Import the packages

In [2]:

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

Read the packages

In [3]:

```
path=r'C:\Users\DELL\Documents\project\data set\loan_data.csv'
loan_df=pd.read_csv(path)
loan_df
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantInco
0	LP001002	Male	No	0	Graduate	No	58
1	LP001003	Male	Yes	1	Graduate	No	45
2	LP001005	Male	Yes	0	Graduate	Yes	30
3	LP001006	Male	Yes	0	Not Graduate	No	25
4	LP001008	Male	No	0	Graduate	No	60
...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	29
610	LP002979	Male	Yes	3+	Graduate	No	41
611	LP002983	Male	Yes	1	Graduate	No	80
612	LP002984	Male	Yes	2	Graduate	No	75
613	LP002990	Female	No	0	Graduate	Yes	45

614 rows × 13 columns

Convert in to cat_columns and num_columns

In [4]:

```
cat_columns=loan_df.select_dtypes(include='object').columns
num_columns=loan_df.select_dtypes(exclude='object').columns
```

In [5]:

```
cat_columns
```

Out[5]:

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [6]:

```
num_columns
```

Out[6]:

```
Index(['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History'],
      dtype='object')
```

Creat a that frame

In [7]:
```
loan_df[['Education']]
```

Out[7]:

	Education
0	Graduate
1	Graduate
2	Graduate
3	Not Graduate
4	Graduate
...	...
609	Graduate
610	Graduate
611	Graduate
612	Graduate
613	Graduate

614 rows × 1 columns

In [8]:
```
loan_df['Married'].unique()
```

Out[8]:
```
array(['No', 'Yes', nan], dtype=object)
```

In [9]:
```
len(loan_df['Married'].unique())
```

Out[9]:
```
3
```

In [10]:
```
cdf=loan_df['Married'].value_counts()
cdf
```

Out[10]:
```
Married
Yes    398
No     213
Name: count, dtype: int64
```

In [11]:
```
type(cdf)
```

Out[11]:
```
pandas.core.series.Series
```

In [12]:
```
keys=cdf.keys()
keys
```

Out[12]:
```
Index(['Yes', 'No'], dtype='object', name='Married')
```

In [13]:
```
values=cdf.values
values
```

Out[13]:

```
array([398, 213], dtype=int64)
```

In [14]:

```
pd.DataFrame(zip(keys,values))
```

Out[14]:

	0	1
0	Yes	398
1	No	213

Coverted into Lables and Count

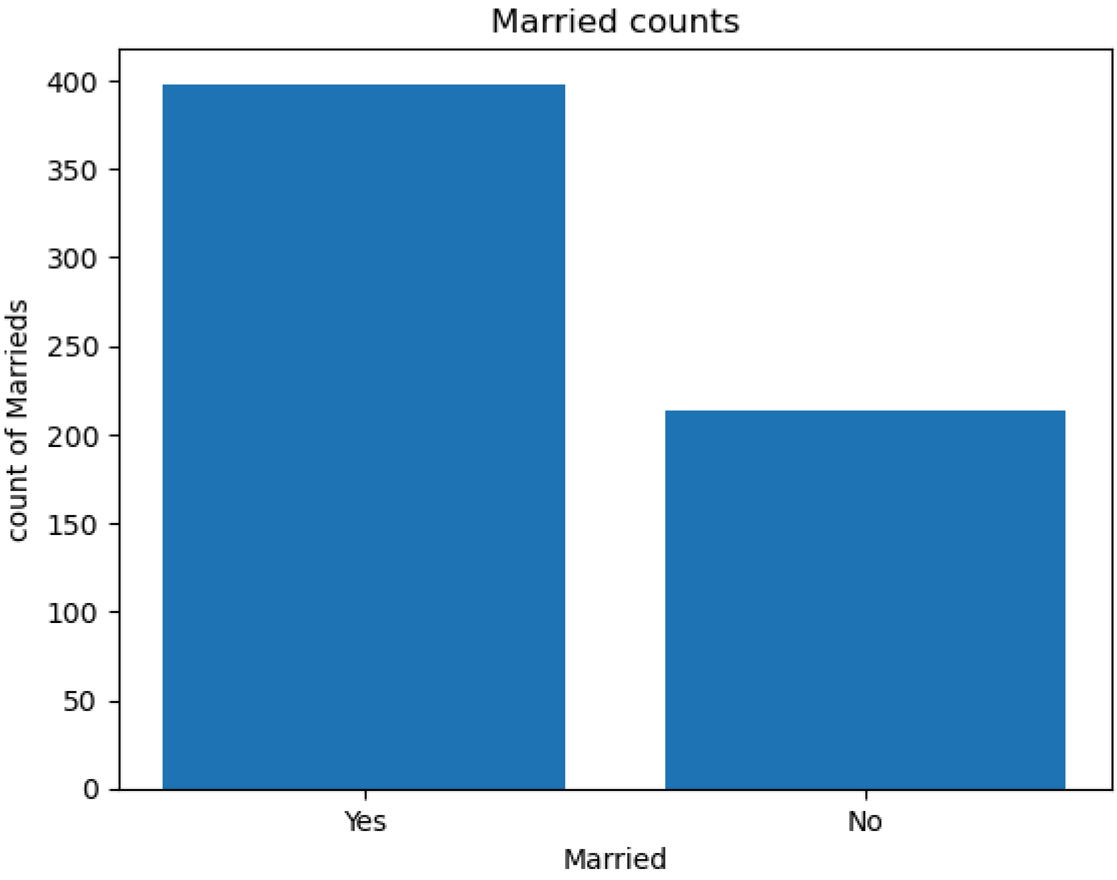
In [15]:

```
cdf=loan_df['Married'].value_counts()
keys=cdf.keys()
values=cdf.values
cols=['Lables','Count']
df=pd.DataFrame(zip(keys,values),columns=cols)
df.to_csv('Married.csv',index=False)
```

using matplotlib.pyplot

In [16]:

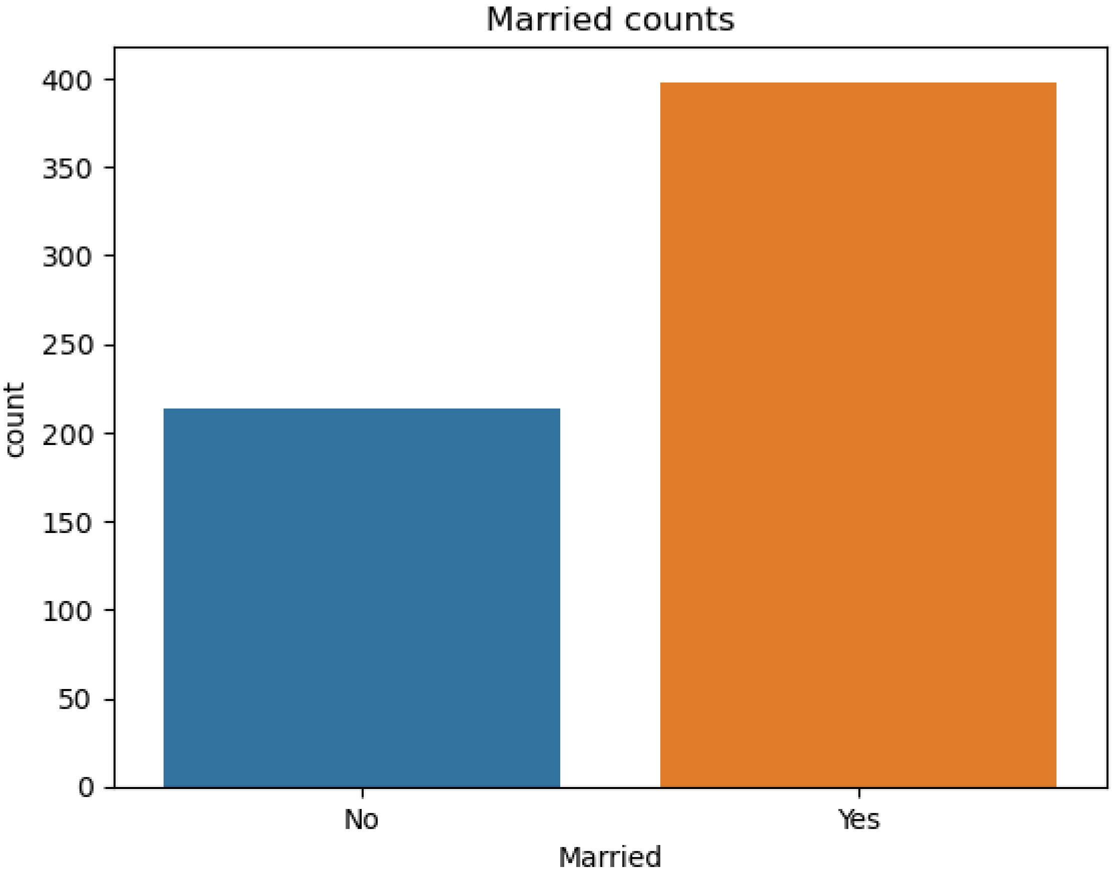
```
import matplotlib.pyplot as plt
plt.bar('Lables','Count',data=df)
plt.title('Married counts')
plt.xlabel('Married')
plt.ylabel('count of Marrieds')
plt.show()
```



using seaborn

In [17]:

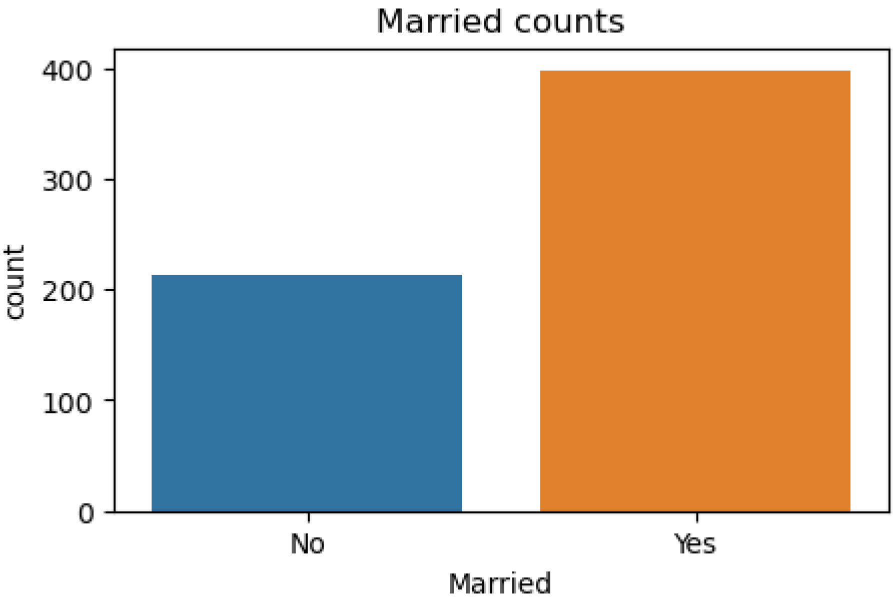
```
import seaborn as sns
sns.countplot(data=loan_df,x='Married')
plt.title('Married counts')
plt.show()
```



if we want take a figure size

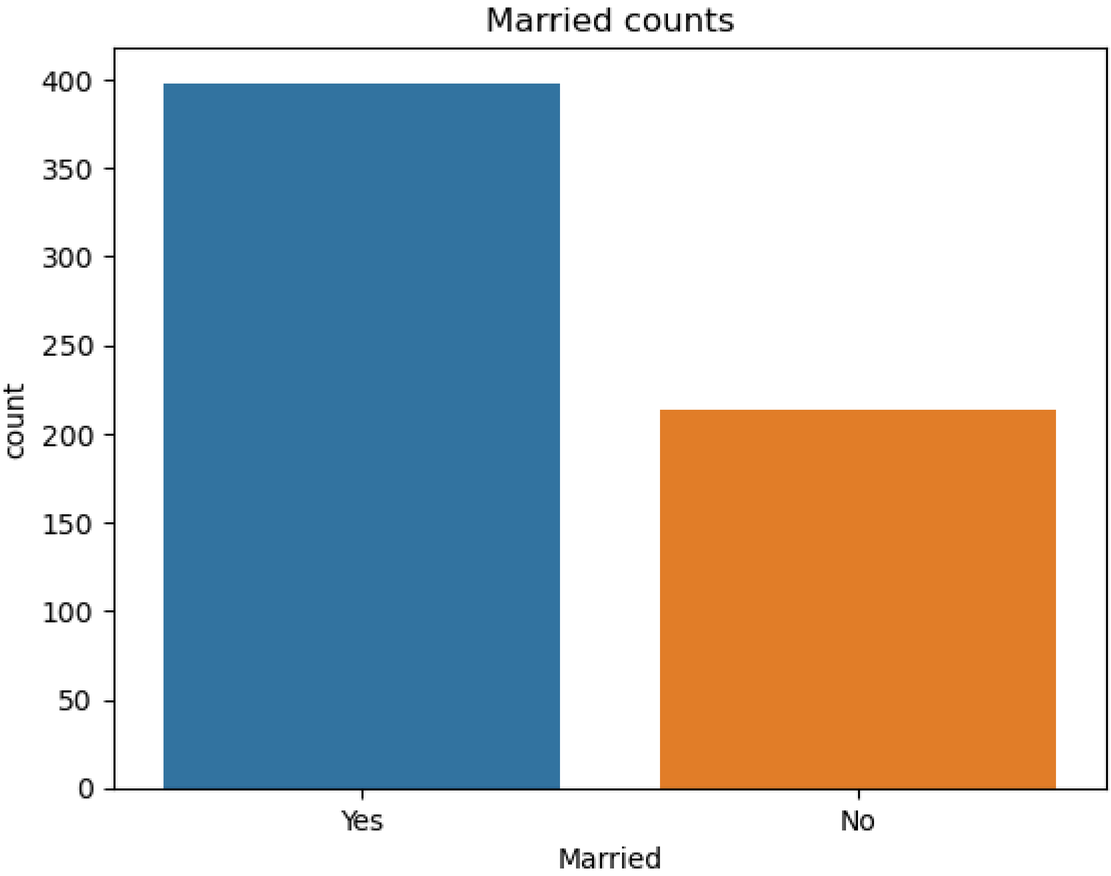
In [18]:

```
import seaborn as sns
plt.figure(figsize=(5,3))
sns.countplot(data=loan_df,x='Married')
plt.title('Married counts')
plt.show()
```



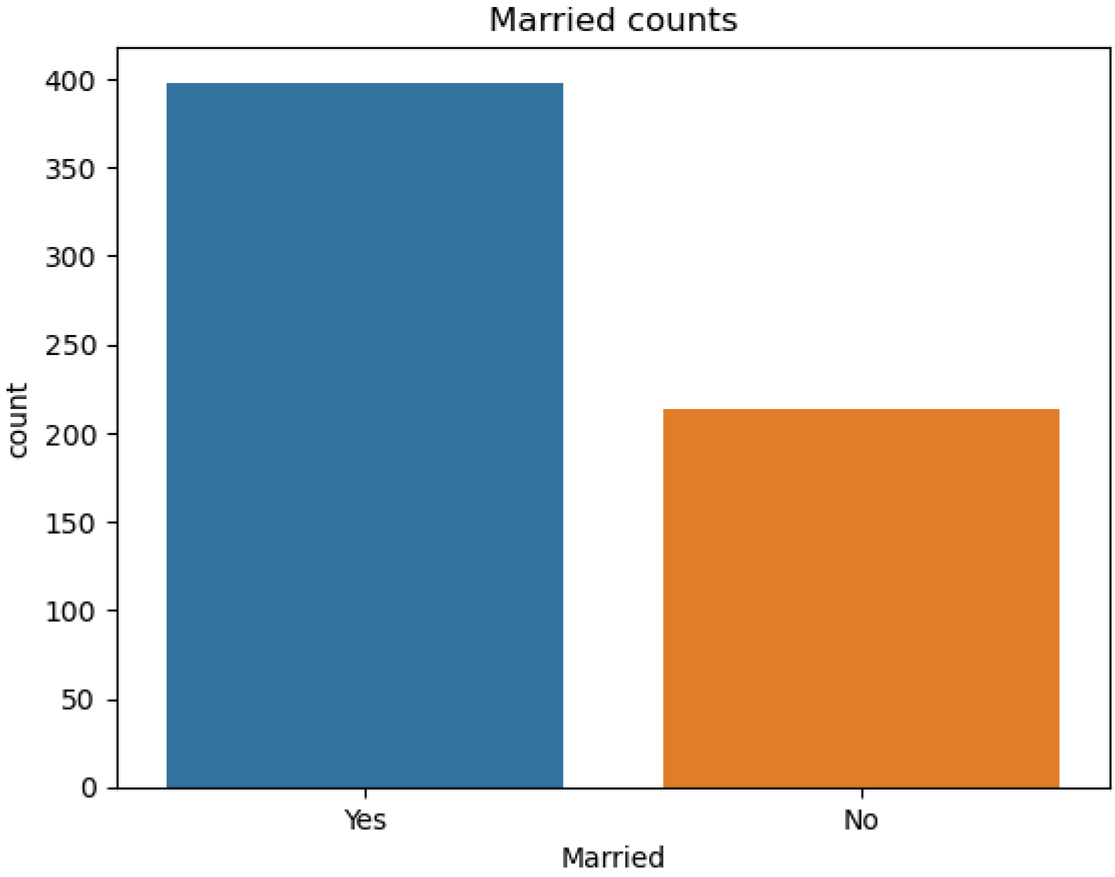
If we can in order format

In [19]:
```
# Method 1
import seaborn as sns
order_con=['Yes','No']
sns.countplot(data=loan_df,x='Married',order=order_con)
plt.title('Married counts')
plt.show()
```



In [20]:
```
# Method 2
import seaborn as sns
order_con=loan_df['Married'].value_counts().keys()
```

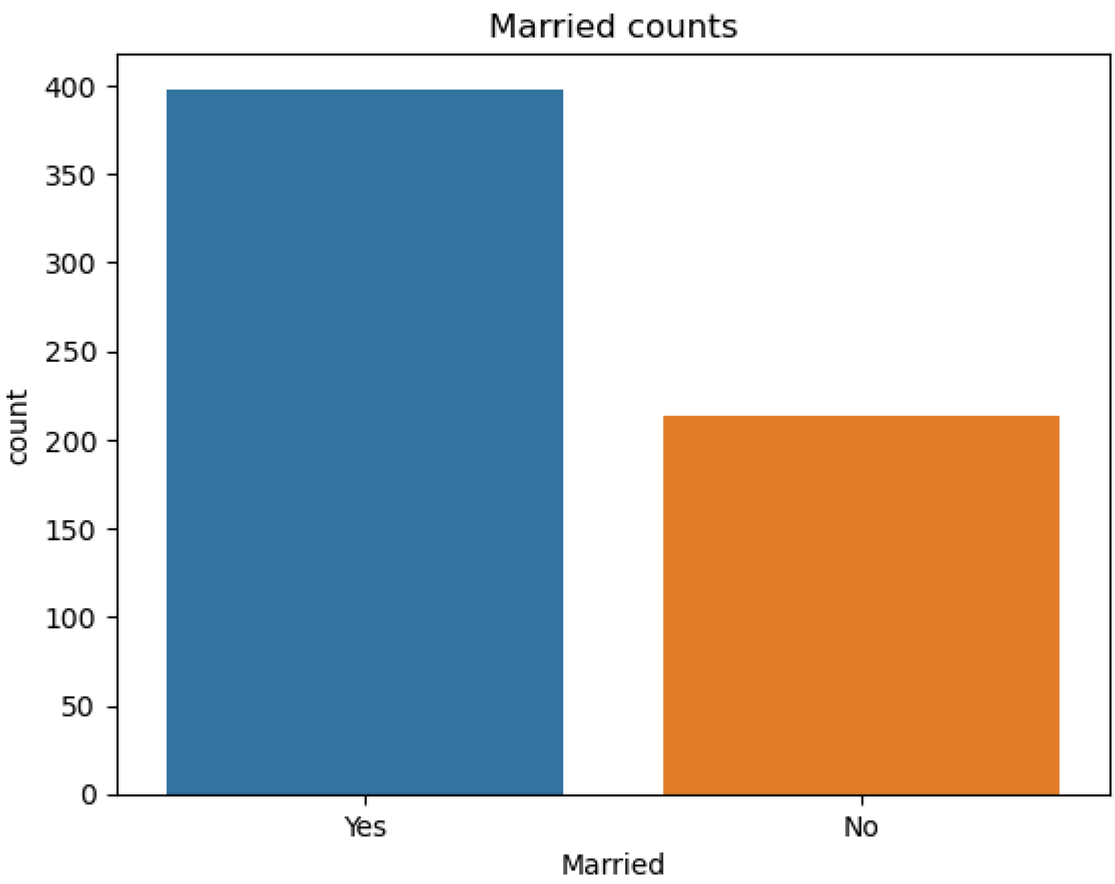
```
sns.countplot(data=loan_df,x='Married',order=order_con)
plt.title('Married counts')
plt.show()
```



For save the figure

In [21]:

```
import seaborn as sns
order_con=['Yes','No']
sns.countplot(data=loan_df,x='Married',order=order_con)
plt.title('Married counts')
plt.savefig('Married_bar_using_sns.jpg')
plt.show()
```



Seaborn using For loop plots the Bar charts of all the Cat columns

In [22]:

```
import os
os.getcwd()
```

Out[22]:

```
'C:\\Users\\DELL\\Documents\\project'
```

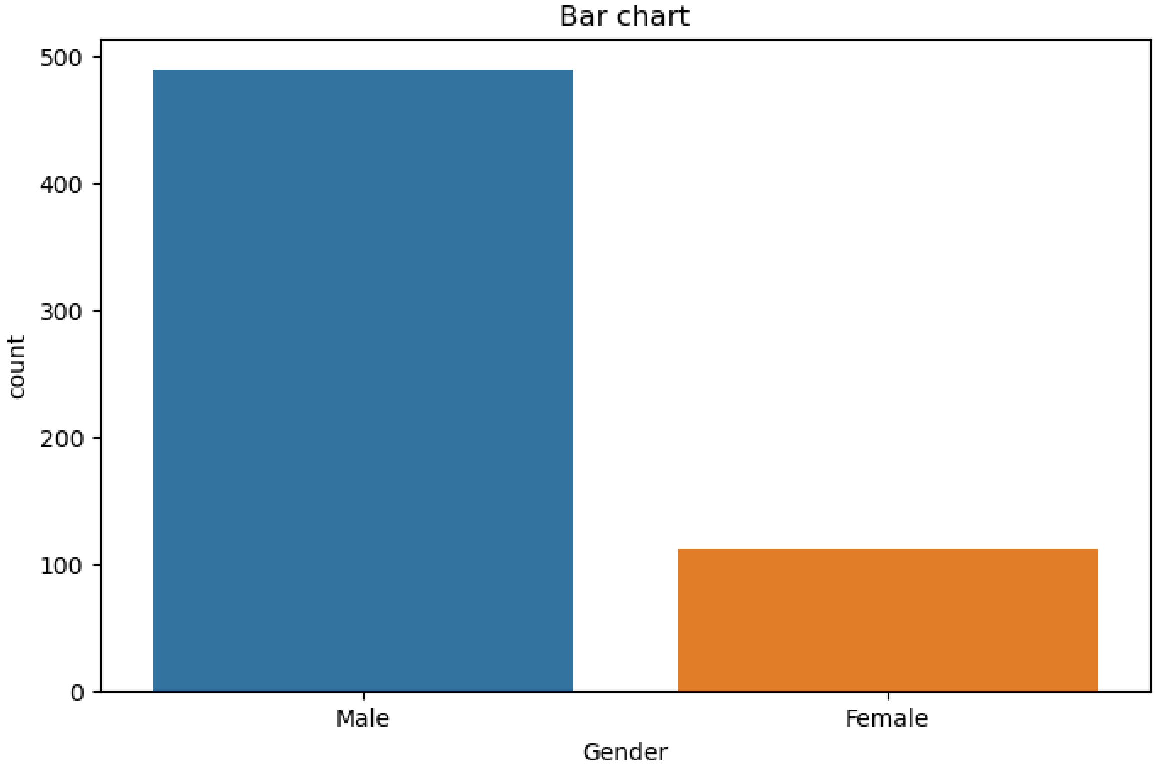
In [23]:

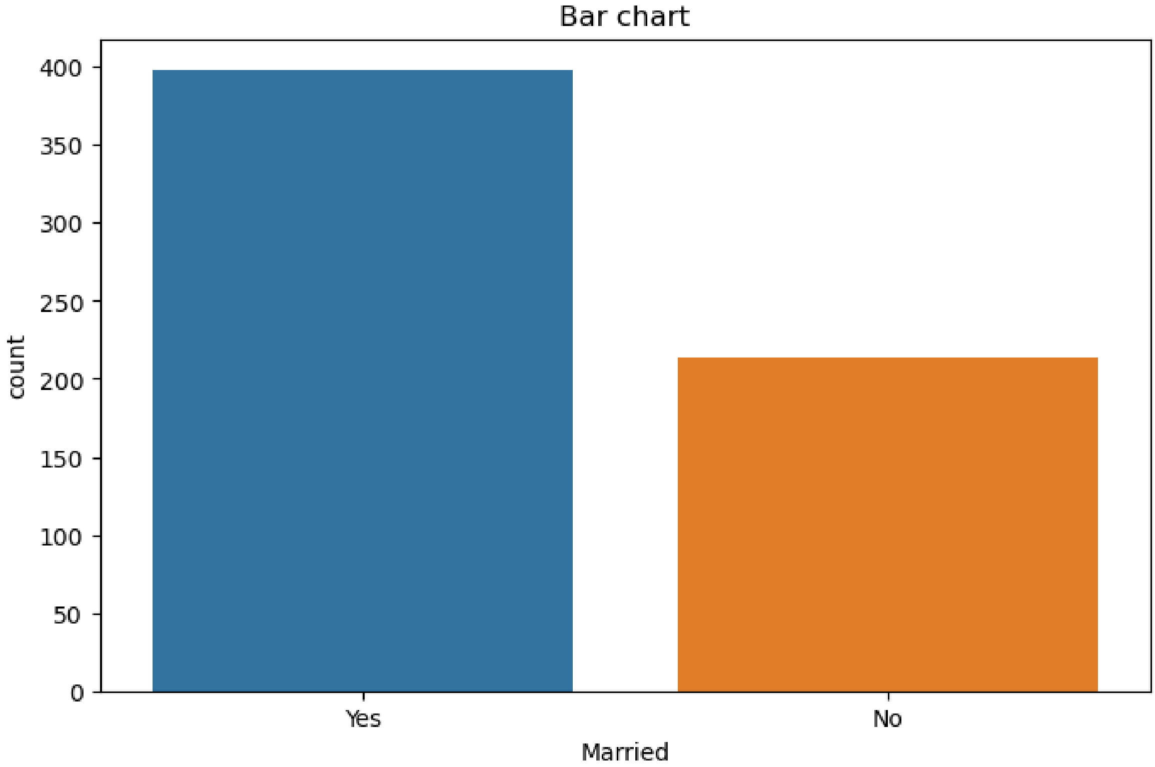
```
try:
    root_directory=os.getcwd()
    new_folder='new data'
    new_dir=os.path.join(root_directory,new_folder)
    os.makedirs(new_dir)
except Exception as e:
    print(e)
```

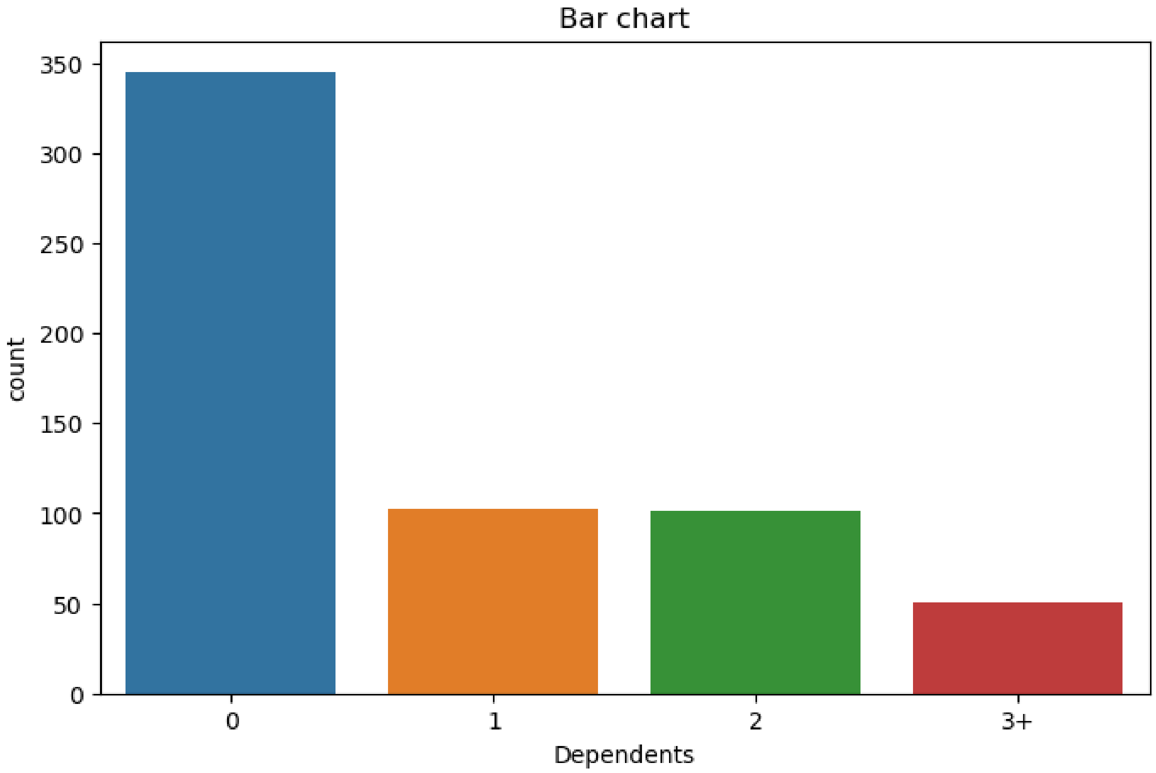
```
[WinError 183] Cannot create a file when that file already exists: 'C:\\Users\\DE
LL\\Documents\\project\\new data'
```

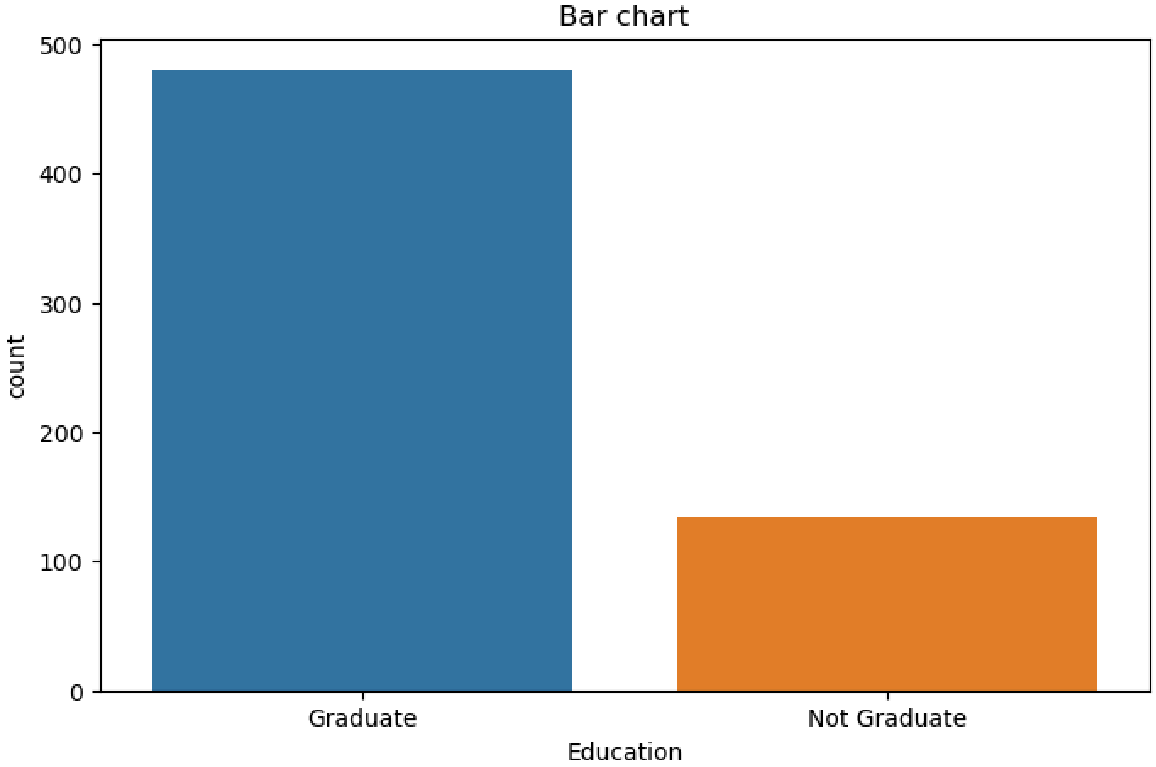
In [24]:

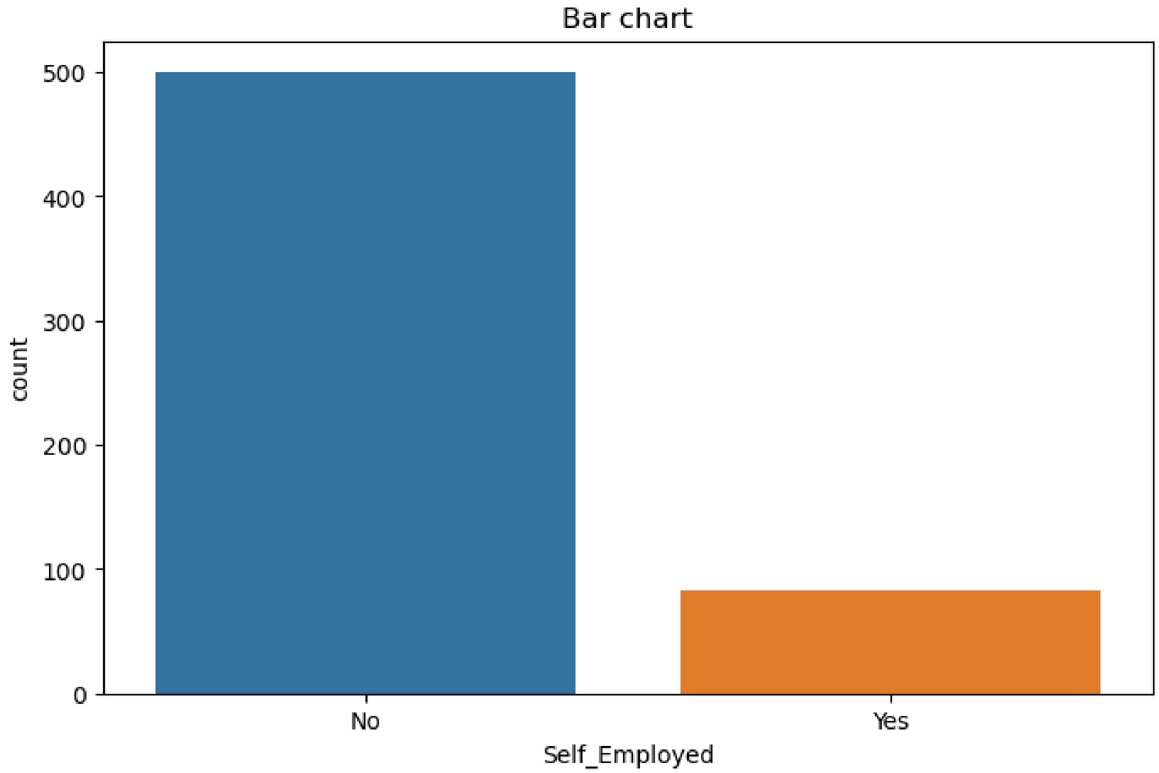
```
import seaborn as sns
for i in cat_columns[1:]:
    plt.figure(figsize=(8,5))
    order_continents=loan_df[i].value_counts().keys()
    sns.countplot(data=loan_df,
                  x=i,
                  order=order_continents)
    plt.title('Bar chart')
    plt.savefig(f"{new_dir}\\{i}_seaborn.jpg")
    plt.show()
```

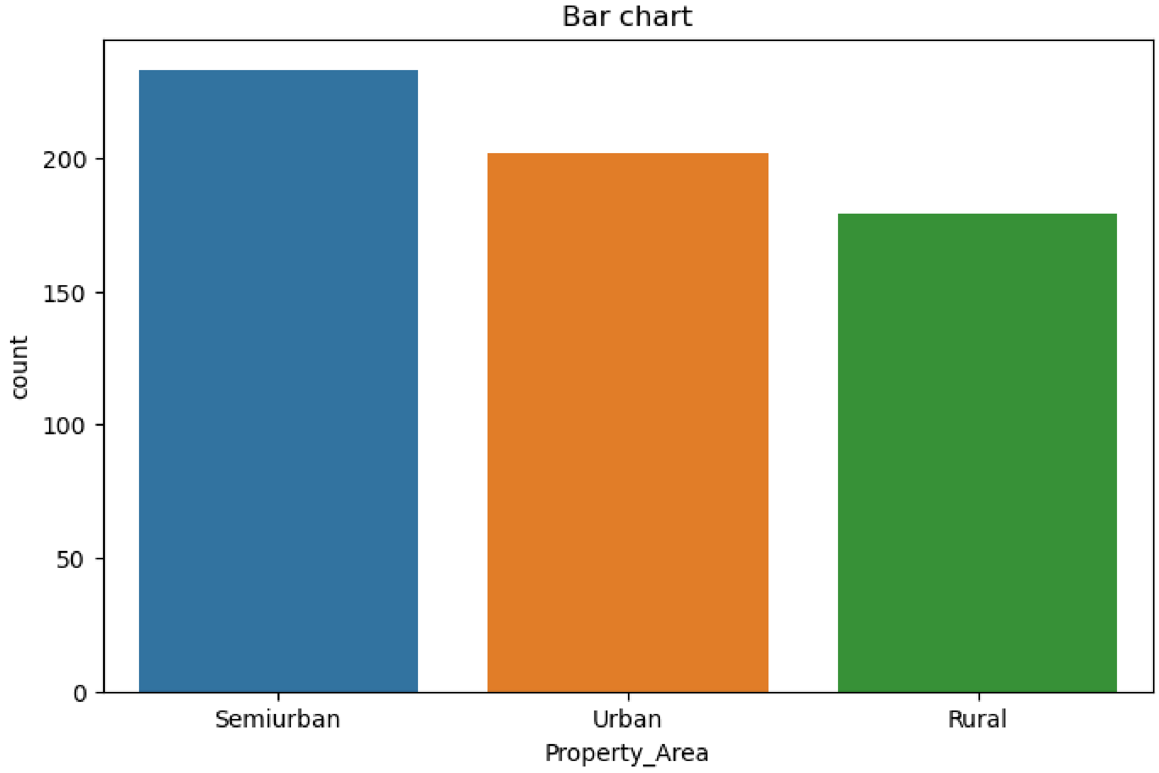

Bar chart
500
400
300
200
100
0
count
Male
Female
Gender

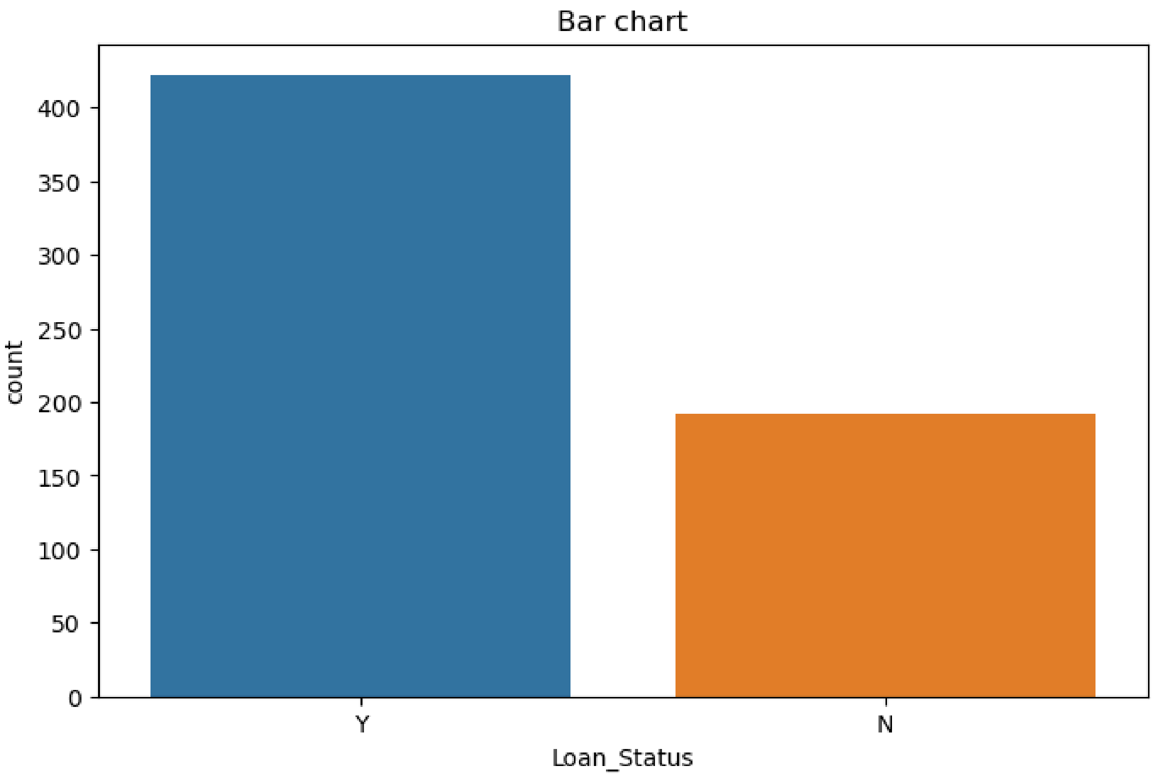

Bar chart
400
350
300
250
200
150
100
50
0
count
Yes
No
Married


Bar chart
count
350
300
250
200
150
100
50
0
0
1
2
3+
Dependents


Bar chart
count
500
400
300
200
100
0
Graduate
Not Graduate
Education


Bar chart
count
500
400
300
200
100
0
No
Yes
Self_Employed


Bar chart
count
200
150
100
50
0
Semiurban
Urban
Rural
Property_Area



Multipltlib.pyplot using For loop plots the Bar charts of all the Cat columns

In [25]:

```
for i in cat_columns[1:]:
    cdf=loan_df[i].value_counts()
    keys=cdf.keys()
    values=cdf.values
    cols=['Lables','Counts']
    df=pd.DataFrame(zip(keys,values),columns=cols)
    df.to_csv(f'{i}.csv',index=False)

df
```

Out[25]:

	Lables	Counts
0	Y	422
1	N	192

In [26]:

```
import os
os.getcwd()
try:
    root_directory=os.getcwd()
    new_folder='loan_data'
    new_dir=os.path.join(root_directory,new_folder)
    os.makedirs(new_dir)
except Exception as e:
    print(e)
```

```
[WinError 183] Cannot create a file when that file already exists: 'C:\\Users\\DE
LL\\Documents\\project\\loan_data'
```

In [27]:

```
dfs=os.listdir(r"C:\Users\DELL\Documents\project\loan_data")
dfs
```

Out[27]:

```
['Dependents.csv',
 'Dependents.csv_matplotlip.jpg',
 'Education.csv',
 'Gender.csv',
 'Loan_Status.csv',
 'Married.csv',
 'Property_Area.csv',
 'Self_Employed.csv']
```

In [28]:

```
root_directory=os.getcwd()
new_folder='loan_data'
dir=os.path.join(root_directory,new_folder)
```

In [29]:

```
dir
```

Out[29]:

```
'C:\\Users\\DELL\\Documents\\project\\loan_data'
```

In [30]:

```
pd.read_csv(r'C:\Users\DELL\Documents\project\loan_data\Married.csv')
```

Out[30]:

	Lables	Counts
0	Yes	398
1	No	213

In [31]:

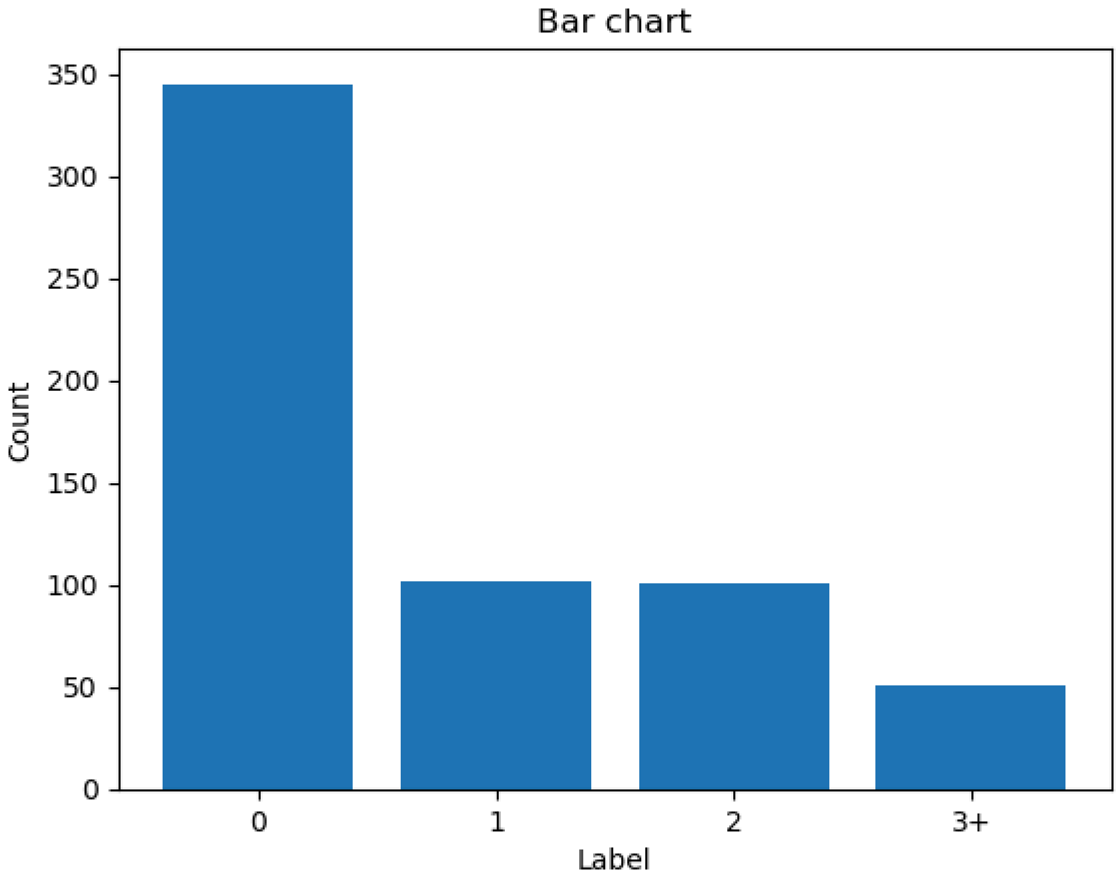
```
dfs=os.listdir(dir)
dfs
```

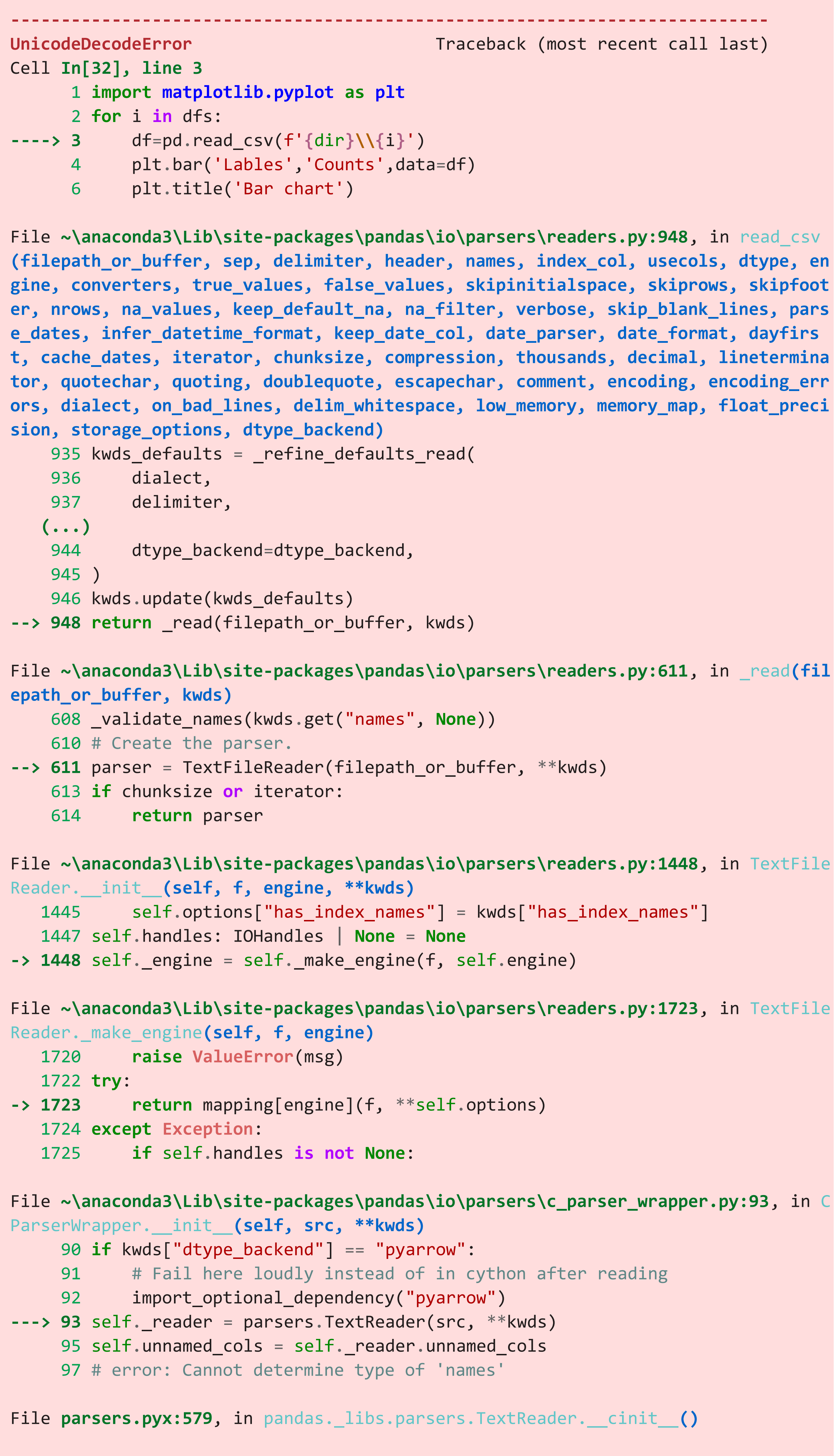
Out[31]:

```
['Dependents.csv',
 'Dependents.csv_matplotlip.jpg',
 'Education.csv',
 'Gender.csv',
 'Loan_Status.csv',
 'Married.csv',
 'Property_Area.csv',
 'Self_Employed.csv']
```

In [32]:

```
import matplotlib.pyplot as plt
for i in dfs:
    df=pd.read_csv(f'{dir}\\{i}')
    plt.bar('Lables','Counts',data=df)

    plt.title('Bar chart')
    plt.xlabel('Label')
    plt.ylabel('Count')
    plt.savefig(f'{new_dir}\\{i}_matplotlip.jpg')
    plt.show()
```


Bar chart
Count
Label
0
50
100
150
200
250
300
350
0
1
2
3+



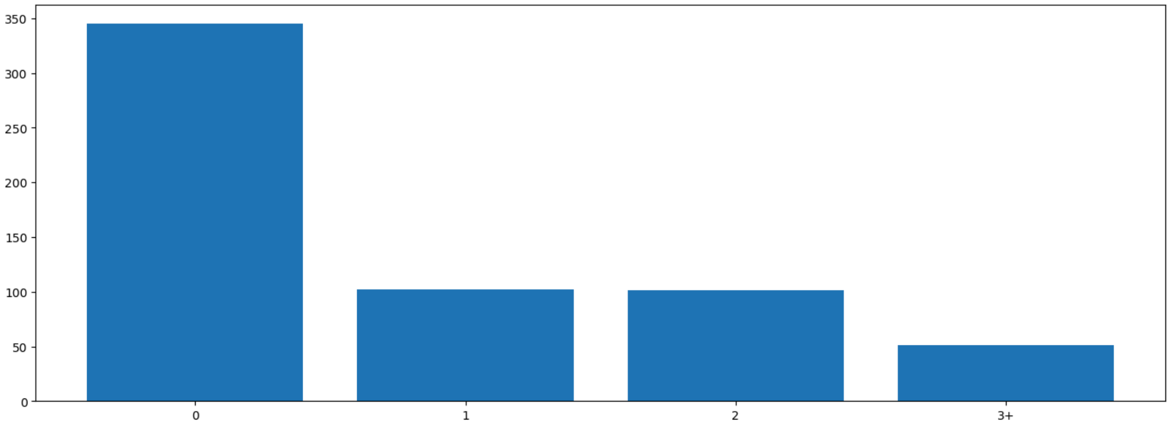
```
---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent call last)
Cell In[32], line 3
      1 import matplotlib.pyplot as plt
      2 for i in dfs:
----> 3     df=pd.read_csv(f'{dir}\\{i}')
      4     plt.bar('Lables','Counts',data=df)
      6     plt.title('Bar chart')

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:948, in read_csv
(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, dtype, en
gine, converters, true_values, false_values, skipinitialspace, skiprows, skipfoot
er, nrows, na_values, keep_default_na, na_filter, verbose, skip_blank_lines, pars
e_dates, infer_datetime_format, keep_date_col, date_parser, date_format, dayfirs
t, cache_dates, iterator, chunksize, compression, thousands, decimal, linetermina
tor, quotechar, quoting, doublequote, escapechar, comment, encoding, encoding_err
ors, dialect, on_bad_lines, delim_whitespace, low_memory, memory_map, float_preci
sion, storage_options, dtype_backend)
    935 kwds_defaults = _refine_defaults_read(
    936     dialect,
    937     delimiter,
   (...)
    944     dtype_backend=dtype_backend,
    945 )
    946 kwds.update(kwds_defaults)
--> 948 return _read(filepath_or_buffer, kwds)

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:611, in _read(fil
epath_or_buffer, kwds)
    608 _validate_names(kwds.get("names", None))
    610 # Create the parser.
--> 611 parser = TextFileReader(filepath_or_buffer, **kwds)
    613 if chunksize or iterator:
    614     return parser

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1448, in TextFile
Reader.__init__(self, f, engine, **kwds)
   1445     self.options["has_index_names"] = kwds["has_index_names"]
   1447 self.handles: IOHandles | None = None
-> 1448 self._engine = self._make_engine(f, self.engine)

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1723, in TextFile
Reader._make_engine(self, f, engine)
   1720     raise ValueError(msg)
   1722 try:
-> 1723     return mapping[engine](f, **self.options)
   1724 except Exception:
   1725     if self.handles is not None:

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\c_parser_wrapper.py:93, in C
ParserWrapper.__init__(self, src, **kwds)
     90 if kwds["dtype_backend"] == "pyarrow":
     91     # Fail here loudly instead of in cython after reading
     92     import_optional_dependency("pyarrow")
---> 93 self._reader = parsers.TextReader(src, **kwds)
     95 self.unnamed_cols = self._reader.unnamed_cols
     97 # error: Cannot determine type of 'names'

File parsers.pyx:579, in pandas._libs.parsers.TextReader.__cinit__()
File parsers.pyx:668, in pandas._libs.parsers.TextReader._get_header()

File parsers.pyx:879, in pandas._libs.parsers.TextReader._tokenize_rows()

File parsers.pyx:890, in pandas._libs.parsers.TextReader._check_tokenize_status()

File parsers.pyx:2050, in pandas._libs.parsers.raise_parser_error()

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xff in position 0: invalid s
tart byte
```

all thogeter

In [34]:

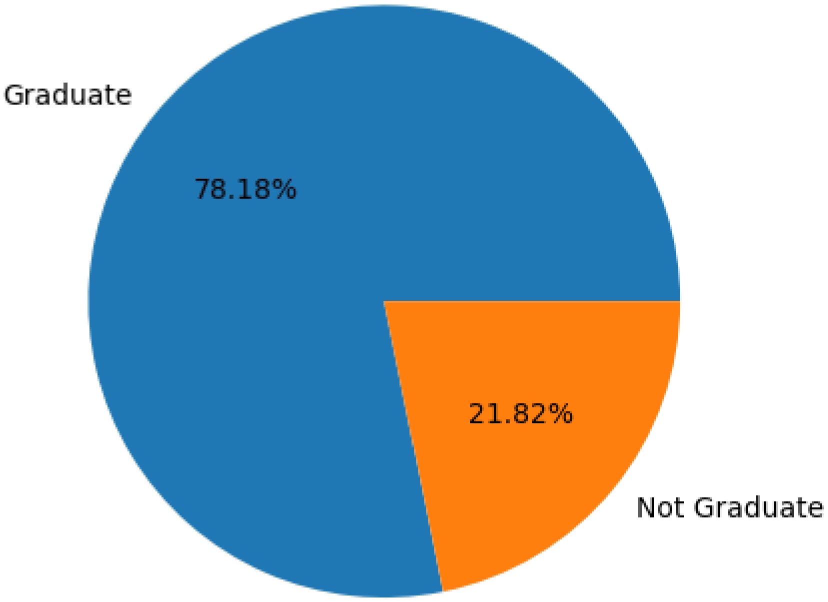
```
import matplotlib.pyplot as plt
plt.figure(figsize=(17,6))
for i in dfs:
    df=pd.read_csv(f'{dir}\\{i}')
    plt.bar('Lables','Counts',data=df)
```

```
---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent call last)
Cell In[34], line 4
      2 plt.figure(figsize=(17,6))
      3 for i in dfs:
----> 4     df=pd.read_csv(f'{dir}\\{i}')
      5     plt.bar('Lables','Counts',data=df)

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:948, in read_csv
(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, dtype, en
gine, converters, true_values, false_values, skipinitialspace, skiprows, skipfoot
er, nrows, na_values, keep_default_na, na_filter, verbose, skip_blank_lines, pars
e_dates, infer_datetime_format, keep_date_col, date_parser, date_format, dayfirs
t, cache_dates, iterator, chunksize, compression, thousands, decimal, linetermina
tor, quotechar, quoting, doublequote, escapechar, comment, encoding, encoding_err
ors, dialect, on_bad_lines, delim_whitespace, low_memory, memory_map, float_preci
sion, storage_options, dtype_backend)
    935 kwds_defaults = _refine_defaults_read(
    936     dialect,
    937     delimiter,
   (...)
    944     dtype_backend=dtype_backend,
    945 )
    946 kwds.update(kwds_defaults)
--> 948 return _read(filepath_or_buffer, kwds)

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:611, in _read(fil
epath_or_buffer, kwds)
    608 _validate_names(kwds.get("names", None))
    610 # Create the parser.
--> 611 parser = TextFileReader(filepath_or_buffer, **kwds)
    613 if chunksize or iterator:
    614     return parser

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1448, in TextFile
Reader.__init__(self, f, engine, **kwds)
   1445     self.options["has_index_names"] = kwds["has_index_names"]
   1447 self.handles: IOHandles | None = None
-> 1448 self._engine = self._make_engine(f, self.engine)

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1723, in TextFile
Reader._make_engine(self, f, engine)
   1720     raise ValueError(msg)
   1722 try:
-> 1723     return mapping[engine](f, **self.options)
   1724 except Exception:
   1725     if self.handles is not None:

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\c_parser_wrapper.py:93, in C
ParserWrapper.__init__(self, src, **kwds)
     90 if kwds["dtype_backend"] == "pyarrow":
     91     # Fail here loudly instead of in cython after reading
     92     import_optional_dependency("pyarrow")
---> 93 self._reader = parsers.TextReader(src, **kwds)
     95 self.unnamed_cols = self._reader.unnamed_cols
     97 # error: Cannot determine type of 'names'

File parsers.pyx:579, in pandas._libs.parsers.TextReader.__cinit__()

File parsers.pyx:668, in pandas._libs.parsers.TextReader._get_header()
File parsers.pyx:879, in pandas._libs.parsers.TextReader._tokenize_rows()

File parsers.pyx:890, in pandas._libs.parsers.TextReader._check_tokenize_status()

File parsers.pyx:2050, in pandas._libs.parsers.raise_parser_error()

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xff in position 0: invalid s
tart byte
```



pie chart

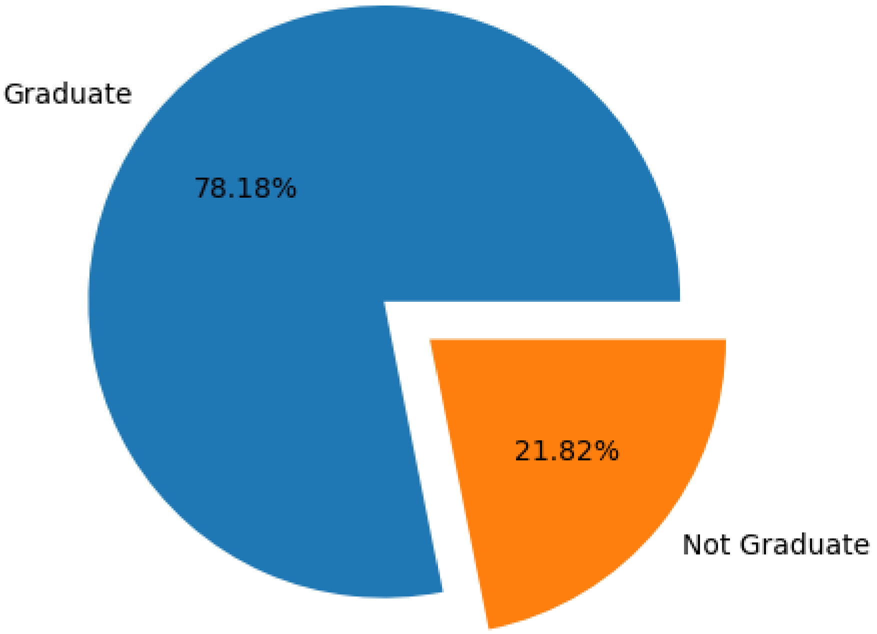
In [33]:
```
import matplotlib.pyplot as plt
keys=loan_df['Education'].value_counts().keys()
values=loan_df['Education'].value_counts().values
plt.pie(values,labels=keys,autopct="%0.2f%%")
plt.show()
```



if you want to sepreter to all the parts

In [72]:

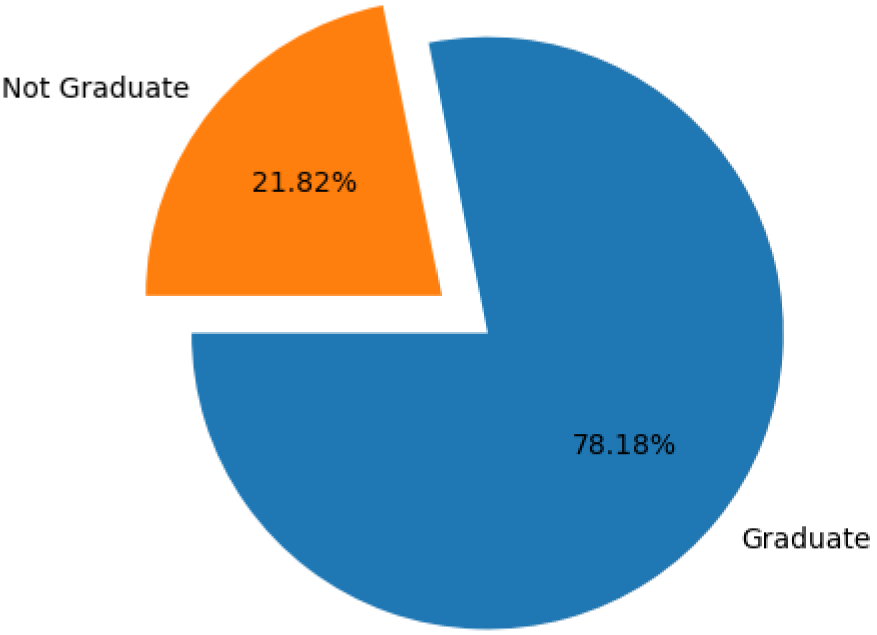
```
keys=loan_df['Education'].value_counts().keys()
values=loan_df['Education'].value_counts().values
plt.pie(values,labels=keys,explode=[0.1]*len(keys),autopct='%0.2f%%')
plt.show()
```



if you roaed the figure

In [73]:

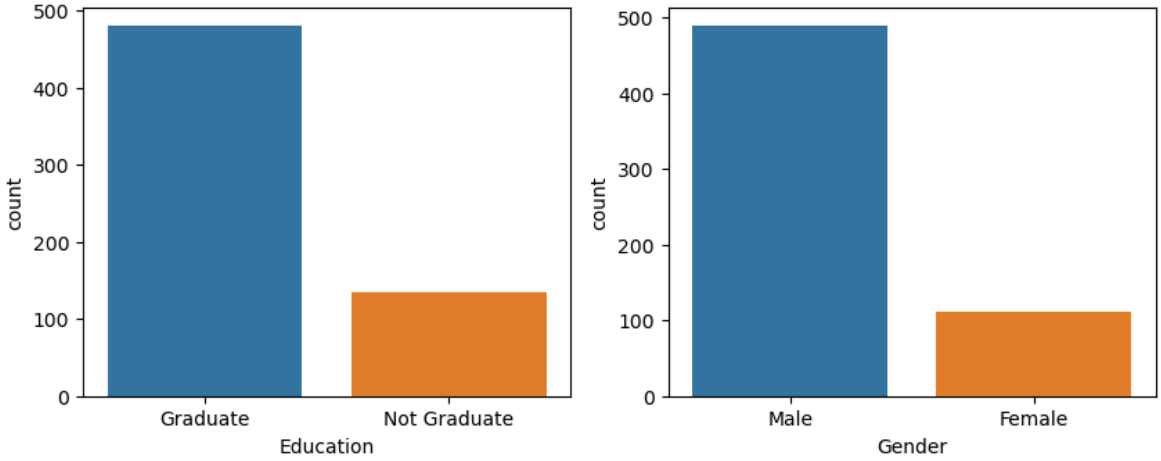
```
keys=loan_df['Education'].value_counts().keys()
values=loan_df['Education'].value_counts().values
plt.pie(values,labels=keys,explode=[0.1]*len(keys),autopct='%0.2f%%',startangle=
plt.show()
```



If you want 2 figurest at a one frame

In [74]:

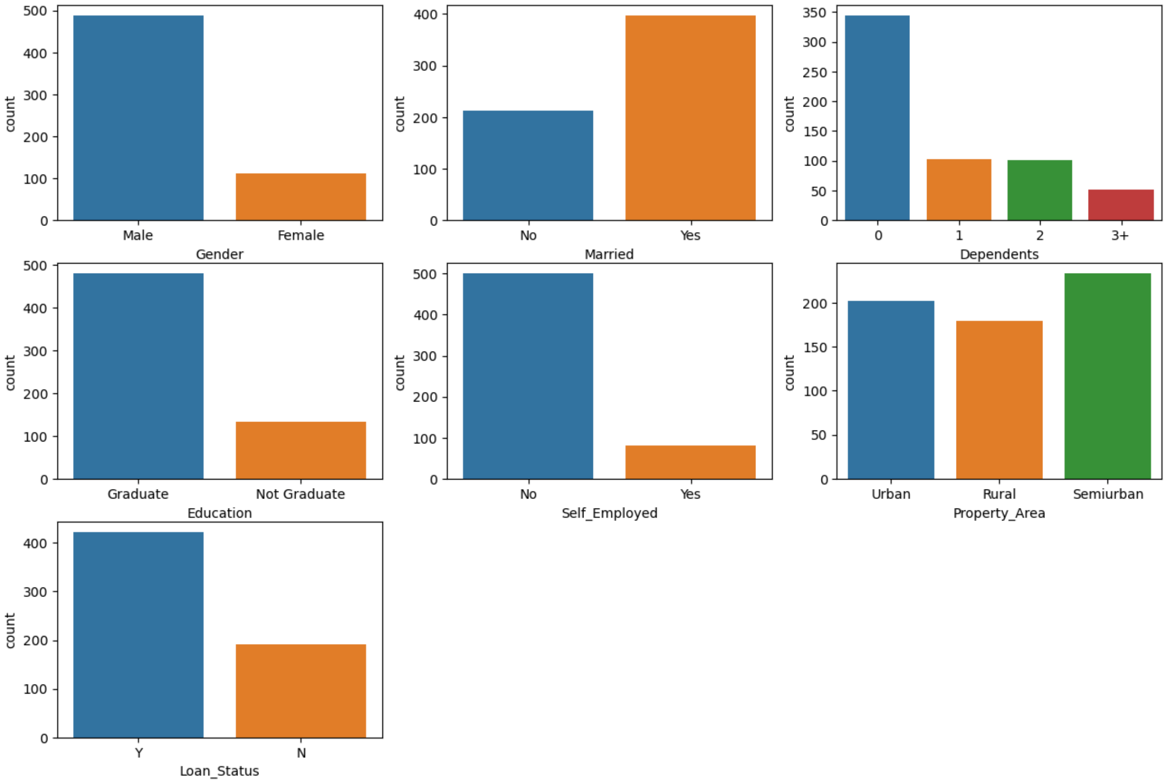
```
plt.figure(figsize=(10,8))

plt.subplot(2,2,1)
sns.countplot(x='Education', data=loan_df)

plt.subplot(2,2,2)
sns.countplot(x='Gender',data=loan_df)

plt.show()
```



we can take a all figure in the one frame

In [75]:

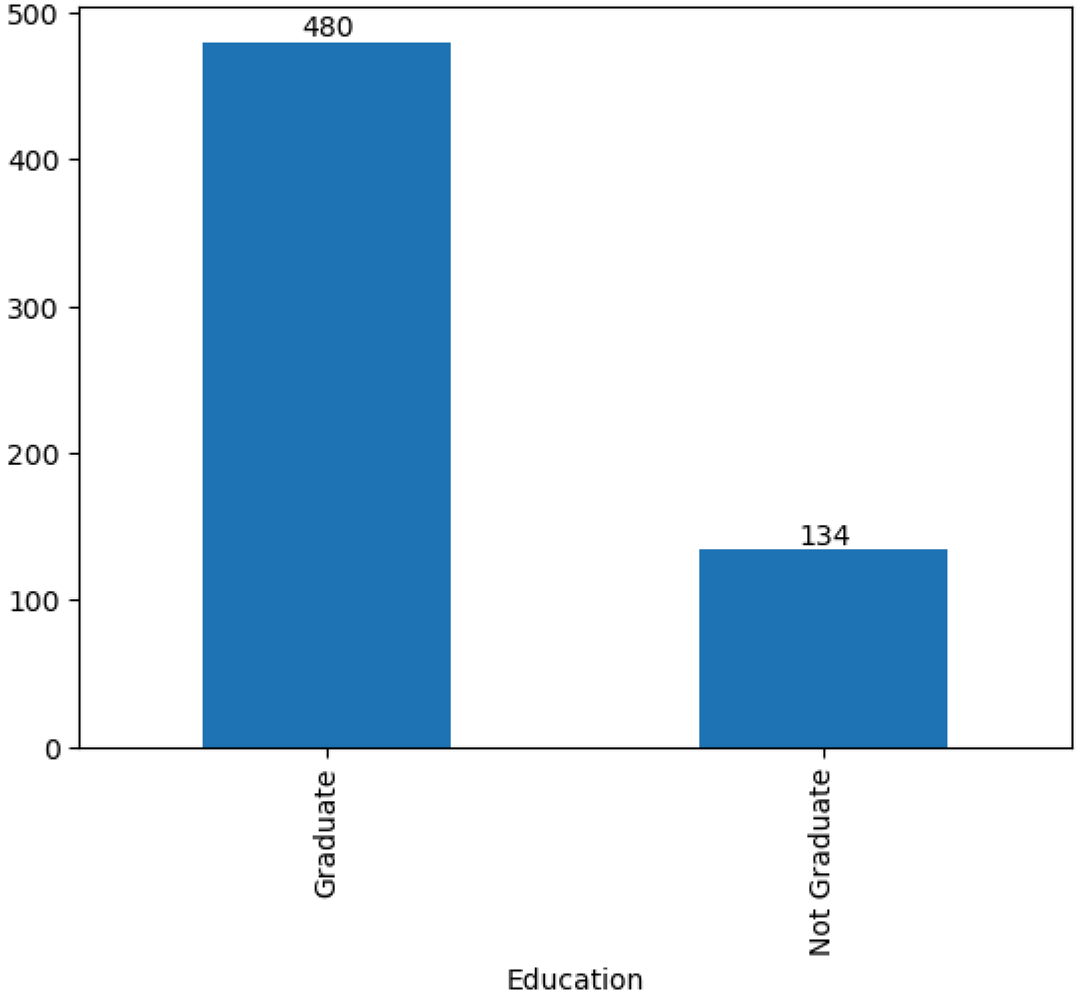
```
plt.figure(figsize=(15,10))
cols=cat_columns[1:]
for i in range(len(cols)):
    plt.subplot(3,3,i+1)
    sns.countplot(x=cols[i],data=loan_df)
```



in that figure the a how much counts in that figure

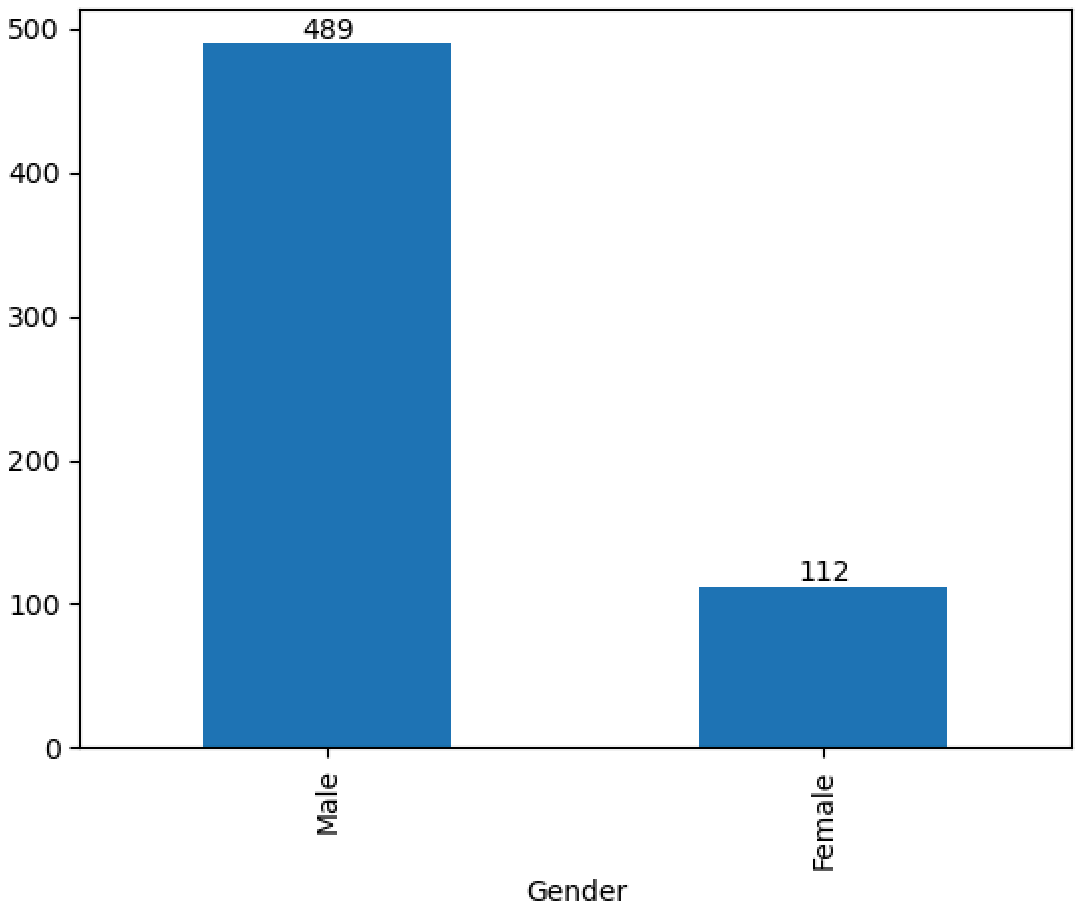
In [76]:

```
edu=loan_df['Education'].value_counts()
ax=edu.plot(kind='bar')
ax.bar_label(ax.containers[0])
plt.show()
```

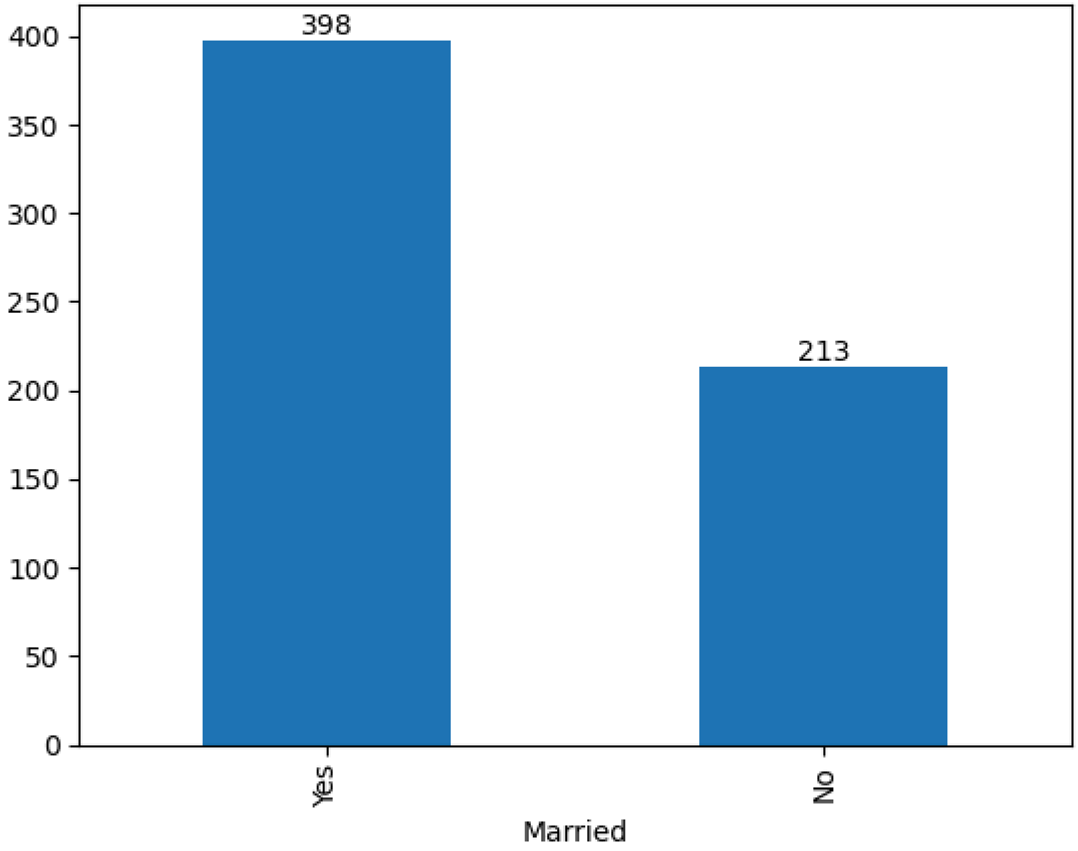


In [77]:

```
gen=loan_df['Gender'].value_counts()
ax=gen.plot(kind='bar')
ax.bar_label(ax.containers[0])
plt.show()
```



In [78]:
```
married=loan_df['Married'].value_counts()
ax=married.plot(kind='bar')
ax.bar_label(ax.containers[0])
plt.show()
```



Describe method Count,Min,Max,Mean,Median,Std,p_25,p_50,p_75

In [79]:

```
df1=pd.DataFrame()

for i in num_columns:
    Count=len(loan_df[i])
    Min=min(loan_df[i])
    Max=max(loan_df[i])
    Mean=round(loan_df[i].mean(),2)
    Median=round(loan_df[i].median(),2)
    Std=round(loan_df[i].std(),2)
    p_25=round(np.percentile(loan_df[i],25),2)
    p_50=round(np.percentile(loan_df[i],50),2)
    p_75=round(np.percentile(loan_df[i],75),2)
    values=[Count,Min,Max,Mean,Median,Std,p_25,p_50,p_75]
    index=['count','min','max','mean','median','std','25p','50p','75p']
    cols=[i]
    df2=pd.DataFrame(values,index=index,columns=cols)
    df1=pd.concat([df1,df2],axis=1)
df1
```

Out[79]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit
count	614.00	614.00	614.00	614.00	
min	150.00	0.00	NaN	12.00	
max	81000.00	41667.00	NaN	480.00	
mean	5403.46	1621.25	146.41	342.00	
median	3812.50	1188.50	128.00	360.00	
std	6109.04	2926.25	85.59	65.12	
25p	2877.50	0.00	NaN	NaN	
50p	3812.50	1188.50	NaN	NaN	
75p	5795.00	2297.25	NaN	NaN	

In [80]:

```
Count=len(loan_df['ApplicantIncome'])
Min=min(loan_df['ApplicantIncome'])
Max=max(loan_df['ApplicantIncome'])
Mean=round(loan_df['ApplicantIncome'].mean(),2)
Median=round(loan_df['ApplicantIncome'].median(),2)
Std=round(loan_df['ApplicantIncome'].std(),2)
p_25=round(np.percentile(loan_df['ApplicantIncome'],25),2)
p_50=round(np.percentile(loan_df['ApplicantIncome'],50),2)
p_75=round(np.percentile(loan_df['ApplicantIncome'],75),2)
values=[Count,Min,Max,Mean,Median,Std,p_25,p_50,p_75]
index=['Count','Min','Max','Mean','Median','Std','25%','50%','75%']
cols=['ApplicantIncome']
pd.DataFrame(values,index=index,columns=cols)
```

Out[80]:

	ApplicantIncome
Count	614.00
Min	150.00
Max	81000.00
Mean	5403.46
Median	3812.50
Std	6109.04
25%	2877.50
50%	3812.50
75%	5795.00

In [81]:

```
Count=len(loan_df['CoapplicantIncome'])
Min=min(loan_df['CoapplicantIncome'])
Max=max(loan_df['CoapplicantIncome'])
Mean=round(loan_df['CoapplicantIncome'].mean(),2)
Median=round(loan_df['CoapplicantIncome'].median(),2)
Std=round(loan_df['CoapplicantIncome'].std(),2)
p_25=round(np.percentile(loan_df['CoapplicantIncome'],25),2)
p_50=round(np.percentile(loan_df['CoapplicantIncome'],25),2)
p_75=round(np.percentile(loan_df['CoapplicantIncome'],25),2)
values=[Count, Min, Max, Mean, Median, Std, p_25, p_50, p_75]
index=['Count','Min','Max','Mean','Median','Std','25%','50%','75%']
cols=['CoapplicantIncome']
pd.DataFrame(values, index=index, columns= cols)
```

Out[81]:

	CoapplicantIncome
Count	614.00
Min	0.00
Max	41667.00
Mean	1621.25
Median	1188.50
Std	2926.25
25%	0.00
50%	0.00
75%	0.00

In [82]:

```
loan_df.describe()
```

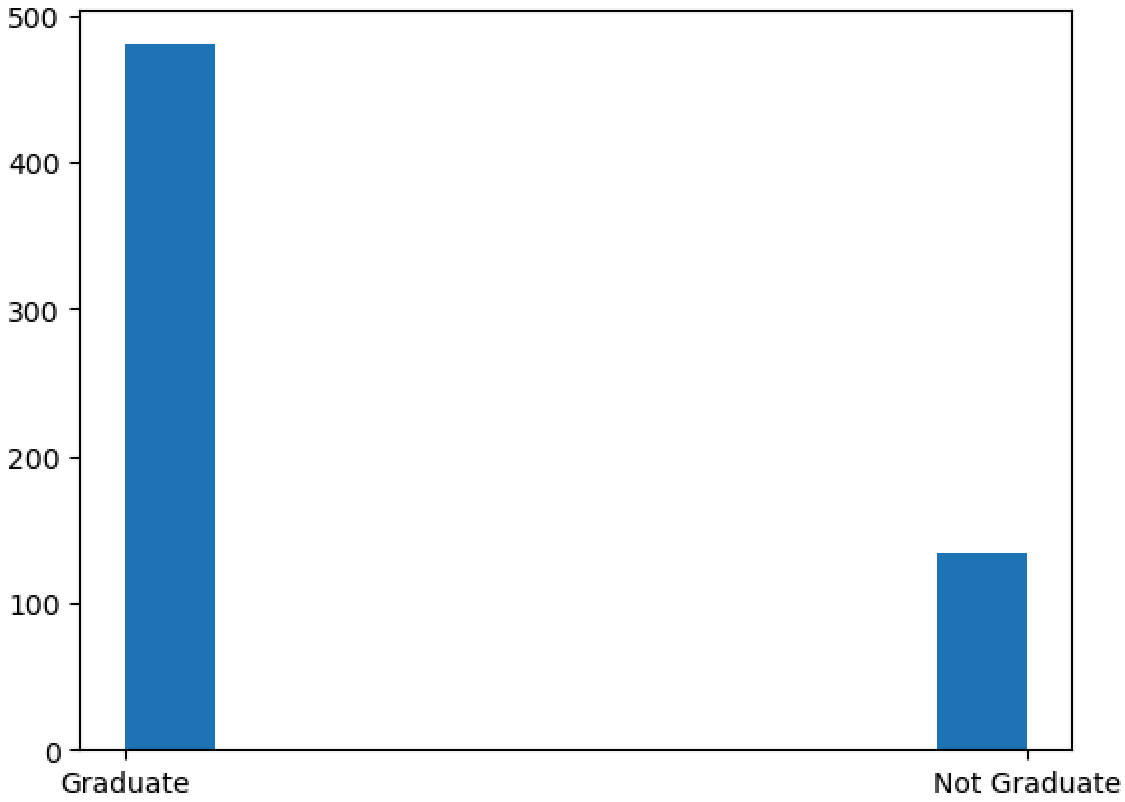
Out[82]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_H
count	614.000000	614.000000	592.000000	600.00000	564.(
mean	5403.459283	1621.245798	146.412162	342.00000	0.8
std	6109.041673	2926.248369	85.587325	65.12041	0.3
min	150.000000	0.000000	9.000000	12.00000	0.(
25%	2877.500000	0.000000	100.000000	360.00000	1.(
50%	3812.500000	1188.500000	128.000000	360.00000	1.(
75%	5795.000000	2297.250000	168.000000	360.00000	1.(
max	81000.000000	41667.000000	700.000000	480.00000	1.(

Histpgram

In [83]:

```
education=loan_df['Education']
plt.hist(education,bins=10)
plt.show()
```

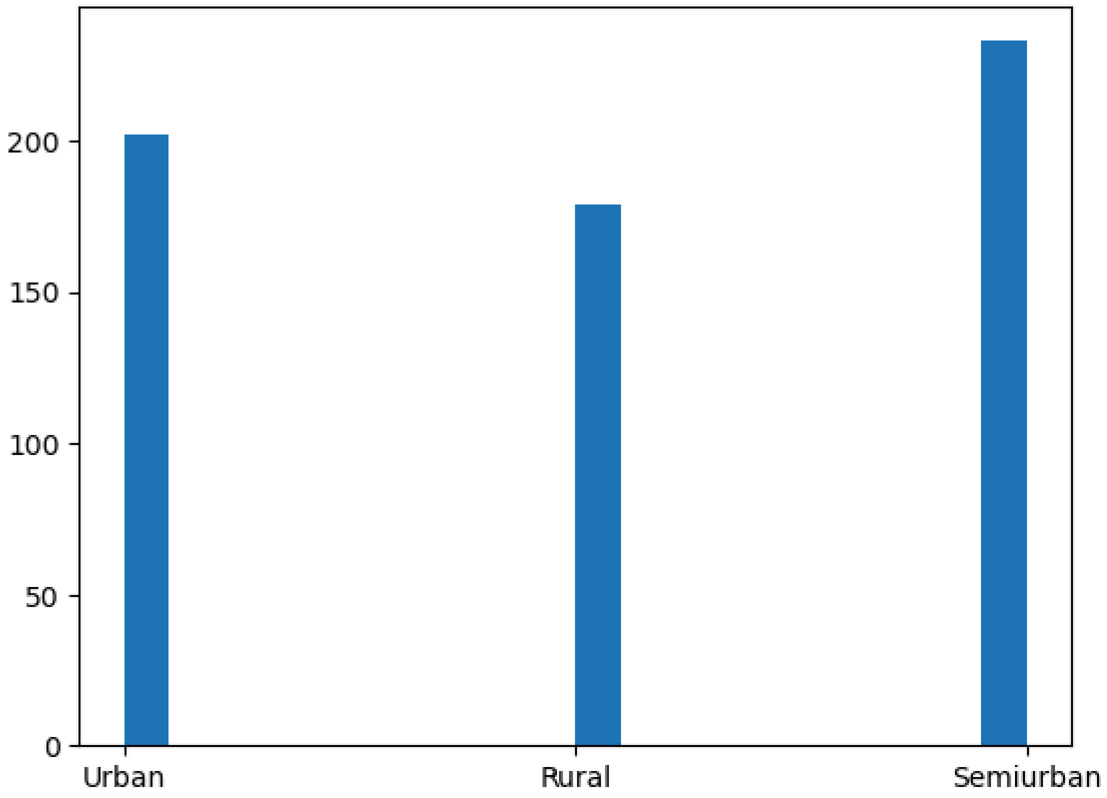


In [84]:

```
plt.hist(loan_df['Property_Area'],bins=20)
```

Out[84]:

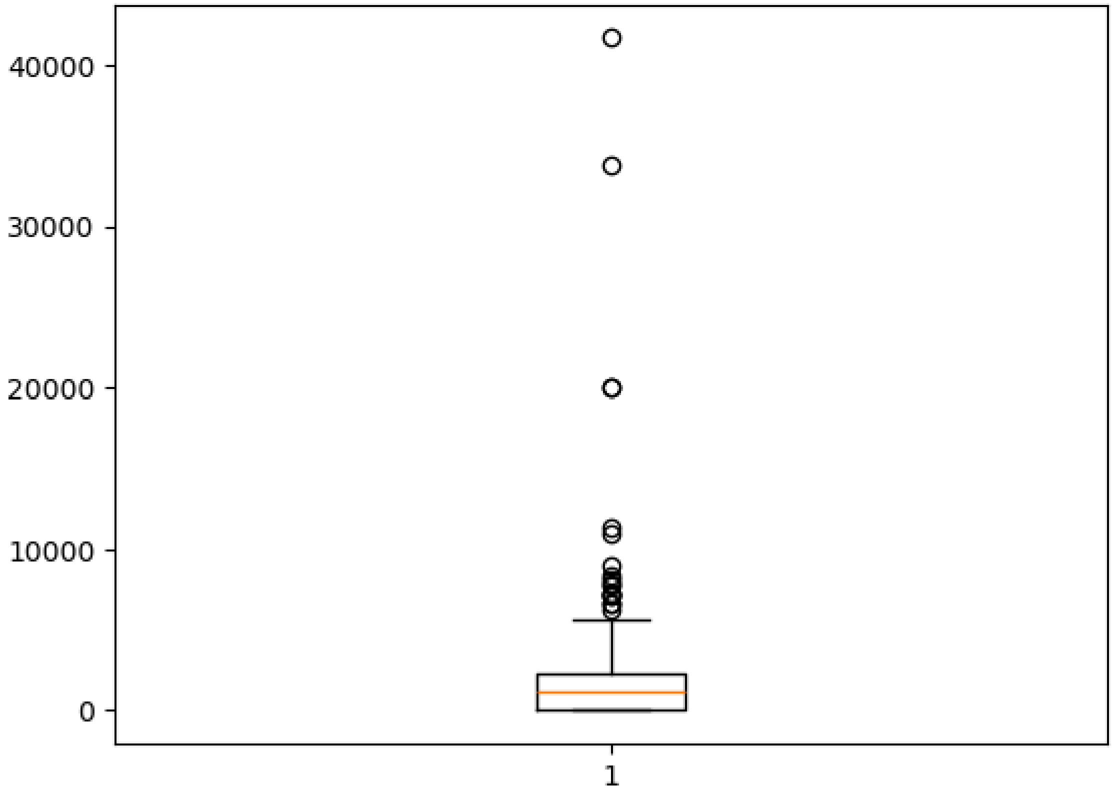
```
(array([202.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 179.,
          0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 233.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. , 1.1, 1.2,
        1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2. ]),
 <BarContainer object of 20 artists>)
```



boxplot

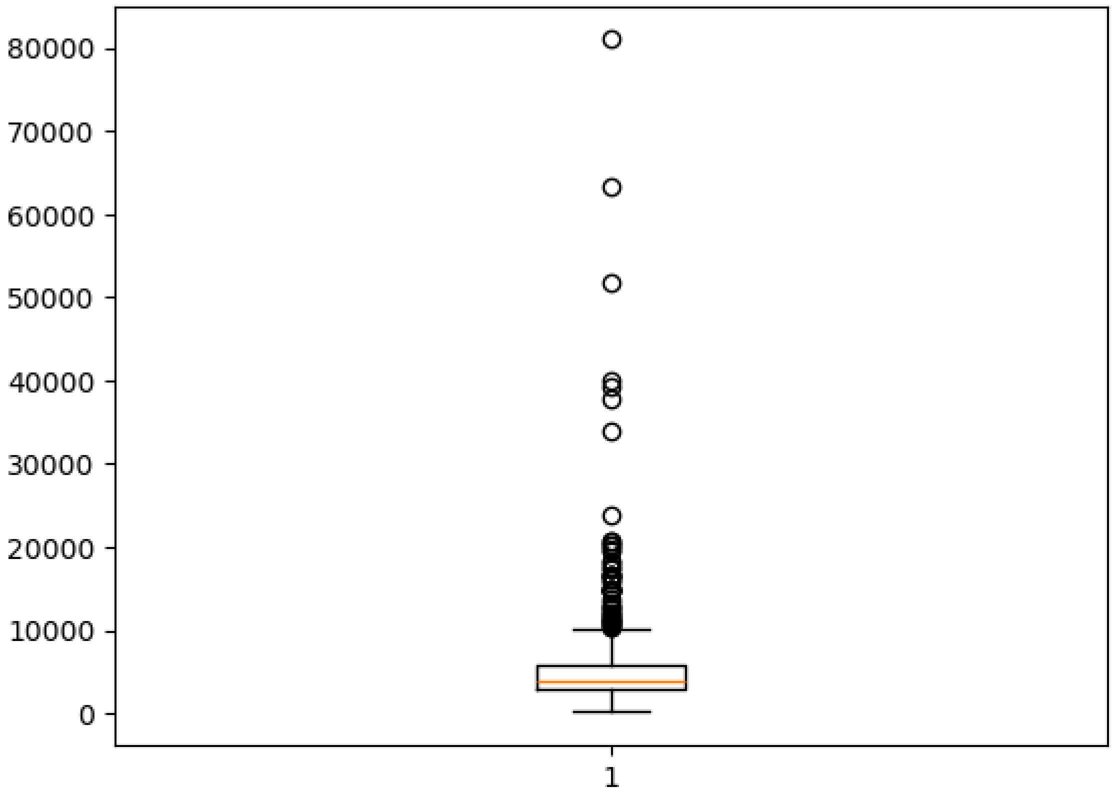
In [85]:

```
plt.boxplot(loan_df['CoapplicantIncome'])
plt.show()
```

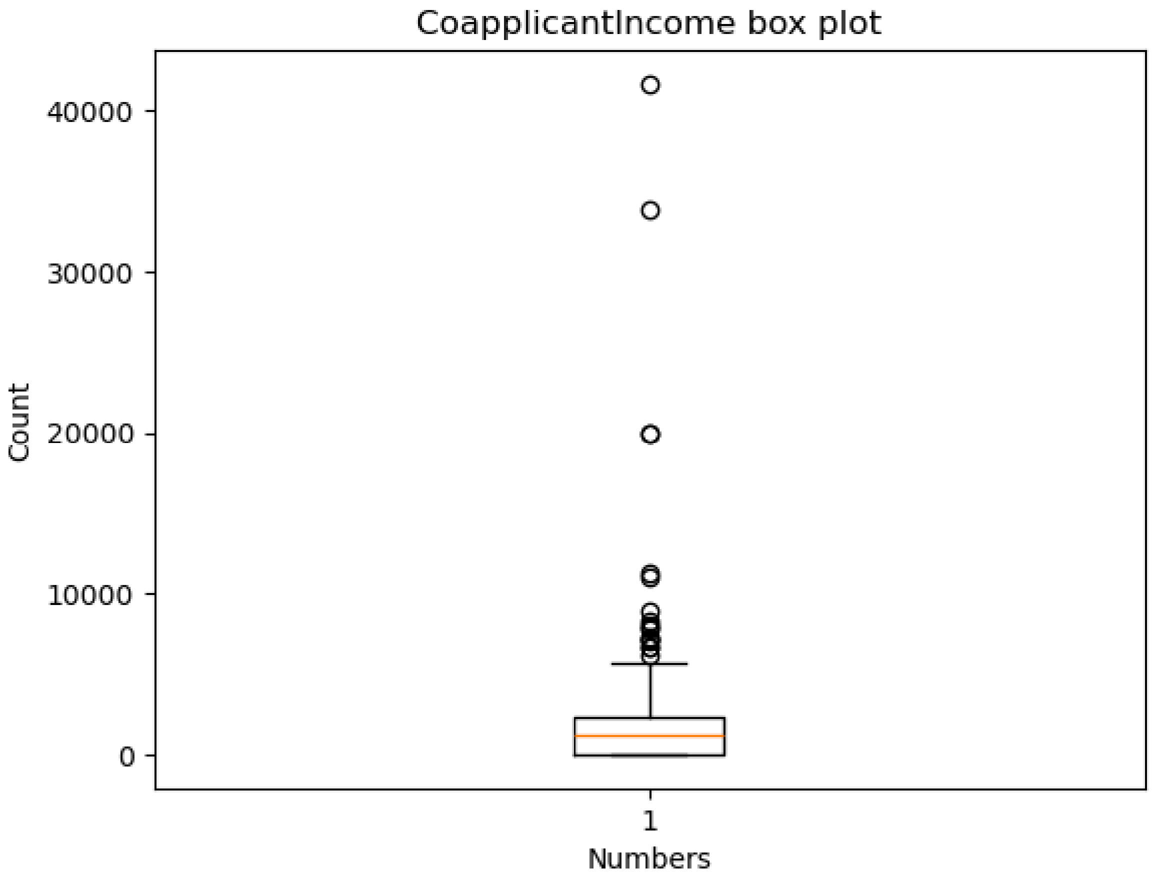


In [86]:

```
plt.boxplot(loan_df['ApplicantIncome'])
plt.show()
```

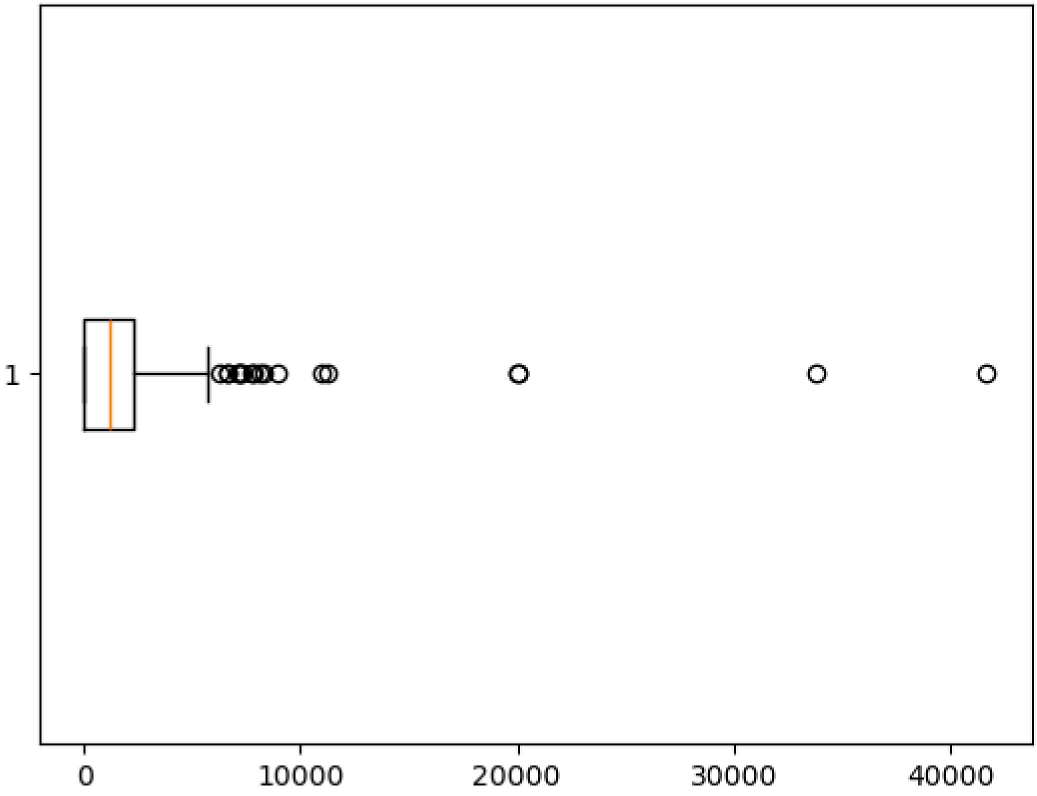


```
In [87]: plt.boxplot(loan_df['CoapplicantIncome'])
         plt.title('CoapplicantIncome box plot')
         plt.xlabel('Numbers')
         plt.ylabel('Count')
         plt.show()
```



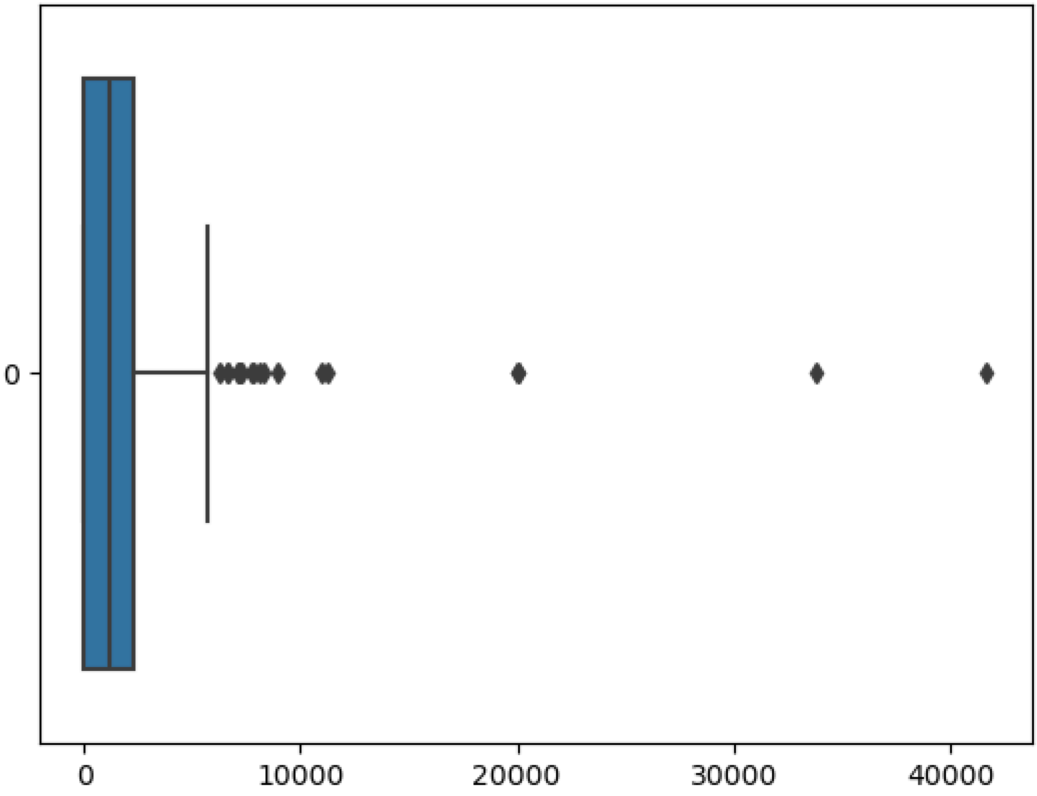
In [88]:

```
plt.boxplot(loan_df['CoapplicantIncome'],vert=False)
plt.show()
```



In [89]:

```
sns.boxplot(loan_df['CoapplicantIncome'],orient='h')
plt.show()
```

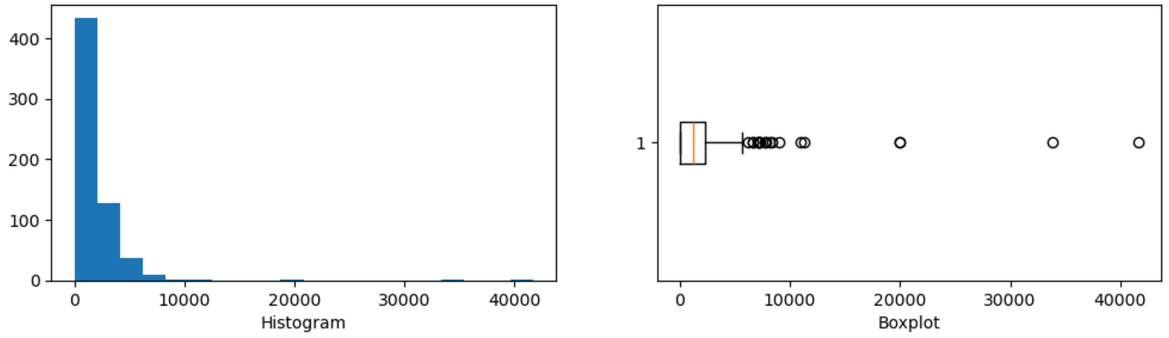


plot the histogram plot and box plot side by side

In [92]:

```
Copp_data=loan_df['CoapplicantIncome']
plt.figure(figsize=(12,3))
plt.suptitle('Comparision of Histogram With Boxplot')
plt.subplot(1,2,1)
plt.hist(Copp_data,bins=20)
plt.xlabel('Histogram')
plt.subplot(1,2,2)
plt.boxplot(Copp_data,vert=False)
plt.xlabel('Boxplot')
plt.show()
```

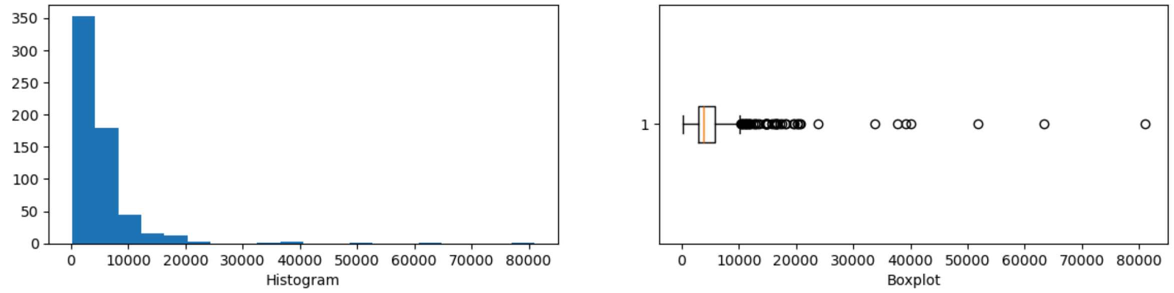
Comparision of Histogram With Boxplot



In [93]:

```
Appl_data=loan_df['ApplicantIncome']
plt.figure(figsize=(14,3))
plt.suptitle('Comparision of Histgram With Boxplot')
plt.subplot(1,2,1)
plt.hist(Appl_data,bins=20)
plt.xlabel('Histogram')
plt.subplot(1,2,2)
plt.boxplot(Appl_data,vert=False)
plt.xlabel('Boxplot')
plt.show()
```

Comparision of Histgram With Boxplot



In [94]:

```
Appl_data=loan_df['ApplicantIncome']
q1=np.percentile(Appl_data,25)
q2=np.percentile(Appl_data,50)
q3=np.percentile(Appl_data,75)

IQR=q3-q1

lb=q1-1.5*IQR
ub=q3+1.5*IQR

con1=Appl_data<lb
con2=Appl_data>ub
con3=con1|con2
```

```
Outliers_data=Appl_data[con3]
Outliers_data
```

Out[94]:

```
9      12841
34     12500
54     11500
67     10750
102    13650
106    11417
115    14583
119    10408
126    23803
128    10513
130    20166
138    14999
144    11757
146    14866
155    39999
171    51763
183    33846
185    39147
191    12000
199    11000
254    16250
258    14683
271    11146
278    14583
284    20667
308    20233
324    15000
333    63337
369    19730
370    15759
409    81000
424    14880
432    12876
438    10416
443    37719
467    16692
475    16525
478    16667
483    10833
487    18333
493    17263
506    20833
509    13262
525    17500
533    11250
534    18165
561    19484
572    16666
594    16120
604    12000
Name: ApplicantIncome, dtype: int64
```

In [95]:

```
print(ub)
```

```
10171.25
```

In [96]:

```
outlier_df=loan_df[con3]
outlier_df
```

Out[96]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantInco
9	LP001020	Male	Yes	1	Graduate	No	128
34	LP001100	Male	No	3+	Graduate	No	125
54	LP001186	Female	Yes	1	Graduate	Yes	115
67	LP001233	Male	Yes	1	Graduate	No	107
102	LP001350	Male	Yes	NaN	Graduate	No	130
106	LP001369	Male	Yes	2	Graduate	No	114
115	LP001401	Male	Yes	1	Graduate	No	145
119	LP001422	Female	No	0	Graduate	No	104
126	LP001448	NaN	Yes	3+	Graduate	No	238
128	LP001451	Male	Yes	1	Graduate	Yes	105
130	LP001469	Male	No	0	Graduate	Yes	201
138	LP001492	Male	No	0	Graduate	No	149
144	LP001508	Male	Yes	2	Graduate	No	117
146	LP001516	Female	Yes	2	Graduate	No	148
155	LP001536	Male	Yes	3+	Graduate	No	399
171	LP001585	NaN	Yes	3+	Graduate	No	517
183	LP001637	Male	Yes	1	Graduate	No	338
185	LP001640	Male	Yes	0	Graduate	Yes	391
191	LP001656	Male	No	0	Graduate	No	120
199	LP001673	Male	No	0	Graduate	Yes	110
254	LP001844	Male	No	0	Graduate	Yes	162
258	LP001859	Male	Yes	0	Graduate	No	140
271	LP001891	Male	Yes	0	Graduate	No	111
278	LP001907	Male	Yes	0	Graduate	No	145
284	LP001922	Male	Yes	0	Graduate	No	200
308	LP001996	Male	No	0	Graduate	No	202
324	LP002065	Male	Yes	3+	Graduate	No	150
333	LP002101	Male	Yes	0	Graduate	NaN	633
369	LP002191	Male	Yes	0	Graduate	No	197
370	LP002194	Female	No	0	Graduate	Yes	157
409	LP002317	Male	Yes	3+	Graduate	No	810
424	LP002364	Male	Yes	0	Graduate	No	148
432	LP002386	Male	No	0	Graduate	NaN	128

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantInco
438	LP002403	Male	No	0	Graduate	Yes	104
443	LP002422	Male	No	1	Graduate	No	37
467	LP002501	NaN	Yes	0	Graduate	No	16
475	LP002527	Male	Yes	2	Graduate	Yes	16
478	LP002531	Male	Yes	1	Graduate	Yes	16
483	LP002541	Male	Yes	0	Graduate	No	10
487	LP002547	Male	Yes	1	Graduate	No	18
493	LP002582	Female	No	0	Not Graduate	Yes	17
506	LP002624	Male	Yes	0	Graduate	No	20
509	LP002634	Female	No	1	Graduate	No	13
525	LP002699	Male	Yes	2	Graduate	Yes	17
533	LP002729	Male	No	1	Graduate	No	11
534	LP002731	Female	No	0	Not Graduate	Yes	18
561	LP002813	Female	Yes	1	Graduate	Yes	19
572	LP002855	Male	Yes	2	Graduate	No	16
594	LP002938	Male	Yes	0	Graduate	Yes	16
604	LP002959	Female	Yes	1	Graduate	No	12

In [97]:

```
Appl_data=loan_df['ApplicantIncome']
q1=np.percentile(Appl_data,25)
q2=np.percentile(Appl_data,50)
q3=np.percentile(Appl_data,75)

IQR=q3-q1

lb=q1-1.5*IQR
ub=q3+1.5*IQR

con1=Appl_data>lb
con2=Appl_data<ub
con3=con1 & con2

Non_outliers_data=Appl_data[con3]
Non_outliers_data
```

Out[97]:

```
0      5849
1      4583
2      3000
3      2583
4      6000
       ... 
609    2900
610    4106
611    8072
612    7583
613    4583
Name: ApplicantIncome, Length: 564, dtype: int64
```

In [98]:

```
print(ub)
```

```
10171.25
```

In [99]:

```
non_outlier_df=loan_df[con3]
non_outlier_df
```

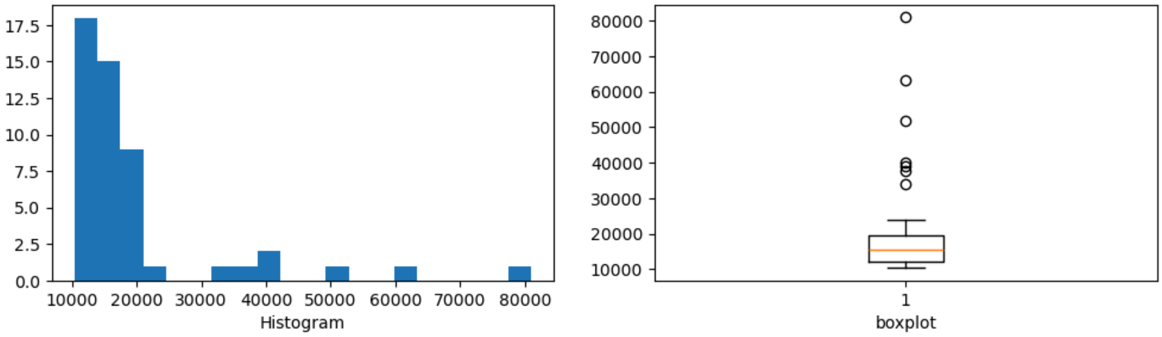
Out[99]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantInco
0	LP001002	Male	No	0	Graduate	No	58
1	LP001003	Male	Yes	1	Graduate	No	45
2	LP001005	Male	Yes	0	Graduate	Yes	30
3	LP001006	Male	Yes	0	Not Graduate	No	25
4	LP001008	Male	No	0	Graduate	No	60
...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	29
610	LP002979	Male	Yes	3+	Graduate	No	41
611	LP002983	Male	Yes	1	Graduate	No	80
612	LP002984	Male	Yes	2	Graduate	No	75
613	LP002990	Female	No	0	Graduate	Yes	45

564 rows × 13 columns

In [100…

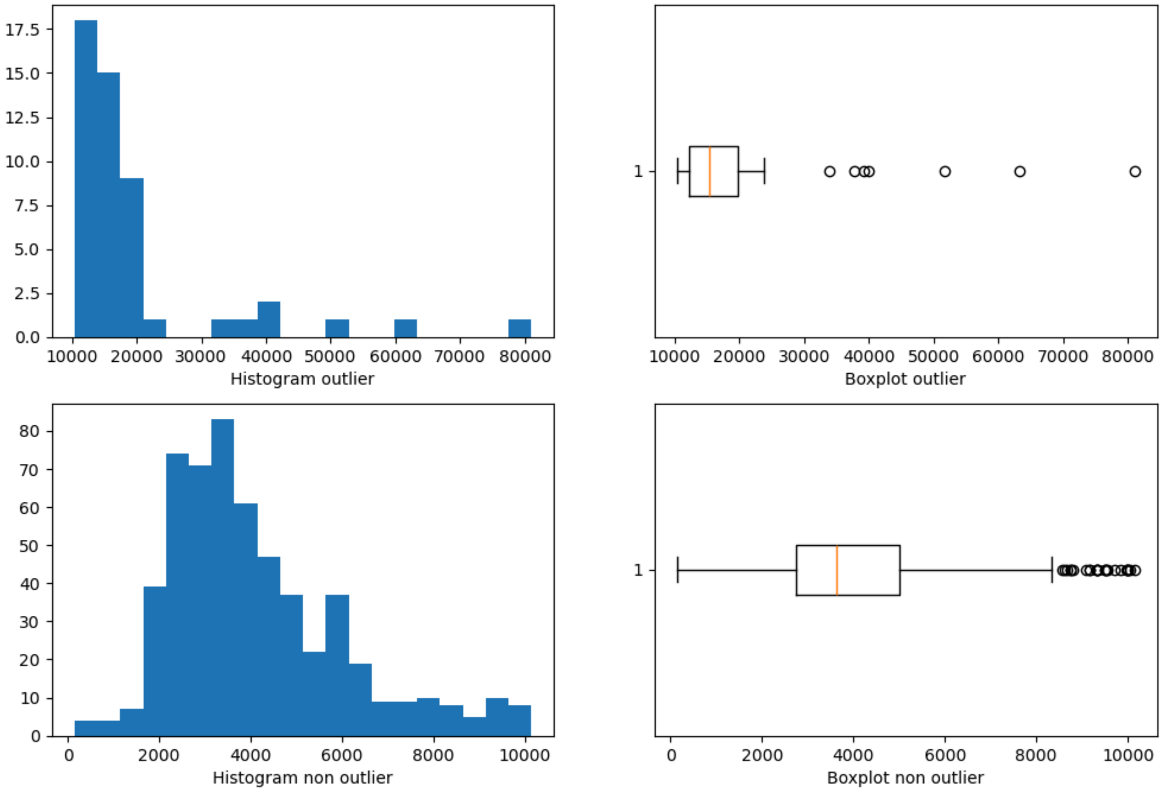
```
Appl_outlier_df=outlier_df['ApplicantIncome']
plt.figure(figsize=(12,3))
plt.subplot(1,2,1).hist(Appl_outlier_df,bins=20)
plt.xlabel('Histogram')
plt.subplot(1,2,2).boxplot(Appl_outlier_df,vert=True)
plt.xlabel('boxplot')
plt.show()
```



In [101…

```
Appl_outliers_df=outlier_df['ApplicantIncome']
Appl_data=non_outlier_df['ApplicantIncome']

#################### Non outliers #####################################
plt.figure(figsize=(12,8))
plt.suptitle('Comparision of Histogram With Boxplot')
plt.subplot(2,2,1).hist(Appl_outliers_df,bins=20)  # we can directly take in the
plt.xlabel('Histogram outlier')
plt.subplot(2,2,2).boxplot(Appl_outliers_df,vert=False)
plt.xlabel('Boxplot outlier')

#################### Entire data #############################################
plt.subplot(2,2,3).hist(Appl_data,bins=20)  # we can directly take in the one li
plt.xlabel('Histogram non outlier')
plt.subplot(2,2,4).boxplot(Appl_data,vert=False)
plt.xlabel('Boxplot non outlier')
plt.show()
```

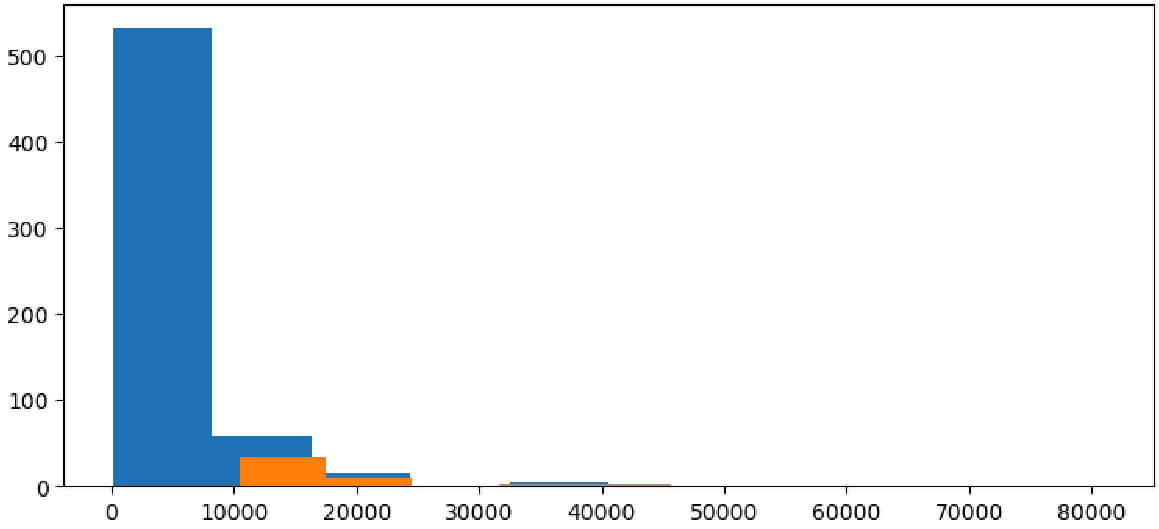
Comparision of Histogram With Boxplot



In [102…

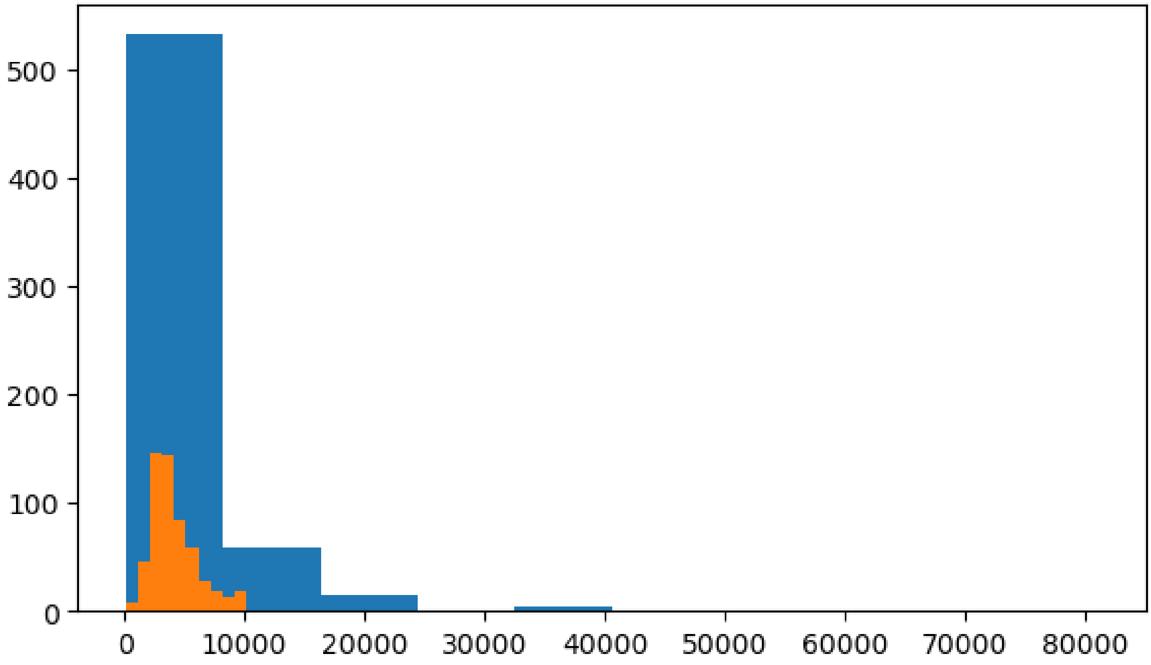
```
Appl_outlier_df=outlier_df['ApplicantIncome']
Appl_data=loan_df['ApplicantIncome']
```

```
plt.figure(figsize=(9,4))
plt.hist(Appl_data)
plt.hist(Appl_outlier_df)
plt.show()
```



In [103…

```
Appl_non_outlier_df=non_outlier_df['ApplicantIncome']
Appl_data=loan_df['ApplicantIncome']

plt.figure(figsize=(7,4))
plt.hist(Appl_data)
plt.hist(Appl_non_outlier_df)
plt.show()
```



In [104…

```
loan_df
```

Out[104…

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantInco
0	LP001002	Male	No	0	Graduate	No	58
1	LP001003	Male	Yes	1	Graduate	No	45
2	LP001005	Male	Yes	0	Graduate	Yes	30
3	LP001006	Male	Yes	0	Not Graduate	No	25
4	LP001008	Male	No	0	Graduate	No	60
...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	29
610	LP002979	Male	Yes	3+	Graduate	No	41
611	LP002983	Male	Yes	1	Graduate	No	80
612	LP002984	Male	Yes	2	Graduate	No	75
613	LP002990	Female	No	0	Graduate	Yes	45

614 rows × 13 columns

In [105…

```
con1=loan_df['Gender']=='Male'
con2=loan_df['Education']=='Graduate'
con3=con1&con2
len(loan_df[con3])
```

Out[105…

```
376
```

In [109…

```
loan_df['Gender'].unique()
loan_df['Gender'].value_counts().keys()
```

Out[109…

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

In [110…

```
loan_df['Education'].unique()
loan_df['Education'].value_counts().keys()
```

Out[110…

```
Index(['Graduate', 'Not Graduate'], dtype='object', name='Education')
```

In [107…

```
genders=loan_df['Gender'].unique()
for i in genders:
    con1=loan_df['Gender']==i
    con2=loan_df['Education']=='Graduate'
    con3=con1 & con2
    count=len(loan_df[con3])
    print(f'the number of certified loan from {i} is {count}')
```

```
the number of certified loan from Male is 376
the number of certified loan from Female is 92
the number of certified loan from nan is 0
```

In [121…

```
genders=loan_df['Gender'].unique()
Graduate_list,Not_Graduate_list=[],[]
for i in genders:
    con1=loan_df['Gender']==i
    con2=loan_df['Education']=='Graduate'
    con3=loan_df['Education']=='Not Graduate'
    gradu_count=con1 & con2
    non_gradu_count=con1 & con3
    gradu_count=len(loan_df[gradu_count])
    non_gradu_count=len(loan_df[non_gradu_count])
    print(f'the number of Graduate loan from {i} is {gradu_count}')
    print(f'the number of Not Graduate loan from {i} is {non_gradu_count}')
    Graduate_list.append(gradu_count)
    Not_Graduate_list.append(non_gradu_count)
```

```
the number of Graduate loan from Male is 376
the number of Not Graduate loan from Male is 113
the number of Graduate loan from Female is 92
the number of Not Graduate loan from Female is 20
the number of Graduate loan from nan is 0
the number of Not Graduate loan from nan is 0
```

In [125…

```
genders
Graduate_list
cols=['Gender','Education']
df1=pd.DataFrame(zip(genders,Graduate_list),columns=cols)
df1
```

Out[125…

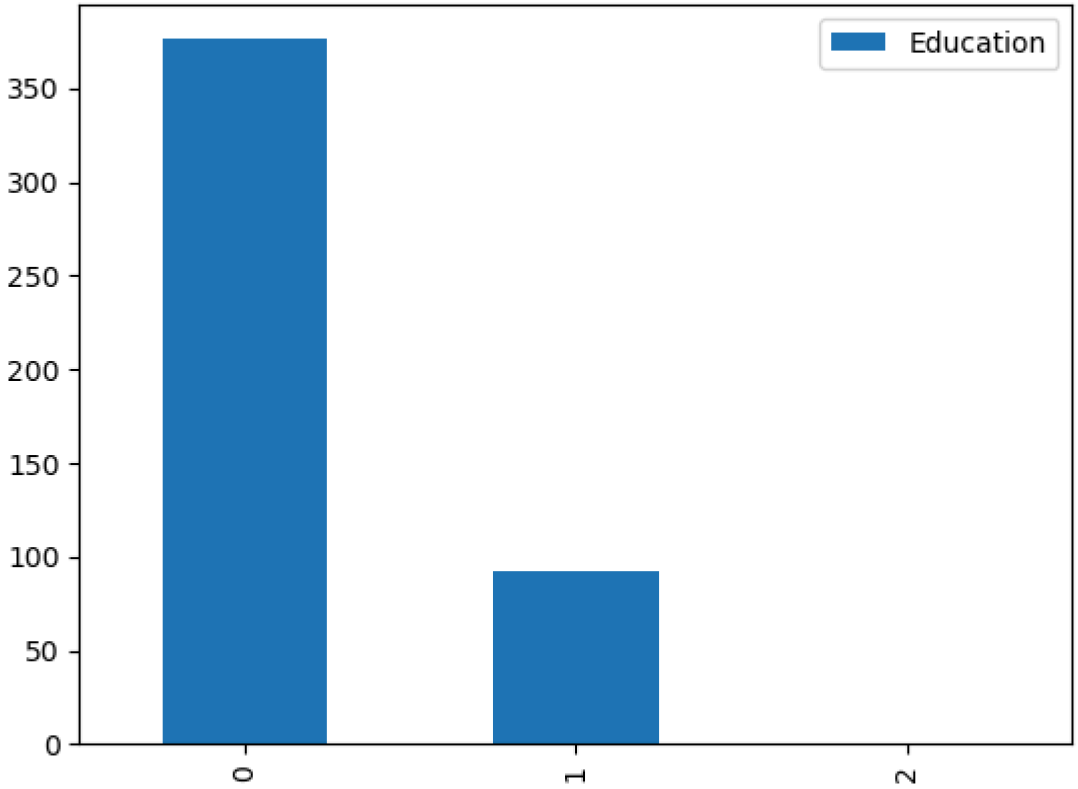
	Gender	Education
0	Male	376
1	Female	92
2	NaN	0

In [127…

```
df1.plot(kind='bar')
```

Out[127…

```
<Axes: >
```



In [131…

```
genders
Not_Graduate_list
cols=['Gender','Education']
df2=pd.DataFrame(zip(genders,Not_Graduate_list),columns=cols)
df2
```

Out[131…

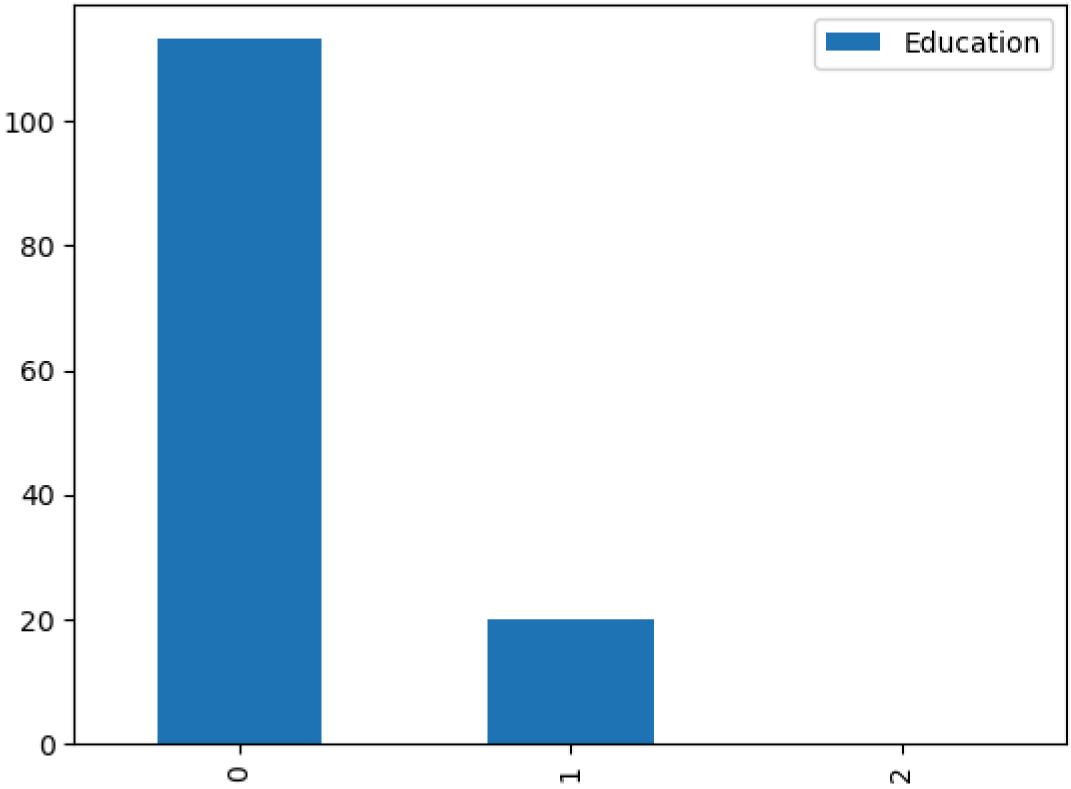
	Gender	Education
0	Male	113
1	Female	20
2	NaN	0

In [132…

```
df2.plot(kind='bar')
```

Out[132…

```
<Axes: >
```



In [134…

```
cols=['Gender','Education']
df3=pd.DataFrame(zip(Graduate_list,Not_Graduate_list),index=genders,
                 columns=cols)
df3
```

Out[134…

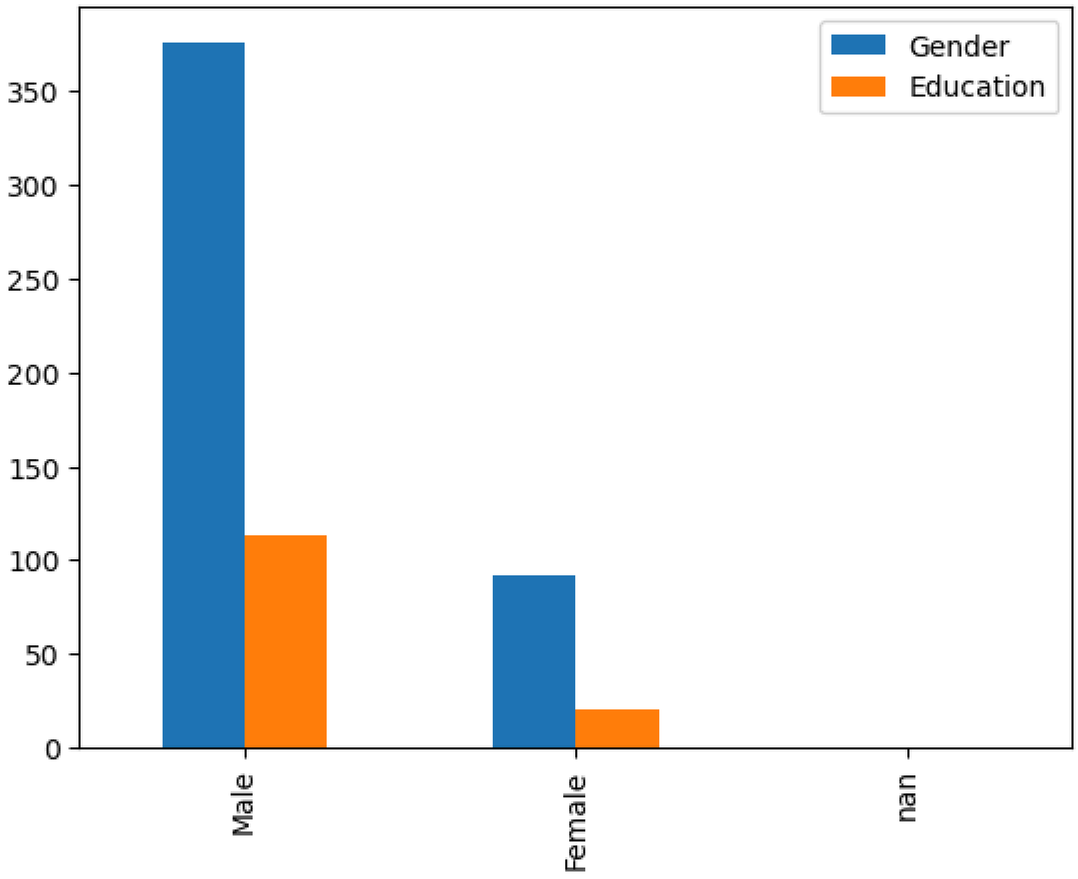
	Gender	Education
Male	376	113
Female	92	20
NaN	0	0

In [135…

```
df3.plot(kind='bar')
```

Out[135…

```
<Axes: >
```

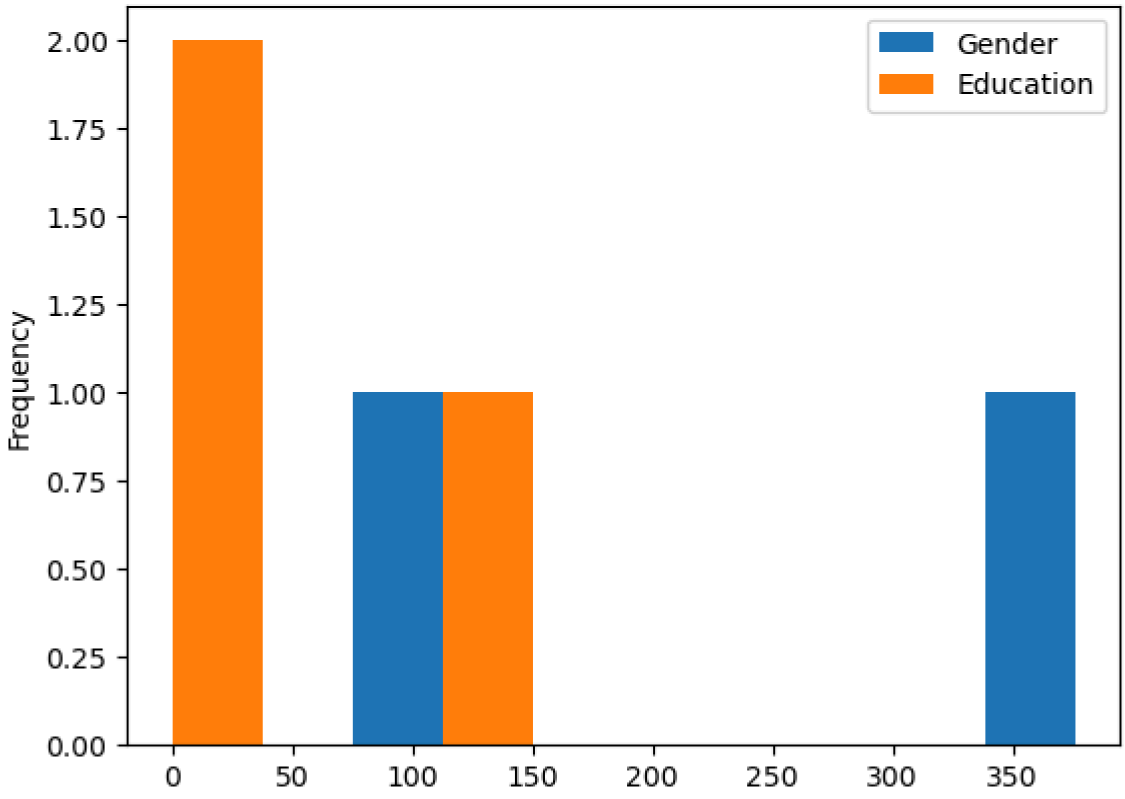


In [136…

```
df3.plot(kind='hist')
```

Out[136…

```
<Axes: ylabel='Frequency'>
```

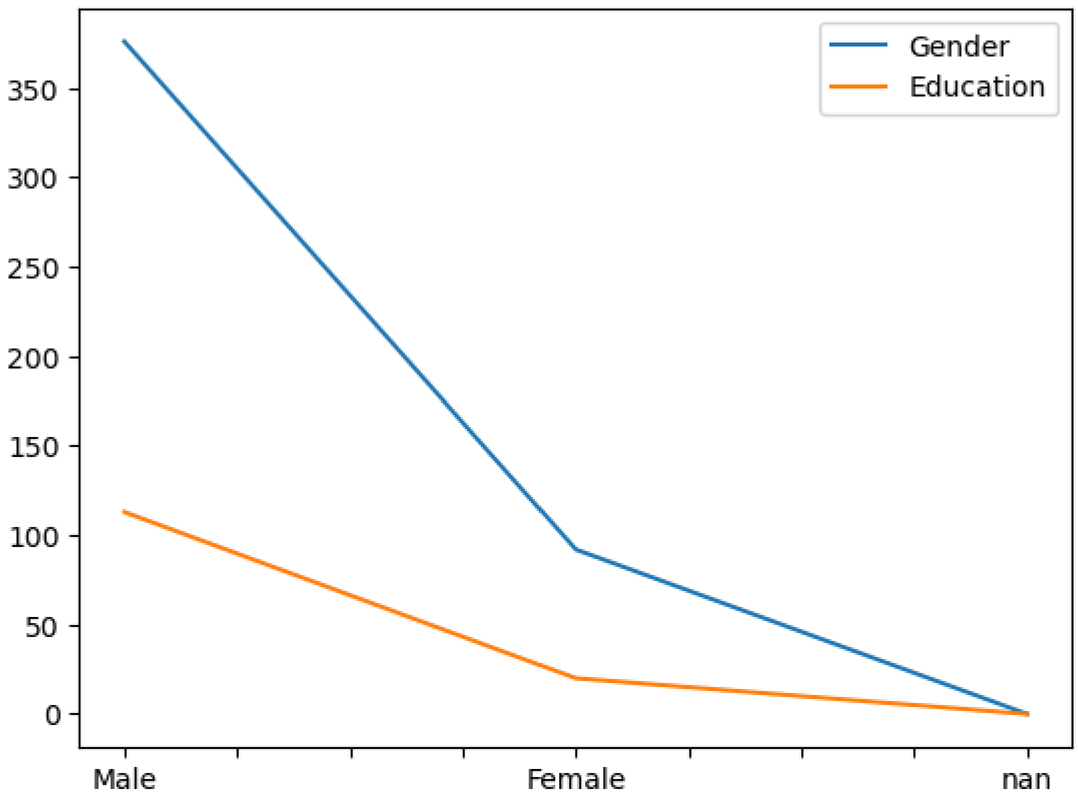


In [137…

```
df3.plot(kind='line')
```

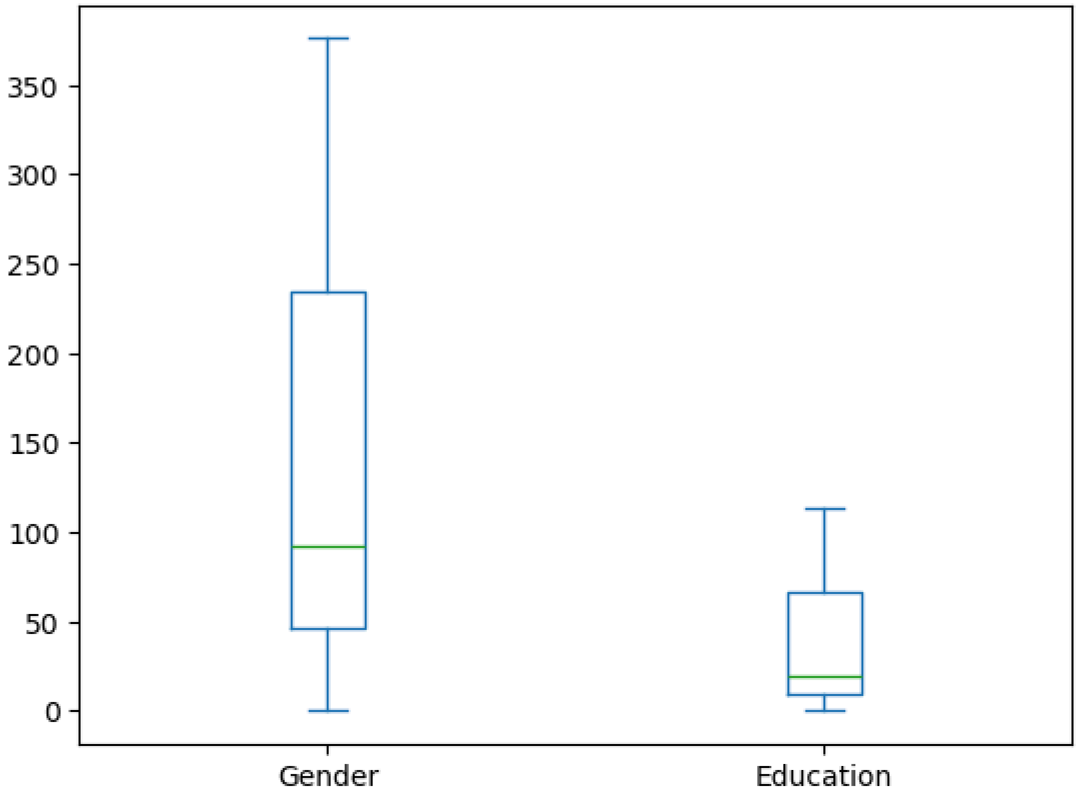
Out[137…

```
<Axes: >
```



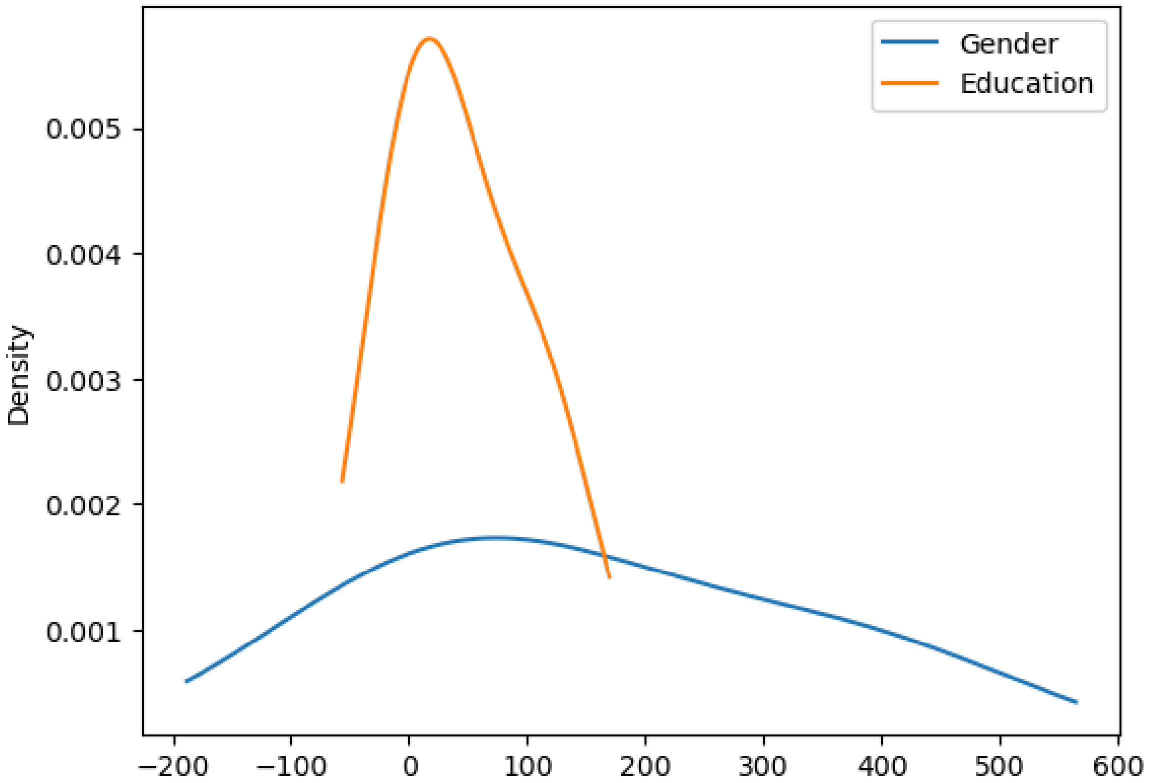
```
In [139…  df3.plot(kind='box')
```

```
Out[139…  <Axes: >
```



```
In [141…  df3.plot(kind='kde')
```

```
Out[141…  <Axes: ylabel='Density'>
```

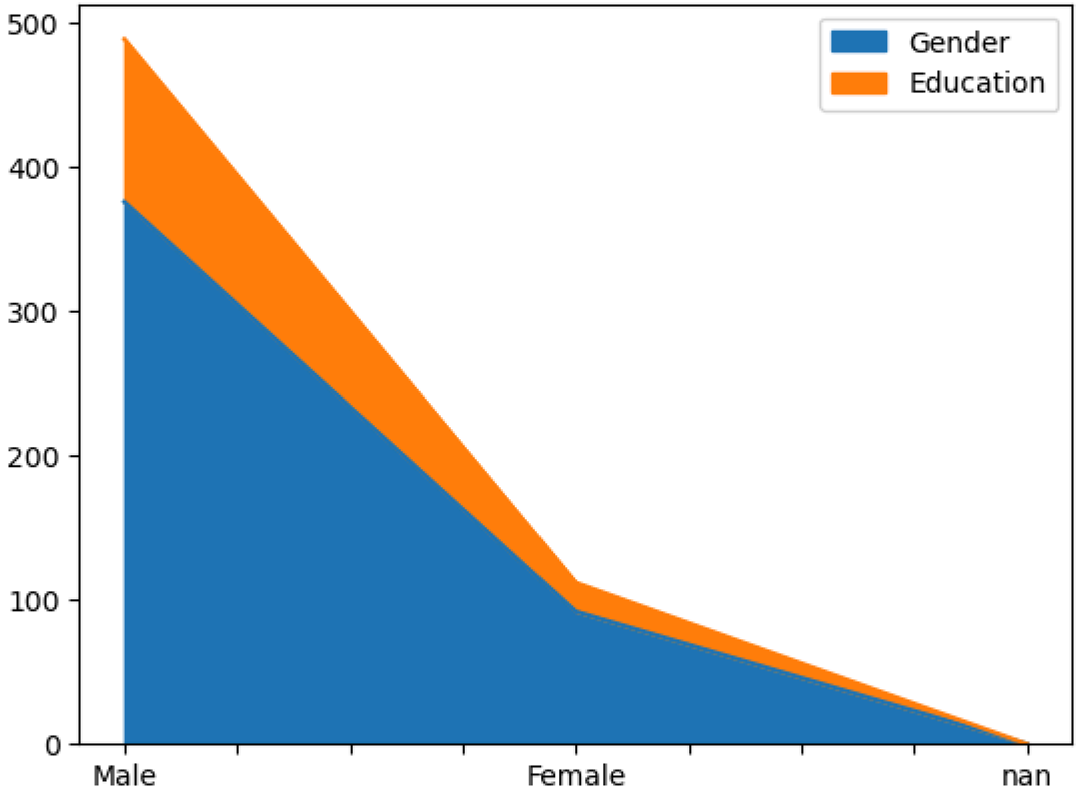


In [142…

```
df3.plot(kind='area')
```

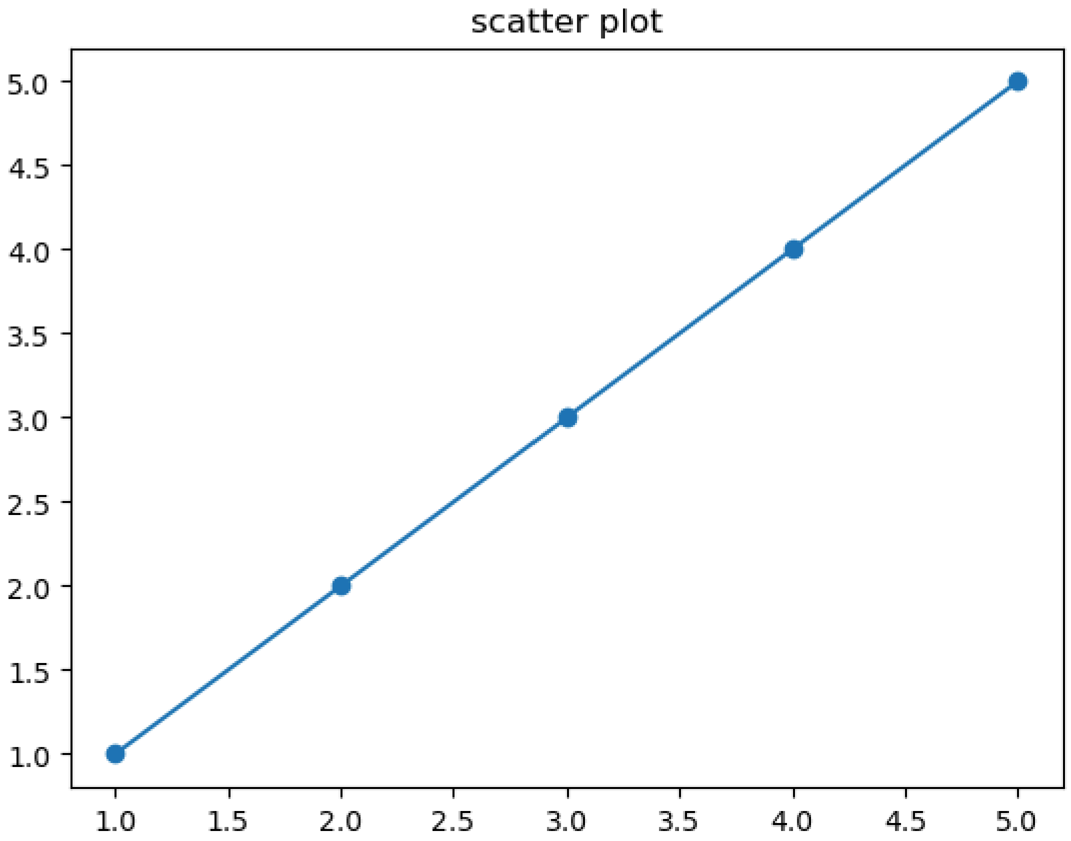
Out[142…

```
<Axes: >
```



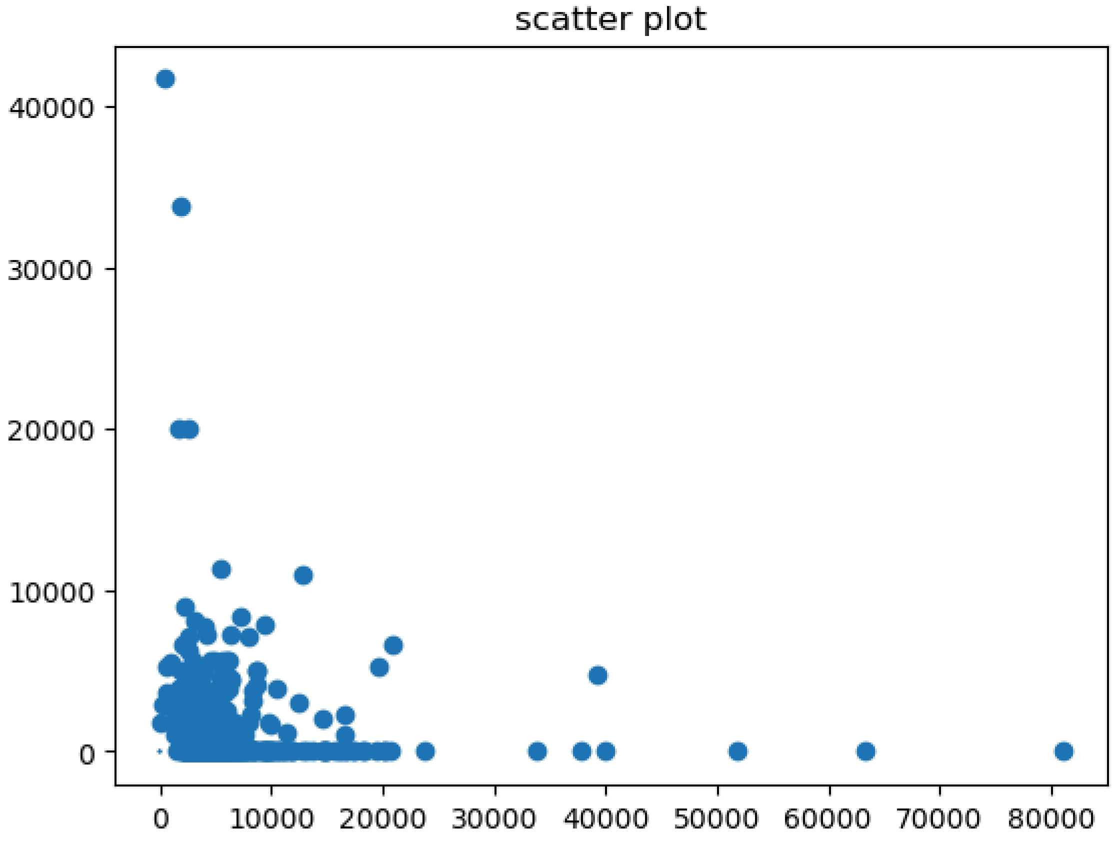
In [144…

```
x=[1,2,3,4,5]
y=[1,2,3,4,5]
plt.title('scatter plot')
plt.scatter(x,y)
plt.plot(x,y)
plt.show()
```



In [147…

```
df=(loan_df['ApplicantIncome'])
df1=(loan_df['CoapplicantIncome'])
plt.title('scatter plot')
plt.scatter(df,df1)
plt.plot(x,y)
plt.show()
```



In [148…

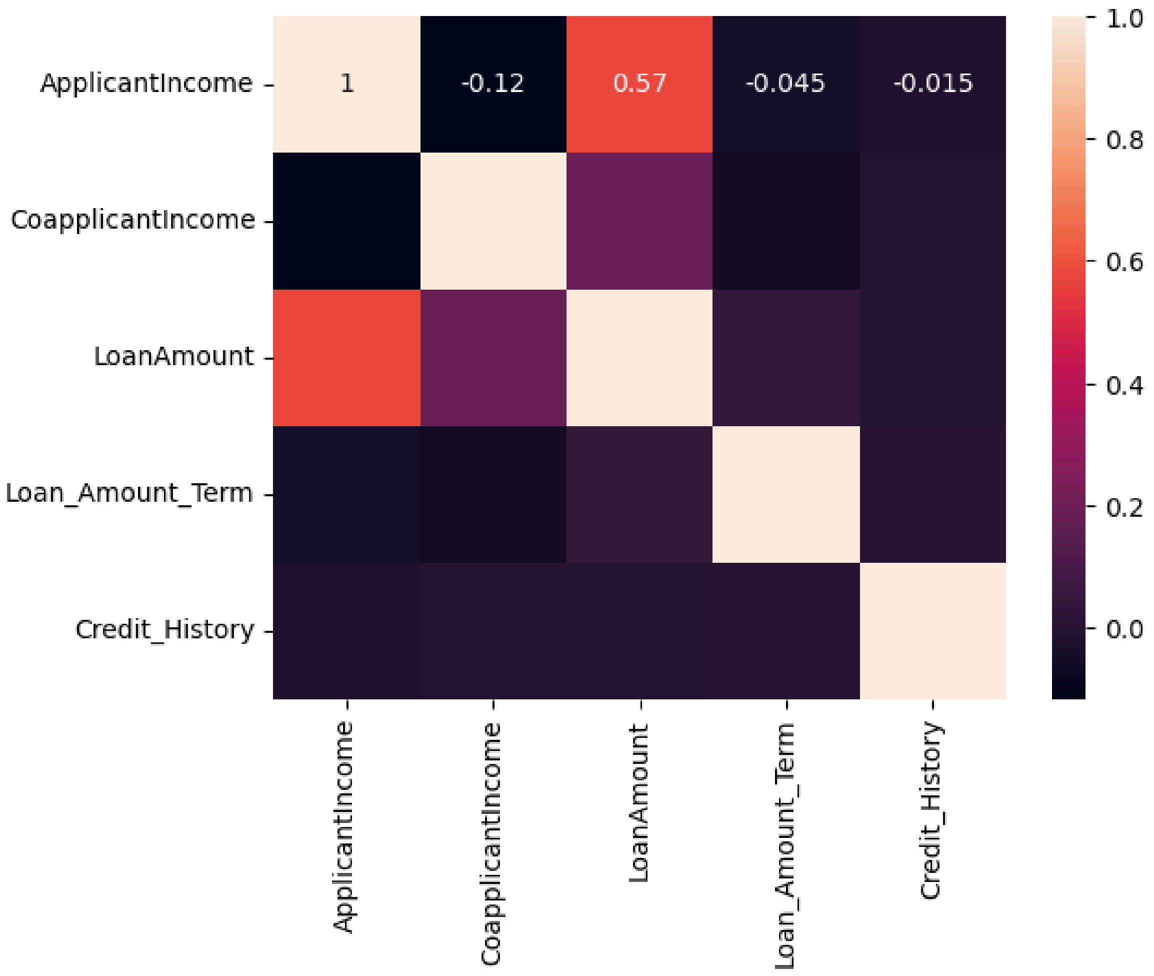
```
loan_df.corr(numeric_only=True)
```

Out[148…

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_T
ApplicantIncome	1.000000	-0.116605	0.570909	-0.045
CoapplicantIncome	-0.116605	1.000000	0.188619	-0.059
LoanAmount	0.570909	0.188619	1.000000	0.039
Loan_Amount_Term	-0.045306	-0.059878	0.039447	1.000
Credit_History	-0.014715	-0.002056	-0.008433	0.001

In [149…

```
corr_data=loan_df.corr(numeric_only=True)
sns.heatmap(corr_data,annot=True)
plt.show()
```



In [ ]: