In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
import mysql.connector

# Connect to MySQL
conn = mysql.connector.connect(
    host="127.0.0.1",
    user="root",
    password="root",
    port=3306  # Optional, since 3306 is default
    # database="your_database"  # Add this if you have a DB name
)

# Check connection
if conn.is_connected():
    print("✅ Successfully connected to MySQL!")

conn.close()
```

```
✅ Successfully connected to MySQL!
```

In [3]:
```
query  = "SELECT * FROM customers.customer"
```

In [4]:
```
import mysql.connector
import pandas as pd

# Step 1: Connect to MySQL database
conn = mysql.connector.connect(
    host="127.0.0.1",
    user="root",
    password="root",
    port=3306,
    database="customers"  # <-- Make sure this matches your DB name
)

# Step 2: Query to load data from 'customer' table
query = "SELECT * FROM customer"

# Step 3: Use pandas to load the table into a DataFrame
df = pd.read_sql(query, conn)

# Step 4: Display results
print(df.head())

# Step 5: Close connection
conn.close()
```

```
C:\Users\Nitish\AppData\Local\Temp\ipykernel_5096\4242630259.py:17: UserWarning: pan
das only supports SQLAlchemy connectable (engine/connection) or database string URI
or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider u
sing SQLAlchemy.
  df = pd.read_sql(query, conn)
```

```
  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]
```

In [5]:
```
df.head()
```

Out[5]:

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

5 rows × 25 columns

In [6]:
```
df.describe()
```

Out[6]:

	Age	Location ID	Estimated Income	Superannuation Savings	Amount of Credit Cards	Credit Ca Balan
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.0000
mean	51.039667	21563.323000	171305.034263	25531.599673	1.463667	3176.2069
std	19.854760	12462.273017	111935.808209	16259.950770	0.676387	2497.0947
min	17.000000	12.000000	15919.480000	1482.030000	1.000000	1.1700
25%	34.000000	10803.500000	82906.595000	12513.775000	1.000000	1236.6300
50%	51.000000	21129.500000	142313.480000	22357.355000	1.000000	2560.8050
75%	69.000000	32054.500000	242290.305000	35464.740000	2.000000	4522.6325
max	85.000000	43369.000000	522330.260000	75963.900000	3.000000	13991.9900

In [7]:
```
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
```

In [8]:
```
bins  =  [0,100000,300000,float('inf')]
labels = ["Low","Medium","High"]
df['income_bucket'] = pd.cut(df['Estimated Income'], bins=bins , labels=labels, rig
```

In [9]:
```
df.head()
```

Out[9]:

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

5 rows × 26 columns

In [11]:
```
df.columns
```

Out[11]:
```
Index(['ï»¿Client ID', 'Name', 'Age', 'Location ID', 'Joined Bank',
       'Banking Contact', 'Nationality', 'Occupation', 'Fee Structure',
       'Loyalty Classification', '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', 'IAId', 'income_bucket'],
      dtype='object')
```

In [13]:
```
df = df.rename(columns={'ï»¿Client ID' :'Client ID'})
```

In [14]:
```
df.head(1)
```

Out[14]:

	Client ID	Name	Age	Location ID	Joined Bank	Banking Contact	Nationality	Occupation	Fe Structur
0	IND81288	Raymond Mills	24	34324	06-05-2019	Anthony Torres	American	Safety Technician IV	Hig

1 rows × 26 columns

In [16]:
```
#  Rename columns (remove spaces, lowercase)
df.columns = df.columns.str.strip().str.lower().str.replace(" ", "_")

# Convert dates to datetime
```

```
df['joined_bank'] = pd.to_datetime(df['joined_bank'], format="%d-%m-%Y", errors="co

