In [1]:

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
```

In [2]:

```
os.chdir('C:\\Users\Ashok kumar\Desktop\chanu\DSML_Course\DataSet')
```

In [4]:

```
#Loading dataset
df = pd.read_csv('walmart_data.csv')
```

In [5]:

```
df.head()
```

Out[5]:

| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years |
|---|---|---|---|---|---|---|---|
| **0** | 1000001 | P00069042 | F | 0-17 | 10 | A | 2 |
| **1** | 1000001 | P00248942 | F | 0-17 | 10 | A | 2 |
| **2** | 1000001 | P00087842 | F | 0-17 | 10 | A | 2 |
| **3** | 1000001 | P00085442 | F | 0-17 | 10 | A | 2 |
| **4** | 1000002 | P00285442 | M | 55+ | 16 | C | 4+ |

In [6]:

```
df.shape
```

Out[6]:

```
(550068, 10)
```

In [7]:

```
df.dtypes
```

Out[7]:

```
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
```

In [8]:

```
#unique columns in each column
df.nunique().sort_values()
```

Out[8]:

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

In [9]:

```
df.isnull().sum()
```

Out[9]:

```
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
```

No Null values in dataset

In [10]:

```
df.Gender.value_counts()
```

Out[10]:

```
M    414259
F    135809
Name: Gender, dtype: int64
```

In [11]:

```
df.Marital_Status=df.Marital_Status.astype('category')
```

In [12]:

```
df.dtypes
```

Out[12]:

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

In [13]:

```
df.Marital_Status.value_counts()
```

Out[13]:

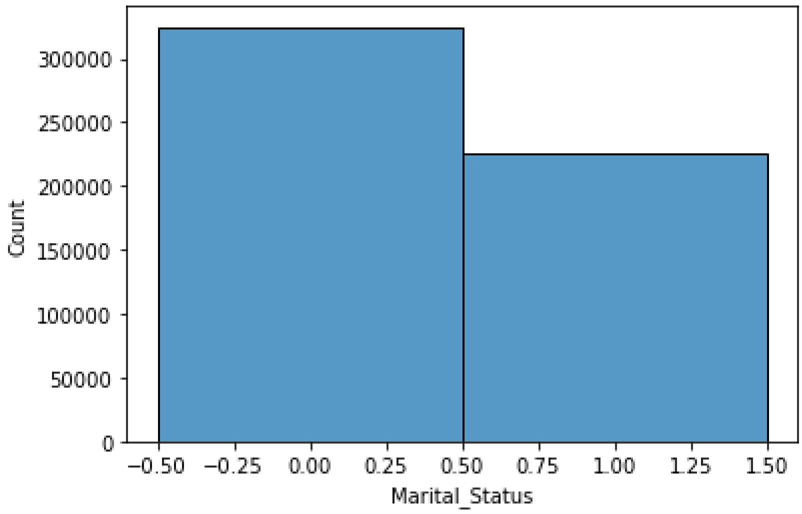
```
0    324731
1    225337
Name: Marital_Status, dtype: int64
```

In [14]:

```
sns.histplot(x=df.Marital_Status)
```

Out[14]:

```
<AxesSubplot:xlabel='Marital_Status', ylabel='Count'>
```




In [15]:

```
df.groupby([df.Gender,df.Marital_Status]).Marital_Status.count().sort_values(ascending=Fals
```

Out[15]:

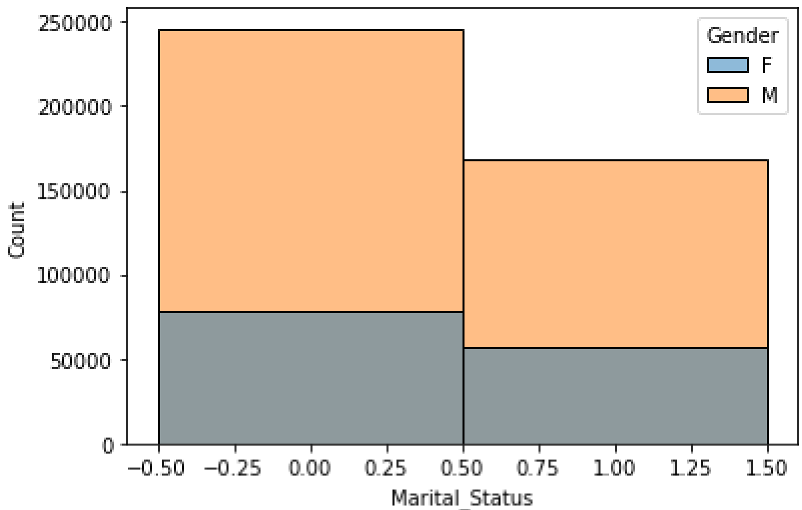
```
Gender  Marital_Status
M       0                 245910
        1                 168349
F       0                  78821
        1                  56988
Name: Marital_Status, dtype: int64
```

In [24]:

```
sns.histplot(x=df.Marital_Status,hue=df.Gender)
```

Out[24]:

```
<AxesSubplot:xlabel='Marital_Status', ylabel='Count'>
```



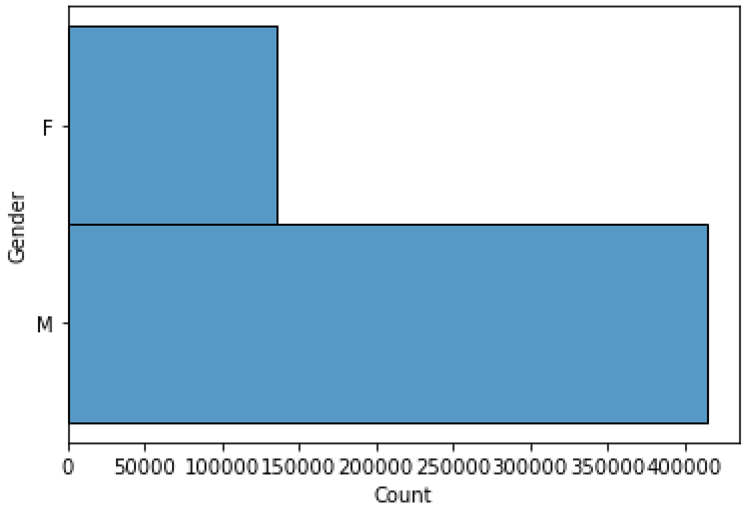
Male customers are more compared to female customers among married and single

In [44]:

```
# plt.figure(figsize=(13,7))
sns.histplot(y=df.Gender)
```

Out[44]:

```
<AxesSubplot:xlabel='Count', ylabel='Gender'>
```




In [35]:

```
df.loc[df.Gender == 'F'].Age.value_counts()
```

Out[35]:

```
26-35    50752
36-45    27170
18-25    24628
46-50    13199
51-55     9894
0-17      5083
55+       5083
Name: Age, dtype: int64
```

In [37]:

```
df.loc[df.Gender == 'M'].Age.value_counts()
```

Out[37]:

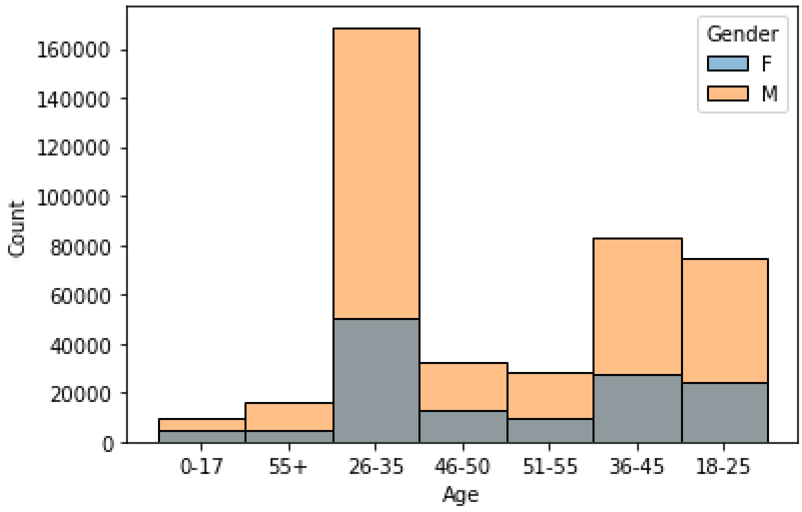
```
26-35    168835
36-45     82843
18-25     75032
46-50     32502
51-55     28607
55+       16421
0-17      10019
Name: Age, dtype: int64
```

In [42]:

```
sns.histplot(x=df.Age,hue=df.Gender)
```

Out[42]:

```
<AxesSubplot:xlabel='Age', ylabel='Count'>
```




Age Group with 26-35 buys more during the blackfriday sale

In [47]:

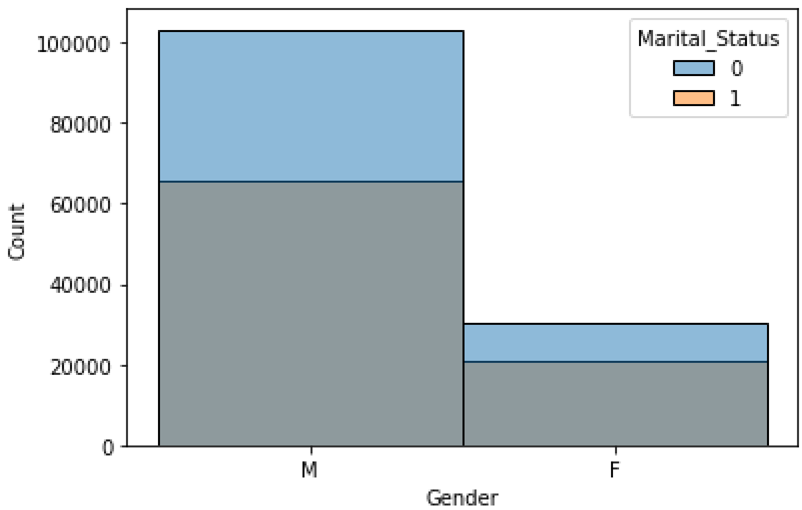
```
d1=df.loc[df.Age== '26-35']
```

In [48]:

```
sns.histplot(x=d1.Gender,hue=d1.Marital_Status)
```

Out[48]:

```
<AxesSubplot:xlabel='Gender', ylabel='Count'>
```




In the age group of 26-35 unmarried people are buying more products

***1 - Unmarried 0 - Married***

In [59]:

```
df.Occupation.value_counts().sort_values(ascending=False)
```

Out[59]:

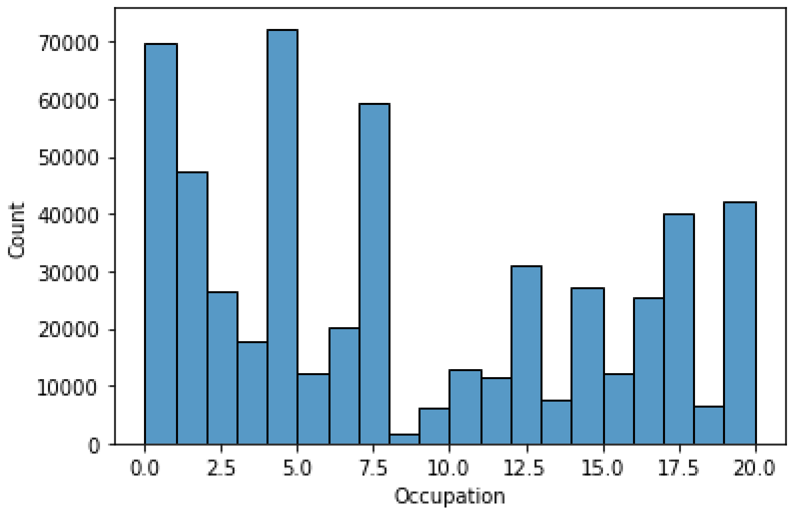
```
4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
2     26588
16    25371
6     20355
3     17650
10    12930
5     12177
15    12165
11    11586
19     8461
13     7728
18     6622
9      6291
8      1546
Name: Occupation, dtype: int64
```

In [57]:

```
sns.histplot(df.Occupation,bins=np.arange(21))
```

Out[57]:

<AxesSubplot:xlabel='Occupation', ylabel='Count'>




In [60]:

```
#we can see that 4,0,7 occupational customers are more contributors than other occupational
```

In [63]:

```
df.City_Category.value_counts().sort_values(ascending=False)
```

Out[63]:

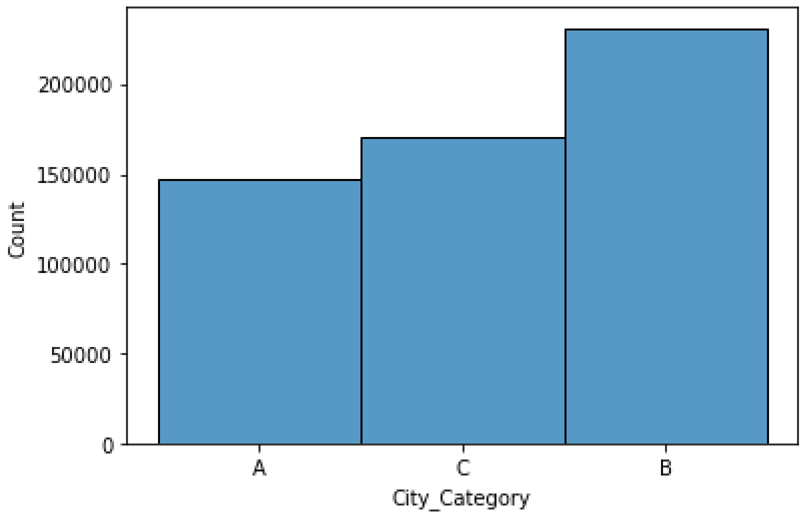
```
B    231173
C    171175
A    147720
Name: City_Category, dtype: int64
```

In [64]:

```
sns.histplot(x=df.City_Category)
```

Out[64]:

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




In [69]:

```
df.Stay_In_Current_City_Years.value_counts()
```

Out[69]:

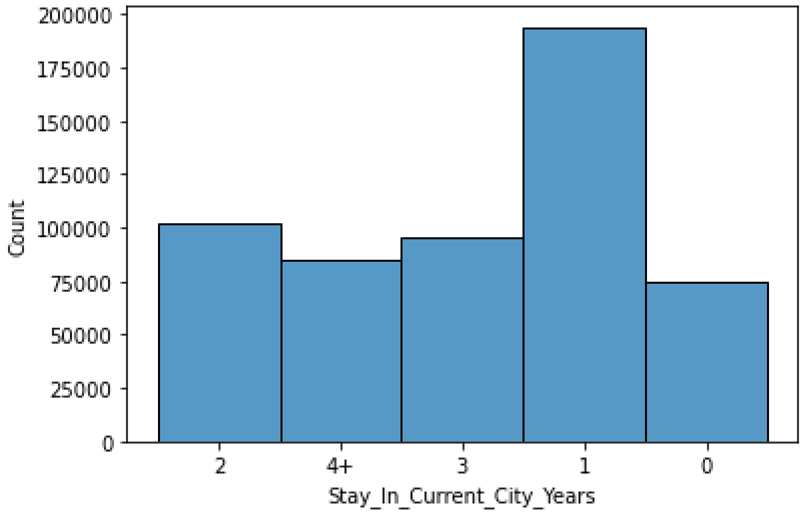
```
1     193821
2     101838
3      95285
4+     84726
0      74398
Name: Stay_In_Current_City_Years, dtype: int64
```

In [70]:

```
sns.histplot(df.Stay_In_Current_City_Years)
```

Out[70]:

```
<AxesSubplot:xlabel='Stay_In_Current_City_Years', ylabel='Count'>
```

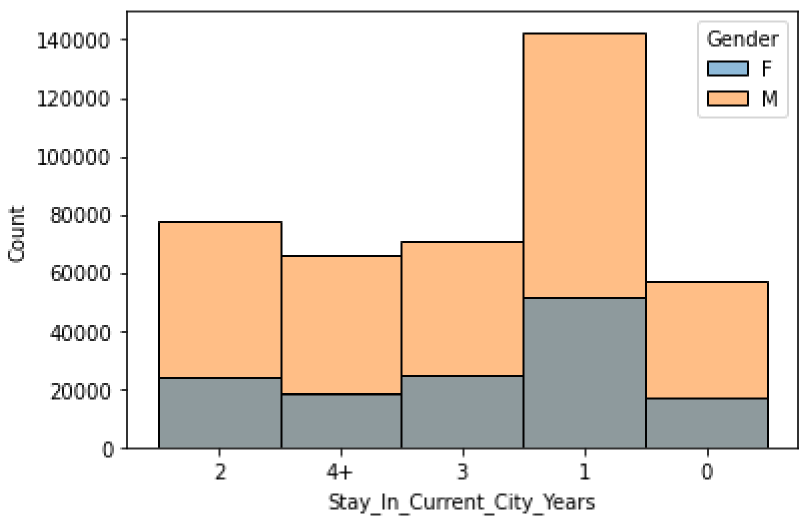



In [72]:

```
sns.histplot(x=df.Stay_In_Current_City_Years,hue=df.Gender)
```

Out[72]:

```
<AxesSubplot:xlabel='Stay_In_Current_City_Years', ylabel='Count'>
```



In [74]:

```
#from this we can say that people who comes to the city in first year buys more things to t
#to setup their house
```

In [95]:

```
d1=pd.DataFrame(df.groupby([df.City_Category,df.Stay_In_Current_City_Years]).Occupation.cou
d1.rename(columns={'Occupation':'Count'},inplace=True)
d1.sort_values(by=['Count'],ascending=False)
```

Out[95]:

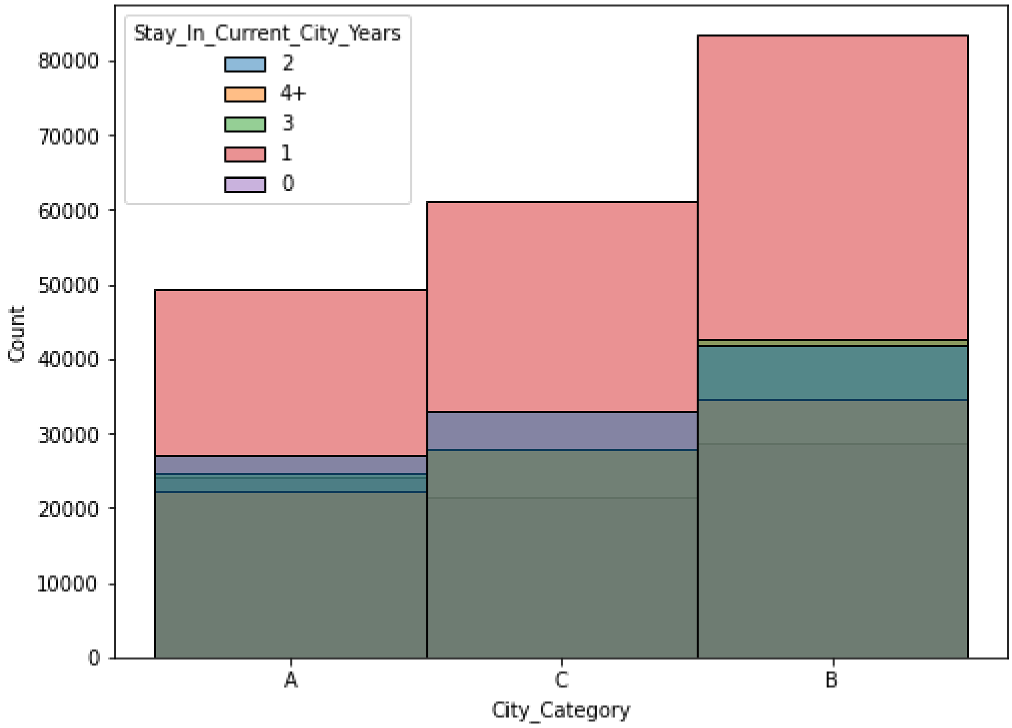
| | City_Category | Stay_In_Current_City_Years | Count |
|---|---|---|---|
| **6** | B | 1 | 83413 |
| **11** | C | 1 | 61103 |
| **1** | A | 1 | 49305 |
| **8** | B | 3 | 42691 |
| **7** | B | 2 | 41772 |
| **9** | B | 4+ | 34610 |
| **12** | C | 2 | 32952 |
| **5** | B | 0 | 28687 |
| **14** | C | 4+ | 27797 |
| **13** | C | 3 | 27790 |
| **2** | A | 2 | 27114 |
| **3** | A | 3 | 24804 |
| **0** | A | 0 | 24178 |
| **4** | A | 4+ | 22319 |
| **10** | C | 0 | 21533 |

In [76]:

```
plt.figure(figsize=(8,6))
sns.histplot(x=df.City_Category,hue=df.Stay_In_Current_City_Years)
```

Out[76]:

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




In [96]:

```
#we can observe that B is the most popular city cateogry and customers who lived for 1 year
#all 3 city cateogries.
```

In [97]:

```
df_copy = df.copy()
```

In [102]:

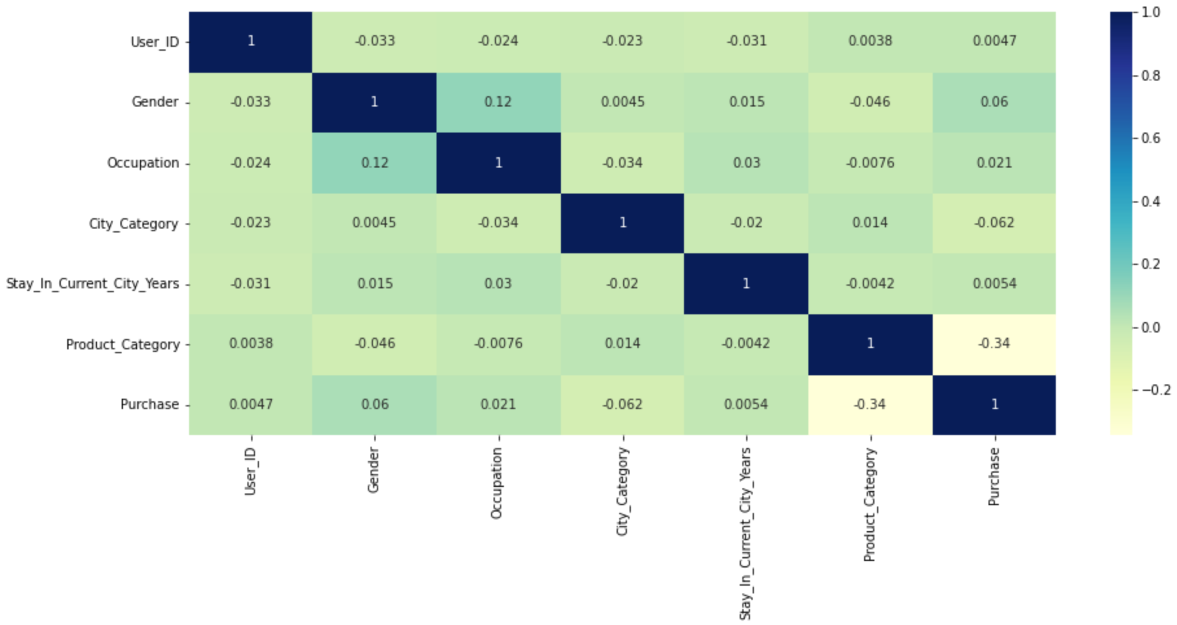
```
df_copy['Gender'].replace(['M', 'F'], [1, 0], inplace=True)
df_copy['City_Category'].replace(['A', 'B', 'C'],[2,1,0],inplace=True)
df_copy.Marital_Status=df_copy.Marital_Status.astype('category')
df_copy['Stay_In_Current_City_Years'].replace(['2', '4+', '3', '1', '0'],[2, 4, 3, 1, 0],in
```

In [104]:

```
plt.figure(figsize=(15,6))
sns.heatmap(df_copy.corr(),cmap="YlGnBu",annot=True)
```

Out[104]:

```
<AxesSubplot:>
```




In [105]:

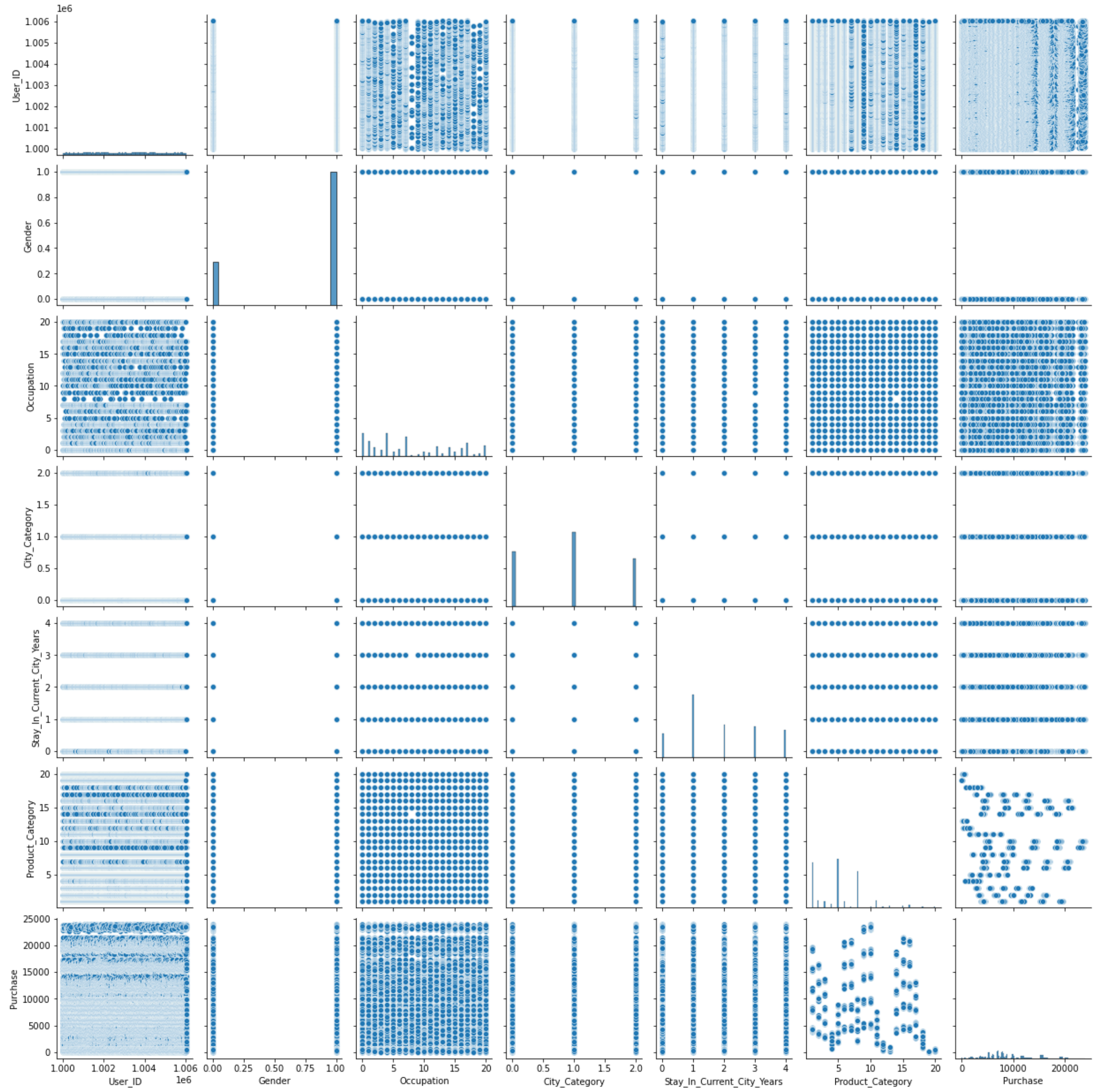
```
#we dont see much correlation between the features
```

In [106]:

```
sns.pairplot(df_copy)
```

Out[106]:

<seaborn.axisgrid.PairGrid at 0x1dd86c54340>



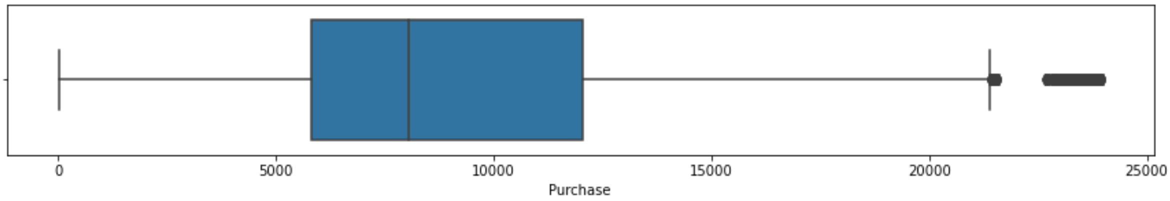
## Outliers

In [111]:

```
plt.figure(figsize=(15,2))
sns.boxplot(x=df.Purchase)
```

Out[111]:

<AxesSubplot:xlabel='Purchase'>



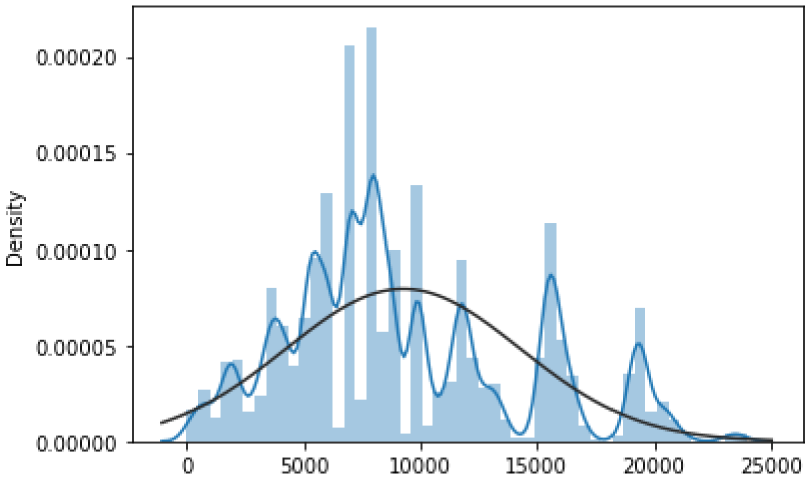

In [120]:

```
sns.distplot(x=df.Purchase,fit=stats.norm)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[120]:

<AxesSubplot:ylabel='Density'>




In [163]:

```
df.loc[df['Gender'] == 'M']['Purchase'].sum()/df.loc[df['Gender'] == 'M'].size
```

Out[163]:

943.7526040472264

In [164]:

```
df.loc[df['Gender'] == 'F']['Purchase'].sum()/df.loc[df['Gender'] == 'F'].size
```

Out[164]:

873.4565765155476

In [165]:

```
#Male Spends more than female
```

In [121]:

```
df['Purchase'].mean()
```

Out[121]:

9263.968712959126

In [122]:

```
#overall mean is 9263.968712959126
```

In [125]:

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

Out[125]:

9437.526040472265

In [126]:

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

Out[126]:

8734.565765155476

In [130]:

```
#Male purchases more than female
```

In [134]:

```
df.loc[(df['Marital_Status'] == 0) & (df['Gender']=='F')]['Purchase'].mean()
```

Out[134]:

8679.845815201532

In [135]:

```
df.loc[(df['Marital_Status'] == 0) & (df['Gender']=='M')]['Purchase'].mean()
```

Out[135]:

9453.75674027083

In [136]:

```
df.loc[(df['Marital_Status'] == 1) & (df['Gender']=='F')]['Purchase'].mean()
```

Out[136]:

```
8810.249789429354
```

In [137]:

```
df.loc[(df['Marital_Status'] == 1) & (df['Gender']=='M')]['Purchase'].mean()
```

Out[137]:

```
9413.81760509418
```

In [138]:

```
#From above we can say that Male customers spend almost same even after marriage.
# Unmarried Females spends more compared to married female's
```

In [139]:

```
df.loc[(df['Marital_Status'] == 0)]['Purchase'].mean()
```

Out[139]:

```
9265.907618921507
```

In [140]:

```
df.loc[(df['Marital_Status'] == 1)]['Purchase'].mean()
```

Out[140]:

```
9261.174574082374
```

In [141]:

```
#From Above we can say that married and unmarried customers spend same amount.
```

In [160]:

```
lis = [1,2,3,'4+']
means=[]
for i in range(len(lis)):
    x=df.loc[(df['Stay_In_Current_City_Years'] == str(lis[i]))]['Purchase'].mean()
    means.append(x)
stay_in_country_means = {stay:mean for stay,mean in zip(lis,means)}
stay_in_country_means
```

Out[160]:

```
{1: 9250.145923300364,
 2: 9320.429810090536,
 3: 9286.904119221284,
 '4+': 9275.59887165687}
```

In [161]:

```
lis = [1,2,3,'4+']
means=[]
for i in range(len(lis)):
    x=df.loc[(df['Stay_In_Current_City_Years'] == str(lis[i])) & (df['Marital_Status']== 0)
    means.append(x)
stay_in_country_means = {stay:mean for stay,mean in zip(lis,means)}
stay_in_country_means
```

Out[161]:

```
{1: 9241.859769097347,
 2: 9307.26052631579,
 3: 9362.867434558606,
 '4+': 9217.201657458563}
```

In [162]:

```
lis = [1,2,3,'4+']
means=[]
for i in range(len(lis)):
    x=df.loc[(df['Stay_In_Current_City_Years'] == str(lis[i])) & (df['Marital_Status']== 1)
    means.append(x)
stay_in_country_means = {stay:mean for stay,mean in zip(lis,means)}
stay_in_country_means
```

Out[162]:

```
{1: 9261.180439097745,
 2: 9339.940810955699,
 3: 9170.557415378076,
 '4+': 9362.527462844388}
```

In [177]:

```
age_bin = ['0-17', '18-25', '26-35', '36-45','46-50', '51-55',  '55+']
means=[]
for i in range(len(age_bin)):
    x=df.loc[(df['Age'] == age_bin[i])]['Purchase'].mean()
    means.append(x)
Age_mean = {Age:mean for Age,mean in zip(age_bin,means)}
Age_mean
```

Out[177]:

```
{'0-17': 8933.464640444974,
 '18-25': 9169.663606261289,
 '26-35': 9252.690632869888,
 '36-45': 9331.350694917874,
 '46-50': 9208.625697468327,
 '51-55': 9534.808030960236,
 '55+': 9336.280459449405}
```

In [179]:

```
#customers with age 51-55 purchases more compared to other age groups.
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

# CLT

# Gender

In [229]:

```
z_interval = stats.norm.interval(alpha = [0.90,0.95,0.99])
z_interval
```

Out[229]:

```
(array([-1.64485363, -1.95996398, -2.5758293 ]),
 array([1.64485363, 1.95996398, 2.5758293 ]))
```

In [231]:

```
ci_90 = 1.64485363
ci_95 = 1.95996398
ci_99 = 2.5758293
```

In [381]:

```
def caluclate_ci(samp_mean,sigma,n):
    ci_90 = 1.64485363
    ci_95 = 1.95996398
    ci_99 = 2.5758293
    ci = [1.64485363,1.95996398,2.5758293]
    per = ['90%','95%','99%']
    for i in range(len(ci)):
        ul = samp_mean + (ci[i] * (sigma/(n)**(1/2)))
        ll = samp_mean - (ci[i] * (sigma/(n)**(1/2)))
        print('We can be confident that {0} of customers will spend in range of {1},{2}'.fo
```

### Sample Size 1000

### Male CI

In [196]:

```
samples = 1000
repetetion = 1000
sample_mean_1 = []
for rep in range(repetetion):
    x = df.loc[df['Gender'] == 'M']['Purchase'].sample(samples).mean()
    sample_mean_1.append(x)
np.mean(sample_mean_1)
```

Out[196]:

```
9431.573809
```

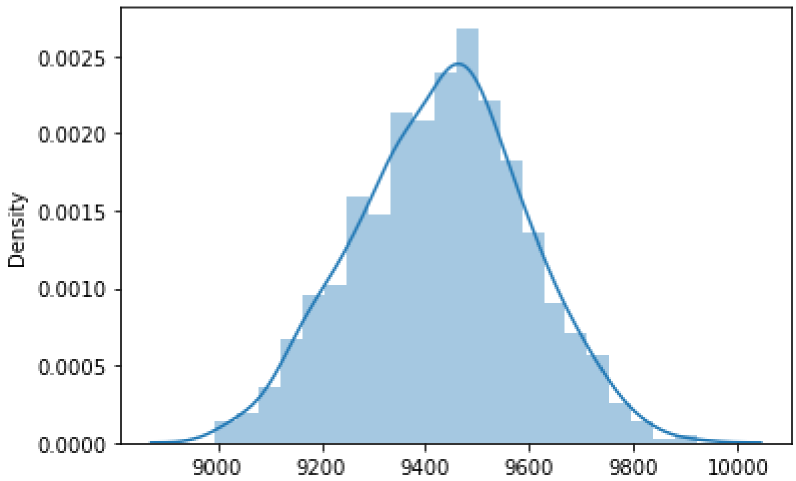
In [206]:

```
sns.distplot(sample_mean_1)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[206]:

```
<AxesSubplot:ylabel='Density'>
```




In [382]:

```
caluclate_ci(np.mean(sample_mean_1),np.std(sample_mean_1),1000)
```

```
We can be confident that 90% of customers will spend in range of 9423.133436
309066,9440.014181690933
We can be confident that 95% of customers will spend in range of 9421.516484
658014,9441.631133341985
We can be confident that 99% of customers will spend in range of 9418.356244
297769,9444.79137370223
```

#### Female CI

In [198]:

```
samples = 1000
repetetion = 1000
sample_mean_2 = []
for rep in range(repetetion):
    x = df.loc[df['Gender'] == 'F']['Purchase'].sample(samples).mean()
    sample_mean_2.append(x)
np.mean(sample_mean_2)
```

Out[198]:

```
8731.657868
```

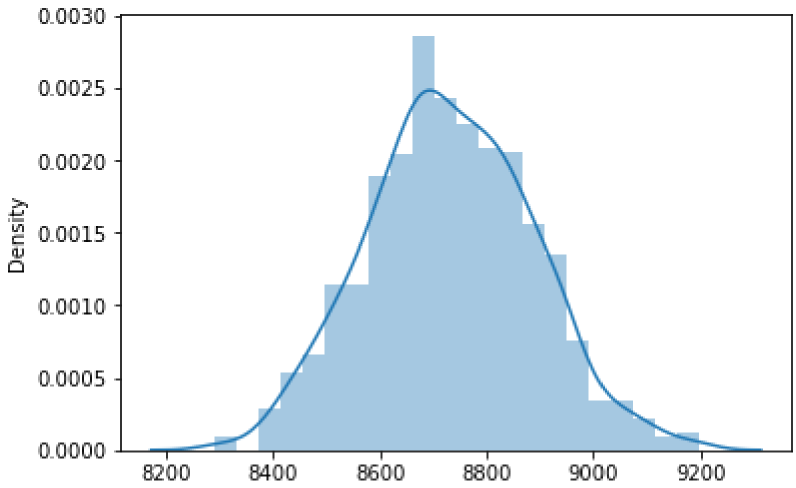
In [205]:

```
sns.distplot(sample_mean_2)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[205]:

```
<AxesSubplot:ylabel='Density'>
```




In [383]:

```
caluclate_ci(np.mean(sample_mean_2),np.std(sample_mean_2),1000)
```

```
We can be confident that 90% of customers will spend in range of 8723.594809
47706,8739.720926522941
We can be confident that 95% of customers will spend in range of 8722.050141
217895,8741.265594782106
We can be confident that 99% of customers will spend in range of 8719.031174
650388,8744.284561349612
```

In [ ]:

```

```

**sample size 10000**

**Male CI**

In [200]:

```
samples = 10000
repetetion = 1000
sample_mean_3 = []
for rep in range(repetetion):
    x = df.loc[df['Gender'] == 'M']['Purchase'].sample(samples).mean()
    sample_mean_3.append(x)
np.mean(sample_mean_3)
```

Out[200]:

```
9436.2865602
```

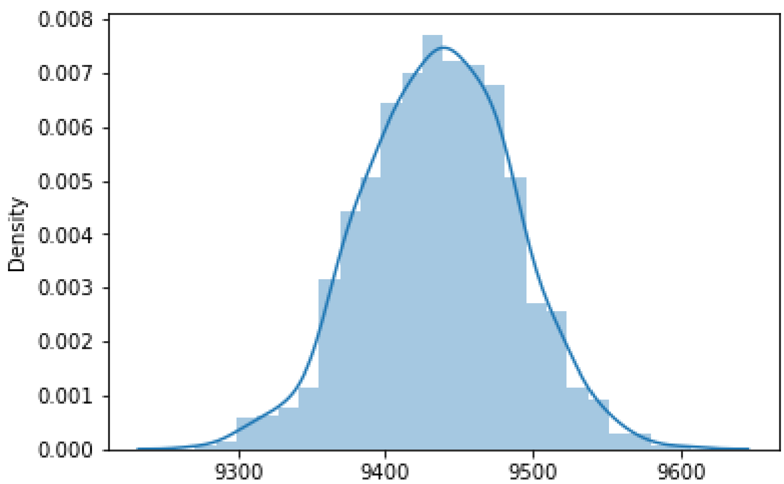
In [207]:

```
sns.distplot(sample_mean_3)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[207]:

```
<AxesSubplot:ylabel='Density'>
```




In [384]:

```
caluclate_ci(np.mean(sample_mean_3),np.std(sample_mean_3),10000)
```

```
We can be confident that 90% of customers will spend in range of 9435.452797
701077,9437.120322698924
We can be confident that 95% of customers will spend in range of 9435.293070
909547,9437.280049490453
We can be confident that 99% of customers will spend in range of 9434.980893
948123,9437.592226451878
```

In [ ]:

```

```

In [203]:

```
samples = 10000
repetetion = 1000
sample_mean_4 = []
for rep in range(repetetion):
    x = df.loc[df['Gender'] == 'F']['Purchase'].sample(samples).mean()
    sample_mean_4.append(x)
np.mean(sample_mean_4)
```

Out[203]:

```
8738.1874266
```

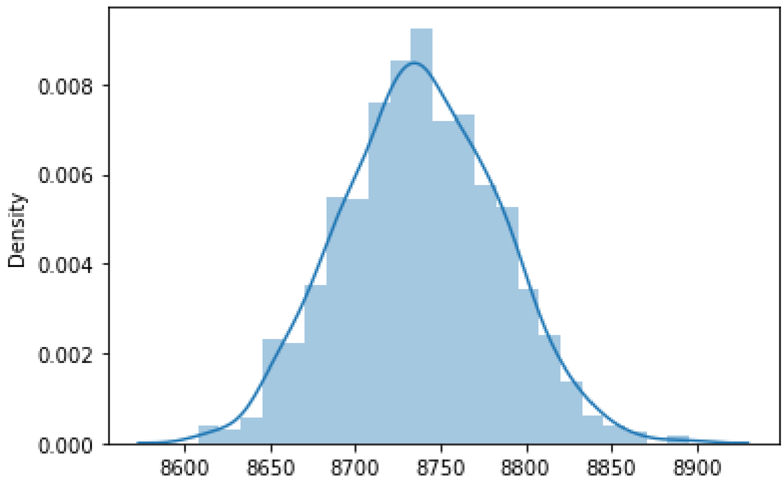
In [208]:

```
sns.distplot(sample_mean_4)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[208]:

```
<AxesSubplot:ylabel='Density'>
```




In [385]:

```
caluclate_ci(np.mean(sample_mean_4),np.std(sample_mean_4),10000)
```

```
We can be confident that 90% of customers will spend in range of 8737.430609
837696,8738.944243362303
We can be confident that 95% of customers will spend in range of 8737.285623
809454,8739.089229390545
We can be confident that 99% of customers will spend in range of 8737.002256
834427,8739.372596365572
```

In [299]:

```
#Female Population mean
df.loc[df.Gender == 'F']['Purchase'].mean()
```

Out[299]:

```
8734.565765155476
```

In [300]:

```
#Male Population mean
df.loc[df.Gender == 'M']['Purchase'].mean()
```

Out[300]:

```
9437.526040472265
```

#### Sample Size - 5000

#### Male CI

In [302]:

```
samples = 5000
repetetion = 1000
sample_mean_5 = []
for rep in range(repetetion):
    x = df.loc[df['Gender'] == 'M']['Purchase'].sample(samples).mean()
    sample_mean_5.append(x)
np.mean(sample_mean_5)
```

Out[302]:

```
9439.4632282
```

In [386]:

```
caluclate_ci(np.mean(sample_mean_5),np.std(sample_mean_5),5000)
```

```
We can be confident that 90% of customers will spend in range of 9437.797519
21965,9441.12893718035
We can be confident that 95% of customers will spend in range of 9437.478413
537403,9441.448042862598
We can be confident that 99% of customers will spend in range of 9436.854739
566967,9442.071716833034
```

#### Female CI

In [288]:

```
samples = 5000
repetetion = 1000
sample_mean_6 = []
for rep in range(repetetion):
    x = df.loc[df['Gender'] == 'F']['Purchase'].sample(samples).mean()
    sample_mean_6.append(x)
np.mean(sample_mean_6)
```

Out[288]:

```
8733.637042
```

In [387]:

```
caluclate_ci(np.mean(sample_mean_6),np.std(sample_mean_6),5000)
```

```
We can be confident that 90% of customers will spend in range of 8732.127594
702279,8735.146489297722
We can be confident that 95% of customers will spend in range of 8731.838424
615884,8735.435659384117
We can be confident that 99% of customers will spend in range of 8731.273258
070923,8736.000825929077
```

In [ ]:

```

```

#### Comparing CI of Different Sample Sizes

#### Female

```
Population Mean -- 8734.565765155476

female 10000 samples -- 8738.1874266

We can be confident that 90% of female customers will spend in range of
8737.430609837696,8738.944243362303
We can be confident that 95% of female customers will spend in range of
8737.285623809454,8739.089229390545
We can be confident that 99% of female customers will spend in range of
8737.002256834427,8739.372596365572

female 5000 samples -- 8733.637042

We can be confident that 90% of female customers will spend in range of
8732.127594702279,8735.146489297722
We can be confident that 95% of female customers will spend in range of
8731.838424615884,8735.435659384117
We can be confident that 99% of female customers will spend in range of
8731.273258070923,8736.000825929077

female 1000 samples -- 8731.657868

We can be confident that 90% of female customers will spend in range of (8723.59480947706,
8739.720926522941)
```

```
We can be confident that 95% of female customers will spend in range of
(8722.050141217895, 8741.265594782106)
We can be confident that 99% of female customers will spend in range of
(8719.031174650388, 8744.284561349612)
```

From above we can say that sample size of 5000 mean is equivalent to population mean compared to other sample size's

In [310]:

```
df.loc[df.Gender=='F']['Purchase'].size
```

Out[310]:

```
135809
```

**Male**

```
Male Popuplation Mean --9437.526040472265

Male 10000 Samples -- 9436.2865602
We can be confident that 90% of customers will spend in range of
9435.452797701077,9437.120322698924
We can be confident that 95% of customers will spend in range of
9435.293070909547,9437.280049490453
We can be confident that 99% of customers will spend in range of
9434.980893948123,9437.592226451878

Male 5000 Samples -- 9439.4632282
We can be confident that 90% of customers will spend in range of
9437.79751921965,9441.12893718035
We can be confident that 95% of customers will spend in range of
9437.478413537403,9441.448042862598
We can be confident that 99% of customers will spend in range of
9436.854739566967,9442.071716833034

Male 1000 Samples -- 9431.573809
We can be confident that 90% of customers will spend in range of
9423.133436309066,9440.014181690933
We can be confident that 95% of customers will spend in range of
9421.516484658014,9441.631133341985
We can be confident that 99% of customers will spend in range of
9418.356244297769,9444.79137370223
```

```
From above we can see that sample mean changes with the sample size and population mean is
equivalent to 10000 samples
```

In [308]:

```
df.loc[df.Gender=='M']['Purchase'].size
```

Out[308]:

```
414259
```

```
Population size of male customers is 3 times larger than female customers
```

## Marital Status

#### 0 - Married 1 - Un-Married

In [313]:

```
df.loc[df['Marital_Status'] == 0]['Purchase'].size
```

Out[313]:

324731

In [314]:

```
df.loc[df['Marital_Status'] == 1]['Purchase'].size
```

Out[314]:

225337

In [315]:

```
df.loc[df['Marital_Status'] == 0]['Purchase'].mean()
```

Out[315]:

9265.907618921507

In [316]:

```
df.loc[df['Marital_Status'] == 1]['Purchase'].mean()
```

Out[316]:

9261.174574082374

In [322]:

```
# Spending mean amount of married and unmarried customers are almost equal
```

#### *Married sample size 1000*

In [327]:

```
samples = 1000
repetetion = 1000
sample_mean_7 = []
for rep in range(repetetion):
    x = df.loc[df['Marital_Status'] == 0]['Purchase'].sample(samples).mean()
    sample_mean_7.append(x)
np.mean(sample_mean_7)
```

Out[327]:

9261.274398

In [388]:

```
caluclate_ci(np.mean(sample_mean_7),np.std(sample_mean_7),1000)
```

```
We can be confident that 90% of customers will spend in range of 9252.951964
814969,9269.59683118503
We can be confident that 95% of customers will spend in range of 9251.357607
246639,9271.19118875336
We can be confident that 99% of customers will spend in range of 9248.241525
740543,9274.307270259456
```

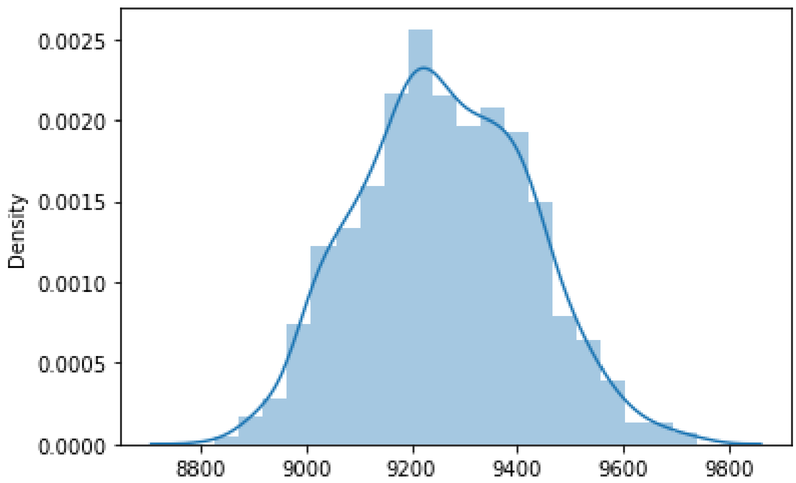
In [329]:

```
sns.distplot(sample_mean_7)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[329]:

<AxesSubplot:ylabel='Density'>




***Un-Married sample size 1000***

In [330]:

```
samples = 1000
repetetion = 1000
sample_mean_8 = []
for rep in range(repetetion):
    x = df.loc[df['Marital_Status'] == 1]['Purchase'].sample(samples).mean()
    sample_mean_8.append(x)
np.mean(sample_mean_8)
```

Out[330]:

9269.046088

In [389]:

```
caluclate_ci(np.mean(sample_mean_8),np.std(sample_mean_8),1000)
```

```
We can be confident that 90% of customers will spend in range of 9260.910274
914764,9277.181901085234
We can be confident that 95% of customers will spend in range of 9259.351668
810205,9278.740507189794
We can be confident that 99% of customers will spend in range of 9256.305461
513815,9281.786714486183
```

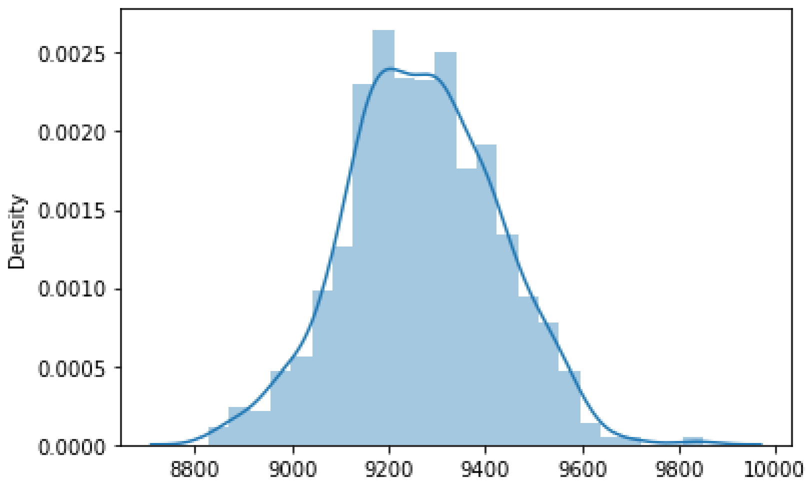
In [335]:

```
sns.distplot(sample_mean_8)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[335]:

```
<AxesSubplot:ylabel='Density'>
```




***Married sample size 5000***

In [333]:

```
samples = 5000
repetetion = 1000
sample_mean_9 = []
for rep in range(repetetion):
    x = df.loc[df['Marital_Status'] == 0]['Purchase'].sample(samples).mean()
    sample_mean_9.append(x)
np.mean(sample_mean_9)
```

Out[333]:

```
9269.079396199999
```

In [390]:

```
caluclate_ci(np.mean(sample_mean_9),np.std(sample_mean_9),5000)
```

```
We can be confident that 90% of customers will spend in range of 9267.465432
678506,9270.693359721492
We can be confident that 95% of customers will spend in range of 9267.156240
05439,9271.002552345608
We can be confident that 99% of customers will spend in range of 9266.551940
594241,9271.606851805756
```

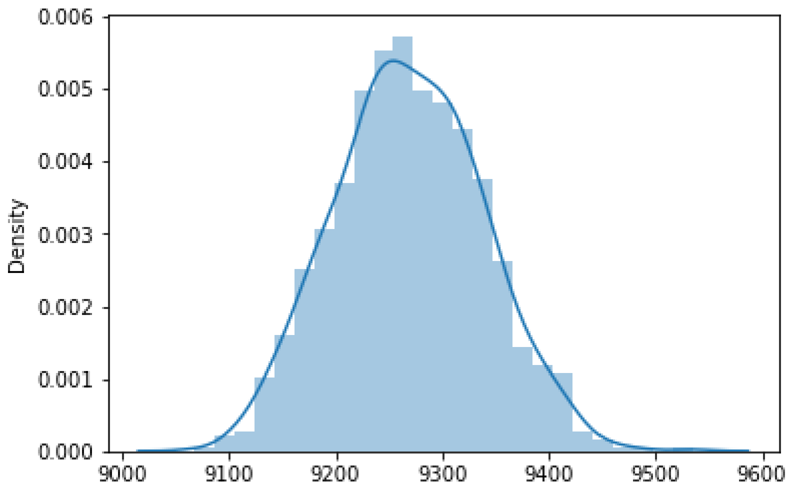
In [337]:

```
sns.distplot(sample_mean_9)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[337]:

```
<AxesSubplot:ylabel='Density'>
```




***Un-Married sample size 5000***

In [338]:

```
samples = 5000
repetition = 1000
sample_mean_10 = []
for rep in range(repetetion):
    x = df.loc[df['Marital_Status'] == 1]['Purchase'].sample(samples).mean()
    sample_mean_10.append(x)
np.mean(sample_mean_10)
```

Out[338]:

```
9256.655027199999
```

In [391]:

```
caluclate_ci(np.mean(sample_mean_10),np.std(sample_mean_10),5000)
```

```
We can be confident that 90% of customers will spend in range of 9255.038655
728682,9258.271398671315
We can be confident that 95% of customers will spend in range of 9254.729001
805215,9258.581052594782
We can be confident that 99% of customers will spend in range of 9254.123800
76163,9259.186253638367
```

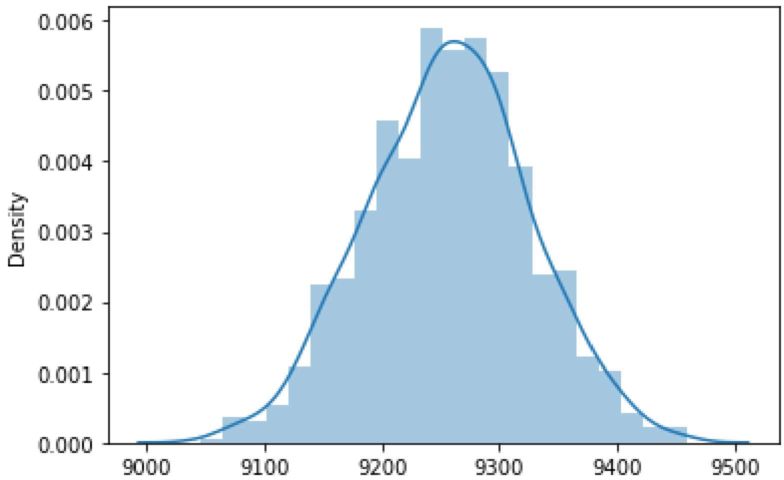
In [341]:

```
sns.distplot(sample_mean_10)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[341]:

```
<AxesSubplot:ylabel='Density'>
```




***Married sample size 10000***

In [355]:

```
samples = 10000
repetetion = 1000
sample_mean_11 = []
for rep in range(repetetion):
    x = df.loc[df['Marital_Status'] == 0]['Purchase'].sample(samples).mean()
    sample_mean_11.append(x)
np.mean(sample_mean_11)
```

Out[355]:

```
9269.1311679
```

In [392]:

```
caluclate_ci(np.mean(sample_mean_11),np.std(sample_mean_11),10000)
```

```
We can be confident that 90% of customers will spend in range of 9268.317086
338899,9269.9452494611
We can be confident that 95% of customers will spend in range of 9268.161129
893311,9270.101205906687
We can be confident that 99% of customers will spend in range of 9267.856321
859339,9270.40601394066
```

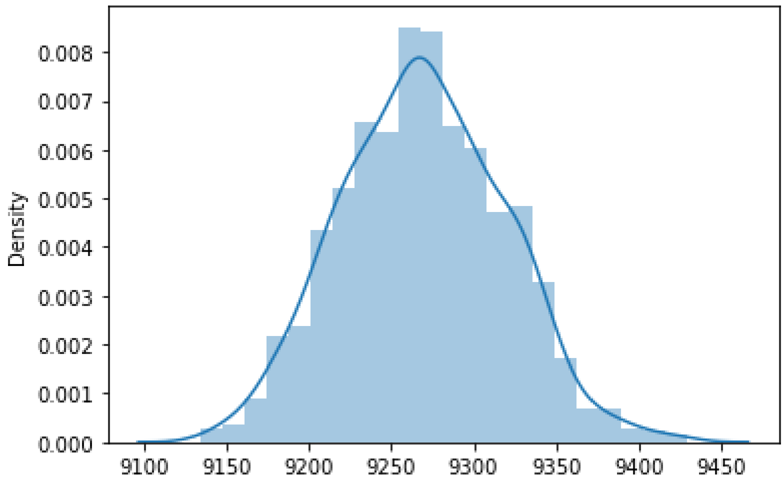
In [358]:

```
sns.distplot(sample_mean_11)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[358]:

```
<AxesSubplot:ylabel='Density'>
```




***Un-Married sample size 10000***

In [359]:

```
samples = 10000
repetetion = 1000
sample_mean_12 = []
for rep in range(repetetion):
    x = df.loc[df['Marital_Status'] == 1]['Purchase'].sample(samples).mean()
    sample_mean_12.append(x)
np.mean(sample_mean_12)
```

Out[359]:

```
9259.2263345
```

In [393]:

```
caluclate_ci(np.mean(sample_mean_12),np.std(sample_mean_12),10000)
```

```
We can be confident that 90% of customers will spend in range of 9258.417822
687972,9260.034846312026
We can be confident that 95% of customers will spend in range of 9258.262933
25865,9260.189735741349
We can be confident that 99% of customers will spend in range of 9257.960210
647498,9260.4924583525
```

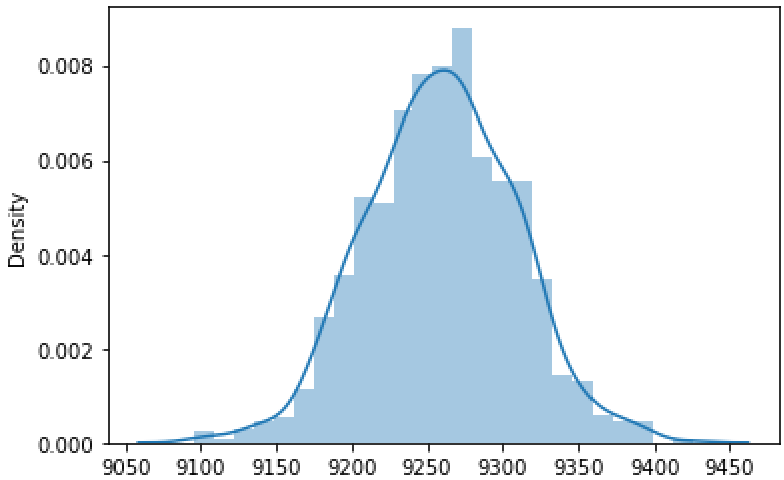
In [361]:

```
sns.distplot(sample_mean_12)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[361]:

```
<AxesSubplot:ylabel='Density'>
```




```
Un-Married:

Population sample mean -- 9261.174574082374

sample size -- 1000
sample mean -- 9269.046088
We can be confident that 90% of customers will spend in range of
9252.951964814969,9269.59683118503
We can be confident that 95% of customers will spend in range of
9251.357607246639,9271.19118875336
We can be confident that 99% of customers will spend in range of
9248.241525740543,9274.307270259456

sample size -- 5000
sample mean -- 9256.655027199999
We can be confident that 90% of customers will spend in range of
9260.910274914764,9277.181901085234
We can be confident that 95% of customers will spend in range of
9259.351668810205,9278.740507189794
```

```
We can be confident that 99% of customers will spend in range of
9256.305461513815,9281.786714486183

sample size -- 10000
sample mean -- 9260.9225613
We can be confident that 90% of customers will spend in range of
9260.11336285742,9261.731759742579
We can be confident that 95% of customers will spend in range of
9259.95834188789,9261.88678071211
We can be confident that 99% of customers will spend in range of
9259.655362188849,9262.189760411151
```

```
Married :

Population Standard mean -- 9265.907618921507

sample size -- 1000
sample mean -- 9261.274
We can be confident that 90% of customers will spend in range of
9252.951964814969,9269.59683118503
We can be confident that 95% of customers will spend in range of
9251.357607246639,9271.19118875336
We can be confident that 99% of customers will spend in range of
9248.241525740543,9274.307270259456

sample size -- 5000
sample mean -- 9269.079
We can be confident that 90% of customers will spend in range of
9267.465432678506,9270.693359721492
We can be confident that 95% of customers will spend in range of
9267.15624005439,9271.002552345608
We can be confident that 99% of customers will spend in range of
9266.551940594241,9271.606851805756

sample size -- 10000
sample mean -- 9269.1311679
We can be confident that 90% of customers will spend in range of
9268.317086338899,9269.9452494611
We can be confident that 95% of customers will spend in range of
9268.161129893311,9270.101205906687
We can be confident that 99% of customers will spend in range of
9267.856321859339,9270.40601394066
```

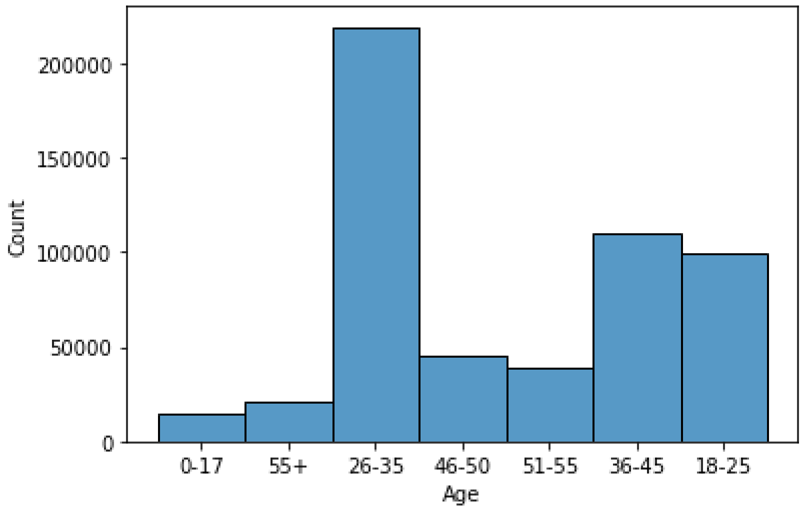
## Age Groups

In [356]:

```
sns.histplot(x=df.Age)
```

Out[356]:

<AxesSubplot:xlabel='Age', ylabel='Count'>




In [363]:

```
#we can see that 26-35, 36-45, 18-25 are the top 3 age groups who are contribuiting.
#lets caluclate for these three age groups.
```

### Age 26-35

In [365]:

```
samples = 10000
repetetion = 1000
sample_mean_13 = []
for rep in range(repetetion):
    x = df.loc[df['Age'] == '26-35']['Purchase'].sample(samples).mean()
    sample_mean_13.append(x)
np.mean(sample_mean_13)
```

Out[365]:

9252.4704451

In [394]:

```
caluclate_ci(np.mean(sample_mean_12),np.std(sample_mean_12),10000)
```

```
We can be confident that 90% of customers will spend in range of 9258.417822
687972,9260.034846312026
We can be confident that 95% of customers will spend in range of 9258.262933
25865,9260.189735741349
We can be confident that 99% of customers will spend in range of 9257.960210
647498,9260.4924583525
```

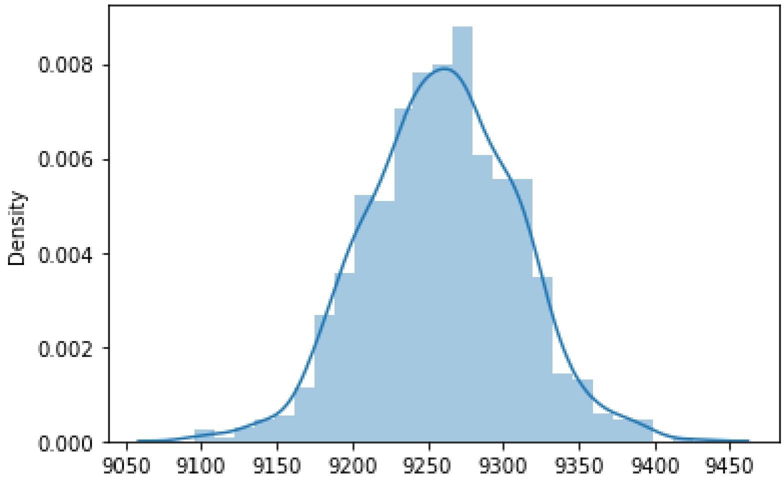
In [373]:

```
sns.distplot(sample_mean_12)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[373]:

```
<AxesSubplot:ylabel='Density'>
```




**Population means and length**

In [366]:

```
df.loc[df['Age'] == '26-35']['Purchase'].size
```

Out[366]:

```
219587
```

In [369]:

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

Out[369]:

```
9252.690632869888
```

In [367]:

```
df.loc[df['Age'] == '36-45']['Purchase'].size
```

Out[367]:

```
110013
```

In [370]:

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

Out[370]:

```
9331.350694917874
```

In [368]:

```
df.loc[df['Age'] == '18-25']['Purchase'].size
```

Out[368]:

```
99660
```

In [371]:

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

Out[371]:

```
9169.663606261289
```

### Age 36-45

In [374]:

```
samples = 10000
repetetion = 1000
sample_mean_14 = []
for rep in range(repetetion):
    x = df.loc[df['Age'] == '36-45']['Purchase'].sample(samples).mean()
    sample_mean_14.append(x)
np.mean(sample_mean_14)
```

Out[374]:

```
9333.7095372
```

In [399]:

```
caluclate_ci(np.mean(sample_mean_14),np.std(sample_mean_14),10000)
```

```
We can be confident that 90% of customers will spend in range of 9332.914177
277638,9334.504897122362
We can be confident that 95% of customers will spend in range of 9332.761807
401717,9334.657266998283
We can be confident that 99% of customers will spend in range of 9332.464009
114887,9334.955065285114
```

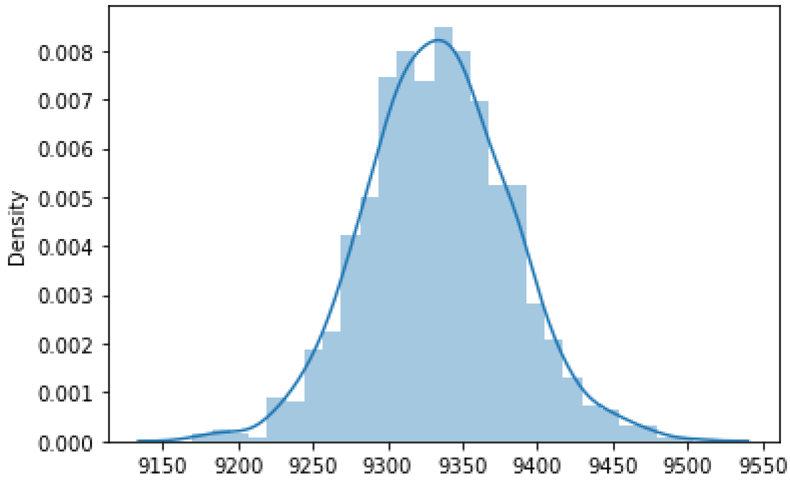
In [400]:

```
sns.distplot(sample_mean_14)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[400]:

<AxesSubplot:ylabel='Density'>




### Age 18-25

In [378]:

```
samples = 10000
repetetion = 1000
sample_mean_15 = []
for rep in range(repetetion):
    x = df.loc[df['Age'] == '18-25']['Purchase'].sample(samples).mean()
    sample_mean_15.append(x)
np.mean(sample_mean_15)
```

Out[378]:

9170.5956748

In [397]:

```
caluclate_ci(np.mean(sample_mean_15),np.std(sample_mean_15),10000)
```

```
We can be confident that 90% of customers will spend in range of 9169.792563
016597,9171.398786583404
We can be confident that 95% of customers will spend in range of 9169.638708
089604,9171.552641510398
We can be confident that 99% of customers will spend in range of 9169.338007
354621,9171.85334224538
```

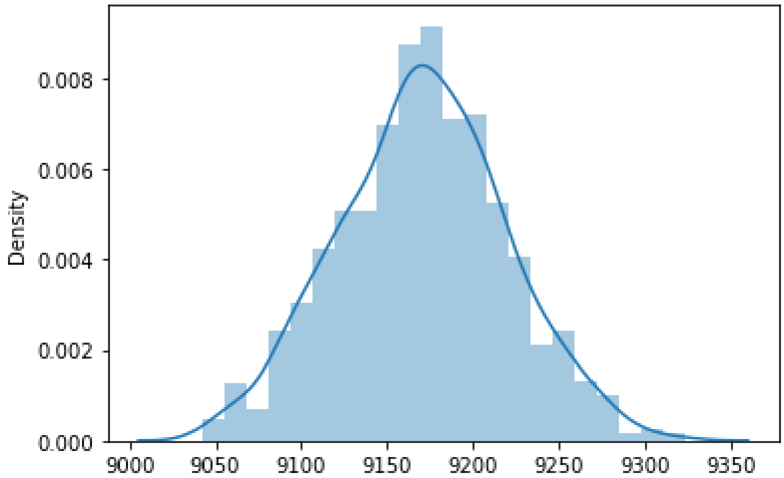
In [398]:

```
sns.distplot(sample_mean_15)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[398]:

```
<AxesSubplot:ylabel='Density'>
```




```
From the above confidence intervals (All the mean analysis is done for the purchase
column):

Males spend more than Females

Males average mean is 9431.573809
we can be 95% sure that males purchase (9421.516484658014 - 9441.631133341985)

Females average mean is 8731.657868
we can be 95% sure that females purchase (8722.050141217895-8741.265594782106)


Marital Status:

There is no much difference in means of Married customers and unmarried customers.

Sample mean of unmarried purchase amount is 9260.9225613.
We can be confident that 95% of customers will spend in range of
9259.95834188789,9261.88678071211

Sample mean of Married customers purchase amount is 9269.1311679
We can be confident that 95% of customers will spend in range of
9268.161129893311,9270.101205906687


Age Group:
```

```
The customers with age groups 26-35, 36-45, 18-25 are the top 3 age groups who are
contribuiting more than other age groups.

Sample mean of age group 26-35 is 9252.4704451
We can be confident that 95% of customers will spend in range of
9258.26293325865,9260.189735741349

Sample mean of age group 36-45 is 9333.7095372
We can be confident that 95% of customers will spend in range of
9332.761807401717,9334.657266998283

Sample mean of age group 18-25 is 9170.5956748
We can be confident that 95% of customers will spend in range of
9169.638708089604,9171.552641510398
```

## Insights

```
-- In Age group 26-35 age group customers purchases more compared to other age groups.
-- Customers from city category B purchase more compare to A and C
-- People who stayed in the city for 1 year buy more products than others
-- Customers with the occupation 4,0,7 purchases more products
-- 5,1,8 are the top selling product category in walmart
-- Male customers purchase more compared to females
-- Male average spend amount is more than female
```

#### Recommendations

```
-- we can target on male customers as they are more spending
-- we can push female customers also to buy products which gives better results by giving
some offers.
-- We can find the most selling product in the category and add combo to it so that we can
make customers to habituated to the
   other product also so that they will try the product later.
-- As we can see customers who lives less than a year in a city buys more we can assume
from the products category that they
   are setting up their house and attract them to buy those kind of products
-- we can Advertise more on the most trending products --> 5,1,8 for males
                                                       --> 5,8 for females
-- we can target customers with occupation 4,0,7 and products with 1,5,8.
-- we can target on age groups of 26-35 by giving good valued offers and attract them.
-- There is no much spending difference between married and unmarried customers so we can
target most selling products.
```