Loan Approval Data Analysis

```
#Importing Major Liberaries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

import warnings
warnings.filterwarnings('ignore')
```

Task 1: Data Exploration

Load the dataset into a Python environment

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

df
```

```
      Loan_ID Gender Married Dependents     Education Self_Employed  \
0    LP001015   Male     Yes          0      Graduate            No
1    LP001022   Male     Yes          1      Graduate            No
2    LP001031   Male     Yes          2      Graduate            No
3    LP001035   Male     Yes          2      Graduate            No
4    LP001051   Male      No          0  Not Graduate            No
..        ...    ...     ...        ...           ...           ...
362  LP002971   Male     Yes         3+  Not Graduate           Yes
363  LP002975   Male     Yes          0      Graduate            No
364  LP002980   Male      No          0      Graduate            No
365  LP002986   Male     Yes          0      Graduate            No
366  LP002989   Male      No          0      Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term
\
0               5720                  0       110.0             360.0

1               3076               1500       126.0             360.0

2               5000               1800       208.0             360.0

3               2340               2546       100.0             360.0

4               3276                  0        78.0             360.0

..               ...                ...         ...               ...

362             4009               1777       113.0             360.0
```

```
363              4158                 709       115.0              360.0

364              3250                1993       126.0              360.0

365              5000                2393       158.0              360.0

366              9200                   0        98.0              180.0


     Credit_History Property_Area
0               1.0         Urban
1               1.0         Urban
2               1.0         Urban
3               NaN         Urban
4               1.0         Urban
..              ...           ...
362             1.0         Urban
363             1.0         Urban
364             NaN     Semiurban
365             1.0         Rural
366             1.0         Rural

[367 rows x 12 columns]
```

Display the first few rows of the dataset to understand its structure.

```
df.head(5)
```

```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001015   Male     Yes          0      Graduate            No
1  LP001022   Male     Yes          1      Graduate            No
2  LP001031   Male     Yes          2      Graduate            No
3  LP001035   Male     Yes          2      Graduate            No
4  LP001051   Male      No          0  Not Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5720                  0       110.0             360.0
1             3076               1500       126.0             360.0
2             5000               1800       208.0             360.0
3             2340               2546       100.0             360.0
4             3276                  0        78.0             360.0

   Credit_History Property_Area
0             1.0         Urban
1             1.0         Urban
2             1.0         Urban
3             NaN         Urban
4             1.0         Urban
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            367 non-null    object 
 1   Gender             356 non-null    object 
 2   Married            367 non-null    object 
 3   Dependents         357 non-null    object 
 4   Education          367 non-null    object 
 5   Self_Employed      344 non-null    object 
 6   ApplicantIncome    367 non-null    int64  
 7   CoapplicantIncome  367 non-null    int64  
 8   LoanAmount         362 non-null    float64
 9   Loan_Amount_Term   361 non-null    float64
 10  Credit_History     338 non-null    float64
 11  Property_Area      367 non-null    object 
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
```

- It contains 367 rows
- It contains 12 columns in total
- It contains null values
- It contains 5 numerical columns and 7 object type(categorical or textual) columns
- It containf 3 float type columns , 2 int type column and 7 textual columns

Check for missing values and handle them if necessary.

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

```
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
```

- Gender has 11 null values
- Dependents has 10 null values
- Self_Employed has 23 null values
- LoanAmount has 5 null values
- Loan_Amount_Term has 6 null values

- Credit_History has 29 null values

Removing Null values of Gender Column with the mode value

```
df.Gender.fillna(df.Gender.mode()[0],inplace=True)

df.Gender.isnull().sum()

0
```

The null values of Gender Column has been filled

Removing Null values of Dependents Column with the mode value

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

10

df.Dependents.fillna(df.Dependents.mode()[0],inplace=True)

df.Dependents.isnull().sum()

0
```

The null values of Dependents Column has been filled

Removing Null Self_Employed of Dependents Column with the mode value

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

23

df.Self_Employed.fillna(df.Self_Employed.mode()[0],inplace=True)

df.Self_Employed.isnull().sum()

0
```

The Null values of Self_Employed column has been filled

Removing Null values of LoanAmount Column with the mean value

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

5

df.LoanAmount.fillna(df.LoanAmount.mean(),inplace=True)

df.LoanAmount.isnull().sum()

0
```

The Null values of LoanAmount column has been filled

Removing Null values of Loan_Amount_Term Column with the mean value

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

```
6
```

```
df.Loan_Amount_Term.fillna(df.Loan_Amount_Term.mean(),inplace=True)
```

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

```
0
```

The Null values of Loan_Amount_Term column has been filled

Removing Null values of Credit_History Column with the mean value

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

```
29
```

```
df.Credit_History.fillna(df.Credit_History.mean(),inplace=True)
```

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

```
0
```

The Null values of Credit_History column has been filled

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

```
Loan_ID               0
Gender                0
Married               0
Dependents            0
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History        0
Property_Area         0
dtype: int64
```

All the null values of the dataset has been filled

Summarize basic statistics (mean, median, standard deviation, etc.)for the numeric columns.

```
df.describe()
```

```
       ApplicantIncome  CoapplicantIncome  LoanAmount
Loan_Amount_Term  \
```

```
count       367.000000          367.000000  367.000000
367.000000
mean       4805.599455         1569.577657  136.132597
342.537396
std        4910.685399         2334.232099   60.946040
64.620366
min           0.000000            0.000000   28.000000
6.000000
25%        2864.000000            0.000000  101.000000
360.000000
50%        3786.000000         1025.000000  126.000000
360.000000
75%        5060.000000         2430.500000  157.500000
360.000000
max       72529.000000        24000.000000  550.000000
480.000000

       Credit_History
count      367.000000
mean         0.825444
std          0.364778
min          0.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          1.000000
```

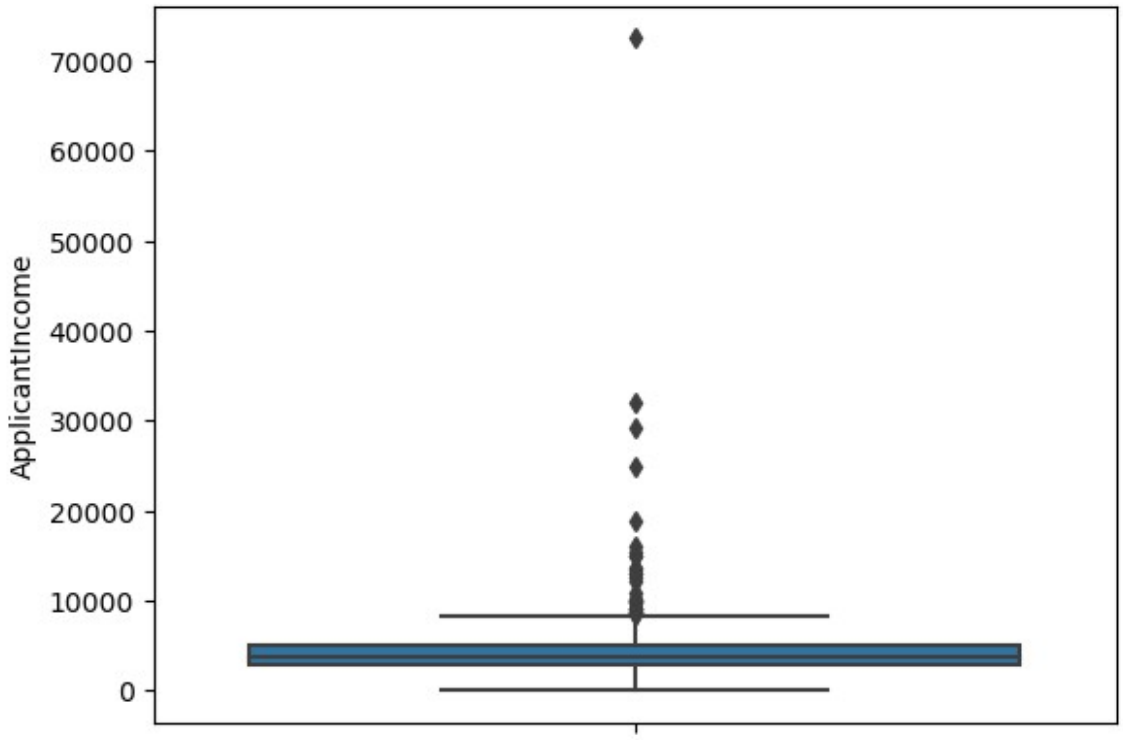
Task 2: Data Visualization

2.1 Univariate Analysis

Explore the distribution of numeric columns using the following visualizations:

- Histograms: Plot the frequency distribution of key numeric variables.
- Box Plots: Identify potential outliers and visualize the spread of data.

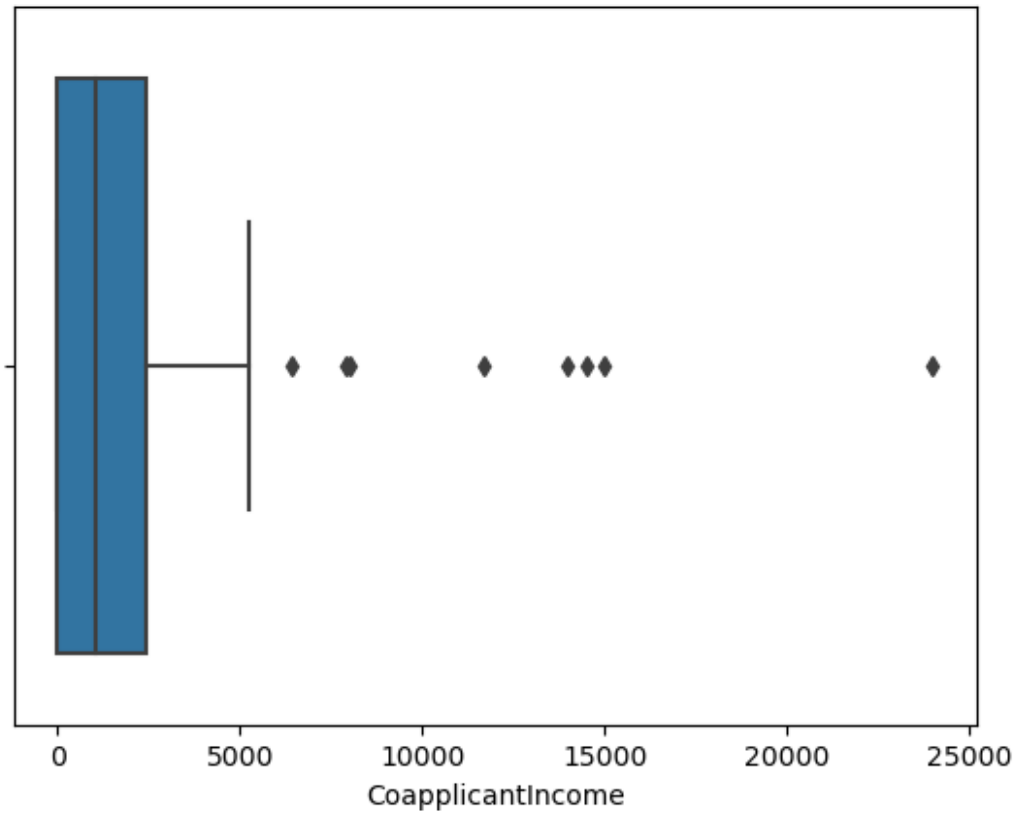
```
sns.boxplot(df,y=df.ApplicantIncome)
plt.show()
```



- The ApplicantIncome column has various outliers
- Most of the Applicants have income less than 10000

```
sns.boxplot(df,x=df.CoapplicantIncome)
```

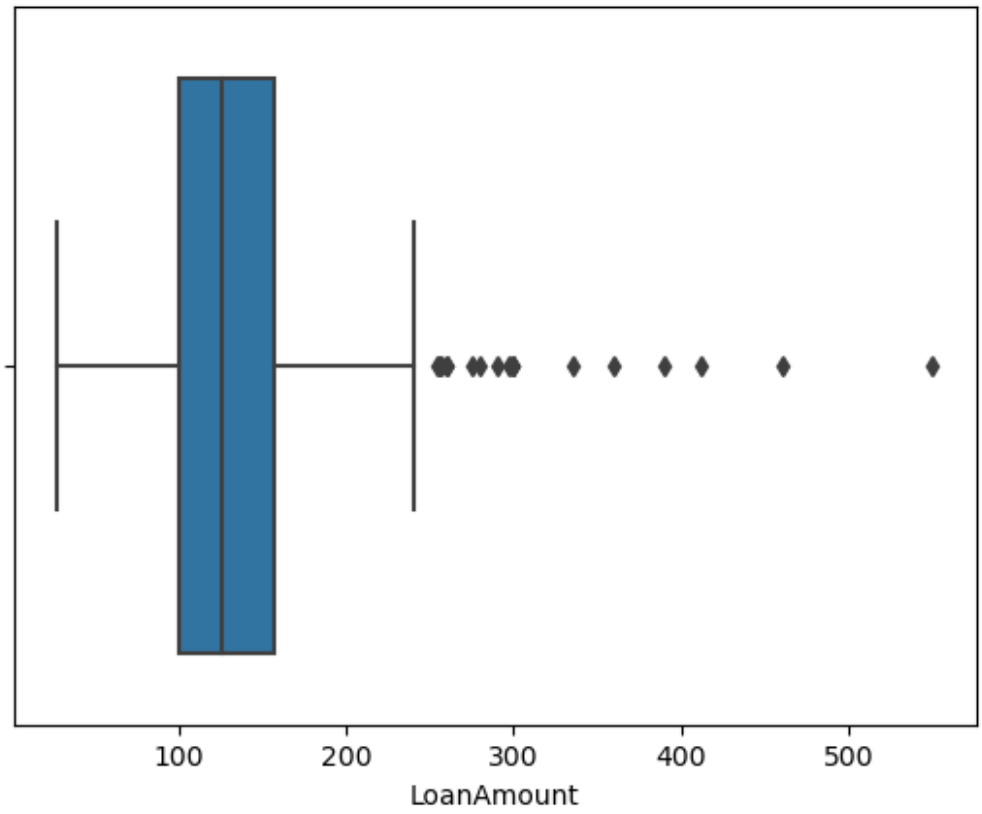
```
<Axes: xlabel='CoapplicantIncome'>
```



- The CoapplicantIncome column too have few outliers
- Most of the Coapplicant have income less than 5000

```
sns.boxplot(df,x=df.LoanAmount)
```

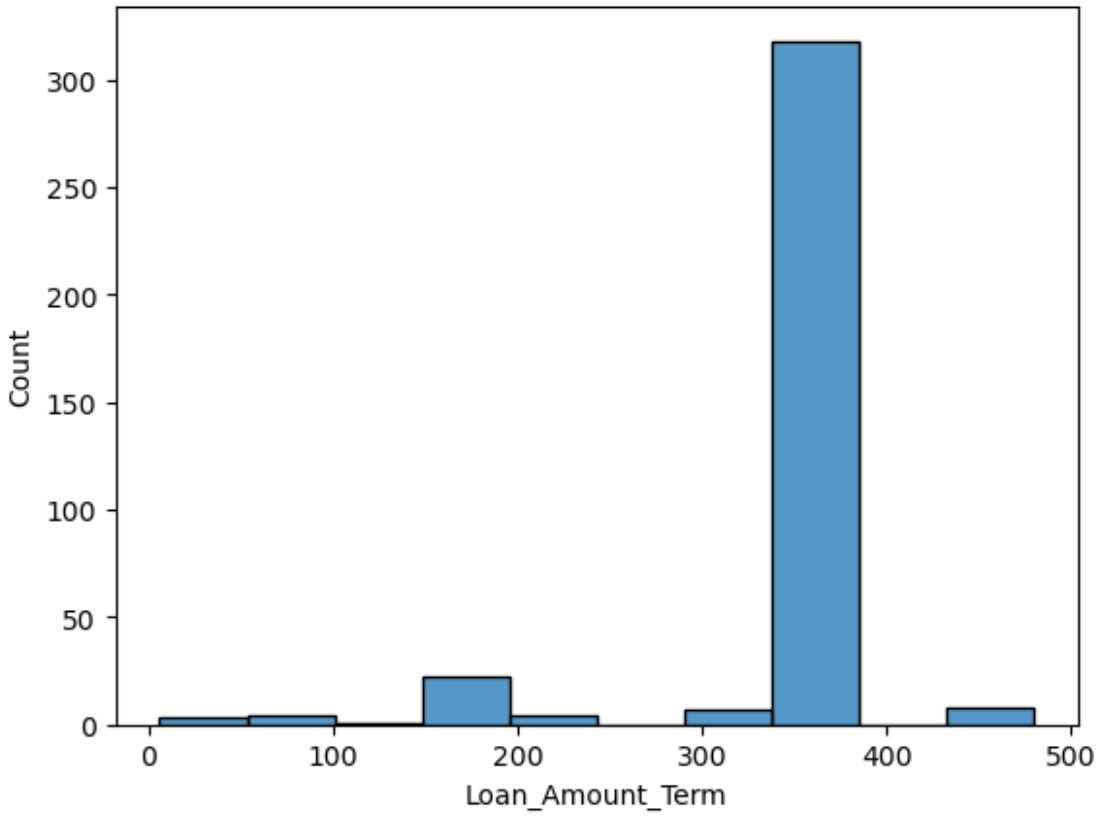
```
<Axes: xlabel='LoanAmount'>
```



The LoanAmount column has many outliers

```
sns.histplot(df.Loan_Amount_Term)
```

```
<Axes: xlabel='Loan_Amount_Term', ylabel='Count'>
```

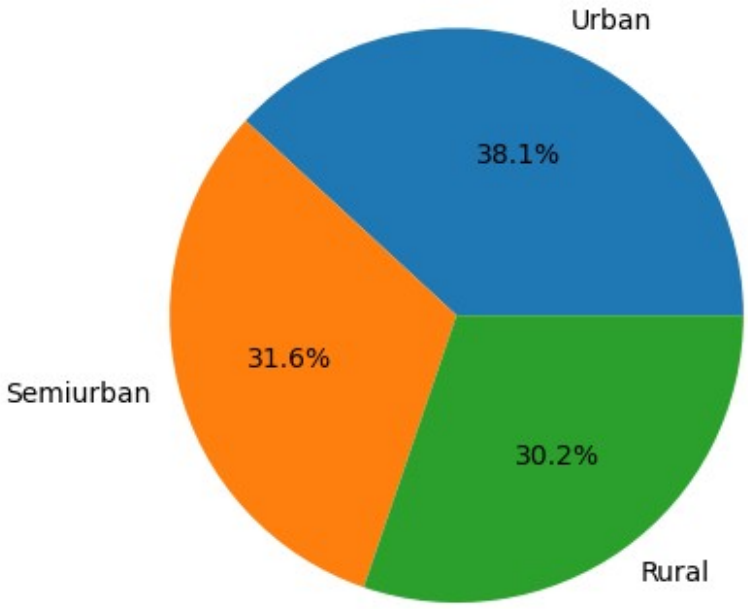


Analyze categorical variables by creating the following plots:

- Bar Charts: Visualize the frequency distribution of categoricalvariables.
- Pie Charts: Represent the composition of categorical variables.

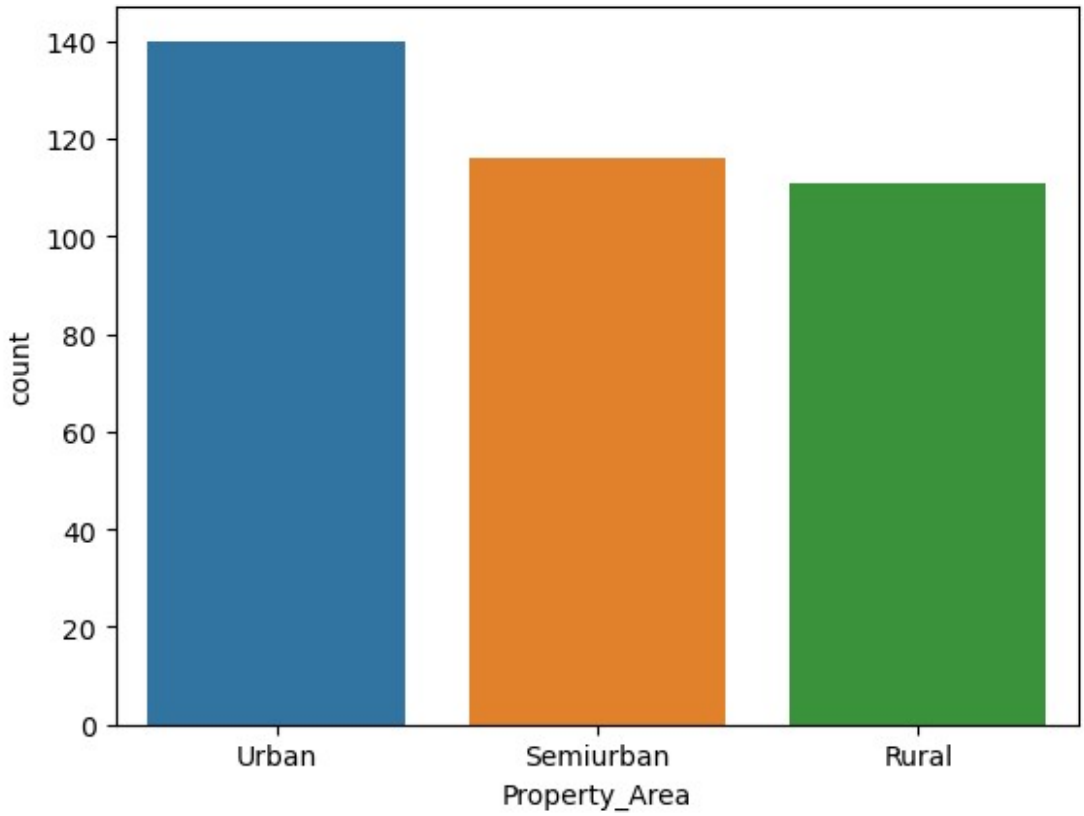
Property_Area

```
plt.pie(df.Property_Area.value_counts(),labels=df.Property_Area.value_
counts().index,autopct='%1.1f%%')
plt.show()
```



- 38.1% of Loan Applicants are from Urban Area
- 31.6% of Loan Apllicants are from Semiurban Area
- 30.2% of Loan Apllicants are from Rural Area

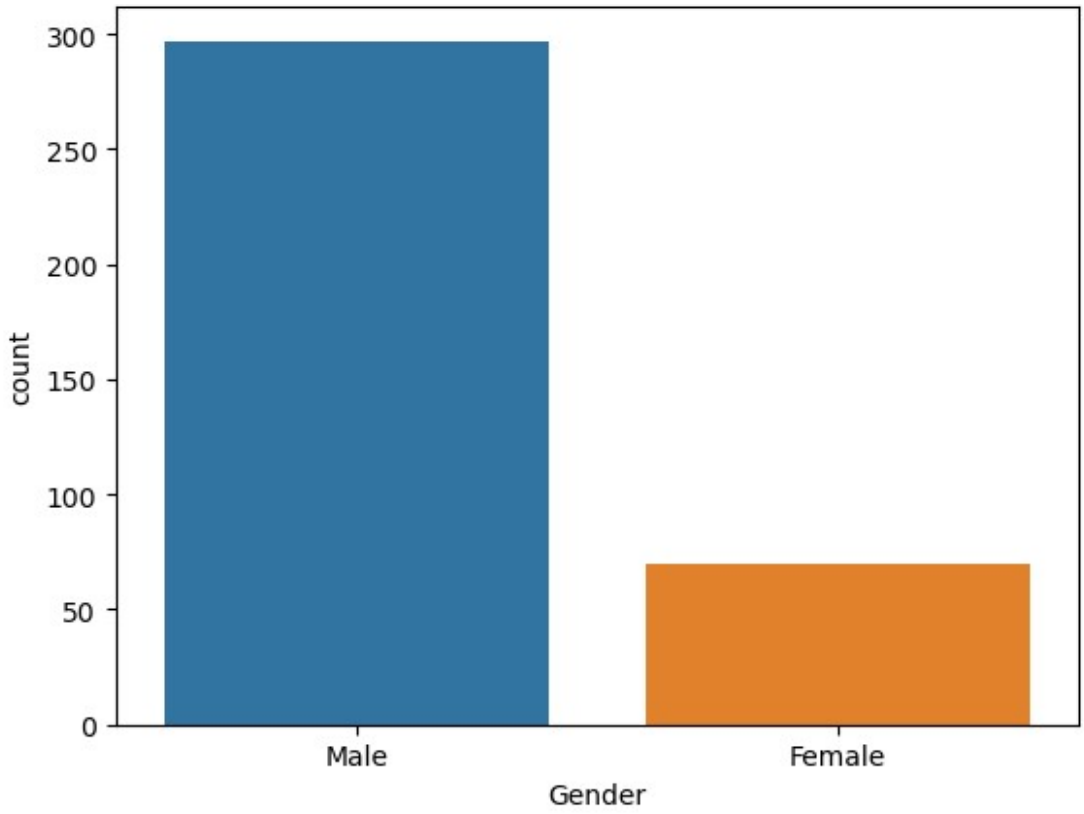
```
sns.colors
sns.barplot(y=df.Property_Area.value_counts(),x=df.Property_Area.value
_counts().index)
plt.show()
```



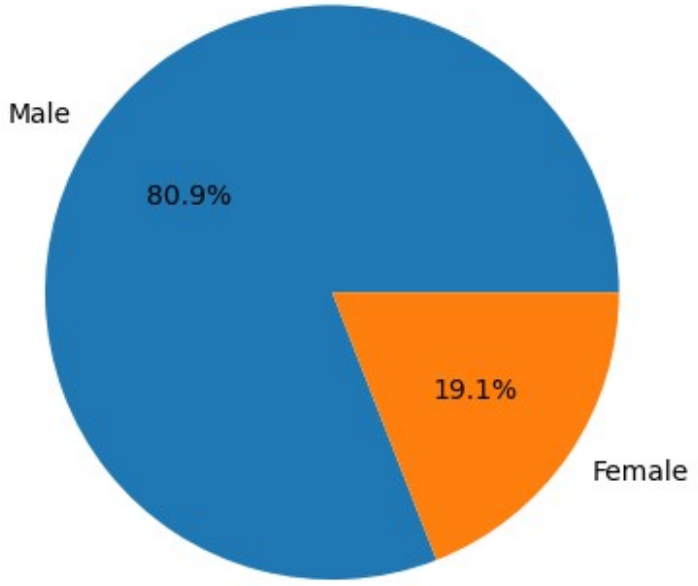
Most of the Loan Applicants are from Urban Area

Gender

```
sns.barplot(x=df.Gender.value_counts().index,y=df.Gender.value_counts(
),)
plt.show()
```



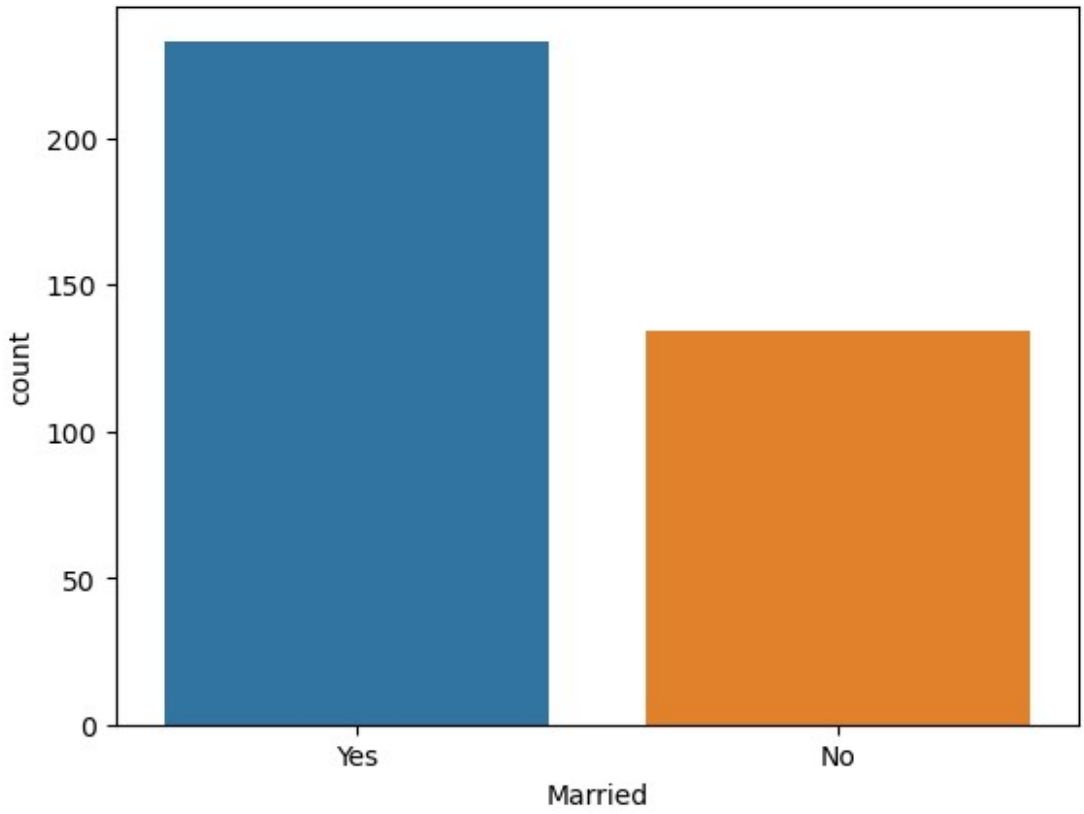

```
plt.pie(df.Gender.value_counts(),labels=df.Gender.value_counts().index
,autopct='%1.1f%%')
plt.show()
```



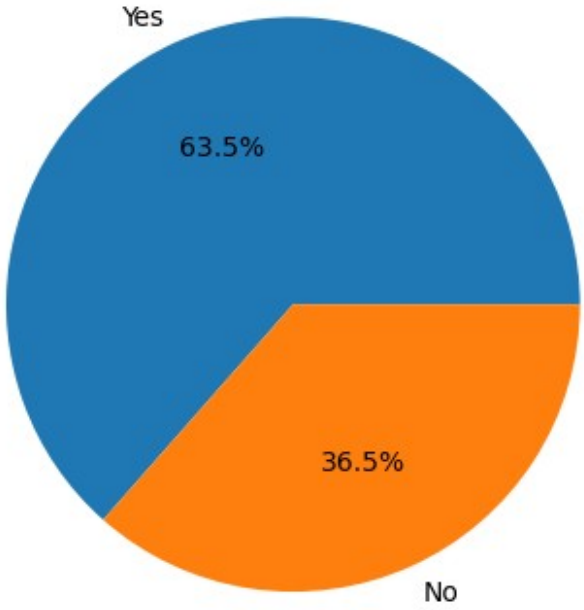
Most of the Loan Applicants are Male

Married

```
sns.barplot(x=df.Married.value_counts().index,y=df.Married.value_count
s())
plt.show()
```



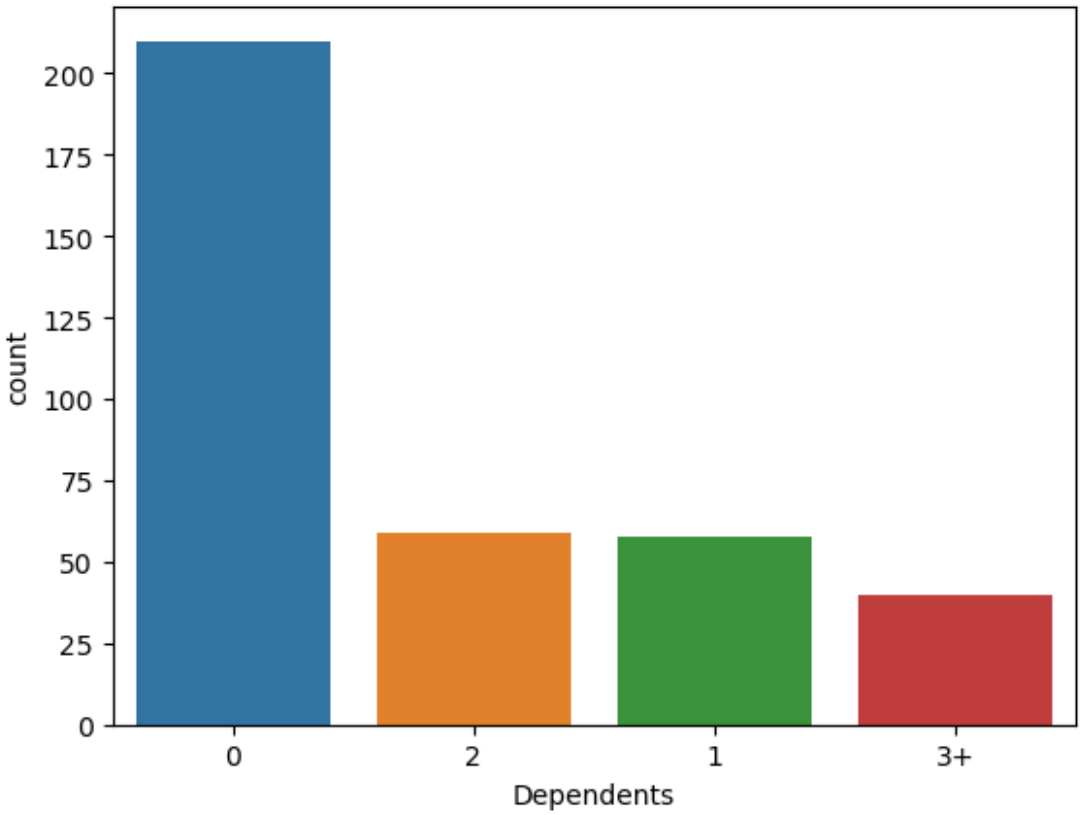

```
plt.pie(df.Married.value_counts(),labels=df.Married.value_counts().ind
ex,autopct='%1.1f%%')
plt.show()
```



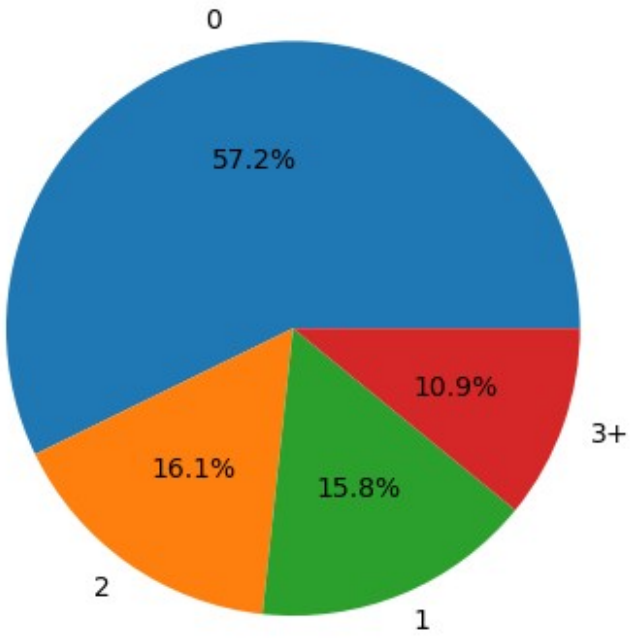
Most of the Loan Apllicants are Married

Dependents

```
sns.barplot(y=df.Dependents.value_counts(),x=df.Dependents.value_count
s().index)
plt.show()
```




```
plt.pie(df.Dependents.value_counts(),labels=df.Dependents.value_counts
().index,autopct='%1.1f%%')
plt.show()
```



57.2% Loan Applicants have 0 Dependents

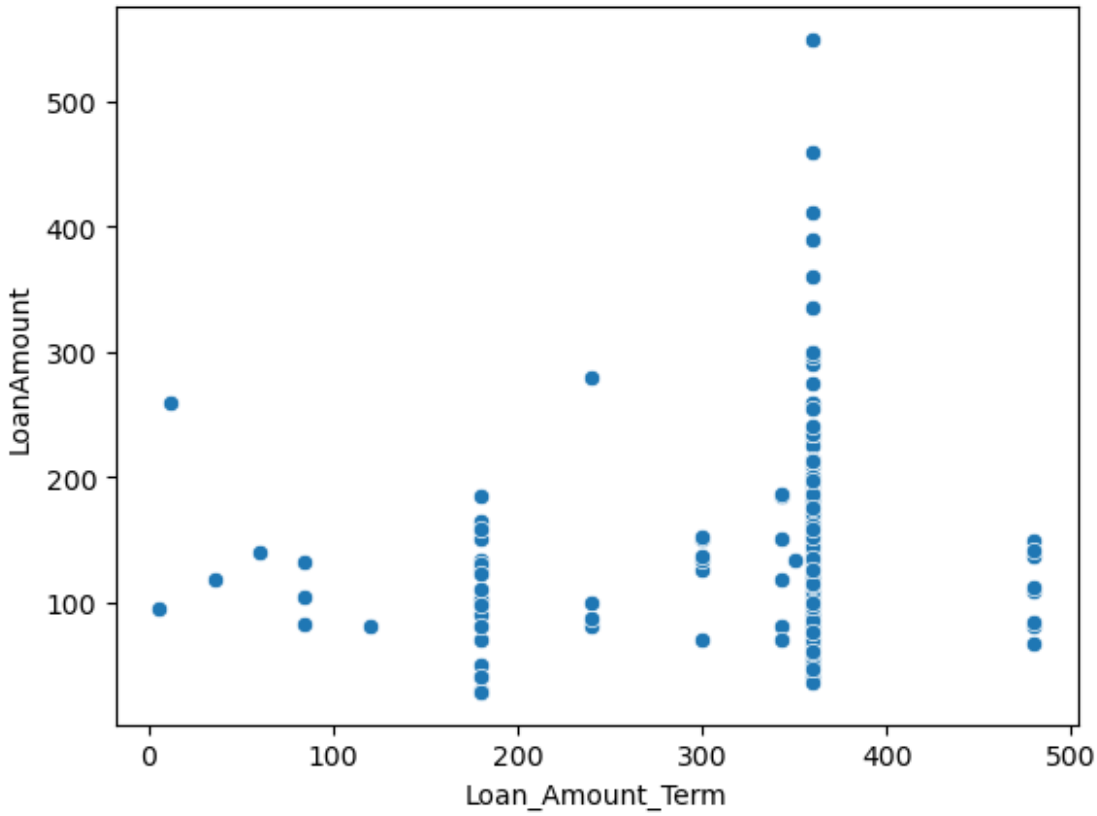
2.2 Bivariate Analysis

Create scatter plots to explore relationships between pairs of numeric variables.

LoanAmount and Loan_Amount_Term

```
sns.scatterplot(df,y=df.LoanAmount,x=df.Loan_Amount_Term)
```

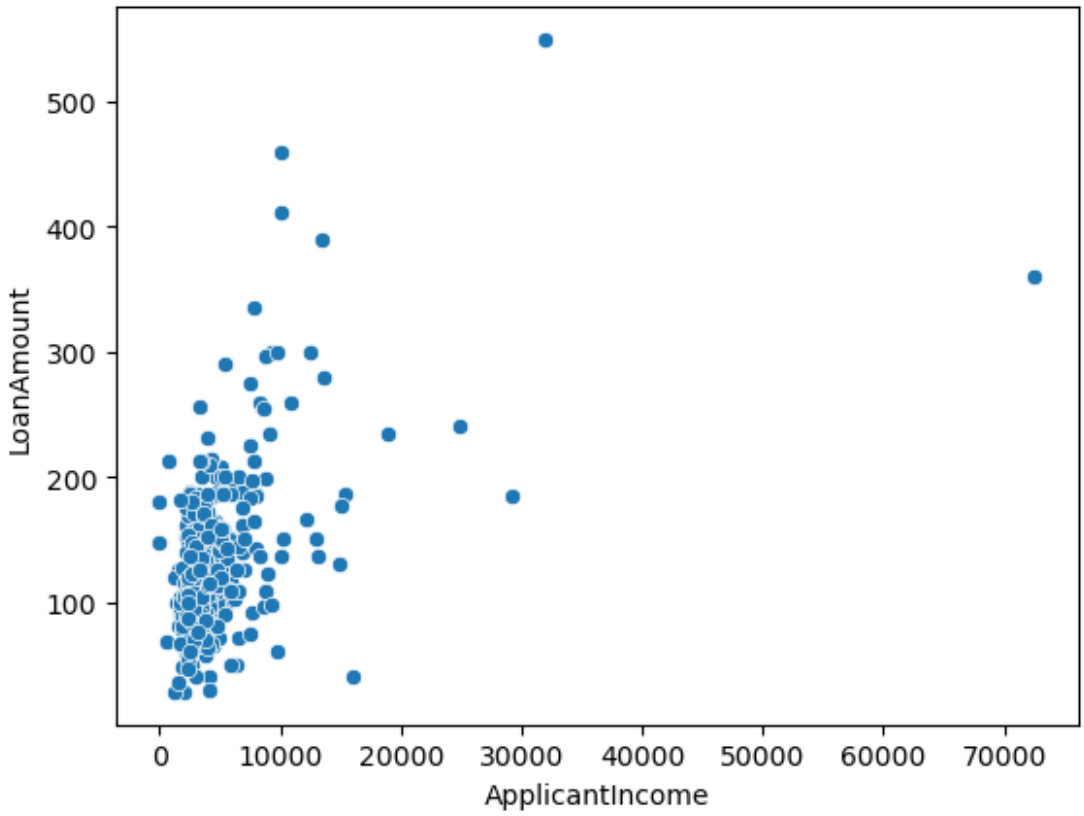
```
<Axes: xlabel='Loan_Amount_Term', ylabel='LoanAmount'>
```



ApplicantIncome and LoanAmount

```
sns.scatterplot(df,x=df.ApplicantIncome,y=df.LoanAmount)
```

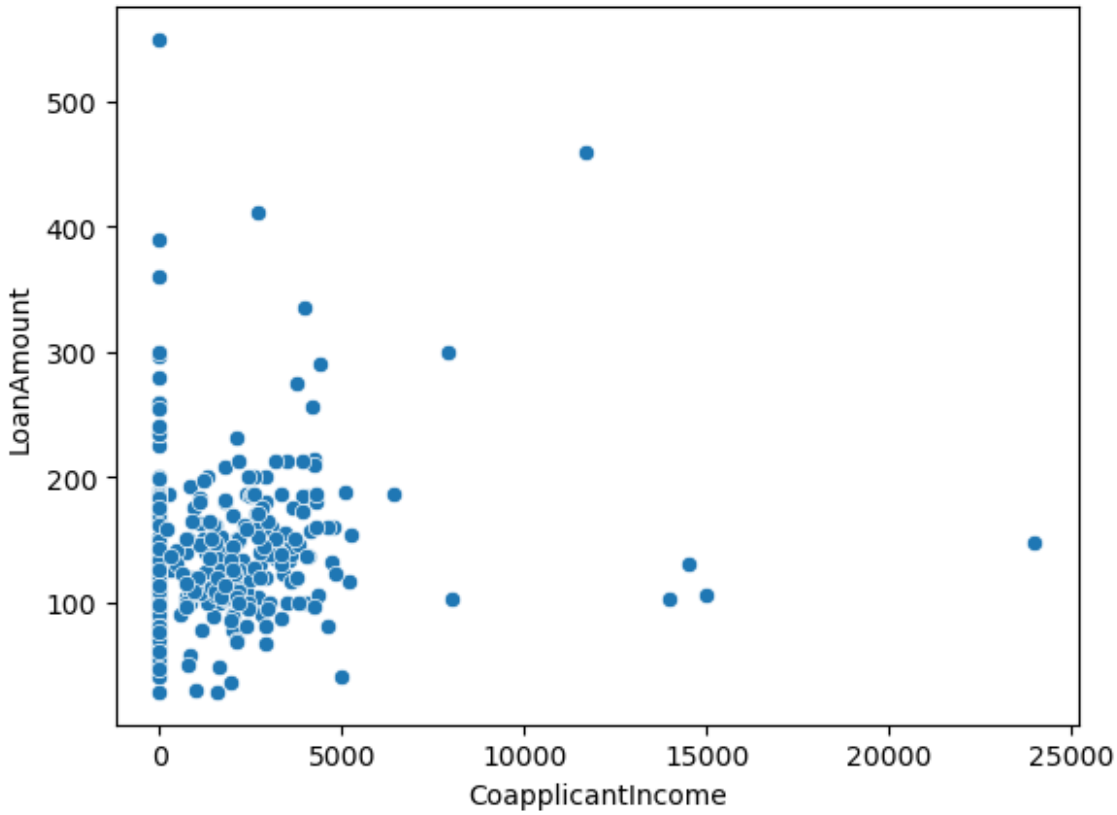
```
<Axes: xlabel='ApplicantIncome', ylabel='LoanAmount'>
```



CoapplicantIncome and LoanAmount

```
sns.scatterplot(df,x=df.CoapplicantIncome,y=df.LoanAmount)
```

```
<Axes: xlabel='CoapplicantIncome', ylabel='LoanAmount'>
```

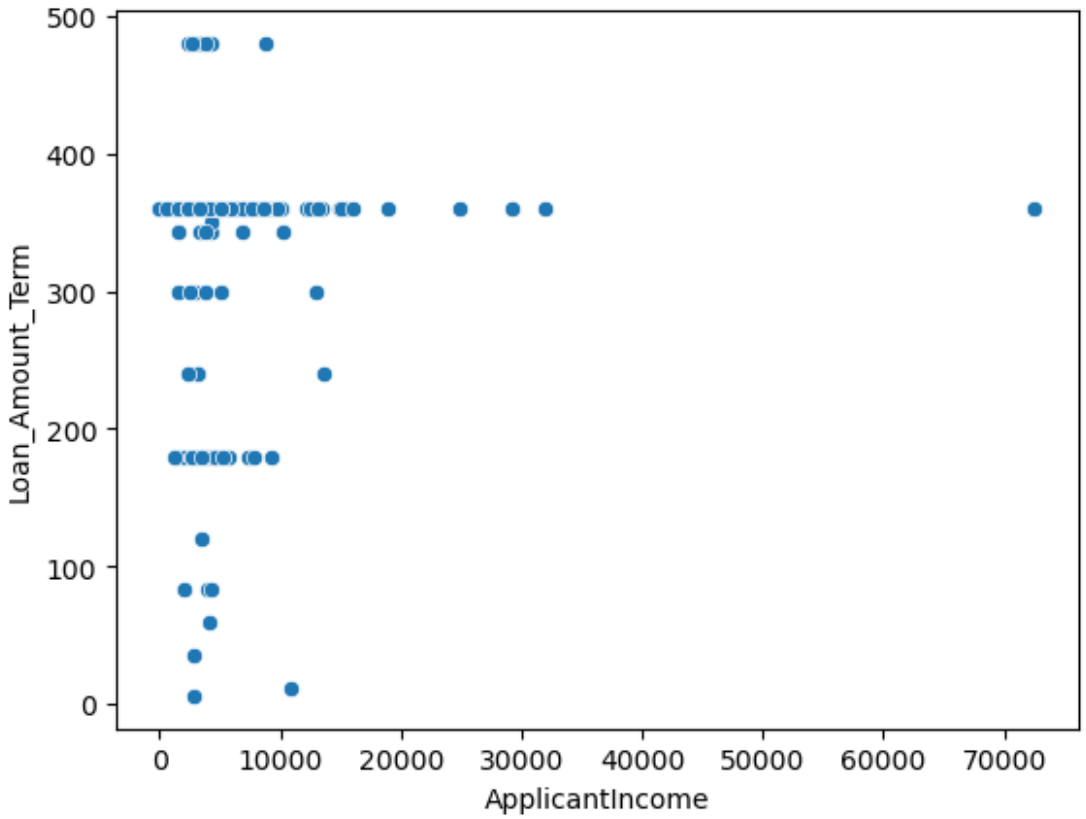


- Many of the Coapplicants have 0 income

ApplicantIncome and Loan_Amount_Term

```
sns.scatterplot(df,x=df.ApplicantIncome,y=df.Loan_Amount_Term)
```

```
<Axes: xlabel='ApplicantIncome', ylabel='Loan_Amount_Term'>
```

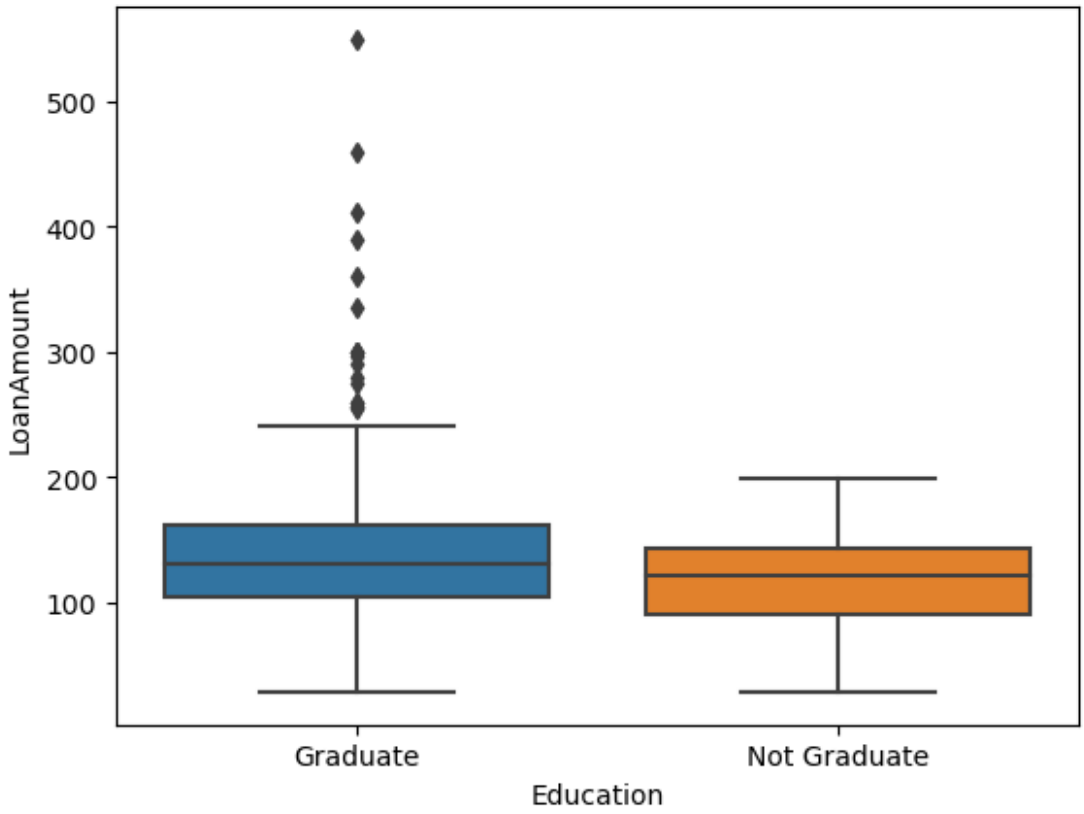


#Investigate the relationship between categorical and numericvariables using box plots or Violin Plots

Education and LoanAmount

```
sns.boxplot(x=df.Education,y=df.LoanAmount)
```

```
<Axes: xlabel='Education', ylabel='LoanAmount'>
```

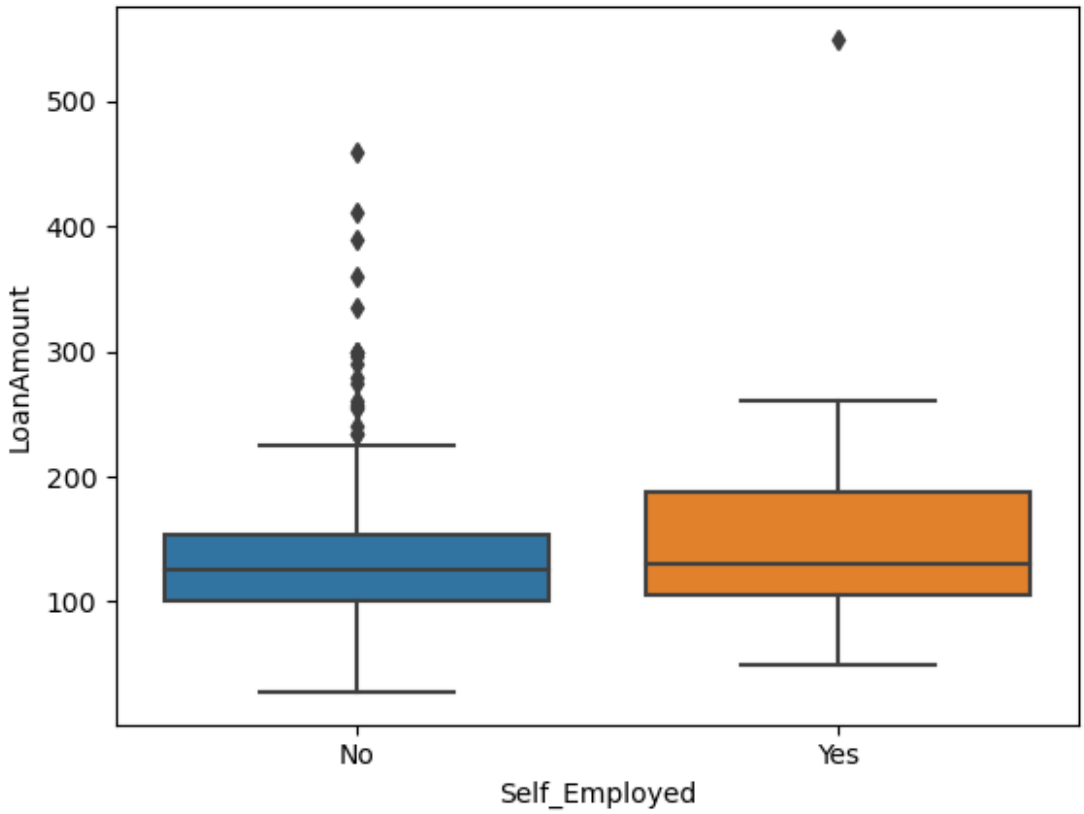


Graduated Applicants have taken more loan than the non graduated ones

Self_Employed and LoanAmount

```
sns.boxplot(x=df.Self_Employed,y=df.LoanAmount)
```

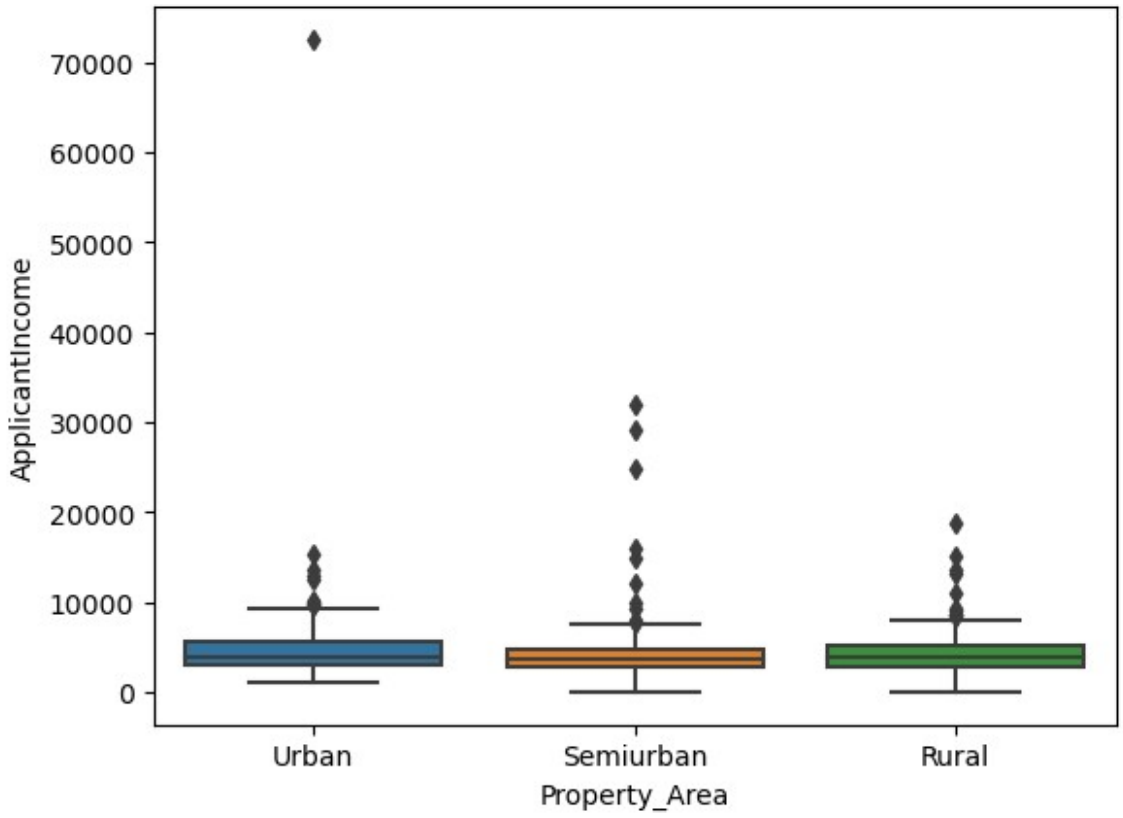
```
<Axes: xlabel='Self_Employed', ylabel='LoanAmount'>
```



Property_Area and ApplicantIncome

```
sns.boxplot(x=df.Property_Area,y=df.ApplicantIncome)
```

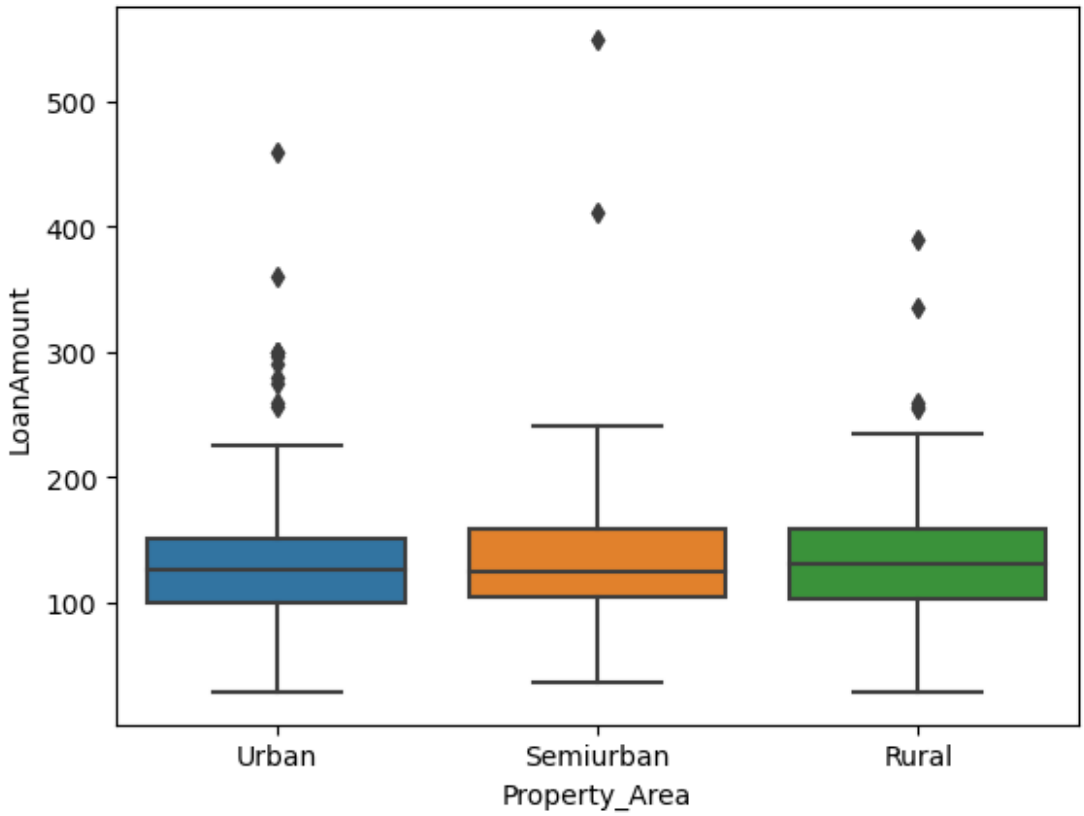
```
<Axes: xlabel='Property_Area', ylabel='ApplicantIncome'>
```



Property_Area and LoanAmount

```
sns.boxplot(x=df.Property_Area,y=df.LoanAmount)
```

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



2.3 Multivariate Analysis

Perform a correlation analysis to identify relationships between numeric variables. Visualize correlations using a heatmap.

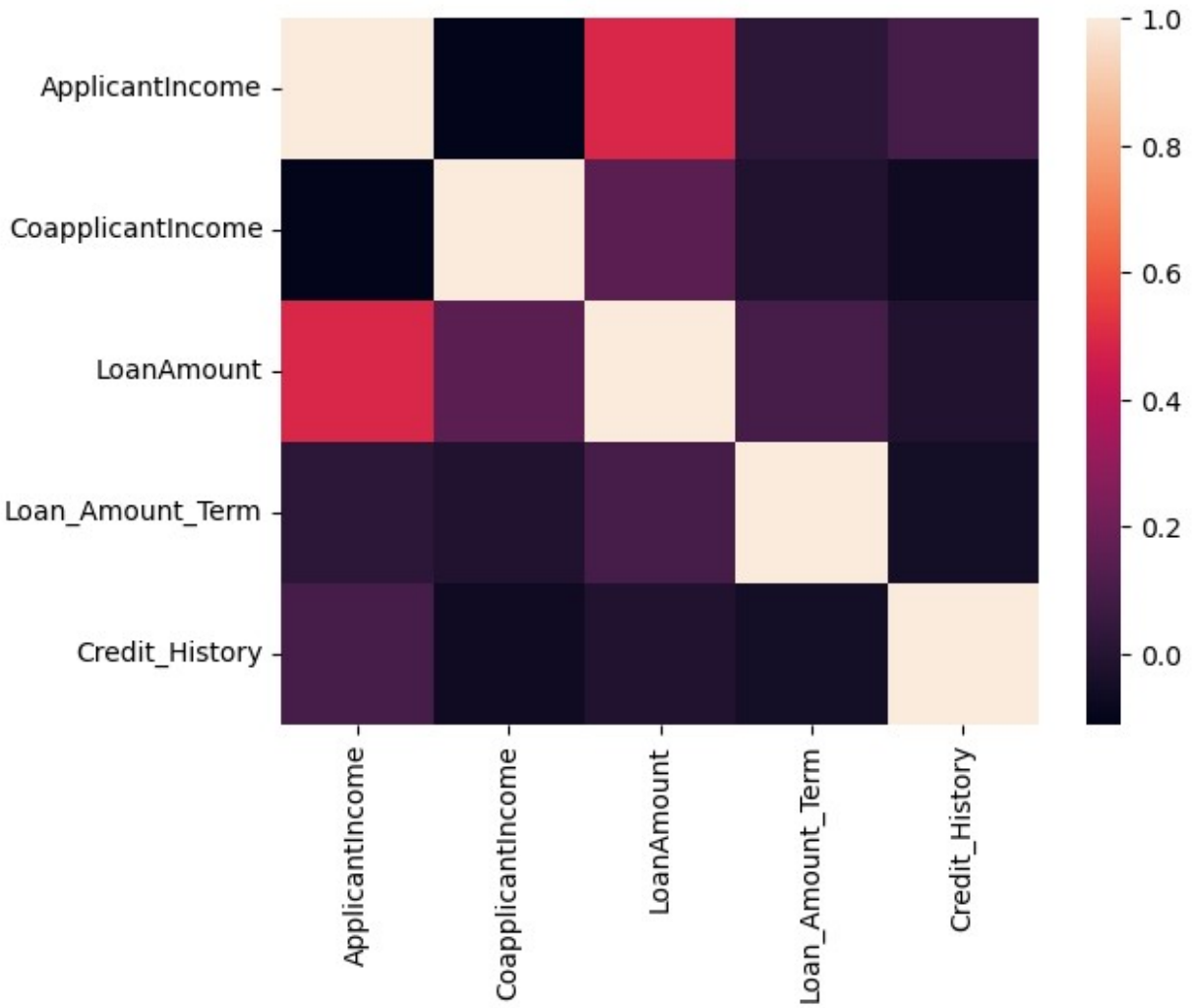
```
df.corr(numeric_only=True)
```

```
                   ApplicantIncome  CoapplicantIncome  LoanAmount  \
ApplicantIncome           1.000000          -0.110335    0.490174
CoapplicantIncome        -0.110335           1.000000    0.150112
LoanAmount                0.490174           0.150112    1.000000
Loan_Amount_Term          0.023187          -0.010940    0.093856
Credit_History            0.094083          -0.066798   -0.011405

                   Loan_Amount_Term  Credit_History
ApplicantIncome            0.023187        0.094083
CoapplicantIncome         -0.010940       -0.066798
LoanAmount                 0.093856       -0.011405
Loan_Amount_Term           1.000000       -0.052370
Credit_History            -0.052370        1.000000
```

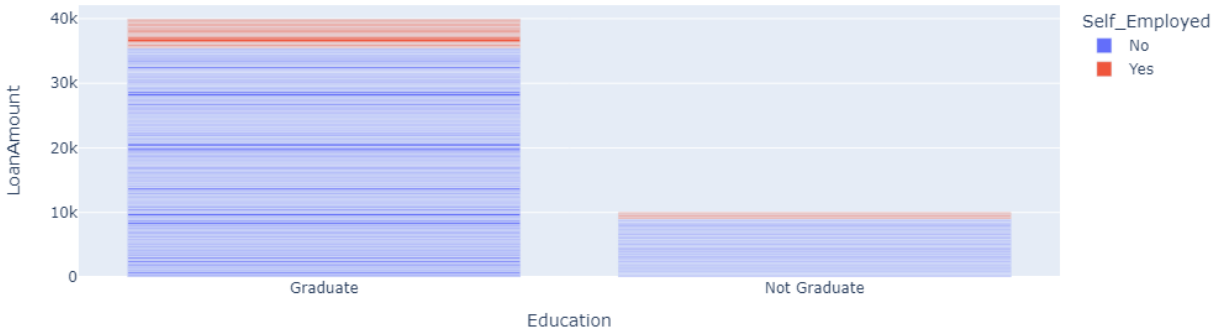
```
sns.heatmap(df.corr(numeric_only=True))
```

```
<Axes: >
```

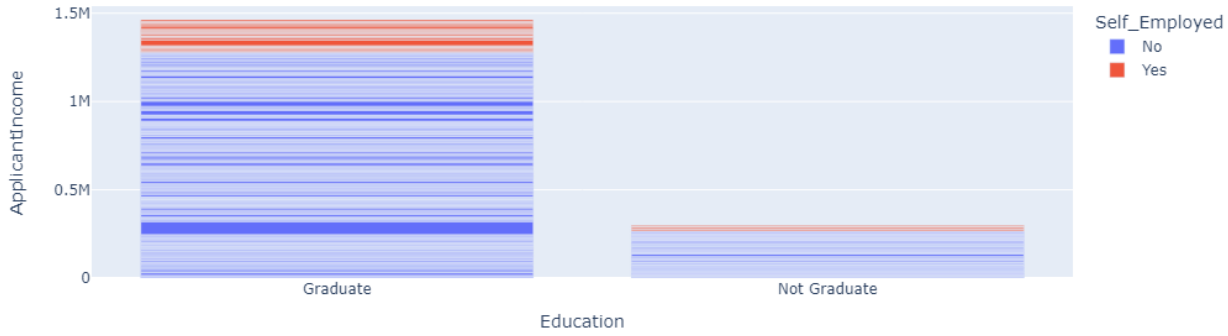


Create a stacked bar chart to show the distribution of categorical variables across multiple categories.

```
px.bar(df,x=df.Education,y=df.LoanAmount,color=df.Self_Employed)
```



```
px.bar(df,x=df.Education,y=df.ApplicantIncome,color=df.Self_Employed)
```




```
px.bar(df,x=df.Self_Employed,y=df.Credit_History,color=df.Education)
```


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

```
Loan_ID               0
Gender                0
Married               0
Dependents            0
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History        0
Property_Area         0
dtype: int64
```

Since the dataset has no null values and is clean

Saving the clean Dataset

```
df.to_csv('Loan_Approval_clean.csv')
```