

IV. EDA-Data Inspection and Analysis

AIM:

- Viewing and inspecting DataFrames
- Filtering and subsetting data using conditions
- Descriptive statistics: measures of central tendency (mean, median, mode) and measures of dispersion (range, variance, standard deviation)

PROGRAM:

```
import pandas as pd

df = pd.read_csv('/content/test_Y3wMUE5_7gLdaTN.csv')

print(df.head())

print(df.tail())

print(df.info())

print(df.describe())

print(df.columns)
```

```
366  LP002989  Male  No  0  Graduate  Yes
ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
362  4009  1777  113.0  360.0
363  4158  709  115.0  360.0
364  3250  1993  126.0  360.0
365  5000  2393  158.0  360.0
366  9200  0  98.0  180.0

Credit_History  Property_Area
362  1.0  Urban
363  1.0  Urban
364  NaN  Semiurban
365  1.0  Rural
366  1.0  Rural
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Loan_ID               367 non-null   object
1   Gender                356 non-null   object
2   Married               367 non-null   object
3   Dependents            357 non-null   object
4   Education             367 non-null   object
5   Self_Employed         344 non-null   object
6   ApplicantIncome       367 non-null   int64
7   CoapplicantIncome     367 non-null   int64
8   LoanAmount            362 non-null   float64
9   Loan_Amount_Term      361 non-null   float64
10  Credit_History         338 non-null   float64
11  Property_Area          367 non-null   object
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
None
```

```

ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
count      367.000000      367.000000      362.000000      361.000000
mean      4805.599455      1569.577657      136.132597      342.537396
std      4910.685399      2334.232099      61.366652      65.156643
min         0.000000         0.000000       28.000000        6.000000
25%      2864.000000         0.000000      100.250000      360.000000
50%      3786.000000      1025.000000      125.000000      360.000000
75%      5060.000000      2430.500000      158.000000      360.000000
max      72529.000000     24000.000000     550.000000     480.000000

Credit_History
count      338.000000
mean         0.825444
std         0.380150
min         0.000000
25%         1.000000
50%         1.000000
75%         1.000000
max         1.000000
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
      dtype='object')

```

```

categorical_cols = ['Gender', 'Married', 'Dependents', 'Education',
'Self_Employed', 'Property_Area']

```

```

for col in categorical_cols:

```

```

    unique_vals = df[col].unique()

```

```

    print(f"\nUnique values in '{col}': {unique_vals}")

```

```

Unique values in 'Gender': ['Male' 'Female' nan]

Unique values in 'Married': ['Yes' 'No']

Unique values in 'Dependents': ['0' '1' '2' '3+' nan]

Unique values in 'Education': ['Graduate' 'Not Graduate']

Unique values in 'Self_Employed': ['No' 'Yes' nan]

Unique values in 'Property_Area': ['Urban' 'Semiurban' 'Rural']

```

```

df['Dependents'] = df['Dependents'].replace('3+', 3)

```

```

df['Dependents'] = pd.to_numeric(df['Dependents'],
errors='coerce').astype('Int64')

```

```

print("\nValue counts in 'Dependents' after conversion:")

```

```
print(df['Dependents'].value_counts(dropna=False))
```

```
Value counts in 'Dependents' after conversion:
Dependents
0      200
2       59
1       58
3       40
<NA>   10
Name: count, dtype: Int64
```

```
cols = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount']
```

```
for col in cols:
```

```
    print(f"\nStatistics for '{col}':")
```

```
    print(f"Mean: {df[col].mean():.2f}")
```

```
    print(f"Median: {df[col].median():.2f}")
```

```
    print(f"Mode: {df[col].mode().values}")
```

```
    print(f"Range: {df[col].max() - df[col].min():.2f}")
```

```
    print(f"Variance: {df[col].var():.2f}")
```

```
    print(f"Standard Deviation: {df[col].std():.2f}")
```

```
Statistics for 'ApplicantIncome':
Mean: 4885.60
Median: 3786.00
Mode: [3500 5000]
Range: 72529.00
Variance: 24114831.09
Standard Deviation: 4910.69

Statistics for 'CoapplicantIncome':
Mean: 1569.58
Median: 1025.00
Mode: [0]
Range: 24000.00
Variance: 5448639.49
Standard Deviation: 2334.23

Statistics for 'LoanAmount':
Mean: 136.13
Median: 125.00
Mode: [150.]
Range: 522.00
Variance: 3765.87
Standard Deviation: 61.37
```

RESULT:

Thus, the given program was written and executed successfully.