**Practical 1:**

**AIM**: Python program to display details about the operating system, working directory, files And directories in the current directory, lists the files and all directories, scan and classify them as directories and files

import os

def scan\_and\_classify(path='.'):

directories = []

files = []

for entry in os.listdir(path):

full\_path = os.path.join(path, entry)

if os.path.isdir(full\_path):

directories.append(entry)

elif os.path.isfile(full\_path):

files.append(entry)

return directories, files

path = '.'

directories, files = scan\_and\_classify(path)

print("Directories:")

for directory in directories:

print(directory)

print("\nFiles:")

for file in files:

print(file)

Output:



**Practical 2:**

**AIM:** Python program to convert an array to an array of machine values and vice versa

import struct

def array\_to\_machine\_values(arr, format\_char):

return struct.pack(f'{len(arr)}{format\_char}', \*arr)

def machine\_values\_to\_array(machine\_values, format\_char):

num\_elements = len(machine\_values) // struct.calcsize(format\_char)

return list(struct.unpack(f'{num\_elements}{format\_char}', machine\_values))

original\_array = [10, 20, 30, 40]

format\_char = 'i' # 'i' is for integer

machine\_values = array\_to\_machine\_values(original\_array, format\_char)

print(f"Machine values (bytes): {machine\_values}")

restored\_array = machine\_values\_to\_array(machine\_values, format\_char)

print(f"Restored array: {restored\_array}")

**Output:**

****

**Practical 3:**

**AIM**: Python program to get information about the file pertaining to the file mode and to get time values with components using local time and gm time.

import os

import time

def get\_file\_info(file\_path):

file\_stat = os.stat(file\_path)

file\_mode = oct(file\_stat.st\_mode) # Convert to octal string representation

access\_time = file\_stat.st\_atime # Time of last access

modify\_time = file\_stat.st\_mtime # Time of last modification

change\_time = file\_stat.st\_ctime # Time of last status change (creation time on Windows)

access\_time\_local = time.localtime(access\_time)

modify\_time\_local = time.localtime(modify\_time)

change\_time\_local = time.localtime(change\_time)

access\_time\_gmt = time.gmtime(access\_time)

modify\_time\_gmt = time.gmtime(modify\_time)

change\_time\_gmt = time.gmtime(change\_time)

file\_info = {

"file\_mode": file\_mode,

"access\_time\_local": time.strftime('%Y-%m-%d %H:%M:%S', access\_time\_local),

"modify\_time\_local": time.strftime('%Y-%m-%d %H:%M:%S', modify\_time\_local),

"change\_time\_local": time.strftime('%Y-%m-%d %H:%M:%S', change\_time\_local),

"access\_time\_gmt": time.strftime('%Y-%m-%d %H:%M:%S', access\_time\_gmt),

"modify\_time\_gmt": time.strftime('%Y-%m-%d %H:%M:%S', modify\_time\_gmt),

"change\_time\_gmt": time.strftime('%Y-%m-%d %H:%M:%S', change\_time\_gmt),

}

return file\_info

file\_path = "Untitled2.ipynb"

file\_info = get\_file\_info(file\_path)

print(f"File Mode: {file\_info['file\_mode']}")

print(f"Access Time (Local): {file\_info['access\_time\_local']}")

print(f"Modify Time (Local): {file\_info['modify\_time\_local']}")

print(f"Change Time (Local): {file\_info['change\_time\_local']}")

print(f"Access Time (GMT): {file\_info['access\_time\_gmt']}")

print(f"Modify Time (GMT): {file\_info['modify\_time\_gmt']}")

print(f"Change Time (GMT): {file\_info['change\_time\_gmt']}")

**Output:**

****

**Practical 4:**

**AIM:** Python program to connect to Google using socket programming

import socket

def connect\_to\_google():

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

host = 'www.google.com'

port = 80 # HTTP port

try:

remote\_ip = socket.gethostbyname(host)

print(f"IP address of {host}: {remote\_ip}")

s.connect((remote\_ip, port))

print(f"Successfully connected to {host} on port {port}")

request = "GET / HTTP/1.1\r\nHost: www.google.com\r\n\r\n"

s.send(request.encode())

response = b""

while True:

part = s.recv(4096)

if not part:

break

response += part

print(f"Received response from {host}:\n{response.decode('utf-8', errors='ignore')}")

except socket.error as e:

print(f"Socket error: {e}")

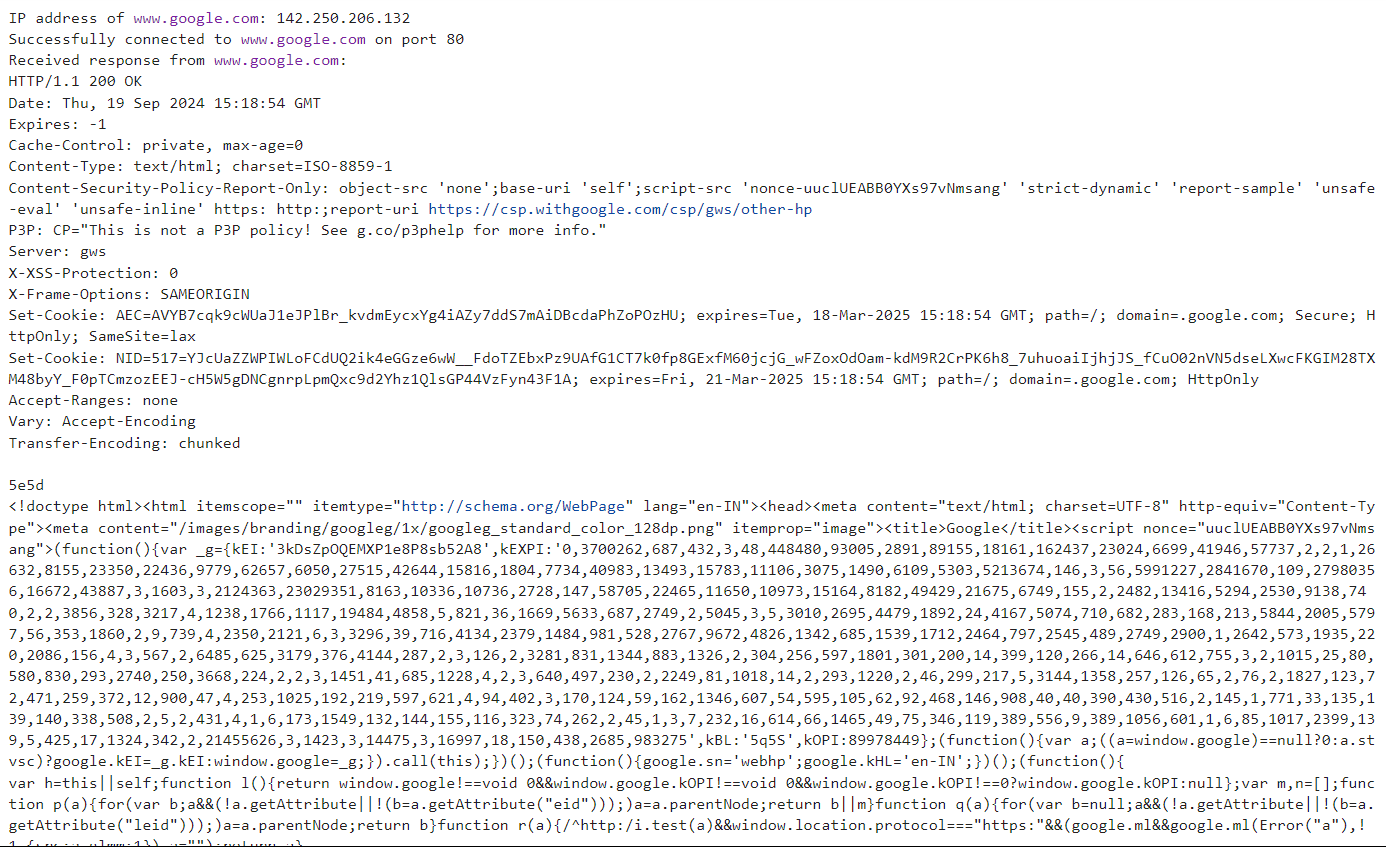
finally:

s.close()

# Run the example

connect\_to\_google()

**Output:**

****

**Practical 5**

**AIM:** Python program to perform Array operations using Numpy package

import numpy as np

def array\_operations():

arr1 = np.array([10, 20, 30, 40])

arr2 = np.array([50, 60, 70, 80])

print("Array 1:", arr1)

print("Array 2:", arr2)

arr\_add = np.add(arr1, arr2)

print("\nAddition of arrays:", arr\_add)

arr\_sub = np.subtract(arr2, arr1)

print("Subtraction of arrays:", arr\_sub)

arr\_mul = np.multiply(arr1, arr2)

print("Multiplication of arrays:", arr\_mul)

arr\_div = np.divide(arr2, arr1)

print("Division of arrays:", arr\_div)

arr\_dot = np.dot(arr1, arr2)

print("\nDot product of arrays:", arr\_dot)

arr\_concat = np.concatenate((arr1, arr2))

print("\nConcatenation of arrays:", arr\_concat)

arr\_reshaped = np.reshape(arr\_concat, (2, 4))

print("\nReshaped array (2x4):\n", arr\_reshaped)

arr\_transposed = np.transpose(arr\_reshaped)

print("\nTransposed array (4x2):\n", arr\_transposed)

max\_element = np.max(arr1)

print("\nMaximum element in Array 1:", max\_element)

min\_element = np.min(arr2)

print("Minimum element in Array 2:", min\_element)

sum\_elements = np.sum(arr1)

print("Sum of elements in Array 1:", sum\_elements)

mean\_elements = np.mean(arr2)

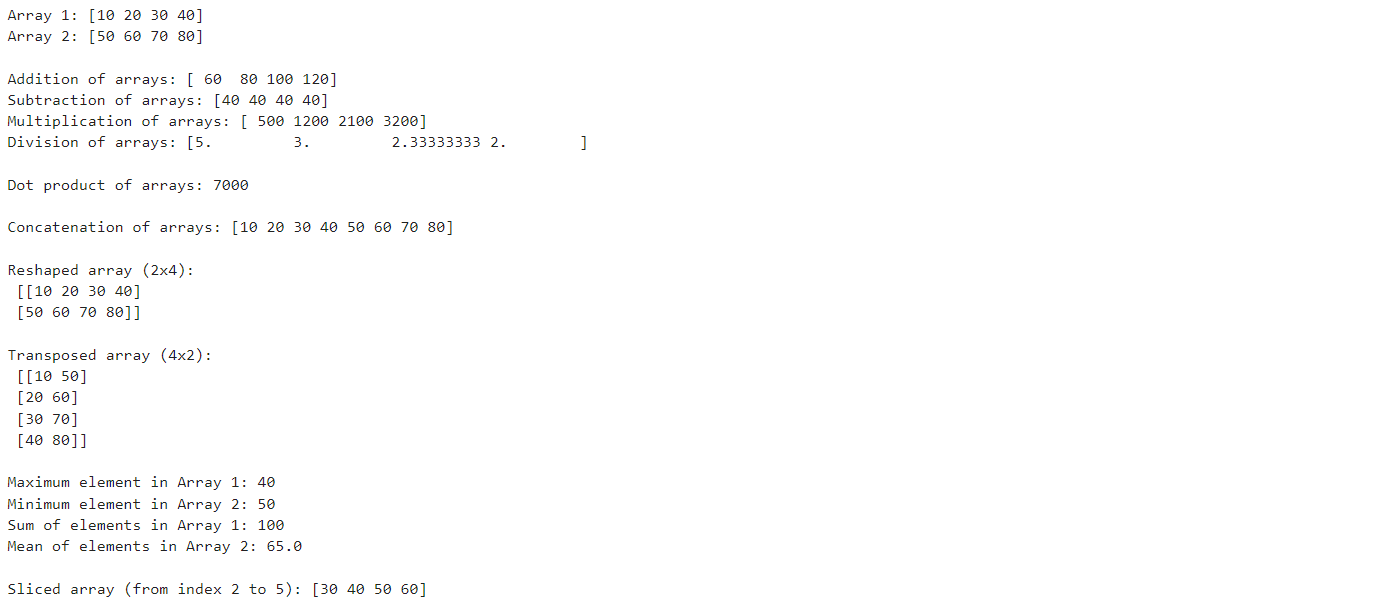
print("Mean of elements in Array 2:", mean\_elements)

arr\_slice = arr\_concat[2:6]

print("\nSliced array (from index 2 to 5):", arr\_slice)

array\_operations()

**Output:**

****

**Practical 6**

**AIM:** Python program to perform Data Manipulation operations using Pandas package.

import pandas as pd

import numpy as np

data = {

'Name': ['Mayank', 'Nilesh', 'Taniya', 'Suhani', 'Rakesh'],

'Age': [20, np.nan, 20, 19, 46],

'City': ['Dwarka', 'Gurugram', 'Delhi', 'Chennai', 'Farrukhnagar'],

'Salary': [90000, 80000, np.nan, 60000, 75000]

}

df = pd.DataFrame(data)

# 2. Basic DataFrame Operations

print("DataFrame:")

print(df)

print("\nFirst 3 rows:")

print(df.head(3))

print("\nDataFrame info:")

print(df.info())

print("\nDescriptive statistics:")

print(df.describe())

# 3. Indexing and Selection

print("\nSelecting 'Name' and 'City':")

print(df[['Name', 'City']])

# 4. Data Cleaning

print("\nDropping rows with missing values:")

df\_cleaned = df.dropna()

print(df\_cleaned)

print("\nFilling missing 'Age' with mean:")

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

print(df)

# 5. Data Transformation

print("\nSorting by 'Salary':")

df\_sorted = df.sort\_values(by='Salary')

print(df\_sorted)

print("\nGrouping by 'City' and calculating mean salary:")

print(df.groupby('City')['Salary'].mean())

# 6. Adding/Removing Columns

df['Experience'] = [2, 3, 1, 5, 4] # Adding a new column

print("\nDataFrame after adding 'Experience':")

print(df)

df.drop(columns=['Experience'], inplace=True) # Removing the 'Experience' column

print("\nDataFrame after dropping 'Experience':")

print(df)

# 7. Filtering and Boolean Indexing

print("\nFiltering for Age > 25:")

filtered\_df = df[df['Age'] > 25]

print(filtered\_df)

# 8. Merging and Joining

data2 = {

'City': ['New York', 'Los Angeles', 'Chicago', 'Houston', 'Phoenix'],

'State': ['NY', 'CA', 'IL', 'TX', 'AZ']

}

df2 = pd.DataFrame(data2)

print("\nMerging DataFrames on 'City':")

merged\_df = pd.merge(df, df2, on='City')

print(merged\_df)

# 9. Time Series

date\_range = pd.date\_range(start='2024-01-01', periods=5)

time\_df = pd.DataFrame({'Date': date\_range, 'Value': [10, 20, 30, 40, 50]})

time\_df.set\_index('Date', inplace=True)

print("\nTime Series DataFrame:")

print(time\_df)

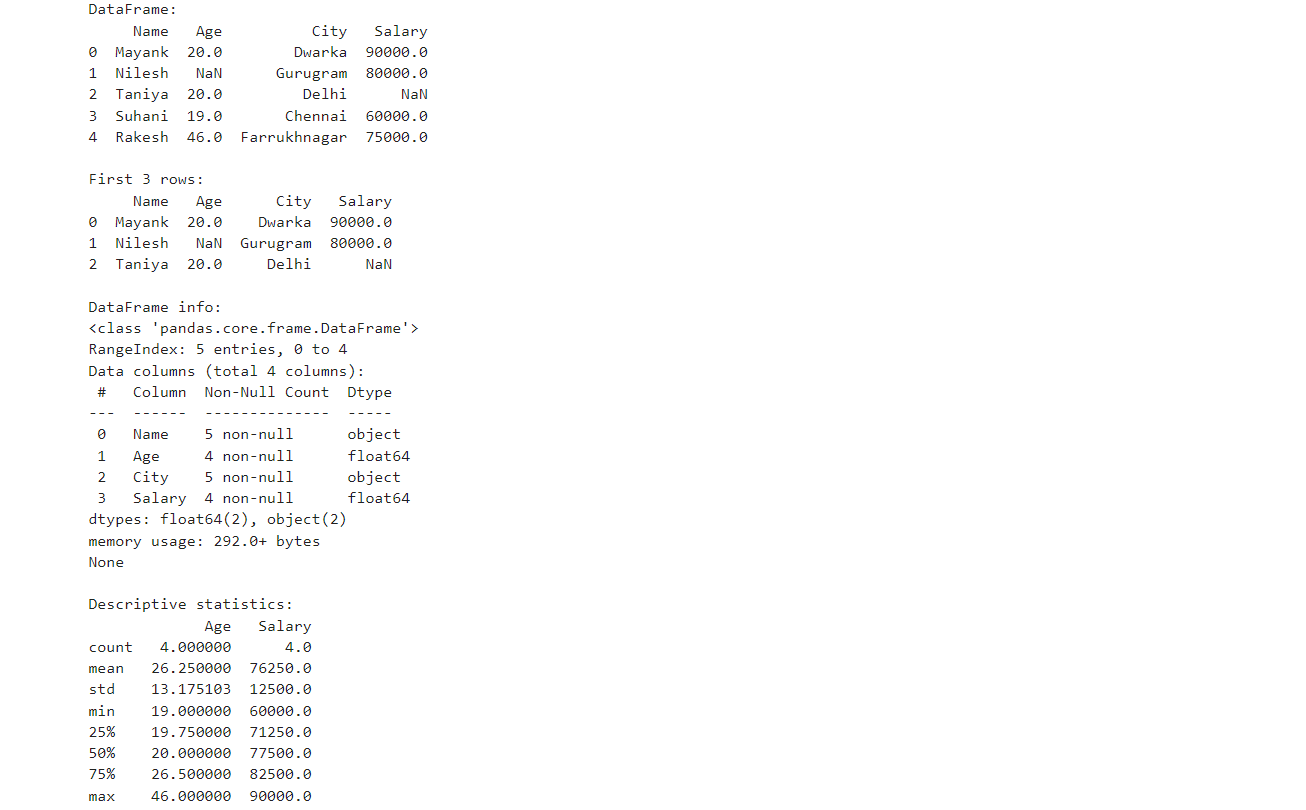
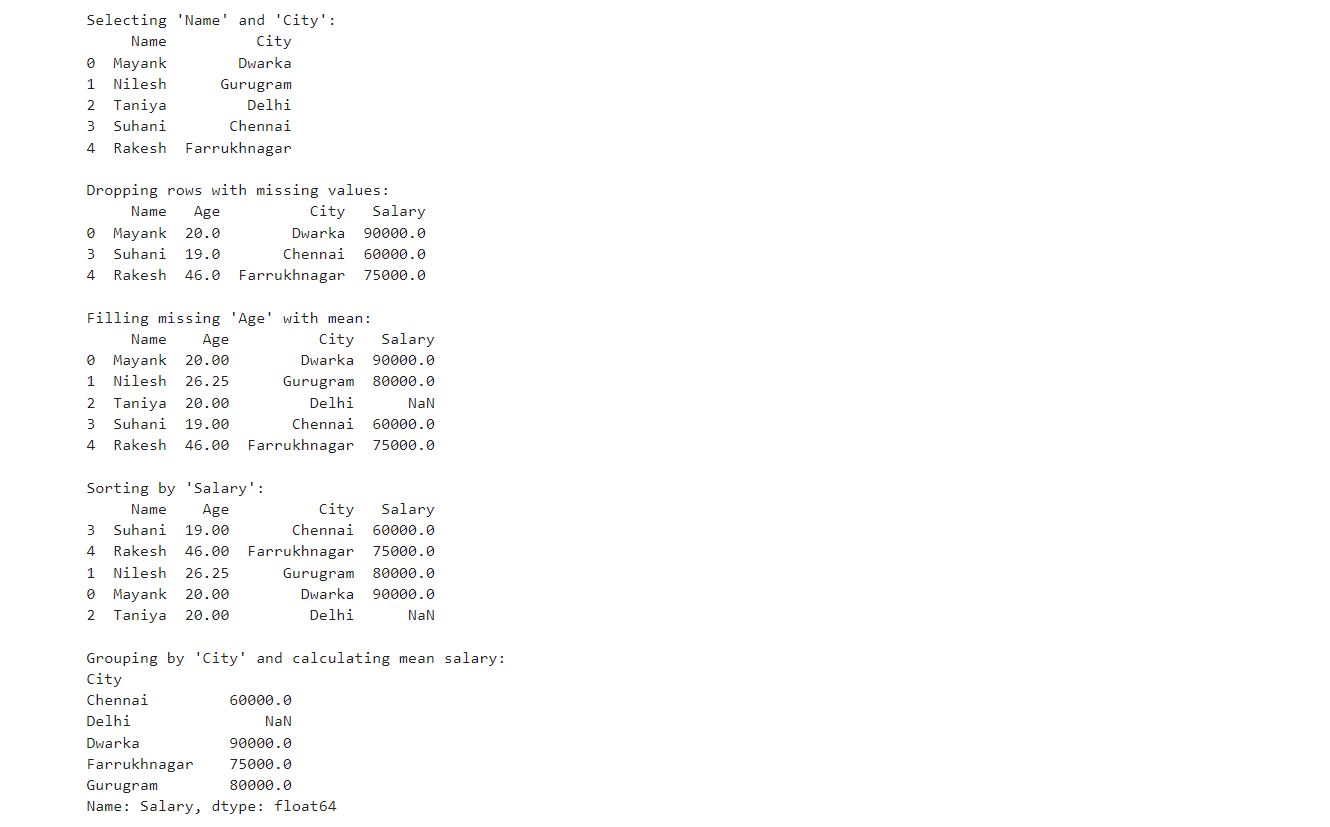
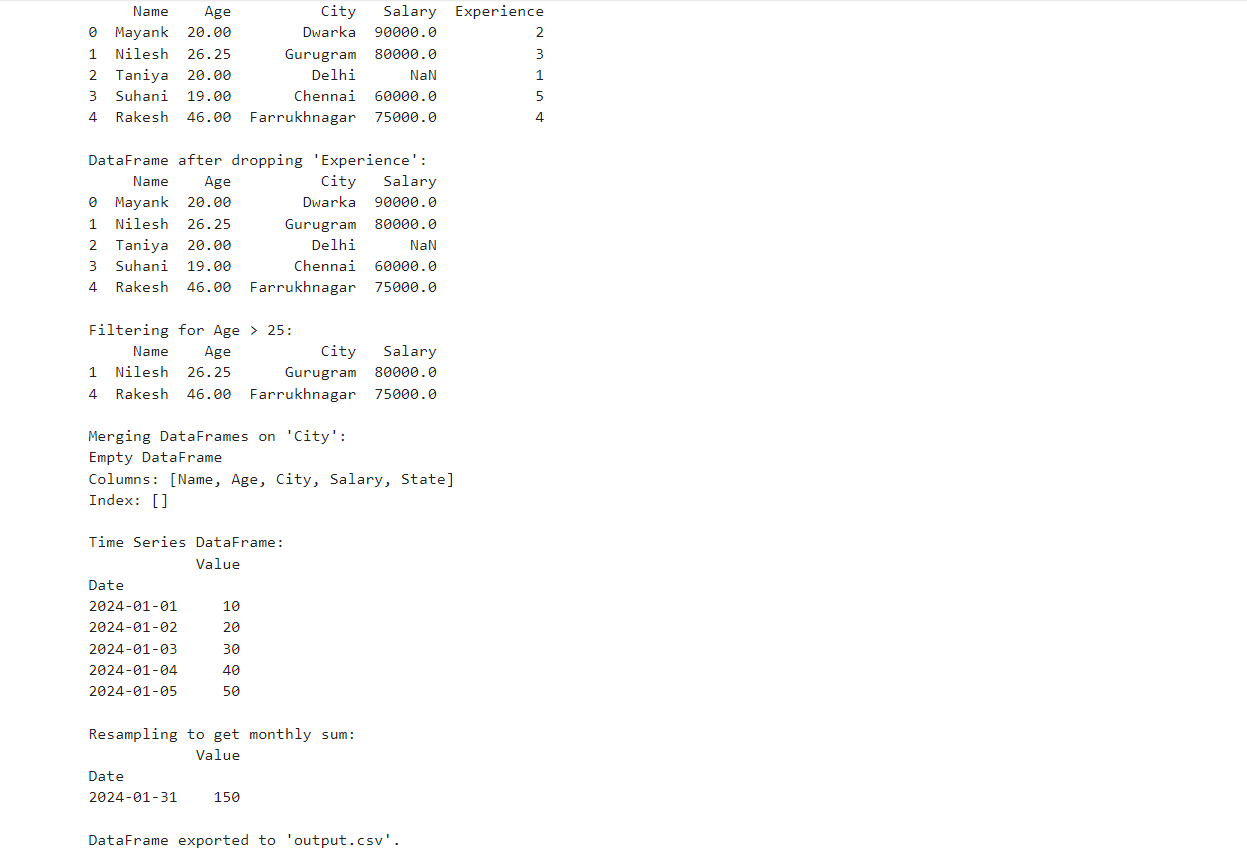
print("\nResampling to get monthly sum:")

print(time\_df.resample('M').sum())

# 10. Exporting Data

df.to\_csv('output.csv', index=False)

print("\nDataFrame exported to 'output.csv'.")

**Output:  **

**Practical 7**

**AIM:** Python program to display multiple types of charts using Matplotlib package.

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

data=pd.read\_csv(r"D:\Programming\Datasets\Dataset\_1\dataset\_1.csv")

top\_10=data.sort\_values(by='Revenue',ascending=False).head(10)

print("The top 10 sorted data of csv file : \n \n",top\_10)

#Line plot

plt.plot(top\_10['Revenue'],top\_10['Country'],marker="o",color="Green")

plt.xlabel("Revenue")

plt.ylabel("Country")

#Bar Plot

plt.bar(top\_10['Country'],top\_10['Revenue'],width=0.5,color="Blue",alpha=0.75)

plt.xlabel("Country")

plt.ylabel("Revenue")

#Scatter Plot

plt.scatter(top\_10['Revenue'],top\_10['Country'],marker="o",color="Green")

plt.xlabel("Revenue")

plt.ylabel("Country")

#Histogram plot

plt.hist(top\_10['Revenue'],bins=10,color="cyan",edgecolor="Red")

#Pie plot

plt.pie(top\_10['Revenue'],labels=top\_10['Country'],autopct='%1.1f')

#Area plot

plt.stackplot(top\_10['Revenue'],top\_10['Country'],color="Green",alpha=0.5)

plt.xlabel("Revenue")

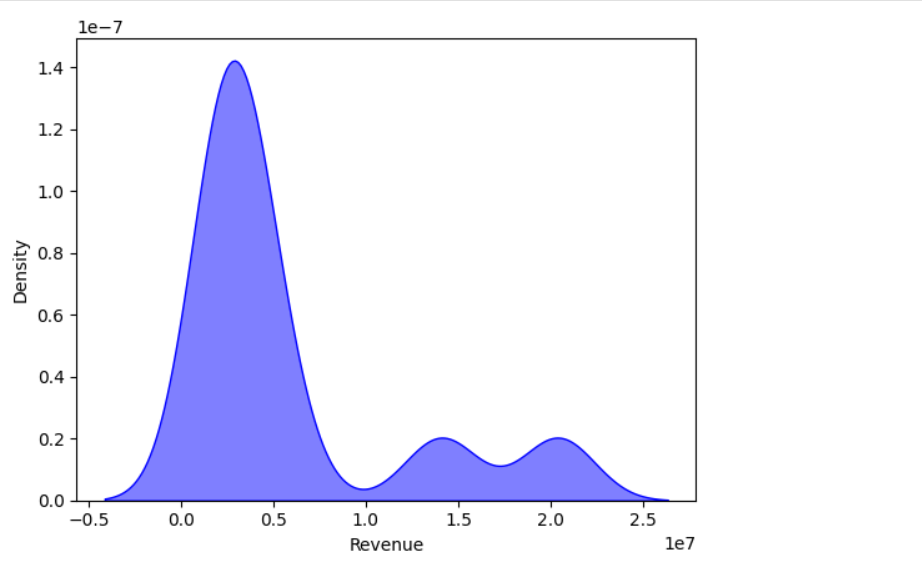
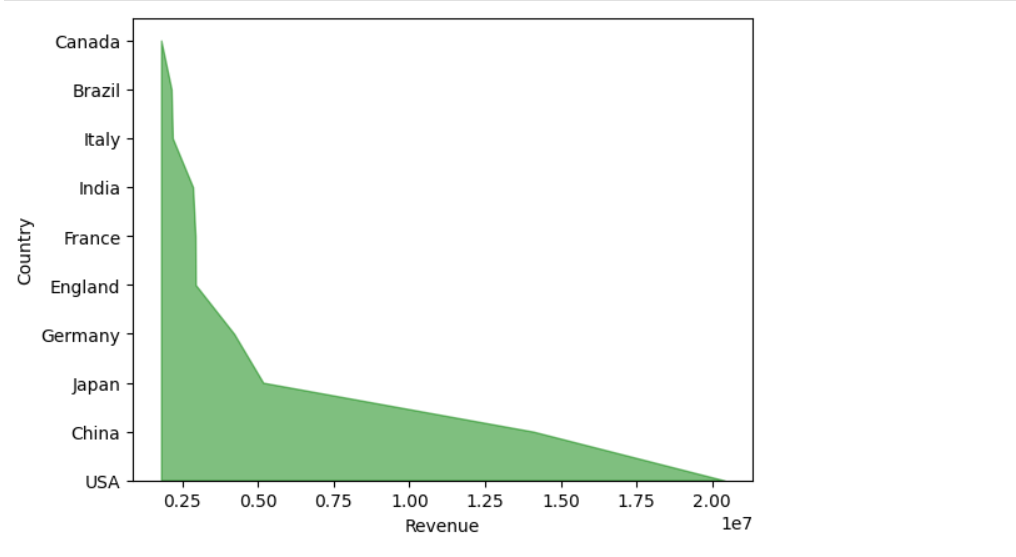
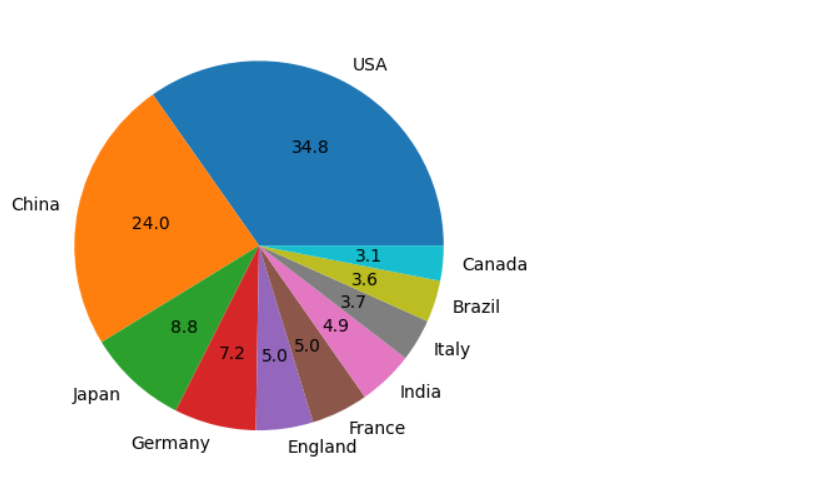
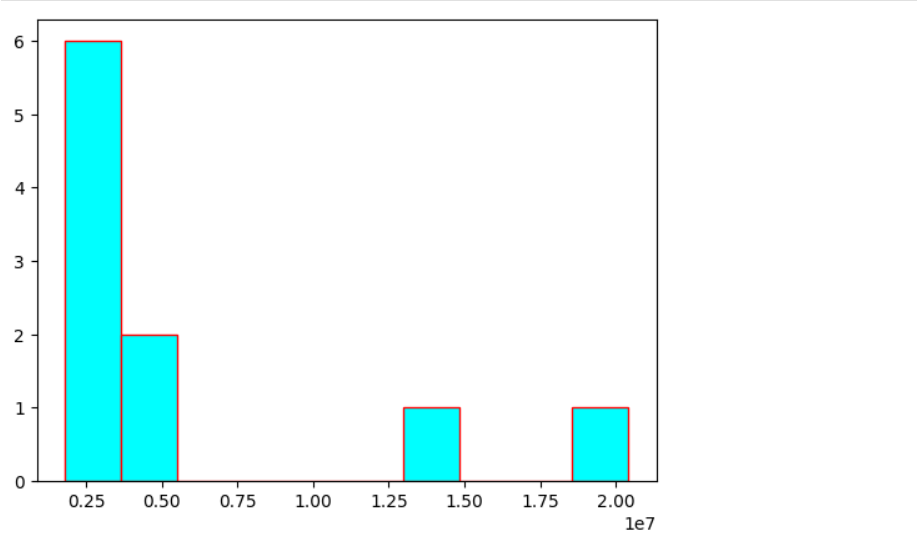
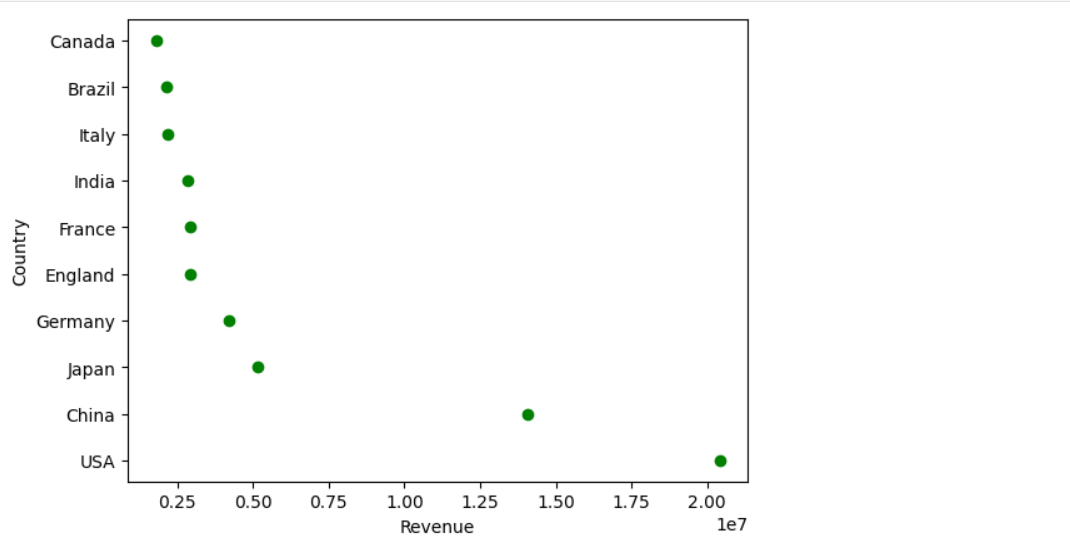
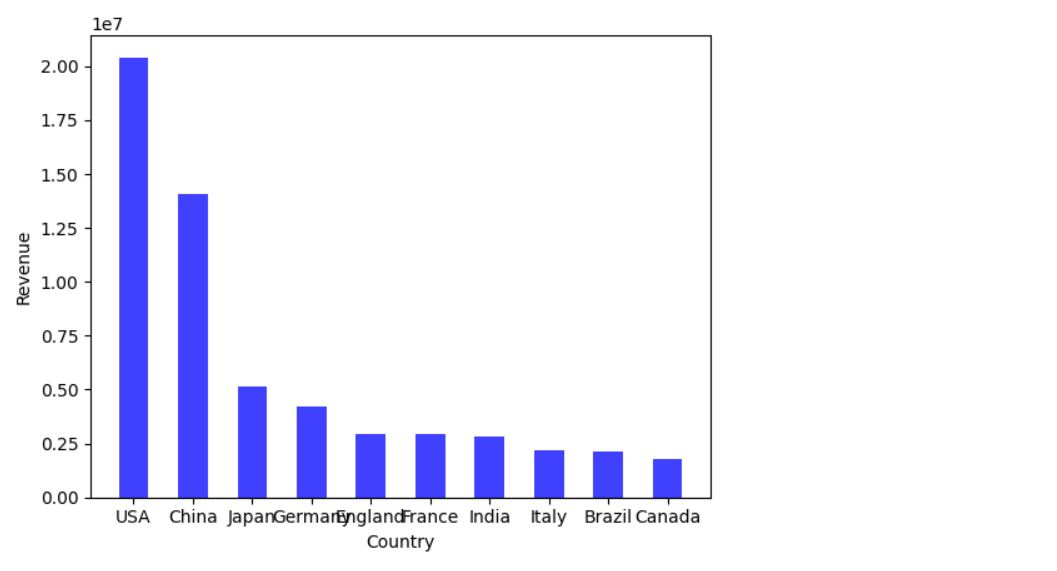
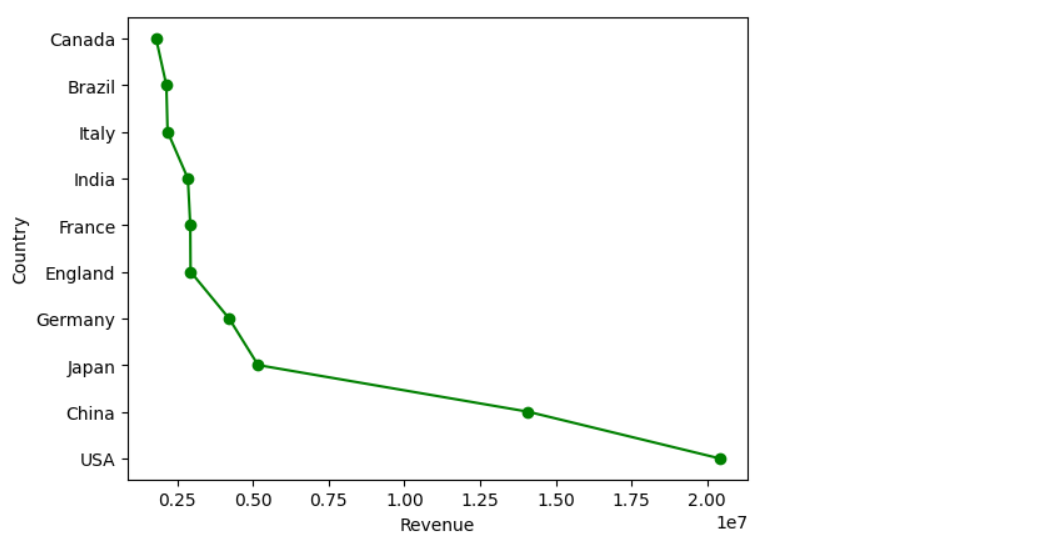
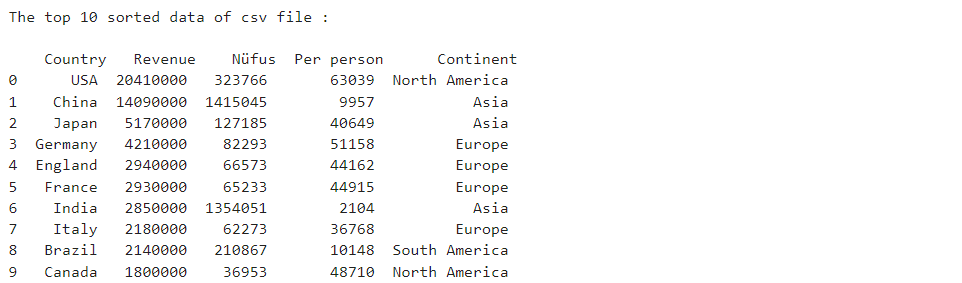
plt.ylabel("Country")

#KDE plot

sns.kdeplot(top\_10['Revenue'], fill=True, color="Blue", alpha=0.5,bw\_adjust=0.5)

plt.show()

**Output:**

****