Credential's

Name : Mayank Anand

Registration Number : 2141001045

In [63]:
```
# importing libraries

import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
```

Exploratory data analysis (EDA)

In [64]:
```
# to load the dataset

df = pd.read_csv("Training Dataset.csv")
```

In [65]:
```
# to view the first few data's of the dataframe

df.head()
```

Out[65]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

In [66]:
```
# to view the last few data's of the dataframe

df.tail()
```

Out[66]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

In [67]:
```
# to display about the total number of rows and columns in the dataset

df.shape
```

Out[67]:
```
(614, 13)
```

In [68]:
```
# to get info of dataframe

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 [69]:
```
# to get statistical information about the dataset

df.describe()
```

Out[69]:

	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 [70]:
```
# to get statistical information about the dataset along with the categorical column

df.describe(include='object')
```

Out[70]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Property_Area	Loan_Status
count	614	601	611	599	614	582	614	614
unique	614	2	2	4	2	2	3	2
top	LP001002	Male	Yes	0	Graduate	No	Semiurban	Y
freq	1	489	398	345	480	500	233	422

In [71]:
```
# to get to know the columns name

df.columns
```

Out[71]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [72]:
```
# to count the how many number of females and males are present

genderCount=df['Gender'].value_counts()
genderCount
```

Out[72]:
```
Male      489
Female    112
Name: Gender, dtype: int64
```

In [73]:
```
# to count how many are married and how many are unmaried

marriedCount=df['Married'].value_counts()
marriedCount
```

Out[73]:
```
Yes    398
No     213
Name: Married, dtype: int64
```

In [74]:
```
# to count how many dependents does the loan applier have

dependentCount=df['Dependents'].value_counts()
dependentCount
```

Out[74]:
```
0     345
1     102
2     101
3+     51
Name: Dependents, dtype: int64
```

In [75]:
```
# to count wheather the loan applier is graduated or not graduated

educationCount=df['Education'].value_counts()
educationCount
```

Out[75]:
```
Graduate        480
Not Graduate    134
Name: Education, dtype: int64
```

In [76]:
```
# to count wheather the person is self employed or not

selfEmployedCount=df['Self_Employed'].value_counts()
selfEmployedCount
```

Out[76]:
```
No     500
Yes     82
Name: Self_Employed, dtype: int64
```

In [77]:
```
df['Loan_Amount_Term'].value_counts()
```

Out[77]:
```
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: Loan_Amount_Term, dtype: int64
```

In [78]:
```
creditHistoryCount=df['Credit_History'].value_counts()
creditHistoryCount
```

Out[78]:
```
1.0    475
0.0     89
Name: Credit_History, dtype: int64
```

In [79]:
```
# to count the property type

propertyAreaCount=df['Property_Area'].value_counts()
propertyAreaCount
```

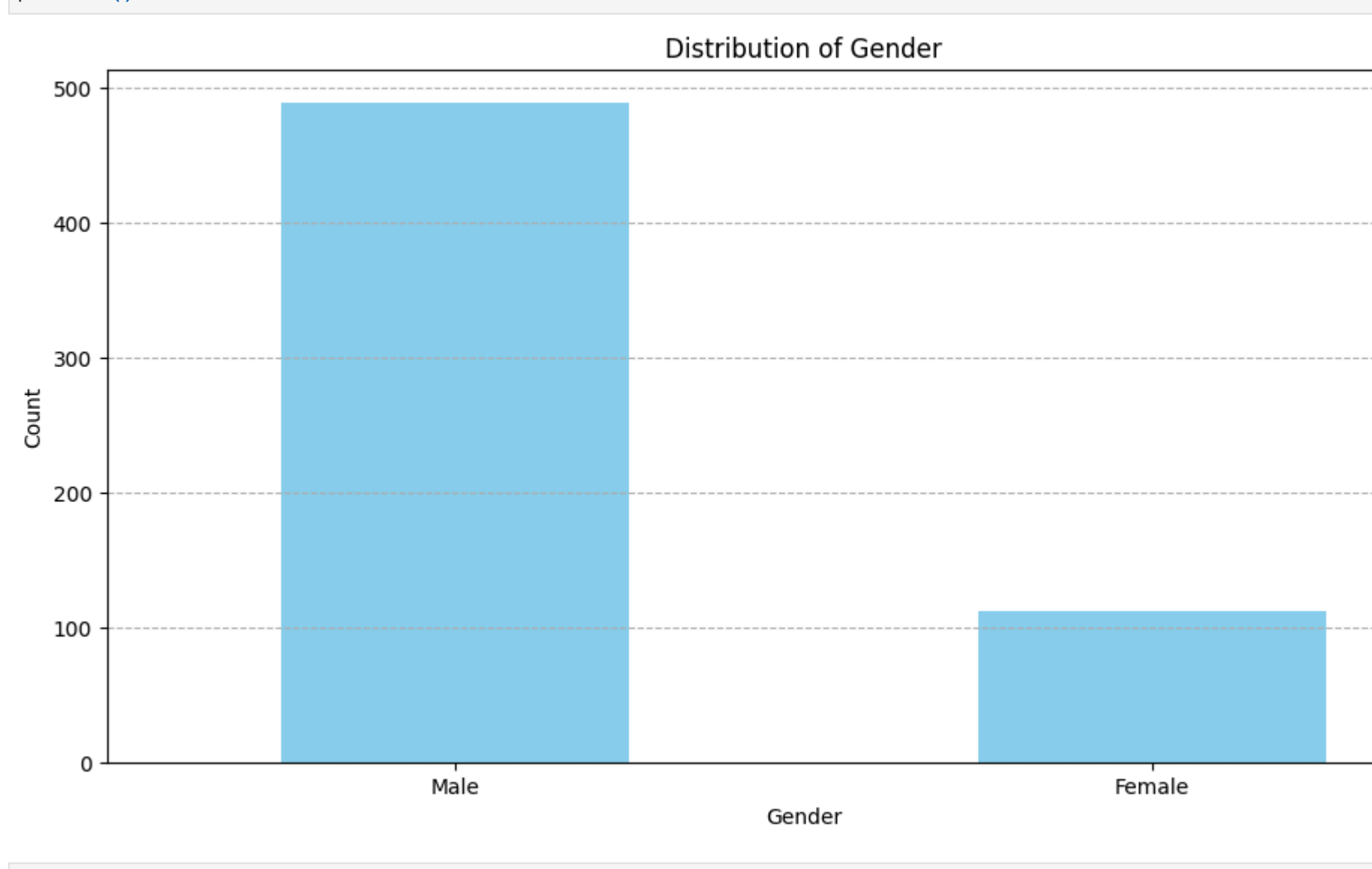
Out[79]:
```
Semiurban    233
Urban        202
Rural        179
Name: Property_Area, dtype: int64
```

In [80]:
```
# to count the loan status

loanStatusCount=df['Loan_Status'].value_counts()
loanStatusCount
```

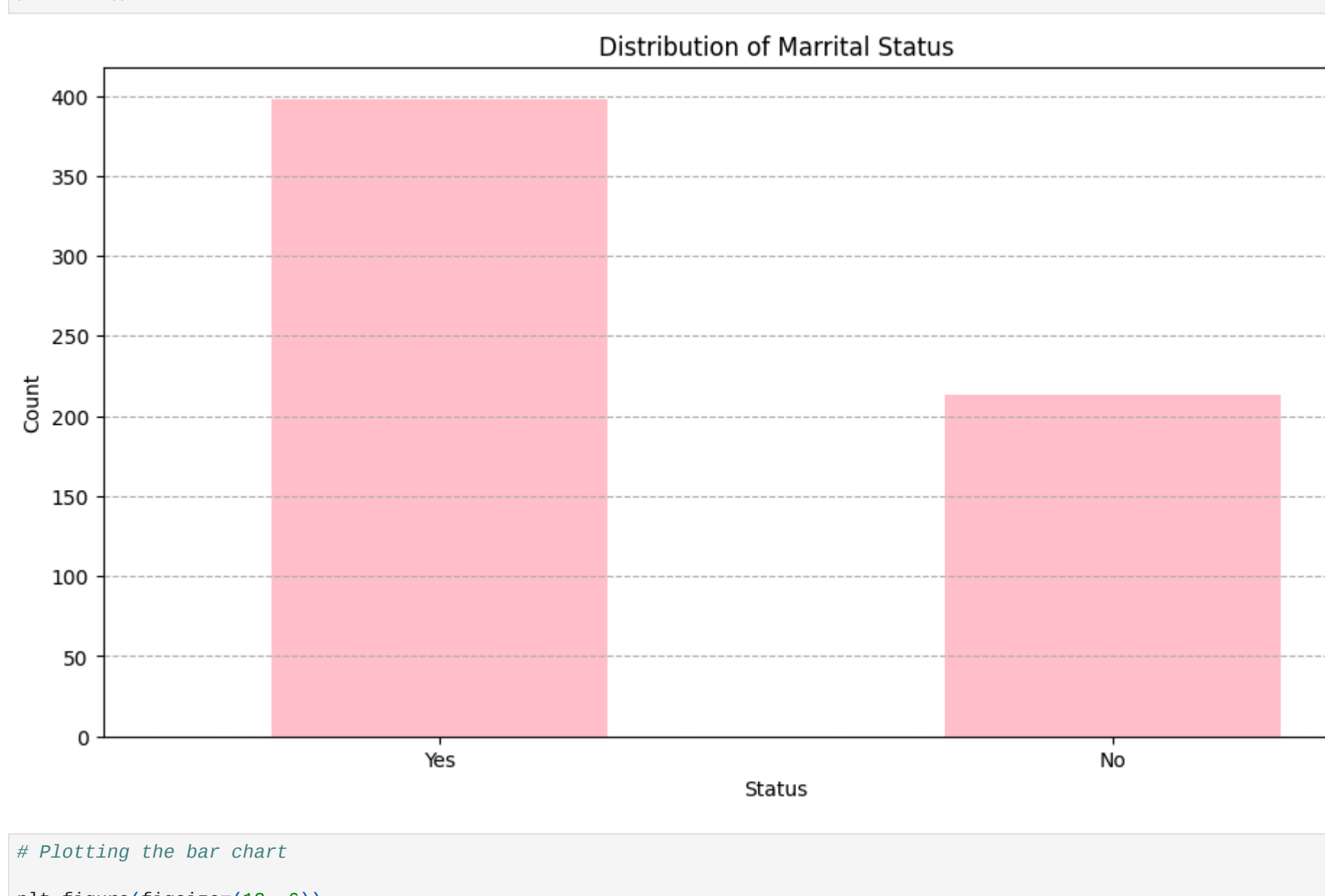
Out[80]:
```
Y    422
N    192
Name: Loan_Status, dtype: int64
```

Data Visualization

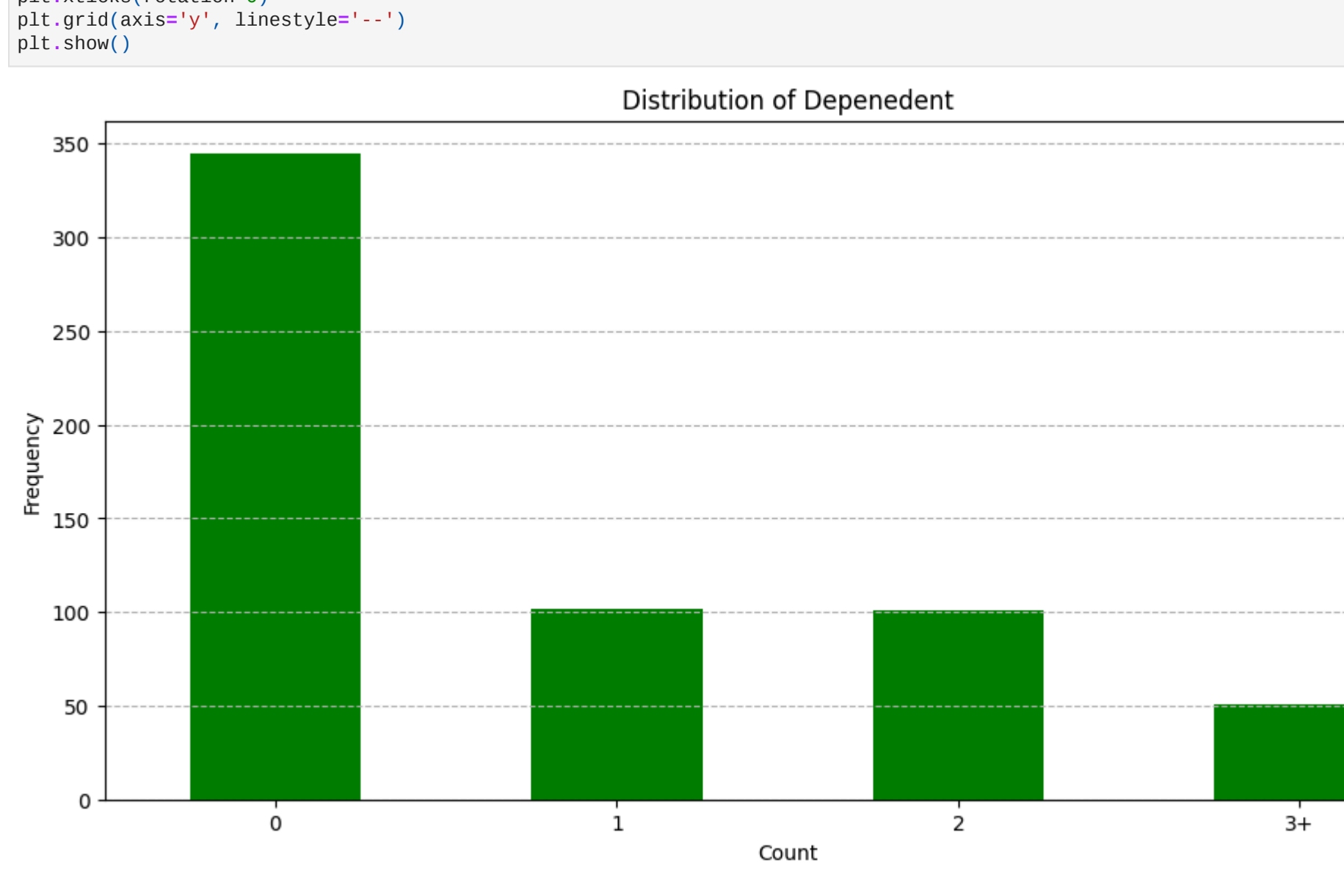
In [81]:
```
# Plotting the bar chart

plt.figure(figsize=(12, 6))
genderCount.plot(kind='bar', color='skyblue')
plt.title('Distribution of Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--')
plt.show()
```



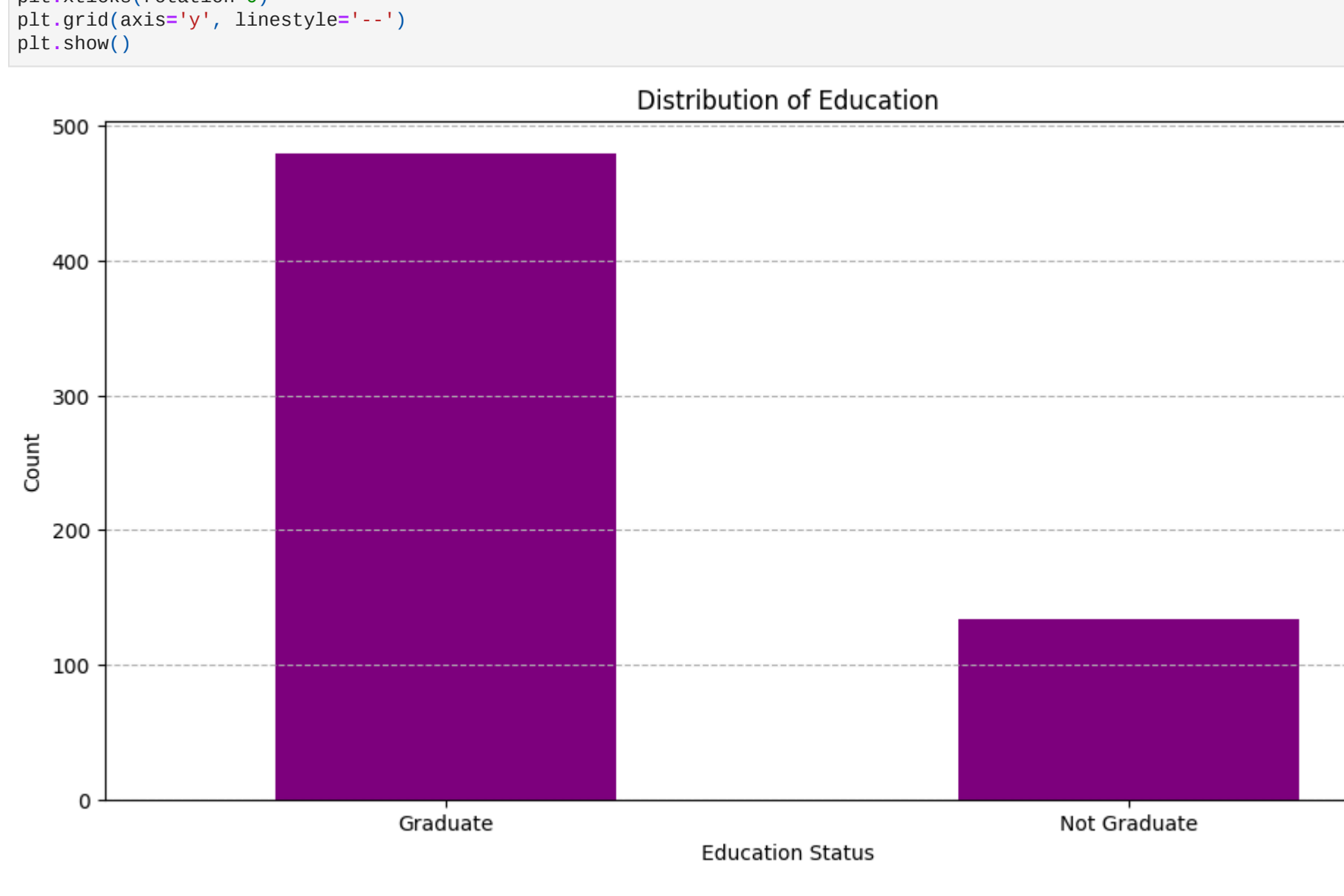
In [82]:
```
# Plotting the bar chart

plt.figure(figsize=(12, 6))
marriedCount.plot(kind='bar', color='pink')
plt.title('Distribution of Marrital Status')
plt.xlabel('Status')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--')
plt.show()
```



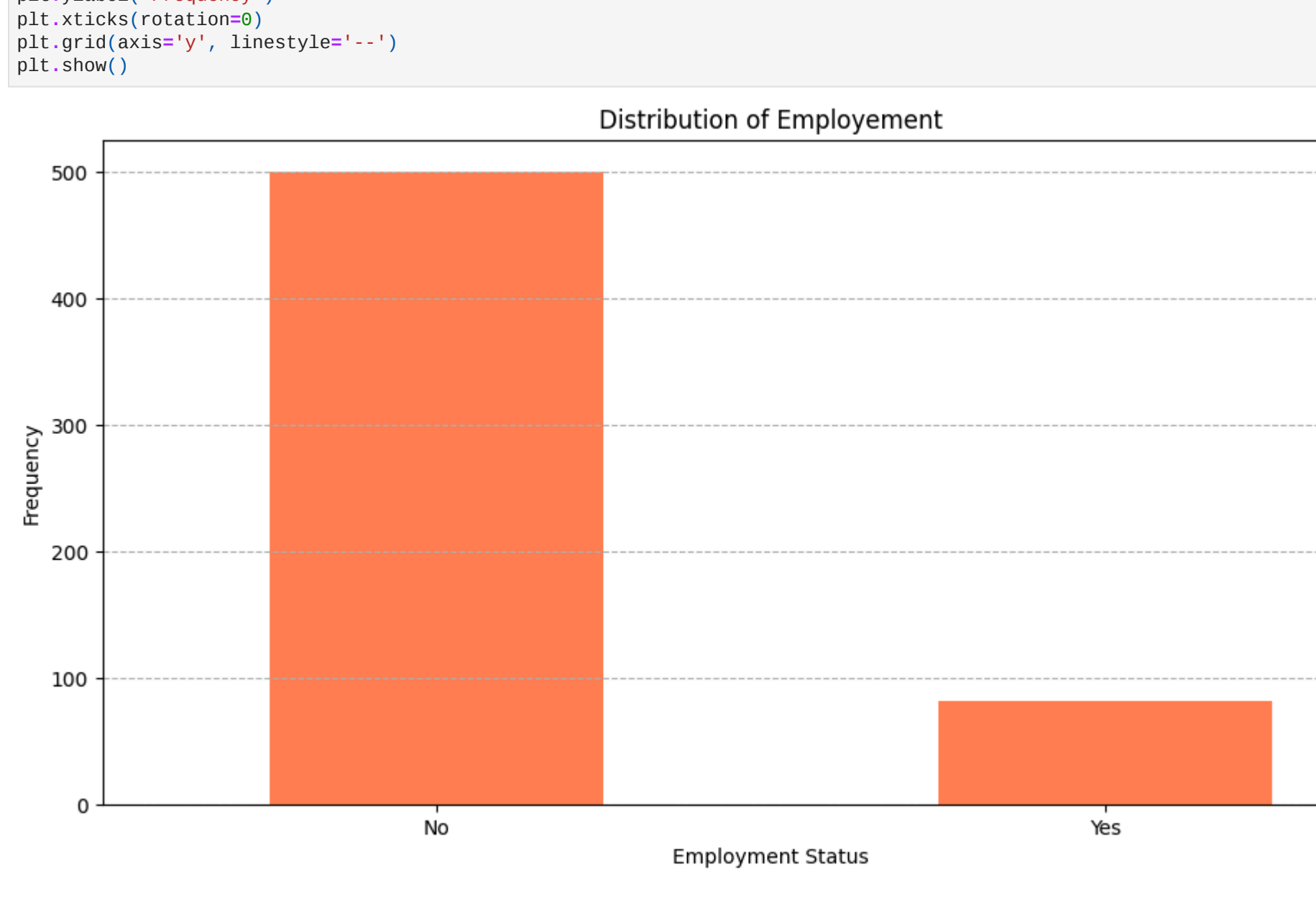
In [83]:
```
# Plotting the bar chart

plt.figure(figsize=(12, 6))
dependentCount.plot(kind='bar', color='green')
plt.title('Distribution of Depenedent')
plt.xlabel('Count')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--')
plt.show()
```



In [84]:
```
# Plotting the bar chart

plt.figure(figsize=(12, 6))
educationCount.plot(kind='bar', color='purple')
plt.title('Distribution of Education')
plt.xlabel('Education Status')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--')
plt.show()
```



In [85]:
```
# Plotting the bar chart

plt.figure(figsize=(12, 6))
selfEmployedCount.plot(kind='bar', color='coral')
plt.title('Distribution of Employement')
plt.xlabel('Employment Status')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--')
plt.show()
```



Handling Missing Values and Data Cleaning

In [86]:
```
df.columns
```

Out[86]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [87]:
```
# To check whether we have any null values present or not

df.isnull().sum()
```

Out[87]:
```
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 [88]:
```
# To copy the dataframe

df1=df.copy()
```

In [89]:
```
# To drop the unnessary column from the dataset

df1= df1.drop(columns=['Loan_ID'])
```

In [90]:
```
# To check the sahpe of the dataframe after droping it

df1.shape
```

Out[90]:
```
(614, 12)
```

In [91]:
```
# To replace the Loan Amount Column with its mean value

df1['LoanAmount']=df1['LoanAmount'].fillna(df1['LoanAmount'].mean())
```

In [92]:
```
df1.isnull().sum()
```

Out[92]:
```
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [93]:
```
# To replace the Loan Amount Term Column with its mean value

df1['Loan_Amount_Term']=df1['Loan_Amount_Term'].fillna(df1['Loan_Amount_Term'].mean())
```

In [94]:
```
df1.isnull().sum()
```

Out[94]:
```
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [95]:
```
# To replace the Credit History Column with its mean value

df1['Credit_History']=df1['Credit_History'].fillna(df1['Credit_History'].mean())
```

In [96]:
```
df1.isnull().sum()
```

Out[96]:
```
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History        0
Property_Area         0
Loan_Status           0
dtype: int64
```

In [97]:
```
# To replace the Gender Column with the mode

df1['Gender']=df1['Gender'].fillna(df1['Gender'].mode()[0])
```

In [98]:
```
df1.isnull().sum()
```

Out[98]:
```
Gender                0
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History        0
Property_Area         0
Loan_Status           0
dtype: int64
```

In [99]:
```
# To fill the missing value of this column with it's mode

df1['Married']=df1['Married'].fillna(df1['Married'].mode()[0])
```

In [100…
```
df1.isnull().sum()
```

Out[100]:
```
Gender                0
Married               0
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History        0
Property_Area         0
Loan_Status           0
dtype: int64
```

In [101…
```
# To fill the missing vlaue of this column with its mode

df1['Dependents']=df1['Dependents'].fillna(df1['Dependents'].mode()[0])
```

In [102…
```
df1.isnull().sum()
```

Out[102]:
```
Gender                0
Married               0
Dependents            0
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History        0
Property_Area         0
Loan_Status           0
dtype: int64
```

In [103…
```
# To fill the missing vlaue of this column with its mode

df1['Self_Employed']=df1['Self_Employed'].fillna(df1['Self_Employed'].mode()[0])
```

In [104…
```
df1.isnull().sum()
```

Out[104]:
```
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
```

Data Transformation & Feature Engineering

In [105…
```
df1.dtypes
```

Out[105]:
```
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 [106…
```
# Importing label encoder
from sklearn import preprocessing

# Creating label encoder object
label_encoder = preprocessing.LabelEncoder()
```

In [107…
```
# Converting the Gender Categorical column to Numerical

df1['Gender']= label_encoder.fit_transform(df1['Gender'])
```

In [108…
```
df1['Gender'].value_counts()
```

Out[108]:
```
1    502
0    112
Name: Gender, dtype: int64
```

In [109…
```
df1['Married']= label_encoder.fit_transform(df1['Married'])
```

In [110…
```
df1['Married'].value_counts()
```

Out[110]:
```
1    401
0    213
Name: Married, dtype: int64
```

In [111…
```
df1['Education']= label_encoder.fit_transform(df1['Education'])
```

In [112…
```
df1['Education'].value_counts()
```

Out[112]:
```
0    480
1    134
Name: Education, dtype: int64
```

In [113…
```
df1['Self_Employed']= label_encoder.fit_transform(df1['Self_Employed'])
```

In [114…
```
df1['Self_Employed'].value_counts()
```

Out[114]:
```
0    532
1     82
Name: Self_Employed, dtype: int64
```

In [115…
```
df1['Property_Area']= label_encoder.fit_transform(df1['Property_Area'])
```

In [116…
```
df1['Property_Area'].value_counts()
```

Out[116]:
```
1    233
2    202
0    179
Name: Property_Area, dtype: int64
```

In [117…
```
df1['Loan_Status']= label_encoder.fit_transform(df1['Loan_Status'])
```

In [118…
```
df1['Loan_Status'].value_counts()
```

Out[118]:
```
1    422
0    192
Name: Loan_Status, dtype: int64
```

In [119…
```
df1.dtypes
```

Out[119]:
```
Gender                 int32
Married                int32
Dependents            object
Education              int32
Self_Employed          int32
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area          int32
Loan_Status            int32
dtype: object
```

In [120…
```
df1.head(10)
```

Out[120]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	1	0	0	0	0	5849	0.0	146.412162	360.0	1.0	2	1
1	1	1	1	0	0	4583	1508.0	128.000000	360.0	1.0	0	0
2	1	1	0	0	1	3000	0.0	66.000000	360.0	1.0	2	1
3	1	1	0	1	0	2583	2358.0	120.000000	360.0	1.0	2	1
4	1	0	0	0	0	6000	0.0	141.000000	360.0	1.0	2	1
5	1	1	2	0	1	5417	4196.0	267.000000	360.0	1.0	2	1
6	1	1	0	1	0	2333	1516.0	95.000000	360.0	1.0	2	1
7	1	1	3+	0	0	3036	2504.0	158.000000	360.0	0.0	1	0
8	1	1	2	0	0	4006	1526.0	168.000000	360.0	1.0	2	1
9	1	1	1	0	0	12841	10968.0	349.000000	360.0	1.0	1	0