# Handle missing values
df = df.dropna(subset=['risk_weighting'])  # if risk_weighting is target

# Convert categorical to category type
cat_cols = df.select_dtypes(include='object').columns
for col in cat_cols:
    df[col] = df[col].astype('category')
```

In [25]:
```
df.head()
```

Out[25]:

	client_id	name	age	location_id	joined_bank	banking_contact	nationality	occupa
0	IND81288	Raymond Mills	24	34324	2019-05-06	Anthony Torres	American	S Techr
1	IND65833	Julia Spencer	23	42205	2001-12-10	Jonathan Hawkins	African	Soft Consu
2	IND47499	Stephen Murray	27	7314	2010-01-25	Anthony Berry	European	Help Ope
3	IND72498	Virginia Garza	40	34594	2019-03-28	Steve Diaz	American	Geolo
4	IND60181	Melissa Sanders	46	41269	2012-07-20	Shawn Long	American	Assi Profe

5 rows × 26 columns

In [26]:
```
df.shape
```

Out[26]:
```
(3000, 26)
```

In [27]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 26 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   client_id                 3000 non-null   category
 1   name                      3000 non-null   category
 2   age                       3000 non-null   int64
 3   location_id               3000 non-null   int64
 4   joined_bank               3000 non-null   datetime64[ns]
 5   banking_contact           3000 non-null   category
 6   nationality               3000 non-null   category
 7   occupation                3000 non-null   category
 8   fee_structure             3000 non-null   category
 9   loyalty_classification    3000 non-null   category
 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
 25  income_bucket             3000 non-null   category
dtypes: category(8), datetime64[ns](1), float64(9), int64(8)
memory usage: 636.7 KB
```

In [28]:
```
df.isnull().sum()
```

Out[28]:
```
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_bucket              0
dtype: int64
```

In [31]:
```
# Show number of unique values for each categorical column
df.select_dtypes(include='category').nunique()
```

Out[31]:
```
client_id                2940
name                     2913
banking_contact            49
nationality                 5
occupation                195
fee_structure               3
loyalty_classification      4
income_bucket               3
dtype: int64
```

distribution of age

In [33]:
```
df['age'].describe()
```

Out[33]:
```
count    3000.000000
mean       51.039667
std        19.854760
min        17.000000
25%        34.000000
50%        51.000000
75%        69.000000
max        85.000000
Name: age, dtype: float64
```

How many customers are in each nationality

In [35]:
```
df['nationality'].value_counts()
```

Out[35]:
```
nationality
European      1309
Asian          754
American       507
Australian     254
African        176
Name: count, dtype: int64
```

In [36]:
```
df['occupation'].value_counts()
```

Out[36]:
```
occupation
Structural Analysis Engineer    28
Associate Professor             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
```

In [ ]:
```
plt.hisplot(
```

In [38]:
```
df['genderid'].value_counts()
```

Out[38]:
```
genderid
2    1512
1    1488
Name: count, dtype: int64
```

average estimated_income by income_bucket

In [42]:
```
df.groupby('income_bucket')['estimated_income'].mean()
```

Out[42]:
```
income_bucket
Low        64689.030399
Medium    182132.356025
High      375405.009825
Name: estimated_income, dtype: float64
```

the total and average superannuation_savings by loyalty_classification

In [43]:
```
df.groupby('loyalty_classification')['superannuation_savings'].sum()
```

Out[43]:
```
loyalty_classification
Gold        14779546.34
Jade        33486482.82
Platinum     7854313.46
Silver      20474456.40
Name: superannuation_savings, dtype: float64
```

credit cards do customers have on average

In [45]:
```
df['amount_of_credit_cards'].mean()
```

Out[45]:
```
1.4636666666666667
```

In [46]:
```
df.groupby('genderid')['amount_of_credit_cards'].mean().reset_index(name='avg_credi
```

Out[46]:

	genderid	avg_credit_cards
0	1	1.461022
1	2	1.466270

average credit_card_balance for each fee_structure type

In [49]:
```
df.groupby('fee_structure')['credit_card_balance'].mean()
```

Out[49]:
```
fee_structure
High    3164.768476
Low     3105.362135
Mid     3235.144532
Name: credit_card_balance, dtype: float64
```

correlation between checking_accounts, saving_accounts, and foreign_currency_account

In [52]:
```
df[['checking_accounts','saving_accounts','foreign_currency_account']].corr()
```

Out[52]:

	checking_accounts	saving_accounts	foreign_currency_account
checking_accounts	1.000000	0.459509	0.312651
saving_accounts	0.459509	1.000000	0.311465
foreign_currency_account	0.312651	0.311465	1.000000

In [ ]:
```

```

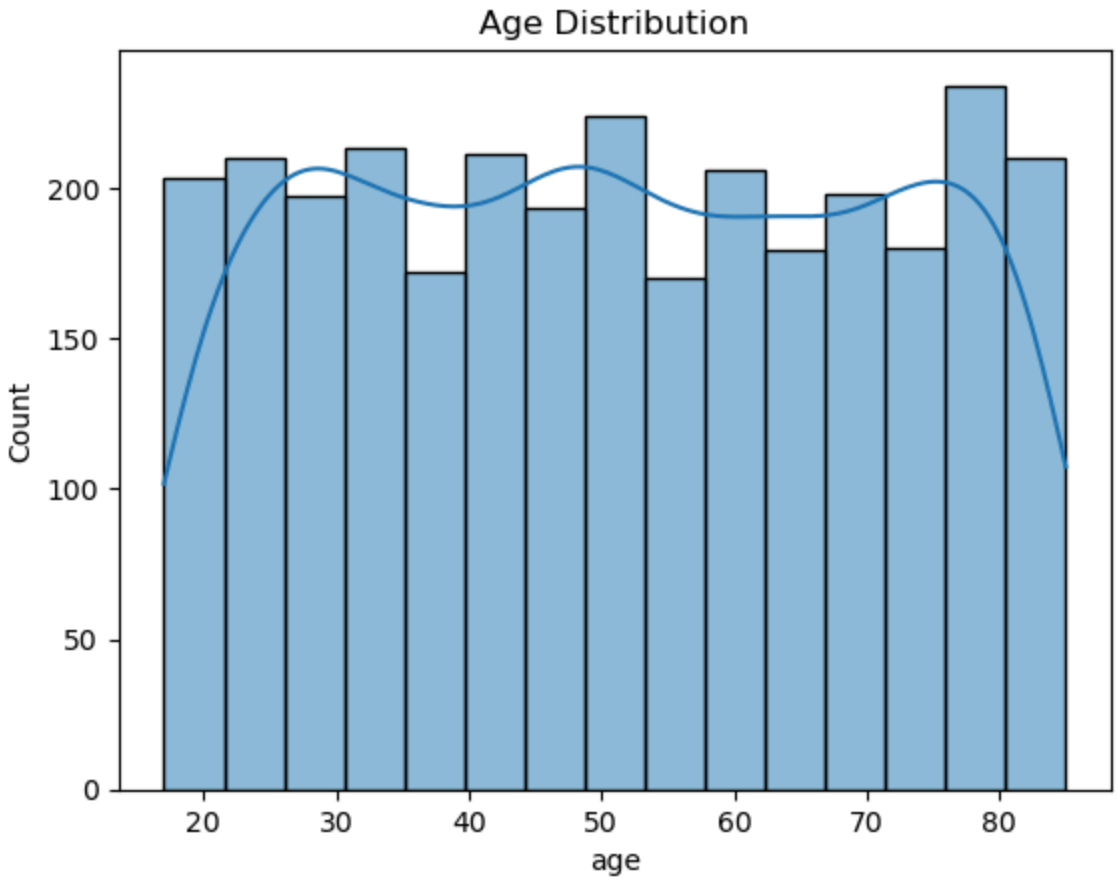
customers have the highest checking_accounts balance

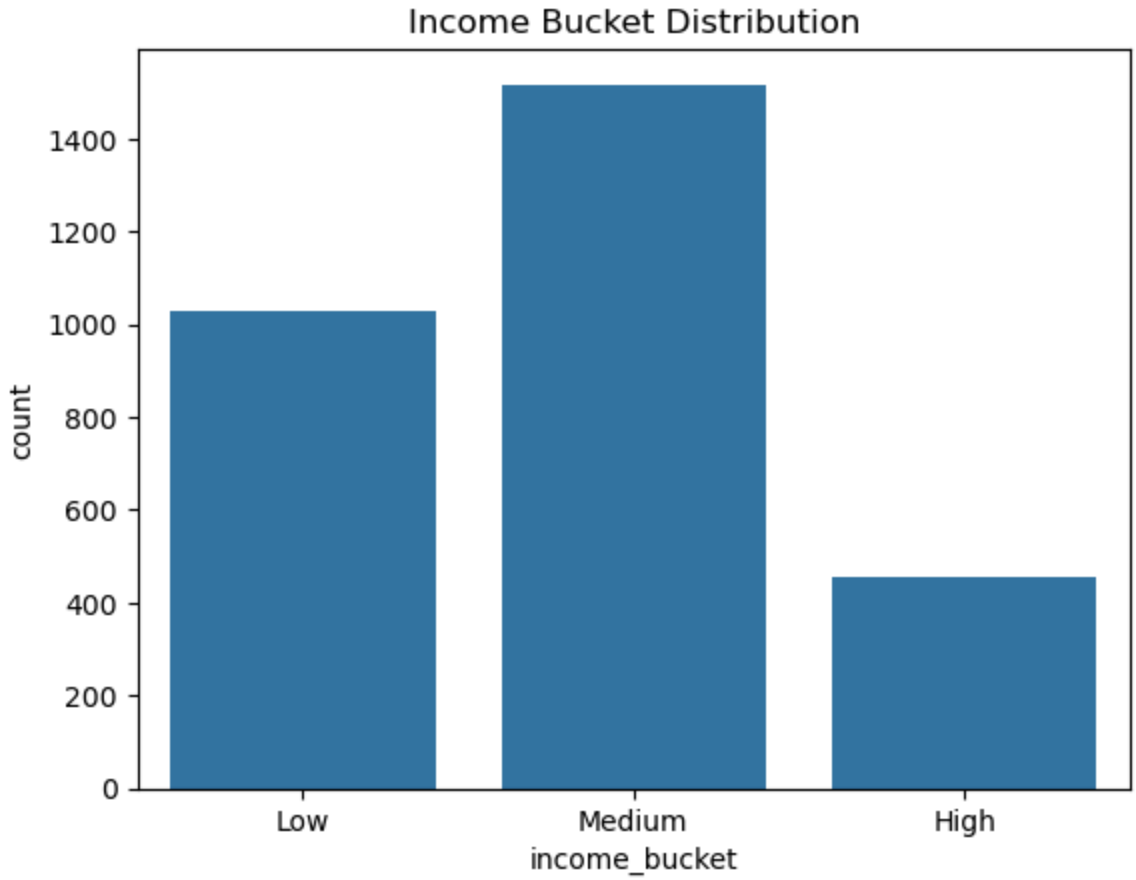
In [53]:
```
# Age distribution
sns.histplot(df['age'], bins=15, kde=True)
plt.title("Age Distribution")
```

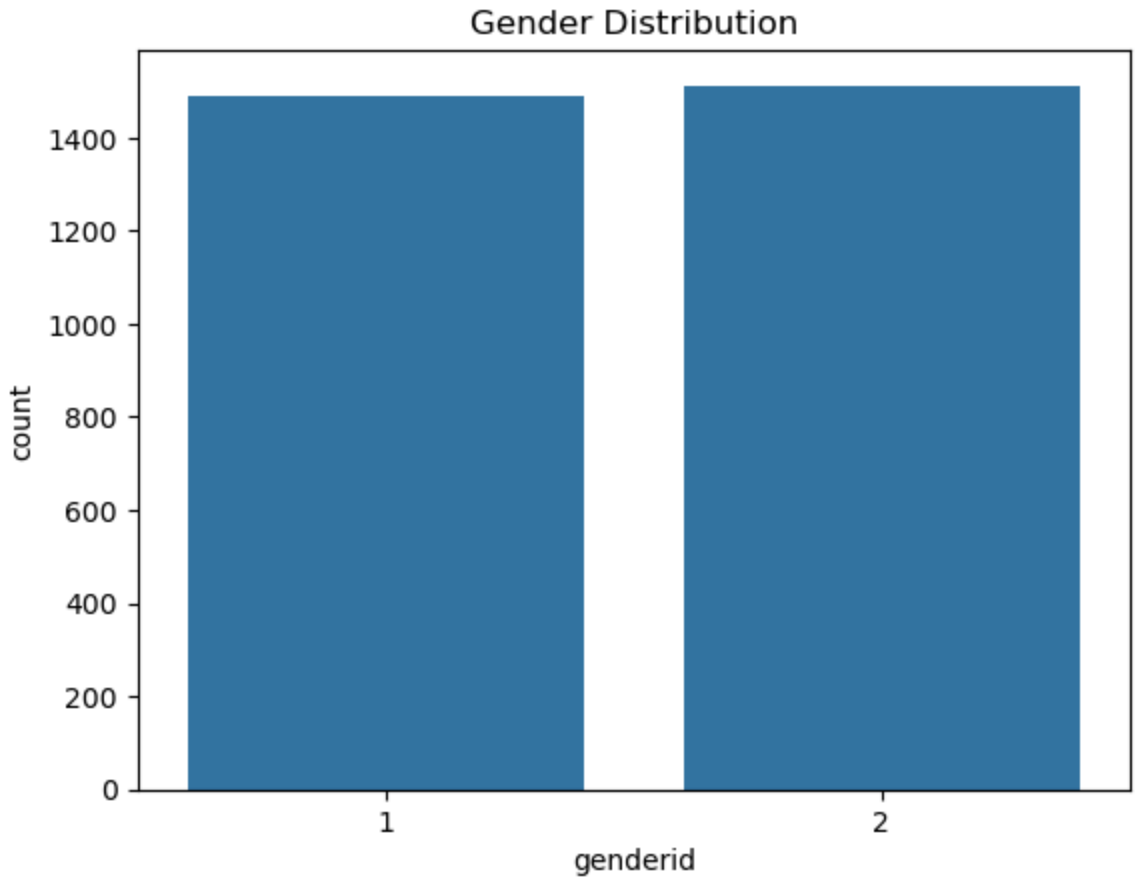
```
plt.show()

# Income bucket counts
sns.countplot(x='income_bucket', data=df)
plt.title("Income Bucket Distribution")
plt.show()

# Gender distribution
sns.countplot(x='genderid', data=df)
plt.title("Gender Distribution")
plt.show()
```

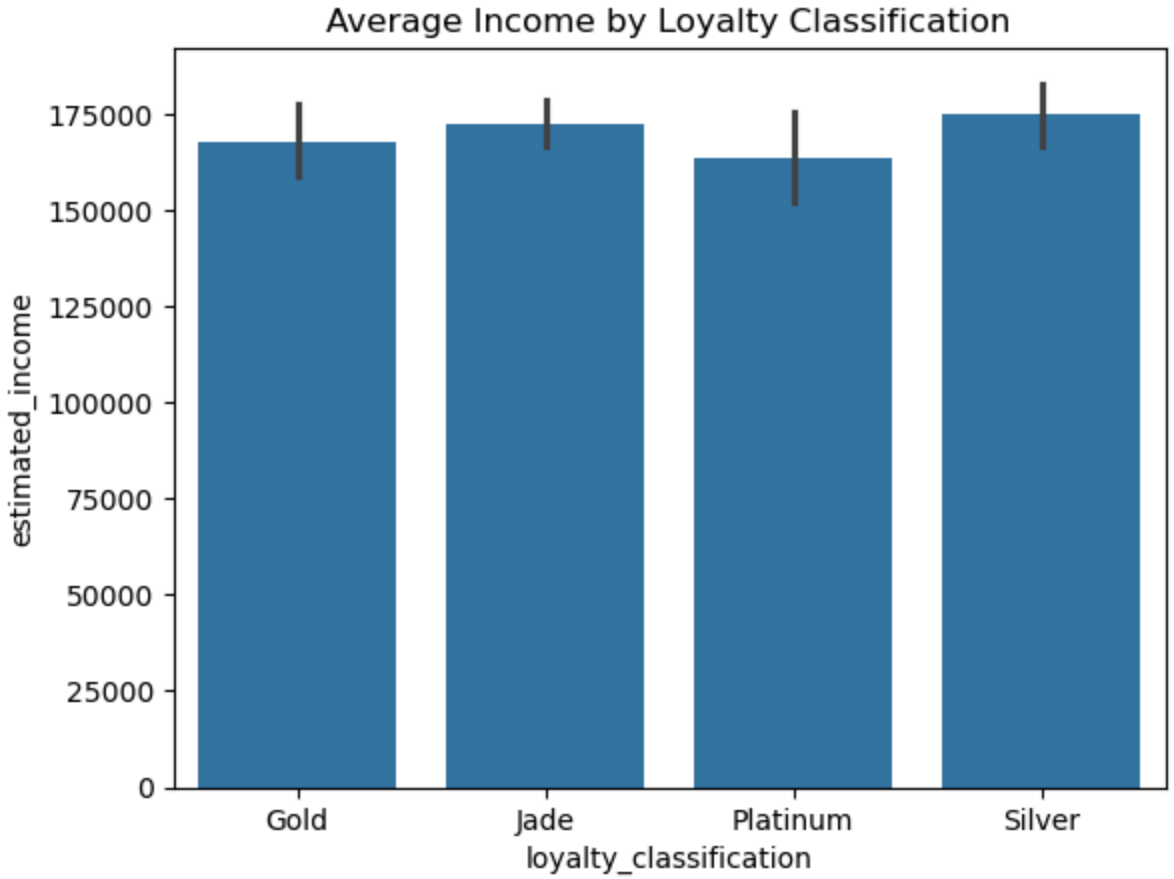


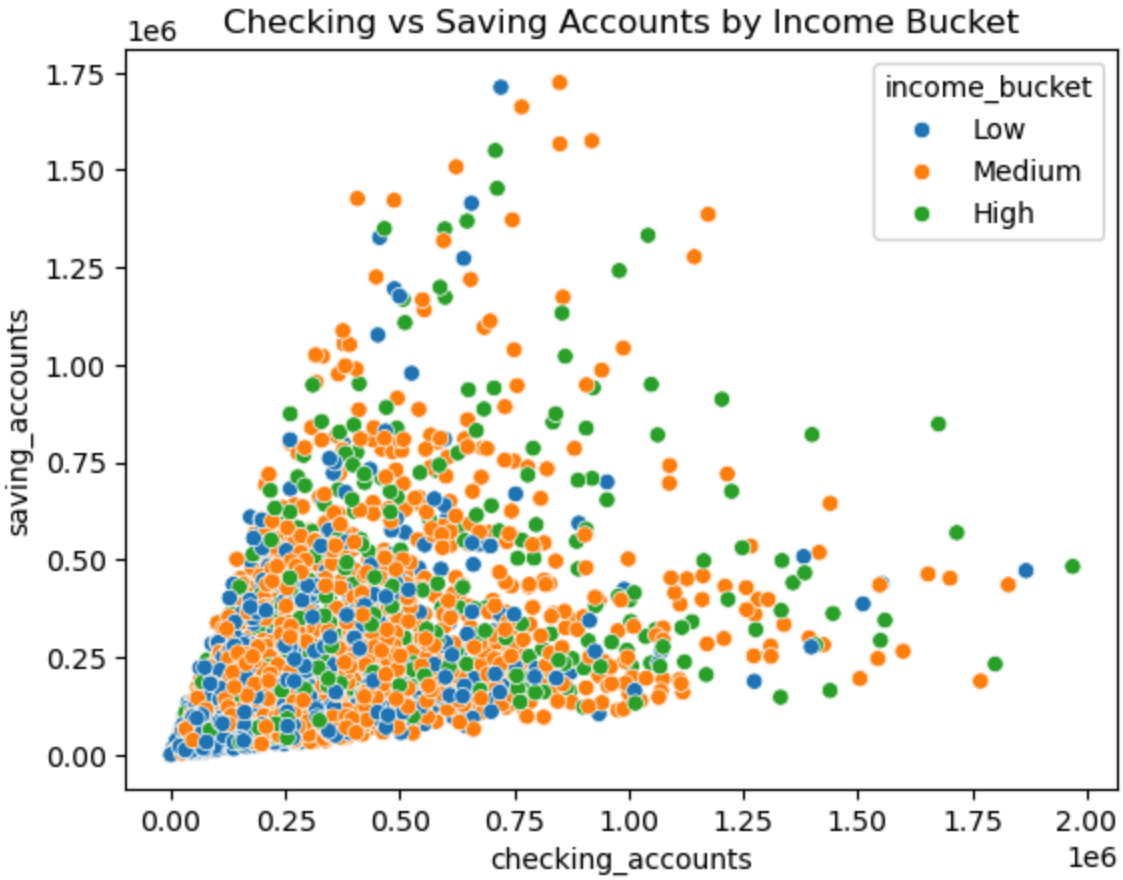

Income Bucket Distribution
count
0
200
400
600
800
1000
1200
1400
Low
Medium
High
income_bucket


Gender Distribution
count
0
200
400
600
800
1000
1200
1400
1
2
genderid

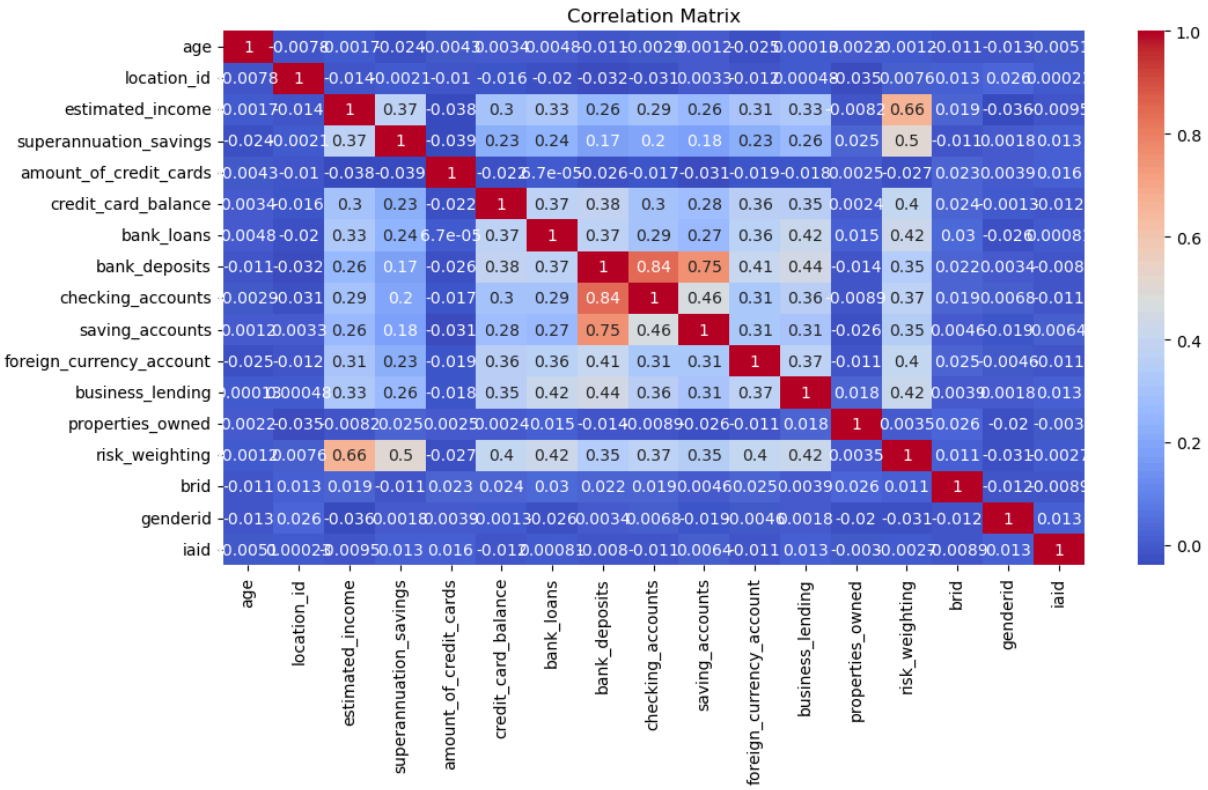
In [54]:
```
# Average income by loyalty classification
sns.barplot(x='loyalty_classification', y='estimated_income', data=df)
plt.title("Average Income by Loyalty Classification")
plt.show()

# Checking vs Saving Accounts
sns.scatterplot(x='checking_accounts', y='saving_accounts', hue='income_bucket', da
plt.title("Checking vs Saving Accounts by Income Bucket")
plt.show()
```

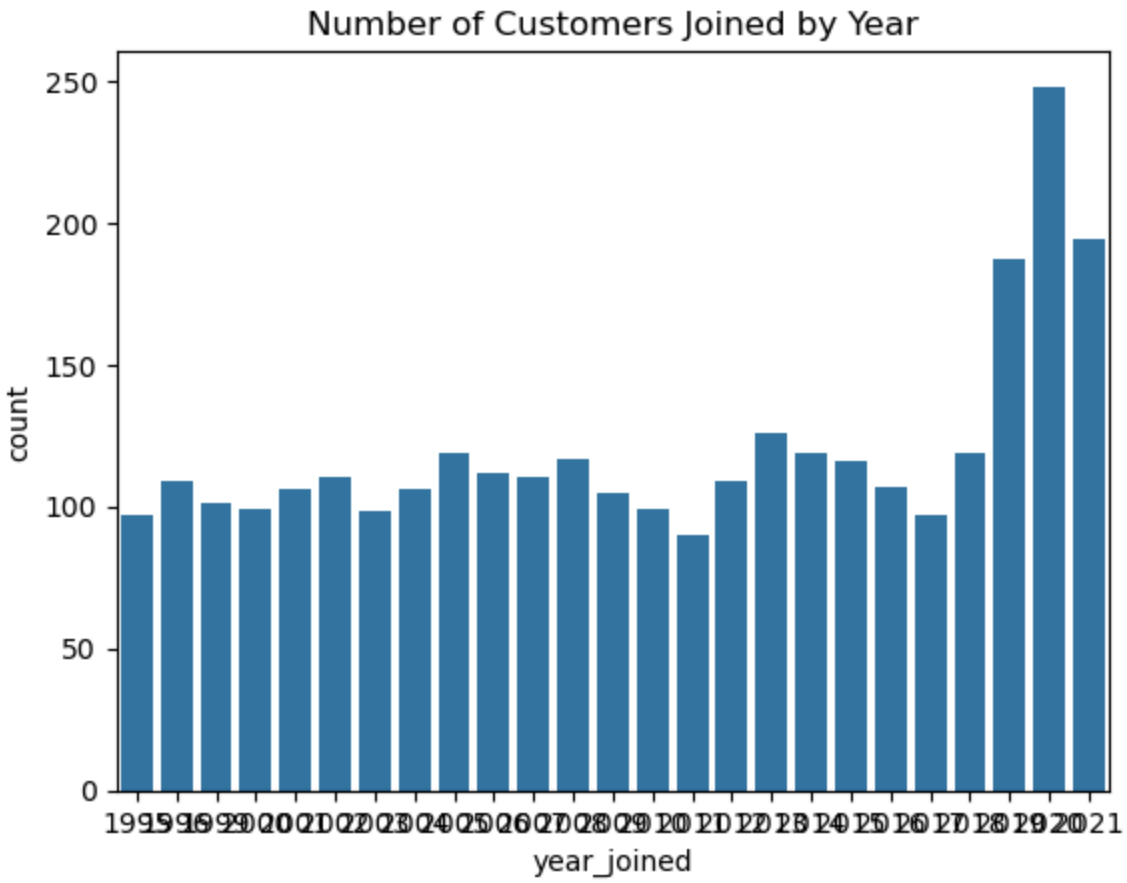




In [55]:
```
num_df = df.select_dtypes(include=['int64', 'float64'])
plt.figure(figsize=(12, 6))
sns.heatmap(num_df.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()
```



In [56]:
```
df['year_joined'] = df['joined_bank'].dt.year
sns.countplot(x='year_joined', data=df)
plt.title("Number of Customers Joined by Year")
plt.show()
```


Number of Customers Joined by Year
count
year_joined
0
50
100
150
200
250

In [ ]: