In [9]:
```
import pandas as pd
import numpy as np
import seaborn as sns
```

In [10]:
```
df=pd.read_csv('walmart_data.csv')
```

In [11]:
```
df.head()
```

Out[11]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mar
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 [12]:
```
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
```

In [13]:
```
df.describe()
# mean in case of purchase has variation wrt to 50% , so it may contain outliers
```

Out[13]:

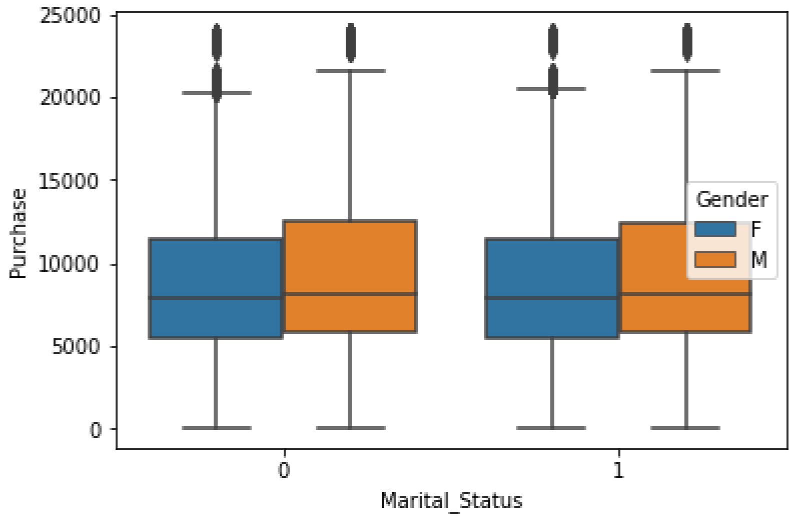
	User_ID	Occupation	Marital_Status	Product_Category	Purchase
count	5.500680e+05	550068.000000	550068.000000	550068.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9263.968713
std	1.727592e+03	6.522660	0.491770	3.936211	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	12.000000
25%	1.001516e+06	2.000000	0.000000	1.000000	5823.000000
50%	1.003077e+06	7.000000	0.000000	5.000000	8047.000000
75%	1.004478e+06	14.000000	1.000000	8.000000	12054.000000
max	1.006040e+06	20.000000	1.000000	20.000000	23961.000000

In [14]:
```
for i in df.columns:
    print(i ,':', df[i].nunique())
```

```
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
```

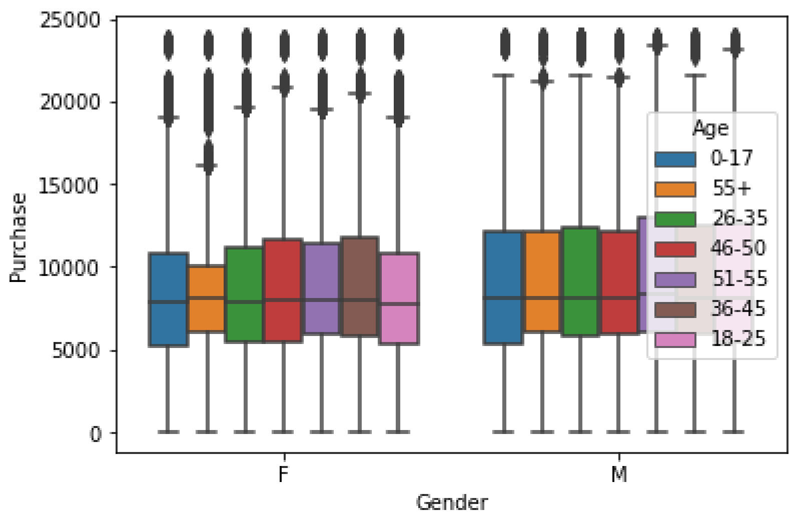
In [15]:
```
sns.boxplot(x='Marital_Status',y='Purchase',hue='Gender',data=df)
#clearly there are some outliers in the data.
```

Out[15]: <AxesSubplot:xlabel='Marital_Status', ylabel='Purchase'>



In [16]:
```
sns.boxplot(x='Gender',y='Purchase',hue='Age',data=df)
#clearly there are some outliers in the data.
```

Out[16]: <AxesSubplot:xlabel='Gender', ylabel='Purchase'>



In [17]:
```
df.isna().sum()
# NO missing values in the dataset
```

Out[17]:
```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category              0
Purchase                      0
dtype: int64
```

In [18]:
```
# amount spend per transaction by females
df[df['Gender']=='F']['Purchase'].sum()/len(df[df['Gender']=='F'])
```

Out[18]:
```
8734.565765155476
```

In [19]:
```
# amount spend per transaction by males
df[df['Gender']=='M']['Purchase'].sum()/len(df[df['Gender']=='M'])
```

Out[19]:
```
9437.526040472265
```

In [20]:
```
# comparing the amount spend per transaction by male and female it can be found t
# purchase per transaction by male >  purchase per transaction by female
```

In [21]:
```
# creating a bootstraping function
```

In [22]:
```
def sample_mean(x,sample_size,stats):
    x=np.array(x)
    resample_set=[]
    for i in range(sample_size):
        index=np.random.randint(0,len(x),len(x))
        sample=x[index]
        bstat=stats(sample)
        resample_set.append(bstat)
        sample_mean_array=resample_set


    return np.mean(resample_set)
```

In [23]:
```
# creating seperate dataframe that contains only females
df_female=df[df['Gender']=='F']
female_purchase=np.array(df_female['Purchase'])
```

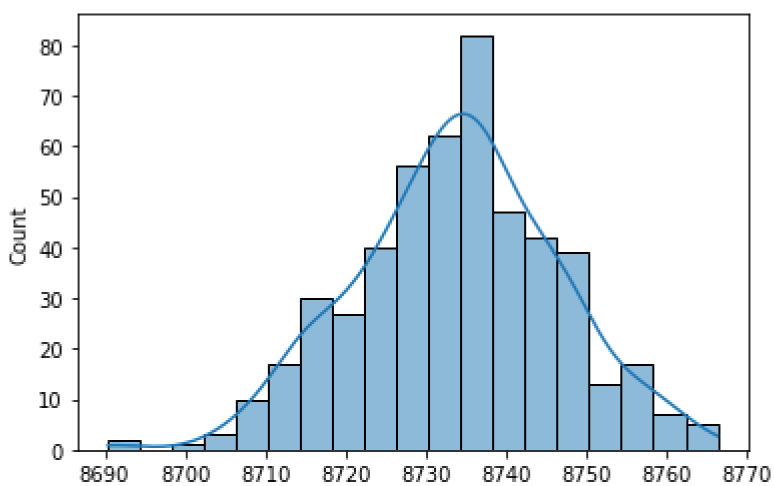
In [24]:
```
# sample mean for female purchases
female_sample_mean=sample_mean(female_purchase,500,np.mean)
female_sample_mean
```

Out[24]: 8734.314264047302

In [25]:
```
# bootstraping function that returns the sample mean array
def sample_mean_array(x,sample_size,stats):
    x=np.array(x)
    resample_set=[]
    for i in range(sample_size):
        index=np.random.randint(0,len(x),len(x))
        sample=x[index]
        bstat=stats(sample)
        resample_set.append(bstat)
    return resample_set
```

In [26]:
```
# distribution of female sample mean
sns.histplot(sample_mean_array(female_purchase,500,np.mean),kde=True)
#the distribution is like gussian distribution
```

Out[26]: <AxesSubplot:ylabel='Count'>



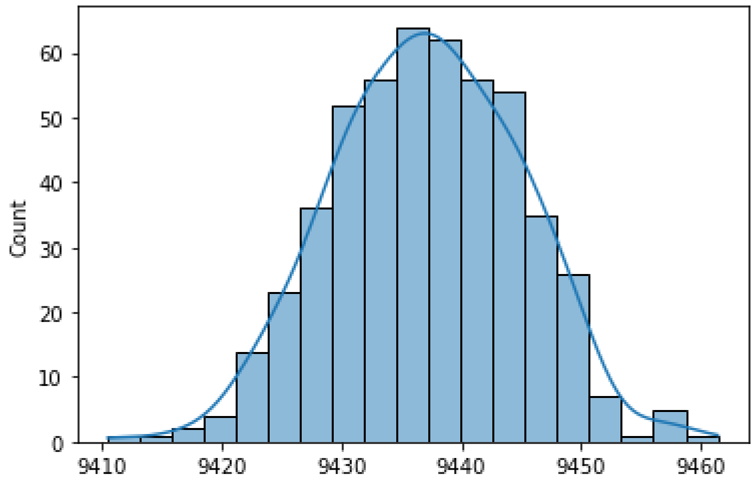
In [27]:
```
# creating seperate dataframe that contains only males
df_male=df[df['Gender']=='M']
male_purchase=np.array(df_male['Purchase'])
```

In [28]:
```
# sample mean for female purchases
male_sample_mean=sample_mean(male_purchase,500,np.mean)
male_sample_mean
```

Out[28]: 9437.347109832255

In [29]:
```
# distribution of male sample mean
sns.histplot(sample_mean_array(male_purchase,500,np.mean),kde=True)
#the distribution is like gussian distribution
```

Out[29]: <AxesSubplot:ylabel='Count'>



In [ ]:

In [30]:
```
# calculating the population mean range for female with 90%,95%,99% confidence ir
```

In [31]:
```
# confidence_interval = (sample_mean - margin_of_error,sample_mean + margin_of_er
# maargin_error=t_critical * sigma
# sigma = sample_stdev/math.sqrt(sample_size)
```

In [32]:
```
# sample mean for female customers
df_female=df[df['Gender']=='F']
female=purchase=df_female['Purchase']
female_sample_mean=sample_mean(female_purchase,500,np.mean)
female_sample_mean
```

Out[32]: 8734.999907841158

In [33]:
```
# calculating the range for 95% confidence interval female
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.95, df=499)
sample_stdev = female_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=female_sample_mean+margin_error
lcv=female_sample_mean-margin_error
print("CI range for female with 95% is :","LCV:",lcv,"UCV:",ucv)
```

```
CI range for female with 95% is : LCV: 8486.57182910481 UCV: 8983.427986577506
```

In [34]:
```
# calculating the range for 90% confidence interval female
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.90, df=499)
sample_stdev = female_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=female_sample_mean+margin_error
lcv=female_sample_mean-margin_error
print("CI range for female with 95% is :","LCV:",lcv,"UCV :",ucv)
```

```
CI range for female with 95% is : LCV: 8541.545860079132 UCV : 8928.45395560318
5
```

In [35]:
```
# calculating the range for 99% confidence interval female
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.99, df=499)
sample_stdev = female_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=female_sample_mean+margin_error
lcv=female_sample_mean-margin_error
print("CI range for female with 95% is :","LCV:",lcv,"UCV :",ucv)
```

```
CI range for female with 95% is : LCV: 8383.165566451178 UCV : 9086.83424923113
9
```

In [ ]:

In [36]:
```
# calculating the population mean range for male with 90%,95%,99% confidence inte
```

In [37]:
```
# confidence_interval = (sample_mean - margin_of_error,sample_mean + margin_of_er
# maargin_error=t_critical * sigma
# sigma = sample_stdev/math.sqrt(sample_size)
```

In [38]:
```
# sample mean for female customers
df_male=df[df['Gender']=='M']
male=purchase=df_male['Purchase']
male_sample_mean=sample_mean(male_purchase,500,np.mean)
male_sample_mean
```

Out[38]: 9437.492166074846

In [39]:
```
# calculating the range for 95% confidence interval male
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.95, df=499)
sample_stdev = male_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=male_sample_mean+margin_error
lcv=male_sample_mean-margin_error
print("CI range for male with 95% is :","LCV:",lcv,"UCV:",ucv)
```

```
CI range for male with 95% is : LCV: 9172.130275876498 UCV: 9702.854056273194
```

In [40]:
```
# calculating the range for 90% confidence interval male
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.90, df=499)
sample_stdev = male_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=male_sample_mean+margin_error
lcv=male_sample_mean-margin_error
print("CI range for male with 95% is :","LCV:",lcv,"UCV:",ucv)
```

```
CI range for male with 95% is : LCV: 9230.85154782519 UCV: 9644.132784324502
```

In [41]:
```
# calculating the range for 99% confidence interval male
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.99, df=499)
sample_stdev = male_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=male_sample_mean+margin_error
lcv=male_sample_mean-margin_error
print("CI range for male with 95% is :","LCV:",lcv,"UCV:",ucv)
```

```
CI range for male with 95% is : LCV: 9061.675445425402 UCV: 9813.30888672429
```

In [42]:
```
# clearly there is no overlapping region in females and males
```

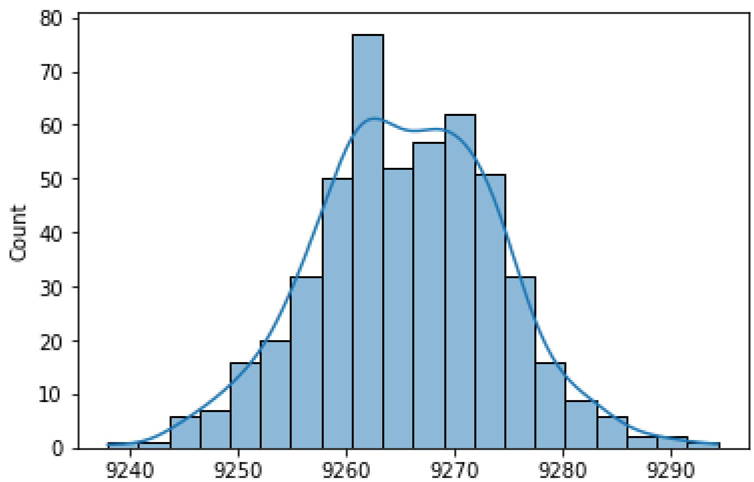
In [ ]:
```
```

In [43]:
```
# creating seperate dataframe that contains only married
df_unmarried=df[df['Marital_Status']==0]
unmarried_purchase=np.array(df_unmarried['Purchase'])
unmarried_sample_mean=sample_mean(unmarried_purchase,500,np.mean)
unmarried_sample_mean
#1. unmarried people are having sample mean more than female and male
```

Out[43]: 9265.2742037625

In [44]:
```
# distribution on unmarried people
sns.histplot(sample_mean_array(unmarried_purchase,500,np.mean),kde=True)
```

Out[44]: <AxesSubplot:ylabel='Count'>

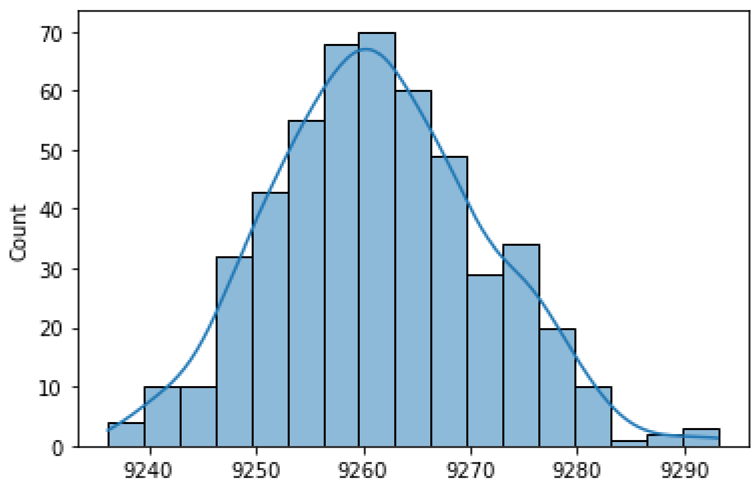


In [45]:
```
df_married=df[df['Marital_Status']==1]
married_purchase=np.array(df_married['Purchase'])
married_sample_mean=sample_mean(married_purchase,500,np.mean)
married_sample_mean
#1. unmarried people are having sample mean more than female and male
```

Out[45]: 9261.545955267002

In [46]:
```
# distribution on unmarried people
sns.histplot(sample_mean_array(married_purchase,500,np.mean),kde=True)
```

Out[46]: <AxesSubplot:ylabel='Count'>



In [47]:
```
# married and unmarried have almost same sample mean
```

In [48]:
```
# calculating the confidence interval for unmarried people
```

In [49]:
```
# calculating the range for 90% confidence interval unmarried
df_unmarried=df[df['Marital_Status']==0]
unmarried_purchase=np.array(df_unmarried['Purchase'])
unmarried_sample_mean=sample_mean(unmarried_purchase,500,np.mean)
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.90, df=499)
sample_stdev = unmarried_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=unmarried_sample_mean+margin_error
lcv=unmarried_sample_mean-margin_error
print("CI range for unmarried with 90% is :","LCV:",lcv,"UCV:",ucv)
```

```
CI range for unmarried with 90% is : LCV: 9061.929785284461 UCV: 9469.948748800
67
```

In [50]:
```
# calculating the range for 95% confidence interval unmarried
df_unmarried=df[df['Marital_Status']==0]
unmarried_purchase=np.array(df_unmarried['Purchase'])
unmarried_sample_mean=sample_mean(unmarried_purchase,500,np.mean)
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.95, df=499)
sample_stdev = unmarried_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=unmarried_sample_mean+margin_error
lcv=unmarried_sample_mean-margin_error
print("CI range for unmarried with 95% is :","LCV:",lcv,"UCV:",ucv)
```

```
CI range for unmarried with 95% is : LCV: 9004.74513968693 UCV: 9528.7112617037
56
```

In [51]:
```
# calculating the range for 99% confidence interval unmarried
df_unmarried=df[df['Marital_Status']==0]
unmarried_purchase=np.array(df_unmarried['Purchase'])
unmarried_sample_mean=sample_mean(unmarried_purchase,500,np.mean)
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.99, df=499)
sample_stdev = unmarried_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=unmarried_sample_mean+margin_error
lcv=unmarried_sample_mean-margin_error
print("CI range for unmarried with 99% is :","LCV:",lcv,"UCV:",ucv)
```

```
CI range for unmarried with 99% is : LCV: 8894.693883289781 UCV: 9636.756843274
668
```

In [52]:
```
#calculating the confidence interval for unmarried people
```

In [53]:
```
# calculating the range for 90% confidence interval married
df_married=df[df['Marital_Status']==1]
married_purchase=np.array(df_married['Purchase'])
married_sample_mean=sample_mean(married_purchase,500,np.mean)
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.90, df=499)
sample_stdev = married_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=married_sample_mean+margin_error
lcv=married_sample_mean-margin_error
print("CI range for married with 90% is :","LCV:",lcv,"UCV:",ucv)
```

```
CI range for married with 90% is : LCV: 9057.647256873124 UCV: 9464.81806058473
2
```

In [54]:
```
# calculating the range for 95% confidence interval unmarried
df_married=df[df['Marital_Status']==1]
married_purchase=np.array(df_married['Purchase'])
married_sample_mean=sample_mean(married_purchase,500,np.mean)
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.95, df=499)
sample_stdev = married_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=married_sample_mean+margin_error
lcv=married_sample_mean-margin_error
print("CI range for married with 95% is :","LCV:",lcv,"UCV:",ucv)
```

```
CI range for married with 95% is : LCV: 8999.490249117456 UCV: 9522.36718890648
4
```

In [55]:
```
# calculating the range for 99% confidence interval unmarried
df_married=df[df['Marital_Status']==1]
married_purchase=np.array(df_married['Purchase'])
married_sample_mean=sample_mean(married_purchase,500,np.mean)
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.99, df=499)
sample_stdev = married_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=married_sample_mean+margin_error
lcv=married_sample_mean-margin_error
print("CI range for married with 99% is :","LCV:",lcv,"UCV:",ucv)
```

```
CI range for married with 99% is : LCV: 8891.31921791153 UCV: 9631.839632006593
```

In [56]:
```
# Confidence interval as per age category is to be calculated
```

In [57]:
```
df.head()
```

Out[57]:

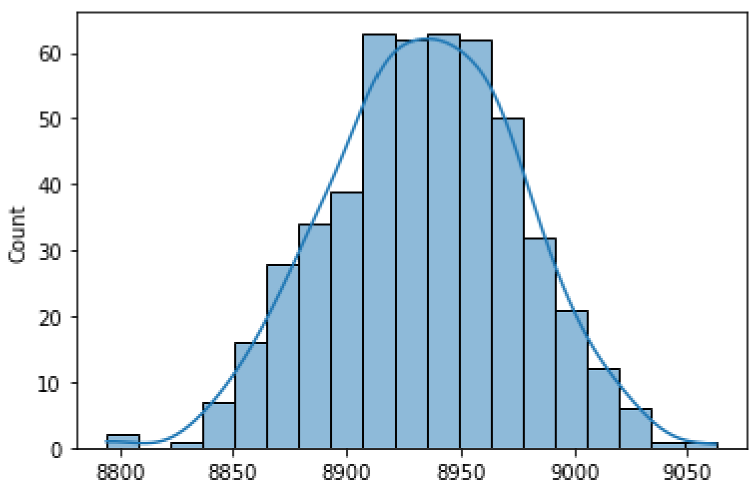
	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mar
0	1000001	P00069042	F	0-17	10	A	2	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	

In [60]:
```
# calculating the age 0-17 for 99% confidence interval unmarried
df_age=df[df['Age']=='0-17']
age_purchase=np.array(df_age['Purchase'])
age_sample_mean=sample_mean(age_purchase,500,np.mean)
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.99, df=499)
sample_stdev = age_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=age_sample_mean+margin_error
lcv=age_sample_mean-margin_error
print("CI range for age 0-17 with 99% is :","LCV:",lcv,"UCV:",ucv)
```

```
CI range for age 0-17 with 99% is : LCV: 8557.54886411608 UCV: 9311.97615349748
```

In [59]:
```
# distribution on 0-17 Age people
sns.histplot(sample_mean_array(age_purchase,500,np.mean),kde=True)
```

Out[59]: <AxesSubplot:ylabel='Count'>

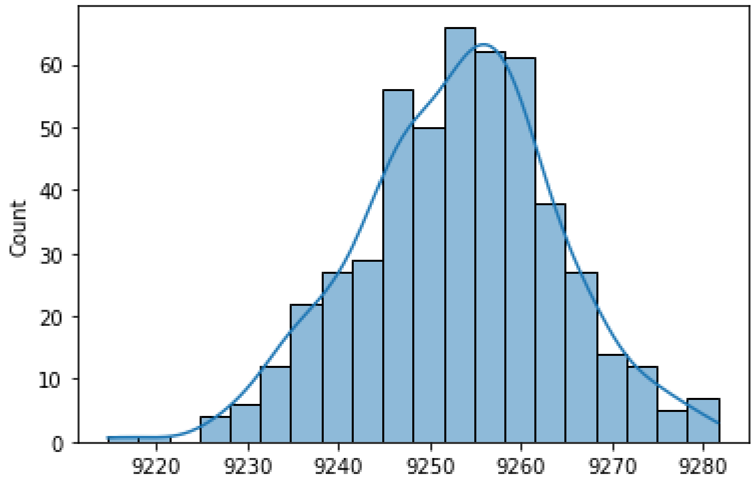


In [62]:
```
# calculating the age 26-35 for 99% confidence interval unmarried
df_age=df[df['Age']=='26-35']
age_purchase=np.array(df_age['Purchase'])
age_sample_mean=sample_mean(age_purchase,500,np.mean)
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.99, df=499)
sample_stdev = age_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=age_sample_mean+margin_error
lcv=age_sample_mean-margin_error
print("CI range for age 26-35 with 99% is :","LCV:",lcv,"UCV:",ucv)
```

```
CI range for age 26-35 with 99% is : LCV: 8882.673869155396 UCV: 9622.254026740
078
```

In [63]:
```
# distribution on 26-35 Age people
sns.histplot(sample_mean_array(age_purchase,500,np.mean),kde=True)
```

Out[63]: <AxesSubplot:ylabel='Count'>

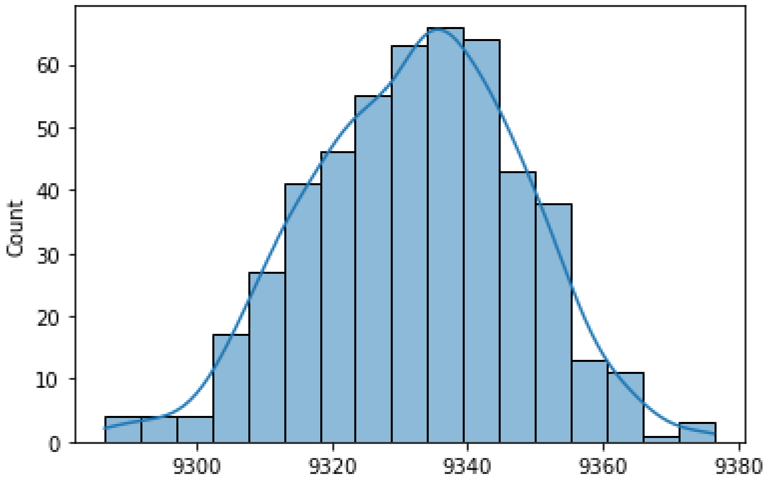


In [64]:
```
# calculating the age 36-45 for 99% confidence interval unmarried
df_age=df[df['Age']=='36-45']
age_purchase=np.array(df_age['Purchase'])
age_sample_mean=sample_mean(age_purchase,500,np.mean)
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.99, df=499)
sample_stdev = age_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=age_sample_mean+margin_error
lcv=age_sample_mean-margin_error
print("CI range for age 36-45 with 99% is :","LCV:",lcv,"UCV:",ucv)
```

```
CI range for age 36-45 with 99% is : LCV: 8961.044744620685 UCV: 9702.454701990
17
```

In [65]:
```
# distribution on 36-45 Age people
sns.histplot(sample_mean_array(age_purchase,500,np.mean),kde=True)
```

Out[65]: <AxesSubplot:ylabel='Count'>

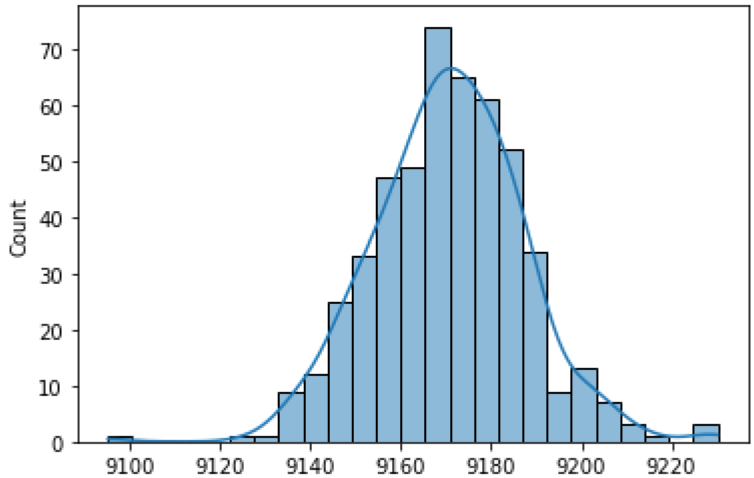


In [68]:
```
# calculating the age 18-25 for 99% confidence interval unmarried
df_age=df[df['Age']=='18-25']
age_purchase=np.array(df_age['Purchase'])
age_sample_mean=sample_mean(age_purchase,500,np.mean)
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.99, df=499)
sample_stdev = age_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=age_sample_mean+margin_error
lcv=age_sample_mean-margin_error
print("CI range for age 18-25 with 99% is :","LCV:",lcv,"UCV:",ucv)
```

```
CI range for age 18-25 with 99% is : LCV: 8799.13738814821 UCV: 9542.2297676214
08
```

In [67]:
```
sns.histplot(sample_mean_array(age_purchase,500,np.mean),kde=True)
```

Out[67]: <AxesSubplot:ylabel='Count'>

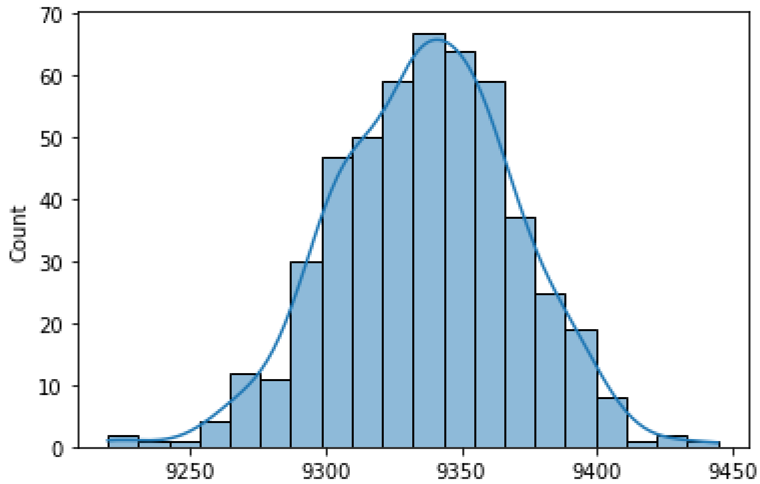


In [70]:
```
# calculating the age 18-25 for 99% confidence interval unmarried
df_age=df[df['Age']=='55+']
age_purchase=np.array(df_age['Purchase'])
age_sample_mean=sample_mean(age_purchase,500,np.mean)
import scipy.stats as stats
t_critical = stats.t.ppf(q = 0.99, df=499)
sample_stdev = age_purchase.std(ddof=1)
sigma=sample_stdev/np.sqrt(1000)
margin_error=t_critical*sigma
ucv=age_sample_mean+margin_error
lcv=age_sample_mean-margin_error
print("CI range for age 55+ with 99% is :","LCV:",lcv,"UCV:",ucv)
```

```
CI range for age 55+ with 99% is : LCV: 8965.587852423654 UCV: 9705.31069891563
```

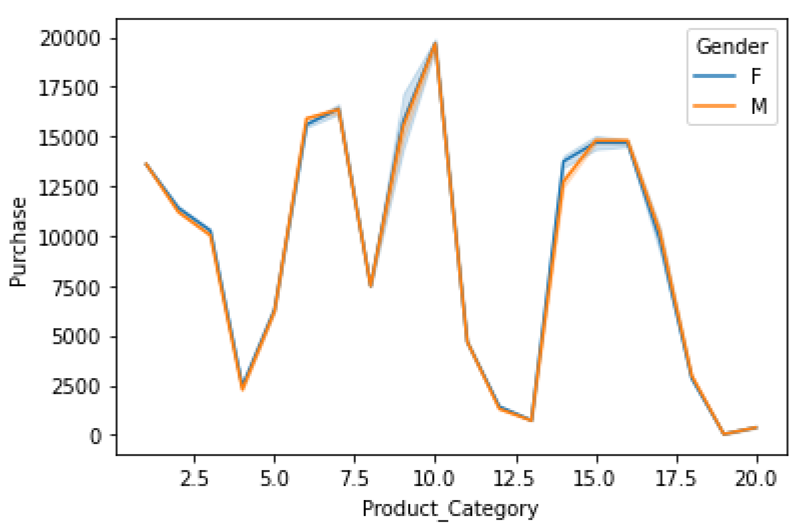
In [71]:
```
sns.histplot(sample_mean_array(age_purchase,500,np.mean),kde=True)
```

Out[71]: <AxesSubplot:ylabel='Count'>



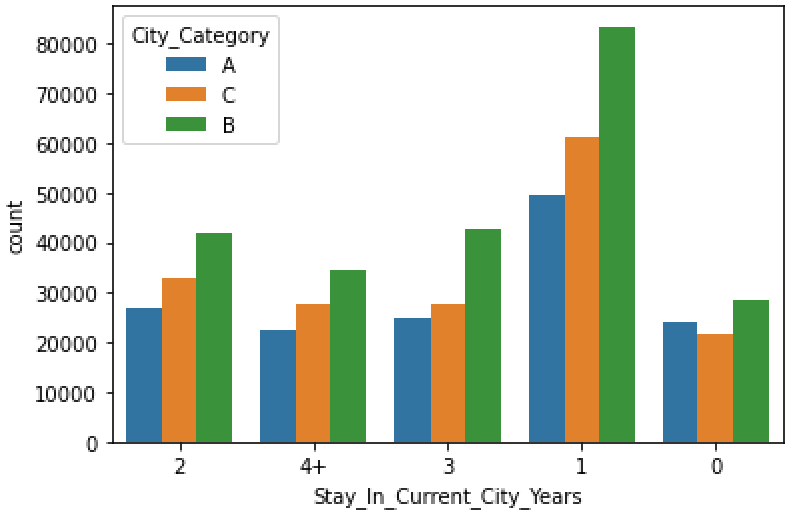
In [99]:
```
#relating between product category and purchase
sns.lineplot(x='Product_Category',y='Purchase',hue='Gender',data=df)
```

Out[99]: <AxesSubplot:xlabel='Product_Category', ylabel='Purchase'>



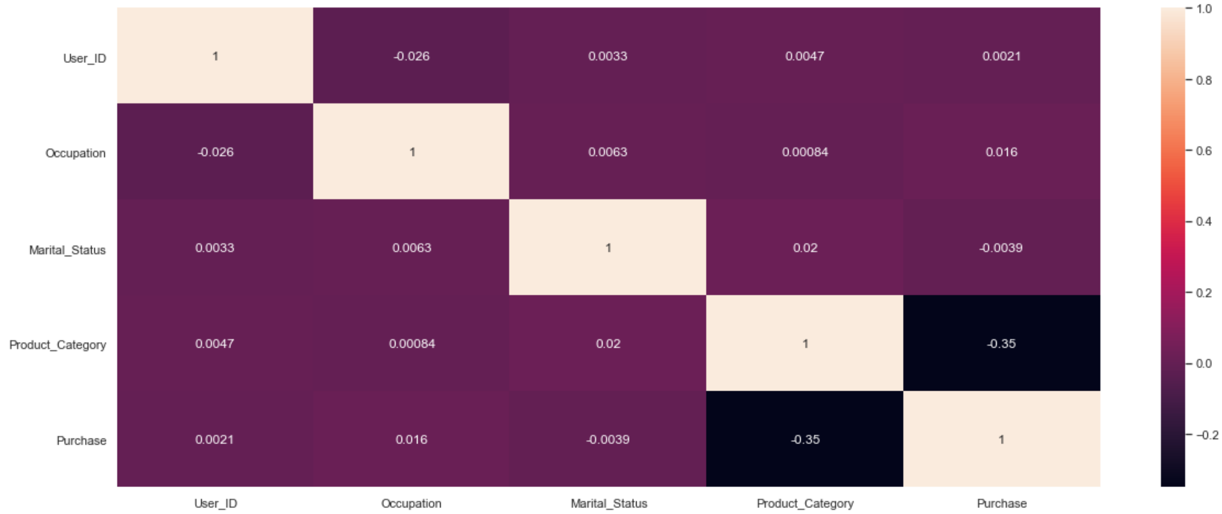
In [15]:
```
sns.countplot(x='Stay_In_Current_City_Years',hue='City_Category',data=df)
```

Out[15]:
```
<AxesSubplot:xlabel='Stay_In_Current_City_Years', ylabel='count'>
```



In [105]:
```
sns.heatmap(df_female.corr(),annot=True)
sns.set(rc={"figure.figsize":(20, 8)})
```

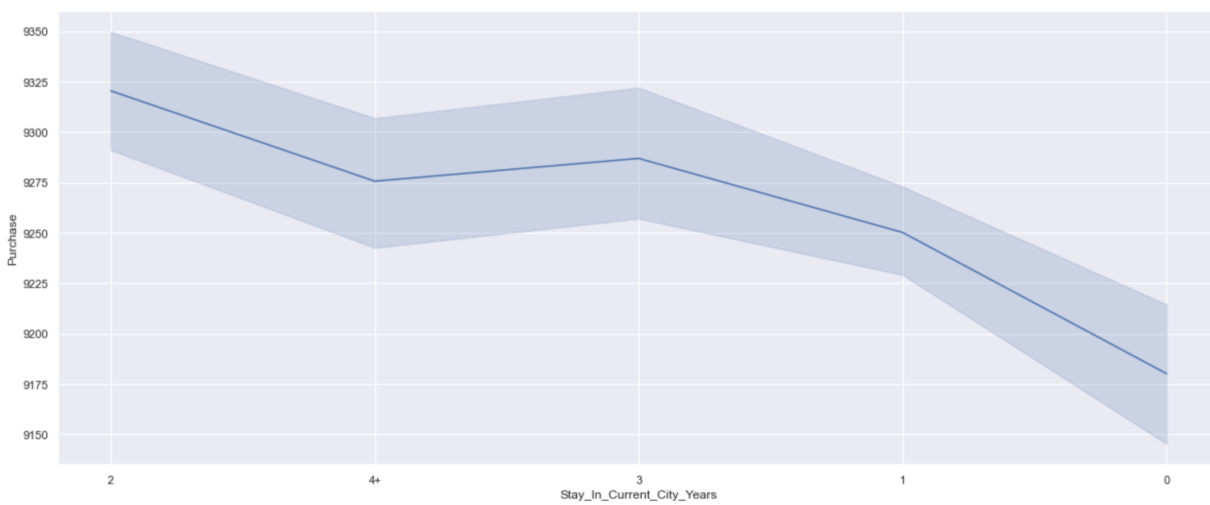
In [106]:
```
sns.heatmap(df_male.corr(),annot=True)
sns.set(rc={"figure.figsize":(20, 8)})
```



In [107]:
```
# an important insight can be that marital status has an negative effect on puchc
```

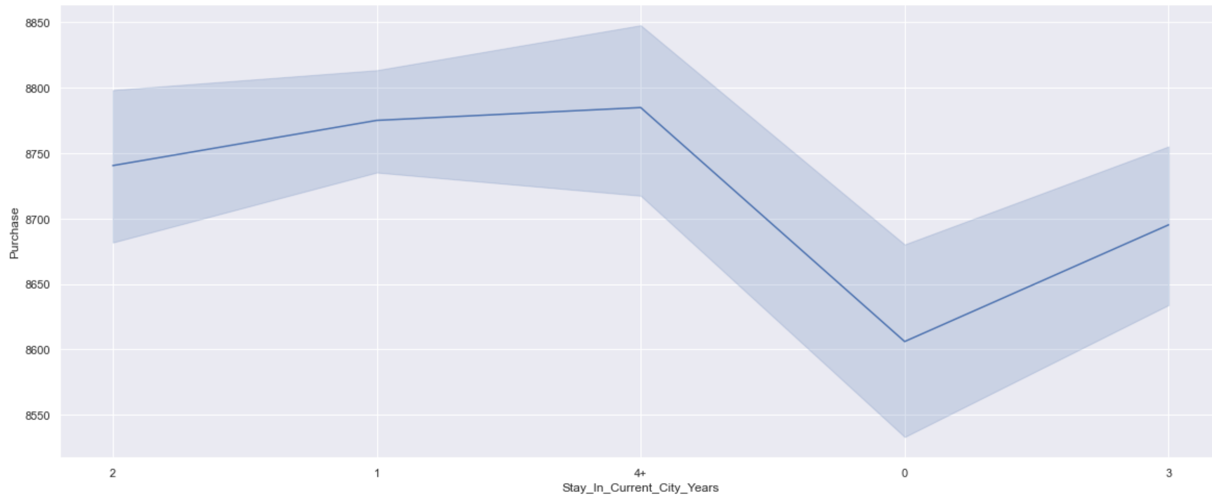
In [112]:
```
sns.lineplot(x='Stay_In_Current_City_Years',y='Purchase',data=df)
# for new people in the city i.e 0 or 1 years are very less contributing to purch
```

Out[112]: <AxesSubplot:xlabel='Stay_In_Current_City_Years', ylabel='Purchase'>



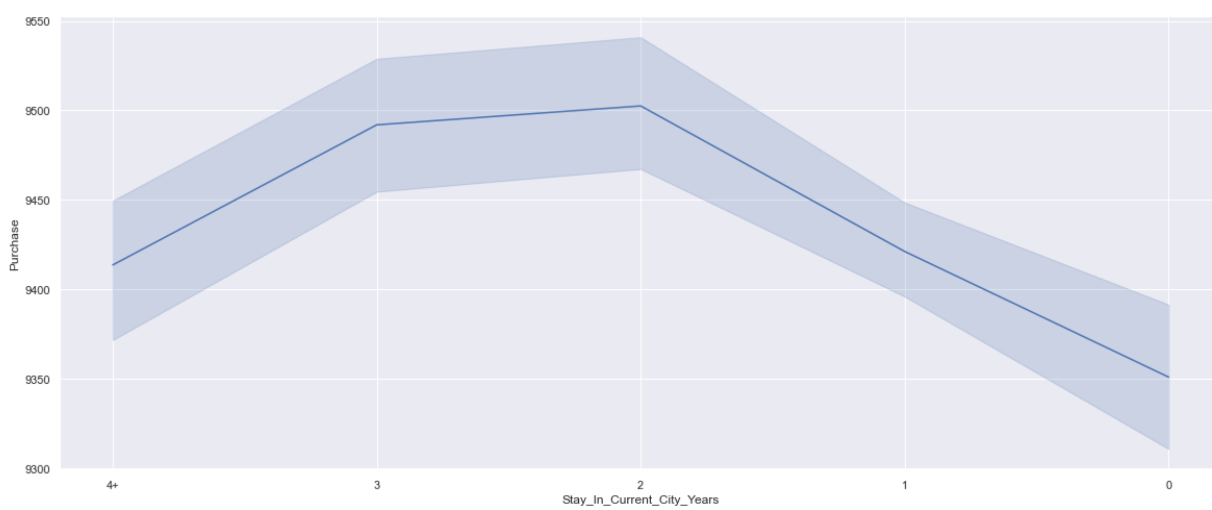
In [110]:
```
sns.lineplot(x='Stay_In_Current_City_Years',y='Purchase',data=df_female)
# walmart can focus more on the female population that is new to the city
```

Out[110]: <AxesSubplot:xlabel='Stay_In_Current_City_Years', ylabel='Purchase'>



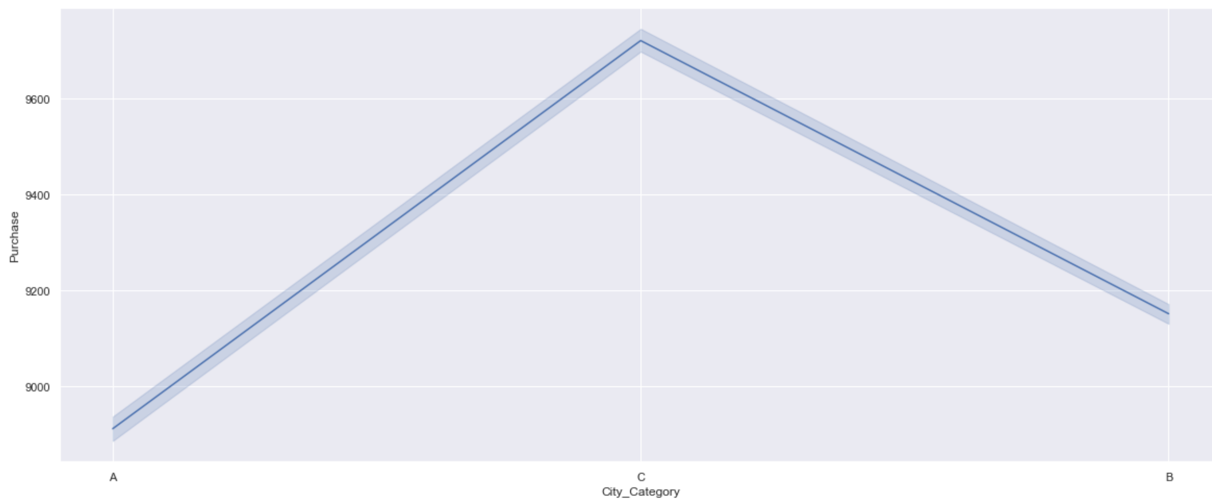
In [111]:
```
sns.lineplot(x='Stay_In_Current_City_Years',y='Purchase',data=df_male)
# walmart can focus more on the male population that is new to the city
```

Out[111]: <AxesSubplot:xlabel='Stay_In_Current_City_Years', ylabel='Purchase'>



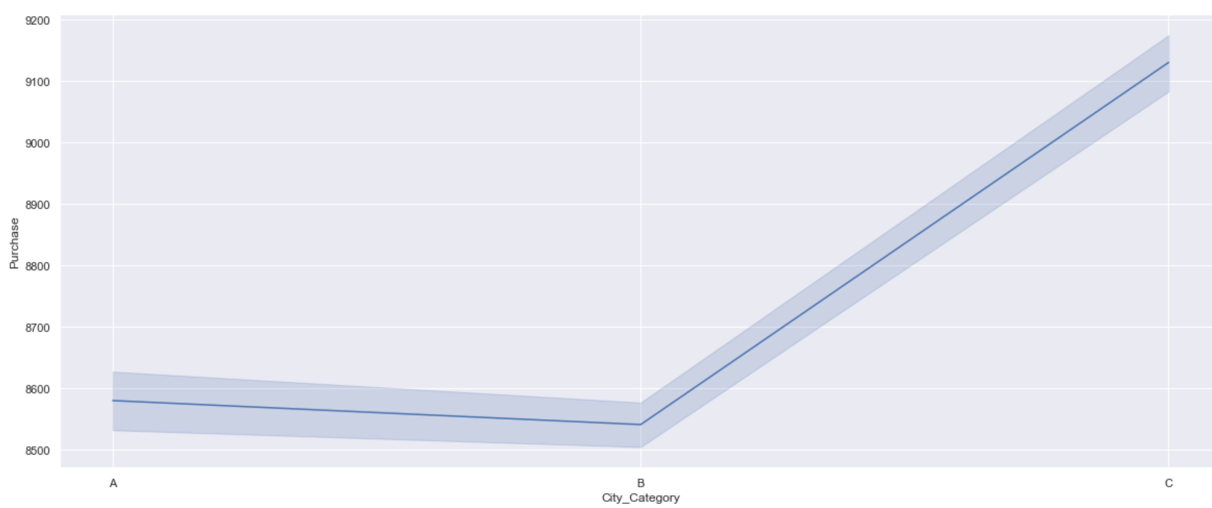
In [117]:
```
sns.lineplot(x='City_Category',y='Purchase',data=df)
# city category wise puchase contribution
#insights : Purchase per city category B is high , while city category A and C ar
```

Out[117]: <AxesSubplot:xlabel='City_Category', ylabel='Purchase'>



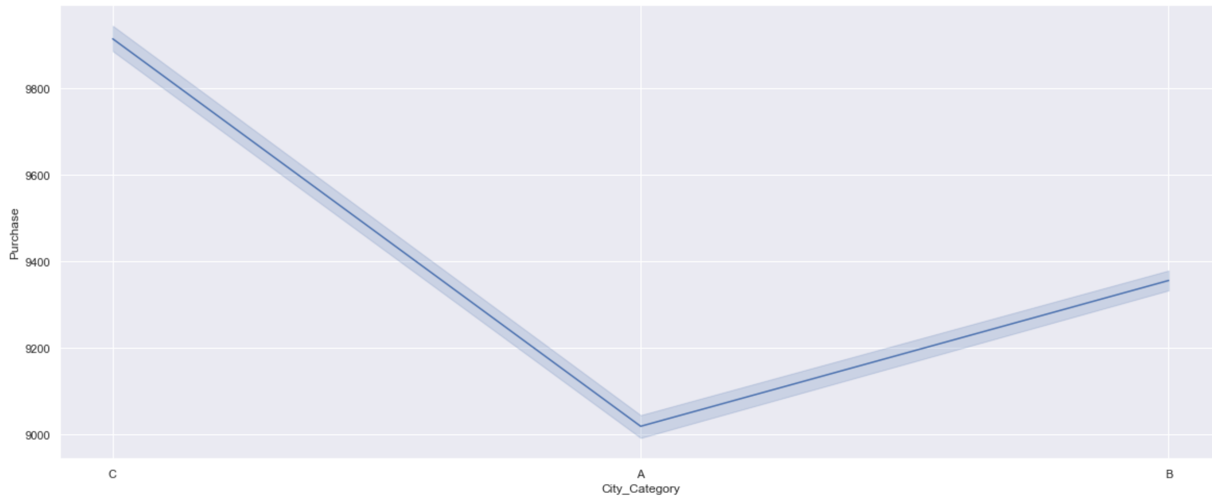
In [118]:
```
sns.lineplot(x='City_Category',y='Purchase',data=df_female)
# city category wise female puchase contribution
#insights : Purchase per city category B is high , while city category A and C ar
```

Out[118]: <AxesSubplot:xlabel='City_Category', ylabel='Purchase'>



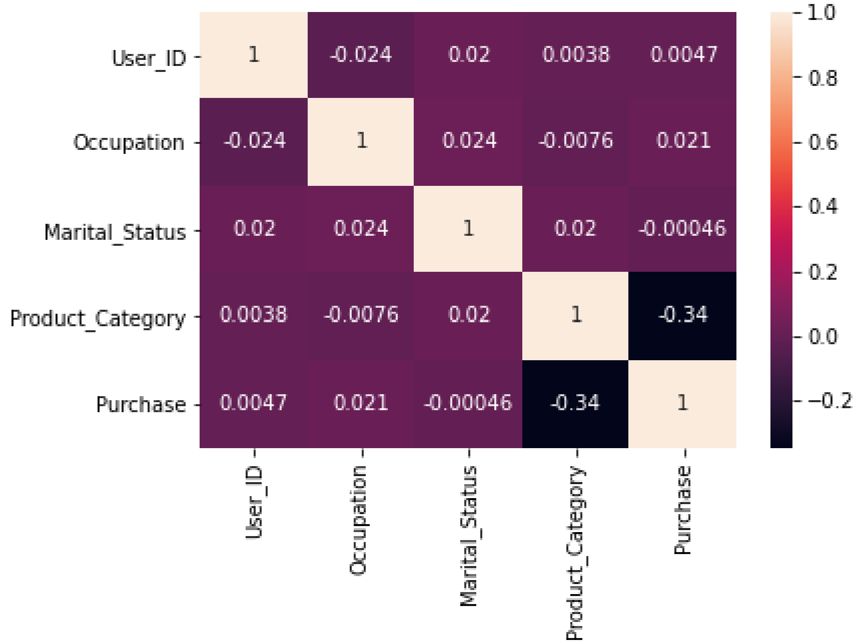
In [119]:
```
sns.lineplot(x='City_Category',y='Purchase',data=df_male)
# city category wise male puchase contribution
#insights : Purchase per city category B is high , while city category A and C ar
```

Out[119]: <AxesSubplot:xlabel='City_Category', ylabel='Purchase'>

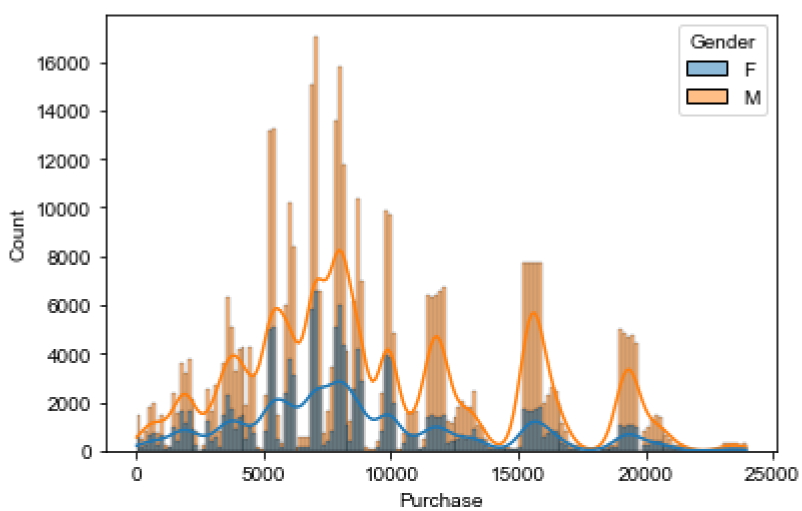


In [74]:
```
# relation between different fields can be found below
sns.heatmap(df.corr(),annot=True)
# insight can be Purchase is negative correlated with the martial status
```

Out[74]: <AxesSubplot:>



In [75]:
```
# Purchase plot of male vs female
sns.histplot(x='Purchase',data=df,kde=True,hue='Gender')
sns.set(rc={"figure.figsize":(20, 6)})
# it can be seen that female purchase is always less than male purchase.
```

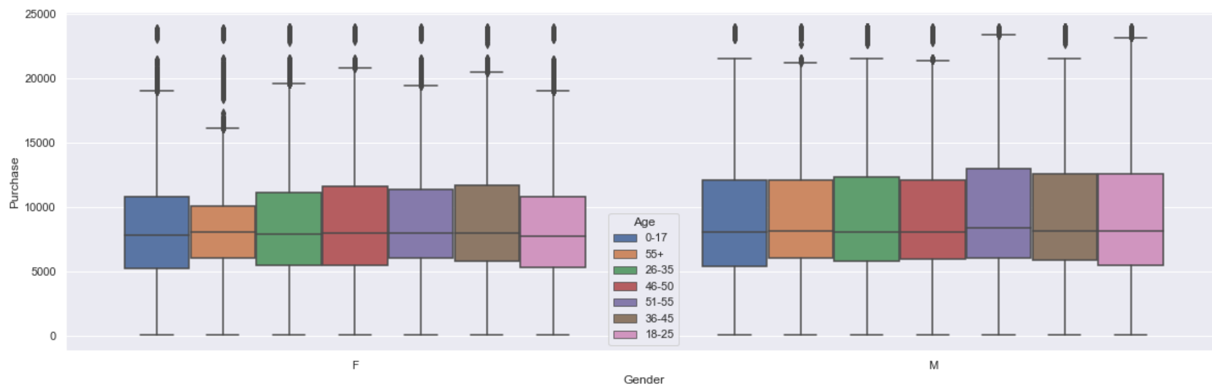


In [76]:

Out[76]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

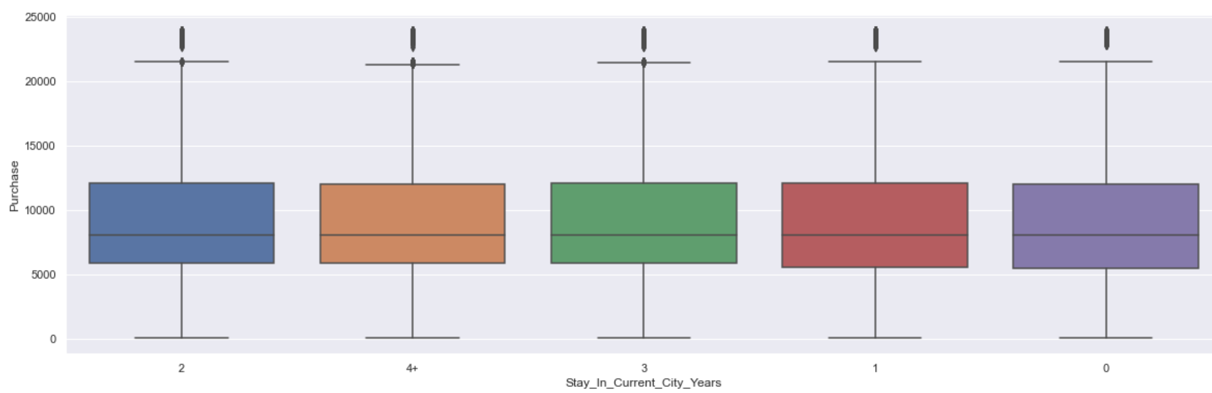
In [79]:
```
sns.boxplot(x='Gender',y='Purchase',hue='Age',data=df)
#The 50% of the purchase is fairly same for both male and female
# but the upper bound of male is slightly higher than female
```

Out[79]: <AxesSubplot:xlabel='Gender', ylabel='Purchase'>



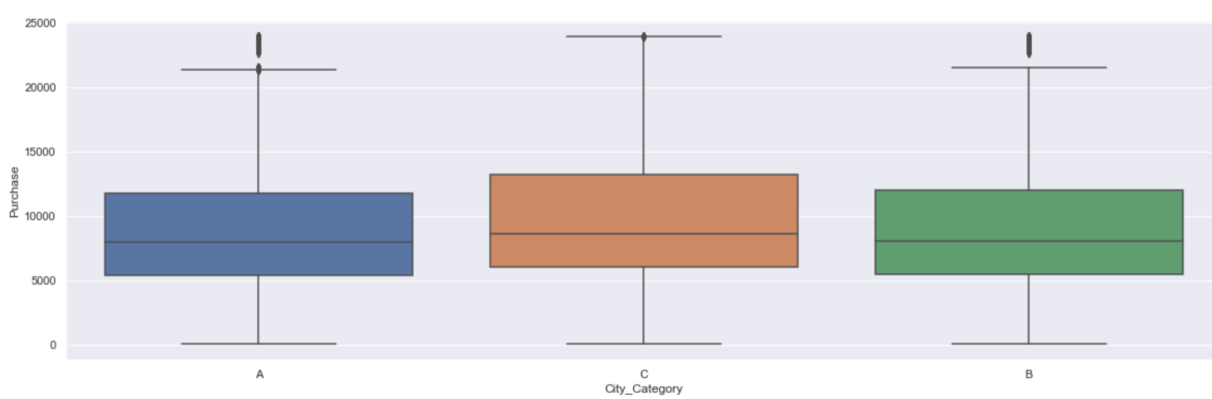
In [81]:
```
sns.boxplot(x='Stay_In_Current_City_Years',y='Purchase',data=df)
```

Out[81]: <AxesSubplot:xlabel='Stay_In_Current_City_Years', ylabel='Purchase'>



In [83]:
```
sns.boxplot(x='City_Category',y='Purchase',data=df)
```

Out[83]: <AxesSubplot:xlabel='City_Category', ylabel='Purchase'>



In [ ]: