# Data Processing with Pandas Casestudy

**Name: PRIYESHWAR**

**Mail: priyesh2664@gmail.com**

---

Automate the loan eligibility process (real-time) based on customer detail provided while filling the online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History, and others.

The major aim of this notebook is to predict which of the customers will have their loan approved.

---

● **Loading Data in Pandas DataFrame**

Load the loan dataset from a CSV file and display its contents.

**data=pd.read_csv('LoanData.csv')**

**print(data)**

```
data=pd.read_csv('LoanData.csv')
print(data)
```

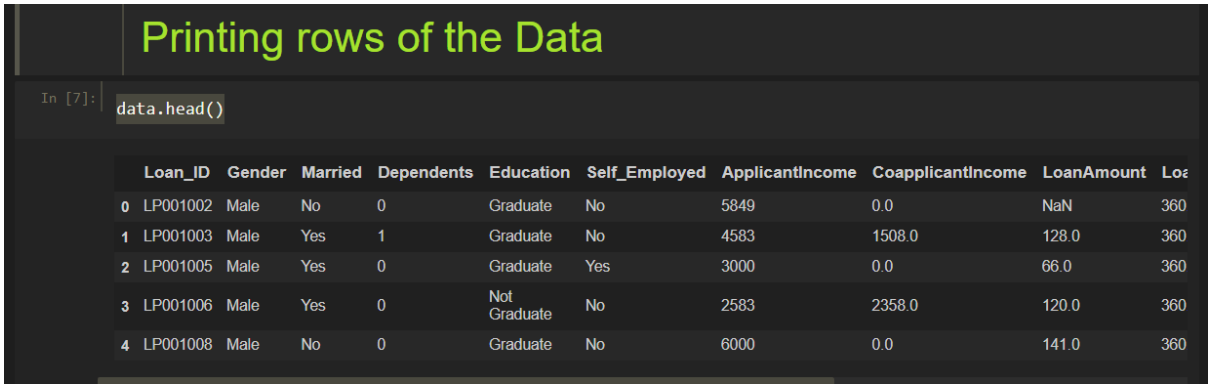
```
      Loan_ID  Gender Married Dependents     Education Self_Employed  \
0    LP001002    Male      No          0      Graduate            No
1    LP001003    Male     Yes          1      Graduate            No
2    LP001005    Male     Yes          0      Graduate           Yes
3    LP001006    Male     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
```

---

● **Printing rows of the Data**

Displays the first 5 rows of the dataset to get an overview of the data.

**data.head()**



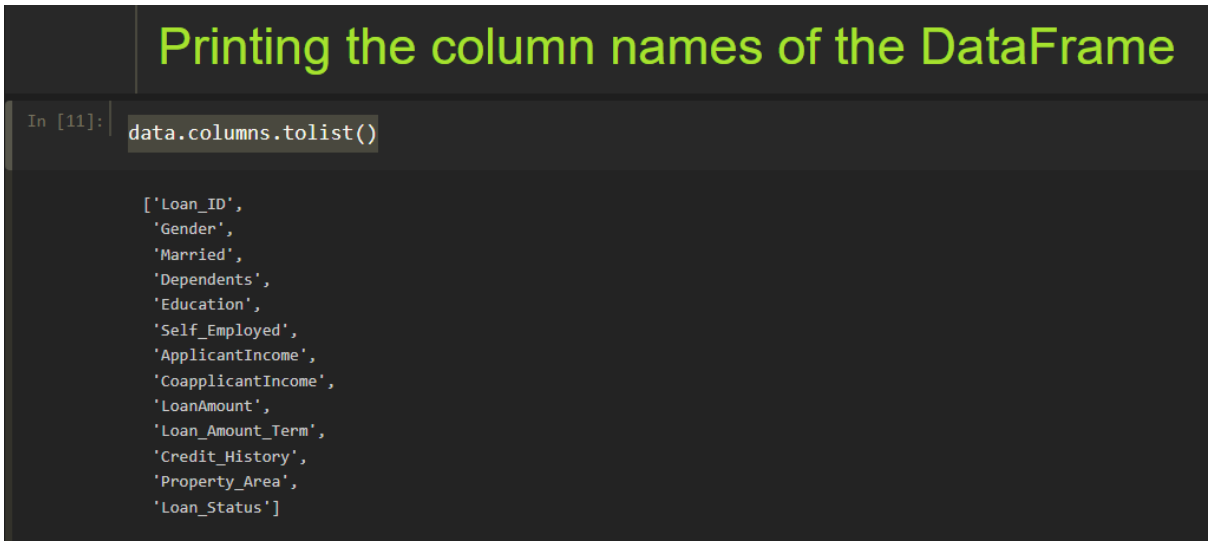

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAmount | Loa |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | LP001002 | Male | No | 0 | Graduate | No | 5849 | 0.0 | NaN | 360 |
| 1 | LP001003 | Male | Yes | 1 | Graduate | No | 4583 | 1508.0 | 128.0 | 360 |
| 2 | LP001005 | Male | Yes | 0 | Graduate | Yes | 3000 | 0.0 | 66.0 | 360 |
| 3 | LP001006 | Male | Yes | 0 | Not Graduate | No | 2583 | 2358.0 | 120.0 | 360 |
| 4 | LP001008 | Male | No | 0 | Graduate | No | 6000 | 0.0 | 141.0 | 360 |

---

● **Printing the column names of the DataFrame**

Lists all column names in the DataFrame.

**data.columns.tolist()**




---

● **Summary of Data Frame**

Provides a concise summary including data types and missing values.

**data.info()**

## Summary of Data Frame

```
In [12]: data.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
```

---

● **Descriptive Statistical Measures of a DataFrame**

Displays statistical summary such as mean, std, min, and max for numerical columns.

**data.describe()**

## Descriptive Statistical Measures of a DataFrame

```
In [13]: data.describe()
```

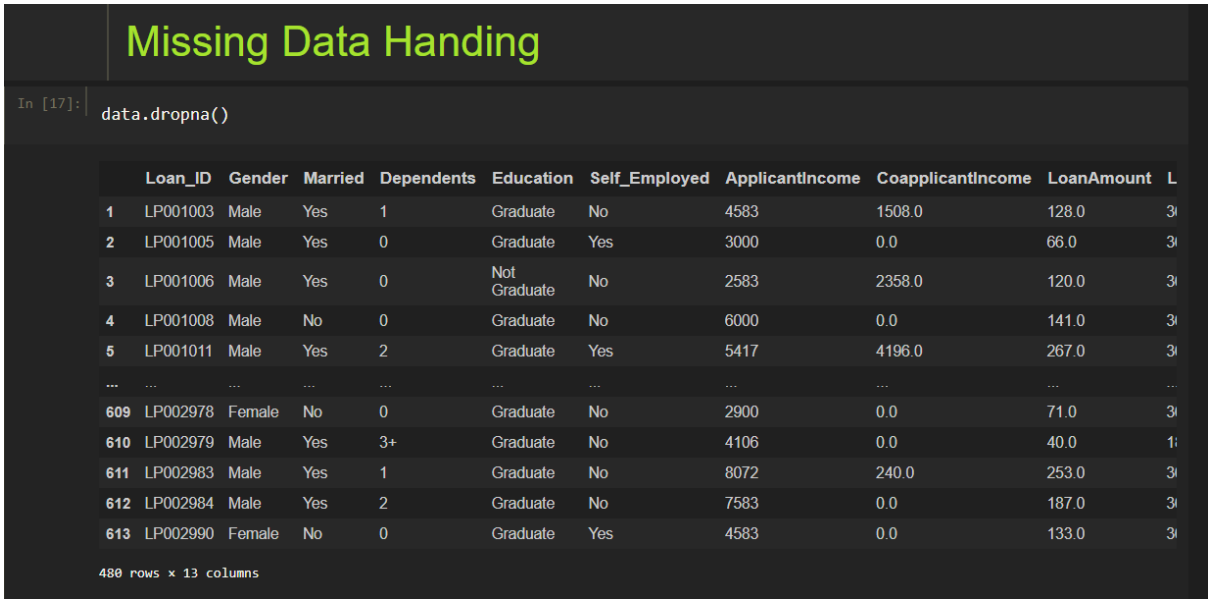
| | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_History |
|---|---|---|---|---|---|
| count | 614.000000 | 614.000000 | 592.000000 | 600.00000 | 564.000000 |
| mean | 5403.459283 | 1621.245798 | 146.412162 | 342.00000 | 0.842199 |
| std | 6109.041673 | 2926.248369 | 85.587325 | 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 |

---

● **Missing Data Handing**

Removes rows with any missing values. (Note: This doesn't change the original data unless assigned or inplace=True is set.)

**data.dropna()**




| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAmount | L |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | LP001003 | Male | Yes | 1 | Graduate | No | 4583 | 1508.0 | 128.0 | 3( |
| 2 | LP001005 | Male | Yes | 0 | Graduate | Yes | 3000 | 0.0 | 66.0 | 3( |
| 3 | LP001006 | Male | Yes | 0 | Not Graduate | No | 2583 | 2358.0 | 120.0 | 3( |
| 4 | LP001008 | Male | No | 0 | Graduate | No | 6000 | 0.0 | 141.0 | 3( |
| 5 | LP001011 | Male | Yes | 2 | Graduate | Yes | 5417 | 4196.0 | 267.0 | 3( |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 609 | LP002978 | Female | No | 0 | Graduate | No | 2900 | 0.0 | 71.0 | 3( |
| 610 | LP002979 | Male | Yes | 3+ | Graduate | No | 4106 | 0.0 | 40.0 | 1( |
| 611 | LP002983 | Male | Yes | 1 | Graduate | No | 8072 | 240.0 | 253.0 | 3( |
| 612 | LP002984 | Male | Yes | 2 | Graduate | No | 7583 | 0.0 | 187.0 | 3( |
| 613 | LP002990 | Female | No | 0 | Graduate | Yes | 4583 | 0.0 | 133.0 | 3( |

480 rows × 13 columns

● **Sorting DataFrame values**

Sorts the DataFrame in descending order based on the LoanAmount column.

**data.sort_values(by='LoanAmount', ascending=False, inplace=True)**

**print(data.head())**

## Sorting DataFrame values

```
data.sort_values(by='LoanAmount', ascending=False, inplace=True)
print(data.head())
```

```
      Loan_ID  Gender Married Dependents Education Self_Employed  \
171  LP001585     NaN     Yes         3+  Graduate            No
130  LP001469    Male      No          0  Graduate           Yes
155  LP001536    Male     Yes         3+  Graduate            No
561  LP002813  Female     Yes          1  Graduate           Yes
369  LP002191    Male     Yes          0  Graduate            No

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
171            51763                0.0       700.0             300.0
130            20166                0.0       650.0             480.0
155            39999                0.0       600.0             180.0
561            19484                0.0       600.0             360.0
369            19730             5266.0       570.0             360.0

     Credit_History Property_Area Loan_Status
171             1.0         Urban           Y
130             NaN         Urban           Y
155             0.0     Semiurban           Y
561             1.0     Semiurban           Y
369             1.0         Rural           N
```

● **Merge Data Frames**

Removes missing data and merges the cleaned dataset (data2) with the original on Loan_ID.

**data2=data.dropna()**

**pd.merge(data, data2, on='Loan_ID', how='inner')**

## Merge Data Frames

In [6]:
```
data2=data.dropna()
pd.merge(data, data2, on='Loan_ID', how='inner')
```

| | Loan_ID | Gender_x | Married_x | Dependents_x | Education_x | Self_Employed_x | ApplicantIncome_x | CoapplicantIncome_x |
|---|---|---|---|---|---|---|---|---|
| 0 | LP001536 | Male | Yes | 3+ | Graduate | No | 39999 | 0.0 |
| 1 | LP002813 | Female | Yes | 1 | Graduate | Yes | 19484 | 0.0 |
| 2 | LP002191 | Male | Yes | 0 | Graduate | No | 19730 | 5266.0 |
| 3 | LP002547 | Male | Yes | 1 | Graduate | No | 18333 | 0.0 |
| 4 | LP002959 | Female | Yes | 1 | Graduate | No | 12000 | 0.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 475 | LP002792 | Male | Yes | 1 | Graduate | No | 5468 | 1032.0 |
| 476 | LP001482 | Male | Yes | 0 | Graduate | Yes | 3459 | 0.0 |
| 477 | LP001325 | Male | No | 0 | Not Graduate | No | 3620 | 0.0 |
| 478 | LP001030 | Male | Yes | 2 | Graduate | No | 1299 | 1086.0 |
| 479 | LP002840 | Female | No | 0 | Graduate | No | 2378 | 0.0 |

480 rows x 25 columns

---

● **Apply Function**

Creates a new column by dividing loan amounts by 10.

**data['outputcolumn'] = data['LoanAmount'].apply(lambda x:x/10)**

**data.head()**

## Apply Function

In [9]:
```
data['outputcolumn'] = data['LoanAmount'].apply(lambda x:x/10)
data.head()
```

| plicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_History | Property_Area | Loan_Status | outputcolumn |
|---|---|---|---|---|---|---|---|
| 63 | 0.0 | 700.0 | 300.0 | 1.0 | Urban | Y | 70.0 |
| 66 | 0.0 | 650.0 | 480.0 | NaN | Urban | Y | 65.0 |
| 99 | 0.0 | 600.0 | 180.0 | 0.0 | Semiurban | Y | 60.0 |
| 84 | 0.0 | 600.0 | 360.0 | 1.0 | Semiurban | Y | 60.0 |
| 30 | 5266.0 | 570.0 | 360.0 | 1.0 | Rural | N | 57.0 |

---

● **By Using The Lambda Operator**

Adds another new column for transformed applicant income.

**data['appcolumn'] = data['ApplicantIncome'].apply(lambda x:x/10)**

**data.head()**

```
In [10]: data['appcolumn'] = data['ApplicantIncome'].apply(lambda x:x/10)
         data.head()
```

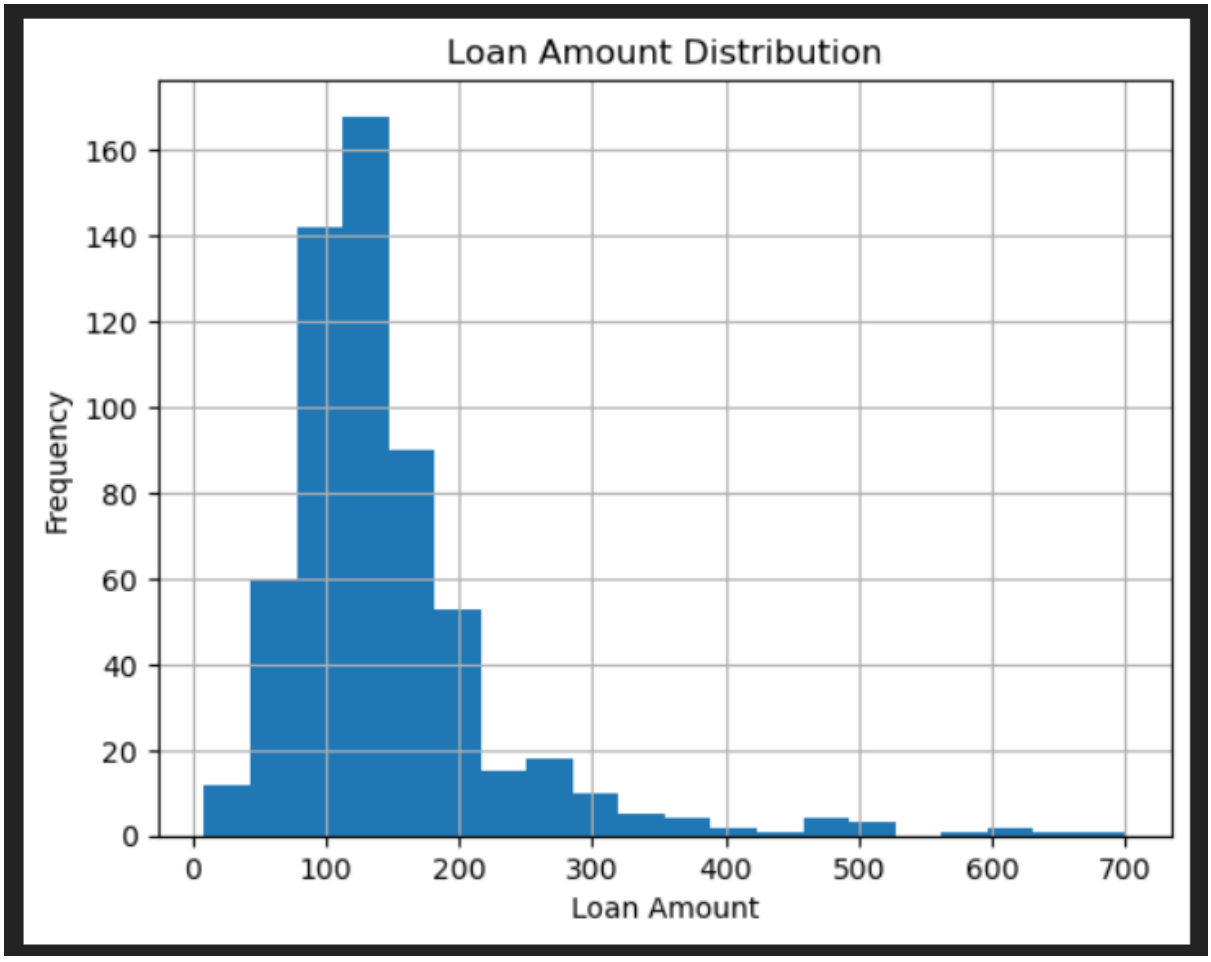
| CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_History | Property_Area | Loan_Status | outputcolumn | appcolumn |
|---|---|---|---|---|---|---|---|
| 0.0 | 700.0 | 300.0 | 1.0 | Urban | Y | 70.0 | 5176.3 |
| 0.0 | 650.0 | 480.0 | NaN | Urban | Y | 65.0 | 2016.6 |
| 0.0 | 600.0 | 180.0 | 0.0 | Semiurban | Y | 60.0 | 3999.9 |
| 0.0 | 600.0 | 360.0 | 1.0 | Semiurban | Y | 60.0 | 1948.4 |
| 5266.0 | 570.0 | 360.0 | 1.0 | Rural | N | 57.0 | 1973.0 |

● **Visualizing DataFrame**

A histogram to show the distribution of loan amounts.

```
import matplotlib.pyplot as plt


data['LoanAmount'].hist(bins=20)
plt.title('Loan Amount Distribution')
plt.xlabel('Loan Amount')
plt.ylabel('Frequency')
plt.show()
```

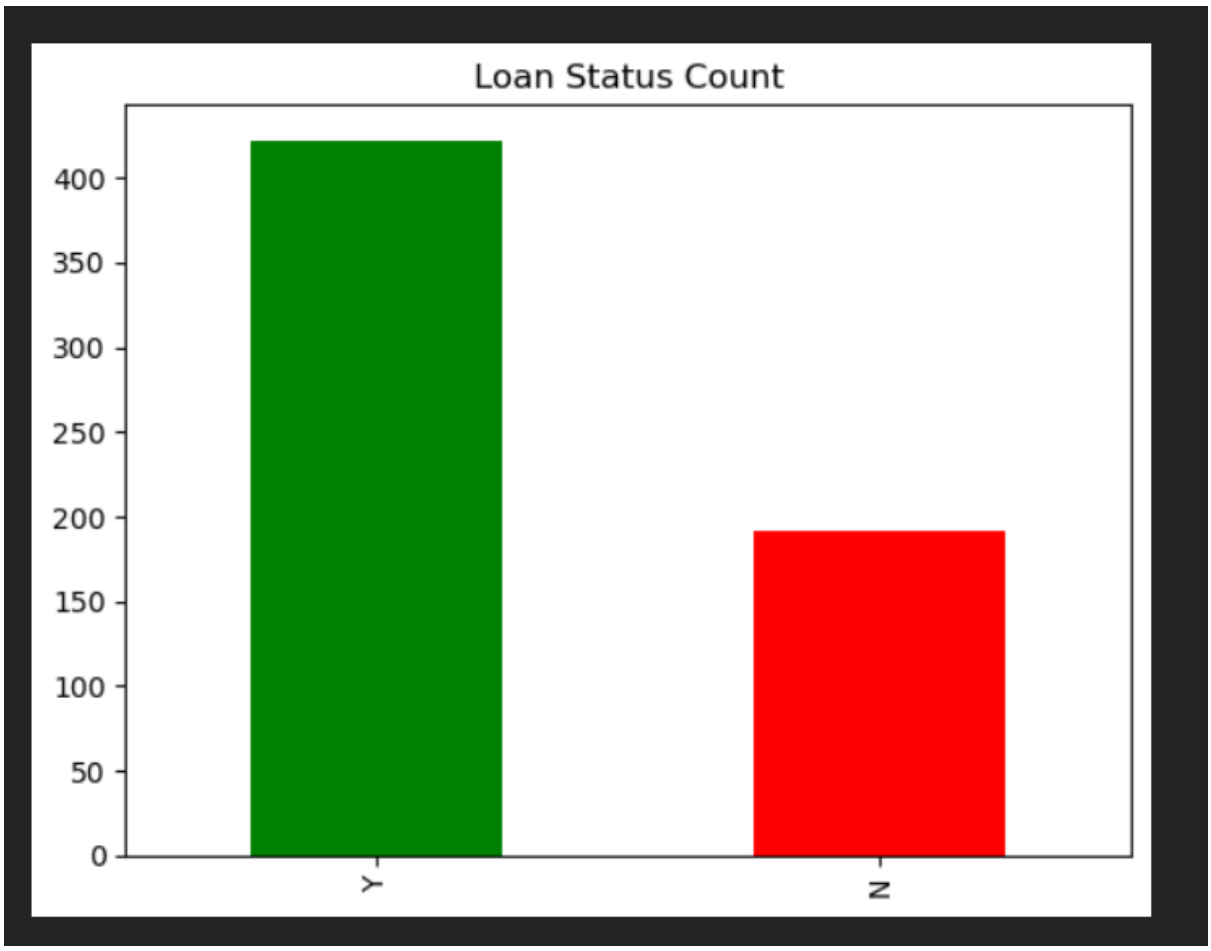


A bar chart to display the count of approved vs. rejected loans.

**data['Loan_Status'].value_counts().plot(kind='bar', color=['green', 'red'])**

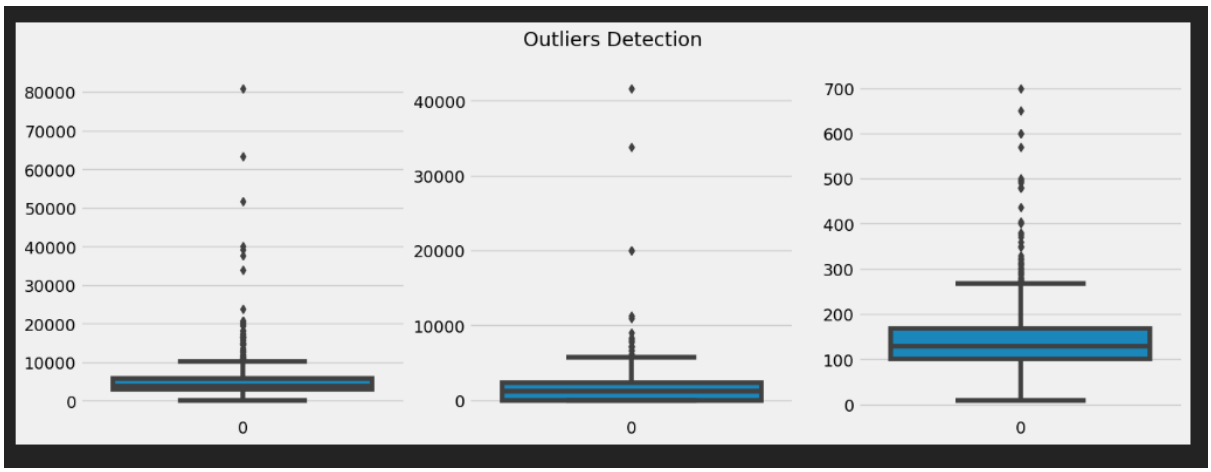
**plt.title('Loan Status Count')**

**plt.show()**



Side-by-side boxplots help detect outliers in applicant income, coapplicant income, and loan amount.

```
import seaborn as sns
import warnings
plt.style.use('fivethirtyeight')
plt.rcParams['figure.figsize'] = (15,5)
plt.subplot(1,3,1)
sns.boxplot(data['ApplicantIncome'])
plt.subplot(1,3,2)
sns.boxplot(data['CoapplicantIncome'])
plt.subplot(1,3,3)
sns.boxplot(data['LoanAmount'])
plt.suptitle('Outliers Detection')
plt.show()
```



Scatter plot visualizing the relationship between applicant income and loan amount.

**data.plot.scatter(x='ApplicantIncome', y='LoanAmount')**

**plt.title('Income vs Loan Amount')**

**plt.show()**