In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
loan_dataset = pd.read_csv(r"C:\Users\s323\Desktop\Gatherings\Data Science\Datasets
```

In [3]:
```
loan_dataset.head()
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplic
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 [4]:
```
loan_dataset.shape
```

Out[4]:
```
(614, 13)
```

In [6]:
```
loan_dataset.describe()
```

Out[6]:

	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 [7]:
```
loan_dataset.columns
```

Out[7]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [9]:
```
loan_dataset.isnull().any()
```

Out[9]:
```
Loan_ID              False
Gender                True
Married               True
Dependents            True
Education            False
Self_Employed         True
ApplicantIncome      False
CoapplicantIncome    False
LoanAmount            True
Loan_Amount_Term      True
Credit_History        True
Property_Area        False
Loan_Status          False
dtype: bool
```

In [15]:
```
loan_dataset.isnull().sum()
```

Out[15]:
```
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 [16]:
```
# drropping the missing values since we have cateogorical value
loan_dataset = loan_dataset.dropna()
```

In [19]:
```
loan_dataset.isnull().sum()
```

Out[19]:
```
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
```

Label Encoding

- Converting cateogrical data/Textual data to numerical data
- Y- 1, N-0

In [29]:
```
loan_dataset.replace({"Loan_Status":{"N":0,"Y":1}},inplace = True)
```

In [30]:
```
loan_dataset.head()
```

Out[30]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplic
1	LP001003	1	1	1	1	0	4583	
2	LP001005	1	1	0	1	1	3000	
3	LP001006	1	1	0	0	0	2583	
4	LP001008	1	0	0	1	0	6000	
5	LP001011	1	1	2	1	1	5417	

In [31]:
```
## Dependent col values
loan_dataset["Dependents"].value_counts()
```

Out[31]:
```
0    274
2     85
1     80
4     41
Name: Dependents, dtype: int64
```

In [32]:
```
# Replacing 3+ values with 4 becuase - 3+ is not good for our dataset
loan_dataset=loan_dataset.replace(to_replace = "3+", value =4 )
```

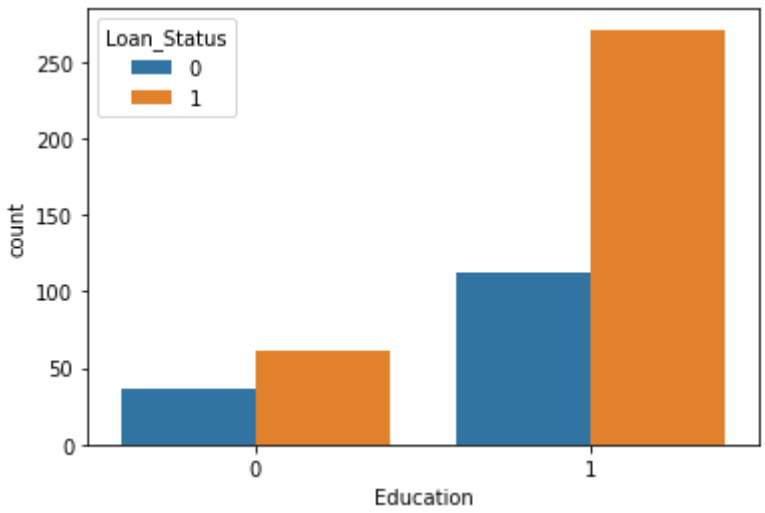
In [33]:
```
loan_dataset['Dependents'].value_counts()
```

Out[33]:
```
0    274
2     85
1     80
4     41
Name: Dependents, dtype: int64
```

Data Visulaization

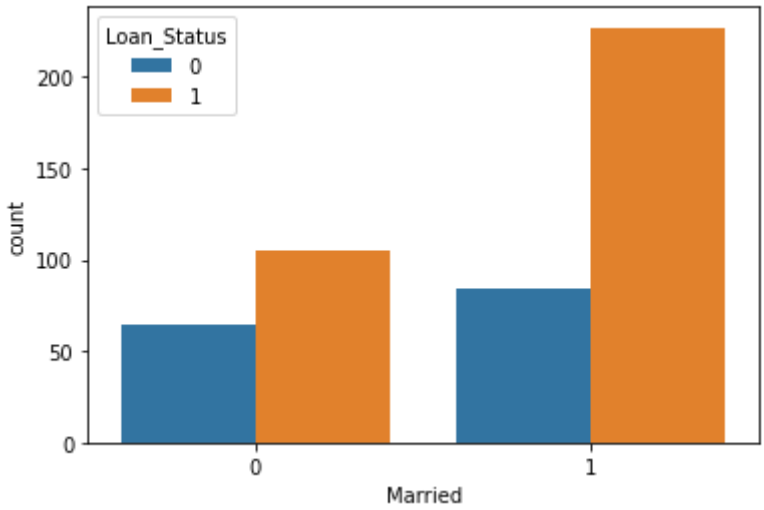
In [34]:
```
# education & Loan Status
sns.countplot(x="Education", hue="Loan_Status",data =loan_dataset)
```

Out[34]:
```
<AxesSubplot:xlabel='Education', ylabel='count'>
```



In [35]:
```
# marital status & Loan Status
sns.countplot(x="Married",hue="Loan_Status",data= loan_dataset)
```

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



Convert categorical columns to numerical values

In [36]:
```
loan_dataset.replace({"Married":{"No":0,"Yes":1},"Gender":{"Male":1,"Female":0},"S
                      "Property_Area":{"Rural":0,"Semiurban":1,"Urban":2},"Educati
```

In [37]:
```
loan_dataset.head()
```

Out[37]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplic
1	LP001003	1	1	1	1	0	4583	
2	LP001005	1	1	0	1	1	3000	
3	LP001006	1	1	0	0	0	2583	
4	LP001008	1	0	0	1	0	6000	
5	LP001011	1	1	2	1	1	5417	

Splitting data into X and Y

- data -x, label-y

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

In [42]:
```
print (X)
print (Y)
```

```
     Gender  Married Dependents  Education  Self_Employed  ApplicantIncome  \
1         1        1           1          1              0             4583
2         1        1           0          1              1             3000
3         1        1           0          0              0             2583
4         1        0           0          1              0             6000
5         1        1           2          1              1             5417
..      ...      ...         ...        ...            ...              ...
609       0        0           0          1              0             2900
610       1        1           4          1              0             4106
611       1        1           1          1              0             8072
612       1        1           2          1              0             7583
613       0        0           0          1              1             4583

     CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
1               1508.0       128.0             360.0             1.0
2                  0.0        66.0             360.0             1.0
3               2358.0       120.0             360.0             1.0
4                  0.0       141.0             360.0             1.0
5               4196.0       267.0             360.0             1.0
..                 ...         ...               ...             ...
609                0.0        71.0             360.0             1.0
610                0.0        40.0             180.0             1.0
611              240.0       253.0             360.0             1.0
612                0.0       187.0             360.0             1.0
613                0.0       133.0             360.0             0.0

     Property_Area
1                0
2                2
3                2
4                2
5                2
..             ...
609              0
610              0
611              2
612              2
613              1

[480 rows x 11 columns]
1      0
2      1
3      1
4      1
5      1
      ..
609    1
610    1
611    1
612    1
613    0
Name: Loan_Status, Length: 480, dtype: int64
```

Splitting the data into train and test

In [44]:
```
from sklearn.model_selection import train_test_split
```

In [45]:
```
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.1,stratify = Y, r
```

In [46]:
```
print(X.shape,x_train.shape,x_test.shape)
```

```
(480, 11) (432, 11) (48, 11)
```

Training the model: Support Vector Machine

In [48]:
```
from sklearn import svm
```

In [49]:
```
classifier = svm.SVC(kernel="linear")
# SVC = Classifier, other type of problem is regression
```

In [50]:
```
classifier.fit(x_train,y_train)
```

Out[50]:
```
SVC(kernel='linear')
```

Model Evaluation

In [51]:
```
from sklearn.metrics import accuracy_score
```

In [53]:
```
trained_predictions = classifier.predict(x_train)
training_data_accuracy = accuracy_score(trained_predictions,y_train)
```

In [54]:
```
training_data_accuracy
# we want training data score, it is not important though to check overfitting
```

Out[54]:
```
0.7986111111111112
```

In [55]:
```
tested_prediction = classifier.predict(x_test)
testing_data_accuracy = accuracy_score(tested_prediction,y_test)
```

In [56]:
```
testing_data_accuracy
```

Out[56]:
```
0.8333333333333334
```