### **Week 4: Data Manipulation with Pandas**

#### **Overview**

This week, we will focus on data manipulation using Pandas, a powerful and flexible data analysis library in Python. Pandas provides essential data structures and functions designed to make data cleaning and analysis fast and straightforward. By the end of this week, you will be able to perform data manipulation tasks, including sorting, filtering, and grouping data, using Pandas.

#### **Learning Objectives**

By the end of this module, students will be able to:

1. Understand and use Pandas DataFrames and Series.
2. Read and write data using Pandas.
3. Perform data manipulation tasks such as sorting, filtering, and grouping.
4. Handle large datasets efficiently with Pandas.

#### **Introduction to Pandas DataFrames and Series**

Pandas introduces two primary data structures: Series and DataFrame.

1. **Series**: A one-dimensional array-like object containing a sequence of values. It is similar to a list or a column in a spreadsheet.
2. **DataFrame**: A two-dimensional, tabular data structure with labeled axes (rows and columns). It is similar to a table in a database or an Excel spreadsheet.

Example:

import pandas as pd

# Creating a Series

data\_series = pd.Series([10, 20, 30, 40])

print(data\_series)

# Creating a DataFrame

data = {

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

'Age': [25, 30, 35, 40],

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

}

data\_frame = pd.DataFrame(data)

print(data\_frame)

#### **Reading and Writing Data with Pandas**

Pandas can read data from various file formats and write data to these formats. The most common formats include CSV, Excel, and JSON.

**Reading Data**:

# Reading data from a CSV file

data = pd.read\_csv('data.csv')

# Reading data from an Excel file

data = pd.read\_excel('data.xlsx')

# Reading data from a JSON file

data = pd.read\_json('data.json')

**Writing Data**:

# Writing data to a CSV file

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

# Writing data to an Excel file

data.to\_excel('output.xlsx', index=False)

# Writing data to a JSON file

data.to\_json('output.json')

#### **Data Manipulation: Sorting, Filtering, and Grouping**

**Sorting Data**: Sorting helps organize data in ascending or descending order.

# Sorting by a single column

sorted\_data = data\_frame.sort\_values(by='Age')

# Sorting by multiple columns

sorted\_data = data\_frame.sort\_values(by=['City', 'Age'])

**Filtering Data**: Filtering allows you to extract rows that meet specific criteria.

# Filtering rows where Age is greater than 30

filtered\_data = data\_frame[data\_frame['Age'] > 30]

# Filtering rows based on multiple conditions

filtered\_data = data\_frame[(data\_frame['Age'] > 30) & (data\_frame['City'] == 'New York')]

**Grouping Data**: Grouping data involves splitting the data into groups based on some criteria and then applying a function to each group independently.

# Grouping by a single column and calculating the mean age

grouped\_data = data\_frame.groupby('City')['Age'].mean()

# Grouping by multiple columns and calculating summary statistics

grouped\_data = data\_frame.groupby(['City', 'Age']).size()

#### **Handling Large Datasets**

Pandas provides several techniques for handling large datasets efficiently, including:

1. **Chunking**: Reading large files in smaller chunks to avoid memory issues.
2. **Optimizing Data Types**: Converting data types to more memory-efficient formats.

**Example**:

# Reading data in chunks

chunk\_size = 1000

chunks = pd.read\_csv('large\_data.csv', chunksize=chunk\_size)

for chunk in chunks:

process(chunk)

# Optimizing data types

data['Age'] = data['Age'].astype('int32')

data['City'] = data['City'].astype('category')

#### **Learning Activities**

To reinforce your understanding of this week's content, complete the following activities:

1. **Reading Assignment**: Read Chapter 4 of "Data Analytics Made Accessible" by Anil Maheshwari, focusing on data manipulation with Pandas.
2. **Video Lecture**: Watch the video "Data Manipulation with Pandas" on YouTube to see practical examples of using Pandas for data manipulation.
3. **Hands-On Exercise**: Download a sample dataset and perform various data manipulation tasks using Pandas in Jupyter Notebook.

#### **Discussion Questions**

Participate in the class discussion by answering the following questions:

1. What are the advantages of using Pandas for data manipulation?
2. How can sorting and filtering data help in gaining insights from a dataset?
3. What challenges might you face when working with large datasets, and how can you address them?

#### **Summary**

This week, we explored data manipulation using Pandas, a powerful Python library. We learned how to work with Pandas DataFrames and Series, read and write data, and perform essential data manipulation tasks such as sorting, filtering, and grouping. We also discussed techniques for handling large datasets efficiently. These skills are crucial for preparing and analyzing data effectively.

#### **Additional Resources**

* **Book**: "Python for Data Analysis" by Wes McKinney.
* **Website**: Visit [DataCamp](https://www.datacamp.com/) for interactive tutorials on data manipulation with Pandas.
* **Tutorial**: Follow the tutorial "Data Manipulation with Pandas" on [Kaggle](https://www.kaggle.com/) to practice with real datasets.

#### **Homework**

1. Write a short essay (300-500 words) on the importance of data manipulation in data analytics.
2. Complete the hands-on exercise and submit your Jupyter Notebook with the manipulated dataset through the course portal.