```
#IMPORTING the dependies
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
```

```
#loading the dataset to pandas dataframe
loan_dataset= pd.read_csv('loan_prediction_data.csv')
```

```
type(loan_dataset)
```

```
pandas.core.frame.DataFrame
def __init__(data=None, index: Axes | None=None, columns: Axes | None=None, dtype: Dtype |
None=None, copy: bool | None=None) -> None
```

```
Notice that the inferred dtype is int64.

>>> df.dtypes
col1    int64
col2    int64
```

```
#printing the first 5 rows of the dataset
loan_dataset.head()
```

	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

Next steps: Generate code with `loan_dataset` View recommended plots New interactive sheet

```
#number of rows and columns
loan_dataset.shape
```

```
(614, 13)
```

```
#statistical measures
loan_dataset.describe()
#we will get numerical data only
```

	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

```
#number of missing values
loan_dataset.isnull().sum()
```

	0
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

```
#dropping the missing values
loan_dataset = loan_dataset.dropna()
```

```
loan_dataset.isnull().sum()
```

	0
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

```
#lable incoding
# 1 for yes and 0 for no
new_loan_dataset = loan_dataset.replace({"Loan_Status": {'N': 0, 'Y': 1}})
```

```
#printing the head
new_loan_dataset.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
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
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0

Next steps: Generate code with new_loan_dataset | View recommended plots | New interactive sheet

```
#dependent column values
new_loan_dataset['Dependents'].value_counts()
```

Dependents	count
0	274
2	85
1	80
3+	41

```
#replace 3+ with 4
new_loan_dataset = new_loan_dataset.replace(to_replace='3+',value=4)
```

```
#dependent values
new_loan_dataset['Dependents'].value_counts()
```

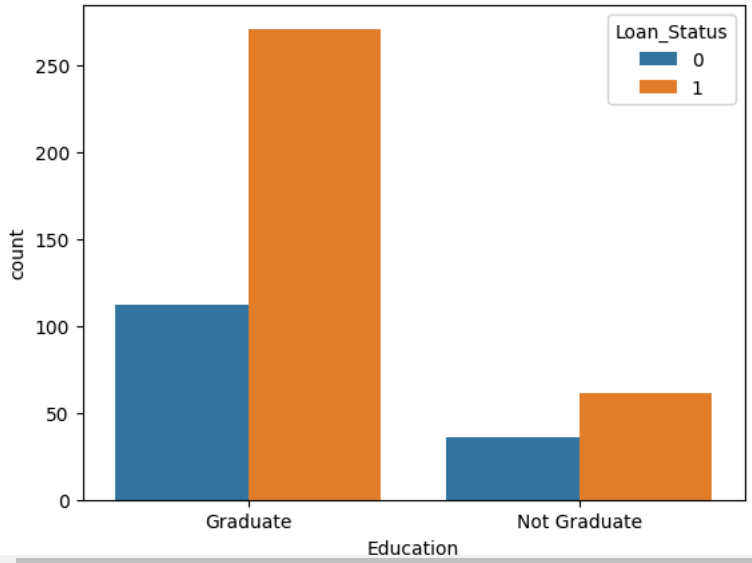
Dependents	count
0	274
2	85
1	80
4	41

Start coding or generate with AI.

Data Visualization

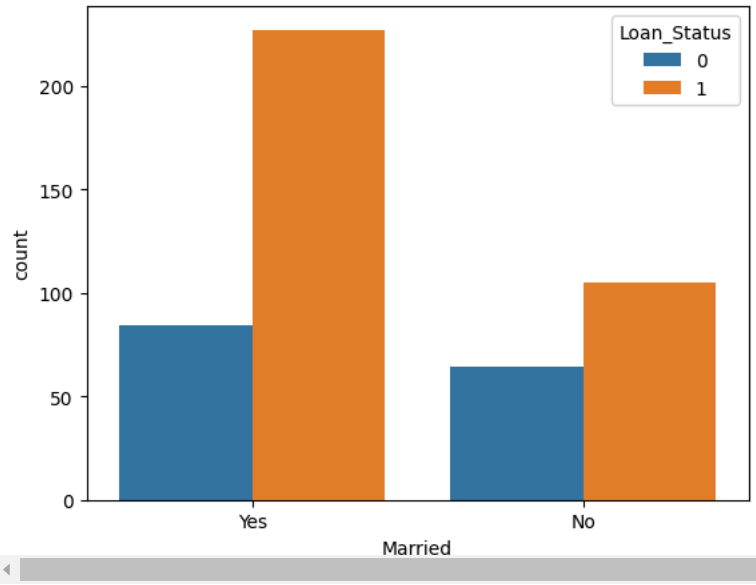
```
#education and loan status
#graph using seaborn ie sns
sns.countplot(x='Education',hue='Loan_Status',data=new_loan_dataset)
```

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




```
#marital status & loan status
sns.countplot(x='Married',hue='Loan_Status',data=new_loan_dataset)
```

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




```
#replace with  0 and 1
new_loan_dataset.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 Gradua
```

```
new_loan_dataset.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
1	LP001003	1	1	1	1	0	4583	1508.0	128.0	360.0
2	LP001005	1	1	0	1	1	3000	0.0	66.0	360.0
3	LP001006	1	1	0	0	0	2583	2358.0	120.0	360.0
4	LP001008	1	0	0	1	0	6000	0.0	141.0	360.0
5	LP001011	1	1	2	1	1	5417	4196.0	267.0	360.0

Next steps: Generate code with new_loan_dataset | View recommended plots | New interactive sheet

```
#seprating the data and lable
X= new_loan_dataset.drop(columns=['Loan_ID','Loan_Status'],axis=1)
Y= new_loan_dataset['Loan_Status']
```

```
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
```

```
#train test split
x_train, x_test, y_train, y_test= train_test_split(X,Y,test_size=0.1,stratify=Y, random_state=2)
```

```
print(X.shape,x_train.shape, x_train.shape)
```

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

Start coding or generate with AI.

Support Vector Machine Model

```
#we have to use classification model
classifier= svm.SVC(kernel= 'linear')
```

```
#training the model
classifier.fit(x_train, y_train)
```

```
SVC
SVC(kernel='linear')
```

Start coding or generate with AI.

Model Evaluation

```
#accuracy score on training data
x_train_prediction= classifier.predict(x_train)
```

```
training_data_accuracy=accuracy_score(x_train_prediction,y_train)
print('Accuracy score on training data =', training_data_accuracy)
```

```
Accuracy score on training data = 0.7986111111111112
```

```
#accuracy score on test data
x_test_prediction= classifier.predict(x_test)
test_data_accuracy=accuracy_score(x_test_prediction,y_test)
print('Accuracy score =', test_data_accuracy)
```

```
Accuracy score = 0.8333333333333334
```

Start coding or generate with AI.

make predictions

```
# using this LP001032   Male    No  0   Graduate    No  4950    0   125 360 1   Urban   Y
input_data=(1,0,0,1,0,4950,0,125,360,1,2)
```

```
#changing the input data to numpy array
ipdata_nparray=np.asarray(input_data)
```

```
#reshape the array for one instance
ipdata_nparray=ipdata_nparray.reshape(1,-1)
```

```
prediction= classifier.predict(ipdata_nparray)
print(prediction)
if(prediction[0]==0):
    print('Loan not approved')

else:
    print('Loan approved')
```

```
[1]
Loan approved
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:465: UserWarning: X does not have valid feature names, but SVC was fitted wi
  warnings.warn(
```

Start coding or generate with AI.