**Data Processing with Pandas Casestudy**

**Name: PRIYESHWAR**

**Mail:** [**priyesh2664@gmail.com**](mailto: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

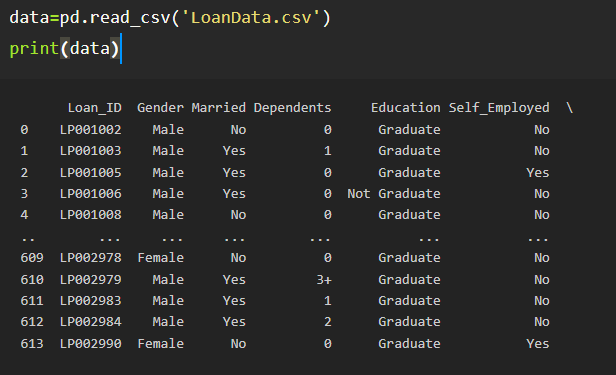
approved.

**● Loading Data in Pandas DataFrame**

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

**data=pd.read\_csv('LoanData.csv')**

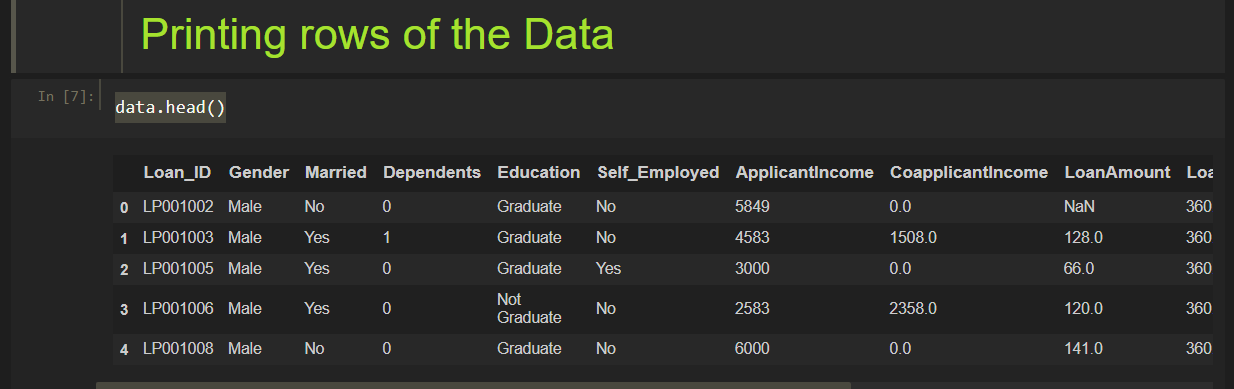
**print(data)**



**● Printing rows of the Data**

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

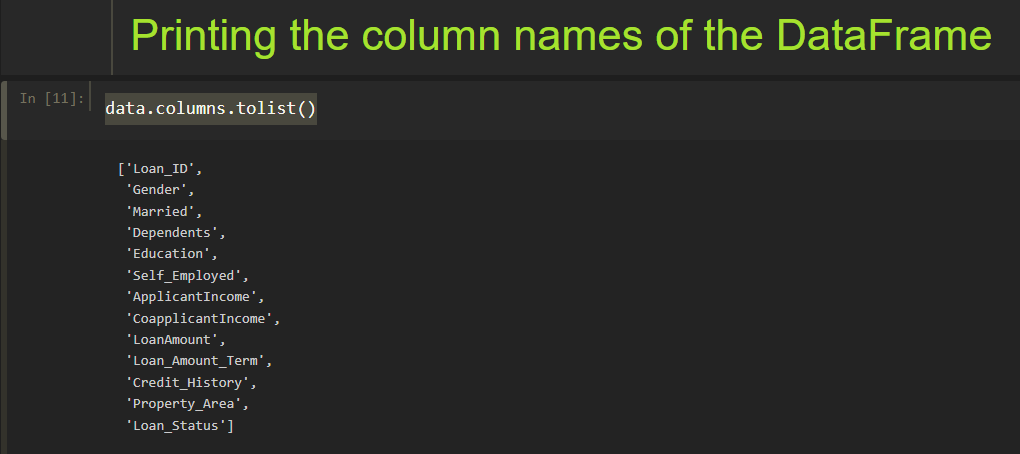
**data.head()**



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

Lists all column names in the DataFrame.

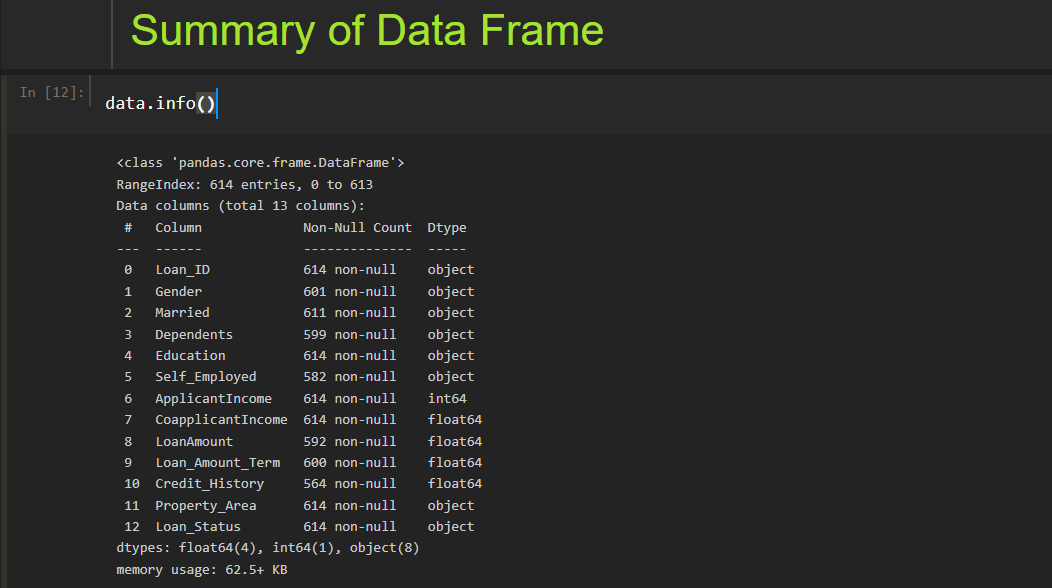
**data.columns.tolist()**



**● Summary of Data Frame**

Provides a concise summary including data types and missing values.

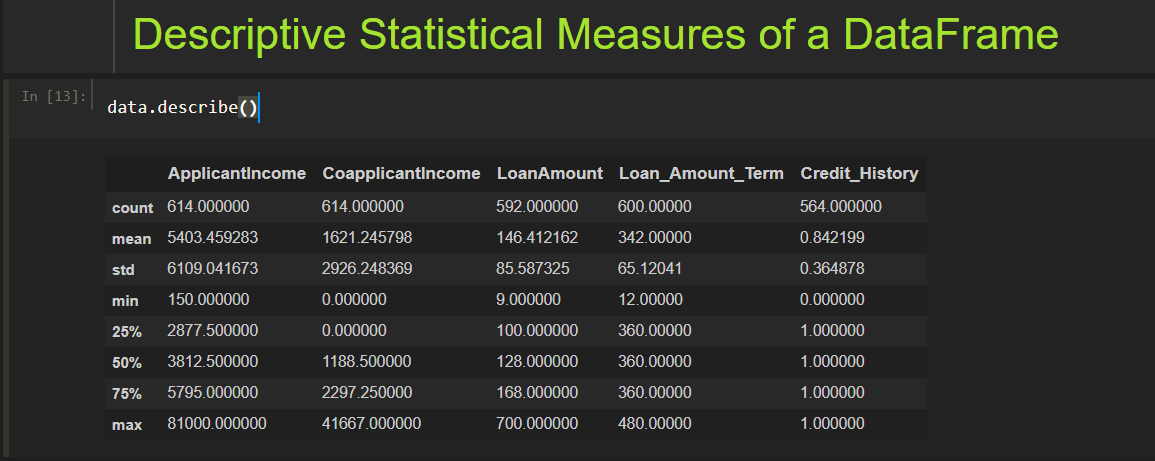
**data.info()**



**● Descriptive Statistical Measures of a DataFrame**

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

**data.describe()**



**● 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()**

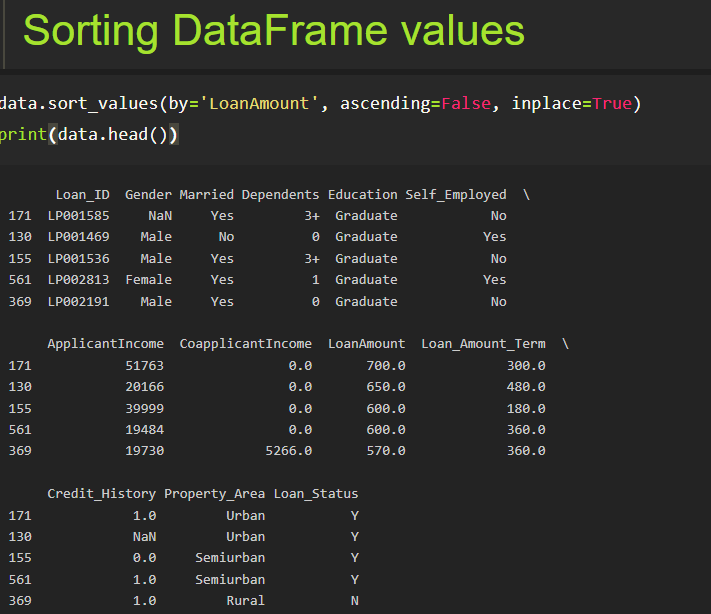


**● Sorting DataFrame values**

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

**data.sort\_values(by='LoanAmount', ascending=False, inplace=True)**

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

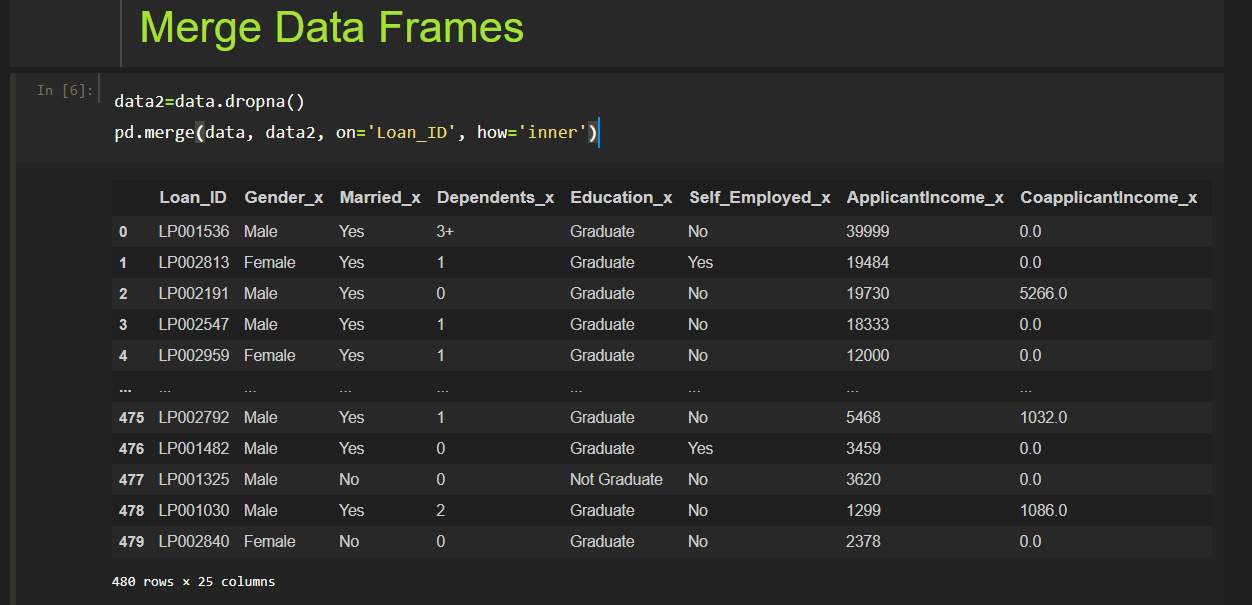


**● Merge Data Frames**

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')**

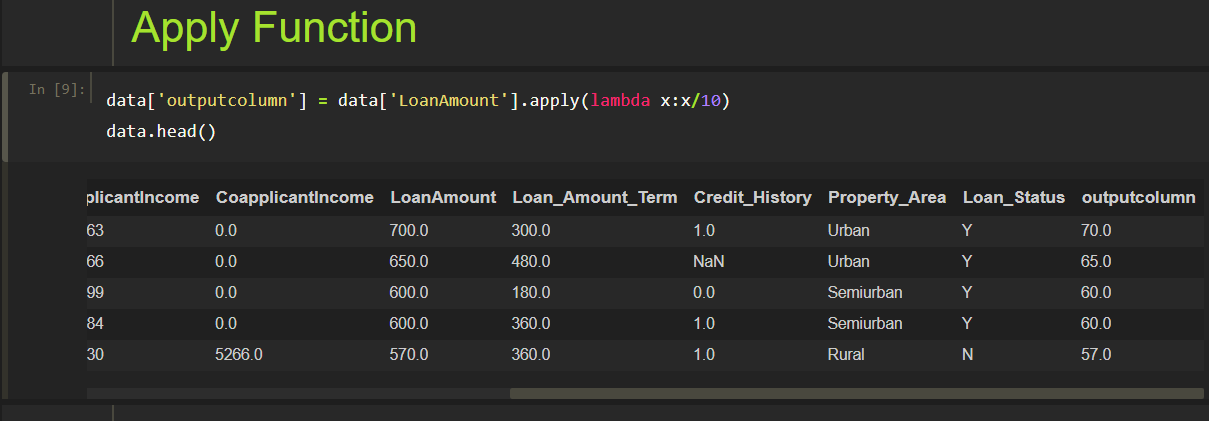


**● Apply Function**

Creates a new column by dividing loan amounts by 10.

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

**data.head()**

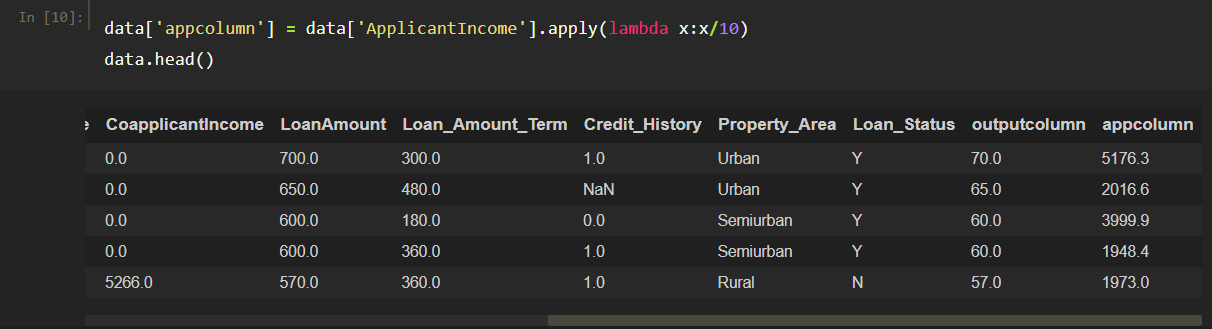


**● By Using The Lambda Operator**

Adds another new column for transformed applicant income.

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

**data.head()**

****

**● Visualizing DataFrame**

A histogram to show the distribution of loan amounts.

**import matplotlib.pyplot as plt**

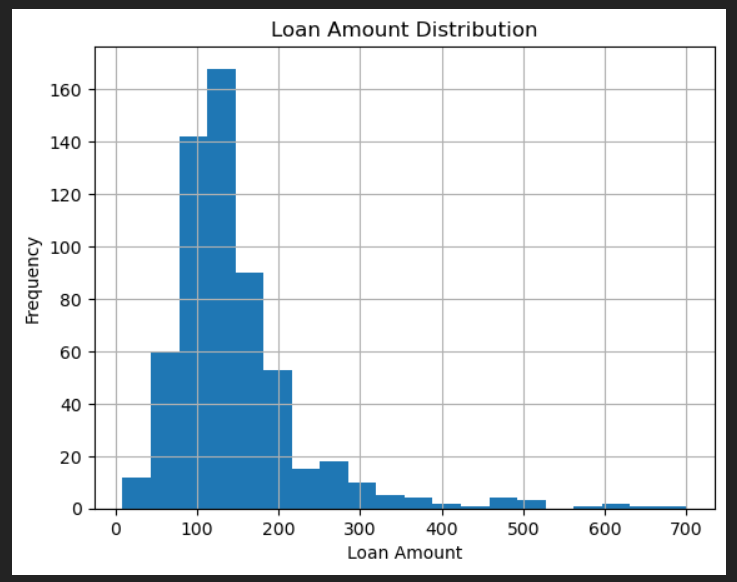
**data['LoanAmount'].hist(bins=20)**

**plt.title('Loan Amount Distribution')**

**plt.xlabel('Loan Amount')**

**plt.ylabel('Frequency')**

**plt.show()**

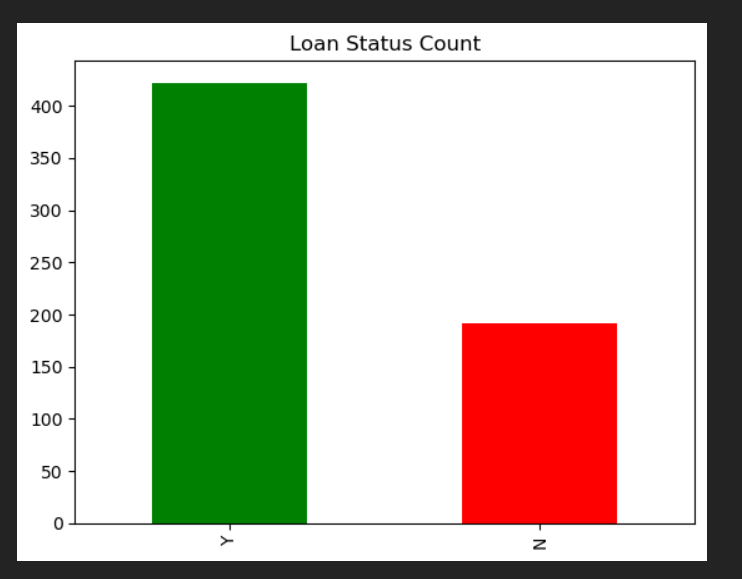


A bar chart to display the count of approved vs. rejected loans.

**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)**

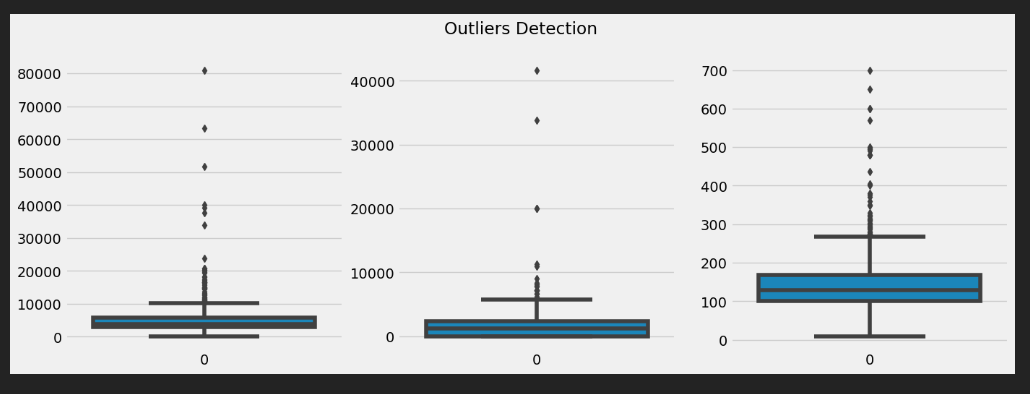
**sns.boxplot(data['CoapplicantIncome'])**

**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()**

