BankEDA (Version 2)

June 2, 2025

[ ]:
```python
import pandas as pd
import numpy as np
```

[12]:
```python
df = pd.read_csv('Banking.csv')
df.head()
```

[12]:
```
  Client ID             Name  Age  Location ID Joined Bank   Banking Contact  \
0  IND81288     Raymond Mills   24        34324  06-05-2019    Anthony Torres
1  IND65833     Julia Spencer   23        42205  10-12-2001  Jonathan Hawkins
2  IND47499    Stephen Murray   27         7314  25-01-2010     Anthony Berry
3  IND72498    Virginia Garza   40        34594  28-03-2019        Steve Diaz
4  IND60181   Melissa Sanders   46        41269  20-07-2012        Shawn Long

  Nationality            Occupation Fee Structure Loyalty Classification  …  \
0    American  Safety Technician IV          High                   Jade  …
1     African   Software Consultant          High                   Jade  …
2    European    Help Desk Operator          High                   Gold  …
3    American          Geologist II           Mid                 Silver  …
4    American   Assistant Professor           Mid               Platinum  …

   Bank Deposits  Checking Accounts  Saving Accounts  \
0     1485828.64          603617.88        607332.46
1      641482.79          229521.37        344635.16
2     1033401.59          652674.69        203054.35
3     1048157.49         1048157.49        234685.02
4      487782.53          446644.25        128351.45

   Foreign Currency Account  Business Lending  Properties Owned  \
0                  12249.96        1134475.30                 1
1                  61162.31        2000526.10                 1
2                  79071.78         548137.58                 1
3                  57513.65        1148402.29                 0
4                  30012.14        1674412.12                 0

   Risk Weighting  BRId  GenderId  IAId
0               2     1         1     1
1               3     2         1     2
```

```
2                3     3          2     3
3                4     4          1     4
4                3     1          2     5

[5 rows x 25 columns]
```

[6]:

```
# Check the shape of the DataFrame
print("Shape of the DataFrame:", df.shape)

# Get a concise summary of the DataFrame
print("\nDataFrame Info:")
df.info()
```

```
Shape of the DataFrame: (3000, 25)

DataFrame 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
```

[13]:
```python
df["Estimated Income"]
```

[13]:
```
0        75384.77
1       289834.31
2       169935.23
3       356808.11
4       130711.68
          …
2995    297617.14
2996     42397.46
2997     48339.88
2998    107265.87
2999     56826.53
Name: Estimated Income, Length: 3000, dtype: float64
```

[14]:
```python
# Define income band boundaries
bins = [0, 100000, 300000, float('inf')]
labels = ['Low', 'Mid', 'High']

# Create the 'Income Band' column using pd.cut
df['Income Band'] = pd.cut(df['Estimated Income'], bins=bins, labels=labels,␣
 ↪include_lowest=True)
```

[17]:
```python
# Examine the distribution of unique categories in categorical columns
categorical_cols = df[["Risk Weighting","Nationality","Occupation","Fee␣
 ↪Structure","Loyalty Classification","Properties Owned","Risk␣
 ↪Weighting","Occupation","Income Band"]].columns
for col in categorical_cols:
  # if col in ["Client ID","Name","Joined Bank"]:
  #   continue
  print(f"\nValue Counts for '{col}':")
  display(df[col].value_counts())
```

```
Value Counts for 'Risk Weighting':

Risk Weighting
2    1222
1     836
3     460
4     322
5     160
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 '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 'Income Band':

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

[9]:
```python
# Generate descriptive statistics for numerical columns
print("\nDescriptive Statistics for Numerical Columns:")
display(df.describe())
```

```
Descriptive Statistics for Numerical Columns:

               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
```

[18]:

```python
# Check for missing values
missing_values = df.isnull().sum()
print("Missing values per column:\n", missing_values)
```

```
Missing values per column:
```

```
 Client ID                 0
Name                      0
Age                       0
Location ID               0
Joined Bank               0
Banking Contact           0
Nationality               0
Occupation                0
Fee Structure             0
Loyalty Classification    0
Estimated Income          0
Superannuation Savings    0
Amount of Credit Cards    0
Credit Card Balance       0
Bank Loans                0
Bank Deposits             0
Checking Accounts         0
Saving Accounts           0
Foreign Currency Account  0
Business Lending          0
Properties Owned          0
Risk Weighting            0
BRId                      0
GenderId                  0
IAId                      0
Income Band               0
dtype: int64
```

[37]:

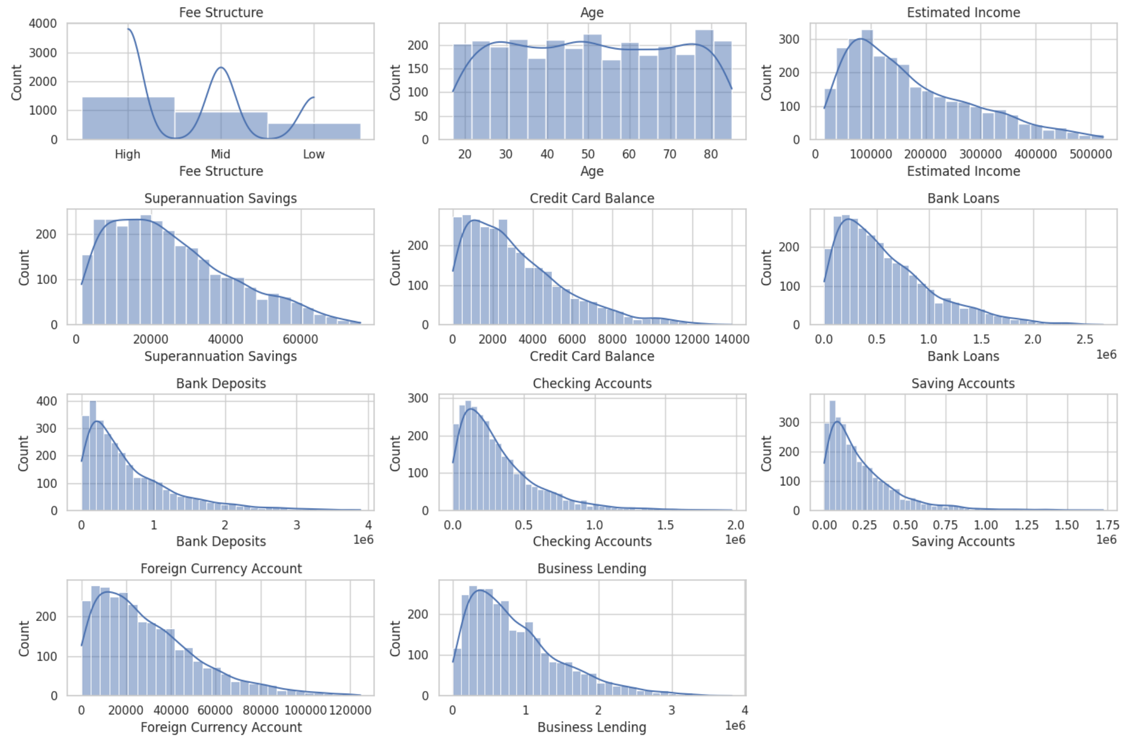
```
df['Joined Bank'] = pd.to_datetime(df['Joined Bank'], format='%d-%m-%Y')
print(df['Joined Bank'].dtype)
```

```
datetime64[ns]
```

[19]:

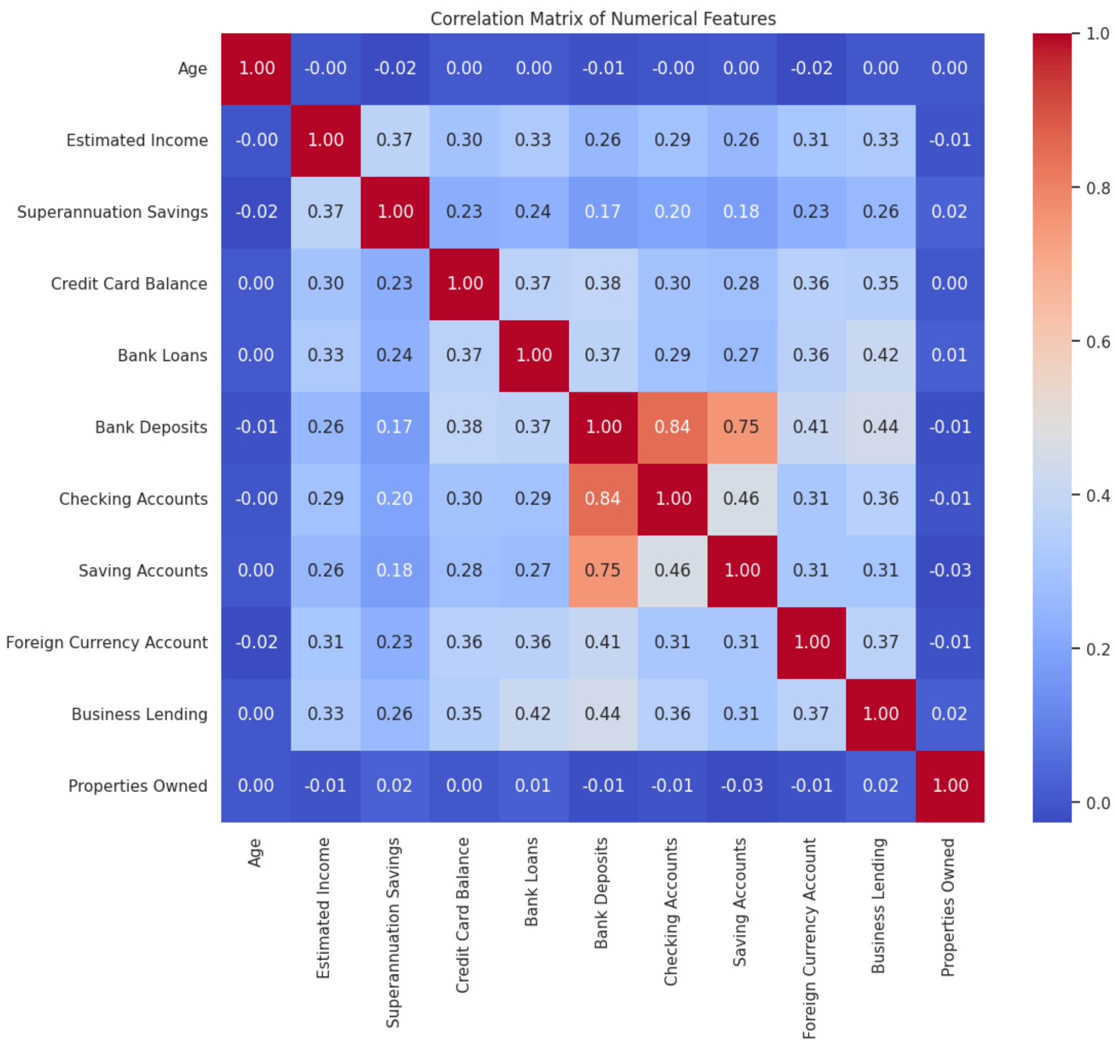
```
import matplotlib.pyplot as plt
import seaborn as sns

# Numerical analysis and exploration
numerical_cols = ['Fee Structure','Age', '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.tight_layout()
```

```
plt.show()
```



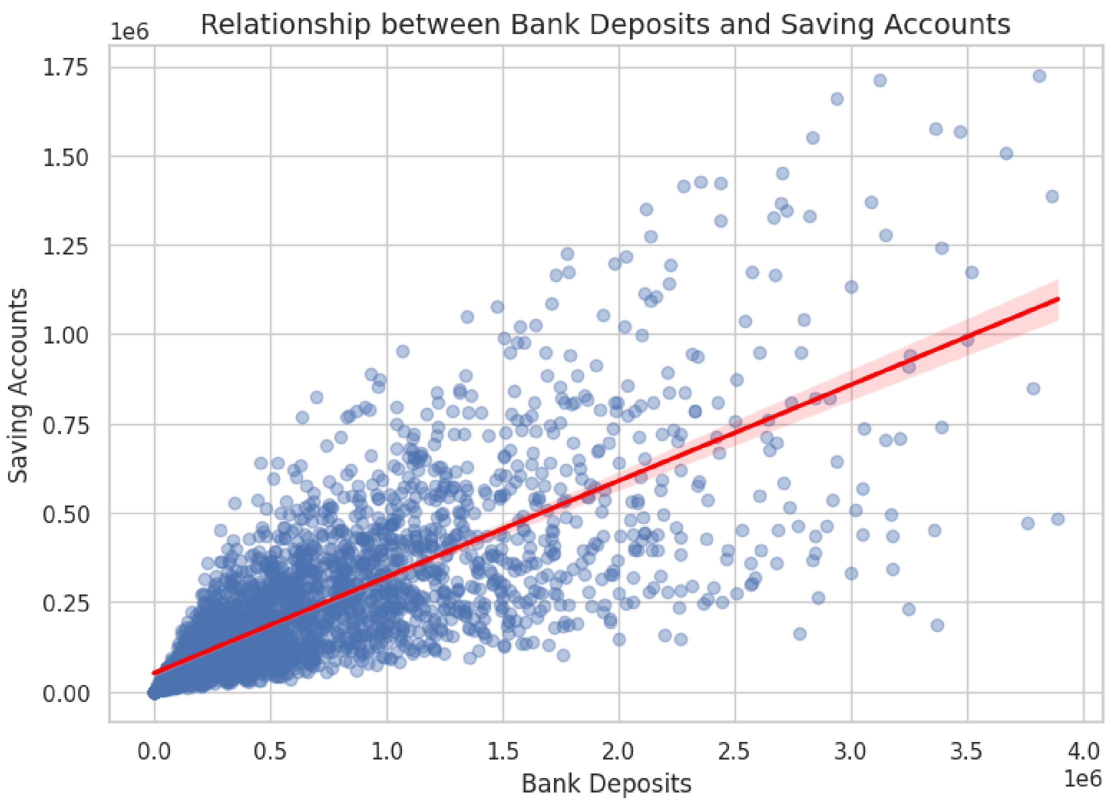
[21]:

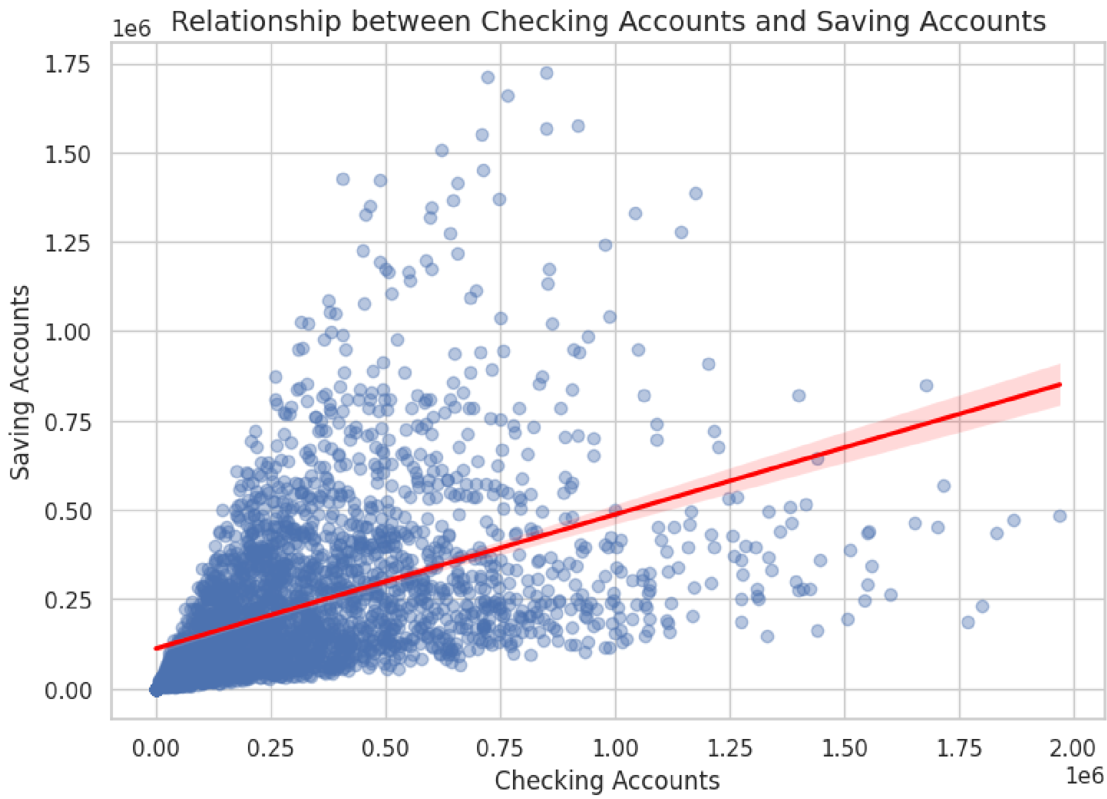
```
# Select numerical columns for correlation analysis
numerical_cols = ['Age', 'Estimated Income', 'Superannuation Savings', 'Credit Card Balance',
                  'Bank Loans', 'Bank Deposits', 'Checking Accounts', 'Saving Accounts',
                  'Foreign Currency Account', 'Business Lending', 'Properties Owned']

# Calculate the correlation matrix
correlation_matrix = df[numerical_cols].corr()

# Create a heatmap of the correlation matrix
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Numerical Features')
plt.show()
```

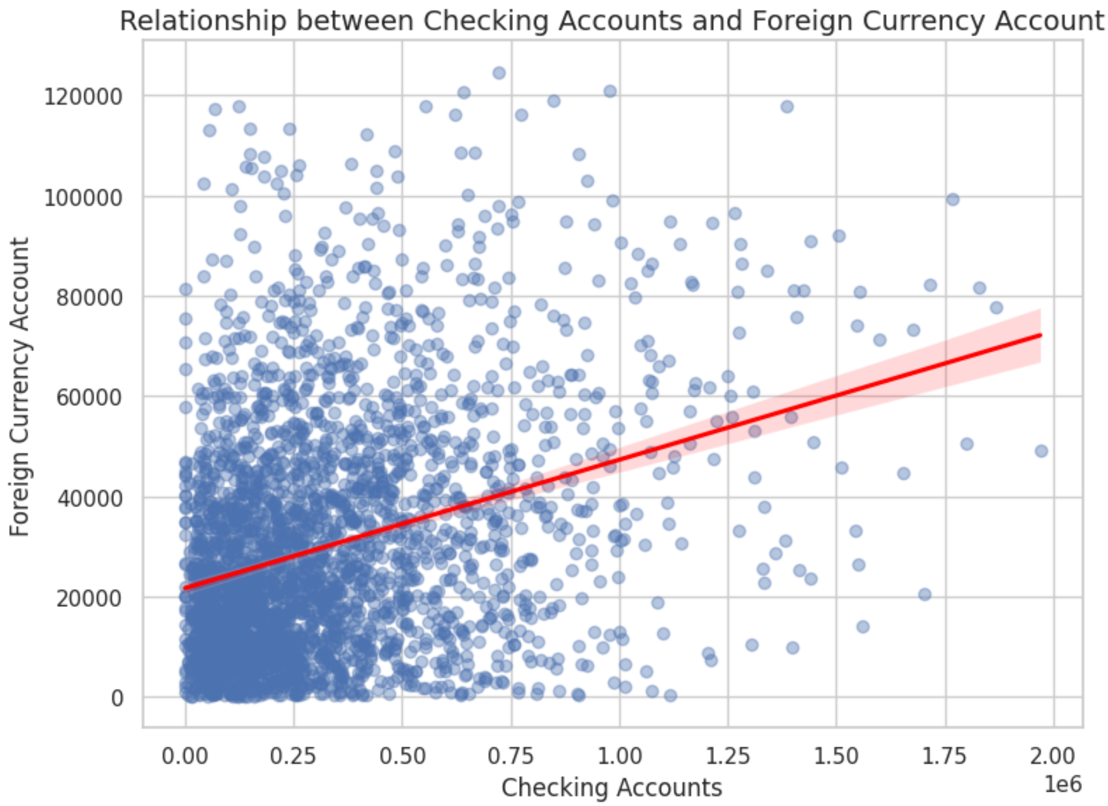


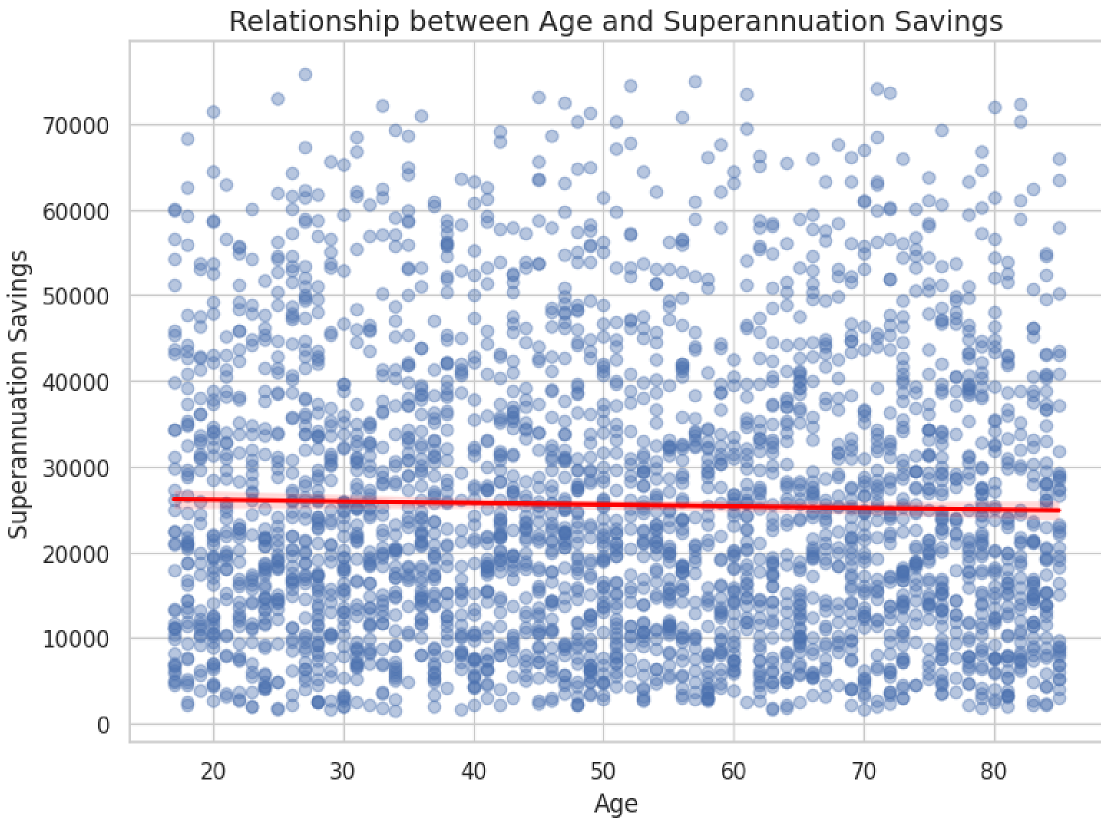
[25]:
```python
pairs_to_plot = [
    ('Bank Deposits', 'Saving Accounts'),
    ('Checking Accounts', 'Saving Accounts'),
    ('Checking Accounts', 'Foreign Currency Account'),
    ('Age', 'Superannuation Savings'),
    ('Estimated Income', 'Checking Accounts'),
    ('Bank Loans', 'Credit Card Balance'),
    ('Business Lending', 'Bank Loans'),
]


for x_col, y_col in pairs_to_plot:
    plt.figure(figsize=(8, 6))
    sns.regplot(
        data=df,
```

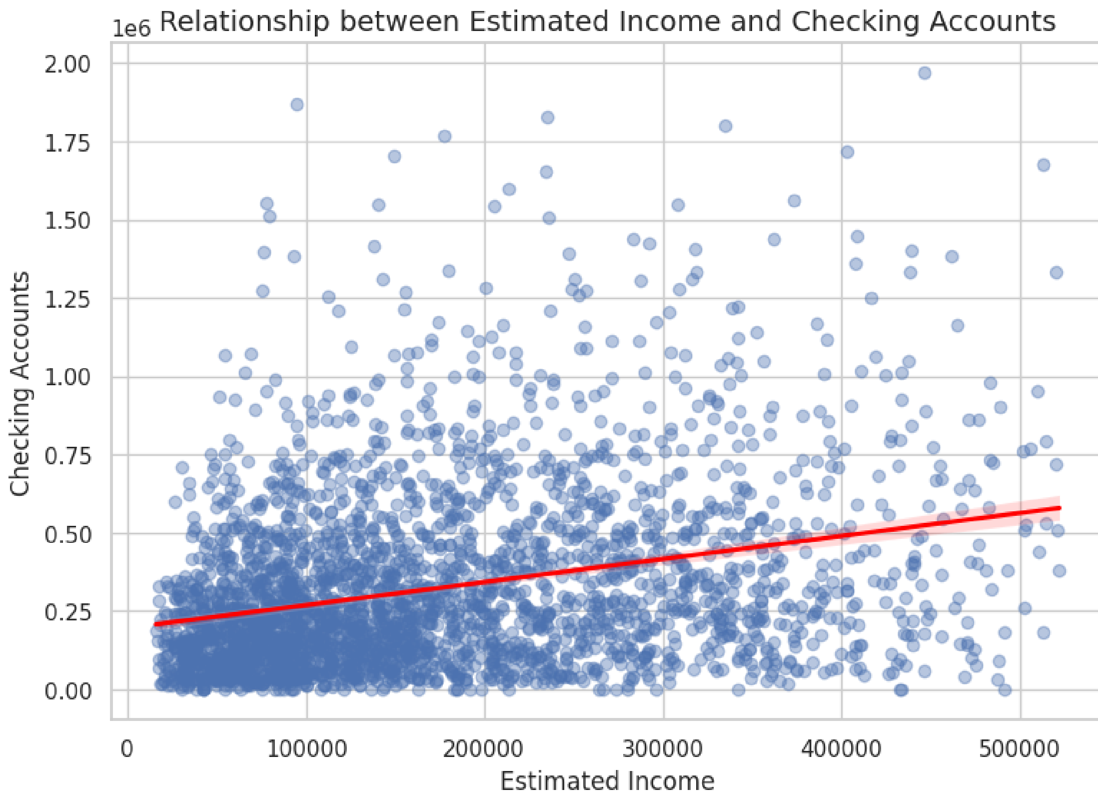
```
        x=x_col,
        y=y_col,
        scatter_kws={'alpha': 0.4},       # semi-transparent points
        line_kws={'color': 'red'}         # best-fit line color
    )
    plt.title(f'Relationship between {x_col} and {y_col}', fontsize=14)
    plt.xlabel(x_col, fontsize=12)
    plt.ylabel(y_col, fontsize=12)
    plt.tight_layout()
    plt.show()
```

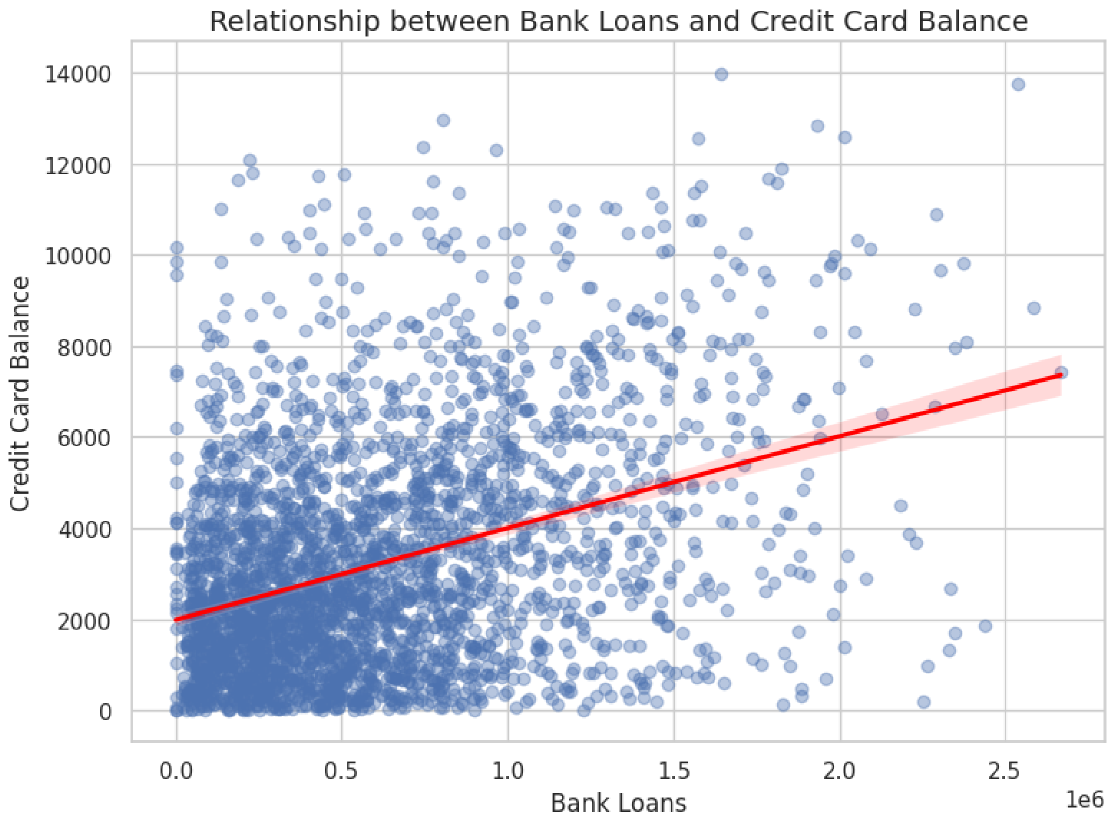



Relationship between Checking Accounts and Saving Accounts
1e6
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Saving Accounts
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
1e6
Checking Accounts

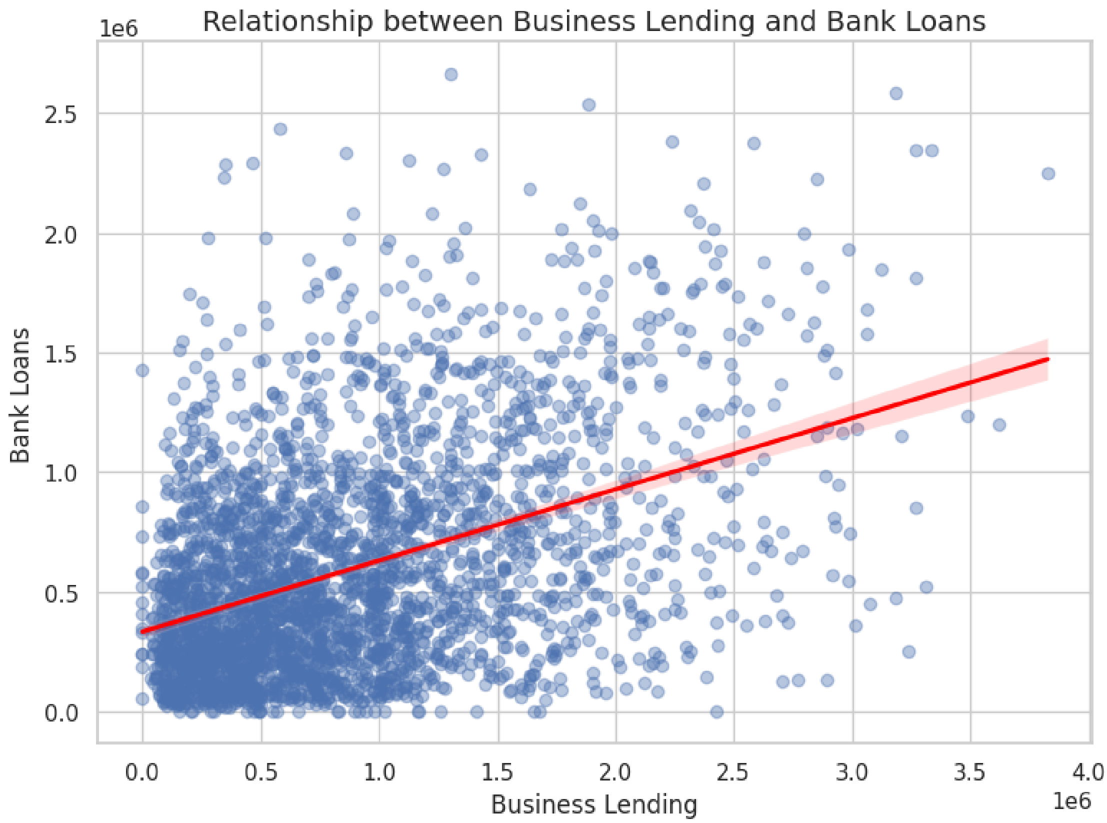

Relationship between Checking Accounts and Foreign Currency Account
Foreign Currency Account
0
20000
40000
60000
80000
100000
120000
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
1e6
Checking Accounts


Relationship between Age and Superannuation Savings
Superannuation Savings
70000
60000
50000
40000
30000
20000
10000
0
20
30
40
50
60
70
80
Age


Relationship between Estimated Income and Checking Accounts
1e6
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Checking Accounts
0
100000
200000
300000
400000
500000
Estimated Income



Relationship between Bank Loans and Credit Card Balance
14000
12000
10000
8000
6000
4000
2000
0
Credit Card Balance
0.0
0.5
1.0
1.5
2.0
2.5
1e6
Bank Loans



1 Insights:

##Deposits and Savings Behavior

The high correlation between Bank Deposits and Saving Accounts suggests that these may either measure overlapping financial behavior (e.g., total funds a customer keeps in the bank) or that people who actively deposit funds also tend to maintain or grow savings balances.

1.1 Income, Age, and Accumulation

Moderate correlations of Age and Estimated Income with various balances (Superannuation, Savings, Checking) reflect a common financial lifecycle trend: higher income earners and older individuals often accumulate more savings, retirement funds, and may carry higher credit card balances or loans.

##Low Correlation with Properties Owned

Property ownership may depend on external factors (location, real estate market conditions, inheritance, etc.) that are not captured by these particular banking variables. Hence, we see weaker correlations here.

##Business vs. Personal Banking

Business Lending's moderate link to Bank Loans suggests some customers may have both personal and business debts. However, business lending is relatively uncorrelated with other deposit or property-related metrics, indicating it may serve a distinct subset of customers or needs.

[ ]:

```

```