Department of Computer Science
DATA ANALYTICS USING PYTHON RECORD

Program No: 1	Date: 09/07/2025
Name:	Reg No:
Program Title : Swap two variables using a temporary variable.	

Program :

```
 a=input("enter 1st number:")
b=input("enter 1st number:")
t=a
a=b
b=t
print(f"a is {a} b is {b}")
```

Output Screenshot/Text

```
enter 1st number:1
enter 1st number:2
a is 2 b is 1
```

Department of Computer Science
DATA ANALYTICS USING PYTHON RECORD

Program No: 2	Date: 09/07/2025
Name:	Reg No:
Program Title : Takes a user input string and returns the number of vowels in it	

Program :

```
 str=input("enter string : ")
c=0
for i in range(len(str)):
  if str[i] == 'a' or str[i] == 'e' or str[i] == 'i' or str[i] == 'o' or str[i] == 'u':
    c=c+1
print(f"count of vowels is {c}")
```

Output Screenshot/Text

```
enter stringaeiouqwe
count of vowels is 6
```

Department of Computer Science
DATA ANALYTICS USING PYTHON RECORD

Program No: 3	Date: 09/07/2025
Name:	Reg No:

Program : Title Write a program to reverse a string without using built-in functions.

Program :

```
def reverse_string(s):
    reversed_str = ''
    for i in range(len(s) - 1, -1, -1):
        reversed_str += s[i]
    return reversed_str

input_str = input("Enter a string: ")
result = reverse_string(input_str)
print("Reversed string:", result)
```

Output Screenshot/Text

```
Enter a string: hi
Reversed string: ih
```

Department of Computer Science
DATA ANALYTICS USING PYTHON RECORD

Program No: 4	Date: 09/07/2025
Name:	Reg No:
Program Title : Concatenate two strings using variables and print the result.	

Program :

```
str1=input("enter string1")
str2=input("enter string2")
str3=str1+str2
print(str3)
```

Output Screenshot/Text

```
enter string1hi world
enter string2hi heell
hi worldhi heell
```

Department of Computer Science
DATA ANALYTICS USING PYTHON RECORD

Program No: 5	Date: 09/07/2025
Name:	Reg No:

Program Title : Write a program that checks whether a substring exists in a given string.

Program :

```
def is_substring(main_str, sub_str):
    main_len = len(main_str)
    sub_len = len(sub_str)

    for i in range(main_len - sub_len + 1):
        match = True
        for j in range(sub_len):
            if main_str[i + j] != sub_str[j]:
                match = False
                break
        if match:
            return True
    return False

main_string = input("Enter the main string: ")
substring = input("Enter the substring to check: ")

if is_substring(main_string, substring):
    print("Substring exists in the main string.")
else:
    print("Substring does not exist in the main string.")
```

Output Screenshot/Text

```
Enter the main string: hello
Enter the substring to check: ll
Substring exists in the main string.
```

Department of Computer Science
DATA ANALYTICS USING PYTHON RECORD

Program No: 6	Date: 09/07/2025
Name:	Reg No:

Program Title : Write a program to find the maximum and minimum elements in a list.

Program :

```
def find_max_min(numbers):
    if not numbers:  # empty list check
        return None, None
        max_val = numbers[0]
    min_val = numbers[0]

    for num in numbers[1:]:  # start from 2nd element
        if num > max_val:
            max_val = num
        elif num < min_val:
            min_val = num

    return max_val, min_val
try:
    input_str = input("Enter numbers separated by spaces: ")
    num_list = [float(x) for x in input_str.split()]  # float allows decimals
    maximum, minimum = find_max_min(num_list)
    if maximum is None:
        print("No numbers entered.")
    else:
        print("Maximum element:", maximum)
        print("Minimum element:", minimum)
except ValueError:
    print("Please enter only numbers separated by spaces.")
```

Output:

```
Enter numbers separated by spaces: 1 2 4 55 22
Maximum element: 55.0
Minimum element: 1.0
```

Department of Computer Science
DATA ANALYTICS USING PYTHON RECORD

Program No: 7	Date: 09/07/2025
Name:	Reg No:
Program Title : to remove duplicates from a list	

Program :

```
def remove_duplicates(lst):
    unique_list = []
    for item in lst:
        if item not in unique_list:
            unique_list.append(item)
    return unique_list

input_str = input("Enter list elements separated by spaces: ")
input_list = [int(x) for x in input_str.split()]

result = remove_duplicates(input_list)
print("List after removing duplicates:", result)
```

Output Screenshot/Text

```
Enter list elements separated by spaces: 1 2 3 3 4 4 5 66
List after removing duplicates: [1, 2, 3, 4, 5, 66]
```

Department of Computer Science
DATA ANALYTICS USING PYTHON RECORD

Program No: 8	Date: 09/07/2025
Name:	Reg No:
Program Title : Create a tuple of 5 numbers. Print the sum and average of the numbers.	

Program :

```
numbers = (10, 20, 30, 40, 50)

total = 0
for num in numbers:
    total += num

average = total / len(numbers)

print("Tuple:", numbers)
print("Sum:", total)
print("Average:", average)
```

Output Screenshot/Text

```
Tuple: (10, 20, 30, 40, 50)
Sum: 150
Average: 30.0
```

Department of Computer Science
DATA ANALYTICS USING PYTHON RECORD

Program No: 9	Date: 09/07/2025
Name:	Reg No:

Program Title : Write a program to find the second largest number in a list.

Program :

```
def find_second_largest(lst):
    if len(lst) < 2:
        return None
    largest = second_largest = float('-inf')
    for num in lst:
        if num > largest:
            second_largest = largest
            largest = num
        elif num > second_largest and num != largest:
            second_largest = num
    if second_largest == float('-inf'):
        return None
    return second_largest

input_str = input("Enter list elements separated by spaces: ")
input_list = [int(x) for x in input_str.split()]

result = find_second_largest(input_list)

if result is not None:
    print("Second largest number is:", result)
else:
    print("Cannot determine second largest (not enough unique values).")
```

Output Screenshot/Text

```
Enter list elements separated by spaces: 1 2 3 4 5 6
Second largest number is: 5
```

Department of Computer Science
DATA ANALYTICS USING PYTHON RECORD

Program No: 10	Date: 09/07/2025
Name:	Reg No:

Program Title : .Write a Python function to count how many times an element appears in a list.

Program :

```
def count_occurrences(lst, element):
    count = 0
    for item in lst:
        if item == element:
            count += 1
    return count

input_str = input("Enter list elements separated by spaces: ")
input_list = input_str.split()

element_to_count = input("Enter the element to count: ")

occurrences = count_occurrences(input_list, element_to_count)
print(f"Element '{element_to_count}' appears {occurrences} time(s) in the list.")
```

Output Screenshot/Text

```
Enter list elements separated by spaces: 1 2 3 4 5 6
Enter the element to count: 4
Element '4' appears 1 time(s) in the list.
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 11	Date: 09/07/2025
Name:	Reg No:

Program Title : Write a program to find the union and intersection of two sets.

Program :

```
# Program to find Union and Intersection of two sets

# Define two sets
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}

# Union of sets
union_set = set1.union(set2)   # or set1 | set2

# Intersection of sets
intersection_set = set1.intersection(set2)   # or set1 & set2

# Display results
print("Set 1:", set1)
print("Set 2:", set2)
print("Union of Set 1 and Set 2:", union_set)
print("Intersection of Set 1 and Set 2:", intersection_set)
```

Output Screenshot/Text

```
Set 1: {1, 2, 3, 4, 5}
Set 2: {4, 5, 6, 7, 8}
Union of Set 1 and Set 2: {1, 2, 3, 4, 5, 6, 7, 8}
Intersection of Set 1 and Set 2: {4, 5}
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 12	Date: 09/07/2025
Name:	Reg No:

Program Title : Create a dictionary with 5 key-value pairs and print all keys and values.

Program :
```
# Create a dictionary with 5 key-value pairs
my_dict = {
    "name": "Alice",
    "age": 22,
    "course": "Computer Science",
    "year": "Final",
    "grade": "A"
}

# Print all keys
print("Keys in dictionary:")
for key in my_dict.keys():
    print(key)

# Print all values
print("\nValues in dictionary:")
for value in my_dict.values():
    print(value)
```

Output :

```
Keys in dictionary:
name
age
course
year
grade

Values in dictionary:
Alice
22
Computer Science
Final
A
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 13	Date: 09/07/2025
Name:	Reg No:

Program Title : Write a program to count the frequency of characters in a string using a dictionary.

Program :

```
# Program to count frequency of characters in a string

# Input string
text = "programming"

# Create an empty dictionary
freq = {}

# Loop through each character in the string
for char in text:
    if char in freq:
        freq[char] += 1   # Increment count if already exists
    else:
        freq[char] = 1    # Initialize with 1 if new character

# Print character frequencies
print("Character Frequency:")
for key, value in freq.items():
    print(f"{key}: {value}")
```

Output Screenshot/Text

```
Character Frequency:
p: 1
r: 2
o: 1
g: 2
a: 1
m: 2
i: 1
n: 1
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 14	Date: 09/07/2025
Name:	Reg No:

Program Title : Write a program to print all prime numbers between 1 and 50.

Program :

```
# Program to print all prime numbers between 1 and 50

print("Prime numbers between 1 and 50 are:")

for num in range(2, 51):  # Start from 2 (since 1 is not prime)
    is_prime = True
    for i in range(2, int(num**0.5) + 1):  # Check divisors up to sqrt(num)
        if num % i == 0:
            is_prime = False
            break
    if is_prime:
        print(num, end=" ")
```

Output Screenshot/Text

```
Prime numbers between 1 and 50 are:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 15	Date: 09/07/2025
Name:	Reg No:

Program Title : Write a program to take user input and generate the multiplication table of the given number using a for loop.

Program :

```
 # Program to generate multiplication table

# Take user input
num = int(input("Enter a number: "))


# Print multiplication table
print(f"\nMultiplication Table of {num}:")
for i in range(1, 11):   # From 1 to 10
    print(f"{num} x {i} = {num * i}")
```

Output Screenshot/Text

```
Enter a number: 5

Multiplication Table of 5:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 16	Date: 09/07/2025
Name:	Reg No:
Program Title : Write a program to print Fibonacci series up to n terms.	

Program :

```
 # Program to print Fibonacci series up to n terms

# Take user input
n = int(input("Enter the number of terms: "))

# First two terms of Fibonacci series
a, b = 0, 1

print("\nFibonacci Series:")

for i in range(n):
    print(a, end=" ")
    a, b = b, a + b
```

Output Screenshot/Text

```
Enter the number of terms: 12

Fibonacci Series:
0 1 1 2 3 5 8 13 21 34 55 89
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 17	Date: 10/09/2025
Name:	Reg No:
Program Title : Create a Calculator class with methods add, subtract, multiply, divide	

Program :# Calculator class with basic operations

```
class Calculator:
    def add(self, a, b):
        return a + b

    def subtract(self, a, b):
        return a - b

    def multiply(self, a, b):
        return a * b

    def divide(self, a, b):
        if b == 0:
            return "Error! Division by zero."
        return a / b


# Main program
calc = Calculator()

print("Simple Calculator")
print("1. Add")
print("2. Subtract")
print("3. Multiply")
print("4. Divide")

choice = int(input("\nEnter your choice (1-4): "))

num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))


if choice == 1:
    print("Result:", calc.add(num1, num2))

elif choice == 2:
```

```
    print("Result:", calc.subtract(num1, num2))


elif choice == 3:
    print("Result:", calc.multiply(num1, num2))

elif choice == 4:
    print("Result:", calc.divide(num1, num2))

else:
    print("Invalid choice!")
```

Output Screenshot/Text

```
Simple Calculator
1. Add
2. Subtract
3. Multiply
4. Divide

Enter your choice (1-4): 1
Enter first number: 1
Enter second number: 2
Result: 3.0
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 18	Date: 10/09/2025
Name:	Reg No:

Program Title : Implement a Vehicle → Car → ElectricCar hierarchy and demonstrate multilevel inheritance.

Program :

```
class Vehicle:
    def __init__(self, brand):
        self.brand = brand

    def display_info(self):
        print(f"Brand: {self.brand}")


class Car(Vehicle):
    def __init__(self, brand, model):
        super().__init__(brand)
        self.model = model

    def display_info(self):
        super().display_info()
        print(f"Model: {self.model}")


class ElectricCar(Car):
    def __init__(self, brand, model, battery_capacity):
        super().__init__(brand, model)
        self.battery_capacity = battery_capacity
    def display_info(self):
        super().display_info()
        print(f"Battery Capacity: {self.battery_capacity} kWh")

brand = input("Enter the brand of the electric car: ")
model = input("Enter the model: ")
battery_capacity = float(input("Enter the battery capacity (kWh): "))

ev = ElectricCar(brand, model, battery_capacity)
print("\nElectric Car Details:")
ev.display_info()
```

Output Screenshot/Text

```
Enter the brand of the electric car: tesla
Enter the model: 3
Enter the battery capacity (kWh): 78

Electric Car Details:
Brand: tesla
Model: 3
Battery Capacity: 78.0 kWh
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 19	Date: 10/09/2025
Name:	Reg No:

Program Title : Create a Shape base class with a method area(). Derive Rectangle and Circle classes overriding area().

Program :

```
import math
# Base class
class Shape:
    def area(self):
        pass  # Base method, to be overridden by derived classes
# Derived class for Rectangle
class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width
# Derived class for Circle
class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius ** 2
# Taking user input
shape_type = input("Enter shape type (rectangle/circle): ").strip().lower()

if shape_type == "rectangle":
    length = float(input("Enter length: "))
    width = float(input("Enter width: "))
    rect = Rectangle(length, width)
    print(f"Area of Rectangle: {rect.area()}")
elif shape_type == "circle":
    radius = float(input("Enter radius: "))
    circ = Circle(radius)
    print(f"Area of Circle: {circ.area():.2f}")
```

```
else:
    print("Invalid shape type!")
```

Output Screenshot/Text

```
Enter shape type (rectangle/circle): circle
Enter radius: 3
Area of Circle: 28.27
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 20	Date: 10/09/2025
Name:	Reg No:

Program Title : Write a program to read a text file and count words, lines, and characters.

Program :# Program to read a file and count lines, words, and characters

```
# Take filename as input
filename = input("Enter the filename: ")
try:
    with open(filename, 'r') as file:
        text = file.read()
    # Count lines
    with open(filename, 'r') as file:
        lines = file.readlines()
        num_lines = len(lines)
    # Count words
    words = text.split()
    num_words = len(words)
    # Count characters
    num_chars = len(text)
    # Display results
    print(f"Lines: {num_lines}")
    print(f"Words: {num_words}")
    print(f"Characters: {num_chars}")

except FileNotFoundError:
    print("File not found. Please check the filename and try again.")
```

Output Screenshot/Text

```
Enter the filename: sample.txt
Lines: 3
Words: 3
Characters: 20
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 21	Date: 10/09/2025
Name:	Reg No:
Program Title : Demonstrate use of read(), readline(), readlines(), write(), and writelines().	

Program :

```
# File operations demonstration

filename = "demo_file.txt"

# 1. Writing to a file using write() and writelines()
with open(filename, "w") as file:
    file.write("Hello World!\n")  # Write a single line
    lines = ["Python is fun.\n", "File handling is easy.\n", "End of file.\n"]
    file.writelines(lines)  # Write multiple lines at once

# 2. Reading entire file using read()
with open(filename, "r") as file:
    content = file.read()
    print("Using read():")
    print(content)

# 3. Reading file line by line using readline()
with open(filename, "r") as file:
    print("Using readline():")
    line = file.readline()
    while line:
        print(line, end="")
        line = file.readline()

# 4. Reading all lines into a list using readlines()
with open(filename, "r") as file:
    lines_list = file.readlines()
    print("\nUsing readlines():")
    print(lines_list)
```

Output Screenshot/Text

```
Using read():
Hello World!
Python is fun.
File handling is easy.
End of file.

Using readline():
Hello World!
Python is fun.
File handling is easy.
End of file.

Using readlines():
['Hello World!\n', 'Python is fun.\n', 'File handling is easy.\n', 'End of file.\n']
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 22	Date: 10/09/2025
Name:	Reg No:

Program Title : Print current date, time, and day of the week.

Program :

```
from datetime import datetime

# Get current date and time
now = datetime.now()

# Extract date, time, and day
current_date = now.date()
current_time = now.time().strftime("%H:%M:%S")  # Format time as HH:MM:SS
day_of_week = now.strftime("%A")  # Full weekday name

# Display results
print(f"Current Date: {current_date}")
print(f"Current Time: {current_time}")
print(f"Day of the Week: {day_of_week}")
```

Output Screenshot/Text

```
Current Date: 2025-10-01
Current Time: 03:10:31
Day of the Week: Wednesday
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 23	Date: 10/09/2025
Name:	Reg No:

Program Title : Write a program to calculate age from a given date of birth.

Program :from datetime import datetime

```
# Take user input for date of birth
dob_input = input("Enter your date of birth (YYYY-MM-DD): ")

# Convert string input to a date object
dob = datetime.strptime(dob_input, "%Y-%m-%d").date()

# Get today's date
today = datetime.today().date()

# Calculate age
age = today.year - dob.year

# Adjust if birthday hasn't occurred yet this year
if (today.month, today.day) < (dob.month, dob.day):
    age -= 1

# Display age
print(f"You are {age} years old.")
```

Output Screenshot/Text

```
Enter your date of birth (YYYY-MM-DD): 2004-01-16
You are 21 years old.
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 24	Date: 10/09/2025
Name:	Reg No:

Program Title : Write a program to handle division by zero exception.

Program :

```
# Program to handle division by zero exception

# Take user input
numerator = float(input("Enter numerator: "))
denominator = float(input("Enter denominator: "))

try:
    result = numerator / denominator
    print(f"Result: {result}")
except ZeroDivisionError:
    print("Error! Division by zero is not allowed.")
```

Output Screenshot/Text

```
Enter numerator: 23
Enter denominator: 0
Error! Division by zero is not allowed.
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 25	Date: 10/09/2025
Name:	Reg No:

Program Title : Create a custom exception class NegativeValueError and raise it when negative input is given.

Program :
```
# Custom exception class
class NegativeValueError(Exception):
    def __init__(self, value):
        super().__init__(f"Negative value not allowed: {value}")
        self.value = value


# Function to take input and check for negative value
def get_positive_number():
    num = float(input("Enter a positive number: "))
    if num < 0:
        raise NegativeValueError(num)
    return num


# Main program
try:
    number = get_positive_number()
    print(f"You entered: {number}")
except NegativeValueError as e:
    print("Error:", e)
except ValueError:
    print("Invalid input! Please enter a number.")
```

Output Screenshot/Text

```
Enter a positive number: -78
Error: Negative value not allowed: -78.0
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 26	Date: 10/09/2025
Name:	Reg No:

Program Title : Demonstrate use of try-except-else-finally

Program :

```
try:
      numerator = float(input("Enter numerator: "))
    denominator = float(input("Enter denominator: "))

    result = numerator / denominator

except ZeroDivisionError:
    print("Error! Division by zero is not allowed.")

except ValueError:
    print("Invalid input! Please enter numeric values.")

else:
    print(f"Result of division: {result}")

finally:
      print("Program execution completed.")
```

Output Screenshot/Text

```
Enter numerator: 20
Enter denominator: 0
Error! Division by zero is not allowed.
Program execution completed.
```

Department of Computer Science

DATA ANALYTICS USING PYTHON RECORD

Program No:27	Date: 11/09/2025
Name:	Reg No:
Program Title: Implementation of MySQL connection using Python	

```
#!/usr/bin/python3

import cgi
import mysql.connector
from mysql.connector import Error

print("Content-Type: text/html\n")

print("<html>")
print("<head><title>MySQL CGI Example</title></head>")
print("<body>")
print("<h1>Users from Database</h1>")

connection = None
try:
    connection = mysql.connector.connect(
        host="localhost",
        database="testdb",
        user="root",
        password=""
    )

    if connection.is_connected():
        cursor = connection.cursor()
        cursor.execute("SELECT id, username, email FROM users")
        records = cursor.fetchall()

        if cursor.rowcount > 0:
            print('<table border="1">')
            print("<tr><th>ID</th><th>Username</th><th>Email</th></tr>")
            for row in records:
                print(f"<tr><td>{row[0]}</td><td>{row[1]}</td><td>{row[2]}</td></tr>")
            print("</table>")
        else:
            print("<p>No records found in the 'users' table.</p>")

        cursor.close()
```

```
except Error as e:
    print(f"<h2>Error connecting to MySQL</h2>")
    print(f"<p>{e}</p>")

finally:
    if connection and connection.is_connected():
        connection.close()

print("</body>")
print("</html>")
```

Output Screenshot/Text

Users from Database

ID	Username	Email
1	john	john@gmail.com

Department of Computer Science

DATA ANALYTICS USING PYTHON RECORD

Program No:28	Date: 11/09/2025
Name:	Reg No:
Program Title: Implementation of SqLite3 connection using Python.	

```
#!/usr/bin/python3

import sqlite3
import cgi

print("Content-Type: text/html\n")

print("<html>")
print("<head><title>SQLite User List</title></head>")
print("<body>")
print("<h1>Current Users in Database</h1>")

db_path = "/opt/lampp/var/userdb.db"
connection = None

try:
    connection = sqlite3.connect(db_path)
    cursor = connection.cursor()

    cursor.execute("SELECT id, username, email FROM users ORDER BY id")
    all_users = cursor.fetchall()

    if all_users:
        print('<table border="1" style="width:50%;">')
        print("<tr><th>ID</th><th>Username</th><th>Email</th></tr>")
        for user in all_users:
            print(f"<tr><td>{user[0]}</td><td>{user[1]}</td><td>{user[2]}</td></tr>")
        print("</table>")
    else:
        print("<p>No users found in the database.</p>")

except sqlite3.Error as e:
    print(f"<h2>Database Error</h2>")
    print(f"<p>An error occurred: {e}</p>")

finally:
    if connection:
        connection.close()
```

```
print("</body>")
print("</html>")
```

Output Screenshot/Text

Current Users in Database

ID	Username	Email
1	indrajith	indrajithpg10@gmail.com
2	denin	denin@gmail.com

Department of Computer Science

DATA ANALYTICS USING PYTHON RECORD

Program No:29	Date: 11/09/2025
Name:	Reg No:
Program Title: Implementation of SqLite3 connection using Python.	

```
#!/usr/bin/python3

import cgi
import mysql.connector
from mysql.connector import Error

DB_CONFIG = {
    'host': 'localhost',
    'user': 'root',
    'password': '',
    'database': 'testdb'
}

def main():
    print("Content-Type: text/html\n")

    form = cgi.FieldStorage()
    action = form.getvalue('action')
    message = ""

    conn = get_db_connection()
    if not conn:
        print_html_page("<h2>Error: Could not connect to the database.</h2>")
        return

    create_users_table(conn)

    try:
        if action == 'add':
            username = form.getvalue('username')
            email = form.getvalue('email')
            if username and email:
                add_user(conn, username, email)
                message = f"Success: User '{username}' was added."
            else:
                message = "Error: Please provide both username and email."

        elif action == 'update':
```

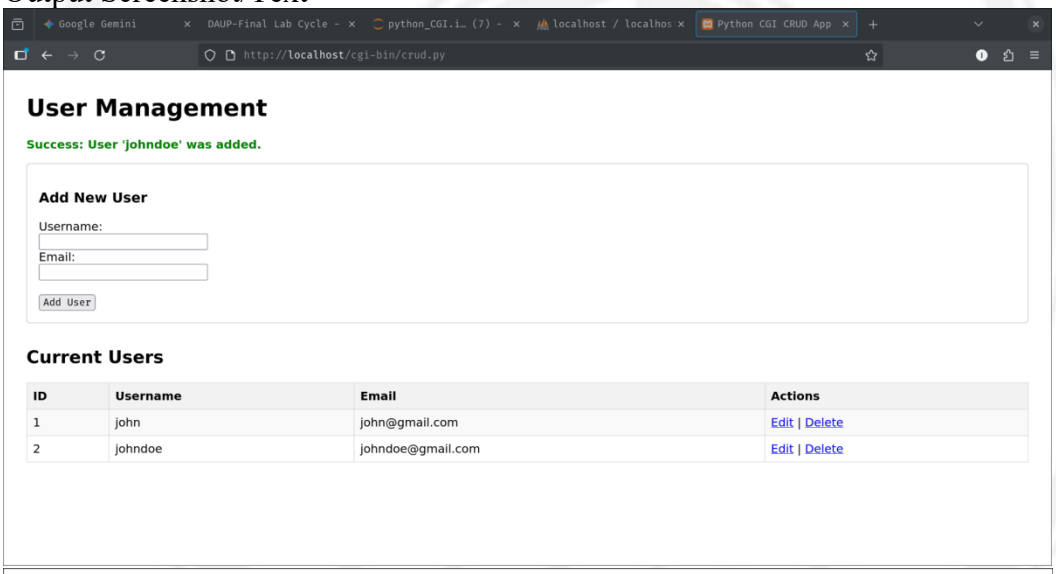
```
            user_id = form.getvalue('id')
            username = form.getvalue('username')
            email = form.getvalue('email')
            if user_id and username and email:
                update_user(conn, user_id, username, email)
                message = f"Success: User ID {user_id} was updated."
            else:
                message = "Error: Missing data for update."

        elif action == 'delete':
            user_id = form.getvalue('id')
            if user_id:
                delete_user(conn, user_id)
                message = f"Success: User ID {user_id} was deleted."

    except Error as e:
        message = f"Database Error: {e}"

    edit_id = form.getvalue('id') if action == 'edit' else None

    html_content = generate_page_content(conn, message, edit_id)
    print(html_content)

    if conn.is_connected():
        conn.close()

def get_db_connection():
    try:
        conn = mysql.connector.connect(**DB_CONFIG)
        return conn
    except Error:
        return None

def create_users_table(conn):
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS users (
            id INT AUTO_INCREMENT PRIMARY KEY,
            username VARCHAR(100) NOT NULL,
            email VARCHAR(100) NOT NULL UNIQUE
        )
    """)
    conn.commit()
    cursor.close()

def add_user(conn, username, email):
    cursor = conn.cursor()
    cursor.execute("INSERT INTO users (username, email) VALUES (%s, %s)", (username,
email))
    conn.commit()
    cursor.close()

def get_all_users(conn):
```

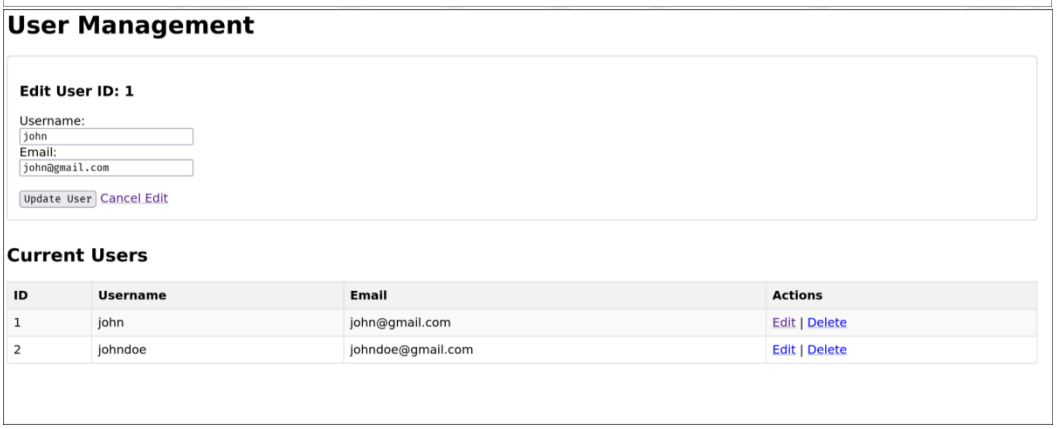
```
    cursor = conn.cursor()
    cursor.execute("SELECT id, username, email FROM users ORDER BY id")
    users = cursor.fetchall()
    cursor.close()
    return users

def get_user_by_id(conn, user_id):
    cursor = conn.cursor(dictionary=True)
    cursor.execute("SELECT id, username, email FROM users WHERE id = %s", (user_id,))
    user = cursor.fetchone()
    cursor.close()
    return user

def update_user(conn, user_id, username, email):
    cursor = conn.cursor()
    cursor.execute("UPDATE users SET username = %s, email = %s WHERE id = %s",
(username, email, user_id))
    conn.commit()
    cursor.close()

def delete_user(conn, user_id):
    cursor = conn.cursor()
    cursor.execute("DELETE FROM users WHERE id = %s", (user_id,))
    conn.commit()
    cursor.close()

def generate_page_content(conn, message, edit_id=None):
    user_to_edit = None
    if edit_id:
        user_to_edit = get_user_by_id(conn, edit_id)

    if user_to_edit:
        form_html = generate_edit_form(user_to_edit)
    else:
        form_html = generate_add_form()

    users_table_html = generate_users_table(conn)

    return f"""
    <html>
    <head>
        <title>Python CGI CRUD App</title>
        <style>
            body {{ font-family: sans-serif; margin: 2em; }}
            table {{ border-collapse: collapse; width: 100%; }}
            th, td {{ border: 1px solid #dddddd; text-align: left; padding: 8px; }}
            th {{ background-color: #f2f2f2; }}
            tr:nth-child(even) {{ background-color: #f9f9f9; }}
            .message {{ color: green; font-weight: bold; }}
            .error {{ color: red; }}
            form {{ margin-bottom: 2em; padding: 1em; border: 1px solid #ccc; border-radius:
5px; }}
        </style>
```

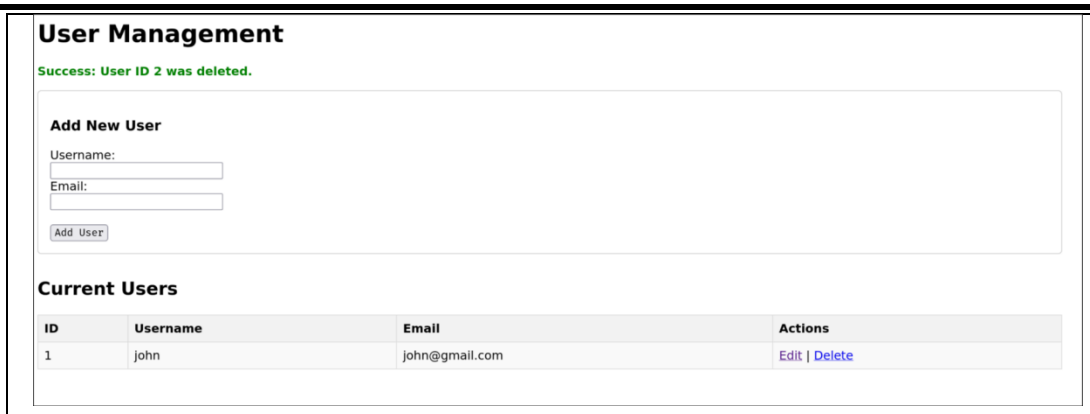
```
    </head>
    <body>
        <h1>User Management</h1>
        <p class="message">{message}</p>

        {form_html}

        <h2>Current Users</h2>
        {users_table_html}
    </body>
    </html>
    """

def generate_add_form():
    return f"""
    <form action="" method="post">
        <input type="hidden" name="action" value="add">
        <h3>Add New User</h3>
        <label>Username:</label><br>
        <input type="text" name="username" required><br>
        <label>Email:</label><br>
        <input type="email" name="email" required><br><br>
        <input type="submit" value="Add User">
    </form>
    """

def generate_edit_form(user):
    return f"""
    <form action="" method="post">
        <input type="hidden" name="action" value="update">
        <input type="hidden" name="id" value="{user['id']}">
        <h3>Edit User ID: {user['id']}</h3>
        <label>Username:</label><br>
        <input type="text" name="username" value="{user['username']}" required><br>
        <label>Email:</label><br>
        <input type="email" name="email" value="{user['email']}" required><br><br>
        <input type="submit" value="Update User">
        <a href="">Cancel Edit</a>
    </form>
    """

def generate_users_table(conn):
    users = get_all_users(conn)
    table_rows = ""
    for user in users:
        user_id, username, email = user
        edit_link = f'<a href="?action=edit&id={user_id}">Edit</a>'
        delete_link = f'<a href="?action=delete&id={user_id}" onclick="return confirm(\'Are
you sure?\');">Delete</a>'
        table_rows +=
f"<tr><td>{user_id}</td><td>{username}</td><td>{email}</td><td>{edit_link} |
{delete_link}</td></tr>"
```

```
    return f"""
    <table>
        <tr>
            <th>ID</th>
            <th>Username</th>
            <th>Email</th>
            <th>Actions</th>
        </tr>
        {table_rows}
    </table>
    """

if __name__ == "__main__":
    main()
```

Output Screenshot/Text






User Management
Success: User ID 2 was deleted.
Add New User
Username:
Email:
Add User
Current Users
ID
Username
Email
Actions
1
john
john@gmail.com
Edit | Delete

Department of Computer Science

DATA ANALYTICS USING PYTHON RECORD

Program No:30	Date: 12/09/2025
Name:	Reg No:
Program Title: Implementation of SqLite3 connection using Python.	

login.py

```
#!/usr/bin/python3

import cgi

print("Content-Type: text/html\n")

print("""
<html>
<head>
   <title>CGI Login Form</title>
   <style>
      body { font-family: sans-serif; margin: 2em; }
      form { padding: 1em; border: 1px solid #ccc; border-radius: 5px; width: 300px; }
      input { margin-bottom: 10px; width: 100%; padding: 8px; box-sizing: border-box; }
      input[type="submit"] { width: auto; cursor: pointer; }
   </style>
</head>
<body>
   <h2>User Login</h2>
   <form action="/cgi-bin/welcome.py" method="post">
      <label for="username">Username:</label><br>
      <input type="text" id="username" name="username" required><br>

      <label for="email">Email (as password):</label><br>
      <input type="email" id="email" name="email" required><br><br>

      <input type="submit" value="Login">
   </form>
</body>
</html>
""")
```

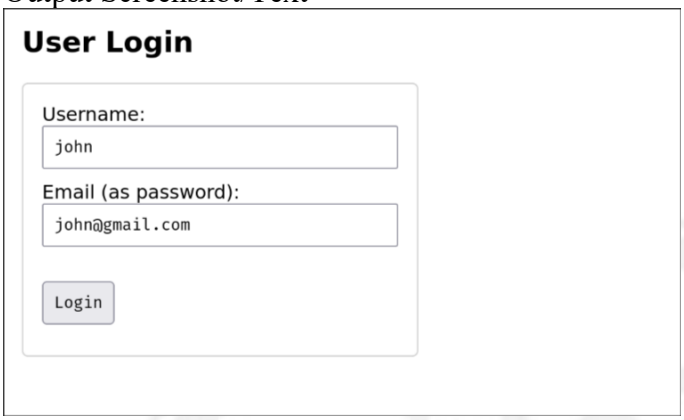
welcome.py

```
#!/usr/bin/python3
```

```
import cgi
import cgitb
cgitb.enable()

import html
import mysql.connector
from mysql.connector import Error

DB_CONFIG = {
    'host': 'localhost',
    'user': 'root',
    'password': '',
    'database': 'testdb'
}

def check_user_credentials(username, email):
    """Checks if a user with the given username and email exists."""
    try:
        conn = mysql.connector.connect(**DB_CONFIG)
        if conn.is_connected():
            cursor = conn.cursor()
            query = "SELECT * FROM users WHERE username = %s AND email = %s"
            cursor.execute(query, (username, email))

            result = cursor.fetchone()

            cursor.close()
            conn.close()

            return result is not None
    except Error as e:
        print(f"<h1>Database Connection Error</h1><p>{e}</p>")
        return False

print("Content-Type: text/html\n")
print("<html><head><title>Login Status</title></head><body>")

form = cgi.FieldStorage()
username = form.getvalue('username')
email = form.getvalue('email')

if username and email:
    if check_user_credentials(username, email):
        # Use html.escape() instead of the old cgi.escape()
        print(f"<h1>Welcome, {html.escape(username)}!</h1>")
        print("<p>You have successfully logged in.</p>")
    else:
        print("<h1>Login Failed</h1>")
        print("<p>Invalid username or email. Please try again.</p>")
else:
    print("<h1>Error</h1>")
    print("<p>Please provide both a username and an email.</p>")
```

```
print("<a href='/cgi-bin/login_form.py'>Back to Login</a>")
print("</body></html>")
```

Output Screenshot/Text



Department of Computer Science

DATA ANALYTICS USING PYTHON RECORD

Program No:31	Date: 12/09/2025
Name:	Reg No:
Program Title: Create a registration form for MCA admission and display the inserted data on the web page..	

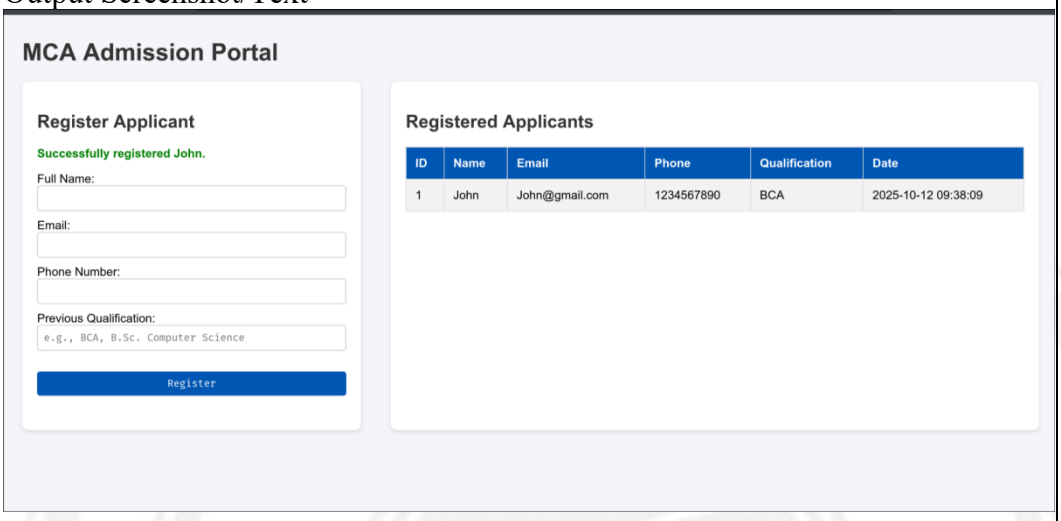
```
#!/usr/bin/python3

import cgi
import cgitb
import mysql.connector
import html
import os

cgitb.enable()

DB_CONFIG = {
    'host': 'localhost',
    'user': 'root',
    'password': '',
    'database': 'testdb'
}

def create_table(conn):
    """Creates the mca_admissions table if it does not exist."""
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS mca_admissions (
            id INT AUTO_INCREMENT PRIMARY KEY,
            full_name VARCHAR(100) NOT NULL,
            email VARCHAR(100) NOT NULL,
            phone VARCHAR(15),
            qualification VARCHAR(100),
            registration_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
        )
    """)
    conn.commit()
    cursor.close()

def insert_applicant(conn, form):
    """Inserts a new applicant's data into the database."""
    full_name = form.getvalue('full_name')
```

```
    email = form.getvalue('email')
    phone = form.getvalue('phone')
    qualification = form.getvalue('qualification')

    if full_name and email and phone and qualification:
        cursor = conn.cursor()
        query = "INSERT INTO mca_admissions (full_name, email, phone, qualification)
VALUES (%s, %s, %s, %s)"
        cursor.execute(query, (full_name, email, phone, qualification))
        conn.commit()
        cursor.close()
        return f"Successfully registered {html.escape(full_name)}."
    return ""

def get_all_applicants(conn):
    """Retrieves all applicant records from the database."""
    cursor = conn.cursor()
    cursor.execute("SELECT id, full_name, email, phone, qualification, registration_date
FROM mca_admissions ORDER BY id")
    records = cursor.fetchall()
    cursor.close()
    return records

# --- Main Script Logic ---

print("Content-Type: text/html\n")
print("<html><head><title>MCA Admission Registration</title>")
print("""
<style>
    body { font-family: Arial, sans-serif; margin: 2em; background-color: #f4f4f9; }
    h1, h2 { color: #333; }
    .container { display: flex; gap: 40px; }
    .form-section, .table-section { background-color: #fff; padding: 20px; border-radius: 8px;
box-shadow: 0 2px 5px rgba(0,0,0,0.1); }
    .form-section { flex: 1; }
    .table-section { flex: 2; }
    form input, form select { width: 100%; padding: 8px; margin-bottom: 10px; border-
radius: 4px; border: 1px solid #ccc; box-sizing: border-box; }
    form input[type="submit"] { background-color: #0056b3; color: white; cursor: pointer;
border: none; }
    .message { color: green; font-weight: bold; }
    table { width: 100%; border-collapse: collapse; margin-top: 20px; }
    th, td { padding: 12px; border: 1px solid #ddd; text-align: left; }
    th { background-color: #0056b3; color: white; }
    tr:nth-child(even) { background-color: #f2f2f2; }
</style>
""")
print("</head><body>")
print("<h1>MCA Admission Portal</h1>")

message = ""
connection = None
```

```
try:
    connection = mysql.connector.connect(**DB_CONFIG)
    create_table(connection)

    if os.environ['REQUEST_METHOD'] == 'POST':
        form = cgi.FieldStorage()
        message = insert_applicant(connection, form)

    applicants = get_all_applicants(connection)

    print('<div class="container">')

    # Registration Form Section
    print('<div class="form-section">')
    print("<h2>Register Applicant</h2>")
    if message:
        print(f'<p class="message">{message}</p>')
    print(f'<form action="{os.environ.get("SCRIPT_NAME", "")}" method="post">')
    print("""
            <label for="full_name">Full Name:</label><br>
            <input type="text" id="full_name" name="full_name" required><br>

            <label for="email">Email:</label><br>
            <input type="email" id="email" name="email" required><br>

            <label for="phone">Phone Number:</label><br>
            <input type="tel" id="phone" name="phone" required><br>

            <label for="qualification">Previous Qualification:</label><br>
            <input type="text" id="qualification" name="qualification" placeholder="e.g., BCA,
B.Sc. Computer Science" required><br><br>

            <input type="submit" value="Register">
        </form>
    </div>""")

    # Display Applicants Section
    print('<div class="table-section">')
    print("<h2>Registered Applicants</h2>")
    if applicants:

print("<table><tr><th>ID</th><th>Name</th><th>Email</th><th>Phone</th><th>Qualifi
cation</th><th>Date</th></tr>")
        for row in applicants:
            print("<tr>")
            for item in row:
                print(f"<td>{html.escape(str(item))}</td>")
            print("</tr>")
        print("</table>")
    else:
        print("<p>No applicants have registered yet.</p>")
    print("</div>") # end table-section
    print("</div>") # end container
```

```
except mysql.connector.Error as e:
    print(f"<h2>Database Error</h2><p>Could not connect or run query: {e}</p>")
finally:
    if connection and connection.is_connected():
        connection.close()

print("</body></html>")
```

Output Screenshot/Text



Department of Computer Science

DATA ANALYTICS USING PYTHON RECORD

Program No:32	Date: 1/09/2025
Name:	Reg No:

Program Title: Create a MySQL database and perform INSERT, UPDATE, DESTROY, and SELECT (display) operations using the CGI interface

```
#!/usr/bin/python3

import cgi
import cgitb
import mysql.connector
import html
import os

cgitb.enable()

DB_CONFIG = {
    'host': 'localhost',
    'user': 'root',
    'password': '',
    'database': 'product_db'
}

def get_db_connection():
    return mysql.connector.connect(**DB_CONFIG)

def initialize_database(conn):
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS products (
            id INT AUTO_INCREMENT PRIMARY KEY,
            name VARCHAR(100) NOT NULL,
            description TEXT,
            price DECIMAL(10, 2) NOT NULL,
            stock INT NOT NULL
        )
    """)
    conn.commit()
    cursor.close()

def get_all_products(conn):
    cursor = conn.cursor()
```

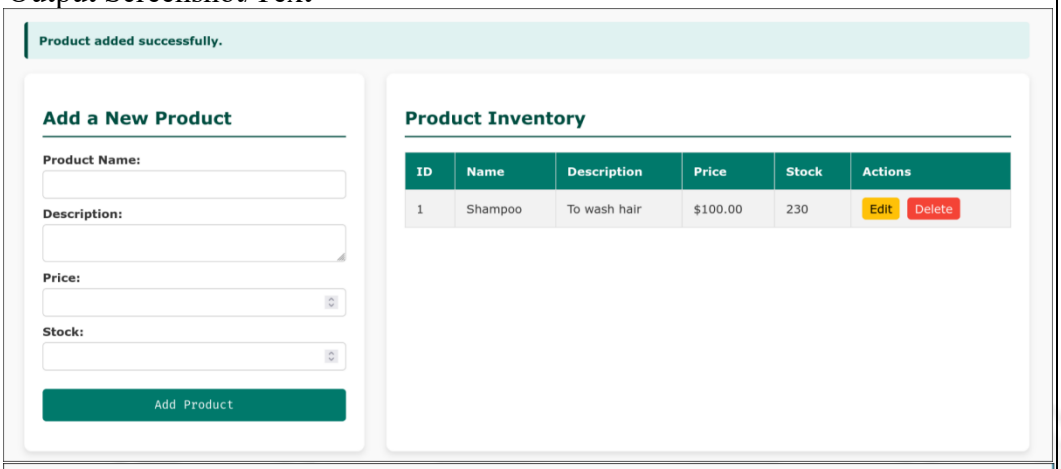
```
    cursor.execute("SELECT id, name, description, price, stock FROM products ORDER BY
id")
    products = cursor.fetchall()
    cursor.close()
    return products

def get_product_by_id(conn, product_id):
    cursor = conn.cursor()
    cursor.execute("SELECT id, name, description, price, stock FROM products WHERE id
= %s", (product_id,))
    product = cursor.fetchone()
    cursor.close()
    return product

def insert_product(conn, form):
    name = form.getvalue('name')
    desc = form.getvalue('description')
    price = form.getvalue('price')
    stock = form.getvalue('stock')

    if name and price and stock:
        cursor = conn.cursor()
        query = "INSERT INTO products (name, description, price, stock) VALUES (%s, %s,
%s, %s)"
        cursor.execute(query, (name, desc, price, stock))
        conn.commit()
        cursor.close()
        return "Product added successfully."
    return "Missing required fields for new product."

def update_product(conn, form):
    product_id = form.getvalue('id')
    name = form.getvalue('name')
    desc = form.getvalue('description')
    price = form.getvalue('price')
    stock = form.getvalue('stock')

    if product_id and name and price and stock:
        cursor = conn.cursor()
        query = "UPDATE products SET name = %s, description = %s, price = %s, stock = %s
WHERE id = %s"
        cursor.execute(query, (name, desc, price, stock, product_id))
        conn.commit()
        cursor.close()
        return "Product updated successfully."
    return "Missing required fields for update."

def delete_product(conn, product_id):
    if product_id:
        cursor = conn.cursor()
        cursor.execute("DELETE FROM products WHERE id = %s", (product_id,))
        conn.commit()
        cursor.close()
```

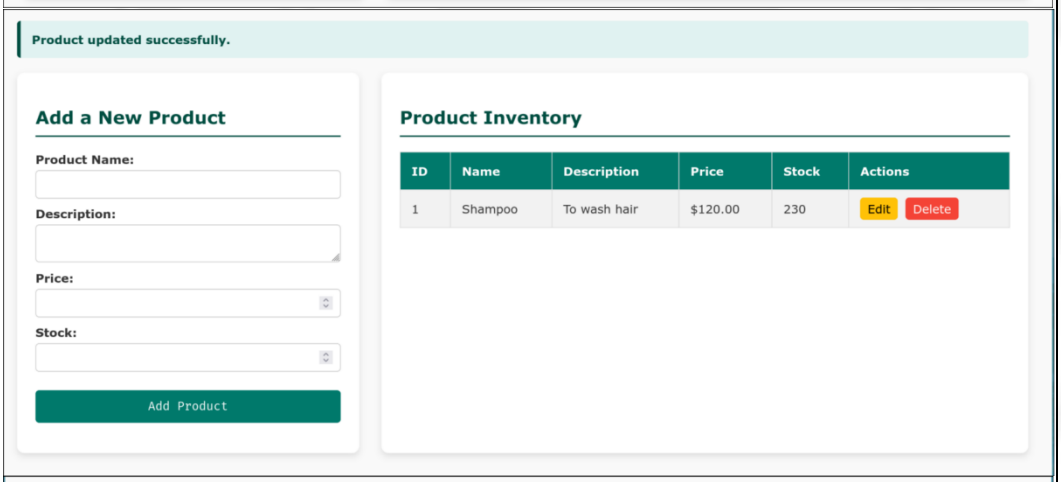
```
        return "Product deleted successfully."
    return "Product ID not provided for deletion."

def print_html_header(title):
    print("Content-Type: text/html\n")
    print(f"<html><head><title>{title}</title>")
    print("""
    <style>
        body { font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; margin: 2em;
background-color: #f9f9f9; color: #333; }
        h1, h2 { color: #004d40; border-bottom: 2px solid #004d40; padding-bottom: 10px; }
        .container { display: flex; gap: 30px; }
        .form-section, .table-section { background: #fff; padding: 25px; border-radius: 10px;
box-shadow: 0 4px 8px rgba(0,0,0,0.1); }
        .form-section { flex: 1; }
        .table-section { flex: 2; }
        form label { display: block; font-weight: bold; margin-top: 10px; }
        form input, form textarea { width: 100%; padding: 10px; margin-top: 5px; border-
radius: 5px; border: 1px solid #ccc; box-sizing: border-box; }
        form input[type="submit"] { background-color: #00796b; color: white; cursor: pointer;
border: none; padding: 12px 20px; font-size: 16px; transition: background-color 0.3s; }
        form input[type="submit"]:hover { background-color: #004d40; }
        .message { padding: 15px; border-radius: 5px; margin-bottom: 20px; font-weight: bold;
}
        .success { background-color: #e0f2f1; color: #004d40; border-left: 5px solid #004d40;
}
        .error { background-color: #ffebee; color: #c62828; border-left: 5px solid #c62828; }
        table { width: 100%; border-collapse: collapse; margin-top: 20px; }
        th, td { padding: 15px; border: 1px solid #ddd; text-align: left; }
        th { background-color: #00796b; color: white; }
        tr:nth-child(even) { background-color: #f2f2f2; }
        .actions a { text-decoration: none; padding: 5px 10px; border-radius: 5px; margin-right:
5px; }
        .edit-btn { background-color: #ffc107; color: black; }
        .delete-btn { background-color: #f44336; color: white; }
    </style>
    """)
    print("</head><body>")

def print_html_footer():
    print("</body></html>")

def print_product_form(product=None):
    is_edit = product is not None
    form_action = os.environ.get("SCRIPT_NAME", "")

    product_id = html.escape(str(product[0])) if is_edit else ""
    name = html.escape(str(product[1])) if is_edit else ""
    desc = html.escape(str(product[2])) if is_edit else ""
    price = html.escape(str(product[3])) if is_edit else ""
    stock = html.escape(str(product[4])) if is_edit else ""

    action = "update" if is_edit else "insert"
```

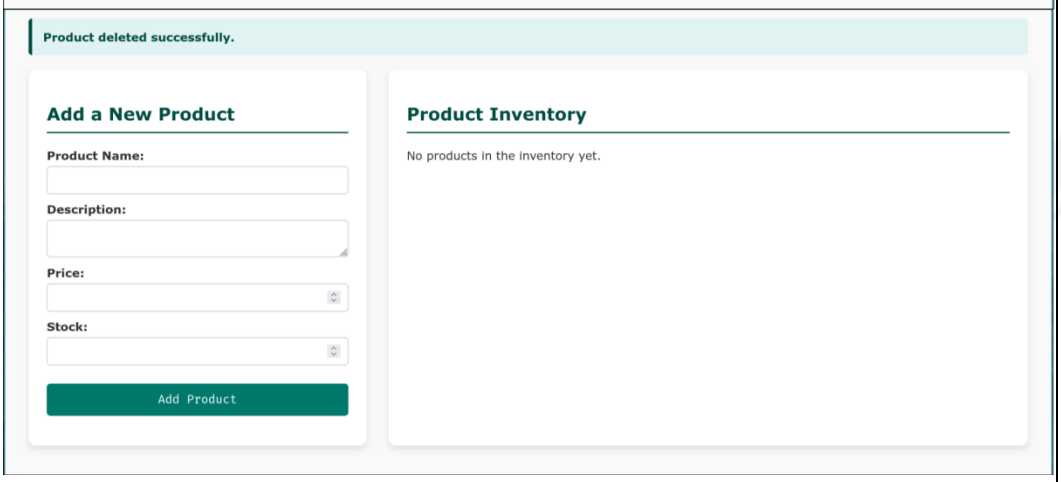
```
    button_text = "Update Product" if is_edit else "Add Product"

    print(f"<h2>{'Edit Product' if is_edit else 'Add a New Product'}</h2>")
    print(f'<form action="{form_action}" method="post">')
    print(f'<input type="hidden" name="action" value="{action}">')
    if is_edit:
        print(f'<input type="hidden" name="id" value="{product_id}">')

    print(f'<label for="name">Product Name:</label><input type="text" id="name"
name="name" value="{name}" required>')
    print(f'<label for="description">Description:</label><textarea id="description"
name="description">{desc}</textarea>')
    print(f'<label for="price">Price:</label><input type="number" step="0.01" id="price"
name="price" value="{price}" required>')
    print(f'<label for="stock">Stock:</label><input type="number" id="stock" name="stock"
value="{stock}" required>')
    print(f'<br><br><input type="submit" value="{button_text}">')
    print('</form>')

def print_products_table(products):
    print("<h2>Product Inventory</h2>")
    if not products:
        print("<p>No products in the inventory yet.</p>")
        return


print("<table><tr><th>ID</th><th>Name</th><th>Description</th><th>Price</th><th>Sto
ck</th><th>Actions</th></tr>")
    for row in products:
        row_escaped = [html.escape(str(item)) for item in row]
        edit_url = f'{os.environ.get("SCRIPT_NAME",
"")}?action=edit&id={row_escaped[0]}'
        delete_url = f'{os.environ.get("SCRIPT_NAME",
"")}?action=delete&id={row_escaped[0]}'
        print("<tr>")

print(f"<td>{row_escaped[0]}</td><td>{row_escaped[1]}</td><td>{row_escaped[2]}</td
><td>${row_escaped[3]}</td><td>{row_escaped[4]}</td>")
        print(f'<td class="actions"><a class="edit-btn" href="{edit_url}">Edit</a> <a
class="delete-btn" href="{delete_url}" onclick="return confirm(\'Are you sure you want to
delete this item?\');">Delete</a></td>')
        print("</tr>")
    print("</table>")

def main():
    form = cgi.FieldStorage()
    action = form.getvalue('action')
    message = ""

    conn = None
    try:
        conn = get_db_connection()
        initialize_database(conn)
```

```
        if os.environ['REQUEST_METHOD'] == 'POST':
            if action == 'insert':
                message = insert_product(conn, form)
            elif action == 'update':
                message = update_product(conn, form)

        elif action == 'delete':
            product_id = form.getvalue('id')
            message = delete_product(conn, product_id)

        product_to_edit = None
        if action == 'edit':
            product_id = form.getvalue('id')
            product_to_edit = get_product_by_id(conn, product_id)

        all_products = get_all_products(conn)

        print_html_header("Product Management System")
        if message:
            print(f'<div class="message success">{message}</div>')

        print('<div class="container">')
        print('<div class="form-section">')
        print_product_form(product_to_edit)
        print('</div>')

        print('<div class="table-section">')
        print_products_table(all_products)
        print('</div>')
        print('</div>')

    except mysql.connector.Error as e:
        print_html_header("Database Error")
        print(f'<div class="message error"><h2>A Database Error
Occurred</h2><p>{e}</p></div>')

    finally:
        if conn and conn.is_connected():
            conn.close()
        print_html_footer()

if __name__ == "__main__":
    main()
```

Output Screenshot/Text







Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 33	Date: 17/09/2025
Name:	Reg No:
Program Title : Create a numpy array filled with all ones by defining its shape.	

Program :import numpy as np

```
# Define the shape of the array (e.g., 3 rows, 4 columns)
shape = (3, 4)

# Create an array filled with ones
ones_array = np.ones(shape)

# Display the array
print("Array filled with ones:")
print(ones_array)
```

Output Screenshot/Text

```
Array filled with ones:
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 34	Date: 17/09/2025
Name:	Reg No:
Program Title :program to remove rows from a Numpy array that contains non-numeric values	

Program :

```
import numpy as np

# Take input for number of rows and columns
rows = int(input("Enter number of rows: "))
cols = int(input("Enter number of columns: "))

# Initialize an empty list to store the rows
data = []

print("Enter the array elements row by row (use numbers or 'nan' for non-numeric values):")

for i in range(rows):
    row_input = input(f"Row {i+1} (space-separated values): ").split()
    # Convert each element to float, convert 'nan' or invalid input to np.nan
    row = []
    for val in row_input:
        try:
            row.append(float(val))
        except ValueError:
            row.append(np.nan)
    data.append(row)

# Convert list to NumPy array
arr = np.array(data)

print("\nOriginal Array:")
print(arr)



# Remove rows containing non-numeric values (np.nan)
clean_arr = arr[~np.isnan(arr).any(axis=1)]
```

```
print("\nArray after removing rows with non-numeric values:")
print(clean_arr)
```

Output Screenshot/Text

```
Enter number of rows: 3
Enter number of columns: 3
Enter the array elements row by row (use numbers or 'nan' for non-numeric values):
Row 1 (space-separated values): 1 2 3
Row 2 (space-separated values): 1 n n
Row 3 (space-separated values): 2 3 4

Original Array:
[[ 1.  2.  3.]
 [ 1. nan nan]
 [ 2.  3.  4.]]

Array after removing rows with non-numeric values:
[[1. 2. 3.]
 [2. 3. 4.]]
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 35	Date: 17/09/2025
Name:	Reg No:

Program Title : Write a program remove single-dimensional entries from the shape of an array

Program :import numpy as np

```
# Create a sample array with single-dimensional entries
arr = np.array([[[1, 2, 3]]])  # Shape is (1, 1, 3)

print("Original array shape:", arr.shape)
print(arr)

# Remove single-dimensional entries
squeezed_arr = np.squeeze(arr)

print("\nArray after removing single-dimensional entries:")
print("New shape:", squeezed_arr.shape)
print(squeezed_arr)
```

Output Screenshot/Text

```
Original array shape: (1, 1, 3)
[[[1 2 3]]]

Array after removing single-dimensional entries:
New shape: (3,)
[1 2 3]
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 36	Date: 17/09/2025
Name:	Reg No:

Program Title : Write a program to check whether specified values are present in the NumPy array?

Program :
```
import numpy as np

# Create a sample NumPy array
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

# Take user input for values to check (space-separated)
values = input("Enter values to check (space-separated): ").split()
values = [int(v) for v in values]

# Check for presence using np.isin()
presence = np.isin(values, arr)

# Display results
for val, is_present in zip(values, presence):
    if is_present:
        print(f"{val} is present in the array.")
    else:
        print(f"{val} is NOT present in the array.")
```

Output Screenshot/Text

```
Enter values to check (space-separated): 6
6 is present in the array.
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 37	Date: 17/09/2025
Name:	Reg No:

Program Title : Write a NumPy program to sort a given array of shape 2 along the first axis, last axis, and f lattened array.

Program :
```
import numpy as np

# Take input for array dimensions
rows = int(input("Enter number of rows: "))
cols = int(input("Enter number of columns: "))

# Take input for array elements row by row
print("Enter array elements row by row (space-separated):")
data = []
for i in range(rows):
    row = list(map(float, input(f"Row {i+1}: ").split()))
    if len(row) != cols:
        print(f"Error: You must enter exactly {cols} values.")
        exit()
    data.append(row)

# Create NumPy array
arr = np.array(data)
print("\nOriginal Array:")
print(arr)

# 1. Sort along the first axis (axis=0)
sorted_first_axis = np.sort(arr, axis=0)
print("\nSorted along the first axis (axis=0):")
print(sorted_first_axis)

# 2. Sort along the last axis (axis=1)
sorted_last_axis = np.sort(arr, axis=1)
print("\nSorted along the last axis (axis=1):")
print(sorted_last_axis)

# 3. Flatten the array and sort
flattened_sorted = np.sort(arr, axis=None)
print("\nFlattened and sorted array:")
print(flattened_sorted)
```

Output Screenshot/Text

```
Enter number of rows: 3
Enter number of columns: 3
Enter array elements row by row (space-separated):
Row 1: 3 6 8
Row 2: 2 1 5
Row 3: 7 3 2

Original Array:
[[3. 6. 8.]
 [2. 1. 5.]
 [7. 3. 2.]]

Sorted along the first axis (axis=0):
[[2. 1. 2.]
 [3. 3. 5.]
 [7. 6. 8.]]

Sorted along the last axis (axis=1):
[[3. 6. 8.]
 [1. 2. 5.]
 [2. 3. 7.]]

Flattened and sorted array:
[1. 2. 2. 3. 3. 5. 6. 7. 8.]
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 38	Date: 17/09/2025
Name:	Reg No:

Program Title : Write a NumPy program to create a structured array from a given student name, height, class, and data type. Now sort by class, then height if the classes are equal.

Program : import numpy as np

```
# Define the data type for the structured array
student_dtype = np.dtype([
    ('name', 'U20'),    # Unicode string of max length 20
    ('height', 'f4'),   # Float
    ('class', 'i4')     # Integer
])

# Take input for number of students
n = int(input("Enter number of students: "))

# Initialize a list to store student data
students = []

# Take user input for each student
for i in range(n):
    print(f"\nEnter details for student {i+1}:")
    name = input("Name: ")
    height = float(input("Height: "))
    cls = int(input("Class: "))
    students.append((name, height, cls))

# Create structured NumPy array
student_arr = np.array(students, dtype=student_dtype)

print("\nOriginal Student Array:")
print(student_arr)

# Sort by class, then by height if classes are equal
sorted_students = np.sort(student_arr, order=['class', 'height'])

print("\nSorted Student Array (by class, then height):")
print(sorted_students)
```

Output Screenshot/Text

```
Enter number of students: 3

Enter details for student 1:
Name: student1
Height: 9
Class: 3

Enter details for student 2:
Name: student2
Height: 56
Class: 3

Enter details for student 3:
Name: studemt45
Height: 345
Class: 6

Original Student Array:
[('student1',   9., 3) ('student2',  56., 3) ('studemt45', 345., 6)]

Sorted Student Array (by class, then height):
[('student1',   9., 3) ('student2',  56., 3) ('studemt45', 345., 6)]
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 39	Date: 17/09/2025
Name:	Reg No:

Program Title : Write a NumPy program to sort a given complex array using the real part first, then the imaginary part.

Program :

```
 import numpy as np

n = int(input("Enter number of complex numbers: "))

arr = []

print("Enter complex numbers in the form a+bj (e.g., 3+4j):")

for i in range(n):
    num_str = input(f"Number {i+1}: ")
    try:
        arr.append(complex(num_str))
    except ValueError:
        print("Invalid format! Please enter in the form a+bj.")
        exit()

# Convert to NumPy array
arr = np.array(arr)


print("\nOriginal Array:")
print(arr)


# Sort by real part first, then imaginary part
sorted_arr = arr[np.lexsort((arr.imag, arr.real))]


print("\nSorted Array (by real part, then imaginary part):")
print(sorted_arr)
```

Output Screenshot/Text

```
Enter number of complex numbers: 4
Enter complex numbers in the form a+bj (e.g., 3+4j):
Number 1: 1+3j
Number 2: 2+1j
Number 3: 3+5j
Number 4: 3+6j

Original Array:
[1.+3.j 2.+1.j 3.+5.j 3.+6.j]

Sorted Array (by real part, then imaginary part):
[1.+3.j 2.+1.j 3.+5.j 3.+6.j]
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 40	Date: 17/09/2025
Name:	Reg No:

Program Title : Write a NumPy program to sort a given array by the nth column.

Program :
```
import numpy as np

# Take input for array dimensions
rows = int(input("Enter number of rows: "))
cols = int(input("Enter number of columns: "))

# Take input for array elements row by row
print("Enter array elements row by row (space-separated):")
data = []
for i in range(rows):
    row = list(map(float, input(f"Row {i+1}: ").split()))
    if len(row) != cols:
        print(f"Error: You must enter exactly {cols} values.")
        exit()
    data.append(row)

# Create NumPy array
arr = np.array(data)
print("\nOriginal Array:")
print(arr)

# Take input for column to sort by
nth_col = int(input(f"\nEnter the column index (0 to {cols-1}) to sort by: "))
if nth_col < 0 or nth_col >= cols:
    print("Invalid column index!")
    exit()

# Sort array by the nth column
sorted_arr = arr[arr[:, nth_col].argsort()]

print(f"\nArray sorted by column {nth_col}:")
print(sorted_arr)
```

Output Screenshot/Text

```
Enter number of rows: 4
Enter number of columns: 4
Enter array elements row by row (space-separated):
Row 1: 5 3 6 1
Row 2: 1 8 3 2
Row 3: 9 2 6 3
Row 4: 2 5 1 3

Original Array:
[[5. 3. 6. 1.]
 [1. 8. 3. 2.]
 [9. 2. 6. 3.]
 [2. 5. 1. 3.]]

Enter the column index (0 to 3) to sort by: 2

Array sorted by column 2:
[[2. 5. 1. 3.]
 [1. 8. 3. 2.]
 [5. 3. 6. 1.]
 [9. 2. 6. 3.]]
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 41	Date: 17/09/2025
Name:	Reg No:
Program Title : Calculate the sum of the diagonal elements of a NumPy array	

Program :
```
import numpy as np

n = int(input("Enter the size of the square matrix: "))

print("Enter the matrix elements row by row (space-separated):")

data = []

for i in range(n):

    row = list(map(float, input(f"Row {i+1}: ").split()))

    if len(row) != n:

        print(f"Error: You must enter exactly {n} values.")
        exit()

    data.append(row)

# Create NumPy array
arr = np.array(data)
print("\nMatrix:")
print(arr)

# Method 1: Using np.trace()

diag_sum = np.trace(arr)

print(f"\nSum of diagonal elements (using np.trace): {diag_sum}")

# Method 2: Using np.diagonal()

diag_sum2 = np.sum(arr.diagonal())

print(f"Sum of diagonal elements (using np.diagonal): {diag_sum2}")
```

Output Screenshot/Text

```
Enter the size of the square matrix: 3
Enter the matrix elements row by row (space-separated):
Row 1: 1 2 3
Row 2: 1 2 3
Row 3: 1 2 3

Matrix:
[[1. 2. 3.]
 [1. 2. 3.]
 [1. 2. 3.]]

Sum of diagonal elements (using np.trace): 6.0
Sum of diagonal elements (using np.diagonal): 6.0
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 42	Date: 17/09/2025
Name:	Reg No:
Program Title : Write a program for Matrix Multiplication in NumPy	

Program :
```
import numpy as np

# Take input for the first matrix
rows1 = int(input("Enter number of rows for Matrix 1: "))
cols1 = int(input("Enter number of columns for Matrix 1: "))

print("Enter elements of Matrix 1 row by row (space-separated):")
matrix1 = []
for i in range(rows1):
    row = list(map(float, input(f"Row {i+1}: ").split()))
    if len(row) != cols1:
        print(f"Error: You must enter exactly {cols1} values.")
        exit()
    matrix1.append(row)
matrix1 = np.array(matrix1)

# Take input for the second matrix
rows2 = int(input("\nEnter number of rows for Matrix 2: "))
cols2 = int(input("Enter number of columns for Matrix 2: "))

# Check if multiplication is possible
if cols1 != rows2:
    print("Error: Number of columns in Matrix 1 must equal number of rows in Matrix 2.")
    exit()

print("Enter elements of Matrix 2 row by row (space-separated):")
matrix2 = []
for i in range(rows2):
    row = list(map(float, input(f"Row {i+1}: ").split()))
    if len(row) != cols2:
        print(f"Error: You must enter exactly {cols2} values.")
        exit()
    matrix2.append(row)
matrix2 = np.array(matrix2)

# Perform matrix multiplication
```

```
result = np.dot(matrix1, matrix2)

# Display results
print("\nMatrix 1:")
print(matrix1)
print("\nMatrix 2:")
print(matrix2)
print("\nResult of Matrix Multiplication:")
print(result)
```

Output Screenshot/Text

```
Enter number of rows for Matrix 1: 2
Enter number of columns for Matrix 1: 2
Enter elements of Matrix 1 row by row (space-separated):
Row 1: 1 2
Row 2: 1 2

Enter number of rows for Matrix 2: 2
Enter number of columns for Matrix 2: 2
Enter elements of Matrix 2 row by row (space-separated):
Row 1: 1 2
Row 2: 1 2

Matrix 1:
[[1. 2.]
 [1. 2.]]

Matrix 2:
[[1. 2.]
 [1. 2.]]

Result of Matrix Multiplication:
[[3. 6.]
 [3. 6.]]
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 43	Date: 17/09/2025
Name:	Reg No:
Program Title : Multiply matrices of complex numbers using NumPy in Python	

Program :
```
import numpy as np

# Take input for the first matrix
rows1 = int(input("Enter number of rows for Matrix 1: "))
cols1 = int(input("Enter number of columns for Matrix 1: "))

print("Enter elements of Matrix 1 row by row (complex numbers a+bj):")
matrix1 = []
for i in range(rows1):
    row = []
    elems = input(f"Row {i+1}: ").split()
    if len(elems) != cols1:
        print(f"Error: You must enter exactly {cols1} values.")
        exit()
    for val in elems:
        try:
            row.append(complex(val))
        except ValueError:
            print("Invalid complex number format!")
            exit()
    matrix1.append(row)
matrix1 = np.array(matrix1)

# Take input for the second matrix
rows2 = int(input("\nEnter number of rows for Matrix 2: "))
cols2 = int(input("Enter number of columns for Matrix 2: "))

if cols1 != rows2:
    print("Error: Number of columns in Matrix 1 must equal number of rows in Matrix 2.")
    exit()

print("Enter elements of Matrix 2 row by row (complex numbers a+bj):")
matrix2 = []
for i in range(rows2):
    row = []
```

```
    elems = input(f"Row {i+1}: ").split()

    if len(elems) != cols2:
        print(f"Error: You must enter exactly {cols2} values.")
        exit()
    for val in elems:
        try:
            row.append(complex(val))
        except ValueError:
            print("Invalid complex number format!")
            exit()
    matrix2.append(row)
matrix2 = np.array(matrix2)

# Perform matrix multiplication
result = np.dot(matrix1, matrix2)

# Display results
print("\nMatrix 1:")
print(matrix1)
print("\nMatrix 2:")
print(matrix2)
print("\nResult of Matrix Multiplication:")
print(result)
```

Output Screenshot/Text

```
Enter number of rows for Matrix 1: 2
Enter number of columns for Matrix 1: 2
Enter elements of Matrix 1 row by row (complex numbers a+bj):
Row 1: 1 2
Row 2: 1 2

Enter number of rows for Matrix 2: 2
Enter number of columns for Matrix 2: 2
Enter elements of Matrix 2 row by row (complex numbers a+bj):
Row 1: 1 2
Row 2: 1 2

Matrix 1:
[[1.+0.j 2.+0.j]
 [1.+0.j 2.+0.j]]

Matrix 2:
[[1.+0.j 2.+0.j]
 [1.+0.j 2.+0.j]]

Result of Matrix Multiplication:
[[3.+0.j 6.+0.j]
 [3.+0.j 6.+0.j]]
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 44	Date: 17/09/2025
Name:	Reg No:

Program Title : Compute the covariance matrix of two given NumPy arrays.

Program :
```
import numpy as np

# Take input for two arrays
arr1 = list(map(float, input("Enter elements of first array (space-separated): ").split()))
arr2 = list(map(float, input("Enter elements of second array (space-separated): ").split()))

# Convert lists to NumPy arrays
x = np.array(arr1)
y = np.array(arr2)

# Check if both arrays have the same length
if len(x) != len(y):
    print("Error: Both arrays must have the same number of elements.")
    exit()

# Compute covariance matrix
cov_matrix = np.cov(x, y)

print("\nCovariance Matrix:")
print(cov_matrix)
```

Output Screenshot/Text

```
Enter elements of first array (space-separated): 1 3 2 6 7 4
Enter elements of second array (space-separated): 1 2 3 4 5 5

Covariance Matrix:
[[5.36666667 3.06666667]
 [3.06666667 2.66666667]]
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 45	Date: 17/09/2025
Name:	Reg No:

Program Title : Convert covariance matrix to correlation matrix using Python

Program :
```
import numpy as np

# Take input for two arrays
arr1 = list(map(float, input("Enter elements of first array (space-separated): ").split()))
arr2 = list(map(float, input("Enter elements of second array (space-separated): ").split()))

# Convert lists to NumPy arrays
x = np.array(arr1)
y = np.array(arr2)

# Check if both arrays have the same length
if len(x) != len(y):
    print("Error: Both arrays must have the same number of elements.")
    exit()
    cov_matrix = np.cov(x, y)

print("\nCovariance Matrix:")
print(cov_matrix)
std_x = np.std(x, ddof=1)  # ddof=1 for sample standard deviation
std_y = np.std(y, ddof=1)
correlation_matrix = cov_matrix / np.outer([std_x, std_y], [std_x, std_y])
print("\nCorrelation Matrix:")
print(correlation_matrix)
```

Output Screenshot/Text

```
Enter elements of first array (space-separated): 1 2 3 4 56
Enter elements of second array (space-separated): 1 2 3 4 52

Covariance Matrix:
[[573.7 530.9]
 [530.9 491.3]]

Correlation Matrix:
[[1.        0.9999929]
 [0.9999929 1.       ]]
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 46	Date: 17/09/2025
Name:	Reg No:

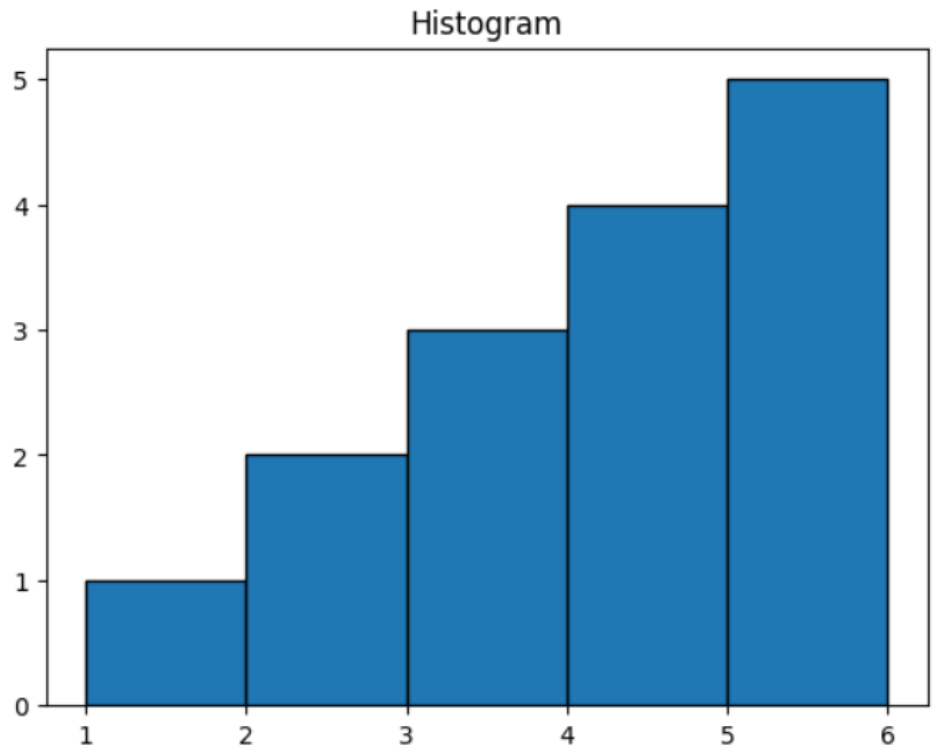
Program Title : Write a NumPy program to compute the histogram of nums against the bins.

Program :

```
import numpy as np
import matplotlib.pyplot as plt
nums = np.array([1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5])
bins = [1, 2, 3, 4, 5, 6]
hist, bin_edges = np.histogram(nums, bins=bins)
print("Histogram:", hist)
print("Bin edges:", bin_edges)
plt.hist(nums, bins=bins, edgecolor='black')
plt.title("Histogram")
plt.show()
```

Output Screenshot/Text

```
Histogram: [1 2 3 4 5]
Bin edges: [1 2 3 4 5 6]
```



Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 47	Date: 17/09/2025
Name:	Reg No:

Program Title : Write a NumPy program to compute the cross-correlation of two given arrays

Program :
```
import numpy as np

# Take input for the first array
arr1 = list(map(float, input("Enter elements of first array (space-separated): ").split()))
# Take input for the second array
arr2 = list(map(float, input("Enter elements of second array (space-separated): ").split()))

# Convert to NumPy arrays
x = np.array(arr1)
y = np.array(arr2)

# Compute cross-correlation
cross_corr = np.correlate(x, y, mode='full')

# Display results
print("\nCross-correlation:")
print(cross_corr)
```

Output Screenshot/Text

```
Enter elements of first array (space-separated): 1 2 3
Enter elements of second array (space-separated): 0 1 0.5


Cross-correlation:
[0.5 2.  3.5 3.  0. ]
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 48	Date: 17/09/2025
Name:	Reg No:

Program Title : Write a NumPy program to compute the mean, standard deviation, and variance of a given array along the second axis.

Program :
```
import numpy as np

arr = np.array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])

mean = np.mean(arr, axis=1)
std_dev = np.std(arr, axis=1)
variance = np.var(arr, axis=1)

print("Mean along second axis:", mean)
print("Standard Deviation along second axis:", std_dev)
print("Variance along second axis:", variance)
```

Output Screenshot/Text

```
Mean along second axis: [2. 5. 8.]
Standard Deviation along second axis: [0.81649658 0.81649658 0.81649658]
Variance along second axis: [0.66666667 0.66666667 0.66666667]
```

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 49	Date: 12/09/2025
Name:	Reg No:

Program Title : Visualize the following using the given dataset-

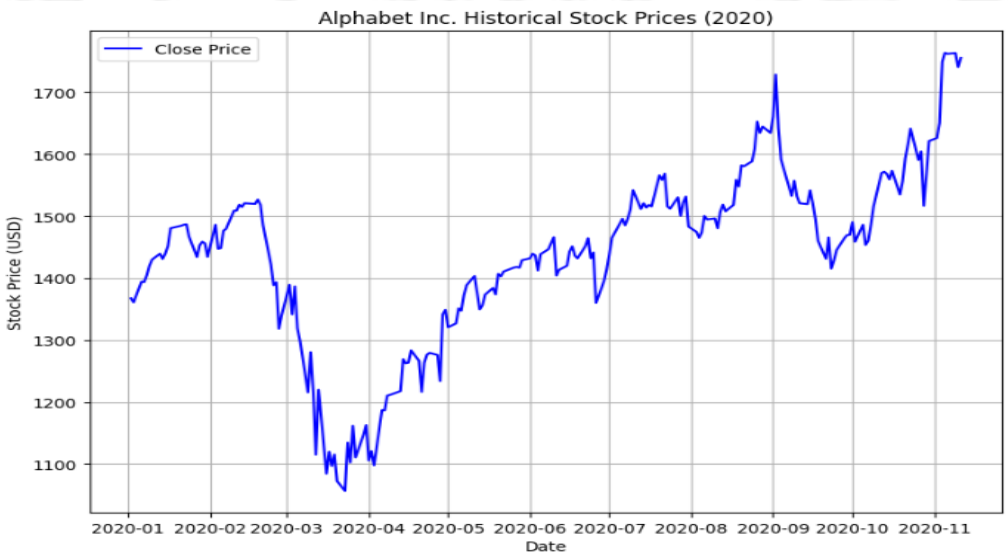
Create a line plot of the historical stock prices of Alphabet Inc. between two specific dates.
Create a bar plot of the trading volume of Alphabet Inc. stock between two specific dates.
Create a stacked histogram plot with more bins of opening, closing, high, and low stock prices of Alphabet Inc. between two specific dates.
Create a scatter plot of the trading volume/stock prices of Alphabet Inc. stock between two specific dates.

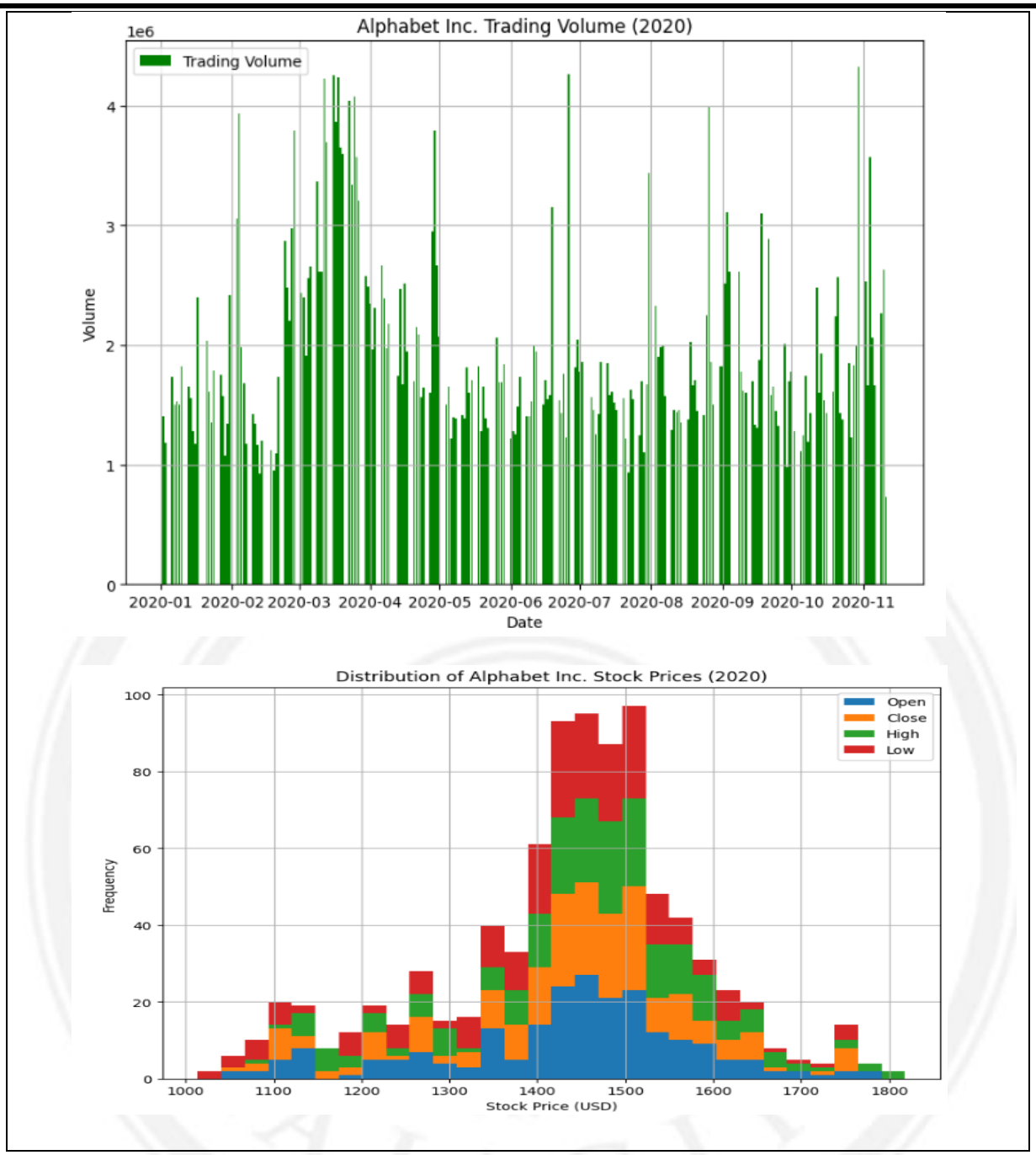
Program :

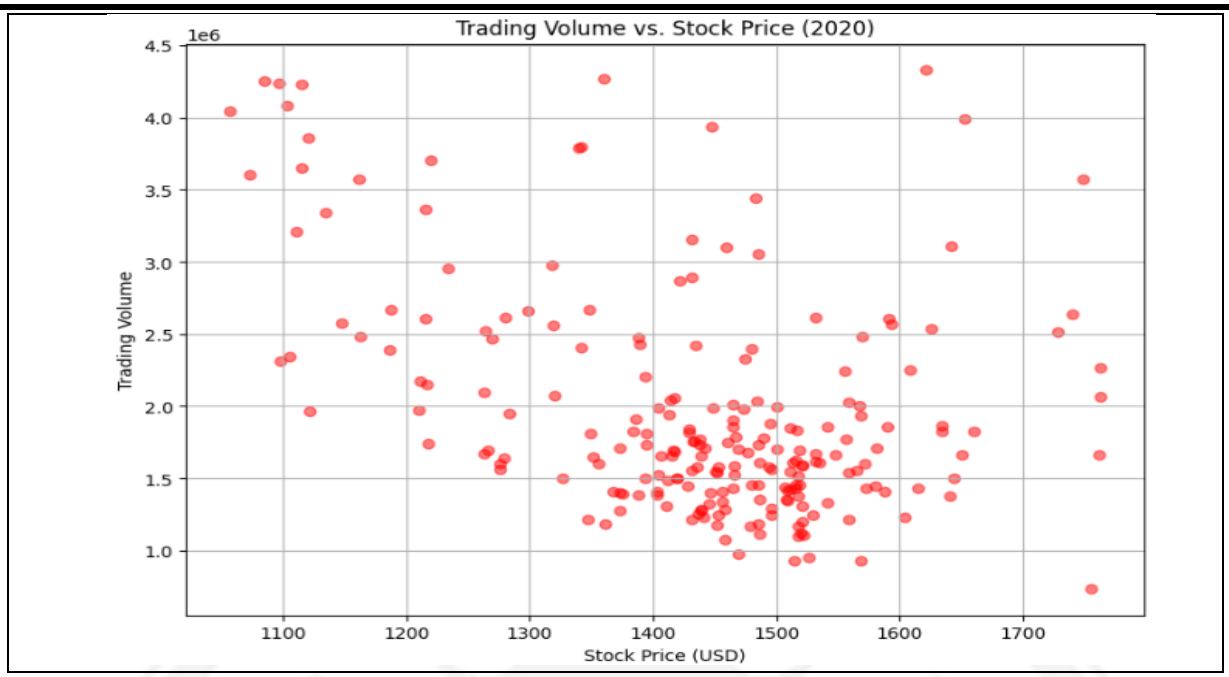
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('alphabet_stocks.csv')

df['Date'] = pd.to_datetime(df['Date'])

start_date = '2020-01-01'
end_date = '2020-12-31'
filtered_df = df[(df['Date'] >= start_date) & (df['Date'] <= end_date)]


plt.figure(figsize=(10, 6))
plt.plot(filtered_df['Date'], filtered_df['Close'], label='Close Price')
plt.title('Alphabet Inc. Historical Stock Prices (2020)')
plt.xlabel('Date')
plt.ylabel('Stock Price (USD)')
plt.legend()
plt.grid(True)
plt.savefig('stock_prices.png')
plt.show()



plt.figure(figsize=(10, 6))
plt.bar(filtered_df['Date'], filtered_df['Volume'], label='Trading Volume')
```

```
plt.title('Alphabet Inc. Trading Volume (2020)')
plt.xlabel('Date')
plt.ylabel('Volume')
plt.legend()
plt.grid(True)
plt.savefig('trading_volume.png')
plt.show()

plt.figure(figsize=(10, 6))
plt.hist([filtered_df['Open'], filtered_df['Close'], filtered_df['High'], filtered_df['Low']],
        bins=30, stacked=True, label=['Open', 'Close', 'High', 'Low'])
plt.title('Distribution of Alphabet Inc. Stock Prices (2020)')
plt.xlabel('Stock Price (USD)')
plt.ylabel('Frequency')
plt.legend()
plt.grid(True)
plt.savefig('stock_price_distribution.png')
plt.show()

plt.figure(figsize=(10, 6))
plt.scatter(filtered_df['Close'], filtered_df['Volume'], alpha=0.5)
plt.title('Alphabet Inc. Trading Volume vs. Stock Price (2020)')
plt.xlabel('Stock Price (USD)')
plt.ylabel('Trading Volume')
plt.grid(True)
plt.savefig('volume_vs_price.png')
plt.show()
```

Output Screenshot/Text




Alphabet Inc. Trading Volume (2020)
1e6
Trading Volume
Volume
0
1
2
3
4
2020-01 2020-02 2020-03 2020-04 2020-05 2020-06 2020-07 2020-08 2020-09 2020-10 2020-11
Date
Distribution of Alphabet Inc. Stock Prices (2020)
Open
Close
High
Low
Frequency
0
20
40
60
80
100
1000 1100 1200 1300 1400 1500 1600 1700 1800
Stock Price (USD)


Trading Volume vs. Stock Price (2020)
1e6
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Trading Volume
1100
1200
1300
1400
1500
1600
1700
Stock Price (USD)

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 50	Date: 12/09/2025
Name:	Reg No:

Program Title : Handle the given datasets with adequate preprocessing steps mentioned and visualize the dataset with appropriate graphs.
Handle Missing Data Values
Encode the categorical data
Scale your features
Normalize the data (if necessary)

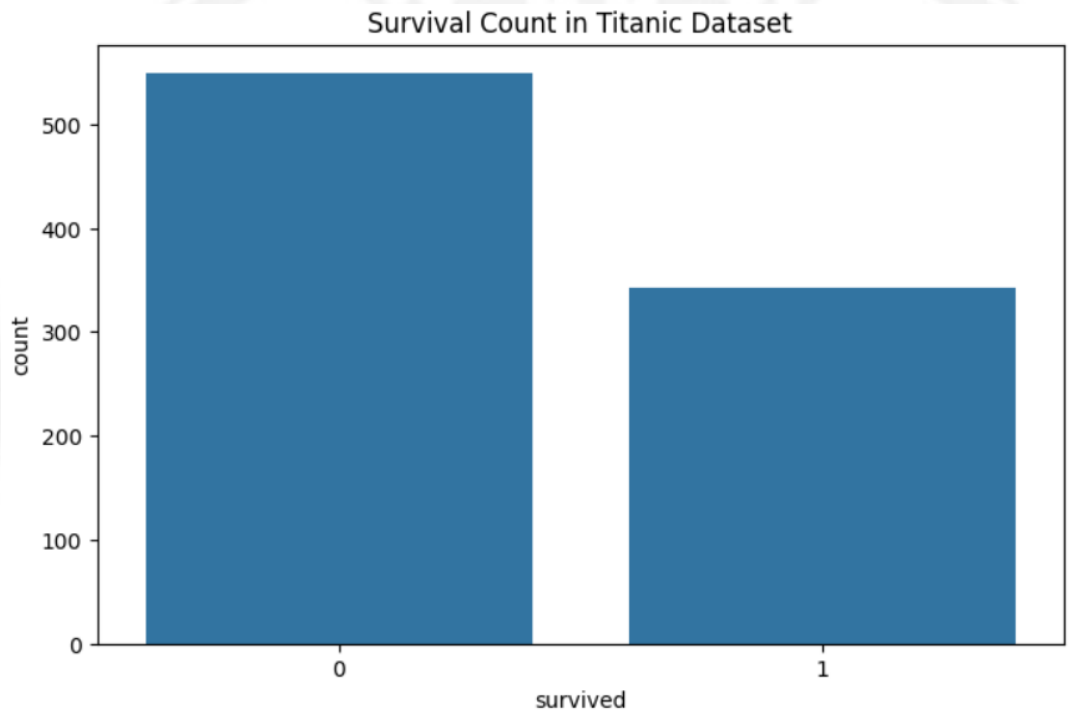
Program :

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder, StandardScaler, normalize
from sklearn.datasets import load_diabetes, load_breast_cancer

# 1. Titanic Dataset
print("\n--- Titanic Dataset ---")
titanic = sns.load_dataset("titanic")

print("Original Titanic shape:", titanic.shape)
print(titanic.head())

titanic['age'] = titanic['age'].fillna(titanic['age'].median())
titanic['embarked'] = titanic['embarked'].fillna(titanic['embarked'].mode()[0])

if 'deck' in titanic.columns:
    titanic = titanic.drop(columns=['deck'])

label_enc = LabelEncoder()
titanic['sex'] = label_enc.fit_transform(titanic['sex'])
titanic['embarked'] = label_enc.fit_transform(titanic['embarked'])
titanic['class'] = label_enc.fit_transform(titanic['class'])

scaler = StandardScaler()
numeric_cols = titanic.select_dtypes(include=[np.number]).columns

titanic[numeric_cols] = scaler.fit_transform(titanic[numeric_cols])

titanic[numeric_cols] = normalize(titanic[numeric_cols])
```

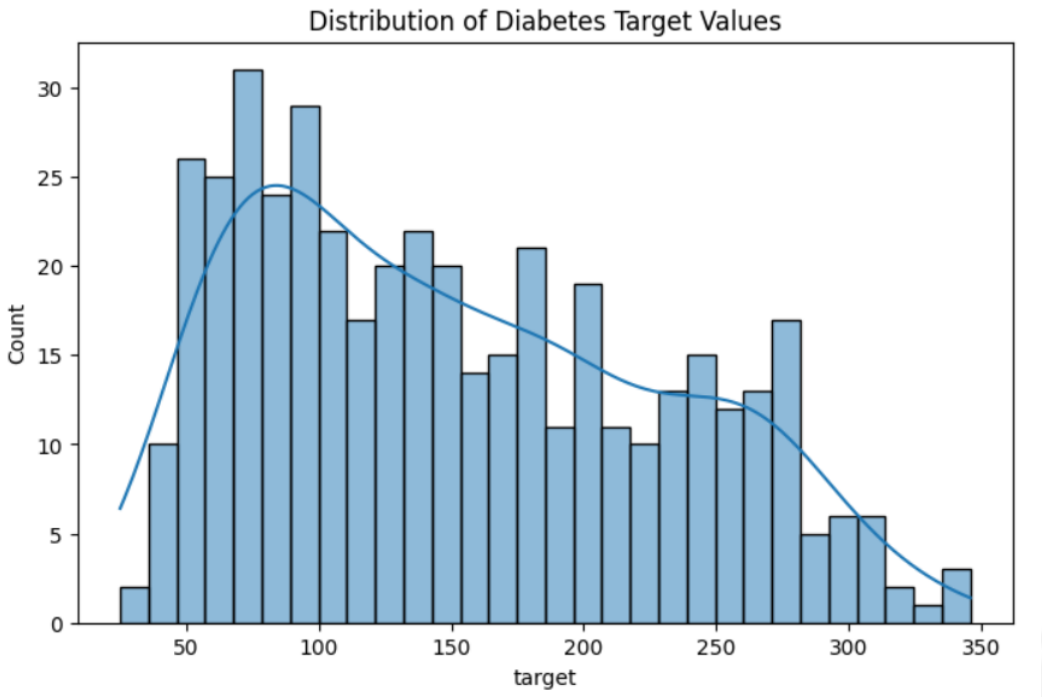
```
plt.figure(figsize=(8,5))
sns.countplot(x="survived", data=sns.load_dataset("titanic"))
plt.title("Survival Count in Titanic Dataset")
plt.show()

# 2. Diabetes Dataset
print("\n--- Diabetes Dataset ---")
diabetes = load_diabetes(as_frame=True)
df_diabetes = diabetes.frame

print("Original Diabetes shape:", df_diabetes.shape)
print(df_diabetes.head())

print("Missing values in Diabetes:", df_diabetes.isnull().sum().sum())

scaler = StandardScaler()
X_diabetes = scaler.fit_transform(df_diabetes.drop(columns=["target"]))

X_diabetes_norm = normalize(X_diabetes)

plt.figure(figsize=(8,5))
sns.histplot(df_diabetes["target"], bins=30, kde=True)
plt.title("Distribution of Diabetes Target Values")
plt.show()

# 3. Breast Cancer Dataset
print("\n--- Breast Cancer Dataset ---")
cancer = load_breast_cancer(as_frame=True)
df_cancer = cancer.frame

print("Original Cancer shape:", df_cancer.shape)
print(df_cancer.head())

print("Missing values in Cancer:", df_cancer.isnull().sum().sum())

scaler = StandardScaler()
X_cancer = scaler.fit_transform(df_cancer.drop(columns=["target"]))

X_cancer_norm = normalize(X_cancer)

plt.figure(figsize=(8,5))
sns.countplot(x=df_cancer["target"])
plt.title("Breast Cancer Diagnosis (0 = Malignant, 1 = Benign)")
plt.show()
```

Output Screenshot/Text

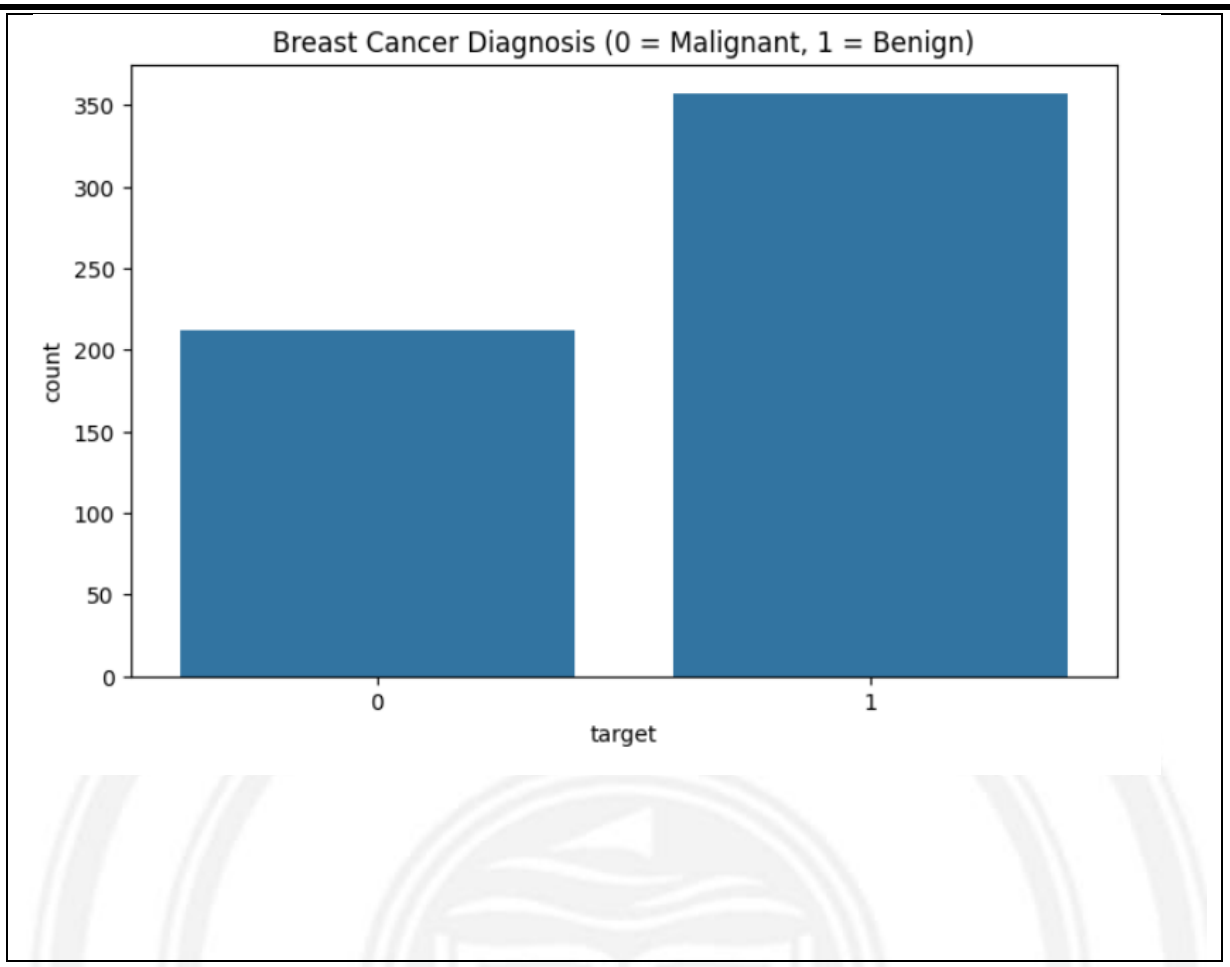
```
--- Titanic Dataset ---
Original Titanic shape: (891, 15)
   survived  pclass     sex   age  sibsp  parch     fare embarked  class
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```




```
--- Diabetes Dataset ---
Original Diabetes shape: (442, 11)
        age       sex       bmi        bp        s1        s2        s3  \
0  0.038076  0.050680  0.061696  0.021872 -0.044223 -0.034821 -0.043401
1 -0.001882 -0.044642 -0.051474 -0.026328 -0.008449 -0.019163  0.074412
2  0.085299  0.050680  0.044451 -0.005670 -0.045599 -0.034194 -0.032356
3 -0.089063 -0.044642 -0.011595 -0.036656  0.012191  0.024991 -0.036038
4  0.005383 -0.044642 -0.036385  0.021872  0.003935  0.015596  0.008142

         s4        s5        s6  target
0 -0.002592  0.019907 -0.017646   151.0
1 -0.039493 -0.068332 -0.092204    75.0
2 -0.002592  0.002861 -0.025930   141.0
3  0.034309  0.022688 -0.009362   206.0
4 -0.002592 -0.031988 -0.046641   135.0
Missing values in Diabetes: 0
```




```
--- Breast Cancer Dataset ---
Original Cancer shape: (569, 31)
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   mean fractal dimension  ...  worst texture  worst perimeter  worst area  \
0                 0.07871  ...          17.33           184.60      2019.0
1                 0.05667  ...          23.41           158.80      1956.0
2                 0.05999  ...          25.53           152.50      1709.0
3                 0.09744  ...          26.50            98.87       567.7
4                 0.05883  ...          16.67           152.20      1575.0

   worst smoothness  worst compactness  worst concavity  worst concave points  \
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430
3            0.2098             0.8663           0.6869                0.2575
4            0.1374             0.2050           0.4000                0.1625

   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890       0
1          0.2750                  0.08902       0
2          0.3613                  0.08758       0
3          0.6638                  0.17300       0
4          0.2364                  0.07678       0

[5 rows x 31 columns]
Missing values in Cancer: 0
```


Breast Cancer Diagnosis (0 = Malignant, 1 = Benign)
count
350
300
250
200
150
100
50
0
0
1
target

Department of Computer Science
DATA ANALYTICS USING PYTHON LAB RECORD

Program No: 51	Date: 12/09/2025
Name:	Reg No:

Program Title : Evaluate the dataset (User_Data.csv) and predict whether a user will purchase the company's product or not.
Compare the performance of any 3 classification models.

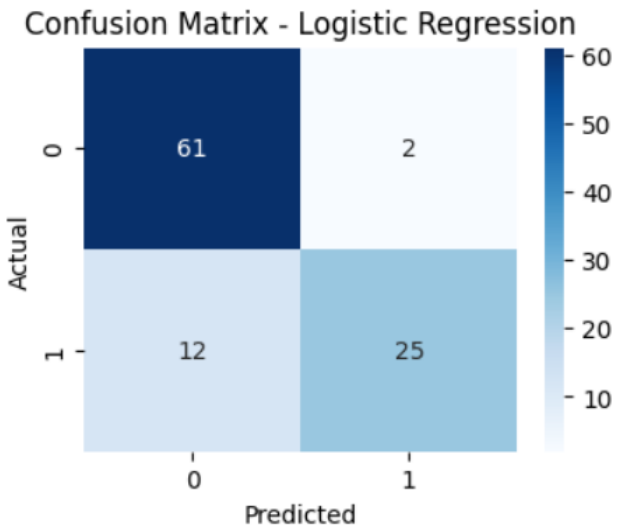
Program :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

df = pd.read_csv("User_Data.csv")

X = df[['Age', 'EstimatedSalary']]
y = df['Purchased']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42
)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

models = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42, n_estimators=100)
}

results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
```

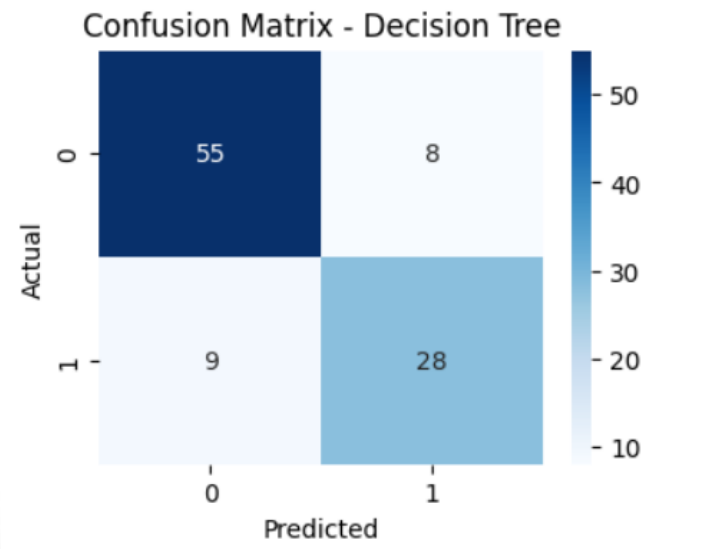
```
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, output_dict=True)
    results[name] = {
        "Accuracy": acc,
        "Precision": report['1']['precision'],
        "Recall": report['1']['recall'],
        "F1-Score": report['1']['f1-score']
    }
    print(f"\nModel: {name}")
    print("Accuracy:", acc)
    print(classification_report(y_test, y_pred))

    plt.figure(figsize=(4,3))
    sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap="Blues")
    plt.title(f"Confusion Matrix - {name}")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()
results_df = pd.DataFrame(results).T
print("\nComparison of Models:\n")
print(results_df)
results_df.plot(kind='bar', figsize=(10,6))
plt.title("Model Performance Comparison")
plt.ylabel("Score")
plt.xticks(rotation=0)
plt.show()
```

Output Screenshot/Text

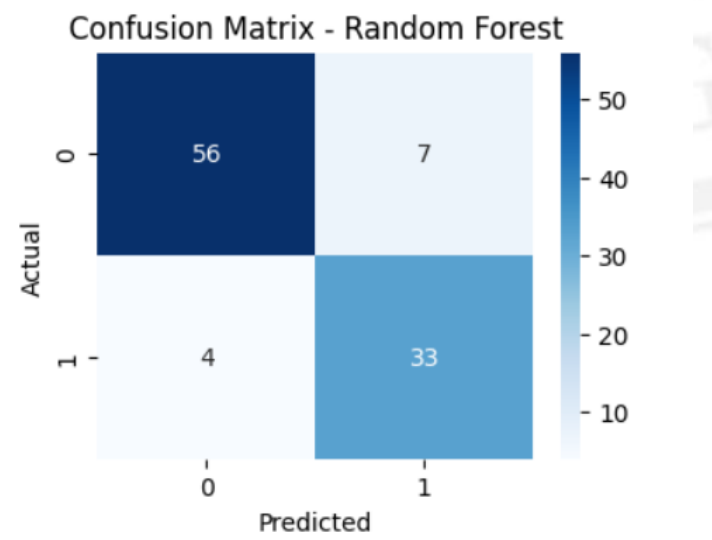
```
Model: Logistic Regression
Accuracy: 0.86
              precision    recall  f1-score   support

           0       0.84      0.97      0.90        63
           1       0.93      0.68      0.78        37

    accuracy                           0.86       100
   macro avg       0.88      0.82      0.84       100
weighted avg       0.87      0.86      0.85       100
```



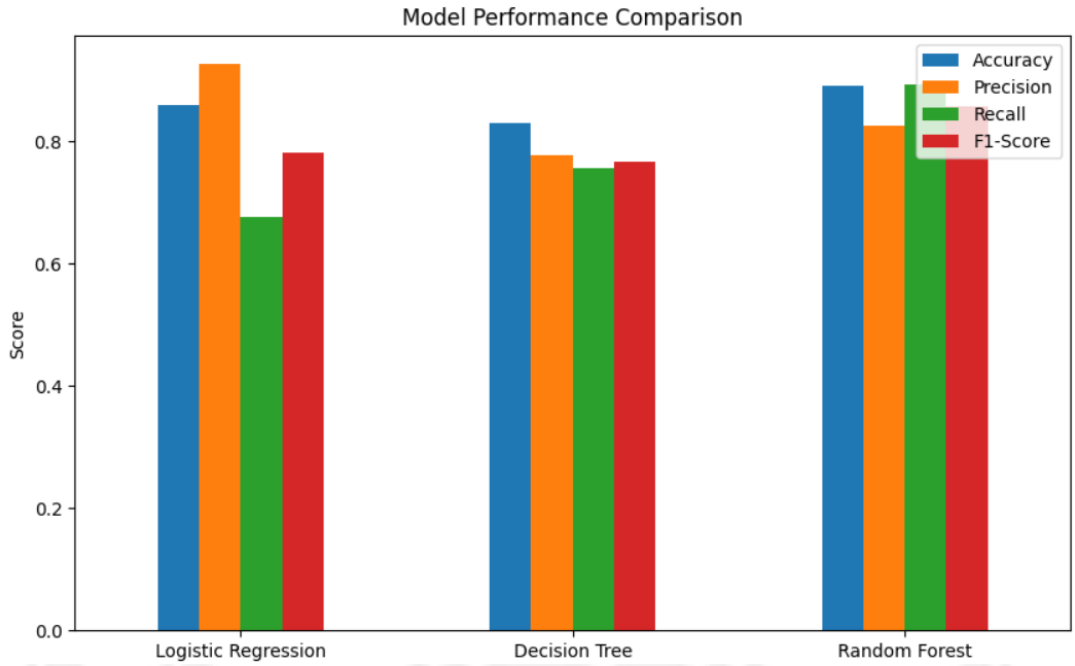
```
Model: Decision Tree
Accuracy: 0.83
              precision    recall  f1-score   support

           0       0.86      0.87      0.87        63
           1       0.78      0.76      0.77        37

    accuracy                           0.83       100
   macro avg       0.82      0.81      0.82       100
weighted avg       0.83      0.83      0.83       100
```




```
Model: Random Forest
Accuracy: 0.89
              precision    recall  f1-score   support

           0       0.93      0.89      0.91        63
           1       0.82      0.89      0.86        37

    accuracy                           0.89       100
   macro avg       0.88      0.89      0.88       100
weighted avg       0.89      0.89      0.89       100
```



```
Comparison of Models:

                    Accuracy  Precision    Recall  F1-Score
Logistic Regression     0.86   0.925926  0.675676  0.781250
Decision Tree           0.83   0.777778  0.756757  0.767123
Random Forest           0.89   0.825000  0.891892  0.857143
```



Department of Computer Science
DATA ANALYTICS USING PYTHON RECORD

Program No:52	Date: 17/09/2025
Name:	Reg No:
Program Title: A Comparative Study of Classification Algorithms on Real-World Data	

IMPORT AND EXPLORE THE DATASET

Loaded the crocodile dataset and examined its structure, types, and basic statistics.

```
import pandas as pd
df = pd.read_csv('/content/crocodile_dataset.csv')
display(df.head())
```

DATA CLEANING AND PREPROCESSING

Fixed data types, handled outliers, and standardized categorical values.

Inspect data – Checking for data types and missing values

```
display(df.info())
display(df.isnull().sum())
display(df.describe())
```

To display unique values and their counts based on various factors

```
for col in ['Common Name', 'Scientific Name', 'Family', 'Genus', 'Age Class', 'Sex', 'Country/Region', 'Habitat Type', 'Conservation Status']:
    print(f"Value counts for {col}:")
    display(df[col].value_counts())
```

Identify and address any extreme values that could skew analysis.

```
display(df[['Observed Length (m)', 'Observed Weight (kg)']].describe())

import matplotlib.pyplot as plt
import seaborn as sns

fig, axes = plt.subplots(1, 2, figsize=(16, 6))
```

```
sns.boxplot(ax=axes[0], y=df['Observed Length (m)'])
axes[0].set_title('Box Plot of Observed Length (m)')
axes[0].set_ylabel('Observed Length (m)')

sns.boxplot(ax=axes[1], y=df['Observed Weight (kg)'])
axes[1].set_title('Box Plot of Observed Weight (kg)')
axes[1].set_ylabel('Observed Weight (kg)')

plt.tight_layout()
plt.show()

fig, axes = plt.subplots(1, 2, figsize=(16, 6))

sns.histplot(ax=axes[0], data=df, x='Observed Length (m)', kde=True)
axes[0].set_title('Histogram of Observed Length (m)')
axes[0].set_xlabel('Observed Length (m)')
axes[0].set_ylabel('Frequency')

sns.histplot(ax=axes[1], data=df, x='Observed Weight (kg)', kde=True)
axes[1].set_title('Histogram of Observed Weight (kg)')
axes[1].set_xlabel('Observed Weight (kg)')
axes[1].set_ylabel('Frequency')

plt.tight_layout()plt.show()
```

Plotted distributions and relationships to understand trends in length, weight, and age class.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Distribution of Observed Length (m)
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Observed Length (m)', kde=True)
plt.title('Distribution of Observed Length (m)')
plt.xlabel('Observed Length (m)')
plt.ylabel('Frequency')
plt.show()

# Distribution of Observed Weight (kg)
plt.figure(figsize=(10, 6))

sns.histplot(data=df, x='Observed Weight (kg)', kde=True)
plt.title('Distribution of Observed Weight (kg)')
plt.xlabel('Observed Weight (kg)')
plt.ylabel('Frequency')
plt.show()

# Relationship between Observed Length and Observed Weight
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Observed Length (m)', y='Observed Weight (kg)', hue='Age
Class')
plt.title('Observed Length vs Observed Weight')
```

```
plt.xlabel('Observed Length (m)')
plt.ylabel('Observed Weight (kg)')
plt.show()

# Box plot of Observed Length by Age Class
plt.figure(figsize=(12, 6))
sns.boxplot(data=df, x='Age Class', y='Observed Length (m)')
plt.title('Observed Length by Age Class')
plt.xlabel('Age Class')
plt.ylabel('Observed Length (m)')
plt.show()

# Box plot of Observed Weight by Age Class
plt.figure(figsize=(12, 6))
sns.boxplot(data=df, x='Age Class', y='Observed Weight (kg)')
plt.title('Observed Weight by Age Class')
plt.xlabel('Age Class')
plt.ylabel('Observed Weight (kg)')
plt.show()
```

Chose Logistic Regression as a baseline and Random Forest as a robust ensemble method.sets.

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

logistic_model = LogisticRegression(max_iter=1000)
logistic_model.fit(X_train, y_train)

y_pred_logistic = logistic_model.predict(X_test)

print("Logistic Regression Performance:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_logistic):.4f}")
print(f"Precision (weighted): {precision_score(y_test, y_pred_logistic, average='weighted'):.4f}")
print(f"Recall (weighted): {recall_score(y_test, y_pred_logistic, average='weighted'):.4f}")
print(f"F1-score (weighted): {f1_score(y_test, y_pred_logistic, average='weighted'):.4f}")

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions with Random Forest Classifier model
y_pred_rf = rf_model.predict(X_test)

# Evaluate Random Forest Classifier model
print("\nRandom Forest Classifier Performance:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_rf):.4f}")
print(f"Precision (weighted): {precision_score(y_test, y_pred_rf, average='weighted'):.4f}")
print(f"Recall (weighted): {recall_score(y_test, y_pred_rf, average='weighted'):.4f}")
print(f"F1-score (weighted): {f1_score(y_test, y_pred_rf, average='weighted'):.4f}")
```

Chose Logistic Regression as a baseline and Random Forest as a robust ensemble method.

```
print("Model Performance Comparison:")
print("\nLogistic Regression Performance:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_logistic):.4f}")
print(f"Precision(weighted):{precision_score(y_test,y_pred_logistic,average='weighted'):.4f}")
print(f"Recall(weighted):{recall_score(y_test,y_pred_logistic, average='weighted'):.4f}")
print(f"F1-score(weighted):{f1_score(y_test,y_pred_logistic, average='weighted'):.4f}")
print("\nRandom Forest Classifier Performance:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_rf):.4f}")
print(f"Precision(weighted):{precision_score(y_test,y_pred_rf, average='weighted'):.4f}")
print(f"Recall(weighted): {recall_score(y_test, y_pred_rf, average='weighted'):.4f}")
print(f"F1-score (weighted): {f1_score(y_test, y_pred_rf, average='weighted'
```

ROC Curve

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.preprocessing import label_binarize

# Binarize the output for multi-class ROC
y_test_bin = label_binarize(y_test, classes=logistic_model.classes_)
n_classes = y_test_bin.shape[1]

# Get the probability of each class for each model
y_prob_logistic = logistic_model.predict_proba(X_test)
y_prob_rf = rf_model.predict_proba(X_test)

# Compute ROC curve and ROC area for each class for Logistic Regression
fpr_logistic = dict()
tpr_logistic = dict()
roc_auc_logistic = dict()
for i in range(n_classes):
```

```
    fpr_logistic[i], tpr_logistic[i], _ = roc_curve(y_test_bin[:, i], y_prob_logistic[:, i])
    roc_auc_logistic[i] = auc(fpr_logistic[i], tpr_logistic[i])

# Compute ROC curve and ROC area for each class for Random Forest Classifier
fpr_rf = dict()
tpr_rf = dict()
roc_auc_rf = dict()
for i in range(n_classes):
    fpr_rf[i], tpr_rf[i], _ = roc_curve(y_test_bin[:, i], y_prob_rf[:, i])
    roc_auc_rf[i] = auc(fpr_rf[i], tpr_rf[i])

# Plot ROC curves for each class
plt.figure(figsize=(12, 10))
colors = ['blue', 'red', 'green', 'purple', 'orange'] # Define colors for each class
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr_logistic[i], tpr_logistic[i], color=color, lw=2,
                 label=f'Logistic Regression (Class {logistic_model.classes_[i]}) (AUC = {roc_auc_logistic[i]:.2f})',
             linestyle='--')
    plt.plot(fpr_rf[i], tpr_rf[i], color=color, lw=2,
            label=f'Random Forest (Class {rf_model.classes_[i]}) (AUC = {roc_auc_rf[i]:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Rece
```

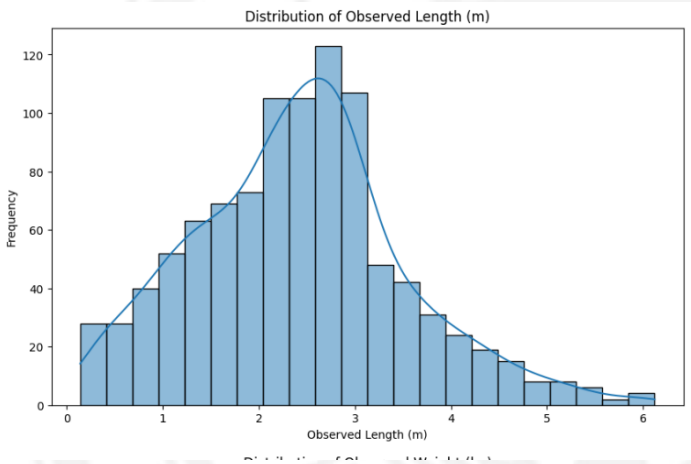
Output:

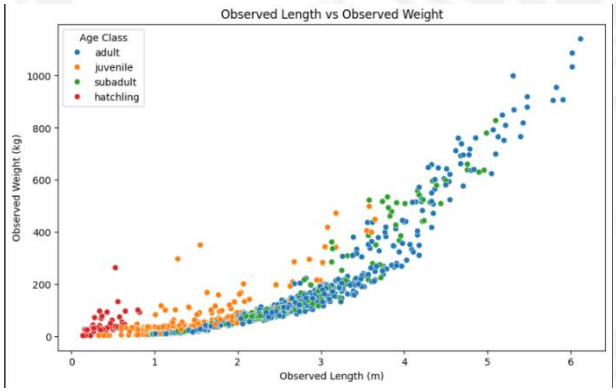
	Observation ID	Common Name	Scientific Name	Family	Genus	Observed Length (m)	Observed Weight (kg)	Age Class	Sex	Date of Observation	Country/Region	Habitat Type	Conservation Status	Observer Name	
0	1	Morelet's Crocodile	Crocodylus moreletii	Crocodylidae	Crocodylus	1.90	62.0	Adult	Male	31-03-2018	Belize	Swamps	Least Concern	Allison Hill	Ca scientisi oppo
1	2	American Crocodile	Crocodylus acutus	Crocodylidae	Crocodylus	4.09	334.5	Adult	Male	28-01-2015	Venezuela	Mangroves	Vulnerable	Brandon Hall	Ago practice det op
2	3	Orinoco Crocodile	Crocodylus intermedius	Crocodylidae	Crocodylus	1.08	118.2	Juvenile	Unknown	07-12-2010	Venezuela	Flooded Savannas	Critically Endangered	Melissa Peterson	Dem shake t gr enoug
3	4	Morelet's Crocodile	Crocodylus moreletii	Crocodylidae	Crocodylus	2.42	90.4	Adult	Male	01-11-2019	Mexico	Rivers	Least Concern	Edward Fuller	Office direct
4	5	Mugger Crocodile (Marsh Crocodile)	Crocodylus palustris	Crocodylidae	Crocodylus	3.75	269.4	Adult	Unknown	15-07-2019	India	Rivers	Vulnerable	Donald Reid	Clas prove raise play

```
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Observation ID        1000 non-null   int64
 1   Common Name           1000 non-null   object
 2   Scientific Name       1000 non-null   object
 3   Family                1000 non-null   object
 4   Genus                 1000 non-null   object
 5   Observed Length (m)   1000 non-null   float64
 6   Observed Weight (kg)  1000 non-null   float64
 7   Age Class             1000 non-null   object
 8   Sex                   1000 non-null   object
 9   Date of Observation   1000 non-null   object
 10  Country/Region        1000 non-null   object
 11  Habitat Type          1000 non-null   object
 12  Conservation Status   1000 non-null   object
 13  Observer Name         1000 non-null   object
 14  Notes                 1000 non-null   object
```

Common Name	
New Guinea Crocodile	68
Borneo Crocodile (disputed)	67
American Crocodile	66
Morelet's Crocodile	64
Cuban Crocodile	59
Orinoco Crocodile	58
Philippine Crocodile	58
Saltwater Crocodile	58
West African Dwarf Crocodile	57
Central African Slender-snouted Crocodile	56
West African Slender-snouted Crocodile	55
West African Crocodile	52
Hall's New Guinea Crocodile	49
Congo Dwarf Crocodile	48
Nile Crocodile	48
Mugger Crocodile (Marsh Crocodile)	47
Siamese Crocodile	45
Freshwater Crocodile (Johnstone's)	45

count	1000.000000	1000.000000
mean	2.415110	155.771900
std	1.097542	175.186788
min	0.140000	4.400000
25%	1.637500	53.225000
50%	2.430000	100.600000
75%	3.010000	168.875000
max	6.120000	1139.700000





```
Logistic Regression Performance:
Accuracy: 1.0000
Precision (weighted): 1.0000
Recall (weighted): 1.0000
F1-score (weighted): 1.0000

Random Forest Classifier Performance:
Accuracy: 1.0000
Precision (weighted): 1.0000
Recall (weighted): 1.0000
F1-score (weighted): 1.0000
```