Loan Eligibility

In [203]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [204]:

```
dataset = pd.read_csv(r"C:\Users\Admin\OneDrive\Desktop\Future Interns\Loans\loan.csv")
```

In [205]:

```
dataset.head(2)
```

Out[205]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantInc
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	1

In [206]:

```
dataset.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
```

EDA

In [207]:

```
dataset.shape
```

Out[207]:

```
(614, 13)
```

In [208]:

```
dataset.describe()
```

Out[208]:

	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 [209]:

```
dataset.isnull().sum()
```

Out[209]:

```
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 [210]:

```
# Replace Missing categorical Values with the Mode
dataset["Gender"].fillna(dataset["Gender"].mode()[0], inplace = True)
```

In [211]:

```
dataset["Married"].fillna(dataset["Married"].mode()[0], inplace = True)
dataset["Self_Employed"].fillna(dataset["Self_Employed"].mode()[0], inplace = True)
dataset["Dependents"].fillna(dataset["Dependents"].mode()[0], inplace = True)
dataset["Loan_Amount_Term"].fillna(dataset["Loan_Amount_Term"].mode()[0], inplace= True)
dataset["Credit_History"].fillna(dataset["Credit_History"].mode()[0], inplace= True)
```

In [212]:

```
# Replace Missing Numerical Values With Mean
dataset["LoanAmount"].fillna(dataset["LoanAmount"].mean(), inplace= True)
```

In [213]:

```
dataset.isnull().sum()
```

Out[213]:

```
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 [214]:

```
pd.crosstab(dataset["Credit_History"], dataset["Loan_Status"], margins = True)
```

Out[214]:

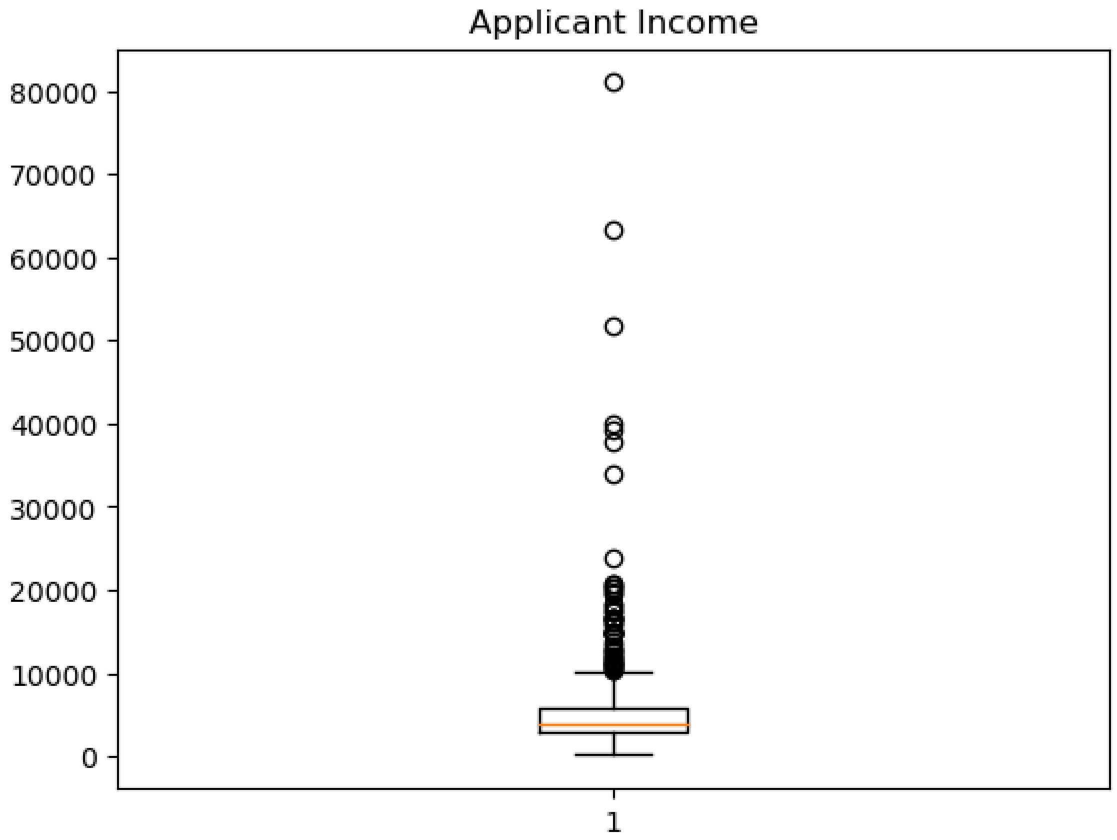
Loan_Status	N	Y	All
Credit_History			
0.0	82	7	89
1.0	110	415	525
All	192	422	614

In [215]:

```
plt.boxplot(dataset["ApplicantIncome"])
plt.title("Applicant Income")
```

Out[215]:

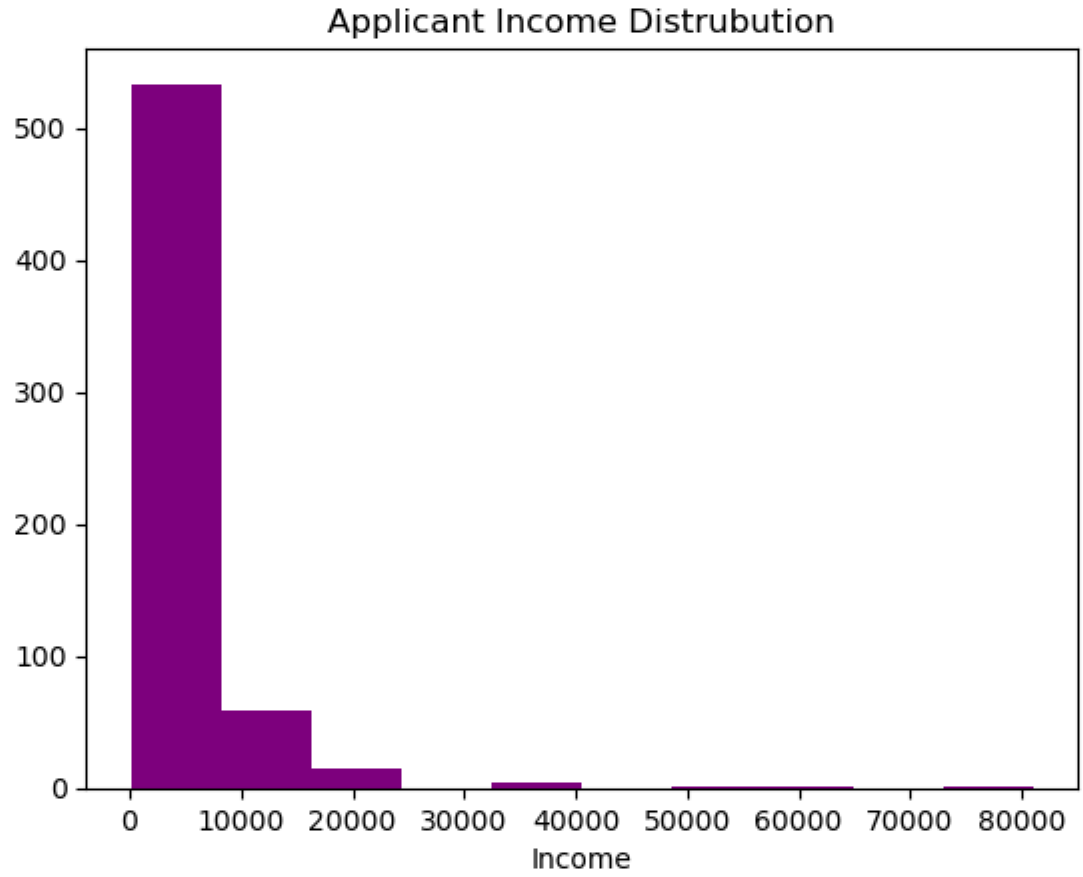
```
Text(0.5, 1.0, 'Applicant Income')
```



```
In [ ]:
```

