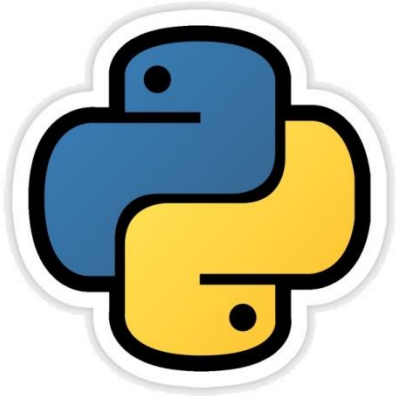
Advanced Operations on DataFrames



Based on CBSE Curriculum

Informatics Practices, Class-12

CHAPTER-2

By:

Mrs. Neha Tyagi (PGT CS)

KV No-5 2nd Shift, Jaipur

KVS RO Jaipur

Pivoting DataFrame

- **Pandas is a popular library for *Data analysis .***
- ***Pivoting*** is one of the key actions for a Data–Analyst. Means providing an axis to the table data, on the basis of that axis the database will work.
- Using Pandas, MS-Excel type of pivot tables can be created.
- These tables ***summarizes*** the big data and create meaningful reports to save your time.
- Pivot table allows us to fetch important record from a large and detailed data set.
- Pivot tables can automatically sort, count and total etc.
- In general, pivoting means to use unique value from a index/column and make dataframe.
- To make pivot table we use ***pivot( )*** or ***pivot_table()*** from pandas.

Pivoting using pivot( ) method

- pivot() method, creates new DataFrame after reshaping the data on the basis of column values.
- This method takes 3 arguments - ***index***, ***columns*** and ***values*** . Minimum two arguments are compulsory.
- In the form of arguments value you have to pass column name of original table.
- Then pivot ( ) creates a new table whose indices of row and column are the same which you have given as argument.
- Cell values of new table will come from the column which you have given as parameter. Its syntax is -

pandas.pivot(index, columns, values)

- Where ***index*** creates a index of new DataFrame, which is the column name from the table.
- Where ***columns*** creates columns of new DataFrame, which are the names of column of table.
- Where ***values*** creates columns of new DataFrame which are the values of the column name from table.

Pivoting using pivot( ) method

syntax → pandas.pivot(index, columns, values)

- Example→ Creating DataFrame

```
>>> df
      Name Subject  Score Grade
0  Pratibha      CS     99    A1
1    Ritika     Phy     87    A2
2    Saumya    Chem     88    A2
3     Aryan   Maths     67     B
```

```
import pandas as pd
ClassXII={'Name':['Pratibha','Ritika','Saumya','Aryan'],\
          'Subject':['CS','Phy','Chem','Maths'],\
          'Score':[99,87,88,67],\
          'Grade':['A1','A2','A2','B']}
df=pd.DataFrame(ClassXII,columns=['Name','Subject','Score','Grade'])
```

- Creating pivot table →

```
>>> pv=df.pivot(index='Name',columns='Subject',values='Score')
>>> pv
Subject      CS   Chem  Maths   Phy
Name
Aryan       NaN    NaN   67.0   NaN
Pratibha   99.0    NaN    NaN   NaN
Ritika      NaN    NaN    NaN  87.0
Saumya      NaN   88.0    NaN   NaN
```

We can see in this pivot table that there is a new table is created and the values of Score column came in to different columns. While its Name and Subject column, is matching with original table. Where values are not matching, NaN (None) is putted automatically.

Using pivot( ) method with .fillna( )

syntax → pandas.pivot(index, columns, values).fillna()

- Example→ Creating DataFrame.

```
>>> df
      Name Subject  Score Grade
0  Pratibha      CS     99    A1
1    Ritika     Phy     87    A2
2    Saumya    Chem     88    A2
3     Aryan   Maths     67     B
```

```
import pandas as pd
ClassXII={'Name':['Pratibha','Ritika','Saumya','Aryan'],\
          'Subject':['CS','Phy','Chem','Maths'],\
          'Score':[99,87,88,67],\
          'Grade':['A1','A2','A2','B']}
df=pd.DataFrame(ClassXII,columns=['Name','Subject','Score','Grade'])
```

- Creating pivot table with .fillna()→

```
>>> pv=df.pivot(index='Name',columns='Subject',values='Score').fillna('')
>>> pv
Subject    CS Chem Maths Phy
Name
Aryan                  67
Pratibha   99
Ritika                    87
Saumya          88
```

We can see in this pivot table that there is a new table is created and the values of Score column came in to different columns. While its Name and Subject column, is matching with original table. Where values are not matching, there is a blank space in spite of NaN.

Pivoting by Multiple columns

We remove ***values*** parameter from syntax only.

syntax → pandas.pivot(index, columns)

- Example → Creating DataFrame

```
>>> df
      Name Subject  Score Grade
0  Pratibha      CS     99    A1
1    Ritika     Phy     87    A2
2    Saumya    Chem     88    A2
3     Aryan   Maths     67     B
```

```
import pandas as pd
ClassXII={'Name':['Pratibha','Ritika','Saumya','Aryan'],\
          'Subject':['CS','Phy','Chem','Maths'],\
          'Score':[99,87,88,67],\
          'Grade':['A1','A2','A2','B']}
df=pd.DataFrame(ClassXII,columns=['Name','Subject','Score','Grade'])
```

- Ceating pivot table with .fillna( ) →

```
>>> pv=df.pivot(index='Name',columns='Subject')
>>> pv
          Score                        Grade
Subject      CS  Chem Maths   Phy     CS Chem Maths  Phy
Name
Aryan       NaN   NaN  67.0   NaN    NaN  NaN     B  NaN
Pratibha   99.0   NaN   NaN   NaN     A1  NaN   NaN  NaN
Ritika      NaN   NaN   NaN  87.0    NaN  NaN   NaN   A2
Saumya      NaN  88.0   NaN   NaN    NaN   A2   NaN  NaN
```

Pivoting by Multiple columns. . .

- In last example we have seen that there are many indices created and their values were seen once for subjects and once for grades for each name.
- We can filter them→

```
>>> pv.Score.fillna('')
Subject    CS Chem Maths Phy
Name
Aryan                  67
Pratibha   99
Ritika                    87
Saumya            88
```

```
>>> pv.Score.CS.fillna('')
Name
Aryan
Pratibha    99
Ritika
Saumya
Name: CS, dtype: object
```

```
>>> pv.Grade.CS.fillna('')
Name
Aryan
Pratibha    A1
Ritika
Saumya
Name: CS, dtype: object
```

```
>>> pv.Grade.Maths.fillna('')
Name
Aryan       B
Pratibha
Ritika
Saumya
Name: Maths, dtype: object
```

Pivot Problem

- We should always remember that if there are combinations of multiple values in indices and columns then a value error will occur.

```
import pandas as pd
ClassXII={'Name':['Pratibha','Ritika','Saumya','Aryan','Pratibha'],\
          'Subject':['CS','Phy','Chem','Maths','CS'],\
          'Score':[99,87,88,67,98],\
          'Grade':['A1','A2','A2','B','A1']}
df=pd.DataFrame(ClassXII,columns=['Name','Subject','Score','Grade'])
```

```
>>> df
      Name Subject  Score Grade
0  Pratibha      CS     99    A1
1    Ritika     Phy     87    A2
2    Saumya    Chem     88    A2
3     Aryan   Maths     67     B
4  Pratibha      CS     98    A1
>>> pv=df.pivot(index='Name',columns='Subject',values='Score')
Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    pv=df.pivot(index='Name',columns='Subject',values='Score')
  File "C:\Users\KVBBKServer\AppData\Local\Programs\Python\Python36\lib\site-packag
es\pandas\core\frame.py", line 5194, in pivot
    return pivot(self, index=index, columns=columns, values=values)
  File "C:\Users\KVBBKServer\AppData\Local\Programs\Python\Python36\lib\site-packag
es\pandas\core\reshape\reshape.py", line 415, in pivot
    return indexed.unstack(columns)
ValueError: Index contains duplicate entries, cannot reshape
```

Using stack( ) and unstack( ) methods

- stack( ) and unstack( ) methods both flip the layout of DataFrame, means these flips the levels of columns into row and flips levels of rows into columns. DataFrame*stacking* means moving the innermost column index to innermost row index and the opposite action is know as *unstacking*

```
import pandas as pd
ClassXII={'Name':['Pratibha','Ritika','Saumya','Aryan'],\
          'Subject':['CS','Phy','Chem','Maths'],\
          'Score':[99,87,88,67],\
          'Grade':['A1','A2','A2','B']}
df=pd.DataFrame(ClassXII,columns=['Name','Subject','Score','Grade'])
```

```
>>> pv=df.pivot(index='Name',columns='Subject')
>>> pv
        Score                       Grade
Subject    CS  Chem Maths   Phy     CS Chem Maths  Phy
Name
Aryan     NaN   NaN  67.0   NaN    NaN  NaN     B  NaN
Pratibha 99.0   NaN   NaN   NaN     A1  NaN   NaN  NaN
Ritika    NaN   NaN   NaN  87.0    NaN  NaN   NaN   A2
Saumya    NaN  88.0   NaN   NaN    NaN   A2   NaN  NaN
```

After using Stack Method all the horrizontal became vertical and it takes last level in column breakdown and converts it into last row breakdown.

Using Stack ( ) Method

```
>>> pv.stack()
                 Score Grade
Name     Subject
Aryan    Maths    67.0     B
Pratibha CS       99.0    A1
Ritika   Phy      87.0    A2
Saumya   Chem     88.0    A2
```

Using stack( ) and unstack( ) methods. . .

```
import pandas as pd
ClassXII={'Name':['Pratibha','Ritika','Saumya','Aryan'],\
          'Subject':['CS','Phy','Chem','Maths'],\
          'Score':[99,87,88,67],\
          'Grade':['A1','A2','A2','B']}
df=pd.DataFrame(ClassXII,columns=['Name','Subject','Score','Grade'])
```

```
>>> pv=df.pivot(index='Name',columns='Subject')
>>> pv
        Score                        Grade
Subject    CS  Chem Maths   Phy    CS Chem Maths  Phy
Name
Aryan     NaN   NaN  67.0   NaN   NaN  NaN     B  NaN
Pratibha 99.0   NaN   NaN   NaN    A1  NaN   NaN  NaN
Ritika    NaN   NaN   NaN  87.0   NaN  NaN   NaN   A2
Saumya    NaN  88.0   NaN   NaN   NaN   A2   NaN  NaN
```

```
>>> pv.stack()
                  Score Grade
Name     Subject
Aryan    Maths     67.0     B
Pratibha CS        99.0    A1
Ritika   Phy       87.0    A2
Saumya   Chem      88.0    A2
```

```
>>> pv.stack().stack()
Name     Subject
Aryan    Maths    Score    67
                  Grade     B
Pratibha CS       Score    99
                  Grade    A1
Ritika   Phy      Score    87
                  Grade    A2
Saumya   Chem     Score    88
                  Grade    A2
dtype: object
```

Stack can be used like this. After stacking there is another stacking, then it moves all the remaining levels.

Using stack( ) and unstack( ) methods. . .

```
import pandas as pd
ClassXII={'Name':['Pratibha','Ritika','Saumya','Aryan'],\
          'Subject':['CS','Phy','Chem','Maths'],\
          'Score':[99,87,88,67],\
          'Grade':['A1','A2','A2','B']}
df=pd.DataFrame(ClassXII,columns=['Name','Subject','Score','Grade'])
```

```
>>> pv=df.pivot(index='Name',columns='Subject')
>>> pv
        Score                         Grade
Subject    CS  Chem Maths   Phy     CS Chem Maths  Phy
Name
Aryan     NaN   NaN  67.0   NaN    NaN  NaN     B  NaN
Pratibha 99.0   NaN   NaN   NaN     A1  NaN   NaN  NaN
Ritika    NaN   NaN   NaN  87.0    NaN  NaN   NaN   A2
Saumya    NaN  88.0   NaN   NaN    NaN   A2   NaN  NaN
```

```
>>> pv.stack(0)
Subject           CS Chem Maths  Phy
Name
Aryan    Grade   NaN  NaN     B  NaN
         Score   NaN  NaN    67  NaN
Pratibha Grade    A1  NaN   NaN  NaN
         Score    99  NaN   NaN  NaN
Ritika   Grade   NaN  NaN   NaN   A2
         Score   NaN  NaN   NaN   87
Saumya   Grade   NaN   A2   NaN  NaN
         Score   NaN   88   NaN  NaN
```

This is unstacking

```
>>> pv.stack(0).stack()
Name                 Subject
Aryan     Grade      Maths          B
          Score      Maths         67
Pratibha  Grade      CS            A1
          Score      CS            99
Ritika    Grade      Phy           A2
          Score      Phy           87
Saumya    Grade      Chem          A2
          Score      Chem          88
dtype: object
```

Unstacking is just like stack the only difference is that there is an argument '0' is passed in stack( ) method.

Pivoting using pivot_table( ) method. . . .

- This is the generalization of pivot( ) method.
- When you have duplicate values for only one index or duplicate values for one column then pivot_table( ) method is used.
- A pivot table contains counts, sums and table data related functions.
- pivot_table( ) method creates a DataFrame, kind of Excel Sheet.
- This method is used to convert row into column and vice-versa.
- It allows grouping of any data field.
- Its syntax is →

```
pandas.pivot_table (DataFrame, values=None, index=None,
                    columns=None, aggfunc='mean',
                        fill_value=None, margins=False, dropna=True,
                        margins_name='All')
```

- All the arguments are not necessary in .pivot_table() method, because there are some default values for some arguments.

Pivoting using pivot_table( ) method. . .

pandas.pivot_table (DataFrame, values=None, index=None, columns=None, aggfunc='mean', fill_value=None, margins=False, dropna=True, margins_name='All')

- All the arguments are not necessary in .pivot_table() method, because there are some default values for some arguments.
- In its syntax -
  - **DataFrame** → is a pandas DataFrame.
  - **values**→this is optional and also a column to be aggregated.
  - **index**→this is column, grouper, array or list name.
  - **columns**→ this is a column, grouper, array or list.
  - **aggfunc** → is an aggregation function.
  - **fill_value**→ we can set default values using this, if the values are not given.
  - **margins**→ this is a boolean whose default is false. If we make it true then the sum of row and column in resulting dataframe.
  - **dropna**→if this is true then it drops row having missing data
  - **margins_name='All'** →if margins is true then it keeps the name of the rows and column of total.

Pivoting using pivot_table( ) method. . .

We create a pivot table considering the following data.→

```
import pandas as pd
ClassXII={'Name':['Pratibha','Ritika','Saumya','Aryan',\
                  'Pratibha','Ritika','Saumya','Aryan',\
                  'Pratibha','Ritika','Saumya','Aryan',\
                  'Pratibha','Ritika','Saumya','Aryan'],\
          'Test':['Semester1','Semester1','Semester1','Semester1',\
                  'Semester1','Semester1','Semester1','Semester1',\
                  'Semester2','Semester2','Semester2','Semester2',\
                  'Semester2','Semester2','Semester2','Semester2'],\
          'Subject':['CS','CS','CS','CS','IP','IP','IP','IP',\
                     'CS','CS','CS','CS','IP','IP','IP','IP'],\
          'Marks':[99,87,88,67,98,86,88,68,97,85,89,62,94,84,81,69]}
df=pd.DataFrame(ClassXII,columns=['Name','Test','Subject','Marks'])
```

```
>>> df
        Name       Test Subject  Marks
0   Pratibha  Semester1      CS     99
1     Ritika  Semester1      CS     87
2     Saumya  Semester1      CS     88
3      Aryan  Semester1      CS     67
4   Pratibha  Semester1      IP     98
5     Ritika  Semester1      IP     86
6     Saumya  Semester1      IP     88
7      Aryan  Semester1      IP     68
8   Pratibha  Semester2      CS     97
9     Ritika  Semester2      CS     85
10    Saumya  Semester2      CS     89
11     Aryan  Semester2      CS     62
12  Pratibha  Semester2      IP     94
13    Ritika  Semester2      IP     84
14    Saumya  Semester2      IP     81
15     Aryan  Semester2      IP     69
```

WE can take this data from CSV File.

```
>>> pv=df.pivot_table(index='Name',values='Marks',aggfunc='mean')
```

or

```
>>> pv=df.pivot_table(index='Name',aggfunc='mean')
>>> pv
           Marks
Name
Aryan       66.5
Pratibha    97.0
Ritika      85.5
Saumya      86.5
```

Pivoting using pivot_table( ) method. . .

pivot table can be created by the following method also.→

```
>>> pd.pivot_table(df,index='Name',aggfunc='mean')
         Marks
Name
Aryan     66.5
Pratibha  97.0
Ritika    85.5
Saumya    86.5
```

```
>>> pv=df.pivot_table(index='Name',columns='Subject',aggfunc='mean')
>>> pv
         Marks
Subject     CS    IP
Name
Aryan     64.5  68.5
Pratibha  98.0  96.0
Ritika    86.0  85.0
Saumya    88.5  84.5
```

Pay attention on the values of aggfunc

```
>>> pv=df.pivot_table(index='Name',columns='Subject',aggfunc='count')
>>> pv
         Marks     Test
Subject     CS IP   CS IP
Name
Aryan        2  2    2  2
Pratibha     2  2    2  2
Ritika       2  2    2  2
Saumya       2  2    2  2
```

Pivoting using pivot_table( ) method. . .

An excercise for you →

An Emp table contains the following data:

Empno	Name	Department	Salary	Commission	Job
100	Sunita Sharma	RESEARCH	45600	5600.0	CLERK
101	Ashok Singhal	SALES	43900	3900.0	SALESMAN
102	Sumit Avasti	SALES	27000	7000.0	SALESMAN
103	Jyoti Lamba	RESEARCH	45900	4900.0	MANAGER
104	Martin S.	SALES	32500	3500.0	SALESMAN
105	Binod Goel	SALES	45200	4200.0	MANAGER
106	Chetan Gupta	ACCOUNTS	36800	6800.0	MANAGER
107	Sudhir Rawat	RESEARCH	37000	7000.0	ANALYST
108	Kavita Sharma	ACCOUNTS	42900	4900.0	CLERK
109	Tushar Tiwari	SALES	49500	4500.0	MANAGER
110	Anand Rathi	OPERATIONS	41600	8200.0	SR. MANAGER
111	Sumit Vats	RESEARCH	47800	NaN	SR. MANAGER
112	Manoj Kaushik	OPERATIONS	43600	NaN	CLERK

(a) Using above table create a DataFrame called dfE.
(*b*) Display the department wise total salary.
(*c*) Display the department wise average salary.
(*d*) Display the department wise total and average salary.
(*e*) Display the department wise maximum and minimum salary.
(*f*) Display the department and job wise maximum salary.

Pivoting using pivot_table( ) method. . .

Solution→

First you will create a dataframe of table using pandas.

After that you have to apply the following functions.-

(*b*) `pd.pivot_table(dfE, index='Department', values='Salary', aggfunc='sum')`

(*c*) `pd.pivot_table(dfE, index='Department', values='Salary')`

Or

`pd.pivot_table(dfE, index='Department', values='Salary', aggfunc='mean')`

(*d*) `pd.pivot_table(dfE, index='Department', values='Salary', aggfunc=['sum', 'mean'])`

(*e*) `pd.pivot_table(dfE, index='Department', values='Salary', aggfunc=['max', 'min'])`

(*f*) `pd.pivot_table(dfE, index=['Department', 'Job'], values='Salary', aggfunc='max')`

Sorting of DataFrames

- Data of DataFrame can be sort according to values of row and column.
- By default sorting is done on row labels in ascending order.
- Pandas DataFrames has two useful sort functions →
  - *sort_values( ):* it sorts the data of given column to the function in ascending or descending order.
  - *sort_index( ):* this function sorts rows (axis=0) or columns (axis=1).
- Its syntax is as follows→
- *DataFrame.sort_values(by = None, axis=0, ascending = True, inplace = False)*
- *DataFrame.sort_index(by = None, axis=0, ascending = True, inplace = False)*
- Here –
  - **by:** column to be sorted.
  - **axis:** here passing 0 means sorting will be done row wise and 1 means column wise
  - **ascending:** by default ascending is true
  - **inplace:** default is false if you don't want a new dataframe then set it true.

DataFrames Sorting...

```
>>> df
      Name Subject  Marks Grade
0  Pratibha      CS     99    A+
1    Ritika      IP     87     A
2    Saumya     PHY     88    B+
3     Aryan    CHEM     67     B
```

by default sorting is in ascending order.

```
>>> dfn=df.sort_values('Name')
```

or

```
>>> dfn=df.sort_values(by='Name')
>>> dfn
      Name Subject  Marks Grade
3     Aryan    CHEM     67     B
0  Pratibha      CS     99    A+
1    Ritika      IP     87     A
2    Saumya     PHY     88    B+
```

To sort in desceiding order the example is as under.

```
>>> dfn=df.sort_values(by='Name',ascending=False)
>>> dfn
      Name Subject  Marks Grade
2    Saumya     PHY     88    B+
1    Ritika      IP     87     A
0  Pratibha      CS     99    A+
3     Aryan    CHEM     67     B
```

Value of Ascending parameter is false

```
>>> dfn=df.sort_values(['Name','Marks'],ascending=True)
>>> dfn
      Name Subject  Marks Grade
3     Aryan    CHEM     67     B
0  Pratibha      CS     99    A+
1    Ritika      IP     87     A
2    Saumya     PHY     88    B+
```

If we give two columns like this then sorting on multiple columns will be done.

Sort by index

```
>>> dfn=df.sort_index()
>>> dfn
      Name Subject  Marks Grade
0  Pratibha      CS     99    A+
1    Ritika      IP     87     A
2    Saumya     PHY     88    B+
3     Aryan    CHEM     67     B
```

Sorting in ascending order

```
>>> dfn=df.sort_index(ascending=False)
>>> dfn
      Name Subject  Marks Grade
3     Aryan    CHEM     67     B
2    Saumya     PHY     88    B+
1    Ritika      IP     87     A
0  Pratibha      CS     99    A+
```

Sorting in descending order

Points to remember:

1. pivot( ) method creates a new table whose row and column are unique.
2. pivot( ) method is used to pivot without aggregation.
3. *stacking* means moving innermost column index to innermost row index.

- Please follow our blog and subscribe youtube channel to get all the chapter and lectures.

www.pythontrends.wordpress.com