In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
file_path="C:\\Users\\surya\\Downloads\\loanpr.csv"
loandf=pd.read_csv(file_path)
loandf
```

Out[5]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
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	
...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

614 rows × 13 columns

In [6]:
```
loandf.shape
```

Out[6]:
```
(614, 13)
```

In [4]:
```
loandf.size
```

Out[4]:
```
7982
```

In [5]:
```
loandf.columns
```

Out[5]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmoun
t',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Statu
s'],
      dtype='object')
```

In [6]:
```
loandf.dtypes
```

Out[6]:
```
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 [7]:
```
loandf.head()
```

Out[7]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coa
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 [8]:
```
loandf.tail()
```

Out[8]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

In [9]:
```
loandf.take([10,20])
```

Out[9]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Cc
10	LP001024	Male	Yes	2	Graduate	No	3200	
20	LP001043	Male	Yes	0	Not Graduate	No	7660	

In [10]:
```
loandf.iloc[10:20]
```

Out[10]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Co
10	LP001024	Male	Yes	2	Graduate	No	3200	
11	LP001027	Male	Yes	2	Graduate	NaN	2500	
12	LP001028	Male	Yes	2	Graduate	No	3073	
13	LP001029	Male	No	0	Graduate	No	1853	
14	LP001030	Male	Yes	2	Graduate	No	1299	
15	LP001032	Male	No	0	Graduate	No	4950	
16	LP001034	Male	No	1	Not Graduate	No	3596	
17	LP001036	Female	No	0	Graduate	No	3510	
18	LP001038	Male	Yes	0	Not Graduate	No	4887	
19	LP001041	Male	Yes	0	Graduate	NaN	2600	

In [11]:
```
loandf.loc[200],['Education','loan_status']
```

Out[11]:
```
(Loan_ID                LP001674
 Gender                     Male
 Married                     Yes
 Dependents                    1
 Education          Not Graduate
 Self_Employed                No
 ApplicantIncome            2600
 CoapplicantIncome        2500.0
 LoanAmount                 90.0
 Loan_Amount_Term          360.0
 Credit_History              1.0
 Property_Area          Semiurban
 Loan_Status                    Y
 Name: 200, dtype: object,
 ['Education', 'loan_status'])
```

In [12]:
```
loandf.isnull().sum().sort_values()
```

Out[12]:
```
Loan_ID              0
Education            0
ApplicantIncome      0
CoapplicantIncome    0
Property_Area        0
Loan_Status          0
Married              3
Gender              13
Loan_Amount_Term    14
Dependents          15
LoanAmount          22
Self_Employed       32
Credit_History      50
dtype: int64
```

In [13]:
```
len(loandf)
print('total number of rows is',len(loandf))
```

```
total number of rows is 614
```

categorical data analysis

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

Out[14]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmoun
t',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Statu
s'],
      dtype='object')
```

In [15]:
```
loandf.dtypes
```

Out[15]:
```
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 [7]:
```
cat_col=[]
num_col=[]
for column_name,data_type in loandf.dtypes.items():
    if data_type=='object':
        cat_col.append(column_name)
    else:
        num_col.append(column_name)
```

In [8]:
```
cat_col
```

Out[8]:
```
['Loan_ID',
 'Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'Property_Area',
 'Loan_Status']
```

In [9]:
```
num_col
```

Out[9]:
```
['ApplicantIncome',
 'CoapplicantIncome',
 'LoanAmount',
 'Loan_Amount_Term',
 'Credit_History']
```

In [19]:
```
for column in cat_col:
    print(column)
    print(loandf[column].unique())
    print()
```

```
['Male' 'Female' nan]

Married
['No' 'Yes' nan]

Dependents
['0' '1' '2' '3+' nan]

Education
['Graduate' 'Not Graduate']

Self_Employed
['No' 'Yes' nan]

Property_Area
['Urban' 'Rural' 'Semiurban']

Loan_Status
['Y' 'N']
```

In [10]:
```
for column in cat_col:
    print(column)
    print(loandf[column].nunique())
    print()
```

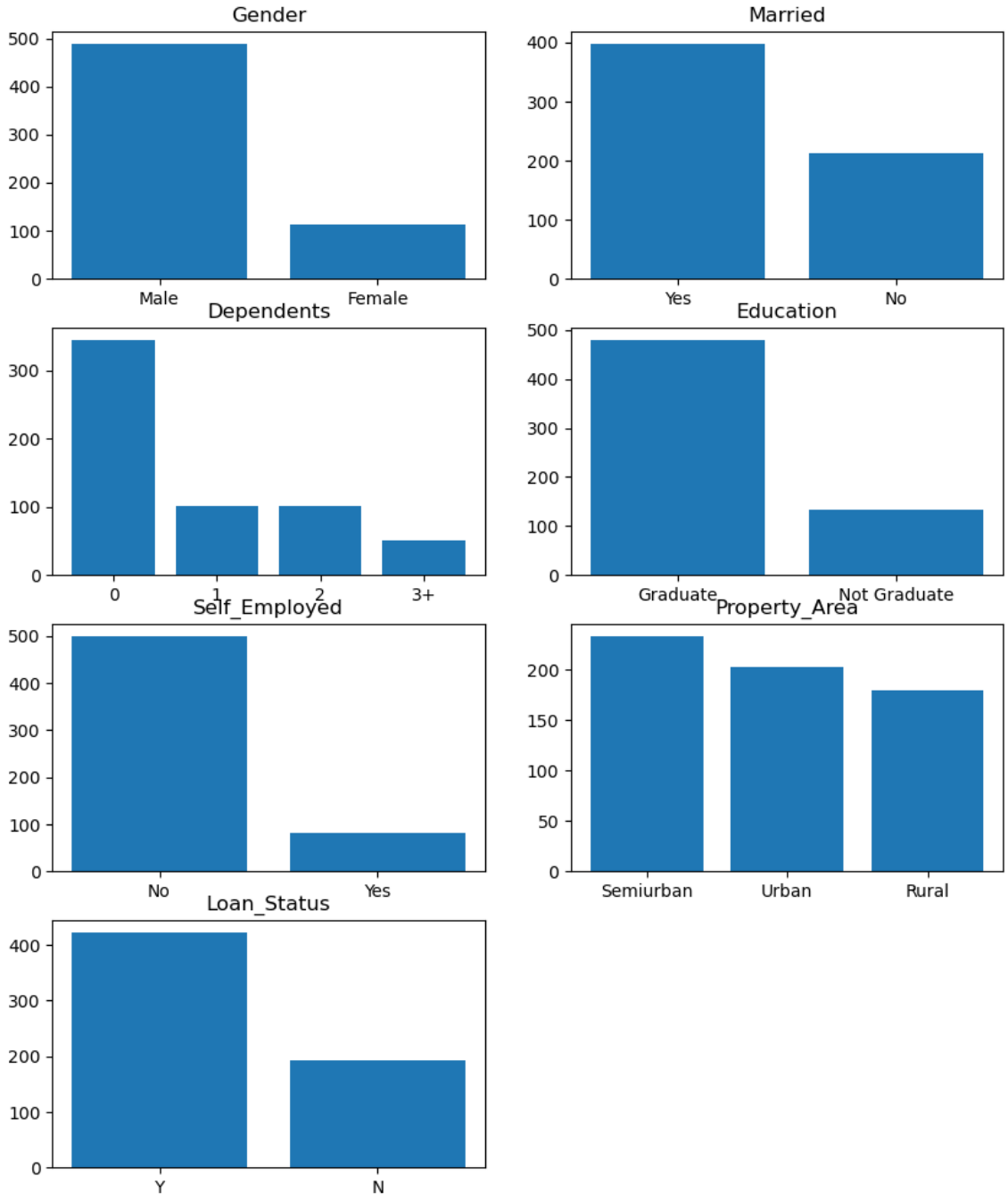
```
Loan_ID
614

Gender
2

Married
2

Dependents
4

Education
2

Self_Employed
2

Property_Area
3

Loan_Status
2
```

In [11]:
```
loandf.value_counts()
```

Out[11]:
```
Loan_ID   Gender  Married  Dependents  Education     Self_Employed  Applic
antIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History
Property_Area  Loan_Status
LP001003  Male    Yes      1           Graduate      No             4583
1508.0             128.0       360.0             1.0             Rural
N              1
LP001005  Male    Yes      0           Graduate      Yes            3000
0.0                66.0        360.0             1.0             Urban
Y              1
LP002347  Male    Yes      0           Graduate      No             3246
1417.0             138.0       360.0             1.0             Semiurban
Y              1
LP002345  Male    Yes      0           Graduate      No             1025
2773.0             112.0       360.0             1.0             Rural
Y              1
LP002342  Male    Yes      2           Graduate      Yes            1600
20000.0            239.0       360.0             1.0             Urban
N              1

..
LP001674  Male    Yes      1           Not Graduate  No             2600
2500.0             90.0        360.0             1.0             Semiurban
Y              1
LP001673  Male    No       0           Graduate      Yes            11000
0.0                83.0        360.0             1.0             Urban
N              1
LP001666  Male    No       0           Graduate      No             8333
3750.0             187.0       360.0             1.0             Rural
Y              1
LP001665  Male    Yes      1           Graduate      No             3125
2583.0             170.0       360.0             1.0             Semiurban
N              1
LP002990  Female  No       0           Graduate      Yes            4583
0.0                133.0       360.0             0.0             Semiurban
N              1
Name: count, Length: 480, dtype: int64
```

categorical column barplots

In [20]:
```
plt.figure(figsize=(10,12))
for i,column in enumerate(cat_col[1:]):
    loandf[column].value_counts()
    a=loandf[column].value_counts().keys()
    b=loandf[column].value_counts().values
    s=pd.DataFrame(zip(a,b),columns=[column,'count'])

    plt.subplot(4,2,i+1)

    plt.title(column)
    plt.bar(column,'count',data=s)
plt.show()
```

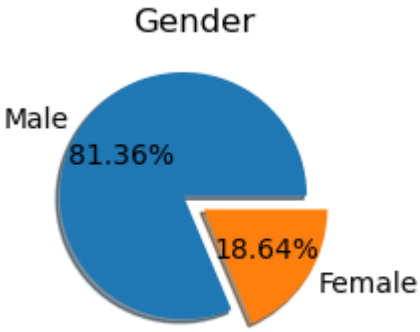


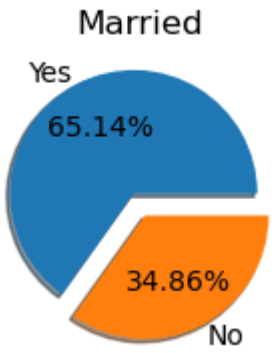
In [19]:
```
len(cat_col)
```

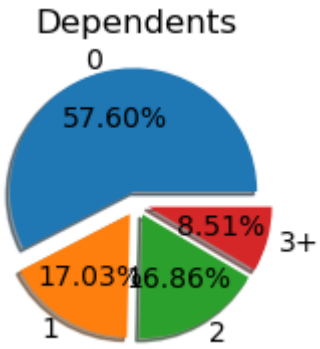
Out[19]: 8

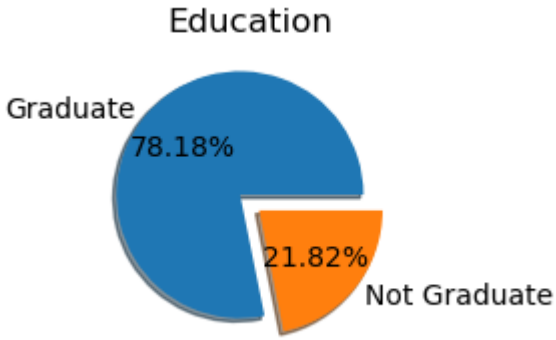
pie chart

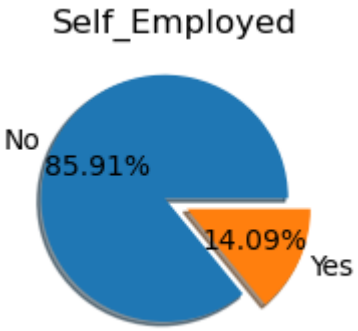
In [45]:
```
for i,column in enumerate(cat_col[1:]):
    #plt.subplot(4,2,i+1)
    loandf[column].value_counts()
    a=loandf[column].value_counts(normalize=True).keys()
    b=loandf[column].value_counts(normalize=True).values
    s=pd.DataFrame(zip(a,b),columns=[column,'count'])
    n=loandf[column].nunique()
    l=[0.1 for i in range(n)]
    plt.figure(figsize=(2,2))
    plt.title(column)
    plt.pie(x=b,labels=a,autopct="%0.2f%%",shadow=True,radius=1,explode=l)
    plt.show()
```

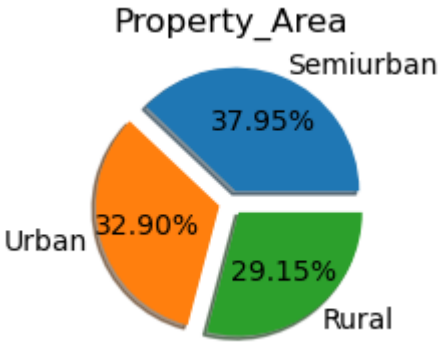


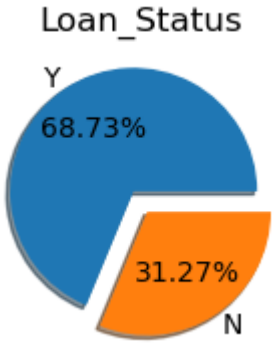





Education
Graduate
78.18%
21.82%
Not Graduate


Self_Employed
No
85.91%
14.09%
Yes


Property_Area
Semiurban
37.95%
Urban
32.90%
29.15%
Rural


Loan_Status
Y
68.73%
31.27%
N

neumerical data analysis

In [28]:
```
cat_col=[]
num_col=[]
for column_name,data_type in loandf.dtypes.items():
    if data_type=='object':
        cat_col.append(column_name)
    else:
        num_col.append(column_name)
```

In [29]:
```
num_col
```

Out[29]:
```
['ApplicantIncome',
 'CoapplicantIncome',
 'LoanAmount',
 'Loan_Amount_Term',
 'Credit_History']
```

In [30]:
```
loandf.describe()
```

Out[30]:

	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 [31]:
```
loandf.describe(include='object')
```

Out[31]:

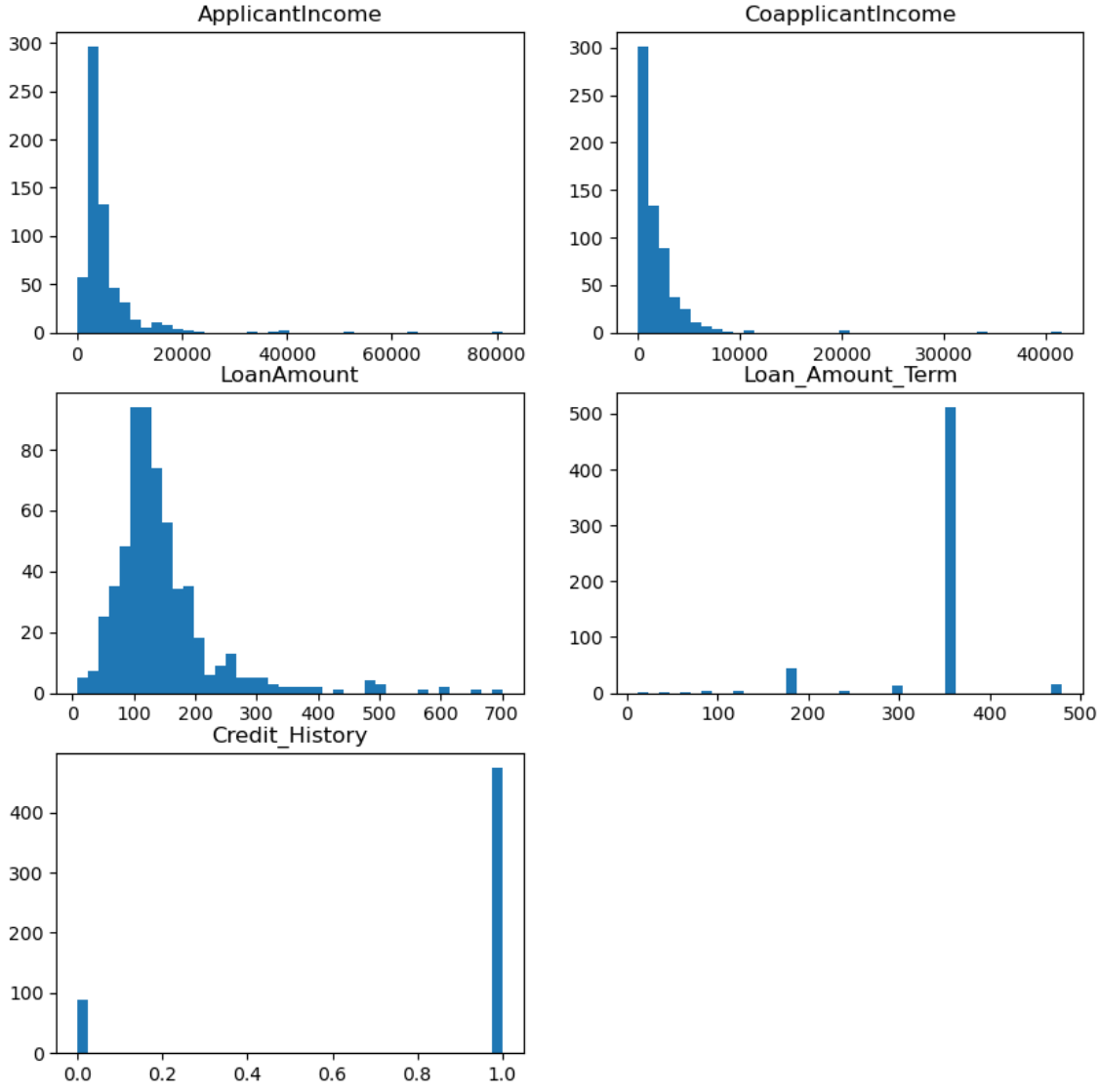
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Property_Area
count	614	601	611	599	614	582	614
unique	614	2	2	4	2	2	3
top	LP001002	Male	Yes	0	Graduate	No	Semiurban
freq	1	489	398	345	480	500	233

In [32]:
```
for column in num_col:
    dict1={}
    ai_count=round(loandf[column].count(),2)
    ai_min=round(loandf[column].min(),2)
    ai_max=round(loandf[column].max(),2)
    ai_mean=round(loandf[column].mean(),2)
    ai_median=round(loandf[column].median(),2)
    ai_std=round(loandf[column].std(),2)
    list1=[ai_count,ai_min,ai_max,ai_mean,ai_median,ai_std]
    dict1[column]=list1
    dict1
    print(pd.DataFrame(dict1,index=['count','min','max','mean','median','st
    print()
```

```
        ApplicantIncome
count            614.00
min              150.00
max            81000.00
mean            5403.46
median          3812.50
std             6109.04

        CoapplicantIncome
count              614.00
min                  0.00
max              41667.00
mean              1621.25
median            1188.50
std               2926.25

        LoanAmount
count       592.00
min           9.00
max         700.00
mean        146.41
median      128.00
std          85.59

        Loan_Amount_Term
count              600.00
min                 12.00
max                480.00
mean               342.00
median             360.00
std                 65.12

        Credit_History
count           564.00
min               0.00
max               1.00
mean              0.84
median            1.00
std               0.36
```

nuemerical column histplot

In [36]:
```
plt.figure(figsize=(10,10))
for i, column in enumerate(num_col):
    plt.subplot(3,2,i+1)
    plt.title(column)
    plt.hist(loandf[ column],bins=40)
plt.show()
```



In [ ]:
```
```

In [37]:
```
ai_mean,ai_std
```

Out[37]: (0.84, 0.36)

In [38]:
```
lamean=loandf['LoanAmount'].mean()
lastd=loandf['LoanAmount'].std()
```

In [39]:
```
lamean,lastd
```

Out[39]:
```
(146.41216216216216, 85.58732523570545)
```

In [40]:
```
val_minus1=round(lamean-1*lastd,2)
val_plus1=round(lamean+1*lastd,2)
val_minus2=round(lamean-2*lastd,2)
val_plus2=round(lamean+2*lastd,2)
val_minus3=round(lamean-3*lastd,2)
val_plus3=round(lamean+3*lastd,2)
print(val_minus1,val_plus1,val_minus2,val_plus2,val_minus3,val_plus3)
print('')
```

```
60.82 232.0 -24.76 317.59 -110.35 403.17
```

In [41]:
```
con1=loandf['LoanAmount']>val_minus1
con2=loandf['LoanAmount']<val_plus1
len(loandf[con1&con2])
len(loandf[con1&con2])/len(loandf)
print('80 % of loan amount falls between 1std devition range,\nit is higher
```

```
80 % of loan amount falls between 1std devition range,
it is higher than 68% so it is not following emperical rule,
this data not not follows emperical rule
```

In [42]:
```
con1=loandf['LoanAmount']>val_minus2
con2=loandf['LoanAmount']<val_plus2
len(loandf[con1&con2])
len(loandf[con1&con2])/len(loandf)
```

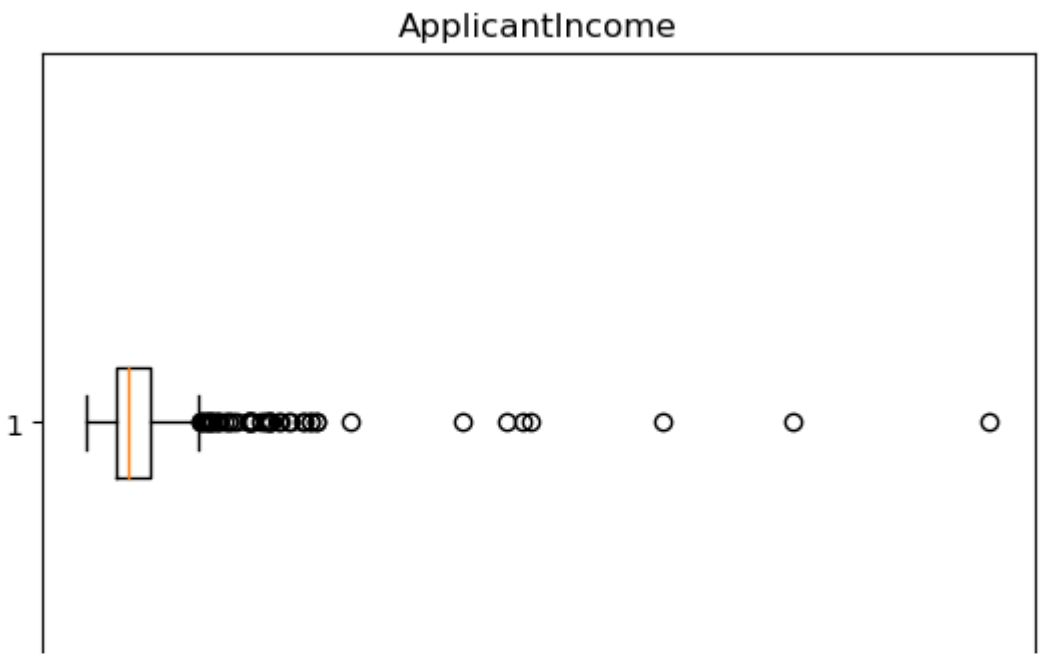
Out[42]:
```
0.9250814332247557
```

In [43]:
```
con1=loandf['LoanAmount']>val_minus3
con2=loandf['LoanAmount']<val_plus3
len(loandf[con1&con2])
len(loandf[con1&con2])/len(loandf)
```

Out[43]:
```
0.9413680781758957
```

box plot

In [44]:
```
for column in num_col:
    plt.boxplot(loandf[column],vert=False)
    plt.title(column)
    plt.show()
```



In [46]:
```
for column in num_col:
    q1=np.quantile(loandf[column],0.25)
    q2=np.quantile(loandf[column],0.50)
    q3=np.quantile(loandf[column],0.75)
        #Step-2:Calculate IQR=(Q3-Q1)
    IQR=q3-q1
        #Step-3: UB=Q3+1.5*IQR
    ub=q3+1.5*IQR
        #Step-4: LB=Q1-1.5*IQR
    lb=q1-1.5*IQR
        #Step-5: con1= col>UB
        #Step-6: con2= col<LB
    con1=loandf[column]>ub
    con2=loandf[column]<lb
        #step-7 and step-8
    outliers=loandf[column][con1|con2]
        # series into array of values by applying a .values
    outliers_data=outliers.values
    print(column,len(outliers_data))
```

```
ApplicantIncome 50
CoapplicantIncome 18
LoanAmount 0
Loan_Amount_Term 0
Credit_History 0
```

case 1

Removal of outliers

we have 50 outliers in Applicantincome column

that means we need to remove 50 rows from entire dataframe

In [46]:
```
18/614*100
```

Out[46]:
```
2.9315960912052117
```

In [47]:
```
len(loandf)
```

Out[47]:
```
614
```

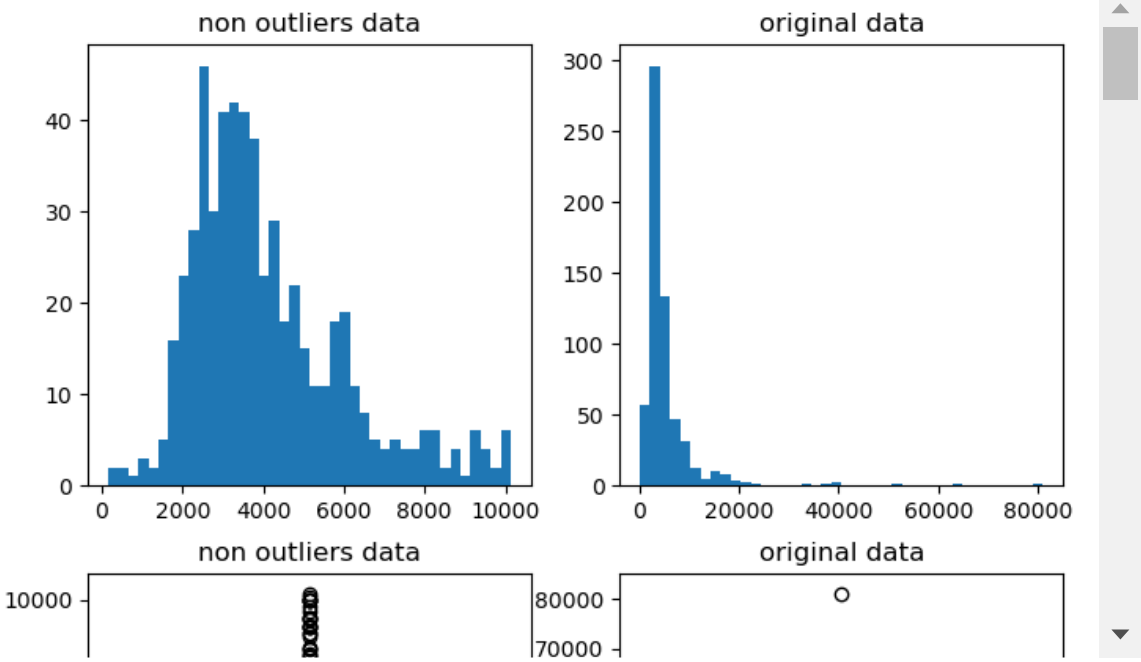
In [48]:
```
q1=np.quantile(loandf['ApplicantIncome'],0.25)
q2=np.quantile(loandf['ApplicantIncome'],0.50)
q3=np.quantile(loandf['ApplicantIncome'],0.75)
IQR=q3-q1
ub=q3+1.5*IQR
lb=q1-1.5*IQR
con1=loandf['ApplicantIncome']<ub
con2=loandf['ApplicantIncome']>lb
################################################
non_outliers_df=loandf[con1&con2]
################################################
non_outliers_df
```

Out[48]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
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	
...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

564 rows × 13 columns

In [49]:
```
for column in num_col:
    plt.figure(figsize=(8,8))
    plt.subplot(2,2,1)
    plt.title("non outliers data")
    plt.hist(non_outliers_df[column],bins=40)
    plt.subplot(2,2,2)
    plt.title("original data")
    plt.hist(loandf[column],bins=40)
    plt.subplot(2,2,3)
    plt.title("non outliers data")
    plt.boxplot(non_outliers_df[column])
    plt.subplot(2,2,4)
    plt.title("original data")
    plt.boxplot(loandf[column])
    plt.show()
```



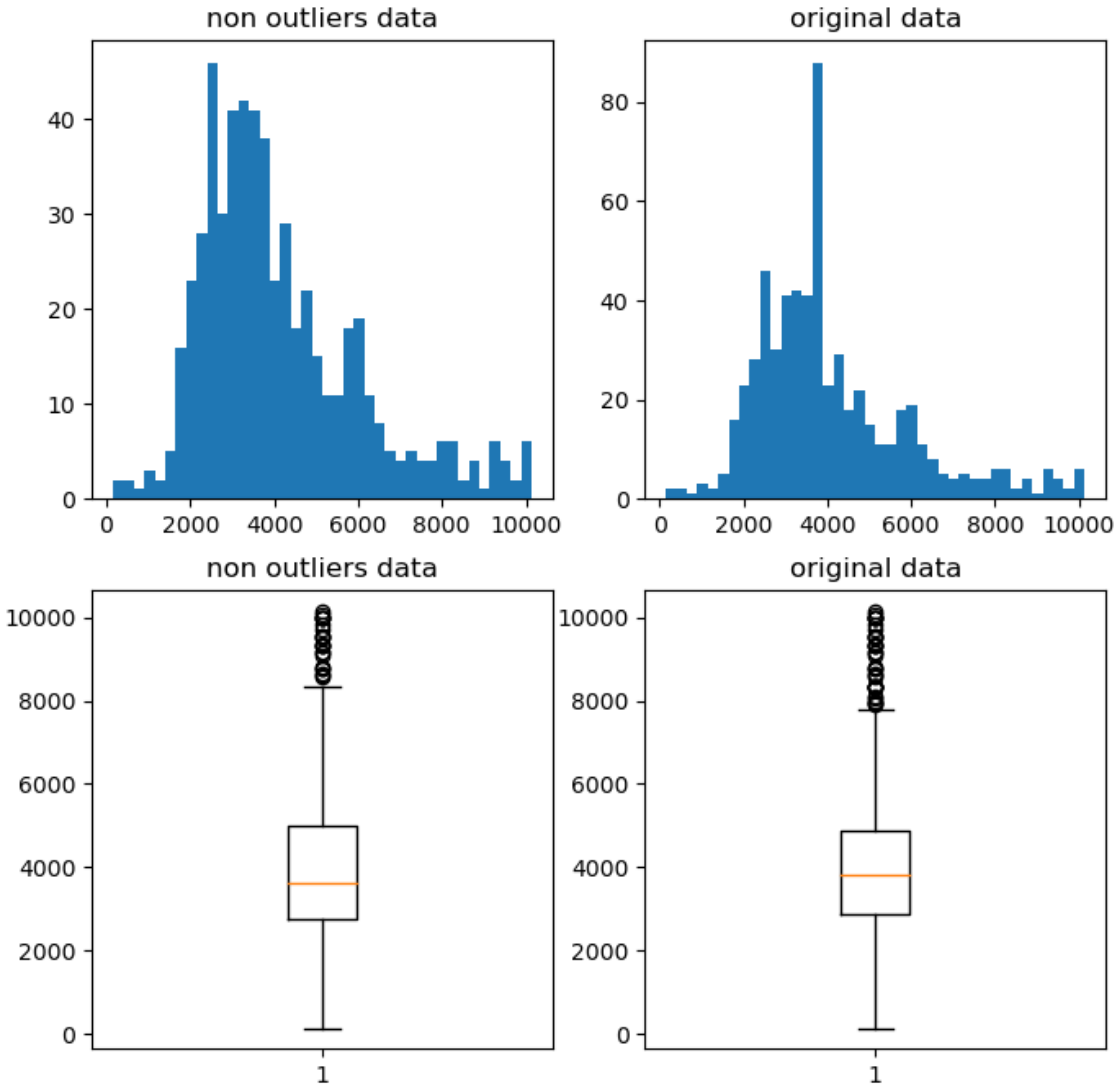
case 2

In [50]:
```
ub,lb
```

Out[50]:
```
(10171.25, -1498.75)
```

In [51]:
```
for column in num_col:
    q1=np.quantile(loandf[column],0.25)
    q2=np.quantile(loandf[column],0.50)
    q3=np.quantile(loandf[column],0.75)
    IQR=q3-q1
    ub=q3+1.5*IQR
    lb=q1-1.5*IQR
    con1=loandf[column]>ub
    con2=loandf[column]<lb
    ################################################
    loandf.loc[con1|con2,column]=round(loandf[column].median(),2)


    ################################################
```

In [52]:
```
plt.figure(figsize=(8,8))
plt.subplot(2,2,1)
plt.title("non outliers data")
plt.hist(non_outliers_df['ApplicantIncome'],bins=40)
plt.subplot(2,2,2)
plt.title("original data")
plt.hist(loandf['ApplicantIncome'],bins=40)
plt.subplot(2,2,3)
plt.title("non outliers data")
plt.boxplot(non_outliers_df['ApplicantIncome'])
plt.subplot(2,2,4)
plt.title("original data")
plt.boxplot(loandf['ApplicantIncome'])
plt.show()
```



bi varient and multivarient analysis

In [53]:
```
## two cat column analysis
```

In [54]:
```
loandf['Gender'].unique()
loandf['Gender'].value_counts().keys()
```

Out[54]:
```
Index(['Male', 'Female'], dtype='object', name='Gender')
```

In [35]:
```
loandf.isnull().sum()
```

Out[35]:
```
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 [ ]:

In [ ]:

In [55]:
```
labeles=loandf['Gender'].unique()
y_count=[]
n_count=[]
for i in labeles:
    con1=loandf['Gender']==i
    con2=loandf['Married']=='Yes'
    con3=loandf['Married']=='No'
    y_count.append(len(loandf[con1&con2]))
    n_count.append(len(loandf[con1&con2]))


pd.DataFrame(zip(labeles,y_count,n_count),columns=['Gender','yes','no'])
```

Out[55]:

	Gender	yes	no
0	Male	357	357
1	Female	31	31
2	NaN	0	0

pd crosstab

In [56]:
```
col1=loandf['Gender']
col2=loandf['Married']
result5=pd.crosstab(col1,col2)
result5
```

Out[56]:

Married	No	Yes
Gender		
Female	80	31
Male	130	357

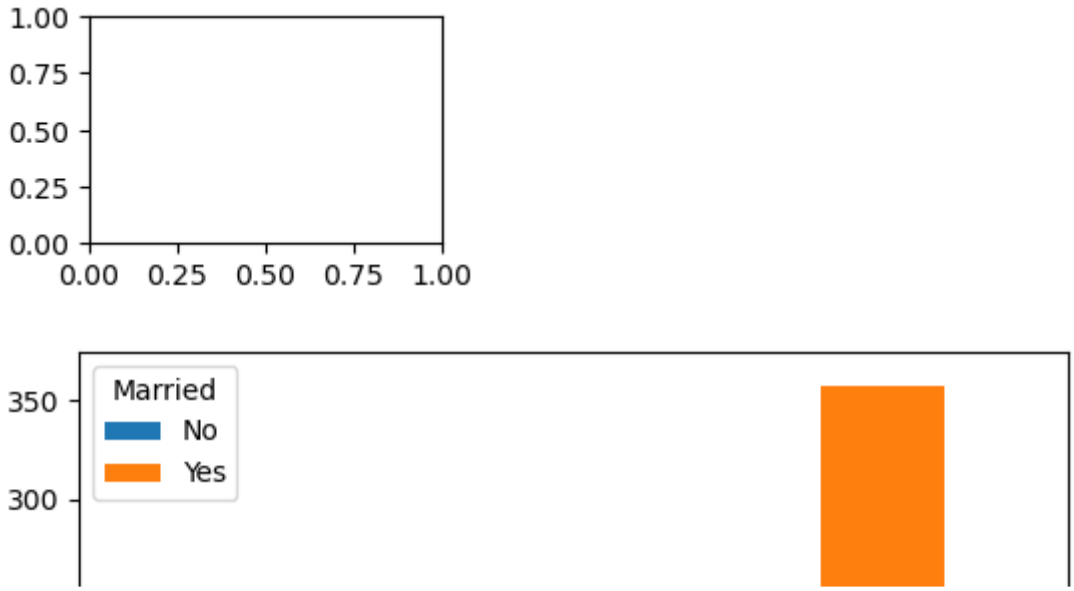
In [57]:
```
cat_col
```

Out[57]:
```
['Loan_ID',
 'Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'Property_Area',
 'Loan_Status']
```

categorical column barplot

In [57]:
```
#import matplotlib.pyplot as plt
#fig, ax = plt.subplots(4, 2)
#warnings.filterwarnings("ignore",category=UserWarning)
#n=len(cat_col)-2
#print(n)
#rows=n//2+n%2
#cols=2
#print(rows,cols)
plt.figure(figsize=(5,5))
col1=loandf['Gender']
for i, column in enumerate(cat_col[2:]):
    print(i,column)
    col2=loandf[column]
    plt.subplot(3,2,i+1)
    result2=pd.crosstab(col1,col2)
    result2.plot(kind='bar')
    plt.subplots_adjust(hspace=0.5)
    plt.show()
```

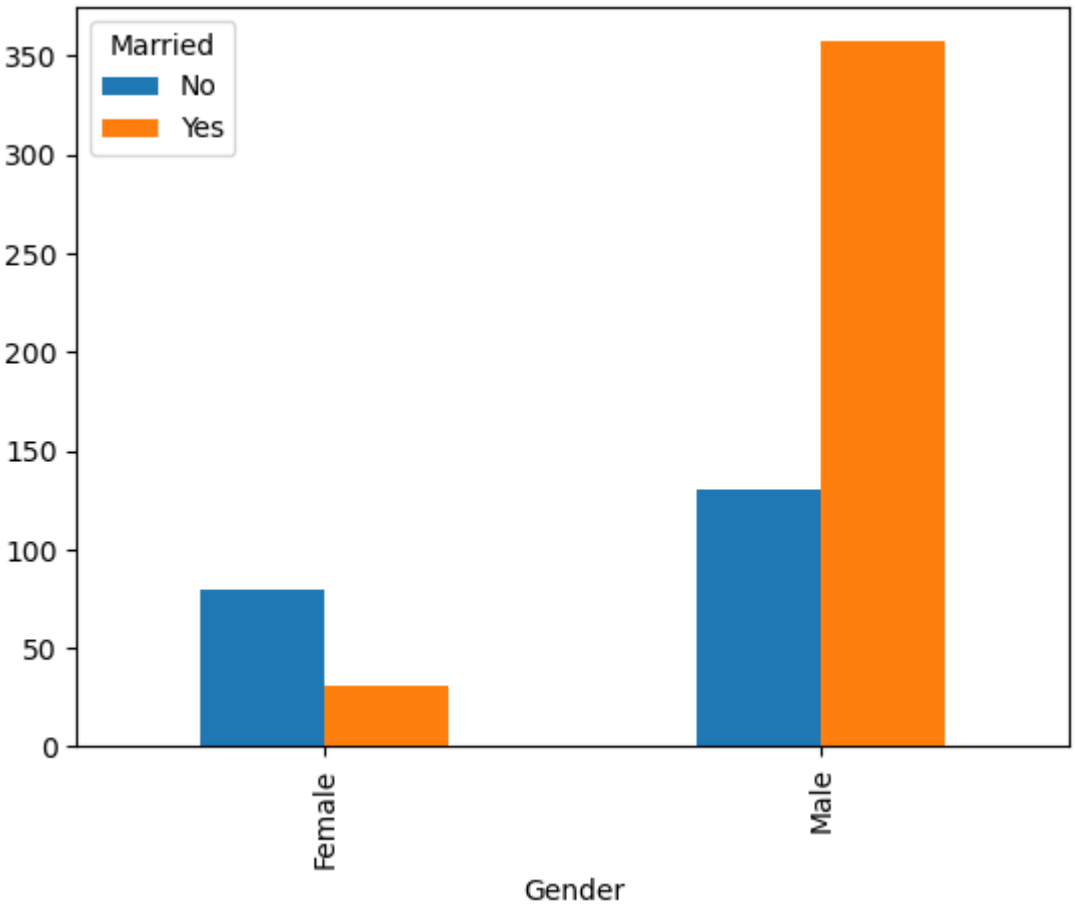
```
0 Married
```



In [48]:
```
import warnings
warnings.filterwarnings("ignore",category=UserWarning)
```

In [23]:
```
col1=loandf['Gender']
col2=loandf['Married']
#plt.subplot(1,1,1)
result2=pd.crosstab(col1,col2)
result2.plot(kind='bar')
```

Out[23]: <Axes: xlabel='Gender'>



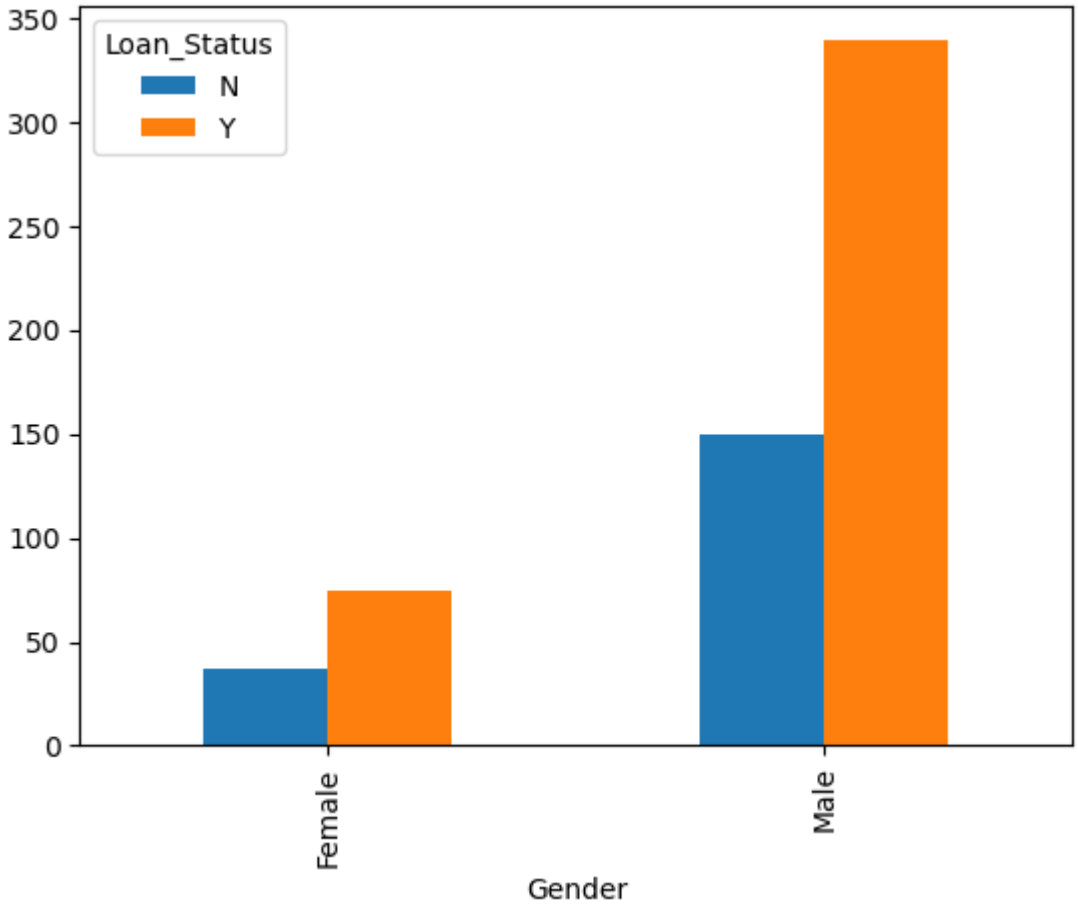
In [27]:
```
plt.figure(figsize=(8,8))
for i, column in enumerate(cat_col[2:]):
    print(i,column)
    col2=loandf[column]
    #plt.subplot(3,2,i+1)
    # where is the plot??
    result2=pd.crosstab(col1,col2)
    result2.plot(kind='bar')
plt.show()
```

```
0 Married
1 Dependents
2 Education
3 Self_Employed
4 Property_Area
5 Loan_Status

<Figure size 800x800 with 0 Axes>
```

In [ ]:

In [62]:
```
result2.plot(kind='bar')
```

Out[62]: <Axes: xlabel='Gender'>

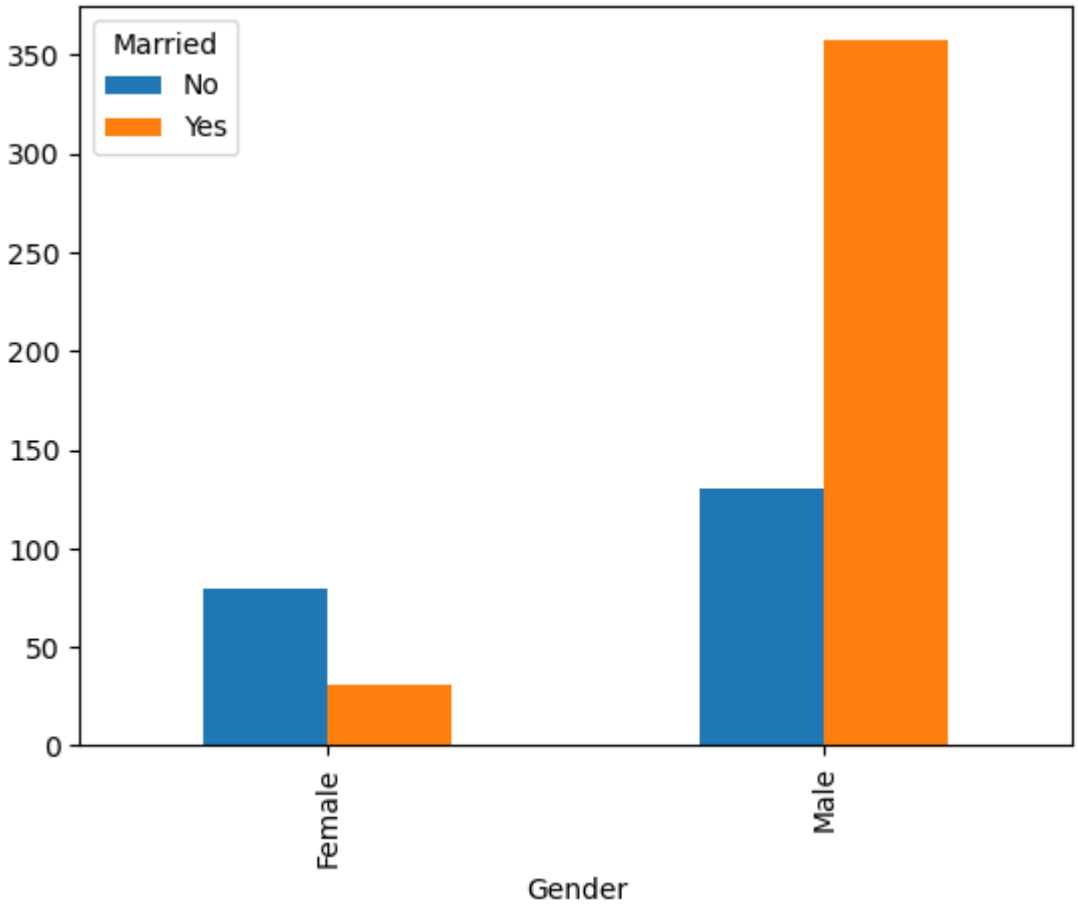


In [31]:
```
col1=loandf['Loan_Status']
for column in cat_col[1:7]:
    col2=loandf[column]
    print(pd.crosstab(col1,col2))
    print()
```

```
Gender       Female  Male
Loan_Status
N                37   150
Y                75   339

Married       No  Yes
Loan_Status
N             79  113
Y            134  285

Dependents     0   1   2  3+
Loan_Status
N            107  36  25  18
Y            238  66  76  33

Education    Graduate  Not Graduate
Loan_Status
N                 140            52
Y                 340            82

Self_Employed   No  Yes
Loan_Status
N              157   26
Y              343   56

Property_Area  Rural  Semiurban  Urban
Loan_Status
N                 69         54     69
Y                110        179    133
```

In [65]:
```
result5.plot(kind='bar')
```

Out[65]:
```
<Axes: xlabel='Gender'>
```



In [33]:
```
labeles=loandf['Gender'].unique()
y_count=[]
n_count=[]
for column in cat_col:
    con1=loandf[column]==i
    con2=loandf[column]=='Yes'
    con3=loandf[column]=='No'
    y_count.append(len(loandf[con1&con2]))
    n_count.append(len(loandf[con1&con2]))


pd.DataFrame(zip(labeles,y_count,n_count),columns=['Gender','yes','no'])
```

Out[33]:

	Gender	yes	no
0	Male	0	0
1	Female	0	0
2	NaN	0	0

In [ ]:
```

```

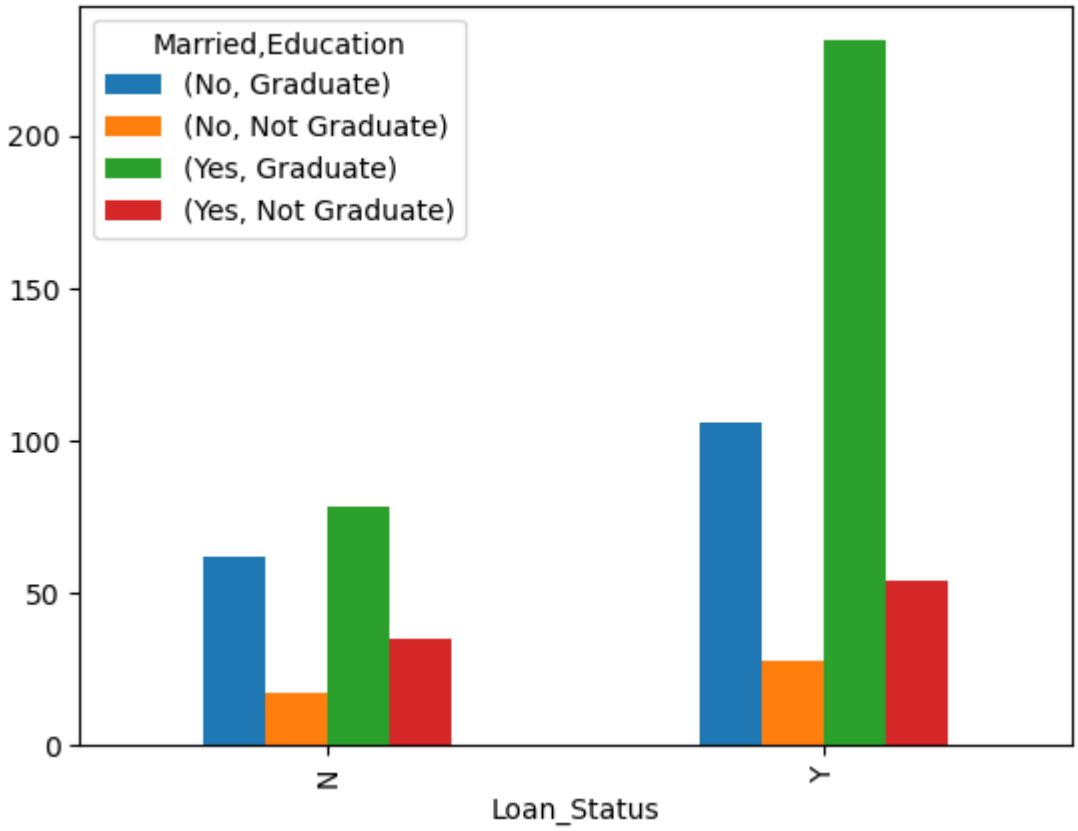
In [34]:
```
con1=loandf['Gender']
con2=loandf['Married']
con3=loandf['Education']
col=[con2,con3]
result4=pd.crosstab(col1,col)
result4
```

Out[34]:

Married	No		Yes	
Education	Graduate	Not Graduate	Graduate	Not Graduate
Loan_Status				
N	62	17	78	35
Y	106	28	231	54

In [68]:
```
result4.plot(kind='bar')
```

Out[68]: <Axes: xlabel='Loan_Status'>



two nuemerical column analysis

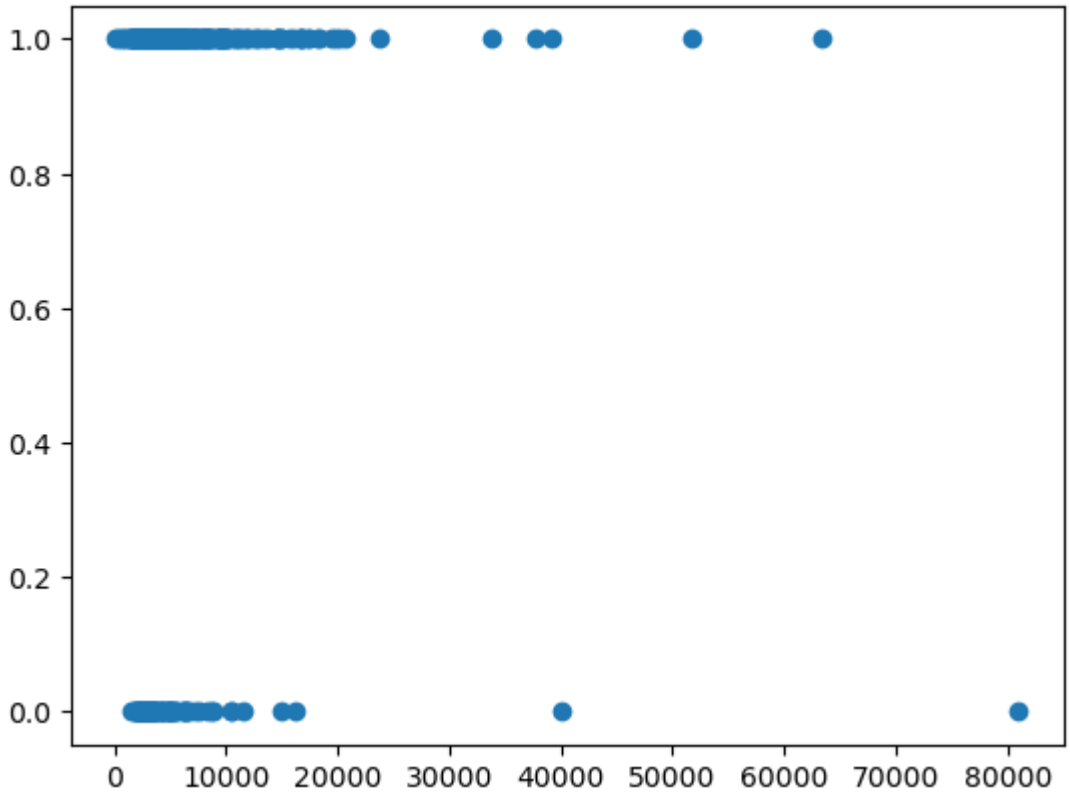
scatter plot

In [150]:
```
num_col
```

Out[150]:
```
['ApplicantIncome',
 'CoapplicantIncome',
 'LoanAmount',
 'Loan_Amount_Term',
 'Credit_History']
```

In [151]:
```
col1=loandf['ApplicantIncome']
col2=loandf['Credit_History']
plt.scatter(col1,col2)
```

Out[151]:
```
<matplotlib.collections.PathCollection at 0x17093696390>
```



correlation data.corr

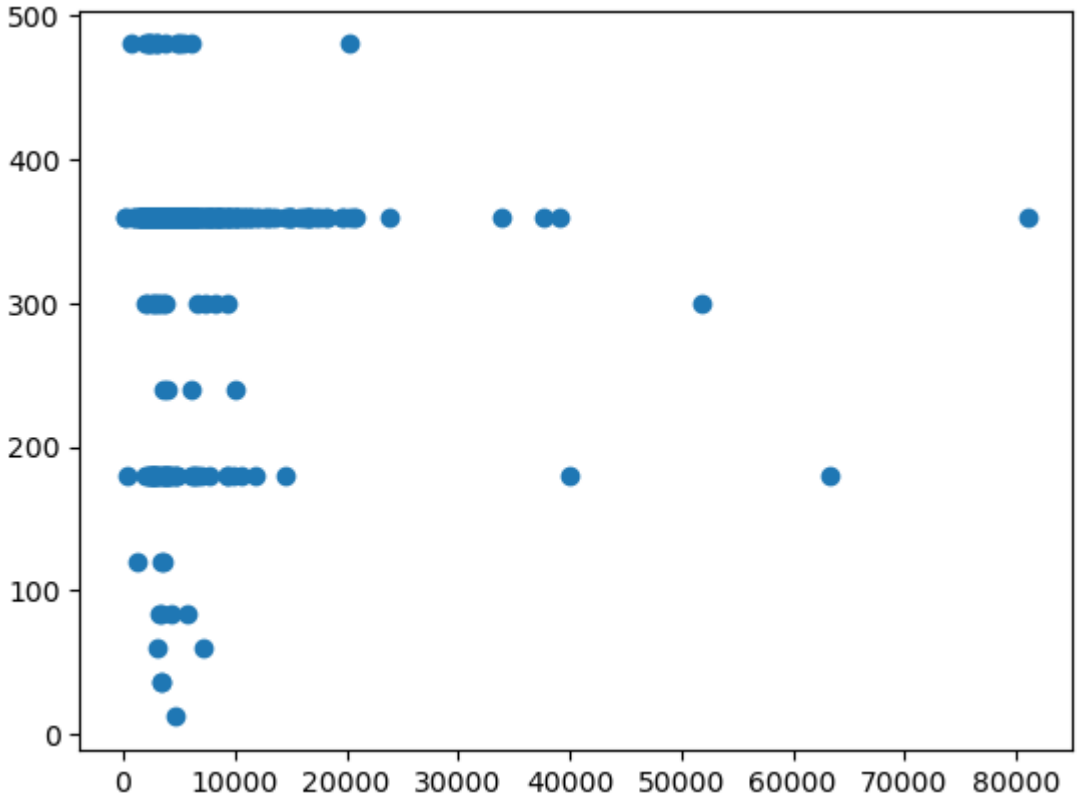
In [152]:
```
loandf.corr(numeric_only=True)
```

Out[152]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
ApplicantIncome	1.000000	-0.116605	0.570909	-0.045306
CoapplicantIncome	-0.116605	1.000000	0.188619	-0.059878
LoanAmount	0.570909	0.188619	1.000000	0.039447
Loan_Amount_Term	-0.045306	-0.059878	0.039447	1.000000
Credit_History	-0.014715	-0.002056	-0.008433	0.001470

In [153]:
```
plt.scatter(loandf['ApplicantIncome'],loandf['Loan_Amount_Term'])
```

Out[153]:
```
<matplotlib.collections.PathCollection at 0x17093b39810>
```



In [ ]:
```

```

heatmap

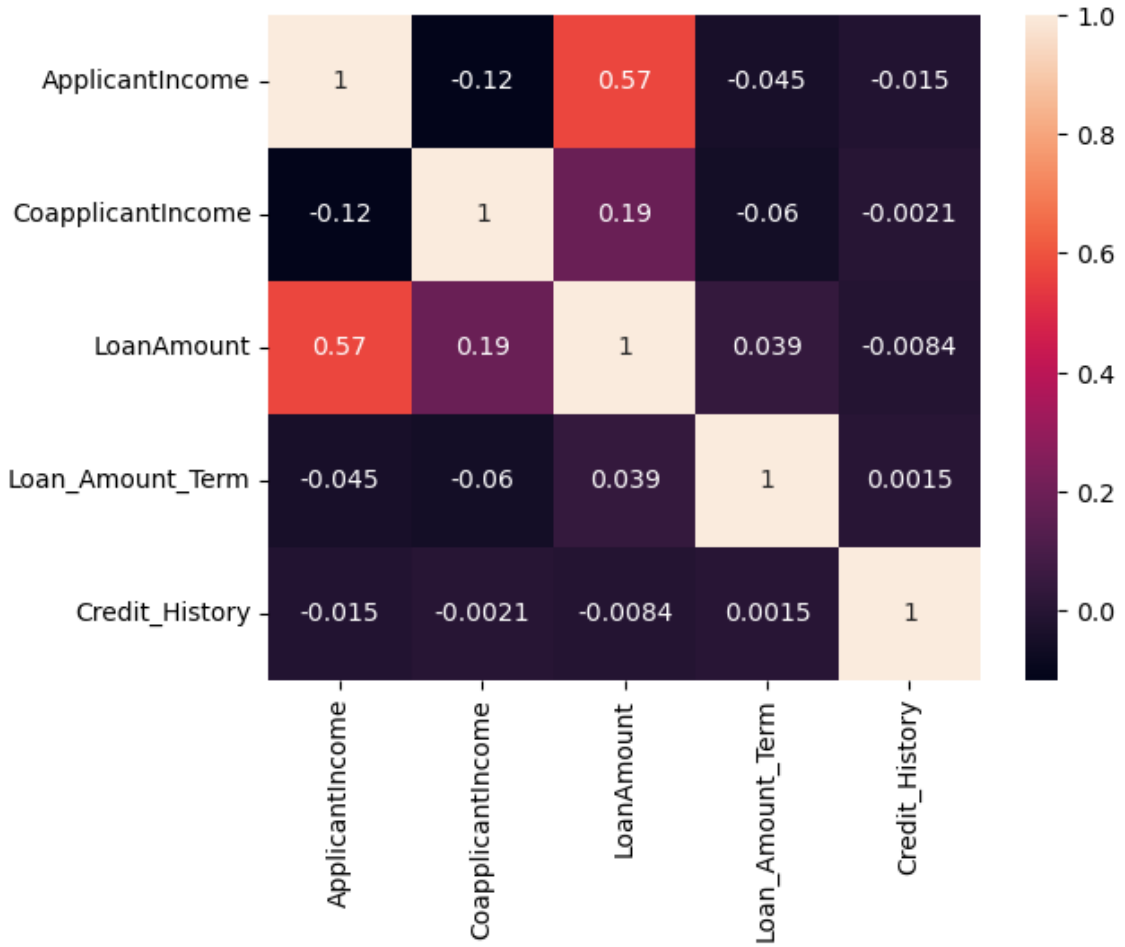
In [154]:
```
loandf.corr(numeric_only=True)
```

Out[154]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
ApplicantIncome	1.000000	-0.116605	0.570909	-0.045306
CoapplicantIncome	-0.116605	1.000000	0.188619	-0.059878
LoanAmount	0.570909	0.188619	1.000000	0.039447
Loan_Amount_Term	-0.045306	-0.059878	0.039447	1.000000
Credit_History	-0.014715	-0.002056	-0.008433	0.001470

In [155]:
```
corr_data=loandf.corr(numeric_only=True)
sns.heatmap(corr_data,annot=True)
```

Out[155]: <Axes: >



eda categorical to nuemerical

label encoder

In [156]:
```
file_path="C:\\Users\\surya\\Downloads\\loanpr.csv"
loandf=pd.read_csv(file_path)
loandf
```

Out[156]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
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	
...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

614 rows × 13 columns

In [157]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
loandf['Gender']=le.fit_transform(loandf['Gender'])
loandf['Married']=le.fit_transform(loandf['Married'])
print(loandf[['Gender','Married']].head(10))
```

```
   Gender  Married
0       1        0
1       1        1
2       1        1
3       1        1
4       1        0
5       1        1
6       1        1
7       1        1
8       1        1
9       1        1
```

In [ ]:
```

```

one hot encoder

In [158]:
```
file_path="C:\\Users\\surya\\Downloads\\loanpr.csv"
loandf=pd.read_csv(file_path)
loandf
```

Out[158]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
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	
...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

614 rows × 13 columns

In [122]:
```
loandf.drop('Loan_ID',axis=1,inplace=True)
```

```
---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
Cell In[122], line 1
----> 1 loandf.drop('Loan_ID',axis=1,inplace=True)
      2 loandf

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:5258, in DataFram
e.drop(self, labels, axis, index, columns, level, inplace, errors)
   5110 def drop(
   5111     self,
   5112     labels: IndexLabel = None,
   (...)
   5119     errors: IgnoreRaise = "raise",
   5120 ) -> DataFrame | None:
   5121     """
   5122     Drop specified labels from rows or columns.
   5123
   (...)
   5256             weight  1.0     0.8
   5257     """
-> 5258     return super().drop(
   5259         labels=labels,
   5260         axis=axis,
   5261         index=index,
   5262         columns=columns,
   5263         level=level,
   5264         inplace=inplace,
   5265         errors=errors,
   5266     )

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4549, in NDFram
e.drop(self, labels, axis, index, columns, level, inplace, errors)
   4547 for axis, labels in axes.items():
   4548     if labels is not None:
-> 4549         obj = obj._drop_axis(labels, axis, level=level, errors=err
ors)
   4551 if inplace:
   4552     self._update_inplace(obj)

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:4591, in NDFram
e._drop_axis(self, labels, axis, level, errors, only_slice)
   4589         new_axis = axis.drop(labels, level=level, errors=errors)
   4590     else:
-> 4591         new_axis = axis.drop(labels, errors=errors)
   4592     indexer = axis.get_indexer(new_axis)
   4594 # Case for non-unique axis
   4595 else:

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:6699, in In
dex.drop(self, labels, errors)
   6697 if mask.any():
   6698     if errors != "ignore":
-> 6699         raise KeyError(f"{list(labels[mask])} not found in axis")
   6700     indexer = indexer[~mask]
   6701 return self.delete(indexer)

KeyError: "['Loan_ID'] not found in axis"
```

In [123]:
```
loandf
```

Out[123]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplicantl
0	Male	No	0	Graduate	No	5849	
1	Male	Yes	1	Graduate	No	4583	
2	Male	Yes	0	Graduate	Yes	3000	
3	Male	Yes	0	Not Graduate	No	2583	
4	Male	No	0	Graduate	No	6000	
...	...	...	...	...	...	...	
609	Female	No	0	Graduate	No	2900	
610	Male	Yes	3+	Graduate	No	4106	
611	Male	Yes	1	Graduate	No	8072	
612	Male	Yes	2	Graduate	No	7583	
613	Female	No	0	Graduate	Yes	4583	

614 rows × 12 columns

In [159]:
```
pd.get_dummies(loandf,dtype='int')
```

Out[159]:

	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
...	...	...	...	...	...
609	2900	0.0	71.0	360.0	1.0
610	4106	0.0	40.0	180.0	1.0
611	8072	240.0	253.0	360.0	1.0
612	7583	0.0	187.0	360.0	1.0
613	4583	0.0	133.0	360.0	0.0

614 rows × 636 columns

In [160]:
```
cat_col
```

Out[160]:
```
['Loan_ID',
 'Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'Property_Area',
 'Loan_Status']
```

In [161]:
```
file_path="C:\\Users\\surya\\Downloads\\loanpr.csv"
loandf=pd.read_csv(file_path)
loandf
```

Out[161]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
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	
...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

614 rows × 13 columns

In [177]:
```
file_path="C:\\Users\\surya\\Downloads\\loanpr.csv"
loandf=pd.read_csv(file_path)
loandf


min_wage=loandf['ApplicantIncome'].min()
max_wage=loandf['ApplicantIncome'].max()
dr=max_wage-min_wage
nr=loandf['ApplicantIncome']-min_wage
loandf['ApplicantIncomenorm']=nr/dr
```

In [178]:
```
loandf[['ApplicantIncome','ApplicantIncomenorm']]
```

Out[178]:

	ApplicantIncome	ApplicantIncomenorm
0	5849	0.070489
1	4583	0.054830
2	3000	0.035250
3	2583	0.030093
4	6000	0.072356
...	...	...
609	2900	0.034014
610	4106	0.048930
611	8072	0.097984
612	7583	0.091936
613	4583	0.054830

614 rows × 2 columns

In [179]:
```
loandf['ApplicantIncomenorm'].max(),loandf['ApplicantIncomenorm'].min()
```

Out[179]:
```
(1.0, 0.0)
```

In [180]:
```
loandf['ApplicantIncome'].max(),loandf['ApplicantIncome'].min()
```

Out[180]:
```
(81000, 150)
```

In [181]:
```
max_id=loandf['ApplicantIncomenorm'].idxmax()
min_id=loandf['ApplicantIncomenorm'].idxmin()
max_id,min_id
```

Out[181]:
```
(409, 216)
```

In [184]:
```
loandf[['ApplicantIncome']].iloc[[max_id,min_id]]
```

Out[184]:

	ApplicantIncome
409	81000
216	150

In [ ]:
```

```

In [ ]:
```
##
```

min max scaler

In [193]:
```
file_path="C:\\Users\\surya\\Downloads\\loanpr.csv"
loandf=pd.read_csv(file_path)
loandf
```

Out[193]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
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	
...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

614 rows × 13 columns

In [ ]:
```
```

In [196]:
```
from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler()
loandf['ApplicantIncomenorm1']=mms.fit_transform(loandf[['ApplicantIncome']
```

In [197]:
```
loandf[['ApplicantIncomenorm1','ApplicantIncome']]
```

Out[197]:

	ApplicantIncomenorm1	ApplicantIncome
0	0.070489	5849
1	0.054830	4583
2	0.035250	3000
3	0.030093	2583
4	0.072356	6000
...	...	...
609	0.034014	2900
610	0.048930	4106
611	0.097984	8072
612	0.091936	7583
613	0.054830	4583

614 rows × 2 columns

In [ ]:
```
v1=np.array([[[]]])
```

In [199]:
```
loandf[['ApplicantIncome']]
```

Out[199]:

	ApplicantIncome
0	5849
1	4583
2	3000
3	2583
4	6000
...	...
609	2900
610	4106
611	8072
612	7583
613	4583

614 rows × 1 columns

Z SCORE

In [200]:
```
mean_wage=loandf['ApplicantIncome'].mean()
std_wage=loandf['ApplicantIncome'].std()
nr=loandf['ApplicantIncome']-mean_wage
loandf['ApplicantIncome_zscore']=nr/std_wage
```

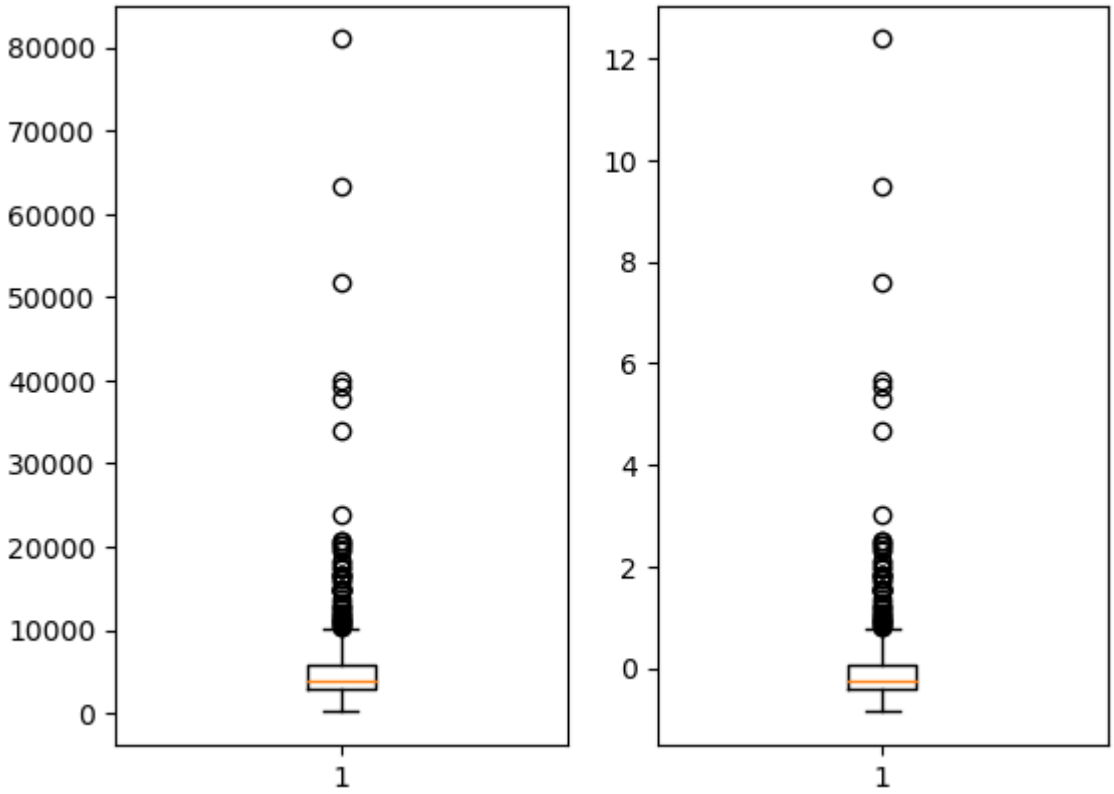
In [201]:
```
loandf[['ApplicantIncome','ApplicantIncome_zscore']]
```

Out[201]:

	ApplicantIncome	ApplicantIncome_zscore
0	5849	0.072931
1	4583	-0.134302
2	3000	-0.393427
3	2583	-0.461686
4	6000	0.097649
...	...	...
609	2900	-0.409796
610	4106	-0.212383
611	8072	0.436818
612	7583	0.356773
613	4583	-0.134302

614 rows × 2 columns

In [202]:
```
plt.subplot(1,2,1)
plt.boxplot(loandf['ApplicantIncome'])
plt.subplot(1,2,2)
plt.boxplot(loandf['ApplicantIncome_zscore'])
plt.show()
```



In [ ]:
```

```