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

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

Out[ ]:

	Client ID	Name	Age	Location ID	Joined Bank	Banking Contact	Nationality	Occupati
0	IND81288	Raymond Mills	24	34324	06-05-2019	Anthony Torres	American	Saf Technic
1	IND65833	Julia Spencer	23	42205	10-12-2001	Jonathan Hawkins	African	Softw Consult
2	IND47499	Stephen Murray	27	7314	25-01-2010	Anthony Berry	European	Help D Opera
3	IND72498	Virginia Garza	40	34594	28-03-2019	Steve Diaz	American	Geologis
4	IND60181	Melissa Sanders	46	41269	20-07-2012	Shawn Long	American	Assist Profes

5 rows × 25 columns

In [ ]:
```
# 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
```

In [ ]:

```
df["Estimated Income"]
```

Out[ ]:

	Estimated Income
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

3000 rows × 1 columns

dtype: float64

In [ ]:

```
# 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, i
```

In [ ]:

```
# Examine the distribution of unique categories in categorical columns
categorical_cols = df[["Risk Weighting","Nationality","Occupation","Fee Struct
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	count
2	1222
1	836
3	460
4	322
5	160

dtype: int64

```
Value Counts for 'Nationality':
```

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

dtype: int64

```
Value Counts for 'Occupation':
```

Occupation	count
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':
```

Fee Structure	count
High	1476
Mid	962
Low	562

dtype: int64

```
Value Counts for 'Loyalty Classification':
```

Loyalty Classification	count
Jade	1331
Silver	767
Gold	585
Platinum	317

dtype: int64

```
Value Counts for 'Properties Owned':
```

Properties Owned	count
2	777
1	776
3	742
0	705

dtype: int64

```
Value Counts for 'Risk Weighting':
```

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

dtype: int64

```
Value Counts for 'Occupation':
```

Occupation	count
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 'Income Band':
```

Income Band	count
Mid	1517
Low	1027
High	456

dtype: int64

In [ ]:

```
# 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	Amount of Credit Cards
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000
mean	51.039667	21563.323000	171305.034263	25531.599673	1.463667
std	19.854760	12462.273017	111935.808209	16259.950770	0.676387
min	17.000000	12.000000	15919.480000	1482.030000	1.000000
25%	34.000000	10803.500000	82906.595000	12513.775000	1.000000
50%	51.000000	21129.500000	142313.480000	22357.355000	1.000000
75%	69.000000	32054.500000	242290.305000	35464.740000	2.000000
max	85.000000	43369.000000	522330.260000	75963.900000	3.000000

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

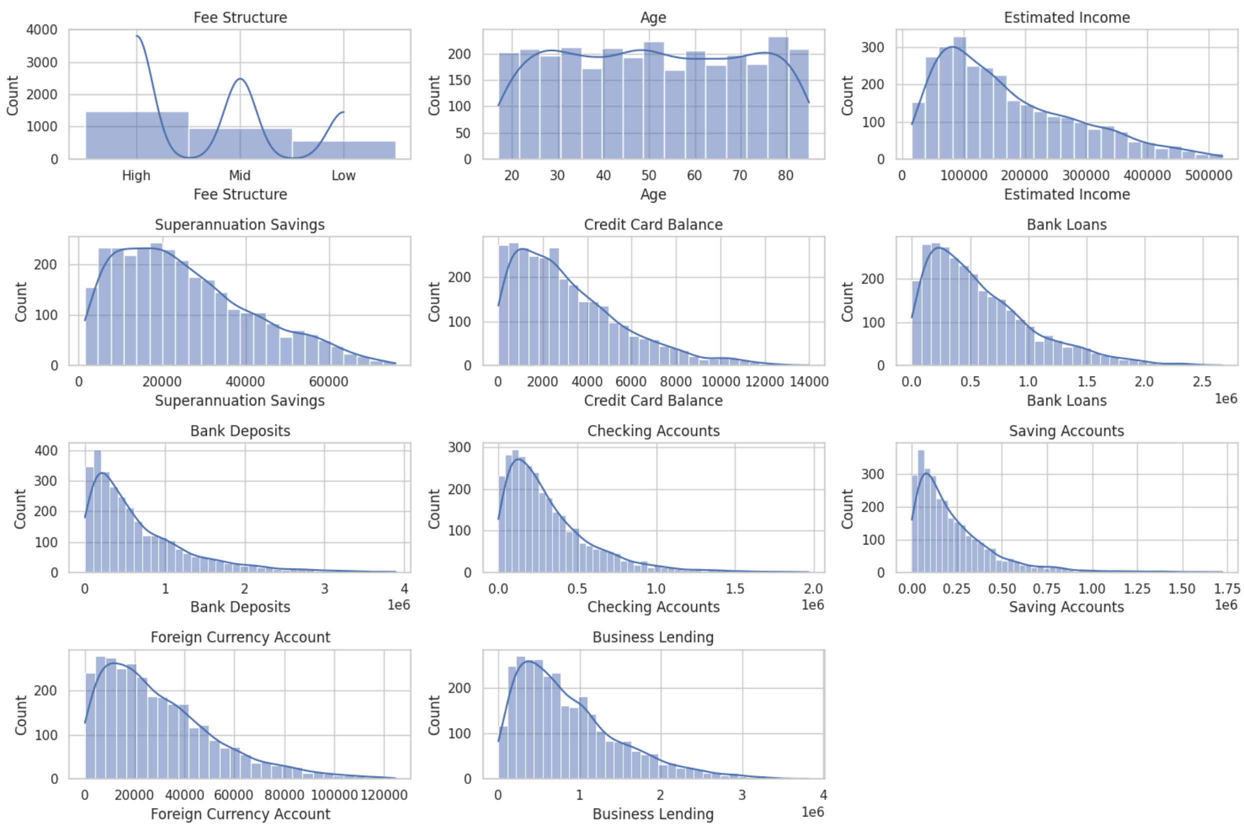
```
Missing values per column:
 Client ID                   0
Name                        0
Age                         0
Location ID                 0
Joined Bank                 0
Banking Contact             0
Nationality                 0
Occupation                  0
Fee Structure               0
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
```

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

datetime64[ns]

In [ ]:

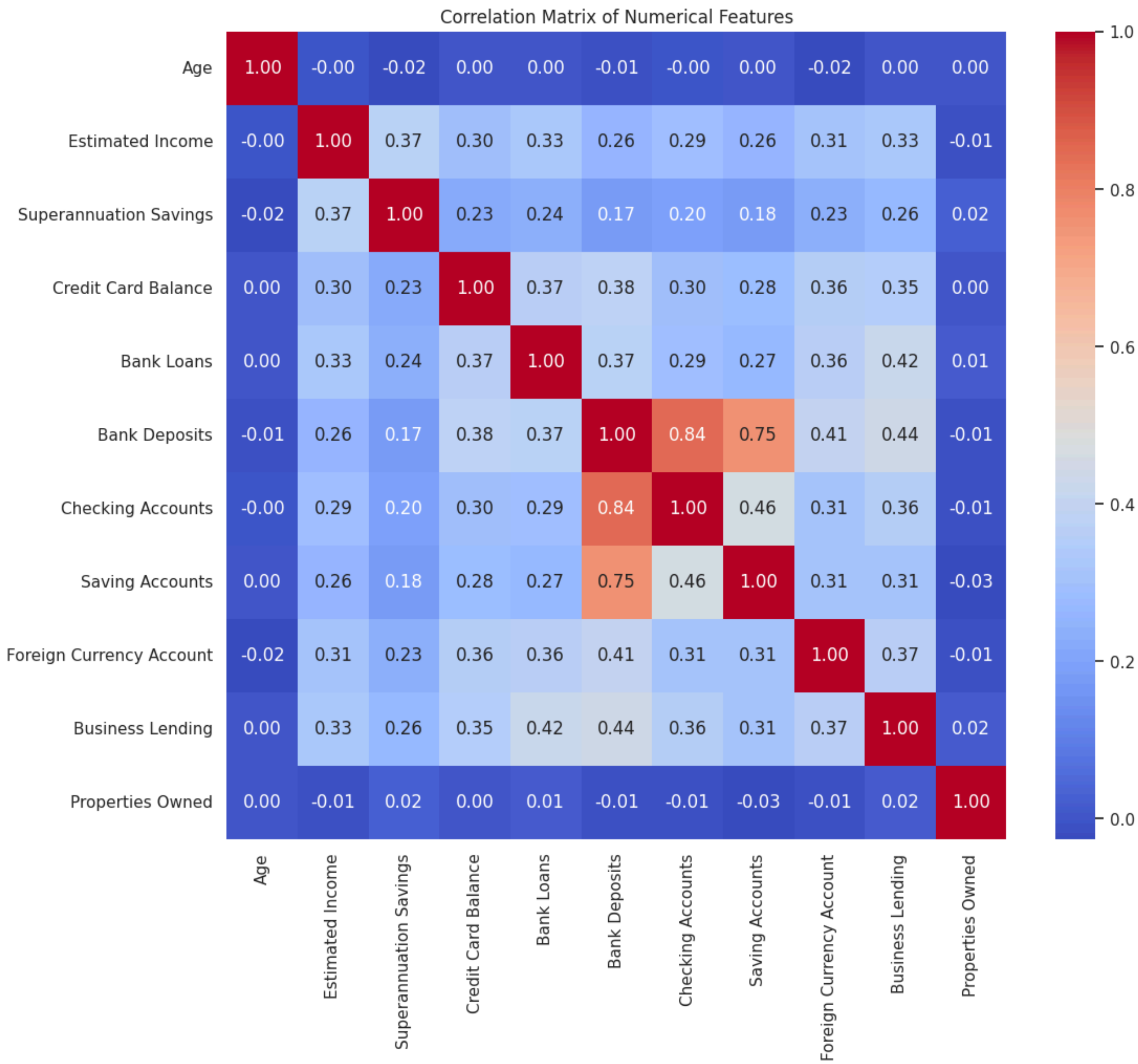
```python
import matplotlib.pyplot as plt
import seaborn as sns

# Numerical analysis and exploration
numerical_cols = ['Fee Structure','Age', 'Estimated Income', 'Superannuation S

# 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()
```



