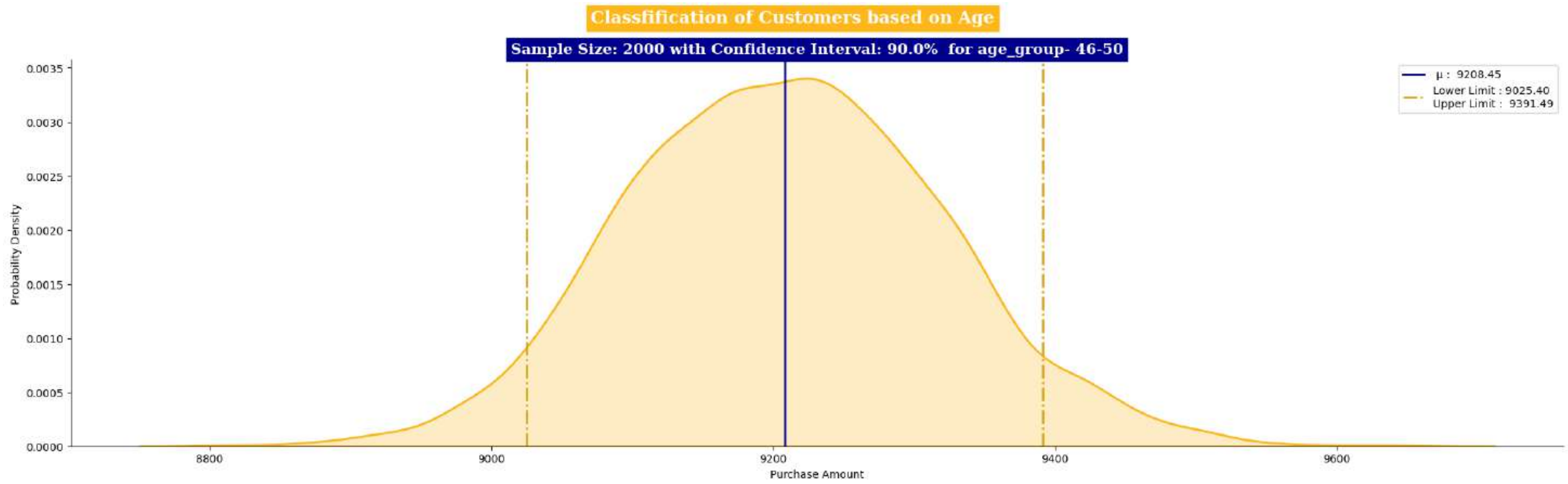
	Age_Group	Sample Size	Lower Limit	Upper Limit	Sample Mean	Confidence Interval	Range
0	0-17	2000	8743.37	9121.25	8932.31	90	377.88
1	18-25	2000	8980.50	9357.05	9168.77	90	376.55
2	26-35	2000	9067.37	9437.56	9252.46	90	370.19
3	36-45	2000	9142.40	9518.14	9330.27	90	375.74
4	46-50	2000	9025.40	9391.49	9208.45	90	366.09
5	51-55	2000	9346.19	9718.83	9532.51	90	372.64
6	55+	2000	9151.77	9521.66	9336.71	90	369.89
7	0-17	2000	8705.93	9157.01	8931.47	95	451.08
8	18-25	2000	8948.81	9388.77	9168.79	95	439.96
9	26-35	2000	9033.73	9472.08	9252.90	95	438.35
10	36-45	2000	9105.77	9557.49	9331.63	95	451.72
11	46-50	2000	8989.61	9426.84	9208.22	95	437.23
12	51-55	2000	9314.29	9753.25	9533.77	95	438.96
13	55+	2000	9113.96	9557.82	9335.89	95	443.86
14	0-17	2000	8643.01	9225.76	8934.38	99	582.75
15	18-25	2000	8875.18	9460.76	9167.97	99	585.58
16	26-35	2000	8962.56	9541.88	9252.22	99	579.32
17	36-45	2000	9041.82	9616.88	9329.35	99	575.06
18	46-50	2000	8923.16	9493.42	9208.29	99	570.26
19	51-55	2000	9236.80	9835.01	9535.91	99	598.21
20	55+	2000	9049.70	9626.28	9337.99	99	576.58

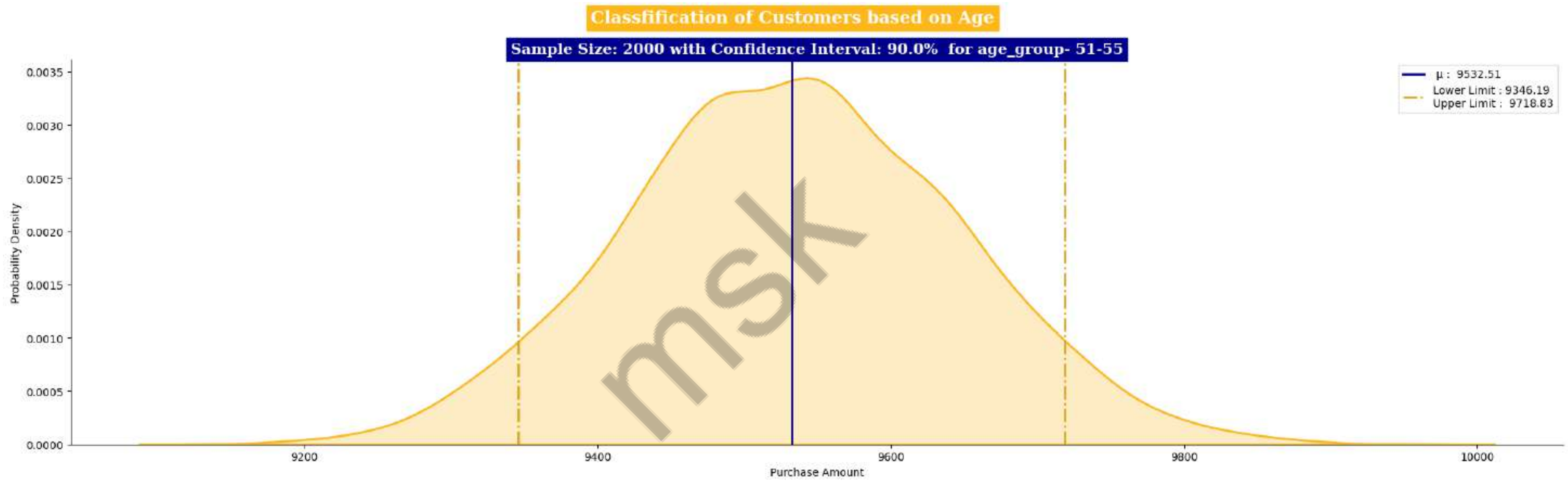


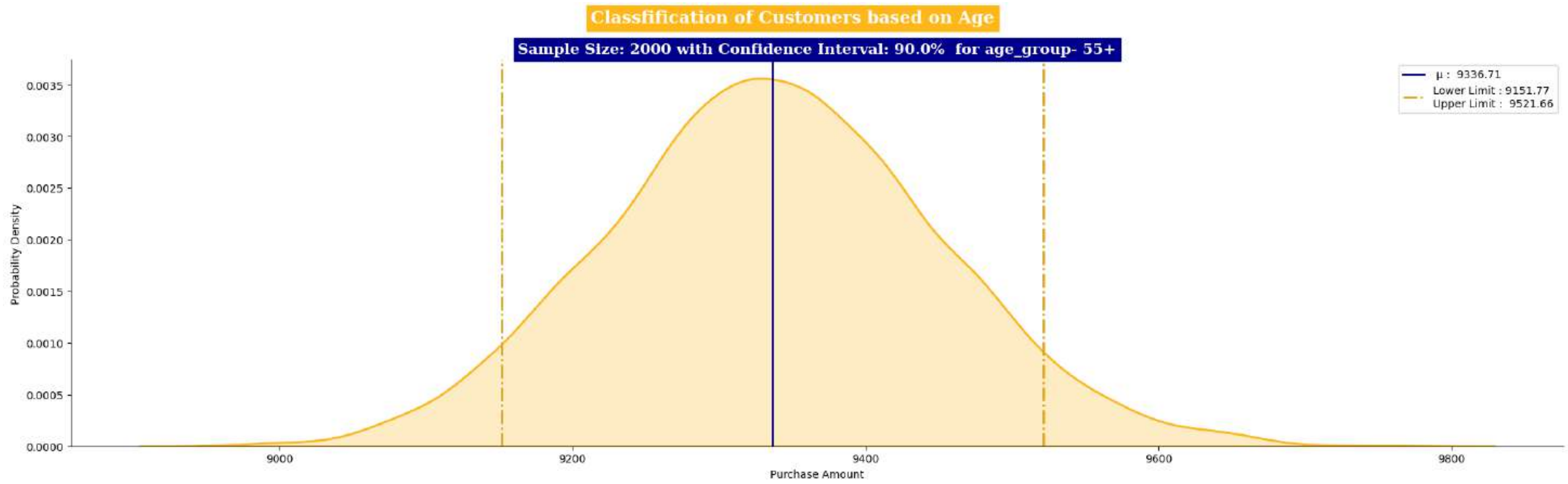


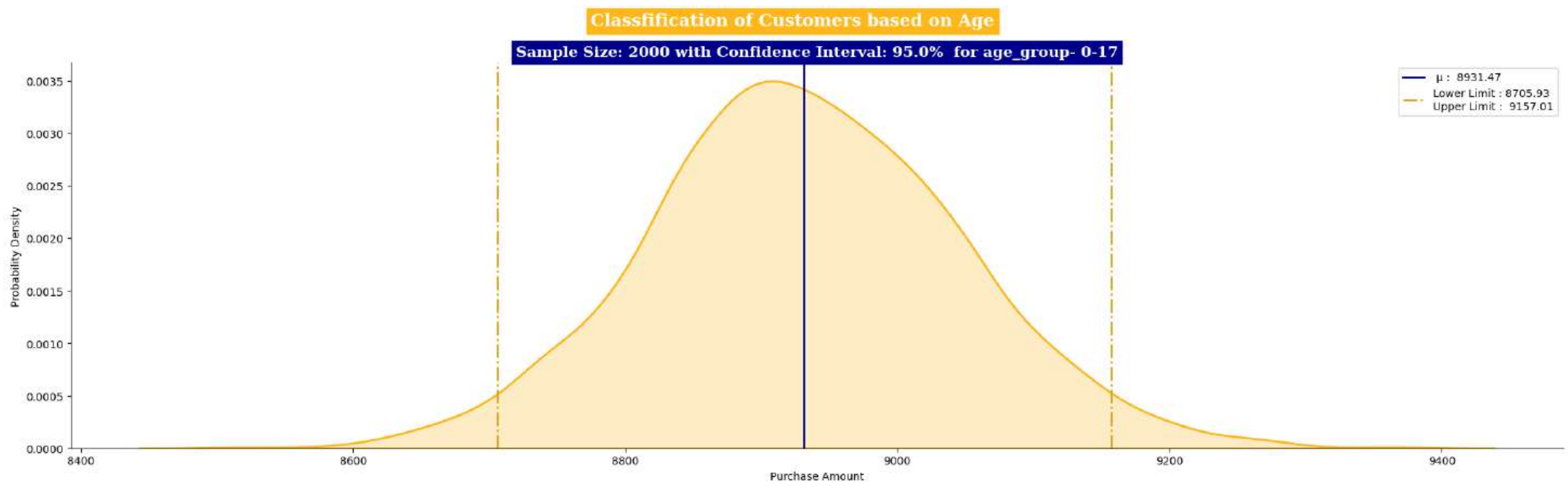


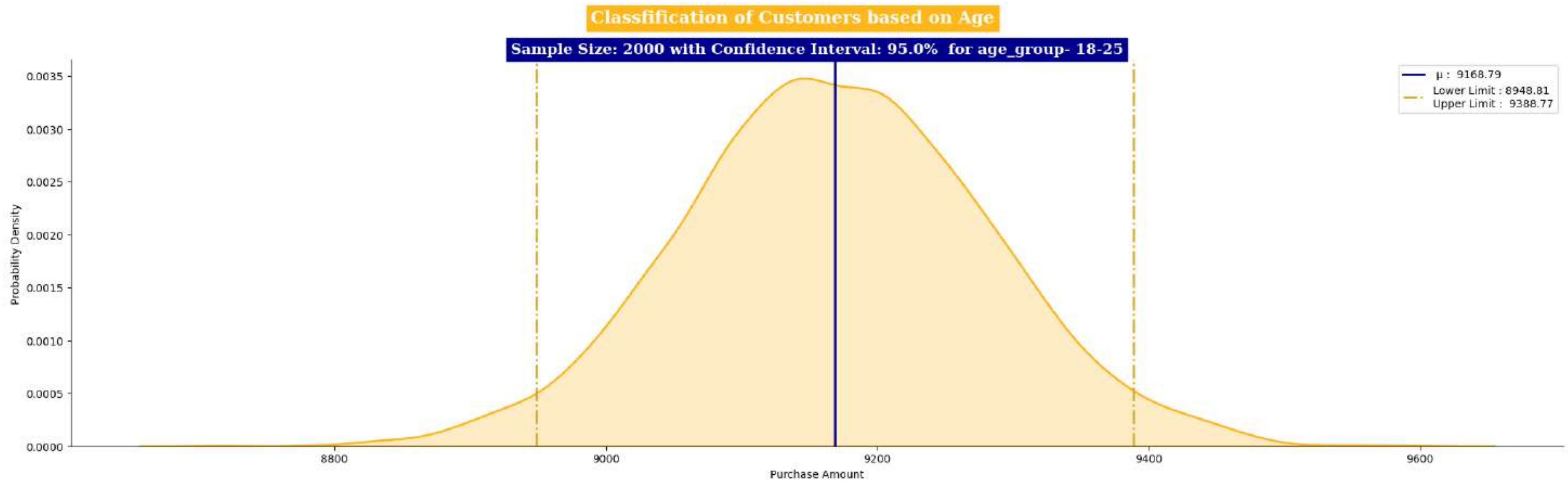

Classfification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 90.0% for age_group- 36-45
μ : 9330.27
Lower Limit : 9142.40
Upper Limit : 9518.14
Probability Density
Purchase Amount

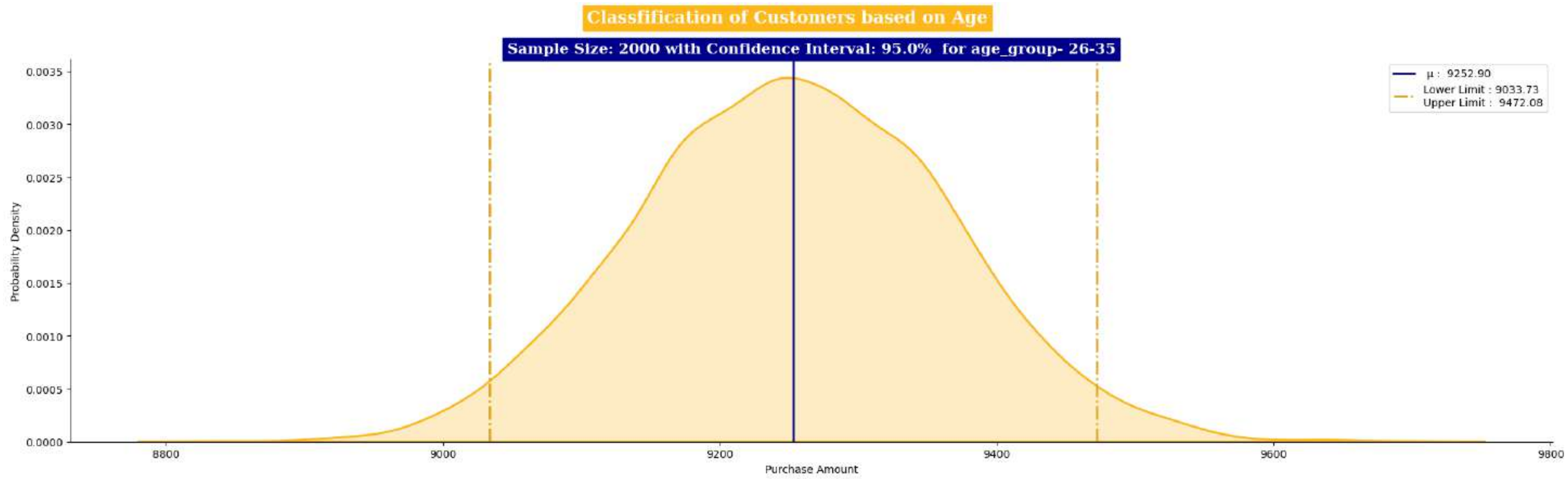

Classfification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 90.0% for age_group- 46-50
μ : 9208.45
Lower Limit : 9025.40
Upper Limit : 9391.49
Probability Density
Purchase Amount

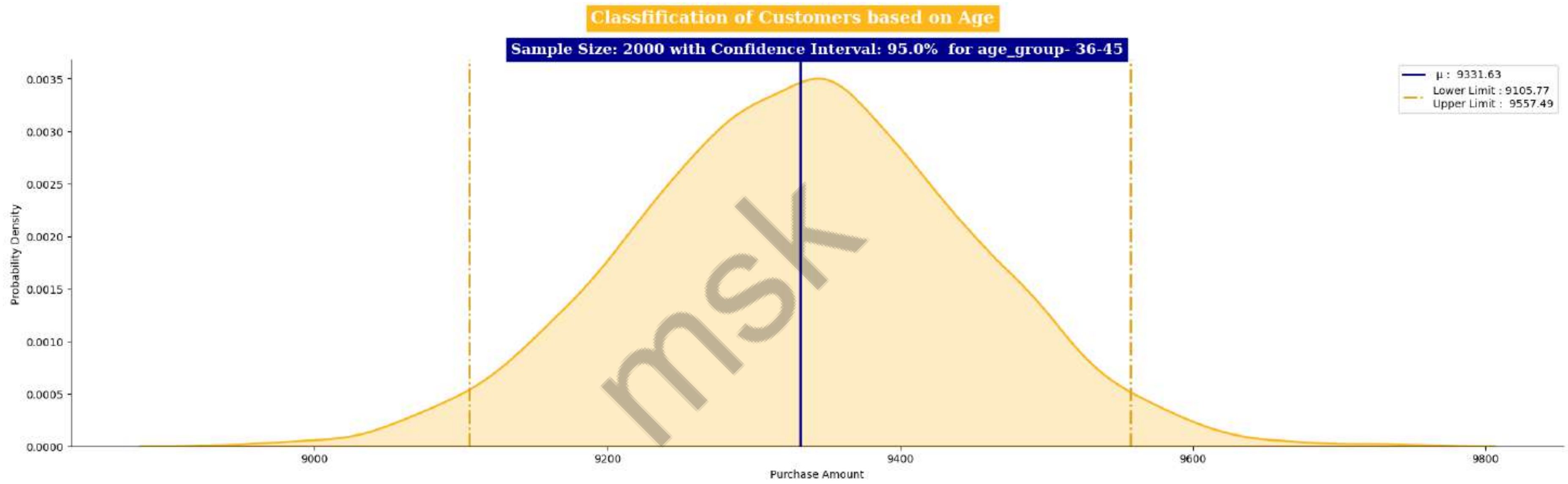

Classfification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 90.0% for age_group- 51-55
μ : 9532.51
Lower Limit : 9346.19
Upper Limit : 9718.83
Probability Density
Purchase Amount

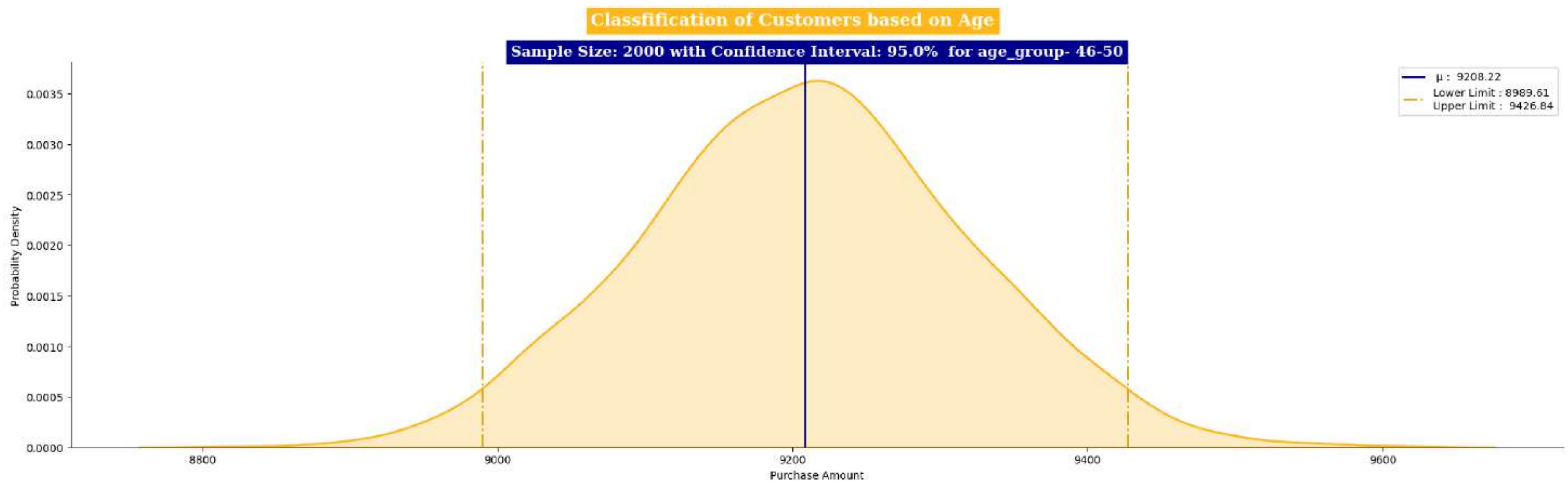

Classfification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 90.0% for age_group- 55+
μ : 9336.71
Lower Limit : 9151.77
Upper Limit : 9521.66
Probability Density
Purchase Amount

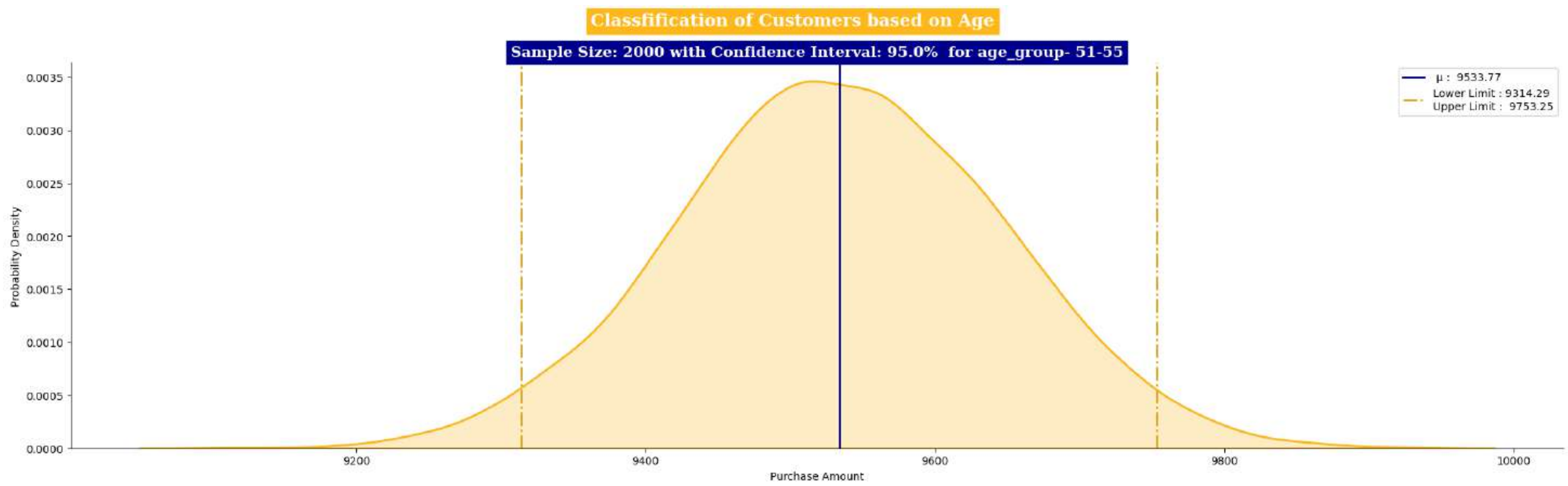

Classfification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 95.0% for age_group- 0-17
μ : 8931.47
Lower Limit : 8705.93
Upper Limit : 9157.01
Probability Density
Purchase Amount

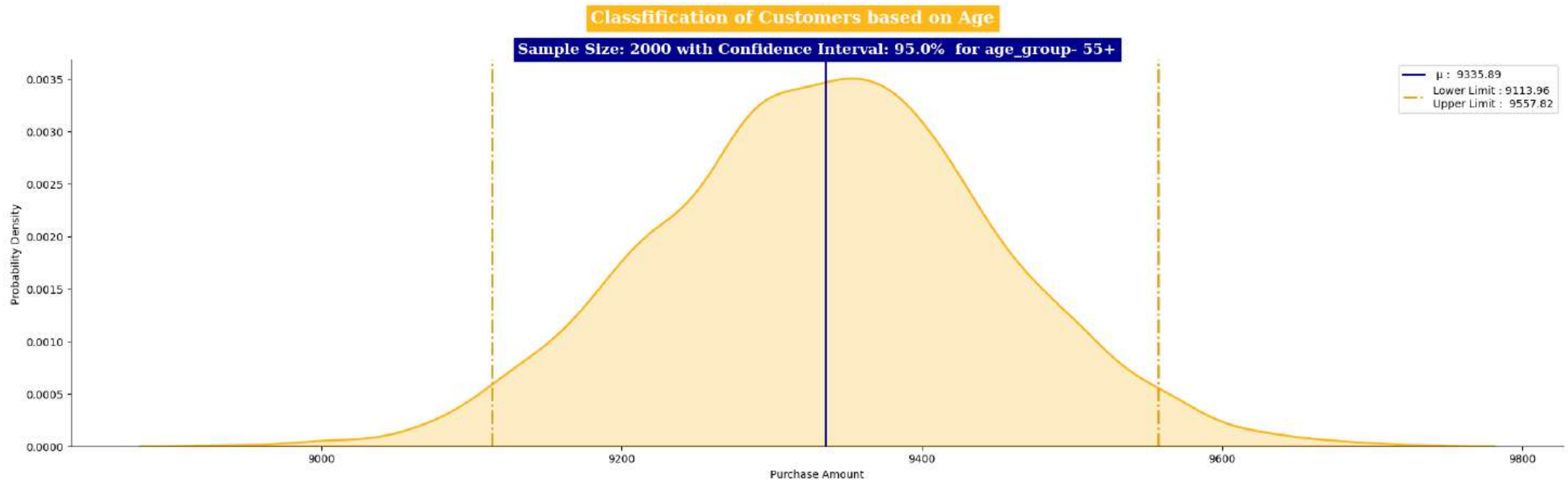

Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 95.0% for age_group- 18-25
μ : 9168.79
Lower Limit : 8948.81
Upper Limit : 9388.77
Probability Density
Purchase Amount

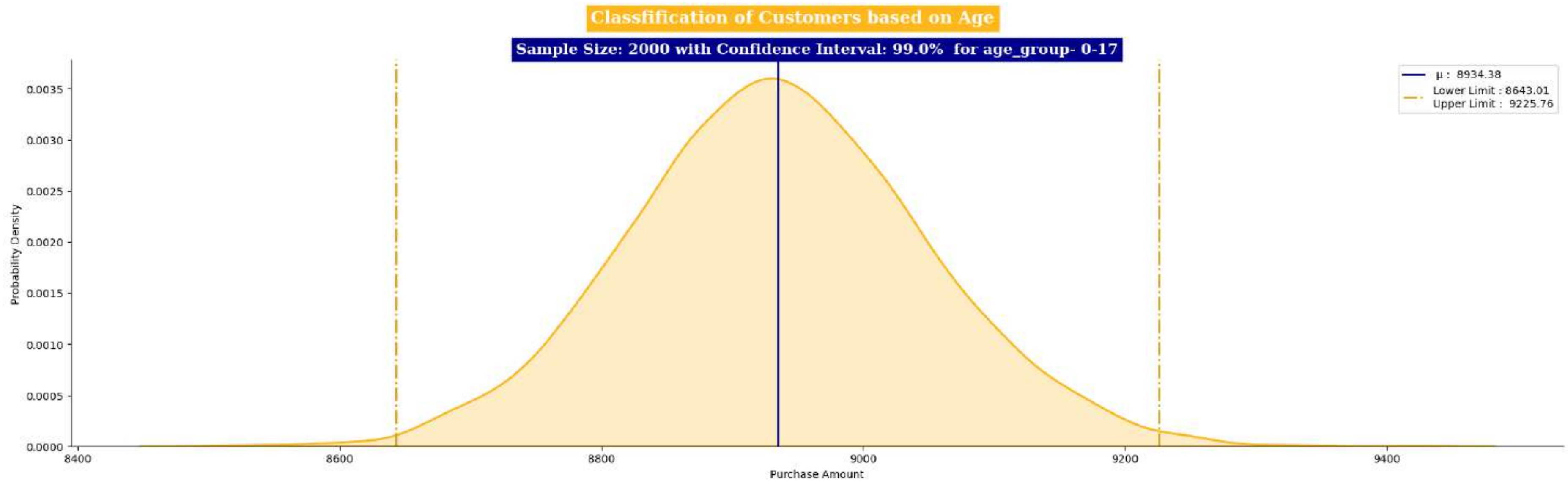

Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 95.0% for age_group- 26-35
μ : 9252.90
Lower Limit : 9033.73
Upper Limit : 9472.08
Probability Density
Purchase Amount

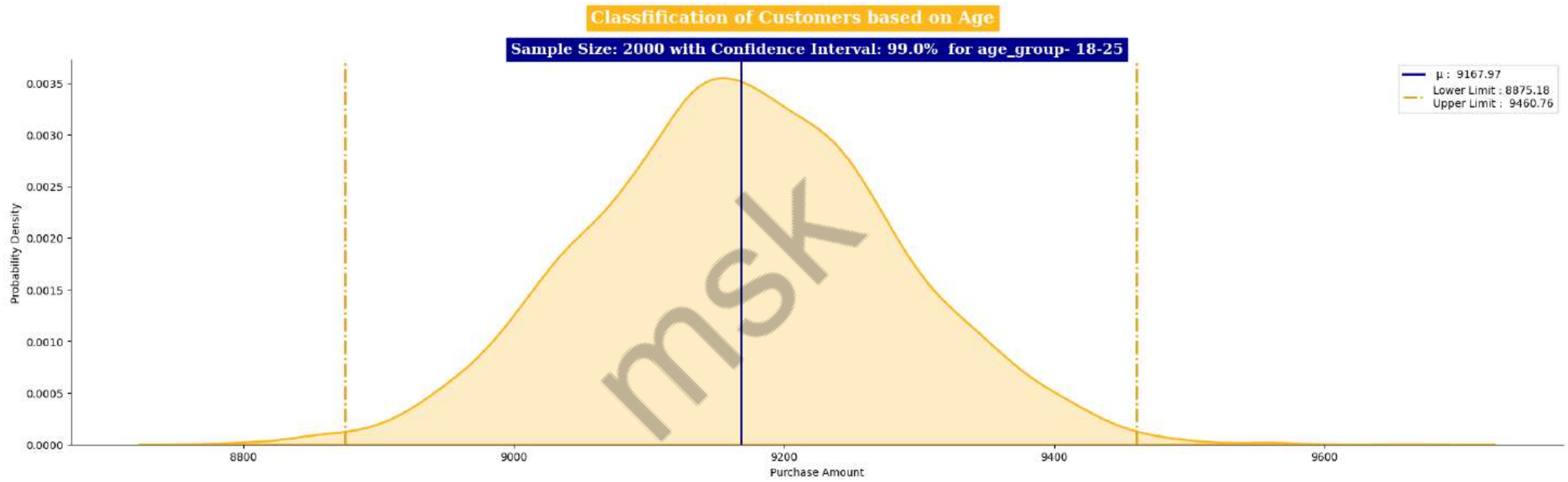

Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 95.0% for age_group- 36-45
μ : 9331.63
Lower Limit : 9105.77
Upper Limit : 9557.49
Probability Density
Purchase Amount

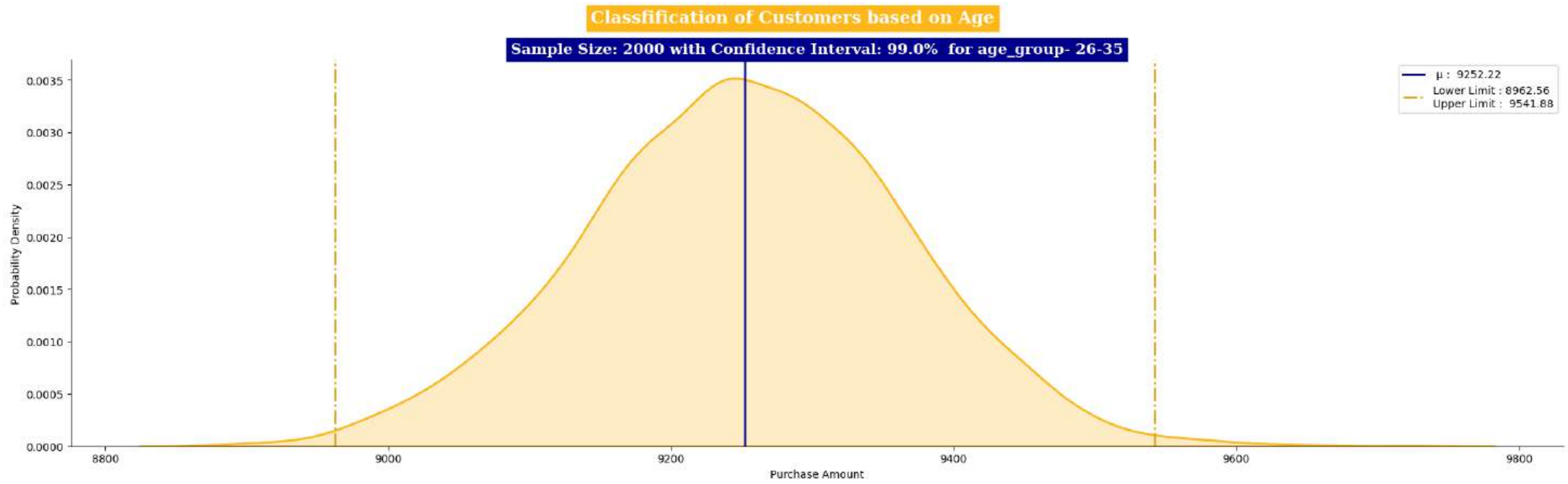

Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 95.0% for age_group- 46-50
μ : 9208.22
Lower Limit : 8989.61
Upper Limit : 9426.84
Probability Density
Purchase Amount

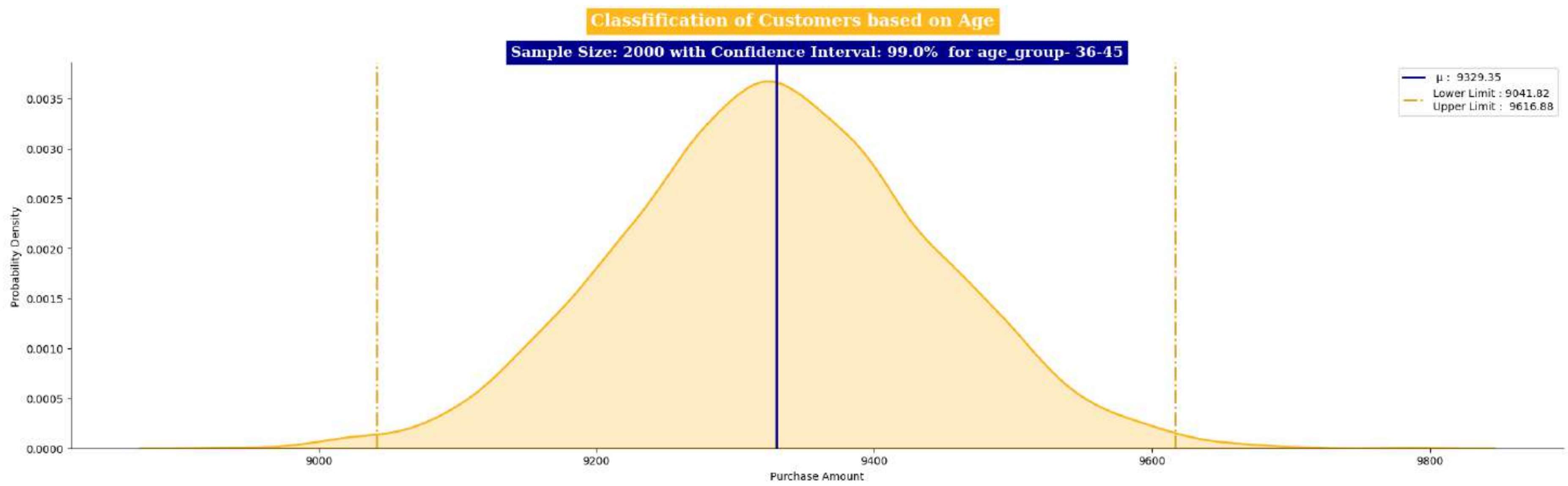

Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 95.0% for age_group- 51-55
μ : 9533.77
Lower Limit : 9314.29
Upper Limit : 9753.25
Probability Density
Purchase Amount

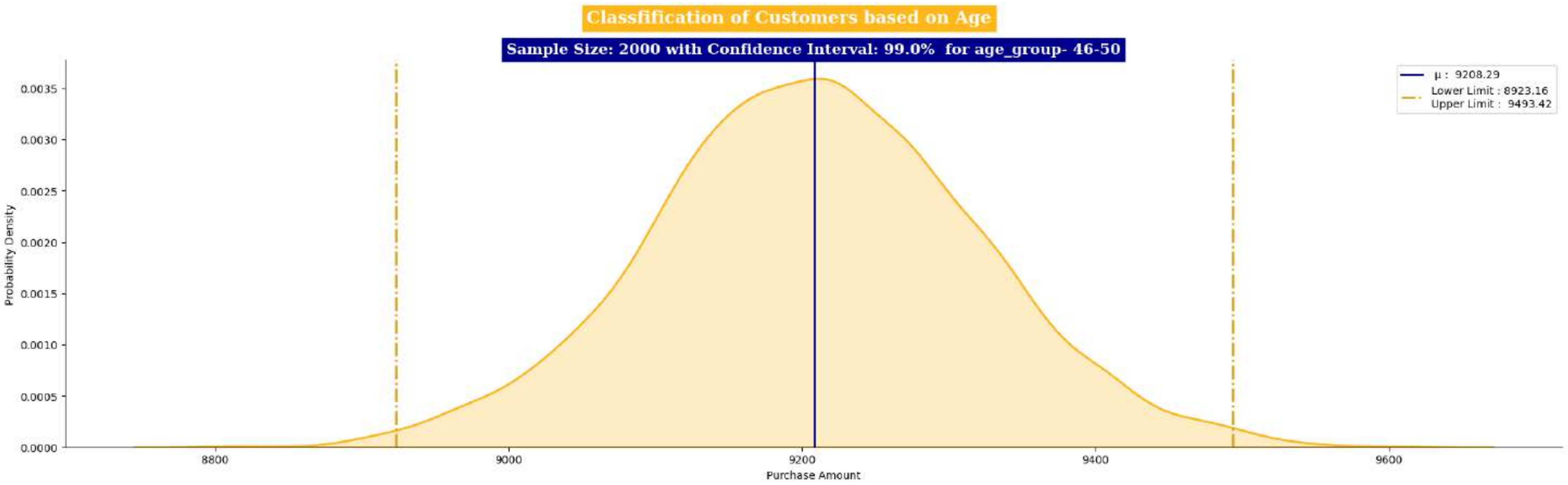

Classfification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 95.0% for age_group- 55+
μ : 9335.89
Lower Limit : 9113.96
Upper Limit : 9557.82
Probability Density
Purchase Amount

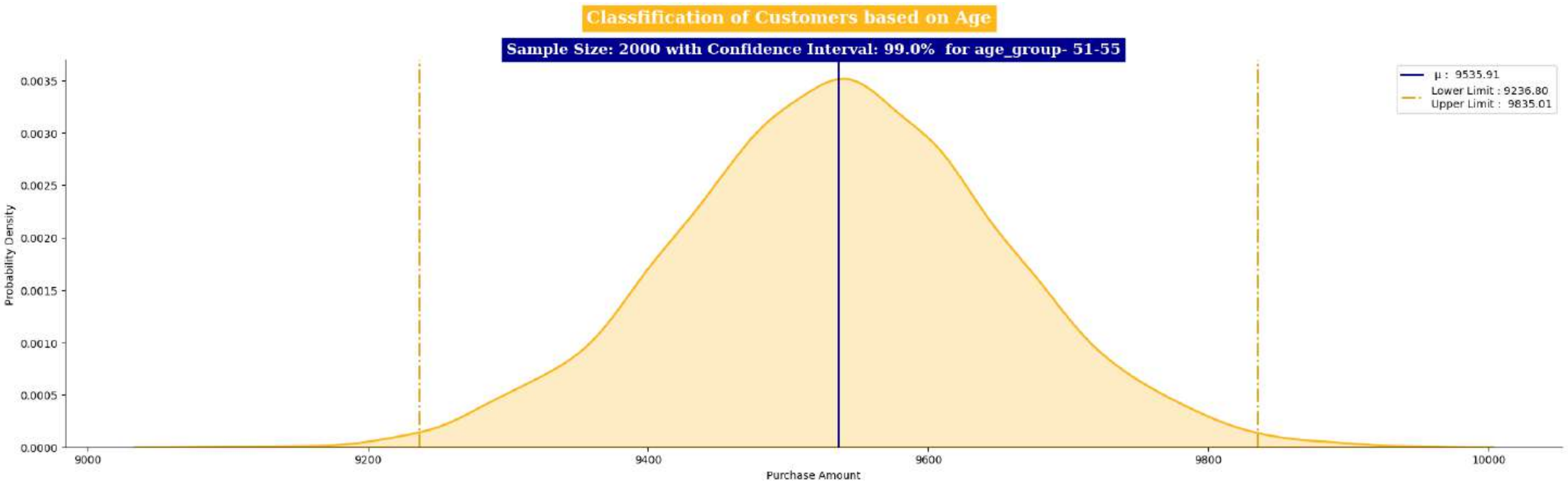

Classfification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 99.0% for age_group- 0-17
μ : 8934.38
Lower Limit : 8643.01
Upper Limit : 9225.76
Probability Density
Purchase Amount

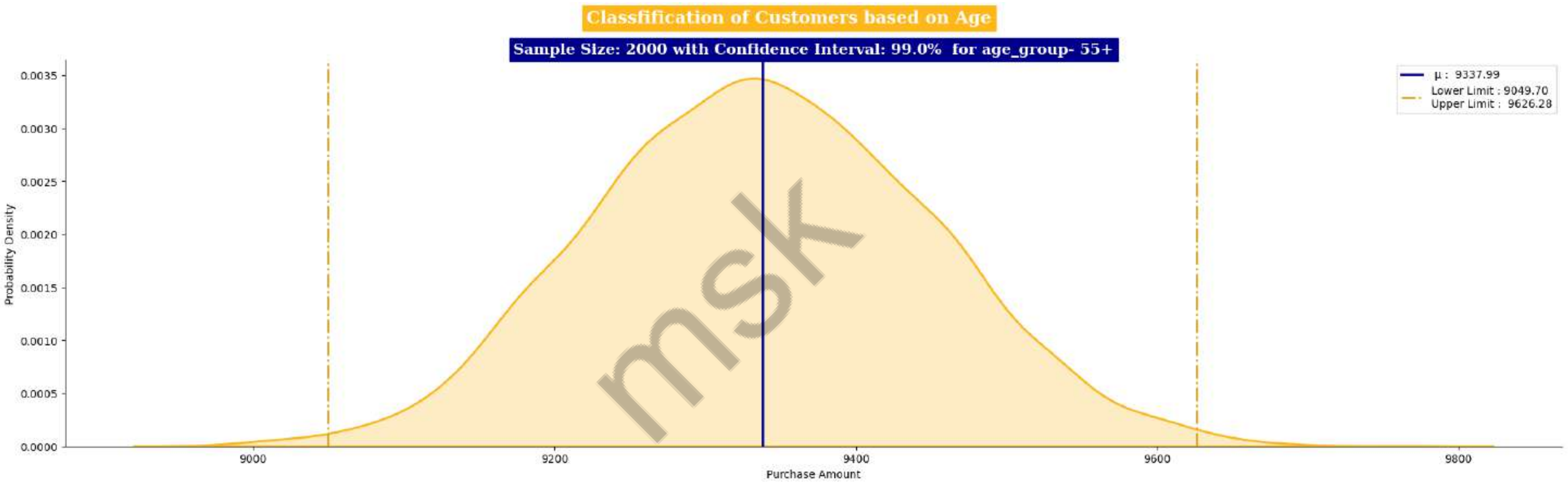

Classfification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 99.0% for age_group- 18-25
μ : 9167.97
Lower Limit : 8875.18
Upper Limit : 9460.76
Probability Density
Purchase Amount


Classfification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 99.0% for age_group- 26-35
μ : 9252.22
Lower Limit : 8962.56
Upper Limit : 9541.88
Probability Density
Purchase Amount


Classfification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 99.0% for age_group- 36-45
μ : 9329.35
Lower Limit : 9041.82
Upper Limit : 9616.88
Probability Density
Purchase Amount







Distribution for sum of purchase amt for unique male and female customers

In [514…

```
age_unique_cust = wm.groupby(['User_ID', 'Age'])[['Purchase']].sum().reset_index()
age_unique_cust = age_unique_cust[age_unique_cust['Purchase']!=0]
age_unique_cust
```

Out[514]:

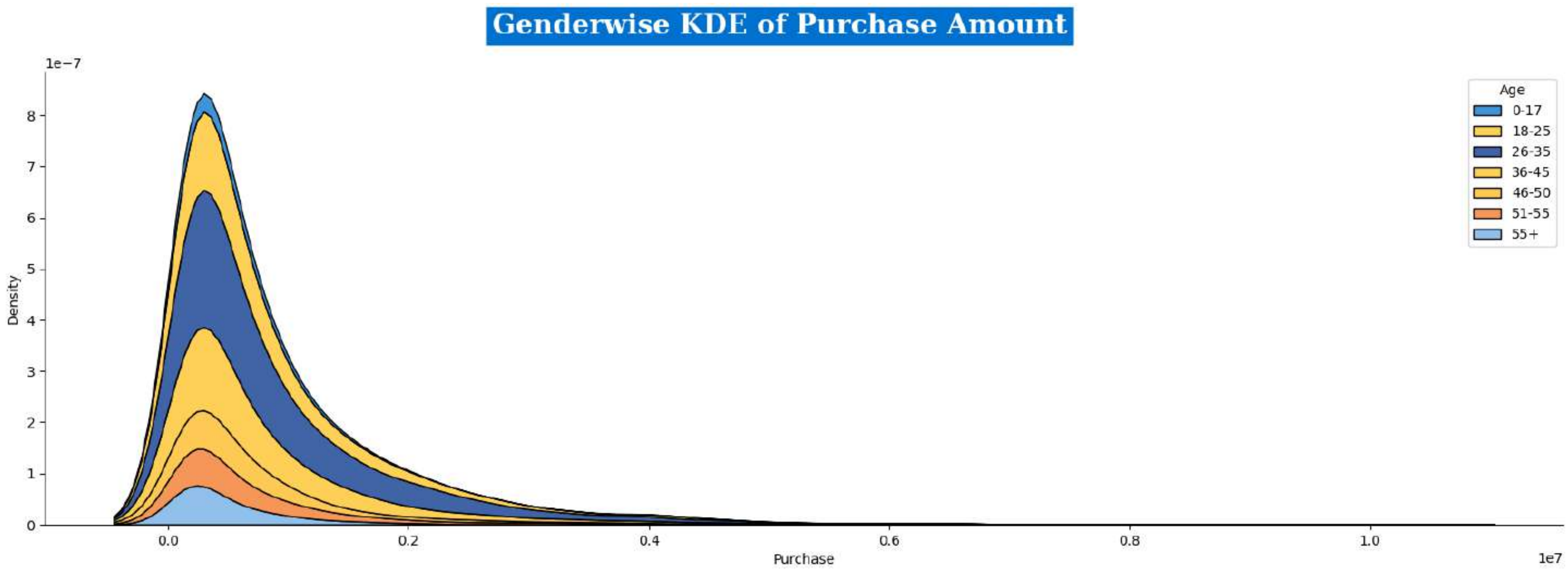
	User_ID	Age	Purchase
0	1000001	0-17	334093
13	1000002	55+	810472
16	1000003	26-35	341635
25	1000004	46-50	206468
30	1000005	26-35	821001
...	...	...	...
41204	1006036	26-35	4116058
41213	1006037	46-50	1119538
41222	1006038	55+	90034
41227	1006039	46-50	590319
41232	1006040	26-35	1653299

5891 rows × 3 columns

In [519…

```
plt.figure(figsize=(20,6))
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')
plt.suptitle('Genderwise KDE of Purchase Amount',fontfamily='serif',fontweight='bold',fontsize=20,
             backgroundcolor=cp2[0],color='w')
sns.kdeplot(data=age_unique_cust, x='Purchase', hue='Age', multiple='stack',palette=cp3)
```

```
sns.despine()
plt.show()
```

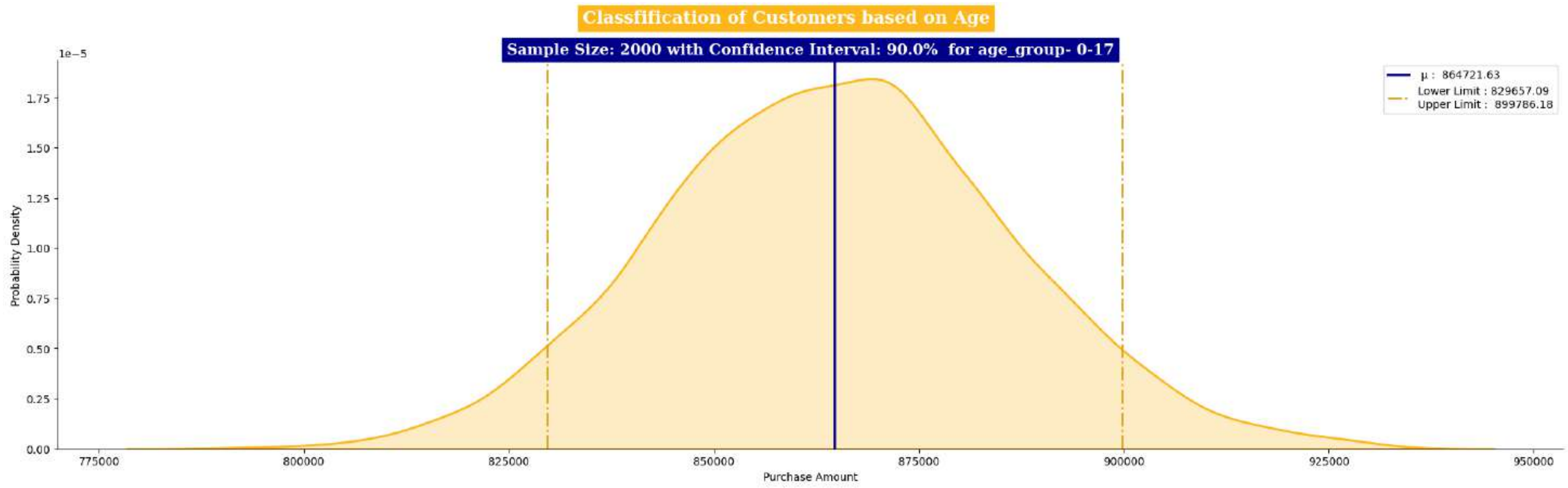


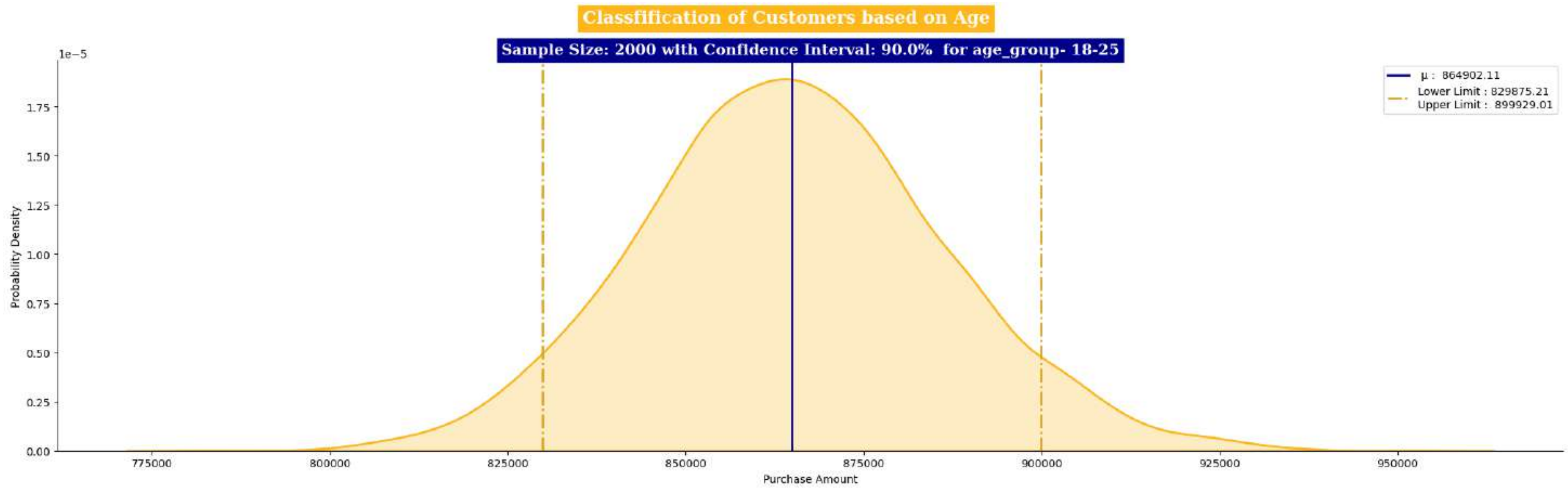
In [516…

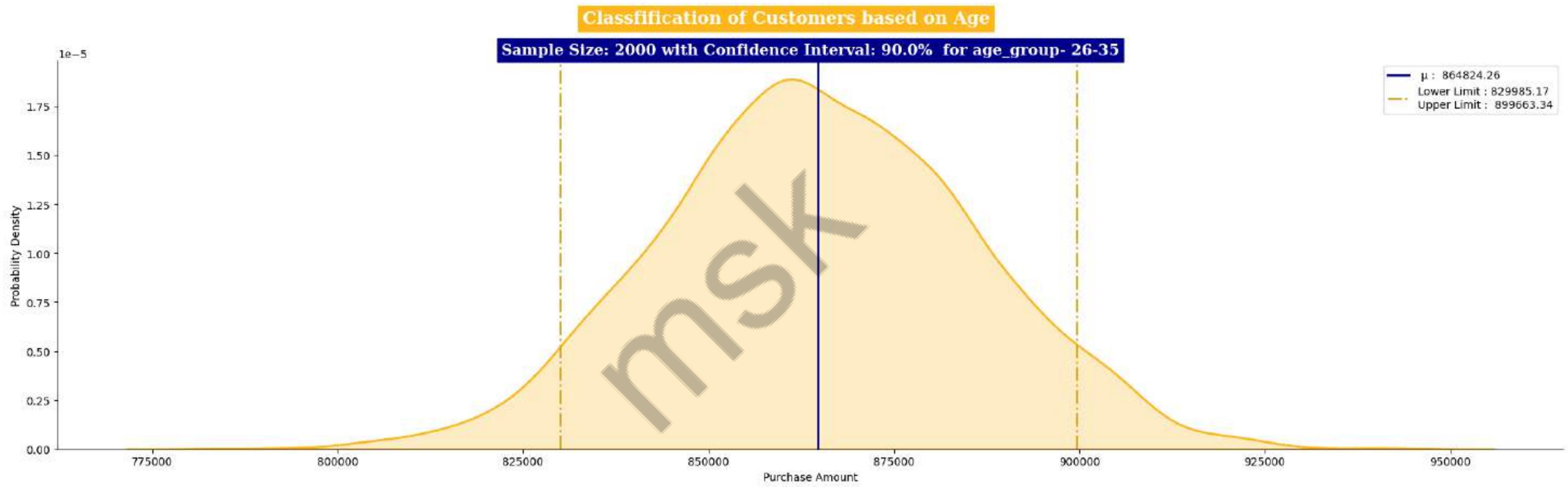
```
sample_size = 2000
ntimes = 5000
ci = [90,95,99]
age_group = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

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

for i in ci:
    for _ in age_group:
        age_data = age_unique_cust['Purchase']
        avg, ll, ul, mean = bootstrapping_age( _ , age_data , sample_size , ntimes , i )
        df.loc[len(df.index)]=[ _ , sample_size , ll , ul , mean , i , (ul-ll) ]

df
```

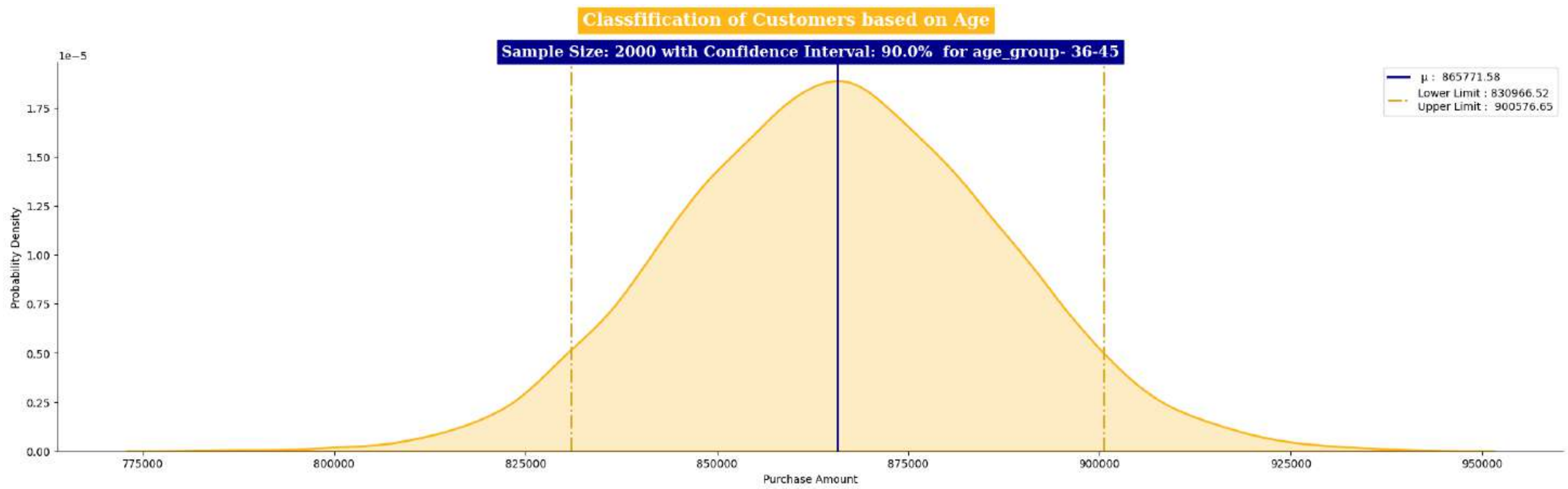
Out[516]:

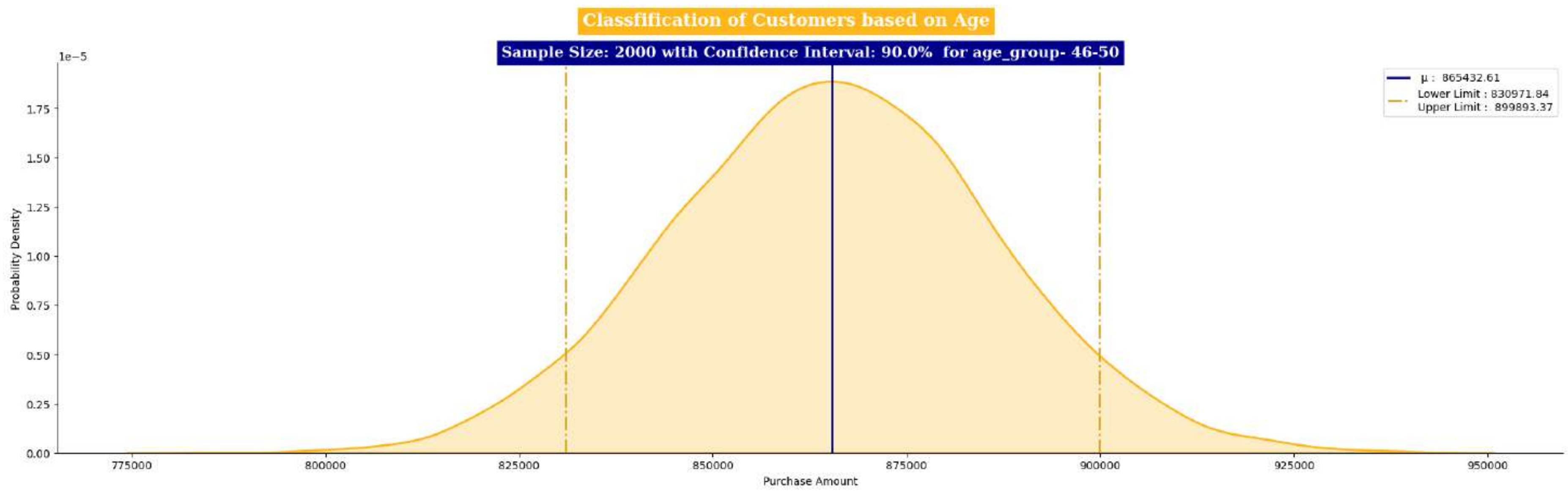
	Age_Group	Sample Size	Lower Limit	Upper Limit	Sample Mean	Confidence Interval	Range
0	0-17	2000	829657.09	899786.18	864721.63	90	70129.09
1	18-25	2000	829875.21	899929.01	864902.11	90	70053.80
2	26-35	2000	829985.17	899663.34	864824.26	90	69678.17
3	36-45	2000	830966.52	900576.65	865771.58	90	69610.13
4	46-50	2000	830971.84	899893.37	865432.61	90	68921.53
5	51-55	2000	830644.34	900418.13	865531.24	90	69773.79
6	55+	2000	830905.00	899665.47	865285.23	90	68760.47
7	0-17	2000	823545.13	906293.58	864919.35	95	82748.45
8	18-25	2000	823338.42	906436.90	864887.66	95	83098.48
9	26-35	2000	823025.79	906401.29	864713.54	95	83375.50
10	36-45	2000	823280.73	906298.79	864789.76	95	83018.06
11	46-50	2000	824733.33	905703.72	865218.53	95	80970.39
12	51-55	2000	823774.78	906600.54	865187.66	95	82825.76
13	55+	2000	824269.56	905656.39	864962.98	95	81386.83
14	0-17	2000	811254.95	918656.09	864955.52	99	107401.14
15	18-25	2000	811194.30	919923.12	865558.71	99	108728.82
16	26-35	2000	811254.93	919582.84	865418.88	99	108327.91
17	36-45	2000	810540.31	918995.77	864768.04	99	108455.46
18	46-50	2000	811900.38	918490.94	865195.66	99	106590.56
19	51-55	2000	809860.08	919052.82	864456.45	99	109192.74
20	55+	2000	809500.94	920304.84	864902.89	99	110803.90

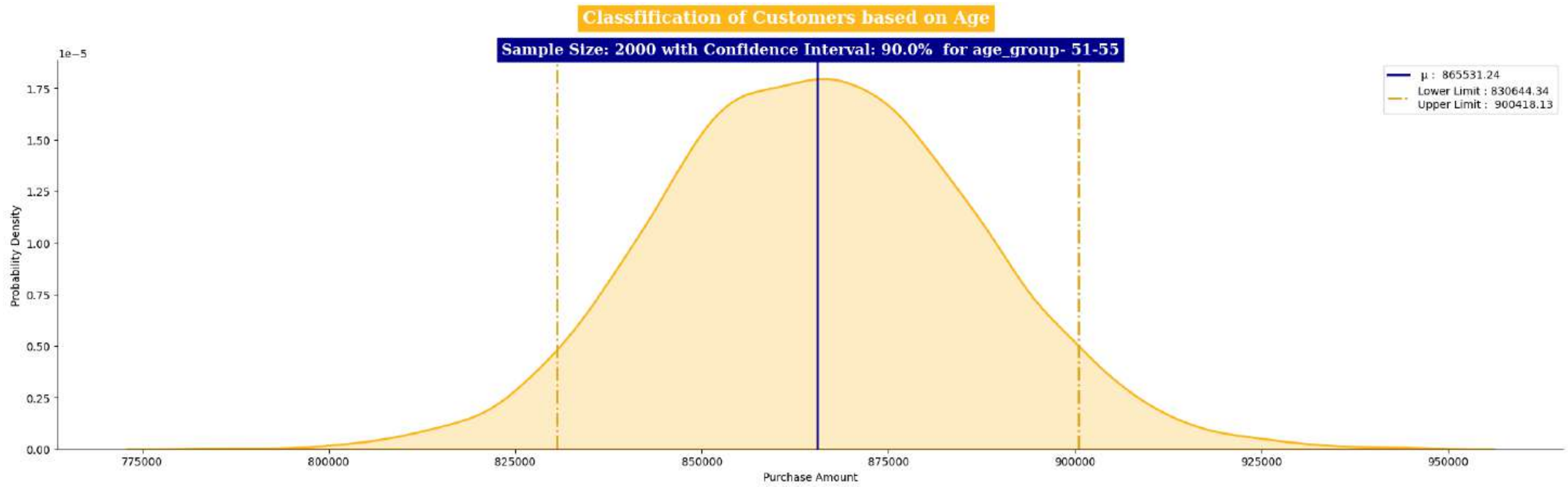

Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 90.0% for age_group- 0-17
μ : 864721.63
Lower Limit : 829657.09
Upper Limit : 899786.18
1e−5
Probability Density
Purchase Amount

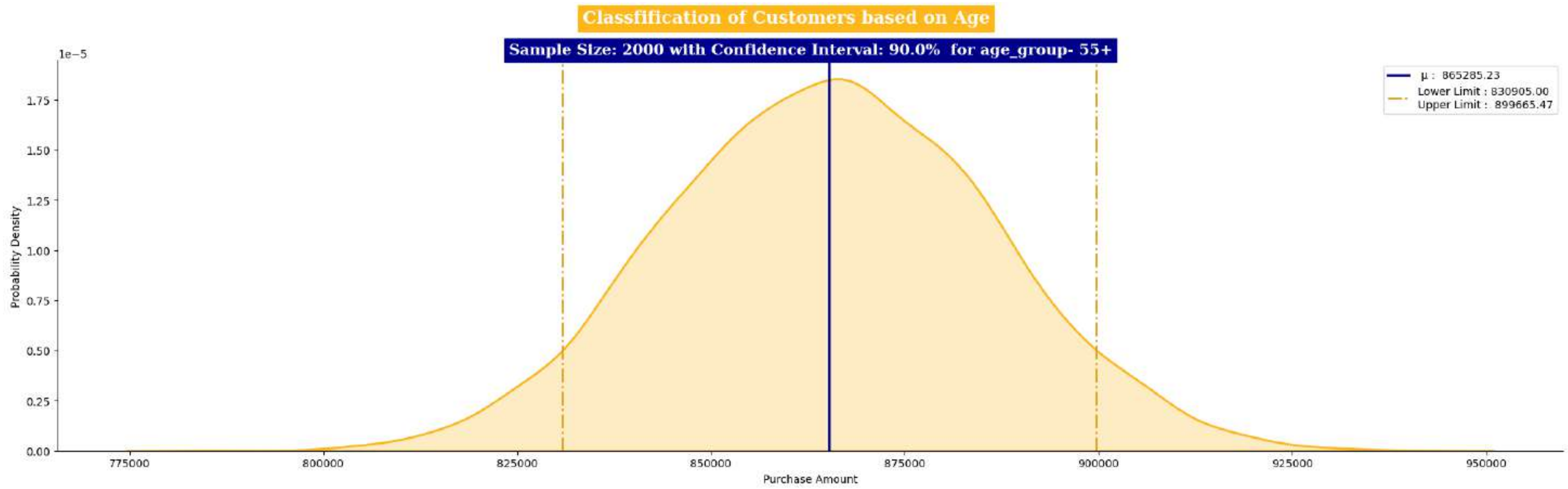

Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 90.0% for age_group- 18-25
μ : 864902.11
Lower Limit : 829875.21
Upper Limit : 899929.01
1e−5
Probability Density
Purchase Amount

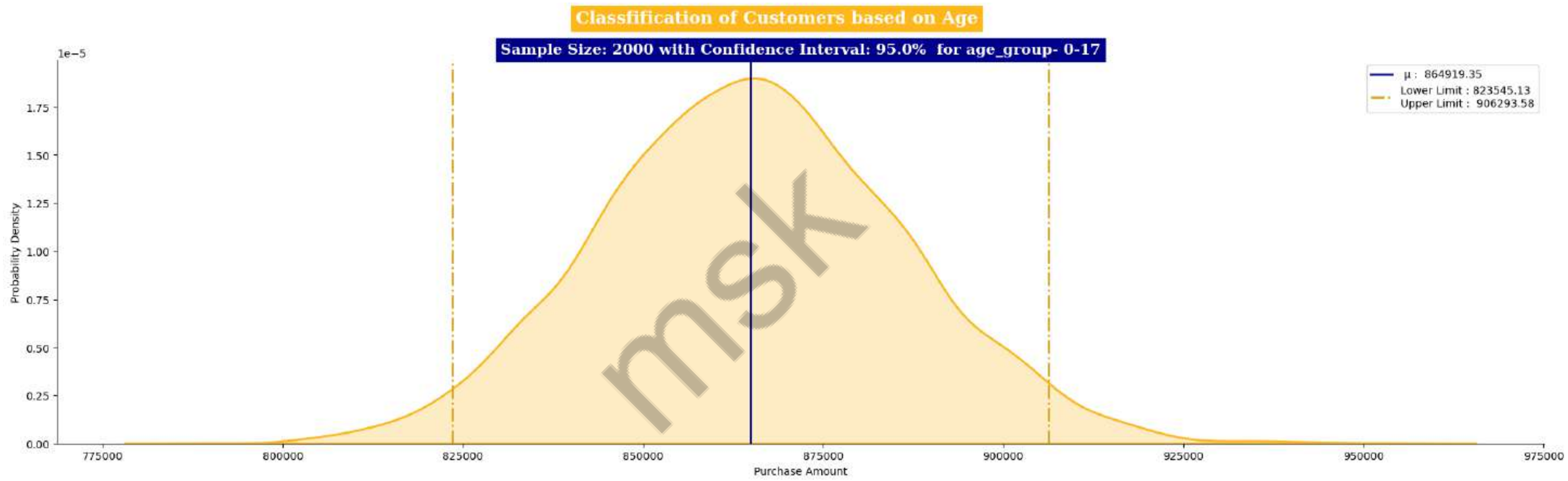

Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 90.0% for age_group- 26-35
μ : 864824.26
Lower Limit : 829985.17
Upper Limit : 899663.34
1e−5
Probability Density
Purchase Amount

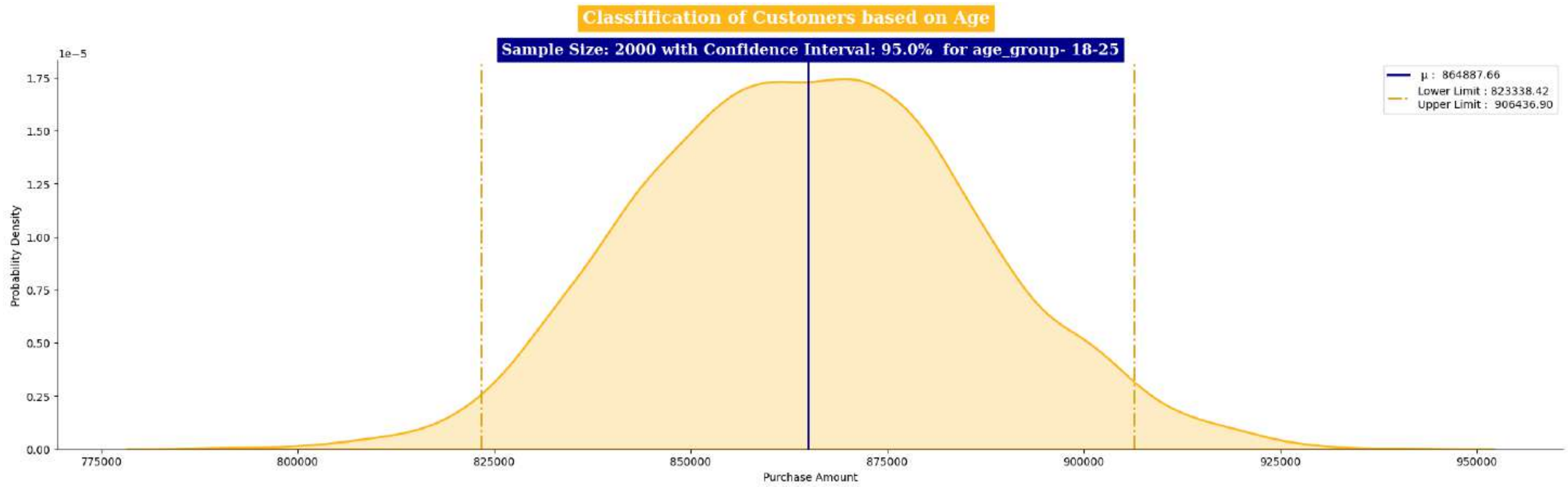

Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 90.0% for age_group- 36-45
μ : 865771.58
Lower Limit : 830966.52
Upper Limit : 900576.65
1e−5
Probability Density
Purchase Amount

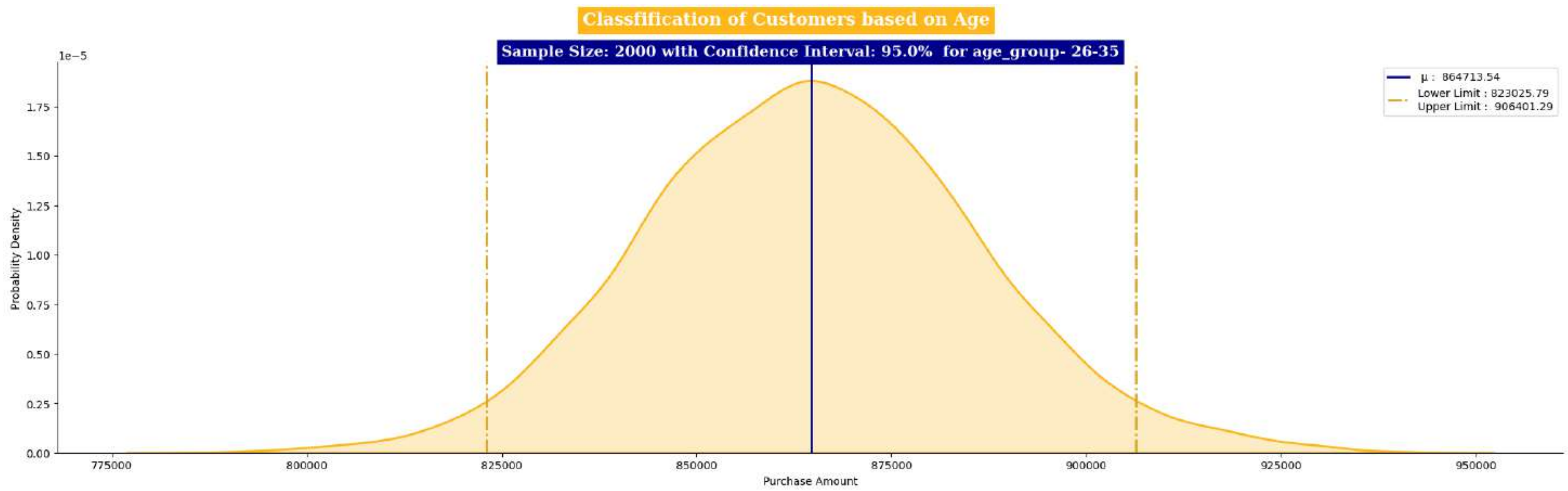

Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 90.0% for age_group- 46-50
μ : 865432.61
Lower Limit : 830971.84
Upper Limit : 899893.37
1e−5
Probability Density
Purchase Amount

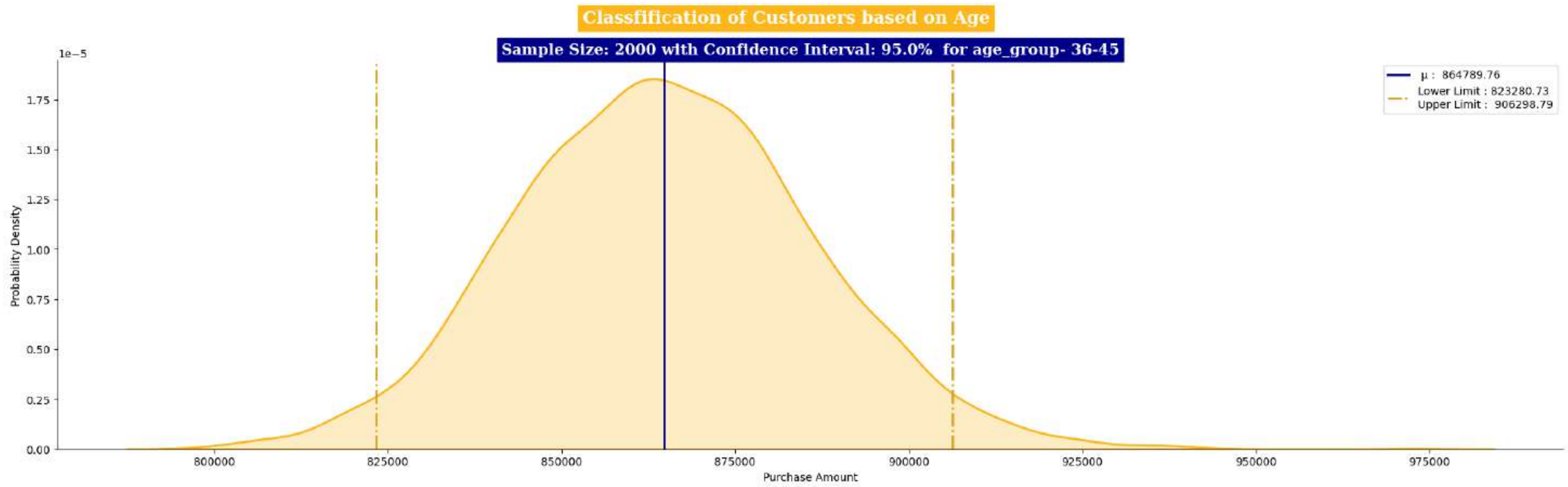

Classfification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 90.0% for age_group- 51-55
μ : 865531.24
Lower Limit : 830644.34
Upper Limit : 900418.13
Probability Density
Purchase Amount

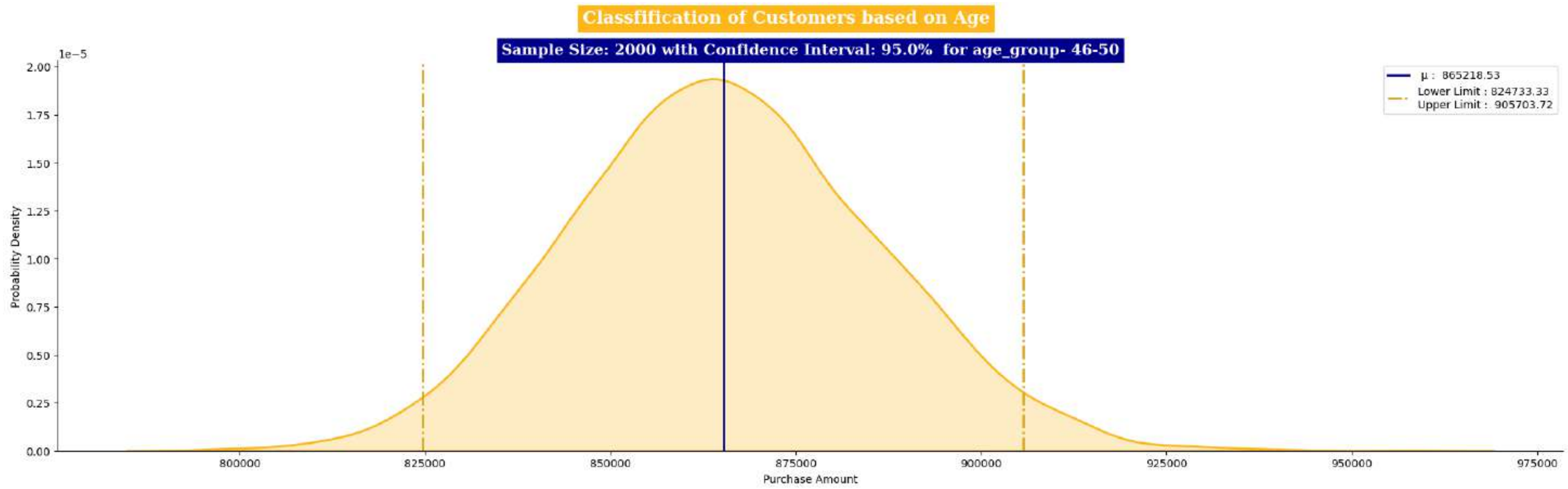

Classfification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 90.0% for age_group- 55+
μ : 865285.23
Lower Limit : 830905.00
Upper Limit : 899665.47
Probability Density
Purchase Amount

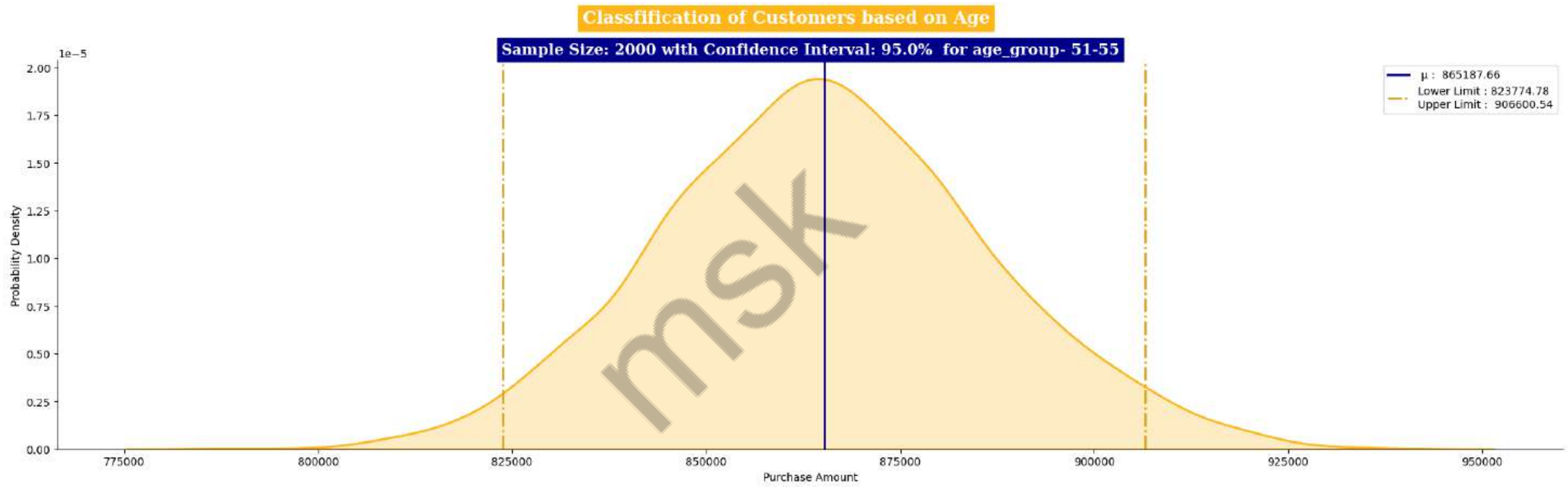

Classfification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 95.0% for age_group- 0-17
μ : 864919.35
Lower Limit : 823545.13
Upper Limit : 906293.58
Probability Density
Purchase Amount

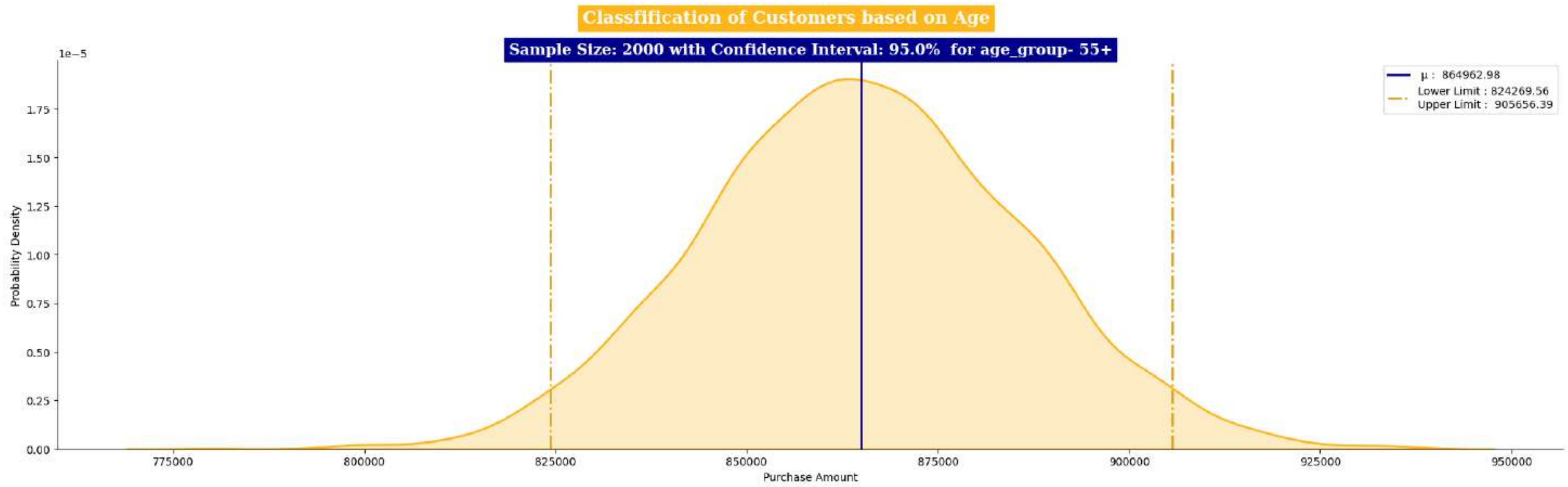

Classfification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 95.0% for age_group- 18-25
μ : 864887.66
Lower Limit : 823338.42
Upper Limit : 906436.90
Probability Density
Purchase Amount

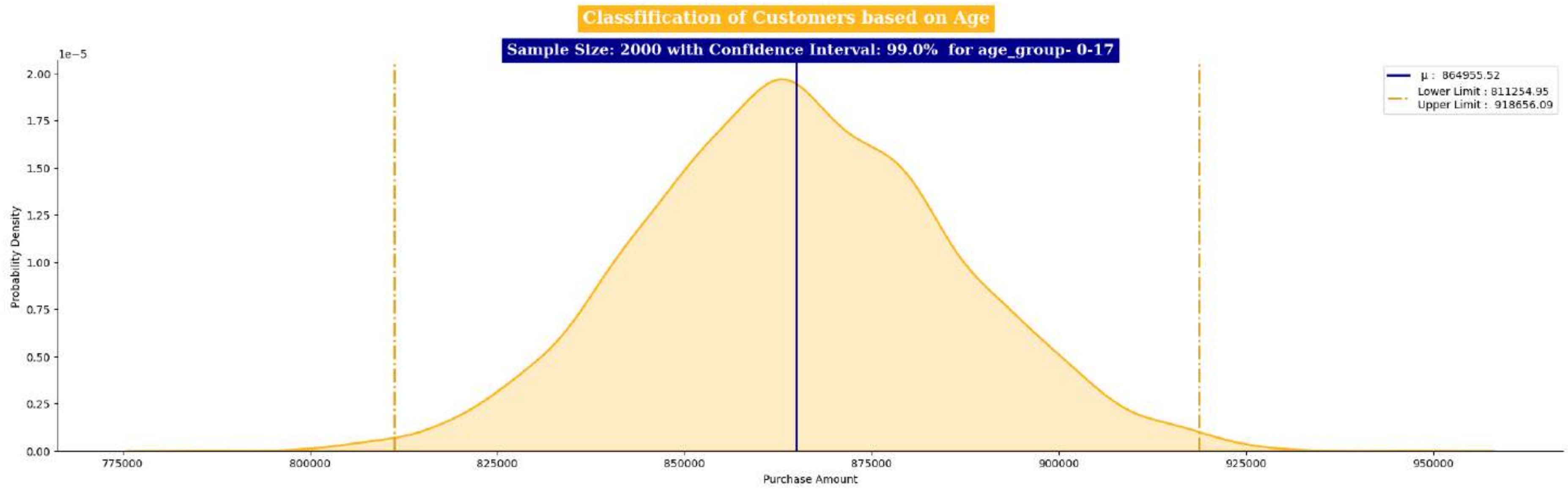

Classfification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 95.0% for age_group- 26-35
μ : 864713.54
Lower Limit : 823025.79
Upper Limit : 906401.29
Probability Density
Purchase Amount

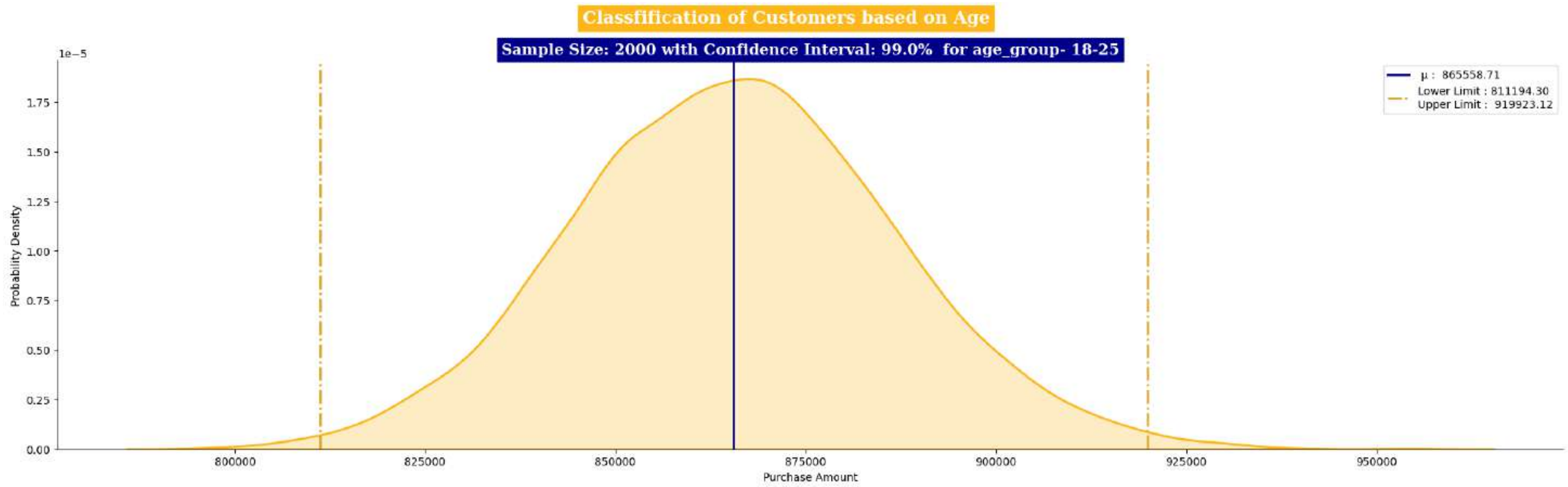

Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 95.0% for age_group- 36-45
μ : 864789.76
Lower Limit : 823280.73
Upper Limit : 906298.79
Probability Density
1e−5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
800000
825000
850000
875000
900000
925000
950000
975000
Purchase Amount

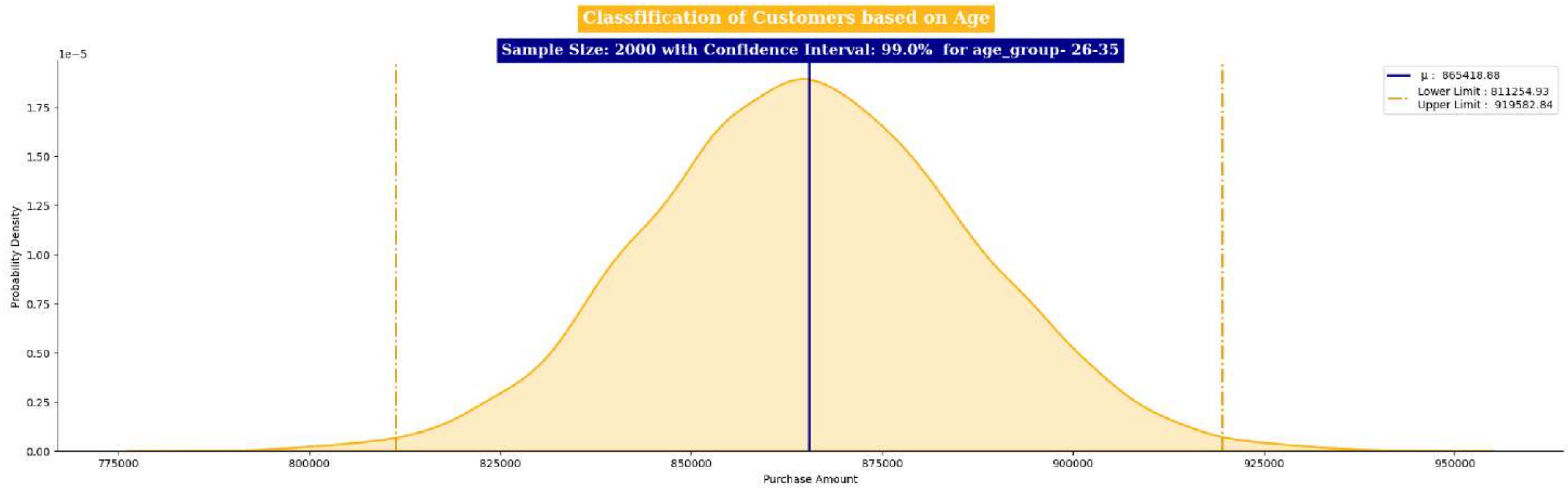

Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 95.0% for age_group- 46-50
μ : 865218.53
Lower Limit : 824733.33
Upper Limit : 905703.72
Probability Density
1e−5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
800000
825000
850000
875000
900000
925000
950000
975000
Purchase Amount

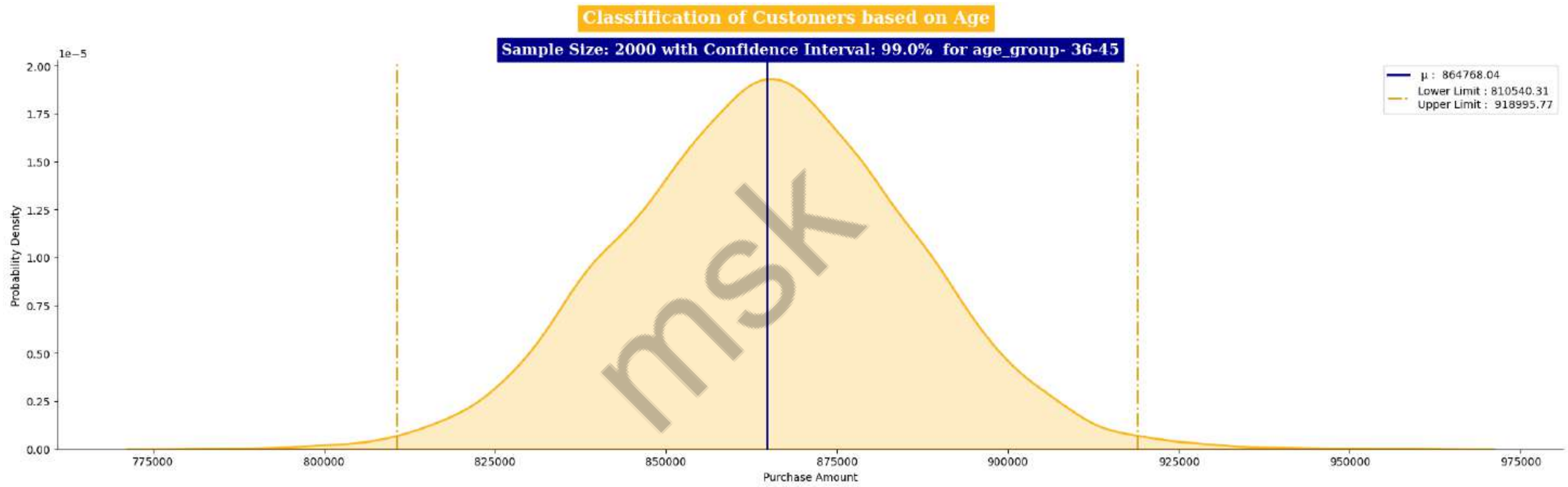

Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 95.0% for age_group- 51-55
μ : 865187.66
Lower Limit : 823774.78
Upper Limit : 906600.54
Probability Density
1e−5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
775000
800000
825000
850000
875000
900000
925000
950000
Purchase Amount

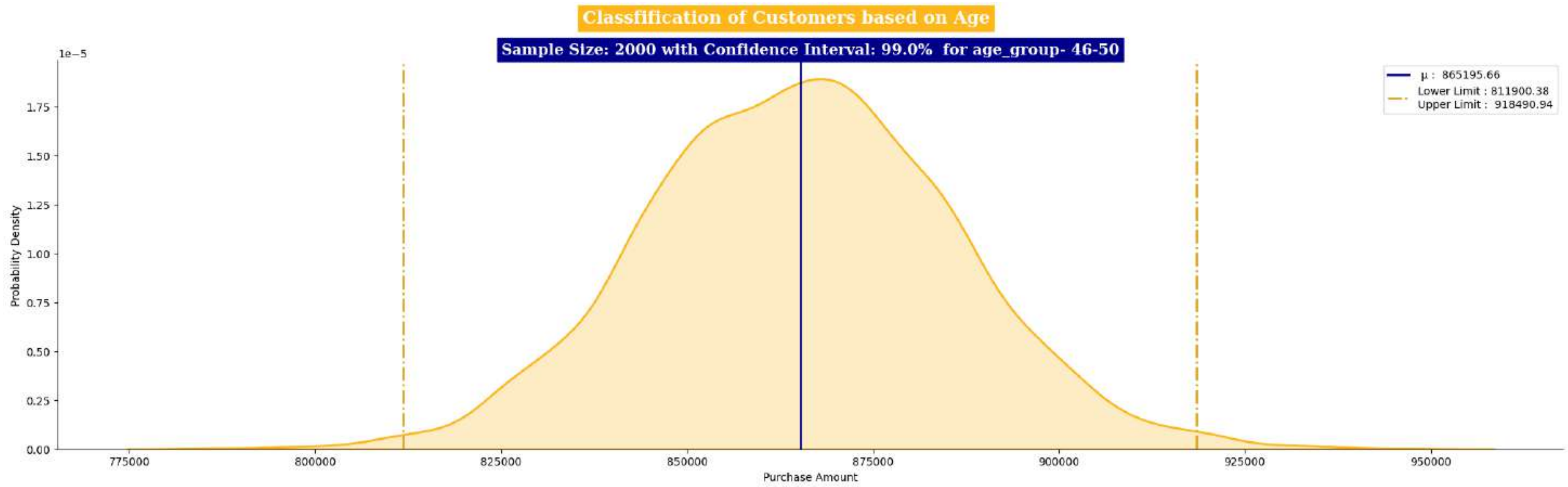

Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 95.0% for age_group- 55+
μ : 864962.98
Lower Limit : 824269.56
Upper Limit : 905656.39
Probability Density
1e−5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
775000
800000
825000
850000
875000
900000
925000
950000
Purchase Amount

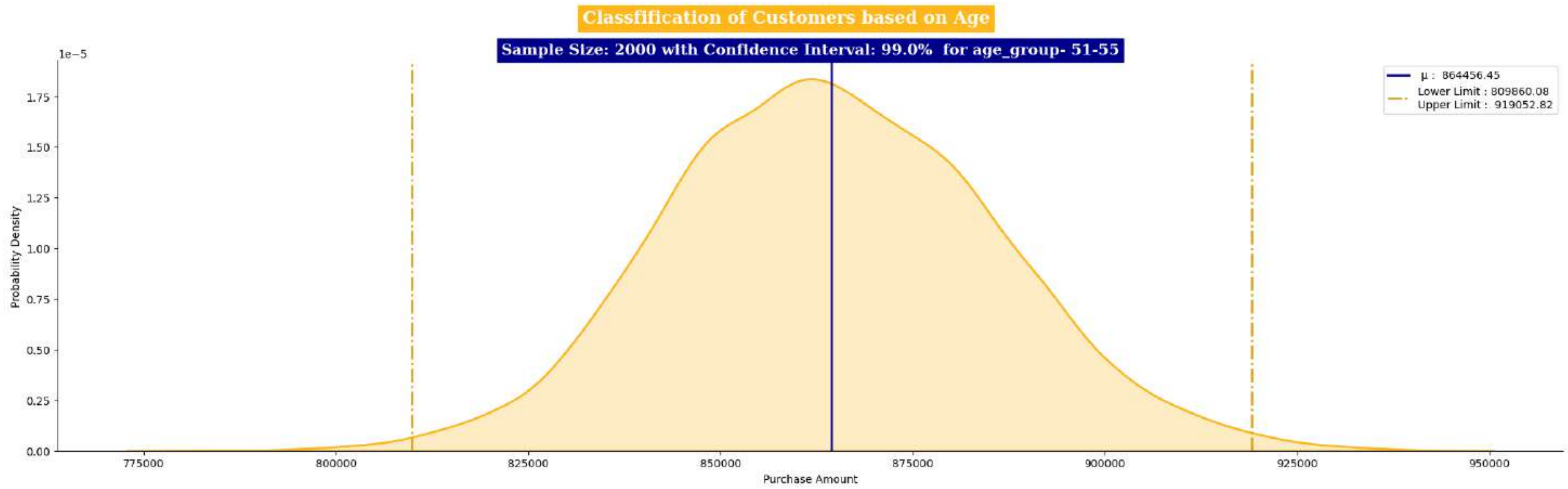

Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 99.0% for age_group- 0-17
μ : 864955.52
Lower Limit : 811254.95
Upper Limit : 918656.09
Probability Density
1e−5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
775000
800000
825000
850000
875000
900000
925000
950000
Purchase Amount

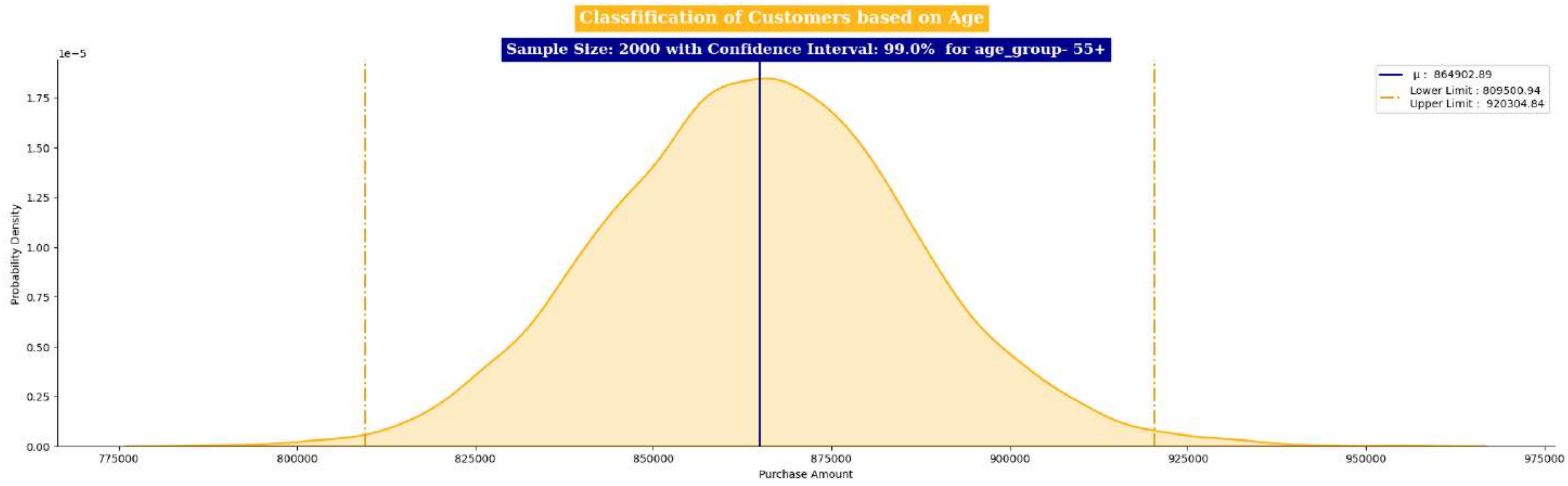

Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 99.0% for age_group- 18-25
1e−5
Probability Density
Purchase Amount
μ : 865558.71
Lower Limit : 811194.30
Upper Limit : 919923.12


Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 99.0% for age_group- 26-35
1e−5
Probability Density
Purchase Amount
μ : 865418.88
Lower Limit : 811254.93
Upper Limit : 919582.84


Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 99.0% for age_group- 36-45
1e−5
Probability Density
Purchase Amount
μ : 864768.04
Lower Limit : 810540.31
Upper Limit : 918995.77


Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 99.0% for age_group- 46-50
1e−5
Probability Density
Purchase Amount
μ : 865195.66
Lower Limit : 811900.38
Upper Limit : 918490.94


Classification of Customers based on Age
Sample Size: 2000 with Confidence Interval: 99.0% for age_group- 51-55
1e−5
Probability Density
Purchase Amount
μ : 864456.45
Lower Limit : 809860.08
Upper Limit : 919052.82



▶ 〰 ◀ Confidence intervals and distribution of the mean of the expenses based on female and male customers

In [376…

```
def bootstrapping(title,data1,data2,sample_size,ntimes,ci):

    plt.figure(figsize=(20,8))
    plt.style.use('seaborn-v0_8-bright')
    plt.suptitle(f'Classfification of Customers based on {title}',
              fontfamily='serif', fontweight='bold', fontsize=16, backgroundcolor='#ffb81c', color='w')

    ci=ci/100

    data1_sample_means = [np.mean(np.random.choice(data1,sample_size)) for i in range(ntimes)]

    data2_sample_means = []
    for i in range(ntimes):
        dsm2 = np.mean(np.random.choice(data2,sample_size))
        data2_sample_means.append(dsm2)


    # male_data_sample_means parameters
    mean1 = np.mean(data1_sample_means)
    sigma1 = np.std(data1_sample_means)
    stderr1 = stats.sem(data1_sample_means)   # sem auto calculates the std.err for mean

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

    # For female_data_sample_means parameters
    mean2 = np.mean(data2_sample_means)
    sigma2 = np.std(data2_sample_means)
    stderr2 = stats.sem(data2_sample_means)

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

    # plot1 # for mu = alt+230
    sns.kdeplot(data = data1_sample_means, color=cp2[0], fill = True, linewidth = 2)
    label_mean1 = (f" µ (Males):  {mean1:.2f}")
    plt.axvline(mean1, color = 'darkblue', linestyle = 'solid', linewidth = 2, label=label_mean1)
    label_limits1=(f"Lower Limit(M): {lower_limit_1:.2f}\nUpper Limit(M):  {upper_limit_1:.2f}")
    plt.axvline(lower_limit_1, color = 'dodgerblue', linestyle = 'dashdot', linewidth = 2, label=label_limits1)
    plt.axvline(upper_limit_1, color = 'dodgerblue', linestyle = 'dashdot', linewidth = 2)

    #plot2
    sns.kdeplot(data = data2_sample_means, color=cp2[1], fill = True, linewidth = 2)
    sns.despine()
    label_mean2 = (f" µ (Females): {mean2:.2f}")
    plt.axvline(mean2, color = 'gold', linestyle = 'solid', linewidth = 2, label=label_mean2)
    label_limits2=(f"Lower Limit(F): {lower_limit_2:.2f}\nUpper Limit(F): {upper_limit_2:.2f}")
    plt.axvline(lower_limit_2, color = 'goldenrod', linestyle = 'dashdot', linewidth = 2, label=label_limits2)
    plt.axvline(upper_limit_2, color = 'goldenrod', linestyle = 'dashdot', linewidth = 2)

    plt.title(f"Sample Size: {sample_size} with Confidence Interval: {ci*100}% ",
              #, Male Avg: {np.round(mean1, 2)} , Male SME: {np.round(stderr1, 2)} ,"
              #f" Female Avg: {np.round(mean2, 2)} , Female SME: {np.round(stderr2, 2)}
          fontfamily='serif', fontweight='bold', fontsize=14, backgroundcolor='#003087', color='w')

    plt.legend()
    plt.xlabel('Purchase Amount')
    plt.ylabel('Probability 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_li
```


〽 Lets plot the mean of 10000 Random Samples of sizes 50,500,5000,50000 and 100000 with 90%,95%,99% Confidence Interval

MALE & FEMALE

✅ 90% CI

In [522…
```
norm.ppf(0.95)
```
Out[522]:
```
1.6448536269514722
```

In [530…
```
sample_male = male_purchase.sample(500)
z_score_90 = norm.ppf(0.9).round(2)
z_score_95 = norm.ppf(0.95).round(2)
z_score_99 = norm.ppf(0.99).round(2)
print(f"Male purchase amount - Confidence interval :")
male_pop_mean = np.mean(sample_male).round(2)
print(f"Population purchase mean for male : {male_pop_mean}")
pop_std_dev = np.round(np.std(sample_male),2)
print(f"Population purchase standard deviation for male : {pop_std_dev}")
print()
se = pop_std_dev/np.sqrt(500)
x1 =  (male_pop_mean - (z_score_90 * se)).round(2)
x2 =  (male_pop_mean + (z_score_90 * se)).round(2)
print(f"The 90% confidence interval --> $ {x1} to $ {x2}")
x1 =  (male_pop_mean - (z_score_95 * se)).round(2)
x2 =  (male_pop_mean + (z_score_95 * se)).round(2)
print(f"The 95% confidence interval --> $ {x1} to $ {x2}")
x1 =  (male_pop_mean - (z_score_99 * se)).round(2)
x2 =  (male_pop_mean + (z_score_99 * se)).round(2)
print(f"The 99% confidence interval --> $ {x1} to $ {x2}")
```

```
Male purchase amount - Confidence interval :
Population purchase mean for male : 936263.85
Population purchase standard deviation for male : 1040459.91

The 90% confidence interval --> $ 876704.45 to $ 995823.25
The 95% confidence interval --> $ 859953.37 to $ 1012574.33
The 99% confidence interval --> $ 827847.13 to $ 1044680.57
```

In [529…
```
sample_female = female_purchase.sample(500)
z_score_90 = norm.ppf(0.9).round(2)
z_score_95 = norm.ppf(0.95).round(2)
z_score_99 = norm.ppf(0.99).round(2)
print(f"Female purchase amount - Confidence interval :")
male_pop_mean = np.mean(sample_female).round(2)
print(f"Population purchase mean for Female : {male_pop_mean}")
pop_std_dev = np.round(np.std(sample_female),2)
print(f"Population purchase standard deviation for Female : {pop_std_dev}")
print()
se = pop_std_dev/np.sqrt(500)
x1 =  (male_pop_mean - (z_score_90 * se)).round(2)
x2 =  (male_pop_mean + (z_score_90 * se)).round(2)
print(f"The 90% confidence interval --> $ {x1} to $ {x2}")
x1 =  (male_pop_mean - (z_score_95 * se)).round(2)
x2 =  (male_pop_mean + (z_score_95 * se)).round(2)
print(f"The 95% confidence interval --> $ {x1} to $ {x2}")
x1 =  (male_pop_mean - (z_score_99 * se)).round(2)
x2 =  (male_pop_mean + (z_score_99 * se)).round(2)
print(f"The 99% confidence interval --> $ {x1} to $ {x2}")
```

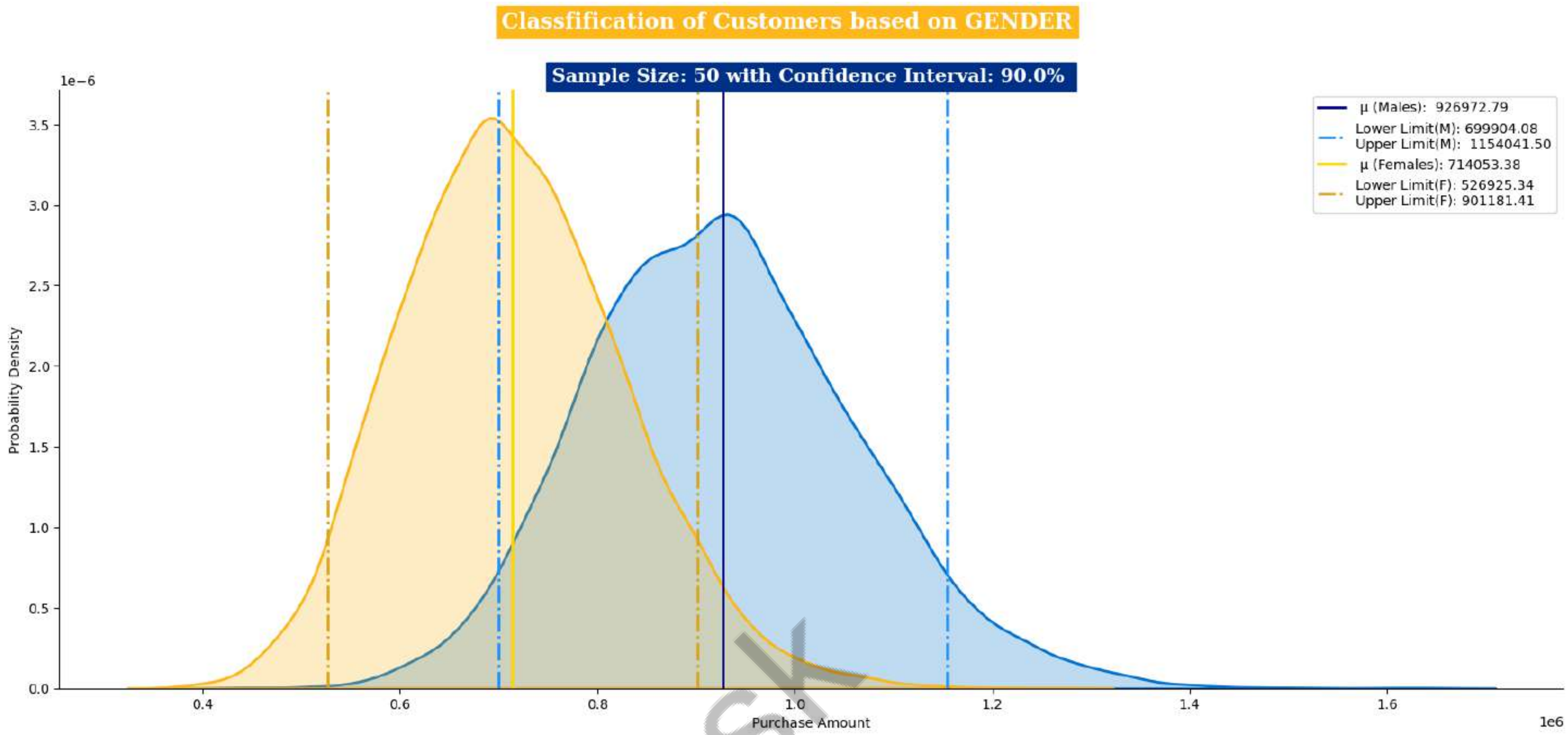


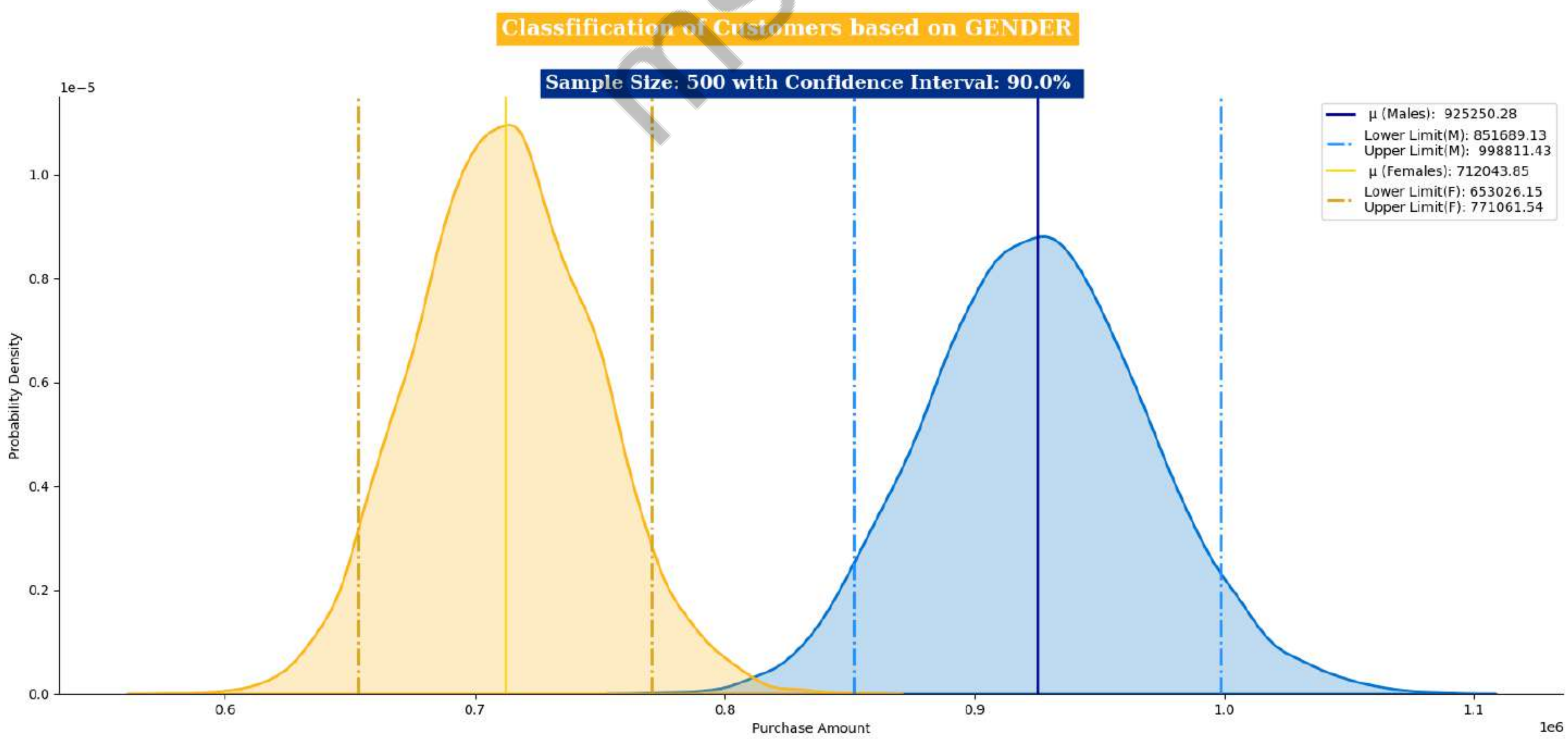
```
Female purchase amount - Confidence interval :
Population purchase mean for Female : 698817.74
Population purchase standard deviation for Female : 794086.24

The 90% confidence interval --> $ 653361.59 to $ 744273.89
The 95% confidence interval --> $ 640577.05 to $ 757058.43
The 99% confidence interval --> $ 616073.34 to $ 781562.14
```

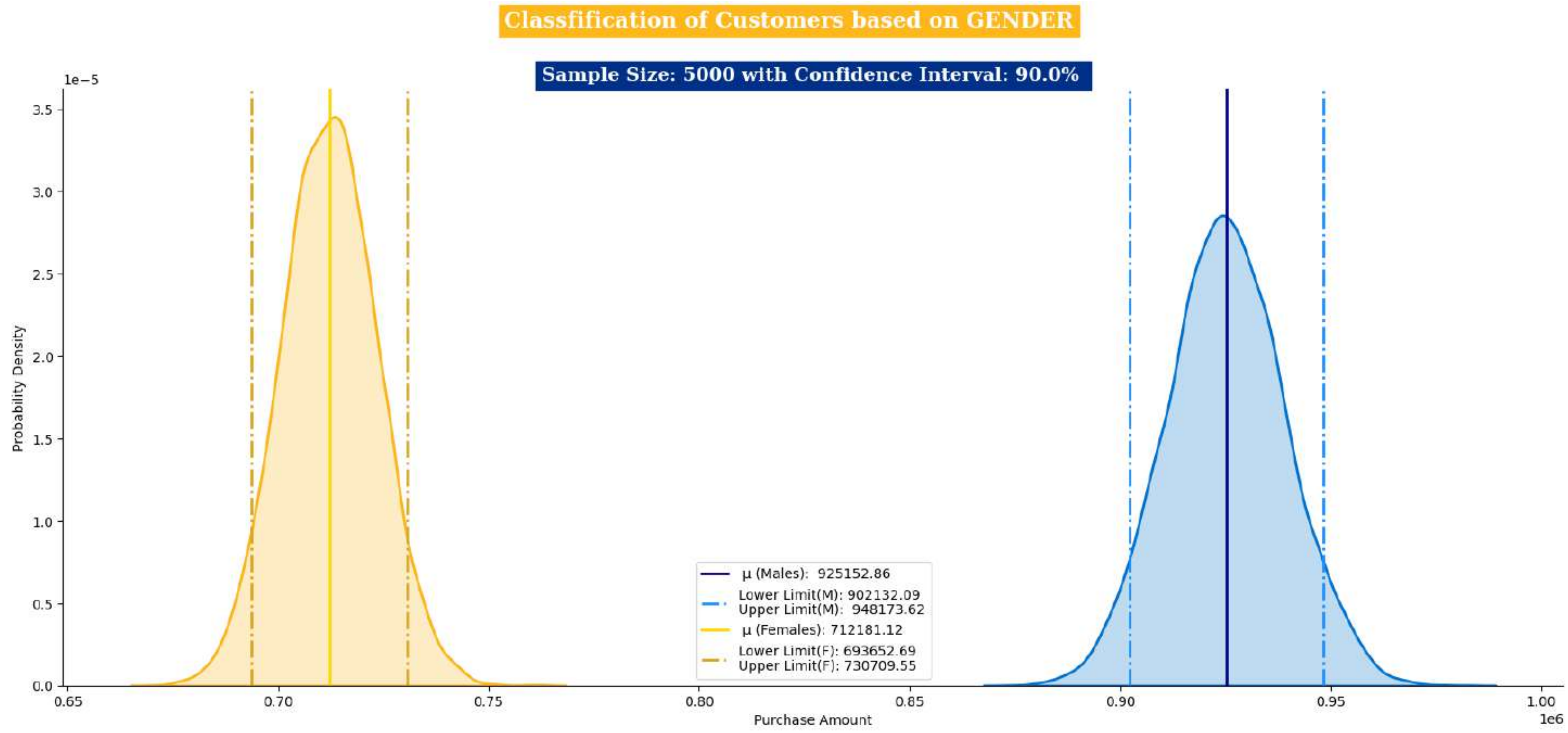
In [377…
```
sample_sizes = [50,500,5000,50000,100000]
ci = [90]
ntimes = 10000

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

for j in ci:
    for i in sample_sizes:
        m_avg, f_avg, ll_m, ul_m, ll_f, ul_f = bootstrapping('GENDER',male_purchase, female_purchase, i, ntimes, j)

        df.loc[len(df.index)] = ['M' , i , ll_m , ul_m , m_avg , j , (ul_m - ll_m)]
        df.loc[len(df.index)] = ['F' , i , ll_f , ul_f , f_avg , j , (ul_f - ll_f)]

df
```

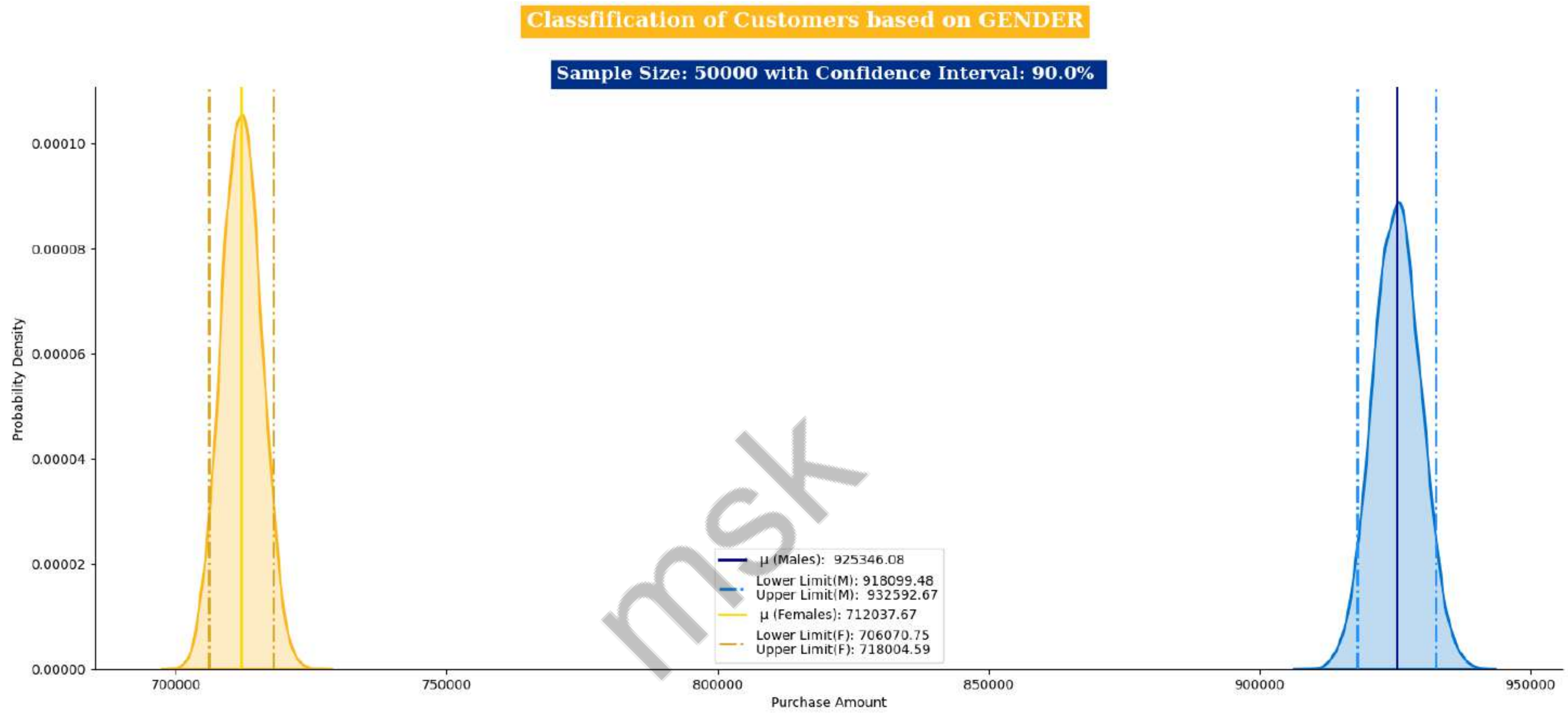
Out[377]:

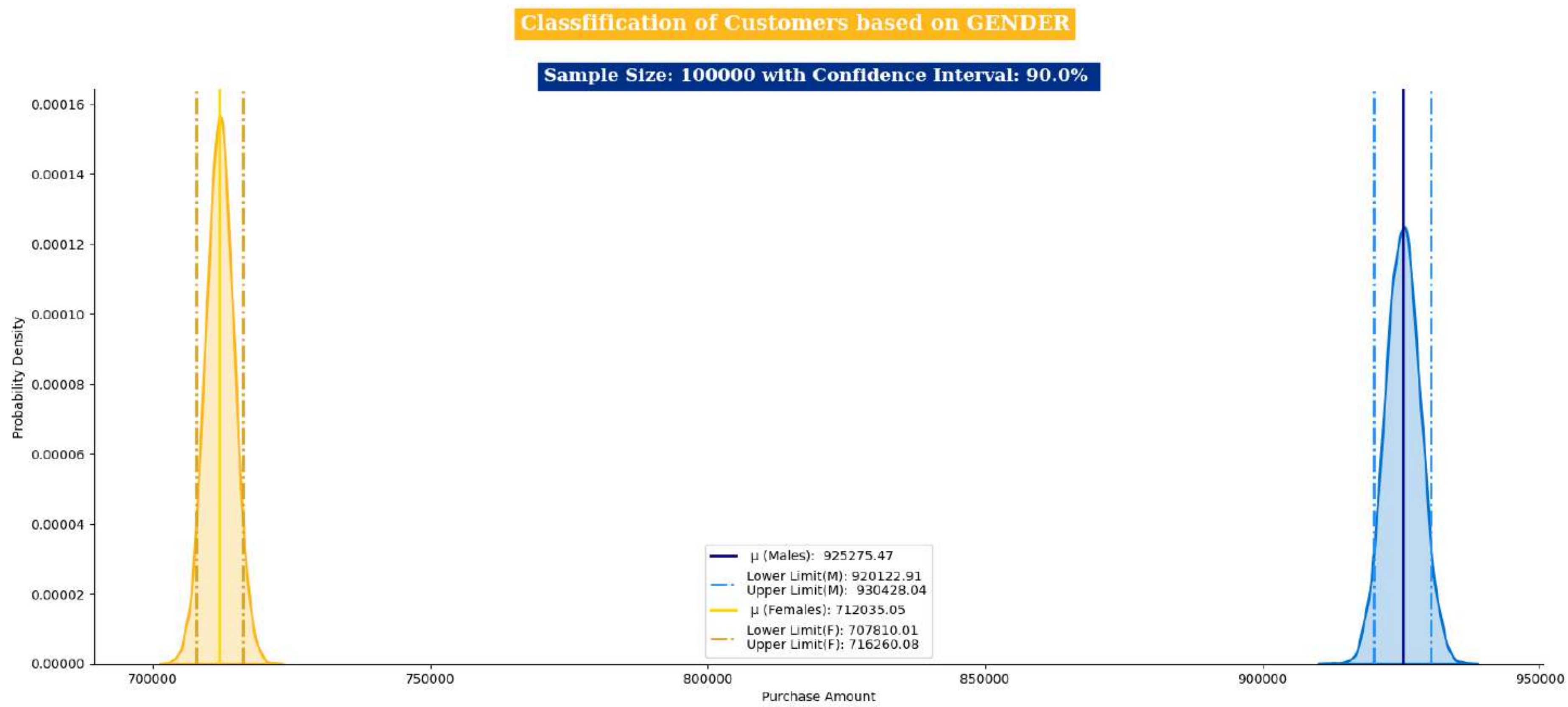
	Gender	Sample Size	Lower Limit	Upper Limit	Sample Mean	Confidence Interval	Range
0	M	50	699904.08	1154041.50	926972.79	90	454137.42
1	F	50	526925.34	901181.41	714053.38	90	374256.07
2	M	500	851689.13	998811.43	925250.28	90	147122.30
3	F	500	653026.15	771061.54	712043.85	90	118035.39
4	M	5000	902132.09	948173.62	925152.86	90	46041.53
5	F	5000	693652.69	730709.55	712181.12	90	37056.86
6	M	50000	918099.48	932592.67	925346.08	90	14493.19
7	F	50000	706070.75	718004.59	712037.67	90	11933.84
8	M	100000	920122.91	930428.04	925275.47	90	10305.13
9	F	100000	707810.01	716260.08	712035.05	90	8450.07











✅ 95% CI

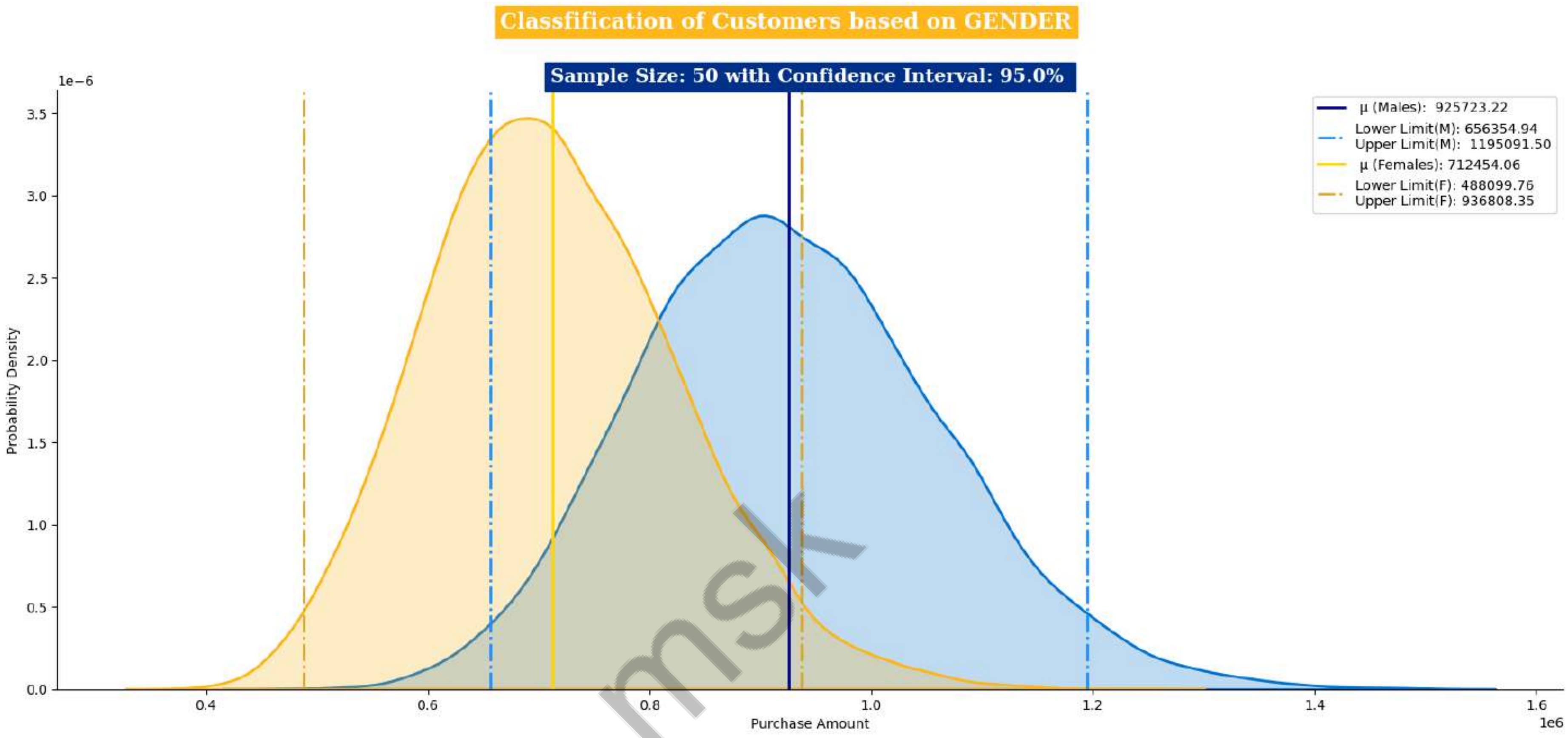
In [378…

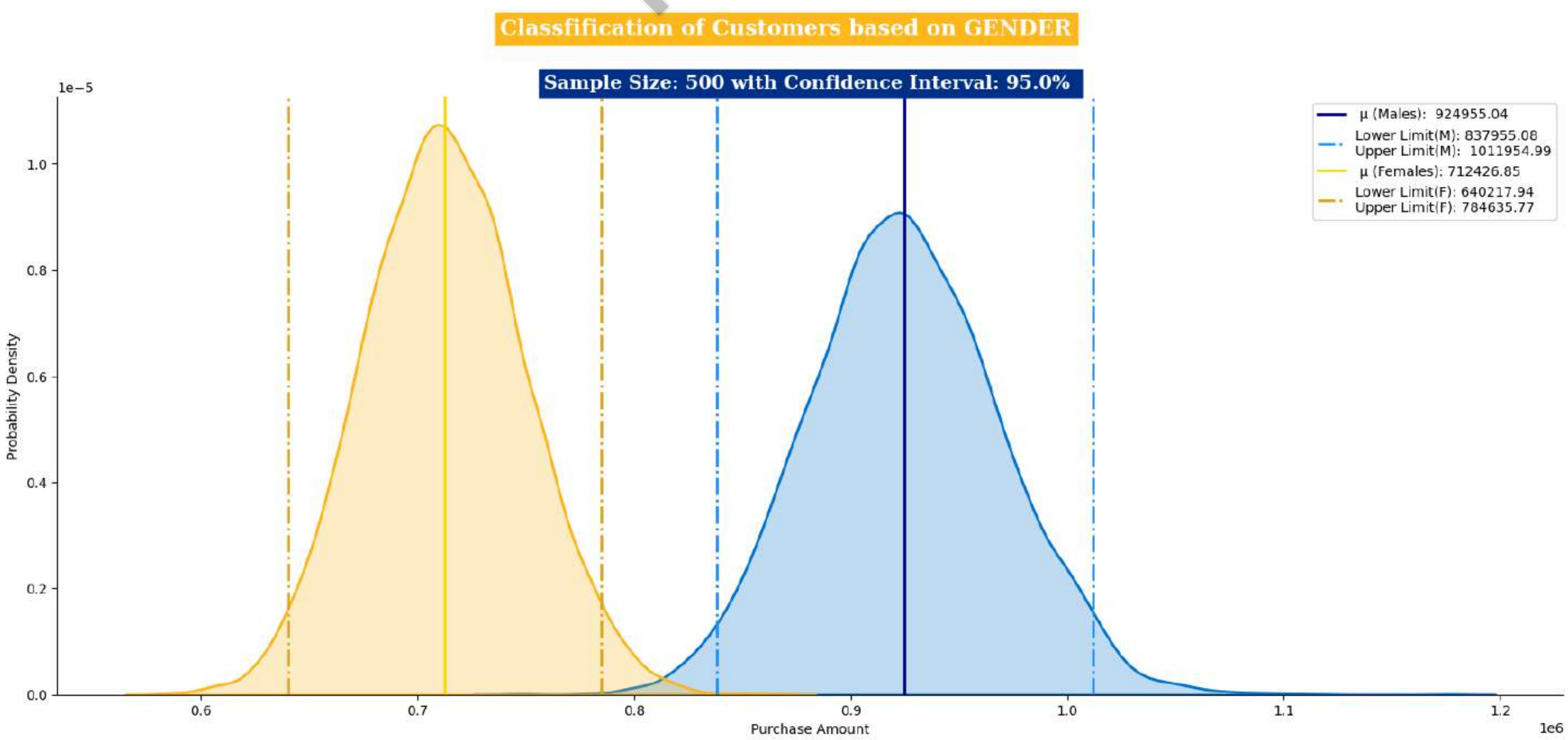
```
sample_sizes = [50,500,5000,50000,100000]
ci = [95]
ntimes = 10000

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

for j in ci:
    for i in sample_sizes:
        m_avg, f_avg, ll_m, ul_m, ll_f, ul_f = bootstrapping('GENDER',male_purchase, female_purchase, i, ntimes, j)
```

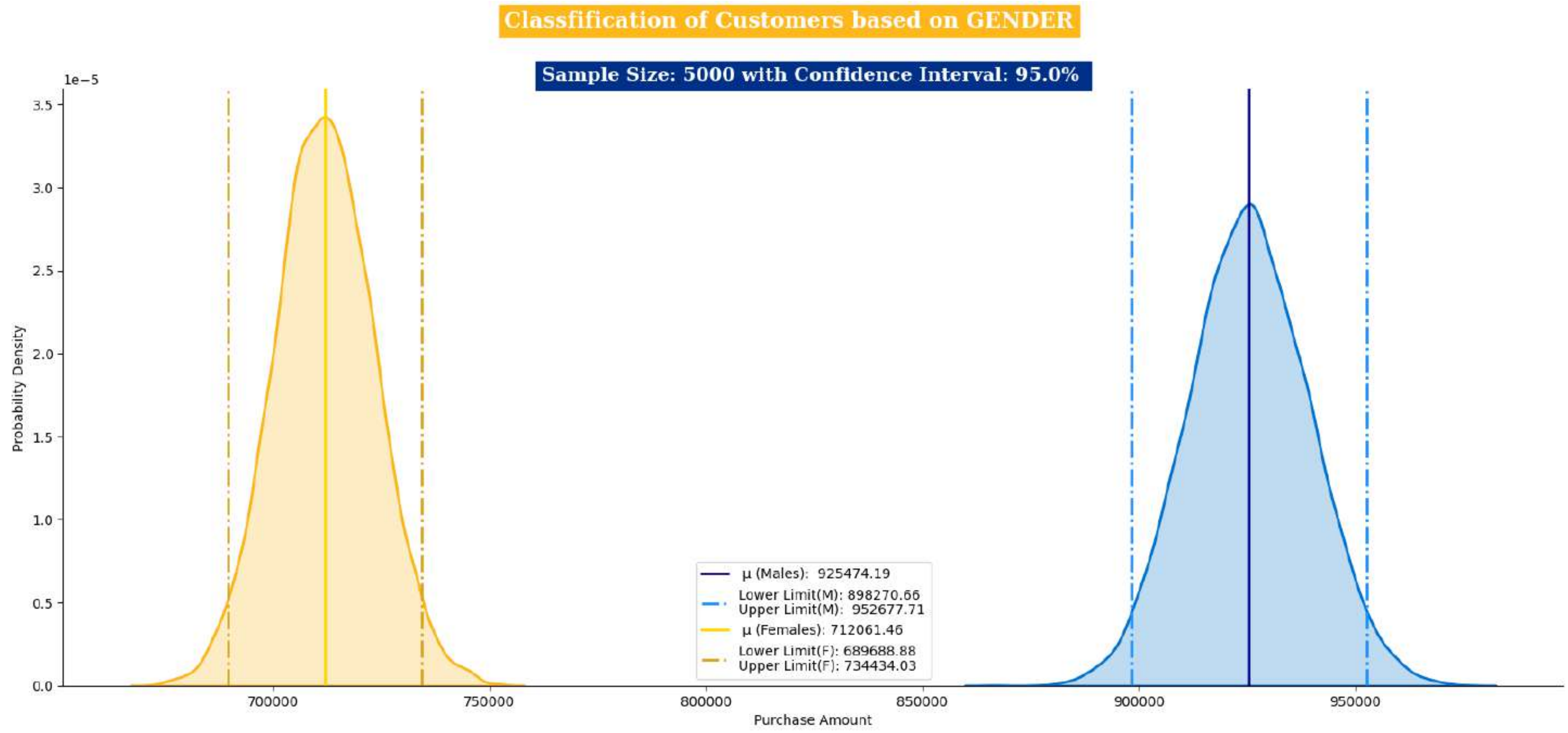
```
        df.loc[len(df.index)] = ['M' , i , ll_m , ul_m , m_avg , j , (ul_m - ll_m)]
        df.loc[len(df.index)] = ['F' , i , ll_f , ul_f , f_avg , j , (ul_f - ll_f)]

df
```

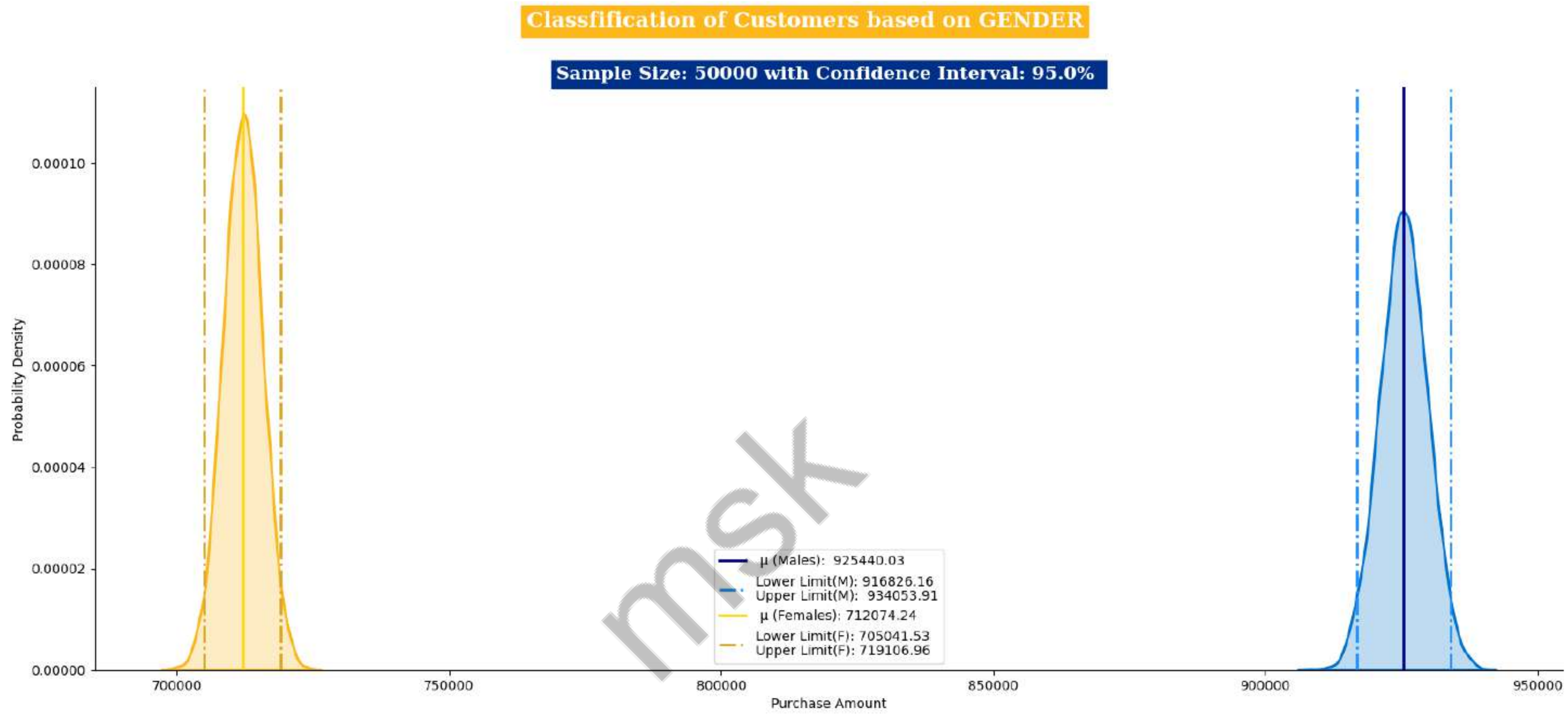
Out[378]:

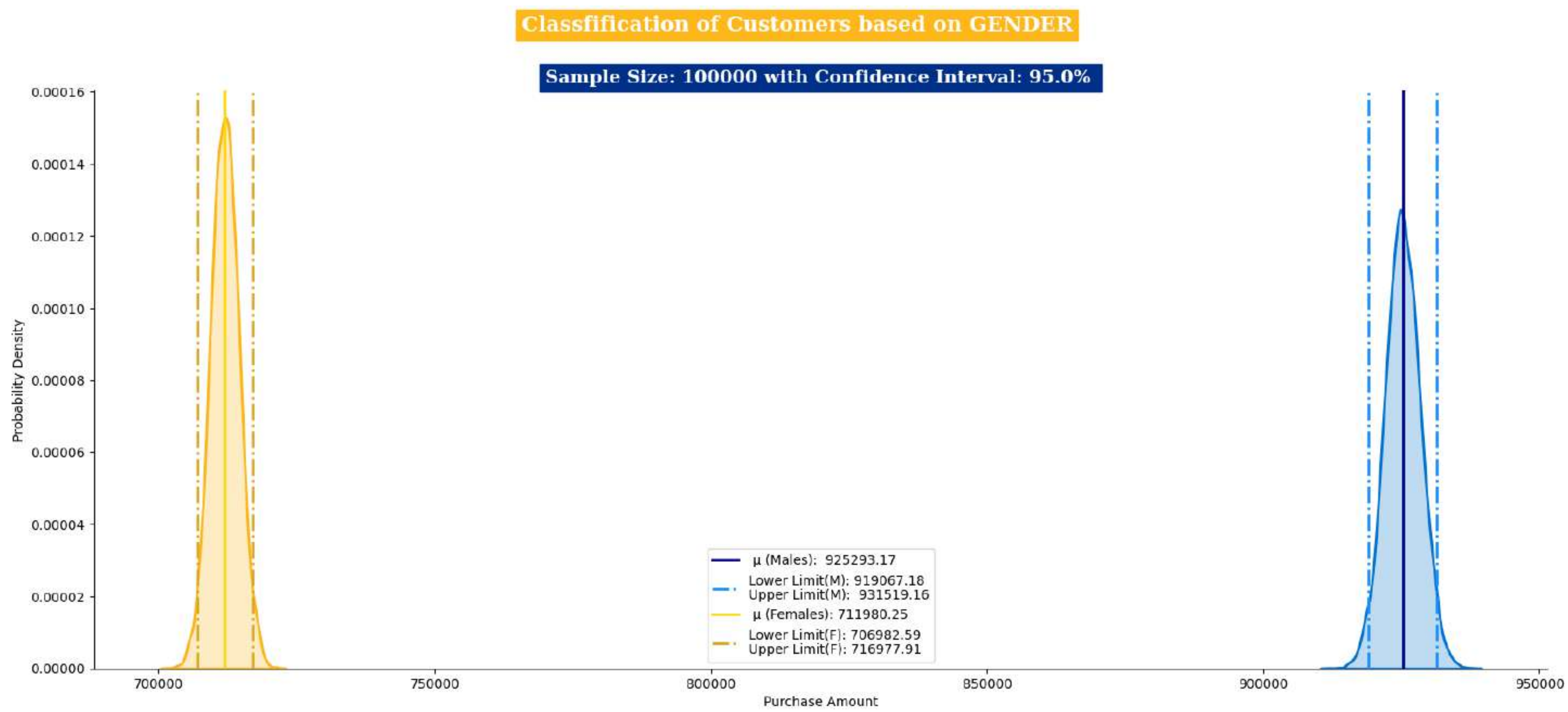
	Gender	Sample Size	Lower Limit	Upper Limit	Sample Mean	Confidence Interval	Range
0	M	50	656354.94	1195091.50	925723.22	95	538736.56
1	F	50	488099.76	936808.35	712454.06	95	448708.59
2	M	500	837955.08	1011954.99	924955.04	95	173999.91
3	F	500	640217.94	784635.77	712426.85	95	144417.83
4	M	5000	898270.66	952677.71	925474.19	95	54407.05
5	F	5000	689688.88	734434.03	712061.46	95	44745.15
6	M	50000	916826.16	934053.91	925440.03	95	17227.75
7	F	50000	705041.53	719106.96	712074.24	95	14065.43
8	M	100000	919067.18	931519.16	925293.17	95	12451.98
9	F	100000	706982.59	716977.91	711980.25	95	9995.32











✅ 99% CI

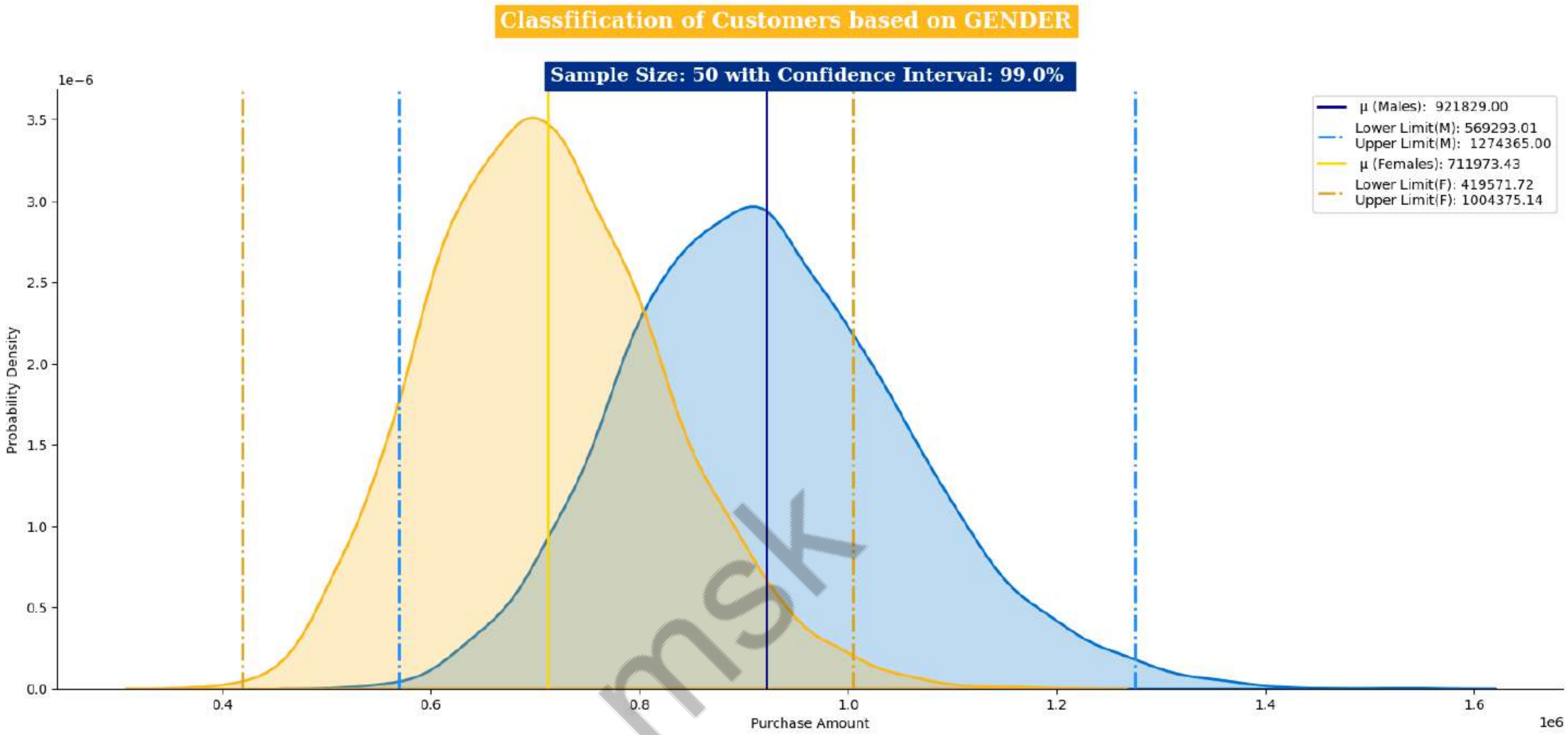
In [379…

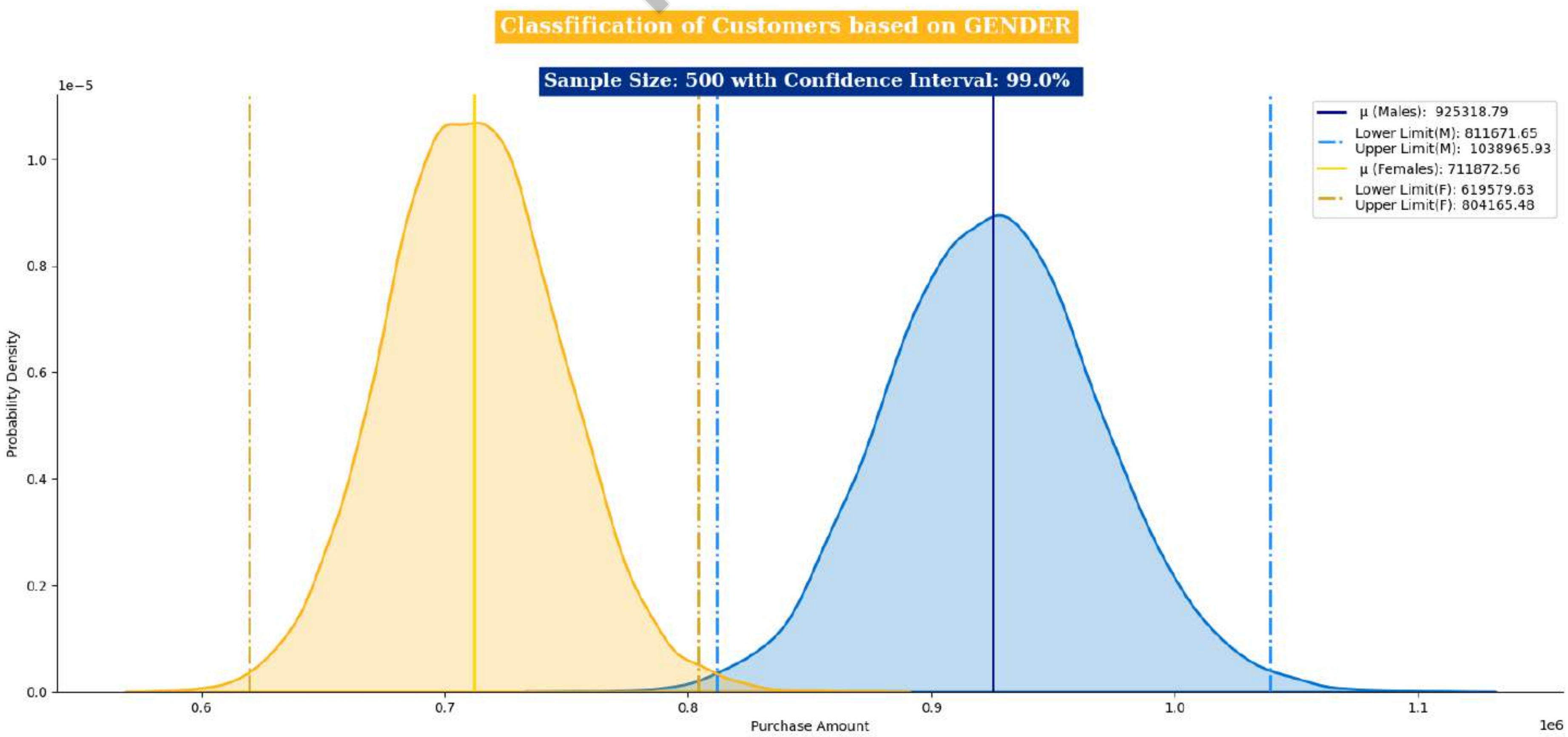
```
sample_sizes = [50,500,5000,50000,100000]
ci = [99]
ntimes = 10000

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

for j in ci:
    for i in sample_sizes:
        m_avg, f_avg, ll_m, ul_m, ll_f, ul_f = bootstrapping('GENDER',male_purchase, female_purchase, i, ntimes, j)
```

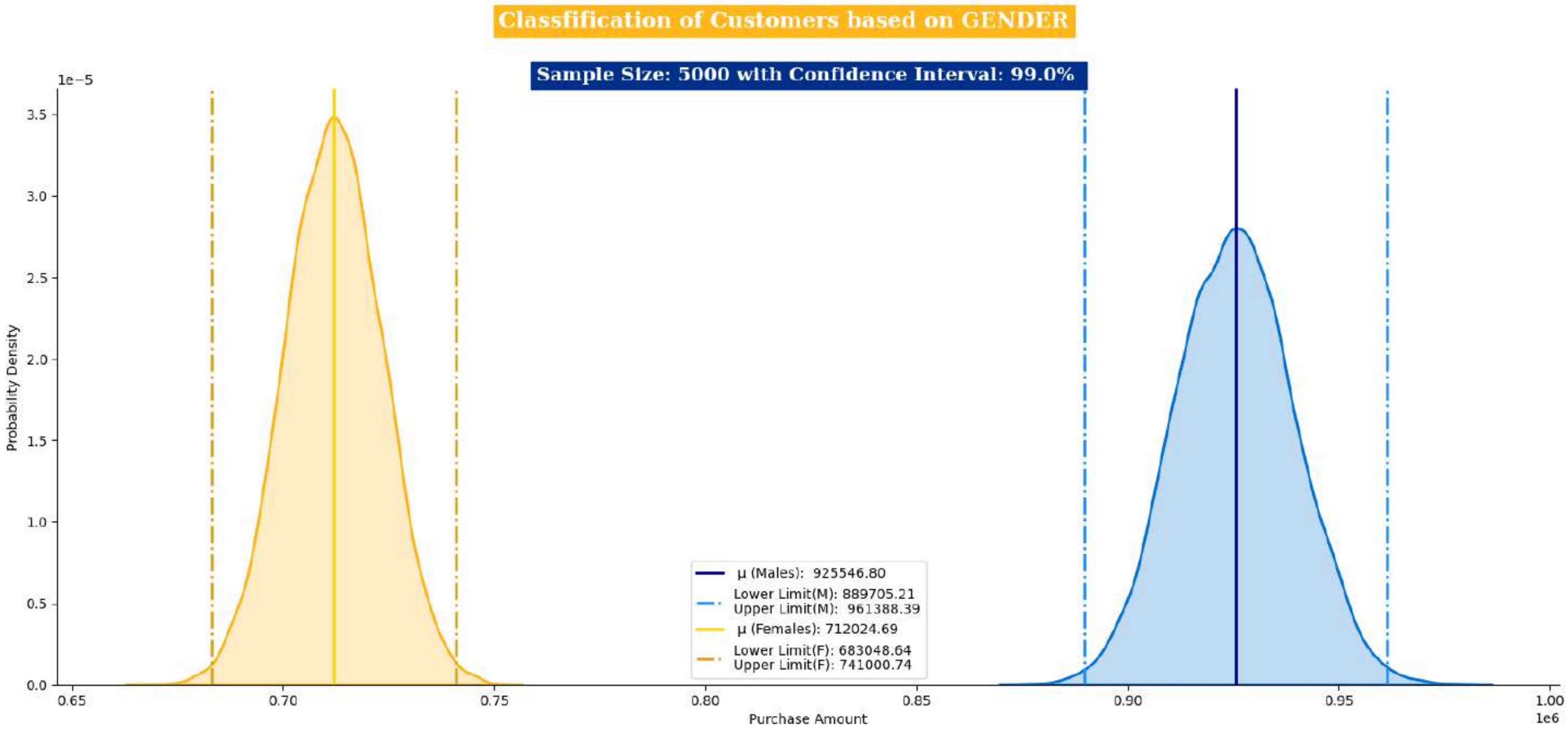
```
        df.loc[len(df.index)] = ['M' , i , ll_m , ul_m , m_avg , j , (ul_m - ll_m)]
        df.loc[len(df.index)] = ['F' , i , ll_f , ul_f , f_avg , j , (ul_f - ll_f)]

df
```

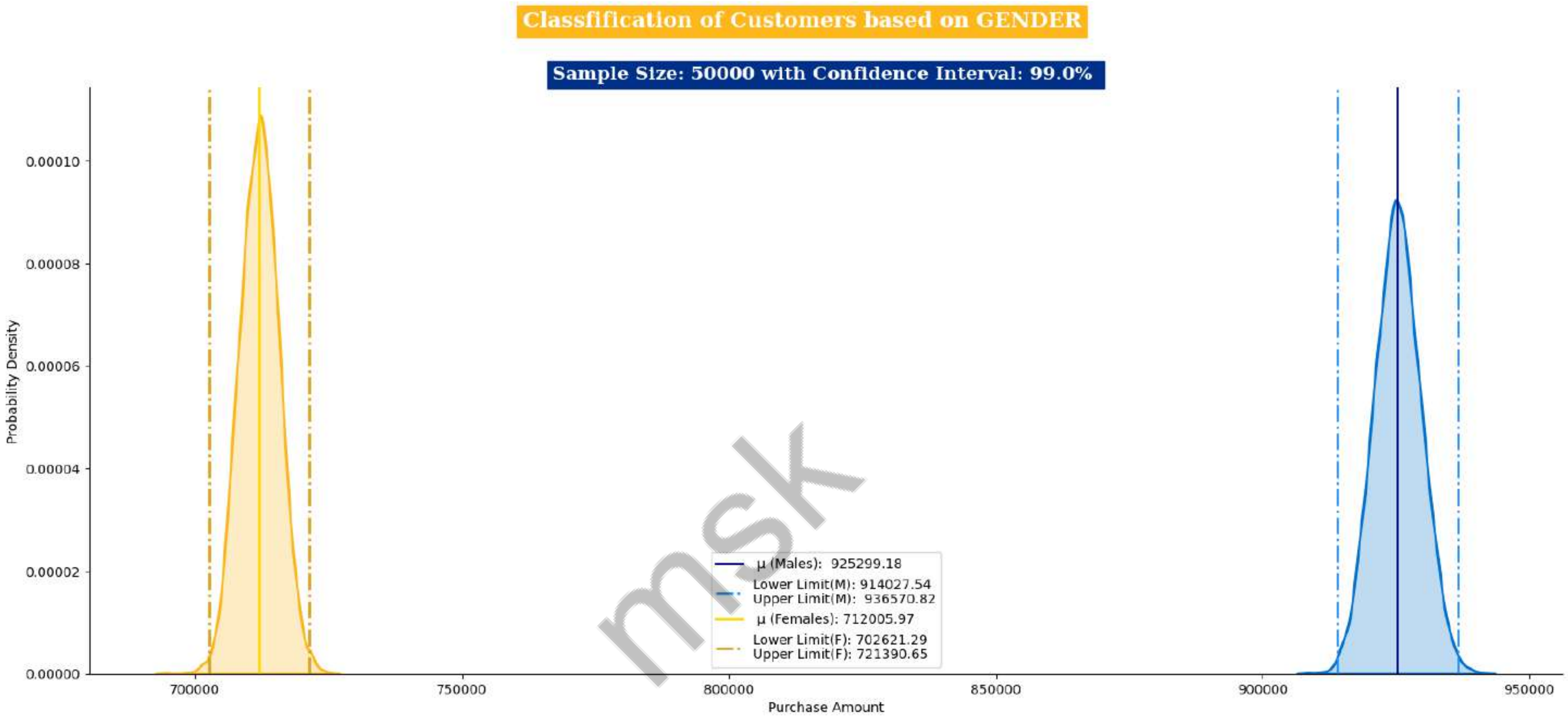
Out[379]:

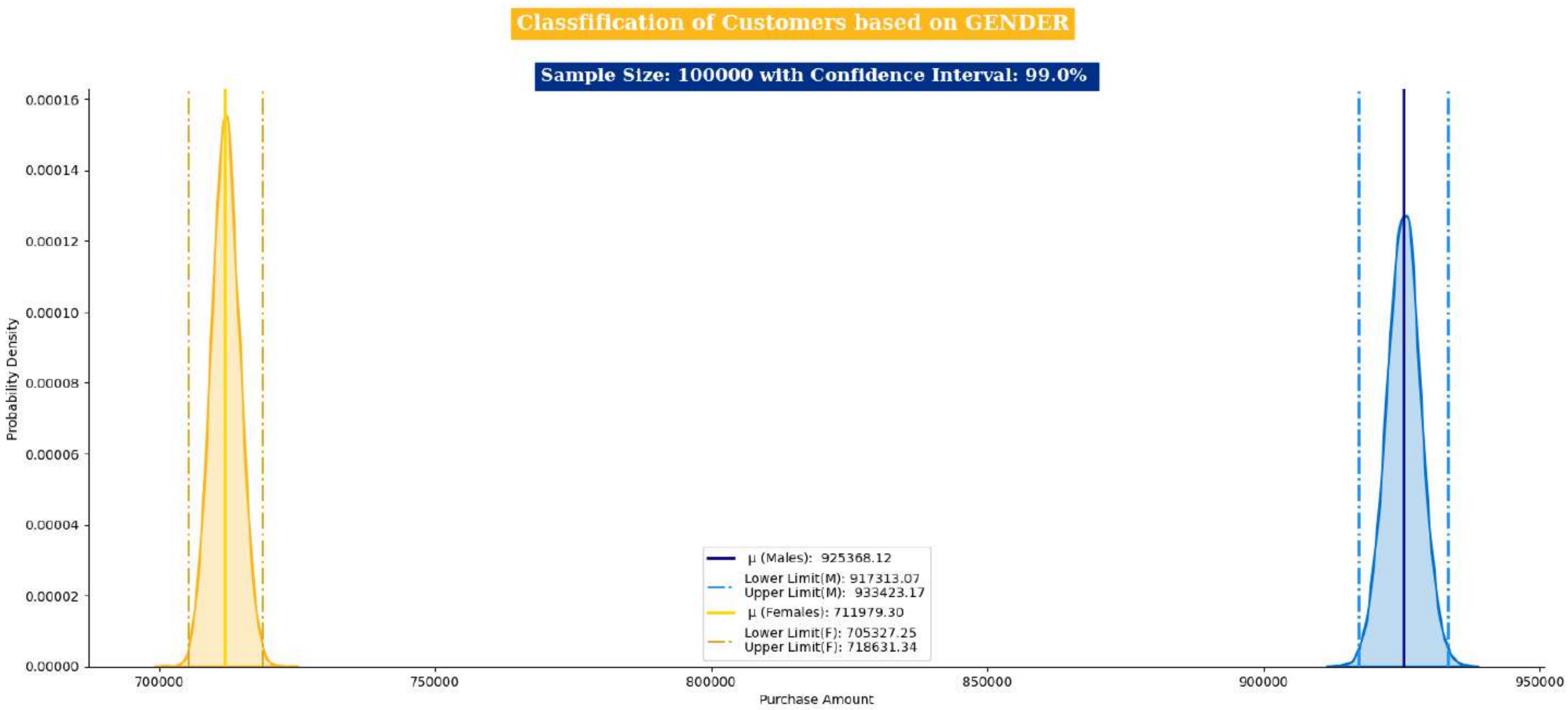
	Gender	Sample Size	Lower Limit	Upper Limit	Sample Mean	Confidence Interval	Range
0	M	50	569293.01	1274365.00	921829.00	99	705071.99
1	F	50	419571.72	1004375.14	711973.43	99	584803.42
2	M	500	811671.65	1038965.93	925318.79	99	227294.28
3	F	500	619579.63	804165.48	711872.56	99	184585.85
4	M	5000	889705.21	961388.39	925546.80	99	71683.18
5	F	5000	683048.64	741000.74	712024.69	99	57952.10
6	M	50000	914027.54	936570.82	925299.18	99	22543.28
7	F	50000	702621.29	721390.65	712005.97	99	18769.36
8	M	100000	917313.07	933423.17	925368.12	99	16110.10
9	F	100000	705327.25	718631.34	711979.30	99	13304.09











🏷️ Insights

- 🧐 ***Observation***
  - The average for both of them changes significantly as the sample size increases:
    - As the **sample size** `increases`, the average values for both genders undergo noticeable changes.
    - **Larger sample sizes** tend to provide *more representative insights* into the population, leading to `more stable` and reliable average values.

- Both plots start to separate and become distinct:
  - With *increasing sample size*, the plots representing the data for males and females start to `diverge` and show distinct patterns.
  - This separation could indicate that the **larger sample sizes** are capturing `more nuances` in the data, revealing differences between males and females that might not be as apparent in smaller samples.

👉 **For *Sample size 50*, The *confidence interval [90%,95%,99%]* 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** `DON'T OVERLAP` **.**

▶️ 〰️ ◀️ Confidence intervals and distribution of the mean of the expenses based on customers Marital_Status

In [370…

```
wm.sample()
```

Out[370]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
32599	1005003	P00249542	M	36-45	12	C	1	Single	1	15696

In [412…

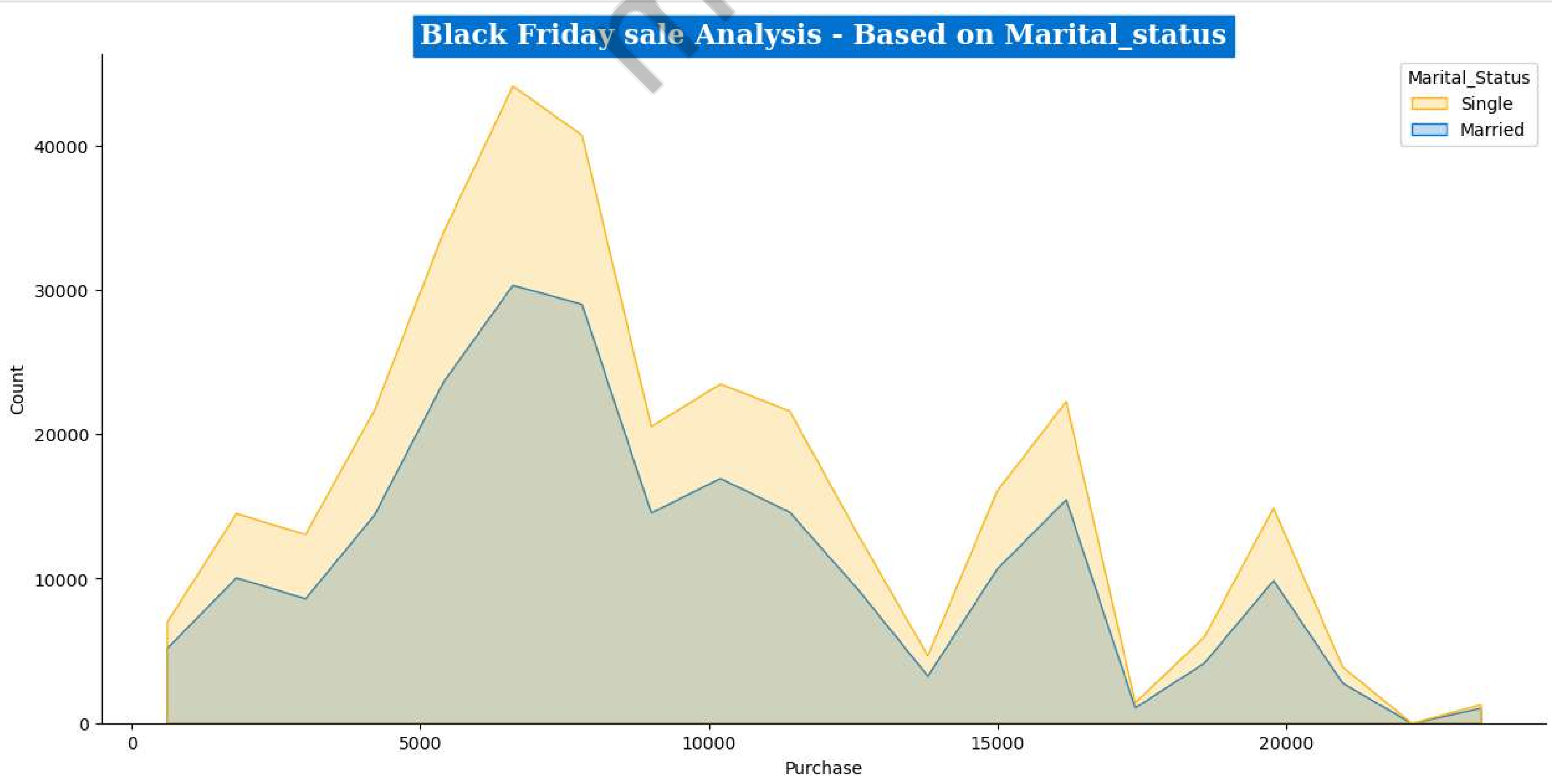
```
wm.groupby(['Marital_Status'])['Purchase'].describe().T
```

Out[412]:

Marital_Status	Single	Married
count	324731.000000	225337.000000
mean	9265.907619	9261.174574
std	5027.347859	5016.897378
min	12.000000	12.000000
25%	5605.000000	5843.000000
50%	8044.000000	8051.000000
75%	12061.000000	12042.000000
max	23961.000000	23961.000000

In [394…

```
plt.figure(figsize=(15,7))
sns.histplot(data=wm, x = "Purchase", bins=20, hue = "Marital_Status",element='poly',palette=cp2[::-1])
sns.despine()
plt.title('Black Friday sale Analysis - Based on Marital_status',fontfamily='serif',fontweight='bold',
          fontsize=16,backgroundcolor=cp2[0],color='w')
plt.show()
```

In [388…

```
wm_married_cust = wm[wm['Marital_Status'] == 'Married']['Purchase']
wm_single_cust = wm[wm['Marital_Status'] == 'Single']['Purchase']
```

In [531…

```
sample_male = wm_married_cust.sample(500)
z_score_90 = norm.ppf(0.9).round(2)
z_score_95 = norm.ppf(0.95).round(2)
z_score_99 = norm.ppf(0.99).round(2)
print(f"Married purchase amount - Confidence interval :")
male_pop_mean = np.mean(sample_male).round(2)
print(f"Population purchase mean for Married Customers : {male_pop_mean}")
pop_std_dev = np.round(np.std(sample_male),2)
print(f"Population purchase standard deviation for Married Customers : {pop_std_dev}")
print()
se = pop_std_dev/np.sqrt(500)
```

```
x1 =  (male_pop_mean - (z_score_90 * se)).round(2)
x2 =  (male_pop_mean + (z_score_90 * se)).round(2)
print(f"The 90% confidence interval --> $ {x1} to $ {x2}")
x1 =  (male_pop_mean - (z_score_95 * se)).round(2)
x2 =  (male_pop_mean + (z_score_95 * se)).round(2)
print(f"The 95% confidence interval --> $ {x1} to $ {x2}")
x1 =  (male_pop_mean - (z_score_99 * se)).round(2)
x2 =  (male_pop_mean + (z_score_99 * se)).round(2)
print(f"The 99% confidence interval --> $ {x1} to $ {x2}")
```

```
Married purchase amount - Confidence interval :
Population purchase mean for Married Customers : 9450.15
Population purchase standard deviation for Married Customers : 5288.46

The 90% confidence interval --> $ 9147.42 to $ 9752.88
The 95% confidence interval --> $ 9062.28 to $ 9838.02
The 99% confidence interval --> $ 8899.09 to $ 10001.21
```

In [532…

```
sample_male = wm_single_cust.sample(500)
z_score_90 = norm.ppf(0.9).round(2)
z_score_95 = norm.ppf(0.95).round(2)
z_score_99 = norm.ppf(0.99).round(2)
print(f"UnMarried purchase amount - Confidence interval :")
male_pop_mean = np.mean(sample_male).round(2)
print(f"Population purchase mean for single Customers : {male_pop_mean}")
pop_std_dev = np.round(np.std(sample_male),2)
print(f"Population purchase standard deviation for single Customers : {pop_std_dev}")
print()
se = pop_std_dev/np.sqrt(500)
x1 =  (male_pop_mean - (z_score_90 * se)).round(2)
x2 =  (male_pop_mean + (z_score_90 * se)).round(2)
print(f"The 90% confidence interval --> $ {x1} to $ {x2}")
x1 =  (male_pop_mean - (z_score_95 * se)).round(2)
x2 =  (male_pop_mean + (z_score_95 * se)).round(2)
print(f"The 95% confidence interval --> $ {x1} to $ {x2}")
x1 =  (male_pop_mean - (z_score_99 * se)).round(2)
x2 =  (male_pop_mean + (z_score_99 * se)).round(2)
print(f"The 99% confidence interval --> $ {x1} to $ {x2}")
```

```
UnMarried purchase amount - Confidence interval :
Population purchase mean for single Customers : 9314.22
Population purchase standard deviation for single Customers : 5118.2

The 90% confidence interval --> $ 9021.24 to $ 9607.2
The 95% confidence interval --> $ 8938.84 to $ 9689.6
The 99% confidence interval --> $ 8780.9 to $ 9847.54
```


SINGLE & MARRIED

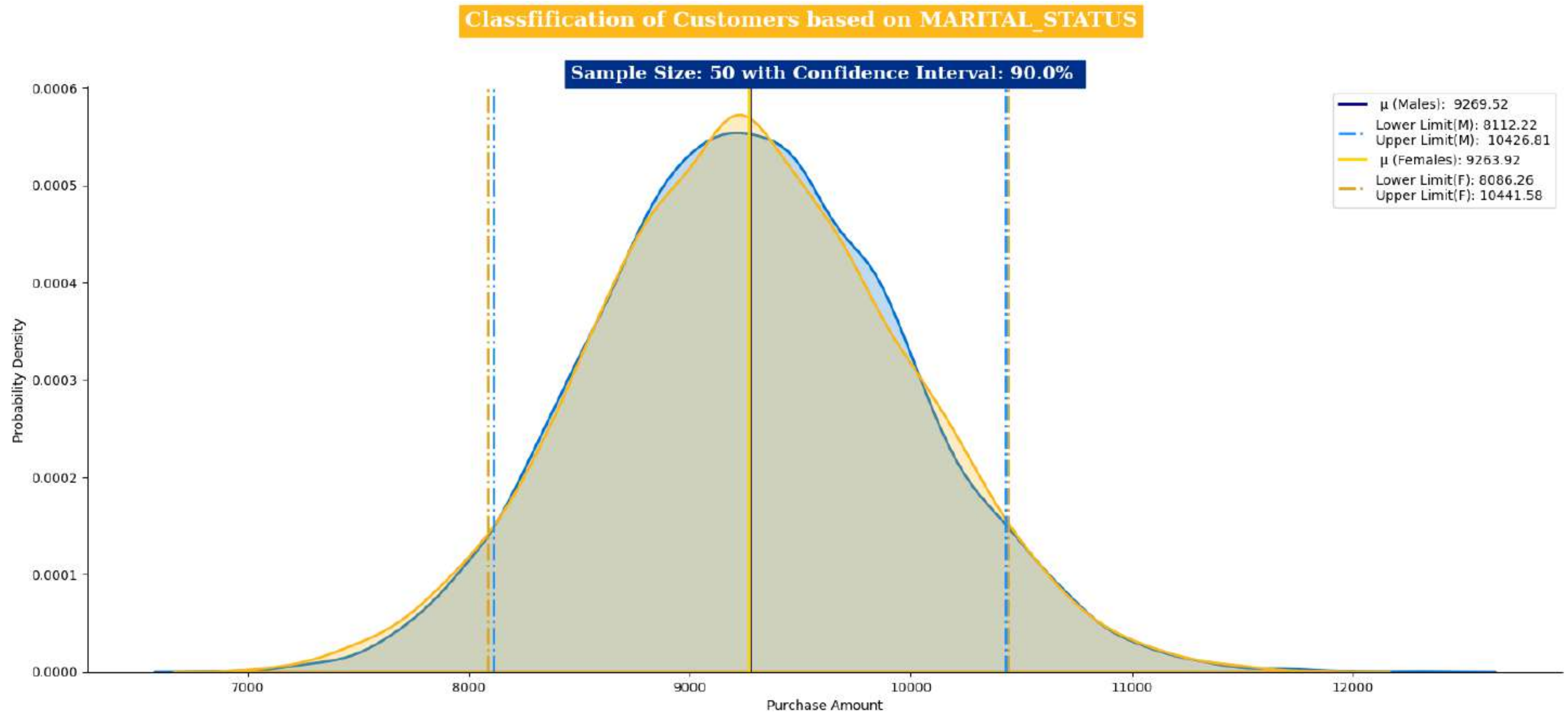
✅ 90% CI

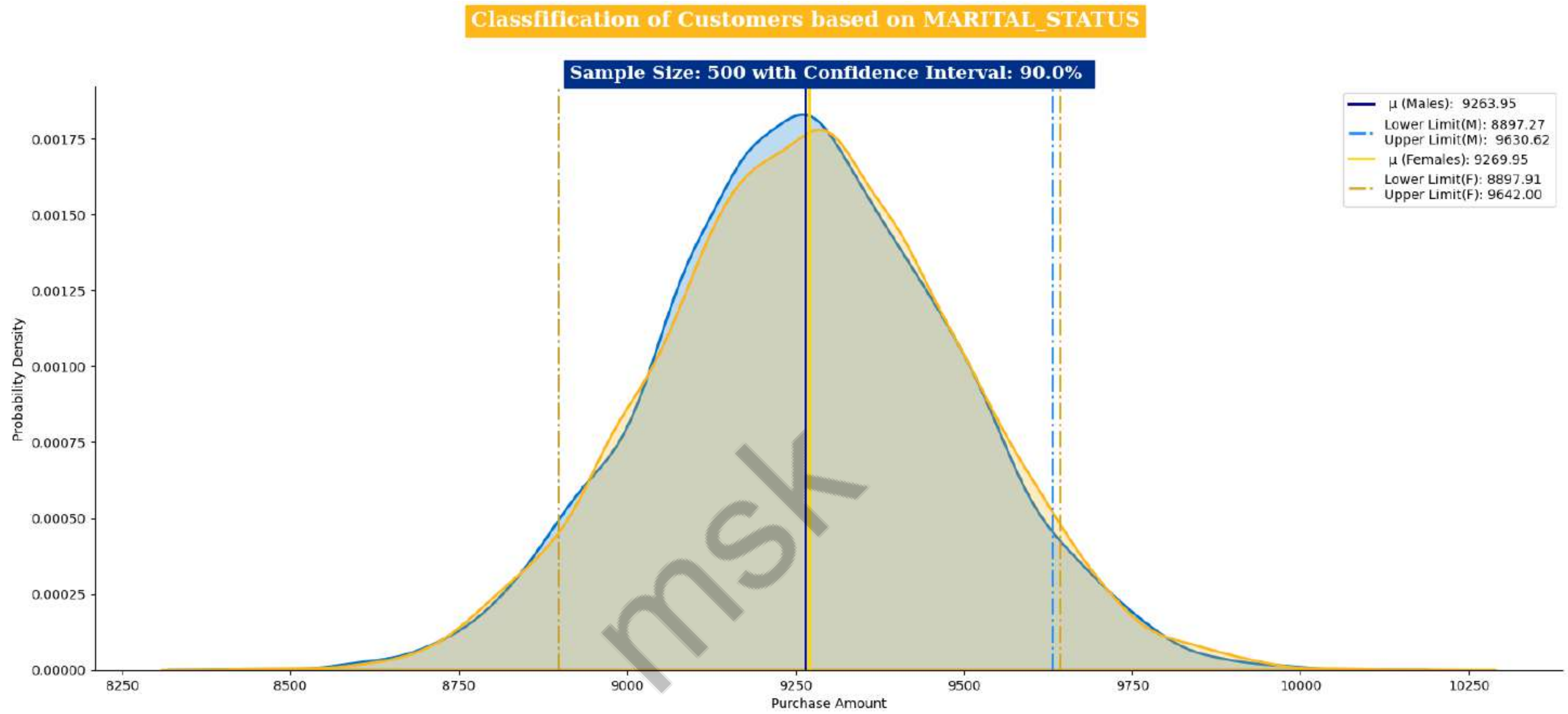
In [389…

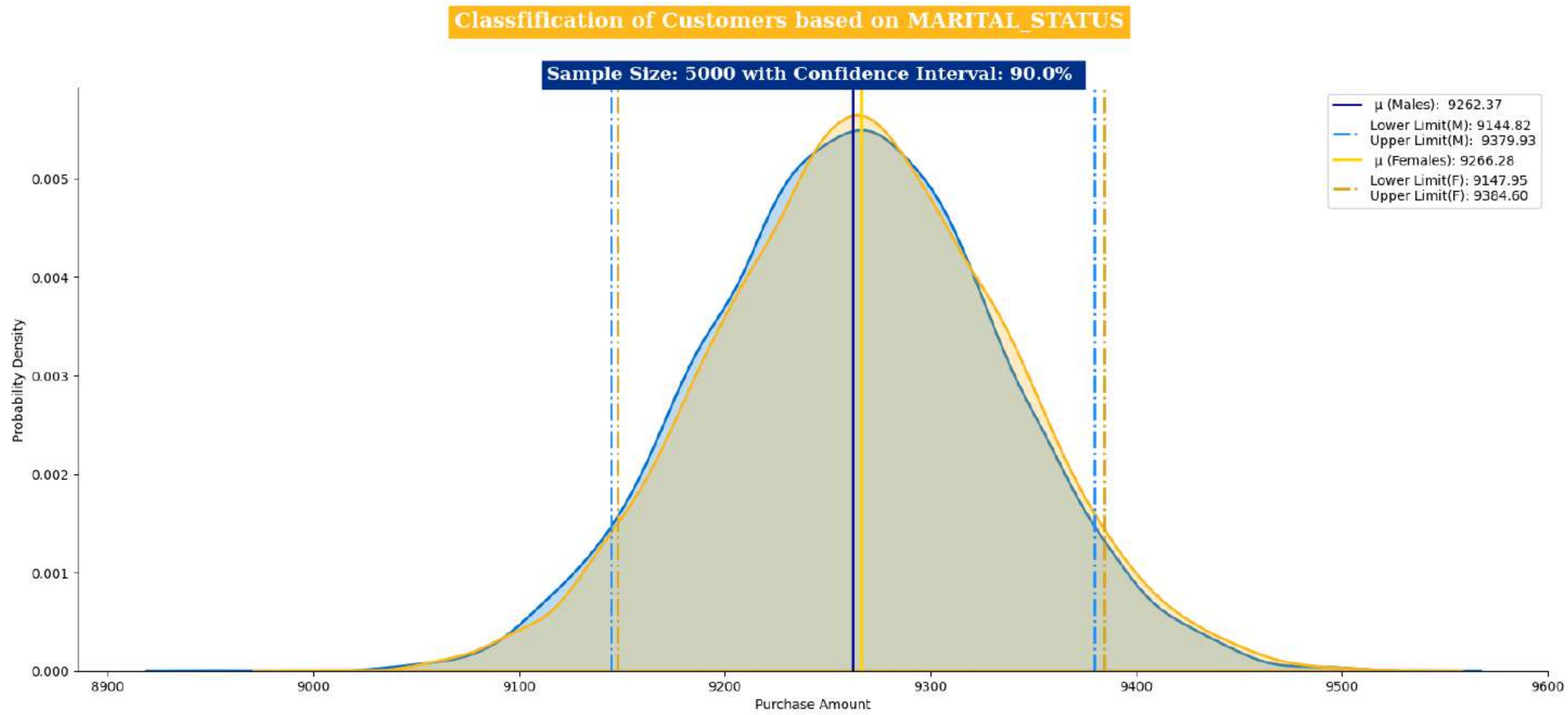
```
sample_sizes = [50,500,5000,50000,100000]
ci = [90]
ntimes = 10000

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

for j in ci:
    for i in sample_sizes:
        m_avg, s_avg, ll_m, ul_m, ll_s, ul_s = bootstrapping('MARITAL_STATUS',wm_married_cust, wm_single_cust, i, ntimes, j)

        df.loc[len(df.index)] = ['M' , i , ll_m , ul_m , m_avg , j , (ul_m - ll_m)]
        df.loc[len(df.index)] = ['F' , i , ll_s , ul_s , s_avg , j , (ul_s - ll_s)]

df
```

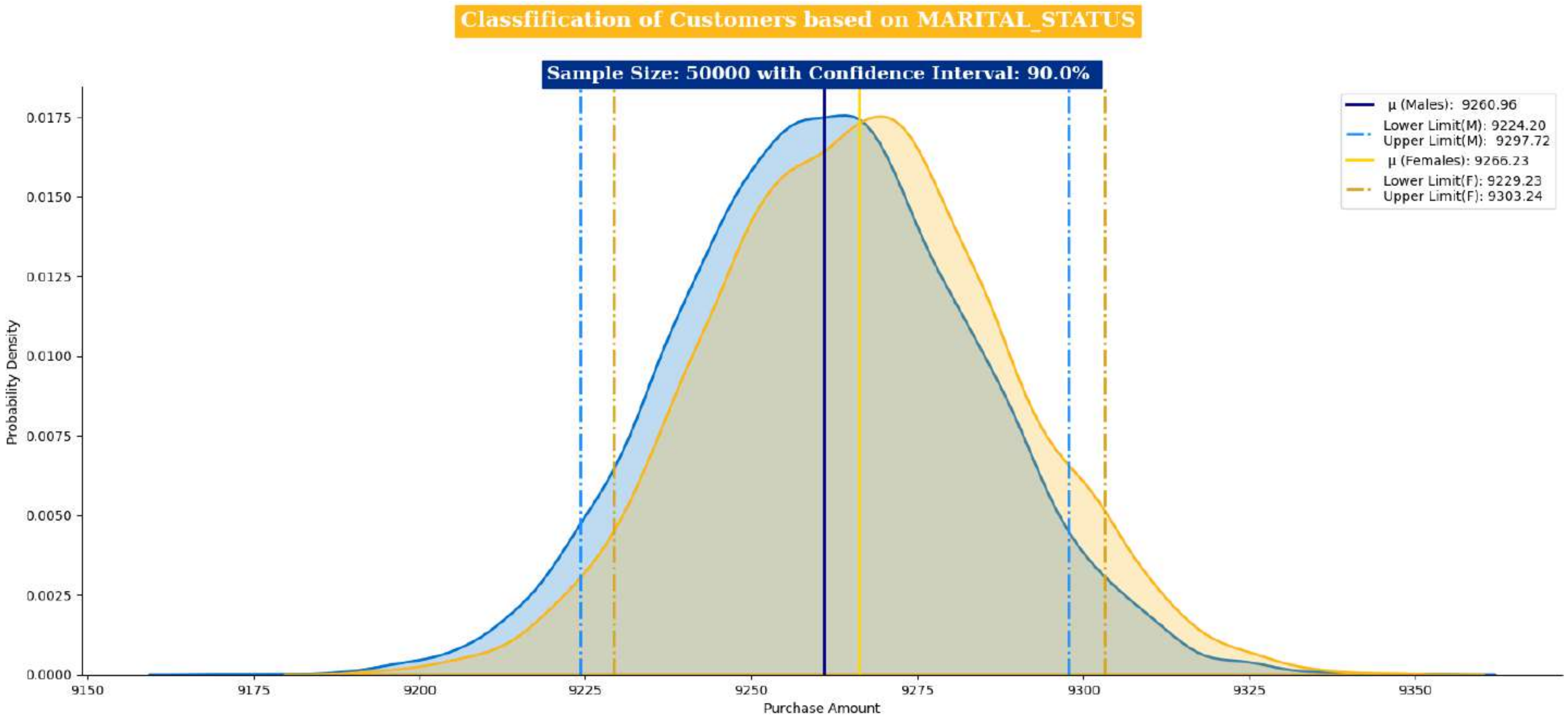
Out[389]:

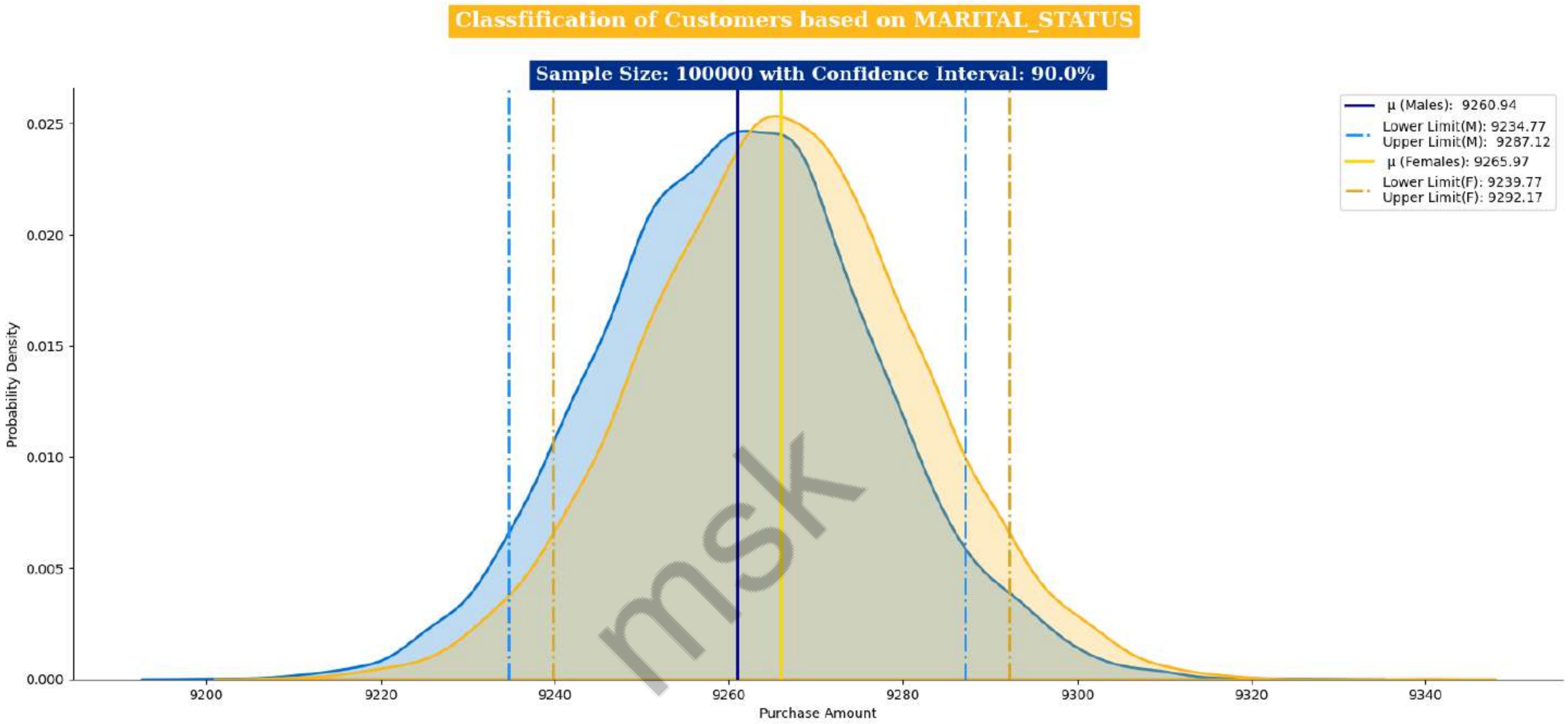
	Gender	Sample Size	Lower Limit	Upper Limit	Sample Mean	Confidence Interval	Range
0	M	50	8112.22	10426.81	9269.52	90	2314.59
1	F	50	8086.26	10441.58	9263.92	90	2355.32
2	M	500	8897.27	9630.62	9263.95	90	733.35
3	F	500	8897.91	9642.00	9269.95	90	744.09
4	M	5000	9144.82	9379.93	9262.37	90	235.11
5	F	5000	9147.95	9384.60	9266.28	90	236.65
6	M	50000	9224.20	9297.72	9260.96	90	73.52
7	F	50000	9229.23	9303.24	9266.23	90	74.01
8	M	100000	9234.77	9287.12	9260.94	90	52.35
9	F	100000	9239.77	9292.17	9265.97	90	52.40


Classfification of Customers based on MARITAL_STATUS
Sample Size: 50 with Confidence Interval: 90.0%
μ (Males): 9269.52
Lower Limit(M): 8112.22
Upper Limit(M): 10426.81
μ (Females): 9263.92
Lower Limit(F): 8086.26
Upper Limit(F): 10441.58
Probability Density
Purchase Amount


Classfification of Customers based on MARITAL_STATUS
Sample Size: 500 with Confidence Interval: 90.0%
μ (Males): 9263.95
Lower Limit(M): 8897.27
Upper Limit(M): 9630.62
μ (Females): 9269.95
Lower Limit(F): 8897.91
Upper Limit(F): 9642.00
Probability Density
Purchase Amount


Classfification of Customers based on MARITAL_STATUS
Sample Size: 5000 with Confidence Interval: 90.0%
μ (Males): 9262.37
Lower Limit(M): 9144.82
Upper Limit(M): 9379.93
μ (Females): 9266.28
Lower Limit(F): 9147.95
Upper Limit(F): 9384.60
Probability Density
Purchase Amount





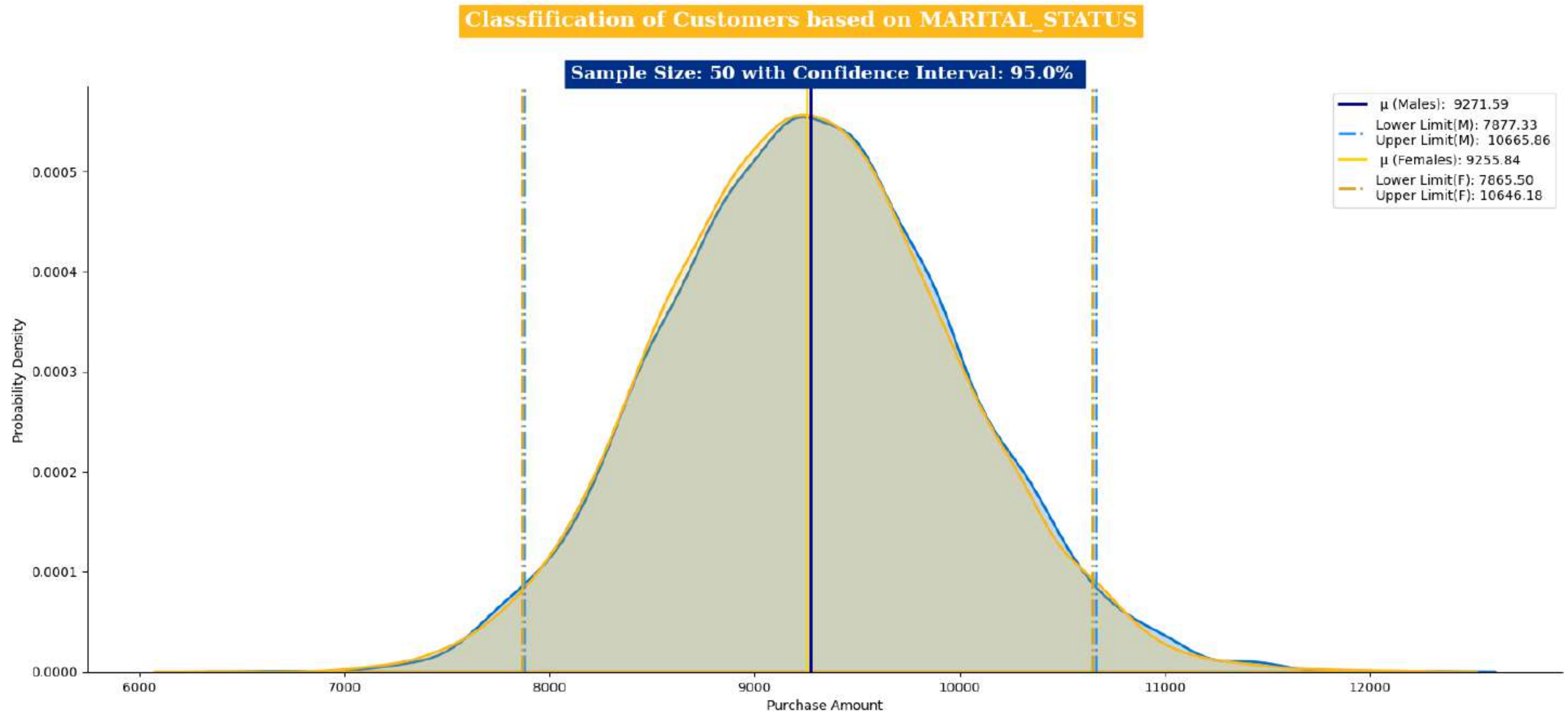
✅ 95% CI

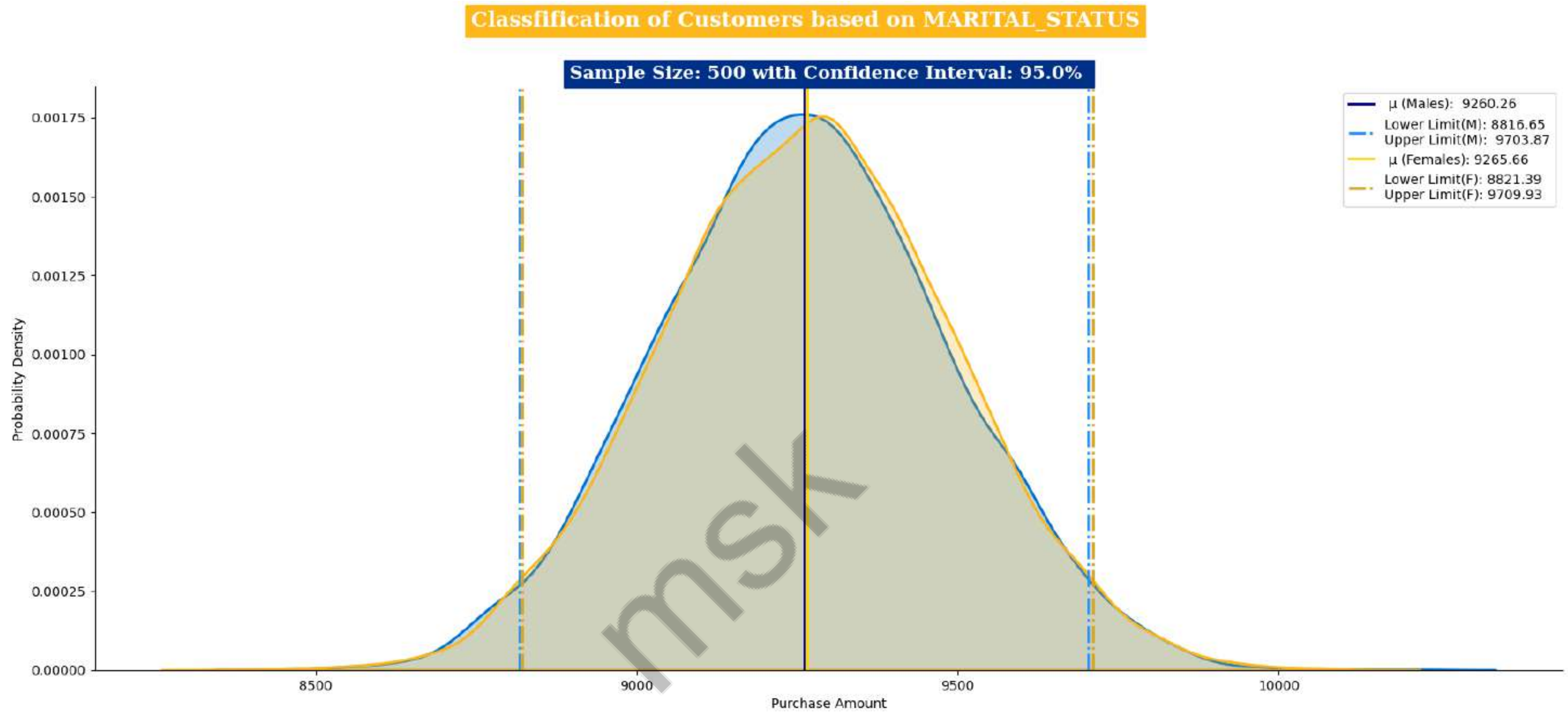
In [390…

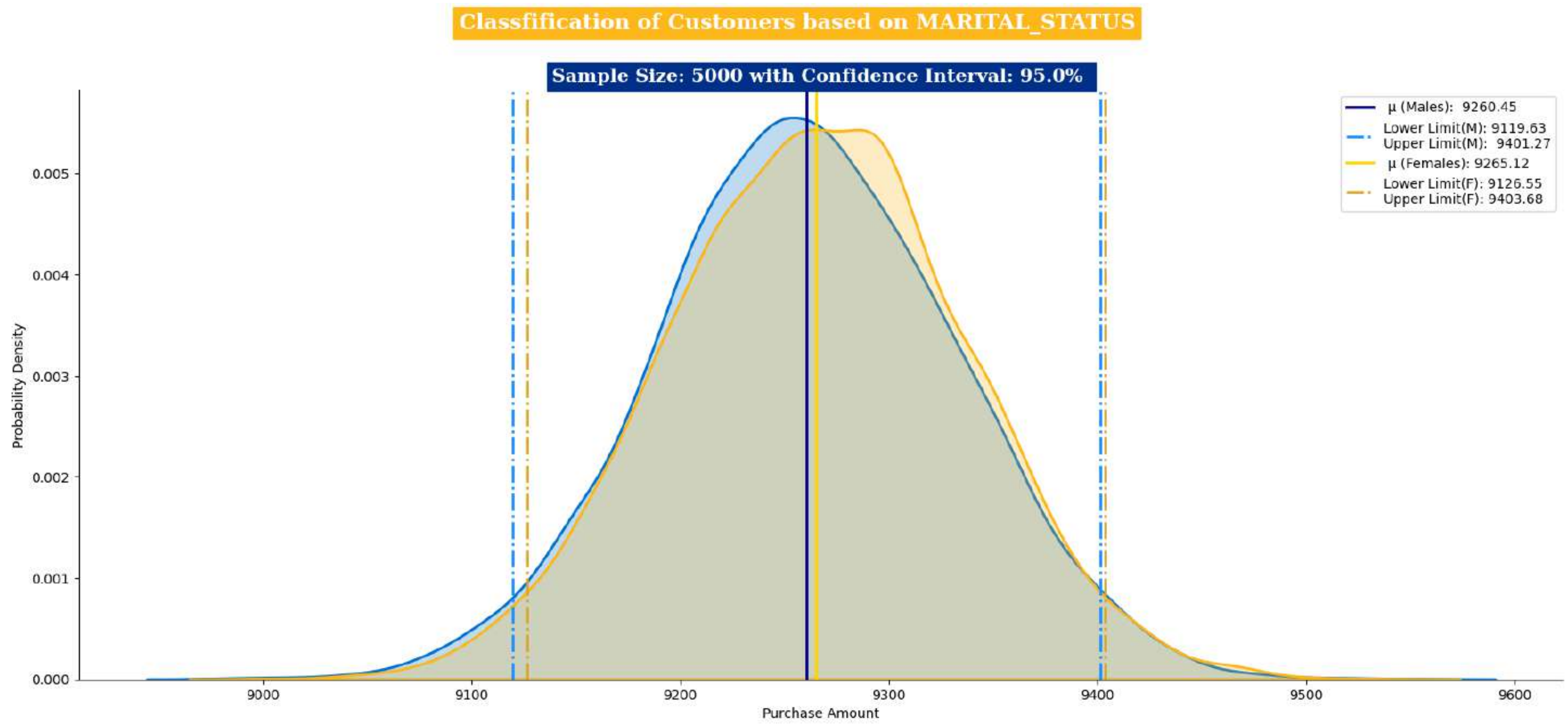
```
sample_sizes = [50,500,5000,50000,100000]
ci = [95]
ntimes = 10000

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

for j in ci:
    for i in sample_sizes:
        m_avg, s_avg, ll_m, ul_m, ll_s, ul_s = bootstrapping('MARITAL_STATUS',wm_married_cust, wm_single_cust, i, ntimes, j)

        df.loc[len(df.index)] = ['M' , i , ll_m , ul_m , m_avg , j , (ul_m - ll_m)]
        df.loc[len(df.index)] = ['F' , i , ll_s , ul_s , s_avg , j , (ul_s - ll_s)]

df
```

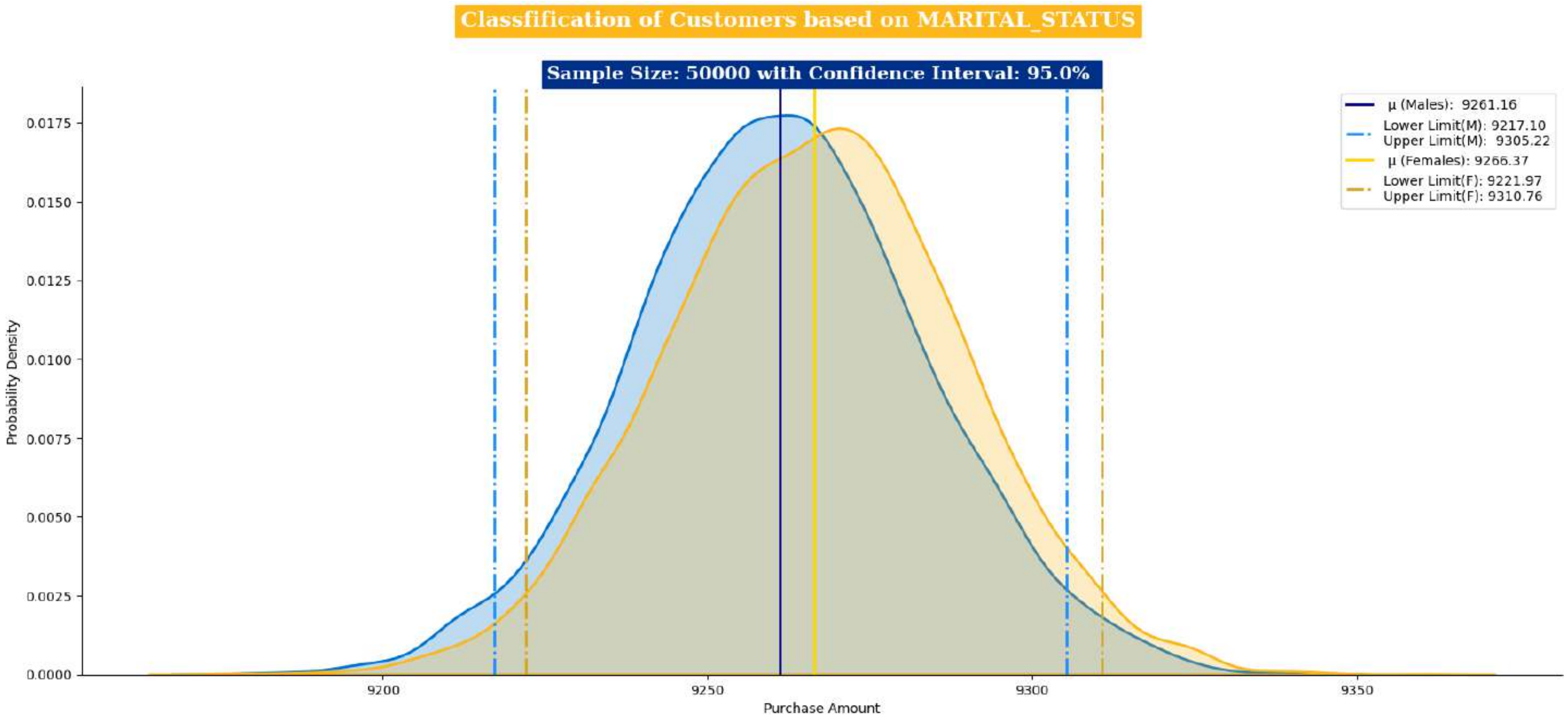
Out[390]:

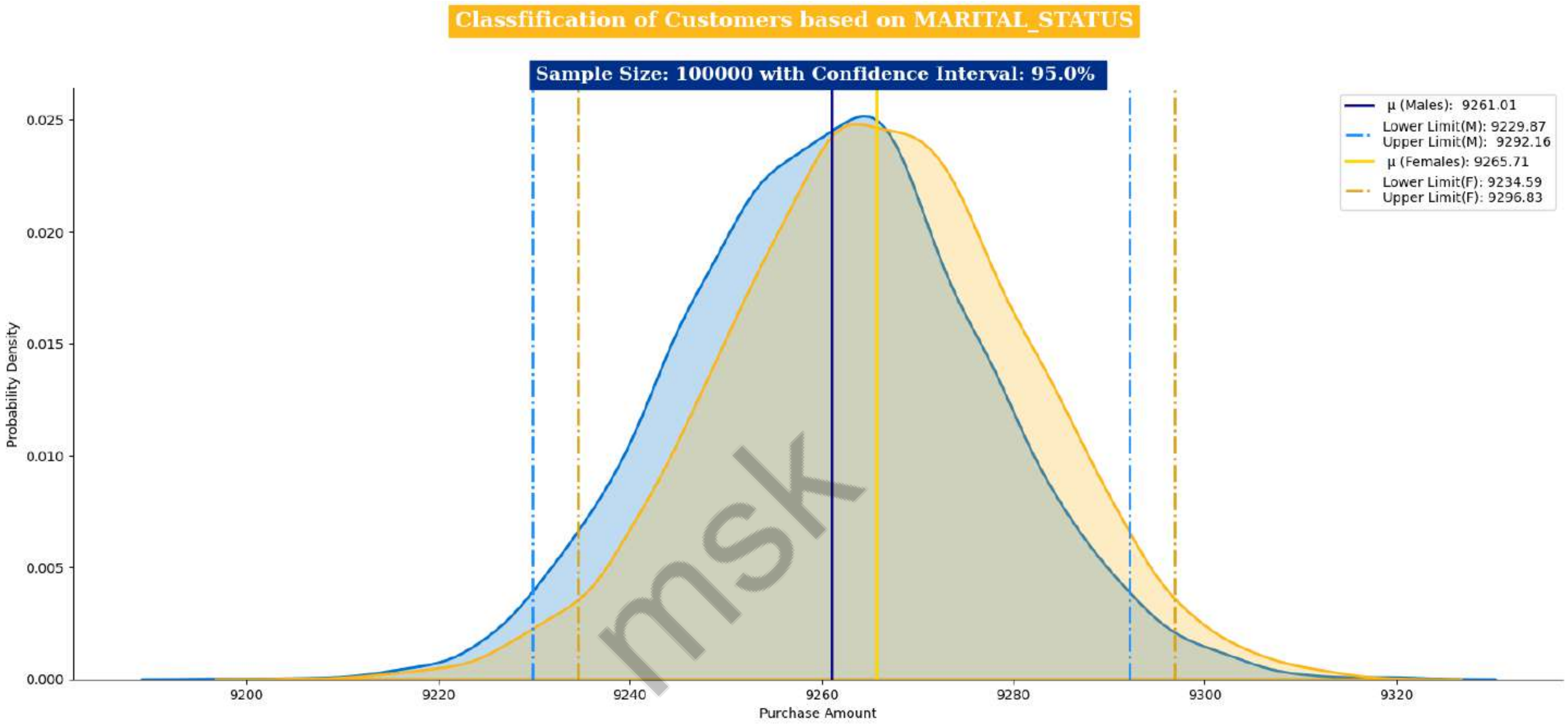
	Gender	Sample Size	Lower Limit	Upper Limit	Sample Mean	Confidence Interval	Range
0	M	50	7877.33	10665.86	9271.59	95	2788.53
1	F	50	7865.50	10646.18	9255.84	95	2780.68
2	M	500	8816.65	9703.87	9260.26	95	887.22
3	F	500	8821.39	9709.93	9265.66	95	888.54
4	M	5000	9119.63	9401.27	9260.45	95	281.64
5	F	5000	9126.55	9403.68	9265.12	95	277.13
6	M	50000	9217.10	9305.22	9261.16	95	88.12
7	F	50000	9221.97	9310.76	9266.37	95	88.79
8	M	100000	9229.87	9292.16	9261.01	95	62.29
9	F	100000	9234.59	9296.83	9265.71	95	62.24


Classfification of Customers based on MARITAL_STATUS
Sample Size: 50 with Confidence Interval: 95.0%
μ (Males): 9271.59
Lower Limit(M): 7877.33
Upper Limit(M): 10665.86
μ (Females): 9255.84
Lower Limit(F): 7865.50
Upper Limit(F): 10646.18
Probability Density
Purchase Amount


Classfification of Customers based on MARITAL_STATUS
Sample Size: 500 with Confidence Interval: 95.0%
μ (Males): 9260.26
Lower Limit(M): 8816.65
Upper Limit(M): 9703.87
μ (Females): 9265.66
Lower Limit(F): 8821.39
Upper Limit(F): 9709.93
Probability Density
Purchase Amount
msk


Classfification of Customers based on MARITAL_STATUS
Sample Size: 5000 with Confidence Interval: 95.0%
μ (Males): 9260.45
Lower Limit(M): 9119.63
Upper Limit(M): 9401.27
μ (Females): 9265.12
Lower Limit(F): 9126.55
Upper Limit(F): 9403.68
Probability Density
Purchase Amount





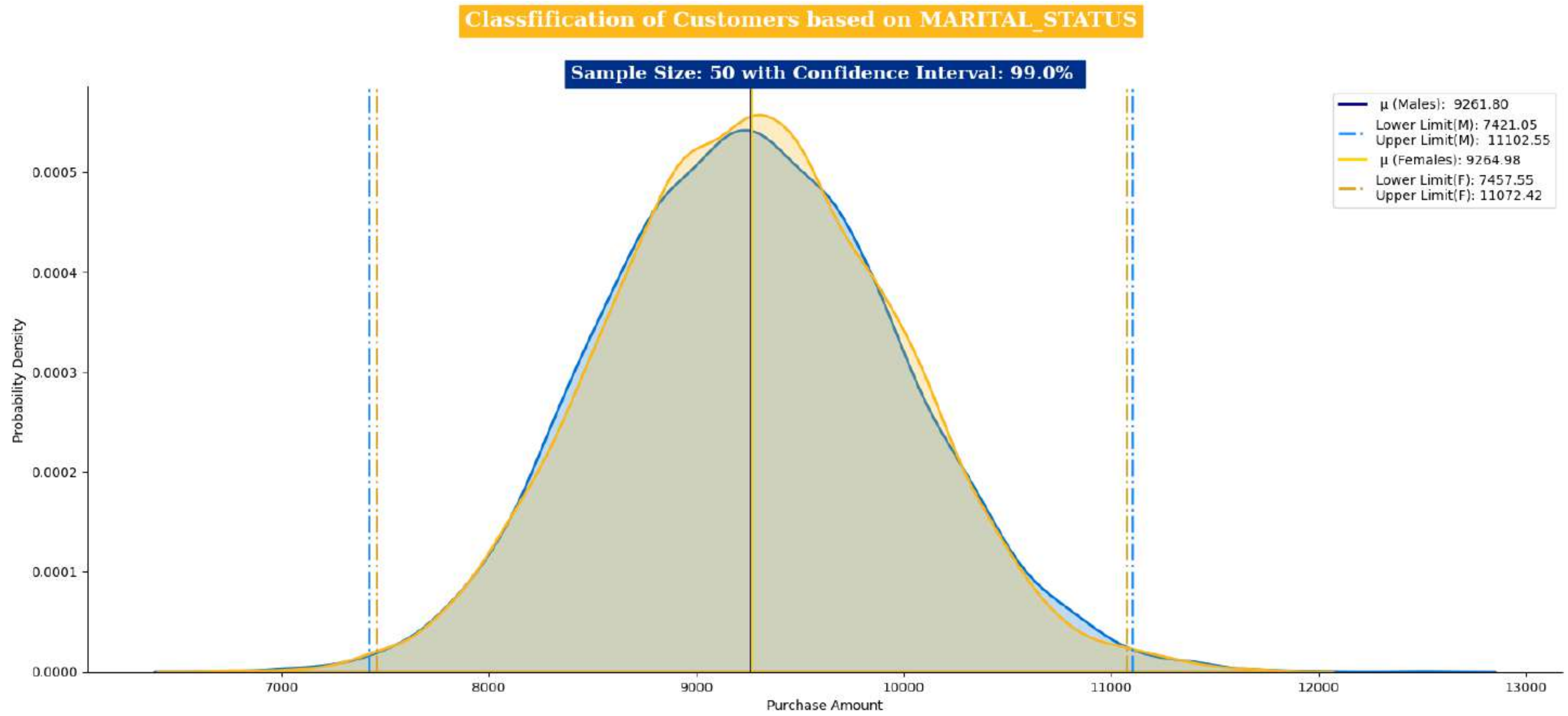
✅ 99% CI

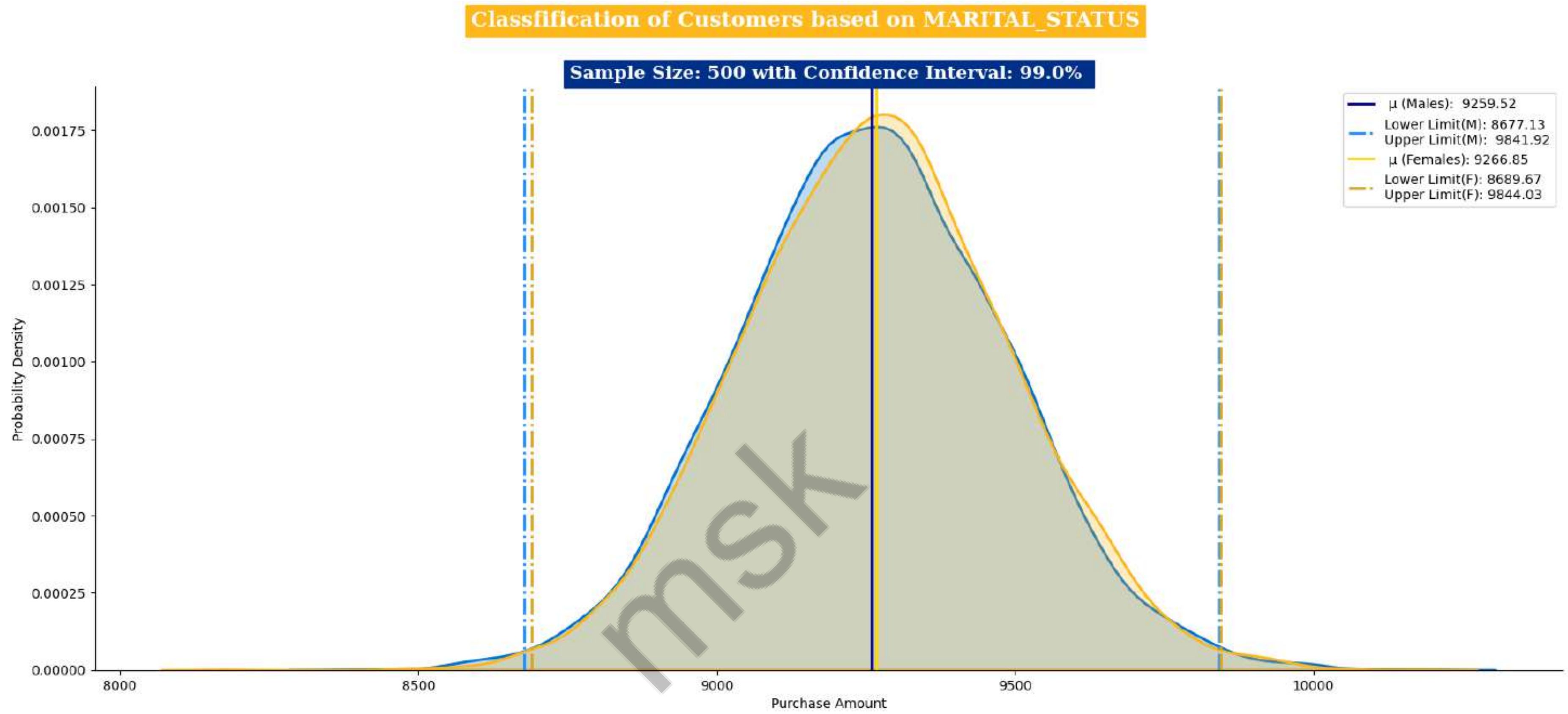
In [391…

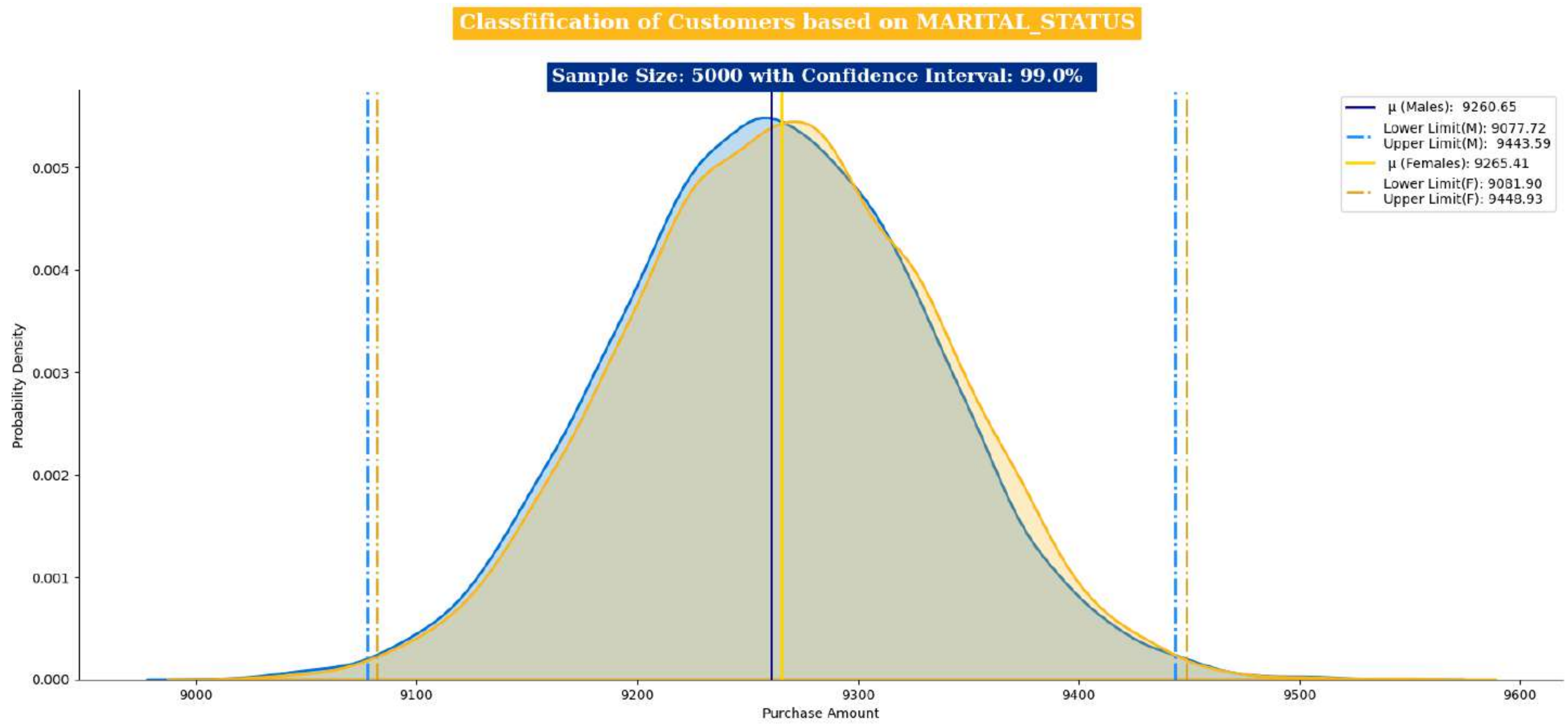
```
sample_sizes = [50,500,5000,50000,100000]
ci = [99]
ntimes = 10000

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

for j in ci:
    for i in sample_sizes:
        m_avg, s_avg, ll_m, ul_m, ll_s, ul_s = bootstrapping('MARITAL_STATUS',wm_married_cust, wm_single_cust, i, ntimes, j)

        df.loc[len(df.index)] = ['M' , i , ll_m , ul_m , m_avg , j , (ul_m - ll_m)]
        df.loc[len(df.index)] = ['F' , i , ll_s , ul_s , s_avg , j , (ul_s - ll_s)]

df
```

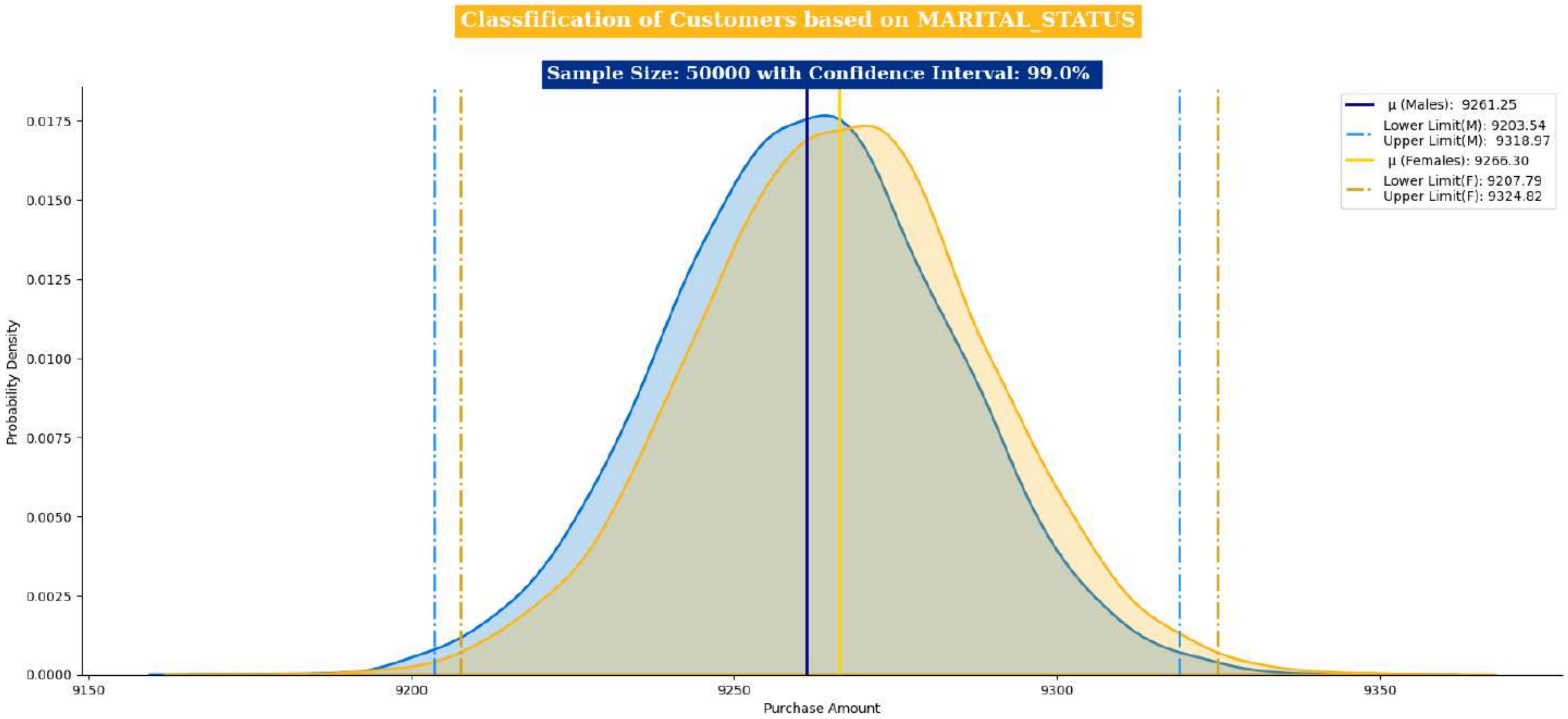
Out[391]:

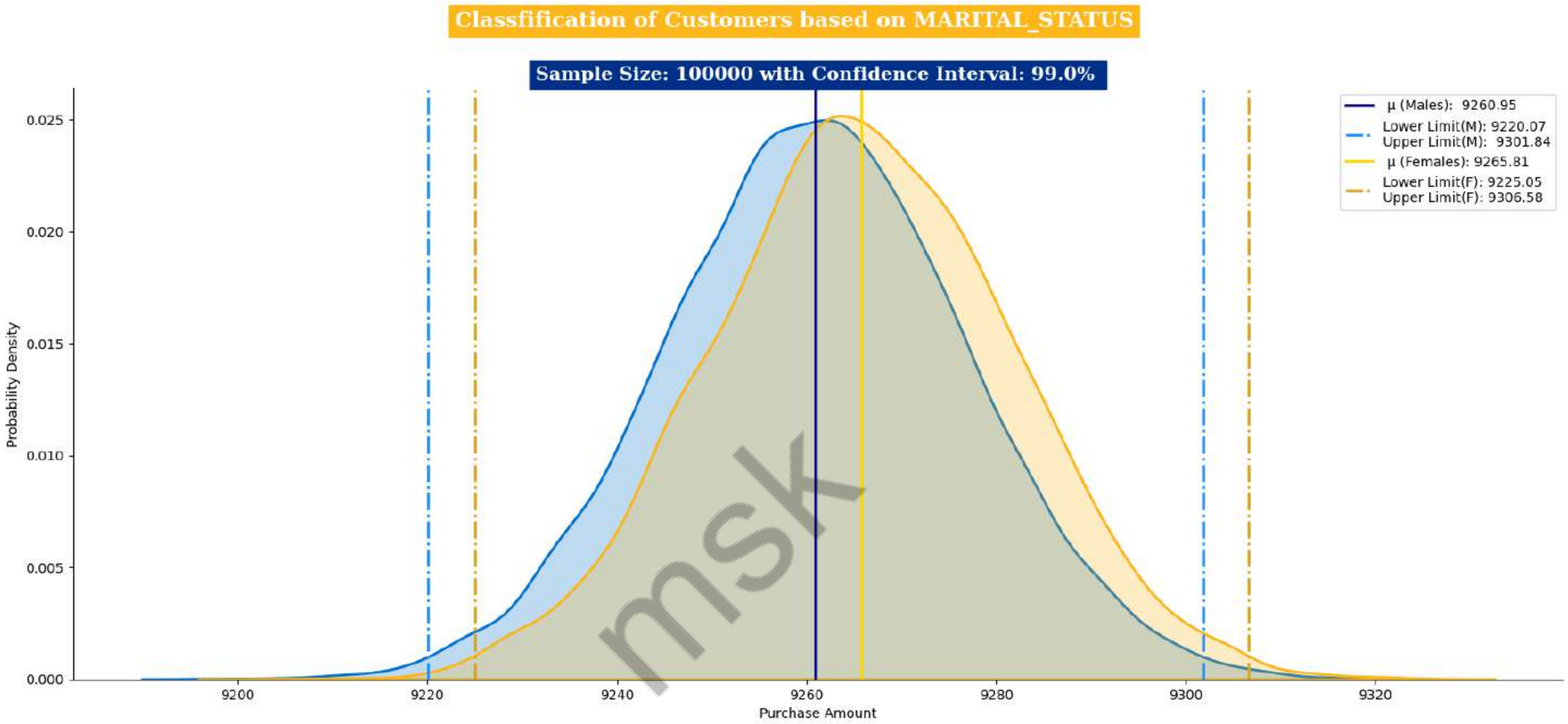
	Gender	Sample Size	Lower Limit	Upper Limit	Sample Mean	Confidence Interval	Range
0	M	50	7421.05	11102.55	9261.80	99	3681.50
1	F	50	7457.55	11072.42	9264.98	99	3614.87
2	M	500	8677.13	9841.92	9259.52	99	1164.79
3	F	500	8689.67	9844.03	9266.85	99	1154.36
4	M	5000	9077.72	9443.59	9260.65	99	365.87
5	F	5000	9081.90	9448.93	9265.41	99	367.03
6	M	50000	9203.54	9318.97	9261.25	99	115.43
7	F	50000	9207.79	9324.82	9266.30	99	117.03
8	M	100000	9220.07	9301.84	9260.95	99	81.77
9	F	100000	9225.05	9306.58	9265.81	99	81.53


Classfification of Customers based on MARITAL_STATUS
Sample Size: 50 with Confidence Interval: 99.0%
μ (Males): 9261.80
Lower Limit(M): 7421.05
Upper Limit(M): 11102.55
μ (Females): 9264.98
Lower Limit(F): 7457.55
Upper Limit(F): 11072.42
Probability Density
Purchase Amount


Classfification of Customers based on MARITAL_STATUS
Sample Size: 500 with Confidence Interval: 99.0%
μ (Males): 9259.52
Lower Limit(M): 8677.13
Upper Limit(M): 9841.92
μ (Females): 9266.85
Lower Limit(F): 8689.67
Upper Limit(F): 9844.03
Probability Density
Purchase Amount


Classfification of Customers based on MARITAL_STATUS
Sample Size: 5000 with Confidence Interval: 99.0%
μ (Males): 9260.65
Lower Limit(M): 9077.72
Upper Limit(M): 9443.59
μ (Females): 9265.41
Lower Limit(F): 9081.90
Upper Limit(F): 9448.93
Probability Density
Purchase Amount





🏷️ Insights

- 🧐 ***Observation***
  - The average for both of them changes significantly as the sample size increases:
    - As the **sample size** `increases`, the average values based on `Marital_status` undergo mineute changes.
    - **Larger sample sizes** tend to provide *more representative insights* into the population, leading to `more stable` and reliable average values.
  - Both plots start to separate slightly and still overlapping:
    - With *increasing sample size*, the plots representing the data for single and married customers start to `diverge a little` and show approx similar patterns.
    - This mineute separation could indicate that the **larger sample sizes** are capturing `more nuances` in the data, revealing no or mineute differences between married and single customers that might be as apparent in smaller samples.

👉 **For *Any Sample size*, The *confidence interval [90%,95%,99%]* for both** *Married and Single* **is `OVERLAPPING` and as the sample size increases, we can see the interval ranges slightly seperating and still they both `OVERLAP`.**

In [ ]:

In [ ]:

In [ ]:

🔔 ✴️ CONCLUSION & RECOMMENDATION :

Recommendations with Actionable Insights

- **Target Male Shoppers:**
  - Launch targeted marketing campaigns showcasing products preferred by men. Offer exclusive deals on male-oriented items.
- **Age-Group Focus:**
  - Analyze popular products within the 26 - 45 age group. Introduce promotions highlighting these products to enhance engagement.
- **Engage Younger Shoppers:**
  - Create a loyalty program for the 0 - 17 age group with rewards for frequent purchases. Implement visually appealing online promotions.
- **Customer Segmentation:**
  - Conduct a detailed analysis of buying behaviors within specific age brackets. Tailor promotions and product placements accordingly.
- **Enhance Shopping Experience (51 - 55):**
  - Implement a personalized shopping experience for customers aged 51 - 55, including early access to sales and exclusive discounts.
- **Post-Black Friday Engagement:**
  - Develop an automated follow-up email system with personalized recommendations based on customers' Black Friday purchases.
- **Differentiated Marketing for Genders:**
  - Launch gender-specific marketing campaigns, emphasizing affordability for men and premium offerings for women.
- **Accessibility for All Age Groups:**
  - Partner with local transport services to provide convenient transportation options. Promote online shopping accessibility for those facing mobility challenges.
- **Analyze High-Spending Individuals:**
  - Conduct surveys or interviews with high-spending individuals to understand preferences. Use insights to refine product offerings and marketing strategies.
- **Collaboration with Local Transport:**
  - Initiate discussions with local transport providers for potential collaborations. Offer discounts or incentives for customers using designated transport services.
- **Evaluate Price Sensitivity:**
  - Implement dynamic pricing strategies based on real-time data analysis. Test price elasticity within different demographic segments.
- **Continuous Data Analysis:**
  - Establish a dedicated data analytics team to continuously monitor and analyze customer data. Implement an agile approach to adapt strategies based on emerging trends.

🧺 Leveraging Conclusions for WALMART 🧺:

- ✳️ **Targeted Marketing**
  - 🔹 Boost spending for 0 - 17 age group with attractive incentives and tailored marketing.
- ✳️ **Customer Segmentation**
  - 🔹 Optimize product selection and pricing for age groups with similar buying behaviors.
- ✳️ **Premium Services**
  - 🔹 Enhance the shopping experience for high-spending 51 - 55 age group with premium services and tailored loyalty programs.
- ✳️ **Identifying Differences:**
  - 🔹 `Walmart` can capitalize on the recognized distinctions between male and female customer behaviors.
  - 🔹 **Tailoring marketing strategies, product offerings, and promotions** based on these differences can enhance customer engagement.
  - 🔹 By understanding gender-specific preferences, Walmart can create **more targeted and appealing campaigns** for each demographic.
- ✳️ **Decision-Making:**
  - 🔹 Decision-makers at Walmart now have valuable insights to inform their strategic decisions.

🔹 Understanding how gender influences customer choices enables more precise decision-making in areas such as product assortment, pricing strategies, and promotional activities.

🔹 Informed decision-making ensures that resources are allocated effectively, maximizing the impact of business initiatives.

- ✳️ **Operational Adjustments:**

  🔹 Operational aspects, such as **inventory management** and **store layout**, can benefit from insights into gender-related patterns.

  🔹 `Walmart` may consider optimizing inventory based on observed preferences, ensuring that popular products are *well-stocked*.

  🔹 Store layouts can be adjusted to enhance the **shopping experience** for both genders, creating a more personalized and enjoyable atmosphere.

In summary, leveraging these conclusions empowers `WALMART` to tailor its approach to different customer segments, make informed decisions grounded in observed behaviors, and optimize operational aspects for a more customer-centric and efficient retail experience. `WALMART` can strategically implement changes to drive customer engagement, increase sales, and enhance the overall shopping experience.
