**Lab Exercise 13 - Data Transformation using Pandas and NumPy**

**Objective**

To understand and apply data transformation techniques using **Pandas** and **NumPy** for preparing data for analysis.

**Prerequisites**

* Python 3.x installed
* pandas and numpy libraries installed  
  *(Use pip install pandas numpy if needed)*

**Dataset**

Use a sample CSV file (you can create this manually or generate it using code). Name it employee\_data.csv:

EmpID,Name,Department,Age,Salary,JoiningDate

101,John Doe,Sales,29,55000,2015-03-15

102,Jane Smith,Marketing,34,62000,2013-07-01

103,Tom Hanks,HR,41,59000,2011-11-23

104,Alice Johnson,Sales,25,48000,2018-01-30

105,Bob Martin,IT,38,73000,2010-06-12

106,Eva Brown,Marketing,30,54000,2017-04-19

**Lab Tasks**

**Step 1: Load and Explore the Dataset**

import pandas as pd

# Load CSV file

df = pd.read\_csv("employee\_data.csv")

# View first 5 rows

print(df.head())

# Get data types and summary

print(df.info())

print(df.describe())

**Step 2: Handle Missing and Incorrect Data**

(Simulate missing/invalid data first if needed.)

import numpy as np

# Add some missing values

df.loc[2, 'Salary'] = np.nan

df.loc[4, 'Age'] = np.nan

# Fill missing salary with mean salary

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

# Fill missing age with median age

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

**Step 3: Apply Transformations using Pandas**

# Add a new column: Salary in thousands

df['SalaryK'] = df['Salary'] / 1000

# Convert JoiningDate to datetime

df['JoiningDate'] = pd.to\_datetime(df['JoiningDate'])

# Calculate years of experience

df['Experience'] = pd.Timestamp.now().year - df['JoiningDate'].dt.year

# Capitalize department names

df['Department'] = df['Department'].str.upper()

# Rename column

df.rename(columns={'Name': 'EmployeeName'}, inplace=True)

**Step 4: Apply NumPy Transformations**

# Create a performance score based on salary (mock logic)

df['PerformanceScore'] = np.where(df['Salary'] > 60000, 'High',

np.where(df['Salary'] > 50000, 'Medium', 'Low'))

# Normalize the salary column (min-max scaling)

df['NormalizedSalary'] = (df['Salary'] - df['Salary'].min()) / (df['Salary'].max() - df['Salary'].min())

**Step 5: Aggregations and Grouping**

# Group by department and calculate average salary

avg\_salary = df.groupby('Department')['Salary'].mean()

print("\nAverage Salary by Department:\n", avg\_salary)

# Count employees in each department

emp\_count = df['Department'].value\_counts()

print("\nEmployee Count by Department:\n", emp\_count)

**Step 6: Export the Transformed Data**

# Save the transformed DataFrame to a new CSV

df.to\_csv("transformed\_employee\_data.csv", index=False)