Importing the Libraries

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

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score
```

Data Collection and Processing

In [3]:
```
df=pd.read_csv("loan_dataset.csv")
```

In [4]:
```
df
```

Out[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
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 [5]:
```
# Print the first 5 rows of the DataFrame
df.head()
```

Out[5]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
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 [6]:
```
# Print the last 5 row of the DataFrame
df.tail()
```

Out[6]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
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

In [7]:
```
# Number of Rows and columns
df.shape
```

Out[7]:
```
(614, 13)
```

In [8]:
```
# Statistics Measure
df.describe()
```

Out[8]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Histo
count	614.000000	614.000000	592.000000	600.00000	564.0000
mean	5403.459283	1621.245798	146.412162	342.00000	0.8421
std	6109.041673	2926.248369	85.587325	65.12041	0.3648
min	150.000000	0.000000	9.000000	12.00000	0.0000
25%	2877.500000	0.000000	100.000000	360.00000	1.0000
50%	3812.500000	1188.500000	128.000000	360.00000	1.0000
75%	5795.000000	2297.250000	168.000000	360.00000	1.0000
max	81000.000000	41667.000000	700.000000	480.00000	1.0000

In [9]:
```
# Number of Missing values in each columns
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
```

In [10]:
```
# Dropping the missing values
df=df.dropna()
```

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          0
Loan_Amount_Term    0
Credit_History      0
Property_Area       0
Loan_Status         0
dtype: int64
```

In [12]:
```
# Label Encoder
df.replace({"Loan_Status": {"N": 0,"Y": 1}},inplace=True)
```

In [13]:
```
df
```

Out[13]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
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
5	LP001011	Male	Yes	2	Graduate	Yes	5417
...	...	...	...	...	...	...	...
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

480 rows × 13 columns

In [14]:
```
df["Dependents"].value_counts()
```

Out[14]:
```
Dependents
0     274
2      85
1      80
3+     41
Name: count, dtype: int64
```

In [15]:
```
# Replacing the value of 3+ to 4
df=df.replace(to_replace="3+",value=4)
```

In [16]:
```
df
```

Out[16]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
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
5	LP001011	Male	Yes	2	Graduate	Yes	5417
...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900
610	LP002979	Male	Yes	4	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

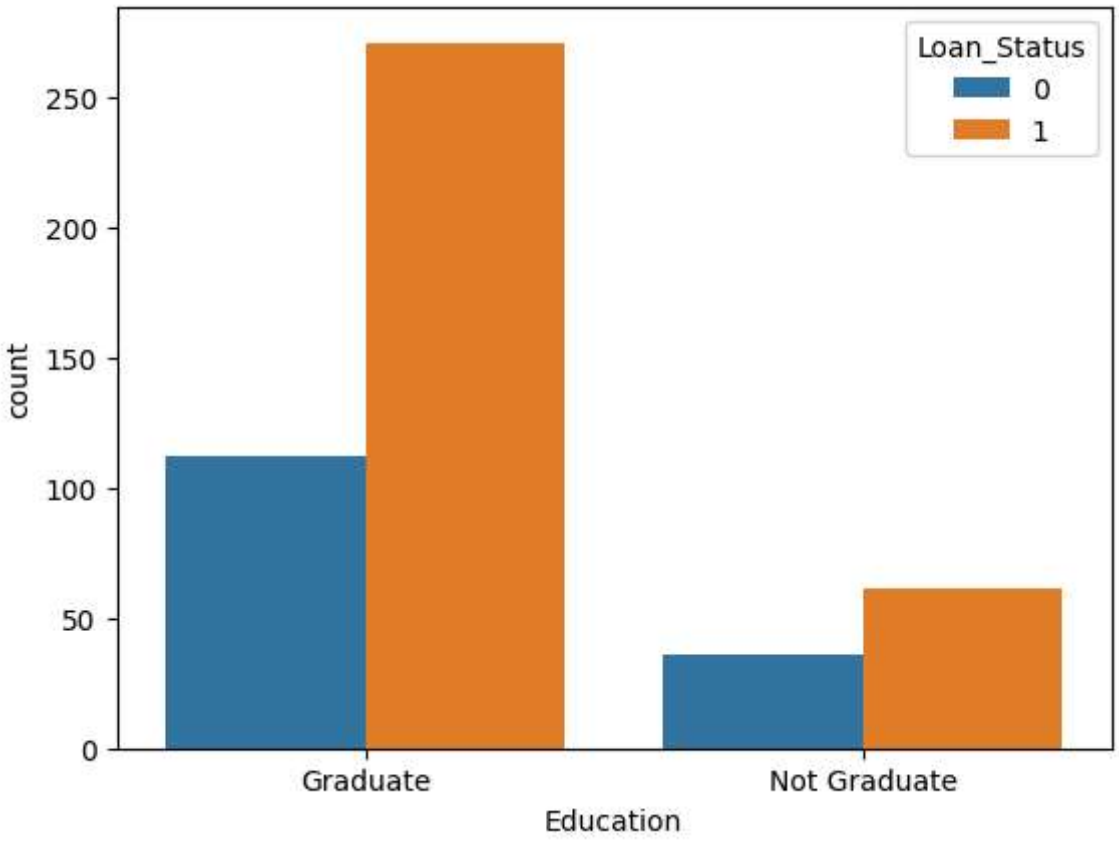
480 rows × 13 columns

In [17]:
```
df["Dependents"].value_counts()
```

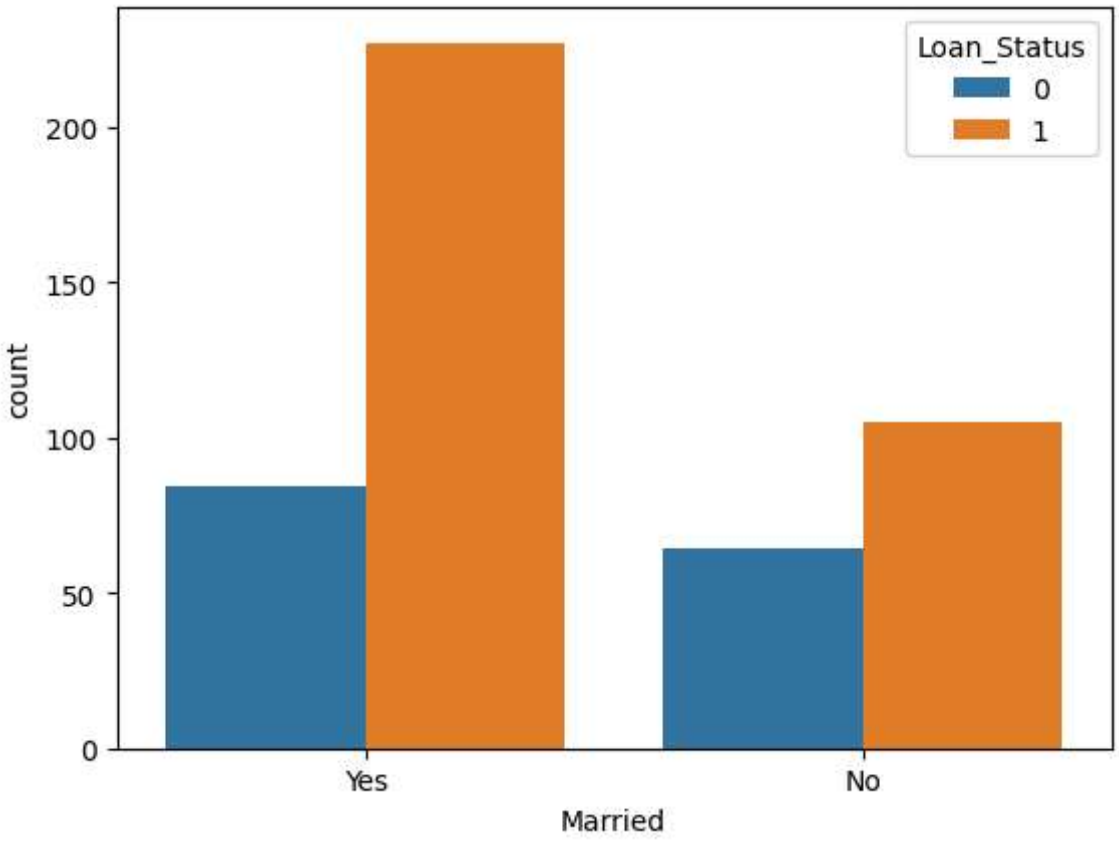
Out[17]:
```
Dependents
0    274
2     85
1     80
4     41
Name: count, dtype: int64
```

Data Visualization

In [18]:
```
# education & Loan_status
sns.countplot(data=df,x="Education",hue="Loan_Status")
plt.show()
```



In [19]:
```
# Marital status & Loan_status
sns.countplot(data=df,x="Married",hue="Loan_Status")
plt.show()
```



In [20]:
```
# Convert the category column into numerical values
df.replace({"Married":{"No":0,"Yes":1},"Gender":{"Male":1,"Female": 0},"Self_Employ
```

In [21]:
```
df.head()
```

Out[21]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
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 [22]:
```
# splitting and the label
X=df.drop(columns=["Loan_ID","Loan_Status"],axis=1)
Y=df["Loan_Status"]
```

In [23]:
```
X
```

Out[23]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
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	

480 rows × 11 columns

In [24]:
```
Y
```

Out[24]:
```
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
```

Train,Test,Split

In [25]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=1)
```

In [26]:
```
print(X.shape,X_train.shape,X_test.shape)
```

```
(480, 11) (336, 11) (144, 11)
```

In [33]:
```
from sklearn.svm import LinearSVC
```

In [34]:
```
svc=LinearSVC()
```

Training tha mode: Support Vector Machine Model

In [35]:
```
svc.fit(X_train,Y_train)
```

Out[35]:
```
LinearSVC()
```

In [36]:
```
# Training the support Vector Machine Model
svc.fit(X_train,Y_train)
```

Out[36]:
```
LinearSVC()
```

In [37]:
```
svc.score(X_train,Y_train)
```

Out[37]:
```
0.8244047619047619
```

Model Evaluation

In [42]:
```
# accuracy score on training data
X_predict=svc.predict(X_train)
training_data_accuracy= accuracy_score(X_predict,Y_train)
```

In [43]:
```
print("Accuracy on training data:",training_data_accuracy)
```

```
Accuracy on training data: 0.8244047619047619
```

In [40]:
```
# accuracy score on testing data
X_predtest=svc.predict(X_test)
testing_data_accuracy= accuracy_score(X_predtest,Y_test)
```

In [41]:
```
print("Accuracy on testing data:",testing_data_accuracy)
```

```
Accuracy on testing data: 0.7777777777777778
```

In [ ]: