**EXP - 1**

**PROGRAM:**

# Create a new notebook for Python

# Write and execute Python code

print("Hello!")

# Demonstrate the application of Jupyter Widgets, Jupyter AI

import ipywidgets as widgets

from IPython.display import display

def on\_slider\_change(change):

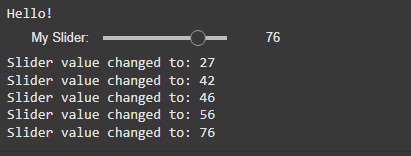
print(f"Slider value changed to: {change['new']}")

slider = widgets.IntSlider(value=25, min=0, max=100, description='My Slider:')

slider.observe(on\_slider\_change, names='value')

display(slider)

**OUTPUT:-**

****

**EXP- 2**

**Dataset used : college\_student\_placement\_dataset.csv**

**PROGRAM:**

import pandas as pd

import requests

from google.colab import files

# 1. Upload CSV File

uploaded = files.upload()

df\_csv = pd.read\_csv(next(iter(uploaded)))

print("CSV loaded successfully")

print(df\_csv.head())

print(df\_csv.shape)

print(df\_csv.columns)

print(df\_csv.describe())

print(df\_csv.isnull().sum())

# 2. Create a new DataFrame from existing CSV DataFrame

df\_excel = df\_csv.copy()

print("New DataFrame created from CSV")

print(df\_excel.head())

# 3. Export CSV DataFrame to Excel (saved in runtime, not downloaded)

df\_csv.to\_excel("college\_placement\_export.xlsx", index=False)

print("Excel file saved successfully")

# 4. Web Scraping from Wikipedia

url = "https://en.wikipedia.org/wiki/List\_of\_countries\_by\_GDP\_(nominal)"

response = requests.get(url)

tables = pd.read\_html(response.text)

df\_web = tables[2]

# Clean column names safely

df\_web.columns = [str(col).strip() for col in df\_web.columns]

print("Scraped table:")

print(df\_web.head())

# 5. Export scraped table to Excel (runtime only)

df\_web.to\_excel("scraped\_gdp\_table.xlsx", index=False)

print("Scraped table saved to Excel")

**OUTPUT:-**

Saving college\_student\_placement\_dataset.csv to college\_student\_placement\_dataset (1).csv

CSV loaded successfully

College\_ID IQ Prev\_Sem\_Result CGPA Academic\_Performance \

0 CLG0030 107 6.61 6.28 8

1 CLG0061 97 5.52 5.37 8

2 CLG0036 109 5.36 5.83 9

3 CLG0055 122 5.47 5.75 6

4 CLG0004 96 7.91 7.69 7

Internship\_Experience Extra\_Curricular\_Score Communication\_Skills \

0 No 8 8

1 No 7 8

2 No 3 1

3 Yes 1 6

4 No 8 10

Projects\_Completed Placement

0 4 No

1 0 No

2 1 No

3 1 No

4 2 No

(10000, 10)

Index(['College\_ID', 'IQ', 'Prev\_Sem\_Result', 'CGPA', 'Academic\_Performance',

'Internship\_Experience', 'Extra\_Curricular\_Score',

'Communication\_Skills', 'Projects\_Completed', 'Placement'],

dtype='object')

IQ Prev\_Sem\_Result CGPA Academic\_Performance \

count 10000.000000 10000.000000 10000.000000 10000.000000

mean 99.471800 7.535673 7.532379 5.546400

std 15.053101 1.447519 1.470141 2.873477

min 41.000000 5.000000 4.540000 1.000000

25% 89.000000 6.290000 6.290000 3.000000

50% 99.000000 7.560000 7.550000 6.000000

75% 110.000000 8.790000 8.770000 8.000000

max 158.000000 10.000000 10.460000 10.000000

Extra\_Curricular\_Score Communication\_Skills Projects\_Completed

count 10000.000000 10000.000000 10000.000000

mean 4.970900 5.561800 2.513400

std 3.160103 2.900866 1.715959

min 0.000000 1.000000 0.000000

25% 2.000000 3.000000 1.000000

50% 5.000000 6.000000 3.000000

75% 8.000000 8.000000 4.000000

max 10.000000 10.000000 5.000000

College\_ID 0

IQ 0

Prev\_Sem\_Result 0

CGPA 0

Academic\_Performance 0

Internship\_Experience 0

Extra\_Curricular\_Score 0

Communication\_Skills 0

Projects\_Completed 0

Placement 0

dtype: int64

New DataFrame created from CSV

College\_ID IQ Prev\_Sem\_Result CGPA Academic\_Performance \

0 CLG0030 107 6.61 6.28 8

1 CLG0061 97 5.52 5.37 8

2 CLG0036 109 5.36 5.83 9

3 CLG0055 122 5.47 5.75 6

4 CLG0004 96 7.91 7.69 7

Internship\_Experience Extra\_Curricular\_Score Communication\_Skills \

0 No 8 8

1 No 7 8

2 No 3 1

3 Yes 1 6

4 No 8 10

Projects\_Completed Placement

0 4 No

1 0 No

2 1 No

3 1 No

4 2 No

Excel file saved successfully

Scraped table:

('Country/Territory', 'Country/Territory') ('IMF[1][12]', 'Forecast') \

0 World 113795678

1 United States 30507217

2 China 19231705

3 Germany 4744804

4 India 4187017

('IMF[1][12]', 'Year') ('World Bank[13]', 'Estimate') \

0 2025 111326370

1 2025 29184890

2 [n 1]2025 18743803

3 2025 4659929

4 2025 3912686

('World Bank[13]', 'Year') ('United Nations[14]', 'Estimate') \

0 2024 100834796

1 2024 27720700

2 [n 3]2024 17794782

3 2024 4525704

4 2024 3575778

('United Nations[14]', 'Year')

0 2022

1 2023

2 [n 1]2023

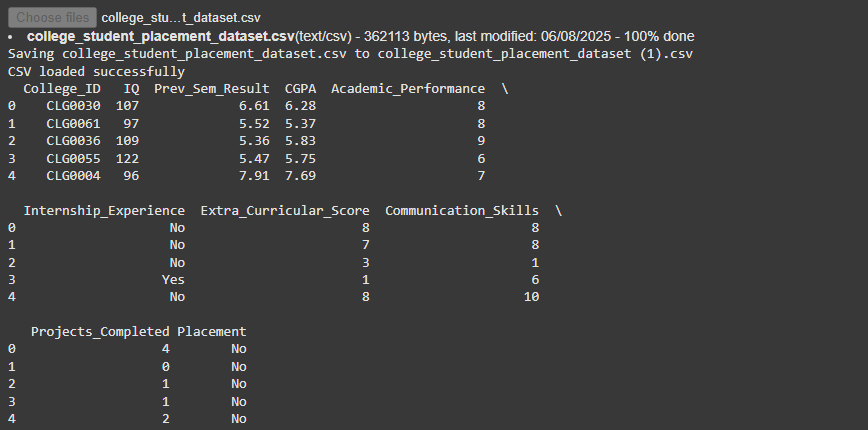
3 2023

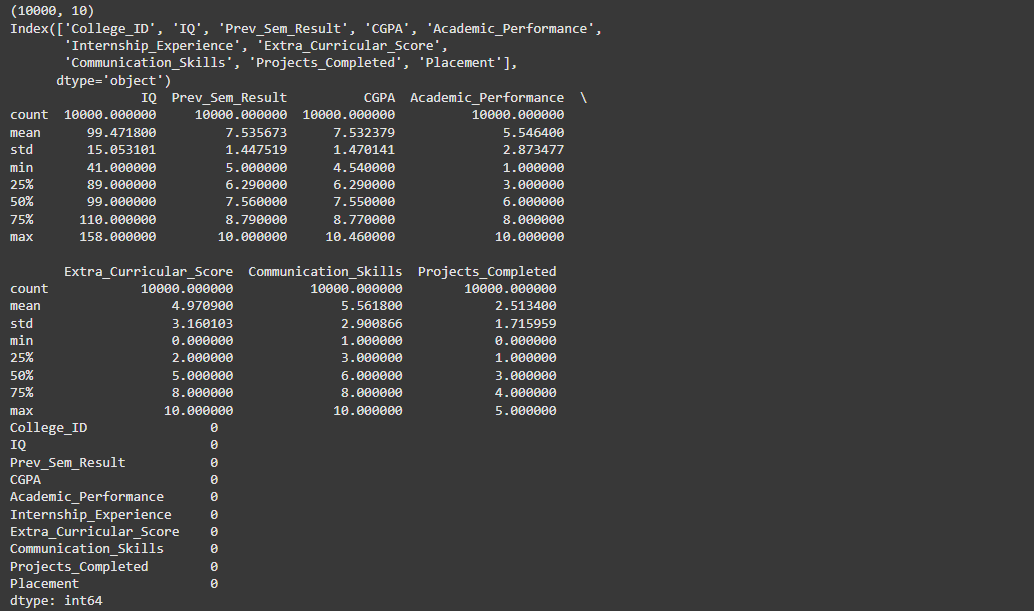
4 2023

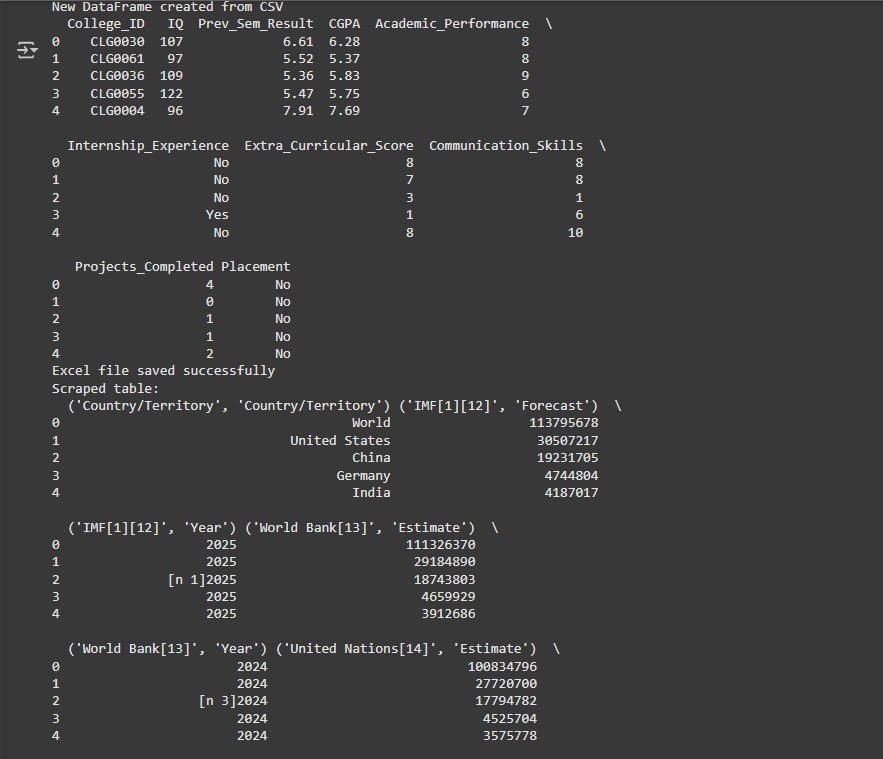
Scraped table saved to Excel

/tmp/ipython-input-1096754866.py:28: FutureWarning: Passing literal html to 'read\_html' is deprecated and will be removed in a future version. To read from a literal string, wrap it in a 'StringIO' object.

tables = pd.read\_html(response.text)

****

****

****

**EXP - 3**

**Dataset used : titanic-dataset.csv**

**PROGRAM**

# Step 1: Import libraries

import pandas as pd

import numpy as np

from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Step 2: Load the dataset

df = pd.read\_csv("/mnt/data/Titanic-Dataset.csv")

# Step 3: Preview dataset

print("🔍 Dataset Preview:")

print(df.iloc[:, :4].head())

# Step 4: Check missing values

print("\n Missing Values:")

print(df.isnull().sum())

# Step 5: Fill missing values

# Fill Age with median

if 'Age' in df.columns:

df['Age'].fillna(df['Age'].median(), inplace=True)

# Fill Embarked with mode

if 'Embarked' in df.columns:

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

# Fill Cabin with 'Unknown' or drop if sparse

if 'Cabin' in df.columns:

missing\_ratio = df['Cabin'].isnull().mean()

if missing\_ratio > 0.8:

df.drop('Cabin', axis=1, inplace=True)

else:

df['Cabin'].fillna('Unknown', inplace=True)

# Drop rows with any remaining missing values (if needed)

df.dropna(inplace=True)

# Step 6: Remove duplicates

duplicates = df.duplicated().sum()

print(f"\n Duplicates Found: {duplicates}")

df.drop\_duplicates(inplace=True)

# Step 7: Drop unnecessary columns (optional)

drop\_cols = ['Name', 'Ticket'] # Add more if needed

df.drop([col for col in drop\_cols if col in df.columns], axis=1, inplace=True)

# Step 8: Convert data types

if 'Survived' in df.columns:

df['Survived'] = df['Survived'].astype('category')

if 'Pclass' in df.columns:

df['Pclass'] = df['Pclass'].astype('category')

# Step 9: Ensure consistency in categorical values

if 'Sex' in df.columns:

df['Sex'] = df['Sex'].str.lower().str.strip()

if 'Embarked' in df.columns:

df['Embarked'] = df['Embarked'].str.upper().str.strip()

# Step 10: Normalize numeric columns

# Choose either Min-Max or Standardization

numeric\_cols = ['Age', 'Fare']

available\_numeric\_cols = [col for col in numeric\_cols if col in df.columns]

# Min-Max Scaling

scaler = MinMaxScaler()

df[available\_numeric\_cols] = scaler.fit\_transform(df[available\_numeric\_cols])

# Final info

print("\n Cleaned Data Info:")

print([df.info](http://df.info/)())

print("\n📊 Summary Statistics:")

print(df.describe(include='all'))

**OUTPUT:-**

Dataset Preview:

PassengerId Survived Pclass \

0 1 0 3

1 2 1 1

2 3 1 3

3 4 1 1

4 5 0 3

Name

0 Braund, Mr. Owen Harris

1 Cumings, Mrs. John Bradley (Florence Briggs Th...

2 Heikkinen, Miss. Laina

3 Futrelle, Mrs. Jacques Heath (Lily May Peel)

4 Allen, Mr. William Henry

Missing Values:

PassengerId 0

Survived 0

Pclass 0

Name 0

Sex 0

Age 177

SibSp 0

Parch 0

Ticket 0

Fare 0

Cabin 687

Embarked 2

dtype: int64

Duplicates Found: 0

Cleaned Data Info:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 891 entries, 0 to 890

Data columns (total 10 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 PassengerId 891 non-null int64

1 Survived 891 non-null category

2 Pclass 891 non-null category

3 Sex 891 non-null object

4 Age 891 non-null float64

5 SibSp 891 non-null int64

6 Parch 891 non-null int64

7 Fare 891 non-null float64

8 Cabin 891 non-null object

9 Embarked 891 non-null object

dtypes: category(2), float64(2), int64(3), object(3)

memory usage: 57.8+ KB

None

Summary Statistics:

PassengerId Survived Pclass Sex Age SibSp \

count 891.000000 891.0 891.0 891 891.000000 891.000000

unique NaN 2.0 3.0 2 NaN NaN

top NaN 0.0 3.0 male NaN NaN

freq NaN 549.0 491.0 577 NaN NaN

mean 446.000000 NaN NaN NaN 0.363679 0.523008

std 257.353842 NaN NaN NaN 0.163605 1.102743

min 1.000000 NaN NaN NaN 0.000000 0.000000

25% 223.500000 NaN NaN NaN 0.271174 0.000000

50% 446.000000 NaN NaN NaN 0.346569 0.000000

75% 668.500000 NaN NaN NaN 0.434531 1.000000

max 891.000000 NaN NaN NaN 1.000000 8.000000

Parch Fare Cabin Embarked

count 891.000000 891.000000 891 891

unique NaN NaN 148 3

top NaN NaN Unknown S

freq NaN NaN 687 646

mean 0.381594 0.062858 NaN NaN

std 0.806057 0.096995 NaN NaN

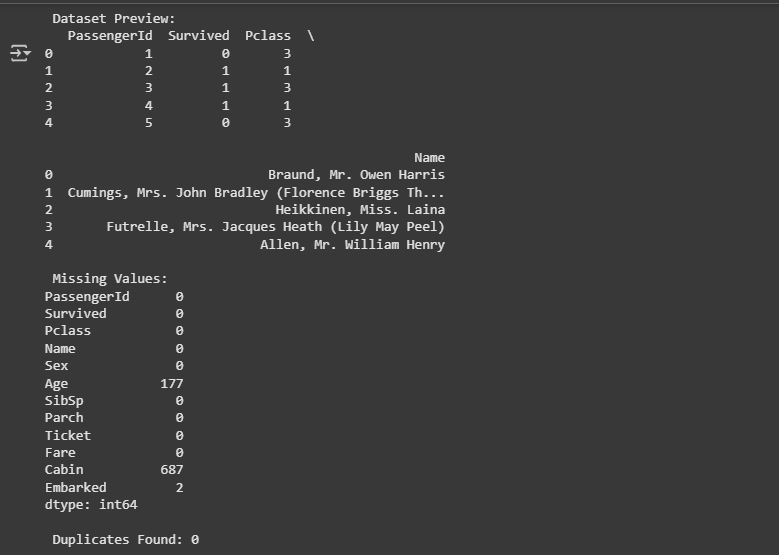
min 0.000000 0.000000 NaN NaN

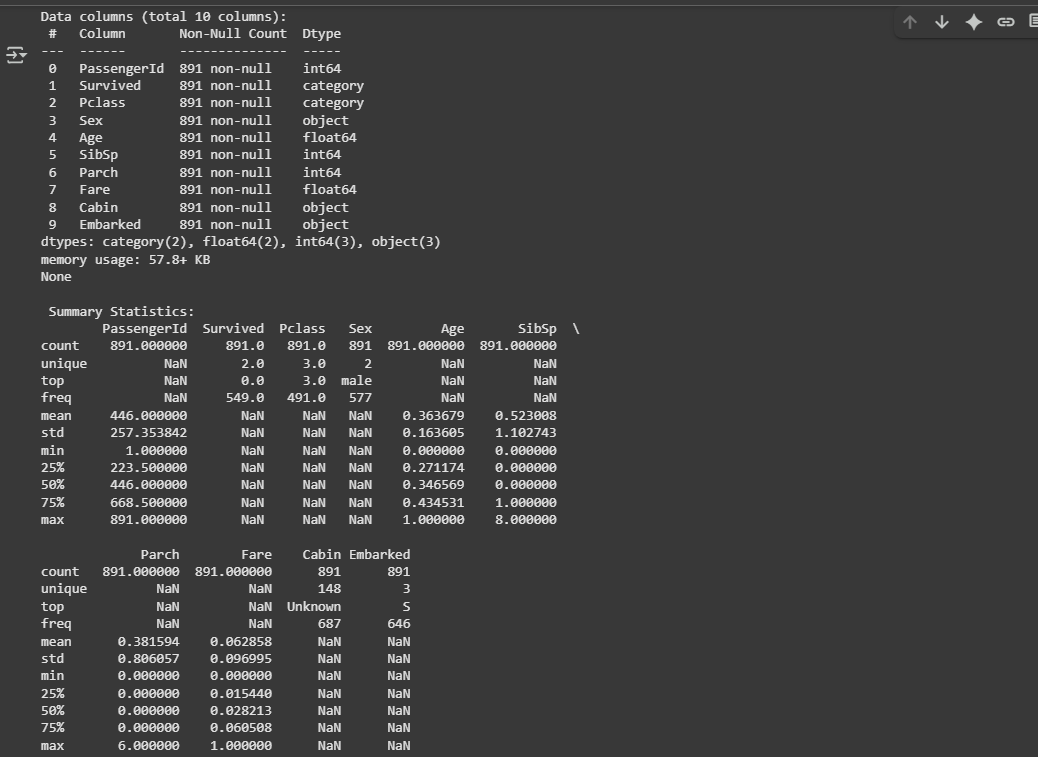
25% 0.000000 0.015440 NaN NaN

50% 0.000000 0.028213 NaN NaN

75% 0.000000 0.060508 NaN NaN

max 6.000000 1.000000 NaN NaN





**Exp - 4**

**Dataset used : Loan\_data.csv**

**PROGRAM:**

import pandas as pd

import numpy as np

# Load dataset

df = pd.read\_csv("/content/Loan\_data.csv")

# 1. Viewing and Inspecting DataFrames

print(" Shape of dataset:", df.shape)

print("\n Data Types and Null Values:")

print(df.info())

print("\n First 5 Rows:")

print(df.head())

print("\n Missing values in each column:")

print(df.isnull().sum())

# 2. Filtering and Subsetting Data

# Applicants with income > 5000

high\_income = df[df['ApplicantIncome'] > 5000]

print(f"\n Number of high income applicants (>5000): {high\_income.shape[0]}")

# Approved loans for self-employed applicants

approved\_self\_employed = df[(df['Self\_Employed'] == 'Yes') & (df['Loan\_Status'] == 'Y')]

print(f" Approved self-employed loans: {approved\_self\_employed.shape[0]}")

# Urban applicants with coapplicants

urban\_with\_coapp = df[(df['Property\_Area'] == 'Urban') & (df['CoapplicantIncome'] > 0)]

print(f" Urban applicants with coapplicants: {urban\_with\_coapp.shape[0]}")

# 3. Descriptive Statistics

# LoanAmount column (drop NaNs)

loan\_amt = df['LoanAmount'].dropna()

mean\_loan = loan\_amt.mean()

median\_loan = loan\_amt.median()

mode\_loan = loan\_amt.mode()[0]

range\_loan = loan\_amt.max() - loan\_amt.min()

variance\_loan = loan\_amt.var()

std\_loan = loan\_amt.std()

print("\n LoanAmount Statistics:")

print(f"Mean: {mean\_loan:.2f}")

print(f"Median: {median\_loan}")

print(f"Mode: {mode\_loan}")

print(f"Range: {range\_loan}")

print(f"Variance: {variance\_loan:.2f}")

print(f"Standard Deviation: {std\_loan:.2f}")

# ApplicantIncome column

income = df['ApplicantIncome']

mean\_income = income.mean()

median\_income = income.median()

mode\_income = income.mode()[0]

range\_income = income.max() - income.min()

variance\_income = income.var()

std\_income = income.std()

print("\n ApplicantIncome Statistics:")

print(f"Mean: {mean\_income:.2f}")

print(f"Median: {median\_income}")

print(f"Mode: {mode\_income}")

print(f"Range: {range\_income}")

print(f"Variance: {variance\_income:.2f}")

print(f"Standard Deviation: {std\_income:.2f}")

# 4. Summary Table

print("\n Summary Statistics for All Numeric Columns:")

print(df.describe())

# 5. Group Analysis: Mean LoanAmount by Education

loan\_by\_education = df.groupby('Education')['LoanAmount'].mean()

print("\n Mean LoanAmount by Education:")

print(loan\_by\_education)

**OUTPUT:-**

Shape of dataset: (614, 13)

Data Types and Null Values:

<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

None

First 5 Rows:

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

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

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

Missing values in each column:

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

Number of high income applicants (>5000): 191

Approved self-employed loans: 56

Urban applicants with coapplicants: 107

LoanAmount Statistics:

Mean: 146.41

Median: 128.0

Mode: 120.0

Range: 691.0

Variance: 7325.19

Standard Deviation: 85.59

ApplicantIncome Statistics:

Mean: 5403.46

Median: 3812.5

Mode: 2500

Range: 80850

Variance: 37320390.17

Standard Deviation: 6109.04

Summary Statistics for All Numeric Columns:

ApplicantIncome CoapplicantIncome LoanAmount Loan\_Amount\_Term \

count 614.000000 614.000000 592.000000 600.00000

mean 5403.459283 1621.245798 146.412162 342.00000

std 6109.041673 2926.248369 85.587325 65.12041

min 150.000000 0.000000 9.000000 12.00000

25% 2877.500000 0.000000 100.000000 360.00000

50% 3812.500000 1188.500000 128.000000 360.00000

75% 5795.000000 2297.250000 168.000000 360.00000

max 81000.000000 41667.000000 700.000000 480.00000

Credit\_History

count 564.000000

mean 0.842199

std 0.364878

min 0.000000

25% 1.000000

50% 1.000000

75% 1.000000

max 1.000000

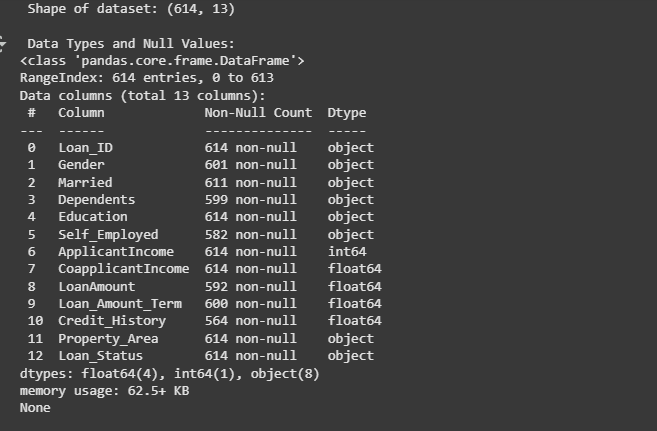
Mean LoanAmount by Education:

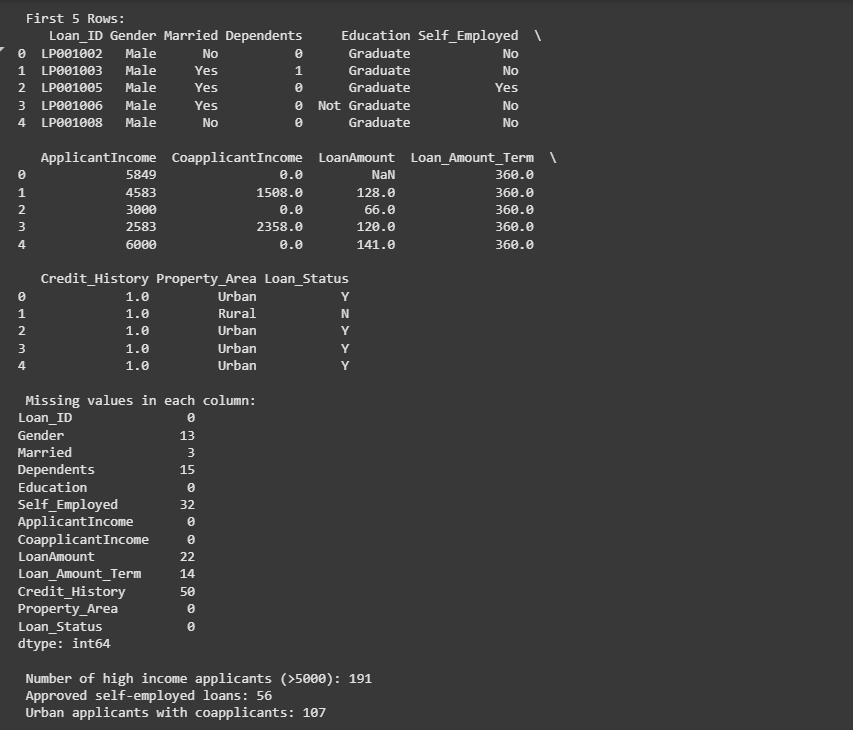
Education

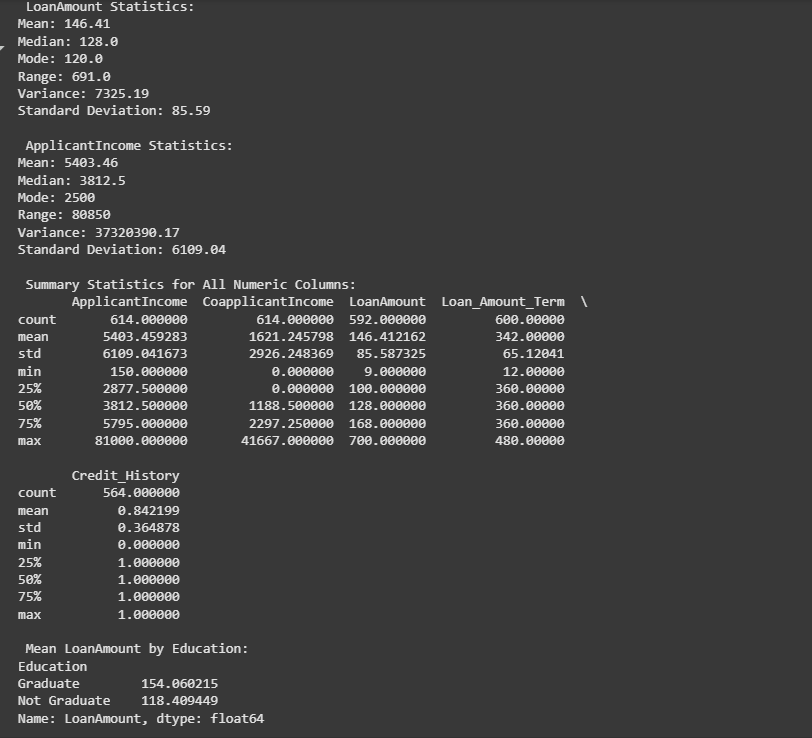
Graduate 154.060215

Not Graduate 118.409449

Name: LoanAmount, dtype: float64







**EXP-5**

**PROGRAM:-**

import matplotlib.pyplot as plt

import numpy as np

# 1. LINE CHART (Monthly Sales Trend)

months = np.arange(1, 13)

sales = [120, 150, 170, 180, 160, 200, 220, 210, 190, 230, 250, 270]

plt.plot(months, sales, marker='o', linestyle='-', color='b', label="Monthly Sales")

plt.title("Line Chart - Monthly Sales Trend")

plt.xlabel("Month")

plt.ylabel("Sales (in units)")

plt.xticks(months)

plt.legend()

plt.grid(True)

plt.show()

# 2. BAR CHART (Average Scores by Subject)

subjects = ['Math', 'Science', 'English', 'History', 'Computer']

avg\_scores = [78, 85, 72, 65, 90]

plt.bar(subjects, avg\_scores, color='orange')

plt.title("Bar Chart - Average Scores by Subject")

plt.xlabel("Subjects")

plt.ylabel("Average Score")

plt.ylim(0, 100)

plt.show()

# 3. HISTOGRAM (Exam Scores Distribution)

np.random.seed(42)

exam\_scores = np.random.normal(loc=70, scale=10, size=200)

plt.hist(exam\_scores, bins=15, color='green', edgecolor='black')

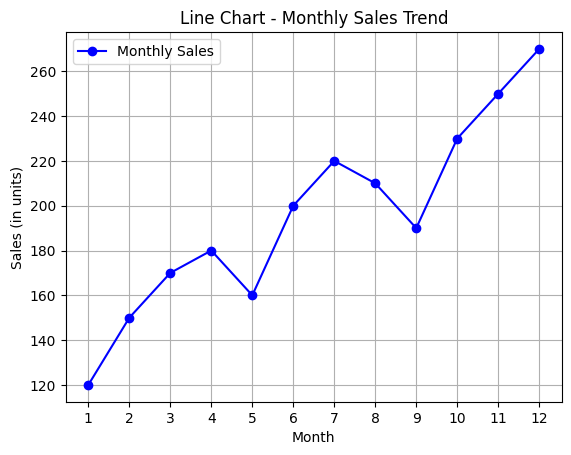
plt.title("Histogram - Exam Scores Distribution")

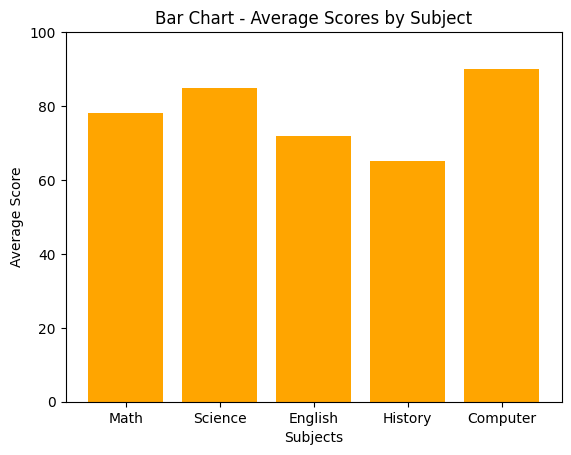
plt.xlabel("Score Range")

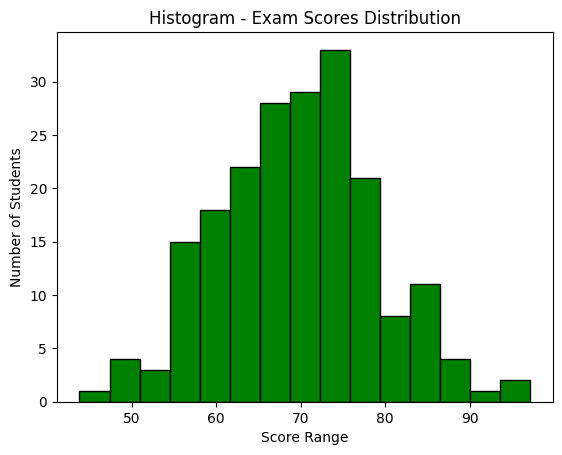
plt.ylabel("Number of Students")

plt.show()

**OUTPUT:-**

****

****

****