In [1]:
```
## Loading Packages
```

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

In [3]:
```
## Reading Data
## train_dataset
## test_dataset
## sample_submission
```

In [2]:
```
train = pd.read_csv('train_dataset.csv')
test = pd.read_csv('test_dataset.csv')
```

In [6]:
```
train.describe()
```

Out[6]:

| | 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 [7]:
```
test.describe()
```

Out[7]:

| | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_History |
|---|---|---|---|---|---|
| **count** | 367.000000 | 367.000000 | 362.000000 | 361.000000 | 338.000000 |
| **mean** | 4805.599455 | 1569.577657 | 136.132597 | 342.537396 | 0.825444 |
| **std** | 4910.685399 | 2334.232099 | 61.366652 | 65.156643 | 0.380150 |
| **min** | 0.000000 | 0.000000 | 28.000000 | 6.000000 | 0.000000 |
| **25%** | 2864.000000 | 0.000000 | 100.250000 | 360.000000 | 1.000000 |
| **50%** | 3786.000000 | 1025.000000 | 125.000000 | 360.000000 | 1.000000 |
| **75%** | 5060.000000 | 2430.500000 | 158.000000 | 360.000000 | 1.000000 |
| **max** | 72529.000000 | 24000.000000 | 550.000000 | 480.000000 | 1.000000 |

In [8]:
```
train.head(2)
```

Out[8]:

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInco |
|---|---|---|---|---|---|---|---|---|
| 0 | LP001002 | Male | No | 0 | Graduate | No | 5849 | |
| 1 | LP001003 | Male | Yes | 1 | Graduate | No | 4583 | 150 |

In [9]:
```
test.head()
```

Out[9]:

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInco |
|---|---|---|---|---|---|---|---|---|
| 0 | LP001015 | Male | Yes | 0 | Graduate | No | 5720 | |
| 1 | LP001022 | Male | Yes | 1 | Graduate | No | 3076 | 15 |
| 2 | LP001031 | Male | Yes | 2 | Graduate | No | 5000 | 18 |
| 3 | LP001035 | Male | Yes | 2 | Graduate | No | 2340 | 25 |
| 4 | LP001051 | Male | No | 0 | Not Graduate | No | 3276 | |

In [10]:
```
## Copying data ( for if changes are needed to make to the orginal data)
```

In [11]:
```
train_orginal = train.copy()
```

In [12]:
```
test_orginal = test.copy()
```

In [13]:
```
## Understannding the Data
```

In [14]:
```
train.columns
```

Out[14]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [15]:
```
test. columns
```

Out[15]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
      dtype='object')
```

In [16]:
```
train.dtypes
```

Out[16]:
```
Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
Loan_Status           object
dtype: object
```

In [17]:
```
## Objects --> Categorical
```

In [18]:
```
train.shape, test.shape
```

Out[18]:
```
((614, 13), (367, 12))
```

In [19]:
```
## Univariate Analysis
##
## Target Variable --> Loan_Status
```

In [20]:
```
train['Loan_Status'].value_counts()
```

Out[20]:
```
Y    422
N    192
Name: Loan_Status, dtype: int64
```

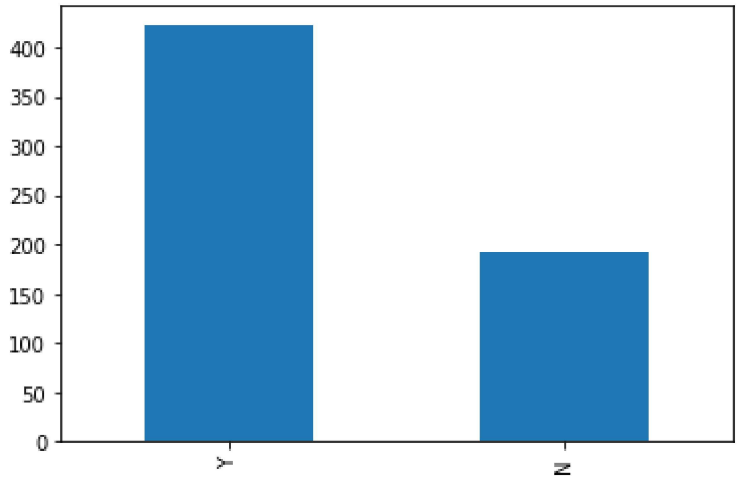
In [21]:
```
# NOrmalize to set to True to print propotion instead of number
```

In [23]:
```
train['Loan_Status'].value_counts(normalize = True)
```

Out[23]:
```
Y    0.687296
N    0.312704
Name: Loan_Status, dtype: float64
```

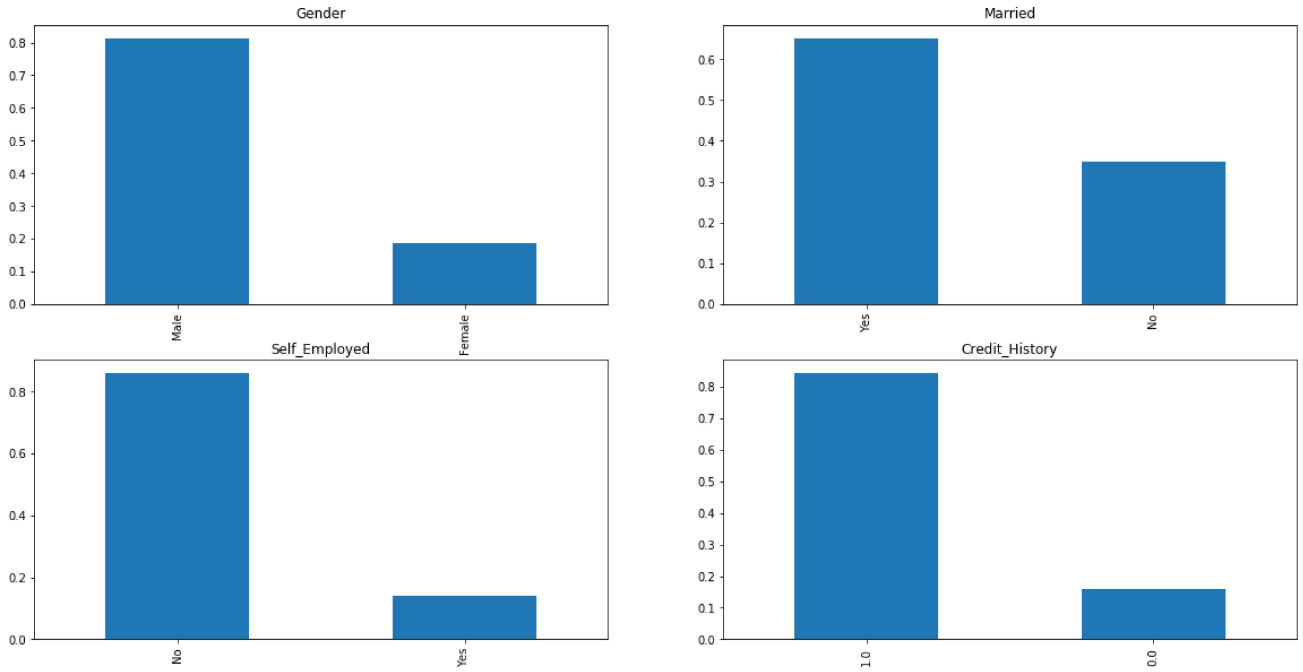
In [26]:
```
train['Loan_Status'].value_counts().plot.bar()
```

Out[26]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xef162a3e48>
```



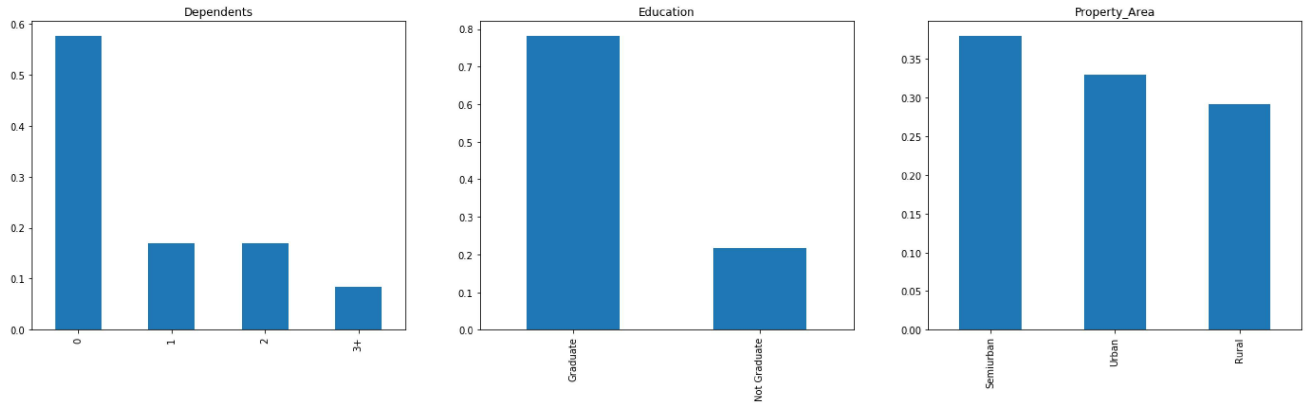
In [27]:
```
## Independent Variable(Categorical)
```

In [42]:
```
plt.figure(1)
plt.subplot(221)
train['Gender'].value_counts(normalize = True).plot.bar(figsize = (20,10), title = 'Gender')
plt.subplot(222)
train['Married'].value_counts( normalize = True).plot.bar(title = 'Married')
plt.subplot(223)
train['Self_Employed'].value_counts( normalize = True).plot.bar(title = 'Self_Employed')
plt.subplot(224)
train['Credit_History'].value_counts( normalize = True). plot.bar(title = 'Credit_History')
plt.show()
```



