```
from google.colab import drive
drive.mount('/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947

Enter your authorization code:
··········
Mounted at /gdrive
```

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
%matplotlib inline
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarnir
  import pandas.util.testing as tm
```

```
df=pd.read_csv('/gdrive/My Drive/Colab Notebooks/train.csv')
```

```
df.head()
```

	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

```
df.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
```

Describing the training Data

```
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
```

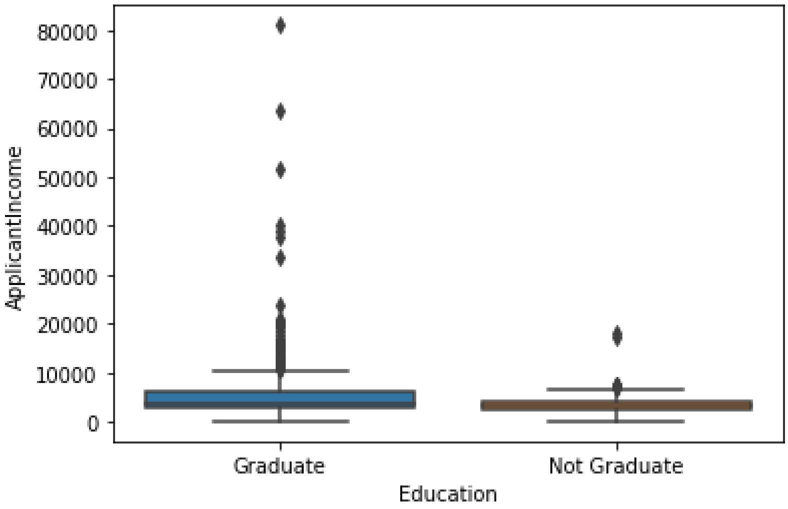
```
df.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Hist
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

BoxPlot for Applicant Income Vs Education

```
sns.boxplot(x=df.Education,y=df.ApplicantIncome)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fada880b080>
```



Total Income of Applicant and Co-Applicant

```
calculatedIncome=df.ApplicantIncome+df.CoapplicantIncome
```

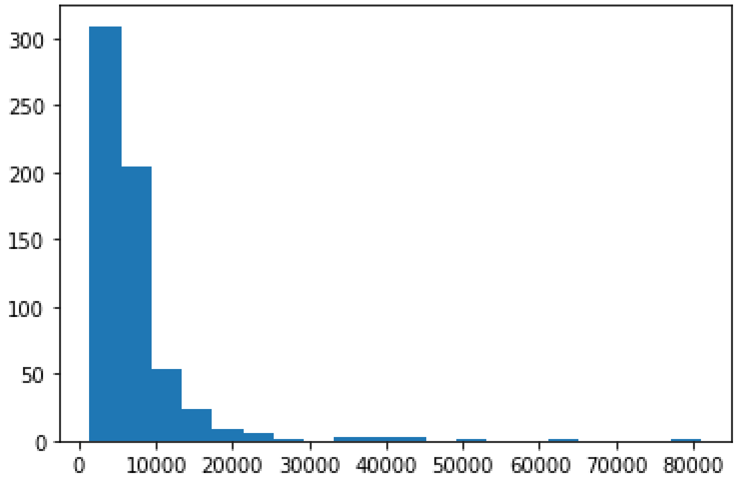
```
calculatedIncome
```

```
0      5849.0
1      6091.0
2      3000.0
3      4941.0
4      6000.0
        ...
609    2900.0
610    4106.0
611    8312.0
612    7583.0
613    4583.0
Length: 614, dtype: float64
```

Histogram for the Calculated Income

```
plt.hist(x=calculatedIncome,bins=20)
```

```
(array([309., 204.,  54.,  23.,   9.,   5.,   1.,   0.,   2.,   2.,   2.,
          0.,   1.,   0.,   0.,   1.,   0.,   0.,   0.,   1.]),
 array([ 1442. ,  5419.9,  9397.8, 13375.7, 17353.6, 21331.5, 25309.4,
        29287.3, 33265.2, 37243.1, 41221. , 45198.9, 49176.8, 53154.7,
        57132.6, 61110.5, 65088.4, 69066.3, 73044.2, 77022.1, 81000. ]),
 <a list of 20 Patch objects>)
```



Frequency Table for Credit History Vs Loan Status

```
freq=df.groupby(df.Credit_History).count()
```

```
print('Frequency Table for Credit History and Loan status')
print('Frequency Table for Credit History and Loan_status')
freq.Loan_Status
```

```
Frequency Table for Credit History and Loan_status
Credit_History
0.0     89
1.0    475
Name: Loan_Status, dtype: int64
```

Missing Values

```
df.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
```

```
missing=(df.isnull().sum()/len(df))*100
```

```
print(round(missing,2))
```

```
Loan_ID              0.00
Gender               2.12
Married              0.49
Dependents           2.44
Education            0.00
Self_Employed        5.21
ApplicantIncome      0.00
CoapplicantIncome    0.00
LoanAmount           3.58
Loan_Amount_Term     2.28
Credit_History       8.14
Property_Area        0.00
Loan_Status          0.00
dtype: float64
```

```
df.shape
```

```
(614, 13)
```

```
df.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
```

Splitting columns as categorical and continuous

```
continuous=df.select_dtypes(exclude=['object'])
```

```
continuous
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Histor
0	5849	0.0	NaN	360.0	1.
1	4583	1508.0	128.0	360.0	1.
2	3000	0.0	66.0	360.0	1.
3	2583	2358.0	120.0	360.0	1.
4	6000	0.0	141.0	360.0	1.
...	...	...	...	...	.
609	2900	0.0	71.0	360.0	1.
610	4106	0.0	40.0	180.0	1.
611	8072	240.0	253.0	360.0	1.
612	7583	0.0	187.0	360.0	1.
613	4583	0.0	133.0	360.0	0.

614 rows × 5 columns

```
categorical=df.select_dtypes(include=['object'])
```

```
categorical
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Property_Area
0	LP001002	Male	No	0	Graduate	No	Urban
1	LP001003	Male	Yes	1	Graduate	No	Rural
2	LP001005	Male	Yes	0	Graduate	Yes	Urban
3	LP001006	Male	Yes	0	Not Graduate	No	Urban
4	LP001008	Male	No	0	Graduate	No	Urban
...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	Rural
610	LP002979	Male	Yes	3+	Graduate	No	Rural
611	LP002983	Male	Yes	1	Graduate	No	Urban
612	LP002984	Male	Yes	2	Graduate	No	Urban
613	LP002990	Female	No	0	Graduate	Yes	Semiurban

614 rows × 8 columns

Filling the missing values

```
for x in continuous:
  mean=df[x].mean()
  df[x]=df[x].fillna(mean)


for x in categorical:
  mode=df[x].mode
  df[x]=df[x].fillna(mode)


missingAfterPreprocessing=(df.isnull().sum()/len(df))*100


print(round(missingAfterPreprocessing,2))
```

```
Loan_ID              0.0
Gender               0.0
Married              0.0
Dependents           0.0
Education            0.0
LoanAmount           0.0
Loan_Amount_Term     0.0
Property_Area        0.0
Loan_Status          0.0
dtype: float64
```

Not Required to apply Standard Scalar for this Data