In [ ]:

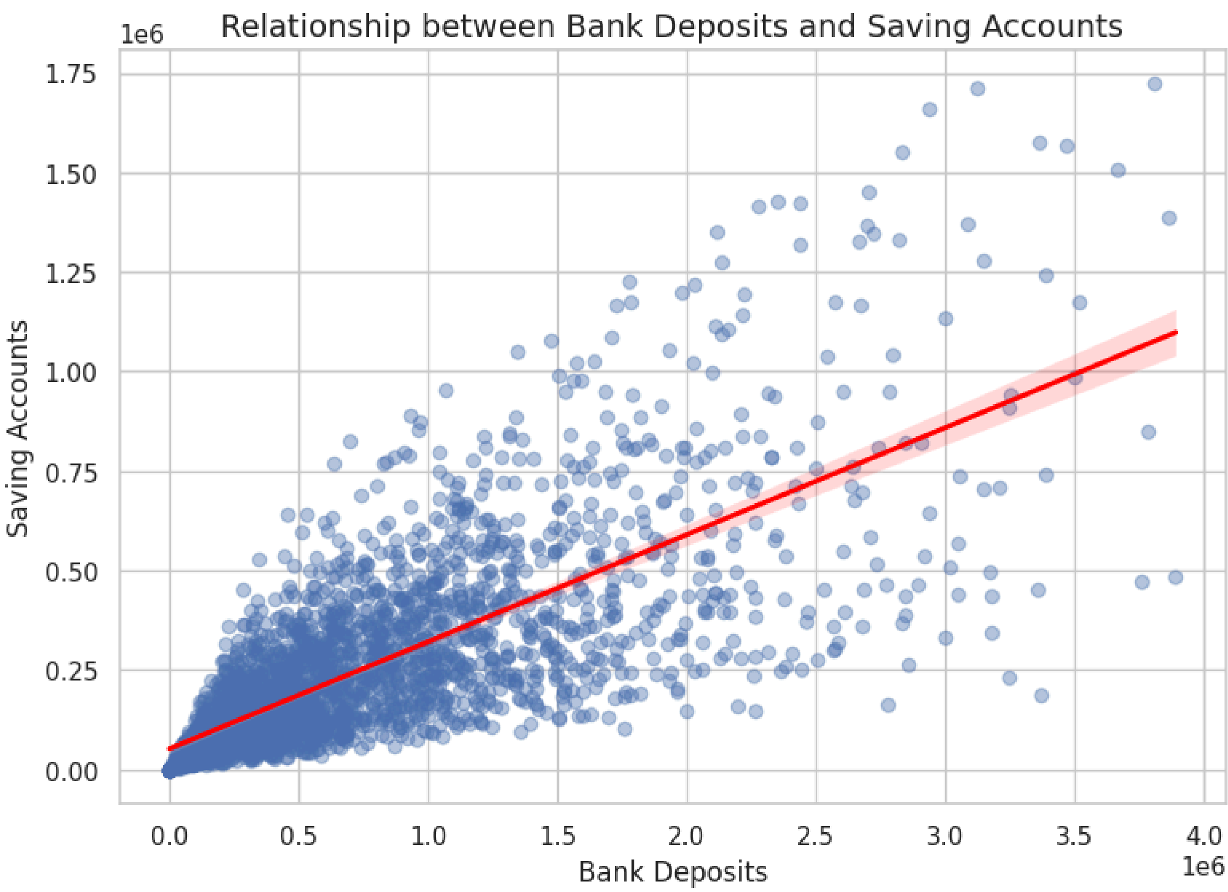
```python
# Select numerical columns for correlation analysis
numerical_cols = ['Age', 'Estimated Income', 'Superannuation Savings', 'Credit
                  'Bank Loans', 'Bank Deposits', 'Checking Accounts', 'Saving
                  'Foreign Currency Account', 'Business Lending', 'Properties

# 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")
```

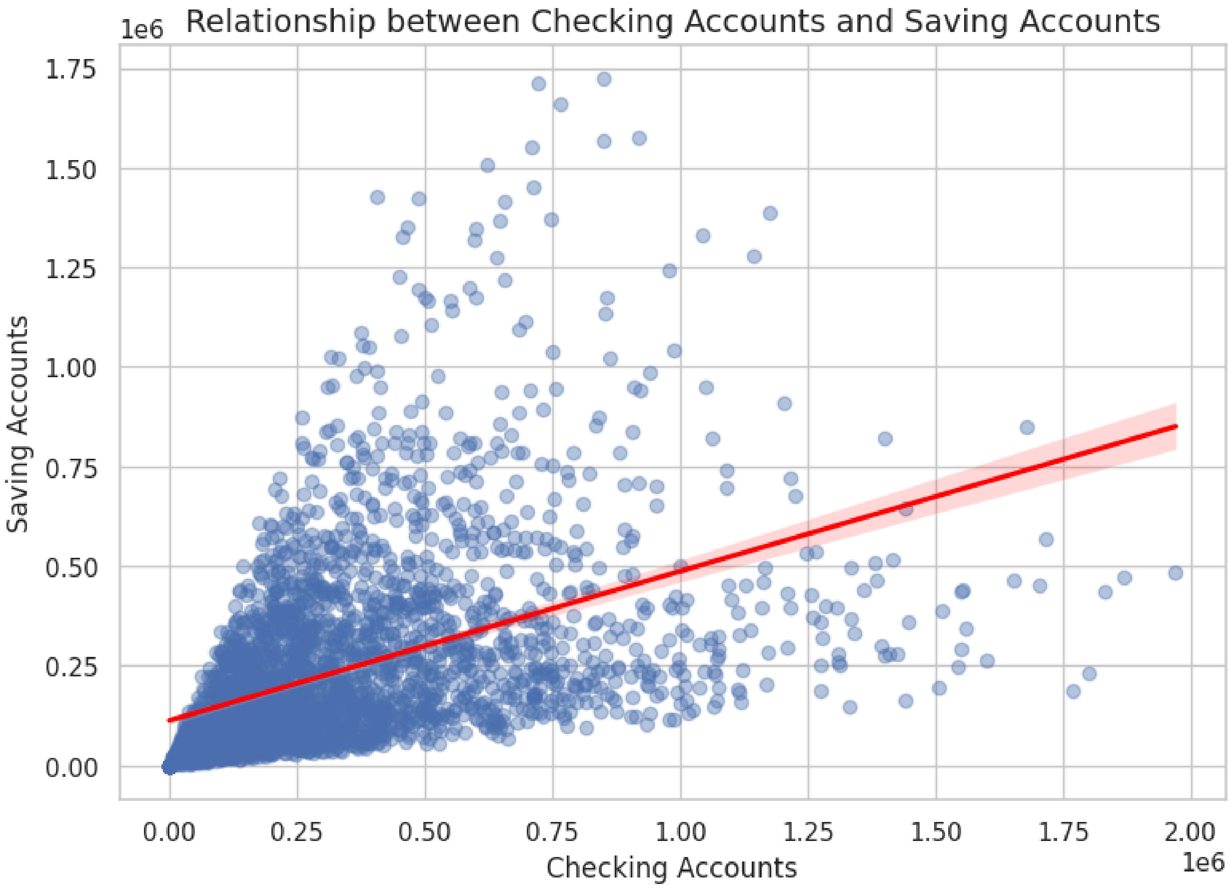
```python
plt.title('Correlation Matrix of Numerical Features')
plt.show()
```

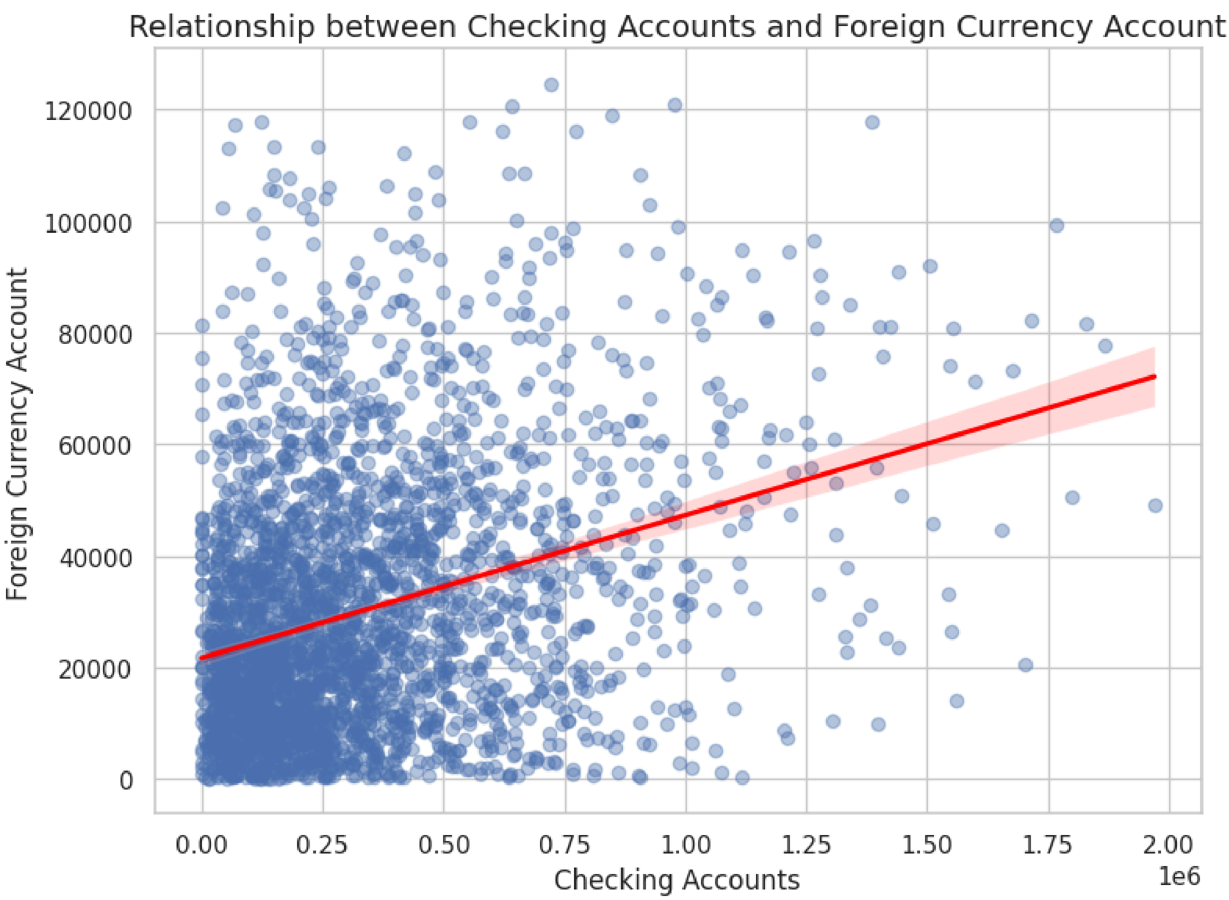


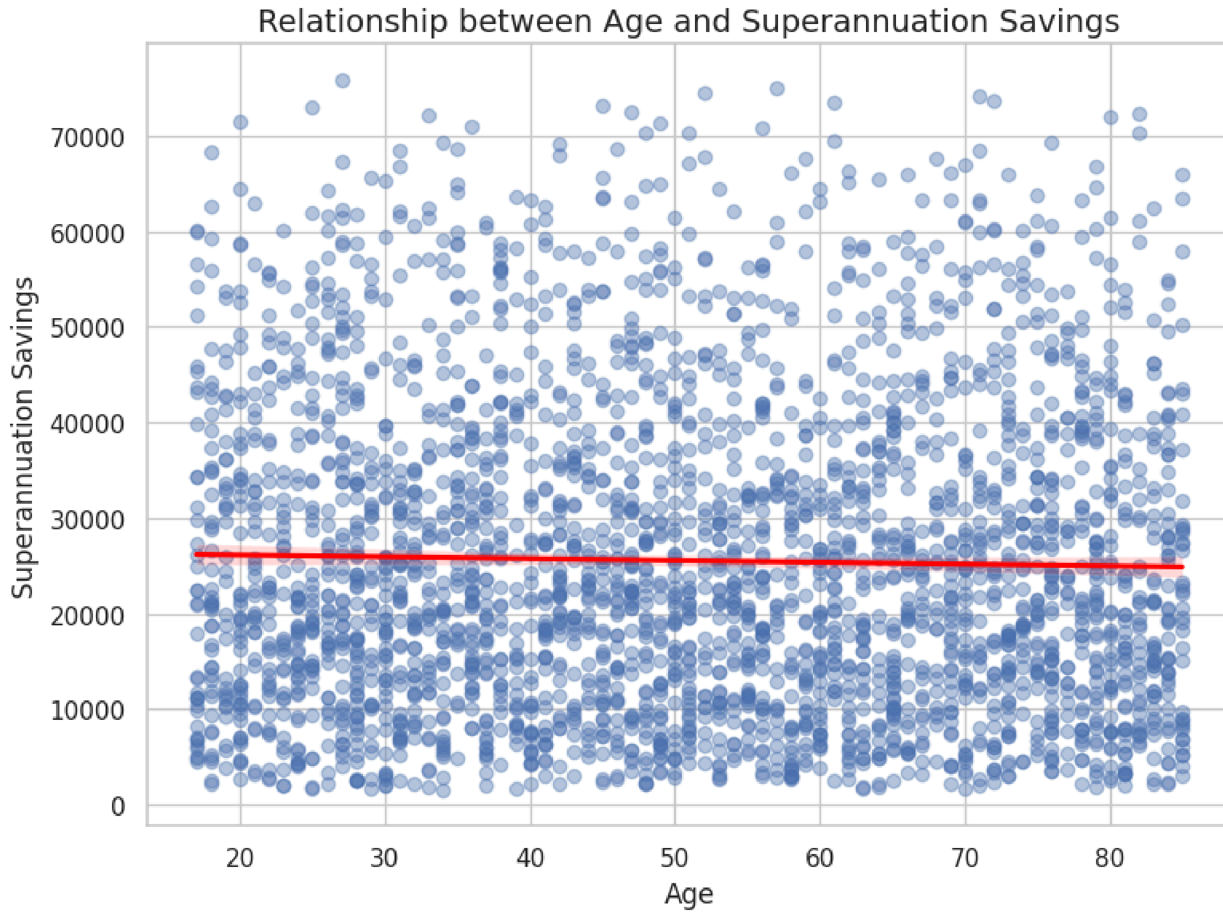
In [ ]:
```python
pairs_to_plot = [
    ('Bank Deposits', 'Saving Accounts'),
    ('Checking Accounts', 'Saving Accounts'),
    ('Checking Accounts', 'Foreign Currency Account'),
    ('Age', 'Superannuation Savings'),
    ('Estimated Income', 'Checking Accounts'),
    ('Bank Loans', 'Credit Card Balance'),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


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

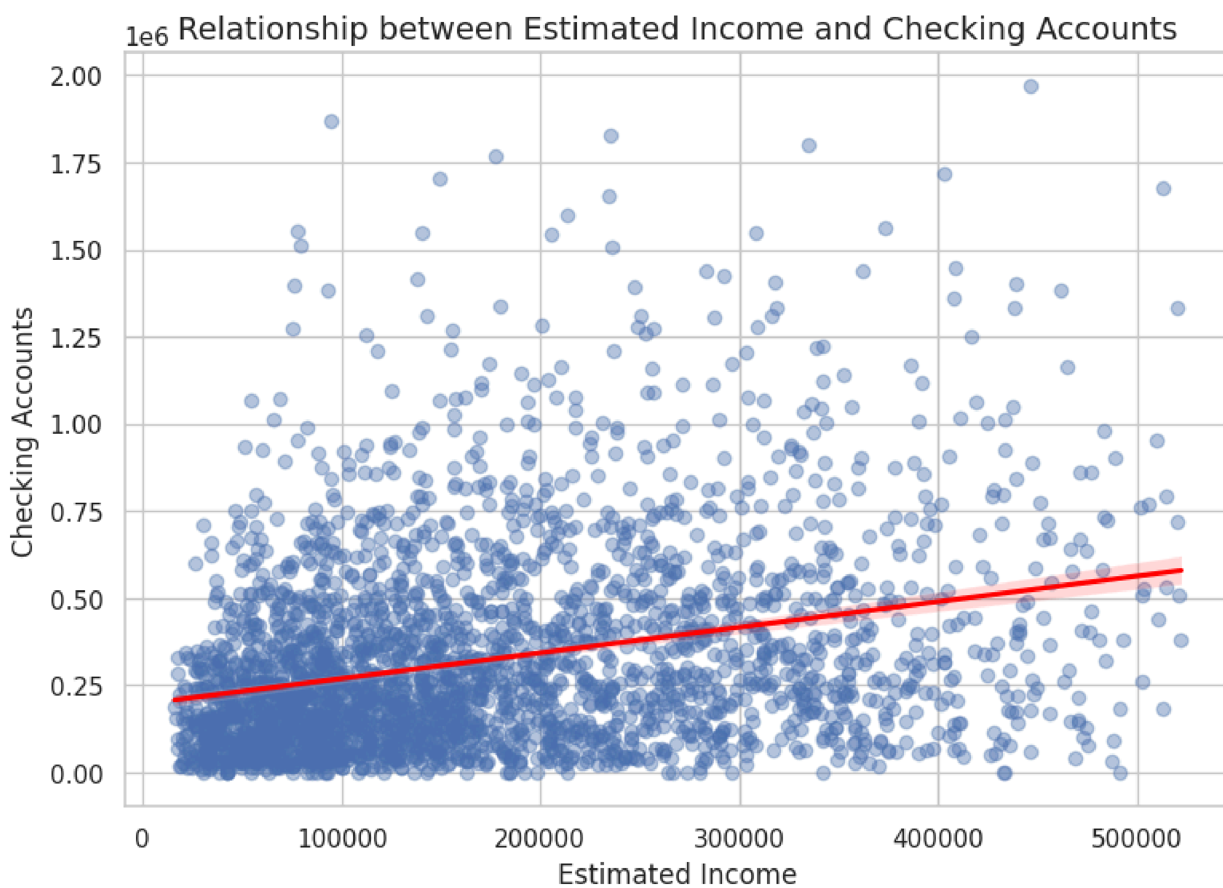
```
        x=x_col,
        y=y_col,
        scatter_kws={'alpha': 0.4},     # semi-transparent points
        line_kws={'color': 'red'}       # best-fit line color
    )
    plt.title(f'Relationship between {x_col} and {y_col}', fontsize=14)
    plt.xlabel(x_col, fontsize=12)
    plt.ylabel(y_col,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

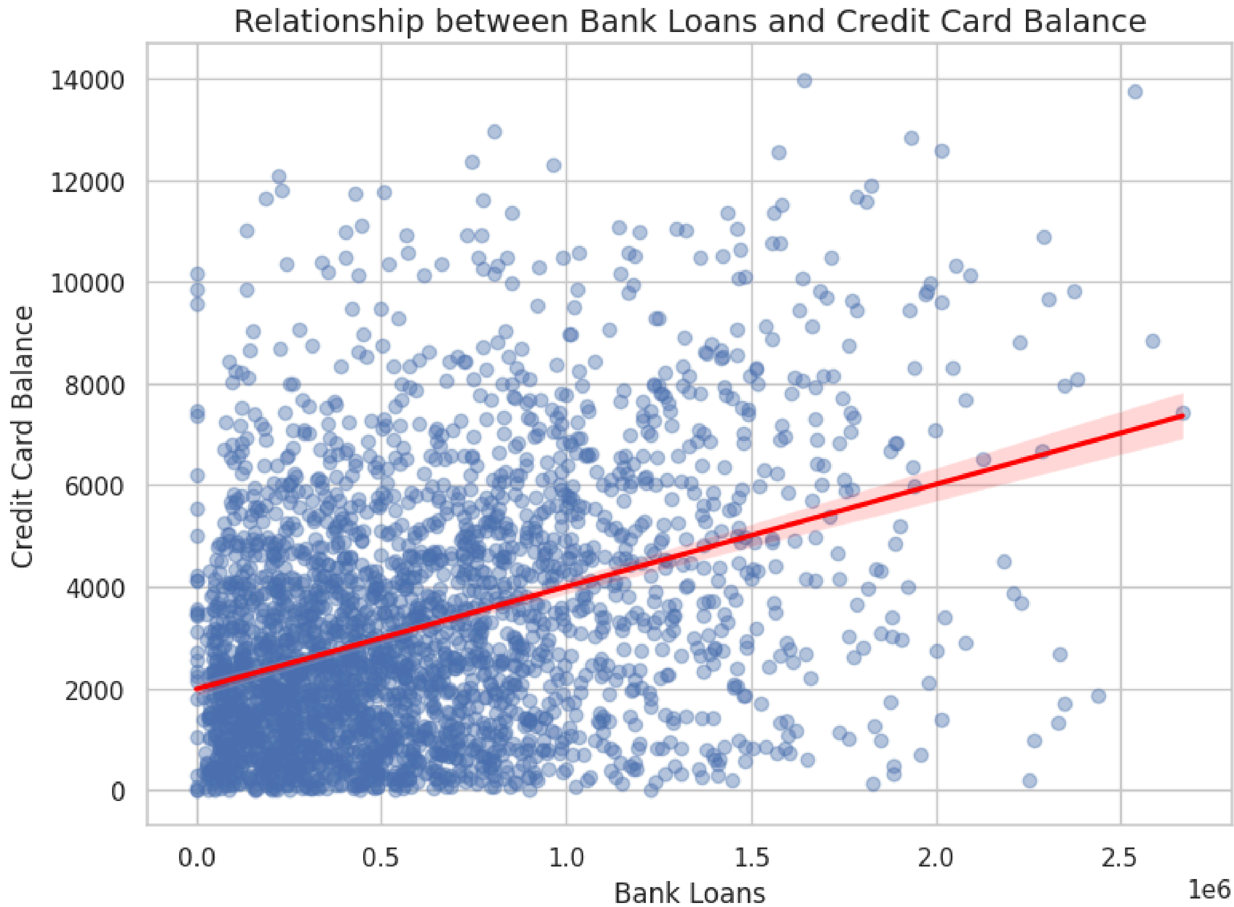


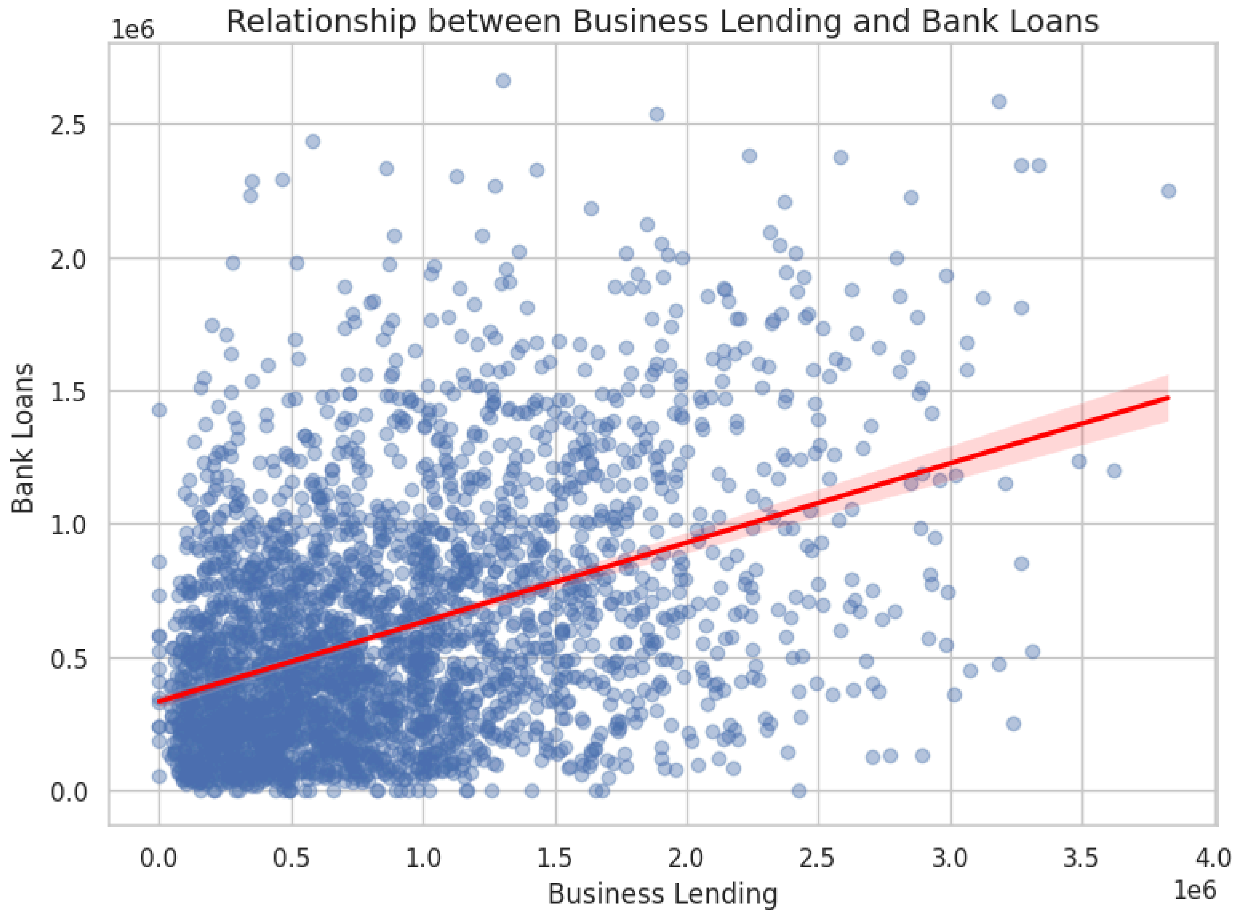

Relationship between Checking Accounts and Saving Accounts
1e6
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Saving Accounts
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
1e6
Checking Accounts


Relationship between Checking Accounts and Foreign Currency Account
120000
100000
80000
60000
40000
20000
0
Foreign Currency Account
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
Age
0
10000
20000
30000
40000
50000
60000
70000
20
30
40
50
60
70
80


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


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



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.

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.

In [ ]:

```

```