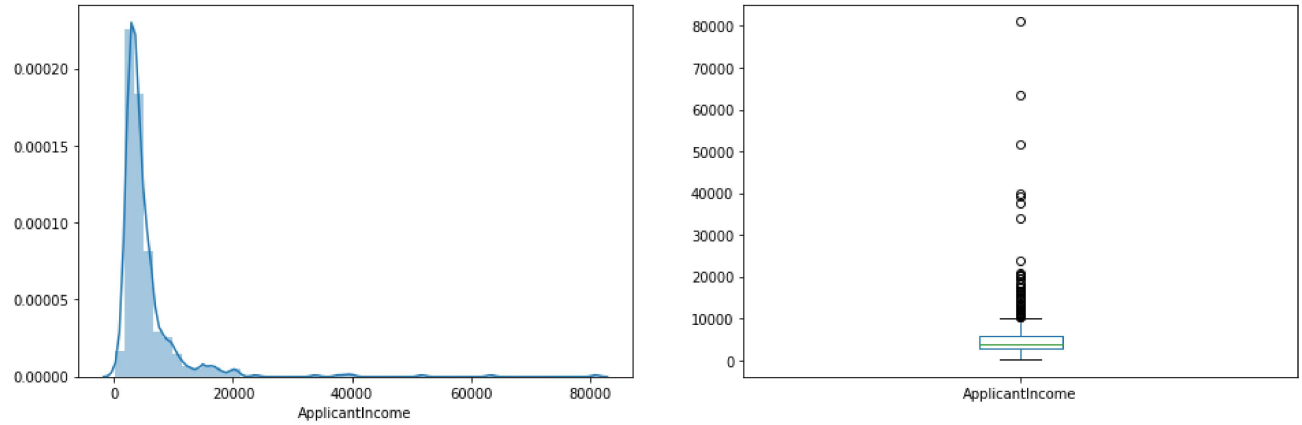

In [43]:
```
## Independent Variable(Ordinal)
```

In [44]:
```
plt.figure(1)
plt.subplot(131)
train['Dependents'].value_counts(normalize = True).plot.bar( figsize = (24,6), title
= 'Dependents')
plt.subplot(132)
train['Education'].value_counts( normalize = True).plot.bar( title = 'Education')
plt.subplot(133)
train['Property_Area'].value_counts( normalize = True).plot.bar( title = 'Property_Ar
ea')
plt.show()
```




In [45]:
```
## Independent Variable(Numerical)
```

In [47]:
```
plt.figure(1)
plt.subplot(121)
sns.distplot(train['ApplicantIncome']);
plt.subplot(122)
train['ApplicantIncome'].plot.box(figsize = (16,5))
plt.show()
```

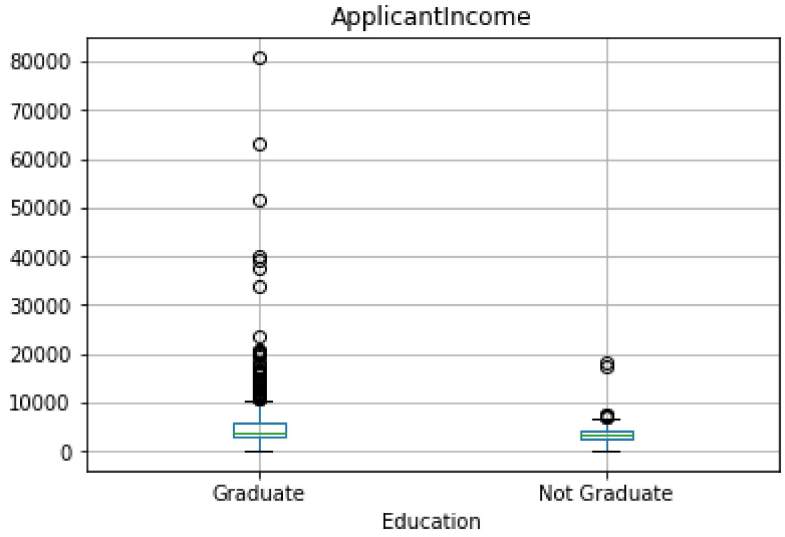



In [48]:
```
## Income is left skewed, not normal, boxplot show presence of outliers
```

In [49]:
```
## Education
```

In [51]:
```
train.boxplot( column = 'ApplicantIncome', by ='Education')
plt.suptitle("")
```

Out[51]:
```
Text(0.5, 0.98, '')
```

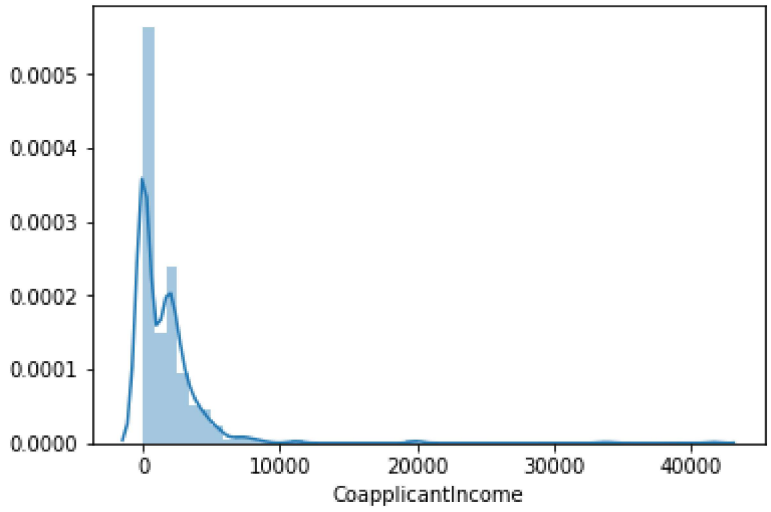


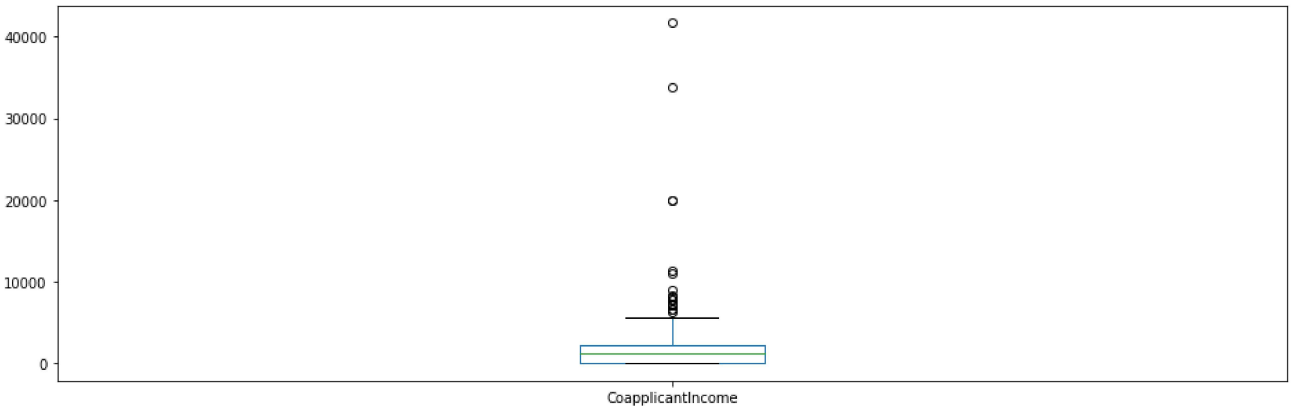

In [52]:
```
## Higher no. of graduates with higher income
```

In [3]:
```
## Co-applicant income distribution
```

In [9]:
```
plt.figure(1)
plt.figure(121)
sns.distplot(train['CoapplicantIncome'])
plt.figure(122)
train['CoapplicantIncome'].plot.box(figsize = (16,5))
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

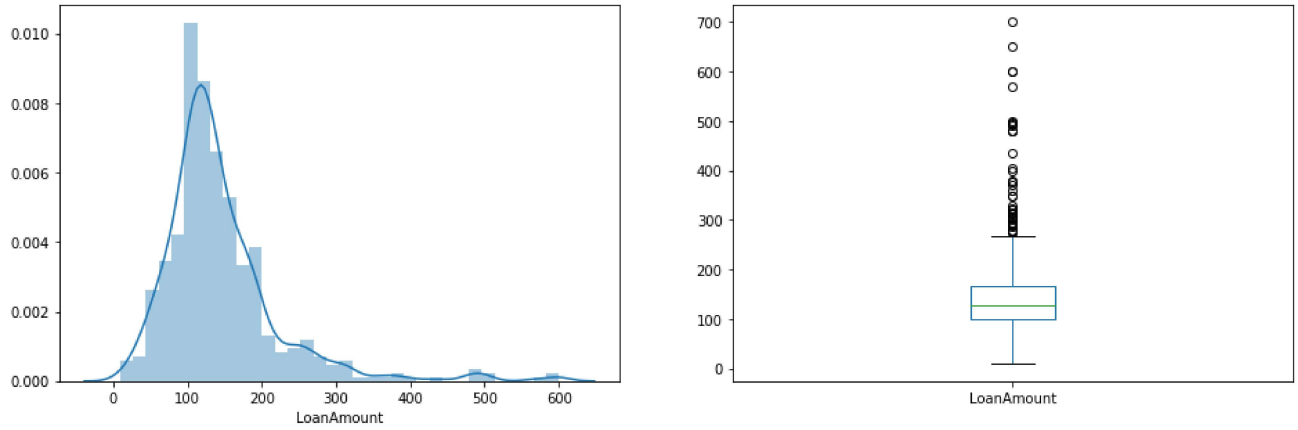



In [6]:
```
## -->Co applicant's income spreads from 0 to 5000 and consists lots of outliers
```

In [7]:
```
## Loan Amount
```

In [29]:
```
plt.figure(1)
plt.subplot(121)
df = train.dropna()
sns.distplot(df['LoanAmount'])
plt.subplot(122)
train['LoanAmount'].plot.box(figsize = (16,5))
plt.show()
```



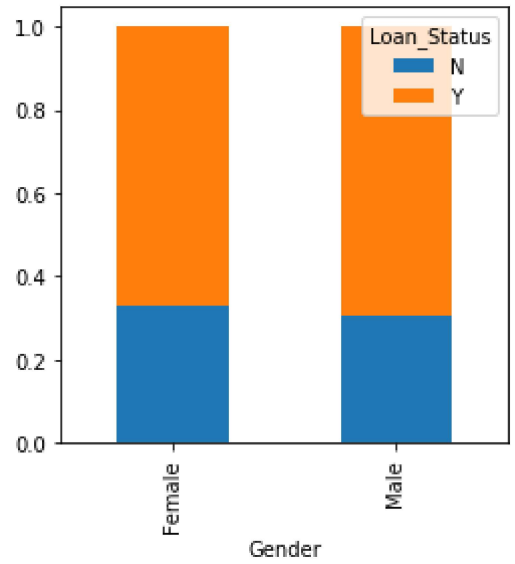

In [10]:
```
## --> Lots of uotliers but fairly distributed
```

In [11]:
```
# hypotheses using bivariate analysis

# Catrgorical Independent Variable vs Target Variable
```

In [12]:
```
Gender = pd.crosstab(train['Gender'], train['Loan_Status'])
Gender.div(Gender.sum(1).astype(float), axis=0).plot(kind = 'bar', stacked = True, fi
gsize = (4,4))
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ca7a73d68>
```

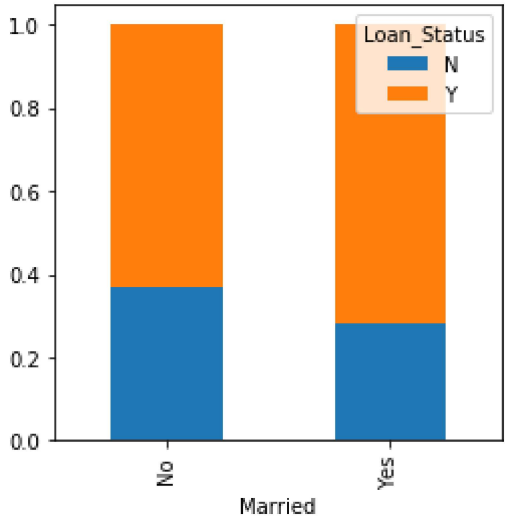


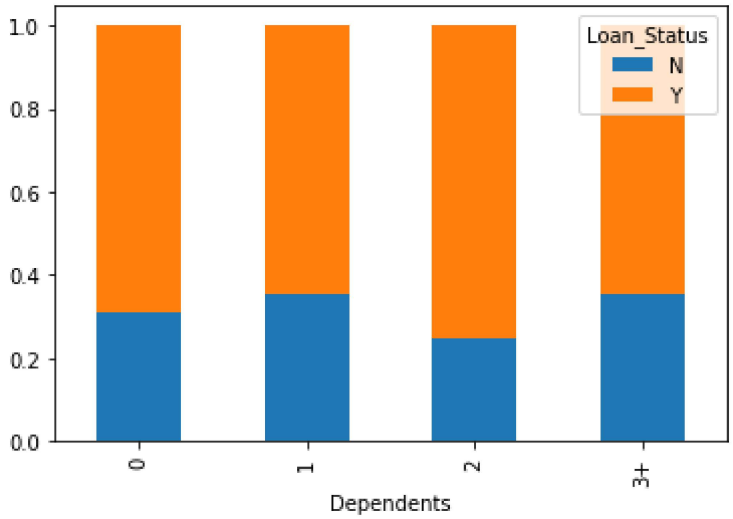

In [3]:
```
# MAle and Female has same propotion of approved/ unapproved
```

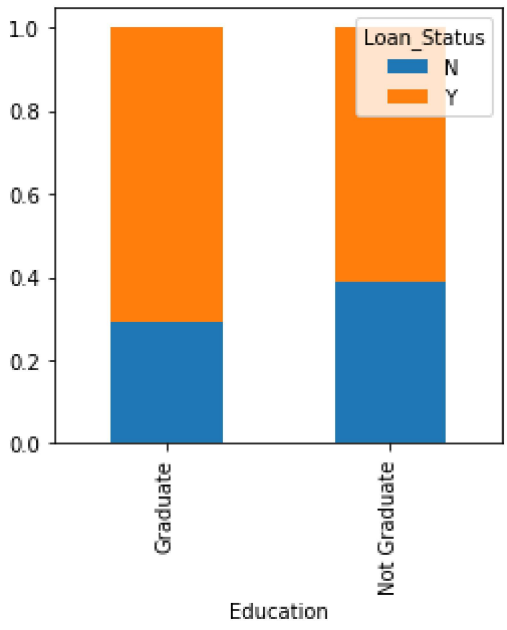
In [ ]:
```
#Categorical variable vs Target Variable
```

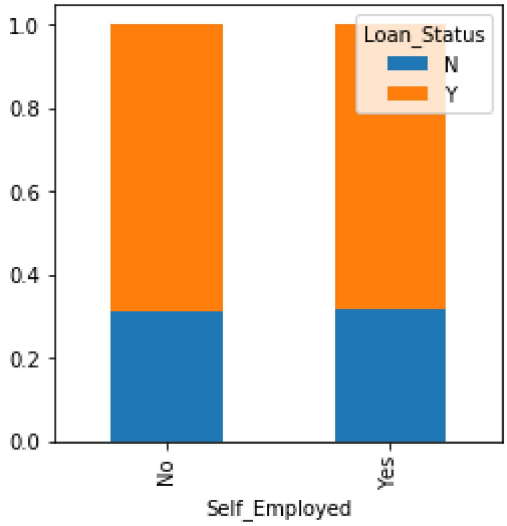
In [15]:

```
Married = pd.crosstab(train['Married'], train ['Loan_Status'])
Dependents = pd.crosstab(train['Dependents'], train['Loan_Status'])
Education = pd.crosstab(train['Education'], train['Loan_Status'])
Self_Employed = pd.crosstab(train['Self_Employed'], train['Loan_Status'])
Married.div(Married.sum(1).astype(float), axis = 0).plot(kind = 'bar', stacked = True
, figsize = (4,4))
plt.show()
Dependents.div(Dependents.sum(1).astype(float), axis = 0).plot (kind = 'bar', stacked
= True)
Education.div(Education.sum(1).astype(float), axis = 0).plot(kind = 'bar', stacked =
True, figsize = (4,4))
plt.show()
Self_Employed.div(Self_Employed.sum(1).astype(float), axis = 0).plot( kind = 'bar', s
tacked = True, figsize = (4,4))
plt.show()
```


Loan_Status
N
Y
1.0
0.8
0.6
0.4
0.2
0.0
No
Yes
Married


Loan_Status
N
Y
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3+
Dependents

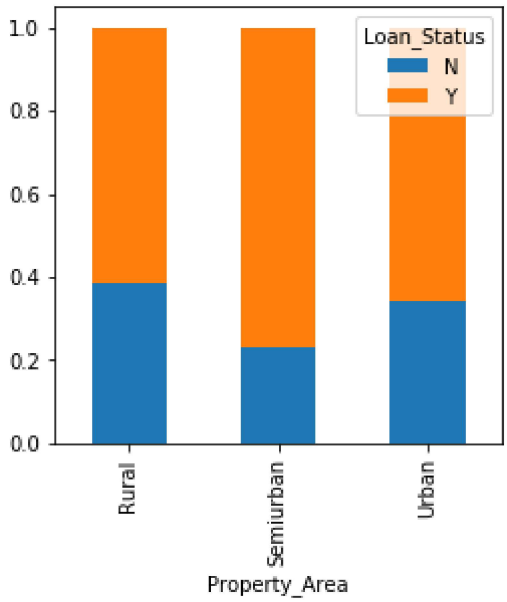

Loan_Status
N
Y
1.0
0.8
0.6
0.4
0.2
0.0
Graduate
Not Graduate
Education




In [16]:
```
# Propotion of Married applicants is higher for approved loan
# Distribution with dependents is similar
# No difference between Self_Employed and Loan_Status
```

In [17]:
```
# Categorical Independent variable and Loan_Status
```

In [22]:
```
Credit_History = pd.crosstab(train['Credit_History'],train['Loan_Status'])
Property_Area = pd.crosstab(train['Property_Area'], train['Loan_Status'])
Credit_History.div(Credit_History.sum(1).astype(float), axis = 0).plot( kind = 'bar',
stacked = True, figsize = (4,4))
plt.show()
Property_Area.div(Property_Area.sum(1).astype(float), axis = 0).plot( kind = 'bar', s
tacked = True, figsize = (4,4))
plt.show()
```

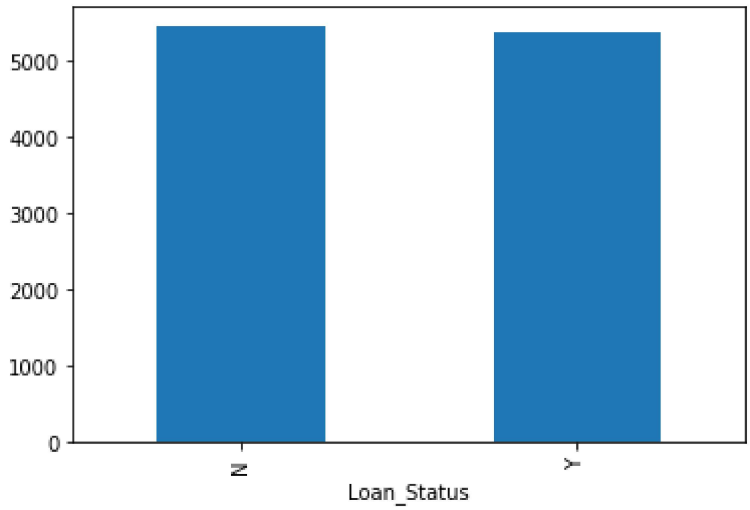




In [23]:
```
# People with Credit History are more lkikely to fet approval
# Propotion of Loan approval in Semi-urban area is higher
```

In [24]:
```
# Numerical Independent Variable vs Target Variable
# It will be determined the mean income of people for those who have their loan appro
ved/ disapproved
```

In [26]:
```
train.groupby('Loan_Status')['ApplicantIncome'].mean().plot.bar()
```

Out[26]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xd51be27940>
```

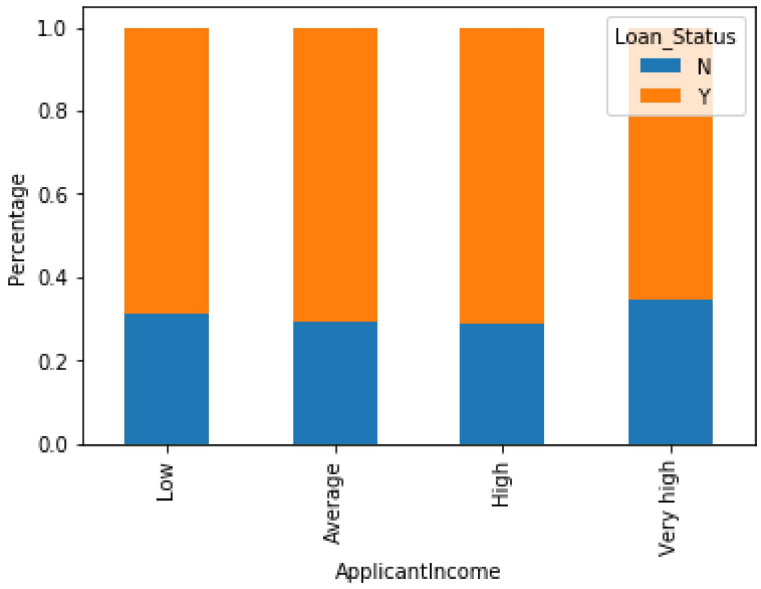



In [27]:
```
# there is no significant change in average income for approved and disapproved clien
ts
```

In [28]:
```
# Now testing by dividing into bis of income 0, 2500, 400, 600, 81000
```

In [33]:
```
bins = [0, 2500, 4000, 6000, 81000]
group = ['Low', 'Average', 'High', 'Very high']
train['Income_bin'] = pd.cut(df['ApplicantIncome'], bins, labels = group)

Income_bin = pd.crosstab(train['Income_bin'], train['Loan_Status'])
Income_bin.div(Income_bin.sum(1).astype(float), axis = 0).plot( kind ='bar', stacked
= True)
plt.xlabel('ApplicantIncome')
P = plt.ylabel('Percentage')
```




In [34]:
```
# Income Doesn't effect the Loan approval
```

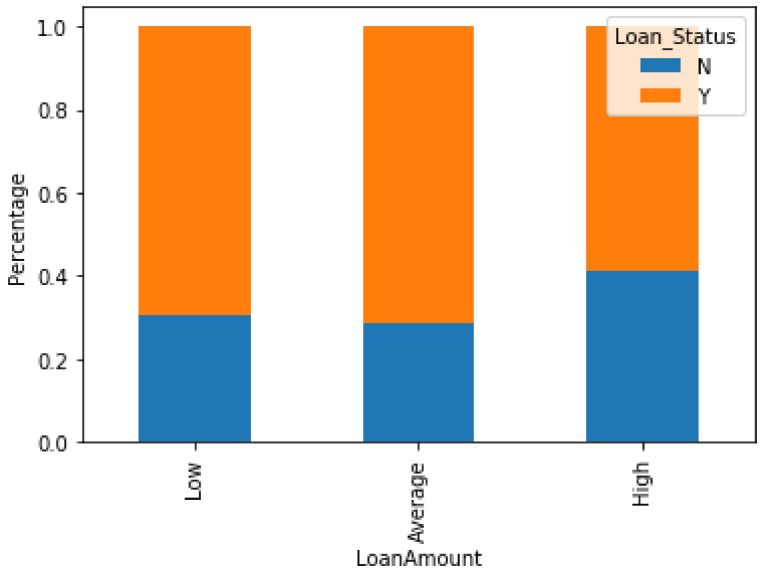
In [35]:
```
## Co-applicant plus combination of co-applicant and applicant income skipped from gi
ven case
# Combo with low incoem very less chance for approval
```

In [36]:
```
# Loan Amount Variable
```

In [39]:
```
bins = [0, 100, 200, 700]
group = ['Low', 'Average', 'High']
train['LoanAmount_bin'] = pd.cut(df['LoanAmount'], bins, labels = group)

LoanAmount_bin = pd.crosstab(train['LoanAmount_bin'], train['Loan_Status'])
LoanAmount_bin.div(LoanAmount_bin.sum(1).astype(float), axis = 0).plot(kind = 'bar',
stacked = True)
plt.xlabel('LoanAmount')
plt.ylabel('Percentage')
```

Out[39]:
```
Text(0, 0.5, 'Percentage')
```



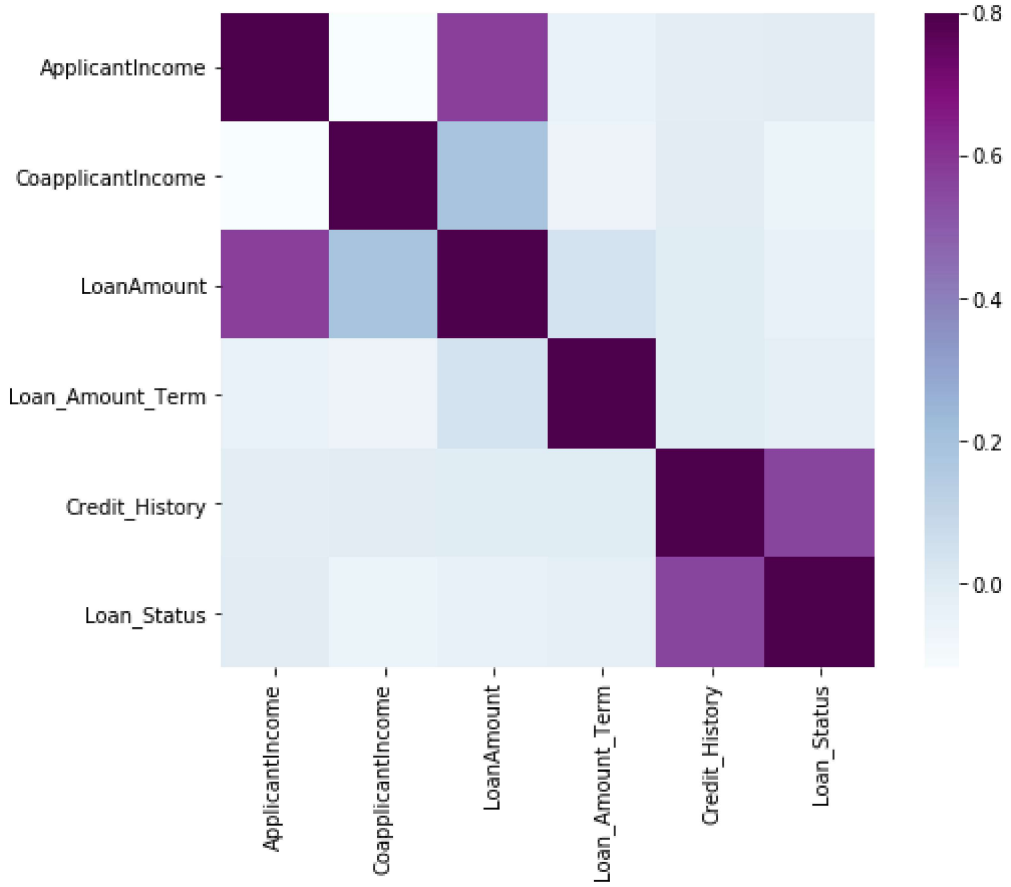

In [40]:
```
# Approval rate for lower loan amount is higher than for one with higher one
```

In [ ]:
```
# MAking all values numerical for co-relation showing
```

In [42]:
```
train['Dependents'].replace('3+', 3, inplace = True)
test['Dependents'].replace('3+', 3, inplace = True)
train['Loan_Status'].replace('N', 0, inplace = True)
train['Loan_Status'].replace('Y', 1, inplace = True)
```

In [50]:
```
matrix = train.corr()
f, ax = plt.subplots(figsize = (9,6))
sns.heatmap(matrix, vmax = .8, square = True, cmap ="BuPu")
```

Out[50]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xd51bf99978>
```




In [3]:
```
# most correlated are ApplicationIncome-LoanAmount and CreditHistory-LoanStatus
```

In [4]:
```
##Missing VAlues and Outliers Treatment
```

In [5]:
```
# Missing Value imputation
```

In [6]:
```
train.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 [9]:
```
# Replacing missing values with 0 for the categorical values
train['Gender'].fillna(train['Gender'].mode()[0],inplace = True)
train['Married'].fillna(train['Married'].mode()[0], inplace = True)
train['Dependents'].fillna(train['Dependents'].mode()[0], inplace = True)
train['Self_Employed'].fillna(train['Self_Employed'].mode()[0], inplace = True)
train['Credit_History'].fillna(train['Credit_History'].mode()[0], inplace = True)
```

In [14]:
```
## For missing Values in Loan Term
train['Loan_Amount_Term'].value_counts(dropna = False).sort_index()
```

Out[14]:
```
12.0       1
36.0       2
60.0       2
84.0       4
120.0      3
180.0     44
240.0      4
300.0     13
360.0    526
480.0     15
Name: Loan_Amount_Term, dtype: int64
```

In [12]:
```
## Replacing the missing value with mode of the variable
train['Loan_Amount_Term'].fillna(train['Loan_Amount_Term'].mode()[0], inplace = True)
```

In [16]:
```
## Replacing the msssing values with Median of teh column
train['LoanAmount'].fillna(train['LoanAmount'].median(), inplace = True)
```

In [17]:
```
train.isnull().sum()
```

Out[17]:
```
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
```

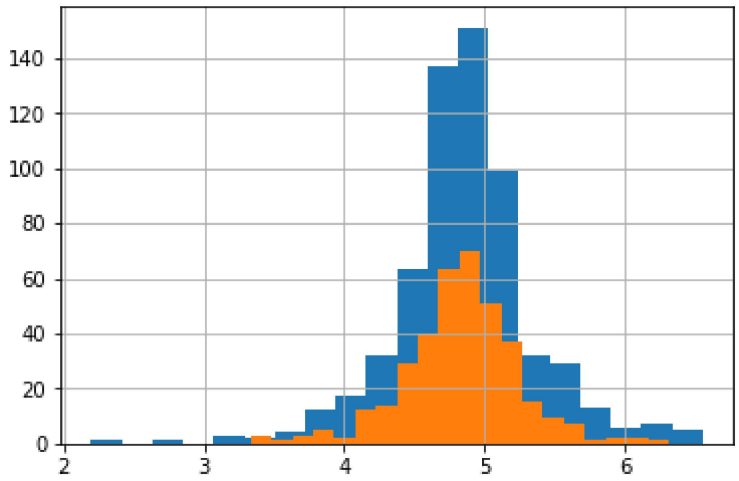
In [18]:
```
## Filling all the missing values in test dataset
test['Gender'].fillna(train['Gender'].mode()[0], inplace = True)
test['Dependents'].fillna(train['Dependents'].mode()[0], inplace = True)
test['Self_Employed'].fillna(train['Self_Employed'].mode()[0], inplace = True)
test['Credit_History'].fillna(train['Credit_History'].mode()[0], inplace = True)
test['Loan_Amount_Term'].fillna(train['Loan_Amount_Term'].mode()[0], inplace = True)
test['LoanAmount'].fillna(train['LoanAmount'].median(), inplace = True)
```

In [19]:
```
test.isnull().sum()
```

Out[19]:
```
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
dtype: int64
```

In [31]:
```
## Outlier Treatment
## Measure outliers with Skewness & remove by logrythmetic transformation
train['LoanAmount_log'] = np.log(train['LoanAmount'])
train['LoanAmount_log'].hist(bins = 20)
#fig1.show()
# fig2 = plt.figure(2)
# test['LoanAmount_log'] = np.log(test['LoanAmount'])
# # f2=fig.add_subplot(122)
# fig.
test['LoanAmount_log'].hist(bins = 20)
```

Out[31]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x467730f4a8>
```




In [ ]:
```
## The Distribution looks more closer to normal
```