Load Dataset Analysis

We will perform data analysis and will explore the Loan Dataset that w downloaded from kaggle.com

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

Read and Load CSV Dataset

In [33]:
```
df = pd.read_csv("loan_data_set.csv")
df
```

Out[33]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Pro
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	

614 rows × 13 columns

In [11]:
```
# summary of data
df.describe()
```

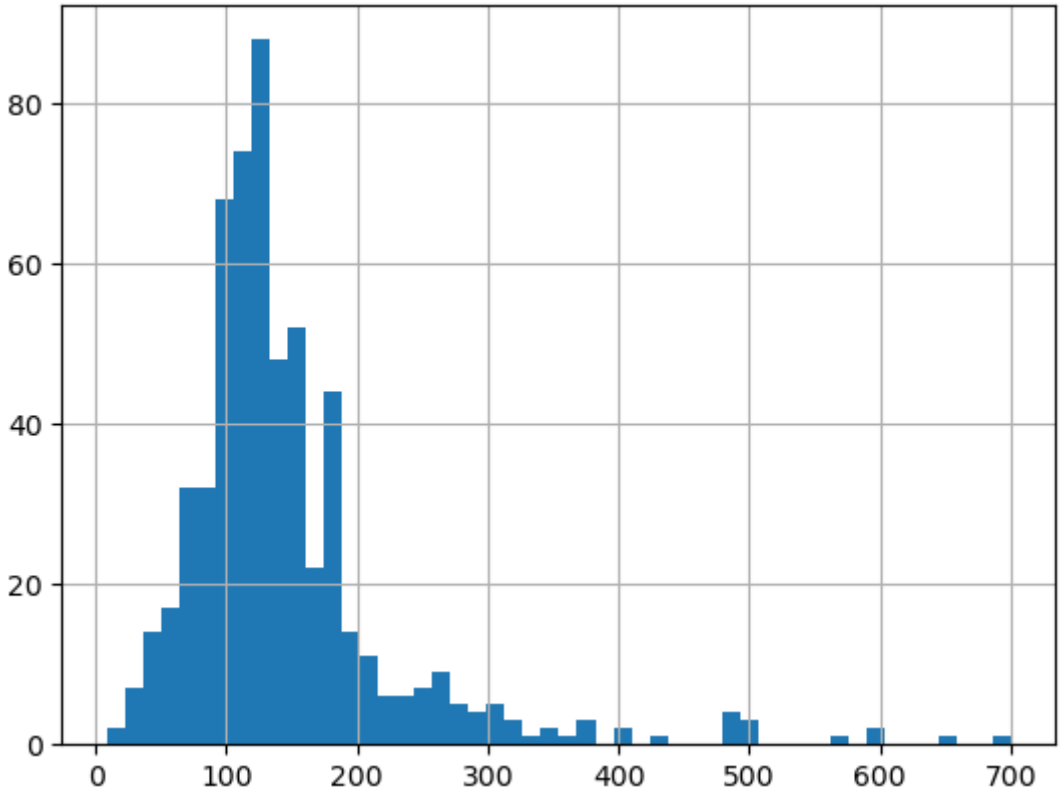
Out[11]:

	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

Visualize Data

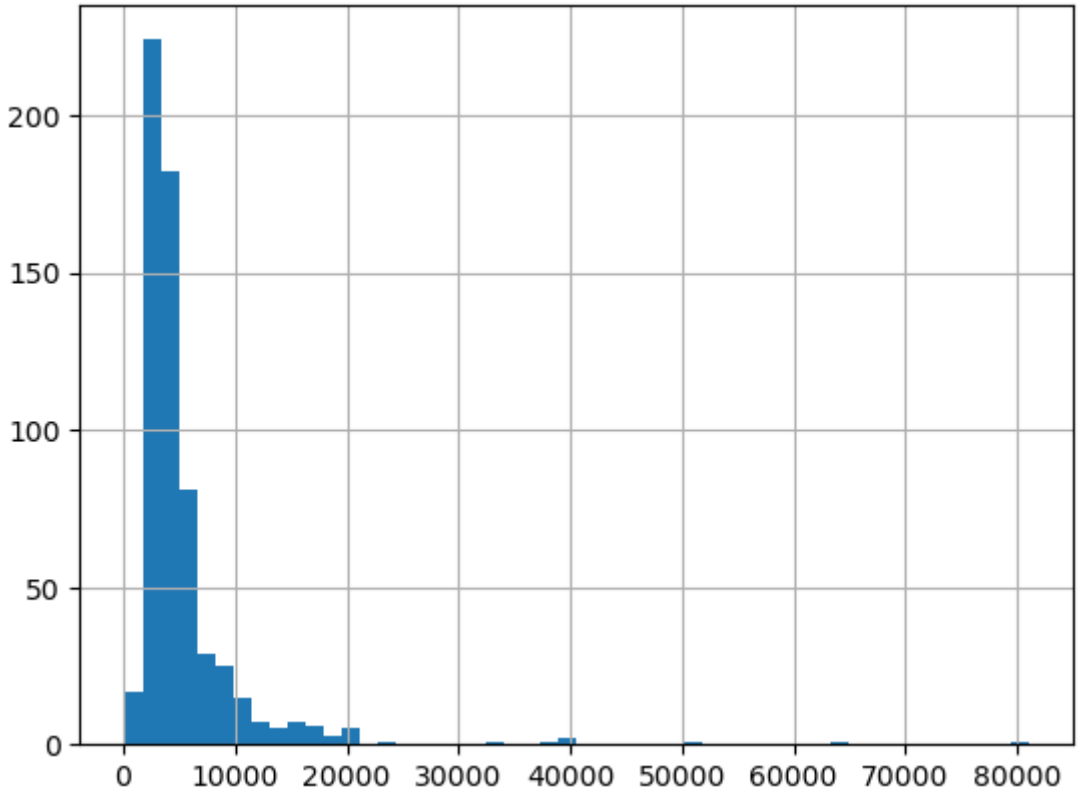
In [20]:
```
df['LoanAmount'].hist(bins=50)   #there are some extreme values
```

Out[20]:
```
<Axes: >
```



In [21]:
```
df['ApplicantIncome'].hist(bins=50)   #there are some extreme values
```

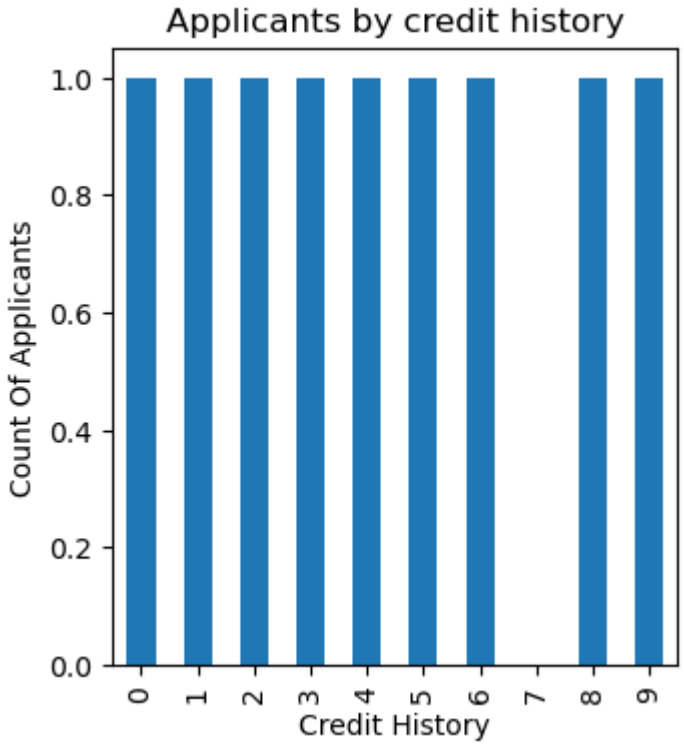
Out[21]: <Axes: >



In [30]:
```
# checking credit hsitory
fig = plt.figure(figsize=(8,4))
ax1 = fig.add_subplot(121)
ax1.set_xlabel('Credit History')
ax1.set_ylabel('Count Of Applicants')
ax1.set_title("Applicants by credit history")


df['Credit_History'][:10].plot(kind="bar")
```

Out[30]:
```
<Axes: title={'center': 'Applicants by credit history'}, xlabel='Credit History', ylabel='Count Of Applicants'>
```



Data wrangling/cleaning

In [51]:
```
df.isnull().sum()
```

Out[51]:
```
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 [65]:
```
df['Dependents'].fillna(df['Dependents'].mode()[0], inplace=True)
```

In [72]:
```
df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
```

In [73]:
```
df['Married'].fillna(df['Married'].mode()[0], inplace=True)
```

In [75]:
```
df['LoanAmount'].fillna(df['LoanAmount'].mean(), inplace=True)
```

In [76]:
```
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean(), inplace=True)
```

In [78]:
```
df['Credit_History'].fillna(df['Credit_History'].mean(), inplace=True)
```

In [79]:
```
df.isnull().sum()
```

Out[79]:
```
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 [80]:
```
df.dtypes
```

Out[80]:
```
Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
Loan_Status           object
dtype: object
```

Model Building

In [82]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import KFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier,export_graphviz
from sklearn import metrics
```

In [83]:
```
df
```

Out[83]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Pro
0	LP001002	Male	No	0	Graduate	No	5849	0.0	146.412162	360.0	1.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.000000	360.0	1.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.000000	360.0	1.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.000000	360.0	1.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.000000	360.0	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.000000	360.0	1.0	
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.000000	180.0	1.0	
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.000000	360.0	1.0	
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.000000	360.0	1.0	
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.000000	360.0	0.0	

614 rows × 13 columns

Extract Independent and Dependent

In [92]:
```
# selects all rows (:) and the columns specified in the list ['LoanAmount', 'Credit_History'].
# It returns a new DataFrame with only these selected columns.

# Independent Variables
X = df.loc[:, ['LoanAmount','Credit_History']].values
```

```
# Dependent Variables
y = df.loc[:, ['Loan_Status']].values
```

Split Data into Training and Testing

In [93]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

Filling Data into Logistic Regression

In [97]:
```
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)
```

```
D:\apps\anaconda\files\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d
ray was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[97]:
```
LogisticRegression
LogisticRegression(random_state=
```

Predicting the Result

In [100…
```
# predicting the test set results
y_pred = classifier.predict(X_test)
y_pred
```

Out[100]:
```
array(['Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y',
       'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'N'],
      dtype=object)
```

Performance of the Model

In [103…]

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

cm

# the output is as below
"""
 ---------------------------------
|Predicted|    No   |  Yes   | Total |
|---------|         |        |       |
|Actual   |         |        |       |
|---------|---------|--------|------ |
|No       |   19    |  24    |  43   |
|---------|---------|--------|-------|
|Yes      |    2    |  109   |  111  |
|---------|---------|--------|-------|
|Total    |   21    |  133   |  154  |
 ---------------------------------

 along the diagonal (19, 109) are the correct values. and these should be greaer
 than the othee diagonal, that will determine how good model accuracy is

 Accuracy:
 (True Positive + True Negative)/total = (109+19)/154 = 0.83 is the accuracy

 Precision:
 True Positive/total predicted Yes = 109/133 = 0.81 is precision
"""
```

Out[103]:

```
array([[ 19,  24],
       [  2, 109]], dtype=int64)
```

Calculate Accuracy using scikit

In [111…]

```
from sklearn.metrics import accuracy_score

accuracy_score(y_test, y_pred)
```

Out[111]:

```
0.8311688311688312
```

In [118…]

```
y_test
```

Out[118]:
```
array([['Y'],
       ['N'],
       ['Y'],
       ['N'],
       ['Y'],
       ['N'],
       ['Y'],
       ['Y'],
       ['N'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['N'],
       ['N'],
       ['Y'],
       ['Y'],
       ['N'],
       ['N'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['N'],
       ['N'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['N'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['N'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['Y'],
```

```
['N'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['N'],
['Y'],
['Y'],
['Y'],
['N'],
['Y'],
['N'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['N'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['N'],
['N'],
['Y'],
['N'],
['Y'],
['N'],
['N'],
['Y'],
['N'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['N'],
['N'],
['N'],
['Y'],
['N'],
```

['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['N'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['N'],
['Y'],
['N'],
['N'],
['Y'],
['N'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['N'],
['Y'],
['Y'],
['Y'],
['N'],
['N'],
['Y'],
['Y'],
['N'],
['Y'],
['Y'],
['Y'],
['Y'],
['N'],
['N'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['Y'],
['N'],
['Y'],

```
       ['Y'],
       ['N'],
       ['Y'],
       ['Y'],
       ['N'],
       ['N'],
       ['Y'],
       ['Y'],
       ['Y'],
       ['N']], dtype=object)
```

In [119…]
```
y_pred
```

Out[119]:
```
array(['Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y',
       'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'N'],
      dtype=object)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: