```
import pandas as pd
import numpy as np
```

```
df = pd.read_csv("loan_sanction_test.csv")
df
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	1.0	Urban
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	1.0	Urban
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	1.0	Urban
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	360.0	NaN	Urban
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	360.0	1.0	Urban
...	...	...	...	...	...	...	...	...	...	...	...	...
362	LP002971	Male	Yes	3+	Not Graduate	Yes	4009	1777	113.0	360.0	1.0	Urban
363	LP002975	Male	Yes	0	Graduate	No	4158	709	115.0	360.0	1.0	Urban
364	LP002980	Male	No	0	Graduate	No	3250	1993	126.0	360.0	NaN	Semiurban
365	LP002986	Male	Yes	0	Graduate	No	5000	2393	158.0	360.0	1.0	Rural
366	LP002989	Male	No	0	Graduate	Yes	9200	0	98.0	180.0	1.0	Rural

367 rows × 12 columns

```
#Display the first few rows of the dataset
df.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	1.0	Urban
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	1.0	Urban
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	1.0	Urban
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	360.0	NaN	Urban
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	360.0	1.0	Urban

```
#Check for missing values
missing_values = df.isnull().sum()
print("Missing Values:")
print(missing_values)
```

```
Missing Values:
Loan_ID               0
Gender               11
Married               0
Dependents           10
Education             0
Self_Employed        23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      6
Credit_History       29
Property_Area         0
dtype: int64
```

```
#Handle missing values
df_cleaned = df.dropna()
```

```
#Summarize basic statistics for numeric columns
numeric_summary = df.describe()
print("\nBasic Statistics for Numeric Columns:")
print(numeric_summary)
```

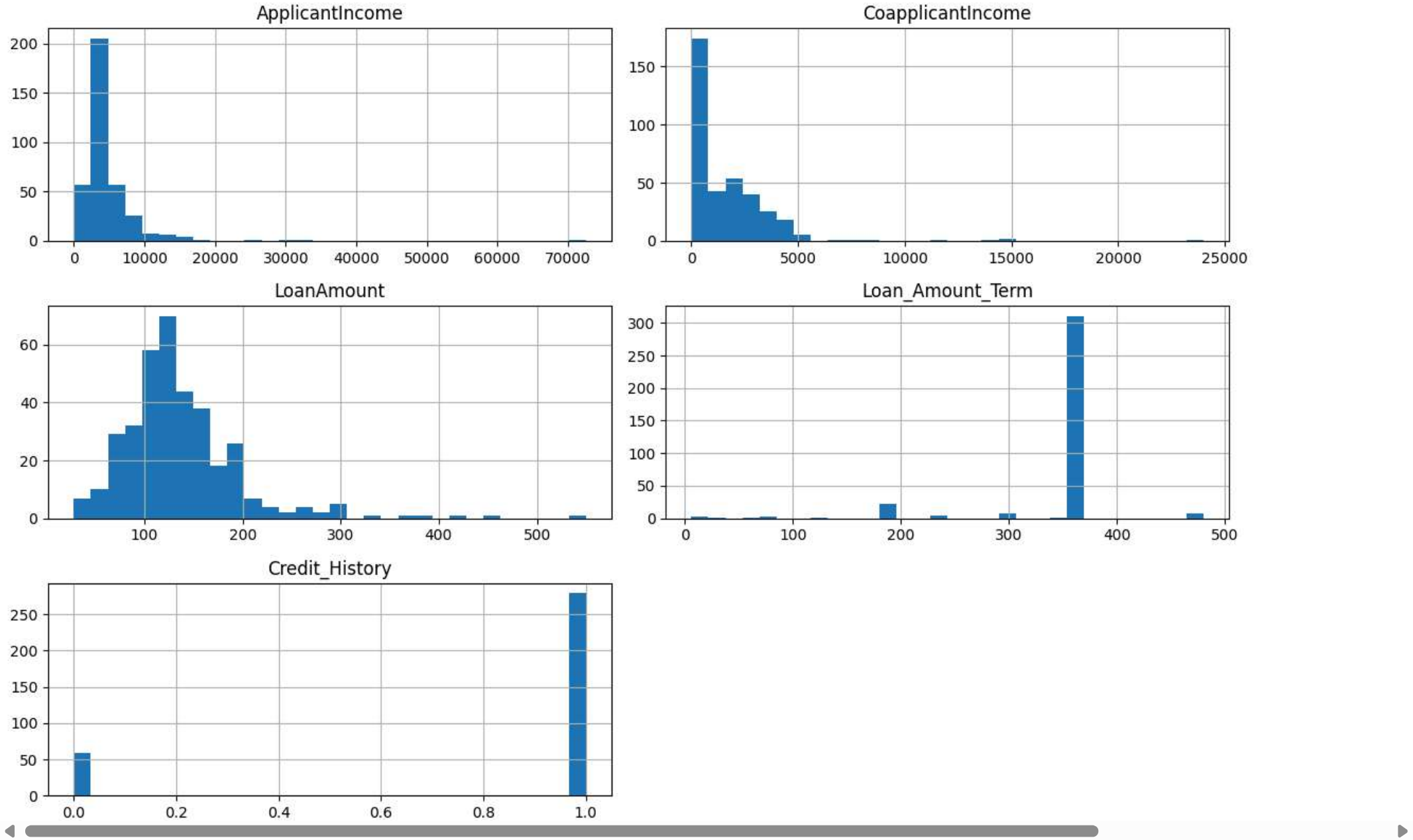
```
Basic Statistics for Numeric Columns:
       ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
count       367.000000         367.000000  362.000000        361.000000
mean       4805.599455        1569.577657  136.132597        342.537396
std        4910.685399        2334.232099   61.366652         65.156643
min           0.000000           0.000000   28.000000          6.000000
25%        2864.000000           0.000000  100.250000        360.000000
50%        3786.000000        1025.000000  125.000000        360.000000
75%        5060.000000        2430.500000  158.000000        360.000000
max       72529.000000       24000.000000  550.000000        480.000000

       Credit_History
count      338.000000
mean         0.825444
std          0.380150
min          0.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          1.000000
```

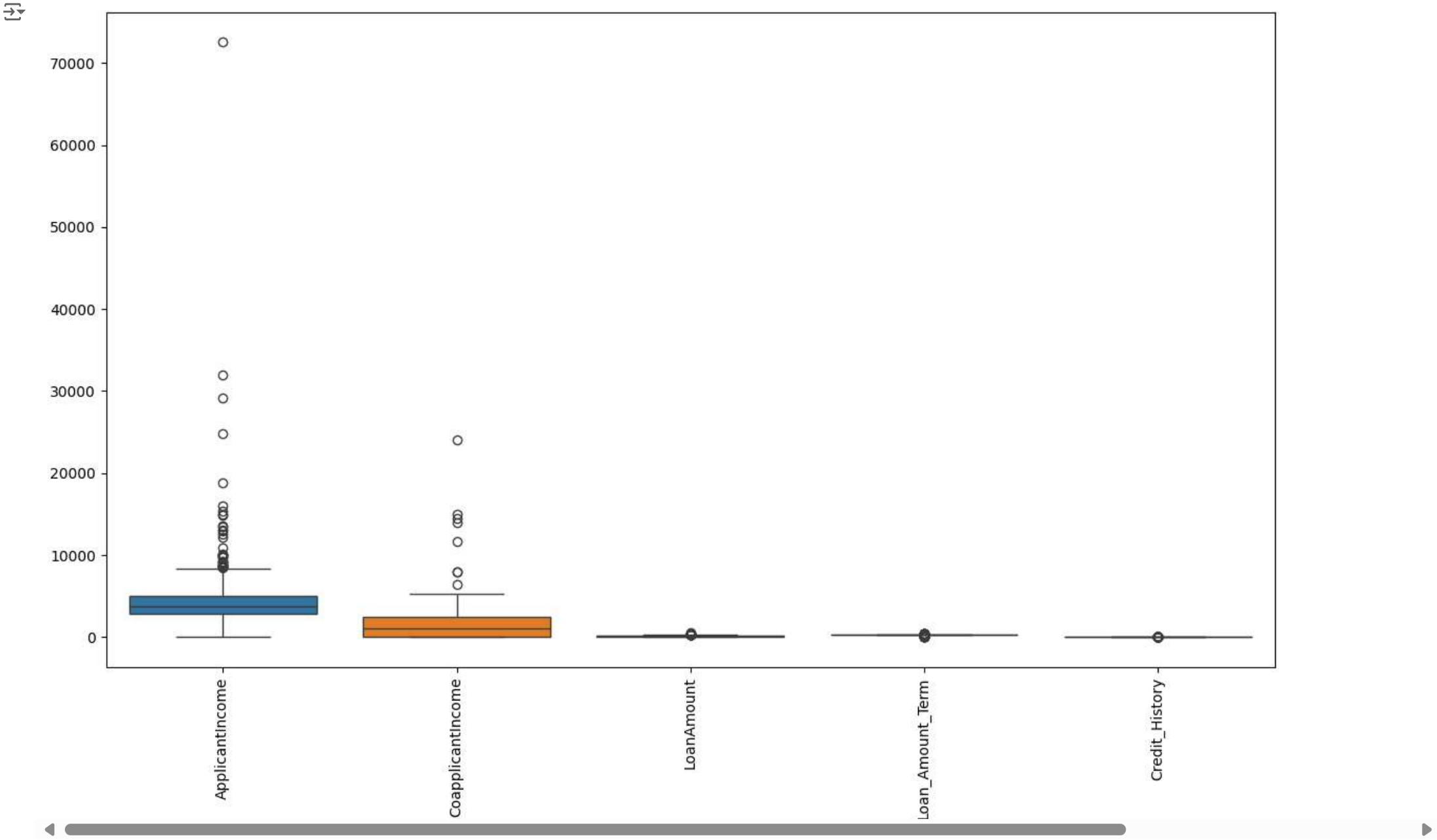
```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Plot histograms for all numeric columns
df.hist(bins=30, figsize=(12, 8))
```

```
plt.tight_layout()
plt.show()
```

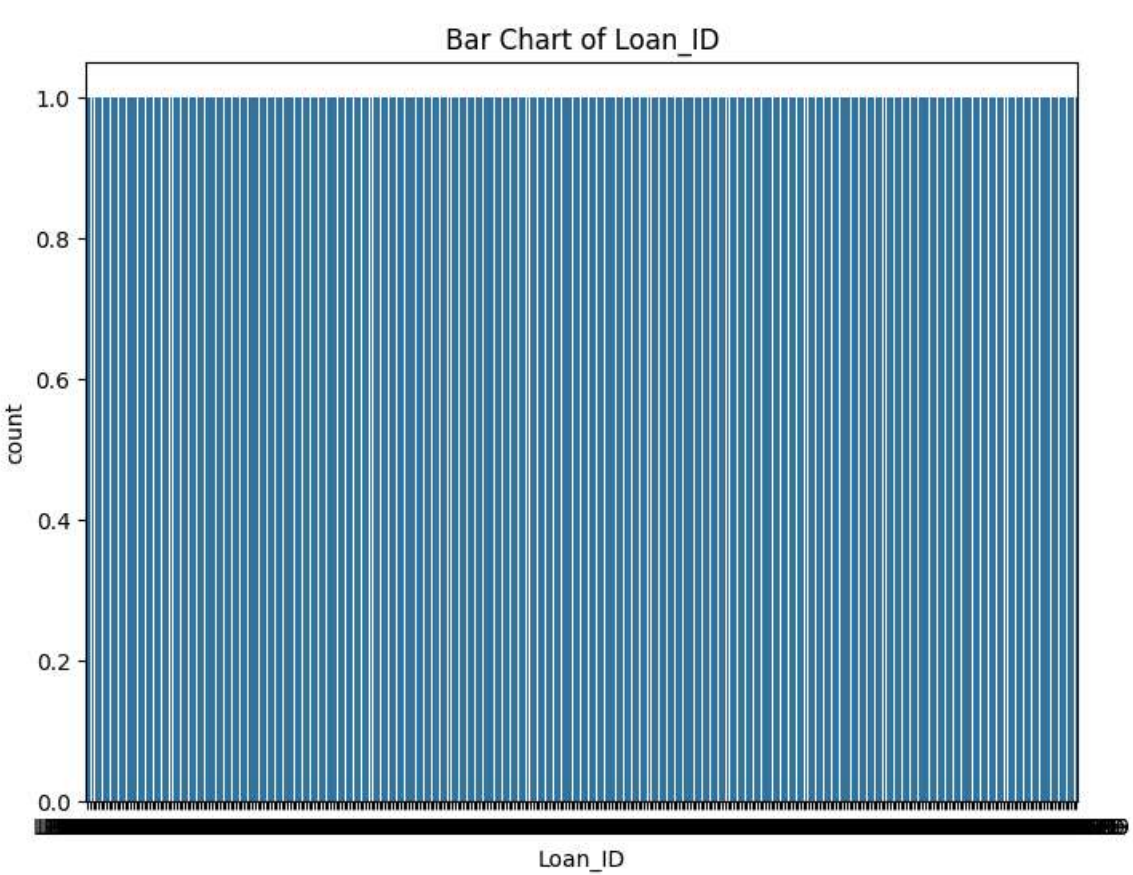


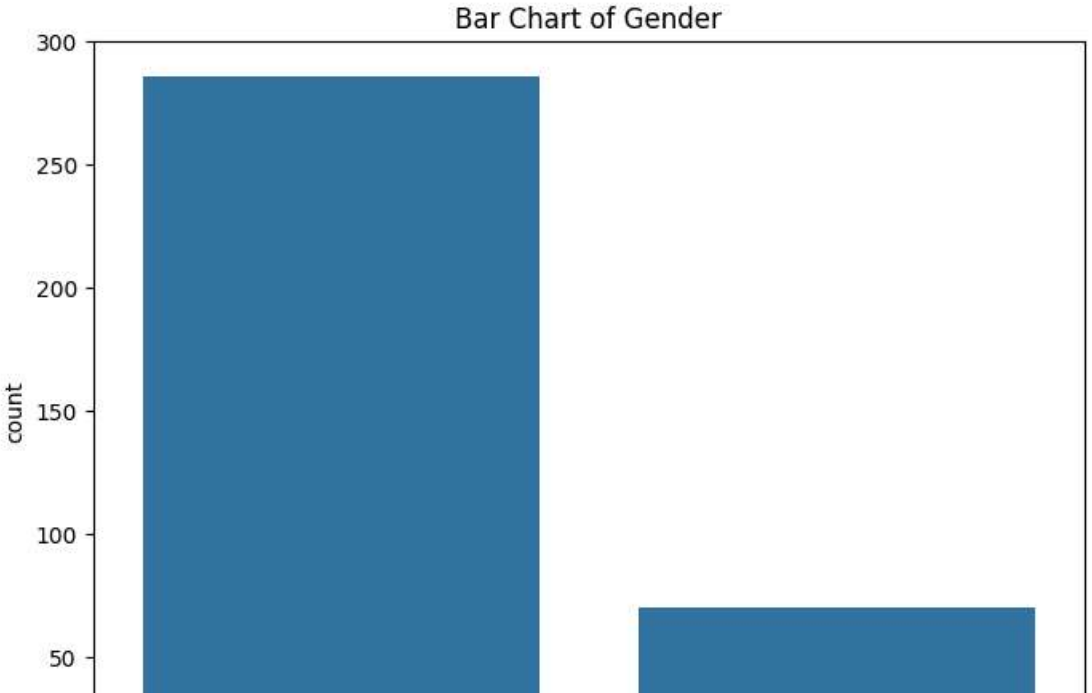
```
# Plot box plots for all numeric columns
plt.figure(figsize=(12, 8))
sns.boxplot(data=df)
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

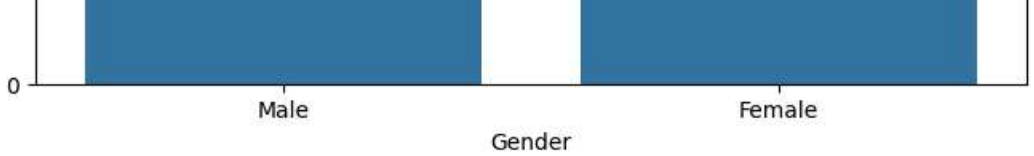


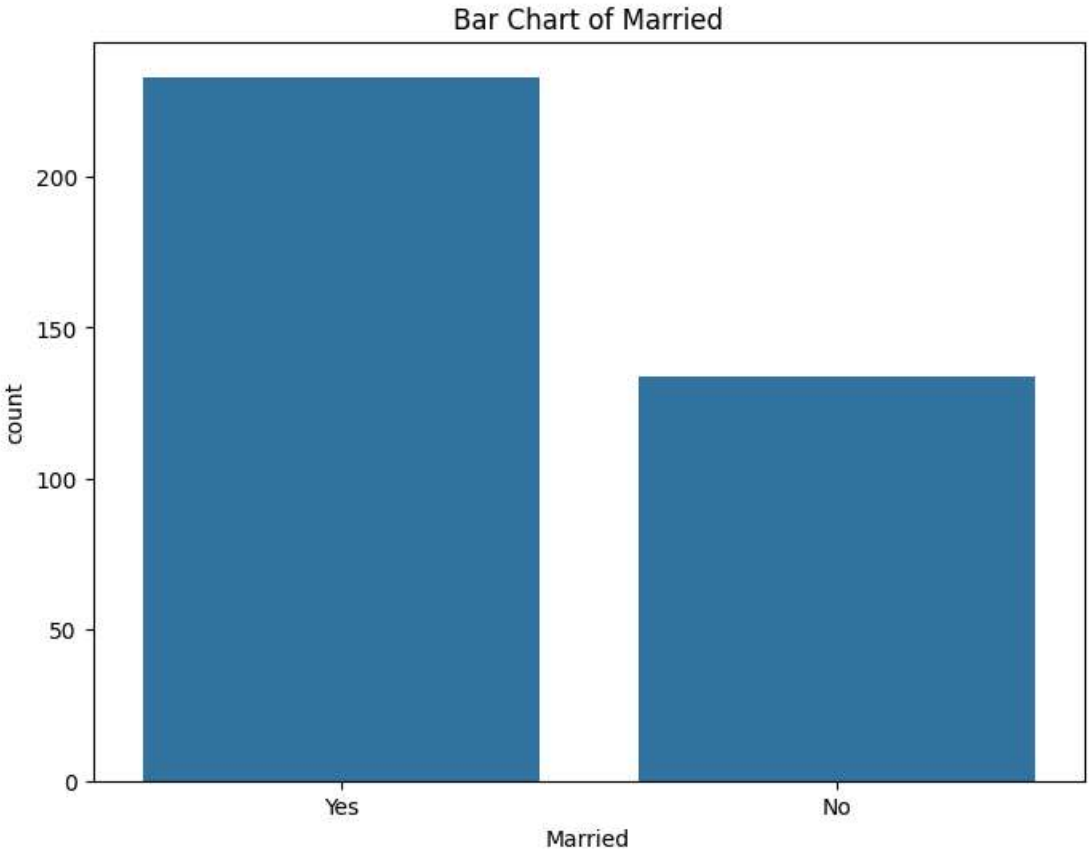

```
# Plot bar charts for categorical variables
categorical_columns = df.select_dtypes(include=['object']).columns
```

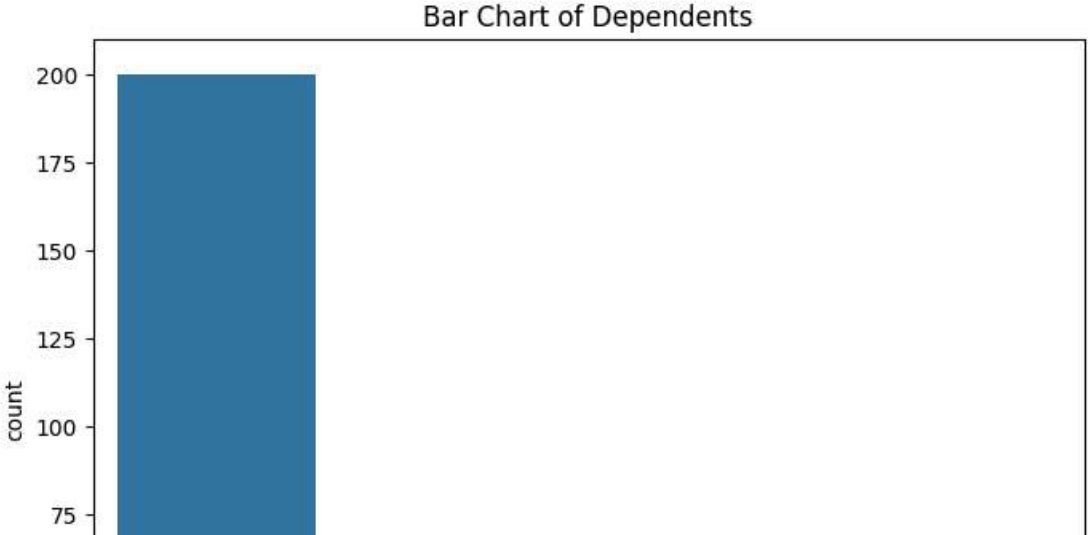
```
for col in categorical_columns:
    plt.figure(figsize=(8, 6))
    sns.countplot(x=df[col])
    plt.title(f'Bar Chart of {col}')
    plt.show()
```

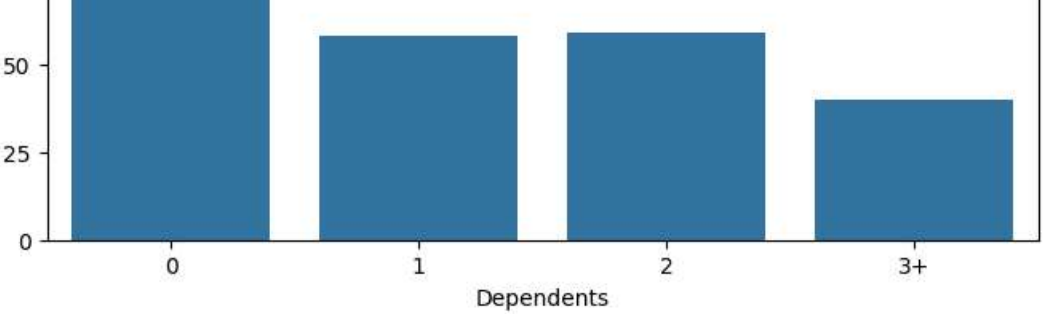

Bar Chart of Loan_ID
1.0
0.8
0.6
0.4
0.2
0.0
count
Loan_ID

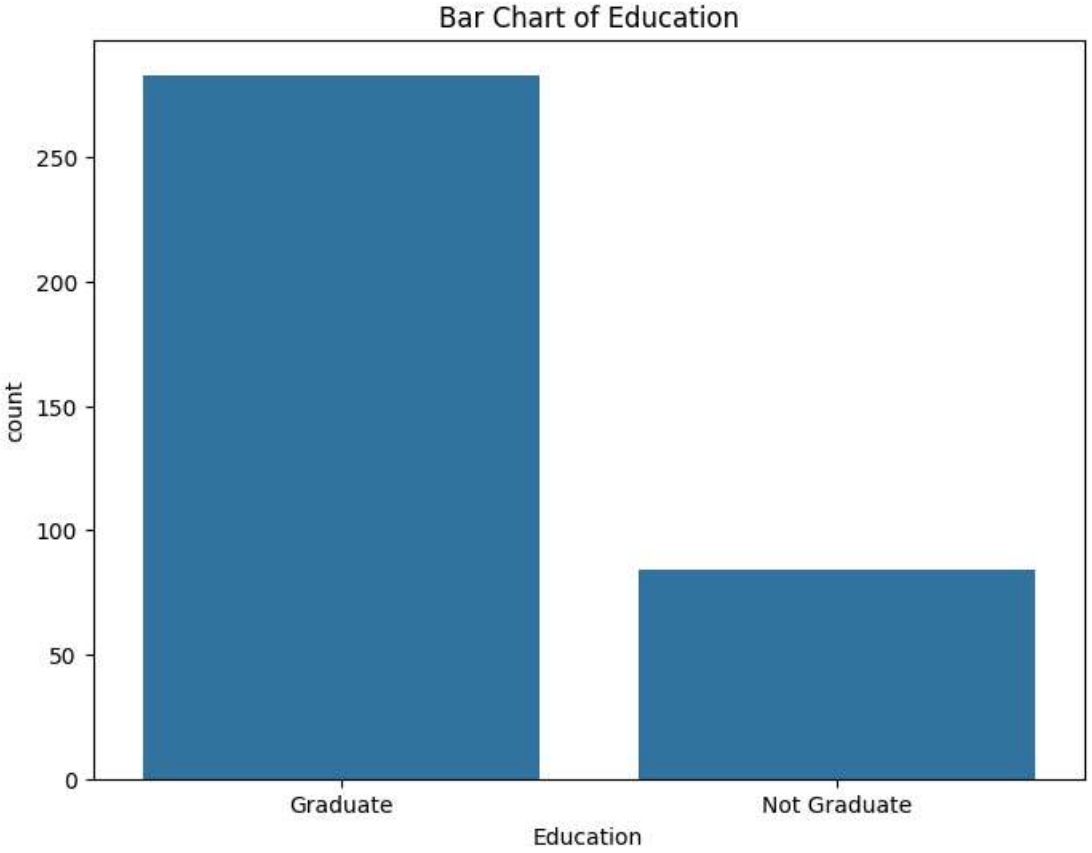

Bar Chart of Gender
300
250
200
150
100
50
count


0
Male
Female
Gender

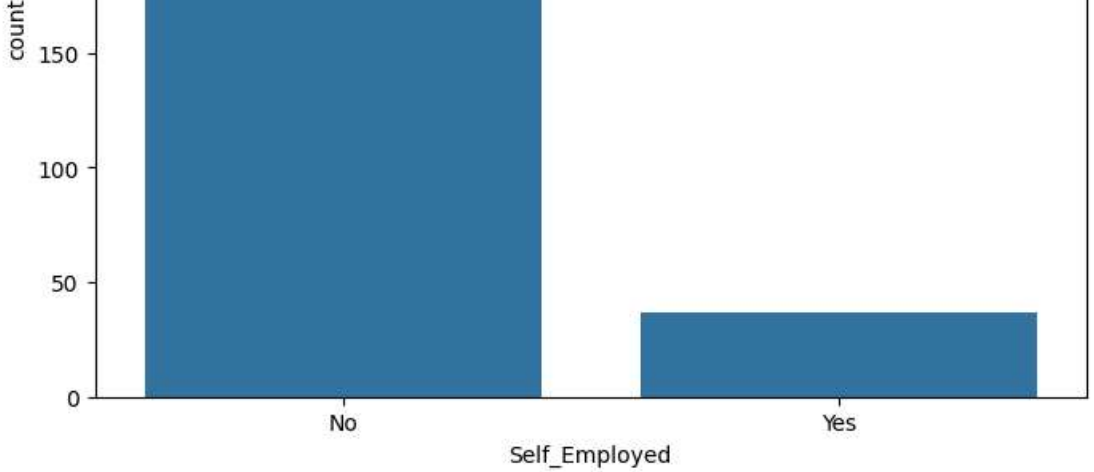

Bar Chart of Married
200
150
count
100
50
0
Yes
No
Married

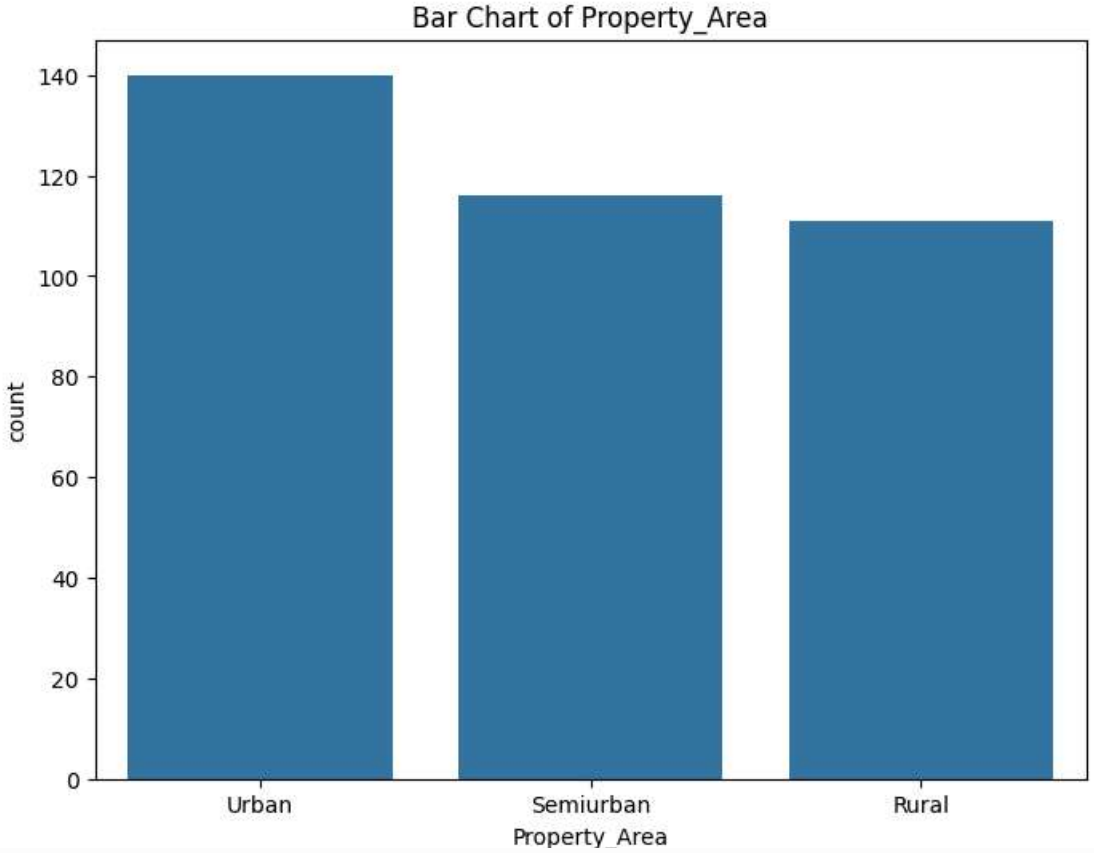

Bar Chart of Dependents
200
175
150
125
count
100
75


50
25
0
0
1
2
3+
Dependents

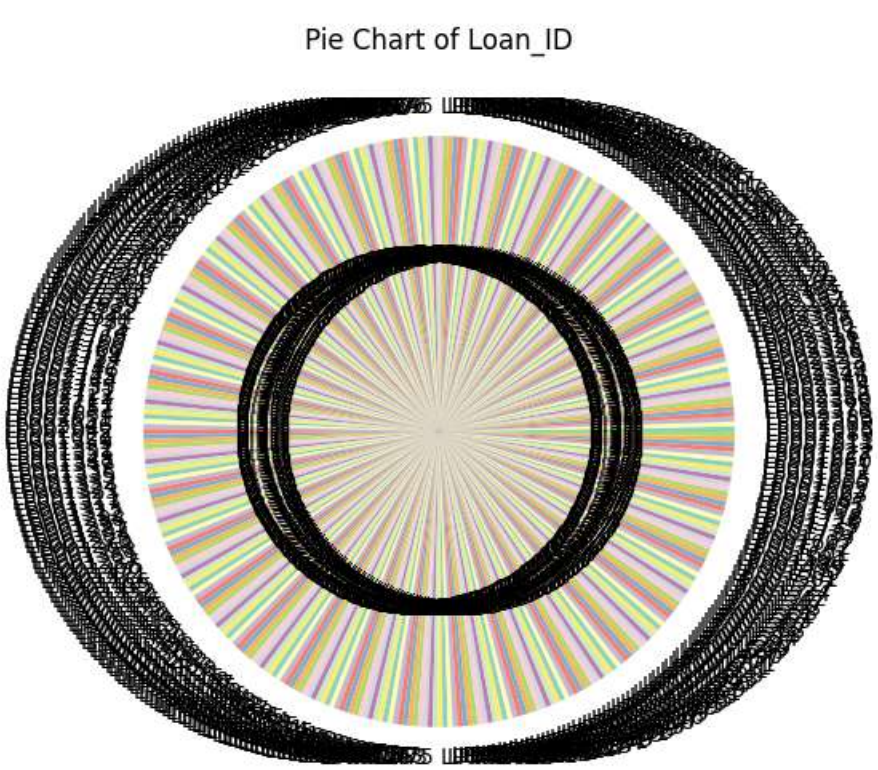

Bar Chart of Education
250
200
150
100
50
0
count
Graduate
Not Graduate
Education

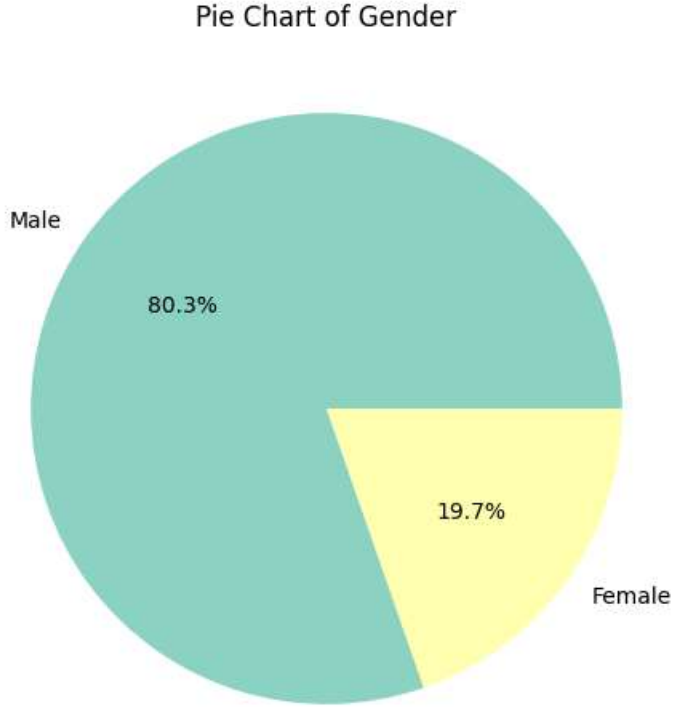

Bar Chart of Self_Employed
300
250
200

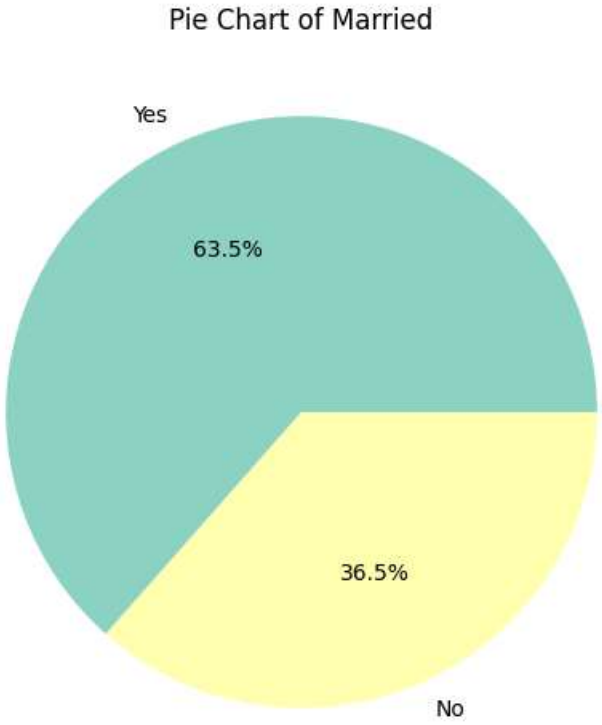

count
150
100
50
0
No
Yes
Self_Employed

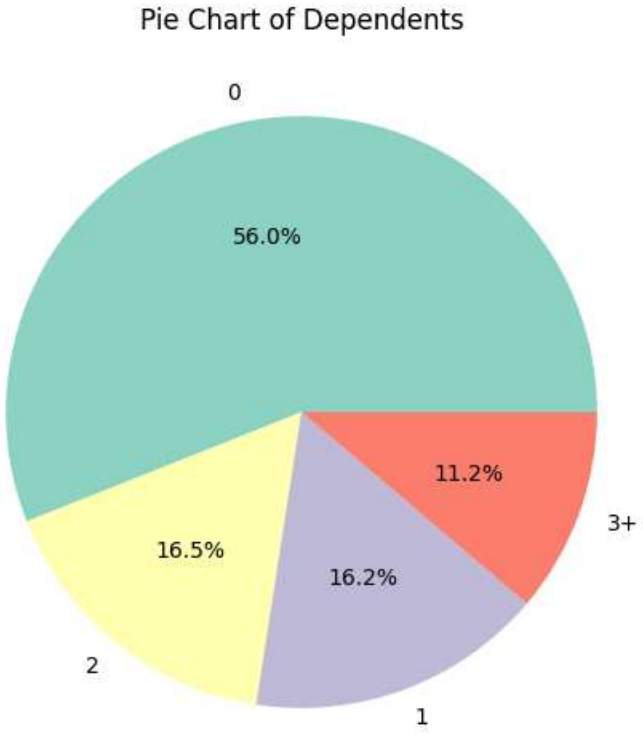

Bar Chart of Property_Area
count
140
120
100
80
60
40
20
0
Urban
Semiurban
Rural
Property_Area

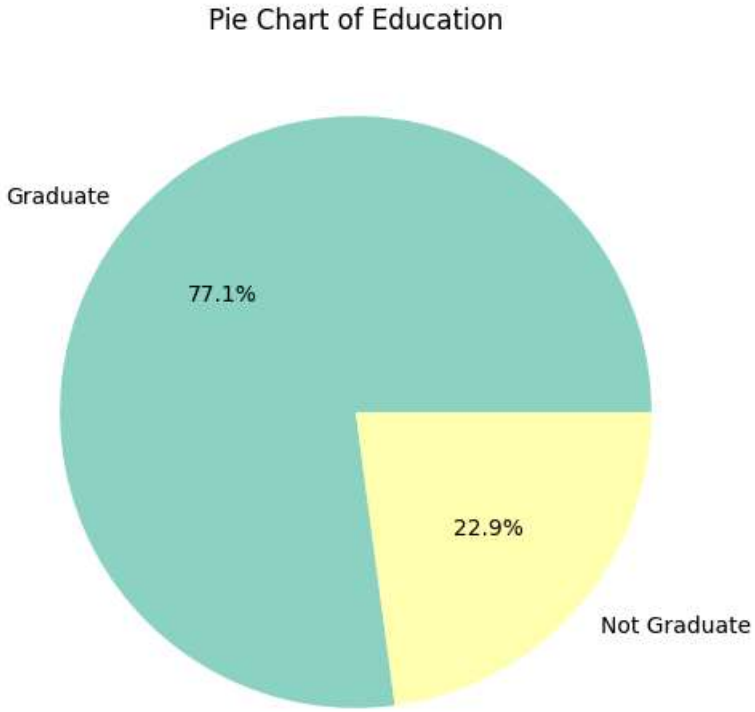
```
# Plot pie charts for categorical variables
for col in categorical_columns:
    plt.figure(figsize=(8, 6))
    df[col].value_counts().plot(kind='pie', autopct='%1.1f%%', colors=sns.color_palette('Set3', len(df[col].unique())))
    plt.title(f'Pie Chart of {col}')
    plt.ylabel('')
    plt.show()
```

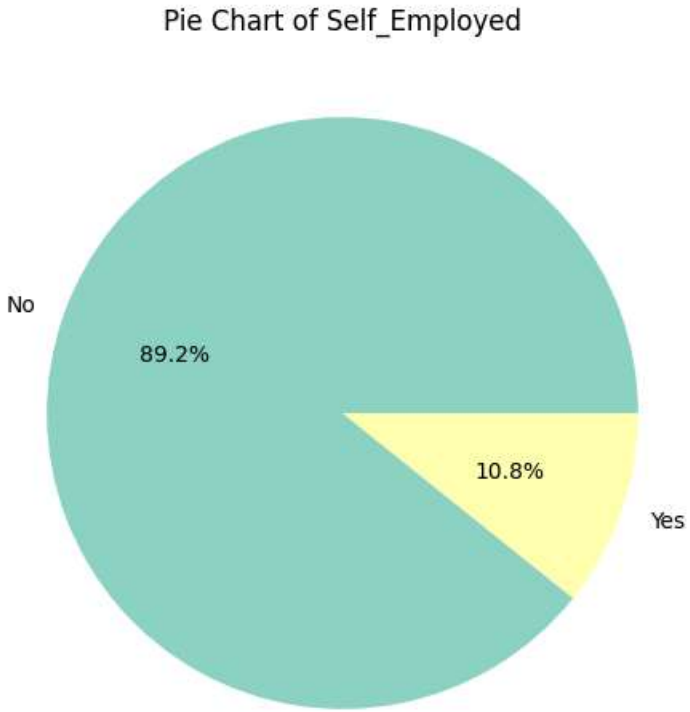

Pie Chart of Loan_ID

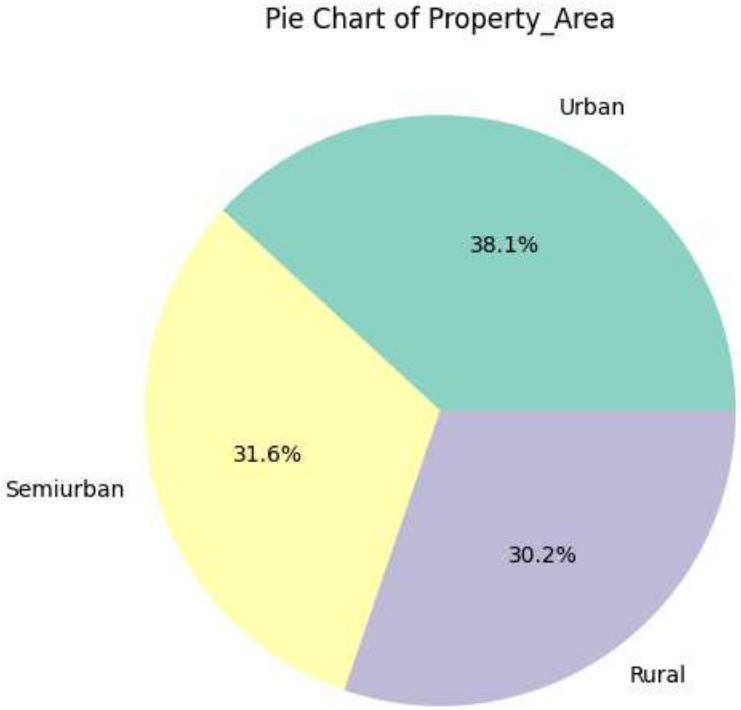

Pie Chart of Gender
Male
80.3%
19.7%
Female


Pie Chart of Married
Yes
63.5%
36.5%
No

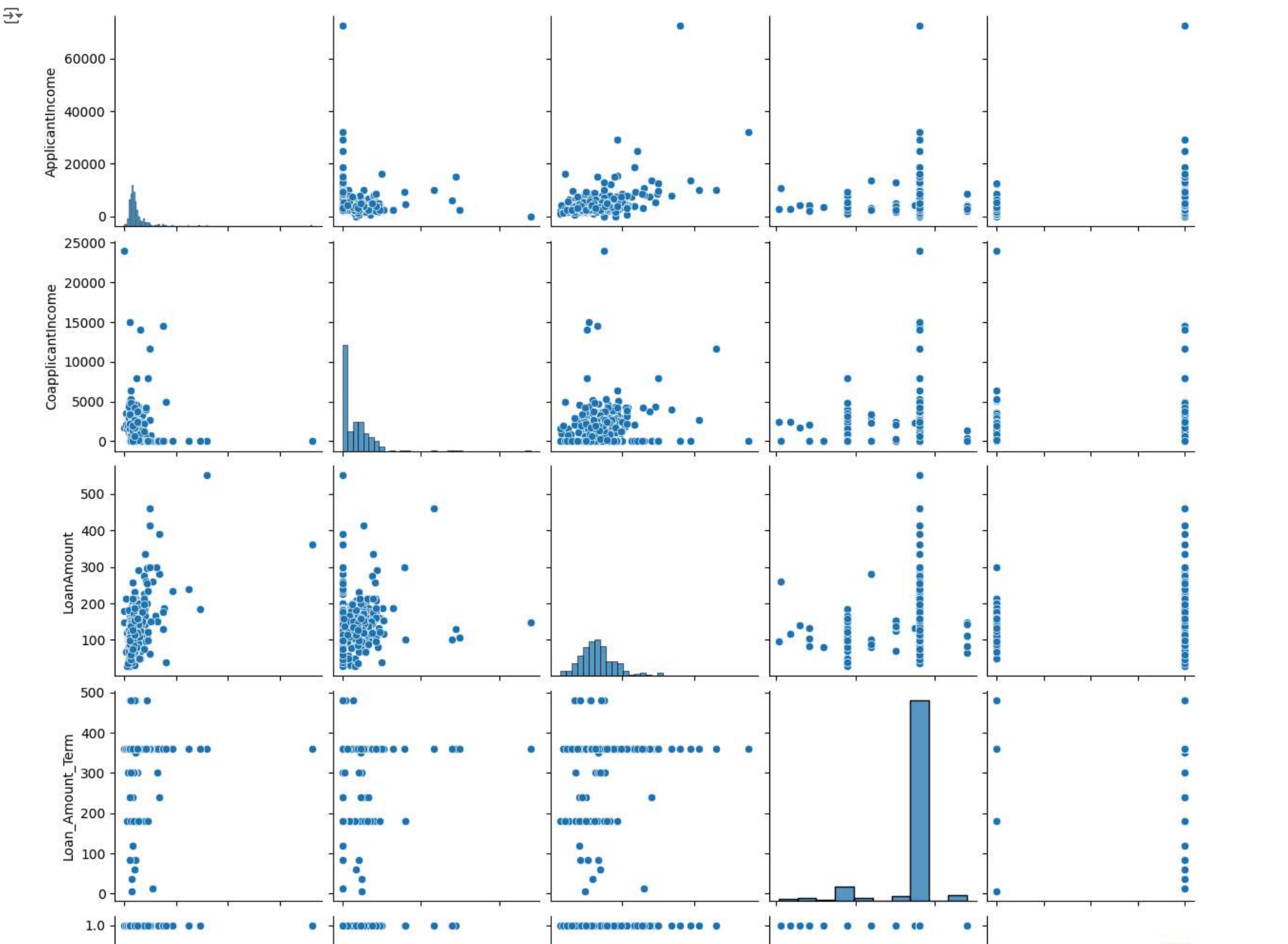

Pie Chart of Dependents
0
56.0%
11.2%
3+
16.5%
16.2%
2
1

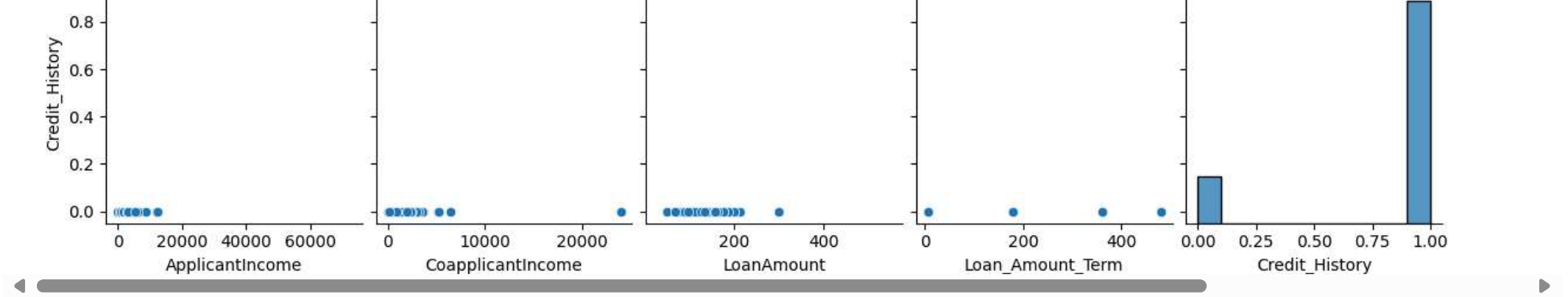

Pie Chart of Education
Graduate
77.1%
22.9%
Not Graduate


Pie Chart of Self_Employed
No
89.2%
10.8%
Yes

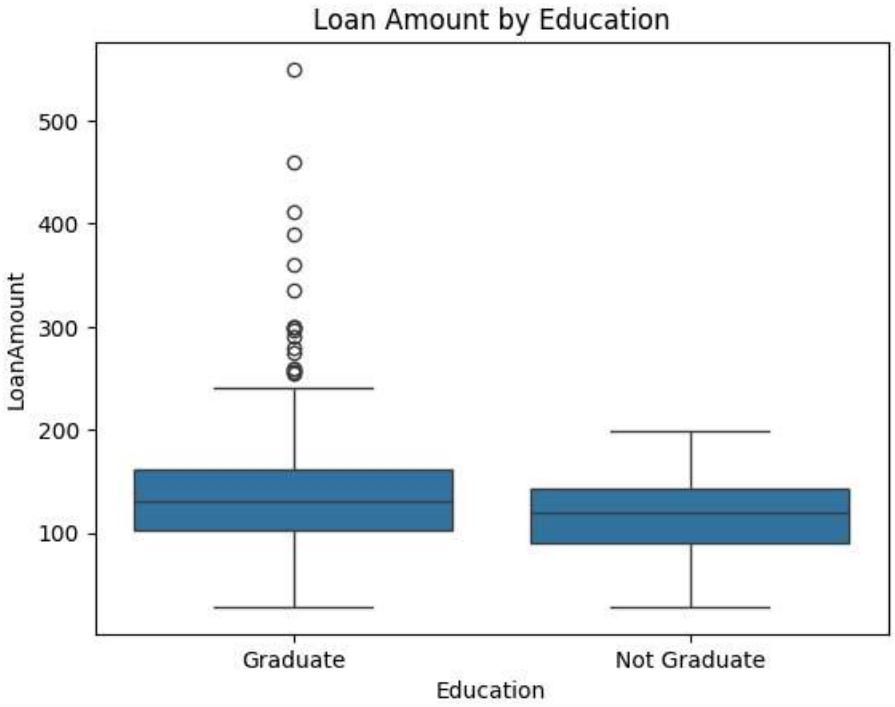

Pie Chart of Property_Area
Urban
38.1%
Semiurban
31.6%
30.2%
Rural

```
#Pair plot for all numeric columns
sns.pairplot(df)
plt.show()
```

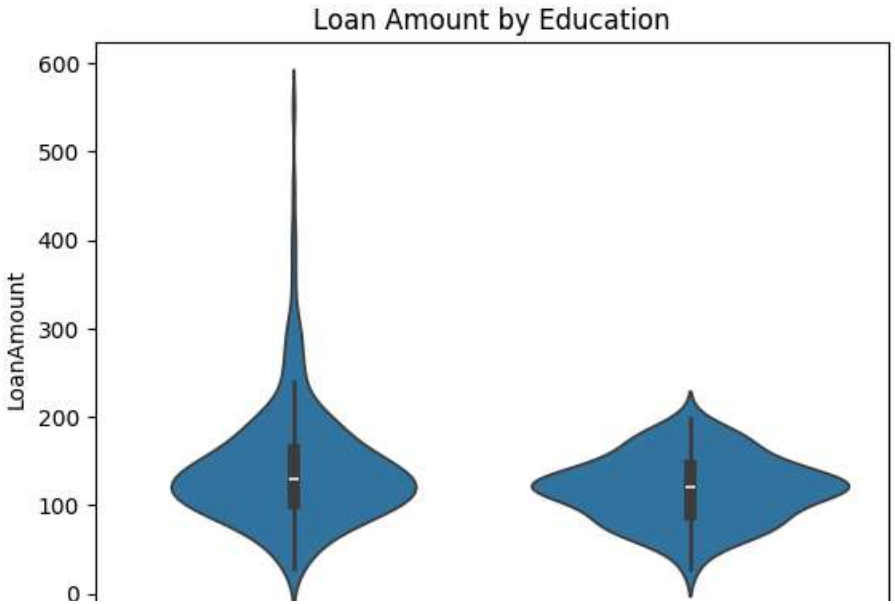

ApplicantIncome
CoapplicantIncome
LoanAmount
Loan_Amount_Term
0
20000
40000
60000
5000
10000
15000
20000
25000
100
200
300
400
500
1.0




```
# Box plot of 'numeric_feature' by 'categorical_feature'
sns.boxplot(x='Education', y='LoanAmount', data=df)
plt.title('Loan Amount by Education')
plt.show()
```




```
# Violin plot
sns.violinplot(x='Education', y='LoanAmount', data=df)
plt.title('Loan Amount by Education')
plt.show()
```

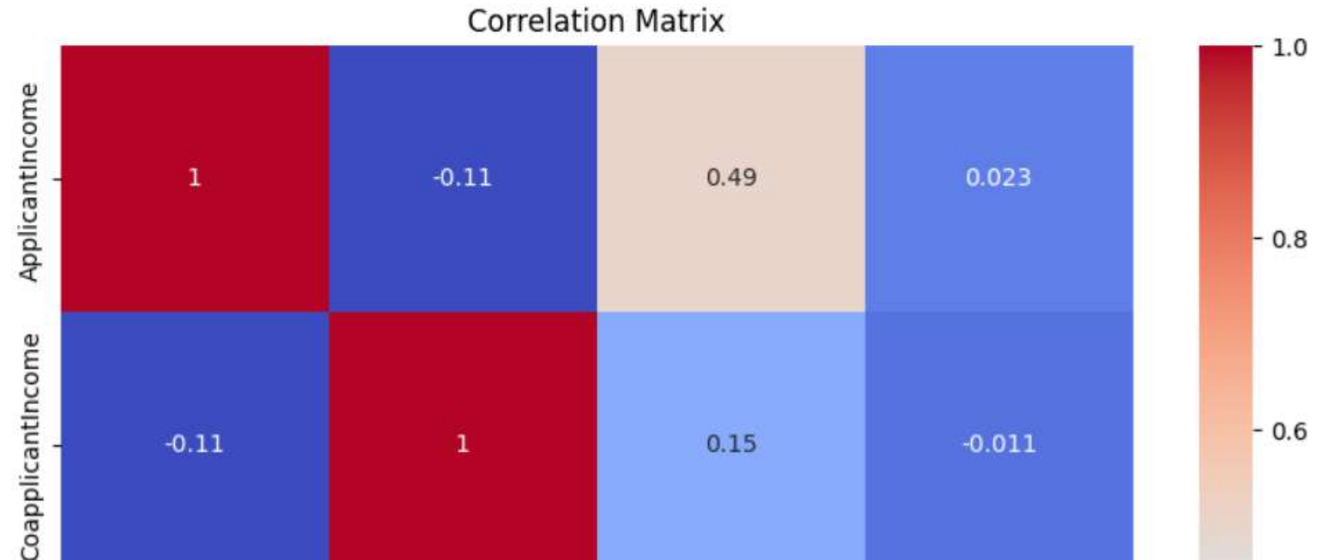

Loan Amount by Education
LoanAmount
600
500
400
300
200
100
0

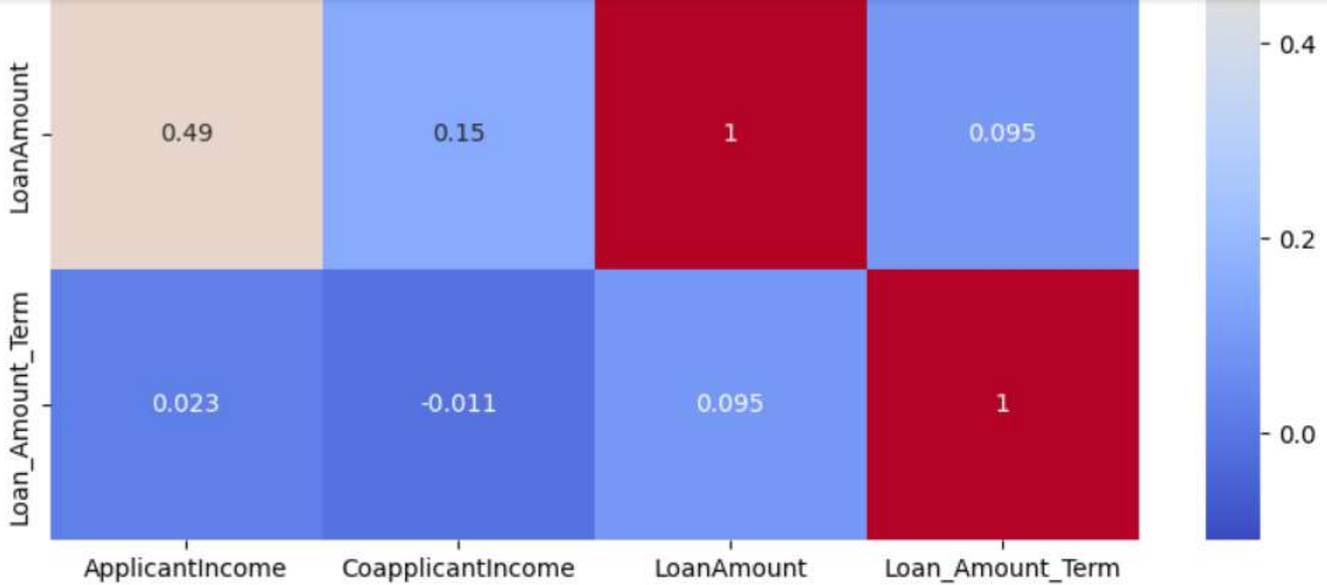
```
# Select numeric columns for correlation analysis
numeric_cols = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term']
```

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

```
# Visualize the correlation matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
```

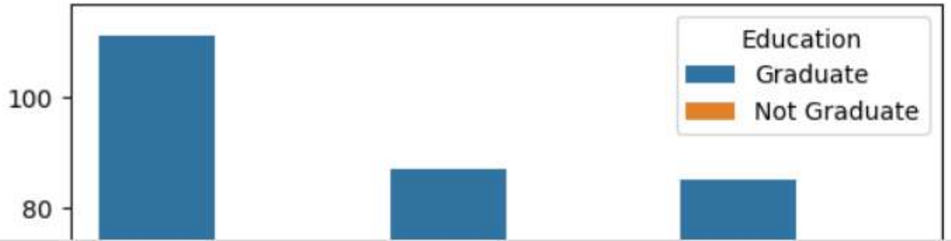
```
Text(0.5, 1.0, 'Correlation Matrix')
```





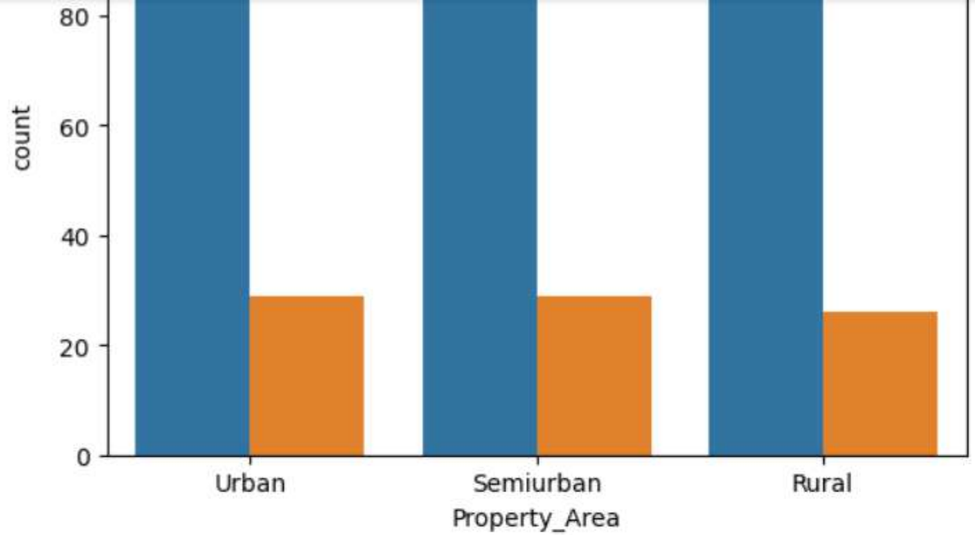

```
# Create the stacked bar chart
sns.countplot(x='Property_Area', hue='Education', data=df)
```

```
<Axes: xlabel='Property_Area', ylabel='count'>
```



[ ]

[↓]



[ ]
```
import geopandas as gpd
import folium
```

[ ]
```
# Assign approximate coordinates to property areas
property_area_coords = {
    'Urban': (28.6139, 77.2090),
    'Semiurban': (27.1750, 78.0422),
    'Rural': (23.0225, 77.4620)
}

# Map property areas to coordinates
df['latitude'] = df['Property_Area'].map(lambda x: property_area_coords[x][0])
```

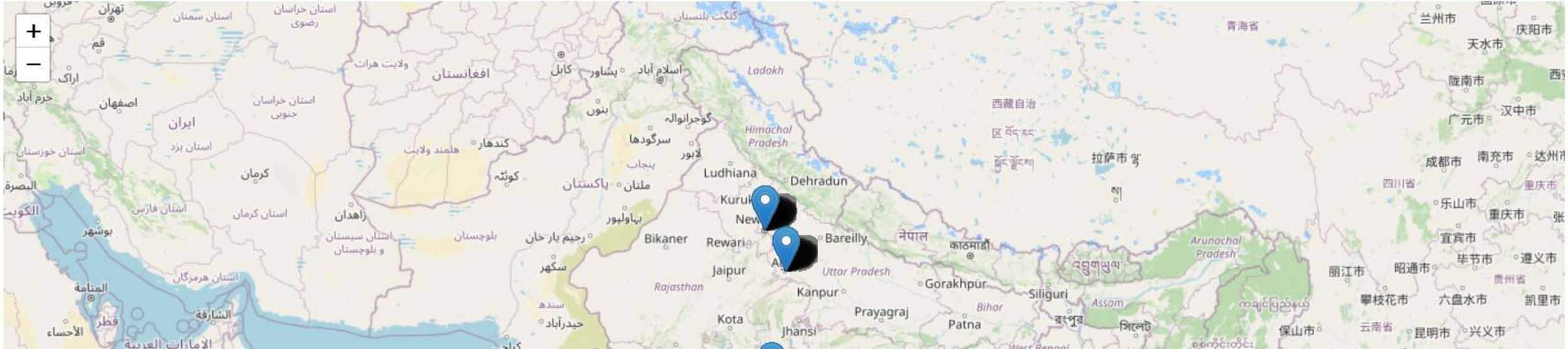
[ ]

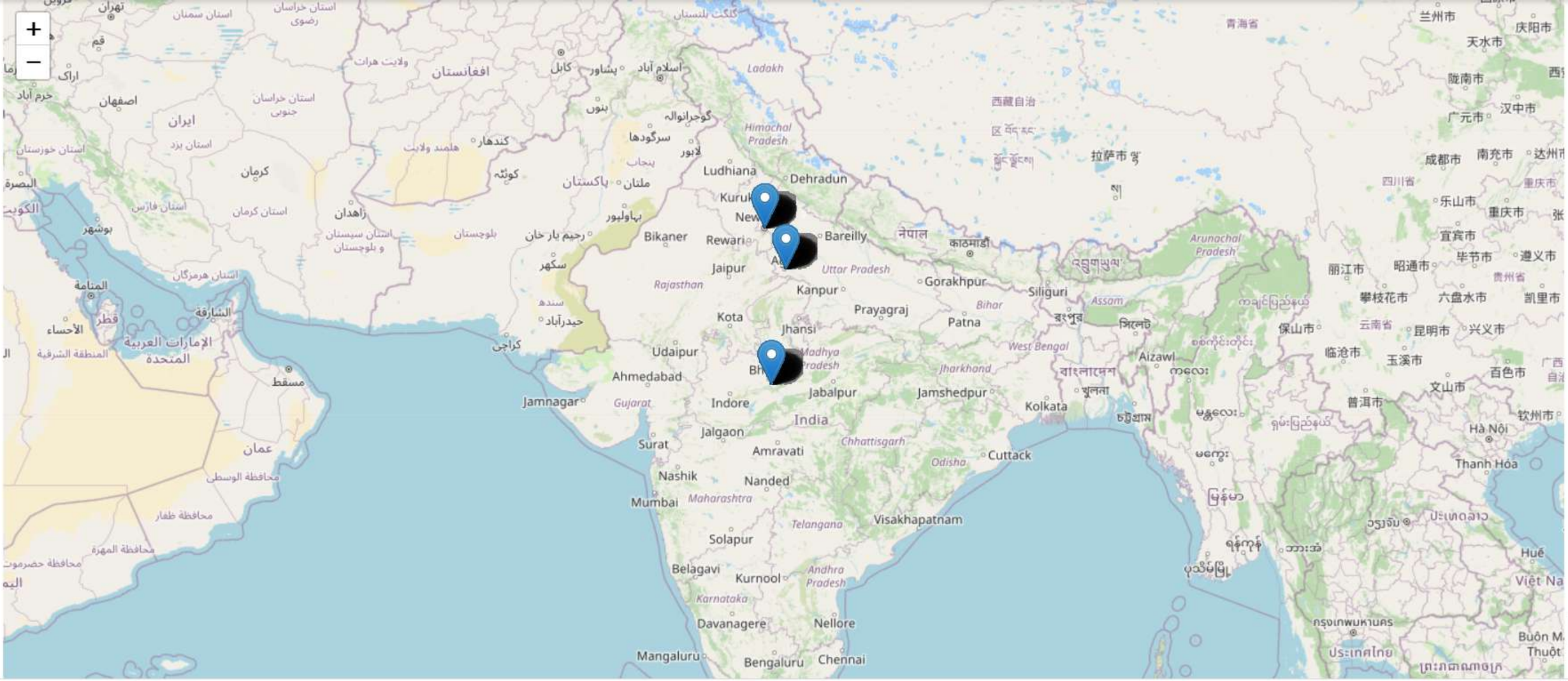
```
# Map property areas to coordinates
df['latitude'] = df['Property_Area'].map(lambda x: property_area_coords[x][0])
df['longitude'] = df['Property_Area'].map(lambda x: property_area_coords[x][1])

# Create a base map
m = folium.Map(location=[20, 78], zoom_start=5)

# Add markers to the map
for index, row in df.iterrows():
    folium.Marker([row['latitude'], row['longitude']], popup=row['Property_Area']).add_to(m)

# Display the map
m
```




Ludhiana
Dehradun
Bareilly
Jaipur
Kanpur
Gorakhpur
Siliguri
Kota
Prayagraj
Patna
Jhansi
Udaipur
Ahmedabad
Jabalpur
Jamshedpur
Kolkata
Indore
Jamnagar
India
Surat
Jalgaon
Amravati
Cuttack
Nashik
Nanded
Mumbai
Visakhapatnam
Solapur
Belagavi
Kurnool
Davanagere
Nellore
Mangaluru
Bengaluru
Chennai
Aizawl
Ladakh
Himachal Pradesh
Uttar Pradesh
Rajasthan
Bihar
Assam
West Bengal
Jharkhand
Madhya Pradesh
Chhattisgarh
Odisha
Maharashtra
Telangana
Andhra Pradesh
Karnataka
Gujarat