**ASSIGNMENT: - 01**

**Problem Statement: -**

Perform the following operations using R/Python on suitable data sets:

a) read data from different formats (like csv, xls).

b) indexing and selecting data, sort data.

c) describe attributes of data, checking data types of each column.

d) counting unique values of data, format of each column, converting variable data type (e.g. from long to short, vice versa).

e) identifying missing values and fill in the missing values.

**S/W, Library, Package: -**

1. Software: Python
   * Python is a high-level programming language widely used in data science, machine learning, and scientific computing.
2. Library: pandas
   * pandas are a powerful data manipulation and analysis library for Python.
   * It provides data structures like Data Frame and Series, along with functions for reading/writing data, indexing, selecting, sorting, and handling missing values.
3. Packages:
   * For reading data from different formats:
     + CSV: No specific package needed; pandas can directly read CSV files using pd. read\_csv ().
     + Excel: openpyxl or xlrd packages might be required alongside pandas for reading Excel files (pd.read\_excel()).
   * For numerical operations and data manipulation:
     + pandas: Used extensively for indexing, selecting, sorting, describing data attributes, counting unique values, converting data types, and handling missing values.
   * For formatting and applying custom functions:
     + Python's built-in functions (lambda functions) are used along with pandas apply () method for custom formatting.
   * For statistical calculations and imputing missing values:
     + NumPy: Often used in conjunction with pandas for numerical operations and statistical calculations.
     + SciPy: Can be used for more advanced statistical functions if needed.

**Theory: -**

**a) Reading data from different formats:**

Methodology: - Python’s pandas library provides functions like read\_csv and read\_excel to read data from CSV and Excel files, respectively.

Advantage: - It's efficient and easy to use, especially for structured data in tabular formats.

Applications: - Data preprocessing, data analysis, machine learning, etc.

Limitations: - May not handle very large datasets efficiently in memory.

Example: - Reading a CSV file named data.csv into a Data Frame named df.

**b) Indexing, selecting, and sorting data:**

Methodology: Using panda’s methods like **loc** for indexing and selecting, and **sort\_values** for sorting.

Advantages: Allows for efficient data manipulation and analysis.

Applications: Data exploration, data transformation, data visualization.

Limitations: May require some learning curve to master advanced selection and sorting techniques.

Example: Selecting specific columns and sorting data based on a column.

**c) Describing attributes, checking data types:**

Methodology: Using pandas methods like **info**, **describe**, and **dtypes** to describe data attributes and check data types.

Advantages: Helps in understanding the structure and properties of the data.

Applications: Data profiling, data quality assessment, data validation.

Limitations: May not capture all nuances of complex data types automatically.

Example: Describing attributes and checking data types of columns in a DataFrame.

**d) Counting unique values, converting variable data types:**

Methodology: Using pandas methods like **nunique** to count unique values and **astype** to convert data types.

Advantages: Enables data type transformations and analysis of categorical variables.

Applications: Data preprocessing, data type conversion for analytics or modeling.

Limitations: May require handling missing or incompatible data before conversion.

Example: Counting unique values in a column and converting data type of another column.

**e) Identifying missing values and filling them:**

Methodology: Using pandas’ methods like **isnull** to identify missing values and **fillna** to fill them.

Advantages: Helps in handling missing data effectively.

Applications: Data preprocessing, data imputation, ensuring data integrity.

Limitations: Filling missing values with a constant may not be suitable for all cases; other imputation methods may be needed.

Example: Identifying missing values and filling them with a constant value (e.g., 0) in a DataFrame.

**Working/ Algorithm:**

1. read data from different formats (like csv, xls).

import pandas as pd

# import the data

heart= pd.read\_csv("/heart.csv")

heart.head()  # first 5 rows diaplay

b) indexing and selecting data, sort data.

# b) indexing and selecting data, sort data,

heart.head()  # first 5 rows diaplay

# indexing

heart["age"]

c) describe attributes of data, checking data types of each column.

heart.loc[[4,5]]  # shows perticular rows and col

heart.iloc[[2],[3]]  # shows perticular postion by using rows  and col

# c) describe attributes of data, checking data types of each column,

heart.tail() # show last 5 rows

d) counting unique values of data, format of each column, converting variable data type (e.g. from long to short, vice versa).

heart.info

heart.describe()

# Checking the format (data type) of each column

heart.dtypes

# counting unique values of data in a col.

heart['age'].nunique()

heart.nunique()

#converting variable data type (e.g.from long to short, vice versa),

heart['oldpeak'].astype(float)

heart.head()

1. identifying missing values and fill in the missing values.
2. # identifying missing values and fill in the missing values
3. heart.isnull().sum()

**Conclusion:**

The Python pandas library offers a comprehensive set of tools for data manipulation and analysis. Through operations like reading data from different formats, indexing, selecting, sorting, describing attributes, checking data types, counting unique values, and filling missing values, pandas enables efficient data handling and exploration. Its versatility and ease of use make it a go-to choice for data scientists and analysts working with structured data.