LOAN APPROVAL PREDICTION

IMPORTING LIBRARIES

In [28]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
from sklearn.metrics import classification_report,confusion_matrix
```

IMPORTING DATASET

In [3]:
```
df=pd.read_excel("/Users/yashikarao/Downloads/Copy of loan.xlsx")
```

In [4]:
```
df.head()
```

Out[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0

In [5]:
```
df.tail()
```

Out[5]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Te
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	36(
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	18(
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	36(
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	36(
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	36(

In [6]:
```
df.shape
```

Out[6]:
```
(614, 13)
```

In [8]:
```
df.describe()
```

Out[8]:

	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

CHECKING FOR NULL VALUES

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
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
```

TREATING NULL VALUES

In [10]:
```
df.fillna(method='ffill',inplace=True)
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           1
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

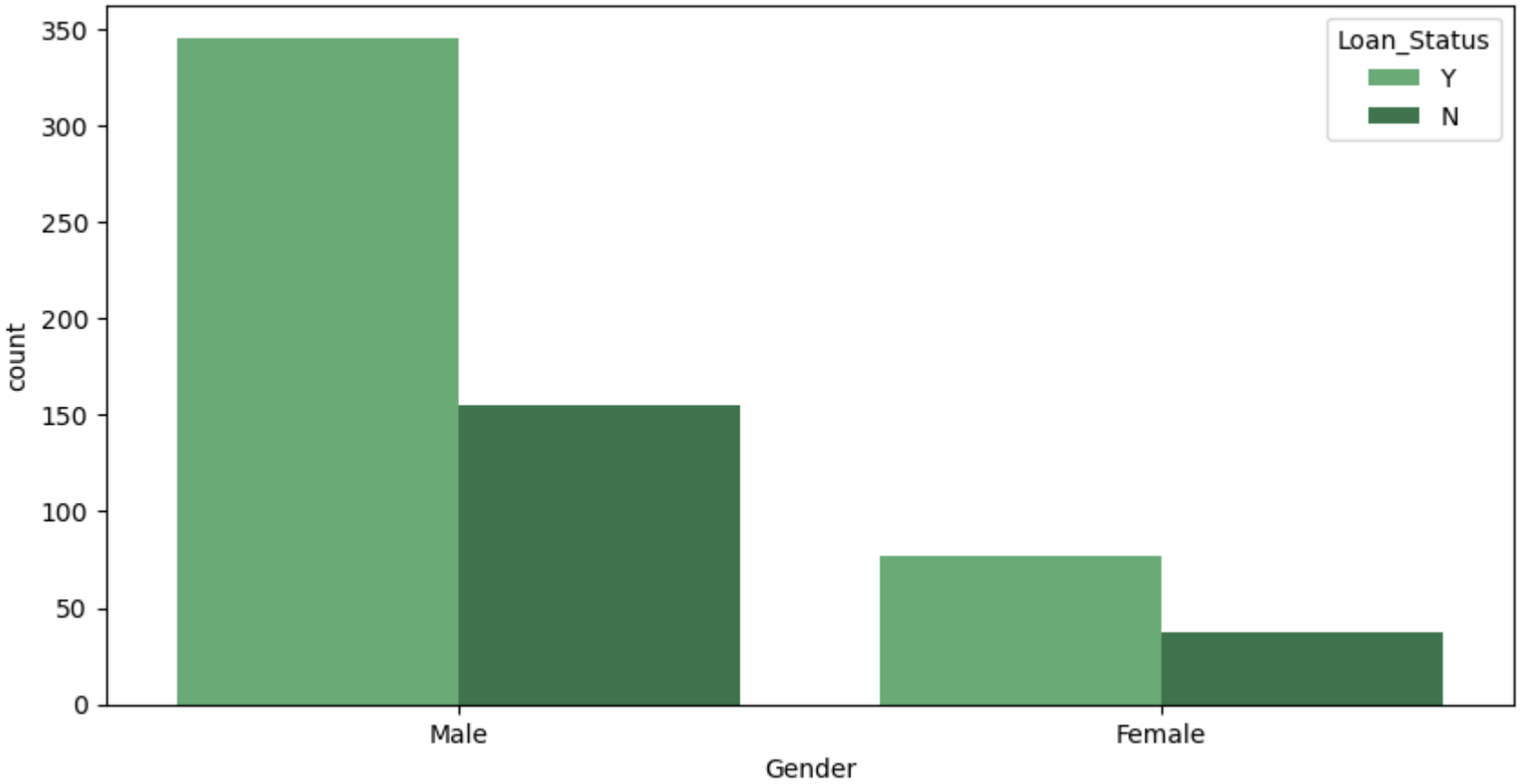
In [12]:
```
df.fillna(method='bfill',inplace=True)
```

In [13]:
```
df.isnull().sum()
```

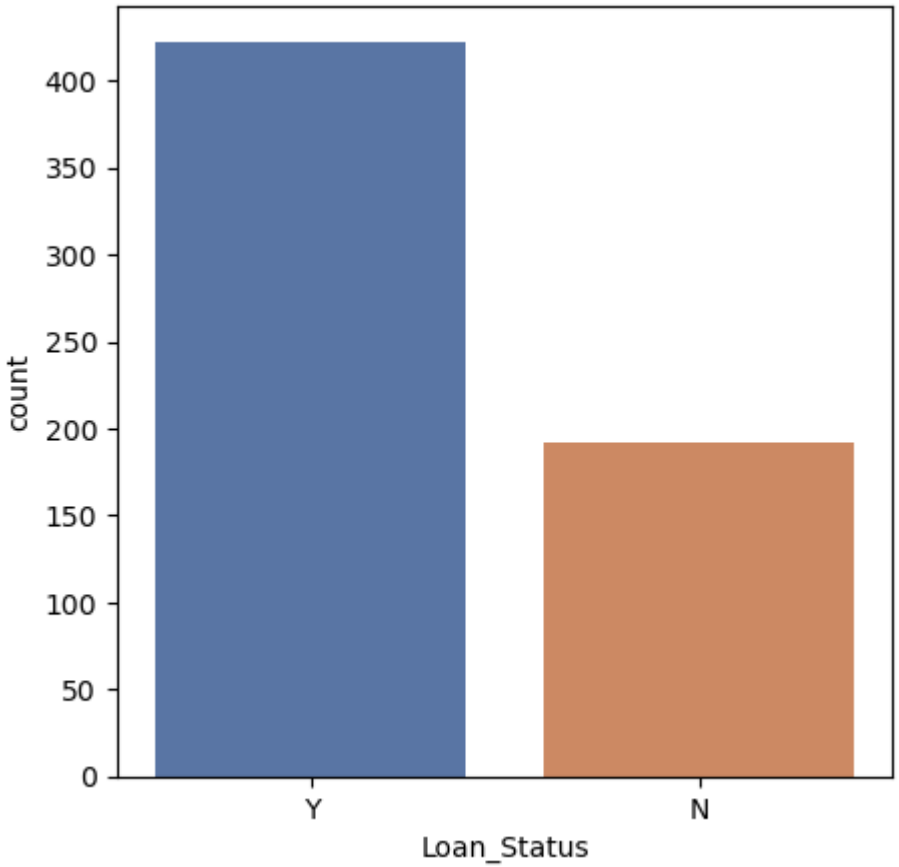
Out[13]:
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

VISUALIZATION

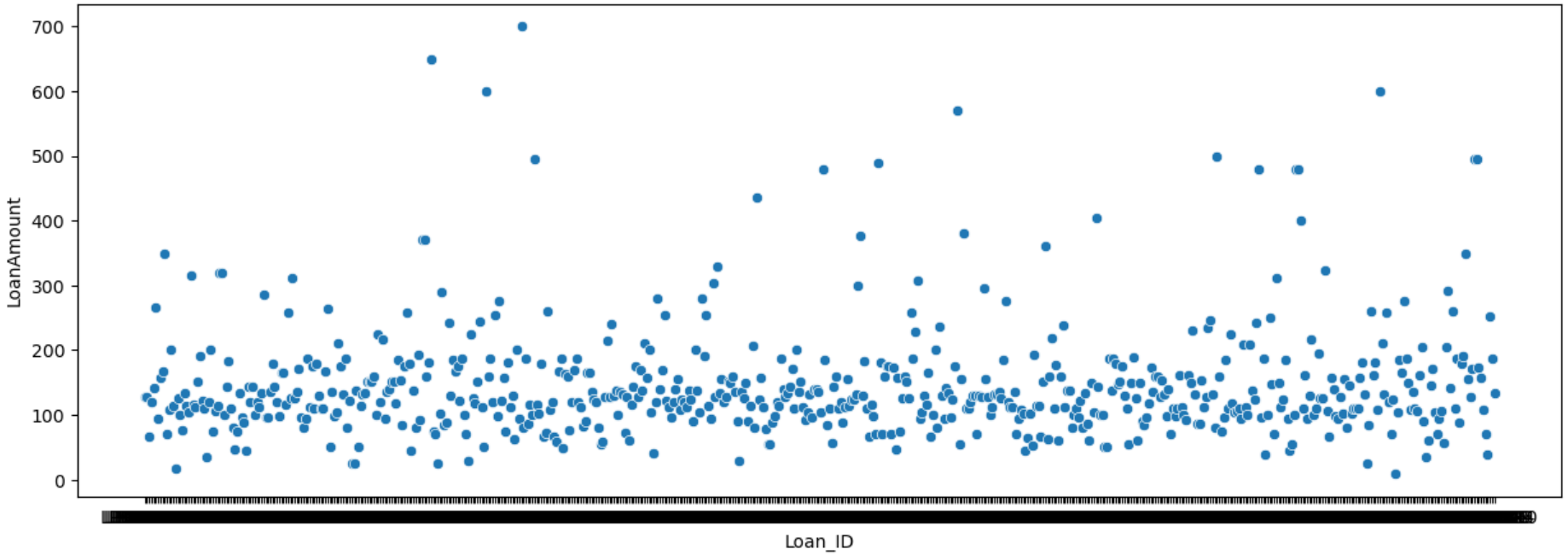
In [14]:
```
plt.figure(figsize=(10,5))
sns.countplot(x='Gender',data=df,hue='Loan_Status',palette='Greens_d')
plt.show()
```



In [15]:
```
plt.figure(figsize=(5,5))
sns.countplot(x='Loan_Status',data=df,palette='deep')
plt.show()
```

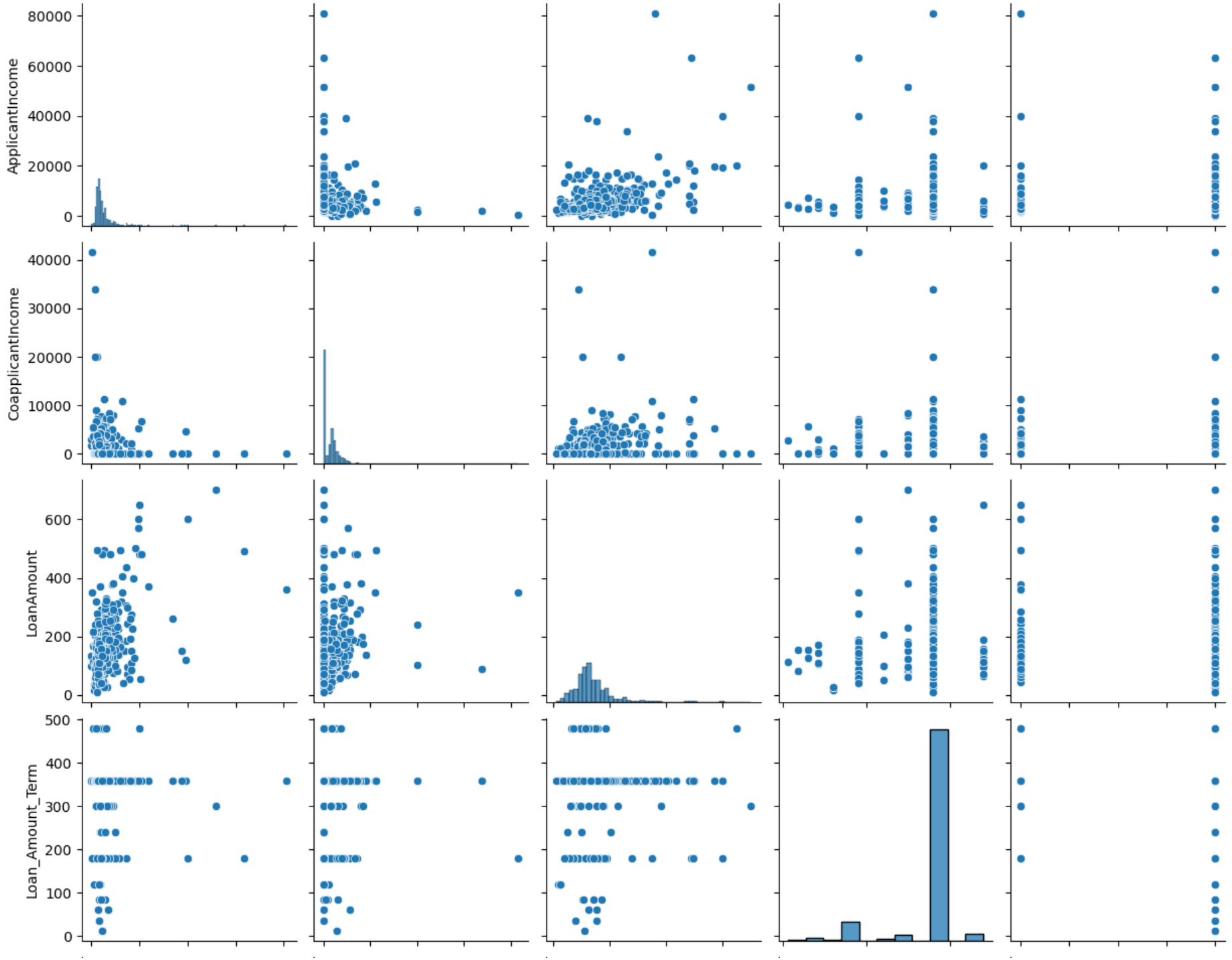


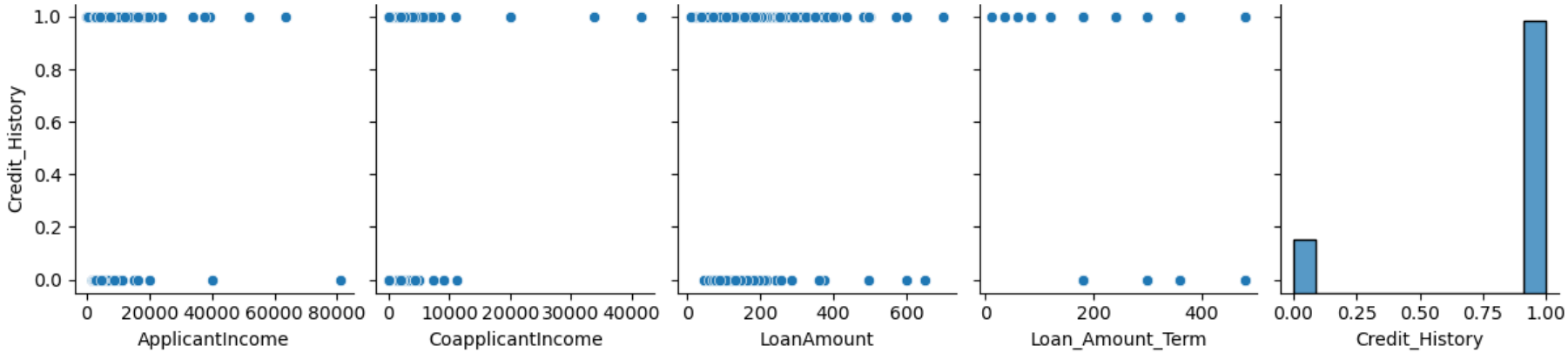
In [16]:
```
plt.figure(figsize=(15,5))
sns.scatterplot(x="Loan_ID",y="LoanAmount",data=df)
plt.show()
```



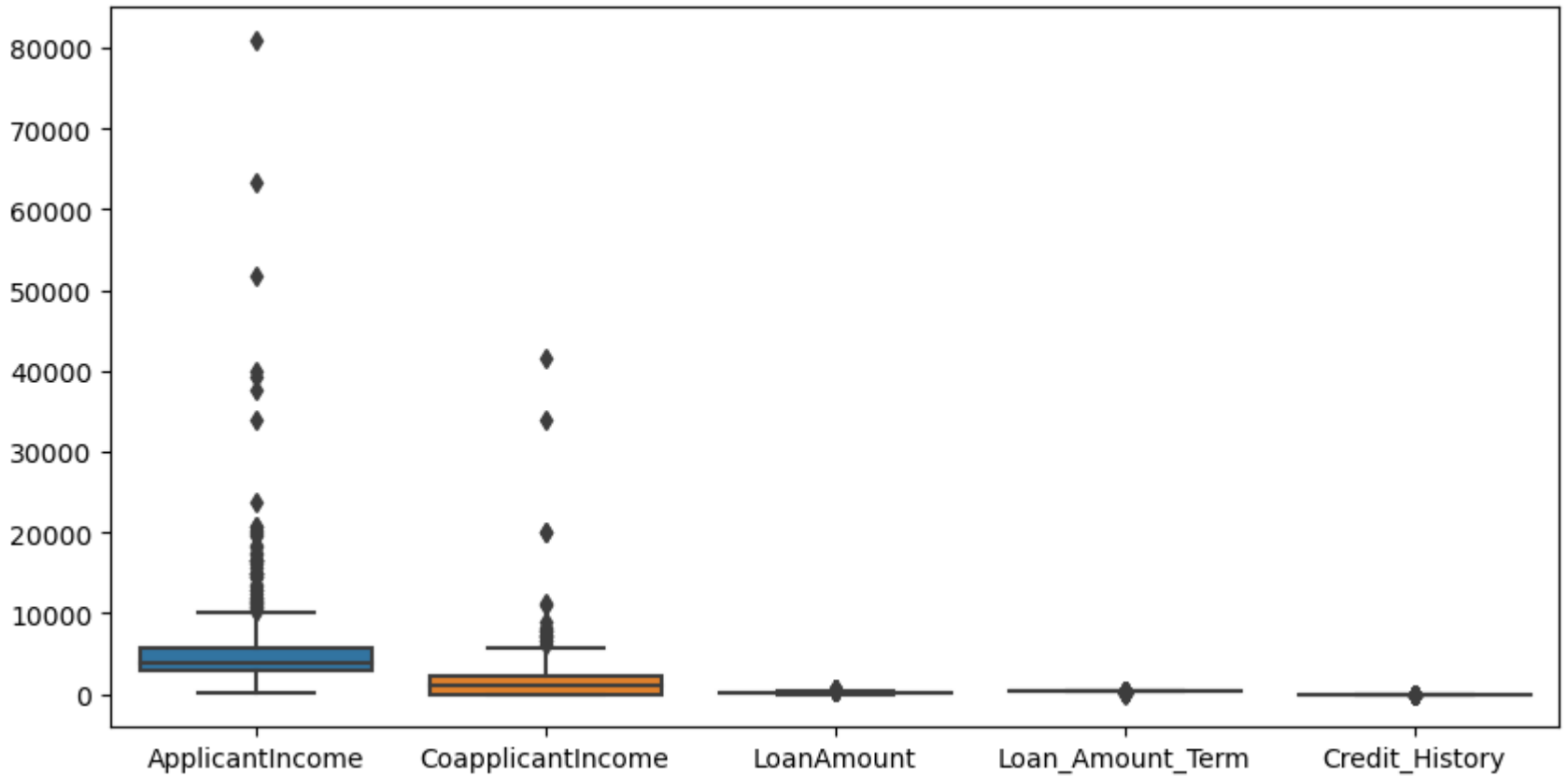
In [17]:
```
plt.figure(figsize=(10,5))
sns.pairplot(df)
plt.show()
```

```
<Figure size 1000x500 with 0 Axes>
```


ApplicantIncome
CoapplicantIncome
LoanAmount
Loan_Amount_Term
0
20000
40000
60000
80000
10000
30000
200
400
600
100
300
500



In [18]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df)
plt.show()
```



FEATURE SELECTION

In [19]:
```
df.drop('Dependents',axis=1,inplace=True)
df.drop('ApplicantIncome',axis=1,inplace=True)
df.drop('CoapplicantIncome',axis=1,inplace=True)
df.drop('LoanAmount',axis=1,inplace=True)
df.drop('Loan_Amount_Term',axis=1,inplace=True)
df.drop('Credit_History',axis=1,inplace=True)
```

ONE HOT ENCODER

In [20]:
```
df1=pd.get_dummies(df,columns=["Gender","Married","Education","Self_Employed","Property_Area","Loan_Status"])
df1.head()
```

Out[20]:

	Loan_ID	Gender_Female	Gender_Male	Married_No	Married_Yes	Education_Graduate	Education_Not Graduate	Self_Employed_No	Self_Employed_Y
0	LP001002	0	1	1	0	1	0	1	
1	LP001003	0	1	0	1	1	0	1	
2	LP001005	0	1	0	1	1	0	0	
3	LP001006	0	1	0	1	0	1	1	
4	LP001008	0	1	1	0	1	0	1	

In [21]:
```
df1.drop('Gender_Female',axis=1,inplace=True)
df1.drop('Married_No',axis=1,inplace=True)
df1.drop('Education_Graduate',axis=1,inplace=True)
df1.drop('Self_Employed_No',axis=1,inplace=True)
df1.drop('Property_Area_Rural',axis=1,inplace=True)
df1.drop('Loan_Status_N',axis=1,inplace=True)
```

In [22]:
```
df1.head()
```

Out[22]:

	Loan_ID	Gender_Male	Married_Yes	Education_Not Graduate	Self_Employed_Yes	Property_Area_Semiurban	Property_Area_Urban	Loan_Status_Y
0	LP001002	1	0	0	0	0	1	1
1	LP001003	1	1	0	0	0	0	0
2	LP001005	1	1	0	1	0	1	1
3	LP001006	1	1	1	0	0	1	1
4	LP001008	1	0	0	0	0	1	1

MODELLING

In [25]:
```
X = df1[["Gender_Male","Married_Yes","Education_Not Graduate","Self_Employed_Yes","Property_Area_Semiurban","Property_
y=df1["Loan_Status_Y"].values
```

In [26]:
```
xtrain,xtest,ytrain,ytest=train_test_split(X,y,test_size=0.2,random_state=0)
```

LOGISTIC REGRESSION MODEL

In [27]:
```
m1=LogisticRegression()
m1.fit(xtrain,ytrain)
yp1=m1.predict(xtest)
```

In [29]:
```
print(" accuracy of logistic model is ",accuracy_score(ytest,yp1))
print(" precision score is ",precision_score(ytest,yp1))
print(" recall is ",recall_score(ytest,yp1))
print(" f1 score is ",f1_score(ytest,yp1))
print("
print(" classification report is ",classification_report(ytest,yp1))
print(" confusion matrix is ",confusion_matrix(ytest,yp1))
```

```
 accuracy of logistic model is  0.7154471544715447
 precision score is  0.7310924369747899
 recall is  0.9666666666666667
 f1 score is  0.832535885167464

 classification report is               precision    recall  f1-score   support

           0       0.25      0.03      0.05        33
           1       0.73      0.97      0.83        90

    accuracy                           0.72       123
   macro avg       0.49      0.50      0.44       123
weighted avg       0.60      0.72      0.62       123

 confusion matrix is  [[ 1 32]
 [ 3 87]]
```

DECISION TREE MODEL

In [30]:
```
m2=DecisionTreeClassifier()
m2.fit(xtrain,ytrain)
yp2=m2.predict(xtest)
```

In [31]:
```
print(" accuracy of decision tree model is ",accuracy_score(ytest,yp2))
print(" precision score is ",precision_score(ytest,yp2))
print(" recall is ",recall_score(ytest,yp2))
```

```
print(" f1 score is ",f1_score(ytest,yp2))
print("                                                                                                                    '
print(" classification report is ",classification_report(ytest,yp2))
print(" confusion matrix is ",confusion_matrix(ytest,yp2))
```

```
 accuracy of decision tree model is  0.7317073170731707
 precision score is  0.7522123893805309
 recall is  0.9444444444444444
 f1 score is  0.8374384236453202

 classification report is               precision    recall  f1-score   support

           0       0.50      0.15      0.23        33
           1       0.75      0.94      0.84        90

    accuracy                           0.73       123
   macro avg       0.63      0.55      0.53       123
weighted avg       0.68      0.73      0.68       123

 confusion matrix is  [[ 5 28]
 [ 5 85]]
```

PREDICTING

In [33]:
```
print("Logistic Regression Predictions ",yp1)
print("_____________________________________________________________________________________________________")
print("Decision Tree predictions ",yp2)
```

```
Logistic Regression Predictions  [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1]

_____________________________________________________________________________________________________
Decision Tree predictions  [1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 0 1 1 0 0
 1 1 1 1 1 1 1 1 1 1 1 1]
```

CONCLUSION

Decision Tree performs with more accuracy than LogisticRegression