# Importing Dependencies

In [30]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings("ignore")
```

# Loading the dataset

In [31]:
```
# Loading the dataset to Pandas DataFrame
loan_data= pd.read_csv("loandata.csv")
```

In [32]:
```
# Printing the first 5 rows of the dataset
loan_data.head()
```

Out[32]:

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | Coapplica |
|---|---|---|---|---|---|---|---|---|
| **0** | LP001002 | Male | No | 0 | Graduate | No | 5849 | |
| **1** | LP001003 | Male | Yes | 1 | Graduate | No | 4583 | |
| **2** | LP001005 | Male | Yes | 0 | Graduate | Yes | 3000 | |
| **3** | LP001006 | Male | Yes | 0 | Not Graduate | No | 2583 | |
| **4** | LP001008 | Male | No | 0 | Graduate | No | 6000 | |

In [33]:
```
# Number of rows and colums
loan_data.shape
```

Out[33]:
```
(614, 13)
```

In [34]:
```
# statistical measures
loan_data.describe()
```

Out[34]:

| | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_History |
|---|---|---|---|---|---|
| **count** | 614.000000 | 614.000000 | 592.000000 | 600.00000 | 564.000000 |
| **mean** | 5403.459283 | 1621.245798 | 146.412162 | 342.00000 | 0.842199 |
| **std** | 6109.041673 | 2926.248369 | 85.587325 | 65.12041 | 0.364878 |
| **min** | 150.000000 | 0.000000 | 9.000000 | 12.00000 | 0.000000 |
| **25%** | 2877.500000 | 0.000000 | 100.000000 | 360.00000 | 1.000000 |
| **50%** | 3812.500000 | 1188.500000 | 128.000000 | 360.00000 | 1.000000 |
| **75%** | 5795.000000 | 2297.250000 | 168.000000 | 360.00000 | 1.000000 |
| **max** | 81000.000000 | 41667.000000 | 700.000000 | 480.00000 | 1.000000 |

In [35]:
```
loan_data.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
```

# Missing value imputation

let's list out the feature-wise count of missing values.

In [36]:
```
loan_data.isnull().sum()
```

Out[36]:
```
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 [37]:
```
# fill the missing values for numerical terms - mean
loan_data['LoanAmount'] = loan_data['LoanAmount'].fillna(loan_data['LoanAmount'].me
loan_data['Loan_Amount_Term'] = loan_data['Loan_Amount_Term'].fillna(loan_data['Loa
loan_data['Credit_History'] = loan_data['Credit_History'].fillna(loan_data['Credit_
```

In [38]:
```
# fill the missing values for categorical terms - mode
loan_data['Gender'] = loan_data["Gender"].fillna(loan_data['Gender'].mode()[0])
loan_data['Married'] = loan_data["Married"].fillna(loan_data['Married'].mode()[0])
loan_data['Dependents'] = loan_data["Dependents"].fillna(loan_data['Dependents'].mo
loan_data['Self_Employed'] = loan_data["Self_Employed"].fillna(loan_data['Self_Empl
```

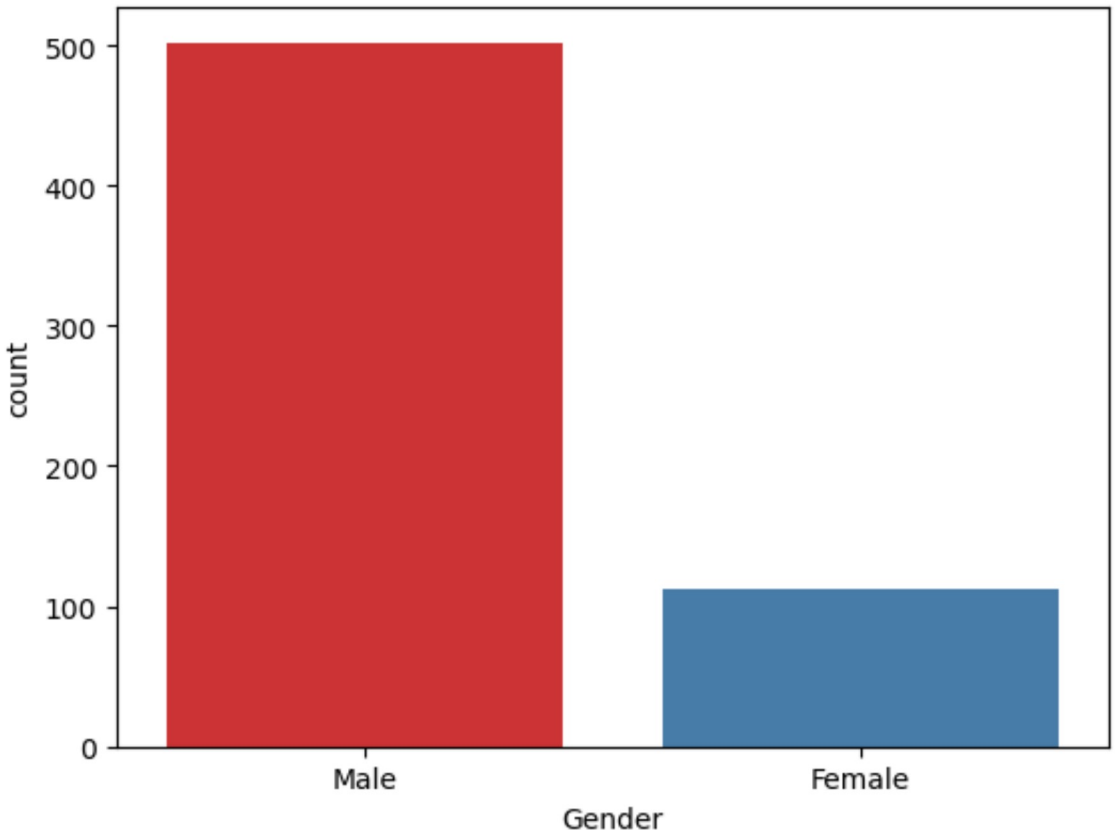
In [39]:
```
# number of missing values in each column
loan_data.isnull().sum()
```

Out[39]:
```
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
```

# Exploratory Data Analysis

In [40]:
```
# categorical attributes visualization
print(loan_data['Gender'].value_counts())
sns.countplot(x='Gender',data=loan_data,palette='Set1')
```

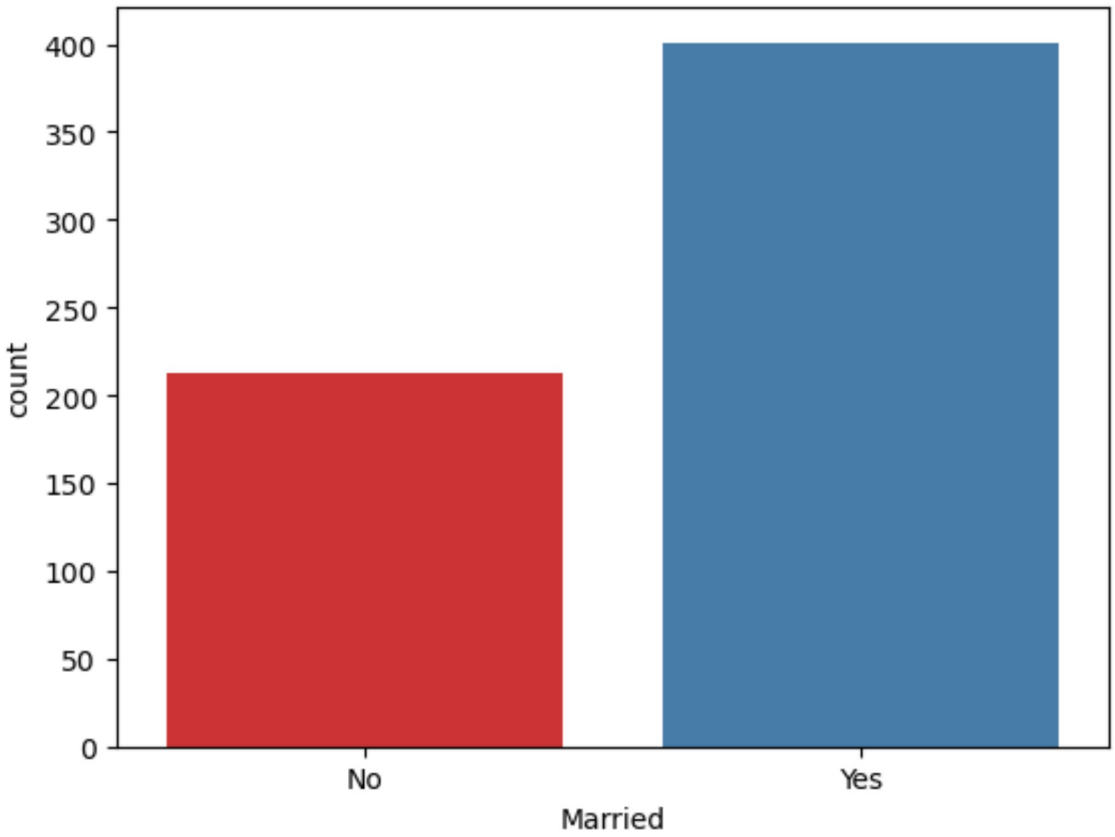
Out[40]:
```
Male      502
Female    112
Name: Gender, dtype: int64
<Axes: xlabel='Gender', ylabel='count'>
```




In [41]:
```
print(loan_data['Married'].value_counts())
sns.countplot(x='Married',data=loan_data,palette='Set1')
```

```
Yes    401
No     213
Name: Married, dtype: int64
```

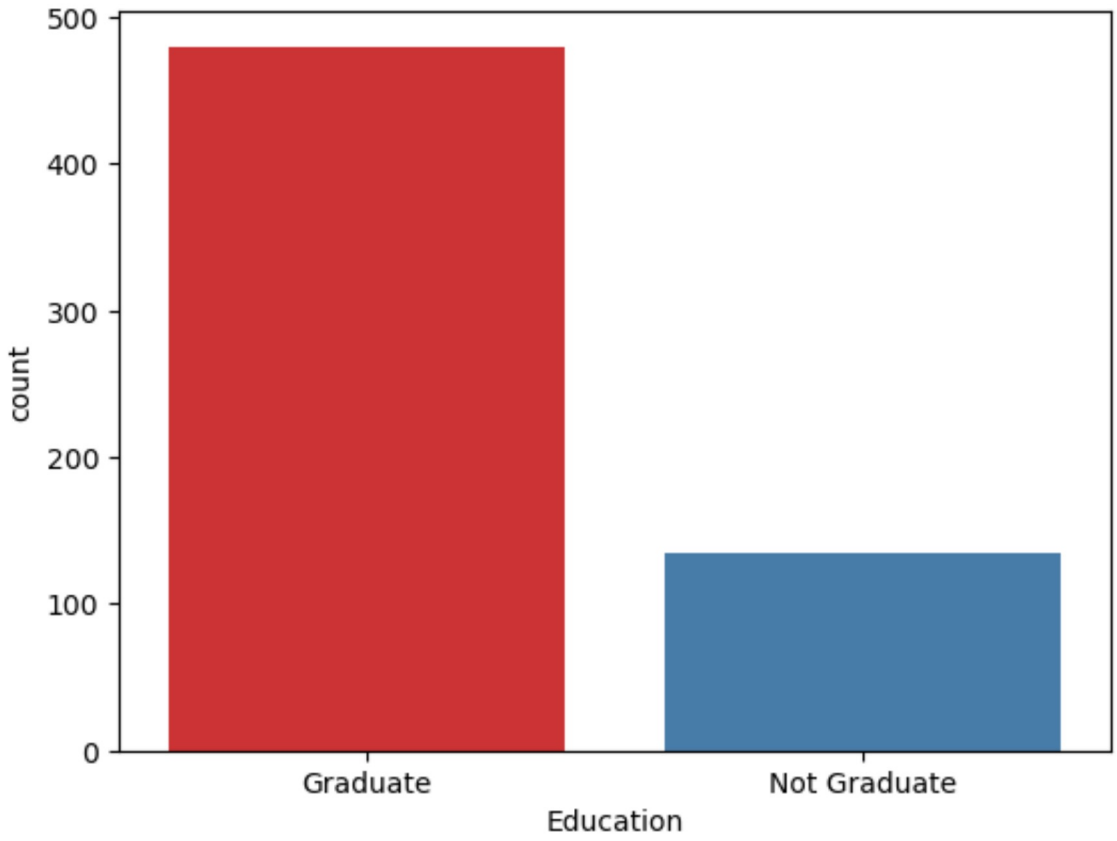
Out[41]:
```
<Axes: xlabel='Married', ylabel='count'>
```




In [42]:
```
print(loan_data['Education'].value_counts())
sns.countplot(x='Education',data=loan_data,palette='Set1')
```

```
Graduate        480
Not Graduate    134
Name: Education, dtype: int64
```

Out[42]: <Axes: xlabel='Education', ylabel='count'>




In [43]:
```
# Label encoding : Converting categorical Loan Status column to numerical values.
loan_data.replace({"Loan_Status":{'N':0,'Y':1}},inplace=True)
```

In [44]:
```
# printing the first 5 rows of the dataframe
loan_data.head()
```

Out[44]:

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | Coapplica |
|---|---|---|---|---|---|---|---|---|
| **0** | LP001002 | Male | No | 0 | Graduate | No | 5849 | |
| **1** | LP001003 | Male | Yes | 1 | Graduate | No | 4583 | |
| **2** | LP001005 | Male | Yes | 0 | Graduate | Yes | 3000 | |
| **3** | LP001006 | Male | Yes | 0 | Not Graduate | No | 2583 | |
| **4** | LP001008 | Male | No | 0 | Graduate | No | 6000 | |

In [45]:
```
# Dependent column values
loan_data['Dependents'].value_counts()
```

Out[45]:
```
0     360
1     102
2     101
3+     51
Name: Dependents, dtype: int64
```

In [46]:
```
# 3+ is not a good data type so we cannot feed this 3+ value to our model.So we wil
loan_data = loan_data.replace(to_replace='3+', value=4)
```

In [47]:
```
# dependent values
loan_data['Dependents'].value_counts()
```

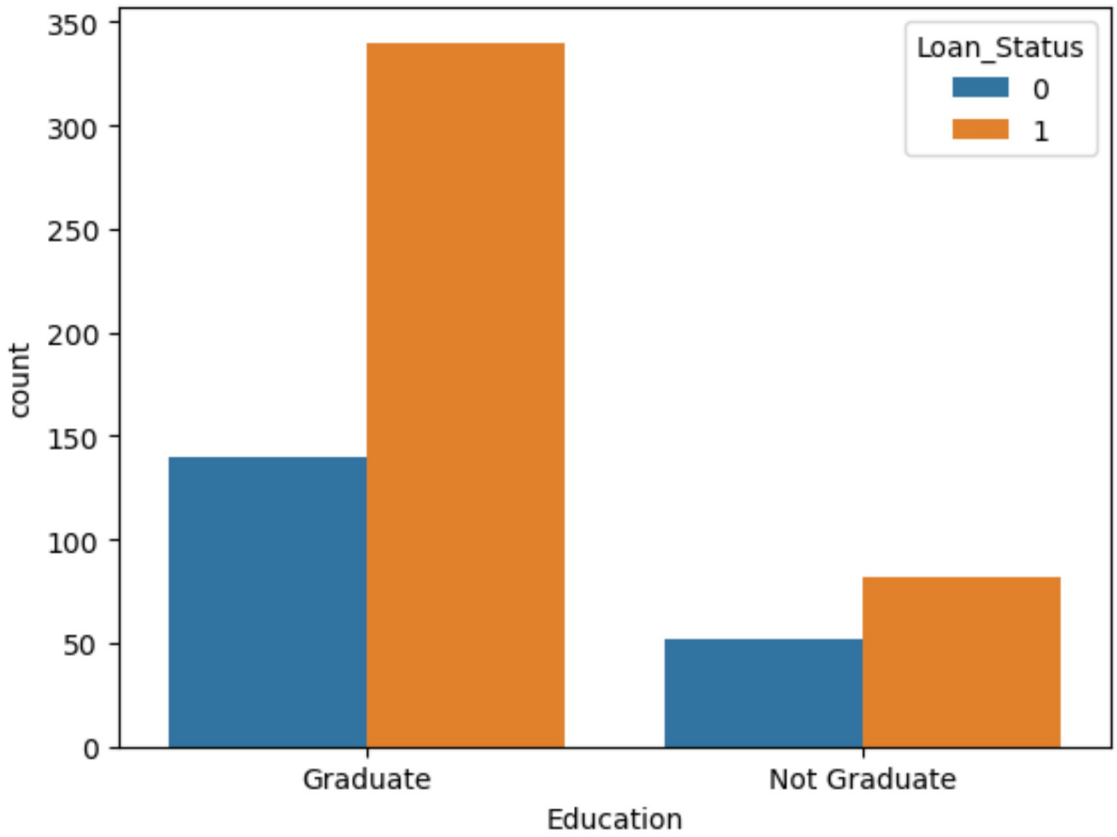
Out[47]:
```
0    360
1    102
2    101
4     51
Name: Dependents, dtype: int64
```

# Bivariate Analysis

Categorical Independent Variable vs Target Variable

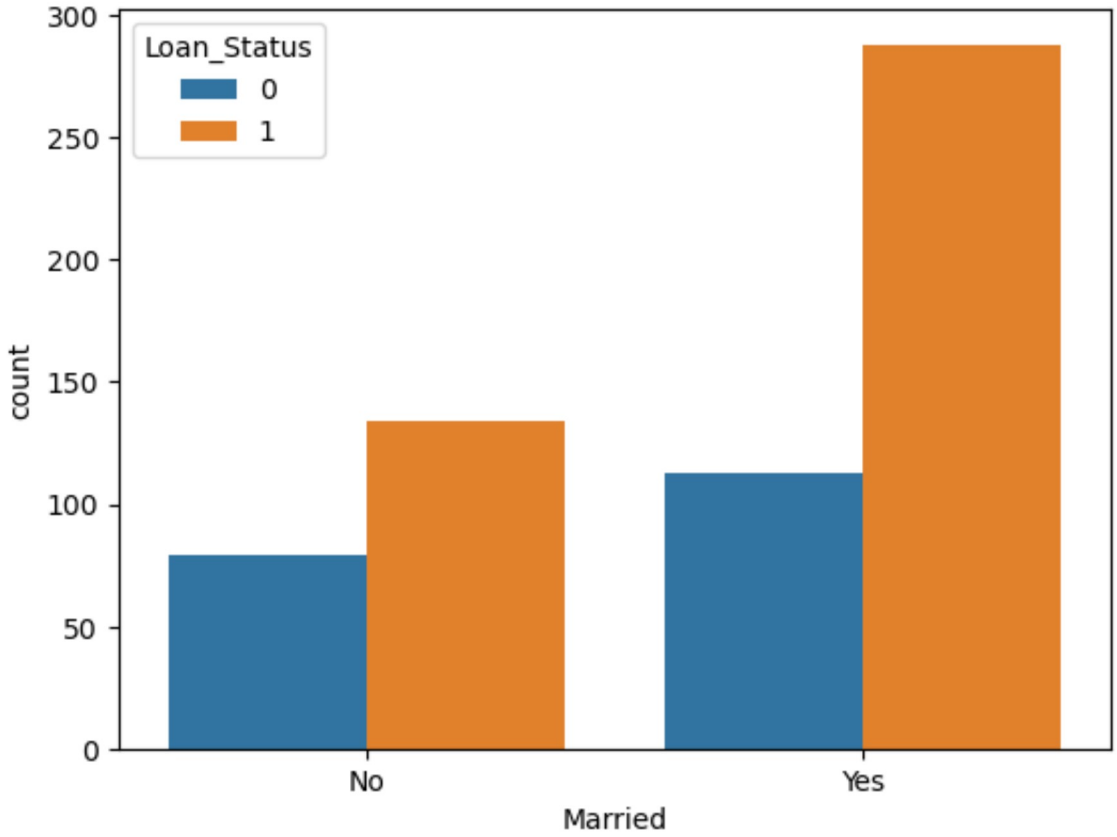
In [48]:
```
# education & Loan Status
sns.countplot(x='Education',hue='Loan_Status',data=loan_data)
```

Out[48]:
```
<Axes: xlabel='Education', ylabel='count'>
```



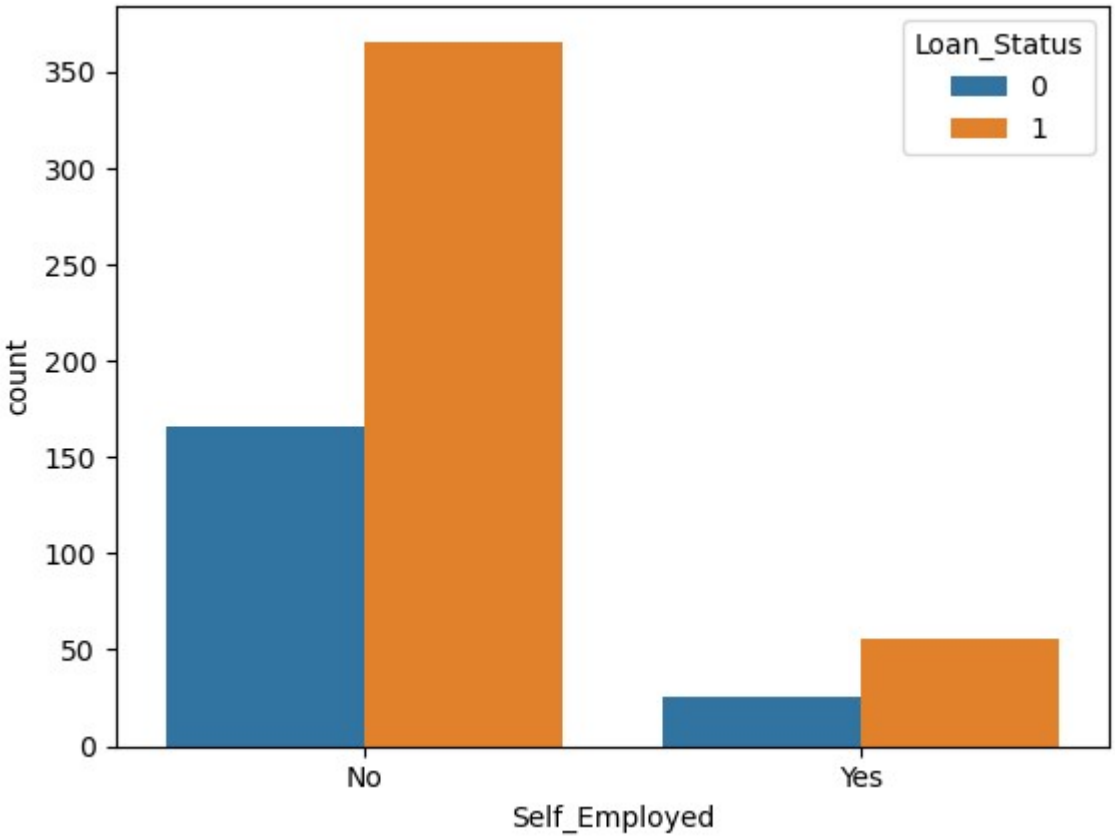

In [49]:
```
# marital status & Loan Status
sns.countplot(x='Married',hue='Loan_Status',data=loan_data)
```

Out[49]:
```
<Axes: xlabel='Married', ylabel='count'>
```



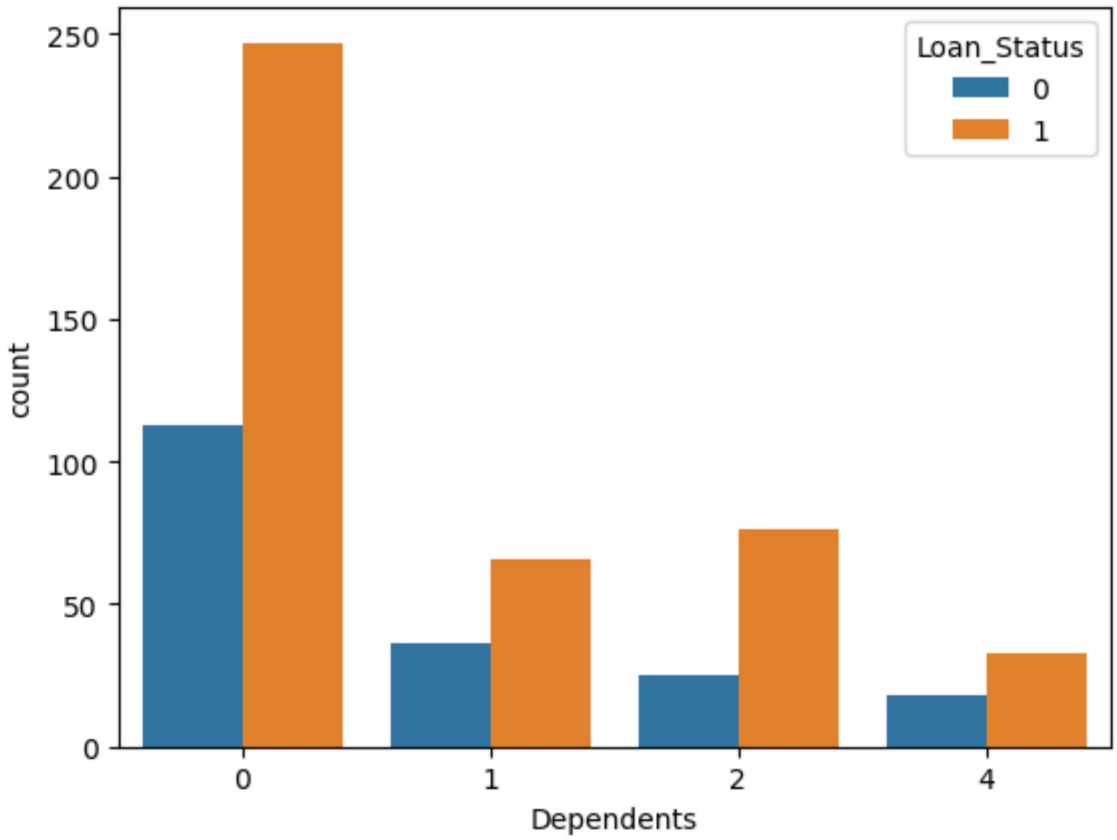

In [50]:
```
# Self Employed & Loan Status
sns.countplot(x='Self_Employed',hue='Loan_Status',data=loan_data)
```

Out[50]:
```
<Axes: xlabel='Self_Employed', ylabel='count'>
```


Loan_Status
0
1
350
300
250
200
150
100
50
0
count
No
Yes
Self_Employed

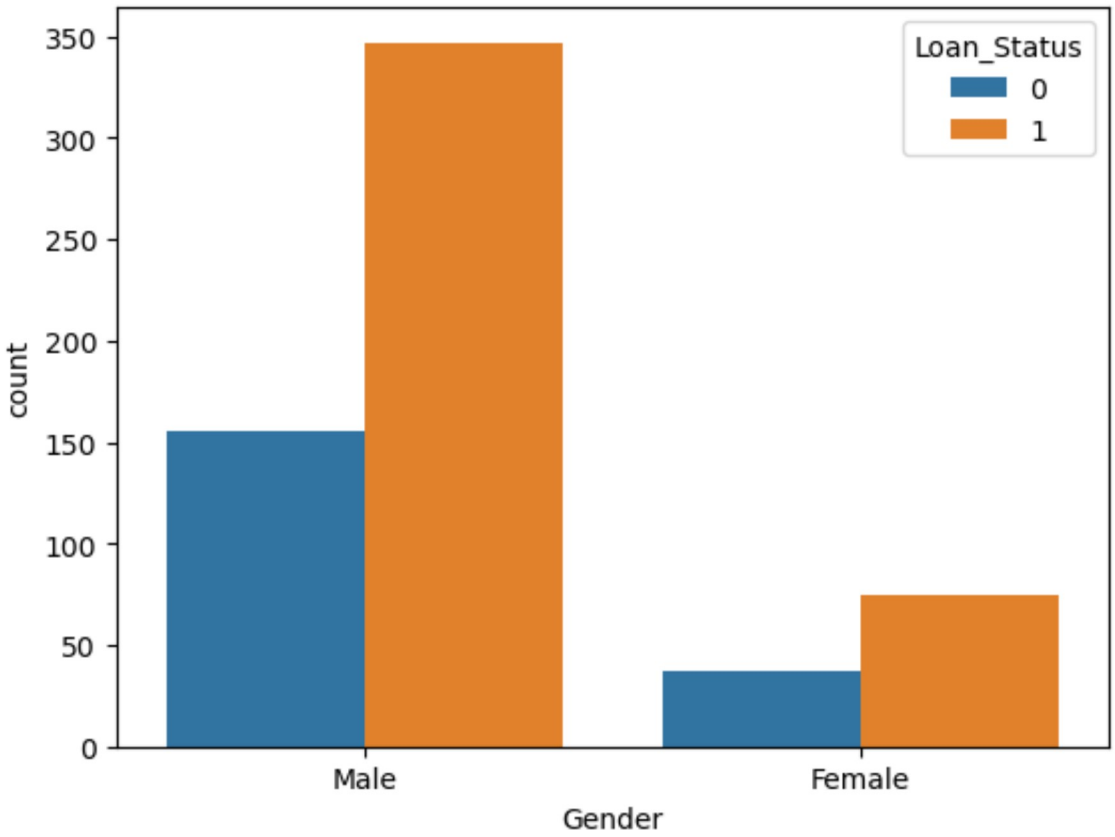
In [51]:
```
# education & Loan Status
sns.countplot(x='Dependents',hue='Loan_Status',data=loan_data)
```

Out[51]:
```
<Axes: xlabel='Dependents', ylabel='count'>
```




In [52]:
```
# education & Loan Status
sns.countplot(x='Gender',hue='Loan_Status',data=loan_data)
```

Out[52]:
```
<Axes: xlabel='Gender', ylabel='count'>
```

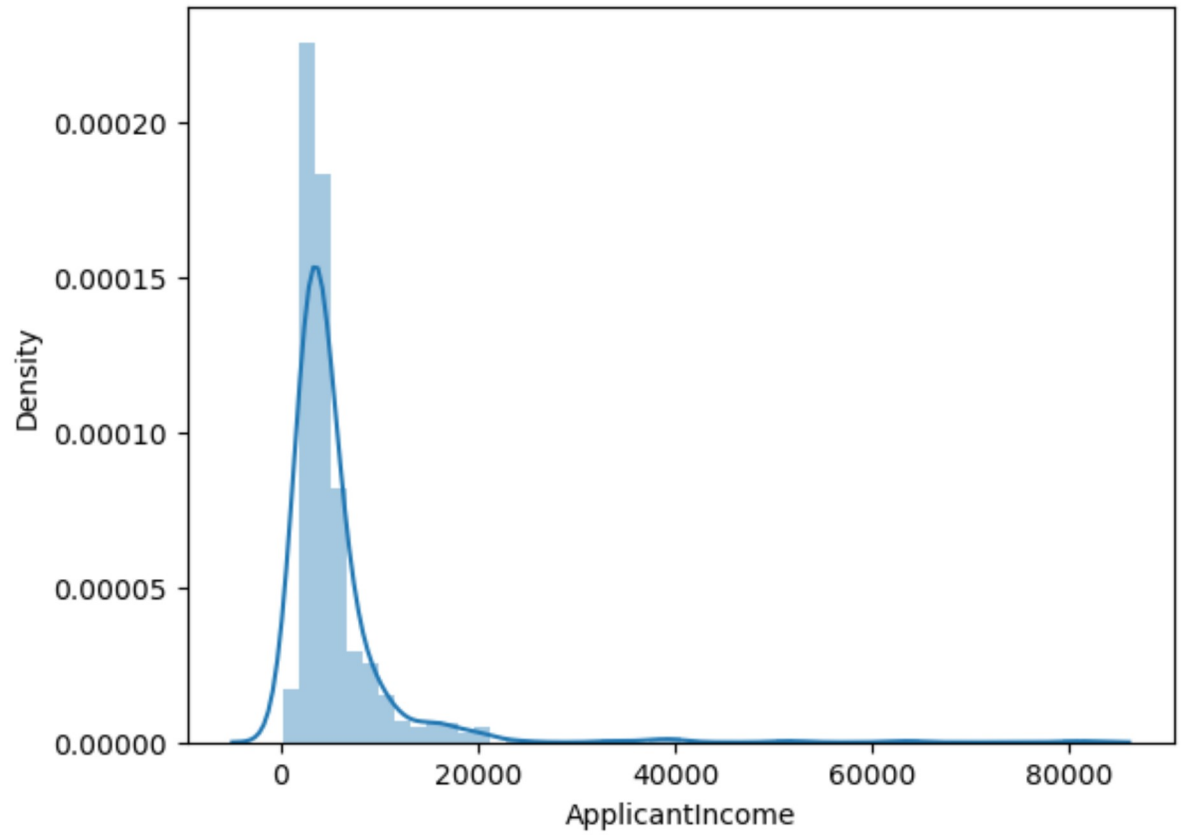


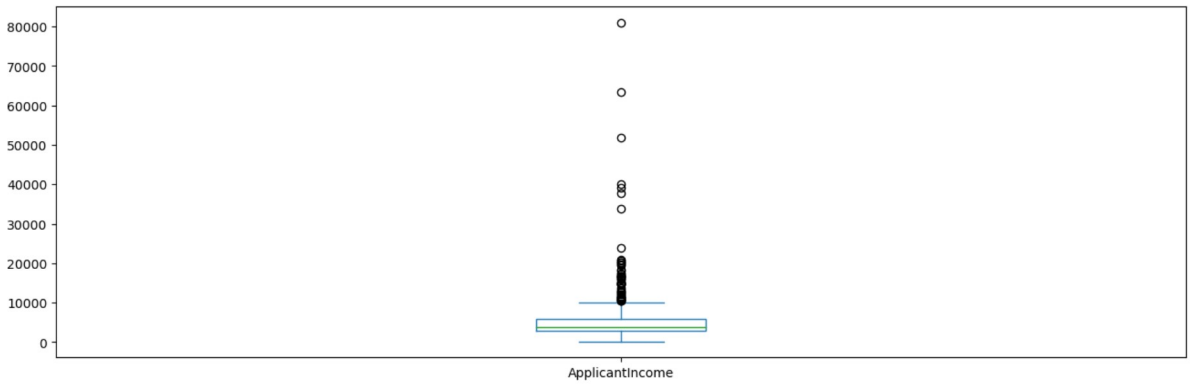

# Independent Variable (Numerical)

Till now we have seen the categorical and ordinal variables and now lets visualize the numerical variables. Lets look at the distribution of Applicant income first.

In [53]:
```
# Creating a distribution plot to visualize the distribution of applicant incomes i
# using the seaborn library's distplot() function\n",
sns.distplot(loan_data['ApplicantIncome'])
plt.show()


#Creating a box plot to visualize the distribution of applicant incomes in the 'loa
# using the pandas library's plot.box() function
# and setting the figure size to 16x5 using the figsize parameter
loan_data['ApplicantIncome'].plot.box(figsize=(16,5))
plt.show()
```



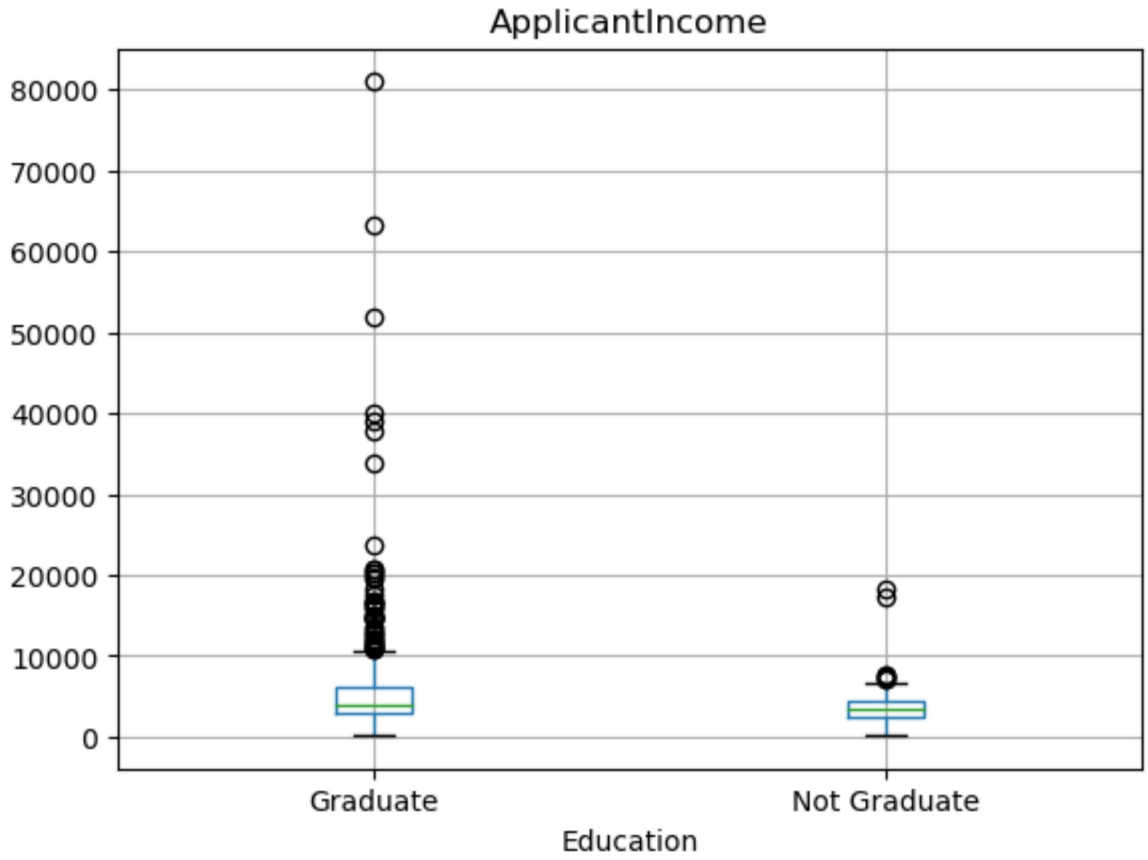




It can be inferred that most of the data in the distribution of applicant income is towards left which means it is not normally distributed. We will try to make it normal in later sections as algorithms works better if the data is normally distributed.

The boxplot confirms the presence of a lot of outliers/extreme values. This can be attributed to the income disparity in the society. Part of this can be driven by the fact that we are looking at people with different eduation levels. Let us segregate them by Education.

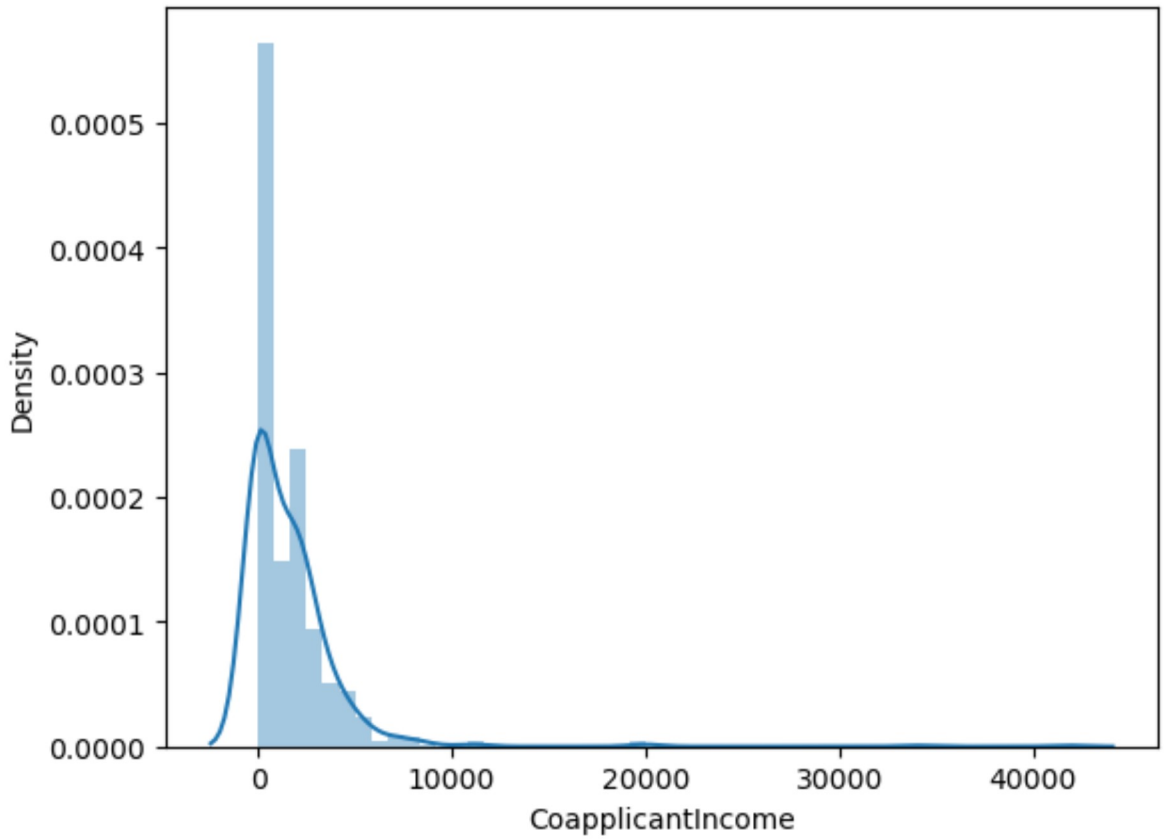
In [54]:
```
# Creating a box plot to visualize the distribution of applicant incomes in the 'lo
# based on the 'Education' column\n",
# using the pandas library's boxplot() function\n",
loan_data.boxplot(column='ApplicantIncome', by='Education')
plt.suptitle("")
```

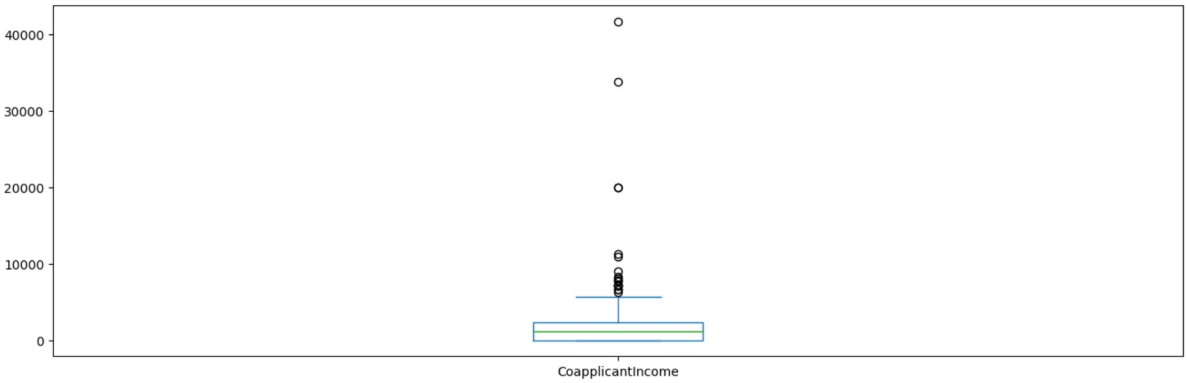
Out[54]:
```
Text(0.5, 0.98, '')
```




We can see that there are a higher number of graduates with very high incomes, which are appearing to be outliers.

In [55]:
```
#Let's look at the Coapplicant income distribution.
sns.distplot(loan_data['CoapplicantIncome'])
plt.show()
loan_data['CoapplicantIncome'].plot.box(figsize=(16,5))
plt.show()
```

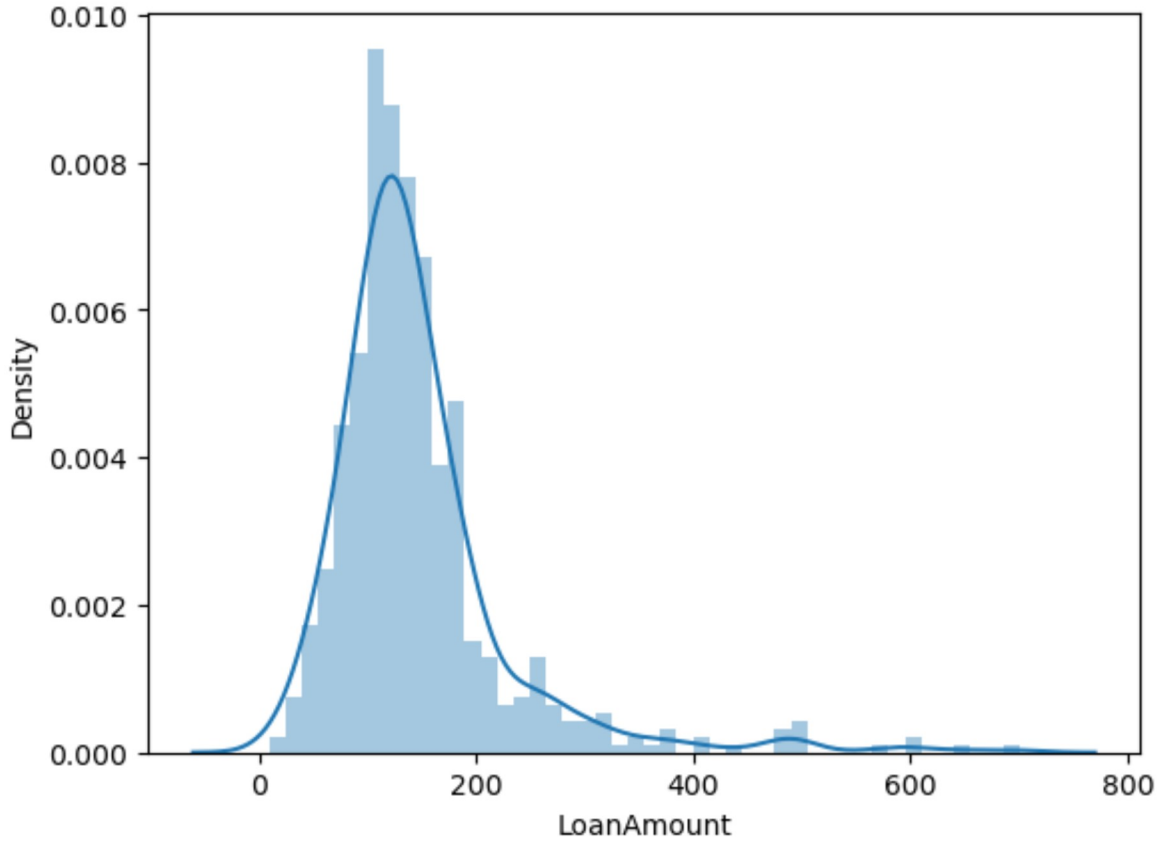


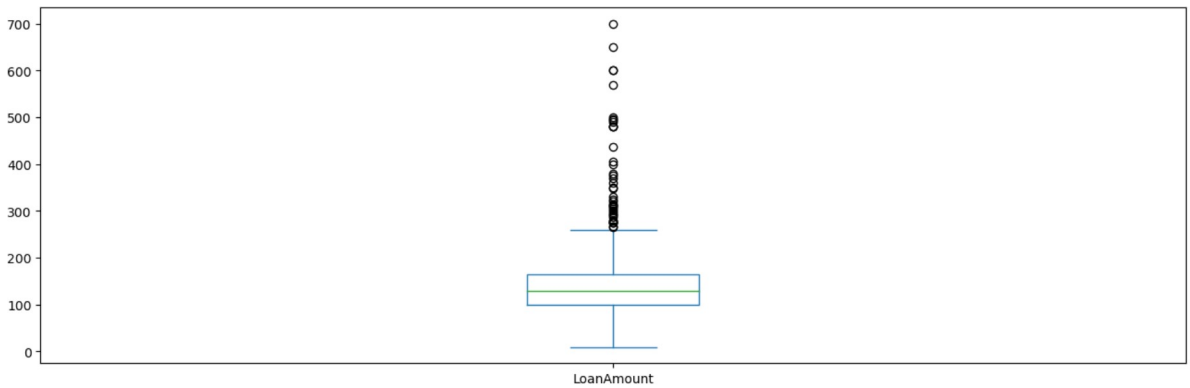




We see a similar distribution as that of the applicant's income. The majority of co-applicants income ranges from 0 to 5000. We also see a lot of outliers in the applicant's income and it is not normally distributed.

In [56]:
```
sns.distplot(loan_data['LoanAmount'])
plt.show()

loan_data['LoanAmount'].plot.box(figsize=(16,5))
plt.show()
```



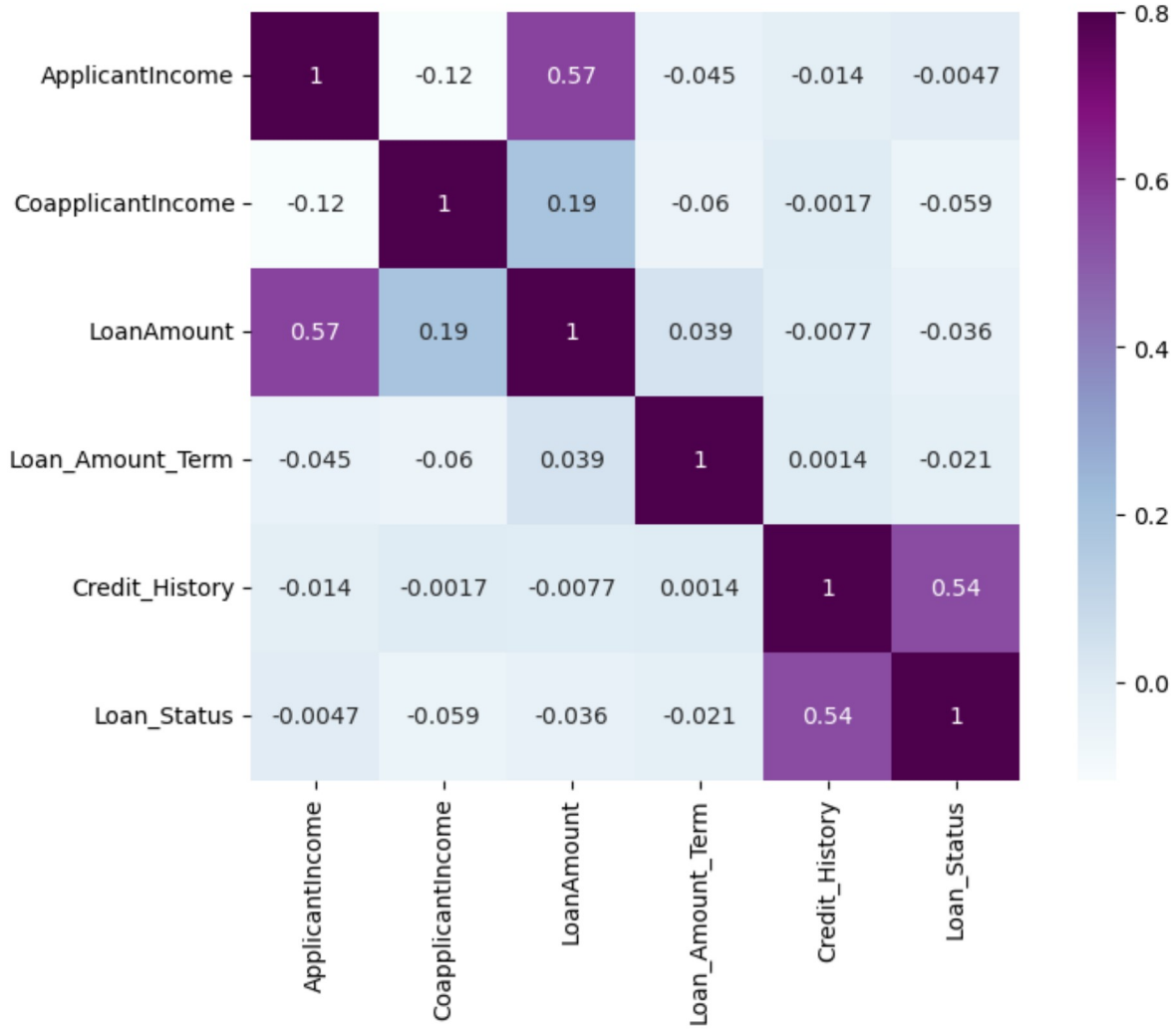




We see a lot of outliers in this variable and the distribution is fairly normal. We will treat the outliers in later sections.

Now let's look at the correlation between all numerical variables.We will use the heatmap to visualize the correlation. Heatmaps visualize data through variations in coloring. The variable with darker colors mean their correlation is more.

In [57]:
```
matrix=loan_data.corr()
plt.subplots(figsize=(9,6))
sns.heatmap(matrix,vmax=.8,square=True,cmap='BuPu',annot=True)
```

Out[57]:
```
<Axes: >
```

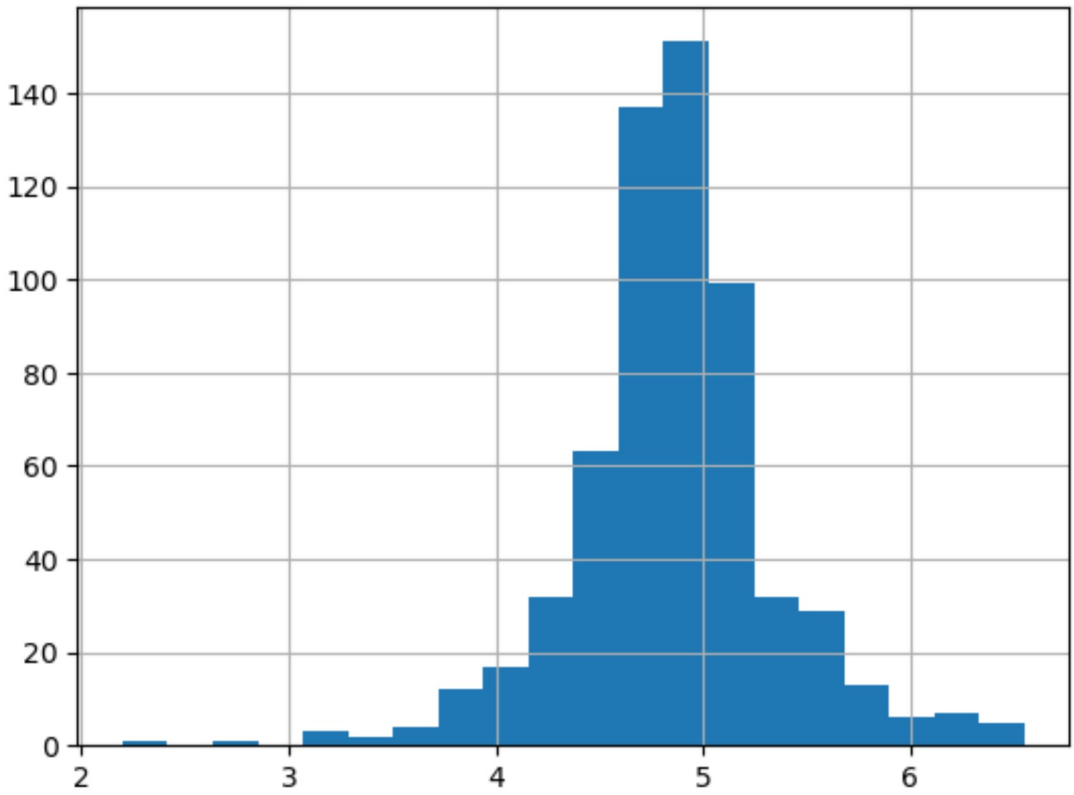



We can see that the most corelated variables are (Applicant Income-Loan Amount and (Credit history-Loan Status).

# Outlier Treatment

In [59]:
```
loan_data['LoanAmount_log']=np.log(loan_data['LoanAmount'])
loan_data['LoanAmount_log'].hist(bins=20)
```

Out[59]:
```
<Axes: >
```




Now the distribution looks much closer to normal and effect of extreme values has been significantly subsided.

In [ ]: