```
import pandas as pd
import numpy as np
from matplotlib import pyplot as pl
import matplotlib.pyplot as plt
import seaborn as sns
```

```
loan=pd.read_csv('/content/loan status prediction.csv')
```

```
loan.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncom
0	LP001002	Male	No	0	Graduate	No	584
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

```
loan.shape
```

```
(614, 13)
```

```
loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12   Loan_status       614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

```
loan.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Hi
count	614.000000	614.000000	592.000000	600.00000	564.0
mean	5403.459283	1621.245798	146.412162	342.00000	0.8
std	6109.041673	2926.248369	85.587325	65.12041	0.3
min	150.000000	0.000000	9.000000	12.00000	0.0
25%	2877.500000	0.000000	100.000000	360.00000	1.0
50%	3812.500000	1188.500000	128.000000	360.00000	1.0
75%	5795.000000	2297.250000	168.000000	360.00000	1.0
max	81000.000000	41667.000000	700.000000	480.00000	1.0

```
loan.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
```

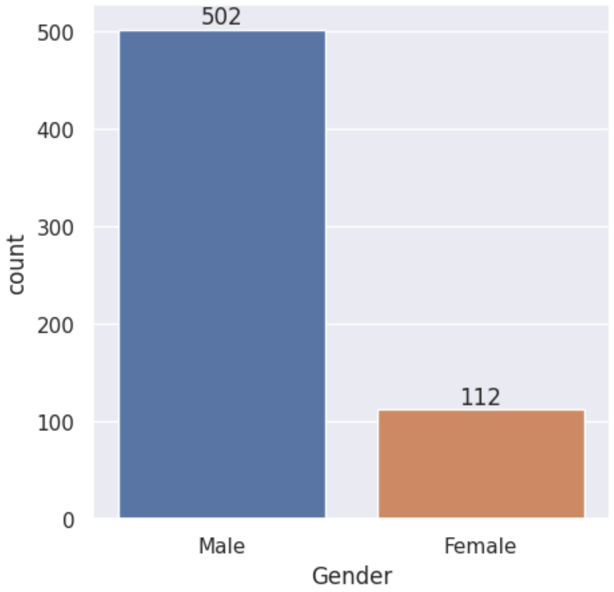
lets start with filling none values

```
loan['Gender'].fillna(loan['Gender'].mode()[0],inplace=True)
loan['Married'].fillna(loan['Married'].mode()[0],inplace=True)
loan['Dependents'].fillna(loan['Dependents'].mode()[0],inplace=True)
loan['Self_Employed'].fillna(loan['Self_Employed'].mode()[0],inplace=True)
loan['Loan_Amount_Term'].fillna(loan['Loan_Amount_Term'].mode()[0],inplace=True)
loan['Credit_History'].fillna(loan['Credit_History'].mode()[0],inplace=True)
loan['LoanAmount'].fillna(loan['LoanAmount'].mean(),inplace=True)
```

```
loan.Dependents.replace(to_replace='3+' , value=4,inplace=True)
```

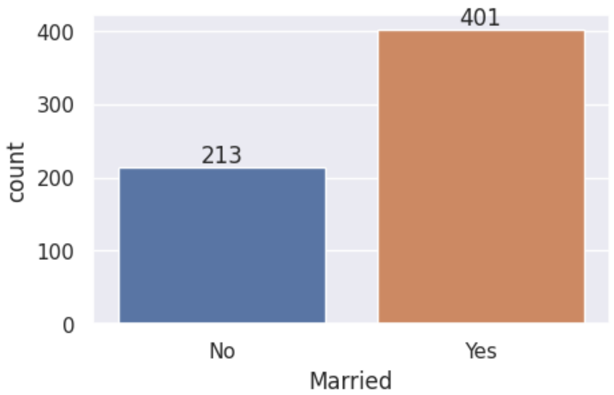
```
***Exploratory Data Analysis***
```

```
#explore the categorical column "Gender".
ax=sns.countplot(x='Gender', data=loan)
sns.set(rc={'figure.figsize':(5,5)})
for bars in ax.containers:
    ax.bar_label(bars)
```



The majority of the applicant is male and a handful is female. Form this data we can target more male for the loans for genrating more business.

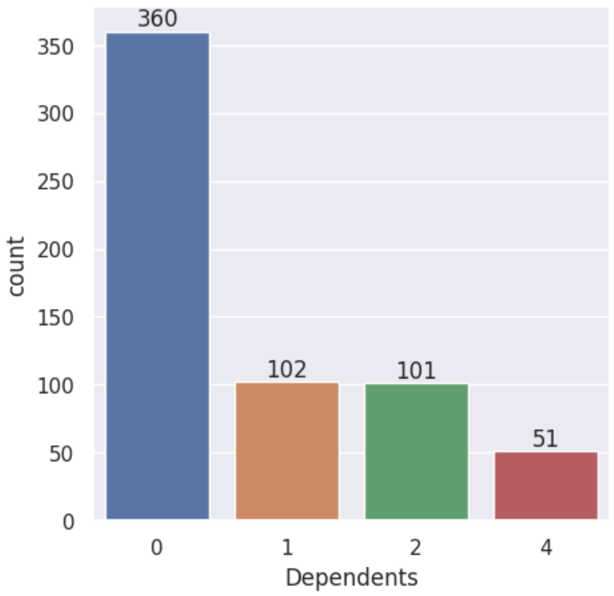
```
# explore the categorical column "Married".
ax=sns.countplot(x='Married',data=loan)
sns.set(rc={'figure.figsize':(5,5)})
for bars in ax.containers:
    ax.bar_label(bars)
```



The majority of the applicants are married.

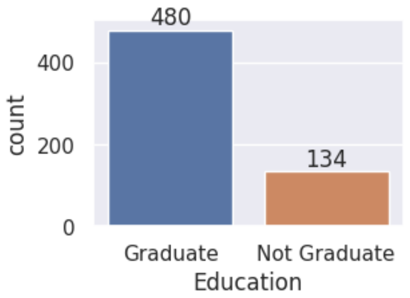
```
# explore the categorical column "Dependents".
ax=sns.countplot(x='Dependents',data=loan)
sns.set(rc={'figure.figsize':(5,5)})
for bars in ax.containers:
```

```
    ax.bar_label(bars)
```

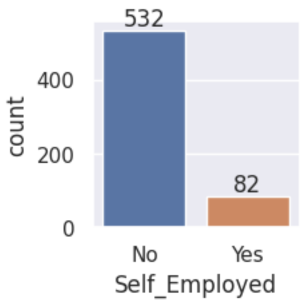


The majority of the applicants have zero dependents, around 100 applicants have one or two dependents and only a few have more than three dependents.

```
# explore the categorical column "Education".
ax=sns.countplot(x='Education',data=loan)
sns.set(rc={'figure.figsize':(3,3)})
for bars in ax.containers:
    ax.bar_label(bars)
```

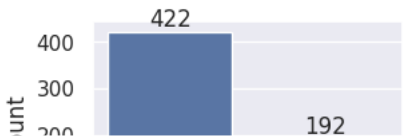



```
## explore the categorical column 'Self Employed'
ax=sns.countplot(x='Self_Employed',data=loan)
sns.set(rc={'figure.figsize':(2,2)})
for bars in ax.containers:
    ax.bar_label(bars)
```



Around 90 applicants are either freelancers or run a business.

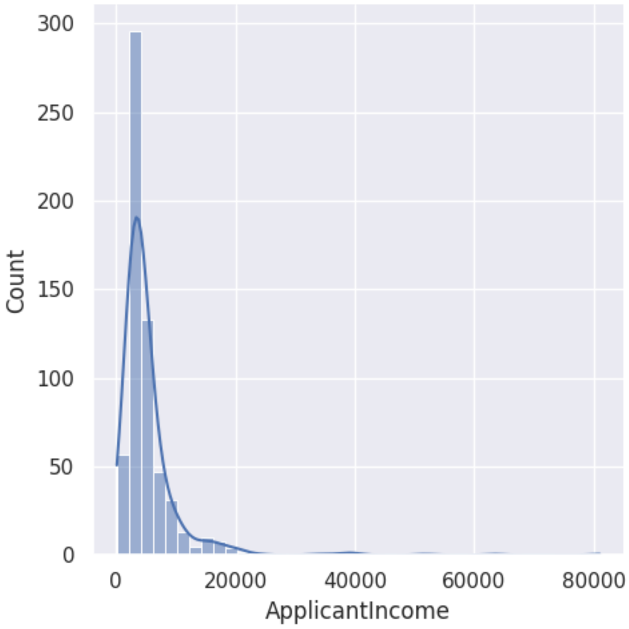
```
## explore the categorical column "Loan Status".
ax=sns.countplot(x=' Loan_status',data=loan)
sns.set(rc={'figure.figsize':(2,2)})
for bars in ax.containers:
    ax.bar_label(bars)
```



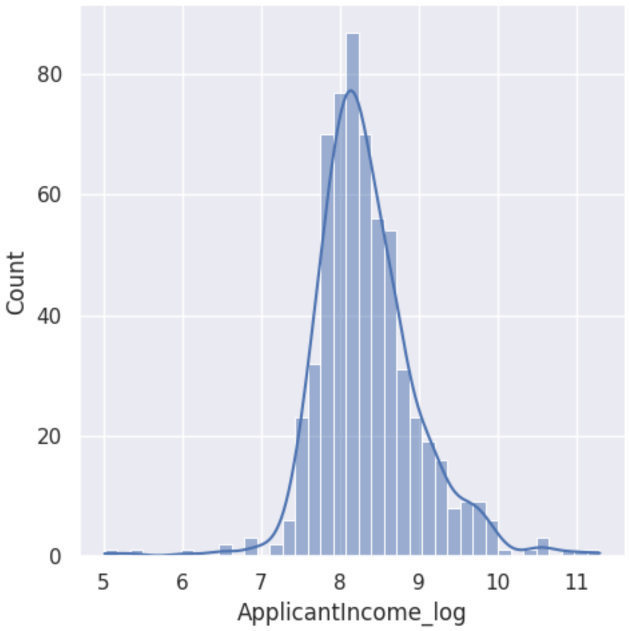
Around 400 loans are accepted and 200 loans are rejected. Its shows the 2:1 ratio.

Numerical attributes visualization

```
## explore the categorical column "Applicant Income".
sns.displot(x='ApplicantIncome',data=loan,kde=True,bins=40)
sns.set(rc={'figure.figsize':(1,1)})
```

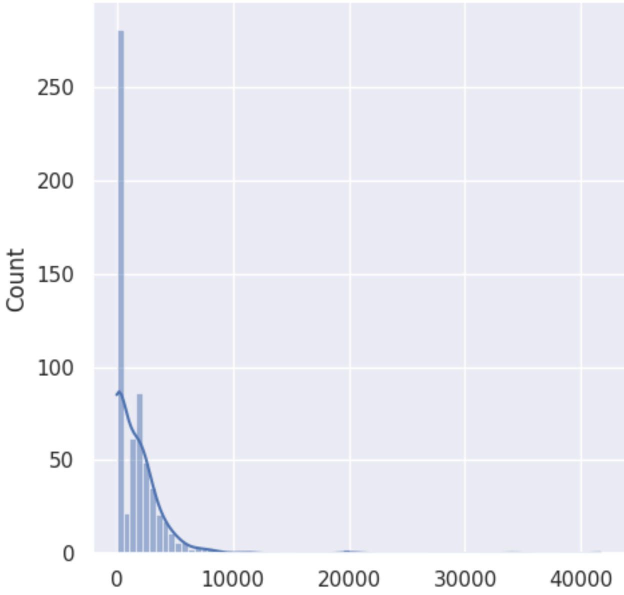



```
# apply log transformation to the attribute
loan['ApplicantIncome_log']=np.log(loan['ApplicantIncome'])
sns.displot(loan.ApplicantIncome_log,kde=True)
sns.set(rc={'figure.figsize':(2,0)})
```



Now we can observe a Normal distribution in a form of a Bell Curve.

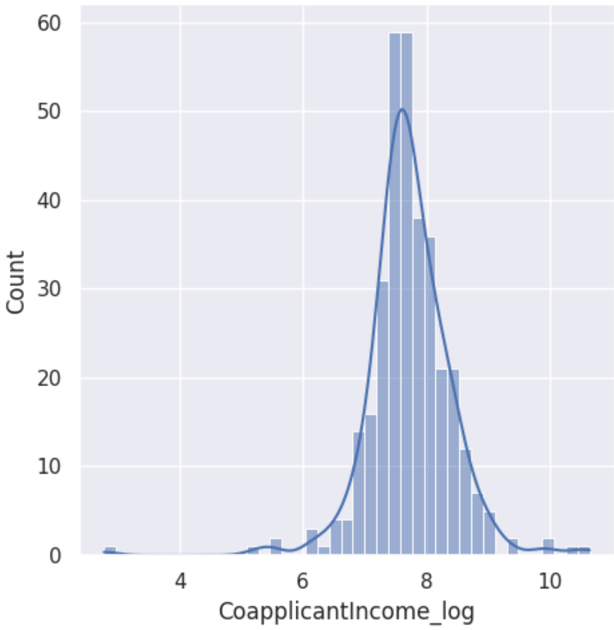
```
# To display the column "Co-applicant Income".
sns.displot(loan.CoapplicantIncome,kde=True,bins=75)
#sns.displot(loan["CoapplicantIncome"])
sns.set(rc={'figure.figsize':(2,2)})
```



We have to normalize this graph as well, using log fuction

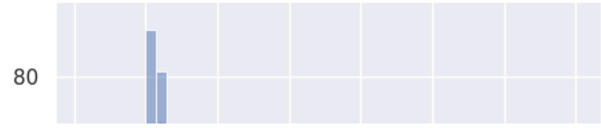
```
loan['CoapplicantIncome_log']=np.log(loan['CoapplicantIncome'])
sns.displot(loan.CoapplicantIncome_log,kde=True)
sns.set(rc={'figure.figsize':(2,2)})
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/arraylike.py:402: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



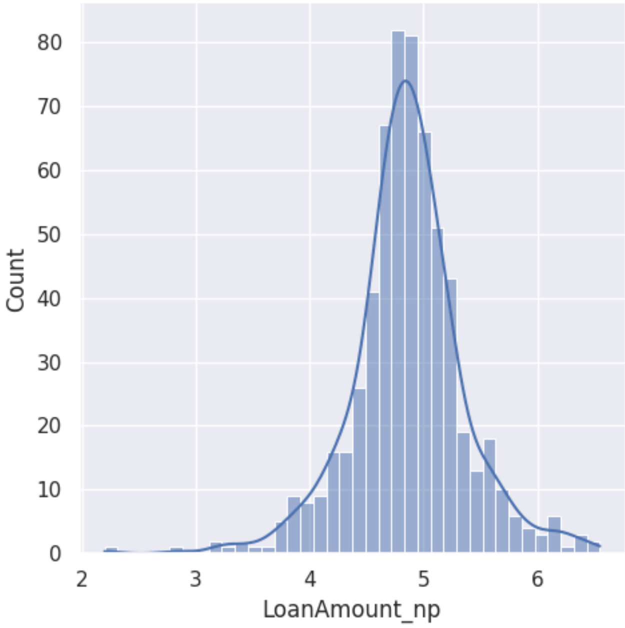
Now we can observe a Normal distribution in a form of a Bell Curve.

```
#To display the column "Loan Amount".
sns.displot(loan.LoanAmount,kde=True)
sns.set(rc={'figure.figsize':(2,2)})
```



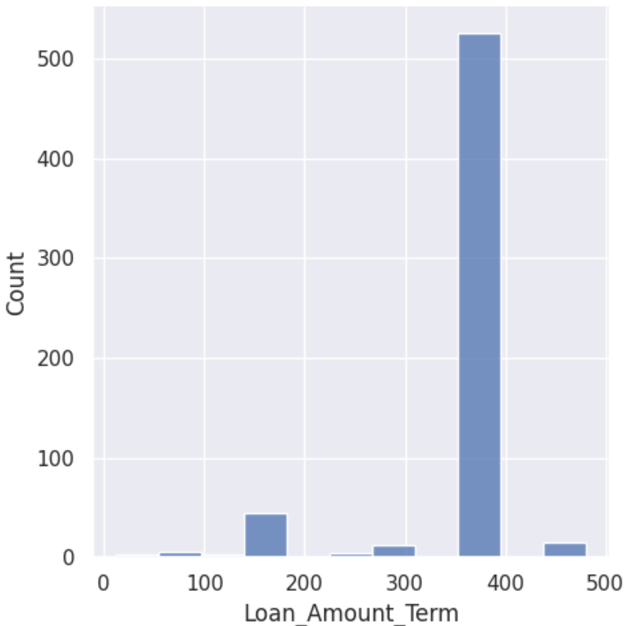
The loan amount graph is slightly right-skewed. We will consider this for Normalization.

```
loan['LoanAmount_np']=np.log(loan['LoanAmount'])
sns.displot(loan.LoanAmount_np,kde=True)
sns.set(rc={'figure.figsize':(0,0)})
```

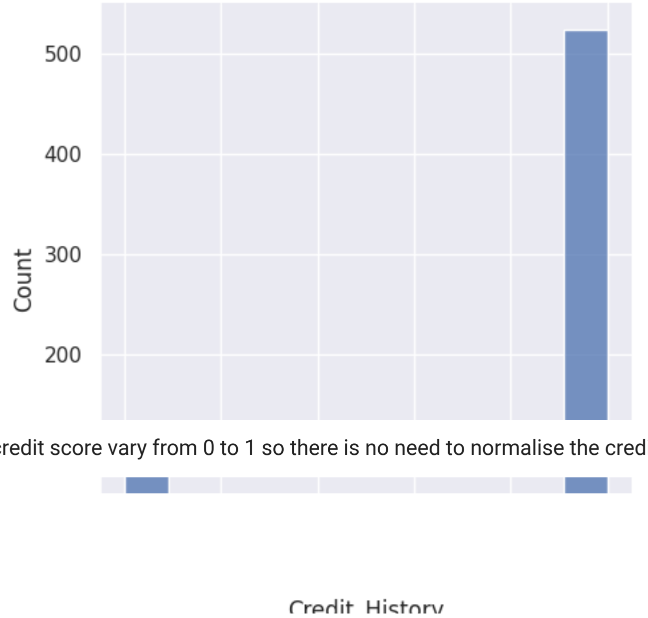


Now the data is look likely to be Bell curve, now the chart give better understanding.

```
#To display the column "Loan Amount Term".
sns.displot(loan.Loan_Amount_Term)
sns.set(rc={'figure.figsize':(2,2)})
```



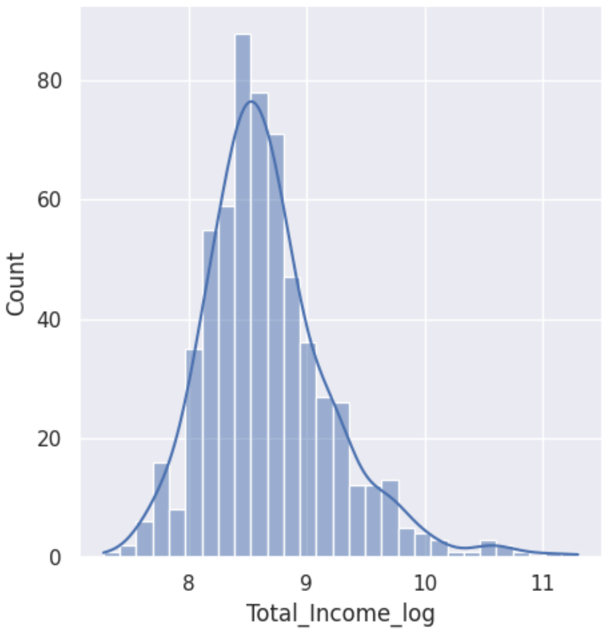

```
# To display the column "Credit History".
sns.displot(loan.Credit_History)
sns.set(rc={'figure.figsize':(2,2)})
```



The credit score vary from 0 to 1 so there is no need to normalise the credit history graph

Creation of new attributes

```
# total income
loan['Total_Income'] = loan['ApplicantIncome'] + loan['CoapplicantIncome']
loan['Total_Income_log']=np.log(loan['Total_Income'])
sns.displot(loan.Total_Income_log,kde=True)
sns.set(rc={'figure.figsize':(2,2)})
```



We can observe the normal distribution of the newly created column 'Total Income'.

Correlation Matrix

```
#For this project, the correlation matrix will discover the correlation for numerical attributes.
corr=loan.corr()
plt.figure(figsize=(10,4))
sns.heatmap(corr,annot=True)
```

```
<ipython-input-33-5d271eb966e8>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will
  corr=loan.corr()
<Axes: >
```

For this project, the correlation matrix will discover the correlation for numerical attributes.