In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
df=pd.read_csv("D:\\Downloads\\loandataset\\loan_data_set.csv")
df
```

Out[1]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Terr
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.
...	...	...	...	...	...	...	...	...	...	.
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.

614 rows × 13 columns

In [2]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

In [4]:

```
#Size of the dataset
df.shape
```

Out[4]:

```
(614, 13)
```

In [5]:

```
#columns in the dataset
df.columns
```

Out[5]:

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [6]:

```
#Check if the dataset has duplicates
df[df.duplicated()==True]
```

Out[6]:

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	C

In [8]:

```
#Check mising values in each column of the dataset
df.apply(lambda x:sum(x.isnull()),axis=0)
```

Out[8]:

```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [9]:

```
df['Gender'].value_counts()
```

Out[9]:

```
Gender
Male      489
Female    112
Name: count, dtype: int64
```

In [10]:

```
df.Gender= df.Gender.fillna('Female')
```

In [11]:

```
df.Married=df.Married.fillna('Yes')
```

In [12]:

```
df['Dependents'].value_counts()
```

Out[12]:

```
Dependents
0     345
1     102
2     101
3+     51
Name: count, dtype: int64
```

In [13]:

```
df.Dependents=df.Dependents.fillna('0')
```

In [14]:

```
df['Self_Employed'].value_counts()
```

Out[14]:

```
Self_Employed
No     500
Yes     82
Name: count, dtype: int64
```

In [15]:

```
df.Self_Employed=df.Self_Employed.fillna('No')
```

In [16]:

```
df.LoanAmount=df.LoanAmount.fillna(df.LoanAmount.mean())
```

In [17]:

```
df.Self_Employed=df.Self_Employed.fillna(df.Self_Employed.mode())
```

In [18]:

```
df['Loan_Amount_Term'].value_counts()
```

Out[18]:

```
Loan_Amount_Term
360.0    512
180.0     44
480.0     15
300.0     13
240.0      4
84.0       4
120.0      3
60.0       2
36.0       2
12.0       1
Name: count, dtype: int64
```

In [19]:

```
df.Loan_Amount_Term=df.Loan_Amount_Term.fillna(360.0)
```

In [20]:

```
df['Credit_History'].value_counts()
```

Out[20]:

```
Credit_History
1.0    475
0.0     89
Name: count, dtype: int64
```

In [21]:

```
df.Credit_History=df.Credit_History.fillna(0.0)
```

In [22]:

```
#Check mising values in each column of the dataset
df.apply(lambda x:sum(x.isnull()),axis=0)
```

Out[22]:

```
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
Loan_Status          0
dtype: int64
```

In [23]:

```
#Distribution of Loan Status,Gender and Other categorcal features
df['Loan_Status'].value_counts()
```

Out[23]:

```
Loan_Status
Y    422
N    192
Name: count, dtype: int64
```

In [24]:

```
df['Loan_Status'].value_counts(normalize=True)*100
```

Out[24]:

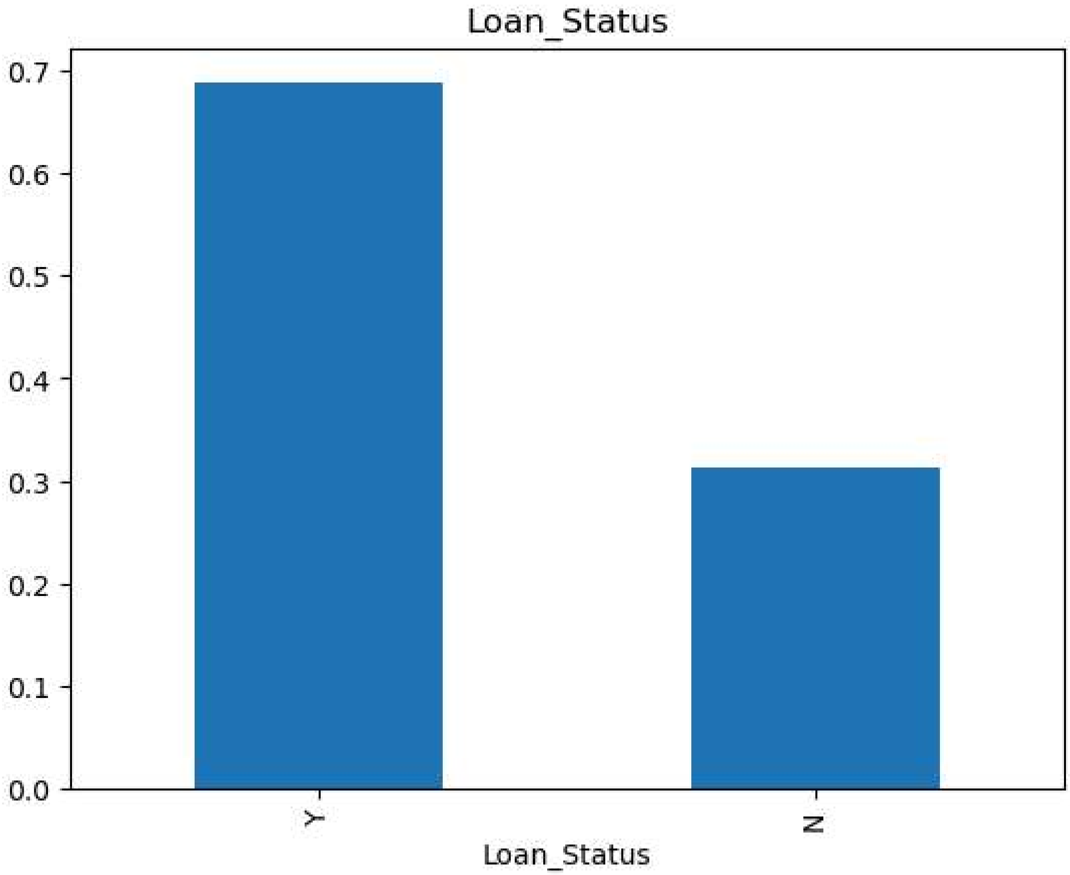
```
Loan_Status
Y    68.729642
N    31.270358
Name: proportion, dtype: float64
```

In [25]:

```
df['Loan_Status'].value_counts(normalize=True).plot.bar(title='Loan_Status')
```

Out[25]:

```
<Axes: title={'center': 'Loan_Status'}, xlabel='Loan_Status'>
```

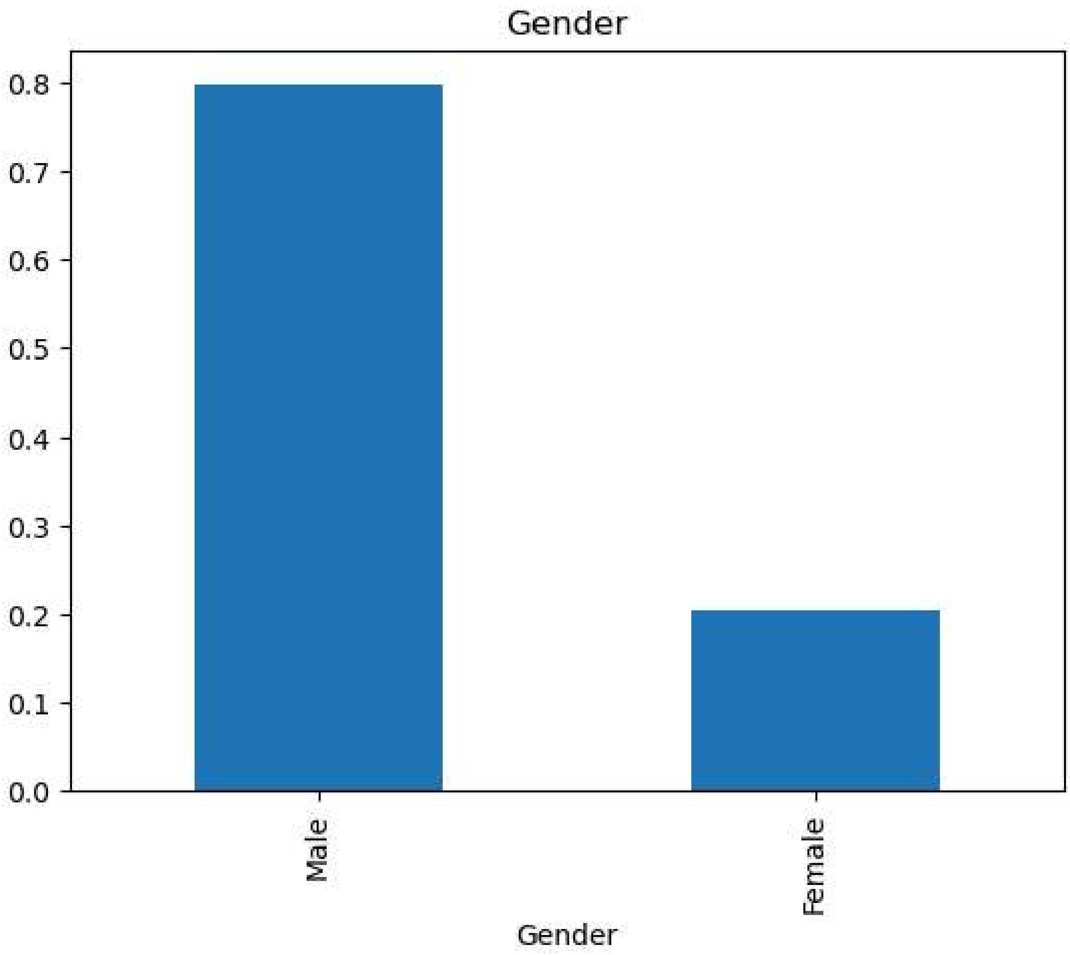


In [26]:
```
df['Gender'].value_counts(normalize=True)*100
```

Out[26]:
```
Gender
Male      79.641694
Female    20.358306
Name: proportion, dtype: float64
```

In [27]:
```
df['Gender'].value_counts(normalize=True).plot.bar(title='Gender')
```

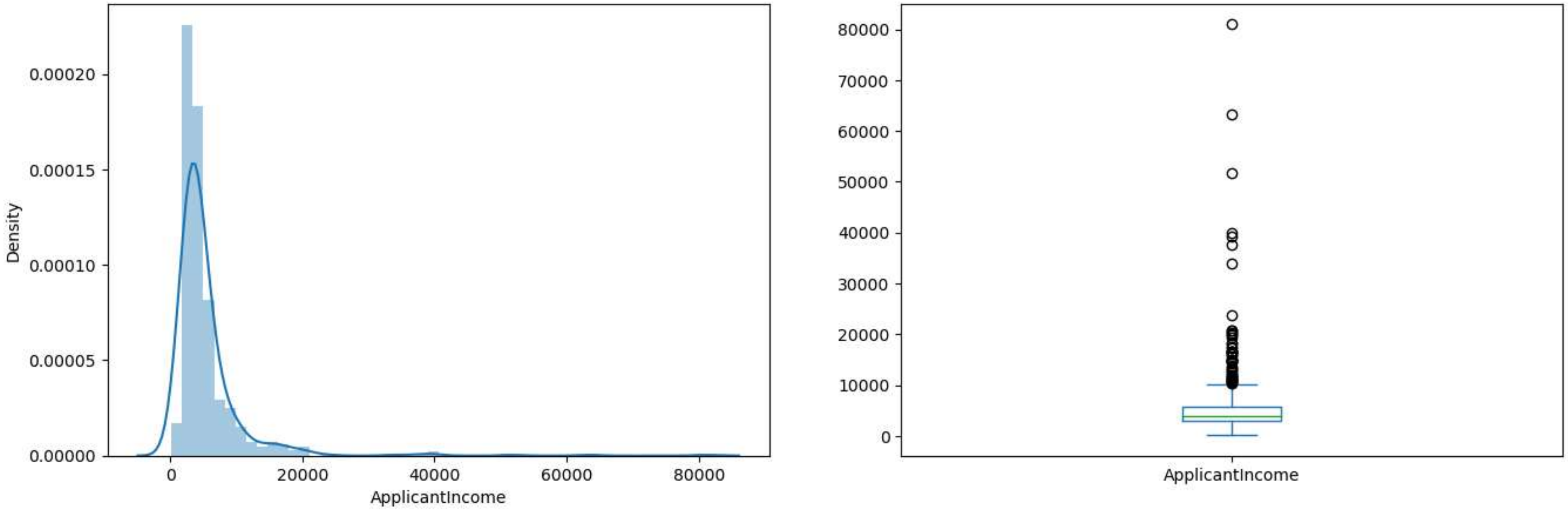
Out[27]:
```
<Axes: title={'center': 'Gender'}, xlabel='Gender'>
```



In [28]:

```
#Univariant Analysis --- Applicant Income, Co Applicant Income and Loan Amount
plt.figure(1)
plt.subplot(121)
sns.distplot(df['ApplicantIncome'])
plt.subplot(122)
df['ApplicantIncome'].plot.box(figsize=(16,5))
plt.show()
```

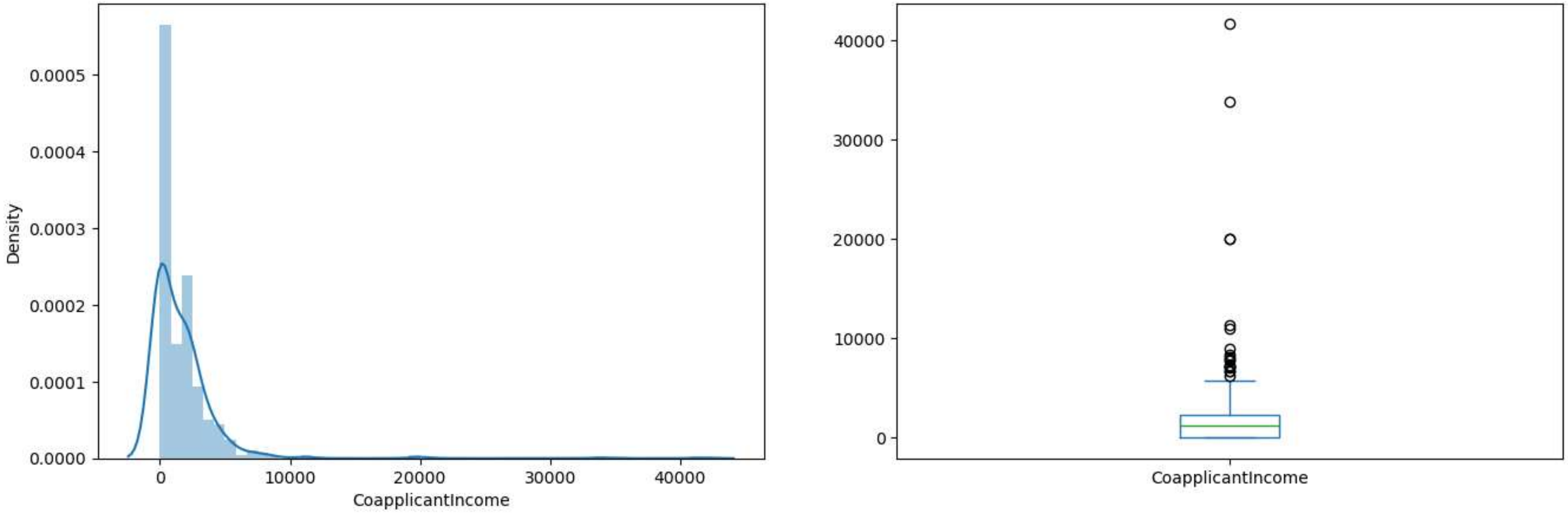
```
C:\Users\UD SYSTEMS\AppData\Local\Temp\ipykernel_8220\1620415607.py:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['ApplicantIncome'])
```



In [29]:

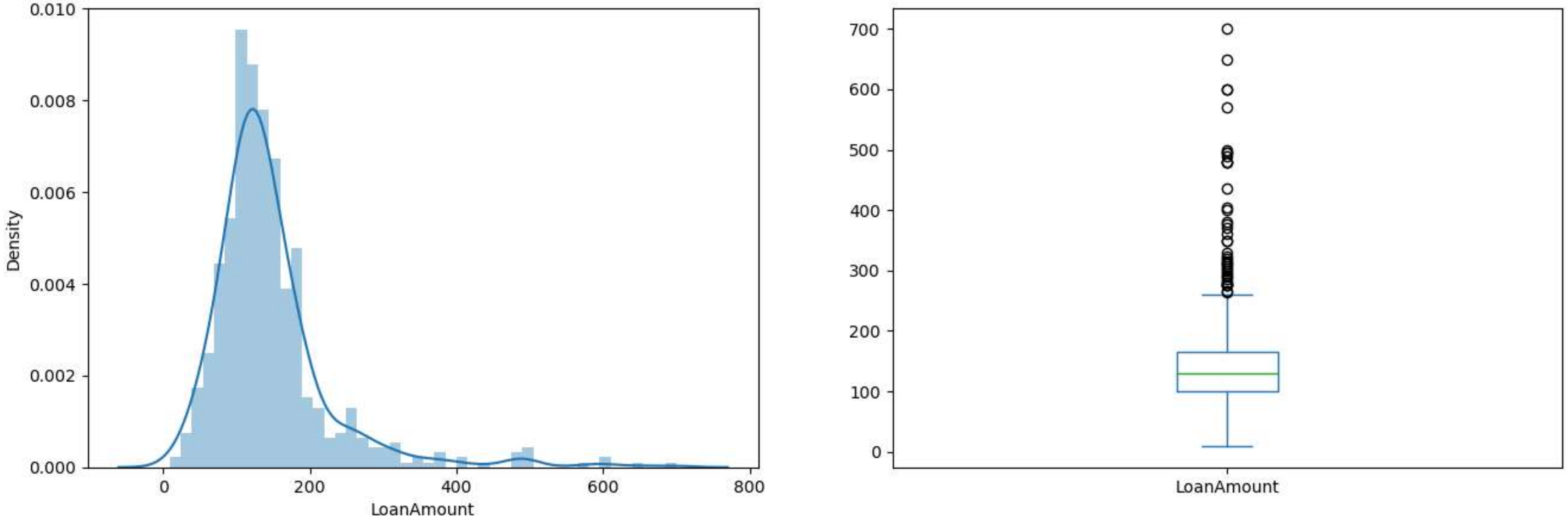
```
plt.figure(1)
plt.subplot(121)
sns.distplot(df['CoapplicantIncome'])
plt.subplot(122)
df['CoapplicantIncome'].plot.box(figsize=(16,5))
plt.show()
```

```
C:\Users\UD SYSTEMS\AppData\Local\Temp\ipykernel_8220\795179466.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['CoapplicantIncome'])
```



In [30]:
```
plt.figure(1)
plt.subplot(121)
sns.distplot(df['LoanAmount'])
plt.subplot(122)
df['LoanAmount'].plot.box(figsize=(16,5))
plt.show()
```

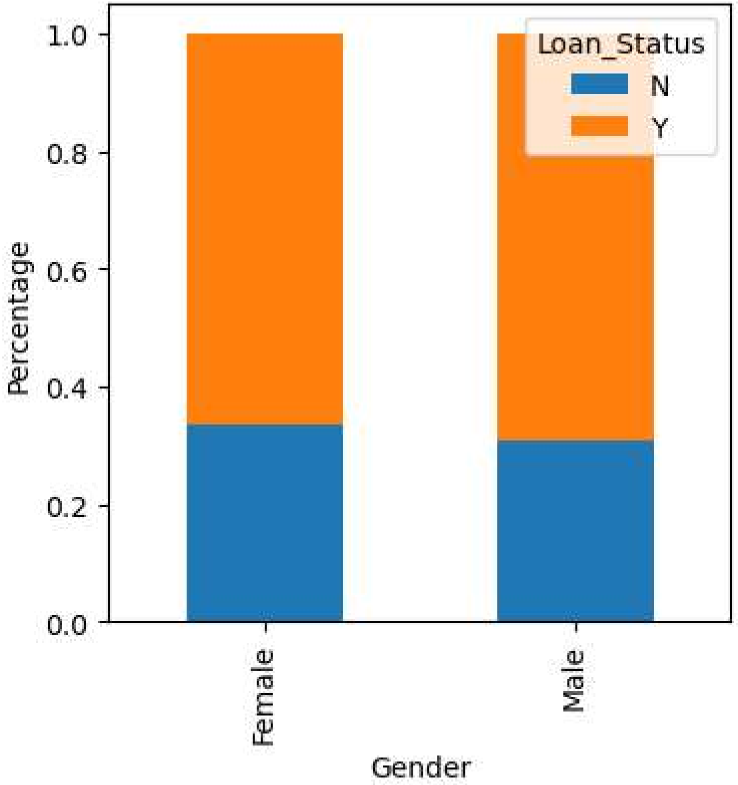
```
C:\Users\UD SYSTEMS\AppData\Local\Temp\ipykernel_8220\64810118.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['LoanAmount'])
```



In [32]:

```
#Bivariant Analysis
#Loan Stauts Vs Gender
print(pd.crosstab(df['Gender'],df['Loan_Status']))
Gender=pd.crosstab(df['Gender'],df['Loan_Status'])
Gender.div(Gender.sum(1).astype(float),axis=0).plot(kind="bar",stacked=True,figsize=(4,4))
plt.xlabel('Gender')
plt.ylabel('Percentage')
plt.show()
```

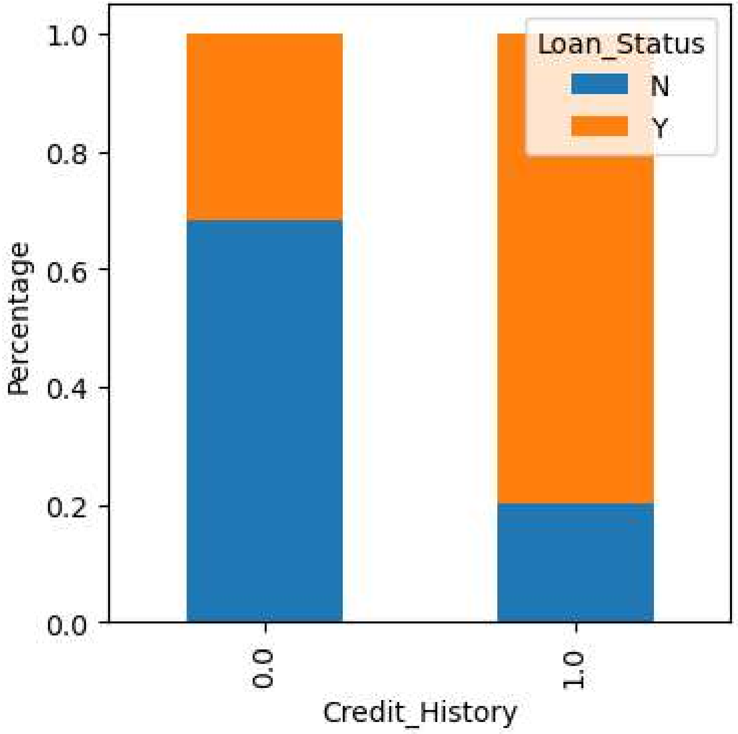
```
Loan_Status    N    Y
Gender
Female        42   83
Male         150  339
```



In [33]:

```
#Loan Stauts Vs Credit History
print(pd.crosstab(df['Credit_History'],df['Loan_Status']))
Gender=pd.crosstab(df['Credit_History'],df['Loan_Status'])
Gender.div(Gender.sum(1).astype(float),axis=0).plot(kind="bar",stacked=True,figsize=(4,4))
plt.xlabel('Credit_History')
plt.ylabel('Percentage')
plt.show()
```

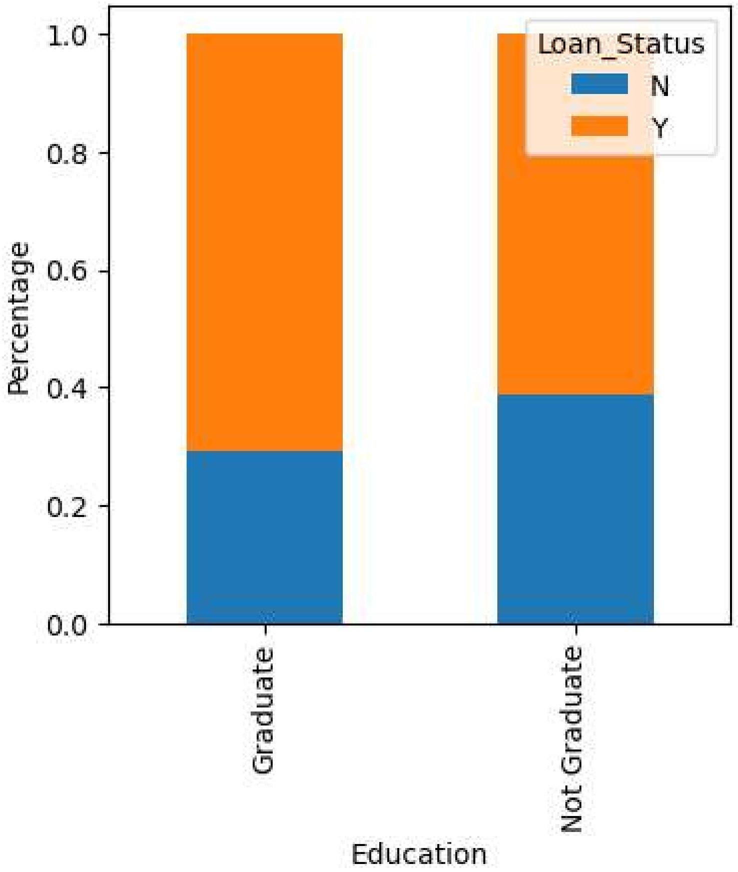
```
Loan_Status      N    Y
Credit_History         
0.0             95   44
1.0             97  378
```



In [34]:

```
#Loan Status Vs Education
print(pd.crosstab(df['Education'],df['Loan_Status']))
Gender=pd.crosstab(df['Education'],df['Loan_Status'])
Gender.div(Gender.sum(1).astype(float),axis=0).plot(kind="bar",stacked=True,figsize=(4,4))
plt.xlabel('Education')
plt.ylabel('Percentage')
plt.show()
```

```
Loan_Status     N    Y
Education
Graduate      140  340
Not Graduate   52   82
```



In [35]:

```
#Numerical Variable vs Target Variable Distribution
bins=[0,2500,5000,6000,81000]
group=['Low','Average','High','Very high']

df['Income_bin']=pd.cut(df['ApplicantIncome'],bins,labels=group)
df.Income_bin
```

Out[35]:

```
0         High
1      Average
2      Average
3      Average
4         High
        ...   
609    Average
610    Average
611  Very high
612  Very high
613    Average
Name: Income_bin, Length: 614, dtype: category
Categories (4, object): ['Low' < 'Average' < 'High' < 'Very high']
```

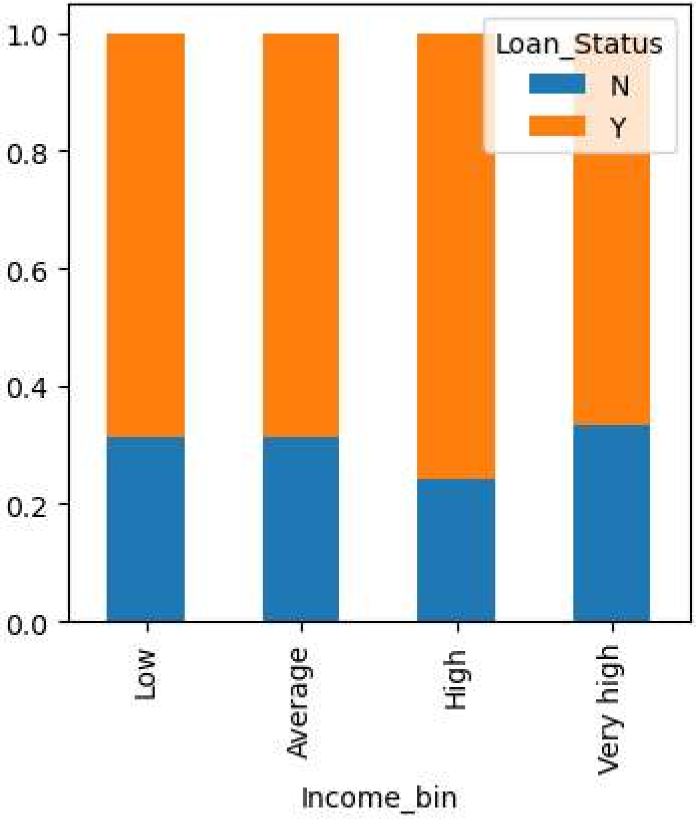
In [38]:

```
print(pd.crosstab(df['Income_bin'],df['Loan_Status']))
Income_bin=pd.crosstab(df['Income_bin'],df['Loan_Status'])
Income_bin.div(Income_bin.sum(1).astype(float),axis=0).plot(kind='bar',stacked=True,figsize=(4,4))
```

```
Loan_Status   N    Y
Income_bin          
Low          34   74
Average      99  216
High         13   41
Very high    46   91
```

Out[38]:

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


1.0
0.8
0.6
0.4
0.2
0.0
Loan_Status
N
Y
Low
Average
High
Very high
Income_bin

```
In [ ]:
```