Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Practical based on Data Loading, Storage and File Formats	
Experiment No: 22	**Date:**	**Enrollment No: 92400133055**

GITHUB

Aim: Practical based on Data Loading, Storage and File Formats **IDE:**
Visual Studio Code
load, manipulate, and store data using Python (over reading and writing CSV, JSON, and Excel files)

Library Installation pip
install pandas openpyxl
Sample Data:
Create a folder for this experiment and add the following sample data files:
sample_data.csv (Name,Age,City
Alice,30,New York
Bob,25,Los Angeles
Charlie,35,Chicago)
sample_data.json ([
{"Name": "David", "Age": 28, "City": "San Francisco"},
{"Name": "Eve", "Age": 22, "City": "Seattle"}
])
Output:

```
CSV Data:
      Name  Age         City
0    Alice   30     New York
1      Bob   25  Los Angeles
2  Charlie   35      Chicago
```

sample_data.xlsx (you can create this using Excel with similar data**)**
Loading Data from CSV
Read the CSV file and perform basic data manipulation.
import pandas as pd

Load data from CSV csv_file_path
= 'sample_data.csv' df_csv =
pd.read_csv(csv_file_path)

# Display the DataFrame
print("CSV Data:")
print(df_csv) Output:

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Practical based on Data Loading, Storage and File Formats	
Experiment No: 22	**Date:**	**Enrollment No: 92400133055**

```
Filtered Data (Age > 30):
      Name  Age     City
2  Charlie   35  Chicago
```

Basic data manipulation: Filter by age

filtered_data = df_csv[df_csv['Age'] > 30]
print("\nFiltered Data (Age > 30):") print(filtered_data)
Loading Data from JSON
Read the JSON file and manipulate the data.
Load data from JSON json_file_path
= 'sample_data.json' df_json =
pd.read_json(json_file_path) output:

```
JSON Data:
      Name  Age           City
0  David   28  San Francisco
1    Eve   22        Seattle
```

Display the DataFrame print("\nJSON
Data:")
print(df_json)

# Basic data manipulation: Find the average age
average_age = df_json['Age'].mean() print("\nAverage
Age:", average_age)
Loading Data from Excel
Read the Excel file and display its contents.
Load data from Excel excel_file_path
= 'sample_data.xlsx'
df_excel = pd.read_excel(excel_file_path)

Display the DataFrame print("\nExcel
Data:")
print(df_excel)

Basic data manipulation: Count the number of entries entry_count
= df_excel.shape[0]
print("\nNumber of entries in Excel file:", entry_count)

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Practical based on Data Loading, Storage and File Formats	
Experiment No: 22	**Date:**	**Enrollment No: 92400133055**

Writing Data to Different Formats
Save manipulated DataFrames to new files in different formats.
# Save filtered CSV data to a new file
filtered_data.to_csv('filtered_data.csv', index=False)
print("\nFiltered data saved to 'filtered_data.csv'.") output:

```
Excel Data:
     Name  Age         City
0   Alice   30     New York
1     Bob   25  Los Angeles
2 Charlie   35      Chicago
```

# Save DataFrame to a new JSON file
df_json.to_json('new_data.json', orient='records', lines=True) print("JSON data saved to 'new_data.json'.")

# Save DataFrame to a new Excel file
df_excel.to_excel('new_data.xlsx', index=False) print("Excel data saved to 'new_data.xlsx'.")

Post Lab:
Write a code snippet to check the data types of each column in a DataFrame.
Code:

```
import pandas as pd data
= {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [30, 25, 35],
    'City': ['New York', 'Los Angeles', 'Chicago']
}
df = pd.DataFrame(data) print("Data
Types of Each Column:")
print(df.dtypes)
```

Output:

```
Data Types of Each Column:
Name      object
Age        int64
City      object
dtype: object
```

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Practical based on Data Loading, Storage and File Formats	
Experiment No: 22	**Date:**	**Enrollment No: 92400133055**

Write a code snippet that demonstrates how to fill missing values with the mean of a column.

Code:

```
import pandas as pd
import numpy as np

# Sample DataFrame with missing values data
= {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Age': [30, np.nan, 35, 28, np.nan],
    'City': ['New York', 'Los Angeles', 'Chicago', 'Houston', 'Seattle']

}
df = pd.DataFrame(data)

print("Original DataFrame:") print(df)

# Calculate the mean of the 'Age' column, ignoring NaN values mean_age
= df['Age'].mean()

# Fill missing values in the 'Age' column with the mean
df['Age'].fillna(mean_age, inplace=True)

print("\nDataFrame after filling missing values with the mean:") print(df)
```

Output:

```
Original DataFrame:
      Name   Age         City
0    Alice  30.0     New York
1      Bob   NaN  Los Angeles
2  Charlie  35.0      Chicago
3    David  28.0      Houston
4      Eve   NaN      Seattle

DataFrame after filling missing values with the mean:
      Name   Age         City
0    Alice  30.0     New York
1      Bob  30.75  Los Angeles
2  Charlie  35.0      Chicago
3    David  28.0      Houston
4      Eve  30.75      Seattle
```