Importing the Dependencies

```
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

```
#loading the dataset to pandas Dataframe
loan_dataset = pd.read_csv('/content/train_u6lujuX_CVtuZ9i.csv')
```

```
type(loan_dataset)
```

```
pandas.core.frame.DataFrame
```

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

1 to 5 of 5 entries Filter

x	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
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	

Show 25 per page
Like what you see? Visit the data table notebook to learn more about interactive tables.

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

```
(614, 13)
```

```
#statistical measures
loan_dataset.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Histo
count	614.000000	614.000000	592.000000	600.00000	564.000(
mean	5403.459283	1621.245798	146.412162	342.00000	0.842'
std	6109.041673	2926.248369	85.587325	65.12041	0.364{
min	150.000000	0.000000	9.000000	12.00000	0.000(
25%	2877.500000	0.000000	100.000000	360.00000	1.000(
50%	3812.500000	1188.500000	128.000000	360.00000	1.000(
75%	5795.000000	2297.250000	168.000000	360.00000	1.000(
max	81000.000000	41667.000000	700.000000	480.00000	1.000(

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

```
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
```

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


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

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


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

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncom
1	LP001003	Male	Yes	1	Graduate	No	458
2	LP001005	Male	Yes	0	Graduate	Yes	300
3	LP001006	Male	Yes	0	Not Graduate	No	258
4	LP001008	Male	No	0	Graduate	No	600
5	LP001011	Male	Yes	2	Graduate	Yes	541

```
#Dependent column values
loan_dataset['Dependents'].value_counts()
```

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

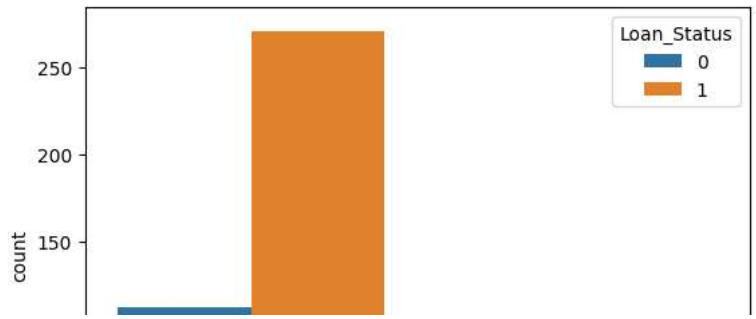
```
#replacing the value of 3+ to 4
loan_dataset = loan_dataset.replace(to_replace='3+', value=4)


#Dependent values
loan_dataset['Dependents'].value_counts()
```

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

Data Visualization

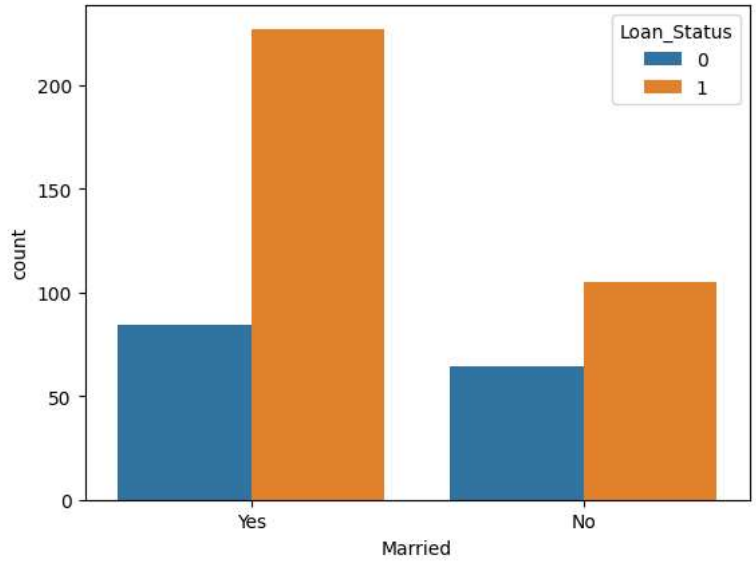
```
# education & loan status
sns.countplot(x='Education' ,hue='Loan_Status',data=loan_dataset)
```

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




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

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




```
# converting categorical columns to numerical values
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 Graduate':0}},inplace=True)
```

```
loan_dataset.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncom
1	LP001003	1	1	1	1	0	458
2	LP001005	1	1	0	1	1	300
3	LP001006	1	1	0	0	0	258
4	LP001008	1	0	0	1	0	600
5	LP001011	1	1	2	1	1	541

```
# separating the data and label
X = loan_dataset.drop(columns=['Loan_ID','Loan_Status'],axis=1)
Y = 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_test.shape)
```

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

Training the model: Support Vector Machine

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

```
#training the support vector machine model
classifier.fit(X_train,Y_train)
```

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

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 on training data : ',training_data_accuracy)
```

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

```
# accuracy score on training data
X_test_prediction = classifier.predict(X_test)
training_data_accuracy = accuracy_score(X_test_prediction,Y_test)
```

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

```
Accuracy on training data :  0.8333333333333334
```

Colab paid products - Cancel contracts here