**Name - Aparna Bharti**

**Batch - Data Engineering batch -1**

**Python Coding Assessment**

Que 1) Explain Pandas for Data Processing

Pandas is a powerful open-source data manipulation and analysis library for Python. It provides easy-to-use data structures and functions needed to manipulate and analyse structured data seamlessly.

The primary data structures in Pandas are Series and DataFrame.

**Series:**

* A one-dimensional labelled array that can hold any data type.
* It is similar to a column in a spreadsheet or a single column in a SQL table.
* Series can be created from lists, arrays, or dictionaries.

import pandas as pd

# Creating a Series

data = pd.Series([1, 2, 3, 4])

**DataFrame:**

* A two-dimensional labelled data structure with columns that can be of different data types.
* It can be thought of as a table or a spreadsheet.
* DataFrames can be created from various data sources such as
* CSV files, SQL queries, Excel files, or Python dictionaries.

import pandas as pd

# Creating a DataFrame from a dictionary

data = {'Name': ['Alice', 'Bob', 'Charlie'],

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

'City': ['New York', 'San Francisco', 'Los Angeles']}

df = pd.DataFrame(data)

### **Key Features of Pandas for Data Processing:**

1. **Data Cleaning:**
   1. Pandas provides functions to handle missing data using methods like dropna() and fillna().
   2. It allows filtering and removing duplicates using drop\_duplicates().
2. **Data Exploration:**
   1. Descriptive statistics and summary functions like describe() help in exploring data quickly.
   2. info() method provides a concise summary of a DataFrame, including data types and missing values.
3. **Data Manipulation:**
   1. Columns can be selected, filtered, and manipulated easily.
   2. Arithmetic operations can be performed on entire columns or rows.
   3. The apply() function allows applying custom functions to data.
4. **Merging and Joining:**
   1. DataFrames can be combined using methods like merge() and concat().
5. **Grouping and Aggregation:**
   1. Grouping data using groupby() allows for applying aggregate functions to specific subsets.
   2. Functions like sum(), mean(), and count() can be applied after grouping.
6. **Time Series Analysis:**
   1. Pandas supports time series data and provides tools for date and time manipulation.
7. **Data Input/Output:**
   1. Pandas can read data from various file formats, such as CSV, Excel, SQL, and more.
   2. Similarly, it can write data back to these formats.

Pandas simplifies the process of data cleaning, exploration, and analysis, making it an essential tool for data scientists, analysts, and researchers working with structured data in Python.

Csv file (new\_file.CSV)

id,name,designation,salary

E001,John Doe,Software Engineer,75000

E002,Jane Smith,Data Analyst,60000

E003,Michael Johnson,Product Manager,90000

E004,Amy White,HR Specialist,80000

Que 2) Execute Reading CSV Data using Pandas

To read CSV data using Pandas, you can use the pd.read\_csv() function. This function reads the contents of a CSV file into a DataFrame

import pandas as pd

file\_path = 'new\_file.csv'

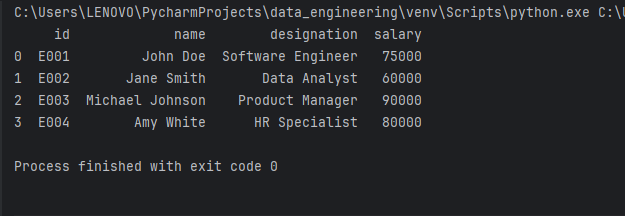
# Read CSV file into a DataFrame

df = pd.read\_csv(file\_path)

# Display the first few rows of the DataFrame

print(df.head())

Output



Que 3) Read Data from CSV Files to Pandas Dataframes

A DataFrame is **a data structure that organises data into a 2-dimensional table of rows and columns, much like a spreadsheet**. DataFrames are one of the most common data structures used in modern data analytics because they are a flexible and intuitive way of storing and working with data.

import pandas as pd

file\_path = 'new\_file.csv'

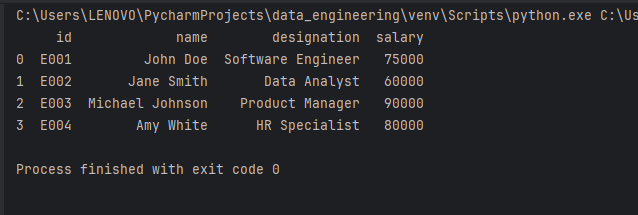
df = pd.read\_csv(file\_path)

# Read CSV file into a DataFrame

data\_frame = pd.DataFrame(df)

print(data\_frame)

Output



Que 4) Filter Data in Pandas Dataframe using query.

You can use the query method in Pandas to filter data in a DataFrame based on a specific condition

import pandas as pd

# Creating a sample DataFrame

data = {'Name': ['Aparna', 'Bharti', 'Chanchal', 'Rishi'],

'Age': [22, 30, 40, 29],

'City': ['indore', 'bhopal', 'Amritsar', 'Chandigarh']}

df = pd.DataFrame(data)

# Display the original DataFrame

print("Original DataFrame:")

print(df)

# Using query to filter data (e.g., selecting individuals older than 30)

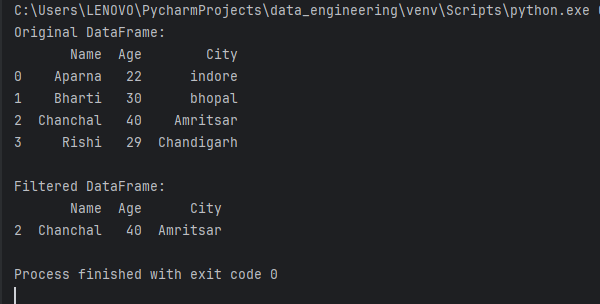
filtered\_df = df.query('Age > 30')

# Display the filtered DataFrame

print("\nFiltered DataFrame:")

print(filtered\_df)

Output



Que 5) Execute with one example Lambda Functions in Python

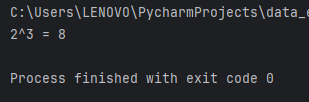
Lambda functions, also known as anonymous functions, are concise and one-liner functions defined using the lambda keyword in Python

power = lambda x, n: x\*\*n

result = power(2, 3)

print("2^3 =", result)

Output



Lambda functions are commonly used as arguments for higher-order functions like **map()**, **filter()**, and **reduce()** to perform operations on iterable objects efficiently.

**Filter()**

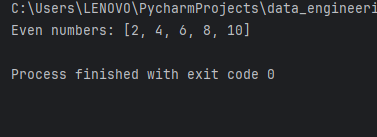
The filter() method filters the given sequence with the help of a function that tests each element in the sequence to be true or not.

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

even\_numbers = list(filter(lambda x: x % 2 == 0, numbers))

print("Even numbers:", even\_numbers)

Output



**Map()**

map() function returns a map object of the results after applying the given function to each item of a given iterable (list, tuple etc.)

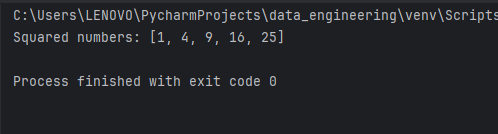
**# Using map() with lambda to square each element in a list**

**numbers = [1, 2, 3, 4, 5]**

**squared\_numbers = list(map(lambda x: x\*\*2, numbers))**

**print("Squared numbers:", squared\_numbers)**

Output



**Reduce()**

The reduce function is used to apply a particular function passed in its argument to all of the list elements mentioned in the sequence passed along.This function is defined in “functools” module.

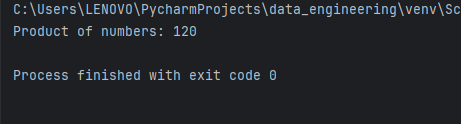
**numbers = [1, 2, 3, 4, 5]**

**# Using reduce with lambda to find the product of numbers**

**product = reduce(lambda x, y: x \* y, numbers)**

**print("Product of numbers:", product)**

Output



Que 6) Read JSON Strings to Python dicts or lists

Parse JSON string to a Python dictionary

import json

# JSON string representing a dictionary

json\_string\_dict = '{"name": "Alice", "age": 30, "city": "New York"}'

# Parse JSON string to a Python dictionary

python\_dict = json.loads(json\_string\_dict)

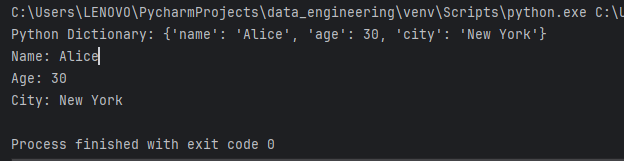
print("Python Dictionary:", python\_dict)

print("Name:", python\_dict['name'])

print("Age:", python\_dict['age'])

print("City:", python\_dict['city'])

Output



Parse JSON string to a Python List

# JSON string representing a list

json\_string\_list = '[1, 2, 3, 4, 5]'

# Parse JSON string to a Python list

python\_list = json.loads(json\_string\_list)

print("\nPython List:", python\_list)

print("Sum of List:", sum(python\_list))

Output

