

```
In [2]: pip install mysql-connector-python
```

```
Requirement already satisfied: mysql-connector-python in c:\users\rajes\anaconda3\lib
\site-packages (8.0.33)
Requirement already satisfied: protobuf<=3.20.3,>=3.11.0 in c:\users\rajes\anaconda3
\lib\site-packages (from mysql-connector-python) (3.20.3)
Note: you may need to restart the kernel to use updated packages.
```

```
In [2]: #Importing all the necessary Libraries
```

```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
# Connect to server
cnx = mysql.connector.connect(
    host="127.0.0.1",
    port=3306,
    user="root",
    password="password")
```

```
In [3]: #Extracting data from the dataset with the use of sql
```

```
q = "select * from banking_case.customer"
df = pd.read_sql(q, cnx)
print(df)
```

	Client ID	Name	Age	Location ID	Joined Bank	\
0	IND81288	Raymond Mills	24	34324	06-05-2019	
1	IND65833	Julia Spencer	23	42205	10-12-2001	
2	IND47499	Stephen Murray	27	7314	25-01-2010	
3	IND72498	Virginia Garza	40	34594	28-03-2019	
4	IND60181	Melissa Sanders	46	41269	20-07-2012	
...	
2995	IND66827	Earl Hall	82	8760	09-10-2014	
2996	IND40556	Billy Williamson	44	32837	05-02-2009	
2997	IND72414	Victor Black	70	36088	29-12-2009	
2998	IND46652	Andrew Ford	56	24871	13-02-2006	
2999	IND40216	Amy Nguyen	79	38518	08-12-2005	

	Banking Contact	Nationality	Occupation	\
0	Anthony Torres	American	Safety Technician IV	
1	Jonathan Hawkins	African	Software Consultant	
2	Anthony Berry	European	Help Desk Operator	
3	Steve Diaz	American	Geologist II	
4	Shawn Long	American	Assistant Professor	
...	
2995	Joshua Bennett	American	Accounting Assistant III	
2996	Dennis Ruiz	European	Paralegal	
2997	Joshua Ryan	American	Statistician IV	
2998	Nicholas Cunningham	European	Human Resources Assistant III	
2999	Joe Hanson	American	Biostatistician III	

	Fee Structure	Loyalty	Classification	...	Bank Deposits	\
0	High	Jade	...	1485828.64		
1	High	Jade	...	641482.79		
2	High	Gold	...	1033401.59		
3	Mid	Silver	...	1048157.49		
4	Mid	Platinum	...	487782.53		
...		
2995	High	Gold	...	1089957.03		
2996	Mid	Gold	...	136891.32		
2997	Low	Jade	...	214860.89		
2998	Mid	Jade	...	742630.22		
2999	High	Jade	...	65617.66		

	Checking Accounts	Saving Accounts	Foreign Currency Account	\
0	603617.88	607332.46	12249.96	
1	229521.37	344635.16	61162.31	
2	652674.69	203054.35	79071.78	
3	1048157.49	234685.02	57513.65	
4	446644.25	128351.45	30012.14	
...	
2995	532867.88	657849.62	12947.31	
2996	56581.74	93195.61	23205.69	
2997	158726.06	35539.15	30291.81	
2998	404638.26	56411.33	6413.14	
2999	77769.08	32371.38	8992.36	

	Business Lending	Properties Owned	Risk Weighting	BRIId	GenderId	IAId
0	1134475.30	1	2	1	1	1
1	2000526.10	1	3	2	1	2
2	548137.58	1	3	3	2	3
3	1148402.29	0	4	4	1	4
4	1674412.12	0	3	1	2	5
...
2995	1238859.91	1	3	3	2	4

2996	277171.07	1	2	3	2	5
2997	502947.22	2	2	3	2	6
2998	1538368.60	3	1	3	2	7
2999	329412.55	1	1	3	2	8

[3000 rows x 25 columns]

```
C:\Users\rajes\anaconda3\lib\site-packages\pandas\io\sql.py:762: UserWarning: pandas
only support SQLAlchemy connectable(engine/connection) or database string URI or sqlite
DBAPI2 connection other DBAPI2 objects are not tested, please consider using SQLAlchemy
warnings.warn(
```

In [4]: # Displaying the first five rows to understand the structure of the dataset
df.head(5)

Out[4]:

	Client ID	Name	Age	Location ID	Joined Bank	Banking Contact	Nationality	Occupation	Fee Structure	Lo
0	IND81288	Raymond Mills	24	34324	06-05-2019	Anthony Torres	American	Safety Technician IV	High	
1	IND65833	Julia Spencer	23	42205	10-12-2001	Jonathan Hawkins	African	Software Consultant	High	
2	IND47499	Stephen Murray	27	7314	25-01-2010	Anthony Berry	European	Help Desk Operator	High	
3	IND72498	Virginia Garza	40	34594	28-03-2019	Steve Diaz	American	Geologist II	Mid	
4	IND60181	Melissa Sanders	46	41269	20-07-2012	Shawn Long	American	Assistant Professor	Mid	Plat

5 rows x 25 columns

In [5]: df.shape

Out[5]: (3000, 25)

In [6]: # Checking basic information about the dataset
df.info()

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 25 columns):
 #   Column           Non-Null Count Dtype  
--- 
 0   i»;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 [7]: `# Generating basic statistical summaries for numerical columns
df.describe()`

Out[7]:

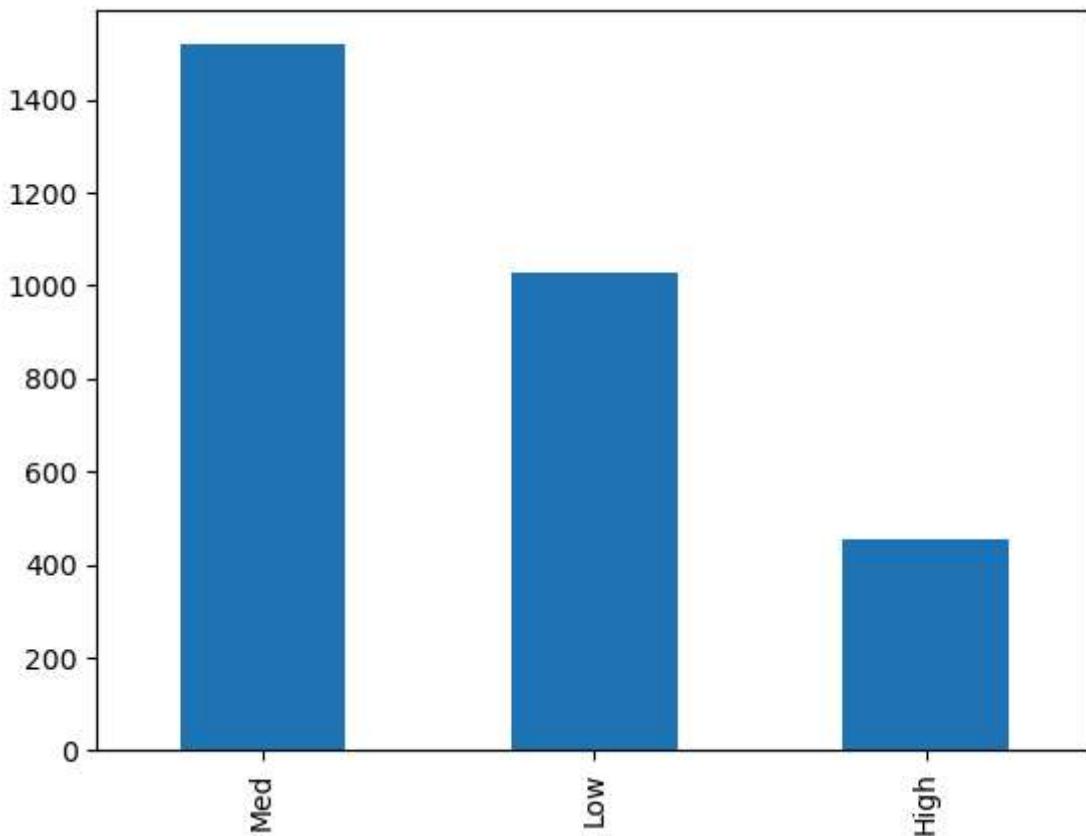
	Age	Location ID	Estimated Income	Superannuation Savings	Amount of Credit Cards	Credit Card Balance	Bank
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3.000000e+000
mean	51.039667	21563.323000	171305.034263	25531.599673	1.463667	3176.206943	5.913862
std	19.854760	12462.273017	111935.808209	16259.950770	0.676387	2497.094709	4.575570
min	17.000000	12.000000	15919.480000	1482.030000	1.000000	1.170000	0.000000
25%	34.000000	10803.500000	82906.595000	12513.775000	1.000000	1236.630000	2.396281
50%	51.000000	21129.500000	142313.480000	22357.355000	1.000000	2560.805000	4.797934
75%	69.000000	32054.500000	242290.305000	35464.740000	2.000000	4522.632500	8.258130
max	85.000000	43369.000000	522330.260000	75963.900000	3.000000	13991.990000	2.667557

In [8]: `bins = [0, 100000, 300000, float('inf')]
labels = ['Low', 'Med', 'High']`

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

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

Out[8]: <AxesSubplot:>



In [9]: categorical_cols = df[['BRIId', 'GenderId', 'IAId', "Amount of Credit Cards", "Nationality"]]

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

Value Counts for 'BRIId':

```
3    1352  
1     660  
2     495  
4     493
```

Name: BRIId, dtype: int64

Value Counts for 'GenderId':

```
2    1512  
1    1488
```

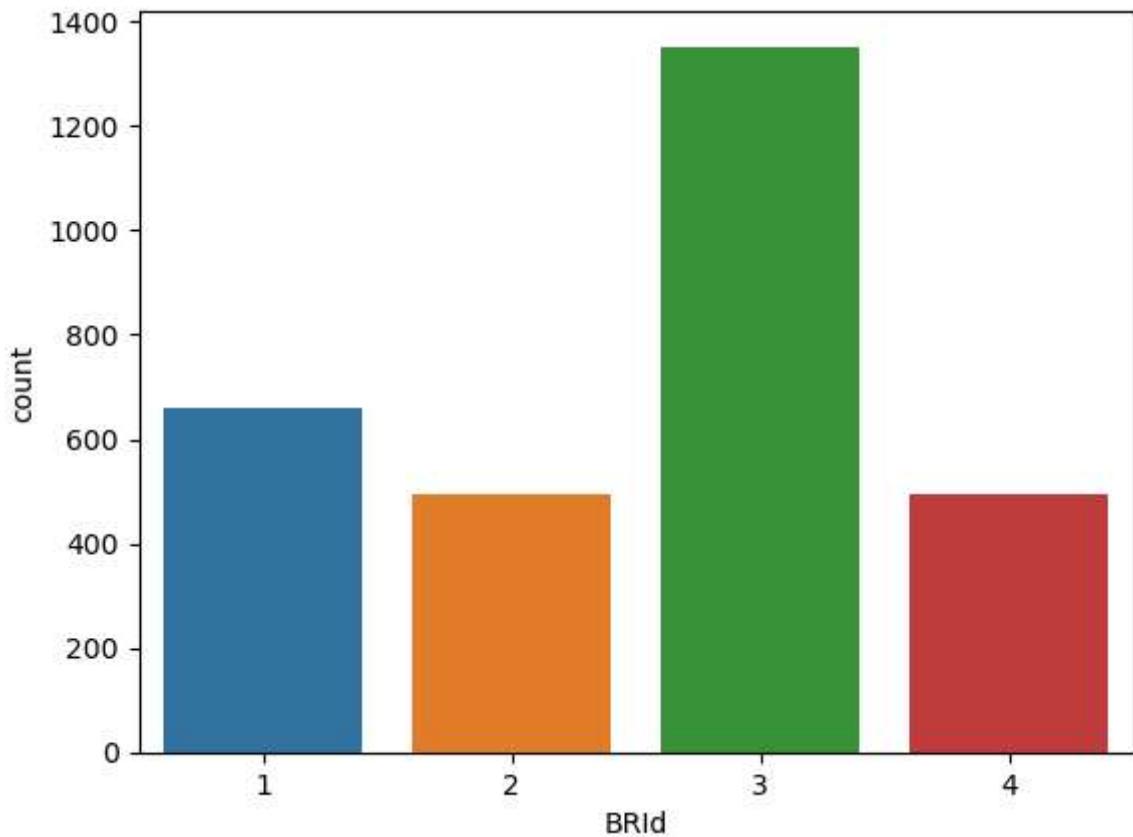
Name: GenderId, dtype: int64

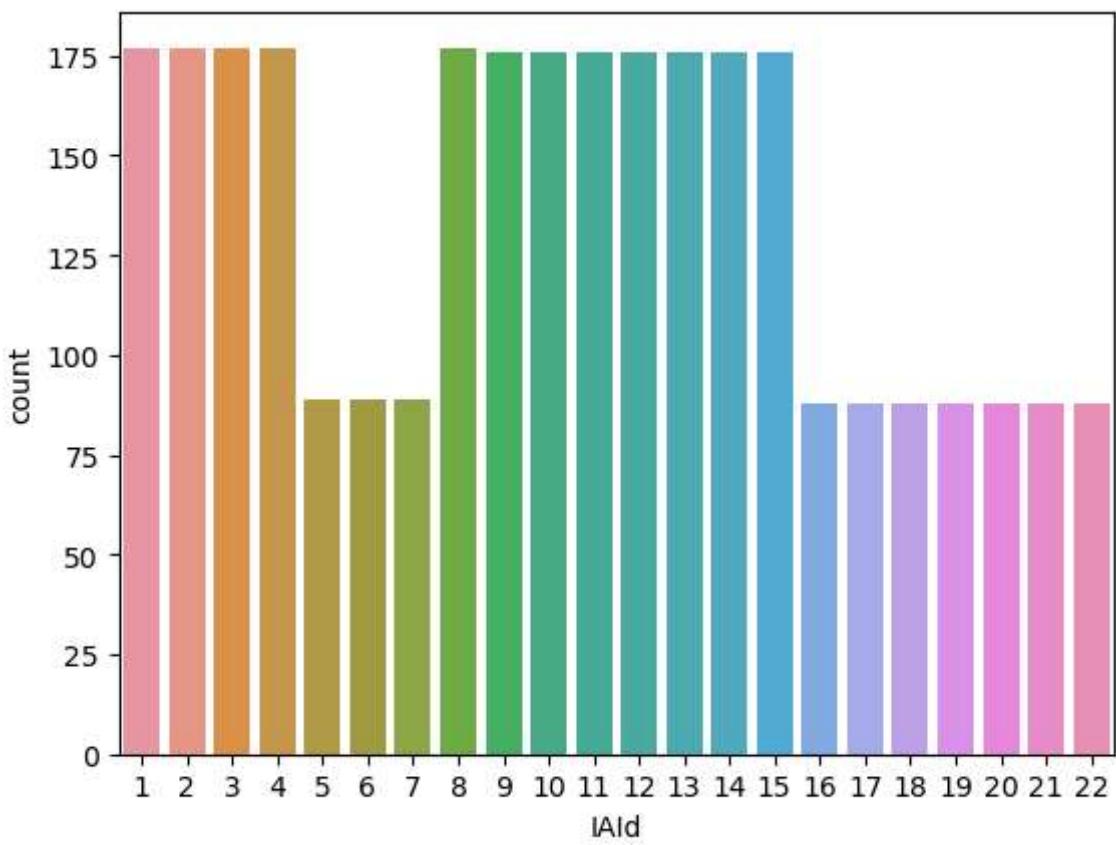
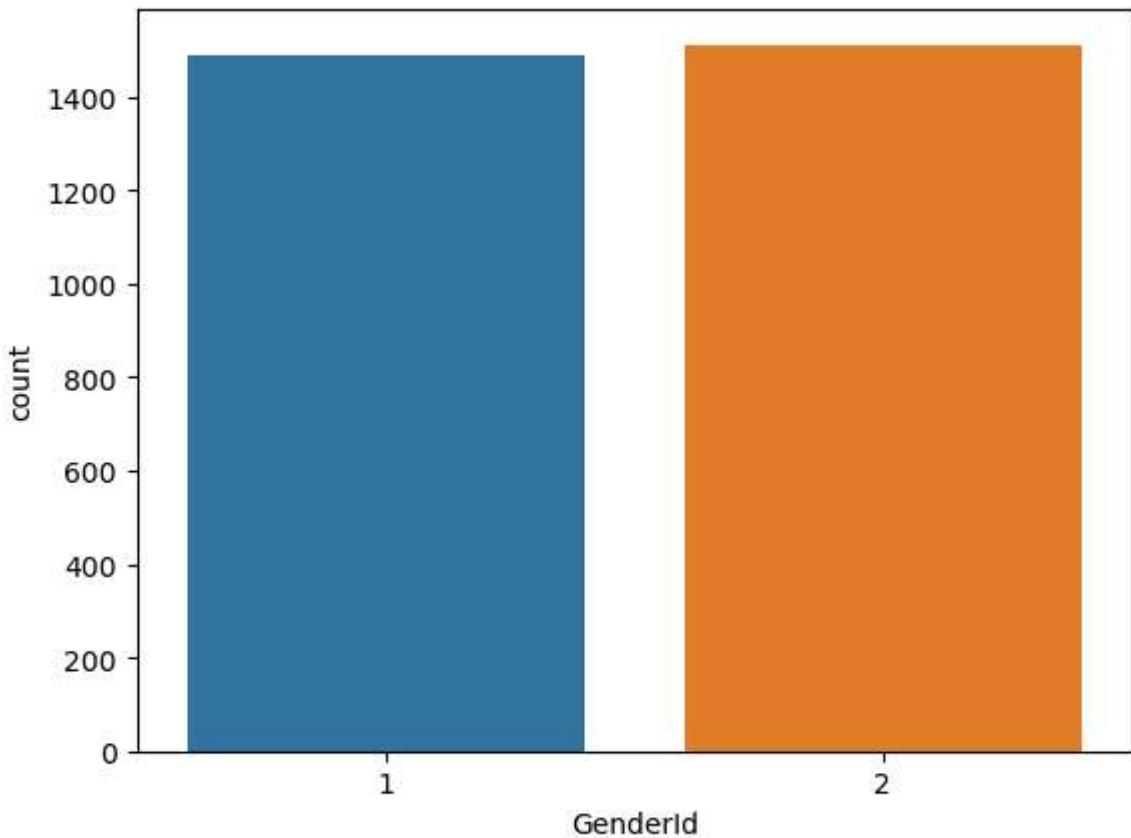
Value Counts for 'IAId':

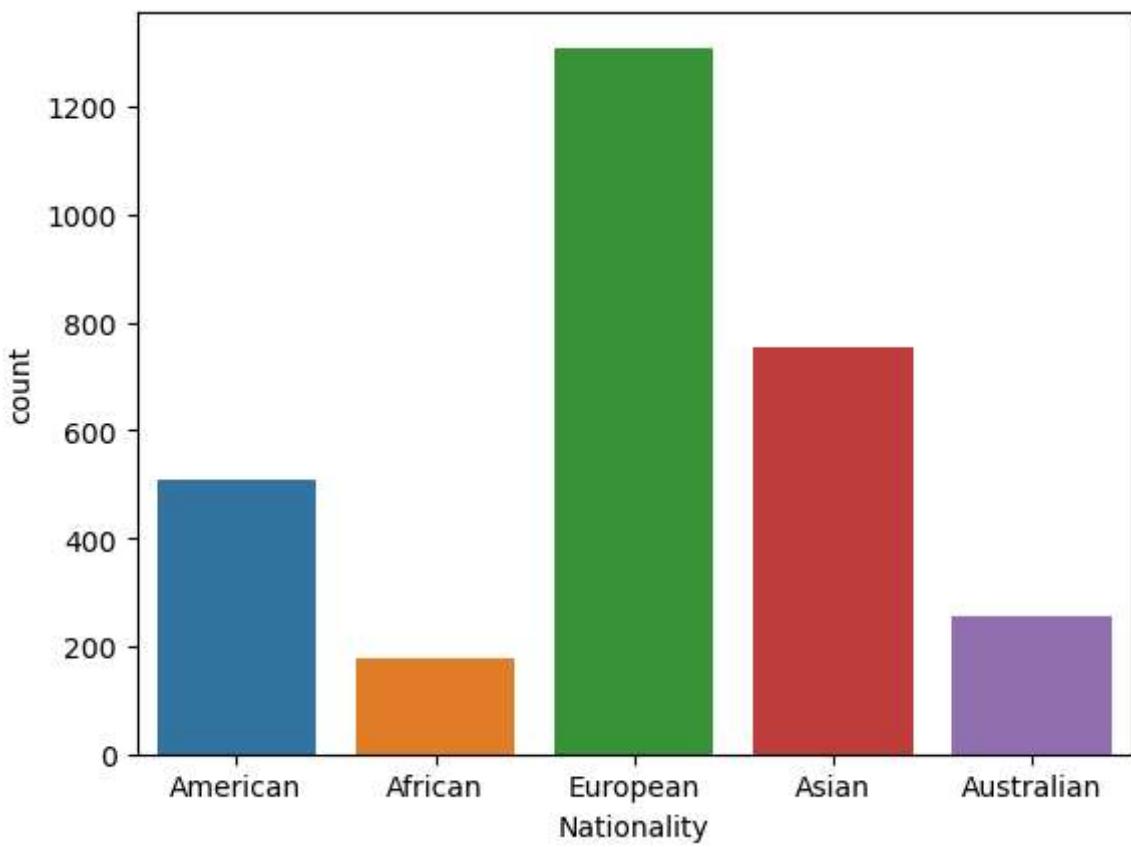
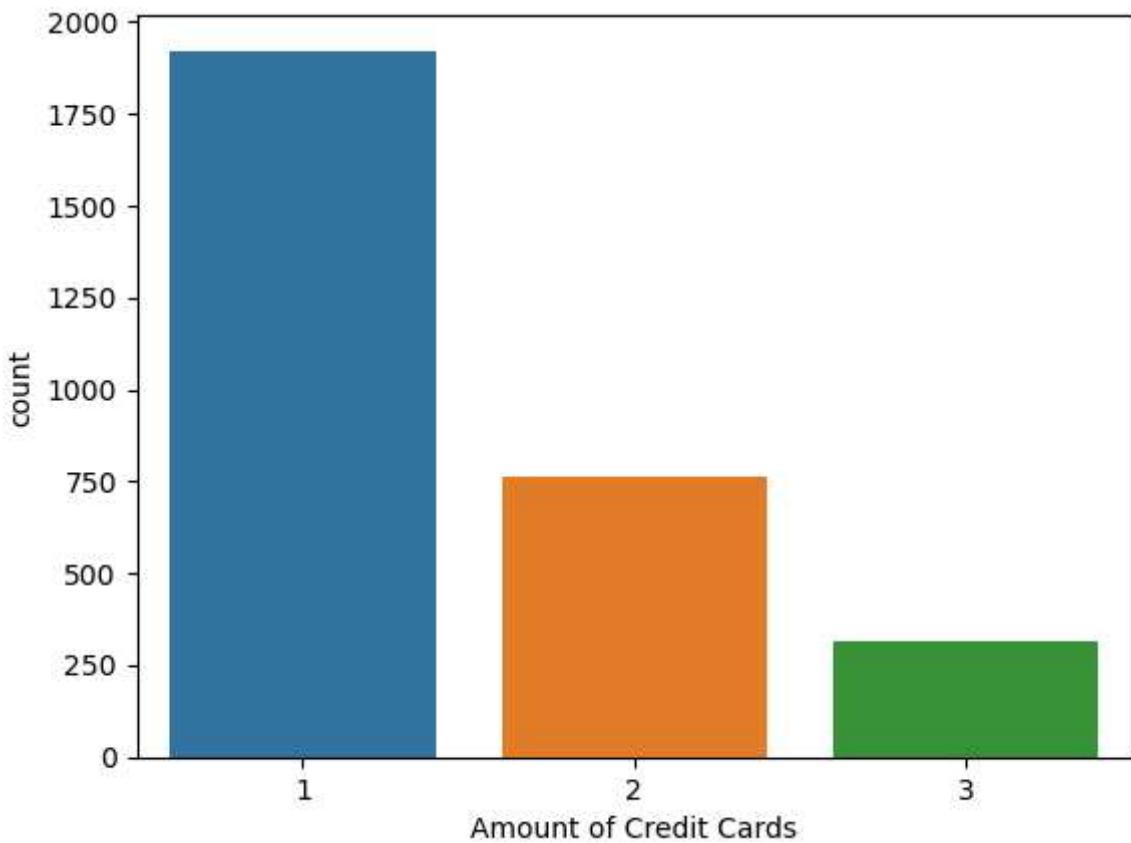
```
1    177
3    177
4    177
8    177
2    177
11   176
15   176
14   176
13   176
12   176
10   176
9    176
7    89
6    89
5    89
16   88
17   88
18   88
19   88
20   88
21   88
22   88
Name: IAId, dtype: int64
Value Counts for 'Amount of Credit Cards':
1    1922
2    765
3    313
Name: Amount of Credit Cards, dtype: int64
Value Counts for 'Nationality':
European      1309
Asian         754
American      507
Australian    254
African        176
Name: Nationality, dtype: int64
Value Counts for '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: Occupation, Length: 195, dtype: int64
Value Counts for 'Fee Structure':
High      1476
Mid       962
Low       562
Name: Fee Structure, dtype: int64
Value Counts for 'Loyalty Classification':
Jade      1331
Silver     767
Gold       585
Platinum   317
Name: Loyalty Classification, dtype: int64
Value Counts for 'Properties Owned':
```

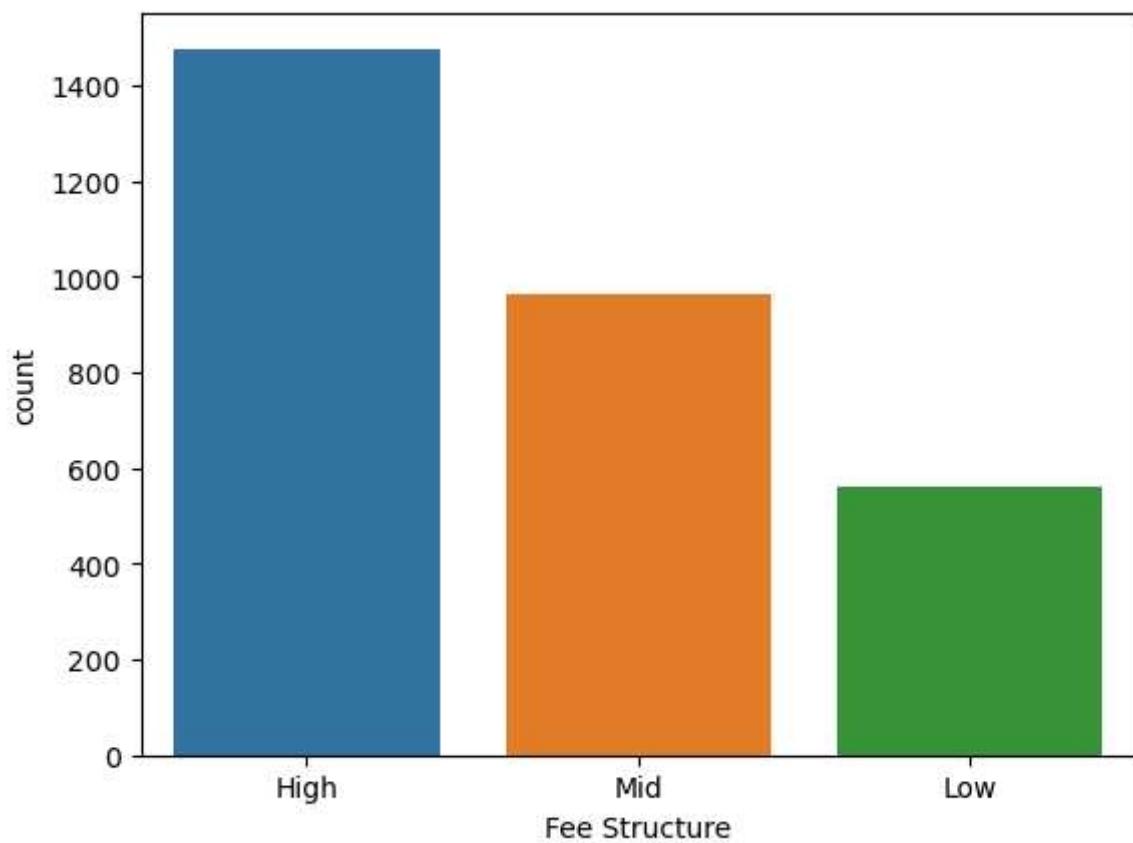
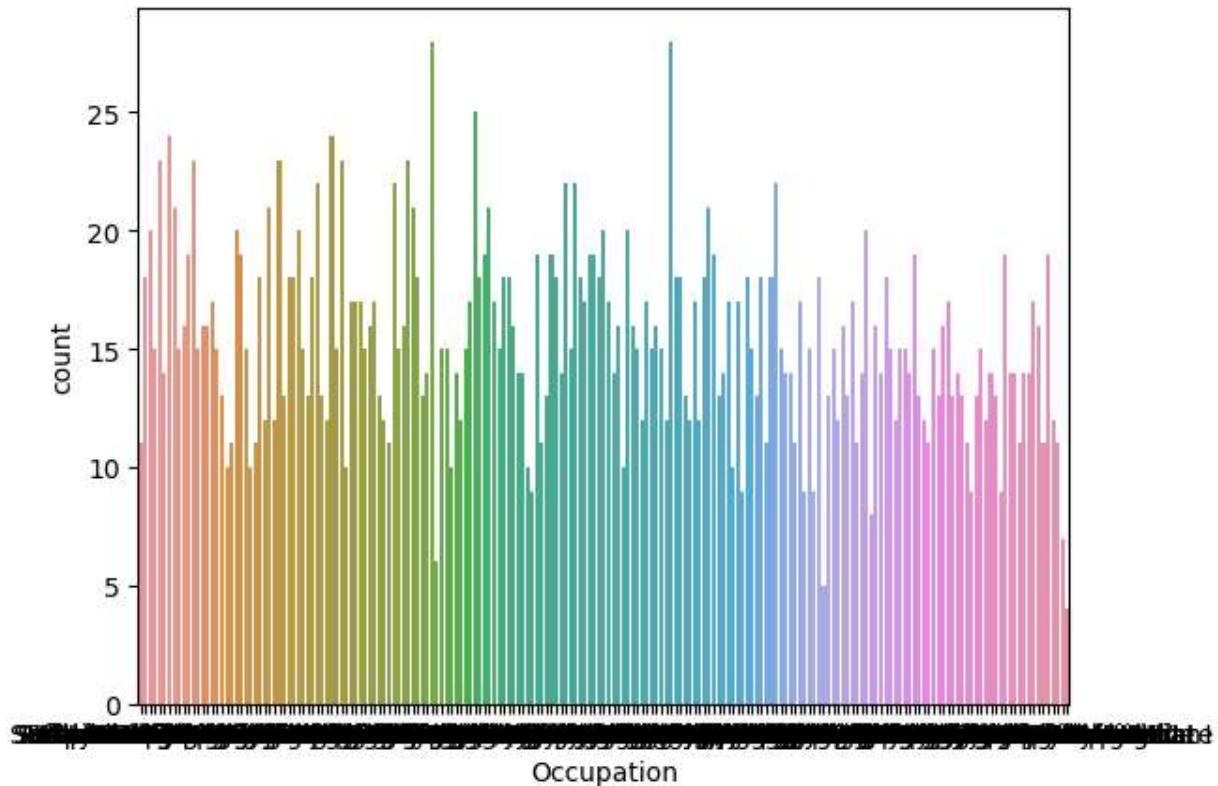
```
2    777
1    776
3    742
0    705
Name: Properties Owned, dtype: int64
Value Counts for 'Risk Weighting':
2    1222
1     836
3     460
4     322
5     160
Name: Risk Weighting, dtype: int64
Value Counts for 'Income Band':
Med      1517
Low      1027
High      456
Name: Income Band, dtype: int64
```

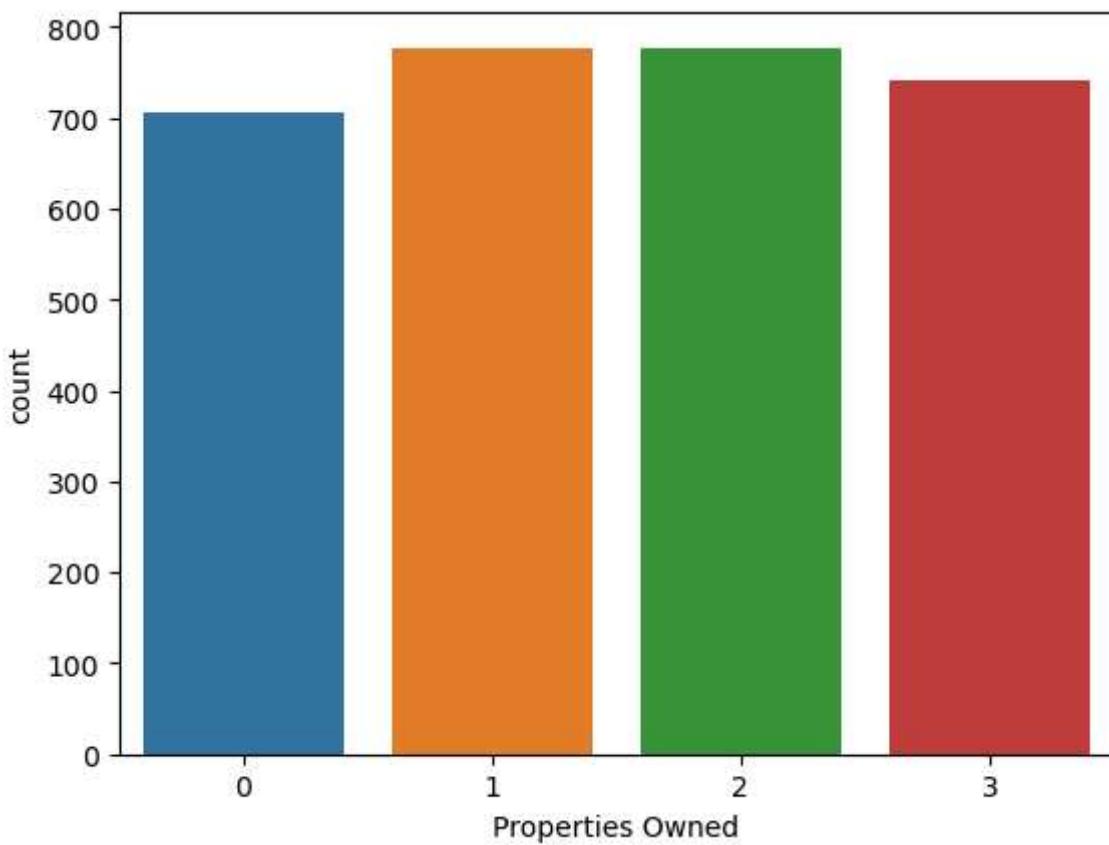
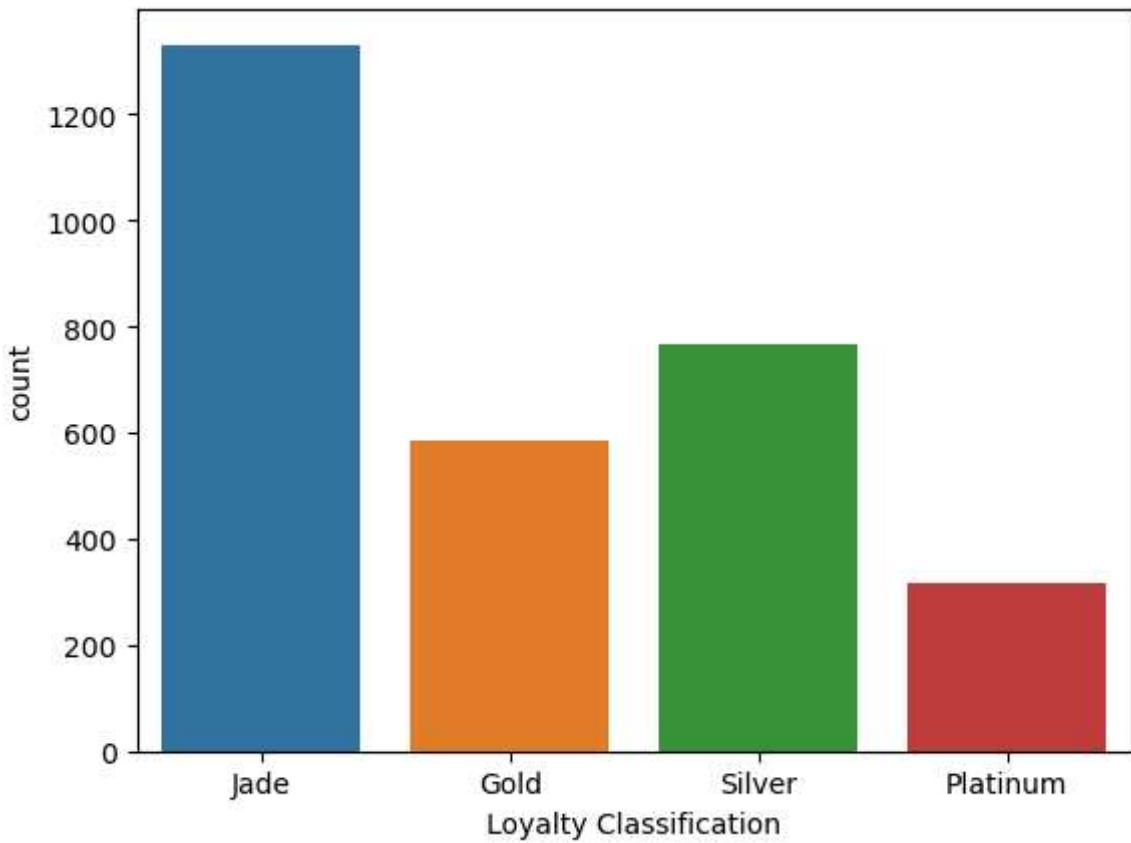
```
In [10]: for i, predictor in enumerate(df[["BRIId", "GenderId", "IAId", "Amount of Credit Cards"]]):
    plt.figure(i)
    sns.countplot(data=df, x=predictor)
```

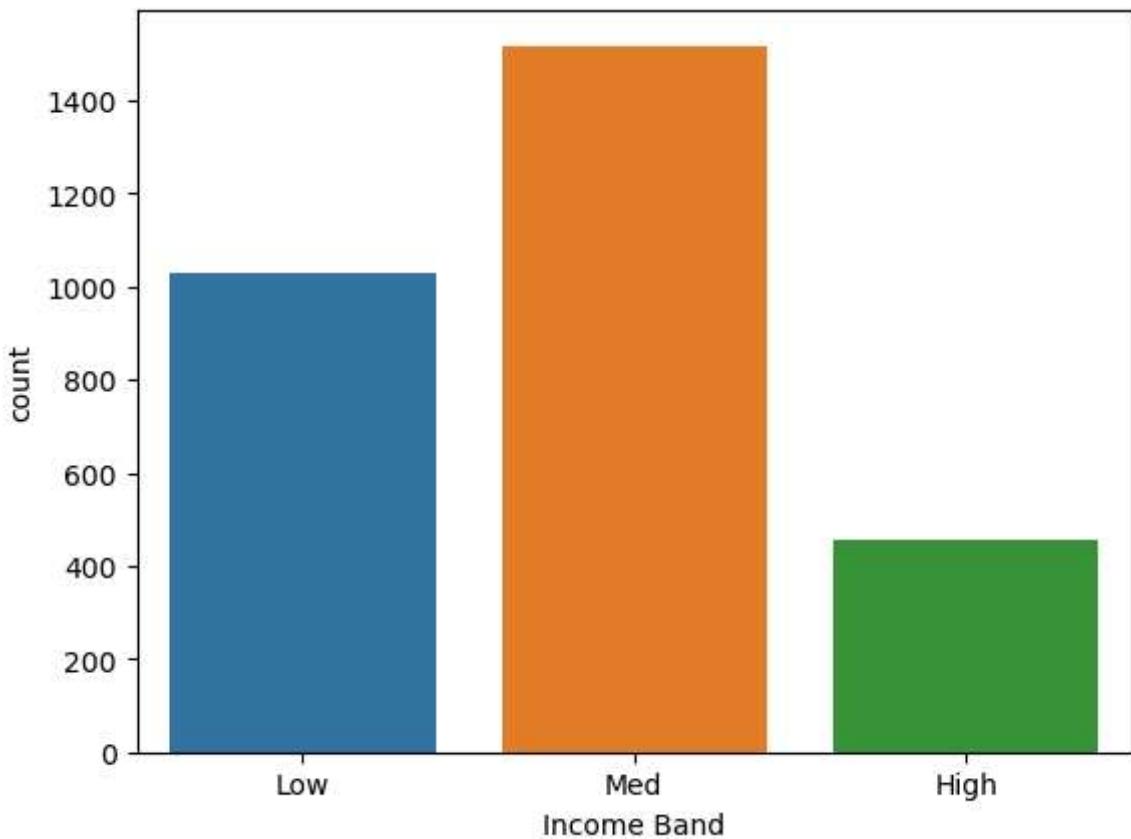
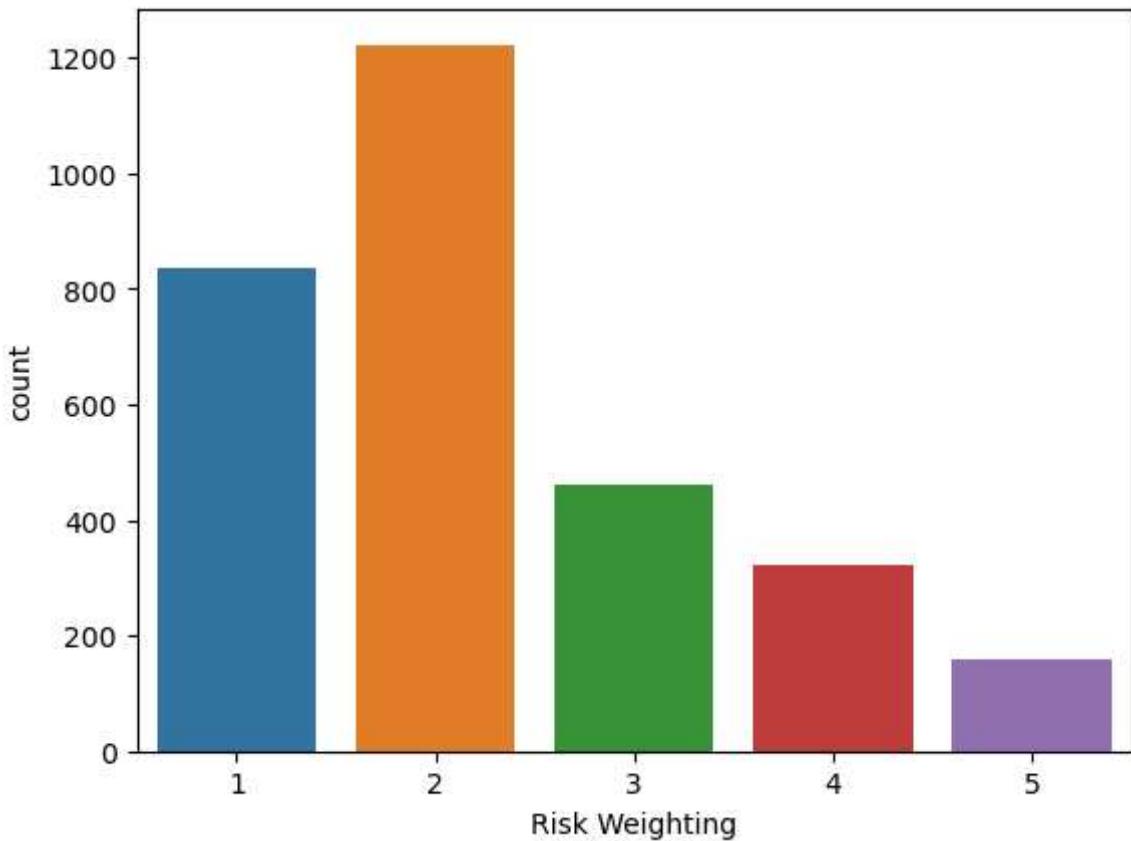




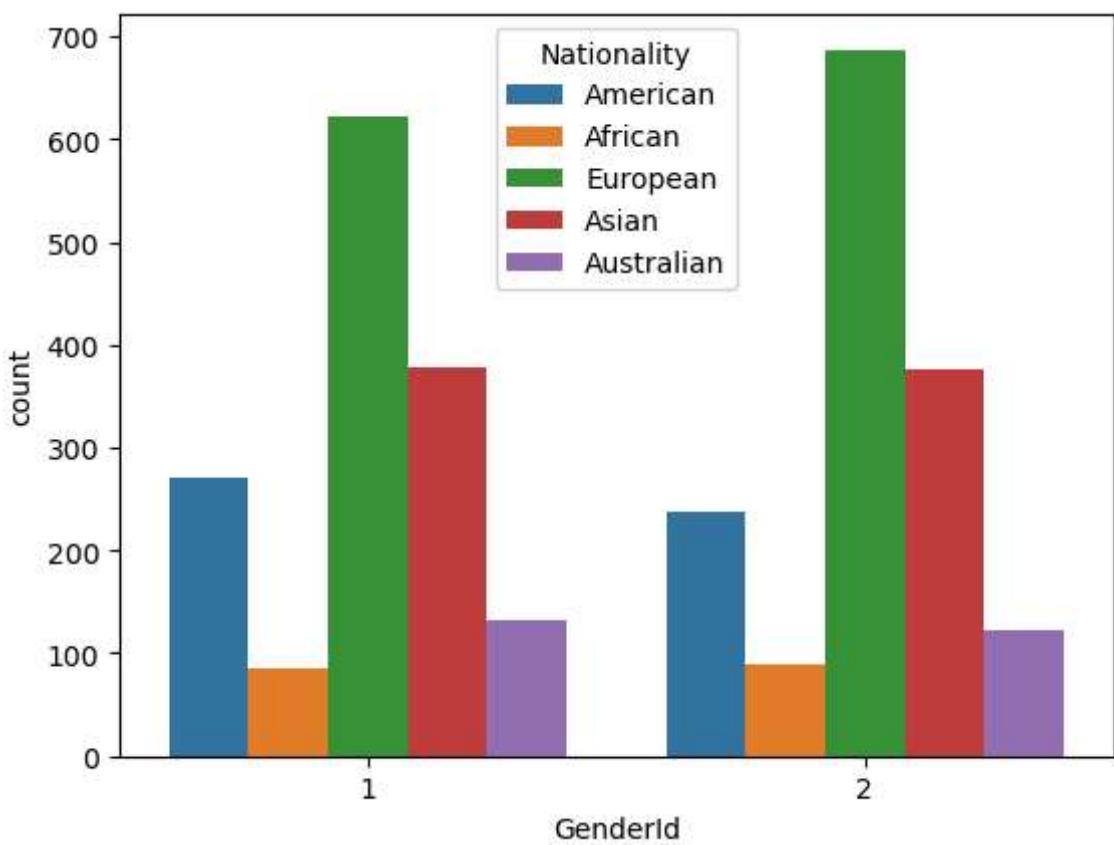
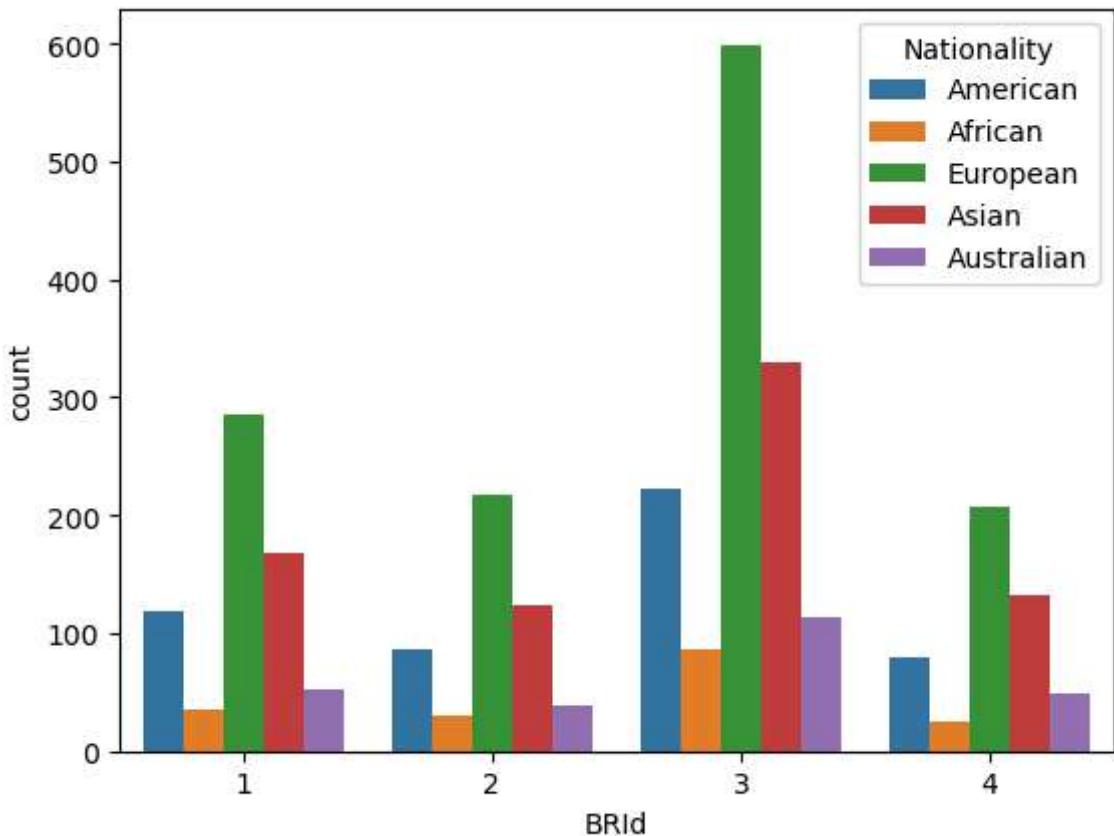


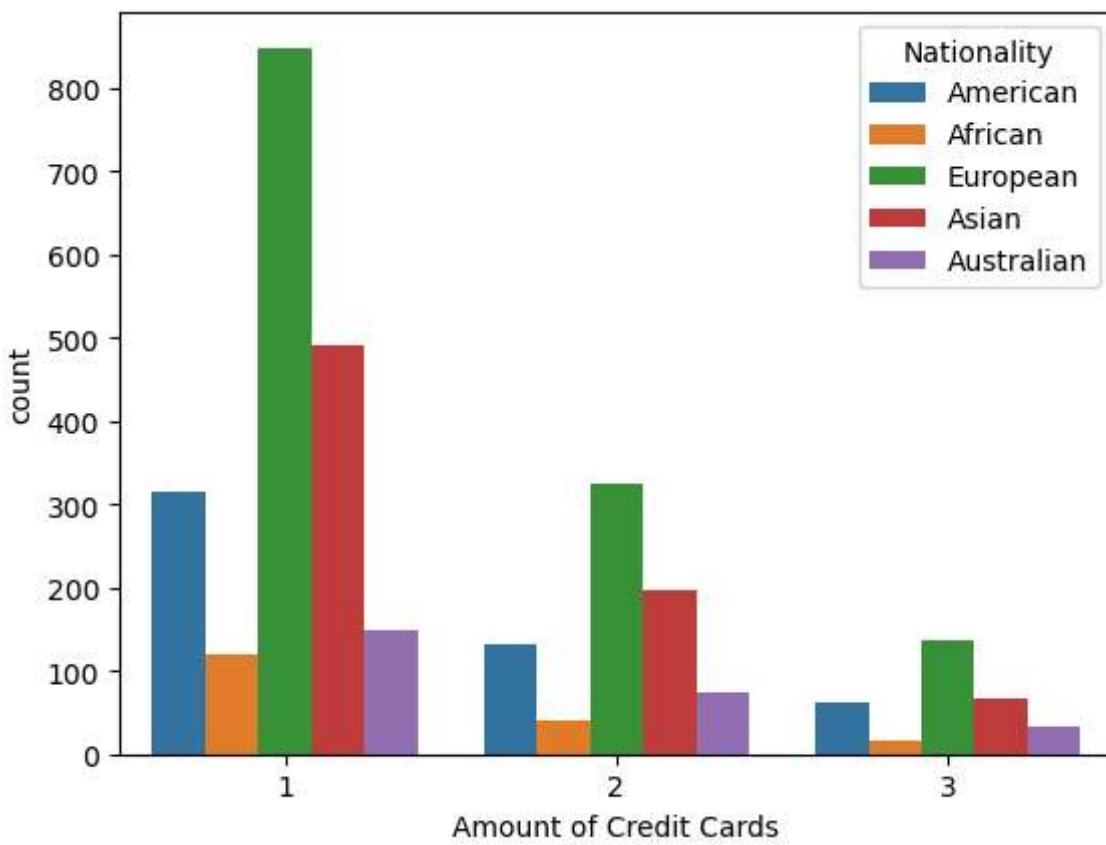
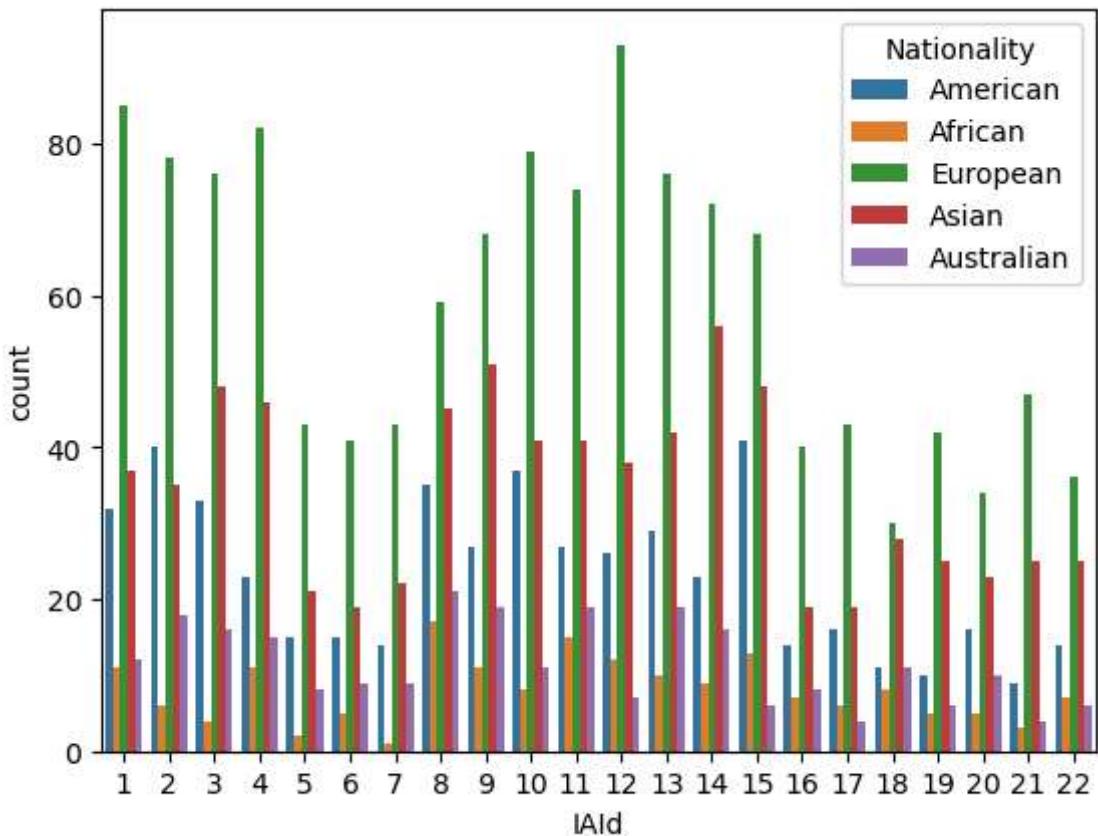


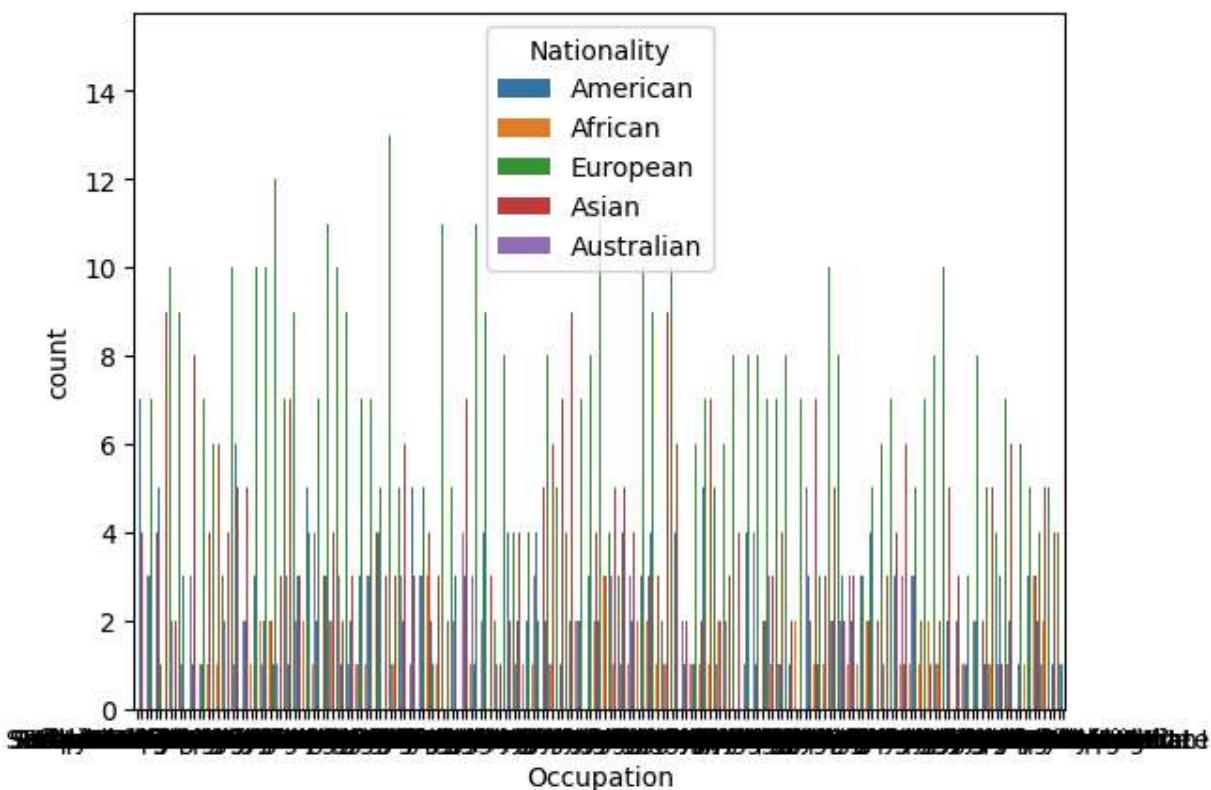
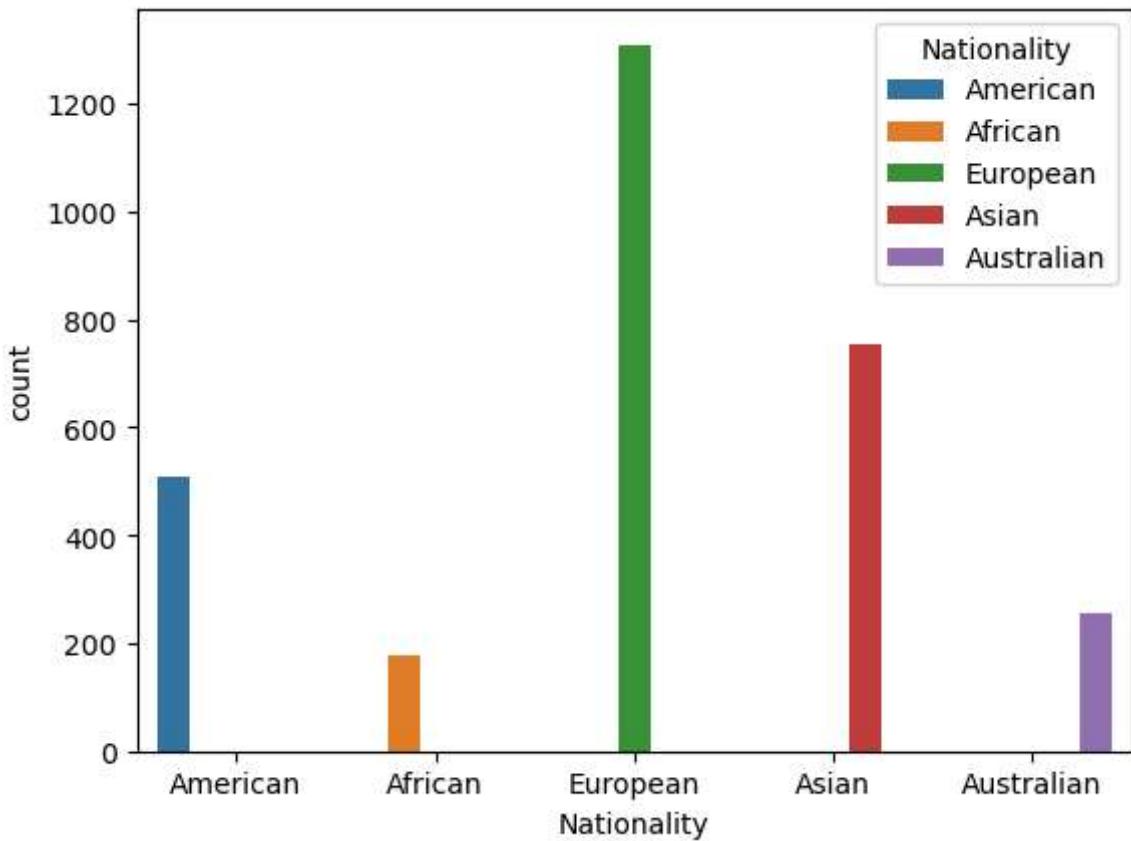


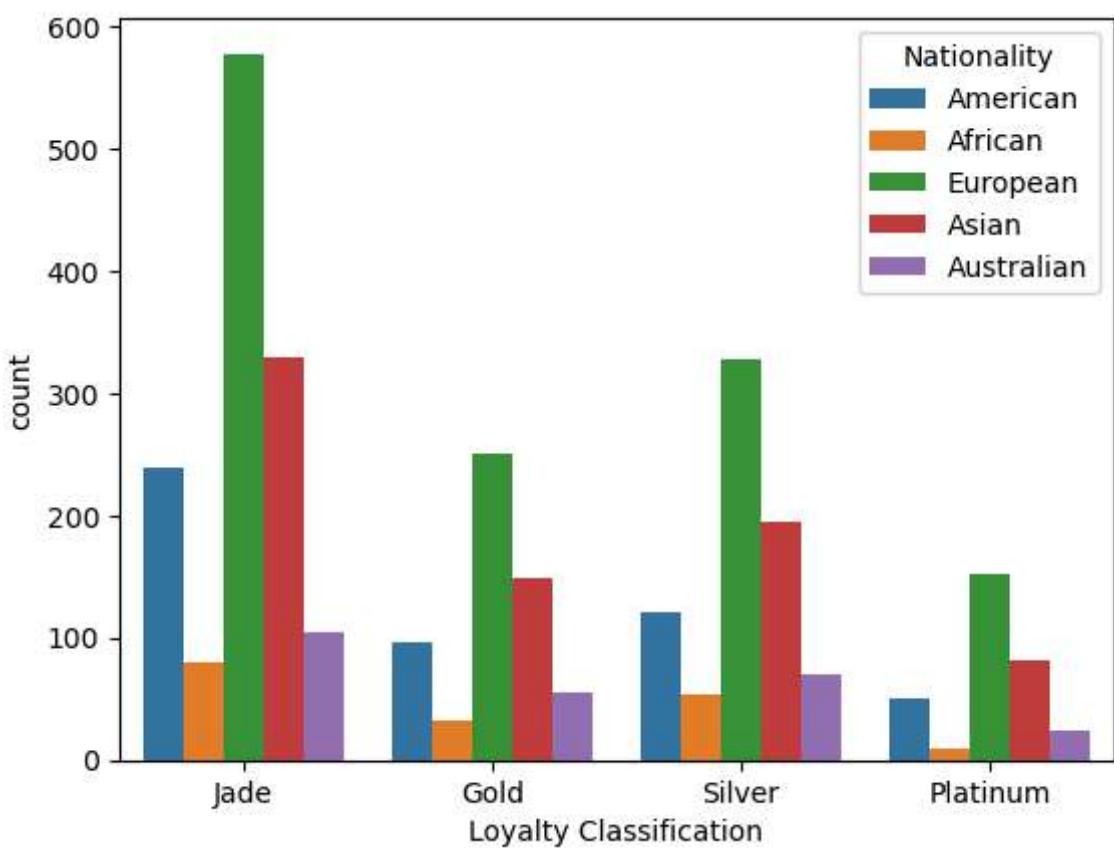
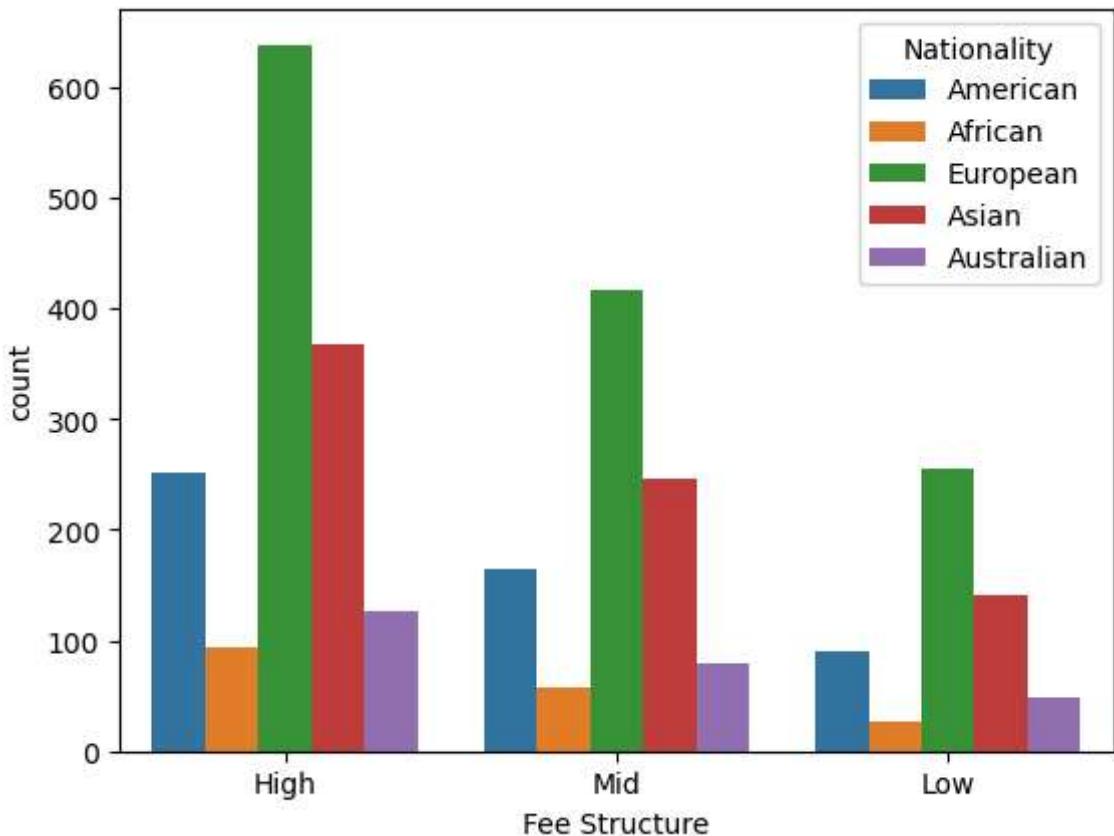


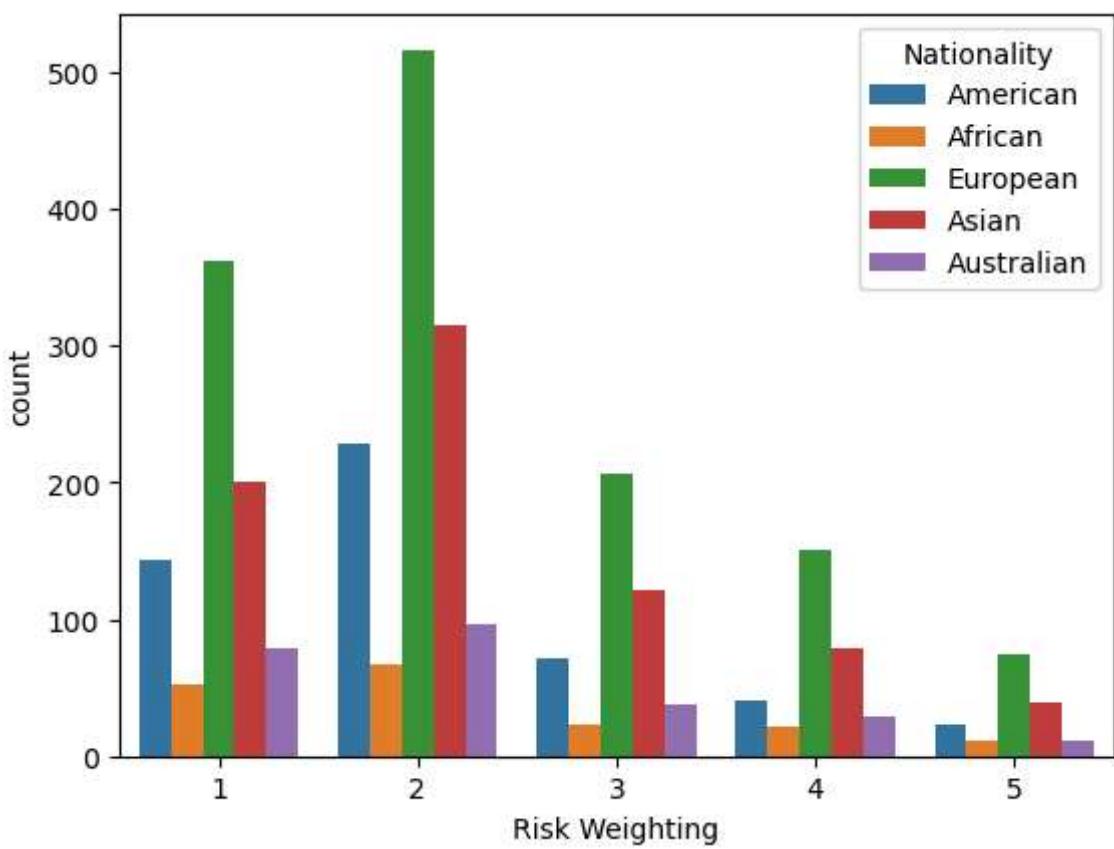
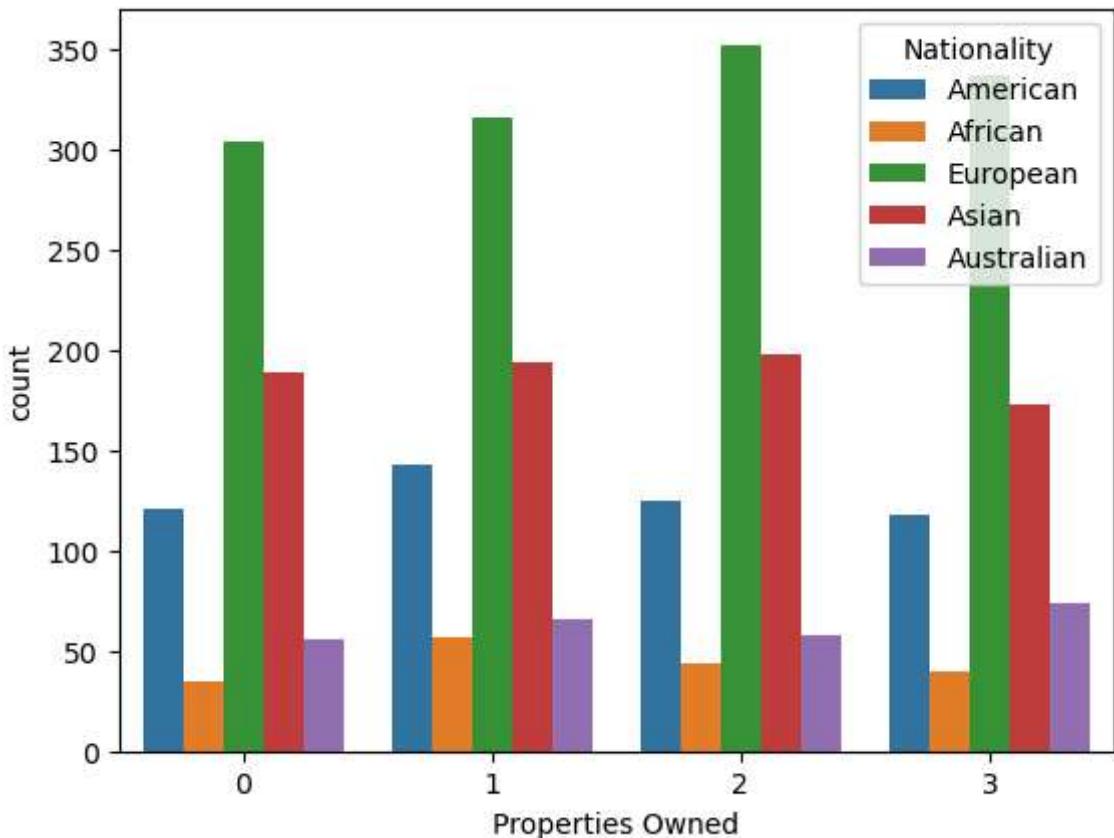
```
In [11]: for i, predictor in enumerate(df[["BRIId", "GenderId", "IAId", "Amount of Credit Cards"]]):
    plt.figure(i)
    sns.countplot(data=df, x=predictor, hue='Nationality')
```

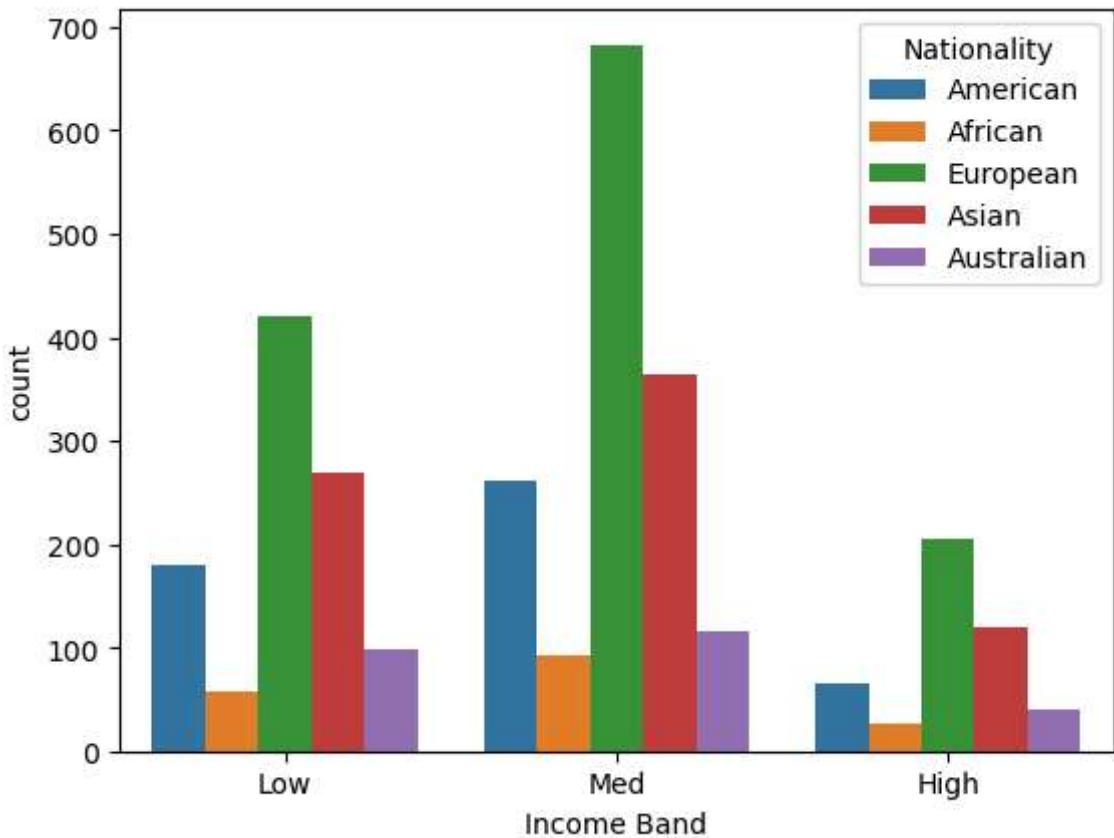




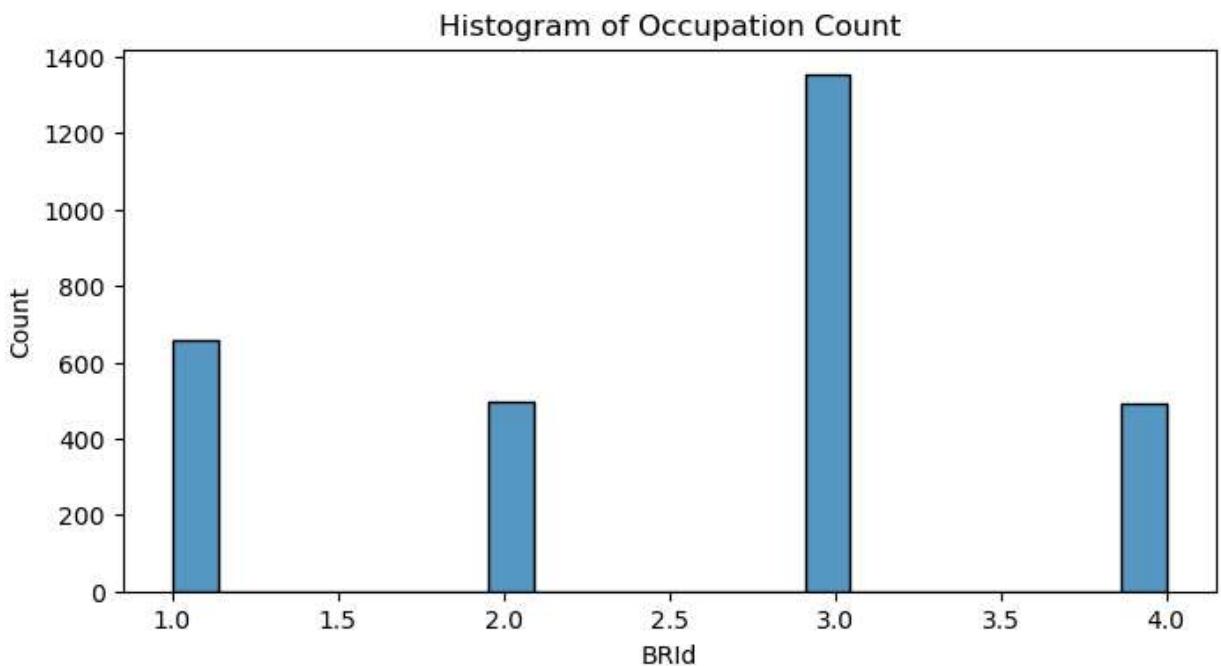




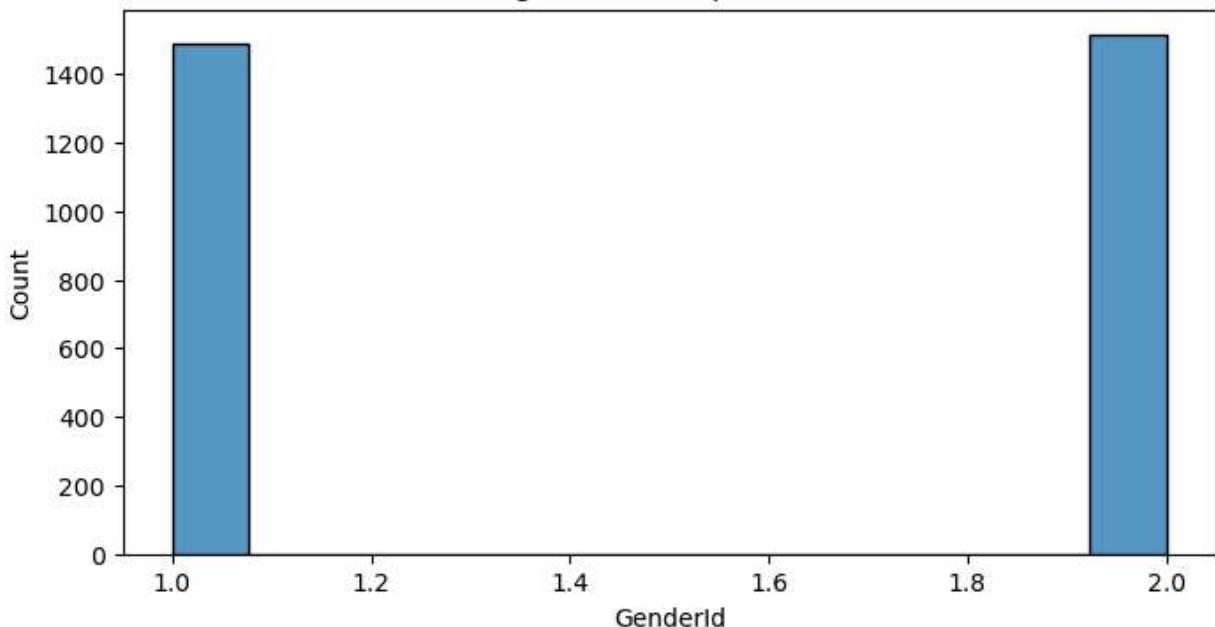




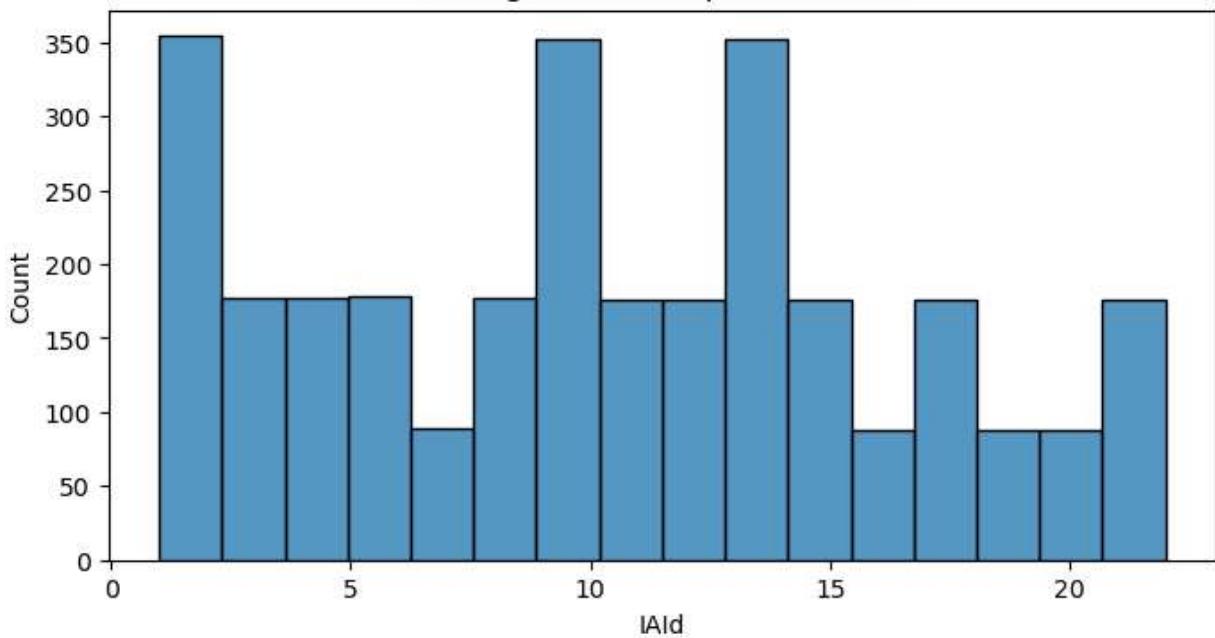
```
In [12]: for col in categorical_cols:  
    if col == "Occupation":  
        continue  
    plt.figure(figsize=(8,4))  
    sns.histplot(df[col])  
    plt.title('Histogram of Occupation Count')  
    plt.xlabel(col)  
    plt.ylabel("Count")  
    plt.show()
```



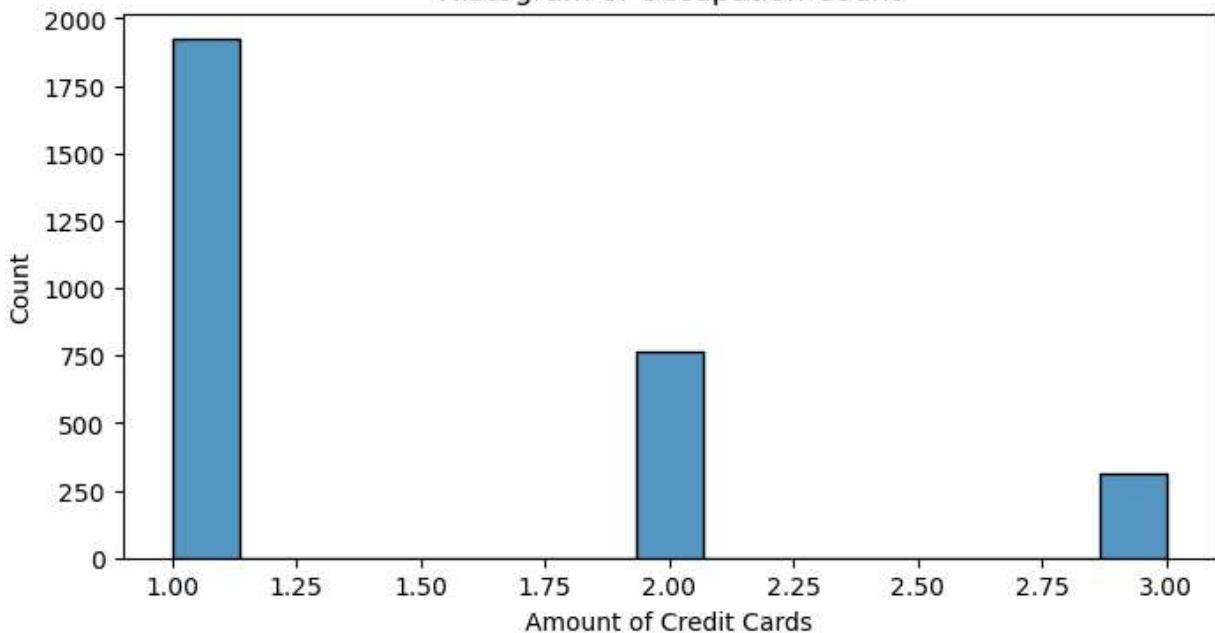
Histogram of Occupation Count



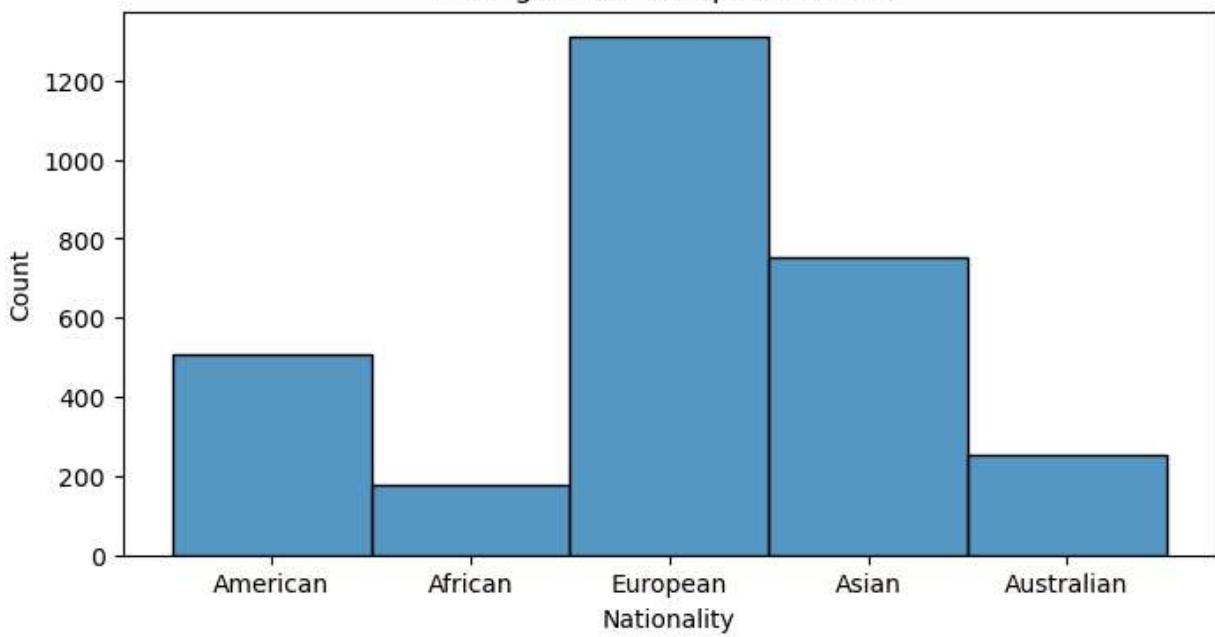
Histogram of Occupation Count



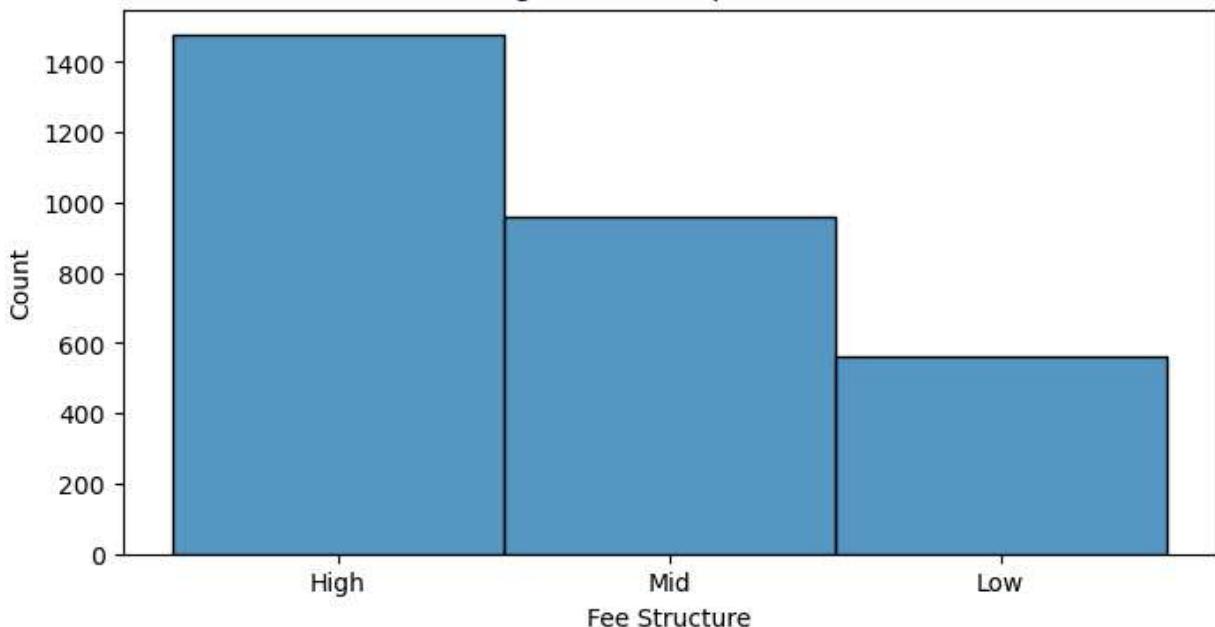
Histogram of Occupation Count



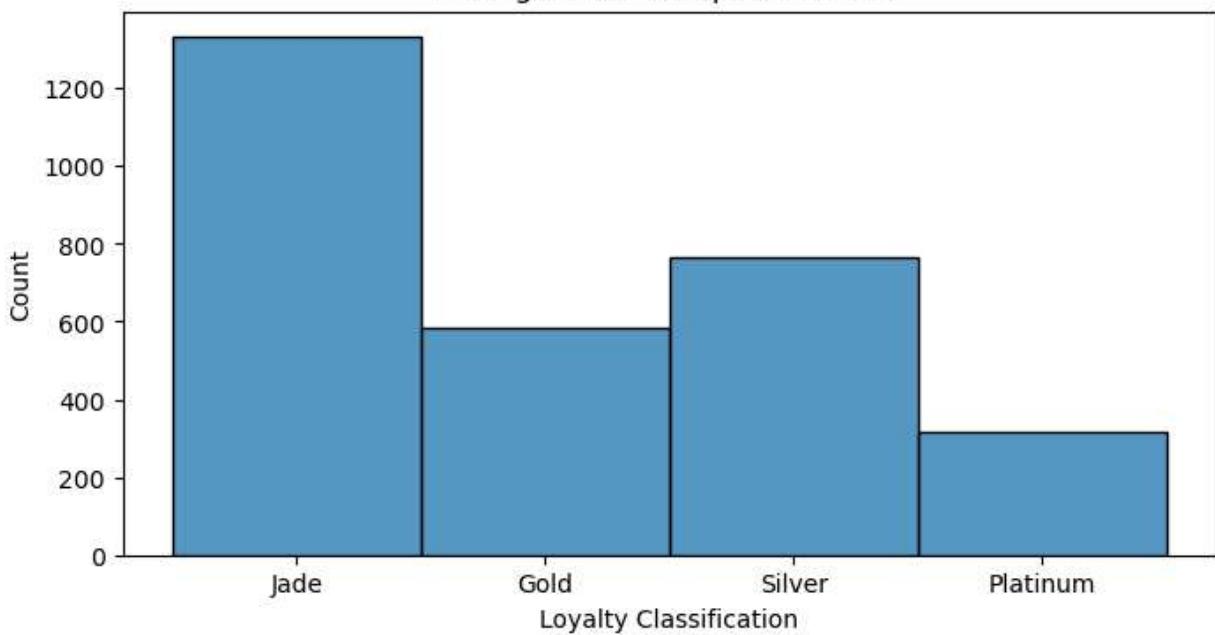
Histogram of Occupation Count



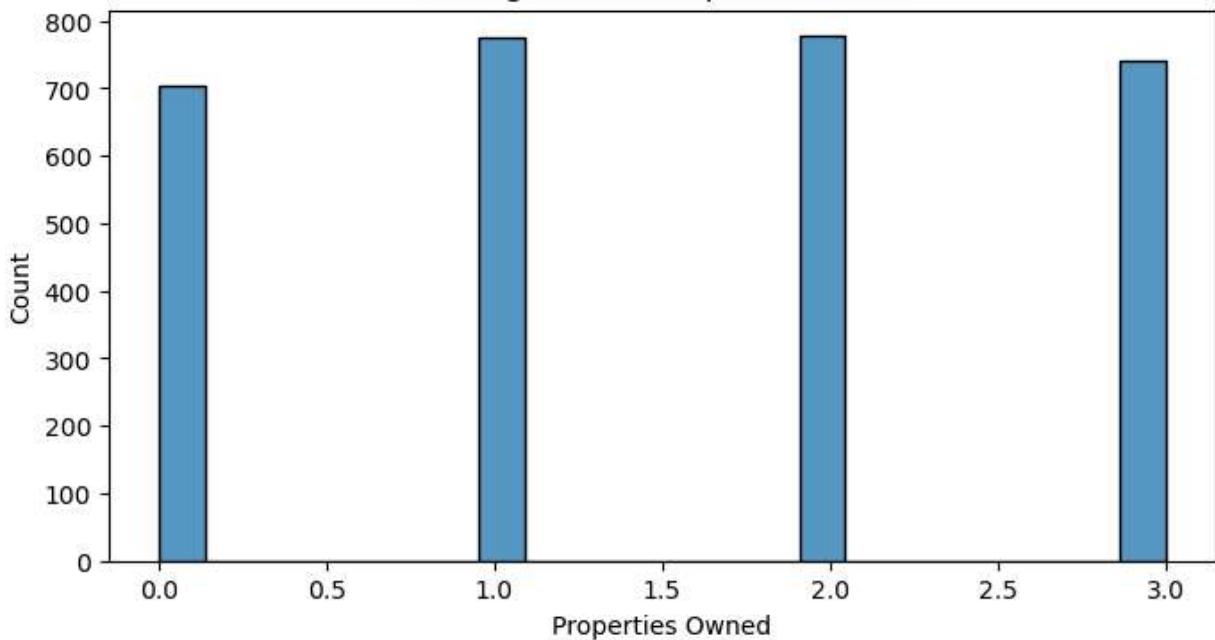
Histogram of Occupation Count



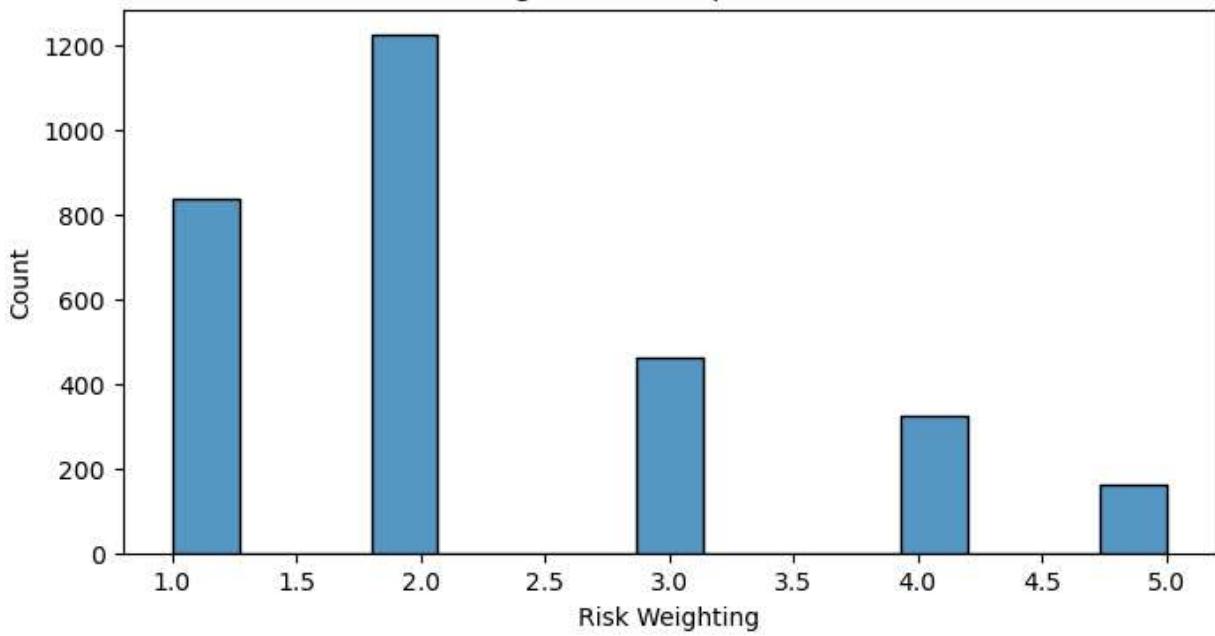
Histogram of Occupation Count



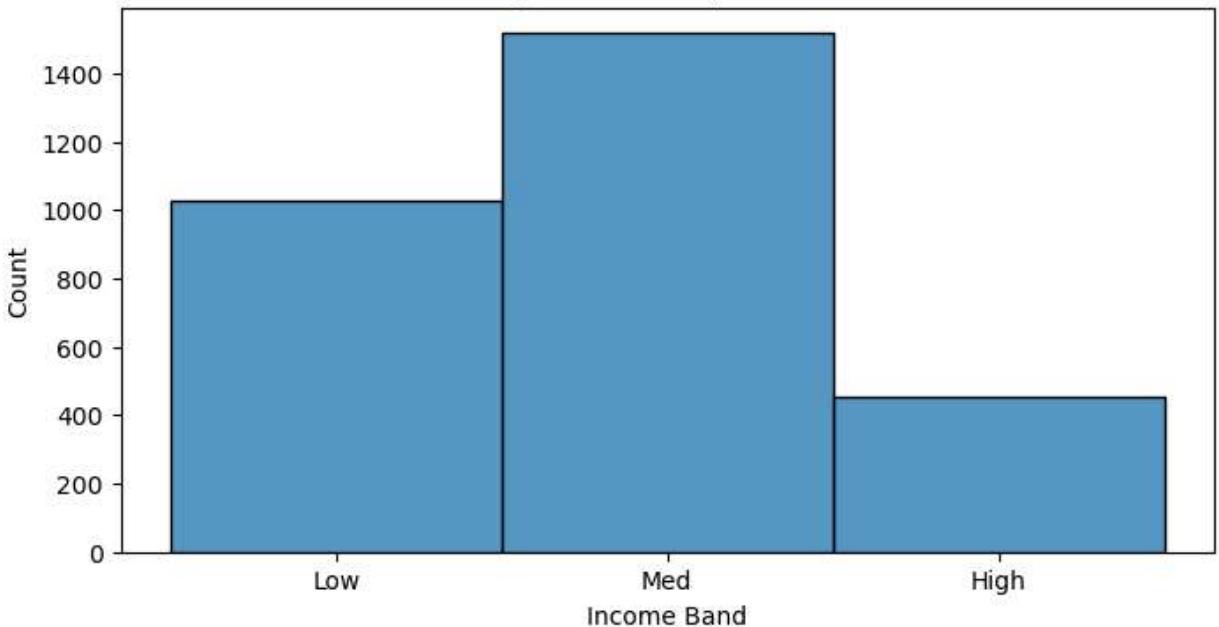
Histogram of Occupation Count



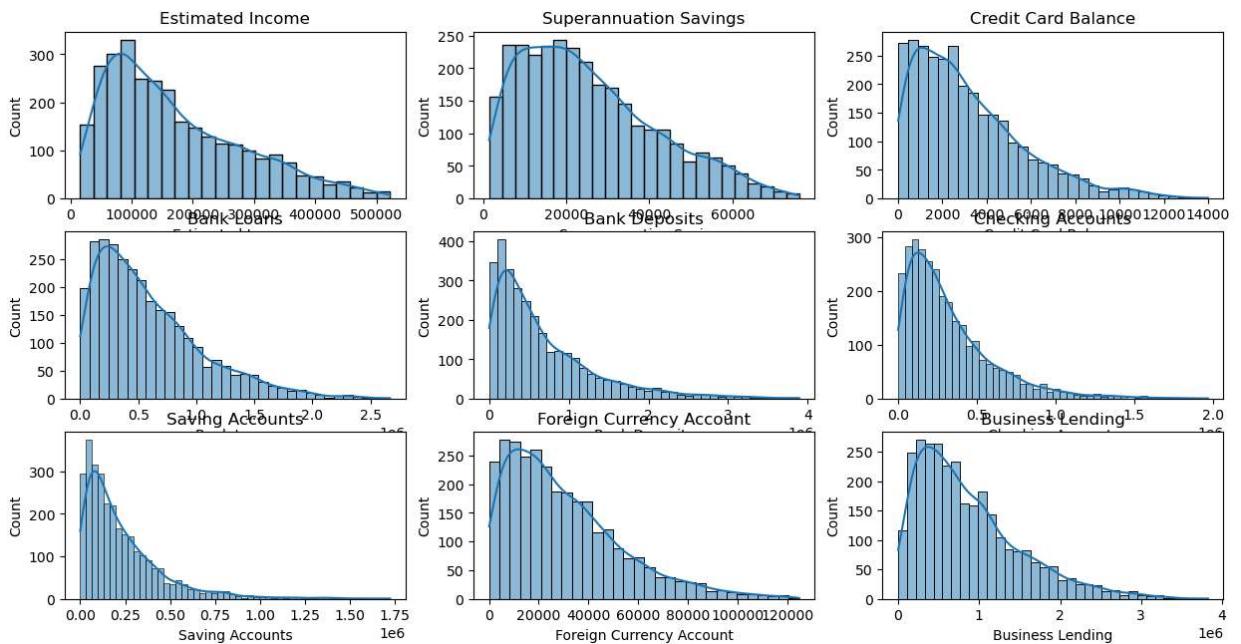
Histogram of Occupation Count



Histogram of Occupation Count



```
In [13]: numerical_cols = ['Estimated Income', 'Superannuation Savings', 'Credit Card Balance',  
plt.figure(figsize=(15,10))  
for i,col in enumerate(numerical_cols):  
    plt.subplot(4,3,i+1)  
    sns.histplot(df[col],kde=True)  
    plt.title(col)  
plt.show()
```



```
In [14]: numerical_cols = ['Estimated Income', 'Superannuation Savings', 'Credit Card Balance',  
correlation_matrix = df[numerical_cols].corr()  
  
plt.figure(figsize=(12,12))  
sns.heatmap(correlation_matrix, annot=True, cmap='crest', fmt=".2f")  
plt.title("Correlation Matrix")  
plt.show()
```



In []: