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

Defining Problem Statement and Analyzing basic metrics

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).

Loading Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
```

```
import math
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv(r"C:\Users\SYEDA TAYABA\OneDrive\Documents\DATASETS\Walmart.csv")
df.head()
```

Out[2]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mar
0	1000001	P00069042	F	0-17	10	A	2	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	

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

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

In [4]:
```
print(f"Number of rows: {df.shape[0]:,} \nNumber of columns: {df.shape[1]}")
```

```
Number of rows: 550,068
Number of columns: 10
```

In [5]:
```
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
```

Change the data types of - user_id, Occupation, Marital_Status, Product_Category

In [6]:
```
cols = ['User_ID','Occupation', 'Marital_Status', 'Product_Category']
df[cols] = df[cols].astype('object')
```

In [7]:
```
df.dtypes
```

Out[7]:
```
User_ID                     object
Product_ID                  object
Gender                      object
Age                         object
Occupation                  object
City_Category               object
Stay_In_Current_City_Years  object
Marital_Status              object
Product_Category            object
Purchase                     int64
dtype: object
```

statistical summary:

In [8]:
```
df.describe()
```

Out[8]:

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

In [9]:
```
df.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	max
Purchase	550068.0	9263.968713	5023.065394	12.0	5823.0	8047.0	12054.0	23961.0

In [10]:
```
df.describe(include ="all")
```

Out[10]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Y
count	550068.0	550068	550068	550068	550068.0	550068	550
unique	5891.0	3631	2	7	21.0	3	
top	1001680.0	P00265242	M	26-35	4.0	B	
freq	1026.0	1880	414259	219587	72308.0	231173	193
mean	NaN	NaN	NaN	NaN	NaN	NaN	I
std	NaN	NaN	NaN	NaN	NaN	NaN	I
min	NaN	NaN	NaN	NaN	NaN	NaN	I
25%	NaN	NaN	NaN	NaN	NaN	NaN	I
50%	NaN	NaN	NaN	NaN	NaN	NaN	I
75%	NaN	NaN	NaN	NaN	NaN	NaN	I
max	NaN	NaN	NaN	NaN	NaN	NaN	I

In [11]:
```
# Checking Null Values

df.isnull().sum()
```

Out[11]:
```
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
```

Non-Graphical Analysis: Value counts and unique attributes

In [12]:
```
df.nunique()
```

Out[12]:
```
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
```

How many users are there in the Dataset ?

In [13]:
```
df['User_ID'].nunique()
```

Out[13]:
```
5891
```

How many products are there?

In [14]:
```
df['Product_ID'].nunique()
```

Out[14]:
```
3631
```

Value counts

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

Out[15]:
```
Gender
F    1666
M    4225
Name: User_ID, dtype: int64
```

In [16]:
```
categorical_cols = ['Gender']
round(df[categorical_cols].melt().groupby(['variable', 'value'])[['value']].count(
```

Out[16]:

		value
variable	**value**	
Gender	**F**	24.69
	M	75.31

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

Out[17]:
```
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
```

In [18]:
```
categorical_cols = ['Age']
round(df[categorical_cols].melt().groupby(['variable', 'value'])[['value']].count(
```

Out[18]:

		value
variable	**value**	
Age	**0-17**	2.75
	18-25	18.12
	26-35	39.92
	36-45	20.00
	46-50	8.31
	51-55	7.00
	55+	3.91

In [19]:
```
df_age=df.groupby(["User_ID"])["Age"].unique()
(df_age.value_counts()/len(df_age))*100
```

Out[19]:
```
[26-35]    34.849771
[36-45]    19.809879
[18-25]    18.146325
[46-50]     9.013750
[51-55]     8.164997
[55+]       6.314717
[0-17]      3.700560
Name: Age, dtype: float64
```

In [20]:
```
df.groupby(['City_Category'])['User_ID'].nunique()
```

Out[20]:
```
City_Category
A    1045
B    1707
C    3139
Name: User_ID, dtype: int64
```

In [21]:
```
categorical_cols = ['City_Category']
round(df[categorical_cols].melt().groupby(['variable', 'value'])[['value']].count(
```

Out[21]:

		value
variable	**value**	
City_Category	**A**	26.85
	B	42.03
	C	31.12

In [22]:
```
df_city = df.groupby(['User_ID'])['City_Category'].unique()
(df_city.value_counts()/len(df_city))*100
```

Out[22]:
```
[C]    53.284672
[B]    28.976405
[A]    17.738924
Name: City_Category, dtype: float64
```

In [23]:
```
df.groupby(['Stay_In_Current_City_Years'])['User_ID'].nunique()
```

Out[23]:
```
Stay_In_Current_City_Years
0      772
1     2086
2     1145
3      979
4+     909
Name: User_ID, dtype: int64
```

In [24]:
```
df_city = df.groupby(['User_ID'])['Stay_In_Current_City_Years'].unique()
round((df_city.value_counts()/len(df_city))*100, 2)
```

Out[24]:
```
[1]     35.41
[2]     19.44
[3]     16.62
[4+]    15.43
[0]     13.10
Name: Stay_In_Current_City_Years, dtype: float64
```

In [25]:
```
categorical_cols = ['Stay_In_Current_City_Years']
round(df[categorical_cols].melt().groupby(['variable', 'value'])[['value']].count(
```

Out[25]:

		value
variable	**value**	
Stay_In_Current_City_Years	**0**	13.53
	1	35.24
	2	18.51
	3	17.32
	4+	15.40

In [26]:
```
df.groupby(['Marital_Status'])['User_ID'].nunique()
```

Out[26]:
```
Marital_Status
0    3417
1    2474
Name: User_ID, dtype: int64
```

In [27]:
```
categorical_cols = ['Marital_Status']
round(df[categorical_cols].melt().groupby(['variable', 'value'])[['value']].count(
```

Out[27]:

		value
variable	**value**	
Marital_Status	**0**	59.03
	1	40.97

In [28]:
```
df_marital=df.groupby(["User_ID"])["Marital_Status"].unique()
(df_marital.value_counts()/len(df_marital))*100
```

Out[28]:
```
[0]    58.003735
[1]    41.996265
Name: Marital_Status, dtype: float64
```

In [29]:
```
categorical_cols = ['Product_Category']
round(df[categorical_cols].melt().groupby(['variable', 'value'])[['value']].count(
```

Out[29]:

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

In [30]:
```
categorical_cols = ['Occupation']
round(df[categorical_cols].melt().groupby(['variable', 'value'])[['value']].count(
```

Out[30]:

		value
variable	**value**	
Occupation	**0**	12.66
	1	8.62
	2	4.83
	3	3.21
	4	13.15
	5	2.21
	6	3.70
	7	10.75
	8	0.28
	9	1.14
	10	2.35
	11	2.11
	12	5.67
	13	1.40
	14	4.96
	15	2.21
	16	4.61
	17	7.28
	18	1.20
	19	1.54
	20	6.10

Observations

- 78% of the users are between the age 18-50 (40%: 26-35, 18%: 18-25, 20%: 36-45)
- 75% of the users are Male and 25% are Female
- 60% Single, 40% Married
- 35% Staying in the city from 1 year, 18% from 2 years, 17% from 3 years
- There are total of 20 product categories.
- There are 20 differnent types of occupations in the city

Crosstab or Contingency table

In [31]:
```
pd.crosstab(index  = df["City_Category"],columns=df["Age"],margins= True, normaliz
```

Out[31]:

Age	0-17	18-25	26-35	36-45	46-50	51-55	55+
City_Category							
A	1.722177	18.639995	49.922150	18.018549	5.149607	4.128757	2.418765
B	2.351053	18.707635	39.617083	20.589775	8.827155	7.674339	2.232960
C	4.161238	16.870454	31.697386	20.913101	10.333285	8.564919	7.459617
All	2.745479	18.117760	39.919974	19.999891	8.308246	6.999316	3.909335

In [32]:
```
pd.crosstab(index  = df["City_Category"],columns=df["Stay_In_Current_City_Years"],
```

Out[32]:

Stay_In_Current_City_Years	0	1	2	3	4+
City_Category					
A	16.367452	33.377335	18.354996	16.791227	15.108990
B	12.409321	36.082501	18.069584	18.467122	14.971472
C	12.579524	35.696217	19.250475	16.234847	16.238937
All	13.525237	35.235825	18.513711	17.322404	15.402823

In [33]:
```
pd.crosstab(index  = df["City_Category"],columns=df["Marital_Status"],margins= Tru
```

Out[33]:

Marital_Status	0	1
City_Category		
A	61.720146	38.279854
B	59.142287	40.857713
C	56.571929	43.428071
All	59.034701	40.965299

In [34]:
```
pd.crosstab(index  = df["Product_Category"],columns=df["Gender"],margins= True, no
```

Out[34]:

Gender	F	M
Product_Category		
1	17.688669	82.311331
2	23.709353	76.290647
3	29.713551	70.286449
4	30.962307	69.037693
5	27.801077	72.198923
6	22.275970	77.724030
7	25.342650	74.657350
8	29.456221	70.543779
9	17.073171	82.926829
10	22.673171	77.326829
11	19.512496	80.487504
12	38.814289	61.185711
13	26.347090	73.652910
14	40.906106	59.093894
15	16.629571	83.370429
16	24.440374	75.559626
17	10.726644	89.273356
18	12.224000	87.776000
19	28.134747	71.865253
20	28.352941	71.647059
All	24.689493	75.310507

In [35]:
```
pd.crosstab(index  = df["City_Category"],columns=df["Gender"],margins= True, norma
```

Out[35]:

Gender	F	M
City_Category		
A	24.170051	75.829949
B	25.001190	74.998810
C	24.716810	75.283190
All	24.689493	75.310507

Non-Visual Analysis of Bivariate

In [36]:
```
# checking how gender are contributing towards total purchase amount
df2 = pd.DataFrame(df.groupby(["Gender"])[["Purchase"]].sum())
df2['percent'] = (df2['Purchase'] /
                  df2['Purchase'].sum())*100
df2
```

Out[36]:

Gender	Purchase	percent
F	1186232642	23.278576
M	3909580100	76.721424

In [37]:
```
# checking how purchase value are spread among different age categories
df2 = pd.DataFrame(df.groupby(["Age"])[["Purchase"]].sum())
df2['percent'] = (df2['Purchase'] /
                  df2['Purchase'].sum())*100
df2
```

Out[37]:

Age	Purchase	percent
0-17	134913183	2.647530
18-25	913848675	17.933325
26-35	2031770578	39.871374
36-45	1026569884	20.145361
46-50	420843403	8.258612
51-55	367099644	7.203947
55+	200767375	3.939850

In [38]:
```
df2 = pd.DataFrame(df.groupby(["Marital_Status"])[["Purchase"]].sum())
df2['percent'] = (df2['Purchase'] /
                  df2['Purchase'].sum())*100
df2
```

Out[38]:

Marital_Status	Purchase	percent
0	3008927447	59.047057
1	2086885295	40.952943

In [39]:
```
df2 = pd.DataFrame(df.groupby(["City_Category"])[["Purchase"]].sum())
df2['percent'] = (df2['Purchase'] /
                  df2['Purchase'].sum())*100
df2
```

Out[39]:

City_Category	Purchase	percent
A	1316471661	25.834381
B	2115533605	41.515136
C	1663807476	32.650483

In [40]:
```
# Users with highest number of purchases
df.groupby(['User_ID'])['Purchase'].count().nlargest(10)
```

Out[40]:
```
User_ID
1001680    1026
1004277     979
1001941     898
1001181     862
1000889     823
1003618     767
1001150     752
1001015     740
1005795     729
1005831     727
Name: Purchase, dtype: int64
```

In [41]:
```
# Users with highest purchases amount
df.groupby(['User_ID'])['Purchase'].sum().nlargest(10)
```

Out[41]:
```
User_ID
1004277    10536909
1001680     8699596
1002909     7577756
1001941     6817493
1000424     6573609
1004448     6566245
1005831     6512433
1001015     6511314
1003391     6477160
1001181     6387961
Name: Purchase, dtype: int64
```

Observation

- The users with high number of purchases contribute more to the purchase amount.
- we can see there are few users who are not in the list of top 10 purchase counts, but they are present in the top 10 purchase amounts, like User_ids 1004448 , 1003391 etc.. are top 10 in purchase amounts but not in purchase counts
- Also, the user 1004277 with lesser purchase count(979) has a much higher purchase amount than the user(1001680) with top purchase count.

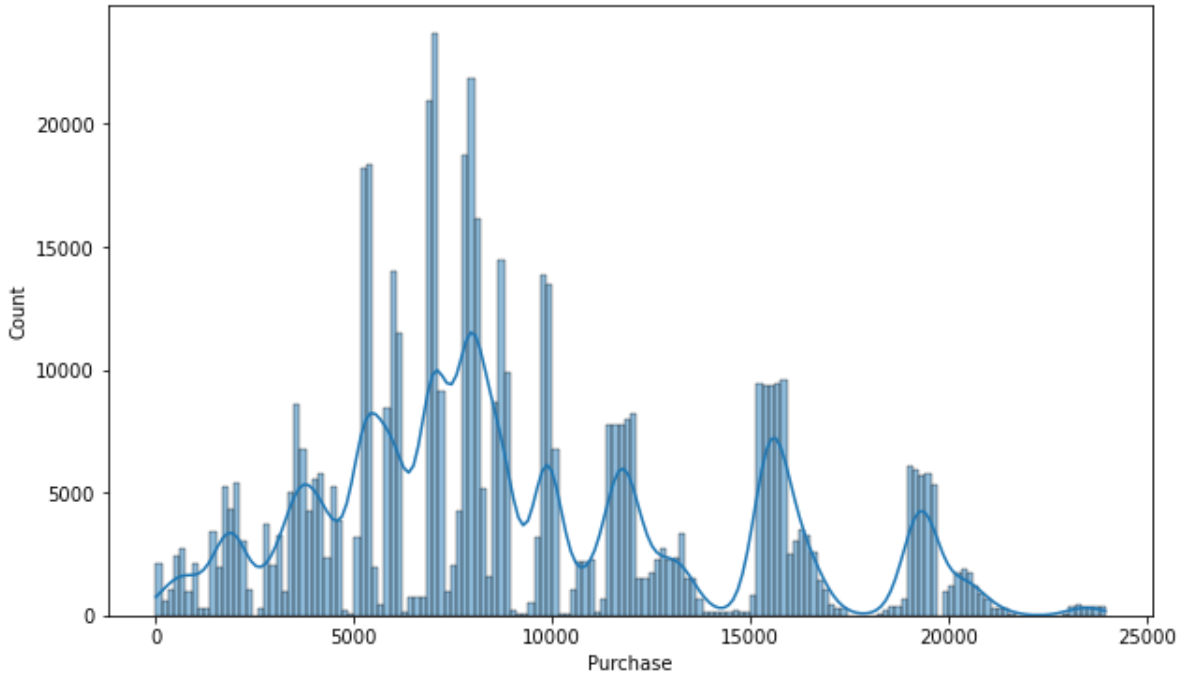
In [42]:
```
df2 = pd.DataFrame(df.groupby(["Product_Category"])[["Purchase"]].sum())
df2['percent'] = (df2['Purchase'] /
                  df2['Purchase'].sum())*100
df2
```

Out[42]:

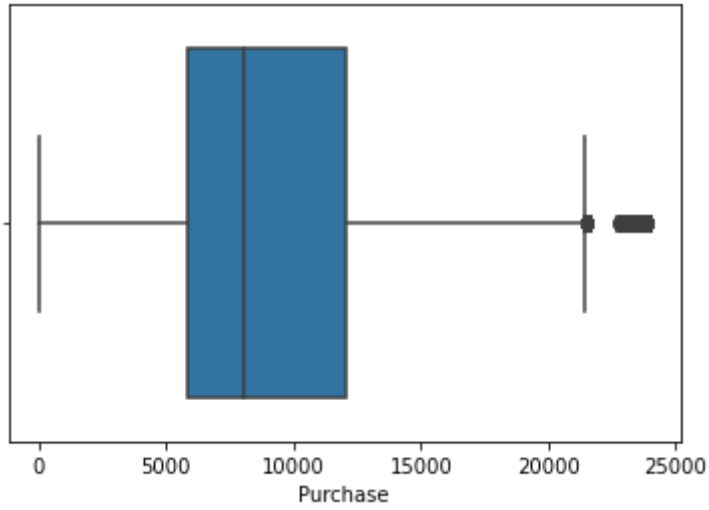
Product_Category	Purchase	percent
1	1910013754	37.482024
2	268516186	5.269350
3	204084713	4.004949
4	27380488	0.537313
5	941835229	18.482532
6	324150302	6.361111
7	60896731	1.195035
8	854318799	16.765114
9	6370324	0.125011
10	100837301	1.978827
11	113791115	2.233032
12	5331844	0.104632
13	4008601	0.078665
14	20014696	0.392767
15	92969042	1.824420
16	145120612	2.847840
17	5878699	0.115363
18	9290201	0.182310
19	59378	0.001165
20	944727	0.018539

Visual Analysis - Univariate¶

In [43]:
```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Purchase', kde=True)
plt.show()
```



In [44]:
```
sns.boxplot(data=df, x ='Purchase')
plt.show()
```

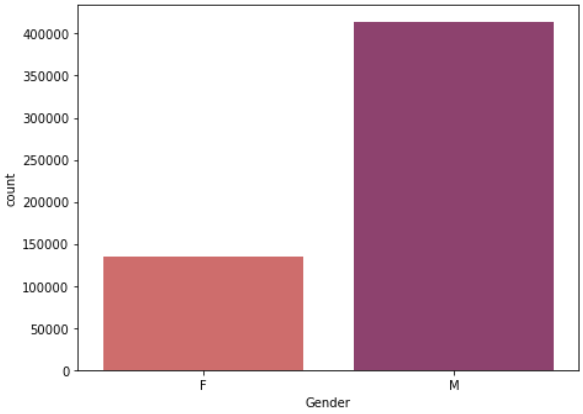


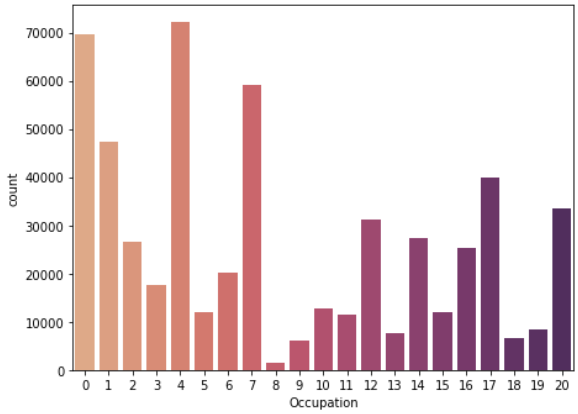
Understanding the distribution of data for the categorical variables

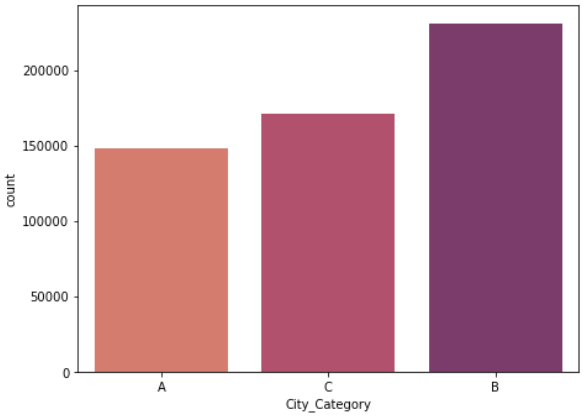
- Gender
- Age
- Occupation
- City_Category
- Stay_In_Current_City_Years
- Marital_Status
- Product_Category

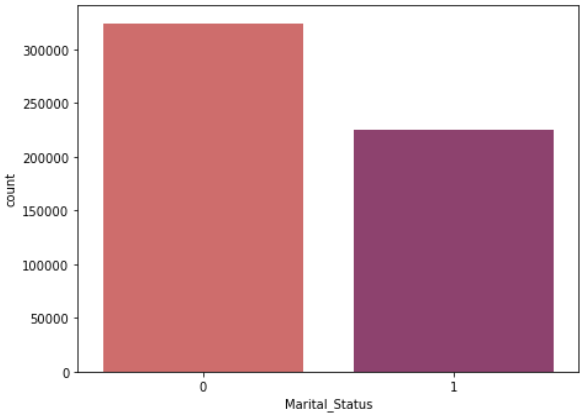
In [45]:
```
categorical_cols = ['Gender', 'Occupation','City_Category','Marital_Status','Produ

fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(16, 12))
sns.countplot(data=df, x='Gender', ax=axs[0,0], palette = 'flare')
sns.countplot(data=df, x='Occupation', ax=axs[0,1], palette = 'flare')
sns.countplot(data=df, x='City_Category', ax=axs[1,0], palette = 'flare')
sns.countplot(data=df, x='Marital_Status', ax=axs[1,1], palette = 'flare')
plt.show()
```

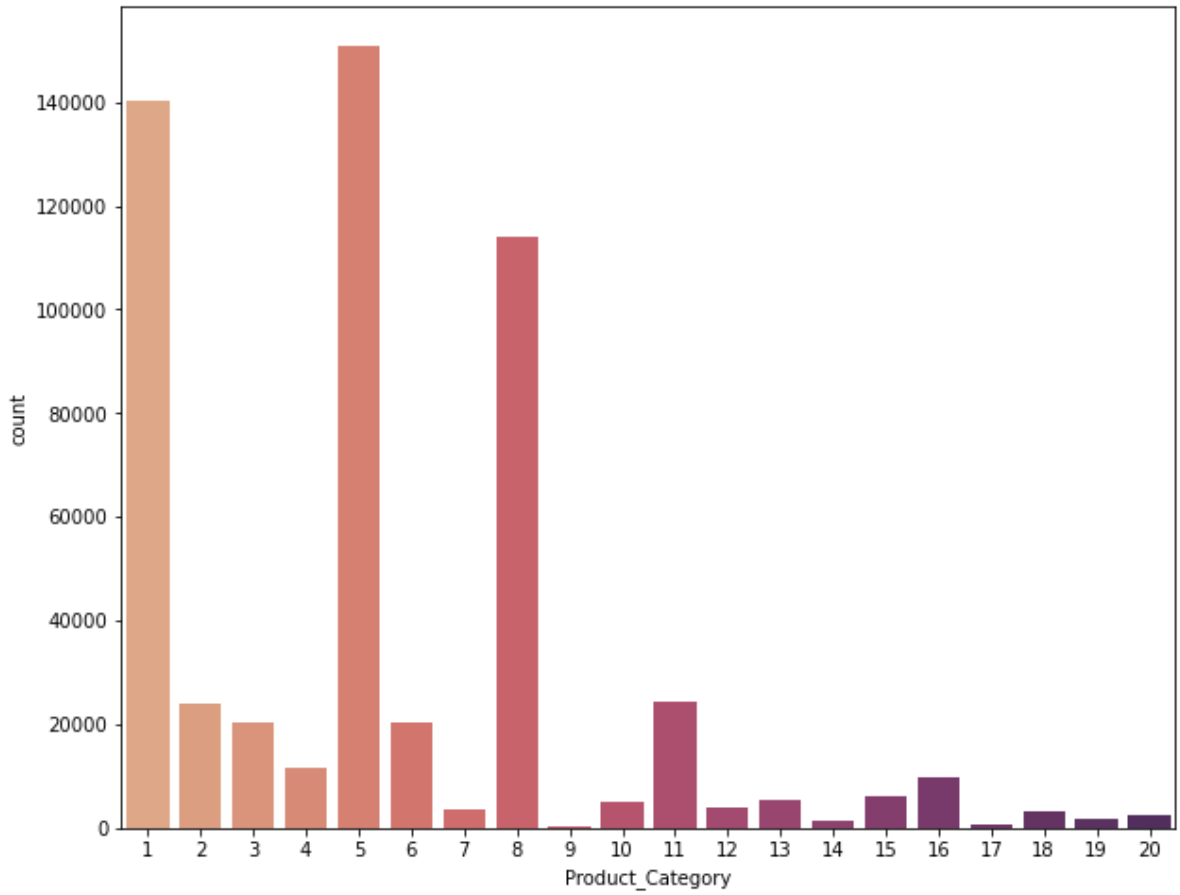
```
plt.figure(figsize=(10, 8))
sns.countplot(data=df, x='Product_Category', palette = 'flare')
plt.show()
```









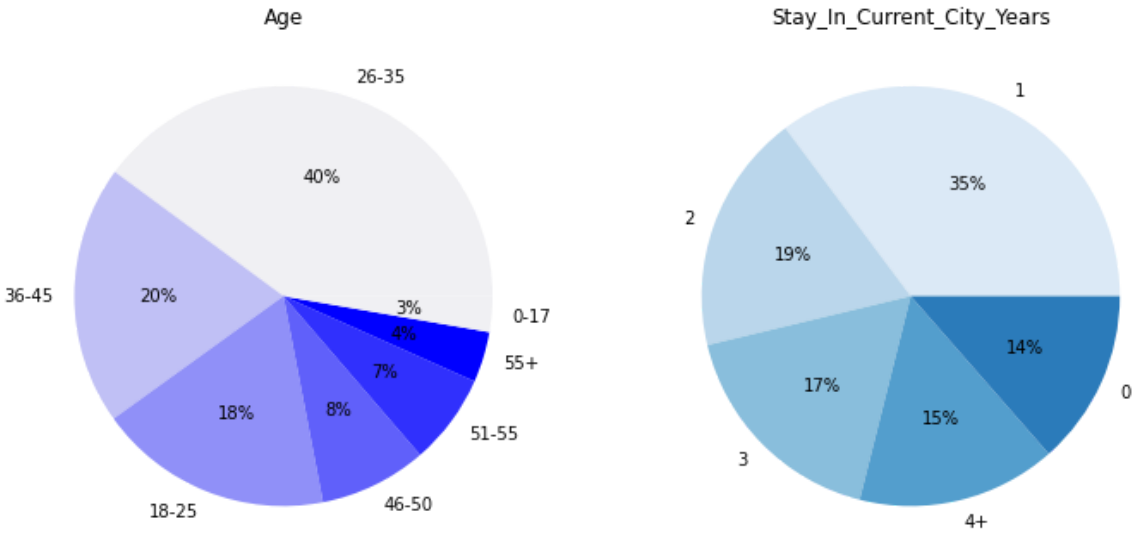


Observations

- Most of the users are Male

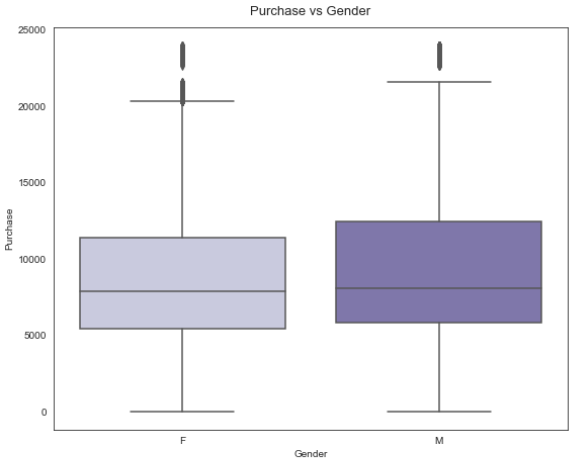
- There are 20 different types of Occupation and Product_Category
- More users belong to B City_Category
- More users are Single as compare to Married
- Product_Category - 1, 5, 8, & 11 have highest purchasing frequency.

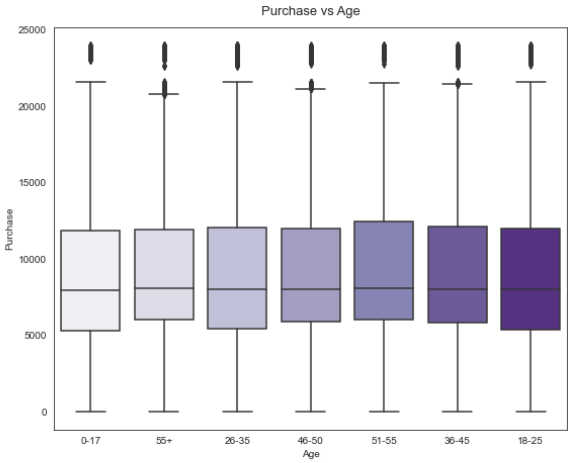
In [46]:
```
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(12, 8))

data = df['Age'].value_counts(normalize=True)*100
palette_color = sns.color_palette('light:b')
axs[0].pie(x=data.values, labels=data.index, autopct='%.0f%%', colors=palette_colo
axs[0].set_title("Age")

data = df['Stay_In_Current_City_Years'].value_counts(normalize=True)*100
palette_color = sns.color_palette('Blues')
axs[1].pie(x=data.values, labels=data.index, autopct='%.0f%%', colors=palette_colo
axs[1].set_title("Stay_In_Current_City_Years")

plt.show()
```

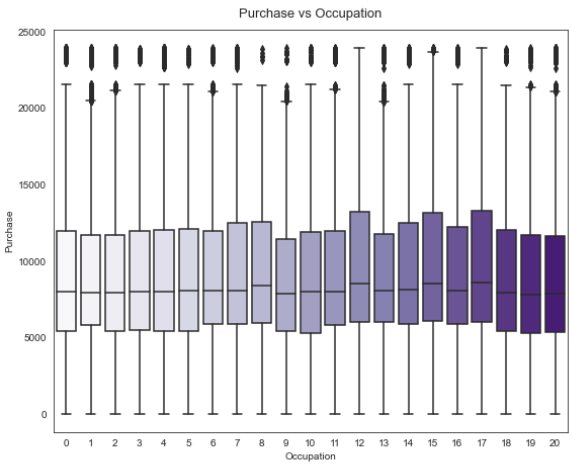


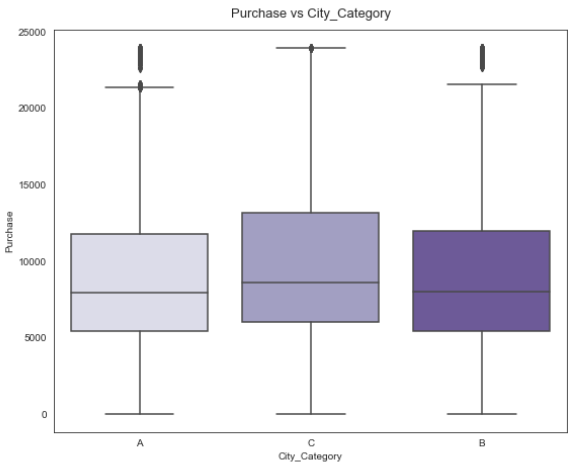
BIVARIATE ANALYSIS

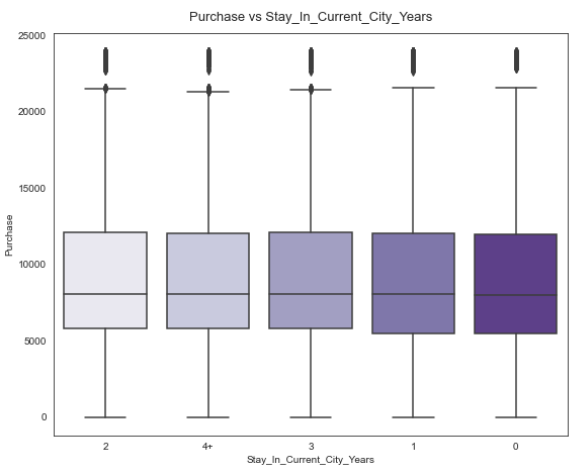
In [47]:
```
attrs = ['Gender', 'Age', 'Occupation', 'City_Category', 'Stay_In_Current_City_Yea
sns.set_style("white")

fig, axs = plt.subplots(nrows=3, ncols=2, figsize=(20, 16))
fig.subplots_adjust(top=1.3)
count = 0
for row in range(3):
    for col in range(2):
        sns.boxplot(data=df, y='Purchase', x=attrs[count], ax=axs[row, col], palet
        axs[row,col].set_title(f"Purchase vs {attrs[count]}", pad=12, fontsize=13)
        count += 1
plt.show()
plt.figure(figsize=(10, 8))
sns.boxplot(data=df, y='Purchase', x=attrs[-1], palette='Purples')
plt.show()
```

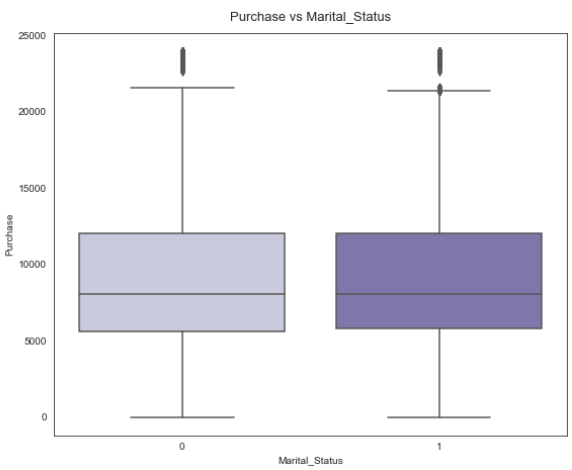

Purchase vs Gender
Purchase
Gender
F
M

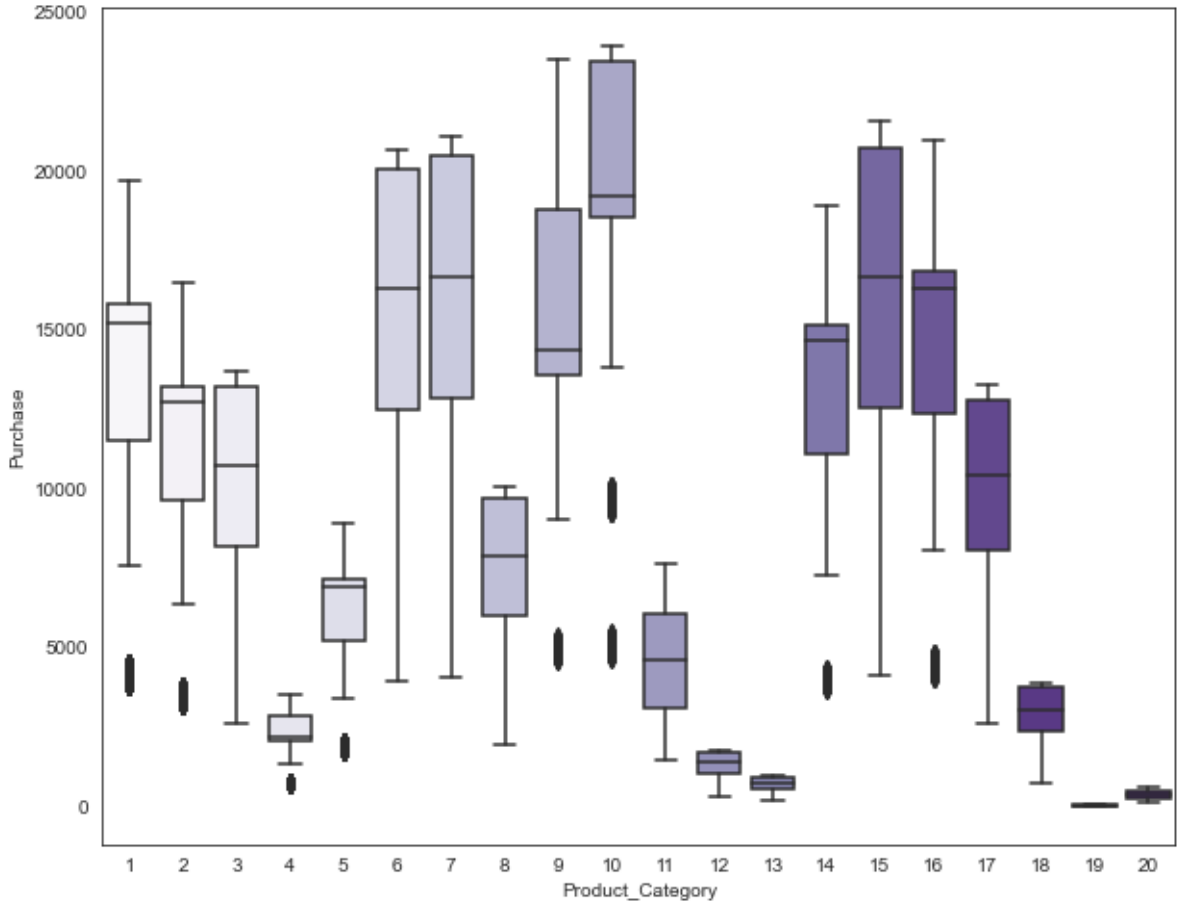

Purchase vs Age
Purchase
Age
0-17
55+
26-35
46-50
51-55
36-45
18-25


Purchase vs Occupation
Purchase
Occupation


Purchase vs City_Category
Purchase
City_Category
A
C
B

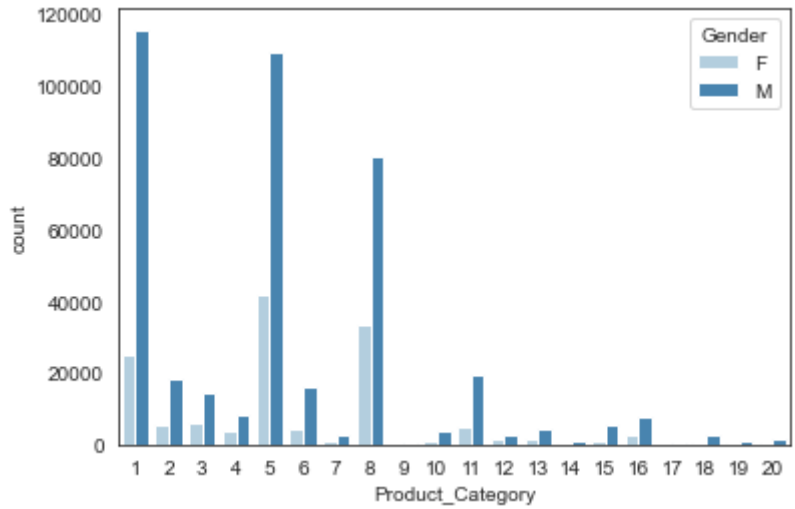

Purchase vs Stay_In_Current_City_Years
Purchase
Stay_In_Current_City_Years
2
4+
3
1
0


Purchase vs Marital_Status
Purchase
Marital_Status



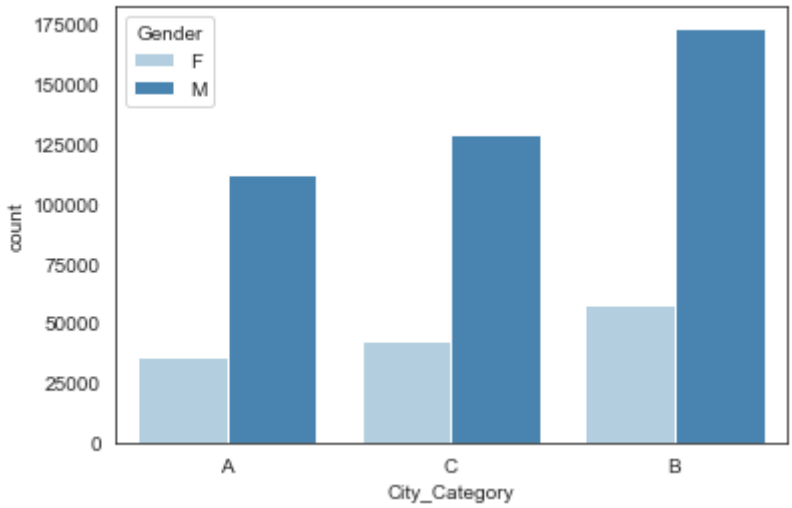
In [48]:
```
sns.countplot(data = df, x ='Product_Category', hue = 'Gender', palette = 'Blues')
```

Out[48]:
```
<AxesSubplot:xlabel='Product_Category', ylabel='count'>
```



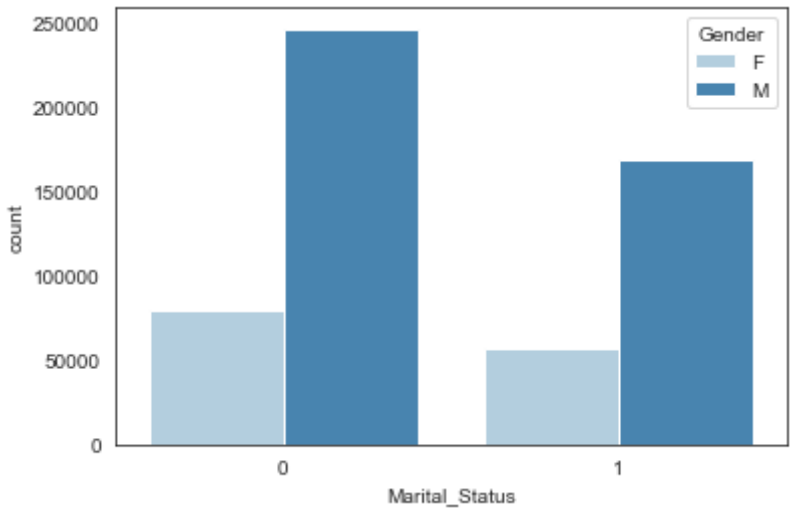
In [49]:
```
sns.countplot(data = df, x ='City_Category', hue = 'Gender', palette = 'Blues')
```

Out[49]:
```
<AxesSubplot:xlabel='City_Category', ylabel='count'>
```



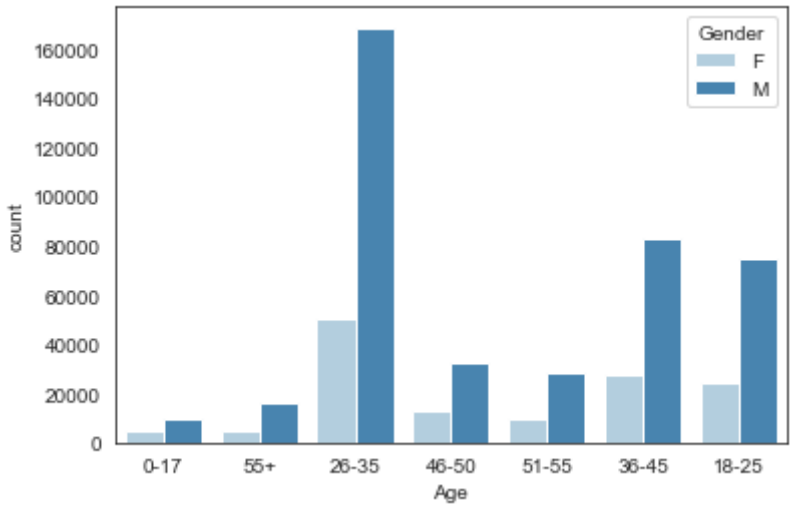
In [50]:
```
sns.countplot(data = df, x ='Marital_Status', hue = 'Gender', palette = 'Blues')
```

Out[50]:
```
<AxesSubplot:xlabel='Marital_Status', ylabel='count'>
```



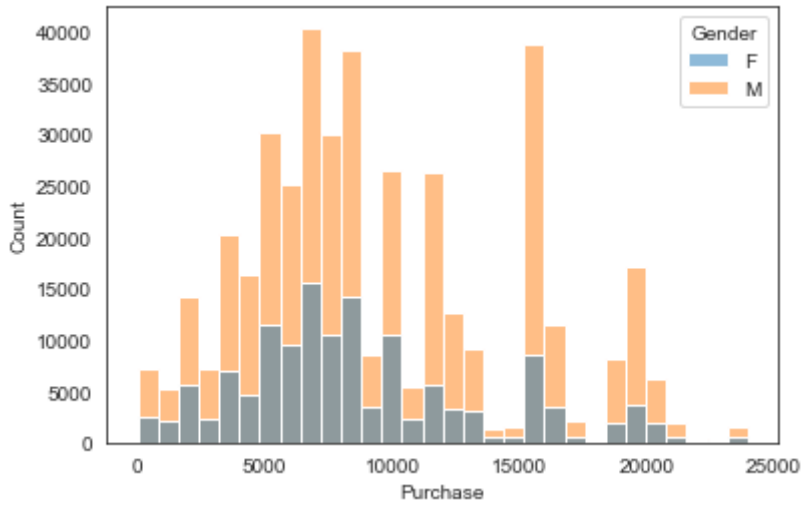
In [51]:
```
sns.countplot(data = df, x = 'Age', hue = 'Gender', palette = 'Blues')
```

Out[51]:
```
<AxesSubplot:xlabel='Age', ylabel='count'>
```



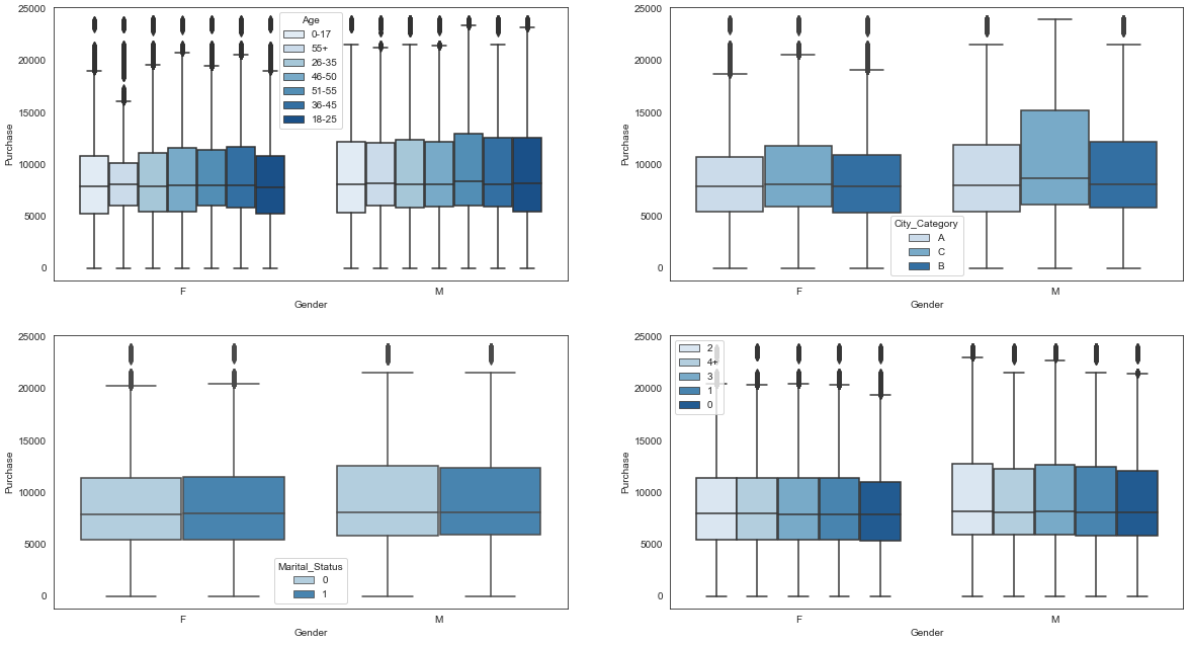
In [52]:
```
sns.histplot(data = df, x = 'Purchase', hue = 'Gender', bins = 30)
```

Out[52]:
```
<AxesSubplot:xlabel='Purchase', ylabel='Count'>
```



MULTIVARIATE ANALYSIS

In [53]:
```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20, 6))
fig.subplots_adjust(top=1.5)
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Age', palette='Blues', ax=axs[(
sns.boxplot(data=df, y='Purchase', x='Gender', hue='City_Category', palette='Blues

sns.boxplot(data=df, y='Purchase', x='Gender', hue='Marital_Status', palette='Blues
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Stay_In_Current_City_Years', pa
axs[1,1].legend(loc='upper left')
plt.show()
```

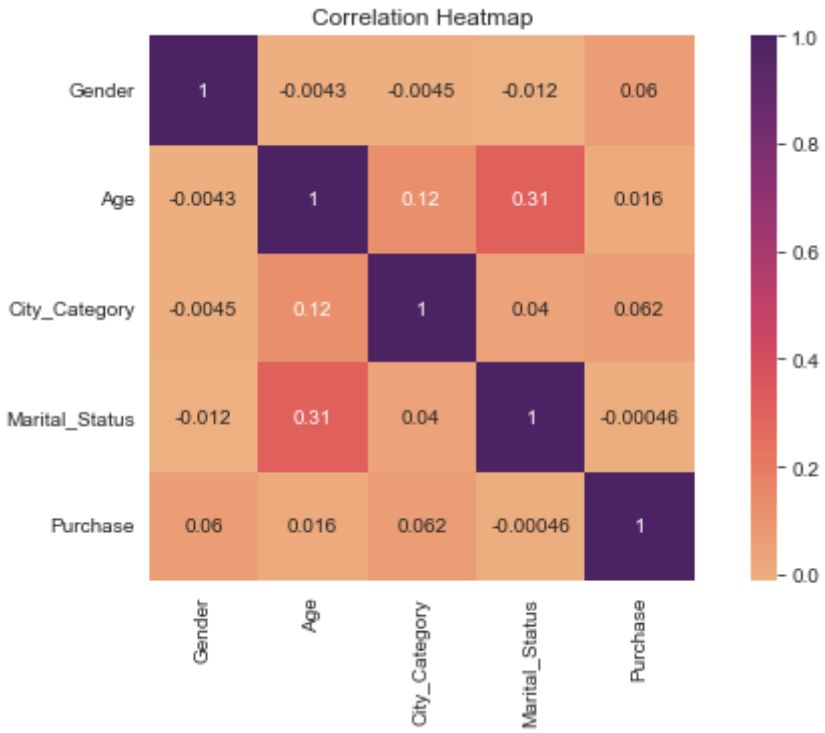


In [54]:
```
df1 = df.copy()
df1 = df[["Gender", "Age","City_Category","Marital_Status","Purchase"]]
df1["Gender"] = df1["Gender"].astype("category").cat.codes
df1["Age"] = df["Age"].astype("category").cat.codes
df1["City_Category"] = df["City_Category"].astype("category").cat.codes
df1["Marital_Status"] = df["Marital_Status"].astype("category").cat.codes
df1.corr()
```

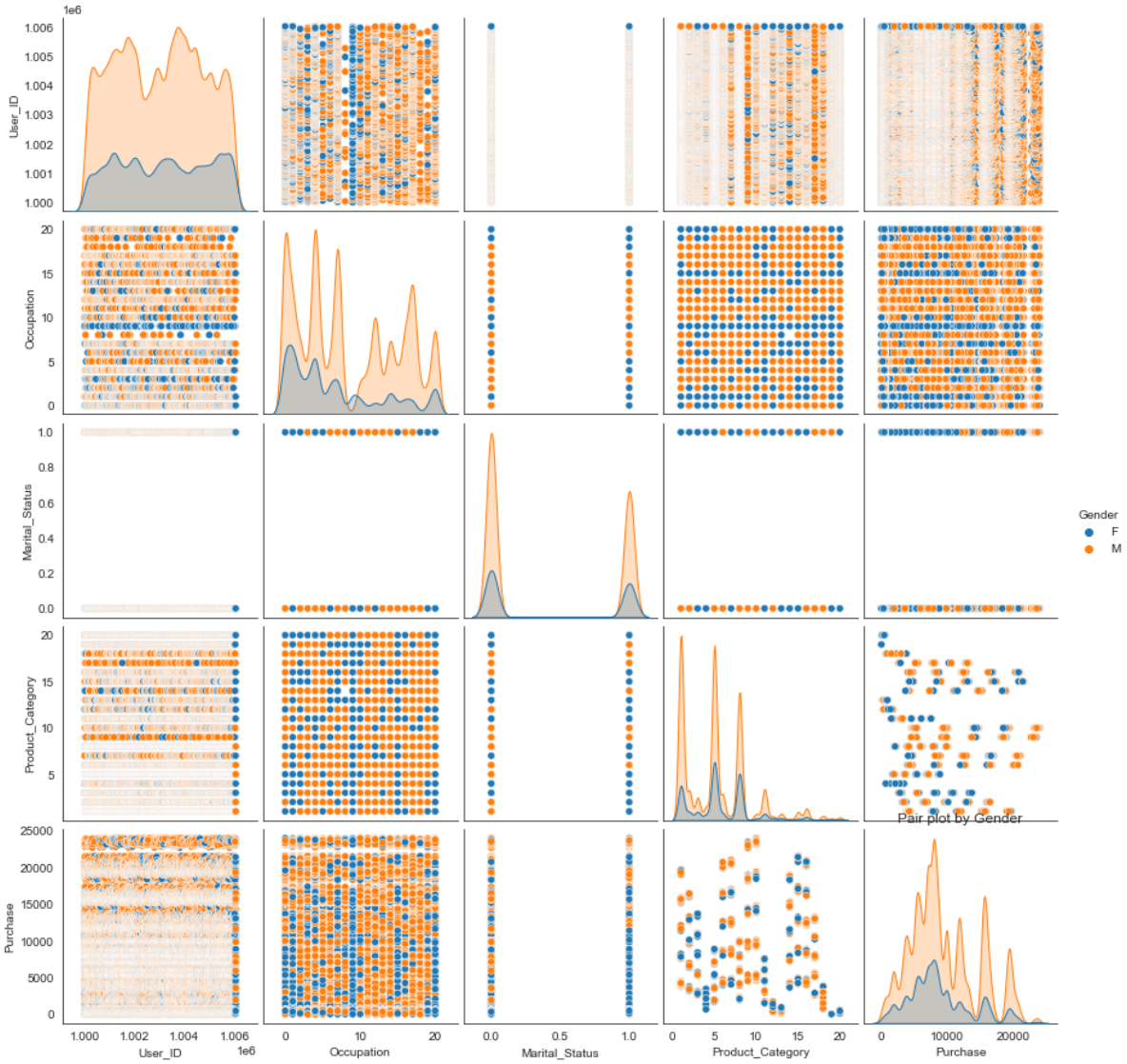
Out[54]:

	Gender	Age	City_Category	Marital_Status	Purchase
Gender	1.000000	-0.004262	-0.004515	-0.011603	0.060346
Age	-0.004262	1.000000	0.123079	0.311738	0.015839
City_Category	-0.004515	0.123079	1.000000	0.039790	0.061914
Marital_Status	-0.011603	0.311738	0.039790	1.000000	-0.000463
Purchase	0.060346	0.015839	0.061914	-0.000463	1.000000

In [55]:
```
plt.figure(figsize = (10,5))
plt.title('Correlation Heatmap')
sns.heatmap(data = df1.corr(),annot = True, cmap = 'flare', square = True)
plt.show()
```



In [89]:
```
sns.pairplot(df,hue="Gender",diag_kind="kde")
plt.title("Pair plot by Gender")
plt.show()
```



Missing Value & Outlier Detection

In [57]:
```
# checking null values
df.isnull().sum()
```

Out[57]:
```
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
```

Observations

- There are no missing values in the dataset.

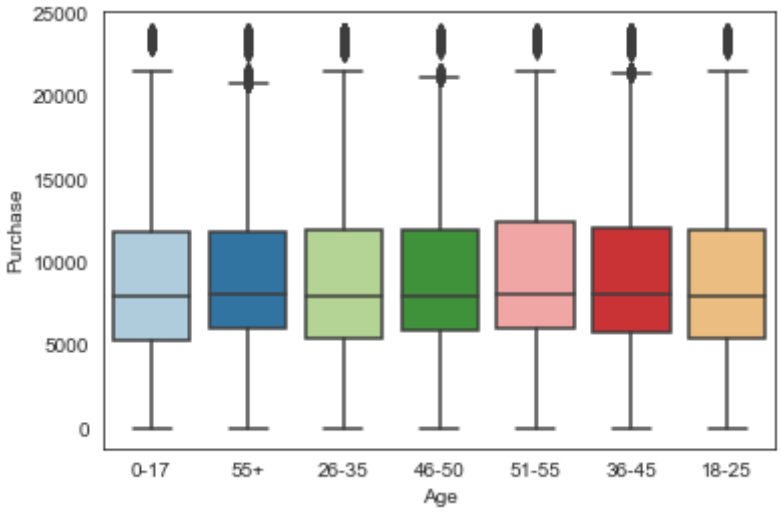
Outlier Detection

In [58]:
```
df.describe()
```

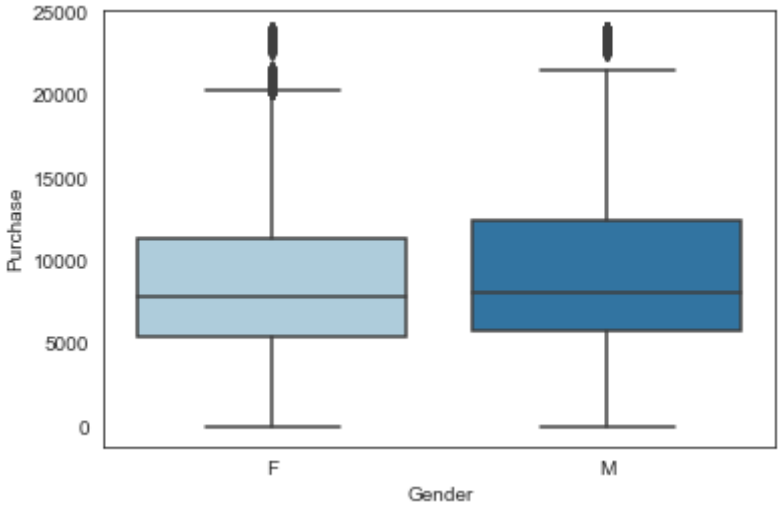
Out[58]:

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

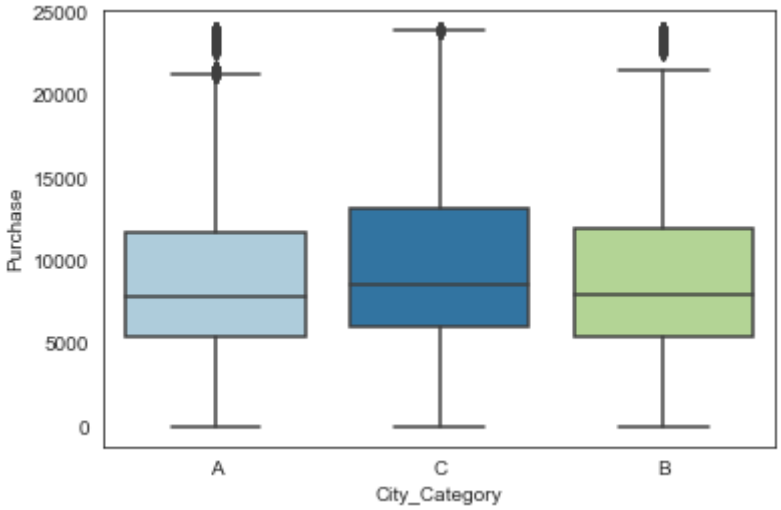
In [59]:
```
sns.boxplot(data=df, x = 'Age',y ='Purchase', palette="Paired")
plt.show()
```



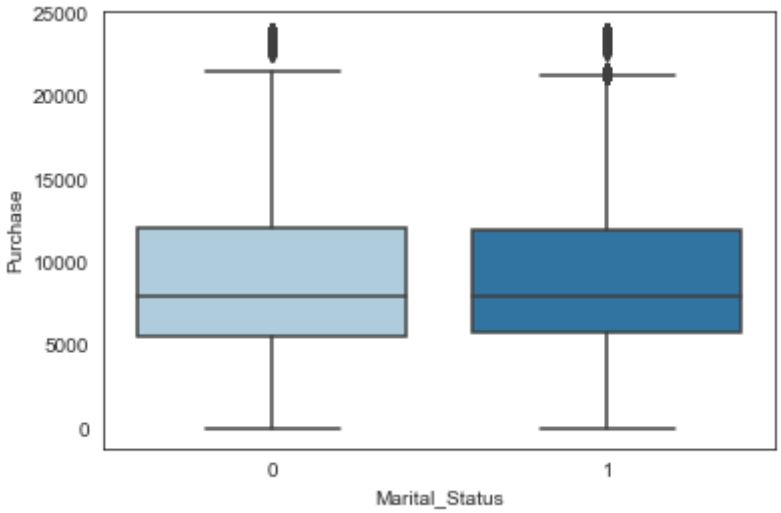
In [60]:
```
sns.boxplot(data=df, x= 'Gender', y ='Purchase', palette="Paired")
plt.show()
```



In [61]:
```
sns.boxplot(data=df, x= 'City_Category', y ='Purchase', palette="Paired")
plt.show()
```



In [62]:
```
sns.boxplot(data=df, x= 'Marital_Status', y ='Purchase', palette="Paired")
plt.show()
```



Using the convenient pandas .quantile() function

In [63]:
```
#create a function to find outliers using IQR
def find_outliers_IQR(df):
 q1=df.quantile(0.25)
 q3=df.quantile(0.75)
 IQR=q3-q1
 outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]
 return outliers
```

In [64]:
```
outliers = find_outliers_IQR(df["Purchase"])
print("number of outliers: "+ str(len(outliers)))
print("max outlier value:"+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))
```

```
number of outliers: 2677
max outlier value:23961
min outlier value: 21401
```

Average amount spend per customer for Male and Female

1. Are women spending more money per transaction than men?

In [65]:
```
amt_df = df.groupby(['User_ID', 'Gender'])[['Purchase']].sum()
amt_df = amt_df.reset_index()
amt_df
```

Out[65]:

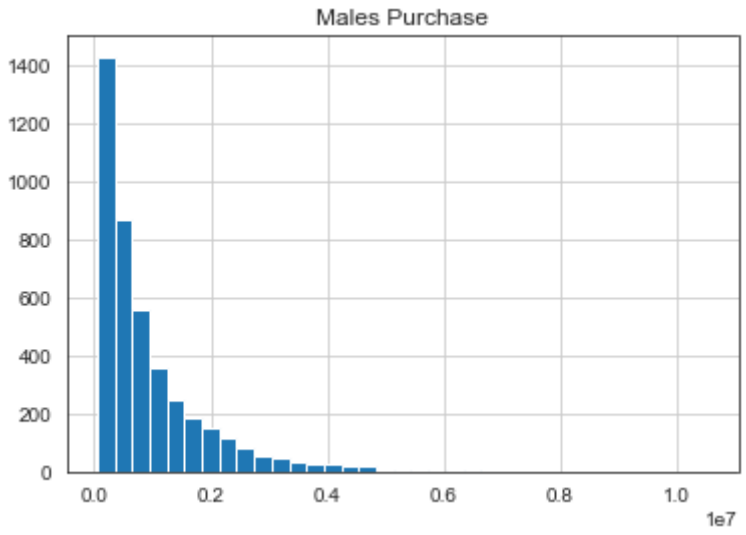
	User_ID	Gender	Purchase
0	1000001	F	334093
1	1000002	M	810472
2	1000003	M	341635
3	1000004	M	206468
4	1000005	M	821001
...	...	...	...
5886	1006036	F	4116058
5887	1006037	F	1119538
5888	1006038	F	90034
5889	1006039	F	590319
5890	1006040	M	1653299

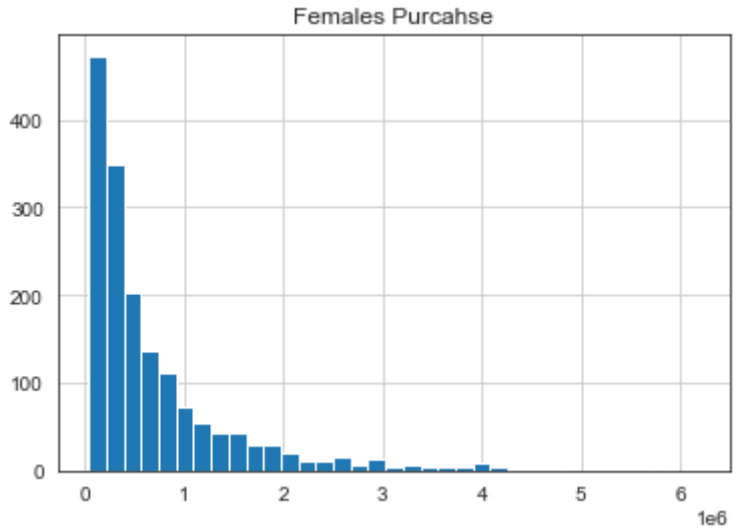
5891 rows × 3 columns

In [66]:
```
avg = amt_df['Gender'].value_counts()
avg
```

Out[66]:
```
M    4225
F    1666
Name: Gender, dtype: int64
```

In [67]:
```
amt_df[amt_df['Gender']=='M']['Purchase'].hist(bins=35)
plt.title('Males Purchase')
plt.show()

amt_df[amt_df['Gender']=='F']['Purchase'].hist(bins=35)
plt.title('Females Purcahse')
plt.show()
```





In [68]:
```
male_avg = amt_df[amt_df['Gender']=='M']['Purchase'].mean()
female_avg = amt_df[amt_df['Gender']=='F']['Purchase'].mean()

print("Average amount spend by Male customers: {:.2f}".format(male_avg))
print("Average amount spend by Female customers: {:.2f}".format(female_avg))
```

```
Average amount spend by Male customers: 925344.40
Average amount spend by Female customers: 712024.39
```

Observation

- Male customers spend more money than female customer

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

In [69]:
```
male_df = amt_df[amt_df['Gender']=='M']
male_df['Purchase'].sum()
```

Out[69]:
```
3909580100
```

In [70]:
```
male_df = amt_df[amt_df['Gender']=='M']
male_df['Purchase'].count()
```

Out[70]:
```
4225
```

In [71]:
```
female_df = amt_df[amt_df['Gender']=='F']
female_df['Purchase'].sum()
```

Out[71]:
```
1186232642
```

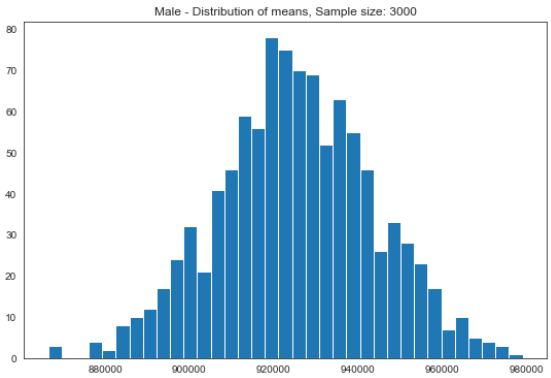
In [72]:
```
female_df = amt_df[amt_df['Gender']=='F']
female_df['Purchase'].count()
```

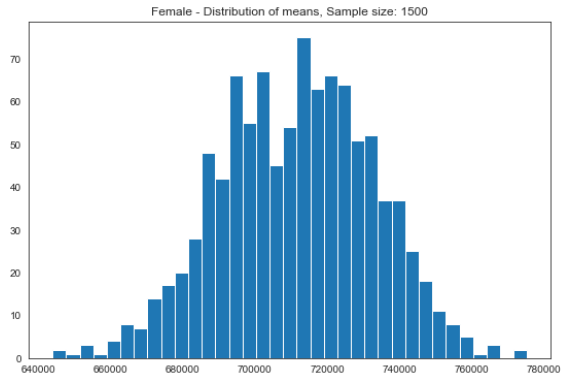
Out[72]:
```
1666
```

Central Limit Theorem

In [73]:
```
genders = ["M", "F"]
```

```
male_sample_size = 3000
female_sample_size = 1500
num_repitions = 1000
male_means = []
female_means = []

for _ in range(num_repitions):
    male_mean = male_df.sample(male_sample_size, replace=True)['Purchase'].mean()
    female_mean = female_df.sample(female_sample_size, replace=True)['Purchase'].m

    male_means.append(male_mean)
    female_means.append(female_mean)

fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))

axis[0].hist(male_means, bins=35)
axis[1].hist(female_means, bins=35)
axis[0].set_title("Male - Distribution of means, Sample size: 3000")
axis[1].set_title("Female - Distribution of means, Sample size: 1500")

plt.show()
```





In [74]:
```
print("Population mean - Mean of sample means of amount spend for Male: {:.2f}".fo
print("Population mean - Mean of sample means of amount spend for Female: {:.2f}".

print("\nMale - Sample mean: {:.2f} Sample std: {:.2f}".format(male_df['Purchase']
print("Female - Sample mean: {:.2f} Sample std: {:.2f}".format(female_df['Purchase
```

```
Population mean - Mean of sample means of amount spend for Male: 925300.27
Population mean - Mean of sample means of amount spend for Female: 711413.25

Male - Sample mean: 925344.40 Sample std: 985830.10
Female - Sample mean: 712024.39 Sample std: 807370.73
```

Observation

Now using the Central Limit Theorem for the population we can say that:

- Average amount spend by male customers is 9,26,341.86
- Average amount spend by female customers is 7,11,704.09

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

In [75]:
```
male_margin_of_error_clt = 1.96*male_df['Purchase'].std()/np.sqrt(len(male_df))
male_sample_mean = male_df['Purchase'].mean()
male_lower_lim = male_sample_mean - male_margin_of_error_clt
```

```
male_upper_lim = male_sample_mean + male_margin_of_error_clt

female_margin_of_error_clt = 1.96*female_df['Purchase'].std()/np.sqrt(len(female_d
female_sample_mean = female_df['Purchase'].mean()
female_lower_lim = female_sample_mean - female_margin_of_error_clt
female_upper_lim = female_sample_mean + female_margin_of_error_clt

print("Male confidence interval of means: ({:.2f}, {:.2f})".format(male_lower_lim,
print("Female confidence interval of means: ({:.2f}, {:.2f})".format(female_lower_
```

```
Male confidence interval of means: (895617.83, 955070.97)
Female confidence interval of means: (673254.77, 750794.02)
```

Now we can infer about the population that, 95% of the times:

- Average amount spend by male customer will lie in between: (895617.83, 955070.97)
- Average amount spend by female customer will lie in between: (673254.77, 750794.02)

Calculating the average amount spent by Married Vs Single

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

In [76]:
```
amt_df
```

Out[76]:

	User_ID	Gender	Purchase
0	1000001	F	334093
1	1000002	M	810472
2	1000003	M	341635
3	1000004	M	206468
4	1000005	M	821001
...	...	...	...
5886	1006036	F	4116058
5887	1006037	F	1119538
5888	1006038	F	90034
5889	1006039	F	590319
5890	1006040	M	1653299

5891 rows × 3 columns

In [77]:
```
amt_df = df.groupby(['User_ID', 'Marital_Status'])[['Purchase']].sum()
amt_df = amt_df.reset_index()
amt_df
```

Out[77]:

	User_ID	Marital_Status	Purchase
0	1000001	0	334093
1	1000002	0	810472
2	1000003	0	341635
3	1000004	1	206468
4	1000005	1	821001
...	...	...	...
5886	1006036	1	4116058
5887	1006037	0	1119538
5888	1006038	0	90034
5889	1006039	1	590319
5890	1006040	0	1653299

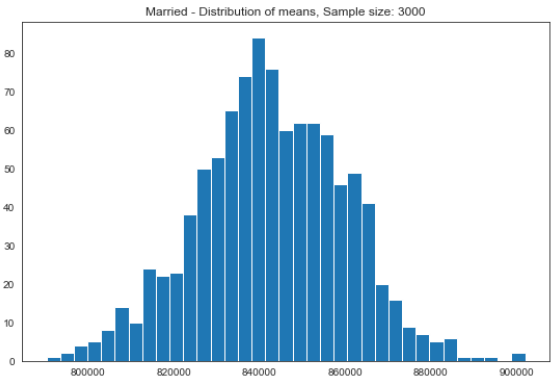
5891 rows × 3 columns

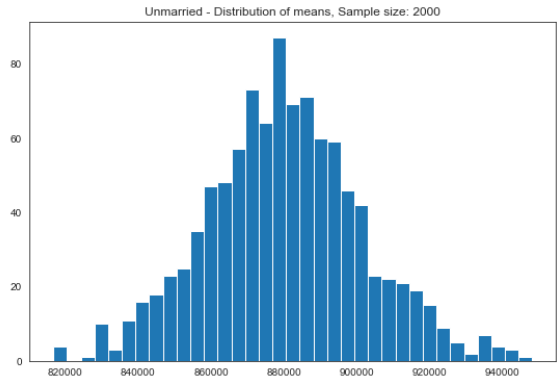
In [78]:
```
amt_df['Marital_Status'].value_counts()
```

Out[78]:
```
0    3417
1    2474
Name: Marital_Status, dtype: int64
```

In [79]:
```
amt_df
amt_df = df.groupby(['User_ID', 'Marital_Status'])[['Purchase']].sum()
amt_df = amt_df.reset_index()
amt_df
amt_df['Marital_Status'].value_counts()
marid_samp_size = 3000
unmarid_sample_size = 2000
num_repitions = 1000
marid_means = []
unmarid_means = []
for i in range(num_repitions):
    marid_mean = amt_df[amt_df['Marital_Status']==1].sample(marid_samp_size,replac
    unmarid_mean = amt_df[amt_df['Marital_Status']==0].sample(unmarid_sample_size,
    marid_means.append(marid_mean)
    unmarid_means.append(unmarid_mean)


fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))
axis[0].hist(marid_means, bins=35)
axis[1].hist(unmarid_means, bins=35)
axis[0].set_title("Married - Distribution of means, Sample size: 3000")
axis[1].set_title("Unmarried - Distribution of means, Sample size: 2000")

plt.show()

print("Population mean - Mean of sample means of amount spend for Married: {:.2f}"
print("Population mean - Mean of sample means of amount spend for Unmarried: {:.2f

print("Married - Sample mean: {:.2f} Sample std: {:.2f}".format(amt_df[amt_df['Mar
print("Unmarried - Sample mean: {:.2f} Sample std: {:.2f}".format(amt_df[amt_df['Ma
for val in ["Married", "Unmarried"]:new_val = 1 if val == "Married" else 0
new_df = amt_df[amt_df['Marital_Status']==new_val]
```

```
margin_of_error_clt = 1.96*new_df['Purchase'].std()/np.sqrt(len(new_df))
sample_mean = new_df['Purchase'].mean()
lower_lim = sample_mean - margin_of_error_clt
upper_lim = sample_mean + margin_of_error_clt

print("{} confidence interval of means: ({:.2f}, {:.2f})".format(val, lower_lim, u
```






```
Population mean - Mean of sample means of amount spend for Married: 842688.88
Population mean - Mean of sample means of amount spend for Unmarried: 880171.70
Married - Sample mean: 843526.80 Sample std: 935352.12
Unmarried - Sample mean: 880575.78 Sample std: 949436.25
Unmarried confidence interval of means: (848741.18, 912410.38)
```

In [80]:
```
for val in ["Married", "Unmarried"]:

    new_val = 1 if val == "Married" else 0

    new_df = amt_df[amt_df['Marital_Status']==new_val]

    margin_of_error_clt = 1.96*new_df['Purchase'].std()/np.sqrt(len(new_df))
    sample_mean = new_df['Purchase'].mean()
    lower_lim = sample_mean - margin_of_error_clt
    upper_lim = sample_mean + margin_of_error_clt

    print("{} confidence interval of means: ({:.2f}, {:.2f})".format(val, lower_li
```

```
Married confidence interval of means: (806668.83, 880384.76)
Unmarried confidence interval of means: (848741.18, 912410.38)
```

Calculating the average amount spent by Age.

1. Results when the same activity is performed for Age

In [81]:
```
amt_df = df.groupby(['User_ID', 'Age'])[['Purchase']].sum()
amt_df = amt_df.reset_index()

amt_df
```

Out[81]:

	User_ID	Age	Purchase
0	1000001	0-17	334093
1	1000002	55+	810472
2	1000003	26-35	341635
3	1000004	46-50	206468
4	1000005	26-35	821001
...	...	...	...
5886	1006036	26-35	4116058
5887	1006037	46-50	1119538
5888	1006038	55+	90034
5889	1006039	46-50	590319
5890	1006040	26-35	1653299

5891 rows × 3 columns

In [82]:
```
amt_df['Age'].value_counts()
```

Out[82]:
```
26-35    2053
36-45    1167
18-25    1069
46-50     531
51-55     481
55+       372
0-17      218
Name: Age, dtype: int64
```

In [83]:
```
sample_size = 200
num_repitions = 1000

all_means = {}

age_intervals = ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']
for age_interval in age_intervals:
    all_means[age_interval] = []

for age_interval in age_intervals:
    for _ in range(num_repitions):
        mean = amt_df[amt_df['Age']==age_interval].sample(sample_size, replace=Tru
        all_means[age_interval].append(mean)
```

In [84]:
```
for val in ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']:

    new_df = amt_df[amt_df['Age']==val]

    margin_of_error_clt = 1.96*new_df['Purchase'].std()/np.sqrt(len(new_df))
    sample_mean = new_df['Purchase'].mean()
    lower_lim = sample_mean - margin_of_error_clt
    upper_lim = sample_mean + margin_of_error_clt

    print("For age {} --> confidence interval of means: ({:.2f}, {:.2f})".format(v
```

```
For age 26-35 --> confidence interval of means: (945034.42, 1034284.21)
For age 36-45 --> confidence interval of means: (823347.80, 935983.62)
For age 18-25 --> confidence interval of means: (801632.78, 908093.46)
For age 46-50 --> confidence interval of means: (713505.63, 871591.93)
For age 51-55 --> confidence interval of means: (692392.43, 834009.42)
For age 55+ --> confidence interval of means: (476948.26, 602446.23)
For age 0-17 --> confidence interval of means: (527662.46, 710073.17)
```

Insights

- 80% of the users are between the age 18-50 ,40%: 26-35, 18%: 18-25, 20%: 36-45
- 75% of the users are Male and 25% are Female.
- 60% Single, 40% Married.
- 35% Staying in the city from 1 year, 18% from 2 years, 17% from 3 years.
- There are Total of 20 product categories.
- There are 20 differnent types of occupations in the city.

- Most of the users are Male

- There are 20 different types of Occupation and Product_Category
- More users belong to B City_Category
- More users are Single as compare to Married
- Product_Category - 1, 5, 8, & 11 have highest purchasing frequency.

- Average amount spend by Male customers: 925344.40

- Average amount spend by Female customers: 712024.39

Confidence Interval by Gender

Now using the Central Limit Theorem for the population:

- Average amount spend by male customers is 9,26,341.86
- Average amount spend by female customers is 7,11,704.09 Now we can infer about the population that, 95% of the times:
- Average amount spend by male customer will lie in between: (895617.83, 955070.97)
- Average amount spend by female customer will lie in between: (673254.77, 750794.02)

Confidence Interval by Marital_Status

- Married confidence interval of means: (806668.83, 880384.76)
- Unmarried confidence interval of means: (848741.18, 912410.38)

Confidence Interval by Age

- For age 26-35 --> confidence interval of means: (945034.42, 1034284.21)
- For age 36-45 --> confidence interval of means: (823347.80, 935983.62)
- For age 18-25 --> confidence interval of means: (801632.78, 908093.46)
- For age 46-50 --> confidence interval of means: (713505.63, 871591.93)
- For age 51-55 --> confidence interval of means: (692392.43, 834009.42)
- For age 55+ --> confidence interval of means: (476948.26, 602446.23)
- For age 0-17 --> confidence interval of means: (527662.46, 710073.17

Recommendations

1. Men spent more money than women, So company should focus on retaining the male customers and getting more male customers.

2. Product_Category - 1, 5, 8, & 11 have highest purchasing frequency. it means these are the products in these categories are liked more by customers. Company can focus on

selling more of these products or selling more of the products which are purchased less.

3. Unmarried customers spend more money than married customers, So company should focus on acquisition of Unmarried customers.
4. Customers in the age 18-45 spend more money than the others, So company should focus on acquisition of customers who are in the age 18-45.
5. 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.
