

# PYTHON CASE STUDY

**Automate the loan eligibility process based on user details provided in an online form.**

## 1.Loading Data in Pandas Data Frame

```
import pandas as pd
```

```
df = pd.read_csv("/content/LoanData (1) (1).csv")
```

```
df.head()
```

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

## 2. Printing the Column Names

```
df.columns
```

```
df.columns
```

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

## 3.Summary of the DataFrame

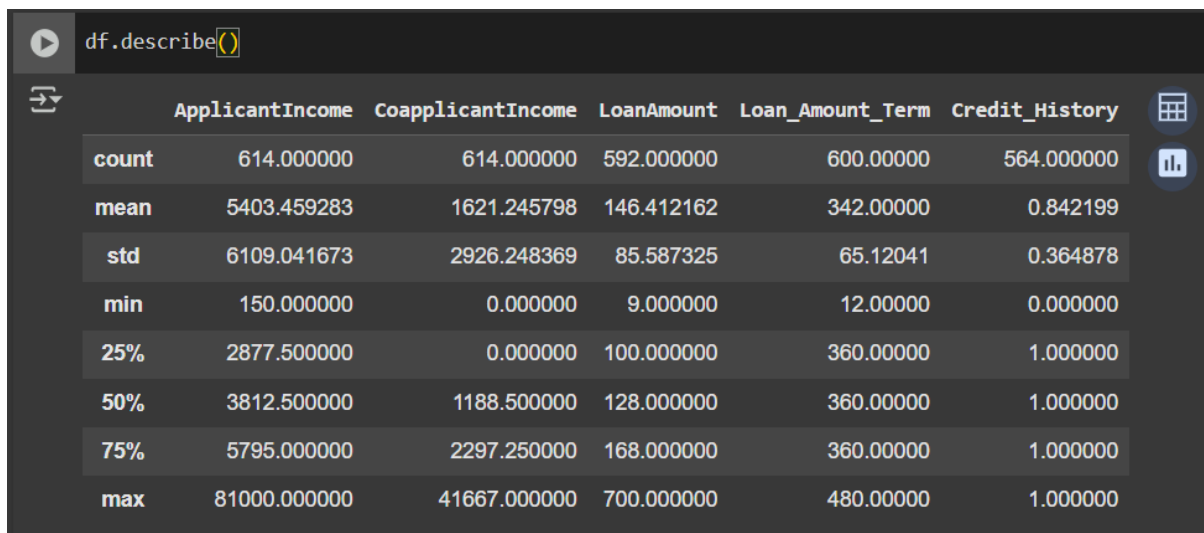
```
df.info()
```

```
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
```

## 4.Descriptive Statistical Measures

df.describe()



A Jupyter Notebook interface showing the execution of `df.describe()`. The code cell contains `df.describe()`. The output is a summary statistics table for five variables: ApplicantIncome, CoapplicantIncome, LoanAmount, Loan\_Amount\_Term, and Credit\_History. The table includes rows for count, mean, std, min, 25%, 50%, 75%, and max.

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.000000	564.000000
mean	5403.459283	1621.245798	146.412162	342.000000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.000000	0.000000
25%	2877.500000	0.000000	100.000000	360.000000	1.000000
50%	3812.500000	1188.500000	128.000000	360.000000	1.000000
75%	5795.000000	2297.250000	168.000000	360.000000	1.000000
max	81000.000000	41667.000000	700.000000	480.000000	1.000000

## 5.Missing Data Handling

- Check Missing Values

df.isnull().sum()



A Jupyter Notebook interface showing the execution of `df.isnull().sum()`. The code cell contains `df.isnull().sum()`. The output is a series of counts for missing values in various columns. The dtype is int64.

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

- Fill Missing

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

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

```
df.head()
```

```
df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
df['LoanAmount'].fillna(df['LoanAmount'].mean(), inplace=True)
df.head()
```

/tmp/ipython-input-8-218709055.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using the inplace method. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always exists. For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or 'df[col] = df[col].method(value, inplace=True)' instead.

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

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001002	Male	No	0	Graduate	No	5849	0.0	146.412162	360.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.000000	360.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.000000	360.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.000000	360.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.000000	360.0	

## 6.Sorting DataFrame Values

`df.sort_values(by='ApplicantIncome', ascending=False)`

```
df.sort_values(by='ApplicantIncome', ascending=False)
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
409	LP002317	Male	Yes	3+	Graduate	No	81000	0.0	360.0	360.0	
333	LP002101	Male	Yes	0	Graduate	NaN	63337	0.0	490.0	180.0	
171	LP001585	Male	Yes	3+	Graduate	No	51763	0.0	700.0	300.0	
155	LP001536	Male	Yes	3+	Graduate	No	39999	0.0	600.0	180.0	
185	LP001640	Male	Yes	0	Graduate	Yes	39147	4750.0	120.0	360.0	
...	...	...	...	...	...	...	...	...	...	...	...
188	LP001644	Male	Yes	0	Graduate	Yes	674	5296.0	168.0	360.0	
500	LP002603	Female	No	0	Graduate	No	645	3683.0	113.0	480.0	
600	LP002949	Female	No	3+	Graduate	NaN	416	41667.0	350.0	180.0	
468	LP002502	Female	Yes	2	Not Graduate	NaN	210	2917.0	98.0	360.0	
216	LP001722	Male	Yes	0	Graduate	No	150	1800.0	135.0	360.0	

614 rows x 12 columns

## 7.Merge DataFrame

```
df1 = pd.read_csv("/content/LoanData (1) (1).csv")
```

```
df2 = pd.read_csv("/content/LoanData (1) (1).csv")
```

```
df = pd.merge(df1, df2)
```

```
print(df)
```

	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

## 8. Apply Function

```
df['LoanAmount_k'] = df['LoanAmount'].apply(lambda x: x / 1000)
```

```
df.head()
```

```
df['LoanAmount_k'] = df['LoanAmount'].apply(lambda x: x / 1000)
df.head()
```

	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status	LoanAmount_k
0	No	5849	0.0	NaN	360.0	1.0	Urban	Y	NaN
1	No	4583	1508.0	128.0	360.0	1.0	Rural	N	0.128
2	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y	0.066
3	No	2583	2358.0	120.0	360.0	1.0	Urban	Y	0.120
4	No	6000	0.0	141.0	360.0	1.0	Urban	Y	0.141

## 9. Using Lambda Operator

```
df['IncomeLevel'] = df['ApplicantIncome'].apply(lambda x: 'High' if x > 5000 else 'Low')
```

```
df.head()
```

```
df['IncomeLevel'] = df['ApplicantIncome'].apply(lambda x: 'High' if x > 5000 else 'Low')
df.head()
```

	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status	LoanAmount_k	IncomeLevel
0	No	5849	0.0	NaN	360.0	1.0	Urban	Y	NaN	High
1	No	4583	1508.0	128.0	360.0	1.0	Rural	N	0.128	Low
2	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y	0.066	Low
3	No	2583	2358.0	120.0	360.0	1.0	Urban	Y	0.120	Low
4	No	6000	0.0	141.0	360.0	1.0	Urban	Y	0.141	High

## 10. Visualizing DataFrame

```
import matplotlib.pyplot as plt
```

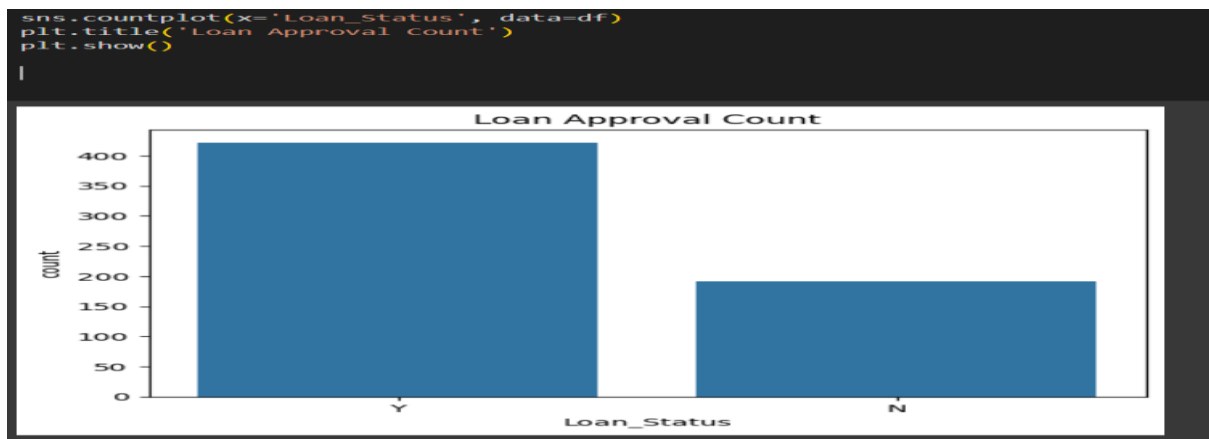
```
import seaborn as sns
```

```
#barchart
```

```
sns.countplot(x='Loan_Status', data=df)
```

```
plt.title('Loan Approval Count')
```

```
plt.show()
```



```
#histogram
```

```
df['ApplicantIncome'].hist(bins=30)
```

```
plt.title("Applicant Income Distribution")
```

```
plt.show()
```

