Walmart_SalesPrediction

February 21, 2022

1 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).

Dataset

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

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

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
url ="https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/
 ↪original/walmart_data.csv?1641285094"
```

```
df= pd.read_csv(url)
df.head(5)
```

[2]:

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

Descriptive analysis

[3]:
```
print("Dataframe has {} rows and {} columns".format(df.shape[0], df.shape[1]))
```

```
Dataframe has 550068 rows and 10 columns
```

[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
```

[5]:
```
df.describe()
```

[5]:

	User_ID	Occupation	Marital_Status	Product_Category
count	5.500680e+05	550068.000000	550068.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270
std	1.727592e+03	6.522660	0.491770	3.936211

```
min    1.000001e+06       0.000000         0.000000          1.000000
25%    1.001516e+06       2.000000         0.000000          1.000000
50%    1.003077e+06       7.000000         0.000000          5.000000
75%    1.004478e+06      14.000000         1.000000          8.000000
max    1.006040e+06      20.000000         1.000000         20.000000

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

[6]:
```
# Statistical summary
df.describe().T
```

[6]:
```
                    count          mean          std        min        25%  \
User_ID          550068.0  1.003029e+06  1727.591586  1000001.0  1001516.0
Occupation       550068.0  8.076707e+00     6.522660        0.0        2.0
Marital_Status   550068.0  4.096530e-01     0.491770        0.0        0.0
Product_Category 550068.0  5.404270e+00     3.936211        1.0        1.0
Purchase         550068.0  9.263969e+03  5023.065394       12.0     5823.0

                       50%        75%        max
User_ID          1003077.0  1004478.0  1006040.0
Occupation             7.0       14.0       20.0
Marital_Status         0.0        1.0        1.0
Product_Category       5.0        8.0       20.0
Purchase            8047.0    12054.0    23961.0
```

[7]:
```
df[df['Gender']=='M'].describe().T
```

[7]:
```
                    count          mean          std        min        25%  \
User_ID          414259.0  1.002996e+06  1706.493873  1000002.0  1001505.0
Occupation       414259.0  8.514750e+00     6.553790        0.0        3.0
Marital_Status   414259.0  4.063859e-01     0.491159        0.0        0.0
Product_Category 414259.0  5.301512e+00     4.006275        1.0        1.0
Purchase         414259.0  9.437526e+03  5092.186210       12.0     5863.0

                       50%        75%        max
User_ID          1003041.0  1004411.0  1006040.0
Occupation             7.0       15.0       20.0
Marital_Status         0.0        1.0        1.0
Product_Category       5.0        8.0       20.0
```

```
Purchase             8098.0    12454.0    23961.0
```

[8]:
```
df[df['Gender']=='F'].describe().T
```

[8]:
```
                      count          mean          std        min        25%  \
User_ID             135809.0  1.003130e+06  1786.630589  1000001.0  1001569.0
Occupation          135809.0  6.740540e+00     6.239639        0.0        1.0
Marital_Status      135809.0  4.196187e-01     0.493498        0.0        0.0
Product_Category    135809.0  5.717714e+00     3.696752        1.0        3.0
Purchase            135809.0  8.734566e+03  4767.233289       12.0     5433.0

                         50%        75%        max
User_ID            1003159.0  1004765.0  1006039.0
Occupation               4.0       11.0       20.0
Marital_Status           0.0        1.0        1.0
Product_Category         5.0        8.0       20.0
Purchase              7914.0    11400.0    23959.0
```

[9]:
```
df.apply(lambda x: len(x.unique()))
```

[9]:
```
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
```

[10]:
```
df.isnull().sum()
```

[10]:
```
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
```

[11]:
```
df['Age'].value_counts()
```

[11]:
```
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
```

[12]:
```
#Checking Percentage count of Age
print('Data Point Percentage per Age Group')
round(df['Age'].value_counts(normalize=True).mul(100), 2).astype(str) + '%'
```

```
Data Point Percentage per Age Group
```

[12]:
```
26-35    39.92%
36-45     20.0%
18-25    18.12%
46-50     8.31%
51-55      7.0%
55+       3.91%
0-17      2.75%
Name: Age, dtype: object
```

[13]:
```
print('Data Point Percentage per No of Years Stayed in Current City')
round(df['Stay_In_Current_City_Years'].value_counts(normalize=True).mul(100),␣
↪2).astype(str)+'%'
```

```
Data Point Percentage per No of Years Stayed in Current City
```

[13]:
```
1     35.24%
2     18.51%
3     17.32%
4+     15.4%
0     13.53%
Name: Stay_In_Current_City_Years, dtype: object
```

[14]:
```
print('Data Point Percentage per Product Category')
round(df['Product_Category'].value_counts(normalize=True).mul(100), 2).
↪astype(str)+'%'
```

```
Data Point Percentage per Product Category
```

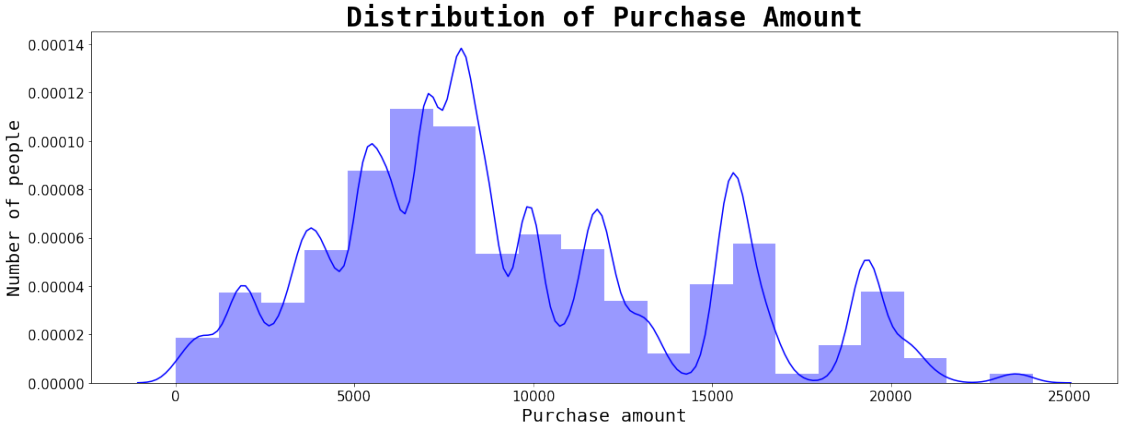
[14]:
```
5     27.44%
1     25.52%
8     20.71%
11     4.42%
2      4.34%
6      3.72%
```

```
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: object
```

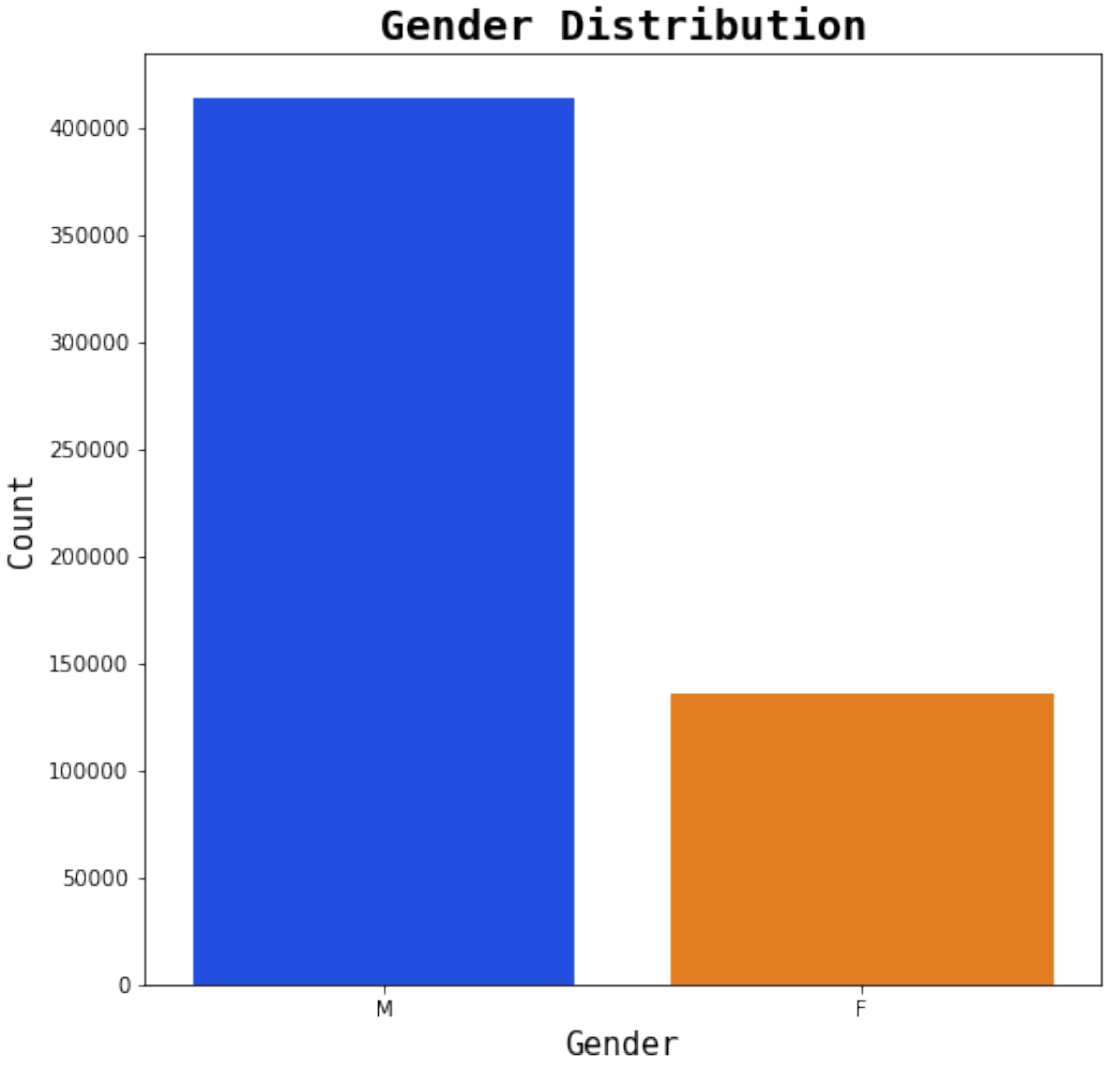
1.1 Exploratory Data Analysis:

1.1.1 Univariate Analysis:

[15]:
```
plt.figure(figsize = (20, 7))
sns.distplot(df['Purchase'], bins = 20, color='BLUE')
plt.title('Distribution of Purchase Amount', fontdict = {'fontname' :␣
↪'Monospace', 'fontsize' : 30, 'fontweight' : 'bold'})
plt.xlabel('Purchase amount', fontdict = {'fontname' : 'Monospace', 'fontsize' :
↪ 20})
plt.ylabel('Number of people', fontdict = {'fontname' : 'Monospace', 'fontsize'␣
↪: 20})
plt.tick_params(labelsize = 15)
plt.show()
```



[16]:
```
gender = df['Gender'].value_counts()
plt.figure(figsize = (8, 8))
sns.barplot(gender.index, gender.values, palette = 'bright')
plt.title('Gender Distribution', fontdict = {'fontname' : 'Monospace',␣
↪'fontsize' : 20, 'fontweight' : 'bold'})
plt.xlabel('Gender', fontdict = {'fontname' : 'Monospace', 'fontsize' : 15})
plt.ylabel('Count', fontdict = {'fontname' : 'Monospace', 'fontsize' : 15})
plt.tick_params(labelsize = 10)
plt.show()
```

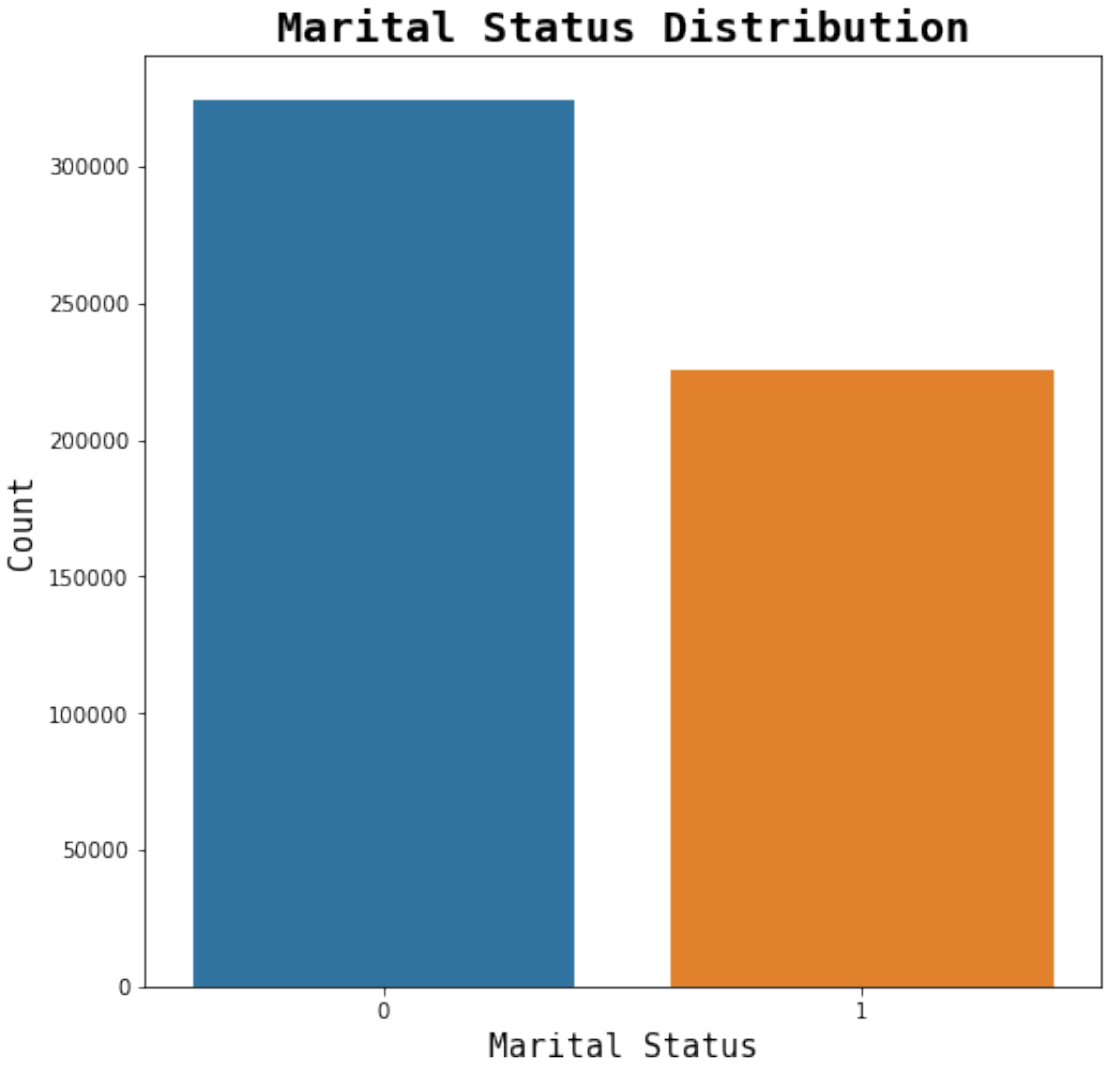


[17]:
```
df.columns
```

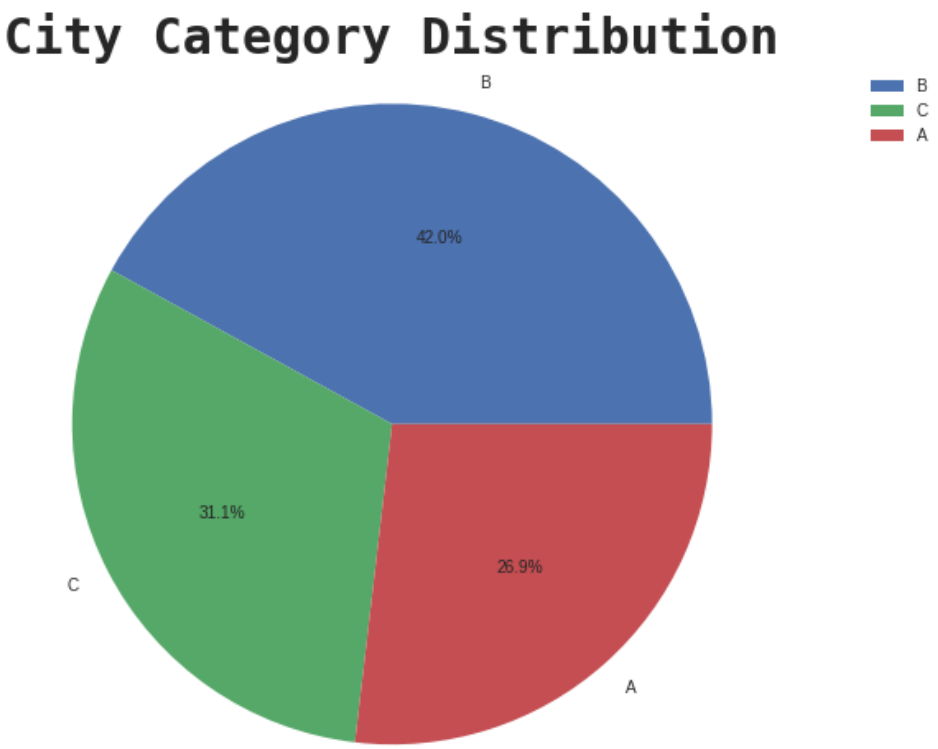
[17]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
```

```
       'Purchase'],
      dtype='object')
```

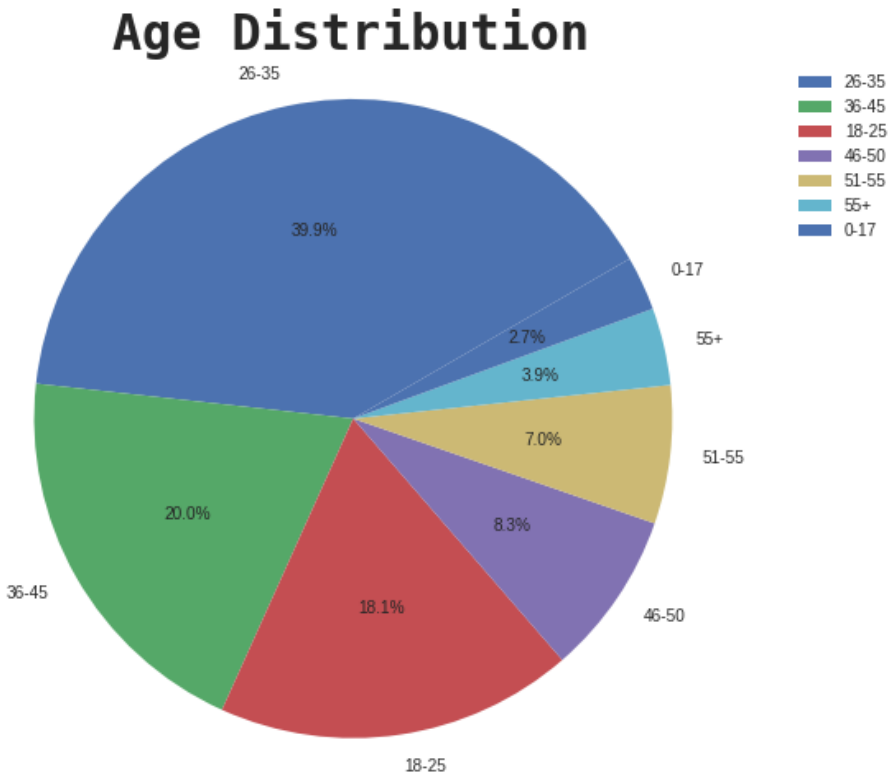
[18]:
```
plt.figure(figsize = (8, 8))
sns.countplot(df['Marital_Status'])
plt.title('Marital Status Distribution', fontdict = {'fontname' : 'Monospace',␣
↪'fontsize' : 20, 'fontweight' : 'bold'})
plt.xlabel('Marital Status', fontdict = {'fontname' : 'Monospace', 'fontsize' :␣
↪15})
plt.ylabel('Count', fontdict = {'fontname' : 'Monospace', 'fontsize' : 15})
plt.tick_params(labelsize = 10)
plt.show()
```



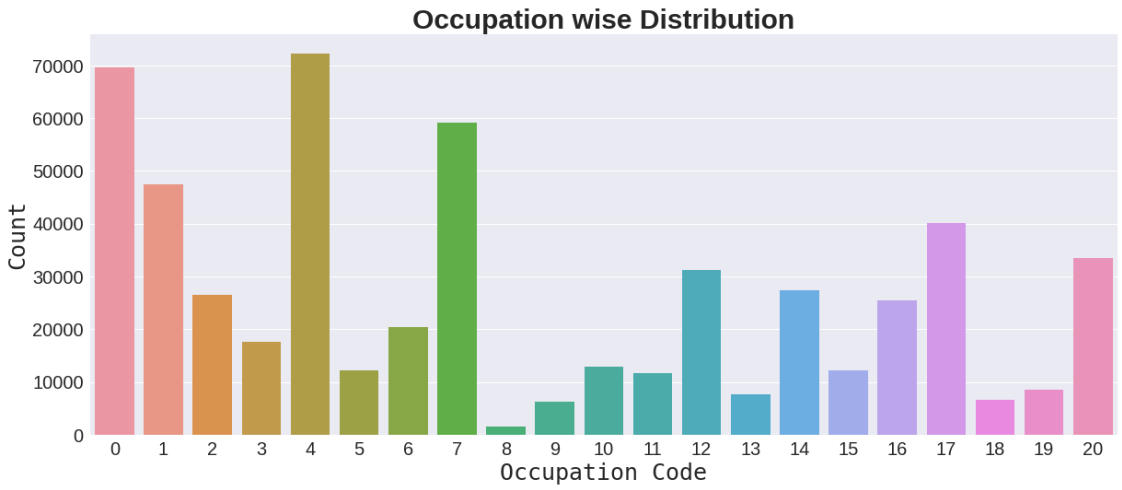
[19]:
```
city = df['City_Category'].value_counts()
plt.style.use('seaborn')
plt.figure(figsize = (12, 8))
plt.pie(city.values, labels = city.index,  autopct = '%1.1f%%')
plt.title('City Category Distribution', fontdict = {'fontname' : 'Monospace',␣
↪'fontsize' : 30, 'fontweight' : 'bold'})
plt.legend()
plt.axis('equal')
plt.show()
```



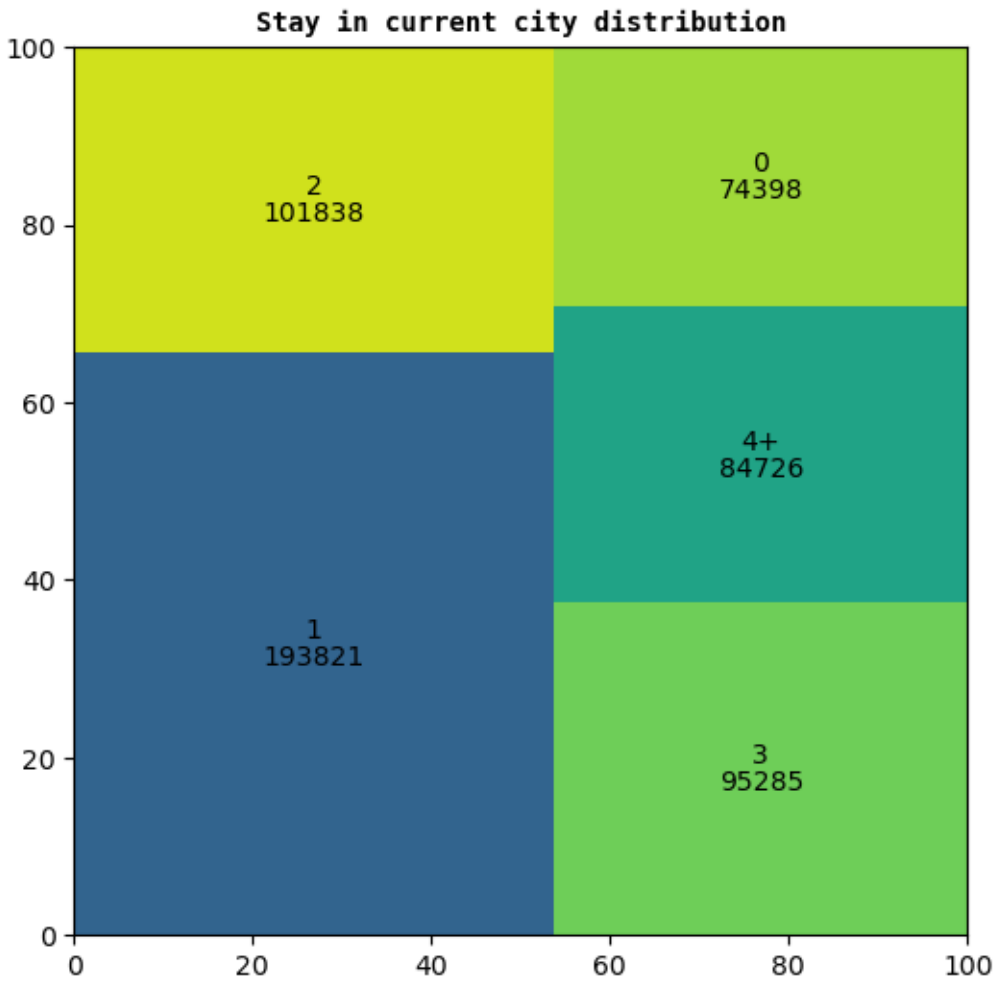
[20]:
```
city = df['Age'].value_counts()
plt.style.use('seaborn')
plt.figure(figsize = (12, 8))
plt.pie(city.values, labels = city.index, startangle = 30, autopct = '%1.1f%%')
plt.title('Age Distribution', fontdict = {'fontname' : 'Monospace', 'fontsize' :
↪ 30, 'fontweight' : 'bold'})
plt.legend()
plt.axis('equal')
plt.show()
```



[21]:
```
occupation = df['Occupation'].value_counts()

plt.figure(figsize = (20, 8))
sns.barplot(occupation.index, occupation.values)
plt.title('Occupation wise Distribution', fontdict = {'fontsize' : 30,␣
↪'fontweight' : 'bold'})
plt.xlabel('Occupation Code', fontdict = {'fontname' : 'Monospace', 'fontsize' :
↪ 25})
plt.ylabel('Count', fontdict = {'fontname' : 'Monospace', 'fontsize' : 25})
plt.tick_params(labelsize = 20)
plt.show()
```



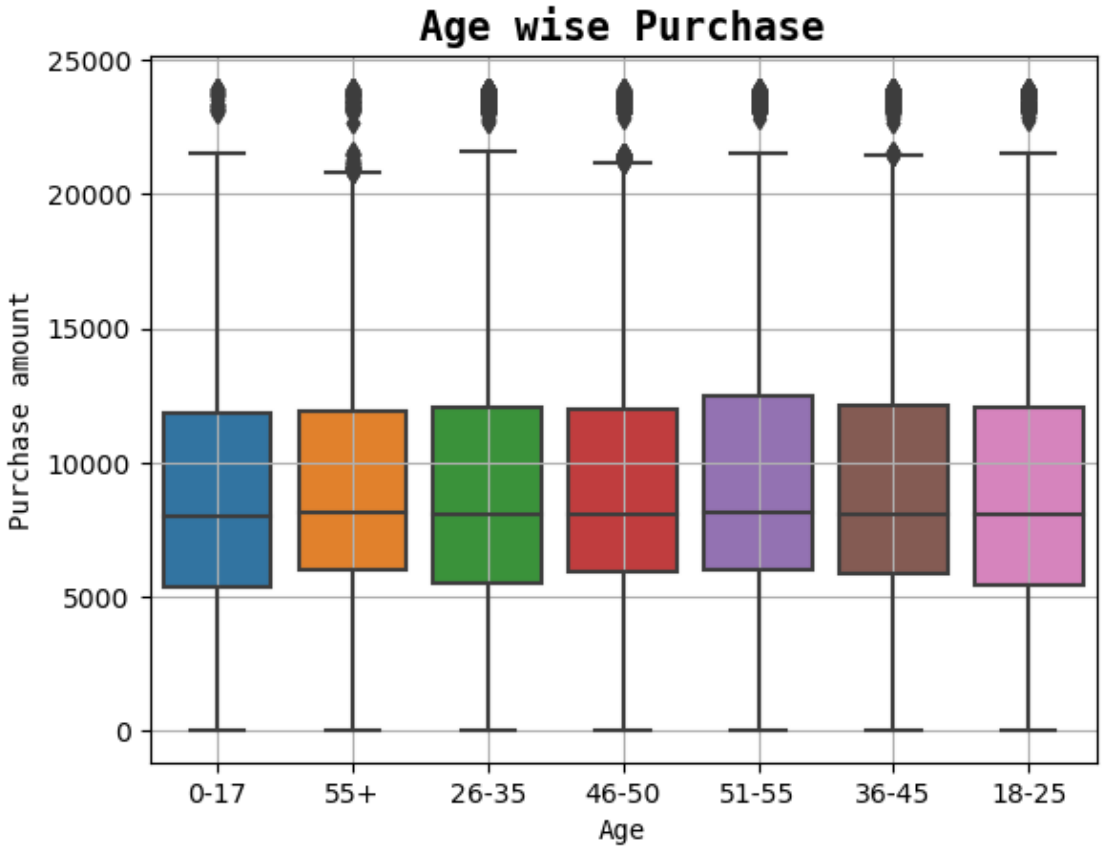
[22]:
```
stay = df['Stay_In_Current_City_Years'].value_counts()

import squarify
plt.style.use('default')
plt.figure(figsize = (6, 6))
squarify.plot(sizes = stay.values, label = stay.index, value = stay.values)
plt.title('Stay in current city distribution', fontdict = {'fontname' :␣
↪'Monospace', 'fontsize' : 10, 'fontweight' : 'bold'})
plt.show()
```



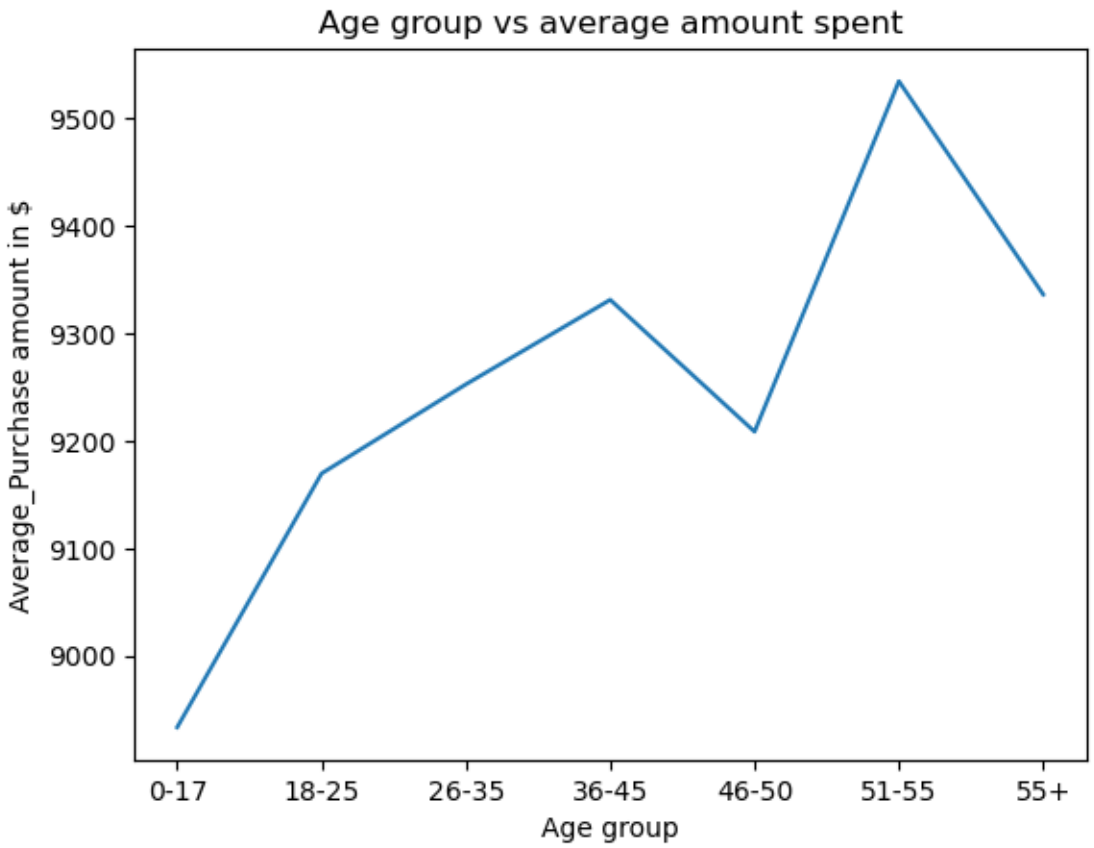
1.1.2 Bivariate Analysis:

Age vs Purchase

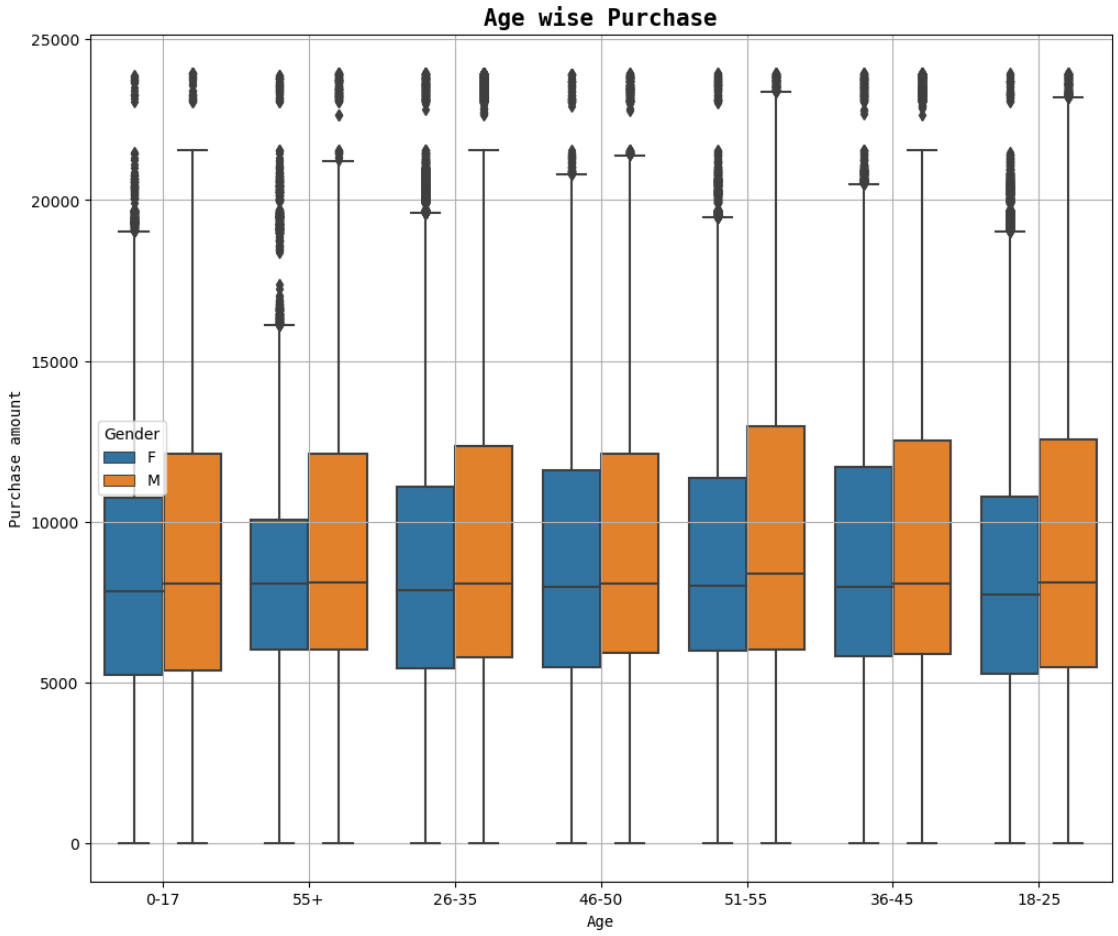
[23]:
```
sns.boxplot(df['Age'], df['Purchase'])
plt.title('Age wise Purchase', fontdict = {'fontname' : 'Monospace', 'fontsize'␣
↪: 15, 'fontweight' : 'bold'})
plt.xlabel('Age', fontdict = {'fontname' : 'Monospace', 'fontsize' : 10})
plt.ylabel('Purchase amount', fontdict = {'fontname' : 'Monospace', 'fontsize' :
↪ 10})
plt.tick_params(labelsize = 10)
plt.grid()
plt.show()
```



[24]:
```
df.groupby('Age')['Purchase'].mean().plot()
plt.xlabel('Age group')
plt.ylabel('Average_Purchase amount in $')
plt.title('Age group vs average amount spent')
plt.show()
```

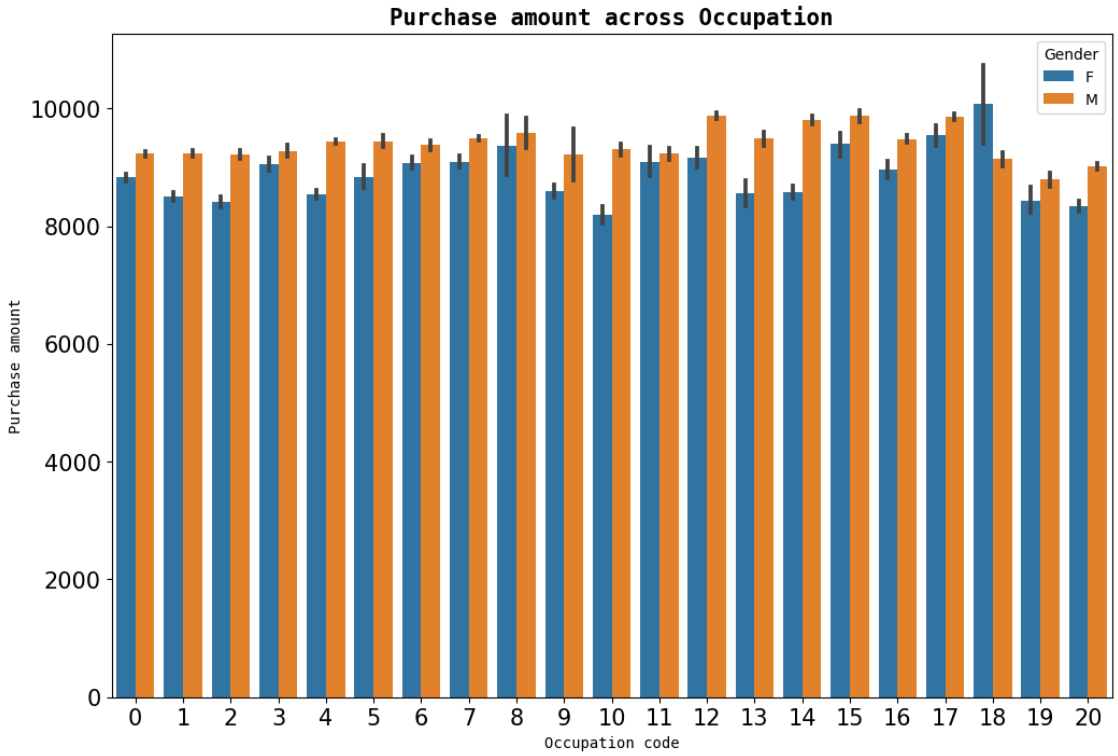


[25]:
```
plt.figure(figsize = (12, 10))
sns.boxplot(df['Age'], df['Purchase'], hue=df['Gender'])
plt.title('Age wise Purchase', fontdict = {'fontname' : 'Monospace', 'fontsize'␣
↪: 15, 'fontweight' : 'bold'})
plt.xlabel('Age', fontdict = {'fontname' : 'Monospace', 'fontsize' : 10})
plt.ylabel('Purchase amount', fontdict = {'fontname' : 'Monospace', 'fontsize' :
↪ 10})
plt.tick_params(labelsize = 10)
plt.grid()
plt.show()
```

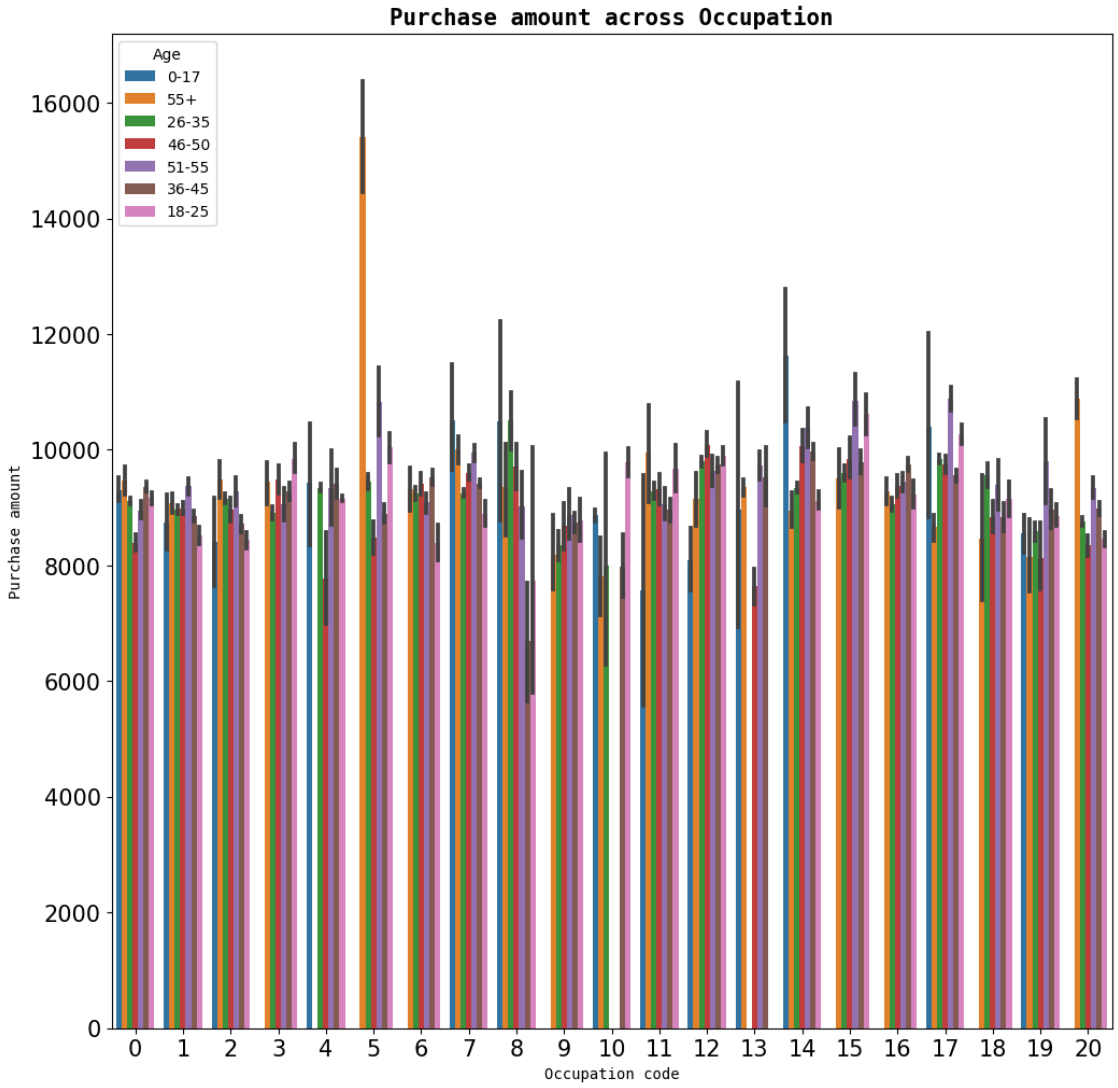


1.1.3 'Occupation vs Purchase'

[26]:
```
plt.figure(figsize = (12, 8))
sns.barplot(df['Occupation'], df['Purchase'], hue=df['Gender'])
plt.title('Purchase amount across Occupation', fontdict = {'fontname' :␣
↪'Monospace', 'fontsize' : 15, 'fontweight' : 'bold'})
plt.xlabel('Occupation code', fontdict = {'fontname' : 'Monospace', 'fontsize' :
↪ 10})
plt.ylabel('Purchase amount', fontdict = {'fontname' : 'Monospace', 'fontsize' :
↪ 10})
plt.tick_params(labelsize = 15)
plt.show()
```

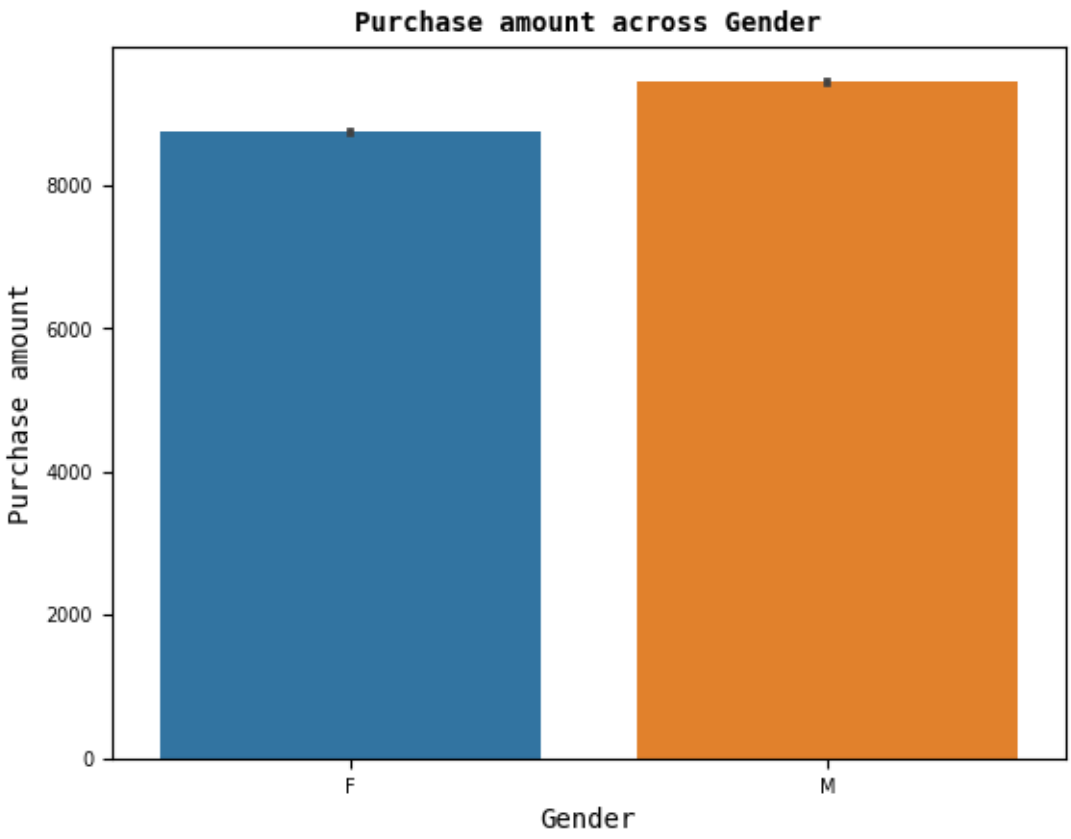


[27]:
```
plt.figure(figsize = (12, 12))
sns.barplot(df['Occupation'], df['Purchase'], hue=df['Age'])
plt.title('Purchase amount across Occupation', fontdict = {'fontname' :␣
↪'Monospace', 'fontsize' : 15, 'fontweight' : 'bold'})
plt.xlabel('Occupation code', fontdict = {'fontname' : 'Monospace', 'fontsize' :
↪ 10})
plt.ylabel('Purchase amount', fontdict = {'fontname' : 'Monospace', 'fontsize' :
↪ 10})
plt.tick_params(labelsize = 15)
plt.show()
```



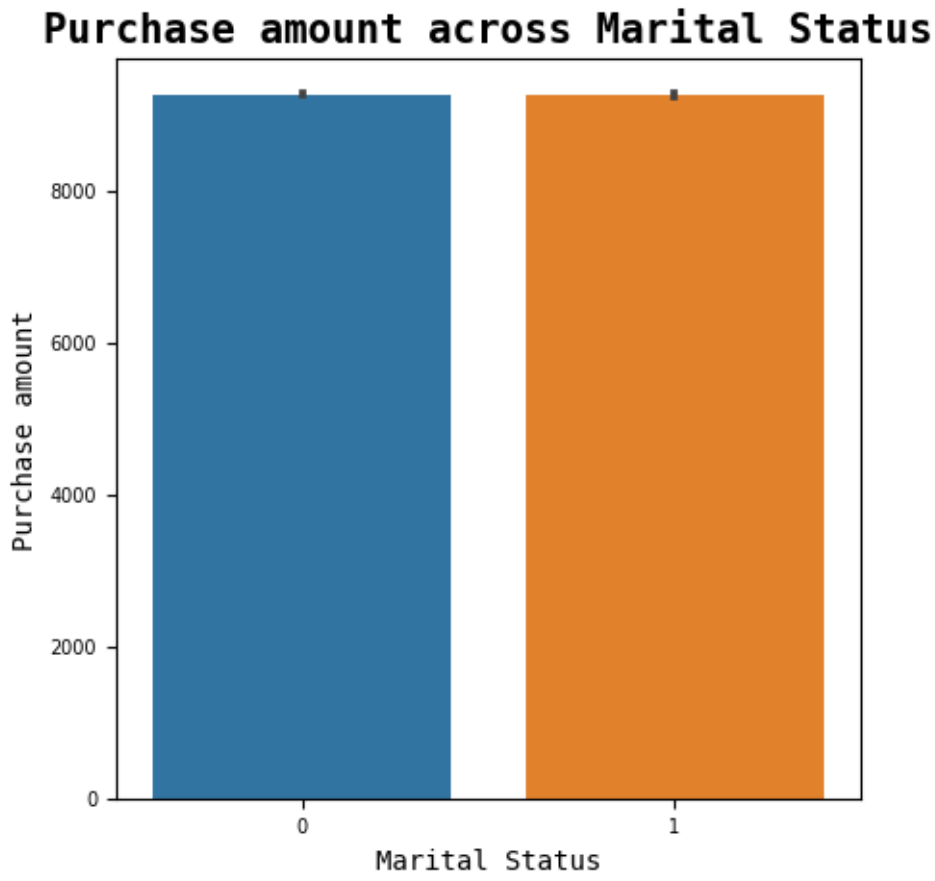
'Gender vs Purchase'

[28]:
```
sns.barplot(df['Gender'], df['Purchase'])
plt.title('Purchase amount across Gender', fontdict = {'fontname' :␣
↪'Monospace', 'fontsize' : 10, 'fontweight' : 'bold'})
plt.xlabel('Gender', fontdict = {'fontname' : 'Monospace', 'fontsize' : 10})
plt.ylabel('Purchase amount', fontdict = {'fontname' : 'Monospace', 'fontsize' :
↪ 10})
plt.tick_params(labelsize = 7)
plt.show()
```

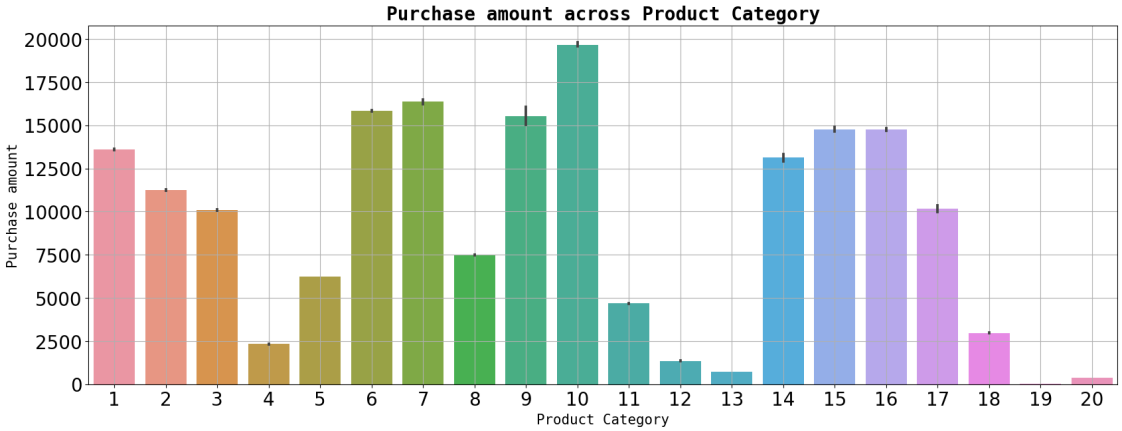


[29]:
```
sns.barplot(df['City_Category'], df['Purchase'])
plt.title('Purchase amount across City Category', fontdict = {'fontname' :␣
↪'Monospace', 'fontsize' : 15, 'fontweight' : 'bold'})
plt.xlabel('City Category', fontdict = {'fontname' : 'Monospace', 'fontsize' :␣
↪15})
plt.ylabel('Purchase amount', fontdict = {'fontname' : 'Monospace', 'fontsize' :
↪ 15})
plt.tick_params(labelsize = 7)
plt.show()
```

[30]:
```
plt.figure(figsize = (5, 5))
sns.barplot(df['Marital_Status'], df['Purchase'])
plt.title('Purchase amount across Marital Status', fontdict = {'fontname' :␣
↪'Monospace', 'fontsize' : 15, 'fontweight' : 'bold'})
plt.xlabel('Marital Status', fontdict = {'fontname' : 'Monospace', 'fontsize' :␣
↪10})
plt.ylabel('Purchase amount', fontdict = {'fontname' : 'Monospace', 'fontsize' :
↪ 10})
plt.tick_params(labelsize = 7)
plt.show()
```



[31]:
```
plt.figure(figsize = (20, 7))
sns.barplot(df['Product_Category'], df['Purchase'])
plt.title('Purchase amount across Product Category', fontdict = {'fontname' :␣
↪'Monospace', 'fontsize' : 20, 'fontweight' : 'bold'})
plt.xlabel('Product Category', fontdict = {'fontname' : 'Monospace', 'fontsize'␣
↪: 15})
plt.ylabel('Purchase amount', fontdict = {'fontname' : 'Monospace', 'fontsize' :
↪ 15})
plt.tick_params(labelsize = 20)
plt.grid()
plt.show()
```



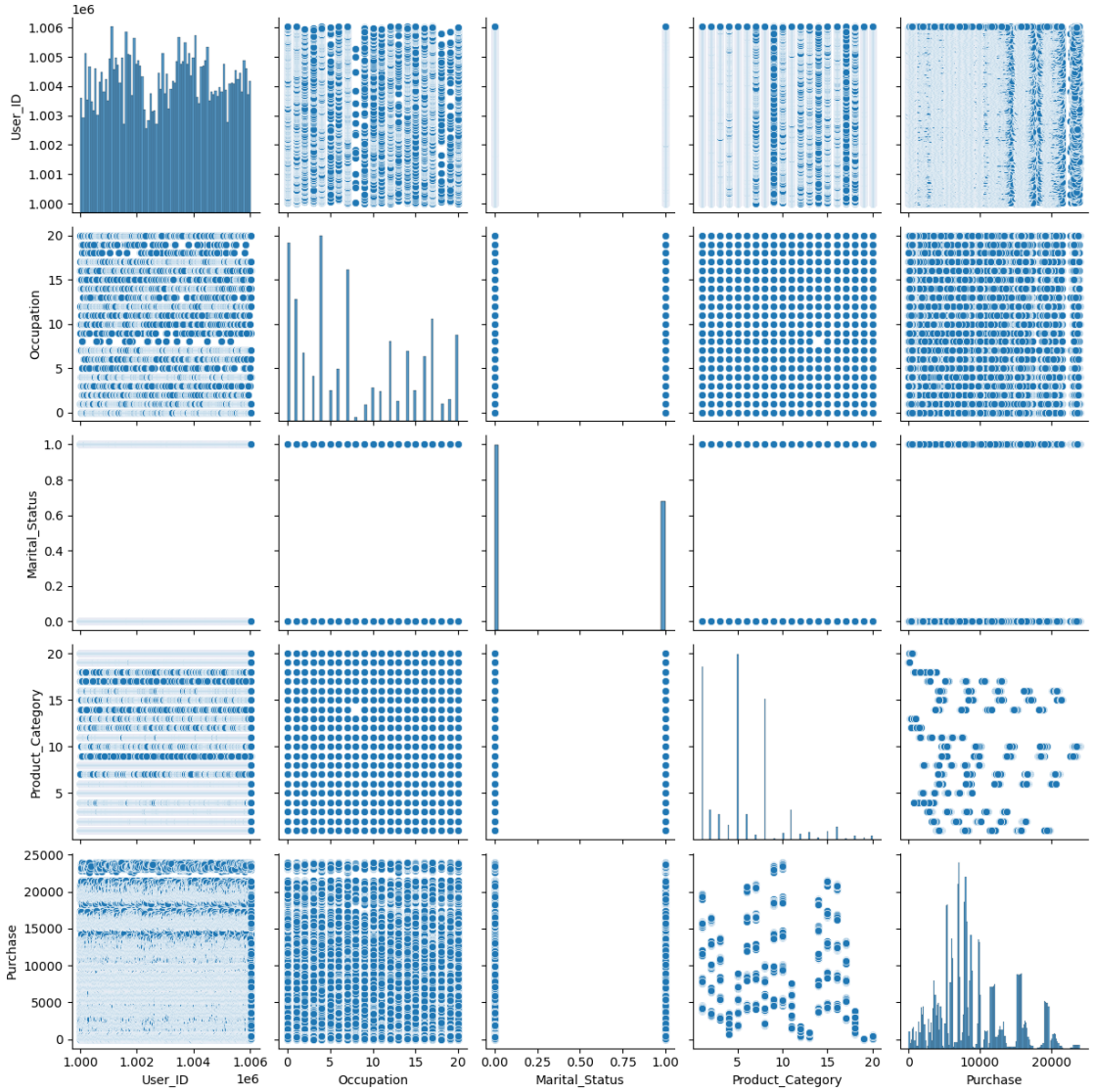
[32]:
```
labels = []
values = []

for uniqueOccupationValue in df['Occupation'].sort_values().unique():
    OccPurchaserData = df.loc[df['Occupation'] == uniqueOccupationValue]
    OccPurchaserMean = np.mean(OccPurchaserData['Purchase'])
    labels.append(uniqueOccupationValue)
    values.append(OccPurchaserMean)

    print("When occupation = ",uniqueOccupationValue," mean purchase value =␣
 ↪",OccPurchaserMean)
    print("------------------------------------------------------------")
```

```
When occupation =  0  mean purchase value =  9124.428587839973
------------------------------------------------------------
When occupation =  1  mean purchase value =  8953.193269514612
------------------------------------------------------------
When occupation =  2  mean purchase value =  8952.481683466225
------------------------------------------------------------
When occupation =  3  mean purchase value =  9178.593087818697
------------------------------------------------------------
When occupation =  4  mean purchase value =  9213.980251147868
------------------------------------------------------------
When occupation =  5  mean purchase value =  9333.149297856615
------------------------------------------------------------
When occupation =  6  mean purchase value =  9256.535691476296
------------------------------------------------------------
When occupation =  7  mean purchase value =  9425.728222819745
------------------------------------------------------------
When occupation =  8  mean purchase value =  9532.592496765847
------------------------------------------------------------
When occupation =  9  mean purchase value =  8637.74376092831
```

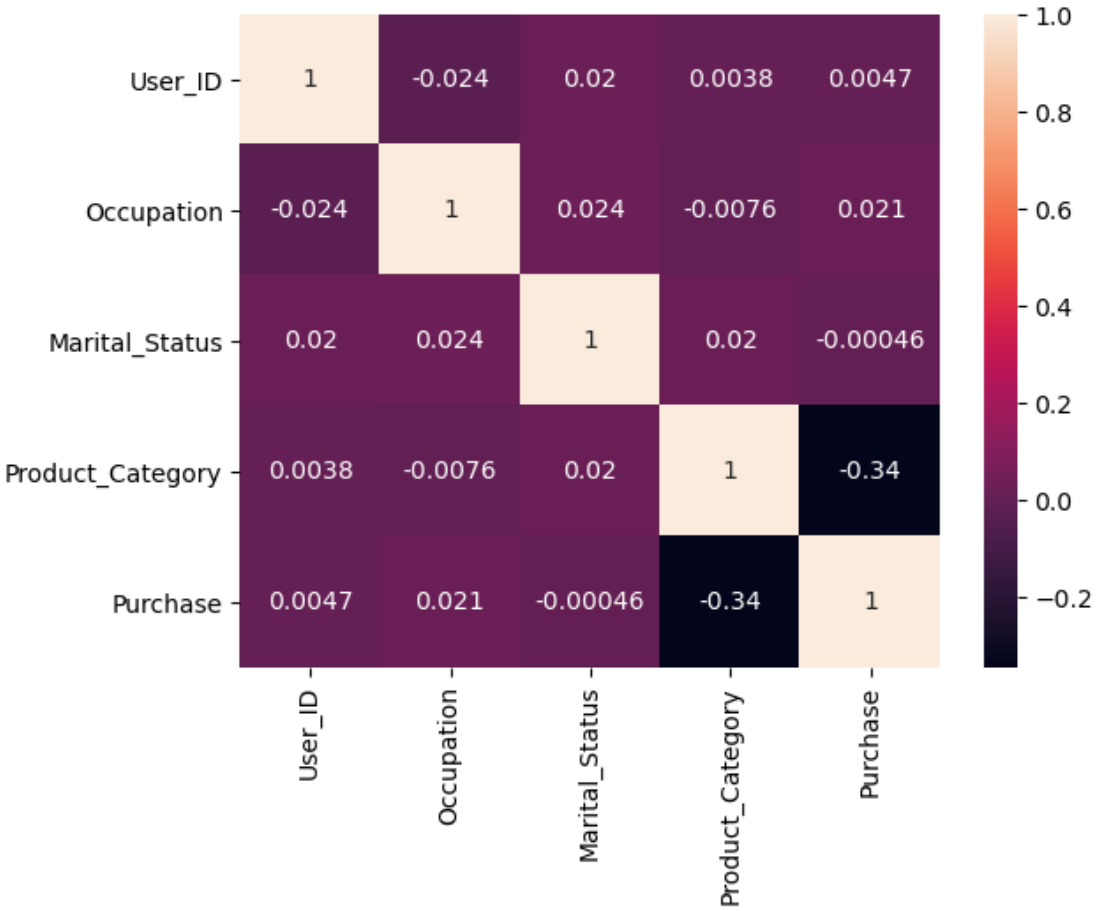
```
------------------------------------------------------------
When occupation =  10  mean purchase value =  8959.355375096675
------------------------------------------------------------
When occupation =  11  mean purchase value =  9213.84584843777
------------------------------------------------------------
When occupation =  12  mean purchase value =  9796.640238622149
------------------------------------------------------------
When occupation =  13  mean purchase value =  9306.351061076604
------------------------------------------------------------
When occupation =  14  mean purchase value =  9500.702771979933
------------------------------------------------------------
When occupation =  15  mean purchase value =  9778.891163173037
------------------------------------------------------------
When occupation =  16  mean purchase value =  9394.46434905995
------------------------------------------------------------
When occupation =  17  mean purchase value =  9821.478235896411
------------------------------------------------------------
When occupation =  18  mean purchase value =  9169.655844155845
------------------------------------------------------------
When occupation =  19  mean purchase value =  8710.62723082378
------------------------------------------------------------
When occupation =  20  mean purchase value =  8836.49490495203
------------------------------------------------------------
```

1.1.4 Multivariate Analysis:

[33]:
```
sns.pairplot(df)
plt.show()
```



[34]:
```
sns.heatmap(df.corr(), annot = True)
plt.show()
```



Central Limit Theorem

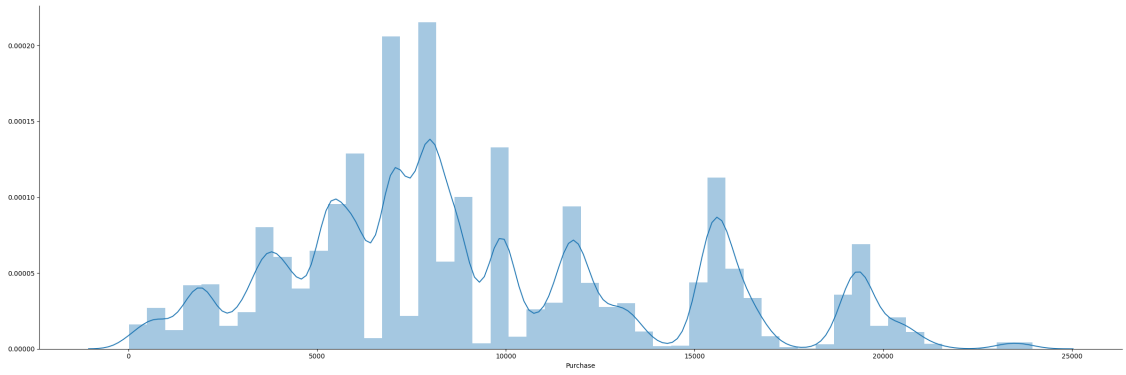
[35]:
```
import random
import sidetable
import prettytable
```

[36]:
```
# Popullation Mean
pop_mean = df["Purchase"].mean()
pop_std = df["Purchase"].std()
print("  and   of overall purchases are {} and {} units respectively.".
 ↪format(round(pop_mean,2),round(pop_std,2)))

# Popullation Mean
pop_mean_F = df[df['Gender']=='F'].Purchase.mean()
pop_std_F = df[df['Gender']=='F'].Purchase.std()
print("  and   of overall purchases for Female are {} and {} units respectively.
 ↪".format(round(pop_mean_F,2),round(pop_std_F,2)))

# Popullation Mean
```

```
pop_mean_M = df[df['Gender']=='M'].Purchase.mean()
pop_std_M = df[df['Gender']=='M'].Purchase.std()
print("  and   of overall purchases for Male are {} and {} units respectively.".
 ↪format(round(pop_mean_M,2),round(pop_std_M,2)))
```

```
  and   of overall purchases are 9263.97 and 5023.07 units respectively.
  and   of overall purchases for Female are 8734.57 and 4767.23 units
respectively.
  and   of overall purchases for Male are 9437.53 and 5092.19 units
respectively.
```

[37]:
```
#distribution plot of overall purchase
sns.FacetGrid(df,height=8,aspect=3).map(sns.distplot,"Purchase")
plt.show()
```



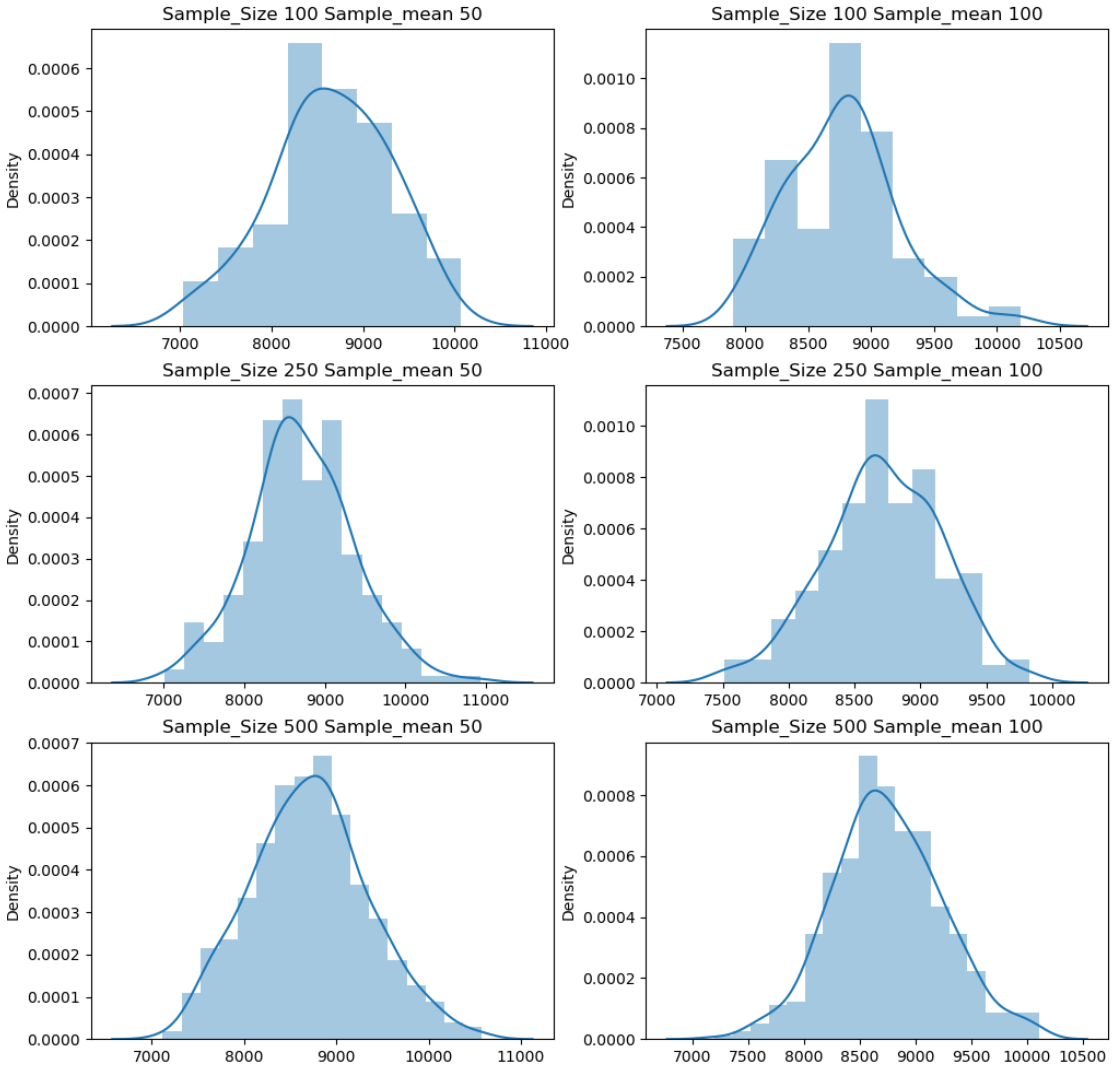
The overall distribution is asymetric. Let's take sample size>30 and plot the mean distribution to check whether it follows normal distribution or not.

[38]:
```
df_female = df[df['Gender']=='F']
df_male = df[df['Gender']=='M']
```

[39]:
```
def sample_mean_distribution(data, samples_count, data_points_count):
    samples_list = list()
    data = np.array(data.values)
    for i in range(0, samples_count):
        samples = random.sample(range(0, data.shape[0]), data_points_count)
        samples_list.append(data[samples].mean())
    return np.array(samples_list)
```

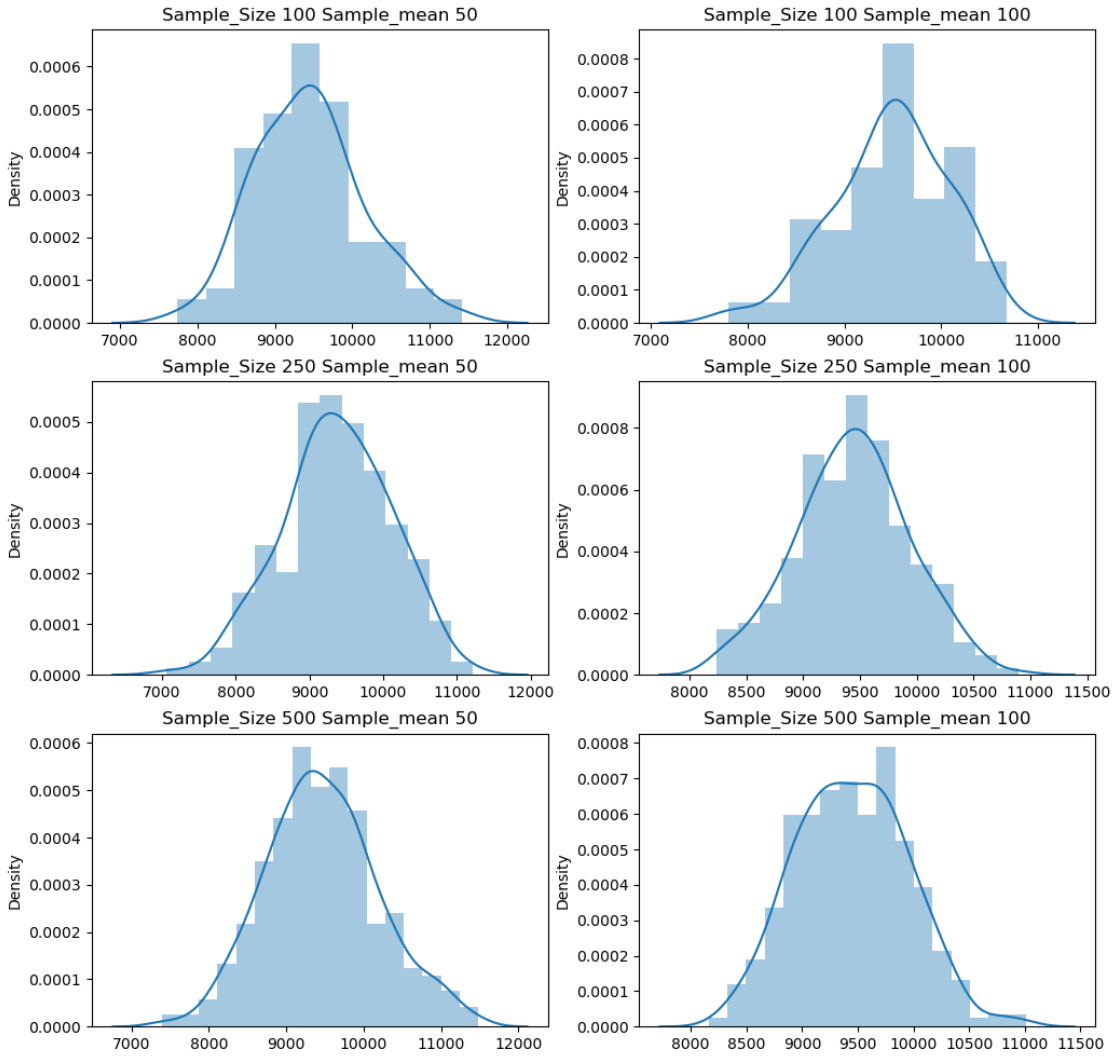
[40]:
```
cnt = 0
sample_mean_F = list()
fg, ax = plt.subplots(nrows=3, ncols=2, figsize=(12, 12))
#list of samples and datapoints combinations
lst = [(100,50),(100,100),(250,50),(250,100),(500,50),(500,100)]
```

```
for i in range(0,3):
    for j in range(0,2):
        ax[i,j].set_title("Sample_Size " + str(lst[cnt][0]) + " Sample_mean " +␣
↪str(lst[cnt][1]))
        sns.distplot(sample_mean_distribution(df_female["Purchase"],␣
↪lst[cnt][0],lst[cnt][1]),ax = ax[i,j])
        sample_mean_F.append(sample_mean_distribution(df_female["Purchase"],␣
↪lst[cnt][0],lst[cnt][1]))
        cnt +=1
```



[41]:
```
cnt = 0
sample_mean_M = list()
fg, ax = plt.subplots(nrows=3, ncols=2, figsize=(12, 12))
```

```
#list of samples and datapoints combinations
lst = [(100,50),(100,100),(250,50),(250,100),(500,50),(500,100)]
for i in range(0,3):
    for j in range(0,2):
        ax[i,j].set_title("Sample_Size " + str(lst[cnt][0]) + " Sample_mean " +␣
↪str(lst[cnt][1]))
        sns.distplot(sample_mean_distribution(df_male["Purchase"],␣
↪lst[cnt][0],lst[cnt][1]),ax = ax[i,j])
        sample_mean_M.append(sample_mean_distribution(df_male["Purchase"],␣
↪lst[cnt][0],lst[cnt][1]))
        cnt +=1
```



[42]:
```
#calculating output
```

```
def sample_output(data, population_mean,␣
↪population_sd,total_sample_points_list):
    temp_df = pd.DataFrame()
    mean_sample = list(); std_sample = list(); std_approx = list();␣
↪strings_list = list()
    pop_mean = list(); pop_std = list()
    #calculating sample values
    for idx, val in enumerate(data):
        pop_mean.append(round(population_mean,2))
        pop_std.append(round(population_sd,2))
        mean_sample.append(round(val.mean(),2))
        std_sample.append(round(val.std(),2))
        std_approx.append(round(population_sd/np.
↪sqrt((total_sample_points_list[idx][1])),2))
        strings_list.append("total_sample_" +␣
↪str(total_sample_points_list[idx][0]) + "_total_mean_"
                            + str(total_sample_points_list[idx][1]))
    temp_df["name"] = pd.Series(strings_list); temp_df["purchase_mean"] = pd.
↪Series(pop_mean)
    temp_df["sample_mean"] = pd.Series(mean_sample); temp_df["purchase_std"] =␣
↪pd.Series(pop_std)
    temp_df["sample_std"] = pd.Series(std_sample); temp_df["approx_std"] = pd.
↪Series(std_approx)

    return temp_df
```

Comparision of Smaple Mean and Std Deviation with Actual Mean and Std Deviation For Female

[43]:
```
output_df_F = sample_output(sample_mean_F, pop_mean_F, pop_std_F,lst)
output_df_F
```

[43]:

	name	purchase_mean	sample_mean	purchase_std	sample_std	approx_std
0	total_sample_100_total_mean_50	8734.57	8939.48	4767.23	705.42	674.19
1	total_sample_100_total_mean_100	8734.57	8743.40	4767.23	537.88	476.72
2	total_sample_250_total_mean_50	8734.57	8663.45	4767.23	686.65	674.19
3	total_sample_250_total_mean_100	8734.57	8733.91	4767.23	467.51	476.72
4	total_sample_500_total_mean_50	8734.57	8712.33	4767.23	677.06	674.19
5	total_sample_500_total_mean_100	8734.57	8730.28	4767.23	485.32	476.72

Comparision of Smaple Mean and Std Deviation with Actual Mean and Std Deviation For Male

[44]:
```
output_df_M = sample_output(sample_mean_M, pop_mean_M, pop_std_M,lst)
output_df_M
```

[44]:

	name	purchase_mean	sample_mean	purchase_std	sample_std	approx_std
0	total_sample_100_total_mean_50	9437.53	9348.49	5092.19	764.53	720.14
1	total_sample_100_total_mean_100	9437.53	9472.32	5092.19	530.62	509.22
2	total_sample_250_total_mean_50	9437.53	9446.45	5092.19	673.35	720.14
3	total_sample_250_total_mean_100	9437.53	9440.49	5092.19	510.15	509.22
4	total_sample_500_total_mean_50	9437.53	9427.64	5092.19	697.03	720.14
5	total_sample_500_total_mean_100	9437.53	9452.02	5092.19	516.26	509.22

As the number of samples increases, the sample mean and sd becomes closer to the original mean and sd. So our approach and observations using CLT are valid.

Getting lower and upper limit of 95% confidence interval with known standard deviation of population for Female and Male Respectively

[45]:
```
lower_lim = list(); upper_lim = list()
for i in range(0,len(output_df_F)):
    lower_limit = output_df_F["sample_mean"][i] -␣
 ↪((output_df_F["approx_std"][i])*1.96); lower_lim.append(round(lower_limit,2))
    upper_limit = output_df_F["sample_mean"][i] +␣
 ↪((output_df_F["approx_std"][i])*1.96); upper_lim.append(round(upper_limit,2))

#appending values into the dataset
output_df_F["lower_limit"] = pd.Series(lower_lim)
output_df_F["upper_limit"] = pd.Series(upper_lim)
```

[46]:
```
output_df_F
```

[46]:

	name	purchase_mean	sample_mean	purchase_std	sample_std	approx_std	lower_limit	upper_limit
0	total_sample_100_total_mean_50	8734.57	8939.48	4767.23	705.42	674.19	7618.07	10260.89
1	total_sample_100_total_mean_100	8734.57	8743.40	4767.23				
2	total_sample_250_total_mean_50	8734.57	8663.45	4767.23				
3	total_sample_250_total_mean_100	8734.57	8733.91	4767.23				
4	total_sample_500_total_mean_50	8734.57	8712.33	4767.23				
5	total_sample_500_total_mean_100	8734.57	8730.28	4767.23				

```
1      537.88      476.72      7809.03      9677.77
2      686.65      674.19      7342.04      9984.86
3      467.51      476.72      7799.54      9668.28
4      677.06      674.19      7390.92     10033.74
5      485.32      476.72      7795.91      9664.65
```

[47]:
```
lower_lim = list(); upper_lim = list()
for i in range(0,len(output_df_M)):
    lower_limit = output_df_M["sample_mean"][i] -␣
 ↪((output_df_M["approx_std"][i])*1.96); lower_lim.append(round(lower_limit,2))
    upper_limit = output_df_M["sample_mean"][i] +␣
 ↪((output_df_M["approx_std"][i])*1.96); upper_lim.append(round(upper_limit,2))

#appending values into the dataset
output_df_M["lower_limit"] = pd.Series(lower_lim)
output_df_M["upper_limit"] = pd.Series(upper_lim)
```

[48]:
```
output_df_M
```

[48]:
```
                              name  purchase_mean  sample_mean  purchase_std  \
0   total_sample_100_total_mean_50        9437.53      9348.49       5092.19
1  total_sample_100_total_mean_100        9437.53      9472.32       5092.19
2   total_sample_250_total_mean_50        9437.53      9446.45       5092.19
3  total_sample_250_total_mean_100        9437.53      9440.49       5092.19
4   total_sample_500_total_mean_50        9437.53      9427.64       5092.19
5  total_sample_500_total_mean_100        9437.53      9452.02       5092.19

   sample_std  approx_std  lower_limit  upper_limit
0      764.53      720.14      7937.02     10759.96
1      530.62      509.22      8474.25     10470.39
2      673.35      720.14      8034.98     10857.92
3      510.15      509.22      8442.42     10438.56
4      697.03      720.14      8016.17     10839.11
5      516.26      509.22      8453.95     10450.09
```

Getting lower and upper limit of 99% confidence interval with known standard deviation of population for Female and Male Respectively

[49]:
```
lower_lim = list(); upper_lim = list()
for i in range(0,len(output_df_F)):
    lower_limit = output_df_F["sample_mean"][i] -␣
 ↪((output_df_F["approx_std"][i])*2.576); lower_lim.
 ↪append(round(lower_limit,2))
    upper_limit = output_df_F["sample_mean"][i] +␣
 ↪((output_df_F["approx_std"][i])*2.576); upper_lim.
 ↪append(round(upper_limit,2))
```

```
#appending values into the dataset
output_df_F["lower_limit"] = pd.Series(lower_lim)
output_df_F["upper_limit"] = pd.Series(upper_lim)
output_df_F
```

[49]:

	name	purchase_mean	sample_mean	purchase_std	sample_std	approx_std	lower_limit	upper_limit
0	total_sample_100_total_mean_50	8734.57	8939.48	4767.23	705.42	674.19	7202.77	10676.19
1	total_sample_100_total_mean_100	8734.57	8743.40	4767.23	537.88	476.72	7515.37	9971.43
2	total_sample_250_total_mean_50	8734.57	8663.45	4767.23	686.65	674.19	6926.74	10400.16
3	total_sample_250_total_mean_100	8734.57	8733.91	4767.23	467.51	476.72	7505.88	9961.94
4	total_sample_500_total_mean_50	8734.57	8712.33	4767.23	677.06	674.19	6975.62	10449.04
5	total_sample_500_total_mean_100	8734.57	8730.28	4767.23	485.32	476.72	7502.25	9958.31

[50]:
```
lower_lim = list(); upper_lim = list()
for i in range(0,len(output_df_M)):
    lower_limit = output_df_M["sample_mean"][i] -␣
 ↪((output_df_M["approx_std"][i])*2.576); lower_lim.
 ↪append(round(lower_limit,2))
    upper_limit = output_df_M["sample_mean"][i] +␣
 ↪((output_df_M["approx_std"][i])*2.576); upper_lim.
 ↪append(round(upper_limit,2))

#appending values into the dataset
output_df_M["lower_limit"] = pd.Series(lower_lim)
output_df_M["upper_limit"] = pd.Series(upper_lim)
output_df_M
```

[50]:

	name	purchase_mean	sample_mean	purchase_std	sample_std	approx_std	lower_limit	upper_limit
0	total_sample_100_total_mean_50	9437.53	9348.49	5092.19	764.53	720.14	7493.41	11203.57
1	total_sample_100_total_mean_100	9437.53	9472.32	5092.19	530.62	509.22	8160.57	10784.07
2	total_sample_250_total_mean_50	9437.53	9446.45	5092.19	673.35	720.14	7591.37	11301.53
3	total_sample_250_total_mean_100	9437.53	9440.49	5092.19				
4	total_sample_500_total_mean_50	9437.53	9427.64	5092.19				
5	total_sample_500_total_mean_100	9437.53	9452.02	5092.19				

```
3      510.15      509.22      8128.74     10752.24
4      697.03      720.14      7572.56     11282.72
5      516.26      509.22      8140.27     10763.77
```

Observations

- Dataframe has 550068 rows and 10 columns and there was no Null Values
- Age, Gender, Occupation, City Category, Marital Status and Product Category are Categorical Variables
- Most users lie between the Age of 26-45 Yrs
- Data points for 0-17 and 55+ Age group is skewed
- Product Category 1,5 8 are most popular but Product Category 6, 7, 9, 10 generate more revenue
- Most of the Purchase happens in range of 5000−1000 Dollars range
- Male Members purchase more than Female Members but the Value of Purchase are equally distributed. So ARPC for Female Customores are higher
- There is almost equal share of purchase from all City Categories
- People with Occupation Code 0, 4, 7 Purchase More, while Occupation Code 7, 8, 14, 15 and 17 have higher basket size
- For all Age group, between male & Female, Money Spent to purchase if uniformly concentrated, but Female with 55+ Age is more likely to purchase in high amount, especially ones with Occupation code 5 and 20
- For Female Customers, Purchase Amount Lies between 7475.92$ -9931.98$ dollars while for Male Customers it ranges between 8138.04$ -10761.54$ dollars with 99% confidence Interval

Recommendations

- More Youth Oriented Products can increase sales
- More products in range of 5000-8000 Dollars can boost sales
- Basked size for Females is higher, so a strong product recommendation system for Females can drive the sales higher
- Male Customers have high frequency of shopping but small basket size. Recommendation system built to suggest daily use, low / mid value products can prove to be better for sales
- Targeted Marketing for Product Category 6, 7, 9, 10 will generate more revenue
- Customers with Occupation Code 7, 8, 14, 15 and 17 have higher basket size, special discounts can be given to these customers
- Female Members with 50+ Age, specially with Occupation code 5 and 20 can be given special offers

This will help in boosting the Walmart's Revenue

[ ]: