Walmart

Save money. Live better.

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm
from scipy import stats
import math
from scipy.stats import binom,geom
from scipy.stats import bernoulli
from scipy.stats import ttest_1samp,ttest_ind
from scipy.stats import chi2_contingency,chisquare,f_oneway,pearsonr,spearmanr
from scipy.stats import poisson
```

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

```python
[28] df.head()
```

	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

```python
#size of data set
df.shape
```

```
(550068, 10)
```

```
#we dont have any nan values
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
```

```
#Numeric variable
df.describe()
```

	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

+ Code + Text

[15]
```
#catagorecal variable
df.describe(include='object')
```

	Product_ID	Gender	Age	City_Category	Stay_In_Current_City_Years
count	550068	550068	550068	550068	550068
unique	3631	2	7	3	5
top	P00265242	M	26-35	B	1
freq	1880	414259	219587	231173	193821

User ID:

[16]
```
df['User_ID']
```

```
0         1000001
1         1000001
2         1000001
3         1000001
4         1000002
           ...
550063    1006033
550064    1006035
550065    1006036
550066    1006038
550067    1006039
Name: User_ID, Length: 550068, dtype: int64
```

```
[17] df['User_ID'].value_counts()
```

```
1001680    1026
1004277     979
1001941     898
1001181     862
1000889     823
           ...
1002690       7
1002111       7
1005810       7
1004991       7
1000708       6
Name: User_ID, Length: 5891, dtype: int64
```

```
#this user id has highest num of purchases
df[df['User_ID']==1001680]
```

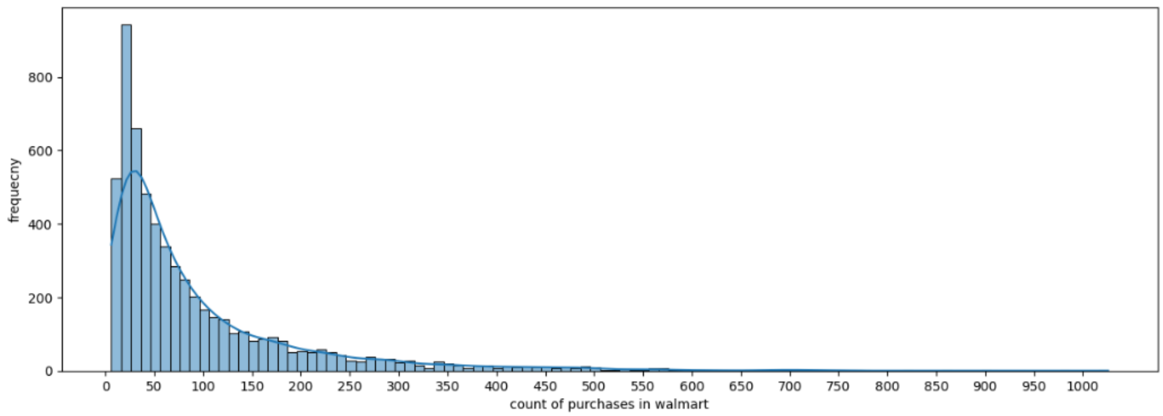
	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purc
11055	1001680	P00036742	M	26-35	20	A	3	1	1	1
11056	1001680	P00130642	M	26-35	20	A	3	1	11	
11057	1001680	P00105442	M	26-35	20	A	3	1	11	
11058	1001680	P00245642	M	26-35	20	A	3	1	5	
11059	1001680	P00123342	M	26-35	20	A	3	1	11	
...	...	...	...	...	...	...	...	...	...	
517447	1001680	P00238742	M	26-35	20	A	3	1	13	
517448	1001680	P00146742	M	26-35	20	A	3	1	1	
517449	1001680	P00285042	M	26-35	20	A	3	1	16	
517450	1001680	P00047742	M	26-35	20	A	3	1	16	1
547057	1001680	P00372445	M	26-35	20	A	3	1	20	

1026 rows × 10 columns

```
[19] #unique no of customer
     df['User_ID'].nunique()
```

```
5891
```

```
#x-axis is the user-id counts
#y-axis is the freq of the counts
fig, ax = plt.subplots(figsize=(15, 5))
sns.histplot(df['User_ID'].value_counts(),kde=True,ax=ax)

plt.xlabel('count of purchases in walmart')
plt.ylabel('frequecny')
plt.xticks([x for x in range(0,1026,50)])
plt.show()
```



x-axis is count of purchases in walmart.

y-axis is the frequency.

most of the users have purchase history of around 0-50 purchases in walmert.

```
[23] df['Product_ID']
```

```
0         P00069042
1         P00248942
2         P00087842
3         P00085442
4         P00285442
             ...
550063    P00372445
550064    P00375436
550065    P00375436
550066    P00375436
550067    P00371644
Name: Product_ID, Length: 550068, dtype: object
```

```
#there are 3631 unique products in walmart
df['Product_ID'].nunique()
```

```
3631
```

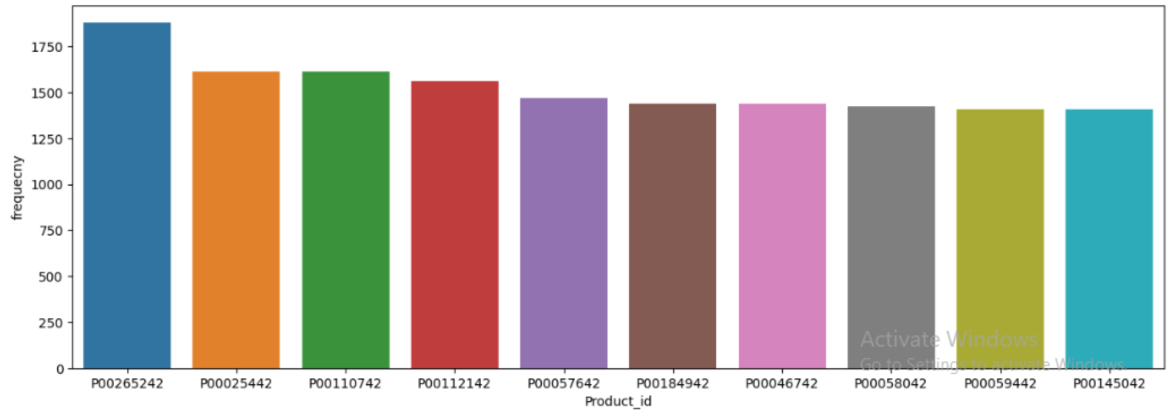
```
[25] df['Product_ID'].value_counts()
```

```
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
             ...
P00314842       1
P00298842       1
P00231642       1
P00204442       1
P00066342       1
Name: Product_ID, Length: 3631, dtype: int64
```

```
[26] top_10=pd.DataFrame(df['Product_ID'].value_counts()[:10]).reset_index()
```

```python
fig, ax = plt.subplots(figsize=(15, 5))
sns.barplot(x=top_10['index'],y=top_10['Product_ID'])
plt.xlabel('Product_id')
plt.ylabel('frequecny')
```

```
Text(0, 0.5, 'frequecny')
```



Top 10 products

customers perfer buying product_id P00265242 more.

Gender:

```python
df['Gender']
```

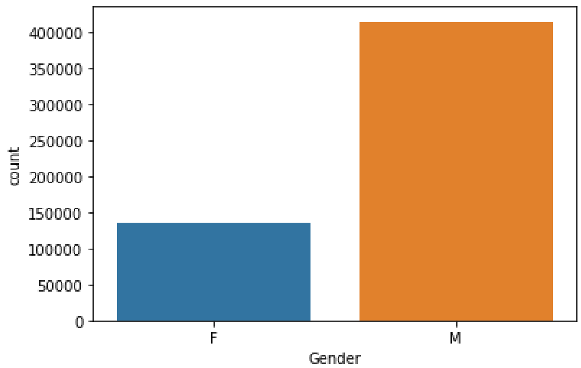
```
0         F
1         F
2         F
3         F
4         M
         ..
550063    M
550064    F
550065    F
550066    F
550067    F
Name: Gender, Length: 550068, dtype: object
```

```python
[30] df['Gender'].value_counts()
```

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

```python
sns.countplot(df['Gender'])
```

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



Males tends to buy more.

Age:

```
[32] df['Age']
```

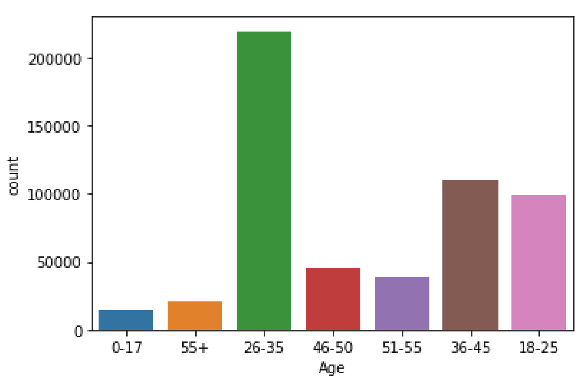
```
0         0-17
1         0-17
2         0-17
3         0-17
4          55+
          ...
550063   51-55
550064   26-35
550065   26-35
550066     55+
550067   46-50
Name: Age, Length: 550068, dtype: object
```

```
[33] df['Age'].value_counts()
```

```
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
```

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

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



Age between 26-35 years tends to buy more.

Occupation:

[34]
```
df['Occupation']
```

```
0         10
1         10
2         10
3         10
4         16
          ..
550063    13
550064     1
550065    15
550066     1
550067     0
Name: Occupation, Length: 550068, dtype: int64
```

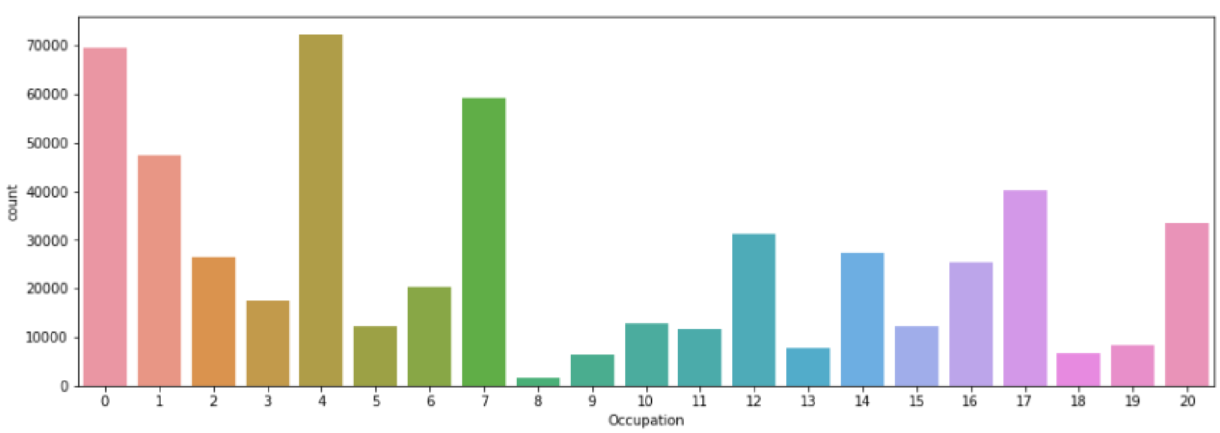
```
df['Occupation'].value_counts()
```

