Business case: Walmart Confidence interval and CLT

Introduction

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

Business problem

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).

In [1]:
```
# importing libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statistics
from scipy.stats import norm
from scipy.stats import kstest
import statsmodels.api as sm
```

Importing the dataset and reading dataset using Pandas

In [2]:
```
df=pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000
df
```

Out[2]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Ye
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	
...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	
550064	1006035	P00375436	F	26-35	1	C	
550065	1006036	P00375436	F	26-35	15	B	
550066	1006038	P00375436	F	55+	1	C	
550067	1006039	P00371644	F	46-50	0	B	

550068 rows × 10 columns

Analyzing Basic metrics

Purpose of analysing basic metrics is to understand the shape of the dataset. Data types, features and unique values of the dataset

In [3]:
```
df.shape
```

Out[3]:
```
(550068, 10)
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64 
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

In [5]:
```
#checking for duplicate records
df.duplicated().sum()
```

Out[5]:
```
0
```

Inference: No duplicate records found

Unique values(counts) for each record

In [6]:
```
df.nunique()
```

Out[6]:
```
User_ID                      5891
Product_ID                   3631
Gender                          2
Age                             7
Occupation                     21
City_Category                   3
Stay_In_Current_City_Years      5
Marital_Status                  2
Product_Category               20
Purchase                    18105
dtype: int64
```

Inference: Total number of records exceeds far more than the userIDs, it may suggest that the customers have visited multiple times in order to buy the products.

Statistical summary

In [7]:
```
df.describe()
```

Out[7]:

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

In [8]:
```
df.describe(include='all')
```

Out[8]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Curre
count	5.500680e+05	550068	550068	550068	550068.000000	550068	
unique	NaN	3631	2	7	NaN	3	
top	NaN	P00265242	M	26-35	NaN	B	
freq	NaN	1880	414259	219587	NaN	231173	
mean	1.003029e+06	NaN	NaN	NaN	8.076707	NaN	
std	1.727592e+03	NaN	NaN	NaN	6.522660	NaN	
min	1.000001e+06	NaN	NaN	NaN	0.000000	NaN	
25%	1.001516e+06	NaN	NaN	NaN	2.000000	NaN	
50%	1.003077e+06	NaN	NaN	NaN	7.000000	NaN	
75%	1.004478e+06	NaN	NaN	NaN	14.000000	NaN	
max	1.006040e+06	NaN	NaN	NaN	20.000000	NaN	

Insights

- Range of purchase amount is Dollars 12 to 23961.
- Mean purchase amount is 9263.968 Dollars.
- Median purchase amount is 8047 Dollars.
- Standard deviation of purchase amount is 5023 Dollars.
- Interquartile range of purchase amount is 5823 to 12054 Dollars.
- Product ID P00265242 is on top

In [9]:
```
# Finding number of null values
df.isna().sum()
```

Out[9]:
```
User_ID                     0
Product_ID                  0
Gender                      0
Age                         0
Occupation                  0
City_Category               0
Stay_In_Current_City_Years  0
Marital_Status              0
Product_Category            0
Purchase                    0
dtype: int64
```

Inference: There are no null values in the dataset

Data Exploration

In [10]:
```
df.groupby(['Gender'])['User_ID'].nunique()
```

Out[10]:
```
Gender
F    1666
M    4225
Name: User_ID, dtype: int64
```

In [11]:
```
# Proportion of males and females
print('Females are',1666/5891)
print('Males are',4225/5891)
```

```
Females are 0.2828042777117637
Males are 0.7171957222882362
```

Inference

72% of the customers are male and 28% are Females

Purchase according to Gender

In [12]:
```
df.groupby('Gender')['Purchase'].describe()
```

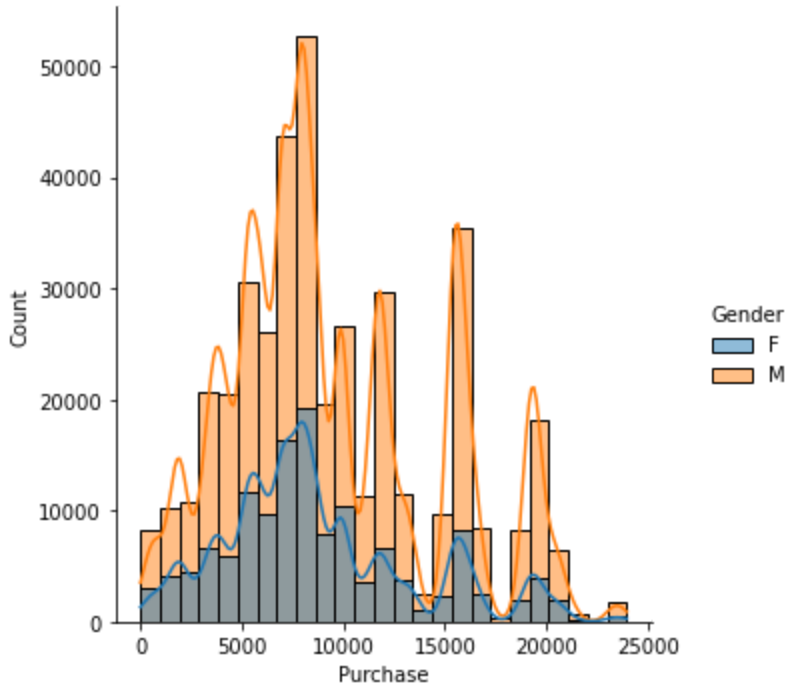
Out[12]:

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

Observation: Average purchasing amount by females is 8734.57 Dollars whereas average purchase amount by males is 9437.53 Dollars

Amount spent per transaction by Gender

In [13]:
```
sns.displot(x='Purchase',bins=25,kde=True, hue= 'Gender',data=df)
plt.show()
```



Observations:

- Majority of customers purchase within the 5,000-20,000 range.
- Maximum purchase amount spent by more than 50,000 males is about 8000 Dollars
- Maximum purchase amount spent by more than 20,000 females is about 8000 Dollars

In [14]:
```
# Grouping the data by age

df.groupby(['Age']) ['User_ID'].nunique()
```

Out[14]:
```
Age
0-17      218
18-25    1069
26-35    2053
36-45    1167
46-50     531
51-55     481
55+       372
Name: User_ID, dtype: int64
```

Observations: The age group of customers in between 26-35 yrs is maximum, followed by 36-45 yrs

In [15]:
```
df.groupby(['City_Category'])['User_ID'].nunique()
```

Out[15]:
```
City_Category
A    1045
B    1707
C    3139
Name: User_ID, dtype: int64
```

Observation: There are 3 categories of cities

In [16]:
```
df.groupby(['Marital_Status'])['User_ID'].nunique()
```

Out[16]:
```
Marital_Status
0    3417
1    2474
Name: User_ID, dtype: int64
```

Observation:Unmarried are 3417 and married are 2474 from the available data. Inference: There are more unmarried people than married people in the customers who made purchases.

In [17]:
```
df.groupby(['Stay_In_Current_City_Years'])['User_ID'].nunique()
```

Out[17]:
```
Stay_In_Current_City_Years
0      772
1     2086
2     1145
3      979
4+     909
Name: User_ID, dtype: int64
```

Distribution based on different categories

In [18]:
```
categorical_col = ['Gender','Age','City_Category','Stay_In_Current_City_Years
df[categorical_col].melt().groupby(['variable','value'])[['value']].count()*1(
```

Out[18]:

		value
variable	value	
Age	0-17	2.745479
	18-25	18.117760
	26-35	39.919974
	36-45	19.999891
	46-50	8.308246
	51-55	6.999316
	55+	3.909335
City_Category	A	26.854862
	B	42.026259
	C	31.118880
Gender	F	24.689493
	M	75.310507
Marital_Status	0	59.034701
	1	40.965299
Occupation	0	12.659889
	1	8.621843
	2	4.833584
	3	3.208694
	4	13.145284
	5	2.213726
	6	3.700452
	7	10.750125
	8	0.281056
	9	1.143677
	10	2.350618
	11	2.106285
	12	5.668208
	13	1.404917
	14	4.964659
	15	2.211545
	16	4.612339
	17	7.279645
	18	1.203851
	19	1.538173
	20	6.101427

variable	value	value
Product_Category	**1**	25.520118
	2	4.338373
	3	3.674637
	4	2.136645
	5	27.438971
	6	3.720631
	7	0.676462
	8	20.711076
	9	0.074536
	10	0.931703
	11	4.415272
	12	0.717548
	13	1.008784
	14	0.276875
	15	1.143495
	16	1.786688
	17	0.105078
	18	0.568112
	19	0.291419
	20	0.463579
Stay_In_Current_City_Years	**0**	13.525237
	1	35.235825
	2	18.513711
	3	17.322404
	4+	15.402823

Inferences:

- There are more single people than married people.
- The majority of the customers are from city B but amount soent by people from city C is more.
- Male customers tend to spend more than female customers, as the mean is more for male customers.
- Product categories 1,5 and 8 are sold more.

In [19]:
```
df.describe().T
```

Out[19]:

	count	mean	std	min	25%	50%	
User_ID	550068.0	1.003029e+06	1727.591586	1000001.0	1001516.0	1003077.0	10044
Occupation	550068.0	8.076707e+00	6.522660	0.0	2.0	7.0	
Marital_Status	550068.0	4.096530e-01	0.491770	0.0	0.0	0.0	
Product_Category	550068.0	5.404270e+00	3.936211	1.0	1.0	5.0	
Purchase	550068.0	9.263969e+03	5023.065394	12.0	5823.0	8047.0	120

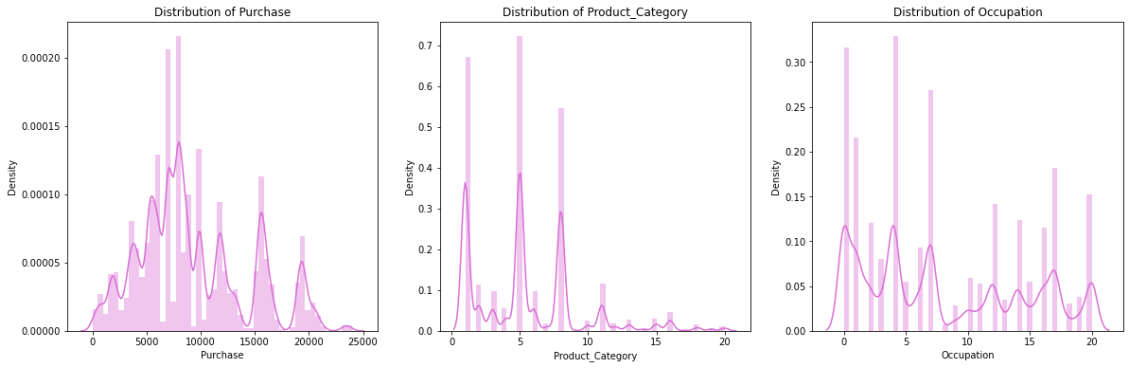
Observation: Mean purchase amount is 8047 Dollars.

Visual Analysis - Univariate and Bivariate

In [20]:
```
# Creating distribution plots for some features
temp = ['Purchase', 'Product_Category', 'Occupation']
plt.figure(figsize=(20,6))
for i in range(len(temp)):
    plt.subplot(1,3,i+1)
    sns.distplot(df[temp[i]], color='orchid')
    plt.title('Distribution of {feature}'.format(feature = temp[i]))

plt.show()
```

```
unction for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figur
e-level function with similar flexibility) or `histplot` (an axes-level f
unction for histograms).
  warnings.warn(msg, FutureWarning)
```



Observations:

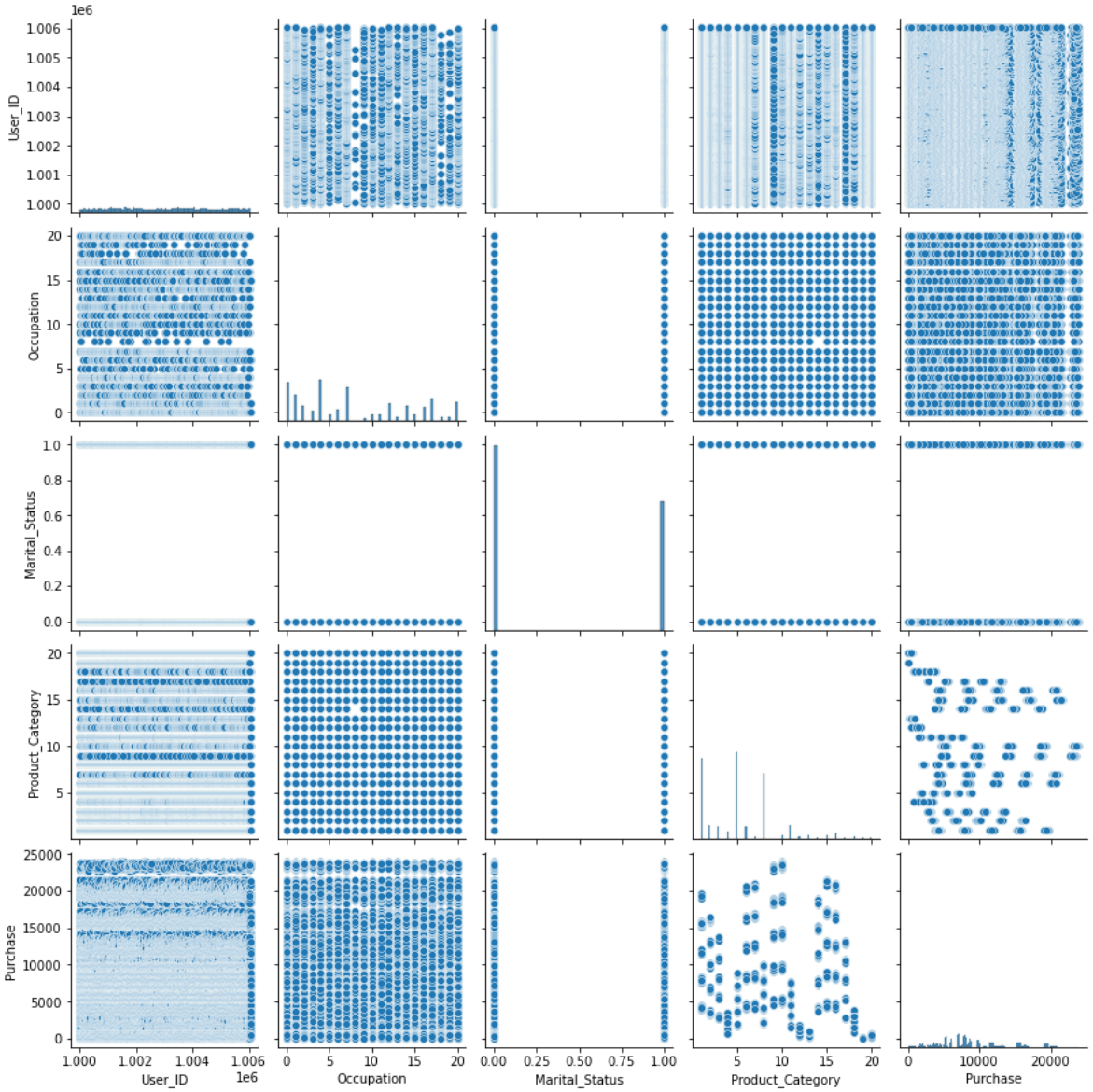
- Purchase amounts of 8000 to 10000 Dollars are more.

- Product categories 1,5, 8 are most frequently purchased.
- There are 20 product categories on sale and some of them are least purchased.

Checking the correlation between different variables

In [21]:
```
sns.pairplot(df)
```

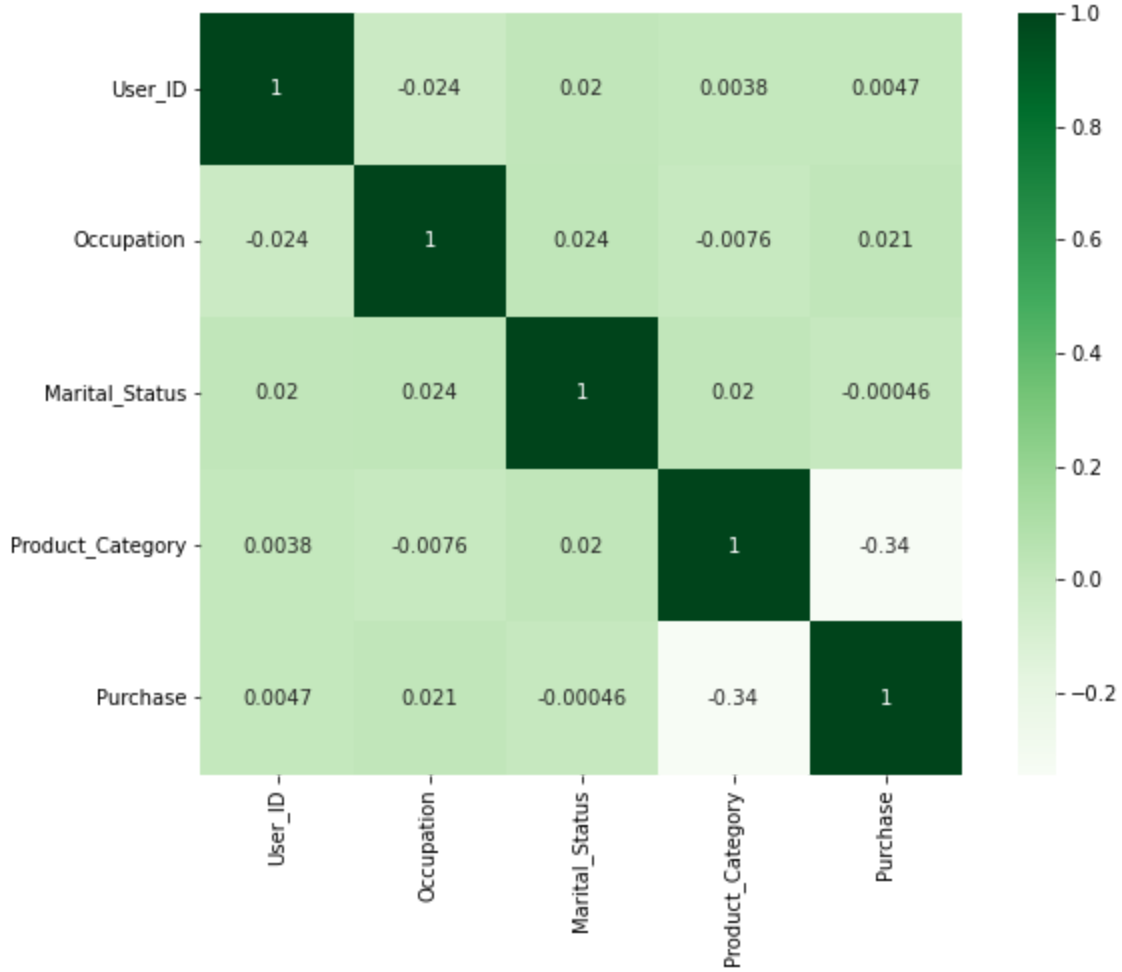
Out[21]:
```
<seaborn.axisgrid.PairGrid at 0x19957c4fdf0>
```



Inference: Mostly features are categorical and not much correlation can be observed from the above graphs

Correlation analysis using Heat map

In [22]:
```
plt.figure(figsize =(10,7))
ax = sns.heatmap(df.corr(),annot = True,cmap='Greens',square=True)
```



Observations:

- We don't see a strong correlation of purchase with any feature.
- There is weak correlation between product category and purchase amount.

In [23]:
```
# Finding Pearson Correlation coefficient
df.corr().round(2)
```

Out[23]:

	User_ID	Occupation	Marital_Status	Product_Category	Purchase
User_ID	1.00	-0.02	0.02	0.00	0.00
Occupation	-0.02	1.00	0.02	-0.01	0.02
Marital_Status	0.02	0.02	1.00	0.02	-0.00
Product_Category	0.00	-0.01	0.02	1.00	-0.34
Purchase	0.00	0.02	-0.00	-0.34	1.00

Univariate Analysis

Box plot for Outlier detection

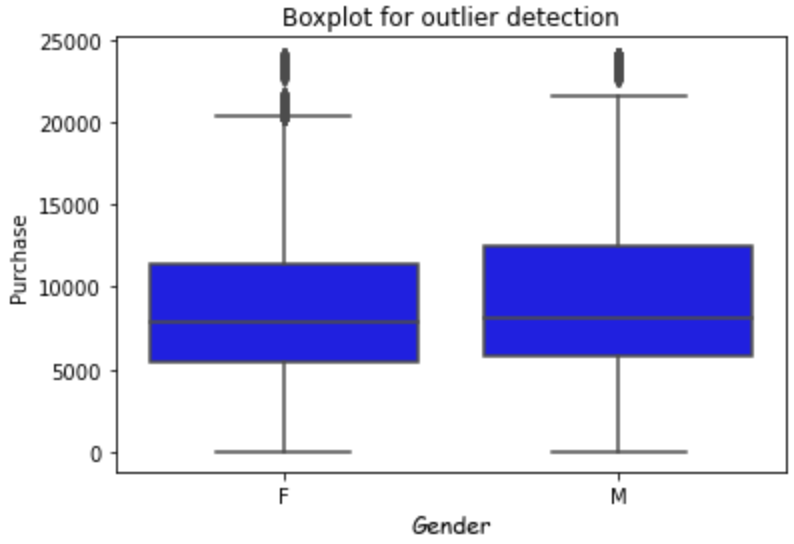
In [24]:
```
df.columns
```

Out[24]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Categor
y',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

In [25]:
```
# Visualising dependent variables for Outlier detection

sns.boxplot(x='Gender', y ='Purchase',data=df,color='blue')
plt.title("Boxplot for outlier detection",fontsize=12)
plt.xlabel('Gender',fontsize=12,family='Comic Sans MS')
```

Out[25]:
```
Text(0.5, 0, 'Gender')
```



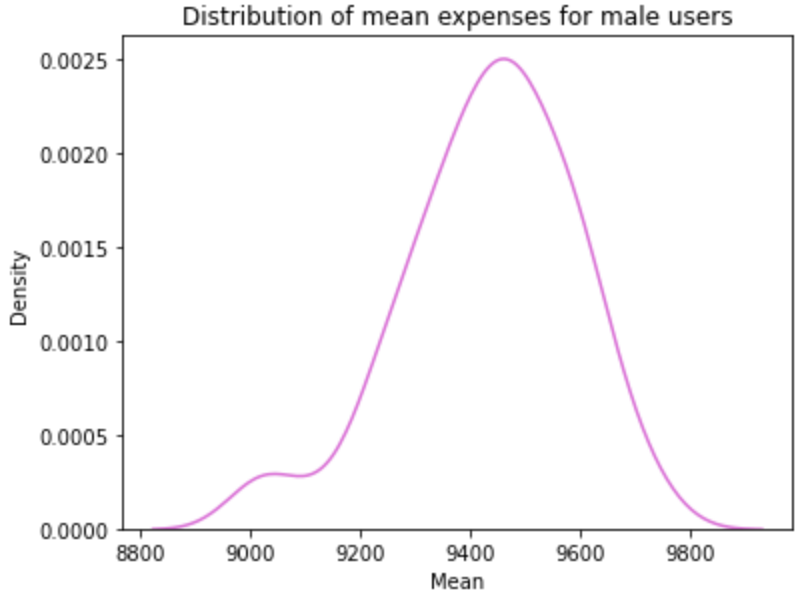
In [26]:
```
#### Outlier detection using IQR method
for i in ['Purchase']:
    outliers = []
    p25 = np.percentile(df[i],25)
    p75 = np.percentile(df[i], 75)
    iqr = p75-p25
    max_val = p75+iqr*1.5
    min_val = p25-iqr*1.5
    outliers = df.loc[(df[i]<min_val) | (df[i]>max_val),i]
    print('Outliers for the column',i,'-')
    print(outliers)
    print('Number of outliers-',len(outliers))
    print('Percentage of outliers =', round((len(outliers)/len(df[i]))*100,2)
```

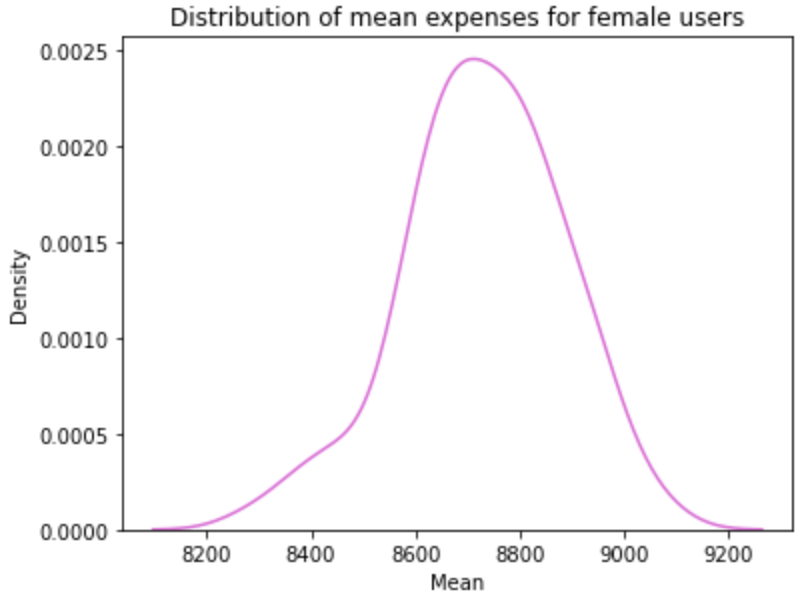
```
Outliers for the column Purchase -
343       23603
375       23792
652       23233
736       23595
1041      23341
          ...
544488    23753
544704    23724
544743    23529
545663    23663
545787    23496
Name: Purchase, Length: 2677, dtype: int64
Number of outliers- 2677
Percentage of outliers = 0.49 %
```

Observations: NUmber of outliers is 2677 and percentage of outliers is 0.49%

Analysis of data using Central limit theorem

In [27]:
```
# Taking samples of 1000 entries for both genders and
# Creating kde plots to check if it appears gaussian.
plt.figure(figsize=(6,10))
x = 1
for j in ['M','F']:
    means = []
    for i in range(100):
        temp = df.loc[df['Gender']==j,'Purchase'].sample(1000)
        avg = temp.mean()
        means.append(avg)
    plt.subplot(2,1,x)
    sns.kdeplot(x = means, color = 'orchid')
    if j == 'M':
        gen = 'male'
        means_m = means
    else:
        gen = 'female'
        means_f = means
    plt.title('Distribution of mean expenses for {g} users'.format(g = gen),
    plt.xlabel('Mean')
    x += 1
plt.show()
```


Distribution of mean expenses for male users
Density
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
8800
9000
9200
9400
9600
9800
Mean


Distribution of mean expenses for female users
Density
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
8200
8400
8600
8800
9000
9200
Mean

In [28]:
```
# Finding different confidence intervals for males and females
for i in ['males', 'females']:
    print('For {g}-'.format(g = i))
    if i == 'males':
        means = means_m
        gen = 'M'
    else:
        means = means_f
        gen = 'F'
    print('Mean of sample means =',np.mean(means))
    print('Population mean =', np.mean(df.loc[df['Gender']==gen, 'Purchase']))
    print('Standard deviation of means (Standard Error) =', np.std(means))
    print('Standard deviation of population =',df.loc[df['Gender']==gen, 'Pur
    print('99% CONFIDENCE INTERVAL for mean expense by {g} users-'.format(g =
    print((np.percentile(means, 0.5).round(2), np.percentile(means, 99.5).rou
    print('95% CONFIDENCE INTERVAL for mean expense by {g} users-'.format(g =
    print((np.percentile(means, 2.5).round(2), np.percentile(means, 97.5).rou
    print('90% CONFIDENCE INTERVAL for mean expense by {g} users-'.format(g =
    print((np.percentile(means, 5).round(2), np.percentile(means, 95).round(2
    print('-'*50)
```

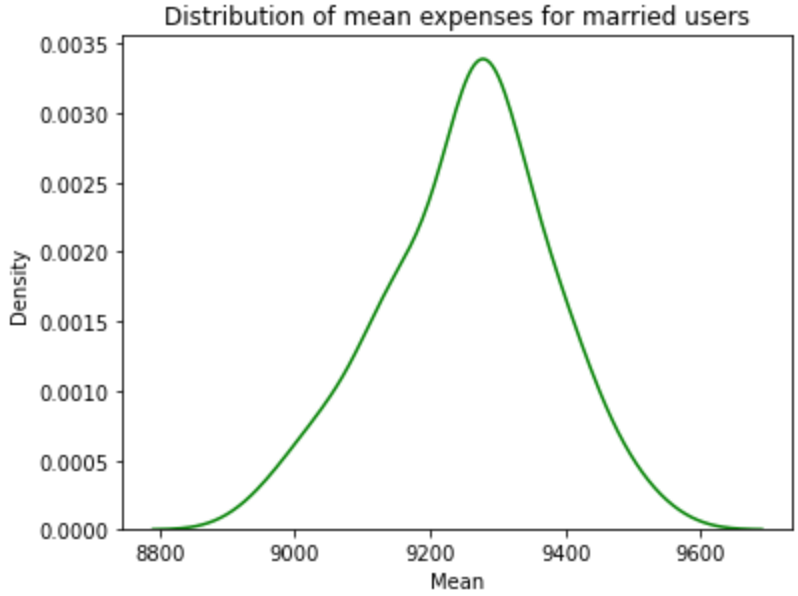
```
For males-
Mean of sample means = 9432.696380000001
Population mean = 9437.526040472265
Standard deviation of means (Standard Error) = 155.1187905149973
Standard deviation of population = 5092.186209777949
99% CONFIDENCE INTERVAL for mean expense by males users-
(9011.42, 9732.86)
95% CONFIDENCE INTERVAL for mean expense by males users-
(9033.8, 9679.1)
90% CONFIDENCE INTERVAL for mean expense by males users-
(9150.68, 9656.24)
--------------------------------------------------
For females-
Mean of sample means = 8728.51283
Population mean = 8734.565765155476
Standard deviation of means (Standard Error) = 153.15329928225864
Standard deviation of population = 4767.233289291444
99% CONFIDENCE INTERVAL for mean expense by females users-
(8317.9, 9052.06)
95% CONFIDENCE INTERVAL for mean expense by females users-
(8385.59, 9003.12)
90% CONFIDENCE INTERVAL for mean expense by females users-
(8458.06, 8952.59)
--------------------------------------------------
```

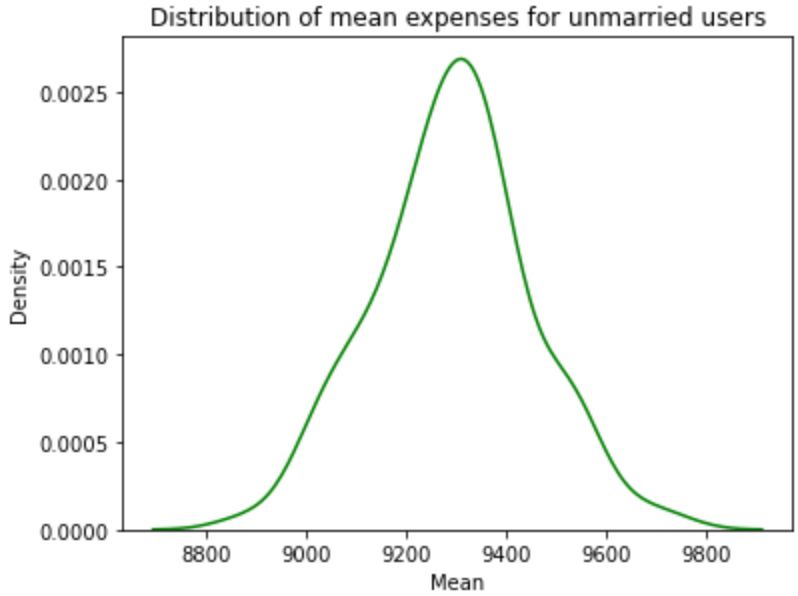
Inference: Using confidence interval 95%, the mean purchase value by gender shows a similar pattern to that found with confidence interval 90%- Overlappings are increasing with a confidence interval of 95%. Due to the increasing CI, we consider higher ranges within which the actual population might fall, so that both mean purchase are more likely to fall within the same range.

In [29]:
```
# for married and Unmarried customers


   # Taking samples of 1000 entries for married and unmarried people and
# Creating kde plots to check if it appears gaussian.
plt.figure(figsize=(6,10))
x = 1
for j in [1,0]:
    means = []
    for i in range(100):
        temp = df.loc[df['Marital_Status']==j,'Purchase'].sample(1000)
        avg = temp.mean()
        means.append(avg)
    plt.subplot(2,1,x)
    sns.kdeplot(x = means, color = 'green')
    if j == 0:
        ms = 'unmarried'
        means_mr = means
    else:
        ms = 'married'
        means_umr = means


    plt.title('Distribution of mean expenses for {m} users'.format(m = ms), f
    plt.xlabel('Mean')
    x += 1
plt.show()
```





Type *Markdown* and LaTeX: α^2

In [30]:
```
# Finding different confidence intervals for mean expense by married and unmar
for i in ['married', 'unmarried']:
    print('For {m}-'.format(m = i))
    if i == 'married':
        means = means_mr
        ms = 1
    else:
        means = means_umr
        ms = 0
    print('Mean of sample means =',np.mean(means))
    print('Population mean =', np.mean(df.loc[df['Marital_Status']==ms, 'Purcl
    print('Standard deviation of means (Standard Error) =', np.std(means))
    print('Standard deviation of population =',df.loc[df['Marital_Status']==ms
    print('99% CONFIDENCE INTERVAL for mean expense by {m} users-'.format(m =
    print((np.percentile(means, 0.5).round(2), np.percentile(means, 99.5).rou
    print('95% CONFIDENCE INTERVAL for mean expense by {m} users-'.format(m =
    print((np.percentile(means, 2.5).round(2), np.percentile(means, 97.5).rou
    print('90% CONFIDENCE INTERVAL for mean expense by {m} users-'.format(m =
    print((np.percentile(means, 5).round(2), np.percentile(means, 95).round(2
    print('-'*50)
```

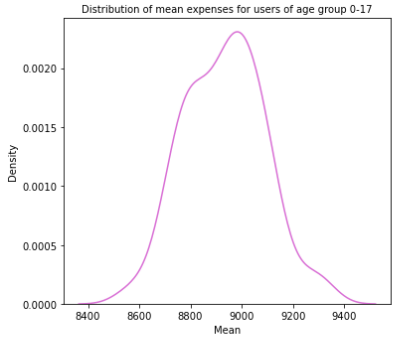
```
For married-
Mean of sample means = 9293.29161
Population mean = 9261.174574082374
Standard deviation of means (Standard Error) = 154.867670989067
Standard deviation of population = 5016.89737779313
99% CONFIDENCE INTERVAL for mean expense by married users-
(8936.5, 9693.57)
95% CONFIDENCE INTERVAL for mean expense by married users-
(9012.39, 9572.28)
90% CONFIDENCE INTERVAL for mean expense by married users-
(9025.22, 9539.55)
--------------------------------------------------
For unmarried-
Mean of sample means = 9253.839890000001
Population mean = 9265.907618921507
Standard deviation of means (Standard Error) = 122.30233869406536
Standard deviation of population = 5027.347858674457
99% CONFIDENCE INTERVAL for mean expense by unmarried users-
(8956.91, 9520.3)
95% CONFIDENCE INTERVAL for mean expense by unmarried users-
(9000.92, 9474.78)
90% CONFIDENCE INTERVAL for mean expense by unmarried users-
(9032.61, 9438.81)
--------------------------------------------------
```

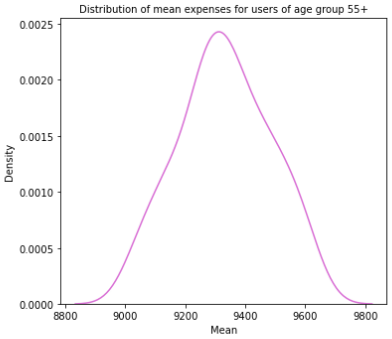
Observations:

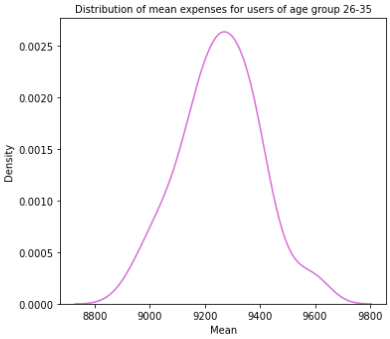
- Mean expense of married customers is 9261.17 Dollars.
- Mean expense of unmarried customers is 9265.90 Dollars.
- There is overlap of confidence intervals but we dont have enough data.

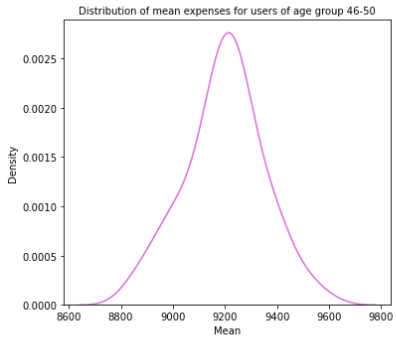
for different age groups

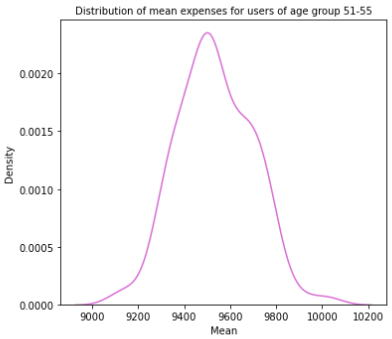
In [31]:
```
#Plotting KDE plots for different age groups
plt.figure(figsize=(20,18))
x = 1
for j in ['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25']:
    means = []
    for i in range(100):
        temp = df.loc[df['Age']==j,'Purchase'].sample(1000)
        avg = temp.mean()
        means.append(avg)
    plt.subplot(3,3,x)
    sns.kdeplot(x = means, color = 'orchid')
    if j == '0-17':
        means_0 = means
    elif j == '55+':
        means_55 = means
    elif j == '26-35':
        means_26 = means
    elif j == '46-50':
        means_46 = means
    elif j == '51-55':
        means_51 = means
    elif j == '36-45':
        means_36 = means
    else:
        means_18 = means
    plt.title('Distribution of mean expenses for users of age group {a}'.forma
    plt.xlabel('Mean')
    x += 1
plt.show()
```

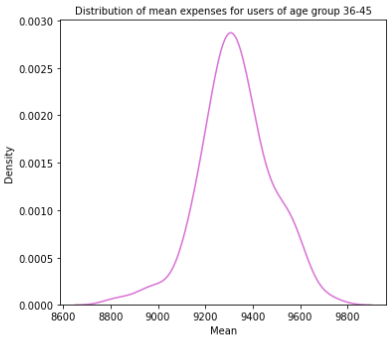

Distribution of mean expenses for users of age group 0-17
Density
Mean

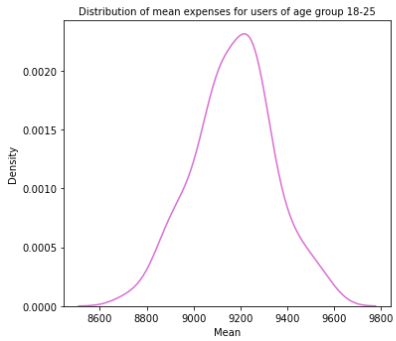

Distribution of mean expenses for users of age group 55+
Density
Mean


Distribution of mean expenses for users of age group 26-35
Density
Mean


Distribution of mean expenses for users of age group 46-50
Density
Mean


Distribution of mean expenses for users of age group 51-55
Density
Mean


Distribution of mean expenses for users of age group 36-45
Density
Mean


Distribution of mean expenses for users of age group 18-25
Density
Mean

In [32]:
```
# Finding confidence intervals for mean purchase for each age group
for i in ['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25']:
    print('For {m}-'.format(m = i))
    if i == '0-17':
        means = means_0
    elif i == '55+':
        means = means_55
    elif i == '26-35':
        means = means_26
    elif i == '46-50':
        means = means_46
    elif i == '51-55':
        means = means_51
    elif i == '36-45':
        means = means_36
    else:
        means = means_18

    print('Mean of sample means =',np.mean(means))
    print('Population mean =', np.mean(df.loc[df['Age']==i, 'Purchase']))
    print('Standard deviation of means (Standard Error) =', np.std(means))
    print('Standard deviation of population =',df.loc[df['Age']==i, 'Purchase
    print('99% CONFIDENCE INTERVAL for mean expense by users of age group {a}
    print((np.percentile(means, 0.5).round(2), np.percentile(means, 99.5).rou
    print('95% CONFIDENCE INTERVAL for mean expense by users of age group {a}
    print((np.percentile(means, 2.5).round(2), np.percentile(means, 97.5).rou
    print('90% CONFIDENCE INTERVAL for mean expense by users of age group {a}
    print((np.percentile(means, 5).round(2), np.percentile(means, 95).round(2
    print('-'*50)
```

```
For 0-17-
Mean of sample means = 8932.709579999999
Population mean = 8933.464640444974
Standard deviation of means (Standard Error) = 155.38325011539567
Standard deviation of population = 5111.11404600277
99% CONFIDENCE INTERVAL for mean expense by users of age group 0-17-
(8571.33, 9318.77)
95% CONFIDENCE INTERVAL for mean expense by users of age group 0-17-
(8670.98, 9267.8)
90% CONFIDENCE INTERVAL for mean expense by users of age group 0-17-
(8687.14, 9163.06)
--------------------------------------------------
For 55+-
Mean of sample means = 9331.93086
Population mean = 9336.280459449405
Standard deviation of means (Standard Error) = 151.60964260389378
Standard deviation of population = 5011.493995603418
99% CONFIDENCE INTERVAL for mean expense by users of age group 55+-
(9021.19, 9630.69)
95% CONFIDENCE INTERVAL for mean expense by users of age group 55+-
(9045.09, 9606.67)
90% CONFIDENCE INTERVAL for mean expense by users of age group 55+-
(9084.8, 9578.48)
--------------------------------------------------
For 26-35-
Mean of sample means = 9255.36436
Population mean = 9252.690632869888
Standard deviation of means (Standard Error) = 145.36742249524275
Standard deviation of population = 5010.527303002927
99% CONFIDENCE INTERVAL for mean expense by users of age group 26-35-
(8929.41, 9612.41)
95% CONFIDENCE INTERVAL for mean expense by users of age group 26-35-
(8972.92, 9575.14)
90% CONFIDENCE INTERVAL for mean expense by users of age group 26-35-
(9006.58, 9482.16)
--------------------------------------------------
For 46-50-
Mean of sample means = 9198.415309999998
Population mean = 9208.625697468327
Standard deviation of means (Standard Error) = 152.14769936332883
Standard deviation of population = 4967.216367142921
99% CONFIDENCE INTERVAL for mean expense by users of age group 46-50-
(8845.53, 9555.48)
95% CONFIDENCE INTERVAL for mean expense by users of age group 46-50-
(8882.06, 9506.27)
90% CONFIDENCE INTERVAL for mean expense by users of age group 46-50-
(8940.06, 9425.23)
--------------------------------------------------
For 51-55-
Mean of sample means = 9534.61479
Population mean = 9534.808030960236
Standard deviation of means (Standard Error) = 161.17670019455636
Standard deviation of population = 5087.368079602116
99% CONFIDENCE INTERVAL for mean expense by users of age group 51-55-
(9141.7, 9962.48)
95% CONFIDENCE INTERVAL for mean expense by users of age group 51-55-
(9276.23, 9796.33)
```

```
90% CONFIDENCE INTERVAL for mean expense by users of age group 51-55-
(9303.55, 9778.03)
--------------------------------------------------
For 36-45-
Mean of sample means = 9327.814680000003
Population mean = 9331.350694917874
Standard deviation of means (Standard Error) = 150.57897975825708
Standard deviation of population = 5022.923879204652
99% CONFIDENCE INTERVAL for mean expense by users of age group 36-45-
(8889.96, 9669.18)
95% CONFIDENCE INTERVAL for mean expense by users of age group 36-45-
(8991.21, 9607.92)
90% CONFIDENCE INTERVAL for mean expense by users of age group 36-45-
(9117.05, 9571.94)
--------------------------------------------------
For 18-25-
Mean of sample means = 9171.28991
Population mean = 9169.663606261289
Standard deviation of means (Standard Error) = 169.54087918576423
Standard deviation of population = 5034.321997176577
99% CONFIDENCE INTERVAL for mean expense by users of age group 18-25-
(8754.48, 9565.45)
95% CONFIDENCE INTERVAL for mean expense by users of age group 18-25-
(8854.5, 9506.53)
90% CONFIDENCE INTERVAL for mean expense by users of age group 18-25-
(8885.13, 9472.15)
--------------------------------------------------
```

Final Insights:

- P00265242 was the most sold product.
- From the given dataset Males were found to make more purchases when compared to females.
- 72% of the customers are male and 28% are Females
- Customers in the age group 26-35 made more purchases than other age group.
- People of city category B made more purchases.
- People who have stayed less than a year or more than 4 years made least purchases.
- Most frequent users have made close to 1000 purchases.
- Least frequent users made 6 or 7 purchases.
- Unmarried people made more purchases than married people.
- Products like P0005632, P00350742 are not being purchased.
- Mean purchase amount for females is Dollars 8734.56
- Mean purchase amount for males is 9437.52
- We can say with cofidnence that 95% of females spend less than males.
- Using confidence interval 95%, the mean purchase value by gender shows a similar pattern to that found with confidence interval 90%-
- Overlappings are increasing with a confidence interval of 95%. Due to the increasing CI, we consider higher ranges within which the actual population might fall, so that both mean purchase are more likely to fall within the same range.

Recommendations:

- Walmart can keep products like P00265242 and P00025442 more in the inventory as they are sold more.
- Products which are not sold more like P00056342, P00350742 need not be kept in store.
- As purchases made by males constitute (72%) and females (28%), campaigns can be made to increase spending by females.
- Management needs to focus on female specific needs and also adding some additional offers for women can increase their spending on Black Friday.
- Ads can be targetted towards people of city category B. Inventory in these cities can be replenished.
- Ads can be targetted towards people who have spent between 1 to 2 years in their cities.
- Ad campaigns need to be tagetted for unmarried people.
- Walmart can keep more products that were purchased by age group 26-35 as they made more purchases than other age groups.