**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):

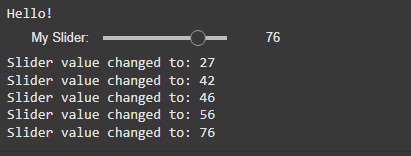
print(f"Slider value changed to: {change['new']}")

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

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())

df\_excel = df\_csv.copy()

print("New DataFrame created from CSV")

print(df\_excel.head())

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

print("Excel file saved successfully")

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]

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

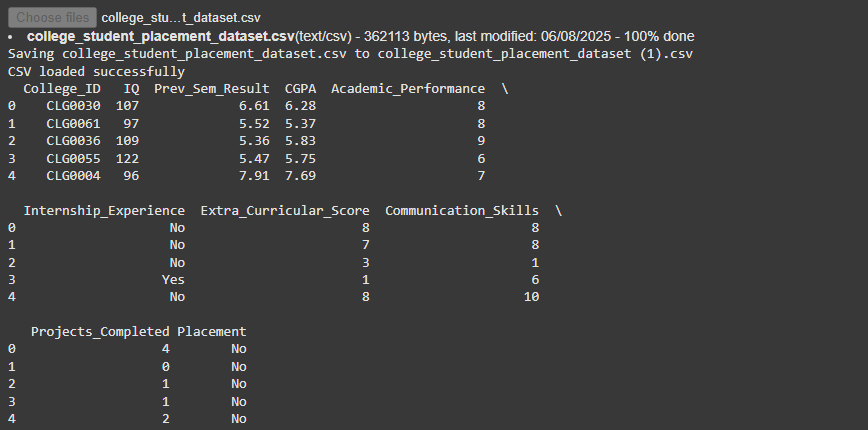
print("Scraped table:")

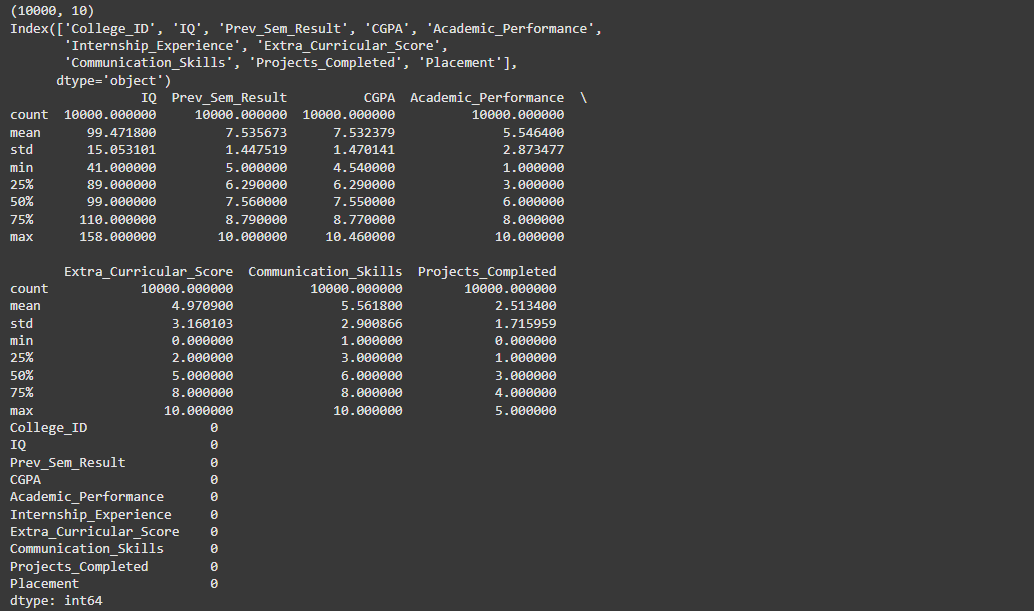
print(df\_web.head())

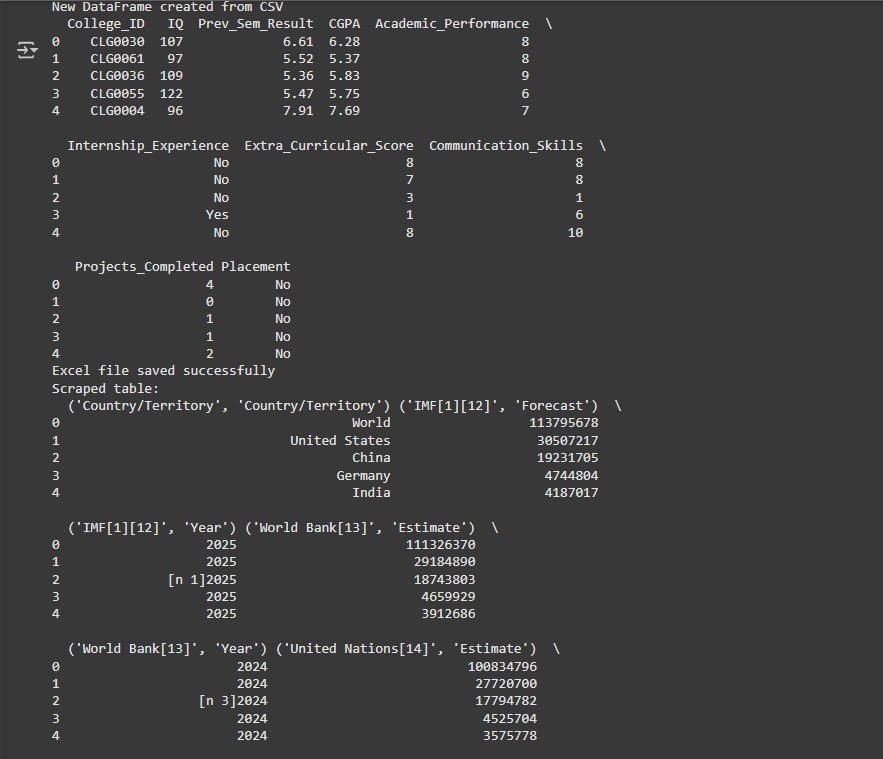
df\_web.to\_excel("scraped\_gdp\_table.xlsx", index=False)

print("Scraped table saved to Excel")

**OUTPUT:-**

****

****

****

**EXP - 3**

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

**PROGRAM**

import pandas as pd

import numpy as np

from sklearn.preprocessing import MinMaxScaler, StandardScaler

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

print("🔍 Dataset Preview:")

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

print("\n Missing Values:")

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

if 'Age' in df.columns:

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

if 'Embarked' in df.columns:

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

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)

df.dropna(inplace=True)

duplicates = df.duplicated().sum()

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

df.drop\_duplicates(inplace=True)

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)

if 'Survived' in df.columns:

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

if 'Pclass' in df.columns:

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

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()

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

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

scaler = MinMaxScaler()

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

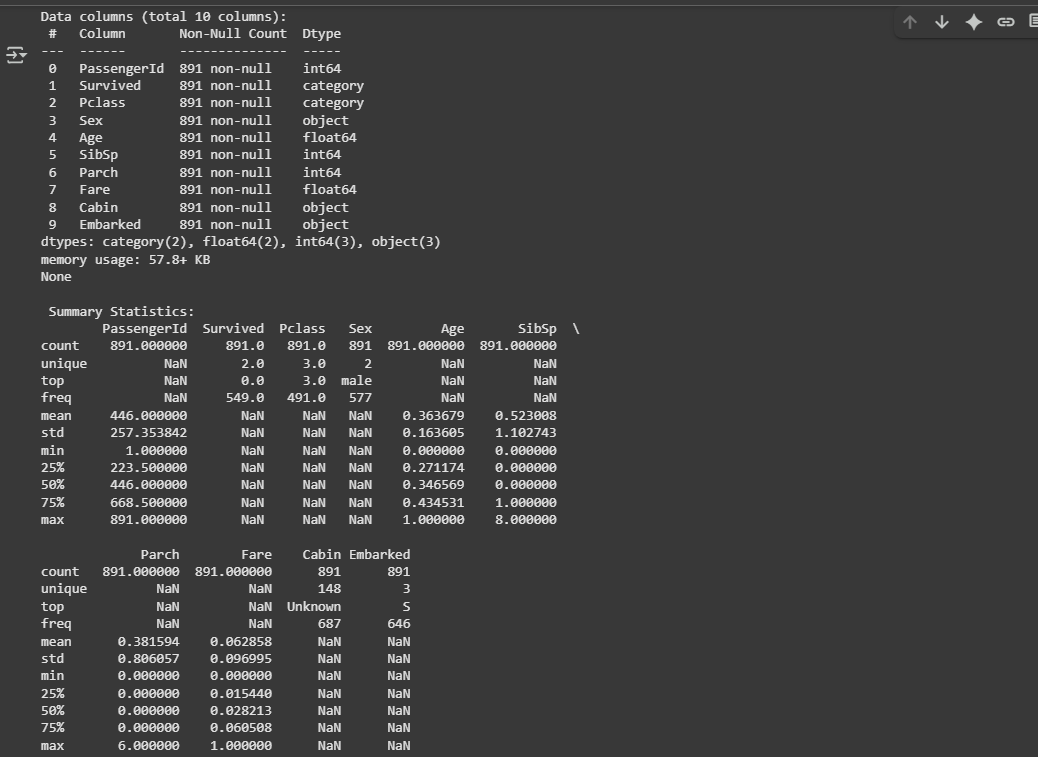
print("\n Cleaned Data Info:")

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

print("\n📊 Summary Statistics:")

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

**OUTPUT:-**



**Exp - 4**

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

**PROGRAM:**

import pandas as pd

import numpy as np

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

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())

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

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

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

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

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

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

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}")

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}")

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

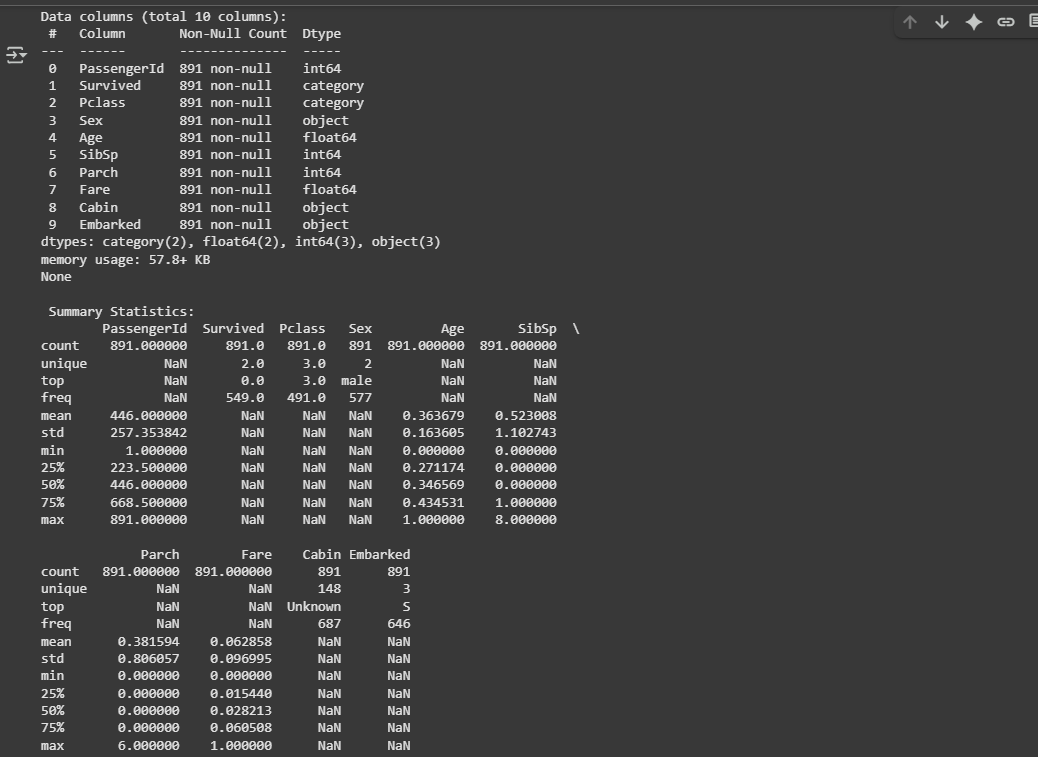
print(df.describe())

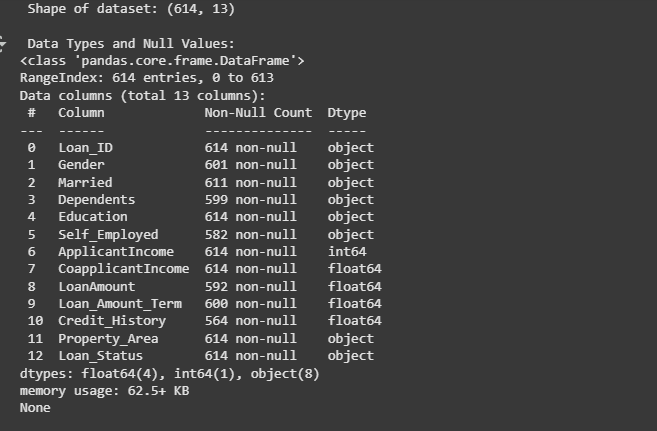
loan\_by\_education = df.groupby('Education')['LoanAmount'].mean()

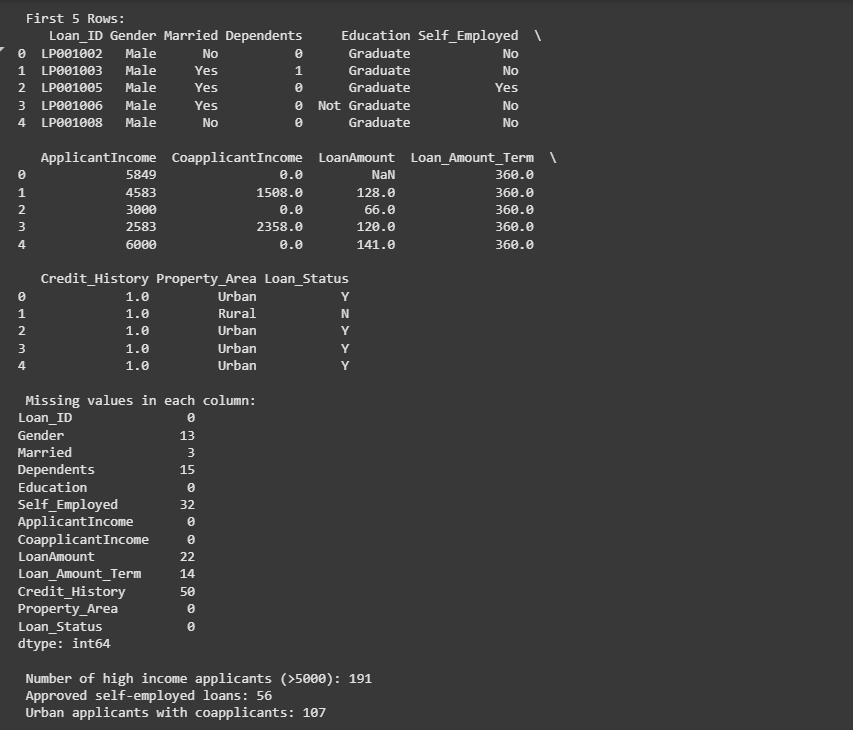
print("\n Mean LoanAmount by Education:")

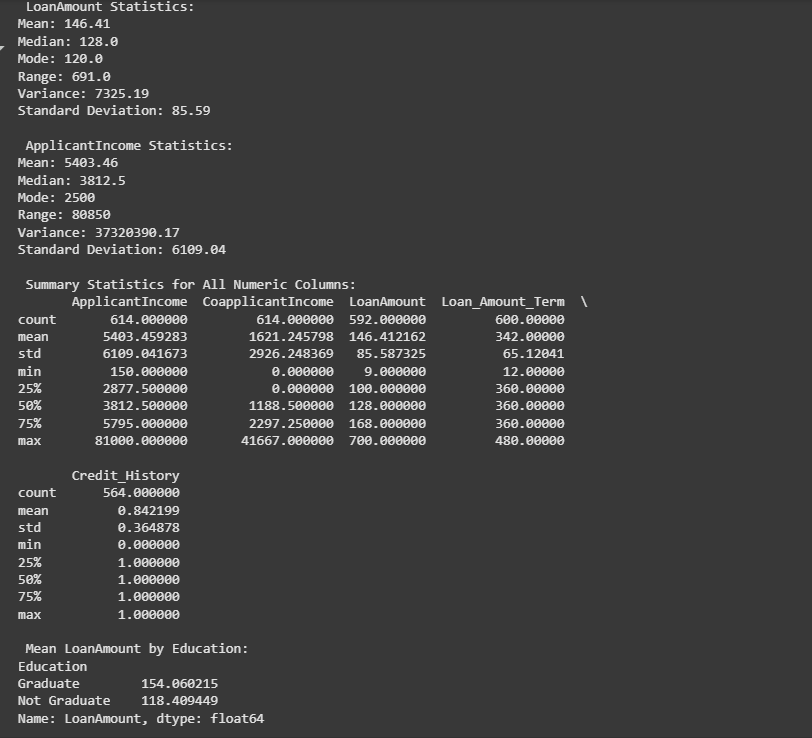
print(loan\_by\_education)

**OUTPUT:-**









**EXP-5**

**PROGRAM:-**

import matplotlib.pyplot as plt

import numpy as np

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()

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()

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')

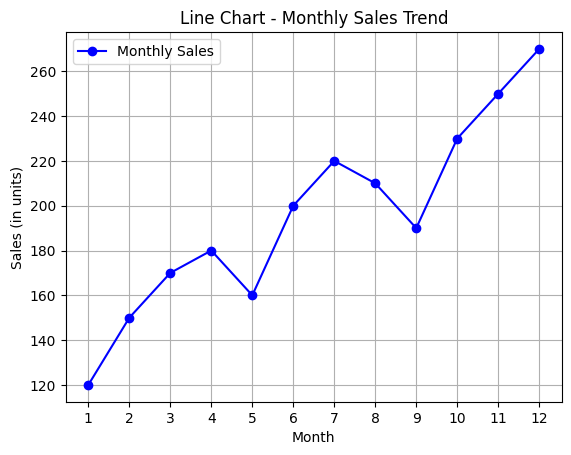
plt.title("Histogram - Exam Scores Distribution")

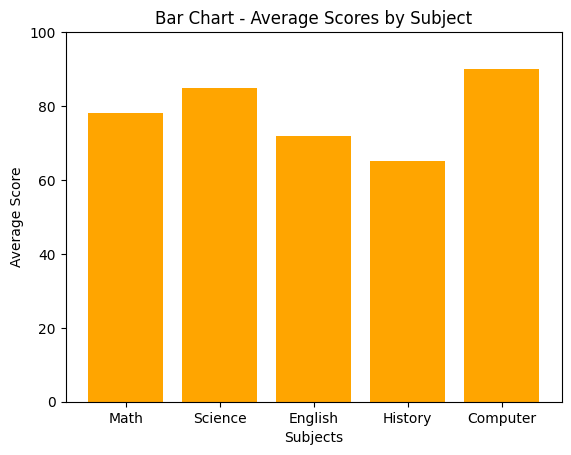
plt.xlabel("Score Range")

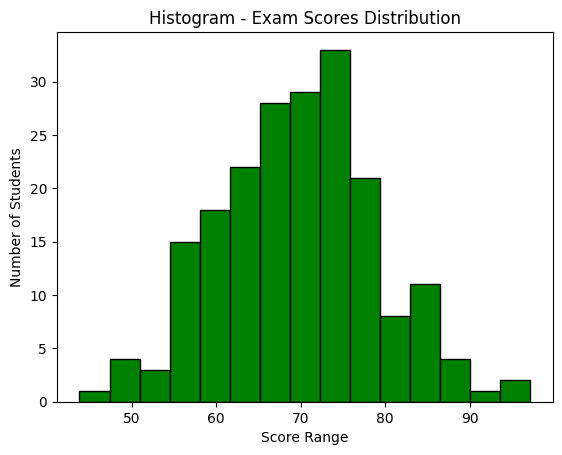
plt.ylabel("Number of Students")

plt.show()

**OUTPUT:-**

****

****

****