About Walmart

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

Business Problem

The Management team at Walmart Inc. wants to analyze the customer purchase behavior (specifically, purchase amount) against the customer's gender and the various other factors to help the business make better decisions. They want to understand if the spending habits differ between male and female customers: Do women spend more on Black Friday than men? (Assume 50 million customers are male and 50 million are female).

Dataset

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

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

In [1]:

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

In [2]:
```
dt = pd.read_csv('walmart_data.csv')
dt.head()
```

Out[2]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	7969

Datatype of all Columns

In [3]:
```
dt.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
```

Converting Product Category, Marital Status Occupation and User_ID to Object type from Integer type

In [4]:
```
dt['Product_Category'] = dt['Product_Category'].astype(object)
dt['Marital_Status'] = dt['Marital_Status'].astype(object)
dt['Occupation'] = dt['Occupation'].astype(object)
dt['User_ID'] = dt['User_ID'].astype(object)
```

Data Shape

In [5]:
```
print('The Dataset has {0} rows and {1} columns'.format(dt.shape[0], dt.shape[1]))
```

```
The Dataset has 550068 rows and 10 columns
```

Statistical Summary

In [6]:
```
# Describing numerical Columns
dt.describe()
```

Out[6]:

	Purchase
count	550068.000000
mean	9263.968713
std	5023.065394
min	12.000000
25%	5823.000000
50%	8047.000000
75%	12054.000000
max	23961.000000

In [7]:
```
# Describing Object type Columns
dt.describe(include = 'object')
```

Out[7]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category
count	550068	550068	550068	550068	550068	550068	550068	550068	550068

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category
unique	5891	3631	2	7	21	3	5	2	20
top	1001680	P00265242	M	26-35	4	B	1	0	5

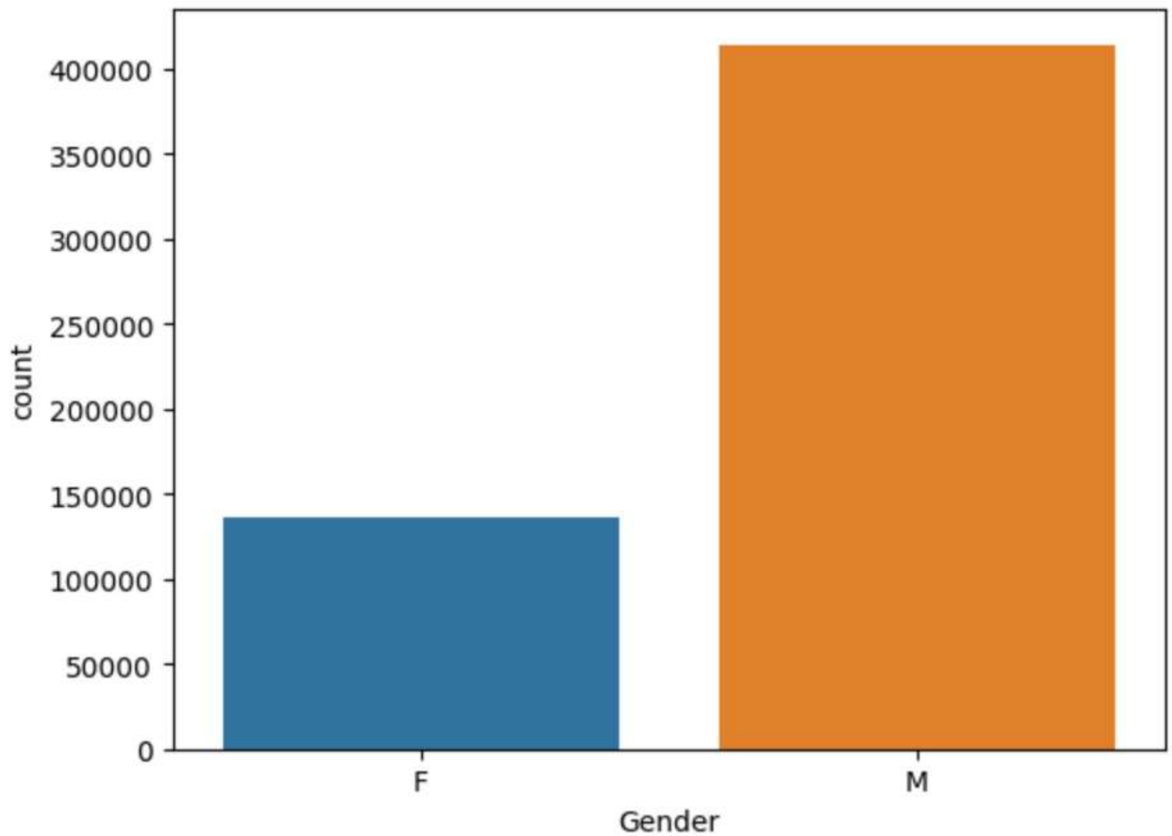
Unique Values and Value Counts

Gender

In [8]:
```
dt['Gender'].value_counts()
```

Out[8]:
```
M    414259
F    135809
Name: Gender, dtype: int64
```

In [9]:
```
sns.countplot(x = 'Gender', data = dt)
plt.show()
```



Insights -

In [10]:
```
male_percent = (dt['Gender'].value_counts()['M']/dt.shape[0])*100
female_percent = (dt['Gender'].value_counts()['F']/dt.shape[0])*100
print('{0}% of the customers are Male and {1}% of the customers are Female'.format(round(male_percent, 2), round(femal
```

```
75.31% of the customers are Male and 24.69% of the customers are Female
```

Age

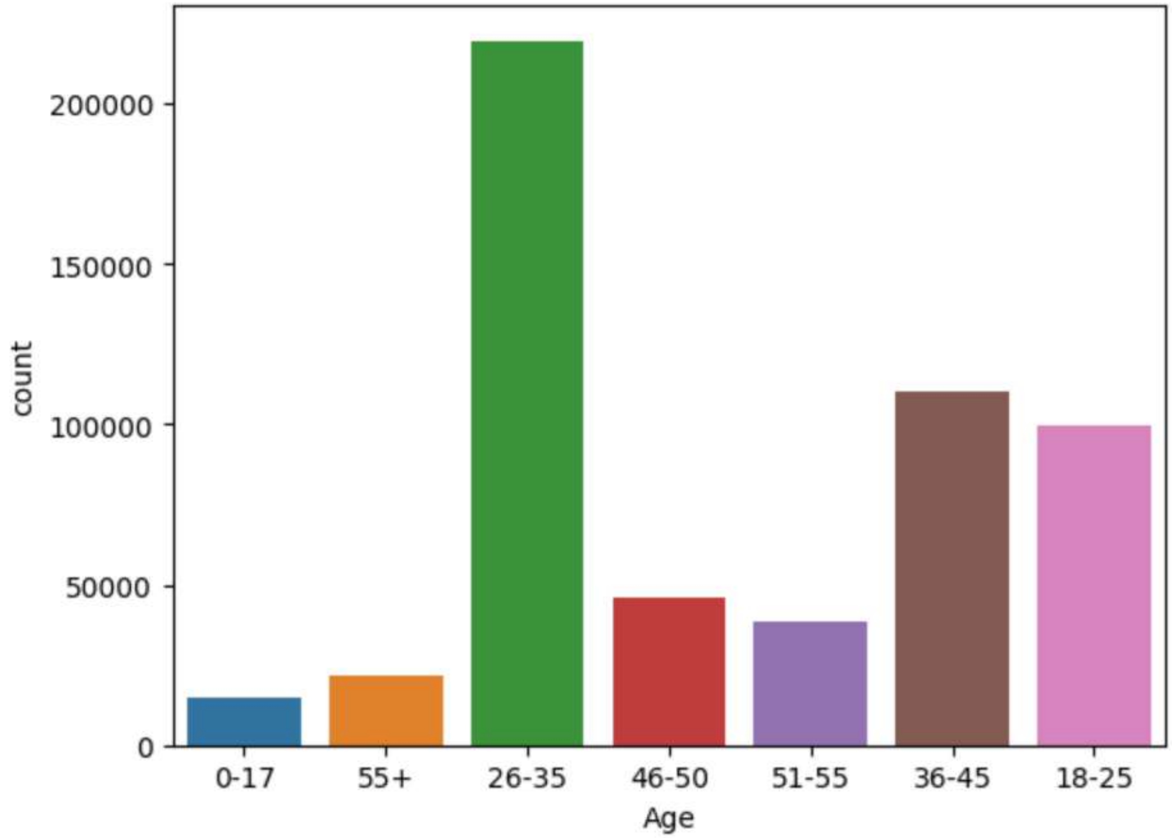
In [11]:
```
dt['Age'].value_counts()
```

Out[11]:
```
26-35    219587
36-45    110013
18-25     99660
46-50     45701
```

```
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
```

In [12]:

```python
sns.countplot(x = 'Age', data = dt)
plt.show()
```



In [13]:

```python
dt['Age'].value_counts(normalize = True)*100
```

Out[13]:

```
26-35    39.919974
36-45    19.999891
18-25    18.117760
46-50     8.308246
51-55     6.999316
```

```
55+        3.909335
0-17       2.745479
```

Insights -

Most of the customers are from 26-35 age group, followed by 36-45 and 18-25.

City Category

In [14]:

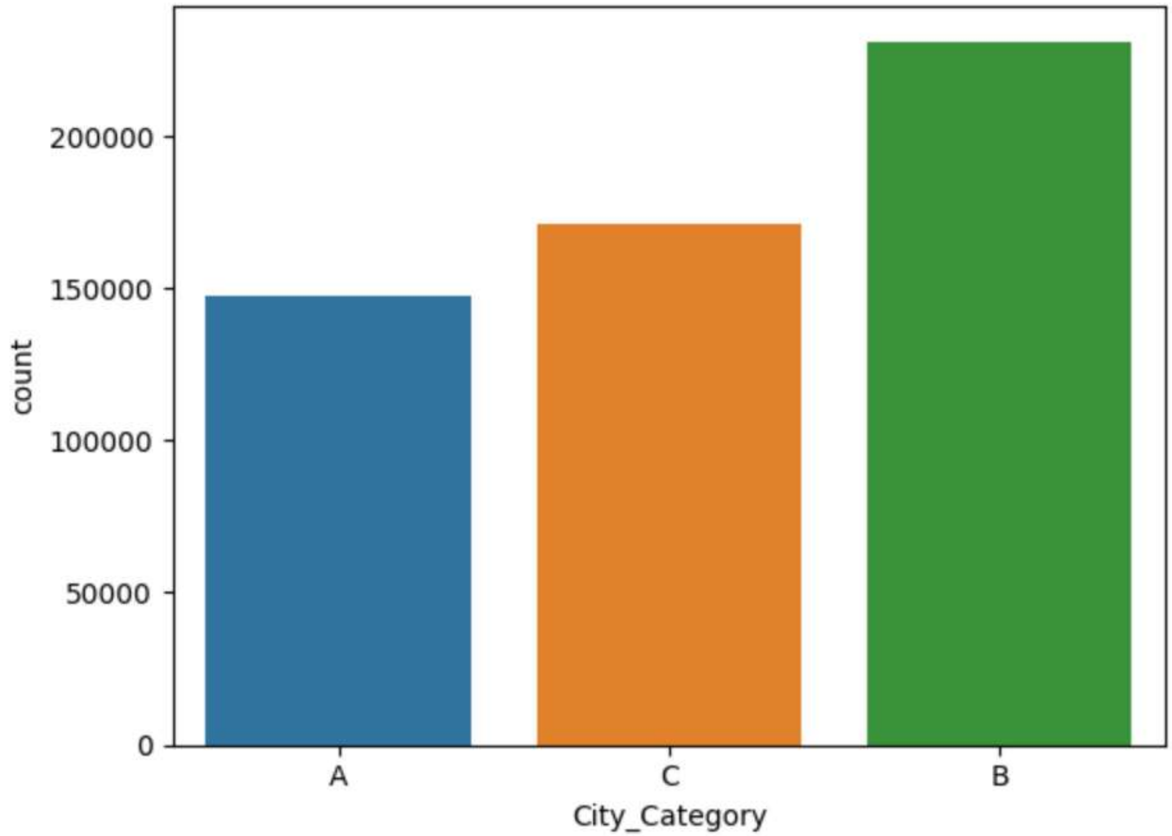
```
dt['City_Category'].value_counts()
```

Out[14]:

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

In [15]:

```
sns.countplot(x = 'City_Category', data = dt)
plt.show()
```



In [17]:

```
# City Percentages
dt['City_Category'].value_counts(normalize = True)*100
```

Out[17]:

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

Stay_In_Current_City_Years

In [18]:

```
dt['Stay_In_Current_City_Years'].value_counts()
```

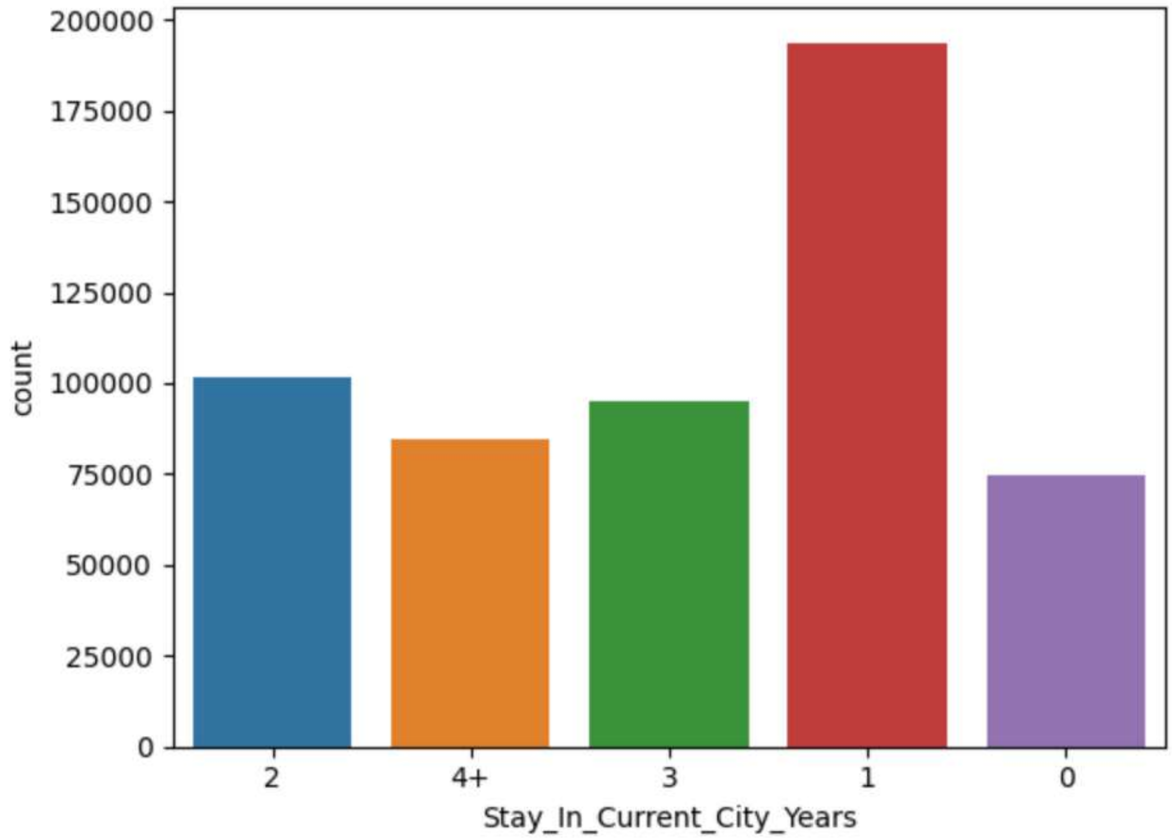
Out[18]:

```
1     193821
2     101838
3      95285
```

```
4+      84726
0       74398
Name: Stay_In_Current_City_Years, dtype: int64
```

In [19]:

```python
sns.countplot(x = 'Stay_In_Current_City_Years', data = dt)
plt.show()
```



In [20]:

```python
# Percentage of Years of Stay
dt['Stay_In_Current_City_Years'].value_counts(normalize = True)*100
```

Out[20]:

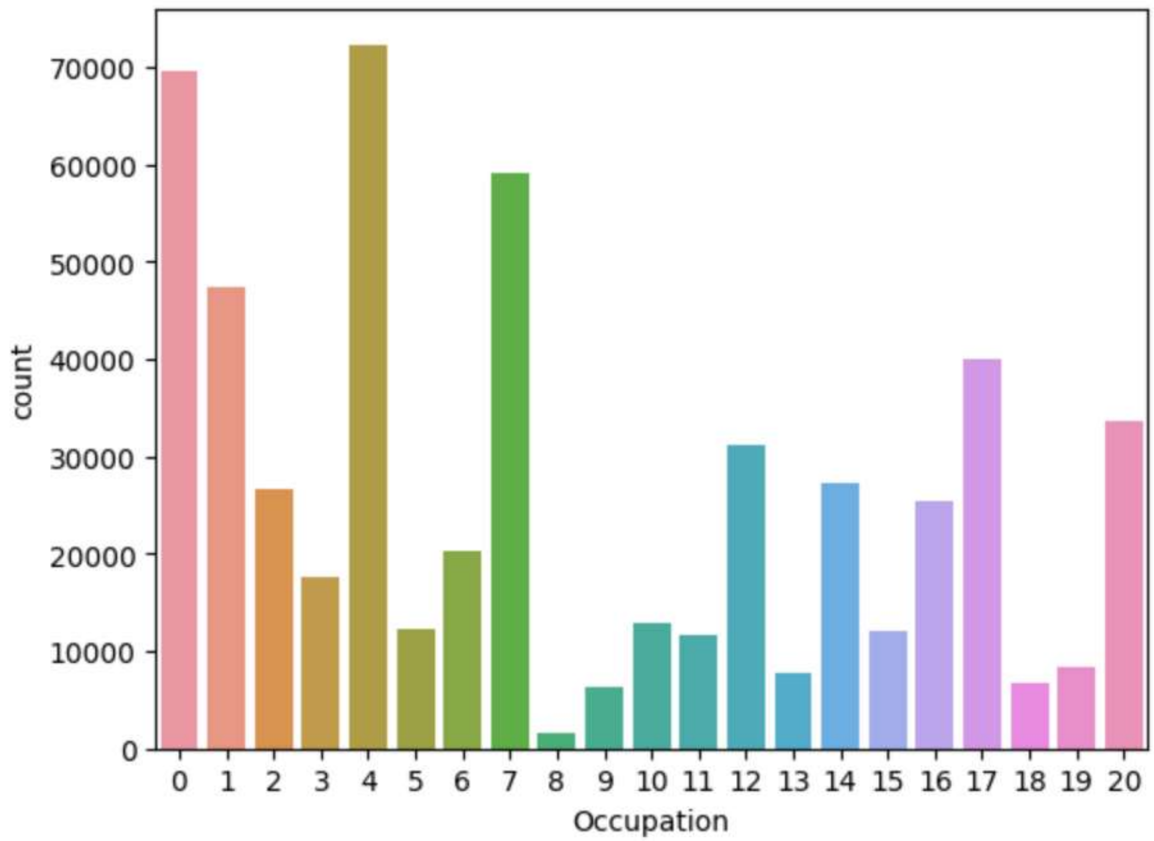
```
1     35.235825
2     18.513711
3     17.322404
4+    15.402823
0     13.525237
```

Occupation

In [21]:
```
dt['Occupation'].value_counts()
```

Out[21]:
```
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 [22]:
```
sns.countplot(x = 'Occupation', data = dt)
plt.show()
```



In [23]:

```
# Percentage of Occupation
dt['Occupation'].value_counts(normalize = True)*100
```

Out[23]:

```
4     13.145284
0     12.659889
7     10.750125
1      8.621843
17     7.279645
20     6.101427
12     5.668208
14     4.964659
2      4.833584
16     4.612339
6      3.700452
3      3.208694
10     2.350618
```

```
5      2.213726
15     2.211545
11     2.106285
19     1.538173
13     1.404917
18     1.203851
9      1.143677
8      0.281056
Name: Occupation, dtype: float64
```

Marital Status

In [24]:

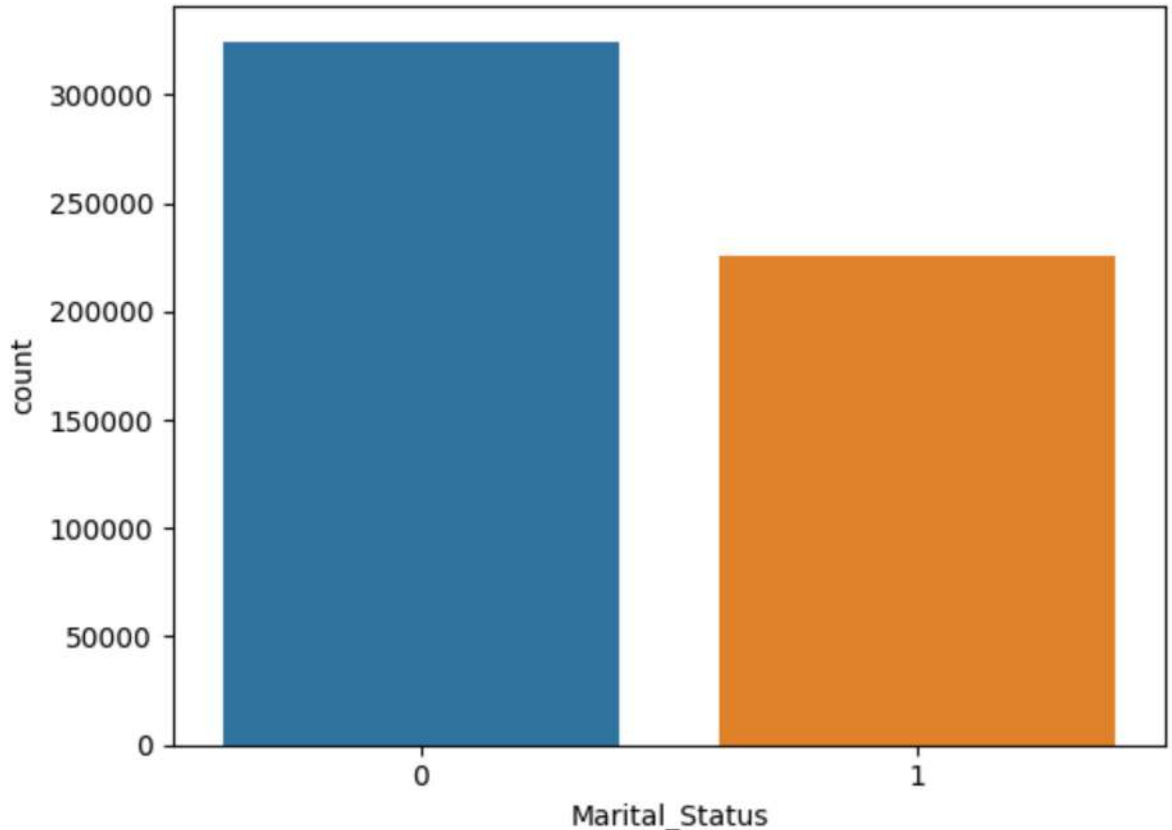
```
dt['Marital_Status'].value_counts()
```

Out[24]:

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

In [25]:

```
sns.countplot(x = 'Marital_Status', data = dt)
plt.show()
```



In [26]:
```
# Marital Status Percentages
dt['Marital_Status'].value_counts(normalize = True)*100
```

Out[26]:
```
0    59.034701
1    40.965299
Name: Marital_Status, dtype: float64
```

Product Category

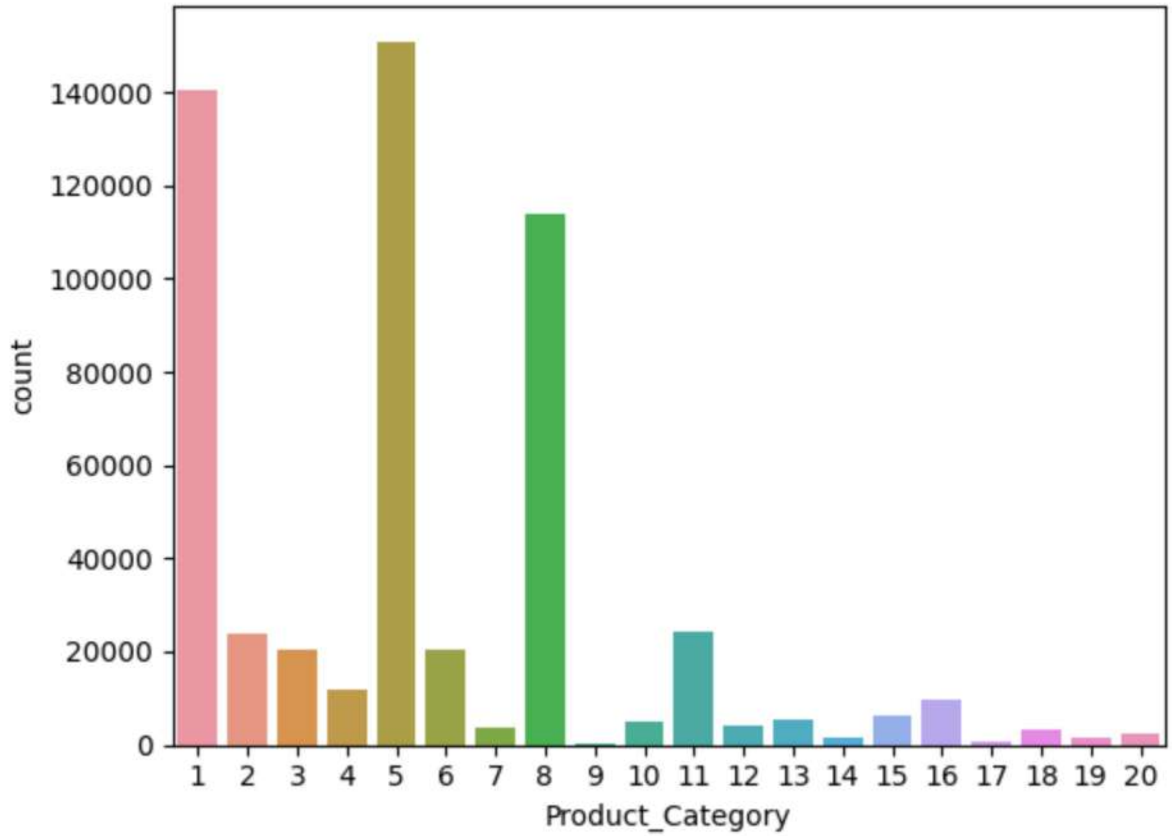
In [27]:
```
dt['Product_Category'].value_counts()
```

Out[27]:
```
5     150933
1     140378
8     113925
11     24287
```

```
2      23864
6      20466
3      20213
4      11753
16      9828
15      6290
13      5549
10      5125
12      3947
7       3721
18      3125
20      2550
19      1603
14      1523
17       578
9        410
Name: Product_Category, dtype: int64
```

In [28]:

```
sns.countplot(x = 'Product_Category', data = dt)
plt.show()
```



In [29]:

```
# Product Category Percentages
dt['Product_Category'].value_counts(normalize = True)*100
```

Out[29]:

```
5     27.438971
1     25.520118
8     20.711076
11     4.415272
2      4.338373
6      3.720631
3      3.674637
4      2.136645
16     1.786688
15     1.143495
13     1.008784
10     0.931703
12     0.717548
```

```
7       0.676462
18      0.568112
20      0.463579
19      0.291419
14      0.276875
17      0.105078
9       0.074536
Name: Product_Category, dtype: float64
```

Purchases -- Comments on Range of Variables

In [30]:

```
print('Minimum purchase =', dt['Purchase'].min())
print('Median purchase =', round(dt['Purchase'].median(), 2))
print('Mean purchase =', round(dt['Purchase'].mean(), 2))
print('Maximum purchase =', dt['Purchase'].max())
```

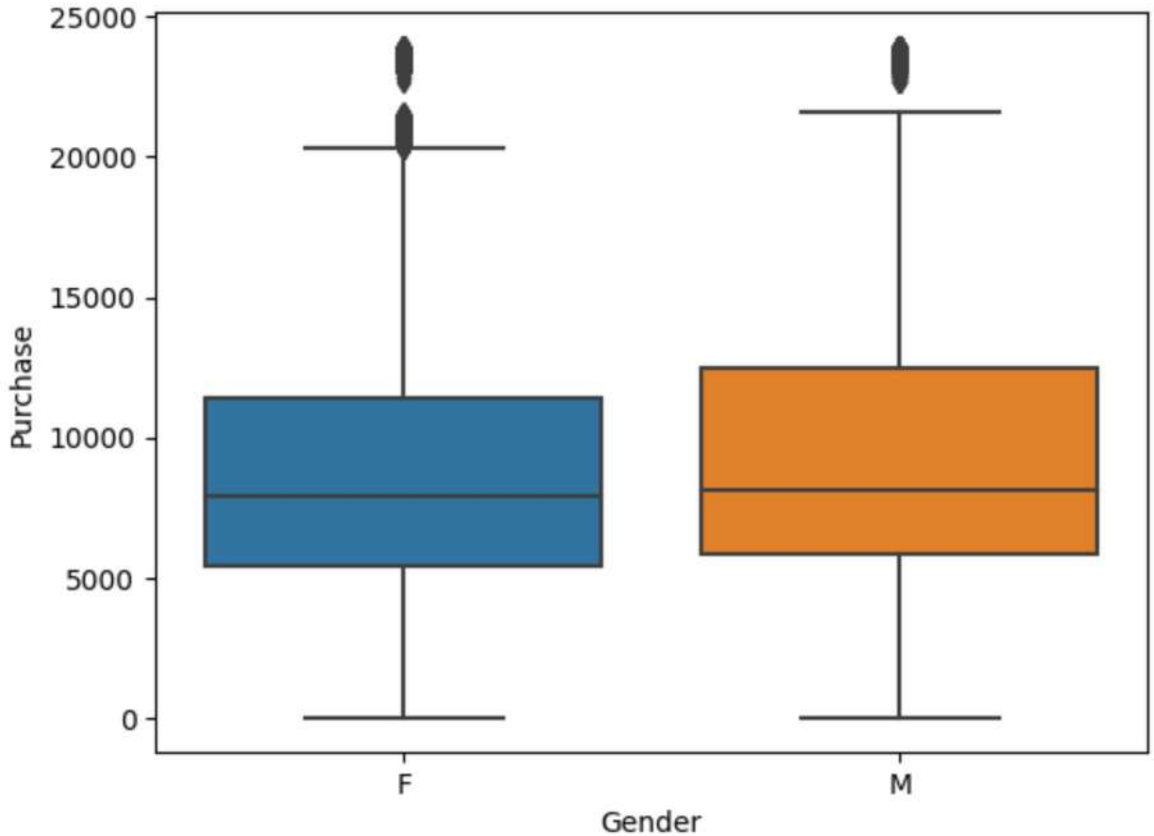
```
Minimum purchase = 12
Median purchase = 8047.0
Mean purchase = 9263.97
Maximum purchase = 23961
```

Bivariate Analysis

Purchases vs Gender

In [31]:

```
sns.boxplot(y = dt['Purchase'], x = dt['Gender'])
plt.show()
```

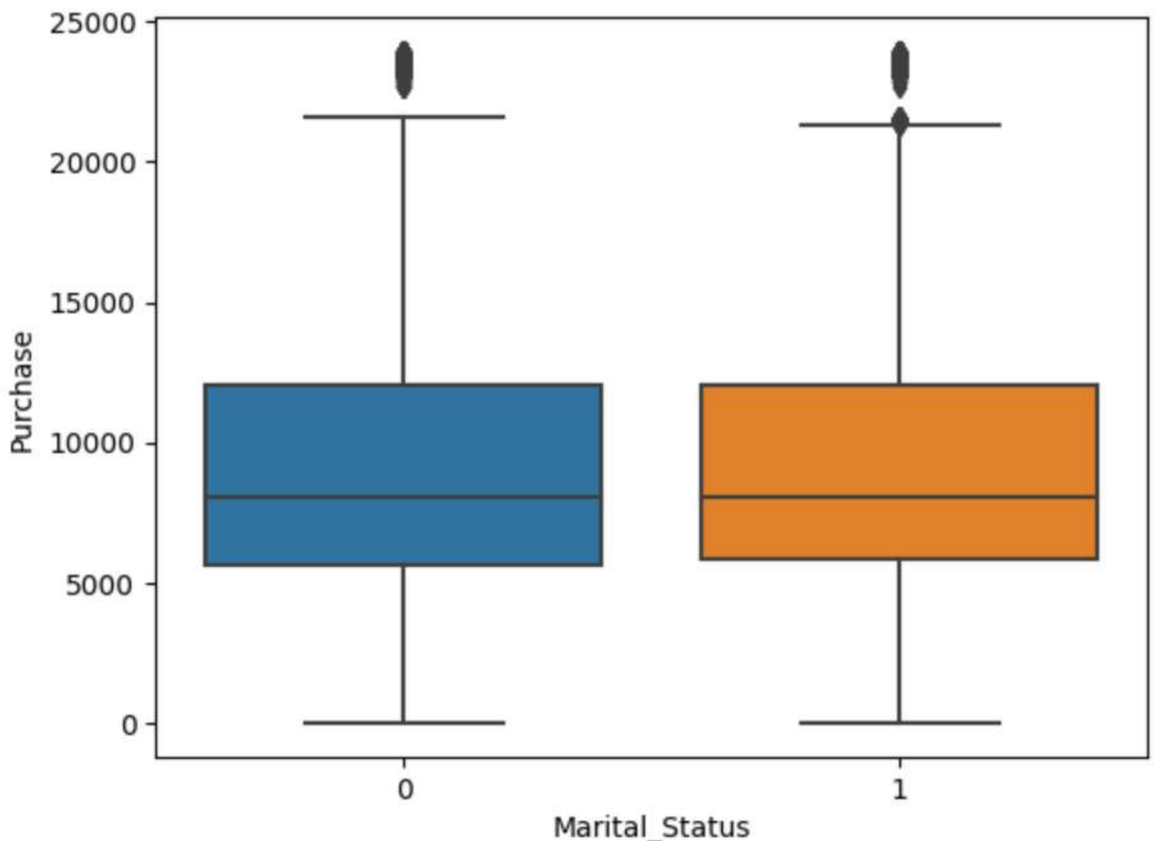


In [32]:
```
#dt['Purchase']
female_iqr = np.percentile(dt[dt['Gender'] == 'F']['Purchase'], 75) - np.percentile(dt[dt['Gender'] == 'F']['Purchase'

male_iqr = np.percentile(dt[dt['Gender'] == 'M']['Purchase'], 75) - np.percentile(dt[dt['Gender'] == 'M']['Purchase'],

print('IQR of Purchases of Male =',male_iqr, 'and Female =', female_iqr)
```

```
IQR of Purchases of Male = 6591.0 and Female = 5967.0
```

Marital Status vs Purchases

In [33]:
```
sns.boxplot(y = dt['Purchase'], x = dt['Marital_Status'])
plt.show()
```



In [34]:

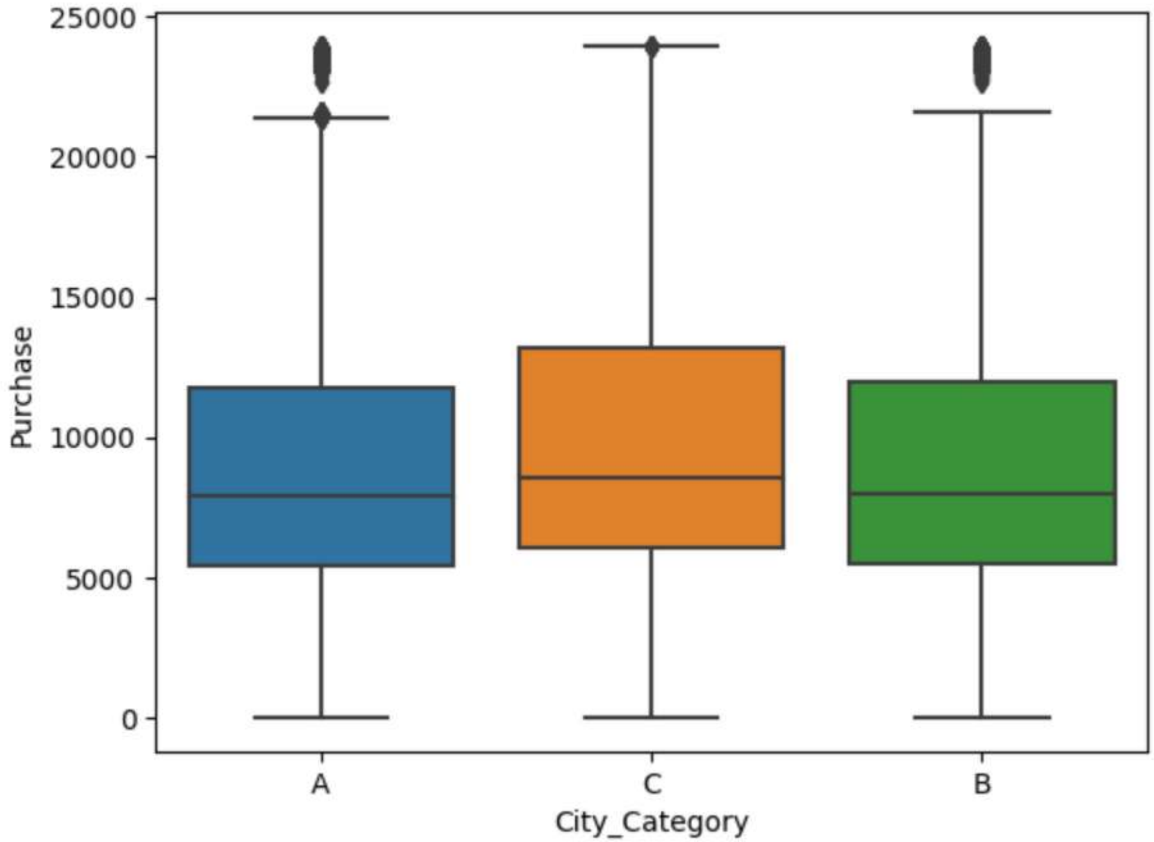
```
#dt['Purchase']
female_iqr = np.percentile(dt[dt['Marital_Status'] == 1]['Purchase'], 75) - np.percentile(dt[dt['Marital_Status'] == 1

male_iqr = np.percentile(dt[dt['Marital_Status'] == 0]['Purchase'], 75) - np.percentile(dt[dt['Marital_Status'] == 0][

print('IQR of Purchases of Male =',male_iqr, 'and Female =', female_iqr)
```

```
IQR of Purchases of Male = 6456.0 and Female = 6199.0
```

Purchases vs City Category

In [35]:

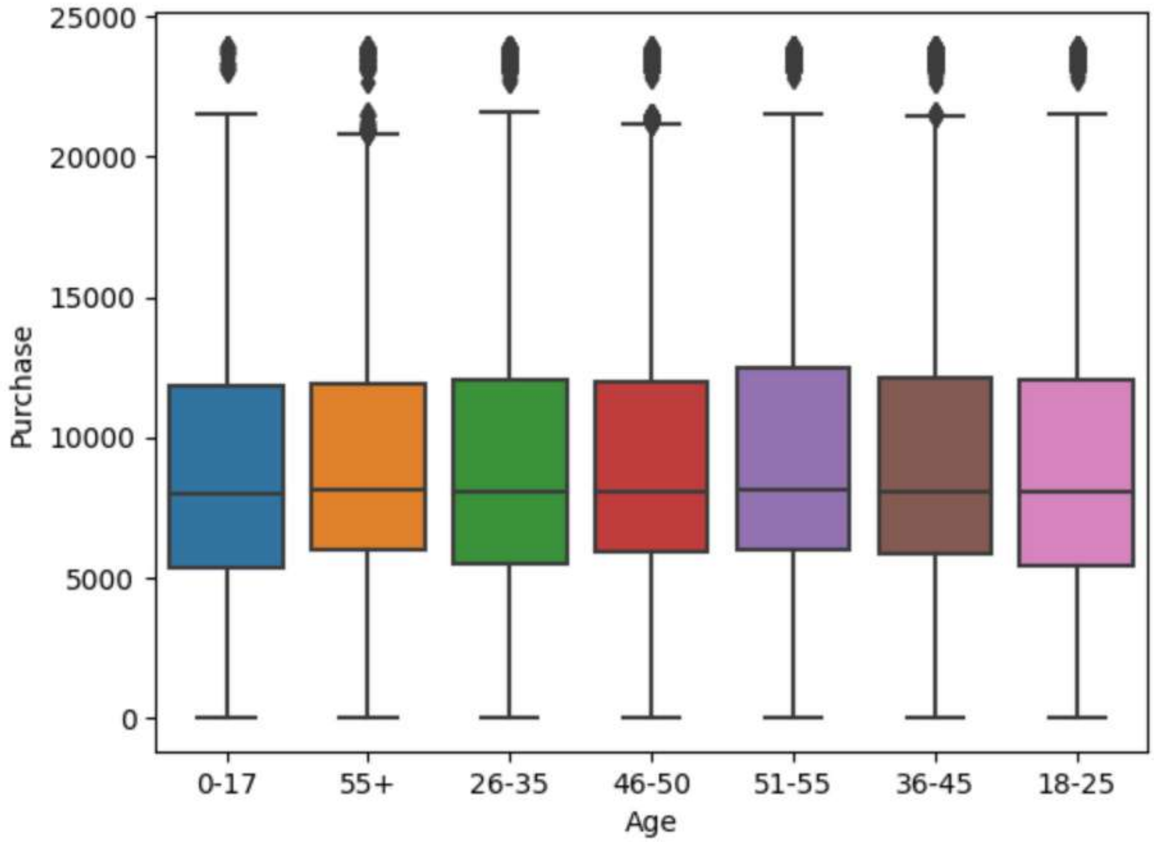
```
sns.boxplot(y = dt['Purchase'], x = dt['City_Category'])
plt.show()
```



Purchases vs Age

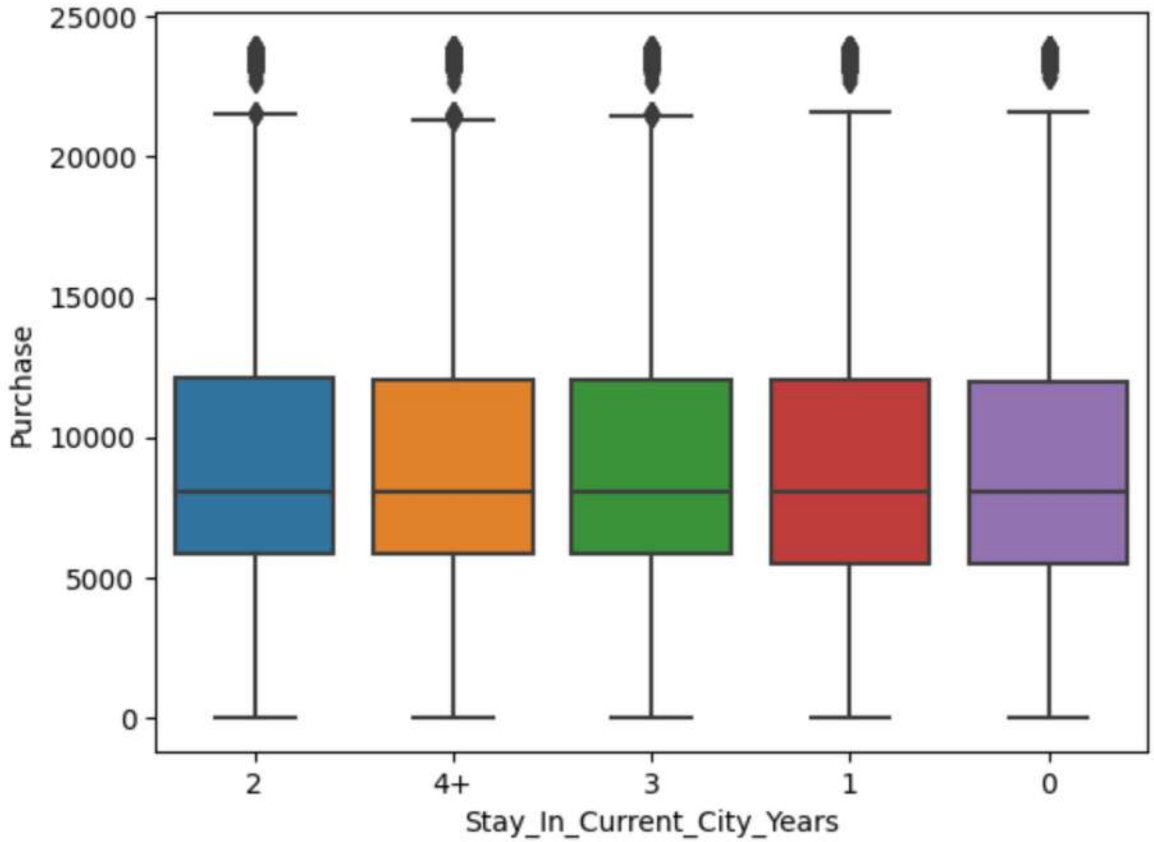
In [36]:

```
sns.boxplot(y = dt['Purchase'], x = dt['Age'])
plt.show()
```



Purchases vs Stay in current city year

In [37]:
```
sns.boxplot(y = dt['Purchase'], x = dt['Stay_In_Current_City_Years'])
plt.show()
```



Sum of Null values/ Missing in each Columns

In [38]:

```
dt.isna().sum()
```

Out[38]:

```
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
```

Popular Product in grouped Gender, Age and City Category

In [67]:
```
def popular_product(df):
    return df['Product_Category'].value_counts(ascending = False).head(1)
```

In [76]:
```
df = dt.groupby(['Gender', 'Age', 'City_Category']).apply(popular_product)
df = pd.DataFrame(df)
df
```

Out[76]:

Gender	Age	City_Category		Product_Category
F	0-17	A	5	447
		B	5	440
		C	5	624
	18-25	A	5	2085
		B	5	3809
		C	5	2034
	26-35	A	5	5790
		B	5	7105
		C	5	3691
	36-45	A	5	2070
		B	5	3233
		C	5	2514
	46-50	A	5	418
		B	8	1760
		C	5	1584
	51-55	A	5	575

				Product_Category
Gender	**Age**	**City_Category**		
		B	**8**	1307
		C	**5**	1115
	55+	**A**	**8**	109
		B	**8**	459
		C	**8**	1210
M	**0-17**	**A**	**5**	300
		B	**1**	1047
		C	**5**	1500
	18-25	**A**	**5**	5915
		B	**1**	9138
		C	**1**	7300
	26-35	**A**	**5**	15770
		B	**1**	20176
		C	**1**	13943
	36-45	**A**	**5**	5185
		B	**5**	9959
		C	**1**	7876
	46-50	**A**	**5**	1573
		B	**5**	3679
		C	**1**	3090
	51-55	**A**	**5**	1119
		B	**1**	3433
		C	**1**	2966
	55+	**A**	**5**	872

Insights

After doing groupby with respect to Gender, Age Groups and City Categories, the most popular products are 5, 8 and 1

In [19]:

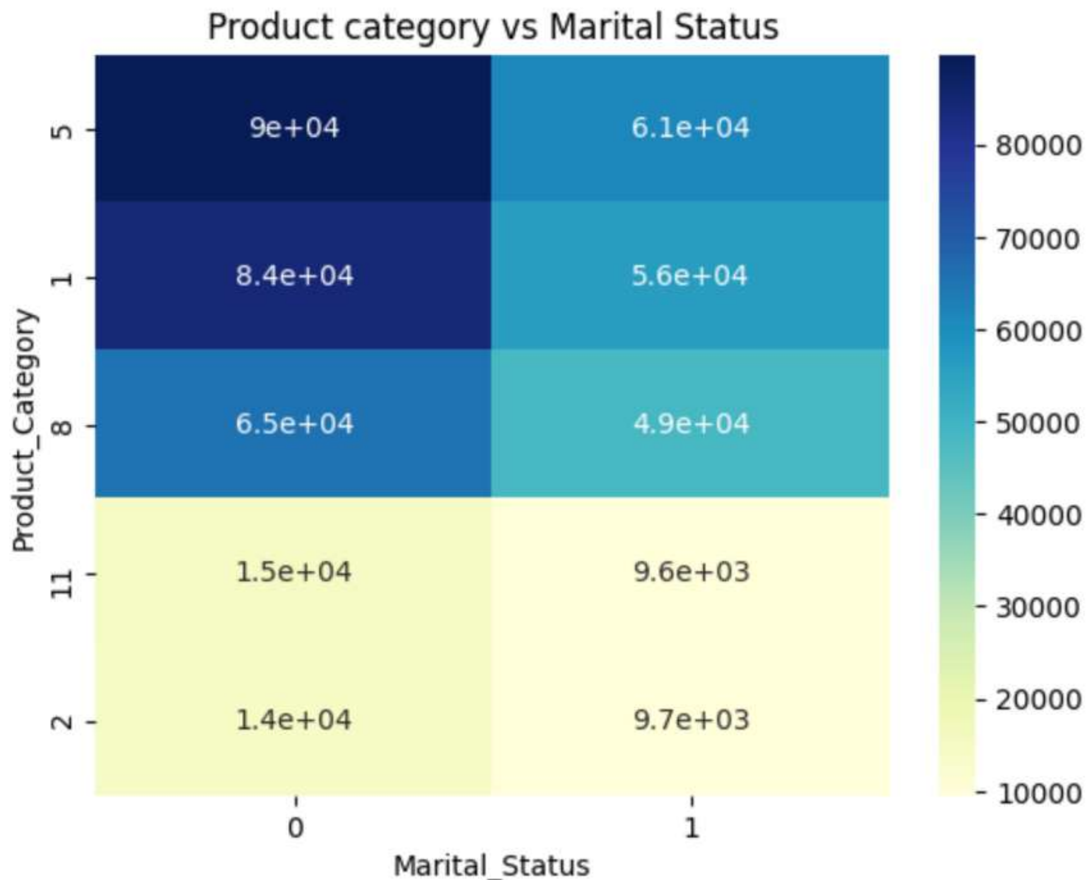
```
a = pd.crosstab(dt['Product_Category'], dt['Marital_Status']).sort_values([0, 1], ascending = False).head(5)
a
```

Out[19]:

Marital_Status / Product_Category	0	1
5	89656	61277
1	84375	56003
8	65411	48514
11	14668	9619
2	14138	9726

In [26]:

```
sns.heatmap(a, cmap = "YlGnBu", annot=True)
plt.title('Product category vs Marital Status')
plt.show()
```



Possible Outliers

In [39]:

```
q75, q25 = np.percentile(dt['Purchase'], [75 ,25])
iqr = q75 - q25

upper_limit = q75 + 1.5*iqr
lower_limit = 0 if 0>(q25 - 1.5*iqr) else (q25 - 1.5*iqr)

print('The upper limit = ', upper_limit, 'and lower limit of the Purchase is =', lower_limit, '(Trunctaed to Zero)' )
```

```
The upper limit =  21400.5 and lower limit of the Purchase is = 0 (Trunctaed to Zero)
```

In [39]:
```
cond = (dt['Purchase']> upper_limit) | (dt['Purchase']< lower_limit)
dt.loc[cond, :].shape
```

Out[39]: (2677, 10)

- There are 2677 possible Outliers present in the Dataset in terms of Purchases

Analysing the data to Answer the Questions -

Comparing Purchases of Male and Female

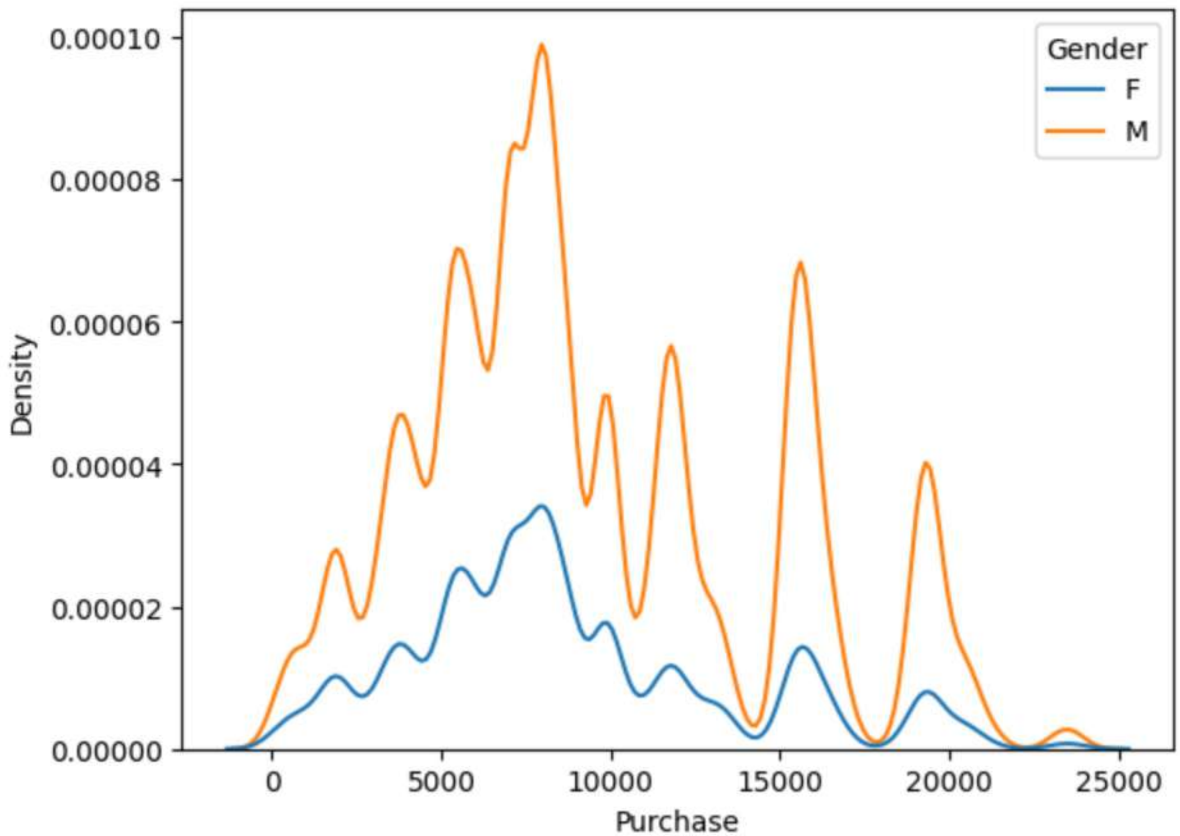
In [7]:
```
d = dt.groupby(['Gender'])['Purchase'].mean().sort_values(ascending = False)
print('Mean of Purchases of Males =', round(d['M'], 2))
print('Mean of Purchases of Females =', round(d['F'], 2))
```

```
Mean of Purchases of Males = 9437.53
Mean of Purchases of Females = 8734.57
```

In [8]:
```
d = dt.groupby(['Gender'])['Purchase'].median().sort_values(ascending = False)
print('Median of Purchases of Males =', round(d['M'], 2))
print('Median of Purchases of Females =', round(d['F'], 2))
```

```
Median of Purchases of Males = 8098
Median of Purchases of Females = 7914
```

In [9]:
```
sns.kdeplot(x = 'Purchase', data = dt, hue = 'Gender')
plt.show()
```



In [10]:

```
pd.crosstab(dt['Product_Category'], dt['Gender'], normalize = 'columns').sort_values(['F', 'M'], ascending = False).he
```

Out[10]:

Gender Product_Category	F	M
5	0.308971	0.263053
8	0.247097	0.194002
1	0.182838	0.278925
3	0.044224	0.034295
2	0.041661	0.043948

In [11]:
```
female_top_product_category = pd.crosstab(dt['Product_Category'], dt['Gender'], normalize = 'columns').sort_values(['F
male_top_product_category = pd.crosstab(dt['Product_Category'], dt['Gender'], normalize = 'columns').sort_values(['M']

print("Female Top Product Categories = ", female_top_product_category)
print("Male Top Product Categories = ", male_top_product_category)
```

```
Female Top Product Categories =  [5 8 1 3 2]
Male Top Product Categories =  [ 1  5  8 11  2]
```

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

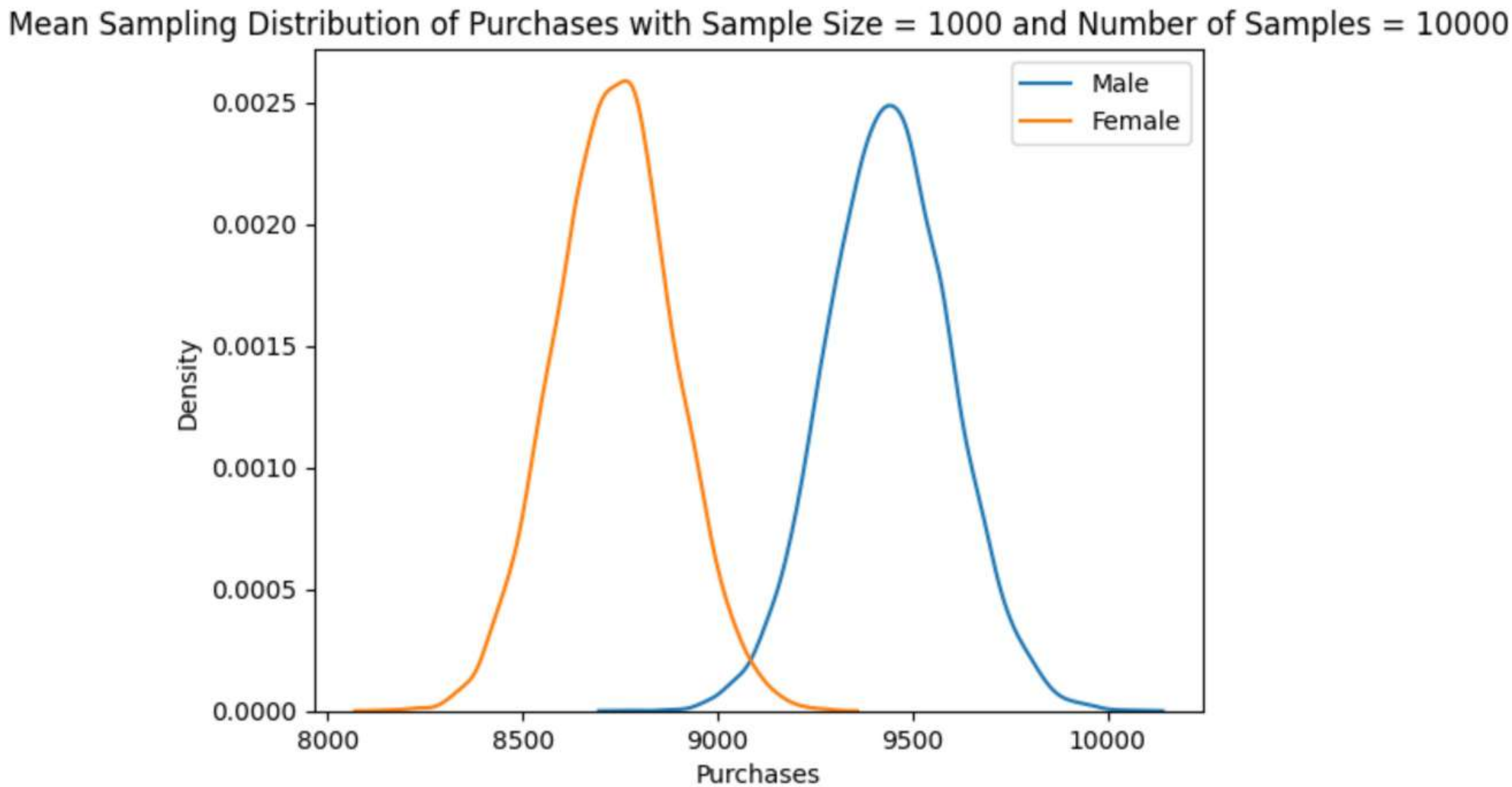
In [12]:
```
from numpy.random import sample # used to pick elements from Normal Distribution
from numpy.random import choice # used to pick elements from a given set, we do it with repetation
import random # We can use random.sample which will help us pick elements from a set without repetation
```

In [13]:
```
female_purchases = dt.loc[dt['Gender']=='F', :]['Purchase'].values
male_purchases = dt.loc[dt['Gender']=='M', :]['Purchase'].values
```

In [14]:
```
def bootstrap_sampling(data, sample_size, number_of_samples):
    sample_mean_list = []
    for _ in range(number_of_samples):
        sample_mean = round(np.mean(random.sample(data, sample_size)), 2)
        sample_mean_list.append(sample_mean)
    return sample_mean_list
```

In [15]:
```
bootstrapped_male_samples = bootstrap_sampling(data = list(male_purchases), sample_size = 1000, number_of_samples = 10
bootstrapped_female_samples = bootstrap_sampling(data = list(female_purchases), sample_size = 1000, number_of_samples
```

In [16]:
```
sns.kdeplot(bootstrapped_male_samples, label = 'Male')
sns.kdeplot(bootstrapped_female_samples, label = 'Female')
plt.title('Mean Sampling Distribution of Purchases with Sample Size = 1000 and Number of Samples = 10000')
plt.xlabel('Purchases')
plt.legend()
plt.show()
```



In [17]:

```
lower_limit = round(np.percentile(bootstrapped_male_samples, 5), 2)
upper_limit = round(np.percentile(bootstrapped_male_samples, 95), 2)
print('The 90% confidence interval of Male purchases is', lower_limit, 'to', upper_limit)

lower_limit = round(np.percentile(bootstrapped_female_samples, 2.5), 2)
upper_limit = round(np.percentile(bootstrapped_female_samples, 97.5), 2)
print('The 90% confidence interval of Female purchases is', lower_limit, 'to', upper_limit)
```

```
The 90% confidence interval of Male purchases is 9179.89 to 9703.28
The 90% confidence interval of Female purchases is 8439.08 to 9037.69
```

In [54]:
```
lower_limit = round(np.percentile(bootstrapped_male_samples, 2.5), 2)
upper_limit = round(np.percentile(bootstrapped_male_samples, 97.5), 2)
print('The 95% confidence interval of Male purchases is', lower_limit, 'to', upper_limit)

lower_limit = round(np.percentile(bootstrapped_female_samples, 2.5), 2)
upper_limit = round(np.percentile(bootstrapped_female_samples, 97.5), 2)
print('The 95% confidence interval of Female purchases is', lower_limit, 'to', upper_limit)
```

```
The 95% confidence interval of Male purchases is 9125.92 to 9757.69
The 95% confidence interval of Female purchases is 8436.94 to 9035.77
```

In [55]:
```
lower_limit = round(np.percentile(bootstrapped_male_samples, 0.5), 2)
upper_limit = round(np.percentile(bootstrapped_male_samples, 99.5), 2)
print('The 99% confidence interval of Male purchases is', lower_limit, 'to', upper_limit)

lower_limit = round(np.percentile(bootstrapped_female_samples, 0.5), 2)
upper_limit = round(np.percentile(bootstrapped_female_samples, 99.5), 2)
print('The 99% confidence interval of Female purchases is', lower_limit, 'to', upper_limit)
```

```
The 99% confidence interval of Male purchases is 9028.94 to 9854.86
The 99% confidence interval of Female purchases is 8350.9 to 9119.63
```

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

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

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

4.Results when the same activity is performed for Married vs Unmarried

Results when the same activity is performed for Age

Answers for Gender Category

- The sampling distribution of Male and Female Purchases has very less overlapping.
- Their 95% Confidence Interval does not has any overlap.
- Even though the top categories bought by Male and female is very similar, still the purchases made by Male is greater than female.

- From these we can conclude that Males spend more than Female.
- From this Walmart can conclude that if the Customer is Male, then there will be higher chances that he would make bigger purchases.
- If offers like buy 2 get 1 free, and any such offers which tempts the customer to by more is offered to males, then chances of success will be more.

Performing the same for Marital_Status

In [8]:
```
dt.Marital_Status.unique()
```

Out[8]:
```
array([0, 1], dtype=object)
```

In [16]:
```
d = dt.groupby(['Marital_Status'])['Purchase'].mean().sort_values(ascending = True)
print('Mean of Marital Status 0 =', round(d[0], 2))
print('Mean of Marital Status 1 =', round(d[1], 2))
```

```
Mean of Marital Status 0 = 9265.91
Mean of Marital Status 1 = 9261.17
```

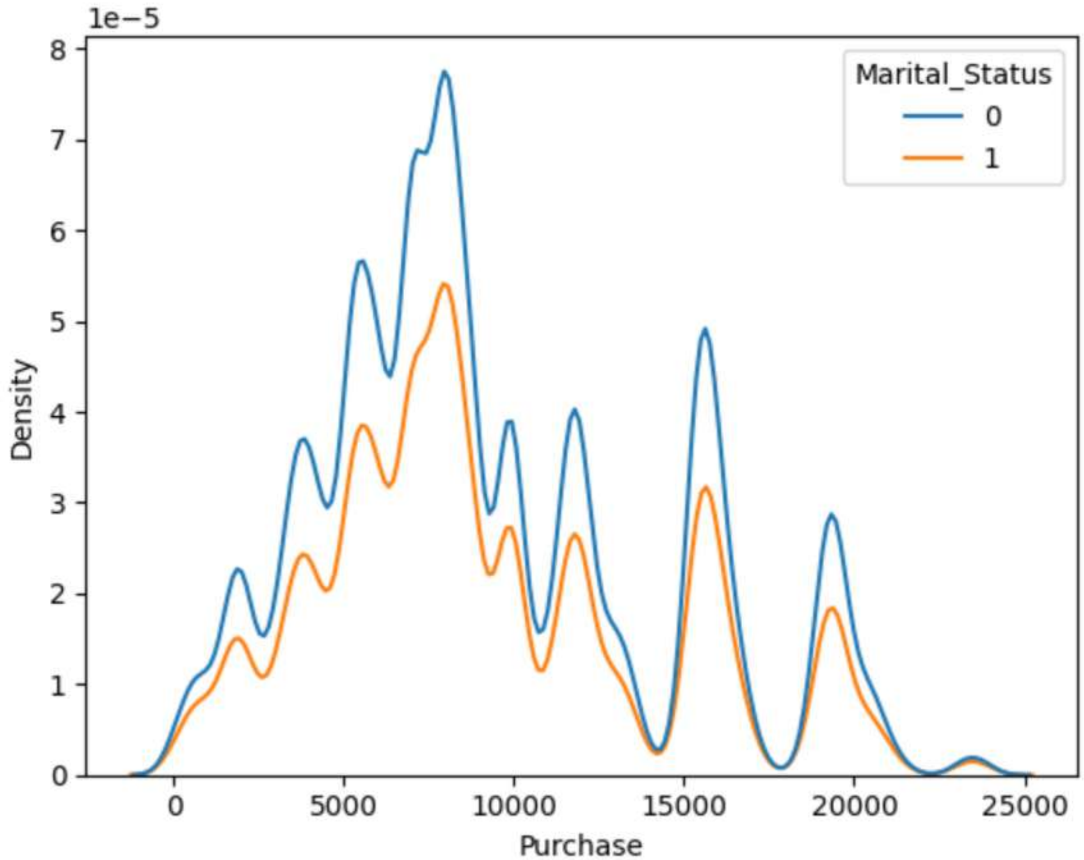
In [17]:
```
d
```

Out[17]:
```
Marital_Status
1    9261.174574
0    9265.907619
Name: Purchase, dtype: float64
```

In [79]:
```
d = dt.groupby(['Marital_Status'])['Purchase'].median().sort_values(ascending = False)
print('Median of Marital Status 0 =', round(d[0], 2))
print('Median of Marital Status 1 =', round(d[1], 2))
```

```
Median of Marital Status 0 = 8044
Median of Marital Status 1 = 8051
```

In [80]:
```
sns.kdeplot(x = 'Purchase', data = dt, hue = 'Marital_Status')
plt.show()
```

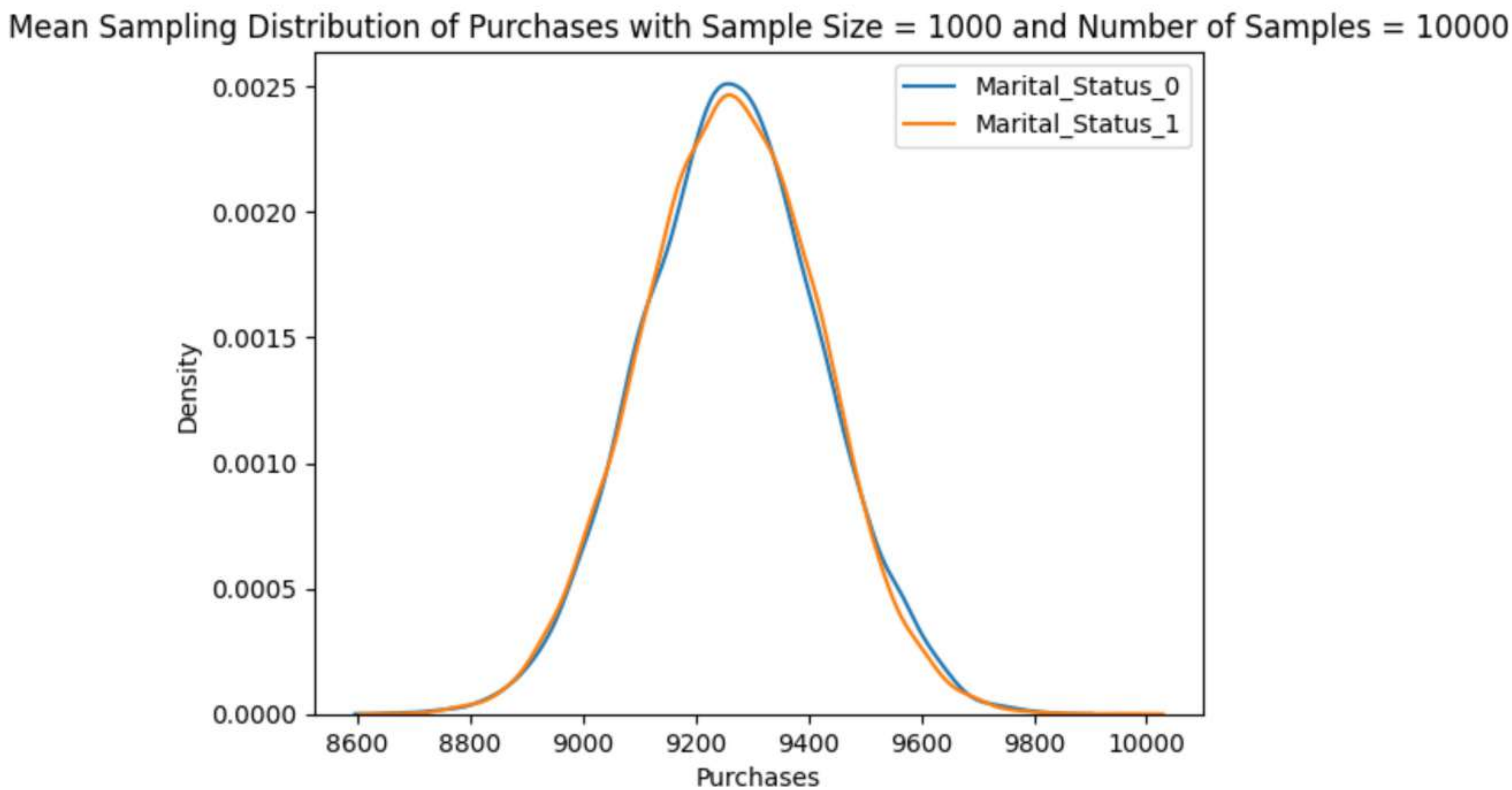


In [117…

```
_0_top_product_category = pd.crosstab(dt['Product_Category'], dt['Marital_Status']).sort_values([0], ascending = False
_1_top_product_category = pd.crosstab(dt['Product_Category'], dt['Marital_Status']).sort_values([1], ascending = False

print("Marital Status 0 Top Product Categories = ", _0_top_product_category)

print("Marital Status 1 Top Product Categories = ", _1_top_product_category)
```

```
Marital Status 0 Top Product Categories =  [ 5  1  8 11  2]
Marital Status 1 Top Product Categories =  [ 5  1  8  2 11]
```

In [89]:
```
_0 = dt.loc[dt['Marital_Status']==0, :]['Purchase'].values
_1 = dt.loc[dt['Marital_Status']==1, :]['Purchase'].values

bootstrapped_0_samples = bootstrap_sampling(data = list(_0), sample_size = 1000, number_of_samples = 10000)
bootstrapped_1_samples = bootstrap_sampling(data = list(_1), sample_size = 1000, number_of_samples = 10000)


sns.kdeplot(bootstrapped_0_samples, label = 'Marital_Status_0')
sns.kdeplot(bootstrapped_1_samples, label = 'Marital_Status_1')
plt.title('Mean Sampling Distribution of Purchases with Sample Size = 1000 and Number of Samples = 10000')
plt.xlabel('Purchases')
plt.legend()
plt.show()
```



In [92]:
```
lower_limit = round(np.percentile(bootstrapped_0_samples, 2.5), 2)
upper_limit = round(np.percentile(bootstrapped_0_samples, 97.5), 2)
print('The 95% confidence interval of Marital Status 0 is', lower_limit, 'to', upper_limit)
```

```
The 95% confidence interval of Marital Status 0 is 8956.15 to 9579.44
```

In [93]:
```
lower_limit = round(np.percentile(bootstrapped_1_samples, 2.5), 2)
upper_limit = round(np.percentile(bootstrapped_1_samples, 97.5), 2)
print('The 95% confidence interval of Marital Status 1 is', lower_limit, 'to', upper_limit)
```

```
The 95% confidence interval of Marital Status 1 is 8950.63 to 9569.95
```

Answers -- Marital Status

- The sampling distribution of Marital_Status0 and Marital_Status1 Purchases has almost complete overlapping.
- Their 95% Confidence Interval also has so much overlapping.
- The top categories bought by Male and female is also same.

- From these we can conclude that the difference in Purchases made by Marital_Status0 and Marital_Status1 is not significant.

- Hence the Marital Status does not hold much difference in deciding the Purchases.

Performing the same for Age

In [23]:
```
d = dt.groupby(['Age']).aggregate({'Purchase':['mean', 'median', 'min', 'max']})
d.columns = ['_'.join(i) for i in d]
d
```

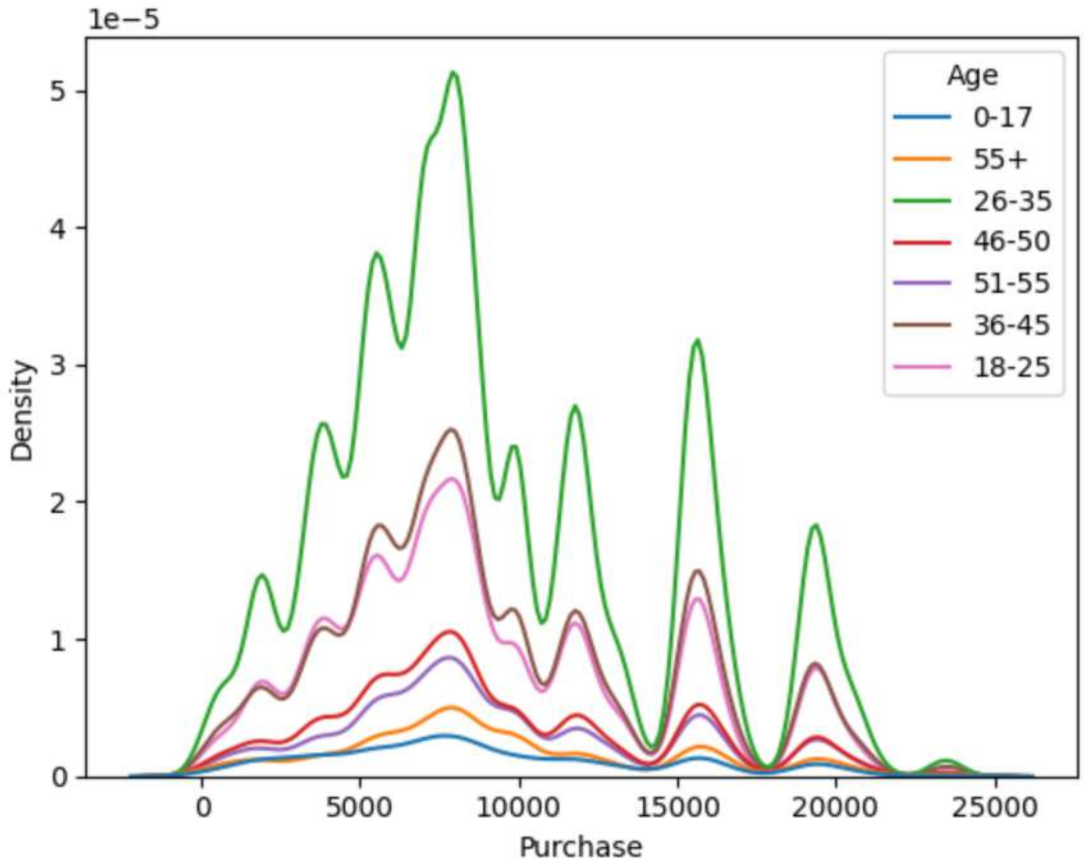
Out[23]:

Age	Purchase_mean	Purchase_median	Purchase_min	Purchase_max
0-17	8933.464640	7986.0	12	23955
18-25	9169.663606	8027.0	12	23958
26-35	9252.690633	8030.0	12	23961
36-45	9331.350695	8061.0	12	23960

	Purchase_mean	Purchase_median	Purchase_min	Purchase_max
Age				
46-50	9208.625697	8036.0	12	23960

In [104…

```
sns.kdeplot(x = 'Purchase', data = dt, hue = 'Age')
plt.show()
```



In [114…

```
pd.crosstab(dt['Product_Category'], dt['Age']).head(5)
```

Out[114…

Age	0-17	18-25	26-35	36-45	46-50	51-55	55+

Product_Category							
1	3585	26962	58249	27648	10474	9049	4411
2	805	4428	8928	4912	2105	1781	905
3	1200	4710	7662	3854	1376	924	487
4	758	2463	4192	2354	990	678	318
5	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

In [123…

```
dt['Age'].unique()
```

Out[123…

```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

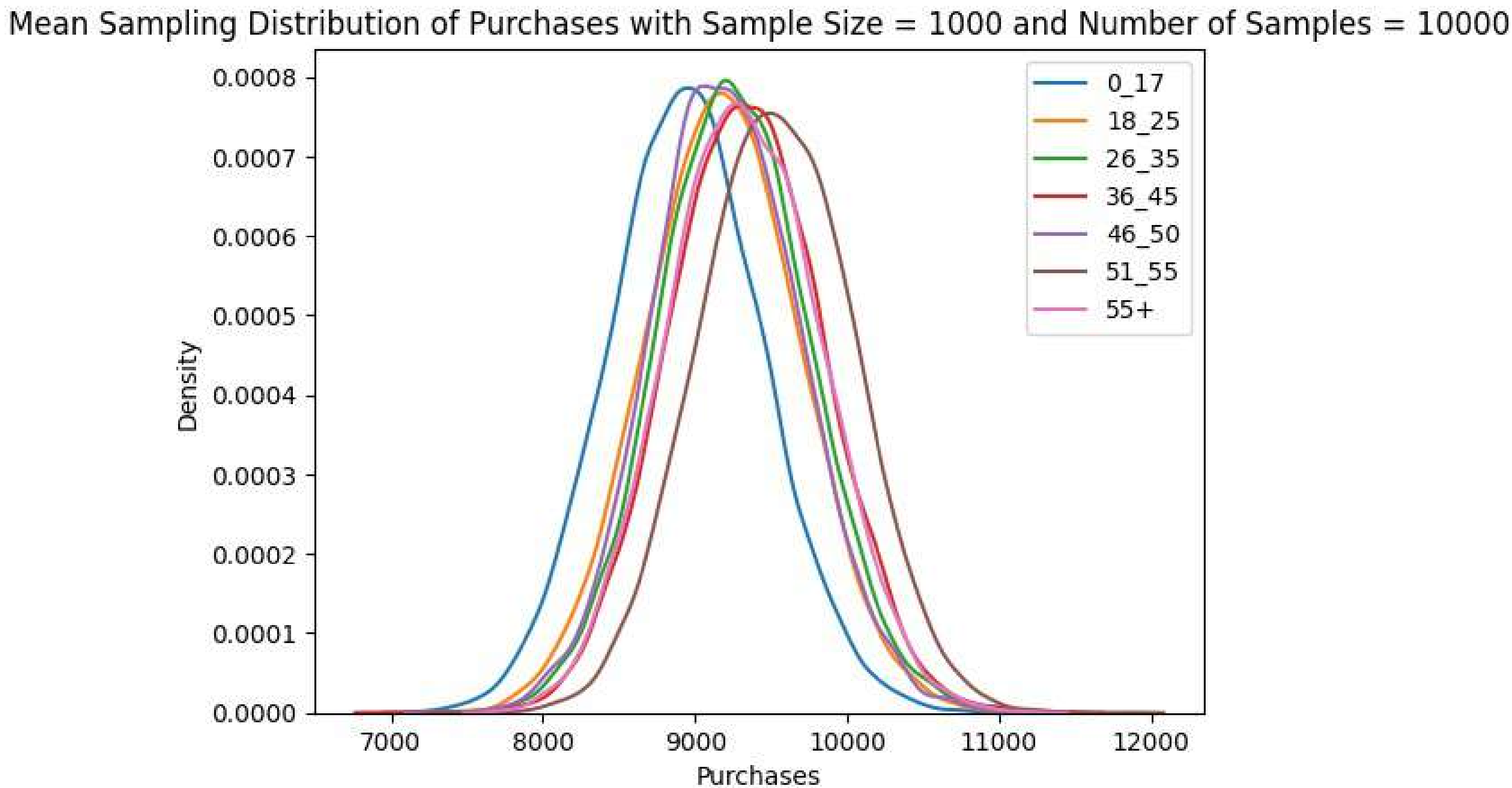
In [132…

```
_0_17_top_product_category = pd.crosstab(dt['Product_Category'], dt['Age']).sort_values(['0-17'], ascending = False).i
_18_25_top_product_category = pd.crosstab(dt['Product_Category'], dt['Age']).sort_values(['18-25'], ascending = False)
_26_35_top_product_category = pd.crosstab(dt['Product_Category'], dt['Age']).sort_values(['26-35'], ascending = False)
_36_45_top_product_category = pd.crosstab(dt['Product_Category'], dt['Age']).sort_values(['36-45'], ascending = False)
_46_50_top_product_category = pd.crosstab(dt['Product_Category'], dt['Age']).sort_values(['46-50'], ascending = False)
_51_55_top_product_category = pd.crosstab(dt['Product_Category'], dt['Age']).sort_values(['51-55'], ascending = False)
_55Plus_top_product_category = pd.crosstab(dt['Product_Category'], dt['Age']).sort_values(['55+'], ascending = False).

print("Top Product Categories of 0 - 17 = ", _0_17_top_product_category)
print("Top Product Categories of 18 - 25 = ", _18_25_top_product_category)
print("Top Product Categories of 26 - 35 = ", _26_35_top_product_category)
print("Top Product Categories of 36 - 45 = ", _36_45_top_product_category)
print("Top Product Categories of 46 - 50 = ", _46_50_top_product_category)
print("Top Product Categories of 51 - 55 = ", _51_55_top_product_category)
print("Top Product Categories of 55+ = ", _55Plus_top_product_category)
```

```
Top Product Categories of 0 - 17 =  [5 1 8 3 2]
Top Product Categories of 18 - 25 =  [ 5  1  8  3 11]
Top Product Categories of 26 - 35 =  [ 5  1  8 11  2]
Top Product Categories of 36 - 45 =  [ 5  1  8 11  2]
Top Product Categories of 46 - 50 =  [ 5  8  1  2 11]
Top Product Categories of 51 - 55 =  [ 5  8  1  2 11]
Top Product Categories of 55+ =  [8 5 1 2 6]
```

In [135…

```
_0_17_purchases = dt.loc[dt['Age']=='0-17', :]['Purchase'].values
_18_25_purchases = dt.loc[dt['Age']=='18-25', :]['Purchase'].values
_26_35_purchases = dt.loc[dt['Age']=='26-35', :]['Purchase'].values
_36_45_purchases = dt.loc[dt['Age']=='36-45', :]['Purchase'].values
_46_50_purchases = dt.loc[dt['Age']=='46-50', :]['Purchase'].values
_51_55_purchases = dt.loc[dt['Age']=='51-55', :]['Purchase'].values
_55Plus_purchases = dt.loc[dt['Age']=='55+', :]['Purchase'].values


bootstrapped_0_17_samples = bootstrap_sampling(data = list(_0_17_purchases), sample_size = 100, number_of_samples = 10
bootstrapped_18_25_samples = bootstrap_sampling(data = list(_18_25_purchases), sample_size = 100, number_of_samples =
bootstrapped_26_35_samples = bootstrap_sampling(data = list(_26_35_purchases), sample_size = 100, number_of_samples =
bootstrapped_36_45_samples = bootstrap_sampling(data = list(_36_45_purchases), sample_size = 100, number_of_samples =
bootstrapped_46_50_samples = bootstrap_sampling(data = list(_46_50_purchases), sample_size = 100, number_of_samples =
bootstrapped_51_55_samples = bootstrap_sampling(data = list(_51_55_purchases), sample_size = 100, number_of_samples =
bootstrapped_55Plus_samples = bootstrap_sampling(data = list(_55Plus_purchases), sample_size = 100, number_of_samples


sns.kdeplot(bootstrapped_0_17_samples, label = '0_17')
sns.kdeplot(bootstrapped_18_25_samples, label = '18_25')
sns.kdeplot(bootstrapped_26_35_samples, label = '26_35')
sns.kdeplot(bootstrapped_36_45_samples, label = '36_45')
sns.kdeplot(bootstrapped_46_50_samples, label = '46_50')
sns.kdeplot(bootstrapped_51_55_samples, label = '51_55')
sns.kdeplot(bootstrapped_55Plus_samples, label = '55+')

plt.title('Mean Sampling Distribution of Purchases with Sample Size = 1000 and Number of Samples = 10000')
plt.xlabel('Purchases')
plt.legend()
plt.show()
```


Mean Sampling Distribution of Purchases with Sample Size = 1000 and Number of Samples = 10000
Density
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
7000
8000
9000
10000
11000
12000
Purchases
0_17
18_25
26_35
36_45
46_50
51_55
55+

In [136…

```
sns.kdeplot(bootstrapped_0_17_samples, label = '0_17')
sns.kdeplot(bootstrapped_18_25_samples, label = '18_25')
sns.kdeplot(bootstrapped_26_35_samples, label = '26_35')
sns.kdeplot(bootstrapped_36_45_samples, label = '36_45')
sns.kdeplot(bootstrapped_46_50_samples, label = '46_50')
sns.kdeplot(bootstrapped_51_55_samples, label = '51_55')
sns.kdeplot(bootstrapped_55Plus_samples, label = '55+')


lower_limit = round(np.percentile(bootstrapped_0_17_samples, 2.5), 2)
upper_limit = round(np.percentile(bootstrapped_0_17_samples, 97.5), 2)
print('The 95% confidence interval of 0 - 17 Age group purchases is', lower_limit, 'to', upper_limit)

lower_limit = round(np.percentile(bootstrapped_18_25_samples, 2.5), 2)
upper_limit = round(np.percentile(bootstrapped_18_25_samples, 97.5), 2)
print('The 95% confidence interval of 18 - 25 Age group purchases is', lower_limit, 'to', upper_limit)

lower_limit = round(np.percentile(bootstrapped_26_35_samples, 2.5), 2)
upper_limit = round(np.percentile(bootstrapped_26_35_samples, 97.5), 2)
print('The 95% confidence interval of 26 - 35 Age group purchases is', lower_limit, 'to', upper_limit)

lower_limit = round(np.percentile(bootstrapped_36_45_samples, 2.5), 2)
upper_limit = round(np.percentile(bootstrapped_36_45_samples, 97.5), 2)
print('The 95% confidence interval of 36 - 45 Age group purchases is', lower_limit, 'to', upper_limit)

lower_limit = round(np.percentile(bootstrapped_46_50_samples, 2.5), 2)
upper_limit = round(np.percentile(bootstrapped_46_50_samples, 97.5), 2)
print('The 95% confidence interval of 46 - 50 Age group purchases is', lower_limit, 'to', upper_limit)

lower_limit = round(np.percentile(bootstrapped_51_55_samples, 2.5), 2)
upper_limit = round(np.percentile(bootstrapped_51_55_samples, 97.5), 2)
print('The 95% confidence interval of 51 - 55 Age group purchases is', lower_limit, 'to', upper_limit)

lower_limit = round(np.percentile(bootstrapped_55Plus_samples, 2.5), 2)
upper_limit = round(np.percentile(bootstrapped_55Plus_samples, 97.5), 2)
print('The 95% confidence interval of 55+ Age group purchases is', lower_limit, 'to', upper_limit)
```

```
The 95% confidence interval of 0 - 17 Age group purchases is 7966.5 to 9960.74
The 95% confidence interval of 18 - 25 Age group purchases is 8184.72 to 10189.26
The 95% confidence interval of 26 - 35 Age group purchases is 8308.94 to 10251.15
The 95% confidence interval of 36 - 45 Age group purchases is 8381.22 to 10329.65
The 95% confidence interval of 46 - 50 Age group purchases is 8268.0 to 10205.33
The 95% confidence interval of 51 - 55 Age group purchases is 8554.67 to 10519.47
```

```
The 95% confidence interval of 55+ Age group purchases is 8366.9 to 10313.17
```

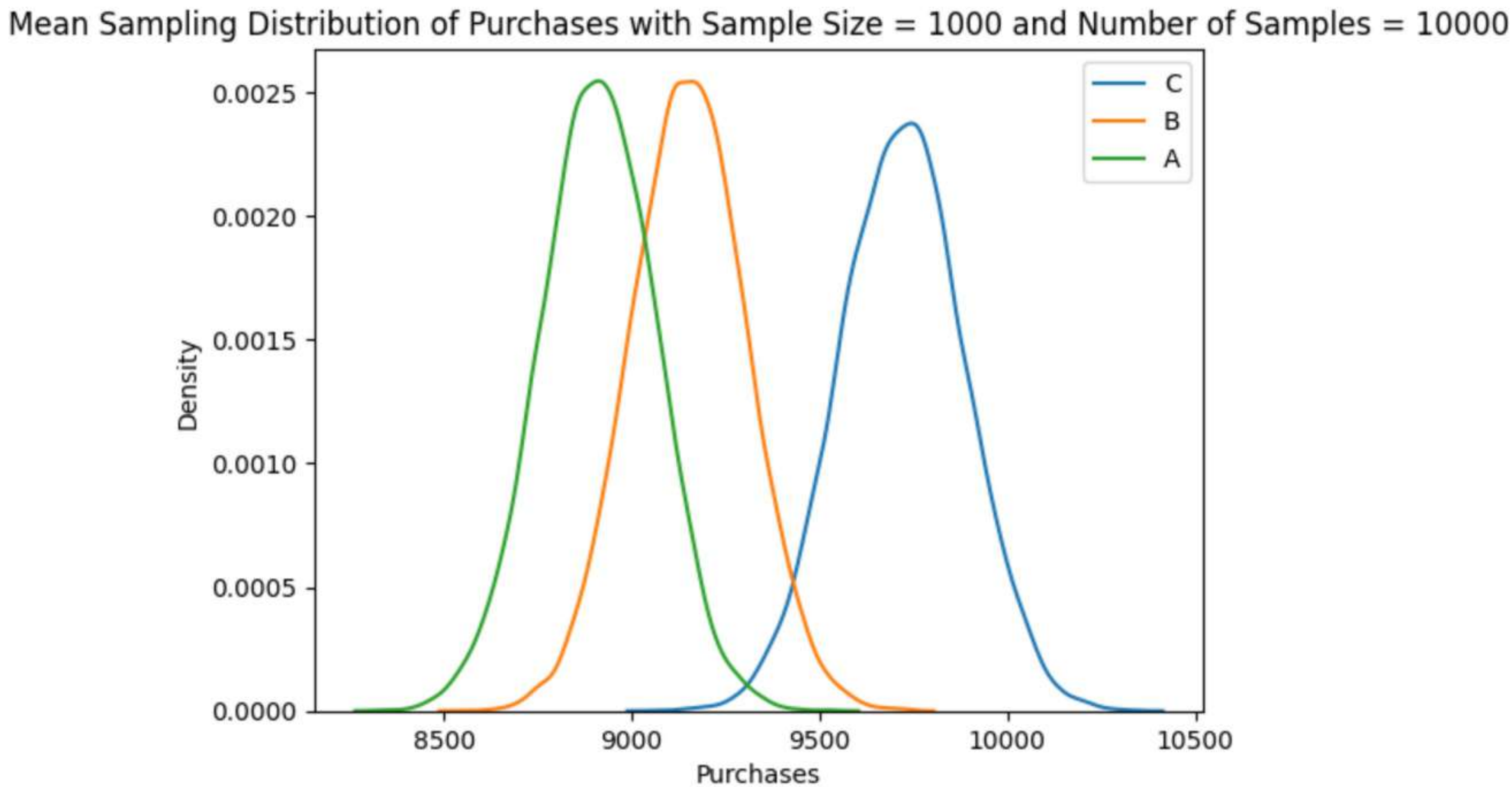
Answers -- Marital Status

- Among all the Age groups, the Common Product categories are - 5, 8 and 1.
- Product Category 6 is specific to 55+ Age group.
- The Purchase distributions has very much considerable amount of overlapping.
- Their 95% confidence interval also has much overlapping
- From these we can conclude that the difference in Purchases made by different Age groups is not significant.
- Hence the Age group does not hold much difference in deciding the Purchases.

City Category

In [25]:

```
A = dt.loc[dt['City_Category']=='A', :]['Purchase'].values
B = dt.loc[dt['City_Category']=='B', :]['Purchase'].values
C = dt.loc[dt['City_Category']=='C', :]['Purchase'].values

bootstrapped_A_samples = bootstrap_sampling(data = list(A), sample_size = 1000, number_of_samples = 10000)
bootstrapped_B_samples = bootstrap_sampling(data = list(B), sample_size = 1000, number_of_samples = 10000)
bootstrapped_C_samples = bootstrap_sampling(data = list(C), sample_size = 1000, number_of_samples = 10000)

sns.kdeplot(bootstrapped_C_samples, label = 'C')
sns.kdeplot(bootstrapped_B_samples, label = 'B')
sns.kdeplot(bootstrapped_A_samples, label = 'A')

plt.title('Mean Sampling Distribution of Purchases with Sample Size = 1000 and Number of Samples = 10000')
plt.xlabel('Purchases')
plt.legend()
plt.show()
```



In [26]:

```
lower_limit = round(np.percentile(bootstrapped_A_samples, 2.5), 2)
upper_limit = round(np.percentile(bootstrapped_A_samples, 97.5), 2)
print('The 95% confidence interval of City Category A purchases is', lower_limit, 'to', upper_limit)

lower_limit = round(np.percentile(bootstrapped_B_samples, 2.5), 2)
upper_limit = round(np.percentile(bootstrapped_B_samples, 97.5), 2)
print('The 95% confidence interval of City Category B purchases is', lower_limit, 'to', upper_limit)


lower_limit = round(np.percentile(bootstrapped_C_samples, 2.5), 2)
upper_limit = round(np.percentile(bootstrapped_C_samples, 97.5), 2)
print('The 95% confidence interval of City Category C purchases is', lower_limit, 'to', upper_limit)
```

```
The 95% confidence interval of City Category A purchases is 8608.01 to 9207.3
```

```
The 95% confidence interval of City Category B purchases is 8851.96 to 9454.17
```

Since the overlap of confidence interval of Category C Purchase is less with Category A and Category B, we can say that the chances of People from City C making bigger purchases is more comparitvely.

Final Insights

- 39.99 % of the Customers come from 26-35 age group, followed by 36-45 which has 19.99% Customers and 18-25 which has 18.11% Customers
- 42.02 % of the Customers come from Category B City, followed by Category C and A.
- People who stay in the current year for 1 year, constitute 35.23 % of all the customers.
- People with Marital status 0 constitute 59.03 % of the population, and rest by Status 1.
- Products belonging to category 5, 1 and 8 are most Purchased products, constituting of 27.43 %, 25.51 % and 20.71 % respectively
- The Purchase of City Category C is more when comapred to Category A and B.

- For Bivariate analysis, Gender has an impact on Purchases made, but Marital Status and Age group does not has significant imact.
- For City Category C, the purchases are higher than Category A and Category B.

- Even though the count of transactions made from City B is more than A and B, but still the amount of transaction made by City C is high

Final Recommendations -

The most popular categories are 5, 8 and 1. Hence its recomeneded to have good variety, quality and always in stock for products from these categories. Sales might see a good jump on giving discounts for products belonging to these categories.

Since most of the custmoers are from 26-35 age group, its recommended to make sure the store has enough products addressing the needs of this age group.

Since Males spend more when compared to Females, there will be more chances of increase of sales on giving offers to Males.

If offers like buy 2 get 1 free, and any such offers which tempts the customer to by more is offered to males, then chances of success will be more.

People from City C tend to spend more per transaction. If there is a expensive product, there will be more chances that someone from City C would buy it. The most number of Transactions however is made by City B. So the overall profit will increase if the quality of

In [ ]:

```

```

In [ ]:

```

```