In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import t
import scipy.stats as stats
from statistics import mean, median, mode, stdev
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
wm  = pd.read_csv(r"C:\Users\Acer\Downloads\walmart_data_new.csv")
```

In [3]:

```
wm
```

Out[3]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Ye
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	
...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	
550064	1006035	P00375436	F	26-35	1	C	
550065	1006036	P00375436	F	26-35	15	B	
550066	1006038	P00375436	F	55+	1	C	
550067	1006039	P00371644	F	46-50	0	B	

550068 rows × 10 columns

INSIGHT :

- In the given sample of the Wall Mart dataset contains 550068 rows × 10 columns.
- The original data cotains 50 million Male and 50 million Female customers BUT in the sampple there is a 'GENDER Discrepency'.

- Therefore the data should be Downsapled if not the extrapolation will yeild a Biased Approximation.

In [4]:

```
df  = wm.copy()
```

In [5]:

```
df
```

Out[5]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Ye
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	
...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	
550064	1006035	P00375436	F	26-35	1	C	
550065	1006036	P00375436	F	26-35	15	B	
550066	1006038	P00375436	F	55+	1	C	
550067	1006039	P00371644	F	46-50	0	B	

550068 rows × 10 columns

In [6]:

```
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
```

Observation

- There is a mix of Integer columsn and Catagorical Objects which we will be working on later

Changing the Columns into Catagory

In [7]:

```
cols = [ 'Age' , 'User_ID' , 'Gender' , 'Marital_Status' , 'Product_Category' , 'City_(

for i in cols :
        df[i] = df[i].astype('category')
```

In [8]:

```
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  category
 1   Product_ID                  550068 non-null  object  
 2   Gender                      550068 non-null  category
 3   Age                         550068 non-null  category
 4   Occupation                  550068 non-null  int64   
 5   City_Category               550068 non-null  category
 6   Stay_In_Current_City_Years  550068 non-null  object  
 7   Marital_Status              550068 non-null  category
 8   Product_Category            550068 non-null  category
 9   Purchase                    550068 non-null  int64   
dtypes: category(6), int64(2), object(2)
memory usage: 20.6+ MB
```

Checking for Null values

In [9]:

```
df.isnull().sum()/len(df)*100
```

Out[9]:

```
User_ID                       0.0
Product_ID                    0.0
Gender                        0.0
Age                           0.0
Occupation                    0.0
City_Category                 0.0
Stay_In_Current_City_Years    0.0
Marital_Status                0.0
Product_Category              0.0
Purchase                      0.0
dtype: float64
```

Changing the Gender Column values into M : Males and F : Female

In [10]:

```
df['Gender'] = df['Gender'].replace({'M': 'Male', 'F': 'Female'})
```

Changing the Marital_status Column values into 1 : Married and 0 : Single

In [11]:

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

In [15]:

```
df.nunique()
```

Out[15]:

```
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
```

UNIQUE VALUES IN EACH COLUMN

In [16]:

```
colname = ['Gender','Age','City_Category','Stay_In_Current_City_Years','Marital_Status'
for col in colname:
    print("\nUnique values of ",col," are : ",list(df[col].unique()))
```

```
Unique values of  Gender  are :  ['Female', 'Male']

Unique values of  Age  are :  ['0-17', '55+', '26-35', '46-50', '51-55', '36
-45', '18-25']

Unique values of  City_Category  are :  ['A', 'C', 'B']

Unique values of  Stay_In_Current_City_Years  are :  ['2', '4+', '3', '1',
'0']

Unique values of  Marital_Status  are :  ['Single', 'Married']

Unique values of  Occupation  are :  [10, 16, 15, 7, 20, 9, 1, 12, 17, 0, 3,
4, 11, 8, 19, 2, 18, 5, 14, 13, 6]
```

In [17]:

```
df.describe()
```

Out[17]:

	Occupation	Purchase
count	550068.000000	550068.000000
mean	8.076707	9263.968713
std	6.522660	5023.065394
min	0.000000	12.000000
25%	2.000000	5823.000000
50%	7.000000	8047.000000
75%	14.000000	12054.000000
max	20.000000	23961.000000

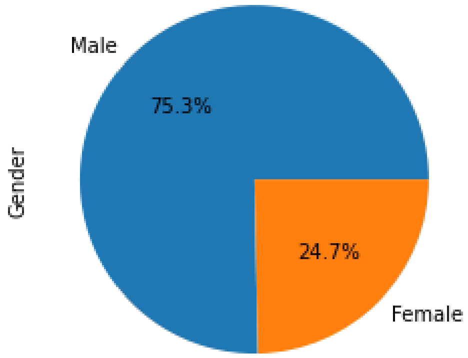
Observations

- There is a large difference in the mean and the median showing alot of outliers in the purchases

Univariate Analysis

In [18]:

```
df['Gender'].value_counts().plot(kind='pie',autopct='%.1f%%')
plt.show()
```



In [19]:

```
df['City_Category'].value_counts(normalize = True)*100
```

Out[19]:

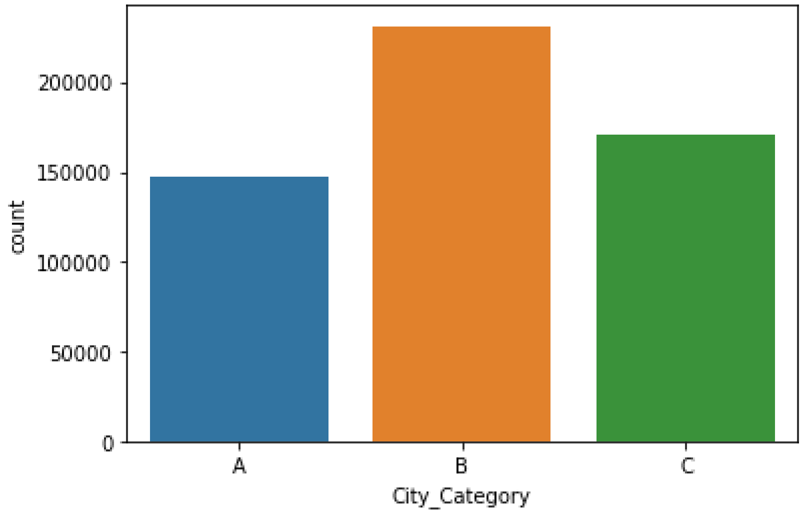
```
B    42.026259
C    31.118880
A    26.854862
Name: City_Category, dtype: float64
```

In [20]:

```
sns.countplot(df['City_Category'])
```

Out[20]:

```
<AxesSubplot:xlabel='City_Category', ylabel='count'>
```



Observations

- The customers of city of category 'B' are purchasing the most(42%), whereas the people from City category A are least interested in purchasing from Black Friday Sales.

In [21]:

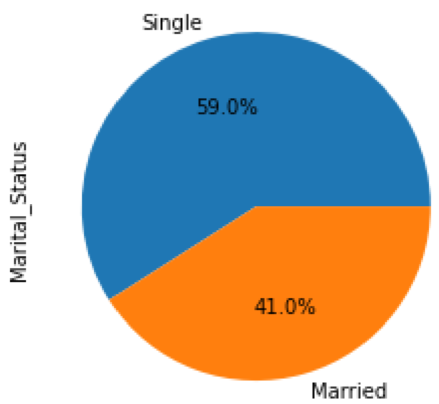
```
# df['Stay_In_Current_City_Years'].value_counts(normalize = True)*100
```

In [22]:

```
# sns.countplot(df['Stay_In_Current_City_Years'])
```

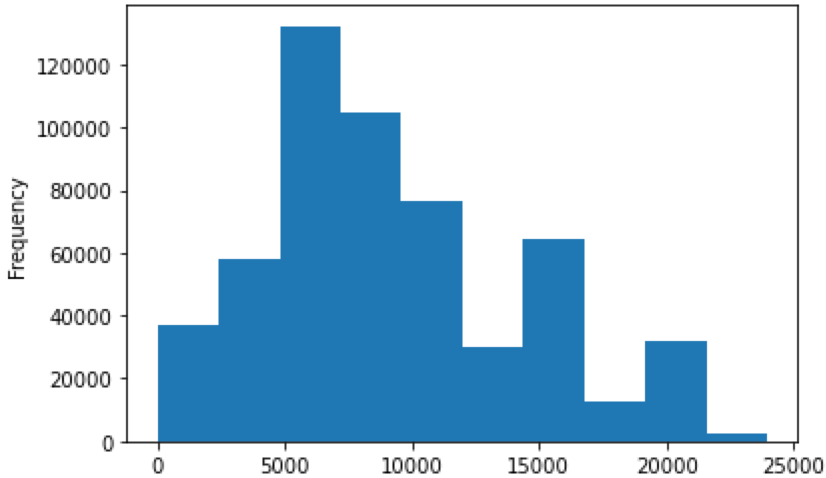
In [23]:

```
df['Marital_Status'].value_counts().plot(kind='pie',autopct='%.1f%%')
plt.show()
```



In [24]:

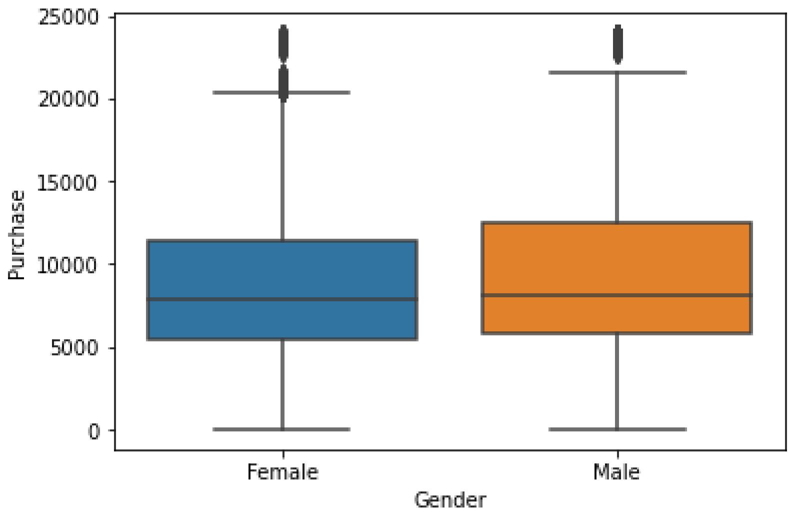
```
df['Purchase'].plot(kind='hist')
plt.show()
# print(df['Purchase'].value_counts())
```



Bivariate Analysis

In [25]:

```
sns.boxplot(x = 'Gender' , y = 'Purchase' , data = df)
plt.show()
```



In [26]:

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

Out[26]:

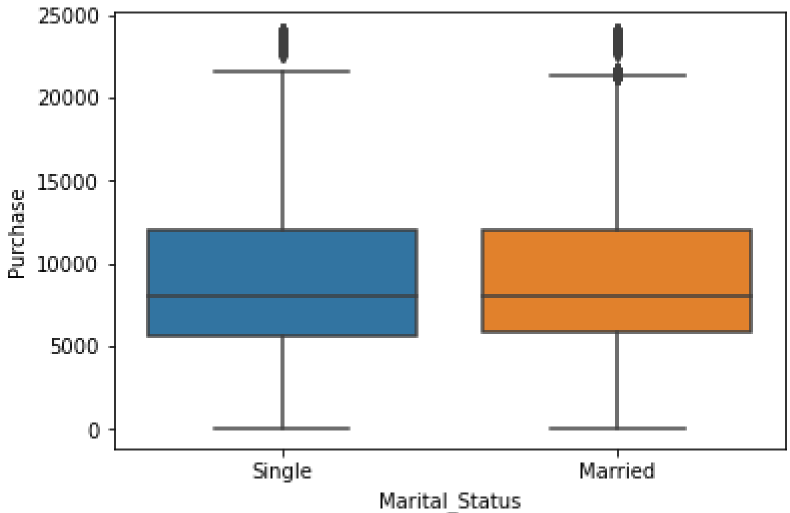
Gender	count	mean	std	min	25%	50%	75%	max
Female	135809.0	8734.565765	4767.233289	12.0	5433.0	7914.0	11400.0	23959.0
Male	414259.0	9437.526040	5092.186210	12.0	5863.0	8098.0	12454.0	23961.0

Observation

- We can see that he Males have bought more when compared to the Females on Black Friday

In [27]:

```
sns.boxplot(x = 'Marital_Status', y = 'Purchase', data = df)
plt.show()
```



In [28]:

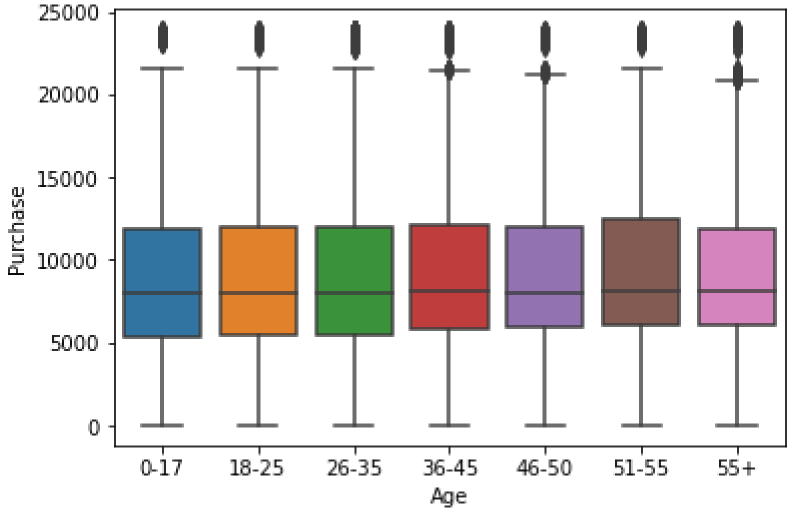
```
df.groupby(['Marital_Status'])['Purchase'].describe()
```

Out[28]:

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

In [29]:

```
sns.boxplot(x = 'Age', y = 'Purchase', data = df)
plt.show()
```



In [30]:

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

Out[30]:

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

In [74]:

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

Out[74]:

City_Category	count	mean	std	min	25%	50%	75%	max
A	147036.0	8845.367393	4804.639577	12.0	5398.0	7922.0	11747.0	21398.0
B	230114.0	9086.502707	4873.509950	12.0	5455.0	7996.0	11952.0	21399.0
C	170241.0	9645.647300	5105.363663	12.0	6021.0	8571.0	13050.0	21398.0

In [31]:

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

Out[31]:

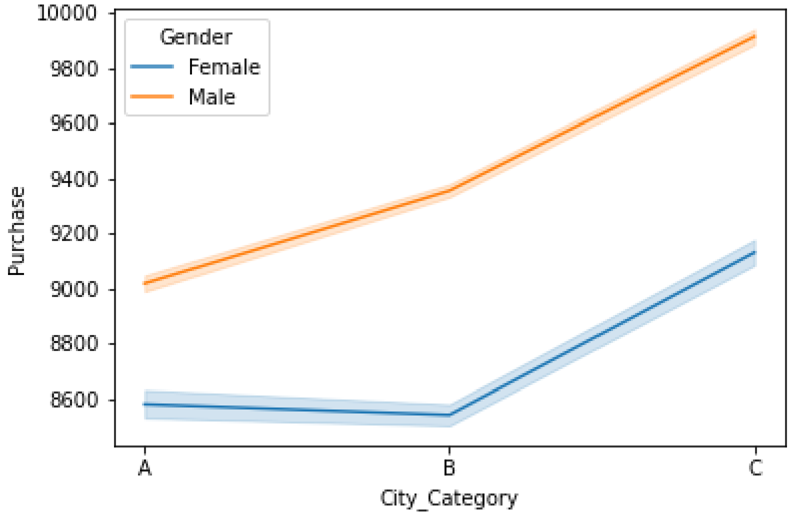
```
City_Category
A    1045
B    1707
C    3139
Name: User_ID, dtype: int64
```

Observations

- There are more single than married in the dataset.
- Most customers are between the ages group of 26 and 35.
- The majority of our customers come from city category B but customers come from City category C spent more as mean is 9645.
- Male customers tend to spend more than female customers, as the mean is higher for male customers.
- The majority of users come from City Category C, but more people from City
- Category B tend to purchase, which suggests the same users visit the mall multiple times in City Category B.

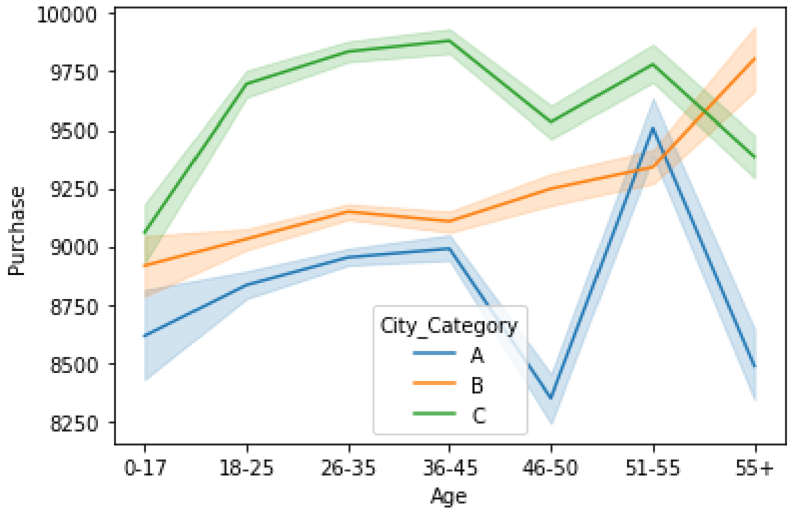
In [32]:

```
sns.lineplot(x='City_Category',y='Purchase',  data=df,  hue='Gender')
plt.show()
```



In [33]:

```
sns.lineplot(x='Age',y='Purchase',  data=df,  hue='City_Category')
plt.show()
```



Observation

- Purchase are higher in city catagory C
- Most of the customers are 55+ and live in city category B
- City category C has more customers between the ages of 18 and 45.

In [34]:

```
gender_nos = df.groupby(['Age','Gender'])['Gender'].count()
gender_nos = gender_nos.unstack(level = 'Gender')
print(gender_nos)
```

```
Gender  Female    Male
Age
0-17      5083   10019
18-25    24628   75032
26-35    50752  168835
36-45    27170   82843
46-50    13199   32502
51-55     9894   28607
55+       5083   16421
```

In [ ]:

```

```

In [35]:

```
print(round(df.Stay_In_Current_City_Years.value_counts(normalize = True) *100,2))
```

```
1     35.24
2     18.51
3     17.32
4+    15.40
0     13.53
Name: Stay_In_Current_City_Years, dtype: float64
```

In [ ]:

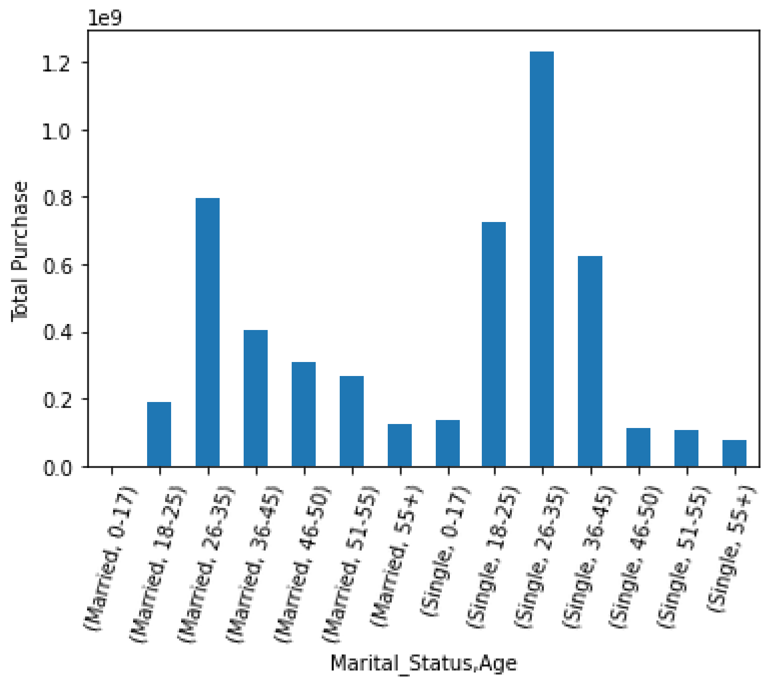
```

```

In [36]:

```
df.groupby(['Marital_Status','Age'])['Purchase'].sum().plot(kind = 'bar')
plt.ylabel('Total Purchase')
plt.xticks(rotation = 75)
```

Out[36]:

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13]),
 [Text(0, 0, '(Married, 0-17)'),
  Text(1, 0, '(Married, 18-25)'),
  Text(2, 0, '(Married, 26-35)'),
  Text(3, 0, '(Married, 36-45)'),
  Text(4, 0, '(Married, 46-50)'),
  Text(5, 0, '(Married, 51-55)'),
  Text(6, 0, '(Married, 55+)'),
  Text(7, 0, '(Single, 0-17)'),
  Text(8, 0, '(Single, 18-25)'),
  Text(9, 0, '(Single, 26-35)'),
  Text(10, 0, '(Single, 36-45)'),
  Text(11, 0, '(Single, 46-50)'),
  Text(12, 0, '(Single, 51-55)'),
  Text(13, 0, '(Single, 55+)')])
```



Observation

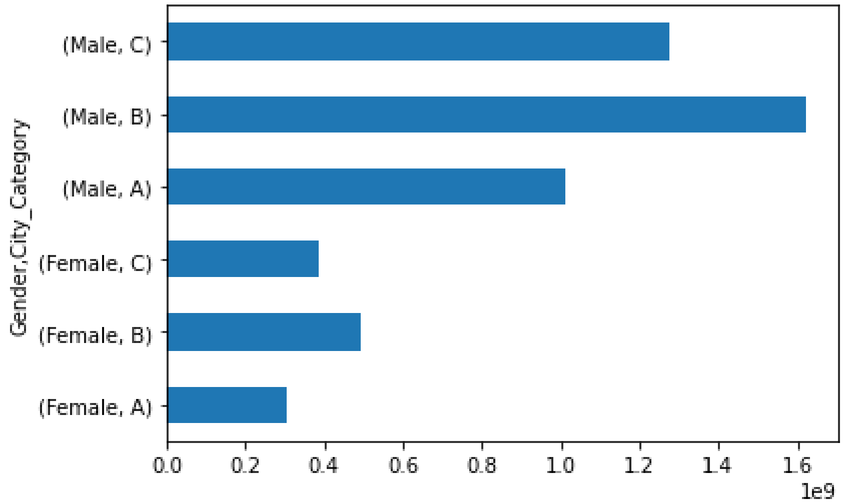
- Age group of 26 - 35 buys more followed by 18 - 35

In [37]:

```
df.groupby(['Gender','City_Category'])['Purchase'].sum()
df.groupby(['Gender','City_Category'])['Purchase'].sum().plot(kind = 'barh')
```

Out[37]:

```
<AxesSubplot:ylabel='Gender,City_Category'>
```



Observation

- We can see that the Males and Females in the city B has brought more on the Black Friday's followed by Male and Female of city C

Handling Outliers for Purchase

In [38]:

```
Q3 = np.percentile(df['Purchase'],75)
Q1 = np.percentile(df['Purchase'],25)
IQR = Q3-Q1
df = df[(df['Purchase'] > Q1 - 1.5*IQR) & (df['Purchase'] < Q3 + 1.5*IQR)]
```

In [39]:

```
df['Purchase']
```

Out[39]:

```
0          8370
1         15200
2          1422
3          1057
4          7969
          ...
550063     368
550064     371
550065     137
550066     365
550067     490
Name: Purchase, Length: 547391, dtype: int64
```

In [40]:

```
Q1
```

Out[40]:

```
5823.0
```

In [41]:

```
Q3
```

Out[41]:

```
12054.0
```

In [42]:

```
df
```

Out[42]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Ye
0	1000001	P00069042	Female	0-17	10	A	
1	1000001	P00248942	Female	0-17	10	A	
2	1000001	P00087842	Female	0-17	10	A	
3	1000001	P00085442	Female	0-17	10	A	
4	1000002	P00285442	Male	55+	16	C	
...	...	...	...	...	...	...	
550063	1006033	P00372445	Male	51-55	13	B	
550064	1006035	P00375436	Female	26-35	1	C	
550065	1006036	P00375436	Female	26-35	15	B	
550066	1006038	P00375436	Female	55+	1	C	
550067	1006039	P00371644	Female	46-50	0	B	

547391 rows × 10 columns

HAVE DONE EXTRAPOLATION IN 2 METHODS -

```
- BOOTSTRAPING
- NORMAL CLT VIA SAMPLING
```

BOOTSTRAPING METHOD

In [43]:

```
import matplotlib.pyplot as plt
import statistics
from math import sqrt

def plot_confidence_interval(x, values,  color='#2187bb', horizontal_line_width=0.25,co

    def CI_with_different_sample_size(data,confidence, sample_size=10000,trials = 500):

        bootstrapped_mean= np.empty(trials)

        for i in range(trials):
            btssample = data.sample(n=sample_size,replace=True)
            bootstrapped_mean[i] = np.mean(btssample)
        sample_mean = np.mean(bootstrapped_mean)
        sample_std = np.std(data)
        standard_error = sample_std/np.sqrt(sample_size)
        talfa_by2 = t.ppf((1-((1-(confidence)/100)/2)),df = sample_size-1)
        margin_of_error = talfa_by2*standard_error

        return margin_of_error,sample_size+margin_of_error,sample_size-margin_of_error




    error,bottom,top = CI_with_different_sample_size(values,confidence)

    left = x - horizontal_line_width / 2
    top = np.mean(values) - error
    right = x + horizontal_line_width / 2
    bottom = np.mean(values) + error
    print("Confidence Interval : ",(top,bottom))
    plt.plot([x, x], [top, bottom], color=color)
    plt.plot([left, right], [top, top], color=color)
    plt.plot([left, right], [bottom, bottom], color=color)
    plt.plot(x, np.mean(values), 'o', color='#f44336')
    print("Sample Mean :",np.mean(values)," and ","Margin of Error :", error)
```

In [44]:

```
def Bootstrapping_CLT_CI(data, confidence=95 , sample_size = 10000,r = 200):


    sns.distplot(data,bins = 20)
    plt.show()

    bootstrapped_mean= np.empty(r)

    for i in range(r):
        btssample = data.sample(n=sample_size,replace=True)
        bootstrapped_mean[i] = np.mean(btssample)

    sns.distplot(bootstrapped_mean,bins = 20)

    sample_mean = np.mean(bootstrapped_mean)
    sample_std = np.std(bootstrapped_mean)

    talfa_by2 = t.ppf((1-((1-(confidence)/100)/2)),df = sample_size-1)
    margin_of_error = talfa_by2 * sample_std

    print("t:",talfa_by2)
    print("sample mean :",sample_mean)
    print("sample standard deviation :",sample_std)
    print("sample size: ",sample_size)
    print("Margin of Error :",margin_of_error)


    lower_ = sample_mean - margin_of_error
    upper_ = sample_mean + margin_of_error
    CI = (lower_,upper_)

    plt.axvline(x = lower_,c = "r")
    plt.axvline(x = upper_,c = "r")
    plt.show()



    print("Confidence Interval : ",CI)
```

In [45]:

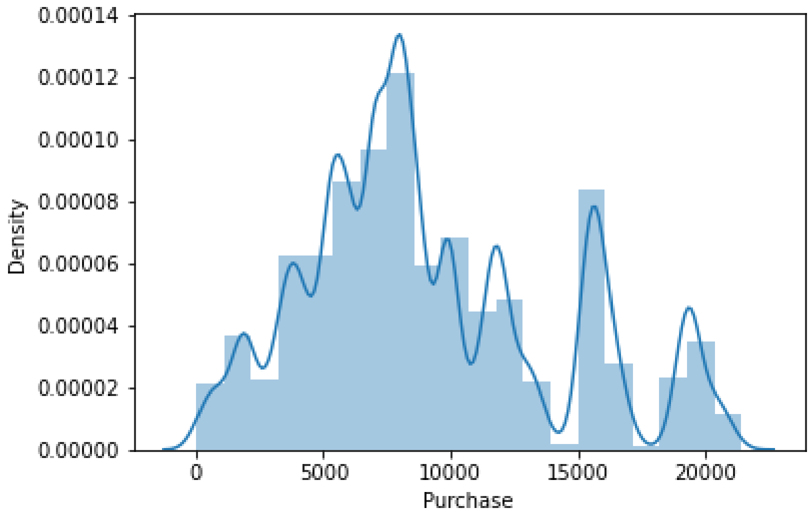
```
df.groupby([df["Marital_Status"]])["Purchase"].describe()
```

Out[45]:

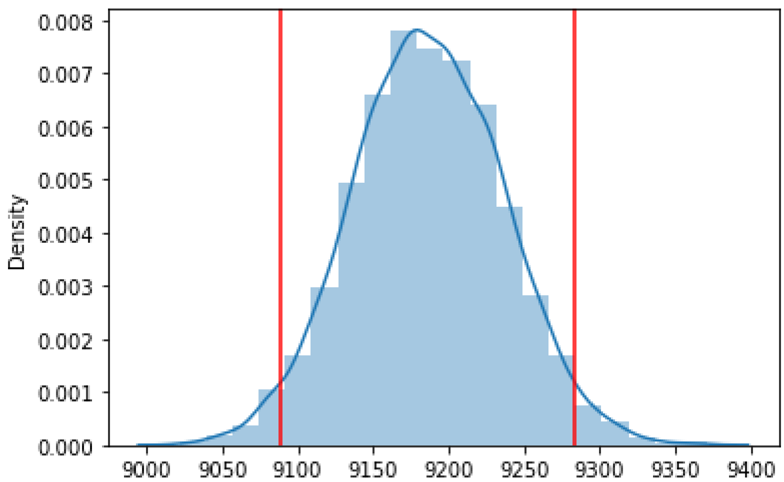
	count	mean	std	min	25%	50%	75%	max
Marital_Status								
Married	224149.0	9187.040076	4925.205232	12.0	5833.0	8042.0	12006.0	21398.0
Single	323242.0	9201.581849	4948.327397	12.0	5480.0	8035.0	12028.0	21399.0

In [46]:

```
1 Bootstrapping_CLT_CI(df.loc[df["Marital_Status"]=='Married']["Purchase"], confidence=9!
```



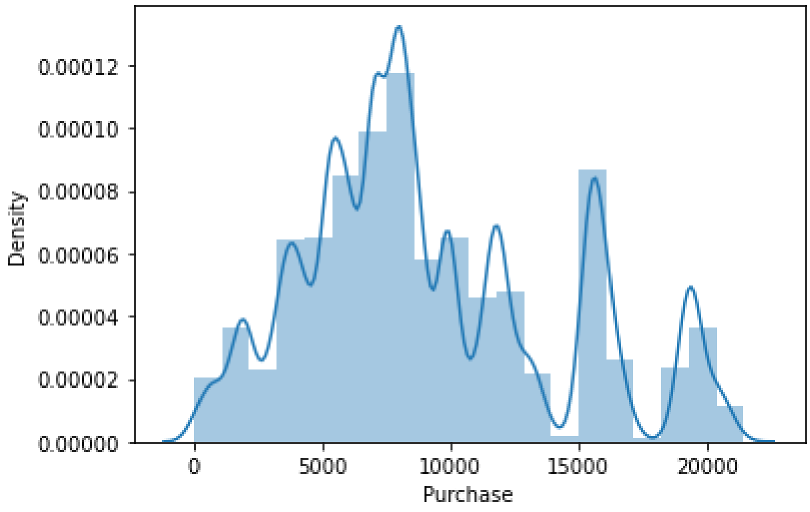

```
t: 1.9602012636213575
sample mean : 9186.56257102
sample standard deviation : 49.645521876509505
sample size:  10000
Margin of Error : 97.31521471547568
```



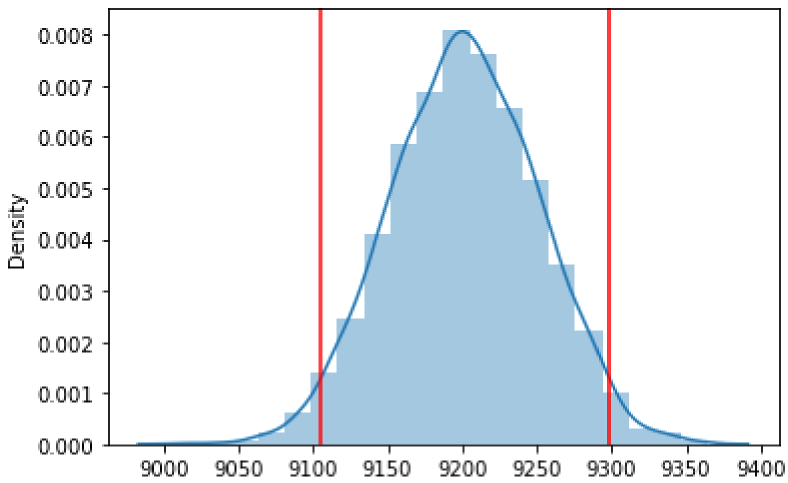

```
Confidence Interval :  (9089.247356304524, 9283.877785735476)
```

In [47]:

```
1 Bootstrapping_CLT_CI(df.loc[df["Marital_Status"]=='Single']["Purchase"], confidence=95
```




```
t: 1.9602012636213575
sample mean : 9201.79322886
sample standard deviation : 49.224612221387886
sample size:  10000
Margin of Error : 96.49014707763585
```

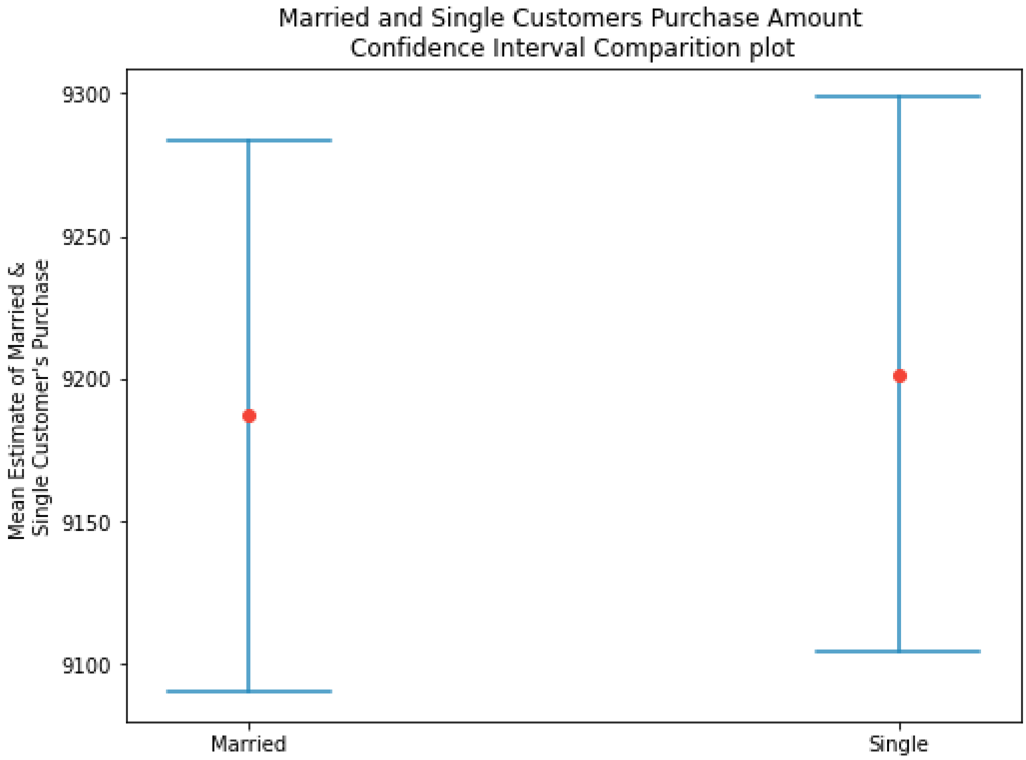



```
Confidence Interval :  (9105.303081782364, 9298.283375937637)
```

In [48]:

```python
plt.figure(figsize=(8,6))
plot_confidence_interval(x=1,values=df[df["Marital_Status"]=='Married']["Purchase"])
plot_confidence_interval(x=2,values=df[df["Marital_Status"]=='Single']["Purchase"])
plt.xticks([1,2],["Married","Single"])
plt.title("Married and Single Customers Purchase Amount \nConfidence Interval Compariti
plt.ylabel("Mean Estimate of Married & \nSingle Customer's Purchase")
plt.show()
```

```
Confidence Interval :  (9090.496356187012, 9283.58379585471)
Sample Mean : 9187.040076020861  and  Margin of Error : 96.54371983384888
Confidence Interval :  (9104.584822758414, 9298.578875028383)
Sample Mean : 9201.581848893398  and  Margin of Error : 96.9970261349839
```



Observations

- 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 range for mean purchase with confidence interval 95% is [9104.584, 9298.57]
- For married customer range for mean purchase with confidence interval 95% is [9090.49, 9283.58]

computing the average female and male expenses

In [49]:

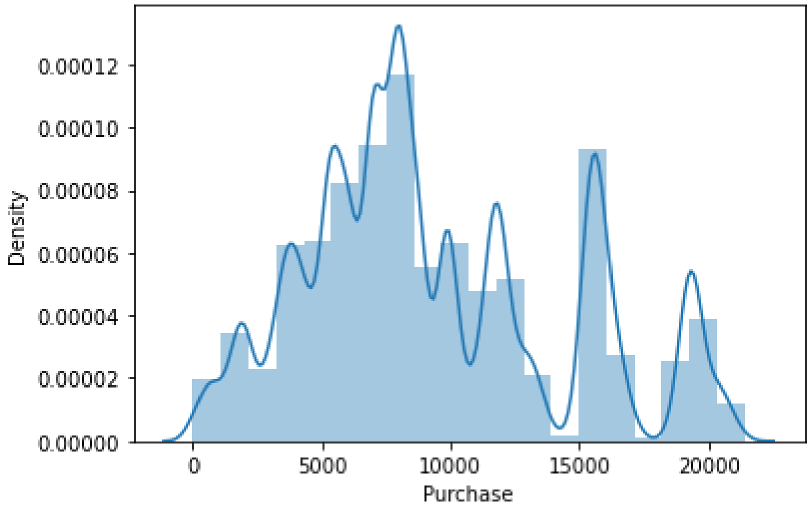
```
df.groupby([df["Gender"]])["Purchase"].describe()
```

Out[49]:

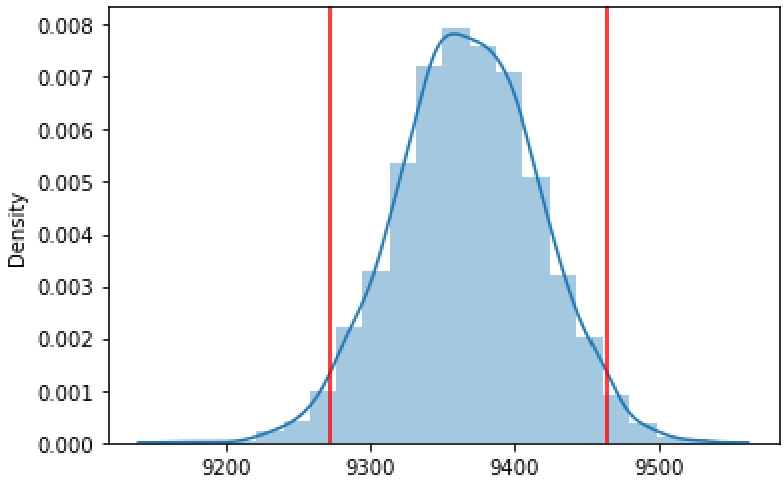
Gender	count	mean	std	min	25%	50%	75%	max
Female	135220.0	8671.049039	4679.058483	12.0	5429.0	7906.0	11064.0	21398.0
Male	412171.0	9367.724355	5009.234088	12.0	5852.0	8089.0	12247.0	21399.0

In [50]:

```
Bootstrapping_CLT_CI(df.loc[df["Gender"]=='Male']["Purchase"], confidence=95 , sample_s
```



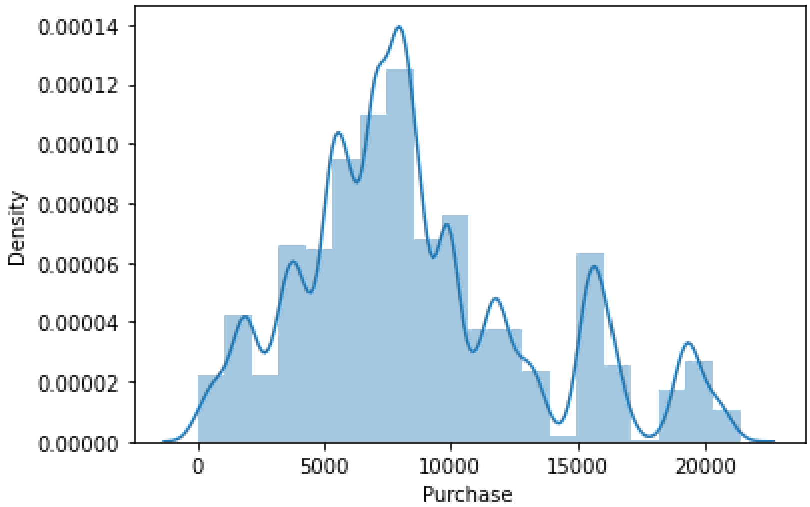

```
t: 1.9602012636213575
sample mean : 9367.229467000001
sample standard deviation : 48.915247488869866
sample size:  10000
Margin of Error : 95.88372993803415
```



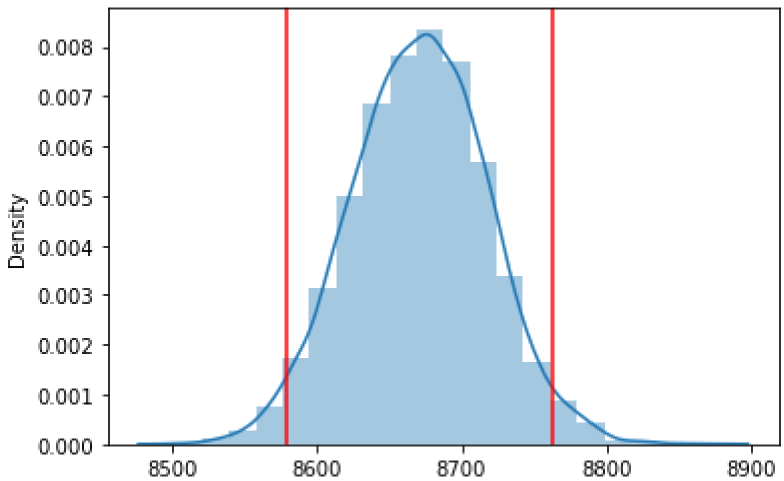

```
Confidence Interval :  (9271.345737061967, 9463.113196938035)
```

In [51]:

```
1 Bootstrapping_CLT_CI(df.loc[df["Gender"]=='Female']["Purchase"], confidence=95 , sample
```




```
t: 1.9602012636213575
sample mean : 8670.75864902
sample standard deviation : 46.910187744100625
sample size:  10000
Margin of Error : 91.95340929270117
```

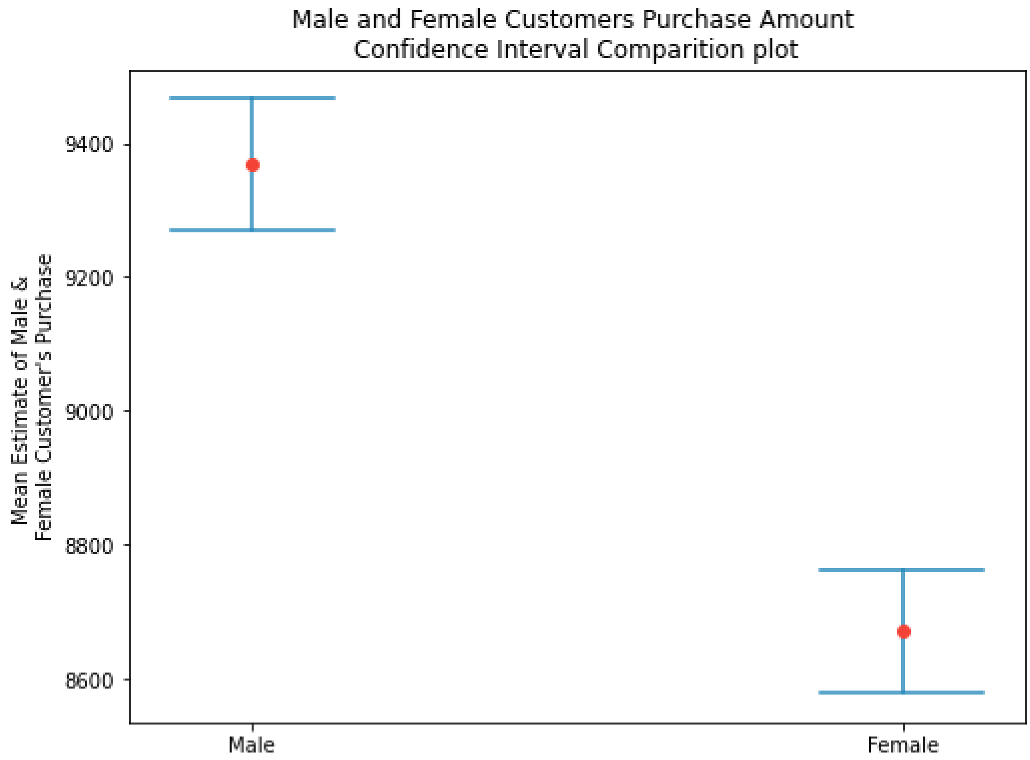



```
Confidence Interval :  (8578.805239727299, 8762.7120583127)
```

In [52]:

```
plt.figure(figsize=(8,6))
plot_confidence_interval(x=1,values=df[df["Gender"]=='Male']["Purchase"])
plot_confidence_interval(x=2,values=df[df["Gender"]=='Female']["Purchase"])
plt.xticks([1,2],["Male","Female"])
plt.title("Male and Female Customers Purchase Amount \nConfidence Interval Comparition
plt.ylabel("Mean Estimate of Male & \nFemale Customer's Purchase")
plt.show()
```

```
Confidence Interval :  (9269.533403922314, 9465.915305472574)
Sample Mean : 9367.724354697444  and  Margin of Error : 98.19095077512894
Confidence Interval :  (8579.330414240561, 8762.767662966951)
Sample Mean : 8671.049038603756  and  Margin of Error : 91.71862436319562
```



Observations

- Overlapping is not evident for Male vs Female customer ,when more samples are analyzed, 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 0.46
- For Male range for mean purchase with confidence interval 95% is [9269.53, 9465.91]
- For married customer range for mean purchase with confidence interval 95% is [8579.33, 8762.76]

In [53]:

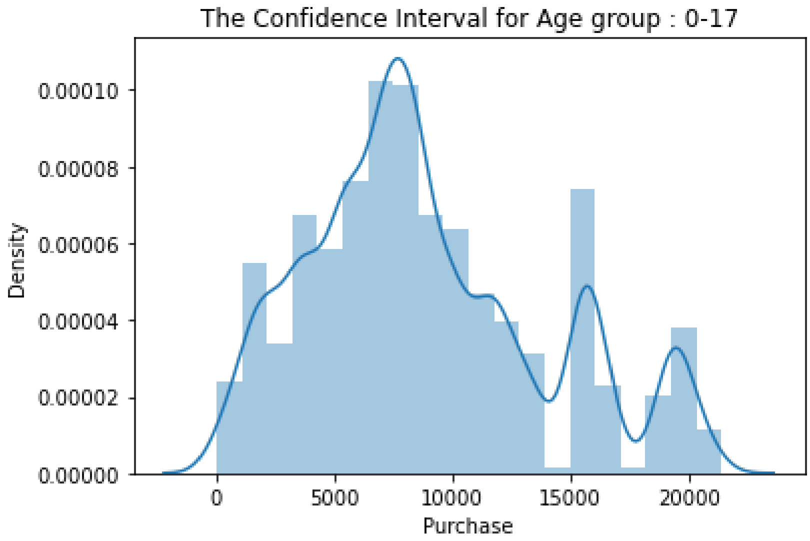
```
df.groupby([df["Age"]])["Purchase"].describe()
```

Out[53]:

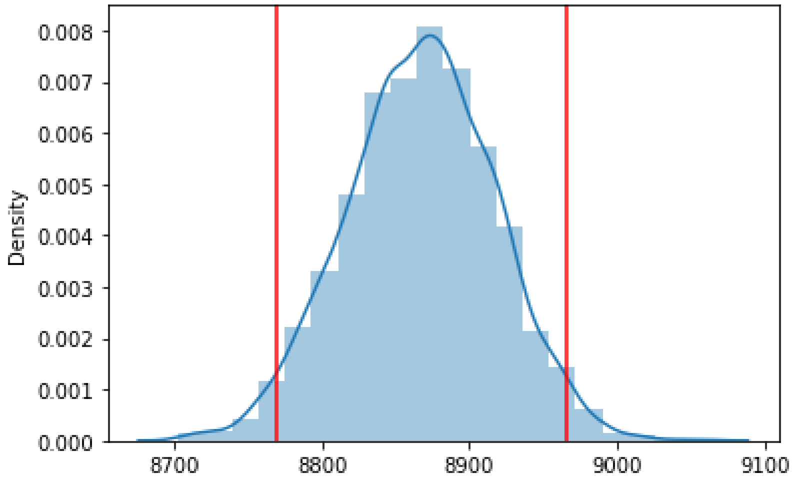
Age	count	mean	std	min	25%	50%	75%	max
0-17	15032.0	8867.447046	5030.052846	12.0	5324.0	7974.5	11833.25	21342.0
18-25	99334.0	9124.031731	4978.831062	12.0	5412.0	8020.0	12004.00	21398.0
26-35	218661.0	9193.469924	4937.410901	12.0	5471.0	8021.0	12018.00	21398.0
36-45	109409.0	9254.202214	4927.744433	12.0	5866.0	8051.0	12065.00	21399.0
46-50	45442.0	9128.985080	4867.413951	12.0	5879.0	8025.0	11958.00	21391.0
51-55	38191.0	9423.121704	4953.644650	12.0	6007.0	8118.0	12123.00	21388.0
55+	21322.0	9216.650220	4861.626596	12.0	6007.0	8092.5	11837.75	21345.0

In [54]:

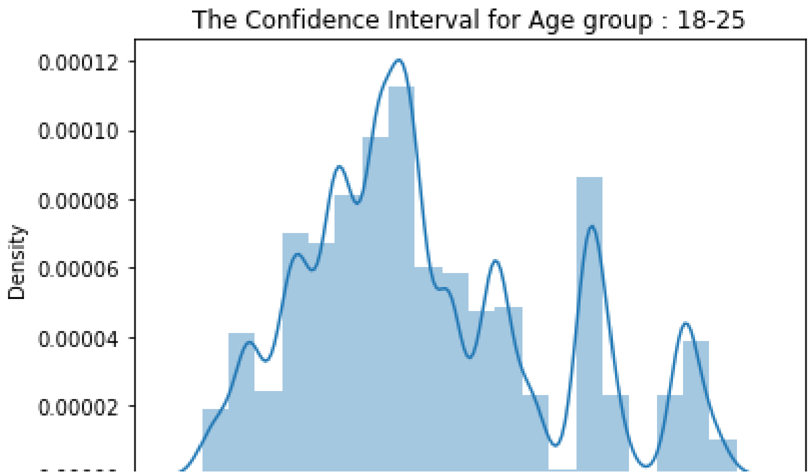
```
age_list =['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+']
for i in age_list:
    plt.title(f"The Confidence Interval for Age group : {i}")
    age = [Bootstrapping_CLT_CI(df.loc[df["Age"]== i ]["Purchase"], confidence=95 , sar
    plt.show()
```



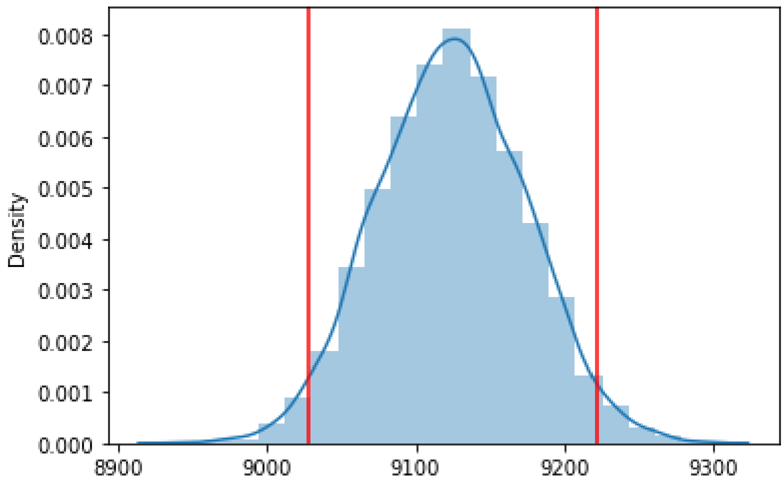

```
t: 1.9602012636213575
sample mean : 8867.20971848
sample standard deviation : 50.270013192320725
sample size:  10000
Margin of Error : 98.53934338184939
```



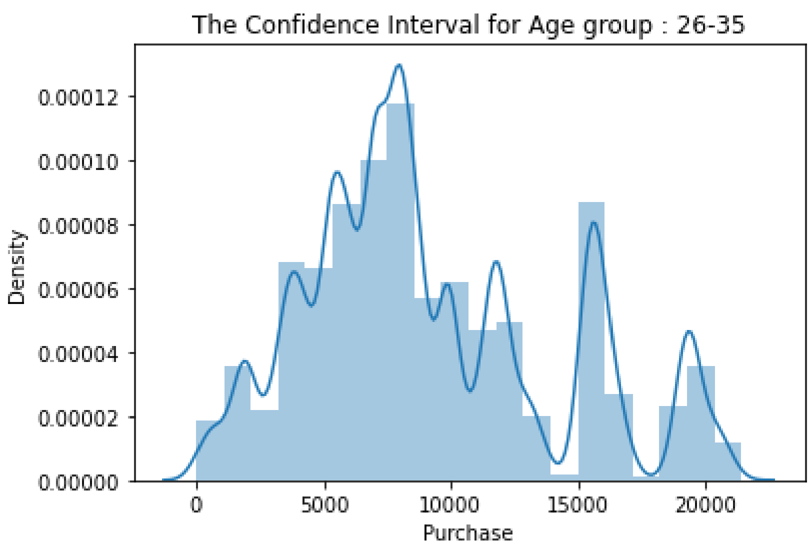

```
Confidence Interval :  (8768.67037509815, 8965.74906186185)
```



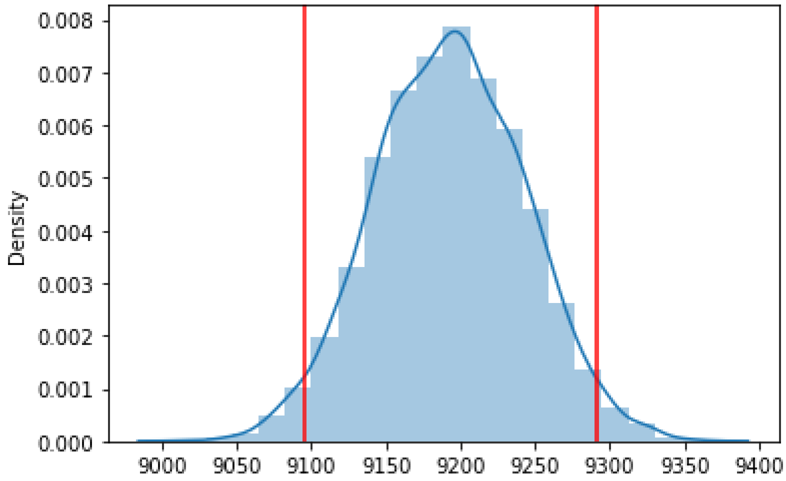

```
t: 1.9602012636213575
sample mean : 9124.36031518
sample standard deviation : 49.6166869653139
sample size:  10000
Margin of Error : 97.25869248611365
```



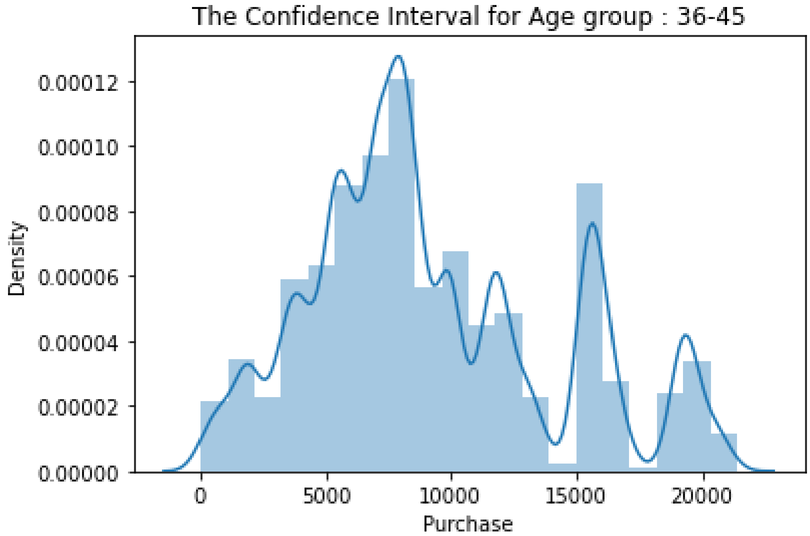

```
Confidence Interval :  (9027.101622693886, 9221.619007666113)
```



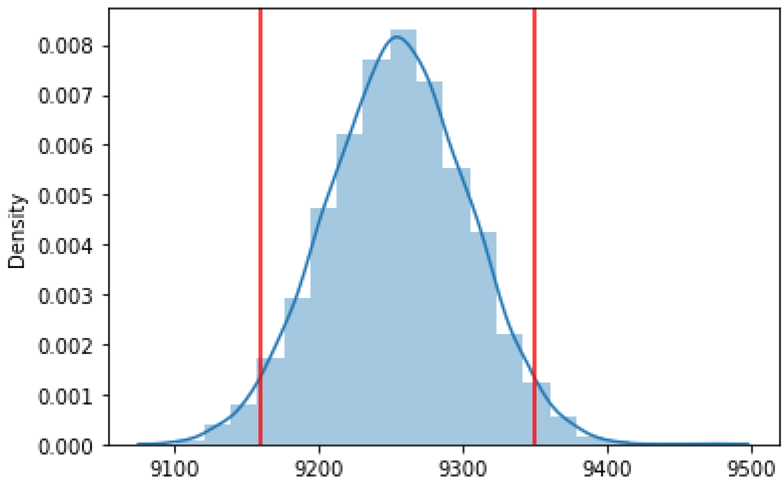

```
t: 1.9602012636213575
sample mean : 9192.905945980001
sample standard deviation : 49.927992694922175
sample size:  10000
Margin of Error : 97.86891437066436
```



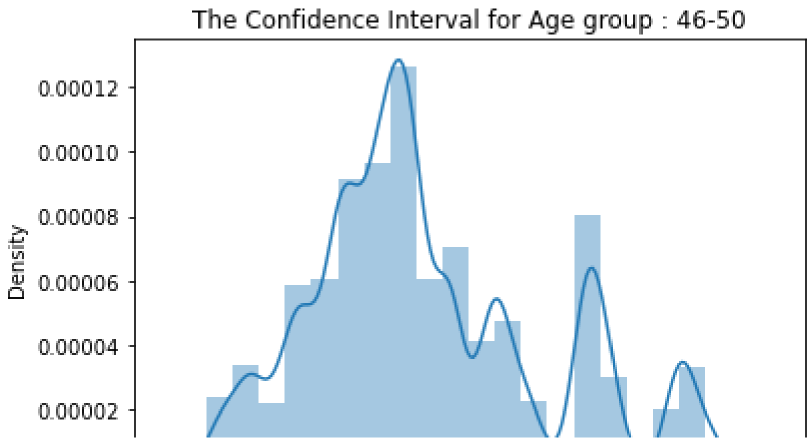

```
Confidence Interval :  (9095.037031609336, 9290.774860350666)
```



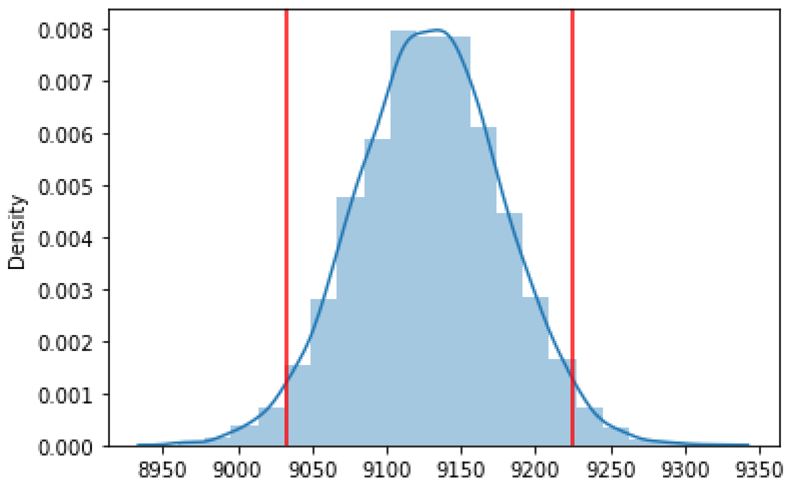

```
t: 1.9602012636213575
sample mean : 9254.5332938
sample standard deviation : 48.83098855740848
sample size:  10000
Margin of Error : 95.71856547411215
```



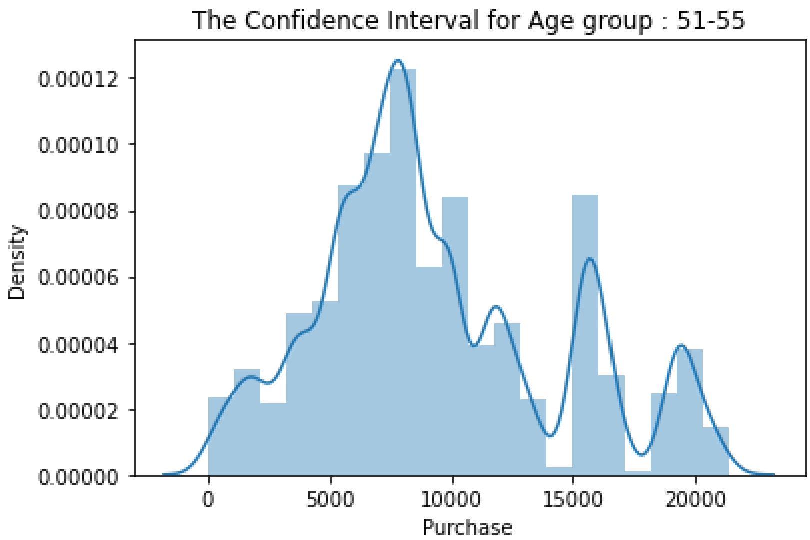

```
Confidence Interval :  (9158.814728325888, 9350.25185927411)
```



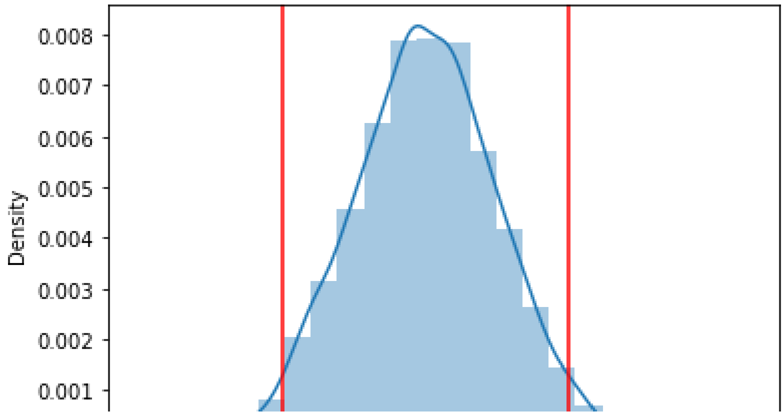

```
t: 1.9602012636213575
sample mean : 9129.028274600001
sample standard deviation : 48.75767916210386
sample size:  10000
Margin of Error : 95.57486430480073
```



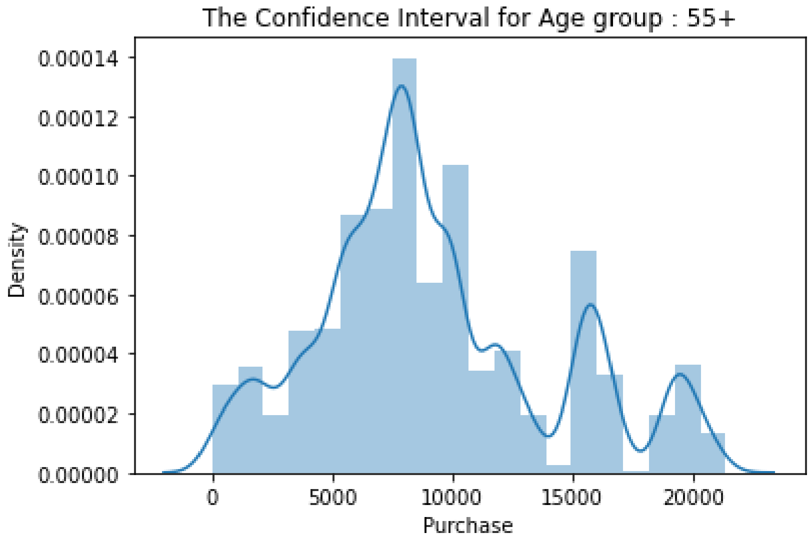

```
Confidence Interval :  (9033.4534102952, 9224.603138904802)
```



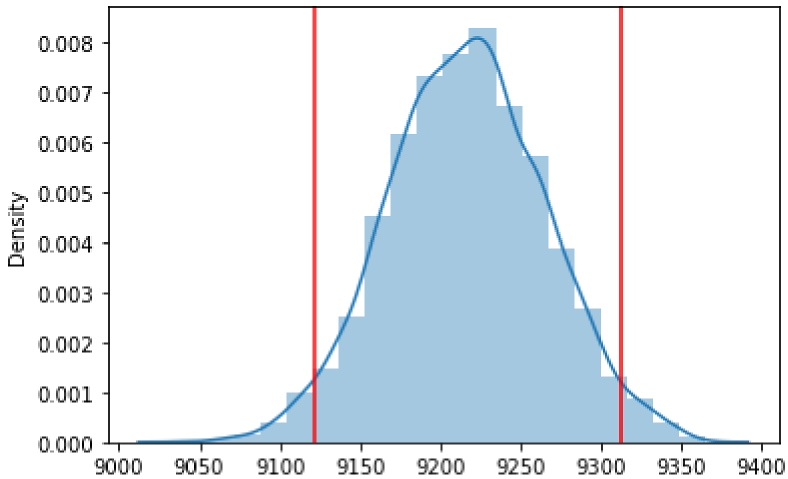

```
t: 1.9602012636213575
sample mean : 9423.315034180001
sample standard deviation : 49.05437150588185
sample size:  10000
Margin of Error : 96.15644101198112
```




```
Confidence Interval :  (9327.15859316802, 9519.471475191982)
```




```
t: 1.9602012636213575
sample mean : 9216.44916162
sample standard deviation : 48.65474117812937
sample size:  10000
Margin of Error : 95.37308513853928
```

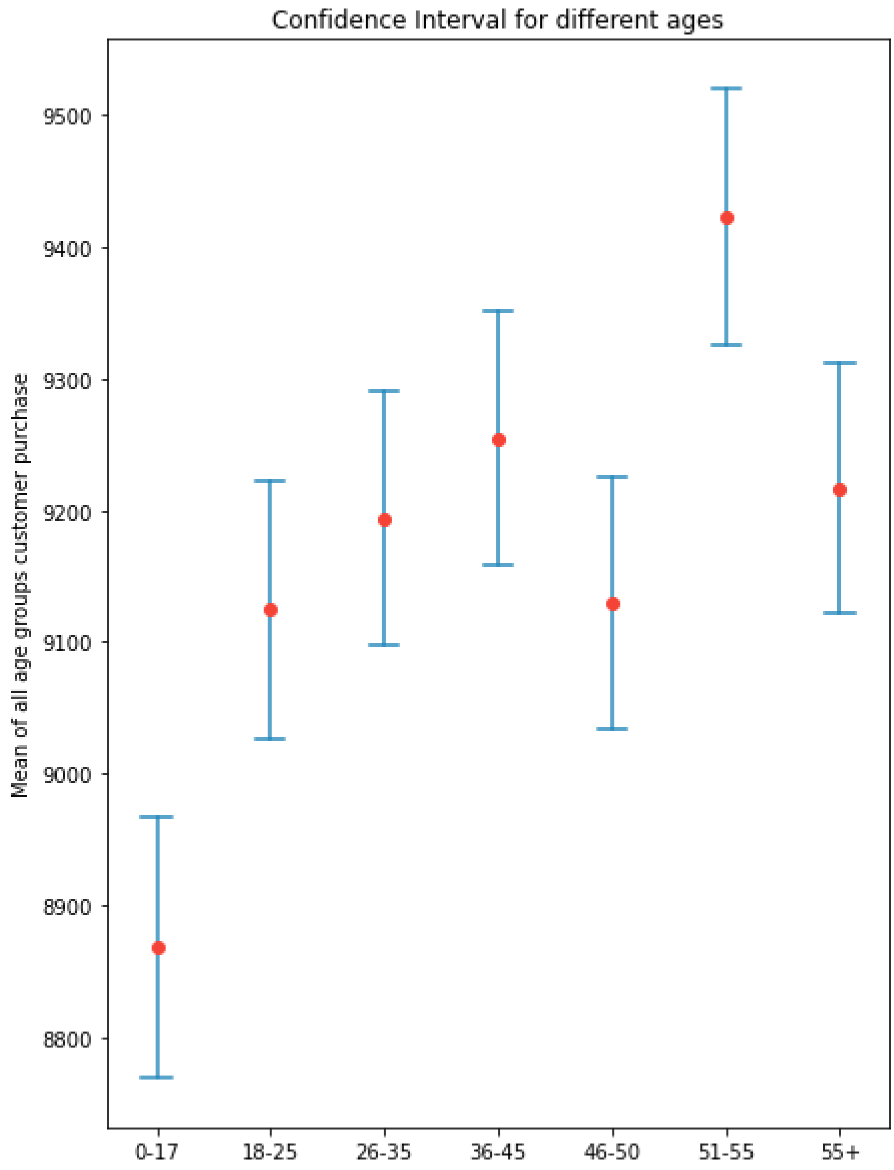



```
Confidence Interval :  (9121.07607648146, 9311.822246758538)
```

In [55]:

```
plt.figure(figsize = (7,10))
i = 1
for age_group in ['0-17' , '18-25' , '26-35' , '36-45' , '46-50' , '51-55' , '55+']:
    print('Age Group of :' , age_group)
    (plot_confidence_interval(i , df.loc[df['Age']== age_group]['Purchase']))
    i += 1
plt.xticks([1,2,3,4,5,6,7] , ['0-17' , '18-25' , '26-35' , '36-45' , '46-50' , '51-55'

plt.title('Confidence Interval for different ages')
plt.ylabel('Mean of all age groups customer purchase')
plt.show()
plt.show()
```

```
Age Group of : 0-17
Confidence Interval :  (8768.851166551302, 8966.042926051146)
Sample Mean : 8867.447046301224  and  Margin of Error : 98.59587974992306
Age Group of : 18-25
Confidence Interval :  (9026.437113190344, 9221.62634947098)
Sample Mean : 9124.031731330662  and  Margin of Error : 97.59461814031708
Age Group of : 26-35
Confidence Interval :  (9096.686954194787, 9290.25289333175)
Sample Mean : 9193.469923763269  and  Margin of Error : 96.78296956848119
Age Group of : 36-45
Confidence Interval :  (9157.608946498, 9350.795480925703)
Sample Mean : 9254.202213711851  and  Margin of Error : 96.59326721385233
Age Group of : 46-50
Confidence Interval :  (9033.575019923457, 9224.395139840639)
Sample Mean : 9128.985079882048  and  Margin of Error : 95.410059958591
Age Group of : 51-55
Confidence Interval :  (9326.02157031302, 9520.221837819787)
Sample Mean : 9423.121704066403  and  Margin of Error : 97.1001337533831
Age Group of : 55+
Confidence Interval :  (9121.3547892136, 9311.945651645607)
Sample Mean : 9216.650220429603  and  Margin of Error : 95.2954312160032
```



Observation

- Spending by Age_group 0-17 is low compared to other age groups at [8768.851, 8966.042]
- Customers in Age_group 51-55 spend the most between [9326.02, 9520.22]

In [67]:

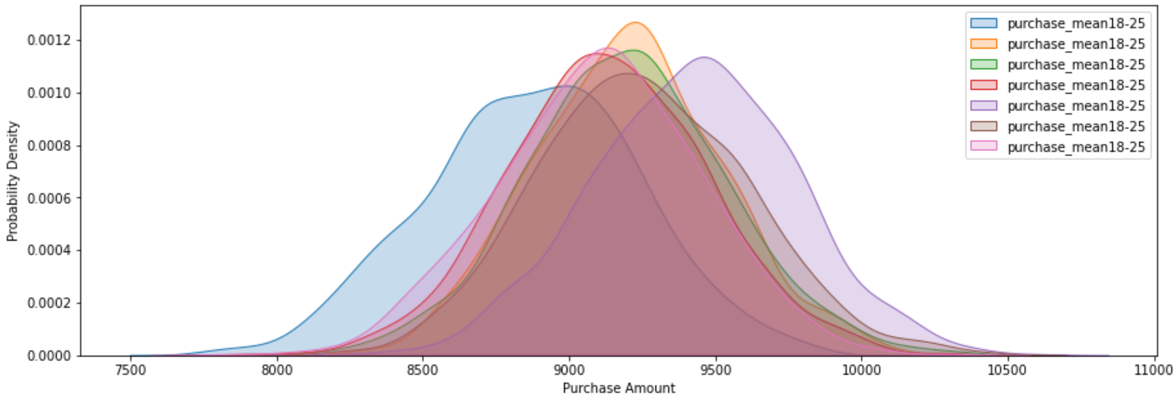
```
age_dict = {}
for i in df['Age'].unique():
    x = "purchase_mean"+i
    age_dict[x] = [df[df['Age'] == i]['Purchase'].sample(200).mean() for j in range(10(

```

In [ ]:

```

```

In [68]:

```
plt.figure(figsize = (15,5))
for i in age_dict.keys():
    sns.kdeplot(age_dict[i], shade = True, label = x)
plt.legend()
plt.xlabel('Purchase Amount')
plt.ylabel('Probability Density')
plt.show()
```



In [ ]:

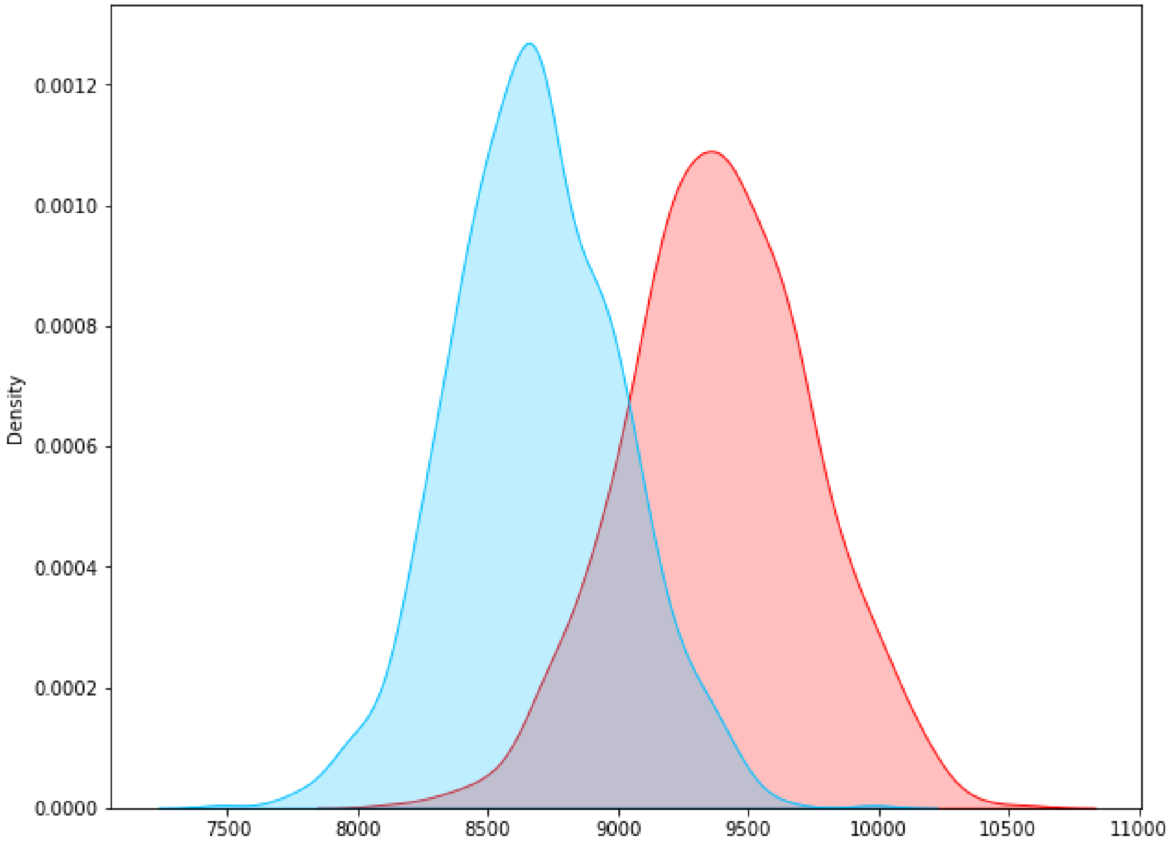
```

```

In [69]:

```
male_data = [df[df['Gender'] == 'Male']['Purchase'].sample(200).mean() for j in range(1
Female_data = [df[df['Gender'] == 'Female']['Purchase'].sample(200).mean() for j in ran


```

In [70]:

```
plt.figure(figsize=(10,8))
sns.kdeplot(male_data,shade=True,color='red')
sns.kdeplot(Female_data,shade=True,color='deepskyblue')
plt.show()
```



In [ ]:

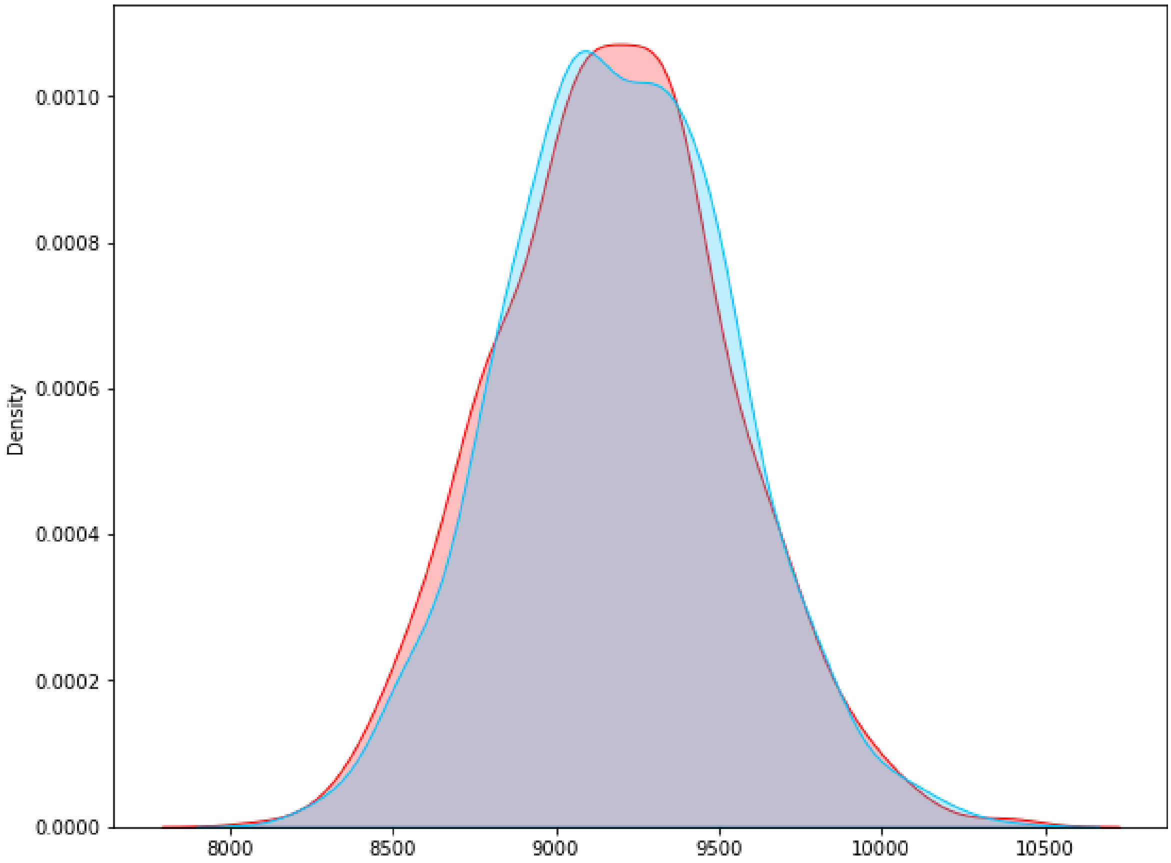
```

```

In [71]:

```
married_data = [df[df['Marital_Status'] == 'Married']['Purchase'].sample(200).mean() fo
single_data = [df[df['Marital_Status'] == 'Single']['Purchase'].sample(200).mean() for
```

In [73]:

```
plt.figure(figsize=(10,8))
sns.kdeplot(married_data,shade=True,color='red')
sns.kdeplot(single_data,shade=True,color='deepskyblue')
plt.show()
```



Observation

- There's no spending behavioral chnage in married and unmarried people in spending habits.
- The age groups of people buying, we can see a huge overlap between them w.r.t purchasing power.
- There is considerable amount of difference in the purchasing power of Male and Female customers.

CLT WITHOUT BOOTSTRAPPING (just for reference)

In [59]:

```
def Confi_Inter(data , val):
    x = val
    sample_size = len(data)
    mean = round(np.mean(data),3)
    standard_deviation = np.std(data)

    print(f"Sampling Distribution with sample size = {len(data)}")
    print(f"Sampling Distribution with mean = {round(np.mean(data),3)}")
    print(f"Sampling Distribution with standard deviation = {round(np.std(data),3)}")
    val = stats.t.ppf(1-(1-val/100)/2,sample_size - 1)
    print(f"Zcritical = {round(val , 3)}")

    CI_upper = mean + ((val *  standard_deviation)/ np.sqrt(sample_size))
    CI_lower = mean - ((val *  standard_deviation)/ np.sqrt(sample_size))

    print(f"So at {x}% confidence the value of the population mean falls within the rar
```

In [ ]:

```

```

CLT FOR MARRIED

In [60]:

```
Confi_Inter(df.loc[df['Marital_Status'] == 'Single']['Purchase'] , 95)
```

```
Sampling Distribution with sample size = 323242
Sampling Distribution with mean = 9201.582
Sampling Distribution with standard deviation = 4948.32
Zcritical = 1.96
So at 95% confidence the value of the population mean falls within the range
9184.523 and 9218.641
```

In [61]:

```
Confi_Inter(df.loc[df['Marital_Status'] == 'Married']['Purchase'] , 95)
```

```
Sampling Distribution with sample size = 224149
Sampling Distribution with mean = 9187.04
Sampling Distribution with standard deviation = 4925.194
Zcritical = 1.96
So at 95% confidence the value of the population mean falls within the range
9166.651 and 9207.429
```

CLT FOR GENDER`

In [62]:

```
Confi_Inter(df.loc[df['Gender'] == 'Male']['Purchase'] , 95)
```

```
Sampling Distribution with sample size = 412171
Sampling Distribution with mean = 9367.724
Sampling Distribution with standard deviation = 5009.228
Zcritical = 1.96
So at 95% confidence the value of the population mean falls within the range
9352.431 and 9383.017
```

In [ ]:

```

```

Inferences & Recommendations

Based On EDA

- The majority of the cutomers of the given sample are male's (75 percentage) compared to Female's
- Majority of cutomers comes from city B but more money is spend by cutomers from city C
- There are more Single's than Married but the behavioral power is very similar.
- Majority of cutomers purchase in the range of 60,000 to 20,000
- The purchasing power of Males is arounf a 700 dollor more than Female. We also need to take into consideration the more often than not large amount purchases by men (visible in the outliers in male data)
- The purchasing power of Married and Unmarried are not very differnt. Only difference of 4 dollors in their mean purchase from the data.
- The purchasing power in differnt age groups shows us that , age group 17 - 25 has the lowest purchasing power with the lowest count as well.
- The purchasing power of age group 51 - 55 is the most even when their count is not that high.
- The purchasing power of age group 26 - 35 stands out with the number of purchases and their not so less average compared to senior citizens

Based on Statistical Analysis (using CLT & CI)

- 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.
- Overlapping is not that evident in case of Male and Female customers even with large samples which showes the purchasing power of male customers are more than the females with male 9271.34, 9463.11 and female - 8578.80 , 8762.71
- When it comes to Age groups - the pending by Age_group 0-17 is low compared to other age groups. Customers in Age_group 51-55 spend the most between 9326.021, 9520.221

Recommendations

- Since Majority of customers are from City B , the quality of products and attractive offers cshould be improved in this city because the overall purchase mean is higher on city C.
- City A has lesser purchasing power and people meaning they are more often moving or travelling customers so more infrastructure and marketing strategies can be focused on city A

- Since there is no difference between the married and single catagories no special consideration of=r changes needs to be taken in that line. Whatever is in place seems to be working perfectly.
- There needs to be a attention defecit in the Gender as the desparity in purchasing power between Male and Female is fairly huge with a diffence of 700 dollors on average. Measures to address these can be taken like - MOntly period day offer , Womens day offers , Single mother offers
- Looking at the Age Groups - purchasing power in differnt age groups shows us that , age group 17 - 25 has the lowest purchasing power with the lowest count as well - This can be improved with more teenage products and University dicounts to improve the counts of purchase as these catagory might not be earning much.
- The purchasing power of age group 51 - 55 is the most even when their count is not that high - this can be improved by making the infrastructure more age friendly and bringing veteran discounts and celebration days that attract this age groups, as these are the catagories with maximum savings who likes to spend of family and grand-children.
-

In [ ]:

```

```