**WEEK-7**

**Data Analysis using Pandas**

**Description:**

This unit focuses on Pandas, the most widely used Python library for data analysis and manipulation. It introduces the primary data structures in Pandas—Series and DataFrame—and explores methods for loading, analyzing, and cleaning datasets.

Students will learn how to:

* Create and manipulate Series and DataFrames
* Handle missing data using fillna(), dropna(), and related functions
* Perform data filtering, sorting, grouping, and aggregation
* Merge, join, and concatenate datasets
* Import/export data from/to CSV, Excel, and other file formats

The unit emphasizes real-world data analysis tasks such as summarizing large datasets, applying statistical functions, and reshaping data structures using operations like pivoting and stacking. By the end of this unit, students will be equipped with practical skills to handle end-to-end data analysis pipelines using Pandas.

**Additional Programs**

**1.Write a Python program to create a Pandas Series and DataFrame and perform basic operations like indexing, slicing, and updating values.**

**Description:**

This program introduces Pandas’ core data structures: Series (1D labeled array) and DataFrame (2D labeled table). It demonstrates how to create them, access specific elements using labels and positions, perform slicing operations, and update values.

Understanding these basic operations is essential before moving on to complex data transformations and analyses.

These operations form the foundation for more advanced data processing and numerical tasks using NumPy.

**Program:**

import pandas as pd

# Creating a Series

marks = pd.Series([85, 90, 78, 92], index=['John', 'Alice', 'Bob', 'Daisy'])

print("Series:\n", marks)

# Accessing values

print("\nMark of Alice:", marks['Alice'])

# Updating value

marks['Bob'] = 88

print("\nUpdated Series:\n", marks)

# Creating a DataFrame

data = {

'Name': ['John', 'Alice', 'Bob', 'Daisy'],

'Math': [85, 90, 88, 92],

'Science': [78, 88, 82, 95]

}

df = pd.DataFrame(data)

print("\nDataFrame:\n", df)

# Indexing and slicing

print("\nFirst two rows:\n", df.iloc[:2])

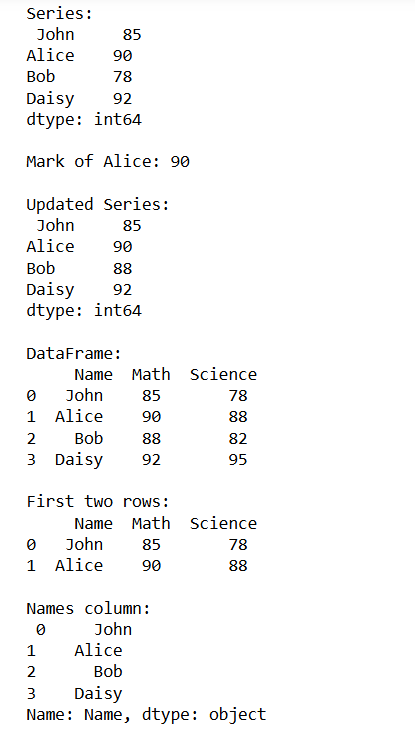
print("\nNames column:\n", df['Name'])

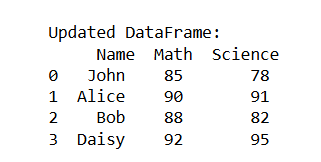
# Updating a value

df.at[1, 'Science'] = 91

print("\nUpdated DataFrame:\n", df)

**Output :**

****

****

**2.Write a program to handle missing data in a Pandas DataFrame using isnull(), fillna(), and dropna().**

**Description:**

Handling missing data is a fundamental part of data cleaning in any analysis process. This experiment introduces functions used to detect, fill, and remove missing values in a Pandas DataFrame. These techniques are crucial in preparing datasets for analysis or machine learning.

**Program:**

import pandas as pd

import numpy as np

# Creating a DataFrame with missing values

data = {

'Name': ['Alice', 'Bob', 'Charlie', 'David'],

'Age': [25, np.nan, 30, np.nan],

'Score': [85, 90, np.nan, 88]

}

df = pd.DataFrame(data)

print("Original DataFrame:\n", df)

# Detect missing values

print("\nMissing values:\n", df.isnull())

# Fill missing values

df\_filled = df.fillna({'Age': df['Age'].mean(), 'Score': 0})

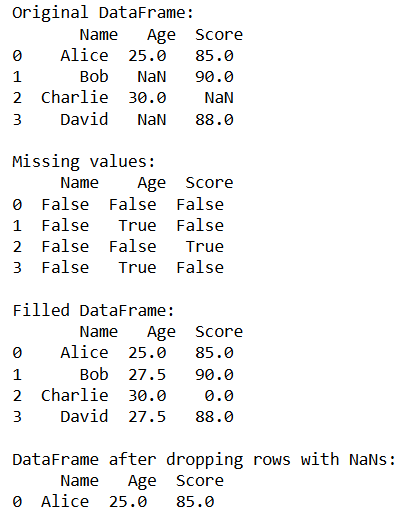
print("\nFilled DataFrame:\n", df\_filled)

# Drop rows with missing data

df\_dropped = df.dropna()

print("\nDataFrame after dropping rows with NaNs:\n", df\_dropped)

**Output:**

****

**3.Write a program to group and summarize data using Pandas groupby() and perform merging of two DataFrames.**

**Description:**

This experiment demonstrates how to group data by one or more columns and apply aggregate functions, and also shows how to merge datasets using merge() and concat(). These operations are widely used in real-world analytics pipelines where data comes from multiple sources or needs summarization.

**Program:**

import pandas as pd

# Sample data for grouping

data = {

'Department': ['Sales', 'Sales', 'HR', 'HR', 'IT'],

'Employee': ['John', 'Alice', 'Bob', 'Eve', 'Charlie'],

'Salary': [50000, 60000, 45000, 47000, 55000]

}

df = pd.DataFrame(data)

# Grouping by Department

grouped = df.groupby('Department')['Salary'].mean()

print("Average salary by department:\n", grouped)

# Sample data for merging

data1 = pd.DataFrame({

'EmpID': [1, 2, 3],

'Name': ['John', 'Alice', 'Bob']

})

data2 = pd.DataFrame({

'EmpID': [1, 2, 4],

'Department': ['Sales', 'HR', 'IT']

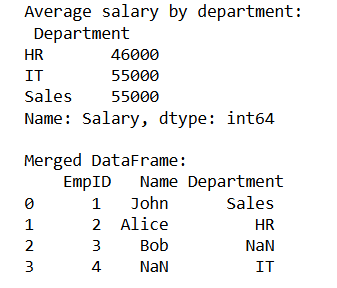
})

# Merging on EmpID

merged = pd.merge(data1, data2, on='EmpID', how='outer')

print("\nMerged DataFrame:\n", merged)

**Output:**

****

**Viva Questions and Answers**

**1. What is broadcasting in Pandas and when does it occur?**Broadcasting is when operations are automatically applied across rows or columns without manual looping. For example, adding a scalar value to a column or performing row-wise operations with a single Series.

1. **What is the difference between axis=0 and axis=1 in Pandas operations?**

* axis=0 refers to operations along rows (down the columns).
* axis=1 refers to operations along columns (across rows).  
  Example: df.dropna(axis=1) drops columns with missing values.

1. **How do you set a custom index for a DataFrame?**You can use the index parameter while creating a DataFrame or assign to df.index:

df = pd.DataFrame(data, index=['row1', 'row2'])

df.index = ['a', 'b', 'c']