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

In [2]:
```
df = pd.read_csv('Walmart_data.csv')
```

In [3]:
```
df.head()
```

Out[3]:

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

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

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

In [5]:
```
col = ['User_ID','Product_ID','Gender', 'Age','City_Category','Marital_Stat
df[col] = df[col].astype('category')
```

In [6]:
```
df.dtypes
```

Out[6]:
```
User_ID                       category
Product_ID                    category
Gender                        category
Age                           category
Occupation                       int64
City_Category                 category
Stay_In_Current_City_Years      object
Marital_Status                category
Product_Category                 int64
Purchase                         int64
dtype: object
```

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

Out[7]:
```
(550068, 10)
```

In [8]:
```
df.isna().sum()
```

Out[8]:
```
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 [9]:
```
df.describe()
```

Out[9]:

	Occupation	Product_Category	Purchase
count	550068.000000	550068.000000	550068.000000
mean	8.076707	5.404270	9263.968713
std	6.522660	3.936211	5023.065394
min	0.000000	1.000000	12.000000
25%	2.000000	1.000000	5823.000000
50%	7.000000	5.000000	8047.000000
75%	14.000000	8.000000	12054.000000
max	20.000000	20.000000	23961.000000

In [10]:
```
df.describe(include=['object','category'])
```

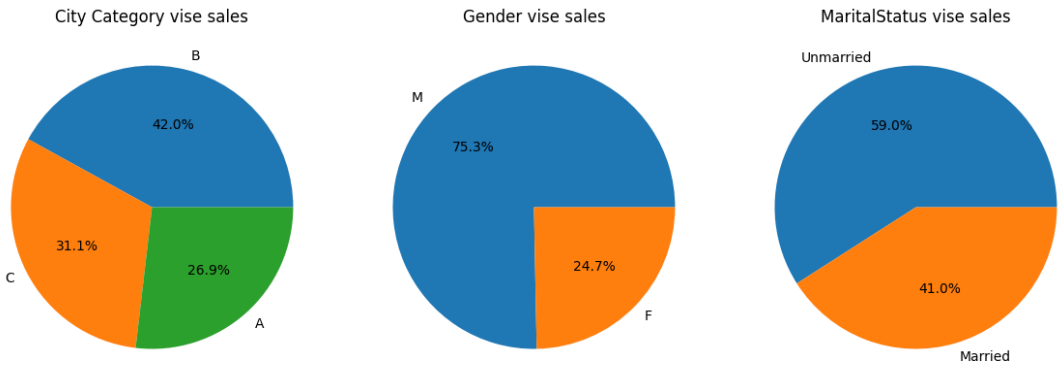
Out[10]:

	User_ID	Product_ID	Gender	Age	City_Category	Stay_In_Current_City_Years	M
count	550068	550068	550068	550068	550068	550068	
unique	5891	3631	2	7	3	5	
top	1001680	P00265242	M	26-35	B	1	
freq	1026	1880	414259	219587	231173	193821	

In [11]:
```
# Display unique attributes for each column
for column in df.columns:
    print(f"\nUnique values for {column}:")
    print(df[column].unique())
```

```
Unique values for User_ID:
[1000001, 1000002, 1000003, 1000004, 1000005, ..., 1004588, 1004871, 1004
113, 1005391, 1001529]
Length: 5891
Categories (5891, int64): [1000001, 1000002, 1000003, 1000004, ..., 10060
37, 1006038, 1006039, 1006040]

Unique values for Product_ID:
['P00069042', 'P00248942', 'P00087842', 'P00085442', 'P00285442', ..., 'P
00375436', 'P00372445', 'P00370293', 'P00371644', 'P00370853']
Length: 3631
Categories (3631, object): ['P00000142', 'P00000242', 'P00000342', 'P0000
0442', ..., 'P0099642',
                            'P0099742', 'P0099842', 'P0099942']

Unique values for Gender:
['F', 'M']
Categories (2, object): ['F', 'M']

Unique values for Age:
['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25']
Categories (7, object): ['0-17', '18-25', '26-35', '36-45', '46-50', '51-
55', '55+']

Unique values for Occupation:
[10 16 15  7 20  9  1 12 17  0  3  4 11  8 19  2 18  5 14 13  6]

Unique values for City_Category:
['A', 'C', 'B']
Categories (3, object): ['A', 'B', 'C']

Unique values for Stay_In_Current_City_Years:
['2' '4+' '3' '1' '0']

Unique values for Marital_Status:
[0, 1]
Categories (2, int64): [0, 1]

Unique values for Product_Category:
[ 3  1 12  8  5  4  2  6 14 11 13 15  7 16 18 10 17  9 20 19]

Unique values for Purchase:
[ 8370 15200  1422 ...   135   123   613]
```

In [12]:
```
plt.figure(figsize=(12,4))

df['Marital_Status'].replace(to_replace = 0, value = 'Unmarried', inplace =
df['Marital_Status'].replace(to_replace = 1, value = 'Married', inplace = T

city_category_counts = df['City_Category'].value_counts()
gender_counts = df['Gender'].value_counts()
marital_counts = df['Marital_Status'].value_counts()

plt.subplot(131)
plt.pie(city_category_counts, labels=city_category_counts.index, autopct='%
plt.title("City Category vise sales")

plt.subplot(132)
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%')
plt.title("Gender vise sales")

plt.subplot(133)
plt.pie(marital_counts, labels=marital_counts.index, autopct='%1.1f%%')
plt.title("MaritalStatus vise sales")

plt.tight_layout()
plt.show()
```

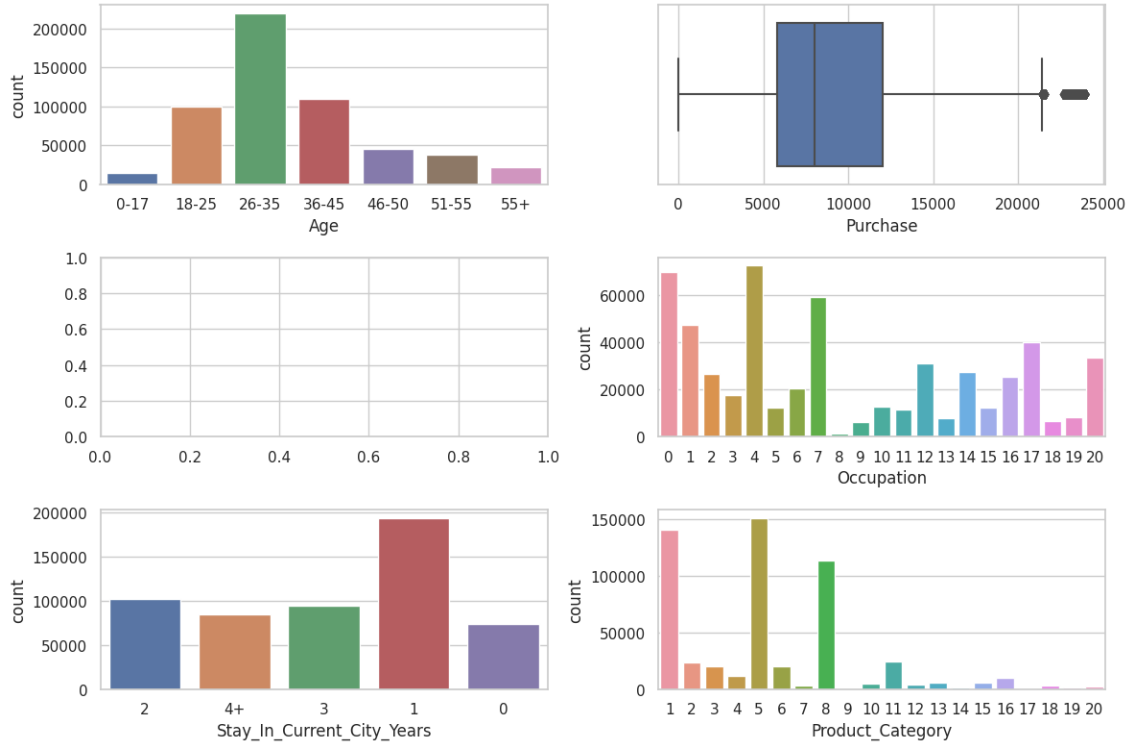


Men account for 75% of the records out of 0.54 million entries, while women make up 25%.

Buyers come from City Category B (42%), Category C (31%), and Category A (27%).

It is evident that 41% of regular buyers are married, whilst 59% of buyers are single.

In [13]:
```
sns.set(style="whitegrid")

fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(12,8))

sns.histplot(data=df, x='Purchase', ax=axes[0, 0], kde=True)
sns.boxplot(data=df, x='Purchase', ax=axes[0,1])

sns.countplot(data=df, x='Age', ax=axes[0,0])
sns.countplot(data=df, x='Occupation', ax=axes[1,1])
sns.countplot(data=df, x='Stay_In_Current_City_Years', ax=axes[2,0])
sns.countplot(data=df, x='Product_Category', ax=axes[2,1])

plt.tight_layout()
plt.show()
```



The age group of 26 to 35 accounts for around 40% of buyers, the greatest percentage of any age group.

We can see that the majority of purchasers are between the ages of 18 and 45.

The majority of buyers have lived in their present city for one, two, and three years, respectively.

While around 8000 was spent by 50% of the buyers.

The majority of order values fall between 5000 and 10,000.

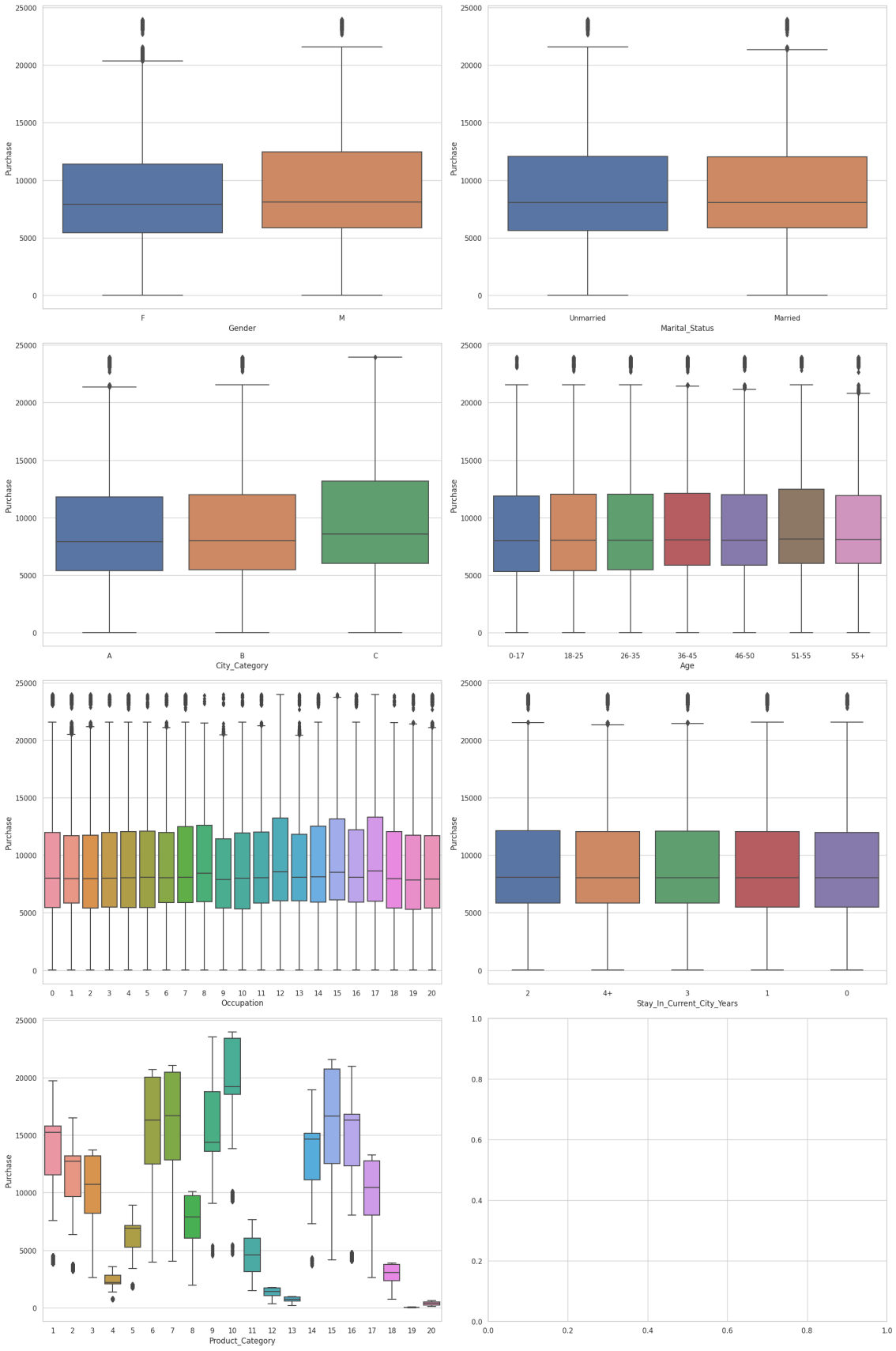
The most frequent purchasers are those in occupations 4 and 7, then 0 and 7.

The least frequent customers are those in occupations 8 and 9, then 18 and 9.

Five is the most often purchased product category, followed by one and eight.

Category 9 is the least frequently purchased, followed by 17 and 14.

In [14]:
```
fig, axes = plt.subplots(nrows=4, ncols=2, figsize=(20, 30))

sns.boxplot(data=df, x='Gender', y='Purchase', ax=axes[0,0])
sns.boxplot(data=df, x='Marital_Status', y='Purchase', ax=axes[0,1])
sns.boxplot(data=df, x='City_Category', y='Purchase', ax=axes[1,0])
sns.boxplot(data=df, x='Age', y='Purchase', ax=axes[1,1])
sns.boxplot(data=df, x='Occupation', y='Purchase', ax=axes[2,0])
sns.boxplot(data=df, x='Stay_In_Current_City_Years', y='Purchase', ax=axes[
sns.boxplot(data=df, x='Product_Category', y='Purchase', ax=axes[3,0])

plt.tight_layout()
plt.show()
```



It is evident that men spend more money than women.

The median purchase values for the various age groups do not appear to differ significantly.

In the data for the purchase vs. occupation graph, there are a lot of outliers.

Comparing occupation 17 to other occupations, occupation 17 has the highest average order values.

With the lowest average order value is occupation 9.

The cities with the greatest median values are City Category C, City B, and City A.

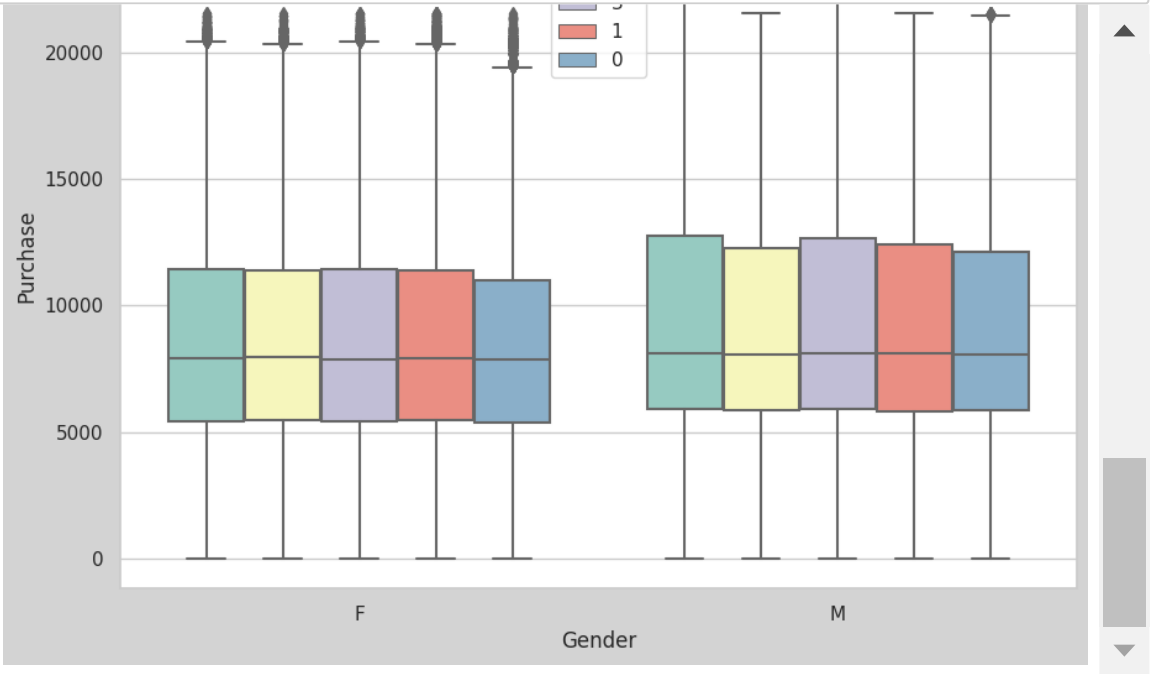
For cities A and B, there are a few outliers.

It is evident that the median value for married and single people is nearly equal.

Category 10 has the greatest average order value of any category.

Category 19 has the lowest average order value of any category.

In [15]:
```
plt.figure(figsize = (12,7))
sns.boxplot(data=df, y='Purchase', x='Gender', hue='City_Category', palette
plt.legend(loc=9)
plt.title("Male vs Female City Category wise Purchase habits")

plt.figure(figsize = (10,7)).set_facecolor("lightgrey")
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Marital_Status', palett
plt.legend(loc=9)
plt.title('Male vs Female Marital Status wise purchase habits')

plt.figure(figsize = (10,7)).set_facecolor("lightgrey")
sns.boxplot(data=df, y='Purchase', x='Gender', hue='Stay_In_Current_City_Ye
plt.legend(loc=9)
plt.title('Male vs Female Stay Years in Current City wise Purchase Habits')

plt.show()
```



The lowest and nearly identical median values are found for girls aged 18 to 25.

The 51–55 age group has the greatest median value, which is about the same for all age groups.

In comparison to cities A and B, the median value for females in city category C is higher.

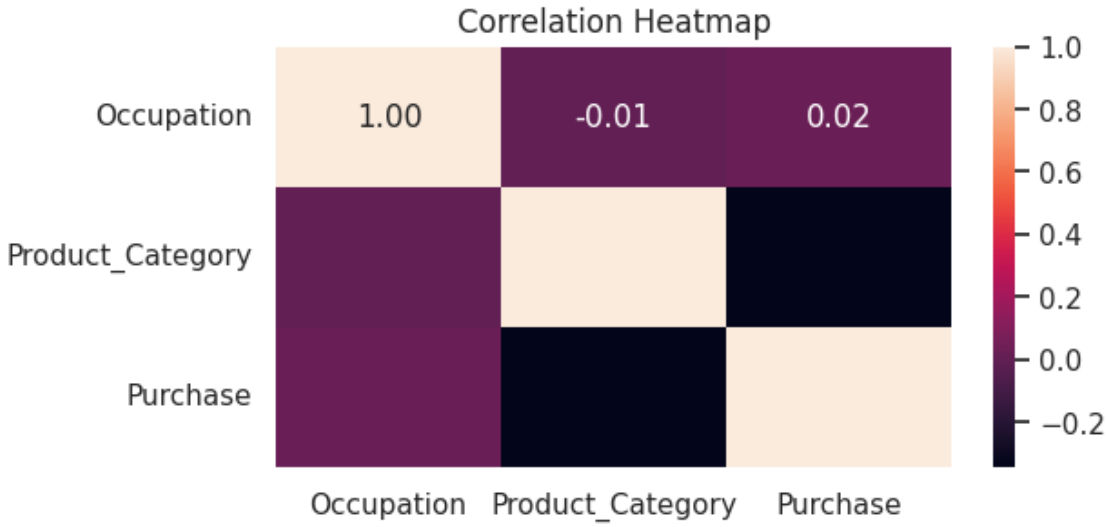
In comparison to cities A and B, the male median value in city category C is likewise the greatest.

Both genders' spending patterns are unaffected by their marital status. However, we can see that the median values for men are higher than those for women.

We can observe for females the median values for purchase amount is a little lower for women staying for 3 and 0 years as compared to others. For men, there is no much

In [16]:
```
plt.figure(figsize=(6, 3))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```

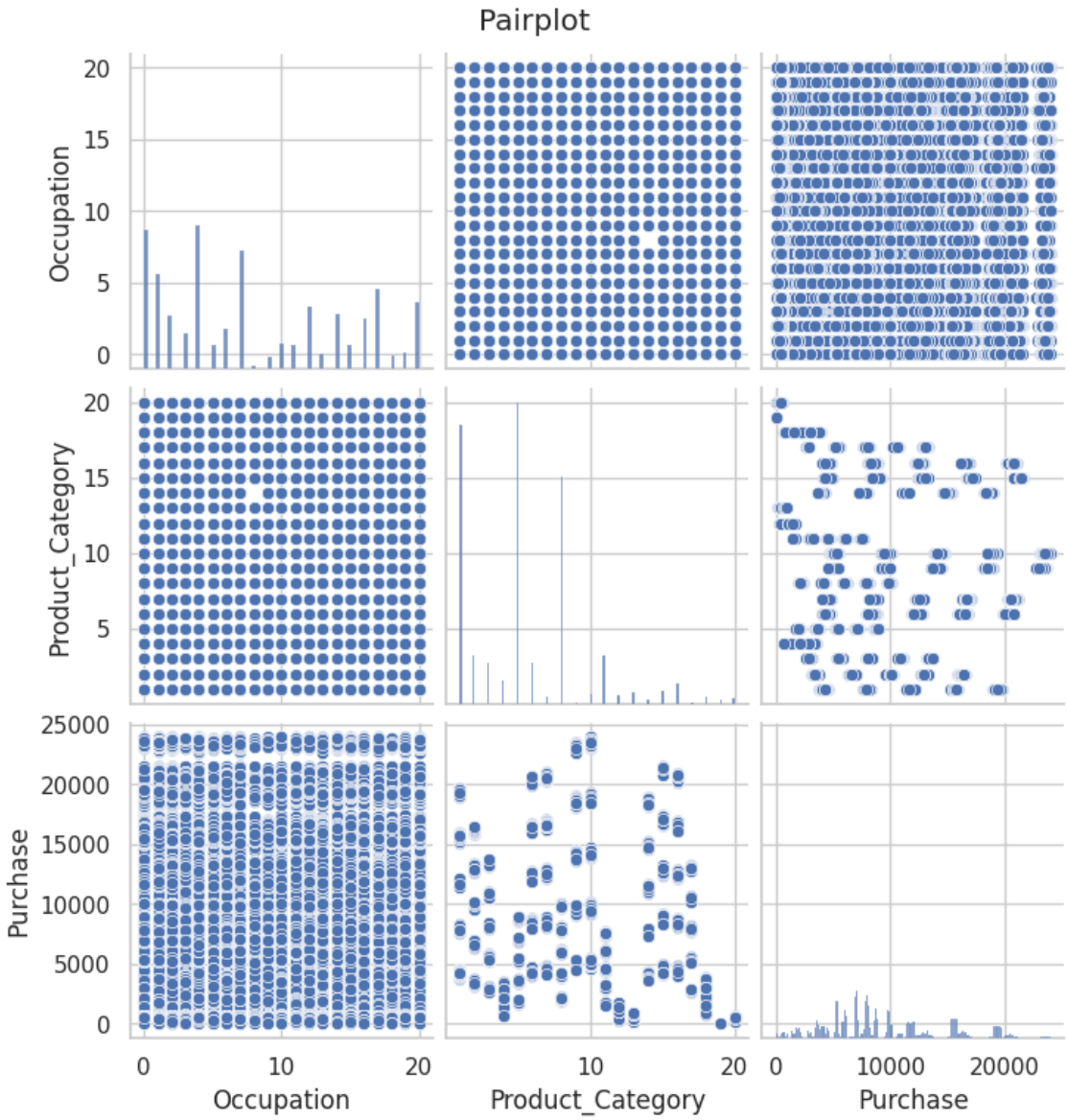
```
<ipython-input-16-4fd6d3f1f9d8>:2: FutureWarning: The default value of nu
meric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numer
ic_only to silence this warning.
  correlation_matrix = df.corr()
```



A little positive correlation was found between occupation and purchase.

Correlation between Product Category and Purchase is negative .

In [17]:
```
sns.pairplot(df, height=2.5)
plt.suptitle('Pairplot', y=1.02)
plt.show()
```



In [18]:
```
from scipy.stats import t

# Separate data for male and female customers
male_data = df[df['Gender'] == 'M']['Purchase']
female_data = df[df['Gender'] == 'F']['Purchase']

# Calculate sample means
mean_male = np.mean(male_data)
mean_female = np.mean(female_data)

# Calculate sample standard deviations
std_dev_male = np.std(male_data, ddof=1)
std_dev_female = np.std(female_data, ddof=1)

# Sample sizes
n_male = len(male_data)
n_female = len(female_data)

# Set confidence level
confidence_level = 0.95

# Calculate standard errors of the means
se_male = std_dev_male / np.sqrt(n_male)
se_female = std_dev_female / np.sqrt(n_female)

# Calculate margin of error
margin_error_male = t.ppf((1 + confidence_level) / 2, n_male - 1) * se_male
margin_error_female = t.ppf((1 + confidence_level) / 2, n_female - 1) * se_

# Calculate confidence intervals
ci_male = (mean_male - margin_error_male, mean_male + margin_error_male)
ci_female = (mean_female - margin_error_female, mean_female + margin_error_

print("Confidence Interval 95% for Mean Purchase Amount (Male):", ci_male)
print("Confidence Interval 95% for Mean Purchase Amount (Female):", ci_fema
```

```
Confidence Interval 95% for Mean Purchase Amount (Male): (9422.0194020558
14, 9453.032678888716)
Confidence Interval 95% for Mean Purchase Amount (Female): (8709.21132117
373, 8759.92020913722)
```

In [19]:
```
# Check if confidence intervals overlap
confidence_interval_overlap = ci_male[1] >= ci_female[0] and ci_female[1] >

# Print the result
print("Do Confidence Intervals at 95% Overlap? ", confidence_interval_overl
```

```
Do Confidence Intervals at 95% Overlap?  False
```

The confidence interval at 95% for both Male and Female is not overlapping.

In [20]:
```
# Set confidence level
confidence_level = 0.90

# Calculate standard errors of the means
se_male = std_dev_male / np.sqrt(n_male)
se_female = std_dev_female / np.sqrt(n_female)

# Calculate margin of error
margin_error_male = t.ppf((1 + confidence_level) / 2, n_male - 1) * se_male
margin_error_female = t.ppf((1 + confidence_level) / 2, n_female - 1) * se_

# Calculate confidence intervals
ci_male = (mean_male - margin_error_male, mean_male + margin_error_male)
ci_female = (mean_female - margin_error_female, mean_female + margin_error_

print("Confidence Interval at 90% for Mean Purchase Amount (Male):", ci_mal
print("Confidence Interval at 90% for Mean Purchase Amount (Female):", ci_f
```

```
Confidence Interval at 90% for Mean Purchase Amount (Male): (9424.5124682
03842, 9450.539612740688)
Confidence Interval at 90% for Mean Purchase Amount (Female): (8713.28768
9504074, 8755.843840806878)
```

In [21]:
```
# Check if confidence intervals overlap
confidence_interval_overlap = ci_male[1] >= ci_female[0] and ci_female[1] >

# Print the result
print("Do Confidence Intervals at 90% Overlap? ", confidence_interval_overl
```

```
Do Confidence Intervals at 90% Overlap?  False
```

The confidence interval at 90% for both Male and Female is not overlapping.

In [22]:
```
# Set confidence level
confidence_level = 0.99

# Calculate standard errors of the means
se_male = std_dev_male / np.sqrt(n_male)
se_female = std_dev_female / np.sqrt(n_female)

# Calculate margin of error
margin_error_male = t.ppf((1 + confidence_level) / 2, n_male - 1) * se_male
margin_error_female = t.ppf((1 + confidence_level) / 2, n_female - 1) * se_

# Calculate confidence intervals
ci_male = (mean_male - margin_error_male, mean_male + margin_error_male)
ci_female = (mean_female - margin_error_female, mean_female + margin_error_

print("Confidence Interval at 99% for Mean Purchase Amount (Male):", ci_mal
print("Confidence Interval at 99% for Mean Purchase Amount (Female):", ci_f
```

```
Confidence Interval at 99% for Mean Purchase Amount (Male): (9417.1468287
7079, 9457.90525217374)
Confidence Interval at 99% for Mean Purchase Amount (Female): (8701.24420
611832, 8767.887324192632)
```

In [23]:
```
# Check if confidence intervals overlap
confidence_interval_overlap = ci_male[1] >= ci_female[0] and ci_female[1] >

# Print the result
print("Do Confidence Intervals at 99% Overlap? ", confidence_interval_over
```

```
Do Confidence Intervals at 99% Overlap?  False
```

The confidence interval at 99% for both Male and Female is not overlapping.

In [24]:
```
# Separate data for married and unmarried customers
married_data = df[df['Marital_Status'] == 'Married']['Purchase']
unmarried_data = df[df['Marital_Status'] == 'Unmarried']['Purchase']

# Calculate sample means
mean_married = np.mean(married_data)
mean_unmarried = np.mean(unmarried_data)

# Calculate sample standard deviations
std_dev_married = np.std(married_data, ddof=1)
std_dev_unmarried = np.std(unmarried_data, ddof=1)

# Sample sizes
n_married = len(married_data)
n_unmarried = len(unmarried_data)

# Set confidence level
confidence_level = 0.95

# Calculate standard errors of the means
se_married = std_dev_married / np.sqrt(n_married)
se_unmarried = std_dev_unmarried / np.sqrt(n_unmarried)

# Calculate margin of error
margin_error_married = t.ppf((1 + confidence_level) / 2, n_married - 1) * s
margin_error_unmarried = t.ppf((1 + confidence_level) / 2, n_unmarried - 1)

# Calculate confidence intervals
ci_married = (mean_married - margin_error_married, mean_married + margin_er
ci_unmarried = (mean_unmarried - margin_error_unmarried, mean_unmarried + m

print("Confidence Interval at 95% for Mean Purchase Amount (Married):", ci_
print("Confidence Interval at 95% for Mean Purchase Amount (Unmarried):", c
```

```
Confidence Interval at 95% for Mean Purchase Amount (Married): (9240.4603
15792989, 9281.888832371758)
Confidence Interval at 95% for Mean Purchase Amount (Unmarried): (9248.61
6353737028, 9283.198884105985)
```

In [25]:
```
# Check if confidence intervals overlap
confidence_interval_overlap = ci_married[1] >= ci_unmarried[0] and ci_unmar

# Print the result
print("Do Confidence Intervals at 95% Overlap? ", confidence_interval_overl
```

```
Do Confidence Intervals at 95% Overlap?  True
```

The confidence interval at 95% for married and unmarried is overlapping.

In [26]:
```
# Set confidence level
confidence_level = 0.99

# Calculate standard errors of the means
se_married = std_dev_married / np.sqrt(n_married)
se_unmarried = std_dev_unmarried / np.sqrt(n_unmarried)

# Calculate margin of error
margin_error_married = t.ppf((1 + confidence_level) / 2, n_married - 1) * s
margin_error_unmarried = t.ppf((1 + confidence_level) / 2, n_unmarried - 1)

# Calculate confidence intervals
ci_married = (mean_married - margin_error_married, mean_married + margin_er
ci_unmarried = (mean_unmarried - margin_error_unmarried, mean_unmarried + m

print("Confidence Interval at 99% for Mean Purchase Amount (Married):", ci_
print("Confidence Interval at 99% for Mean Purchase Amount (Unmarried):", c
```

```
Confidence Interval at 99% for Mean Purchase Amount (Married): (9233.9513
39733765, 9288.397808430982)
Confidence Interval at 99% for Mean Purchase Amount (Unmarried): (9243.18
2995563593, 9288.63224227942)
```

In [27]:
```
# Check if confidence intervals overlap
confidence_interval_overlap = ci_married[1] >= ci_unmarried[0] and ci_unmar

# Print the result
print("Do Confidence Intervals at 99% Overlap? ", confidence_interval_overl
```

```
Do Confidence Intervals at 99% Overlap?  True
```

The confidence interval at 99% for married and unmarried is overlapping.

In [28]:
```
# Set confidence level
confidence_level = 0.90

# Calculate standard errors of the means
se_married = std_dev_married / np.sqrt(n_married)
se_unmarried = std_dev_unmarried / np.sqrt(n_unmarried)

# Calculate margin of error
margin_error_married = t.ppf((1 + confidence_level) / 2, n_married - 1) * s
margin_error_unmarried = t.ppf((1 + confidence_level) / 2, n_unmarried - 1)

# Calculate confidence intervals
ci_married = (mean_married - margin_error_married, mean_married + margin_er
ci_unmarried = (mean_unmarried - margin_error_unmarried, mean_unmarried + m

print("Confidence Interval at 90% for Mean Purchase Amount (Married):", ci_
print("Confidence Interval at 90% for Mean Purchase Amount (Unmarried):", c
```

```
Confidence Interval at 90% for Mean Purchase Amount (Married): (9243.7906
4243542, 9278.558505729326)
Confidence Interval at 90% for Mean Purchase Amount (Unmarried): (9251.39
6344426079, 9280.418893416934)
```

In [29]:
```
# Check if confidence intervals overlap
confidence_interval_overlap = ci_married[1] >= ci_unmarried[0] and ci_unmar

# Print the result
print("Do Confidence Intervals at 90% Overlap? ", confidence_interval_overl
```

```
Do Confidence Intervals at 90% Overlap?  True
```

The confidence interval at 90% for married and unmarried is overlapping.

There is no effect of their marital status on their purchases.

In [30]:
```
# Separate data for different age groups
age_groups = df['Age'].unique()

# Create an empty dictionary to store confidence intervals
age_ci_dict = {}

# Set confidence level
confidence_level = 0.95

# Calculate confidence intervals for each age group
for age_group in age_groups:
    age_data = df[df['Age'] == age_group]['Purchase']
    mean_age = np.mean(age_data)
    std_dev_age = np.std(age_data, ddof=1)
    n_age = len(age_data)
    se_age = std_dev_age / np.sqrt(n_age)
    margin_error_age = t.ppf((1 + confidence_level) / 2, n_age - 1) * se_ag
    ci_age = (mean_age - margin_error_age, mean_age + margin_error_age)
    age_ci_dict[age_group] = ci_age

# Print confidence intervals for each age group
for age_group, ci_age in age_ci_dict.items():
    print(f"Confidence Interval for Mean Purchase Amount ({age_group}):", c
```

```
Confidence Interval for Mean Purchase Amount (0-17): (8851.941436361221,
9014.987844528727)
Confidence Interval for Mean Purchase Amount (55+): (9269.295063935433, 9
403.265854963376)
Confidence Interval for Mean Purchase Amount (26-35): (9231.733560884022,
9273.647704855754)
Confidence Interval for Mean Purchase Amount (46-50): (9163.08393647555,
9254.167458461105)
Confidence Interval for Mean Purchase Amount (51-55): (9483.989875153999,
9585.626186766473)
Confidence Interval for Mean Purchase Amount (36-45): (9301.669084404875,
9361.032305430872)
Confidence Interval for Mean Purchase Amount (18-25): (9138.40756914702,
9200.919643375557)
```

In [31]:
```
# Check if confidence intervals overlap
overlap_check = False
for age_group1, ci_age1 in age_ci_dict.items():
    for age_group2, ci_age2 in age_ci_dict.items():
        if age_group1 != age_group2:
            overlap = ci_age1[1] >= ci_age2[0] and ci_age2[1] >= ci_age1[0]
            print(f"Do Confidence Intervals Overlap? ({age_group1} vs {age_
            if overlap:
                overlap_check = True

# Print overall result
if overlap_check:
    print("There is an overlap in confidence intervals between at least two
else:
    print("There is no overlap in confidence intervals between any two age
```

```
Do Confidence Intervals Overlap? (0-17 vs 55+): False
Do Confidence Intervals Overlap? (0-17 vs 26-35): False
Do Confidence Intervals Overlap? (0-17 vs 46-50): False
Do Confidence Intervals Overlap? (0-17 vs 51-55): False
Do Confidence Intervals Overlap? (0-17 vs 36-45): False
Do Confidence Intervals Overlap? (0-17 vs 18-25): False
Do Confidence Intervals Overlap? (55+ vs 0-17): False
Do Confidence Intervals Overlap? (55+ vs 26-35): True
Do Confidence Intervals Overlap? (55+ vs 46-50): False
Do Confidence Intervals Overlap? (55+ vs 51-55): False
Do Confidence Intervals Overlap? (55+ vs 36-45): True
Do Confidence Intervals Overlap? (55+ vs 18-25): False
Do Confidence Intervals Overlap? (26-35 vs 0-17): False
Do Confidence Intervals Overlap? (26-35 vs 55+): True
Do Confidence Intervals Overlap? (26-35 vs 46-50): True
Do Confidence Intervals Overlap? (26-35 vs 51-55): False
Do Confidence Intervals Overlap? (26-35 vs 36-45): False
Do Confidence Intervals Overlap? (26-35 vs 18-25): False
Do Confidence Intervals Overlap? (46-50 vs 0-17): False
Do Confidence Intervals Overlap? (46-50 vs 55+): False
Do Confidence Intervals Overlap? (46-50 vs 26-35): True
Do Confidence Intervals Overlap? (46-50 vs 51-55): False
Do Confidence Intervals Overlap? (46-50 vs 36-45): False
Do Confidence Intervals Overlap? (46-50 vs 18-25): True
Do Confidence Intervals Overlap? (51-55 vs 0-17): False
Do Confidence Intervals Overlap? (51-55 vs 55+): False
Do Confidence Intervals Overlap? (51-55 vs 26-35): False
Do Confidence Intervals Overlap? (51-55 vs 46-50): False
Do Confidence Intervals Overlap? (51-55 vs 36-45): False
Do Confidence Intervals Overlap? (51-55 vs 18-25): False
Do Confidence Intervals Overlap? (36-45 vs 0-17): False
Do Confidence Intervals Overlap? (36-45 vs 55+): True
Do Confidence Intervals Overlap? (36-45 vs 26-35): False
Do Confidence Intervals Overlap? (36-45 vs 46-50): False
Do Confidence Intervals Overlap? (36-45 vs 51-55): False
Do Confidence Intervals Overlap? (36-45 vs 18-25): False
Do Confidence Intervals Overlap? (18-25 vs 0-17): False
Do Confidence Intervals Overlap? (18-25 vs 55+): False
Do Confidence Intervals Overlap? (18-25 vs 26-35): False
Do Confidence Intervals Overlap? (18-25 vs 46-50): True
Do Confidence Intervals Overlap? (18-25 vs 51-55): False
Do Confidence Intervals Overlap? (18-25 vs 36-45): False
There is an overlap in confidence intervals between at least two age grou
ps.
```

In [32]:
```
# Separate data for different age groups
age_groups = df['Age'].unique()

# Create an empty dictionary to store confidence intervals
age_ci_dict = {}

# Set confidence level
confidence_level = 0.99

# Calculate confidence intervals for each age group
for age_group in age_groups:
    age_data = df[df['Age'] == age_group]['Purchase']
    mean_age = np.mean(age_data)
    std_dev_age = np.std(age_data, ddof=1)
    n_age = len(age_data)
    se_age = std_dev_age / np.sqrt(n_age)
    margin_error_age = t.ppf((1 + confidence_level) / 2, n_age - 1) * se_ag
    ci_age = (mean_age - margin_error_age, mean_age + margin_error_age)
    age_ci_dict[age_group] = ci_age

# Print confidence intervals for each age group
for age_group, ci_age in age_ci_dict.items():
    print(f"Confidence Interval for Mean Purchase Amount ({age_group}):", c
```

```
Confidence Interval for Mean Purchase Amount (0-17): (8826.320033768494,
9040.609247121454)
Confidence Interval for Mean Purchase Amount (55+): (9248.243867862855, 9
424.317051035954)
Confidence Interval for Mean Purchase Amount (26-35): (9225.148284007466,
9280.23298173231)
Confidence Interval for Mean Purchase Amount (46-50): (9148.772763375606,
9268.478631561049)
Confidence Interval for Mean Purchase Amount (51-55): (9468.020441793446,
9601.595620127026)
Confidence Interval for Mean Purchase Amount (36-45): (9292.34219880095,
9370.359191034797)
Confidence Interval for Mean Purchase Amount (18-25): (9128.585922624949,
9210.741289897629)
```

In [33]:
```
# Check if confidence intervals overlap
overlap_check = False
for age_group1, ci_age1 in age_ci_dict.items():
    for age_group2, ci_age2 in age_ci_dict.items():
        if age_group1 != age_group2:
            overlap = ci_age1[1] >= ci_age2[0] and ci_age2[1] >= ci_age1[0]
            print(f"Do Confidence Intervals Overlap? ({age_group1} vs {age_
            if overlap:
                overlap_check = True

# Print overall result
if overlap_check:
    print("There is an overlap in confidence intervals between at least two
else:
    print("There is no overlap in confidence intervals between any two age
```

```
Do Confidence Intervals Overlap? (0-17 vs 55+): False
Do Confidence Intervals Overlap? (0-17 vs 26-35): False
Do Confidence Intervals Overlap? (0-17 vs 46-50): False
Do Confidence Intervals Overlap? (0-17 vs 51-55): False
Do Confidence Intervals Overlap? (0-17 vs 36-45): False
Do Confidence Intervals Overlap? (0-17 vs 18-25): False
Do Confidence Intervals Overlap? (55+ vs 0-17): False
Do Confidence Intervals Overlap? (55+ vs 26-35): True
Do Confidence Intervals Overlap? (55+ vs 46-50): True
Do Confidence Intervals Overlap? (55+ vs 51-55): False
Do Confidence Intervals Overlap? (55+ vs 36-45): True
Do Confidence Intervals Overlap? (55+ vs 18-25): False
Do Confidence Intervals Overlap? (26-35 vs 0-17): False
Do Confidence Intervals Overlap? (26-35 vs 55+): True
Do Confidence Intervals Overlap? (26-35 vs 46-50): True
Do Confidence Intervals Overlap? (26-35 vs 51-55): False
Do Confidence Intervals Overlap? (26-35 vs 36-45): False
Do Confidence Intervals Overlap? (26-35 vs 18-25): False
Do Confidence Intervals Overlap? (46-50 vs 0-17): False
Do Confidence Intervals Overlap? (46-50 vs 55+): True
Do Confidence Intervals Overlap? (46-50 vs 26-35): True
Do Confidence Intervals Overlap? (46-50 vs 51-55): False
Do Confidence Intervals Overlap? (46-50 vs 36-45): False
Do Confidence Intervals Overlap? (46-50 vs 18-25): True
Do Confidence Intervals Overlap? (51-55 vs 0-17): False
Do Confidence Intervals Overlap? (51-55 vs 55+): False
Do Confidence Intervals Overlap? (51-55 vs 26-35): False
Do Confidence Intervals Overlap? (51-55 vs 46-50): False
Do Confidence Intervals Overlap? (51-55 vs 36-45): False
Do Confidence Intervals Overlap? (51-55 vs 18-25): False
Do Confidence Intervals Overlap? (36-45 vs 0-17): False
Do Confidence Intervals Overlap? (36-45 vs 55+): True
Do Confidence Intervals Overlap? (36-45 vs 26-35): False
Do Confidence Intervals Overlap? (36-45 vs 46-50): False
Do Confidence Intervals Overlap? (36-45 vs 51-55): False
Do Confidence Intervals Overlap? (36-45 vs 18-25): False
Do Confidence Intervals Overlap? (18-25 vs 0-17): False
Do Confidence Intervals Overlap? (18-25 vs 55+): False
Do Confidence Intervals Overlap? (18-25 vs 26-35): False
Do Confidence Intervals Overlap? (18-25 vs 46-50): True
Do Confidence Intervals Overlap? (18-25 vs 51-55): False
Do Confidence Intervals Overlap? (18-25 vs 36-45): False
There is an overlap in confidence intervals between at least two age grou
ps.
```

In [34]:
```
# Separate data for different age groups
age_groups = df['Age'].unique()

# Create an empty dictionary to store confidence intervals
age_ci_dict = {}

# Set confidence level
confidence_level = 0.90

# Calculate confidence intervals for each age group
for age_group in age_groups:
    age_data = df[df['Age'] == age_group]['Purchase']
    mean_age = np.mean(age_data)
    std_dev_age = np.std(age_data, ddof=1)
    n_age = len(age_data)
    se_age = std_dev_age / np.sqrt(n_age)
    margin_error_age = t.ppf((1 + confidence_level) / 2, n_age - 1) * se_ag
    ci_age = (mean_age - margin_error_age, mean_age + margin_error_age)
    age_ci_dict[age_group] = ci_age

# Print confidence intervals for each age group
for age_group, ci_age in age_ci_dict.items():
    print(f"Confidence Interval for Mean Purchase Amount ({age_group}):", c
```

```
Confidence Interval for Mean Purchase Amount (0-17): (8865.049497531349,
9001.8797833586)
Confidence Interval for Mean Purchase Amount (55+): (9280.065285868366, 9
392.495633030443)
Confidence Interval for Mean Purchase Amount (26-35): (9235.102926382391,
9270.278339357385)
Confidence Interval for Mean Purchase Amount (46-50): (9170.406084331049,
9246.845310605606)
Confidence Interval for Mean Purchase Amount (51-55): (9492.160404787175,
9577.455657133296)
Confidence Interval for Mean Purchase Amount (36-45): (9306.441166444858,
9356.26022339089)
Confidence Interval for Mean Purchase Amount (18-25): (9143.432787777778,
9195.8944247448)
```

In [35]:
```
# Check if confidence intervals overlap
overlap_check = False
for age_group1, ci_age1 in age_ci_dict.items():
    for age_group2, ci_age2 in age_ci_dict.items():
        if age_group1 != age_group2:
            overlap = ci_age1[1] >= ci_age2[0] and ci_age2[1] >= ci_age1[0]
            print(f"Do Confidence Intervals Overlap? ({age_group1} vs {age_
            if overlap:
                overlap_check = True

# Print overall result
if overlap_check:
    print("There is an overlap in confidence intervals between at least two
else:
    print("There is no overlap in confidence intervals between any two age
```

```
Do Confidence Intervals Overlap? (0-17 vs 55+): False
Do Confidence Intervals Overlap? (0-17 vs 26-35): False
Do Confidence Intervals Overlap? (0-17 vs 46-50): False
Do Confidence Intervals Overlap? (0-17 vs 51-55): False
Do Confidence Intervals Overlap? (0-17 vs 36-45): False
Do Confidence Intervals Overlap? (0-17 vs 18-25): False
Do Confidence Intervals Overlap? (55+ vs 0-17): False
Do Confidence Intervals Overlap? (55+ vs 26-35): False
Do Confidence Intervals Overlap? (55+ vs 46-50): False
Do Confidence Intervals Overlap? (55+ vs 51-55): False
Do Confidence Intervals Overlap? (55+ vs 36-45): True
Do Confidence Intervals Overlap? (55+ vs 18-25): False
Do Confidence Intervals Overlap? (26-35 vs 0-17): False
Do Confidence Intervals Overlap? (26-35 vs 55+): False
Do Confidence Intervals Overlap? (26-35 vs 46-50): True
Do Confidence Intervals Overlap? (26-35 vs 51-55): False
Do Confidence Intervals Overlap? (26-35 vs 36-45): False
Do Confidence Intervals Overlap? (26-35 vs 18-25): False
Do Confidence Intervals Overlap? (46-50 vs 0-17): False
Do Confidence Intervals Overlap? (46-50 vs 55+): False
Do Confidence Intervals Overlap? (46-50 vs 26-35): True
Do Confidence Intervals Overlap? (46-50 vs 51-55): False
Do Confidence Intervals Overlap? (46-50 vs 36-45): False
Do Confidence Intervals Overlap? (46-50 vs 18-25): True
Do Confidence Intervals Overlap? (51-55 vs 0-17): False
Do Confidence Intervals Overlap? (51-55 vs 55+): False
Do Confidence Intervals Overlap? (51-55 vs 26-35): False
Do Confidence Intervals Overlap? (51-55 vs 46-50): False
Do Confidence Intervals Overlap? (51-55 vs 36-45): False
Do Confidence Intervals Overlap? (51-55 vs 18-25): False
Do Confidence Intervals Overlap? (36-45 vs 0-17): False
Do Confidence Intervals Overlap? (36-45 vs 55+): True
Do Confidence Intervals Overlap? (36-45 vs 26-35): False
Do Confidence Intervals Overlap? (36-45 vs 46-50): False
Do Confidence Intervals Overlap? (36-45 vs 51-55): False
Do Confidence Intervals Overlap? (36-45 vs 18-25): False
Do Confidence Intervals Overlap? (18-25 vs 0-17): False
Do Confidence Intervals Overlap? (18-25 vs 55+): False
Do Confidence Intervals Overlap? (18-25 vs 26-35): False
Do Confidence Intervals Overlap? (18-25 vs 46-50): True
Do Confidence Intervals Overlap? (18-25 vs 51-55): False
Do Confidence Intervals Overlap? (18-25 vs 36-45): False
There is an overlap in confidence intervals between at least two age grou
ps.
```

We can say that age group does not have much effect on the spending of customers as their interval range is overalpping with 90%, 95% and 99% confidence intervals.

Final Insights

Nearly 80% of users are in the 18–50 age range (40%: 26–35, 18%: 18–25, 20%: 36-45)

Males make up 75% of users, while females make up 25%. Clearly, men buy more than women do.

41% customers are married & 59% customers are single.

35% of people stay in the city for a year, 18% for two years, and 17% for three years.

The bulk of our clients are from City Category B, however City Category C clients spend more on average—9719.

Though more visitors from City Category B prefer to make purchases, the majority of users are from City Category C, suggesting that the same users visit the mall more than once in City Category B.

The majority of customers make purchases between 5000 and 20000.

The majority of mall patrons are in the 26–35 age range. An estimated 60% of purchases are done by those in the 26–45 age range.

Of all client purchases, City Category B accounts for 42%, City Category C for 31%, and City Category A for 27%.Purchases in city category C are substantial.

The majority of mall patrons are in the 26–35 age range. Customers in city category C are primarily between the ages of 18 and 45.

There are somewhat more female clients in City Category C.

Product numbers 5 and 8 are typical for women.

Recommendations

Men spend more money than women do, thus businesses should concentrate on attracting and keeping more men as clients.

Product Category: The most often purchased items are 1, 5, 8, and 11. It indicates that buyers enjoy these products more than others in these categories. The company can concentrate on selling more of these products or on selling more of the less frequently purchased products.

Customers who are single spend more money than those who are married, so businesses should concentrate on attracting single clients.

Consumers between the ages of 18 and 45 spend more money than other groups of consumers, thus businesses should concentrate on attracting these clients.

Selling more products in City_Category C will help the business boost revenue since male clients there spend more money than male customers in B or C.

Given that women spend on average less than men do, managers must pay different attention to the demands of women. Women's Black Friday spending can be increased by adding a few more offerings.

In order to enhance sales, management could develop some games in the mall to draw in younger customers.

To boost sales, the management ought to make certain deals for children ages 0 to 17.

They may provide some younger-oriented games to draw in more youthful customers.