In [1]:

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

In [2]:

```
data = pd.read_csv("loan_prediction.csv")
```

In [4]:

```
data.head
```

Out[4]:

```
<bound method NDFrame.head of       Loan_ID  Gender Married Dependents
Education Self_Employed  \
0    LP001002    Male      No          0      Graduate            No
1    LP001003    Male     Yes          1      Graduate            No
2    LP001005    Male     Yes          0      Graduate           Yes
3    LP001006    Male     Yes          0  Not Graduate            No
4    LP001008    Male      No          0      Graduate            No
..        ...     ...     ...        ...           ...           ...
609  LP002978  Female      No          0      Graduate            No
610  LP002979    Male     Yes         3+      Graduate            No
611  LP002983    Male     Yes          1      Graduate            No
612  LP002984    Male     Yes          2      Graduate            No
613  LP002990  Female      No          0      Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0               5849                0.0         NaN             360.0
1               4583             1508.0       128.0             360.0
2               3000                0.0        66.0             360.0
3               2583             2358.0       120.0             360.0
4               6000                0.0       141.0             360.0
..               ...                ...         ...               ...
609             2900                0.0        71.0             360.0
610             4106                0.0        40.0             180.0
611             8072              240.0       253.0             360.0
612             7583                0.0       187.0             360.0
613             4583                0.0       133.0             360.0

     Credit_History Property_Area Loan_Status
0               1.0         Urban           Y
1               1.0         Rural           N
2               1.0         Urban           Y
3               1.0         Urban           Y
4               1.0         Urban           Y
..              ...           ...         ...
609             1.0         Rural           Y
610             1.0         Rural           Y
611             1.0         Urban           Y
612             1.0         Urban           Y
613             0.0     Semiurban           N

[614 rows x 13 columns]>
```

In [5]:

```

obj = (data.dtypes == 'object')
print("Categorical variables:",len(list(obj[obj].index)))
```

```
Categorical variables: 8
```

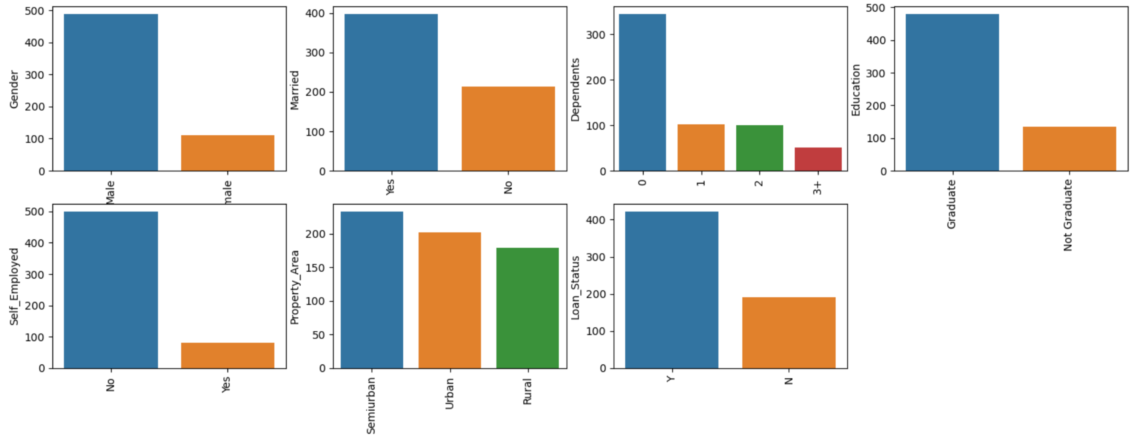
In [6]:

```
data.drop(['Loan_ID'],axis=1,inplace=True)
```

In [7]:

```
obj = (data.dtypes == 'object')
object_cols = list(obj[obj].index)
plt.figure(figsize=(18,36))
index = 1

for col in object_cols:
  y = data[col].value_counts()
  plt.subplot(11,4,index)
  plt.xticks(rotation=90)
  sns.barplot(x=list(y.index), y=y)
  index +=1
```



In [8]:

```
from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()
obj = (data.dtypes == 'object')
for col in list(obj[obj].index):
  data[col] = label_encoder.fit_transform(data[col])
```

In [9]:

```
obj = (data.dtypes == 'object')
print("Categorical variables:",len(list(obj[obj].index)))
```

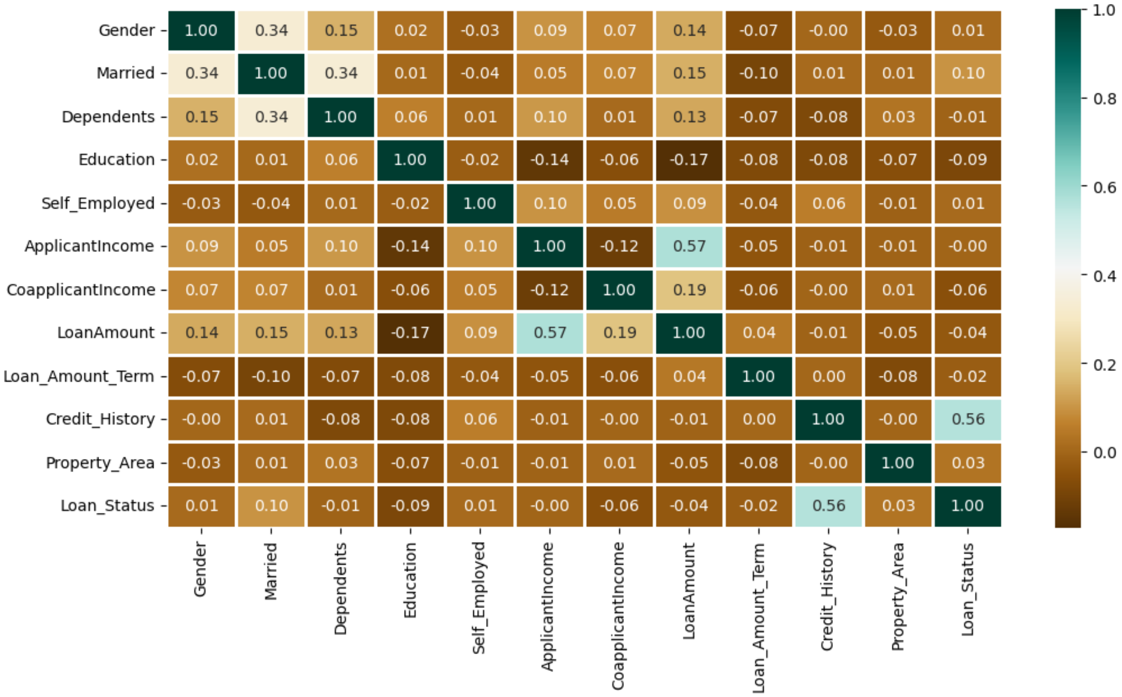
```
Categorical variables: 0
```

In [10]:

```
plt.figure(figsize=(12,6))

sns.heatmap(data.corr(),cmap='BrBG',fmt='.2f',
            linewidths=2,annot=True)
```

Out[10]: `<Axes: >`



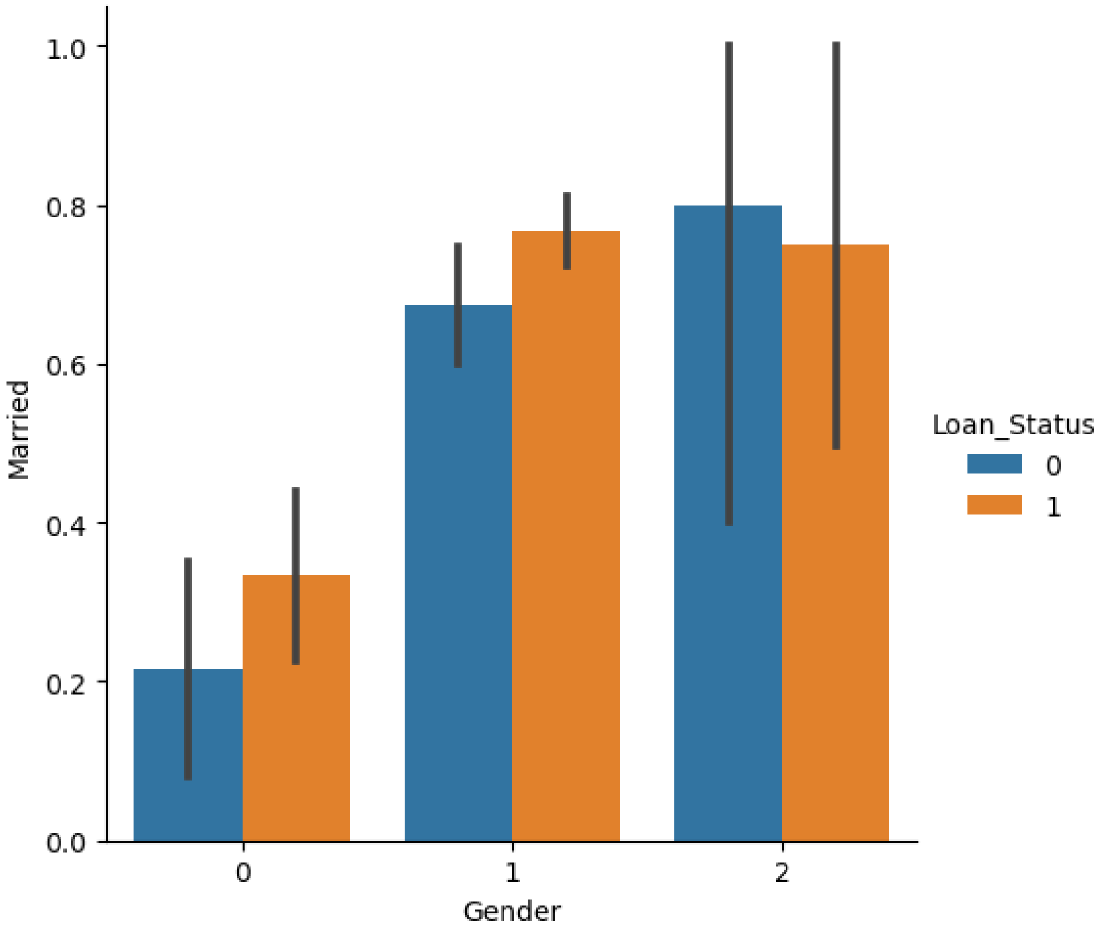
	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
Gender	1.00	0.34	0.15	0.02	-0.03	0.09	0.07	0.14	-0.07	-0.00	-0.03	0.01
Married	0.34	1.00	0.34	0.01	-0.04	0.05	0.07	0.15	-0.10	0.01	0.01	0.10
Dependents	0.15	0.34	1.00	0.06	0.01	0.10	0.01	0.13	-0.07	-0.08	0.03	-0.01
Education	0.02	0.01	0.06	1.00	-0.02	-0.14	-0.06	-0.17	-0.08	-0.08	-0.07	-0.09
Self_Employed	-0.03	-0.04	0.01	-0.02	1.00	0.10	0.05	0.09	-0.04	0.06	-0.01	0.01
ApplicantIncome	0.09	0.05	0.10	-0.14	0.10	1.00	-0.12	0.57	-0.05	-0.01	-0.01	-0.00
CoapplicantIncome	0.07	0.07	0.01	-0.06	0.05	-0.12	1.00	0.19	-0.06	-0.00	0.01	-0.06
LoanAmount	0.14	0.15	0.13	-0.17	0.09	0.57	0.19	1.00	0.04	-0.01	-0.05	-0.04
Loan_Amount_Term	-0.07	-0.10	-0.07	-0.08	-0.04	-0.05	-0.06	0.04	1.00	0.00	-0.08	-0.02
Credit_History	-0.00	0.01	-0.08	-0.08	0.06	-0.01	-0.00	-0.01	0.00	1.00	-0.00	0.56
Property_Area	-0.03	0.01	0.03	-0.07	-0.01	-0.01	0.01	-0.05	-0.08	-0.00	1.00	0.03
Loan_Status	0.01	0.10	-0.01	-0.09	0.01	-0.00	-0.06	-0.04	-0.02	0.56	0.03	1.00

In [11]:

```
sns.catplot(x="Gender", y="Married",
            hue="Loan_Status",
            kind="bar",
            data=data)
```

Out[11]:

```
<seaborn.axisgrid.FacetGrid at 0x169e3451050>
```



In [12]:

```
for col in data.columns:
  data[col] = data[col].fillna(data[col].mean())

data.isna().sum()
```

Out[12]:

```
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 [13]:

```
from sklearn.model_selection import train_test_split

X = data.drop(['Loan_Status'],axis=1)
Y = data['Loan_Status']
X.shape,Y.shape

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    test_size=0.4,
                                                    random_state=1)
X_train.shape, X_test.shape, Y_train.shape, Y_test.shape
```

Out[13]:

```
((368, 11), (246, 11), (368,), (246,))
```

In [14]:

```

from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression

from sklearn import metrics

knn = KNeighborsClassifier(n_neighbors=3)
rfc = RandomForestClassifier(n_estimators = 7,
                             criterion = 'entropy',
                             random_state =7)
svc = SVC()
lc = LogisticRegression()

# making predictions on the training set
for clf in (rfc, knn, svc,lc):
    clf.fit(X_train, Y_train)
    Y_pred = clf.predict(X_train)
    print("Accuracy score of ",
          clf.__class__.__name__,
          "=",100*metrics.accuracy_score(Y_train,
                                         Y_pred))
```

```
Accuracy score of  RandomForestClassifier = 97.01086956521739
Accuracy score of  KNeighborsClassifier = 77.17391304347827
Accuracy score of  SVC = 70.38043478260869
Accuracy score of  LogisticRegression = 82.88043478260869

C:\Users\lenovo\anaconda3\Lib\site-packages\sklearn\linear_model\_logisti
c.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://s
cikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression (https://scikit-learn.org/stable/modules/linear_model.html#logis
tic-regression)
  n_iter_i = _check_optimize_result(
```

In [15]:

```
for clf in (rfc, knn, svc,lc):
    clf.fit(X_train, Y_train)
    Y_pred = clf.predict(X_test)
    print("Accuracy score of ",
          clf.__class__.__name__,"=",
          100*metrics.accuracy_score(Y_test,
                                      Y_pred))
```

```
Accuracy score of  RandomForestClassifier = 76.42276422764228
Accuracy score of  KNeighborsClassifier = 62.19512195121951
Accuracy score of  SVC = 67.07317073170732
Accuracy score of  LogisticRegression = 78.86178861788618

C:\Users\lenovo\anaconda3\Lib\site-packages\sklearn\linear_model\_logisti
c.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://s
cikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression (https://scikit-learn.org/stable/modules/linear_model.html#logis
tic-regression)
  n_iter_i = _check_optimize_result(
```

In [ ]:

In [ ]: