In [1]:
```python
# Importing the necessary libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
```

In [2]:
```python
# Load the dataset

data = pd.read_csv('walmart.csv')
```

In [3]:
```python
# Data Types of Columns

data.dtypes
```

Out[3]:
```
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 [4]:
```python
# Dataset Shape

data.shape
```

Out[4]:
```
(550068, 10)
```

In [5]:
```python
# Data info

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 [6]:
```python
# Top 5 rows of the dataframe

data.head()
```

Out[6]:

	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

In [7]:
```python
# Missing Values in Each Column

data.isnull().sum()
```

Out[7]:
```
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 [8]:
```python
# Duplicate values check
```

```
data.duplicated().sum()
```

Out[8]:

```
0
```

In [9]:

```
# Uniques values of each columns

data.nunique()
```

Out[9]:

```
User_ID                        5891
Product_ID                     3631
Gender                            2
Age                               7
Occupation                       21
City_Category                     3
Stay_In_Current_City_Years        5
Marital_Status                    2
Product_Category                 20
Purchase                      18105
dtype: int64
```

In [10]:

```
# Convert all columns (except Purchase) to categorical type in the DataFrame

for _ in data.columns[:-1]:
 data[_] = data[_].astype('category')

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  category
 2   Gender                      550068 non-null  category
 3   Age                         550068 non-null  category
 4   Occupation                  550068 non-null  category
 5   City_Category               550068 non-null  category
 6   Stay_In_Current_City_Years  550068 non-null  category
 7   Marital_Status              550068 non-null  category
 8   Product_Category            550068 non-null  category
 9   Purchase                    550068 non-null  int64
dtypes: category(9), int64(1)
memory usage: 10.3 MB
```

In [11]:

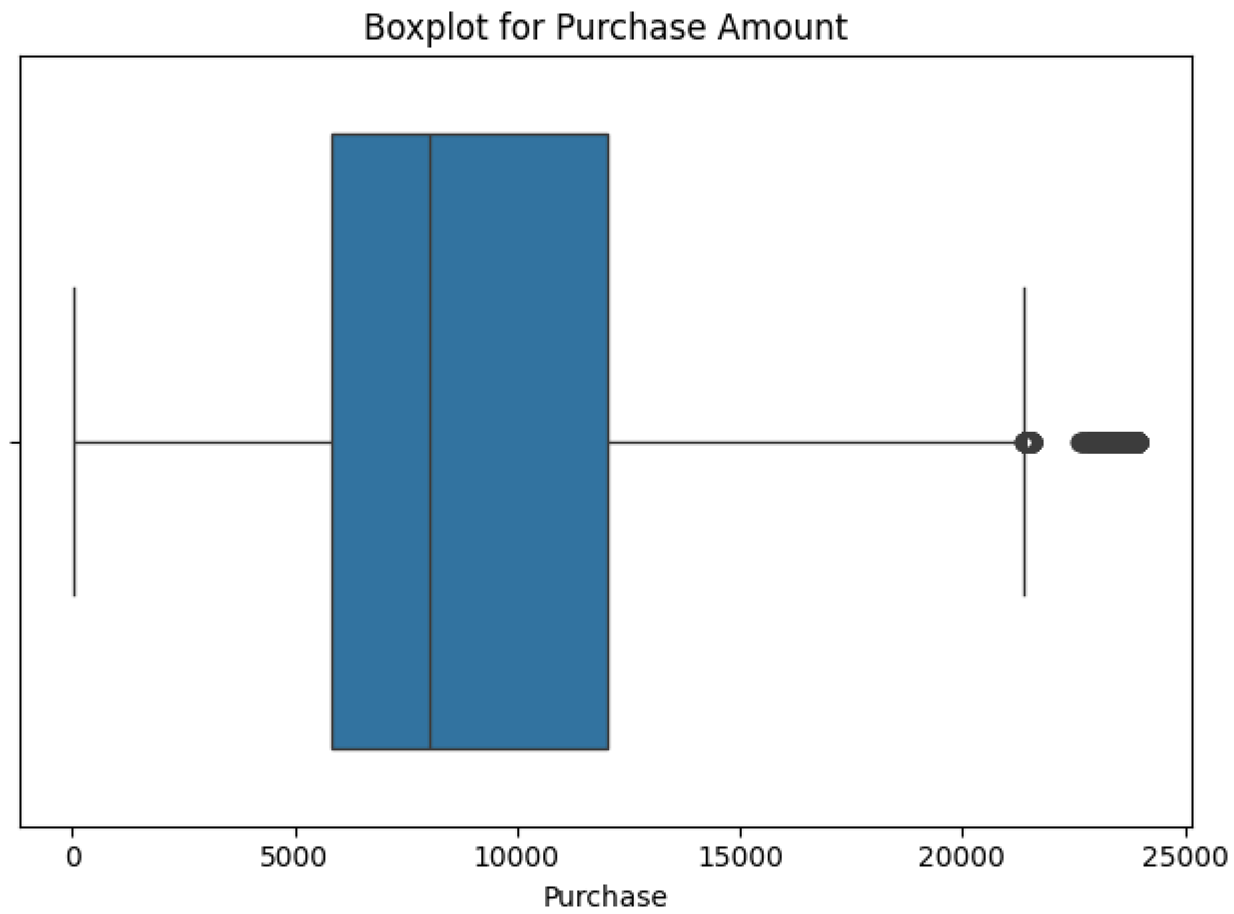
```
data.describe()
```

Out[11]:

	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 [12]:

```
# 2. Detect Null values and outliers
# Boxplot for Purchase Amount (Outlier Detection)

sns.boxplot(data=data['Purchase'], orient='h')
plt.title('Boxplot for Purchase Amount')
plt.tight_layout()
plt.show()
```



In [13]:
```
# Calculate quartiles and IQR for the specified column
Q1 = np.percentile(data['Purchase'], 25)
Q3 = np.percentile(data['Purchase'], 75)
IQR = Q3 - Q1

# Upper and lower bounds for outliers
upper_bound = Q3 + (1.5 * IQR)
lower_bound = Q1 - (1.5 * IQR)

# Outliers in the specified column
upper_outliers_data = data[data['Purchase'] > upper_bound]
lower_outliers_data = data[data['Purchase'] < lower_bound]

# Count of outliers
upper_count = len(upper_outliers_data)
lower_count = len(lower_outliers_data)

total_count = upper_count + lower_count
```

In [14]:
```
print(f"Upper Outliers Count: {upper_count}")
print(f"Lower Outliers Count: {lower_count}")
print(f"Overall Outliers Count: {total_count}")
```

```
Upper Outliers Count: 2677
Lower Outliers Count: 0
Overall Outliers Count: 2677
```

In [15]:
```
# Extract rows where 'Purchase' values are greater than the upper bound to identify outliers

outliers_data = data[data['Purchase'] > upper_bound]
outliers_data
```

Out[15]:

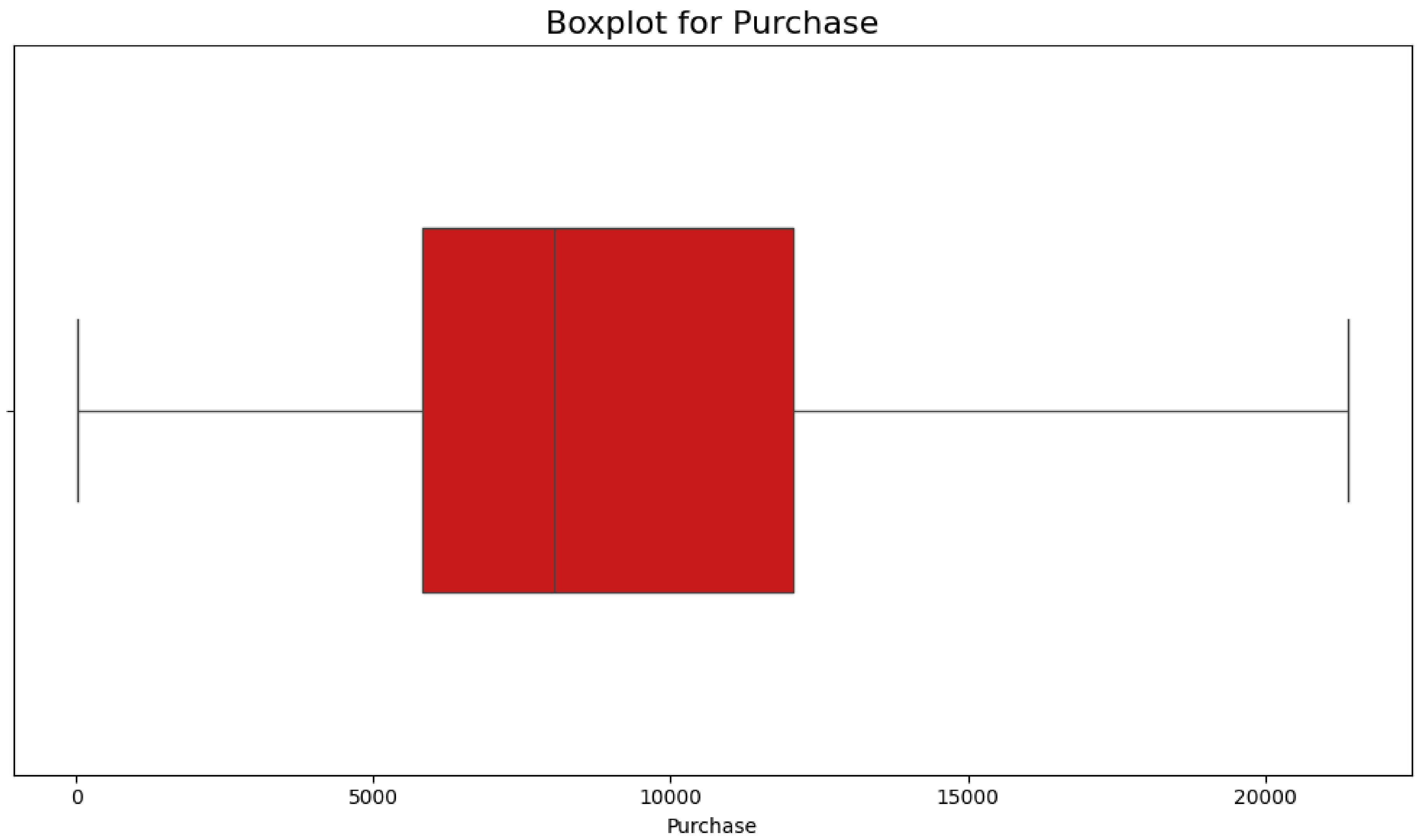
	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
343	1000058	P00117642	M	26-35	2	B	3	0	10	23603
375	1000062	P00119342	F	36-45	3	A	1	0	10	23792
652	1000126	P00087042	M	18-25	9	B	1	0	10	23233
736	1000139	P00159542	F	26-35	20	C	2	0	10	23595
1041	1000175	P00052842	F	26-35	2	B	1	0	10	23341
...	...	...	...	...	...	...	...	...	...	...
544488	1005815	P00116142	M	26-35	20	B	1	0	10	23753
544704	1005847	P00085342	F	18-25	4	B	2	0	10	23724
544743	1005852	P00202242	F	26-35	1	A	0	1	10	23529
545663	1006002	P00116142	M	51-55	0	C	1	1	10	23663
545787	1006018	P00052842	M	36-45	1	C	3	0	10	23496

2677 rows × 10 columns

In [16]:
```
clipped_data = np.clip(data['Purchase'], lower_bound, upper_bound)
```

In [17]:
```
plt.figure(figsize=(10, 6))

# Create a box plot for clipped data
```

```
sns.boxplot(x=clipped_data,color='#e60000', width=0.5, orient='h')
plt.title('Boxplot for Purchase', fontsize=16)
plt.tight_layout()
plt.show()
```



In [18]:
```
# Map numerical values in 'Marital_Status' to categorical labels\

data['Marital_Status'] = data['Marital_Status'].apply(lambda x: 'Married' if x == 1 else 'Single')
```

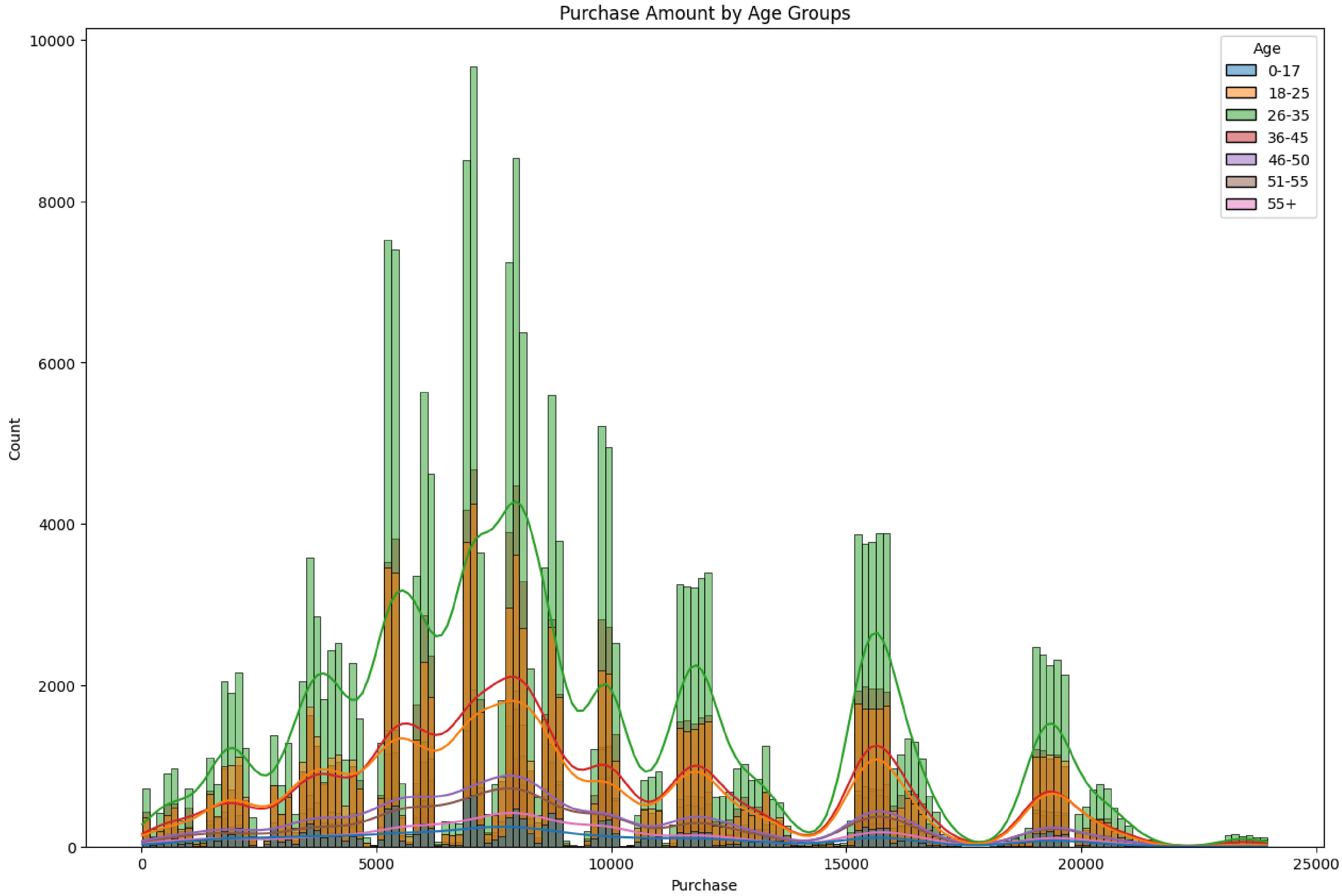
In [19]:
```
data.head()
```

Out[19]:

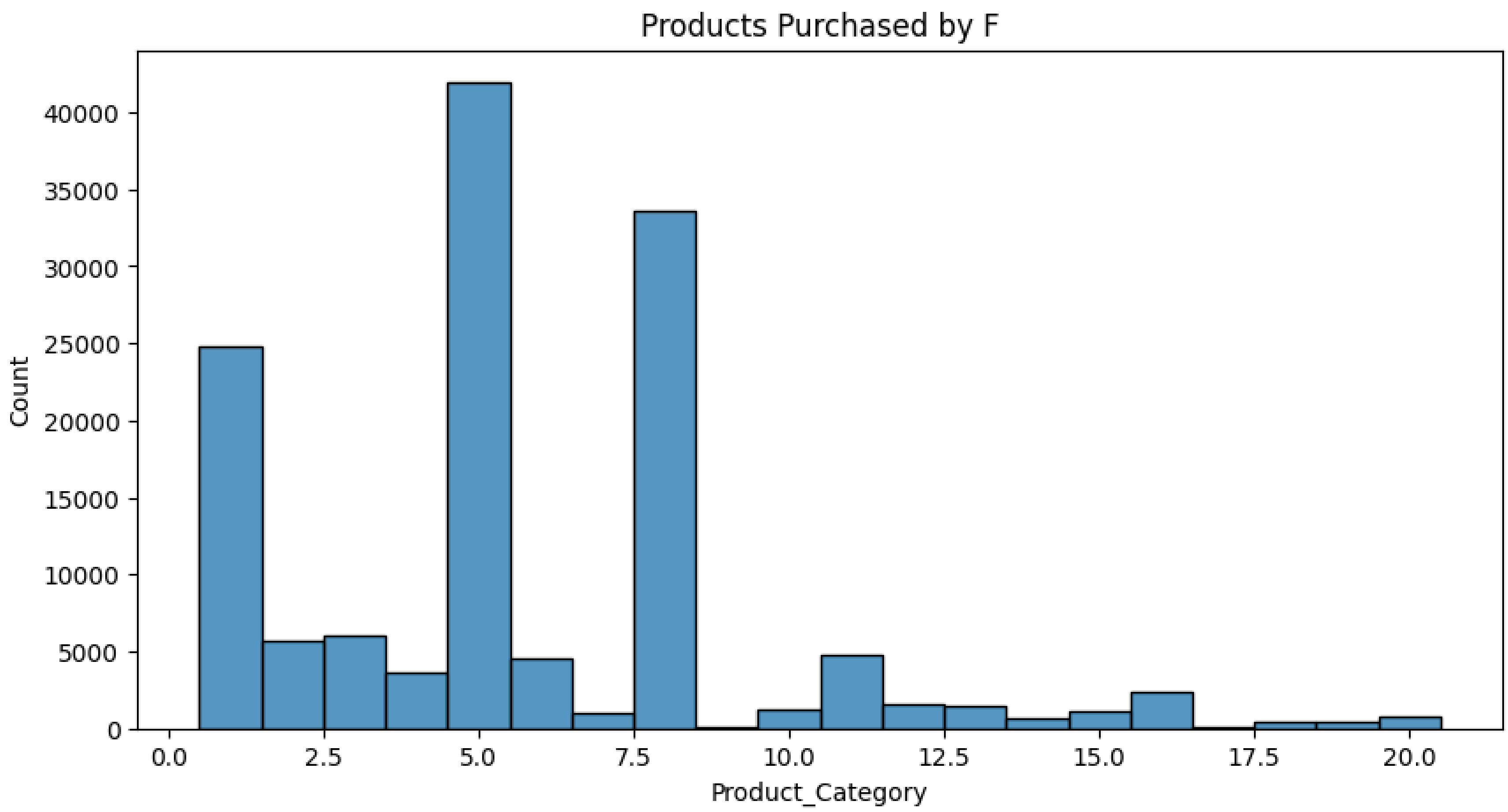
	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10	A	2	Single	3	8370
1	1000001	P00248942	F	0-17	10	A	2	Single	1	15200
2	1000001	P00087842	F	0-17	10	A	2	Single	12	1422
3	1000001	P00085442	F	0-17	10	A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

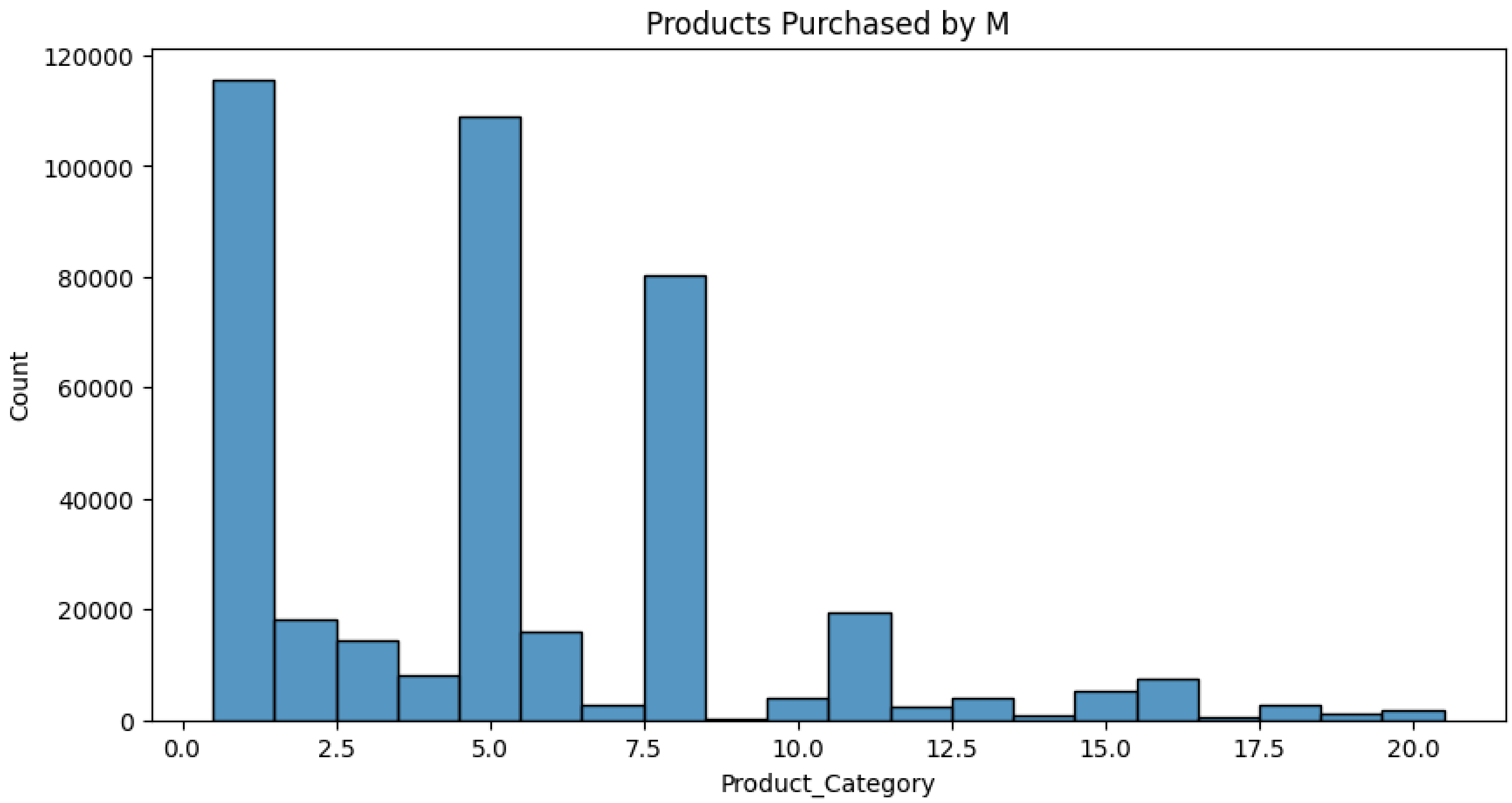
Univariate Analysis

In [22]:
```
# Products Purchased by Age Groups

plt.figure(figsize=(15,10))
sns.histplot(data=data, x='Purchase', hue='Age', kde=True)
plt.title('Purchase Amount by Age Groups')
plt.show()
```



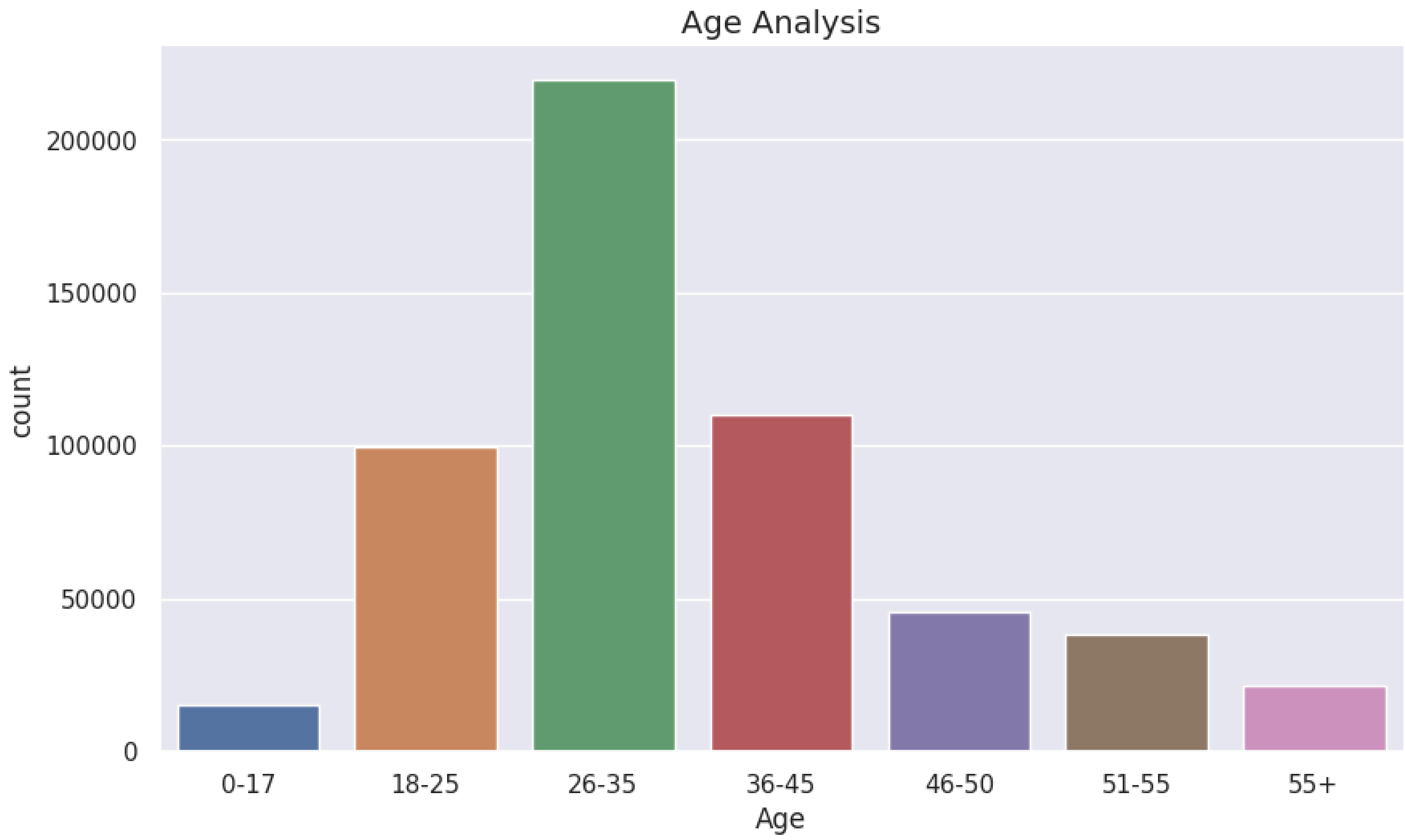
In [33]:
```
# Product Category Preferences by Gender

genders = data['Gender'].unique()

for gender in genders:
    plt.figure(figsize=(10,5))
    sns.histplot(data=data[data['Gender'] == gender], x='Product_Category')
    plt.title(f'Products Purchased by {gender}')
    plt.show()
```

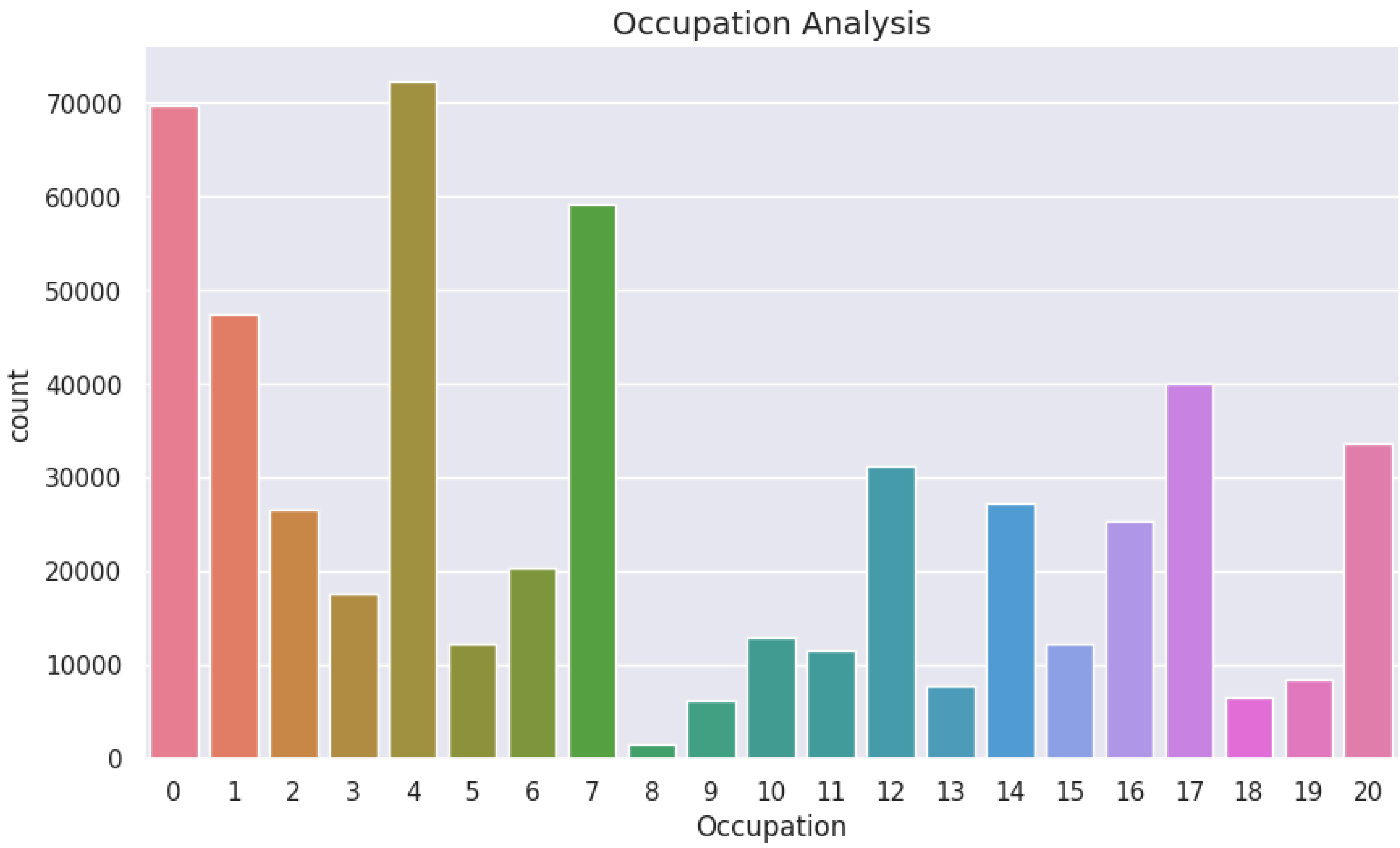


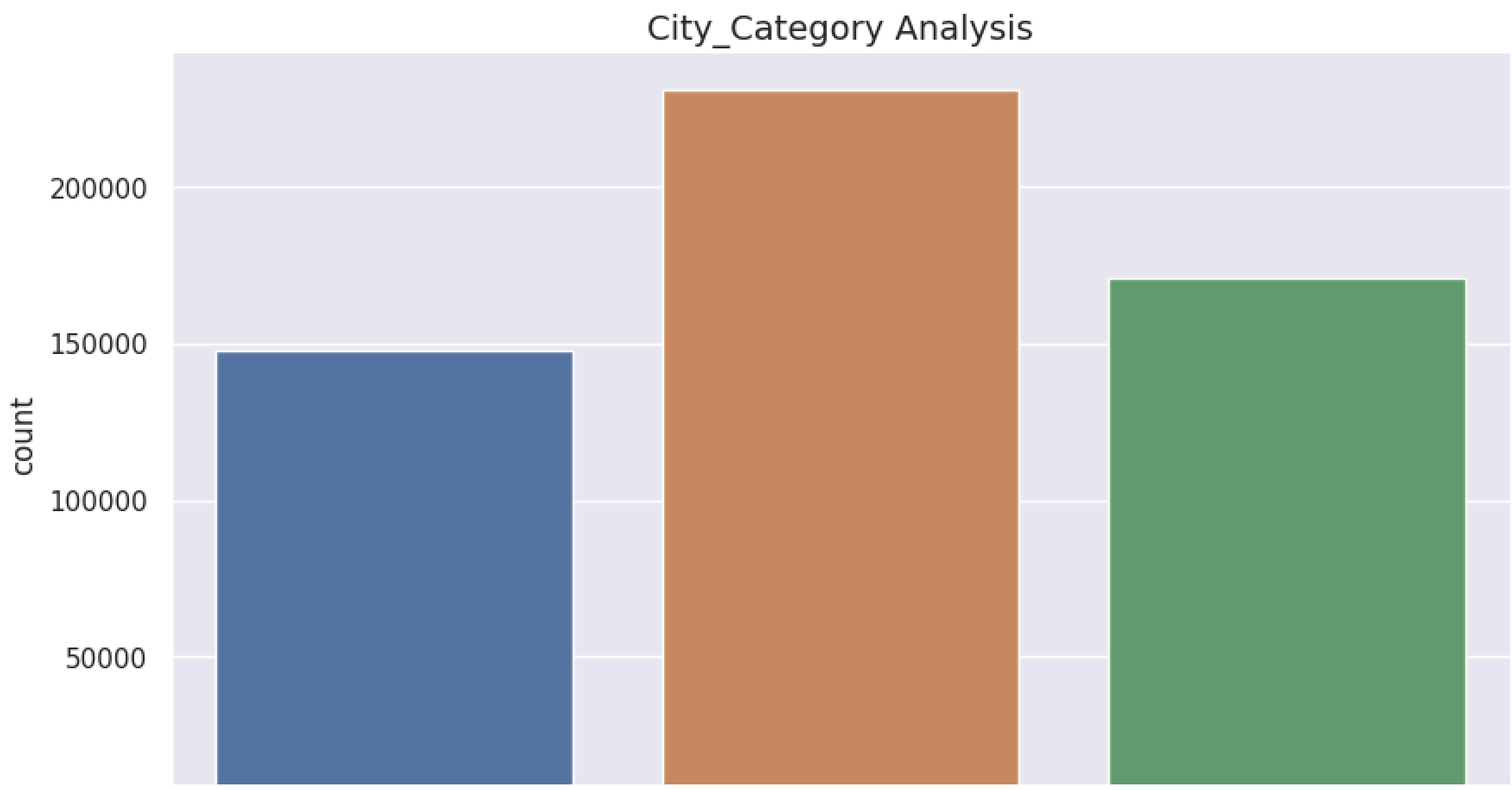


In [97]:

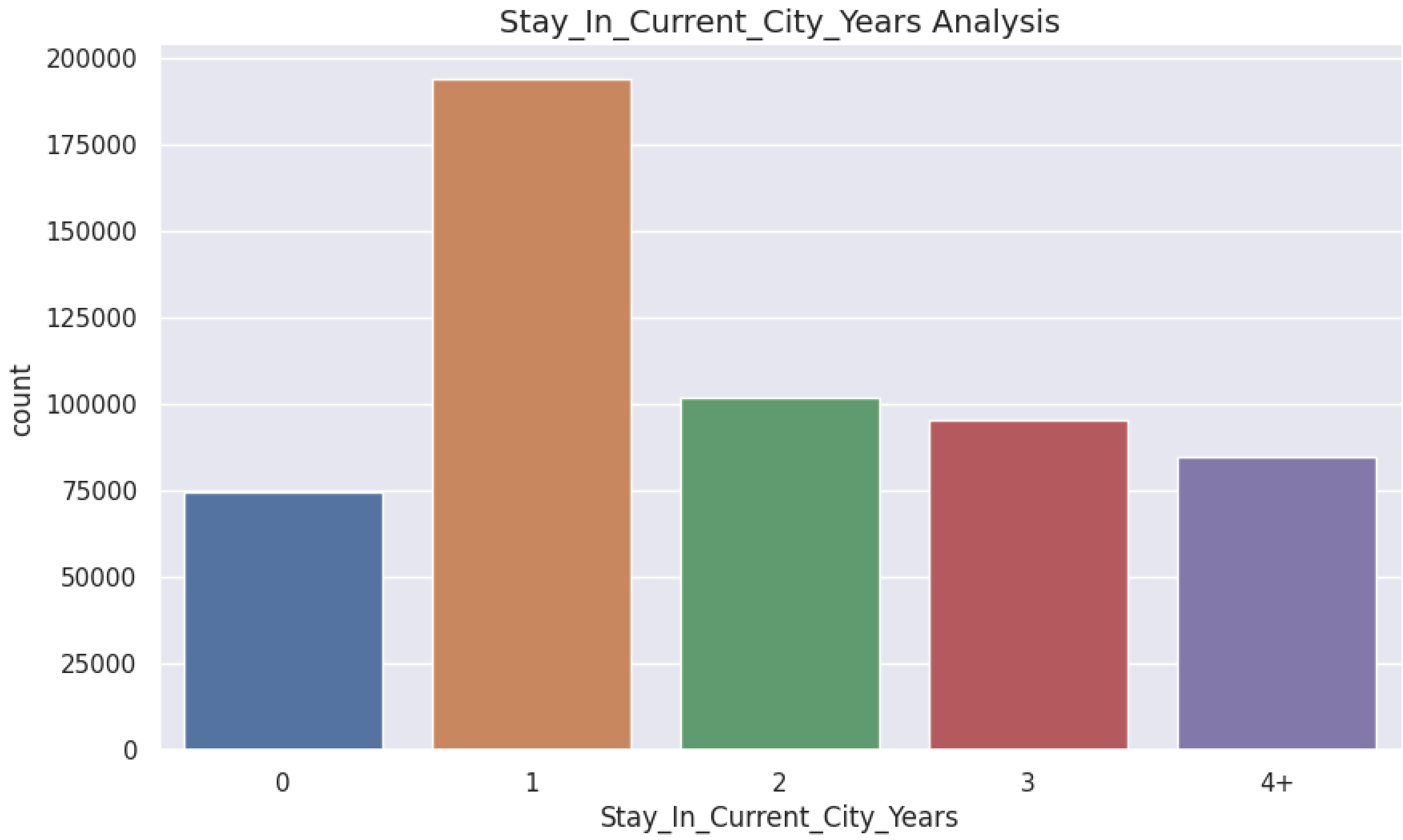
```
category = ['Age', 'Occupation', 'City_Category', 'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category']

plt.figure(figsize=(10,40))

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

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

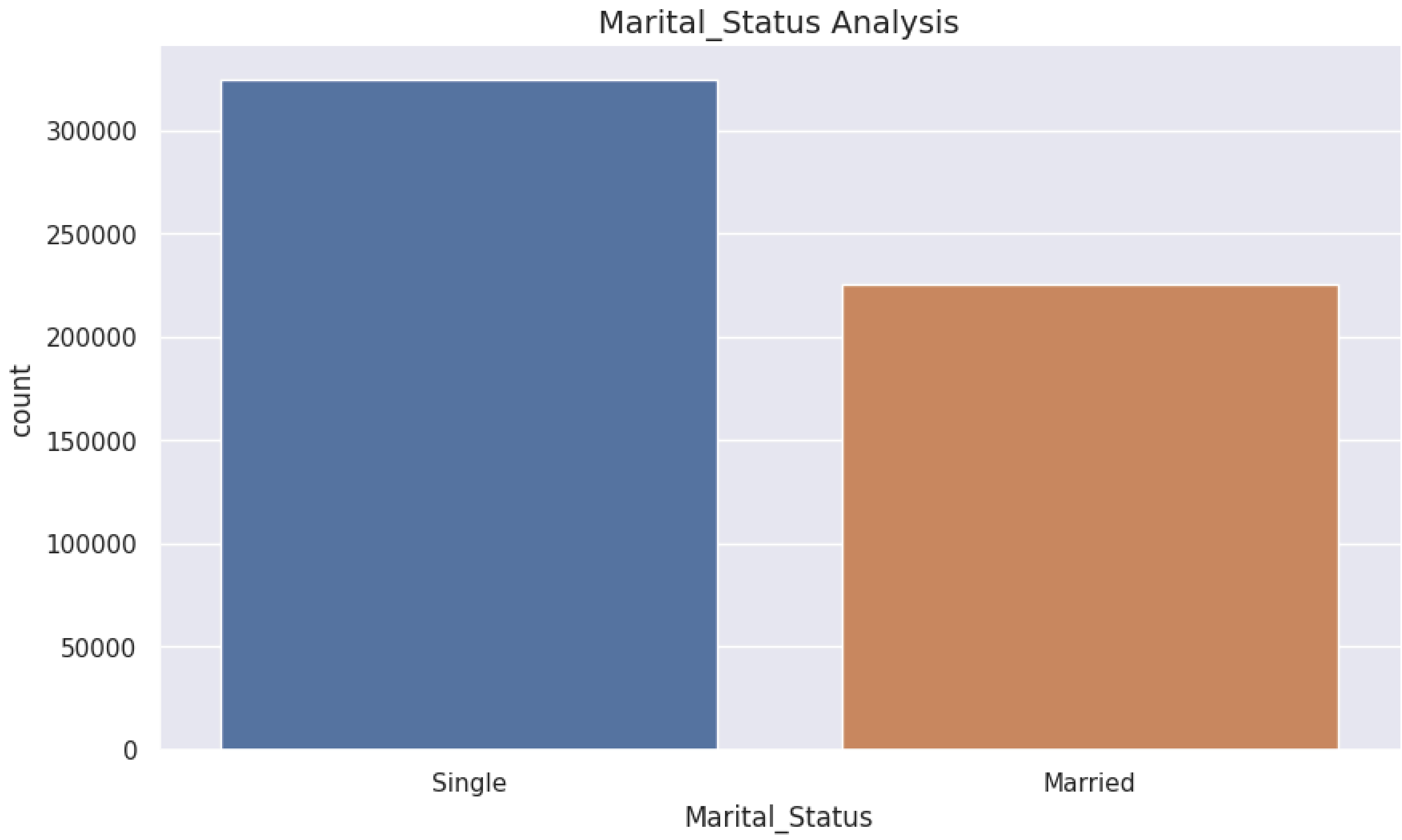

Age Analysis
count
200000
150000
100000
50000
0
0-17
18-25
26-35
36-45
46-50
51-55
55+
Age

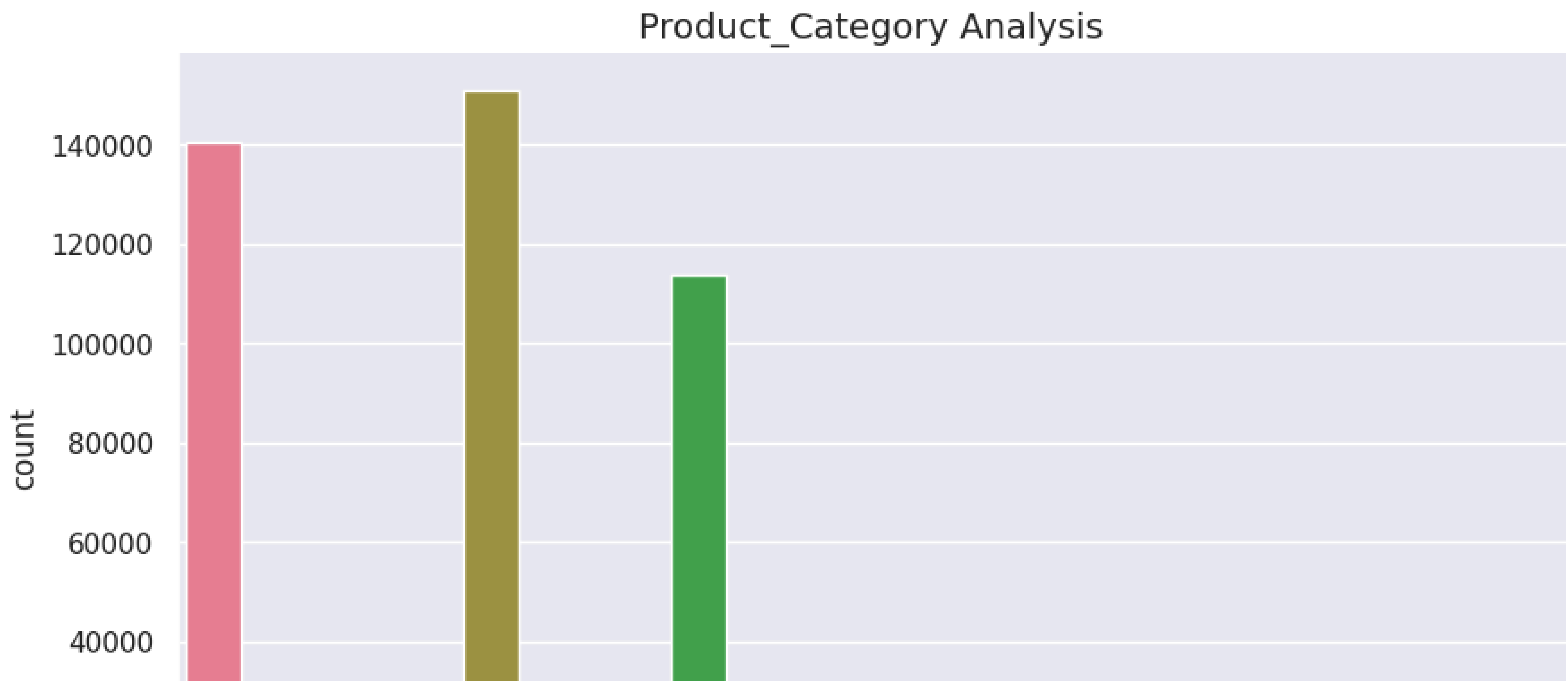

Occupation Analysis
count
70000
60000
50000
40000
30000
20000
10000
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Occupation


City_Category Analysis
count
200000
150000
100000
50000


0
A
B
C
City_Category


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


Marital_Status Analysis
300000
250000
200000
150000
100000
50000
0
count
Single
Married
Marital_Status


Product_Category Analysis
140000
120000
100000
80000
60000
40000
count

Insights:

Age Group Distribution:

The age group '26-35' has the highest count, indicating that customers in this age range make the most purchases. It is followed by the age groups '36-45' and '18-25'.

Occupation Analysis:

Occupation '4' has the highest count, suggesting that customers with occupation '4' have the highest representation in the dataset. Occupations '0', '7', and '1' also have significant counts.

City Category Distribution:

City_Category 'B' has the highest count, indicating that customers from City_Category 'B' have made the most purchases. City_Category 'C' and 'A' follow in terms of count.

Marital Status Impact:

Customers with a marital status of 'Single' have a higher count compared to those who are 'Married', suggesting that single individuals make more purchases in the dataset.

City Residence Duration Impact:

Customers who have stayed in their current city for more than 1 year show a higher purchase tendency, suggesting a positive correlation between the duration of stay and purchasing behavior.

Product Category Purchase Analysis:

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

Bivariate Analysis

In [40]:
```
pivot = lambda index: data.pivot_table(index=data[index], columns='Gender', aggfunc='size', fill_value=0)
```

In [41]:
```
pivot('Age')
```

Out[41]:

Gender	F	M
Age		
0-17	5083	10019
18-25	24628	75032
26-35	50752	168835
36-45	27170	82843
46-50	13199	32502
51-55	9894	28607
55+	5083	16421

In [42]:
```
pivot('Occupation')
```

Out[42]:

Gender	F	M
Occupation		
0	18112	51526
1	17984	29442
2	8629	17959
3	7919	9731
4	17836	54472
5	2220	9957
6	8160	12195
7	10028	49105
8	361	1185
9	5843	448
10	4003	8927
11	1500	10086
12	3469	27710
13	1498	6230
14	6763	20546
15	2390	9775
16	4107	21264
17	3929	36114
18	230	6392
19	2017	6444
20	8811	24751

In [43]:
```
pivot('City_Category')
```

Out[43]:

Gender	F	M
City_Category		
A	35704	112016
B	57796	173377
C	42309	128866

In [44]:
```
pivot('Stay_In_Current_City_Years')
```

Out[44]:

Gender	F	M
Stay_In_Current_City_Years		
0	17063	57335
1	51298	142523
2	24332	77506
3	24520	70765
4+	18596	66130

In [45]:
```
pivot('Marital_Status')
```

Out[45]:

Gender	F	M
Marital_Status		
Single	78821	245910
Married	56988	168349

In [46]:
```
data.columns
```

Out[46]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

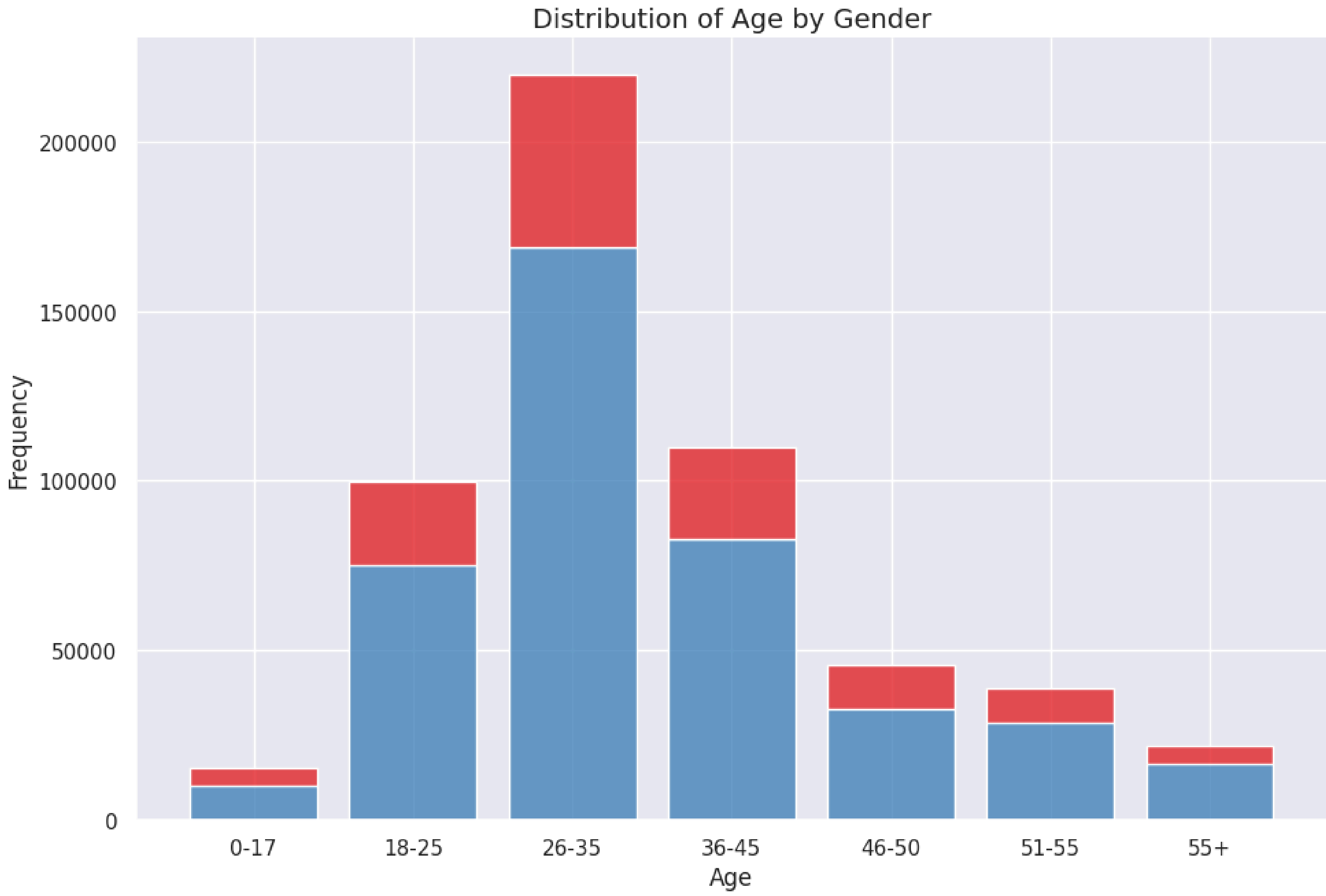
In [47]:
```
pivot('Product_Category')
```

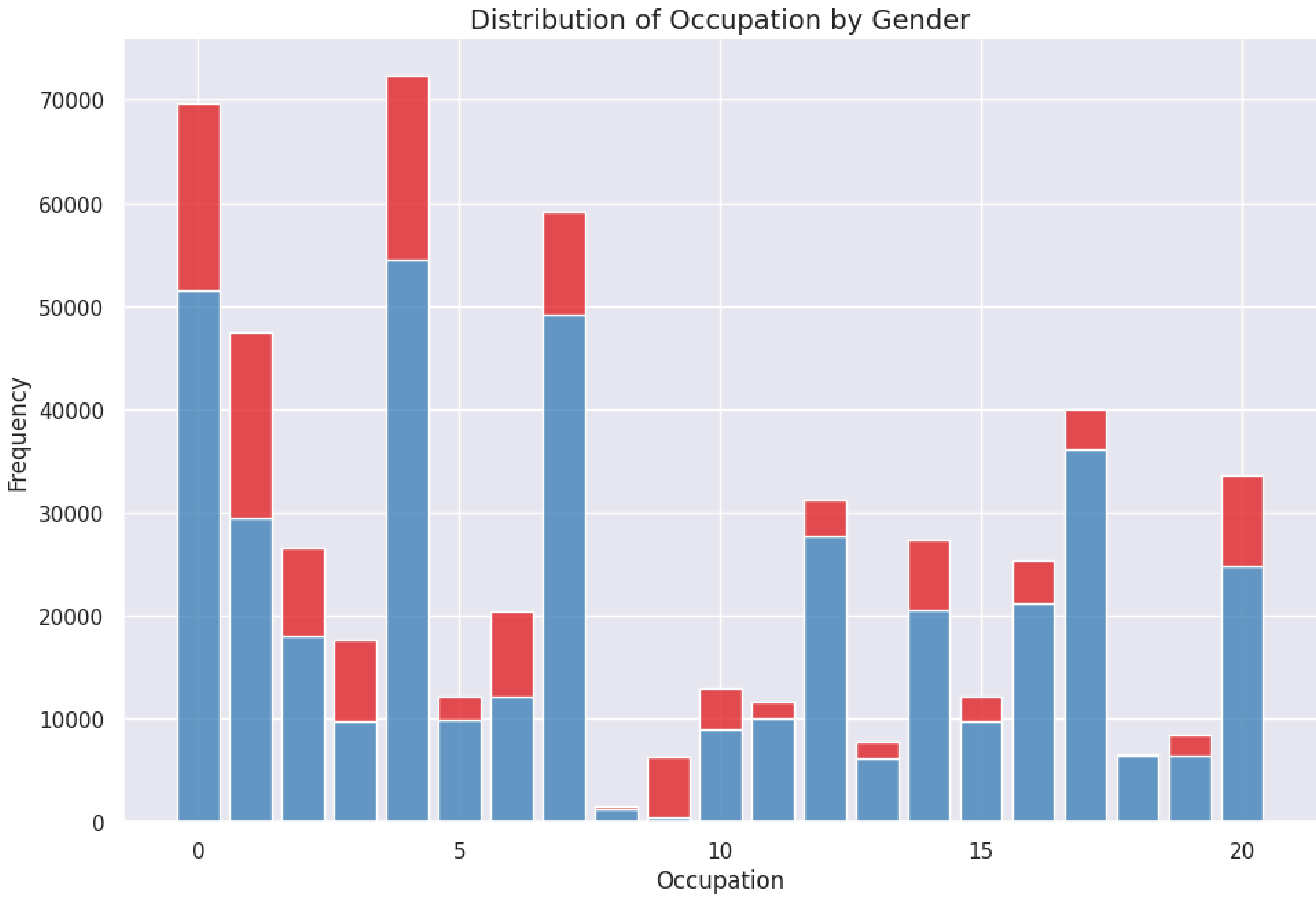
Out[47]:

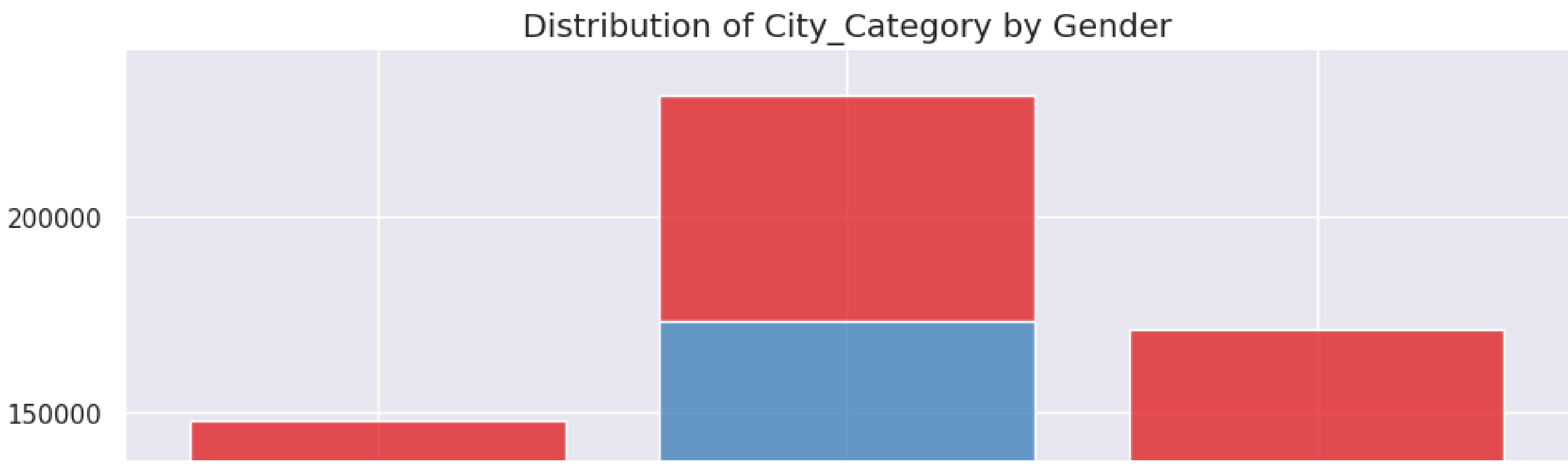
Gender	F	M
Product_Category		
1	24831	115547
2	5658	18206
3	6006	14207
4	3639	8114
5	41961	108972
6	4559	15907
7	943	2778
8	33558	80367
9	70	340
10	1162	3963
11	4739	19548
12	1532	2415
13	1462	4087
14	623	900
15	1046	5244
16	2402	7426
17	62	516
18	382	2743
19	451	1152
20	723	1827

In [50]:

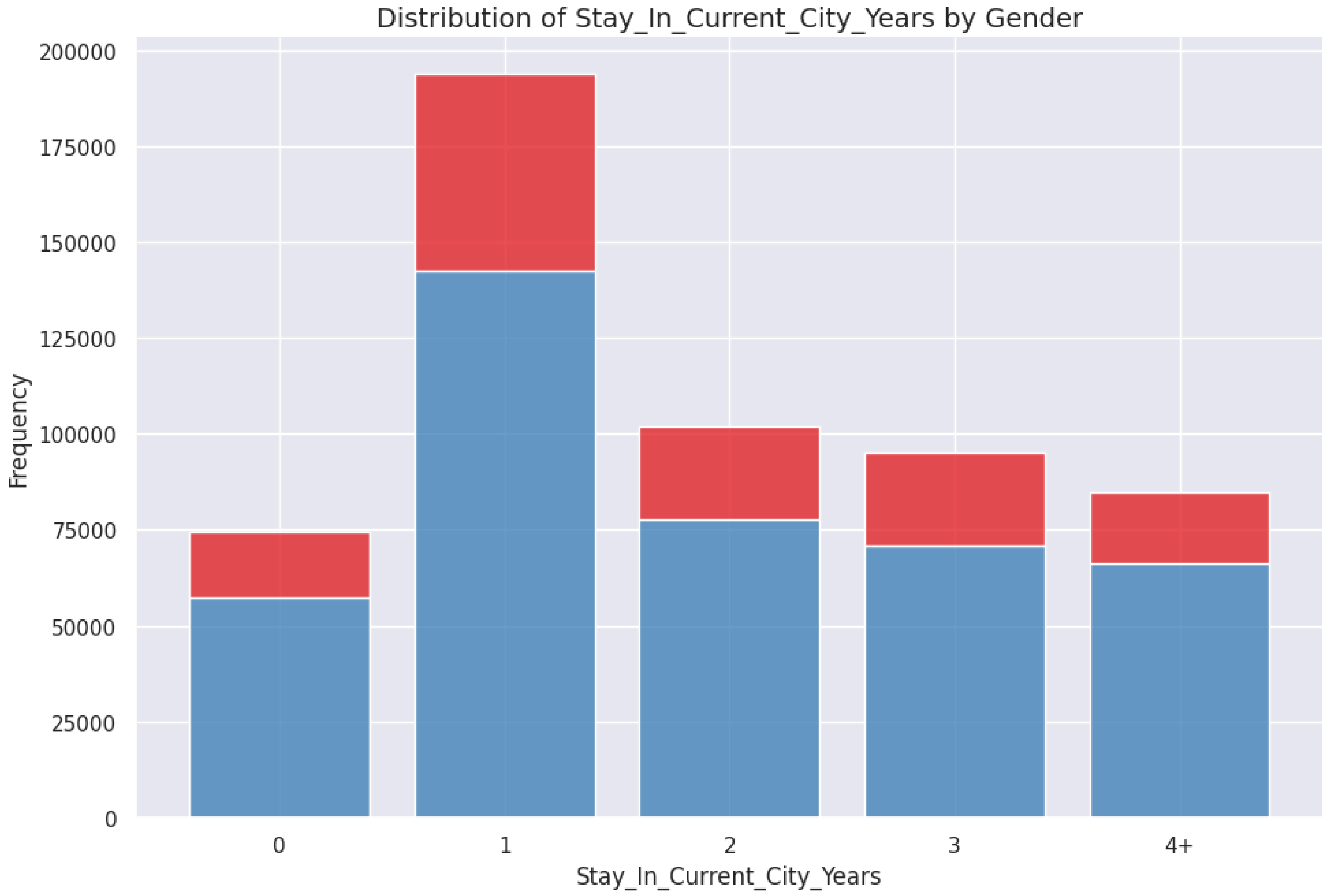
```
plt.figure(figsize=(10, 40))
sns.set(style='darkgrid')

# 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)

    plt.tight_layout()

plt.show()
```

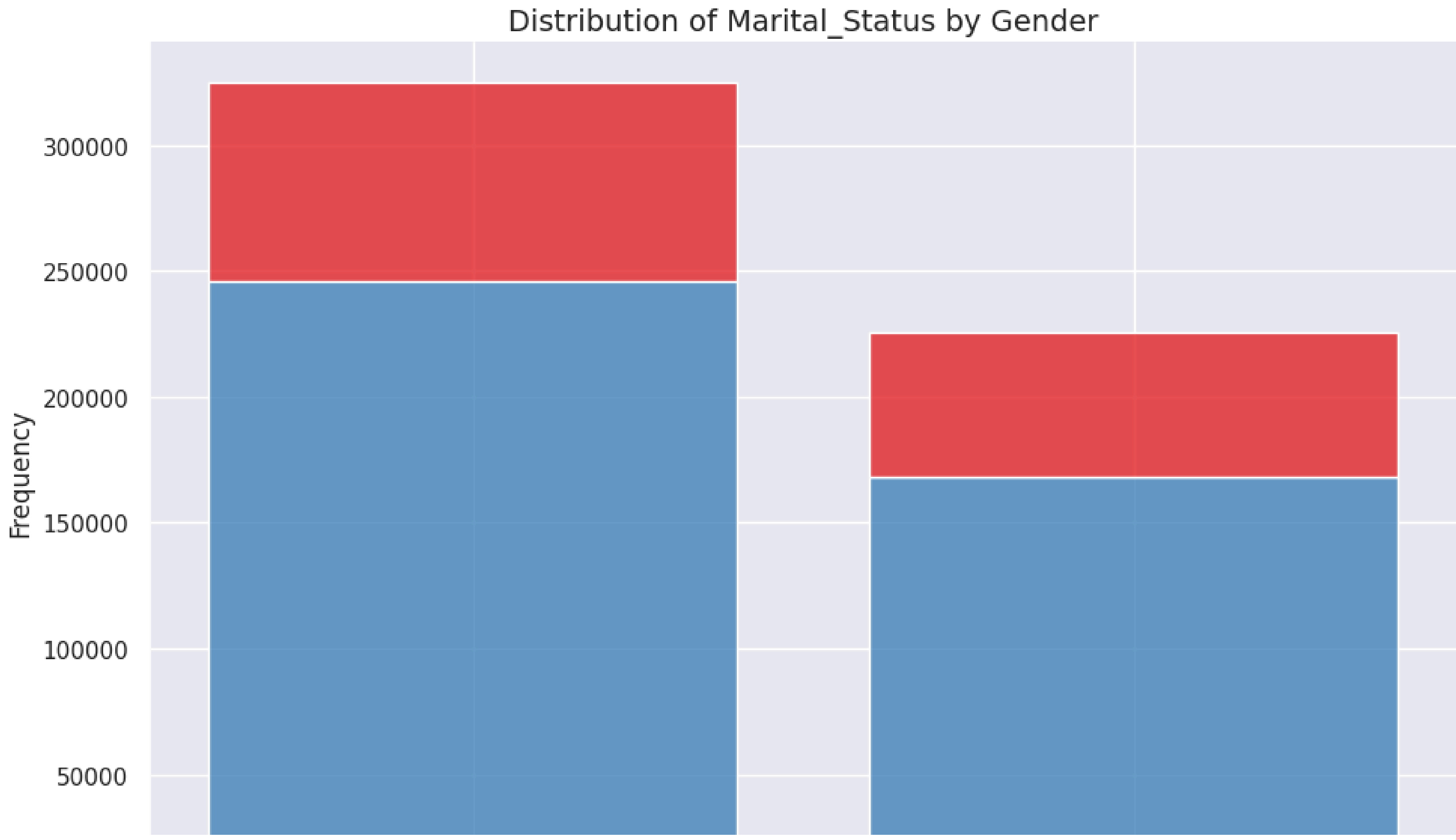

Distribution of Age by Gender
Frequency
200000
150000
100000
50000
0
0-17
18-25
26-35
36-45
46-50
51-55
55+
Age


Distribution of Occupation by Gender
Frequency
70000
60000
50000
40000
30000
20000
10000
0
0
5
10
15
20
Occupation


Distribution of City_Category by Gender
200000
150000


Frequency
100000
50000
0
A
B
C
City_Category


Distribution of Stay_In_Current_City_Years by Gender
200000
175000
150000
125000
100000
75000
50000
25000
0
Frequency
0
1
2
3
4+
Stay_In_Current_City_Years


Distribution of Marital_Status by Gender
300000
250000
200000
150000
100000
50000
Frequency

Insights:

Gender-Related Purchase Analysis:

- Males tend to make more purchases across various age groups compared to females. Among all age brackets, the group between 26 to 35 years old exhibits the most noticeable difference, with males making notably higher purchase counts than females.

Occupation-Related Purchase Analysis:

- Occupations labeled as '0' and '4' stand out with the highest purchase counts, suggesting that individuals in these occupation categories contribute significantly to overall sales. Particularly, occupation '4' shows notably higher purchases compared to others, indicating its significant impact on sales.

City Category-Related Purchase Analysis:

- City_Category 'B' emerges as the top contributor to purchase counts for both genders. This indicates that customers residing in City_Category 'B' play a crucial role in driving overall sales compared to those in 'A' and 'C' categories.

Stay in Current City Duration Impact:

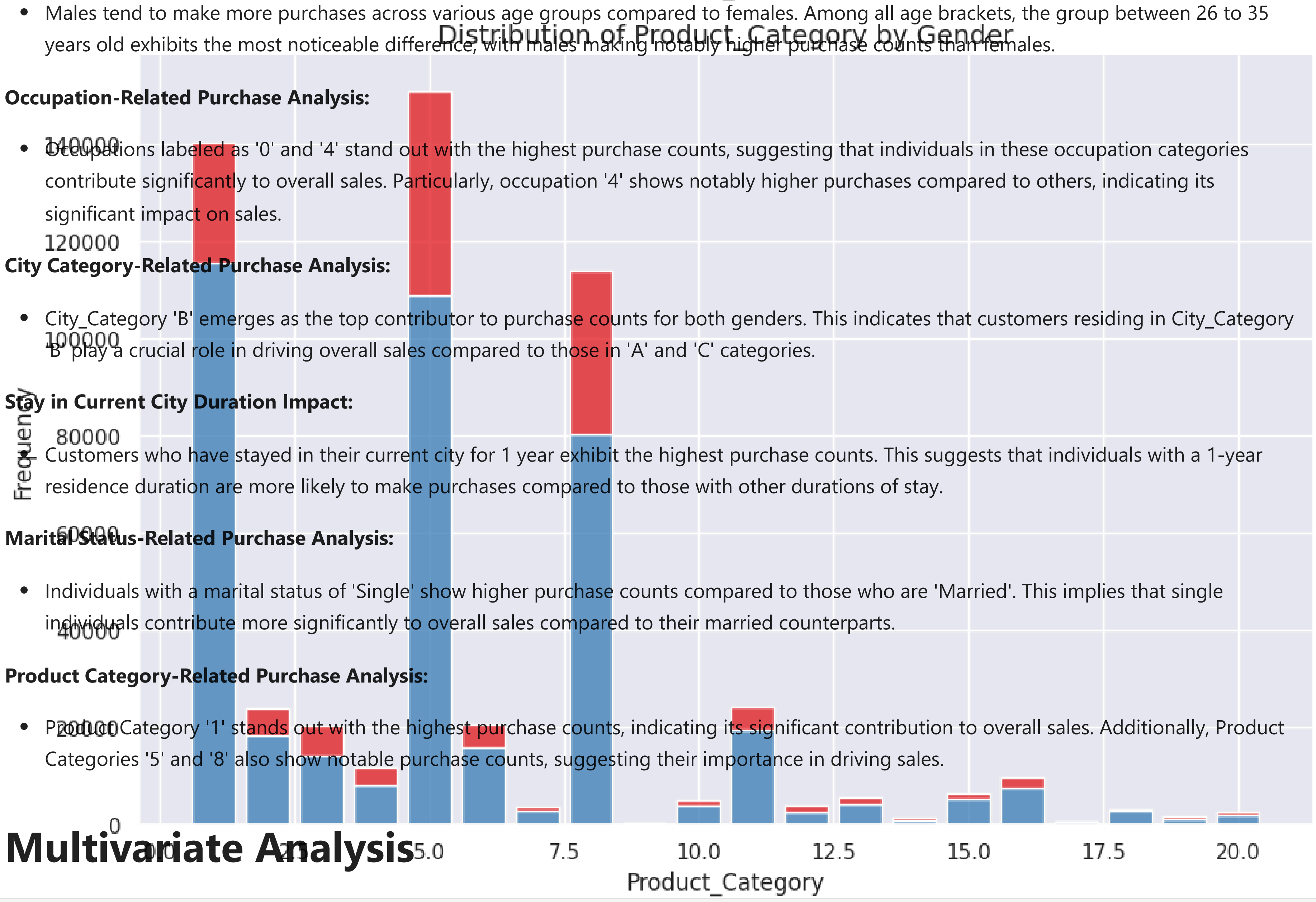
- Customers who have stayed in their current city for 1 year exhibit the highest purchase counts. This suggests that individuals with a 1-year residence duration are more likely to make purchases compared to those with other durations of stay.

Marital Status-Related Purchase Analysis:

- Individuals with a marital status of 'Single' show higher purchase counts compared to those who are 'Married'. This implies that single individuals contribute more significantly to overall sales compared to their married counterparts.

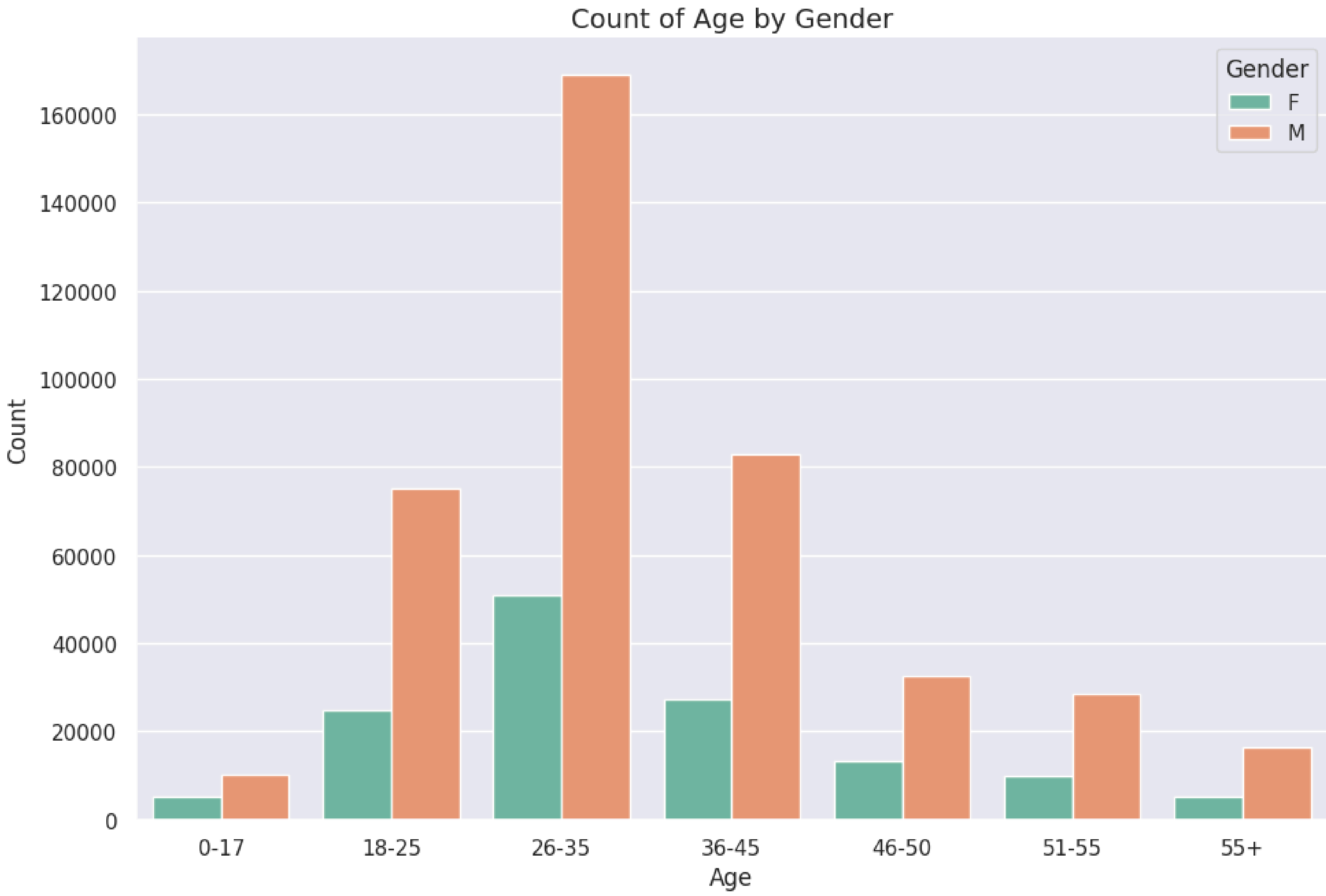
Product Category-Related Purchase Analysis:

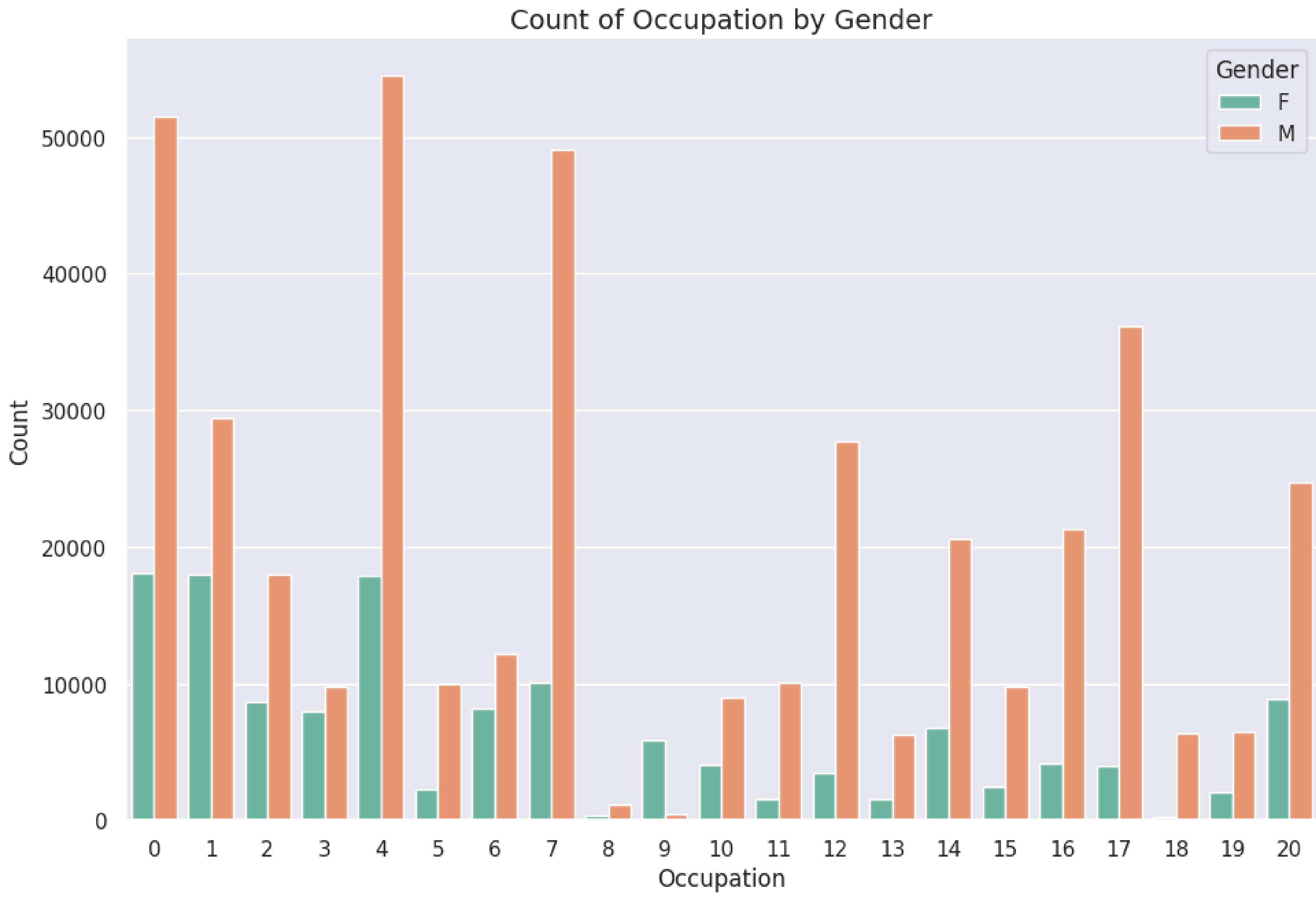
- Product Category '1' stands out with the highest purchase counts, indicating its significant contribution to overall sales. Additionally, Product Categories '5' and '8' also show notable purchase counts, suggesting their importance in driving sales.

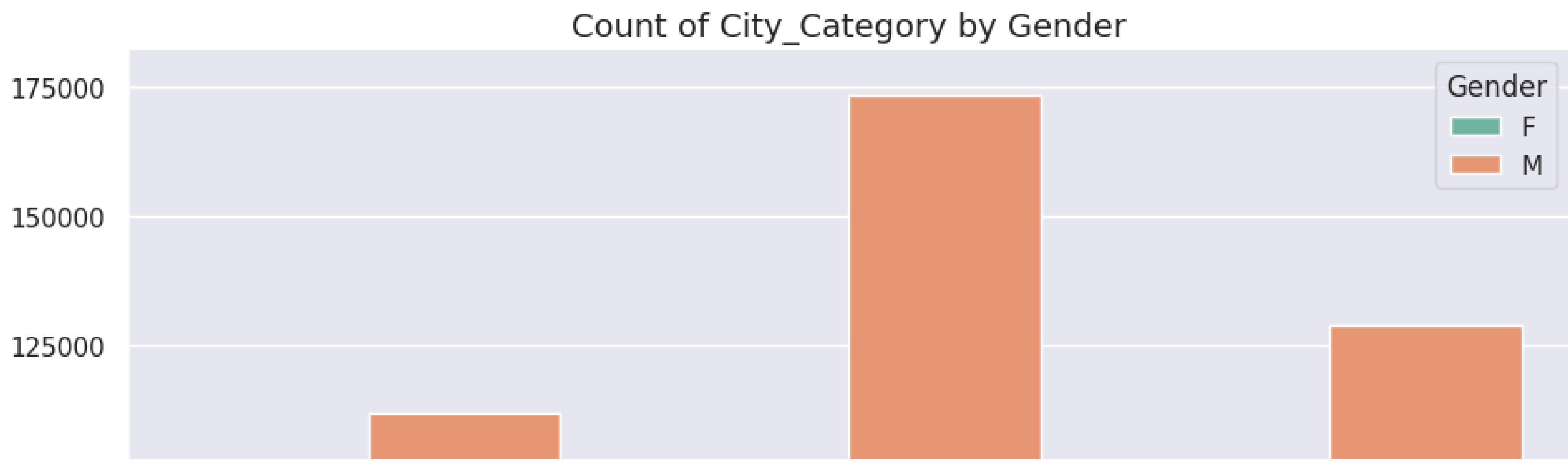


Multivariate Analysis

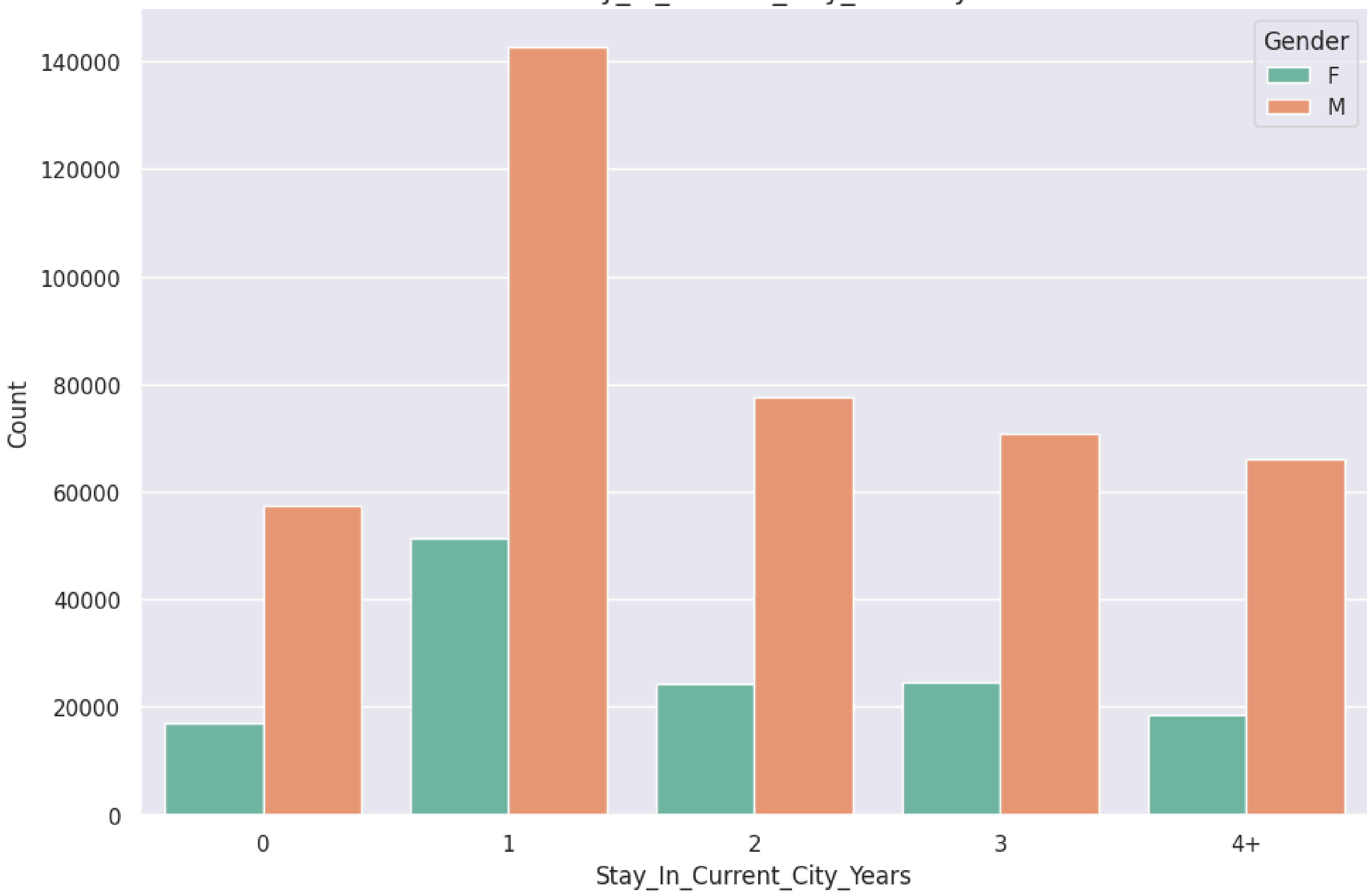
In [54]:
```
plt.figure(figsize=(10, 40))
sns.set(style='darkgrid')

# 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')

    plt.tight_layout()

plt.show()
```


Count of Age by Gender
Gender
F
M
Count
0
20000
40000
60000
80000
100000
120000
140000
160000
0-17
18-25
26-35
36-45
46-50
51-55
55+
Age

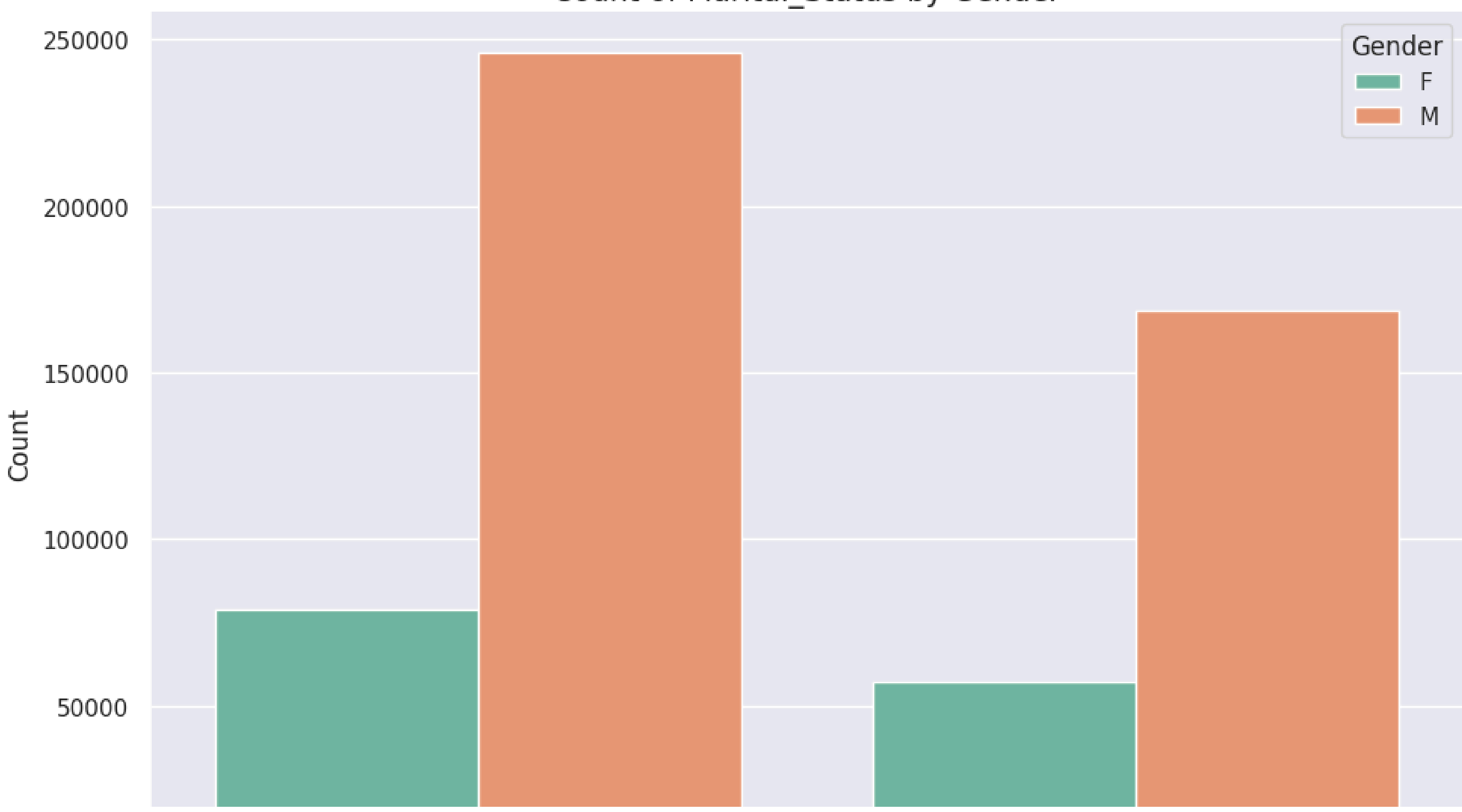

Count of Occupation by Gender
Gender
F
M
Count
0
10000
20000
30000
40000
50000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Occupation


Count of City_Category by Gender
Gender
F
M
125000
150000
175000

Count of Stay_In_Current_City_Years by Gender



Count of Marital_Status by Gender



In [55]:
```
import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)
```

Balck friday Sales analysis on gender

In [56]:
```
avg_purchase = data.groupby('Gender')[['Purchase']].mean().reset_index().round(2)
avg_purchase
```

Out[56]:

	Gender	Purchase
0	F	8734.57
1	M	9437.53

In [57]:
```
data_male = data[data['Gender']=='M']
data_female = data[data['Gender']=='F']
```

In [59]:
```
print(f'Male customers - {len(data_male)}')
print(f'Female customers - {len(data_female)}')
```

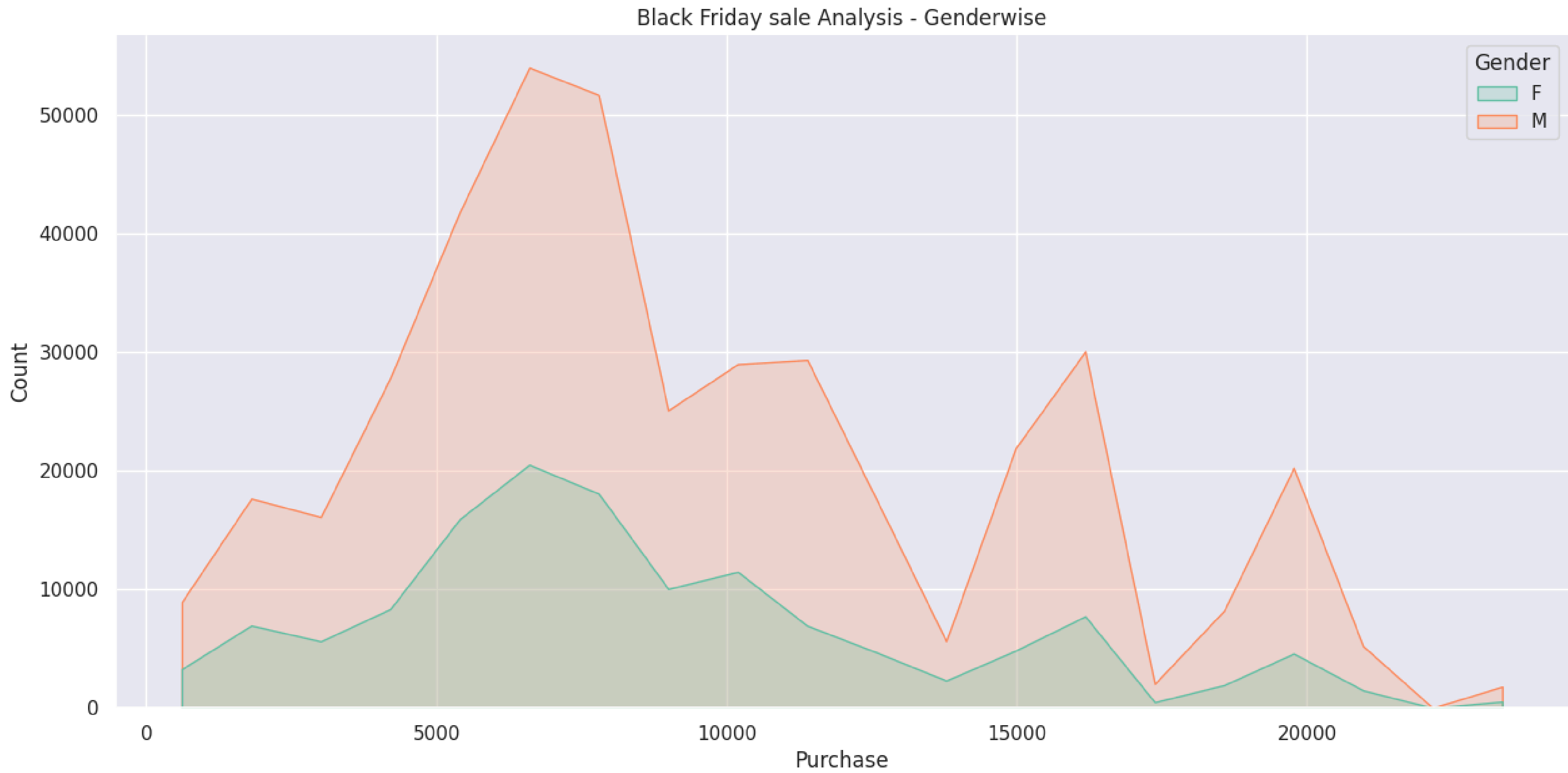
```
Male customers - 414259
Female customers - 135809
```

In [60]:
```
data.groupby('Gender')['Purchase'].describe().T
```

Out[60]:

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

In [61]:
```
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()
```



Insights:

1. During the Black Friday sale, guys tend to splurge more than gals.
2. There are more guys (4225) shopping than gals (1666) in total.
3. On average, guys (averaging at 9437) spend more bucks compared to gals (averaging at 8734).
4. Since there are more guys in the mix, they're likely to rack up more purchases than gals.
5. Maybe the reason guys spend more is because there are more products tailored to their tastes, which makes them open their wallets wider.

In [63]:
```
# Calculates the 95% confidence interval and width for a specified category within a given variable in the dataset.

def data_ci(data, variable, category, confidence_level=0.95):

    category_data = data[data[variable] == category]['Purchase']
    category_mean = category_data.mean()
    category_std = category_data.std()

    # standard error of the mean
    category_sem = category_std / np.sqrt(len(category_data))

    # margin of error
    category_moe = category_sem * norm.ppf((1 + confidence_level) / 2)

    # confidence interval
    category_ci = (category_mean - category_moe, category_mean + category_moe)

    # width
    category_width = category_ci[1] - category_ci[0]

    print(f'{category} 95% confidence interval: {category_ci}')
    print(f'{category} Width: {category_width}')
```

In [64]:
```
# Calculates the 95% confidence interval and width for a specified category within a given variable in a sampled dataset.

def sample_ci(data, variable, category, sample_size):

    category_data = data[data[variable] == category]['Purchase']
    sample_data = category_data.sample(n=sample_size, random_state=42)
    mean_val = sample_data.mean()
    std_dev = sample_data.std()

    # standard error of the mean
    sem = std_dev / np.sqrt(sample_size)

    # margin of error
    moe = sem * norm.ppf((1 + 0.95) / 2)  # 1.96 corresponds to the Z-score for a 95% confidence interval

    # confidence interval
    ci = (mean_val - moe, mean_val + moe)

    category_width = ci[1] - ci[0]

    print(f"\nSample Size: {sample_size}")
    print(f'{category} 95% confidence interval: {ci}')
    print(f'{category} Width: {category_width}')
```

Confidence intervals for the Average amount spent per gender.

95% confidence interval of Entire Dataset

In [66]:
```
data_ci(data, 'Gender', 'M')
```

```
M 95% confidence interval: (9422.01944736257, 9453.032633581959)
M Width: 31.013186219388444
```

In [67]:
```
data_ci(data, 'Gender', 'F')
```

```
F 95% confidence interval: (8709.21154714068, 8759.919983170272)
F Width: 50.70843602959212
```

95% confidence interval of 300 samples

In [68]:
```
sample_ci(data, 'Gender', 'M', 300)
```

```
Sample Size: 300
M 95% confidence interval: (9283.731565877591, 10491.715100789075)
M Width: 1207.9835349114837
```

In [69]:
```
sample_ci(data, 'Gender', 'F', 300)
```

```
Sample Size: 300
F 95% confidence interval: (8308.865304074718, 9426.034695925284)
F Width: 1117.1693918505662
```

In [ ]:

95% confidence interval of 3000 samples

In [71]:
```
sample_ci(data, 'Gender', 'M', 3000)
```

```
Sample Size: 3000
M 95% confidence interval: (9460.10182838994, 9831.170171610062)
M Width: 371.0683432201222
```

In [72]:
```
sample_ci(data, 'Gender', 'F', 3000)
```

```
Sample Size: 3000
F 95% confidence interval: (8630.48138780842, 8982.545945524911)
F Width: 352.0645577164905
```

95% confidence interval of 30000 samples

In [73]:
```
sample_ci(data, 'Gender', 'M', 30000)
```

```
Sample Size: 30000
M 95% confidence interval: (9428.950211018666, 9544.881322314668)
M Width: 115.9311112960022
```

In [74]:
```
sample_ci(data, 'Gender', 'M', 30000)
```

```
Sample Size: 30000
M 95% confidence interval: (9428.950211018666, 9544.881322314668)
M Width: 115.9311112960022
```

Insights:

1. The confidence interval for males is wider than for females, showing that spending by males varies more.
2. Bigger sample sizes make the confidence interval narrower, giving more accurate estimates.
3. Overlapping confidence intervals for different sample sizes hint that the observed spending differences may not be statistically big.
4. With larger sample sizes, the distribution of means becomes more like a normal curve, as predicted by the Central Limit Theorem.

Confidence intervals for the average amount spent per Marital_Status.

95% confidence interval of Entire Dataset

In [75]:
```
data_ci(data, 'Marital_Status', 'Married')
```

```
Married 95% confidence interval: (9240.460427057078, 9281.888721107669)
Married Width: 41.42829405059092
```

In [76]:
```
data_ci(data, 'Marital_Status', 'Single')
```

```
Single 95% confidence interval: (9248.61641818668, 9283.198819656332)
Single Width: 34.58240146965181
```

95% confidence interval of 300 samples

In [77]:
```
sample_ci(data, 'Marital_Status', 'Married', 300)
```

```
Sample Size: 300
Married 95% confidence interval: (8887.305881933493, 10041.72745139984)
Married Width: 1154.4215694663471
```

In [78]:
```
sample_ci(data, 'Marital_Status', 'Single', 300)
```

```
Sample Size: 300
Single 95% confidence interval: (9051.928693931213, 10213.504639402121)
Single Width: 1161.5759454709078
```

95% confidence interval of 3000 samples

In [80]:
```
sample_ci(data, 'Marital_Status', 'Married', 3000)
```

```
Sample Size: 3000
Married 95% confidence interval: (9118.562018709765, 9482.974647956902)
Married Width: 364.4126292471374
```

In [81]:
```
sample_ci(data, 'Marital_Status', 'Single', 3000)
```

```
Sample Size: 3000
Single 95% confidence interval: (9246.175079645862, 9612.375587020804)
Single Width: 366.2005073749424
```

95% confidence interval of 30000 samples

In [82]:
```
sample_ci(data, 'Marital_Status', 'Married', 30000)
```

```
Sample Size: 30000
Married 95% confidence interval: (9198.156166015178, 9312.029900651487)
Married Width: 113.87373463630865
```

In [83]:
```
sample_ci(data, 'Marital_Status', 'Single', 30000)
```

```
Sample Size: 30000
Single 95% confidence interval: (9229.816006946752, 9343.573126386582)
Single Width: 113.7571194398297
```

Insights:

1. The confidence interval for the 'Married' group is broader compared to the 'Single' group, signaling more spending variability among married individuals.

2. With larger sample sizes, the confidence interval tightens up, showing how precision improves with more data.
3. The overlapping confidence intervals for 'Married' and 'Single' groups imply that the spending differences observed may not be statistically significant across different sample sizes.
4. As sample sizes grow, confidence intervals narrow down, providing a sharper estimate of the average and leading to distributions of sample means that resemble a normal curve.

Confidence intervals for the average amount spent per City_Category

95% confidence interval of Entire Dataset

In [84]:
```
data_ci(data, 'City_Category', 'A')
```
```
A 95% confidence interval: (8886.991825864907, 8936.88660630406)
A Width: 49.89478043915369
```

In [85]:
```
data_ci(data, 'City_Category', 'B')
```
```
B 95% confidence interval: (9131.099848963764, 9171.501276600207)
B Width: 40.40142763644326
```

In [86]:
```
data_ci(data, 'City_Category', 'C')
```
```
C 95% confidence interval: (9695.337107885243, 9744.504878386117)
C Width: 49.1677705008733
```

95% confidence interval of 300 samples

In [87]:
```
sample_ci(data, 'City_Category', 'A', 300)
```
```
Sample Size: 300
A 95% confidence interval: (8098.995845827299, 9266.9641541727)
A Width: 1167.968308345401
```

In [89]:
```
sample_ci(data, 'City_Category', 'B', 300)
```
```
Sample Size: 300
B 95% confidence interval: (8571.45829896875, 9684.755034364583)
B Width: 1113.2967353958338
```

In [90]:
```
sample_ci(data, 'City_Category', 'C', 300)
```
```
Sample Size: 300
C 95% confidence interval: (8630.994793994194, 9728.831872672474)
C Width: 1097.8370786782798
```

95% confidence interval of 3000 samples

In [91]:
```
sample_ci(data, 'City_Category', 'A', 3000)
```
```
Sample Size: 3000
A 95% confidence interval: (8812.739396324683, 9167.82993700865)
A Width: 355.09054068396654
```

In [92]:
```
sample_ci(data, 'City_Category', 'B', 3000)
```
```
Sample Size: 3000
B 95% confidence interval: (8791.70616073309, 9141.478505933577)
B Width: 349.7723452004866
```

In [93]:
```
sample_ci(data, 'City_Category', 'C', 3000)
```
```
Sample Size: 3000
C 95% confidence interval: (9442.853994951975, 9813.490671714693)
C Width: 370.6366767627187
```

95% confidence interval of 30000 samples

In [94]:
```
sample_ci(data, 'City_Category', 'A', 30000)
```
```
Sample Size: 30000
A 95% confidence interval: (8836.46007218682, 8947.056727813182)
A Width: 110.59665562636292
```

In [95]:
```
sample_ci(data, 'City_Category', 'B', 30000)
```
```
Sample Size: 30000
B 95% confidence interval: (9079.173984592268, 9191.2066820744)
B Width: 112.03269748213279
```

In [96]:
```
sample_ci(data, 'City_Category', 'C', 30000)
```
```
Sample Size: 30000
C 95% confidence interval: (9656.973563549582, 9774.566303117084)
C Width: 117.59273956750258
```

Insights:

1. The confidence interval for City Category C is broader compared to others, indicating more spending variability within this category across the entire dataset.
2. As sample sizes grow, the confidence intervals become narrower. This trend is visible across all city categories (A, B, C) as sample sizes increase.
3. The overlapping confidence intervals for different sample sizes suggest that there are no significant mean differences between these sample sizes.
4. With larger sample sizes, confidence intervals tighten, offering a more accurate mean estimate and leading to distributions of means that adhere closely to the normal distribution as predicted by the Central Limit Theorem.

Business Recommendations:

1. **Targeted Marketing for Age Group '26-35':**
   - Direct marketing efforts towards individuals aged between 26 to 35, as they show the highest purchasing activity. Customize promotions and ads to resonate with this age group's interests and preferences.
2. **Occupation-Based Product Offerings:**
   - Tailor product offerings or promotions to suit the needs of individuals in Occupation '4', considering its significant representation and high purchase levels.
3. **Strategic City_Category 'B' Promotions:**
   - Concentrate promotional efforts in City_Category 'B', where the highest purchases occur. Craft promotions that appeal to the preferences of customers in this category.
4. **Targeted Campaigns for Singles:**
   - Launch marketing campaigns tailored to singles, as they contribute substantially more to overall sales. Understand their preferences and tailor marketing messages to resonate with this group.
5. **Encourage Long-Term Residency:**
   - Develop strategies to incentivize customers to stay in their current city for longer durations, such as loyalty programs or special perks for long-term residents, to boost their purchasing behavior.
6. **Product Category Optimization:**
   - Optimize inventory and promotions for categories '1' and '5', which exhibit higher purchase amounts. Strategically manage these categories to maximize overall sales revenue.
7. **Gender-Targeted Marketing Strategies:**
   - Implement marketing strategies targeted towards males across different age groups. Utilize insights from age-based gender analysis to effectively tailor promotions and advertisements.
8. **Occupation-Driven Promotions:**
   - Design promotions or incentives based on the top occupations, particularly '0' and '4', to further drive sales from these occupational groups.
9. **City_Category 'B' Specific Initiatives:**
   - Introduce specific initiatives, offers, or events in City_Category 'B' to capitalize on its higher purchasing behavior and engage customers more effectively.
10. **Data-Driven Product Development:**
    - Analyze the preferences of male customers to inform product development decisions. Ensure that the product range aligns with the preferences of the larger male customer base to boost sales.

In [ ]: