In [1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import matplotlib
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```python
df = pd.read_csv(r"C:\Users\Aditya Singh\Downloads\train_u6lujuX_CVtuZ9i.csv")
df.head()
```

Out[3]:

	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

In [4]:
```python
df.describe()
```

Out[4]:

	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 [5]:
```python
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
```

Taking care of Missing Values

In [6]:
```python
df.isnull().sum()
```

Out[6]:
```
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
```

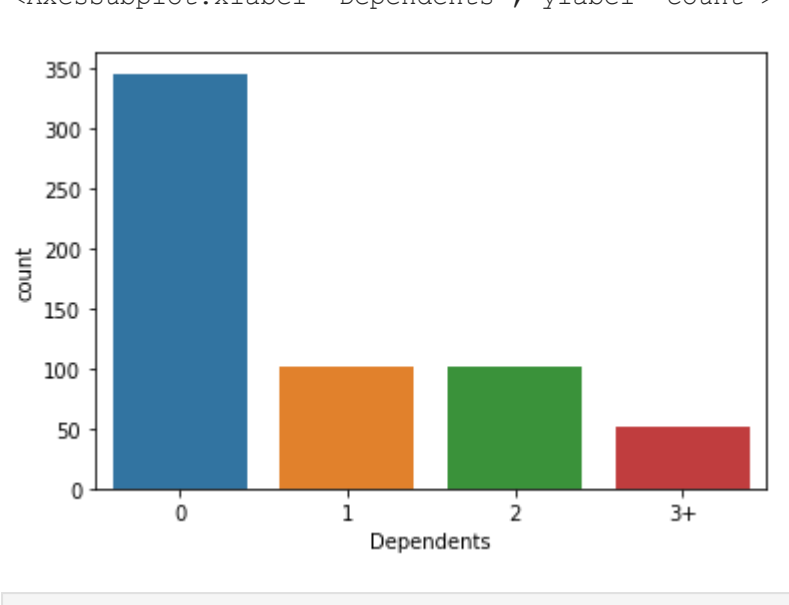
In [7]:
```python
#1 first filling up categorical misising values
```

In [8]:
```python
df['Gender'] = df["Gender"].fillna(df['Gender'].mode()[0])
df['Married'] = df["Married"].fillna(df['Married'].mode()[0])
df['Self_Employed'] = df["Self_Employed"].fillna(df['Self_Employed'].mode()[0])
```

In [9]:
```python
# counting the Dependents for better understanding about the data before filling it up.
```

In [10]:
```python
sns.countplot(x ='Dependents', data = df)
```

Out[10]: <AxesSubplot:xlabel='Dependents', ylabel='count'>



In [11]:
```python
#as we can see filling with mode make sense here.
```

In [12]:
```python
df['Dependents'] = df["Dependents"].fillna(df['Dependents'].mode()[0])
```

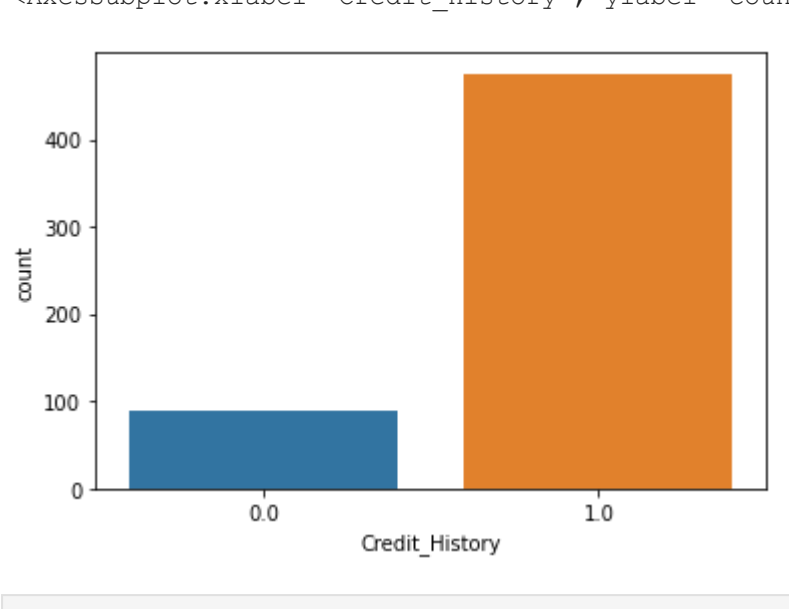
In [14]:
```python
#2 filling numerical values
#we should check for outliers before filling up numerical values.
```

In [15]:
```python
df['Credit_History'].unique()
```

Out[15]: array([1., 0., nan])

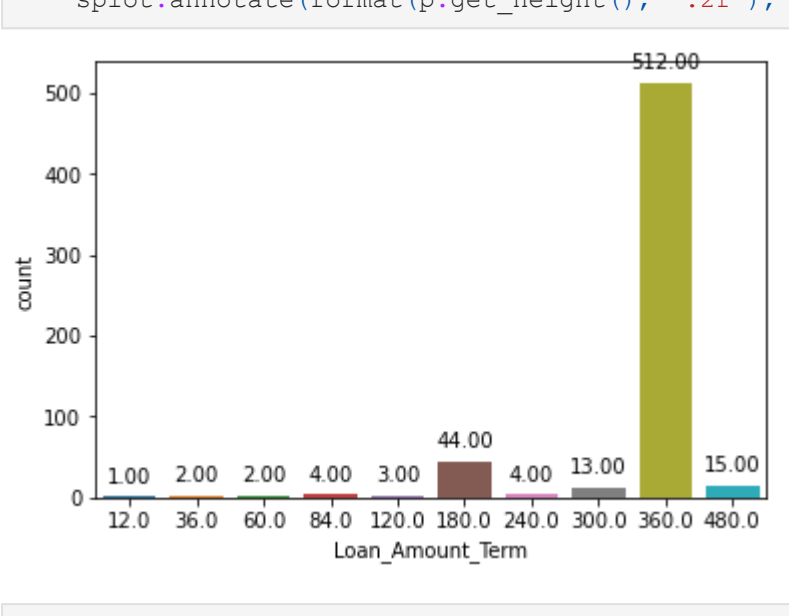
In [16]:
```python
sns.countplot(x ='Credit_History', data = df)
```

Out[16]: <AxesSubplot:xlabel='Credit_History', ylabel='count'>



In [17]:
```python
df['Credit_History']=df['Credit_History'].fillna(df['Credit_History'].mode()[0])
```

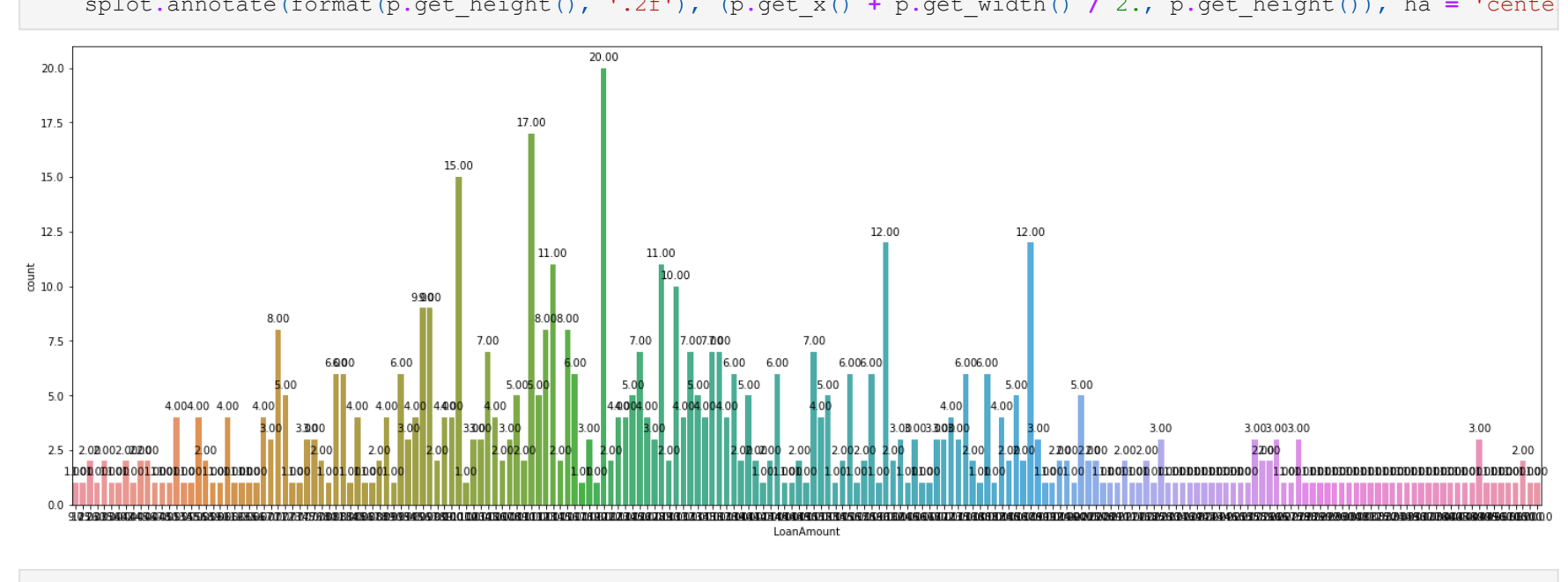
In [18]:
```python
splot = sns.countplot(x ='Loan_Amount_Term', data = df)
for p in splot.patches:
    splot.annotate(format(p.get_height(), '.2f'), (p.get_x() + p.get_width() / 2., p.get_height()), ha = 'cente
```



In [19]:
```python
# we can see that 360 has count of 512 so thats why replacing the loan_amount_term by mode will be smarter choi
```

In [20]:
```python
df['Loan_Amount_Term']=df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0])
```

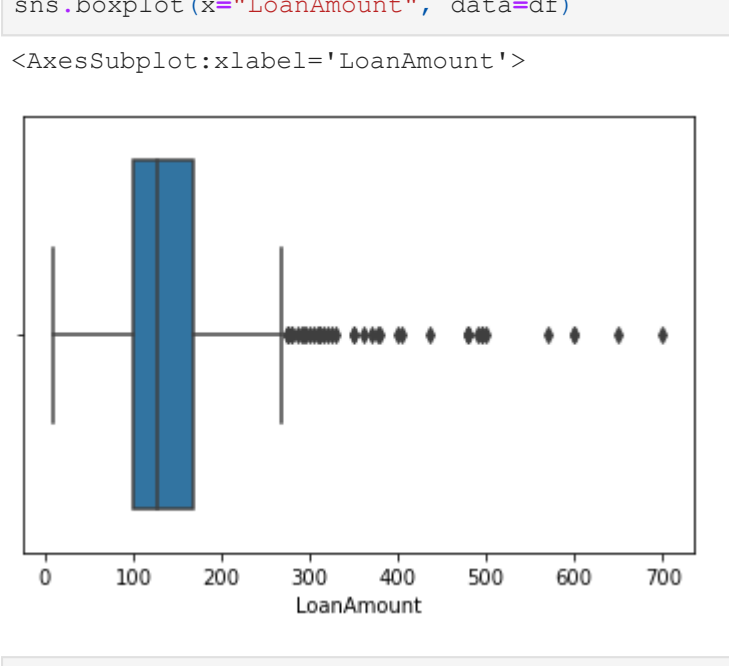
In [21]:
```python
plt.figure(figsize=(25, 8))
splot = sns.countplot(x ='LoanAmount', data = df)
for p in splot.patches:
    splot.annotate(format(p.get_height(), '.2f'), (p.get_x() + p.get_width() / 2., p.get_height()), ha = 'cente
```



In [22]:
```python
#we can not replace loan amount with mode because here mean or median will be better.
#before making choice betweeen mean and median we have to check for outliers.
# becuase mean is affected by outliers.
```

In [23]:
```python
sns.boxplot(x="LoanAmount", data=df)
```

Out[23]: <AxesSubplot:xlabel='LoanAmount'>



In [24]:
```python
#there is outlier so
```

In [25]:
```python
Q1 = df['LoanAmount'].quantile(0.25)
Q3 = df['LoanAmount'].quantile(0.75)
IQR = Q3 - Q1
```

In [26]:
```python
low_lim = Q1 - 1.5 * IQR
up_lim = Q3 + 1.5 * IQR
print('low_limit is', low_lim)
print('up_limit is', up_lim)
```

```
low_limit is -2.0
up_limit is 270.0
```

In [27]:
```python
outlier = []
for x in df['LoanAmount']:
    if ((x> up_lim) or (x<low_lim)):
        outlier.append(x)
print(' outlier in the dataset is', outlier)
```

```
 outlier in the dataset is [349.0, 315.0, 320.0, 286.0, 312.0, 370.0, 650.0, 290.0, 600.0, 275.0, 700.0, 495.0,
280.0, 279.0, 304.0, 330.0, 436.0, 480.0, 300.0, 376.0, 490.0, 308.0, 570.0, 380.0, 296.0, 275.0, 360.0, 405.0,
500.0, 480.0, 311.0, 480.0, 400.0, 324.0, 600.0, 275.0, 292.0, 350.0, 496.0]
```

In [28]:
```python
len(outlier)
```

Out[28]: 39

In [29]:
```python
#we will not remove the outliers becuase it has 39/592, which means it has 6.5% amount of data in whole.
```

In [30]:
```python
#we will use median to replace the missing value.
#becuase median is not affected by the outliers.
```

In [31]:
```python
df['LoanAmount']=df['LoanAmount'].fillna(df['LoanAmount'].median())
```

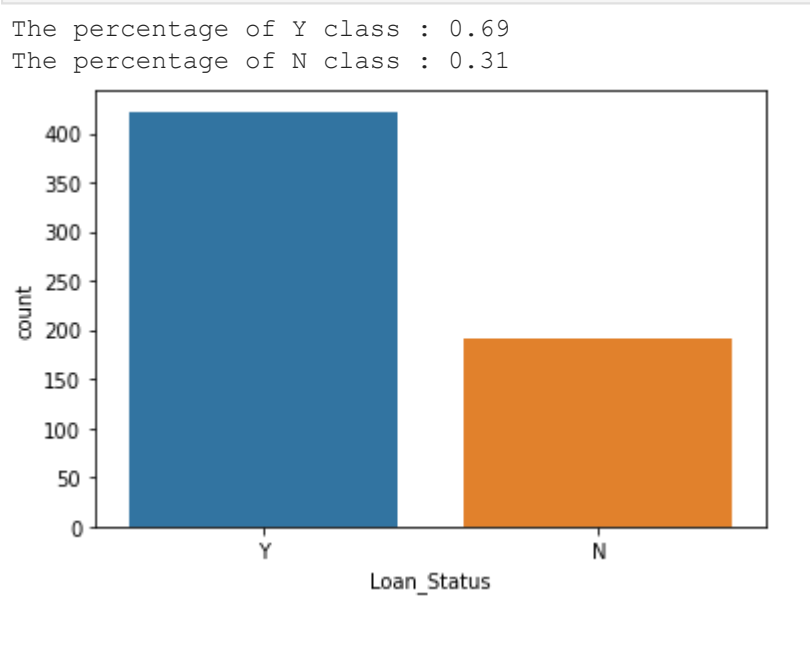
In [32]:
```python
df.isnull().sum()
# now there is no missing values.
```

Out[32]:
```
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
```

Checking for data imbalance

In [33]:
```python
sns.countplot(df['Loan_Status'])
print('The percentage of Y class : %.2f' % (df['Loan_Status'].value_counts()[0] / len(df)))
print('The percentage of N class : %.2f' % (df['Loan_Status'].value_counts()[1] / len(df)))

#there is almost balance we don't need to worry about that.
```

```
The percentage of Y class : 0.69
The percentage of N class : 0.31
```

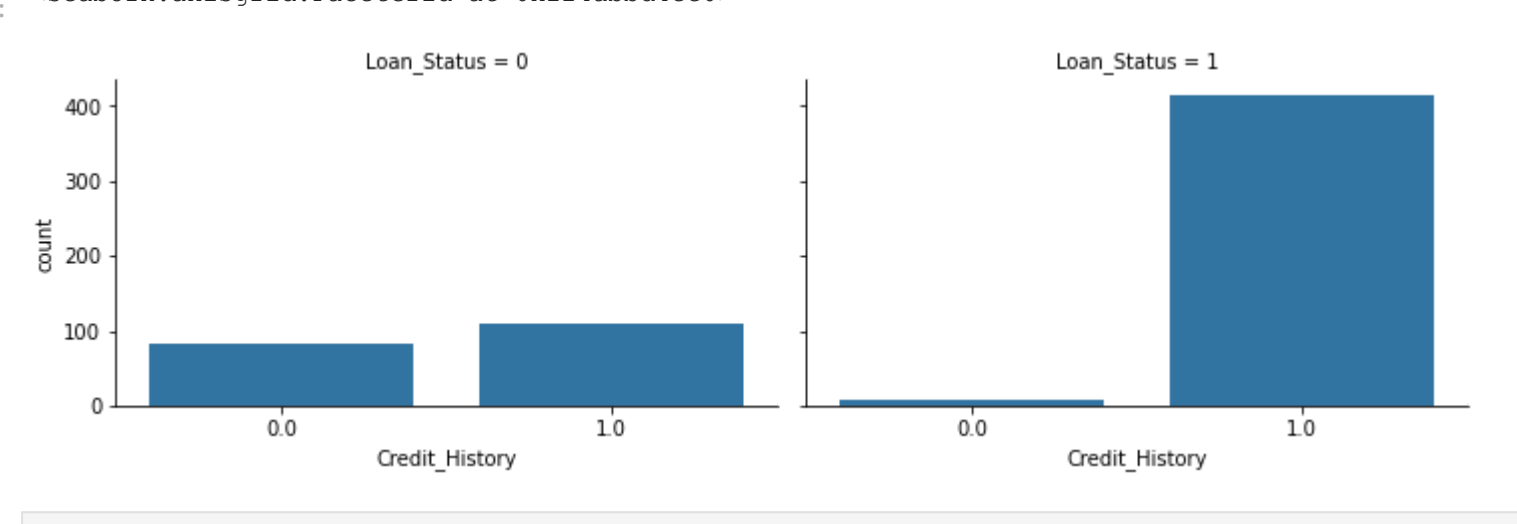


EDA

In [34]:
```python
df['Loan_Status'].replace('N',0,inplace=True)
df['Loan_Status'].replace('Y',1,inplace=True)
```

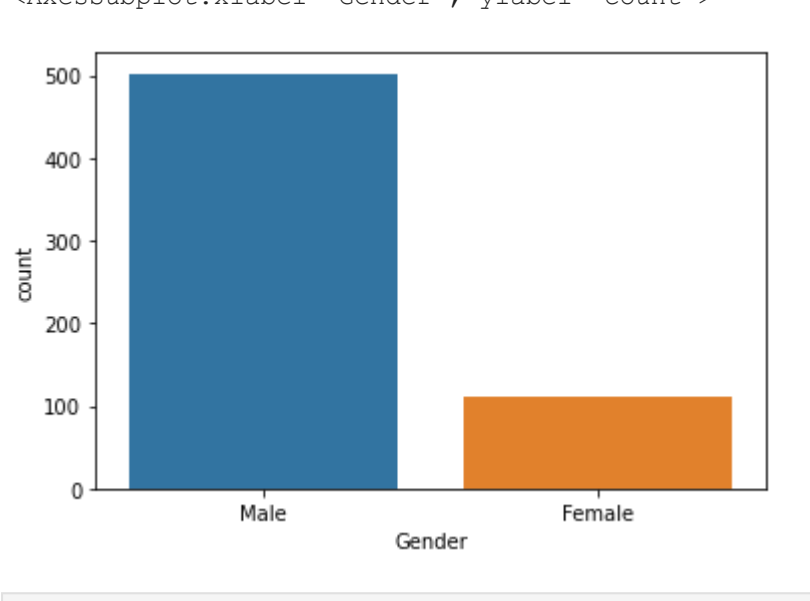
In [35]:
```python
#Credit history vs loan status
grid = sns.FacetGrid(df,col='Loan_Status', size=3.2, aspect=1.6)
grid.map(sns.countplot, 'Credit_History')

#people having credit history have easy time getting loan
```

Out[35]: <seaborn.axisgrid.FacetGrid at 0x224abbd4ee0>



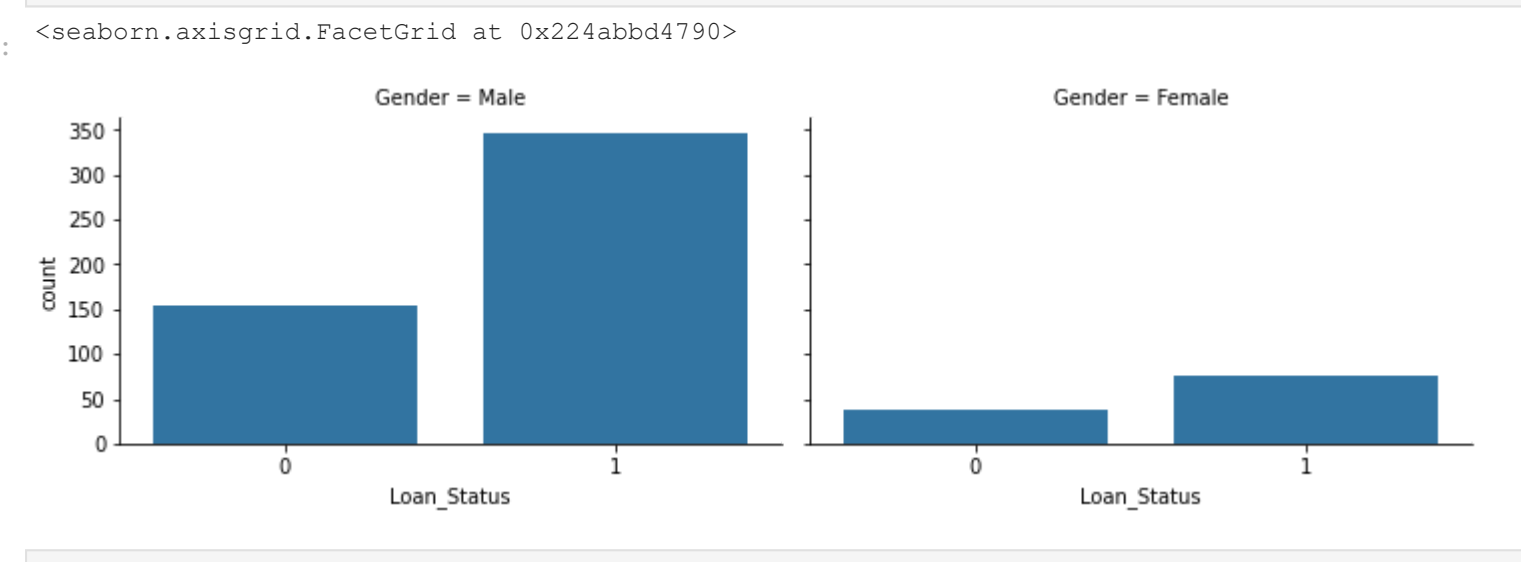
In [36]:
```python
#Gender vs loan status
sns.countplot(x ='Gender', data = df)
```

Out[36]: <AxesSubplot:xlabel='Gender', ylabel='count'>



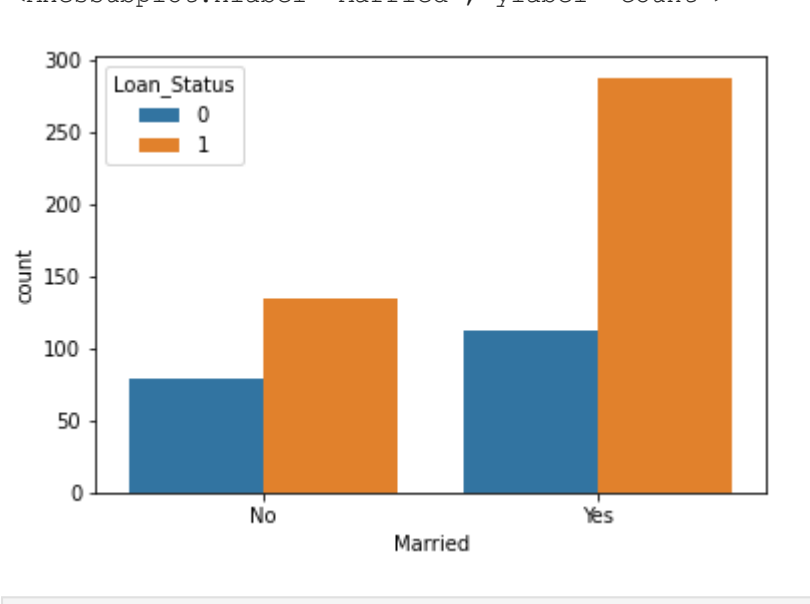
In [37]:
```python
grid = sns.FacetGrid(df,col='Gender', size=3.2, aspect=1.6)
grid.map(sns.countplot, 'Loan_Status')

