Loan Status Prediction

(1) Import Python Libraries

In [25]:
```
import numpy as np
import pandas as pd
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

(2) Loading the Data Set

In [26]:
```
path = r'D:\IITG\portfolio_finance\loan_status\loan_info.csv'
```

In [27]:
```
loan_dataset = pd.read_csv(path)
loan_dataset.head()
```

Out[27]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantInc
0	LP001002	Male	No	0	Graduate	No	
1	LP001003	Male	Yes	1	Graduate	No	
2	LP001005	Male	Yes	0	Graduate	Yes	
3	LP001006	Male	Yes	0	Not Graduate	No	
4	LP001008	Male	No	0	Graduate	No	

In [28]:
```
loan_dataset.describe()
```

Out[28]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Hi
count	614.000000	614.000000	592.000000	600.00000	564
mean	5403.459283	1621.245798	146.412162	342.00000	0
std	6109.041673	2926.248369	85.587325	65.12041	0
min	150.000000	0.000000	9.000000	12.00000	0
25%	2877.500000	0.000000	100.000000	360.00000	1
50%	3812.500000	1188.500000	128.000000	360.00000	1
75%	5795.000000	2297.250000	168.000000	360.00000	1
max	81000.000000	41667.000000	700.000000	480.00000	1

(3) Cleaning the Data Set

In [29]:
```
loan_dataset.shape
```

Out[29]:
```
(614, 13)
```

In [30]:
```
loan_dataset.isnull().sum()
```

Out[30]:
```
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 [31]:
```
loan_dataset = loan_dataset.dropna()
loan_dataset.isnull().sum()
```

Out[31]:
```
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 [32]:
```
loan_dataset.replace({"Loan_Status":{'N':0,'Y':1}},inplace=True)
loan_dataset.head()
```

Out[32]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantInc
1	LP001003	Male	Yes	1	Graduate	No	
2	LP001005	Male	Yes	0	Graduate	Yes	
3	LP001006	Male	Yes	0	Not Graduate	No	
4	LP001008	Male	No	0	Graduate	No	
5	LP001011	Male	Yes	2	Graduate	Yes	

In [33]:
```
loan_dataset['Dependents'].value_counts()
```

Out[33]:
```
0     274
2      85
1      80
3+     41
Name: Dependents, dtype: int64
```

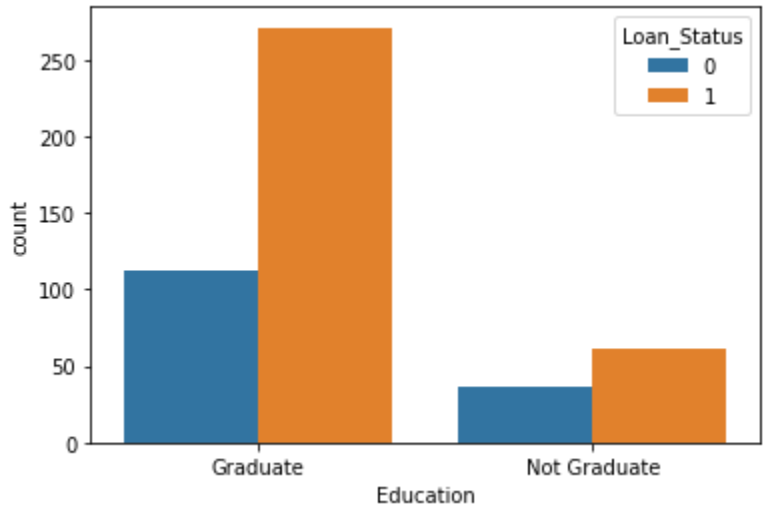
In [34]:
```
loan_dataset = loan_dataset.replace(to_replace='3+', value=4)
loan_dataset['Dependents'].value_counts()
```

Out[34]:
```
0    274
2     85
1     80
4     41
Name: Dependents, dtype: int64
```

(4) Visualizing the Data Set

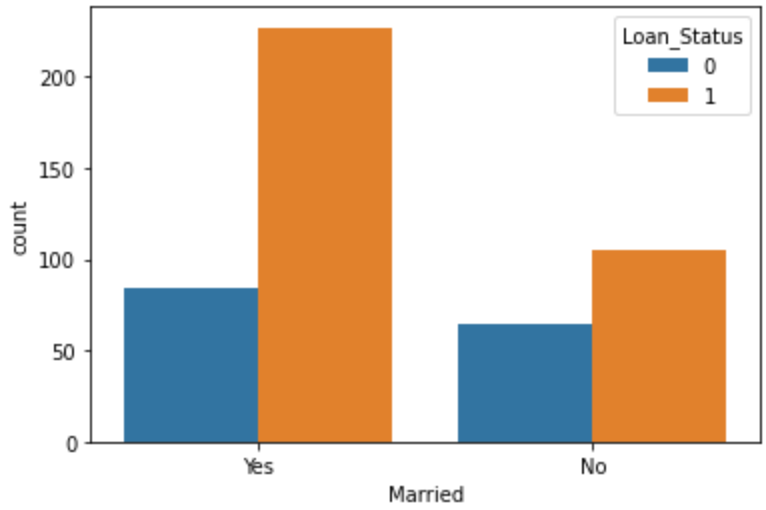
In [35]:
```
sns.countplot(x='Education', hue='Loan_Status', data=loan_dataset)
```

Out[35]: <AxesSubplot:xlabel='Education', ylabel='count'>



In [36]:
```
sns.countplot(x='Married', hue='Loan_Status', data=loan_dataset)
```

Out[36]: <AxesSubplot:xlabel='Married', ylabel='count'>



In [37]:
```
# convert categorical columns to numerical values
loan_dataset.replace({'Married':{'No':0,'Yes':1},'Gender':{'Male':1,'Female'
                      'Property_Area':{'Rural':0,'Semiurban':1,'Urban':2},'E
```

In [38]:
```
loan_dataset.head()
```

Out[38]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantInc
1	LP001003	1	1	1	1	0	
2	LP001005	1	1	0	1	1	
3	LP001006	1	1	0	0	0	
4	LP001008	1	0	0	1	0	
5	LP001011	1	1	2	1	1	

(5) Building the Model

Split the Data

In [39]:
```
X = loan_dataset.drop(columns=['Loan_ID','Loan_Status'],axis=1)
Y = loan_dataset['Loan_Status']
```

In [40]:
```
X_train, X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.1,stratify
```

Create and Fit the Model

In [41]:
```
classifier = svm.SVC(kernel='linear')
classifier.fit(X_train,Y_train)
```

Out[41]:
```
SVC(kernel='linear')
```

(6) Evaluating the Model

In [42]:
```
X_train_prediction = classifier.predict(X_train)
X_train_prediction[:5]
```

Out[42]:
```
array([0, 1, 1, 1, 1], dtype=int64)
```

In [43]:
```
training_data_accuray = accuracy_score(X_train_prediction,Y_train)
print('Accuracy on training data : ', training_data_accuray)
```

```
Accuracy on training data :  0.7986111111111112
```

In [44]:
```
X_test_prediction = classifier.predict(X_test)
X_test_prediction[:5]
```

Out[44]:
```
array([1, 1, 1, 1, 1], dtype=int64)
```

In [45]:
```
test_data_accuray = accuracy_score(X_test_prediction,Y_test)
print('Accuracy on test data : ', test_data_accuray)
```

```
Accuracy on test data :  0.8333333333333334
```