In [9]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as ny
```

In [1]:
```python
pip install pymysql sqlalchemy pandas
```

```
Requirement already satisfied: pymysql in c:\users\admin\appdata\local\progr
ams\python\python313\lib\site-packages (1.1.1)
Requirement already satisfied: sqlalchemy in c:\users\admin\appdata\local\pr
ograms\python\python313\lib\site-packages (2.0.41)
Requirement already satisfied: pandas in c:\users\admin\appdata\local\progra
ms\python\python313\lib\site-packages (2.3.0)
Requirement already satisfied: greenlet>=1 in c:\users\admin\appdata\local\p
rograms\python\python313\lib\site-packages (from sqlalchemy) (3.2.3)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\users\admin\ap
pdata\local\programs\python\python313\lib\site-packages (from sqlalchemy)
(4.14.0)
Requirement already satisfied: numpy>=1.26.0 in c:\users\admin\appdata\local
\programs\python\python313\lib\site-packages (from pandas) (2.3.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\admin\appd
ata\local\programs\python\python313\lib\site-packages (from pandas) (2.9.0.p
ost0)
Requirement already satisfied: pytz>=2020.1 in c:\users\admin\appdata\local
\programs\python\python313\lib\site-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\admin\appdata\loca
l\programs\python\python313\lib\site-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\admin\appdata\local\prog
rams\python\python313\lib\site-packages (from python-dateutil>=2.8.2->panda
s) (1.17.0)
Note: you may need to restart the kernel to use updated packages.
```

In [1]:
```python
from sqlalchemy import create_engine
import pandas as pd

user = "root"
password = "Aryan%402004"  # '@' becomes '%40'
host = "localhost"
port = 3306
database = "banking_case"  # replace with your actual DB name

engine = create_engine(f"mysql+pymysql://{user}:{password}@{host}:{port}/{da

# Test query
df = pd.read_sql("SHOW TABLES", engine)
print(df)
```

```
  Tables_in_banking_case
0               customer
```

In [3]:
```python
query= "SELECT * FROM banking_case.customer"
```

In [6]:
```python
df= pd.read_sql(query,engine)
df.head(5)
```

Out[6]:

	ï»¿Client ID	Name	Age	Location ID	Joined Bank	Banking Contact	Nationality	Occupatio
0	IND81288	Raymond Mills	24	34324	06-05-2019	Anthony Torres	American	Safet Technicia I'
1	IND65833	Julia Spencer	23	42205	10-12-2001	Jonathan Hawkins	African	Softwar Consultar
2	IND47499	Stephen Murray	27	7314	25-01-2010	Anthony Berry	European	Help Des Operatc
3	IND72498	Virginia Garza	40	34594	28-03-2019	Steve Diaz	American	Geologist
4	IND60181	Melissa Sanders	46	41269	20-07-2012	Shawn Long	American	Assistar Professc

5 rows × 25 columns

In [7]:

```
#Generate Descriptive statistics for the database
df.describe()
```

Out[7]:

	Age	Location ID	Estimated Income	Superannuation Savings	Amount o Credit Cards
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.00000
mean	51.039667	21563.323000	171305.034263	25531.599673	1.46366
std	19.854760	12462.273017	111935.808209	16259.950770	0.67638
min	17.000000	12.000000	15919.480000	1482.030000	1.00000
25%	34.000000	10803.500000	82906.595000	12513.775000	1.00000
50%	51.000000	21129.500000	142313.480000	22357.355000	1.00000
75%	69.000000	32054.500000	242290.305000	35464.740000	2.00000
max	85.000000	43369.000000	522330.260000	75963.900000	3.00000

In [8]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 25 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   ï»¿Client ID              3000 non-null   object
 1   Name                      3000 non-null   object
 2   Age                       3000 non-null   int64
 3   Location ID               3000 non-null   int64
 4   Joined Bank               3000 non-null   object
 5   Banking Contact           3000 non-null   object
 6   Nationality               3000 non-null   object
 7   Occupation                3000 non-null   object
 8   Fee Structure             3000 non-null   object
 9   Loyalty Classification    3000 non-null   object
 10  Estimated Income          3000 non-null   float64
 11  Superannuation Savings    3000 non-null   float64
 12  Amount of Credit Cards    3000 non-null   int64
 13  Credit Card Balance       3000 non-null   float64
 14  Bank Loans                3000 non-null   float64
 15  Bank Deposits             3000 non-null   float64
 16  Checking Accounts         3000 non-null   float64
 17  Saving Accounts           3000 non-null   float64
 18  Foreign Currency Account  3000 non-null   float64
 19  Business Lending          3000 non-null   float64
 20  Properties Owned          3000 non-null   int64
 21  Risk Weighting            3000 non-null   int64
 22  BRId                      3000 non-null   int64
 23  GenderId                  3000 non-null   int64
 24  IAId                      3000 non-null   int64
dtypes: float64(9), int64(8), object(8)
memory usage: 586.1+ KB
```

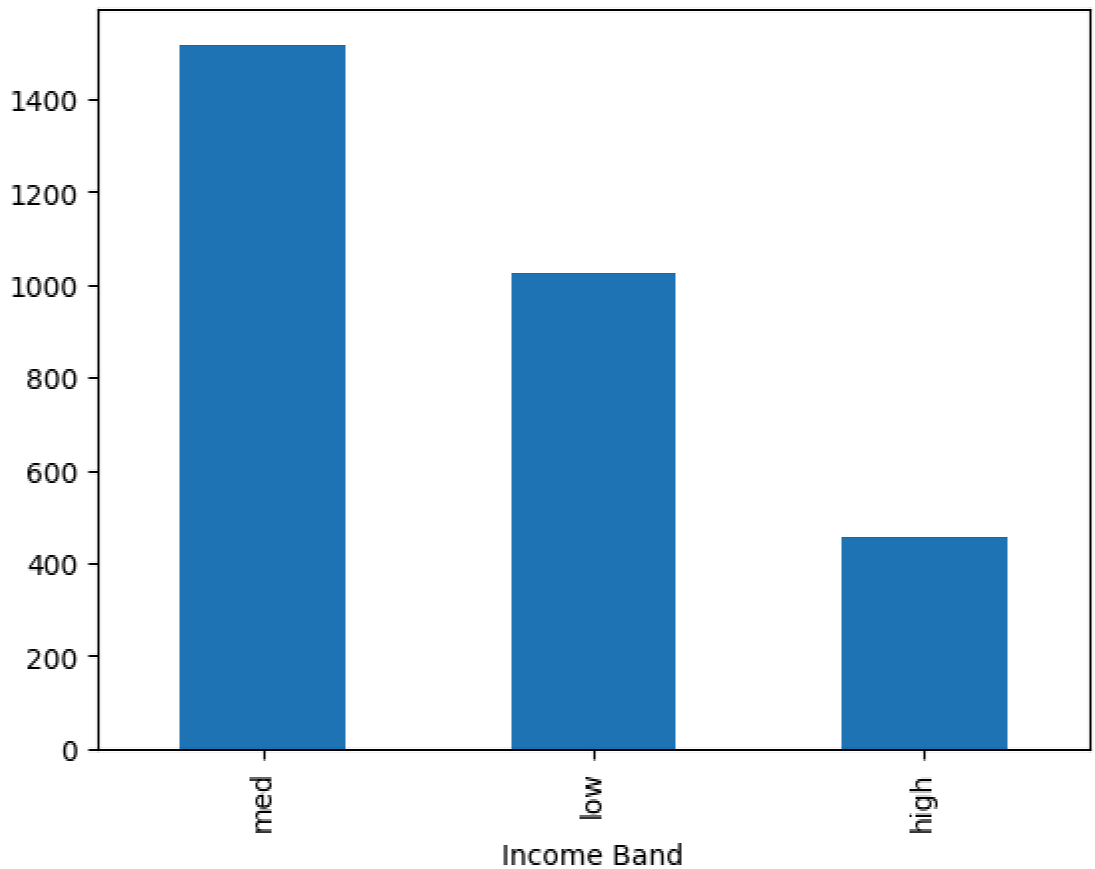
In [10]:
```
df.shape
```

Out[10]: (3000, 25)

In [12]:
```
bins=[0,100000,300000, float('inf')]
labels=['low', 'med', 'high']

df['Income Band'] =pd.cut(df['Estimated Income'], bins=bins, labels=labels,
```

In [13]:
```
df['Income Band'].value_counts().plot(kind='bar')
```

Out[13]: <Axes: xlabel='Income Band'>



In [28]:
```
#Examine the distribution of unique categories in categorical columns
categorical_cols = df[["BRId", "GenderId","IAId","Amount of Credit Cards","N
for col in categorical_cols:
    print(f"Value Counts for '{col}':")
    display(df[col].value_counts())
```

```
Value Counts for 'BRId':
BRId
3    1352
1     660
2     495
4     493
Name: count, dtype: int64
Value Counts for 'GenderId':
GenderId
2    1512
1    1488
Name: count, dtype: int64
Value Counts for 'IAId':
```

```
IAId
1     177
2     177
3     177
4     177
8     177
9     176
13    176
12    176
10    176
11    176
14    176
15    176
6      89
5      89
7      89
16     88
17     88
18     88
19     88
20     88
21     88
22     88
Name: count, dtype: int64
Value Counts for 'Amount of Credit Cards':
Amount of Credit Cards
1    1922
2     765
3     313
Name: count, dtype: int64
Value Counts for 'Nationality':
Nationality
European      1309
Asian          754
American       507
Australian     254
African        176
Name: count, dtype: int64
Value Counts for 'Occupation':
Occupation
Associate Professor             28
Structural Analysis Engineer    28
Recruiter                       25
Account Coordinator             24
Human Resources Manager         24
                                ..
Office Assistant IV              8
Automation Specialist I          7
Computer Systems Analyst I       6
Developer III                    5
Senior Sales Associate           4
Name: count, Length: 195, dtype: int64
Value Counts for 'Fee Structure':
```

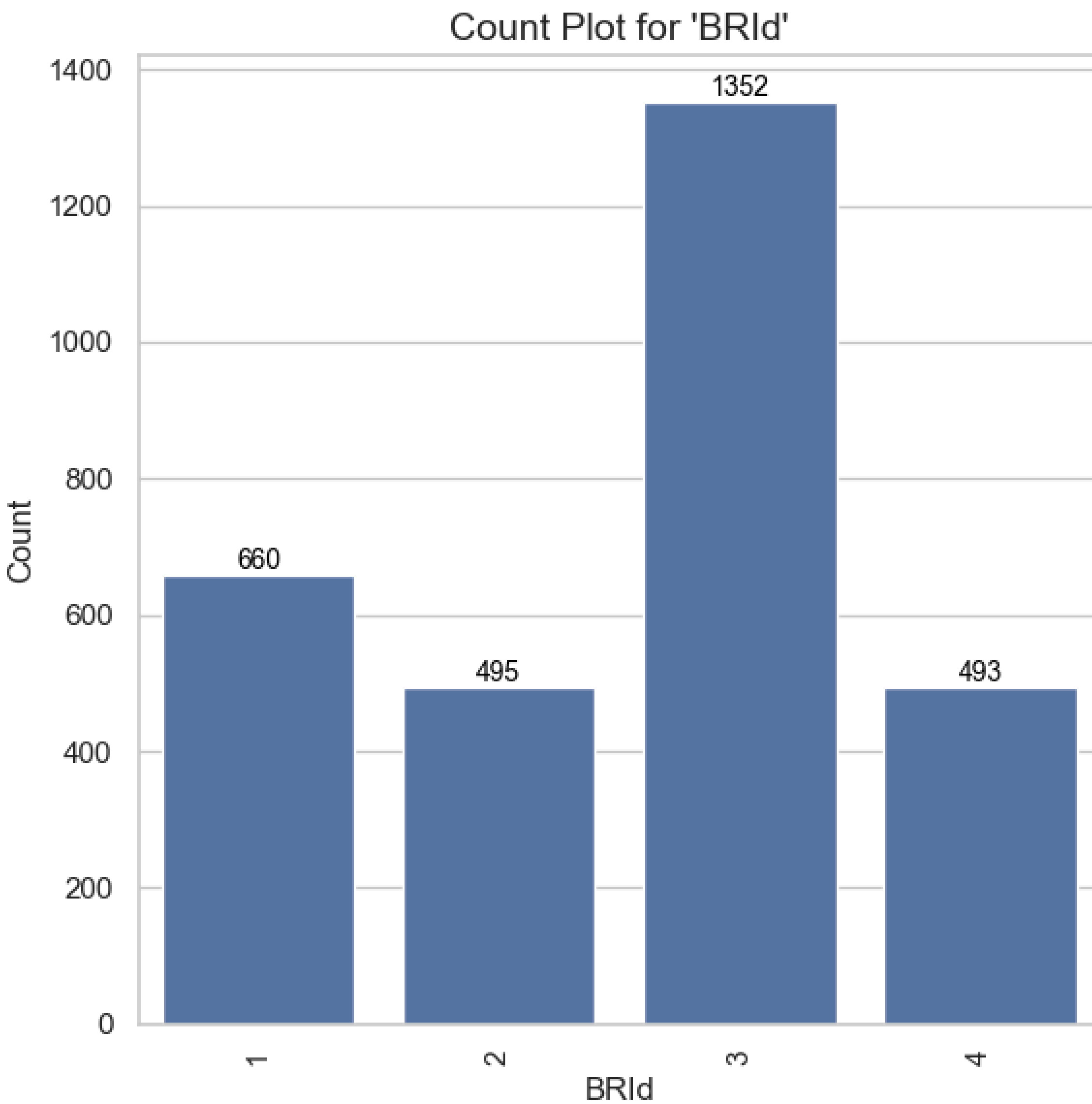
```
Fee Structure
High    1476
Mid      962
Low      562
Name: count, dtype: int64
Value Counts for 'Loyalty Classification':
Loyalty Classification
Jade        1331
Silver       767
Gold         585
Platinum     317
Name: count, dtype: int64
Value Counts for 'Properties Owned':
Properties Owned
2    777
1    776
3    742
0    705
Name: count, dtype: int64
Value Counts for 'Risk Weighting':
Risk Weighting
2    1222
1     836
3     460
4     322
5     160
Name: count, dtype: int64
Value Counts for 'Income Band':
Income Band
med     1517
low     1027
high     456
Name: count, dtype: int64
```

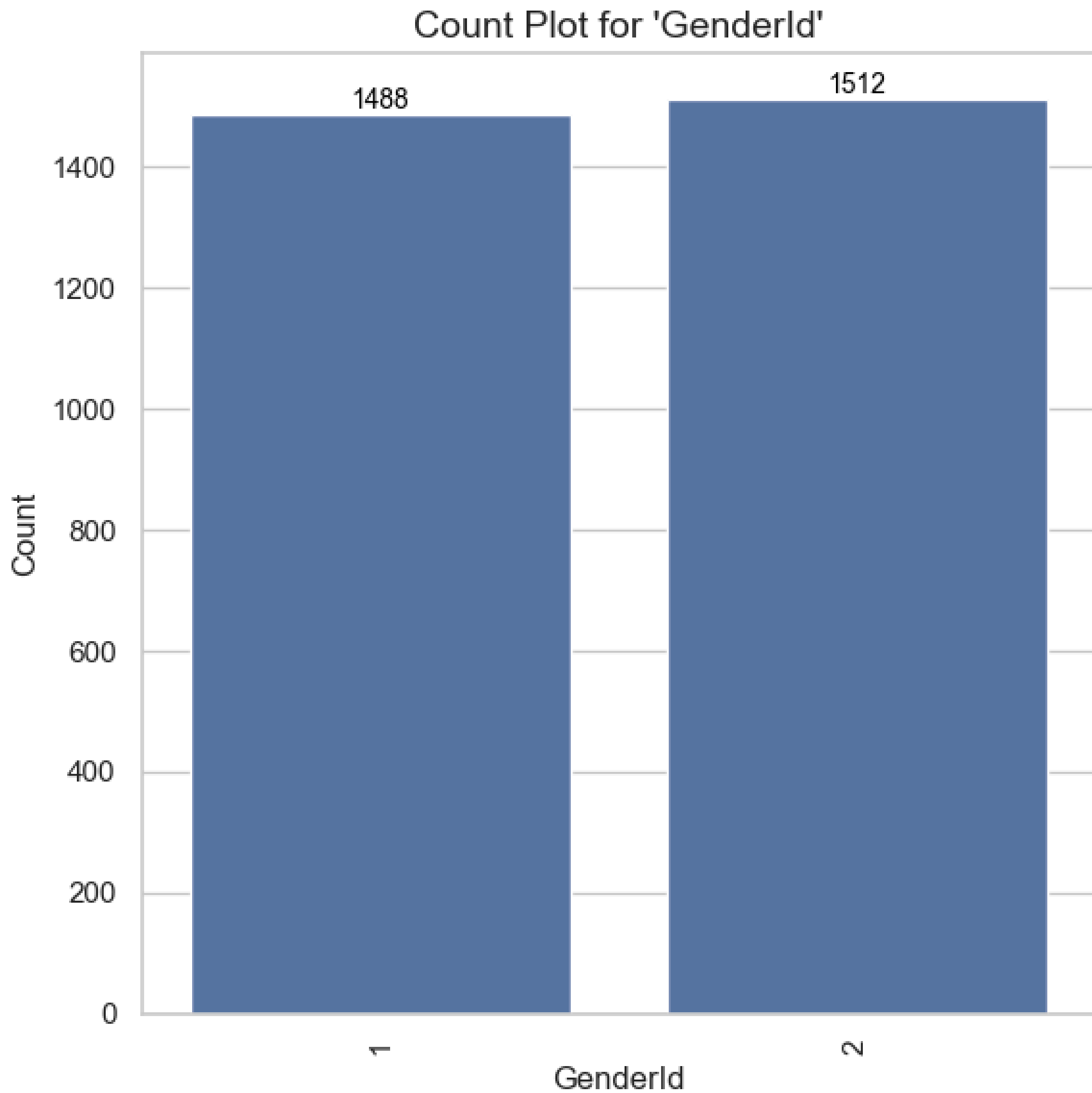
Univariate Analysis

In [29]:

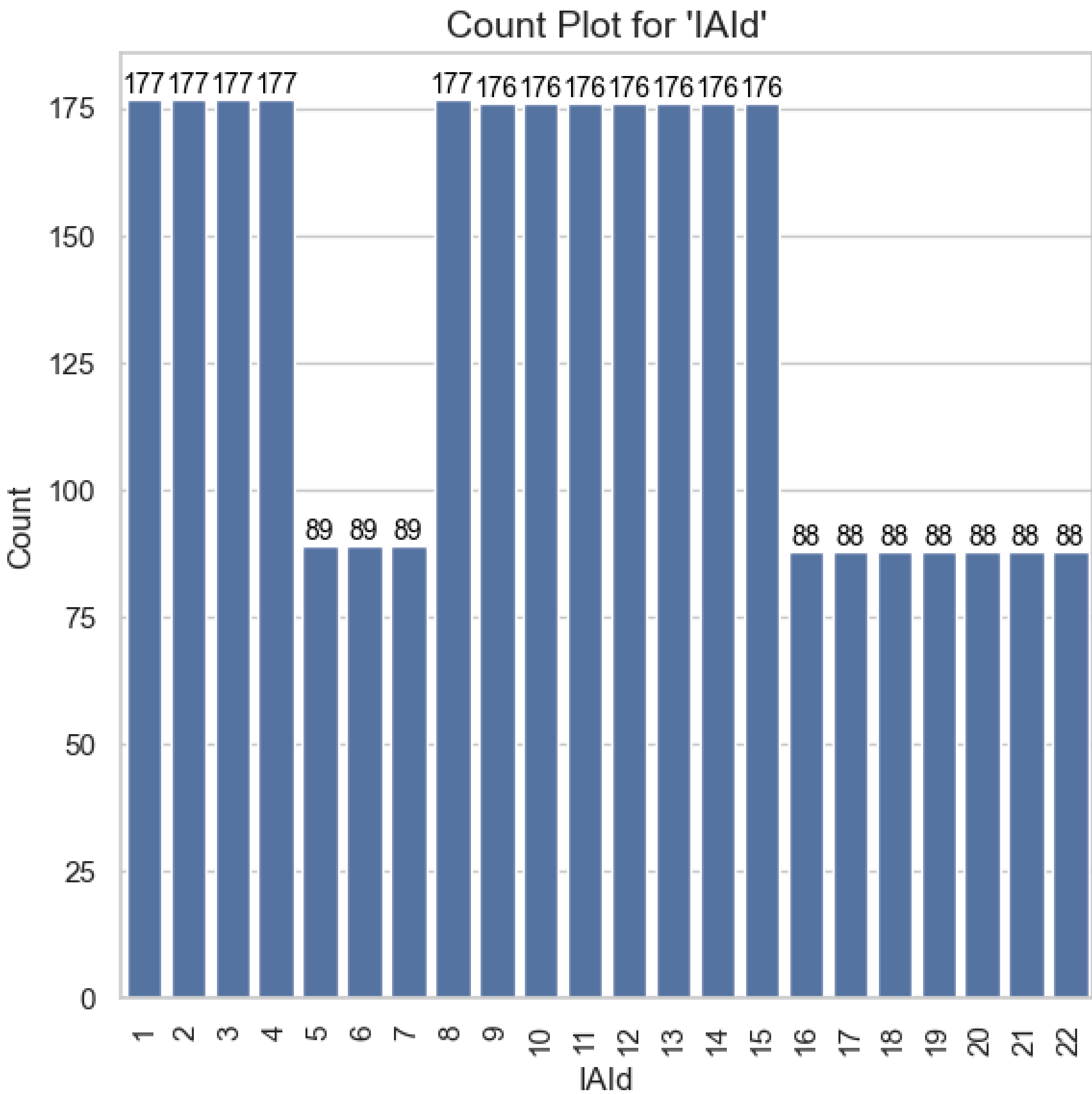
```
sns.set(style="whitegrid")

for col in categorical_cols:
    plt.figure(figsize=(6, 6))

    # Try to sort the categories numerically if possible
    try:
        categories = sorted(df[col].dropna().unique(), key=lambda x: int(x))
    except:
        categories = df[col].value_counts().index

    # Draw countplot
    ax = sns.countplot(data=df, x=col, order=categories, color="#4c72b0")

    # Add value labels on top of bars
    for p in ax.patches:
        height = int(p.get_height())
        ax.annotate(f'{height}', (p.get_x() + p.get_width() / 2., height),
```

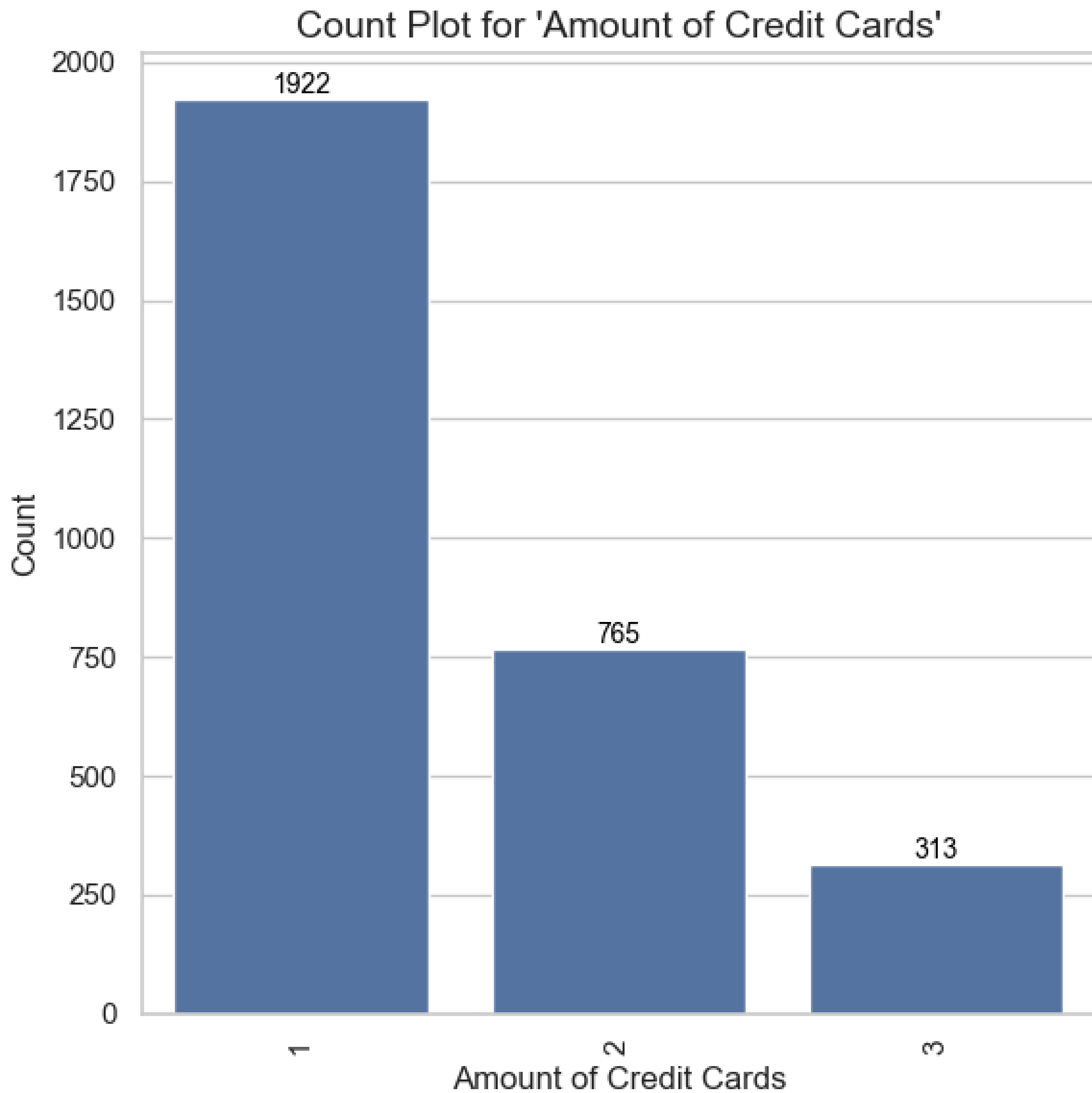
```
                ha='center', va='bottom', fontsize=10, color='black')

    # Final formatting
    plt.title(f"Count Plot for '{col}'", fontsize=14)
    plt.xlabel(col, fontsize=12)
    plt.ylabel("Count", fontsize=12)
    plt.xticks(rotation=90)
    plt.tight_layout()
    plt.show()
```

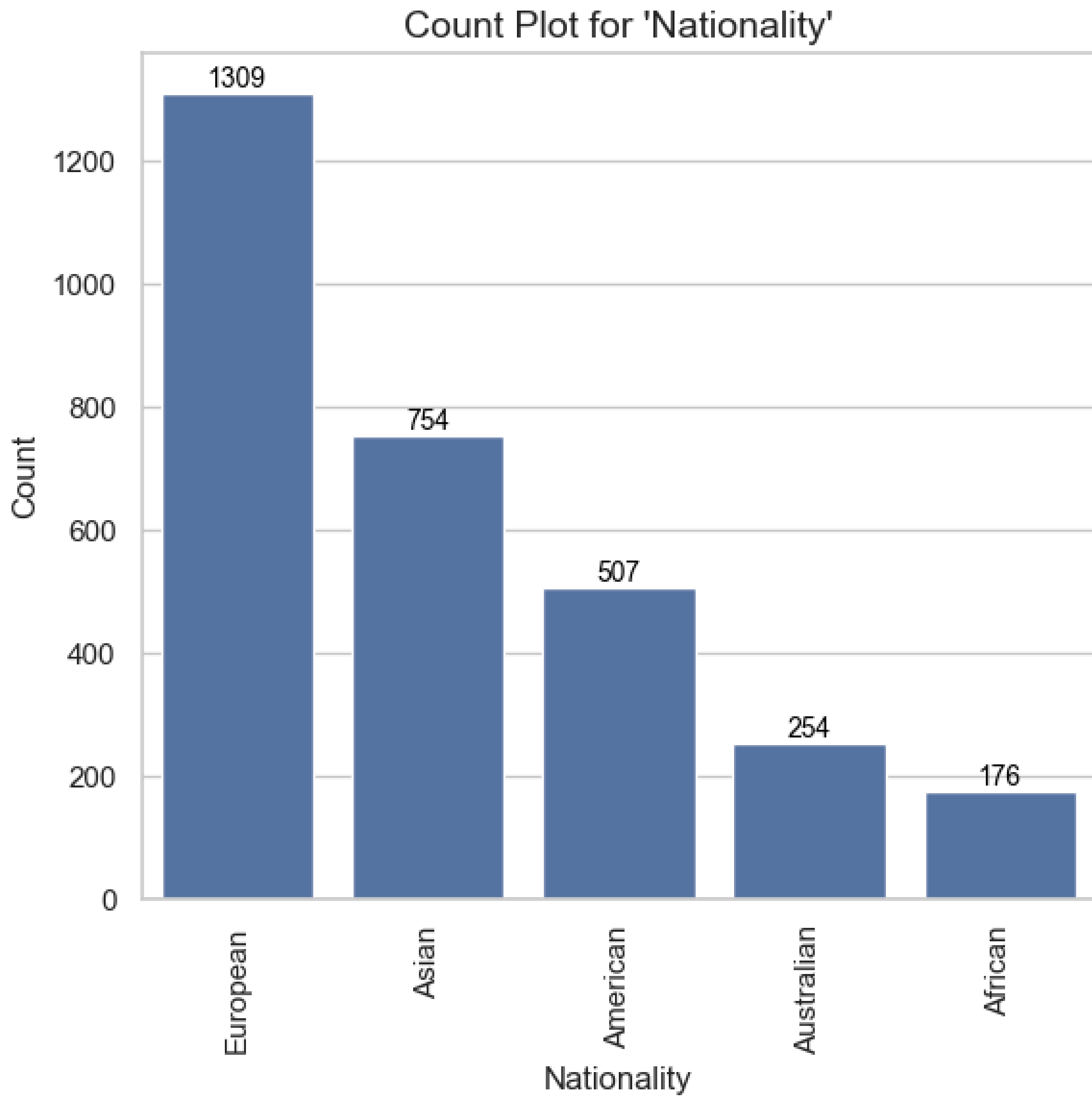




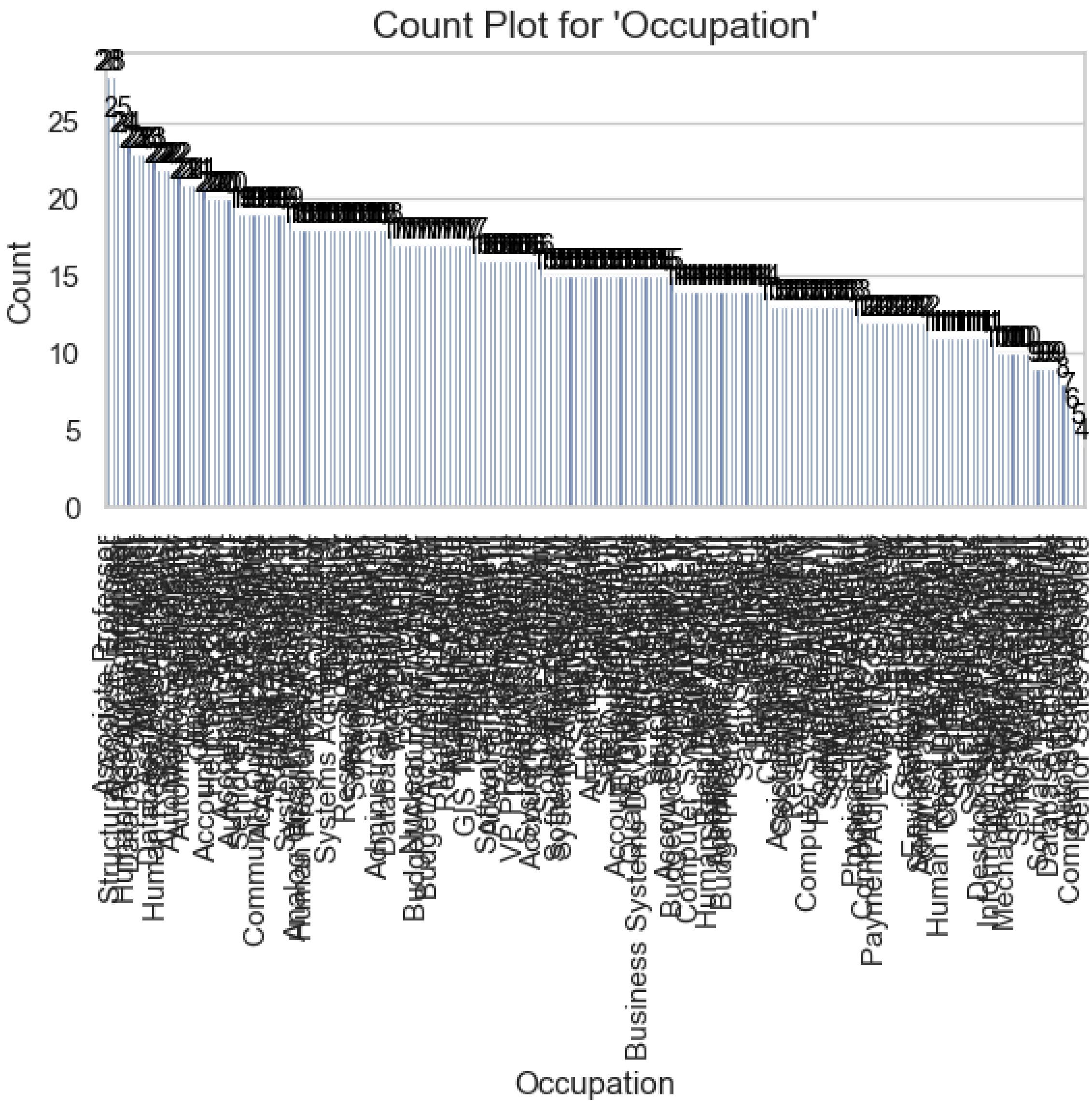
Count Plot for 'GenderId'
1488
1512
1400
1200
1000
800
600
400
200
0
Count
1
2
GenderId

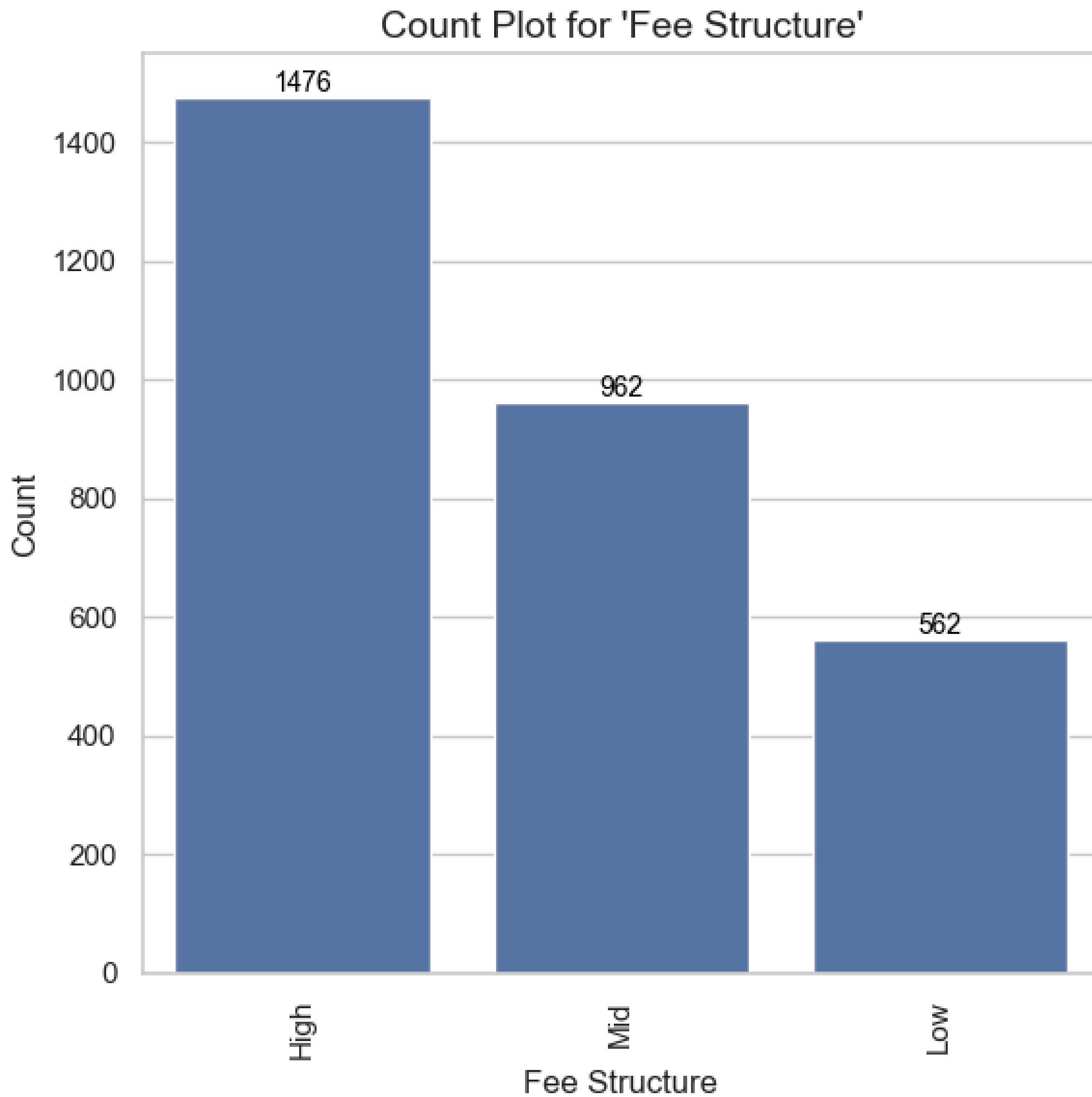

Count Plot for 'IAId'
Count
IAId
175
150
125
100
75
50
25
0
177 177 177 177 89 89 89 177 176 176 176 176 176 176 176 88 88 88 88 88 88 88
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

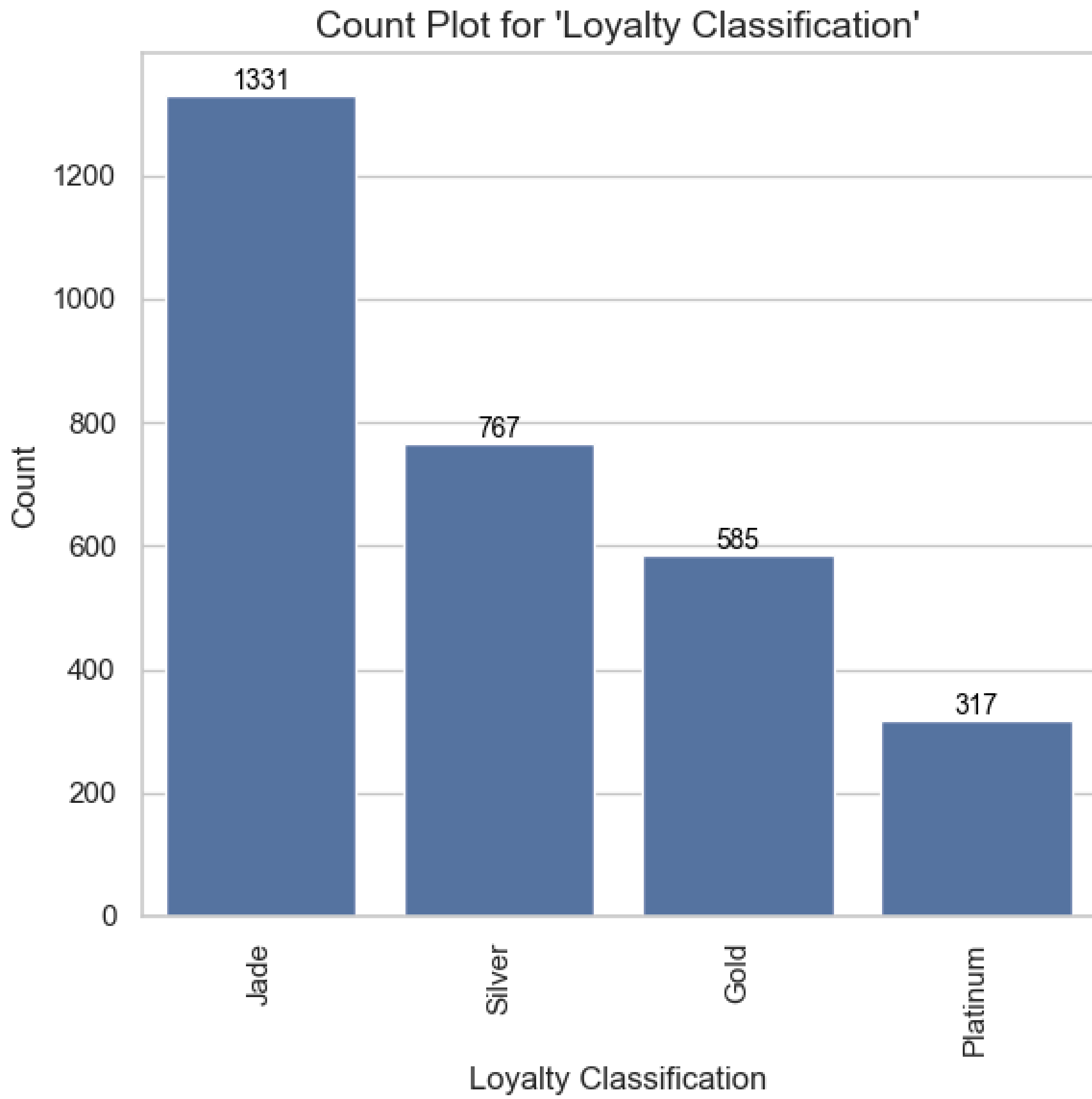

Count Plot for 'Amount of Credit Cards'
Count
2000
1750
1500
1250
1000
750
500
250
0
1922
765
313
1
2
3
Amount of Credit Cards

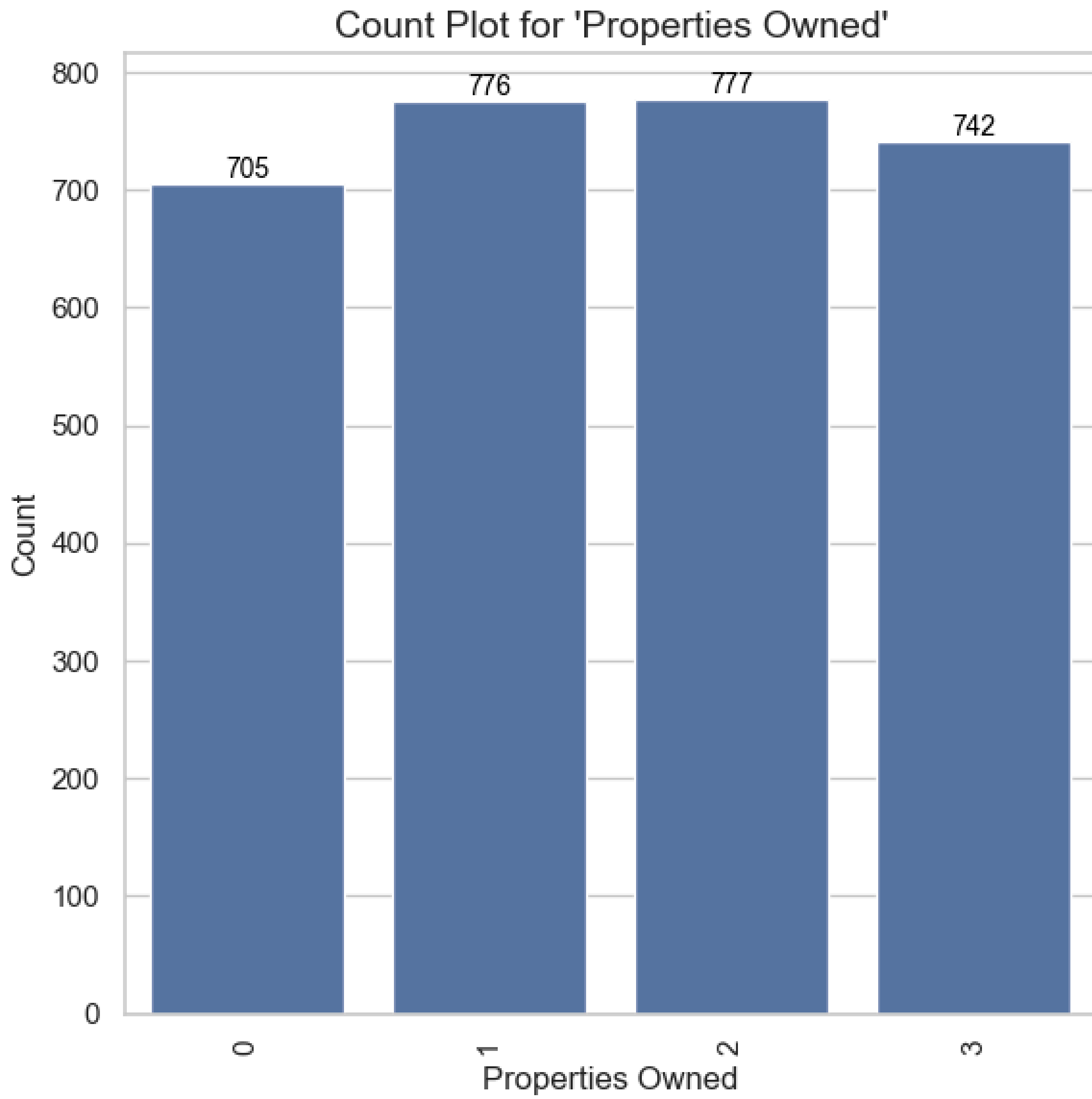


Count Plot for 'Nationality'
1309
754
507
254
176
1200
1000
800
600
400
200
0
Count
European
Asian
American
Australian
African
Nationality

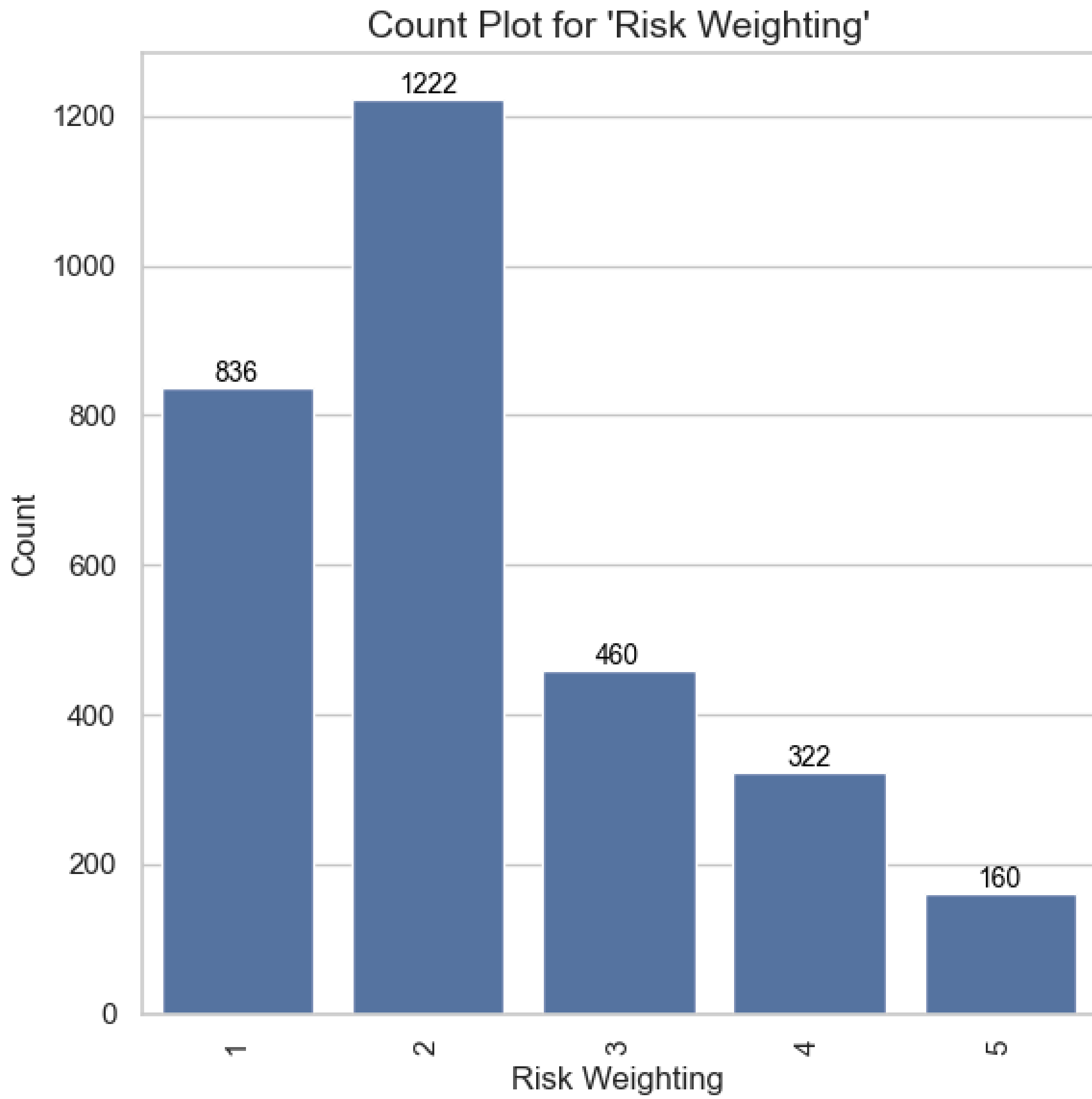

Count Plot for 'Occupation'
Count
0
5
10
15
20
25
Occupation


Count Plot for 'Fee Structure'
1476
962
562
1400
1200
1000
800
600
400
200
0
Count
High
Mid
Low
Fee Structure

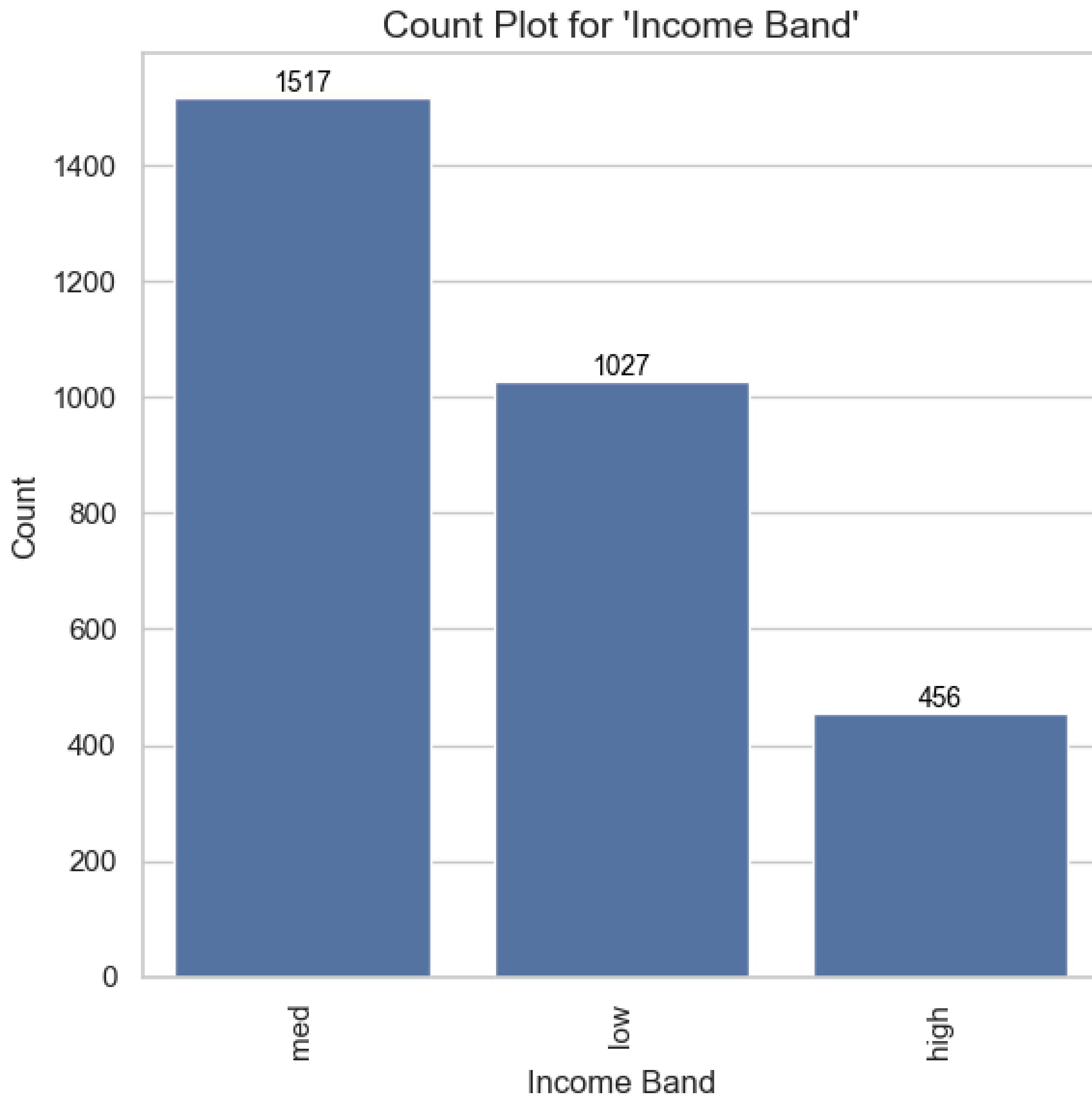

Count Plot for 'Loyalty Classification'
1331
767
585
317
Count
0
200
400
600
800
1000
1200
Jade
Silver
Gold
Platinum
Loyalty Classification



Count Plot for 'Properties Owned'
800
700
600
500
400
300
200
100
0
Count
705
776
777
742
0
1
2
3
Properties Owned



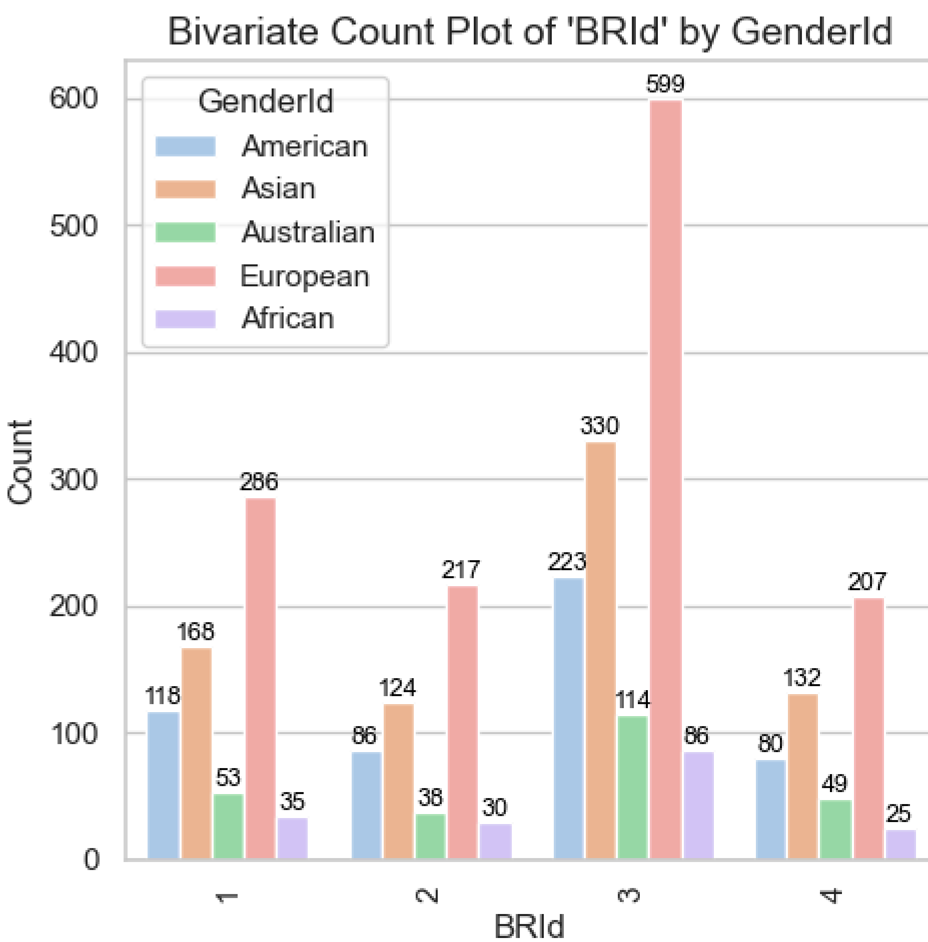
Count Plot for 'Risk Weighting'
Count
Risk Weighting
0
200
400
600
800
1000
1200
836
1222
460
322
160
1
2
3
4
5

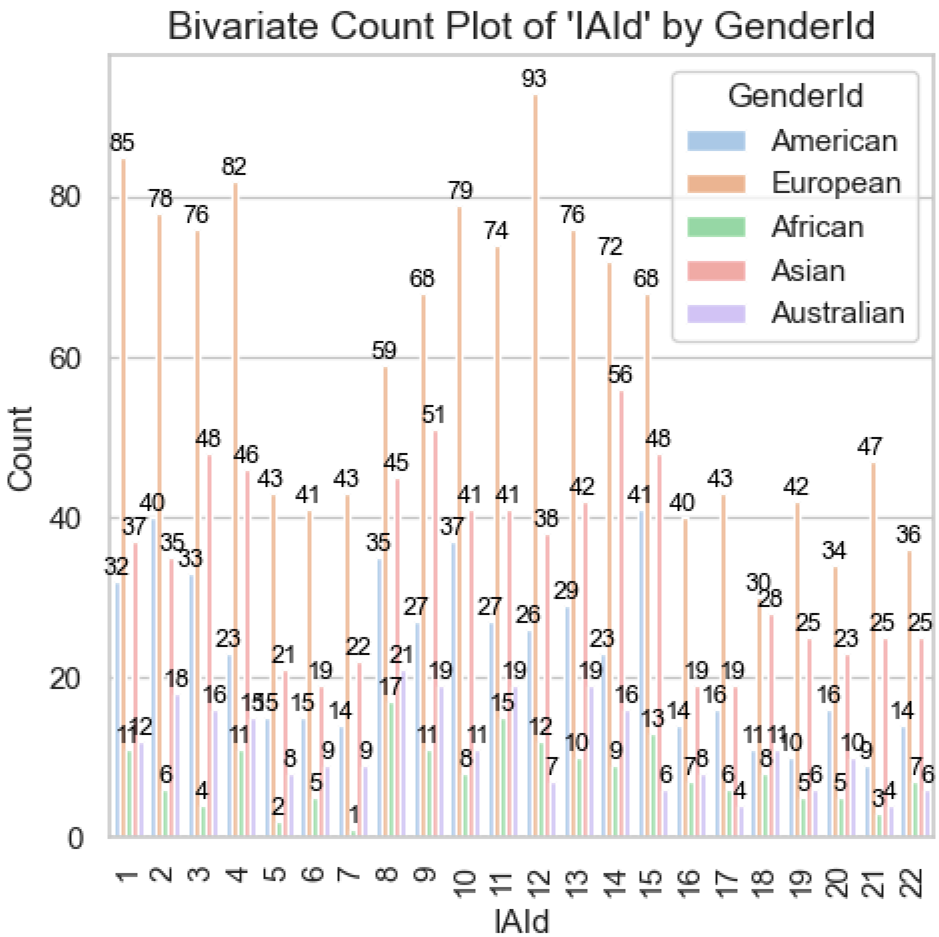


Bivariate Analysis

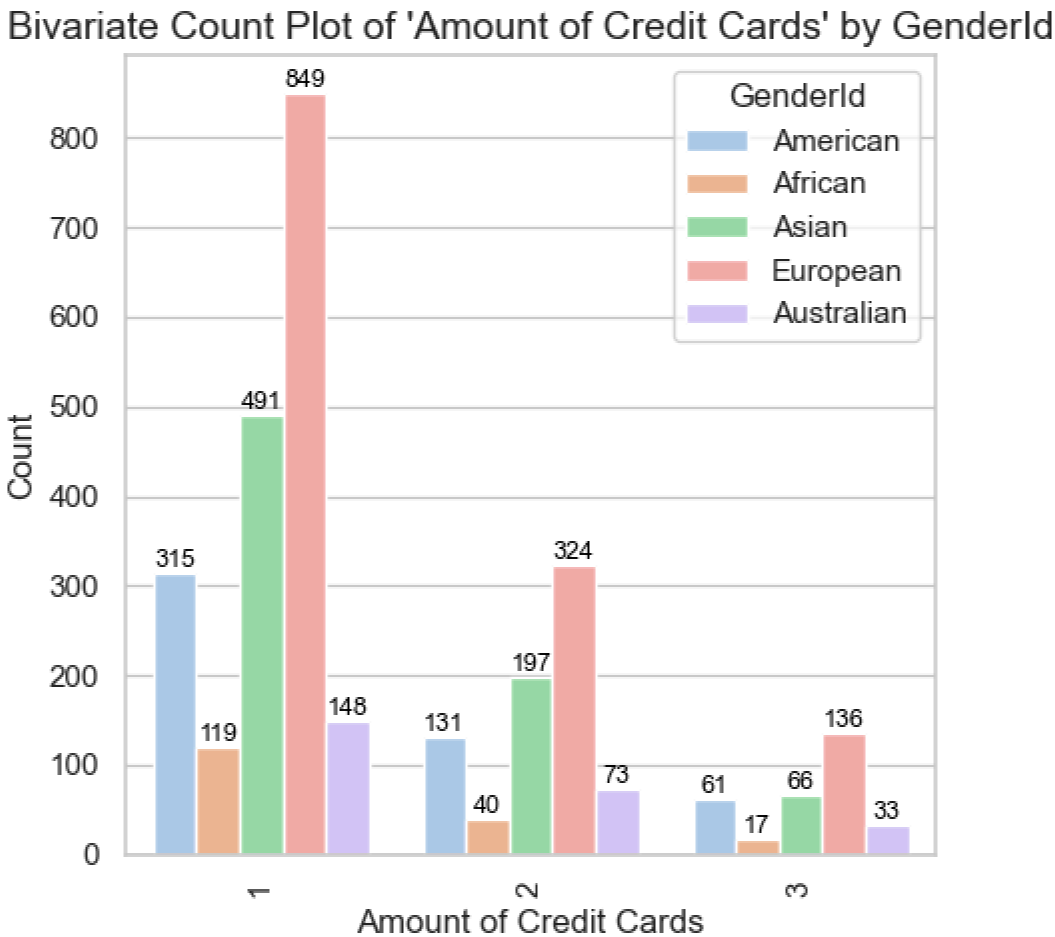
In [32]:
```
sns.set(style="whitegrid")

for col in categorical_cols:
    if col == "GenderId":
        continue  # Skip GenderId since it's used as hue

    plt.figure(figsize=(5, 5))

    # Try to sort categories numerically if possible
    try:
        categories = sorted(df[col].dropna().unique(), key=lambda x: int(x))
    except:
        categories = df[col].value_counts().index

    # Plot with hue
    ax = sns.countplot(data=df, x=col, hue="Nationality", order=categories,
```

```
    # Add value labels on each bar
    for p in ax.patches:
        height = int(p.get_height())
        if height > 0:
            ax.annotate(f'{height}',
                        (p.get_x() + p.get_width() / 2., height),
                        ha='center', va='bottom', fontsize=9, color='black',

    # Formatting
    plt.title(f"Bivariate Count Plot of '{col}' by GenderId", fontsize=14)
    plt.xlabel(col, fontsize=12)
    plt.ylabel("Count", fontsize=12)
    plt.xticks(rotation=90)
    plt.legend(title="GenderId")
    plt.tight_layout()
    plt.show()
```

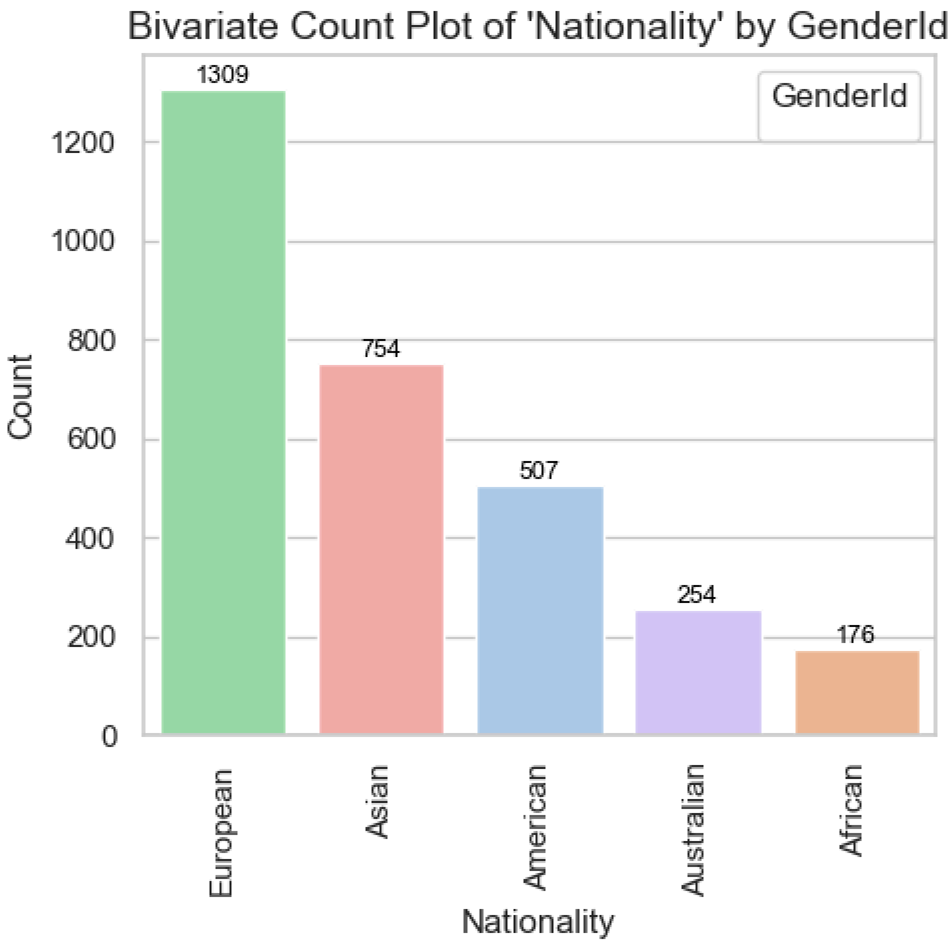




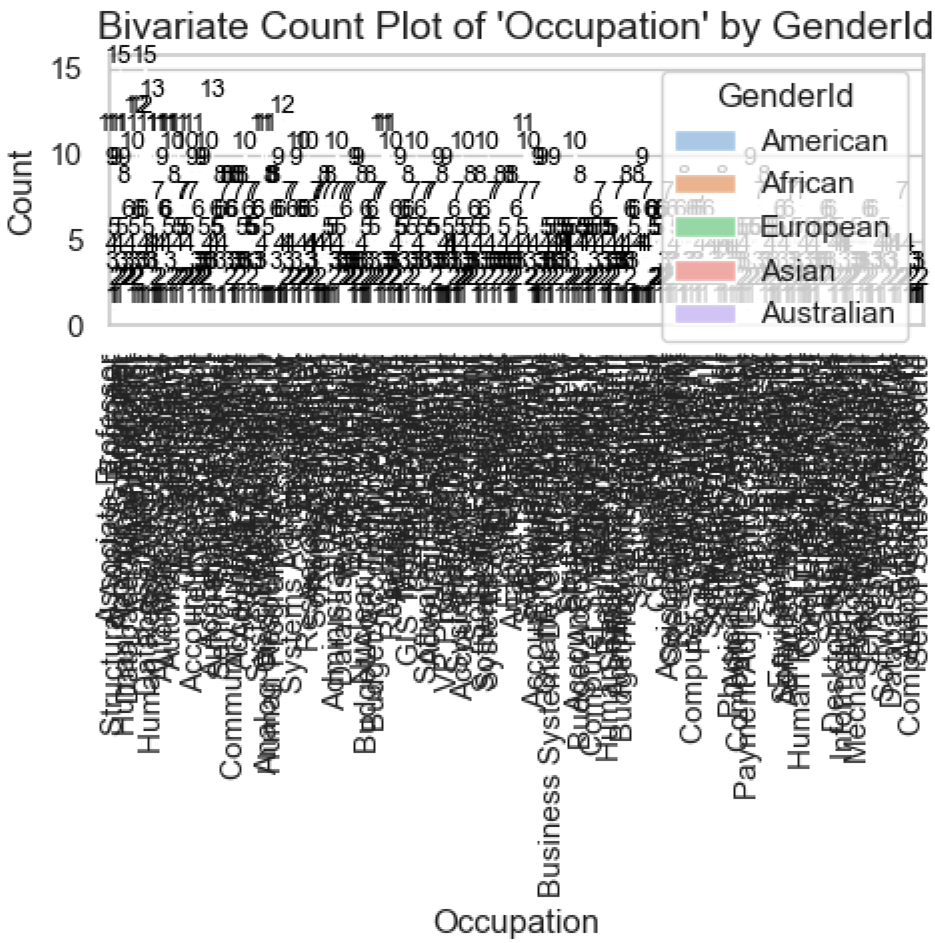
Bivariate Count Plot of 'IAId' by GenderId
GenderId
American
European
African
Asian
Australian
Count
IAId



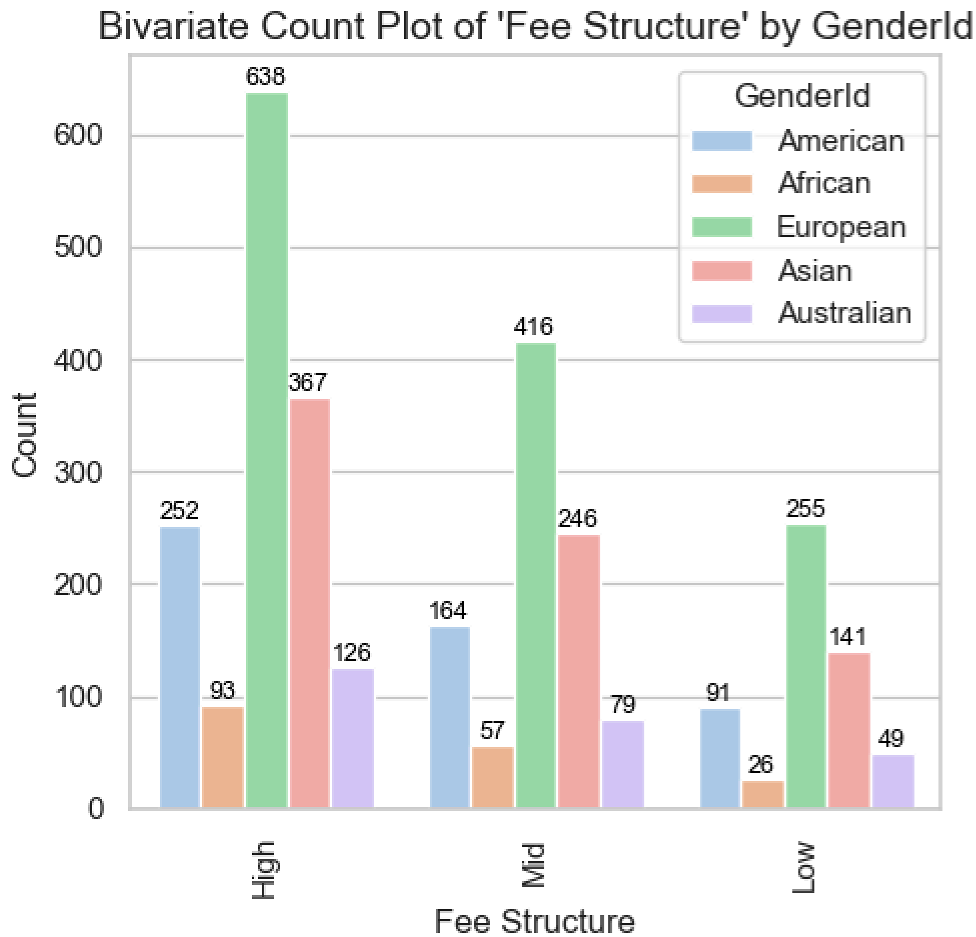

```
C:\Users\admin\AppData\Local\Temp\ipykernel_11452\1761577546.py:31: UserWarn
ing: No artists with labels found to put in legend.  Note that artists whose
label start with an underscore are ignored when legend() is called with no a
rgument.
  plt.legend(title="GenderId")
```

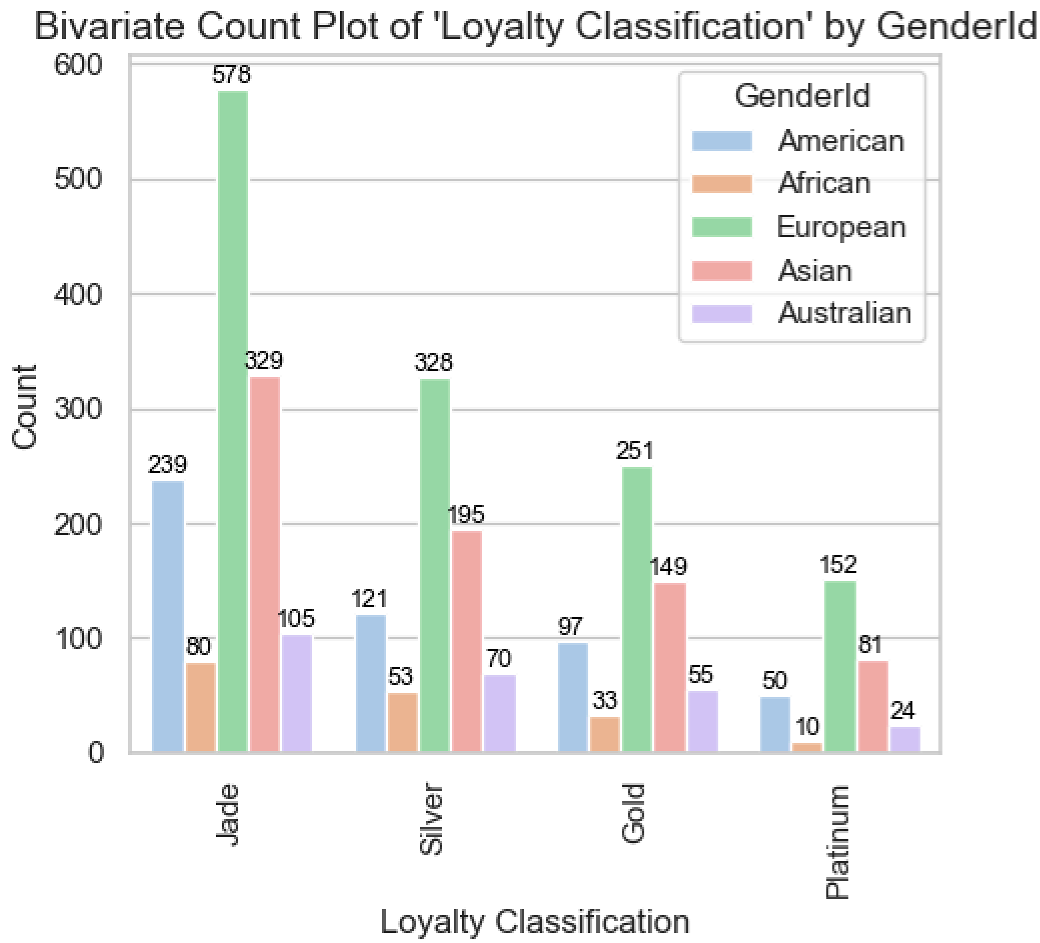


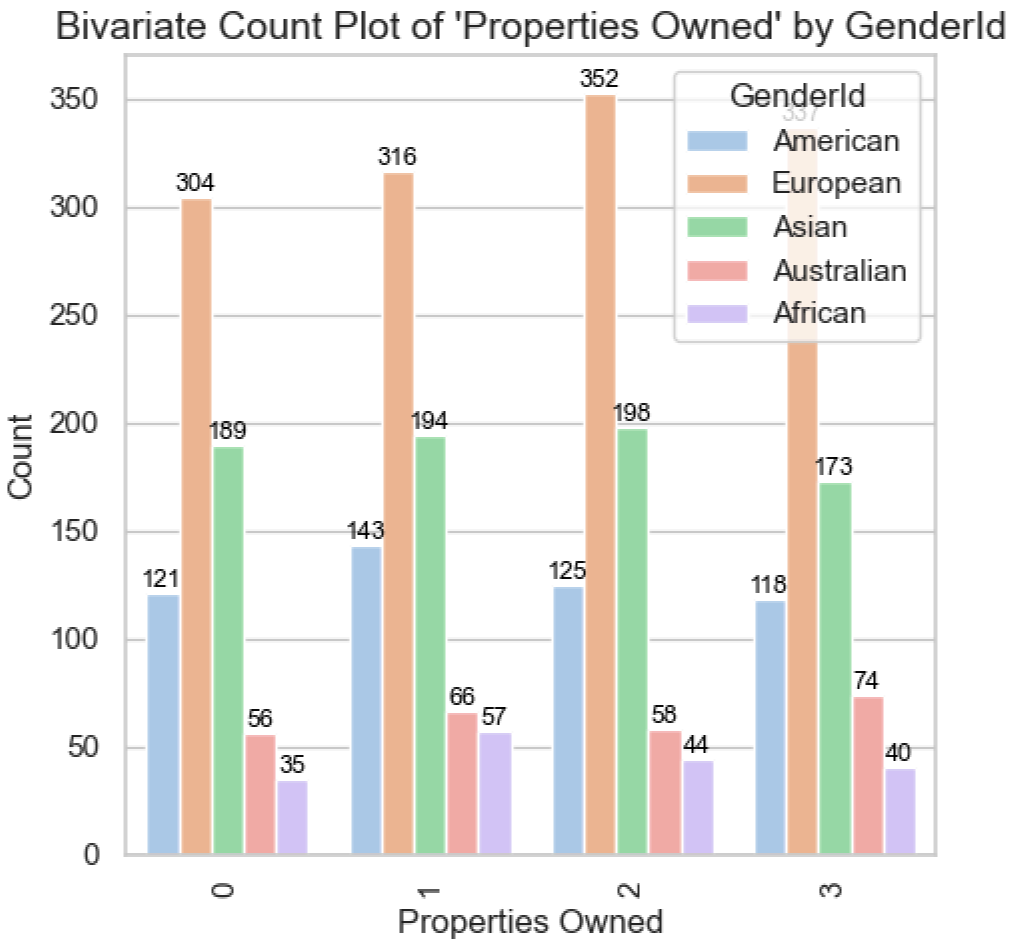
Bivariate Count Plot of 'Nationality' by GenderId
GenderId
1309
754
507
254
176
1200
1000
800
600
400
200
0
Count
European
Asian
American
Australian
African
Nationality



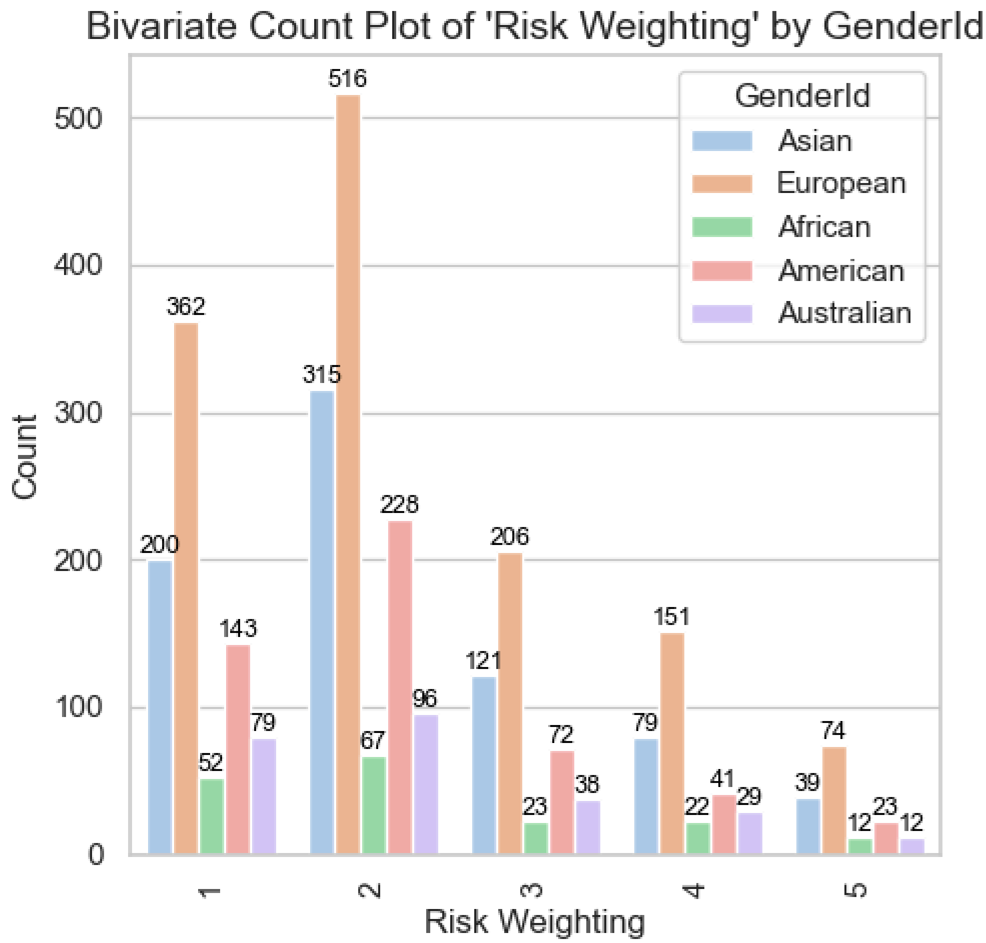
Bivariate Count Plot of 'Occupation' by GenderId
Count
Occupation
GenderId
American
African
European
Asian
Australian


Bivariate Count Plot of 'Fee Structure' by GenderId
GenderId
American
African
European
Asian
Australian
Count
Fee Structure
High
Mid
Low
0
100
200
300
400
500
600
638
367
252
93
126
416
246
164
57
79
255
141
91
26
49

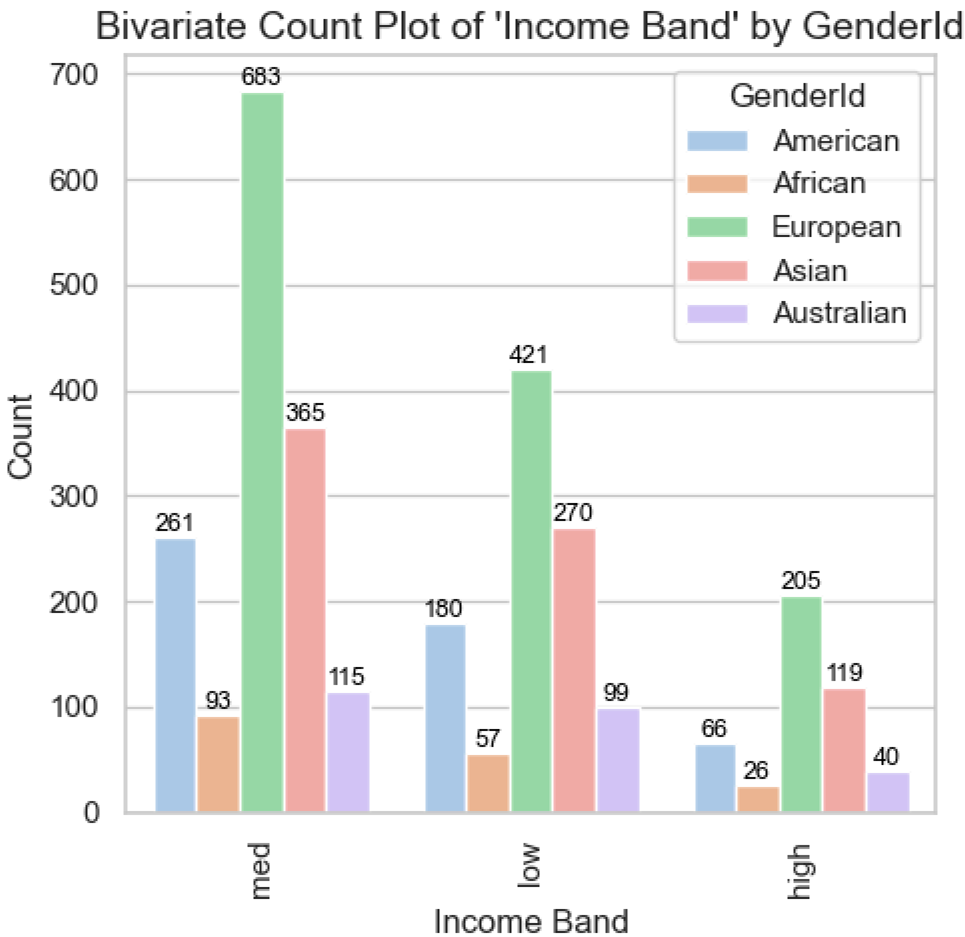

Bivariate Count Plot of 'Loyalty Classification' by GenderId
Count
Loyalty Classification
GenderId
American
African
European
Asian
Australian
Jade
Silver
Gold
Platinum
0
100
200
300
400
500
600
239
80
578
329
105
121
53
328
195
70
97
33
251
149
55
50
10
152
81
24



Bivariate Count Plot of 'Properties Owned' by GenderId
GenderId
American
European
Asian
Australian
African
Count
Properties Owned
0
50
100
150
200
250
300
350
0
1
2
3
121
304
189
56
35
143
316
194
66
57
125
352
198
58
44
118
337
173
74
40

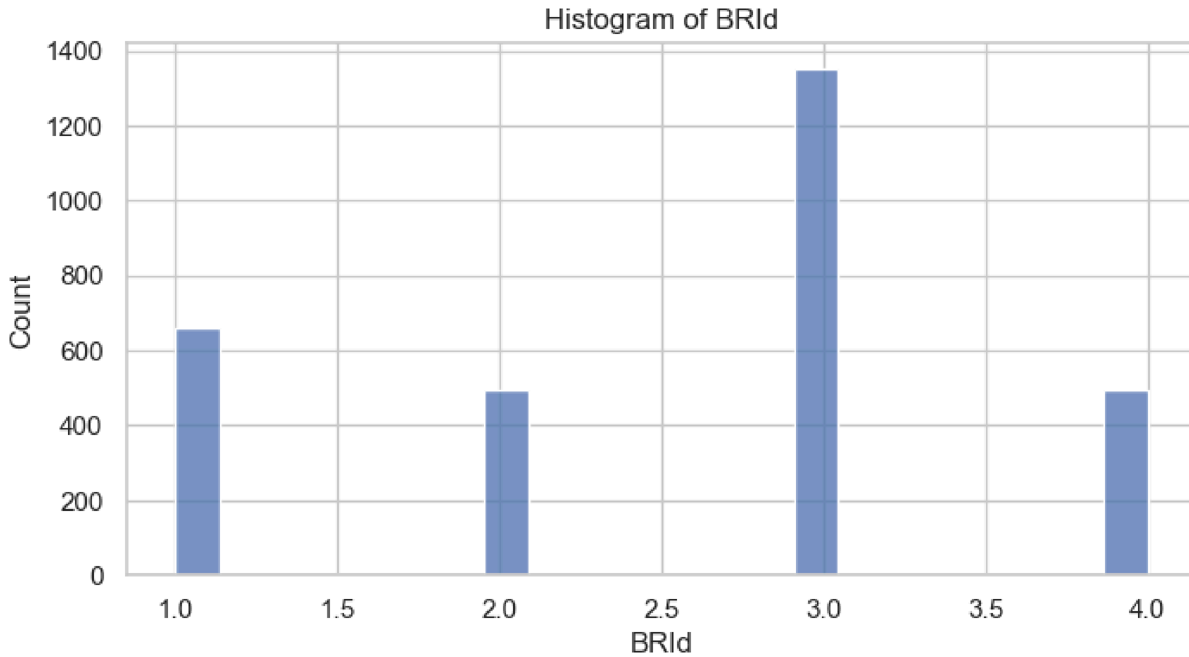


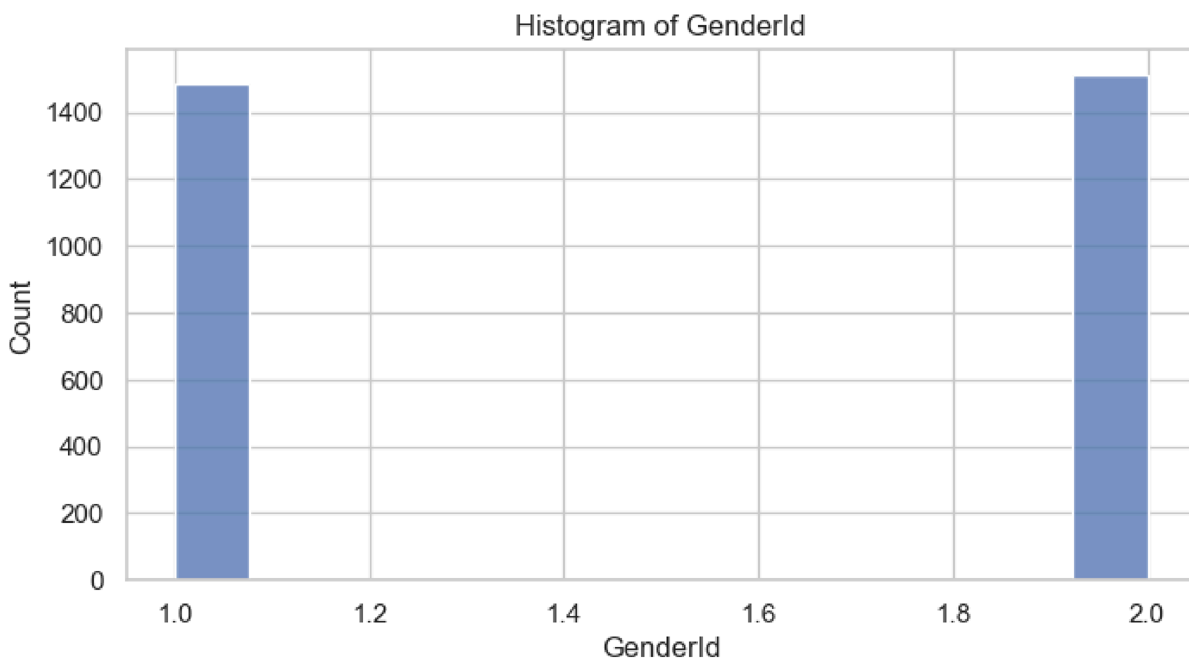
Bivariate Count Plot of 'Risk Weighting' by GenderId
GenderId
Asian
European
African
American
Australian
Count
0
100
200
300
400
500
Risk Weighting
1
2
3
4
5
200
362
52
143
79
315
516
67
228
96
121
206
23
72
38
79
151
22
41
29
39
74
12
23
12

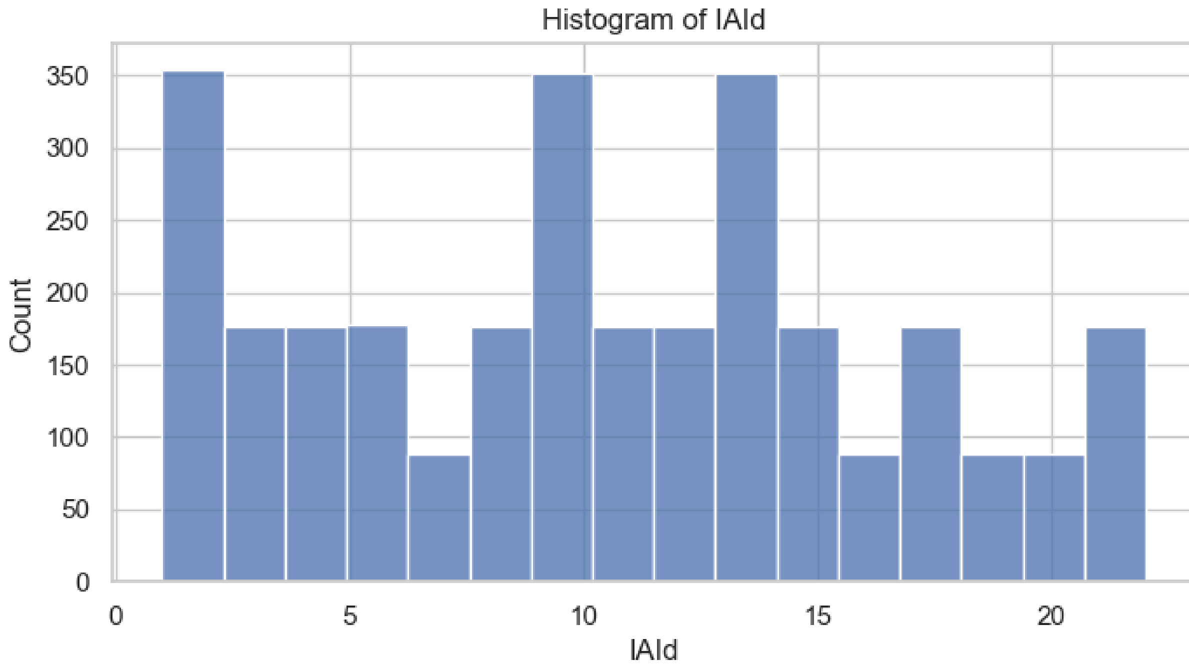


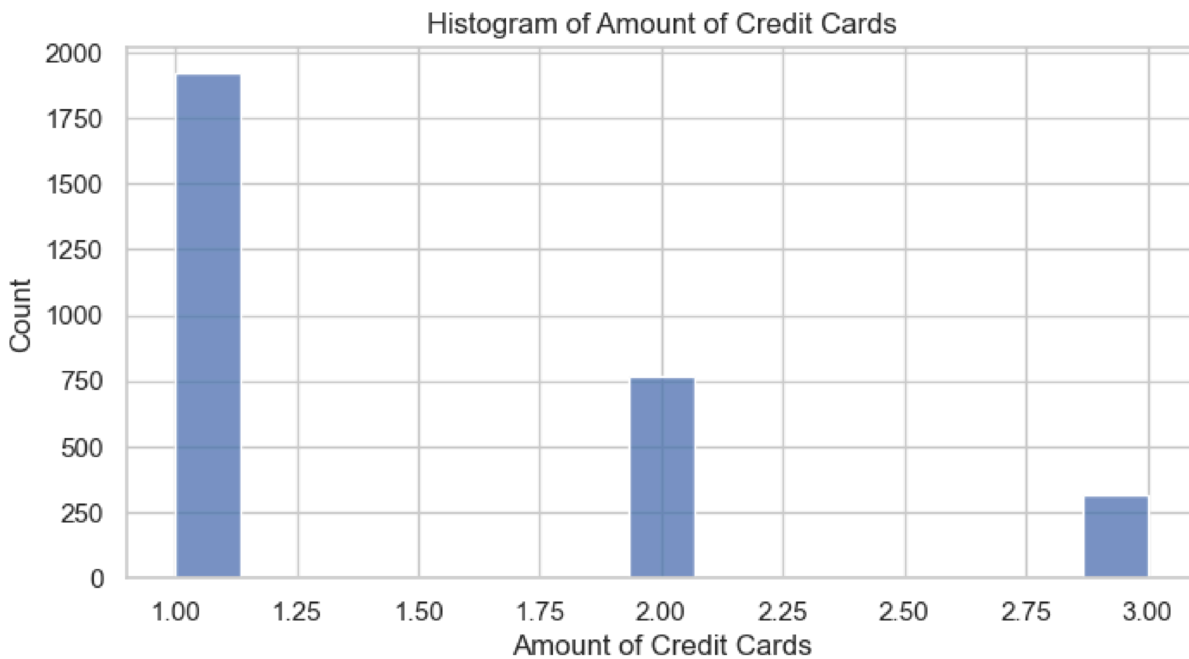
In [ ]:
```
# Histplot of value counts for different occupation
```

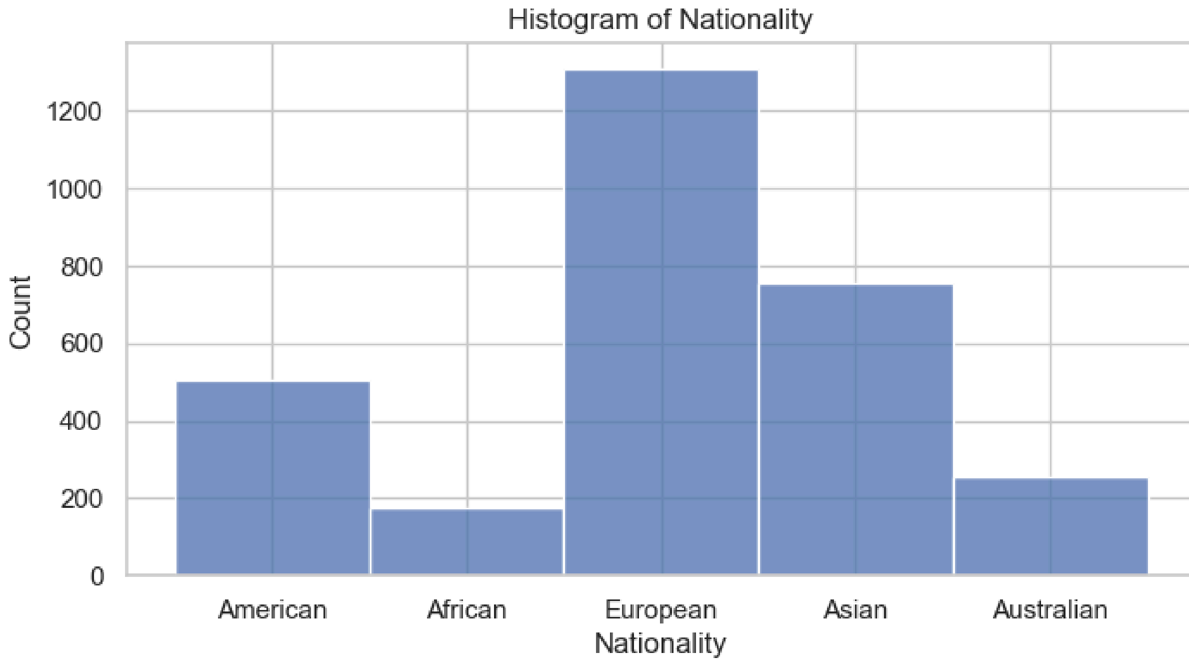
In [38]:
```
for col in categorical_cols:
    if col == "Occupation":
        continue  # Skip Occupation column

    plt.figure(figsize=(8, 4))
    sns.histplot(df[col])
    plt.title(f'Histogram of {col}')
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.show()
```

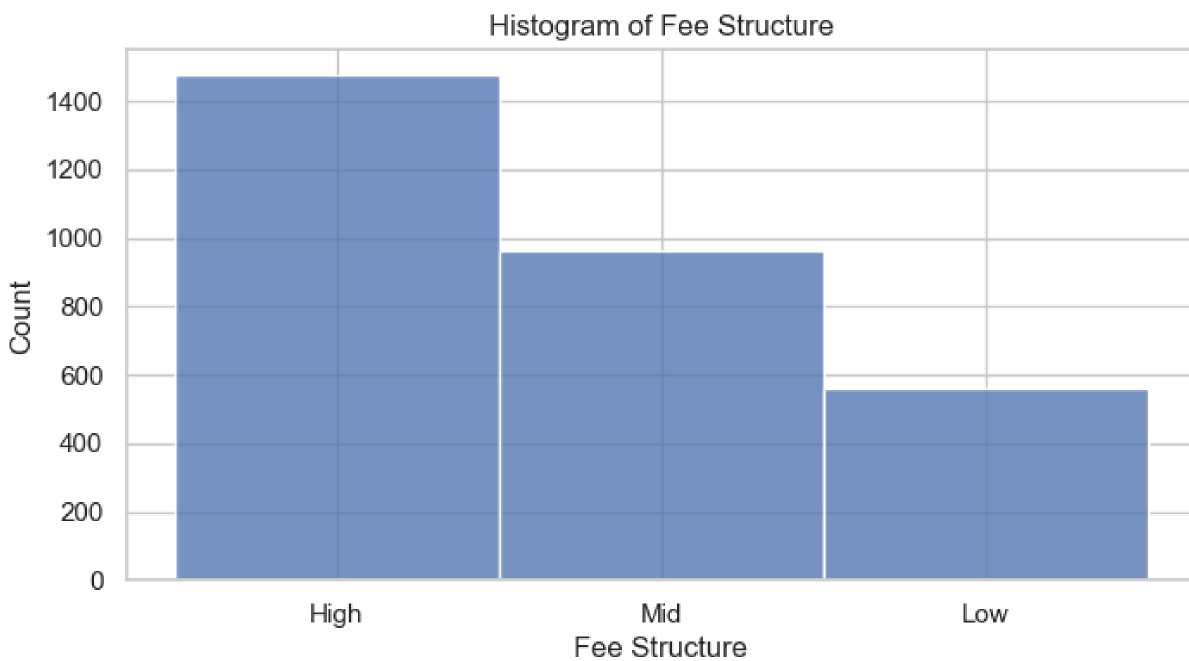

Histogram of BRId
Count
BRId
1400
1200
1000
800
600
400
200
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0

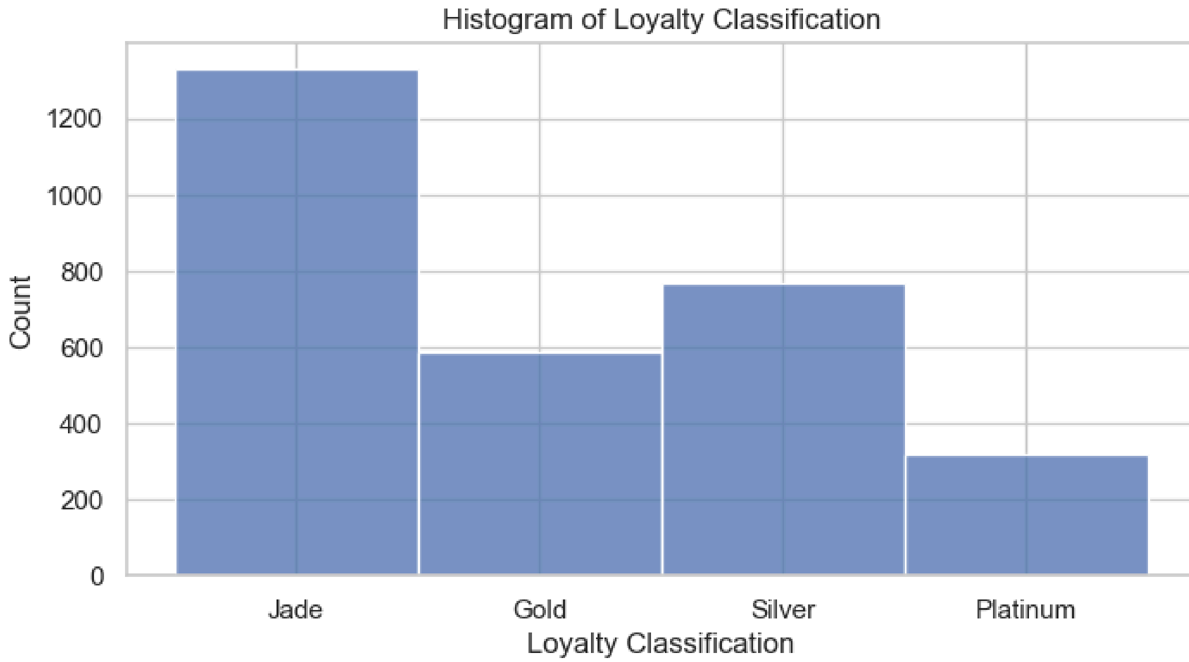

Histogram of GenderId
Count
GenderId
1400
1200
1000
800
600
400
200
0
1.0
1.2
1.4
1.6
1.8
2.0

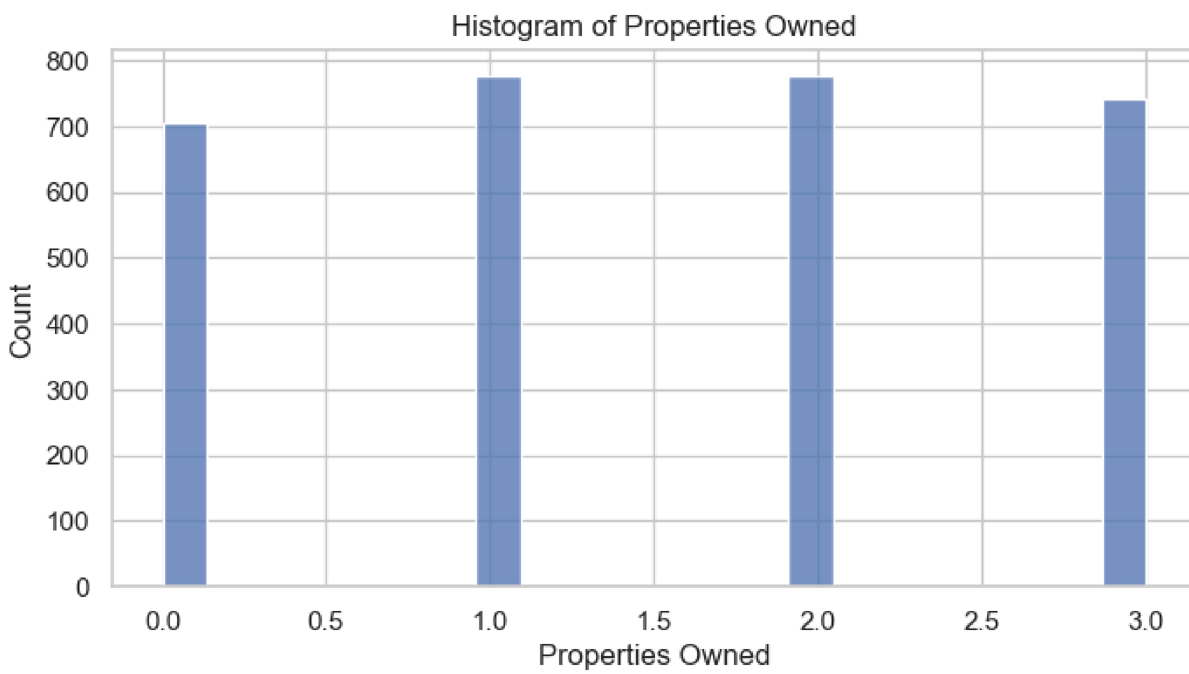

Histogram of IAId
Count
IAId
0
50
100
150
200
250
300
350
0
5
10
15
20

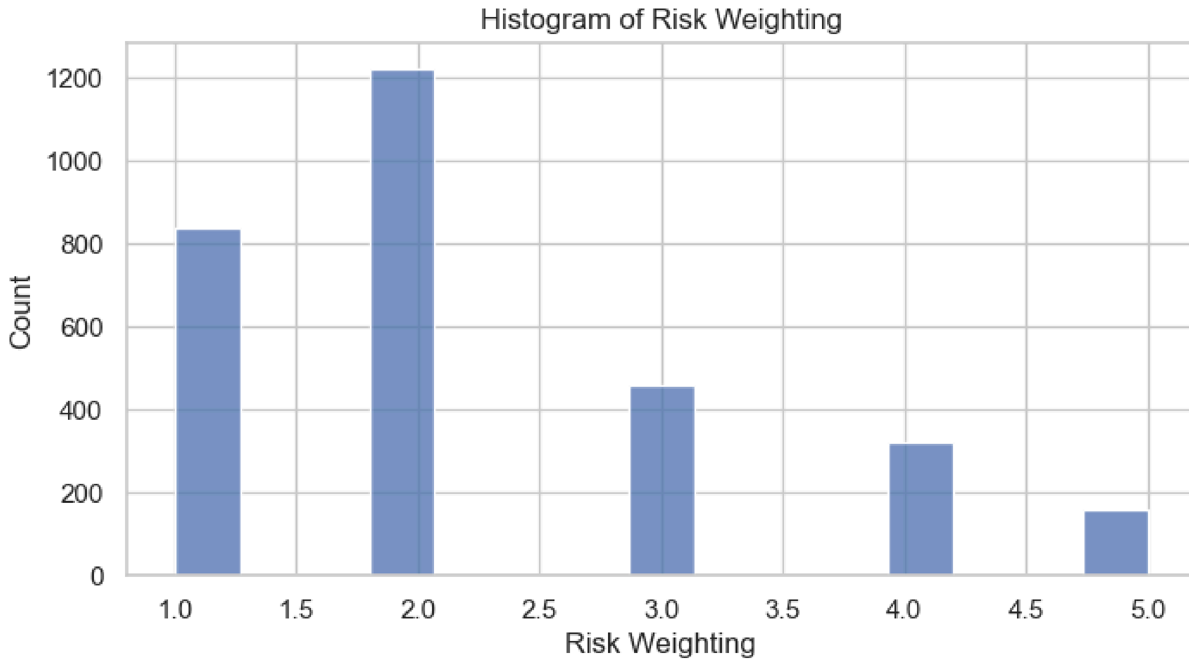

Histogram of Amount of Credit Cards
Count
Amount of Credit Cards
0
250
500
750
1000
1250
1500
1750
2000
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00

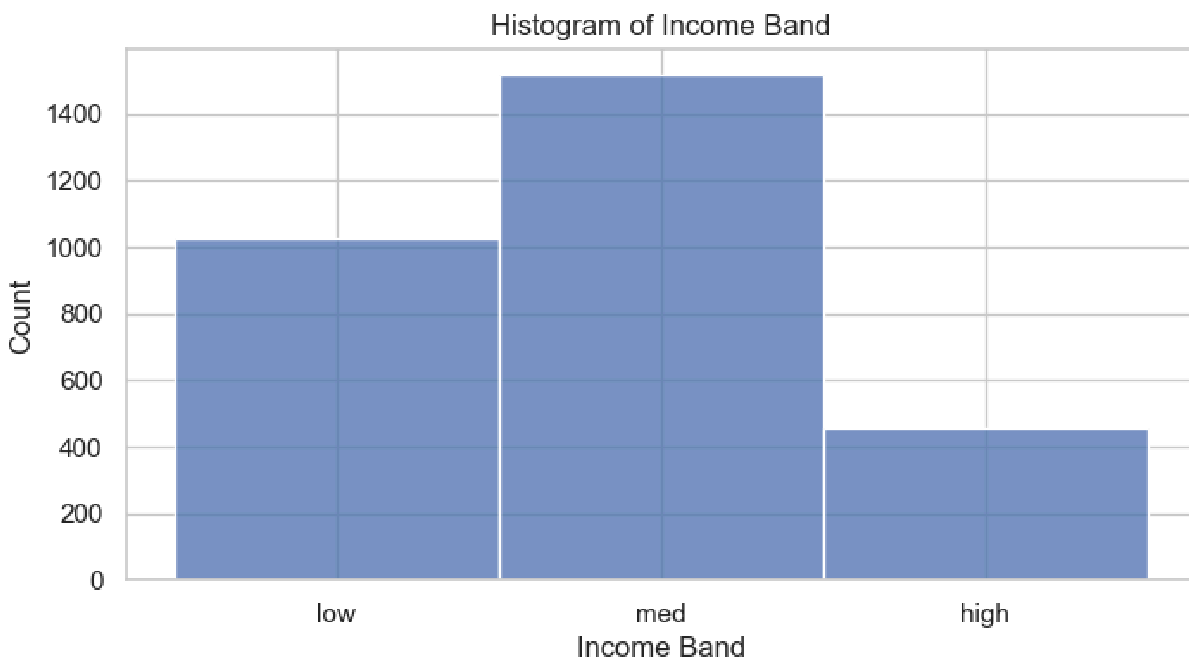

Histogram of Nationality
Count
0
200
400
600
800
1000
1200
American
African
European
Asian
Australian
Nationality


Histogram of Fee Structure
Count
0
200
400
600
800
1000
1200
1400
High
Mid
Low
Fee Structure


Histogram of Loyalty Classification
Count
1200
1000
800
600
400
200
0
Jade
Gold
Silver
Platinum
Loyalty Classification

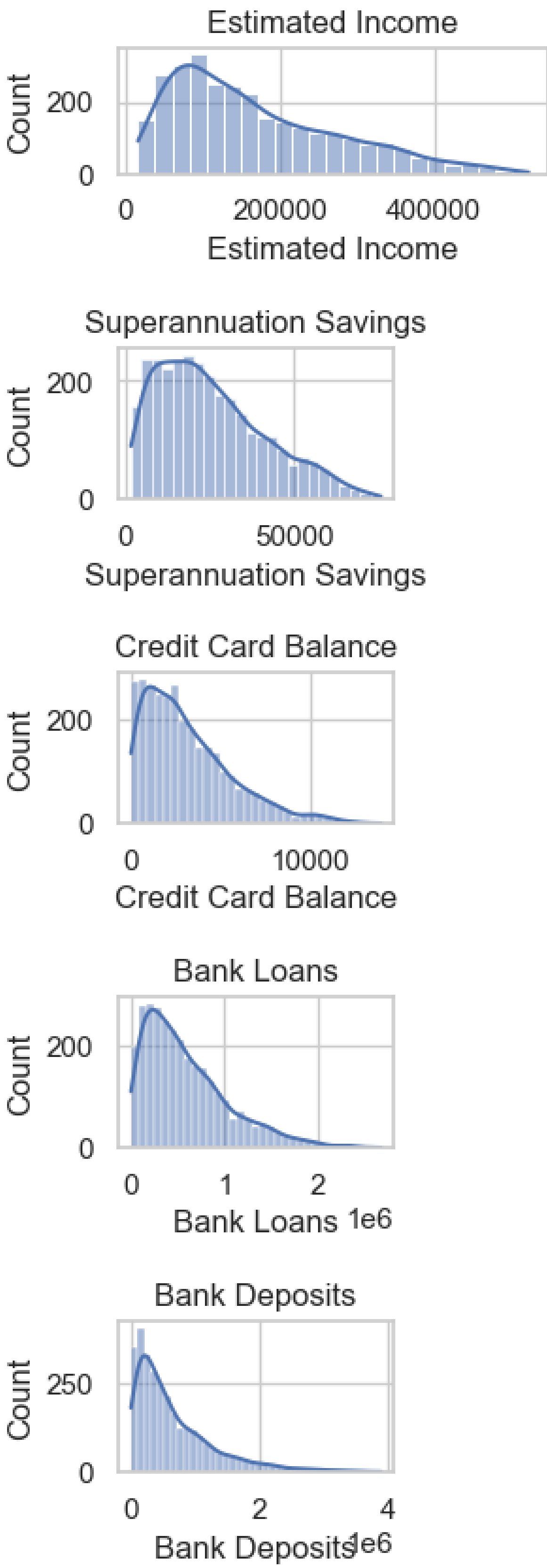

Histogram of Properties Owned
Count
800
700
600
500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Properties Owned

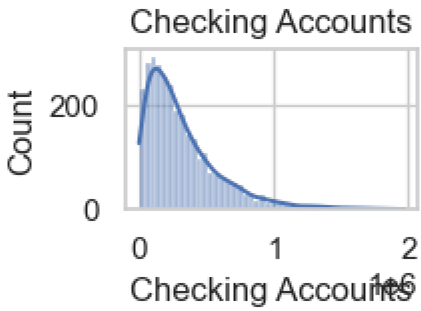


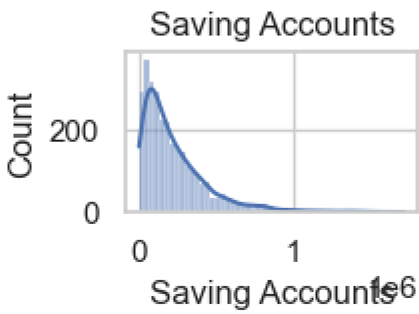


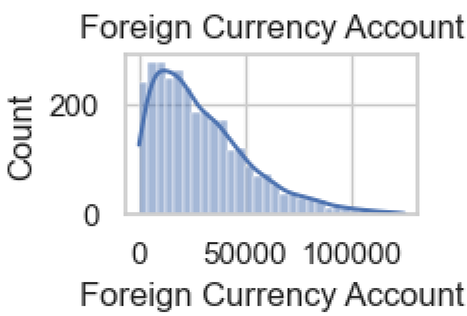
Numerical Analysis

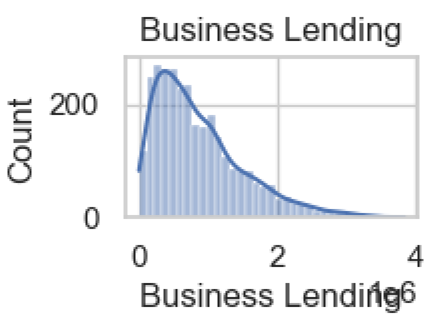
In [47]:
```
numerical_cols=['Estimated Income', 'Superannuation Savings','Credit Card Ba

#Univariate analysis and Visualization
plt.figure(figsize=(10,4))
for i,col in enumerate(numerical_cols):
    plt.subplot(4,3,i+1)
    sns.histplot(df[col],kde=True)
    plt.title(col)
    plt.show()
```


Estimated Income
Count
200
0
0
200000
400000
Estimated Income
Superannuation Savings
Count
200
0
0
50000
Superannuation Savings
Credit Card Balance
Count
200
0
0
10000
Credit Card Balance
Bank Loans
Count
200
0
0
1
2
Bank Loans
1e6
Bank Deposits
Count
250
0
0
2
4
Bank Deposits
1e6









In [53]:

```
numerical_cols=['Estimated Income', 'Superannuation Savings','Credit Card Ba

correlation_matrix= df[numerical_cols].corr()
plt.figure(figsize=(10,8))
sns.heatmap(correlation_matrix, annot=True, cmap='crest',fmt=".2f")
plt.title("Correlation matrix")
plt.show()
```

Insight of EDA:

1. The strongest posituve correlation occurs among "Bank Deposits" and "Checking Accounts", "Saving Account" AND "Foreign Currency Account" imdicating customers wjo maintain high balances in one account type often hold substantial amount/funds across other accounts as well.