# chances for getting loan for female is easier compared to male.
#Loan status clearly depend upon the gender.
```

Out[37]: <seaborn.axisgrid.FacetGrid at 0x224abbd4790>



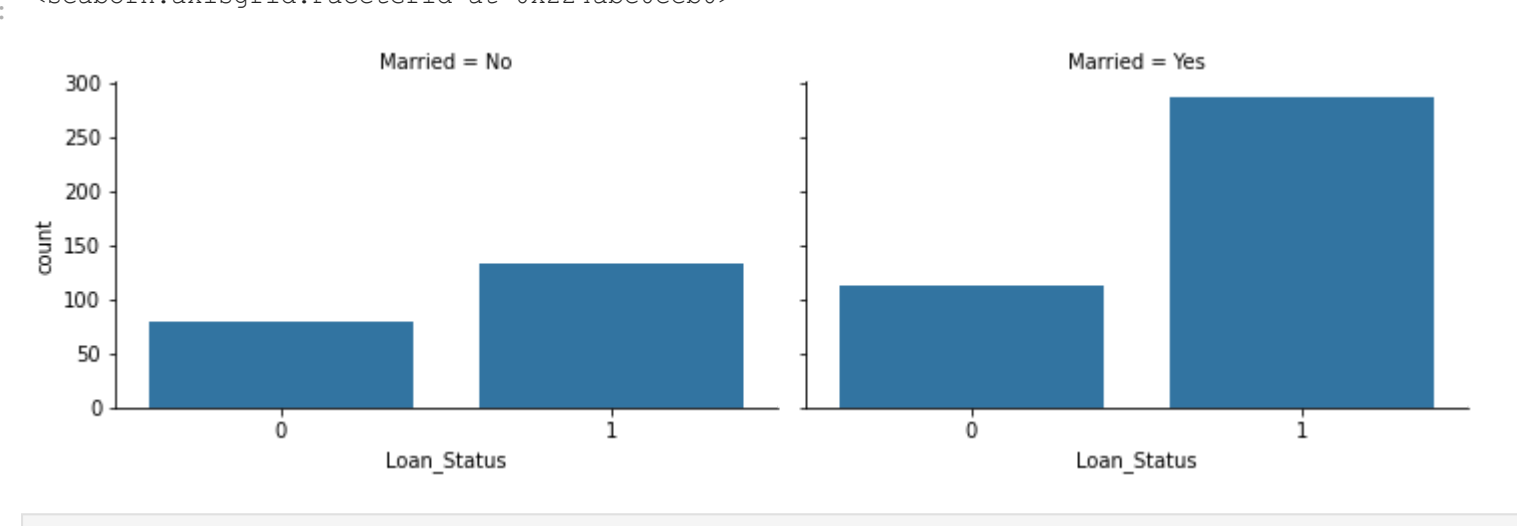
In [38]:
```python
#Married vs loan status
sns.countplot(x='Married', hue='Loan_Status', data=df)

#people who are married have better chance at loan approval
```

Out[38]: <AxesSubplot:xlabel='Married', ylabel='count'>



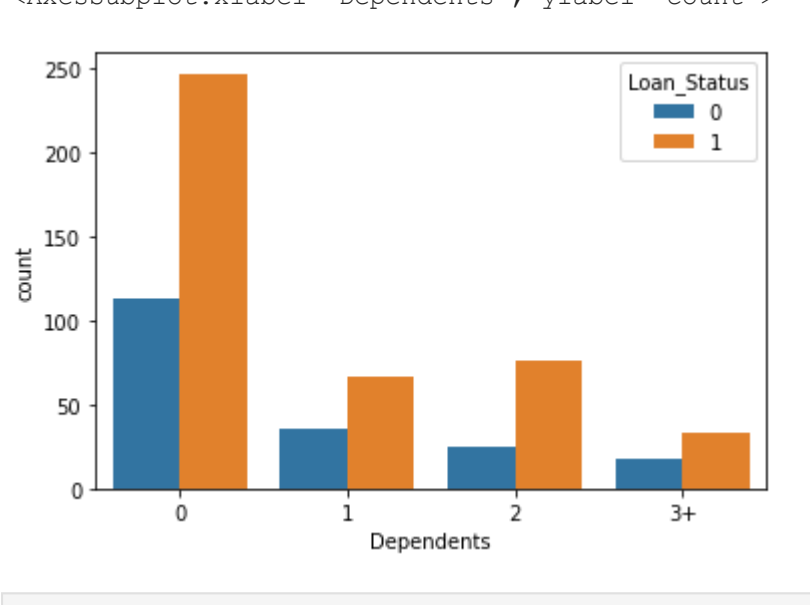
In [39]:
```python
grid = sns.FacetGrid(df,col='Married', size=3.2, aspect=1.6)
grid.map(sns.countplot, 'Loan_Status')
```

Out[39]: <seaborn.axisgrid.FacetGrid at 0x224abe0eeb0>



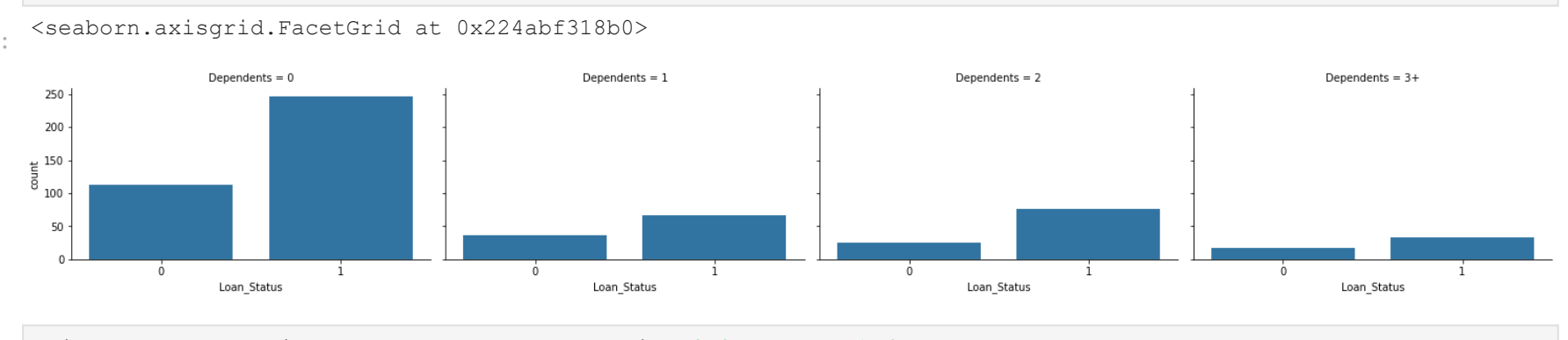
In [40]:
```python
#Dependents vs loan status

#sns.barplot(x='Dependents', y='Loan_Status', data=df)
sns.countplot(x="Dependents", hue="Loan_Status", data=df)
```

Out[40]: <AxesSubplot:xlabel='Dependents', ylabel='count'>



In [41]:
```python
grid = sns.FacetGrid(df,col='Dependents', size=3.2, aspect=1.6)
grid.map(sns.countplot, 'Loan_Status')
```

Out[41]: <seaborn.axisgrid.FacetGrid at 0x224abf318b0>

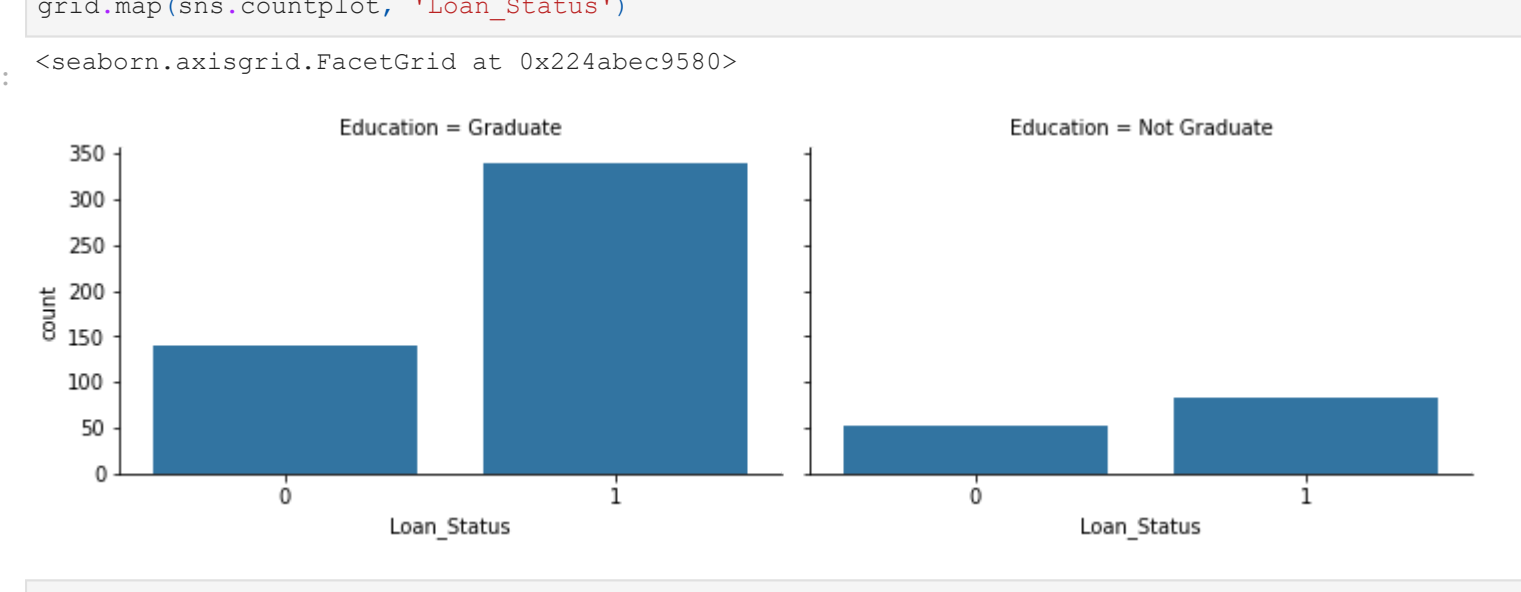


In [42]:
```python
grid = sns.FacetGrid(df,col='Loan_Status', size=3.2, aspect=1.6)
grid.map(sns.countplot, 'Dependents')

#we should drop the dependents as it has no relation with loan status
```

Out[42]: <seaborn.axisgrid.FacetGrid at 0x224ad4f8340>

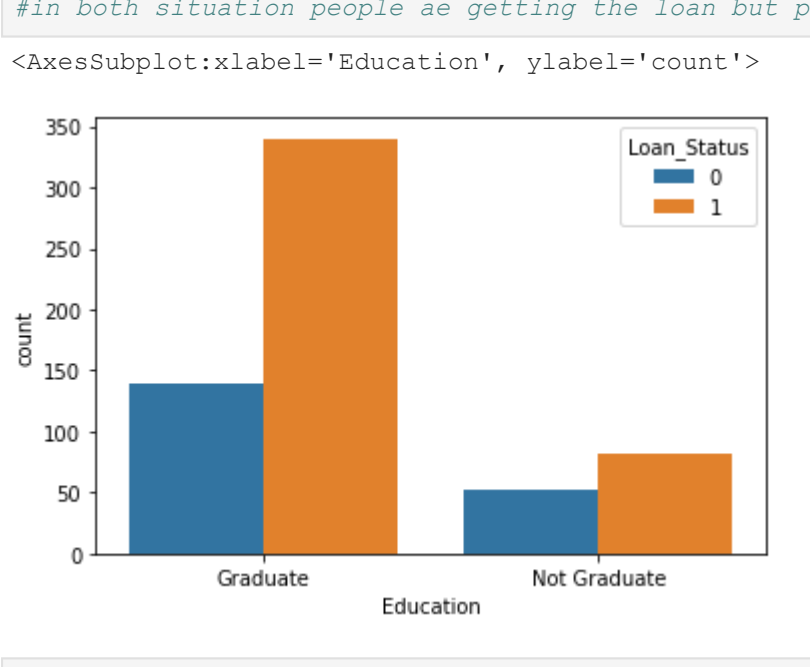
In [43]:
```python
#loan status vs Education
grid = sns.FacetGrid(df,col='Education', size=3.2, aspect=1.6)
grid.map(sns.countplot, 'Loan_Status')
```

Out[43]: <seaborn.axisgrid.FacetGrid at 0x224abec9580>



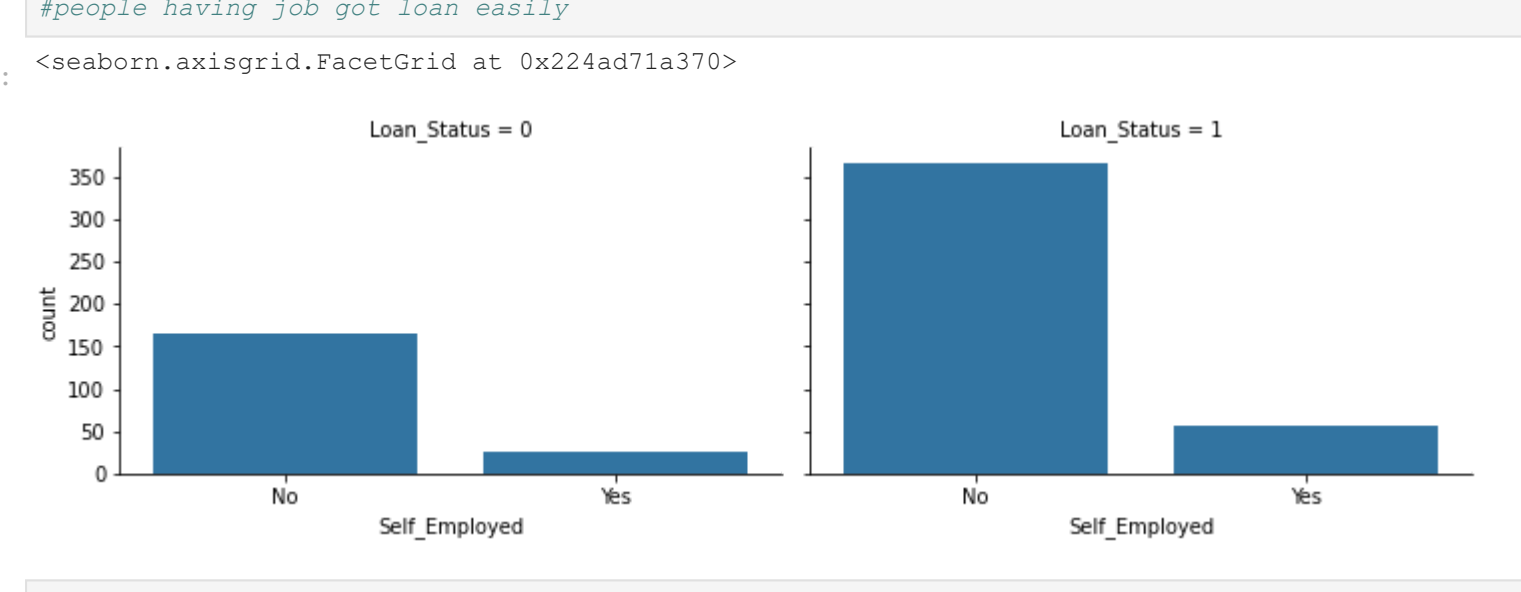
In [44]:
```python
sns.countplot(x="Education", hue="Loan_Status", data=df)
#in both situation people ae getting the loan but people who are graduate are getting loan easier compared to o
```

Out[44]: <AxesSubplot:xlabel='Education', ylabel='count'>



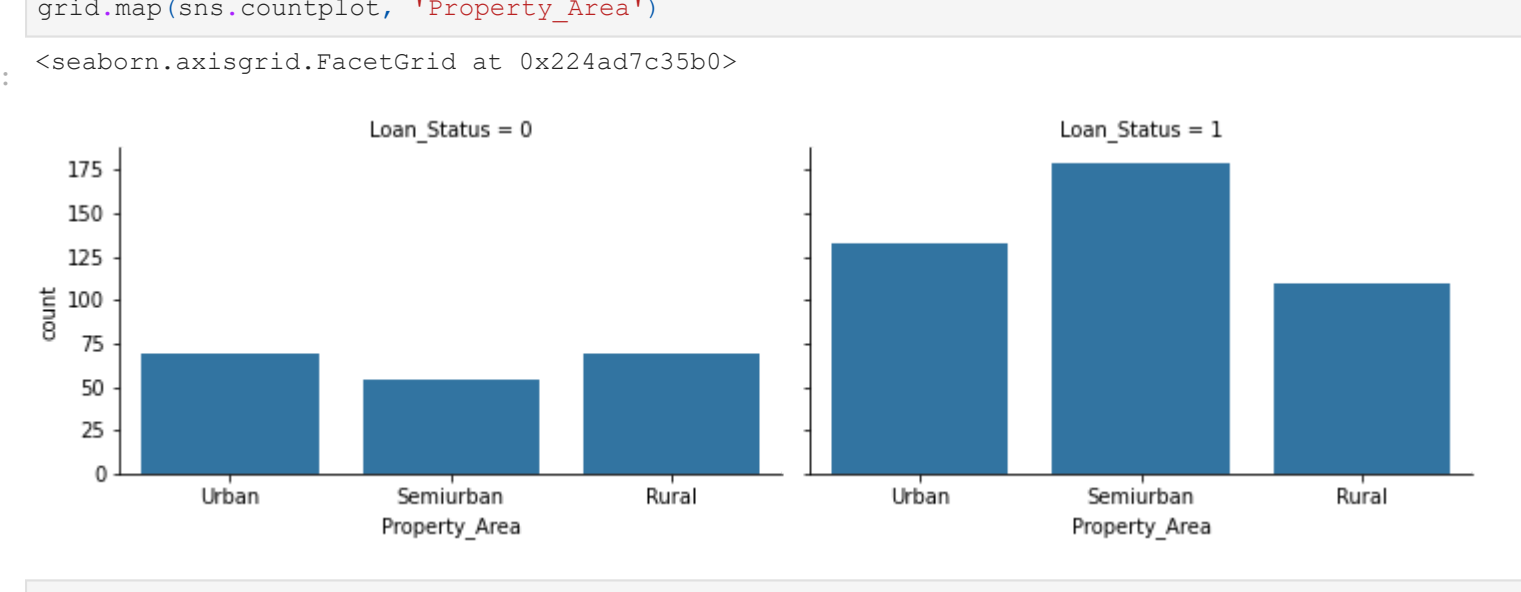
In [45]:
```python
#Self_Employed vs Education
grid = sns.FacetGrid(df,col='Loan_Status', size=3.2, aspect=1.6)
grid.map(sns.countplot, 'Self_Employed')

#people having job got loan easily
```

Out[45]: <seaborn.axisgrid.FacetGrid at 0x224ad71a370>



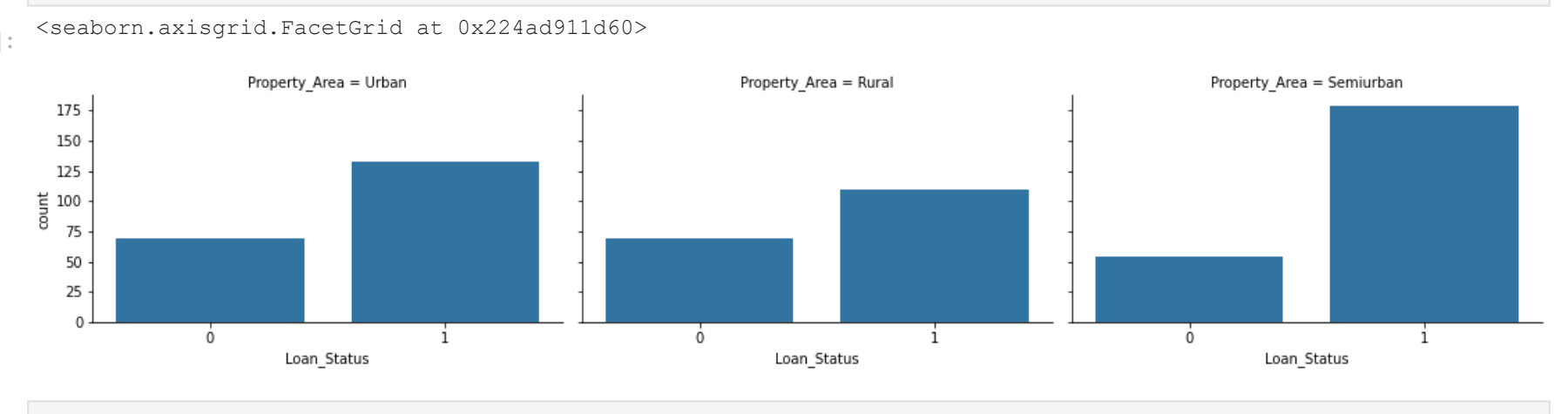
In [46]:
```python
grid = sns.FacetGrid(df,col='Loan_Status', size=3.2, aspect=1.6)
grid.map(sns.countplot, 'Property_Area')
```

Out[46]: <seaborn.axisgrid.FacetGrid at 0x224ad7c35b0>



In [47]:
```python
grid = sns.FacetGrid(df,col='Property_Area', size=3.2, aspect=1.6)
grid.map(sns.countplot, 'Loan_Status')

# property area has impact on loan status
```

Out[47]: <seaborn.axisgrid.FacetGrid at 0x224ad911d60>



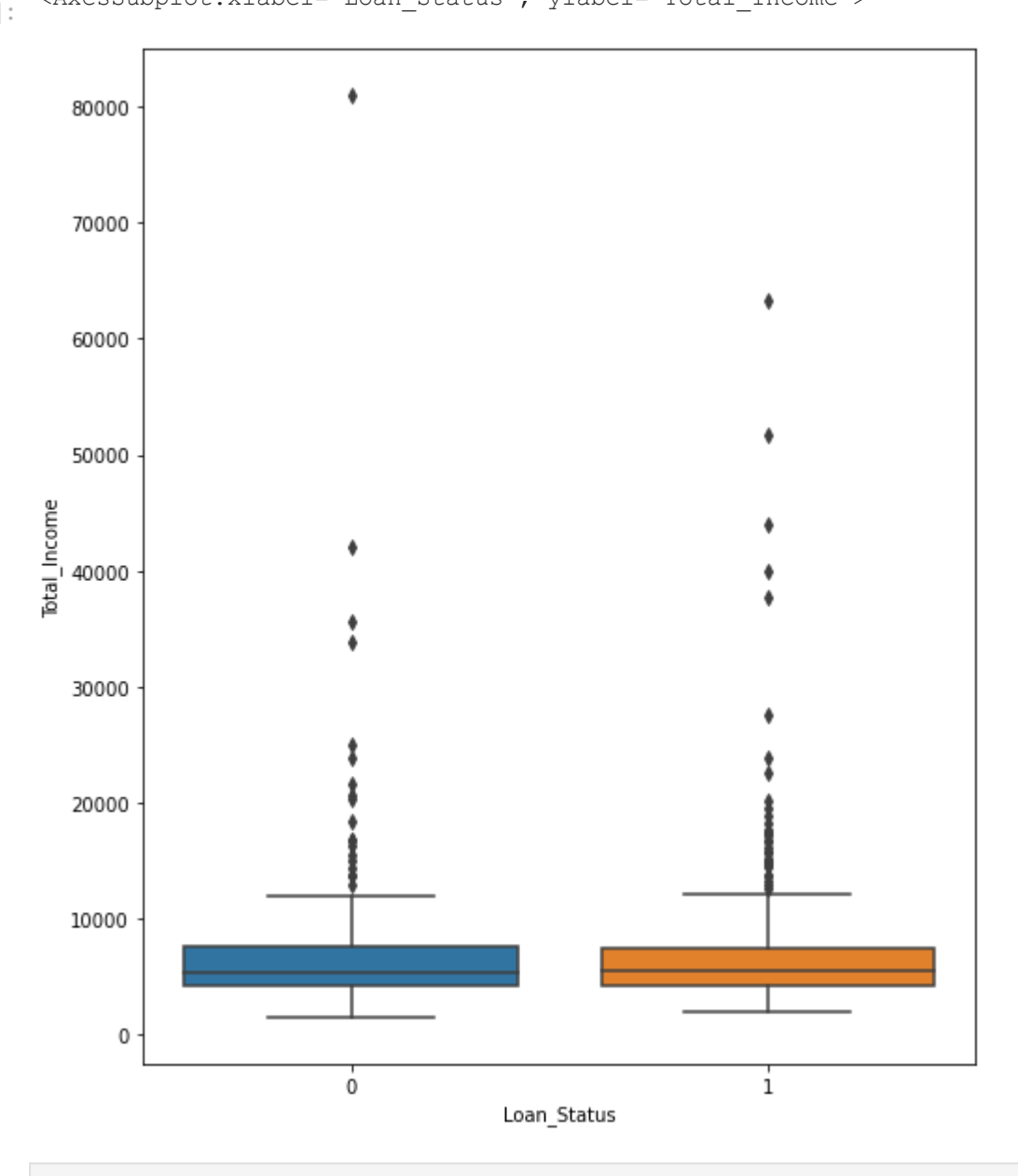
In [48]:
```python
df['Total_Income'] = df['ApplicantIncome'] + df['CoapplicantIncome']
df.head()
```

Out[48]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	LP001002	Male	No	0	Graduate	No	5849	0.0	128.0	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

In [49]:
```python
plt.figure(figsize=(8,10))
sns.boxplot(x="Loan_Status",y="Total_Income", data=df)
```

Out[49]: <AxesSubplot:xlabel='Loan_Status', ylabel='Total_Income'>

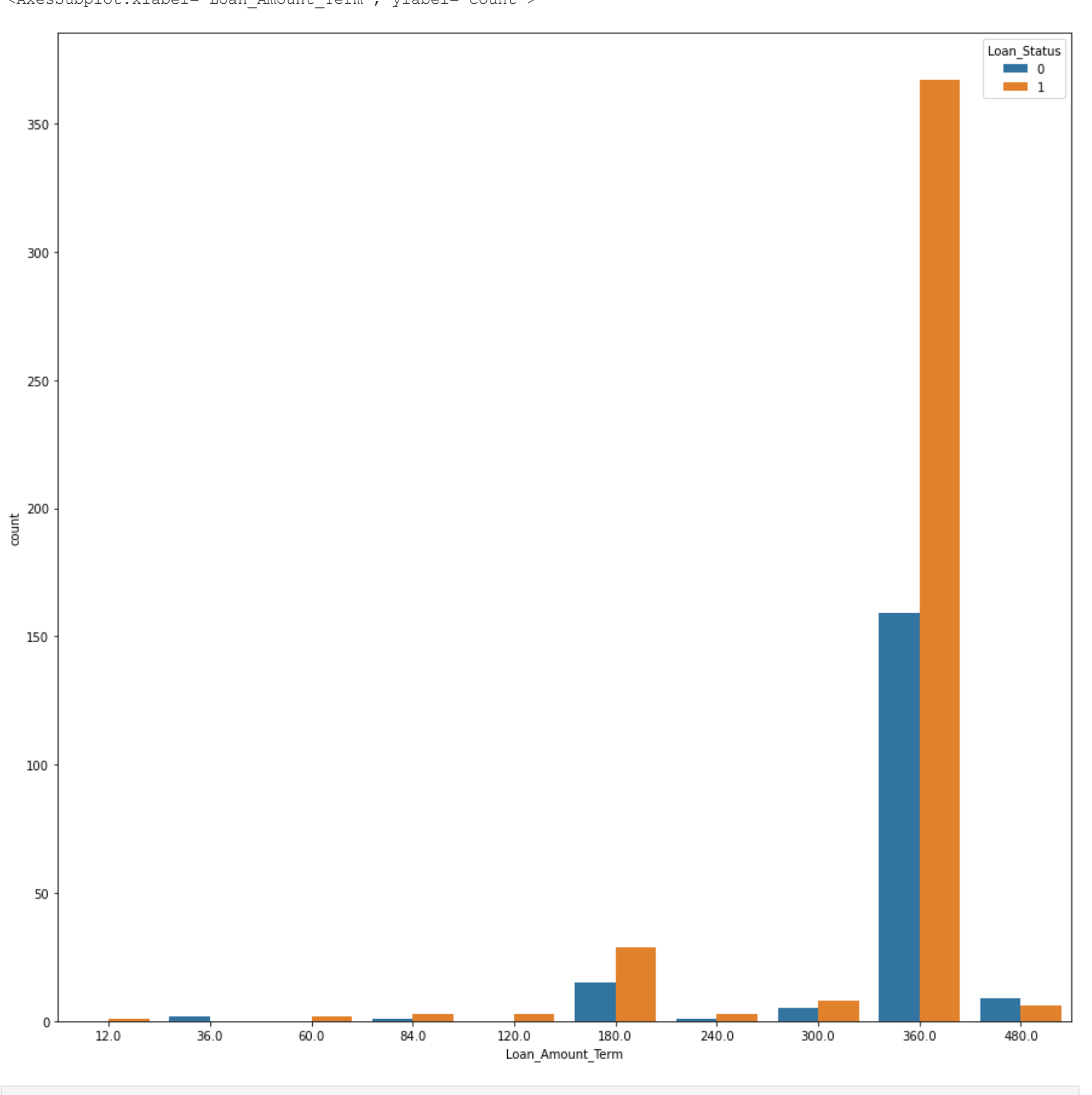


In [50]:
```python
df['Loan_Amount_Term'].unique()
```

Out[50]: array([360., 120., 240., 180., 60., 300., 480., 36., 84., 12.])

In [51]:
```python
plt.figure(figsize=(15,15))
sns.countplot(x="Loan_Amount_Term", hue="Loan_Status", data=df)
#no patter
```

Out[51]: <AxesSubplot:xlabel='Loan_Amount_Term', ylabel='count'>

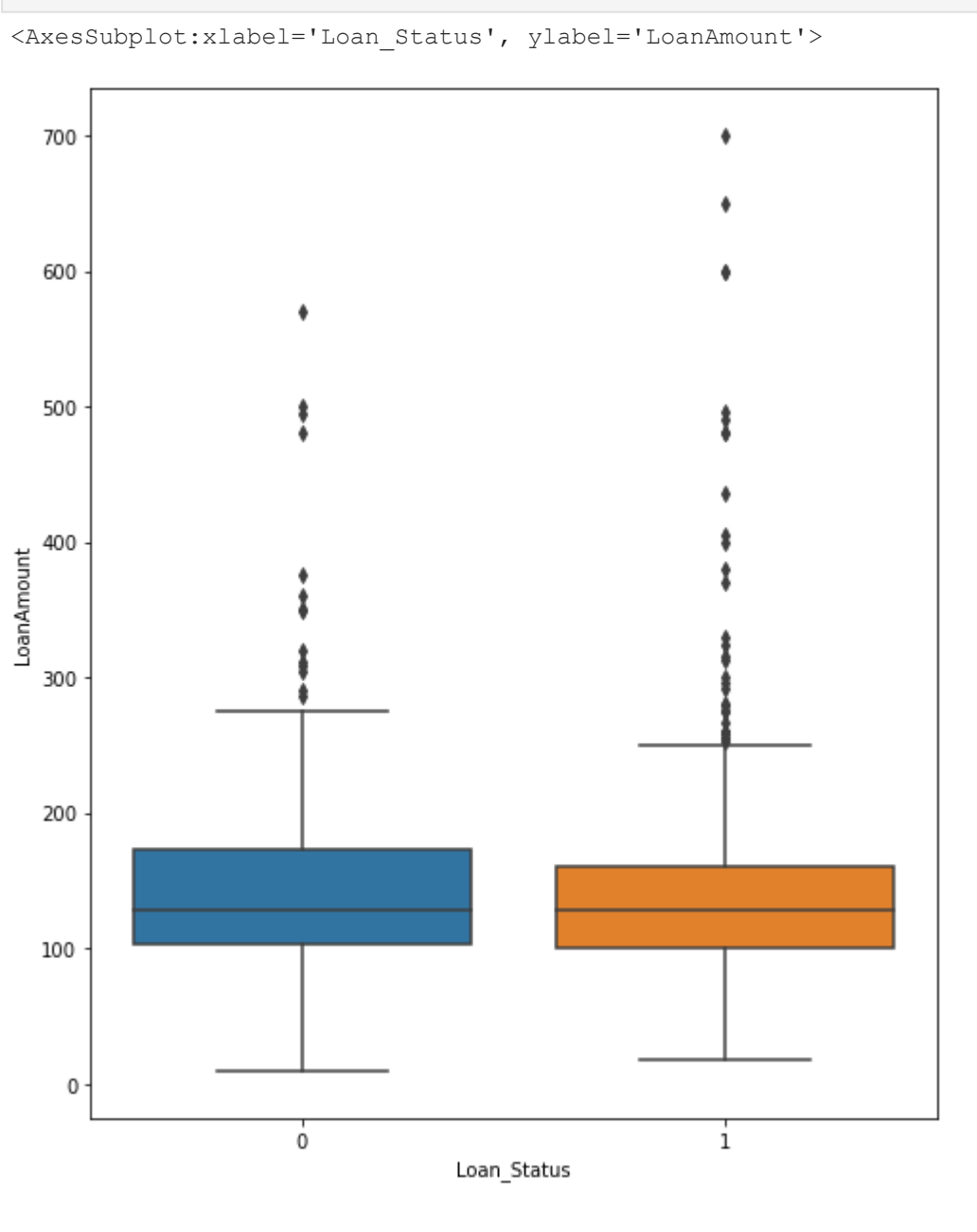


In [52]:
```python
df['LoanAmount'].unique()
```

Out[52]:
```
array([128.,  66., 120., 141., 267.,  95., 158., 168., 349.,  70., 109.,
       200., 114.,  17., 125., 100.,  76., 133., 115., 104., 315., 116.,
       112., 151., 191., 122., 110.,  35., 201.,  74., 106., 320., 144.,
       184.,  80.,  47.,  75., 134.,  96.,  88.,  44., 286.,  97., 135.,
       180.,  99., 165., 258., 126., 312., 136., 172.,  81., 187., 113.,
       176., 130., 111., 167., 265.,  50., 210., 175., 131., 188.,  25.,
       137., 160., 225., 216.,  94., 139., 152., 118., 185., 154.,  85.,
       259., 194.,  93., 370., 182., 650., 102., 290.,  84., 242., 129.,
        30., 244., 600., 255.,  98., 275., 121.,  63., 700.,  87., 101.,
       495.,  67.,  73., 260., 108.,  58.,  48., 164., 170.,  83.,  90.,
       166., 124.,  55.,  59., 127., 214., 240.,  72.,  60., 138.,  42.,
       280., 140., 155., 123., 279., 192., 304., 330., 150., 207., 436.,
        78.,  54.,  89., 143., 105., 132., 480.,  56., 159., 300., 376.,
       117.,  71., 490., 173.,  46., 228., 308., 236., 570., 380., 296.,
       156., 103.,  45.,  65.,  53., 360.,  62., 218., 178., 239., 405.,
       148., 190., 149., 153., 162., 230.,  86., 234., 246., 500., 186.,
       119., 107., 209., 208., 243.,  40., 250., 311., 400., 161., 196.,
       324., 157., 145., 181.,  26., 211.,   9., 205.,  36.,  61., 146.,
       292., 142., 350., 496., 253.])
```

In [53]:
```python
plt.figure(figsize=(8,10))
sns.boxplot(x="Loan_Status",y="LoanAmount", data=df)
#no pattern
```

Out[53]: <AxesSubplot:xlabel='Loan_Status', ylabel='LoanAmount'>



Drop featrues which has no use

In [54]:
```python
cols = ['ApplicantIncome', 'CoapplicantIncome', "LoanAmount", "Loan_Amount_Term", "Total_Income", 'Loan_ID', 'C
df = df.drop(columns=cols, axis=1)
df.head()
```

Out[54]:

	Gender	Married	Education	Self_Employed	Credit_History	Property_Area	Loan_Status
0	Male	No	Graduate	No	1.0	Urban	1
1	Male	Yes	Graduate	No	1.0	Rural	0
2	Male	Yes	Graduate	Yes	1.0	Urban	1
3	Male	Yes	Not Graduate	No	1.0	Urban	1
4	Male	No	Graduate	No	1.0	Urban	1

In [ ]: