Business Case: Walmart - Confidence Interval and CLT

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).

Defining Problem Statement and Analyzing basic metrics

Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), statistical summary

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objs as go
from scipy import stats
import plotly.subplots as sp
from scipy.stats import norm
```

```
!wget https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094
```

```
--2023-10-17 12:40:57--  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 108.157.172.183, 108.157.172.10, 108.157.172.176, ...
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|108.157.172.183|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 23027994 (22M) [text/plain]
Saving to: 'walmart_data.csv?1641285094'

walmart_data.csv?16 100%[===================>]  21.96M  92.7MB/s    in 0.2s

2023-10-17 12:40:57 (92.7 MB/s) - 'walmart_data.csv?1641285094' saved [23027994/23027994]
```

```
df = pd.read_csv('walmart_data.csv?1641285094')
df
```

	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
...	...	...	...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	B	1	1	20	368
550064	1006035	P00375436	F	26-35	1	C	3	0	20	371
550065	1006036	P00375436	F	26-35	15	B	4+	1	20	137
550066	1006038	P00375436	F	55+	1	C	2	0	20	365
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	490

550068 rows × 10 columns

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

```
df.shape
```

```
(550068, 10)
```

```
df.columns
```

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

```
df.isna().sum()
```

```
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
```

There are no missing values in the dataset.

```
df.nunique().sort_values(ascending=False)
```

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

```
df.duplicated().sum()
```

```
0
```

```
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
```

Most of the columns are of object type except User_ID, Occupation, Marital_Status, Product_Category and Purchase.

```
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

```
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

Altering Data types

Replacing numerical values with meaningful labels in the 'Marital_Status' By using 'Married' and 'Unmarried', we make it easier younderstand the data

```
df['User_ID']=df['User_ID'].astype(object)
df['Marital_Status']=df['Marital_Status'].astype(object)
df['Occupation']=df['Occupation'].astype(object)
df['Product_Category']=df['Product_Category'].astype(object)
```

```
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  object
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  object
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  object
 8   Product_Category            550068 non-null  object
 9   Purchase                    550068 non-null  int64
dtypes: int64(1), object(9)
memory usage: 42.0+ MB
```

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

```
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	Unmarried	3	8370
1	1000001	P00248942	F	0-17	10	A	2	Unmarried	1	15200
2	1000001	P00087842	F	0-17	10	A	2	Unmarried	12	1422
3	1000001	P00085442	F	0-17	10	A	2	Unmarried	12	1057
4	1000002	P00285442	M	55+	16	C	4+	Unmarried	8	7969

- There are 5891 unique users. User ID 1001680 has shopped the most frequent from Walmart.
- There are 3631 unique products. Product ID P00265242 is the most frequent sold item.
- Men are more frequent buyers than Females.
- There are 7 unique age categories. The most frequent buyers fall under the age group of 26-35.
- There are 3 different city categories. Most frequent buyers fal under category B.
- Most people are in the current city since 1 year.
- Most customerd are unmarried.

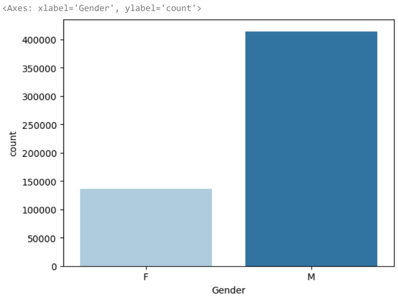
Univariate Analysis

```
np.round(df['Gender'].value_counts(normalize=True)*100,2)
```

```
M    75.31
F    24.69
Name: Gender, dtype: float64
```

```
sns.countplot(df,x='Gender',palette = "Paired")
```

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

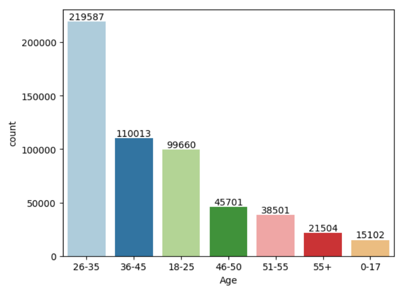


75% records are of men and 25% of women.

```
np.round(df['Age'].value_counts(normalize=True)*100,2)
```

```
26-35    39.92
36-45    20.00
18-25    18.12
46-50     8.31
51-55     7.00
55+       3.91
0-17      2.75
Name: Age, dtype: float64
```

```
label =sns.countplot(df,x='Age',palette = "Paired",order = df['Age'].value_counts().index)
for i in label.containers:
    label.bar_label(i)
```



40% of the buyers fall under the age group of 26-35 which is the highest amongst all age groups.

Approximately 0.21 million records are present for age group 26-35 followed by 0.11 million records for group 36-45.

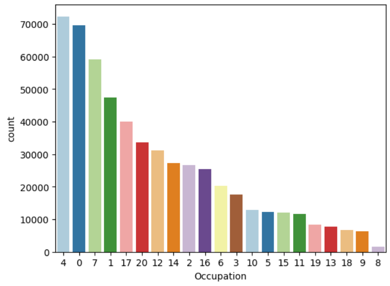
Age group 0-17 and 55+ are the least frequent buyers which is only 3% and 4% of the data respectively.

Approximately only 15k and 21k records are there for age group 0-17 and group 55+.

We can observe that most buyers are in within the age of 18-45 before and after this range we can see less buyers.

```
sns.countplot(df,x='Occupation',palette = "Paired",order = df['Occupation'].value_counts().index)
```

```
<Axes: xlabel='Occupation', ylabel='count'>
```



People having occupation 4 are the most frequent buyers followed by occupation 0 and 7.

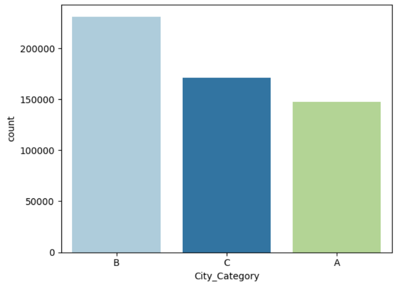
People having occupation 8 are the least frequent buyers followed by occupation 9 and 18.

```
np.round(df['City_Category'].value_counts(normalize=True)*100,2)
```

```
B    42.03
C    31.12
A    26.85
Name: City_Category, dtype: float64
```

```
sns.countplot(df,x='City_Category',palette = "Paired",order = df['City_Category'].value_counts().index)
```

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



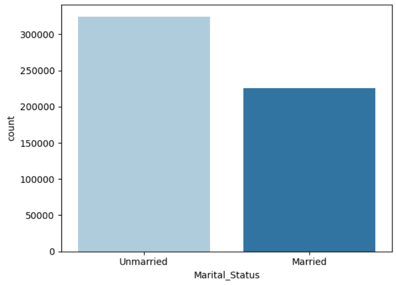
There are 42% buyers from City Category B, 31% from Category C and 27% from Category A

```
np.round(df['Marital_Status'].value_counts(normalize=True)*100,2)
```

```
Unmarried    59.03
Married      40.97
Name: Marital_Status, dtype: float64
```

```
sns.countplot(df,x='Marital_Status',palette = "Paired",order = df['Marital_Status'].value_counts().index)
```

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



We can observe that 59% of the frequent buyers are of unmarried people, while 41% of married.

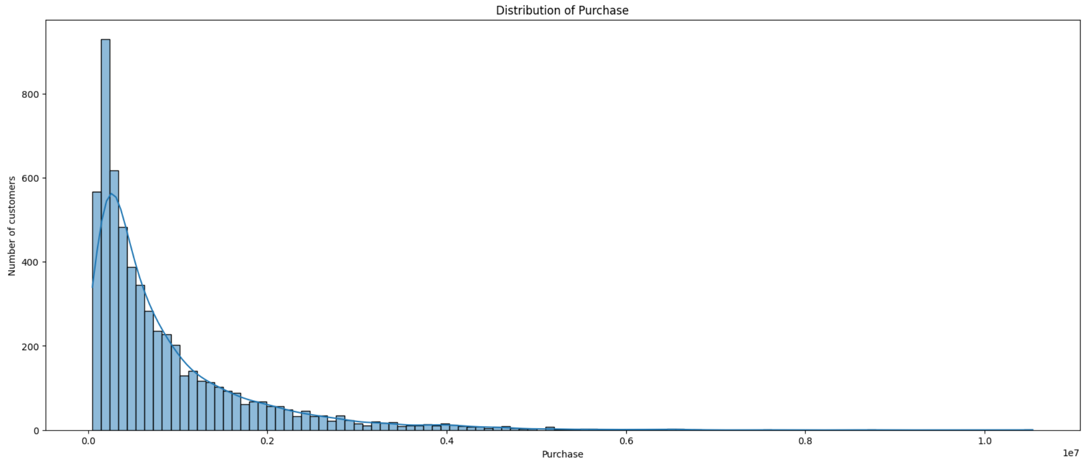
```
np.round(df['Product_Category'].value_counts(normalize=True)*100,2)
```

```
5     27.44
1     25.52
8     20.71
11     4.42
2      4.34
6      3.72
3      3.67
4      2.14
16     1.79
15     1.14
13     1.01
10     0.93
12     0.72
7      0.68
18     0.57
20     0.46
19     0.29
14     0.28
17     0.11
9      0.07
Name: Product_Category, dtype: float64
```

```
sns.countplot(df,x='Product_Category',palette = "Paired",order = df['Product_Category'].value_counts().index)
```

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

```
purchase_df=df.groupby(['User_ID']).agg(purchase_sum=('Purchase','sum')).reset_index()
plt.figure(figsize=(20,8))
sns.histplot(data=purchase_df,x='purchase_sum',kde=True)
plt.xlabel('Purchase')
plt.ylabel('Number of customers')
plt.title('Distribution of Purchase')
plt.show()
```



The most frequent bought product category is 5 followed by 1 and 8.

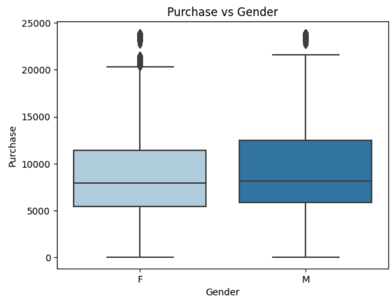
All the other categories are not much touched.

The least frequent bought are category 9 followed by 17 and 14.

Bi-variate Analysis

Pruchase habits based on gender

```
sns.boxplot(data = df, y ='Purchase', x = 'Gender', palette = "Paired")
plt.title('Purchase vs Gender')
plt.show()
```



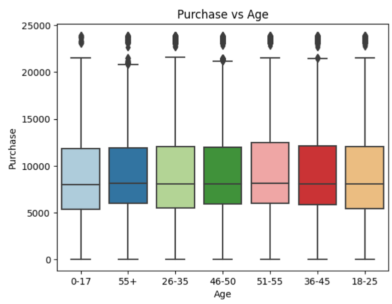
We can observe Males spend more than Females.

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

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

Pruchase habits based on Age

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



We can not see much difference in the median purchase values for different age groups.

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

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

The average order value is highest for age group 51-55 which is around 9534.

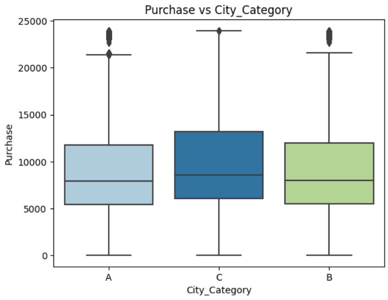
While, the average amount is lowest for age group 0-17 which is arouns 8933.

The highest order value for all the groups is around 23960.

The losest order value is 12 for all the groups.

Purchase habits based on city

```
sns.boxplot(data = df, y ='Purchase', x = 'City_Category', palette = 'Paired')
plt.title('Purchase vs City_Category')
plt.show()
```



City Category c has the highest median value followed by city B and city A.

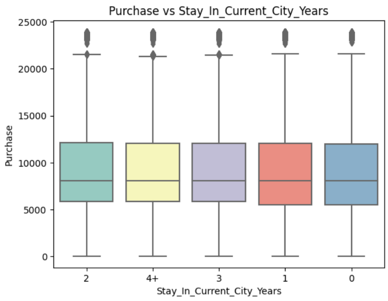
There are a few outliers fro city A and B.

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

City_Category	count	mean	std	min	25%	50%	75%	max
A	147720.0	8911.939216	4892.115238	12.0	5403.0	7931.0	11786.0	23961.0
B	231173.0	9151.300563	4955.496566	12.0	5460.0	8005.0	11986.0	23960.0
C	171175.0	9719.920993	5189.465121	12.0	6031.5	8585.0	13197.0	23961.0

Lets see if stay years of a person in a city affects customer's purchase habits or not.

```
sns.boxplot(data = df, y ='Purchase', x = 'Stay_In_Current_City_Years', palette = 'Set3')
plt.title('Purchase vs Stay_In_Current_City_Years')
plt.show()
```



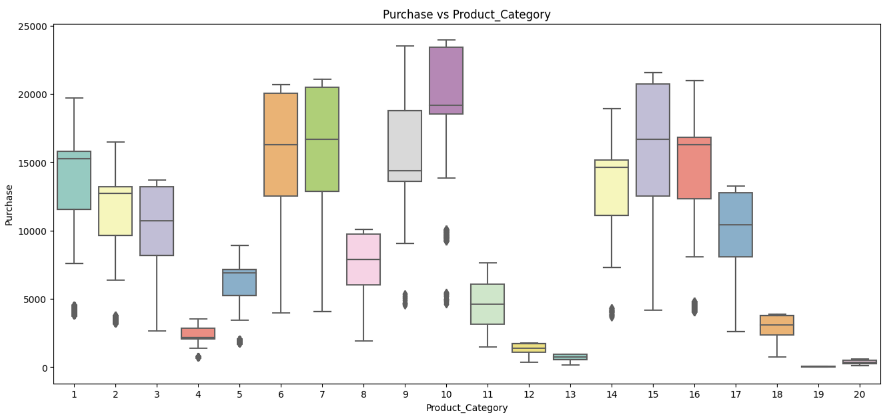
We can see that the median value is almost the same for all the years.

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

Stay_In_Current_City_Years	count	mean	std	min	25%	50%	75%	max
0	74398.0	9180.075123	4990.479940	12.0	5480.0	8025.0	11990.0	23960.0
1	193821.0	9250.145923	5027.476933	12.0	5500.0	8041.0	12042.0	23961.0
2	101838.0	9320.429810	5044.588224	12.0	5846.0	8072.0	12117.0	23961.0
3	95285.0	9286.904119	5020.343541	12.0	5832.0	8047.0	12075.0	23961.0
4+	84726.0	9275.598872	5017.627594	12.0	5844.0	8052.0	12038.0	23958.0

we can observe here is that the highest order value is also the same for all the years.

```
plt.figure(figsize = (16,7))
sns.boxplot(data = df, y ='Purchase', x = 'Product_Category', palette = 'Set3')
plt.title('Purchase vs Product_Category')
plt.show()
```




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

	count	mean	std	min	25%	50%	75%	max
Product_Category								
1	140378.0	13606.218596	4298.834894	3790.0	11546.00	15245.0	15812.00	19708.0
2	23864.0	11251.935384	3570.642713	3176.0	9645.75	12728.5	13212.00	16504.0
3	20213.0	10096.705734	2824.626957	2638.0	8198.00	10742.0	13211.00	13717.0
4	11753.0	2329.659491	812.540292	684.0	2058.00	2175.0	2837.00	3556.0
5	150933.0	6240.088178	1909.091687	1713.0	5242.00	6912.0	7156.00	8907.0
6	20466.0	15838.478550	4011.233690	3981.0	12505.00	16312.0	20051.00	20690.0
7	3721.0	16365.689600	4174.554105	4061.0	12848.00	16700.0	20486.00	21080.0
8	113925.0	7498.958078	2013.015062	1939.0	6036.00	7905.0	9722.00	10082.0
9	410.0	15537.375610	5330.847116	4528.0	13583.50	14388.5	18764.00	23531.0
10	5125.0	19675.570927	4225.721898	4624.0	18546.00	19197.0	23438.00	23961.0
11	24287.0	4685.268456	1834.901184	1472.0	3131.00	4611.0	6058.00	7654.0
12	3947.0	1350.859894	362.510258	342.0	1071.00	1401.0	1723.00	1778.0
13	5549.0	722.400613	183.493126	185.0	578.00	755.0	927.00	962.0

The median value for product category 10 is the highest which is 19197.

The median value for product category 19 is the lowest which is only 37.

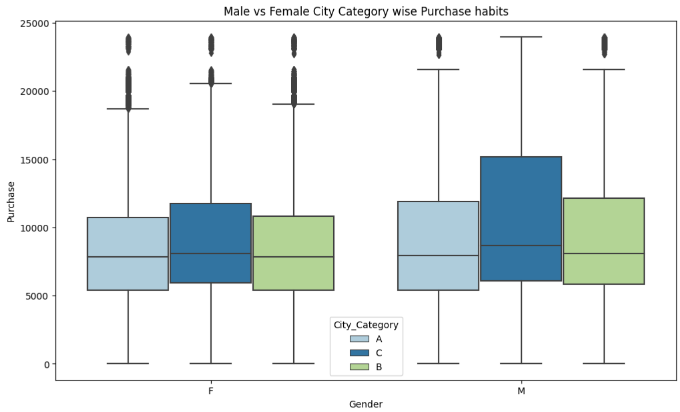
The average order value for category 10 is the highest which is 19675.

The average order value for category 19 is also the lowest which is 37.

Clearly, category 19 is the least preferred or least frequent bought product category.

Male vs Female City Category wise Purchase habits

```
plt.figure(figsize = (12,7))
sns.boxplot(data=df, y='Purchase', x='Gender', hue='City_Category',palette='Paired')

plt.title("Male vs Female City Category wise Purchase habits")
plt.show()
```



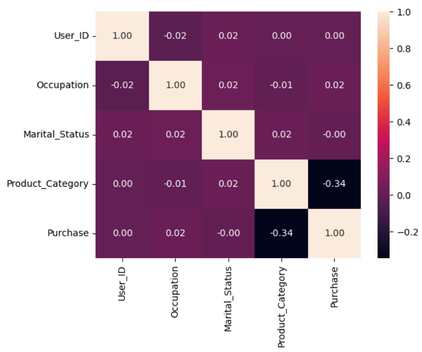
The median value for females in city category C is highest compared to city A and B.

The median value for males in city category C is also highest compared to city A and B.

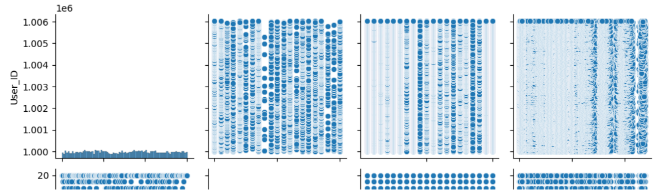
Correlation in the numerical values of the dataset.

As most the variables are object type so, co relation heatmap is irrelevant here,however we can chech the co relation before converting the data types

```
sns.heatmap(df_copy.corr(),annot=True,fmt='.2f')
plt.show()
```




```
sns.pairplot(df)
plt.show()
```



Sample Analysis Using Central Limit Theorem and Confidence Interval

CLT and CI analysis for Male customers Creating a Samples of size 1000 and computing means through bootstraping

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

```
9437.526040472265
```

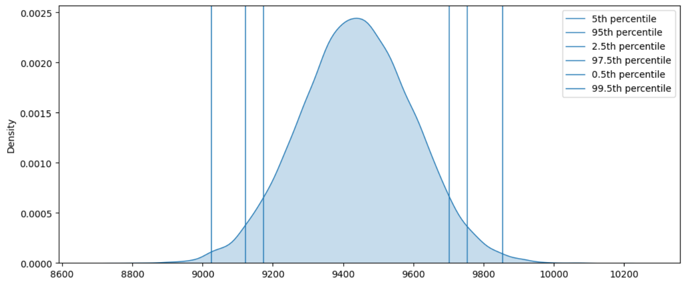
```
male_purchase_sample=[]
for i in range(20000):
    bootstraped_sample=np.random.choice(male_df,size=1000)
    bootstraped_mean=np.mean(bootstraped_sample)
    male_purchase_sample.append(bootstraped_mean)
```

Calculating CI at 90%,95% and 99% for sample of size 1000

```
CI_male_90=np.percentile(male_purchase_sample,[5,95])
CI_male_95=np.percentile(male_purchase_sample,[2.5,97.5])
CI_male_99=np.percentile(male_purchase_sample,[0.5,99.5])
print(f'CI at 90% for sample of size 1000 is {np.round(CI_male_90[0],2)} - {np.round(CI_male_90[1],2)}')
print(f'CI at 95% for sample of size 1000 is {np.round(CI_male_95[0],2)} - {np.round(CI_male_95[1],2)}')
print(f'CI at 99% for sample of size 1000 is {np.round(CI_male_99[0],2)} - {np.round(CI_male_99[1],2)}')
```

```
CI at 90% for sample of size 1000 is 9171.7 - 9701.3
CI at 95% for sample of size 1000 is 9121.64 - 9753.31
CI at 99% for sample of size 1000 is 9023.82 - 9852.79
```

```
# visulaizing CI for sample size of 1000 for male customer
plt.figure(figsize=(12,5))
sns.kdeplot(male_purchase_sample,fill=True)
plt.axvline(x=np.percentile(male_purchase_sample,[5]),ymin=0,ymax=1,linewidth=1.0,label='5th percentile')
plt.axvline(x=np.percentile(male_purchase_sample,[95]),ymin=0,ymax=1,linewidth=1.0,label='95th percentile')
plt.axvline(x=np.percentile(male_purchase_sample,[2.5]),ymin=0,ymax=1,linewidth=1.0,label='2.5th percentile')
plt.axvline(x=np.percentile(male_purchase_sample,[97.5]),ymin=0,ymax=1,linewidth=1.0,label='97.5th percentile')
plt.axvline(x=np.percentile(male_purchase_sample,[0.5]),ymin=0,ymax=1,linewidth=1.0,label='0.5th percentile')
plt.axvline(x=np.percentile(male_purchase_sample,[99.5]),ymin=0,ymax=1,linewidth=1.0,label='99.5th percentile')
plt.legend()
plt.show()
```



CLT and CI analysis for Feale customers Creating a Samples of size 1000 and computing means through bootstraping

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

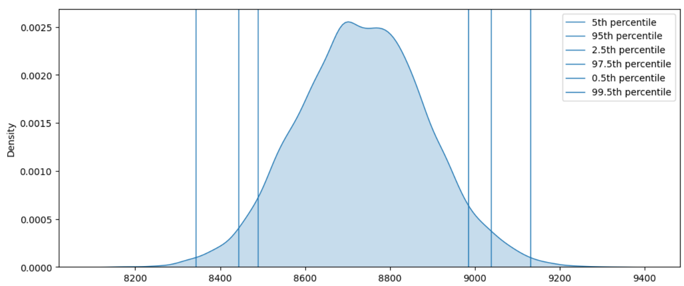
```
8734.565765155476
```

```
female_purchase_sample=[]
for i in range(10000):
    bootstraped_sample=np.random.choice(female_df,size=1000)
    bootstraped_mean=np.mean(bootstraped_sample)
    female_purchase_sample.append(bootstraped_mean)
```

```
CI_female_90_1000=np.percentile(female_purchase_sample,[5,95])
CI_female_95_1000=np.percentile(female_purchase_sample,[2.5,97.5])
CI_female_99_1000=np.percentile(female_purchase_sample,[0.5,99.5])
print(f'CI at 90% for sample of size 1000 is {np.round(CI_female_90_1000[0],2)} - {np.round(CI_female_90_1000[1],2)}')
print(f'CI at 95% for sample of size 1000 is {np.round(CI_female_95_1000[0],2)} - {np.round(CI_female_95_1000[1],2)}')
print(f'CI at 99% for sample of size 1000 is {np.round(CI_female_99_1000[0],2)} - {np.round(CI_female_99_1000[1],2)}')
```

```
CI at 90% for sample of size 1000 is 8488.29 - 8984.24
CI at 95% for sample of size 1000 is 8443.11 - 9037.63
CI at 99% for sample of size 1000 is 8341.74 - 9130.8
```

```
plt.figure(figsize=(12,5))
sns.kdeplot(female_purchase_sample,fill=True)
plt.axvline(x=np.percentile(female_purchase_sample,[5]),ymin=0,ymax=1,linewidth=1.0,label='5th percentile')
plt.axvline(x=np.percentile(female_purchase_sample,[95]),ymin=0,ymax=1,linewidth=1.0,label='95th percentile')
plt.axvline(x=np.percentile(female_purchase_sample,[2.5]),ymin=0,ymax=1,linewidth=1.0,label='2.5th percentile')
plt.axvline(x=np.percentile(female_purchase_sample,[97.5]),ymin=0,ymax=1,linewidth=1.0,label='97.5th percentile')
plt.axvline(x=np.percentile(female_purchase_sample,[0.5]),ymin=0,ymax=1,linewidth=1.0,label='0.5th percentile')
plt.axvline(x=np.percentile(female_purchase_sample,[99.5]),ymin=0,ymax=1,linewidth=1.0,label='99.5th percentile')
plt.legend()
plt.show()
```



Calculating the standard error for male and female sample

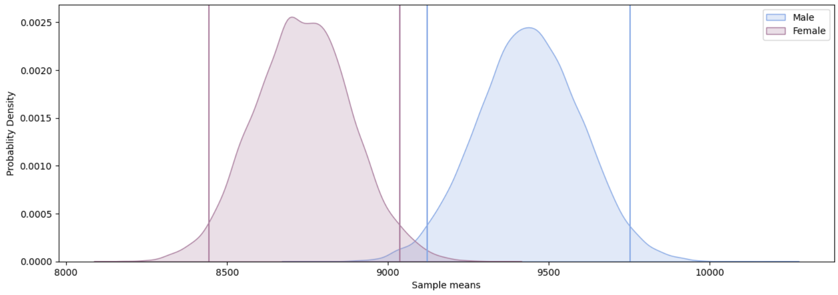
```
SE_female_1000 = (female_df.std()/np.sqrt(1000))
print(f'Standard error for female for sample size of 1000: {np.round(SE_female_1000,2)}')
```

```
Standard error for female for sample size of 1000: 150.75
```

```
SE_male_1000 = (male_df.std()/np.sqrt(1000))
print(f'Standard error for male for sample size of 1000: {np.round(SE_male_1000,2)}')
```

```
Standard error for male for sample size of 1000: 161.03
```

```
plt.figure(figsize=(15,5))

sns.kdeplot(male_purchase_sample,color='#89AAE6',fill=True,label='Male')
sns.kdeplot(female_purchase_sample,color='#AC80A0',fill=True,label='Female')
plt.axvline(np.percentile(male_purchase_sample,[2.5]),0,1,color='#89AAE6')
plt.axvline(np.percentile(male_purchase_sample,[97.5]),0,1,color='#89AAE6')
plt.axvline(np.percentile(female_purchase_sample,[2.5]),0,1,color='#AC80A0')
plt.axvline(np.percentile(female_purchase_sample,[97.5]),0,1,color='#AC80A0')
plt.xlabel('Sample means')
plt.ylabel('Probablity Density')
plt.legend()
plt.show()
```



Confidence intervals at 95 % for male and female customers does not overlap.

With a 95% confidence level, the confidence interval for male customers is consistently both higher and wider than the confidence interval for female customers for a sample size of 1000. This statistically indicates that male customers tend to spend more money per transaction than female customers.

CI and CLT analysis for Married Customers

Creating a Samples of size 1000 and computing means through bootstraping

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

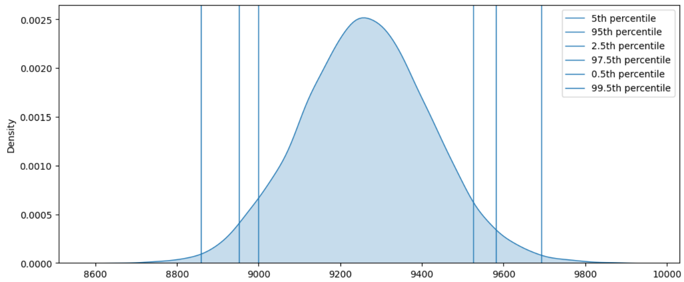
```
9261.174574082374
```

```
married_mean=[]
for i in range(10000):
    bootstraped_sample=np.random.choice(married_df,size=1000)
    bootstraped_mean=np.mean(bootstraped_sample)
    married_mean.append(bootstraped_mean)
```

```
CI_married_90_1000=np.percentile(married_mean,[5,95])
CI_married_95_1000=np.percentile(married_mean,[2.5,97.5])
CI_married_99_1000=np.percentile(married_mean,[0.5,99.5])
print(f'CI at 90% for sample of size 1000: {np.round(CI_married_90_1000[0],2)} - {np.round(CI_married_90_1000[1],2)}')
print(f'CI at 95% for sample of size 1000: {np.round(CI_married_95_1000[0],2)} - {np.round(CI_married_95_1000[1],2)}')
print(f'CI at 99% for sample of size 1000: {np.round(CI_married_99_1000[0],2)} - {np.round(CI_married_99_1000[1],2)}')
```

```
CI at 90% for sample of size 1000: 8999.09 - 9526.56
CI at 95% for sample of size 1000: 8952.46 - 9581.68
CI at 99% for sample of size 1000: 8858.16 - 9692.82
```

```
plt.figure(figsize=(12,5))
sns.kdeplot(married_mean,fill=True)
plt.axvline(x=np.percentile(married_mean,[5]),ymin=0,ymax=1,linewidth=1.0,label='5th percentile')
plt.axvline(x=np.percentile(married_mean,[95]),ymin=0,ymax=1,linewidth=1.0,label='95th percentile')
plt.axvline(x=np.percentile(married_mean,[2.5]),ymin=0,ymax=1,linewidth=1.0,label='2.5th percentile')
plt.axvline(x=np.percentile(married_mean,[97.5]),ymin=0,ymax=1,linewidth=1.0,label='97.5th percentile')
plt.axvline(x=np.percentile(married_mean,[0.5]),ymin=0,ymax=1,linewidth=1.0,label='0.5th percentile')
plt.axvline(x=np.percentile(married_mean,[99.5]),ymin=0,ymax=1,linewidth=1.0,label='99.5th percentile')
plt.legend()
plt.show()
```




```
# Standard error using formulla --->(population standard deviation/sqrt(sample size))

SE_married_1000 = (married_df.std()/np.sqrt(1000))

print(f'Standard error for sample size of 1000: {np.round(SE_married_1000,2)}')
```

```
Standard error for sample size of 1000: 158.65
```

CI and CLT analysis for unmarried Customers

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

```
9265.907618921507
```

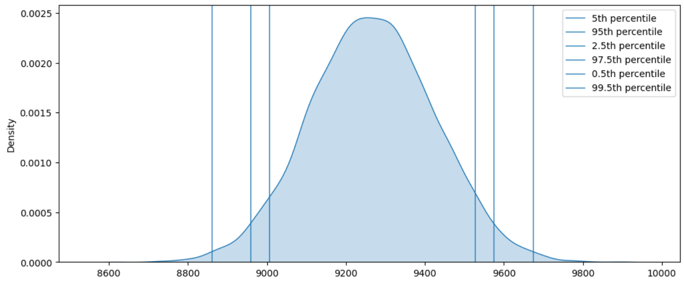
```
unmarried_purchase_mean_1000=[]
for i in range(10000):
    bootstraped_sample=np.random.choice(unmarried_df,size=1000)
    bootstraped_mean=np.mean(bootstraped_sample)
    unmarried_purchase_mean_1000.append(bootstraped_mean)
```

```
CI_unmarried_90_1000=np.percentile(unmarried_purchase_mean_1000,[5,95])
CI_unmarried_95_1000=np.percentile(unmarried_purchase_mean_1000,[2.5,97.5])
CI_unmarried_99_1000=np.percentile(unmarried_purchase_mean_1000,[0.5,99.5])
print(f'CI at 90% for sample of size 1000: {np.round(CI_unmarried_90_1000[0],2)} - {np.round(CI_unmarried_90_1000[1],2)}')
print(f'CI at 95% for sample of size 1000: {np.round(CI_unmarried_95_1000[0],2)} - {np.round(CI_unmarried_95_1000[1],2)}')
print(f'CI at 99% for sample of size 1000: {np.round(CI_unmarried_99_1000[0],2)} - {np.round(CI_unmarried_99_1000[1],2)}')
```

```
CI at 90% for sample of size 1000: 9006.21 - 9527.15
CI at 95% for sample of size 1000: 8958.85 - 9573.94
CI at 99% for sample of size 1000: 8861.08 - 9673.89
```

```
plt.figure(figsize=(12,5))
sns.kdeplot(unmarried_purchase_mean_1000,fill=True)
plt.axvline(x=np.percentile(unmarried_purchase_mean_1000,[5]),ymin=0,ymax=1,linewidth=1.0,label='5th percentile')
plt.axvline(x=np.percentile(unmarried_purchase_mean_1000,[95]),ymin=0,ymax=1,linewidth=1.0,label='95th percentile')
plt.axvline(x=np.percentile(unmarried_purchase_mean_1000,[2.5]),ymin=0,ymax=1,linewidth=1.0,label='2.5th percentile')
plt.axvline(x=np.percentile(unmarried_purchase_mean_1000,[97.5]),ymin=0,ymax=1,linewidth=1.0,label='97.5th percentile')
plt.axvline(x=np.percentile(unmarried_purchase_mean_1000,[0.5]),ymin=0,ymax=1,linewidth=1.0,label='0.5th percentile')
plt.axvline(x=np.percentile(unmarried_purchase_mean_1000,[99.5]),ymin=0,ymax=1,linewidth=1.0,label='99.5th percentile')
```

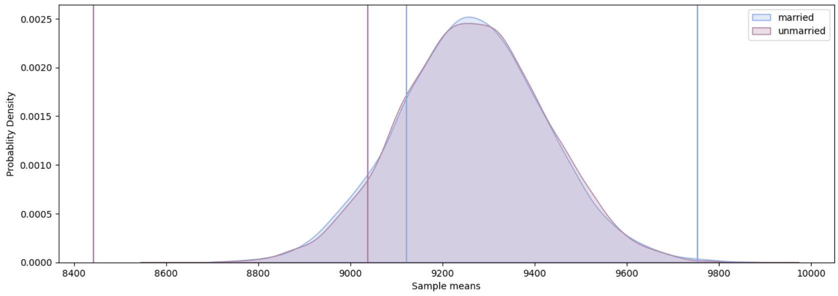
```
plt.legend()
plt.show()
```




```
# Standard error using formulla --->(population standard deviation/sqrt(sample size))

SE_unmarried_1000 = (unmarried_df.std()/np.sqrt(1000))

print(f'Standard error for sample size of 1000: {np.round(SE_unmarried_1000,2)}')
```

```
Standard error for sample size of 1000: 158.98
```

```
plt.figure(figsize=(15,5))

sns.kdeplot(married_mean,color='#89AAE6',fill=True,label='married')
sns.kdeplot(unmarried_purchase_mean_1000,color='#AC80A0',fill=True,label='unmarried')
plt.axvline(np.percentile(male_purchase_sample,[2.5]),0,1,color='#89AAE6')
plt.axvline(np.percentile(male_purchase_sample,[97.5]),0,1,color='#89AAE6')
plt.axvline(np.percentile(female_purchase_sample,[2.5]),0,1,color='#AC80A0')
plt.axvline(np.percentile(female_purchase_sample,[97.5]),0,1,color='#AC80A0')
plt.xlabel('Sample means')
plt.ylabel('Probablity Density')
plt.legend()
plt.show()
```



Inference:

The confidence intervals for married and unmarried customers overlap, suggesting that both male and female customers spend a similar amount per transaction. This means that the spending behavior of married and unmarried customers is alike.

CI analysis for age-group

Creating a Samples of size 1000 and computing means through bootstraping

```
youth_df=df.loc[df['Age']=='0-17']['Purchase']
youth_df.mean()
```

```
8933.464640444974
```

```
youth_purchase_mean=[]
for i in range(10000):
    bootstraped_sample=np.random.choice(youth_df,size=1000)
    bootstraped_mean=np.mean(bootstraped_sample)
    youth_purchase_mean.append(bootstraped_mean)
```

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

```
9169.663606261289
```

```
young_purchase_mean=[]
for i in range(10000):
    bootstraped_sample=np.random.choice(young_df,size=1000)
    bootstraped_mean=np.mean(bootstraped_sample)
    young_purchase_mean.append(bootstraped_mean)
```

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

```
9252.690632869888
```

```
adult_purchase_mean=[]
for i in range(10000):
    bootstraped_sample=np.random.choice(adult_df,size=1000)
    bootstraped_mean=np.mean(bootstraped_sample)
    adult_purchase_mean.append(bootstraped_mean)
```

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

```
9331.350694917874
```

```
midage_purchase_mean=[]
for i in range(10000):
    bootstraped_sample=np.random.choice(midage_df,size=1000)
    bootstraped_mean=np.mean(bootstraped_sample)
    midage_purchase_mean.append(bootstraped_mean)
```

```
midlife_df=df.loc[df['Age']=='46-50']['Purchase']
midlife_df.mean()
```

```
9208.625697468327
```

```
midlife_purchase_mean=[]
for i in range(10000):
    bootstraped_sample=np.random.choice(midlife_df,size=1000)
    bootstraped_mean=np.mean(bootstraped_sample)
    midlife_purchase_mean.append(bootstraped_mean)
```

```
old_df=df.loc[df['Age']=='51-55']['Purchase']
old_df.mean()
```

```
9534.808030960236
```

```
old_purchase_mean=[]
for i in range(10000):
    bootstraped_sample=np.random.choice(old_df,size=1000)
```

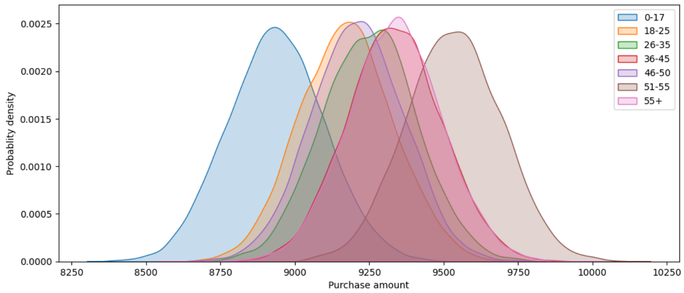
```
    bootstraped_sample=np.random.choice(old_df,size=1000)
    bootstraped_mean=np.mean(bootstraped_sample)
    old_purchase_mean.append(bootstraped_mean)
```

```
senior_df=df.loc[df['Age']=='55+']['Purchase']
senior_df.mean()
```

```
9336.280459449405
```

```
senior_purchase_mean=[]
for i in range(10000):
    bootstraped_sample=np.random.choice(senior_df,size=1000)
    bootstraped_mean=np.mean(bootstraped_sample)
    senior_purchase_mean.append(bootstraped_mean)
```

```
sample_means=[youth_purchase_mean,young_purchase_mean,adult_purchase_mean,midage_purchase_mean,midlife_purchase_mean,old_purchase_mean,senior_purchase_mean]
labels=['0-17','18-25','26-35','36-45','46-50','51-55','55+']
plt.figure(figsize=(12,5))
for i in range(len(labels)):
    sns.kdeplot(sample_means[i],fill=True,label=labels[i])
plt.xlabel('Purchase amount')
plt.ylabel('Probablity density')
plt.legend()
plt.show()
```



Inference: The majority of age groups' purchasing behaviors exhibit overlapping patterns, with the exception of the (0-17) and (51-55) age categories.

Business Insights

• So,75.31% customers are male, 24.69 % are female

• The confidence interval for male purchases consistently exhibits both a higher upper bound and a wider spread in comparison to the confidence interval for female purchases.

• Product ID P00265242 is the most frequent sold item.

• Most people are in the current city since 1 year.

• We can observe that most buyers are in within the age of 18-45 before and after this range we can see less buyers.

• We can observe that 59% of the frequent buyers are of unmarried people, while 41% of married.

• category 19 is the least preferred or least frequent bought product category.

• The confidence intervals for married and unmarried customers overlap, suggesting that both male and female customers spend a similar amount per transaction.

• The (0-17) and (51-55) age categories spends lowest and highest amount on avarage

Now answering the business problems

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

answer-

No,we can see CI's of male and female do not overlap and upper limits of female purchase CI are lesser than lower limits of male purchase CI. This proves that men usually spend more than women

As per data 75.3% purchases are from male and only 24.7% purchases are from female

The confidence interval for male purchases is consistently both higher and wider than the confidence interval for female purchases. This statistically indicates that male customers tend to spend more money per transaction than female customers.

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

answer- For females CI at 90% for sample of size 1000 is 8488.29 - 8984.24 CI at 95% for sample of size 1000 is 8443.11 - 9037.63 CI at 99% for sample of size 1000 is 8341.74 - 9130.8

For males CI at 90% for sample of size 1000 is 9171.7 - 9701.3 CI at 95% for sample of size 1000 is 9121.64 - 9753.31 CI at 99% for sample of size 1000 is 9023.82 - 9852.79

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

answer- As we discussed in the data exploration

Results when the same activity is performed for Married vs Unmarried answer- As we discussed in the data exploration

Business Recommendations

- Both type of gender customer spends similar amount per order but we can see male customers order more than female customer ,so action is needed to increase female customers (like relevant ads)
- Most of the custmers are younger in age but spends comparatively lower than older age people so more emi options can increase their spendings
- Male customers living in City_Category C spend more money than other male customers living in B or C, Selling more products in the City_Category C will help the company increase the revenue.
- Unmarried customers spend more money than married customers, So company should focus on acquisition of Unmarried customers