```
#pymysql connector
```

```
pip install mysql-connector-python
```

```
Requirement already satisfied: mysql-connector-python in d:\anaconda\
lib\site-packages (9.3.0)
Note: you may need to restart the kernel to use updated packages.
```

```
import mysql.connector as connection
```

```
# Connect to server
cnx = connection.connect(
    host="127.0.0.1",
    port=3306,
    user="root",
    password="PkdSql@246")
```

```
query = "SELECT * FROM bank_case.customer"
```

```
import pandas as pd
```

```
df = pd.read_sql(query, cnx)
```

```
C:\Users\prate\AppData\Local\Temp\ipykernel_3396\2703287818.py:3:
UserWarning: pandas only supports SQLAlchemy connectable
(engine/connection) or database string URI or sqlite3 DBAPI2
connection. Other DBAPI2 objects are not tested. Please consider using
SQLAlchemy.
  df = pd.read_sql(query, cnx)
```

```
cnx.close()
```

```
df.head(5)
```

```
  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   

    Banking Contact Nationality             Occupation Fee Structure  \
0    Anthony Torres    American  Safety Technician IV          High   
1  Jonathan Hawkins     African   Software Consultant          High   
2     Anthony Berry    European    Help Desk Operator          High   
3        Steve Diaz    American          Geologist II           Mid   
4        Shawn Long    American   Assistant Professor           Mid   

  Loyalty Classification  ...  Bank Deposits  Checking Accounts  \
0                   Jade  ...     1485828.64          603617.88   
1                   Jade  ...      641482.79          229521.37   
2                   Gold  ...     1033401.59          652674.69   
```

```
3                 Silver  ...     1048157.49         1048157.49
4               Platinum  ...      487782.53          446644.25

   Saving Accounts  Foreign Currency Account  Business Lending  \
0        607332.46                  12249.96        1134475.30   
1        344635.16                  61162.31        2000526.10   
2        203054.35                  79071.78         548137.58   
3        234685.02                  57513.65        1148402.29   
4        128351.45                  30012.14        1674412.12   

   Properties Owned  Risk Weighting  BRId  GenderId  IAId  
0                 1               2     1         1     1  
1                 1               3     2         1     2  
2                 1               3     3         2     3  
3                 0               4     4         1     4  
4                 0               3     1         2     5  

[5 rows x 25 columns]
```

```
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
```

```python
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```python
df.shape
```

```
(3000, 25)
```

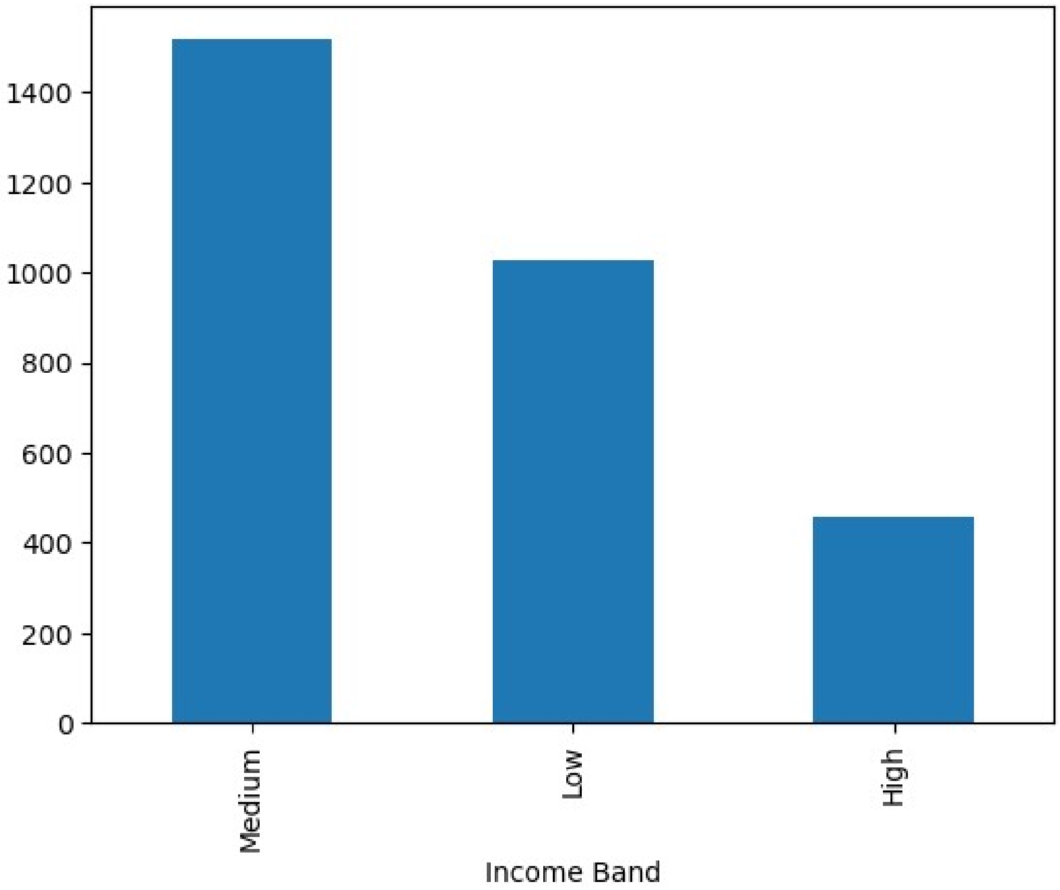
```python
#Generate Descriptive Stsstistics for the dataframe
df.describe()
```

```
               Age   Location ID  Estimated Income  Superannuation
Savings  \
count  3000.000000   3000.000000       3000.000000
3000.000000
mean     51.039667  21563.323000     171305.034263
25531.599673
std      19.854760  12462.273017     111935.808209
16259.950770
min      17.000000     12.000000      15919.480000
1482.030000
25%      34.000000  10803.500000      82906.595000
12513.775000
50%      51.000000  21129.500000     142313.480000
22357.355000
75%      69.000000  32054.500000     242290.305000
35464.740000
max      85.000000  43369.000000     522330.260000
75963.900000

       Amount of Credit Cards  Credit Card Balance    Bank Loans  \
count             3000.000000          3000.000000  3.000000e+03
mean                 1.463667          3176.206943  5.913862e+05
std                  0.676387          2497.094709  4.575570e+05
min                  1.000000             1.170000  0.000000e+00
25%                  1.000000          1236.630000  2.396281e+05
50%                  1.000000          2560.805000  4.797934e+05
75%                  2.000000          4522.632500  8.258130e+05
max                  3.000000         13991.990000  2.667557e+06

       Bank Deposits  Checking Accounts  Saving Accounts  \
count   3.000000e+03       3.000000e+03     3.000000e+03
mean    6.715602e+05       3.210929e+05     2.329084e+05
std     6.457169e+05       2.820796e+05     2.300078e+05
min     0.000000e+00       0.000000e+00     0.000000e+00
25%     2.044004e+05       1.199475e+05     7.479440e+04
50%     4.633165e+05       2.428157e+05     1.640866e+05
```

```
75%     9.427546e+05       4.348749e+05     3.155750e+05
max     3.890598e+06       1.969923e+06     1.724118e+06

       Foreign Currency Account  Business Lending  Properties Owned  \
count               3000.000000      3.000000e+03       3000.000000
mean               29883.529993      8.667598e+05          1.518667
std                23109.924010      6.412303e+05          1.102145
min                   45.000000      0.000000e+00          0.000000
25%                11916.542500      3.748251e+05          1.000000
50%                24341.190000      7.113147e+05          2.000000
75%                41966.392500      1.185110e+06          2.000000
max               124704.870000      3.825962e+06          3.000000

       Risk Weighting         BRId     GenderId         IAId
count     3000.000000  3000.000000  3000.000000  3000.000000
mean         2.249333     2.559333     1.504000    10.425333
std          1.131191     1.007713     0.500067     5.988242
min          1.000000     1.000000     1.000000     1.000000
25%          1.000000     2.000000     1.000000     5.000000
50%          2.000000     3.000000     2.000000    10.000000
75%          3.000000     3.000000     2.000000    15.000000
max          5.000000     4.000000     2.000000    22.000000
```

```
groups = [0,100000, 300000, float('inf')]
label = ['Low', 'Medium', 'High']

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

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

```
<Axes: xlabel='Income Band'>
```

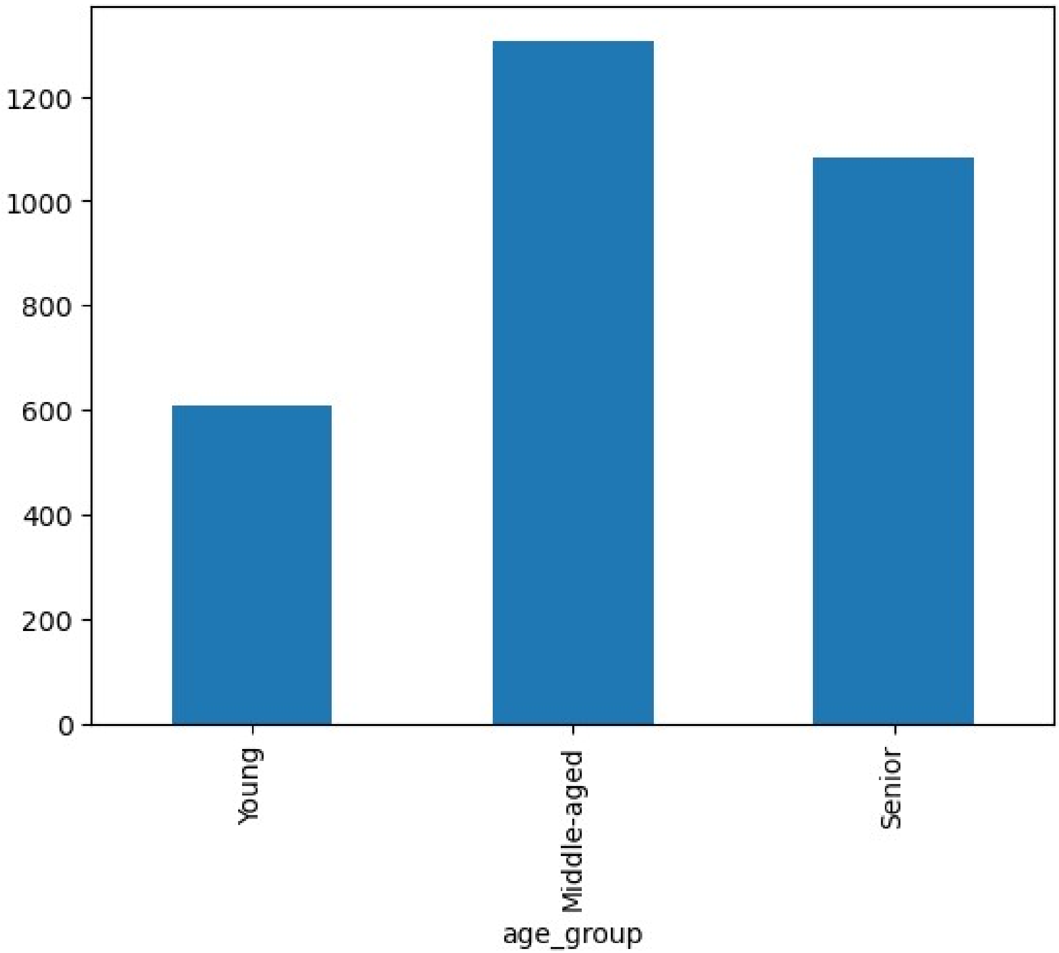


```
bins = [15, 30, 60, 100]
labels = ['Young', 'Middle Aged', 'Old Aged']

df['age_group'] = pd.cut(df['Age'], bins = bins, labels = label,
right=True, include_lowest=True)

age_group_counts = df['age_group'].value_counts().sort_index()
print(age_group_counts)
```

```
age_group
Young           610
Middle-aged    1306
Senior         1084
Name: count, dtype: int64
```

```
age_group_counts.plot(kind='bar')
```

```
<Axes: xlabel='age_group'>
```



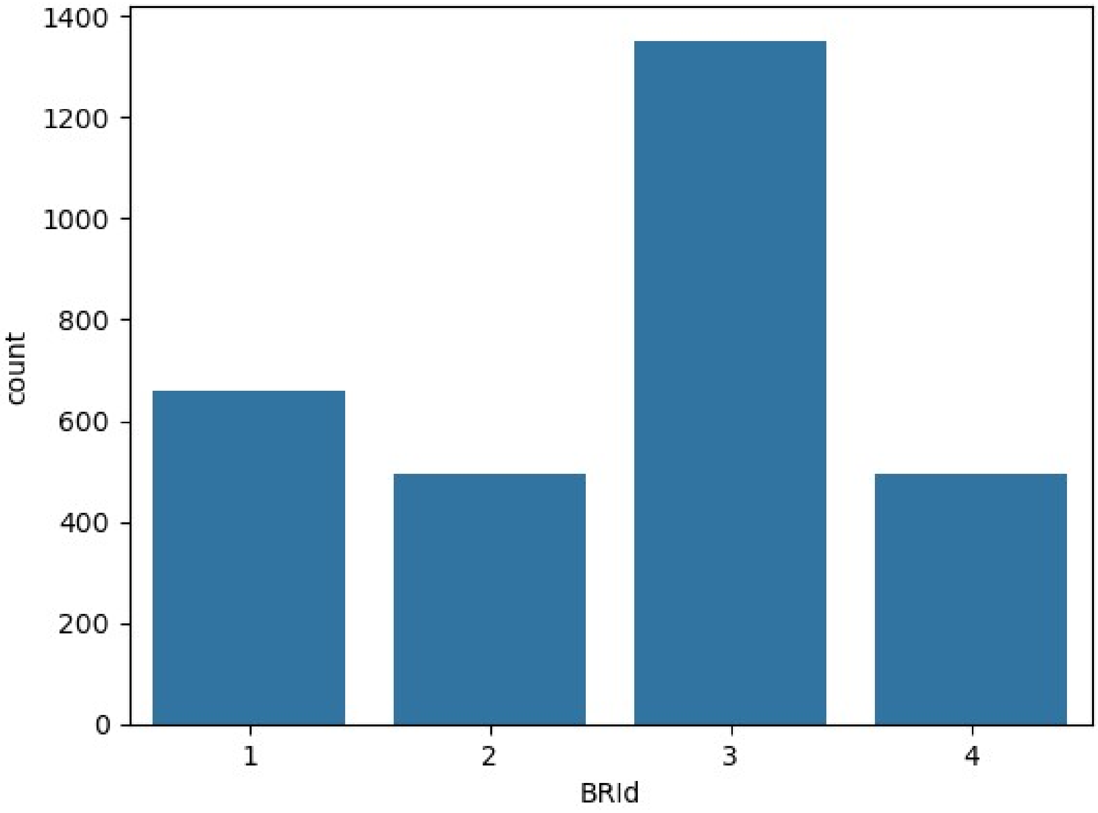
```
#Examine distribution of unique categories in categorical columns
categorical_cols = df[["BRId", "GenderId", "Amount of Credit Cards",
"Nationality", "Occupation", "Fee Structure", "Loyalty
Classification", "Properties Owned", "Risk Weighting", "IAId", "Income
Band"]].columns

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

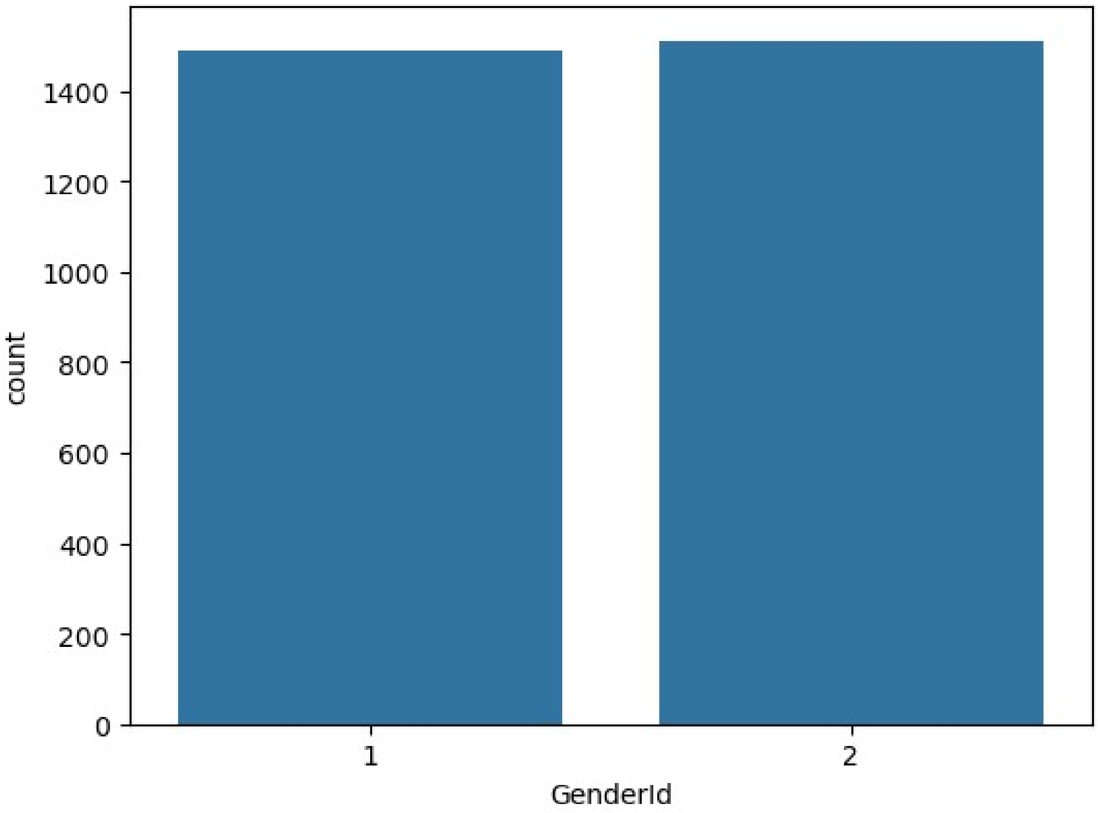
```
Value count for 'BRId':

BRId
3    1352
1     660
2     495
4     493
Name: count, dtype: int64

Value count for 'GenderId':
```

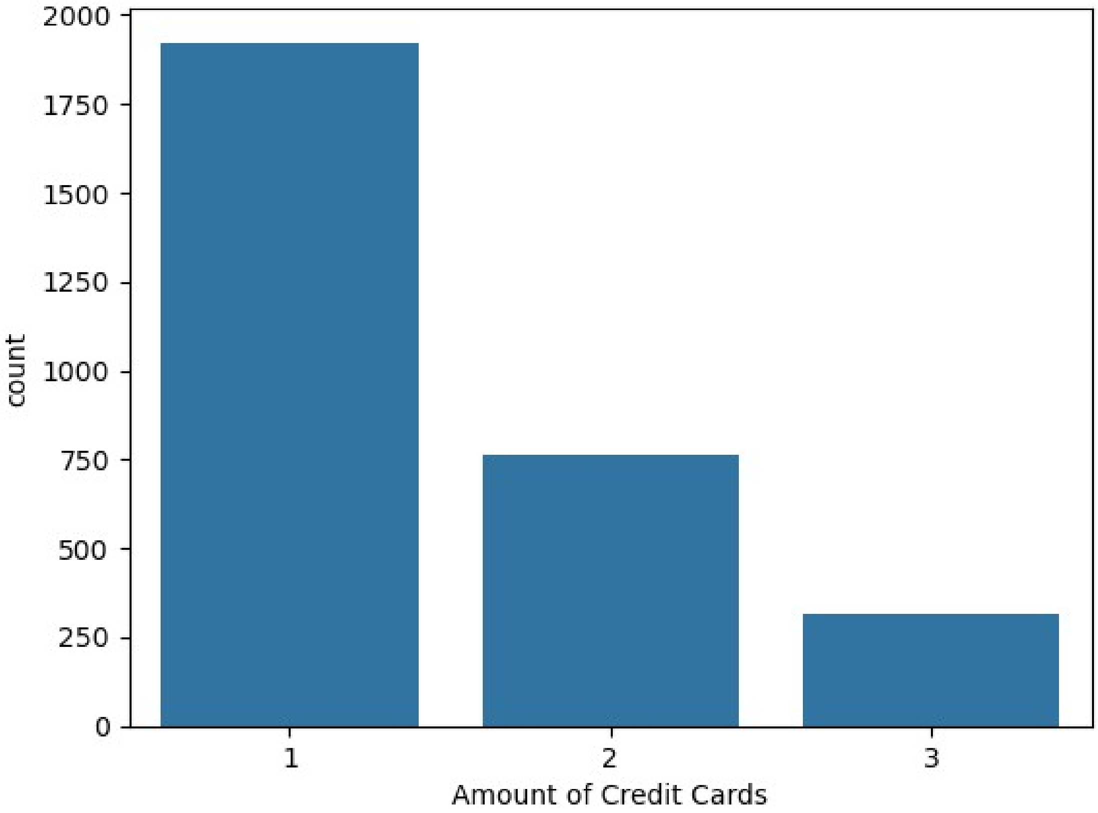
```
GenderId
2    1512
1    1488
Name: count, dtype: int64

Value count for 'Amount of Credit Cards':

Amount of Credit Cards
1    1922
2     765
3     313
Name: count, dtype: int64

Value count for 'Nationality':

Nationality
European      1309
Asian          754
American       507
Australian     254
African        176
Name: count, dtype: int64

Value count 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 count for 'Fee Structure':

Fee Structure
High    1476
Mid      962
Low      562
Name: count, dtype: int64

Value count for 'Loyalty Classification':

Loyalty Classification
Jade        1331
```

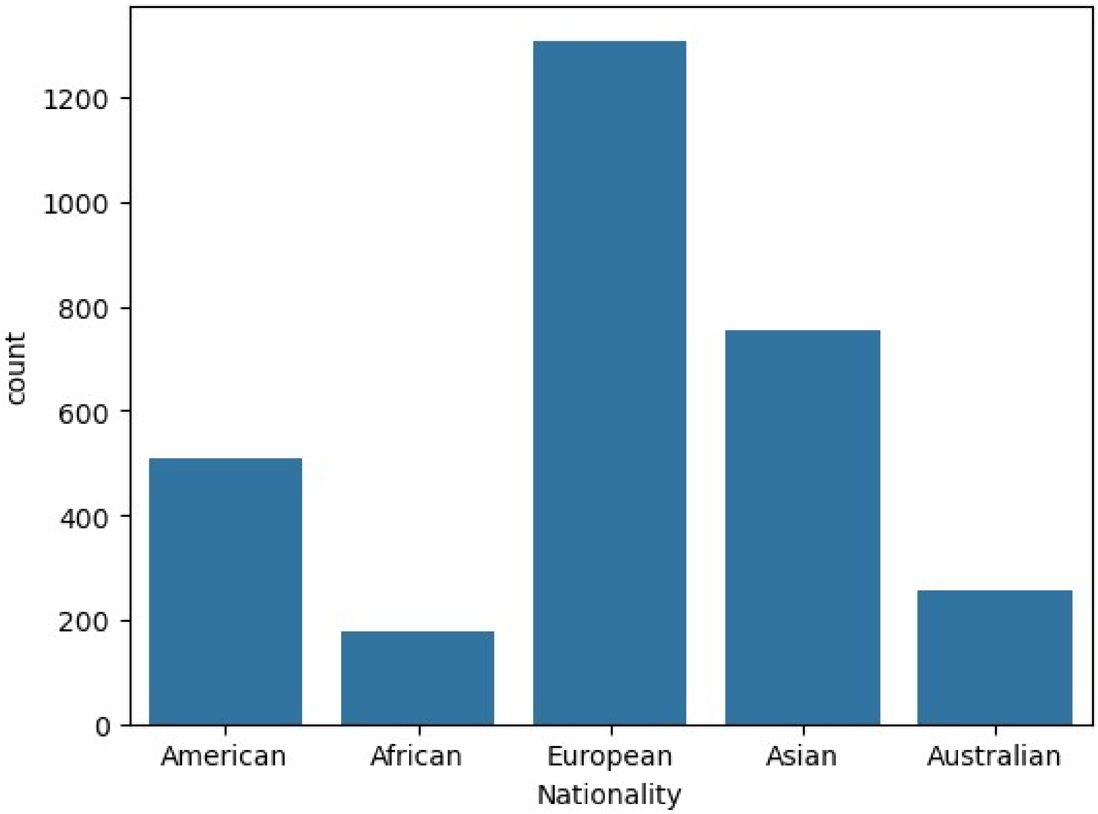
```
Silver       767
Gold         585
Platinum     317
Name: count, dtype: int64

Value count for 'Properties Owned':

Properties Owned
2    777
1    776
3    742
0    705
Name: count, dtype: int64

Value count for 'Risk Weighting':

Risk Weighting
2    1222
1     836
3     460
4     322
5     160
Name: count, dtype: int64

Value count 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
```

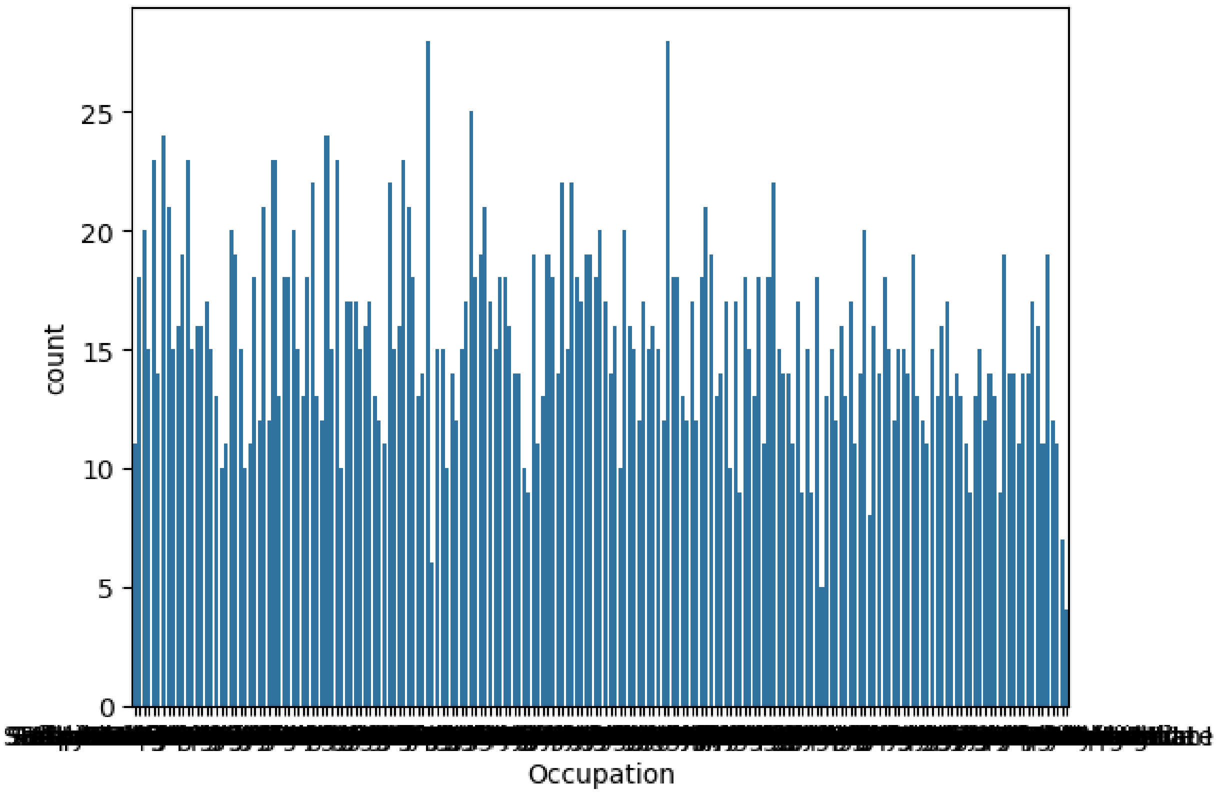
```
Value count for 'Income Band':

Income Band
Medium    1517
Low       1027
High       456
Name: count, dtype: int64

#Univariate Analysis

for i, predictor in enumerate(df[["BRId", "GenderId", "Amount of
Credit Cards", "Nationality", "Occupation", "Fee Structure", "Loyalty
Classification", "Properties Owned", "Risk Weighting", "IAId", "Income
Band"]].columns
):
    plt.figure(i)
    sns.countplot(data = df, x = predictor)
```

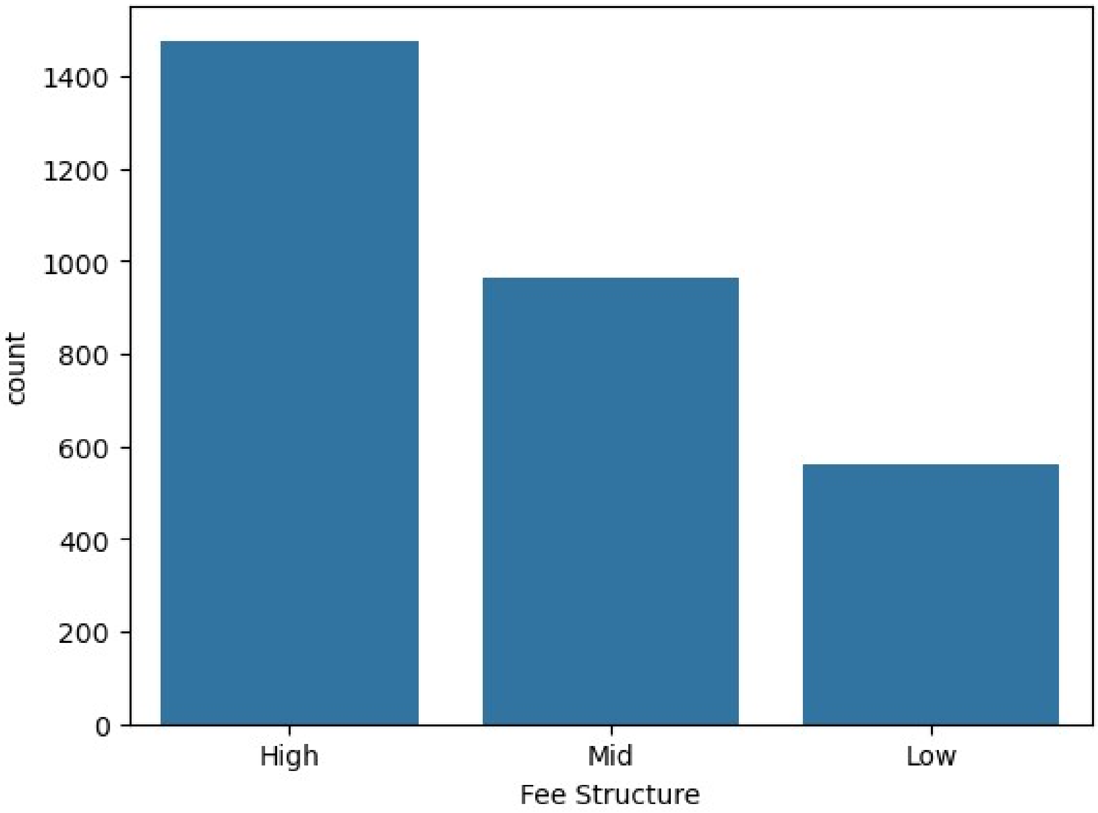


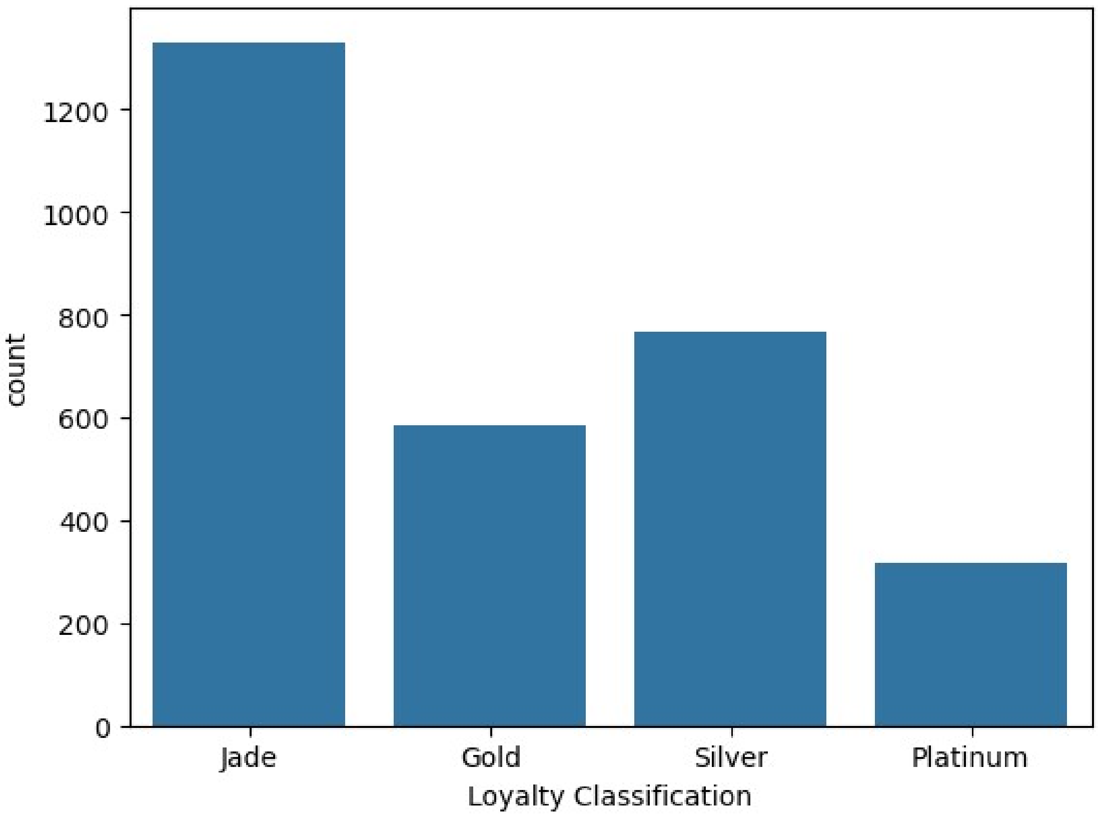

count
1400
1200
1000
800
600
400
200
0
1
2
GenderId

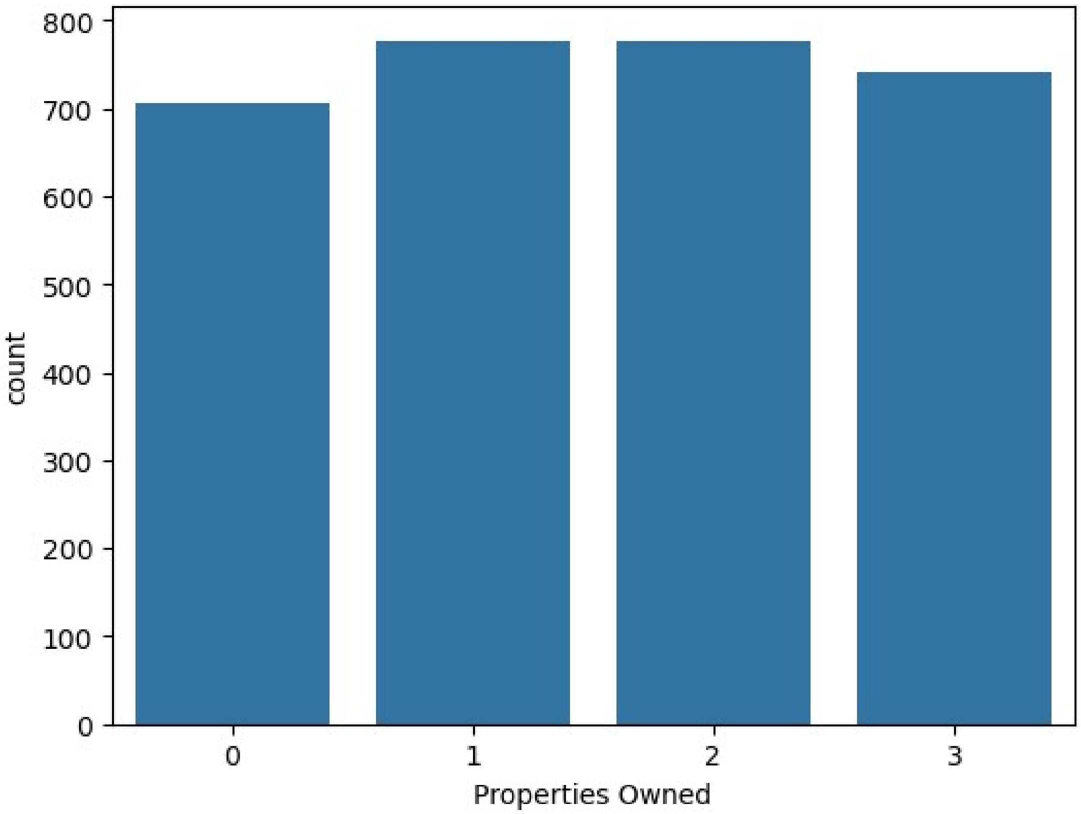

count
2000
1750
1500
1250
1000
750
500
250
0
1
2
3
Amount of Credit Cards

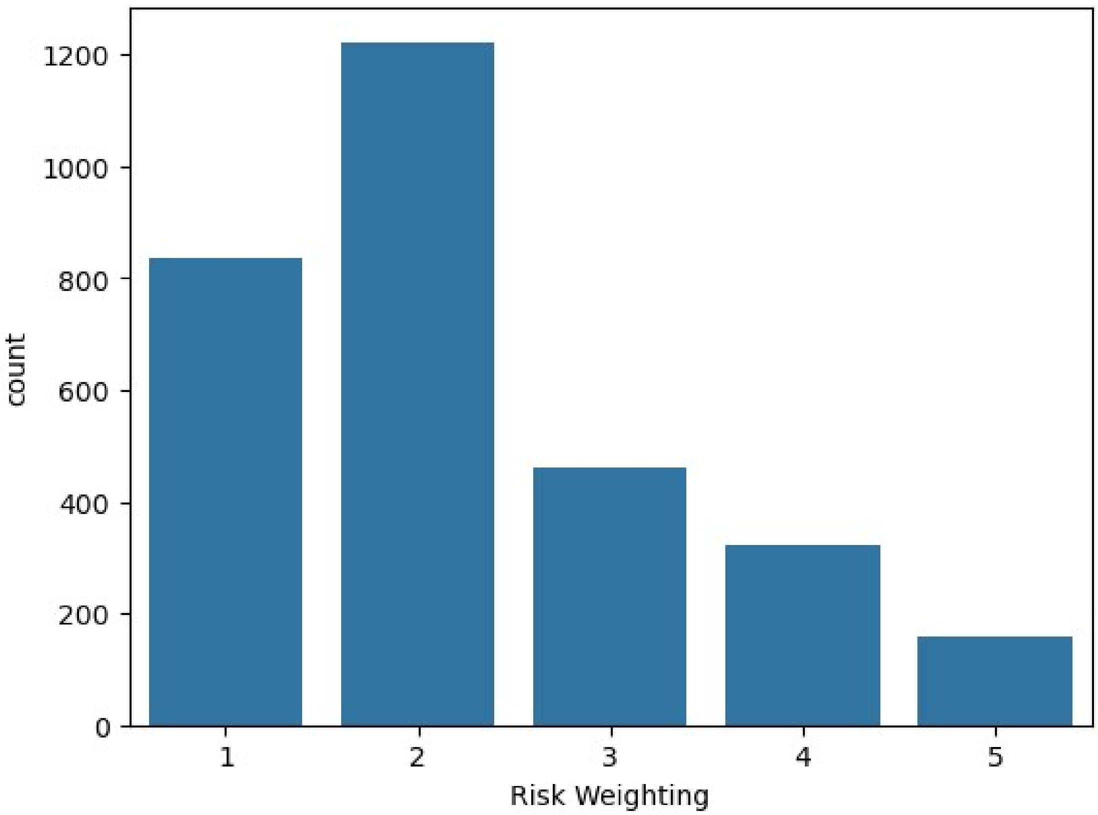

count
0
200
400
600
800
1000
1200
American
African
European
Asian
Australian
Nationality

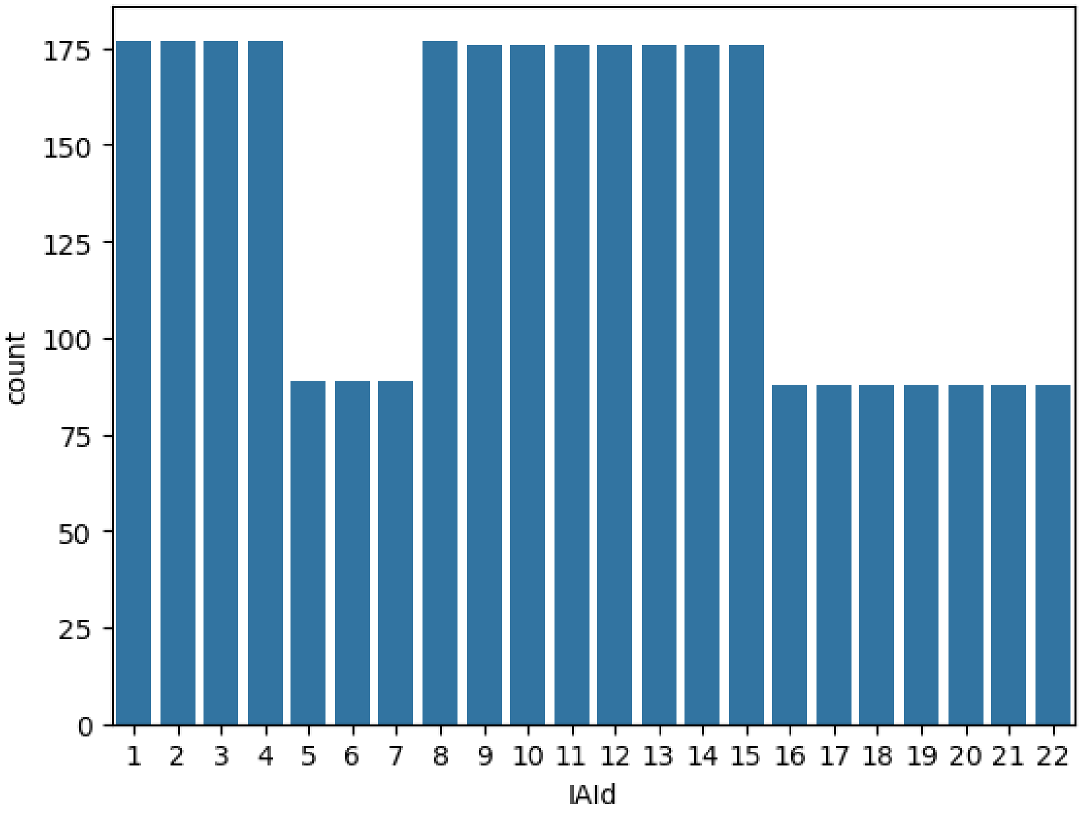

count
0
5
10
15
20
25
Occupation

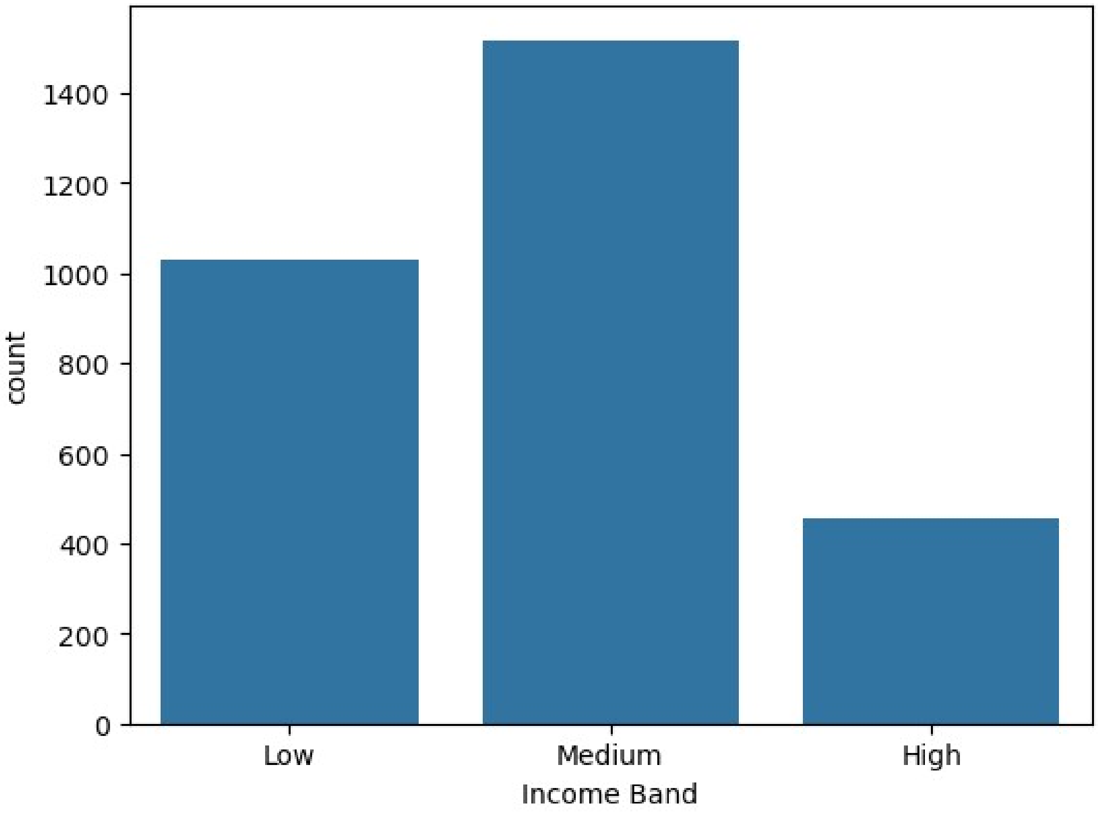

1400
1200
1000
800
600
400
200
0
count
High
Mid
Low
Fee Structure


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

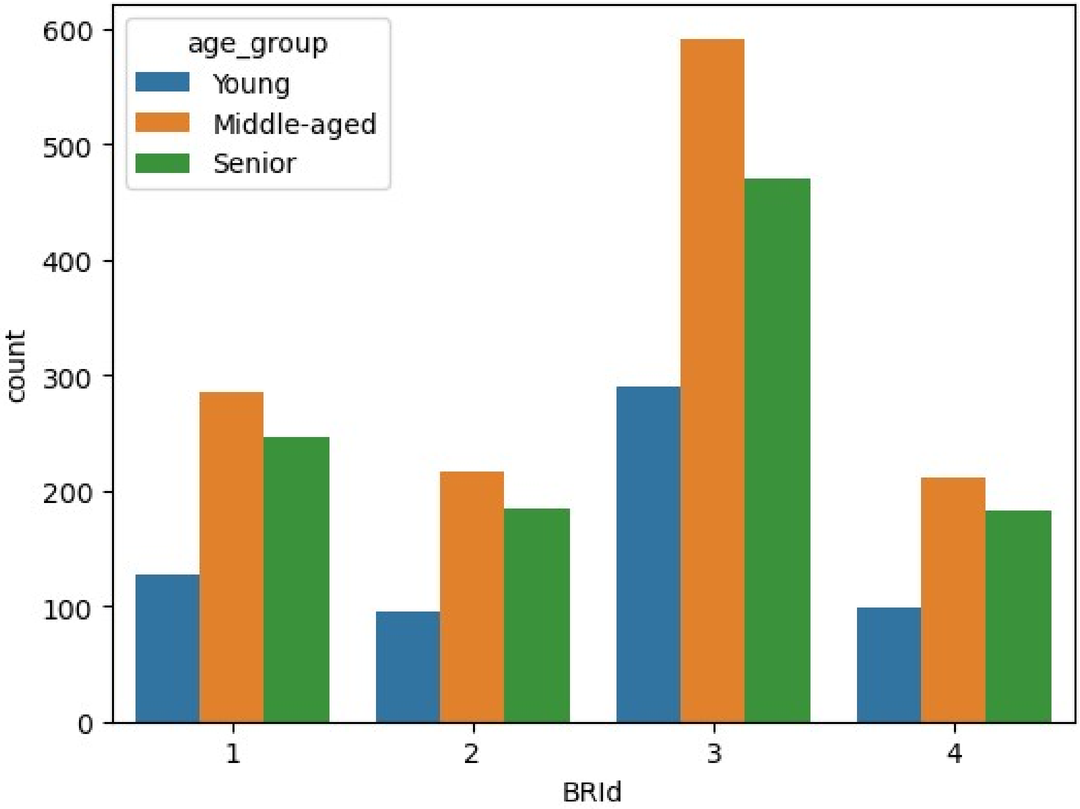

count
0
100
200
300
400
500
600
700
800
0
1
2
3
Properties Owned

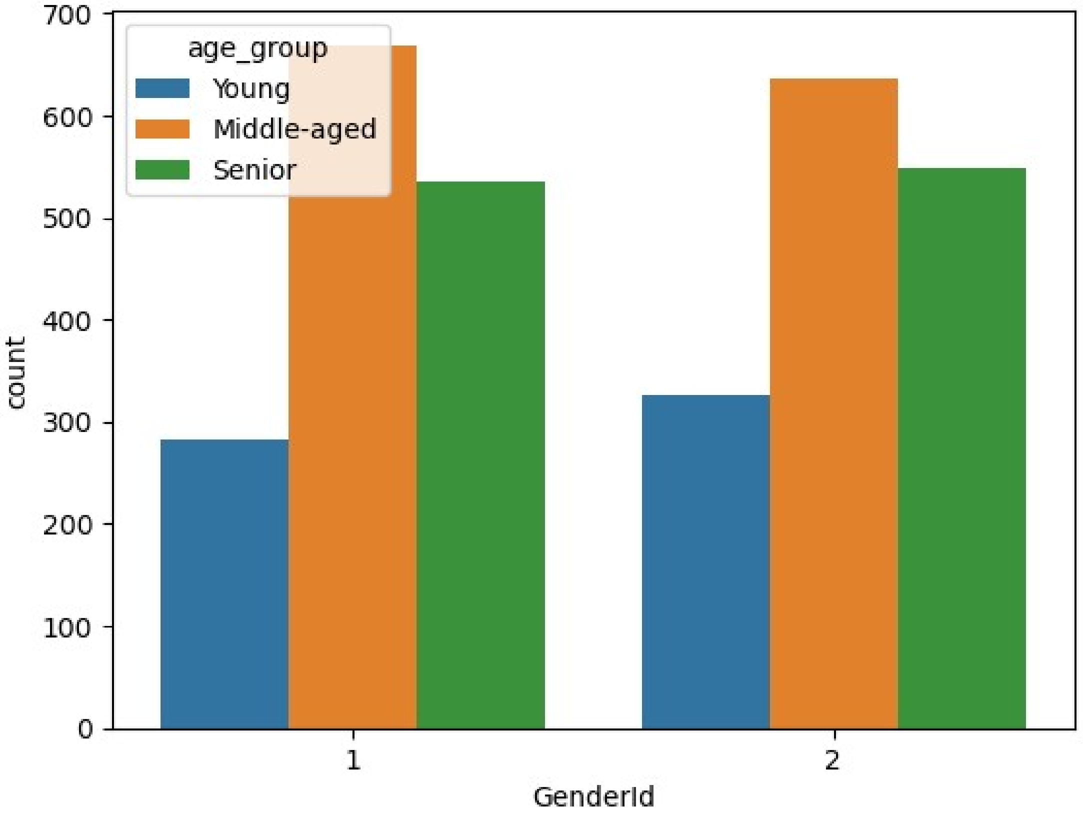

count
0
200
400
600
800
1000
1200
1
2
3
4
5
Risk Weighting

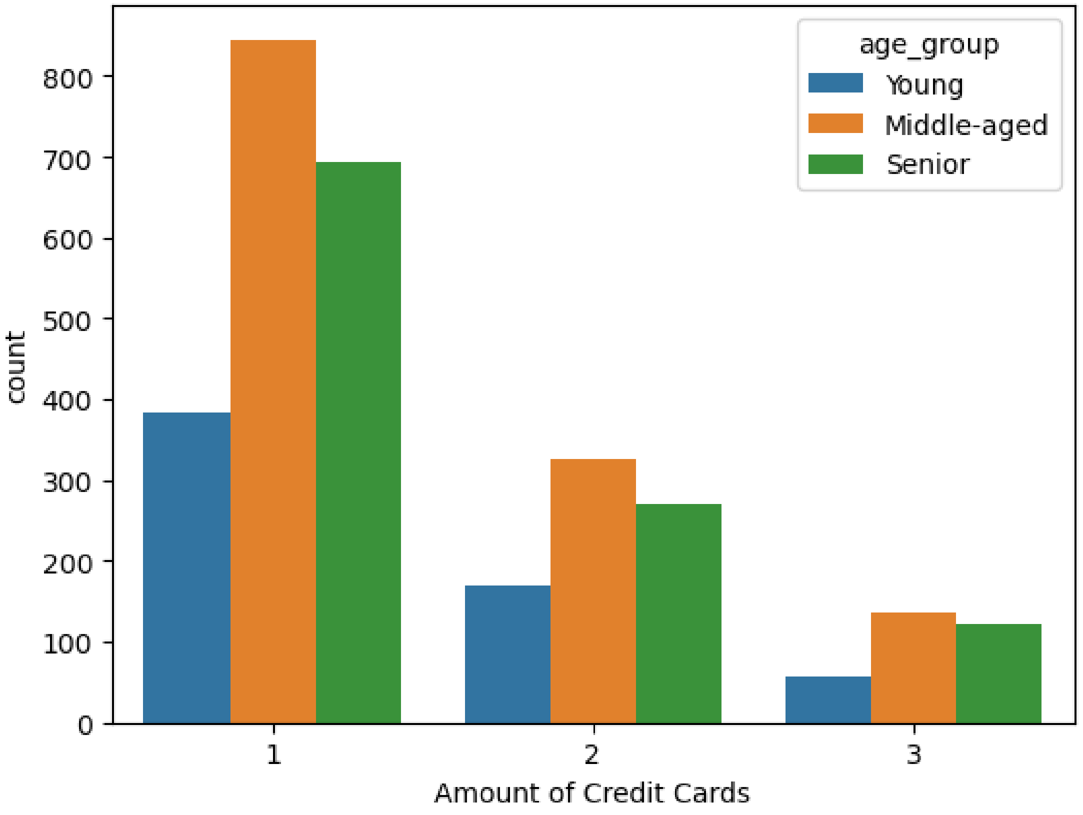

175
150
125
100
75
50
25
0
count
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
IAId

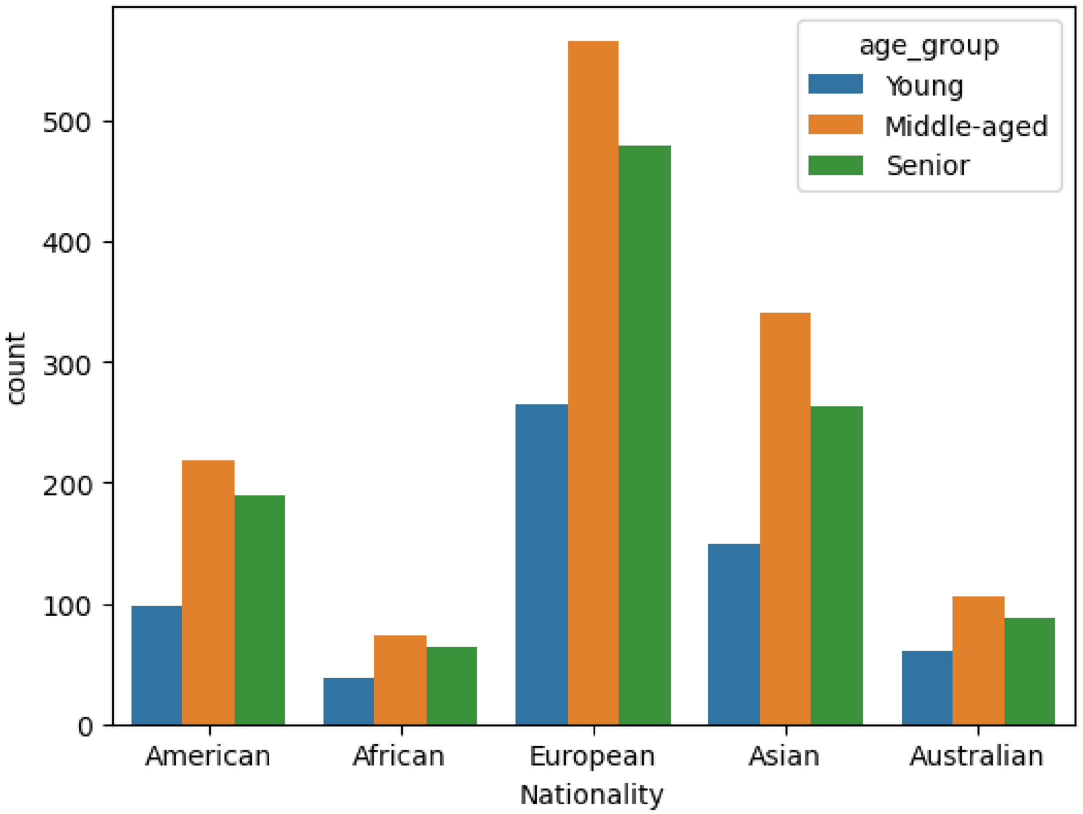

1400
1200
1000
800
600
400
200
0
count
Low
Medium
High
Income Band

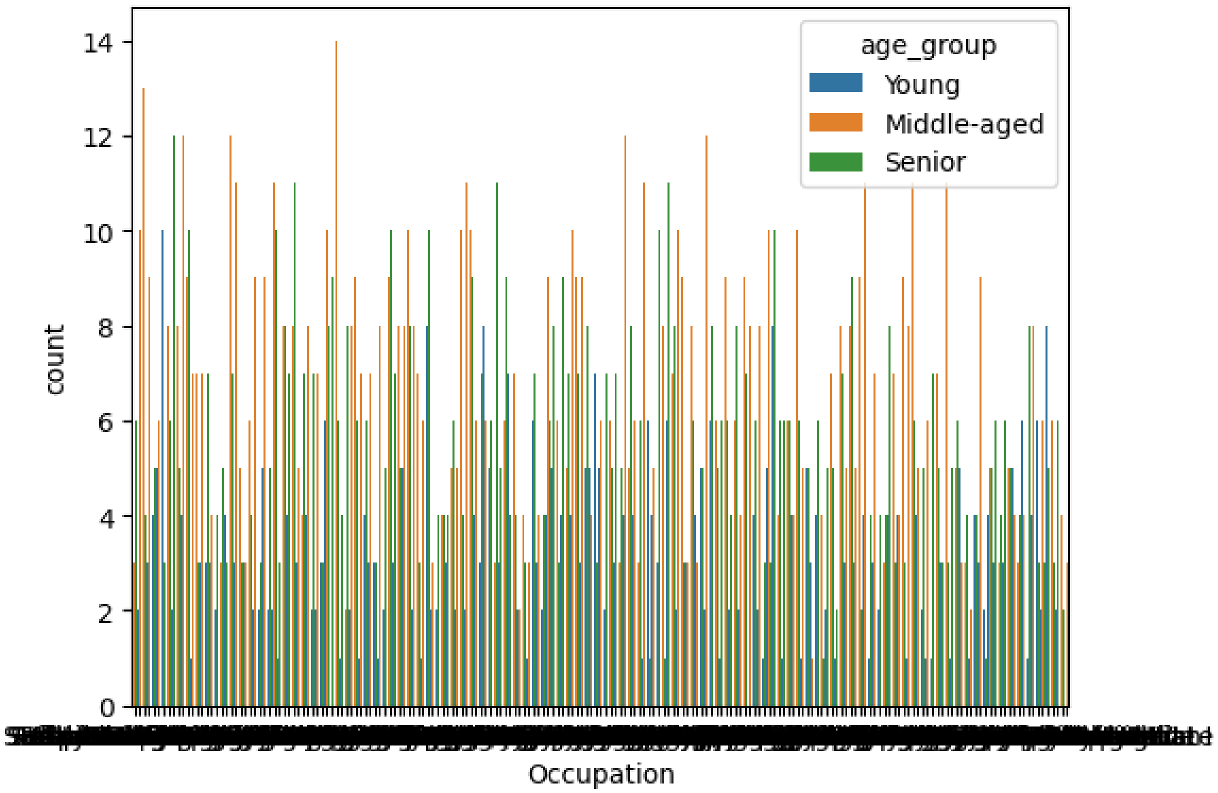
```python
#Bivariate Analysis

for i, predictor in enumerate(df[["BRId", "GenderId", "Amount of
Credit Cards", "Nationality", "Occupation", "Fee Structure", "Loyalty
Classification", "Properties Owned", "Risk Weighting", "IAId", "Income
Band"]].columns
):
    plt.figure(i)
    sns.countplot(data = df, x = predictor, hue = 'age_group')
```

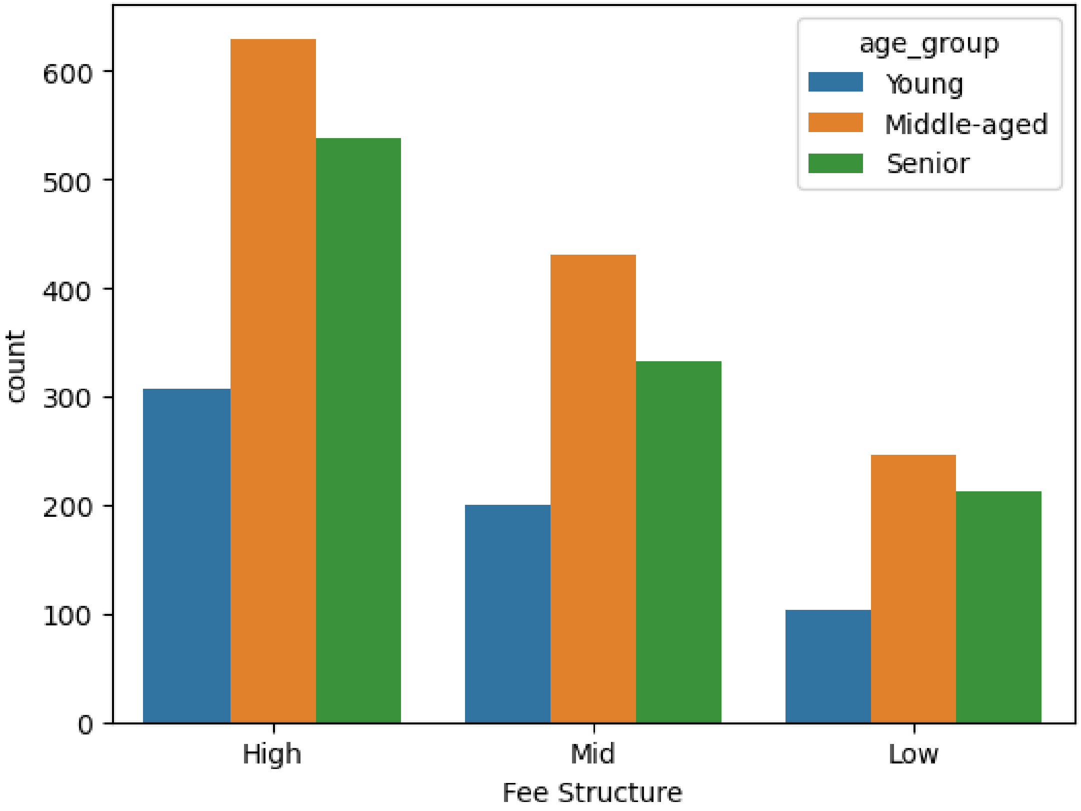


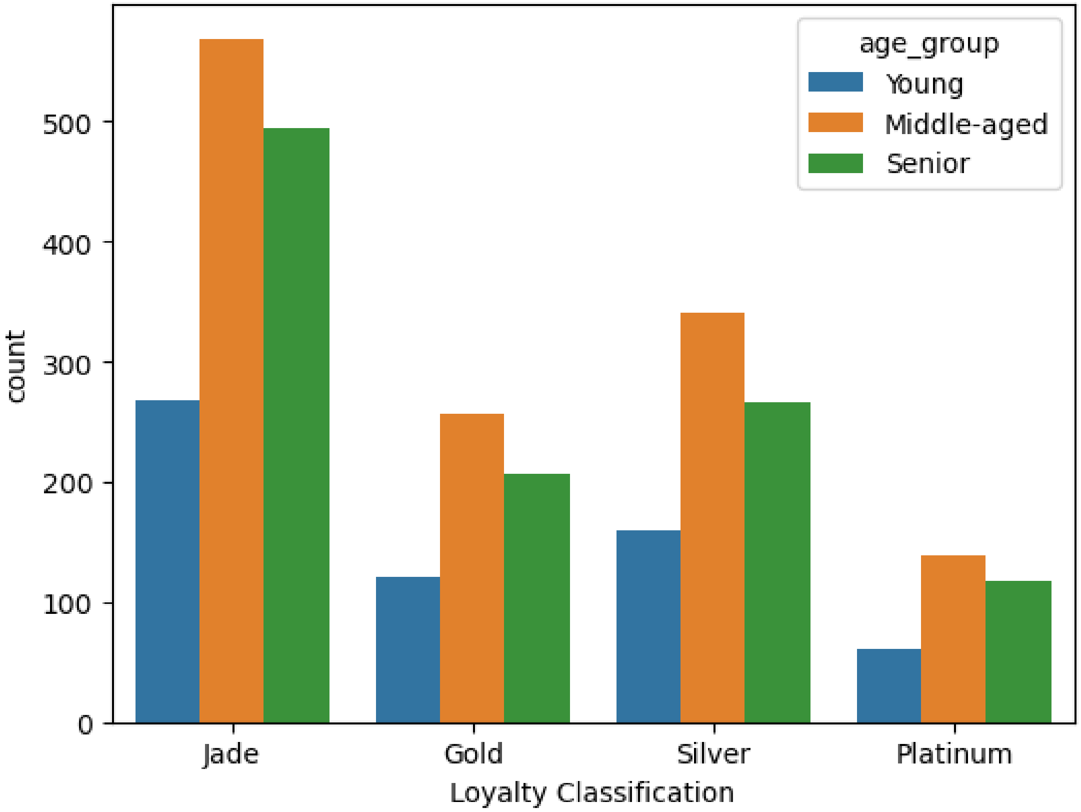

age_group
Young
Middle-aged
Senior
count
0
100
200
300
400
500
600
700
1
2
GenderId

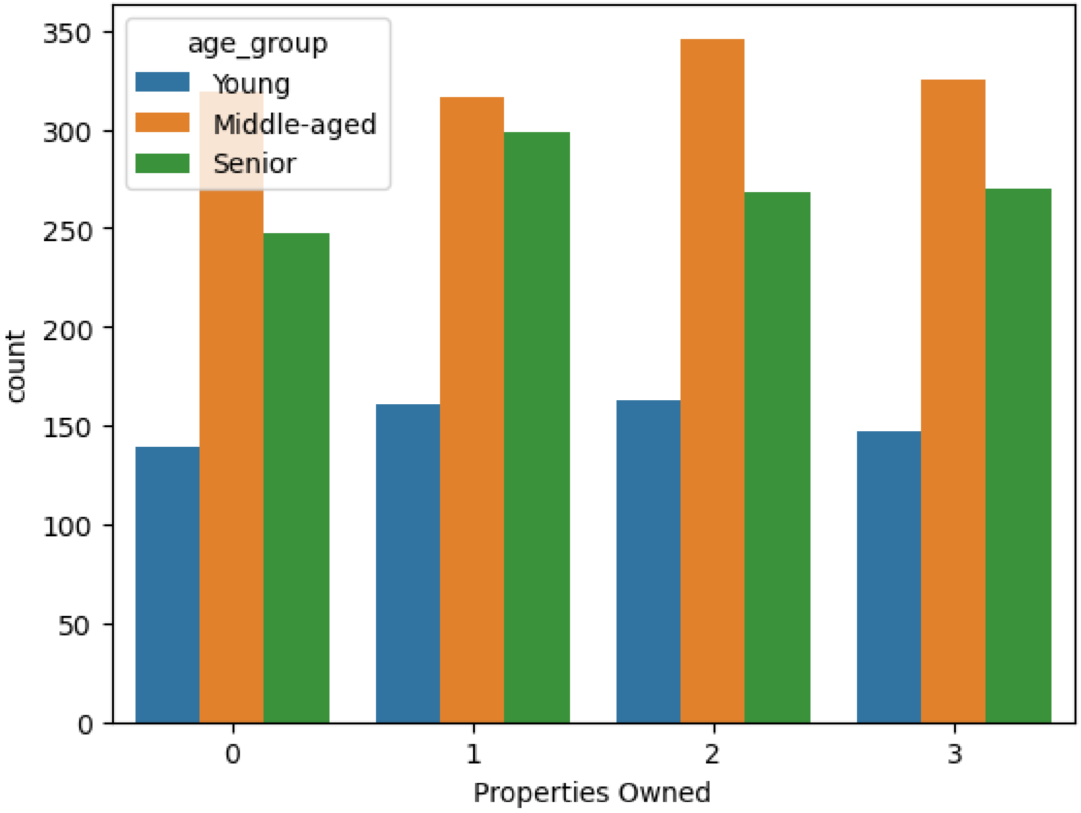

age_group
Young
Middle-aged
Senior
count
0
100
200
300
400
500
600
700
800
1
2
3
Amount of Credit Cards

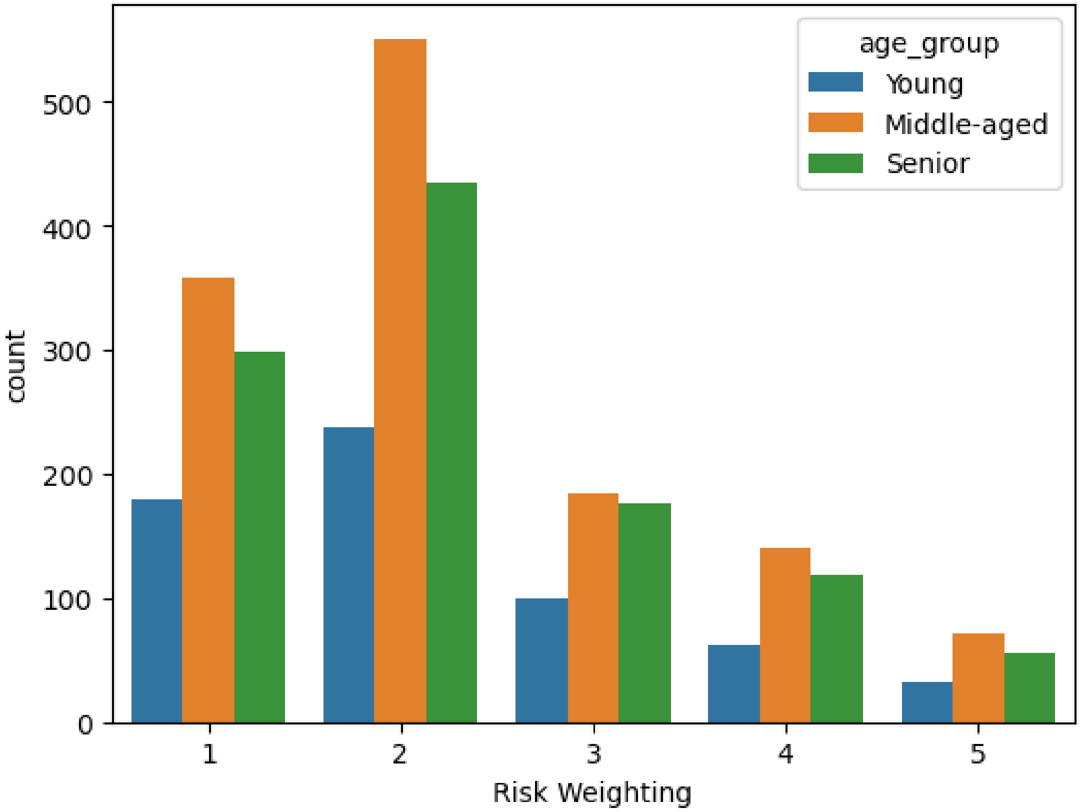

age_group
Young
Middle-aged
Senior
count
0
100
200
300
400
500
American
African
European
Asian
Australian
Nationality

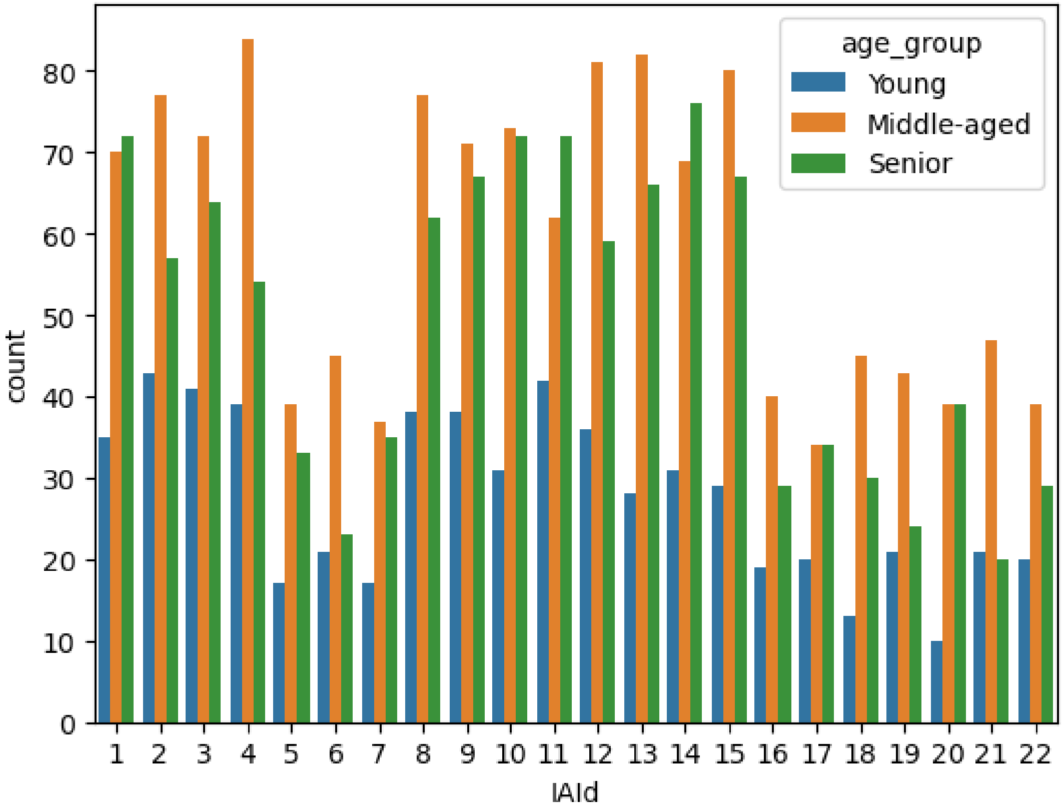

age_group
Young
Middle-aged
Senior
count
0
2
4
6
8
10
12
14
Occupation

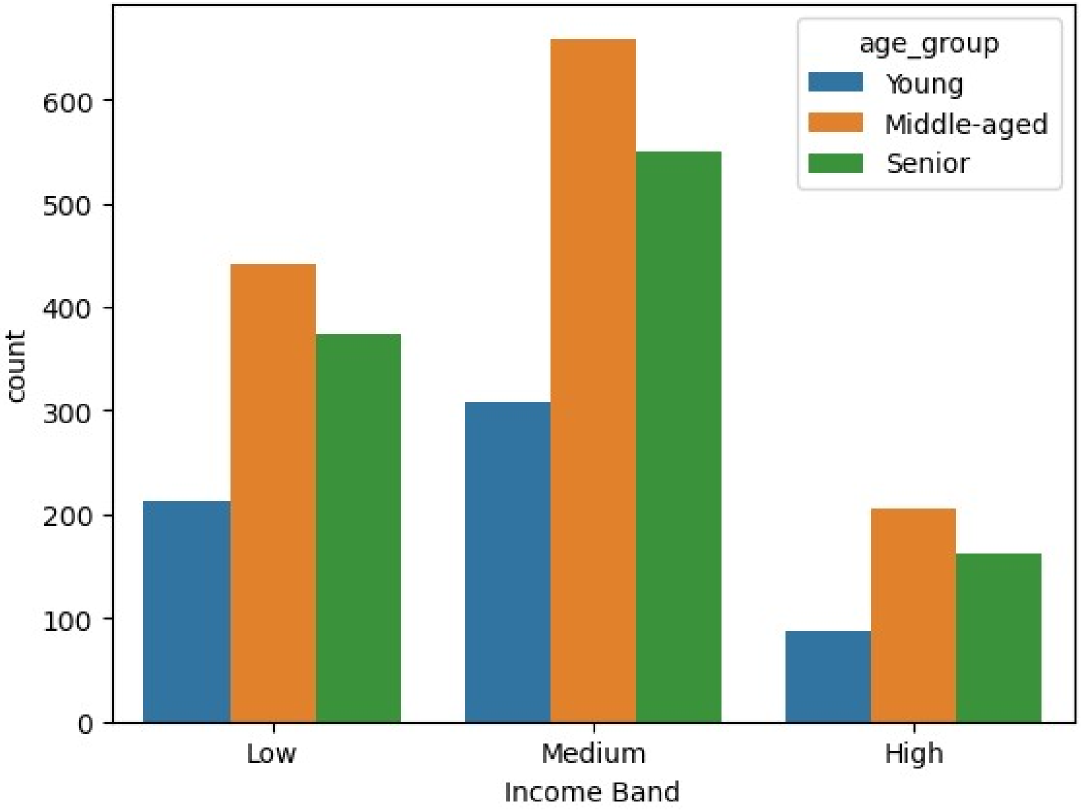

age_group
Young
Middle-aged
Senior
count
0
100
200
300
400
500
600
High
Mid
Low
Fee Structure


age_group
Young
Middle-aged
Senior
count
0
100
200
300
400
500
Jade
Gold
Silver
Platinum
Loyalty Classification

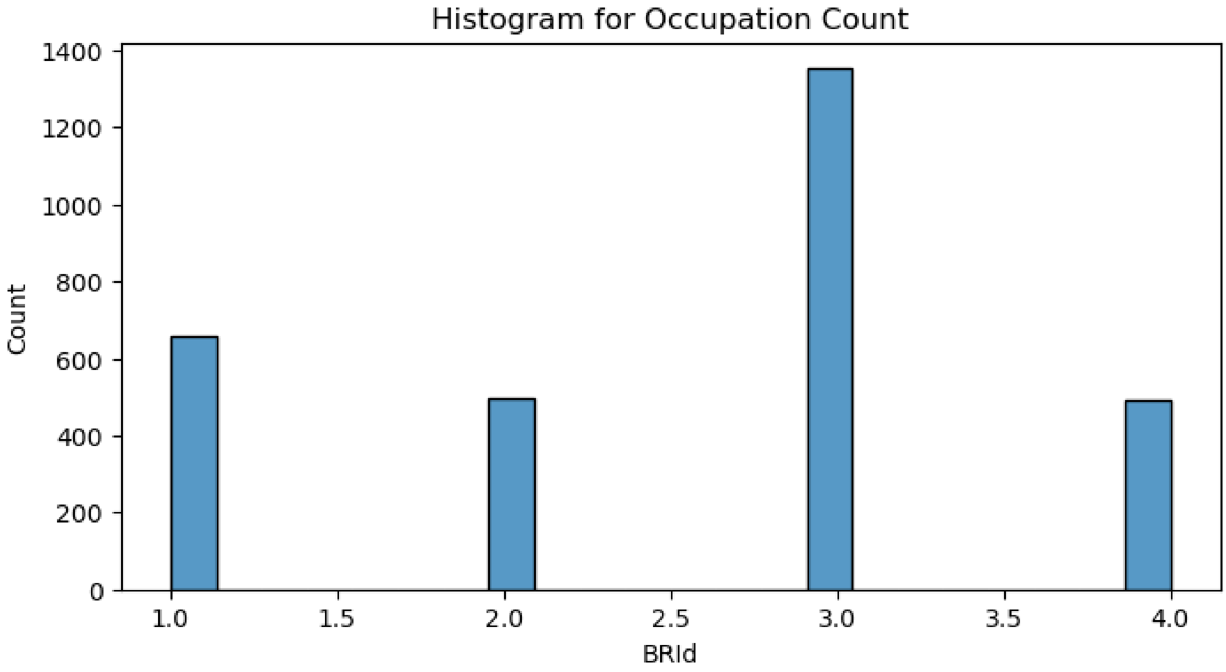

age_group
Young
Middle-aged
Senior
count
350
300
250
200
150
100
50
0
0
1
2
3
Properties Owned

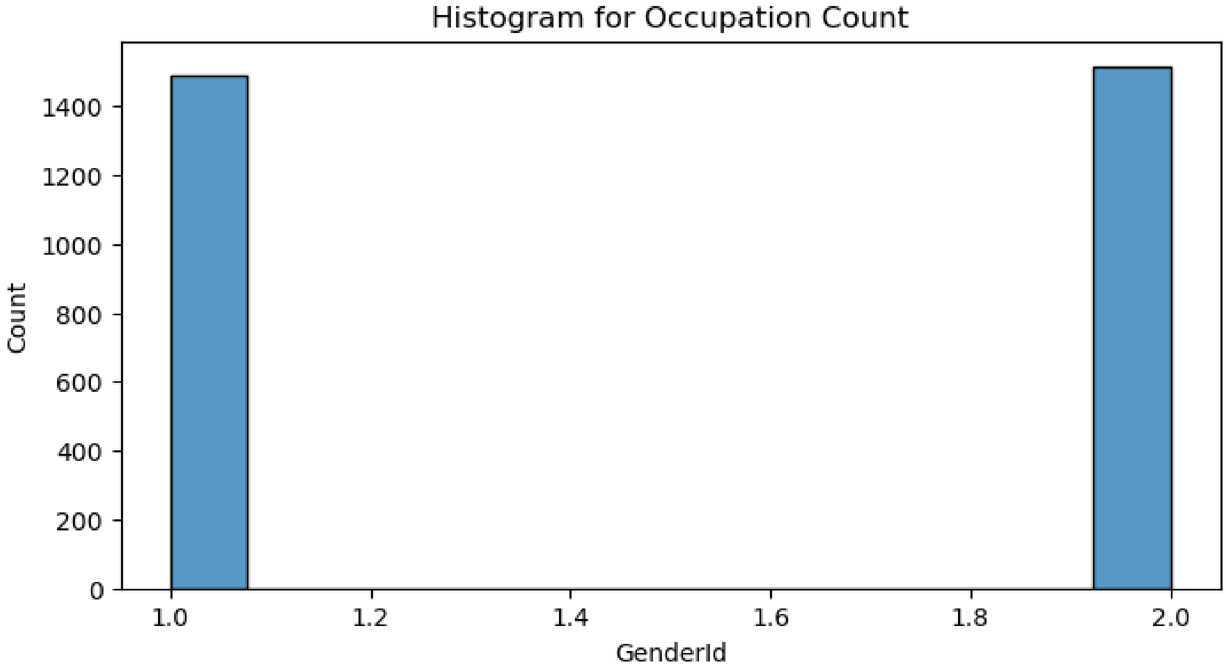

age_group
Young
Middle-aged
Senior
count
500
400
300
200
100
0
1
2
3
4
5
Risk Weighting

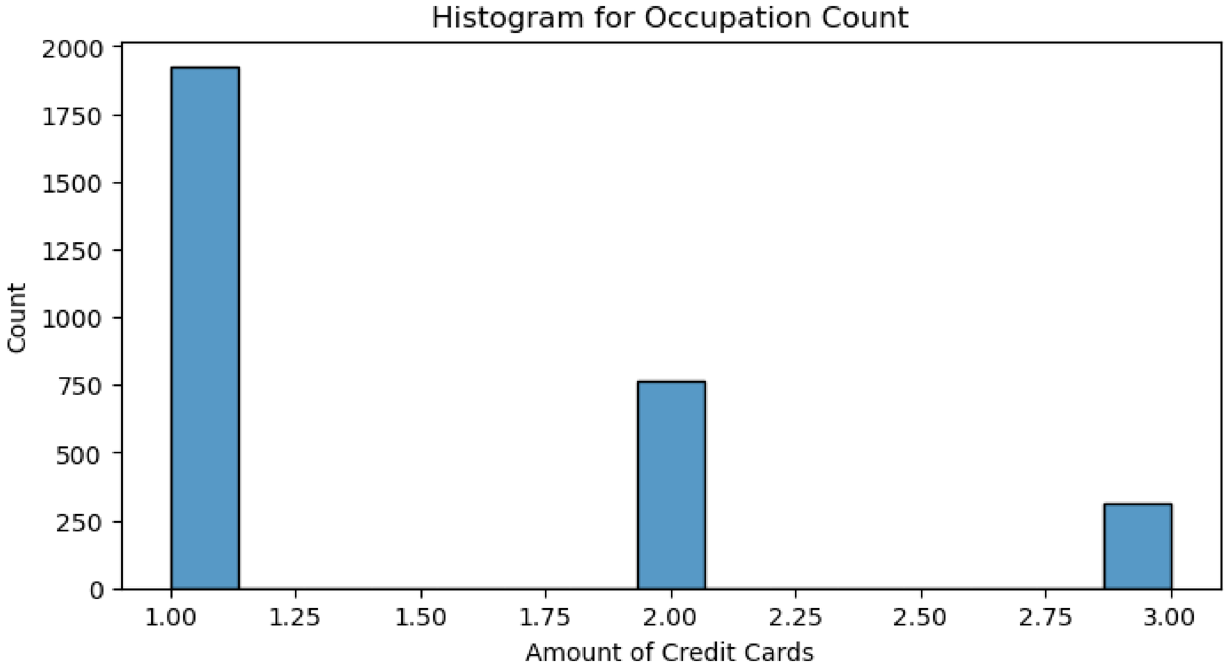

age_group
Young
Middle-aged
Senior
count
0
10
20
30
40
50
60
70
80
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
IAId

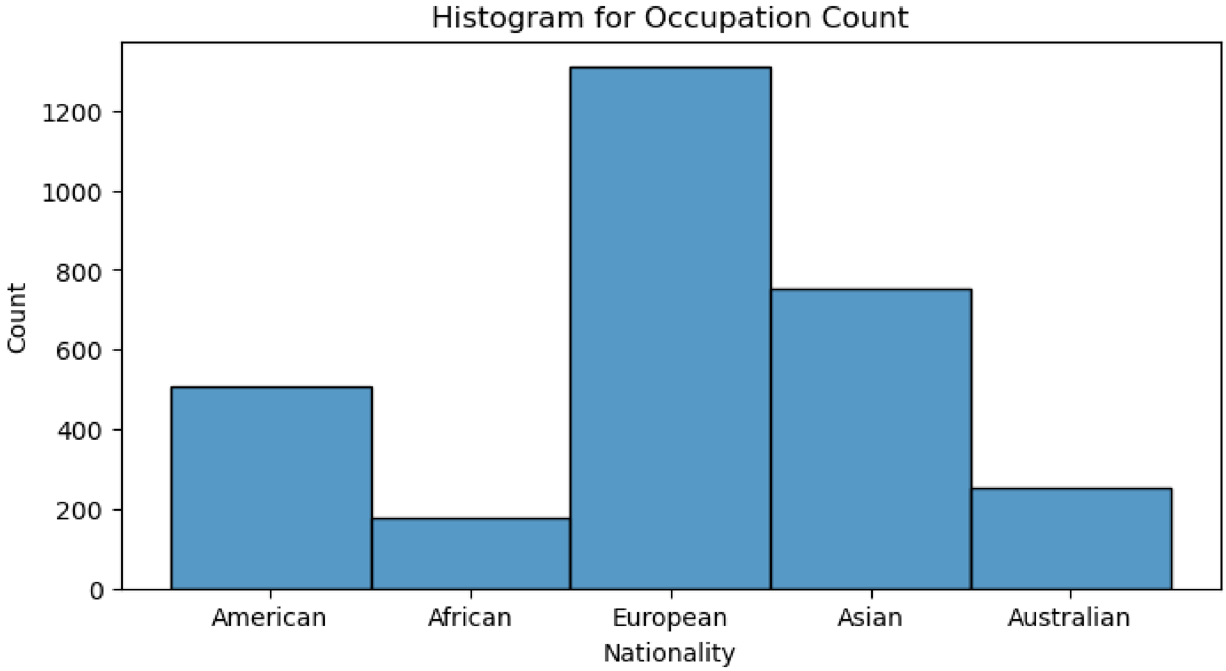

age_group
Young
Middle-aged
Senior
count
0
100
200
300
400
500
600
Low
Medium
High
Income Band

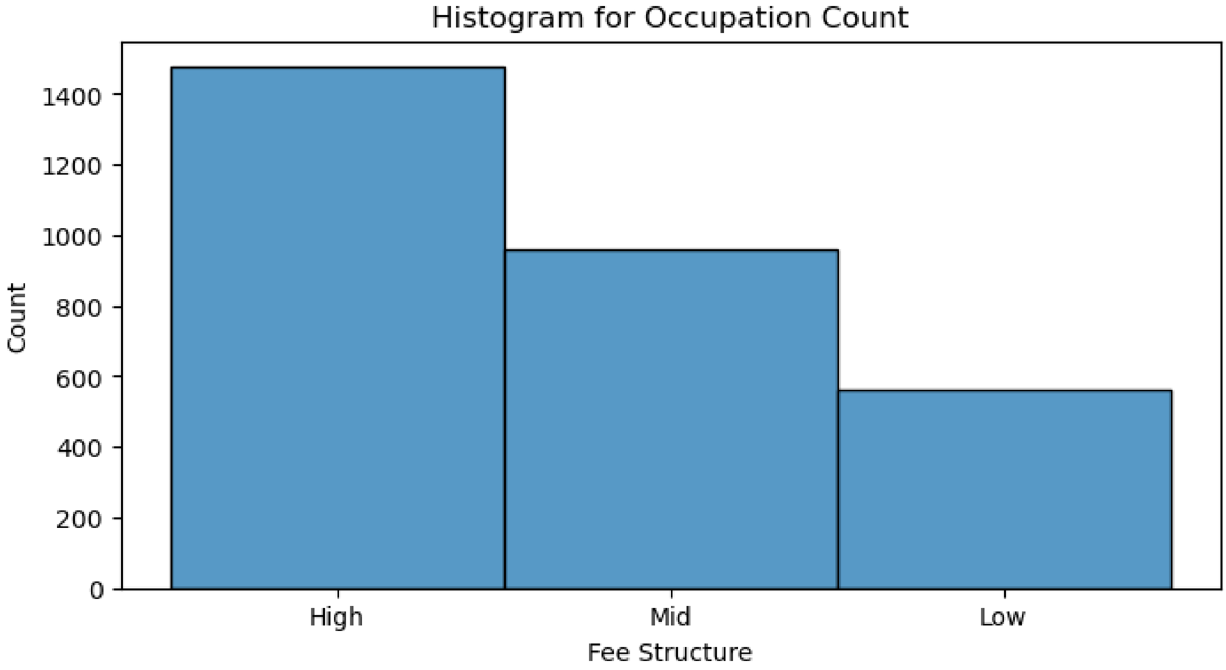
```python
#Histogram of value counts for different occupation

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

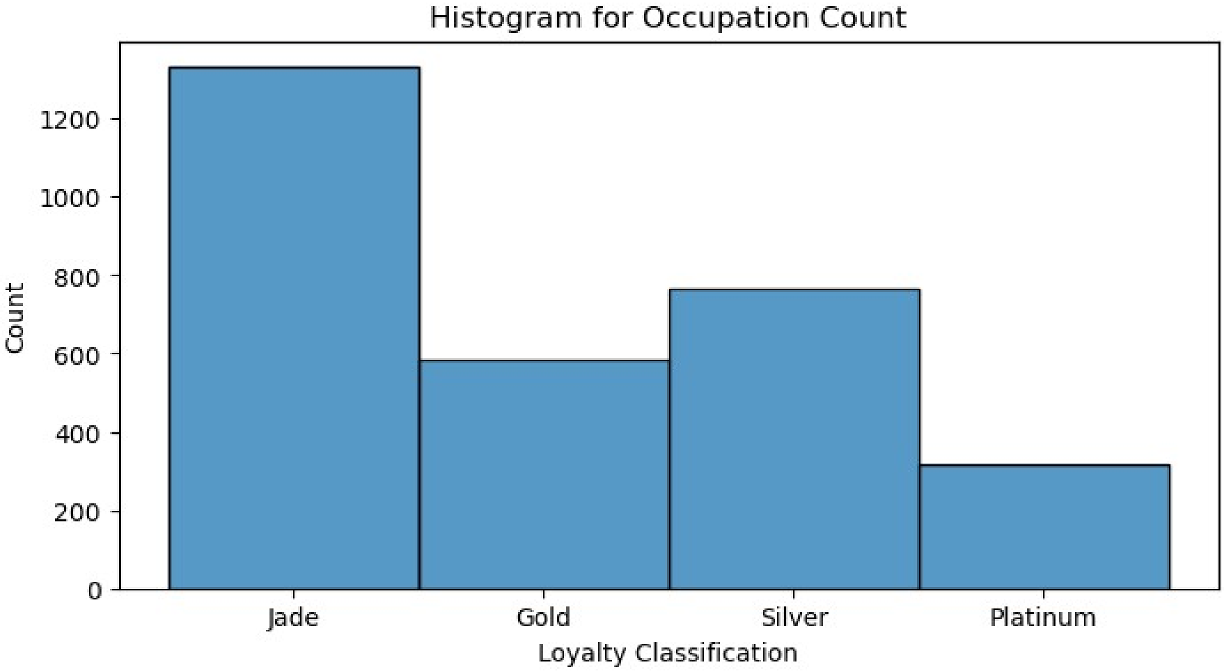


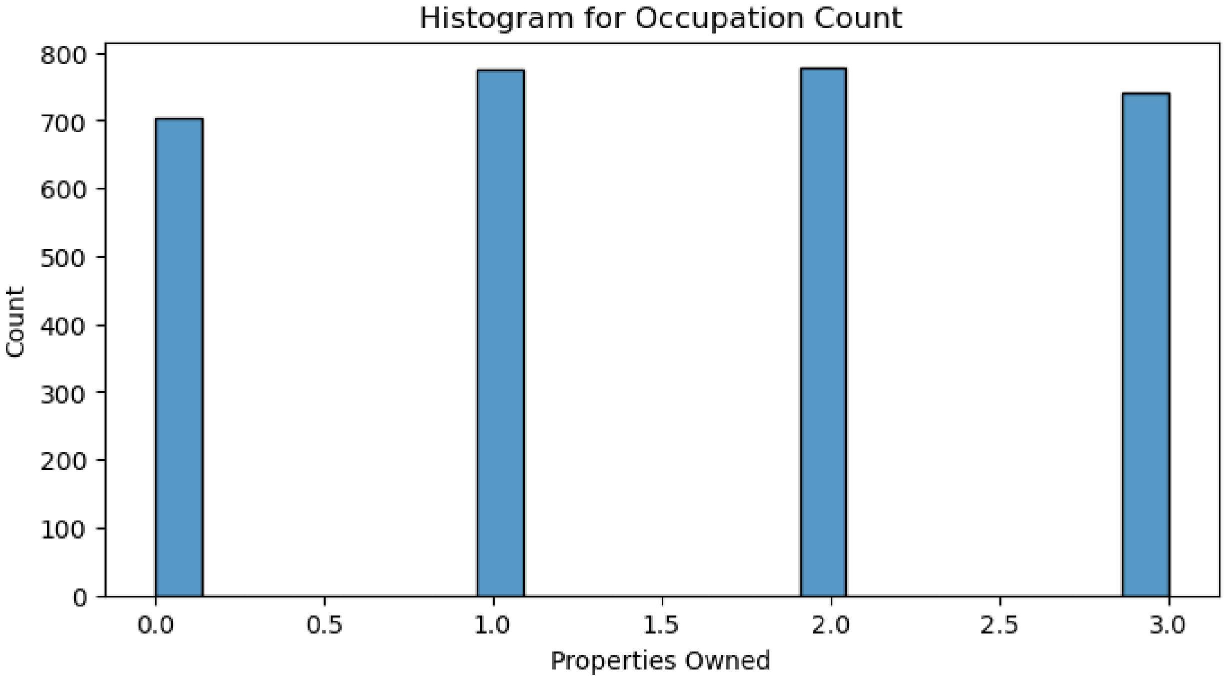

Histogram for Occupation Count
Count
0
200
400
600
800
1000
1200
1400
1.0
1.2
1.4
1.6
1.8
2.0
GenderId

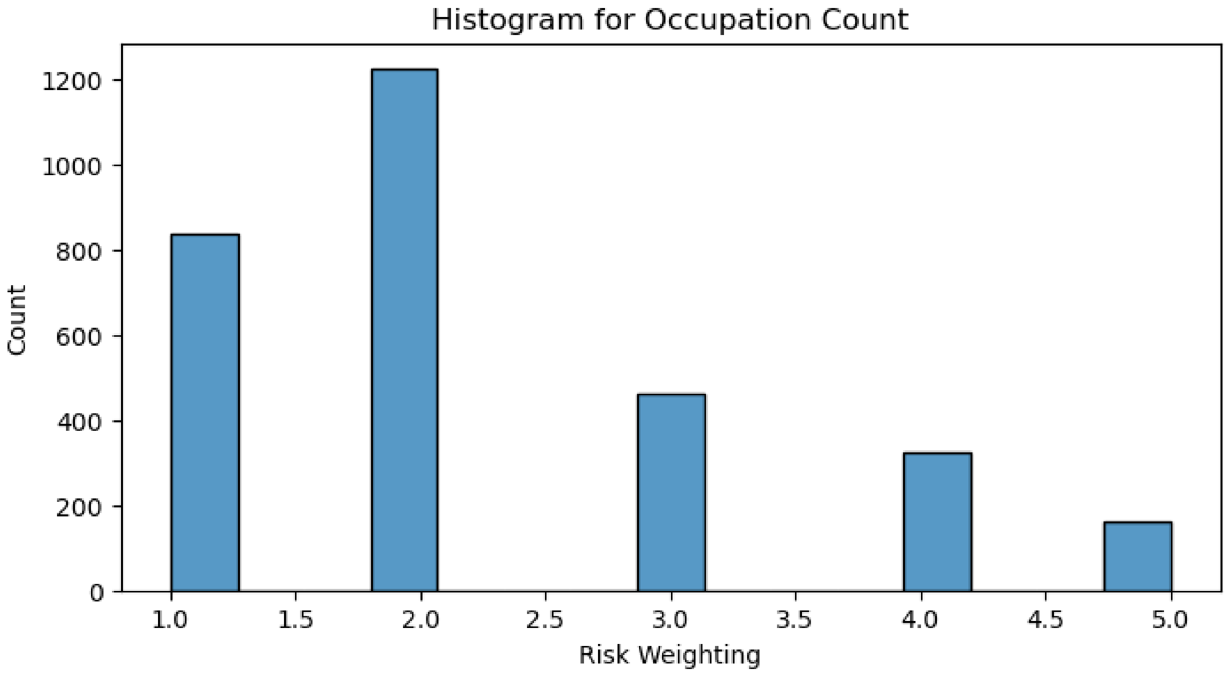

Histogram for Occupation Count
Count
0
250
500
750
1000
1250
1500
1750
2000
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Amount of Credit Cards


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

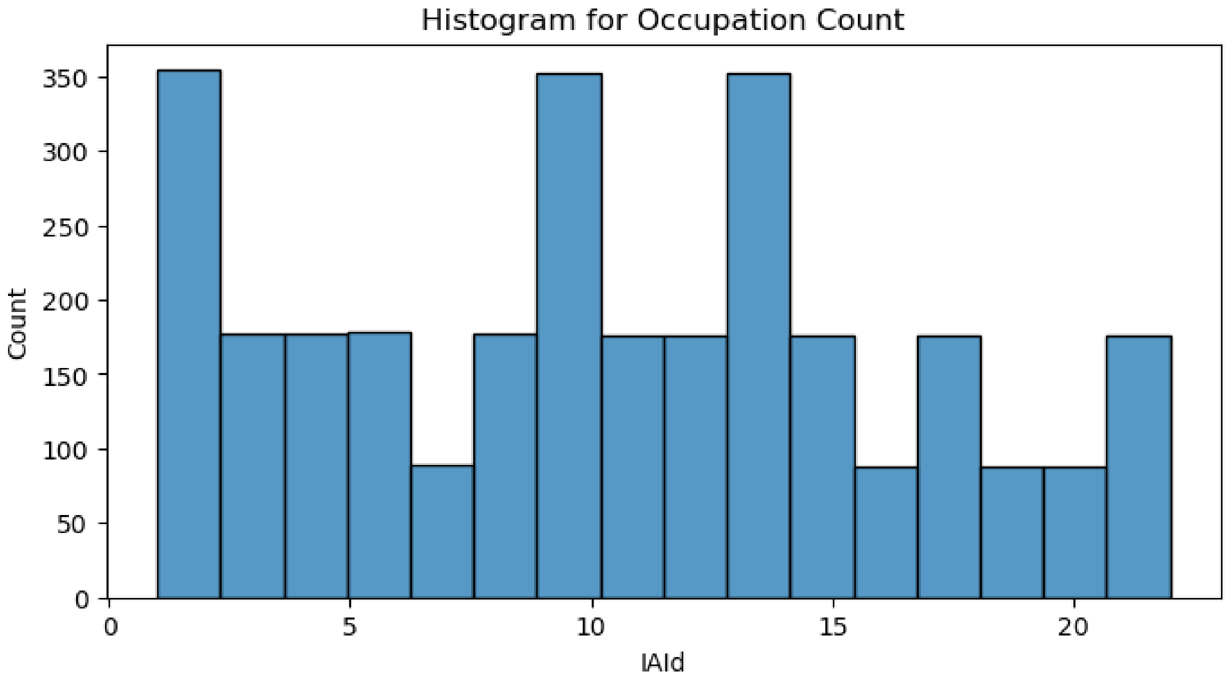

Histogram for Occupation Count
Count
0
200
400
600
800
1000
1200
1400
High
Mid
Low
Fee Structure


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

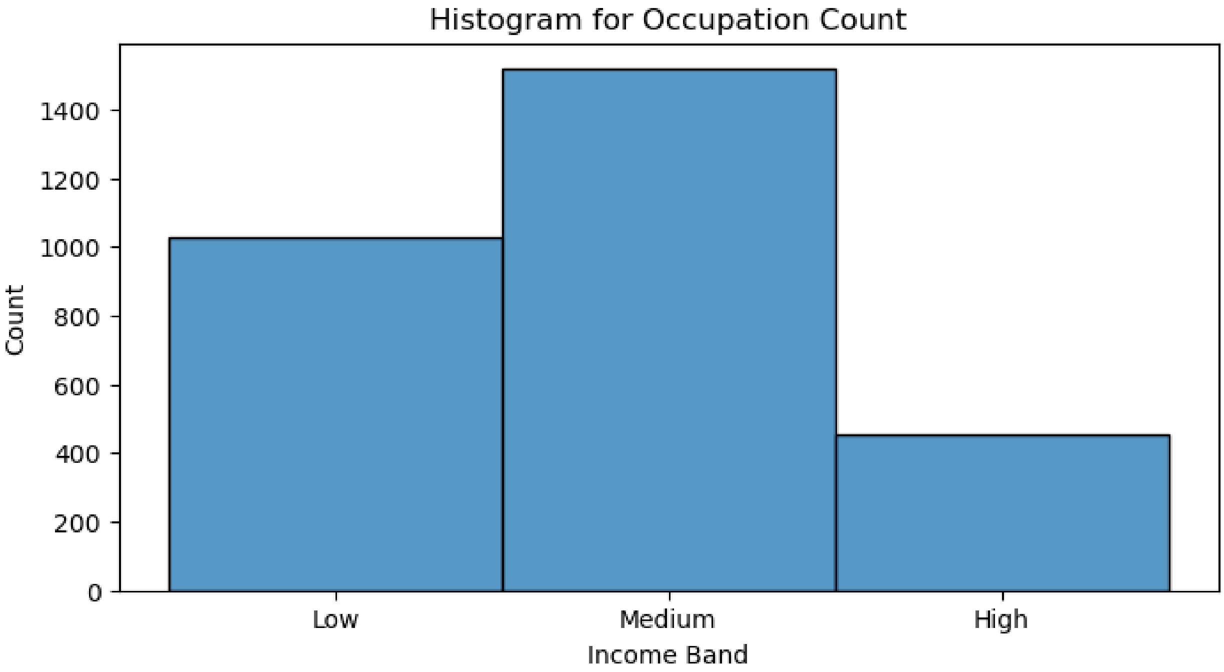

Histogram for Occupation Count
Count
800
700
600
500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Properties Owned



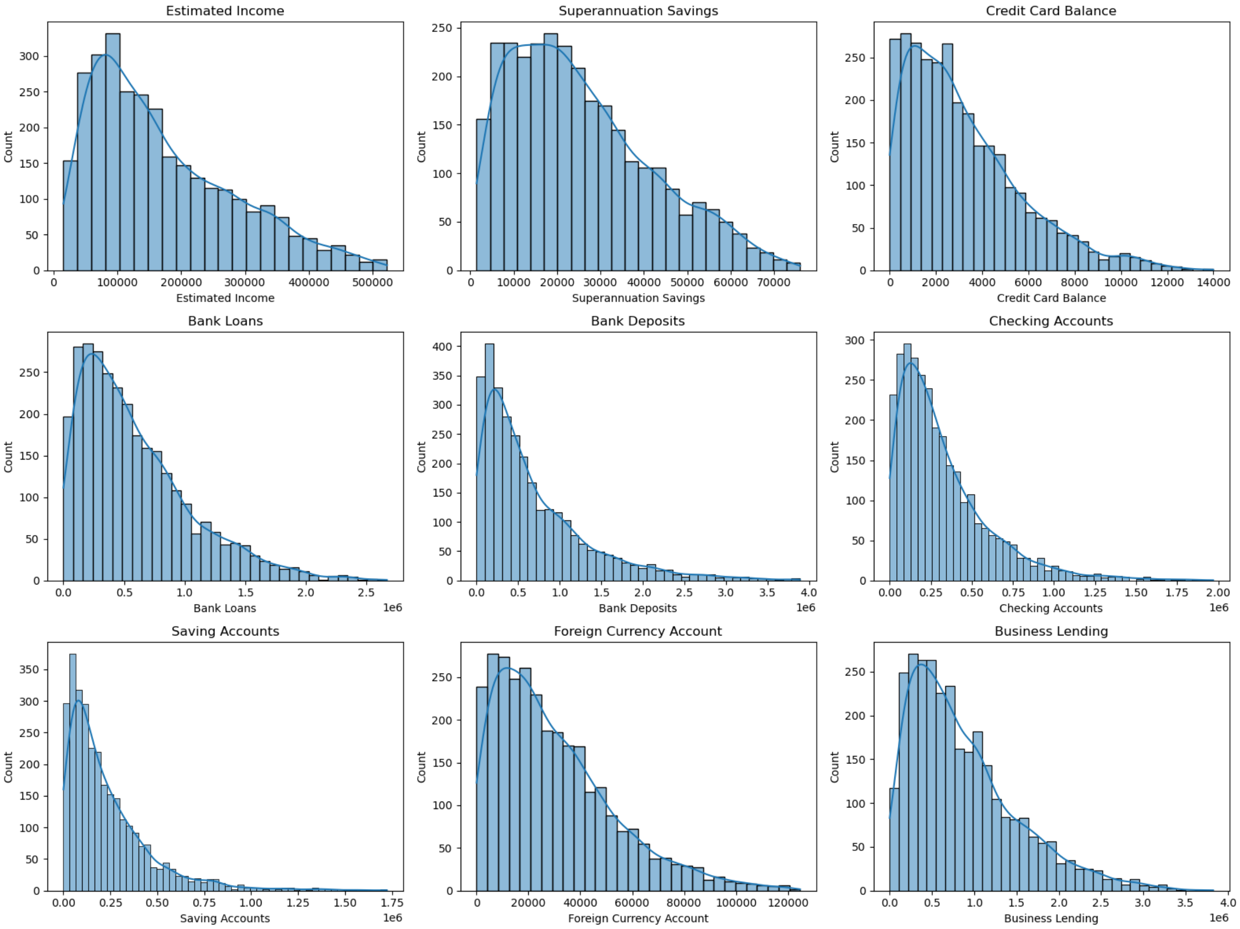
Histogram for Occupation Count
Count
0
200
400
600
800
1000
1200
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Risk Weighting



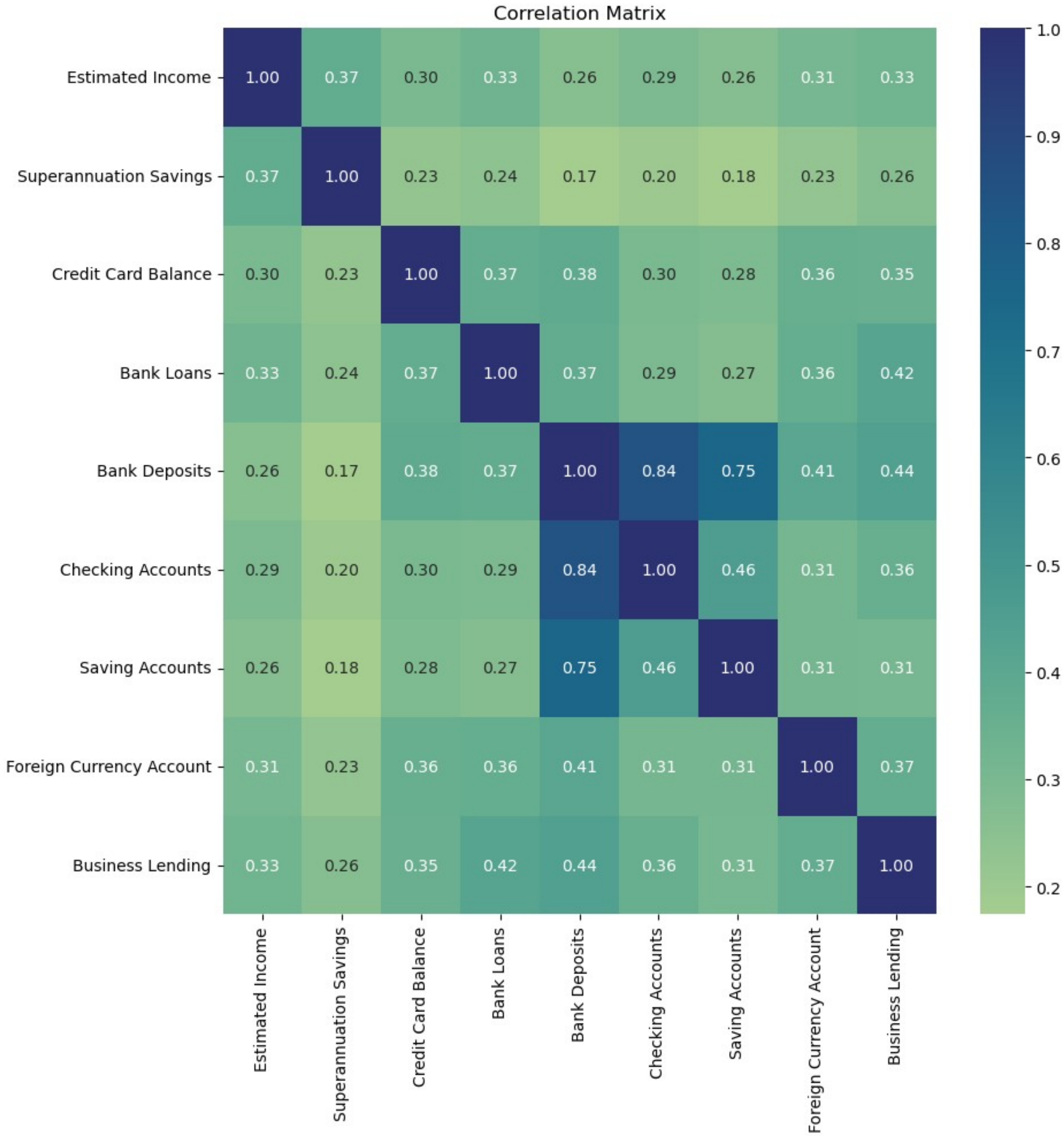
Histogram for Occupation Count
Count
0
50
100
150
200
250
300
350
0
5
10
15
20
IAId



```
#Numerical columns Univariate Analysis

numerical_cols = ['Estimated Income', 'Superannuation Savings',
'Credit Card Balance', 'Bank Loans', 'Bank Deposits', 'Checking
Accounts', 'Saving Accounts', 'Foreign Currency Account', 'Business
Lending']

plt.figure(figsize= (16,12))
for i, col in enumerate(numerical_cols):
    plt.subplot(3,3,i+1)
    sns.histplot(df[col], kde=True)
    plt.title(col)
    plt.tight_layout()
plt.show()
```



```
# Heatmaps for viewing correlation between the numerical columns

correlation_matrix = df[numerical_cols].corr()

plt.figure(figsize = (10,10))
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
1.00 0.37 0.30 0.33 0.26 0.29 0.26 0.31 0.33
0.37 1.00 0.23 0.24 0.17 0.20 0.18 0.23 0.26
0.30 0.23 1.00 0.37 0.38 0.30 0.28 0.36 0.35
0.33 0.24 0.37 1.00 0.37 0.29 0.27 0.36 0.42
0.26 0.17 0.38 0.37 1.00 0.84 0.75 0.41 0.44
0.29 0.20 0.30 0.29 0.84 1.00 0.46 0.31 0.36
0.26 0.18 0.28 0.27 0.75 0.46 1.00 0.31 0.31
0.31 0.23 0.36 0.36 0.41 0.31 0.31 1.00 0.37
0.33 0.26 0.35 0.42 0.44 0.36 0.31 0.37 1.00
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2