Contents

8. PANDAS – DATAFRAME – ATTRIBUTES

1. DataFrame Attributes

- ✓ DataFrame is a predefined class.
- ✓ DataFrame having different attributes.
- ✓ These attributes return information about the DataFrame object.

1.1. Length of DataFrame

- ✓ We can check length of DataFrame by using len(p) function
- ✓ This function returns the total number of rows from the DataFrame

Program Name: Checking total number of rows/length from the DataFrame
Input file: demo1.py
sales1.csv

```
import pandas as pd

df = pd.read_csv("sales1.csv")
print("Total number of rows in DataFrame:", len(df))
```

Output

```
Total number of rows in DataFrame: 600
```

1.2. columns

- ✓ columns is predefined attribute in DataFrame class.
- ✓ We can access columns attribute by using DataFrame object.
- ✓ This attribute returns all column names from the DataFrame

Program Accessing columns attribute from DataFrame.
Name demo2.py
Input file sales1.csv

```
import pandas as pd

df = pd.read_csv("sales1.csv")
print(df.columns)
```

Output

```
Index(['Order ID', 'Customer Name', 'Product', 'Quantity'],
dtype='object')
```

1.3. shape

- ✓ shape is predefined attribute in DataFrame class.
- ✓ We can access shape attribute by using DataFrame object.
- ✓ This attribute returns the total number of rows and columns in tuple format.
    - From the tuple, first value represents total number of rows
    - From the tuple, second value represents total number of columns

Program Name: Accessing shape attribute from DataFrame.
Input file: demo3.py
sales1.csv

```
import pandas as pd

df = pd.read_csv("sales1.csv")
print(df.shape)
```

Output

```
(600, 4)
```

1.4. shape[0]

- ✓ shape is predefined attribute in DataFrame class.
- ✓ We can access shape attribute by using DataFrame object.
- ✓ This attribute returns the total number of rows and columns in tuple
- ✓ Shape[0] returns total number for rows from the DataFrame

Program Name: Accessing shape attribute and checking total number of rows
Input file: demo4.py
sales1.csv

```
import pandas as pd

df = pd.read_csv("sales1.csv")
print(df.shape[0])
```

Output

```
600
```

1.5. shape[1]

- ✓ shape is predefined attribute in DataFrame class.
- ✓ We can access shape attribute by using DataFrame object.
- ✓ This attribute returns the total number of rows and columns in tuple
- ✓ Shape[1] returns total number for columns from the DataFrame

Program Accessing shape attribute and checking total number of columns
Name demo5.py
Input file sales1.csv

```
import pandas as pd

df = pd.read_csv("sales1.csv")
print(df.shape[1])
```

Output

```
4
```

1.6. size

- ✓ size is predefined attribute in DataFrame class.
- ✓ We can access size attribute by using DataFrame object.
- ✓ This attribute returns the total number of elements/values in DataFrame

Program Name: demo6.py
Input file: sales1.csv

Accessing total number of elements/values from DataFrame.

```
import pandas as pd

df = pd.read_csv("sales1.csv")
print(df.size)
```

Output

```
2400
```

Make a note:

- ✓ size = Number of rows **X** Number of columns
- ✓ size = Row_count **X** Column_count

Program Name: Accessing total number of elements/values from DataFrame.
demo7.py
Input file: sales1.csv

```
import pandas as pd

df = pd.read_csv("sales1.csv")
print("Number of elements in DataFrame:", df.size)
print("Number of elements in DataFrame:",
df.shape[0]*df.shape[1])
```

Output

```
Number of elements in DataFrame: 2400
Number of elements in DataFrame: 2400
```

1.7. dtypes

- ✓ dtypes is predefined attribute in DataFrame class.
- ✓ We can access dtypes attribute by using DataFrame object.
- ✓ This attribute returns the datatype of each column.

Program Name: Checking all columns datatype from DataFrame
Input file: demo8.py
sales1.csv

```
import pandas as pd

df = pd.read_csv("sales1.csv")
print(df.dtypes)
```

Output

```
Order ID          int64
Customer Name    object
Product          object
Quantity          int64
dtype: object
```

1.8. empty

- ✓ empty is predefined attribute in DataFrame class.
- ✓ We can access empty attribute by using DataFrame object.
- ✓ This attribute check DataFrame is empty or not,
  - If DataFrame is empty then it returns True other False.

Program Checking DataFrame empty or not
Name demo9.py
Input file sales1.csv

```
import pandas as pd

df = pd.read_csv("sales1.csv")
print(df.empty)
```

Output

```
False
```

Program Name: Checking DataFrame empty or not
Input file: demo10.py
sales1.csv

```
import pandas as pd

df = pd.DataFrame()

print(df)
print()
print(df.empty)
```

Output

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

True
```

1.9. index

- ✓ index is predefined attribute in DataFrame class.
- ✓ We can access index attribute by using DataFrame object.
- ✓ This attribute return index start and end value from the DataFrame.

Program Name: Accessing index attribute from DataFrame
Name: demo11.py
Input file: sales1.csv

```
import pandas as pd

df = pd.read_csv("sales1.csv")
print(df.index)
```

Output

```
RangeIndex(start=0, stop=600, step=1)
```

1.10. values

- ✓ values is predefined attribute in DataFrame class.
- ✓ We can access values attribute by using DataFrame object.
- ✓ This attribute return values of DataFrame,
  - Each row values in one array from starting to last row.

Program Name: Accessing values attribute from DataFrame
Input file: demo12.py
sales1.csv

```
import pandas as pd

df = pd.read_csv("sales1.csv")
print(df.values)
```

Output

```
[[166837 'Veeru' '34in Ultrawide Monitor' 2]
 [166838 'Tarun' 'Samsung m10' 3]
 [166839 'Kedar' '20in Monitor' 1]
 ...
 [167405 'Venu' 'Flatscreen TV' 1]
 [167406 'Siddhu' 'Samsung m20' 2]
 [167407 'Tarun' 'LG Washing Machine' 1]]
```

1.11. T

- ✓ T is predefined attribute in DataFrame class.
- ✓ We can access **T** attribute by using DataFrame object.
- ✓ T means its transpose, it returns rows as columns and columns as rows

Program Name: Converting rows as column and columns as rows from DataFrame
demo13.py

```
import pandas as pd

details = [["Sagar", 20, 10000], ["Daniel", 16, 20000], ["Veeru", 24, 30000], ["Raju", 25, 40000], ["Kiran", 26, 50000], ["Kedar", 27, 60000], ["Reena", 28, 70000]]

df = pd.DataFrame(details, columns = ["Name", "Age", "Salary"])

print(df)
print()
print(df.T)
```

Output

```
     Name  Age  Salary
0   Sagar   20   10000
1  Daniel   16   20000
2   Veeru   24   30000
3    Raju   25   40000
4   Kiran   26   50000
5   Kedar   27   60000
6   Reena   28   70000

            0       1       2       3       4       5       6
Name    Sagar  Daniel  Veeru   Raju  Kiran  Kedar  Reena
Age        20      16      24      25      26      27      28
Salary  10000   20000  30000  40000  50000  60000  70000
```