```
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
```


[36]
```
# there about 21 occupation catagoery
df['Occupation'].nunique()
```

```
21
```

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



customers having occupational catagorery as 4,0,7 tends to buy more.

City_Category:

```
[39] df['City_Category']
```

```
0         A
1         A
2         A
3         A
4         C
         ..
550063    B
550064    C
550065    B
550066    C
550067    B
Name: City_Category, Length: 550068, dtype: object
```

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

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

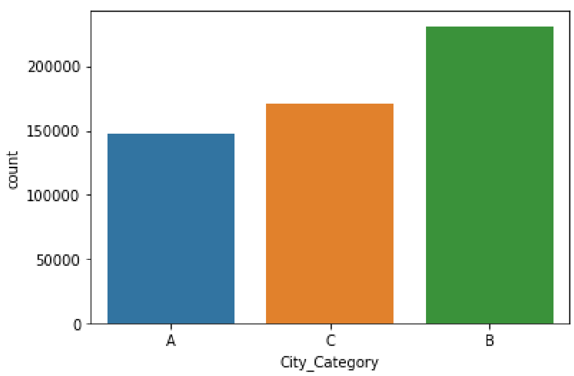


```
[41] df['City_Category'].nunique()
```

```
3
```

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

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



Customers from city B tends to buy more.

```
[43] df['Stay_In_Current_City_Years']
```

```
0          2
1          2
2          2
3          2
4         4+
          ..
550063     1
550064     3
550065    4+
550066     2
550067    4+
Name: Stay_In_Current_City_Years, Length: 550068, dtype: object
```

```python
df['Stay_In_Current_City_Years'].value_counts()
```

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

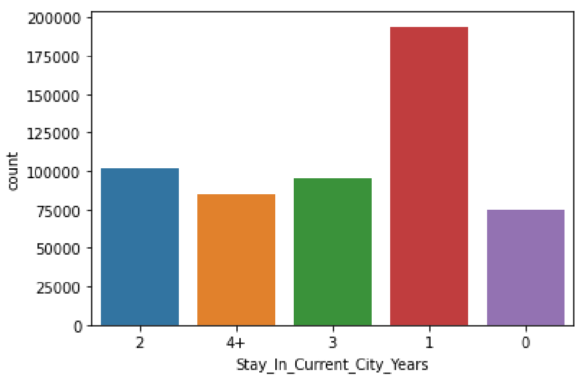


```python
[45] df['Stay_In_Current_City_Years'].nunique()
```

```
5
```

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

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



Customers who stayed 1 year in current city tends to buy more.

Marital_Status:

```python
df['Marital_Status']
```

```
0         0
1         0
2         0
3         0
4         0
         ..
550063    1
550064    0
550065    1
550066    0
550067    1
Name: Marital_Status, Length: 550068, dtype: int64
```

```python
df['Marital_Status'].value_counts()
```

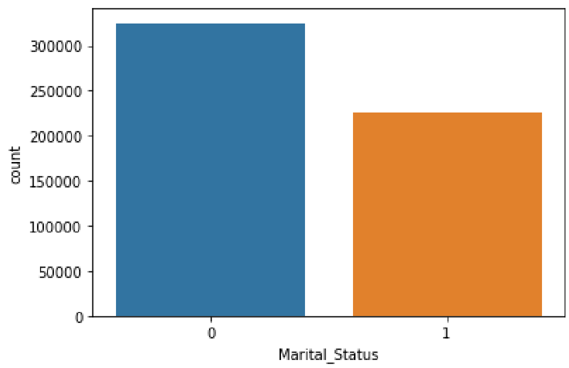
```
0    324731
1    225337
Name: Marital_Status, dtype: int64
```

```
[49] df['Marital_Status'].unique()
```

```
array([0, 1])
```

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

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



customers having marital status as 0 tends to buy more.

Product_Category:

```
[51] df['Product_Category']
```

```
0          3
1          1
2         12
3         12
4          8
          ..
550063    20
550064    20
550065    20
550066    20
550067    20
Name: Product_Category, Length: 550068, dtype: int64
```

```
df['Product_Category'].value_counts()
```

```
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
```

```
[53] df['Product_Category'].unique()
```

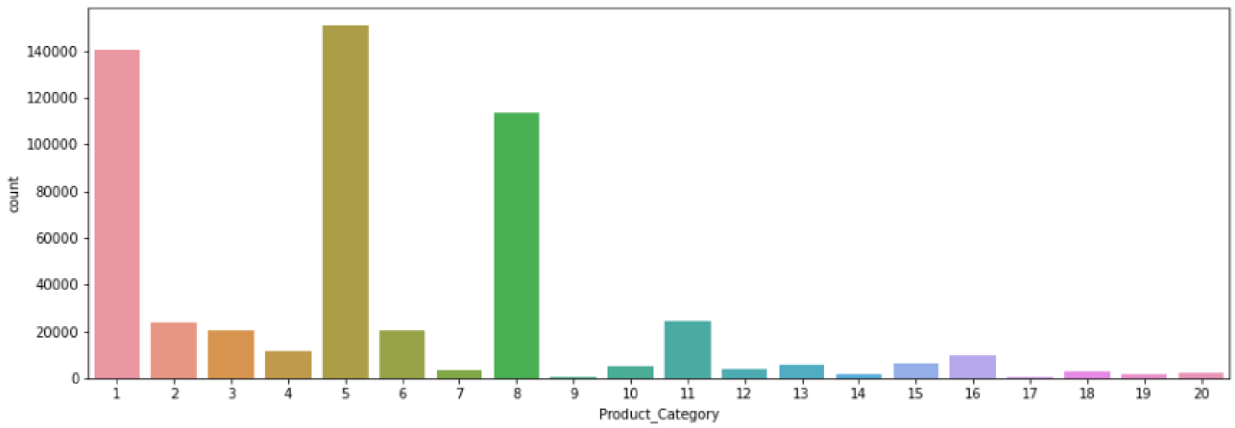
```
array([ 3,  1, 12,  8,  5,  4,  2,  6, 14, 11, 13, 15,  7, 16, 18, 10, 17,
        9, 20, 19])
```

```
[54] df['Product_Category'].nunique()
```

```
20
```

```
fig, ax = plt.subplots(figsize=(15, 5))
sns.countplot(df['Product_Category'])
```

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

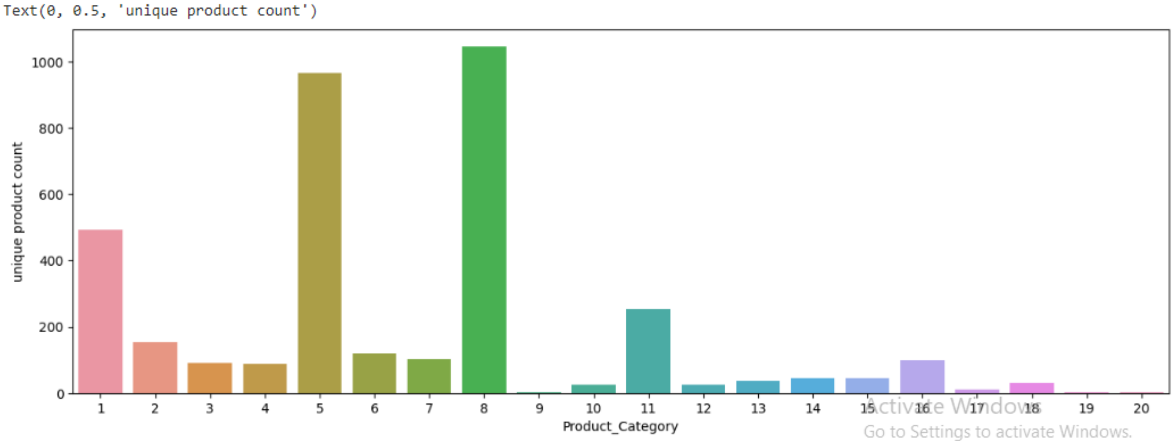


Customers tends to buy product_category 5,1,8 more.

```
[55] # counts of unique products in each product_catagoery
     df_cat_prod=pd.DataFrame(df.groupby(df['Product_Category'])['Product_ID'].nunique()).reset_index()
```

```
fig, ax = plt.subplots(figsize=(15, 5))
sns.barplot(x=df_cat_prod['Product_Category'],y=df_cat_prod['Product_ID'])
plt.ylabel('unique product count')
```

```
Text(0, 0.5, 'unique product count')
```



Product category 8 have highest number of unique products in walmart.

Purchase:

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

```
0          8370
1         15200
2          1422
3          1057
4          7969
          ...
550063     368
550064     371
550065     137
550066     365
550067     490
Name: Purchase, Length: 550068, dtype: int64
```

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

```
9263.968712959126
```



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

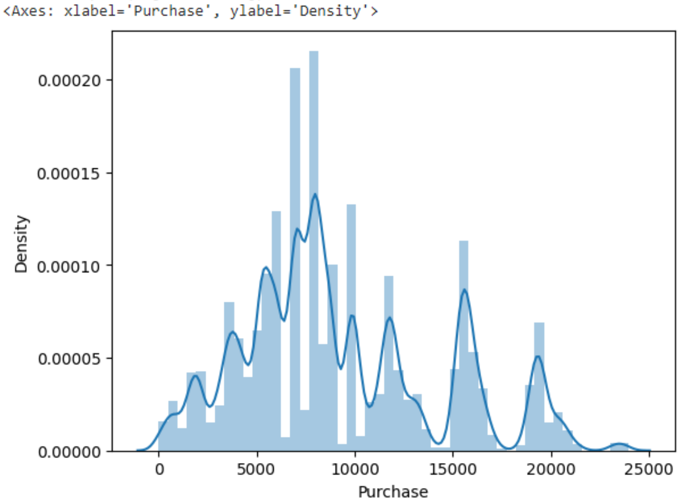
```
12
```

```
[61] df['Purchase'].max()
```

```
23961
```

```
#drist of purchase price
sns.distplot(df['Purchase'])
```

```
sns.distplot(df['Purchase'])
<Axes: xlabel='Purchase', ylabel='Density'>
```



Bivariate:

```
df
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purc
0	1000001	P00069042	F	0-17	10	A	2	0	3	
1	1000001	P00248942	F	0-17	10	A	2	0	1	1
2	1000001	P00087842	F	0-17	10	A	2	0	12	
3	1000001	P00085442	F	0-17	10	A	2	0	12	
4	1000002	P00285442	M	55+	16	C	4+	0	8	
...	...	...	...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	1	1	20	
550064	1006035	P00375436	F	26-35	1	C	3	0	20	
550065	1006036	P00375436	F	26-35	15	B	4+	1	20	

550065	1006036	P00375436	F	26-35	15	B	4+	1	20
550066	1006038	P00375436	F	55+	1	C	2	0	20
550067	1006039	P00371644	F	46-50	0	B	4+	1	20

550068 rows × 10 columns

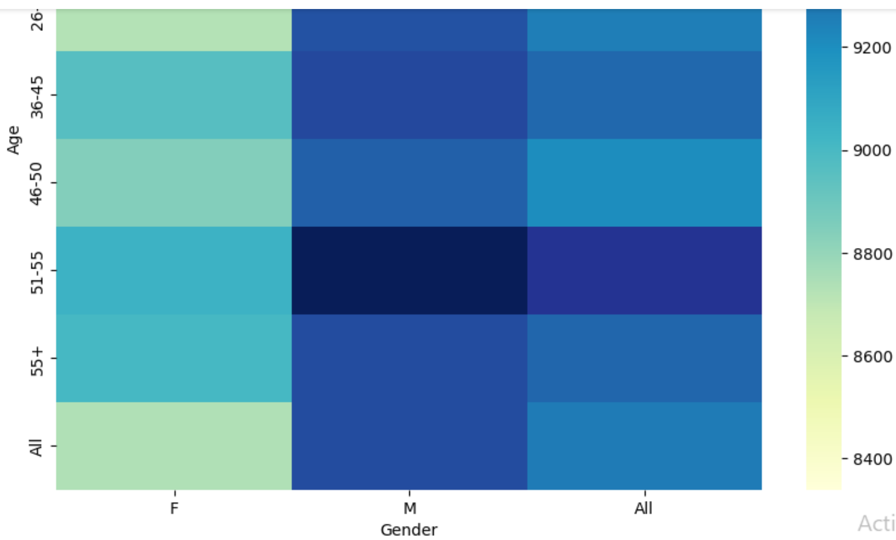
Since we have only one countinous column we cannot buid corelation matrix.

Cross tabs (with respective to mean of purchase)

```
def cross_tabs(x,y):
    df_c_age_gender=pd.crosstab(df[x],df[y],values=df['Purchase'],aggfunc=np.mean,margins=True)
    print(df_c_age_gender)
    fig, ax = plt.subplots(figsize=(10, 8))
    sns.heatmap(df_c_age_gender,cmap="YlGnBu")
```

```
cross_tabs('Age','Gender')
```

Gender	F	M	All
Age			
0-17	8338.771985	9235.173670	8933.464640
18-25	8343.180201	9440.942971	9169.663606
26-35	8728.251754	9410.337578	9252.690633
36-45	8959.844056	9453.193643	9331.350695
46-50	8842.098947	9357.471509	9208.625697
51-55	9042.449666	9705.094802	9534.808031
55+	9007.036199	9438.195603	9336.280459
All	8734.565765	9437.526040	9263.968713



- we can observe here that customer who's age is between 51-55 and gender =male tends to have more purchase mean.
- Purchase mean(Male) > Purchase mean(Female).
- Purchase mean(age=51-55) > then remaining category.

```
cross_tabs('Age','Occupation')
```

```
Occupation            0            1            2            3            4  \
Age
0-17        9305.183224  8751.248062  8416.319444          NaN  9431.964602
18-25       9167.149863  8524.334031  8443.224565  9846.722043  9153.604776
26-35       9113.090048  8961.626468  9161.169375  8906.152102  9337.273215
36-45       9356.152169  8856.423113  8719.387035  9280.210616  9414.961649
46-50       8391.753788  8991.659331  8957.813559  9483.148843  7781.914729
51-55       8954.350282  9366.317234  9276.491071  9050.461609  9335.269076
55+         9461.835075  9074.359350  9479.487685  9442.193350          NaN
All         9124.428588  8953.193270  8952.481683  9178.593088  9213.980251

Occupation             5            6             7             8  \
Age
0-17                 NaN          NaN  10509.791367  10485.862069
18-25       10037.702759  8396.458916   8894.561116   7737.357143
26-35        9447.950345  9241.864329   9252.979925  10512.060847
36-45        8889.509785  9532.839486   9419.520627   6688.438776
46-50        8478.474305  9406.490433   9607.381303   9704.191257
51-55       10828.307692  9086.876012   9951.579645   9023.652997
55+         15413.200000  9323.081818  10003.752771   9365.527950
All          9333.149298  9256.535691   9425.728223   9532.592497

Occupation            9  ...           12           13            14  \
Age                      ...
0-17                NaN  ...  8095.827004  8970.200000  11624.344086
18-25       8771.542039  ...  9897.944820          NaN   9123.637192
26-35       8349.282740  ...  9797.160678          NaN   9337.963707
```

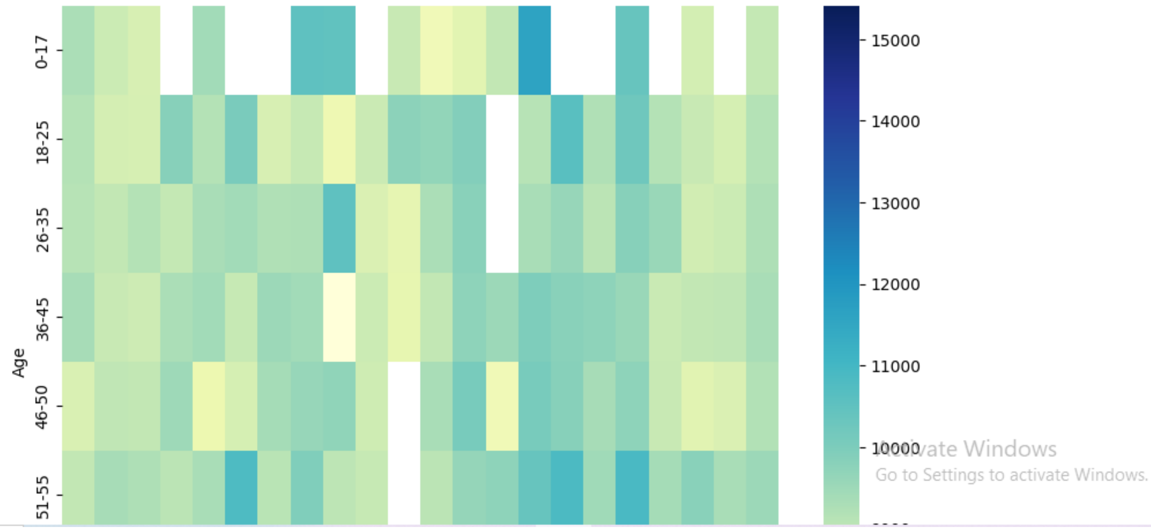
Activate Windows
Go to Settings to activate Windows.

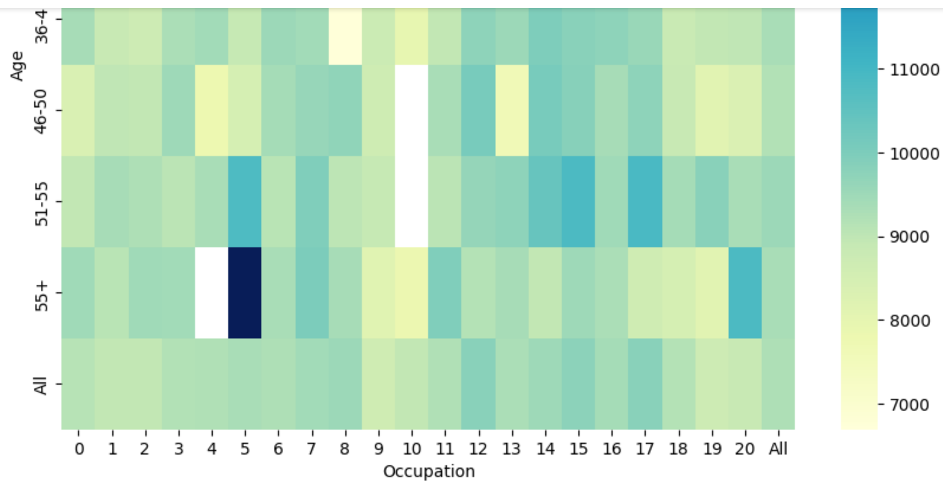
```
cross_tabs('Age','Occupation')
```

```
Occupation            9  ...           12           13            14  \
Age                      ...
0-17                NaN  ...  8095.827004  8970.200000  11624.344086
18-25       8771.542039  ...  9897.944820          NaN   9123.637192
26-35       8349.282740  ...  9797.160678          NaN   9337.963707
36-45       8746.010982  ...  9742.664574  9527.646370   9975.593918
46-50       8682.647727  ... 10087.766760  7635.979398  10071.946713
51-55       8877.236181  ...  9642.756528  9724.249300  10389.155138
55+         8187.529412  ...  9154.223602  9351.239220   8950.928839
All         8637.743761  ...  9796.640239  9306.351061   9500.702772

Occupation            15           16            17           18           19  \
Age
0-17                 NaN          NaN  10401.028571          NaN  8549.653036
18-25       10624.453642  9229.897026  10264.782454  9147.540092  8859.068400
26-35        9591.972651  9058.853607   9846.795476  9559.067321  8585.604671
36-45        9790.425145  9752.947966   9552.746001  8829.324165  8959.566468
46-50        9848.736534  9379.450528   9752.035178  8841.854982  8120.570881
51-55       10846.690661  9450.198826  10883.478639  9390.875706  9794.480000
55+          9502.238426  9284.601121   8661.888960  8466.196429  8151.571429
All          9778.891163  9394.464349   9821.478236  9169.655844  8710.627231

Occupation           20          All
Age
0-17                NaN  8933.464640
18-25       8461.383655  9169.663606
26-35       8763.913320  9252.690633
36-45       8992.289025  9331.350695
```

Activate Windows
Go to Settings to activate Windows.

```
All         8836.494905  9263.968713

[8 rows x 22 columns]
```





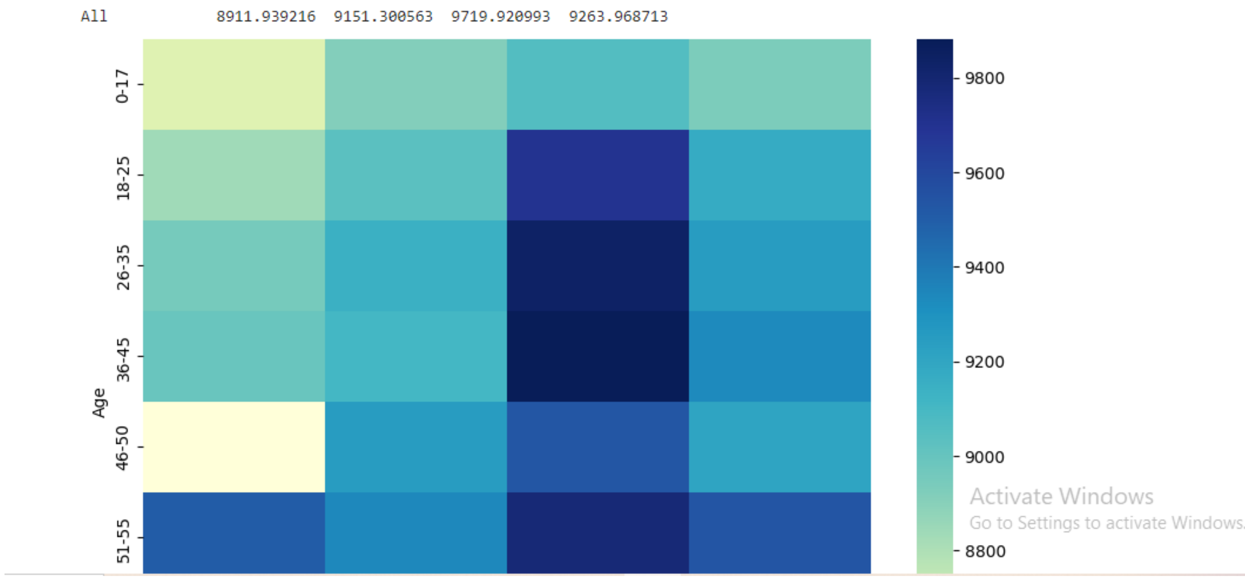
Age = 55+ and occupation =5 have highest purchase mean.

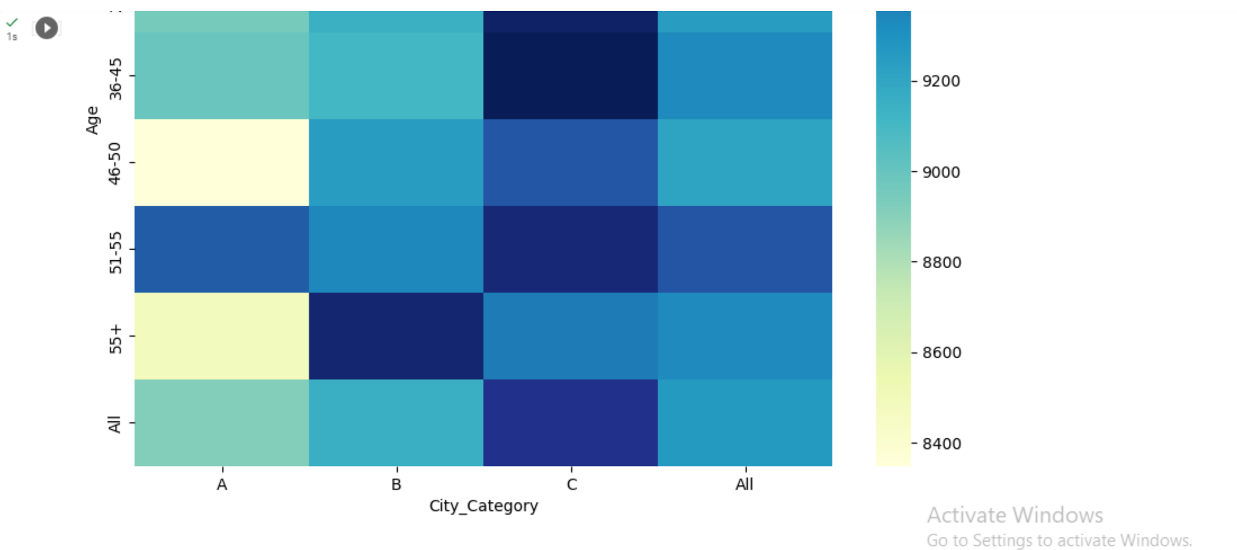
```
[67] df.columns
```

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

```
cross_tabs('Age','City_Category')
```

City_Category	A	B	C	All
Age				
0-17	8615.110456	8917.295308	9059.503299	8933.464640
18-25	8833.734084	9031.706985	9696.570919	9169.663606
26-35	8952.503004	9149.193178	9835.388993	9252.690633
36-45	8990.333997	9107.901067	9882.012654	9331.350695
46-50	8348.526752	9247.927129	9533.184023	9208.625697
51-55	9508.505001	9340.911392	9780.380806	9534.808031
55+	8485.945424	9803.560635	9385.316939	9336.280459
All	8911.939216	9151.300563	9719.920993	9263.968713

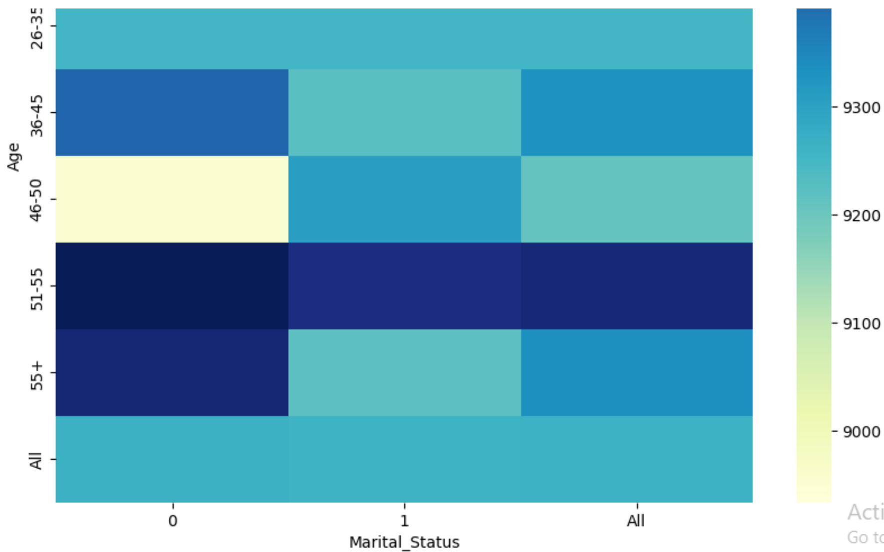




- City_category=C has highest purchase mean then compared to rest of them.
- Age=36-45 and city_category =c has highest concentration of purchase mean.

```
cross_tabs('Age','Marital_Status')
```

Marital_Status	0	1	All
Age			
0-17	8933.464640	NaN	8933.464640
18-25	9216.752419	8994.509992	9169.663606
26-35	9252.566484	9252.882410	9252.690633
36-45	9402.515329	9223.098451	9331.350695
46-50	8956.529551	9305.535821	9208.625697
51-55	9575.827475	9518.735088	9534.808031
55+	9539.774959	9218.510315	9336.280459
All	9265.907619	9261.174574	9263.968713



- purchase mean(marital =0) is nearly equal to purchase mean(marital =1).
- marital=0 and age =51-55 have highest purchase mean.

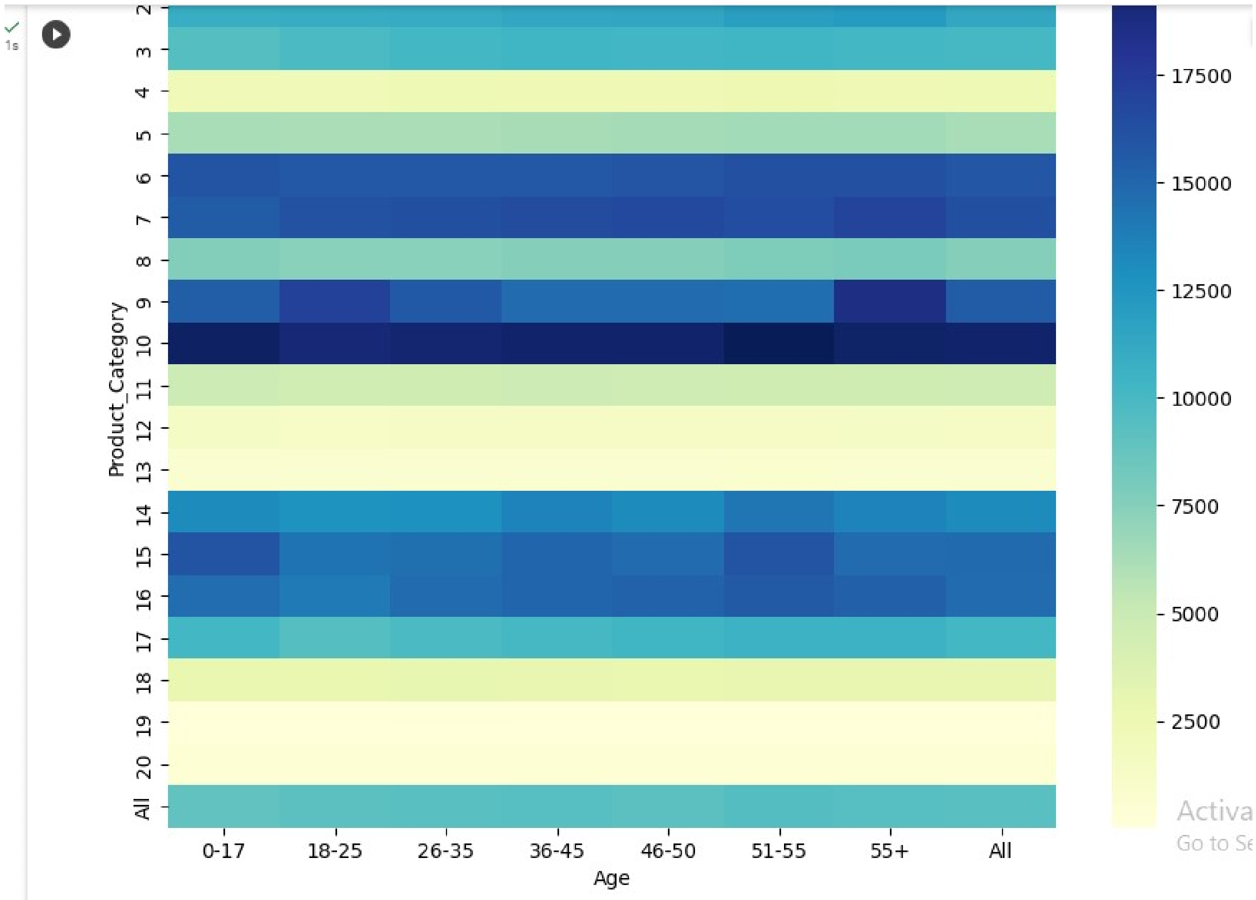
```python
cross_tabs('Product_Category','Age')
```

```
Age                  0-17          18-25         26-35         36-45  \
Product_Category
1                13607.600279  13448.852904  13456.256056  13767.068287
2                10851.982609  10966.741870  11083.914427  11375.946254
3                 9431.505000   9871.727601  10154.785043  10340.294240
4                 2244.659631   2194.358912   2340.182729   2400.508496
5                 6249.356120   6142.584040   6176.736014   6283.077612
6                15982.842105  15768.623900  15758.713848  15803.359323
7                15490.830189  16062.850312  16341.205330  16450.993820
8                 7632.767493   7387.642287   7400.653900   7528.592033
9                15434.875000  17127.650794  15673.753247  14689.448598
10               20038.495495  19192.218905  19560.216004  19651.978138
11                4808.166216   4597.116163   4677.053069   4762.744195
12                1423.712000   1273.038724   1322.249088   1350.266600
13                 737.258929    701.903439    716.744752    719.819200
14               13134.025641  12704.517391  12781.127660  13523.483974
15               15984.587500  14358.240234  14525.945616  14966.862366
16               14635.951965  13993.362328  14698.257649  14968.702302
17               10143.833333   9476.487805   9829.015748  10111.229630
18                2766.777778   2909.808260   3039.208253   2948.787749
19                  38.491525     36.047273     36.836590     37.025000
20                 368.011111    367.253731    376.505568    364.104743
All               8933.464640   9169.663606   9252.690633   9331.350695

Age                 46-50         51-55           55+           All
Product_Category
1                13778.098148  14125.773014  14065.121741  13606.218596
```



```python
cross_tabs('Product_Category','Age')
```

```
Age                 46-50         51-55           55+           All
Product_Category
1                13778.098148  14125.773014  14065.121741  13606.218596
2                11479.071734  11901.649635  12180.648619  11251.935384
3                10261.654797  10327.424242  10166.398357  10096.705734
4                 2420.013131   2445.980826   2387.732704   2329.659491
5                 6372.036672   6502.194885   6463.649339   6240.088178
6                15916.663379  16227.108276  16218.627610  15838.478550
7                16619.957187  16376.913534  16942.664179  16365.689600
8                 7532.639452   7772.095824   7892.292043   7498.958078
9                14759.393939  14576.689655  18626.375000  15537.375610
10               19654.736538  20530.747592  19828.285714  19675.570927
11                4742.285171   4628.307270   4640.276292   4685.268456
12                1383.450000   1380.413395   1431.038235   1350.859894
13                 730.704174    762.842650    738.362126    722.400613
14               13117.140940  14161.207792  13563.560000  13141.625739
15               14709.913621  16021.627953  14759.820961  14780.451828
16               15186.213879  15629.892857  15290.127321  14766.037037
17               10296.610526  10538.009346  10600.805970  10170.759516
18                2896.242165   2964.732861   2993.800830   2972.864320
19                  37.053691     37.910448     38.893204     37.041797
20                 384.180617    361.800000    359.100000    370.481176
All               9208.625697   9534.808031   9336.280459   9263.968713
```



Outlier:

since purchase is the only continous variable we will check that

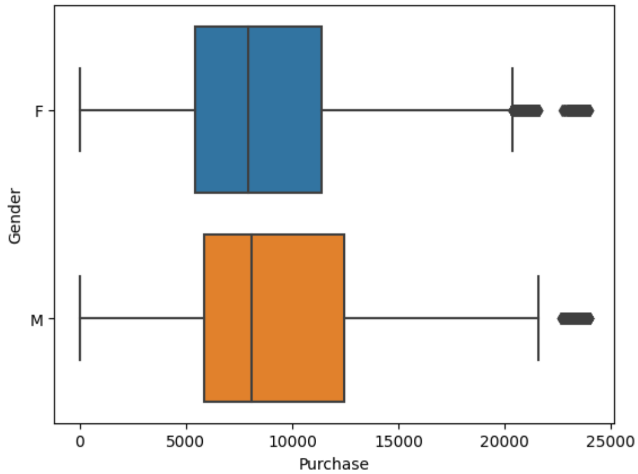
```
[71] def detetct_out(df,a):
        q1=np.quantile(df[a],0.25)
        q3=np.quantile(df[a],0.75)
        low_end=q1-1.5*(q3-q1)
        high_end=q3+1.5*(q3-q1)
        b=df[df[a]>high_end][a].tolist()
        c=df[df[a]<low_end][a].tolist()
        b=set(b)
        c=set(c)
        print("Outliers")
        print("Outliers which has high values --->",b)
        print("---------------------------------------------------")
        print("Outliers which has low values --->",c)
```

[72]
```
detetct_out(df,'Purchase')
```

```
Outliers
Outliers which has high values ---> {23610, 23612, 23624, 23856, 23630, 22651, 22656, 22666, 22668, 22678, 22684, 22710, 22719, 22730,
--------------------------------------------------
Outliers which has low values ---> set()
```

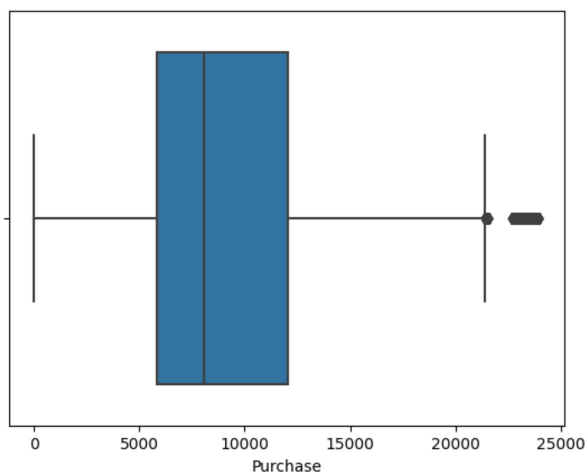
[73]
```
sns.boxplot(x=df['Purchase'],y=df['Gender'])
```

<Axes: xlabel='Purchase', ylabel='Gender'>



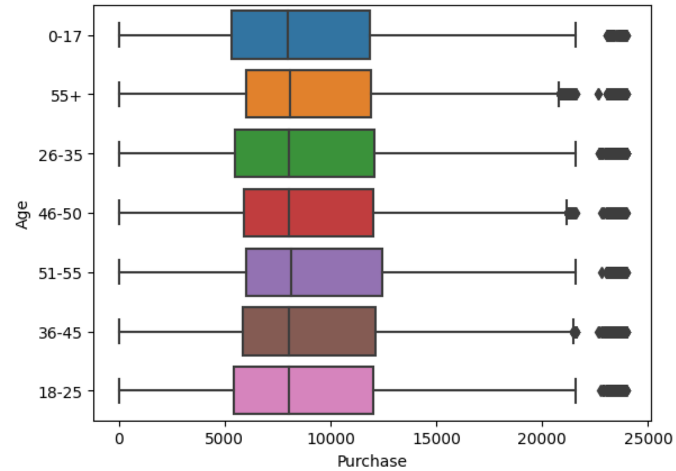

```
sns.boxplot(x=df['Purchase'])
```

<Axes: xlabel='Purchase'>



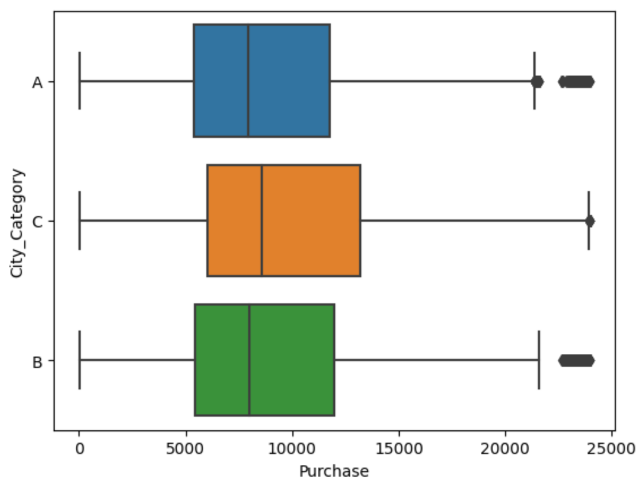
```
[75] sns.boxplot(x=df['Purchase'],y=df['Age'])
```

```
<Axes: xlabel='Purchase', ylabel='Age'>
```




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

```
<Axes: xlabel='Purchase', ylabel='City_Category'>
```



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

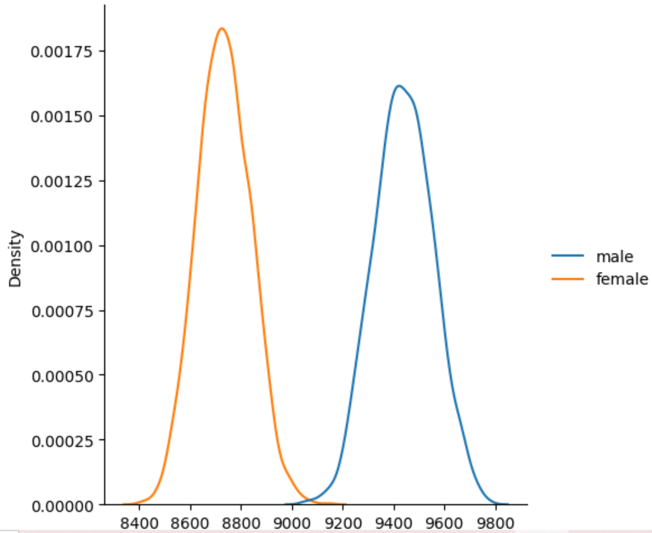
```
df_male=df[df['Gender']=='M']
df_fmale=df[df['Gender']=='F']
```

- By this density graph we can say that males spend more money per transaction than female.
- But these dist doesn’t follow Gaussian.

```
# this is the boot strap fuction which gives us the list_of_means
def boot_strap(data,sample_size,no_of_samples):
    list_of_means11 = []
    for i in range(no_of_samples):
        bootstrapped_samples = np.random.choice(data, size=sample_size)
        list_of_means11.append(np.mean(bootstrapped_samples))
    return list_of_means11
```

```
male_means=boot_strap(df_male['Purchase'],2000,2000)
fmale_means=boot_strap(df_fmale['Purchase'],2000,2000)
```

```
a=sns.displot({'male':male_means,'female':fmale_means},kind='kde')
```



Activate Windows
Go to Settings to activate Windows.

- Seeing above plot we can say that mean_purchase_amount_per_trans(female)<mean_purchase_amount_per_trans(male)
- for sample size=2000
- These data are for each transactions.

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

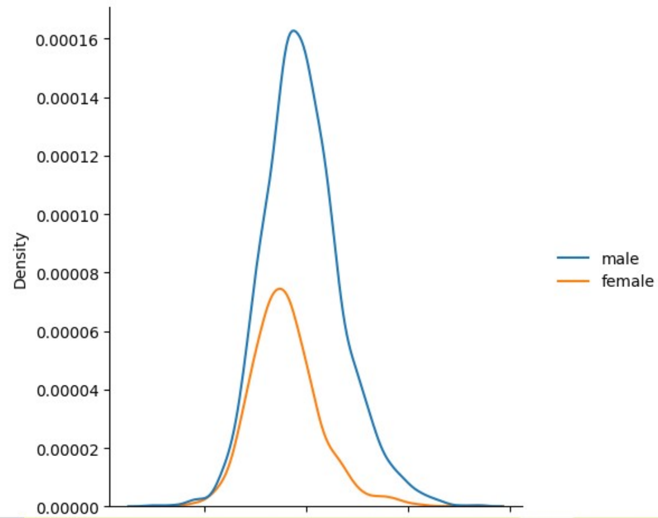
```
df_male_expense=df
```

```
[82] #total expense of each user
     # filter by gender and group by userid  agg as purchase mean
     #this data is for each customers
     df_male_agg=pd.DataFrame(df_male.groupby(df['User_ID'])['Purchase'].mean()).reset_index()
     df_fmale_agg=pd.DataFrame(df_fmale.groupby(df['User_ID'])['Purchase'].mean()).reset_index()
```

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

9806.867524226629
```

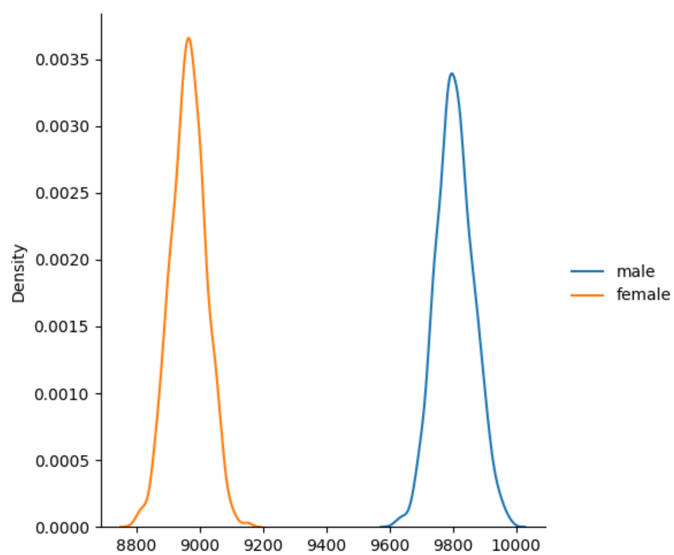
```
[ ] df_fmale_agg['Purchase'].mean()

8965.19846393646
```

```python
a=sns.displot({'male':df_male_agg['Purchase'],'female':df_fmale_agg['Purchase']},kind='kde')
```




```python
[87] male_means_agg=boot_strap(df_male_agg['Purchase'],1000,1000)
     fmale_means_agg=boot_strap(df_fmale_agg['Purchase'],1000,1000)
```

```python
a=sns.displot({'male':male_means_agg,'female':fmale_means_agg},kind='kde')
```




```python
[89] np.mean(fmale_means_agg)
```

```
8963.536885498597
```

```python
[90] np.mean(male_means_agg)
```

```
9805.344194854972
```

- Seeing above plot we can say that mean_expense(female)<mean_expense(male).

```python
#this gives a confidnce interval
def Confidance_interverl(list_ofmeans,left_val,right_val):
    left = np.percentile(list_ofmeans, 2.5)
    right = np.percentile(list_ofmeans, 97.5)
    return left,right
```

```python
def Boot_Strap_Main(data,sample_size,no_of_samples,CI_left,CI_right):
    list_of_means=boot_strap(data,sample_size,no_of_samples)
    left,right=Confidance_interverl(list_of_means,CI_left,CI_right)
    diff_left_right=right-left
    print("Sample Size =",sample_size)
    print("no_of_samples =",no_of_samples)
    print("Confidence interval: ", [left, right])
    print("difference",diff_left_right)
    return diff_left_right,left,right,list_of_means
    #sns.histplot(list_of_means)
```

- ## Confidence Interval for CI=95

```python
#male
diff_left_right,left,right,list_of_means=Boot_Strap_Main(df_male_agg['Purchase'],sample_size=1000,no_of_samples=1000,
```

```
Sample Size = 1000
no_of_samples = 1000
Confidence interval:  [9695.785767042229, 9933.130500112686]
difference 237.34473307045664
```

```python
#male
diff_left_right,left,right,list_of_means=Boot_Strap_Main(df_male_agg['Purchase'],sample_size=2000,no_of_samples=2000,
```

```
Sample Size = 2000
no_of_samples = 2000
Confidence interval:  [9724.77720180956, 9895.708049377508]
difference 170.93084756794815
```

```python
#male
diff_left_right,left,right,list_of_means=Boot_Strap_Main(df_male_agg['Purchase'],sample_size=3000,no_of_samples=3000,
```

```
Sample Size = 3000
no_of_samples = 3000
Confidence interval:  [9740.976465217387, 9876.401658896211]
difference 135.42519367882414
```

```
#female

diff_left_right,left,right,list_of_means=Boot_Strap_Main(df_fmale_agg['Purchase'],sample_size=1000,no_of_samples=1000
```

```
Sample Size = 1000
no_of_samples = 1000
Confidence interval:  [8865.583416769196, 9072.401911990624]
difference 206.81849522142875
```

```
#female

diff_left_right,left,right,list_of_means=Boot_Strap_Main(df_fmale_agg['Purchase'],sample_size=2000,no_of_samples=2000
```

```
Sample Size = 2000
no_of_samples = 2000
Confidence interval:  [8893.002292525807, 9039.765151241858]
difference 146.7628587160507
```

```
#female

diff_left_right,left,right,list_of_means=Boot_Strap_Main(df_fmale_agg['Purchase'],sample_size=3000,no_of_samples=3000
```

```
Sample Size = 3000
no_of_samples = 3000
Confidence interval:  [8905.464480920102, 9025.165019475422]
difference 119.70053855532024
```

- As the sample size increase the range of the CI will become shorter.

Confidence Interval for CI=90

```
#male

diff_left_right,left,right,list_of_means=Boot_Strap_Main(df_male_agg['Purchase'],sample_size=1000,no_of_samples=1000,
```

```
Sample Size = 1000
no_of_samples = 1000
Confidence interval:  [9697.242159777115, 9922.12545115513]
difference 224.8832913780152
```

```
#male

diff_left_right,left,right,list_of_means=Boot_Strap_Main(df_male_agg['Purchase'],sample_size=2000,no_of_samples=2000,
```

```
Sample Size = 2000
no_of_samples = 2000
Confidence interval:  [9722.906920377265, 9890.760087585371]
difference 167.85316720810624
```

```
#female

diff_left_right,left,right,list_of_means=Boot_Strap_Main(df_fmale_agg['Purchase'],sample_size=1000,no_of_samples=1000
```

```
Sample Size = 1000
no_of_samples = 1000
Confidence interval:  [8863.905741319242, 9072.589449924466]
difference 208.68370860522373
```

```
#female

diff_left_right,left,right,list_of_means=Boot_Strap_Main(df_fmale_agg['Purchase'],sample_size=2000,no_of_samples=2000
```

```
Sample Size = 2000
no_of_samples = 2000
Confidence interval:  [8891.516362658947, 9038.795940008627]
difference 147.27957734968004
```

- From the above bootstrap experiment we can say that
- expense_of_male > expense_of_female
- We have gving CI for different sample sizes and diff no_of_sample
- FOR EXMP:

  Sample Size = 2000 no_of_samples = 2000 Confidence interval: [8893.661760614004, 9042.044923426136] difference 148.38316281213156.

- we are 95% confidence that the population of female expense lie between [8893.661760614004, 9042.044923426136].

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

male:

- Sample Size = 1000
- no_of_samples = 1000
- Confidence interval: [9691.280523604808, 9923.624993824715]
- difference 232.34447021990673
- Sample Size = 2000
- no_of_samples = 2000
- Confidence interval: [9725.1728787738, 9889.376693510063]
- difference 164.20381473626367
- female:
- Sample Size = 1000
- no_of_samples = 1000
- Confidence interval: [8860.035528719416, 9068.177174773075]
- difference 208.1416460536584

- Sample Size = 2000
- no_of_samples = 2000
- Confidence interval: [8893.661760614004, 9042.044923426136]
- difference 148.38316281213156
- we see here that CI of avg. male and avg. female spending does not over lap
- the spending habbits of males and females are different
- walmart should show diff offers for males and diff for females customers

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

```
[95] df_unmarried=df[df['Marital_Status']==0]
     df_married=df[df['Marital_Status']==1]
```

```
[96] df_unmarried['Purchase'].mean()
```

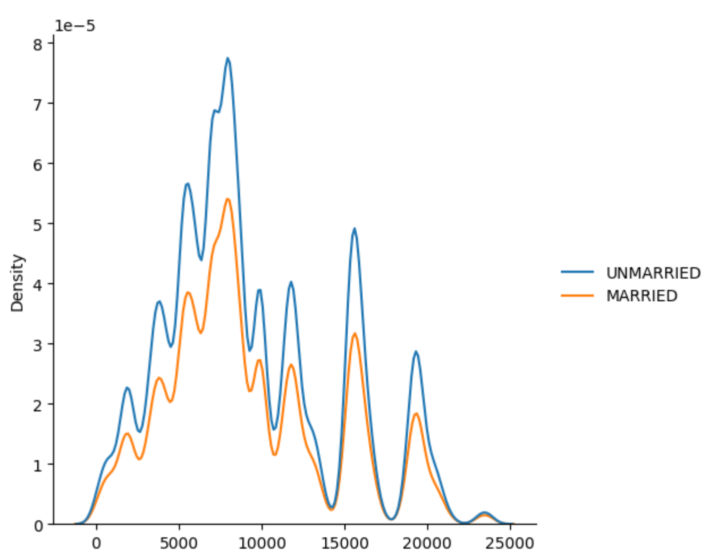
```
9265.907618921507
```



```
[97] df_married['Purchase'].mean()
```

```
9261.174574082374
```

```
a=sns.displot({'UNMARRIED':df_unmarried['Purchase'],'MARRIED':df_married['Purchase']},kind='kde')
```

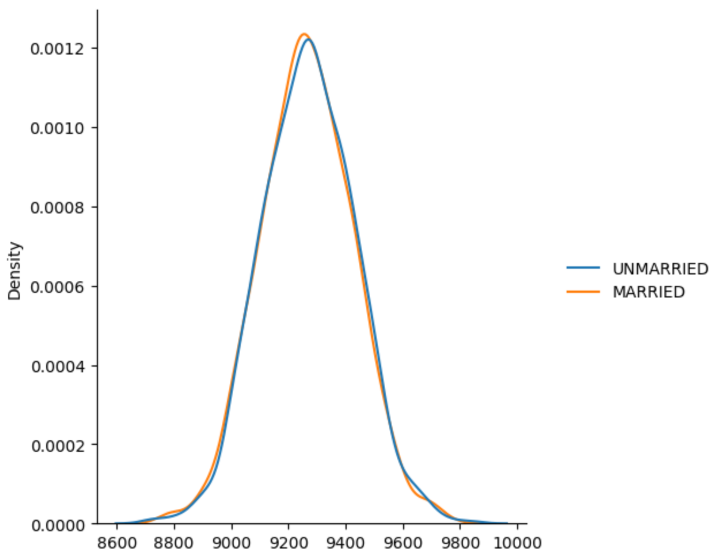


- By this density graph we cannot say that unmarried spend more money per transaction than married.

```
[99] # this is the boot strap fuction which gives us the list_of_means
     def boot_strap(data,sample_size,no_of_samples):
         list_of_means11 = []
         for i in range(no_of_samples):
             bootstrapped_samples = np.random.choice(data, size=sample_size)
             list_of_means11.append(np.mean(bootstrapped_samples))
         return list_of_means11
```

```
[100] unmarried_df=boot_strap(df_unmarried['Purchase'],1000,1000)
      married_df=boot_strap(df_married['Purchase'],1000,1000)
```

```
a=sns.displot({'UNMARRIED':unmarried_df,'MARRIED':married_df},kind='kde')
```



- Both married and unmarried have mean_purchase_per_trans

  their spending habbits are all most same.

```
#total expense of each user
# filter by marital and group by userid  agg as purchase mean
#this data is for each customers
df_married_agg=pd.DataFrame(df_married.groupby(df['User_ID'])['Purchase'].mean()).reset_index()
df_unmarried_agg=pd.DataFrame(df_unmarried.groupby(df['User_ID'])['Purchase'].mean()).reset_index()
```

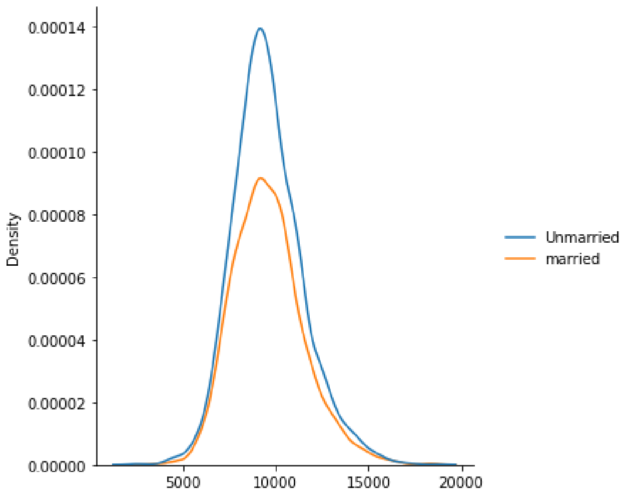
```
df_married_agg['Purchase'].mean()
```

```
9574.96229903175
```

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

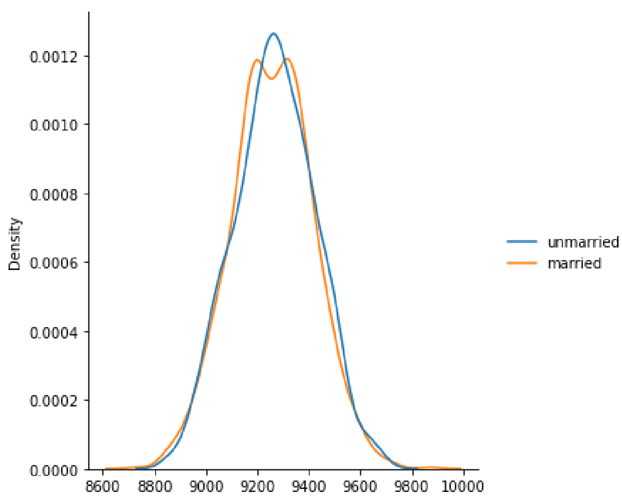
```
9564.407141636288
```

```
a=sns.displot({'Unmarried':df_unmarried_agg['Purchase'],'married':df_married_agg['Purchase']},kind='kde')
```




```
unmarried_means_agg=boot_strap(df_unmarried['Purchase'],1000,1000)
married_means_agg=boot_strap(df_married['Purchase'],1000,1000)
```

```
a=sns.displot({'unmarried':unmarried_means_agg,'married':married_means_agg},kind='kde')
```



Confidence Interval for CI=95

```
#unmarried

diff_left_right,left,right,list_of_means=Boot_Strap_Main(df_unmarried_agg['Purchase'],sample_size=1000,no_of_samples=
```

```
Sample Size = 1000
no_of_samples = 1000
Confidence interval:  [9453.970767662451, 9680.16655655537]
difference 226.19578889291915
```

```
#unmarried

diff_left_right,left,right,list_of_means=Boot_Strap_Main(df_unmarried_agg['Purchase'],sample_size=2000,no_of_samples=
```

```
Sample Size = 2000
no_of_samples = 2000
Confidence interval:  [9480.19275685655, 9647.901860109312]
difference 167.70910325276236
```

```
#married

diff_left_right,left,right,list_of_means=Boot_Strap_Main(df_married_agg['Purchase'],sample_size=1000,no_of_samples=10
```

```
Sample Size = 1000
no_of_samples = 1000
Confidence interval:  [9450.577918379573, 9693.48242479901]
difference 242.90450641943607
```

```
#married

diff_left_right,left,right,list_of_means=Boot_Strap_Main(df_married_agg['Purchase'],sample_size=2000,no_of_samples=20
```



```
#married

diff_left_right,left,right,list_of_means=Boot_Strap_Main(df_married_agg['Purchase'],sample_size=2000,no_of_samples=20
```

```
Sample Size = 2000
no_of_samples = 2000
Confidence interval:  [9492.45794465459, 9660.311567213]
difference 167.85362255841028
```

- From the above bootstrapping we can say that expense of both married and unmarried are having same distribution.
- Population mean of married people lie in [9492.437336895322, 9655.03598509862] with Confidence of 95%.
- Population mean of unmarried people lie in [9480.699752043949, 9646.014401806344] with Confidence of 95%.
- we see that there is clear overlapping btw the two distribution , which means these people are having same spending pattern.
- Walmart can send same offers or products to these segments.

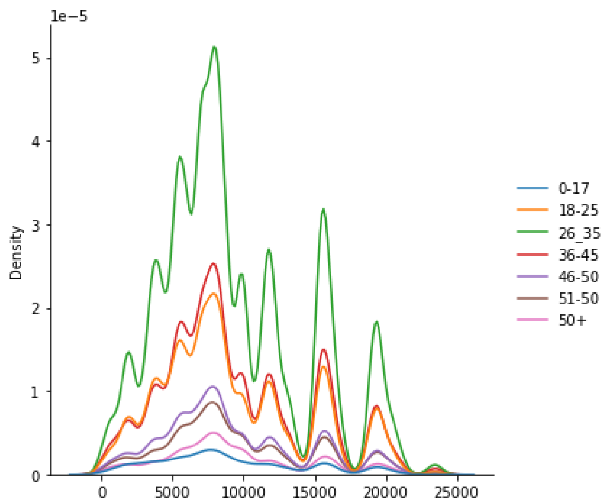
Results when the same activity is performed for Age

```
[19] df['Age'].unique()

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

```
df_age_0_17=df[df['Age']=='0-17']
df_age_18_25=df[df['Age']=='18-25']
df_age_26_35=df[df['Age']=='26-35']
df_age_36_45=df[df['Age']=='36-45']
df_age_46_50=df[df['Age']=='46-50']
df_age_51_55=  df[df['Age']=='51-55']
df_age_55plus=df[df['Age']=='55+']
```

```
d={'0-17':df_age_0_17['Purchase'],'18-25':df_age_18_25['Purchase'],'26_35':df_age_26_35['Purchase'],'36-45':df_age_36
```

```
a=sns.displot(d,kind='kde')
```



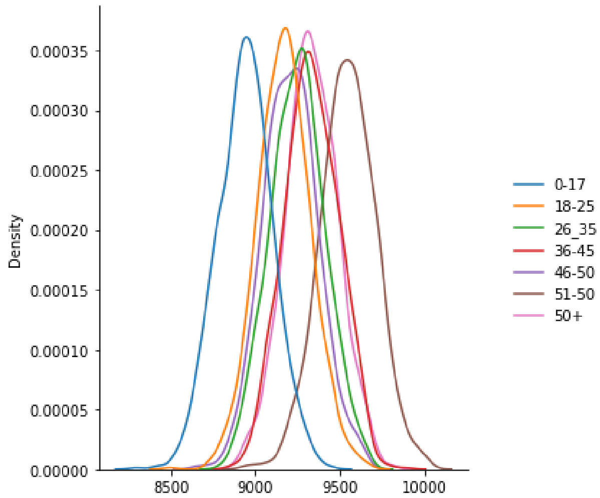
- By this density we cannot conclude anything for now.

```
df_age_0_17_df=boot_strap(df_age_0_17['Purchase'],1000,1000)
df_age_18_25_df=boot_strap(df_age_18_25['Purchase'],1000,1000)
df_age_26_35_df=boot_strap(df_age_26_35['Purchase'],1000,1000)
df_age_36_45_df=boot_strap(df_age_36_45['Purchase'],1000,1000)
df_age_46_50_df=boot_strap(df_age_46_50['Purchase'],1000,1000)
df_age_51_55_df=boot_strap(df_age_51_55['Purchase'],1000,1000)
df_age_55plus_df=boot_strap(df_age_55plus['Purchase'],1000,1000)
```

```
d1={'0-17':df_age_0_17_df,'18-25':df_age_18_25_df,'26_35':df_age_26_35_df,'36-45':df_age_36_45_df,'46-50':df_age_46_5
```

```
sns.displot(d1,kind='kde')
```

<seaborn.axisgrid.FacetGrid at 0x1e551d838e0>

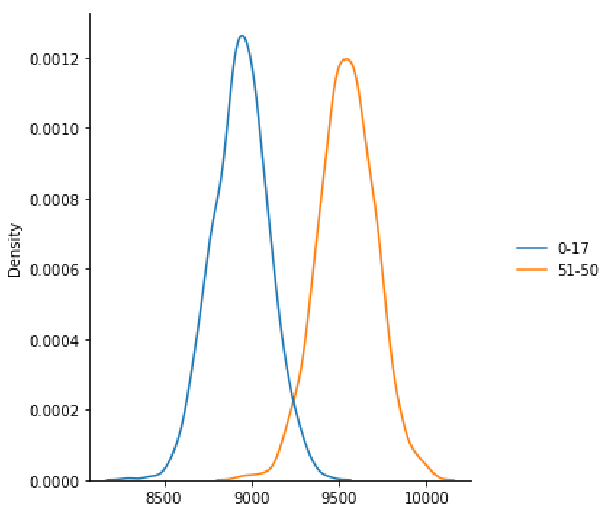


Note : this distribution is for per each transaction

- from this plot we can observe that 0-17 group has less mean_purchase_per_trans
- 51-50 has more mean_purchase_per_trans.

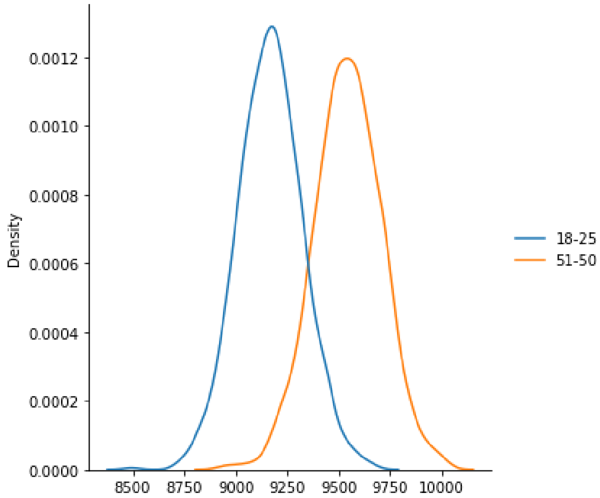
```
sns.displot({'0-17':d1['0-17'],'51-50':d1['51-50']},kind='kde')
```

<seaborn.axisgrid.FacetGrid at 0x1e5522c23d0>



```
sns.displot({'18-25':d1['18-25'],'51-50':d1['51-50']},kind='kde')
```

```
<seaborn.axisgrid.FacetGrid at 0x1e551208ac0>
```




```
c_a=[]
for i in d1:
    print(i,np.mean(d1[i]))
    c_a.append((i,np.mean(d1[i])))
```

```
0-17 8937.591179000001
18-25 9168.293978
26_35 9247.623647
36-45 9330.673756999999
46-50 9207.396045
51-50 9544.658206999999
50+ 9330.484889
```

```
c_a.sort(key=lambda x : x[1])
print("MEANS IN ACCENDING ORDER")
for i in c_a:
    print(i)
```

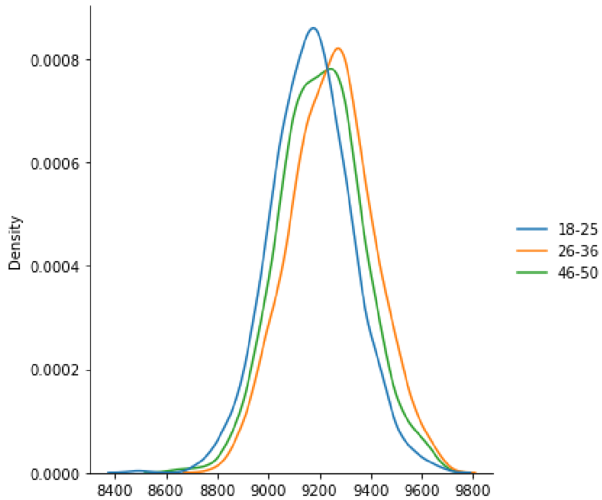
```
MEANS IN ACCENDING ORDER
('0-17', 8937.591179000001)
('18-25', 9168.293978)
('46-50', 9207.396045)
('26_35', 9247.623647)
('50+', 9330.484889)
('36-45', 9330.673756999999)
('51-50', 9544.658206999999)
```

- form here we can see that age group of 18-25 26-35 46-50 have similar mean_purchase_per_trans

- 36-45 and 50+ have similar mean_purchase_per_trans.

```
sns.displot({'18-25':d1['18-25'],'26-36':d1['26_35'],'46-50':d1['46-50']},kind='kde')
```

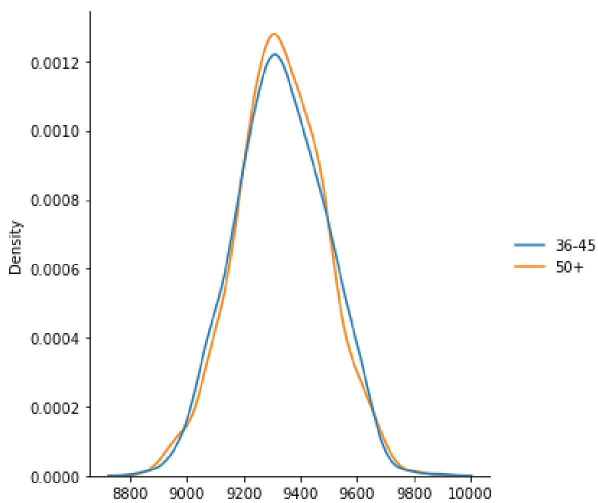
<seaborn.axisgrid.FacetGrid at 0x1e55199b340>



- these three are having similar mean_purchase_per_trans.

```
sns.displot({'36-45':d1['36-45'],'50+':d1['50+']},kind='kde')
```

<seaborn.axisgrid.FacetGrid at 0x1e5521f3220>



- These two are having similar mean_purchase_per_trans
- By all these observation we can say that :
    a. 18-25 26-35 46-50 ---> similar purchase habbit.
    b. 36-45 and 50+ -----> similar purchase habbit.

```python
li=[df_age_0_17,df_age_18_25,df_age_26_35,df_age_36_45,df_age_46_50,df_age_51_55,df_age_55plus]
ki=['0-17','18-25',  '26-35', '36-45','46-50', '51-55', '55+' ]
dd_1={}
for i,j in zip(li,ki):
    caa=pd.DataFrame(i.groupby(df['User_ID'])['Purchase'].mean()).reset_index()
    dd_1[j]=caa
```

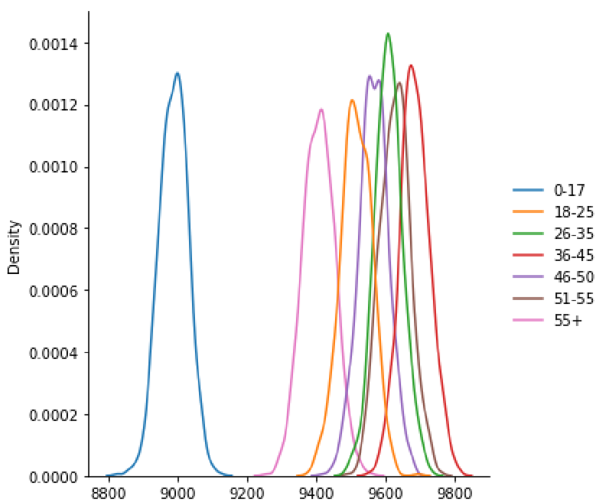
```python
for i in dd_1:
    print(i,"--------->",dd_1[i]['Purchase'].mean())
```

```
0-17 ---------> 9230.887075064571
18-25 ---------> 9693.617202810385
26-35 ---------> 9782.847513500566
36-45 ---------> 9871.050692299219
46-50 ---------> 9736.78272404507
51-55 ---------> 9862.537887574703
55+ ---------> 9652.856875499612
```

```python
dd_agg_val={}
for i in dd_1:
    dd_agg_val[i]=boot_strap(dd_1[i]['Purchase'],2000,1000)
```

```python
sns.displot(dd_agg_val,kind='kde')
```

```
<seaborn.axisgrid.FacetGrid at 0x1e553696070>
```



- By this plot be can say that age group 0-17 people has less expense
- age group 36-45 has more expense

Note : this distribution is for per each customer (ie group by customer mean purchase),

Confidence Interval for CI=95

```
for i in dd_agg_val:
    print(i)
    Boot_Strap_Main(dd_agg_val[i],sample_size=1000,no_of_samples=1000,CI_left=2.5,CI_right=97.5)
    print("-------------------------------------------------------------------------------------------------")
```

```
0-17
Sample Size = 1000
no_of_samples = 1000
Confidence interval:  [8985.024506812832, 8989.932183423472]
difference 4.9076766106409195
-------------------------------------------------------------------------------------------------
18-25
Sample Size = 1000
no_of_samples = 1000
Confidence interval:  [9512.345092769223, 9517.929874486354]
difference 5.5847817171306815
-------------------------------------------------------------------------------------------------
26-35
Sample Size = 1000
no_of_samples = 1000
Confidence interval:  [9606.221232453303, 9611.381572302442]
difference 5.160339849138836
-------------------------------------------------------------------------------------------------
36-45
Sample Size = 1000
no_of_samples = 1000
Confidence interval:  [9683.03618018958, 9688.192904407266]
difference 5.156724217686133
-------------------------------------------------------------------------------------------------
46-50
Sample Size = 1000
no_of_samples = 1000
Confidence interval:  [9563.214162392609, 9568.444491852164]
difference 5.230329459554923

-------------------------------------------------------------------------------------------------
51-55
Sample Size = 1000
no_of_samples = 1000
Confidence interval:  [9625.302556984489, 9630.772660531858]
difference 5.470103547369945
-------------------------------------------------------------------------------------------------
55+
Sample Size = 1000
no_of_samples = 1000
Confidence interval:  [9404.031913662106, 9409.844870796253]
difference 5.812957134146927
-------------------------------------------------------------------------------------------------
```

```python
for i in dd_agg_val:
    print(i)
    Boot_Strap_Main(dd_agg_val[i],sample_size=2000,no_of_samples=1000,CI_left=2.5,CI_right=97.5)
    print("------------------------------------------------------------------------------------------------")
```

```
0-17
Sample Size = 2000
no_of_samples = 1000
Confidence interval:  [8985.670922734364, 8989.200994276343]
difference 3.5300715419798507
------------------------------------------------------------------------------------------------
18-25
Sample Size = 2000
no_of_samples = 1000
Confidence interval:  [9513.099232797209, 9517.053659503894]
difference 3.954426706684899
------------------------------------------------------------------------------------------------
26-35
Sample Size = 2000
no_of_samples = 1000
Confidence interval:  [9607.048191911228, 9610.615217921533]
difference 3.5670260103051987
------------------------------------------------------------------------------------------------
36-45
Sample Size = 2000
no_of_samples = 1000
Confidence interval:  [9683.768751240566, 9687.157977561617]
difference 3.3892263210509554
------------------------------------------------------------------------------------------------
46-50
Sample Size = 2000
no_of_samples = 1000
Confidence interval:  [9564.141374623709, 9567.83443381828]
difference 3.6930591945711058
```



```
51-55
Sample Size = 2000
no_of_samples = 1000
Confidence interval:  [9626.217945403787, 9629.969938744862]
difference 3.7519933410749218
------------------------------------------------------------------------------------------------
55+
Sample Size = 2000
no_of_samples = 1000
Confidence interval:  [9404.830228852543, 9408.679633437247]
difference 3.849404584703734
------------------------------------------------------------------------------------------------
```

- From these observation we can say that :
- age group 0-17 do not overlap with any other age group
- rest all the age group overlap with each other
- We can say that population mean lies btw these given CI with confidence of 95%
- expense of 0-17 is less when compared to all
- expense of 36-45 is more.

With All these sample means with certain confidence interval , we can suggest that the population mean lies between these CI, with certain confidence.


Walmart