Problem Statement

- **Analyzing the customer purchase behavior (specifically, purchase amount) against the customer's gender and the various other factors to help the business make better decisions.**
- **Understand the spending behaviour of both male and female to help business make right decisions.**
  - **Do women spend more on Black Friday than men?**

Installing Dependencies

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

Loading Dataset

In [2]:
```
walmart = pd.read_csv("https://raw.githubusercontent.com/abinash25/Walmart_CaseStudy/refs/heads/main/Walmart_data
walmart
```

Out[2]:

	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
...	...	...	...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	B	1	1	20	368
550064	1006035	P00375436	F	26-35	1	C	3	0	20	371
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

550068 rows × 10 columns

Shape of Data

In [3]:
```
walmart.shape
```

Out[3]:
```
(550068, 10)
```

No. of rows (datapoints) - `5,50,068` || No. of columns (features) - `10`

Datatype of features

In [4]:

In [4]:
```
walmart.dtypes
```

Out[4]:
```
User_ID                      int64
Product_ID                  object
Gender                      object
Age                         object
Occupation                   int64
City_Category               object
Stay_In_Current_City_Years  object
Marital_Status               int64
Product_Category             int64
Purchase                     int64
dtype: object
```

No. of categorical feature - `5` || No. of continuous (numerical) feature - `5`

Unique Value (counts) for each features

In [5]:
```
walmart.nunique()
```

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

`Insights`

- There are total **`550068`** datapoints out of which only **`5891`** User_ID found means same users bought multiple product or do multiple transactions.

Missing Values Detection

In [6]:
```
walmart.isnull().sum()
```

Out[6]:
```
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 missing values found in the dataset`

Validating duplicate values

In [7]:
```
walmart.duplicated().sum()
```

Out[7]:
```
0
```

`There is no duplicated values found in the dataset`

In [8]:
```
cat_col = ["User_ID", "Product_ID", "Gender", "Age", "City_Category", "Marital_Status"]
```

```
for i in cat_col:
    walmart[i] = walmart[i].astype("category")

walmart.dtypes
```

Out[8]:
```
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
```

Statistical Summary

In [9]:
```
walmart.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	max
Occupation	550068.0	8.076707	6.522660	0.0	2.0	7.0	14.0	20.0
Product_Category	550068.0	5.404270	3.936211	1.0	1.0	5.0	8.0	20.0
Purchase	550068.0	9263.968713	5023.065394	12.0	5823.0	8047.0	12054.0	23961.0

- **As it is quite obvious that there is significant difference in the mean and std for Purchase column which might be containing outliers.**

In [10]:
```
walmart.describe(include = ["object", "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
City_Category	550068	3	B	231173
Stay_In_Current_City_Years	550068	5	1	193821
Marital_Status	550068	2	0	324731

- **Customer (1001680) has purchased more than others**
- **Product (P00265242) is most bought item**
- **Most of the customers are Male**
- **Most of customers lies in [26-35] Age bracket**
- **Majority of the customers are Unmarried**

Non-Graphical Analysis

Value counts and Unique Attributes

In [11]:
```
walmart.groupby(["Gender"])["User_ID"].nunique() / walmart["User_ID"].nunique() * 100
```

Out[11]:
```
Gender
F    28.280428
```

```
M    71.719572
Name: User_ID, dtype: float64
```

In [12]:
```
walmart.groupby(["Gender"])["Purchase"].describe()
```

Out[12]:

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

- **There are only 28% female customers. While male customers constitute around 72% in our dataset**

In [13]:
```
walmart.groupby(["Age"])["User_ID"].nunique() / walmart["User_ID"].nunique() * 100
```

Out[13]:
```
Age
0-17      3.700560
18-25    18.146325
26-35    34.849771
36-45    19.809879
46-50     9.013750
51-55     8.164997
55+       6.314717
Name: User_ID, dtype: float64
```

In [14]:
```
walmart.groupby(["Age"])["Purchase"].describe()
```

Out[14]:

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

- **Around 35% customers are in [26-35] age bracket. On other hand customers in age bracket [0-17], [51-55] & [55+] showing less purchasing behaviour.**

In [15]:
```
walmart.groupby(["Stay_In_Current_City_Years"])["User_ID"].nunique()
```

Out[15]:
```
Stay_In_Current_City_Years
0      772
1     2086
2     1145
3      979
4+     909
Name: User_ID, dtype: int64
```

- **Customers staying for 1-2 years in same city purchased more than others**

In [16]:
```
walmart.groupby(["Marital_Status"])["User_ID"].nunique() / walmart["User_ID"].nunique() * 100
```

Out[16]:
```
Marital_Status
0    58.003735
1    41.996265
Name: User_ID, dtype: float64
```

- **58:42 is the ratio between married and unmarried customers**

In [17]:
```
walmart.groupby(["City_Category"])["User_ID"].nunique()
```

Out[17]:
```
City_Category
A    1045
B    1707
C    3139
Name: User_ID, dtype: int64
```

- **Most of customers belogs to C city_category**

In [18]:
```
walmart.groupby(['City_Category'])['Purchase'].describe()
```

Out[18]:

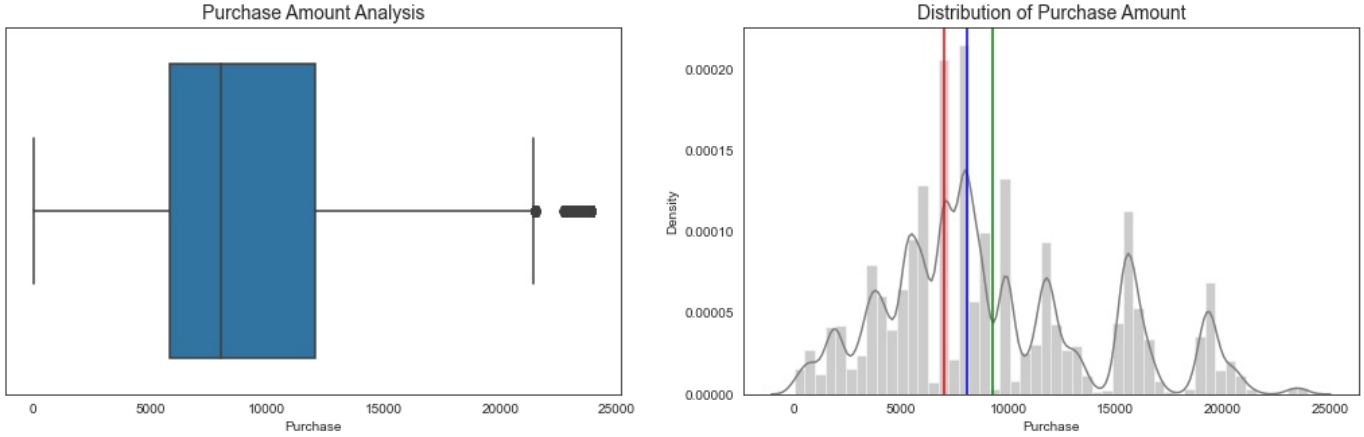
City_Category	count	mean	std	min	25%	50%	75%	max
A	147720.0	8911.939216	4892.115238	12.0	5403.0	7931.0	11786.0	23961.0
B	231173.0	9151.300563	4955.496566	12.0	5460.0	8005.0	11986.0	23960.0
C	171175.0	9719.920993	5189.465121	12.0	6031.5	8585.0	13197.0	23961.0

- **Most of the customers belongs to City C But customers from city_category B tend to purchase more suggest the fact they visit store multiple times during that period**

Graphical Analysis

Univariate Countinous Variable Analysis

In [19]:
```
fig = plt.figure(figsize=(18,5))
sns.set_style("white")

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

plt.title('Purchase Amount Analysis', fontsize = '14')


plt.subplot(1, 2, 2)
sns.distplot(a = walmart["Purchase"], color = 'gray')
plt.title("Distribution of Purchase Amount", fontsize = '14')
plt.axvline(walmart["Purchase"].mean(),color="g")
plt.axvline(walmart["Purchase"].median(),color="b")
plt.axvline(walmart["Purchase"].mode()[0],color="r")

plt.show()
```



Insights

- There are outliers in purchase amount.
- While observing the distribution of purchase amount from density plot. It is quite obvious that the distribution is right skewed means majority of data concentrated on left side.
- Majority of customer purchase within 5,000 - 20,000 range.

Detecting Outliers

In [20]:
```
print('Mean of Purchase Amount   = ', walmart["Purchase"].mean())
print('Median of Purchase Amount = ', walmart["Purchase"].median())
```

```
Mean of Purchase Amount   =  9263.968712959126
Median of Purchase Amount =  8047.0
```

- **`There is significant difference in Mean & Median of Purchase Amounts lead to outliers`**

Handling Outliers

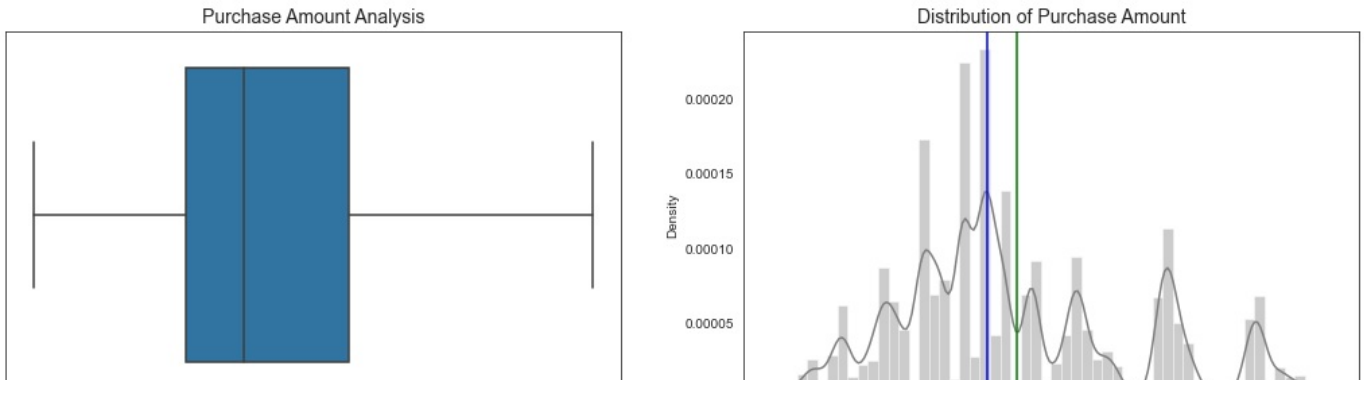
In [21]:
```
q1 = walmart["Purchase"].quantile(0.25)
q3 = walmart["Purchase"].quantile(0.75)

IQR = q3 - q1
print("1. First Quartile (q1) = ", q1)
print("2. Third Quartile (q2) = ", q3)
print("3. IQR                 = ", IQR)
```

```
1. First Quartile (q1) =  5823.0
2. Third Quartile (q2) =  12054.0
3. IQR                 =  6231.0
```

In [22]:
```
walmart_new = walmart.copy()

# walmart_new = walmart_new.loc[(walmart_new["Purchase"] > q1 - 1.5*IQR) & (walmart_new["Purchase"] < q3 + 1.5*IQ
walmart_new["Purchase"] = np.clip(walmart_new["Purchase"], q1-1.5*IQR, q3 + 1.5*IQR)
walmart_new.shape
```

Out[22]:
```
(550068, 10)
```

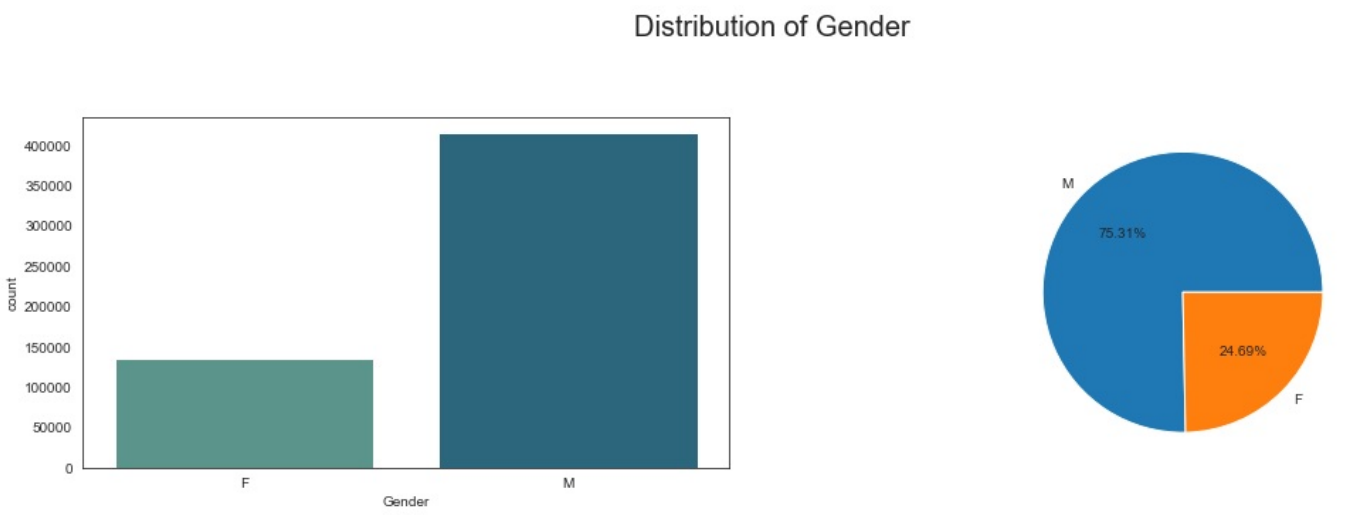
In [23]:
```
fig = plt.figure(figsize=(18,5))
sns.set_style("white")

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

plt.title('Purchase Amount Analysis', fontsize = '14')


plt.subplot(1, 2, 2)
sns.distplot(a = walmart_new["Purchase"], color = 'gray')
plt.title("Distribution of Purchase Amount", fontsize = '14')
plt.axvline(walmart_new["Purchase"].mean(),color="g")
plt.axvline(walmart_new["Purchase"].median(),color="b")

plt.show()
```



- **`Outliers has been clipped now`**

Univariate Categorical Variable Analysis

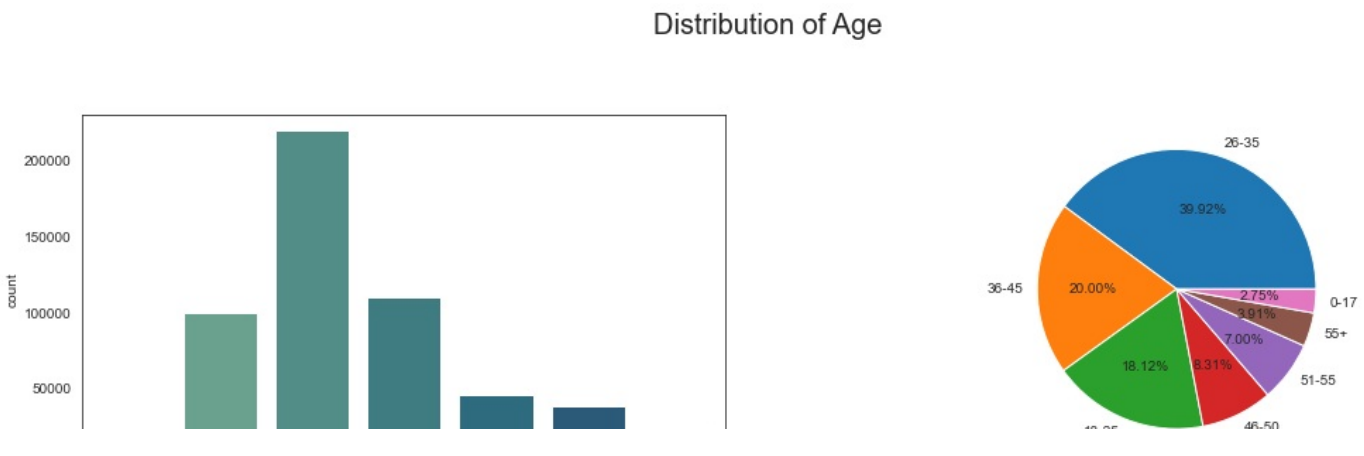
In [24]:
```
fig = plt.figure(figsize=(18,10))
sns.set_style(style='white')
ax1 = plt.subplot2grid((2,2),(0,0))
sns.countplot(data=walmart_new, x="Gender", palette="crest")

#first row sec column
ax1 = plt.subplot2grid((2,2), (0, 1))
plt.pie(walmart_new["Gender"].value_counts(),
        labels = walmart_new["Gender"].value_counts().index,
        autopct = '%1.2f%%')
plt.suptitle('Distribution of Gender', fontsize = 20)

plt.show()
```

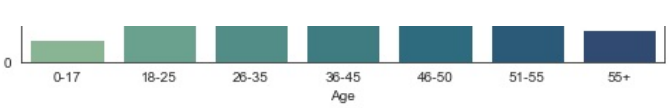


`Insights`

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

In [25]:
```
fig = plt.figure(figsize=(18,10))
sns.set_style(style='white')
ax1 = plt.subplot2grid((2,2),(0,0))
sns.countplot(data=walmart_new, x="Age", palette="crest")

#first row sec column
ax1 = plt.subplot2grid((2,2), (0, 1))
plt.pie(walmart_new["Age"].value_counts(),
        labels = walmart_new["Age"].value_counts().index,
        autopct = '%1.2f%%')
plt.suptitle('Distribution of Age', fontsize = 20)

plt.show()
```



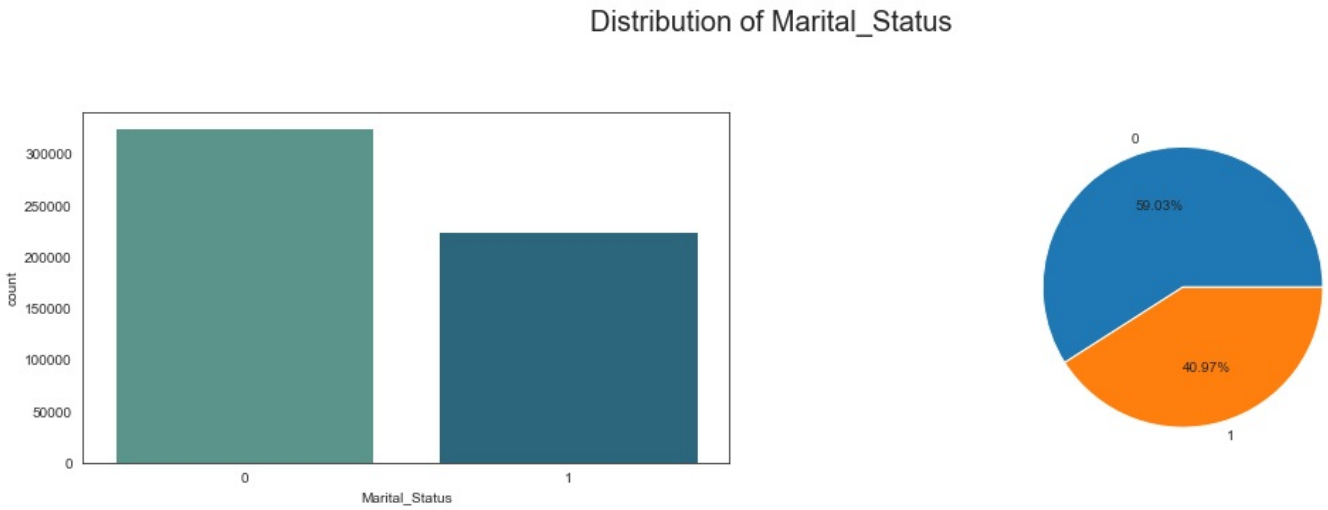


Insights

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

In [26]:

```
fig = plt.figure(figsize=(18,10))
sns.set_style(style='white')
ax1 = plt.subplot2grid((2,2),(0,0))
sns.countplot(data=walmart, x="Marital_Status", palette="crest")

#first row sec column
ax1 = plt.subplot2grid((2,2), (0, 1))
plt.pie(walmart_new["Marital_Status"].value_counts(),
        labels = walmart_new["Marital_Status"].value_counts().index,
        autopct = '%1.2f%%')
plt.suptitle('Distribution of Marital_Status', fontsize = 20)

plt.show()
```



Insights

- Most of customers are unmarried.

In [27]:

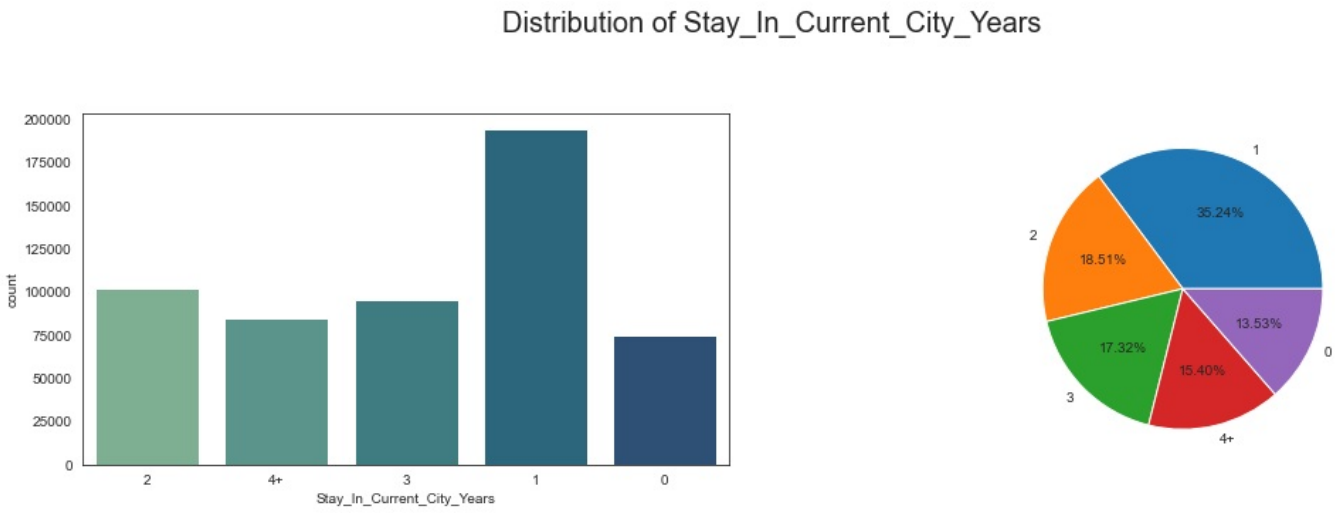
```
fig = plt.figure(figsize=(18,10))
sns.set_style(style='white')
ax1 = plt.subplot2grid((2,2),(0,0))
sns.countplot(data=walmart_new, x="City_Category", palette="crest")

#first row sec column
ax1 = plt.subplot2grid((2,2), (0, 1))
plt.pie(walmart_new["City_Category"].value_counts(),
        labels = walmart_new["City_Category"].value_counts().index,
        autopct = '%1.2f%%')
plt.suptitle('Distribution of City_Category', fontsize = 20)

plt.show()
```

City_Category

Insights

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

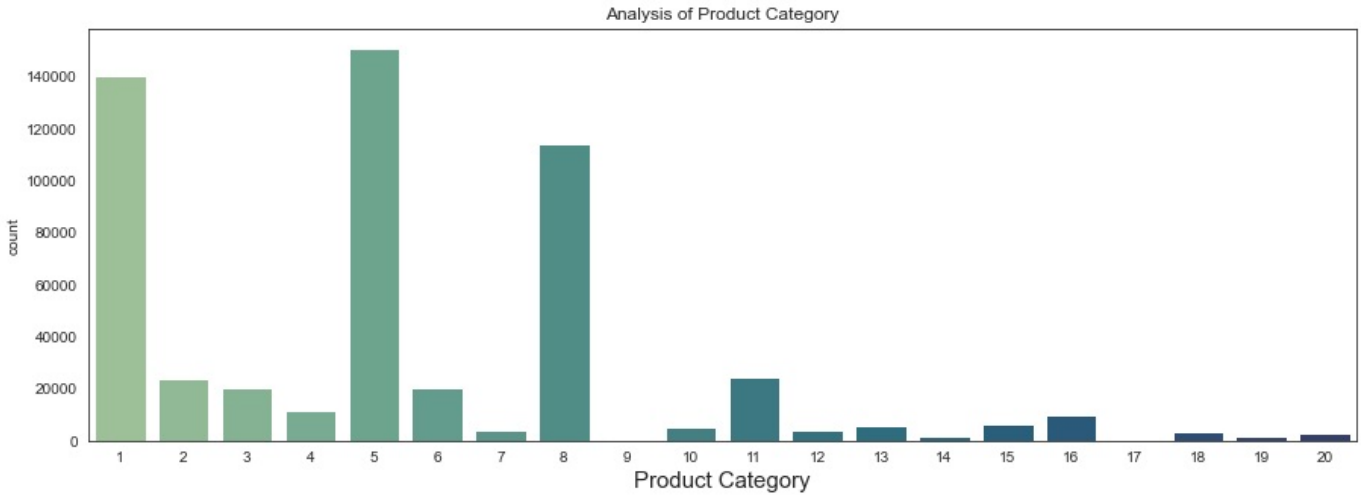
In [28]:
```
fig = plt.figure(figsize=(18,10))
sns.set_style(style='white')
ax1 = plt.subplot2grid((2,2),(0,0))
sns.countplot(data=walmart_new, x="Stay_In_Current_City_Years", palette="crest")

#first row sec column
ax1 = plt.subplot2grid((2,2), (0, 1))
plt.pie(walmart_new["Stay_In_Current_City_Years"].value_counts(),
        labels = walmart_new["Stay_In_Current_City_Years"].value_counts().index,
        autopct = '%1.2f%%')
plt.suptitle('Distribution of Stay_In_Current_City_Years', fontsize = 20)
plt.show()
```



Insights

- Most of customer stayed around 1 years in the current city

In [29]:
```
fig = plt.figure(figsize = (15,5))

sns.countplot(data = walmart_new, x = "Product_Category", palette="crest")
plt.xlabel("Product Category", fontsize = 15)
plt.title("Analysis of Product Category")
plt.show()
```

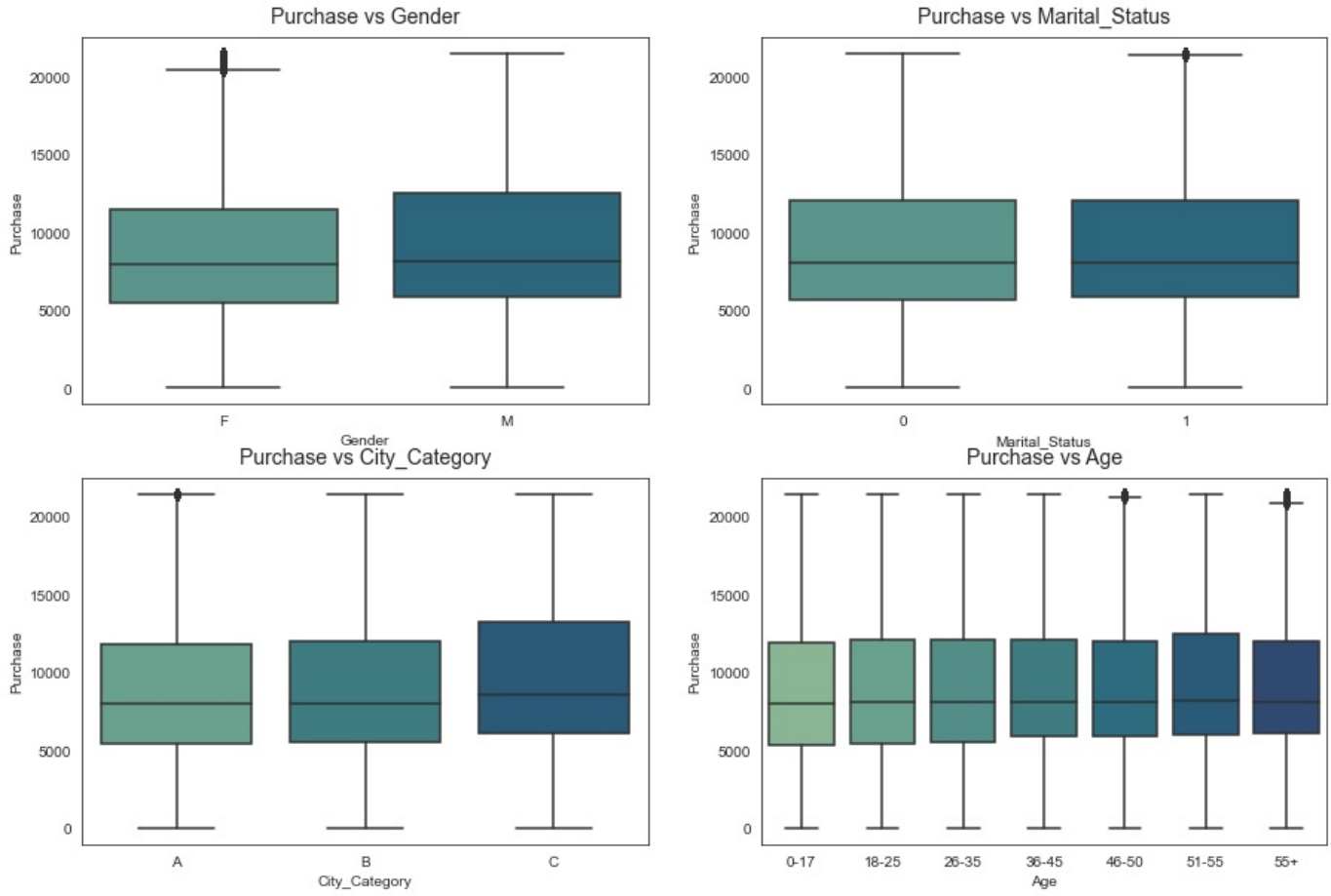


Insights

- Product categories 1, 5 and 8 most bought products.

Bivariate Analysis

In [30]:
```
cat_col = ["Gender", "Marital_Status", "City_Category", "Age"]

fig, axs = plt.subplots(nrows=2, ncols = 2, figsize=(15,10))
k = 0
sns.set_style("dark")
for i in range(2):
    for j in range(2):
        sns.boxplot(data=walmart_new, x=cat_col[k], y="Purchase", palette='crest', ax=axs[i, j])
        axs[i, j].set_title("Purchase vs " + cat_col[k], pad = 10, fontsize = 14)
        k += 1
plt.show()
```

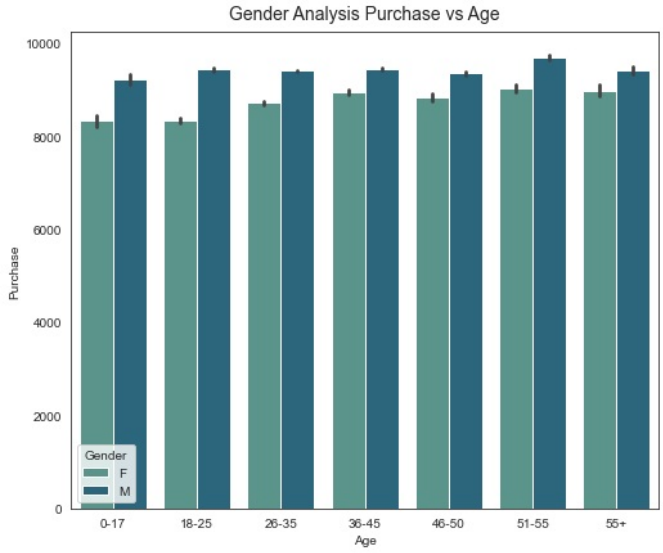


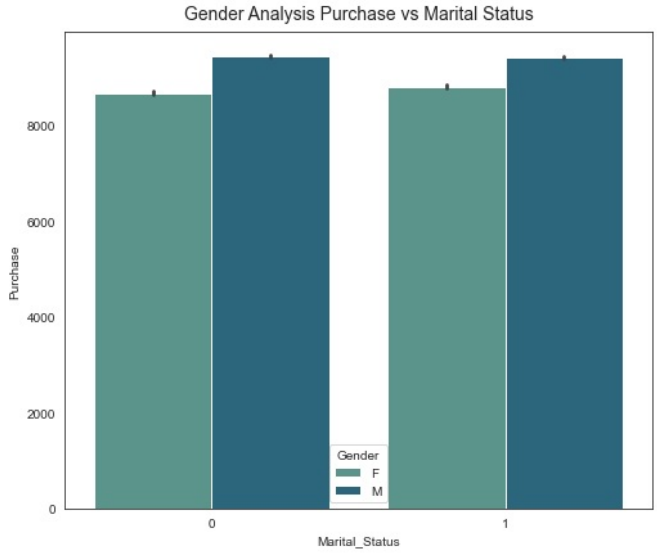
Insights

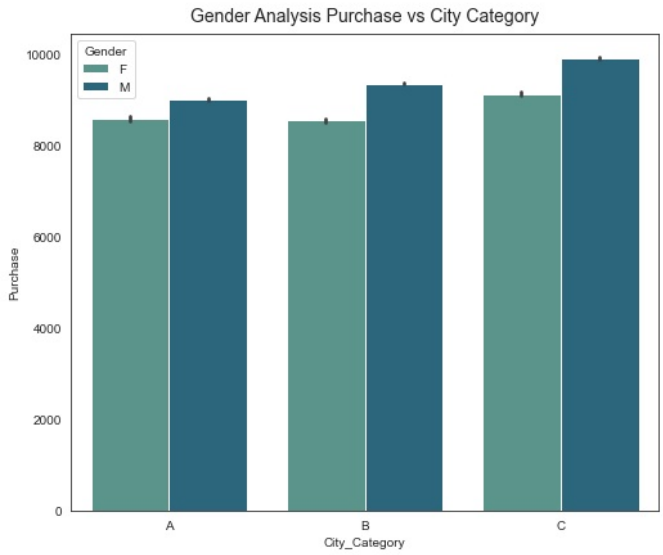
- Customers puchasing behaviour are almost stable at equilibrium irrespective on `Gender` , `Marital_status` , `City_Category` & `Age`
- Median purchase from each features are almost same

Lets draw analysis for Gender

In [31]:
```
fig = plt.figure(figsize=(18, 15))

sns.set_style("white")

ax1 = plt.subplot2grid((2, 2), (0, 0))
sns.barplot(data=walmart_new, x = "Age", y = "Purchase", ax=ax1, hue = "Gender", palette="crest")
plt.title("Gender Analysis Purchase vs Age", pad = 10, fontsize = 14)

ax1 = plt.subplot2grid((2, 2), (0, 1))
sns.barplot(data=walmart_new, x = "Marital_Status", y = "Purchase", ax=ax1, hue = "Gender", palette="crest")
plt.title("Gender Analysis Purchase vs Marital Status", pad = 10, fontsize = 14)

ax1 = plt.subplot2grid((2, 2), (1, 0))
sns.barplot(data=walmart_new, x = "City_Category", y = "Purchase", ax=ax1, hue = "Gender", palette="crest")
plt.title("Gender Analysis Purchase vs City Category", pad = 10, fontsize = 14)

plt.show()
```





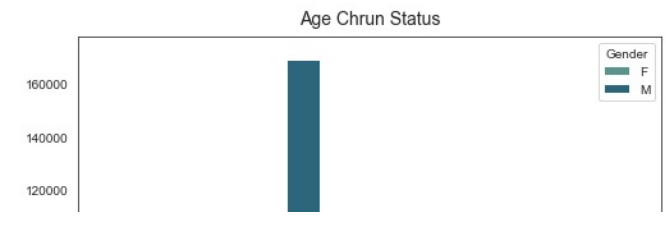


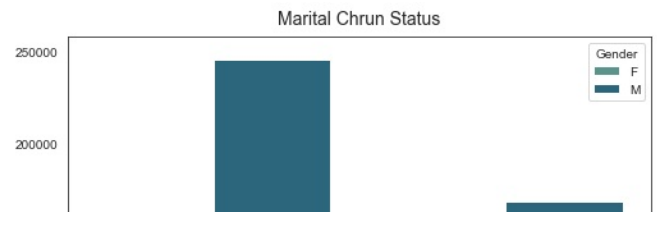
Insights

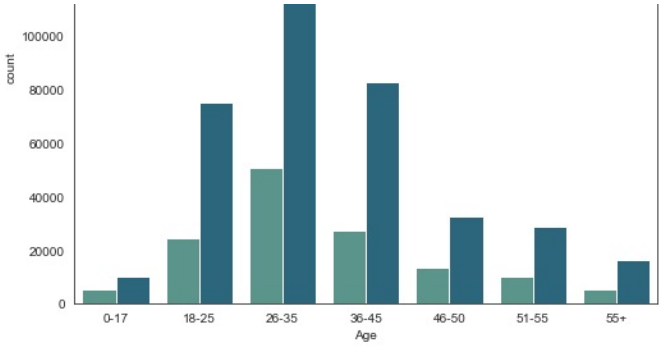
- Purchases are high in city category C
- Purchase is the somehow same for all age groups
- Female Customers form City category C is more than other city category
- Purchases are same for all Marital Status

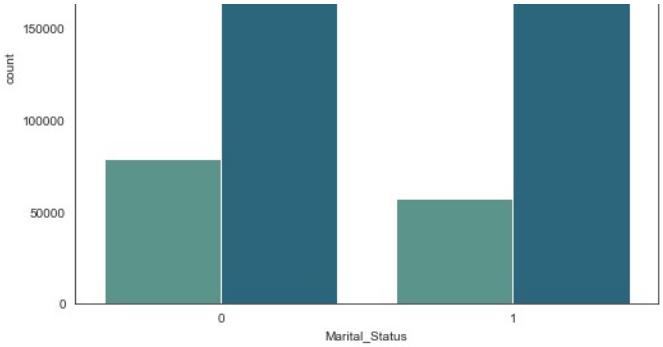
In [32]:

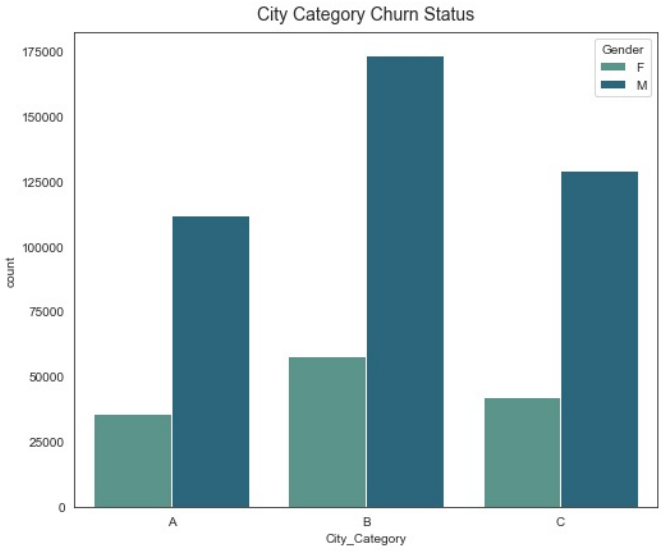
```
fig = plt.figure(figsize=(18, 15))

sns.set_style("white")

ax1 = plt.subplot2grid((2, 2), (0, 0))
sns.countplot(data=walmart_new, x = "Age", ax=ax1, hue = "Gender", palette="crest")
plt.title("Age Chrun Status", pad = 10, fontsize = 14)

ax1 = plt.subplot2grid((2, 2), (0, 1))
sns.countplot(data=walmart_new, x = "Marital_Status", ax=ax1, hue = "Gender", palette="crest")
plt.title("Marital Chrun Status", pad = 10, fontsize = 14)

ax1 = plt.subplot2grid((2, 2), (1, 0))
sns.countplot(data=walmart_new, x = "City_Category", ax=ax1, hue = "Gender", palette="crest")
plt.title("City Category Churn Status", pad = 10, fontsize = 14)

ax1 = plt.subplot2grid((2, 2), (1, 1))
sns.countplot(data=walmart_new, x = "Stay_In_Current_City_Years", ax=ax1, hue = "Gender", palette="crest")
plt.title("Stay_In_Current_City_Years Churn Status", pad = 10, fontsize = 14)

plt.show()
```

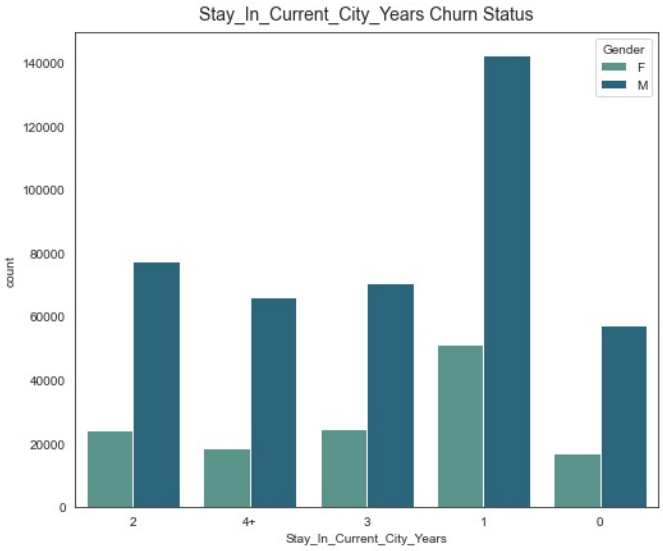










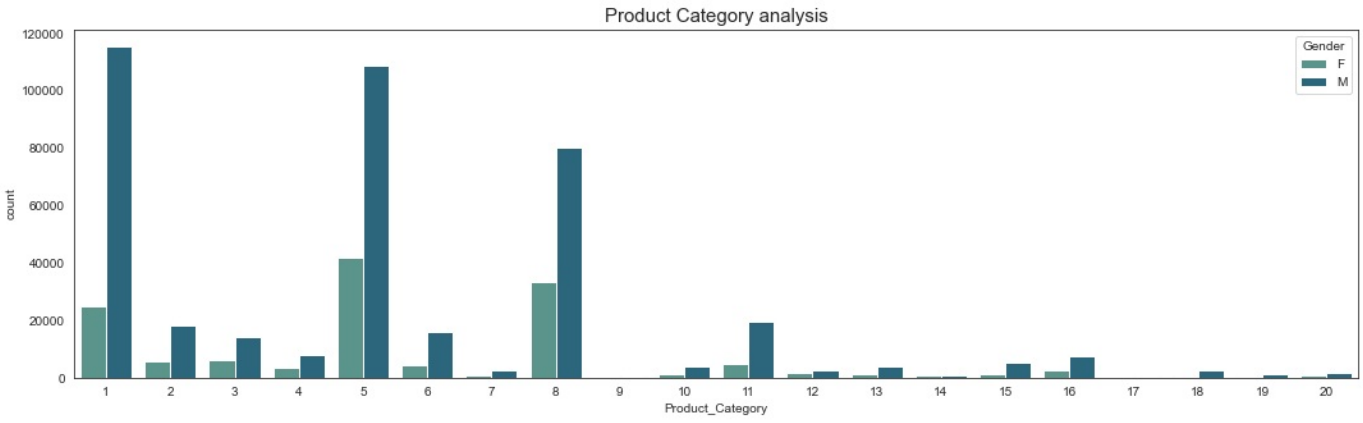


Insights

- Most of female customers are from city category C
- Most of female customer stay 1 year in the current city than others
- Most of the customers are in range 18-45 age bracket
- There is huge difference in the male and female customer for age bracket [26-35]
- There is significant difference of male customers. Most of the male customers are single. While female customers marital status not more significant as there is not so much difference.

In [33]:

```
fig = plt.figure(figsize=(18, 5))

plt.title("Product Category analysis", fontsize = 15)
sns.countplot(data=walmart_new, x = "Product_Category", hue = "Gender", palette="crest")

plt.show()
```

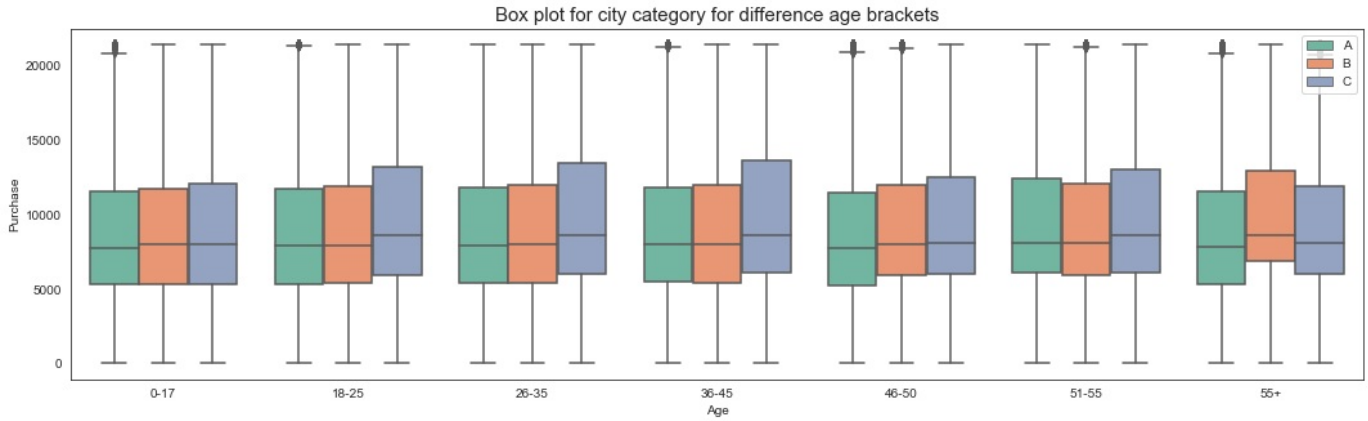


Insights

- Product Category 5 and 8 are demanding among female customers
- Proudct Category 1 and 5 are most demanding among male customers

Lets draw analysis for City Category Among Different Age Brackets

In [34]:
```
fig = plt.figure(figsize=(18, 5))

plt.title("Box plot for city category for difference age brackets", fontsize = 15)
sns.boxplot(data=walmart_new, x = "Age", y = "Purchase", hue = "City_Category", palette="Set2")

plt.legend(loc = "upper right")
plt.show()
```



Insights

- Product Category B is popular among age group 55+
- Product Category A & B are quite same in demand among other age groups except 55+
- While Product Category C is in demand for age group 18-45 and slightly in 51-55 age group

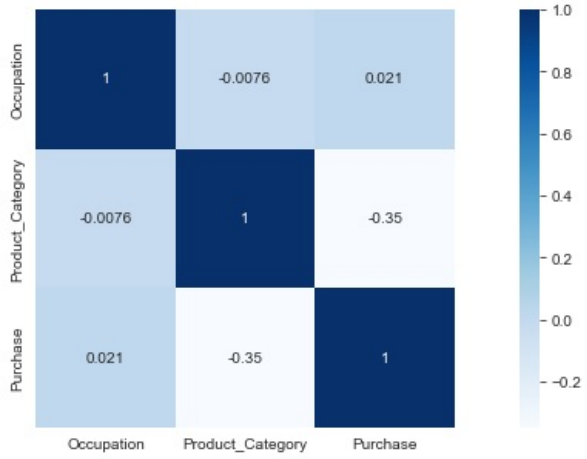
Correlation Analysis

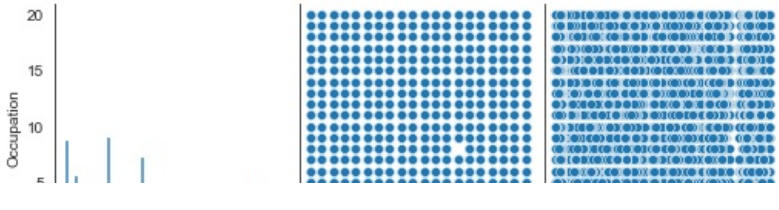
In [35]:
```
walmart_new.corr()
```

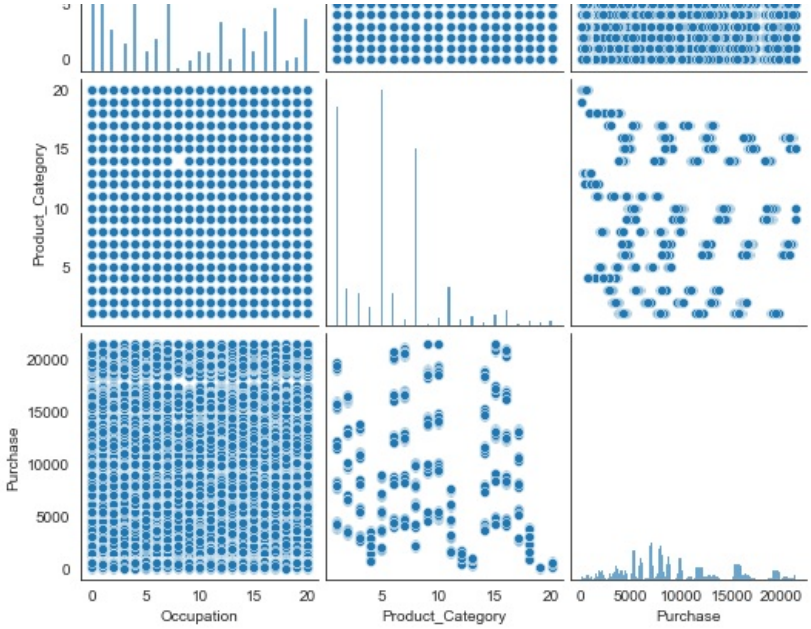
Out[35]:

	Occupation	Product_Category	Purchase
Occupation	1.000000	-0.007618	0.020853
Product_Category	-0.007618	1.000000	-0.347413
Purchase	0.020853	-0.347413	1.000000

In [36]:
```
plt.figure(figsize = (15, 5))
sns.heatmap(data=walmart_new.corr(), annot=True, cmap="Blues", square=True)

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







Insights

- Our dataset is hightly categorial variable centric. So we can hardly find correlation in our dataset.

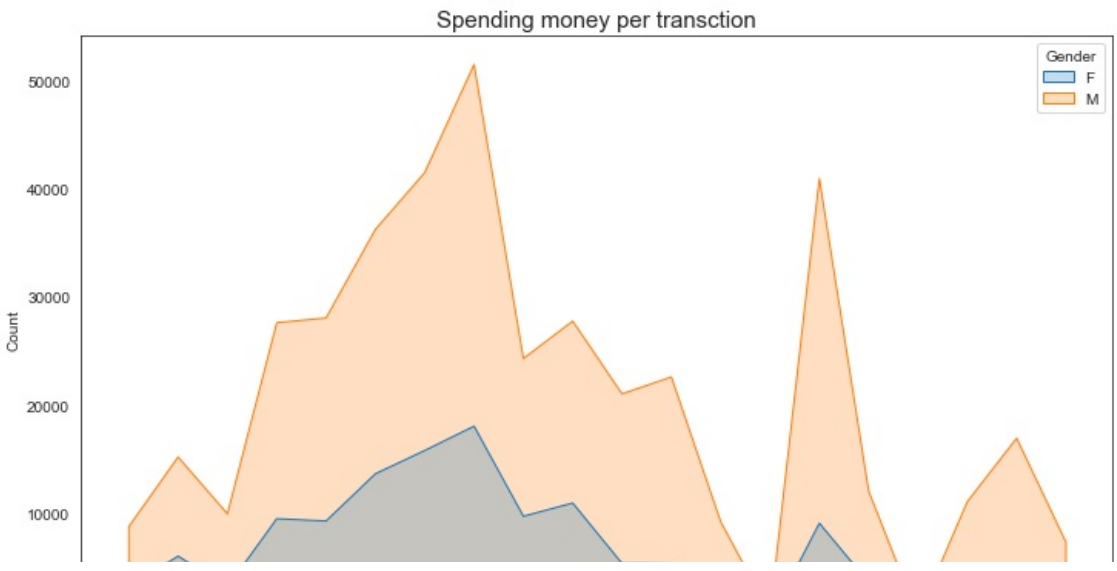
4. Answering questions

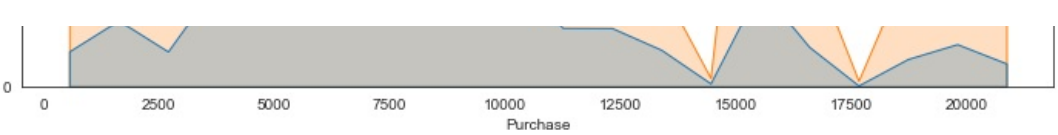
4.1 Are women spending more money per transaction than men? Why or Why not?

In [37]:
```
walmart_new.groupby(["Gender", "City_Category"])["User_ID"].count()
```

Out[37]:
```
Gender  City_Category
F       A                35704
        B                57796
        C                42309
M       A               112016
        B               173377
        C               128866
Name: User_ID, dtype: int64
```

In [38]:
```
plt.figure(figsize=(12, 7))

plt.title("Spending money per transction", fontsize = 15)
sns.histplot(data=walmart_new, x = "Purchase", bins=20, hue = "Gender", element="poly")

plt.show()
```





Insights

- The amount of money spent by women customers per transaction is quite less than that of men. There would be multiple reasons for that : -
  - Socio-economic status
  - Generally male starts earning way before females
  - Generally male customers earns more than females

Lets take sample of data to verify this fact

In [39]:
```
walmart_new.sample(5000, replace=True).groupby(["Gender"])["Purchase"].describe()
```

Out[39]:

Gender	count	mean	std	min	25%	50%	75%	max
F	1276.0	8686.983542	4857.656372	37.0	5386.50	7870.0	11053.0	21400.5
M	3724.0	9452.649570	5100.822250	13.0	5480.75	8099.0	12632.5	21400.5

Insights

- Even after taking sample of 5000 customer, male customers tends to spend more than that of females

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

In [40]:
```
walmart_new.groupby(["Gender"])["Purchase"].describe()
```

Out[40]:

Gender	count	mean	std	min	25%	50%	75%	max
F	135809.0	8726.256327	4743.242716	12.0	5433.0	7914.0	11400.0	21400.5
M	414259.0	9428.373455	5068.732811	12.0	5863.0	8098.0	12454.0	21400.5

In [41]:
```
walmart_new.shape
```

Out[41]:
```
(550068, 10)
```

Insights

- After clipping outlier in our dataset. We have found that out of 550k datapoint -
  - Males are 414K
  - Females are 135K

In [42]:
```
walmart_new_smp_male = walmart_new.loc[walmart["Gender"] == "M"]["Purchase"]
walmart_new_smp_female = walmart_new.loc[walmart["Gender"] == "F"]["Purchase"]
```

In [43]:
```
print("Male customers are: ", walmart_new_smp_male.shape[0])
print("Female customers are: ", walmart_new_smp_female.shape[0])
```

```
Male customers are:  414259
Female customers are:  135809
```

Calculating CI (90%, 95%, 99%) using Bootstrapping for Puchases

based on Gender using CLT

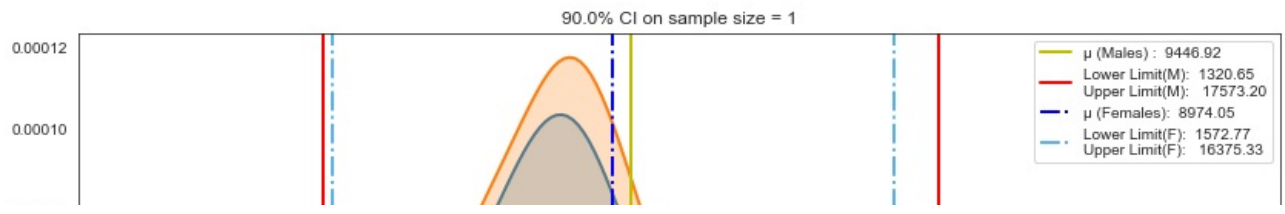
In [44]:

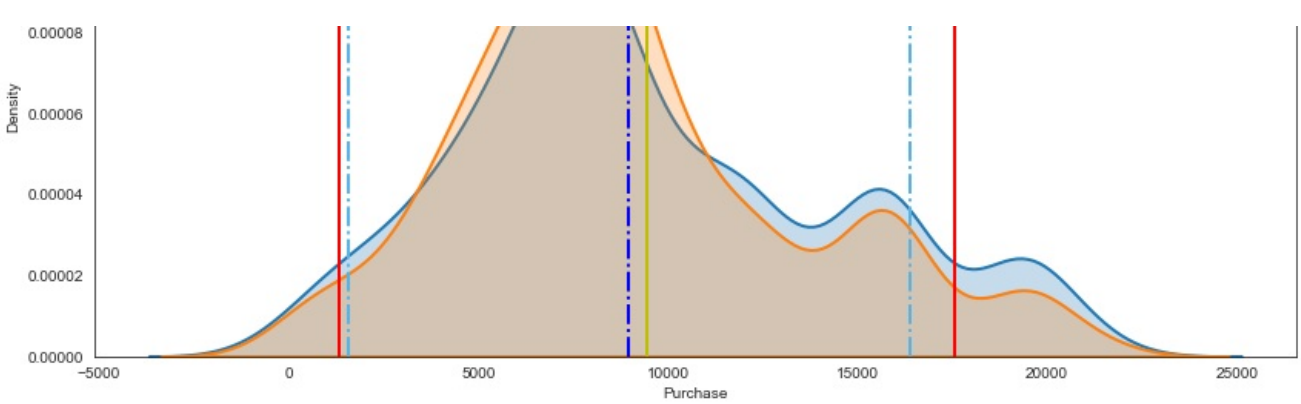
```
def bootstrapping_gender_purchase(sample1,sample2,smp_siz=500,itr_size=5000,confidence_level=0.95,no_of_tails=2):
    smp1_means_m = np.empty(itr_size)
    smp2_means_m = np.empty(itr_size)
    for i in range(itr_size):
        smp1_n = np.empty(smp_siz)
        smp2_n = np.empty(smp_siz)
        smp1_n = np.random.choice(sample1, size = smp_siz,replace=True)
        smp2_n = np.random.choice(sample2, size = smp_siz,replace=True)
        smp1_means_m[i] = np.mean(smp1_n)
        smp2_means_m[i] = np.mean(smp2_n)

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

    # Calculate the mean, standard deviation & standard Error of sampling distribution of a sample mean
    mean1  = np.mean(smp1_means_m)
    sigma1 = np.std(smp1_means_m)
    sem1   = stats.sem(smp1_means_m)

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

    # Calculate the mean, standard deviation & standard Error of sampling distribution of a sample mean
    mean2  = np.mean(smp2_means_m)
    sigma2 = np.std(smp2_means_m)
    sem2   = stats.sem(smp2_means_m)

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

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

    sns.kdeplot(data=smp1_means_m,fill=True,linewidth=2)
    sns.kdeplot(data=smp2_means_m,fill=True,linewidth=2)

    label_mean1=("μ (Males) :  {:.2f}".format(mean1))
    label_ult1=("Lower Limit(M):  {:.2f}\nUpper Limit(M):   {:.2f}".format(lower_limit1,upper_limit1))
    label_mean2=("μ (Females):  {:.2f}".format(mean2))
    label_ult2=("Lower Limit(F):  {:.2f}\nUpper Limit(F):   {:.2f}".format(lower_limit2,upper_limit2))

    plt.title(f"{confidence_level * 100}% CI on sample size = {smp_siz}")
    plt.xlabel('Purchase')
    plt.axvline(mean1, color = 'y', linestyle = 'solid', linewidth = 2,label=label_mean1)
    plt.axvline(upper_limit1, color = 'r', linestyle = 'solid', linewidth = 2,label=label_ult1)
    plt.axvline(lower_limit1, color = 'r', linestyle = 'solid', linewidth = 2)
    plt.axvline(mean2, color = 'b', linestyle = 'dashdot', linewidth = 2,label=label_mean2)
    plt.axvline(upper_limit2, color = '#56B4E9', linestyle = 'dashdot', linewidth = 2,label=label_ult2)
    plt.axvline(lower_limit2, color = '#56B4E9', linestyle = 'dashdot', linewidth = 2)
    plt.legend(loc='best')

    plt.show()

    return smp1_means_m,smp2_means_m ,np.round(lower_limit1,2),np.round(upper_limit1,2),np.round(lower_limit2,2),
```

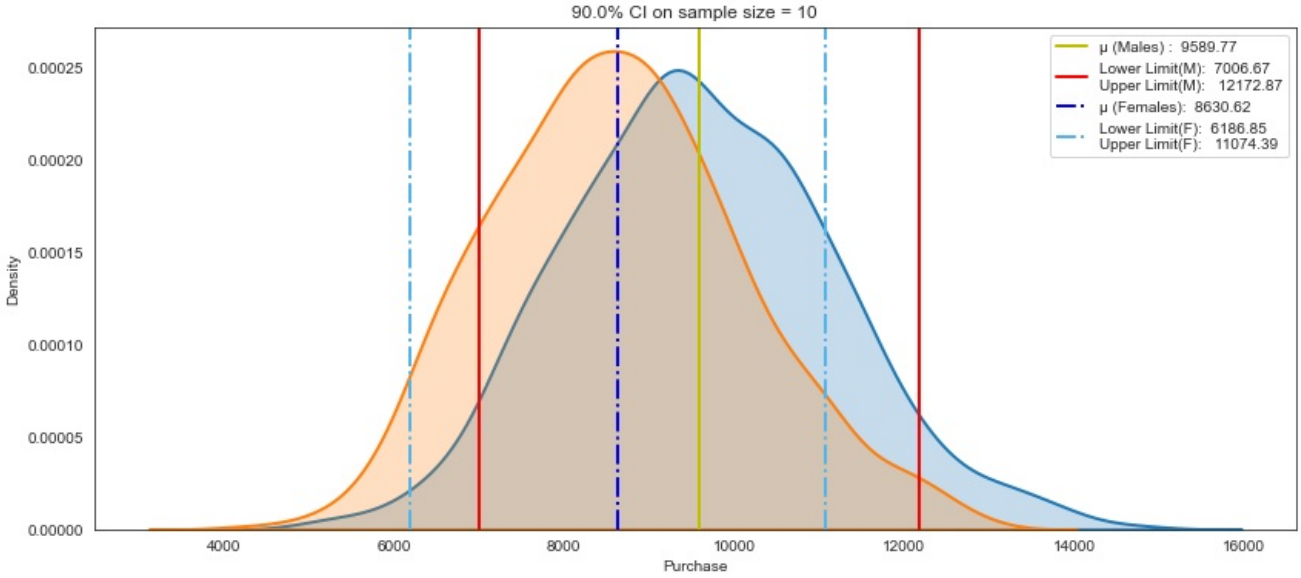
Calculation 90% CI

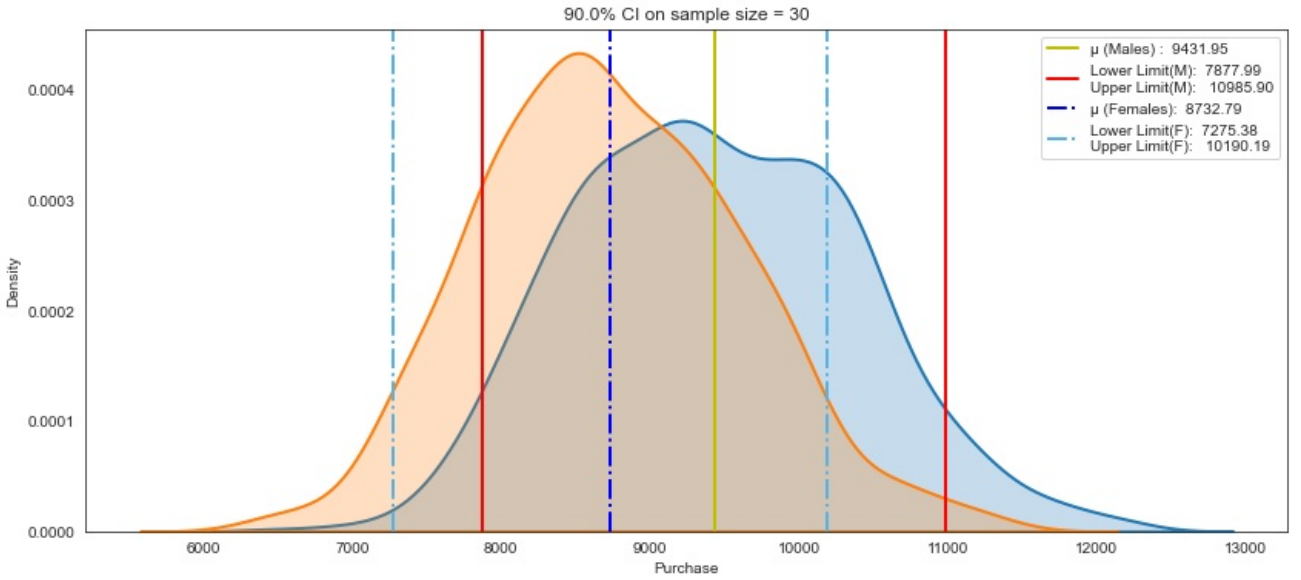
In [45]:

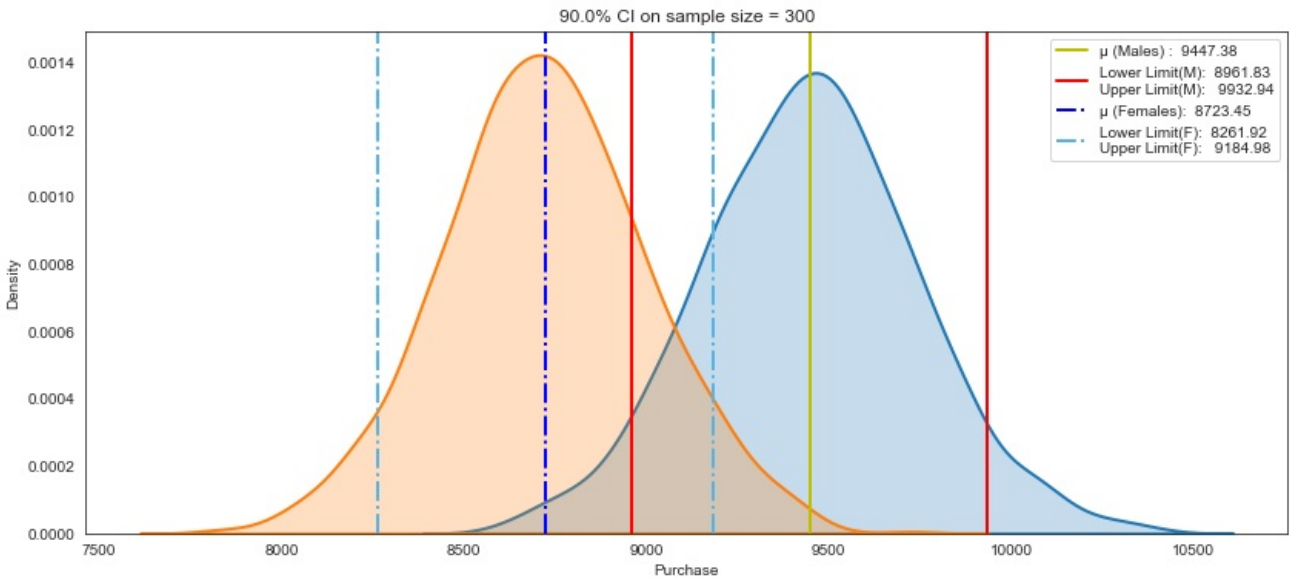
```
itr_size = 1000
size_list = [1, 10, 30, 300, 1000, 5000, 100000]
ci = 0.90

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

for smp_siz in size_list:
    m_avg, f_avg, ll_m, ul_m, ll_f, ul_f = bootstrapping_gender_purchase(walmart_new_smp_male,walmart_new_smp_fem

    array = np.append(array, np.array([['M', ll_m, ul_m, smp_siz, ([ll_m,ul_m]) ,(ul_m-ll_m),90]]), axis=0)
    array = np.append(array, np.array([['F', ll_f, ul_f, smp_siz, ([ll_f,ul_f]) ,(ul_f-ll_f),90]]), axis=0)

overlap = pd.DataFrame(array, columns = ['Gender','Lower_limit','Upper_limit','Sample_Size','CI','Range','Confide
```

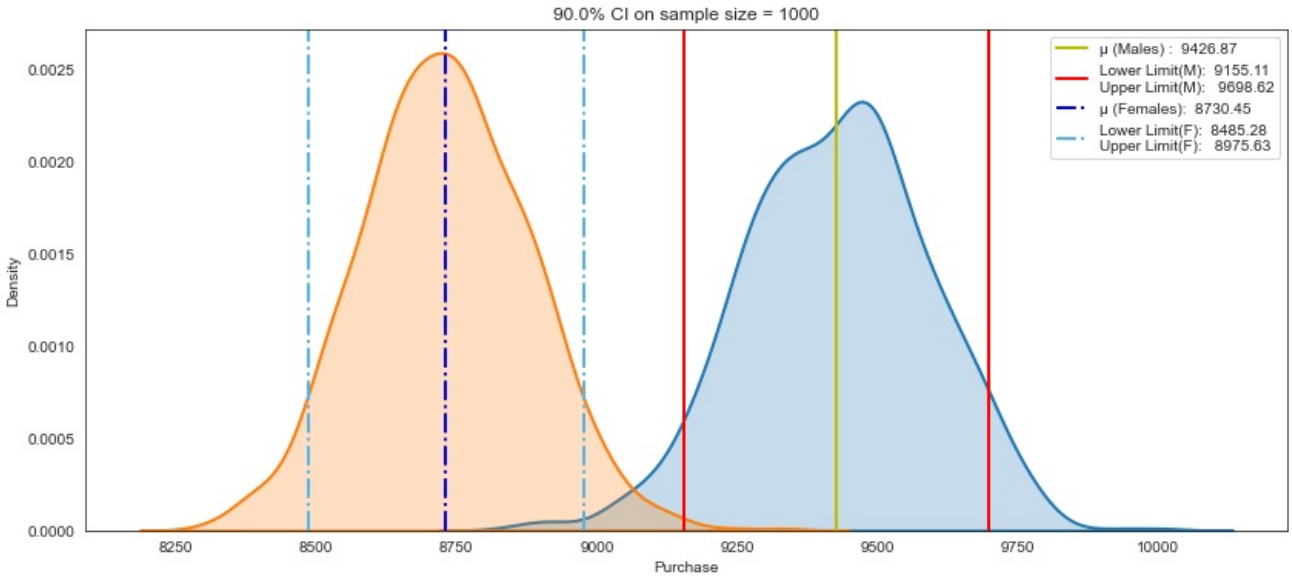


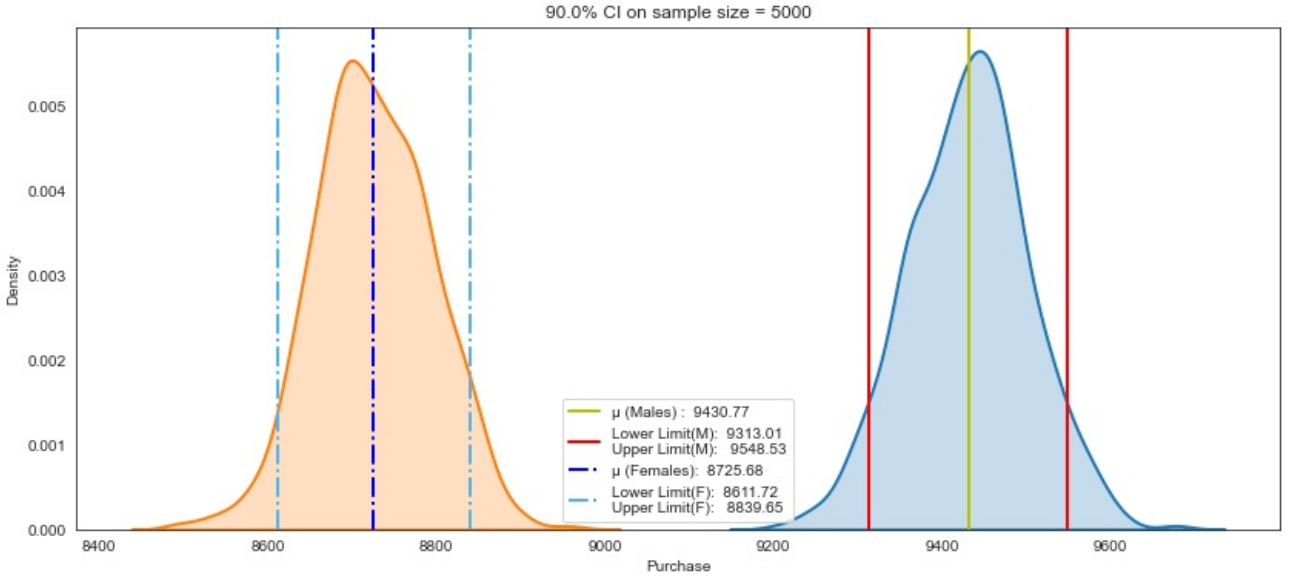

Density
0.00008
0.00006
0.00004
0.00002
0.00000
−5000
0
5000
10000
15000
20000
25000
Purchase

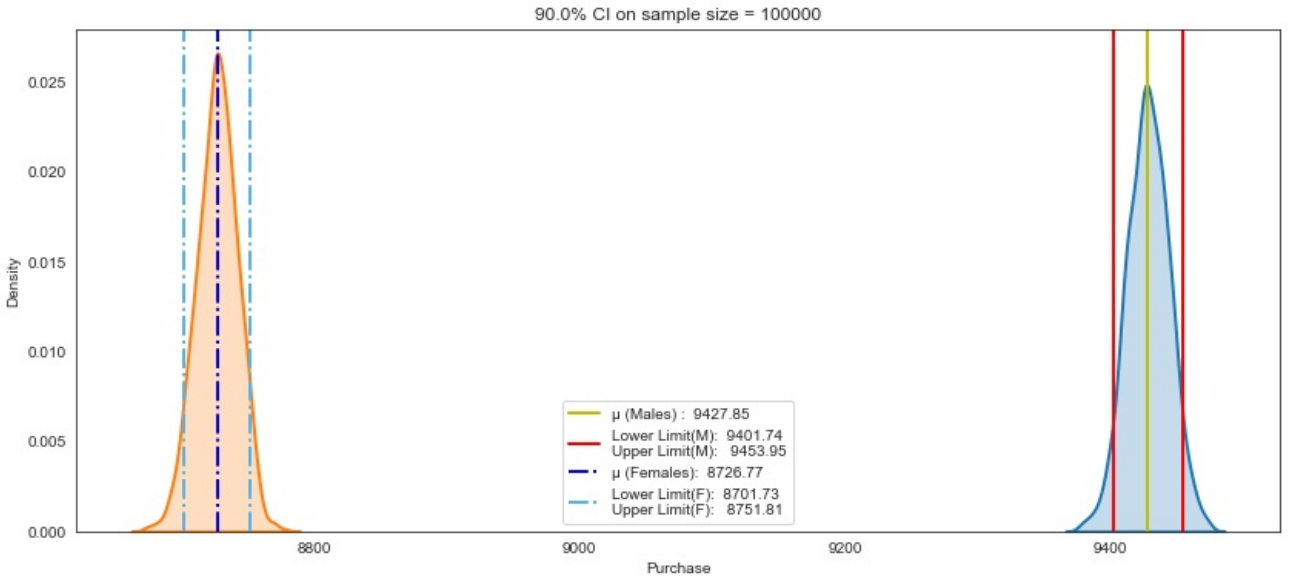

90.0% CI on sample size = 10
μ (Males) : 9589.77
Lower Limit(M): 7006.67
Upper Limit(M): 12172.87
μ (Females): 8630.62
Lower Limit(F): 6186.85
Upper Limit(F): 11074.39
Density
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
4000
6000
8000
10000
12000
14000
16000
Purchase


90.0% CI on sample size = 30
μ (Males) : 9431.95
Lower Limit(M): 7877.99
Upper Limit(M): 10985.90
μ (Females): 8732.79
Lower Limit(F): 7275.38
Upper Limit(F): 10190.19
Density
0.0004
0.0003
0.0002
0.0001
0.0000
6000
7000
8000
9000
10000
11000
12000
13000
Purchase


90.0% CI on sample size = 300
μ (Males) : 9447.38
Lower Limit(M): 8961.83
Upper Limit(M): 9932.94
μ (Females): 8723.45
Lower Limit(F): 8261.92
Upper Limit(F): 9184.98
Density
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
7500
8000
8500
9000
9500
10000
10500
Purchase







In [ ]:

In [46]:
```
overlap.loc[(overlap["Gender"] == "M") & (overlap["Sample_Size"] > 300)]
```

Out[46]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
8	M	9155.11	9698.62	1000		543.51	90

					[9155.11, 9698.62]		
10	M	9313.01	9548.53	5000	[9313.01, 9548.53]	235.52	90
12	M	9401.74	9453.95	100000	[9401.74, 9453.95]	52.21	90

In [47]:
```
overlap.loc[(overlap["Gender"] == "F") & (overlap["Sample_Size"] > 300)]
```

Out[47]:

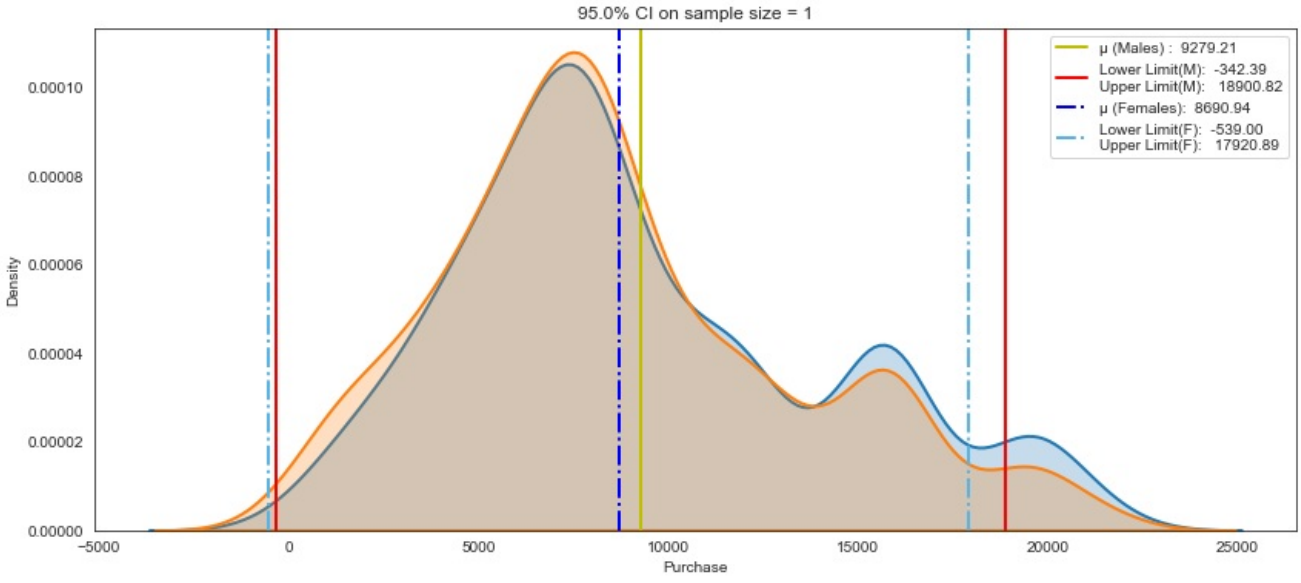
	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
9	F	8485.28	8975.63	1000	[8485.28, 8975.63]	490.35	90
11	F	8611.72	8839.65	5000	[8611.72, 8839.65]	227.93	90
13	F	8701.73	8751.81	100000	[8701.73, 8751.81]	50.08	90

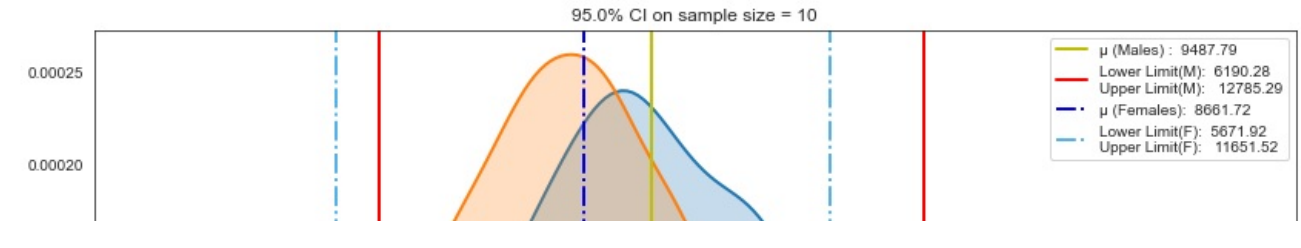
Insights

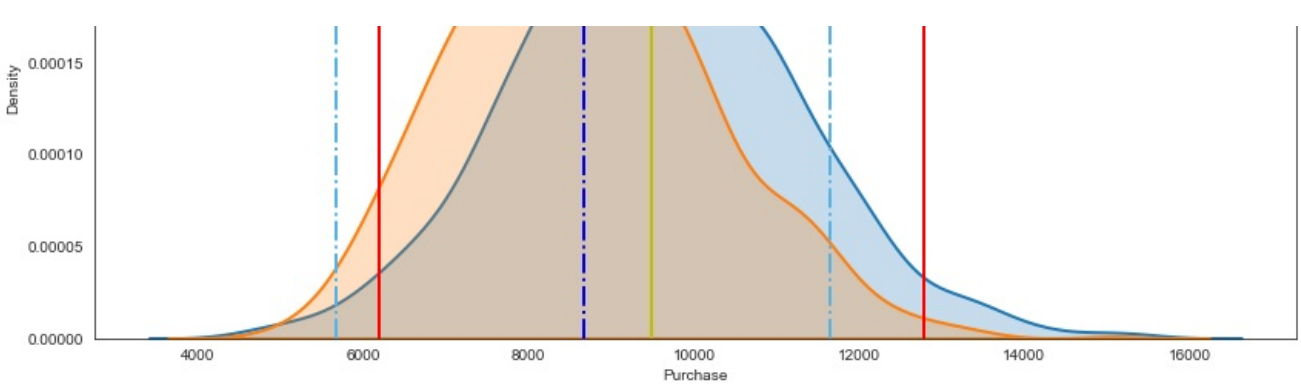
- As the sample size increases, the two groups start to become distinct
- With increasing sample size, Standard error of the mean in the samples decreases.
- For Female (sample size 100000) range for mean purchase with confidence interval 90% is [8702.35, 8751.09]
- For Male range for mean purchase with confidence interval 90% is [9402.28, 9455.2]
- There is no overlapping found

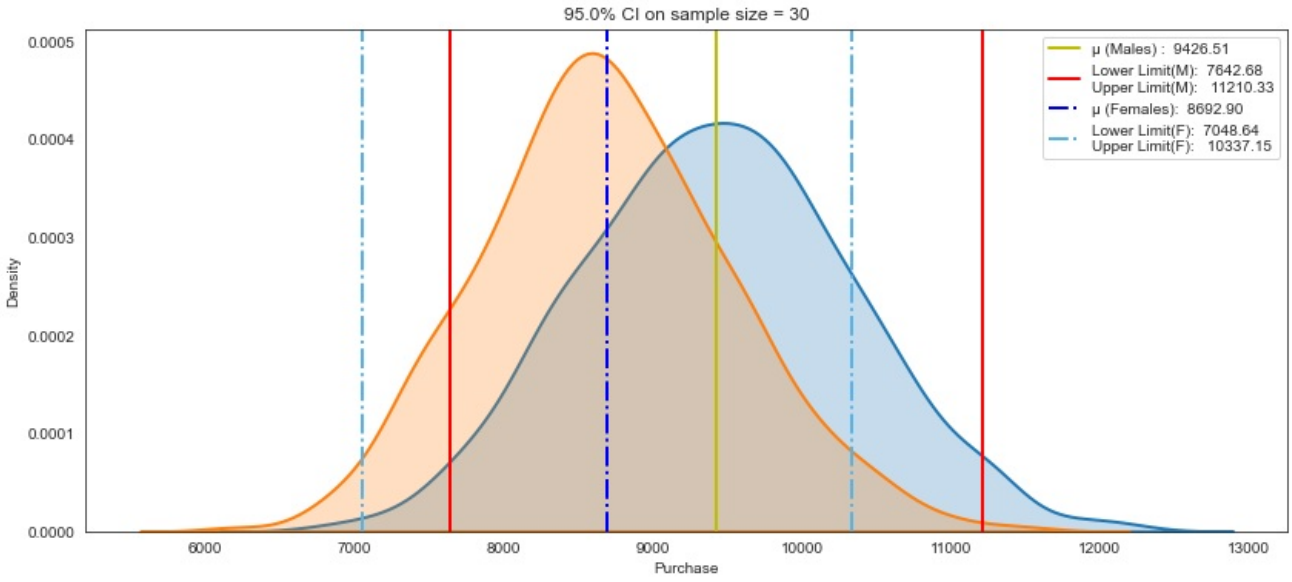
Calculate 95% CI

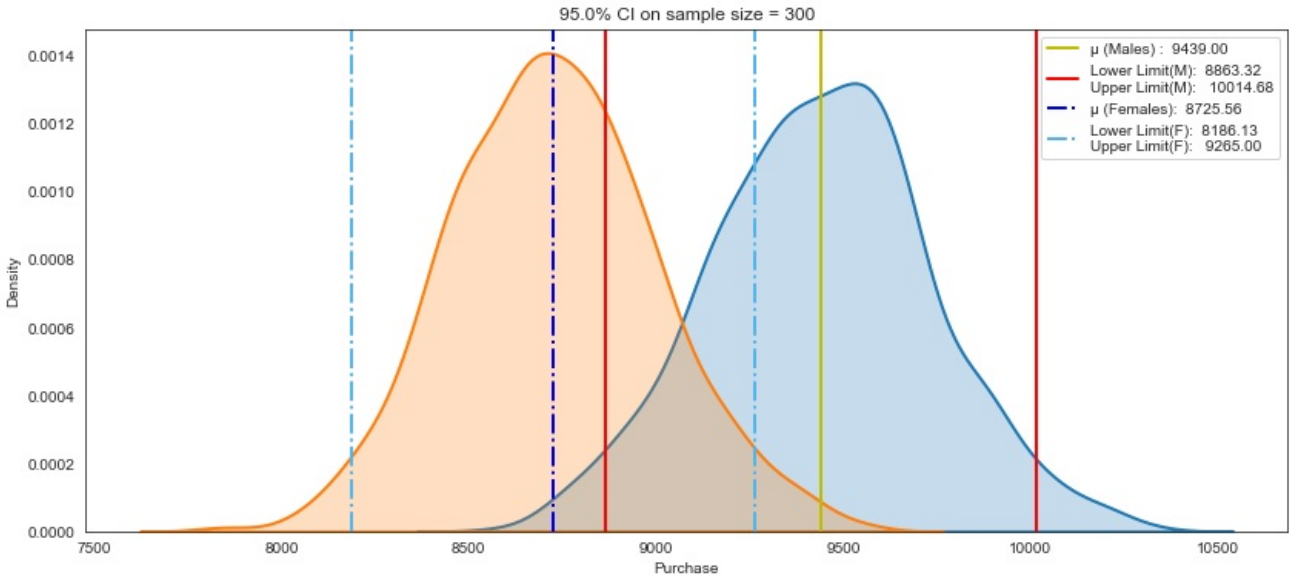
In [48]:
```
itr_size = 1000
size_list = [1, 10, 30, 300, 1000, 5000, 100000]
ci = 0.95

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

for smp_siz in size_list:
    m_avg, f_avg, ll_m, ul_m, ll_f, ul_f = bootstrapping_gender_purchase(walmart_new_smp_male,walmart_new_smp_fem

    array = np.append(array, np.array([['M', ll_m, ul_m, smp_siz, ([ll_m,ul_m]) ,(ul_m-ll_m),95]]), axis=0)
    array = np.append(array, np.array([['F', ll_f, ul_f, smp_siz, ([ll_f,ul_f]) ,(ul_f-ll_f),95]]), axis=0)

overlap = pd.DataFrame(array, columns = ['Gender','Lower_limit','Upper_limit','Sample_Size','CI','Range','Confide
```

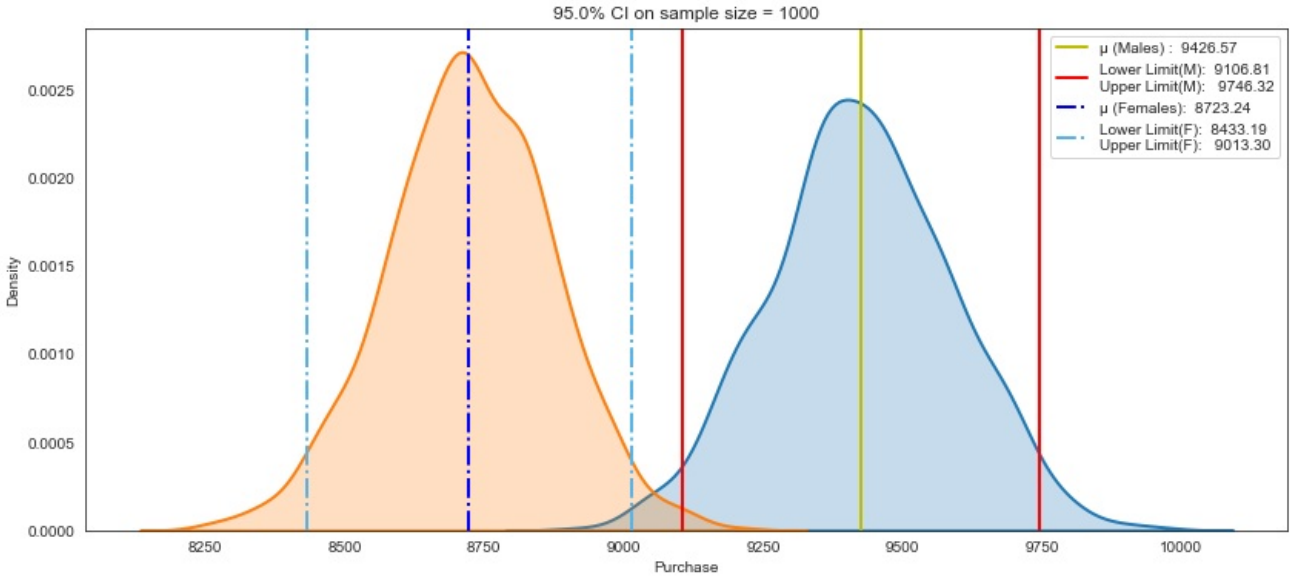


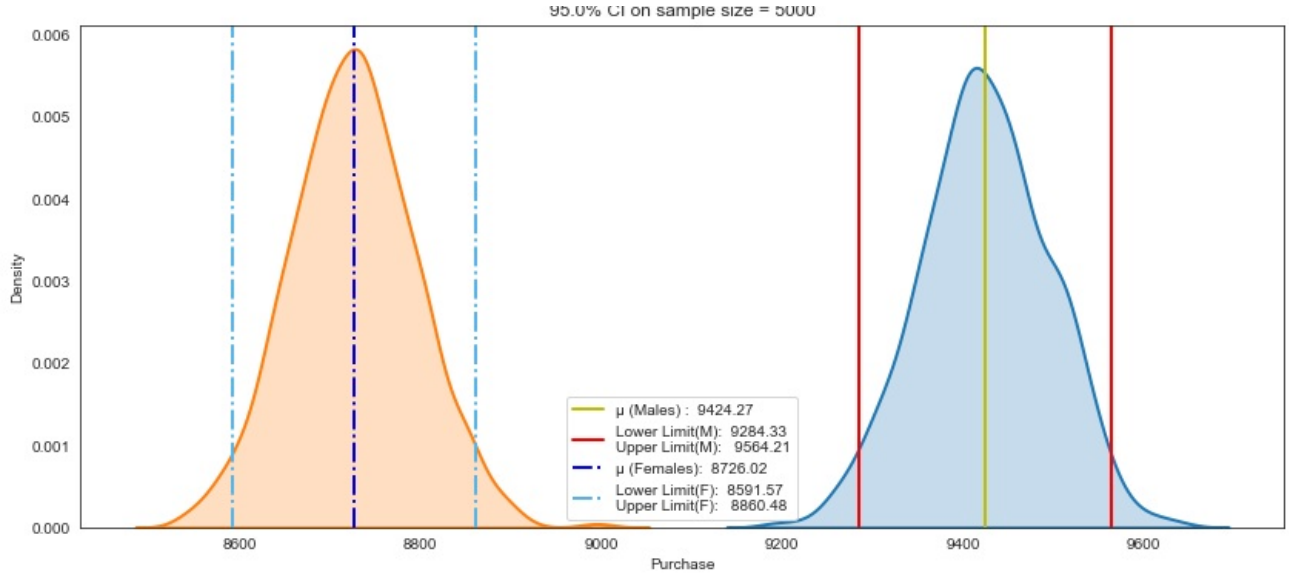


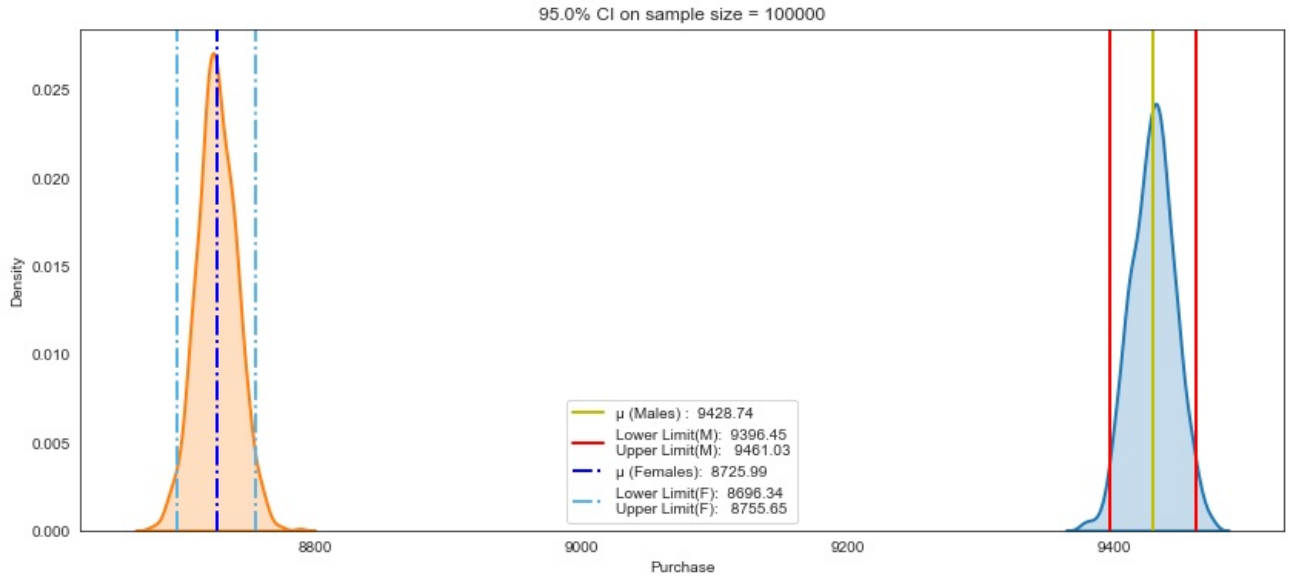

Density
0.00015
0.00010
0.00005
0.00000
4000
6000
8000
10000
12000
14000
16000
Purchase


95.0% CI on sample size = 30
μ (Males) : 9426.51
Lower Limit(M): 7642.68
Upper Limit(M): 11210.33
μ (Females): 8692.90
Lower Limit(F): 7048.64
Upper Limit(F): 10337.15
Density
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
6000
7000
8000
9000
10000
11000
12000
13000
Purchase


95.0% CI on sample size = 300
μ (Males) : 9439.00
Lower Limit(M): 8863.32
Upper Limit(M): 10014.68
μ (Females): 8725.56
Lower Limit(F): 8186.13
Upper Limit(F): 9265.00
Density
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
7500
8000
8500
9000
9500
10000
10500
Purchase


95.0% CI on sample size = 1000
μ (Males) : 9426.57
Lower Limit(M): 9106.81
Upper Limit(M): 9746.32
μ (Females): 8723.24
Lower Limit(F): 8433.19
Upper Limit(F): 9013.30
Density
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
8250
8500
8750
9000
9250
9500
9750
10000
Purchase





In [49]:
```
overlap.loc[(overlap["Gender"] == "M") & (overlap["Sample_Size"] > 300)]
```

Out[49]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
8	M	9106.81	9746.32	1000	[9106.81, 9746.32]	639.51	95
10	M	9284.33	9564.21	5000	[9284.33, 9564.21]	279.88	95
12	M	9396.45	9461.03	100000	[9396.45, 9461.03]	64.58	95

In [50]:
```
overlap.loc[(overlap["Gender"] == "F") & (overlap["Sample_Size"] > 300)]
```

Out[50]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
9	F	8433.19	9013.3	1000	[8433.19, 9013.3]	580.11	95
11	F	8591.57	8860.48	5000	[8591.57, 8860.48]	268.91	95
13	F	8696.34	8755.65	100000	[8696.34, 8755.65]	59.31	95

Insights

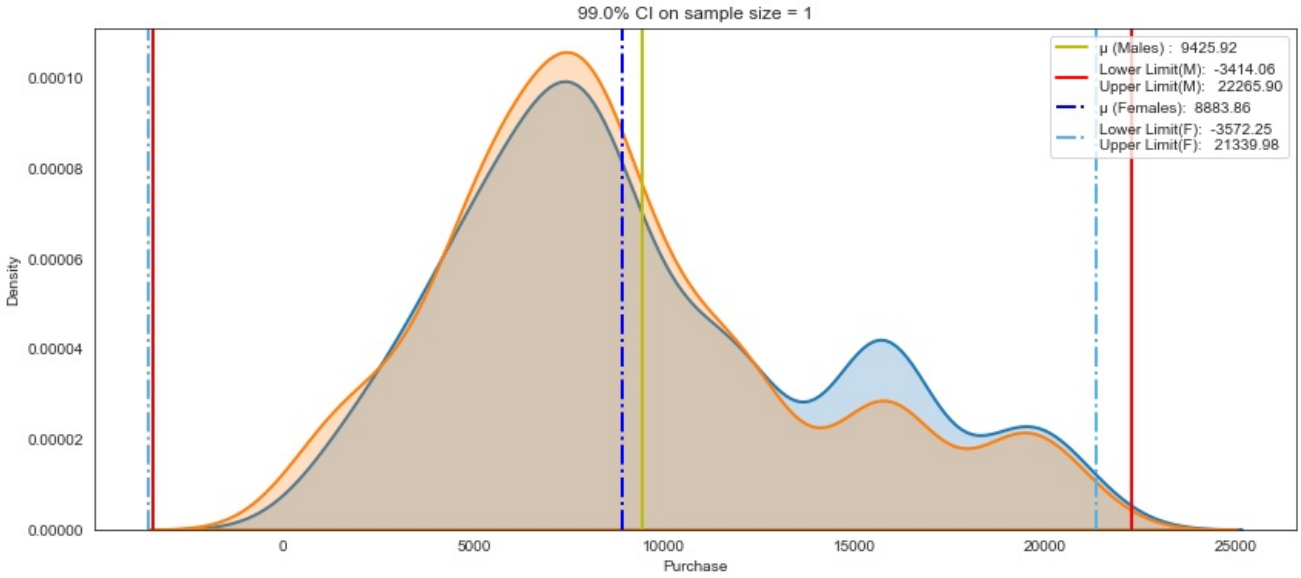
- Using confidence interval 95%, the mean purchase value by gender shows a similar pattern to that found with confidence interval 90%-
- As the sample size increases, the Male and female groups start to become distinct
- With increasing sample size, Standard error of the mean in the samples decreases. For sample size 100000 is 0.47

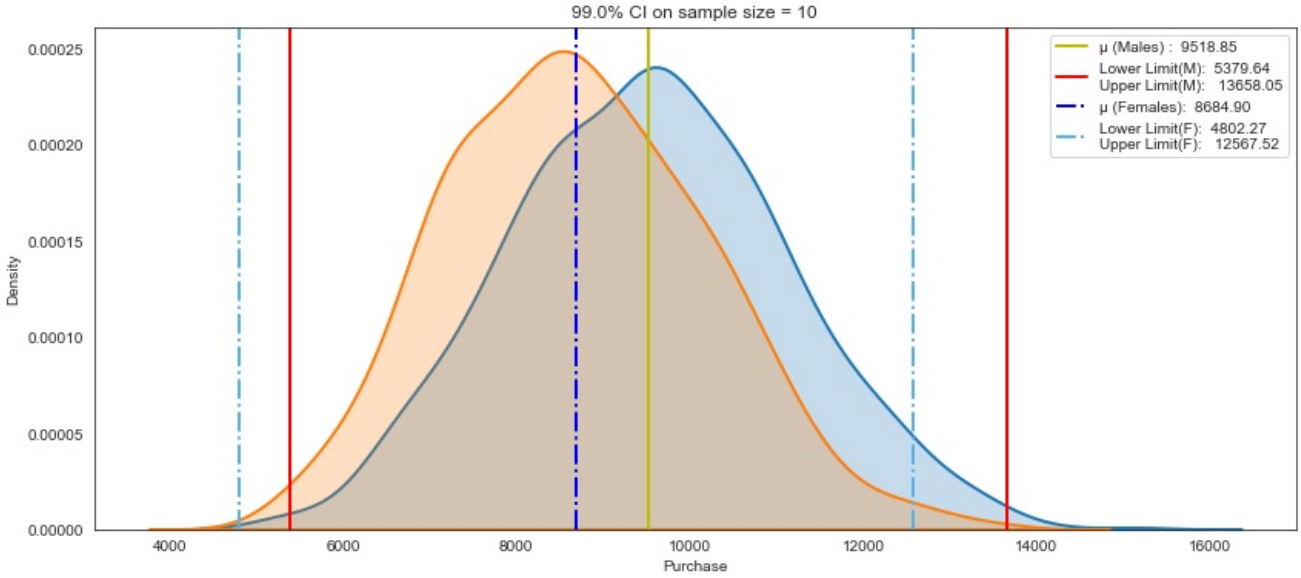
- For Female (sample size 100000) range for mean purchase with confidence interval 90% is [8695.0, 8755.56]
- For Male range for mean purchase with confidence interval 95% is [9397.76, 9460.52]

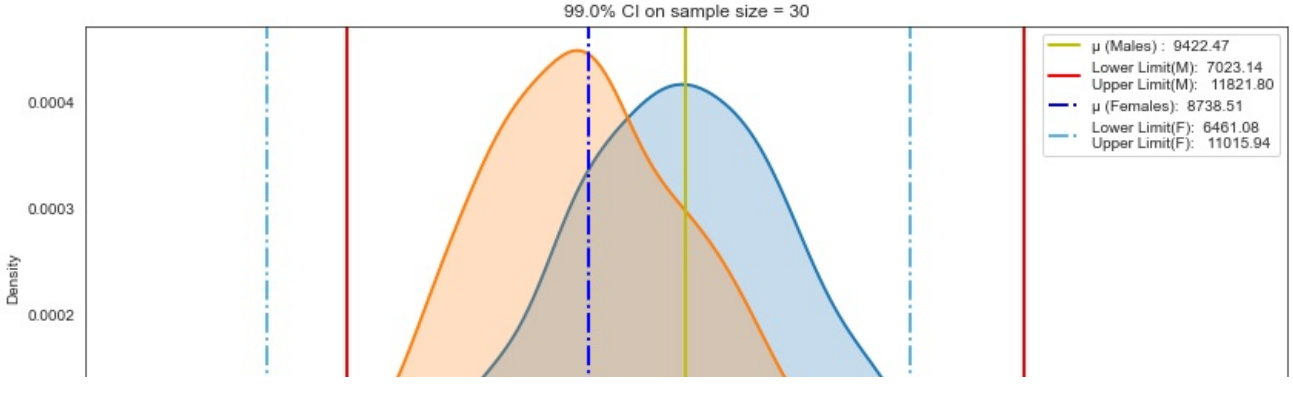
Calculate 99% CI

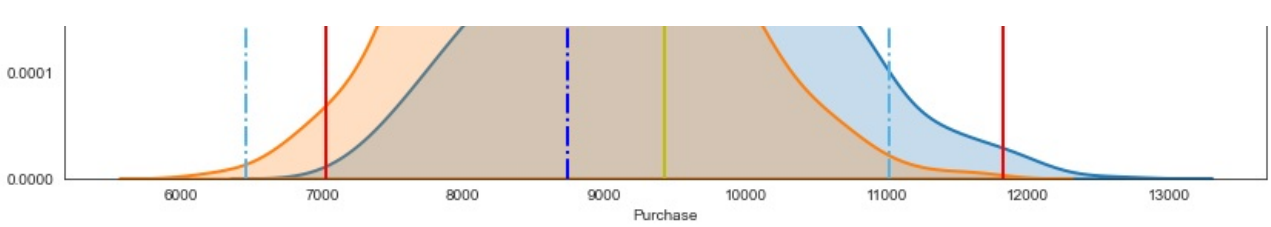
In [51]:

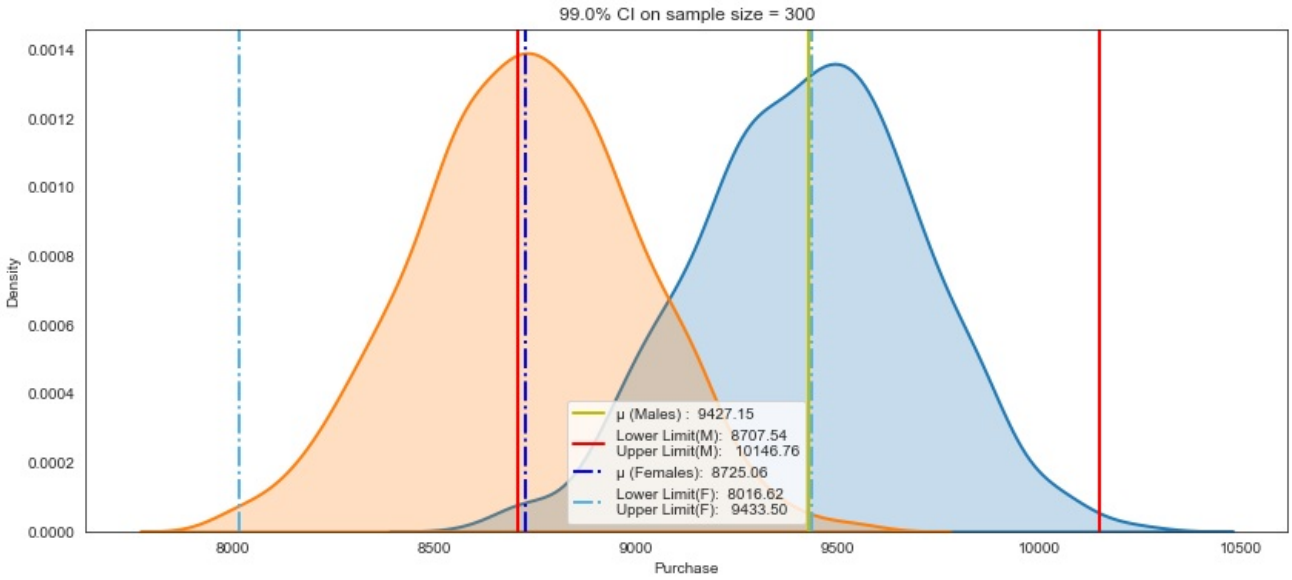
```
itr_size = 1000
size_list = [1, 10, 30, 300, 1000, 5000, 100000]
ci = 0.99

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

for smp_siz in size_list:
    m_avg, f_avg, ll_m, ul_m, ll_f, ul_f = bootstrapping_gender_purchase(walmart_new_smp_male,walmart_new_smp_fem

    array = np.append(array, np.array([['M', ll_m, ul_m, smp_siz, ([ll_m,ul_m]) ,(ul_m-ll_m),99]]), axis=0)
    array = np.append(array, np.array([['F', ll_f, ul_f, smp_siz, ([ll_f,ul_f]) ,(ul_f-ll_f),99]]), axis=0)

overlap = pd.DataFrame(array, columns = ['Gender','Lower_limit','Upper_limit','Sample_Size','CI','Range','Confide
```

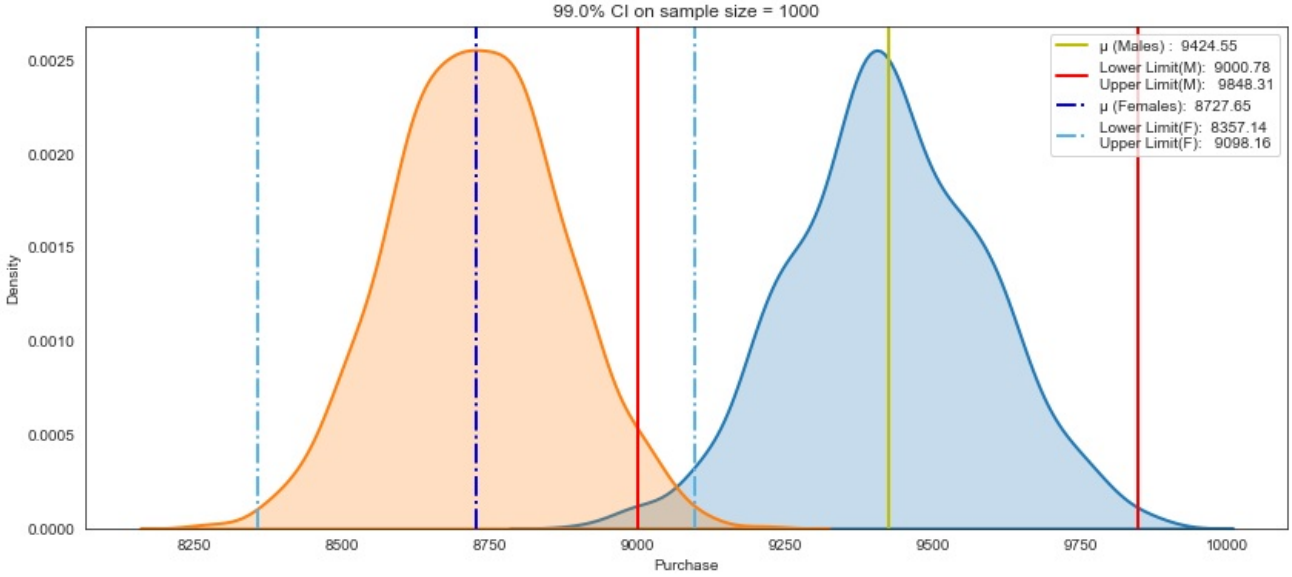


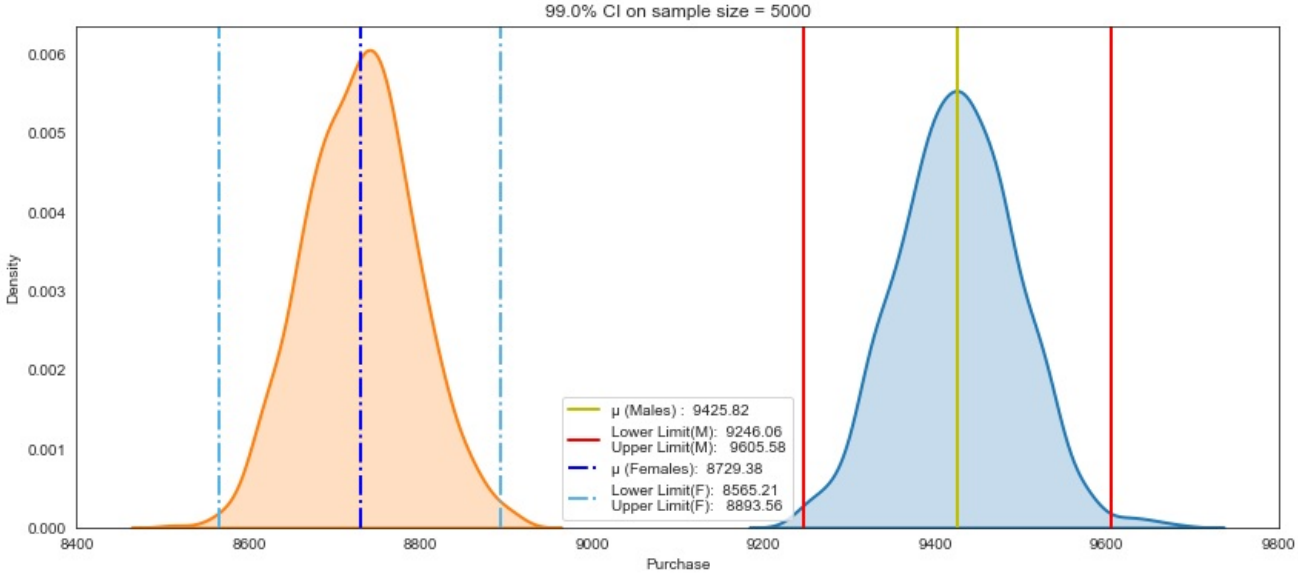


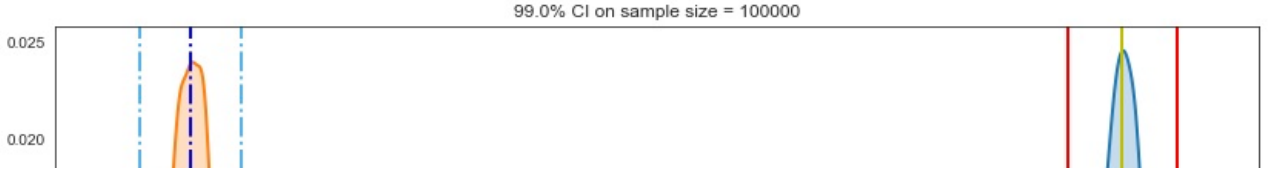


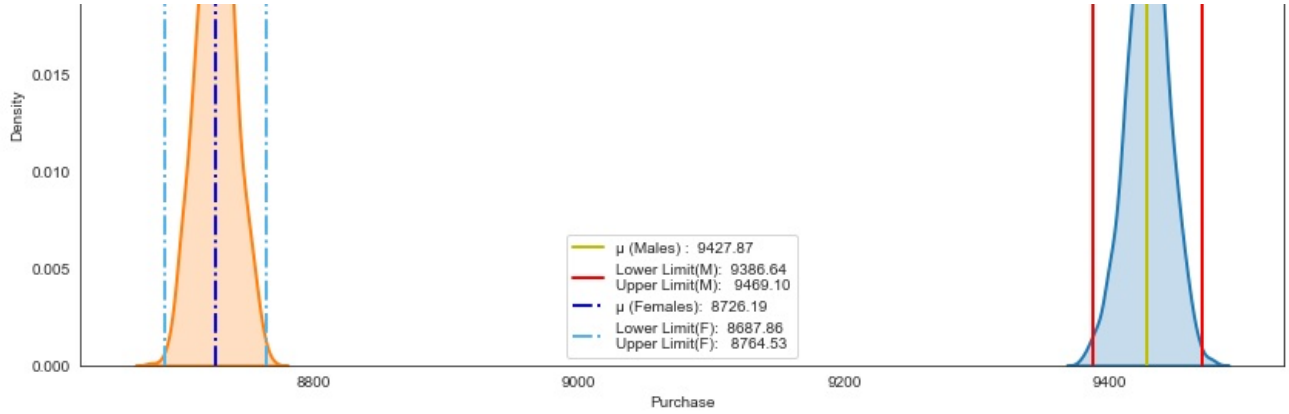

0.0001
0.0000
6000
7000
8000
9000
10000
11000
12000
13000
Purchase


99.0% CI on sample size = 300
Density
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
8000
8500
9000
9500
10000
10500
Purchase
μ (Males) : 9427.15
Lower Limit(M): 8707.54
Upper Limit(M): 10146.76
μ (Females): 8725.06
Lower Limit(F): 8016.62
Upper Limit(F): 9433.50


99.0% CI on sample size = 1000
Density
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
8250
8500
8750
9000
9250
9500
9750
10000
Purchase
μ (Males) : 9424.55
Lower Limit(M): 9000.78
Upper Limit(M): 9848.31
μ (Females): 8727.65
Lower Limit(F): 8357.14
Upper Limit(F): 9098.16


99.0% CI on sample size = 5000
Density
0.006
0.005
0.004
0.003
0.002
0.001
0.000
8400
8600
8800
9000
9200
9400
9600
9800
Purchase
μ (Males) : 9425.82
Lower Limit(M): 9246.06
Upper Limit(M): 9605.58
μ (Females): 8729.38
Lower Limit(F): 8565.21
Upper Limit(F): 8893.56


99.0% CI on sample size = 100000
0.025
0.020



In [52]:
```
overlap.loc[(overlap["Gender"] == "M") & (overlap["Sample_Size"] > 300)]
```

Out[52]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
8	M	9000.78	9848.31	1000	[9000.78, 9848.31]	847.53	99
10	M	9246.06	9605.58	5000	[9246.06, 9605.58]	359.52	99
12	M	9386.64	9469.1	100000	[9386.64, 9469.1]	82.46	99

In [53]:
```
overlap.loc[(overlap["Gender"] == "F") & (overlap["Sample_Size"] > 300)]
```

Out[53]:

	Gender	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
9	F	8357.14	9098.16	1000	[8357.14, 9098.16]	741.02	99
11	F	8565.21	8893.56	5000	[8565.21, 8893.56]	328.35	99
13	F	8687.86	8764.53	100000	[8687.86, 8764.53]	76.67	99

Insights

- Using confidence interval 99%, the mean purchase value by gender shows a similar pattern to that found with confidence interval 90% & 95%-
- As the sample size increases, the Male and female groups start to become distinct
- With increasing sample size, Standard error of the mean in the samples decreases. For sample size 100000 is 0.45
- For Female (sample size 100000) range for mean purchase with confidence interval 99% is [8688.31, 8763.93]
- For Male range for mean purchase with confidence interval 90% is [9388.64, 9469.8]

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

Answer

- **Overlappings are increasing with a confidence interval of 95%. Due to the increasing CI, we consider higher ranges within which the actual population might fall, so that both mean purchase are more likely to fall within the same range.**
- **When the confidence percentage increases, the spread, that is the difference between the upper and lower limits, also increases. For Female Confidence percent as [90,95,99] have difference between the upper & lower limits as [50.46,59,73.31]**

Conclusion to make changes and improvements

- **As per analysis that females spend less than males on average, management needs to focus on their specific needs differently. Adding some additional offers for women can increase their spending on Black Friday.**

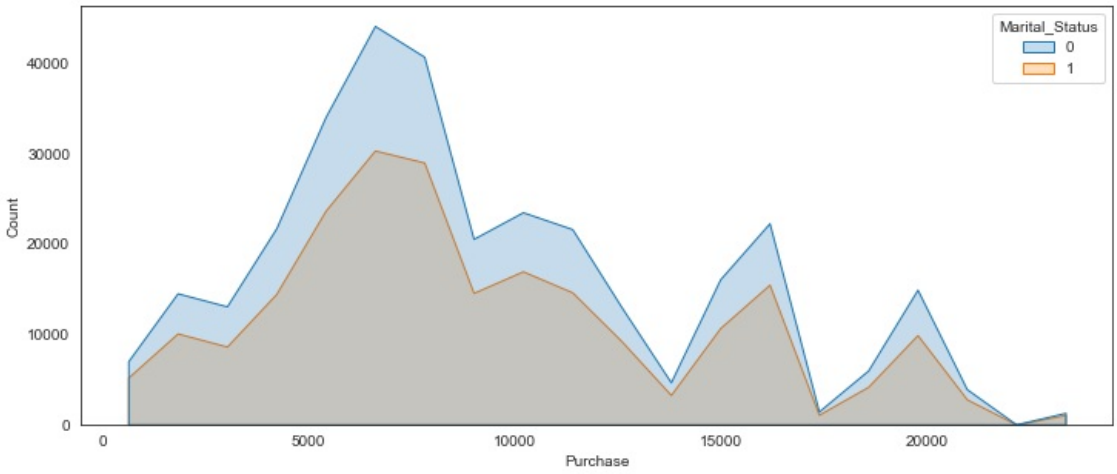
4.4 Results when the same activity is performed for Married vs Unmarried

In [54]:
```
walmart_new.groupby(["Marital_Status"])["Purchase"].describe()
```

Out[54]:

Marital_Status	count	mean	std	min	25%	50%	75%	max
0	324731.0	9257.517959	5005.287278	12.0	5605.0	8044.0	12061.0	21400.5
1	225337.0	9251.430702	4991.196933	12.0	5843.0	8051.0	12042.0	21400.5

In [55]:
```
plt.figure(figsize = (12, 5))

sns.histplot(data = walmart, x = "Purchase", hue = "Marital_Status", element="poly", bins = 20)

plt.show()
```



Insights

- Single customers tends to purchase slightly more than married one

Calculating CI (90%, 95%, 99%) using Bootstrapping for Puchases based on Marital_Status using CLT

In [56]:
```
def bootstrapping_marital_status(sample1,sample2,smp_siz=500,itr_size=5000,confidence_level=0.95,no_of_tails=2):

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

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

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

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

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

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

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

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

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

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

    label_mean1=("μ (Married) :  {:.2f}".format(mean1))
    label_ult1=("Lower Limit(M):  {:.2f}\nUpper Limit(M):   {:.2f}".format(lower_limit1,upper_limit1))
    label_mean2=("μ (Unmarried):  {:.2f}".format(mean2))
    label_ult2=("Lower Limit(F):  {:.2f}\nUpper Limit(F):   {:.2f}".format(lower_limit2,upper_limit2))

    plt.title(f"{confidence_level*100}% CI on Sample Size = {smp_siz}")
    plt.xlabel('Purchase')
    plt.axvline(mean1, color = 'y', linestyle = 'solid', linewidth = 2,label=label_mean1)
    plt.axvline(upper_limit1, color = 'r', linestyle = 'solid', linewidth = 2,label=label_ult1)
    plt.axvline(lower_limit1, color = 'r', linestyle = 'solid', linewidth = 2)
    plt.axvline(mean2, color = 'b', linestyle = 'dashdot', linewidth = 2,label=label_mean2)
    plt.axvline(upper_limit2, color = '#56B4E9', linestyle = 'dashdot', linewidth = 2,label=label_ult2)
    plt.axvline(lower_limit2, color = '#56B4E9', linestyle = 'dashdot', linewidth = 2)
    plt.legend(loc='upper right')

    plt.show()

    return smp1_means_m,smp2_means_m ,np.round(lower_limit1,2),np.round(upper_limit1,2),np.round(lower_limit2,2),
```

In [57]:

```
walmart_new_smp_married = walmart.loc[walmart_new["Marital_Status"] == 1]["Purchase"]
walmart_new_smp_unmarried = walmart.loc[walmart_new["Marital_Status"] == 0]["Purchase"]

print(f"Married Customers: {walmart_new_smp_married.shape[0]}\nUnmarried Customers: {walmart_new_smp_unmarried[0]
```

```
Married Customers: 225337
Unmarried Customers: 8370
```

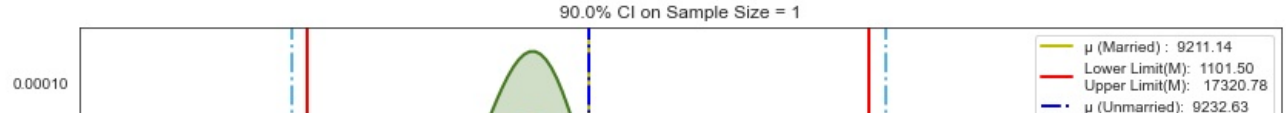
Insights

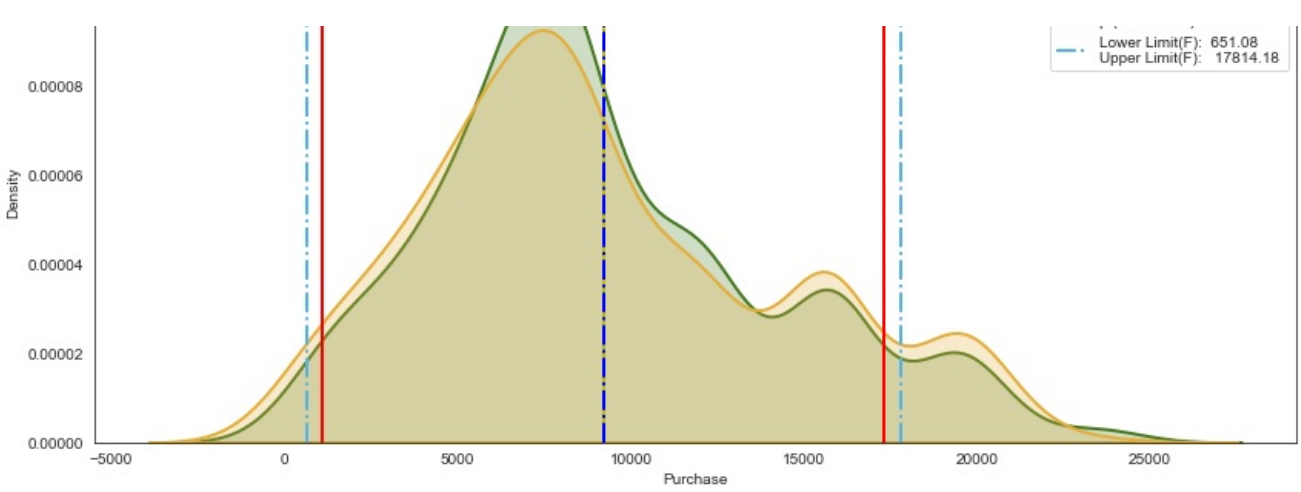
- Infact most of the customers are married but the single customer tends to purchase more than married one

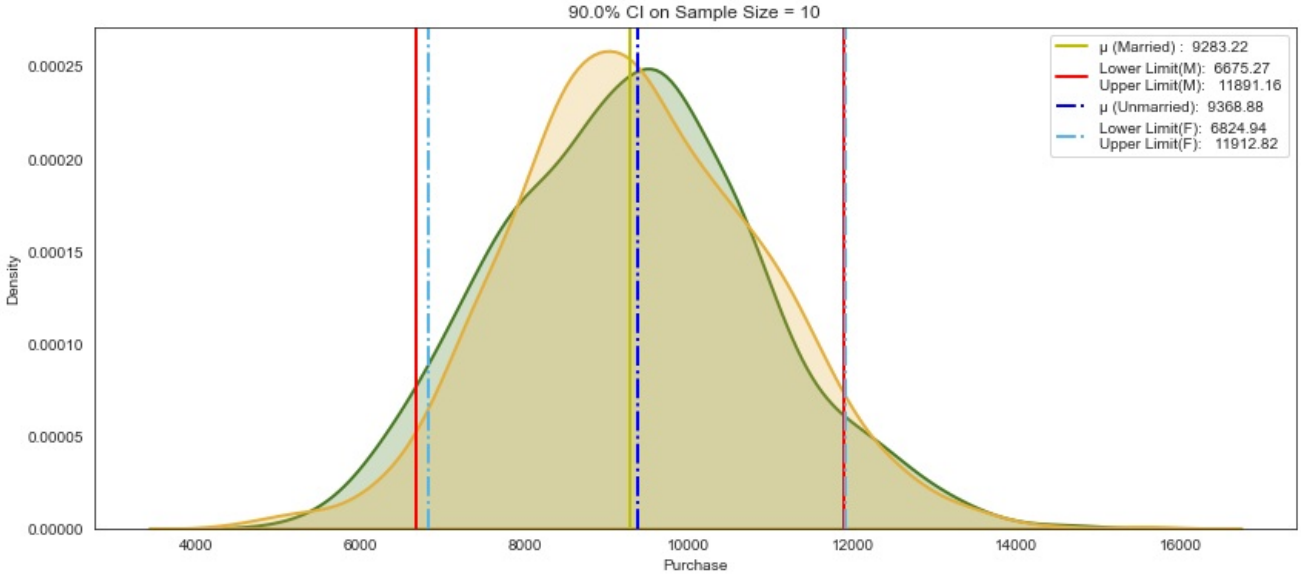
Calculating 90% CI

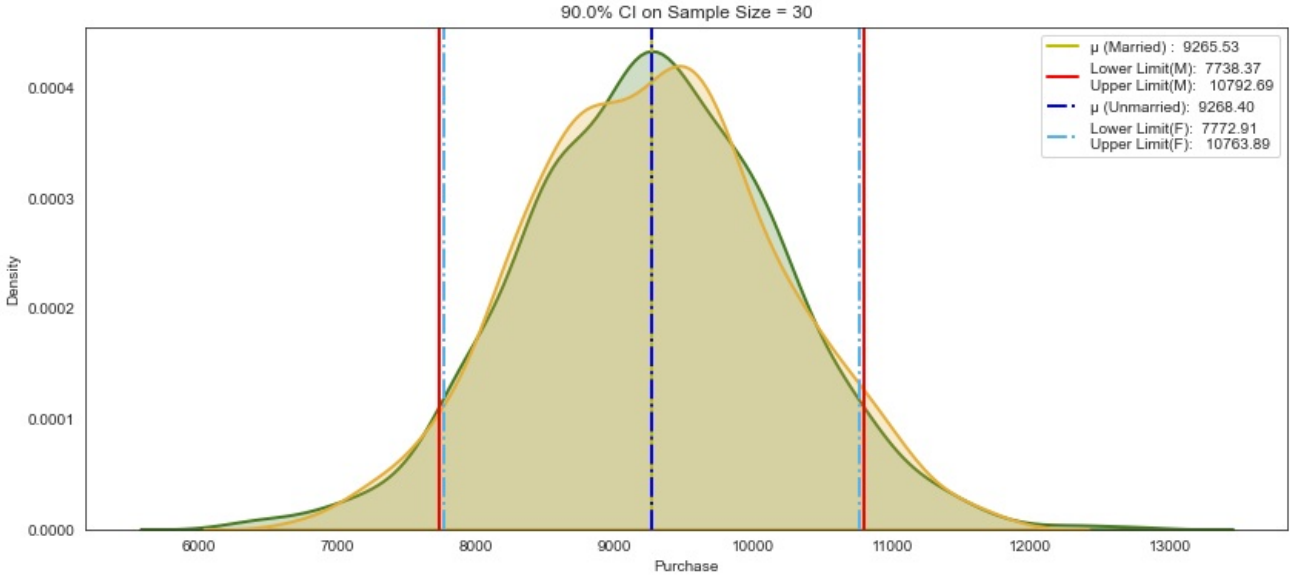
In [58]:

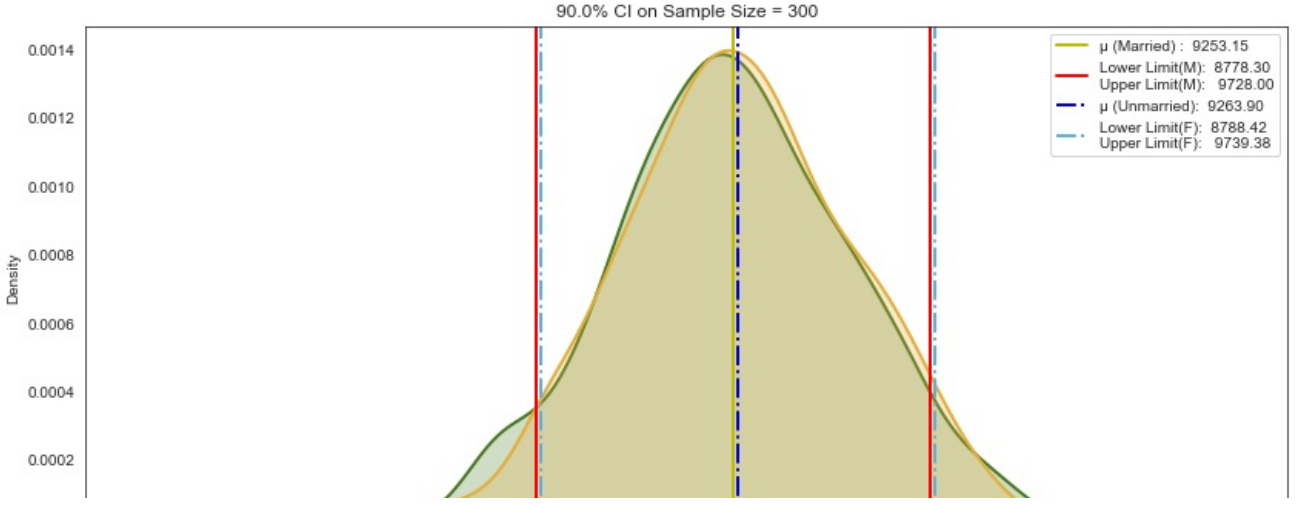
```
itr_size = 1000
size_list = [1, 10, 30, 300, 1000, 100000]
ci = 0.90

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

for smp_siz in size_list:
    m_avg, f_avg, ll_m, ul_m, ll_u, ul_u = bootstrapping_marital_status(walmart_new_smp_married,walmart_new_smp_u

    array = np.append(array, np.array([['Married', ll_m, ul_m, smp_siz, ([ll_m,ul_m]) ,(ul_m-ll_m),90]]), axis=0)
    array = np.append(array, np.array([['Unmarried', ll_u, ul_u, smp_siz, ([ll_u,ul_u]) ,(ul_u-ll_u),90]]), axis=

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

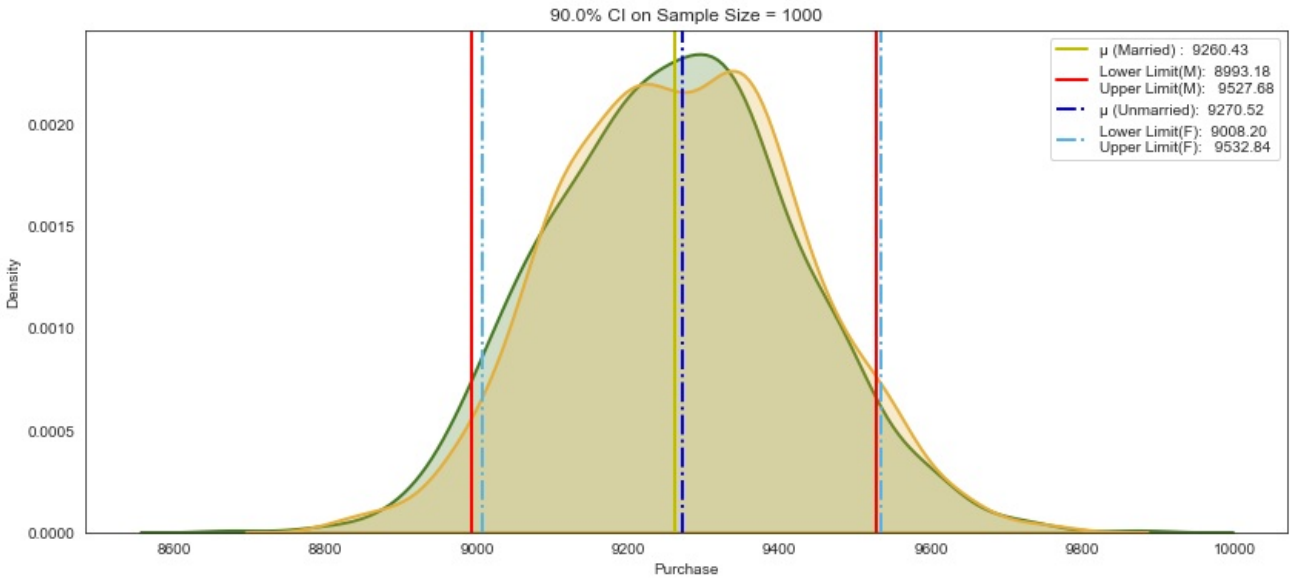


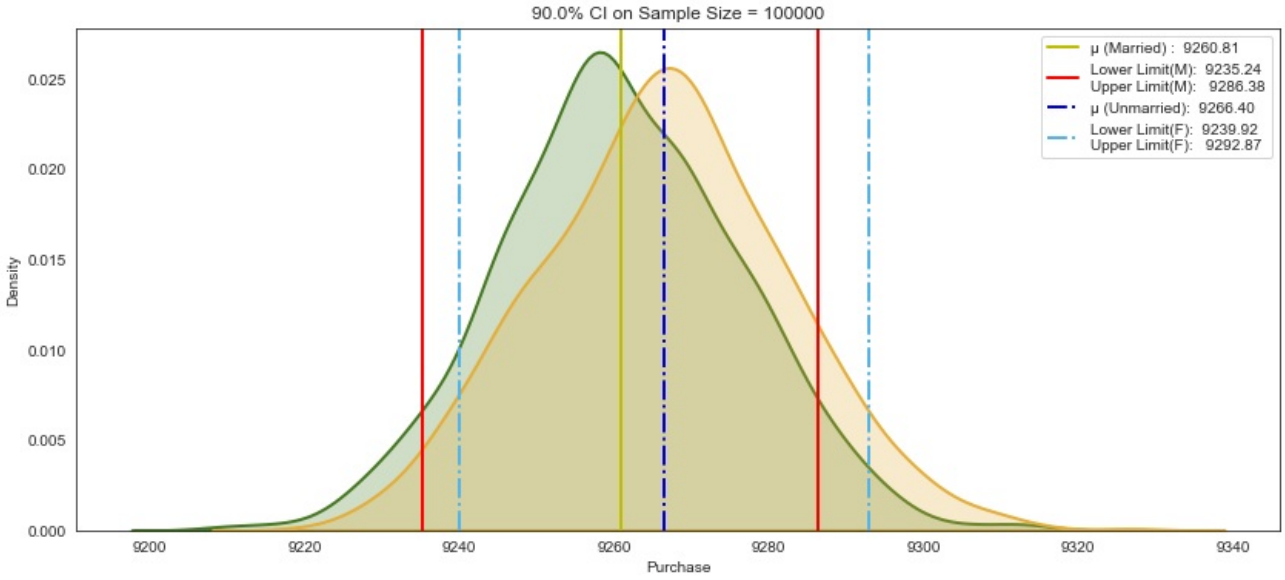

Lower Limit(F): 651.08
Upper Limit(F): 17814.18
Density
0.00008
0.00006
0.00004
0.00002
0.00000
−5000
0
5000
10000
15000
20000
25000
Purchase


90.0% CI on Sample Size = 10
μ (Married) : 9283.22
Lower Limit(M): 6675.27
Upper Limit(M): 11891.16
μ (Unmarried): 9368.88
Lower Limit(F): 6824.94
Upper Limit(F): 11912.82
Density
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
4000
6000
8000
10000
12000
14000
16000
Purchase


90.0% CI on Sample Size = 30
μ (Married) : 9265.53
Lower Limit(M): 7738.37
Upper Limit(M): 10792.69
μ (Unmarried): 9268.40
Lower Limit(F): 7772.91
Upper Limit(F): 10763.89
Density
0.0004
0.0003
0.0002
0.0001
0.0000
6000
7000
8000
9000
10000
11000
12000
13000
Purchase


90.0% CI on Sample Size = 300
μ (Married) : 9253.15
Lower Limit(M): 8778.30
Upper Limit(M): 9728.00
μ (Unmarried): 9263.90
Lower Limit(F): 8788.42
Upper Limit(F): 9739.38
Density
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002





In [59]:
```
overlap.loc[(overlap["Marital_Status"] == "Married") & (overlap["Sample_Size"] >= 300)]
```

Out[59]:

	Marital_Status	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
6	Married	8778.3	9728.0	300	[8778.3, 9728.0]	949.7	90
8	Married	8993.18	9527.68	1000	[8993.18, 9527.68]	534.5	90
10	Married	9235.24	9286.38	100000	[9235.24, 9286.38]	51.14	90

In [60]:
```
overlap.loc[(overlap["Marital_Status"] == "Unmarried") & (overlap["Sample_Size"] >= 300)]
```

Out[60]:

	Marital_Status	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
7	Unmarried	8788.42	9739.38	300	[8788.42, 9739.38]	950.96	90
9	Unmarried	9008.2	9532.84	1000	[9008.2, 9532.84]	524.64	90
11	Unmarried	9239.92	9292.87	100000	[9239.92, 9292.87]	52.95	90

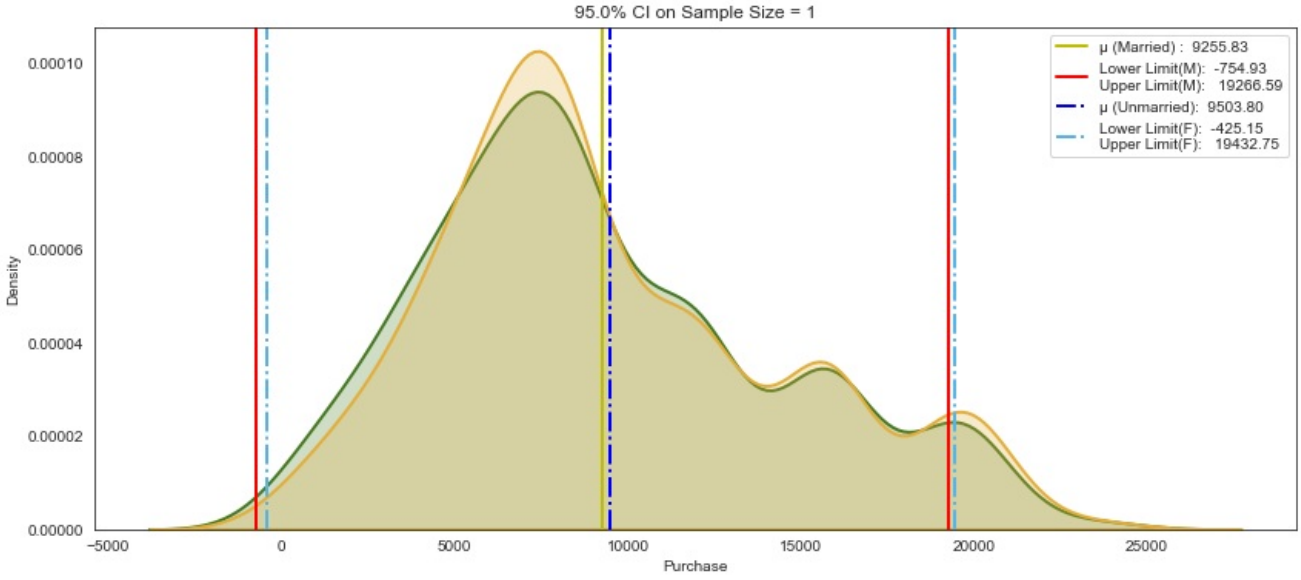
Insights

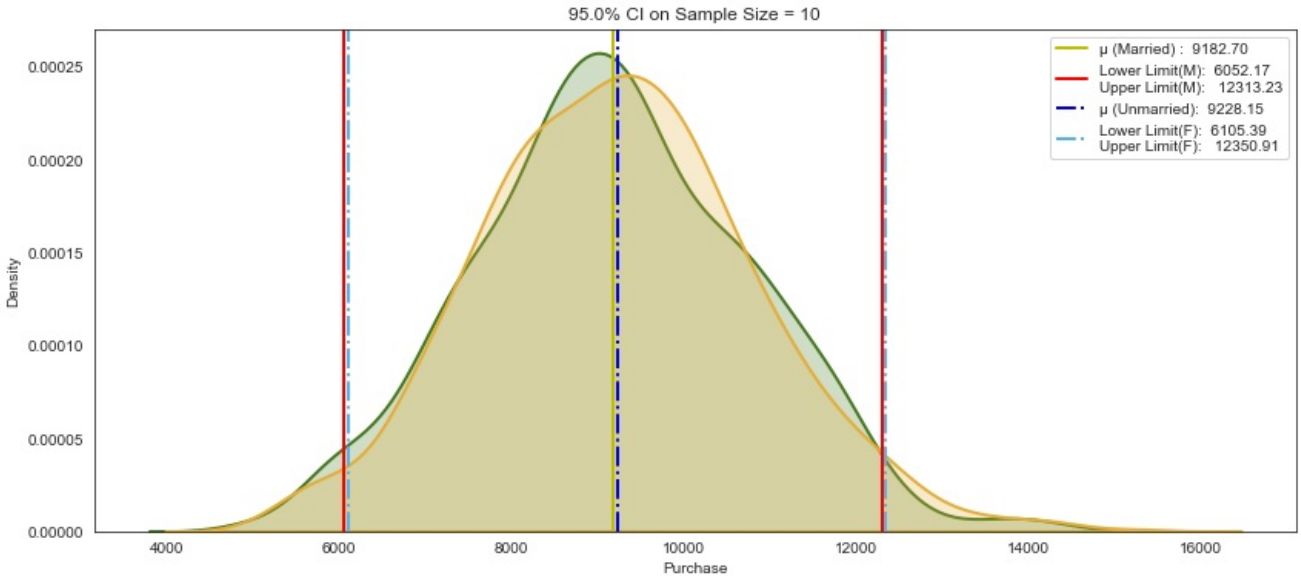
- For Unmarried customer (sample size 100000) range for mean purchase with confidence interval 90% is [9240.36, 9290.87]
- For married customer range for mean purchase with confidence interval 90% is [9233.54, 9288.07]

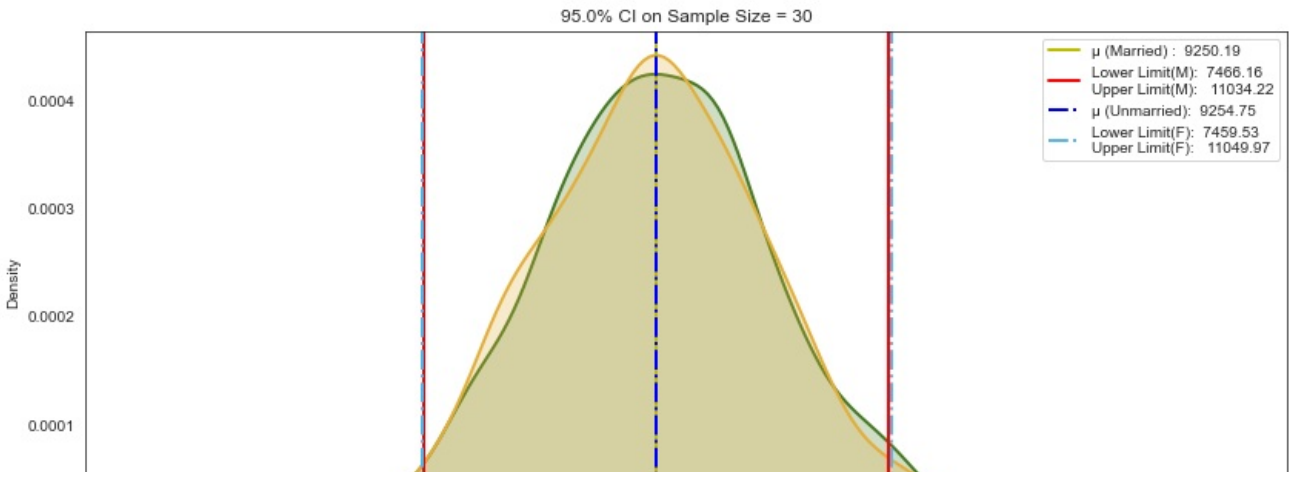
Calculating 95% CI

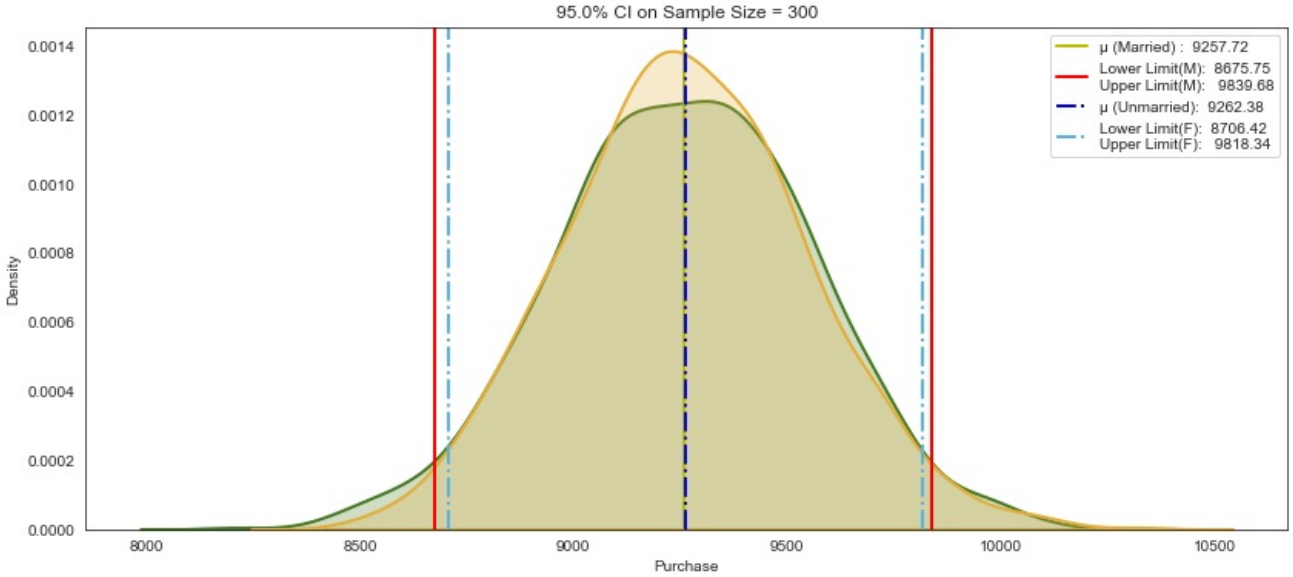
In [61]:

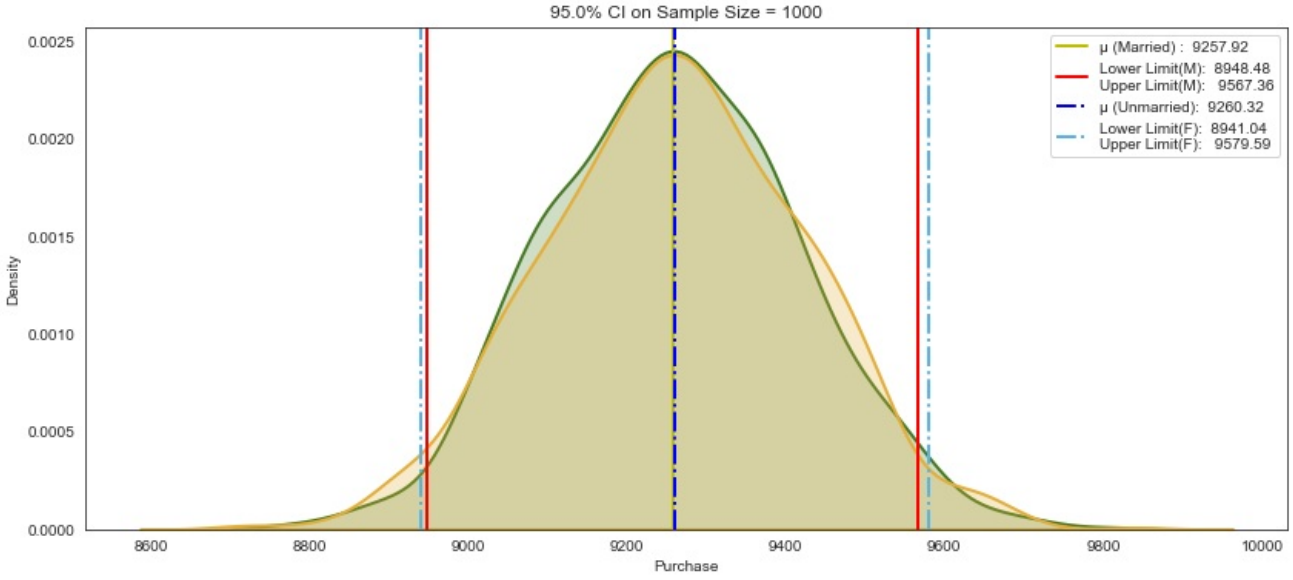
```
itr_size = 1000
size_list = [1, 10, 30, 300, 1000, 100000]
ci = 0.95

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

for smp_siz in size_list:
    m_avg, f_avg, ll_m, ul_m, ll_u, ul_u = bootstrapping_marital_status(walmart_new_smp_married,walmart_new_smp_u

    array = np.append(array, np.array([['Married', ll_m, ul_m, smp_siz, ([ll_m,ul_m]) ,(ul_m-ll_m),95]]), axis=0)
    array = np.append(array, np.array([['Unmarried', ll_u, ul_u, smp_siz, ([ll_u,ul_u]) ,(ul_u-ll_u),95]]), axis=

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

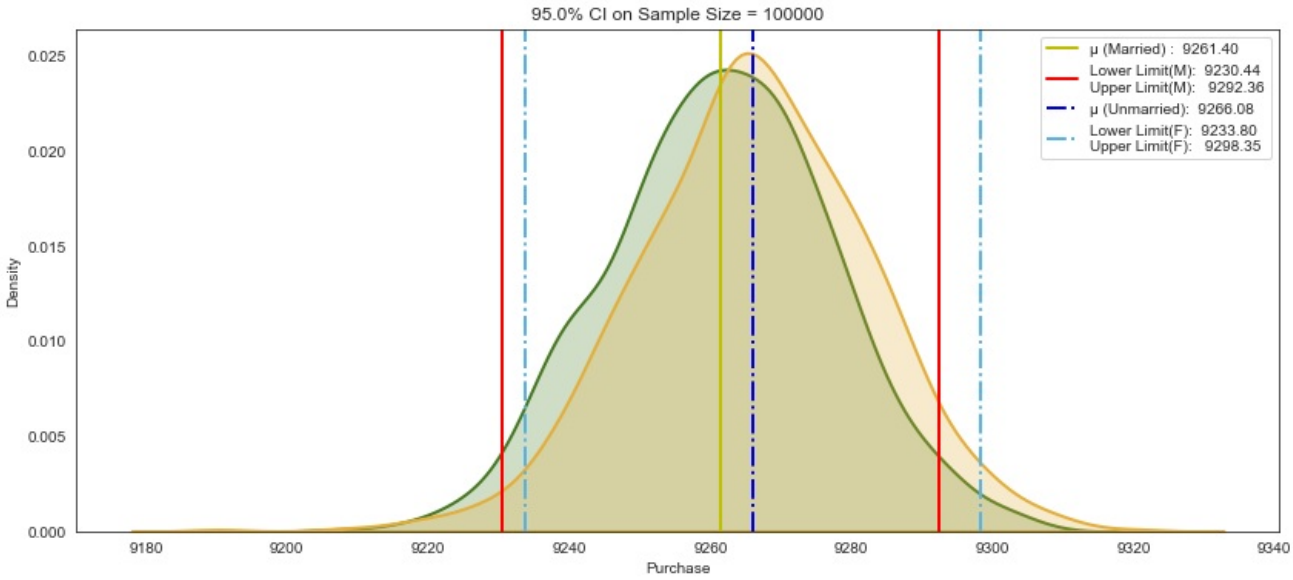












In [62]:
```
overlap.loc[(overlap["Marital_Status"] == "Married") & (overlap["Sample_Size"] >= 300)]
```

Out[62]:

	Marital_Status	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct

6	Married	8675.75	9839.68	300	[8675.75, 9839.68]	1163.93	95
8	Married	8948.48	9567.36	1000	[8948.48, 9567.36]	618.88	95
10	Married	9230.44	9292.36	100000	[9230.44, 9292.36]	61.92	95

In [63]:
```
overlap.loc[(overlap["Marital_Status"] == "Unmarried") & (overlap["Sample_Size"] >= 300)]
```

Out[63]:

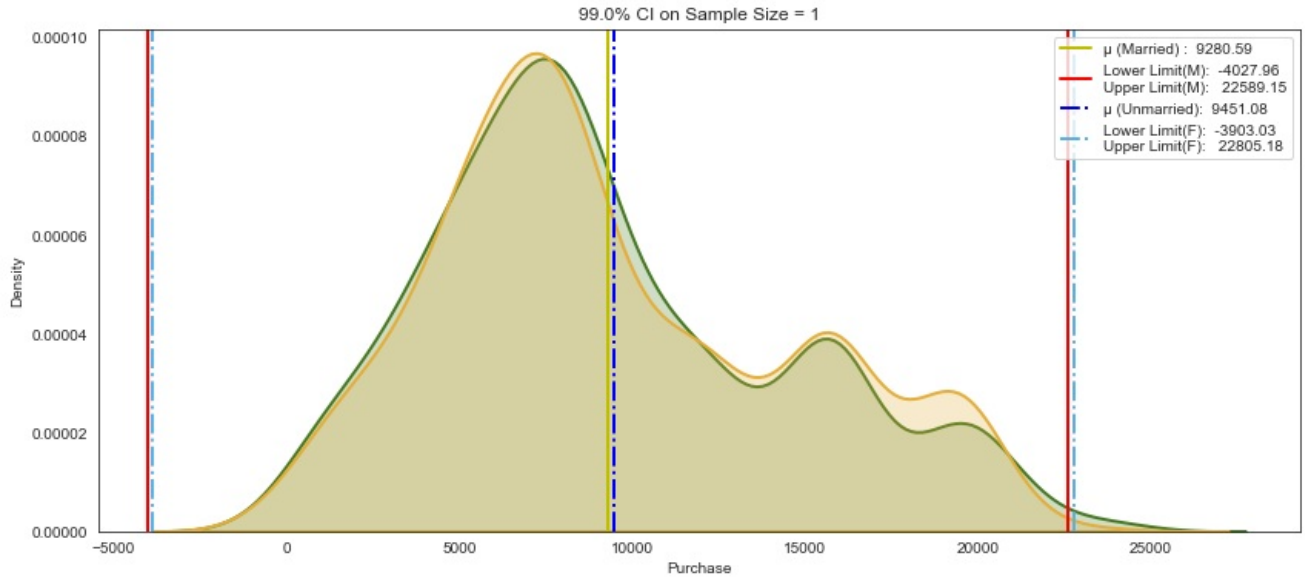
	Marital_Status	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
7	Unmarried	8706.42	9818.34	300	[8706.42, 9818.34]	1111.92	95
9	Unmarried	8941.04	9579.59	1000	[8941.04, 9579.59]	638.55	95
11	Unmarried	9233.8	9298.35	100000	[9233.8, 9298.35]	64.55	95

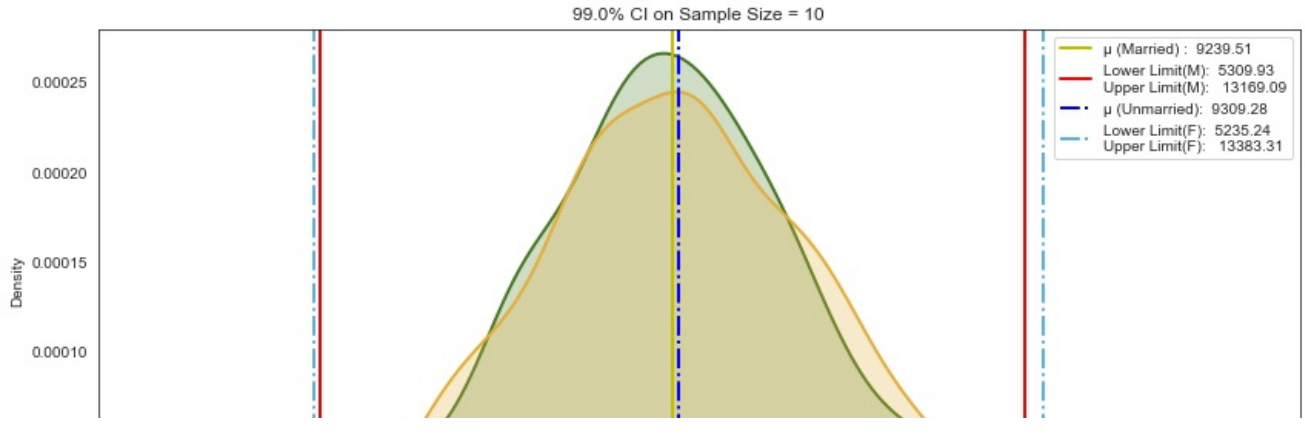
Insights

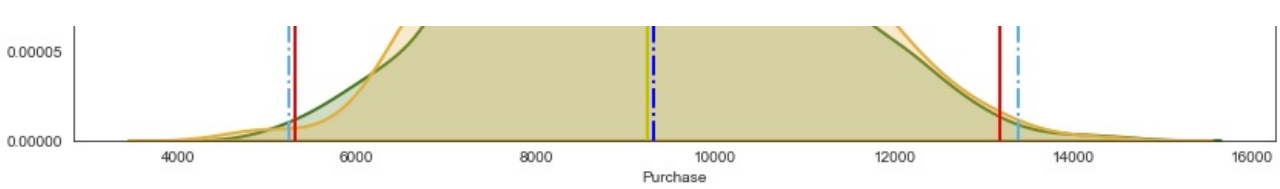
- For Unmarried customer (sample size 100000) range for mean purchase with confidence interval 95% is [9234.75, 9297.58]
- For married customer range for mean purchase with confidence interval 95% is [9230.34, 9293.32]

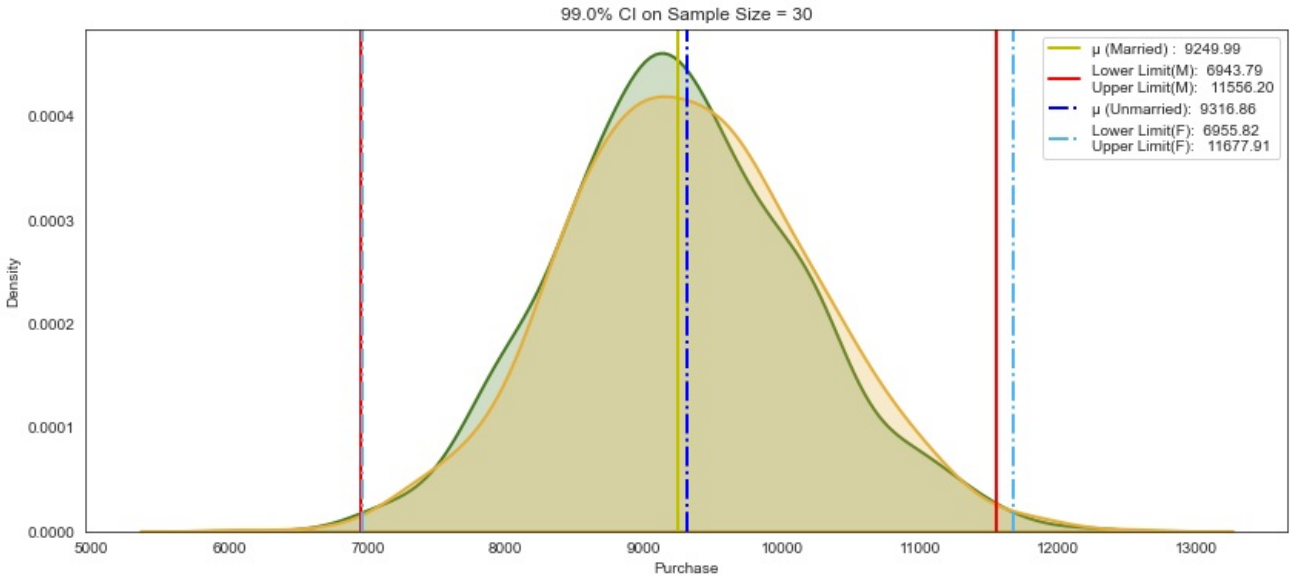
Calculating 99% CI

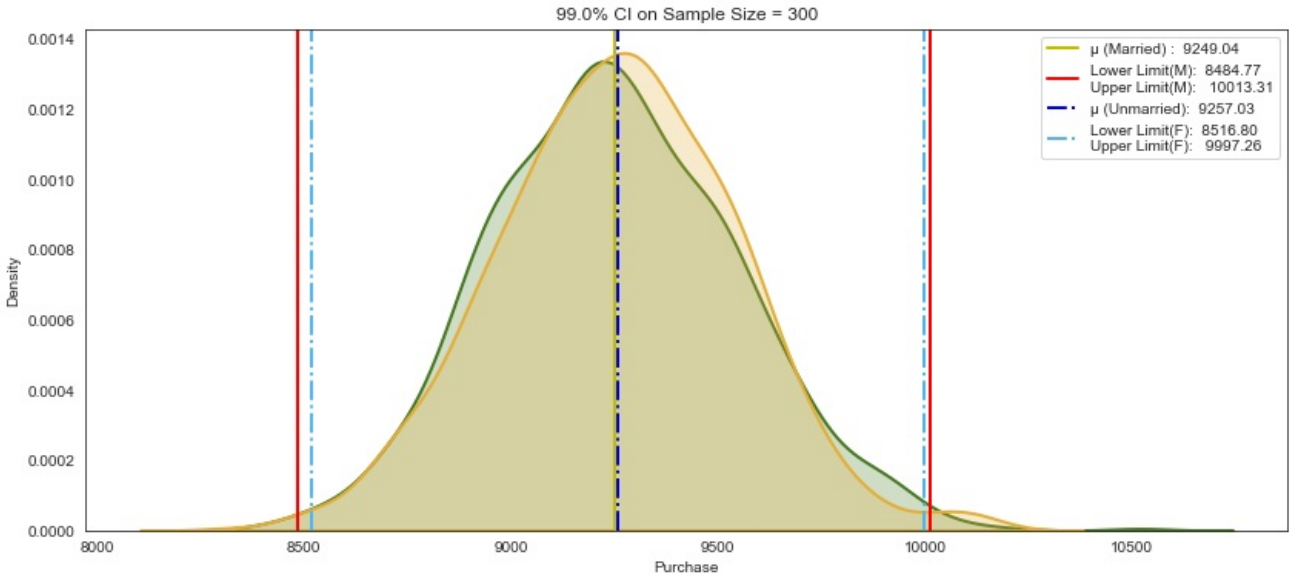
In [64]:
```
itr_size = 1000
size_list = [1, 10, 30, 300, 1000, 100000]
ci = 0.99

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

for smp_siz in size_list:
    m_avg, f_avg, ll_m, ul_m, ll_u, ul_u = bootstrapping_marital_status(walmart_new_smp_married,walmart_new_smp_u

    array = np.append(array, np.array([['Married', ll_m, ul_m, smp_siz, ([ll_m,ul_m]) ,(ul_m-ll_m),99]]), axis=0)
    array = np.append(array, np.array([['Unmarried', ll_u, ul_u, smp_siz, ([ll_u,ul_u]) ,(ul_u-ll_u),99]]), axis=

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

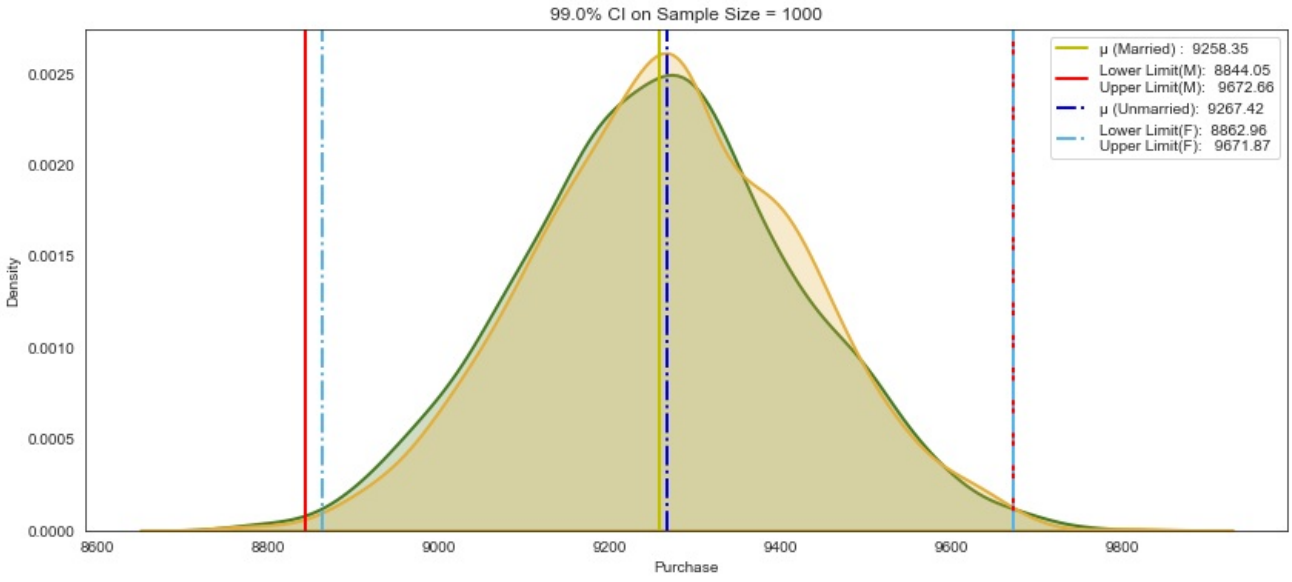


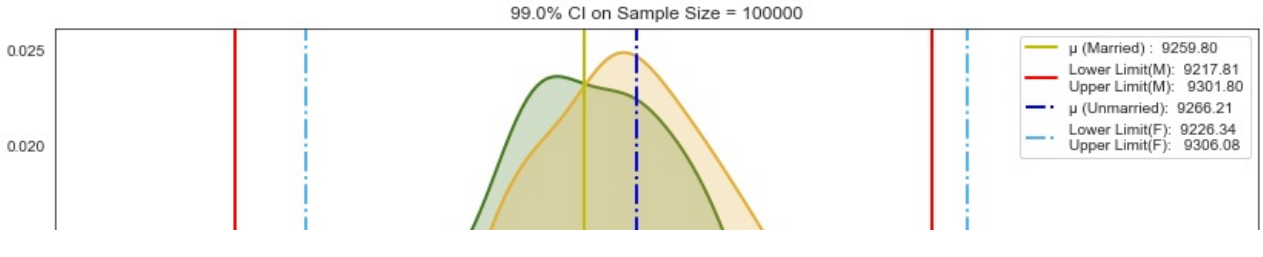


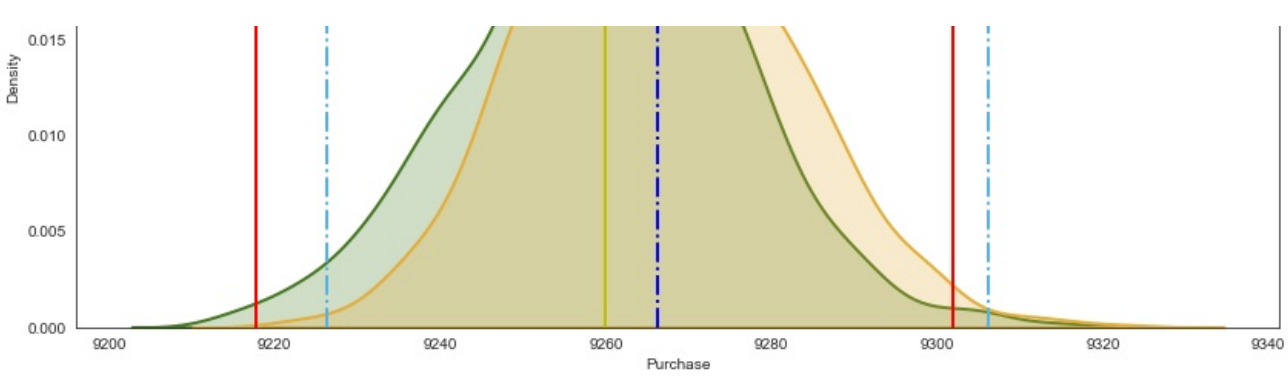

0.00005
0.00000
4000
6000
8000
10000
12000
14000
16000
Purchase


99.0% CI on Sample Size = 30
μ (Married) : 9249.99
Lower Limit(M): 6943.79
Upper Limit(M): 11556.20
μ (Unmarried): 9316.86
Lower Limit(F): 6955.82
Upper Limit(F): 11677.91
Density
0.0004
0.0003
0.0002
0.0001
0.0000
5000
6000
7000
8000
9000
10000
11000
12000
13000
Purchase


99.0% CI on Sample Size = 300
μ (Married) : 9249.04
Lower Limit(M): 8484.77
Upper Limit(M): 10013.31
μ (Unmarried): 9257.03
Lower Limit(F): 8516.80
Upper Limit(F): 9997.26
Density
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
8000
8500
9000
9500
10000
10500
Purchase


99.0% CI on Sample Size = 1000
μ (Married) : 9258.35
Lower Limit(M): 8844.05
Upper Limit(M): 9672.66
μ (Unmarried): 9267.42
Lower Limit(F): 8862.96
Upper Limit(F): 9671.87
Density
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
8600
8800
9000
9200
9400
9600
9800
Purchase


99.0% CI on Sample Size = 100000
μ (Married) : 9259.80
Lower Limit(M): 9217.81
Upper Limit(M): 9301.80
μ (Unmarried): 9266.21
Lower Limit(F): 9226.34
Upper Limit(F): 9306.08
0.025
0.020



In [65]:
```
overlap.loc[(overlap["Marital_Status"] == "Married") & (overlap["Sample_Size"] >= 300)]
```

Out[65]:

	Marital_Status	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
6	Married	8484.77	10013.31	300	[8484.77, 10013.31]	1528.54	99
8	Married	8844.05	9672.66	1000	[8844.05, 9672.66]	828.61	99
10	Married	9217.81	9301.8	100000	[9217.81, 9301.8]	83.99	99

In [66]:
```
overlap.loc[(overlap["Marital_Status"] == "Unmarried") & (overlap["Sample_Size"] >= 300)]
```

Out[66]:

	Marital_Status	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
7	Unmarried	8516.8	9997.26	300	[8516.8, 9997.26]	1480.46	99
9	Unmarried	8862.96	9671.87	1000	[8862.96, 9671.87]	808.91	99
11	Unmarried	9226.34	9306.08	100000	[9226.34, 9306.08]	79.74	99

Insights

- For Unmarried customer (sample size 100000) range for mean purchase with confidence interval 99% is [9225.71, 9305.43]
- For married customer range for mean purchase with confidence interval 99% is [9218.62, 9303.25]

Major Inferences

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

4.5 Results when the same activity is performed for Age

In [67]:
```
age_data = walmart_new.groupby(["Age"])["Purchase"].describe()
```

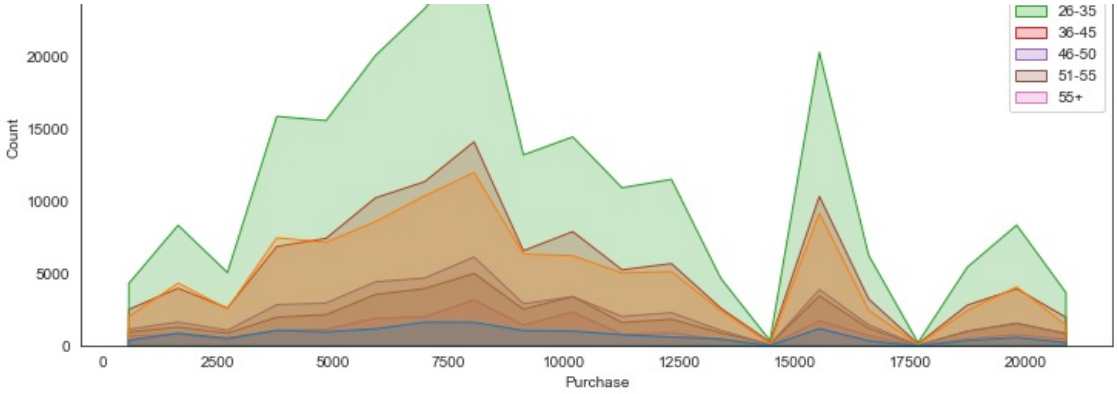
In [68]:
```
age_dict = {}
age_list = ['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

# Print the columns of age_data for debugging
print("Columns in age_data:", age_data.columns)

# Check that age_data has enough rows
num_ages = min(len(age_list), age_data.shape[0])

for i in range(num_ages):
    # Debug: Print the row being accessed
    print(f"Accessing row {i}: {age_data.iloc[i]}")

    # Adjust the column name if necessary
```

```
    try:
        age_dict[age_list[i]] = age_data.iloc[i]["Mean"]  # Adjust if column name differs
    except KeyError as e:
        print(f"KeyError: {e}. Check if the column name is correct.")
```

```
Columns in age_data: Index(['count', 'mean', 'std', 'min', '25%', '50%', '75%', 'max'], dtype='object')
Accessing row 0: count    15102.000000
mean      8925.539597
std       5090.078414
min         12.000000
25%       5328.000000
50%       7986.000000
75%      11874.000000
max      21400.500000
Name: 0-17, dtype: float64
KeyError: 'Mean'. Check if the column name is correct.
Accessing row 1: count    99660.000000
mean      9164.189554
std       5019.866472
min         12.000000
25%       5415.000000
50%       8027.000000
75%      12028.000000
max      21400.500000
Name: 18-25, dtype: float64
KeyError: 'Mean'. Check if the column name is correct.
Accessing row 2: count    219587.000000
mean       9244.947060
std        4990.086124
min          12.000000
25%        5475.000000
50%        8030.000000
75%       12047.000000
max       21400.500000
Name: 26-35, dtype: float64
KeyError: 'Mean'. Check if the column name is correct.
Accessing row 3: count    110013.000000
mean       9320.888550
std        4995.487782
min          12.000000
25%        5876.000000
50%        8061.000000
75%       12107.000000
max       21400.500000
Name: 36-45, dtype: float64
KeyError: 'Mean'. Check if the column name is correct.
Accessing row 4: count    45701.000000
mean      9198.531093
std       4940.249496
min         12.000000
25%       5888.000000
50%       8036.000000
75%      11997.000000
max      21400.500000
Name: 46-50, dtype: float64
KeyError: 'Mean'. Check if the column name is correct.
Accessing row 5: count    38501.000000
mean      9519.560427
std       5048.447704
min         12.000000
25%       6017.000000
50%       8130.000000
75%      12462.000000
max      21400.500000
Name: 51-55, dtype: float64
KeyError: 'Mean'. Check if the column name is correct.
Accessing row 6: count    21504.000000
mean      9319.768741
std       4968.014656
min         12.000000
25%       6018.000000
50%       8105.500000
75%      11932.000000
max      21400.500000
Name: 55+, dtype: float64
KeyError: 'Mean'. Check if the column name is correct.
```

In [69]:

```
plt.figure(figsize = (12, 5))

sns.histplot(data = walmart_new, x = "Purchase", bins = 20, hue = "Age", element="poly")
plt.show()
```



Calculate CI (90%, 95%, 99%) for purchase based on age using CLT

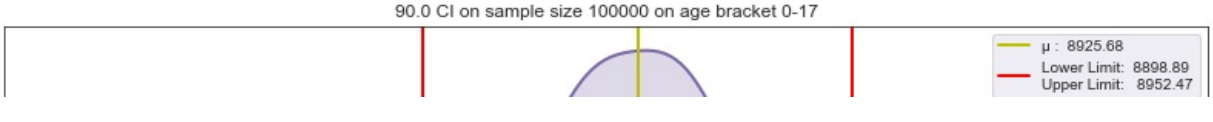
In [70]:

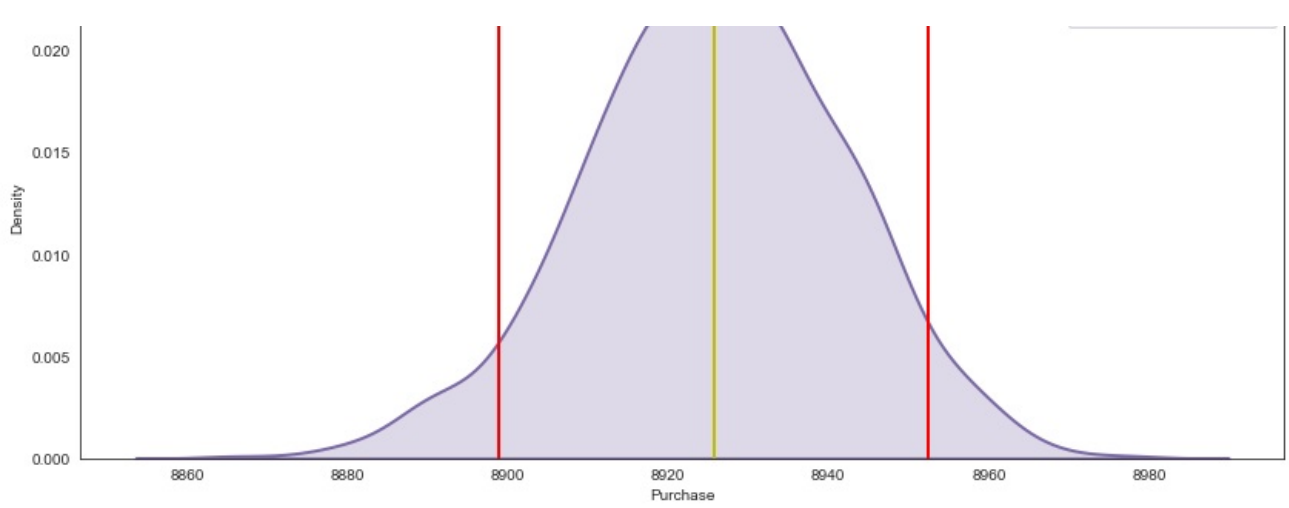
```
def bootstrapping_age(sample,smp_siz=500,itr_size=5000,confidence_level=0.95,age= "0-17", no_of_tails=2):

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

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

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

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

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

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

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

    plt.title(f"{confidence_level * 100} CI on sample size {smp_siz} on age bracket {age}")
    plt.xlabel('Purchase')
    plt.axvline(mean, color = 'y', linestyle = 'solid', linewidth = 2,label=label_mean)
    plt.axvline(upper_limit, color = 'r', linestyle = 'solid', linewidth = 2,label=label_ult)
    plt.axvline(lower_limit, color = 'r', linestyle = 'solid', linewidth = 2)
    plt.legend(loc='upper right')

    plt.show()

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

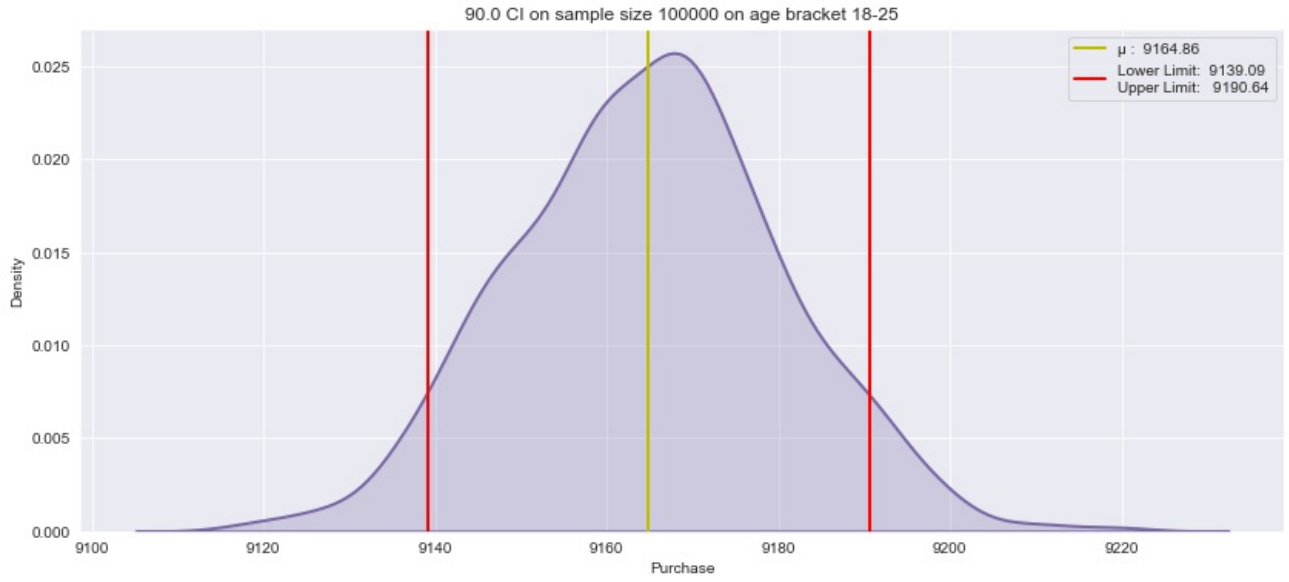
Calculating 90% CI

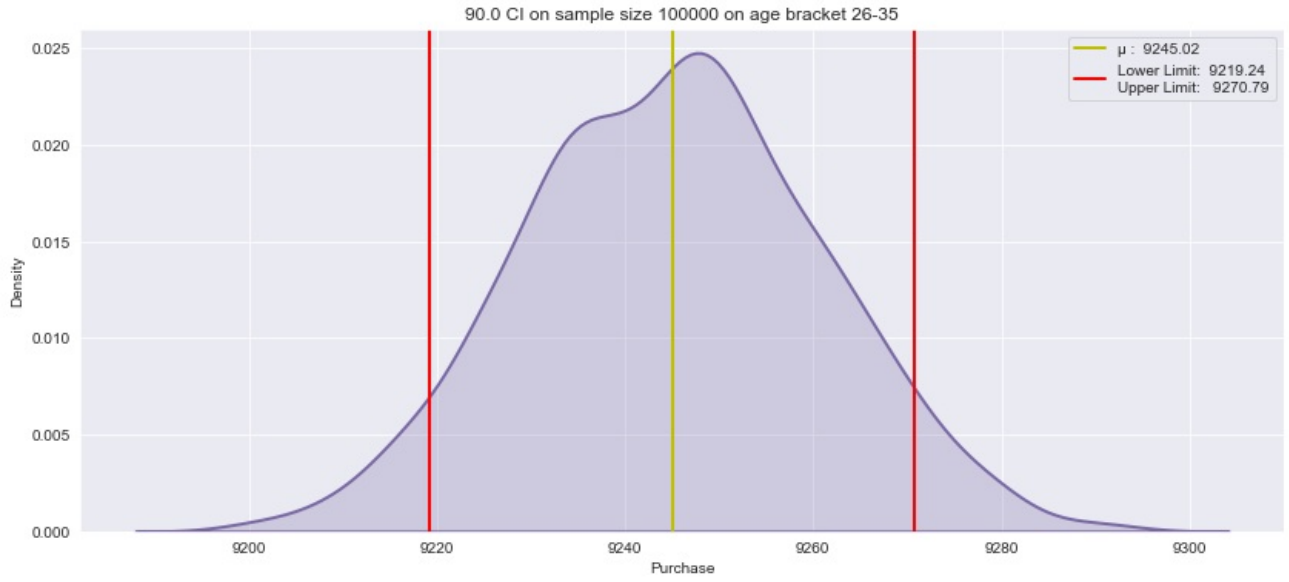
In [71]:

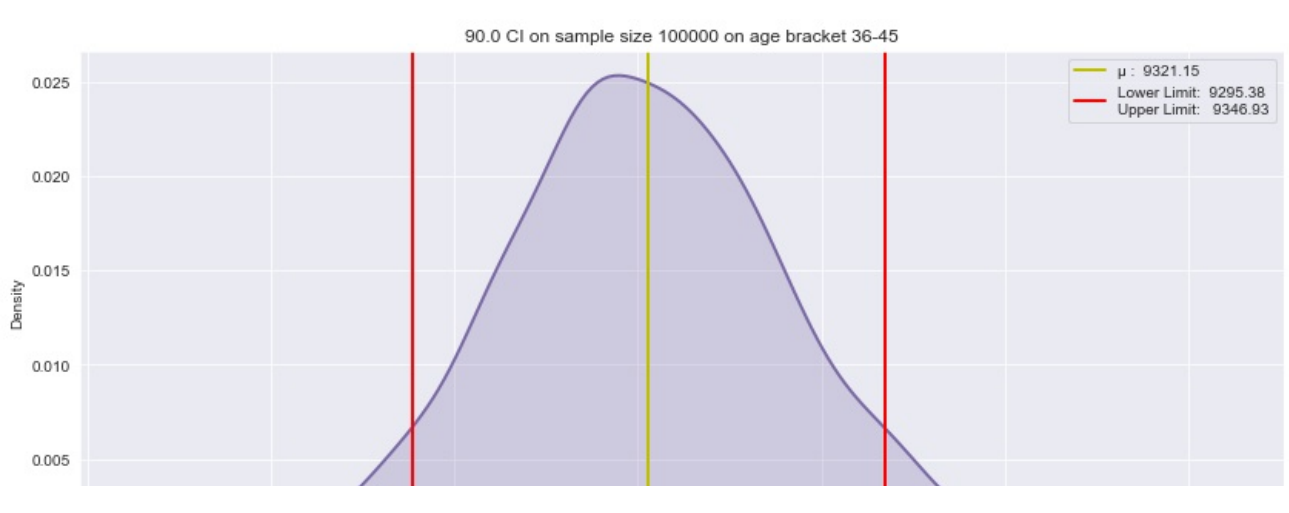
```
itr_size = 1000
smp_size = 10000
ci = 0.90
age_list =['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

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

for age in age_list:
    mean, ll_m, ul_m = bootstrapping_age(walmart_new[walmart_new['Age'] == age]['Purchase'],smp_siz,itr_size,ci,

    array = np.append(array, np.array([[age,np.round(mean,2), ll_m, ul_m, smp_siz, ([ll_m,ul_m]) ,(ul_m-ll_m),90]

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

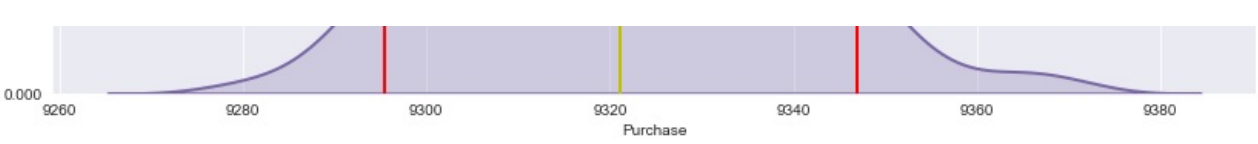


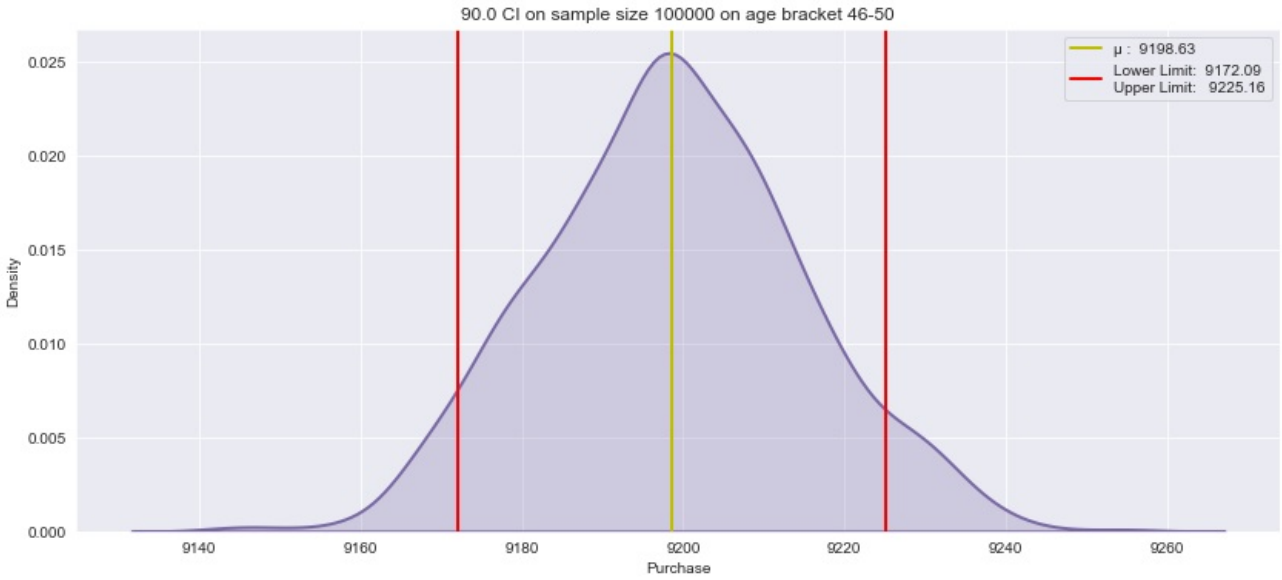

0.020
0.015
0.010
0.005
0.000
Density
8860
8880
8900
8920
8940
8960
8980
Purchase

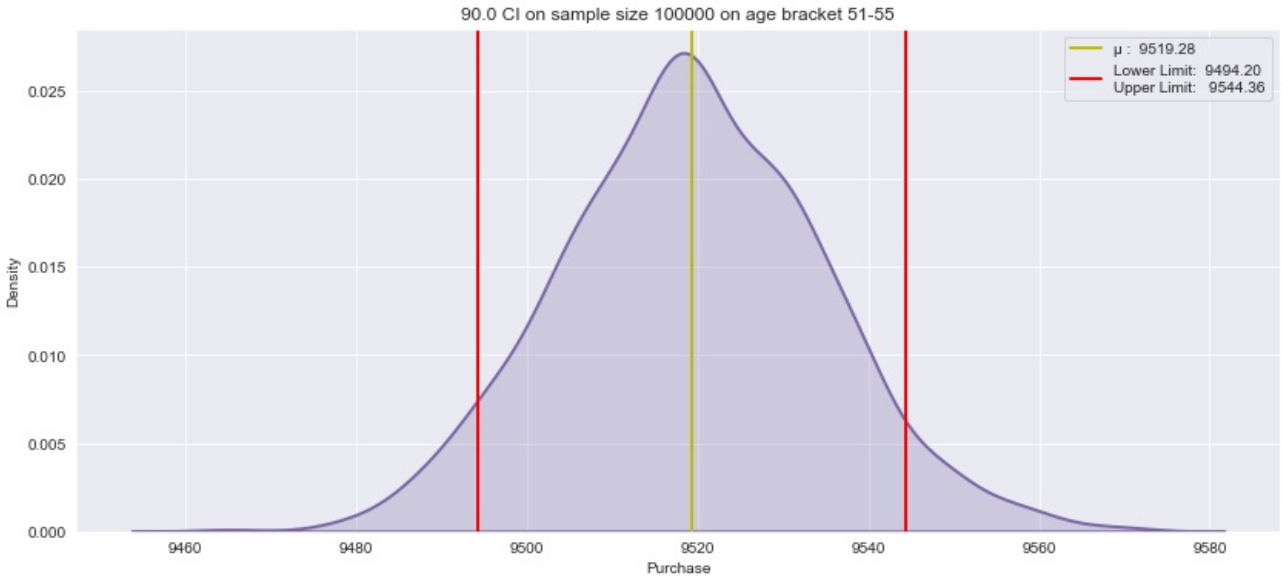

90.0 CI on sample size 100000 on age bracket 18-25
μ : 9164.86
Lower Limit: 9139.09
Upper Limit: 9190.64
0.025
0.020
0.015
0.010
0.005
0.000
Density
9100
9120
9140
9160
9180
9200
9220
Purchase

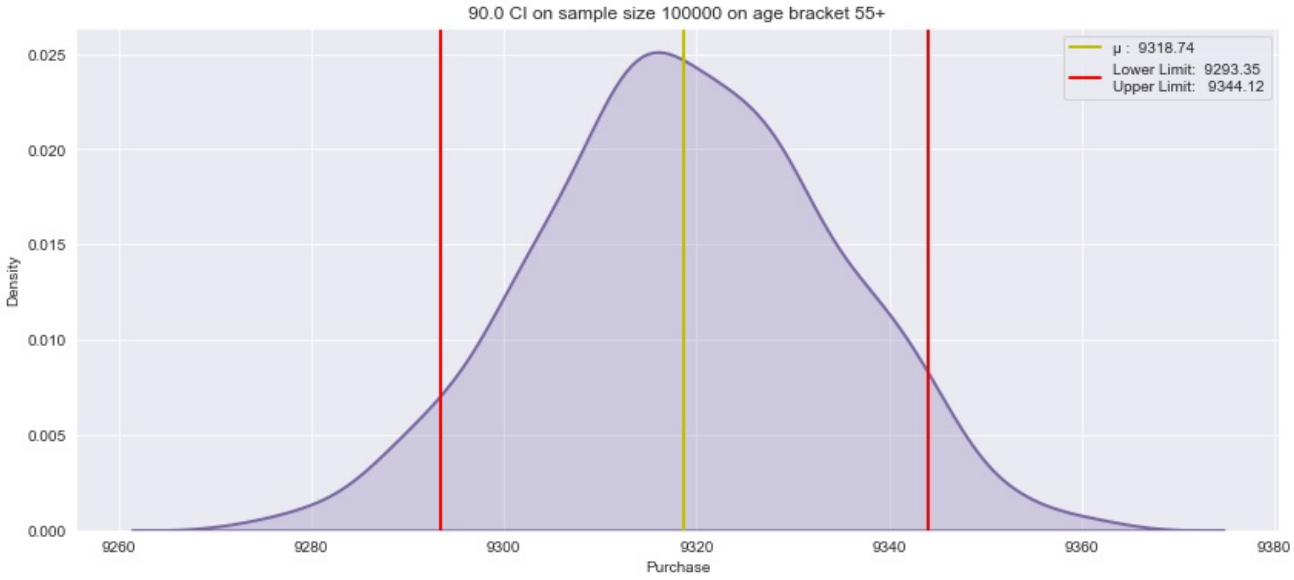

90.0 CI on sample size 100000 on age bracket 26-35
μ : 9245.02
Lower Limit: 9219.24
Upper Limit: 9270.79
0.025
0.020
0.015
0.010
0.005
0.000
Density
9200
9220
9240
9260
9280
9300
Purchase


90.0 CI on sample size 100000 on age bracket 36-45
μ : 9321.15
Lower Limit: 9295.38
Upper Limit: 9346.93
0.025
0.020
0.015
0.010
0.005
Density





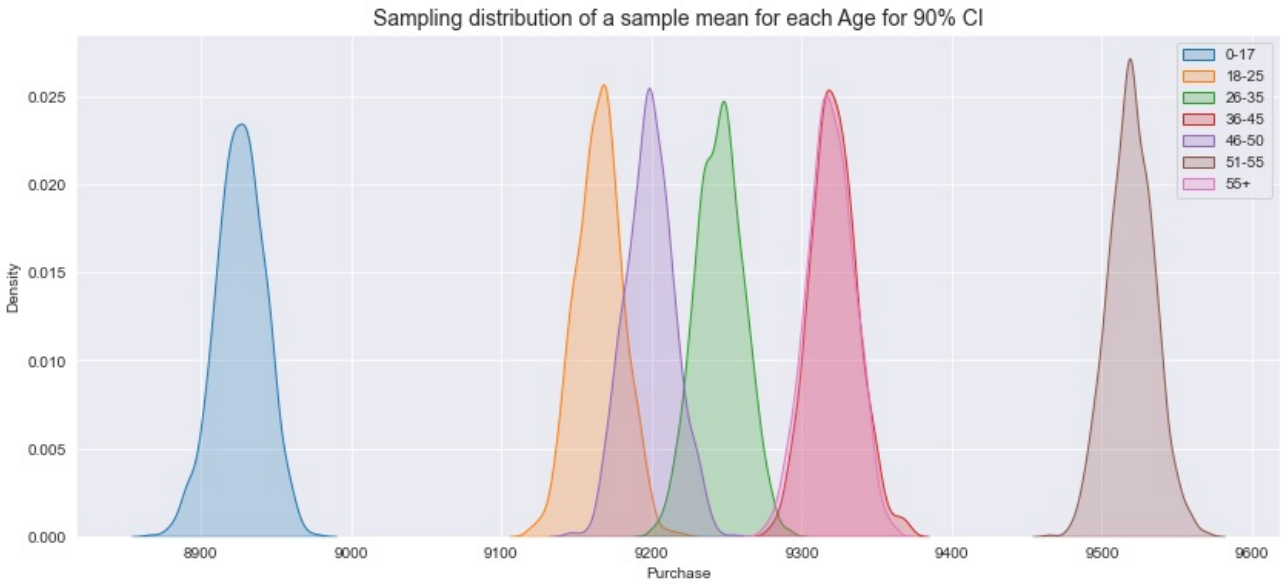




Checking the Sampling distribution of a sample mean for each Age Group for 90% CI

In [72]:

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

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

plt.title("Sampling distribution of a sample mean for each Age for 90% CI",fontsize=14)
plt.xlabel('Purchase')
plt.legend(loc='upper right')

plt.show()
```



In [73]:

```
age_data.head(10)
```

Out[73]:

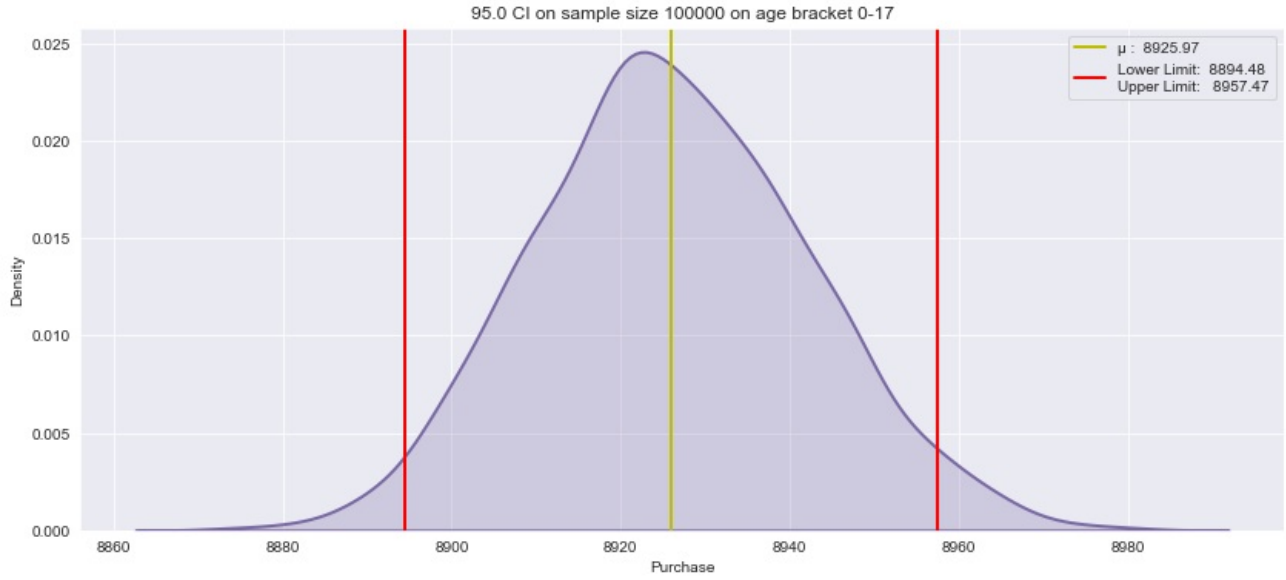
	Age_Group	Mean	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
0	0-17	[8941.35, 8944.93, 8924.8, 8940.68, 8926.98, 8...	8898.89	8952.47	100000	[8898.89, 8952.47]	53.58	90
1	18-25	[9158.29, 9163.78, 9148.03, 9145.51, 9187.84, ...	9139.09	9190.64	100000	[9139.09, 9190.64]	51.55	90
2	26-35	[9217.29, 9246.88, 9234.6, 9237.24, 9257.65, 9...	9219.24	9270.79	100000	[9219.24, 9270.79]	51.55	90
3	36-45	[9311.72, 9277.78, 9313.8, 9321.94, 9300.1, 93...	9295.38	9346.93	100000	[9295.38, 9346.93]	51.55	90
4	46-50	[9178.59, 9211.77, 9165.91, 9178.5, 9206.45, 9...	9172.09	9225.16	100000	[9172.09, 9225.16]	53.07	90
5	51-55	[9509.98, 9510.0, 9506.27, 9518.82, 9513.11, 9...	9494.2	9544.36	100000	[9494.2, 9544.36]	50.16	90
6	55+	[9320.55, 9315.74, 9352.57, 9337.68, 9312.58, ...	9293.35	9344.12	100000	[9293.35, 9344.12]	50.77	90

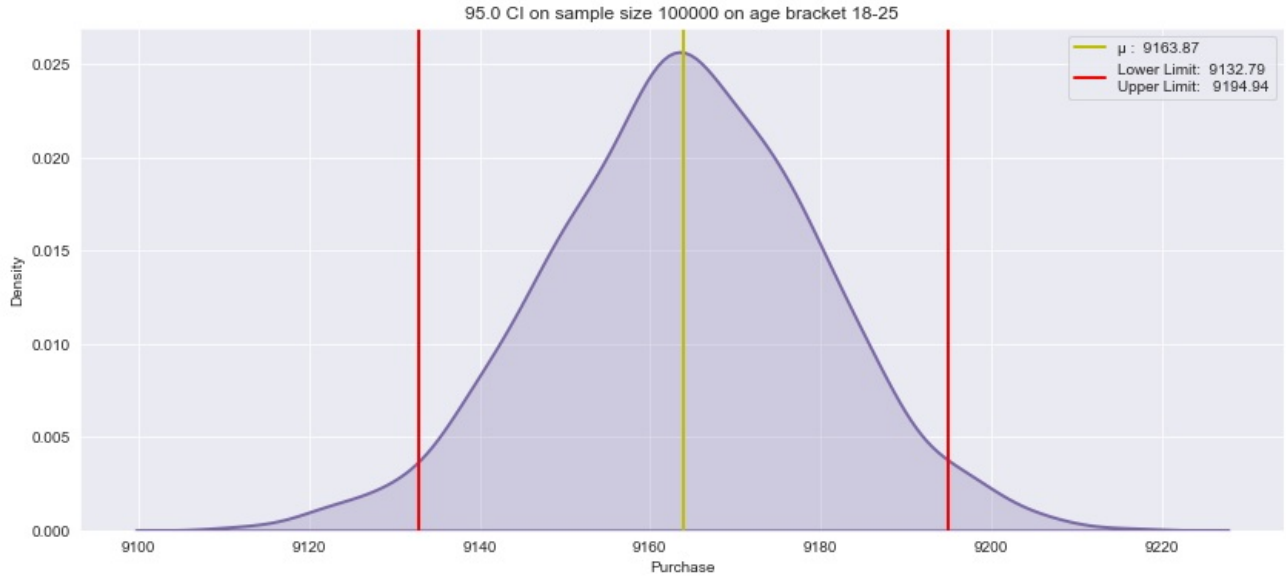
Calculating 95% CI

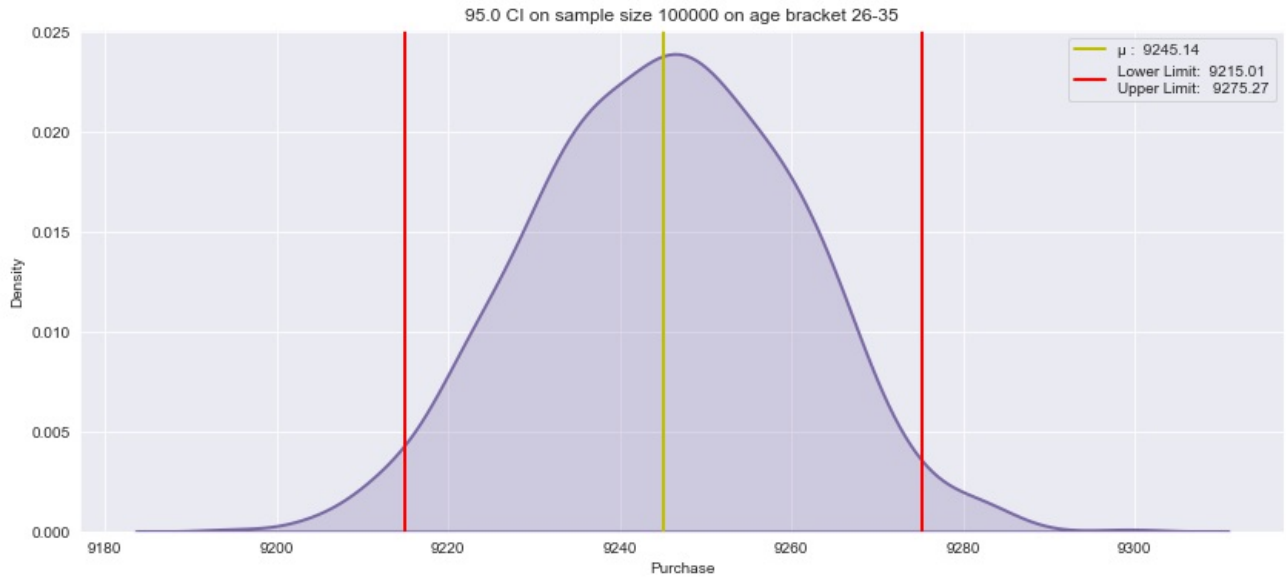
In [74]:

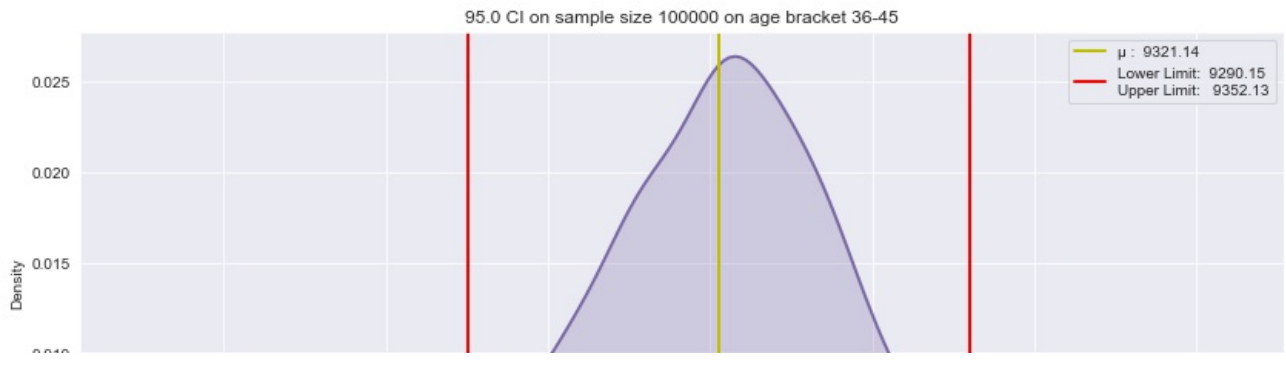
```
itr_size = 1000
smp_size = 10000
ci = 0.95
age_list =['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

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

for age in age_list:
    mean, ll_m, ul_m = bootstrapping_age(walmart_new[walmart_new['Age'] == age]['Purchase'],smp_siz,itr_size,ci,

    array = np.append(array, np.array([[age,np.round(mean,2), ll_m, ul_m, smp_siz, ([ll_m,ul_m]) ,(ul_m-ll_m),95]
```

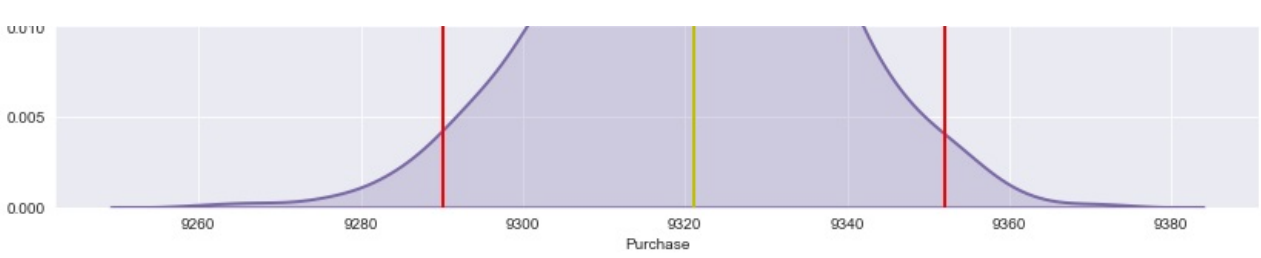
```
age_data = pd.DataFrame(array, columns = ['Age_Group','Mean','Lower_limit','Upper_limit','Sample_Size','CI','Rang
```

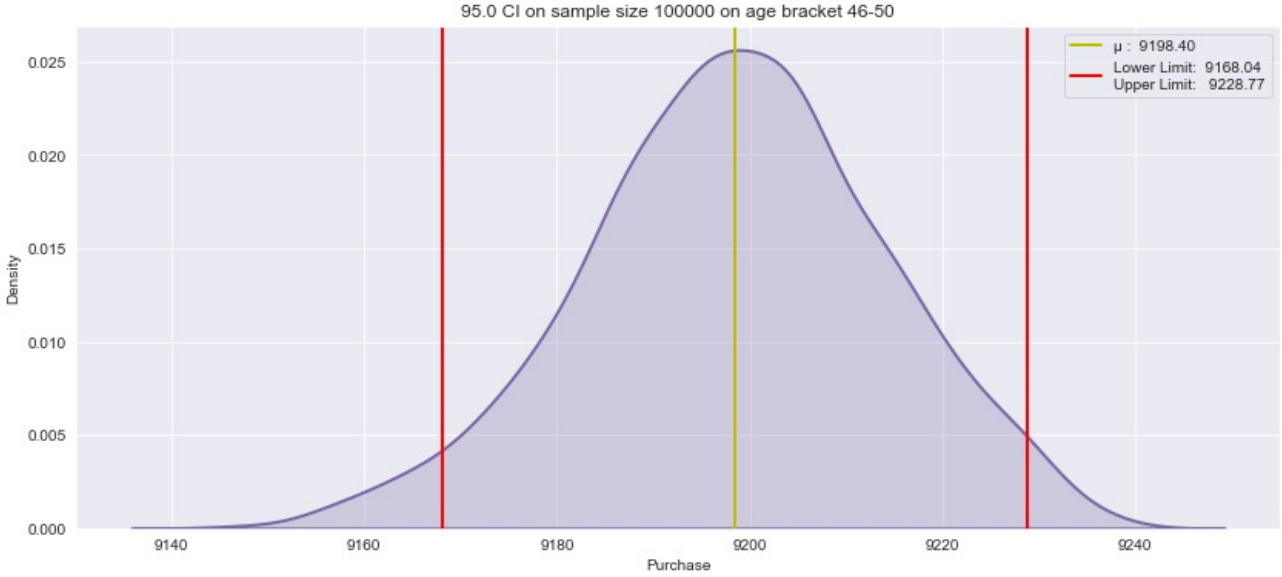


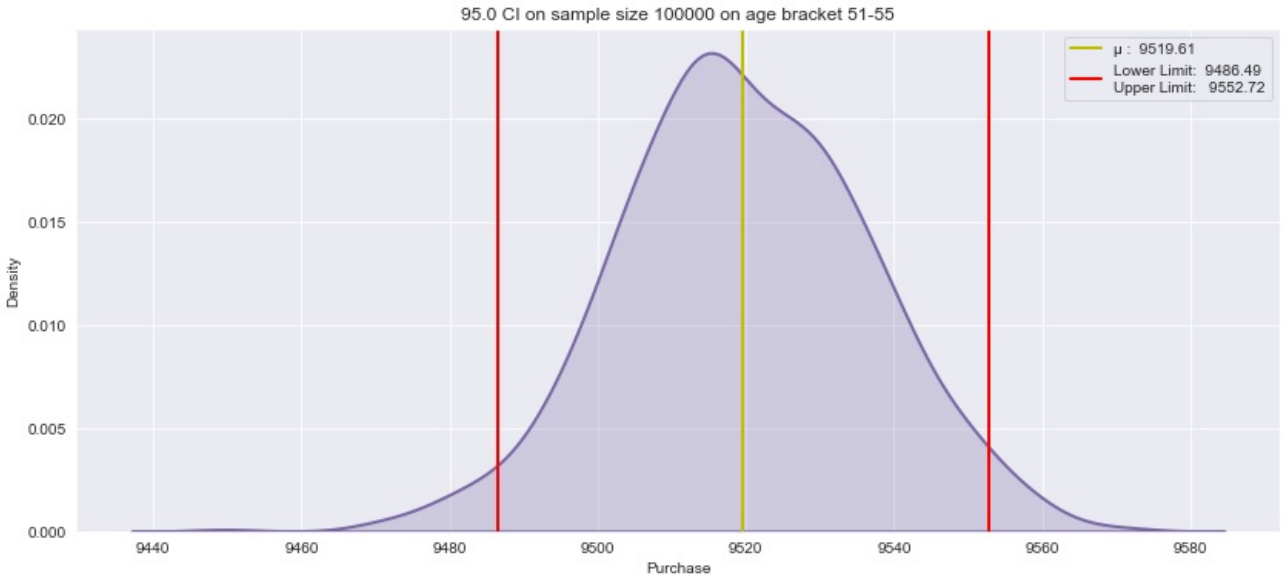


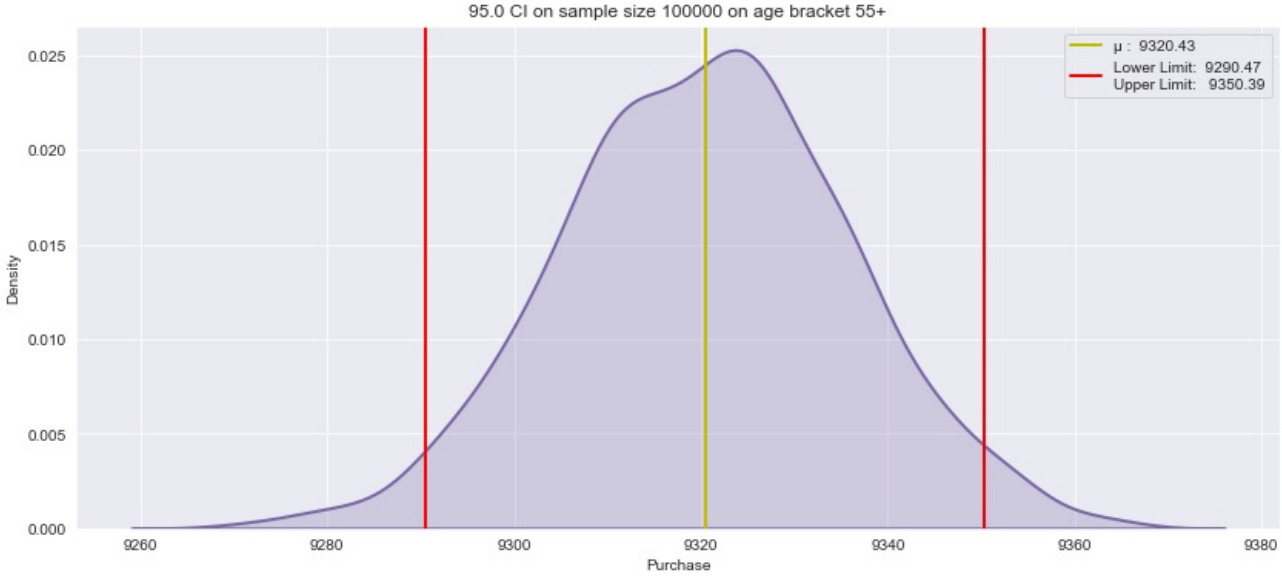





0.010
0.005
0.000
9260
9280
9300
9320
9340
9360
9380
Purchase


95.0 CI on sample size 100000 on age bracket 46-50
μ : 9198.40
Lower Limit: 9168.04
Upper Limit: 9228.77
Density
0.025
0.020
0.015
0.010
0.005
0.000
9140
9160
9180
9200
9220
9240
Purchase

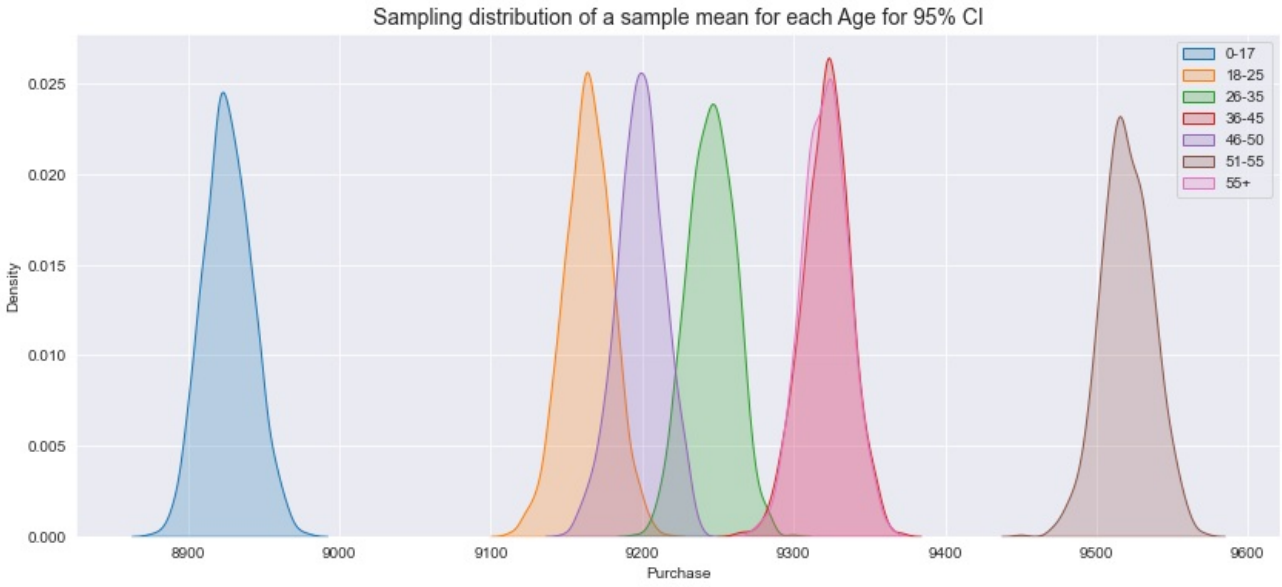

95.0 CI on sample size 100000 on age bracket 51-55
μ : 9519.61
Lower Limit: 9486.49
Upper Limit: 9552.72
Density
0.020
0.015
0.010
0.005
0.000
9440
9460
9480
9500
9520
9540
9560
9580
Purchase


95.0 CI on sample size 100000 on age bracket 55+
μ : 9320.43
Lower Limit: 9290.47
Upper Limit: 9350.39
Density
0.025
0.020
0.015
0.010
0.005
0.000
9260
9280
9300
9320
9340
9360
9380
Purchase

Checking the Sampling distribution of a sample mean for each Age Group for 95% CI

In [75]:

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

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

plt.title("Sampling distribution of a sample mean for each Age for 95% CI",fontsize=14)
plt.xlabel('Purchase')
plt.legend(loc='upper right')

plt.show()
```



In [76]:

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

Out[76]:

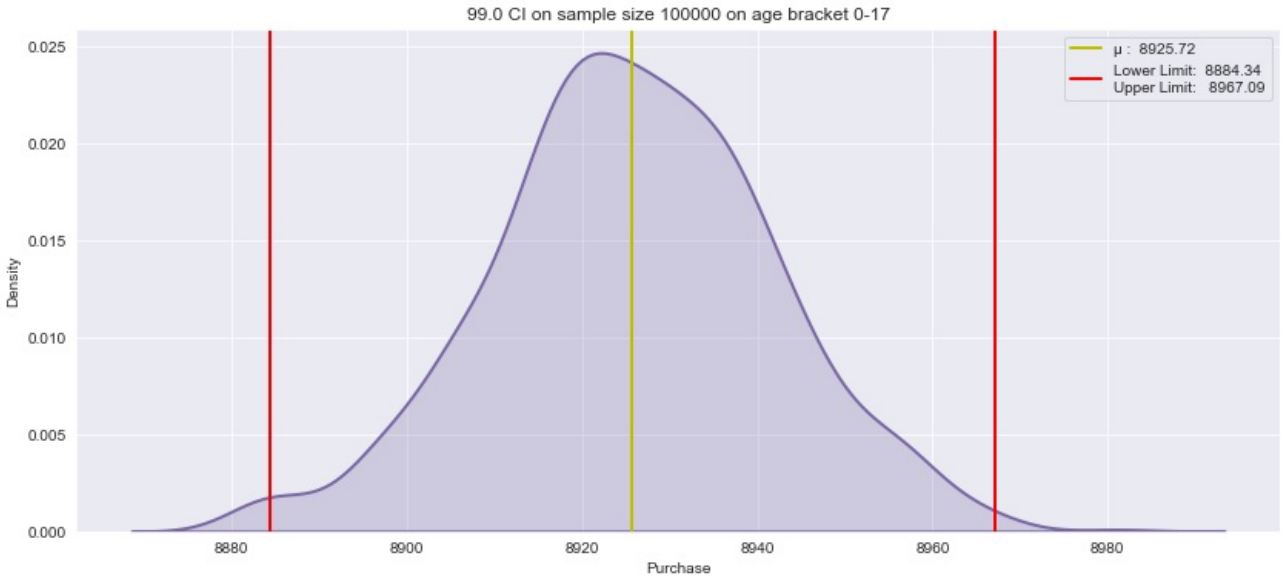
	Age_Group	Mean	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
0	0-17	[8947.89, 8942.76, 8925.93, 8915.89, 8912.76, ...	8894.48	8957.47	100000	[8894.48, 8957.47]	62.99	95
1	18-25	[9183.68, 9189.49, 9155.77, 9186.09, 9179.09, ...	9132.79	9194.94	100000	[9132.79, 9194.94]	62.15	95
2	26-35	[9252.98, 9218.91, 9234.28, 9235.98, 9258.41, ...	9215.01	9275.27	100000	[9215.01, 9275.27]	60.26	95
3	36-45	[9323.51, 9311.93, 9322.01, 9325.35, 9303.73, ...	9290.15	9352.13	100000	[9290.15, 9352.13]	61.98	95
4	46-50	[9206.01, 9221.36, 9172.65, 9196.1, 9214.81, 9...	9168.04	9228.77	100000	[9168.04, 9228.77]	60.73	95
5	51-55	[9540.11, 9535.16, 9553.96, 9540.37, 9514.08, ...	9486.49	9552.72	100000	[9486.49, 9552.72]	66.23	95
6	55+	[9321.24, 9302.96, 9324.06, 9312.92, 9320.13, ...	9290.47	9350.39	100000	[9290.47, 9350.39]	59.92	95

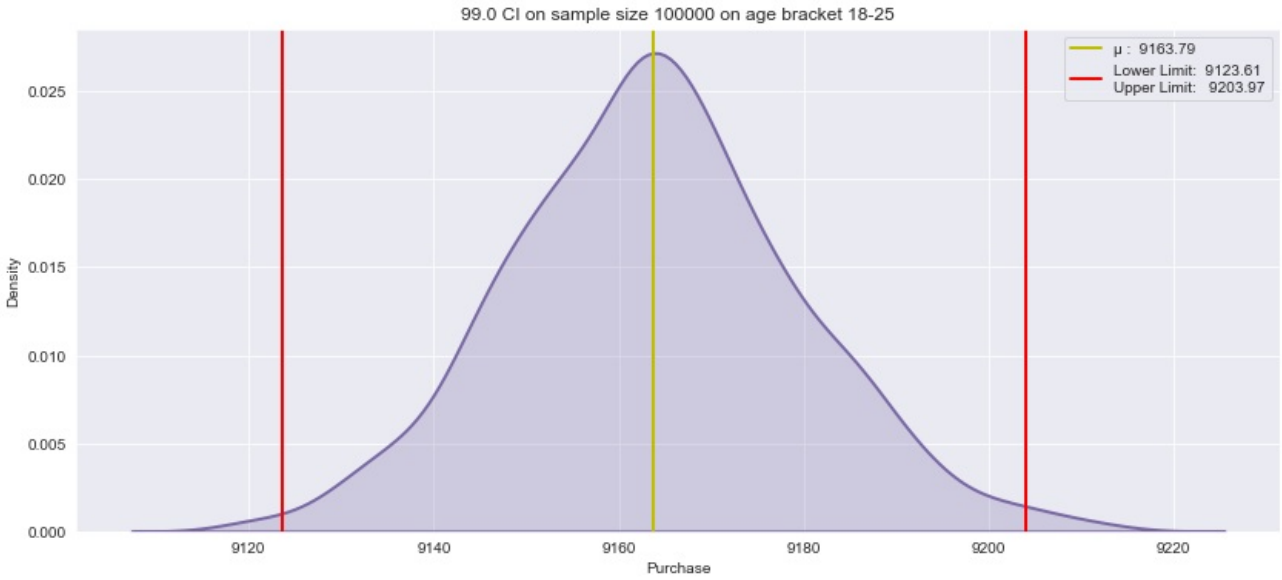
Calculating 99% CI

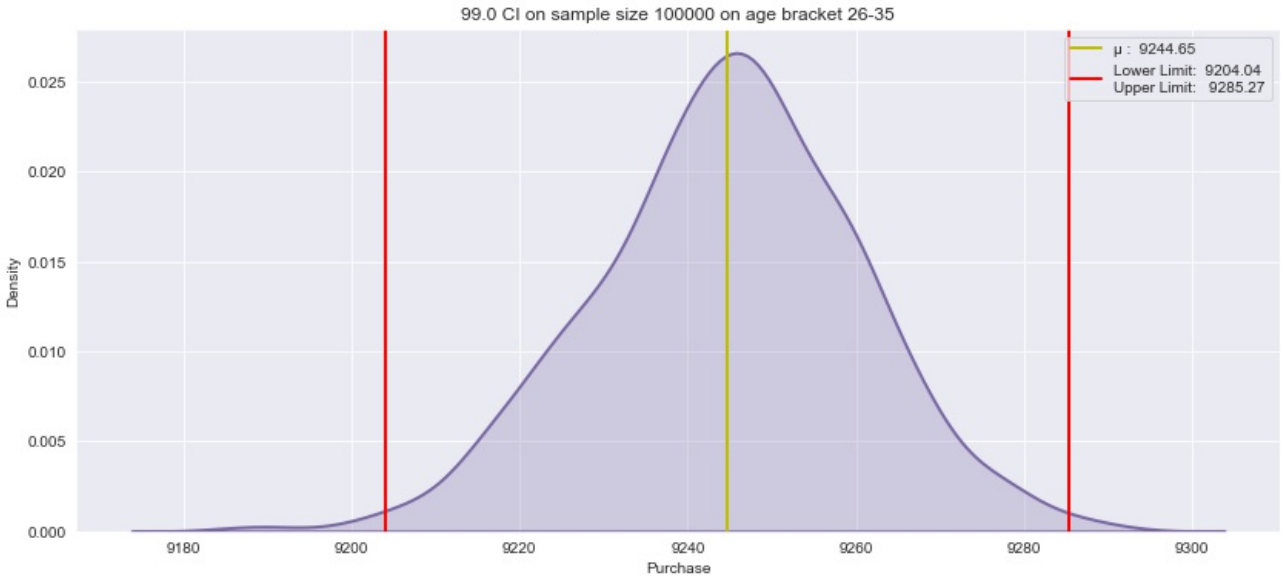
In [77]:

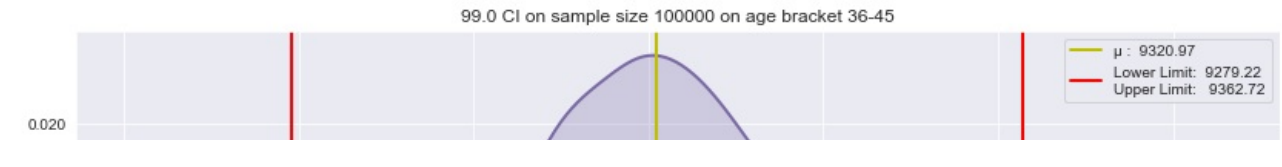
```
itr_size = 1000
smp_size = 10000
ci = 0.99
age_list =['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']

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

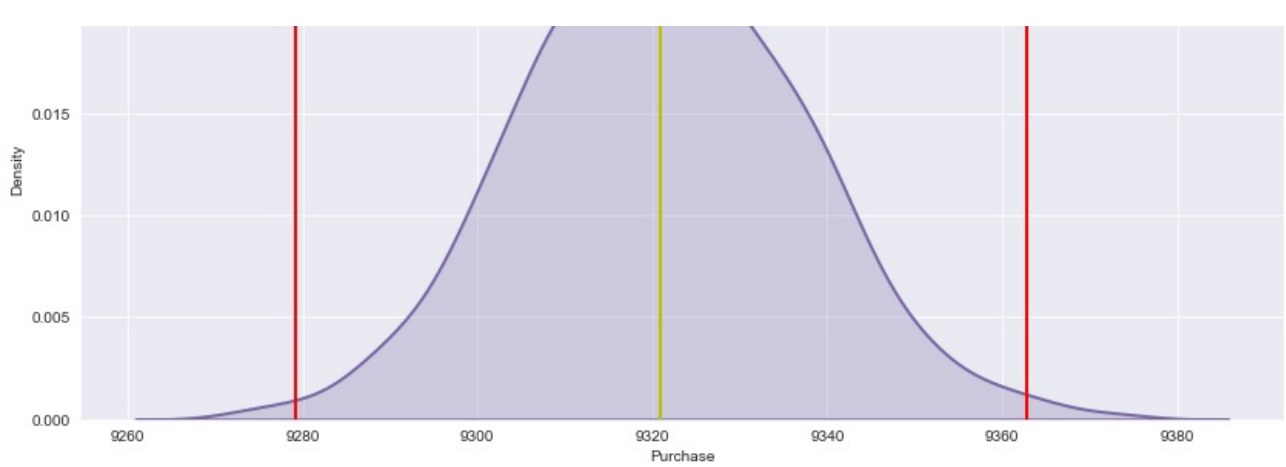
```
for age in age_list:
    mean, ll_m, ul_m = bootstrapping_age(walmart_new[walmart_new['Age'] == age]['Purchase'],smp_siz,itr_size,ci,

    array = np.append(array, np.array([[age,np.round(mean,2), ll_m, ul_m, smp_siz, ([ll_m,ul_m]) ,(ul_m-ll_m),99]

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

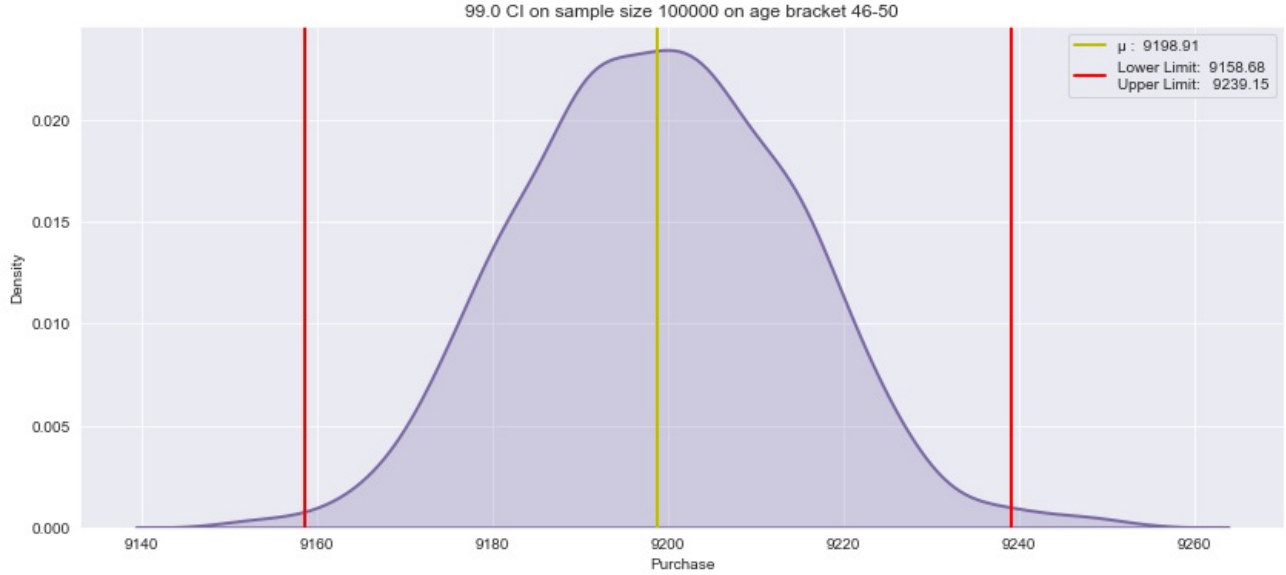


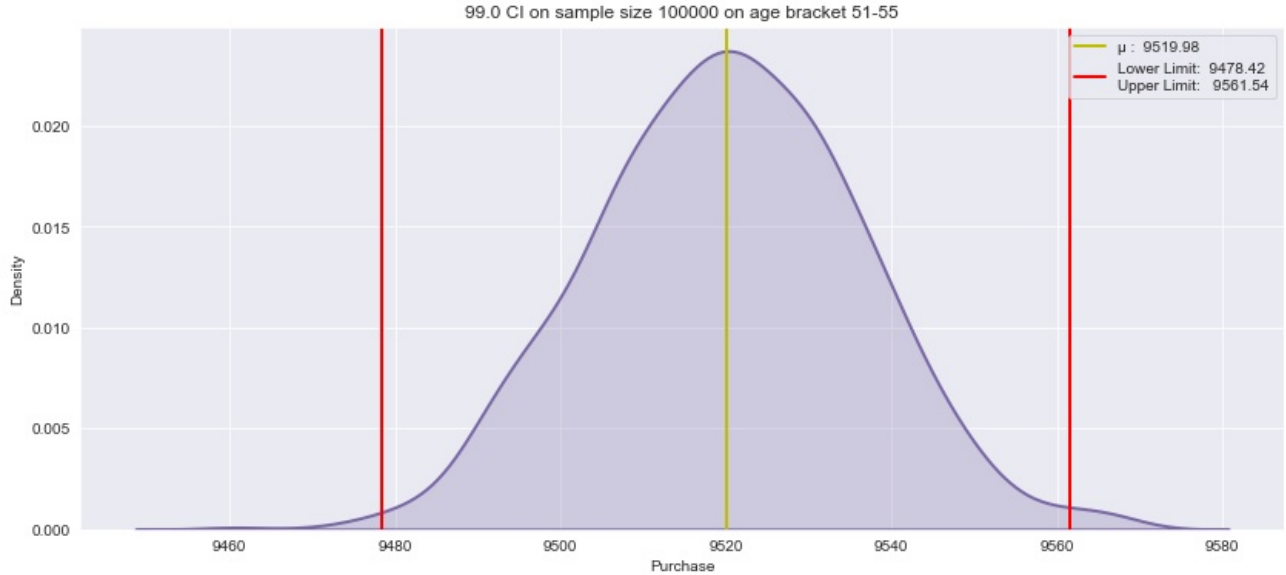


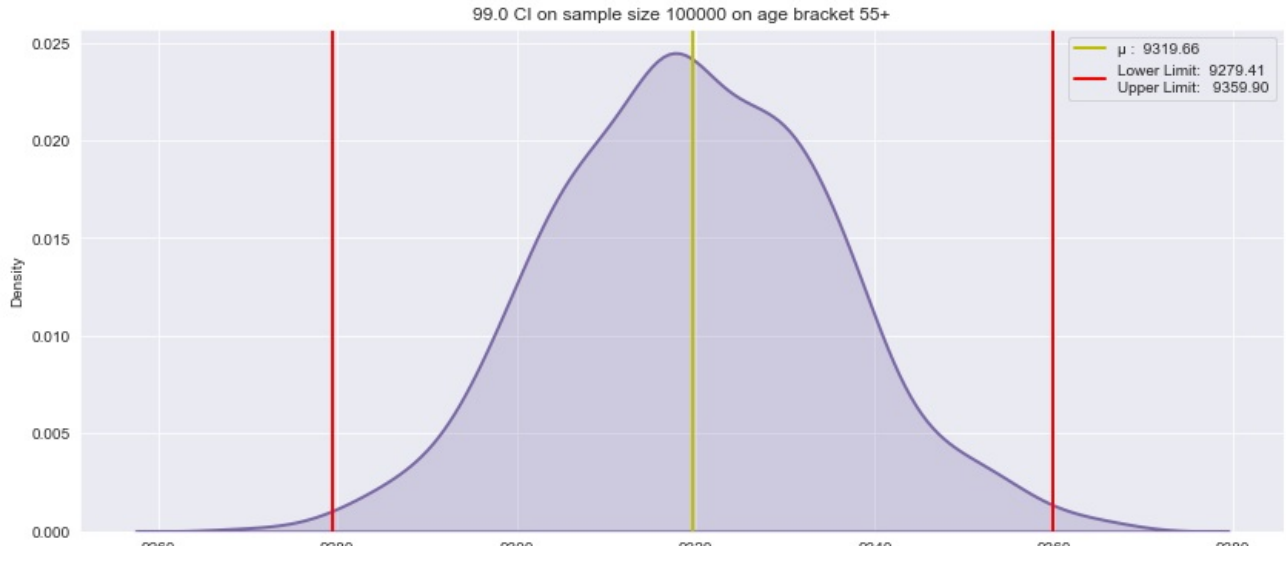


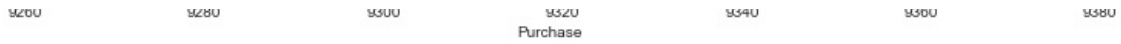



0.015
0.010
0.005
0.000
Density
9260
9280
9300
9320
9340
9360
9380
Purchase


99.0 CI on sample size 100000 on age bracket 46-50
μ : 9198.91
Lower Limit: 9158.68
Upper Limit: 9239.15
0.020
0.015
0.010
0.005
0.000
Density
9140
9160
9180
9200
9220
9240
9260
Purchase


99.0 CI on sample size 100000 on age bracket 51-55
μ : 9519.98
Lower Limit: 9478.42
Upper Limit: 9561.54
0.020
0.015
0.010
0.005
0.000
Density
9460
9480
9500
9520
9540
9560
9580
Purchase

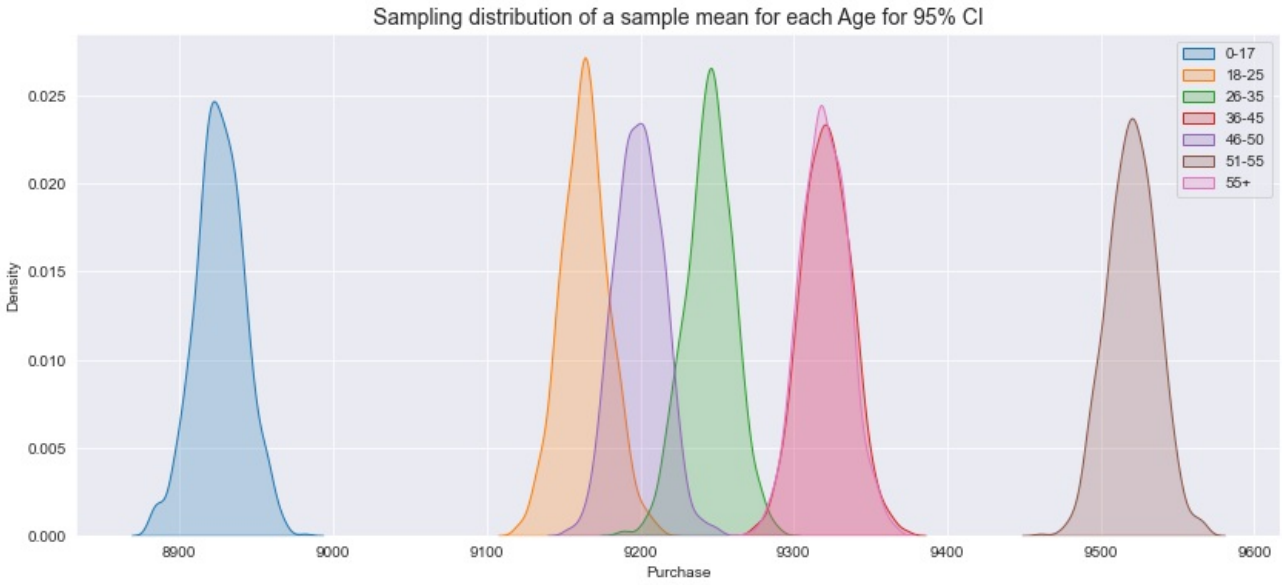

99.0 CI on sample size 100000 on age bracket 55+
μ : 9319.66
Lower Limit: 9279.41
Upper Limit: 9359.90
0.025
0.020
0.015
0.010
0.005
0.000
Density



Checking the Sampling distribution of a sample mean for each Age Group for 99% CI

In [78]:

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

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

plt.title("Sampling distribution of a sample mean for each Age for 95% CI",fontsize=14)
plt.xlabel('Purchase')
plt.legend(loc='upper right')

plt.show()
```



In [79]:

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

Out[79]:

	Age_Group	Mean	Lower_limit	Upper_limit	Sample_Size	CI	Range	Confidence_pct
0	0-17	[8928.29, 8922.77, 8916.81, 8906.55, 8919.38, ...	8884.34	8967.09	100000	[8884.34, 8967.09]	82.75	99
1	18-25	[9134.7, 9173.27, 9161.54, 9178.74, 9170.43, 9...	9123.61	9203.97	100000	[9123.61, 9203.97]	80.36	99
2	26-35	[9257.45, 9225.01, 9240.23, 9255.58, 9235.93, ...	9204.04	9285.27	100000	[9204.04, 9285.27]	81.23	99
3	36-45	[9313.53, 9328.46, 9300.9, 9342.83, 9318.04, 9...	9279.22	9362.72	100000	[9279.22, 9362.72]	83.5	99
4	46-50	[9197.02, 9189.32, 9221.9, 9179.71, 9175.52, 9...	9158.68	9239.15	100000	[9158.68, 9239.15]	80.47	99
5	51-55	[9542.91, 9534.92, 9537.76, 9531.29, 9510.16, ...	9478.42	9561.54	100000	[9478.42, 9561.54]	83.12	99
6	55+	[9318.33, 9309.57, 9332.09, 9334.56, 9325.97, ...	9279.41	9359.9	100000	[9279.41, 9359.9]	80.49	99

Major Inferences

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

- Customers in Age_group 51-55 spend the most between [9478.25, 9560.11] for 99% CI

5. Final Insights

Based on EDA

- The majority of our customers come from city category B but customers come from City category C spent more as mean is 9719.
- The majority of users come from City Category C, but more people from City Category B tend to purchase, which suggests the same users visit the mall multiple times in City Category B.
- Majority of Customers purchase within the 5,000 - 20,000 range.
- Males clearly purchase more than females. 75% of men and only 25% of women purchase products.
- Most mall customers are between the ages of 26 and 35.60% of purchases are made by people between the ages of 26 and 45
- City Category B accounts for 42%, City Category C 31%, and City Category A represents 27% of all customer purchases.Purchases are high in city category C
- Most mall customers are between the ages of 26 and 35.City category C has more customers between the ages of 18 and 45.
- In City Category C, there are slightly more female customers.
- Product 5 and 8 is common among females.
- Product Category B is popular among age group 55+
- 58:42 is the ratio between married and unmarried customers

Based on CLT & CI

- As the sample size increases, the two groups start to become distinct. With increasing sample size, Standard error of the mean in the samples decreases. For sample size 100000 is 0.49 with confidence is 90%.
- Overlappings are increasing with a confidence interval of 95%. Due to the increasing CI, we consider higher ranges within which the actual population might fall, so that both mean purchase are more likely to fall within the same range.
- Using confidence interval 99%, the mean purchase value by gender shows a similar pattern to that found with confidence interval 90% & 95%
- For Female (sample size 100000) range for mean purchase with confidence interval 99% is [8634.54, 8707.85]
- For Male range for mean purchase with confidence interval 99% is [9328.03, 9409.07]
- When the confidence percentage increases, the spread, that is the difference between the upper and lower limits, also increases. For Female Confidence percent as [90,95,99] have difference between the upper & lower limits as [50.46,59,73.31]
- Overlapping is evident for married vs single customer spend even when more samples are analyzed, which indicates that customers spend the same regardless of whether they are single or married.
- For Unmarried customer (sample size 100000) range for mean purchase with confidence interval 99% is [9225.71, 9305.43]
- For married customer (sample size 100000) range for mean purchase with confidence interval 99% is [9218.62, 9303.25]
- For Female (sample size 100000) range for mean purchase with confidence interval 90% is [8702.35, 8751.09]
- For Male (sample size 100000) range for mean purchase with confidence interval 90% is [9402.28, 9455.2]

6. Recommendations

- As per the analysis females spend less than males on average, management needs to focus on their specific needs differently. Adding some additional offers for women can increase their spending on Black Friday.

- The management should have some offers on kids (0-17 years) in order to increase sales as there is lowest mean purchase in this age group.
- In order to attract more young shoppers, they can offer some discounts, video games subscription etc to lure more younger generation.
- There are more married customers than single. But as per analysis single customers mean purchase is higher than married one. Management needs to focus more on married customers as they are large in numbers but purchasing less.
- Product Category B is quite demanded for 55+ age group. There is sudden drop in other categories A & C for the same age group. Management need to focus more on other categories for 55+ age group.
- Most of customer stayed more than 2 years in same city having slightly lower mean purchase. Managment should perform research for all the stores why it is low
- As product category 5 & 8 are quite demanding among females. In order to increase sale, management needs to introduce more items of these product categories in their stores
- Management should conduct a survey for different age group, what they like most about some particular brand of specific product category to get more insights which will help businesses in customer aquistion & retention.
- As product category 1 & 5 are popular among both male and female, in order to increase sales management need to give additional offers and introduce some strategy regarding cross-sale or upsell.
- Management need to find out the reason why proudct category 9 & 17 are less bought. If customers don't like this product that needs to be removed from inventory by slashing the price down.
- As this dataset contituted on Black Friday Sales data, it would be quite obvious that for specific age group, specific product category there would be more demand and churn. Management can draw insights as per analysis for the customer purchase behaviour.

In [ ]: