

BUSINESS CASE: WALMART - CONFIDENCE INTERVAL AND CLT

About Walmart

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

Business Problem

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).

1. Defining Problem Statement and Analyzing basic metrics (10 Points)

- 1. Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), statistical summary**
- 2. Non-Graphical Analysis: Value counts and unique attributes**
- 3. Visual Analysis - Univariate & Bivariate**
 - **For continuous variable(s): Distplot, countplot, histogram for univariate analysis**
 - **For categorical variable(s): Boxplot**
 - **For correlation: Heatmaps, Pairplots**

1. Missing Value & Outlier Detection (10 Points)

2. Business Insights based on Non- Graphical and Visual Analysis (10 Points)

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

3. Answering questions (50 Points)

- **Are women spending more money per transaction than men? Why or Why not? (10 Points)**

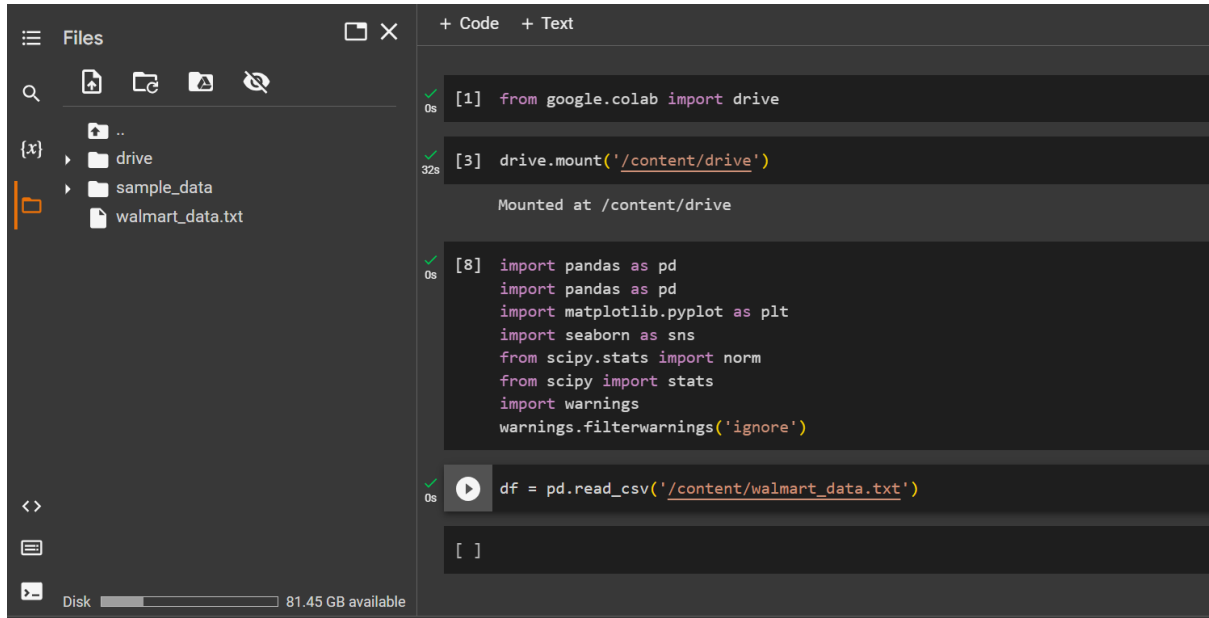
- **Confidence intervals and distribution of the mean of the expenses by female and male customers (10 Points)**
 - **Are confidence intervals of average male and female spending overlapping? How can Walmart leverage this conclusion to make changes or improvements? (10 Points)**
 - **Results when the same activity is performed for Married vs Unmarried (10 Points)**
 - **Results when the same activity is performed for Age (10 Points)**
- 4. Final Insights (10 Points) - Illustrate the insights based on exploration and CLT**
- **Comments on the distribution of the variables and relationship between them**
 - **Comments for each univariate and bivariate plots**
 - **Comments on different variables when generalizing it for Population**
- 5. Recommendations (10 Points)**
- **Actionable items for business. No technical jargon. No complications. Simple action items that everyone can understand**

LOADING DATASET

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

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

IMPORTING LIBRARIES



The screenshot shows a JupyterLab environment. On the left, a file explorer pane displays a directory structure with folders 'drive' and 'sample_data', and a file 'walmart_data.txt'. The main area on the right contains a code editor with four cells. The first cell imports 'drive' from 'google.colab'. The second cell mounts the drive at '/content/drive', showing a confirmation message. The third cell imports various libraries: 'pandas' (twice), 'matplotlib.pyplot', 'seaborn', 'scipy.stats' (with 'norm' imported), 'scipy' (with 'stats' imported), and 'warnings' (with 'filterwarnings' set to 'ignore'). The fourth cell uses 'pd.read_csv' to load 'walmart_data.txt' from the mounted drive. The output of the fourth cell is an empty list '[]'.

```
[1] from google.colab import drive

[3] drive.mount('/content/drive')

Mounted at /content/drive

[8] import pandas as pd
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
from scipy import stats
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('/content/walmart_data.txt')

[ ]
```

```
import pandas as pd
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
```

CHECKING AND READING THE DATASET AND DISPLAYING ITS CONTENT

```
df = pd.read_csv('/content/walmart_data.txt')
df.head()
```

```
[9] df = pd.read_csv('/content/walmart_data.txt')

[11] df.head()
```

	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

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

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

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

```
[ ]
```

CHECKING NULL VALUES

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

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

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

```
[ ]
```

THERE IS NO NULL VALUES

CHECKING THE UNIQUE VALUES

```
df.nunique().sort_values(ascending=False)
```

```
[14] df.nunique().sort_values(ascending=False)
```

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

CHECKING DUPLICATES

```
df.duplicated().sum()
```

```
df.duplicated().sum()
```

```
0
```

```
[ ]
```

THERE IS NO DUPLICATE VALUE

```
df.info()
```

```
df.info()

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

HERE USER_ID , PRODUCT_ID , GENDER , AGE , CITY_CATEGORY AND MARTIAL_STATUS HAVE CATEGORY SO WE NEED TO CHANGE THE DATATYPES

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

df.dtypes

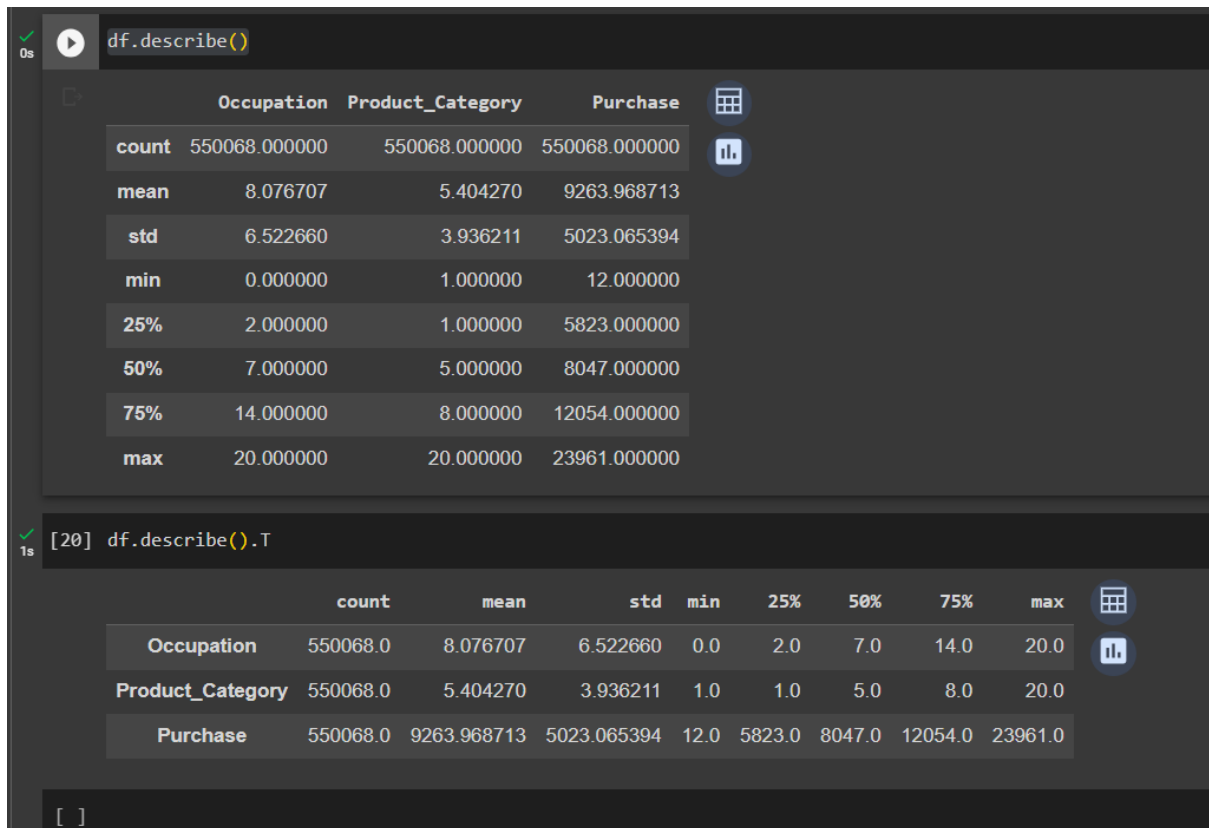
```
df.dtypes

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

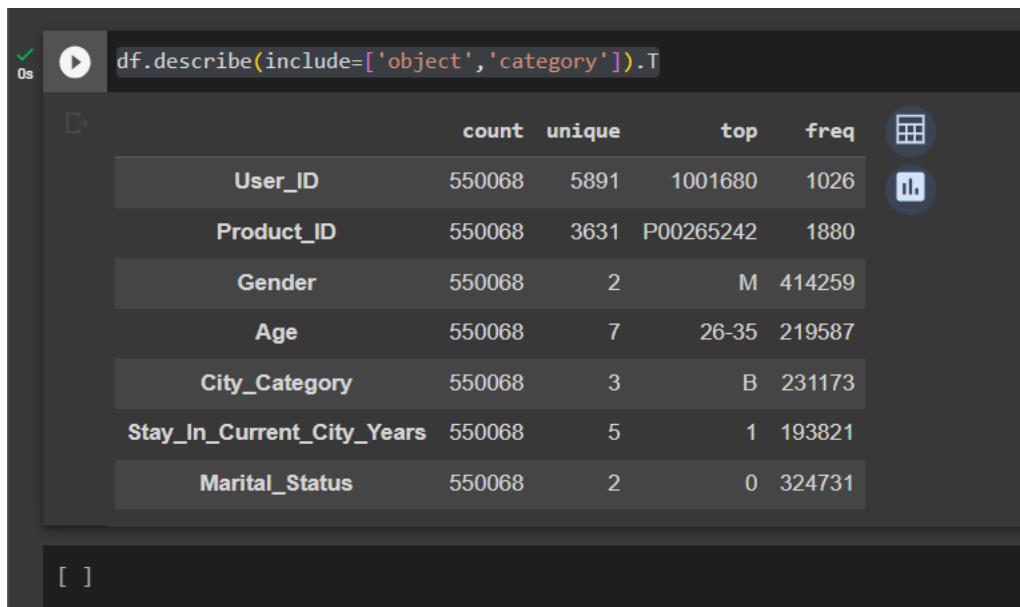
[ ]
```

DATA TYPES HAS NOW CHANGED

df.describe()



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



- THE USER_ID 1001680 HAS FREQUENT PURCHASE IN WALMART .IT HAS 5991 UNIQUE USERS .

- THE PRODUCT_ID P00265242 HAS FREQUENT SOLD ITEMS .IT HAS 3631 UNIQUE USERS.
- THERE ARE FREQUENT MALES WHO HAVE PURCHASED ITEMS FROM WALMART THAN FEMALES.
- THE AGE OF MOST FREQUENT WHO HAVE PURCHASED FROM WALMART COMES UNDER 26-35.IT HAS 7 UNIQUE VALUES.
- THERE ARE 3 DIFFERENT CITY CATEGORIES WHERE MOST FREQUENT BUYERS COME UNDER CATEGORY B.
- IN STAY_IN_CURRENT_CITY_YEARS HAS 5 UNIQUE VALUES WHICH STAYS IN CURRENT CITY FOR 1 YEAR.
- IN MARITAL_STATUS THERE ARE 2 UNIQUE VALUES AND MOST CUSTOMERS ARE UNMARRIED.

UNIVARIATE ANALYSIS

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

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

```
✓ [18] df['User_ID'].nunique()  
0s 5891
```

```
✓ [19] df['Product_ID'].nunique()  
0s 3631
```

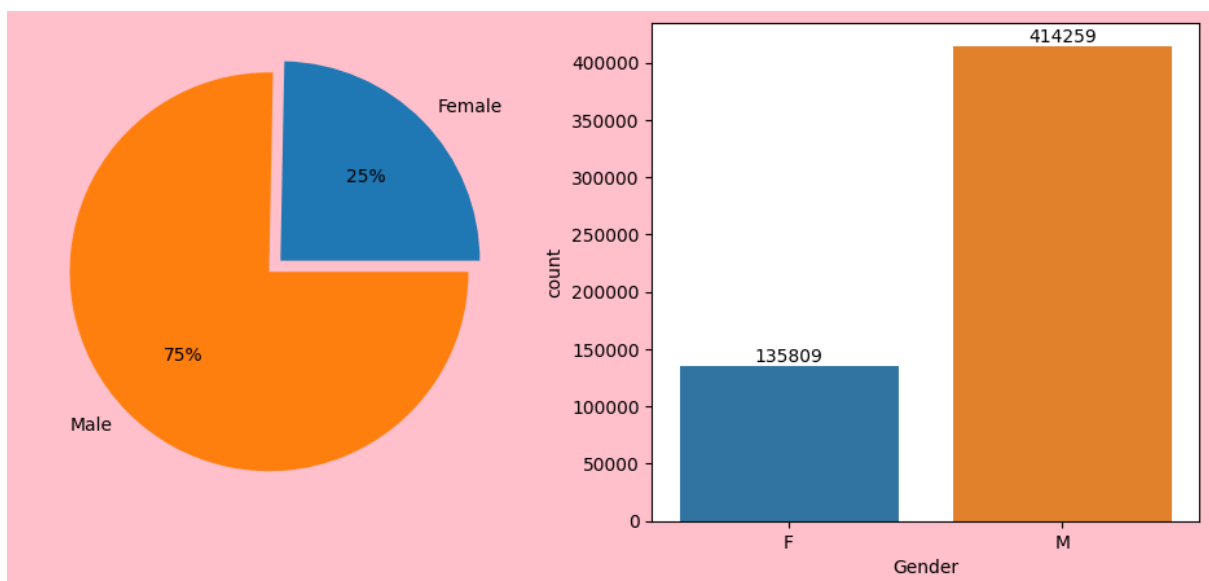

COUNTPLOT

```
plt.figure(figsize = (12,5)).set_facecolor("pink")

plt.subplot(1,2,1)
labels = ['Female','Male']
plt.pie(df.groupby('Gender')['Gender'].count(), labels = labels,
explode = (0.08,0), autopct = '%0.0f%%')

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

plt.show()
```



Tracking the amount spent per transaction of all the 50 million females customers, and all the 50 million males customers there are 25% females and 75% males records and 135809 females records and 414259 males record.

df['Age'].unique()

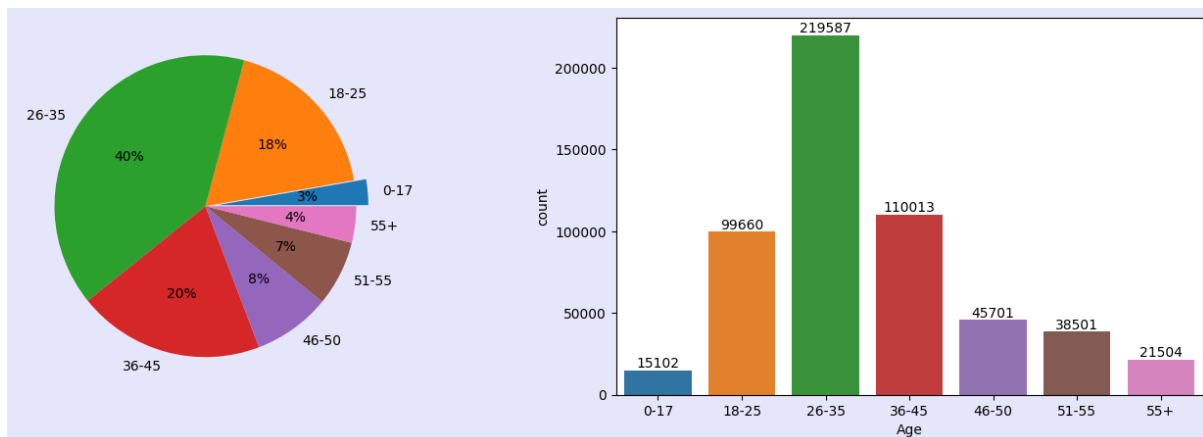
```
df['Age'].unique()
['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25']
Categories (7, object): ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
```

```
plt.figure(figsize = (17,5)).set_facecolor("lavender")

plt.subplot(1,2,1)
labels = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
plt.pie(df.groupby('Age')['Age'].count(), labels = labels, explode =
(0.08,0,0,0,0,0,0), autopct = '%0.0f%%')

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

plt.show()
```



- **40% of the buyers fall under the age group of 26-35 which is the highest amongst all age groups.**
- **Approximately 0.21 million records are present for age group 26-35 followed by 0.11 million records for group 36-45.**
- **Age group 0-17 and 55+ are the least frequent buyers which is only 3% and 4% of the data respectively.**

- **Approximately only 15k and 21k records are there for age group 0-17 and group 55+.**
- **We can observe that most buyers are in within the age of 18-45 before and after this range we can see less buyers.**

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

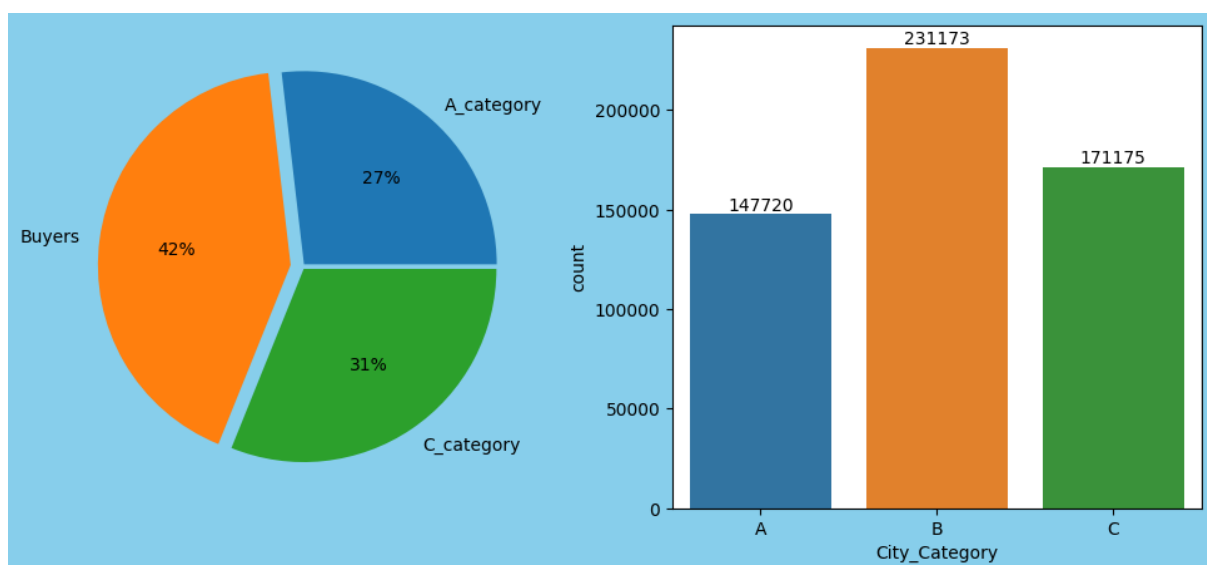
```
0s df['City_Category'].unique()
['A', 'C', 'B']
Categories (3, object): ['A', 'B', 'C']
[ ]
```

```
plt.figure(figsize = (12,5)).set_facecolor("skyblue")

plt.subplot(1,2,1)
labels = ['A_category', 'Buyers', 'C_category']
plt.pie(df.groupby('City_Category')['City_Category'].count(), labels =
labels, explode = (0.015,0.06,0.015), autopct = '%0.0f%%')

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

plt.show()
```



-THERE ARE 42& BUYERS

-THERE ARE 27% A_CATEGORY

-THERE ARE 31% C_CATEGORY

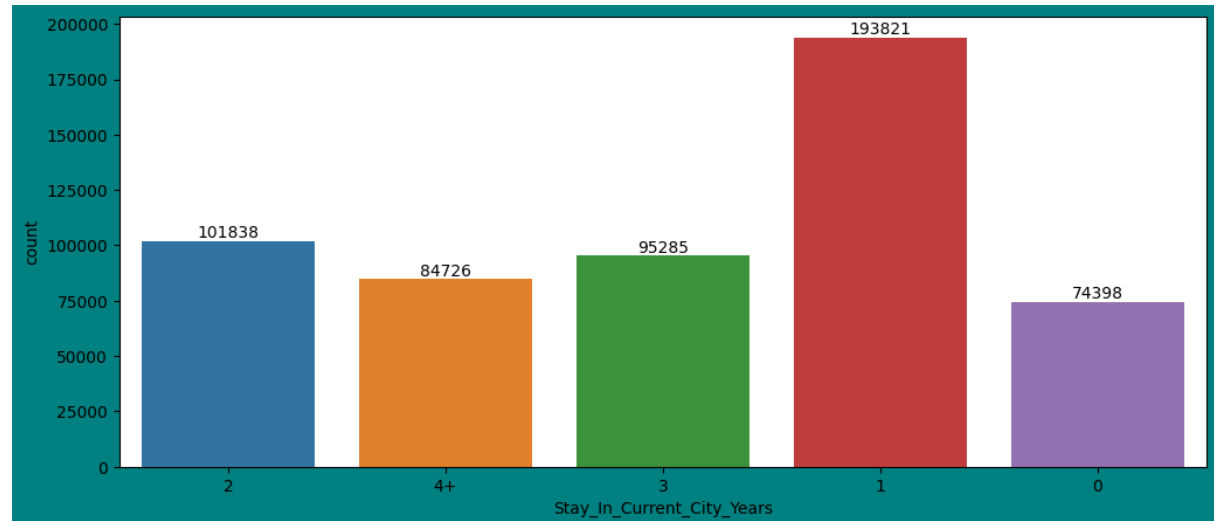
- A HAS 147720 RECORDS ,B HAS 231173 RECORDS AND C HAS 171175 RECORDS.

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

```
df['Stay_In_Current_City_Years'].unique()
array(['2', '4+', '3', '1', '0'], dtype=object)

[ ]
```

```
plt.figure(figsize = (12,5)).set_facecolor("teal")
label = sns.countplot(data = df, x='Stay_In_Current_City_Years')
for i in label.containers:
    label.bar_label(i)
```



MOSTLY BUYERS STAYS IN CURRENT CITY FOR 1 YEAR FOLLOWED BY 2 AND 3 YEARS.

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

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

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

[ ]
```

HERE 0 IS UNMARRIED AND 1 IS MARRIED .SO LETS REPLACE THE MARTIAL STATUS OF DATASET

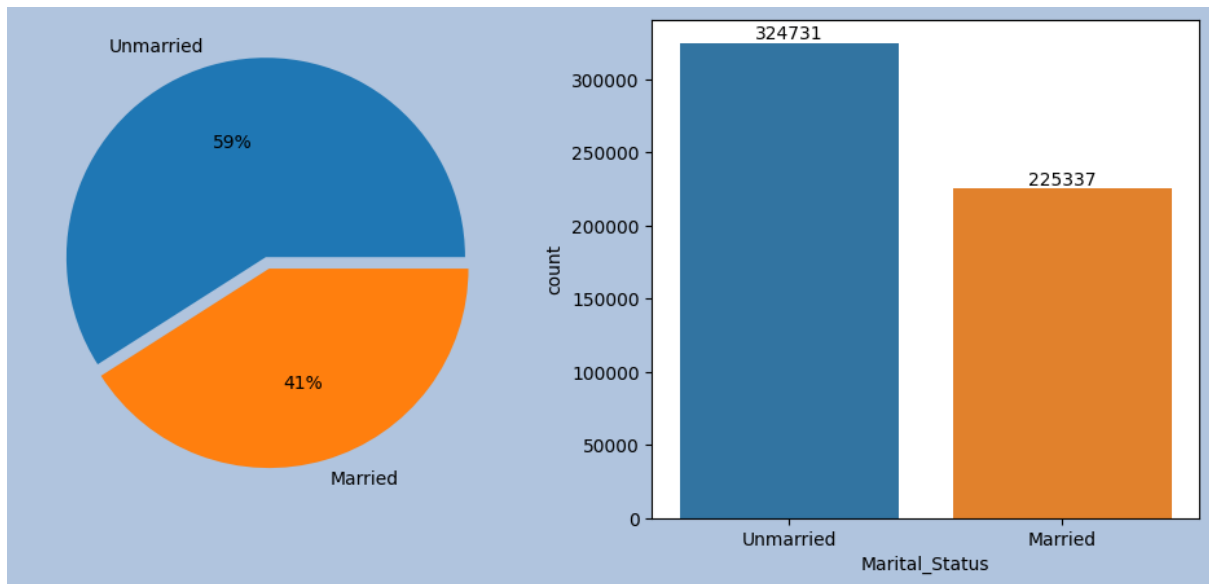
```
df['Marital_Status'].replace(to_replace = 0, value = 'Unmarried', inplace = True)
df['Marital_Status'].replace(to_replace = 1, value = 'Married', inplace = True)
```

```
plt.figure(figsize = (12,5)).set_facecolor("lightsteelblue")

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

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

plt.show()
```

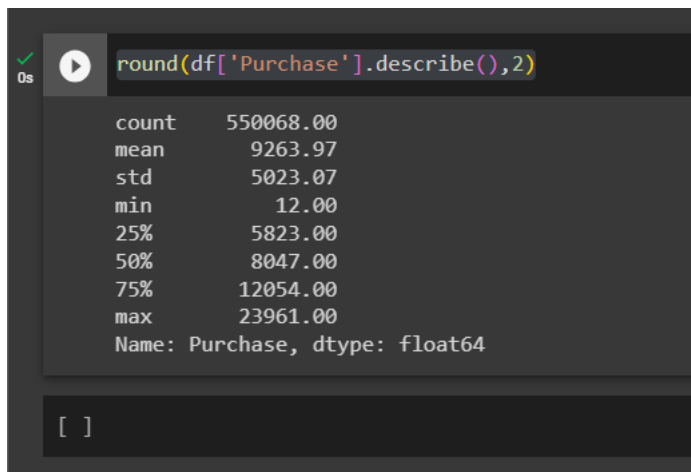


-UNMARRIED IS 59% WHICH IS FREQUENT BUYERS

-MARRIED IS 41%

-MARTIAL STATUS OF UNMARRIED IS APPROX 0.32 MILLION AND MARRIED IS APPROX 0.22 MILLION .

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



-The average order value is 9263.97

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

-The lowest order value is as low as 12.

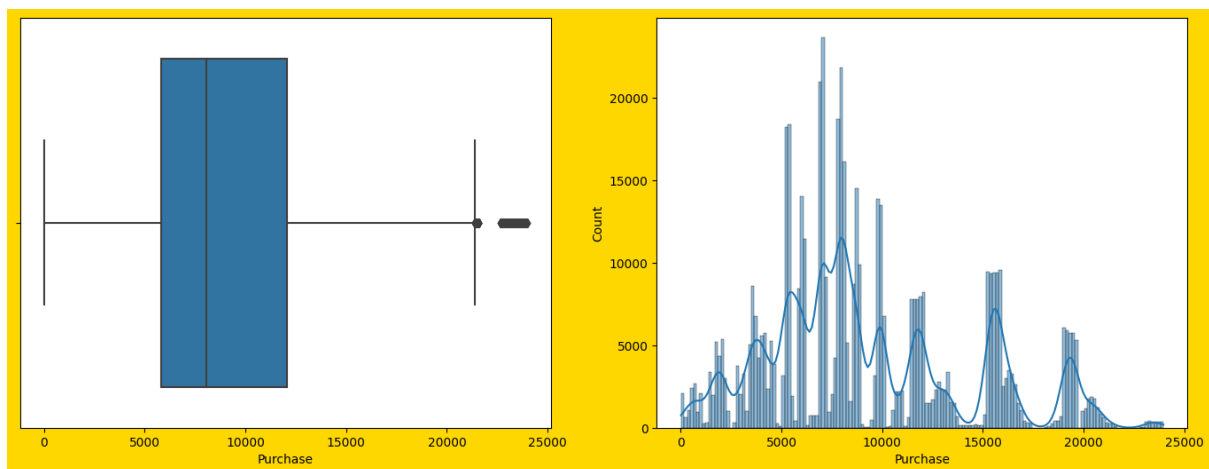
-While, the highest order value is of 23961.

BOXPLOT AND HISTPLOT

```
plt.figure(figsize=(17, 6)).set_facecolor("gold")

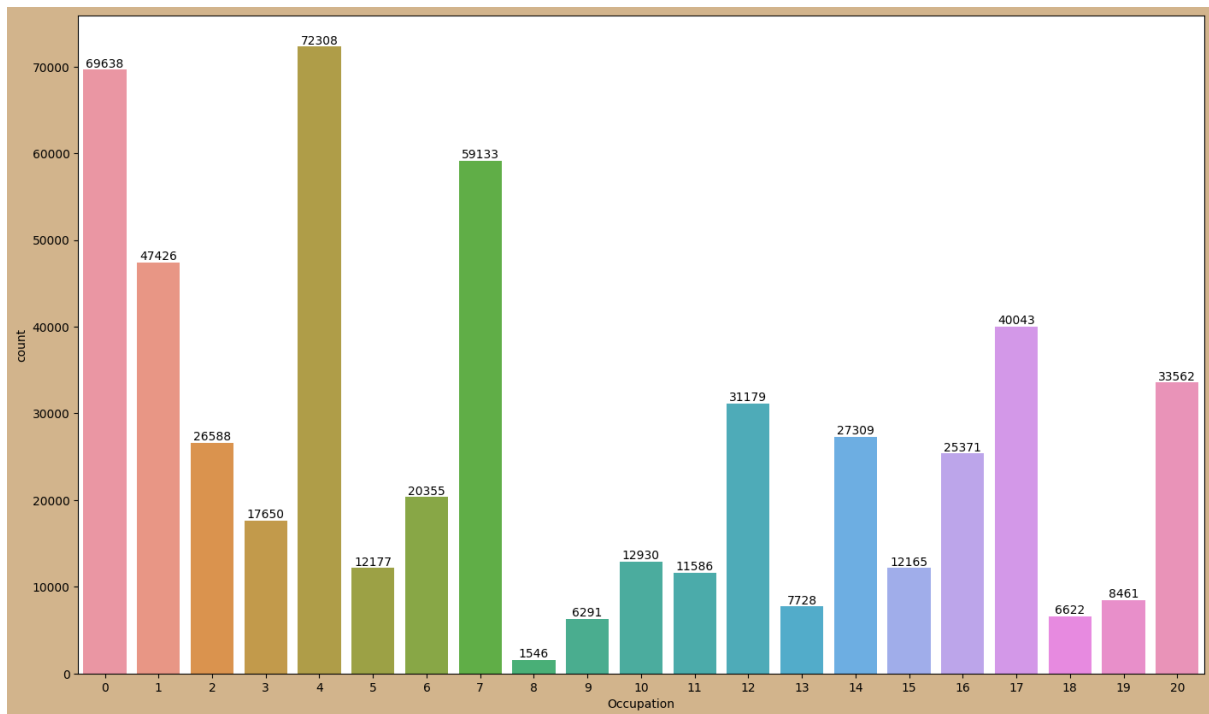
plt.subplot(1,2,1)
sns.boxplot(data=df, x='Purchase', orient='h')

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



- Most of the values are in between 6000 and 12000.**
- Most order values are in the range of 5000 - 10000**
- More orders are in the range of 15000 - 16000 followed by 11000 - 11500 range and very few are also in the 19000 - 20000 range.**

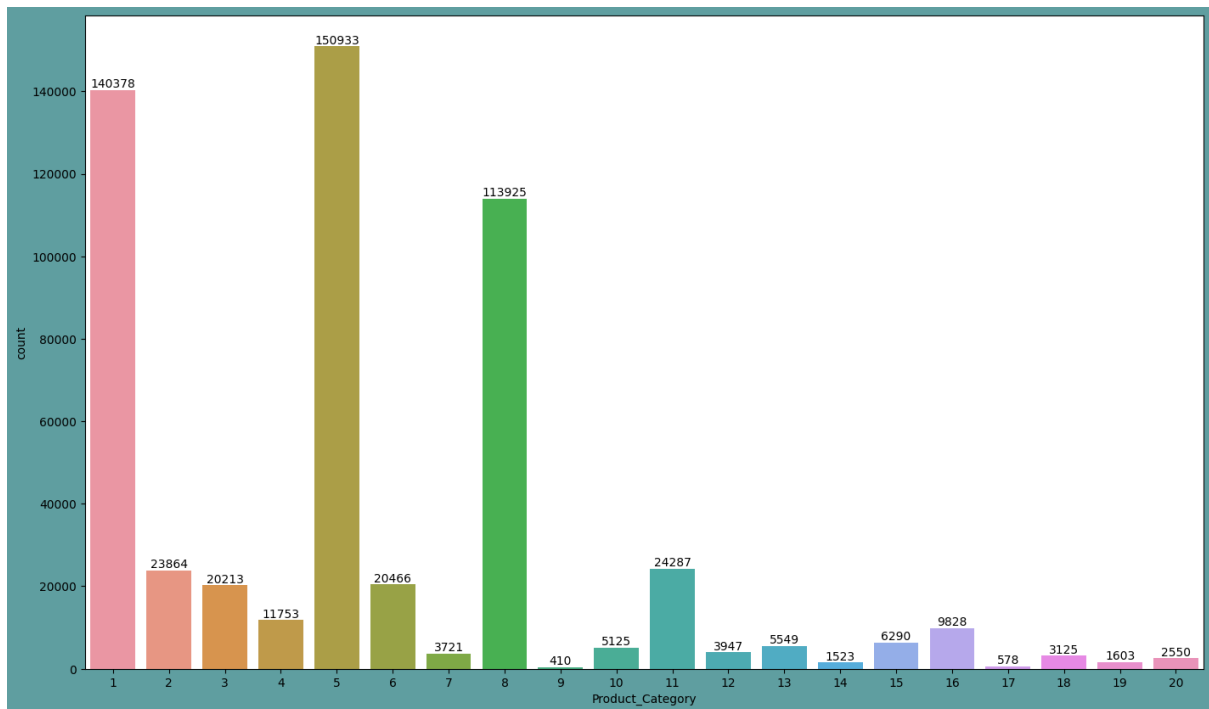
```
plt.figure(figsize=(17, 10)).set_facecolor("tan")
label = sns.countplot(data = df, x='Occupation')
for i in label.containers:
    label.bar_label(i)
```



THE OCCUPATION NUMBER 4 ARE THE MOST FREQUENT BUYERS WHICH HAS 72308 COUNT.

AND THE OCCUPATION WITH NUMBER 8 ARE THE LEAST BUYERS WHICH HAS 1546 COUNT.

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

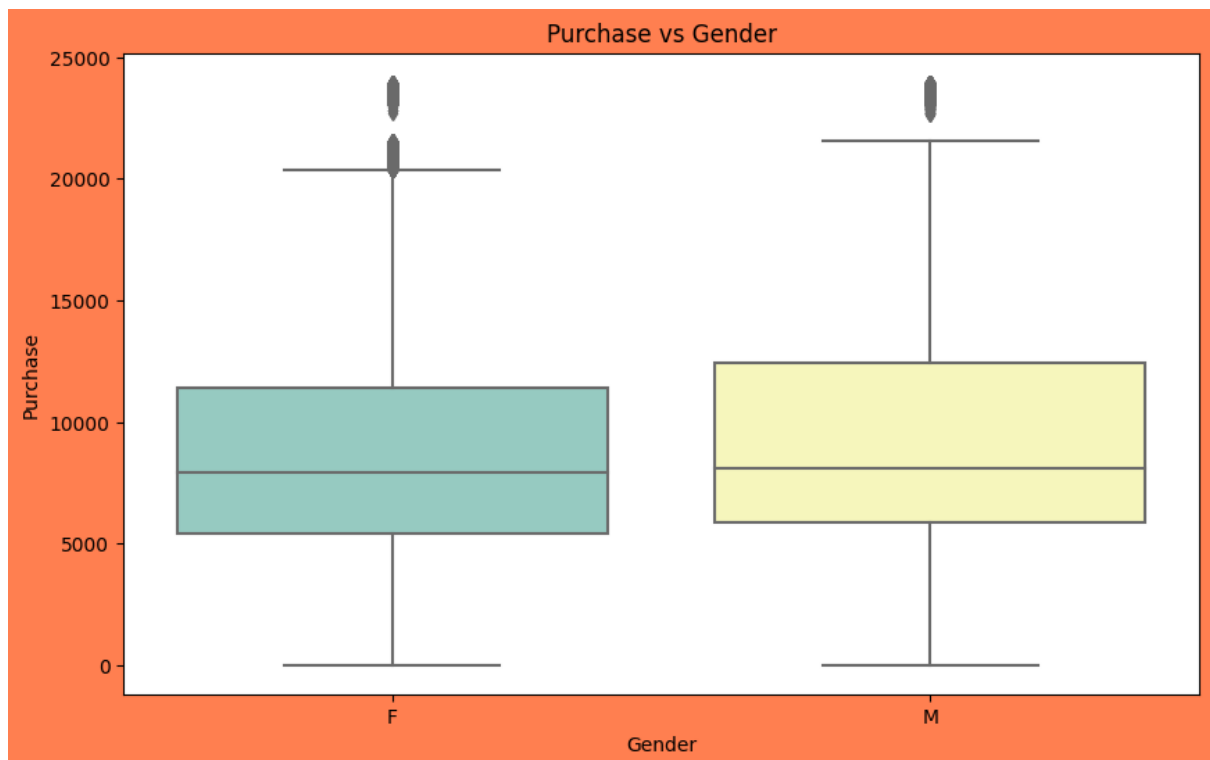



-THE MOST FREQUENT PRODUCT_CATEGORY IS NUMBER 5 WHICH HAS 150933 COUNTS WHICH IS FOLLOWED BY 1 WHICH HAS 140378 COUNTS AND 8 WHICH HAS 113925.

-REST 9 WITH LEAST PRODUCT_CATEGORY WHICH HAS 410 COUNTS FOLLOWED BY 17 WHICH HAS 578 AND 14 WHICH HAS 1523 .

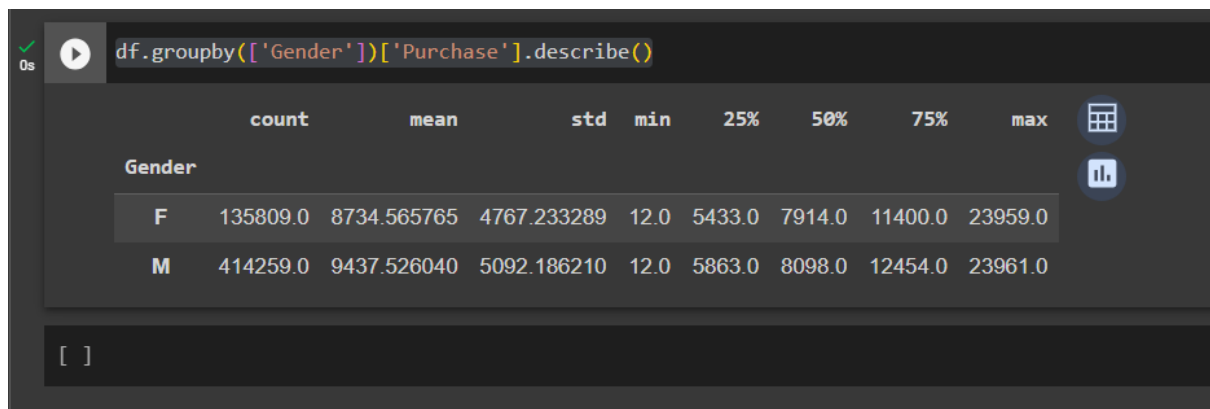
BIVARIATE ANALYSIS

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



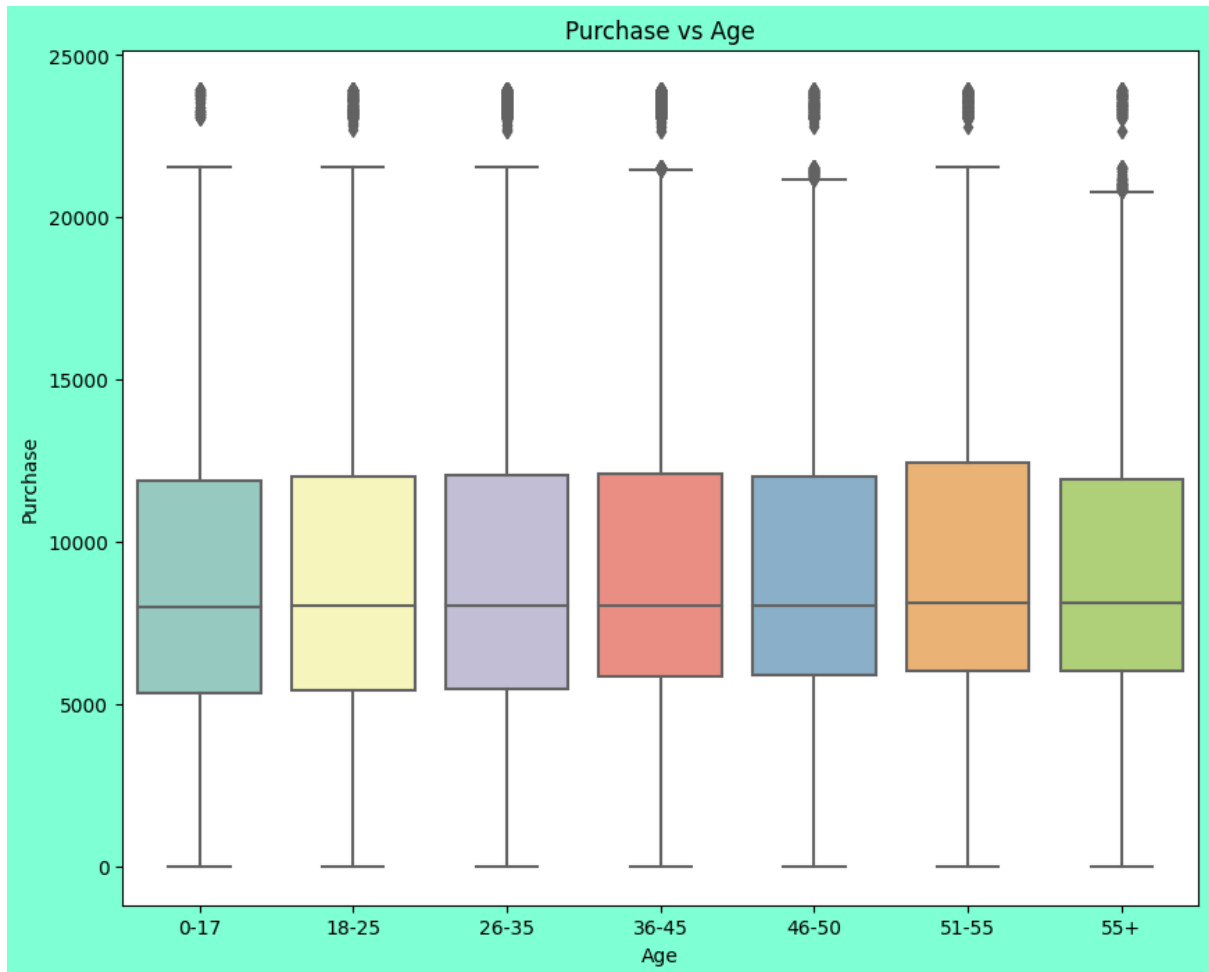
HERE WE CAN CLEARLY SEE THAT MALES PURCHASE MORE THAN FEMALE .

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




HERE AVERAGE PURCHASE VALUE OF MALE IS 9437 AND FEMALE IS 8734 AND MOST FREQUENT PURCHASE OF MALE IS 8098.0 AND FEMALE HAS 7914.0

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



NOT MUCH DIFFERENCE FOUND IN THE PURCHASE HABIT AGE GROUP WISE.

```
df.groupby(['Age'])['Purchase'].describe().T
```

0s  `df.groupby(['Age'])['Purchase'].describe().T`

	Age	0-17	18-25	26-35	36-45	46-50	51-55	55+
count		15102.000000	99660.000000	219587.000000	110013.000000	45701.000000	38501.000000	21504.000000
mean		8933.464640	9169.663606	9252.690633	9331.350695	9208.625697	9534.808031	9336.280459
std		5111.114046	5034.321997	5010.527303	5022.923879	4967.216367	5087.368080	5011.493996
min		12.000000	12.000000	12.000000	12.000000	12.000000	12.000000	12.000000
25%		5328.000000	5415.000000	5475.000000	5876.000000	5888.000000	6017.000000	6018.000000
50%		7986.000000	8027.000000	8030.000000	8061.000000	8036.000000	8130.000000	8105.500000
75%		11874.000000	12028.000000	12047.000000	12107.000000	11997.000000	12462.000000	11932.000000
max		23955.000000	23958.000000	23961.000000	23960.000000	23960.000000	23960.000000	23960.000000

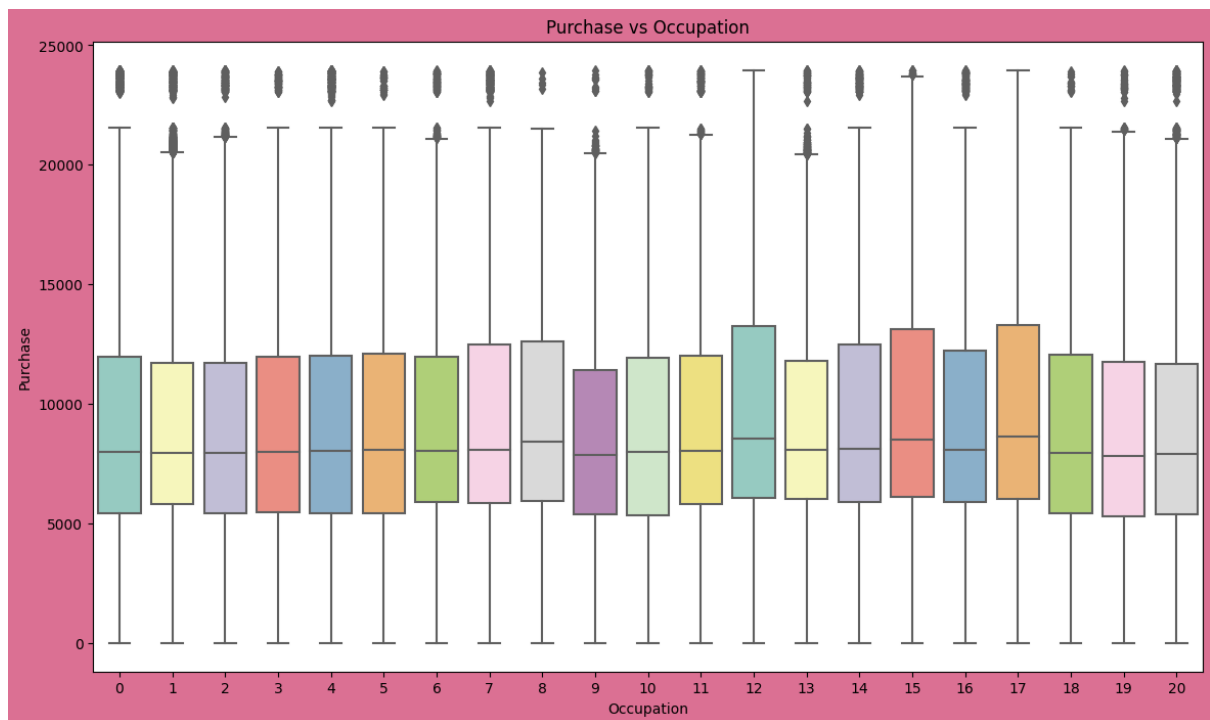
-The average order value is highest for age group 51-55 which is around 9534.

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

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

-The lowest order value is 12 for all the groups.

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



- **There are many outliers in the data.**
- **We can not see much difference in the median values**

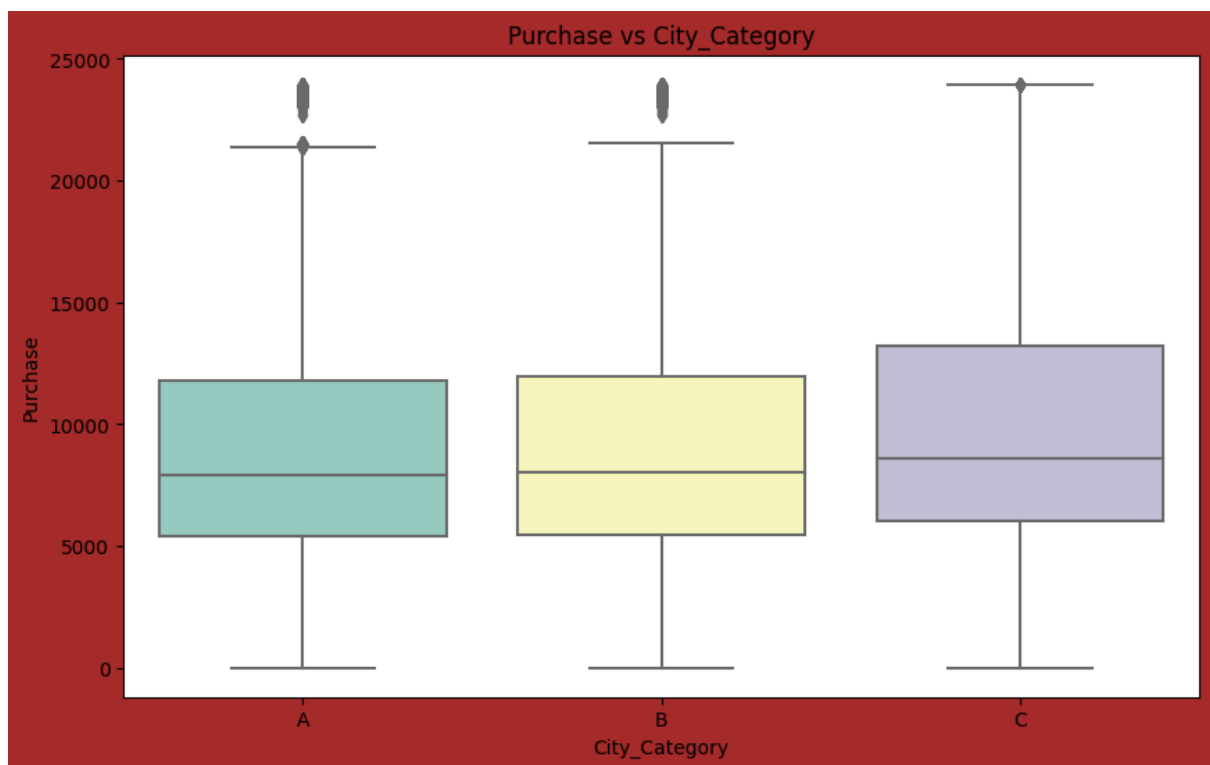
```
df.groupby(['Occupation'])['Purchase'].describe()
```

	count	mean	std	min	25%	50%	75%	max
Occupation								
0	69638.0	9124.428588	4971.757402	12.0	5445.00	8001.0	11957.00	23961.0
1	47426.0	8953.193270	4838.482159	12.0	5825.00	7966.0	11702.75	23960.0
2	26588.0	8952.481683	4939.418663	12.0	5419.00	7952.0	11718.00	23955.0
3	17650.0	9178.593088	5000.942719	12.0	5478.00	8008.0	11961.00	23914.0
4	72308.0	9213.980251	5043.674855	12.0	5441.75	8043.0	12034.00	23961.0
5	12177.0	9333.149298	5025.616603	12.0	5452.00	8080.0	12091.00	23924.0
6	20355.0	9256.535691	4989.216005	12.0	5888.00	8050.0	11971.50	23951.0
7	59133.0	9425.728223	5086.097089	12.0	5878.00	8069.0	12486.00	23948.0
8	1546.0	9532.592497	4916.641374	14.0	5961.75	8419.5	12607.00	23869.0
9	6291.0	8637.743761	4653.290986	13.0	5403.00	7886.0	11436.00	23943.0
10	12930.0	8959.355375	5124.339999	12.0	5326.25	8012.5	11931.75	23955.0
11	11586.0	9213.845848	5103.802992	12.0	5835.75	8041.5	12010.00	23946.0
12	31179.0	9796.640239	5140.437446	12.0	6054.00	8569.0	13239.00	23960.0
13	7728.0	9306.351061	4940.156591	12.0	6038.00	8090.5	11798.50	23959.0
14	27309.0	9500.702772	5069.600234	12.0	5922.00	8122.0	12508.00	23941.0
15	12165.0	9778.891163	5088.424301	12.0	6109.00	8513.0	13150.00	23949.0
16	25371.0	9394.464349	4995.918117	12.0	5917.00	8070.0	12218.50	23947.0
17	40043.0	9821.478236	5137.024383	12.0	6012.00	8635.0	13292.50	23961.0
18	6622.0	9169.655844	4987.697451	12.0	5420.00	7955.0	12062.75	23894.0
19	8461.0	8710.627231	5024.181000	12.0	5292.00	7840.0	11745.00	23939.0
20	33562.0	8836.494905	4919.662409	12.0	5389.00	7903.5	11677.00	23960.0

- But, here we can observe that the highest median value is for occupation 17
- The lowest median value is for occupation 19.
- Occupation 17 have the high average order values compared to other occupations which is 9821.
- Occupation 9 have the lowest average order value which is 8637.

Now, let's see city wise purchase habits

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



- **City Category c has the highest median value followed by city B and city A.**
- **There are a few outliers fro city A and B.**

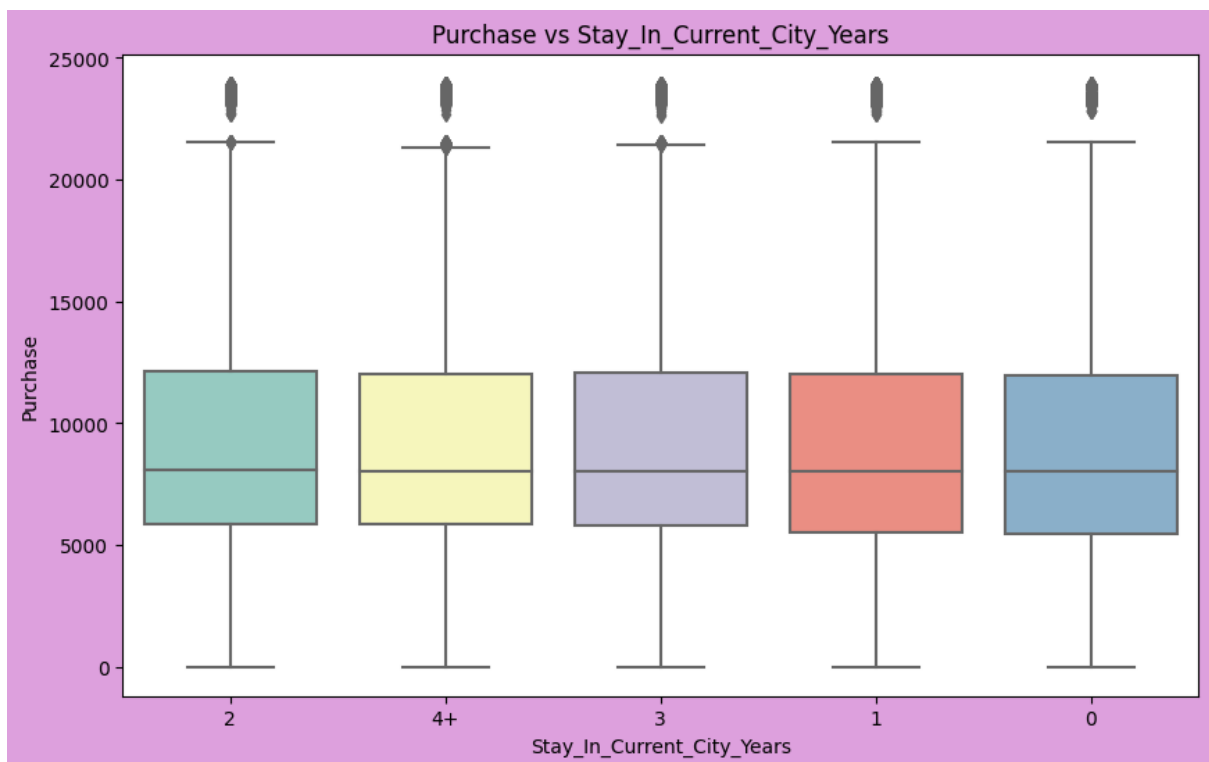
```
df.groupby(['City_Category'])['Purchase'].describe()
```

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

- **We can also observe that the mean value for a order is highest for city C followed by B and A.**

Lets see if stay years of a person in a city affects his/her purchase habits or not.

```
plt.figure(figsize = (10,6)).set_facecolor("plum")
sns.boxplot(data = df, y = 'Purchase', x =
'Stay_In_Current_City_Years', palette = 'Set3')
plt.title('Purchase vs Stay_In_Current_City_Years')
plt.show()
```



- We can see that the median value is almost the same for all the years.

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

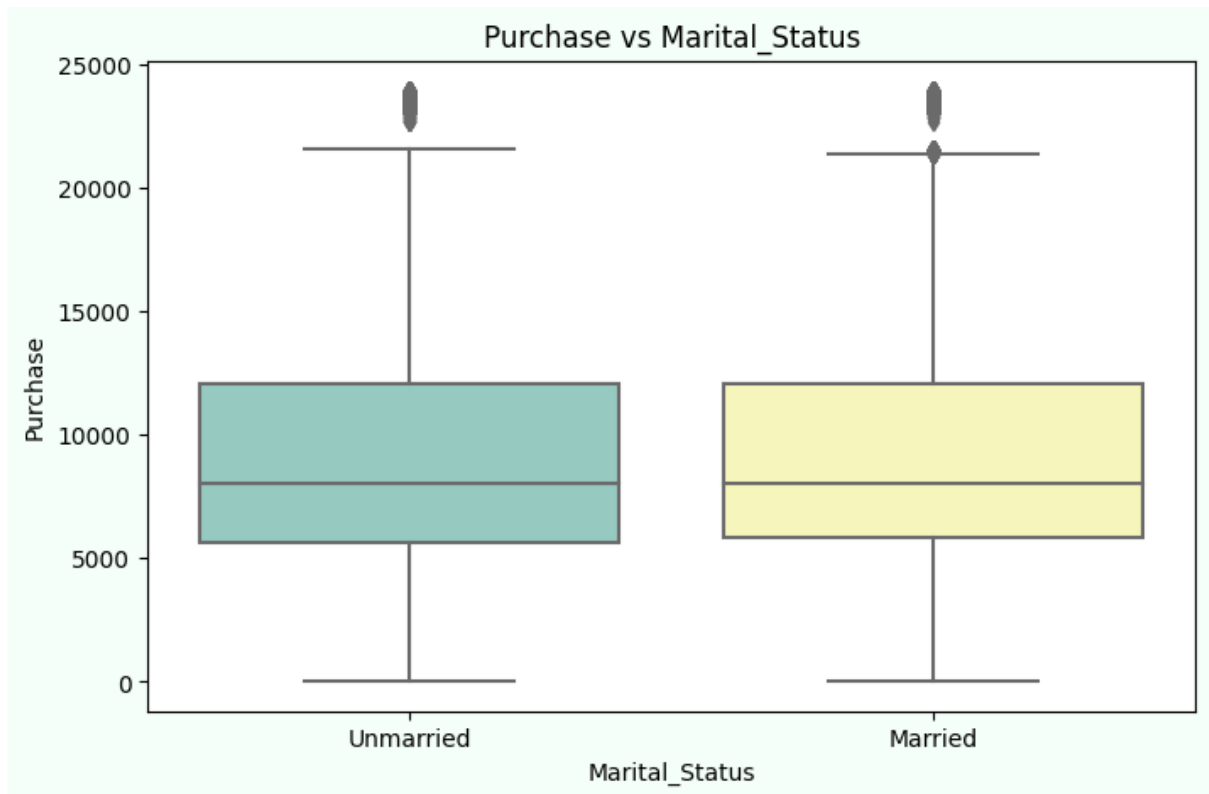

df.groupby(['Stay_In_Current_City_Years'])['Purchase'].describe()

	count	mean	std	min	25%	50%	75%	max
Stay_In_Current_City_Years								
0	74398.0	9180.075123	4990.479940	12.0	5480.0	8025.0	11990.0	23960.0
1	193821.0	9250.145923	5027.476933	12.0	5500.0	8041.0	12042.0	23961.0
2	101838.0	9320.429810	5044.588224	12.0	5846.0	8072.0	12117.0	23961.0
3	95285.0	9286.904119	5020.343541	12.0	5832.0	8047.0	12075.0	23961.0
4+	84726.0	9275.598872	5017.627594	12.0	5844.0	8052.0	12038.0	23958.0

- We can also see that the average order value is also almost the same which lies in the range of 9180 to 9286.
- One more thing we can observe here is that the highest order value is also the same for all the years.

Lets see if Marital Status affects the spending habits of a person

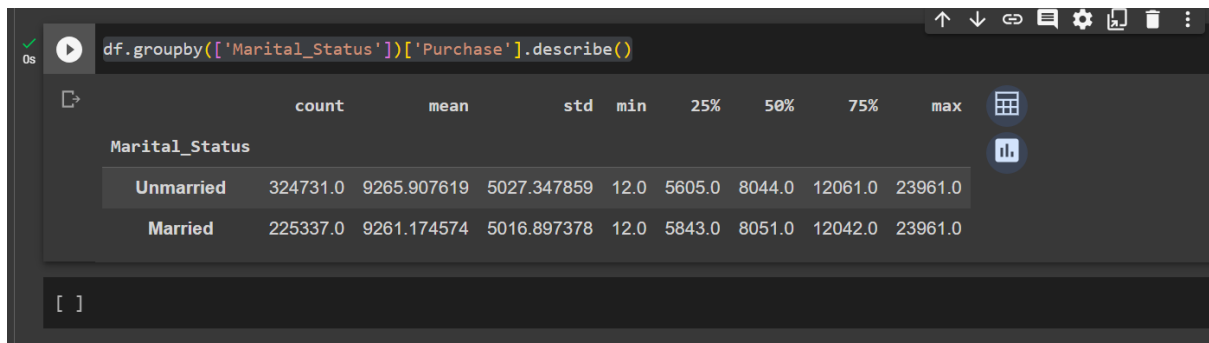
```
plt.figure(figsize = (8,5)).set_facecolor("mintcream")
sns.boxplot(data = df, y = 'Purchase', x = 'Marital_Status',
palette = 'Set3')
plt.title('Purchase vs Marital_Status')
plt.show()
```



- We can observe that the median value is almost the same.

Let's check the minimum, maximum and average order value.

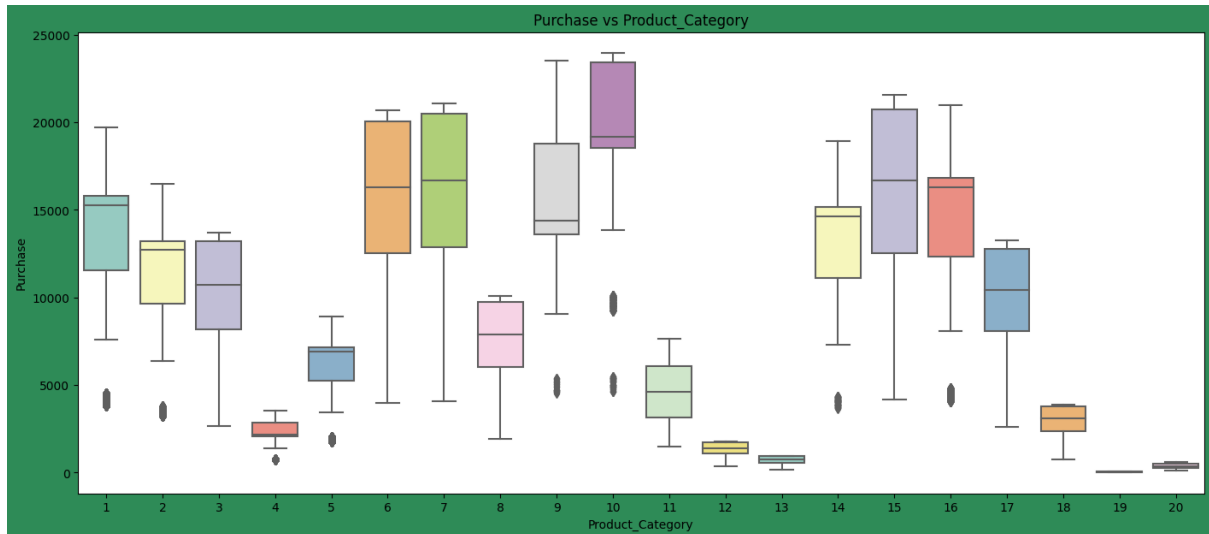
```
df.groupby(['Marital_Status'])['Purchase'].describe()
```



- The minimum and maximum order value is same for both types of people.
- We can observe that the average is also almost the same for both.

Lets see on which product category people spend more or less

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



We can clearly observe high differences in the median values for all the product categories.

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

	count	mean	std	min	25%	50%	75%	max
Product_Category								
1	140378.0	13606.218596	4298.834894	3790.0	11546.00	15245.0	15812.00	19708.0
2	23864.0	11251.935384	3570.642713	3176.0	9645.75	12728.5	13212.00	16504.0
3	20213.0	10096.705734	2824.626957	2638.0	8198.00	10742.0	13211.00	13717.0
4	11753.0	2329.659491	812.540292	684.0	2058.00	2175.0	2837.00	3556.0
5	150933.0	6240.088178	1909.091687	1713.0	5242.00	6912.0	7156.00	8907.0
6	20466.0	15838.478550	4011.233690	3981.0	12505.00	16312.0	20051.00	20690.0
7	3721.0	16365.689600	4174.554105	4061.0	12848.00	16700.0	20486.00	21080.0
8	113925.0	7498.958078	2013.015062	1939.0	6036.00	7905.0	9722.00	10082.0
9	410.0	15537.375610	5330.847116	4528.0	13583.50	14388.5	18764.00	23531.0
10	5125.0	19675.570927	4225.721898	4624.0	18546.00	19197.0	23438.00	23961.0
11	24287.0	4685.268456	1834.901184	1472.0	3131.00	4611.0	6058.00	7654.0
12	3947.0	1350.859894	362.510258	342.0	1071.00	1401.0	1723.00	1778.0
13	5549.0	722.400613	183.493126	185.0	578.00	755.0	927.00	962.0
14	1523.0	13141.625739	4069.009293	3657.0	11097.00	14654.0	15176.50	18931.0
15	6290.0	14780.451828	5175.465852	4148.0	12523.25	16660.0	20745.75	21569.0
16	9828.0	14766.037037	4360.213198	4036.0	12354.00	16292.5	16831.00	20971.0
17	578.0	10170.759516	2333.993073	2616.0	8063.50	10435.5	12776.75	13264.0
18	3125.0	2972.864320	727.051652	754.0	2359.00	3071.0	3769.00	3900.0
19	1603.0	37.041797	16.869148	12.0	24.00	37.0	50.00	62.0
20	2550.0	370.481176	167.116975	118.0	242.00	368.0	490.00	613.0

- The median value for product category 10 is the highest which is 19197.
- The median value for product category 19 is the lowest which is only 37.
- The average order value for category 10 is the highest which is 19675.
- The average order value for category 19 is also the lowest which is 37.
- Clearly, category 19 is the least preferred or least frequent bought product category.

Multi-variate Analysis

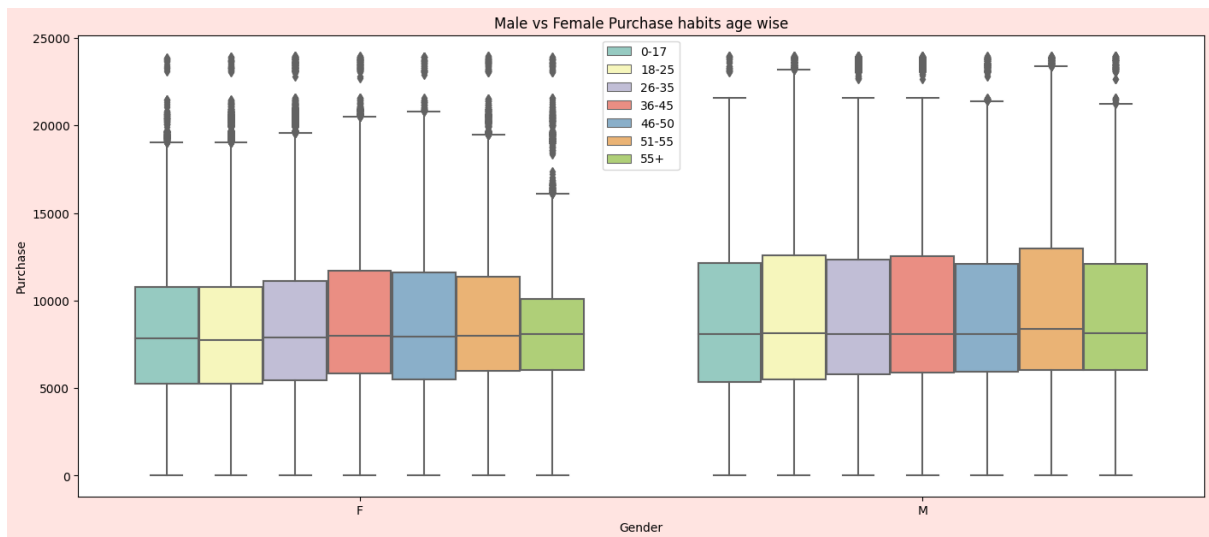
Lets see Male vs Female Purchase habits age wise.

```
plt.figure(figsize = (17,7)).set_facecolor("mistyrose")
```

```

sns.boxplot(data=df, y='Purchase', x='Gender', hue='Age',
palette='Set3')
plt.legend(loc=9)
plt.title('Male vs Female Purchase habits age wise')
plt.show()

```



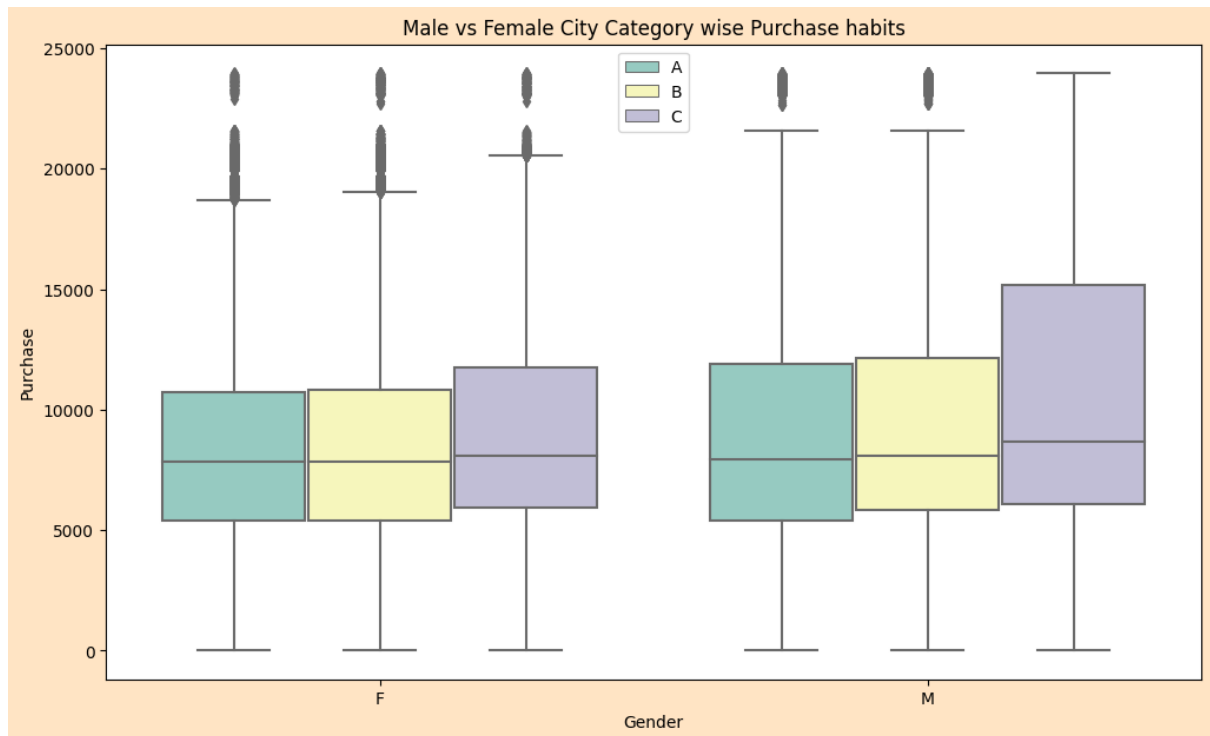
- The median values for 18-25 age females is the lowest and almost same for the rest.
- The median values for all age categories is almost the same and is highest for 51-55 age group.

Lets see Male vs Female City wise purchase habits

```

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

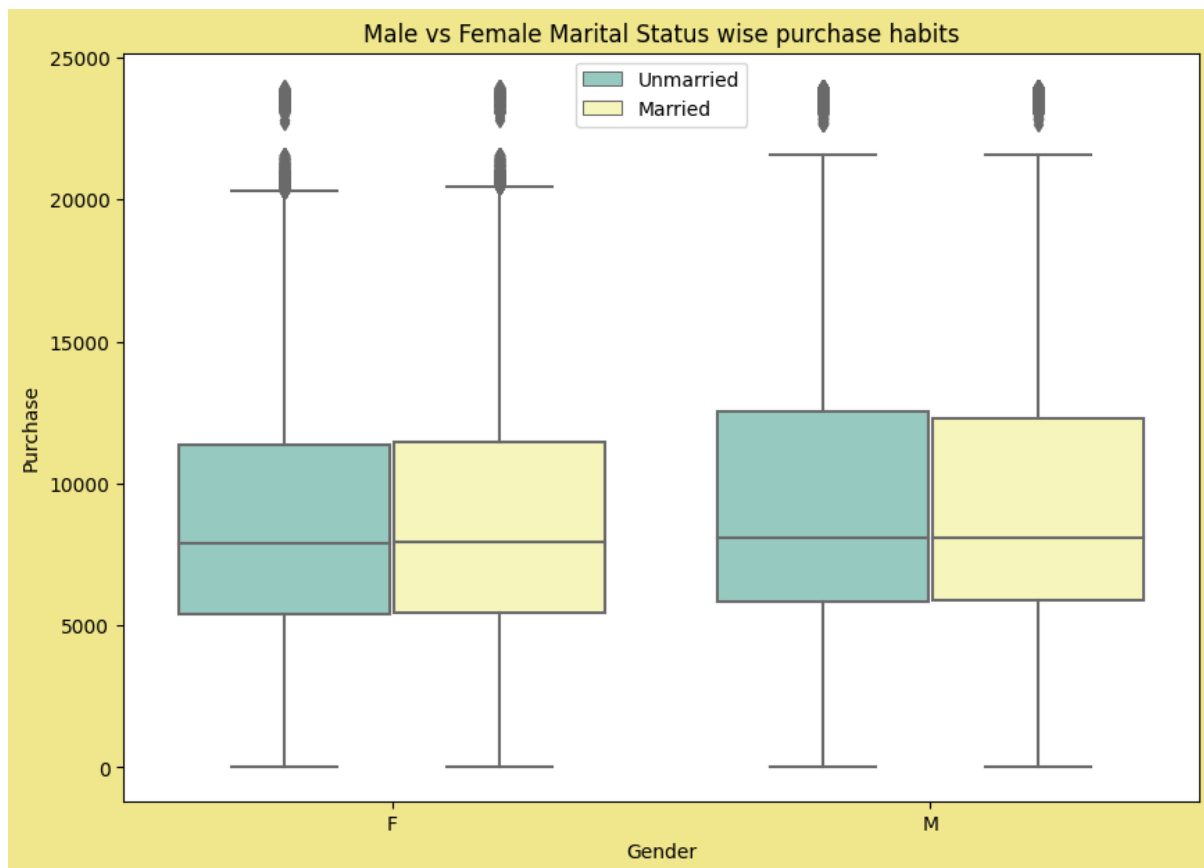
```



- The median value for females in city category C is highest compared to city A and B.
- The median value for males in city category C is also highest compared to city A and B.

Lets see Male vs Female Marital Status wise purchase habits.

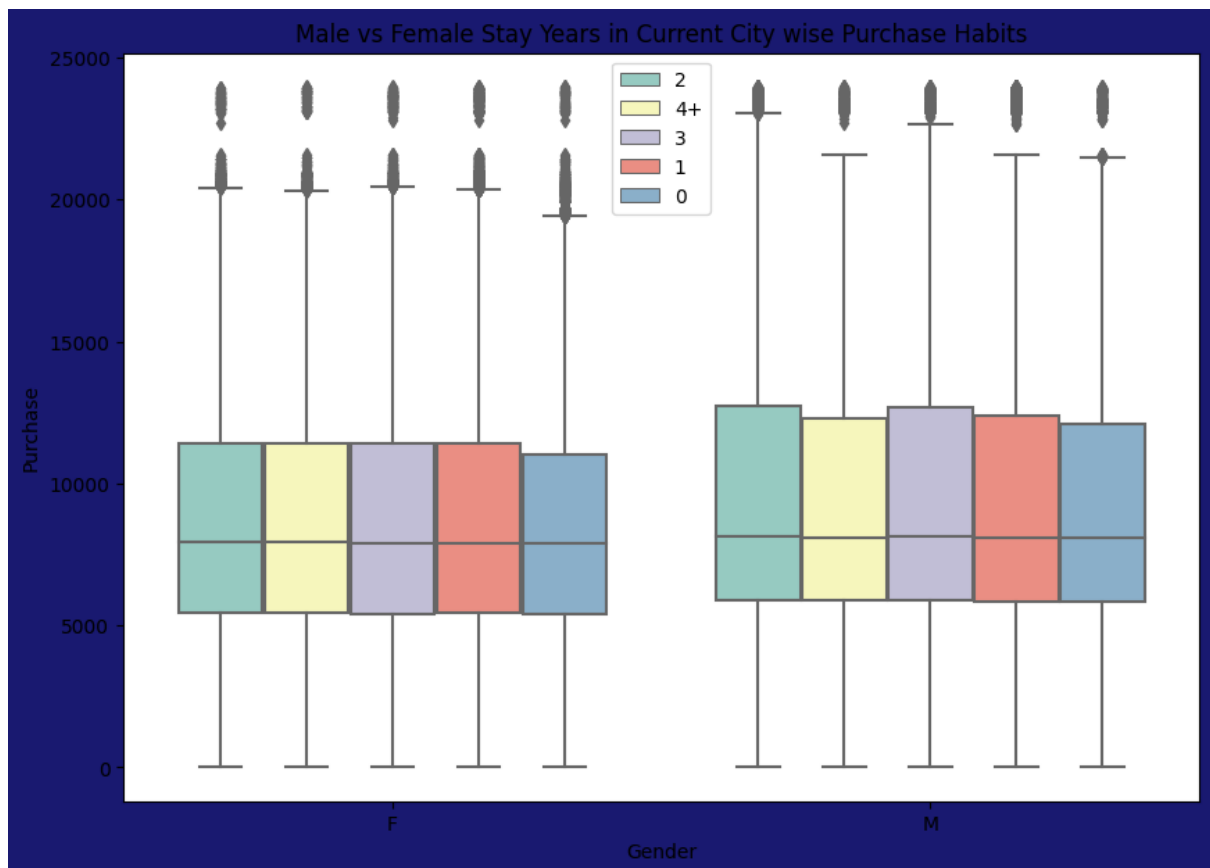
```
plt.figure(figsize = (10,7)).set_facecolor("khaki")
sns.boxplot(data=df, y='Purchase', x='Gender',
hue='Marital_Status', palette='Set3')
plt.legend(loc=9)
plt.title('Male vs Female Marital Status wise purchase
habits')
plt.show()
```



- There is no effect of marital status on the spending habits of both the genders.
- While we can observe that the median values for Male is higher compared to Females.

Lets see Male vs Female Stay Years in Current City wise Purchase Habits

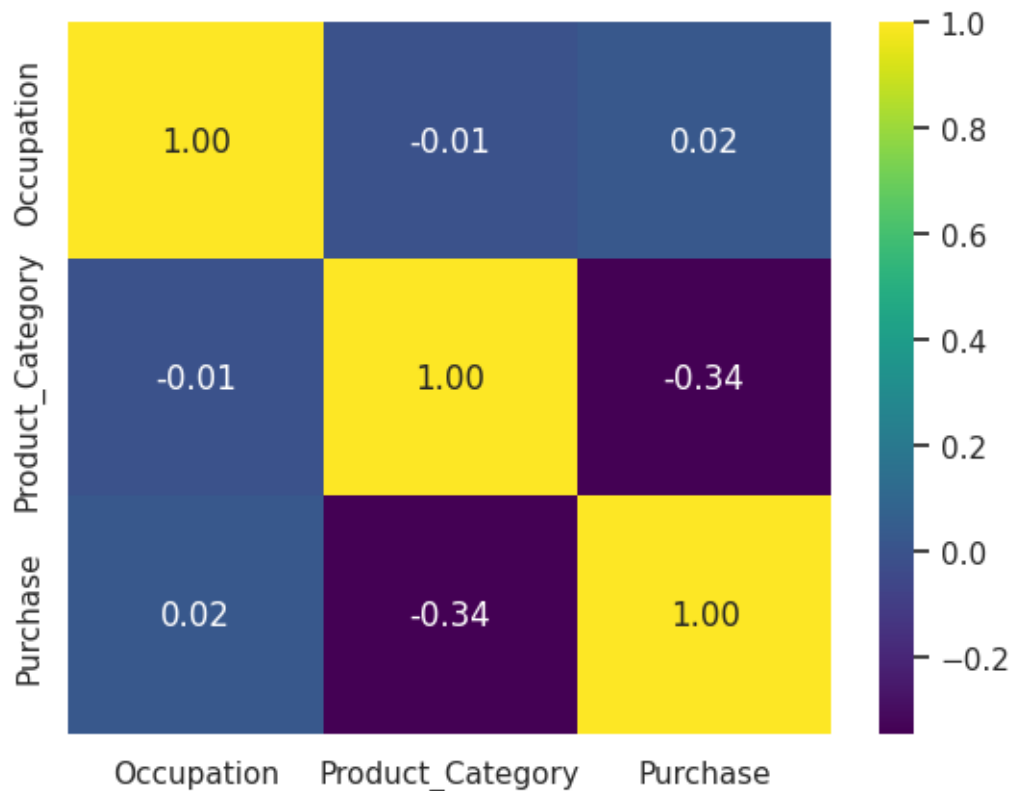
```
plt.figure(figsize = (10,7)).set_facecolor("midnightblue")
sns.boxplot(data=df, y='Purchase', x='Gender',
hue='Stay_In_Current_City_Years', palette='Set3')
plt.legend(loc=9)
plt.title('Male vs Female Stay Years in Current City wise
Purchase Habits')
plt.show()
```



- We can observe for females the median values for purchase amount is a little lower for women staying for 3 and 0 years as compared to others.
- For men, there is no much difference.

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

```
sns.heatmap(df.corr(), annot =
True,cmap='viridis',fmt='.2f')
plt.show()
```

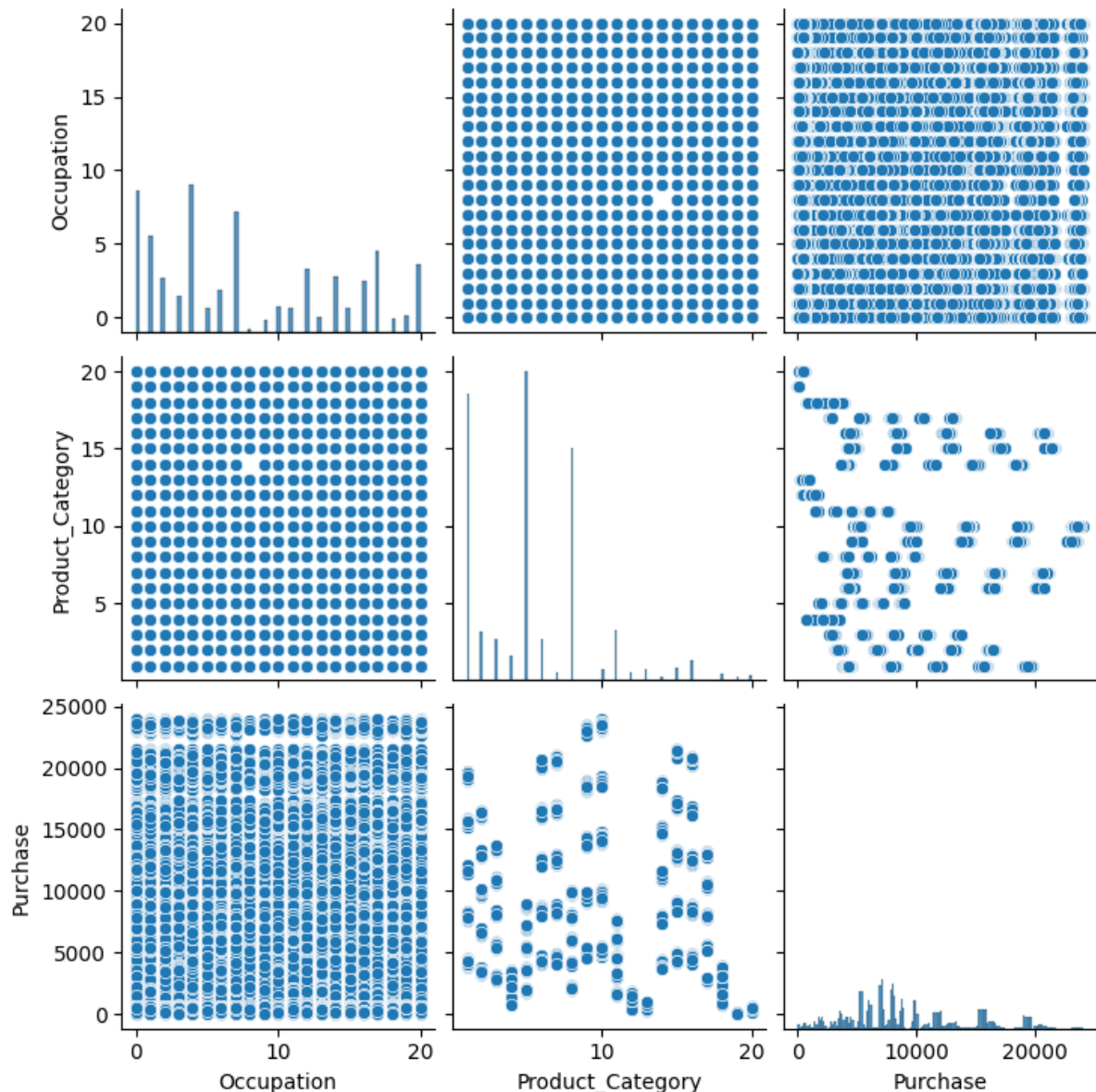



We can observe that there is

- **High Negative Correlation(-0.0076) between Product Category and Occupation.**
- **Slight Positive Correlation(0.021) between Purchase and Occupation.**
- **Negative Correlation(-0.34) between Product Category and Purchase.**

Lets plot the pairplot and see relations between the columns.

```
sns.pairplot(df)  
plt.show()
```



Sample Analysis Using Central Limit Theorem and Confidence Interval

CLT and CI analysis for Gender

CLT and CI analysis for Male customers

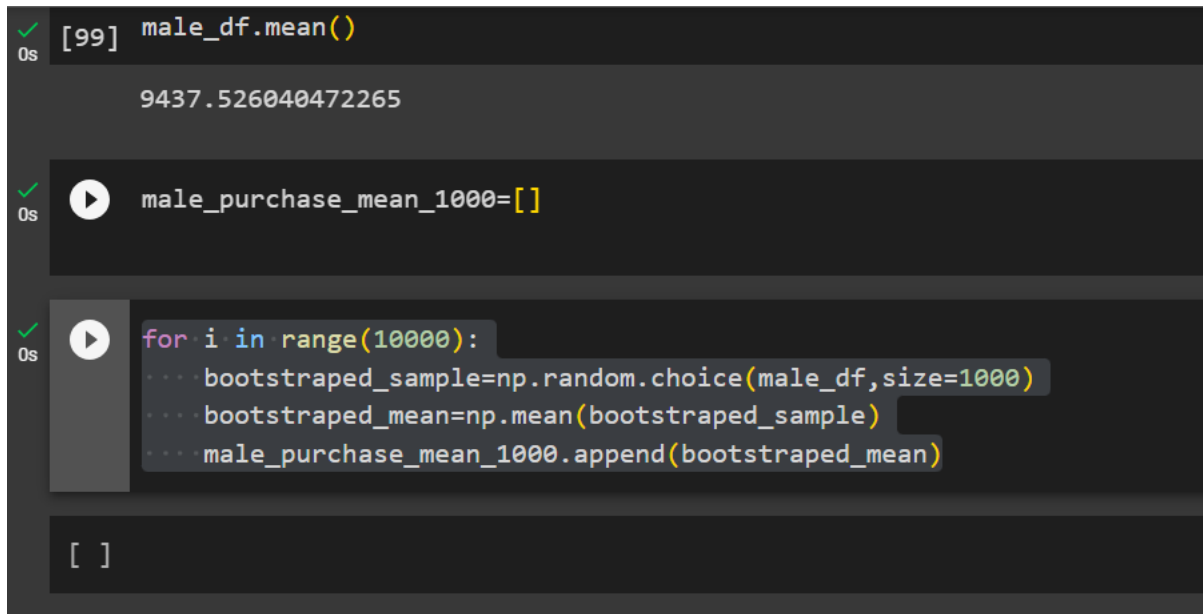
Creating a Samples of size 1000 and computing means through bootstrapping

```
male_df=df.loc[df['Gender']=='M']['Purchase']
male_df.mean()
```

```

for i in range(10000):
    bootstrapped_sample=np.random.choice(male_df,size=1000)
    bootstrapped_mean=np.mean(bootstrapped_sample)
    male_purchase_mean_1000.append(bootstrapped_mean)

```



A screenshot of a Jupyter Notebook interface with a dark theme. It shows three code cells. The first cell contains `male_df.mean()` and the output is `9437.526040472265`. The second cell contains `male_purchase_mean_1000=[]`. The third cell contains a `for` loop that bootstraps 10,000 samples, calculates their means, and appends them to `male_purchase_mean_1000`. The output of the third cell is an empty list `[]`.

```

[99] male_df.mean()
9437.526040472265

male_purchase_mean_1000=[]

for i in range(10000):
    bootstrapped_sample=np.random.choice(male_df,size=1000)
    bootstrapped_mean=np.mean(bootstrapped_sample)
    male_purchase_mean_1000.append(bootstrapped_mean)

[ ]

```

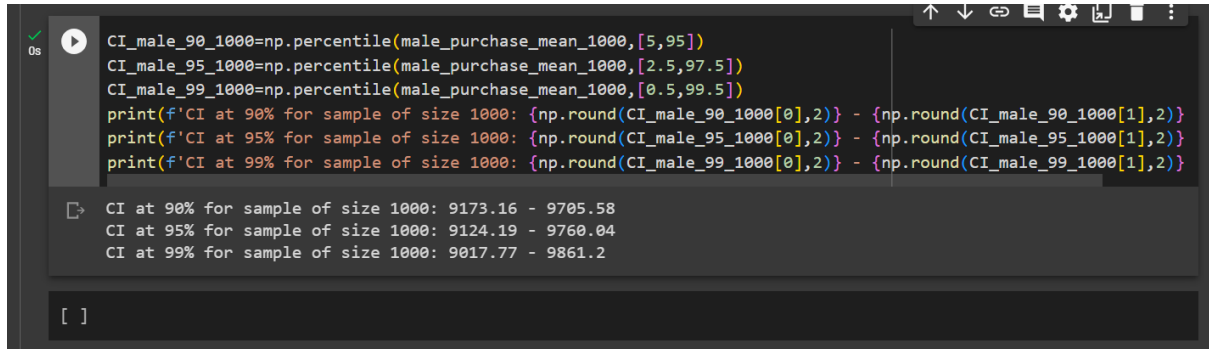
Calculating CI at 90%,95% and 99% for sample of size 1000

```

CI_male_90_1000=np.percentile(male_purchase_mean_1000,[5,95])
CI_male_95_1000=np.percentile(male_purchase_mean_1000,[2.5,97.5])
CI_male_99_1000=np.percentile(male_purchase_mean_1000,[0.5,99.5])
print(f'CI at 90% for sample of size 1000:
{np.round(CI_male_90_1000[0],2)} -
{np.round(CI_male_90_1000[1],2)}')
print(f'CI at 95% for sample of size 1000:
{np.round(CI_male_95_1000[0],2)} -
{np.round(CI_male_95_1000[1],2)}')

```

```
print(f'CI at 99% for sample of size 1000:  
{np.round(CI_male_99_1000[0],2)} -  
{np.round(CI_male_99_1000[1],2)}')
```

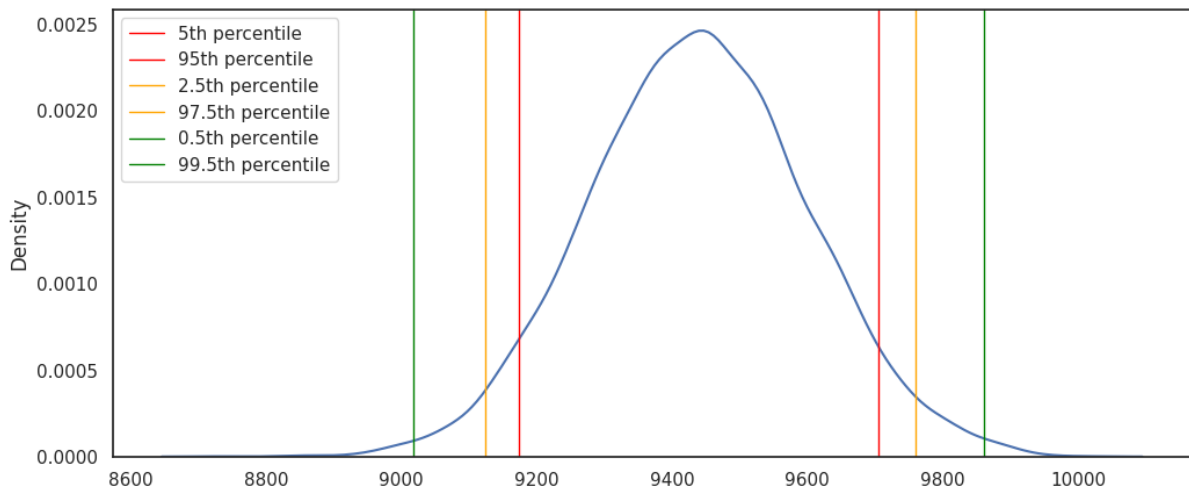


```
CI_male_90_1000=np.percentile(male_purchase_mean_1000,[5,95])  
CI_male_95_1000=np.percentile(male_purchase_mean_1000,[2.5,97.5])  
CI_male_99_1000=np.percentile(male_purchase_mean_1000,[0.5,99.5])  
print(f'CI at 90% for sample of size 1000: {np.round(CI_male_90_1000[0],2)} - {np.round(CI_male_90_1000[1],2)}')  
print(f'CI at 95% for sample of size 1000: {np.round(CI_male_95_1000[0],2)} - {np.round(CI_male_95_1000[1],2)}')  
print(f'CI at 99% for sample of size 1000: {np.round(CI_male_99_1000[0],2)} - {np.round(CI_male_99_1000[1],2)}')
```

```
CI at 90% for sample of size 1000: 9173.16 - 9705.58  
CI at 95% for sample of size 1000: 9124.19 - 9760.04  
CI at 99% for sample of size 1000: 9017.77 - 9861.2
```

VISUALISING CI FOR THE SAMPLE SIZE OF 1000

```
plt.figure(figsize=(12,5))  
sns.kdeplot(male_purchase_mean_1000)  
plt.axvline(x=np.percentile(male_purchase_mean_1000,[5]),ymin=0,ymax=1,color='red',linewidth=1.0,label='5th percentile')  
plt.axvline(x=np.percentile(male_purchase_mean_1000,[95]),ymin=0,ymax=1,color='red',linewidth=1.0,label='95th percentile')  
plt.axvline(x=np.percentile(male_purchase_mean_1000,[2.5]),ymin=0,ymax=1,color='orange',linewidth=1.0,label='2.5th percentile')  
plt.axvline(x=np.percentile(male_purchase_mean_1000,[97.5]),ymin=0,ymax=1,color='orange',linewidth=1.0,label='97.5th percentile')  
plt.axvline(x=np.percentile(male_purchase_mean_1000,[0.5]),ymin=0,ymax=1,color='green',linewidth=1.0,label='0.5th percentile')  
plt.axvline(x=np.percentile(male_purchase_mean_1000,[99.5]),ymin=0,ymax=1,color='green',linewidth=1.0,label='99.5th percentile')  
plt.legend()  
plt.show()
```



Repeating the analysis for sample of size 2500 and 5000

Sample for 2500

```
male_purchase_mean_2500=[]
for i in range(10000):
    bootstrapped_sample=np.random.choice(male_df,size=2500)
    bootstrapped_mean=np.mean(bootstrapped_sample)
    male_purchase_mean_2500.append(bootstrapped_mean)
```

SAMPLE FOR 5000

```
male_purchase_mean_5000=[]
for i in range(10000):
    bootstrapped_sample=np.random.choice(male_df,size=5000)
    bootstrapped_mean=np.mean(bootstrapped_sample)
    male_purchase_mean_5000.append(bootstrapped_mean)
```

```
CI_male_90_2500=np.percentile(male_purchase_mean_2500,[5,95])
CI_male_95_2500=np.percentile(male_purchase_mean_2500,[2.5,97.5])
CI_male_99_2500=np.percentile(male_purchase_mean_2500,[0.5,99.5])
CI_male_90_5000=np.percentile(male_purchase_mean_5000,[5,95])
CI_male_95_5000=np.percentile(male_purchase_mean_5000,[2.5,97.5])
CI_male_99_5000=np.percentile(male_purchase_mean_5000,[0.5,99.5])
print(f'CI at 90% for sample of size 2500: {np.round(CI_male_90_2500[0],2)} - {np.round(CI_male_90_2500[1],2)}')
print(f'CI at 95% for sample of size 2500: {np.round(CI_male_95_2500[0],2)} - {np.round(CI_male_95_2500[1],2)}')
print(f'CI at 99% for sample of size 2500: {np.round(CI_male_99_2500[0],2)} - {np.round(CI_male_99_2500[1],2)}')
```

```

print(f'CI at 90% for sample of size 5000: {np.round(CI_male_90_5000[0],2)} - {np.round(CI_male_90_5000[1],2)}')
print(f'CI at 95% for sample of size 5000: {np.round(CI_male_95_5000[0],2)} - {np.round(CI_male_95_5000[1],2)}')
print(f'CI at 99% for sample of size 5000: {np.round(CI_male_99_5000[0],2)} - {np.round(CI_male_99_5000[1],2)}')

```

```

CI_male_90_2500=np.percentile(male_purchase_mean_2500,[5,95])
CI_male_95_2500=np.percentile(male_purchase_mean_2500,[2.5,97.5])
CI_male_99_2500=np.percentile(male_purchase_mean_2500,[0.5,99.5])
CI_male_90_5000=np.percentile(male_purchase_mean_5000,[5,95])
CI_male_95_5000=np.percentile(male_purchase_mean_5000,[2.5,97.5])
CI_male_99_5000=np.percentile(male_purchase_mean_5000,[0.5,99.5])
print(f'CI at 90% for sample of size 2500: {np.round(CI_male_90_2500[0],2)} - {np.round(CI_male_90_2500[1],2)}')
print(f'CI at 95% for sample of size 2500: {np.round(CI_male_95_2500[0],2)} - {np.round(CI_male_95_2500[1],2)}')
print(f'CI at 99% for sample of size 2500: {np.round(CI_male_99_2500[0],2)} - {np.round(CI_male_99_2500[1],2)}')
print(f'CI at 90% for sample of size 5000: {np.round(CI_male_90_5000[0],2)} - {np.round(CI_male_90_5000[1],2)}')
print(f'CI at 95% for sample of size 5000: {np.round(CI_male_95_5000[0],2)} - {np.round(CI_male_95_5000[1],2)}')
print(f'CI at 99% for sample of size 5000: {np.round(CI_male_99_5000[0],2)} - {np.round(CI_male_99_5000[1],2)}')

```

```

CI at 90% for sample of size 2500: 9267.59 - 9603.63
CI at 95% for sample of size 2500: 9233.01 - 9632.69
CI at 99% for sample of size 2500: 9171.25 - 9697.3
CI at 90% for sample of size 5000: 9321.47 - 9555.68
CI at 95% for sample of size 5000: 9300.31 - 9580.59
CI at 99% for sample of size 5000: 9254.2 - 9625.41

```

Visualising bootstrapped means through histogram in order to check that it follows normal distribution

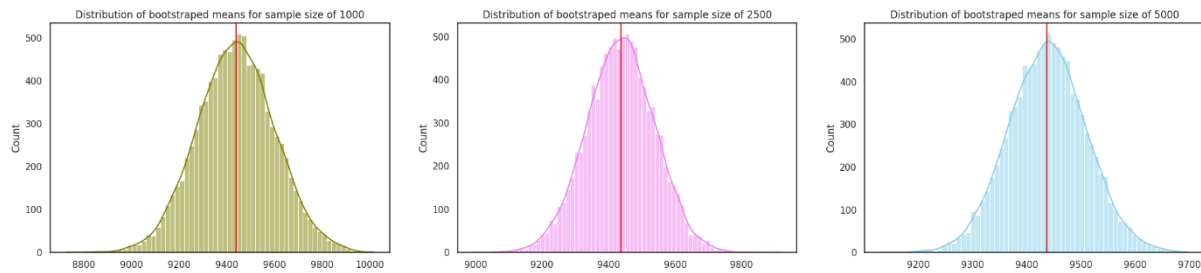
```

plt.figure(figsize=(25,5))
plt.subplot(1,3,1)
sns.histplot(male_purchase_mean_1000,kde=True,color='olive')
plt.axvline(male_df.mean(),color='red',linestyle='-',linewidth=1.5)
plt.title('Distribution of bootstrapped means for sample size of 1000')

plt.subplot(1,3,2)
sns.histplot(male_purchase_mean_2500,kde=True,color='violet')
plt.axvline(male_df.mean(),color='red',linestyle='-',linewidth=1.5)
plt.title('Distribution of bootstrapped means for sample size of 2500')

plt.subplot(1,3,3)
sns.histplot(male_purchase_mean_5000,kde=True,color='skyblue')
plt.axvline(male_df.mean(),color='red',linestyle='-',linewidth=1.5)
plt.title('Distribution of bootstrapped means for sample size of 5000')
plt.show()

```



1. In accordance with the Central Limit Theorem, the distribution of sample means tends to become normal, even when the population distribution does not exhibit normality. This phenomenon is clearly evident in this scenario, as the distribution of bootstrapped means conforms to a normal distribution, despite the population mean not adhering to such a distribution.
2. As the sample size increases, the distribution of sample means becomes closer to a normal distribution.

Calculating the standard error for sample sizes of 1000, 2500, and 5000.

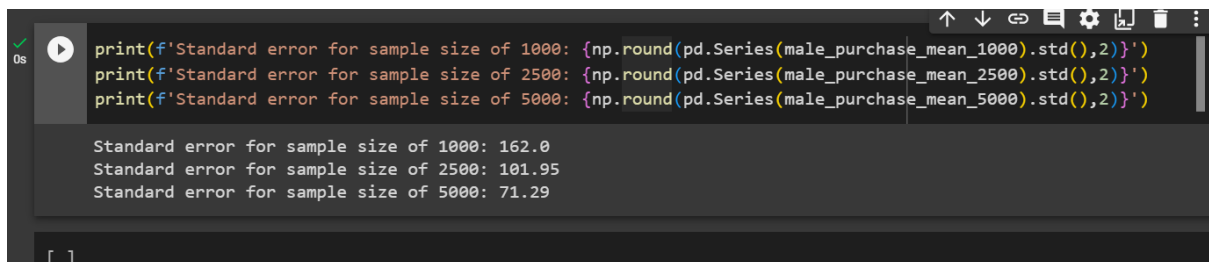
```
SE_male_1000 = (male_df.std()/np.sqrt(1000))
SE_male_2500 = (male_df.std()/np.sqrt(2500))
SE_male_5000 = (male_df.std()/np.sqrt(5000))
print(f'Standard error for sample size of 1000: {np.round(SE_male_1000,2)}')
print(f'Standard error for sample size of 2500: {np.round(SE_male_2500,2)}')
print(f'Standard error for sample size of 5000: {np.round(SE_male_5000,2)}')
```

```
SE_male_1000 = (male_df.std()/np.sqrt(1000))
SE_male_2500 = (male_df.std()/np.sqrt(2500))
SE_male_5000 = (male_df.std()/np.sqrt(5000))
print(f'Standard error for sample size of 1000: {np.round(SE_male_1000,2)}')
print(f'Standard error for sample size of 2500: {np.round(SE_male_2500,2)}')
print(f'Standard error for sample size of 5000: {np.round(SE_male_5000,2)}')
```

Standard error for sample size of 1000: 161.03
Standard error for sample size of 2500: 101.84
Standard error for sample size of 5000: 72.01

```
print(f'Standard error for sample size of 1000: {np.round(pd.Series(male_purchase_mean_1000).std(),2)}')
print(f'Standard error for sample size of 2500: {np.round(pd.Series(male_purchase_mean_2500).std(),2)}')
```

```
print(f'Standard error for sample size of 5000:  
{np.round(pd.Series(male_purchase_mean_5000).std(),2)}')
```



A screenshot of a Jupyter Notebook interface. The top bar shows a green checkmark, a play button, and a '0s' timer. The code cell contains three print statements for sample sizes 1000, 2500, and 5000. The output cell shows the results: 162.0 for 1000, 101.95 for 2500, and 71.29 for 5000. The bottom of the notebook shows an empty list '[]'.

```
print(f'Standard error for sample size of 1000: {np.round(pd.Series(male_purchase_mean_1000).std(),2)}')
```

```
print(f'Standard error for sample size of 2500: {np.round(pd.Series(male_purchase_mean_2500).std(),2)}')
```

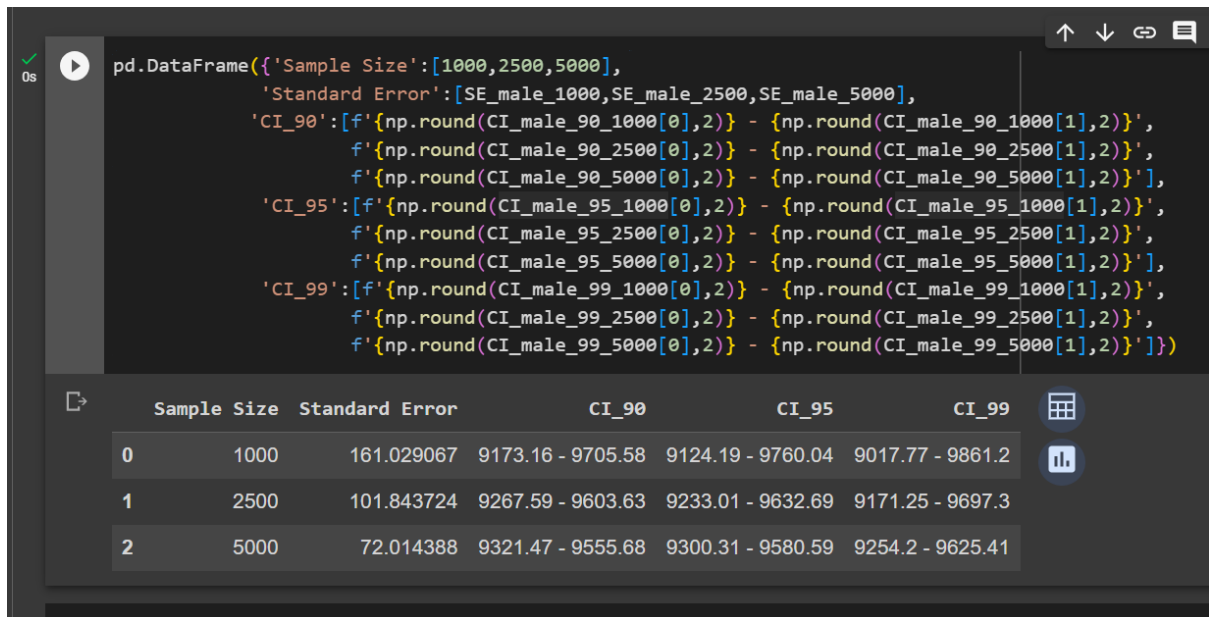
```
print(f'Standard error for sample size of 5000: {np.round(pd.Series(male_purchase_mean_5000).std(),2)}')
```

Standard error for sample size of 1000: 162.0
Standard error for sample size of 2500: 101.95
Standard error for sample size of 5000: 71.29

[]

Visualising all analysis using a table

```
pd.DataFrame({'Sample Size':[1000,2500,5000],  
             'Standard Error':[SE_male_1000,SE_male_2500,SE_male_5000],  
             'CI_90':[f'{np.round(CI_male_90_1000[0],2)} -  
{np.round(CI_male_90_1000[1],2)}',  
                      f'{np.round(CI_male_90_2500[0],2)} -  
{np.round(CI_male_90_2500[1],2)}',  
                      f'{np.round(CI_male_90_5000[0],2)} -  
{np.round(CI_male_90_5000[1],2)}'],  
             'CI_95':[f'{np.round(CI_male_95_1000[0],2)} -  
{np.round(CI_male_95_1000[1],2)}',  
                      f'{np.round(CI_male_95_2500[0],2)} -  
{np.round(CI_male_95_2500[1],2)}',  
                      f'{np.round(CI_male_95_5000[0],2)} -  
{np.round(CI_male_95_5000[1],2)}'],  
             'CI_99':[f'{np.round(CI_male_99_1000[0],2)} -  
{np.round(CI_male_99_1000[1],2)}',  
                      f'{np.round(CI_male_99_2500[0],2)} -  
{np.round(CI_male_99_2500[1],2)}',  
                      f'{np.round(CI_male_99_5000[0],2)} -  
{np.round(CI_male_99_5000[1],2)}']])
```

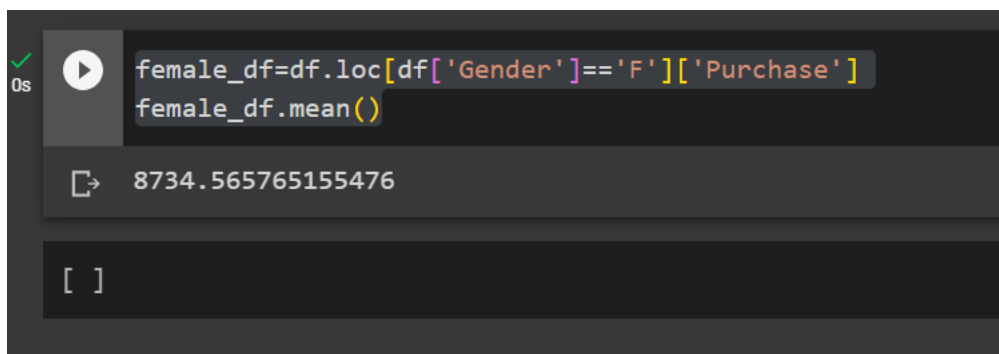



1. As the sample size increases, the standard error decreases which essentially translates to the range decreasing between which the population mean lies.
2. The standard error of the mean will approach zero with the increasing number of observations in the sample, as the sample becomes more and more representative of the population, and the sample mean approaches the actual population mean.

CLT and CI analysis for Female customers

Creating a Samples of size 1000 and computing means through bootstrapping

```
female_df=df.loc[df['Gender']=='F']['Purchase']
female_df.mean()
```



```
female_purchase_mean_1000=[]
for i in range(10000):
```

```

bootstrapped_sample=np.random.choice(female_df,size=1000)
bootstrapped_mean=np.mean(bootstrapped_sample)
female_purchase_mean_1000.append(bootstrapped_mean)

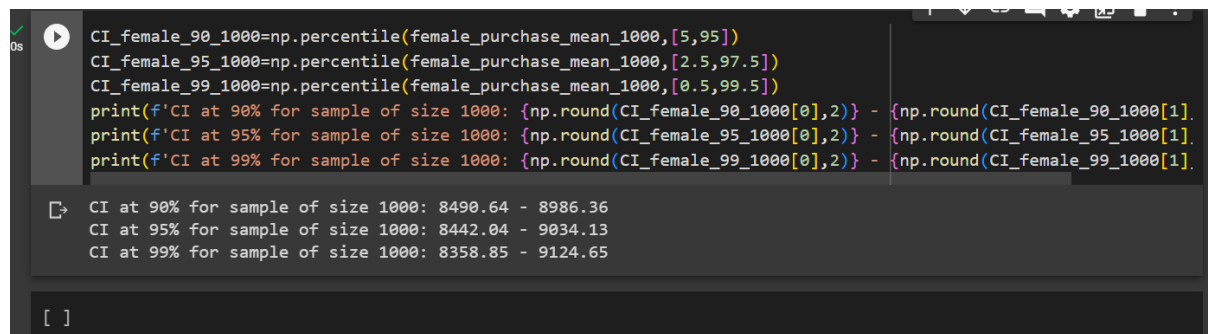
```

Calculating CI at 90%,95% and 99% for sample of size 1000

```

CI_female_90_1000=np.percentile(female_purchase_mean_1000,[5,95])
CI_female_95_1000=np.percentile(female_purchase_mean_1000,[2.5,97.5])
CI_female_99_1000=np.percentile(female_purchase_mean_1000,[0.5,99.5])
print(f'CI at 90% for sample of size 1000: {np.round(CI_female_90_1000[0],2)} - {np.round(CI_female_90_1000[1],2)}')
print(f'CI at 95% for sample of size 1000: {np.round(CI_female_95_1000[0],2)} - {np.round(CI_female_95_1000[1],2)}')
print(f'CI at 99% for sample of size 1000: {np.round(CI_female_99_1000[0],2)} - {np.round(CI_female_99_1000[1],2)}')

```



```

CI_female_90_1000=np.percentile(female_purchase_mean_1000,[5,95])
CI_female_95_1000=np.percentile(female_purchase_mean_1000,[2.5,97.5])
CI_female_99_1000=np.percentile(female_purchase_mean_1000,[0.5,99.5])
print(f'CI at 90% for sample of size 1000: {np.round(CI_female_90_1000[0],2)} - {np.round(CI_female_90_1000[1],2)}')
print(f'CI at 95% for sample of size 1000: {np.round(CI_female_95_1000[0],2)} - {np.round(CI_female_95_1000[1],2)}')
print(f'CI at 99% for sample of size 1000: {np.round(CI_female_99_1000[0],2)} - {np.round(CI_female_99_1000[1],2)}')

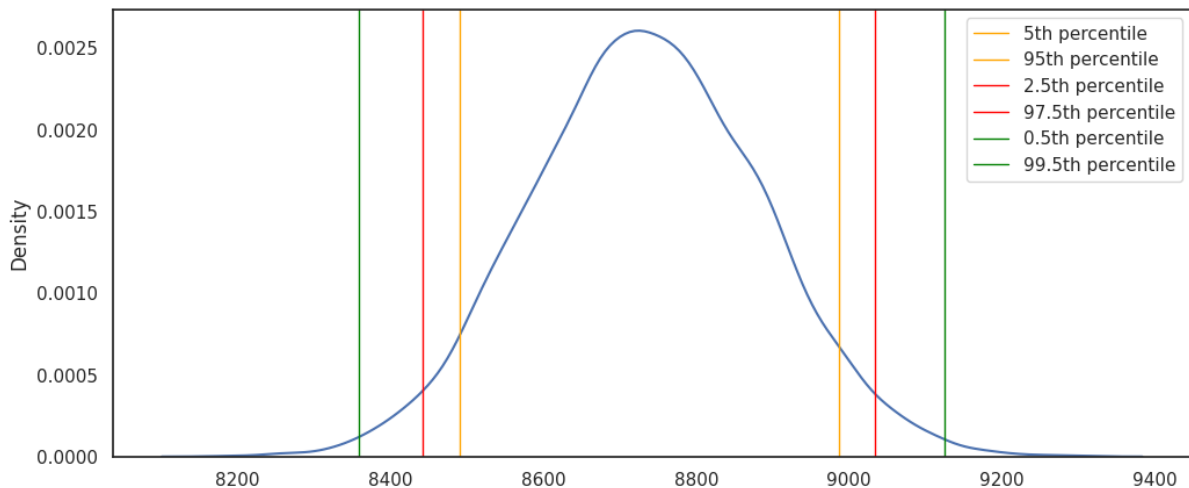
```

CI at 90% for sample of size 1000: 8490.64 - 8986.36
 CI at 95% for sample of size 1000: 8442.04 - 9034.13
 CI at 99% for sample of size 1000: 8358.85 - 9124.65

```

plt.figure(figsize=(12,5))
sns.kdeplot(female_purchase_mean_1000)
plt.axvline(x=np.percentile(female_purchase_mean_1000,[5]),ymin=0,ymax=1,
color='orange',linewidth=1.0,label='5th percentile')
plt.axvline(x=np.percentile(female_purchase_mean_1000,[95]),ymin=0,ymax=1,
color='orange',linewidth=1.0,label='95th percentile')
plt.axvline(x=np.percentile(female_purchase_mean_1000,[2.5]),ymin=0,ymax=1,
color='red',linewidth=1.0,label='2.5th percentile')
plt.axvline(x=np.percentile(female_purchase_mean_1000,[97.5]),ymin=0,ymax=1,
color='red',linewidth=1.0,label='97.5th percentile')
plt.axvline(x=np.percentile(female_purchase_mean_1000,[0.5]),ymin=0,ymax=1,
color='green',linewidth=1.0,label='0.5th percentile')
plt.axvline(x=np.percentile(female_purchase_mean_1000,[99.5]),ymin=0,ymax=1,
color='green',linewidth=1.0,label='99.5th percentile')
plt.legend()
plt.show()

```



Repeating the analysis for sample of sizes 2500 and 5000

Bootstrapping means for sample of size 2500

```
female_purchase_mean_2500=[]
```

```
for i in range(10000):
```

```
    bootstrapped_sample=np.random.choice(female_df,size=2500)
```

```
    bootstrapped_mean=np.mean(bootstrapped_sample)
```

```
    female_purchase_mean_2500.append(bootstrapped_mean)
```

Bootstrapping means for sample of size 5000

```
female_purchase_mean_5000=[]
```

```
for i in range(10000):
```

```
    bootstrapped_sample=np.random.choice(female_df,size=5000)
```

```
    bootstrapped_mean=np.mean(bootstrapped_sample)
```

```
    female_purchase_mean_5000.append(bootstrapped_mean)
```

```
CI_female_90_2500=np.percentile(female_purchase_mean_2500,[5,95])
```

```
CI_female_95_2500=np.percentile(female_purchase_mean_2500,[2.5,97.5])
```

```
CI_female_99_2500=np.percentile(female_purchase_mean_2500,[0.5,99.5])
```

```
CI_female_90_5000=np.percentile(female_purchase_mean_5000,[5,95])
```

```
CI_female_95_5000=np.percentile(female_purchase_mean_5000,[2.5,97.5])
```

```
CI_female_99_5000=np.percentile(female_purchase_mean_5000,[0.5,99.5])
```

```
print(f'CI at 90% for sample of size 2500: {np.round(CI_female_90_2500[0],2)} - {np.round(CI_female_90_2500[1],2)}')
```

```
print(f'CI at 95% for sample of size 2500: {np.round(CI_female_95_2500[0],2)} - {np.round(CI_female_95_2500[1],2)}')
```

```
print(f'CI at 99% for sample of size 2500: {np.round(CI_female_99_2500[0],2)} - {np.round(CI_female_99_2500[1],2)}')
```

```
print(f'CI at 90% for sample of size 5000: {np.round(CI_female_90_5000[0],2)} - {np.round(CI_female_90_5000[1],2)}')
```

```
print(f'CI at 95% for sample of size 5000: {np.round(CI_female_95_5000[0],2)} - {np.round(CI_female_95_5000[1],2)}')
```

```
print(f'CI at 99% for sample of size 5000: {np.round(CI_female_99_5000[0],2)} - {np.round(CI_female_99_5000[1],2)}')
```

```

_90_2500=np.percentile(female_purchase_mean_2500,[5,95])
_95_2500=np.percentile(female_purchase_mean_2500,[2.5,97.5])
_99_2500=np.percentile(female_purchase_mean_2500,[0.5,99.5])
_90_5000=np.percentile(female_purchase_mean_5000,[5,95])
_95_5000=np.percentile(female_purchase_mean_5000,[2.5,97.5])
_99_5000=np.percentile(female_purchase_mean_5000,[0.5,99.5])
[ at 90% for sample of size 2500: {np.round(CI_female_90_2500[0],2)} - {np.round(CI_female_90_2500[1],2)}']
[ at 95% for sample of size 2500: {np.round(CI_female_95_2500[0],2)} - {np.round(CI_female_95_2500[1],2)}']
[ at 99% for sample of size 2500: {np.round(CI_female_99_2500[0],2)} - {np.round(CI_female_99_2500[1],2)}']
[ at 90% for sample of size 5000: {np.round(CI_female_90_5000[0],2)} - {np.round(CI_female_90_5000[1],2)}']
[ at 95% for sample of size 5000: {np.round(CI_female_95_5000[0],2)} - {np.round(CI_female_95_5000[1],2)}']
[ at 99% for sample of size 5000: {np.round(CI_female_99_5000[0],2)} - {np.round(CI_female_99_5000[1],2)}']

CI at 90% for sample of size 2500: 8577.81 - 8888.74
CI at 95% for sample of size 2500: 8546.62 - 8917.12
CI at 99% for sample of size 2500: 8496.21 - 8965.87
CI at 90% for sample of size 5000: 8621.99 - 8844.41
CI at 95% for sample of size 5000: 8600.53 - 8866.65
CI at 99% for sample of size 5000: 8560.66 - 8911.75

```

Visualising bootstrapped means through histogram in order to check that it follows normal distribution

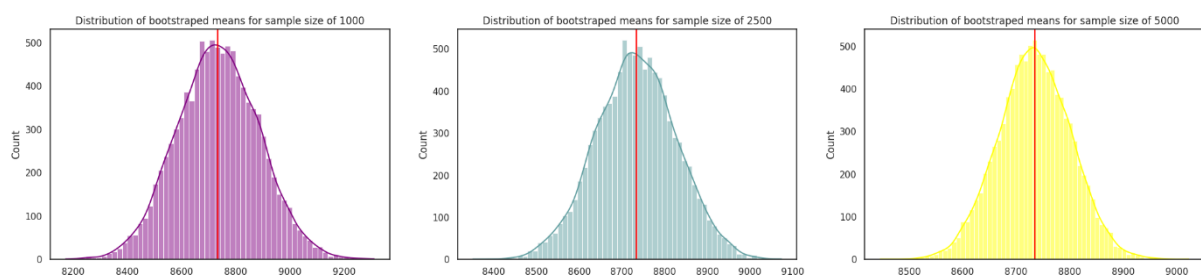
```

plt.figure(figsize=(25,5))
plt.subplot(1,3,1)
sns.histplot(female_purchase_mean_1000,kde=True,color='purple')
plt.axvline(female_df.mean(),color='red',linestyle='-',linewidth=1.5)
plt.title('Distribution of bootstrapped means for sample size of 1000')

plt.subplot(1,3,2)
sns.histplot(female_purchase_mean_2500,kde=True,color='cadetblue')
plt.axvline(female_df.mean(),color='red',linestyle='-',linewidth=1.5)
plt.title('Distribution of bootstrapped means for sample size of 2500')

plt.subplot(1,3,3)
sns.histplot(female_purchase_mean_5000,kde=True,color='yellow')
plt.axvline(female_df.mean(),color='red',linestyle='-',linewidth=1.5)
plt.title('Distribution of bootstrapped means for sample size of 5000')
plt.show()

```

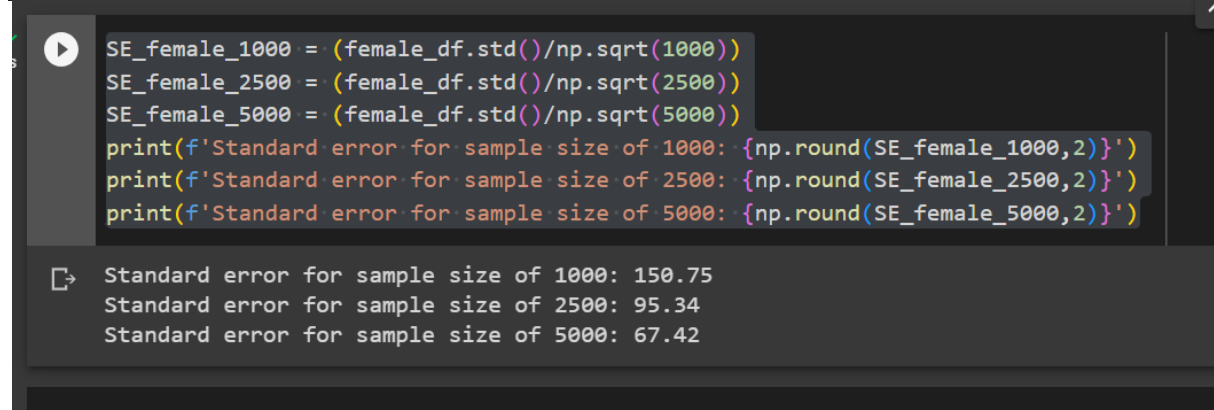


In accordance with the Central Limit Theorem, the distribution of sample means tends to become normal, even when the population distribution does not exhibit normality. This phenomenon is clearly evident in this scenario, as the distribution of bootstrapped means conforms to a normal distribution, despite the population mean not adhering to such a distribution.

As the sample size increases, the distribution of sample means becomes closer to a normal distribution.

Calculating the standard error for sample sizes of 1000, 2500, and 5000.

```
SE_female_1000 = (female_df.std()/np.sqrt(1000))
SE_female_2500 = (female_df.std()/np.sqrt(2500))
SE_female_5000 = (female_df.std()/np.sqrt(5000))
print(f'Standard error for sample size of 1000: {np.round(SE_female_1000,2)}')
print(f'Standard error for sample size of 2500: {np.round(SE_female_2500,2)}')
print(f'Standard error for sample size of 5000: {np.round(SE_female_5000,2)}')
```



```
SE_female_1000 = (female_df.std()/np.sqrt(1000))
SE_female_2500 = (female_df.std()/np.sqrt(2500))
SE_female_5000 = (female_df.std()/np.sqrt(5000))
print(f'Standard error for sample size of 1000: {np.round(SE_female_1000,2)}')
print(f'Standard error for sample size of 2500: {np.round(SE_female_2500,2)}')
print(f'Standard error for sample size of 5000: {np.round(SE_female_5000,2)}')
```

Standard error for sample size of 1000: 150.75
Standard error for sample size of 2500: 95.34
Standard error for sample size of 5000: 67.42

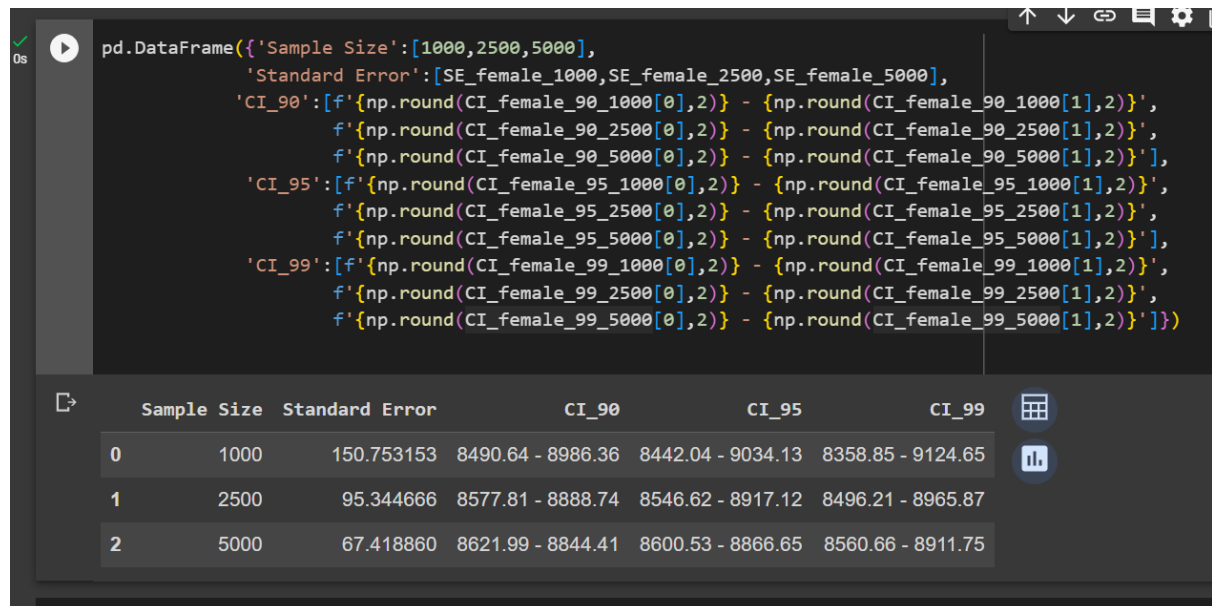
Visualising all analysis using a table

```
pd.DataFrame({'Sample Size':[1000,2500,5000],
              'Standard Error':[SE_female_1000,SE_female_2500,SE_female_5000],
              'CI_90':[f'{np.round(CI_female_90_1000[0],2)} - {np.round(CI_female_90_1000[1],2)}',
                       f'{np.round(CI_female_90_2500[0],2)} - {np.round(CI_female_90_2500[1],2)}',
                       f'{np.round(CI_female_90_5000[0],2)} - {np.round(CI_female_90_5000[1],2)}'],
              'CI_95':[f'{np.round(CI_female_95_1000[0],2)} - {np.round(CI_female_95_1000[1],2)}',
                       f'{np.round(CI_female_95_2500[0],2)} - {np.round(CI_female_95_2500[1],2)}']})
```

```

f'{np.round(CI_female_95_5000[0],2)} -
{np.round(CI_female_95_5000[1],2)}',
'CI_99':f'{np.round(CI_female_99_1000[0],2)} -
{np.round(CI_female_99_1000[1],2)}',
f'{np.round(CI_female_99_2500[0],2)} -
{np.round(CI_female_99_2500[1],2)}',
f'{np.round(CI_female_99_5000[0],2)} -
{np.round(CI_female_99_5000[1],2)}}))

```



```

pd.DataFrame({'Sample Size':[1000,2500,5000],
              'Standard Error':[SE_female_1000,SE_female_2500,SE_female_5000],
              'CI_90': [f'{np.round(CI_female_90_1000[0],2)} - {np.round(CI_female_90_1000[1],2)}',
                        f'{np.round(CI_female_90_2500[0],2)} - {np.round(CI_female_90_2500[1],2)}',
                        f'{np.round(CI_female_90_5000[0],2)} - {np.round(CI_female_90_5000[1],2)}'],
              'CI_95': [f'{np.round(CI_female_95_1000[0],2)} - {np.round(CI_female_95_1000[1],2)}',
                        f'{np.round(CI_female_95_2500[0],2)} - {np.round(CI_female_95_2500[1],2)}',
                        f'{np.round(CI_female_95_5000[0],2)} - {np.round(CI_female_95_5000[1],2)}'],
              'CI_99': [f'{np.round(CI_female_99_1000[0],2)} - {np.round(CI_female_99_1000[1],2)}',
                        f'{np.round(CI_female_99_2500[0],2)} - {np.round(CI_female_99_2500[1],2)}',
                        f'{np.round(CI_female_99_5000[0],2)} - {np.round(CI_female_99_5000[1],2)}']])

```

	Sample Size	Standard Error	CI_90	CI_95	CI_99
0	1000	150.753153	8490.64 - 8986.36	8442.04 - 9034.13	8358.85 - 9124.65
1	2500	95.344666	8577.81 - 8888.74	8546.62 - 8917.12	8496.21 - 8965.87
2	5000	67.418860	8621.99 - 8844.41	8600.53 - 8866.65	8560.66 - 8911.75

1. As the sample size increases, the standard error decreases which essentially translates to the range decreasing between which the population mean lies.
2. The standard error of the mean will approach zero with the increasing number of observations in the sample, as the sample becomes more and more representative of the population, and the sample mean approaches the actual population mean.

Conclusion

Comparing CI at 95% for female and male customers for sample sizes of 1000,2500 and 5000

```

pd.DataFrame({'Sample Size':[1000,2500,5000],
              'CI_95_male': [f'{np.round(CI_male_95_1000[0],2)} -
{np.round(CI_male_95_1000[1],2)}',
f'{np.round(CI_male_95_2500[0],2)} -
{np.round(CI_male_95_2500[1],2)}',

```

```

f'{np.round(CI_male_95_5000[0],2)} -
{np.round(CI_male_95_5000[1],2)}'],
    'CI_95_female': [f'{np.round(CI_female_95_1000[0],2)} -
{np.round(CI_female_95_1000[1],2)}',
    f'{np.round(CI_female_95_2500[0],2)} -
{np.round(CI_female_95_2500[1],2)}',
    f'{np.round(CI_female_95_5000[0],2)} -
{np.round(CI_female_95_5000[1],2)}']]

```

0s

```

pd.DataFrame({'Sample Size': [1000, 2500, 5000],
    'CI_95_male': [f'{np.round(CI_male_95_1000[0],2)} - {np.round(CI_male_95_1000[1],2)}',
    f'{np.round(CI_male_95_2500[0],2)} - {np.round(CI_male_95_2500[1],2)}',
    f'{np.round(CI_male_95_5000[0],2)} - {np.round(CI_male_95_5000[1],2)}'],
    'CI_95_female': [f'{np.round(CI_female_95_1000[0],2)} - {np.round(CI_female_95_1000[1],2)}',
    f'{np.round(CI_female_95_2500[0],2)} - {np.round(CI_female_95_2500[1],2)}',
    f'{np.round(CI_female_95_5000[0],2)} - {np.round(CI_female_95_5000[1],2)}']]

```

	Sample Size	CI_95_male	CI_95_female
0	1000	9124.19 - 9760.04	8442.04 - 9034.13
1	2500	9233.01 - 9632.69	8546.62 - 8917.12
2	5000	9300.31 - 9580.59	8600.53 - 8866.65

With a 95% confidence level, the confidence interval for male customers is consistently both higher and wider than the confidence interval for female customers across all provided sample sizes(1000,2500,5000). This suggests a statistically supported conclusion that male customers spend more money per transaction than female customers.

Comparing CI for male and Female customers at 95% using KDE plot

```

plt.figure(figsize=(15,5))

sns.kdeplot(male_purchase_mean_1000,color='#89AAE6',fill=True,label='Male')
sns.kdeplot(female_purchase_mean_1000,color='#AC80A0',fill=True,label='Female')

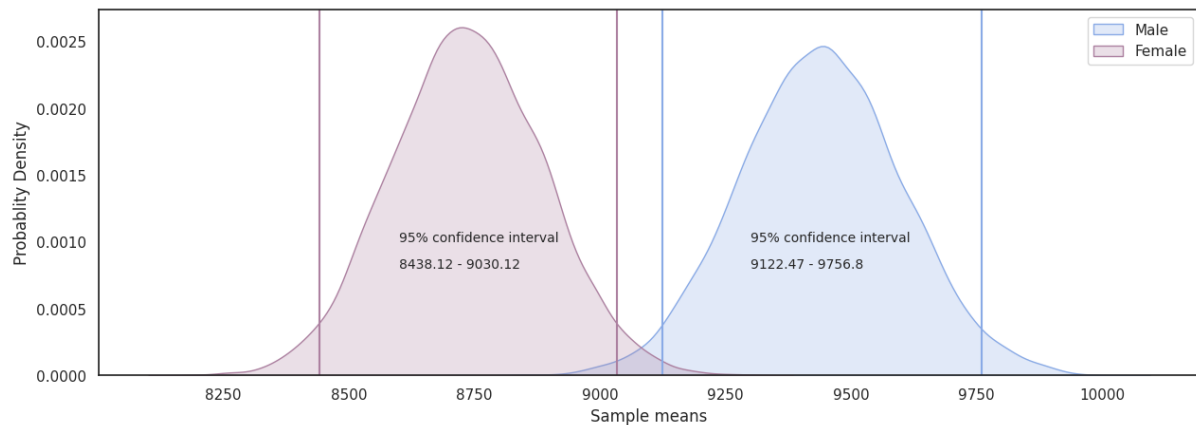
plt.axvline(np.percentile(male_purchase_mean_1000,[2.5]),0,1,color='#89AAE6')
plt.axvline(np.percentile(male_purchase_mean_1000,[97.5]),0,1,color='#89AAE6')
plt.axvline(np.percentile(female_purchase_mean_1000,[2.5]),0,1,color='#AC80A0')
plt.axvline(np.percentile(female_purchase_mean_1000,[97.5]),0,1,color='#AC80A0')

plt.annotate('95% confidence interval',xy=(9300,0.0010),size=10)
plt.annotate('9122.47 - 9756.8',xy=(9300,0.0008),size=10)
plt.annotate('95% confidence interval',xy=(8600,0.0010),size=10)

```



```
plt.annotate('8438.12 - 9030.12',xy=(8600,0.0008),size=10)
plt.xlabel('Sample means')
plt.ylabel('Probability Density')
plt.legend()
plt.show()
```



- 1. Confidence intervals at 95 % for male and female customers does not overlap.**
- 2. With a 95% confidence level, the confidence interval for male customers is consistently both higher and wider than the confidence interval for female customers for a sample size of 1000. This statistically indicates that male customers tend to spend more money per transaction than female customers.**

Answers to the Question

Q1. Are women spending more money per transaction than men? Why or Why not?

- 1. No, CI's of male and female do not overlap and upper limits of female purchase CI are lesser than lower limits of male purchase CI. This proves that men usually spend more than women (NOTE: as per data 75.3% purchases are from male and only 24.7% purchases are from female).**
- 2. The confidence interval for male purchases is consistently both higher and wider than the confidence interval for female purchases. This statistically indicates that male customers tend to spend more money per transaction than female customers.**

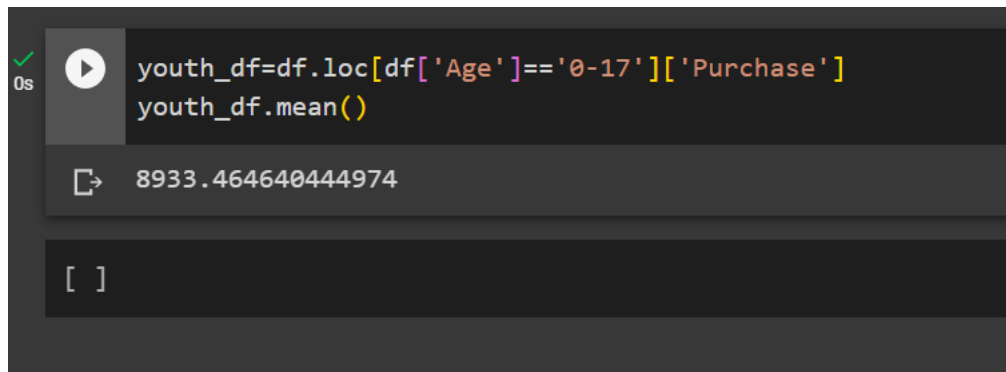
Q2: Confidence intervals and distribution of the mean of the expenses by female and male customers

CI analysis for age-group

CI analysis for 0-17 age-group

Creating a Samples of size 1000 and computing means through bootstrapping

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



The screenshot shows a Jupyter Notebook cell with the following code: `youth_df=df.loc[df['Age']=='0-17']['Purchase']` and `youth_df.mean()`. The cell is executed, and the output is displayed as `8933.464640444974`. Below the output, there is an empty list `[]`.

```
youth_purchase_mean=[]  
for i in range(10000):  
    bootstrapped_sample=np.random.choice(youth_df,size=1000)  
    bootstrapped_mean=np.mean(bootstrapped_sample)  
    youth_purchase_mean.append(bootstrapped_mean)
```

Calculating CI at 90%,95% and 99% for sample of size 1000

```
CI_youth_90=np.percentile(youth_purchase_mean,[5,95])  
CI_youth_95=np.percentile(youth_purchase_mean,[2.5,97.5])  
CI_youth_99=np.percentile(youth_purchase_mean,[0.5,99.5])  
print(f'CI at 90% for sample of size 1000: {np.round(CI_youth_90[0],2)} -  
{np.round(CI_youth_90[1],2)}')  
print(f'CI at 95% for sample of size 1000: {np.round(CI_youth_95[0],2)} -  
{np.round(CI_youth_95[1],2)}')  
print(f'CI at 99% for sample of size 1000: {np.round(CI_youth_99[0],2)} -  
{np.round(CI_youth_99[1],2)}')
```

```

1s youth_purchase_mean=[]
   for i in range(10000):
       bootstrapped_sample=np.random.choice(youth_df,size=1000)
       bootstrapped_mean=np.mean(bootstrapped_sample)
       youth_purchase_mean.append(bootstrapped_mean)

0s CI_youth_90=np.percentile(youth_purchase_mean,[5,95])
   CI_youth_95=np.percentile(youth_purchase_mean,[2.5,97.5])
   CI_youth_99=np.percentile(youth_purchase_mean,[0.5,99.5])
   print(f'CI at 90% for sample of size 1000: {np.round(CI_youth_90[0],2)} - {np.round(CI_youth_90[1],2)}')
   print(f'CI at 95% for sample of size 1000: {np.round(CI_youth_95[0],2)} - {np.round(CI_youth_95[1],2)}')
   print(f'CI at 99% for sample of size 1000: {np.round(CI_youth_99[0],2)} - {np.round(CI_youth_99[1],2)}')

   CI at 90% for sample of size 1000: 8672.78 - 9200.57
   CI at 95% for sample of size 1000: 8622.53 - 9254.29
   CI at 99% for sample of size 1000: 8520.13 - 9343.47

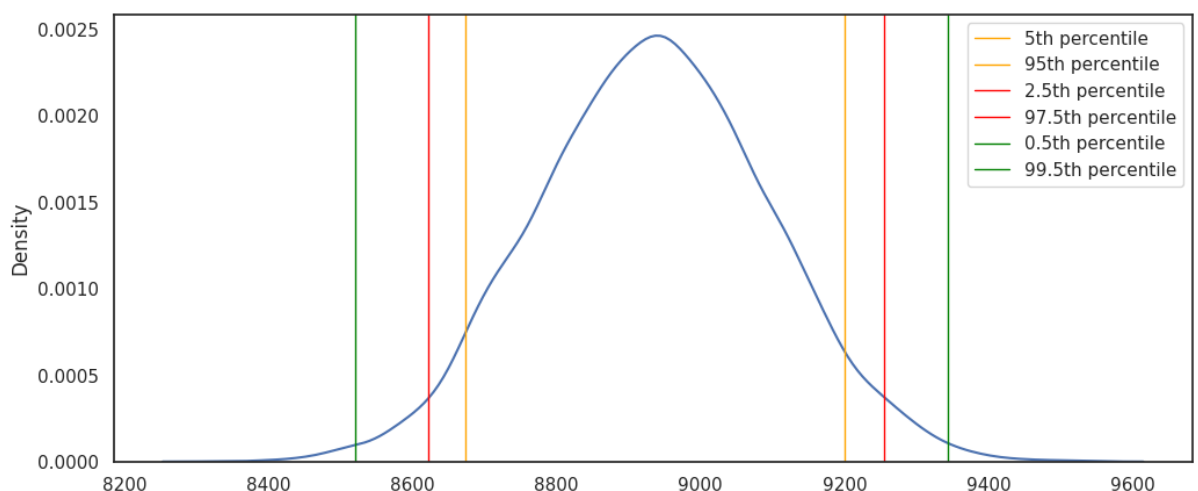
[ ]

```

```

# visualaizing CI for sample size of 1000
plt.figure(figsize=(12,5))
sns.kdeplot(youth_purchase_mean)
plt.axvline(x=np.percentile(youth_purchase_mean,[5]),ymin=0,ymax=1,color='orange',linewidth=1.0,label='5th percentile')
plt.axvline(x=np.percentile(youth_purchase_mean,[95]),ymin=0,ymax=1,color='orange',linewidth=1.0,label='95th percentile')
plt.axvline(x=np.percentile(youth_purchase_mean,[2.5]),ymin=0,ymax=1,color='red',linewidth=1.0,label='2.5th percentile')
plt.axvline(x=np.percentile(youth_purchase_mean,[97.5]),ymin=0,ymax=1,color='red',linewidth=1.0,label='97.5th percentile')
plt.axvline(x=np.percentile(youth_purchase_mean,[0.5]),ymin=0,ymax=1,color='green',linewidth=1.0,label='0.5th percentile')
plt.axvline(x=np.percentile(youth_purchase_mean,[99.5]),ymin=0,ymax=1,color='green',linewidth=1.0,label='99.5th percentile')
plt.legend()
plt.show()

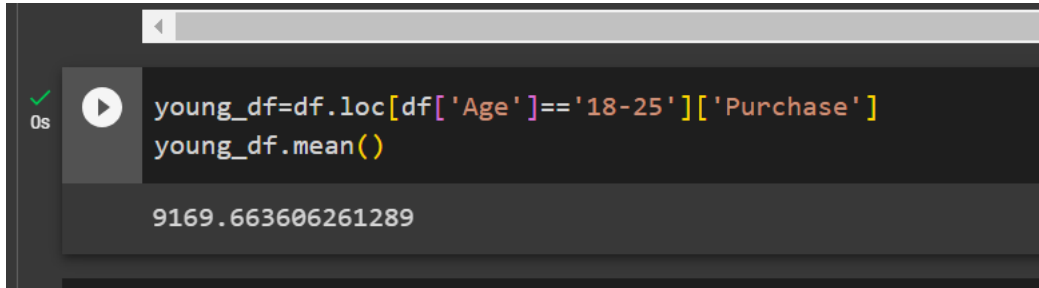
```



CI analysis for 18-25 age-group

Creating a Samples of size 1000 and computing means through bootstrapping

```
young_df=df.loc[df['Age']=='18-25']['Purchase']
young_df.mean()
```



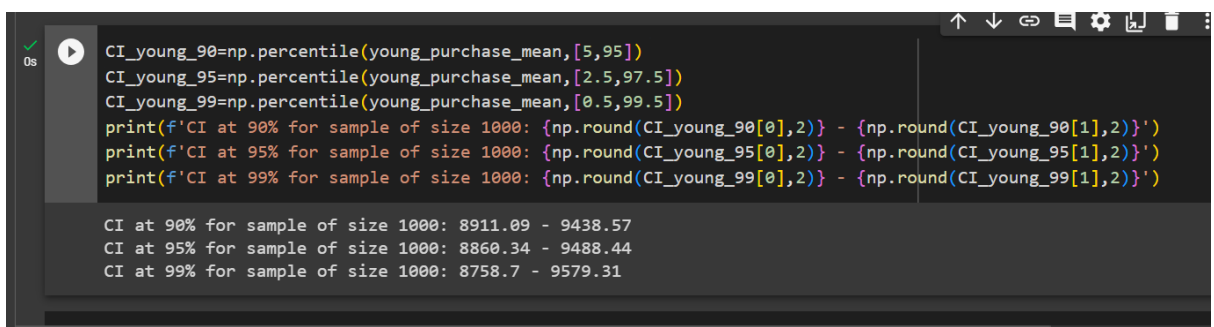
```
young_df=df.loc[df['Age']=='18-25']['Purchase']
young_df.mean()
```

9169.663606261289

```
young_purchase_mean=[]
for i in range(10000):
    bootstrapped_sample=np.random.choice(young_df,size=1000)
    bootstrapped_mean=np.mean(bootstrapped_sample)
    young_purchase_mean.append(bootstrapped_mean)
```

Calculating CI at 90%,95% and 99% for sample of size 1000

```
CI_young_90=np.percentile(young_purchase_mean,[5,95])
CI_young_95=np.percentile(young_purchase_mean,[2.5,97.5])
CI_young_99=np.percentile(young_purchase_mean,[0.5,99.5])
print(f'CI at 90% for sample of size 1000: {np.round(CI_young_90[0],2)} - {np.round(CI_young_90[1],2)}')
print(f'CI at 95% for sample of size 1000: {np.round(CI_young_95[0],2)} - {np.round(CI_young_95[1],2)}')
print(f'CI at 99% for sample of size 1000: {np.round(CI_young_99[0],2)} - {np.round(CI_young_99[1],2)}')
```

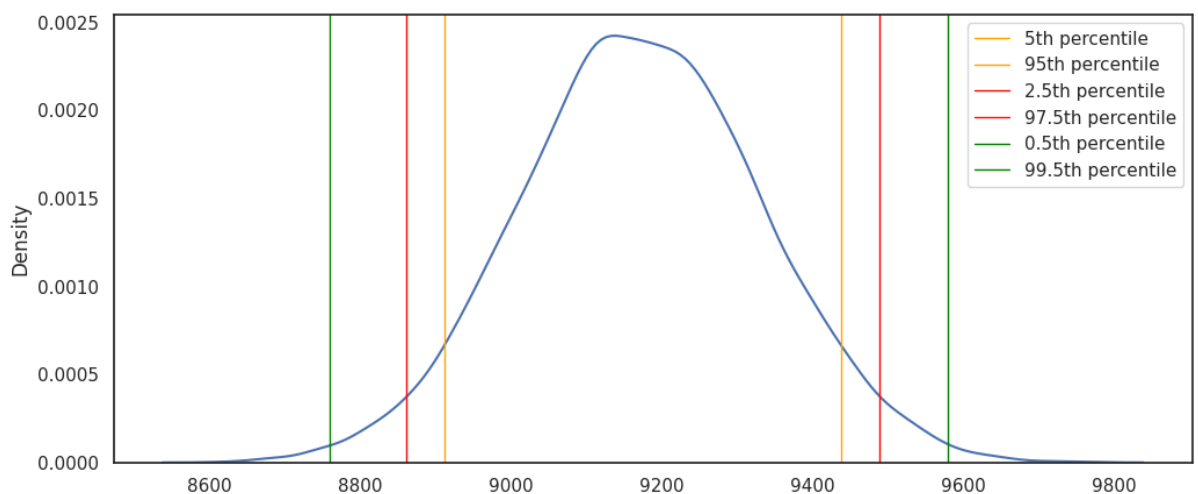


```
CI_young_90=np.percentile(young_purchase_mean,[5,95])
CI_young_95=np.percentile(young_purchase_mean,[2.5,97.5])
CI_young_99=np.percentile(young_purchase_mean,[0.5,99.5])
print(f'CI at 90% for sample of size 1000: {np.round(CI_young_90[0],2)} - {np.round(CI_young_90[1],2)}')
print(f'CI at 95% for sample of size 1000: {np.round(CI_young_95[0],2)} - {np.round(CI_young_95[1],2)}')
print(f'CI at 99% for sample of size 1000: {np.round(CI_young_99[0],2)} - {np.round(CI_young_99[1],2)}')
```

CI at 90% for sample of size 1000: 8911.09 - 9438.57
CI at 95% for sample of size 1000: 8860.34 - 9488.44
CI at 99% for sample of size 1000: 8758.7 - 9579.31

```
plt.figure(figsize=(12,5))
sns.kdeplot(young_purchase_mean)
plt.axvline(x=np.percentile(young_purchase_mean,[5]),ymin=0,ymax=1,color='orange',linewidth=1.0,label='5th percentile')
plt.axvline(x=np.percentile(young_purchase_mean,[95]),ymin=0,ymax=1,color='orange',linewidth=1.0,label='95th percentile')
```

```
plt.axvline(x=np.percentile(young_purchase_mean,[2.5]),ymin=0,ymax=1,color='red',linewidth=1.0,label='2.5th percentile')
plt.axvline(x=np.percentile(young_purchase_mean,[97.5]),ymin=0,ymax=1,color='red',linewidth=1.0,label='97.5th percentile')
plt.axvline(x=np.percentile(young_purchase_mean,[0.5]),ymin=0,ymax=1,color='green',linewidth=1.0,label='0.5th percentile')
plt.axvline(x=np.percentile(young_purchase_mean,[99.5]),ymin=0,ymax=1,color='green',linewidth=1.0,label='99.5th percentile')
plt.legend()
plt.show()
```



CI analysis for 26-35 age-group

Creating a Samples of size 1000 and computing means through bootstrapping

```
adult_df=df.loc[df['Age']=='26-35']['Purchase']
adult_df.mean()
```

```
adult_df=df.loc[df['Age']=='26-35']['Purchase']
adult_df.mean()
```

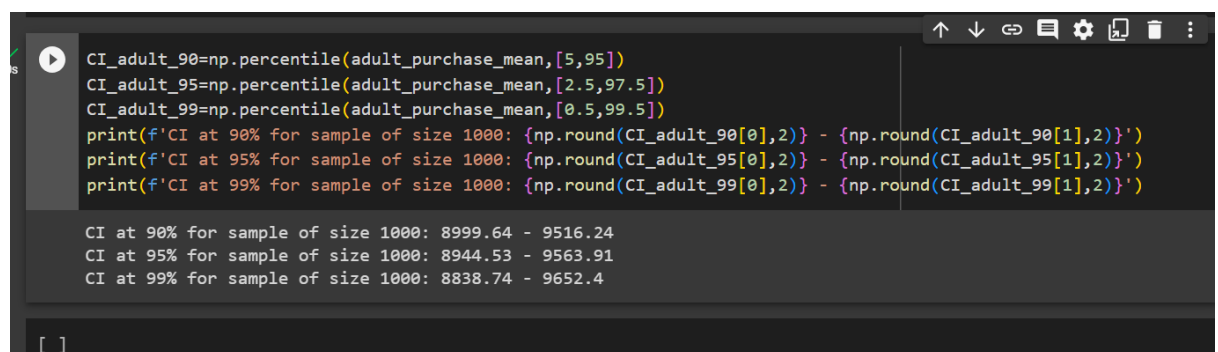
9252.690632869888

[]

```
adult_purchase_mean=[]
for i in range(10000):
    bootstrapped_sample=np.random.choice(adult_df,size=1000)
    bootstrapped_mean=np.mean(bootstrapped_sample)
    adult_purchase_mean.append(bootstrapped_mean)
```

Calculating CI at 90%,95% and 99% for samples of size 1000

```
CI_adult_90=np.percentile(adult_purchase_mean,[5,95])
CI_adult_95=np.percentile(adult_purchase_mean,[2.5,97.5])
CI_adult_99=np.percentile(adult_purchase_mean,[0.5,99.5])
print(f'CI at 90% for sample of size 1000: {np.round(CI_adult_90[0],2)} - {np.round(CI_adult_90[1],2)}')
print(f'CI at 95% for sample of size 1000: {np.round(CI_adult_95[0],2)} - {np.round(CI_adult_95[1],2)}')
print(f'CI at 99% for sample of size 1000: {np.round(CI_adult_99[0],2)} - {np.round(CI_adult_99[1],2)}')
```

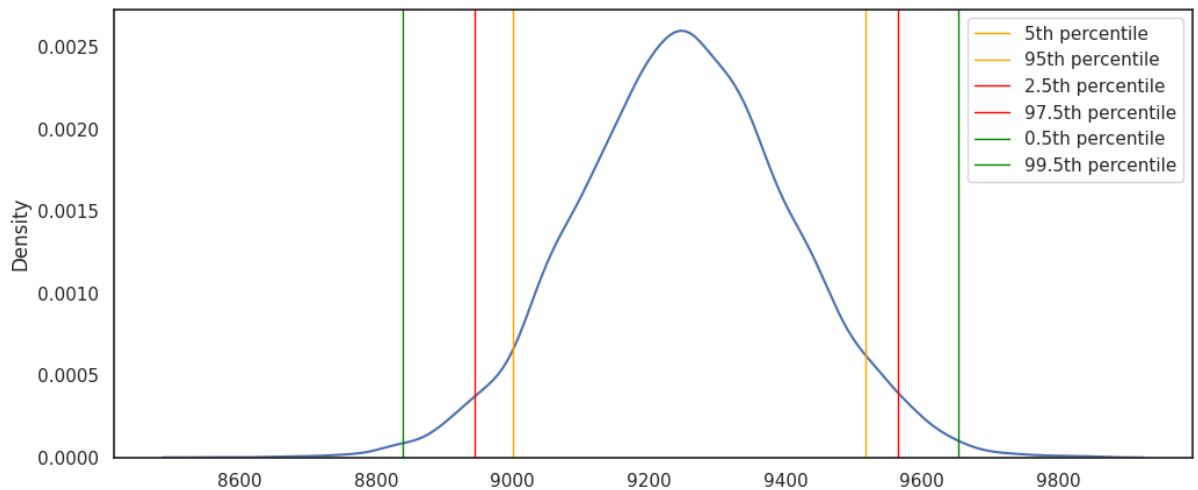


```
CI_adult_90=np.percentile(adult_purchase_mean,[5,95])
CI_adult_95=np.percentile(adult_purchase_mean,[2.5,97.5])
CI_adult_99=np.percentile(adult_purchase_mean,[0.5,99.5])
print(f'CI at 90% for sample of size 1000: {np.round(CI_adult_90[0],2)} - {np.round(CI_adult_90[1],2)}')
print(f'CI at 95% for sample of size 1000: {np.round(CI_adult_95[0],2)} - {np.round(CI_adult_95[1],2)}')
print(f'CI at 99% for sample of size 1000: {np.round(CI_adult_99[0],2)} - {np.round(CI_adult_99[1],2)}')
```

CI at 90% for sample of size 1000: 8999.64 - 9516.24
 CI at 95% for sample of size 1000: 8944.53 - 9563.91
 CI at 99% for sample of size 1000: 8838.74 - 9652.4

```
# visualaizing CI for sample size of 1000
plt.figure(figsize=(12,5))
sns.kdeplot(adult_purchase_mean)
plt.axvline(x=np.percentile(adult_purchase_mean,[5]),ymin=0,ymax=1,color='orange',linewidth=1.0,label='5th percentile')
plt.axvline(x=np.percentile(adult_purchase_mean,[95]),ymin=0,ymax=1,color='orange',linewidth=1.0,label='95th percentile')
plt.axvline(x=np.percentile(adult_purchase_mean,[2.5]),ymin=0,ymax=1,color='red',linewidth=1.0,label='2.5th percentile')
plt.axvline(x=np.percentile(adult_purchase_mean,[97.5]),ymin=0,ymax=1,color='red',linewidth=1.0,label='97.5th percentile')
plt.axvline(x=np.percentile(adult_purchase_mean,[0.5]),ymin=0,ymax=1,color='green',linewidth=1.0,label='0.5th percentile')
plt.axvline(x=np.percentile(adult_purchase_mean,[99.5]),ymin=0,ymax=1,color='green',linewidth=1.0,label='99.5th percentile')
```

```
plt.legend()
plt.show()
```



CI analysis for 36-45 age-group

Creating a Samples of size 1000 and computing means through bootstrapping

```
midage_df=df.loc[df['Age']=='36-45']['Purchase']
midage_df.mean()
```

```
0s midage_df=df.loc[df['Age']=='36-45']['Purchase']
midage_df.mean()

9331.350694917874

[ ]
```

```
midage_purchase_mean=[]
for i in range(10000):
    bootstrapped_sample=np.random.choice(midage_df,size=1000)
    bootstrapped_mean=np.mean(bootstrapped_sample)
    midage_purchase_mean.append(bootstrapped_mean)
```

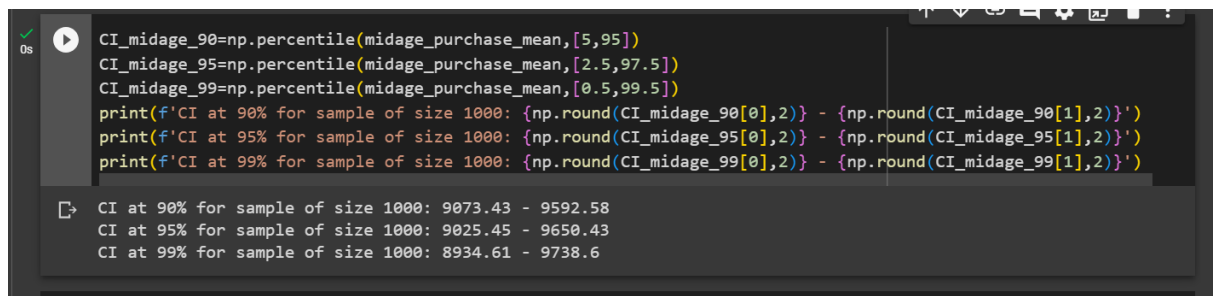
Calculating CI at 90%,95% and 99% for samples of size 1000

```
CI_midage_90=np.percentile(midage_purchase_mean,[5,95])
```

```

CI_midage_95=np.percentile(midage_purchase_mean,[2.5,97.5])
CI_midage_99=np.percentile(midage_purchase_mean,[0.5,99.5])
print(f'CI at 90% for sample of size 1000: {np.round(CI_midage_90[0],2)} - {np.round(CI_midage_90[1],2)}')
print(f'CI at 95% for sample of size 1000: {np.round(CI_midage_95[0],2)} - {np.round(CI_midage_95[1],2)}')
print(f'CI at 99% for sample of size 1000: {np.round(CI_midage_99[0],2)} - {np.round(CI_midage_99[1],2)}')

```



The screenshot shows a Jupyter Notebook interface. The top part is the code editor with the following code:

```

CI_midage_90=np.percentile(midage_purchase_mean,[5,95])
CI_midage_95=np.percentile(midage_purchase_mean,[2.5,97.5])
CI_midage_99=np.percentile(midage_purchase_mean,[0.5,99.5])
print(f'CI at 90% for sample of size 1000: {np.round(CI_midage_90[0],2)} - {np.round(CI_midage_90[1],2)}')
print(f'CI at 95% for sample of size 1000: {np.round(CI_midage_95[0],2)} - {np.round(CI_midage_95[1],2)}')
print(f'CI at 99% for sample of size 1000: {np.round(CI_midage_99[0],2)} - {np.round(CI_midage_99[1],2)}')

```

The bottom part is the output area, which displays the results of the code execution:

```

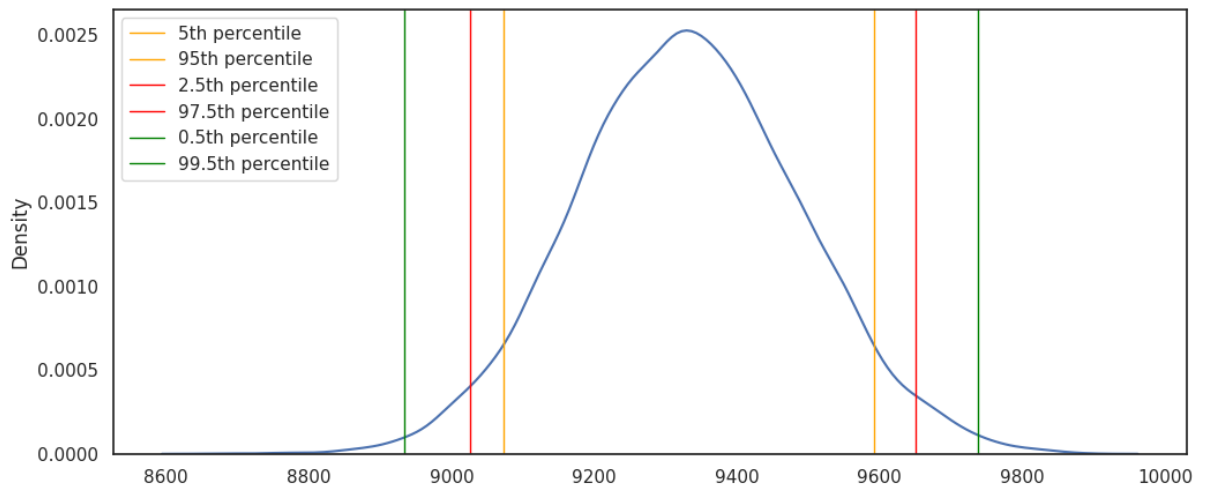
CI at 90% for sample of size 1000: 9073.43 - 9592.58
CI at 95% for sample of size 1000: 9025.45 - 9650.43
CI at 99% for sample of size 1000: 8934.61 - 9738.6

```

```

plt.figure(figsize=(12,5))
sns.kdeplot(midage_purchase_mean)
plt.axvline(x=np.percentile(midage_purchase_mean,[5]),ymin=0,ymax=1,color='orange',linewidth=1.0,label='5th percentile')
plt.axvline(x=np.percentile(midage_purchase_mean,[95]),ymin=0,ymax=1,color='orange',linewidth=1.0,label='95th percentile')
plt.axvline(x=np.percentile(midage_purchase_mean,[2.5]),ymin=0,ymax=1,color='red',linewidth=1.0,label='2.5th percentile')
plt.axvline(x=np.percentile(midage_purchase_mean,[97.5]),ymin=0,ymax=1,color='red',linewidth=1.0,label='97.5th percentile')
plt.axvline(x=np.percentile(midage_purchase_mean,[0.5]),ymin=0,ymax=1,color='green',linewidth=1.0,label='0.5th percentile')
plt.axvline(x=np.percentile(midage_purchase_mean,[99.5]),ymin=0,ymax=1,color='green',linewidth=1.0,label='99.5th percentile')
plt.legend()
plt.show()

```



CI analysis for 46-50 Age-group

Creating a Samples of size 1000 and computing means through bootstrapping

```
midlife_df=df.loc[df['Age']=='46-50']['Purchase']
midlife_df.mean()
```

```
✓ 0s midlife_df=df.loc[df['Age']=='46-50']['Purchase']
midlife_df.mean()

9208.625697468327

[ ]
```

```
midlife_purchase_mean=[]
for i in range(10000):
    bootstrapped_sample=np.random.choice(midlife_df,size=1000)
    bootstrapped_mean=np.mean(bootstrapped_sample)
    midlife_purchase_mean.append(bootstrapped_mean)
```

Calculating CI at 90%,95% and 99% for samples of size 1000

```
CI_midlife_90=np.percentile(midlife_purchase_mean,[5,95])
CI_midlife_95=np.percentile(midlife_purchase_mean,[2.5,97.5])
CI_midlife_99=np.percentile(midlife_purchase_mean,[0.5,99.5])
```



```

print(f'CI at 90% for sample of size 1000: {np.round(CI_midlife_90[0],2)} - {np.round(CI_midlife_90[1],2)}')
print(f'CI at 95% for sample of size 1000: {np.round(CI_midlife_95[0],2)} - {np.round(CI_midlife_95[1],2)}')
print(f'CI at 99% for sample of size 1000: {np.round(CI_midlife_99[0],2)} - {np.round(CI_midlife_99[1],2)}')

```

```

1s [181] midlife_purchase_mean=[]
    for i in range(10000):
        bootstrapped_sample=np.random.choice(midlife_df,size=1000)
        bootstrapped_mean=np.mean(bootstrapped_sample)
        midlife_purchase_mean.append(bootstrapped_mean)

0s CI_midlife_90=np.percentile(midlife_purchase_mean,[5,95])
    CI_midlife_95=np.percentile(midlife_purchase_mean,[2.5,97.5])
    CI_midlife_99=np.percentile(midlife_purchase_mean,[0.5,99.5])
    print(f'CI at 90% for sample of size 1000: {np.round(CI_midlife_90[0],2)} - {np.round(CI_midlife_90[1],2)}')
    print(f'CI at 95% for sample of size 1000: {np.round(CI_midlife_95[0],2)} - {np.round(CI_midlife_95[1],2)}')
    print(f'CI at 99% for sample of size 1000: {np.round(CI_midlife_99[0],2)} - {np.round(CI_midlife_99[1],2)}')

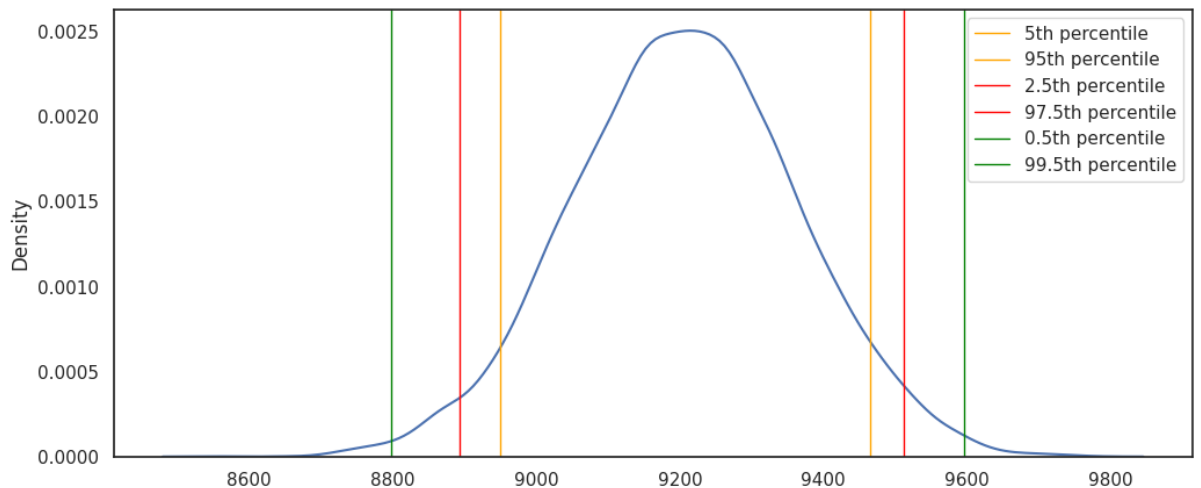
    CI at 90% for sample of size 1000: 8949.38 - 9465.92
    CI at 95% for sample of size 1000: 8893.41 - 9512.31
    CI at 99% for sample of size 1000: 8797.65 - 9596.25

```

```

plt.figure(figsize=(12,5))
sns.kdeplot(midlife_purchase_mean)
plt.axvline(x=np.percentile(midlife_purchase_mean,[5]),ymin=0,ymax=1,color='orange',linewidth=1.0,label='5th percentile')
plt.axvline(x=np.percentile(midlife_purchase_mean,[95]),ymin=0,ymax=1,color='orange',linewidth=1.0,label='95th percentile')
plt.axvline(x=np.percentile(midlife_purchase_mean,[2.5]),ymin=0,ymax=1,color='red',linewidth=1.0,label='2.5th percentile')
plt.axvline(x=np.percentile(midlife_purchase_mean,[97.5]),ymin=0,ymax=1,color='red',linewidth=1.0,label='97.5th percentile')
plt.axvline(x=np.percentile(midlife_purchase_mean,[0.5]),ymin=0,ymax=1,color='green',linewidth=1.0,label='0.5th percentile')
plt.axvline(x=np.percentile(midlife_purchase_mean,[99.5]),ymin=0,ymax=1,color='green',linewidth=1.0,label='99.5th percentile')
plt.legend()
plt.show()

```



CI analysis for 51-55 age-group

Creating a Samples of size 1000 and computing means through bootstraping

```
old_df=df.loc[df['Age']=='51-55']['Purchase']
old_df.mean()
```

```
[184] old_df=df.loc[df['Age']=='51-55']['Purchase']
      old_df.mean()

9534.808030960236
```

```
old_purchase_mean=[]
for i in range(10000):
    bootstraped_sample=np.random.choice(old_df,size=1000)
    bootstraped_mean=np.mean(bootstraped_sample)
    old_purchase_mean.append(bootstraped_mean)
```

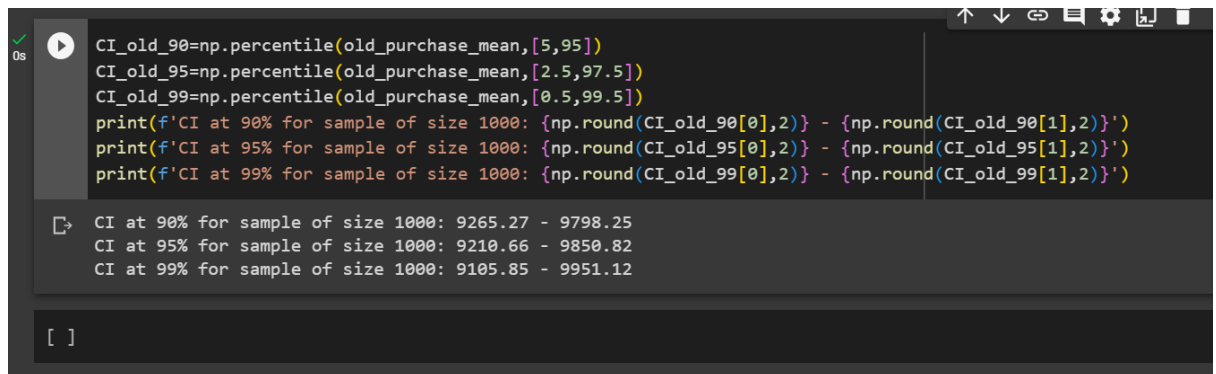
Calculating CI at 90%,95% and 99% for samples of size 1000

```
CI_old_90=np.percentile(old_purchase_mean,[5,95])
CI_old_95=np.percentile(old_purchase_mean,[2.5,97.5])
CI_old_99=np.percentile(old_purchase_mean,[0.5,99.5])
```

```

print(f'CI at 90% for sample of size 1000: {np.round(CI_old_90[0],2)} - {np.round(CI_old_90[1],2)}')
print(f'CI at 95% for sample of size 1000: {np.round(CI_old_95[0],2)} - {np.round(CI_old_95[1],2)}')
print(f'CI at 99% for sample of size 1000: {np.round(CI_old_99[0],2)} - {np.round(CI_old_99[1],2)}')

```



The screenshot shows a Jupyter Notebook interface. The top part is the code editor with the following code:

```

CI_old_90=np.percentile(old_purchase_mean,[5,95])
CI_old_95=np.percentile(old_purchase_mean,[2.5,97.5])
CI_old_99=np.percentile(old_purchase_mean,[0.5,99.5])
print(f'CI at 90% for sample of size 1000: {np.round(CI_old_90[0],2)} - {np.round(CI_old_90[1],2)}')
print(f'CI at 95% for sample of size 1000: {np.round(CI_old_95[0],2)} - {np.round(CI_old_95[1],2)}')
print(f'CI at 99% for sample of size 1000: {np.round(CI_old_99[0],2)} - {np.round(CI_old_99[1],2)}')

```

The bottom part shows the output of the code:

```

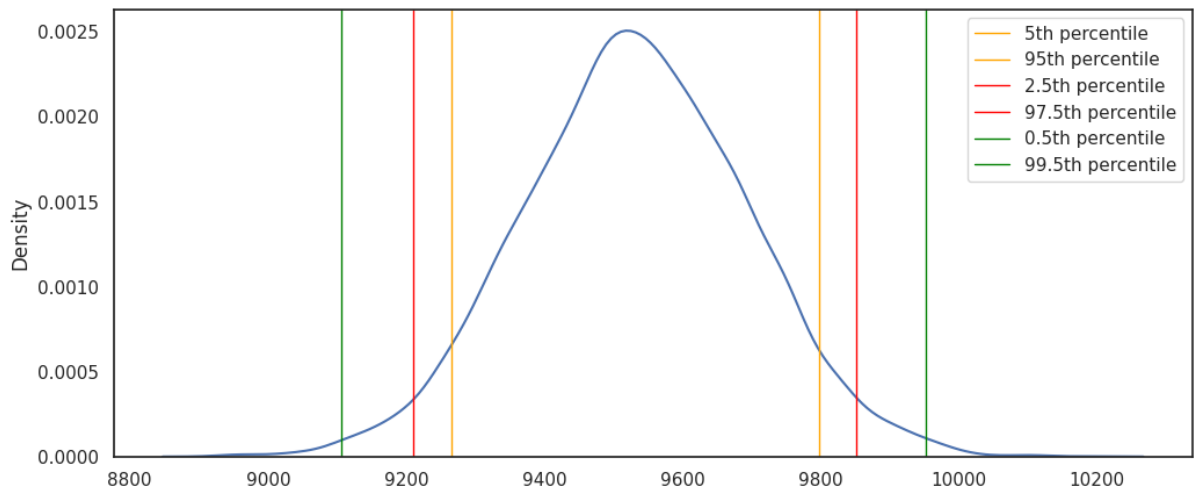
CI at 90% for sample of size 1000: 9265.27 - 9798.25
CI at 95% for sample of size 1000: 9210.66 - 9850.82
CI at 99% for sample of size 1000: 9105.85 - 9951.12

```

```

plt.figure(figsize=(12,5))
sns.kdeplot(old_purchase_mean)
plt.axvline(x=np.percentile(old_purchase_mean,[5]),ymin=0,ymax=1,color='orange',linewidth=1.0,label='5th percentile')
plt.axvline(x=np.percentile(old_purchase_mean,[95]),ymin=0,ymax=1,color='orange',linewidth=1.0,label='95th percentile')
plt.axvline(x=np.percentile(old_purchase_mean,[2.5]),ymin=0,ymax=1,color='red',linewidth=1.0,label='2.5th percentile')
plt.axvline(x=np.percentile(old_purchase_mean,[97.5]),ymin=0,ymax=1,color='red',linewidth=1.0,label='97.5th percentile')
plt.axvline(x=np.percentile(old_purchase_mean,[0.5]),ymin=0,ymax=1,color='green',linewidth=1.0,label='0.5th percentile')
plt.axvline(x=np.percentile(old_purchase_mean,[99.5]),ymin=0,ymax=1,color='green',linewidth=1.0,label='99.5th percentile')
plt.legend()
plt.show()

```



CI analysis for 55+ age-group

Creating a Samples of size 1000 and computing means through bootstrapping

```
senior_df=df.loc[df['Age']=='55+']['Purchase']
senior_df.mean()
```

```
senior_df=df.loc[df['Age']=='55+']['Purchase']
senior_df.mean()

9336.280459449405
```

```
senior_purchase_mean=[]
for i in range(10000):
    bootstrapped_sample=np.random.choice(senior_df,size=1000)
    bootstrapped_mean=np.mean(bootstrapped_sample)
    senior_purchase_mean.append(bootstrapped_mean)
```

Calculating CI at 90%,95% and 99% for samples of size 1000

```
CI_senior_90=np.percentile(senior_purchase_mean,[5,95])
CI_senior_95=np.percentile(senior_purchase_mean,[2.5,97.5])
CI_senior_99=np.percentile(senior_purchase_mean,[0.5,99.5])
print(f'CI at 90% for sample of size 1000: {np.round(CI_senior_90[0],2)} - {np.round(CI_senior_90[1],2)}')
```

```
print(f'CI at 95% for sample of size 1000: {np.round(CI_senior_95[0],2)} - {np.round(CI_senior_95[1],2)}')
print(f'CI at 99% for sample of size 1000: {np.round(CI_senior_99[0],2)} - {np.round(CI_senior_99[1],2)}')
```

```

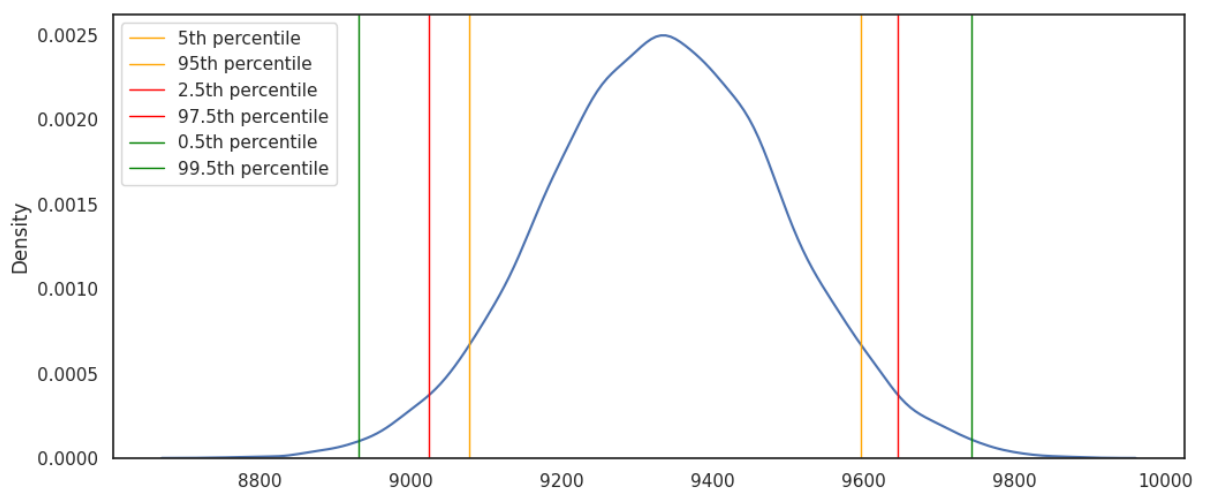
CI_senior_90=np.percentile(senior_purchase_mean,[5,95])
CI_senior_95=np.percentile(senior_purchase_mean,[2.5,97.5])
CI_senior_99=np.percentile(senior_purchase_mean,[0.5,99.5])
print(f'CI at 90% for sample of size 1000: {np.round(CI_senior_90[0],2)} - {np.round(CI_senior_90[1],2)}')
print(f'CI at 95% for sample of size 1000: {np.round(CI_senior_95[0],2)} - {np.round(CI_senior_95[1],2)}')
print(f'CI at 99% for sample of size 1000: {np.round(CI_senior_99[0],2)} - {np.round(CI_senior_99[1],2)}')

CI at 90% for sample of size 1000: 9076.9 - 9597.2
CI at 95% for sample of size 1000: 9024.39 - 9645.73
CI at 99% for sample of size 1000: 8929.9 - 9742.87

[ ]

```

```
plt.figure(figsize=(12,5))
sns.kdeplot(senior_purchase_mean)
plt.axvline(x=np.percentile(senior_purchase_mean,[5]),ymin=0,ymax=1,color='orange',linewidth=1.0,label='5th percentile')
plt.axvline(x=np.percentile(senior_purchase_mean,[95]),ymin=0,ymax=1,color='orange',linewidth=1.0,label='95th percentile')
plt.axvline(x=np.percentile(senior_purchase_mean,[2.5]),ymin=0,ymax=1,color='red',linewidth=1.0,label='2.5th percentile')
plt.axvline(x=np.percentile(senior_purchase_mean,[97.5]),ymin=0,ymax=1,color='red',linewidth=1.0,label='97.5th percentile')
plt.axvline(x=np.percentile(senior_purchase_mean,[0.5]),ymin=0,ymax=1,color='green',linewidth=1.0,label='0.5th percentile')
plt.axvline(x=np.percentile(senior_purchase_mean,[99.5]),ymin=0,ymax=1,color='green',linewidth=1.0,label='99.5th percentile')
plt.legend()
plt.show()
```

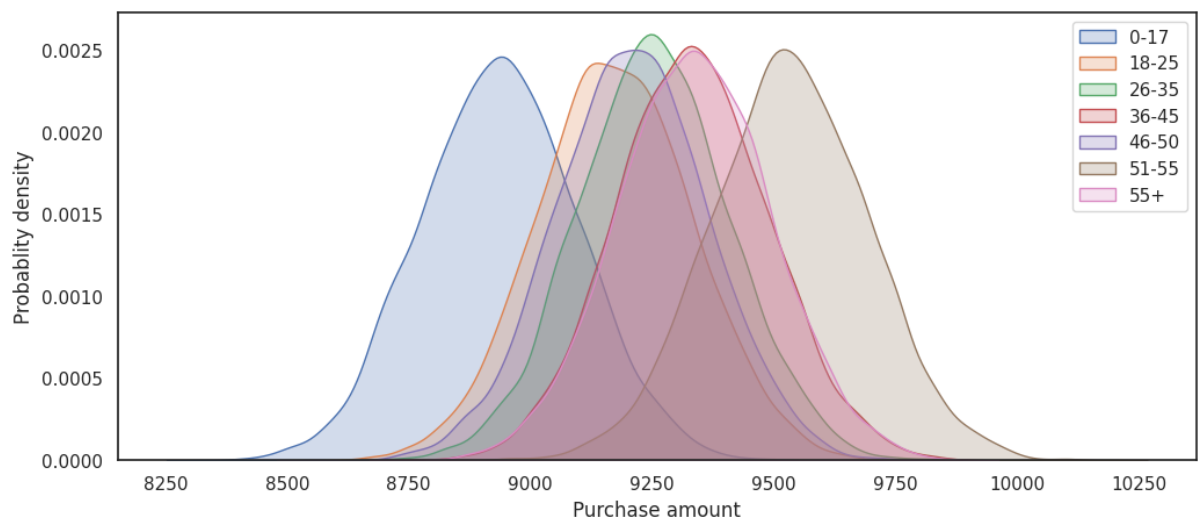


Conclusion

```

sample_means=[youth_purchase_mean,young_purchase_mean,adult_purchase_mean,midage_purchase_mean,midlife_purchase_mean,old_purchase_mean,se
nior_purchase_mean]
labels=['0-17','18-25','26-35','36-45','46-50','51-55','55+']
plt.figure(figsize=(12,5))
for i in range(len(labels)):
    sns.kdeplot(sample_means[i],fill=True,label=labels[i])
plt.xlabel('Purchase amount')
plt.ylabel('Probablity density')
plt.legend()
plt.show()

```



The majority of age groups' purchasing behaviors exhibit overlapping patterns, with the exception of the (0-17) and (51-55) age categories.

Insights:

- 1. At Walmart, 75.3% of transactions are attributed to male customers, while 24.7% of transactions are attributed to female customers.**
- 2. With 75.3% of transactions being initiated by male customers and 24.7% by female customers, the confidence interval for male purchases consistently exhibits both a higher upper bound and a wider spread in comparison to the confidence interval for female purchases. This statistical trend strongly implies that male customers tend to have a higher average transaction spending compared to their female counterparts.**
- 3. Inter Quartile Range for male customers :5863.0-12454.0**
- 4. Median purchases for male customers: 8098.0**
- 5. Inter Quartile Range for female customer:5433.0-11400.0**
- 6. Median purchases for female customers: 7914.0**

- 7. CI for male customers at 95% for sample size of 1000: 9122.77 - 9750.98**
- 8. CI for female customers at 95% for sample size of 1000: 8437.86 - 9029.25**
- 9. Out of the total transactions, 59% are carried out by unmarried customers, while the remaining 41% are attributed to married customers.**
- 10. The confidence intervals for married and unmarried customers overlap, suggesting that both male and female customers spend a similar amount per transaction. This means that the spending behavior of married and unmarried customers is alike.**
- 11. The spending behavior of both married and unmarried customers is consistent, as they both spend nearly the same amount per transaction. This is notably evident from the boxplot analysis, where the interquartile range and median purchases for both groups remain identical.**
- 12. CI for married customers at 95% for sample size of 1000: 8956.45 - 9582.07**
- 13. CI for unmarried customers at 95% for sample size of 1000: 8951.62 - 9572.98**
- 14. Approximately 86.31% of transactions are attributed to the age group 26-50, while those aged above 50 account for 10.91%. The youngest group (0-17 years) constitutes 2.75% of the total transactions.**
- 15. The majority of age groups' purchasing behaviors exhibit overlapping patterns, with the exception of the (0-17) and (51-55) age categories.**
- 16. Purchase distribution of customers exhibits positive skewness, with a pronounced tail extending towards the higher values on the right side.**
- 17. Among Walmart's customers, a notable 27.4% exhibit a preference for purchasing items within product category 5. On the other hand, product category 9 experiences the least popularity, with only a minimal 0.07% of customers expressing a preference for it.**
- 18. The majority of customers fall under Occupation category 4.**
- 19. Walmart customers are distributed across different city categories: 42.03% belong to category B, 31.12% belong to category C, and the remaining 27% belong to category A.**
- 20. The majority of customers reside in a single city for a duration of one year.**
- 21. Within Walmart, product categories 13, 19, and 20 record the lowest spending per transaction among customers.**
- 22. Among Walmart customers, product categories 6, 7, and 15 stand out as the most favored, evidenced by their higher spending per transaction.**

- 23. Customers residing in a 'C' city category demonstrate higher spending per transaction compared to customers in other city categories.**
- 24. Female customers predominantly favor product category 5 for their purchases, while male customers show a preference for product category 1 in their shopping choices.**
- 25. Both unmarried and married customers at Walmart exhibit a shared preference for product category 5 when making their shopping choices.**

Recommendations

Actionable Insight: With 75.3% of transactions being initiated by male customers and 24.7% by female customers, the confidence interval for male purchases consistently exhibits both a higher upper bound and a wider spread in comparison to the confidence interval for female purchases. This statistical trend strongly implies that male customers tend to have a higher average transaction spending compared to their female counterparts.

Recommendations:

- 1. For Black Friday sales, make sure the store provides a wide selection of products that align with the preferences and requirements of female customers. This might encompass a diverse range, spanning from clothing and beauty products to household essentials and beyond.**
- 2. Actively seek feedback from female customers to understand their preferences, concerns, and suggestions. Use this feedback to make improvements and adjustments that cater to their needs.**
- 3. For Black Friday sales, introduce loyalty programs tailored to female customers, featuring exclusive perks like discounts, early access to sales, and unique rewards. This approach can enhance engagement and attract female shoppers.**
- 4. Enhance the online shopping experience for female customers. Ensure that the website is user-friendly, offers detailed product information, and provides convenient options for delivery and returns.**
- 5. Collaborate with brands or influencers that resonate with female customers. This can help expand Walmart's reach and attract new female customers.**

Actionable Insight: Confidence intervals of average married and unmarried spending overlapping.

Recommendations:

- 1. The overlapping confidence intervals of average spending for married and unmarried customers indicate that both male and female customers spend**

a similar amount per transaction. This implies a resemblance in spending behavior between the two groups.

- 2. Instead of allocating resources and time to differentiate between "Married" and "Unmarried" subcategories, Walmart could enhance efficiency by treating both categories as unified. This approach is likely to yield more effective results.**

Actionable Insight: Approximately 86.31% of transactions are attributed to the age group 26-50, while those aged above 50 account for 10.91%. The youngest group (0-17 years) constitutes 2.75% of the total transactions.

Recommendations:

- 1. Designate safe and fun play areas within the store where children can engage in age-appropriate activities. This can provide parents with a convenient shopping experience while keeping kids entertained.**
- 2. Create dedicated zones with comfortable seating and Wi-Fi for teenagers. This can become a place where they can socialize, study, or relax while their parents shop.**
- 3. Stock a wide range of products that appeal to kids and teenagers, including toys, clothes, books, electronics, and school supplies. Ensure that the products are both engaging and age-appropriate.**
- 4. Offer educational toys, craft kits, and learning materials that promote creativity and skill development among children.**
- 5. Introduce special discounts and offers exclusively for the 0-17 age group, encouraging both children and parents to choose Walmart for their shopping needs.**