In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
data = pd.read_csv('walmart_data.csv')
```

In [5]:
```
data.head()
```

Out[5]:

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

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.

Features of the dataset:

Feature	Description
User_ID	User ID of the Customer
Product ID	Product ID of the Purchased Product
Gender	Gender of the Customer (Male/Female)
Age	Age of the Customer (in bins)
Occupation	Occupation of the Customer (Masked)
City_Category	Category of the City (A,B,C)
StayInCurrentCityYears	Number of years stay in current city
Marital_Status	Marital Status (0 - Unmarried / 1 - Married)
ProductCategory	Product Category (Masked)
Purchase	Purchase Amount

Exploratory Data Analysis

In [7]:
```
data.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 [10]:
```
data.shape
```

Out[10]: (550068, 10)

In [11]:
```
print(f"TOTAL ROWS : {data.shape[0]}")
print(f"TOTAL COLUMNS : {data.shape[1]}")
```

```
TOTAL ROWS : 550068
TOTAL COLUMNS : 10
```

In [12]:
```
print(f"SIZE OF DataFrame : {data.size}")
```

```
SIZE OF DataFrame : 5500680
```

In [27]:
```
print(f"Index of the DataFrame : {data.index}")
```

```
Index of the DataFrame : RangeIndex(start=0, stop=550068, step=1)
```

In [14]:
```
print(f"Coulumns : {data.columns}")
```

```
Coulumns : Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

In [15]:
```
data.describe()
```

Out[15]:

	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 [16]:
```
data.describe().T
```

Out[16]:

	count	mean	std	min	25%	50%	75%	max
User_ID	550068.0	1.003029e+06	1727.591586	1000001.0	1001516.0	1003077.0	1004478.0	1006040.0
Occupation	550068.0	8.076707e+00	6.522660	0.0	2.0	7.0	14.0	20.0
Marital_Status	550068.0	4.096530e-01	0.491770	0.0	0.0	0.0	1.0	1.0
Product_Category	550068.0	5.404270e+00	3.936211	1.0	1.0	5.0	8.0	20.0
Purchase	550068.0	9.263969e+03	5023.065394	12.0	5823.0	8047.0	12054.0	23961.0

In [17]:
```
data.isnull()
```

Out[17]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purc
0	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	
550063	False	False	False	False	False	False	False	False	False	
550064	False	False	False	False	False	False	False	False	False	
550065	False	False	False	False	False	False	False	False	False	
550066	False	False	False	False	False	False	False	False	False	
550067	False	False	False	False	False	False	False	False	False	

550068 rows × 10 columns

In [18]:
```
data.isnull().sum()
```

Out[18]:
```
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
```

Note: There are zero null values in the dataset

In [19]:
```
data.duplicated()
```

Out[19]:
```
0         False
1         False
2         False
3         False
4         False
          ...
550063    False
550064    False
550065    False
550066    False
550067    False
Length: 550068, dtype: bool
```

In [20]:
```
data.duplicated().sum()
```

Out[20]:
```
0
```

Note: There are zero duplicate values in the dataset

In [21]:
```
#Taking a copy of data into data_copy
data_copy = data.copy()
```

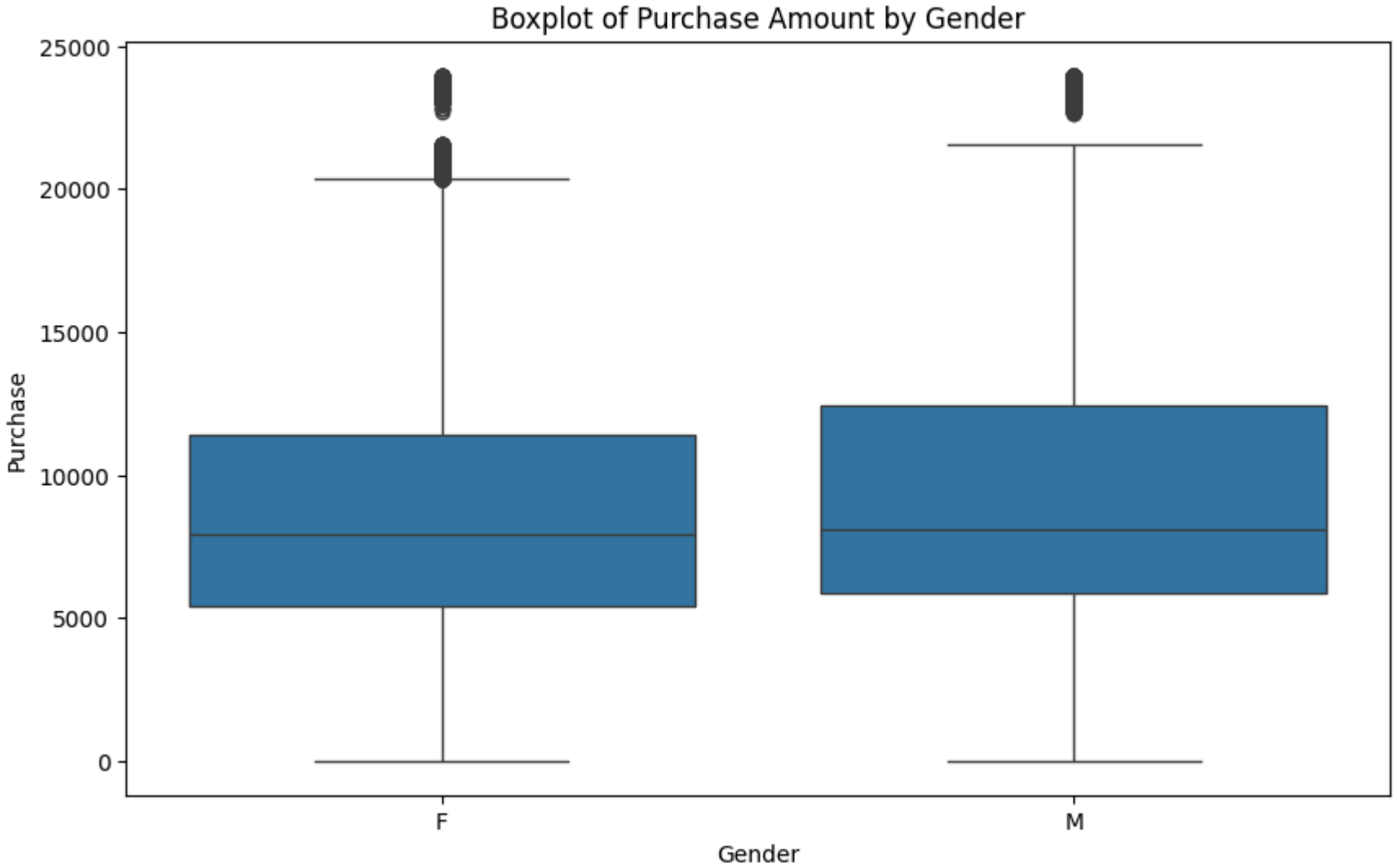
In [22]:
```
data.nunique()
```

Out[22]:
```
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
```

In [23]:
```
from scipy import stats
import warnings

# Ignore warnings
warnings.filterwarnings('ignore')
```

Detect Outliers using boxplot

In [24]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Gender', y='Purchase', data=data)
plt.title('Boxplot of Purchase Amount by Gender')
plt.show()
```



Data exploration

In [25]:
```
# Calculate total number of purchases
total_purchases = len(data)
print(f'Total Number of Purchases: {total_purchases}')
```

```
Total Number of Purchases: 550068
```

In [28]:
```python
# Calculate the total number of unique User_IDs
unique_user_ids = data['User_ID'].nunique()
print(f'Unique users count: {unique_user_ids}')
```

```
Unique users count: 5891
```

In [29]:
```python
# Calculate average number of purchases per user
average_purchases_per_user = data.groupby('User_ID').size().mean()
print(f'Average Number of Purchases per User: {average_purchases_per_user}')
```

```
Average Number of Purchases per User: 93.37429977932439
```

In [30]:
```python
# Calculate total and average number of purchases for female customers
total_purchases_female = len(data[data['Gender'] == 'F'])
average_purchases_per_female_user = data[data['Gender'] == 'F'].groupby('User_ID').size().mean()
print(f'Total Number of Purchases for Female Customers: {total_purchases_female}')
print(f'Average Number of Purchases per Female User: {average_purchases_per_female_user}')
```

```
Total Number of Purchases for Female Customers: 135809
Average Number of Purchases per Female User: 81.51800720288115
```

In [31]:
```python
# Calculate total and average number of purchases for male customers
total_purchases_male = len(data[data['Gender'] == 'M'])
average_purchases_per_male_user = data[data['Gender'] == 'M'].groupby('User_ID').size().mean()
print(f'Total Number of Purchases for Male Customers: {total_purchases_male}')
print(f'Average Number of Purchases per Male User: {average_purchases_per_male_user}')
```

```
Total Number of Purchases for Male Customers: 414259
Average Number of Purchases per Male User: 98.0494674556213
```

In [32]:
```python
# Calculate total purchase amount
total_purchase = data['Purchase'].sum()
print(f'Total Purchase Amount: {total_purchase}')
```

```
Total Purchase Amount: 5095812742
```

In [33]:
```python
# Calculate average purchase amount per user
average_purchase_per_user = data.groupby('User_ID')['Purchase'].mean().mean()
print(f'Average Purchase Amount per User: {average_purchase_per_user}')
```

```
Average Purchase Amount per User: 9568.83991355893
```

In [34]:
```
# Calculate total and average purchase amount for female customers
total_purchase_female = data[data['Gender'] == 'F']['Purchase'].sum()
average_purchase_per_female_user = data[data['Gender'] == 'F'].groupby('User_ID')['Purchase'].mean().mean()
print(f'Total Purchase Amount for Female Customers: {total_purchase_female}')
print(f'Average Purchase Amount per Female User: {average_purchase_per_female_user}')
```

```
Total Purchase Amount for Female Customers: 1186232642
Average Purchase Amount per Female User: 8965.19846393646
```

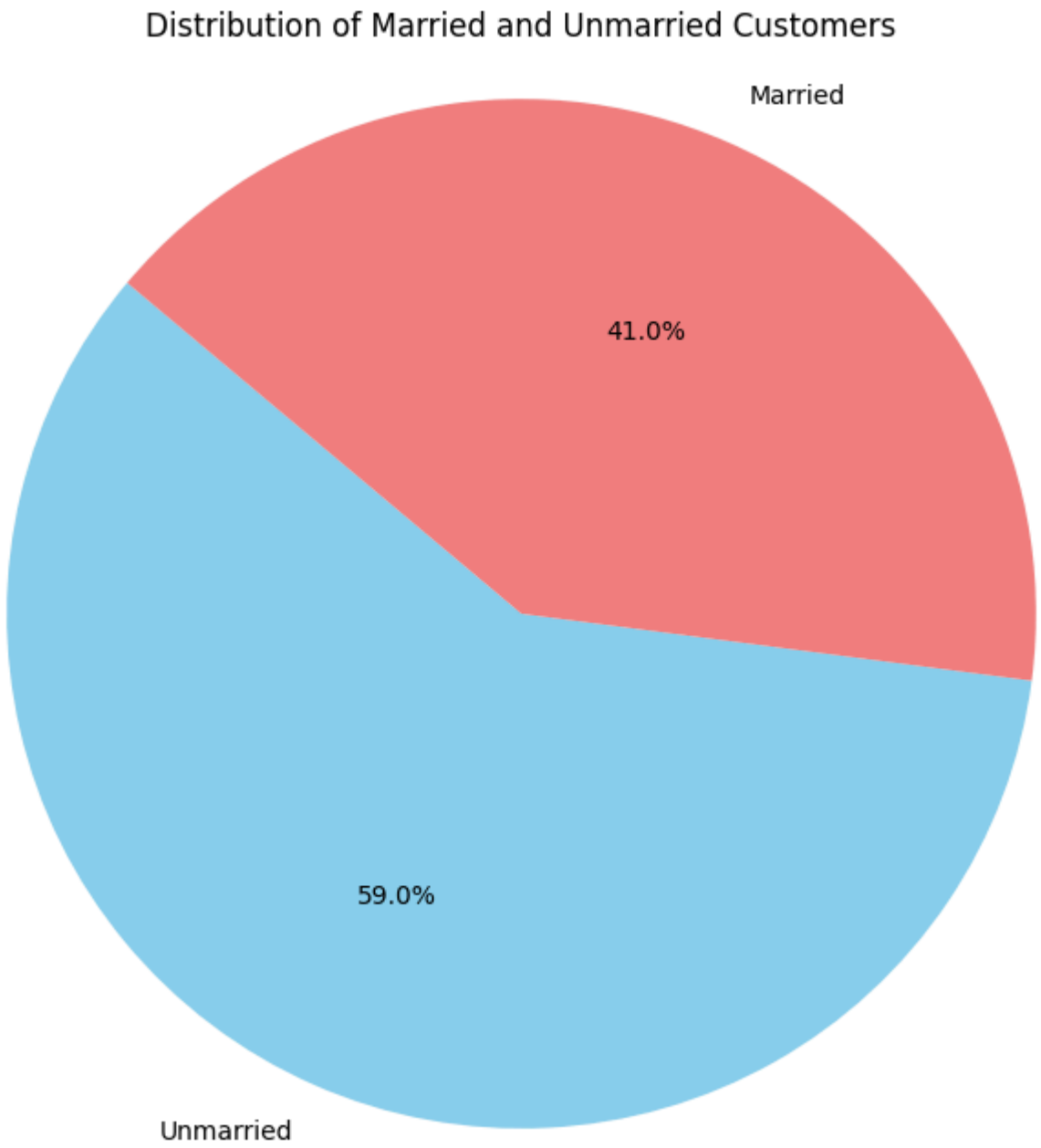
In [35]:
```
# Calculate total and average purchase amount for male customers
total_purchase_male = data[data['Gender'] == 'M']['Purchase'].sum()
average_purchase_per_male_user = data[data['Gender'] == 'M'].groupby('User_ID')['Purchase'].mean().mean()
print(f'Total Purchase Amount for Male Customers: {total_purchase_male}')
print(f'Average Purchase Amount per Male User: {average_purchase_per_male_user}')
```

```
Total Purchase Amount for Male Customers: 3909580100
Average Purchase Amount per Male User: 9806.867524226629
```

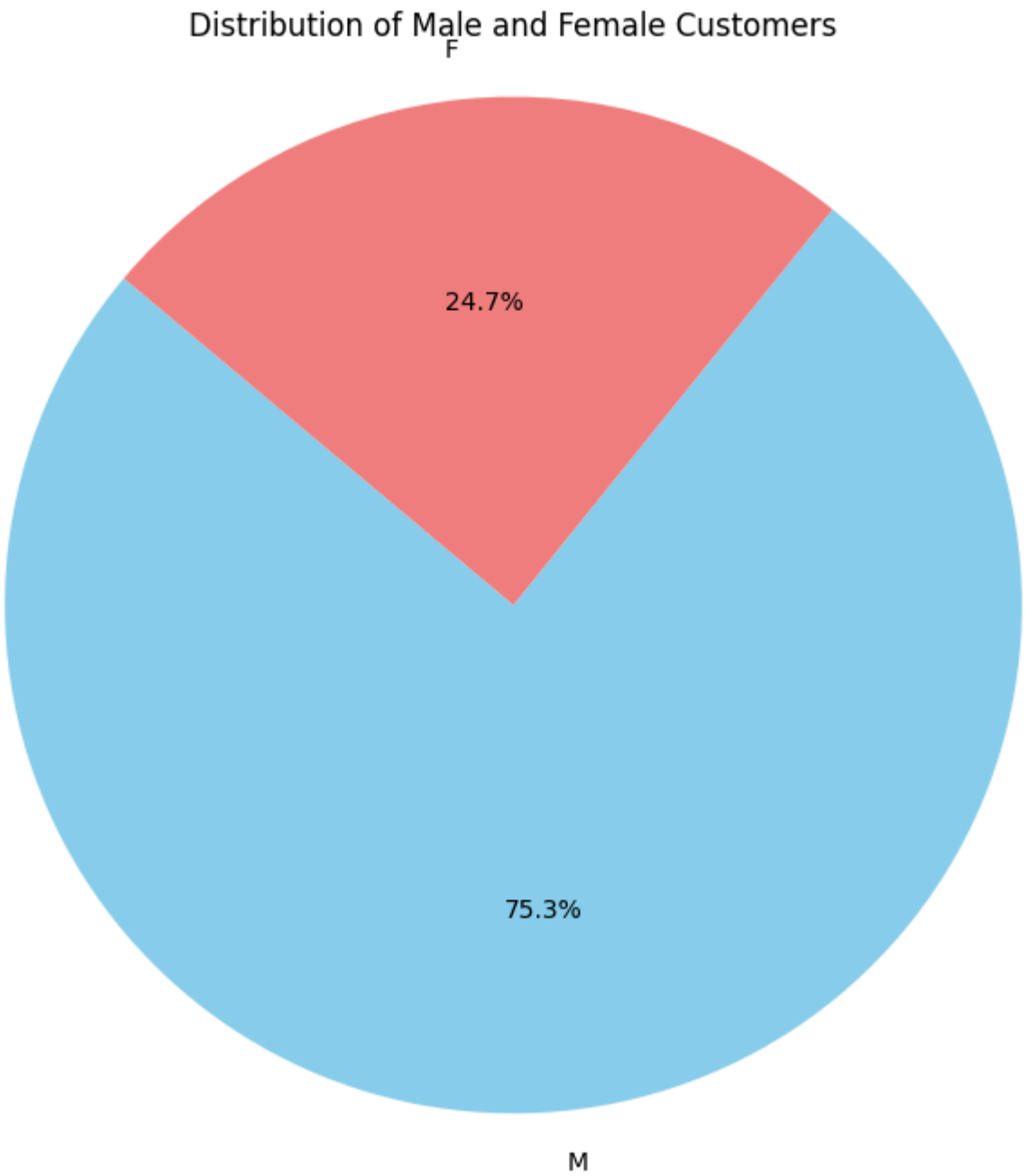
In [36]:
```
#replacing the values in marital_status column

data['Marital_Status'] = data['Marital_Status'].replace({0:'Unmarried',1:'Married'})
data['Marital_Status'].unique()
```

Out[36]:
```
array(['Unmarried', 'Married'], dtype=object)
```

In [37]:
```
# Calculate the number of male and female customers
marital_status_counts = data['Marital_Status'].value_counts()

# Plot the pie chart
plt.figure(figsize=(8, 8))
plt.pie(marital_status_counts, labels=marital_status_counts.index, autopct='%1.1f%%', startangle=140, colors=['skyblue', 'ligh
plt.title('Distribution of Married and Unmarried Customers')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```


Distribution of Married and Unmarried Customers
Married
41.0%
59.0%
Unmarried

In [38]:
```
# Calculate the number of married and unmarried customers
gender_counts = data['Gender'].value_counts()

# Plot the pie chart
plt.figure(figsize=(8, 8))
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=140, colors=['skyblue', 'lightcoral'])
plt.title('Distribution of Male and Female Customers')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```


Distribution of Male and Female Customers
F
24.7%
75.3%
M

In [39]:
```
# Calculate the average purchase amount for male and female customers
average_purchase_female = data[data['Gender'] == 'F']['Purchase'].mean()
average_purchase_male = data[data['Gender'] == 'M']['Purchase'].mean()

print(f'Average Purchase Amount for Female: {average_purchase_female}')
print(f'Average Purchase Amount for Male: {average_purchase_male}')
```

```
Average Purchase Amount for Female: 8734.565765155476
Average Purchase Amount for Male: 9437.526040472265
```

In [40]:
```
data.head()
```

Out[40]:

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

In [41]:
```
data['User_ID'].value_counts()
```

Out[41]:
```
User_ID
1001680    1026
1004277     979
1001941     898
1001181     862
1000889     823
           ...
1002690       7
1002111       7
1005810       7
1004991       7
1000708       6
Name: count, Length: 5891, dtype: int64
```

In [42]:
```
data['Product_ID'].value_counts()
```

Out[42]:
```
Product_ID
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
             ...
P00314842       1
P00298842       1
P00231642       1
P00204442       1
P00066342       1
Name: count, Length: 3631, dtype: int64
```

In [43]:
```
data['Gender'].value_counts()
```

Out[43]:
```
Gender
M    414259
F    135809
Name: count, dtype: int64
```

In [44]:
```
data['Age'].value_counts()
```

Out[44]:
```
Age
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: count, dtype: int64
```

In [45]:
```
data['Occupation'].value_counts()
```

Out[45]:
```
Occupation
4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
2     26588
16    25371
6     20355
3     17650
10    12930
5     12177
15    12165
11    11586
19     8461
13     7728
18     6622
9      6291
8      1546
Name: count, dtype: int64
```

In [46]:
```
data['City_Category'].value_counts()
```

Out[46]:
```
City_Category
B    231173
C    171175
A    147720
Name: count, dtype: int64
```

In [47]:
```
data['Stay_In_Current_City_Years'].value_counts()
```

Out[47]:
```
Stay_In_Current_City_Years
1     193821
2     101838
3      95285
4+     84726
0      74398
Name: count, dtype: int64
```

In [48]:
```
data['Marital_Status'].value_counts()
```

Out[48]:
```
Marital_Status
Unmarried    324731
Married      225337
Name: count, dtype: int64
```

In [49]:
```
data['Product_Category'].value_counts()
```

Out[49]:
```
Product_Category
5     150933
1     140378
8     113925
11     24287
2      23864
6      20466
3      20213
4      11753
16      9828
15      6290
13      5549
10      5125
12      3947
7       3721
18      3125
20      2550
19      1603
14      1523
17       578
9        410
Name: count, dtype: int64
```

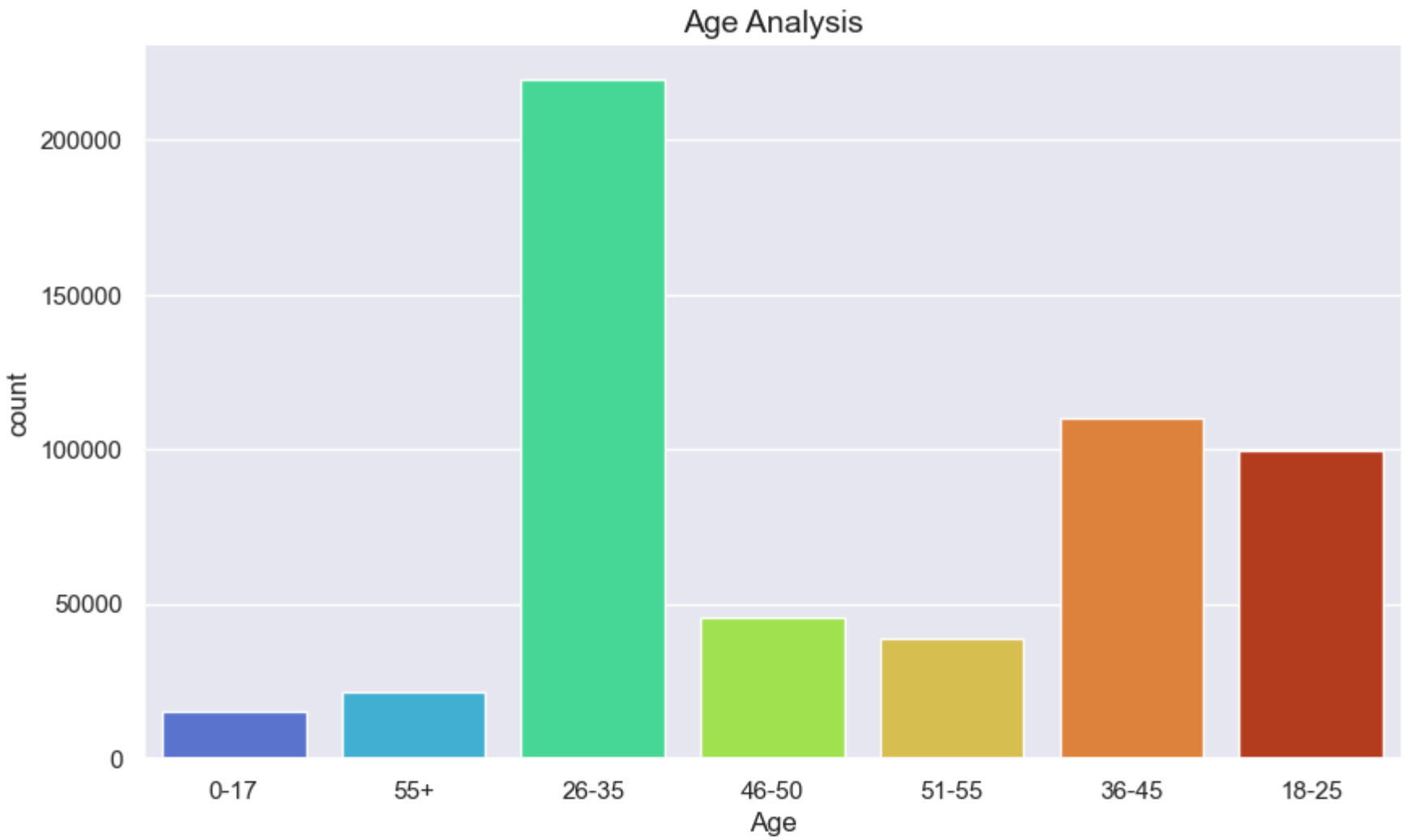
In [50]:
```
data['Purchase'].value_counts()
```

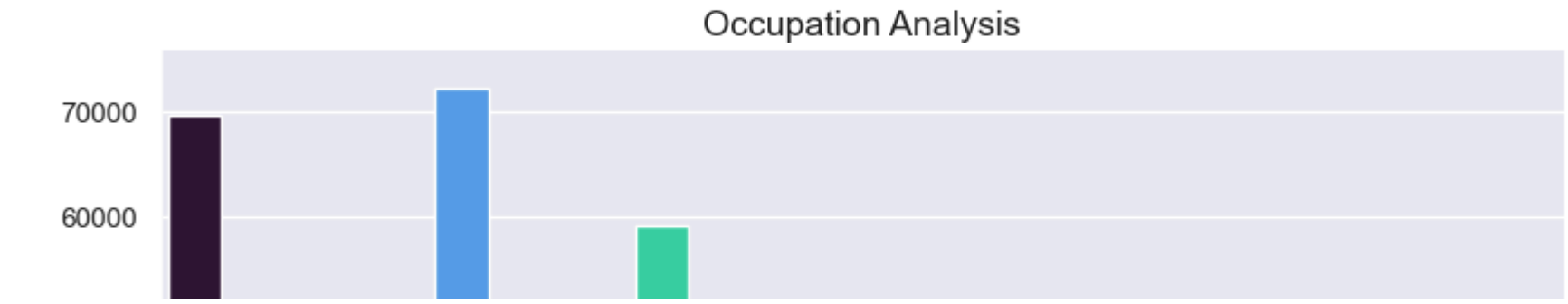
Out[50]:
```
Purchase
7011     191
7193     188
6855     187
6891     184
7012     183
        ...
23491      1
18345      1
3372       1
855        1
21489      1
Name: count, Length: 18105, dtype: int64
```

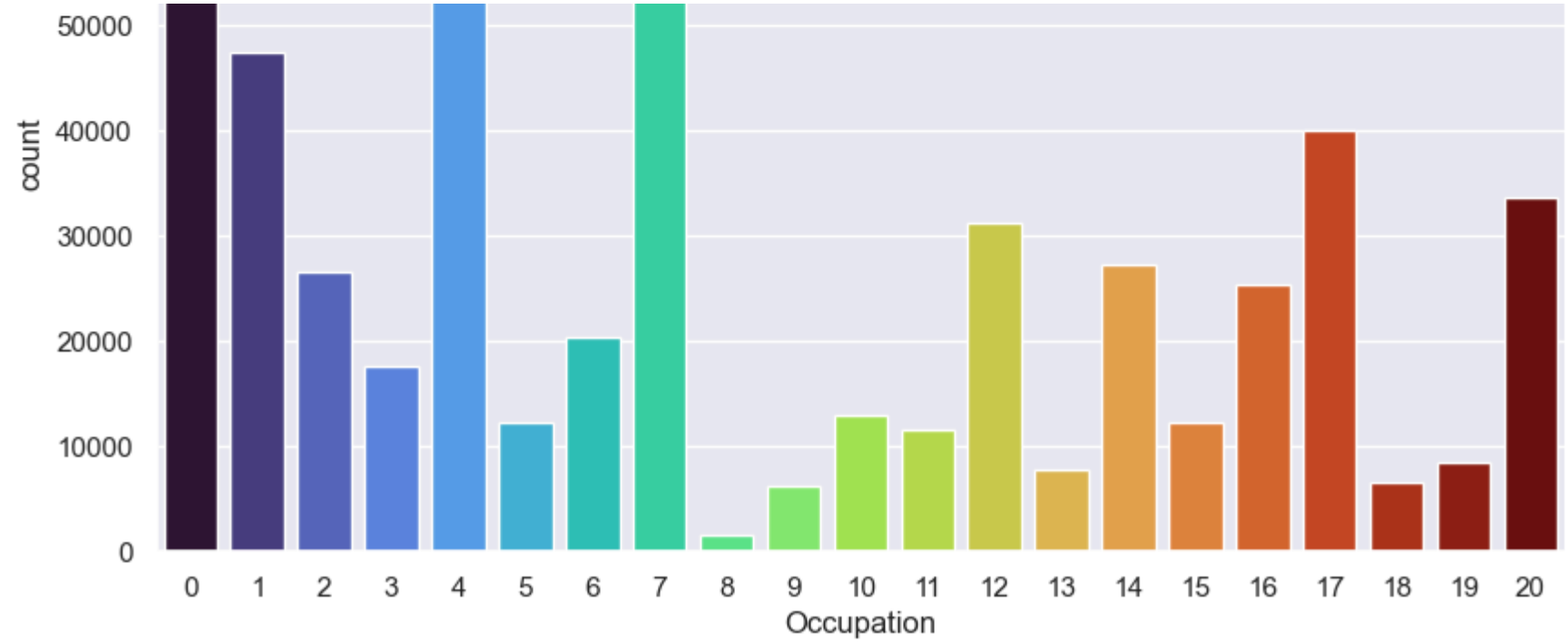
Univariate Analysis

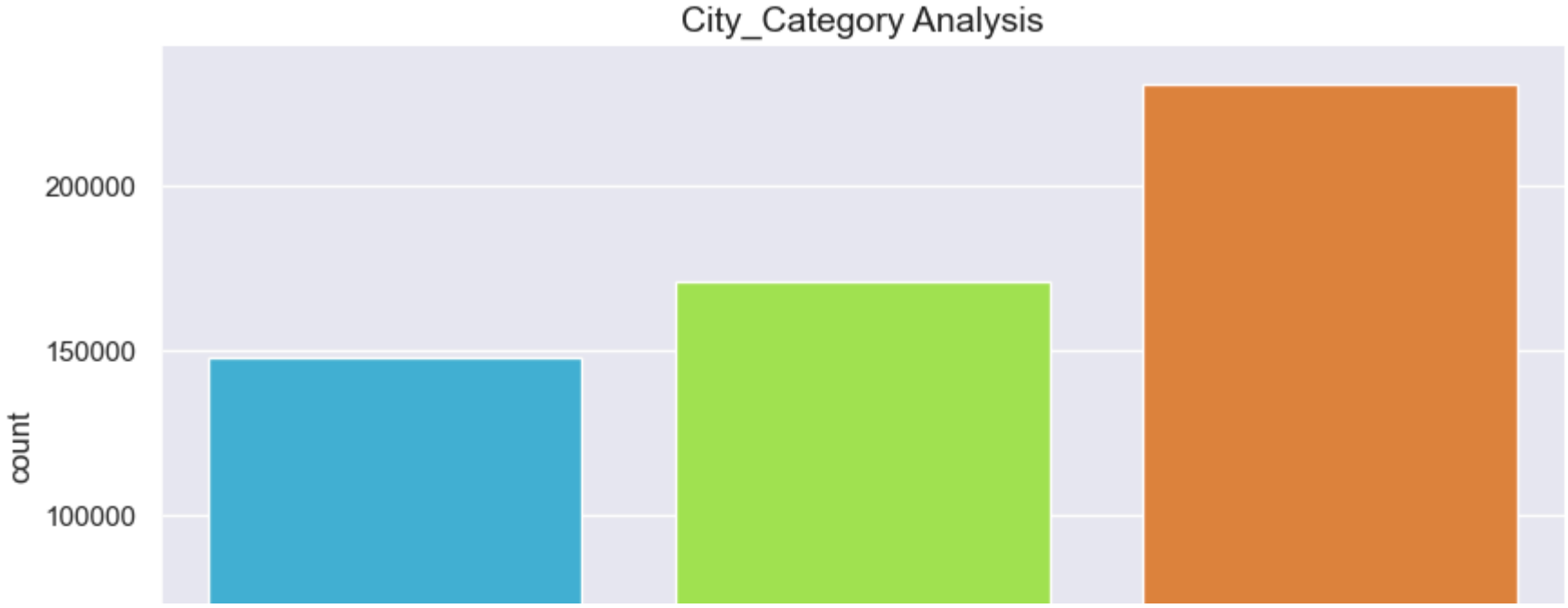
In [51]:
```
category = ['Age', 'Occupation', 'City_Category', 'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category']
```

In [52]:
```
plt.figure(figsize=(10,40))
sns.set(style='darkgrid')

# Plot each categorical column
for i, col in enumerate(category, 1):
    plt.subplot(6, 1, i)
    sns.countplot(data=data, x=col, hue=col, palette='turbo', legend=False)
    sns.despine()
    plt.title(f'{col} Analysis', fontsize=14, fontfamily='sans-serif')

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

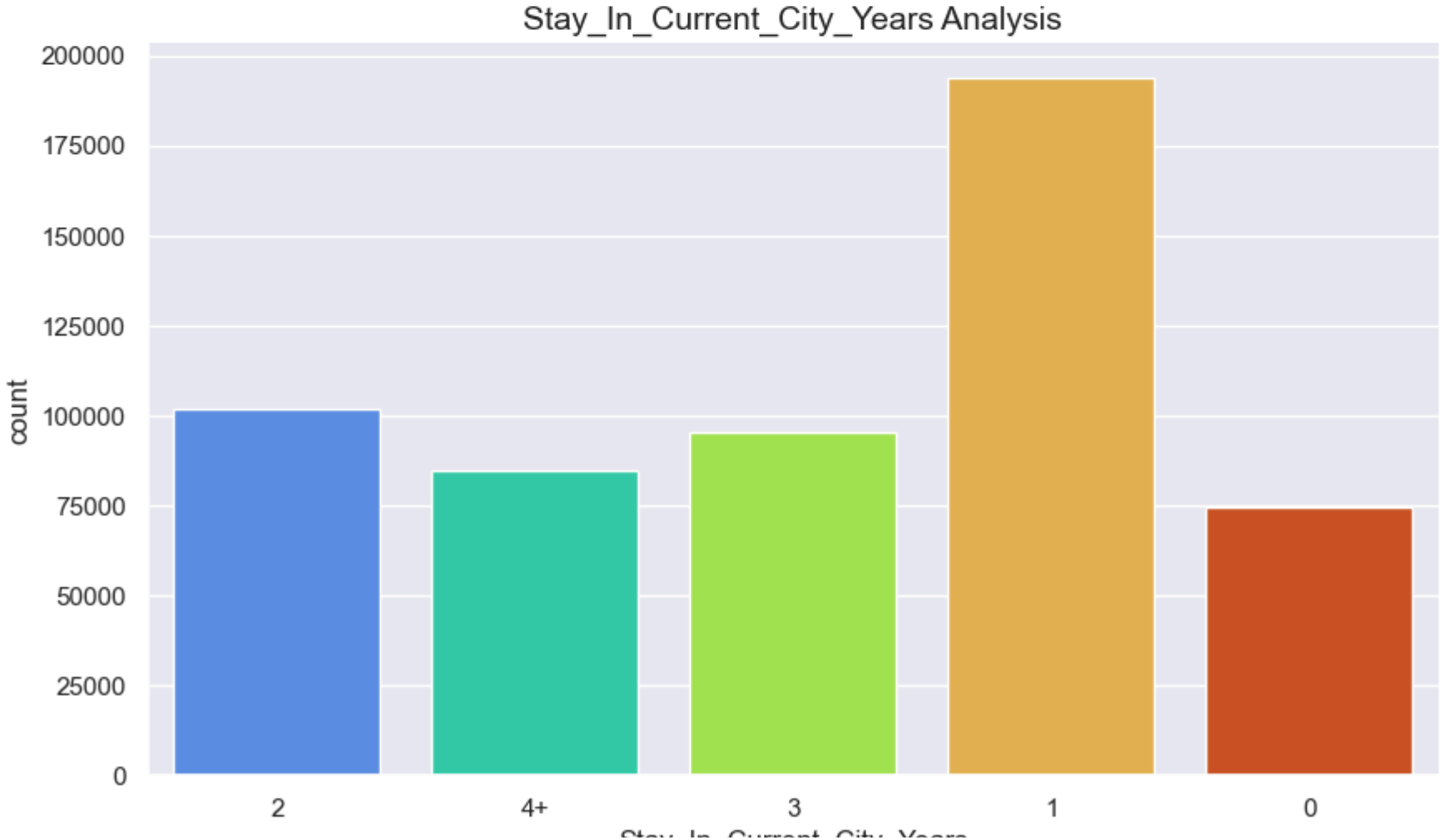

Age Analysis
200000
150000
100000
50000
0
count
0-17
55+
26-35
46-50
51-55
36-45
18-25
Age


Occupation Analysis
70000
60000

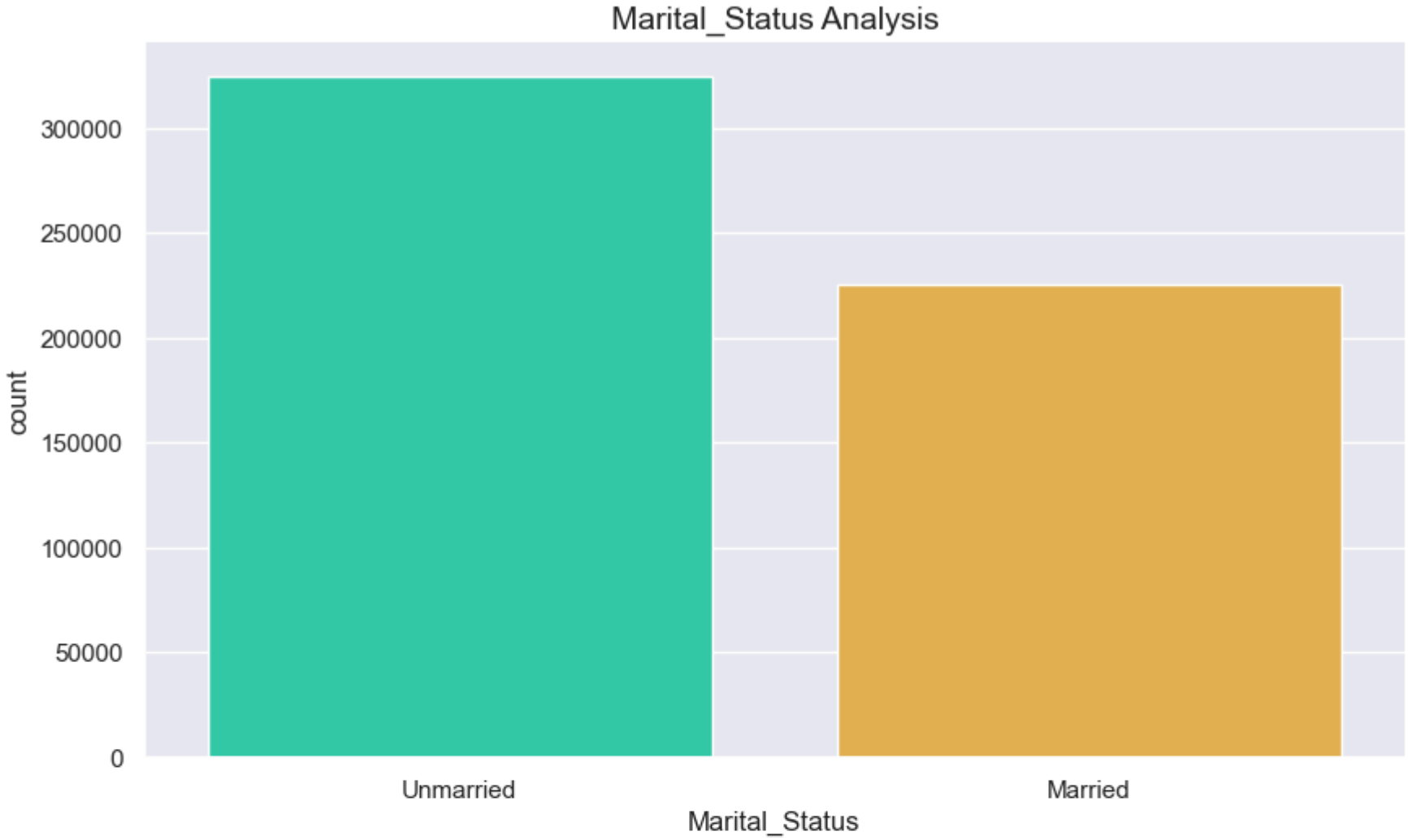

50000
40000
30000
20000
10000
0
count
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Occupation

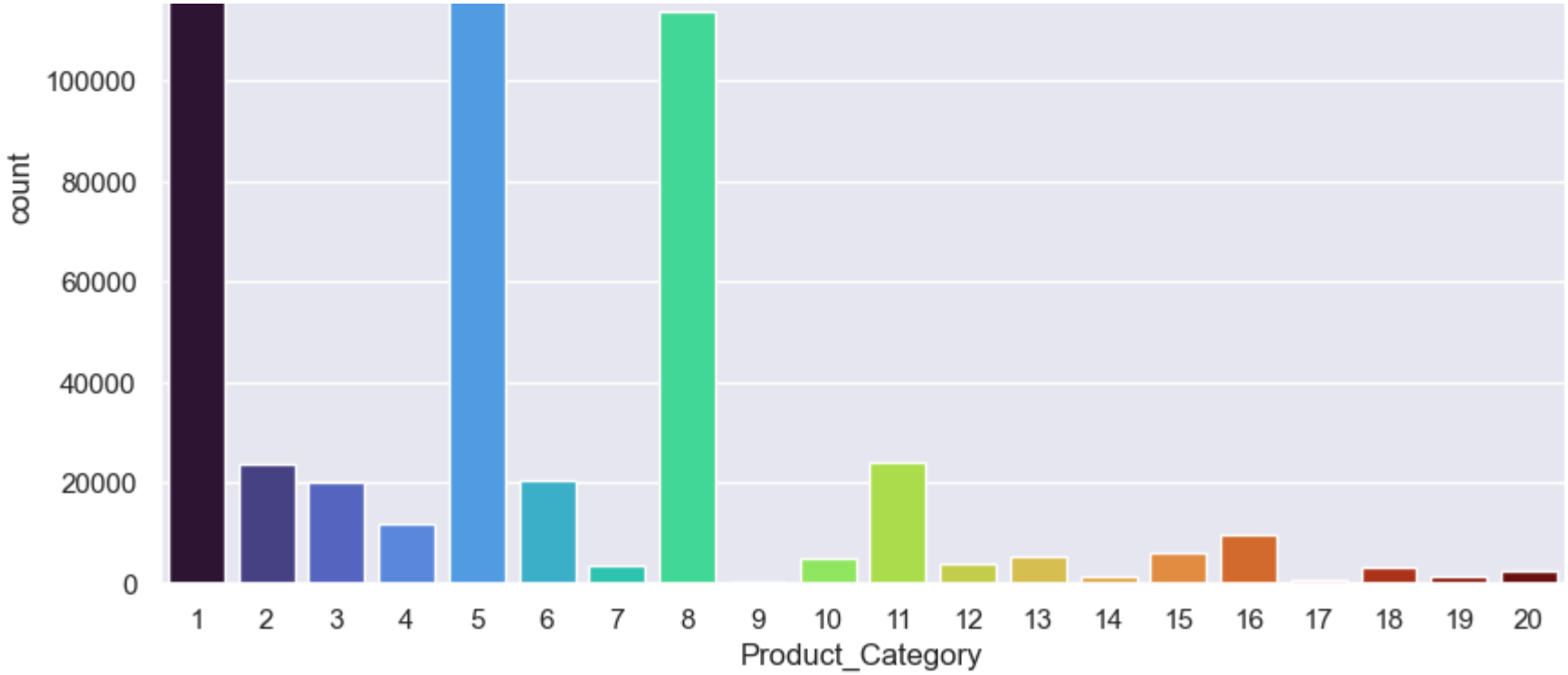

City_Category Analysis
200000
150000
100000
count


50000
0
A
C
B
City_Category


Stay_In_Current_City_Years Analysis
200000
175000
150000
125000
100000
75000
50000
25000
0
count
2
4+
3
1
0

Stay_In_Current_City_Years

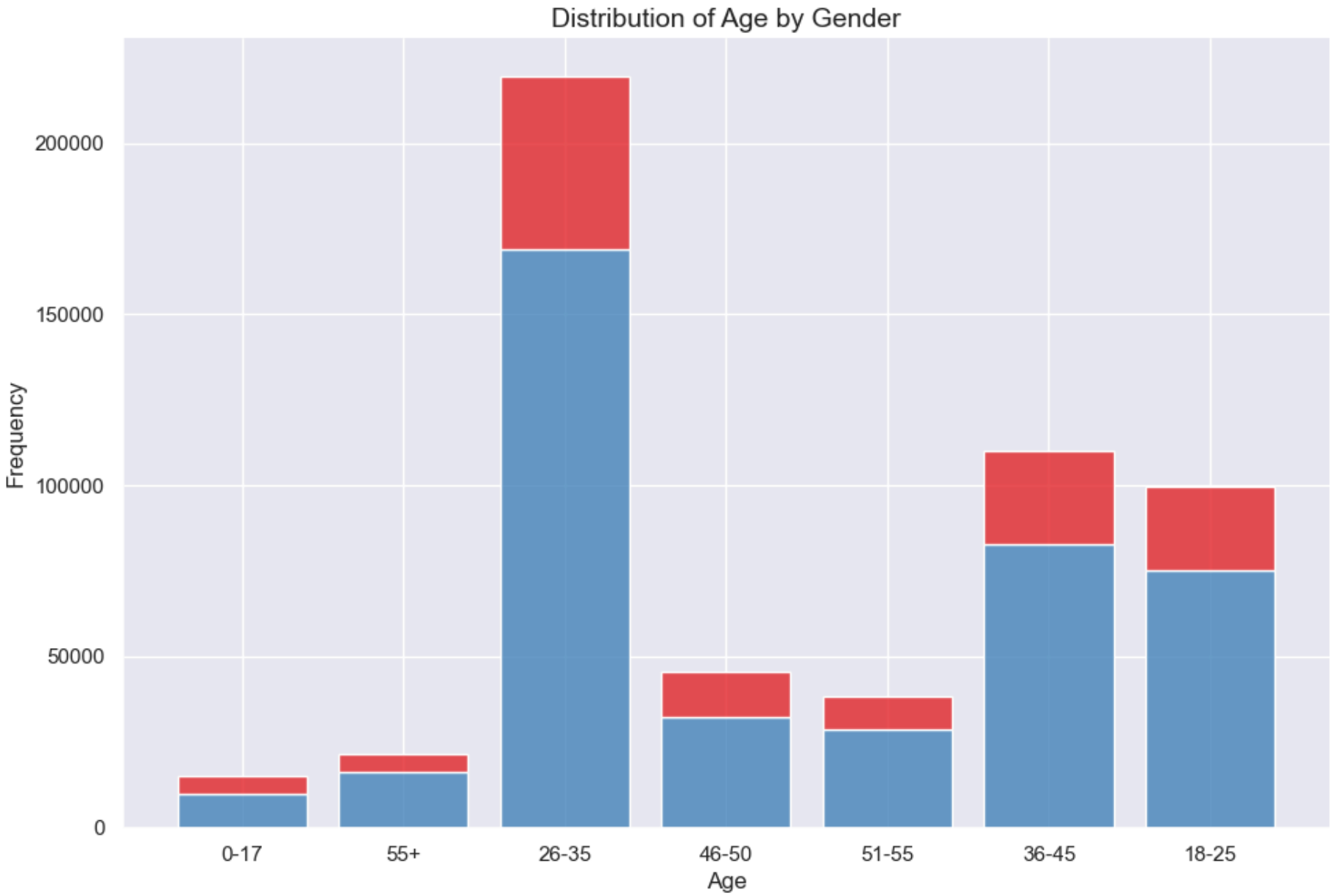


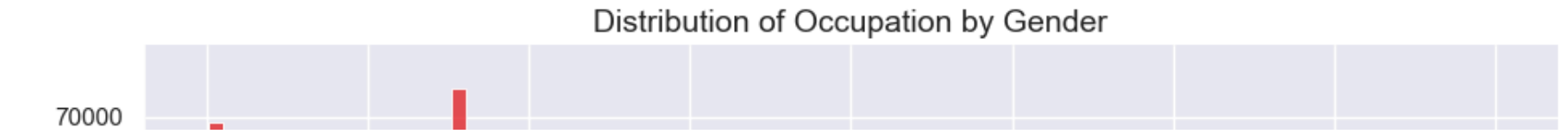


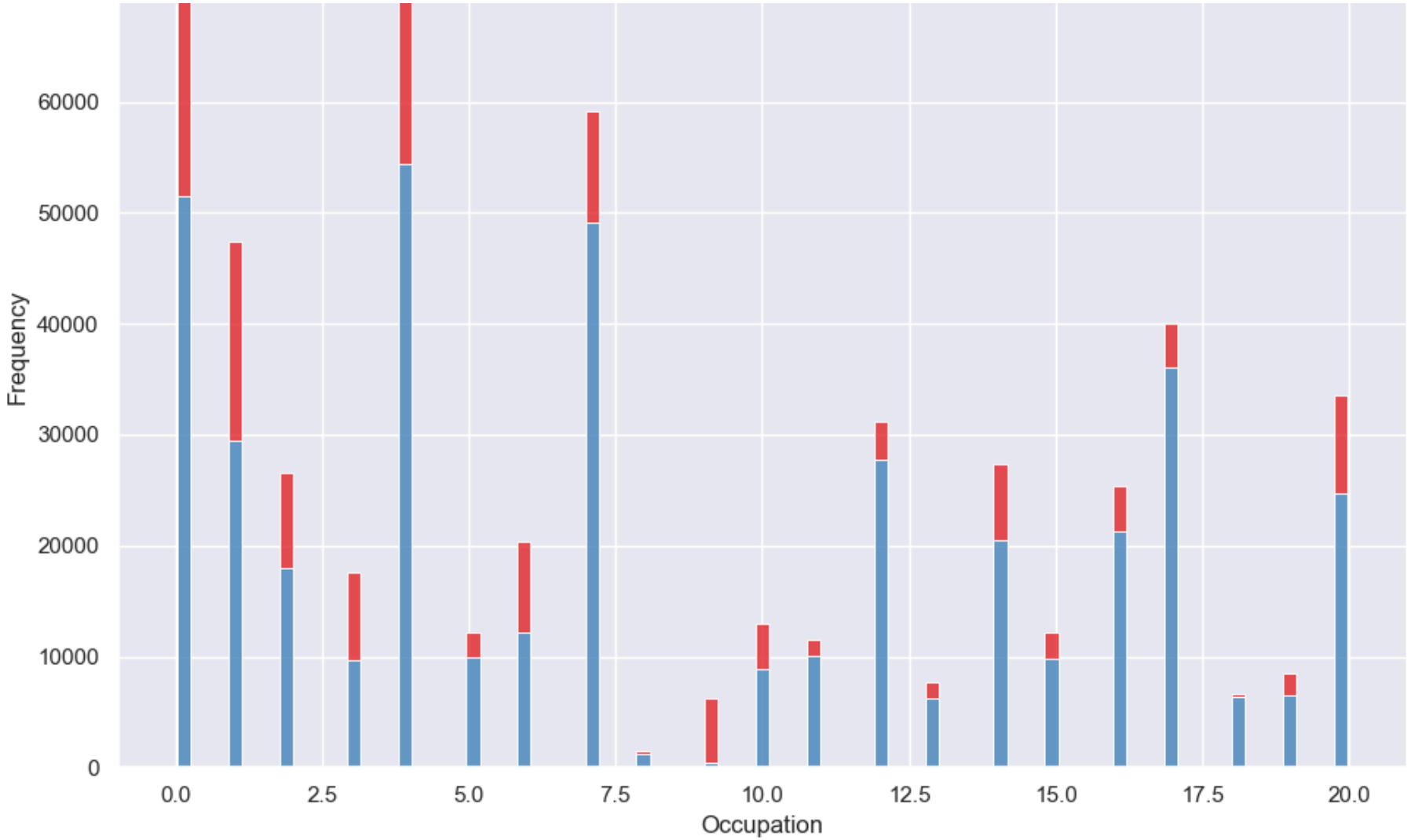
Bivariate AnalysisMultivariate Analysis

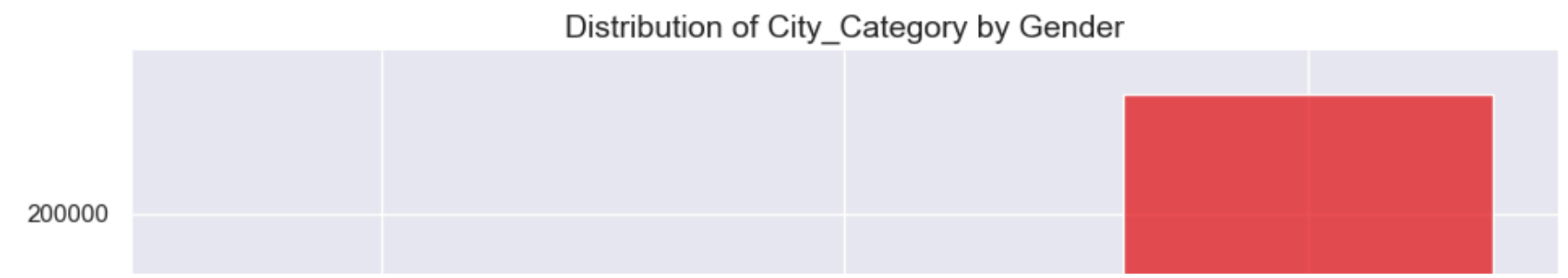
In [53]:
```
plt.figure(figsize=(10, 40))
sns.set(style='darkgrid')

# Plot each categorical column
for i, col in enumerate(category, 1):
    plt.subplot(6, 1, i)
    sns.histplot(data=data, x=col, hue='Gender', palette='Set1', legend=False, multiple='stack', shrink=0.8)
    sns.despine()

    # Set labels and title
    plt.xlabel(f'{col}', fontsize=12)
    plt.ylabel('Frequency', fontsize=12)
    plt.title(f'Distribution of {col} by Gender', fontsize=14, fontfamily='sans-serif')
```

```
    plt.tight_layout()

plt.show()
```

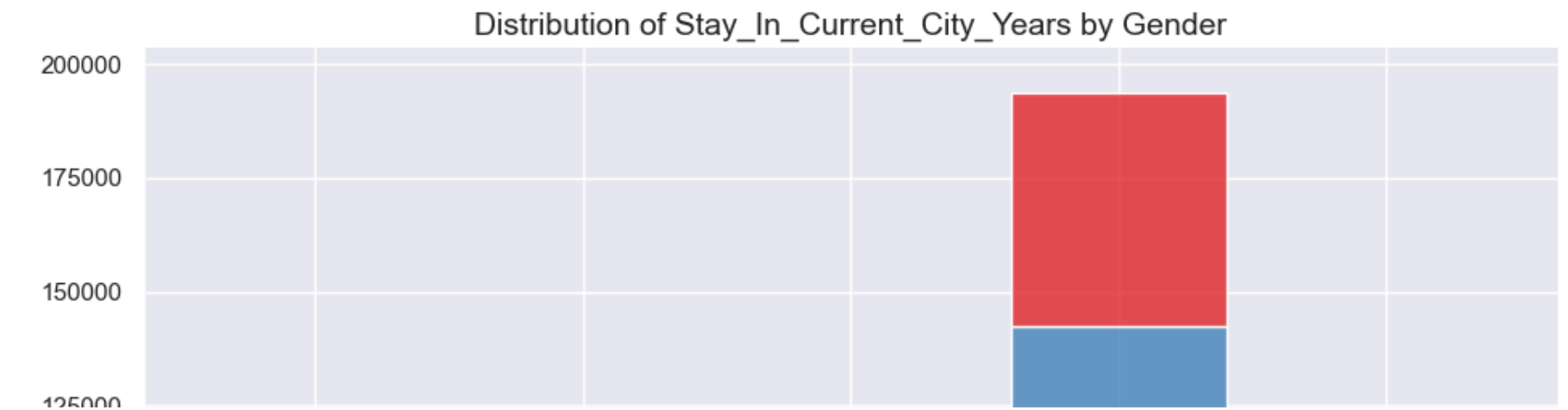

Distribution of Age by Gender
Frequency
200000
150000
100000
50000
0
0-17
55+
26-35
46-50
51-55
36-45
18-25
Age

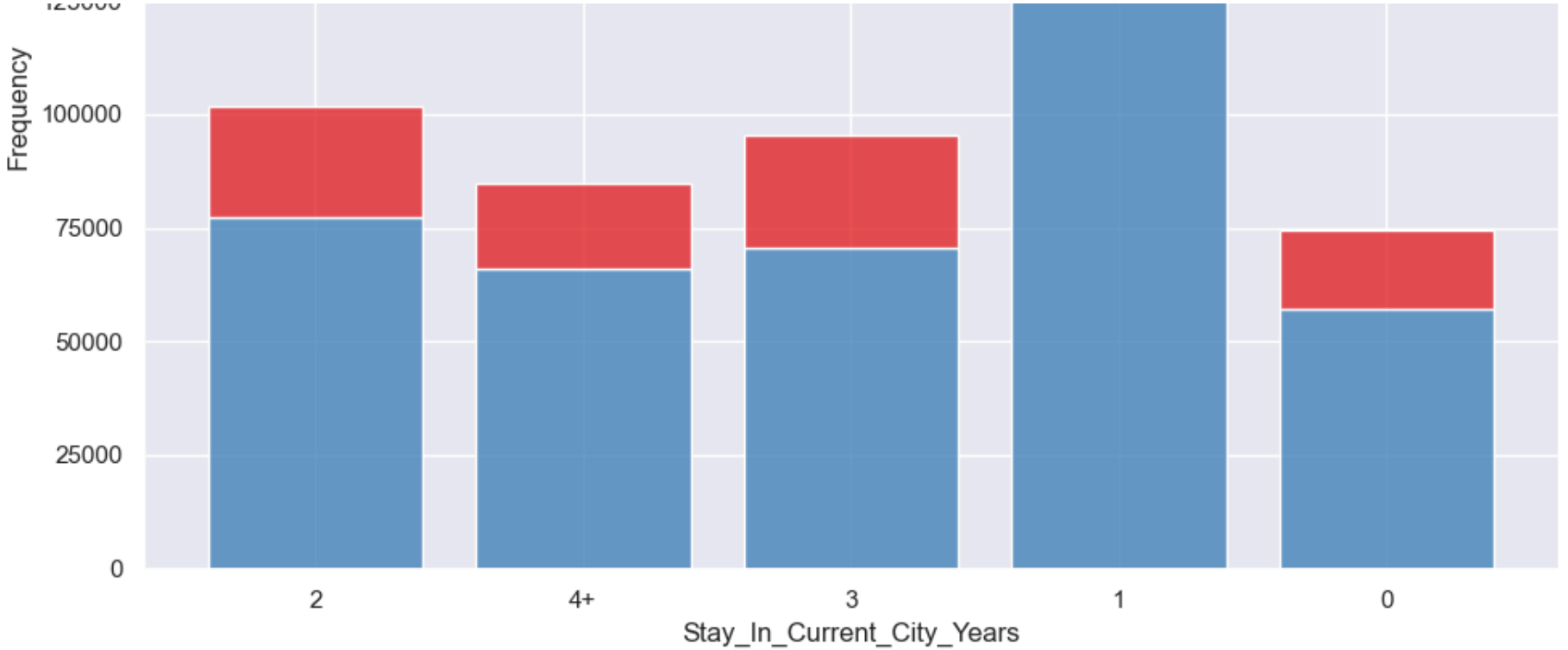

Distribution of Occupation by Gender
70000

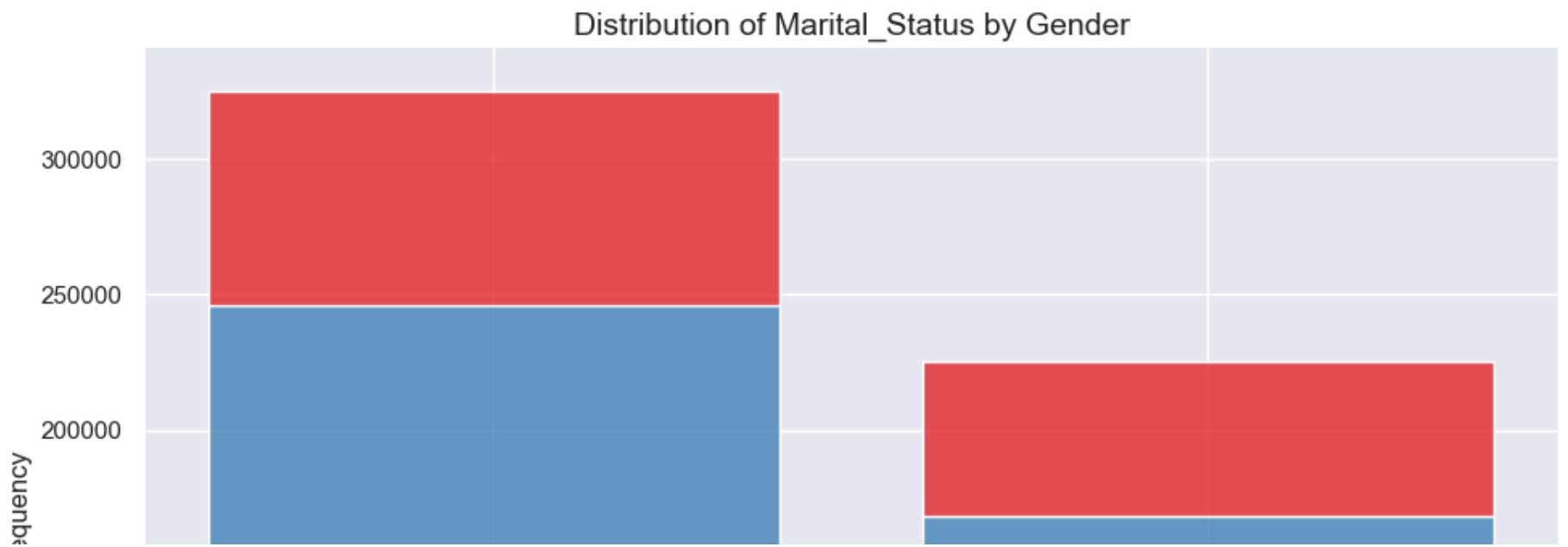

Frequency
60000
50000
40000
30000
20000
10000
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Occupation

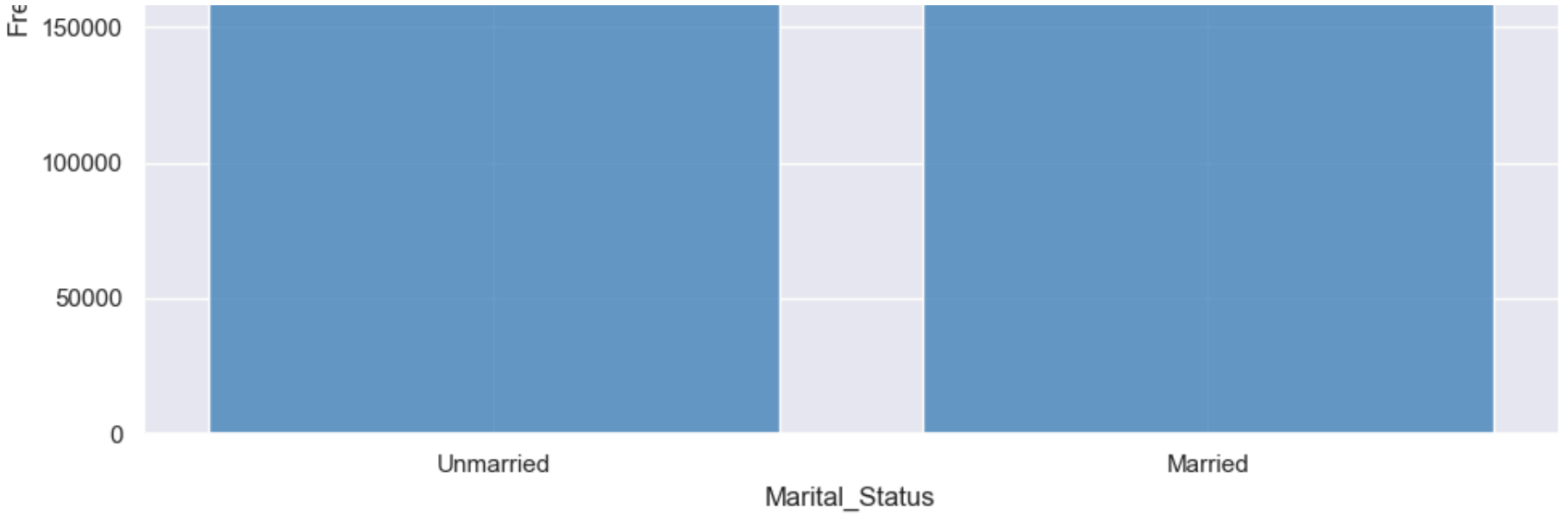

Distribution of City_Category by Gender
200000


150000
100000
50000
0
Frequency
A
C
B
City_Category

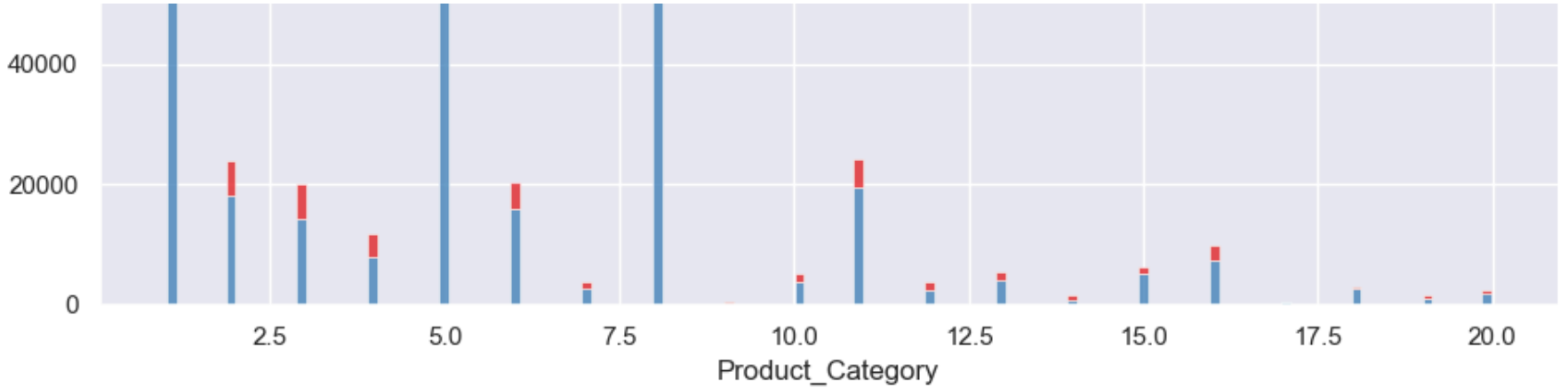

Distribution of Stay_In_Current_City_Years by Gender
200000
175000
150000
125000


Frequency
100000
75000
50000
25000
0
2
4+
3
1
0
Stay_In_Current_City_Years


Distribution of Marital_Status by Gender
300000
250000
200000


Fre
150000
100000
50000
0
Unmarried
Married
Marital_Status

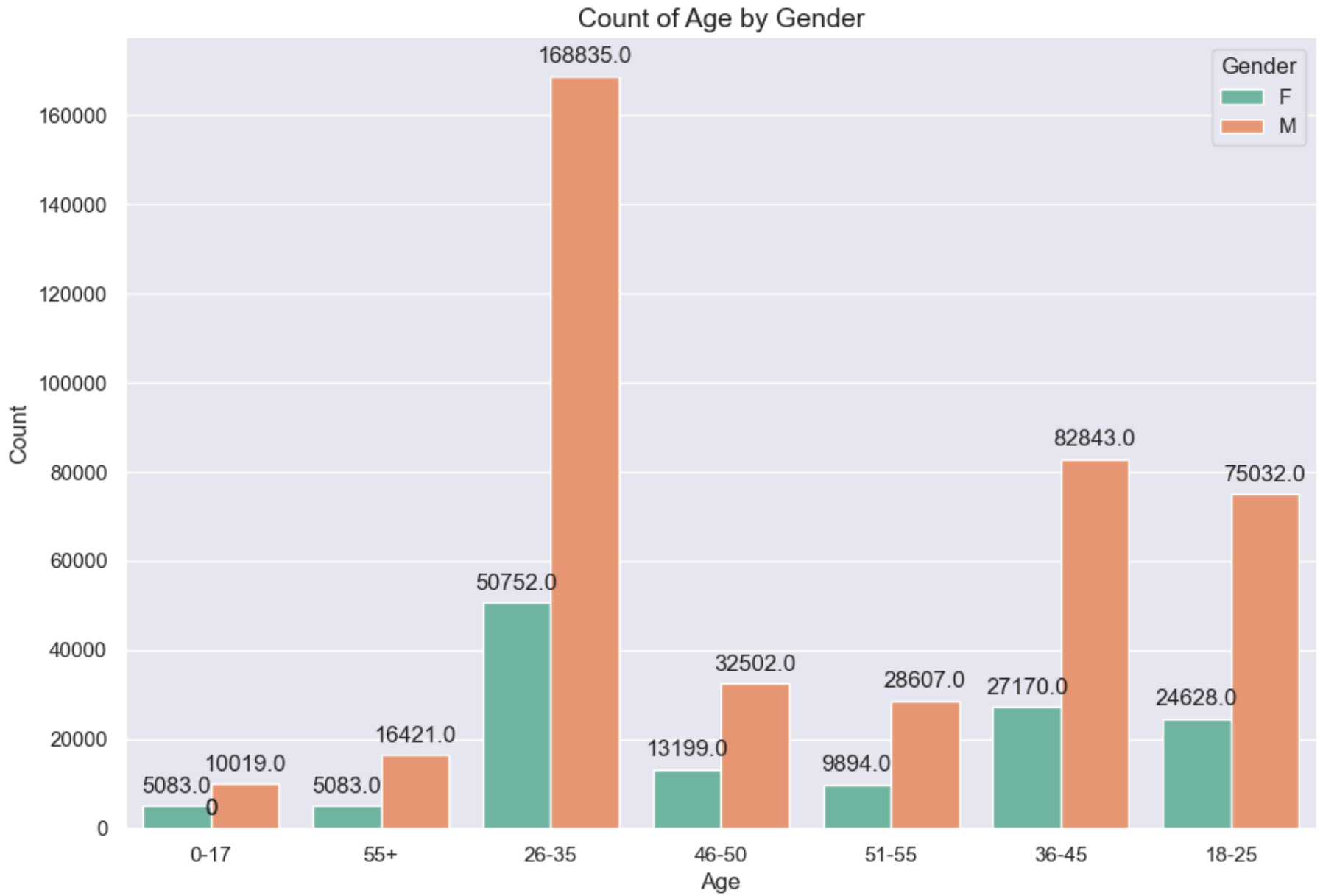

Distribution of Product_Category by Gender
140000
120000
100000
80000
60000
Frequency

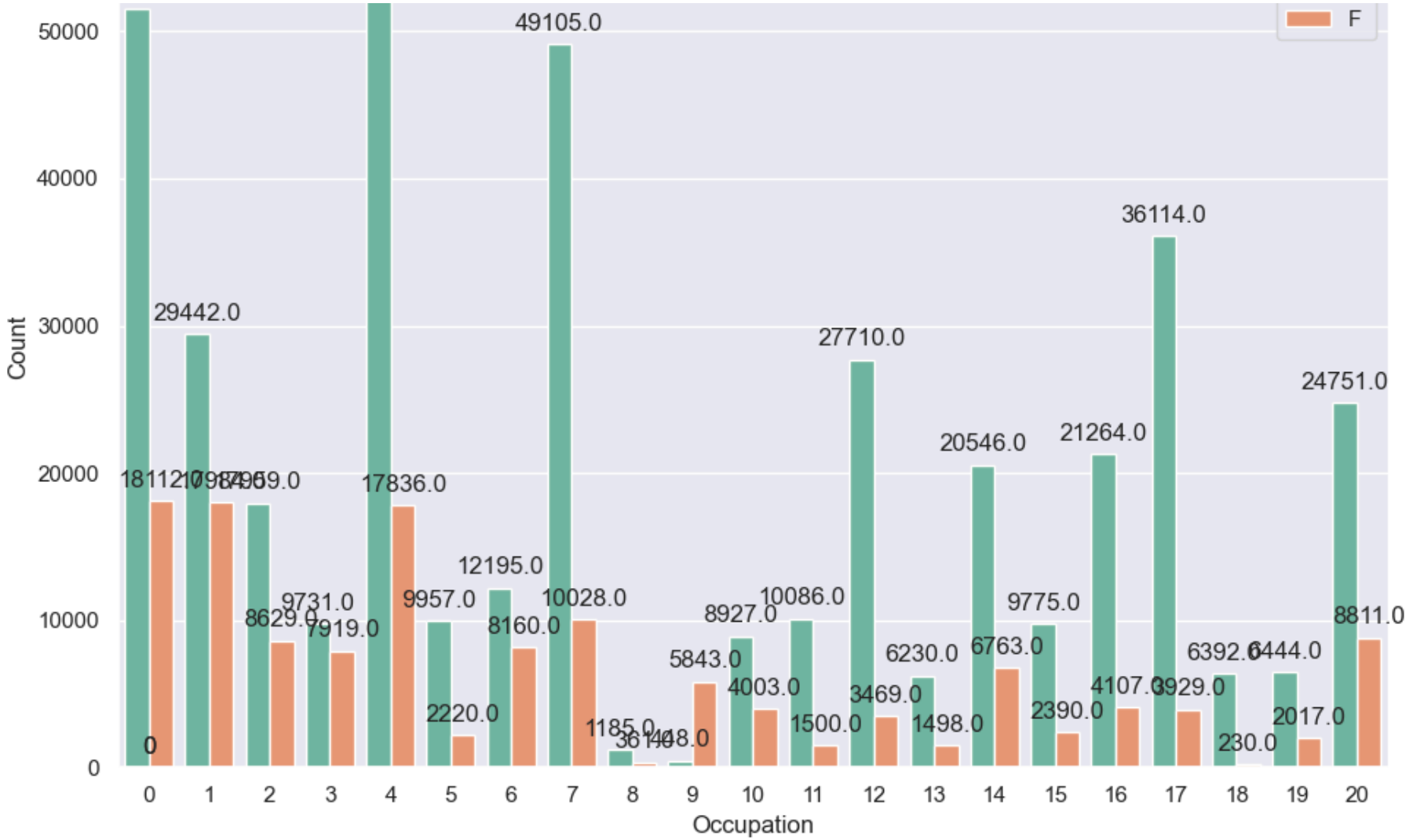


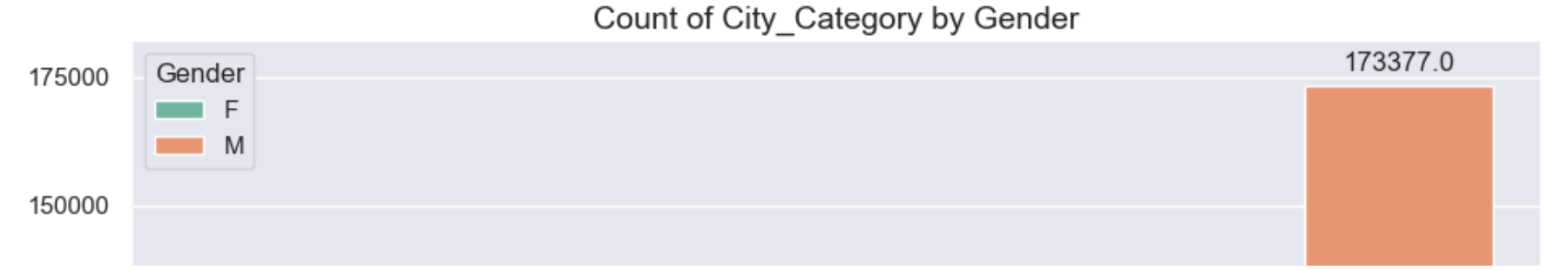
Multivariate Analysis

In [54]:
```
plt.figure(figsize=(10, 40))
sns.set(style='darkgrid')

# Plot each categorical column
for i, col in enumerate(category, 1):
    plt.subplot(6, 1, i)
    ax = sns.countplot(data=data, x=col, hue='Gender', palette='Set2')
    sns.despine()

    plt.title(f'Count of {col} by Gender', fontsize=14, fontfamily='sans-serif')
    plt.xlabel(col)
    plt.ylabel('Count')

    # Add bar counts as text labels
    for p in ax.patches:
        ax.annotate(f'{p.get_height()}',
                    (p.get_x() + p.get_width() / 2.,
                     p.get_height()),
                    ha='center',
                    va='center',
                    xytext=(0, 10),
                    textcoords='offset points')
```

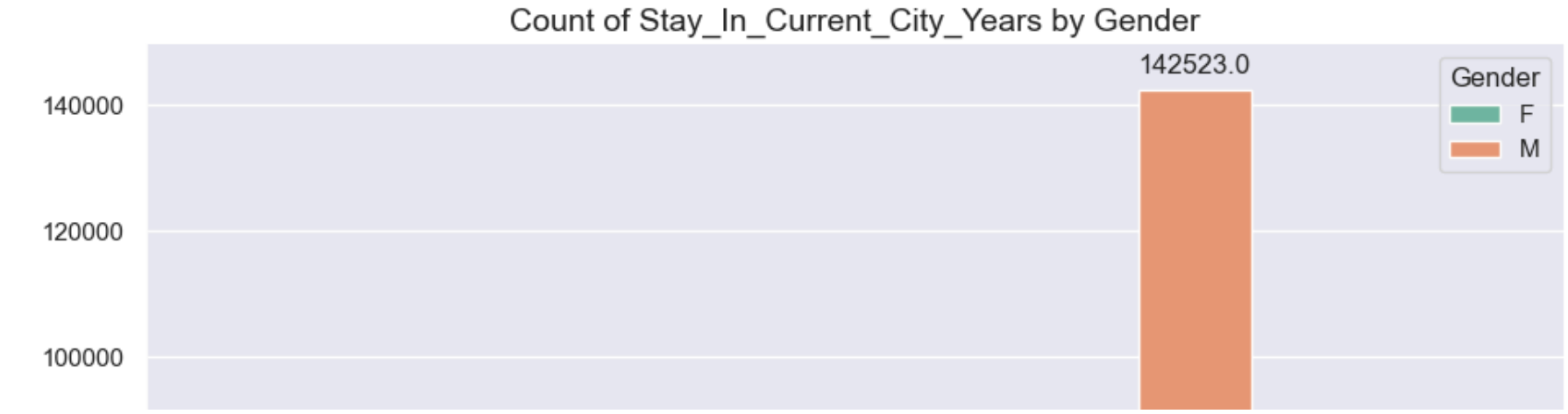
```
    plt.tight_layout()

plt.show()
```

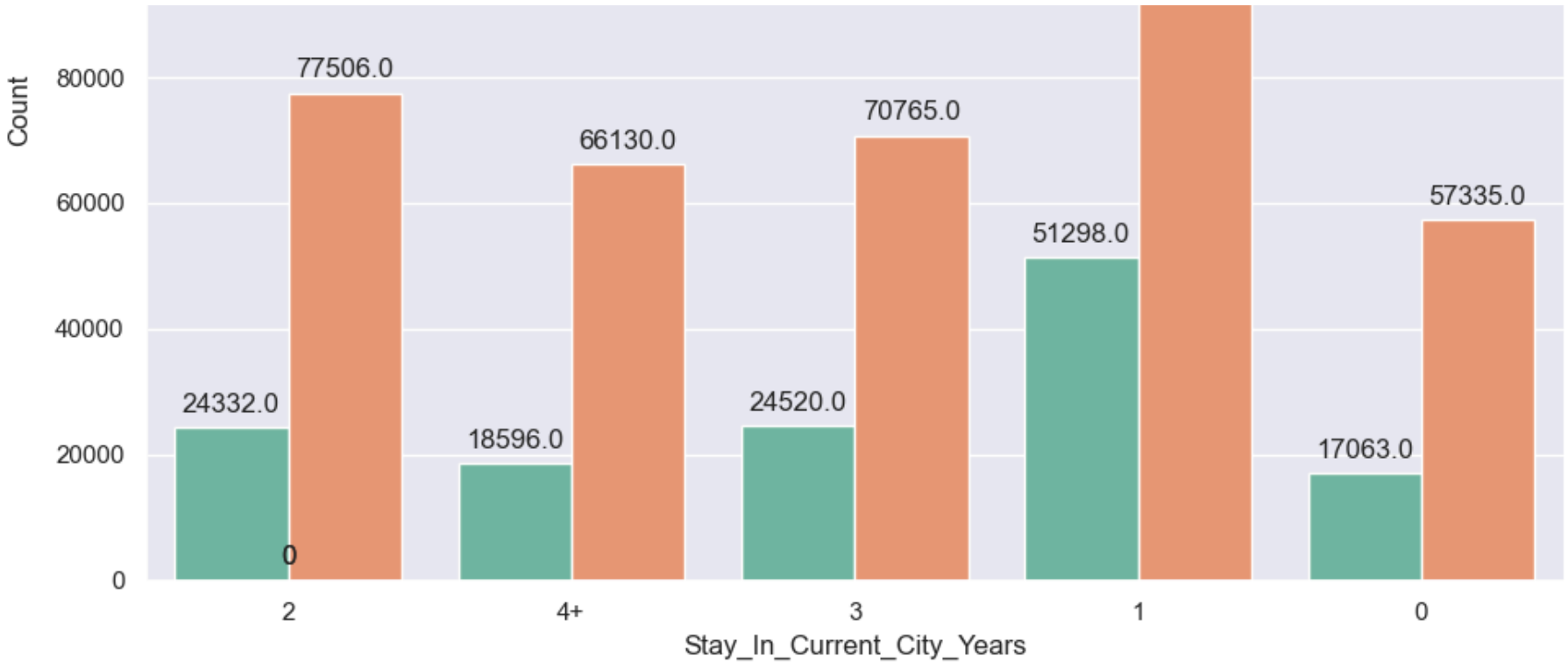

Count of Age by Gender
Gender
F
M
Count
Age
0
20000
40000
60000
80000
100000
120000
140000
160000
0-17
55+
26-35
46-50
51-55
36-45
18-25
5083.0
10019.0
5083.0
16421.0
50752.0
168835.0
13199.0
32502.0
9894.0
28607.0
27170.0
82843.0
24628.0
75032.0
Count of Occupation by Gender
Gender
M
51526.0
54472.0

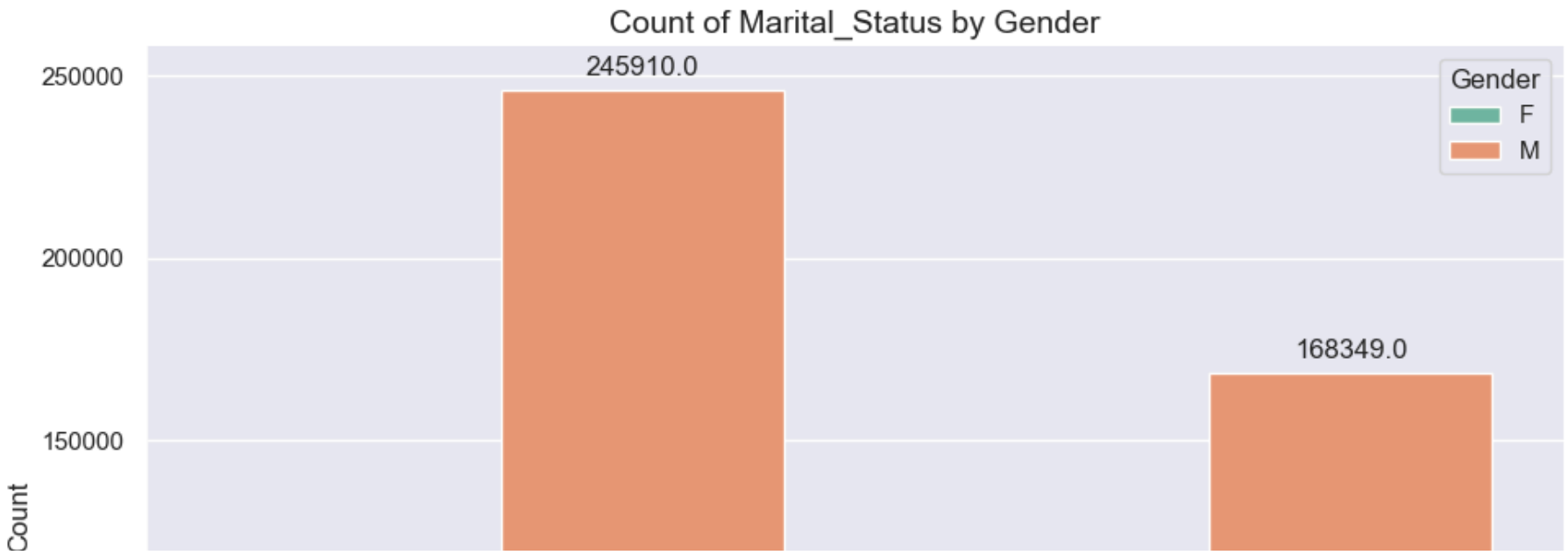

F
Count
Occupation
50000
40000
30000
20000
10000
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
49105.0
36114.0
29442.0
27710.0
24751.0
20546.0
21264.0
17836.0
12195.0
10086.0
9957.0
10028.0
9775.0
8927.0
8811.0
8629.0
9731.0
7919.0
6763.0
6230.0
6392.0
6444.0
5843.0
4003.0
4107.0
3929.0
3469.0
2390.0
2220.0
2017.0
1500.0
1498.0
1185.0
230.0
0

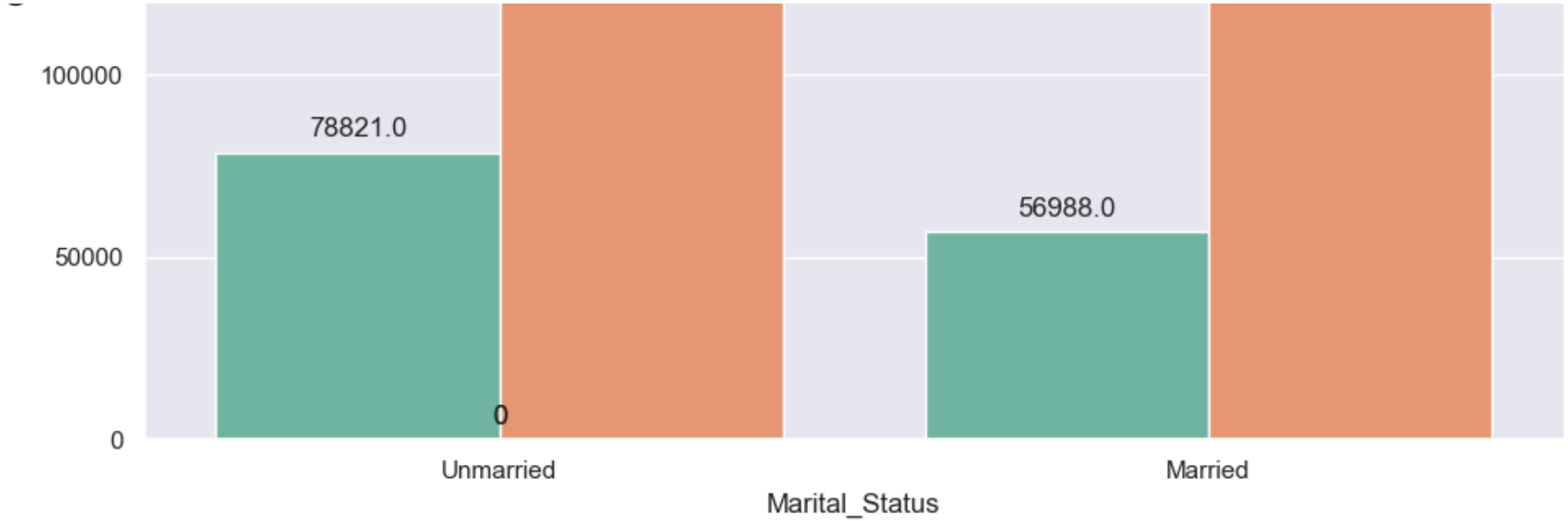

Count of City_Category by Gender
Gender
F
M
175000
150000
173377.0

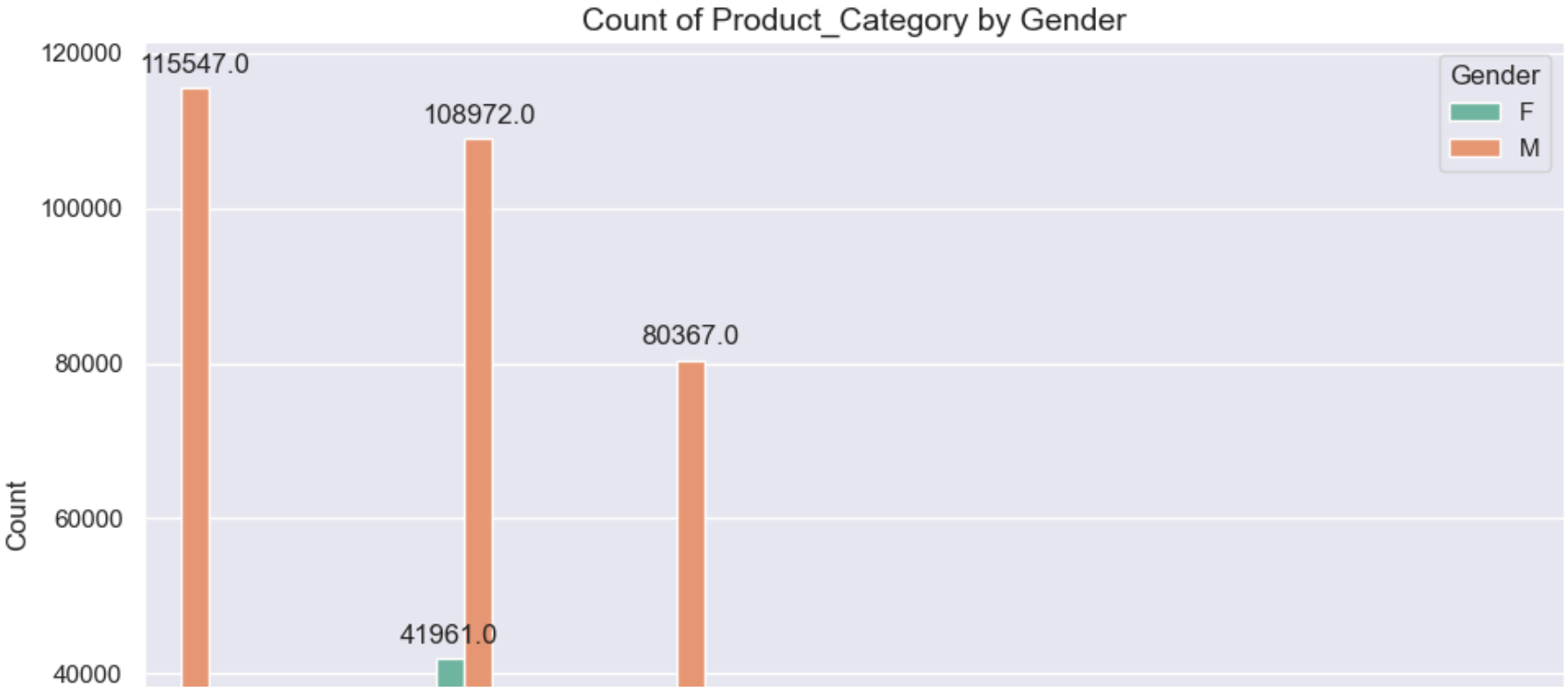

125000
100000
75000
50000
25000
0
Count
35704.0
112016.0
0
42309.0
128866.0
57796.0
A
C
B
City_Category

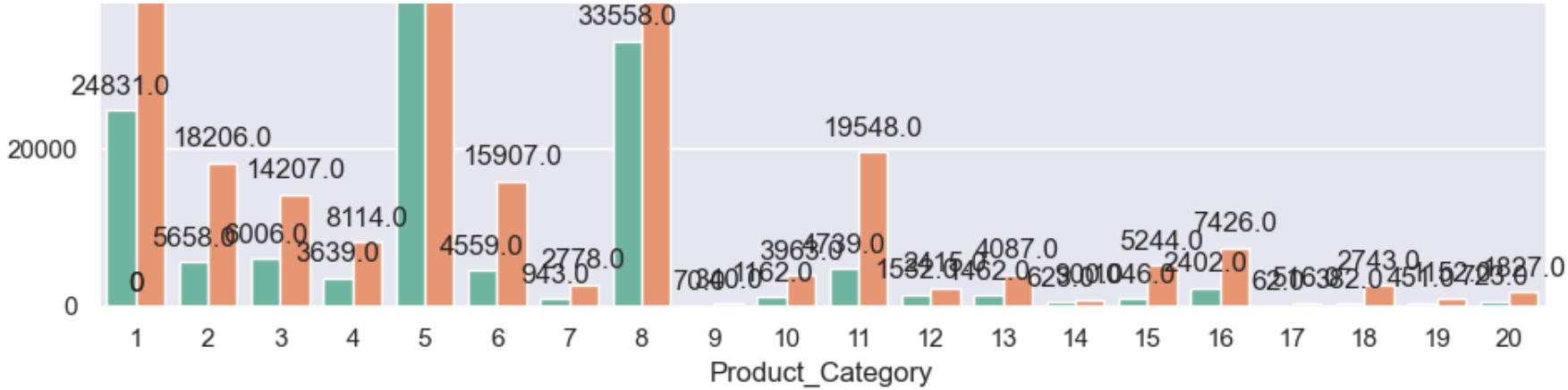

Count of Stay_In_Current_City_Years by Gender
140000
120000
100000
142523.0
Gender
F
M


Count
80000
60000
40000
20000
0
77506.0
66130.0
70765.0
57335.0
51298.0
24332.0
18596.0
24520.0
17063.0
0
2
4+
3
1
0
Stay_In_Current_City_Years


Count of Marital_Status by Gender
250000
200000
150000
Count
245910.0
168349.0
Gender
F
M


100000
50000
0
78821.0
0
56988.0
Unmarried
Married
Marital_Status


Count of Product_Category by Gender
120000
100000
80000
60000
40000
Count
115547.0
108972.0
80367.0
41961.0
Gender
F
M



Insights

Note: Among 550068 transactions there are 5891 unique user_id, indicating same customers buying multiple products.

Note: Among 550068 transactions there are 3631 unique products,with the product having the code P00265242 being the highest seller, with a maximum of 1,880 units sold.

Note: Out of 550068 transactions, 414259 (nearly 75%) were done by male gender indicating a significant disparity in purchase behavior between males and females during the Black Friday event.

Note: We have 7 unique age groups in the dataset. 26 - 35 Age group has maximum of 219587 transactions

Note: Stay_In_Current_City_Years - Customers with 1 year of stay in current city accounted to maximum of 193821 transactions among all the other customers with (0,2,3,4+) years of stay in current city

Note: Marital_Status - 59% of the total transactions were done by Unmarried Customers and 41% by Married Customers.

Note: Customers with Occupation '4' has the highest purchase, and Occupations '0', '7', and '1' also have significant counts.

Note: City_Category 'B' has the most purchases. City_Category 'C' and 'A' follow in terms of count.

Note: Product categories '1' and '5' has higher purchase amounts, indicating that these categories contribute significantly to the overall sales revenue.

In [55]:
```
data.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  object
 8   Product_Category            550068 non-null  int64
 9   Purchase                    550068 non-null  int64
dtypes: int64(4), object(6)
memory usage: 42.0+ MB
```

In [56]:
```
data["User_ID"] = data["User_ID"].astype('category')
data["Occupation"] = data["Occupation"].astype('category')
data["Product_Category"] = data["Product_Category"].astype('category')
```

In [57]:
```
data.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  category
 1   Product_ID                  550068 non-null  object  
 2   Gender                      550068 non-null  object  
 3   Age                         550068 non-null  object  
 4   Occupation                  550068 non-null  category
 5   City_Category               550068 non-null  object  
 6   Stay_In_Current_City_Years  550068 non-null  object  
 7   Marital_Status              550068 non-null  object  
 8   Product_Category            550068 non-null  category
 9   Purchase                    550068 non-null  int64   
dtypes: category(3), int64(1), object(6)
memory usage: 31.6+ MB
```

In [58]:
```
data.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]:
```
data.groupby('Gender')['Purchase'].describe().T
```

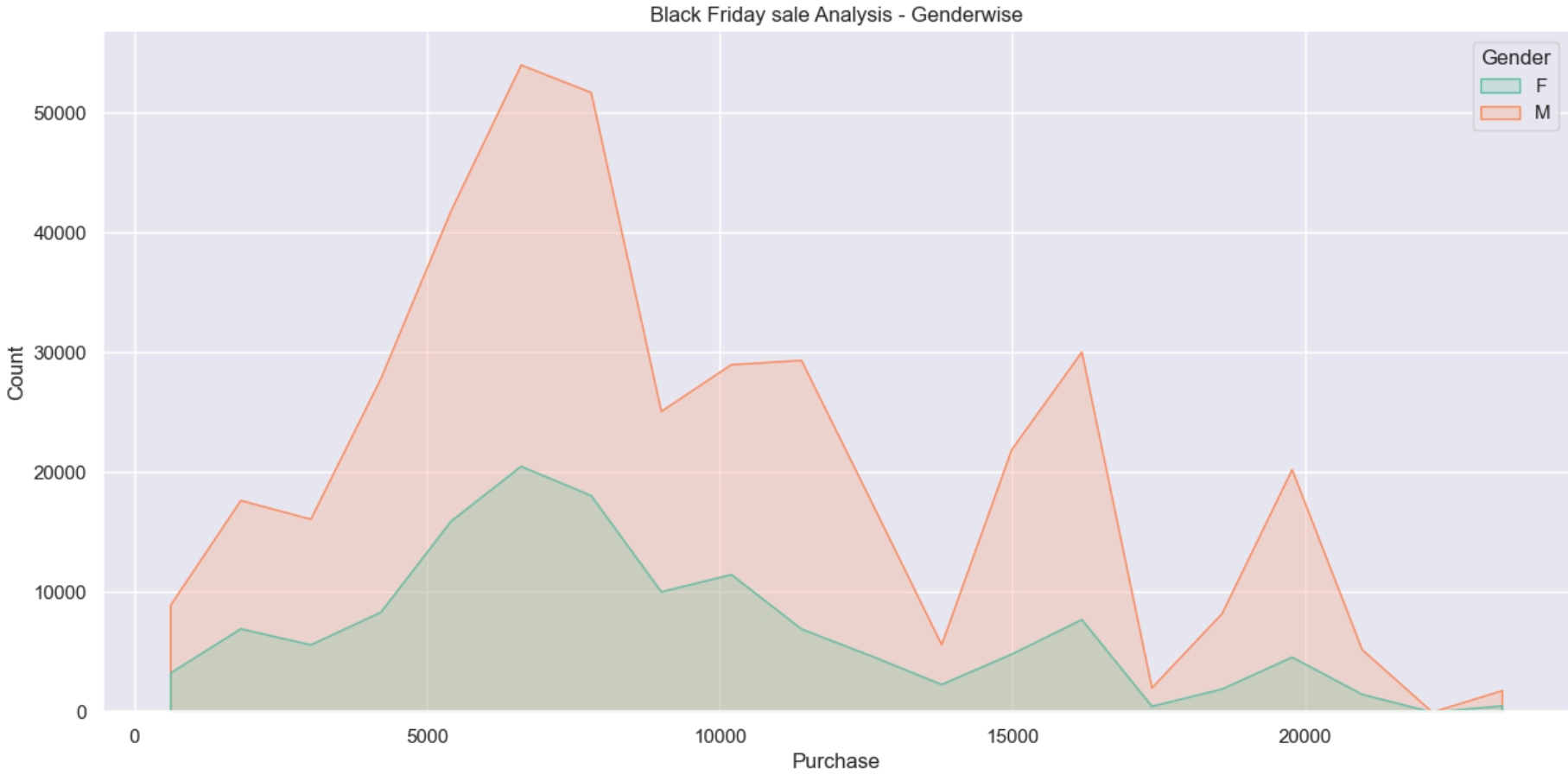
Out[59]:

Gender	F	M
count	135809.000000	414259.00000
mean	8734.565765	9437.52604
std	4767.233289	5092.18621
min	12.000000	12.00000
25%	5433.000000	5863.00000
50%	7914.000000	8098.00000
75%	11400.000000	12454.00000
max	23959.000000	23961.00000

Note: The purchase amounts vary widely, with the minimum recorded purchase being 12 and the maximum reaching to 23961. The median purchase amount of 8047 is notably lower than the mean purchase amount of 9264, indicating a right-skewed distribution where a few high-value purchases pull up the mean

In [60]:

```
plt.figure(figsize=(15,7))
sns.set(style='darkgrid')
sns.histplot(data=data, x = "Purchase", bins=20, hue = "Gender",element='poly',palette= 'Set2')
sns.despine()
plt.title('Black Friday sale Analysis - Genderwise')
plt.show()
```



Note: The total number of male customers (4225) exceeds the total number of female customers (1666).

Note: The average amount spent by male customers (9437) is higher than the average amount spent by female customers (8734).

Note: With a larger male customer base, it is likely that men will make more purchases compared to females.

Note: The higher sales among male customers could be attributed to a product range better suited to their preferences, leading to increased sales of products targeted towards men.

Use the sample average to find out the confidence interval

In [61]:
```
# For female customers
sample_female = data[data['Gender'] == 'F']['Purchase']
mean_female = np.mean(sample_female)
std_female = np.std(sample_female, ddof=1)
n_female = len(sample_female)
confidence_level = 0.95
z_score = stats.norm.ppf(1 - (1 - confidence_level) / 2)
margin_of_error_female = z_score * (std_female / np.sqrt(n_female))
ci_female = (mean_female - margin_of_error_female, mean_female + margin_of_error_female)

print(f'95% Confidence Interval for Female Purchase Amount: {ci_female}')
```

```
95% Confidence Interval for Female Purchase Amount: (8709.21154714068, 8759.919983170272)
```

In [62]:
```
# For male customers
sample_male = data[data['Gender'] == 'M']['Purchase']
mean_male = np.mean(sample_male)
std_male = np.std(sample_male, ddof=1)
n_male = len(sample_male)
margin_of_error_male = z_score * (std_male / np.sqrt(n_male))
ci_male = (mean_male - margin_of_error_male, mean_male + margin_of_error_male)

print(f'95% Confidence Interval for Male Purchase Amount: {ci_male}')
```

```
95% Confidence Interval for Male Purchase Amount: (9422.01944736257, 9453.032633581959)
```

Compare the confidence intervals

In [63]:
```
if (ci_female[0] <= ci_male[1] and ci_female[1] >= ci_male[0]):
    print("The confidence intervals for male and female purchase amounts overlap.")
```

```
else:
    print("The confidence intervals for male and female purchase amounts do not overlap.")
```

```
The confidence intervals for male and female purchase amounts do not overlap.
```

Analysis for Marital Status

In [64]:
```
# Calculate the average purchase amount for married and unmarried customers
average_purchase_married = data[data['Marital_Status'] == 1]['Purchase'].mean()
average_purchase_unmarried = data[data['Marital_Status'] == 0]['Purchase'].mean()

print(f'Average Purchase Amount for Married: {average_purchase_married}')
print(f'Average Purchase Amount for Unmarried: {average_purchase_unmarried}')
```

```
Average Purchase Amount for Married: nan
Average Purchase Amount for Unmarried: nan
```

Confidence interval calculation for married and unmarried customers

In [65]:
```
# For married customers
sample_married = data[data['Marital_Status'] == 1]['Purchase']
mean_married = np.mean(sample_married)
std_married = np.std(sample_married, ddof=1)
n_married = len(sample_married)
margin_of_error_married = z_score * (std_married / np.sqrt(n_married))
ci_married = (mean_married - margin_of_error_married, mean_married + margin_of_error_married)

print(f'95% Confidence Interval for Married Purchase Amount: {ci_married}')
```

```
95% Confidence Interval for Married Purchase Amount: (nan, nan)
```

In [66]:
```
# For unmarried customers
sample_unmarried = data[data['Marital_Status'] == 0]['Purchase']
mean_unmarried = np.mean(sample_unmarried)
std_unmarried = np.std(sample_unmarried, ddof=1)
n_unmarried = len(sample_unmarried)
margin_of_error_unmarried = z_score * (std_unmarried / np.sqrt(n_unmarried))
ci_unmarried = (mean_unmarried - margin_of_error_unmarried, mean_unmarried + margin_of_error_unmarried)
```

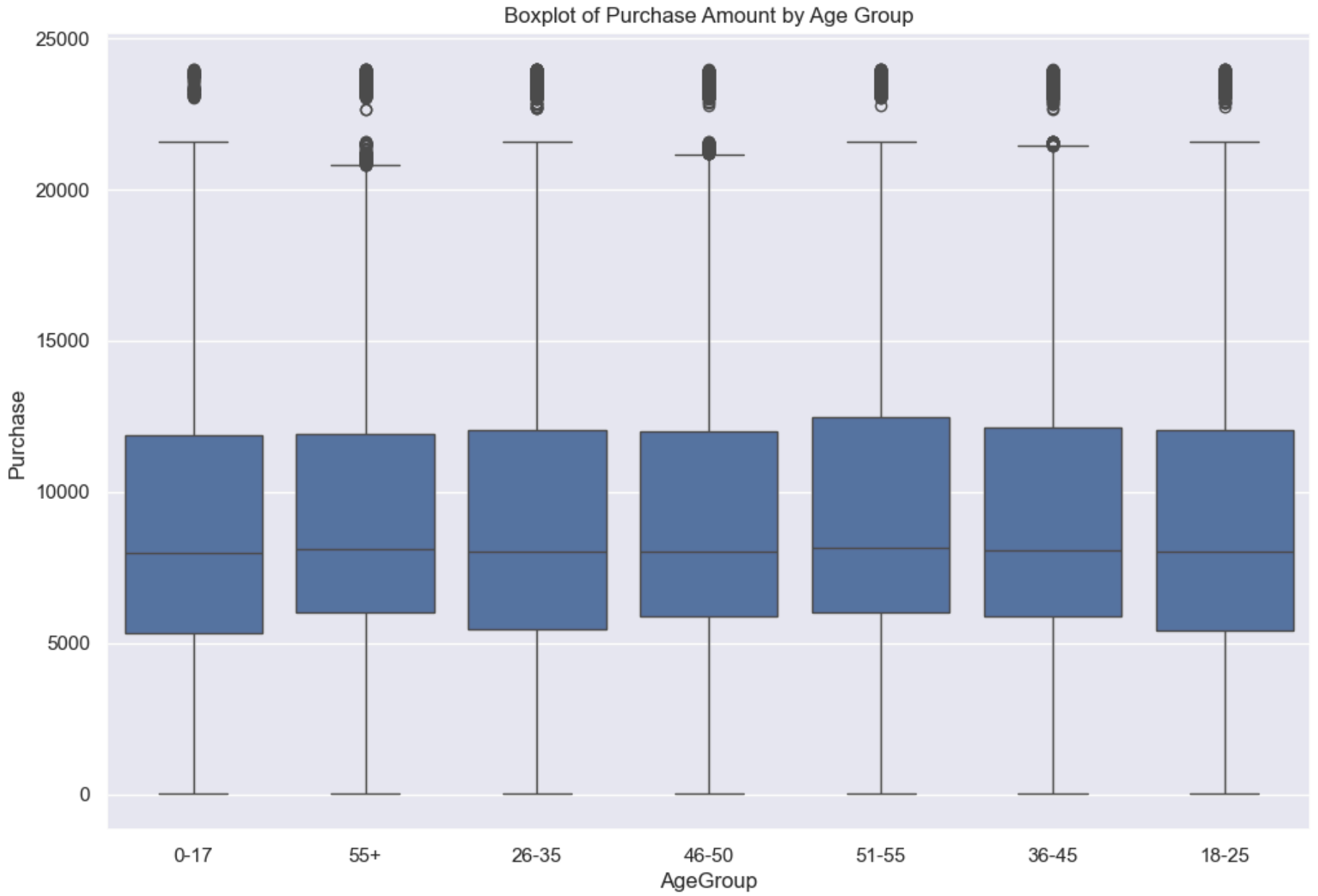
```
print(f'95% Confidence Interval for Unmarried Purchase Amount: {ci_unmarried}')
```

```
95% Confidence Interval for Unmarried Purchase Amount: (nan, nan)
```

Analysis for Age groups

In [67]:
```
data["AgeGroup"] = data["Age"]
```

In [68]:
```
# Plot the distribution of purchase amounts by Age Group
plt.figure(figsize=(12, 8))
sns.boxplot(x='AgeGroup', y='Purchase', data=data)
plt.title('Boxplot of Purchase Amount by Age Group')
plt.show()
```



In [69]:
```
# Calculate average purchase for each age group
age_group_means = data.groupby('AgeGroup')['Purchase'].mean()
```

```
print(age_group_means)
```

```
AgeGroup
0-17     8933.464640
18-25    9169.663606
26-35    9252.690633
36-45    9331.350695
46-50    9208.625697
51-55    9534.808031
55+      9336.280459
Name: Purchase, dtype: float64
```

Note: The confidence interval computed using the entire dataset is wider for males compared to females, indicating higher variability in the amount spent by males.

Note: The confidence intervals for different sample sizes overlap, suggesting that observed differences may not be statistically significant.

Note: Larger sample sizes result in more normally shaped distributions of means due to the Central Limit Theorem.

Note: The confidence interval for the 'Married' group is wider than that for the 'Single' group, indicating higher variability in the amount spent for married individuals.

Recommendations for Walmart

Recommendations:

1. Age Group Targeting: Concentrate marketing efforts on individuals aged 26-35, as they show the highest purchase rates. Design promotions and advertisements to appeal to this demographic.
2. Occupation-Specific Offers: Since occupation category '4' has the highest representation and significant purchases, customize product offerings or promotions to cater to individuals in this occupation.

3. City Category Focus: Allocate promotional resources strategically with an emphasis on City_Category 'B', where the highest purchase levels are observed. Tailor promotions to match the preferences of customers in this category.

4. Marital Status Campaigns: Launch targeted marketing campaigns for single individuals, as they contribute significantly more to overall sales. Understand and cater to the preferences of this demographic.

5. Loyalty Programs: Develop strategies to encourage customers to stay in their current city for more than a year. Consider loyalty programs or special incentives for long-term residents to boost their purchasing behavior.

6. Product Category Optimization: Optimize inventory and promotions for product categories '1' and '5', as they show higher purchase amounts. Manage these categories strategically to maximize overall sales revenue.

7. Gender-Targeted Marketing: Implement gender-targeted marketing strategies, especially focusing on males across various age groups. Use insights from age-based gender analysis to tailor promotions effectively.

8. Top Occupation Incentives: Design promotions or incentives based on the top occupations, such as '0' and '4', to further boost sales from these occupational groups.

9. City-Specific Initiatives: Implement specific initiatives, offers, or events in City_Category 'B' to capitalize on the higher purchasing behavior observed in this category.

10. Male Customer Preferences: Analyze the product preferences of male customers to inform product development. Ensure the product range aligns with the preferences of the larger male customer base to increase sales.

11. Female Customer Targeting: Develop marketing strategies to target female customers more effectively, especially around Black Friday.

12. Marital Status Spending Analysis: Investigate the spending patterns of married vs. unmarried customers to identify potential market opportunities.

13. Age-Specific Promotions: Focus on age-specific promotions and products to better cater to the needs of different age groups.

14. Spending Behavior Analysis: Conduct further analysis to explore the underlying factors contributing to spending differences between genders and marital statuses.

15. Expanded Dataset Analysis: Consider expanding the dataset to include other relevant features that may impact customer spending behavior.

Action items for Walmart

Actionable Items:

1. Targeted Female Marketing Campaigns: Design and implement marketing campaigns specifically for female customers during peak shopping periods such as Black Friday.
2. Loyalty Programs for Married Customers: Develop loyalty programs or special discount offers to encourage higher spending among married customers.
3. Age-Specific Product Customization: Tailor product offerings and promotions for various age groups based on their distinct spending patterns.
4. Inventory and Product Placement Optimization: Utilize insights from the analysis to enhance inventory management and optimize product placement in stores.