

BankEDA (Version 1).ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all Copy to Drive

1

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

2

df = pd.read_csv('/dherya/Banking.csv')
df.head(5)

| | Client ID | Name | Age | Location ID | Joined Bank | Banking Contact | Nationality | Occupation | Fee Structure | Loyalty Classification | ... | Bank Deposits | Checking Accounts | Saving Accounts | Foreign Currency Account | Business Lending | Properties Owned | Risk Weighting | BRId | GenderId | IAId |
|---|-----------|-----------------|-----|-------------|-------------|------------------|-------------|----------------------|---------------|------------------------|-----|---------------|-------------------|-----------------|--------------------------|------------------|------------------|----------------|------|----------|------|
| 0 | IND01288 | Raymond Mills | 24 | 34324 | 06-05-2019 | Anthony Torres | American | Safety Technician IV | High | Jade | ... | 1485828.64 | 603617.88 | 607332.46 | 12249.96 | 1134475.30 | 1 | 2 | 1 | 1 | 1 |
| 1 | IND65833 | Julia Spencer | 23 | 42205 | 10-12-2001 | Jonathan Hawkins | African | Software Consultant | High | Jade | ... | 641482.79 | 229521.37 | 344635.16 | 61162.31 | 2000526.10 | 1 | 3 | 2 | 1 | 2 |
| 2 | IND47499 | Stephen Murray | 27 | 7314 | 25-01-2010 | Anthony Barry | European | Help Desk Operator | High | Gold | ... | 1033401.59 | 652674.69 | 203054.35 | 79071.78 | 548137.58 | 1 | 3 | 3 | 2 | 3 |
| 3 | IND72498 | Virginia Garza | 40 | 34594 | 28-03-2019 | Steve Diaz | American | Geologist II | Mid | Silver | ... | 1048157.49 | 1048157.49 | 234685.02 | 57513.05 | 1148402.29 | 0 | 4 | 4 | 1 | 4 |
| 4 | IND60181 | Melissa Sanders | 46 | 41269 | 20-07-2012 | Shawn Long | American | Assistant Professor | Mid | Platinum | ... | 487782.53 | 446644.25 | 128351.45 | 30012.14 | 1674412.12 | 0 | 3 | 1 | 2 | 5 |

5 rows × 25 columns

df.shape

(3000, 25)

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

```
# Generate descriptive statistics for the dataframe
df.describe()
```

| | Age | Location ID | Estimated Income | Superannuation Savings | Amount of Credit Cards | Credit Card Balance | Bank Loans | Bank Deposits | Checking Accounts | Saving Accounts | Foreign Currency Account | Business Lending | Properties Owned | Risk Weighting | BRID | GenderId | JAID |
|-------|-------------|--------------|------------------|------------------------|------------------------|---------------------|--------------|---------------|-------------------|-----------------|--------------------------|------------------|------------------|----------------|-------------|-------------|-------------|
| count | 3000.000000 | 3000.000000 | 3000.000000 | 3000.000000 | 3000.000000 | 3000.000000 | 3.000000e+03 | 3.000000e+03 | 3.000000e+03 | 3.000000e+03 | 3000.000000 | 3.000000e+03 | 3000.000000 | 3000.000000 | 3000.000000 | 3000.000000 | 3000.000000 |
| mean | 51.039667 | 21563.323000 | 171305.034263 | 25531.599673 | 1.463667 | 3176.206943 | 5.913862e+05 | 6.715602e+05 | 3.210929e+05 | 2.329084e+05 | 29803.529993 | 8.667598e+05 | 1.518667 | 2.249333 | 2.559333 | 1.504000 | 10.425333 |
| std | 19.054760 | 12462.273017 | 111935.808209 | 16259.950770 | 0.676387 | 2497.094709 | 4.579570e+05 | 6.457169e+05 | 2.820796e+05 | 2.300078e+05 | 23109.924010 | 6.412303e+05 | 1.102145 | 1.131191 | 1.007713 | 0.500067 | 5.988242 |
| min | 17.000000 | 12.000000 | 15919.480000 | 1482.030000 | 1.000000 | 1.170000 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 45.000000 | 0.000000e+00 | 0.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| 25% | 34.000000 | 10803.500000 | 82906.595000 | 12513.775000 | 1.000000 | 1236.630000 | 2.396281e+05 | 2.044004e+05 | 1.199475e+05 | 7.479440e+04 | 11916.542500 | 3.748251e+05 | 1.000000 | 1.000000 | 2.000000 | 1.000000 | 5.000000 |
| 50% | 51.000000 | 21129.500000 | 142313.480000 | 22357.355000 | 1.000000 | 2560.805000 | 4.797934e+05 | 4.633165e+05 | 2.428157e+05 | 1.640866e+05 | 24341.190000 | 7.113147e+05 | 2.000000 | 2.000000 | 3.000000 | 2.000000 | 10.000000 |
| 75% | 69.000000 | 32054.500000 | 242290.305000 | 35464.740000 | 2.000000 | 4522.632500 | 8.258130e+05 | 9.427546e+05 | 4.348740e+05 | 3.155750e+05 | 41966.392500 | 1.185110e+06 | 2.000000 | 3.000000 | 3.000000 | 2.000000 | 15.000000 |
| max | 85.000000 | 43369.000000 | 522330.260000 | 75963.900000 | 3.000000 | 13991.990000 | 2.667557e+06 | 3.890598e+06 | 1.969923e+06 | 1.724118e+06 | 124704.870000 | 3.825962e+06 | 3.000000 | 5.000000 | 4.000000 | 2.000000 | 22.000000 |

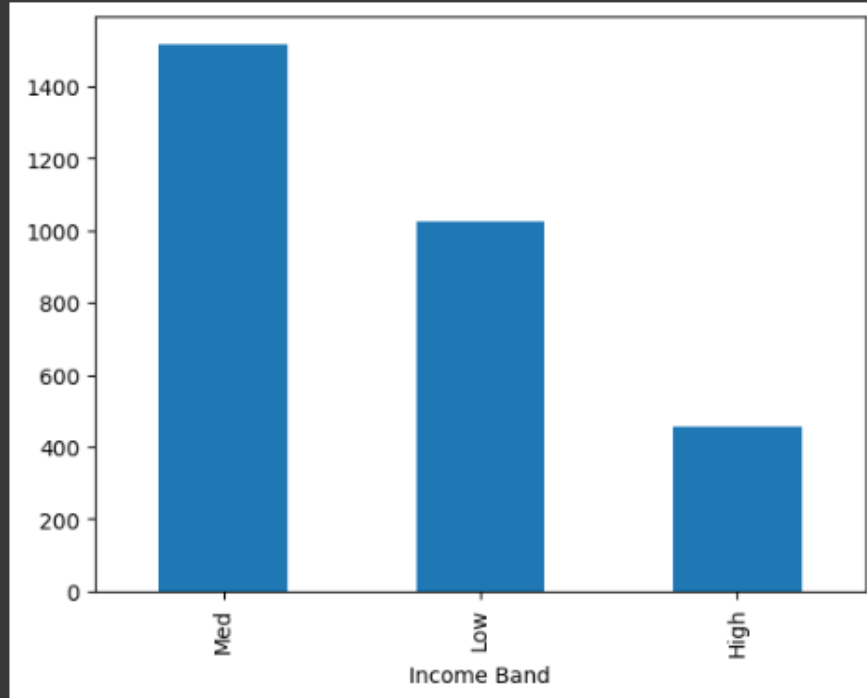
```
bins = [0, 100000, 300000, float('inf')]
labels = ['Low', 'Med', 'High']
```

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

Loading...

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

<Axes: xlabel='Income Band'>



```

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

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

```

Value Counts for 'BRId':

| | count |
|------|-------|
| BRId | |
| 3 | 1352 |
| 1 | 660 |
| 2 | 495 |
| 4 | 493 |

dtype: int64

Value Counts for 'GenderId':

| | count |
|----------|-------|
| GenderId | |
| 2 | 1512 |
| 1 | 1488 |

dtype: int64

Value Counts for 'IAId':

| | count |
|------------------------|-------|
| Amount of Credit Cards | |
| 1 | 1922 |
| 2 | 765 |
| 3 | 313 |

dtype: int64

Value Counts for 'Nationality':

| | count |
|-------------|-------|
| Nationality | |
| European | 1309 |
| Asian | 754 |
| American | 507 |
| Australian | 254 |
| African | 176 |

dtype: int64

Value Counts for 'Occupation':

```

count
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
195 rows × 1 columns

dtype: int64
Value Counts for 'Fee Structure':

```

```

count
Properties Owned
2      777
1      776
3      742
0      705

dtype: int64
Value Counts for 'Risk Weighting':

count
Risk Weighting
2      1222
1      836
3      460
4      322
5      160

dtype: int64
Value Counts for 'Income Band':

count
Income Band
Med      1517
Low      1027
High      456

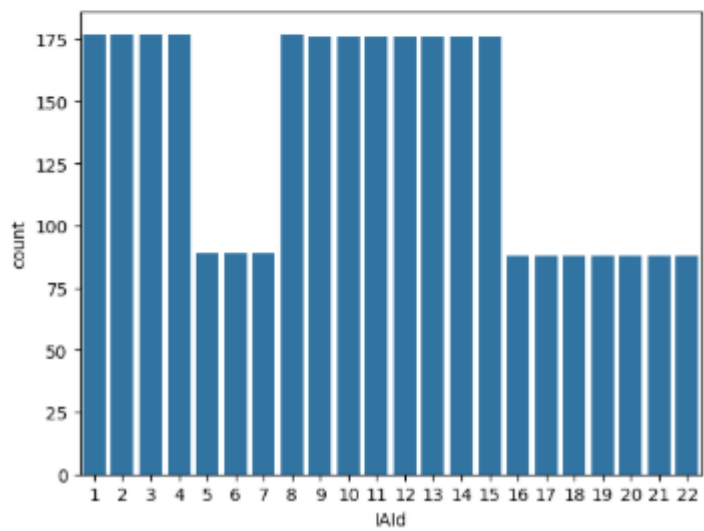
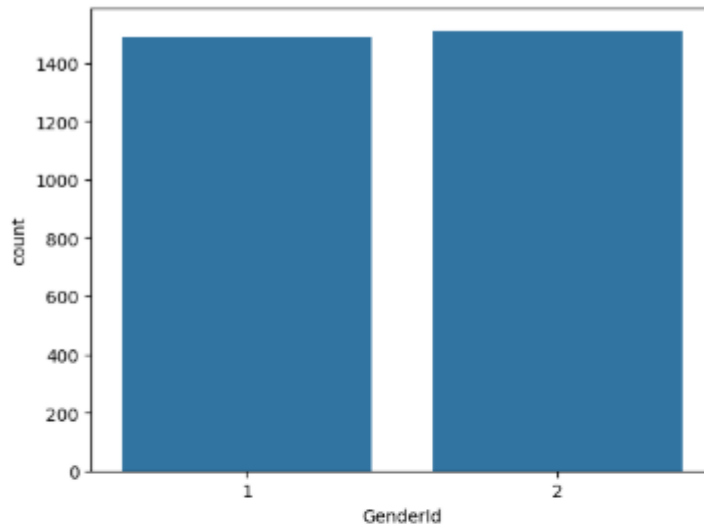
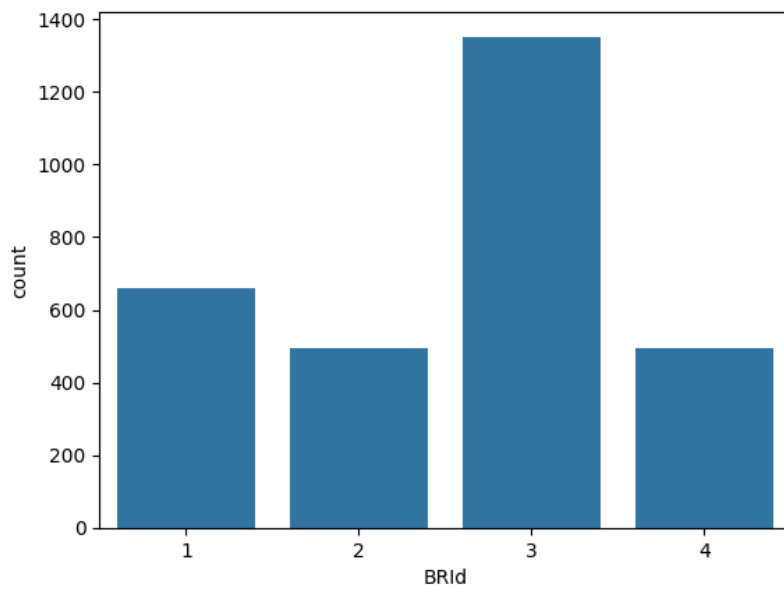
dtype: int64

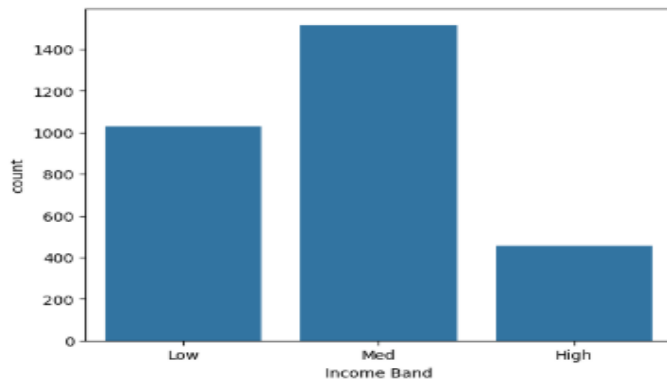
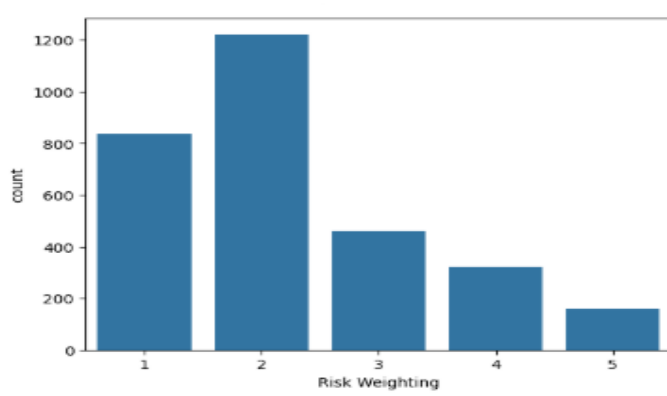
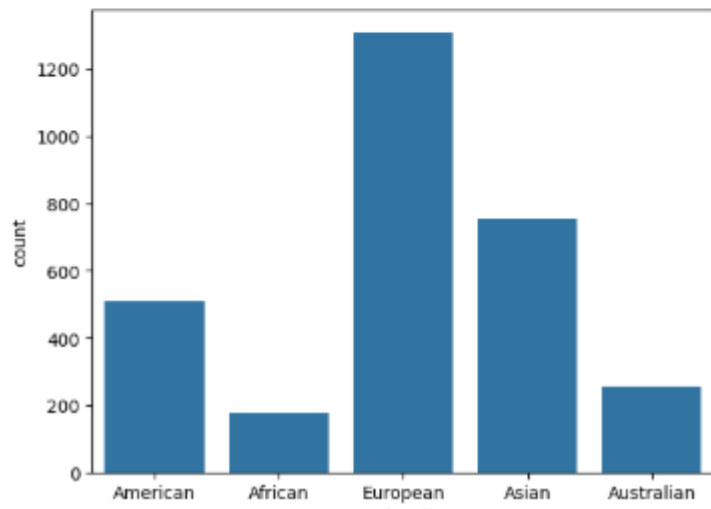
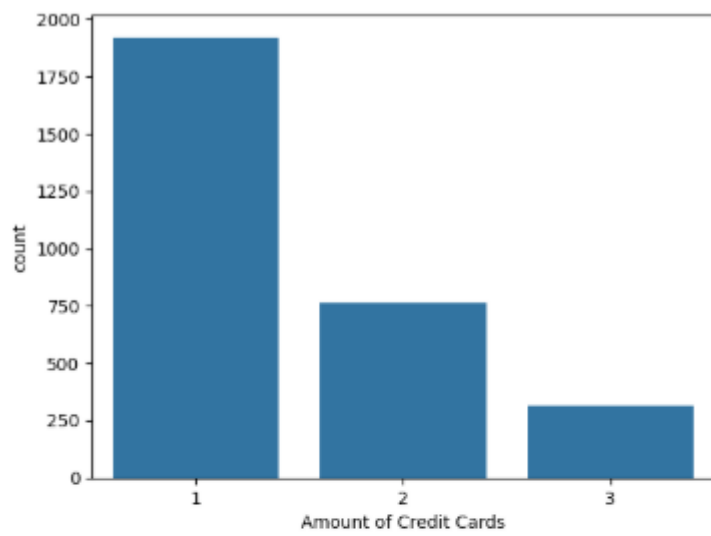
```

```

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

```

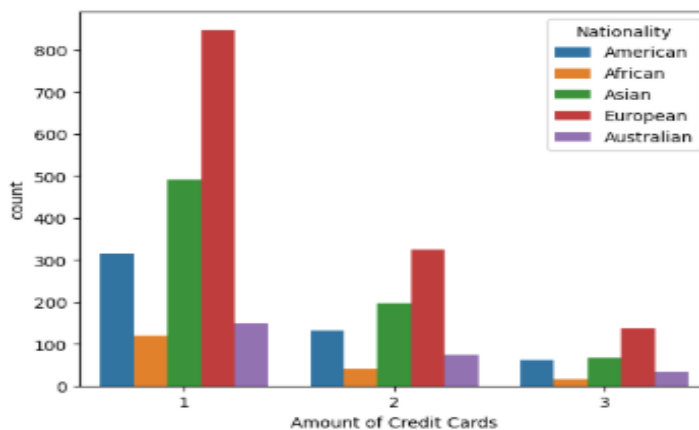
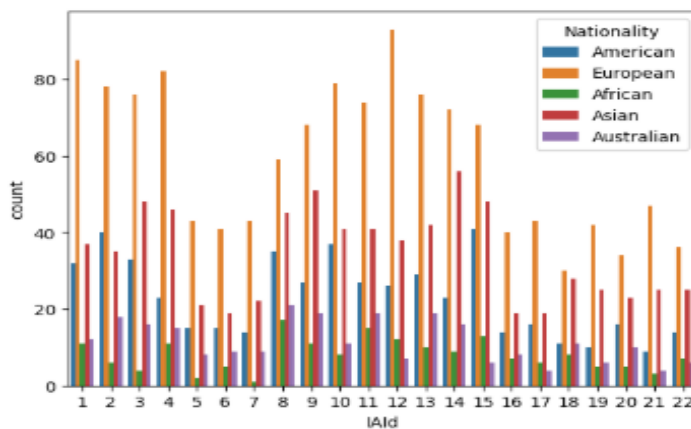
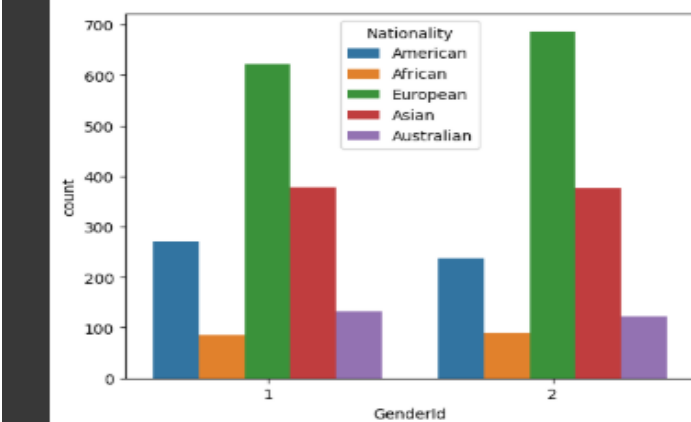
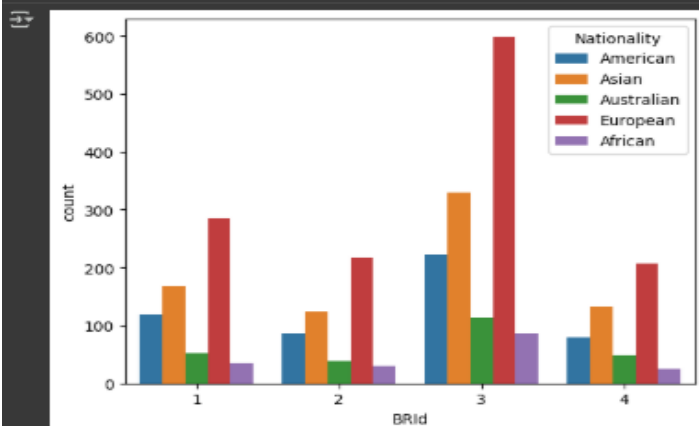


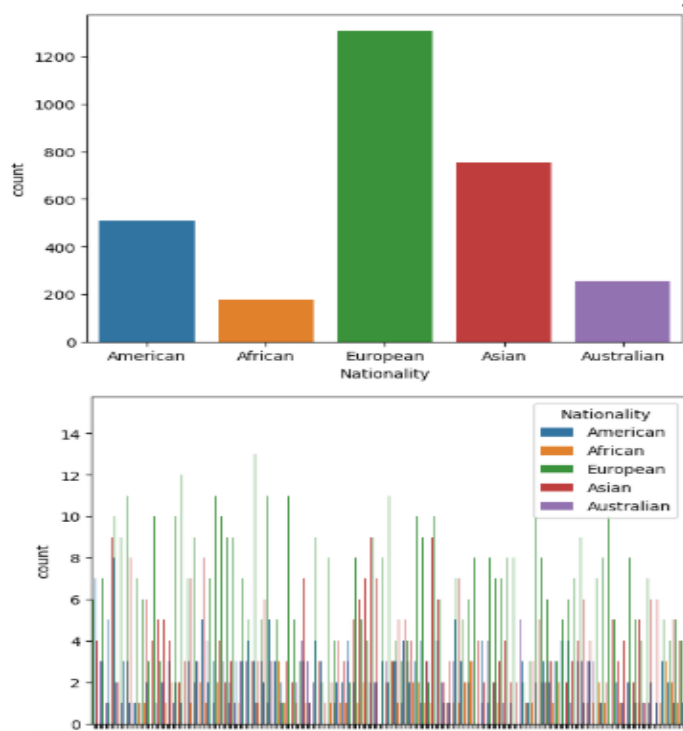


```

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

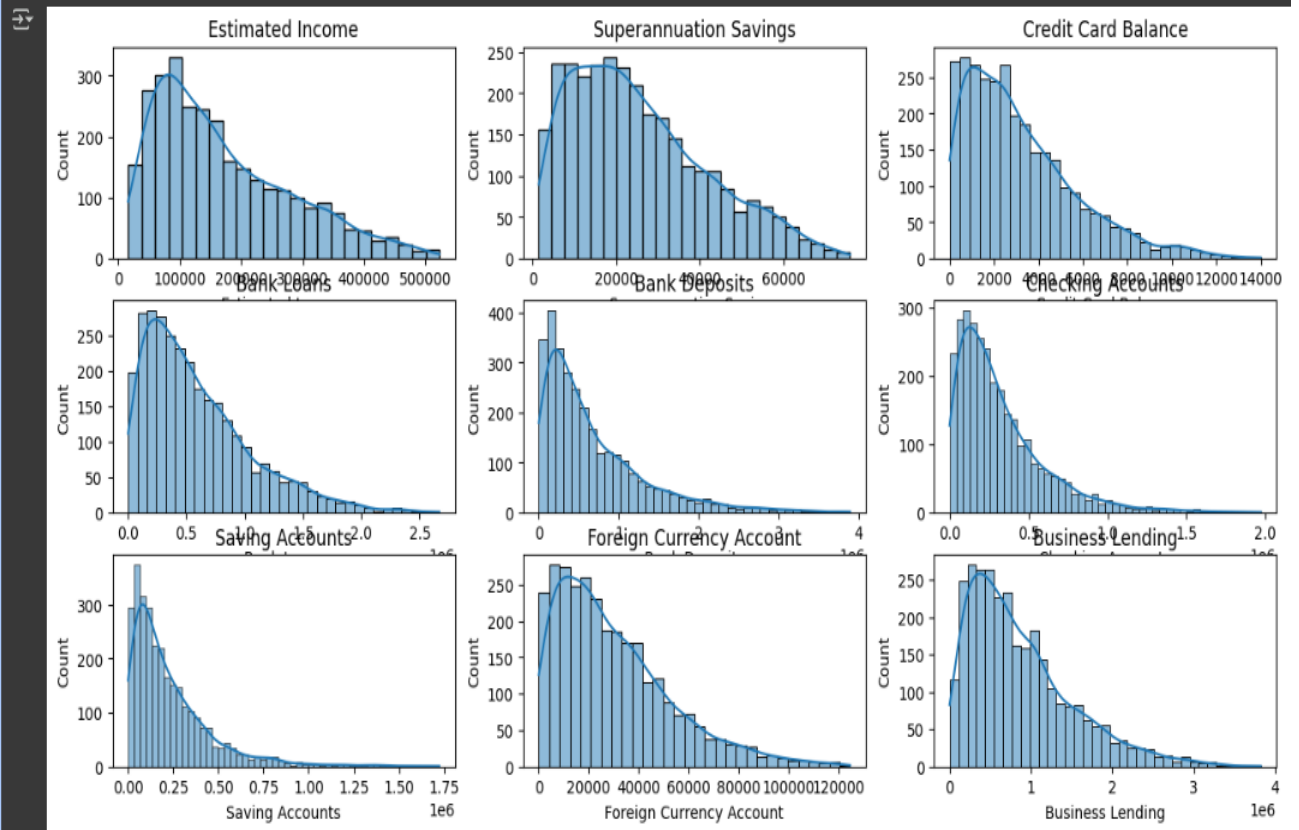
```





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

# Univariate analysis and visualization
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()
```




```

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

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()

```

