Dataset

Import modules

In [1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib as plt
%matplotlib inline
```

Loading the dataset

In [2]:
```python
df = pd.read_csv('Loan Prediction Dataset.csv')
df.head()
```

Out[2]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loa
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	

In [3]:
```python
df.describe()
```

Out[3]:

	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 [4]:
```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
```

preprocessing the dataset

In [5]:
```python
# find the null values
df.isnull().sum()
```

Out[5]:
```
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 [6]:
```python
# fill the missing values for numerical terms - mean
df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].mean())
df['Loan_Amount_Term'] = df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean())
df['Credit_History'] = df['Credit_History'].fillna(df['Credit_History'].mean())
```

In [7]:
```python
# fill the missing values for categorical terms - mode
df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])
df['Married'] = df['Married'].fillna(df['Married'].mode()[0])
df['Dependents'] = df['Dependents'].fillna(df['Dependents'].mode()[0])
df['Self_Employed'] = df['Self_Employed'].fillna(df['Self_Employed'].mode()[0])
```

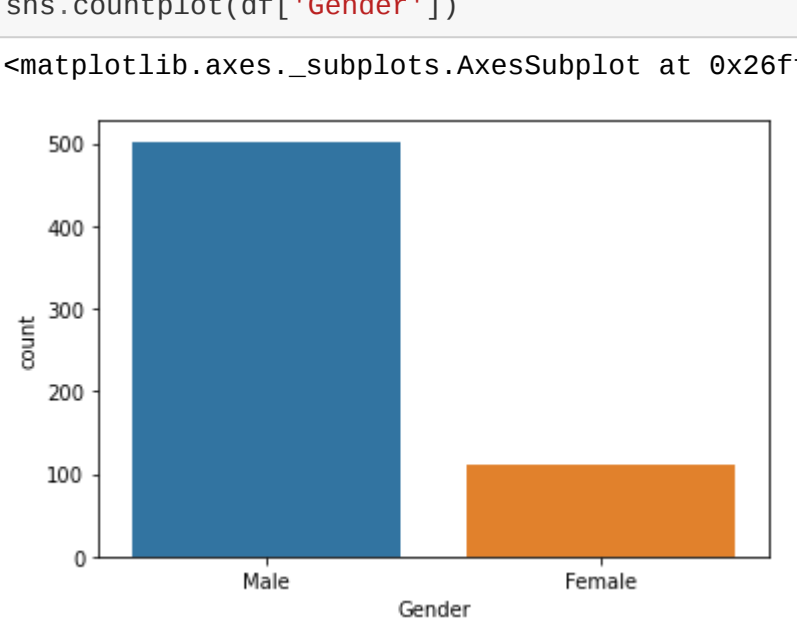
In [8]:
```python
df.isnull().sum()
```

Out[8]:
```
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
```

Exploratory Data Analysis

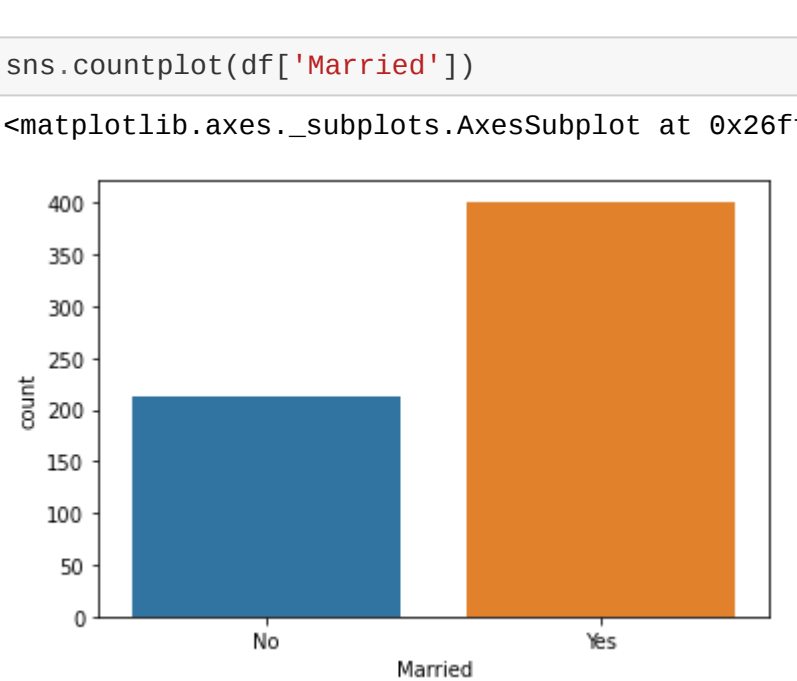
In [49]:
```python
#categorical attributes visualization
sns.countplot(df['Gender'])
```

Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffa6347f0>



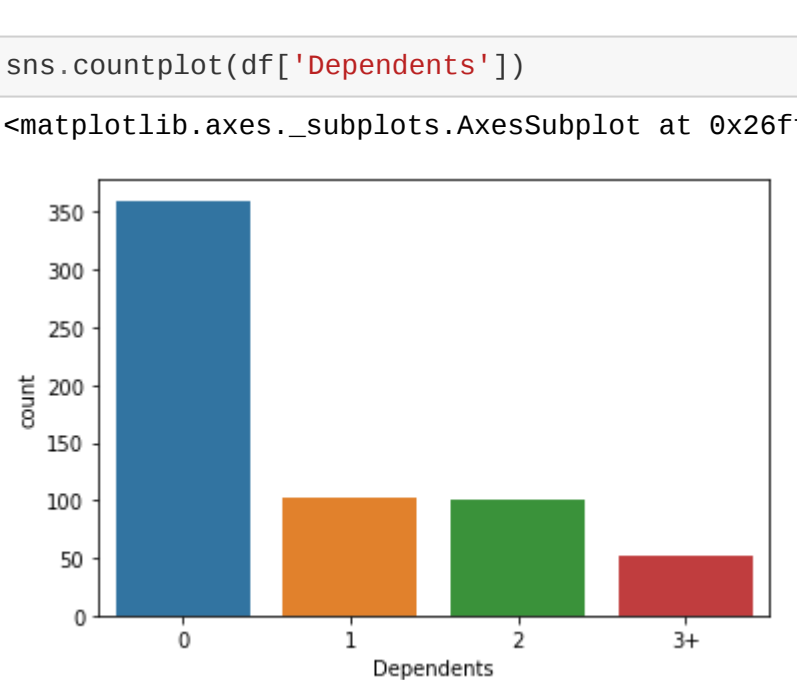
In [50]:
```python
sns.countplot(df['Married'])
```

Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffa66c610>



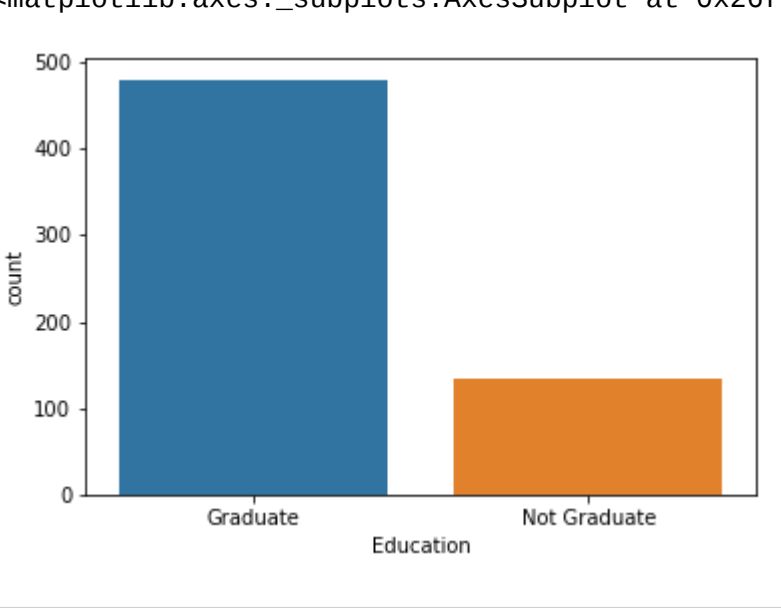
In [51]:
```python
sns.countplot(df['Dependents'])
```

Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffa6341f0>



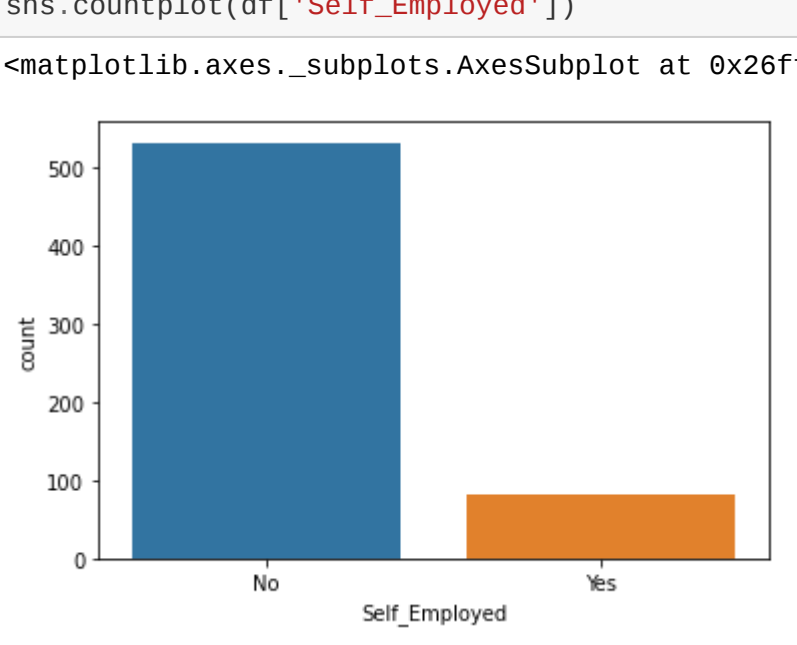
In [52]:
```python
sns.countplot(df['Education'])
```

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffa258970>



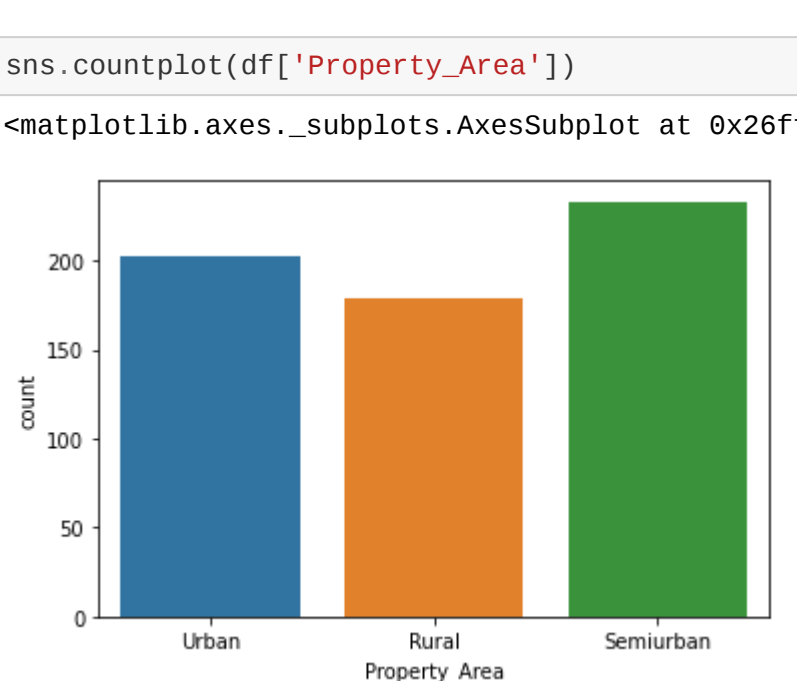
In [53]:
```python
sns.countplot(df['Self_Employed'])
```

Out[53]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffa72e8b0>



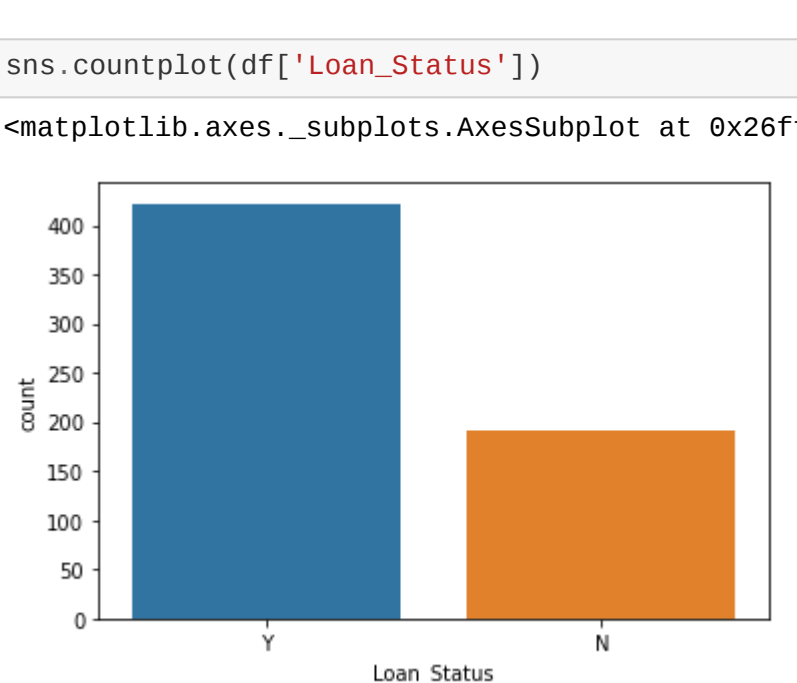
In [54]:
```python
sns.countplot(df['Property_Area'])
```

Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffa7789d0>



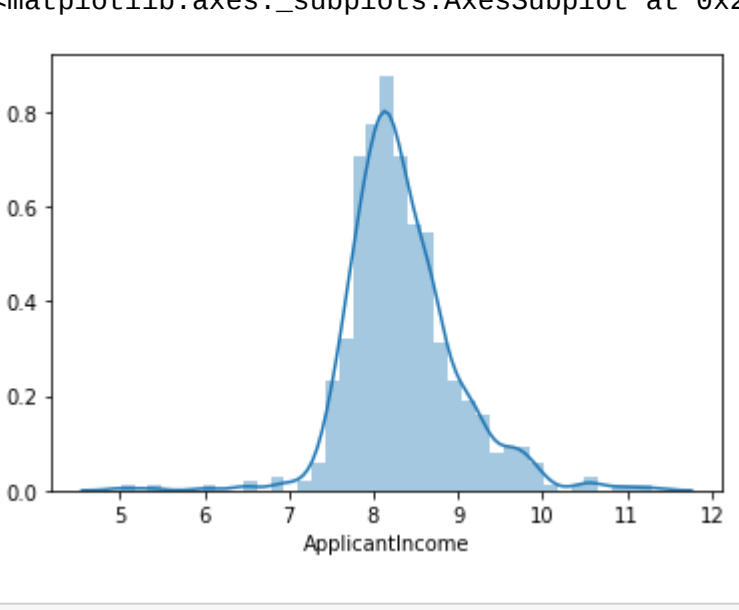
In [55]:
```python
sns.countplot(df['Loan_Status'])
```

Out[55]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffa7b5d60>



In [56]:
```python
# numerical attributes visualization
sns.distplot(df['ApplicantIncome'])
```

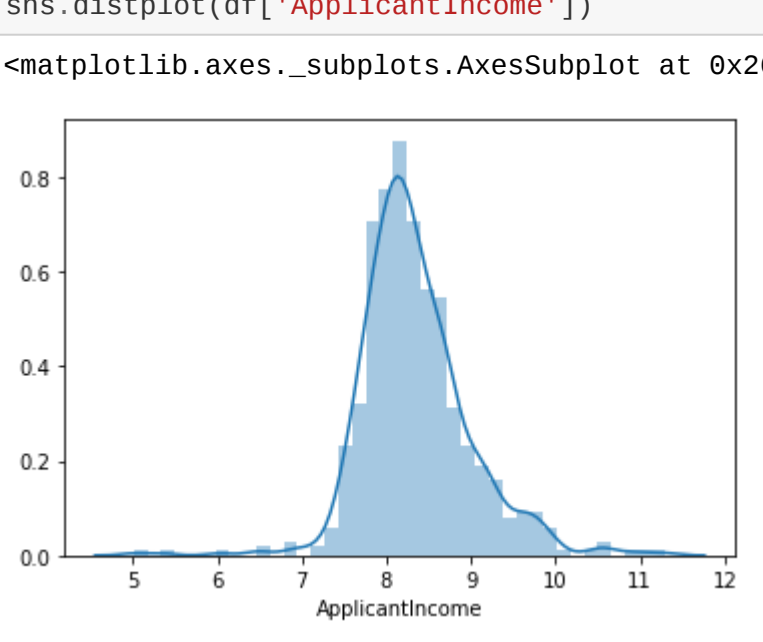
Out[56]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffb7de6d0>



In [17]:
```python
#apply log transformation to the attribute
df['ApplicantIncome'] = np.log(df['ApplicantIncome'])
```

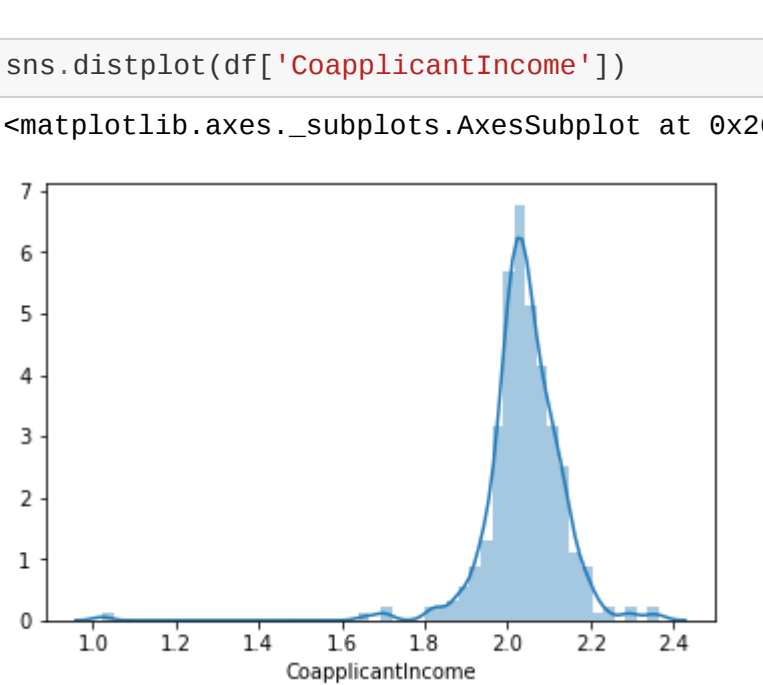
In [18]:
```python
sns.distplot(df['ApplicantIncome'])
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x26ff87504c0>



In [57]:
```python
sns.distplot(df['CoapplicantIncome'])
```

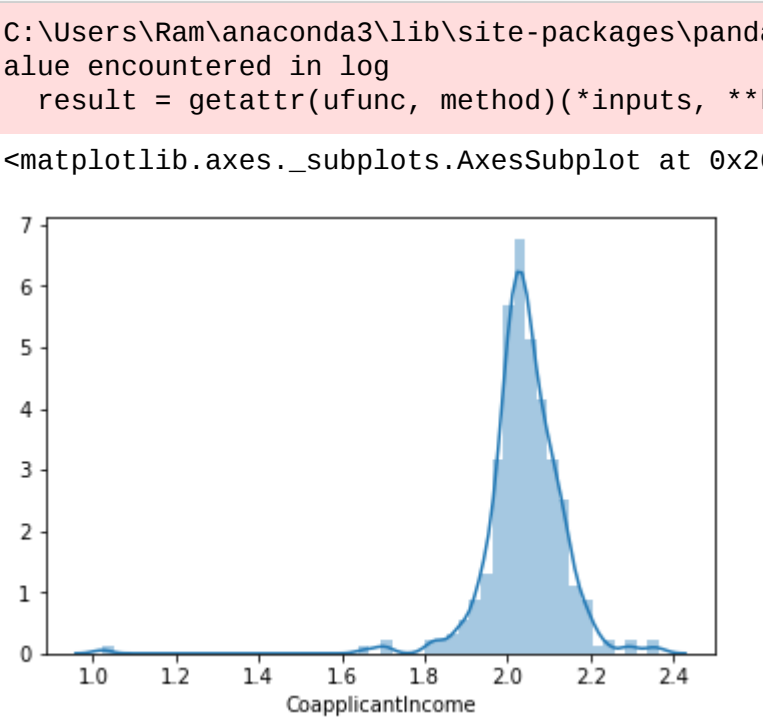
Out[57]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffb8aa820>



In [21]:
```python
df['CoapplicantIncome'] = np.log(df['CoapplicantIncome'])
sns.distplot(df['CoapplicantIncome'])
```

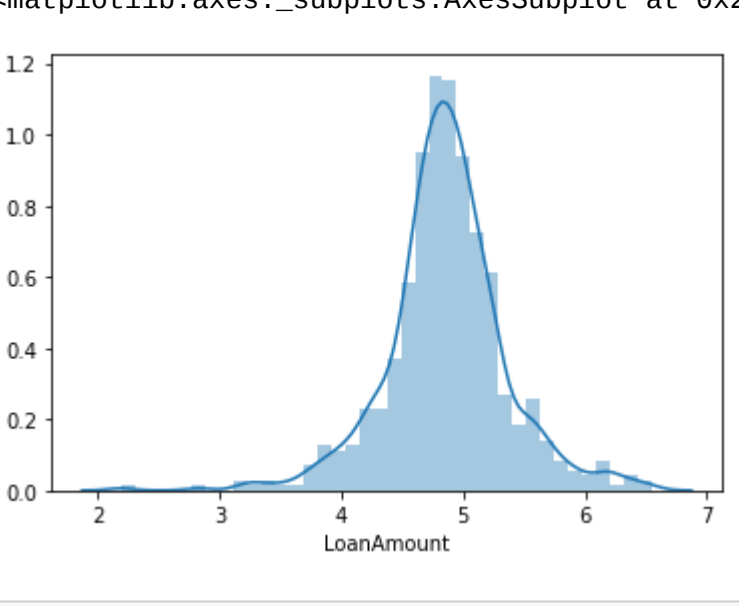
```
C:\Users\Ram\anaconda3\lib\site-packages\pandas\core\series.py:679: RuntimeWarning: invalid v
alue encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x26ff86746a0>



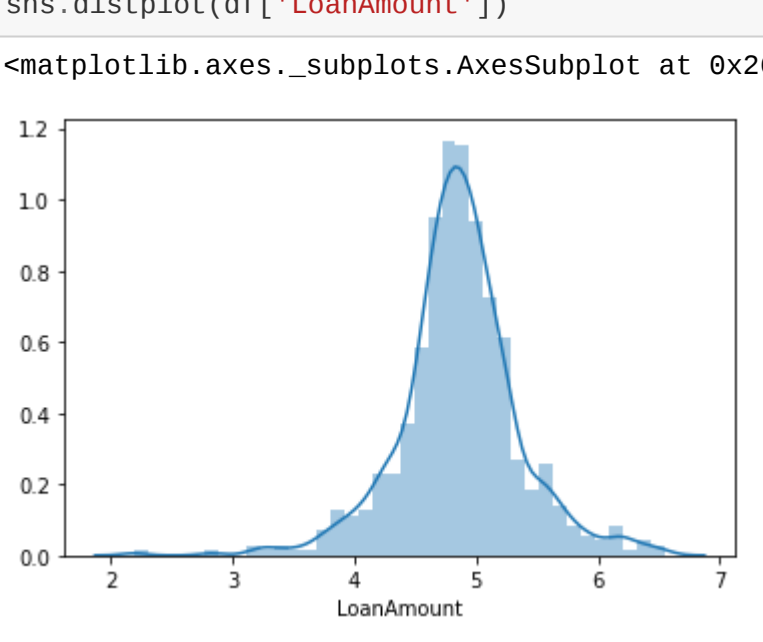
In [58]:
```python
sns.distplot(df['LoanAmount'])
```

Out[58]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffb90b850>



In [23]:
```python
df['LoanAmount'] = np.log(df['LoanAmount'])
sns.distplot(df['LoanAmount'])
```

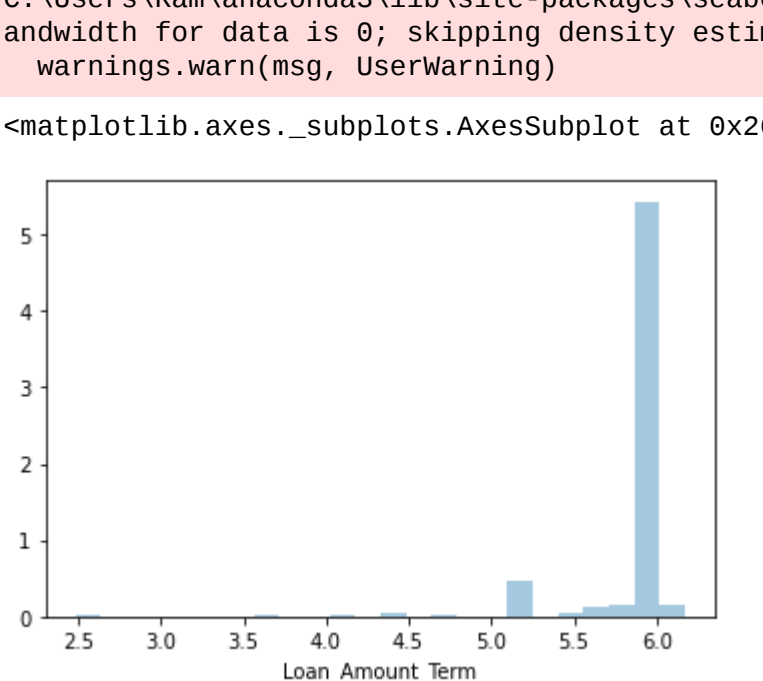
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x26ff8c0f8e0>



In [59]:
```python
sns.distplot(df['Loan_Amount_Term'])
```

```
C:\Users\Ram\anaconda3\lib\site-packages\seaborn\distributions.py:369: UserWarning: Default b
andwidth for data is 0; skipping density estimation.
  warnings.warn(msg, UserWarning)
```

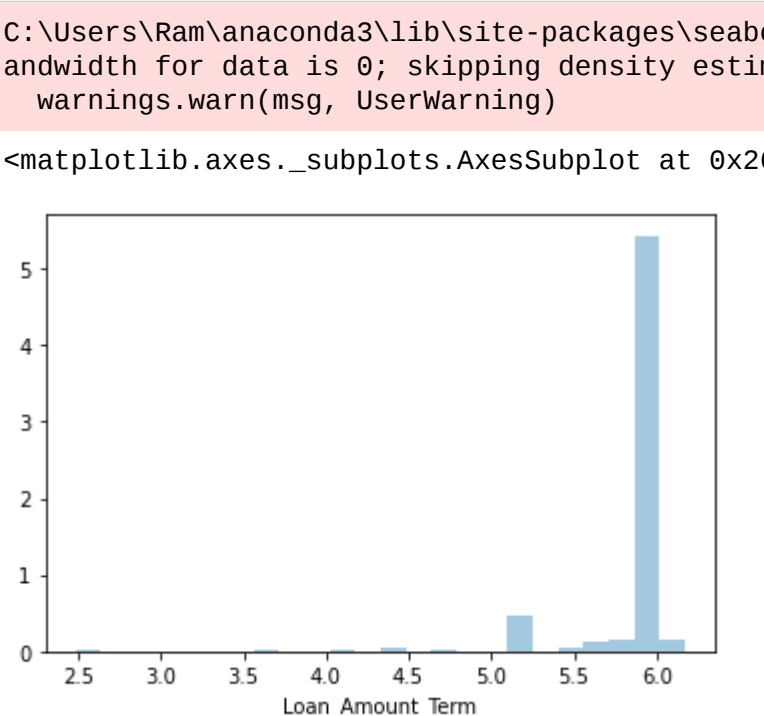
Out[59]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffba36760>



In [25]:
```python
df['Loan_Amount_Term'] = np.log(df['Loan_Amount_Term'])
sns.distplot(df['Loan_Amount_Term'])
```

```
C:\Users\Ram\anaconda3\lib\site-packages\seaborn\distributions.py:369: UserWarning: Default b
andwidth for data is 0; skipping density estimation.
  warnings.warn(msg, UserWarning)
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x26ff8d63190>



In [60]:
```python
sns.distplot(df['Credit_History'])
```

```
C:\Users\Ram\anaconda3\lib\site-packages\seaborn\distributions.py:369: UserWarning: Default b
andwidth for data is 0; skipping density estimation.
  warnings.warn(msg, UserWarning)
```

Out[60]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffbad1d00>

Creation of new attribute

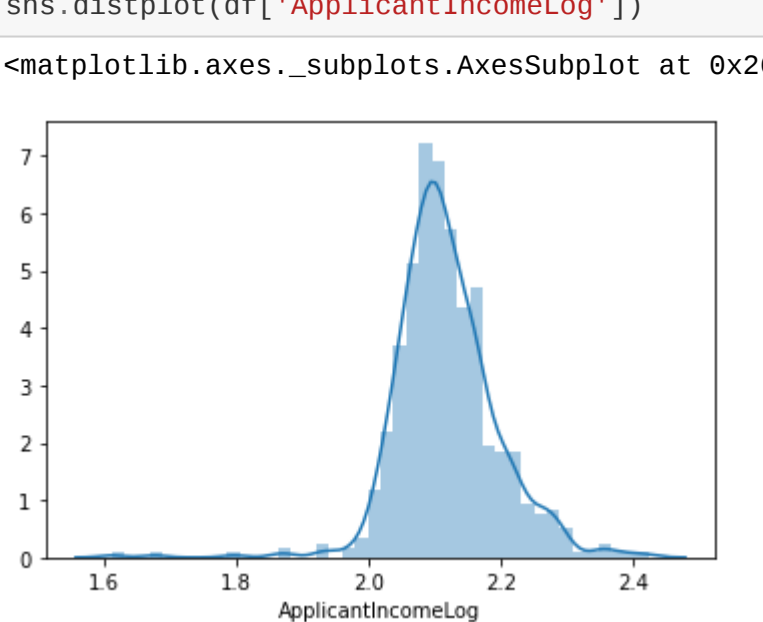
In [61]:
```python
#total income
df['Total_Income'] = df['ApplicantIncome'] + df['CoapplicantIncome']
df.head()
```

Out[61]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loa
0	LP001002	Male	No	0	Graduate	No	8.674026	NaN	4.986426	
1	LP001003	Male	Yes	1	Graduate	No	8.430109	1.990411	4.852030	
2	LP001005	Male	Yes	0	Graduate	Yes	8.006368	NaN	4.189655	
3	LP001006	Male	Yes	0	Not Graduate	No	7.856707	2.049700	4.787492	
4	LP001008	Male	No	0	Graduate	No	8.699515	NaN	4.948760	

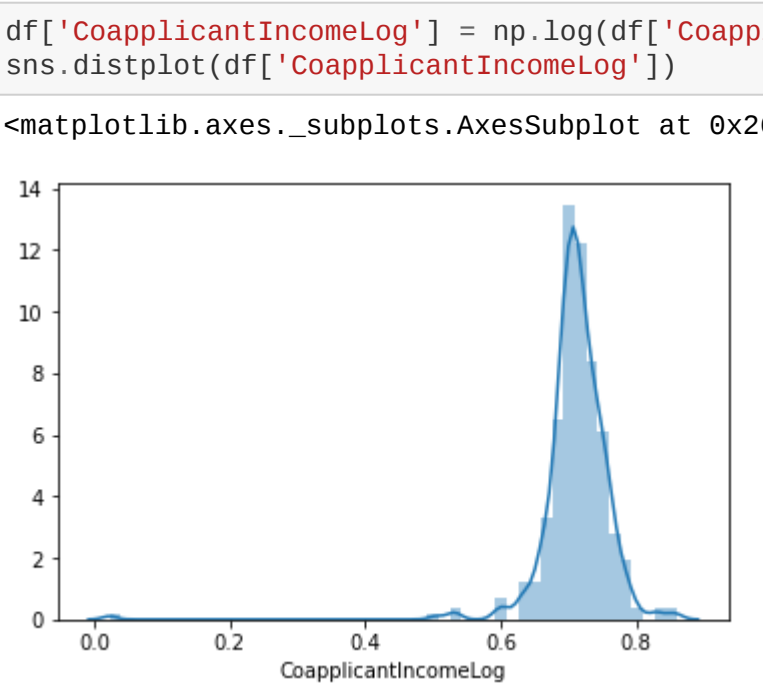
In [62]:
```python
df['ApplicantIncomeLog'] = np.log(df['ApplicantIncome'])
sns.distplot(df['ApplicantIncomeLog'])
```

Out[62]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffbb67fa0>



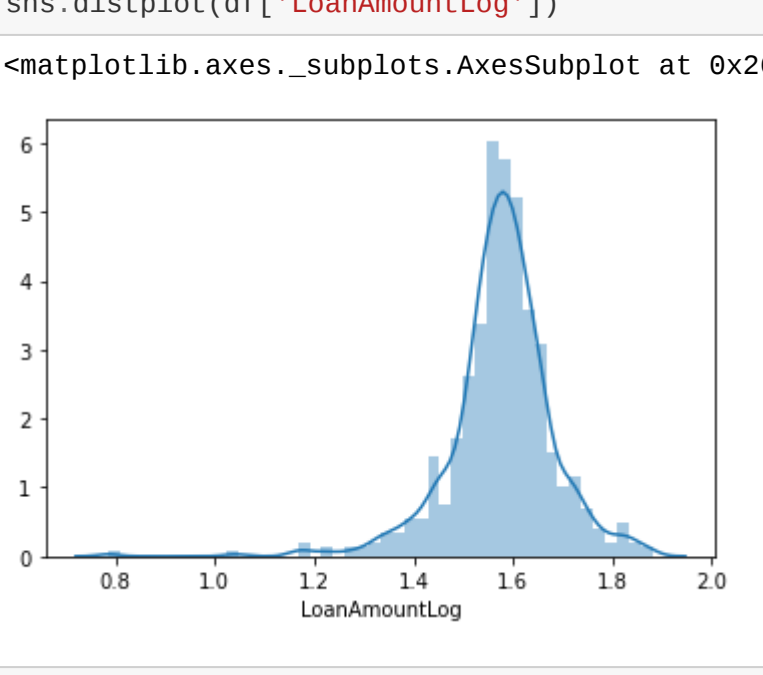
In [63]:
```python
df['CoapplicantIncomeLog'] = np.log(df['CoapplicantIncome'])
sns.distplot(df['CoapplicantIncomeLog'])
```

Out[63]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffbc2fb80>



In [64]:
```python
df['LoanAmountLog'] = np.log(df['LoanAmount'])
sns.distplot(df['LoanAmountLog'])
```

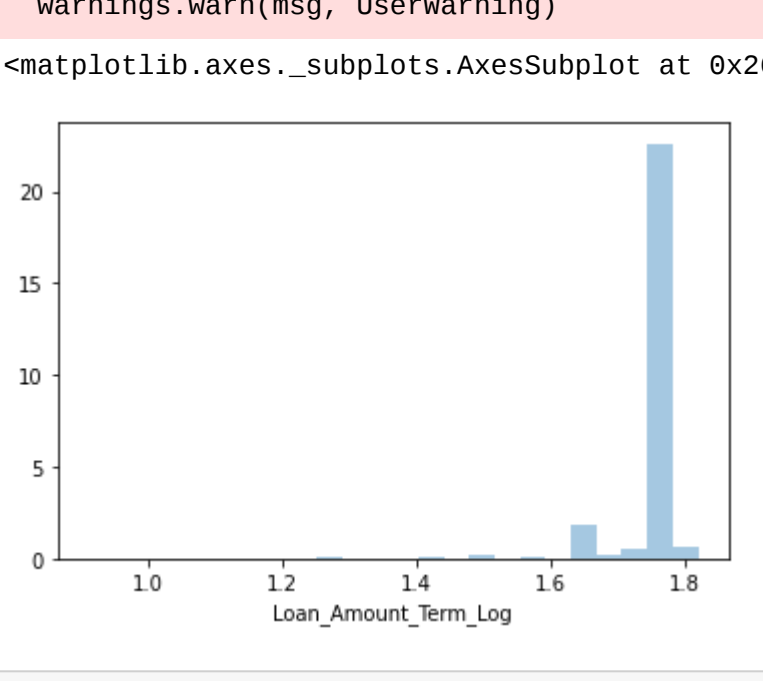
Out[64]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffbce68b0>



In [65]:
```python
df['Loan_Amount_Term_Log'] = np.log(df['Loan_Amount_Term'])
sns.distplot(df['Loan_Amount_Term_Log'])
```

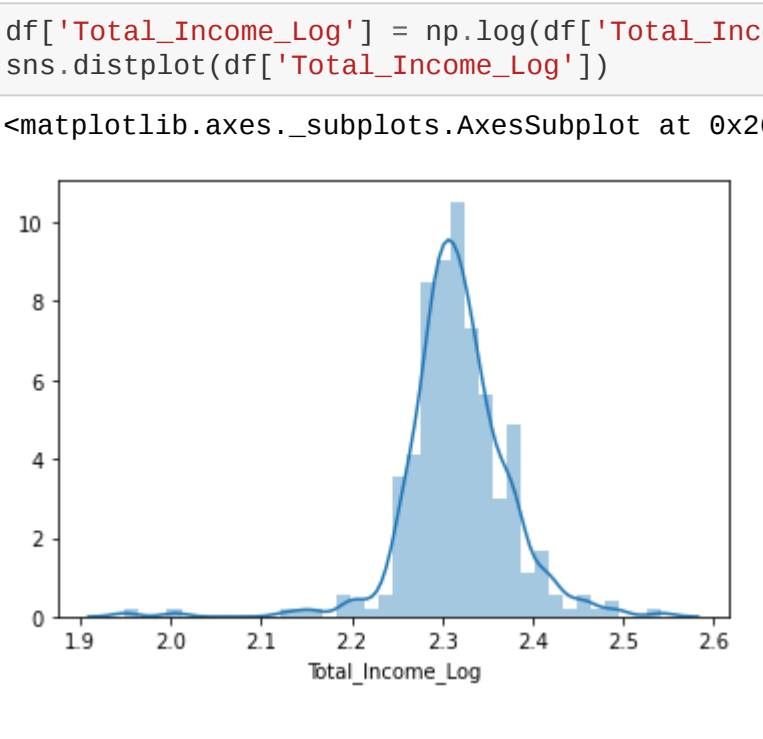
```
C:\Users\Ram\anaconda3\lib\site-packages\seaborn\distributions.py:369: UserWarning: Default b
andwidth for data is 0; skipping density estimation.
  warnings.warn(msg, UserWarning)
```

Out[65]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffbdc02e0>



In [66]:
```python
df['Total_Income_Log'] = np.log(df['Total_Income'])
sns.distplot(df['Total_Income_Log'])
```

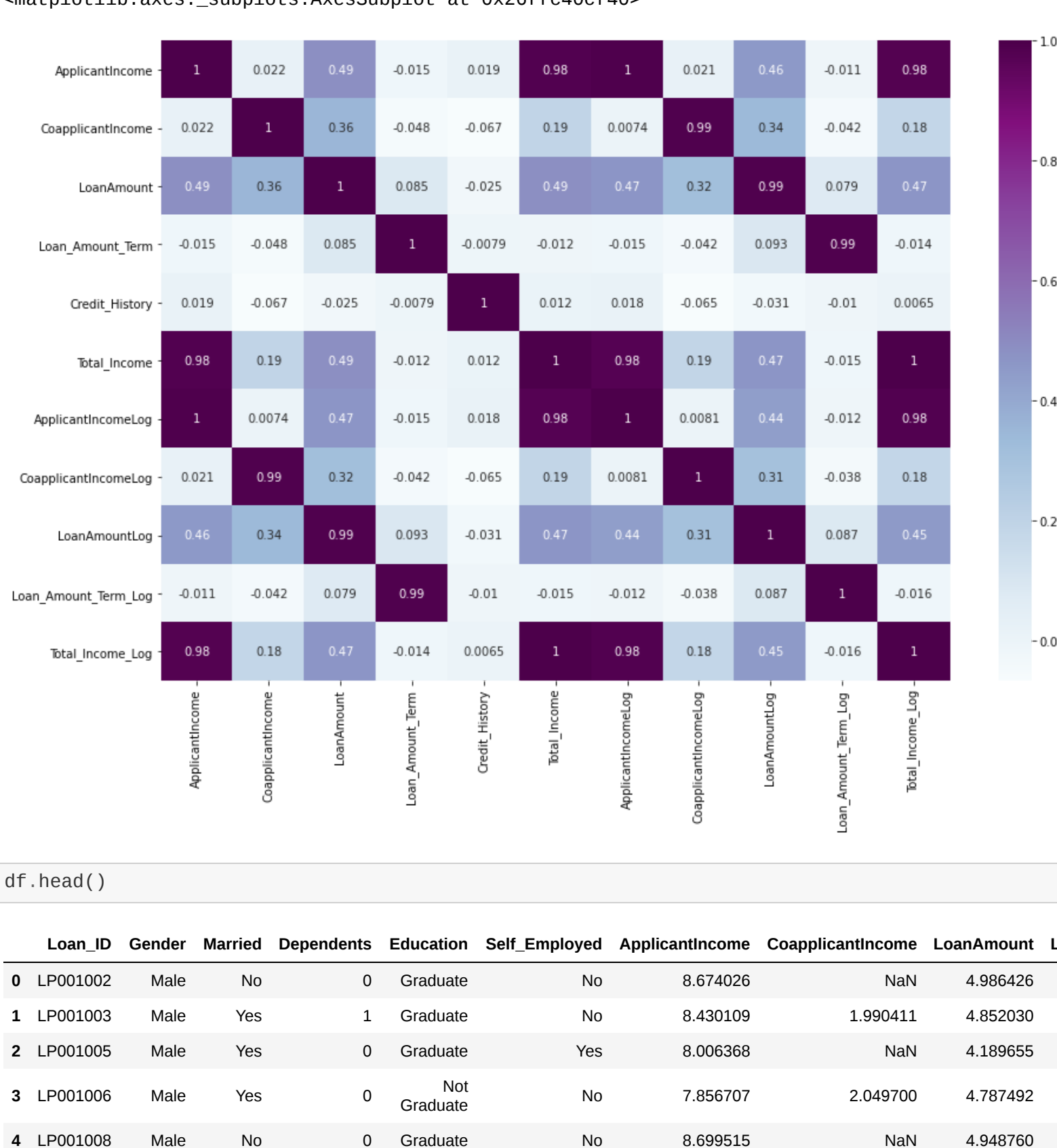
Out[66]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffbcf7520>



Coorelation Matrix

In [69]:
```python
from matplotlib import pyplot as plt
corr = df.corr()
plt.figure(figsize=(15,10))
sns.heatmap(corr, annot = True, cmap="BuPu")
```

Out[69]: <matplotlib.axes._subplots.AxesSubplot at 0x26ffc40cf40>



In [70]:
```python
df.head()
```

Out[70]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loa
0	LP001002	Male	No	0	Graduate	No	8.674026	NaN	4.986426	
1	LP001003	Male	Yes	1	Graduate	No	8.430109	1.990411	4.852030	
2	LP001005	Male	Yes	0	Graduate	Yes	8.006368	NaN	4.189655	
3	LP001006	Male	Yes	0	Not Graduate	No	7.856707	2.049700	4.787492	
4	LP001008	Male	No	0	Graduate	No	8.699515	NaN	4.948760	

In [71]:
```python
#drop unnecessary columns
cols = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term', 'Total_Inc
ome', 'Loan_ID', 'CoapplicantIncome']
df = df.drop(columns=cols, axis=1)
df.head()
```

Out[71]:

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	Loan_Status	ApplicantIncomeLog
0	Male	No	0	Graduate	No	1.0	Urban	Y	2.160333
1	Male	Yes	1	Graduate	No	1.0	Rural	N	2.131810
2	Male	Yes	0	Graduate	Yes	1.0	Urban	Y	2.080237
3	Male	Yes	0	Not Graduate	No	1.0	Urban	Y	2.061368
4	Male	No	0	Graduate	No	1.0	Urban	Y	2.163267

Label Encoding

In [86]:
```python
from sklearn.preprocessing import LabelEncoder
cols = ['Gender', 'Married', 'Education', 'Self_Employed', 'Property_Area', 'Loan_Status',
'Dependents']
le = LabelEncoder()
for col in cols:
    df[col] = le.fit_transform(df[col])
```

In [87]:
```python
df.head()
```

Out[87]:

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	Loan_Status	ApplicantIncomeLog
0	1	0	0	0	0	1.0	2	1	2.160333
1	1	1	1	0	0	1.0	0	0	2.131810
2	1	1	0	0	1	1.0	2	1	2.080237
3	1	1	0	1	0	1.0	2	1	2.061368
4	1	0	0	0	0	1.0	2	1	2.163267

Train-Test Split

In [80]:
```python
#specify input and output attributes
x = df.drop(columns=['Loan_Status'], axis=1)
y = df['Loan_Status']
```

In [81]:
```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42)
```

Model training

In [89]:
```python
# classify function
def classify(model, x,y):
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=4
2)
    model.fit(x_train, y_train)
    print("Accuracy is", model.score(x_test, y_test)*100)
    # cross validation - it is used for better validation of model
    #eg: cv-5, train-4, test-1
    #split the data in same manner
    score = cross_val_score(model, x, y, cv=5)
    print("Cross validation is",np.mean(score)*100)
```

In [ ]: