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

In [2]:
```
df = pd.read_csv("Banking.csv")
```

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

Out[3]:

	Client ID	Name	Age	Location ID	Joined Bank	Banking Contact	Nationality	Occupation	Fee Structure	Loyalty Classification	...	Bank Deposits	Check Accou
0	IND81288	Raymond Mills	24	34324	06-05-2019	Anthony Torres	American	Safety Technician IV	High	Jade	...	1485828.64	60361
1	IND65833	Julia Spencer	23	42205	10-12-2001	Jonathan Hawkins	African	Software Consultant	High	Jade	...	641482.79	22952
2	IND47499	Stephen Murray	27	7314	25-01-2010	Anthony Berry	European	Help Desk Operator	High	Gold	...	1033401.59	65267
3	IND72498	Virginia Garza	40	34594	28-03-2019	Steve Diaz	American	Geologist II	Mid	Silver	...	1048157.49	104815
4	IND60181	Melissa Sanders	46	41269	20-07-2012	Shawn Long	American	Assistant Professor	Mid	Platinum	...	487782.53	44664

5 rows × 25 columns

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

Out[4]:

	Age	Location ID	Estimated Income	Superannuation Savings	Amount of Credit Cards	Credit Card Balance	Bank Loans	Bank Deposits	Ch Ac
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3.000000e+03	3.000000e+03	3.00000
mean	51.039667	21563.323000	171305.034263	25531.599673	1.463667	3176.206943	5.913862e+05	6.715602e+05	3.21092
std	19.854760	12462.273017	111935.808209	16259.950770	0.676387	2497.094709	4.575570e+05	6.457169e+05	2.82079
min	17.000000	12.000000	15919.480000	1482.030000	1.000000	1.170000	0.000000e+00	0.000000e+00	0.00000
25%	34.000000	10803.500000	82906.595000	12513.775000	1.000000	1236.630000	2.396281e+05	2.044004e+05	1.19947
50%	51.000000	21129.500000	142313.480000	22357.355000	1.000000	2560.805000	4.797934e+05	4.633165e+05	2.42815
75%	69.000000	32054.500000	242290.305000	35464.740000	2.000000	4522.632500	8.258130e+05	9.427546e+05	4.34874
max	85.000000	43369.000000	522330.260000	75963.900000	3.000000	13991.990000	2.667557e+06	3.890598e+06	1.96992

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 25 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   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 [6]:
```
df.shape
```

Out[6]:
```
(3000, 25)
```

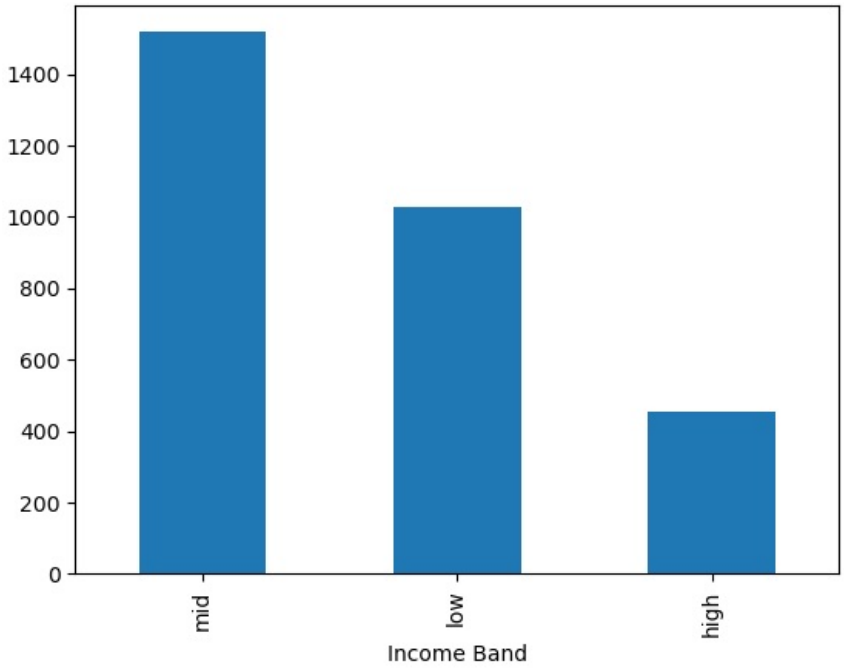
In [7]:
```
bins = [0, 100000, 300000, float('inf')]
labels = ["low", "mid", "high"]
df['Income Band'] = pd.cut(df['Estimated Income'], bins=bins, labels=labels, right=False)
```

In [8]:
```
df["Income Band"].head()
```

Out[8]:
```
0     low
1     mid
2     mid
3    high
4     mid
Name: Income Band, dtype: category
Categories (3, object): ['low' < 'mid' < 'high']
```

In [9]:
```
df['Income Band'].value_counts().plot(kind='bar')
```

Out[9]:
```
<Axes: xlabel='Income Band'>
```



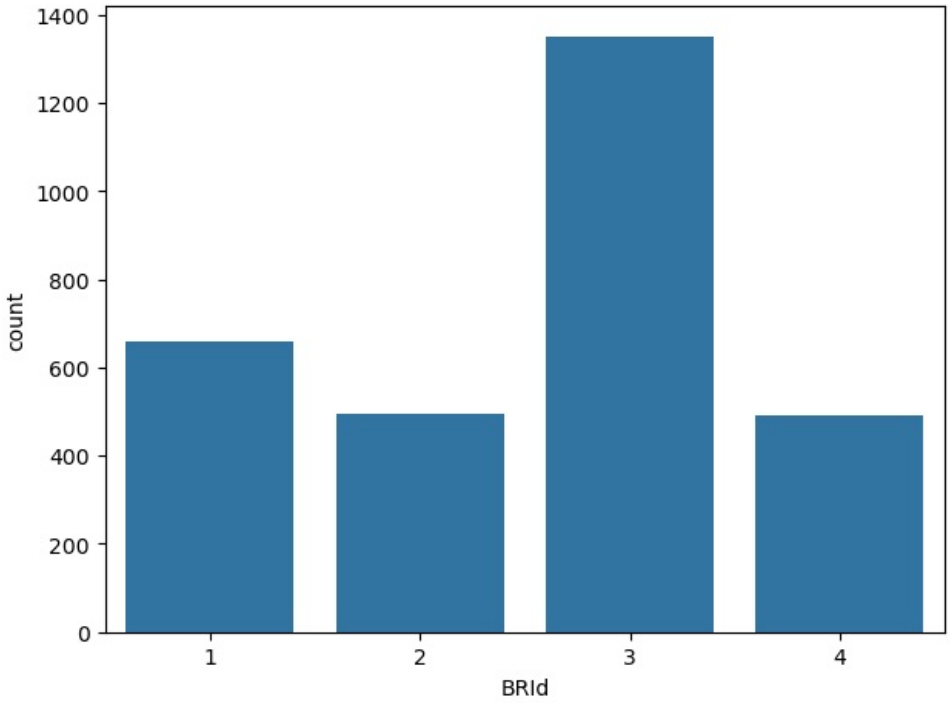
In [10]:
```
#Examine the distribution of unique categories in categorical columns
```

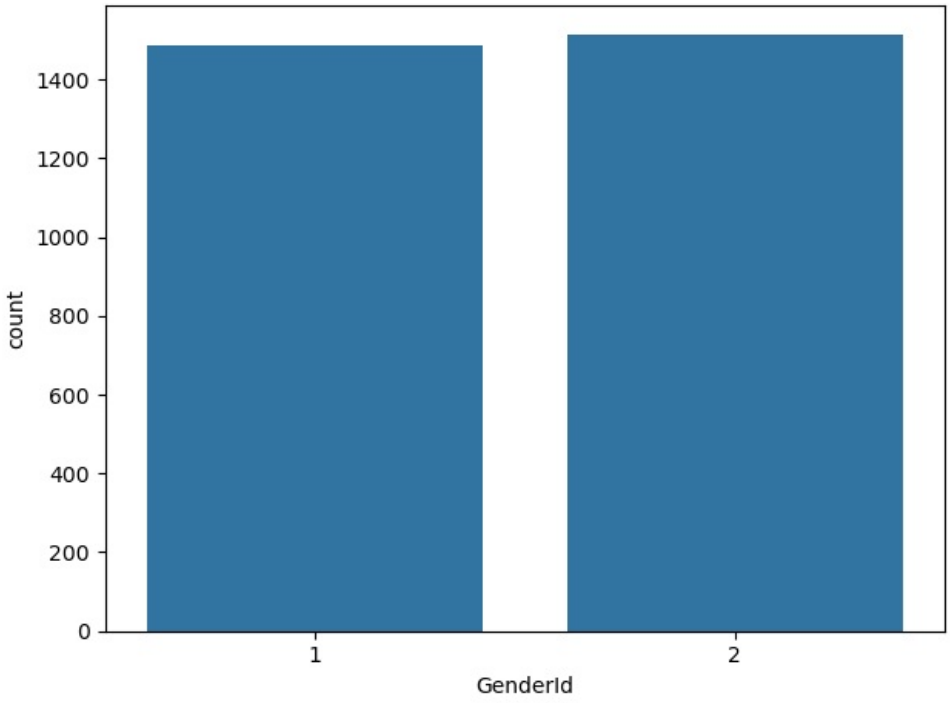
```
categorical_cols = df[["BRId", "GenderId", "IAId", "Amount of Credit Cards", "Nationality", "Occupation", "Fee S

for col in categorical_cols:
    print(f"value Counts for '{col}' :")
    display(df[col].value_counts())
```

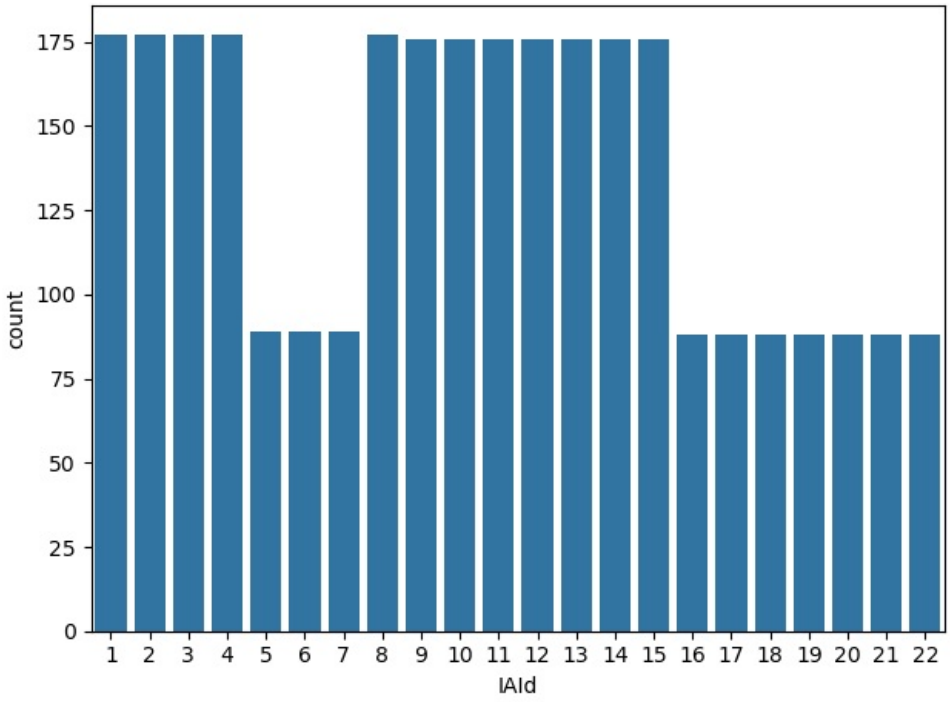
```
value Counts for 'BRId' :
BRId
3    1352
1     660
2     495
4     493
Name: count, dtype: int64
value Counts for 'GenderId' :
GenderId
2    1512
1    1488
Name: count, dtype: int64
value Counts for 'IAId' :
IAId
1     177
3     177
4     177
8     177
2     177
11    176
15    176
14    176
13    176
12    176
10    176
9     176
7      89
6      89
5      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
Structural Analysis Engineer    28
Associate Professor             28
Recruiter                       25
Human Resources Manager         24
Account Coordinator             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
```

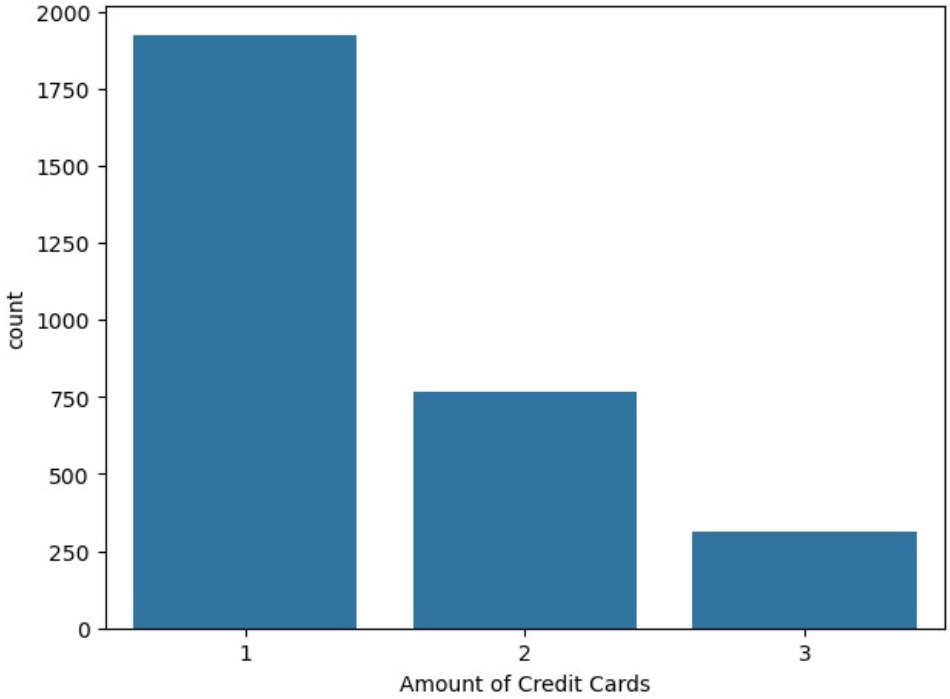
```
value Counts for 'Properties Owned' :
Properties Owned
2    777
1    776
3    742
0    705
Name: count, dtype: int64
value Counts for 'Risk Weighting' :
Risk Weighting
2    1222
1     836
3     460
4     322
5     160
Name: count, dtype: int64
value Counts for 'Income Band' :
Income Band
mid     1517
low     1027
high     456
Name: count, dtype: int64
```

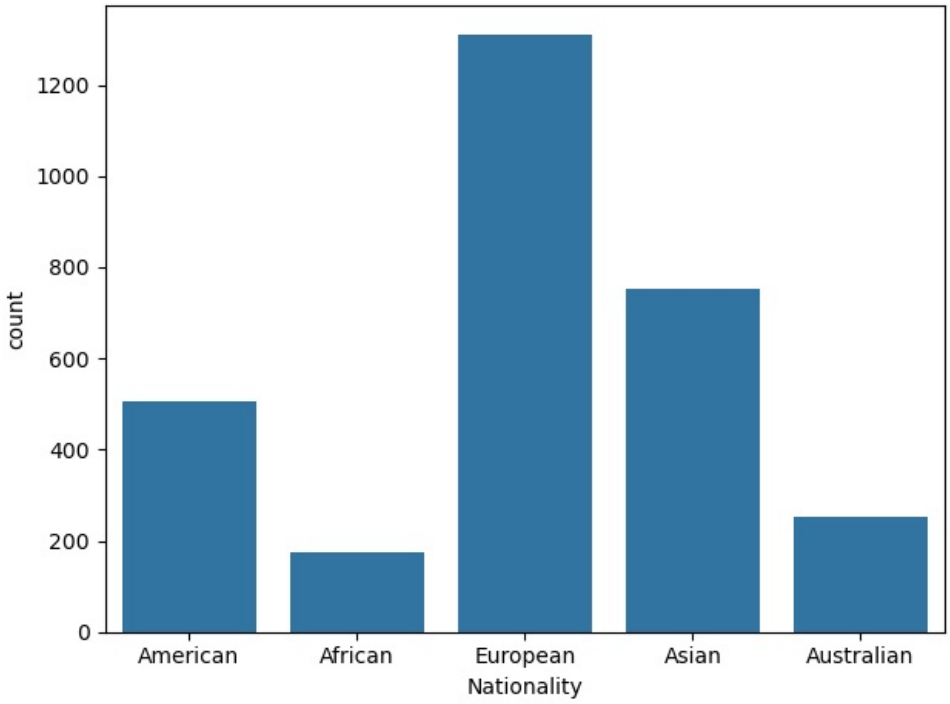
In [11]:
```
for i, predictor in enumerate(df[["BRId", "GenderId", "IAId", "Amount of Credit Cards", "Nationality", "Occupat
    plt.figure(i)
    sns.countplot(data=df, x=predictor)
    plt.tight_layout()
```

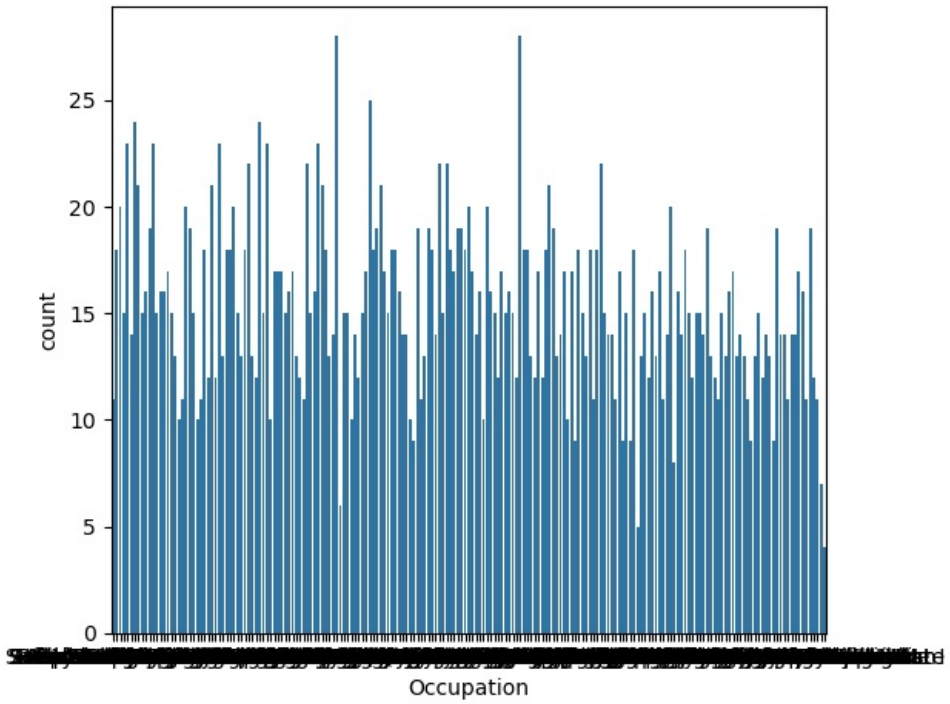


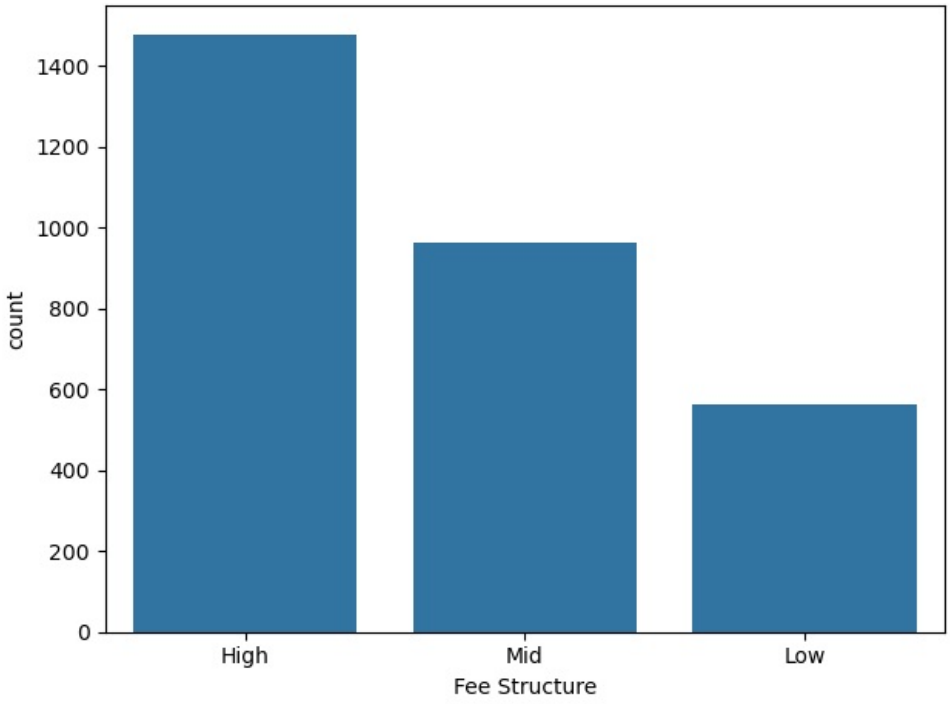


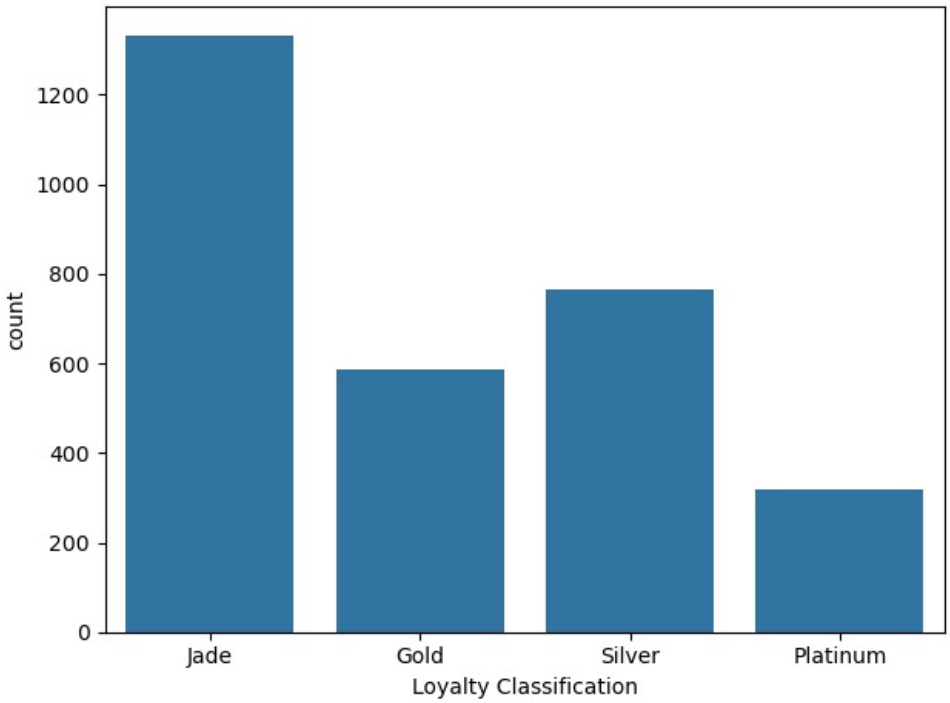

count
0
25
50
75
100
125
150
175
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
IAId

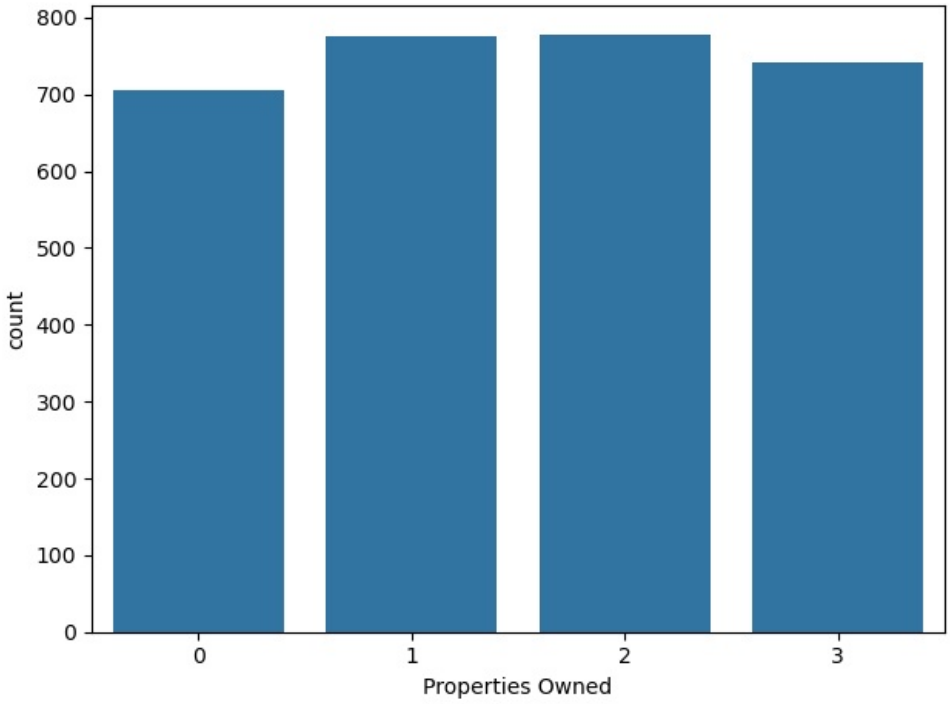

count
0
250
500
750
1000
1250
1500
1750
2000
1
2
3
Amount of Credit Cards

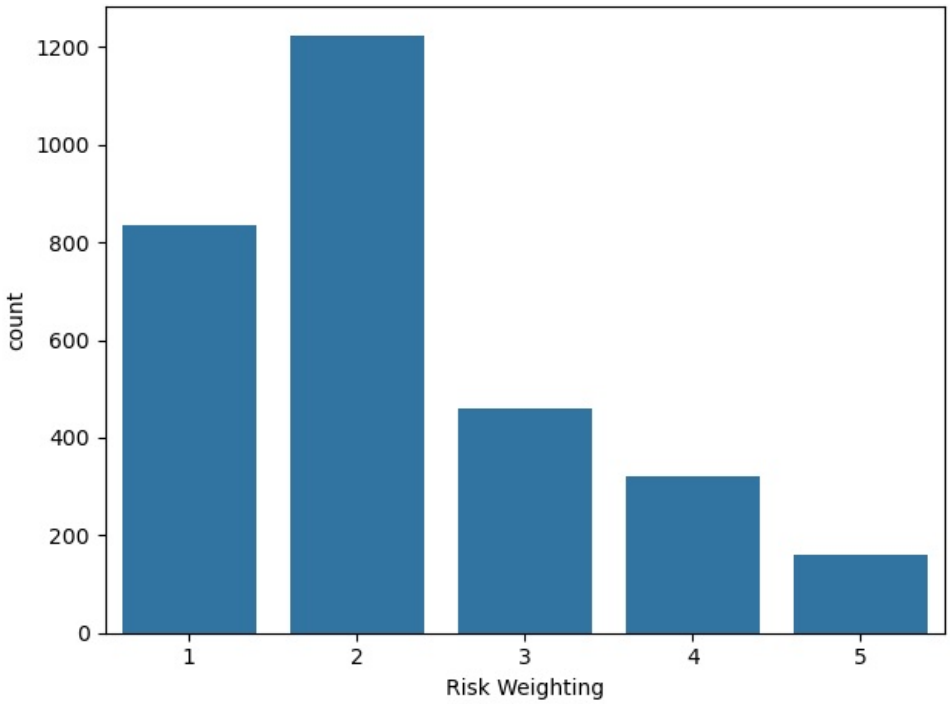

1200
1000
800
600
400
200
0
count
American
African
European
Asian
Australian
Nationality

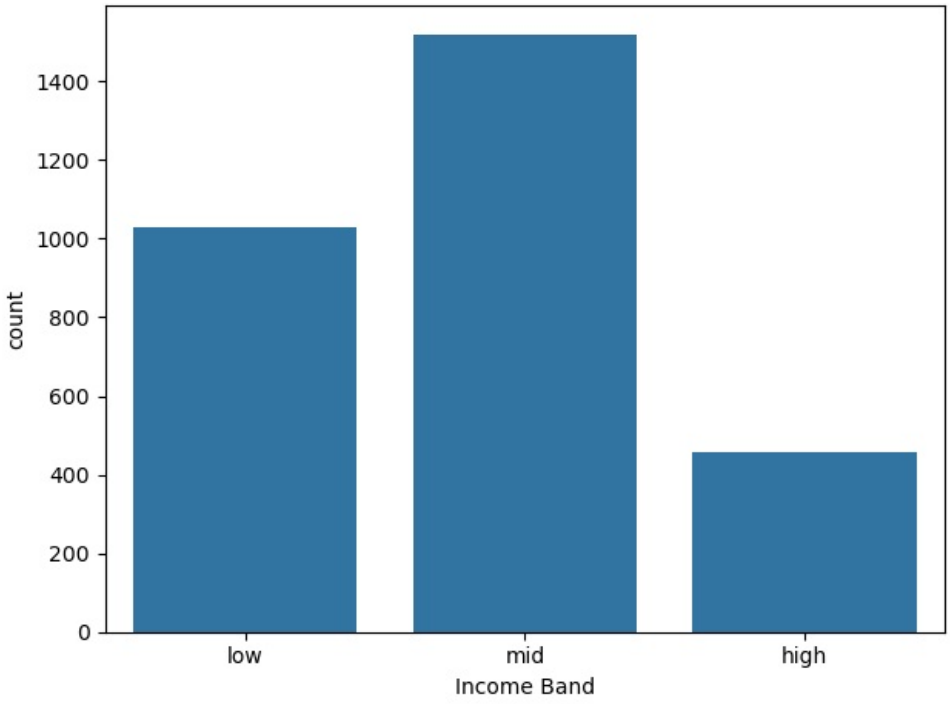

25
20
15
10
5
0
count
Occupation


1400
1200
1000
800
600
400
200
0
count
High
Mid
Low
Fee Structure

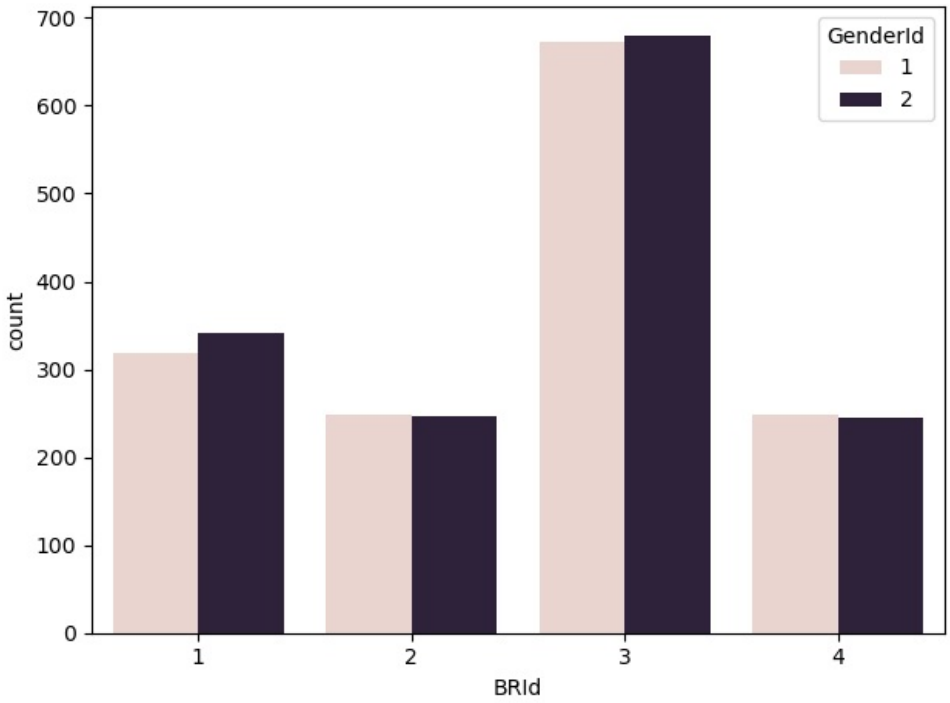

1200
1000
800
600
400
200
0
count
Jade
Gold
Silver
Platinum
Loyalty Classification

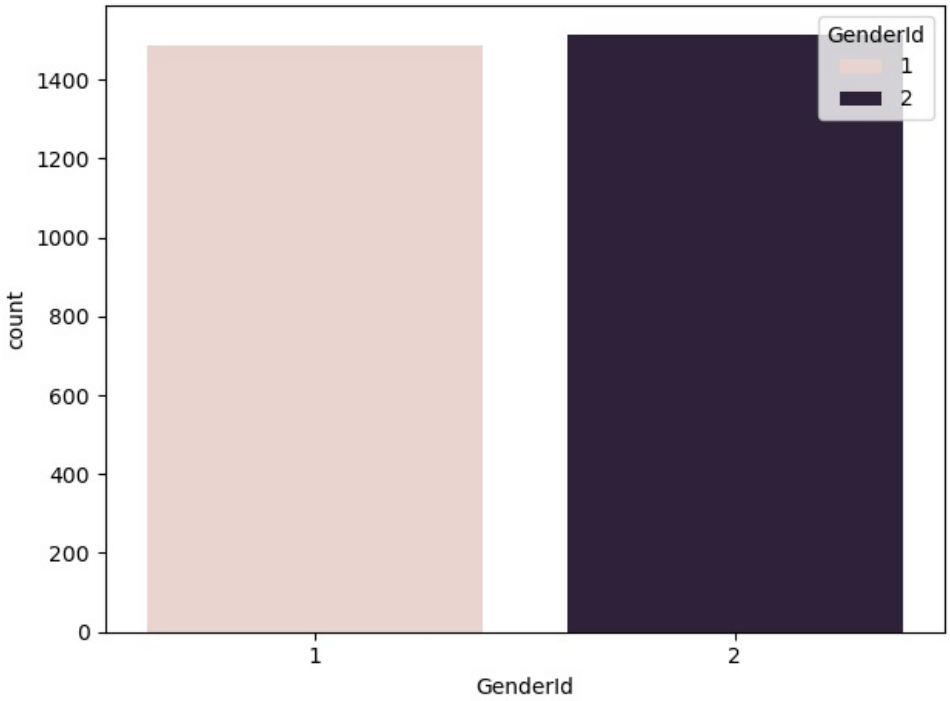

800
700
600
500
400
300
200
100
0
count
0
1
2
3
Properties Owned

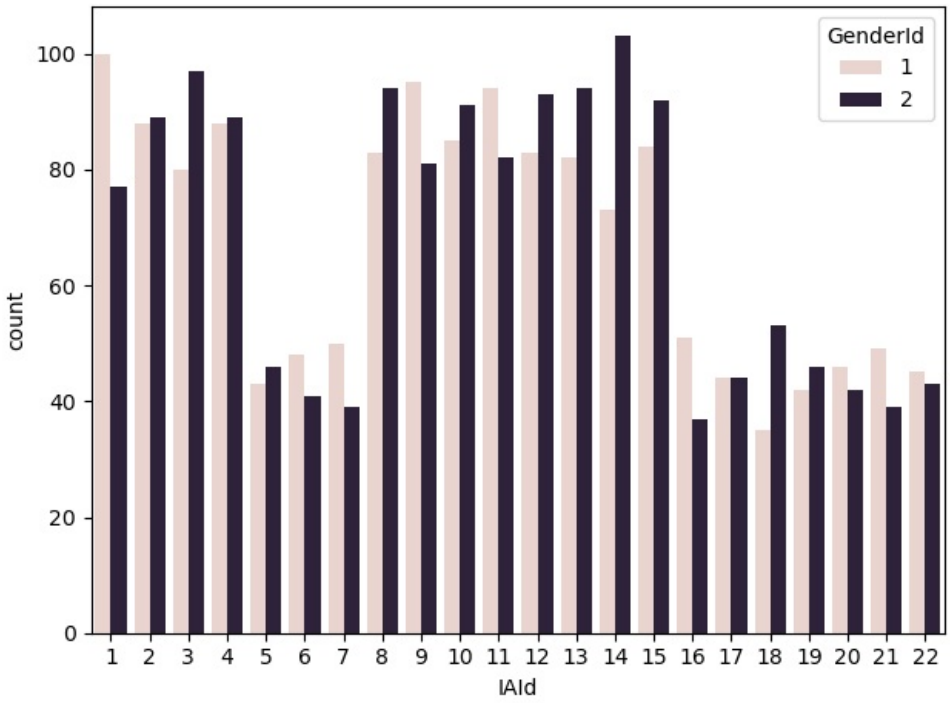

1200
1000
800
600
400
200
0
count
1
2
3
4
5
Risk Weighting

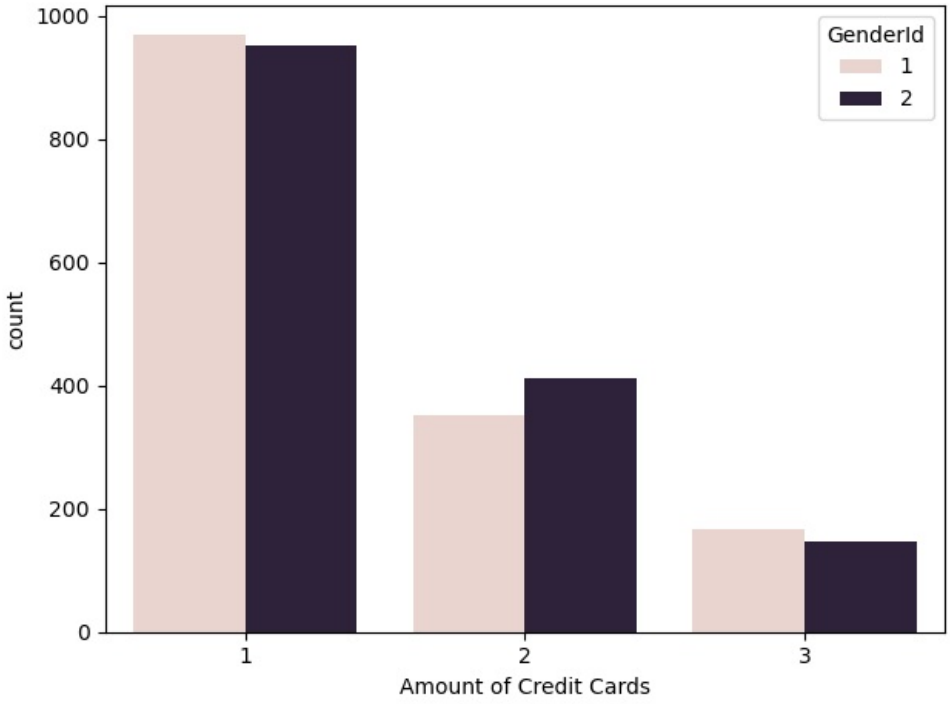


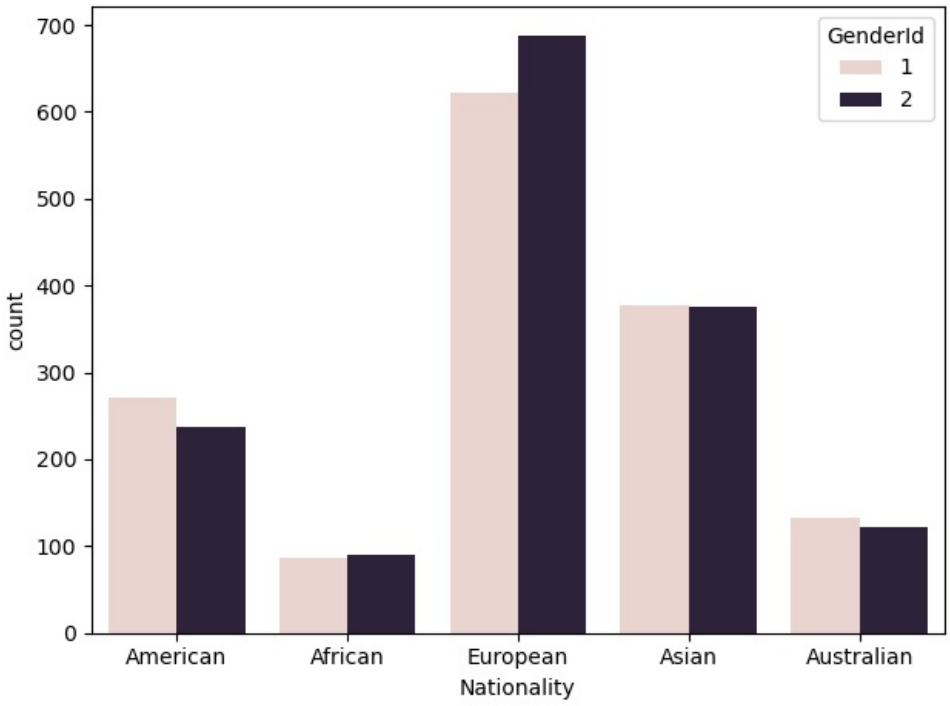
In [12]:
```
for i, predictor in enumerate(df[["BRId", "GenderId", "IAId", "Amount of Credit Cards", "Nationality", "Occupati
    plt.figure(i)
    sns.countplot(data=df, x=predictor, hue="GenderId")
    plt.tight_layout()
```

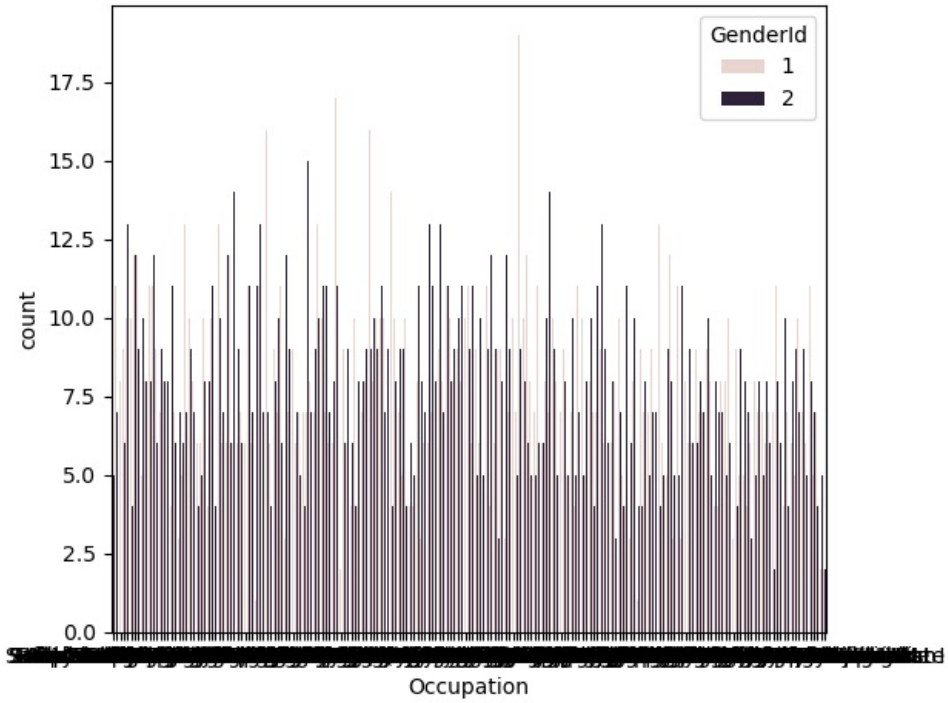


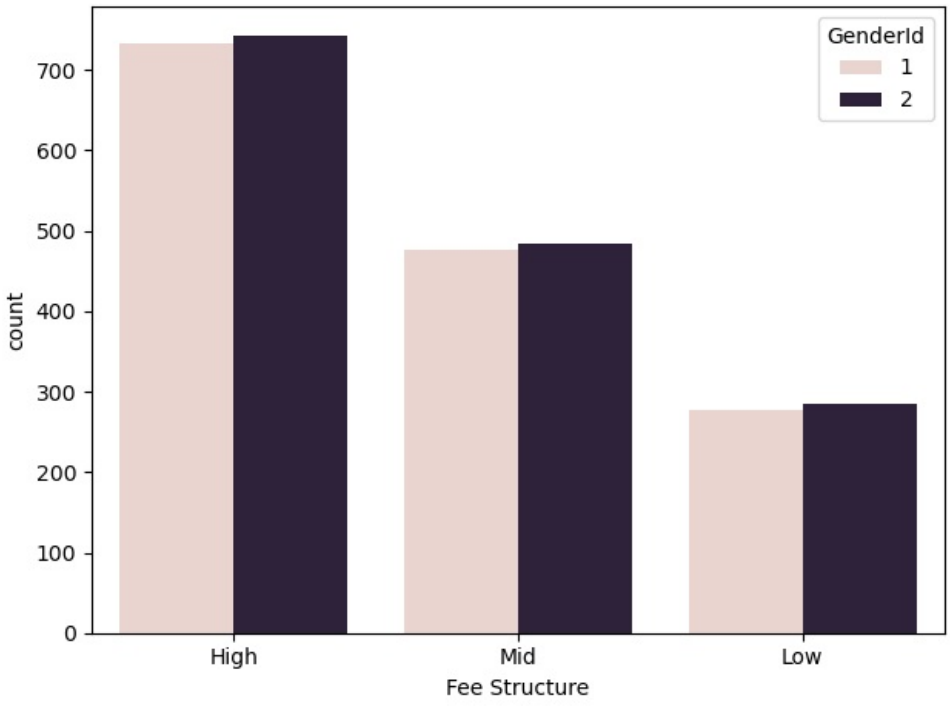

GenderId
1
2
1400
1200
1000
800
600
400
200
0
count
1
2
GenderId

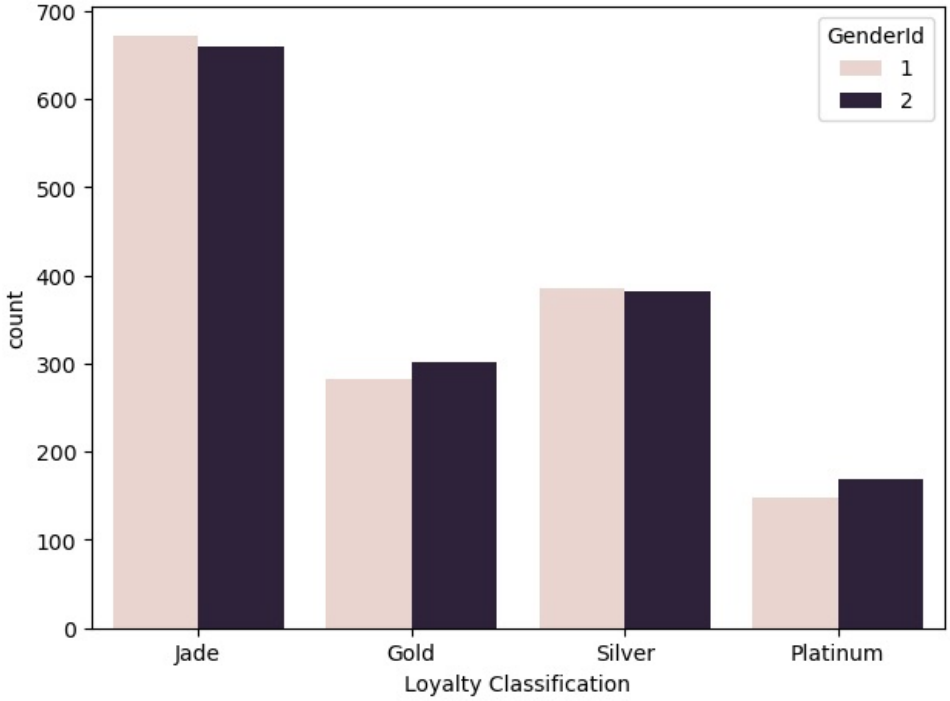

GenderId
1
2
100
80
60
40
20
0
count
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
IAId

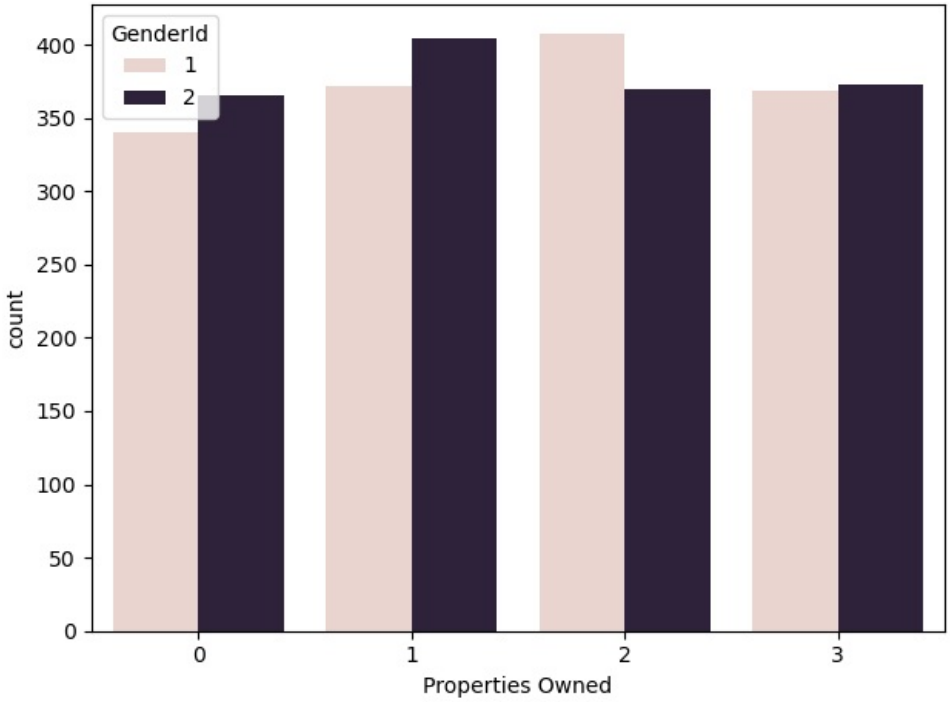

GenderId
1
2
1000
800
600
400
200
0
count
1
2
3
Amount of Credit Cards

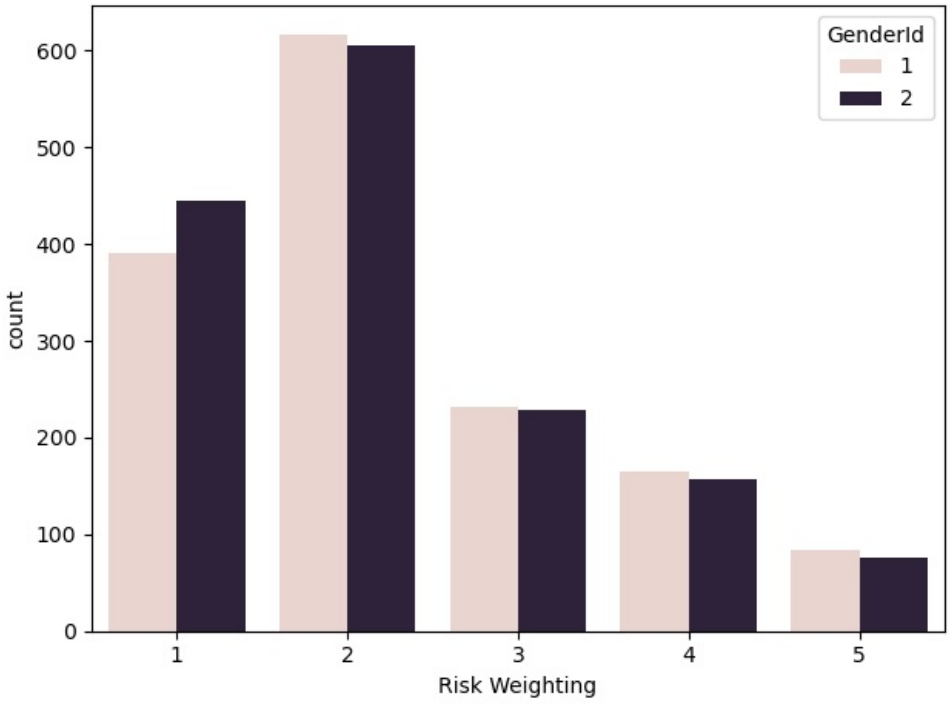

GenderId
1
2
700
600
500
400
300
200
100
0
count
American
African
European
Asian
Australian
Nationality

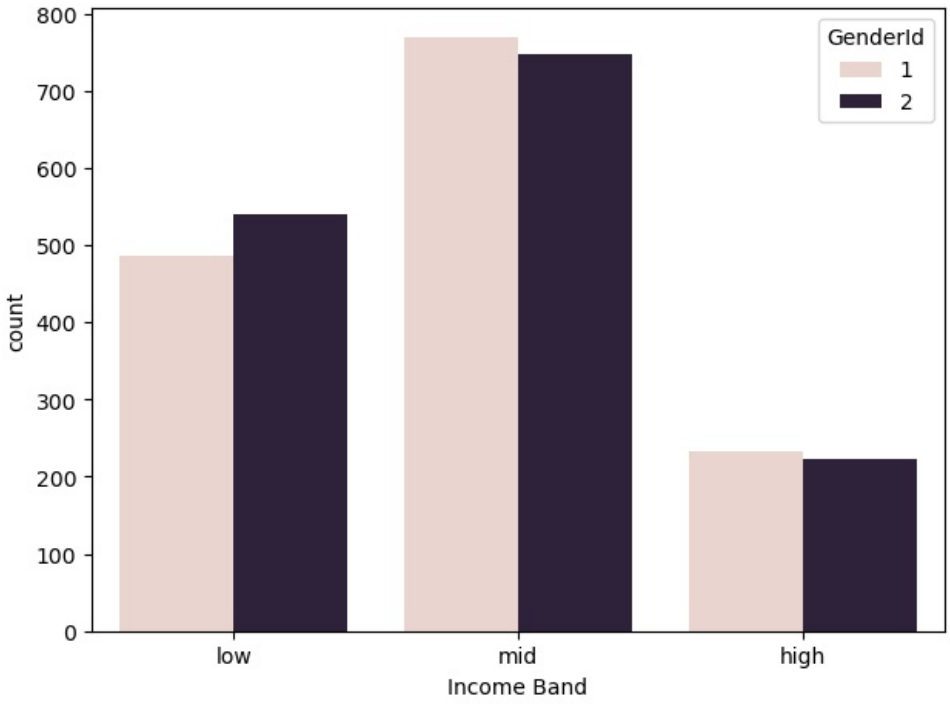

GenderId
1
2
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
count
Occupation


GenderId
1
2
700
600
500
400
300
200
100
0
count
High
Mid
Low
Fee Structure

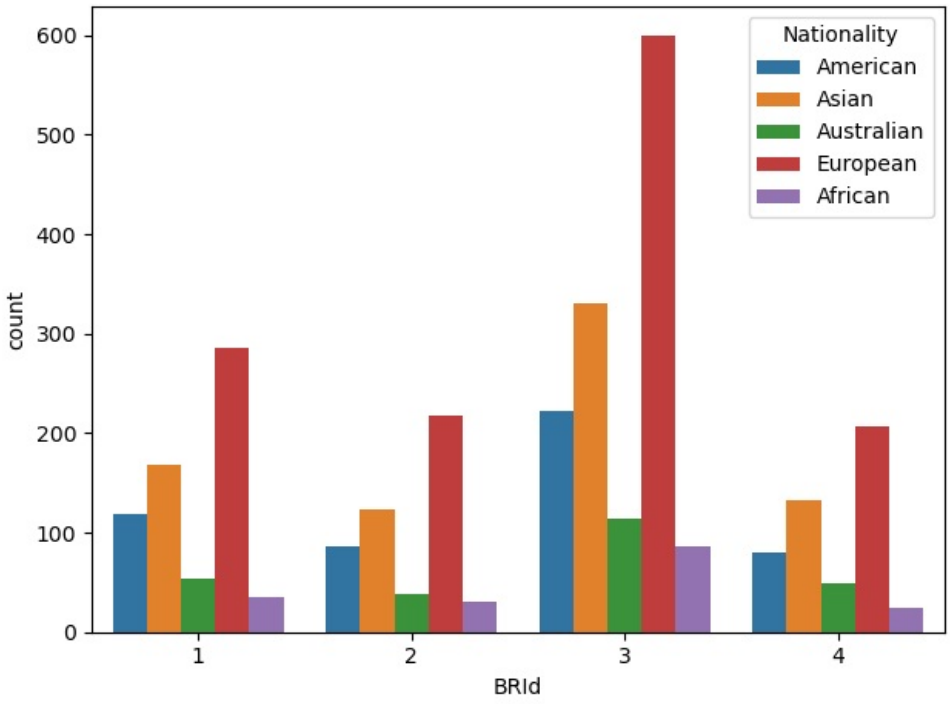

GenderId
1
2
700
600
500
400
300
200
100
0
count
Jade
Gold
Silver
Platinum
Loyalty Classification

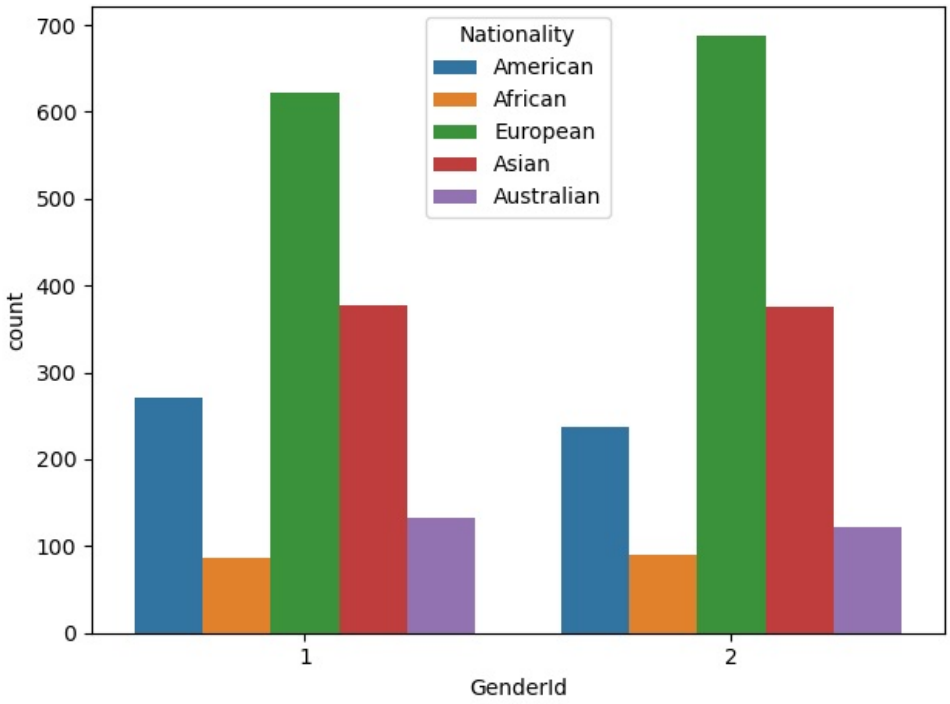

GenderId
1
2
400
350
300
250
200
150
100
50
0
count
0
1
2
3
Properties Owned

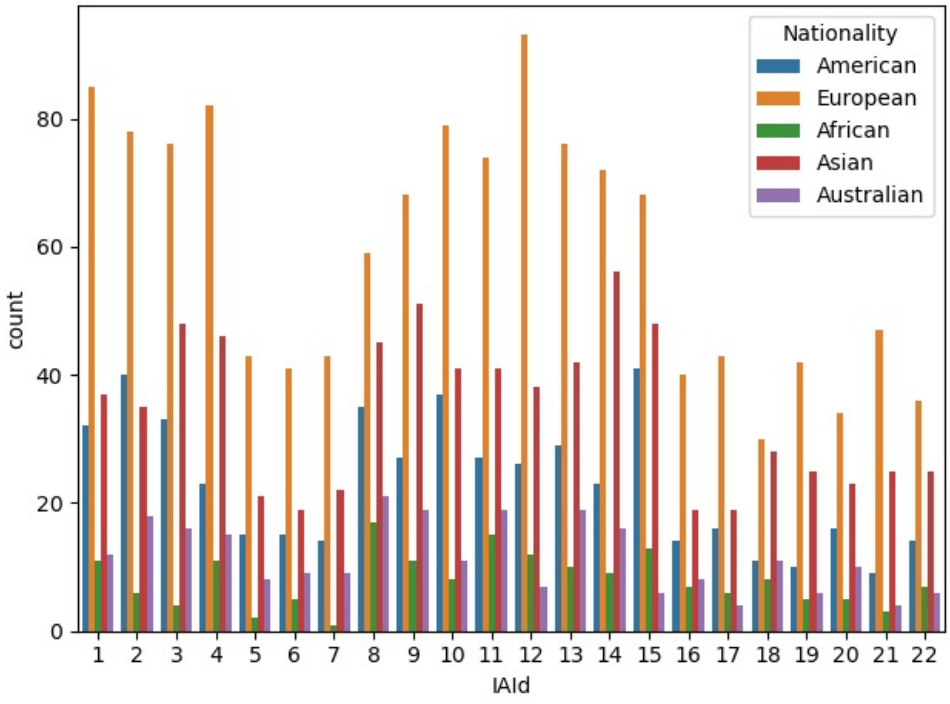


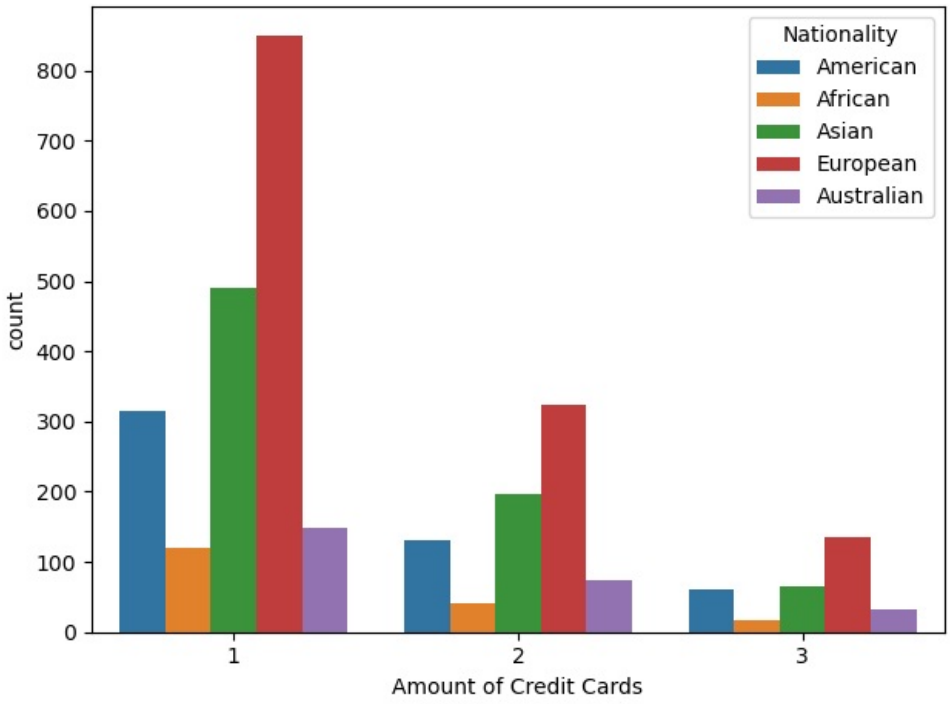


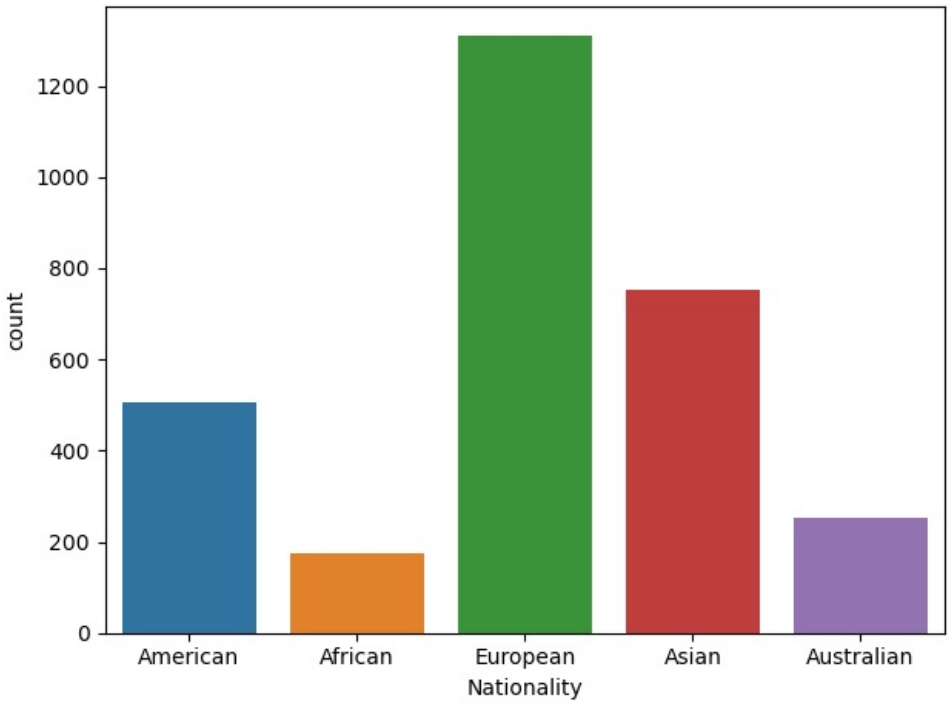
In [13]:
```
for i, predictor in enumerate(df[["BRId", "GenderId", "IAId", "Amount of Credit Cards", "Nationality", "Occupat
    plt.figure(i)
    sns.countplot(data=df, x=predictor, hue="Nationality")
    plt.tight_layout()
```

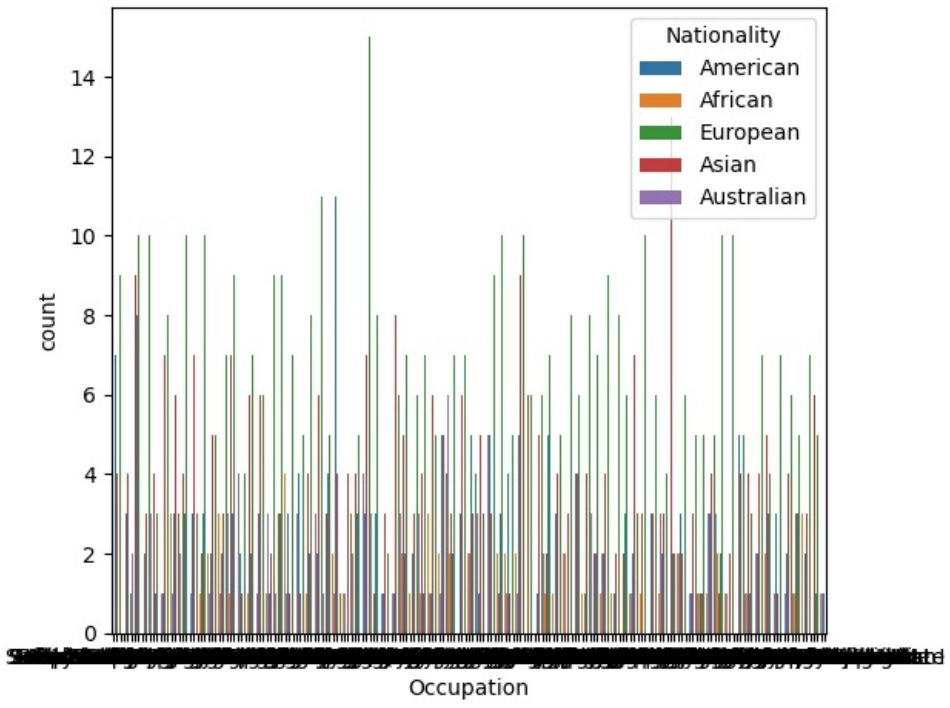

Nationality
American
Asian
Australian
European
African
count
BRId

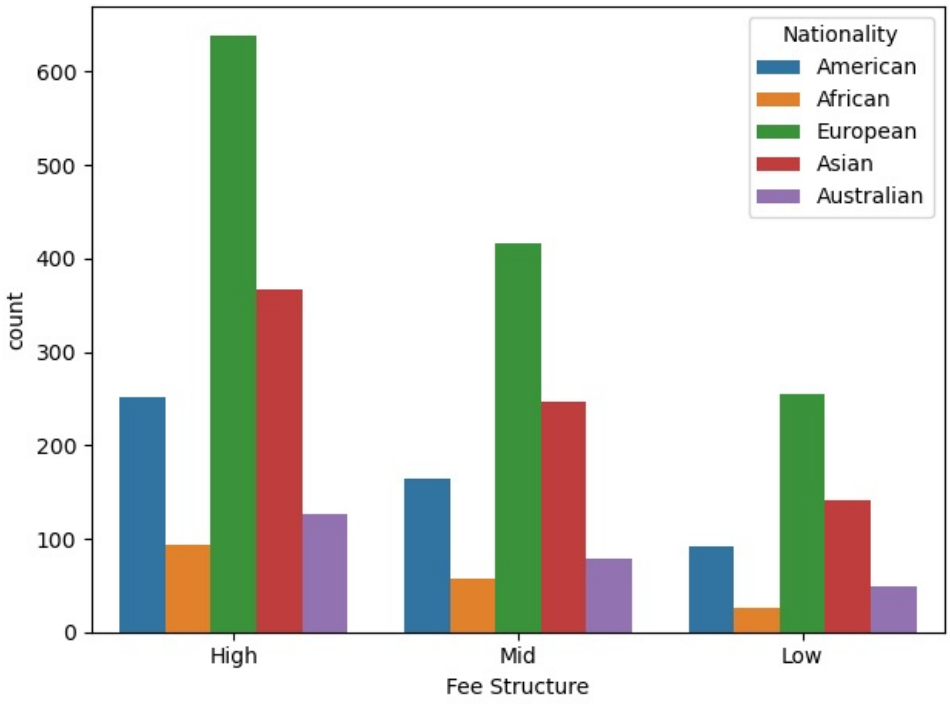

Nationality
American
African
European
Asian
Australian
count
GenderId

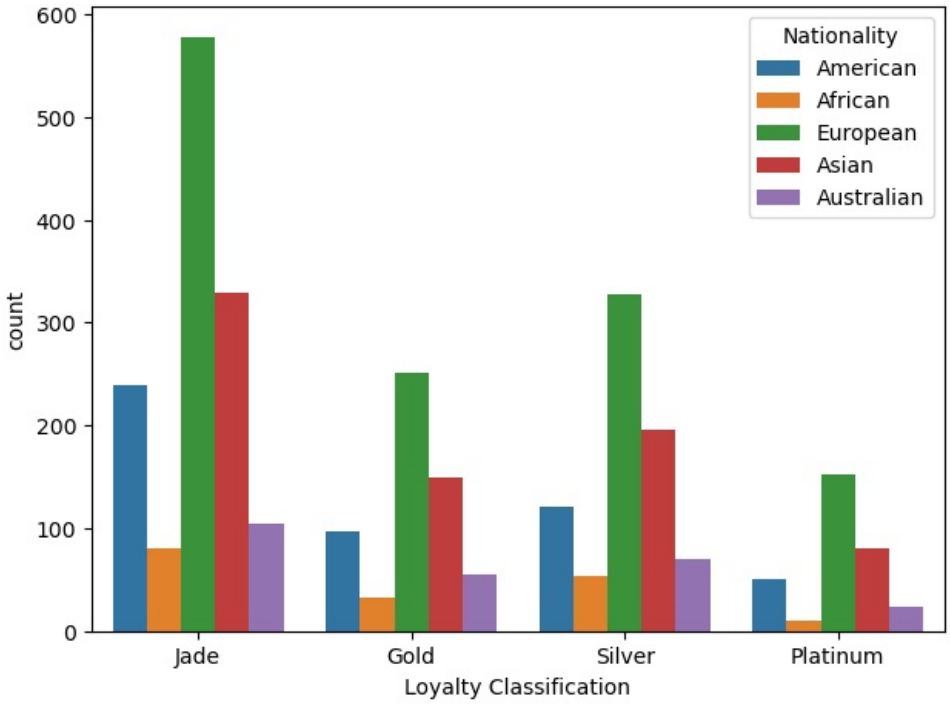

Nationality
American
European
African
Asian
Australian
count
IAId

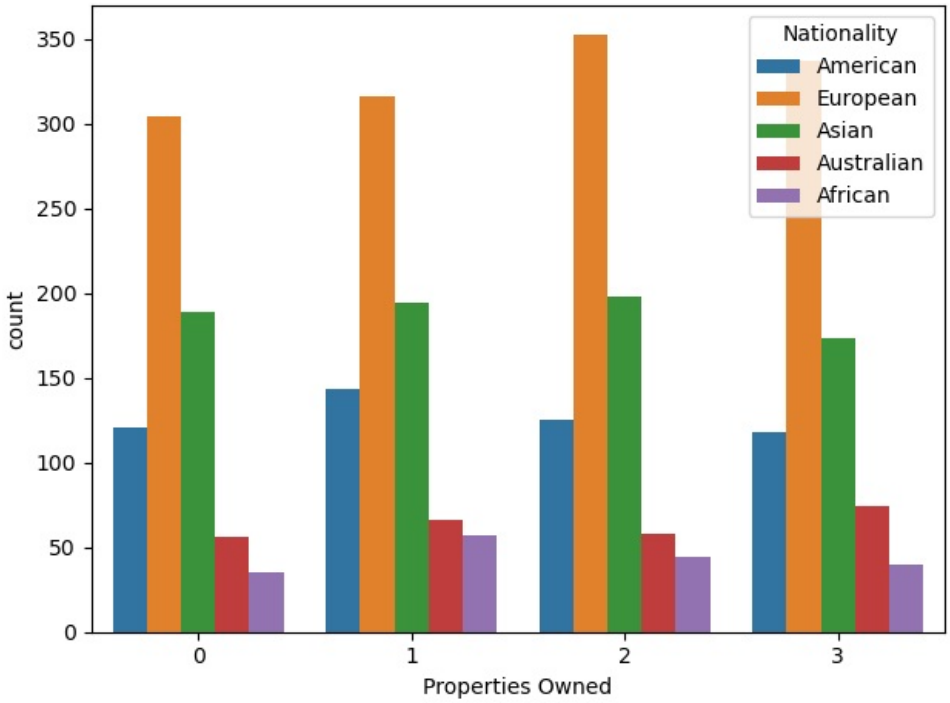

Nationality
American
African
Asian
European
Australian
count
0
100
200
300
400
500
600
700
800
1
2
3
Amount of Credit Cards

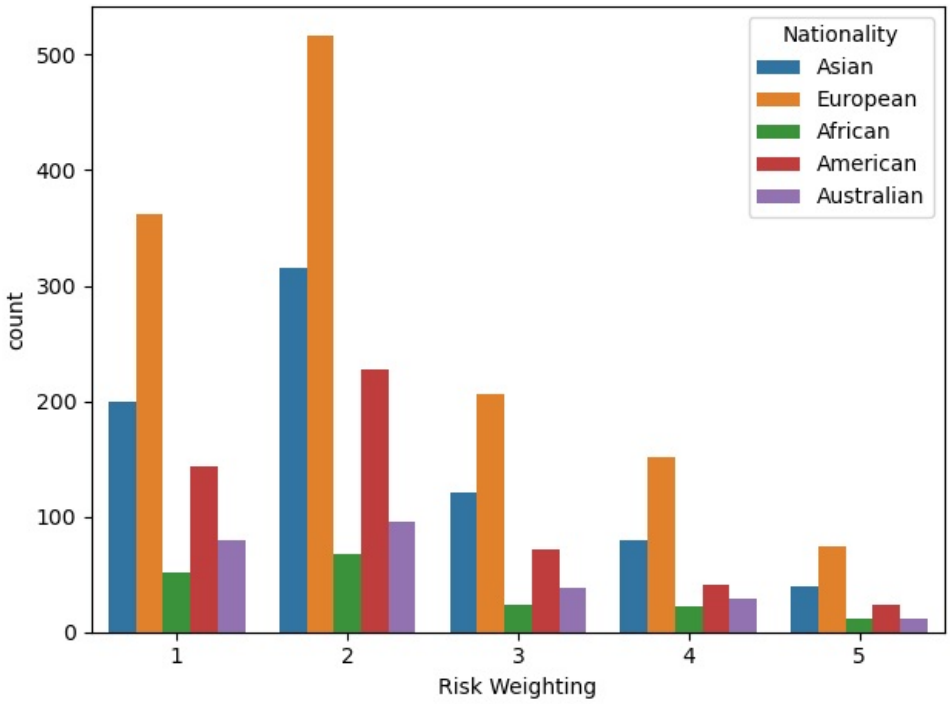

count
0
200
400
600
800
1000
1200
American
African
European
Asian
Australian
Nationality

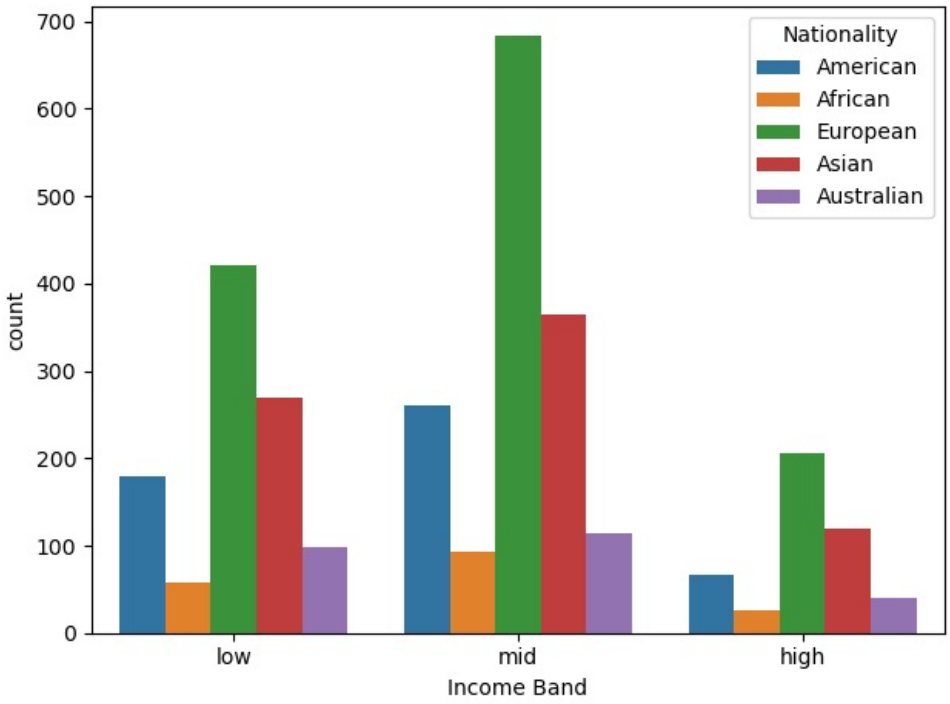

Nationality
American
African
European
Asian
Australian
count
0
2
4
6
8
10
12
14
Occupation


Nationality
American
African
European
Asian
Australian
count
0
100
200
300
400
500
600
High
Mid
Low
Fee Structure


Nationality
American
African
European
Asian
Australian
count
0
100
200
300
400
500
600
Jade
Gold
Silver
Platinum
Loyalty Classification

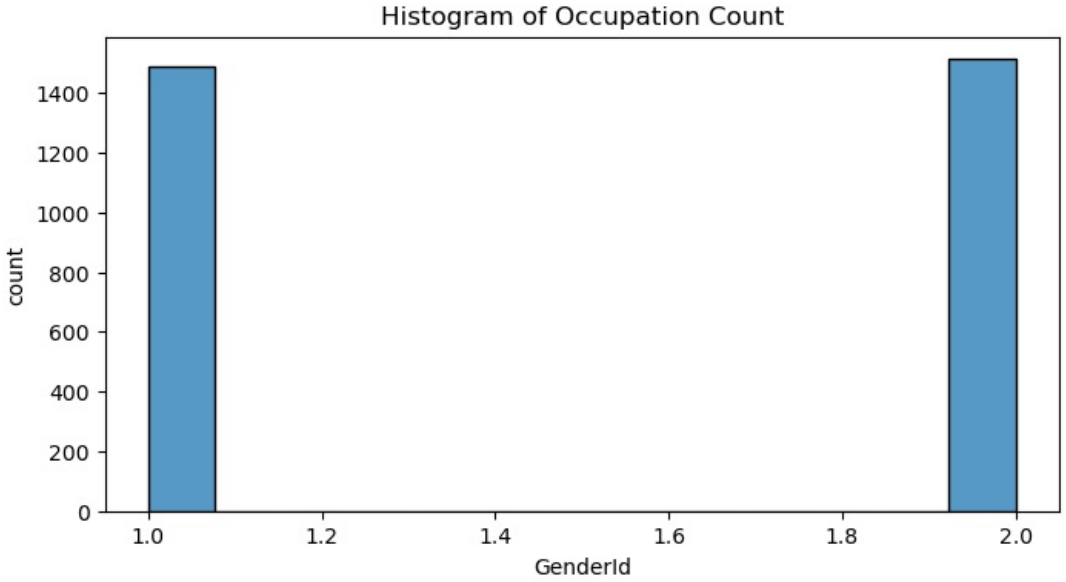

Nationality
American
European
Asian
Australian
African
count
0
50
100
150
200
250
300
350
0
1
2
3
Properties Owned

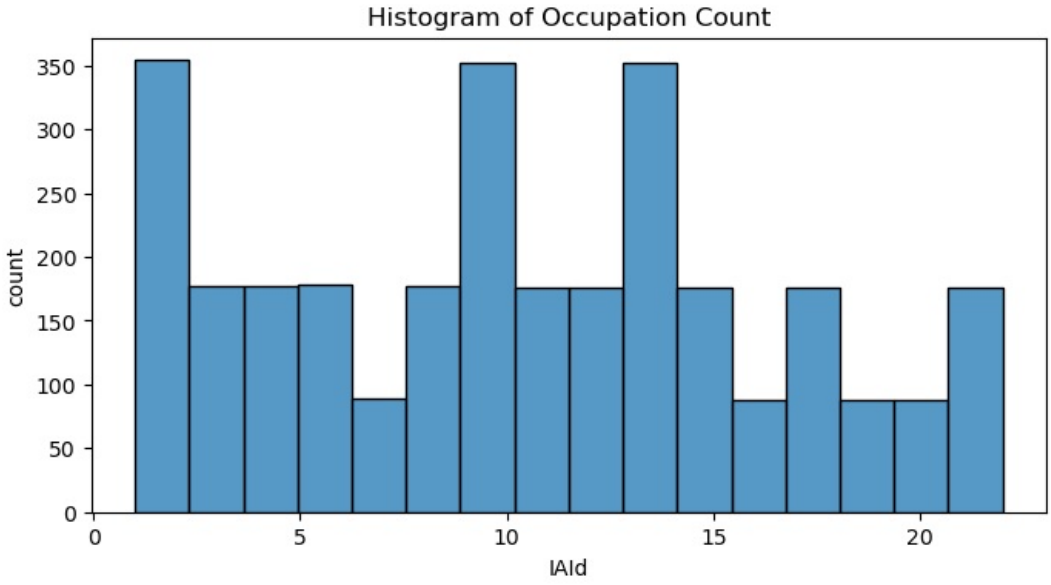


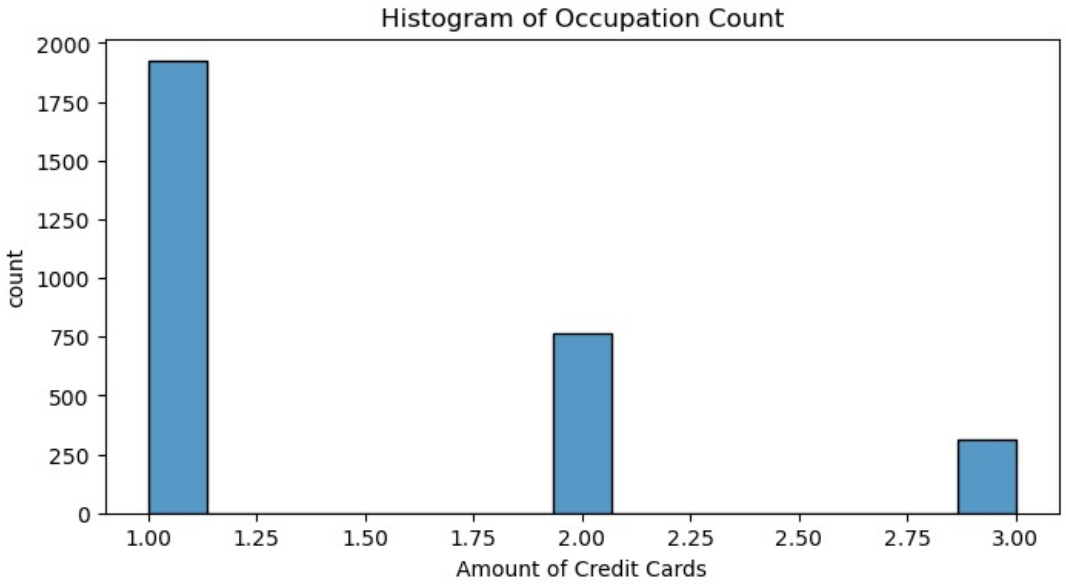


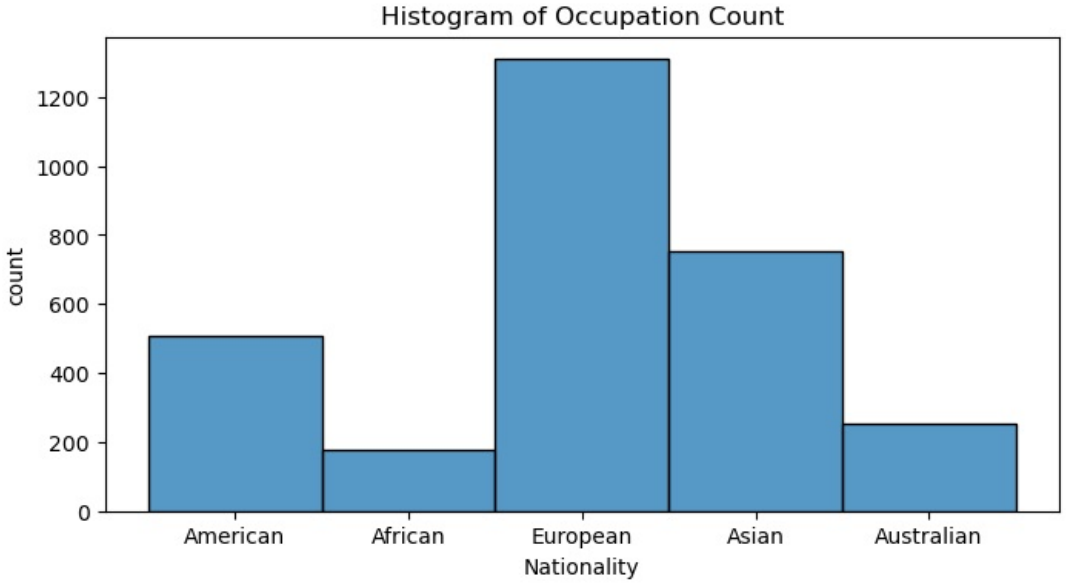
In [14]:
```python
#Histoplot of value counts for different occupation

for col in categorical_cols:
    if col == "Occupation":
        continue
    plt.figure(figsize=(8,4))
    sns.histplot(df[col])
    plt.title("Histogram of Occupation Count")
    plt.xlabel(col)
    plt.ylabel("count")
    plt.show()
```

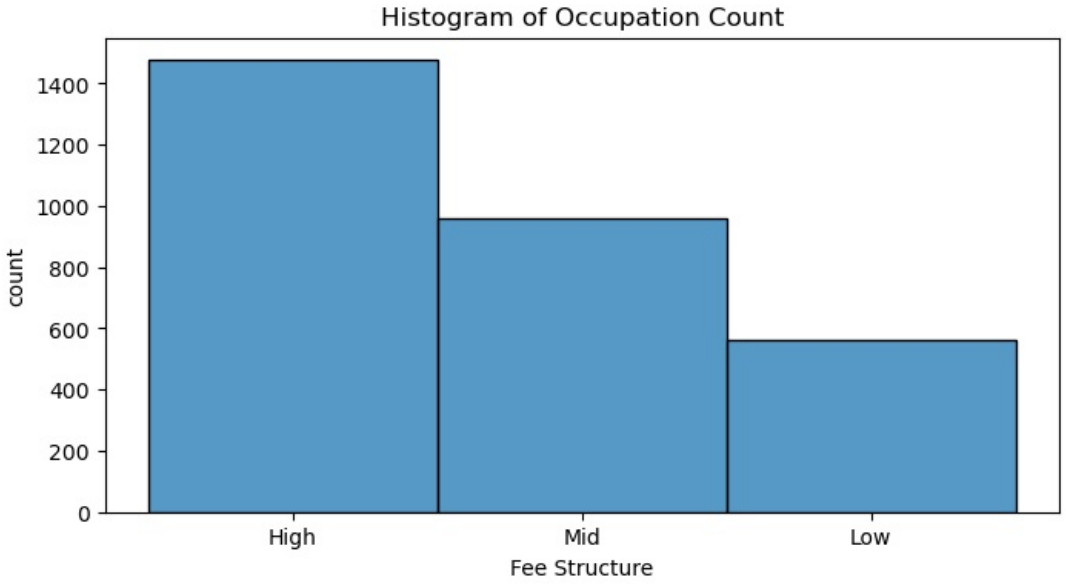

Histogram of Occupation Count
count
BRId
0
200
400
600
800
1000
1200
1400
1.0
1.5
2.0
2.5
3.0
3.5
4.0

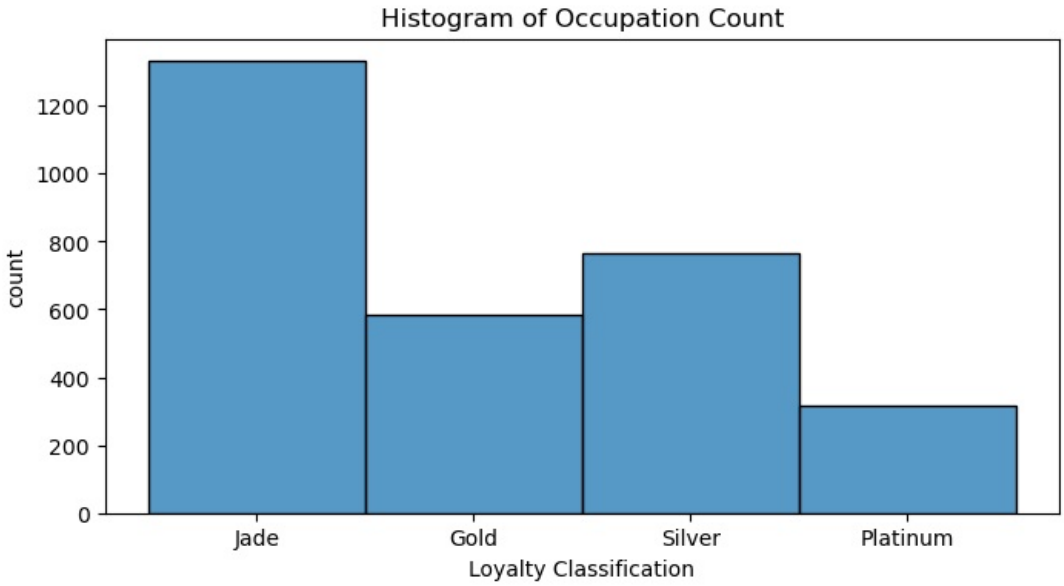

Histogram of Occupation Count
count
GenderId
0
200
400
600
800
1000
1200
1400
1.0
1.2
1.4
1.6
1.8
2.0

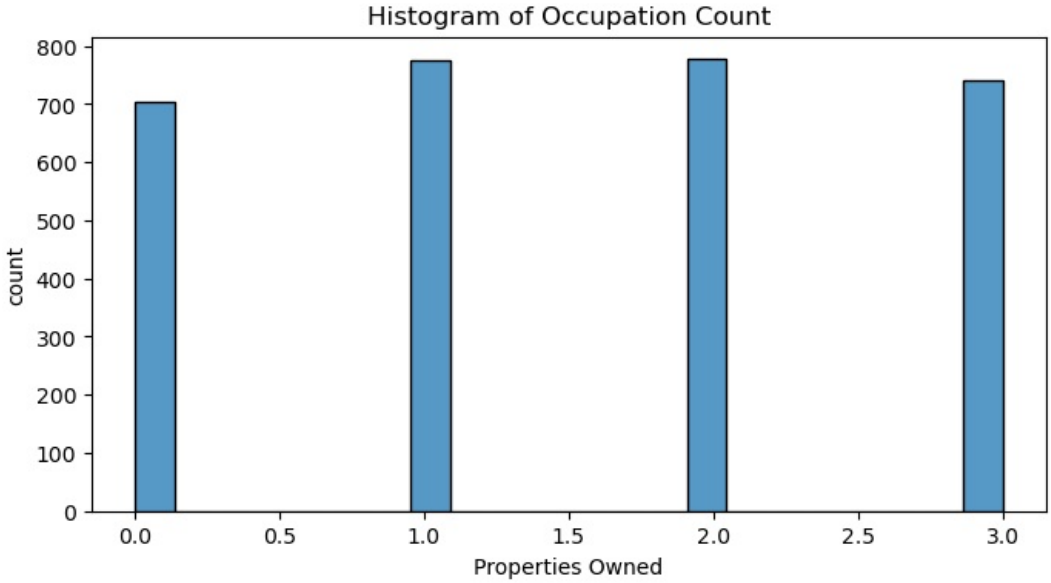

Histogram of Occupation Count
count
IAId
0
50
100
150
200
250
300
350
0
5
10
15
20

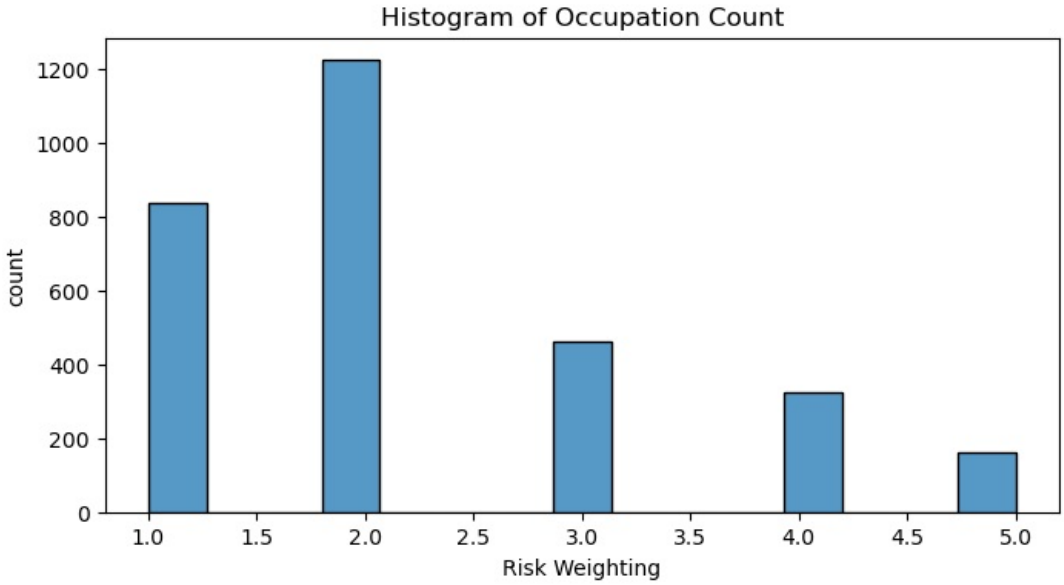

Histogram of Occupation Count
count
Amount of Credit Cards
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00

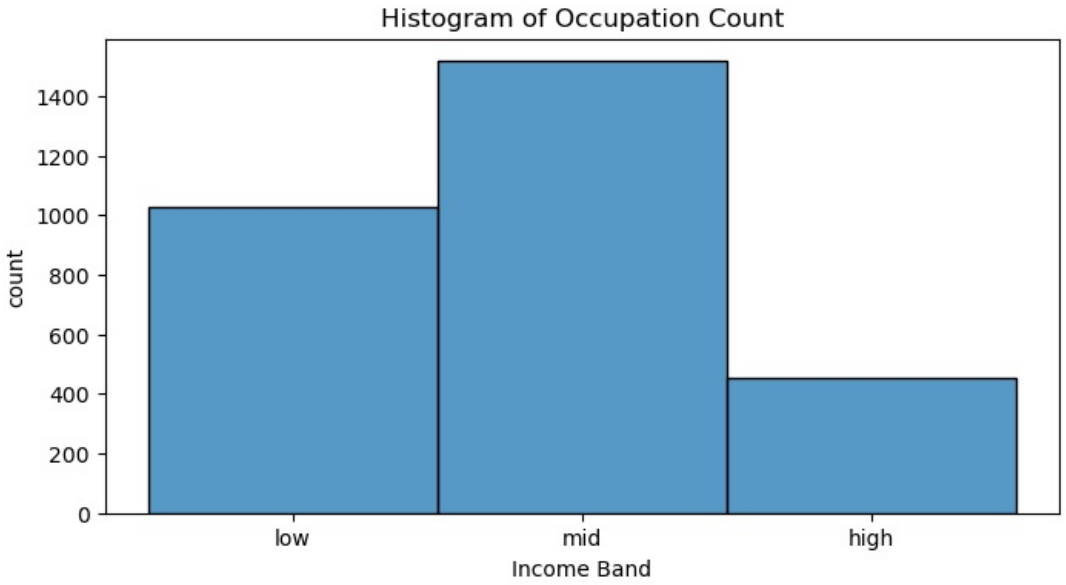

Histogram of Occupation Count
count
Nationality
American
African
European
Asian
Australian


Histogram of Occupation Count
count
Fee Structure
High
Mid
Low

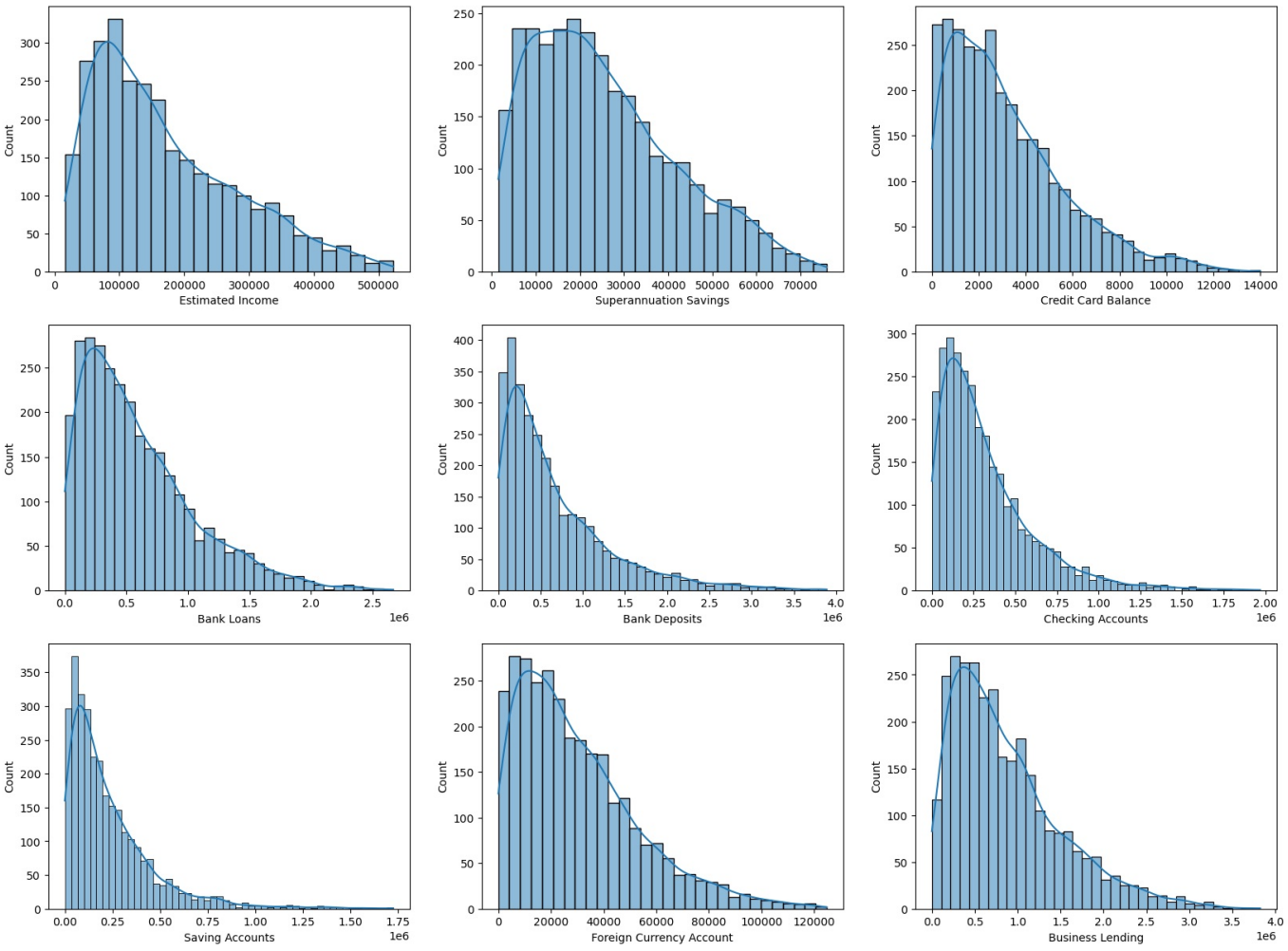

Histogram of Occupation Count
count
0
200
400
600
800
1000
1200
Jade
Gold
Silver
Platinum
Loyalty Classification


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


Histogram of Occupation Count
count
0
200
400
600
800
1000
1200
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Risk Weighting



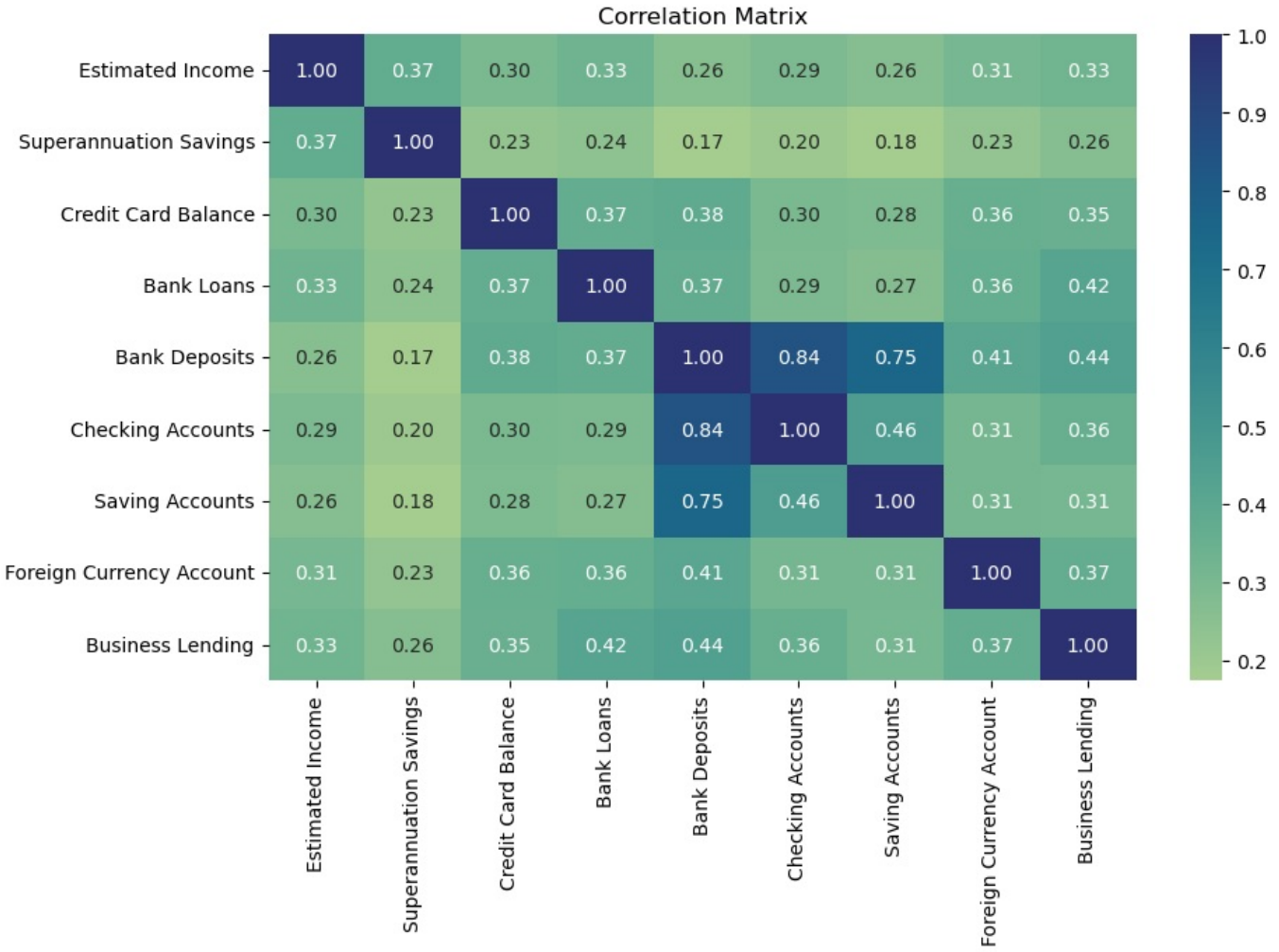
In [46]:
```
#Numerical Analysis
numerical_cols = ['Estimated Income', 'Superannuation Savings','Credit Card Balance', 'Bank Loans', 'Bank Depos

#Univariate Analysis and Visualization
plt.figure(figsize=(20,20))
for i,col in enumerate(numerical_cols):
    plt.subplot(4,3,i+1)
    sns.histplot(df[col],kde=True)
    #plt.title(col)
plt.show()
plt.tight_layout()
```



<Figure size 640x480 with 0 Axes>

In [56]:
```
numerical_cols = ['Estimated Income', 'Superannuation Savings','Credit Card Balance', 'Bank Loans', 'Bank Depos

correlation_matrix = df[numerical_cols].corr()

plt.figure(figsize=(10,6))
sns.heatmap(correlation_matrix, annot = True, cmap = 'crest', fmt = '.2f')
plt.title("Correlation Matrix")
```

```
plt.show()
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js