In [7]:
```
import mysql.connector
import pandas as pd
# Connect to server
cnx = mysql.connector.connect(
    host="127.0.0.1",
    port=3306,
    user="root",
    password="*******")
```

In [ ]:
```
```

In [6]:
```
query = "select * from banking_case.customer"
```

In [8]:
```
df = pd.read_sql(query, cnx)
```

```
C:\Users\zoyag\AppData\Local\Temp\ipykernel_12724\1600954950.py:1: UserWarning: p
andas only supports SQLAlchemy connectable (engine/connection) or database string
URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please con
sider using SQLAlchemy.
  df = pd.read_sql(query, cnx)
```

In [9]:
```
cnx.close()
```

In [10]:
```
print(df)
```

	ï»¿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	BRId	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]
```

In [11]:
```
df.head()
```

Out[11]:

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

5 rows × 25 columns

In [12]:
```
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
```

Data processing and data cleaning

In [13]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [15]:
```
df.shape
```

Out[15]: (3000, 25)

In [16]:
```
# generate descriptive statistics for the dataframe
```

In [17]:
```
df.describe()
```

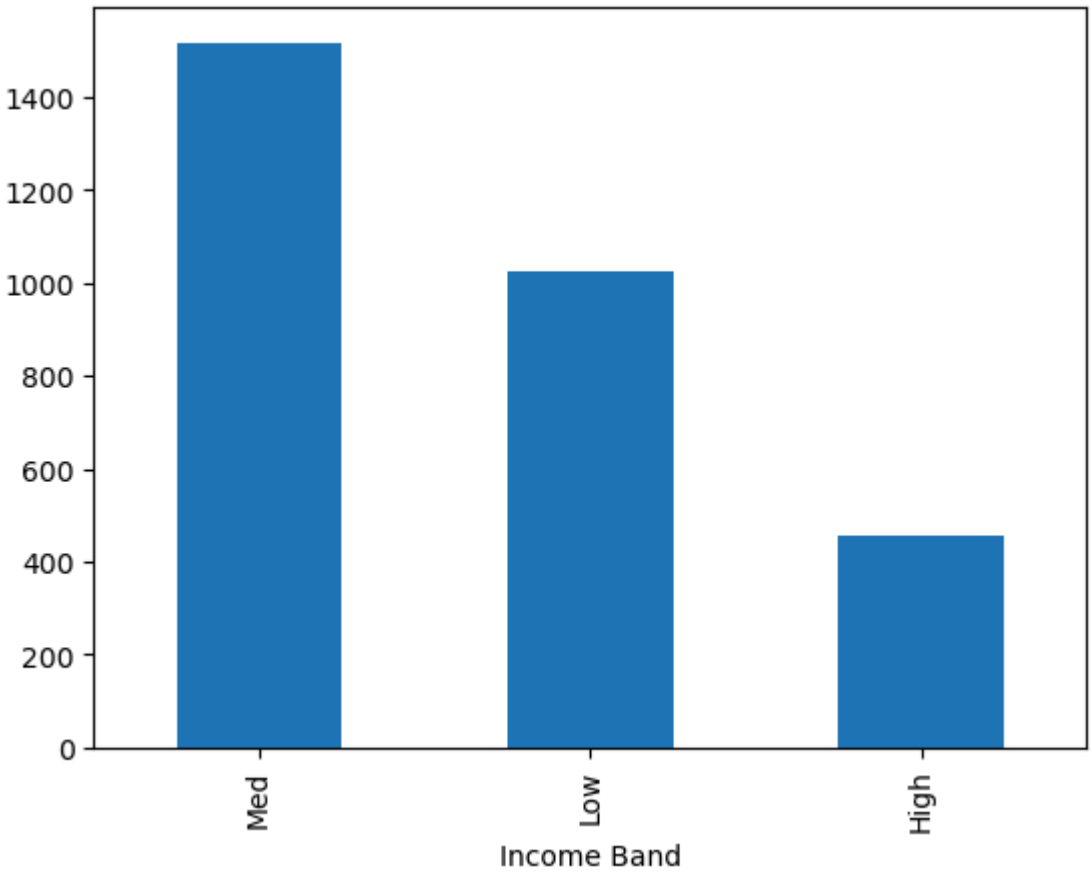
Out[17]:

	Age	Location ID	Estimated Income	Superannuation Savings	Amount of Credit Cards	Credit Ba
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.0(
mean	51.039667	21563.323000	171305.034263	25531.599673	1.463667	3176.2(
std	19.854760	12462.273017	111935.808209	16259.950770	0.676387	2497.0(
min	17.000000	12.000000	15919.480000	1482.030000	1.000000	1.1(
25%	34.000000	10803.500000	82906.595000	12513.775000	1.000000	1236.6(
50%	51.000000	21129.500000	142313.480000	22357.355000	1.000000	2560.8(
75%	69.000000	32054.500000	242290.305000	35464.740000	2.000000	4522.6(
max	85.000000	43369.000000	522330.260000	75963.900000	3.000000	13991.9(

In [21]:
```
bins = [0, 100000, 300000, float('inf')]
labels = ['Low','Med','High']
df['Income Band'] = pd.cut(df['Estimated Income'], bins=bins, labels=labels, rig
```

In [24]:
```
df['Income Band'].value_counts().plot(kind='bar')
```

Out[24]: <Axes: xlabel='Income Band'>



In [29]:
```
# examine the distribution of unique categories in categorial columns
categorial_cos = df[["BRId" , "GenderId", "IAId", "Amount of Credit Cards", "Nat
for col in categorial_cos:
```

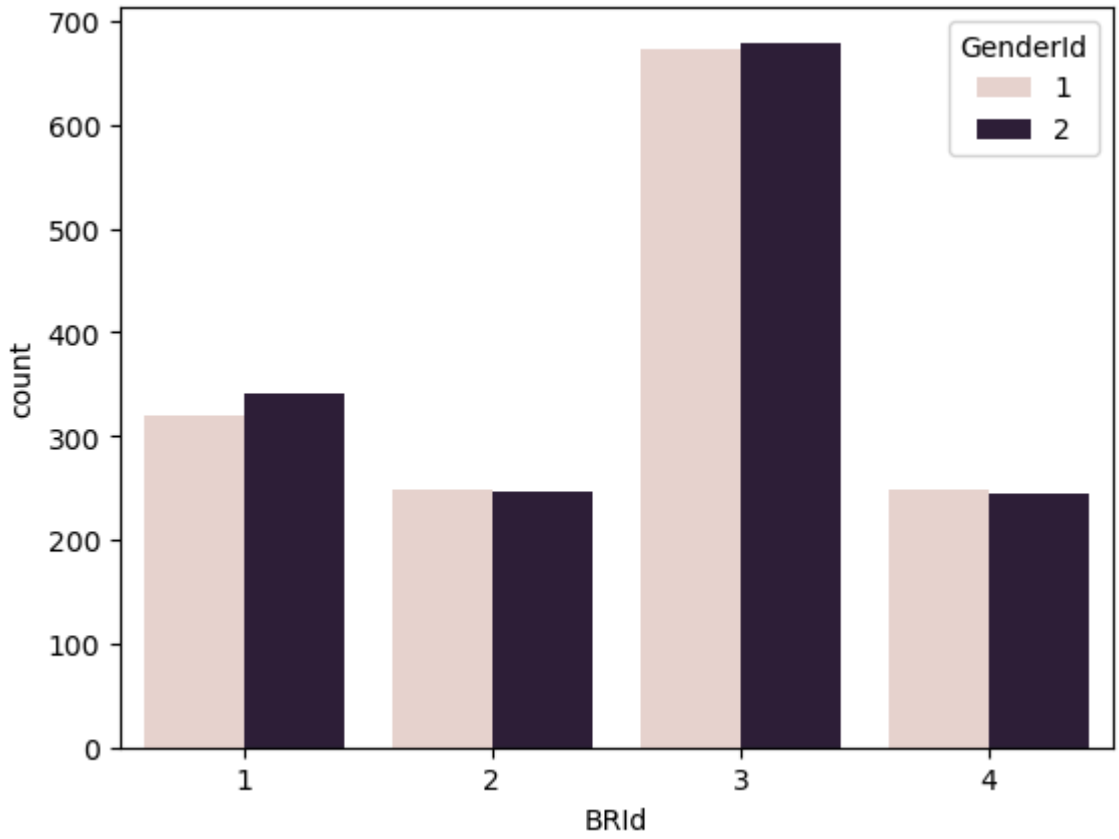
```
    print(f"Value Counts for '{col}':")
    display(df[col].value_counts())
```

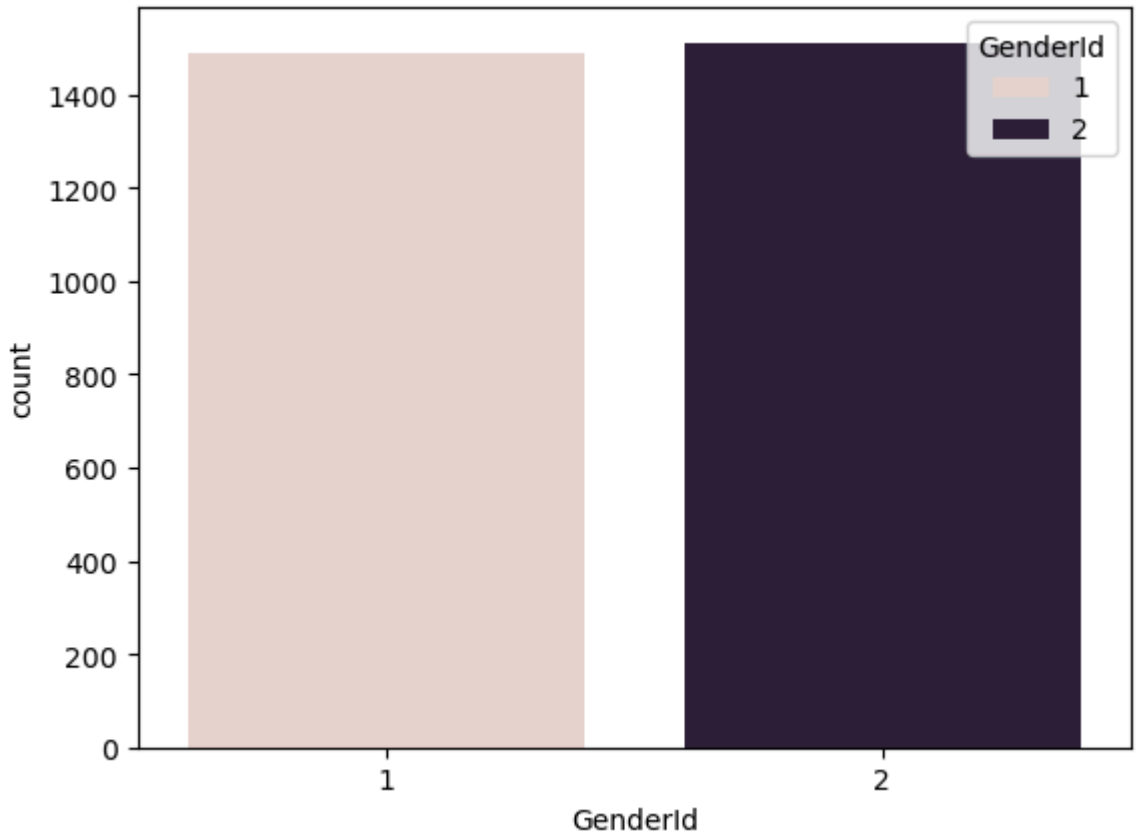
```
Value Counts for 'BRId':
BRId
3    1352
1     660
2     495
4     493
Name: count, dtype: int64
Value Counts for 'GenderId':
GenderId
2    1512
1    1488
Name: count, dtype: int64
Value Counts for 'IAId':
IAId
1     177
2     177
3     177
4     177
8     177
9     176
13    176
12    176
10    176
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':
```

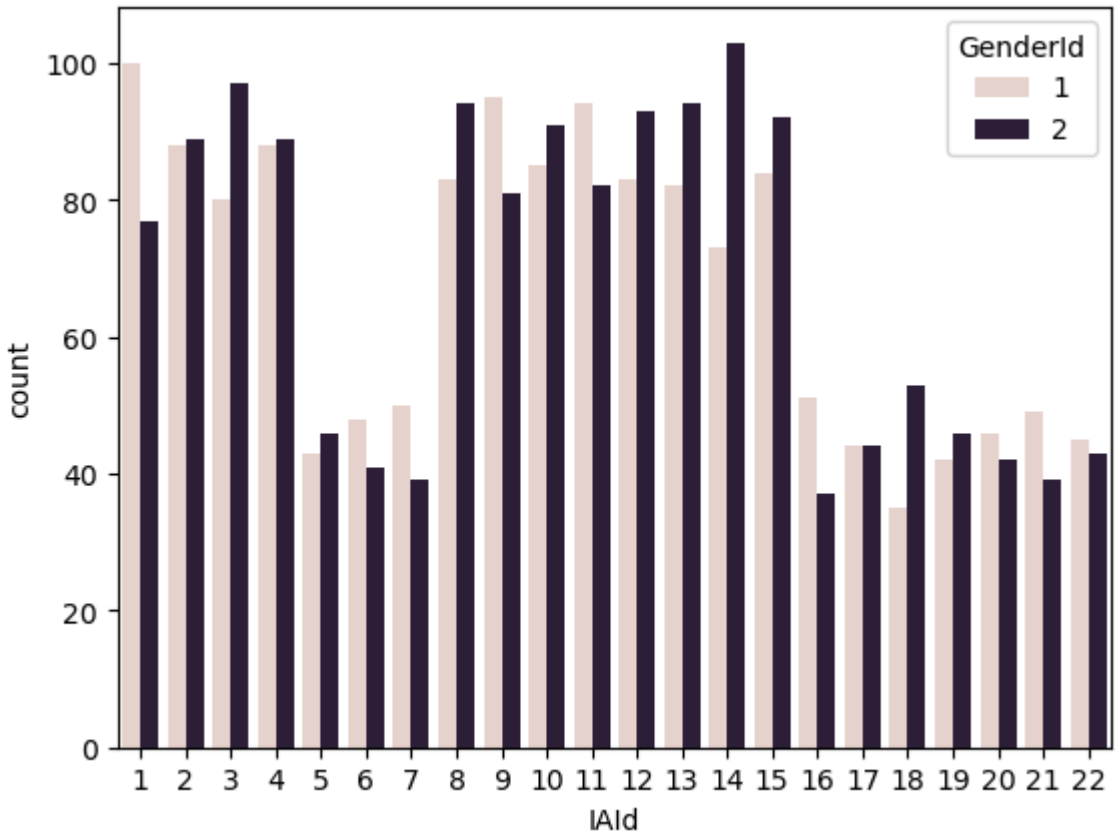
```
Occupation
Associate Professor             28
Structural Analysis Engineer    28
Recruiter                       25
Account Coordinator             24
Human Resources Manager         24
                                ..
Office Assistant IV              8
Automation Specialist I          7
Computer 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
```

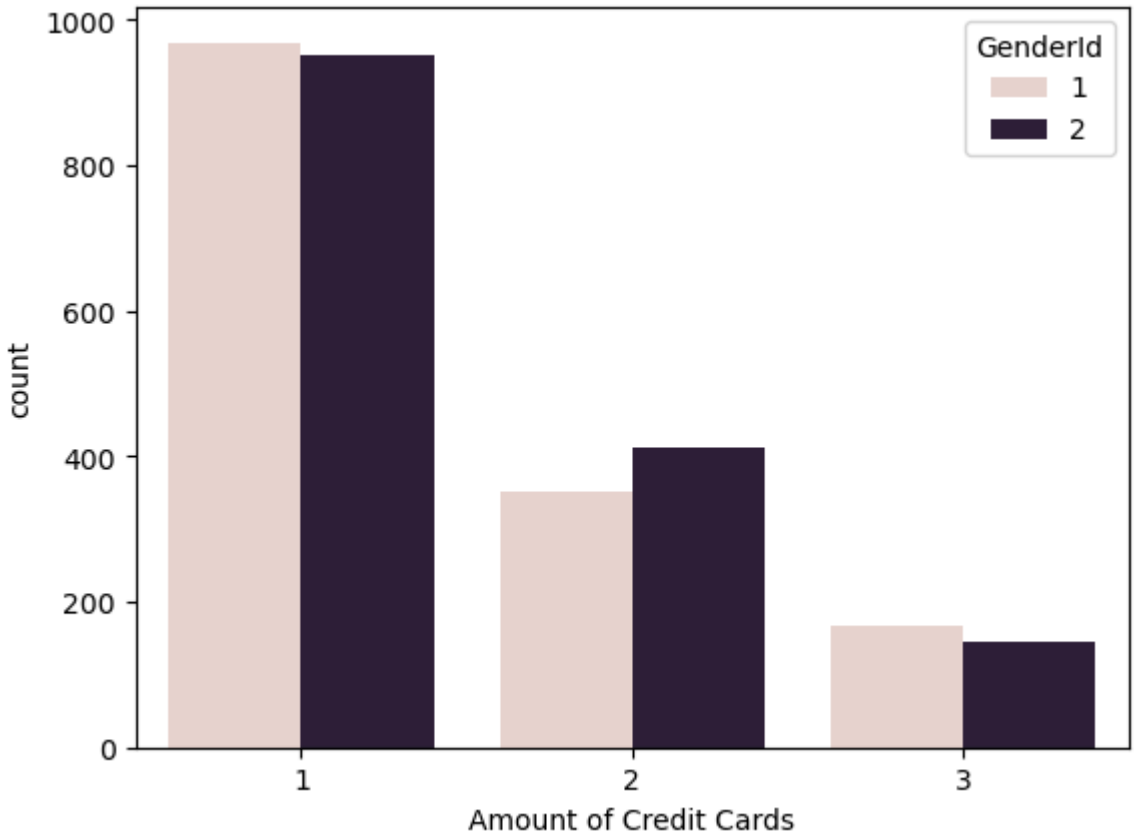
In [ ]:
```
# with the help of hue you can analysis the data based on bivariate
# if you will remove the hue it will give the analysis based on bivariate
```

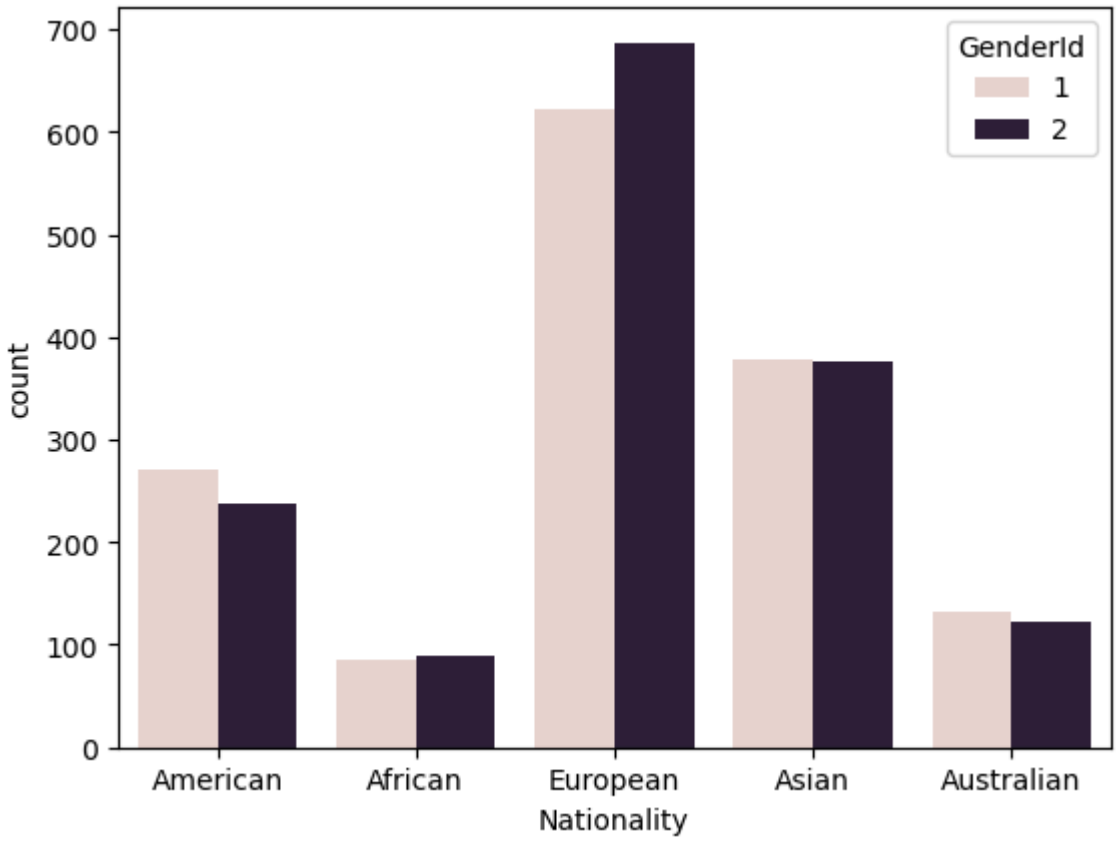
In [33]:
```
for i,predictor in enumerate(df[["BRId" , "GenderId", "IAId", "Amount of Credit
    plt.figure(i)
    sns.countplot(data=df, x=predictor, hue="GenderId")
```

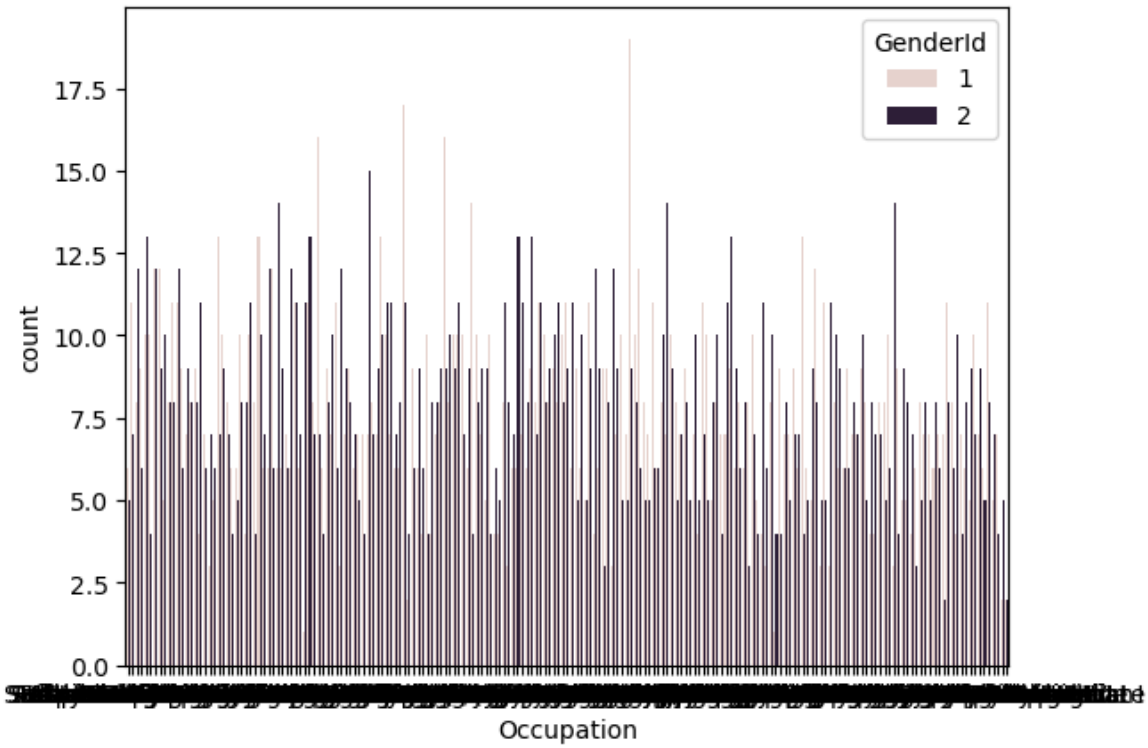

GenderId
1
2
count
BRId
0
100
200
300
400
500
600
700
1
2
3
4

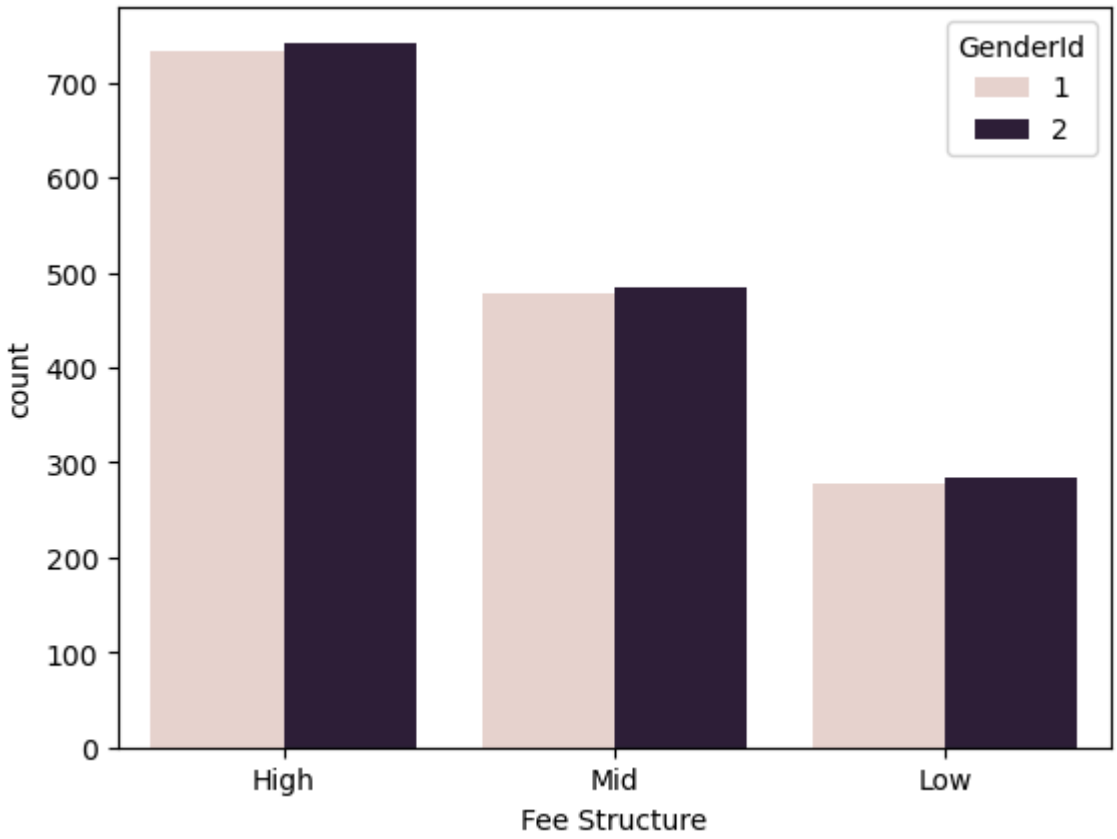

GenderId
1
2
count
GenderId
0
200
400
600
800
1000
1200
1400
1
2

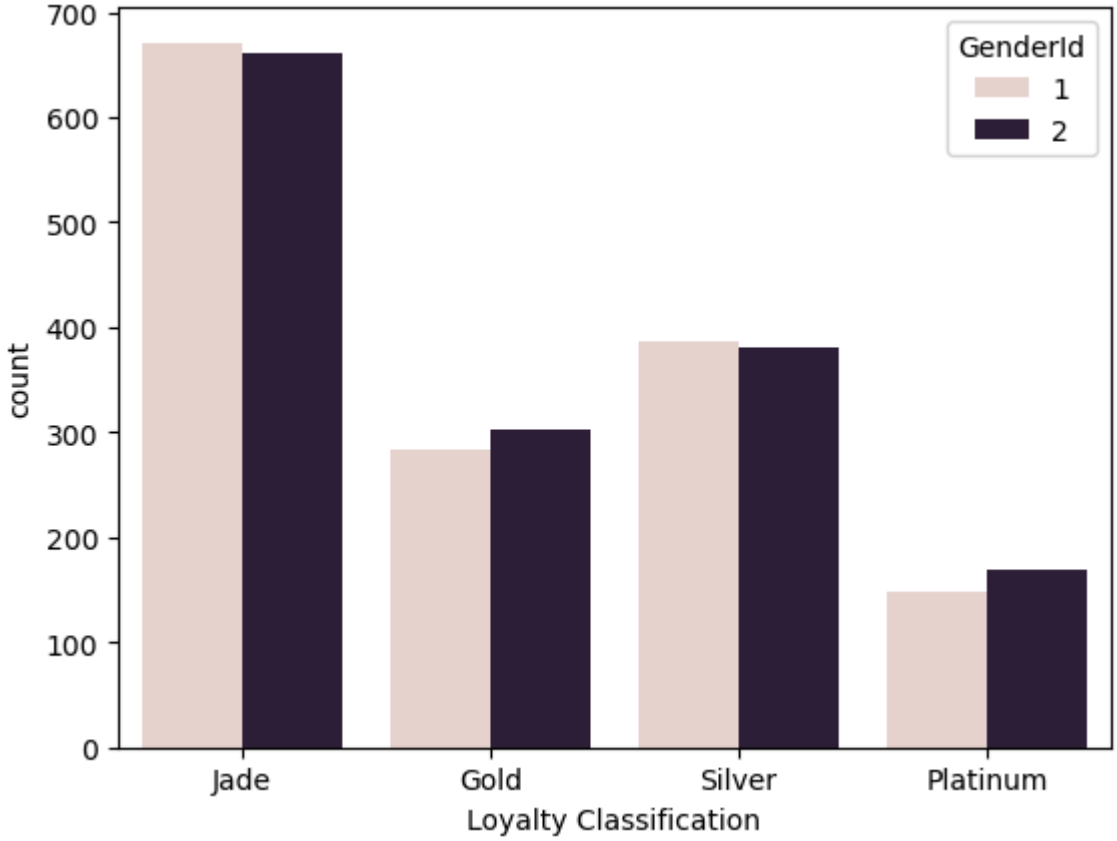

GenderId
1
2
count
0
20
40
60
80
100
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
IAId

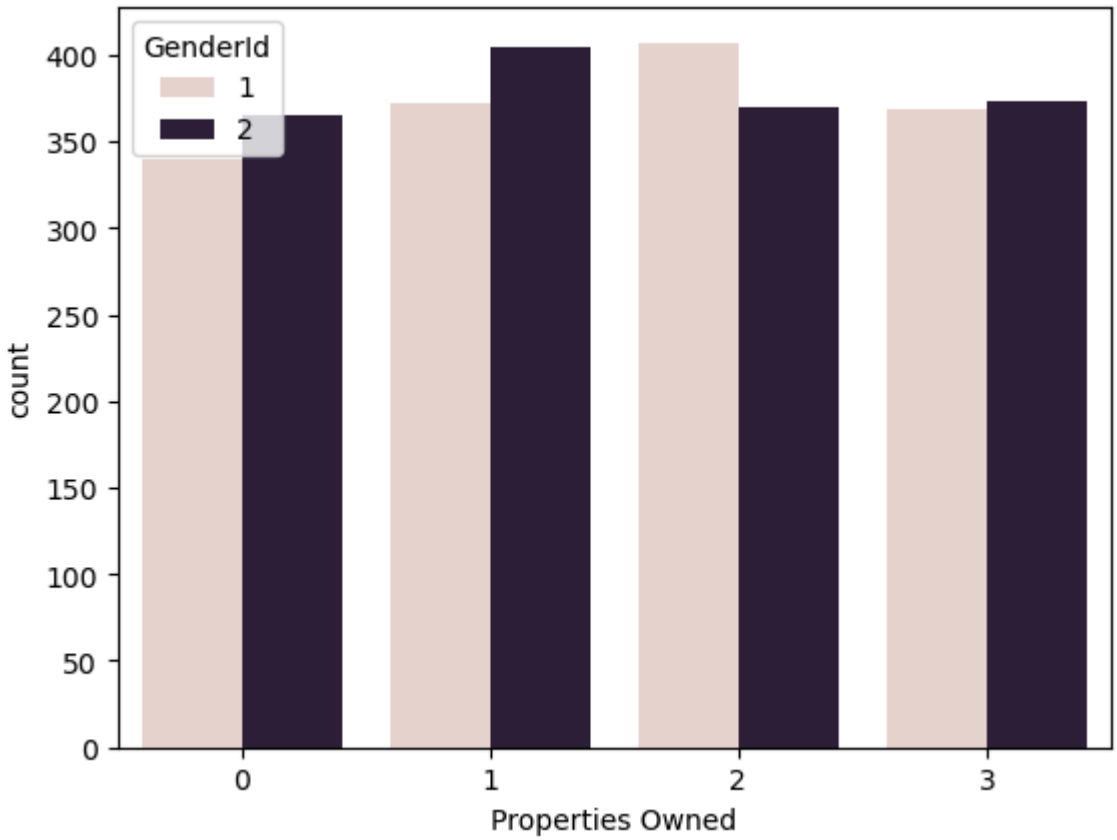

GenderId
1
2
count
0
200
400
600
800
1000
1
2
3
Amount of Credit Cards

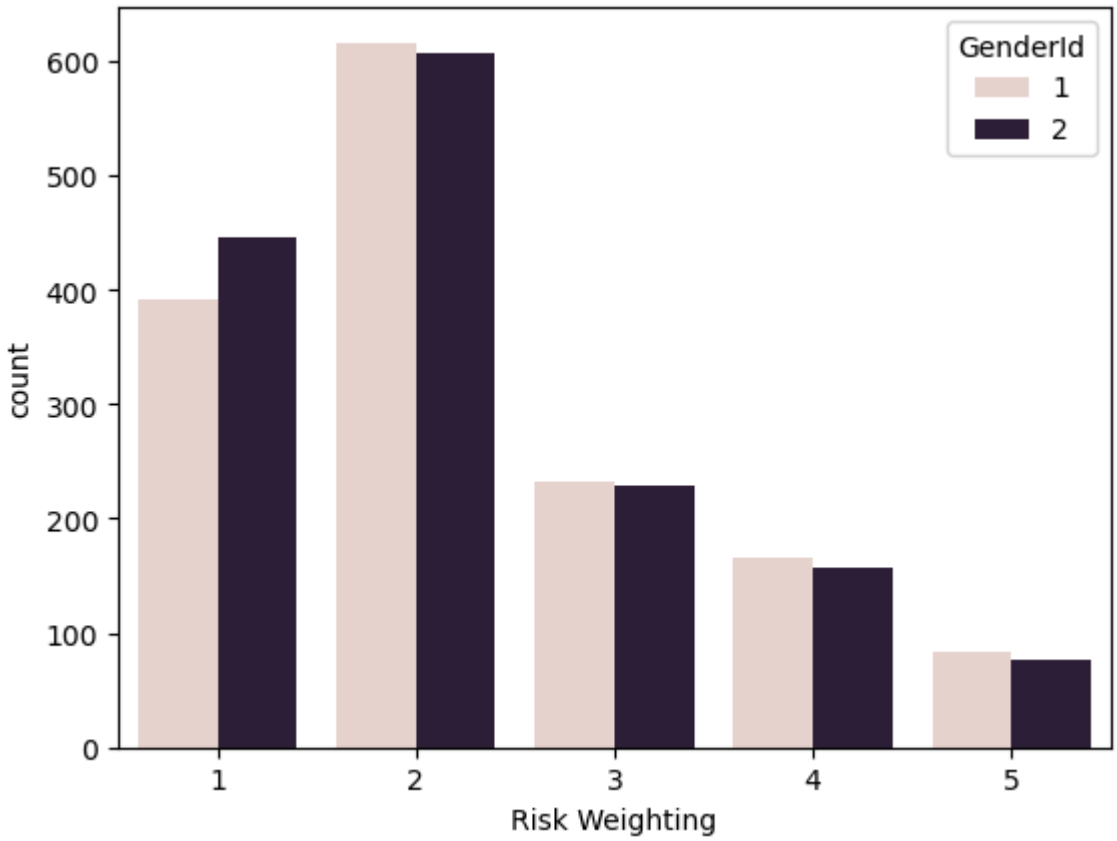

GenderId
1
2
700
600
500
400
300
200
100
0
count
American
African
European
Asian
Australian
Nationality

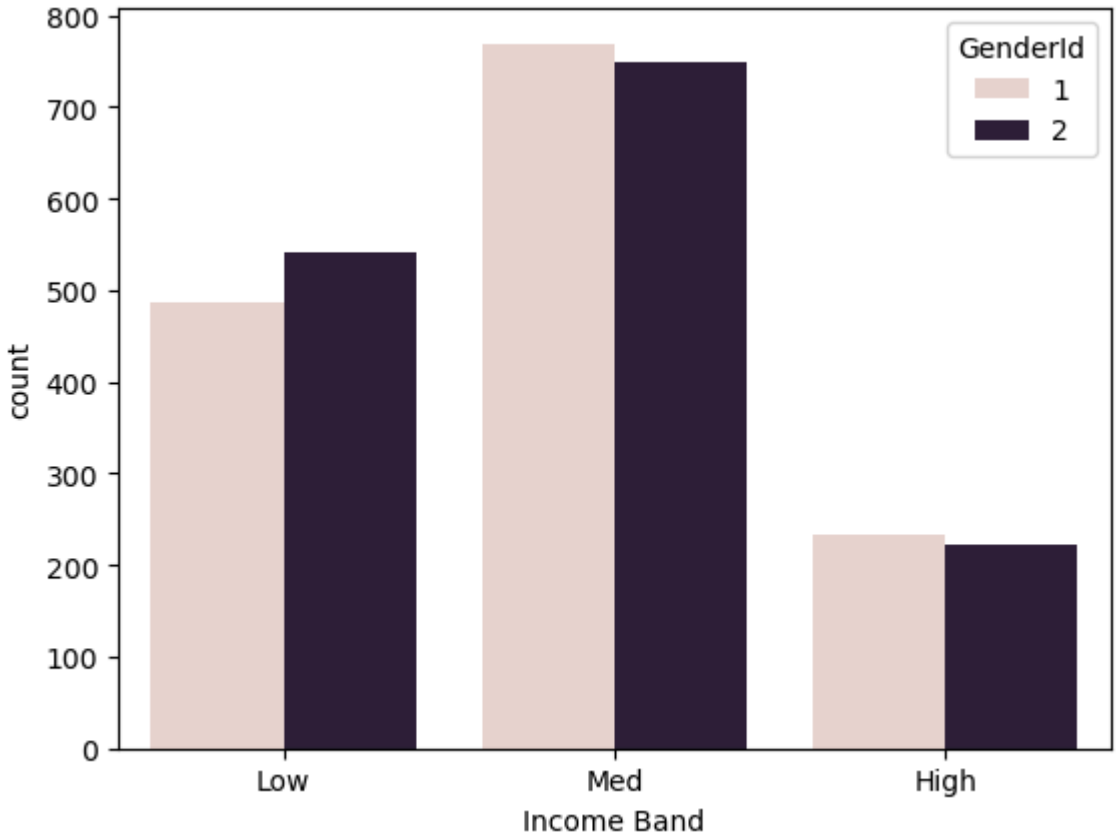

GenderId
1
2
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
count
Occupation


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

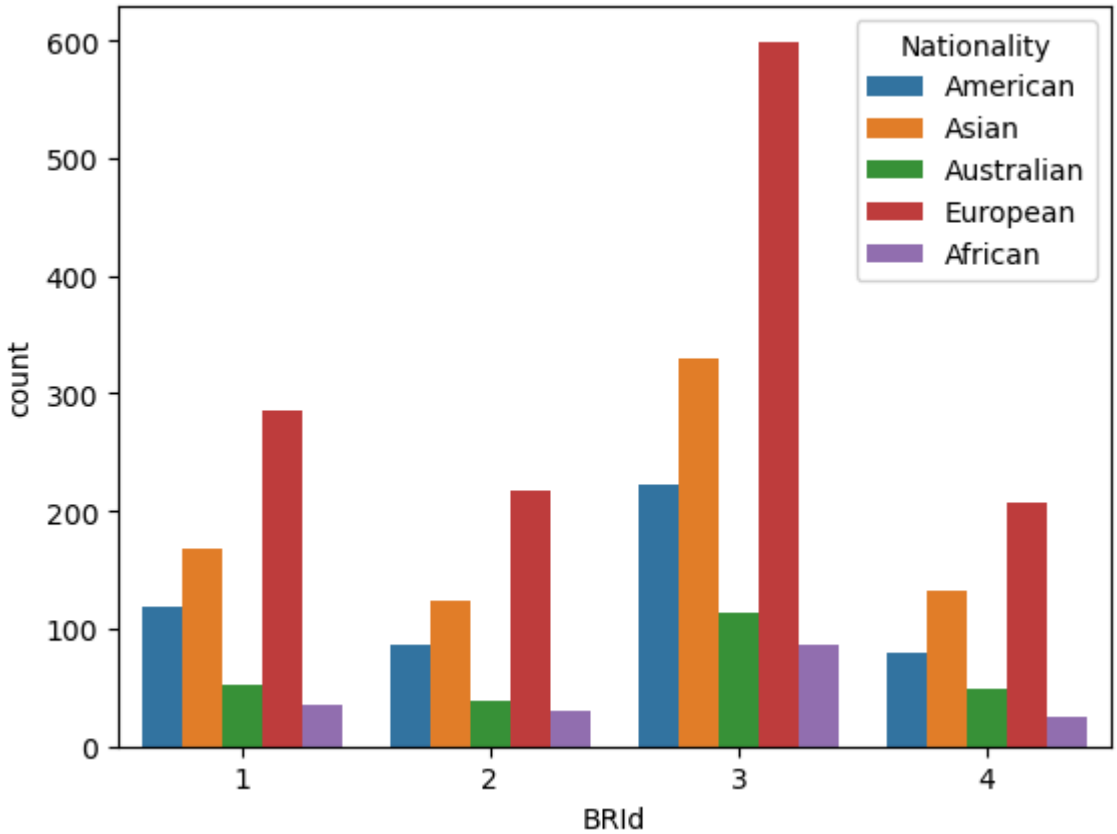

GenderId
1
2
count
0
100
200
300
400
500
600
700
Jade
Gold
Silver
Platinum
Loyalty Classification

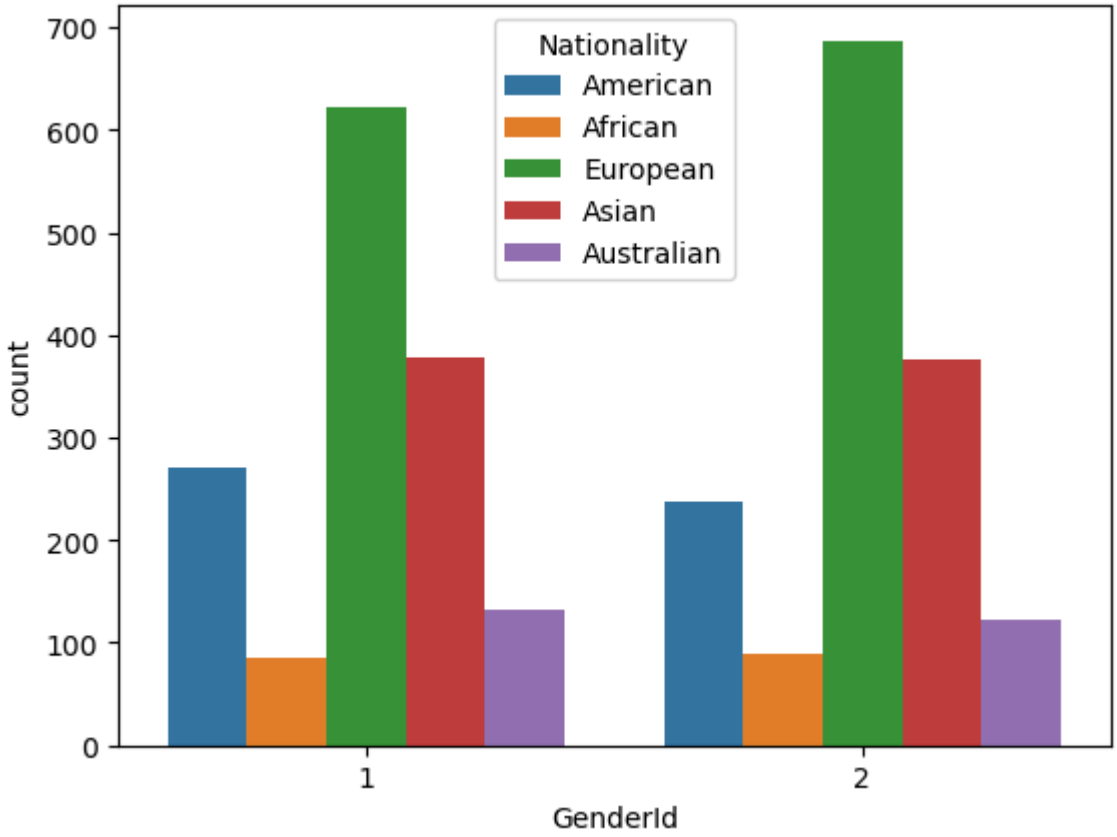

GenderId
1
2
count
Properties Owned
0
1
2
3
0
50
100
150
200
250
300
350
400

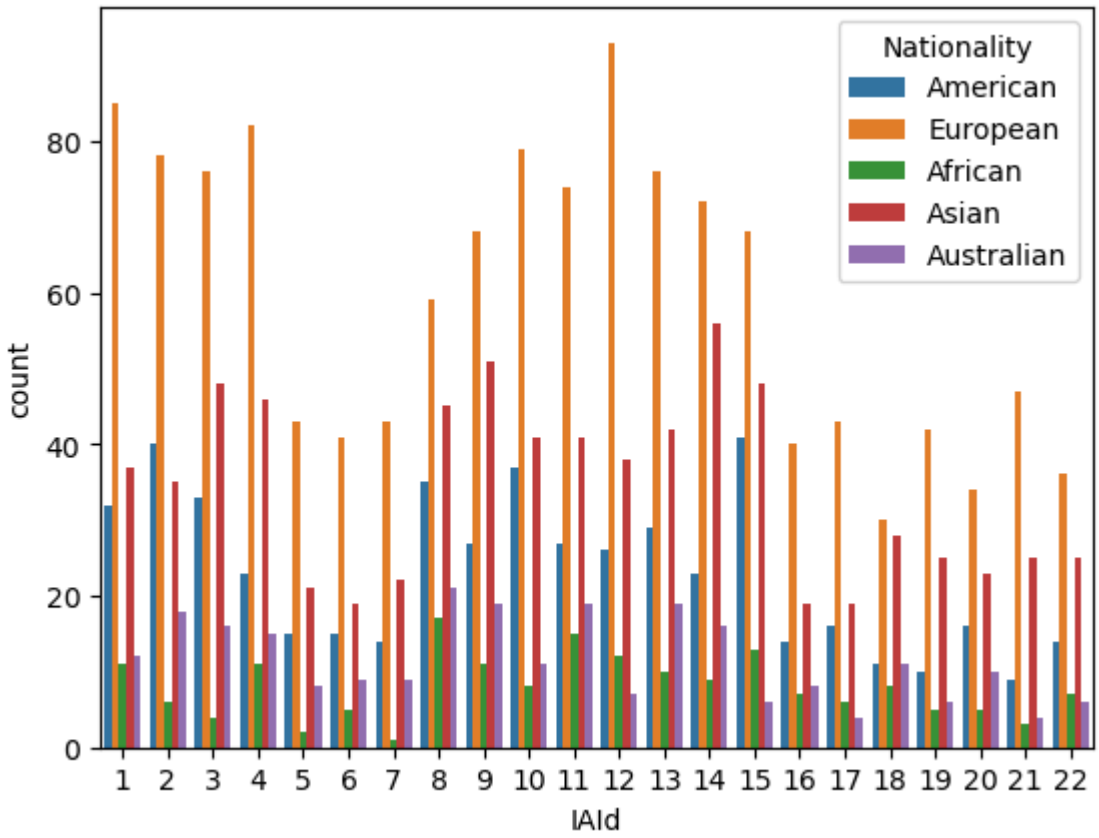

GenderId
1
2
count
Risk Weighting
1
2
3
4
5
0
100
200
300
400
500
600

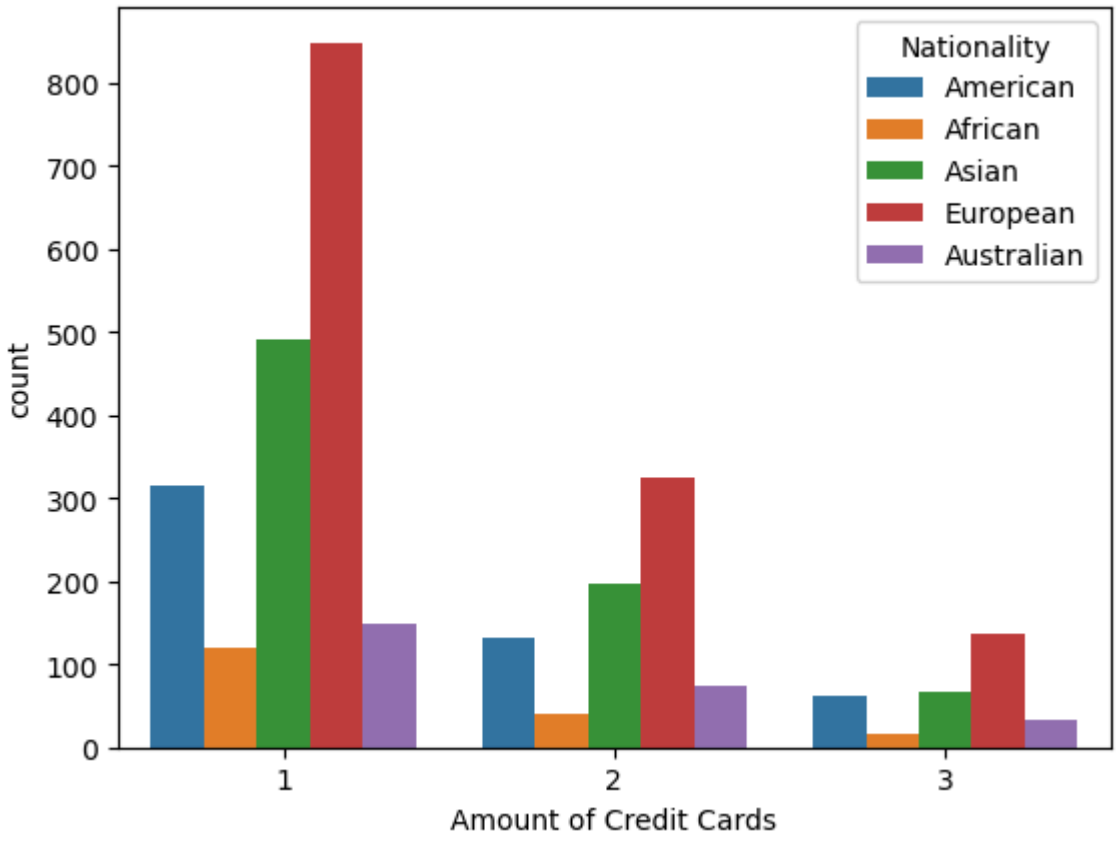


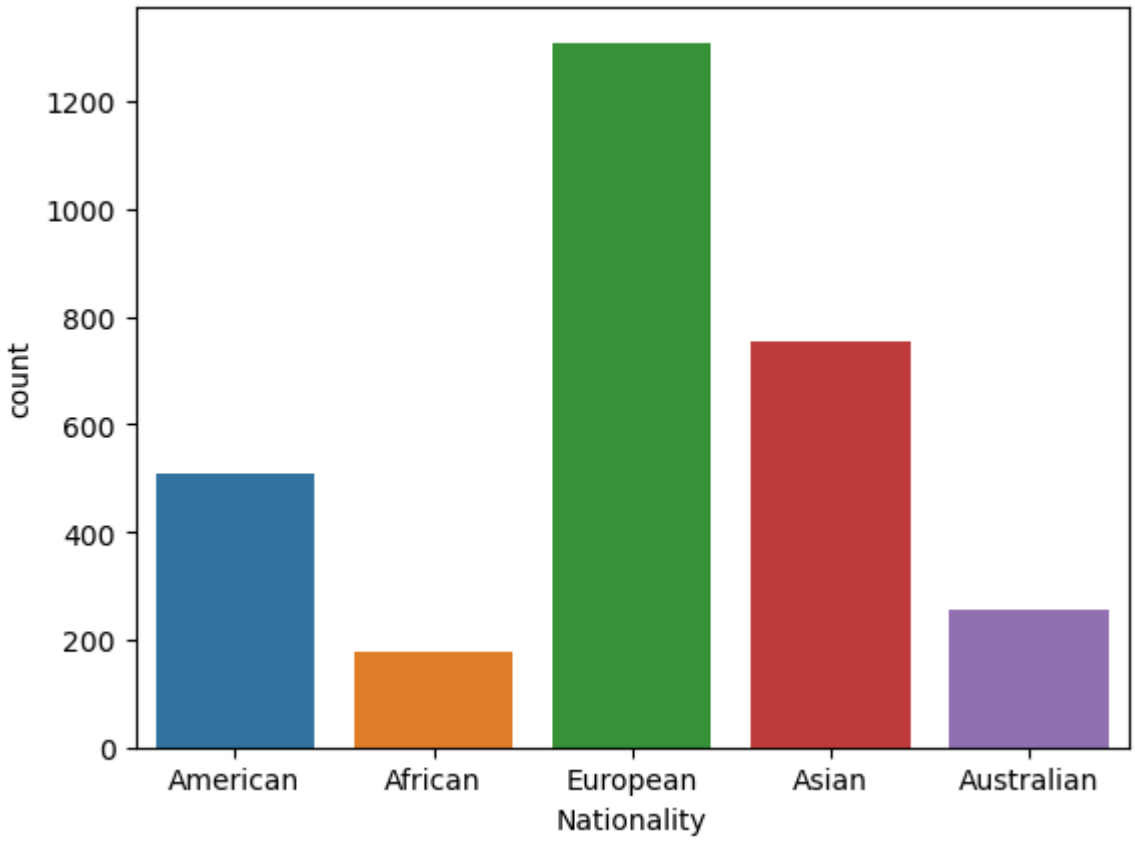
In [34]:
```
for i,predictor in enumerate(df[["BRId" , "GenderId", "IAId", "Amount of Credit
    plt.figure(i)
    sns.countplot(data=df, x=predictor, hue="Nationality")
```

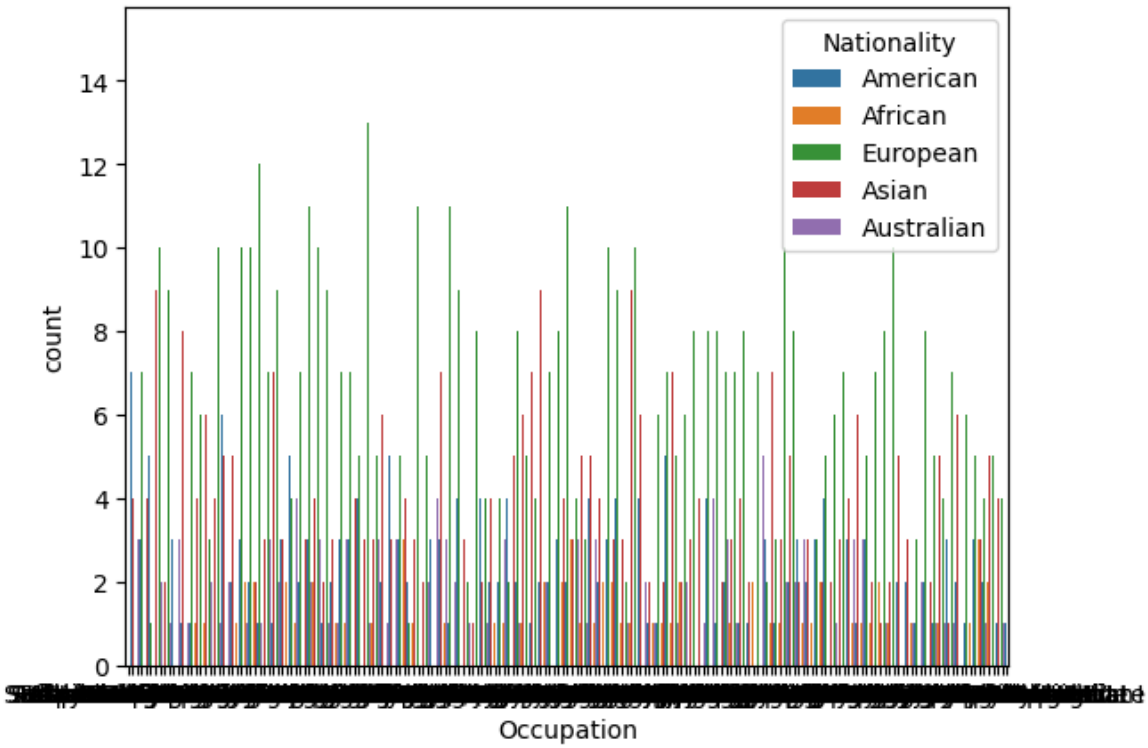


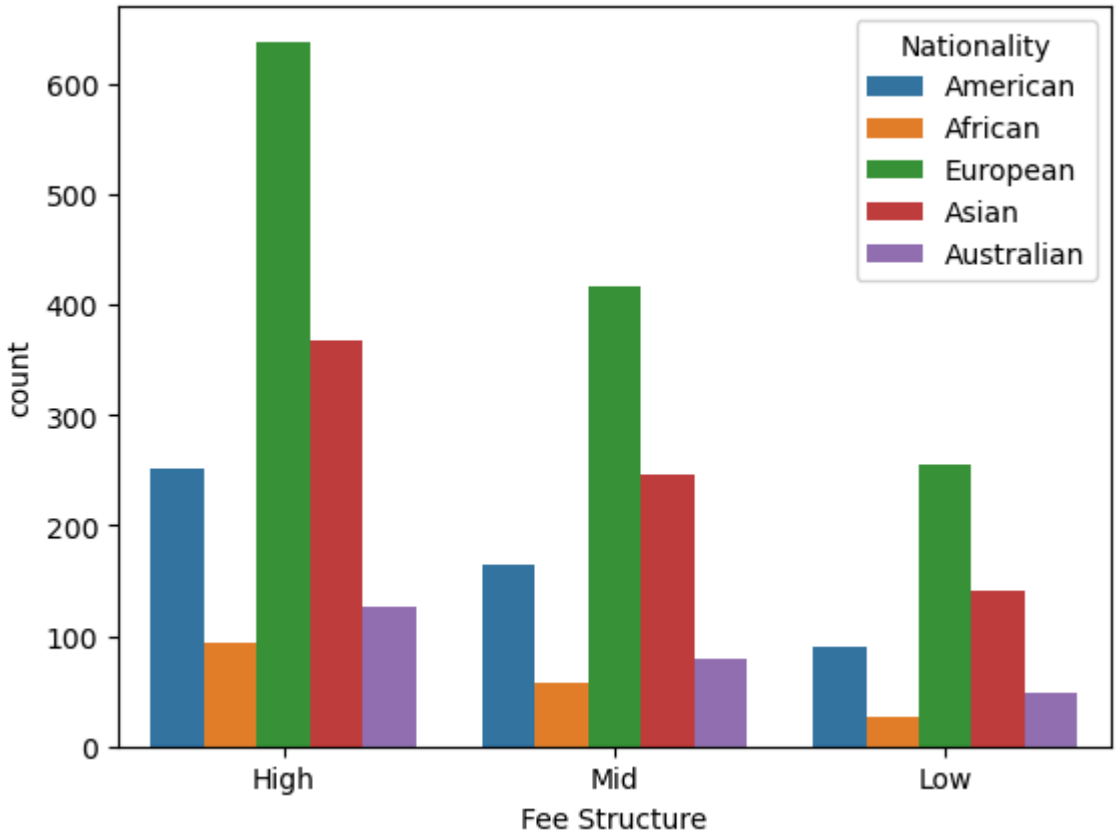

Nationality
American
African
European
Asian
Australian
count
0
100
200
300
400
500
600
700
1
2
GenderId

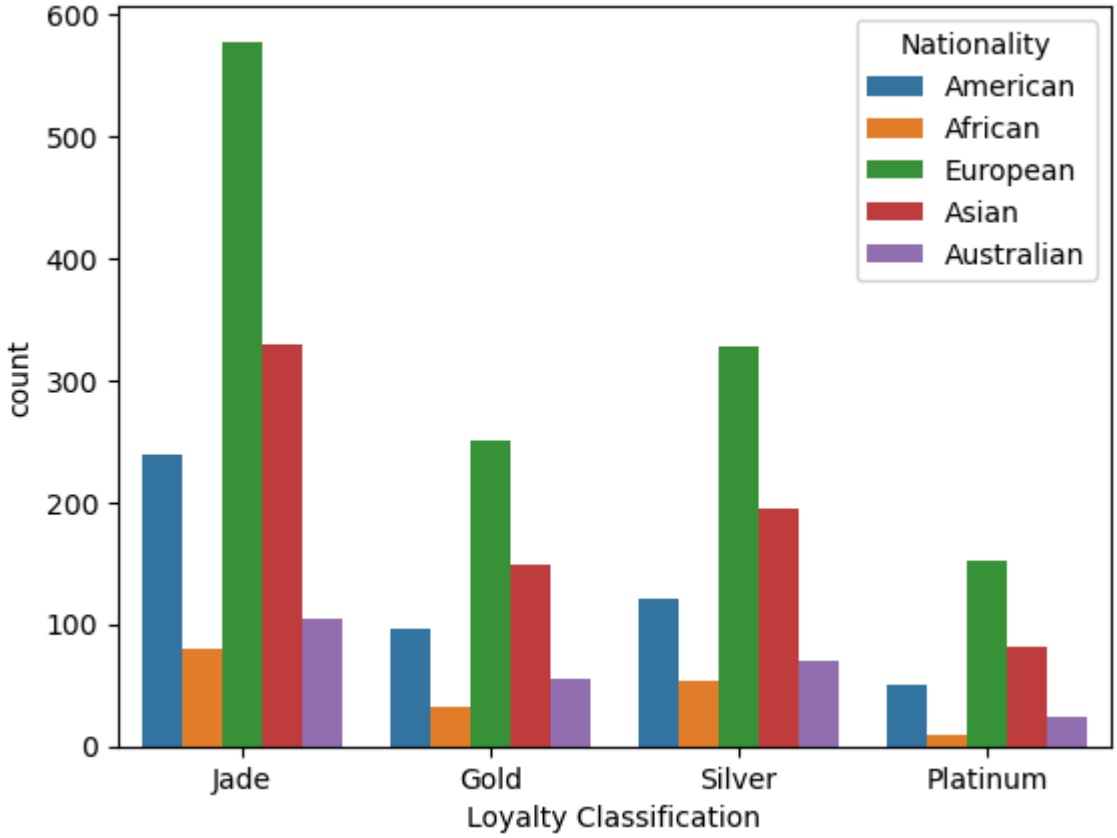

Nationality
American
European
African
Asian
Australian
count
0
20
40
60
80
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
IAId

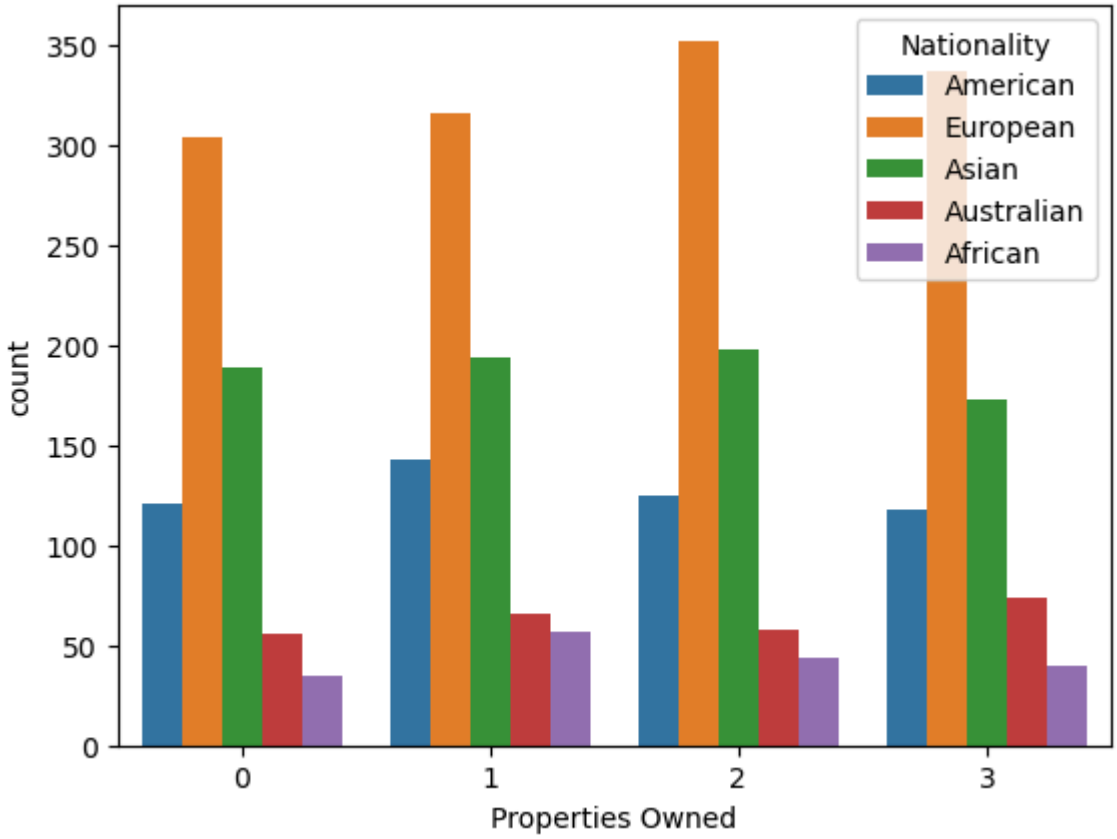

Nationality
American
African
Asian
European
Australian
count
0
100
200
300
400
500
600
700
800
1
2
3
Amount of Credit Cards

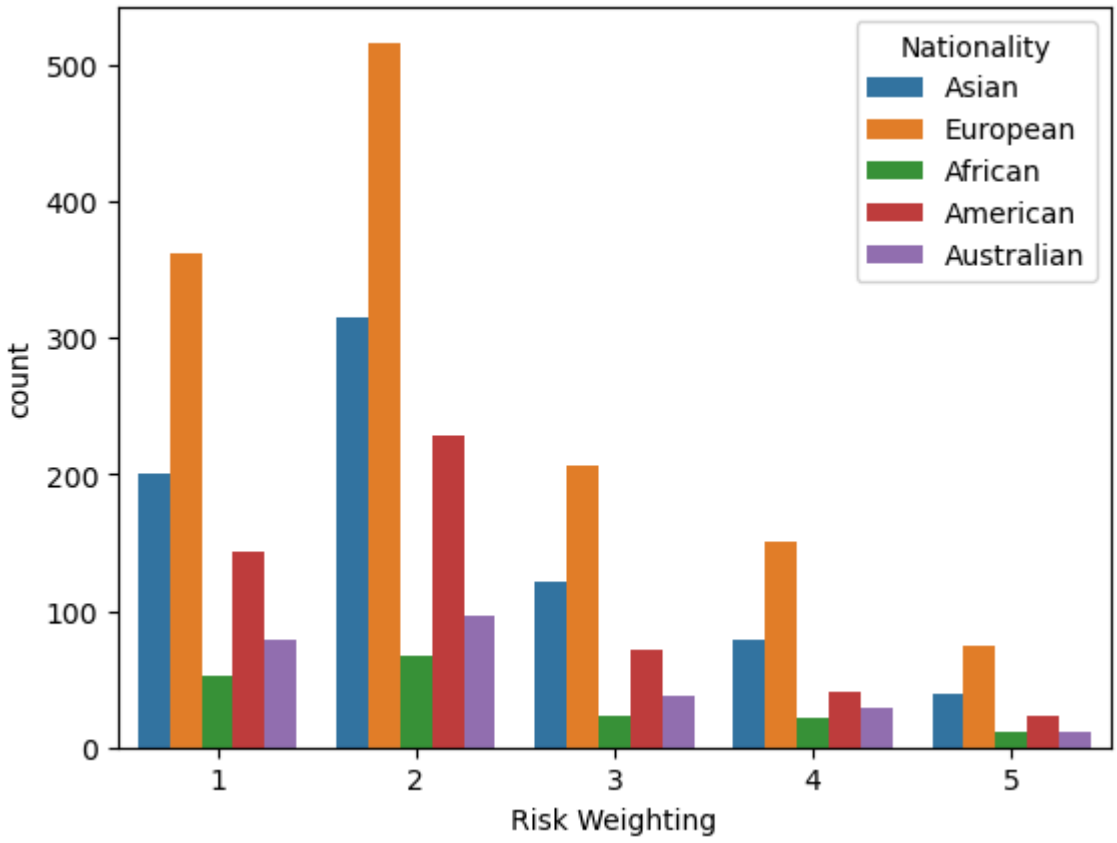

count
0
200
400
600
800
1000
1200
American
African
European
Asian
Australian
Nationality

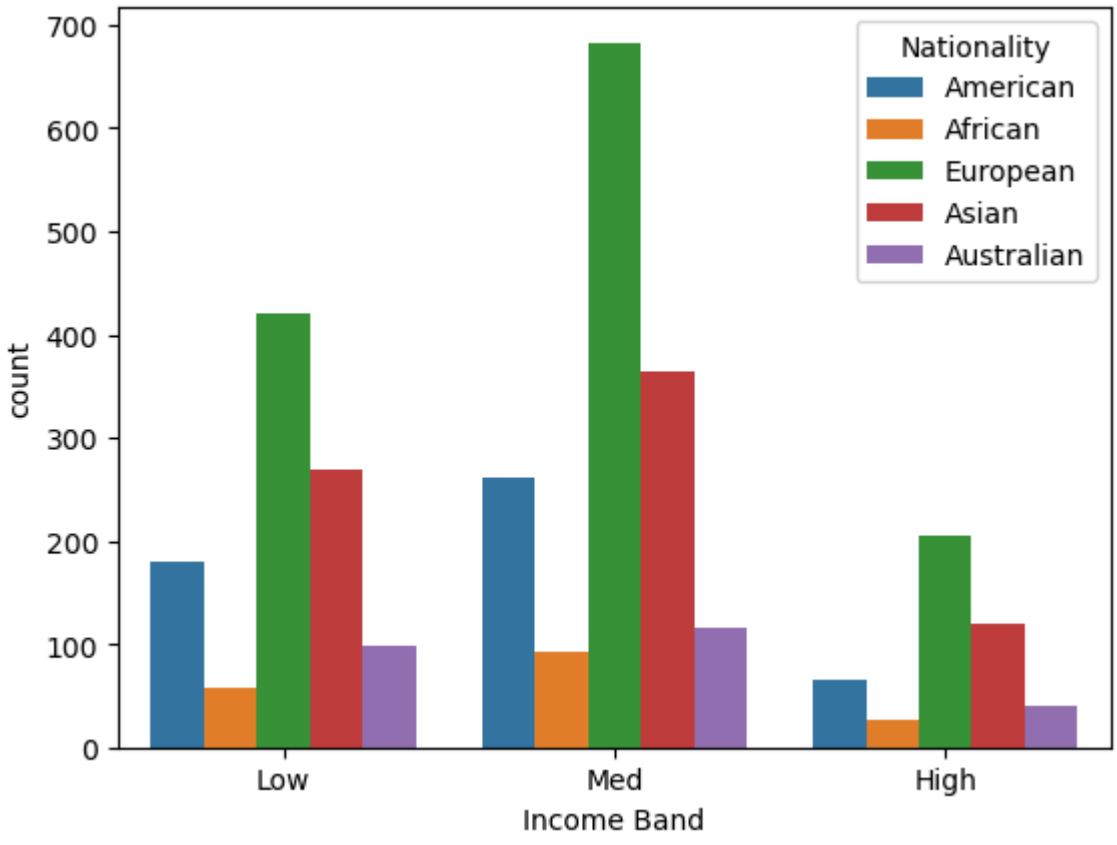

Nationality
American
African
European
Asian
Australian
count
0
2
4
6
8
10
12
14
Occupation


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
600
500
400
300
200
100
0
Jade
Gold
Silver
Platinum
Loyalty Classification

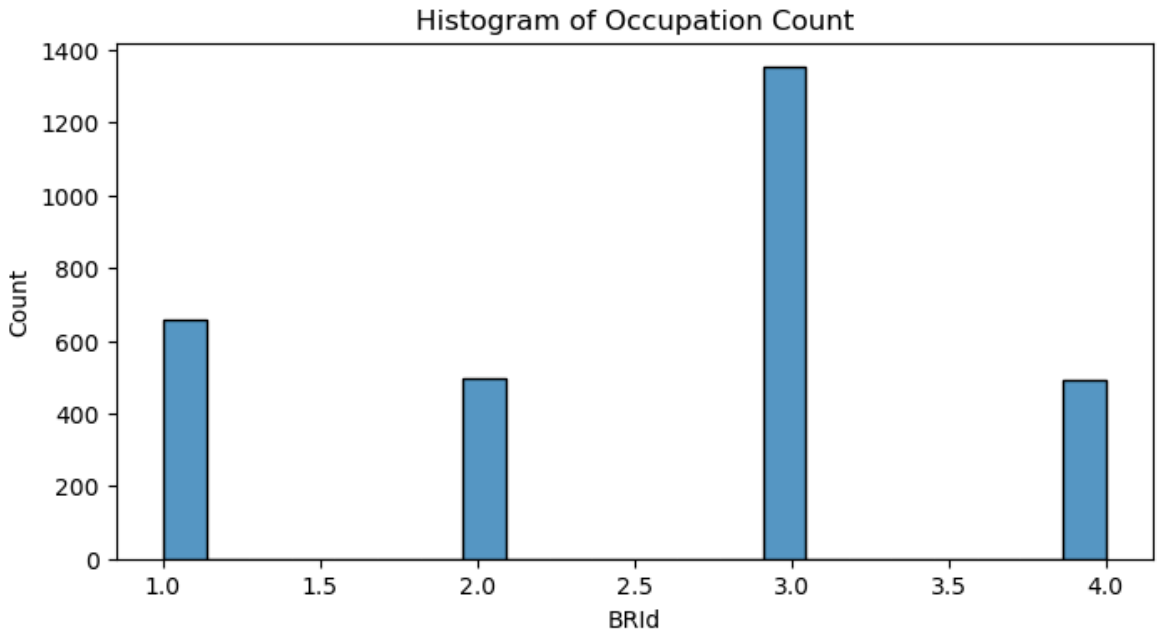

Nationality
American
European
Asian
Australian
African
count
350
300
250
200
150
100
50
0
0
1
2
3
Properties Owned

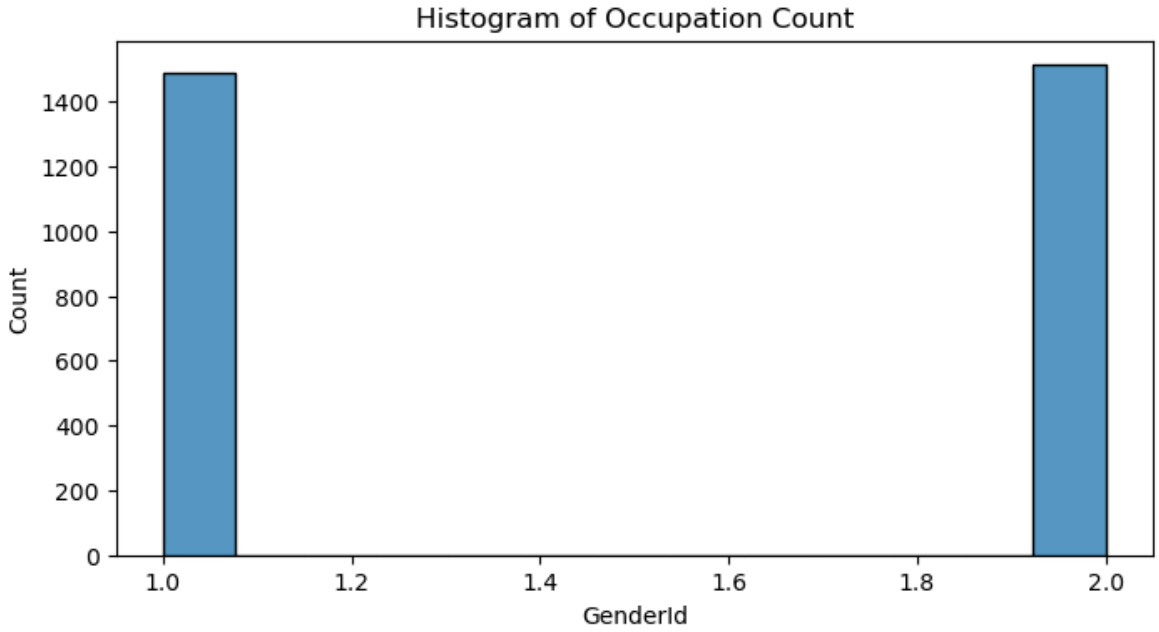


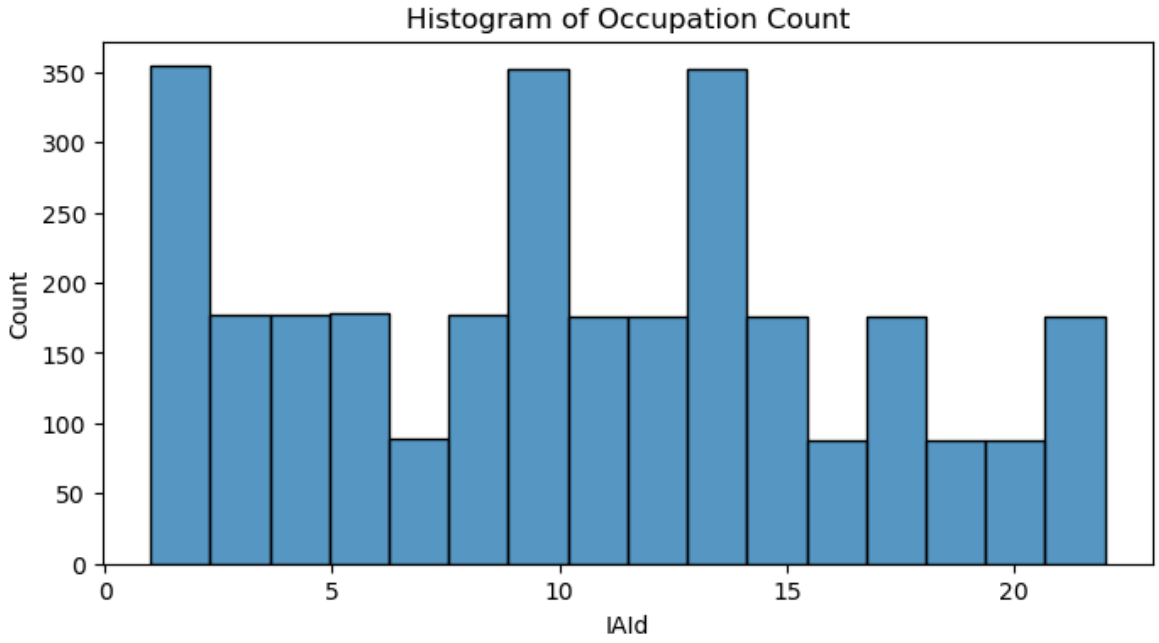


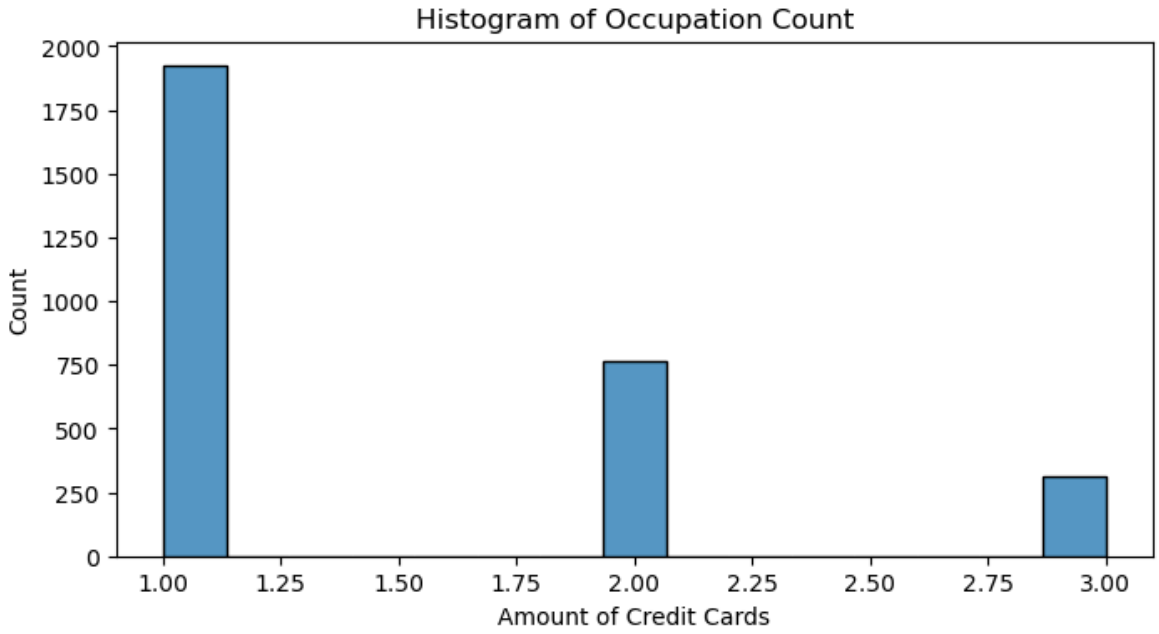
In [39]:
```
# hisplot of value count for different occupation
for col in categorial_cos:
 if col == "Occupation":
     continue
 plt.figure(figsize=(8,4))
 sns.histplot(df[col])
 plt.title('Histogram of Occupation Count')
 plt.xlabel(col)
```

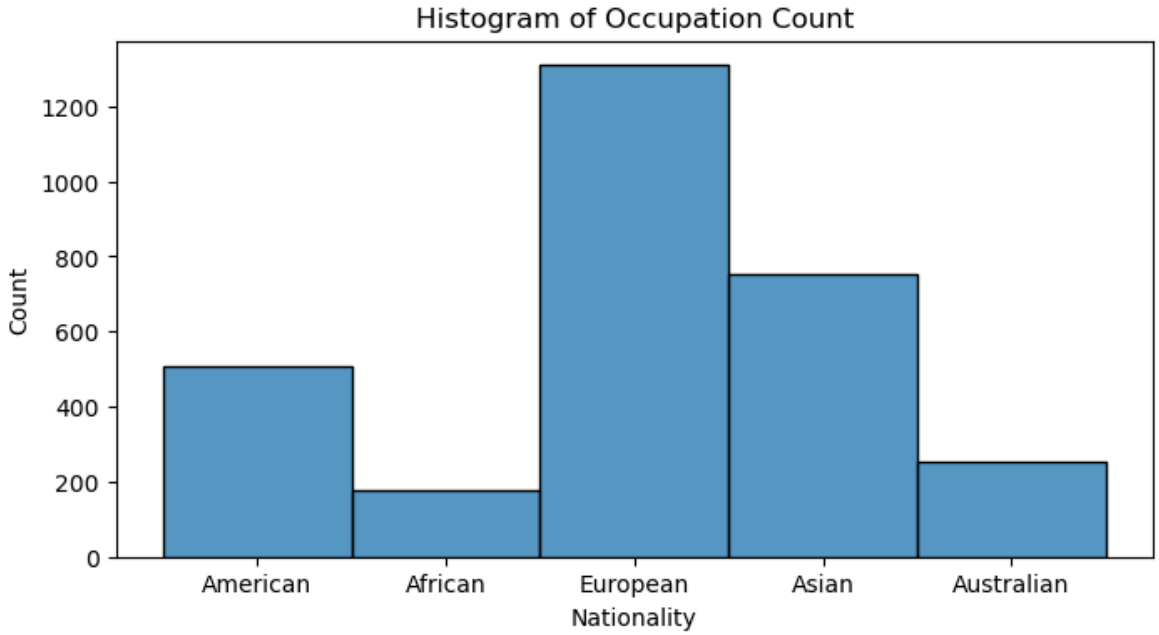
```
plt.ylabel("Count")
plt.show()
```

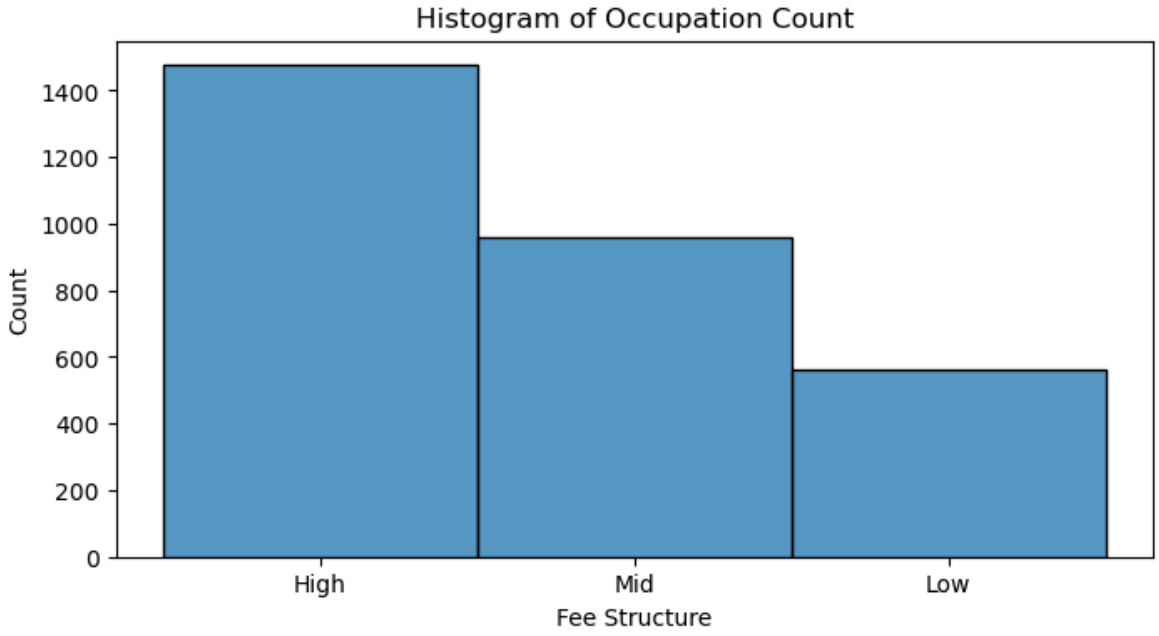


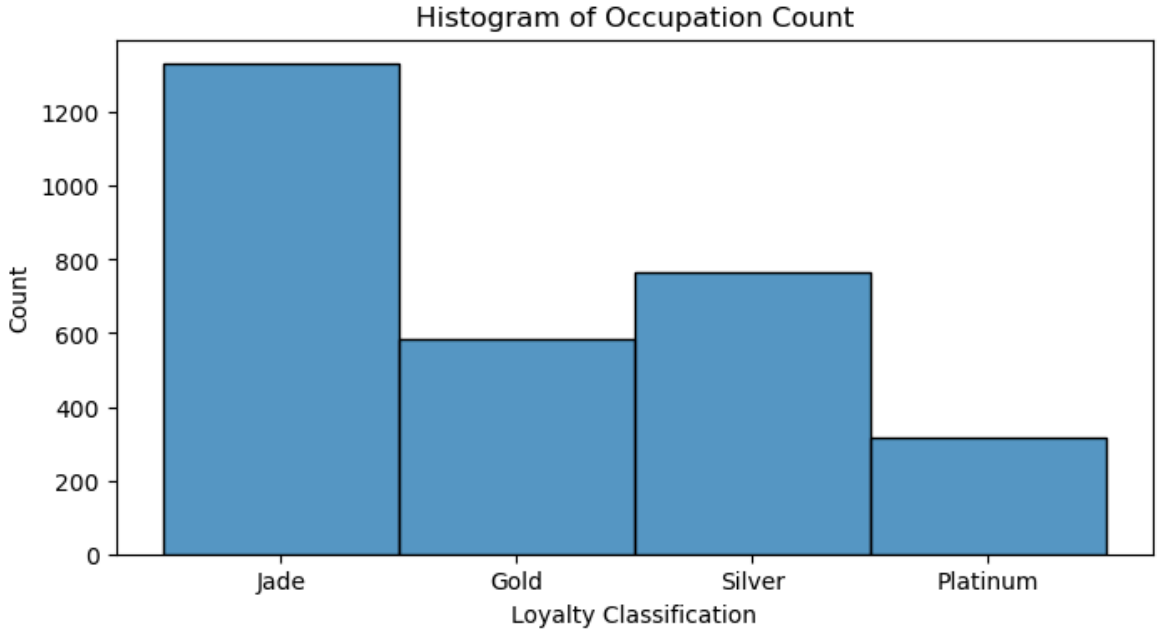


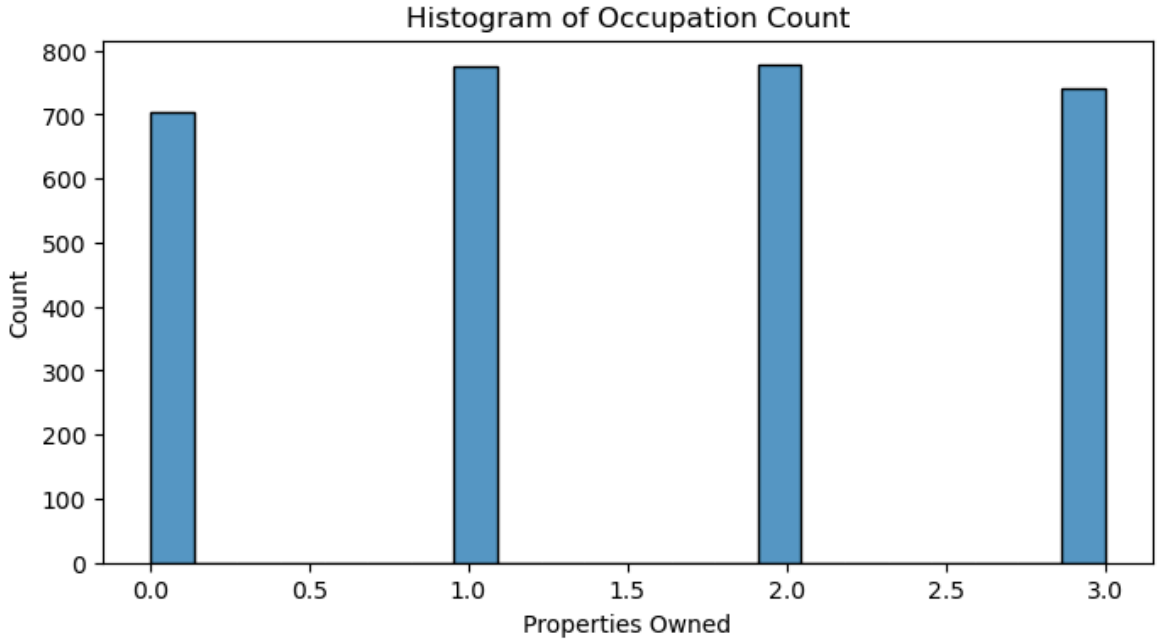

Histogram of Occupation Count
Count
IAId
0
50
100
150
200
250
300
350
0
5
10
15
20

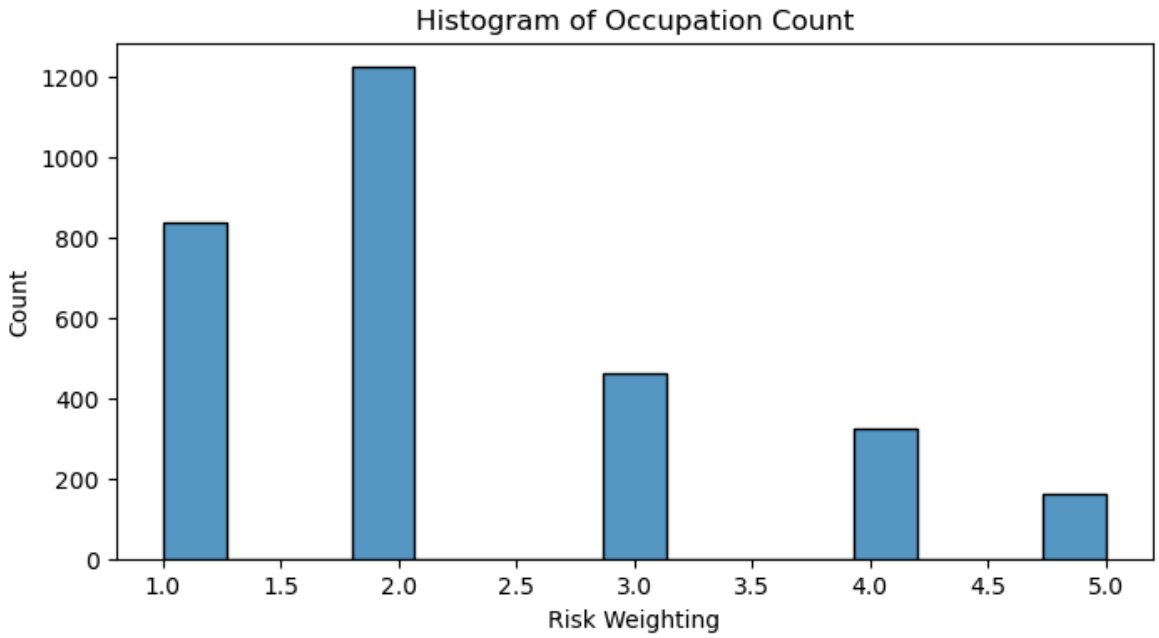

Histogram of Occupation Count
Count
Amount of Credit Cards
0
250
500
750
1000
1250
1500
1750
2000
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00

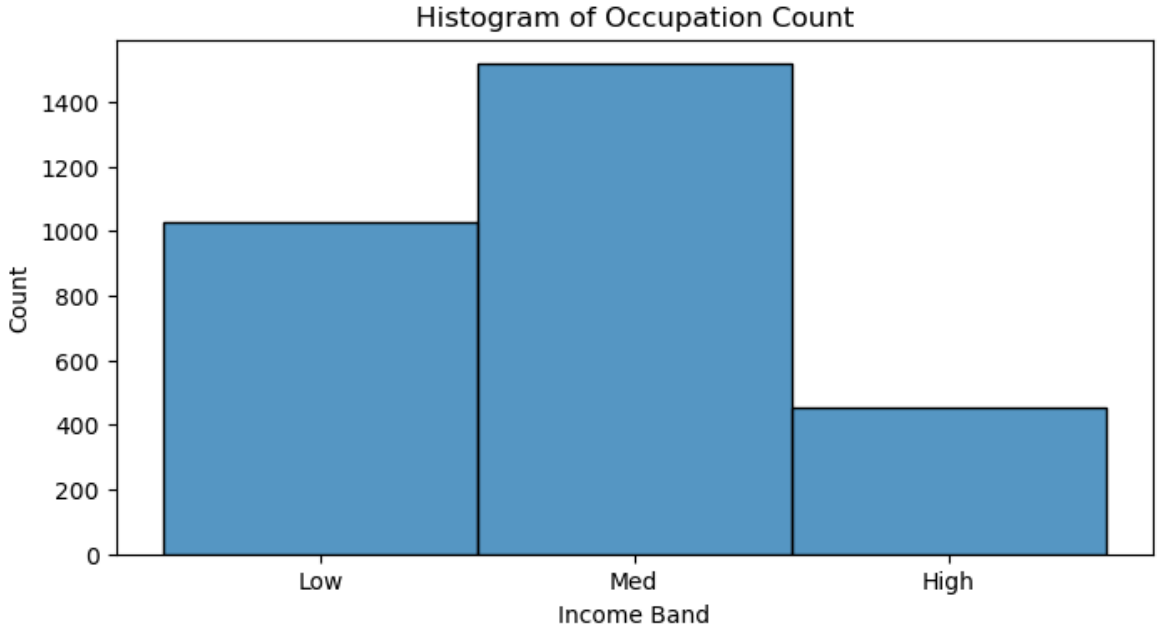

Histogram of Occupation Count
Count
Nationality
0
200
400
600
800
1000
1200
American
African
European
Asian
Australian


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


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

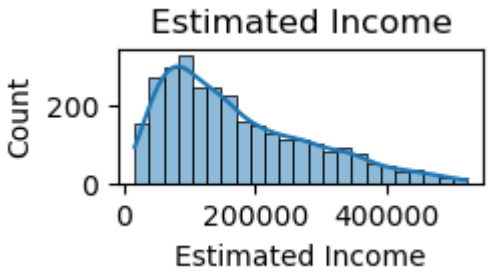

Histogram of Occupation Count
Count
0
100
200
300
400
500
600
700
800
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Properties Owned

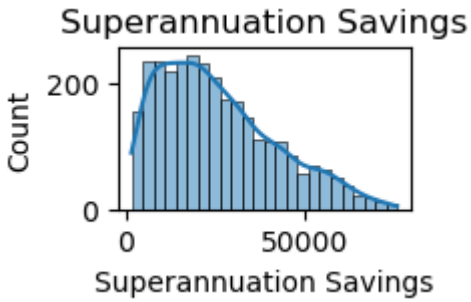


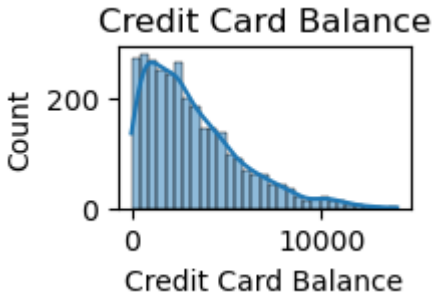


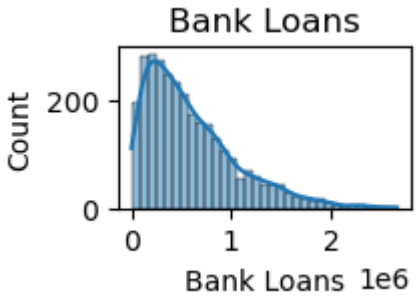
Numerical Analysis

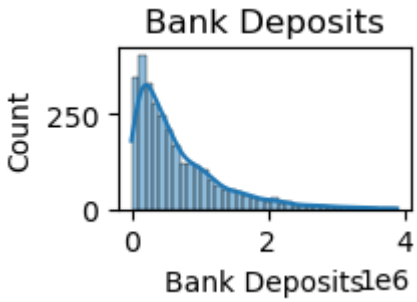
In [61]:
```
  numerical_cos =  ['Estimated Income', 'Superannuation Savings', 'Credit Card

  plt.figure(figsize=(8,4))
  for i,col in enumerate(numerical_cos):
   plt.subplot(4,3,i+1)
   sns.histplot(df[col],kde=True)
   plt.title(col)
plt.show()
```

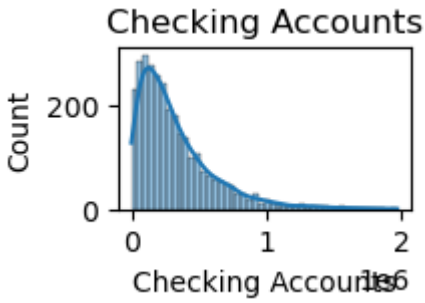

Estimated Income
Count
200
0
0
200000
400000
Estimated Income

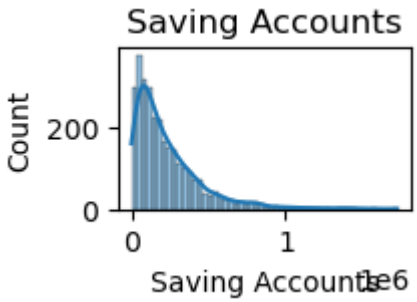

Superannuation Savings
Count
200
0
0
50000
Superannuation Savings

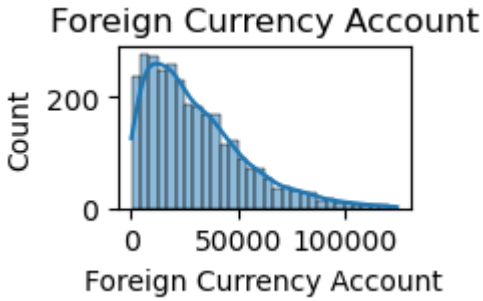

Credit Card Balance
Count
200
0
0
10000
Credit Card Balance

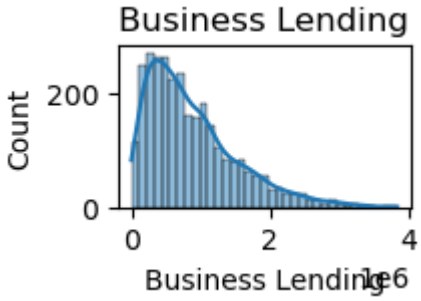

Bank Loans
Count
200
0
0
1
2
Bank Loans
1e6


Bank Deposits
Count
250
0
0
2
4
Bank Deposits
1e6


Checking Accounts
Count
200
0
0
1
2
Checking Accounts
1e6







HeatMaps

In [62]:
```
 numerical_cos =  ['Estimated Income', 'Superannuation Savings', 'Credit Card Ba
correlation_matrix = df[numerical_cos].corr()
plt.figure(figsize=(12,12))
sns.heatmap(correlation_matrix, annot=True, cmap='crest', fmt=" 2f")
plt.title("correlation_matrix")
plt.show()
```


correlation_matrix
Estimated Income
Superannuation Savings
Credit Card Balance
Bank Loans
Bank Deposits
Checking Accounts
Saving Accounts
Foreign Currency Account
Business Lending
1.000000 0.374802 0.298527 0.329926 0.260332 0.291412 0.261299 0.306999 0.328531
0.374802 1.000000 0.225562 0.241457 0.174084 0.198188 0.177132 0.228103 0.264919
0.298527 0.225562 1.000000 0.369509 0.383877 0.298672 0.284696 0.357720 0.351063
0.329926 0.241457 0.369509 1.000000 0.373155 0.292082 0.268882 0.364391 0.417095
0.260332 0.174084 0.383877 0.373155 1.000000 0.844278 0.754744 0.406347 0.441298
0.291412 0.198188 0.298672 0.292082 0.844278 1.000000 0.459509 0.312651 0.355904
0.261299 0.177132 0.284696 0.268882 0.754744 0.459509 1.000000 0.311465 0.307550
0.306999 0.228103 0.357720 0.364391 0.406347 0.312651 0.311465 1.000000 0.369749
0.328531 0.264919 0.351063 0.417095 0.441298 0.355904 0.307550 0.369749 1.000000
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

In [ ]: