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

Column Description

- Loan_ID :- Unique Loan ID
- Gender :- Male/ Female
- Married :- Loan Applicant married (Yes/No)
- Dependents :- Number of dependents
- Education :- Applicant Education (Graduate/ Under Graduate)
- Self_Employed :- Self employed (Yes/No)
- ApplicantIncome :- Loan Applicant income
- CoapplicantIncome :- Loan Coapplicant income
- LoanAmount :- Loan amount in thousands
- Loan_Amount_Term :- Term of loan in months
- Credit_History :- credit history meets guidelines If you have no credit history or a poor credit history, it could be harder for you to get a loan
- Property_Area :- Urban/ Semi Urban/ Rural
- Loan_Status :- Loan approved (Yes/No)

In [2]:
```
df=pd.read_csv("Loan_Data.csv")
```

In [3]:
```
df
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coap
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 [4]:
```
df.shape
```

Out[4]:
```
(614, 13)
```

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

Out[5]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

In [7]:
```
df.isna().sum()
```

Out[7]:
```
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 [8]:
```
### Filling nan value in Gender colum


df["Gender"].unique()
```

Out[8]:
```
array(['Male', 'Female', nan], dtype=object)
```

In [9]:
```
df.Gender.value_counts()
```

Out[9]:
```
Gender
Male      489
Female    112
Name: count, dtype: int64
```

In [10]:
```
df.Gender=df["Gender"].fillna("Male")
```

In [11]:
```
### Filling nan value in Married colum


df["Married"].unique()
```

Out[11]:
```
array(['No', 'Yes', nan], dtype=object)
```

In [12]:
```
df.Married.value_counts()
```

Out[12]:
```
Married
Yes    398
No     213
Name: count, dtype: int64
```

In [13]:
```
df.Married=df["Married"].fillna("Yes")
```

In [14]:
```
### Filling nan value in Dependents colum

df["Dependents"].unique()
```

Out[14]:
```
array(['0', '1', '2', '3+', nan], dtype=object)
```

In [15]:
```
df.Dependents.value_counts()
```

Out[15]:
```
Dependents
0     345
1     102
2     101
3+     51
Name: count, dtype: int64
```

In [16]:
```
df.Dependents=df["Dependents"].fillna("0")
```

In [17]:
```
### Filling nan value in Self_Employed colum


df["Self_Employed"].unique()
```

Out[17]:
```
array(['No', 'Yes', nan], dtype=object)
```

In [18]:
```
df.Self_Employed.value_counts()
```

Out[18]:
```
Self_Employed
No     500
Yes     82
Name: count, dtype: int64
```

In [19]:
```
df.Self_Employed=df["Self_Employed"].fillna("No")
```

In [20]:
```
### Filling nan value in LoanAmount colum


df["LoanAmount"].unique()
```

Out[20]:
```
array([ nan, 128.,  66., 120., 141., 267.,  95., 158., 168., 349.,  70.,
       109., 200., 114.,  17., 125., 100.,  76., 133., 115., 104., 315.,
       116., 112., 151., 191., 122., 110.,  35., 201.,  74., 106., 320.,
       144., 184.,  80.,  47.,  75., 134.,  96.,  88.,  44., 286.,  97.,
       135., 180.,  99., 165., 258., 126., 312., 136., 172.,  81., 187.,
       113., 176., 130., 111., 167., 265.,  50., 210., 175., 131., 188.,
        25., 137., 160., 225., 216.,  94., 139., 152., 118., 185., 154.,
        85., 259., 194.,  93., 370., 182., 650., 102., 290.,  84., 242.,
       129.,  30., 244., 600., 255.,  98., 275., 121.,  63., 700.,  87.,
       101., 495.,  67.,  73., 260., 108.,  58.,  48., 164., 170.,  83.,
        90., 166., 124.,  55.,  59., 127., 214., 240.,  72.,  60., 138.,
        42., 280., 140., 155., 123., 279., 192., 304., 330., 150., 207.,
       436.,  78.,  54.,  89., 143., 105., 132., 480.,  56., 159., 300.,
       376., 117.,  71., 490., 173.,  46., 228., 308., 236., 570., 380.,
       296., 156., 103.,  45.,  65.,  53., 360.,  62., 218., 178., 239.,
       405., 148., 190., 149., 153., 162., 230.,  86., 234., 246., 500.,
       186., 119., 107., 209., 208., 243.,  40., 250., 311., 400., 161.,
       196., 324., 157., 145., 181.,  26., 211.,   9., 205.,  36.,  61.,
       146., 292., 142., 350., 496., 253.])
```

In [21]:
```
df.LoanAmount.value_counts()
```

Out[21]:
```
LoanAmount
120.0    20
110.0    17
100.0    15
160.0    12
187.0    12
         ..
240.0     1
214.0     1
59.0      1
166.0     1
253.0     1
Name: count, Length: 203, dtype: int64
```

In [22]:
```
df.LoanAmount=df["LoanAmount"].fillna(df.LoanAmount.mean())
```

In [23]:
```
### Filling nan value in Loan_Amount_Term colum

df["Loan_Amount_Term"].unique()
```

Out[23]:
```
array([360., 120., 240.,  nan, 180.,  60., 300., 480.,  36.,  84.,  12.])
```

In [24]:
```
df.Loan_Amount_Term.value_counts()
```

Out[24]:
```
Loan_Amount_Term
360.0    512
180.0     44
480.0     15
300.0     13
240.0      4
84.0       4
120.0      3
60.0       2
36.0       2
12.0       1
Name: count, dtype: int64
```

In [25]:
```
df.Loan_Amount_Term=df["Loan_Amount_Term"].fillna(df.Loan_Amount_Term.mean())
```

In [26]:
```
### Filling nan value in Credit_History colum


df["Credit_History"].unique()
```

Out[26]:
```
array([ 1.,  0., nan])
```

In [27]:
```
df.Credit_History.value_counts()
```

Out[27]:
```
Credit_History
1.0    475
0.0     89
Name: count, dtype: int64
```

In [28]:
```
df.Credit_History=df["Credit_History"].fillna(df.Credit_History.mean())
```

In [29]:
```
## We remove all the nan value in data

df.isna().sum()
```

Out[29]:
```
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 [30]:
```
df.describe()
```

Out[30]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	614.000000	614.000000	614.000000
mean	5403.459283	1621.245798	146.412162	342.000000	0.842199
std	6109.041673	2926.248369	84.037468	64.372489	0.349681
min	150.000000	0.000000	9.000000	12.000000	0.000000
25%	2877.500000	0.000000	100.250000	360.000000	1.000000
50%	3812.500000	1188.500000	129.000000	360.000000	1.000000
75%	5795.000000	2297.250000	164.750000	360.000000	1.000000
max	81000.000000	41667.000000	700.000000	480.000000	1.000000

In [31]:
```
df.head()
```

Out[31]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coappl
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	

PERFORMING EXPLORATOTY DATA ANALYSIS (EDA)

Analyse Gender colum

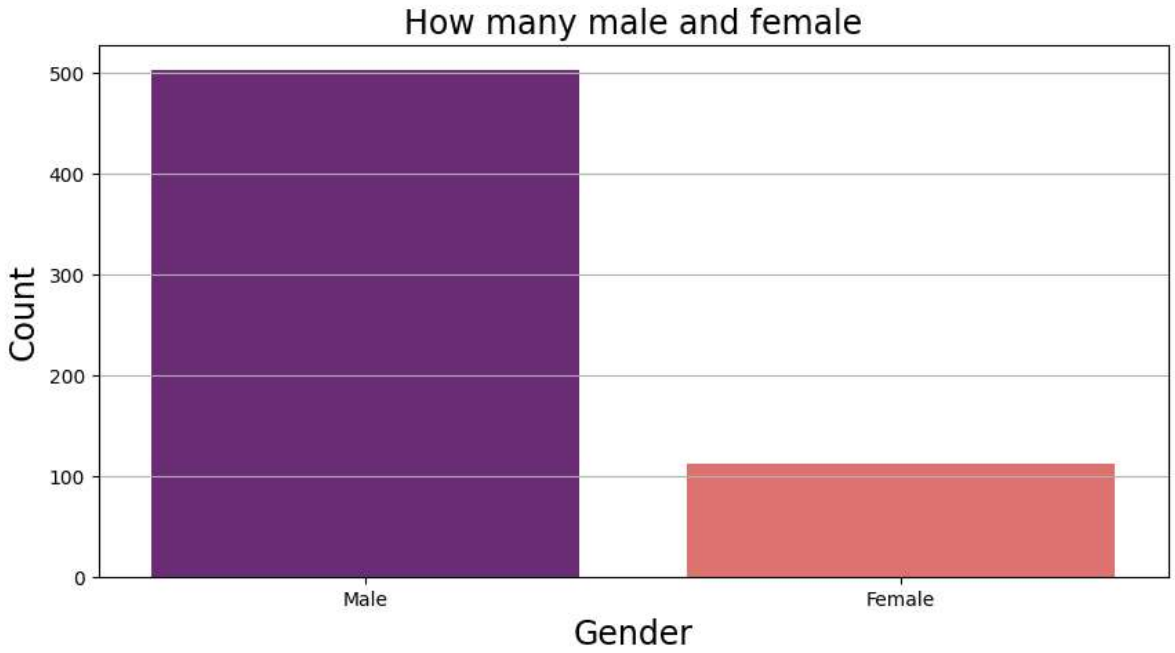
In [32]:
```
df["Gender"].unique()
```

Out[32]:
```
array(['Male', 'Female'], dtype=object)
```

In [33]:
```
df.Gender.value_counts()
```

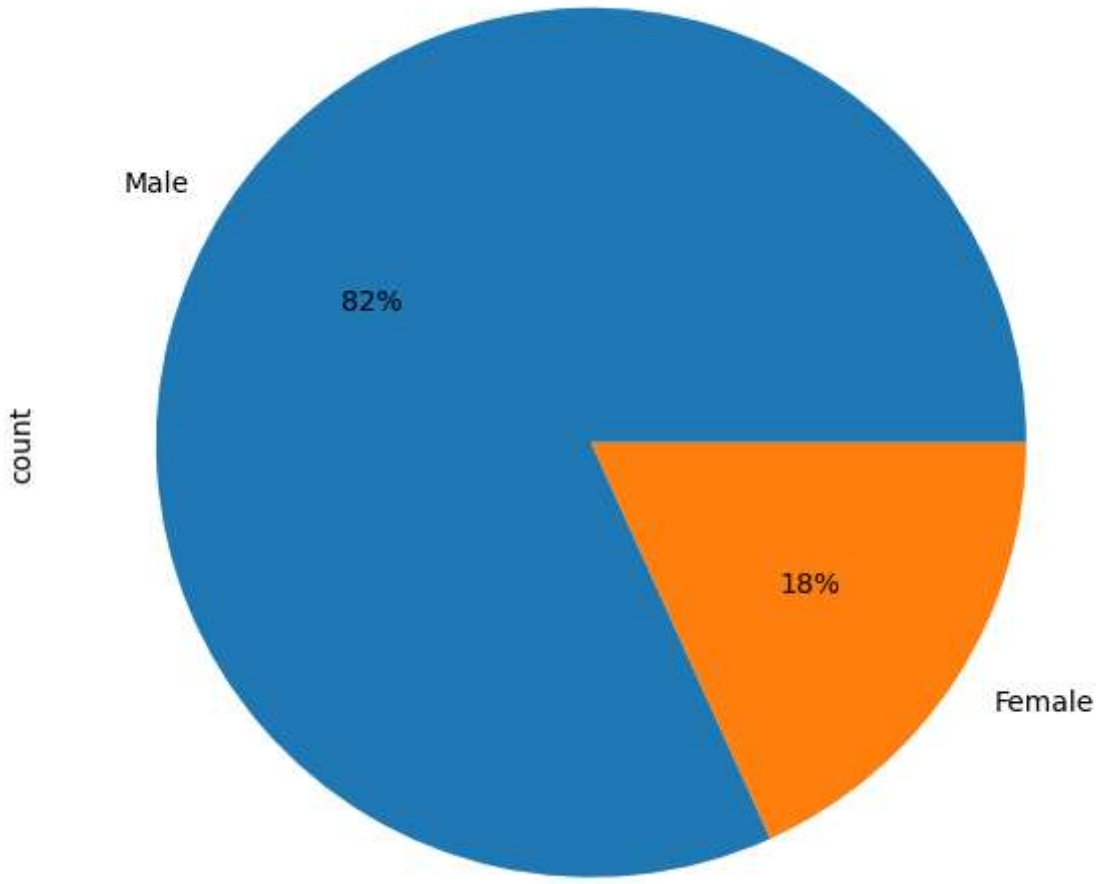
Out[33]:
```
Gender
Male      502
Female    112
Name: count, dtype: int64
```

In [34]:
```
## countplot for Gender

plt.figure(figsize=(10,5))
sns.countplot(x="Gender",data=df,palette="magma")
plt.title("How many male and female",fontsize=17)
plt.xlabel("Gender",fontsize=17)
plt.ylabel("Count",fontsize=17)
plt.grid(axis="y")
plt.show()
```



In [35]:
```
## piechart for Gender

plt.figure(figsize=(7,15))
df["Gender"].value_counts().plot.pie(autopct="%.0f%%")
```

Out[35]:
```
<Axes: ylabel='count'>
```



Observation:

- There are over 500 Male
- There are over 100 female

Analyse Married colum

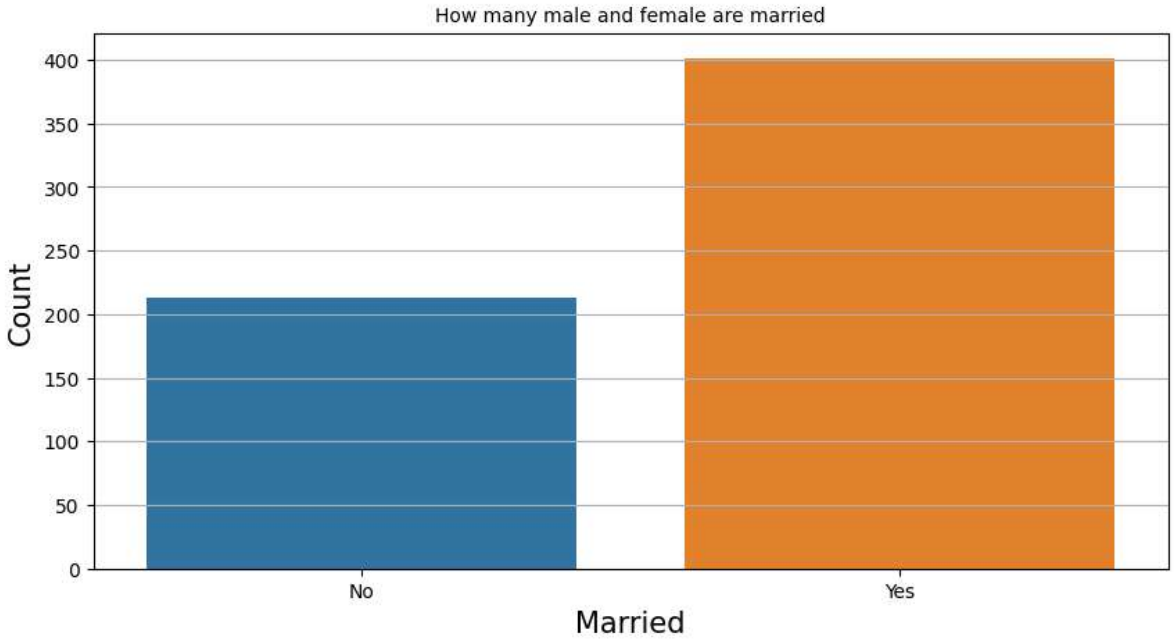
In [36]:
```
df["Married"].unique()
```

Out[36]:
```
array(['No', 'Yes'], dtype=object)
```

In [37]:
```
df.Married.value_counts()
```

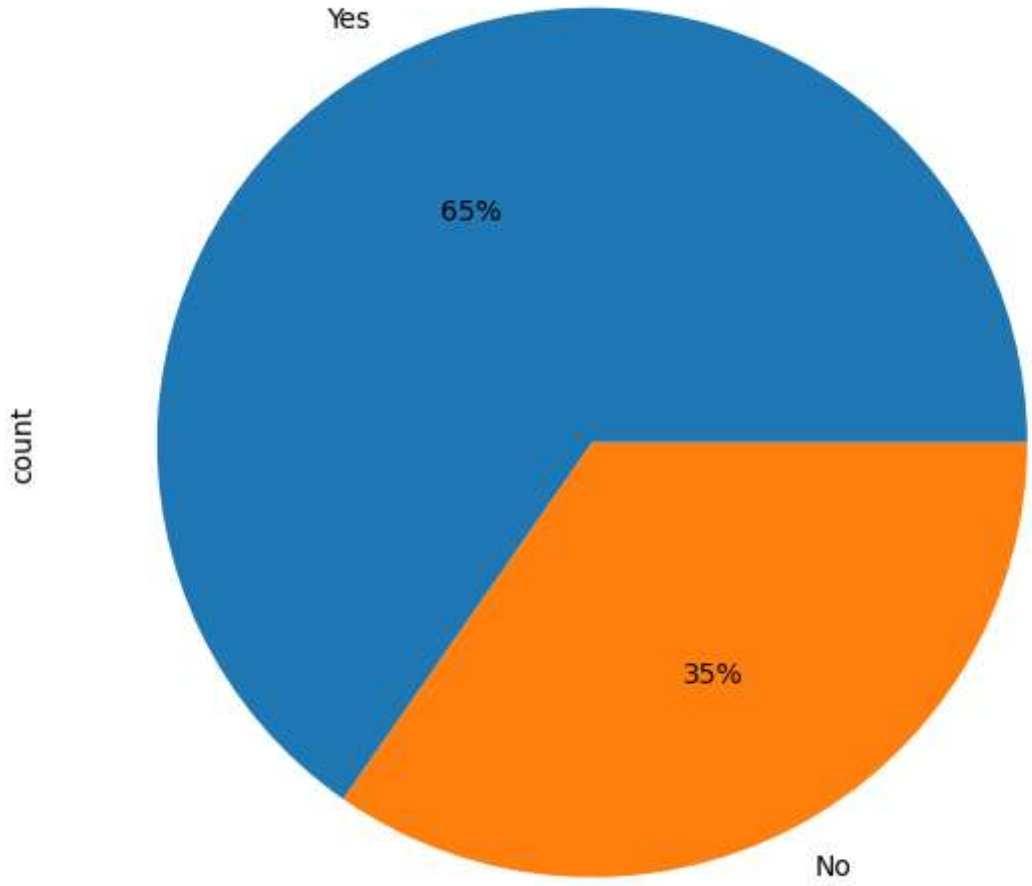
Out[37]:
```
Married
Yes    401
No     213
Name: count, dtype: int64
```

In [38]:
```
## countplot for Married


plt.figure(figsize=(10,5))
sns.countplot(x="Married",data=df)
plt.title("How many male and female are married",fontsize=10)
plt.xlabel("Married",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```



In [39]:
```
## piechart for Married

plt.figure(figsize=(7,15))
df["Married"].value_counts().plot.pie(autopct="%.0f%%")
```

Out[39]: <Axes: ylabel='count'>



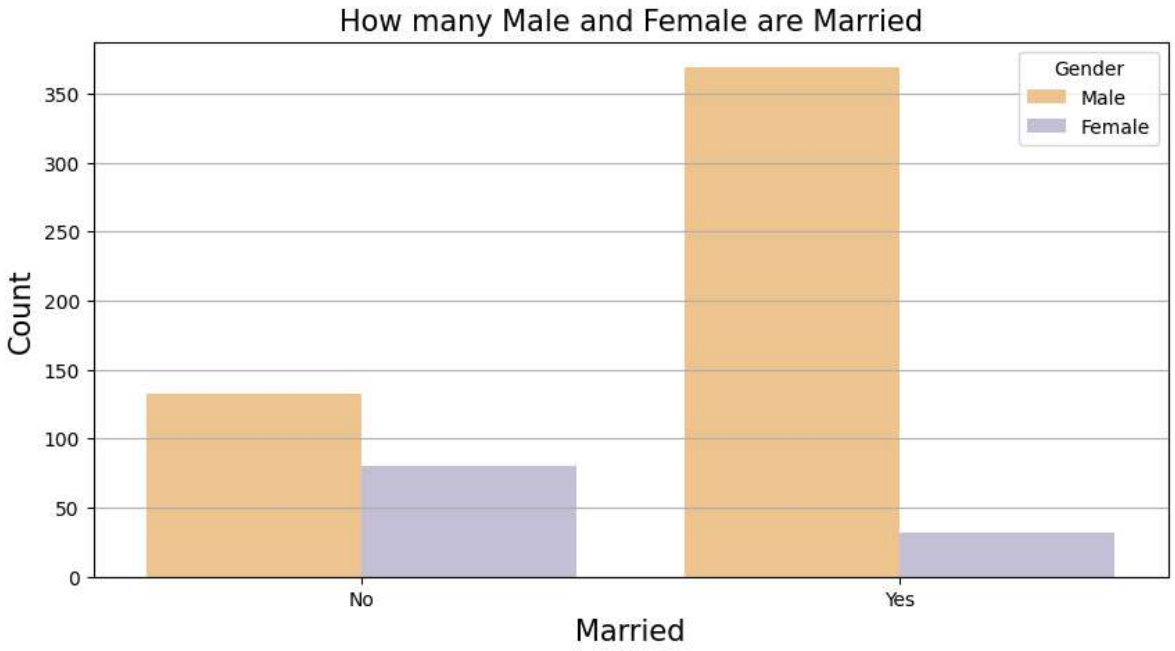
Observation:

* Most of the Male and Female are married
* There are over more than 200 Male and Female not married

Compare colum Gender and married

Compare relation between how many Male and Female are Married

In [40]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="Married",hue="Gender",data=df,palette="PuOr")
plt.title("How many Male and Female are Married",fontsize=15)
plt.xlabel("Married",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```



Observation:

* From above comparison
* More than 350 male and less than 50 Female are Married
* More than 100 male and less than 100 female are not married

Analyse Dependents colum

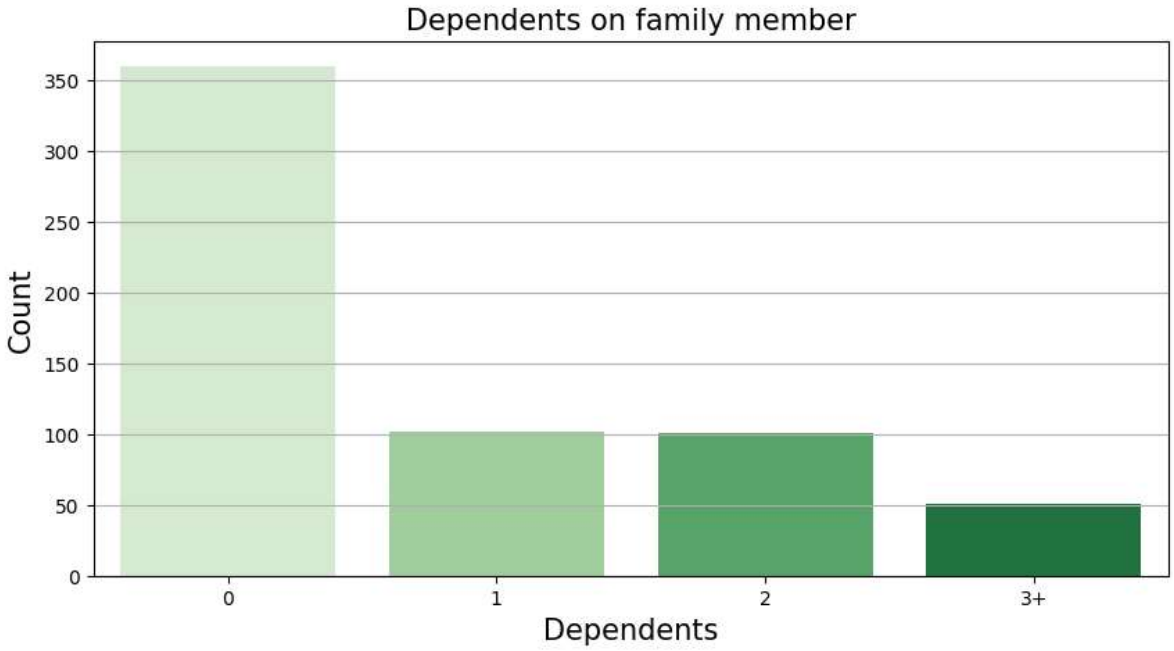
In [41]:
```
df["Dependents"].unique()
```

Out[41]:
```
array(['0', '1', '2', '3+'], dtype=object)
```

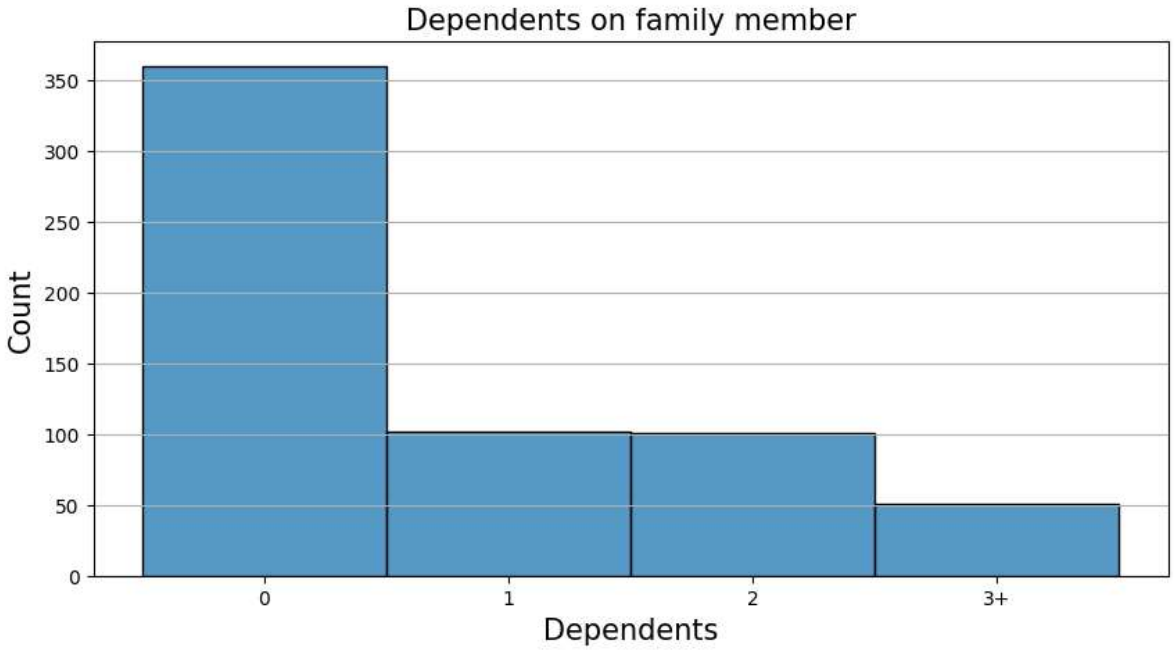
In [42]:
```
df.Dependents.value_counts()
```

Out[42]:
```
Dependents
0     360
1     102
2     101
3+     51
Name: count, dtype: int64
```

In [43]:
```
## countplot for Dependents

plt.figure(figsize=(10,5))
sns.countplot(x="Dependents",data=df,palette="Greens")
plt.title("Dependents on family member",fontsize=15)
plt.xlabel("Dependents",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```

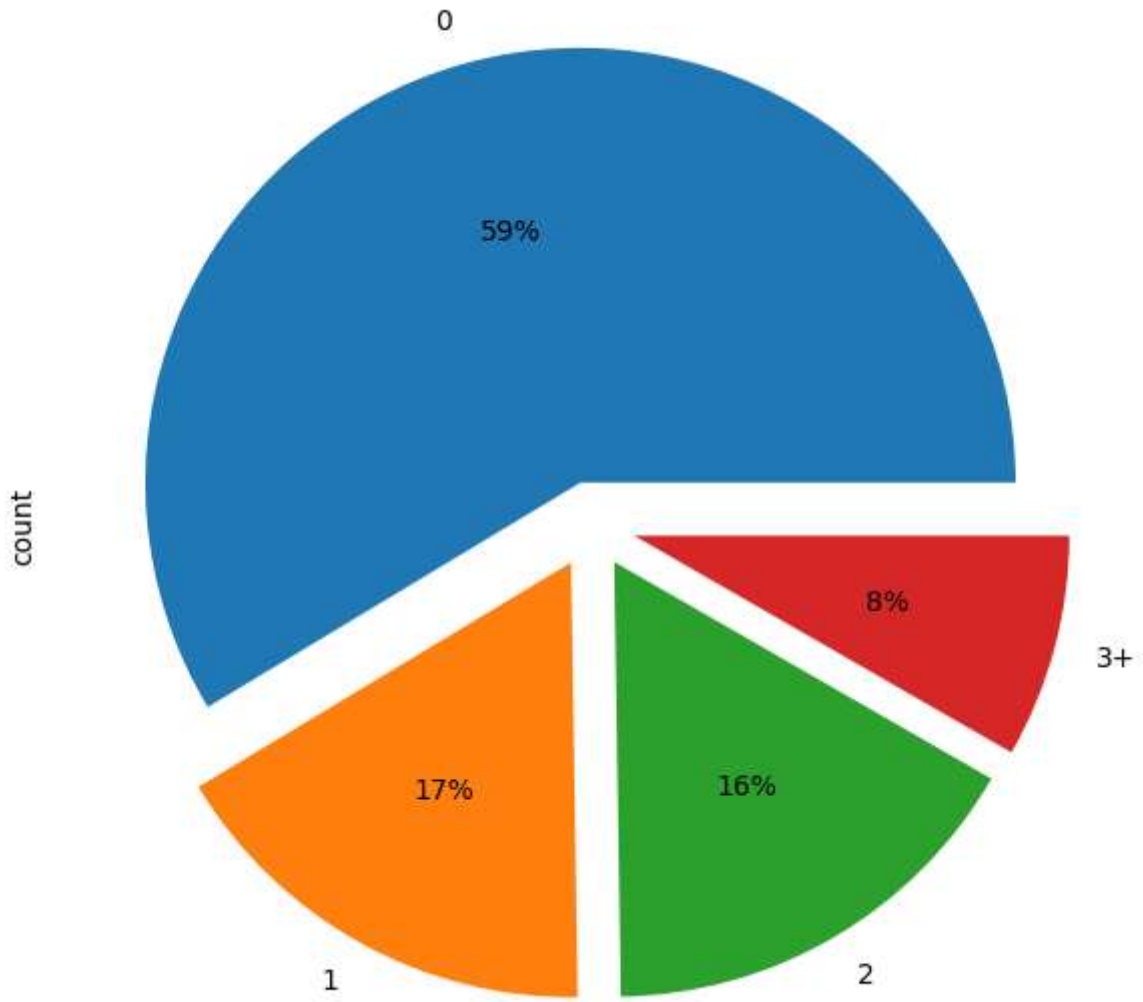


In [44]:
```
## histogram plot for Dependents

plt.figure(figsize=(10,5))
sns.histplot(x="Dependents",data=df)
plt.title("Dependents on family member",fontsize=15)
plt.xlabel("Dependents",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```



In [45]:
```
## piechart  for Dependents

plt.figure(figsize=(7,15))
df["Dependents"].value_counts().plot.pie(autopct="%.0f%%",explode=(.1,.1,.1,.1
```

Out[45]: <Axes: ylabel='count'>



Observation:

* Most of the male and female has Zero Dependency.

Analyse Education colum

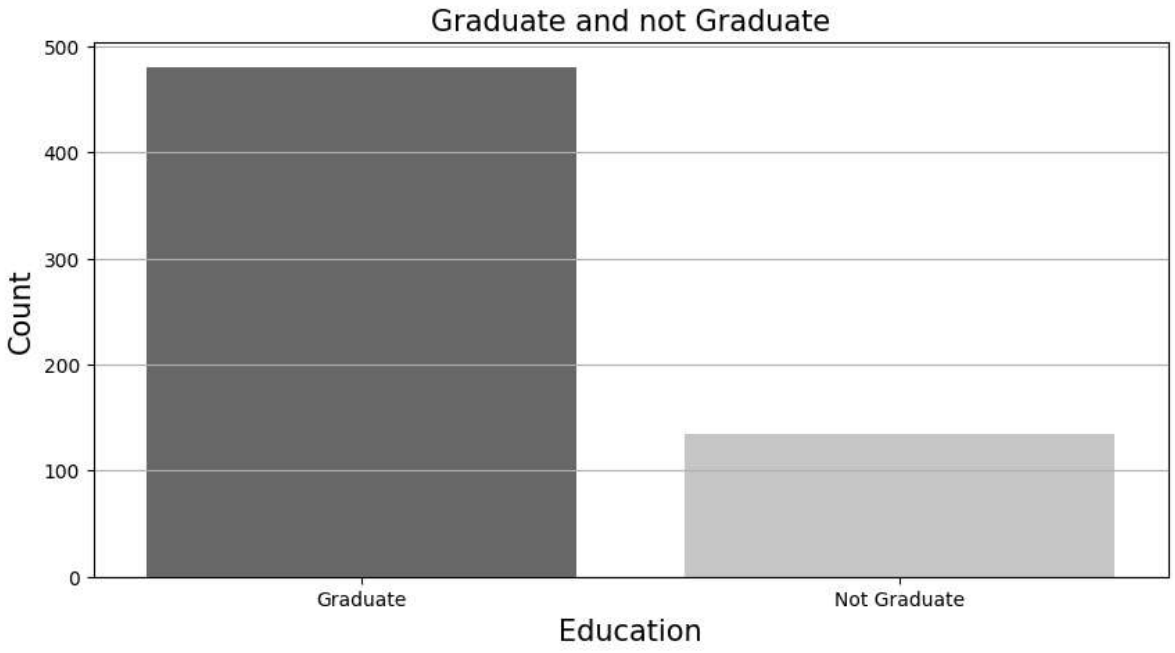
In [46]:
```
df["Education"].unique()
```

Out[46]: array(['Graduate', 'Not Graduate'], dtype=object)

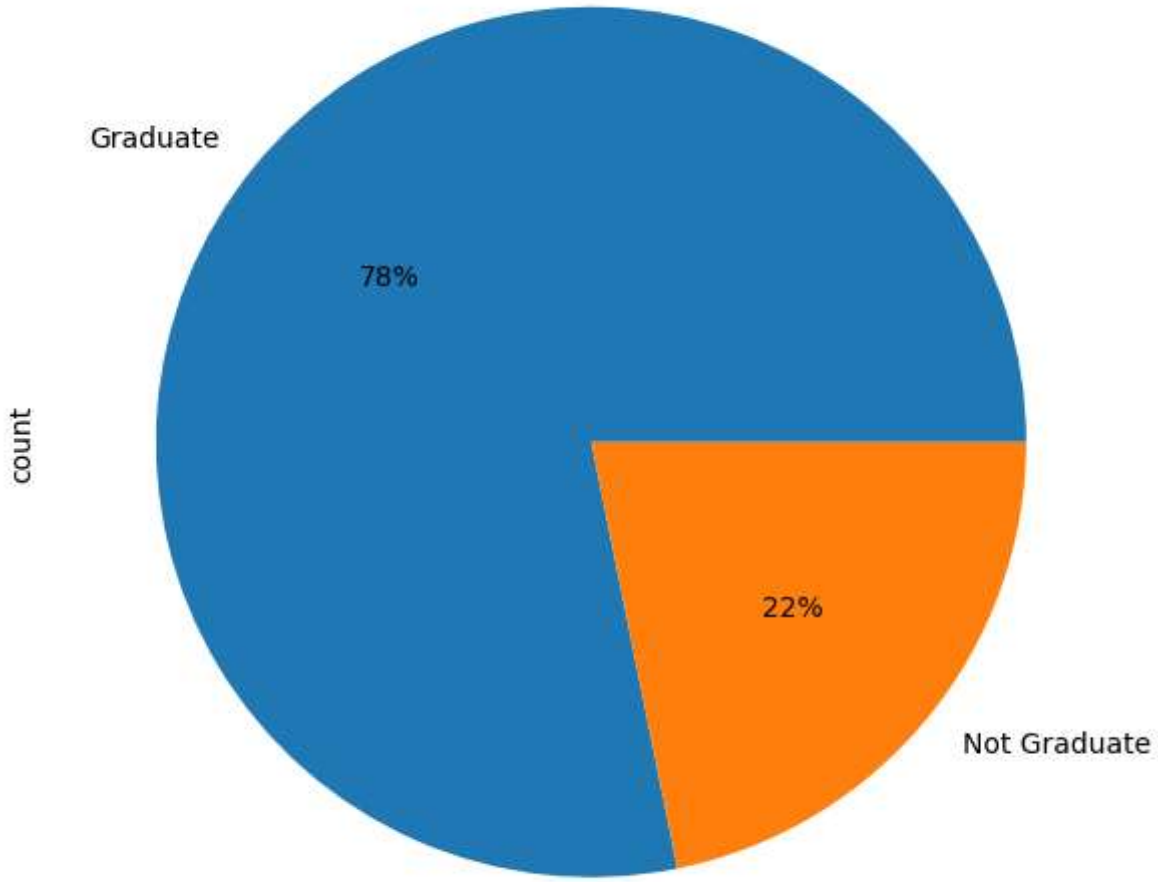
In [47]:
```
df.Education.value_counts()
```

Out[47]:
```
Education
Graduate        480
Not Graduate    134
Name: count, dtype: int64
```

In [48]:
```
## countplot  for Education

plt.figure(figsize=(10,5))
sns.countplot(x="Education",data=df,palette="Greys_r")
plt.title("Graduate and not Graduate",fontsize=15)
plt.xlabel("Education",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```



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

plt.figure(figsize=(7,10))
df["Education"].value_counts().plot.pie(autopct="%.0f%%")
plt.show()
```



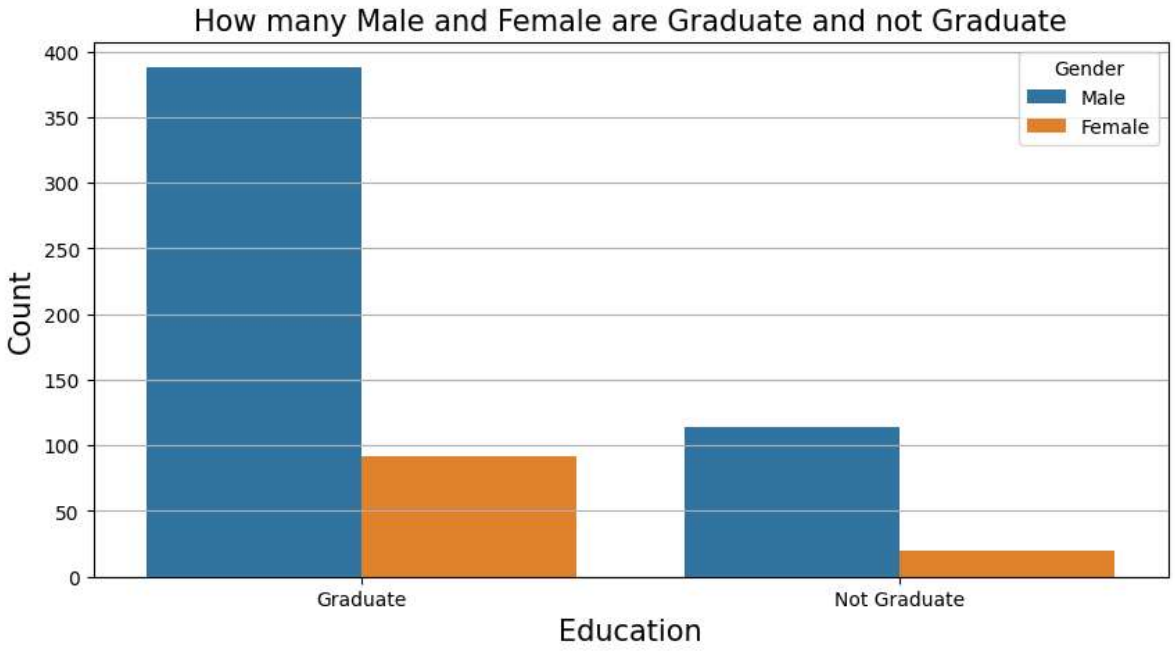
Observation:

* Above pie chart show that 78% Male and Female are Graduate
* 22% Male and Female are not Graduate

Compare colum Gender and Education

Compare relation between how many Male and Female are Graduate and not Graduate

In [50]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="Education",hue="Gender",data=df)
plt.title("How many Male and Female are Graduate and not Graduate",fontsize=15
plt.xlabel("Education",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```



Observation:

* More than 350 male and less than 100 female are Graduate
* More than 100 male and less than 50 female are not Graduate

Analyse Self_Employed colum

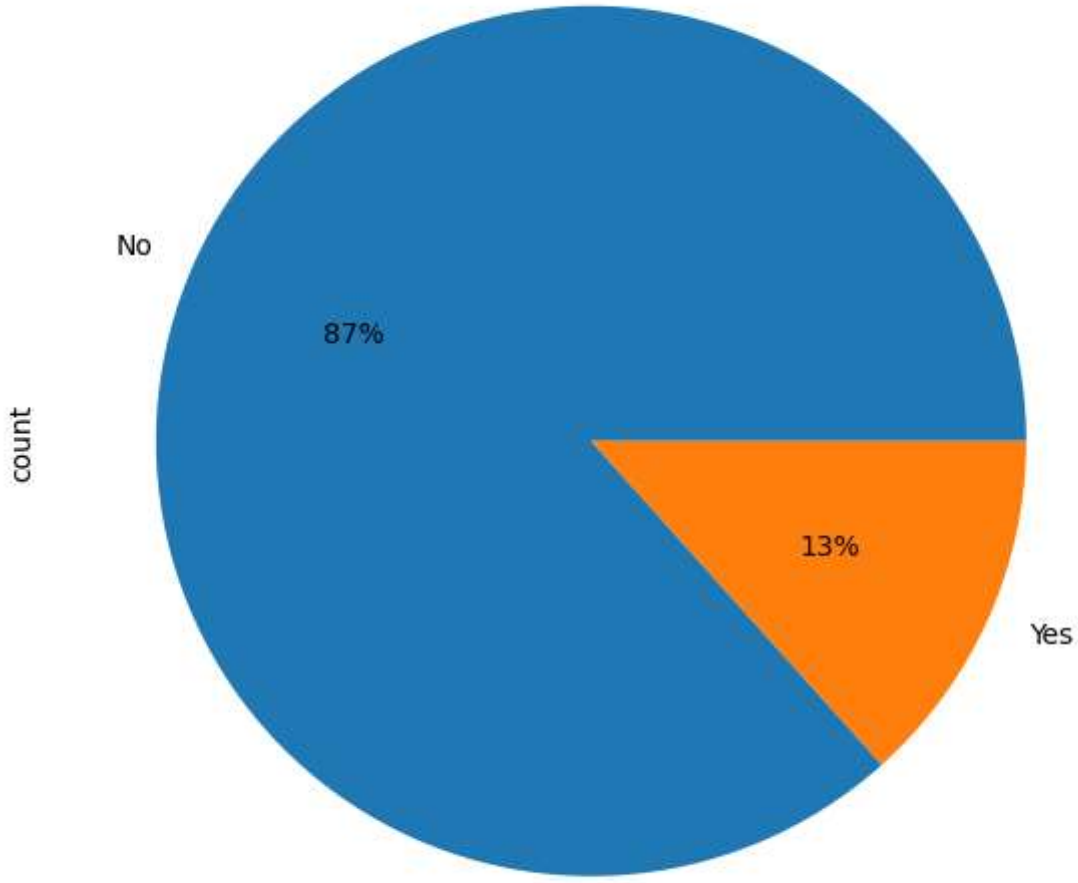
In [51]:
```
df["Self_Employed"].unique()
```

Out[51]:
```
array(['No', 'Yes'], dtype=object)
```

In [52]:
```
df.Self_Employed.value_counts()
```

Out[52]:
```
Self_Employed
No     532
Yes     82
Name: count, dtype: int64
```

In [53]:
```
plt.figure(figsize=(7,10))
df["Self_Employed"].value_counts().plot.pie(autopct="%.0f%%")
plt.show()
```



Observation:

* 87% Male and female are not self Employed
* 13% Male and female are self Employed

Analyse ApplicantIncome colum

In [54]:
```
df["ApplicantIncome"].unique()
```

Out[54]:
```
array([ 5849,  4583,  3000,  2583,  6000,  5417,  2333,  3036,  4006,
       12841,  3200,  2500,  3073,  1853,  1299,  4950,  3596,  3510,
        4887,  2600,  7660,  5955,  3365,  3717,  9560,  2799,  4226,
        1442,  3750,  4166,  3167,  4692,  3500, 12500,  2275,  1828,
        3667,  3748,  3600,  1800,  2400,  3941,  4695,  3410,  5649,
        5821,  2645,  4000,  1928,  3086,  4230,  4616, 11500,  2708,
        2132,  3366,  8080,  3357,  3029,  2609,  4945,  5726, 10750,
        7100,  4300,  3208,  1875,  4755,  5266,  1000,  3333,  3846,
        2395,  1378,  3988,  2366,  8566,  5695,  2958,  6250,  3273,
        4133,  3620,  6782,  2484,  1977,  4188,  1759,  4288,  4843,
       13650,  4652,  3816,  3052, 11417,  7333,  3800,  2071,  5316,
        2929,  3572,  7451,  5050, 14583,  2214,  5568, 10408,  5667,
        2137,  2957,  3692, 23803,  3865, 10513,  6080, 20166,  2014,
        2718,  3459,  4895,  3316, 14999,  4200,  5042,  6950,  2698,
       11757,  2330, 14866,  1538, 10000,  4860,  6277,  2577,  9166,
        2281,  3254, 39999,  9538,  2980,  1863,  7933,  3089,  4167,
        9323,  3707,  2439,  2237,  8000,  1820, 51763,  3522,  5708,
        4344,  3497,  2045,  5516,  6400,  1916,  4600, 33846,  3625,
       39147,  2178,  2383,   674,  9328,  4885, 12000,  6033,  3858,
```

In [55]:
```
df.ApplicantIncome.value_counts()
```

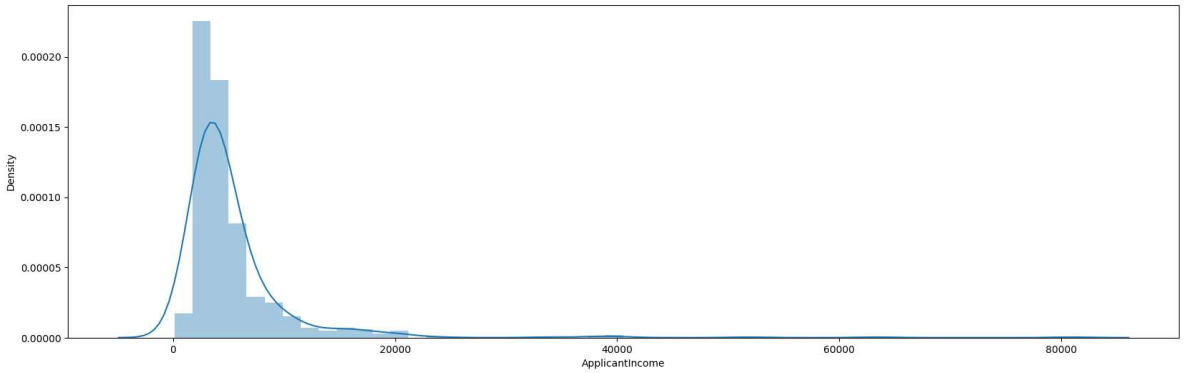
Out[55]:
```
ApplicantIncome
2500    9
4583    6
6000    6
2600    6
3333    5
       ..
3244    1
4408    1
3917    1
3992    1
7583    1
Name: count, Length: 505, dtype: int64
```

In [56]:
```
df["ApplicantIncome"].min(),df["ApplicantIncome"].max()
```

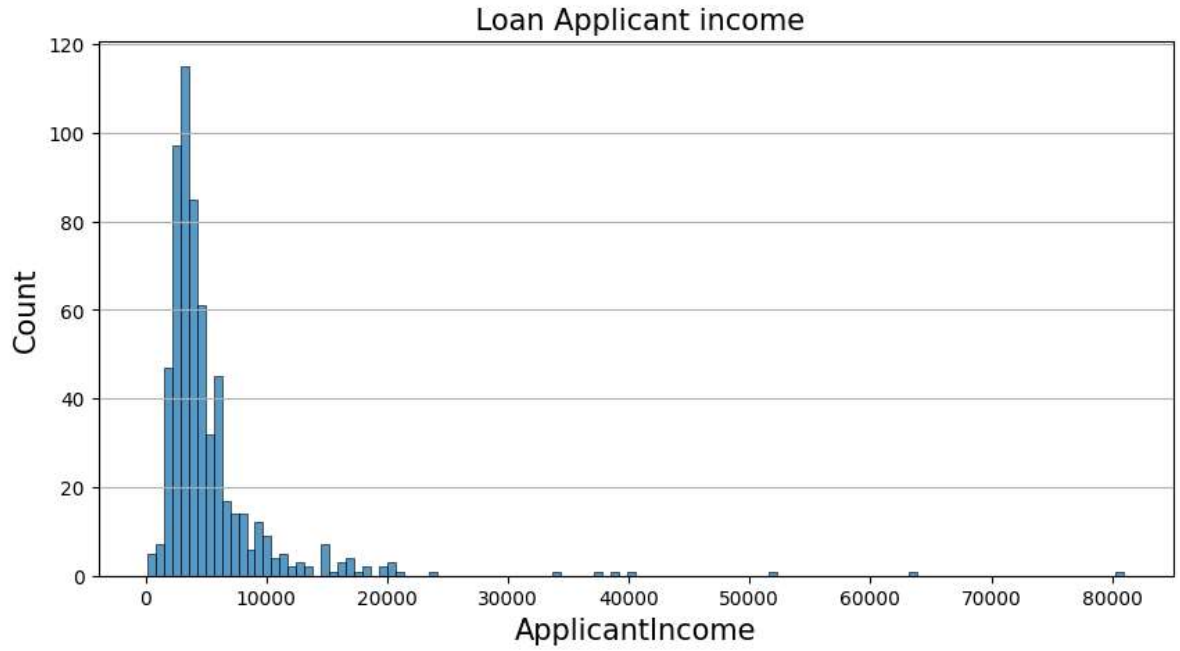
Out[56]:
```
(150, 81000)
```

In [57]:
```
## distplot  for ApplicantIncome

plt.figure(figsize=(20,6))
sns.distplot(df["ApplicantIncome"])
```

Out[57]:
```
<Axes: xlabel='ApplicantIncome', ylabel='Density'>
```



In [58]:
```
## histplot  for ApplicantIncome

plt.figure(figsize=(10,5))
sns.histplot(x="ApplicantIncome",data=df)
plt.title(" Loan Applicant income",fontsize=15)
plt.xlabel("ApplicantIncome",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```



Observation:

* Most of the Applicant Income in between 0 to 10000

Analyse CoapplicantIncome colum

In [59]:
```
df["CoapplicantIncome"].unique()
```

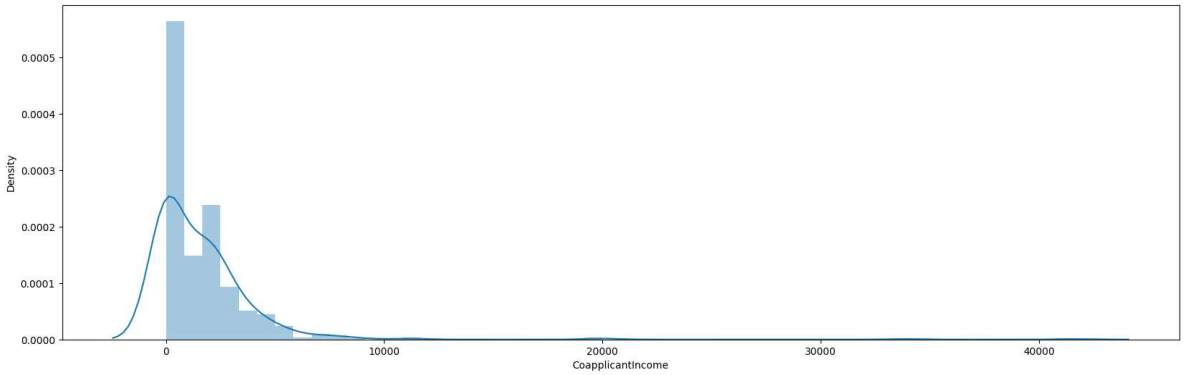
Out[59]:
```
array([0.00000000e+00, 1.50800000e+03, 2.35800000e+03, 4.19600000e+03,
       1.51600000e+03, 2.50400000e+03, 1.52600000e+03, 1.09680000e+04,
       7.00000000e+02, 1.84000000e+03, 8.10600000e+03, 2.84000000e+03,
       1.08600000e+03, 3.50000000e+03, 5.62500000e+03, 1.91100000e+03,
       1.91700000e+03, 2.92500000e+03, 2.25300000e+03, 1.04000000e+03,
       2.08300000e+03, 3.36900000e+03, 1.66700000e+03, 3.00000000e+03,
       2.06700000e+03, 1.33000000e+03, 1.45900000e+03, 7.21000000e+03,
       1.66800000e+03, 1.21300000e+03, 2.33600000e+03, 3.44000000e+03,
       2.27500000e+03, 1.64400000e+03, 1.16700000e+03, 1.59100000e+03,
       2.20000000e+03, 2.25000000e+03, 2.85900000e+03, 3.79600000e+03,
       3.44900000e+03, 4.59500000e+03, 2.25400000e+03, 3.06600000e+03,
       1.87500000e+03, 1.77400000e+03, 4.75000000e+03, 3.02200000e+03,
       4.00000000e+03, 2.16600000e+03, 1.88100000e+03, 2.53100000e+03,
       2.00000000e+03, 2.11800000e+03, 4.16700000e+03, 2.90000000e+03,
       5.65400000e+03, 1.82000000e+03, 2.30200000e+03, 9.97000000e+02,
       3.54100000e+03, 3.26300000e+03, 3.80600000e+03, 3.58300000e+03,
       7.54000000e+02, 1.03000000e+03, 1.12600000e+03, 3.60000000e+03,
       2.33300000e+03, 4.11400000e+03, 2.28300000e+03, 1.39800000e+03,
       2.14200000e+03, 2.66700000e+03, 8.98000000e+03, 2.01400000e+03,
```

In [60]:
```
df.CoapplicantIncome.value_counts()
```

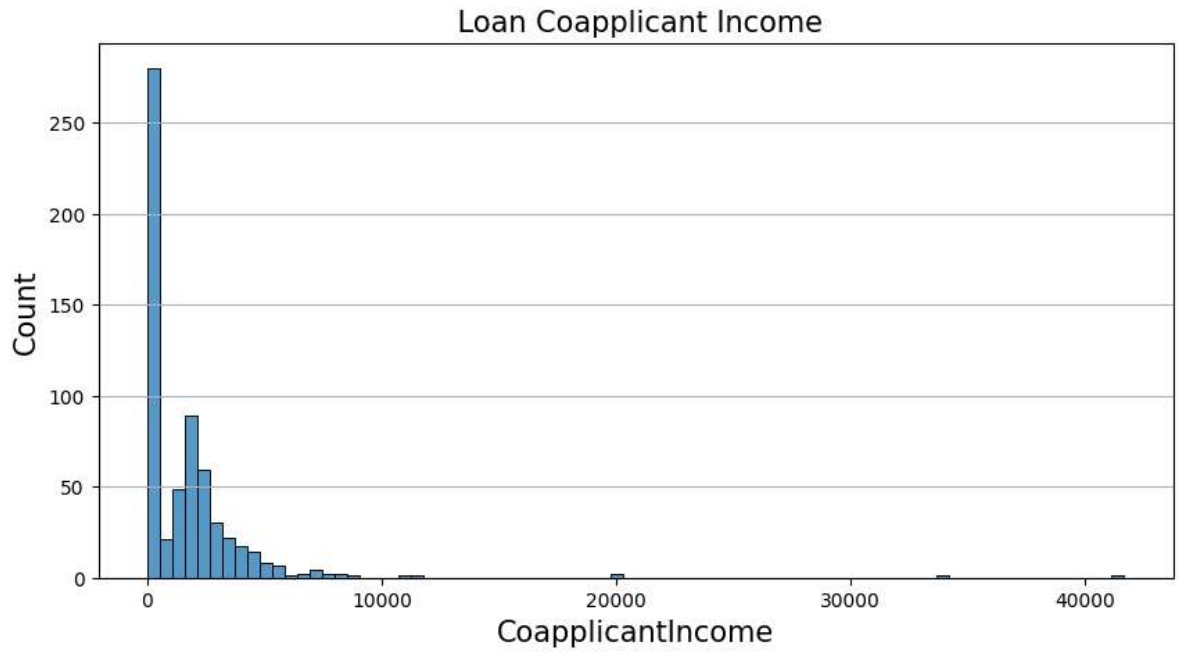
Out[60]:
```
CoapplicantIncome
0.0       273
2500.0      5
2083.0      5
1666.0      5
2250.0      3
         ...
2791.0      1
1010.0      1
1695.0      1
2598.0      1
240.0       1
Name: count, Length: 287, dtype: int64
```

In [61]:
```
## distplot  for CoapplicantIncome

plt.figure(figsize=(20,6))
sns.distplot(df["CoapplicantIncome"])
```

Out[61]:
```
<Axes: xlabel='CoapplicantIncome', ylabel='Density'>
```



In [62]:
```
## histplot  for CoapplicantIncome

plt.figure(figsize=(10,5))
sns.histplot(x="CoapplicantIncome",data=df)
plt.title(" Loan Coapplicant Income",fontsize=15)
plt.xlabel("CoapplicantIncome",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```



Analyse LoanAmount colum

In [63]:
```
df["LoanAmount"].unique()
```

Out[63]:
```
array([146.41216216, 128.        ,  66.        , 120.        ,
       141.        , 267.        ,  95.        , 158.        ,
       168.        , 349.        ,  70.        , 109.        ,
       200.        , 114.        ,  17.        , 125.        ,
       100.        ,  76.        , 133.        , 115.        ,
       104.        , 315.        , 116.        , 112.        ,
       151.        , 191.        , 122.        , 110.        ,
        35.        , 201.        ,  74.        , 106.        ,
       320.        , 144.        , 184.        ,  80.        ,
        47.        ,  75.        , 134.        ,  96.        ,
        88.        ,  44.        , 286.        ,  97.        ,
       135.        , 180.        ,  99.        , 165.        ,
       258.        , 126.        , 312.        , 136.        ,
       172.        ,  81.        , 187.        , 113.        ,
       176.        , 130.        , 111.        , 167.        ,
       265.        ,  50.        , 210.        , 175.        ,
       131.        , 188.        ,  25.        , 137.        ,
       160.        , 225.        , 216.        ,  94.        ,
       139.        , 152.        , 118.        , 185.        ,
```

In [64]:
```
df.LoanAmount.value_counts()
```

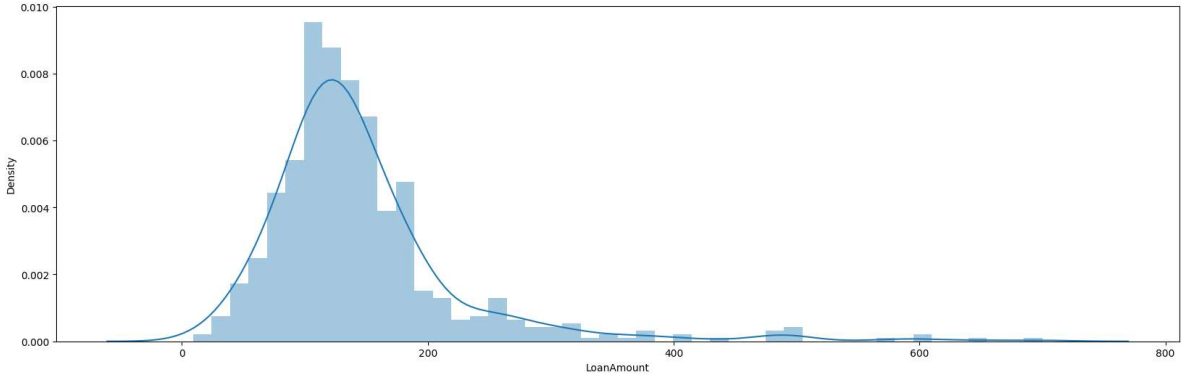
Out[64]:
```
LoanAmount
146.412162    22
120.000000    20
110.000000    17
100.000000    15
160.000000    12
              ..
240.000000     1
214.000000     1
59.000000      1
166.000000     1
253.000000     1
Name: count, Length: 204, dtype: int64
```

In [65]:
```
df["LoanAmount"].min(),df["LoanAmount"].max()
```

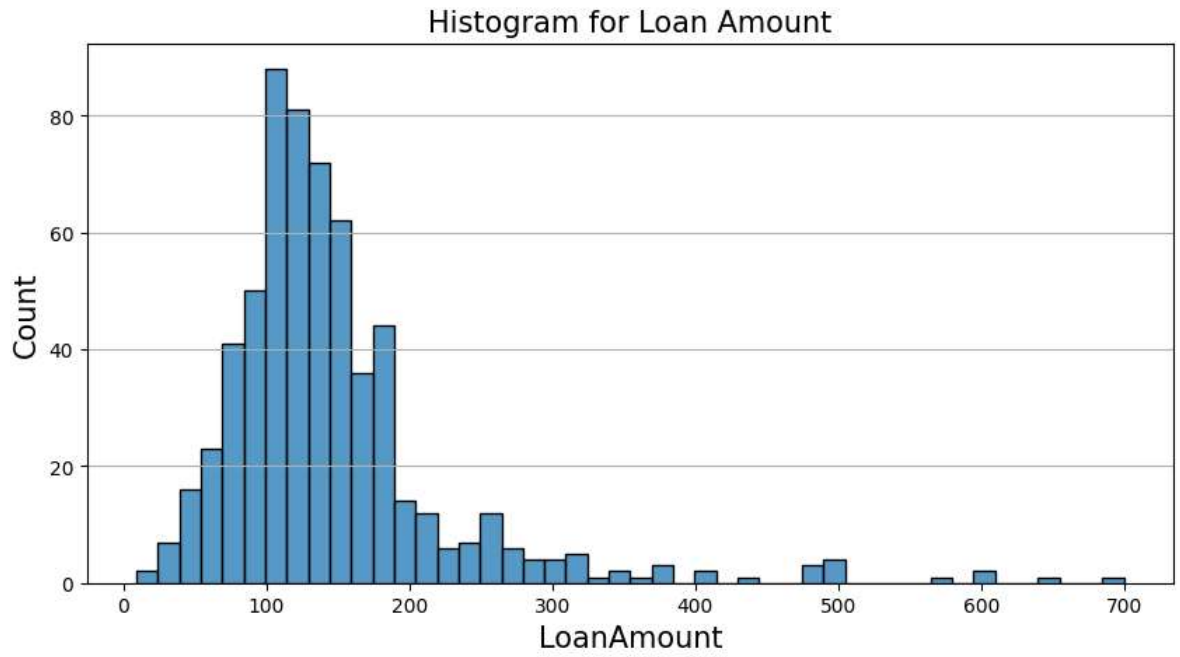
Out[65]:
```
(9.0, 700.0)
```

In [66]:
```
## distplot  for LoanAmount


plt.figure(figsize=(20,6))
sns.distplot(df["LoanAmount"])
```

Out[66]:
```
<Axes: xlabel='LoanAmount', ylabel='Density'>
```



In [67]:
```
## histplot  for LoanAmount

plt.figure(figsize=(10,5))
sns.histplot(x="LoanAmount",data=df)
plt.title("Histogram for Loan Amount",fontsize=15)
plt.xlabel("LoanAmount",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```



Observation:

* Most of LoanAmount in between 0 to 200

Analyse Loan_Amount_Term colum

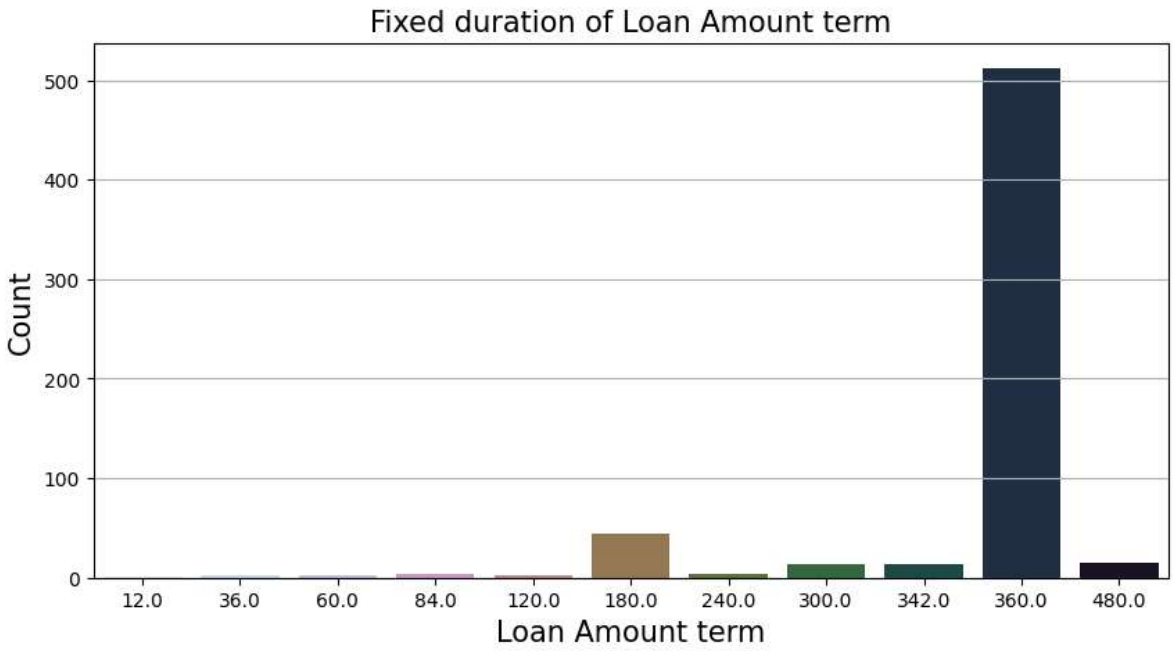
In [68]:
```
df["Loan_Amount_Term"].unique()
```

Out[68]:
```
array([360., 120., 240., 342., 180.,  60., 300., 480.,  36.,  84.,  12.])
```

In [69]:
```
df.Loan_Amount_Term.value_counts()
```

Out[69]:
```
Loan_Amount_Term
360.0    512
180.0     44
480.0     15
342.0     14
300.0     13
240.0      4
84.0       4
120.0      3
60.0       2
36.0       2
12.0       1
Name: count, dtype: int64
```

In [70]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="Loan_Amount_Term",data=df,palette="cubehelix_r")
plt.title("Fixed duration of Loan Amount term",fontsize=15)
plt.xlabel("Loan Amount term",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```



Observation:

* Fixed duration of Loan Amount term is 360

Analyse Credit_History colum

In [71]:
```
df["Credit_History"].unique()
```

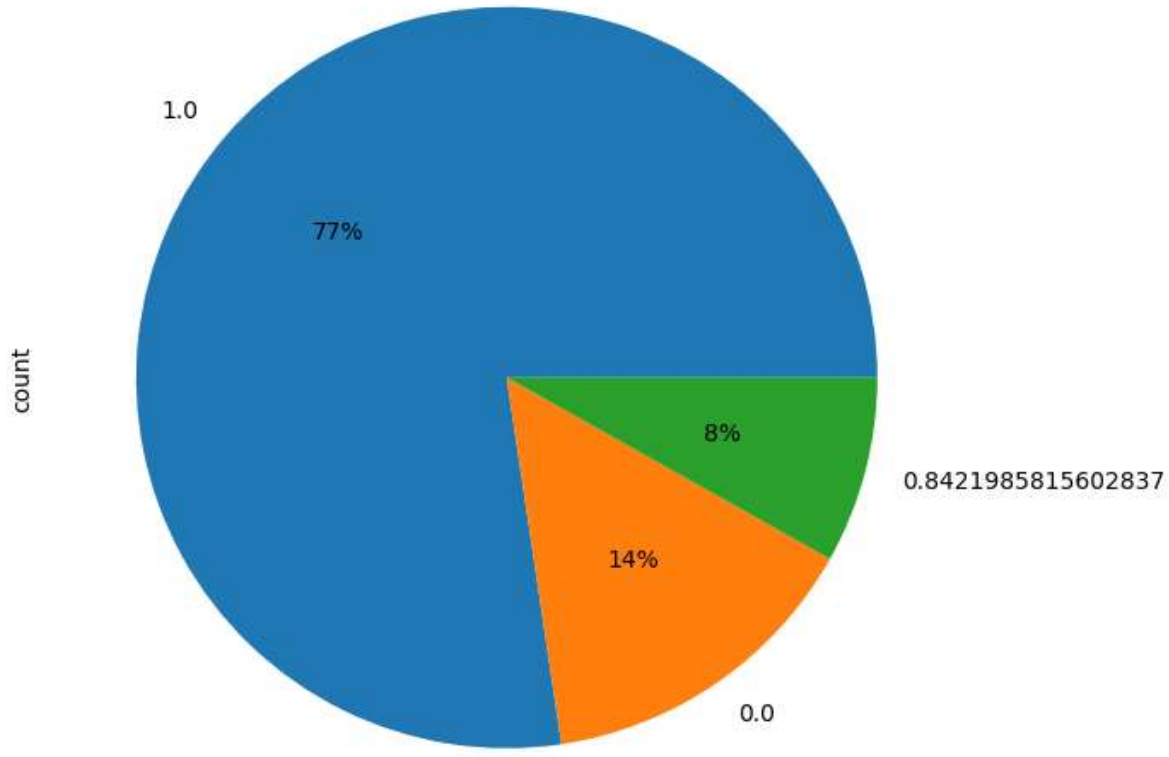
Out[71]:
```
array([1.        , 0.        , 0.84219858])
```

In [72]:
```
df.Credit_History.value_counts()
```

Out[72]:
```
Credit_History
1.000000    475
0.000000     89
0.842199     50
Name: count, dtype: int64
```

In [73]:
```
plt.figure(figsize=(7,15))
df["Credit_History"].value_counts().plot.pie(autopct="%.0f%%")
```

Out[73]:
```
<Axes: ylabel='count'>
```



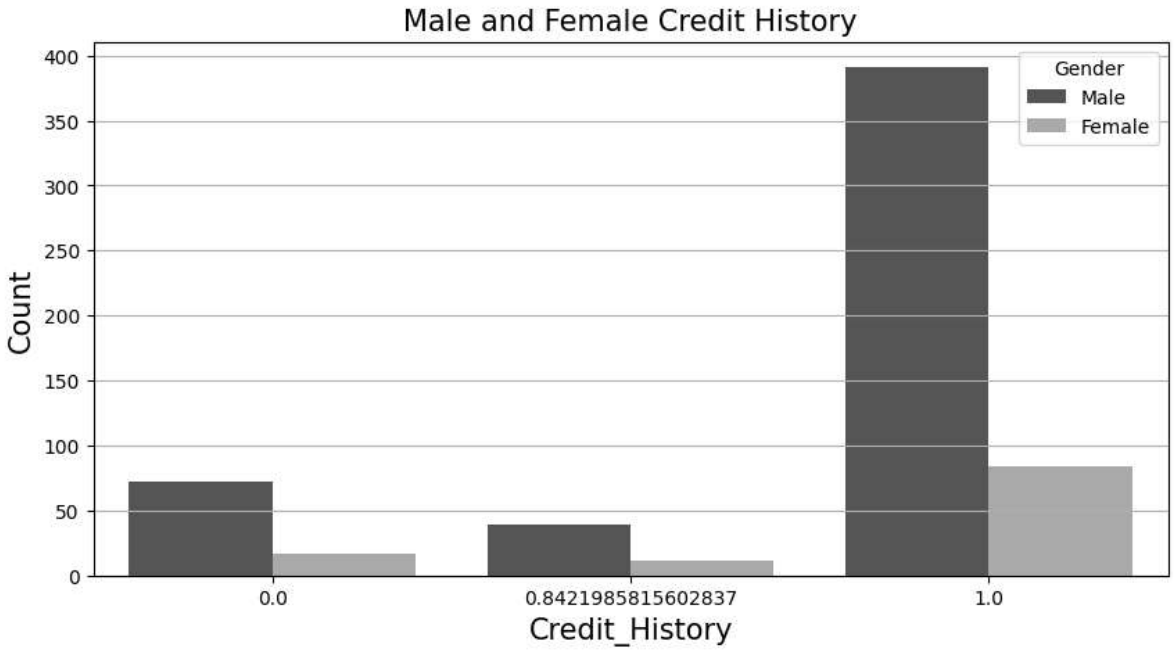
Observation:

* Above chart show
* 77% Male and female pay there repayments in 30days thats why the
y have a good credit history
* 14% and 8% Male and female Take a time for repayments thats why

Compare colum Gender and Credit_History

Compare relation between how many Male and Female are paid there repayments in 30 days

In [74]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="Credit_History",hue="Gender",data=df,palette="binary_r")
plt.title("Male and Female Credit History",fontsize=15)
plt.xlabel("Credit_History",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```



Observation:

* More than 350 male and more than 50 female have a good credit h
istory

Analyse Property_Area colum

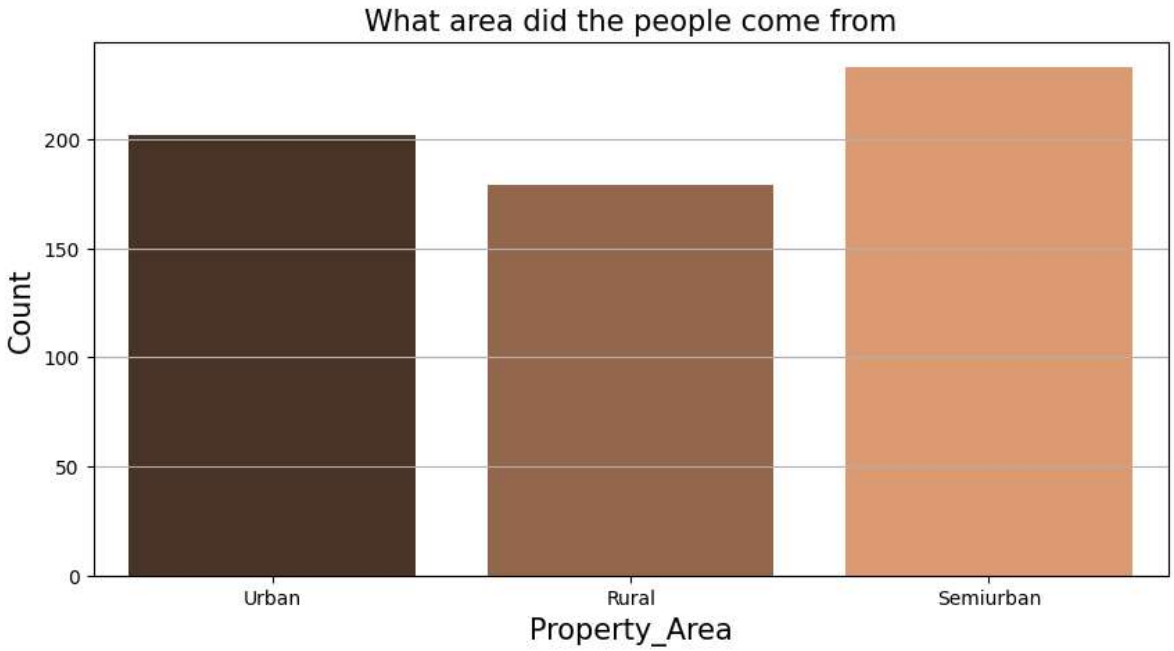
In [75]:
```
df["Property_Area"].unique()
```

Out[75]:
```
array(['Urban', 'Rural', 'Semiurban'], dtype=object)
```

In [76]:
```
df.Property_Area.value_counts()
```

Out[76]:
```
Property_Area
Semiurban    233
Urban        202
Rural        179
Name: count, dtype: int64
```

In [77]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="Property_Area",data=df,palette="copper")
plt.title("What area did the people come from",fontsize=15)
plt.xlabel("Property_Area",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```



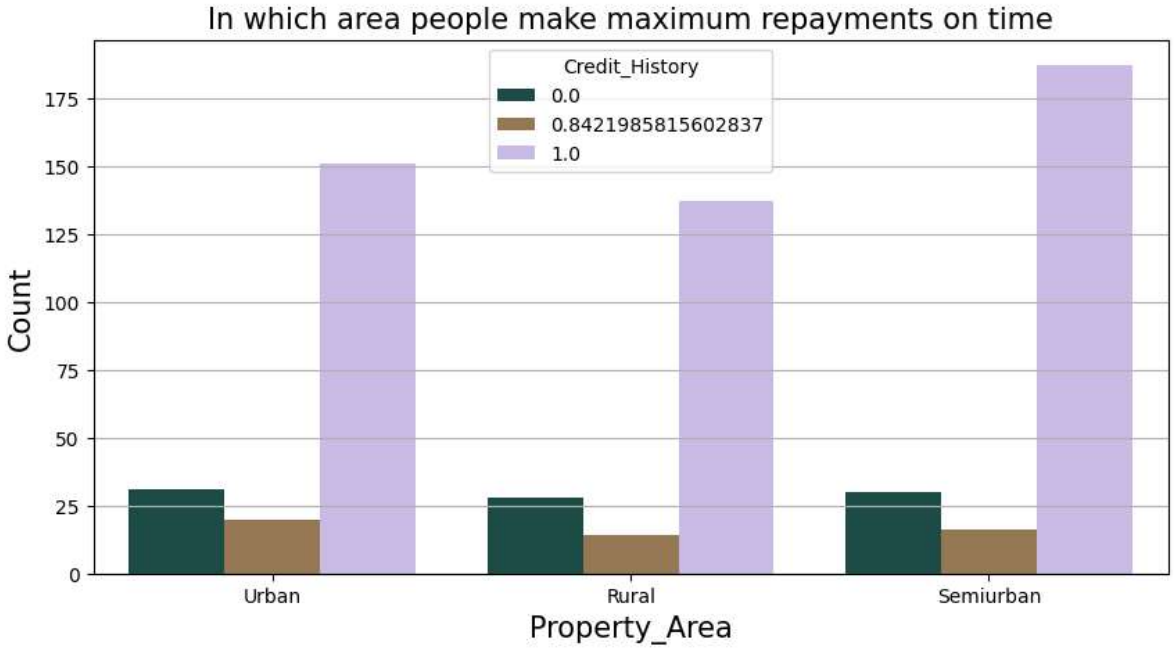
Observation:

* From above plot
* 200 people come from urban area
* 170 to 180 people come from rural area
* more than 200 people come from semiurban area

Compare colum Property_Area and Credit_History

Compare relation between in which area people are paid there repayments in 30 days

In [78]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="Property_Area",hue="Credit_History",data=df,palette="cubeheli
plt.title("In which area people make maximum repayments on time",fontsize=15)
plt.xlabel("Property_Area",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```



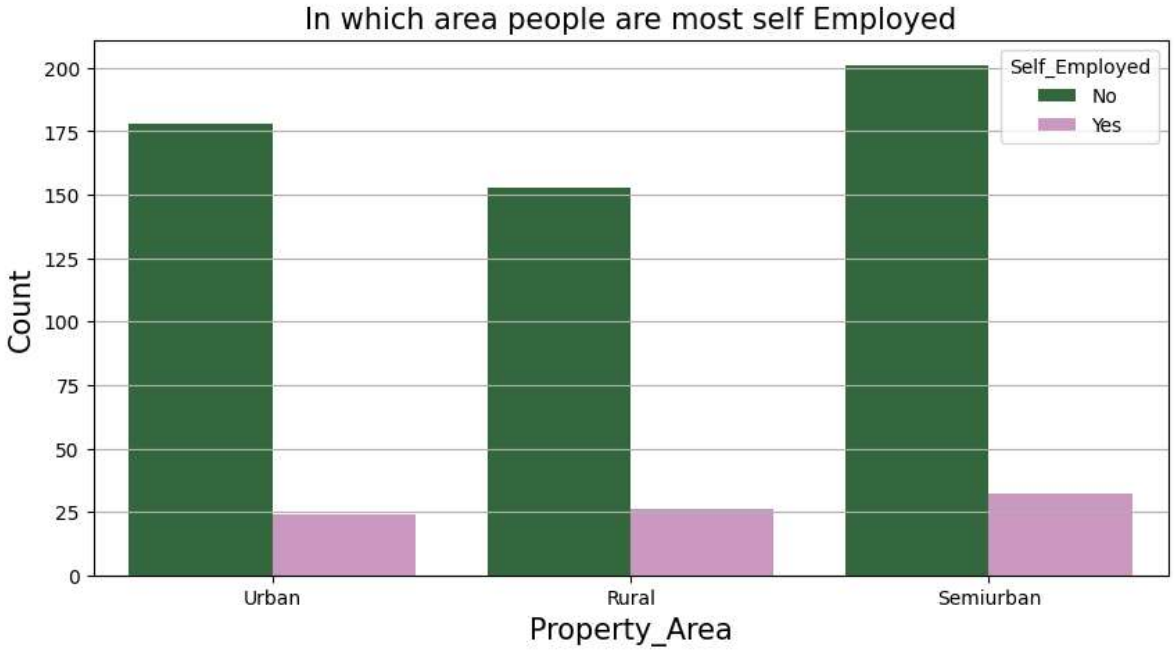
Observation:

```
    * Most People of Semiurban area make a repayments in 30days as com
pare to other thats why they have good credit historu
```

Compare colum Property_Area and Self_Employed

Compare relation between in which area people are Self Employed

In [79]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="Property_Area",hue="Self_Employed",data=df,palette="cubehelix
plt.title("In which area people are most self Employed",fontsize=15)
plt.xlabel("Property_Area",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```



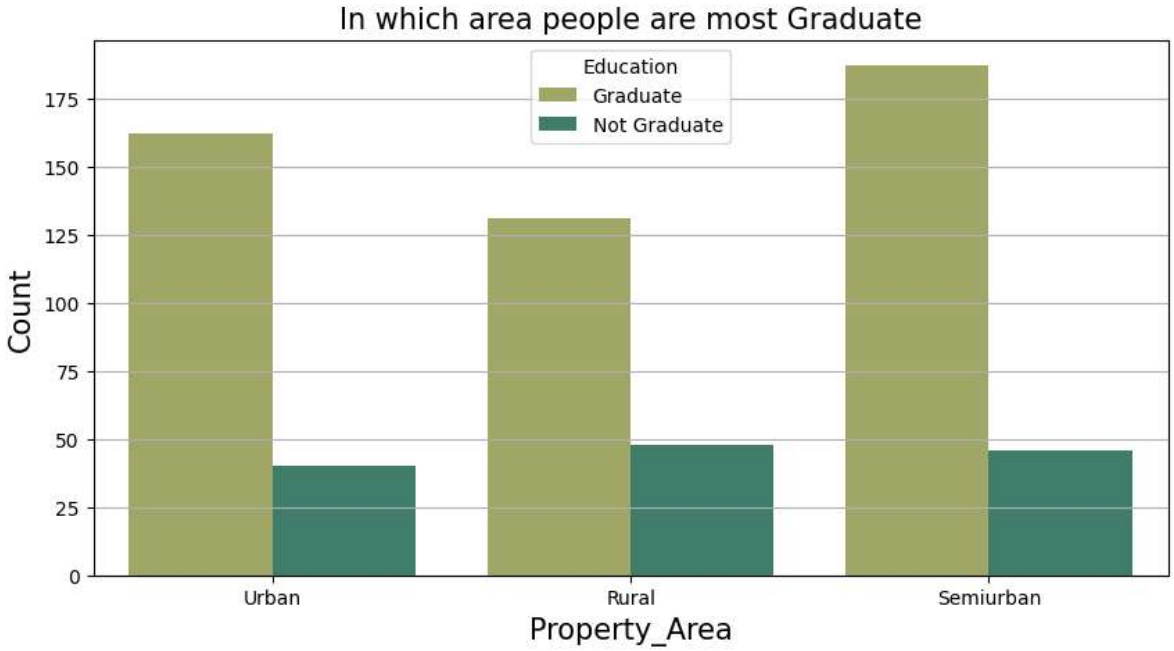
Observation:

* Most of the people are not Self Employed in three area

Compare colum Property_Area and Education

Compare relation between in which area people are most Graduate

In [80]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="Property_Area",hue="Education",data=df,palette="gist_earth_r"
plt.title("In which area people are most Graduate",fontsize=15)
plt.xlabel("Property_Area",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```



Observation:

```
    * More than 150 people are Graduate in urban area and between 25
to 35 people are Not Graduate in urban area.
    * More than 125 people are Graduate in Rural area and between 35
to 40 people are Not Graduate in Rural area.
    * More than 175 people are Graduate in Semiurban area and less th
an 50 people are Not Graduate in Semiurban                    area.
```

Analyse Loan_Status colum

In [81]:
```
df["Loan_Status"].unique()
```

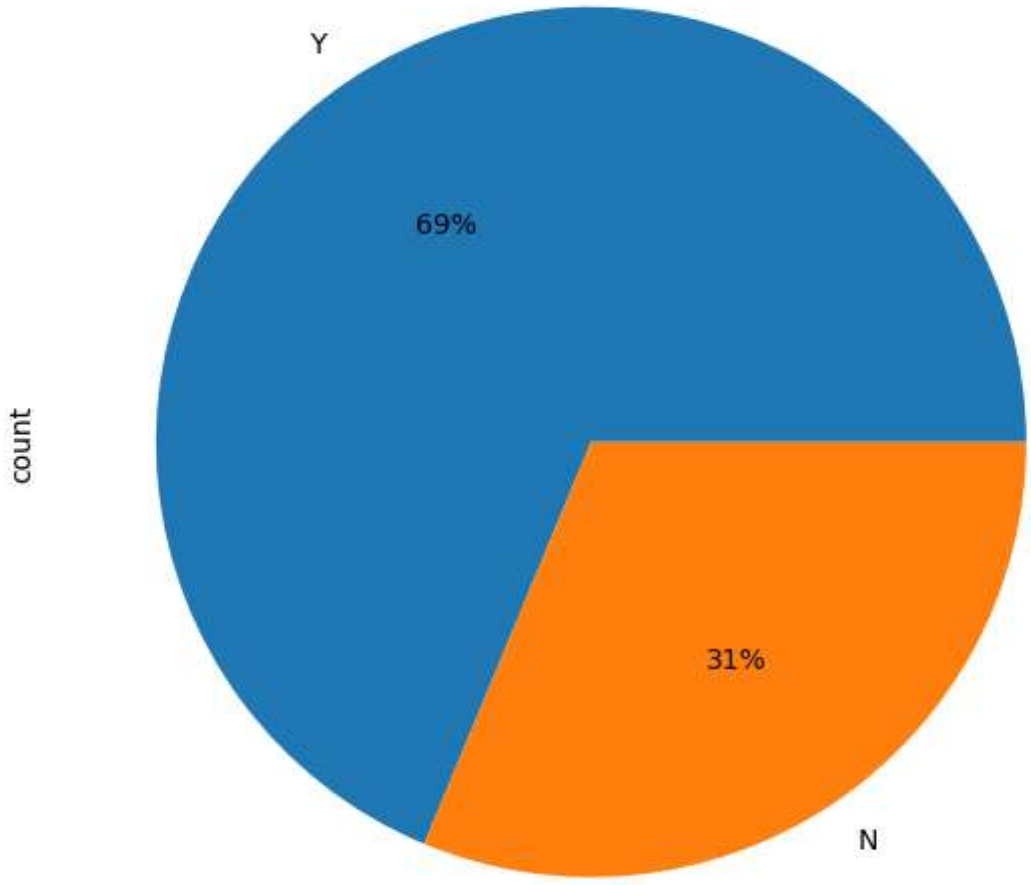
Out[81]:
```
array(['Y', 'N'], dtype=object)
```

In [82]:
```
df.Loan_Status.value_counts()  ### y means loan is approved and N means  loan
```

Out[82]:
```
Loan_Status
Y    422
N    192
Name: count, dtype: int64
```

In [83]:
```
plt.figure(figsize=(7,12))
df["Loan_Status"].value_counts().plot.pie(autopct="%.0f%%")
```

Out[83]: <Axes: ylabel='count'>



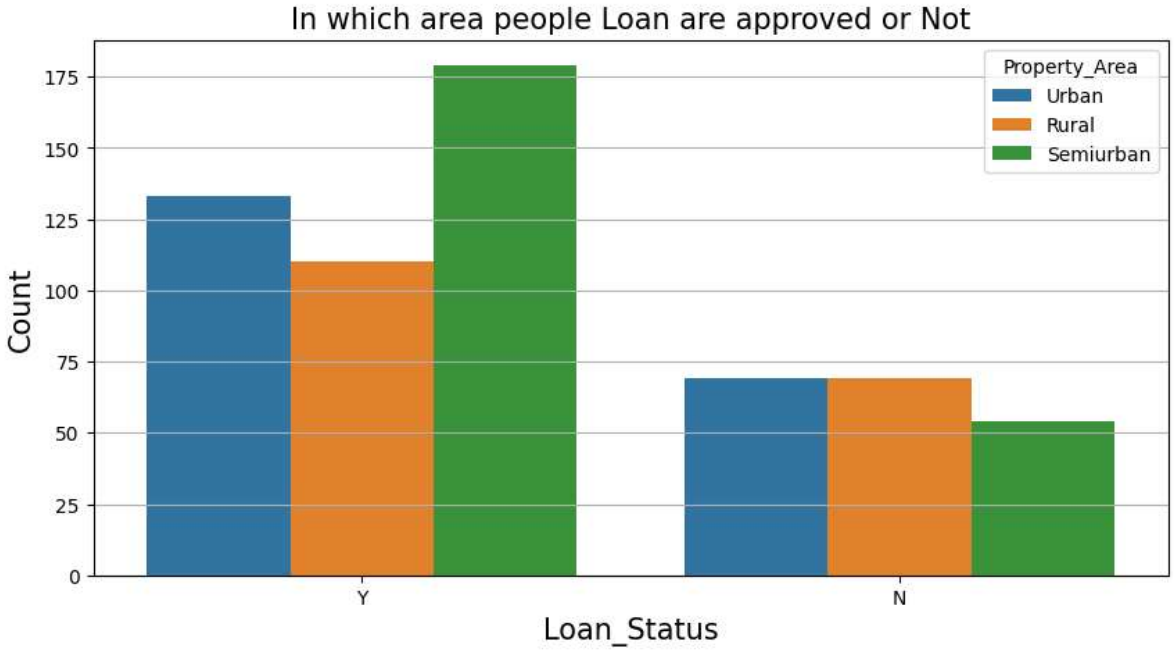
Observation:

* Above pie chart show
* Loan have been approved for 69% of the people
* Loan have been Not approved for 31% of the people

Compare colum Loan_Status and Property_Area

Compare relation between in which area people loan are approved

In [84]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="Loan_Status",hue="Property_Area",data=df)
plt.title("In which area people Loan are approved or Not",fontsize=15)
plt.xlabel("Loan_Status",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```

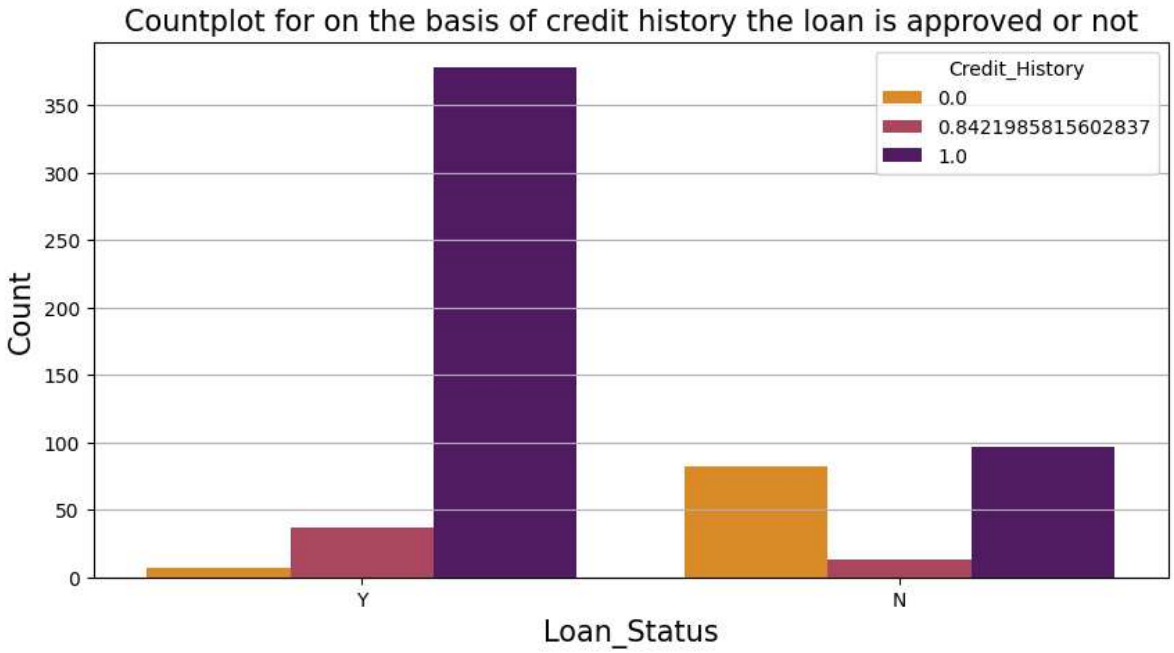


Observation:

* In all three area most of the people loan has been approved
* Loans of less than 75 people were not approved in all three area

Compare colum Loan_Status and Credit_History

In [85]:
```
plt.figure(figsize=(10,5))
sns.countplot(x="Loan_Status",hue="Credit_History",data=df,palette="inferno_r"
plt.title("Countplot for on the basis of credit history the loan is approved o
plt.xlabel("Loan_Status",fontsize=15)
plt.ylabel("Count",fontsize=15)
plt.grid(axis="y")
plt.show()
```



Observation:

* Most of the applicant credit history is one it means their loan is approved.

Finding the correlation of data

In [86]:
```
A = df.select_dtypes(include='number')
A
```

Out[86]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	5849	0.0	146.412162	360.0	1.0
1	4583	1508.0	128.000000	360.0	1.0
2	3000	0.0	66.000000	360.0	1.0
3	2583	2358.0	120.000000	360.0	1.0
4	6000	0.0	141.000000	360.0	1.0
...	...	...	...	...	...
609	2900	0.0	71.000000	360.0	1.0
610	4106	0.0	40.000000	180.0	1.0
611	8072	240.0	253.000000	360.0	1.0
612	7583	0.0	187.000000	360.0	1.0
613	4583	0.0	133.000000	360.0	0.0

614 rows × 5 columns

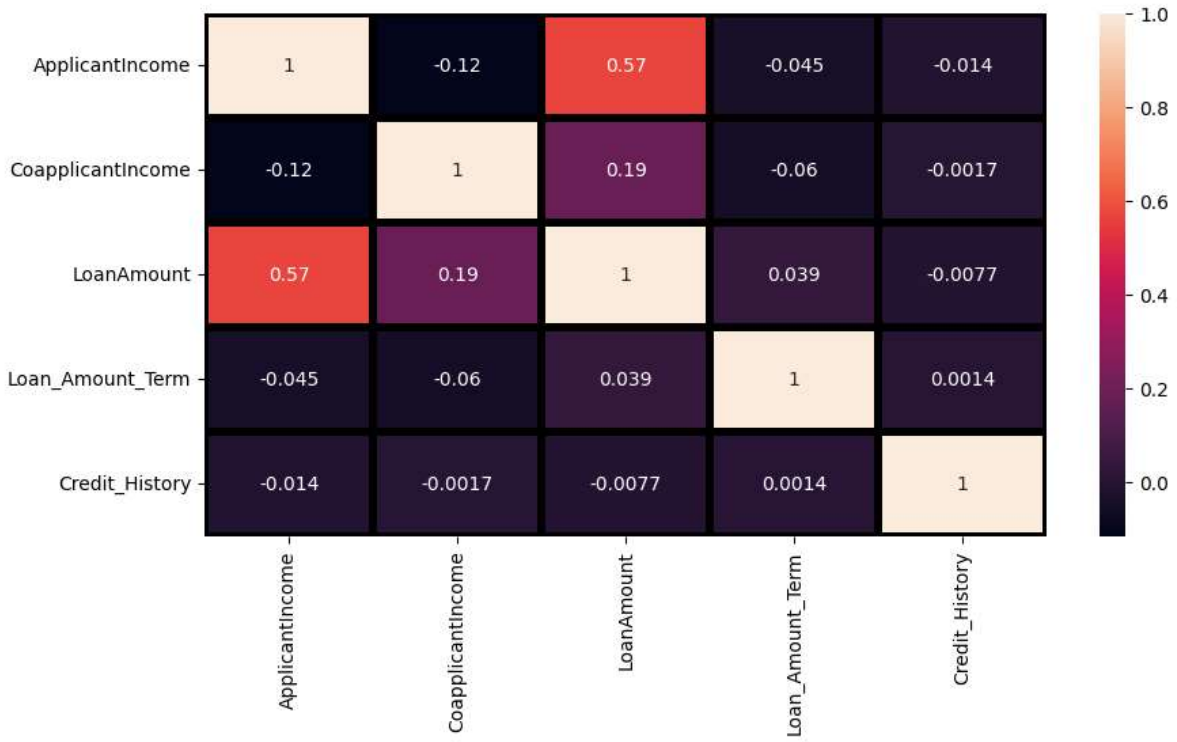
In [87]:
```
correlation=A.corr()
correlation
```

Out[87]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Cre
ApplicantIncome	1.000000	-0.116605	0.565620	-0.045242	
CoapplicantIncome	-0.116605	1.000000	0.187828	-0.059675	
LoanAmount	0.565620	0.187828	1.000000	0.038801	
Loan_Amount_Term	-0.045242	-0.059675	0.038801	1.000000	
Credit_History	-0.014477	-0.001665	-0.007738	0.001395	

In [88]:
```
plt.figure(figsize=(10,5))
sns.heatmap(A.corr(),annot=True,linewidths=4,linecolor="k")
```

Out[88]: <Axes: >



In [89]:
```
df.head()
```

Out[89]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coappl
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	

Feature Engineering

- We use here one hot encoding.
- Because there is multiple independent colum with categorical value.

In [90]:
```
## Drop Loan_ID colum no use of this colum

df2 =df.drop(columns=["Loan_ID"], axis=1)
df2
```

Out[90]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantInco
0	Male	No	0	Graduate	No	5849	
1	Male	Yes	1	Graduate	No	4583	150
2	Male	Yes	0	Graduate	Yes	3000	
3	Male	Yes	0	Not Graduate	No	2583	235
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	24
612	Male	Yes	2	Graduate	No	7583	
613	Female	No	0	Graduate	Yes	4583	

614 rows × 12 columns

In [91]:
```
# Convert all categorical colum in to numeric form ( 0 , 1)

colm=["Gender","Married","Dependents","Education","Self_Employed","ApplicantIr
dataset=pd.get_dummies(df2[colm],dtype=int)
```

In [92]:
```
dataset
```

Out[92]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Ger
0	5849	0.0	146.412162	360.0	1.0	
1	4583	1508.0	128.000000	360.0	1.0	
2	3000	0.0	66.000000	360.0	1.0	
3	2583	2358.0	120.000000	360.0	1.0	
4	6000	0.0	141.000000	360.0	1.0	
...	...	...	...	...	...	
609	2900	0.0	71.000000	360.0	1.0	
610	4106	0.0	40.000000	180.0	1.0	
611	8072	240.0	253.000000	360.0	1.0	
612	7583	0.0	187.000000	360.0	1.0	
613	4583	0.0	133.000000	360.0	0.0	

614 rows × 20 columns

In [93]:
```
from sklearn.preprocessing import LabelEncoder
```

In [94]:
```
# We use label encoding here for Loan_Status colum beacuse loan status has a c

Le=LabelEncoder()
```

In [95]:
```
df2["Loan_Status"]=Le.fit_transform(df2["Loan_Status"])
```

In [96]:
```
df2.head()
```

Out[96]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	Male	No	0	Graduate	No	5849	0.(
1	Male	Yes	1	Graduate	No	4583	1508.(
2	Male	Yes	0	Graduate	Yes	3000	0.(
3	Male	Yes	0	Not Graduate	No	2583	2358.(
4	Male	No	0	Graduate	No	6000	0.(

In [97]:
```
X = dataset.drop(columns=["Gender_Female",'Married_No','Dependents_0','Educati
Y = df2["Loan_Status"]
```

Spliting data into train and test set

In [98]:
```
from sklearn.model_selection import train_test_split
```

In [99]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.25)
```

In [100]:
```
X_train.shape
```

Out[100]: (460, 14)

In [101]:
```
Y_train.shape
```

Out[101]: (460,)

In [102]:
```
X_test.shape
```

Out[102]: (154, 14)

In [103]:
```
Y_test.shape
```

Out[103]: (154,)

In [104]:
```
from sklearn.linear_model import LogisticRegression
```

In [105]:
```
reg=LogisticRegression()
```

In [106]:
```
reg.fit(X_train,Y_train)
```

Out[106]:
```
LogisticRegression
LogisticRegression()
```

In [107]:
```
reg.score(X_test,Y_test)*100
```

Out[107]: 84.4155844155844

In [108]:
```
reg.score(X_train,Y_train)*100
```

Out[108]: 79.56521739130434

Use Support Vector Machine Algorithm

In [109]:
```
from sklearn.svm import SVC
```

In [110]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.25)
```

In [111]:
```
Model=SVC()
```

In [112]:
```
Model.fit(X_train,Y_train)
```

Out[112]:
```
SVC()
```

In [113]:
```
Model.score(X_test,Y_test)*100
```

Out[113]:
```
65.5844155844156
```

In [114]:
```
Model.score(X_train,Y_train)*100
```

Out[114]:
```
70.0
```

Use K-Nearest Neighbor(KNN) Algorithm

In [115]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [116]:
```
knn=KNeighborsClassifier(n_neighbors=5)
```

In [117]:
```
knn.fit(X_train,Y_train)
```

Out[117]:
```
KNeighborsClassifier()
```

In [118]:
```
knn.score(X_test,Y_test)*100
```

Out[118]:
```
61.68831168831169
```

In [119]:
```
knn.score(X_train,Y_train)*100
```

Out[119]:
```
75.0
```

In [ ]:

In [ ]: