In [23]:

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

In [2]:

```
data = pd.read_csv('train.csv')
data.shape
```

Out[2]:

```
(614, 13)
```

In [3]:

```
data.columns
```

Out[3]:

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanA
mount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_St
atus'],
      dtype='object')
```

In [4]:

```
data.head()
```

Out[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coappl
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 [5]:

```
data.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
```

In [6]:

```
data = data.drop(columns=['Loan_ID'])
```

In [7]:

```
data.head()
```

Out[7]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	Male	No	0	Graduate	No	5849	0.(
1	Male	Yes	1	Graduate	No	4583	1508.(
2	Male	Yes	0	Graduate	Yes	3000	0.(
3	Male	Yes	0	Not Graduate	No	2583	2358.(
4	Male	No	0	Graduate	No	6000	0.(

Basic data exploration

In [8]:

```
data.describe()
```

Out[8]:

	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

In [9]:

```
data.describe(include=['object'])
```

Out[9]:

	Gender	Married	Dependents	Education	Self_Employed	Property_Area	Loan_Status
count	601	611	599	614	582	614	614
unique	2	2	4	2	2	3	2
top	Male	Yes	0	Graduate	No	Semiurban	Y
freq	489	398	345	480	500	233	422

In [ ]:

```

```

In [11]:

```
data.isna().sum()
```

Out[11]:

```
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 [17]:

```
cat_cols = data.dtypes == 'object'
cat_cols = list(cat_cols[cat_cols].index)

num_cols = data.dtypes != 'object'
num_cols = list(num_cols[num_cols].index)
```

In [18]:

```
cat_cols
```

Out[18]:

```
['Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'Property_Area',
 'Loan_Status']
```

In [19]:

```
num_cols
```

Out[19]:

```
['ApplicantIncome',
 'CoapplicantIncome',
 'LoanAmount',
 'Loan_Amount_Term',
 'Credit_History']
```

In [21]:

```
data[cat_cols].head()
```

Out[21]:

	Gender	Married	Dependents	Education	Self_Employed	Property_Area	Loan_Status
0	Male	No	0	Graduate	No	Urban	Y
1	Male	Yes	1	Graduate	No	Rural	N
2	Male	Yes	0	Graduate	Yes	Urban	Y
3	Male	Yes	0	Not Graduate	No	Urban	Y
4	Male	No	0	Graduate	No	Urban	Y

In [22]:

```
data[num_cols].head()
```

Out[22]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	5849	0.0	NaN	360.0	1.0
1	4583	1508.0	128.0	360.0	1.0
2	3000	0.0	66.0	360.0	1.0
3	2583	2358.0	120.0	360.0	1.0
4	6000	0.0	141.0	360.0	1.0

In [ ]:

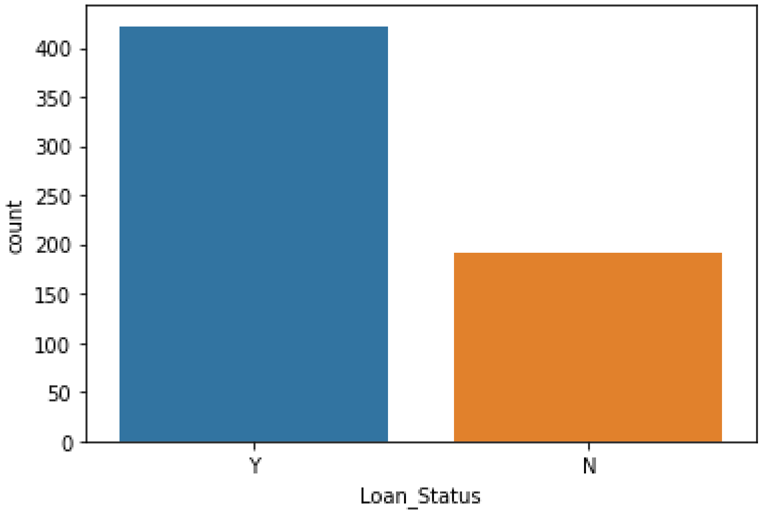
```

```

In [24]:

```
sns.countplot(data= data, x='Loan_Status')
```

Out[24]:

```
<AxesSubplot:xlabel='Loan_Status', ylabel='count'>
```



In [25]:

```
data['Loan_Status'].value_counts()
```

Out[25]:

```
Y    422
N    192
Name: Loan_Status, dtype: int64
```

In [ ]:

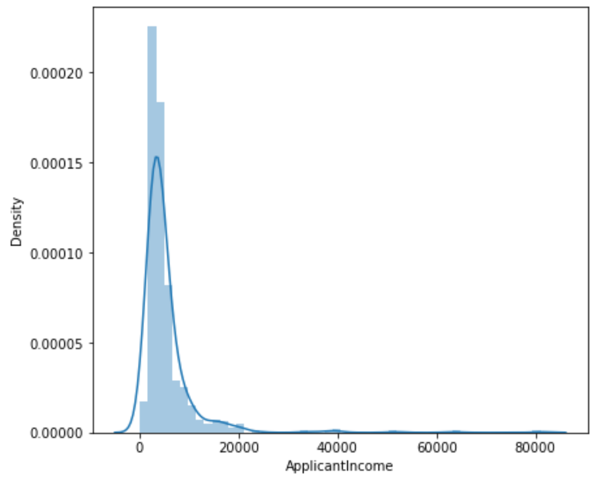
```

```

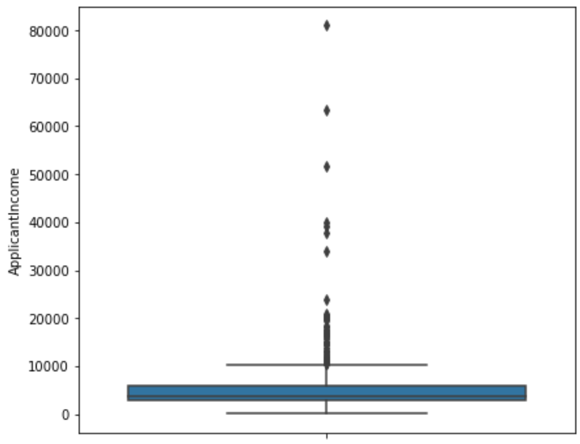
Applicant's income

In [31]:

```
plt.figure(figsize=(15, 6))

plt.subplot(121)
sns.distplot(data['ApplicantIncome'])

plt.subplot(122)
sns.boxplot(y= data['ApplicantIncome'])

plt.show()
```

```
/Users/mohit/opt/anaconda3/lib/python3.8/site-packages/seaborn/distrib
utions.py:2557: FutureWarning: `distplot` is a deprecated function and
will be removed in a future version. Please adapt your code to use eit
her `displot` (a figure-level function with similar flexibility) or `h
istplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



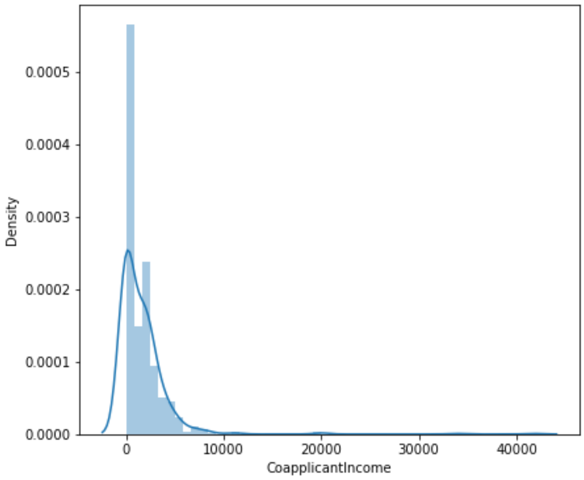


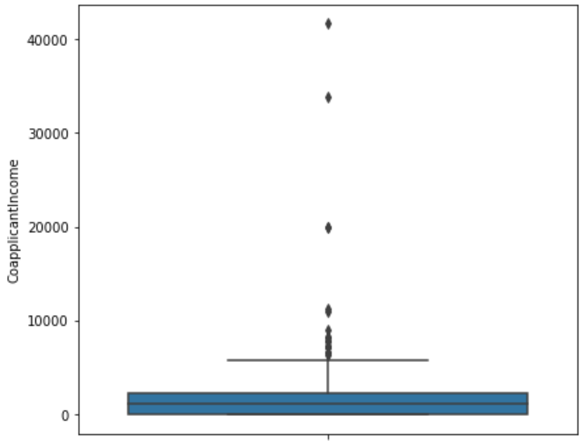
In [ ]:

In [32]:

```
plt.figure(figsize=(15, 6))

plt.subplot(121)
sns.distplot(data['CoapplicantIncome'])

plt.subplot(122)
sns.boxplot(y= data['CoapplicantIncome'])

plt.show()
```

```
/Users/mohit/opt/anaconda3/lib/python3.8/site-packages/seaborn/distrib
utions.py:2557: FutureWarning: `distplot` is a deprecated function and
will be removed in a future version. Please adapt your code to use eit
her `displot` (a figure-level function with similar flexibility) or `h
istplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```





In [33]:

```
np.quantile(data['CoapplicantIncome'], 0.25)
```

Out[33]:

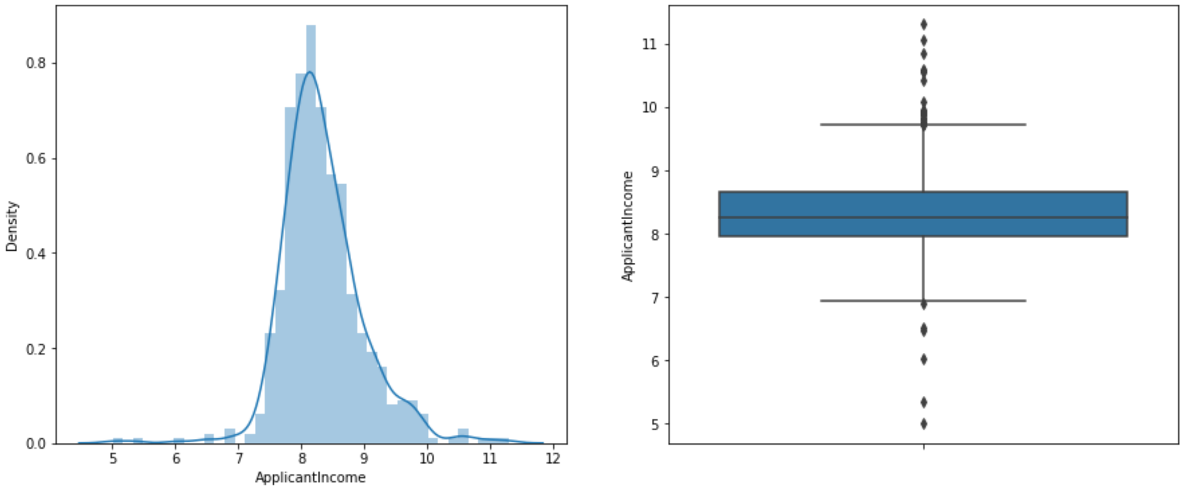
```
0.0
```

In [ ]:

In [34]:

```
plt.figure(figsize=(15, 6))

plt.subplot(121)
sns.distplot(np.log(data['ApplicantIncome']))

plt.subplot(122)
sns.boxplot(y= np.log(data['ApplicantIncome']))

plt.show()
```

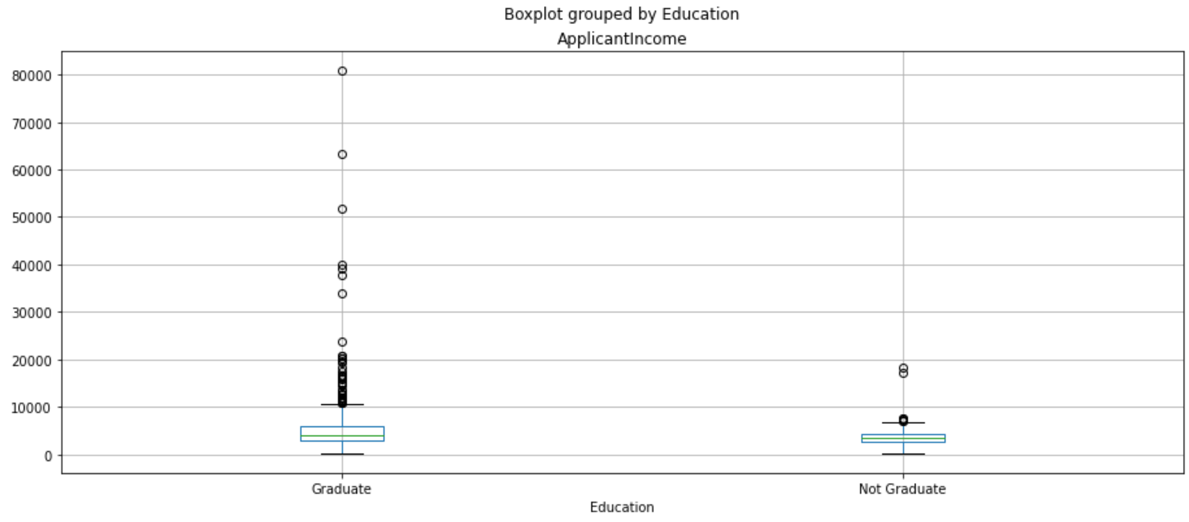
```
/Users/mohit/opt/anaconda3/lib/python3.8/site-packages/seaborn/distrib
utions.py:2557: FutureWarning: `distplot` is a deprecated function and
will be removed in a future version. Please adapt your code to use eit
her `displot` (a figure-level function with similar flexibility) or `h
istplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [ ]:

In [38]:

```
data.boxplot(column = 'ApplicantIncome', by = 'Education', figsize=(15, 6))
plt.show()
```



In [ ]:

```

```

In [40]:

```
data.groupby(by='Loan_Status').mean()['ApplicantIncome']
```

Out[40]:

```
Loan_Status
N    5446.078125
Y    5384.068720
Name: ApplicantIncome, dtype: float64
```

In [ ]:

```

```

Simple Feature Engineering

In [41]:

```
bins = [0, 2500, 4000, 6000, 81000]
group = ['Low', 'Avg', 'High', 'Very High']
```

In [45]:

```
data['Income_bin'] = pd.cut(data['ApplicantIncome'], bins, labels=group)
```

In [46]:

```
data.head()
```

Out[46]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	Male	No	0	Graduate	No	5849	0.(
1	Male	Yes	1	Graduate	No	4583	1508.(
2	Male	Yes	0	Graduate	Yes	3000	0.(
3	Male	Yes	0	Not Graduate	No	2583	2358.(
4	Male	No	0	Graduate	No	6000	0.(

In [ ]:

In [47]:

```
pd.crosstab(data['Income_bin'],  data['Loan_Status'])
```

Out[47]:

Loan_Status	**N**	**Y**
Income_bin		
Low	34	74
Avg	67	159
High	45	98
Very High	46	91

In [ ]:

In [ ]:

```
bins = [0, 2500, 4000, 6000, 81000]
group = ['Low', 'Avg', 'High', 'Very High']
```

In [58]:

```
data['CoApplicantIncome_bin'] = pd.cut(data['CoapplicantIncome'], bins, labels=grour
```

In [ ]:

```
```

In [59]:

```
data.head()
```

Out[59]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	Male	No	0	Graduate	No	5849	0.(
1	Male	Yes	1	Graduate	No	4583	1508.(
2	Male	Yes	0	Graduate	Yes	3000	0.(
3	Male	Yes	0	Not Graduate	No	2583	2358.(
4	Male	No	0	Graduate	No	6000	0.(

In [61]:

```
CoapplicantIncome = pd.crosstab(data['CoApplicantIncome_bin'],  data['Loan_Status'])
CoapplicantIncome
```

Out[61]:

Loan_Status	N	Y
CoApplicantIncome_bin		
Low	53	161
Avg	24	48
High	11	26
Very High	8	10

In [62]:

```
CoapplicantIncome.div(CoapplicantIncome.sum(axis=1), axis=0)
```

Out[62]:

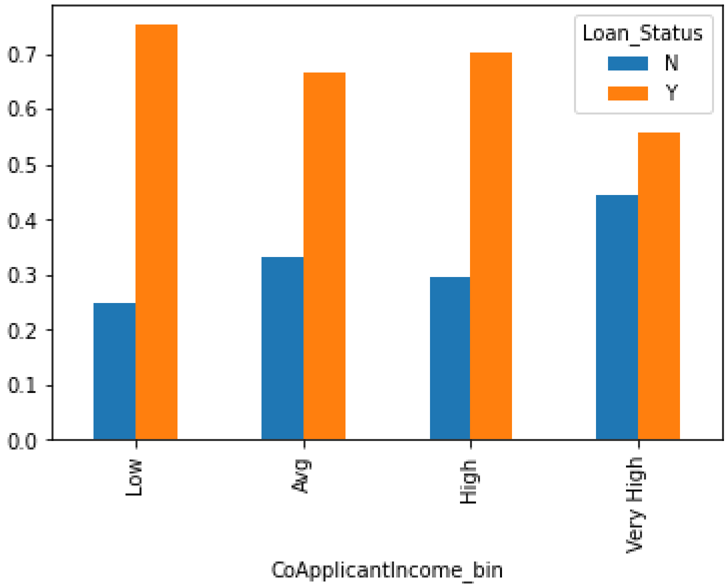
Loan_Status	N	Y
CoApplicantIncome_bin		
Low	0.247664	0.752336
Avg	0.333333	0.666667
High	0.297297	0.702703
Very High	0.444444	0.555556

In [63]:

```
CoapplicantIncome = pd.crosstab(data['CoApplicantIncome_bin'],  data['Loan_Status'])
CoapplicantIncome.div(CoapplicantIncome.sum(axis=1), axis=0).plot(kind='bar')
```

Out[63]:

<AxesSubplot:xlabel='CoApplicantIncome_bin'>



In [64]:

```
data['CoapplicantIncome'].value_counts().head()
```

Out[64]:

```
0.0       273
2500.0      5
2083.0      5
1666.0      5
1625.0      3
Name: CoapplicantIncome, dtype: int64
```

In [ ]:

```

```

In [65]:

```
data['TotalIncome'] = data['ApplicantIncome'] + data['CoapplicantIncome']
```

In [66]:

```
data.head()
```

Out[66]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	Male	No	0	Graduate	No	5849	0.(
1	Male	Yes	1	Graduate	No	4583	1508.(
2	Male	Yes	0	Graduate	Yes	3000	0.(
3	Male	Yes	0	Not Graduate	No	2583	2358.(
4	Male	No	0	Graduate	No	6000	0.(

In [ ]:

```

```

In [67]:

```
bins = [0, 2500, 4000, 6000, 81000]
group = ['Low', 'Avg', 'High', 'Very High']
```

In [68]:

```
data['TotalIncome_bin'] = pd.cut(data['TotalIncome'], bins, labels=group)
```

In [69]:

```
data.head()
```

Out[69]:

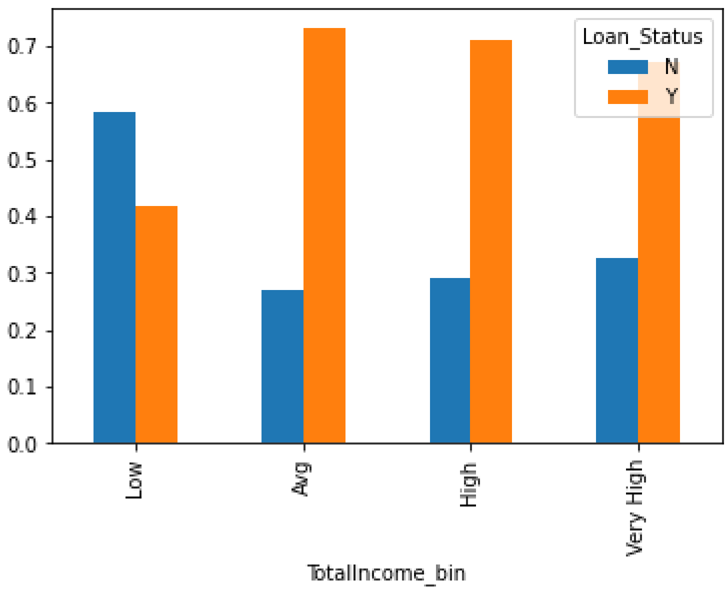
	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	Male	No	0	Graduate	No	5849	0.(
1	Male	Yes	1	Graduate	No	4583	1508.(
2	Male	Yes	0	Graduate	Yes	3000	0.(
3	Male	Yes	0	Not Graduate	No	2583	2358.(
4	Male	No	0	Graduate	No	6000	0.(

In [70]:

```
TotalIncome = pd.crosstab(data['TotalIncome_bin'],  data['Loan_Status'])
TotalIncome.div(TotalIncome.sum(axis=1), axis=0).plot(kind='bar')
```

Out[70]:

```
<AxesSubplot:xlabel='TotalIncome_bin'>
```



In [ ]:

```

```

Loan term & Loan amaount

In [75]:

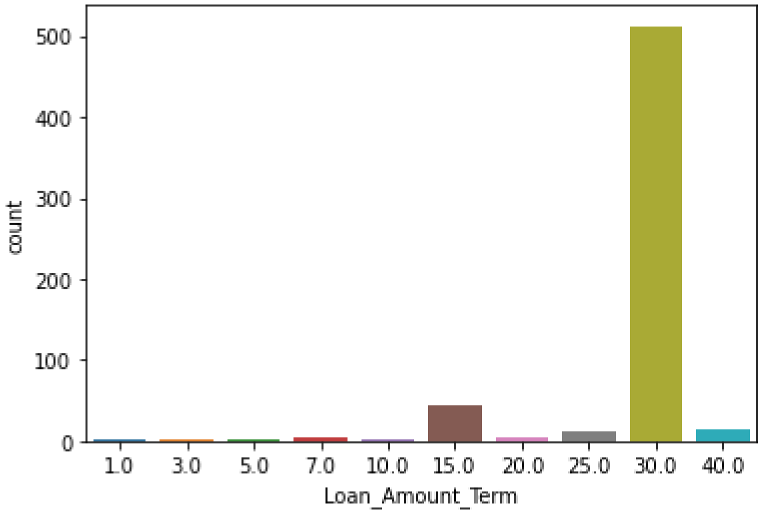
```
data['Loan_Amount_Term']= (data['Loan_Amount_Term']/12).astype('float')
```

In [76]:

```
sns.countplot(x='Loan_Amount_Term',  data=data)
```

Out[76]:

```
<AxesSubplot:xlabel='Loan_Amount_Term', ylabel='count'>
```



In [ ]:

```

```

In [77]:

```
data.head()
```

Out[77]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	Male	No	0	Graduate	No	5849	0.(
1	Male	Yes	1	Graduate	No	4583	1508.(
2	Male	Yes	0	Graduate	Yes	3000	0.(
3	Male	Yes	0	Not Graduate	No	2583	2358.(
4	Male	No	0	Graduate	No	6000	0.(

In [78]:

```
data['Loan_Amount_per_year'] = data['LoanAmount']/ data['Loan_Amount_Term']
```

In [ ]:

```

```

In [79]:

```
data.head()
```

Out[79]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	Male	No	0	Graduate	No	5849	0.(
1	Male	Yes	1	Graduate	No	4583	1508.(
2	Male	Yes	0	Graduate	Yes	3000	0.(
3	Male	Yes	0	Not Graduate	No	2583	2358.(
4	Male	No	0	Graduate	No	6000	0.(

In [84]:

```
data['EMI'] = np.round(data['Loan_Amount_per_year']*1000/12, 2)
```

In [85]:

```
data.head()
```

Out[85]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	Male	No	0	Graduate	No	5849	0.(
1	Male	Yes	1	Graduate	No	4583	1508.(
2	Male	Yes	0	Graduate	Yes	3000	0.(
3	Male	Yes	0	Not Graduate	No	2583	2358.(
4	Male	No	0	Graduate	No	6000	0.(

In [86]:

```
data['Able_to_pay_EMI'] = (data['EMI'] < data['TotalIncome']*0.1)
```

In [88]:

```
data.head()
```

Out[88]:

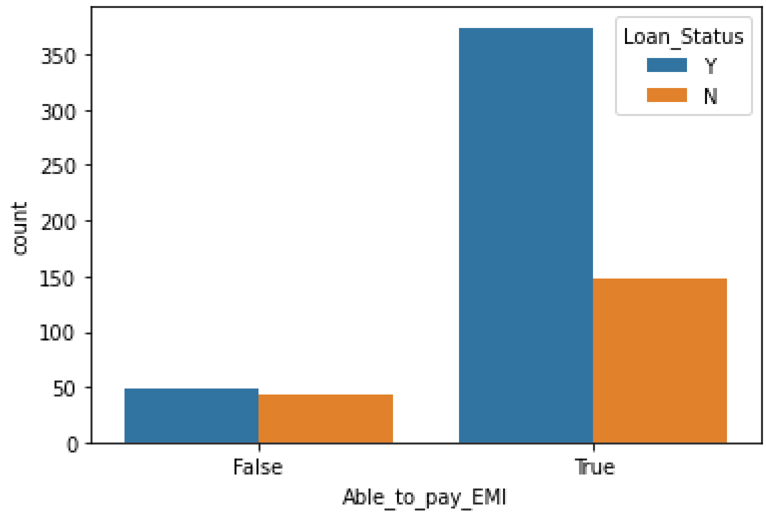
	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	Male	No	0	Graduate	No	5849	0.(
1	Male	Yes	1	Graduate	No	4583	1508.(
2	Male	Yes	0	Graduate	Yes	3000	0.(
3	Male	Yes	0	Not Graduate	No	2583	2358.(
4	Male	No	0	Graduate	No	6000	0.(

In [90]:

```
sns.countplot(x='Able_to_pay_EMI', data=data, hue='Loan_Status')
```

Out[90]:

<AxesSubplot:xlabel='Able_to_pay_EMI', ylabel='count'>



In [ ]:

In [101]:

```
data['Dependents'].value_counts()
```

Out[101]:

```
0    345
1    102
2    101
3     51
Name: Dependents, dtype: int64
```

In [102]:

```
data['Dependents'].replace('3+', 3, inplace=True)
```

In [106]:

```
data['Dependents'] = data['Dependents'].astype('float')
```

In [ ]:

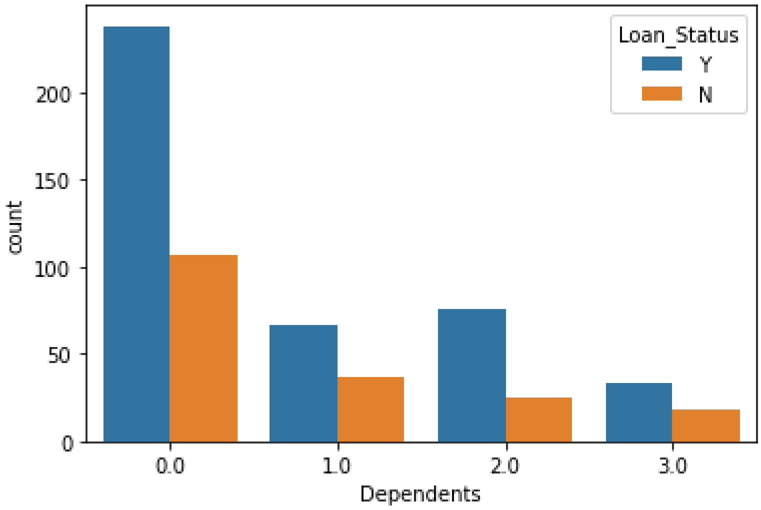
```
```

In [108]:

```
sns.countplot(x='Dependents', data=data, hue='Loan_Status')
```

Out[108]:

```
<AxesSubplot:xlabel='Dependents', ylabel='count'>
```



In [ ]:

```
```

In [109]:

```
data['Credit_History'].value_counts()
```

Out[109]:

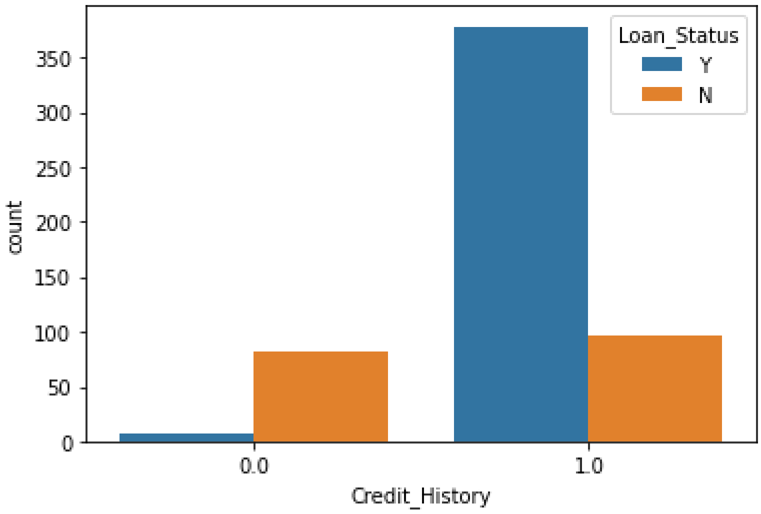
```
1.0    475
0.0     89
Name: Credit_History, dtype: int64
```

In [110]:

```
sns.countplot(x='Credit_History', data=data, hue='Loan_Status')
```

Out[110]:

```
<AxesSubplot:xlabel='Credit_History', ylabel='count'>
```



In [ ]:

```

```

Missing Values

In [111]:

```
data.isna().sum()
```

Out[111]:

```
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
Income_bin                   0
CoApplicantIncome_bin      273
TotalIncome                  0
TotalIncome_bin              0
Loan_Amount_per_year        36
EMI                         36
Able_to_pay_EMI              0
dtype: int64
```

In [ ]:

```

```

In [113]:

```
def missing_to_df(df):
    #Number and percentage of missing data in training data set for each column
    total_missing_df = df.isnull().sum().sort_values(ascending =False)
    percent_missing_df = (df.isnull().sum()/df.isnull().count()*100).sort_values(asc
    missing_data_df = pd.concat([total_missing_df, percent_missing_df], axis=1, keys
    return missing_data_df
```

In [115]:

```
missing_df = missing_to_df(data)
missing_df[missing_df['Total']> 0]
```

Out[115]:

	Total	Percent
CoApplicantIncome_bin	273	44.462541
Credit_History	50	8.143322
EMI	36	5.863192
Loan_Amount_per_year	36	5.863192
Self_Employed	32	5.211726
LoanAmount	22	3.583062
Dependents	15	2.442997
Loan_Amount_Term	14	2.280130
Gender	13	2.117264
Married	3	0.488599

In [116]:

```
data['Credit_History'].fillna(2, inplace=True)
```

In [117]:

```
data['Self_Employed'].value_counts()
```

Out[117]:

```
No      500
Yes      82
Name: Self_Employed, dtype: int64
```

In [118]:

```
data['Self_Employed'].fillna('Others', inplace=True)
```

In [ ]:

```
```

In [124]:

```
from sklearn.impute import SimpleImputer

median_imputer = SimpleImputer(strategy='median')

data['EMI'] = median_imputer.fit_transform(pd.DataFrame(data['EMI']))
data['LoanAmount'] = median_imputer.fit_transform(pd.DataFrame(data['LoanAmount']))
data['Loan_Amount_per_year'] = median_imputer.fit_transform(pd.DataFrame(data['Loan_
data['Loan_Amount_Term'] = median_imputer.fit_transform(pd.DataFrame(data['Loan_Amou
```

In [125]:

```
missing_df = missing_to_df(data)
missing_df[missing_df['Total']> 0]
```

Out[125]:

	Total	Percent
CoApplicantIncome_bin	273	44.462541
Dependents	15	2.442997
Gender	13	2.117264
Married	3	0.488599

In [131]:

```
# data[pd.isna(data['Married'])]
```

In [132]:

```
freq_imputer = SimpleImputer(strategy='most_frequent')
```

In [135]:

```
data['Dependents'] = freq_imputer.fit_transform(pd.DataFrame(data['Dependents']))
data['Gender'] = freq_imputer.fit_transform(pd.DataFrame(data['Gender']))
data['Married'] = freq_imputer.fit_transform(pd.DataFrame(data['Married']))
```

In [136]:

```
missing_df = missing_to_df(data)
missing_df[missing_df['Total']> 0]
```

Out[136]:

	Total	Percent
CoApplicantIncome_bin	273	44.462541

In [137]:

```
data.drop('CoApplicantIncome_bin', axis=1, inplace=True)
```

In [138]:

```
missing_df = missing_to_df(data)
missing_df[missing_df['Total']> 0]
```

Out[138]:

Total	Percent

In [ ]:

```
```

Converting categorical to Numeric Encoding

1. LabelEncoding

In [139]:

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

In [143]:

```
enc_label = LabelEncoder()
# enc_label.fit_transform()
```

In [142]:

```
ohe = OneHotEncoder()
# ohe.fit_transform()
```

In [144]:

```
! pip install category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.5.0-py2.py3-none-any.whl (69 kB)
     |████████████████████████████████| 69 kB 6.4 MB/s eta 0:00:011
Requirement already satisfied: statsmodels>=0.9.0 in /Users/mohit/opt/
anaconda3/lib/python3.8/site-packages (from category_encoders) (0.12.
2)
Requirement already satisfied: pandas>=1.0.5 in /Users/mohit/opt/anaco
nda3/lib/python3.8/site-packages (from category_encoders) (1.2.4)
Requirement already satisfied: patsy>=0.5.1 in /Users/mohit/opt/anacon
da3/lib/python3.8/site-packages (from category_encoders) (0.5.1)
Requirement already satisfied: scipy>=1.0.0 in /Users/mohit/opt/anacon
da3/lib/python3.8/site-packages (from category_encoders) (1.6.2)
Requirement already satisfied: numpy>=1.14.0 in /Users/mohit/opt/anaco
nda3/lib/python3.8/site-packages (from category_encoders) (1.20.1)
Requirement already satisfied: scikit-learn>=0.20.0 in /Users/mohit/op
t/anaconda3/lib/python3.8/site-packages (from category_encoders) (0.2
4.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /Users/mohit/
opt/anaconda3/lib/python3.8/site-packages (from pandas>=1.0.5->categor
y_encoders) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in /Users/mohit/opt/anacon
da3/lib/python3.8/site-packages (from pandas>=1.0.5->category_encoder
s) (2021.1)
Requirement already satisfied: six in /Users/mohit/opt/anaconda3/lib/p
ython3.8/site-packages (from patsy>=0.5.1->category_encoders) (1.15.0)
Requirement already satisfied: joblib>=0.11 in /Users/mohit/opt/anacon
da3/lib/python3.8/site-packages (from scikit-learn>=0.20.0->category_e
ncoders) (1.0.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/mohit/op
t/anaconda3/lib/python3.8/site-packages (from scikit-learn>=0.20.0->ca
tegory_encoders) (2.1.0)
Installing collected packages: category-encoders
Successfully installed category-encoders-2.5.0
```

In [145]:

```
from category_encoders import TargetEncoder
```

In [146]:

```
te = TargetEncoder()
# te.fit_transform()
```

```
/Users/mohit/opt/anaconda3/lib/python3.8/site-packages/category_encode
rs/target_encoder.py:92: FutureWarning: Default parameter min_samples_
leaf will change in version 2.6.See https://github.com/scikit-learn-co
ntrib/category_encoders/issues/327 (https://github.com/scikit-learn-co
ntrib/category_encoders/issues/327)
  warnings.warn("Default parameter min_samples_leaf will change in ver
sion 2.6."
/Users/mohit/opt/anaconda3/lib/python3.8/site-packages/category_encode
rs/target_encoder.py:97: FutureWarning: Default parameter smoothing wi
ll change in version 2.6.See https://github.com/scikit-learn-contrib/c
ategory_encoders/issues/327 (https://github.com/scikit-learn-contrib/c
ategory_encoders/issues/327)
  warnings.warn("Default parameter smoothing will change in version 2.
6."
```

In [147]:

```
data
```

Out[147]:

tatus	Income_bin	TotalIncome	TotalIncome_bin	Loan_Amount_per_year	EMI	Able_to_pay_EMI
Y	High	5849.0	High	4.383333	365.28	False
N	High	6091.0	Very High	4.266667	355.56	True
Y	Avg	3000.0	Avg	2.200000	183.33	True
Y	Avg	4941.0	High	4.000000	333.33	True
Y	High	6000.0	High	4.700000	391.67	True
...	...	...	...	...	...	...
Y	Avg	2900.0	Avg	2.366667	197.22	True
Y	High	4106.0	High	2.666667	222.22	True
Y	Very High	8312.0	Very High	8.433333	702.78	True
Y	Very High	7583.0	Very High	6.233333	519.44	True
N	High	4583.0	High	4.433333	369.44	True

In [152]:

```
data.drop(columns=['Income_bin', 'TotalIncome_bin'], axis=1, inplace=True)
```

In [153]:

```
data.dtypes
```

Out[153]:

```
Gender                   object
Married                  object
Dependents              float64
Education                object
Self_Employed            object
ApplicantIncome           int64
CoapplicantIncome       float64
LoanAmount              float64
Loan_Amount_Term        float64
Credit_History          float64
Property_Area            object
Loan_Status              object
TotalIncome             float64
Loan_Amount_per_year    float64
EMI                     float64
Able_to_pay_EMI            bool
dtype: object
```

In [155]:

```
s = data.dtypes == 'object'
object_cols = list(s[s].index)
object_cols
```

Out[155]:

```
['Gender',
 'Married',
 'Education',
 'Self_Employed',
 'Property_Area',
 'Loan_Status']
```

In [156]:

```
col = "Loan_Status"
data[col].value_counts()
```

Out[156]:

```
Y    422
N    192
Name: Loan_Status, dtype: int64
```

In [158]:

```
enc_label = LabelEncoder()
data[col] = enc_label.fit_transform(data[col])
```

In [159]:

```
col = "Loan_Status"
data[col].value_counts()
```

Out[159]:

```
1    422
0    192
Name: Loan_Status, dtype: int64
```

In [160]:

```
col = "Married"
data[col].value_counts()
```

Out[160]:

```
Yes    401
No     213
Name: Married, dtype: int64
```

In [161]:

```
enc_label = LabelEncoder()
data[col] = enc_label.fit_transform(data[col])
```

In [162]:

```
col = "Married"
data[col].value_counts()
```

Out[162]:

```
1    401
0    213
Name: Married, dtype: int64
```

In [163]:

```
enc_label = LabelEncoder()
data[col] = enc_label.fit_transform(data[col])
```

Out[163]:

```
Graduate        480
Not Graduate    134
Name: Education, dtype: int64
```

In [164]:

```
enc_label = LabelEncoder()
data[col] = enc_label.fit_transform(data[col])
```

In [ ]:

```

```

In [184]:

```
col = "Property_Area"
data[col].value_counts()
```

Out[184]:

```
Semiurban    233
Urban        202
Rural        179
Name: Property_Area, dtype: int64
```

In [185]:

```
ohe = OneHotEncoder()
ohe.fit_transform(pd.DataFrame(data[col])).toarray()
```

Out[185]:

```
array([[0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.],
       ...,
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 1., 0.]])
```

In [186]:

```
prop = pd.get_dummies(data[col], )
prop
```

Out[186]:

	Rural	Semiurban	Urban
0	0	0	1
1	1	0	0
2	0	0	1
3	0	0	1
4	0	0	1
...	...	...	...
609	1	0	0
610	1	0	0
611	0	0	1
612	0	0	1
613	0	1	0

614 rows × 3 columns

In [189]:

```
data=pd.concat([data, prop], axis=1)
```

In [190]:

```
data.drop(columns=['Property_Area'], inplace=True)
```

In [ ]:

```

```

In [178]:

```
s = data.dtypes == 'object'
object_cols = list(s[s].index)
object_cols
```

Out[178]:

```
['Gender', 'Self_Employed', 'Property_Area']
```

In [ ]:

```

```

In [182]:

```
# label encoding
data['Gender'] = data['Gender'].astype('category').cat.codes
data['Self_Employed'] = data['Self_Employed'].astype('category').cat.codes
```

In [191]:

```
s = data.dtypes == 'object'
object_cols = list(s[s].index)
object_cols
```

Out[191]:

```
[]
```

In [192]:

```
data.head()
```

Out[192]:

Credit_History	Loan_Status	TotalIncome	Loan_Amount_per_year	EMI	Able_to_pay_EMI	Rural
1.0	1	5849.0	4.383333	365.28	False	0
1.0	0	6091.0	4.266667	355.56	True	1
1.0	1	3000.0	2.200000	183.33	True	0
1.0	1	4941.0	4.000000	333.33	True	0
1.0	1	6000.0	4.700000	391.67	True	0

In [195]:

```
data['Able_to_pay_EMI'] = data['Able_to_pay_EMI'].astype('int')
```

In [ ]:

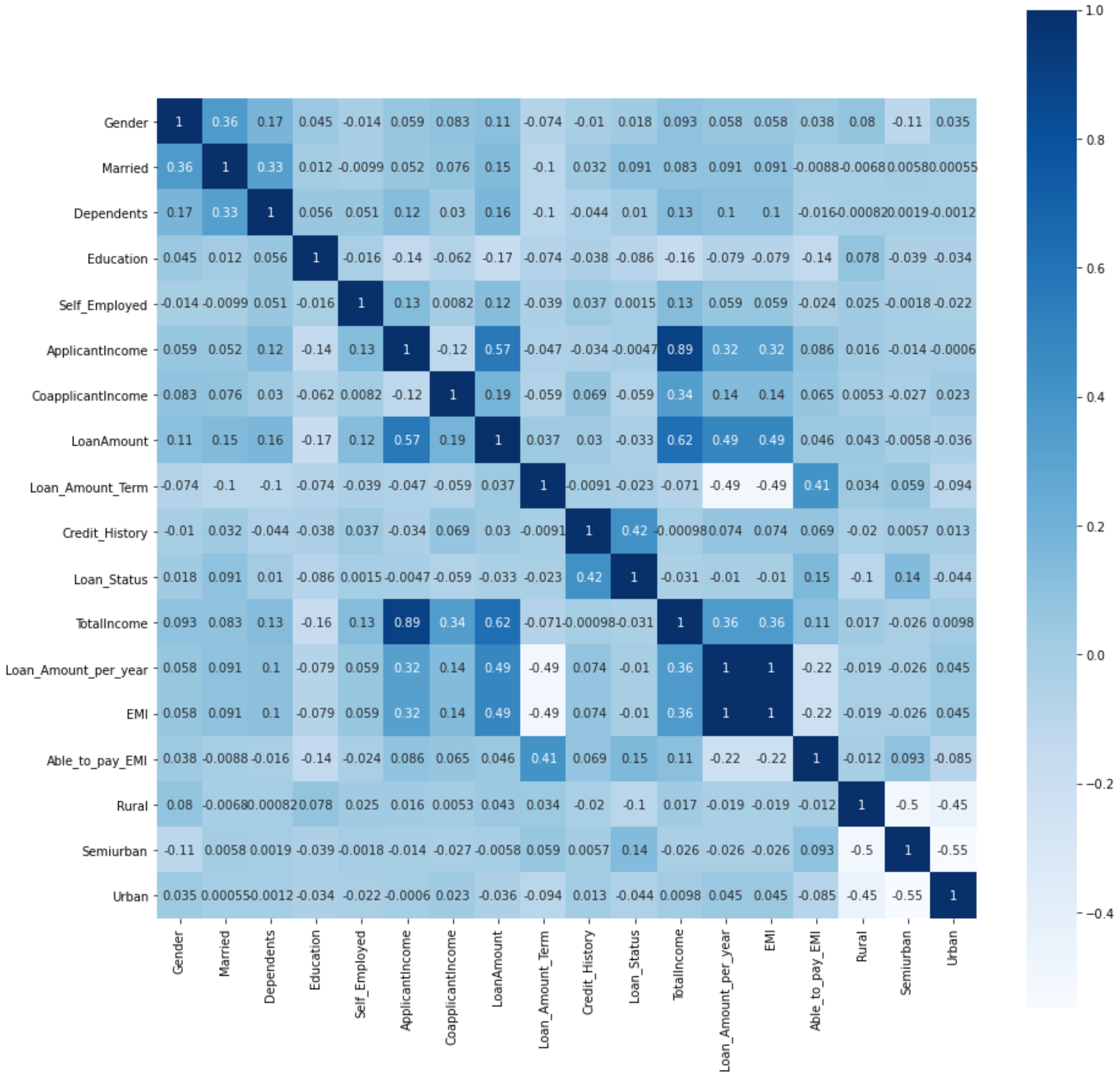
```

```

In [198]:

```
plt.figure(figsize=(15,15))
sns.heatmap(data.corr(), square=True, annot = True, cmap='Blues')
```

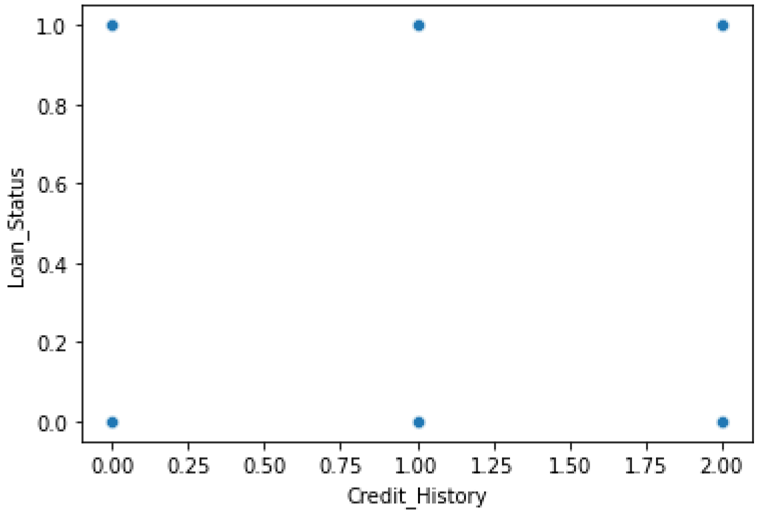
Out[198]:

```
<AxesSubplot:>
```



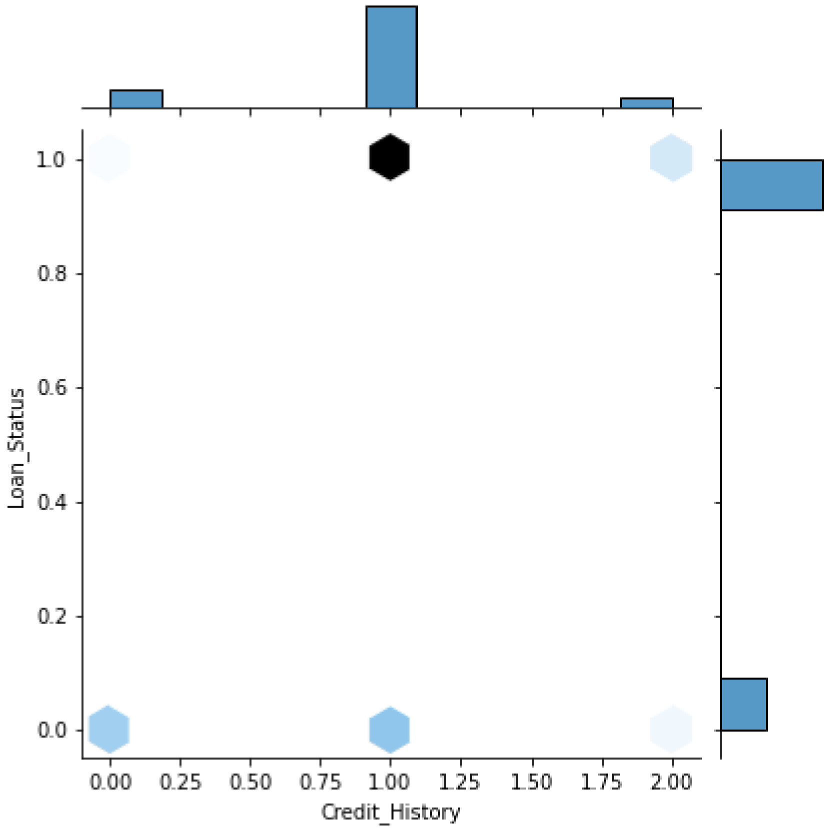
In [200]:

```
sns.scatterplot(data=data, x='Credit_History', y = 'Loan_Status')
plt.show()
```



In [201]:

```
sns.jointplot(data=data, x='Credit_History', y = 'Loan_Status', kind='hex')
plt.show()
```



In [ ]:

```

```

Feature Scaling

- Standaridation
- Normalisation

In [ ]:

```

```

In [207]:

```
mu = data.mean()
sig = data.std()
```

In [210]:

```
data = (data-mu)/sig
```

In [212]:

```
data.mean()
```

Out[212]:

```
Gender                  -9.474868e-17
Married                  1.855194e-16
Dependents              -9.402540e-18
Education               -1.117456e-16
Self_Employed            2.444660e-16
ApplicantIncome          5.243724e-18
CoapplicantIncome        7.883668e-17
LoanAmount              -1.387779e-17
Loan_Amount_Term         3.817070e-16
Credit_History          -2.169817e-17
Loan_Status              1.851577e-16
TotalIncome              1.073155e-16
Loan_Amount_per_year    -2.257062e-16
EMI                     -1.299178e-16
Able_to_pay_EMI         -1.795524e-16
Rural                   -4.585547e-16
Semiurban               -7.594359e-18
Urban                    1.600240e-16
dtype: float64
```

In [213]:

```
data.std()
```

Out[213]:

```
Gender                  1.0
Married                 1.0
Dependents              1.0
Education               1.0
Self_Employed           1.0
ApplicantIncome         1.0
CoapplicantIncome       1.0
LoanAmount              1.0
Loan_Amount_Term        1.0
Credit_History          1.0
Loan_Status             1.0
TotalIncome             1.0
Loan_Amount_per_year    1.0
EMI                     1.0
Able_to_pay_EMI         1.0
Rural                   1.0
Semiurban               1.0
Urban                   1.0
dtype: float64
```

In [ ]:

```

```