Practical: 1

Aim: Installation and study of any one Data Analytics Tool Framework.

Source Code:

```
import pandas as pd
data = {
   "Name": ["Rohit", "Rutuja", "Sameer", "Ratnesh"],
   "Age": [22, 21, 21, 21],
   "Salary": [50000, 40000, 30000, 20000]
}
df = pd.DataFrame(data)
print("Original Data:")
print(df)
avg_age = df["Age"].mean()
avg_salary = df["Salary"].mean()
print("\nAverage Age:", avg_age)
print("Average Salary:", avg_salary)
filtered_data = df[df["Age"] > 21]
print("\nPeople above 30:")
print(filtered_data)
```

Practical: 2

Aim: Write a python program to demonstrate the use of Numpy.

Source Code:

```
import numpy as np
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([5, 4, 3, 2, 1])
sum_arr = arr1 + arr2
dot_product = np.dot(arr1, arr2)
mean_arr1 = np.mean(arr1)
matrix = arr1.reshape(1, 5)
```

```
print("Array 1:", arr1)
print("Array 2:", arr2)
print("\nElement-wise Sum:", sum_arr)
print("Dot Product:", dot_product)
print("Mean of Array 1:", mean_arr1)
print("\nReshaped Matrix:\n", matrix)
```

Practical: 3

Aim: Design and develop at least 10 problem statements which demonstrate the use of data structure, functions_ Importing / Exporting Data in any data analytics tool.

- **10 Problem Statements with Solutions**

1. Store Employee Data in a Dictionary & Export to JSON

Problem: Create a dictionary of employees (name, age, salary) and save it as JSON.
Solution:

Source Code:

```
import pandas as pd
employees = {
   "Name": ["Rohit", "Sameer", "Rutuja"],
   "Age": [21, 22, 21],
   "Salary": [50000, 40000, 30000]
}
df = pd.DataFrame(employees)
df.to_json("employees.json")
print("Data exported to JSON!")
```

2. Read CSV, Filter Rows, and Export to Excel

Problem: Load a CSV file, filter rows where age > 30, and save as Excel.
Solution:

Source Code:

```
df = pd.read_csv("Student_Marks.csv")
filtered_data = df[df["Marks"] > 30]
filtered_data.to_excel("filtered_data.xlsx")
```

3. Merge Two DataFrames and Save as CSV

Problem: Combine two datasets (e.g., customers.csv and orders.csv) and export the result.
Solution:

Source Code:

```
customers = pd.read_csv("customers.csv")

orders = pd.read_csv("orders.csv")

merged_data = pd.merge(customers, orders, on="customer_id")

merged_data.to_csv("merged_data.csv")
```

4. Convert List of Tuples to DataFrame & Export

Problem: Take a list of tuples (product, price) and save as CSV.
Solution:

Source Code:

```
products = [("Laptop", 1000), ("Phone", 800), ("Tablet", 500)]

df = pd.DataFrame(products, columns=["Product", "Price"])

df.to_csv("products.csv")
```

5. Read JSON Data, Compute Average, and Export

Problem: Load JSON data, compute the average salary, and save results.
Solution:

Source Code:

```
data = pd.read_json("employees.json")

avg_salary = data["Salary"].mean()

result = {"Average Salary": avg_salary}

pd.DataFrame(result, index=[0]).to_csv("avg_salary.csv")
```

6. Generate Random Data, Store in DataFrame, Export to Excel

Problem: Create random sales data and export it.
Solution:

Source Code:

```
import numpy as np

sales_data = {
```

```
    "Product": ["A", "B", "C"],
    "Sales": np.random.randint(100, 1000, 3)
}
df = pd.DataFrame(sales_data)
df.to_excel("sales.xlsx")
```

7. Read Excel, Apply Discount, and Save

Problem: Load an Excel file, apply a 10% discount on prices, and save.
Solution:

Source Code:

```
df = pd.read_excel("products.xlsx")
df["Discounted_Price"] = df["Price"] * 0.9
df.to_csv("discounted_products.csv")
```

8. Group Data by Category & Export Summary

Problem: Group sales data by category and compute total sales.
Solution:

Source Code:

```
sales = pd.read_csv("sales.csv")
grouped = sales.groupby("Product")["Sales"].sum()
grouped.to_csv("sales_summary.csv")
```

9. Convert Dictionary to DataFrame & Save as CSV

Problem: Store student records (name, marks) in a dictionary and export.
Solution:

Source Code:

```
students = {"Name": ["Rohit", "Priya", "Rutuja"], "Marks": [85, 80, 75]}
df = pd.DataFrame(students)
df.to_csv("students.csv")
```

10. Read Web Data (API) & Store in CSV

Problem: Fetch data from a JSON API and save it.
Solution:

Source Code:

```
import requests

response =
requests.get("https://data.gov.sg/api/action/datastore_search?resource_id=f1765b54-a209-
4718-8d38-a39237f502b3")

data = pd.DataFrame(response.json())

data.to_csv("api_data.csv")
```