



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No 12:

Aim: Program to demonstrate DataFrame using Pandas

Theory:

Pandas DataFrame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the data, rows, and columns.

Creating an empty dataframe :

A basic DataFrame, which can be created is an Empty Dataframe. An Empty Dataframe is created just by calling a dataframe constructor.

```
# import pandas as pd
import pandas as pd

# Calling DataFrame constructor
df = pd.DataFrame()

print(df)
```

Output :

```
Empty DataFrame
Columns: []
Index: []
```

Creating a dataframe using List:

DataFrame can be created using a single list or a list of lists.

```
# import pandas as pd
import pandas as pd

# list of strings
lst = ['Geeks', 'For', 'Geeks', 'is',
       'portal', 'for', 'Geeks']

# Calling DataFrame constructor on list
df = pd.DataFrame(lst)
print(df)
```



Creating DataFrame from dict of ndarray/lists:

To create DataFrame from dict of ndarray/list, all the ndarray must be of same length. If index is passed then the length index should be equal to the length of arrays. If no index is passed, then by default, index will be range(n) where n is the array length.

```
# Python code demonstrate creating  
# DataFrame from dict ndarray / lists  
# By default addresses.
```

```
import pandas as pd
```

```
# initialise data of lists.  
data = {'Name':['Tom', 'nick', 'krish', 'jack'], 'Age':[20, 21, 19, 18]}
```

```
# Create DataFrame  
df = pd.DataFrame(data)
```

```
# Print the output.  
print(df)
```

Create pandas dataframe from lists using dictionary:

Creating pandas data-frame from lists using dictionary can be achieved in different ways. We can create pandas dataframe from lists using dictionary using pandas.DataFrame. With this method in Pandas we can transform a dictionary of list to a dataframe.

```
# importing pandas as pd  
import pandas as pd  
  
# dictionary of lists  
dict = {'name':['aparna', 'pankaj', 'sudhir', 'Geeku'],  
        'degree': ["MBA", "BCA", "M.Tech", "MBA"],  
        'score':[90, 40, 80, 98]}  
  
df = pd.DataFrame(dict)  
  
print(df)
```

Dataframe methods

Few methods of Dataframe are mentioned below:

1. Pandas **head()** method is used to return top n (5 by default) rows of a data frame or series.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

2. Pandas **describe()** is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values.
3. Pandas **tail()** method is used to return bottom n (5 by default) rows of a data frame or series
4. **query()**: Pandas provide many methods to filter a Data frame and **Dataframe.query()** is one of them.
5. Pandas provide a unique method to retrieve rows from a Data frame. **DataFrame.loc[]** method is used to retrieve rows from Pandas DataFrame. Rows can also be selected by passing integer location to an **iloc[]** function.
6. **drop()** method is used to delete columns or rows of a dataframe.

PROGRAM:

Program 12.1: Program to query dataframe

```
import pandas as pd

# Sample DataFrame
data = {'Name': ['John', 'Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35, 40],
        'City': ['New York', 'Los Angeles', 'Chicago', 'Houston']}
df = pd.DataFrame(data)

# Displaying the DataFrame
print("Original DataFrame:")
print(df)
print()

# Querying the DataFrame
# Example 1: Filtering based on a condition
print("Query 1: People older than 30:")
query_result_1 = df[df['Age'] > 30]
print(query_result_1)
print()

# Example 2: Selecting specific columns
print("Query 2: Selecting Name and City columns:")
query_result_2 = df[['Name', 'City']]
print(query_result_2)
print()

# Example 3: Using multiple conditions
print("Query 3: People aged 30 and living in Chicago:")
query_result_3 = df[(df['Age'] == 30) & (df['City'] == 'Chicago')]
print(query_result_3)
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Output:

```
PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\dataframe.py"
Original DataFrame:
   Name  Age  City
0  John   25  New York
1  Alice  30  Los Angeles
2   Bob   35  Chicago
3 Charlie  40  Houston

Query 1: People older than 30:
   Name  Age  City
2   Bob   35  Chicago
3 Charlie  40  Houston

Query 2: Selecting Name and City columns:
   Name  City
0  John  New York
1  Alice  Los Angeles
2   Bob   Chicago
3 Charlie  Houston

Query 3: People aged 30 and living in Chicago:
Empty DataFrame
Columns: [Name, Age, City]
Index: []
PS C:\Users\Lenovo\Downloads\Python Prgs>
```

Conclusion:

The experiment successfully showcased the versatility and efficiency of Pandas DataFrame for data manipulation and analysis. Through its intuitive functionality and comprehensive features, Pandas proved to be an indispensable tool for handling structured data, offering researchers and analysts powerful capabilities for exploring and visualizing datasets withease.