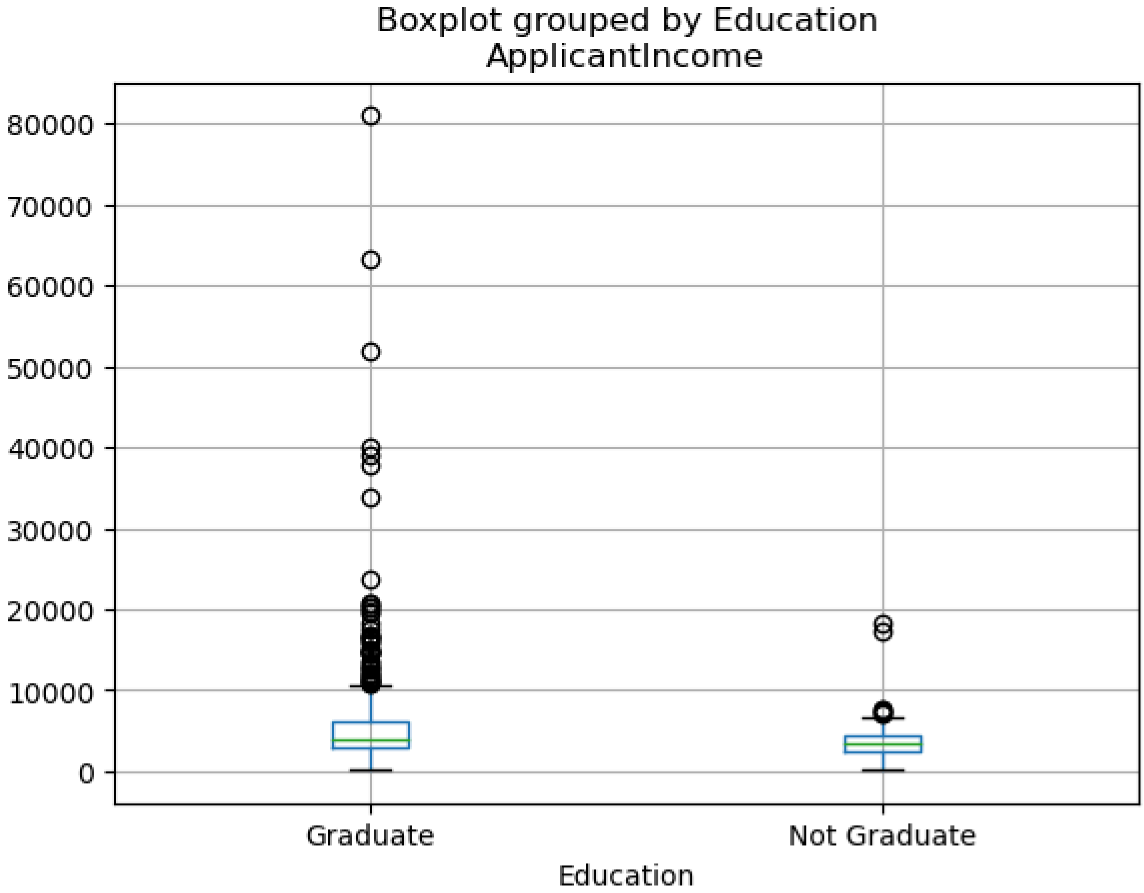
```
In [216]: plt.hist(dataset["ApplicantIncome"], color = "Purple")
          plt.title("Applicant Income Distrubution")
          plt.xlabel("Income")
```

```
Out[216]: Text(0.5, 0, 'Income')
```



```
In [217]: dataset.boxplot(column = "ApplicantIncome", by = "Education")
```

```
Out[217]: <Axes: title={'center': 'ApplicantIncome'}, xlabel='Education'>
```



Normalize Applicant Income(Right Skewed)

```
In [218]: #Use log function
          dataset
```

Out[218]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplicant
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	
...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

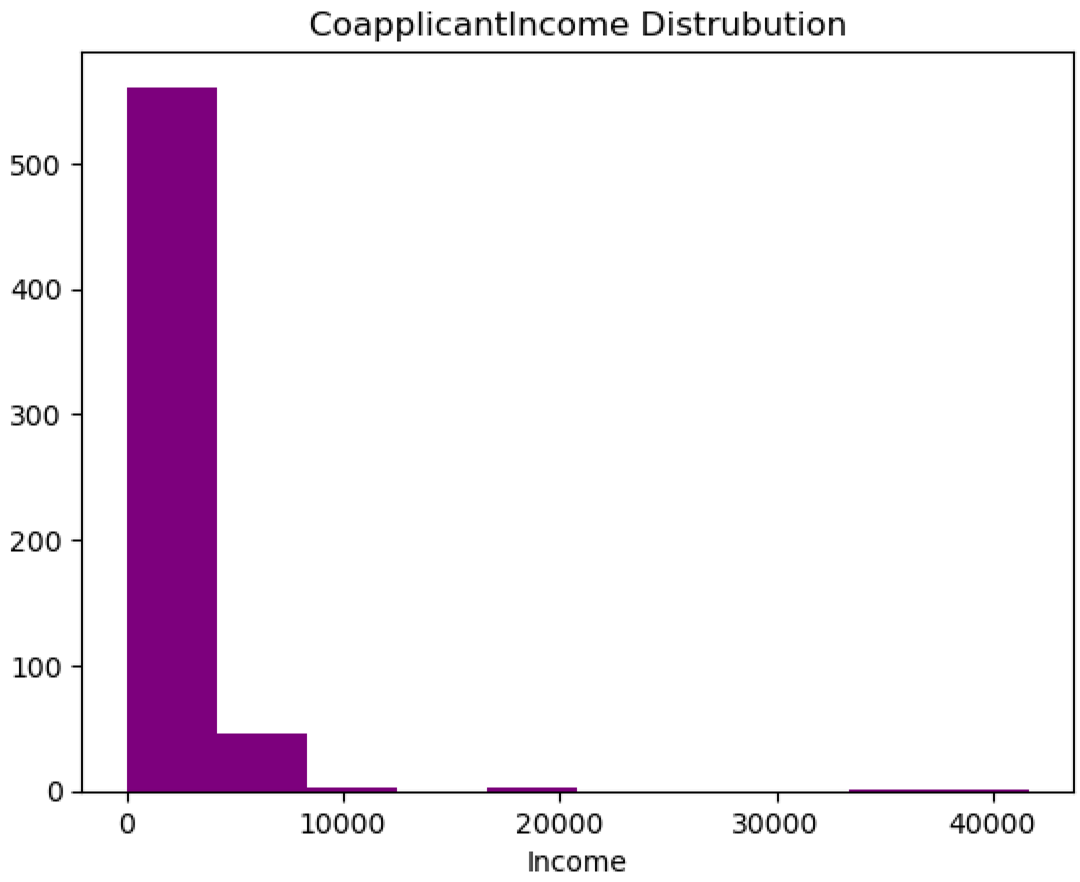
614 rows × 13 columns

In [219]:

```
plt.hist(dataset["CoapplicantIncome"], color = "Purple")
plt.title("CoapplicantIncome Distrubution")
plt.xlabel("Income")
```

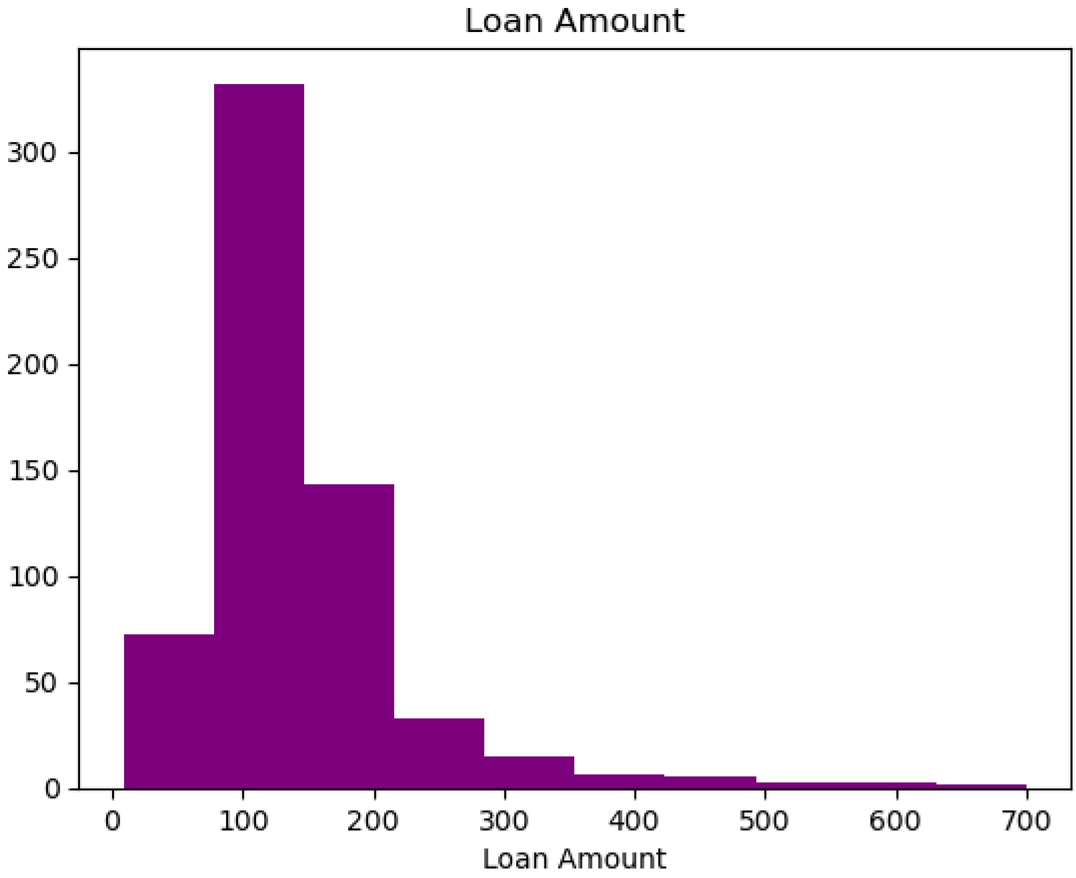
Out[219]:

```
Text(0.5, 0, 'Income')
```



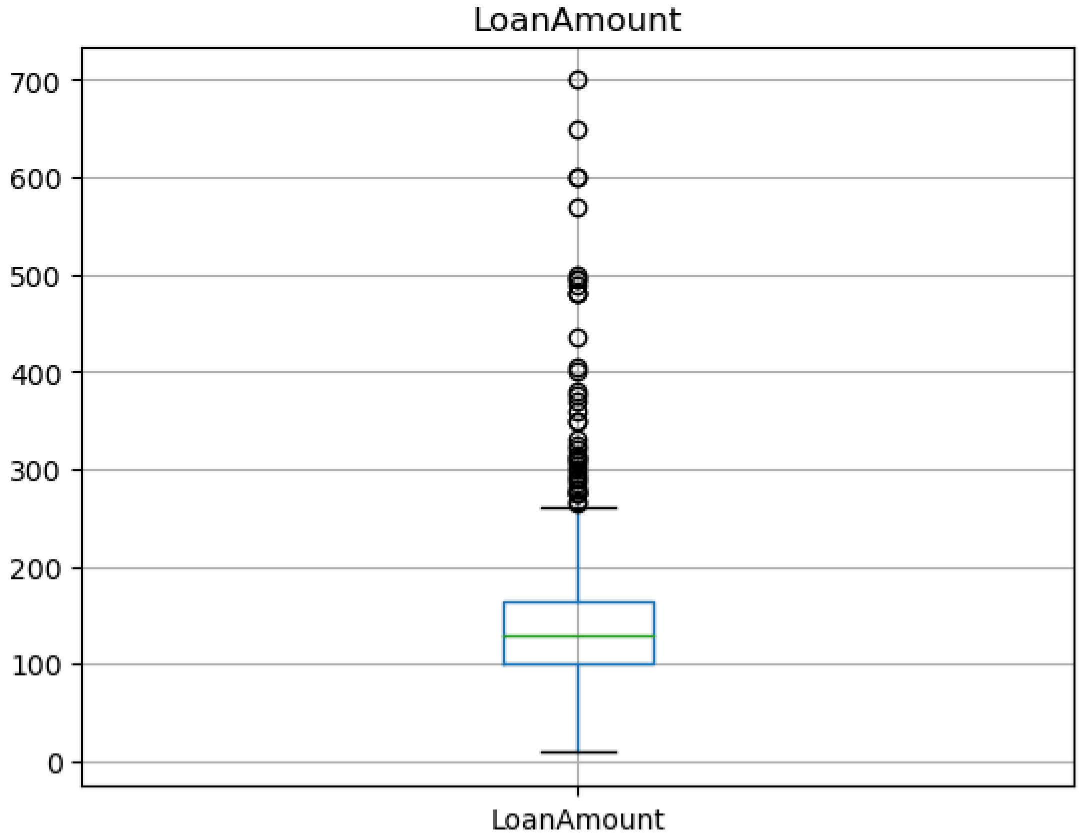
In [220]:
```
plt.hist(dataset["LoanAmount"], color = "Purple")
plt.title("Loan Amount")
plt.xlabel("Loan Amount")
```

Out[220]:
```
Text(0.5, 0, 'Loan Amount')
```



In [221]:
```
dataset.boxplot(column = 'LoanAmount')
plt.title("LoanAmount")
```

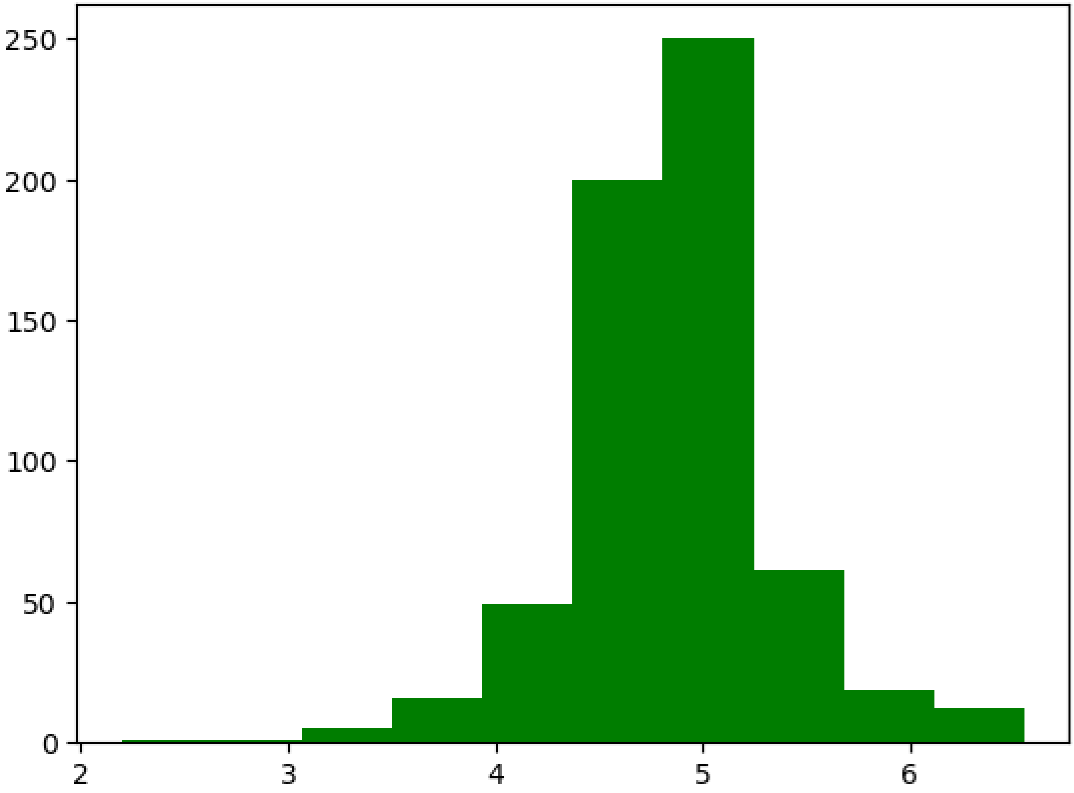
Out[221]:
```
Text(0.5, 1.0, 'LoanAmount')
```



Normalize Loan Amounts(Right Skewed)

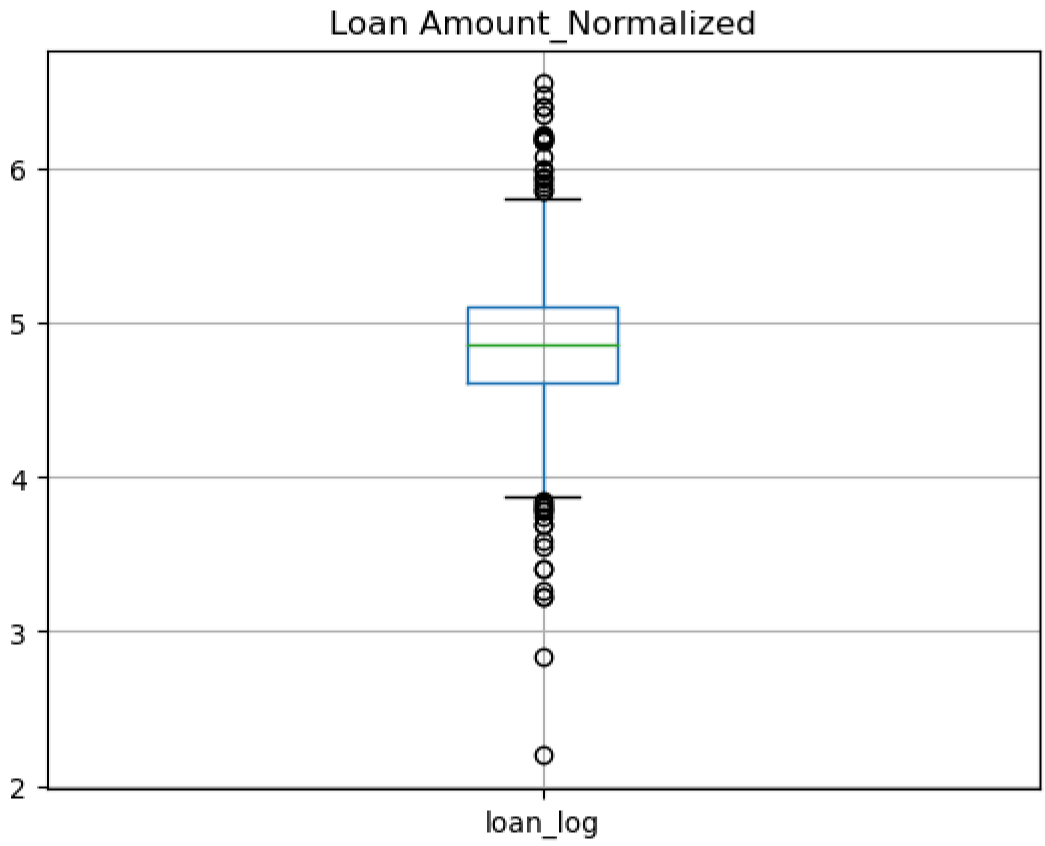
- We will use numpy and the log function

In [222]:
```
dataset["loan_log"] = np.log(dataset["LoanAmount"])
plt.hist(dataset["loan_log"], color = "green")
plt.show()
```



In [223]:
```
dataset.boxplot(column = "loan_log")
plt.title("Loan Amount_Normalized")
```

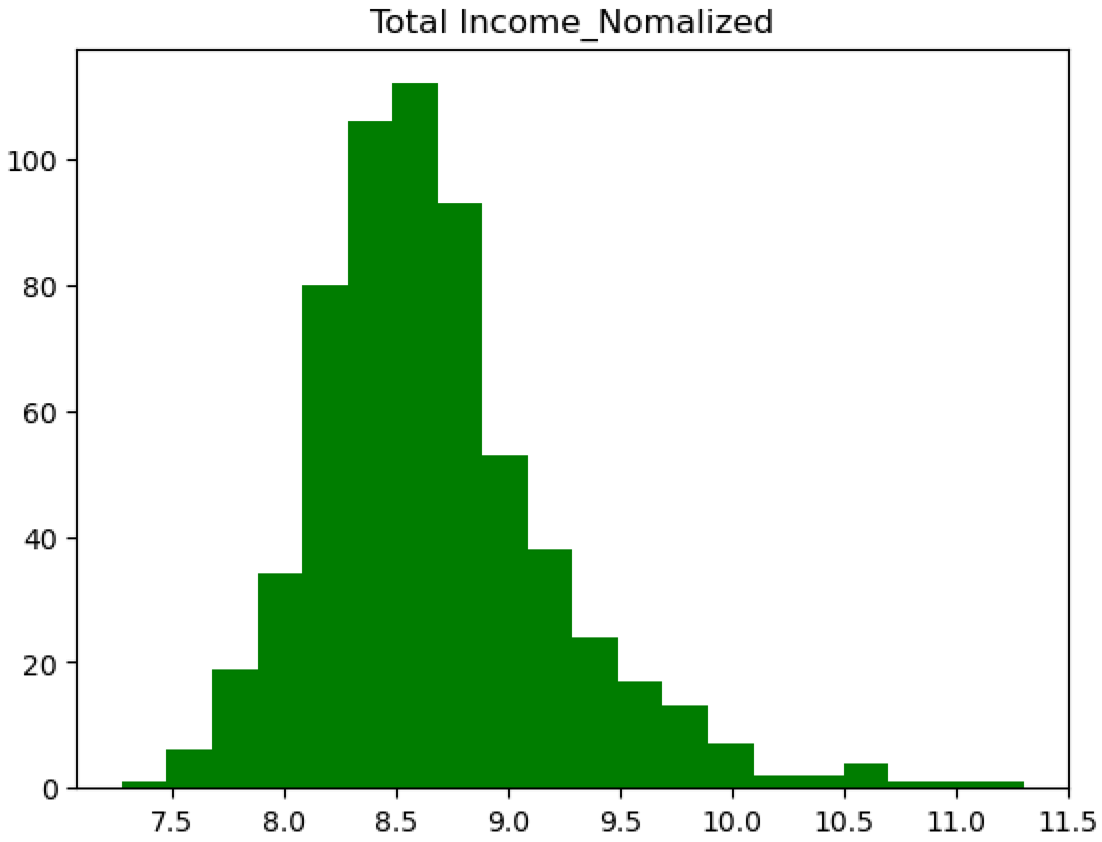
Out[223]:
```
Text(0.5, 1.0, 'Loan Amount_Normalized')
```



Normalize Total Income

In [224]:

```
dataset["TotalIncome"] = dataset["ApplicantIncome"] + dataset["CoapplicantIncome"]
dataset["TotalIncome_log"] = np.log(dataset["TotalIncome"])
plt.hist(dataset["TotalIncome_log"], color = "green",bins = 20)
plt.title('Total Income_Nomalized')
plt.show()
```



In [225]:

```
df = dataset[[ "ApplicantIncome", "CoapplicantIncome","LoanAmount", "Credit_History"]]
```

In [226]:

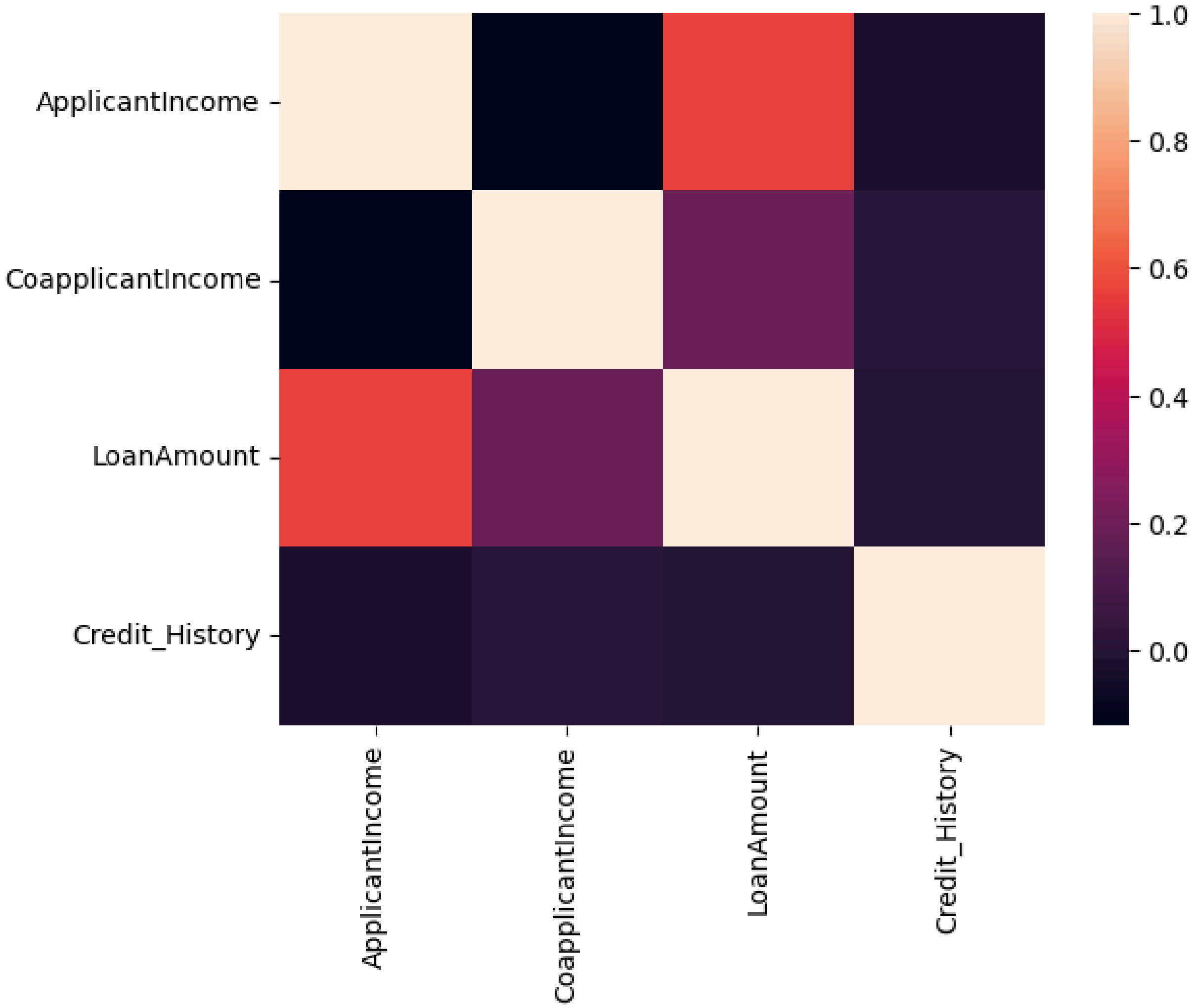
```
# df.corr()
```

In [227]:

```
sns.heatmap(df.corr())
```

Out[227]:

```
<Axes: >
```



In [228]:

```
dataset.head(1)
```

Out[228]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantInc
0	LP001002	Male	No	0	Graduate	No	5849	

In [229]:

```
males= len(dataset[dataset["Gender"]=="Male"])
print(males)
```

```
502
```

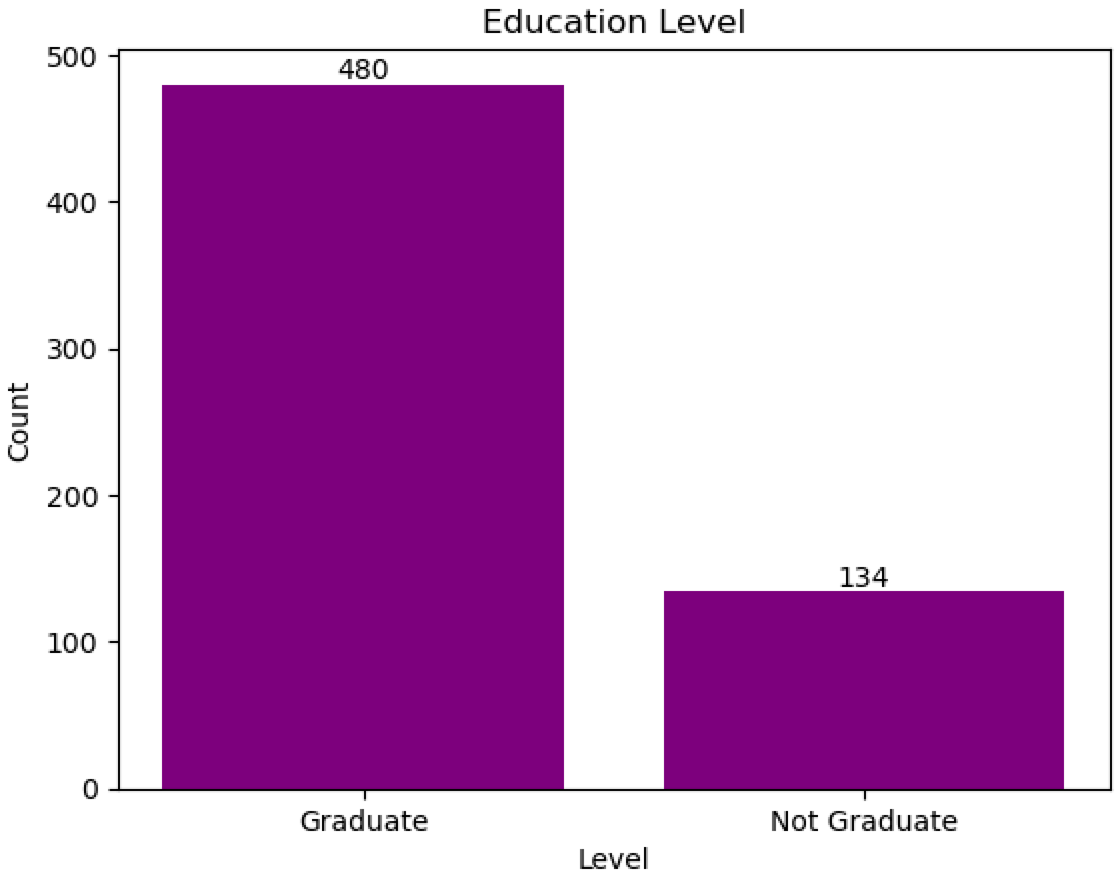
In [230]:

```
females= len(dataset[dataset["Gender"]=="Female"])
print(females)
```

```
112
```

In [231]:

```
education_count= dataset["Education"].value_counts()
plt.bar(education_count.index, education_count.values, color = "purple")
plt.title("Education Level")
plt.ylabel("Count")
plt.xlabel("Level")
# Add data labels
for index, value in enumerate(education_count.values):
    plt.text(index, value, str(value), ha='center', va='bottom')
```



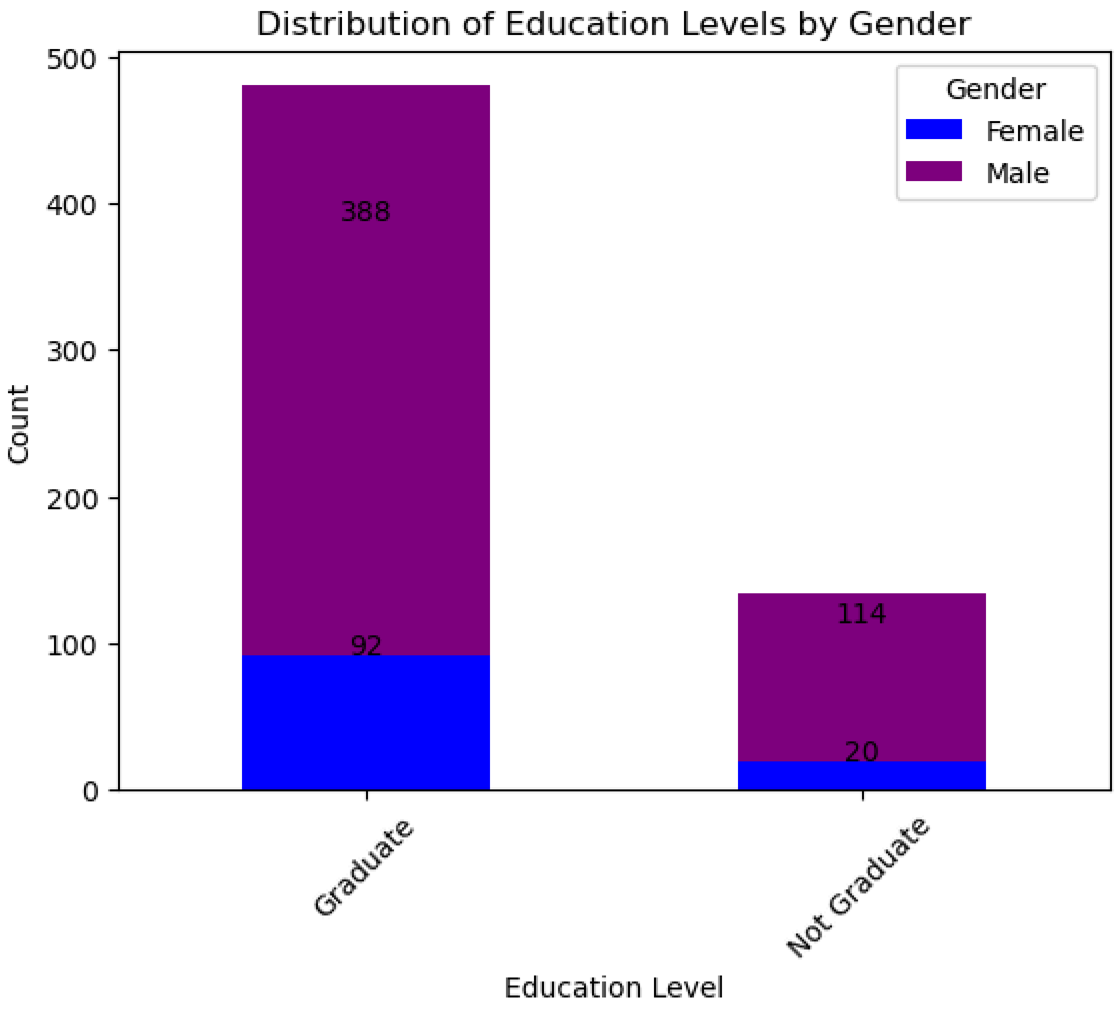
In [232]:

```
education_gender_counts = dataset.groupby(['Education', 'Gender']).size().unstack()
# Assuming 'Gender' and 'Education' columns exist in the dataset
education_gender_counts = dataset.groupby(['Education', 'Gender']).size().unstack()
# Define colors for the bars
colors = {'Male': 'purple', 'Female': 'blue'}

# Plot the bar chart
education_gender_counts.plot(kind='bar', stacked=True, color=[colors.get(gender) for gen
plt.xlabel('Education Level')
plt.ylabel('Count')
plt.title('Distribution of Education Levels by Gender')
plt.xticks(rotation=45)  # Rotate x-axis labels if they are too long
plt.legend(title='Gender')

# Add data labels
for i in range(education_gender_counts.shape[0]):
    for j in range(education_gender_counts.shape[1]):
        plt.text(i, education_gender_counts.iloc[i, j] + 1, str(education_gender_counts.

plt.show()
```



In [233]:

```
dataset.head(1)
```

Out[233]:

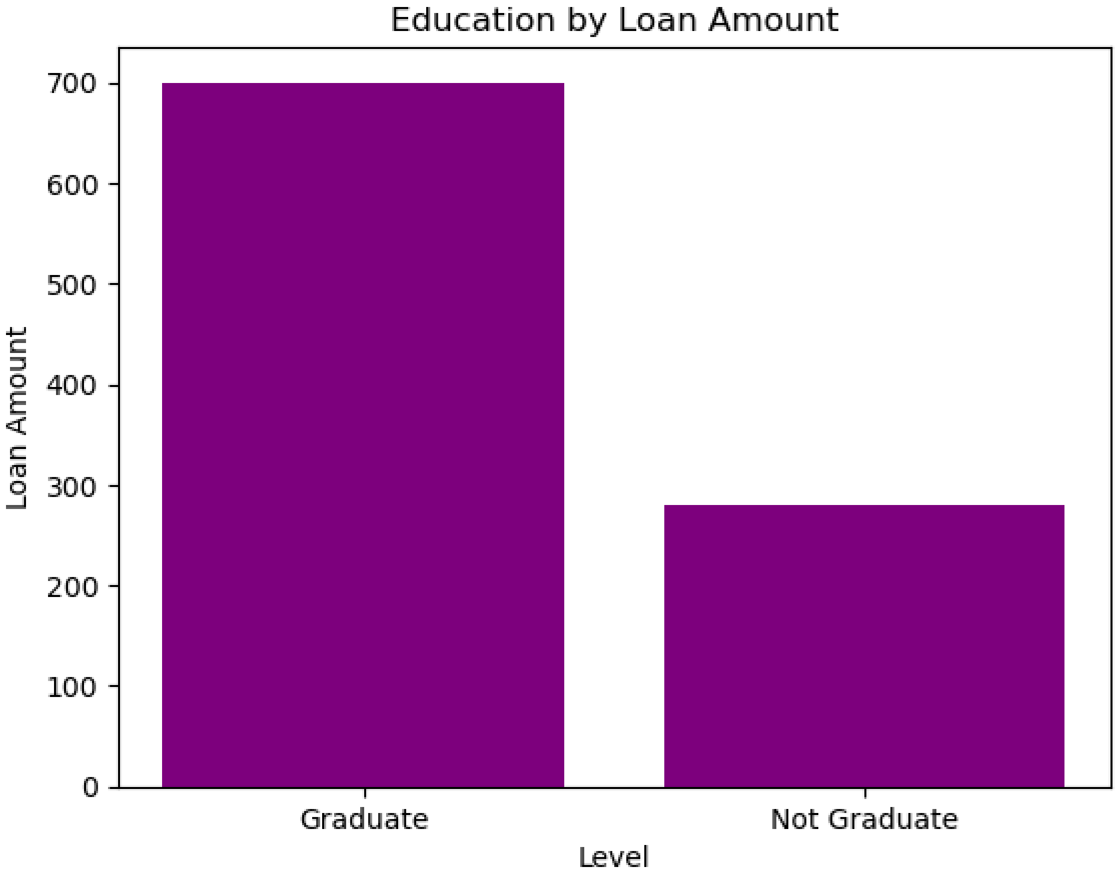
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantInc
0	LP001002	Male	No	0	Graduate	No	5849	

In [234]:

```
# Create the bar chart
plt.bar(dataset["Education"], dataset["LoanAmount"], color="purple")
plt.title("Education by Loan Amount")
plt.xlabel("Level")
plt.ylabel("Loan Amount")
```

Out[234]:

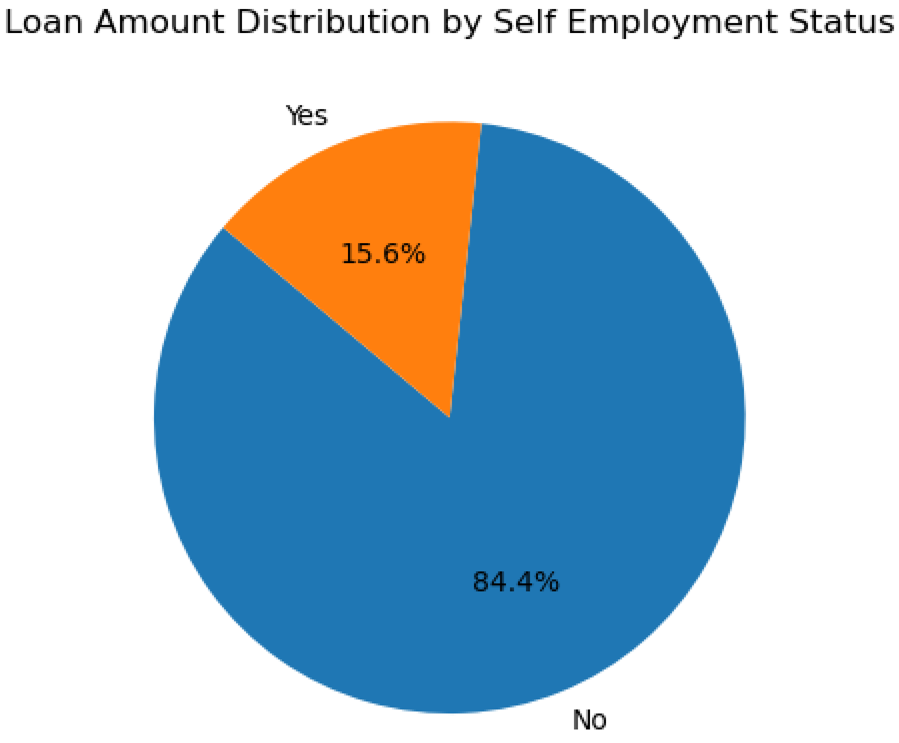
```
Text(0, 0.5, 'Loan Amount')
```



In [235]:
```
# Group by 'Self_Employed' and sum the 'LoanAmount'
loan_amounts = dataset.groupby('Self_Employed')['LoanAmount'].sum()

# Create the pie chart
plt.pie(loan_amounts, labels=loan_amounts.index, autopct='%1.1f%%', startangle=140)
plt.title('Loan Amount Distribution by Self Employment Status')
```

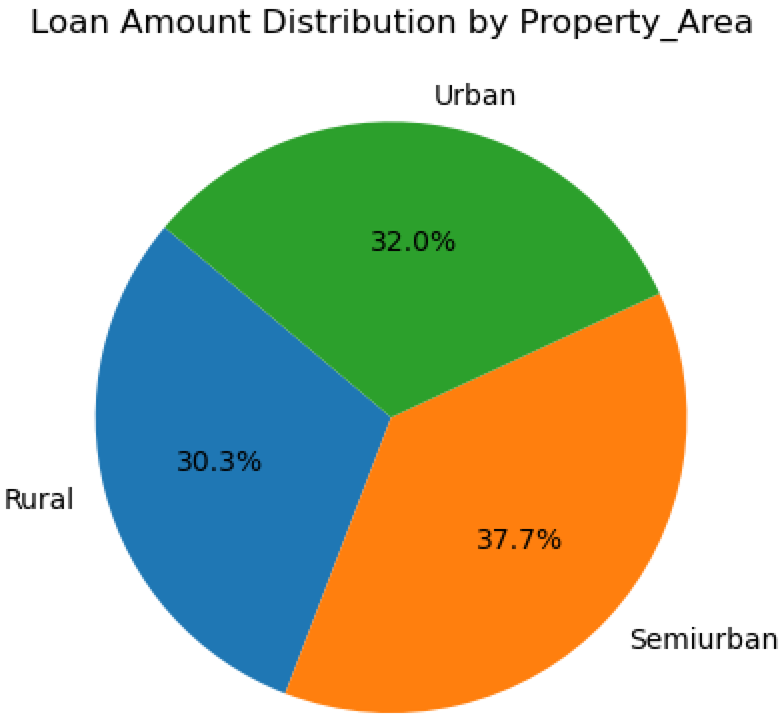
Out[235]:
```
Text(0.5, 1.0, 'Loan Amount Distribution by Self Employment Status')
```



```
In [236]:
# Group by 'Self_Employed' and sum the 'LoanAmount'
loan_amounts2 = dataset.groupby('Property_Area')['LoanAmount'].sum()

# Create the pie chart
plt.pie(loan_amounts2, labels=loan_amounts2
        .index, autopct='%1.1f%%', startangle=140)
plt.title('Loan Amount Distribution by Property_Area')
```

```
Out[236]: Text(0.5, 1.0, 'Loan Amount Distribution by Property_Area')
```



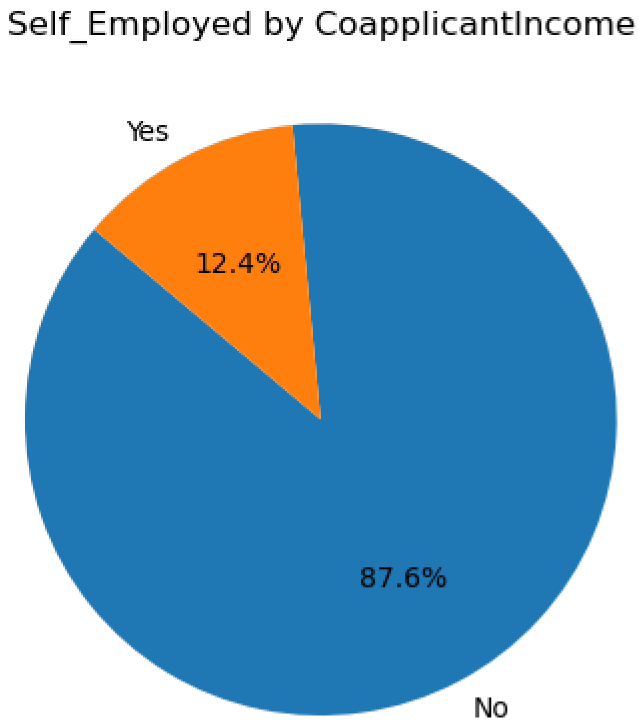
In [237]:

```
# Group by 'Self_Employed' and sum the 'LoanAmount'
loan_amounts3 = dataset.groupby('Self_Employed')['CoapplicantIncome'].sum()

# Create the pie chart
plt.pie(loan_amounts3, labels=loan_amounts3
        .index, autopct='%1.1f%%', startangle=140)
plt.title('Self_Employed by CoapplicantIncome')
```

Out[237]:

```
Text(0.5, 1.0, 'Self_Employed by CoapplicantIncome')
```


Self_Employed by CoapplicantIncome

Select Dependent and Independent variables

In [238]:
```
from numpy import r_
x = dataset.iloc[:,np.r_[1:5, 9:11, 13:15]].values
y = dataset.iloc[:,12].values
```

Split Data

In [ ]:
```
```

In [239]:
```
dataset.head(1)
```

Out[239]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantInc
0	LP001002	Male	No	0	Graduate	No	5849	

In [240]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state
```

Create Dummy Variables

In [241]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder_x  = LabelEncoder()
```

In [242]:
```
x_test[:,7] = label_encoder_x.fit_transform(x_test[:,7])
x_test[:,3]= label_encoder_x.fit_transform(x_test[:,3])
x_test[:,2] = label_encoder_x.fit_transform(x_test[:,2])
```

```
x_test[:,1] = label_encoder_x.fit_transform(x_test[:,1])
x_test[:,0] = label_encoder_x.fit_transform(x_test[:,0])
```

In [243]:

```
x_train[:,7] = label_encoder_x.fit_transform(x_train[:,7])
x_train[:,2] = label_encoder_x.fit_transform(x_train[:,2])
x_train[:,1] = label_encoder_x.fit_transform(x_train[:,1])
x_train[:,0] = label_encoder_x.fit_transform(x_train[:,0])
x_train[:,3] = label_encoder_x.fit_transform(x_train[:,3])
```

In [244]:

```
x_test[:,7] = label_encoder_x.fit_transform(x_test[:,7])
```

In [245]:

```
# x_test
```

In [246]:

```
labelencoder_y = LabelEncoder()
```

In [247]:

```
y_train= labelencoder_y.fit_transform(y_train)
```

In [248]:

```
y_test = labelencoder_y.fit_transform(y_test)
```

Scale DataSet

- Scale the data set to account fot yue various ranges and improve Accuracy

In [249]:

```
from sklearn.preprocessing import  StandardScaler
ss = StandardScaler()
```

In [250]:

```
np.set_printoptions(threshold=np.inf)
print(x_train[:2])
```

```
[[1 1 0 0 360.0 1.0 4.875197323201151 267]
 [1 0 1 0 360.0 1.0 5.278114659230517 407]]
```

In [251]:

```
x_train = ss.fit_transform(x_train)
```

In [252]:

```
x_test = ss.fit_transform(x_test)
```

In [253]:

```
np.set_printoptions(threshold=np.inf)
print(x_train[:2])
```

```
[[ 0.47374983  0.71143163 -0.76304669 -0.53102197  0.26983787  0.41790088
   0.02443538  0.29186348]
 [ 0.47374983 -1.40561644  0.22549137 -0.53102197  0.26983787  0.41790088
   0.81960159  1.36113256]]
```

In [254]:

```
np.set_printoptions(threshold=np.inf)
print(x[:2])
```

```
[['Male' 'No' '0' 'Graduate' 360.0 1.0 4.986425672954842 5849.0]
 ['Male' 'Yes' '1' 'Graduate' 360.0 1.0 4.852030263919617 6091.0]]
```

In [255]:

```
np.set_printoptions(threshold=np.inf)
print(y_test[:2])
```

```
[1 0]
```

Build Model

Decision Trees Classifier

In [256]:
```
from sklearn.tree import DecisionTreeClassifier
DTClassifier = DecisionTreeClassifier(criterion="entropy", random_state=0)
```

In [257]:
```
DTClassifier.fit(x_train, y_train)
```

Out[257]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

Predict Test Data Set Values

In [258]:
```
#Give Data Set
y_pred = DTClassifier.predict(x_test)
```

In [259]:
```
y_pred
```

Out[259]:
```
array([0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
       1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1])
```

Evaluate Accuracy

In [260]:
```
from sklearn import metrics
print(f'DTC Accuracy: {metrics.accuracy_score(y_pred, y_test)}')
```

```
DTC Accuracy: 0.6991869918699187
```

Naive Bayes Algoriyhm

In [261]:
```
from sklearn.naive_bayes import GaussianNB
NB = GaussianNB()
```

In [262]:
```
NB.fit(x_train, y_train)
```

Out[262]:
```
▾ GaussianNB
GaussianNB()
```

In [263]:
```
y_pred = NB.predict(x_test)
```

In [264]:

```
print(f"NB A ccuracy: {metrics.accuracy_score(y_pred, y_test)}")
```

```
NB A ccuracy: 0.8292682926829268
```

Import Test Data

In [265]:

```
test_data = pd.read_csv(r"C:\Users\Admin\OneDrive\Desktop\Future Interns\Loans\Tesr Dara
```

In [266]:

```
test_data.head(5)
```

Out[266]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Loa
0	Female	No	0	Graduate	No	5849	0.0	
1	Male	Yes	1	Graduate	No	4583	1508.0	
2	Male	Yes	0	Graduate	Yes	3000	0.0	
3	Male	Yes	0	Not Graduate	No	2583	2358.0	
4	Male	Yes	0	Graduate	No	6000	0.0	

In [267]:

```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Gender             601 non-null    object 
 1   Married            611 non-null    object 
 2   Dependents         599 non-null    object 
 3   Education          614 non-null    object 
 4   Self_Employed      583 non-null    object 
 5   ApplicantIncome    614 non-null    int64  
 6   CoapplicantIncome  614 non-null    float64
 7   LoanAmount         592 non-null    float64
 8   Loan_Amount_Term   600 non-null    float64
 9   Credit_History     564 non-null    float64
 10  Property_Area      614 non-null    object 
dtypes: float64(4), int64(1), object(6)
memory usage: 52.9+ KB
```

In [268]:

```
test_data.isnull().sum()
```

Out[268]:

```
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        31
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
dtype: int64
```

In [269]:

```
test_data["Gender"].fillna(test_data["Gender"].mode()[0], inplace=True)
test_data["Married"].fillna(test_data["Married"].mode()[0], inplace=True)
test_data["Dependents"].fillna(test_data["Dependents"].mode()[0], inplace=True)
test_data["Self_Employed"].fillna(test_data["Self_Employed"].mode()[0], inplace=True)
test_data["Credit_History"].fillna(test_data["Credit_History"].mode()[0], inplace=True)
test_data["Loan_Amount_Term"].fillna(test_data["Loan_Amount_Term"].mode()[0], inplace=Tr
```

In [270]:

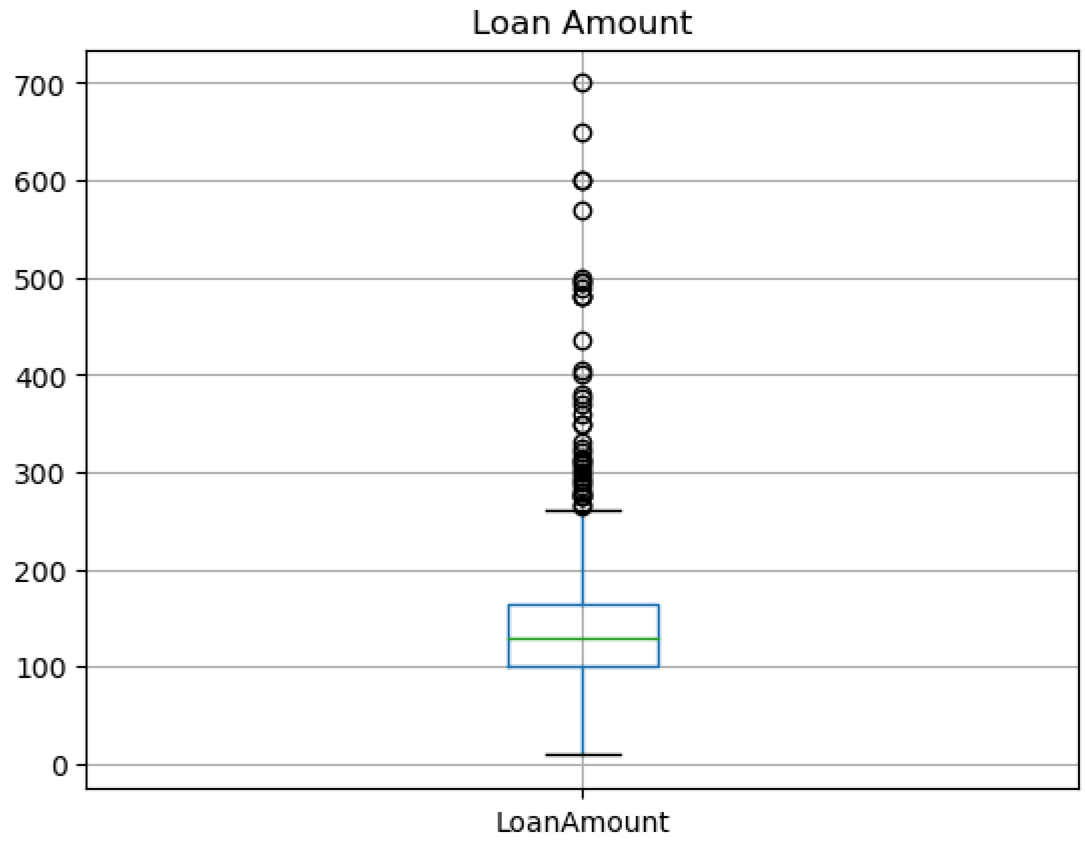
```
test_data["LoanAmount"].fillna(test_data["LoanAmount"].mean(), inplace=True)
```

In [271]:

```
test_data.boxplot(column="LoanAmount")
plt.title("Loan Amount")
```

Out[271]:

```
Text(0.5, 1.0, 'Loan Amount')
```

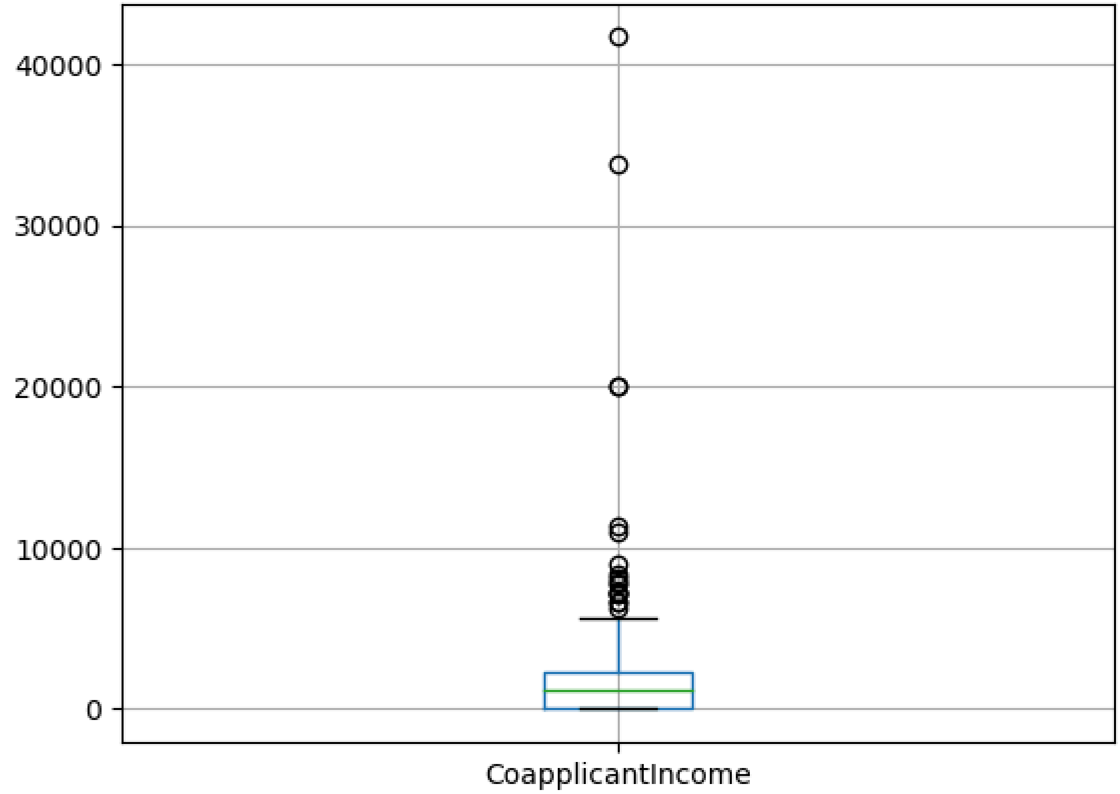


In [272]:

```
test_data.boxplot(column="CoapplicantIncome")
```

Out[272]:

```
<Axes: >
```



In [273]:
```
# test_data["Total_Income"] = test_data["ApplicantIncome"] + test_data["CoapplicantIncor
```

In [274]:
```
test_data['loan_log'] = np.log(test_data['LoanAmount'])
```

In [275]:
```
test_data["ToatalIncome"]  = test_data["ApplicantIncome"] + test_data["CoapplicantIncome
test_data["TotalIncome_log"] = np.log(test_data["ToatalIncome"])
```

In [276]:
```
# test_data.isnull().sum()
```

In [277]:
```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Gender             614 non-null    object 
 1   Married            614 non-null    object 
 2   Dependents         614 non-null    object 
 3   Education          614 non-null    object 
 4   Self_Employed      614 non-null    object 
 5   ApplicantIncome    614 non-null    int64  
 6   CoapplicantIncome  614 non-null    float64
 7   LoanAmount         614 non-null    float64
 8   Loan_Amount_Term   614 non-null    float64
 9   Credit_History     614 non-null    float64
 10  Property_Area      614 non-null    object 
 11  loan_log           614 non-null    float64
 12  ToatalIncome       614 non-null    float64
 13  TotalIncome_log    614 non-null    float64
dtypes: float64(7), int64(1), object(6)
memory usage: 67.3+ KB
```

In [278]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 16 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             614 non-null    object 
 2   Married            614 non-null    object 
 3   Dependents         614 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      614 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         614 non-null    float64
 9   Loan_Amount_Term   614 non-null    float64
 10  Credit_History     614 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
 13  loan_log           614 non-null    float64
 14  TotalIncome        614 non-null    float64
 15  TotalIncome_log    614 non-null    float64
dtypes: float64(7), int64(1), object(8)
memory usage: 76.9+ KB
```

In [279]:

```
# test = test_data.iloc[:,np.r_[1:5, 9:11, 13:14]].values
x2 = test_data.iloc[:,np.r_[0:4, 8:10, 11:13]].values
```

In [285]:

```
x2[:,0] = label_encoder_x.fit_transform(x[:,0])
x2[:,1] = label_encoder_x.fit_transform(x[:,1])
x2[:,2] = label_encoder_x.fit_transform(x[:,2])
x2[:,3] = label_encoder_x.fit_transform(x[:,3])
```

In [287]:

```
x2 = ss.fit_transform(x2)
```

In [288]:

```
prediction = NB.predict(x2)
```

In [291]:

```
# prediction
```

In [297]:

```
test_data["Loan_Status"] = prediction
test_data["Loan_Status"] = test_data["Loan_Status"].replace({1:"Yes", 0:"No"})
```

In [298]:

```
test_data
```

Out[298]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	L
0	Female	No	0	Graduate	No	5849	0.0	
1	Male	Yes	1	Graduate	No	4583	1508.0	
2	Male	Yes	0	Graduate	Yes	3000	0.0	
3	Male	Yes	0	Not Graduate	No	2583	2358.0	
4	Male	Yes	0	Graduate	No	6000	0.0	
...	...	...	...	...	...	...	...	
609	Female	No	0	Graduate	No	2900	0.0	
610	Male	Yes	3+	Graduate	No	4106	0.0	
611	Male	Yes	1	Graduate	No	8072	240.0	
612	Male	Yes	2	Graduate	No	7583	0.0	
613	Female	No	0	Graduate	Yes	4583	0.0	

614 rows × 16 columns