Importing the dependencies

In [1]:

```
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
```

Data collection and processing

In [2]:

```
data = pd.read_csv("dataset.csv")
```

In [3]:

```
data.head()
```

Out[3]:

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

In [6]:

```
num_of_rows, num_of_features = data.shape
num_of_rows, num_of_features
```

Out[6]:

```
(614, 13)
```

In [7]:

```
data.describe()
```

Out[7]:

	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 [8]:

```
data.isnull().sum()
```

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 [11]:

```
data = data.dropna()
```

In [12]:

```
data.isna().sum()
```

Out[12]:

```
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 [14]:

```
data.replace({"Loan_Status": {"N":0, "Y":1}}, inplace=True)
```

In [15]:

```
data.head()
```

Out[15]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Lo
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	

In [16]:

```
data["Dependents"].value_counts()
```

Out[16]:

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

In [17]:

```
data = data.replace(to_replace="3+", value=4)
```

In [18]:

```
data["Dependents"].value_counts()
```

Out[18]:

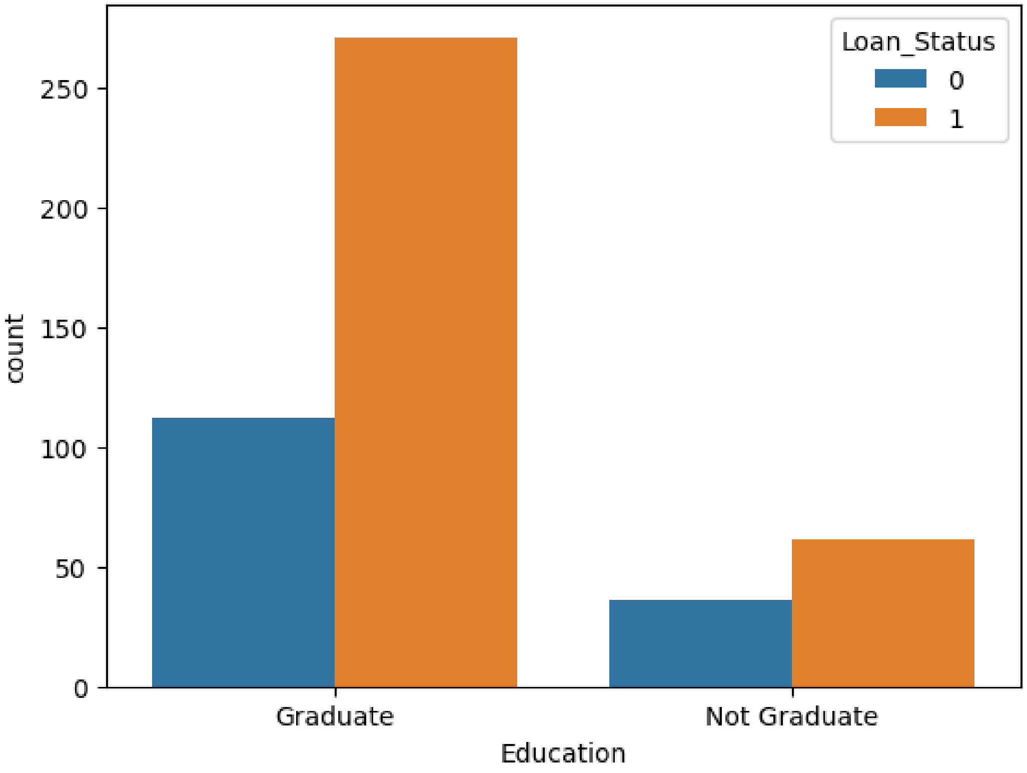
```
0    274
2     85
1     80
4     41
Name: Dependents, dtype: int64
```

In [21]:

```
sns.countplot(x="Education", hue="Loan_Status", data=data)
```

Out[21]:

```
<AxesSubplot:xlabel='Education', ylabel='count'>
```

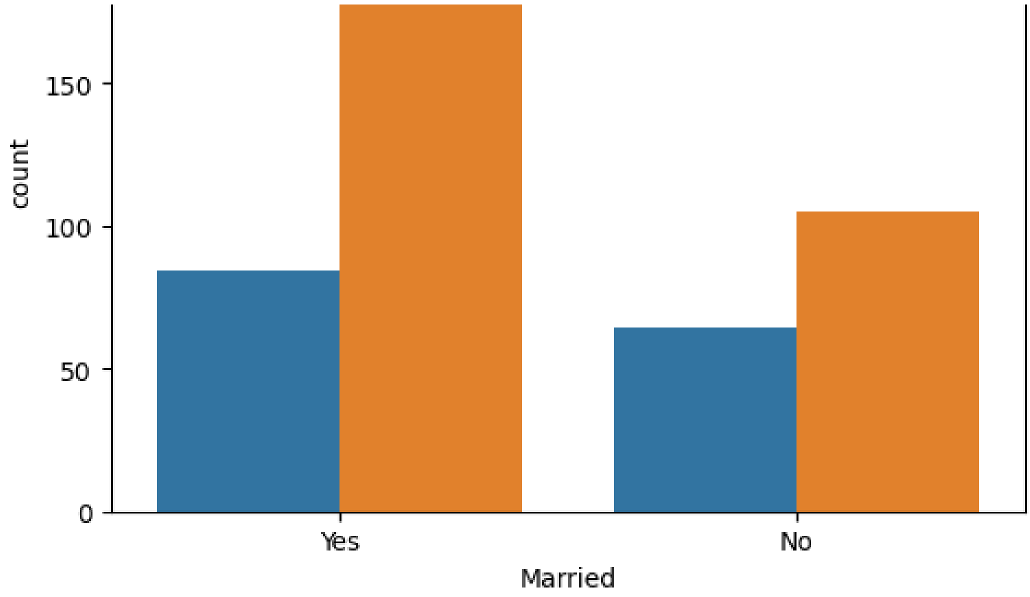


In [23]:

```
sns.countplot(x="Married", hue="Loan_Status", data=data)
```

Out[23]:

```
<AxesSubplot:xlabel='Married', ylabel='count'>
```



In [24]:

```
data.replace(
    {
        "Married": {"No":0, "Yes":1},
        "Gender": {"Male":1, "Female":0},
        "Self_Employed": {"No":0, "Yes":1},
        "Property_Area": {"Rural":0, "Semiurban":1, "Urban":2},
        "Education": {"Graduate":1, "Not Graduate":0}
    },
    inplace=True
)
```

In [25]:

```
data.head()
```

Out[25]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Lo
1	LP001003	1	1	1	1	0	4583	1508.0	128.0	
2	LP001005	1	1	0	1	1	3000	0.0	66.0	
3	LP001006	1	1	0	0	0	2583	2358.0	120.0	
4	LP001008	1	0	0	1	0	6000	0.0	141.0	
5	LP001011	1	1	2	1	1	5417	4196.0	267.0	

In [27]:

```
X = data.drop(columns=['Loan_ID','Loan_Status'],axis=1)
Y = data['Loan_Status']
```

In [28]:

```
X_train, X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.1,stratify=Y,random_st
ate=2)
```

In [29]:

```
classifier = svm.SVC(kernel='linear')
```

In [30]:

```
classifier.fit(X_train,Y_train)
```

Out[30]:

```
▾ SVC
SVC(kernel='linear')
```

In [31]:

```
X_train_prediction = classifier.predict(X_train)
training_data_accuray = accuracy_score(X_train_prediction,Y_train)
```

In [32]:

```
print('Accuracy on training data : ', training_data_accuray)
```

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

In [33]:

```
X_test_prediction = classifier.predict(X_test)
test_data_accuray = accuracy_score(X_test_prediction,Y_test)
```

In [34]:

```
print('Accuracy on test data : ', test_data_accuray)
```

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

In [ ]:

```

```