

✚ Financial Behavior Analysis

✚ Exploratory Data Analysis(EDA)

Importing libraries

```
%pip install ydata-profiling
```


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import missingno as msno
from ydata_profiling import ProfileReport
from scipy import stats
```

Load & Explore Data

```
df = pd.read_csv('financial_behavior_dataset.csv')
df.head()
```

<

```
df.shape
```

 (600, 12)

```
df.info()
```

```


➡ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 600 entries, 0 to 599
Data columns (total 12 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Customer_ID                          600 non-null    object
1   Month                                600 non-null    object
2   Age                                   600 non-null    int64
3   Gender                               600 non-null    object
4   Income                               600 non-null    int64
5   Total_Spending                       600 non-null    float64
6   Essential_Spending                   600 non-null    float64
7   Non_Essential_Spending               600 non-null    float64
8   Savings_Amount                       596 non-null    float64
9   Investments_Amount                   600 non-null    float64
10  Debt_Payments                        600 non-null    float64
11  Account_Balance                      600 non-null    float64
dtypes: float64(7), int64(2), object(3)
memory usage: 56.4+ KB

```

Clean Data

```
# Convert Month to datetime
df["Month"] = pd.to_datetime(df["Month"], format="%m-%Y", errors="coerce").dt.month
```

```
#Data Type Check
print(df.dtypes)
```

	Customer_ID	object
	Month	int32
	Age	int64
	Gender	object
	Income	int64
	Total_Spending	float64
	Essential_Spending	float64
	Non_Essential_Spending	float64
	Savings_Amount	float64
	Investments_Amount	float64
	Debt_Payments	float64
	Account_Balance	float64
	Savings_Rate	float64
	Investment_Rate	float64
	Debt_Ratio	float64
	Spending_Rate	float64
	Spending_Volatility	float64
	dtype:	object

```
df.isnull().sum()
```



0

Customer_ID

0

Month

0

Age

0

Gender

0

Income

0

Total_Spending

0

Essential_Spending

0

Non_Essential_Spending

0

Savings_Amount

4

Investments_Amount

0

Debt_Payments

0

Account_Balance

0

dtype: int64

```
#replaces null cell with 0
df['Savings_Amount'] = df['Savings_Amount'].fillna(0)
```

```
df['Savings_Amount'].isnull().sum()
```



```
np.int64(0)
```

```
# Duplicate rows
df.duplicated().sum()
```



```
np.int64(0)
```

Descriptive Statistics

df.describe()



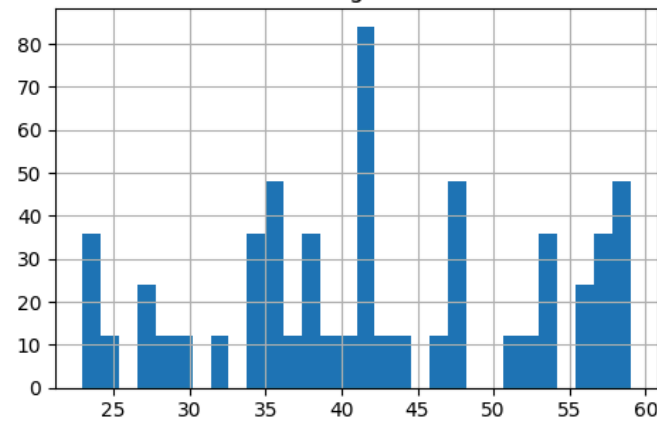
	Age	Income	Total_Spending	Essential_Spending	Non_Essential_Spending	Savings_Amount	Investments_Amount	Debt_Payme
count	600.000000	600.000000	600.000000	600.000000	600.000000	596.000000	600.000000	600.000000
mean	42.140000	16310.740000	14128.112100	6227.572500	2698.384683	1175.772047	1746.065883	2288.154000
std	10.591854	5965.844351	5841.683996	2308.639048	1794.914198	1935.423439	2354.777819	2904.956000
min	23.000000	6711.000000	3765.350000	1500.000000	371.340000	0.000000	0.000000	0.000000
25%	35.000000	11637.000000	9881.205000	4500.000000	1478.645000	0.000000	0.000000	0.000000
50%	41.000000	14939.500000	13428.510000	5500.000000	2195.000000	0.000000	588.950000	1300.000000
75%	52.000000	21071.000000	18555.615000	7500.000000	3580.502500	1646.100000	2379.795000	2500.000000
max	59.000000	28829.000000	28790.150000	11000.000000	11600.000000	8151.300000	8648.700000	10987.000000

```
# Analysis of numerical variables (such as spending and saving)
num_cols = ['Age', 'Income', 'Total_Spending', 'Essential_Spending',
            'Non_Essential_Spending', 'Savings_Amount',
            'Investments_Amount', 'Debt_Payments', 'Account_Balance']

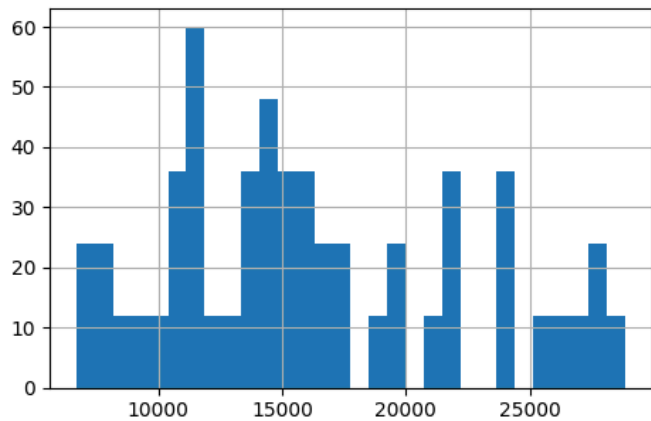
df[num_cols].hist(bins=30, figsize=(15, 10))
plt.tight_layout()
plt.show()
```



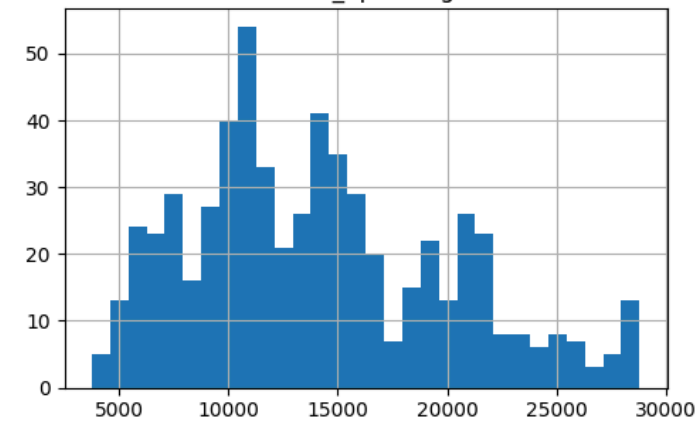
Age



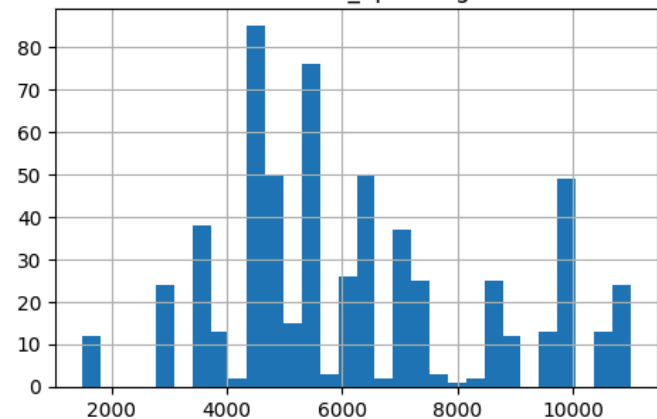
Income



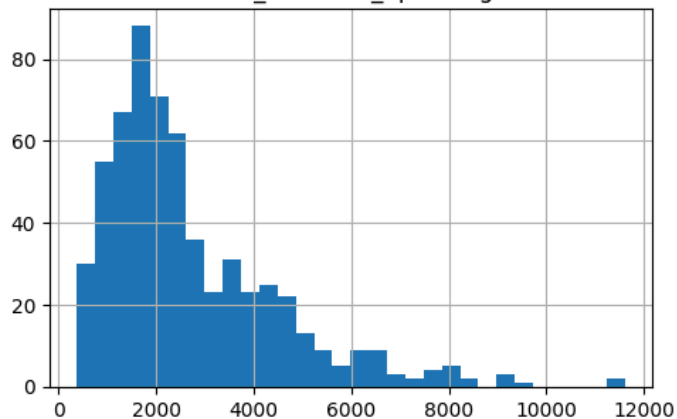
Total_Spending



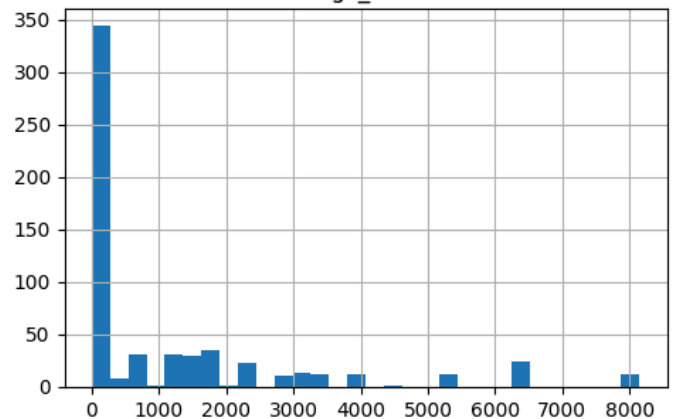
Essential_Spending



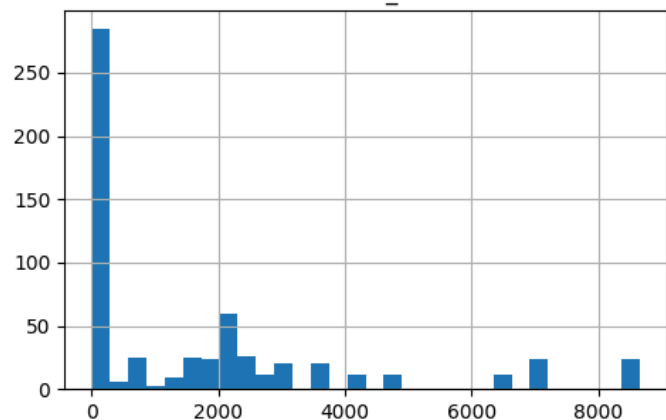
Non_Essential_Spending



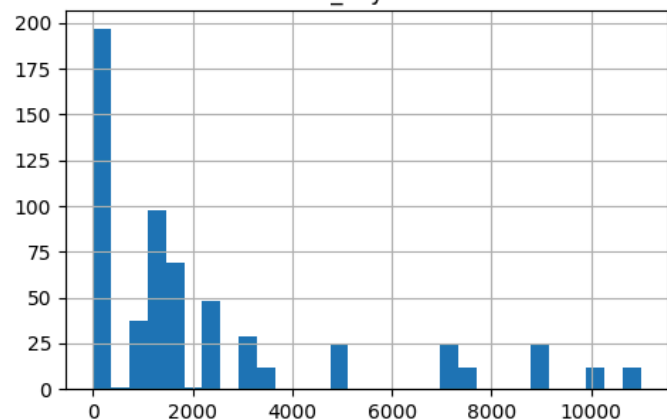
Savings_Amount



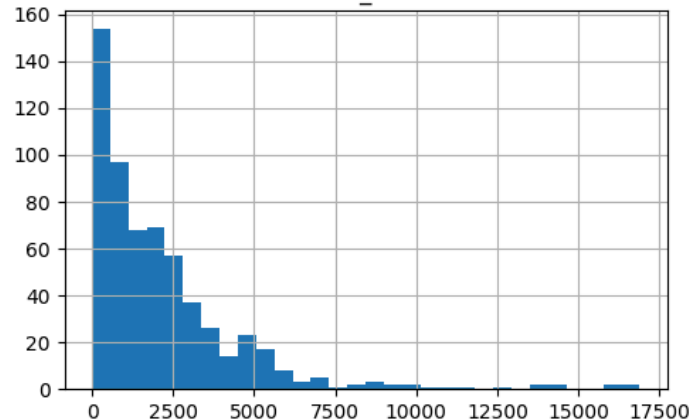
Investments_Amount



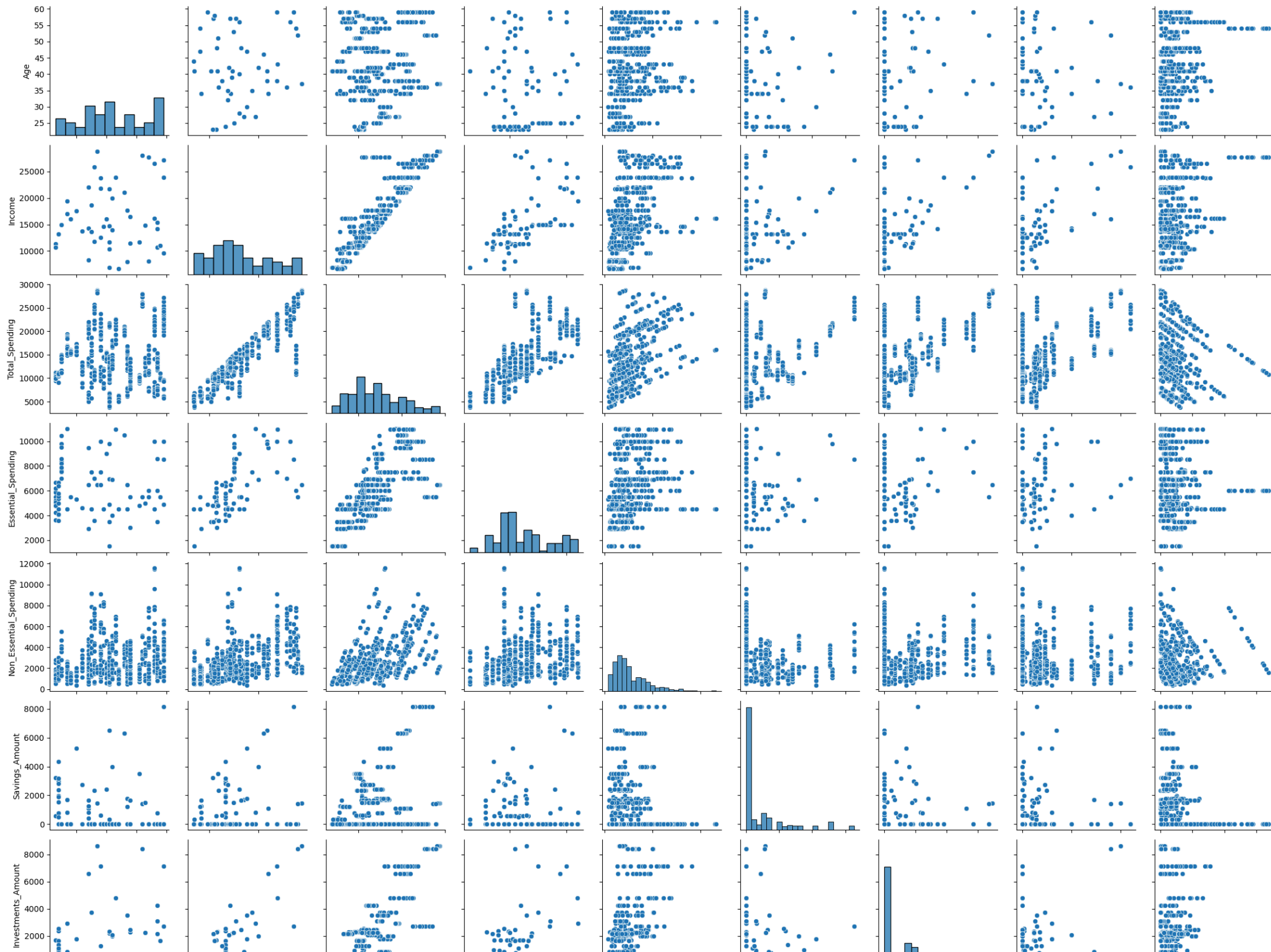
Debt_Payments

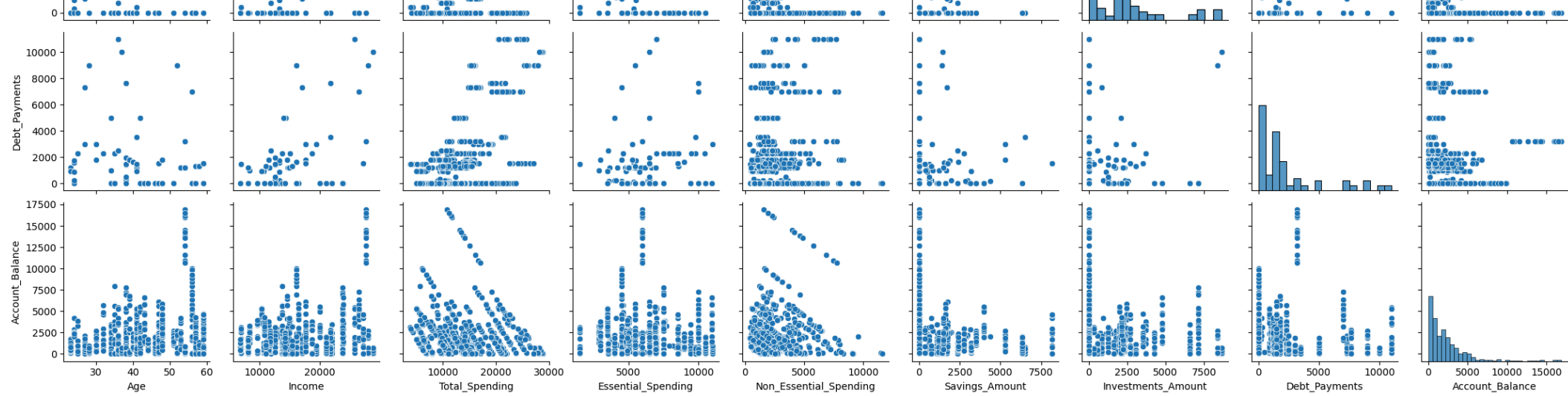


Account_Balance



```
#Relationships between Age,income,spending, saving, and debt
sns.pairplot(df[num_cols])
plt.show()
```

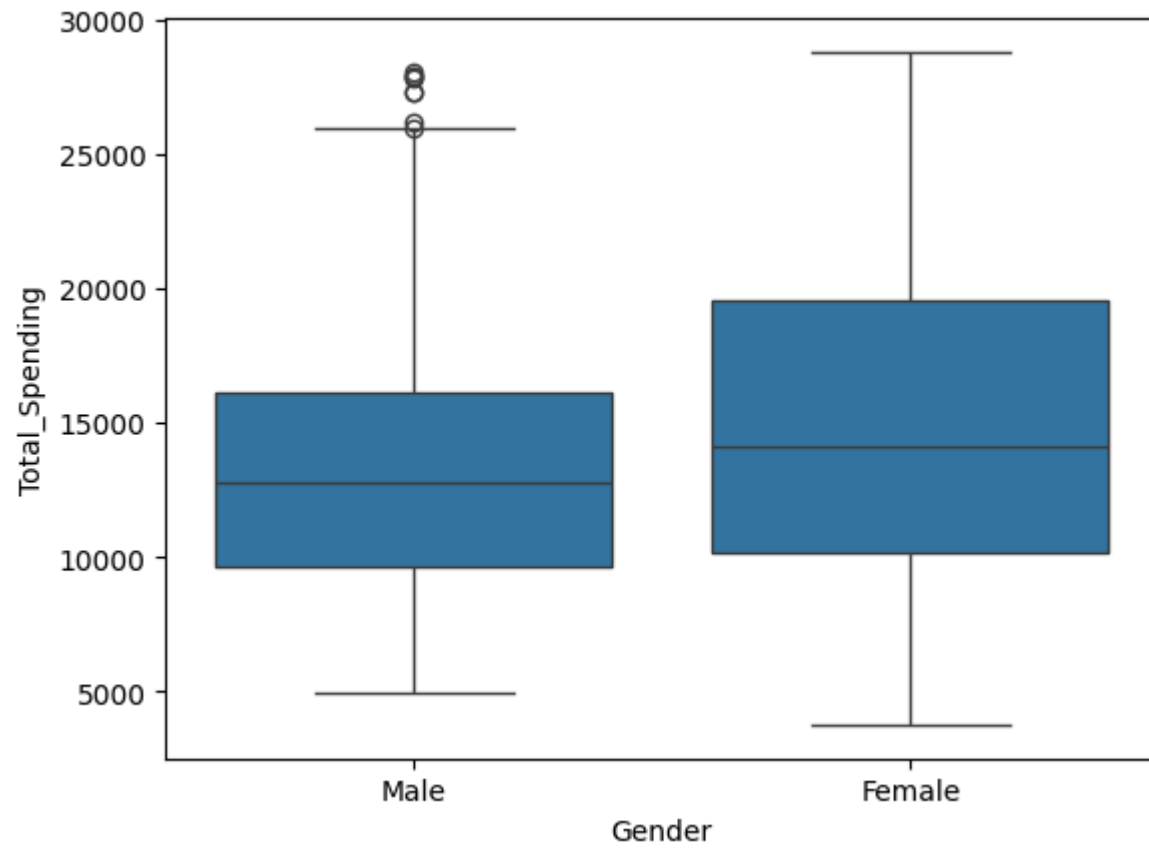





```
#Distribution by gender
sns.boxplot(x='Gender', y='Total_Spending', data=df)
```

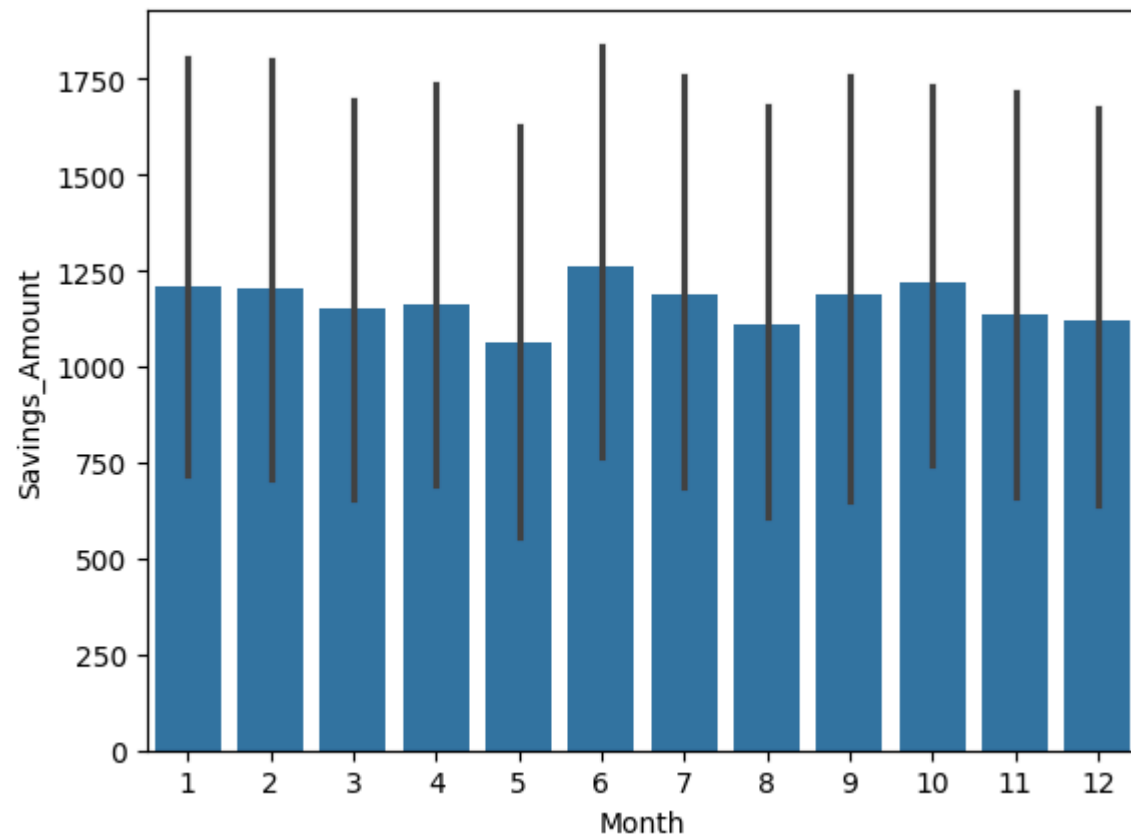


<Axes: xlabel='Gender', ylabel='Total_Spending'>



```
# Distribution by month
sns.barplot(x='Month', y='Savings_Amount', data=df)
```

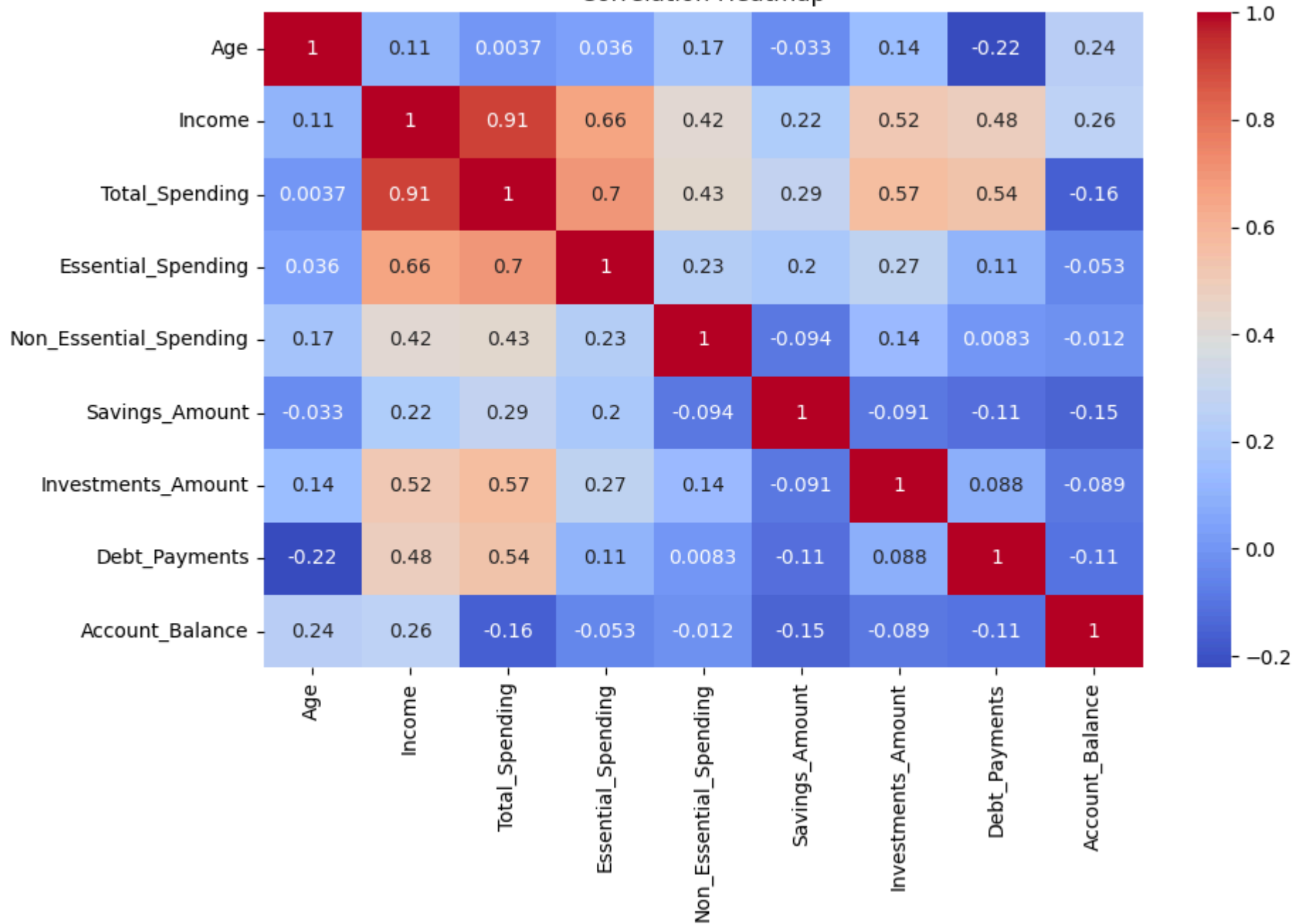
 <Axes: xlabel='Month', ylabel='Savings_Amount'>



```
# Check the correlation
plt.figure(figsize=(10, 6))
sns.heatmap(df[num_cols].corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```



Correlation Heatmap



Derivation of indicators

```
#Financial Volatility Index
# If Spending_Volatility is high → the client is "seasonal" or "random"
Spending_Volatility = df.groupby("Customer_ID")["Total_Spending"].std().reset_index()
Spending_Volatility.columns = ["Customer_ID", "Spending_Volatility"]
Spending_Volatility
```



	Customer_ID	Spending_Volatility
0	CUST1000	1127.963236
1	CUST1001	598.619274
2	CUST1002	2371.200345
3	CUST1003	1574.507687
4	CUST1004	1242.601960
5	CUST1005	860.532848
6	CUST1006	2410.825360
7	CUST1007	1756.175787
8	CUST1008	1689.360972
9	CUST1009	1778.260666
10	CUST1010	282.007318
11	CUST1011	575.437034
12	CUST1012	912.915278
13	CUST1013	688.980411
14	CUST1014	761.741130
15	CUST1015	588.769249
16	CUST1016	714.904341
17	CUST1017	660.715154
18	CUST1018	334.580235
19	CUST1019	1899.329536
20	CUST1020	1208.667994
21	CUST1021	744.547908
22	CUST1022	600.502111
23	CUST1023	1228.045735

24	CUST1024	825.109550
25	CUST1025	949.080498
26	CUST1026	320.006517
27	CUST1027	2478.318015
28	CUST1028	1463.164416
29	CUST1029	1252.259668
30	CUST1030	1685.652099
31	CUST1031	662.858109
32	CUST1032	1399.881352
33	CUST1033	1538.421199
34	CUST1034	1029.849982
35	CUST1035	1071.913887
36	CUST1036	686.757549
37	CUST1037	2170.406711
38	CUST1038	1250.524049
39	CUST1039	1799.330922
40	CUST1040	358.760962
41	CUST1041	276.239723
42	CUST1042	544.182754
43	CUST1043	656.806010
44	CUST1044	440.477496
45	CUST1045	484.072561
46	CUST1046	265.559355
47	CUST1047	3792.014206
48	CUST1048	1625.966066
49	CUST1049	818.888158


```
# Savings Rate
df["Savings_Rate"] = df["Savings_Amount"] / df["Income"]
df["Savings_Rate"]
```



	Savings_Rate
0	0.05
1	0.05
2	0.05
3	0.05
4	0.05
...	...
595	0.30
596	0.30
597	0.30
598	0.30
599	0.30

600 rows × 1 columns

dtype: float64

```
# Investment Rate
df["Investment_Rate"] = df["Investments_Amount"] / df["Income"]
df["Investment_Rate"]
```



Investment_Rate	
0	0.15
1	0.15
2	0.15
3	0.15
4	0.15
...	...
595	0.10
596	0.10
597	0.10
598	0.10
599	0.10

600 rows × 1 columns

dtype: float64

```
# Debt to Income Ratio
df["Debt_Ratio"] = df["Debt_Payments"] / df["Income"]
df["Debt_Ratio"]
```



	Debt_Ratio
0	0.105848
1	0.105848
2	0.105848
3	0.105848
4	0.105848
...	...
595	0.055206
596	0.055206
597	0.055206
598	0.055206
599	0.055206

600 rows × 1 columns

dtype: float64

```
# Total Spending Rate
df["Spending_Rate"] = df["Total_Spending"] / df["Income"]
df["Spending_Rate"]
```



	Spending_Rate
0	0.943491
1	0.913638
2	0.846186
3	0.964283
4	0.992500
...	...
595	0.937963
596	0.907838
597	0.970682
598	0.998064
599	0.890185

600 rows × 1 columns

dtype: float64

```
# Merge the Spending_Volatility DataFrame with the main DataFrame only if Spending_Volatility column doesn't exist
if "Spending_Volatility" not in df.columns:
    df = pd.merge(df, Spending_Volatility, on="Customer_ID", how="left")

# Calculate initial customer metrics
customer_metrics = df.groupby("Customer_ID")[["Savings_Rate", "Investment_Rate", "Debt_Ratio", "Spending_Volatility", "Spending_Rate", "Total_
display(customer_metrics)
```



	Customer_ID	Savings_Rate	Investment_Rate	Debt_Ratio	Spending_Volatility	Spending_Rate	Total_Spending
0	CUST1000	0.000000	0.150000	0.081627	1127.963236	0.885581	13018.931667
1	CUST1001	0.050000	0.150000	0.105848	598.619274	0.921293	10444.695833
2	CUST1002	0.000000	0.000000	0.000000	2371.200345	0.749652	10252.994167
3	CUST1003	0.000000	0.200000	0.123179	1574.507687	0.897389	16756.049167
4	CUST1004	0.000000	0.200000	0.156767	1242.601960	0.802353	9212.619167
5	CUST1005	0.000000	0.200000	0.103119	860.532848	0.871997	10147.425833
6	CUST1006	0.000000	0.300000	0.000000	2410.825360	0.827022	19628.550833
7	CUST1007	0.000000	0.200000	0.000000	1756.175787	0.842499	20160.155833
8	CUST1008	0.075000	0.133333	0.084938	1689.360972	0.826367	14593.646667
9	CUST1009	0.133333	0.000000	0.167565	1778.260666	0.800001	10980.808333
10	CUST1010	0.000000	0.000000	0.148736	282.007318	0.952005	7680.776667
11	CUST1011	0.000000	0.000000	0.348918	575.437034	0.950438	13619.780000
12	CUST1012	0.025000	0.150000	0.084219	912.915278	0.952613	14704.537500
13	CUST1013	0.150000	0.000000	0.000000	688.980411	0.886819	7092.777500
14	CUST1014	0.300000	0.100000	0.130682	761.741130	0.909676	16010.298333
15	CUST1015	0.125000	0.000000	0.102765	588.769249	0.926150	14870.264167
16	CUST1016	0.000000	0.200000	0.000000	714.904341	0.708169	7596.533333
17	CUST1017	0.100000	0.050000	0.429412	660.715154	0.950402	16156.833333
18	CUST1018	0.010312	0.150000	0.154679	334.580235	0.978605	18980.040000
19	CUST1019	0.000000	0.000000	0.425952	1899.329536	0.907155	23399.150833
20	CUST1020	0.050000	0.300000	0.000000	1208.667994	0.937590	20625.102500
21	CUST1021	0.300000	0.000000	0.000000	744.547908	0.957958	20185.130833
22	CUST1022	0.159030	0.074870	0.033473	600.502111	0.814242	10775.683333
23	CUST1023	0.025000	0.025000	0.210297	1228.045735	0.781862	5390.936667

24	CUST1024	0.037500	0.066667	0.042263	825.109550	0.868593	10866.961667
25	CUST1025	0.000000	0.300000	0.000000	949.080498	0.906942	12864.968333
26	CUST1026	0.300000	0.000000	0.161097	320.006517	0.984968	21399.425000
27	CUST1027	0.000000	0.000000	0.123178	2478.318015	0.802193	11722.447500
28	CUST1028	0.000000	0.000000	0.153333	1463.164416	0.910238	13653.569167
29	CUST1029	0.100000	0.150000	0.080651	1252.259668	0.857594	12760.143333
30	CUST1030	0.000000	0.000000	0.264410	1685.652099	0.828434	21931.958333
31	CUST1031	0.116667	0.000000	0.120289	662.858109	0.734881	6103.185000
32	CUST1032	0.200000	0.100000	0.000000	1399.881352	0.829427	16546.248333
33	CUST1033	0.000000	0.000000	0.000000	1538.421199	0.817164	7881.550833
34	CUST1034	0.000000	0.000000	0.350998	1029.849982	0.919210	20034.172500
35	CUST1035	0.016667	0.300000	0.321211	1071.913887	0.953820	26725.080000
36	CUST1036	0.200000	0.050000	0.212242	686.757549	0.925803	10905.032500
37	CUST1037	0.000000	0.000000	0.115699	2170.406711	0.495127	13694.231667
38	CUST1038	0.100000	0.150000	0.000000	1250.524049	0.752129	12380.803333
39	CUST1039	0.000000	0.300000	0.000000	1799.330922	0.913817	21773.516667
40	CUST1040	0.300000	0.000000	0.084112	358.760962	0.946518	10127.745000
41	CUST1041	0.000000	0.000000	0.559667	276.239723	0.970134	15600.720833
42	CUST1042	0.300000	0.000000	0.000000	544.182754	0.845055	9856.716667
43	CUST1043	0.000000	0.150000	0.358809	656.806010	0.944470	13161.185000
44	CUST1044	0.000000	0.150000	0.118053	440.477496	0.920689	10138.625000
45	CUST1045	0.000000	0.000000	0.000000	484.072561	0.847331	5686.435833
46	CUST1046	0.050000	0.300000	0.346873	265.559355	0.988023	28483.711667
47	CUST1047	0.000000	0.000000	0.000000	3792.014206	0.577700	9305.589167
48	CUST1048	0.300000	0.100000	0.055206	1625.966066	0.915783	24882.733333
49	CUST1049	0.000000	0.000000	0.000000	218.000000	0.500000	5000.000000

49	COST1049	0.000000	0.000000	0.087015	646.832153	0.544825	5635.125833
----	----------	----------	----------	----------	------------	----------	-------------

Financial Behavior Type and Financial Goal Type

```
def classify_behavior(row):
    if row["Savings_Rate"] > 0.1 and row["Investment_Rate"] > 0.1:
        return "disciplined"
    elif row["Spending_Volatility"] > 2000:
        return "volatile"
    elif row["Savings_Rate"] < 0.05 and row["Investment_Rate"] < 0.05:
        return "spender"
    else:
        return "unclassified"
customer_metrics["Behavior_Type"] = customer_metrics.apply(classify_behavior, axis=1)
```

```
def classify_goal(row):
    if row["Savings_Rate"] > 0.10 and row["Investment_Rate"] < 0.05:
        return "Saver"
    elif row["Investment_Rate"] > 0.1:
        return "Investor"
    else:
        return "Spender"
customer_metrics["Financial_Goal"] = customer_metrics.apply(classify_goal, axis=1)
```

```
display(customer_metrics)
```




	Customer_ID	Savings_Rate	Investment_Rate	Debt_Ratio	Spending_Volatility	Spending_Rate	Total_Spending	Behavior_Type	Financial
0	CUST1000	0.000000	0.150000	0.081627	1127.963236	0.885581	13018.931667	unclassified	In
1	CUST1001	0.050000	0.150000	0.105848	598.619274	0.921293	10444.695833	unclassified	In
2	CUST1002	0.000000	0.000000	0.000000	2371.200345	0.749652	10252.994167	volatile	Sp
3	CUST1003	0.000000	0.200000	0.123179	1574.507687	0.897389	16756.049167	unclassified	In
4	CUST1004	0.000000	0.200000	0.156767	1242.601960	0.802353	9212.619167	unclassified	In
5	CUST1005	0.000000	0.200000	0.103119	860.532848	0.871997	10147.425833	unclassified	In
6	CUST1006	0.000000	0.300000	0.000000	2410.825360	0.827022	19628.550833	volatile	In
7	CUST1007	0.000000	0.200000	0.000000	1756.175787	0.842499	20160.155833	unclassified	In
8	CUST1008	0.075000	0.133333	0.084938	1689.360972	0.826367	14593.646667	unclassified	In
9	CUST1009	0.133333	0.000000	0.167565	1778.260666	0.800001	10980.808333	unclassified	In
10	CUST1010	0.000000	0.000000	0.148736	282.007318	0.952005	7680.776667	spender	Sp
11	CUST1011	0.000000	0.000000	0.348918	575.437034	0.950438	13619.780000	spender	Sp
12	CUST1012	0.025000	0.150000	0.084219	912.915278	0.952613	14704.537500	unclassified	In
13	CUST1013	0.150000	0.000000	0.000000	688.980411	0.886819	7092.777500	unclassified	In
14	CUST1014	0.300000	0.100000	0.130682	761.741130	0.909676	16010.298333	disciplined	In
15	CUST1015	0.125000	0.000000	0.102765	588.769249	0.926150	14870.264167	unclassified	In
16	CUST1016	0.000000	0.200000	0.000000	714.904341	0.708169	7596.533333	unclassified	In
17	CUST1017	0.100000	0.050000	0.429412	660.715154	0.950402	16156.833333	unclassified	Sp
18	CUST1018	0.010312	0.150000	0.154679	334.580235	0.978605	18980.040000	unclassified	In
19	CUST1019	0.000000	0.000000	0.425952	1899.329536	0.907155	23399.150833	spender	Sp
20	CUST1020	0.050000	0.300000	0.000000	1208.667994	0.937590	20625.102500	unclassified	In
21	CUST1021	0.300000	0.000000	0.000000	744.547908	0.957958	20185.130833	unclassified	In
22	CUST1022	0.159030	0.074870	0.033473	600.502111	0.814242	10775.683333	unclassified	Sp
23	CUST1023	0.025000	0.025000	0.210297	1228.045735	0.781862	5390.936667	spender	Sp

24	CUST1024	0.037500	0.066667	0.042263	825.109550	0.868593	10866.961667	unclassified	Sp
25	CUST1025	0.000000	0.300000	0.000000	949.080498	0.906942	12864.968333	unclassified	In
26	CUST1026	0.300000	0.000000	0.161097	320.006517	0.984968	21399.425000	unclassified	
27	CUST1027	0.000000	0.000000	0.123178	2478.318015	0.802193	11722.447500	volatile	Sp
28	CUST1028	0.000000	0.000000	0.153333	1463.164416	0.910238	13653.569167	spender	Sp
29	CUST1029	0.100000	0.150000	0.080651	1252.259668	0.857594	12760.143333	disciplined	In
30	CUST1030	0.000000	0.000000	0.264410	1685.652099	0.828434	21931.958333	spender	Sp
31	CUST1031	0.116667	0.000000	0.120289	662.858109	0.734881	6103.185000	unclassified	
32	CUST1032	0.200000	0.100000	0.000000	1399.881352	0.829427	16546.248333	disciplined	In
33	CUST1033	0.000000	0.000000	0.000000	1538.421199	0.817164	7881.550833	spender	Sp
34	CUST1034	0.000000	0.000000	0.350998	1029.849982	0.919210	20034.172500	spender	Sp
35	CUST1035	0.016667	0.300000	0.321211	1071.913887	0.953820	26725.080000	unclassified	In
36	CUST1036	0.200000	0.050000	0.212242	686.757549	0.925803	10905.032500	unclassified	Sp
37	CUST1037	0.000000	0.000000	0.115699	2170.406711	0.495127	13694.231667	volatile	Sp
38	CUST1038	0.100000	0.150000	0.000000	1250.524049	0.752129	12380.803333	unclassified	In
39	CUST1039	0.000000	0.300000	0.000000	1799.330922	0.913817	21773.516667	unclassified	In
40	CUST1040	0.300000	0.000000	0.084112	358.760962	0.946518	10127.745000	unclassified	
41	CUST1041	0.000000	0.000000	0.559667	276.239723	0.970134	15600.720833	spender	Sp
42	CUST1042	0.300000	0.000000	0.000000	544.182754	0.845055	9856.716667	unclassified	
43	CUST1043	0.000000	0.150000	0.358809	656.806010	0.944470	13161.185000	unclassified	In
44	CUST1044	0.000000	0.150000	0.118053	440.477496	0.920689	10138.625000	unclassified	In
45	CUST1045	0.000000	0.000000	0.000000	484.072561	0.847331	5686.435833	spender	Sp
46	CUST1046	0.050000	0.300000	0.346873	265.559355	0.988023	28483.711667	unclassified	In
47	CUST1047	0.000000	0.000000	0.000000	3792.014206	0.577700	9305.589167	volatile	Sp
48	CUST1048	0.300000	0.100000	0.055206	1625.966066	0.915783	24882.733333	unclassified	Sp
49	CUST1049	0.000000	0.000000	0.000000	218.000000	0.500000	5000.000000	unclassified	Sp

49 COST1049 0.000000 0.000000 0.087015 646.832153 0.544825 5635.125833 spender Sp

```
plt.figure(figsize=(8, 5))
sns.countplot(x='Behavior_Type', data=customer_metrics, order=customer_metrics["Behavior_Type"].value_counts().index)
plt.title("Client Distribution by Financial Behavior")
plt.xlabel("Financial behavior ")
plt.ylabel("Number of clients")
plt.xticks(rotation=45)
```