In [5]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm
```

In [6]:
```
walmart = pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/
walmart
```

Out[6]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Ma
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+	
...	...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	1	
550064	1006035	P00375436	F	26-35	1	C	3	
550065	1006036	P00375436	F	26-35	15	B	4+	
550066	1006038	P00375436	F	55+	1	C	2	
550067	1006039	P00371644	F	46-50	0	B	4+	

550068 rows × 10 columns

In [7]:
```
walmart.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 [57]:

```
# No of rows and columns
walmart.shape
```

Out[57]:

```
(550068, 10)
```

In [14]:

```
#top 5 rows
walmart.head()
```

Out[14]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_S
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 [52]:

```
# Description of all numeric data
walmart.describe()
```

Out[52]:

	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 [18]:

```
# datatype of all the columns
walmart.dtypes
```

Out[18]:

```
User_ID                        int64
Product_ID                    object
Gender                        object
Age                           object
Occupation                     int64
City_Category                 object
Stay_In_Current_City_Years    object
Marital_Status                 int64
Product_Category               int64
Purchase                       int64
dtype: object
```

In [19]:

```
# checking the information about the data
walmart.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 [20]:

```
# Checking last 5 rows of data
walmart.tail()
```

Out[20]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Ma
550063	1006033	P00372445	M	51-55	13	B	1	
550064	1006035	P00375436	F	26-35	1	C	3	
550065	1006036	P00375436	F	26-35	15	B	4+	
550066	1006038	P00375436	F	55+	1	C	2	
550067	1006039	P00371644	F	46-50	0	B	4+	

In [59]:

```
# Most Popular product is P00265242
walmart["Product_ID"].value_counts()
```

Out[59]:

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

In [30]:

```
# walmart data has 3631 unique products

walmart["Product_ID"].nunique()
```

Out[30]:

```
3631
```

In [33]:

```
# walmart data has no null values

walmart.isnull().sum()
```

Out[33]:

```
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
```

In [39]:

```
# Number of unique ages given in data
walmart["Age"].nunique()
```

Out[39]:

```
7
```

In [41]:
```
# Age categories
walmart["Age"].unique()
```

Out[41]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

In [56]:
```
# Male and Female count of data as we can see males has more data
walmart.groupby(by=["Gender"])["Gender"].count()
```

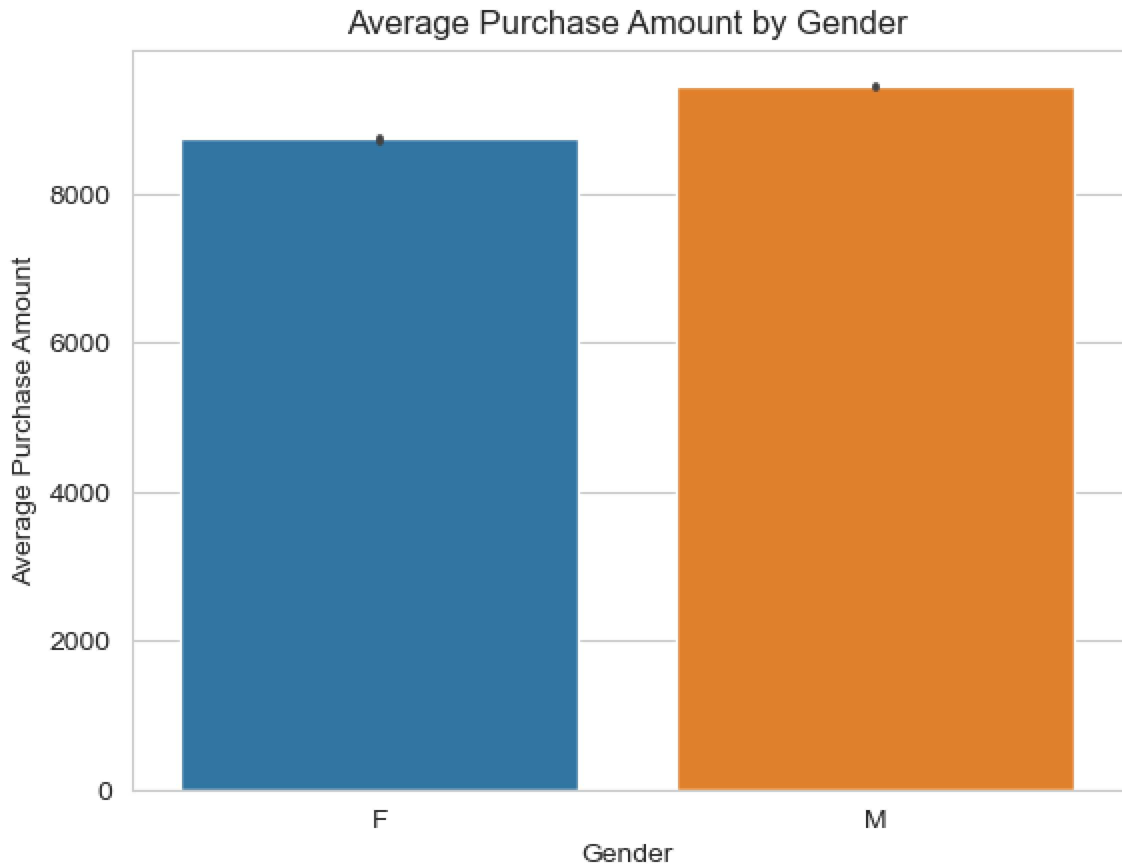
Out[56]:
```
Gender
F    135809
M    414259
Name: Gender, dtype: int64
```

Soution 5

In [124…
```
# Set the style for the plot (optional)
sns.set_style('whitegrid')

# Create a barplot for 'Gender'
sns.countplot(data=walmart, x='Gender', palette='Set2')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Gender Distribution')

# Display the plot
plt.show()
```

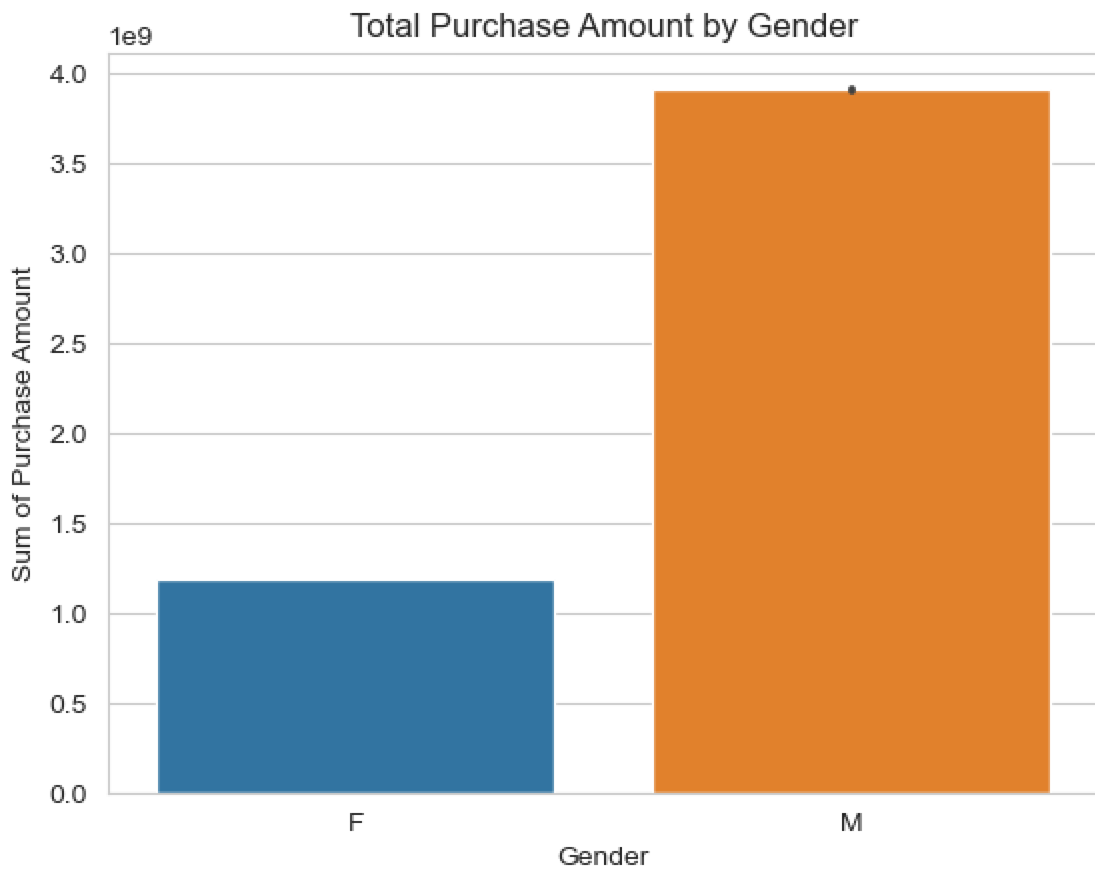
In [128…

```
# Create a barplot using Seaborn
sns.barplot(data=walmart, x='Gender', y='Purchase', estimator=np.mean)
plt.xlabel('Gender')
plt.ylabel('Average Purchase Amount')
plt.title('Average Purchase Amount by Gender')

# Display the plot
plt.show()
```



In [130…

```
sns.barplot(data=walmart, x='Gender', y='Purchase', estimator=np.sum)
plt.xlabel('Gender')
plt.ylabel('Sum of Purchase Amount')
plt.title('Total Purchase Amount by Gender')

# Display the plot
plt.show()
```



In [120…:
```
purchase_sum = np.sum(walmart['Purchase'])
purchase_sum
```

Out[120]:
```
5095812742
```

In [58]:
```
# Chceking Duplicated rows there is no duplicated rows
walmart.duplicated().sum()
```

Out[58]:
```
0
```

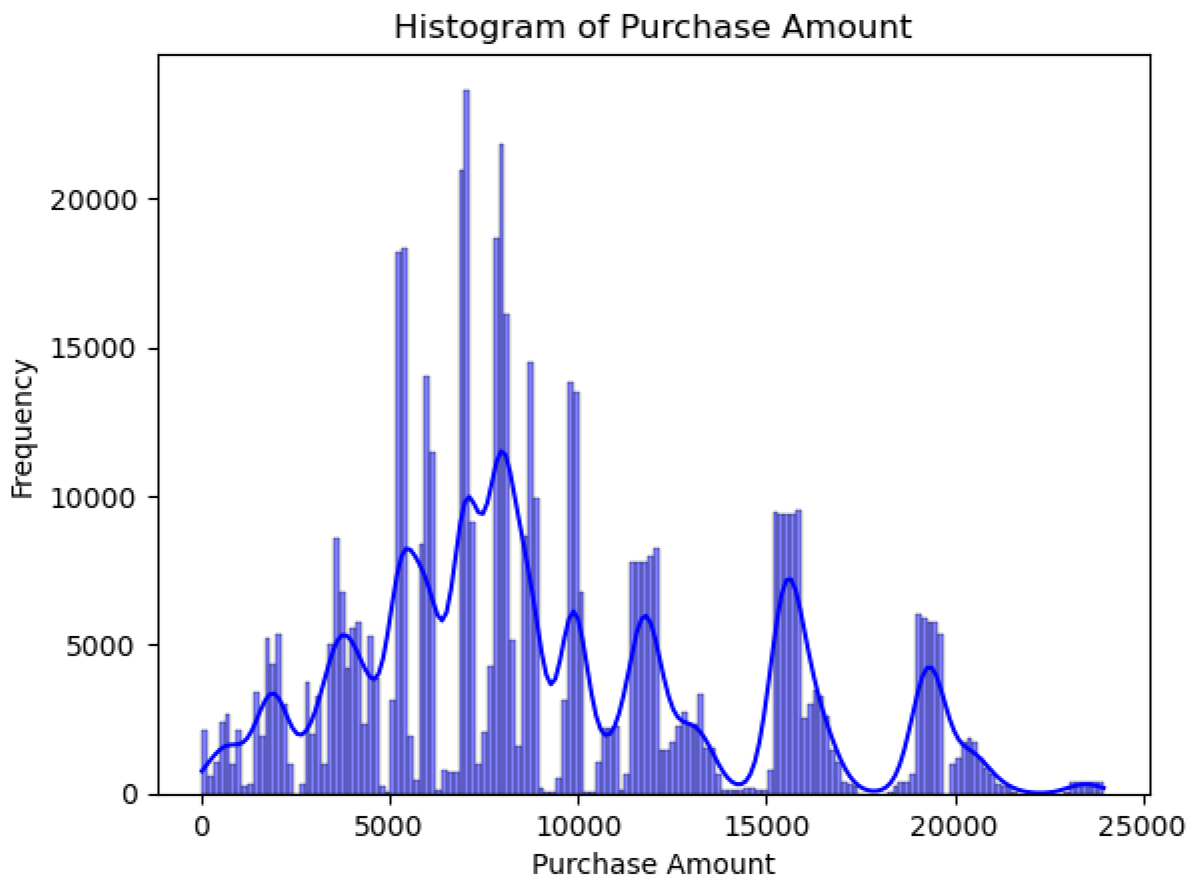
In [60]:
```
walmart.head()
```

Out[60]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_S
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+	

Walmart Data is clean as there is neither null values nor duplicated rows

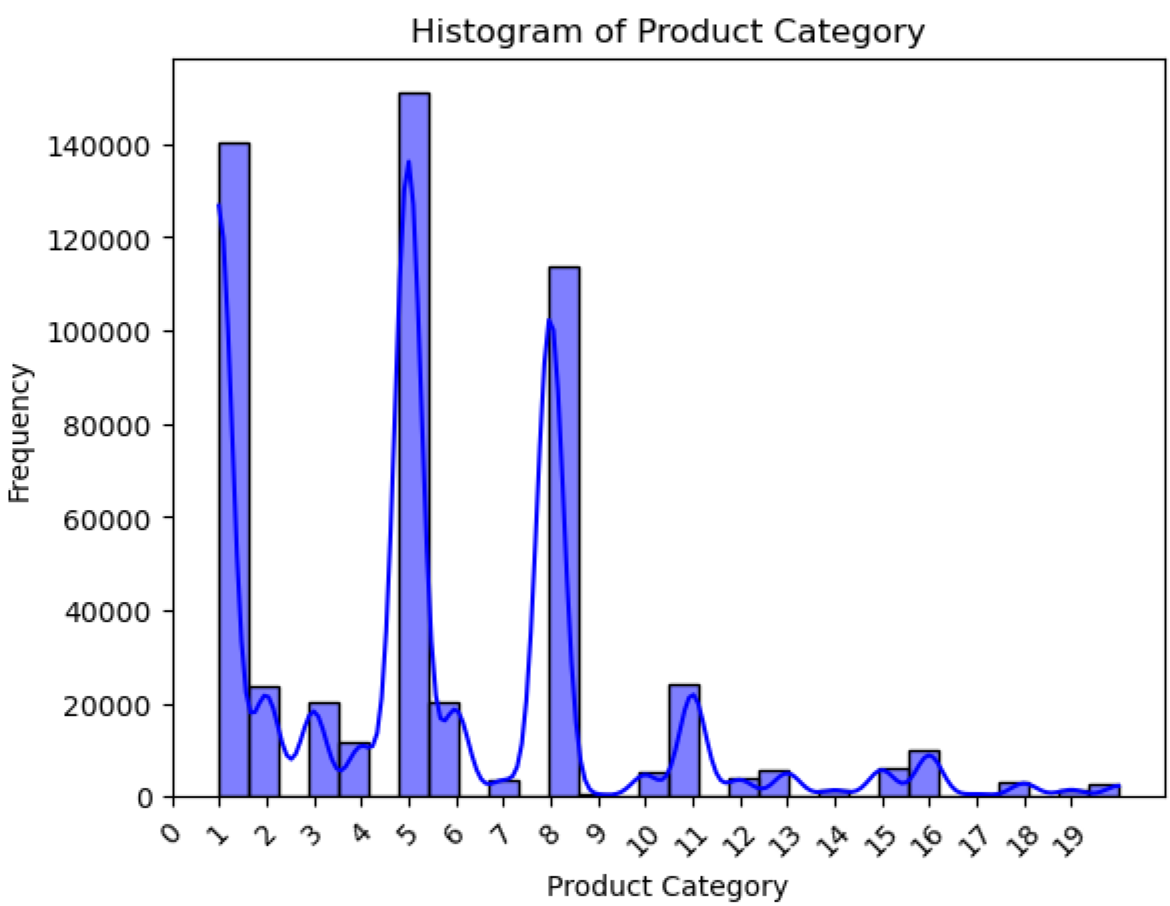
In [97]:
```
sns.histplot(walmart['Purchase'], kde=True, color='blue')
plt.xlabel('Purchase Amount')
plt.ylabel('Frequency')
plt.title('Histogram of Purchase Amount')

# Display the plot
plt.show()
```



In [116…
```
walmart.groupby(by="Product_Category" )["Product_Category"].count()
```

Out[116]:
```
Product_Category
1     140378
2      23864
3      20213
4      11753
5     150933
6      20466
7       3721
8     113925
9        410
10      5125
11     24287
12      3947
13      5549
14      1523
15      6290
16      9828
17       578
18      3125
19      1603
20      2550
Name: Product_Category, dtype: int64
```

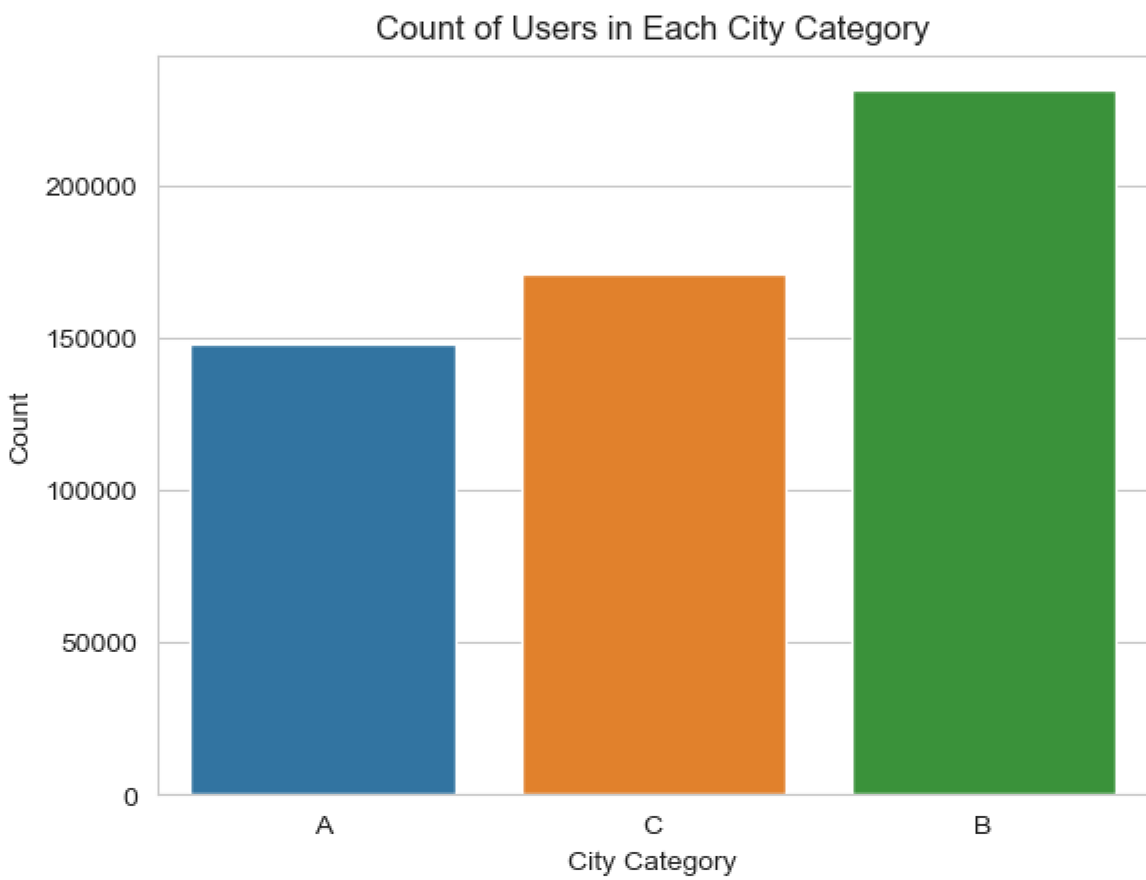
In [111…
```
sns.histplot(x=walmart['Product_Category'], bins=30, kde=True, color='blue')
plt.xlabel('Product Category')
plt.ylabel('Frequency')
plt.xticks(range(20), rotation=45)
plt.title('Histogram of Product Category')
# Display the plot
plt.show()
```



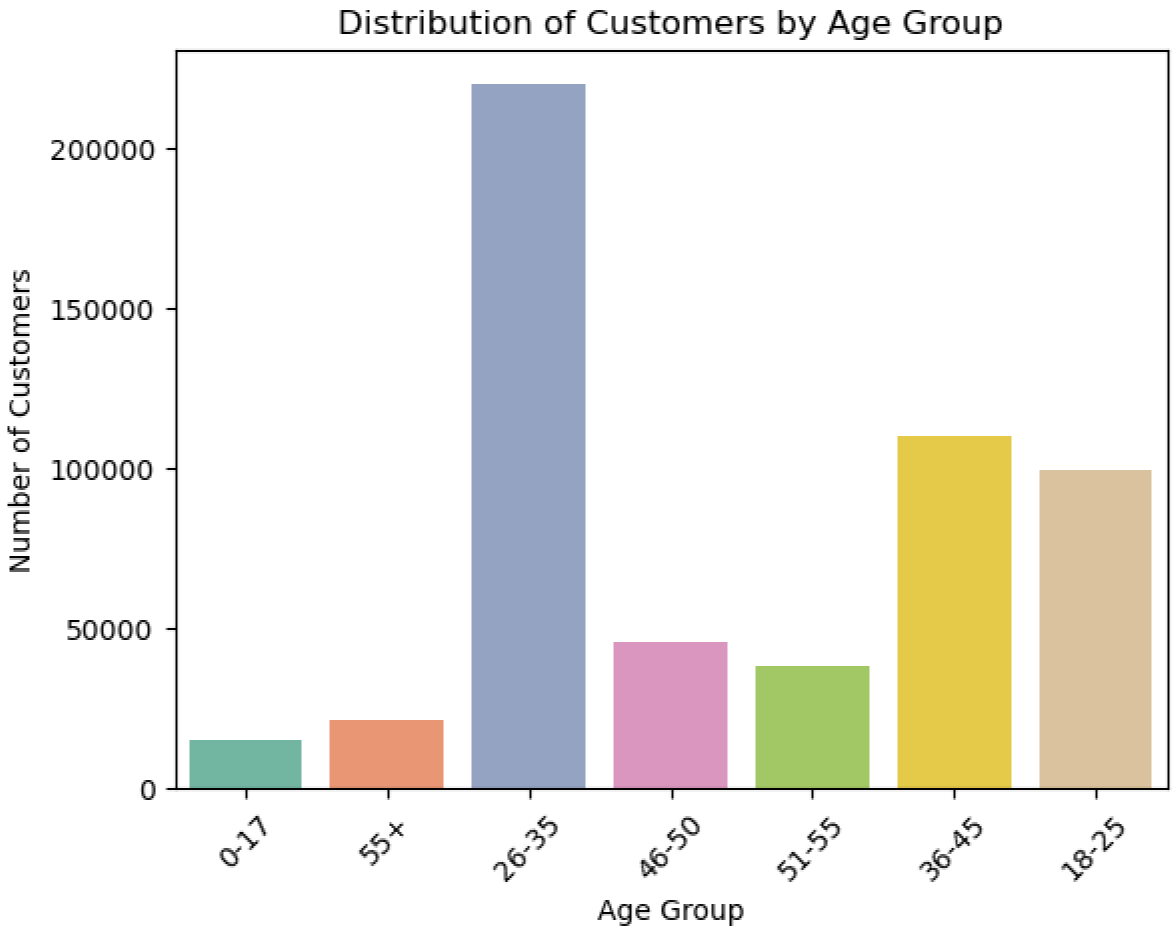
In [100…
```
walmart["Product_Category"].nunique()
```
Out[100]:
```
20
```

In [117…
```
walmart["Product_Category"].unique()
```
Out[117]:
```
array([ 3,  1, 12,  8,  5,  4,  2,  6, 14, 11, 13, 15,  7, 16, 18, 10, 17,
        9, 20, 19], dtype=int64)
```

In [132…
```
sns.countplot(data=walmart, x='City_Category')
plt.xlabel('City Category')
plt.ylabel('Count')
plt.title('Count of Users in Each City Category')
plt.show()
```



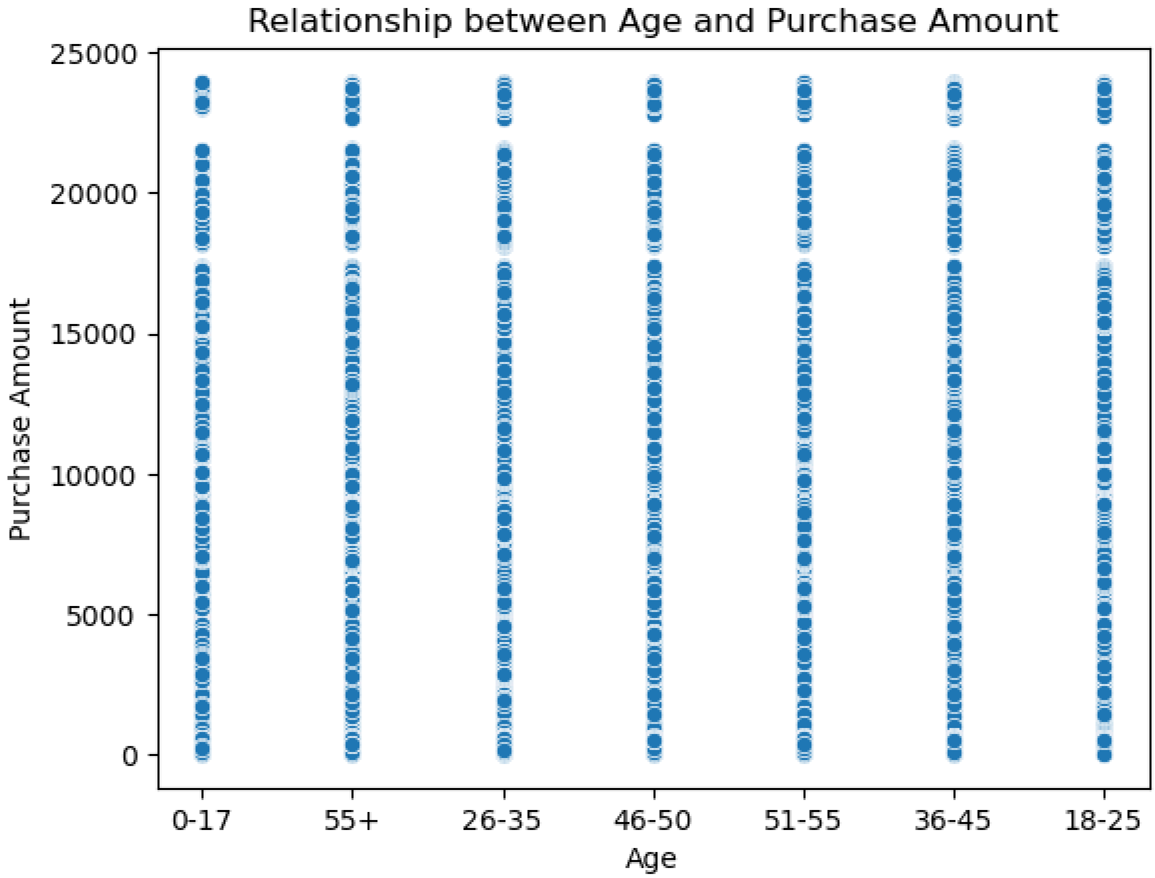
In [80]:
```
sns.countplot(data=walmart, x='Age', palette='Set2')

# Customize the plot (add titles, labels, etc.)
plt.title("Distribution of Customers by Age Group")
plt.xlabel("Age Group")
plt.ylabel("Number of Customers")

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show the plot
plt.show()
```



In [82]:
```
# Create the scatter plot
sns.scatterplot(data=walmart, x='Age', y='Purchase', palette='Set2')

# Customize the plot (add titles, labels, etc.)
plt.title("Relationship between Age and Purchase Amount")
plt.xlabel("Age")
plt.ylabel("Purchase Amount")
```

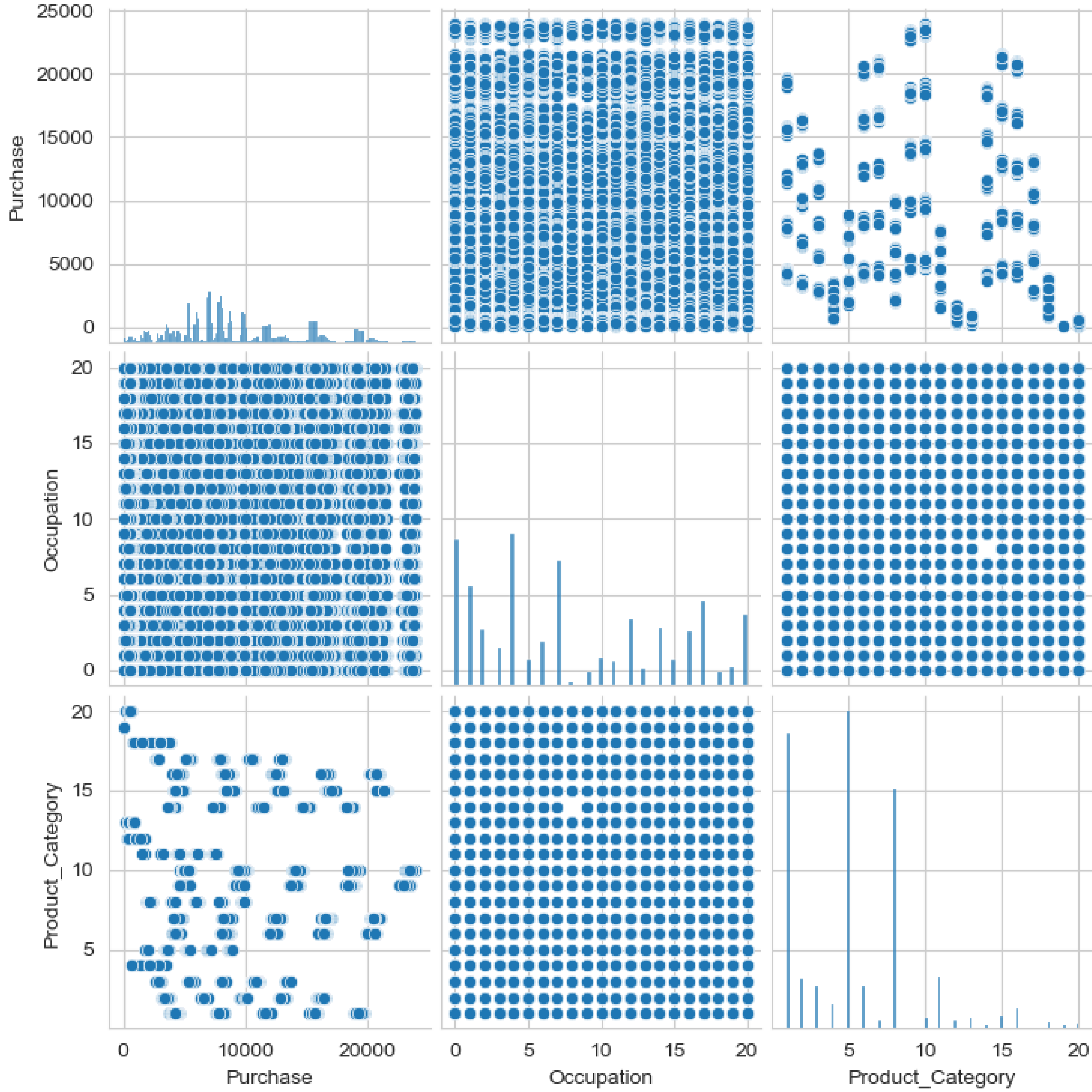
```
C:\Users\harsh\AppData\Local\Temp\ipykernel_27344\3758588963.py:2: UserWarning: Ignor
ing `palette` because no `hue` variable has been assigned.
  sns.scatterplot(data=walmart, x='Age', y='Purchase', palette='Set2')
```

Out[82]:
```
Text(0, 0.5, 'Purchase Amount')
```



In [135…

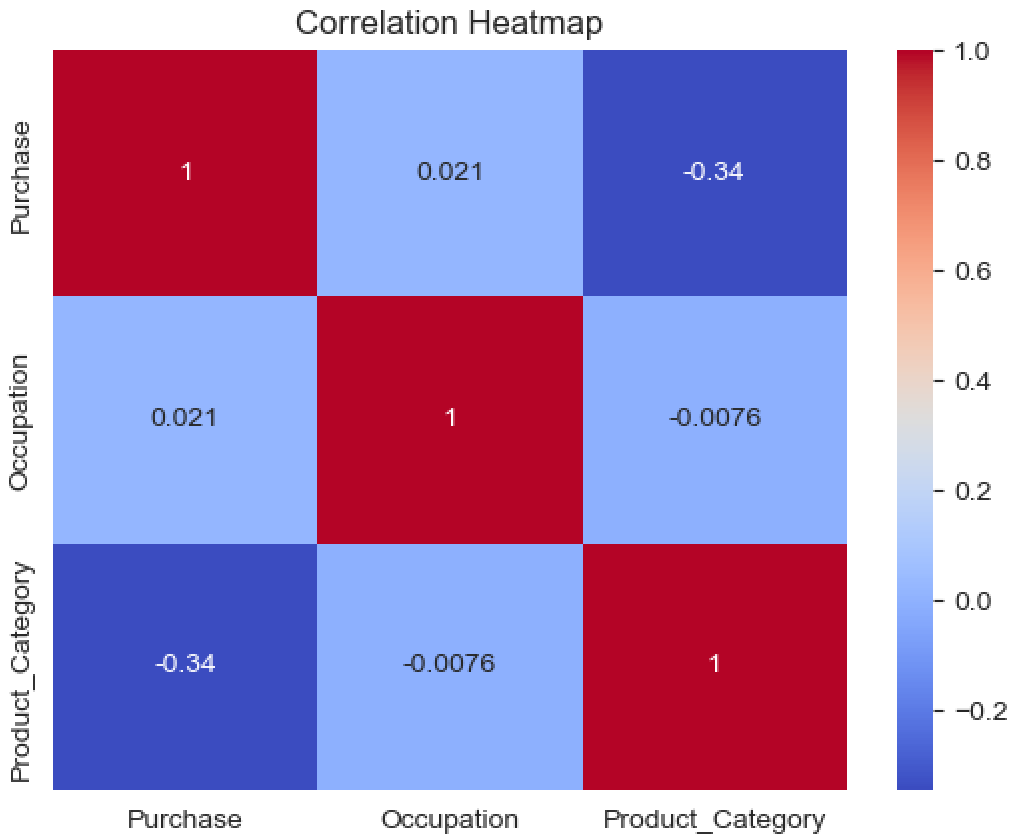
```
sns.pairplot(walmart[['Purchase', 'Age', 'Occupation', 'Product_Category']])
plt.show()
```



In [137…

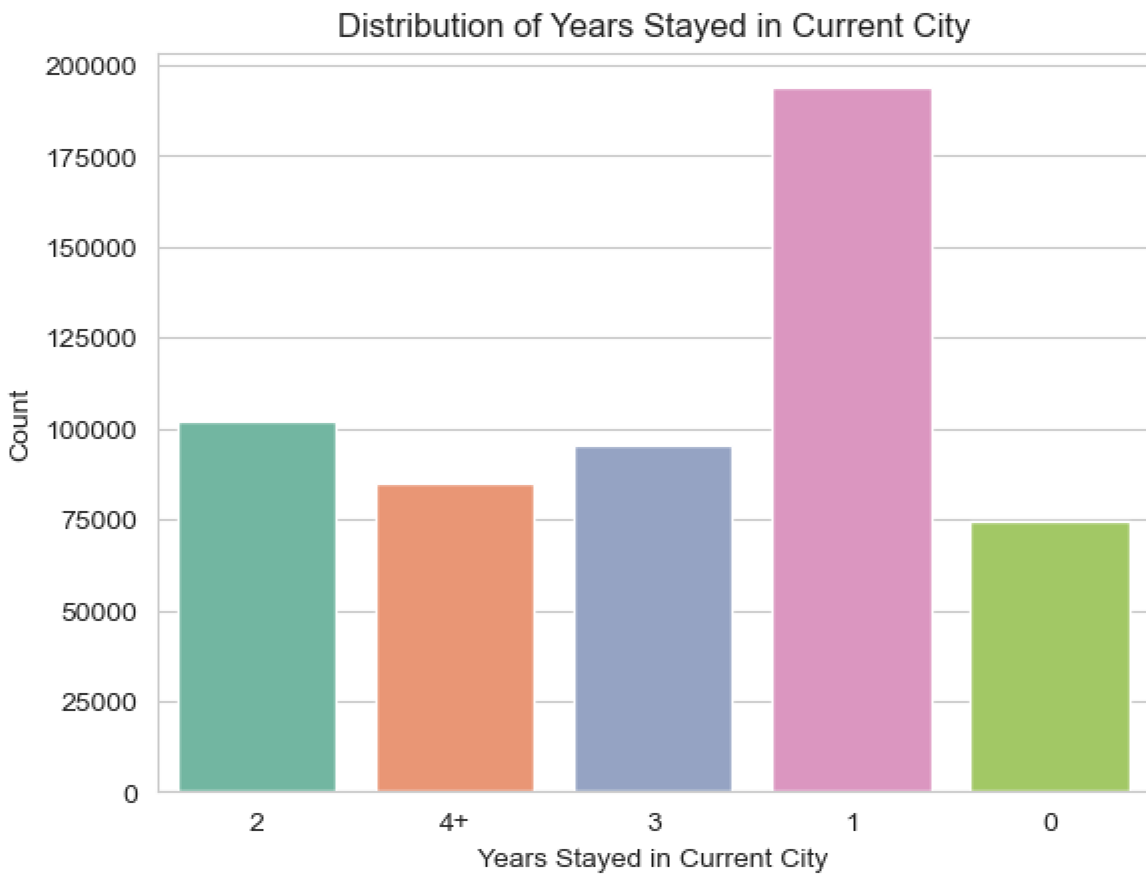
```
correlation_matrix = walmart[['Purchase', 'Age', 'Occupation', 'Product_Category']].co
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

```
C:\Users\harsh\AppData\Local\Temp\ipykernel_10752\2055096445.py:1: FutureWarning: The
default value of numeric_only in DataFrame.corr is deprecated. In a future version, i
t will default to False. Select only valid columns or specify the value of numeric_on
ly to silence this warning.
  correlation_matrix = walmart[['Purchase', 'Age', 'Occupation', 'Product_Categor
y']].corr()
```

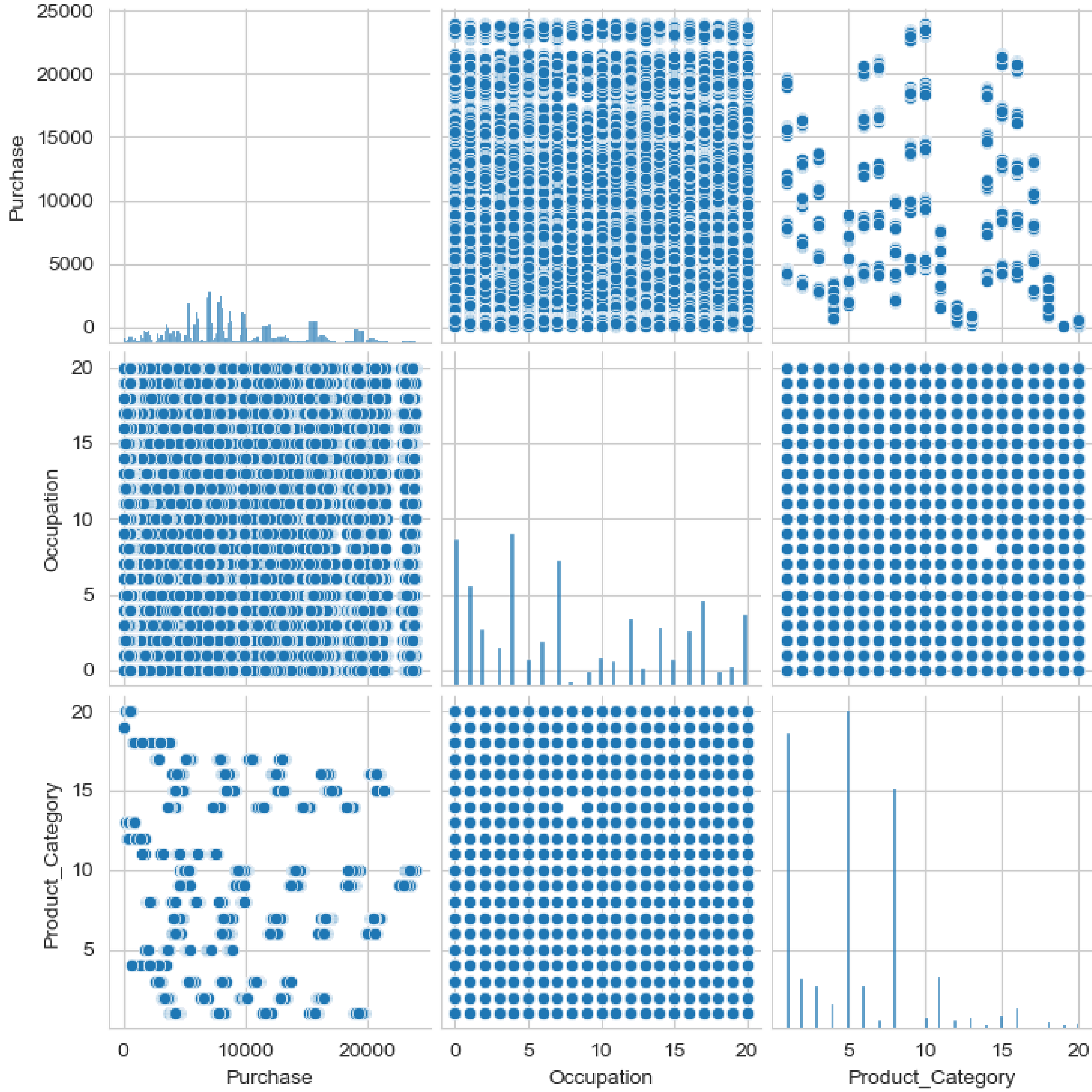


In [139…

```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a countplot for 'Stay_In_Current_City_Years'
sns.countplot(data=walmart, x='Stay_In_Current_City_Years', palette='Set2')
plt.xlabel('Years Stayed in Current City')
plt.ylabel('Count')
plt.title('Distribution of Years Stayed in Current City')
plt.show()
```



In [142…
```
sns.pairplot(walmart[['Purchase', 'Age', 'Occupation', 'Product_Category']])
plt.show()
```



Solution 2 (Missing value and outlier Detection)

In [144…

```
# walmart data has no null values

walmart.isnull().sum()
```

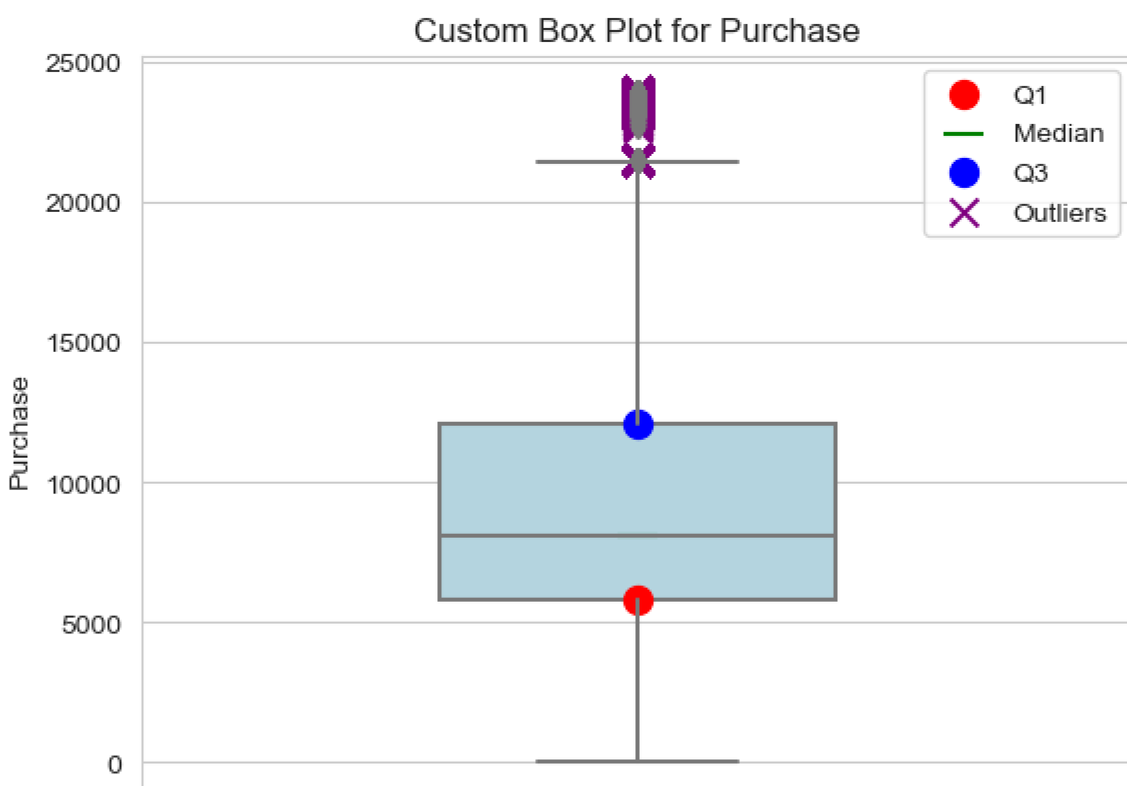
Out[144]:

```
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
```

In [ ]:

In [145…

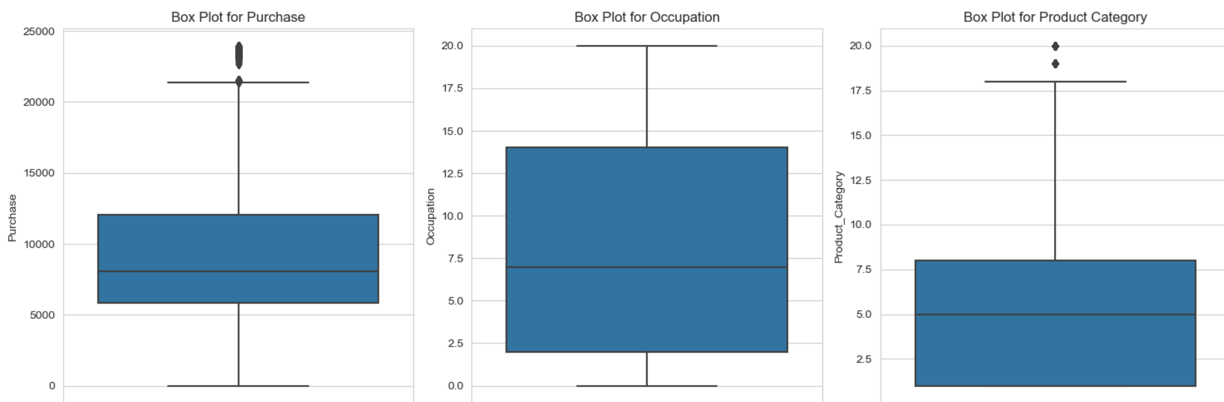
```
# To Detect Outliers
# Box Plot for 'Purchase'
sns.boxplot(y=walmart['Purchase'], color='lightblue', width=0.4)

# Plot Q1, Q3, and Median
plt.scatter([0], [walmart['Purchase'].quantile(0.25)], marker='o', color='red', s=100,
plt.scatter([0], [walmart['Purchase'].median()], marker='_', color='green', s=200, lab
plt.scatter([0], [walmart['Purchase'].quantile(0.75)], marker='o', color='blue', s=100

# Identify and mark outliers
outliers = walmart[(walmart['Purchase'] < walmart['Purchase'].quantile(0.25) - 1.5 * (
plt.scatter([0] * len(outliers), outliers['Purchase'], marker='x', color='purple', s=1

# Set labels and legend
plt.ylabel('Purchase')
plt.legend()

# Show the plot
plt.title('Custom Box Plot for Purchase')
plt.show()
```



In [149…

```
# Create subplots for each box plot
# Median is exactly at 50 everything below is
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Box Plot for 'Purchase'
sns.boxplot(y=walmart['Purchase'], ax=axes[0])
axes[0].set_title('Box Plot for Purchase')

# Box Plot for 'Occupation'
sns.boxplot(y=walmart['Occupation'], ax=axes[1])
```

```
axes[1].set_title('Box Plot for Occupation')

# Box Plot for 'Product_Category'
sns.boxplot(y=walmart['Product_Category'], ax=axes[2])
axes[2].set_title('Box Plot for Product Category')

# Adjust the layout
plt.tight_layout()

# Display the box plots
plt.show()
```



Solution 3:

User_ID (int64): This attribute appears to represent a unique identifier for each user. The range of values will depend on the number of unique users in the dataset.

Product_ID (object): This attribute is a string identifier for products. The range of values will be determined by the different product IDs present in the dataset.

Gender (object): Gender is a categorical attribute, and the range of values will typically include categories like 'Male' and 'Female.'

Age (object): Age is another categorical attribute, and its range will include different age categories ('0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'). Occupation (int64): Occupation is a numerical attribute representing the occupation of users. The range of values depends on the unique occupation codes or categories in the dataset.

City_Category (object): This categorical attribute will include different city categories such as 'A,' 'B, and C.

Stay_In_Current_City_Years (object): This attribute represents the number of years a user has stayed in the current city. The range of values will include different year categories (e.g., '0,' '1,' '2,' '3,' '4+,' etc.).

Marital_Status (int64): This numerical attribute typically represents marital status, with a range of values that may include '0' for unmarried and '1' for married.

Product_Category (int64): This numerical attribute represents product categories, and the range of values will depend on the unique product category codes or categories in the dataset.

Purchase (int64): This numerical attribute represents the purchase amount. The range of values will depend on the minimum and maximum purchase amounts in the dataset.

It's important to understand the range of attributes in your dataset as it helps in data exploration, preprocessing, and selecting appropriate analytical methods for your specific analysis or machine learning tasks.

Distribution of Variables:

For numeric variables like 'Purchase,' 'Occupation,' and 'Product_Category,'

As We can see for the "Purchase" Box plot :

Medain about: 7000 to 7500(approx), which mean 7000 to 7500 is the middle amount meaning that 50% of data points have a value smaller or equal to the median and 50% of data points have a value higher or equal to the median. InterQuartile Range: 6000 to 12000 (approx), which mean 50% of people buy the product of amount 6000 to 12000. BoxPlot is right skew means most purchased amount is positive, meaning above median. More than 20000 amount will be considered as outliers

As We can see for the 'Occupation' Box plot : Medain: 7, meaning mostly people have 7 occupation` InterQuartile Range: 2 to 14 (approx) There is no outlier in occupation

As We can see for the 'Product_Category' Box plot : Medain: 5 InterQuartile Range: 2 to 8 (approx) Outlier: Product categories above 17.5 will be considered as outliers.

For categorical variables like 'Gender,' 'Age,' 'City_Category,' and 'Stay_In_Current_City_Years,' you can create bar plots to visualize the frequency of each category.

In [157…
```
# Number of product category
walmart['Product_Category'].nunique()
```
Out[157]:
```
20
```

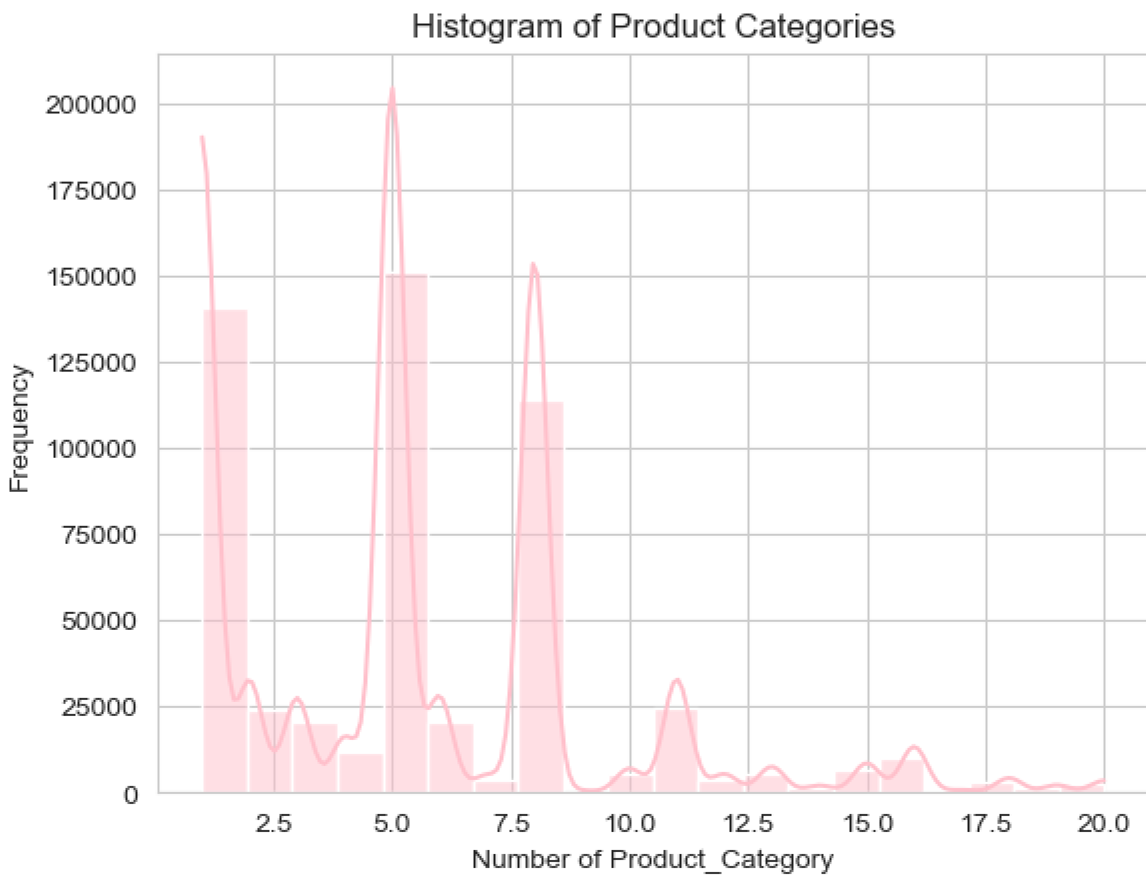
In [158…
```
#Product category names
walmart['Product_Category'].unique()
```
Out[158]:
```
array([ 3,  1, 12,  8,  5,  4,  2,  6, 14, 11, 13, 15,  7, 16, 18, 10, 17,
        9, 20, 19], dtype=int64)
```

In [167…
```
#number of occupation
walmart['Occupation'].nunique()
```
Out[167]:
```
21
```

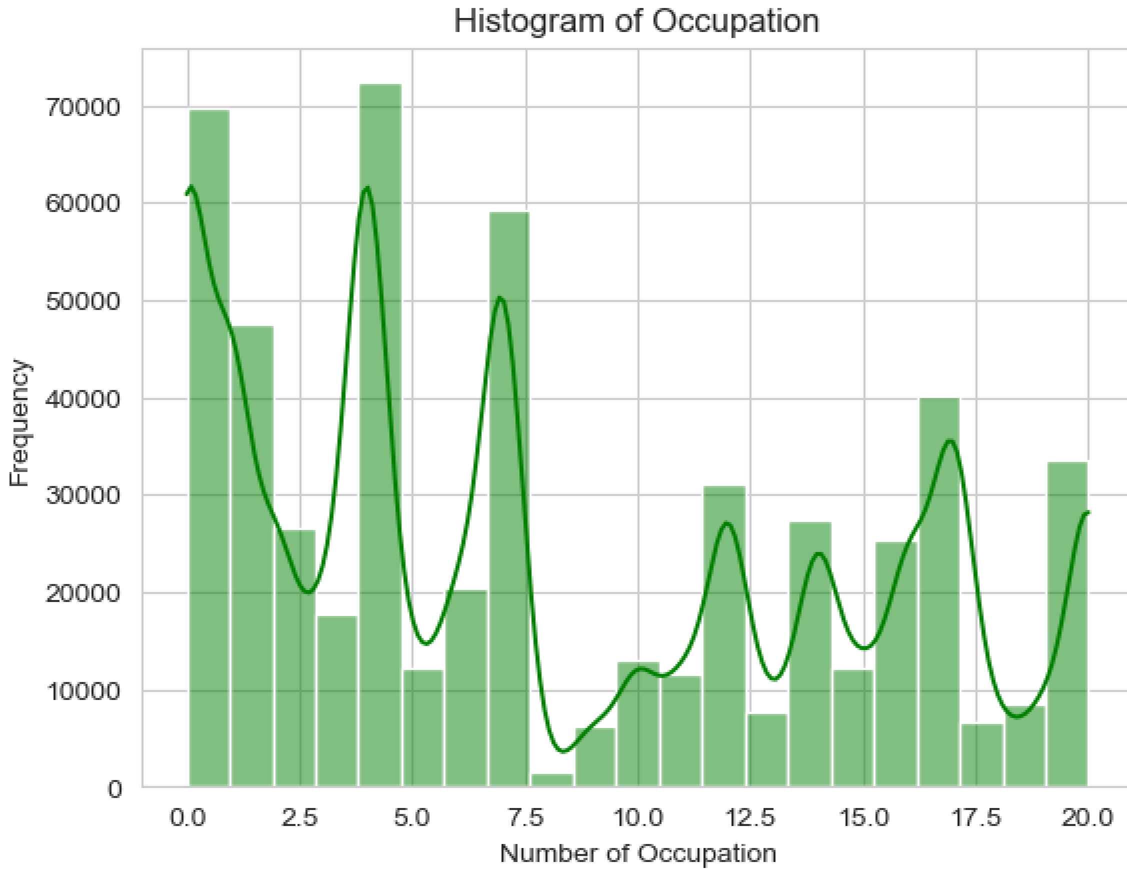
In [168…
```
# Occupation named
walmart['Occupation'].unique()
```

Out[168]:
```
array([10, 16, 15,  7, 20,  9,  1, 12, 17,  0,  3,  4, 11,  8, 19,  2, 18,
        5, 14, 13,  6], dtype=int64)
```

In [166…
```
sns.histplot(x=walmart['Product_Category'], bins=20, kde=True, color="pink")
plt.xlabel('Number of Product_Category')
plt.ylabel('Frequency')
plt.title('Histogram of Product Categories')

# Display the plot
plt.show()
```



In [165…
```
sns.histplot(x=walmart['Occupation'], bins=21,kde=True, color="green")
plt.xlabel('Number of Occupation')
plt.ylabel('Frequency')
plt.title('Histogram of Occupation')

# Display the plot
plt.show()
```



Solution 4

4(1): There is a statistically significant difference.

    Men spend more money per transaction than women.

In [1]:
```
from scipy.stats import ttest_ind
```

In [8]:
```
# Separate data by gender
women_data = walmart[walmart['Gender'] == 'F']
men_data = walmart[walmart['Gender'] == 'M']
```

In [ ]:
```
Null Hypothesis: There is no significant difference in the mean purchase amounts betwe
Alternative Hypothes :Women spend more or less than men on average.
```

In [9]:
```
# Perform the t-test
t_stat, p_value = ttest_ind(women_data['Purchase'], men_data['Purchase'])

# Set the significance level (alpha)
alpha = 0.05

# Make a decision based on the p-value
if p_value < alpha:
```

```
    print("There is a statistically significant difference.")
    if t_stat > 0:
        print("Women spend more money per transaction than men.")
    else:
        print("Men spend more money per transaction than women.")
else:
    print("There is no statistically significant difference.")
```

```
There is a statistically significant difference.
Men spend more money per transaction than women.
```

Solution 4(2)

In [13]:

```
# Calculate mean and standard deviation for women
mean_female = women_data['Purchase'].mean()
std_female = women_data['Purchase'].std()
n_female = len(women_data)
print("Mean", mean_female)
print("std", std_female)
print("female length", n_female)
```

```
Mean 8734.565765155476
std 4767.233289291444
female length 135809
```

In [14]:

```
# Calculate mean and standard deviation for male
mean_male = men_data['Purchase'].mean()
std_male = men_data['Purchase'].std()
n_male = len(men_data)
print("Mean", mean_male)
print("std", std_male)
print("female length", n_male)
```

```
Mean 9437.526040472265
std 5092.186209777949
female length 414259
```

In [16]:

```
import scipy.stats as stats
```

In [18]:

```
# Choose the desired confidence level
confidence_level = 0.95

# Calculate z-critical value for a two-tailed test
z_critical = stats.norm.ppf((1 + confidence_level) / 2)
z_critical
```

Out[18]:

```
1.959963984540054
```

In [20]:

```
# Calculate standard error for both groups
se_female = std_female / (n_female**0.5)
print("female standard error",se_female)
se_male = std_male / (n_male**0.5)
print("male standard error",se_male)
```

```
female standard error 12.936063220950688
male standard error 7.91167247562093
```

In [21]:
```
# Calculate margin of error for both groups
margin_error_female = z_critical * se_female
margin_error_male = z_critical * se_male
```

In [25]:
```
# Calculate the confidence intervals
ci_female = (mean_female - margin_error_female, mean_female + margin_error_female)
ci_male = (mean_male - margin_error_male, mean_male + margin_error_male)
```

Out[25]:
```
(9422.01944736257, 9453.032633581959)
```

In [23]:
```
# Print the results
print("Confidence Interval for Female Customers:", ci_female)
print("Confidence Interval for Male Customers:", ci_male)
```

```
Confidence Interval for Female Customers: (8709.21154714068, 8759.919983170272)
Confidence Interval for Male Customers: (9422.01944736257, 9453.032633581959)
```

Solution 4(3)

Since the confidence intervals for female and male customers do not overlap, it suggests that there is a statistically significant difference between the mean expenses of female and male customers. In this case, it appears that male customers spend more on average, as the lower bound of the male customer's confidence interval is higher than the upper bound of the female customer's confidence interval.

Solution 4(4)

In [27]:
```
walmart.head()
```

Out[27]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_S
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 [ ]:

Null Hypothesis : here is no significant difference in the mean purchase amounts between married and unmarried customers. Alternative Hypothesis : Married customers spend more or less than unmarried customers on average.

In [28]:
```
# Separate data by gender
unmarried_data = walmart[walmart['Marital_Status'] == 0]
```

```
married_data = walmart[walmart['Marital_Status'] == 1]
```

In [41]:
```
walmart['Marital_Status'].value_counts()
```

Out[41]:
```
0    324731
1    225337
Name: Marital_Status, dtype: int64
```

In [51]:
```
# Perform the t-test
t_stat, p_value = ttest_ind(unmarried_data['Purchase'], married_data['Purchase'])
t_stat, p_value
```

Out[51]:
```
(0.3436698055440526, 0.7310947525758316)
```

In [47]:
```
# Set the significance level (alpha)
alpha = 0.05
```

In [53]:
```
# Make a decision based on the p-value
if p_value < alpha:
    print("Alternative Hypothesis:", "There is a statistically significant difference.
else:
    print("Null Hypothesis:", "There is no statistically significant difference in the
```

```
Null Hypothesis: There is no statistically significant difference in the mean purchas
e amounts between married and unmarried customers..
```

In [54]:
```
# Calculate mean and standard deviation for unmarried
mean_unmarried = unmarried_data['Purchase'].mean()
std_unmarried = unmarried_data['Purchase'].std()
n_unmarried = len(unmarried_data)
print("Mean", mean_unmarried)
print("std", std_unmarried)
print("female length", n_unmarried)
```

```
Mean 9265.907618921507
std 5027.347858674457
female length 324731
```

In [55]:
```
# Calculate mean and standard deviation for married
mean_married = married_data['Purchase'].mean()
std_married = married_data['Purchase'].std()
n_married = len(married_data)
print("Mean", mean_married)
print("std", std_married)
print("female length", n_married)
```

```
Mean 9261.174574082374
std 5016.89737779313
female length 225337
```

In [57]:
```
# Choose the desired confidence level
confidence_level = 0.95

# Calculate z-critical value for a two-tailed test
z_critical = stats.norm.ppf((1 + confidence_level) / 2)
z_critical
```

Out[57]:
```
1.959963984540054
```

In [58]:
```
# Calculate standard error for both groups
se_married = std_married/ (n_married**0.5)
print("married standard error",se_married)
se_unmarried = std_unmarried/ (n_unmarried**0.5)
print("unmarried standard error",se_unmarried)
```

```
married standard error 10.568636561021444
unmarried standard error 8.82220330129379
```

In [59]:
```
# Calculate margin of error for both groups
margin_error_married = z_critical * se_married
margin_error_unmarried = z_critical * se_unmarried
```

In [61]:
```
# Calculate the confidence intervals
ci_married = (mean_married - margin_error_married, mean_married + margin_error_marriec
ci_unmarried= (mean_unmarried - margin_error_unmarried, mean_unmarried + margin_error_
```

In [62]:
```
# Print the results
print("Confidence Interval for Married Customers:", ci_married)
print("Confidence Interval for Uarried Customers:", ci_unmarried)
```

```
Confidence Interval for Married Customers: (9240.460427057078, 9281.888721107669)
Confidence Interval for Uarried Customers: (9248.61641818668, 9283.198819656332)
```

Since these two confidence intervals have an overlap in their ranges, it suggests that there is no strong evidence to suggest a significant difference in mean expenses between married and unmarried customers. In other words, the means of these two groups are within a similar range, and any observed differences are not statistically significant at the 95% confidence level.

Solution 4(5)

In [65]:
```
walmart["Age"].unique()
```

Out[65]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

In [79]:
```
walmart["Age"].value_counts()
```

Out[79]:
```
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
```

In [66]:
```
# Define the age groups
age_groups = ['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25']
confidence_level = 0.95  # 95% confidence level
```

In [67]:
```
# Create a dictionary to store results for each age group
age_group_results = {}
```

In [69]:
```
# Loop through age groups and calculate confidence intervals
for age_group in age_groups:
    group_data = walmart[walmart['Age'] == age_group]
    mean_purchase = group_data['Purchase'].mean()
    std_purchase = group_data['Purchase'].std()
    n = len(group_data)
    se = std_purchase / (n**0.5)
```

In [70]:
```
# Calculate the t-critical value
t_critical = stats.t.ppf((1 + confidence_level) / 2, n - 1)
```

In [71]:
```
# Calculate the margin of error
margin_error = t_critical * se
```

In [72]:
```
 # Calculate the confidence interval
lower_bound = mean_purchase - margin_error
upper_bound = mean_purchase + margin_error
```

In [73]:
```
 # Store the results in the dictionary
age_group_results[age_group] = (lower_bound, upper_bound)
```

In [74]:
```
# Print the results
for age_group, ci in age_group_results.items():
    print(f"Confidence Interval for {age_group} Customers: {ci}")
```

```
Confidence Interval for 18-25 Customers: (9138.40756914702, 9200.919643375557)
```

This confidence interval indicates the range in which the true population mean expenses for customers aged 18-25 are likely to fall with 95% confidence.

Solution 6

In [ ]:
```
1. There is a huge difference between number of male and female customers
   Female are  135809
   Male are 414259
2. B ttest found There is a statistically significant difference.
      Men spend more money per transaction than women.
3. Purchase Amount most people purcahse is range of 7000 to 7500.
   50% people buy product of amount 6000 to 12000.
   More than 20000 puchase amount will be considered as outlier very rare people buy t
4. Out of 20 product categories mostly user prefer category number 5.
   50% users prefer category number 2, 3, 4, and 5.
   category above 17.5 opt rarely.
5. Out of city A, B and C. Most users are from city B.
6. Approx 2 Lakhs users living in current city since last 1 years.
7. Most user have occupation number 7. 50% users have occupation 2 to 14(approx)
8. Most number of users are from age group 26-35 which is 219587
```

In [ ]:
```
Thank-You
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: