LAB 12

ADVANCE PYTHON PROGRAMMING G REENA SRI - 22MID0009

Handling csv in Python

1. Read a CSV File

In [1]:
```
import pandas as pd

# Reading a CSV file
df = pd.read_csv("sample_data.csv")
print(df.head()) # accessing first 5 rows in csv file
```

```
   ID      Name  Age  Gender Department  Salary Joining_Date Performance_Score
0   1  Person_1   56  Female         HR  112800   2010-01-01                 C
1   2  Person_2   46  Female         IT   98685   2010-01-02                 A
2   3  Person_3   32    Male         HR   54660   2010-01-03                 D
3   4  Person_4   25  Female         HR  116064   2010-01-04                 A
4   5  Person_5   38    Male         HR   31168   2010-01-05                 A
```

2. Extract CSV into a Pandas DataFrame

In [2]:
```
# DataFrame is already created when reading CSV
df = pd.read_csv("sample_data.csv")

# Show basic info
print(df.info())
print(df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   ID                 3000 non-null   int64 
 1   Name               3000 non-null   object
 2   Age                3000 non-null   int64 
 3   Gender             3000 non-null   object
 4   Department         3000 non-null   object
 5   Salary             3000 non-null   int64 
 6   Joining_Date       3000 non-null   object
 7   Performance_Score  3000 non-null   object
dtypes: int64(3), object(5)
memory usage: 187.6+ KB
None
                ID          Age         Salary
count  3000.000000  3000.000000    3000.000000
mean   1500.500000    38.644000   74858.263333
std     866.169729    12.050104   26368.878797
min       1.000000    18.000000   30002.000000
25%     750.750000    28.000000   51357.250000
50%    1500.500000    39.000000   74661.000000
75%    2250.250000    49.000000   98136.000000
max    3000.000000    59.000000  119941.000000
```

In [3]:
```
#priting df
print(df)
```

```
        ID         Name  Age  Gender Department  Salary Joining_Date  \
0        1     Person_1   56  Female         HR  112800   2010-01-01
1        2     Person_2   46  Female         IT   98685   2010-01-02
2        3     Person_3   32    Male         HR   54660   2010-01-03
3        4     Person_4   25  Female         HR  116064   2010-01-04
4        5     Person_5   38    Male         HR   31168   2010-01-05
...    ...          ...  ...     ...        ...     ...          ...
2995  2996  Person_2996   36    Male  Marketing   83811   2018-03-15
2996  2997  Person_2997   59    Male    Finance   34001   2018-03-16
2997  2998  Person_2998   18    Male         HR   90848   2018-03-17
2998  2999  Person_2999   46  Female  Marketing   82635   2018-03-18
2999  3000  Person_3000   52    Male         HR   50946   2018-03-19

     Performance_Score
0                    C
1                    A
2                    D
3                    A
4                    A
...                ...
2995                 A
2996                 A
2997                 D
2998                 D
2999                 B

[3000 rows x 8 columns]
```

In [4]:
```
#extracting columns in a csv file
print(df.columns)
```

```
Index(['ID', 'Name', 'Age', 'Gender', 'Department', 'Salary', 'Joining_Date',
       'Performance_Score'],
      dtype='object')
```

In [5]:
```
#displaying the shape of dataset
print("Shape of the dataset: ",df.shape)
```

```
Shape of the dataset:  (3000, 8)
```

In [6]:
```
#printing values in a dataframe
print(df.values)
```

```
[[1 'Person_1' 56 ... 112800 '2010-01-01' 'C']
 [2 'Person_2' 46 ... 98685 '2010-01-02' 'A']
 [3 'Person_3' 32 ... 54660 '2010-01-03' 'D']
 ...
 [2998 'Person_2998' 18 ... 90848 '2018-03-17' 'D']
 [2999 'Person_2999' 46 ... 82635 '2018-03-18' 'D']
 [3000 'Person_3000' 52 ... 50946 '2018-03-19' 'B']]
```

EXCEPTION HANDLING WHILE EXTRACTING

In [7]:
```
#extracting the contents of csv into df using pandas and handling the exceptions
import pandas as pd
try:
    df=pd.read_csv("sample_data.csv") #reading csv file
    print(df.head()) #displaying top 5 rows
except FileNotFoundError:
    print("File not found at the location") # handling errors in a customised way
except Excption as e:
    print(f"Error occured: {e}") # displaying the error message
```

	ID	Name	Age	Gender	Department	Salary	Joining_Date	Performance_Score
0	1	Person_1	56	Female	HR	112800	2010-01-01	C
1	2	Person_2	46	Female	IT	98685	2010-01-02	A
2	3	Person_3	32	Male	HR	54660	2010-01-03	D
3	4	Person_4	25	Female	HR	116064	2010-01-04	A
4	5	Person_5	38	Male	HR	31168	2010-01-05	A

3. Append to a CSV

In [8]:
```
# Create new data to append
new_data = {
    "ID": [3001, 3002],
    "Name": ["Person_3001", "Person_3002"],
    "Age": [29, 35],
    "Gender": ["Male", "Female"],
```

```
    "Department": ["IT", "HR"],
    "Salary": [75000, 64000],
    "Joining_Date": ["2025-09-09", "2025-09-10"],
    "Performance_Score": ["B", "A"]
}

new_df = pd.DataFrame(new_data)

# Append to CSV
new_df.to_csv("sample_data.csv", mode="a", header=False, index=False)
```

In [9]:
```
#above code , new_data is type<dict> therefore , we explicitly convert them into dataframe.
```

In [11]:
```
# Get the last 2 rows (since we added 2 new rows)
new_rows = new_df.tail(2)
print(new_rows)
```

```
     ID         Name  Age  Gender Department  Salary Joining_Date  \
0  3001  Person_3001   29    Male         IT   75000   2025-09-09
1  3002  Person_3002   35  Female         HR   64000   2025-09-10

  Performance_Score
0                 B
1                 A
```

In [13]:
```
#Get rows by index position
new_rows = new_df.iloc[-2:]  # last 2 rows
print(new_rows)
```

```
     ID         Name  Age  Gender Department  Salary Joining_Date  \
0  3001  Person_3001   29    Male         IT   75000   2025-09-09
1  3002  Person_3002   35  Female         HR   64000   2025-09-10

  Performance_Score
0                 B
1                 A
```

In [14]:
```
#Read only the newly added rows by their IDs
access_rows = df[df["ID"].isin([1001, 1002])]
print(access_rows)
```

```
        ID         Name  Age  Gender Department  Salary Joining_Date  \
1000  1001  Person_1001   46  Female    Finance   43237   2012-09-27
1001  1002  Person_1002   21  Female  Marketing   81586   2012-09-28

     Performance_Score
1000                 D
1001                 D
```

4. Read a CSV Chunk-by-chunk

In [15]:
```
# Read CSV in chunks of 500 rows
chunk_size = 500
for chunk in pd.read_csv("sample_data.csv", chunksize=chunk_size):
    print(chunk.head(2))  # Show first 2 rows of each chunk
```

```
   ID      Name  Age  Gender Department  Salary Joining_Date Performance_Score
0   1  Person_1   56  Female         HR  112800   2010-01-01                 C
1   2  Person_2   46  Female         IT   98685   2010-01-02                 A
      ID        Name  Age  Gender Department  Salary Joining_Date  \
500  501  Person_501   54    Male         HR   96617   2011-05-16
501  502  Person_502   29  Female         HR   50269   2011-05-17

    Performance_Score
500                 B
501                 B
        ID         Name  Age  Gender Department  Salary Joining_Date  \
1000  1001  Person_1001   46  Female    Finance   43237   2012-09-27
1001  1002  Person_1002   21  Female  Marketing   81586   2012-09-28

     Performance_Score
1000                 D
1001                 D
        ID         Name  Age  Gender Department  Salary Joining_Date  \
1500  1501  Person_1501   19    Male    Finance   47459   2014-02-09
1501  1502  Person_1502   31  Female    Finance   65418   2014-02-10

     Performance_Score
1500                 A
1501                 D
        ID         Name  Age  Gender Department  Salary Joining_Date  \
2000  2001  Person_2001   48  Female         HR   58549   2015-06-24
2001  2002  Person_2002   27  Female         IT  111782   2015-06-25

     Performance_Score
2000                 C
2001                 D
        ID         Name  Age  Gender Department  Salary Joining_Date  \
2500  2501  Person_2501   26  Female         IT   43746   2016-11-05
2501  2502  Person_2502   53  Female  Marketing  116457   2016-11-06

     Performance_Score
2500                 D
2501                 D
        ID         Name  Age  Gender Department  Salary Joining_Date  \
3000  3001  Person_3001   29    Male         IT   75000   2025-09-09
3001  3002  Person_3002   35  Female         HR   64000   2025-09-10
```

```
     Performance_Score
3000                 B
3001                 A
```

ALTERNATIVE METHOD TO READ CSV , USING FILE OBJECT AND ACCESSING THE FILE CHUNK WISE

In [19]:
```python
#reading csv chunk by chunk

with pd.read_csv("sample_data.csv",chunksize=500) as reader:
    print(reader) # printing file obj
    for c in reader:
        print(c.head(2)) #accessing first 2 rows of each chunk
```

```
<pandas.io.parsers.readers.TextFileReader object at 0x0000019D1A762950>
   ID      Name  Age  Gender Department  Salary Joining_Date Performance_Score
0   1  Person_1   56  Female         HR  112800   2010-01-01                 C
1   2  Person_2   46  Female         IT   98685   2010-01-02                 A
      ID        Name  Age  Gender Department  Salary Joining_Date  \
500  501  Person_501   54    Male         HR   96617   2011-05-16
501  502  Person_502   29  Female         HR   50269   2011-05-17

    Performance_Score
500                 B
501                 B
        ID         Name  Age  Gender Department  Salary Joining_Date  \
1000  1001  Person_1001   46  Female    Finance   43237   2012-09-27
1001  1002  Person_1002   21  Female  Marketing   81586   2012-09-28

     Performance_Score
1000                 D
1001                 D
        ID         Name  Age  Gender Department  Salary Joining_Date  \
1500  1501  Person_1501   19    Male    Finance   47459   2014-02-09
1501  1502  Person_1502   31  Female    Finance   65418   2014-02-10

     Performance_Score
1500                 A
1501                 D
        ID         Name  Age  Gender Department  Salary Joining_Date  \
2000  2001  Person_2001   48  Female         HR   58549   2015-06-24
2001  2002  Person_2002   27  Female         IT  111782   2015-06-25

     Performance_Score
2000                 C
2001                 D
        ID         Name  Age  Gender Department  Salary Joining_Date  \
2500  2501  Person_2501   26  Female         IT   43746   2016-11-05
2501  2502  Person_2502   53  Female  Marketing  116457   2016-11-06

     Performance_Score
2500                 D
2501                 D
        ID         Name  Age  Gender Department  Salary Joining_Date  \
3000  3001  Person_3001   29    Male         IT   75000   2025-09-09
```

```
3001  3002  Person_3002   35  Female         HR   64000   2025-09-10

    Performance_Score
3000                 B
3001                 A
```

In [20]:
```
print(type(reader))
print(type(c))
```

```
<class 'pandas.io.parsers.readers.TextFileReader'>
<class 'pandas.core.frame.DataFrame'>
```

In [21]:
```
#c is the dataframe
#reader is the iterator
#using (with ...as) is the context manager , that makes sure that the file is properly closed.
```

Note :

- Appending with to_csv(mode="a") → just adds new rows at the end of the CSV file.
- .iloc → is a read-only indexing operation. It only helps you access rows by their position (like slicing in lists). It never modifies or replaces rows unless you explicitly assign values.
- What if you do assignment?

```
df.iloc[10, df.columns.get_loc("Salary")] = 99999
```

- This updates the value in memory (the DataFrame), but the CSV file itself won't change unless you explicitly save it again with:

```
df.to_csv("sample_data.csv", index=False)
```

- That index=False you see in to_csv() is just an instruction about whether Pandas should write the DataFrame's index column into the CSV file.

In [23]:
```
with pd.read_csv("sample_data.csv",chunksize=500) as reader:
    print( reader.get_chunk(5)) #reading through specific chunk of data
```

```
   ID      Name  Age  Gender Department  Salary Joining_Date Performance_Score
0   1  Person_1   56  Female         HR  112800   2010-01-01                 C
1   2  Person_2   46  Female         IT   98685   2010-01-02                 A
2   3  Person_3   32    Male         HR   54660   2010-01-03                 D
3   4  Person_4   25  Female         HR  116064   2010-01-04                 A
4   5  Person_5   38    Male         HR   31168   2010-01-05                 A
```

Example: Reading in Chunks and Concatenating

In [24]:
```python
import pandas as pd

chunk_size = 500
chunks = []

# Read CSV in chunks
for chunk in pd.read_csv("sample_data.csv", chunksize=chunk_size):
    # You can process each chunk here if needed
    chunks.append(chunk)

# Concatenate all chunks into a single DataFrame
full_df = pd.concat(chunks, ignore_index=True)

print(full_df.shape)   # (3002, 8) if we had appended 2 new rows earlier
print(full_df.head())
```

```
(3002, 8)
   ID      Name  Age  Gender Department  Salary Joining_Date Performance_Score
0   1  Person_1   56  Female         HR  112800   2010-01-01                 C
1   2  Person_2   46  Female         IT   98685   2010-01-02                 A
2   3  Person_3   32    Male         HR   54660   2010-01-03                 D
3   4  Person_4   25  Female         HR  116064   2010-01-04                 A
4   5  Person_5   38    Male         HR   31168   2010-01-05                 A
```

Notes:

Why concat is needed here?

- Each chunk is a separate DataFrame.

- concat() stitches them back together (row-wise).
- Without concat, you'd just have a list of DataFrames instead of one complete DataFrame.

So:

- pd.concat() = "merge/join multiple DataFrames or Series"
- pd.DataFrame() = "convert raw data (list, dict, array) into a DataFrame"

Using Enumerate function to read the csv chunk by chunk

In [25]:
```
import pandas as pd

chunk_size = 500

for i, chunk in enumerate(pd.read_csv("sample_data.csv", chunksize=chunk_size)): #using enumerate function
    print(f"\n--- Chunk {i+1} ---")
    print(chunk.head(2))   # Show first 2 rows of this chunk
```

```
--- Chunk 1 ---
   ID      Name  Age  Gender Department  Salary Joining_Date Performance_Score
0   1  Person_1   56  Female         HR  112800   2010-01-01                 C
1   2  Person_2   46  Female         IT   98685   2010-01-02                 A

--- Chunk 2 ---
      ID        Name  Age  Gender Department  Salary Joining_Date  \
500  501  Person_501   54    Male         HR   96617   2011-05-16
501  502  Person_502   29  Female         HR   50269   2011-05-17

    Performance_Score
500                 B
501                 B

--- Chunk 3 ---
        ID         Name  Age  Gender Department  Salary Joining_Date  \
1000  1001  Person_1001   46  Female    Finance   43237   2012-09-27
1001  1002  Person_1002   21  Female  Marketing   81586   2012-09-28

     Performance_Score
1000                 D
1001                 D

--- Chunk 4 ---
        ID         Name  Age  Gender Department  Salary Joining_Date  \
1500  1501  Person_1501   19    Male    Finance   47459   2014-02-09
1501  1502  Person_1502   31  Female    Finance   65418   2014-02-10

     Performance_Score
1500                 A
1501                 D

--- Chunk 5 ---
        ID         Name  Age  Gender Department  Salary Joining_Date  \
2000  2001  Person_2001   48  Female         HR   58549   2015-06-24
2001  2002  Person_2002   27  Female         IT  111782   2015-06-25

     Performance_Score
2000                 C
2001                 D
```

```
--- Chunk 6 ---
        ID         Name  Age  Gender Department  Salary Joining_Date  \
2500  2501  Person_2501   26  Female         IT   43746   2016-11-05
2501  2502  Person_2502   53  Female  Marketing  116457   2016-11-06

     Performance_Score
2500                 D
2501                 D

--- Chunk 7 ---
        ID         Name  Age  Gender Department  Salary Joining_Date  \
3000  3001  Person_3001   29    Male         IT   75000   2025-09-09
3001  3002  Person_3002   35  Female         HR   64000   2025-09-10

     Performance_Score
3000                 B
3001                 A
```

EXPLANATION:

- pd.read_csv(..., chunksize=500) → returns an iterator of DataFrames.
- enumerate() → gives you (index, chunk) while looping.
- i+1 → human-readable chunk number (since enumerate starts at 0).

5. Write Numeric Data into CSV

In [26]:
```python
import pandas as pd
import numpy as np

# Create numeric dataset
numeric_data = pd.DataFrame({
    "ID": np.arange(1, 11),  # 1 to 10
    "Value1": np.random.randint(10, 100, size=10),
    "Value2": np.random.uniform(1.5, 9.9, size=10).round(2)
})
```

```
# Save numeric data into CSV
numeric_data.to_csv("numeric_data.csv", index=False)

print("Numeric CSV written successfully!")
print(numeric_data)
```

```
Numeric CSV written successfully!
   ID  Value1  Value2
0   1      71    7.92
1   2      16    1.77
2   3      29    8.48
3   4      24    7.57
4   5      80    2.31
5   6      13    6.09
6   7      67    8.73
7   8      17    7.19
8   9      90    8.50
9  10      21    2.63
```

6. Write Text Data into CSV

In [27]:
```
# Create text dataset
text_data = pd.DataFrame({
    "Name": ["Alice", "Bob", "Charlie", "Diana", "Ethan"],
    "City": ["New York", "London", "Tokyo", "Paris", "Berlin"],
    "Department": ["HR", "IT", "Finance", "Marketing", "Sales"]
})

# Save text data into CSV
text_data.to_csv("text_data.csv", index=False)

print("Text CSV written successfully!")
print(text_data)
```

```
Text CSV written successfully!
      Name      City Department
0    Alice  New York         HR
1      Bob    London         IT
2  Charlie     Tokyo    Finance
3    Diana     Paris  Marketing
4    Ethan    Berlin      Sales
```

In [ ]: