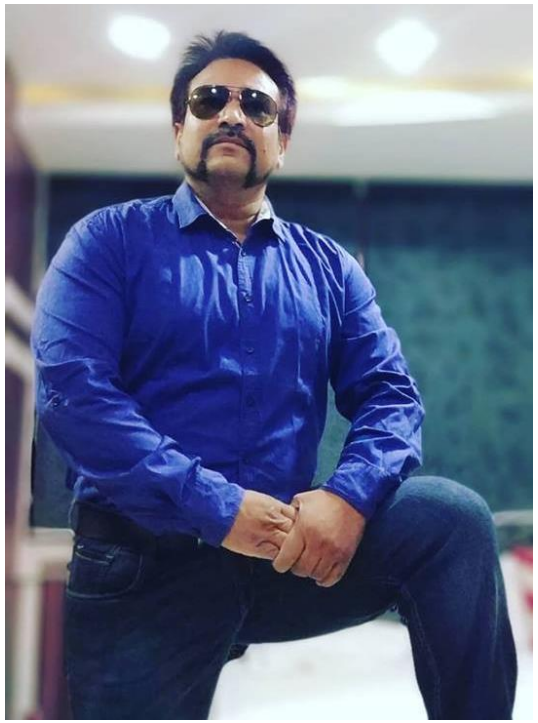
Advanced Operations on DataFrames

सीबीएसई पाठ्यक्रम पर आधारित

इन्फार्मेटिक्स प्रैक्टिसेज कक्षा -12

अध्याय -2



द्वारा:
संजीव भदौरिया
स्नातकोत्तर शिक्षक (संगणक विज्ञान)
के० वि० बाराबंकी (लखनऊ संभाग)

Pivoting DataFrame

- ***Data analysis*** के लिए Pandas एक प्रचलित library है |
- किसी भी data analyst के लिए प्रमुख कार्यों में से एक डेटा टेबल को ***pivot*** करना है। अर्थात table data को एक धुरी प्रदान करना जिसके आधार पर database काम करे |
- Pandas का प्रयोग करके MS-Excel के प्रकार के pivot tables बनाये जा सकते हैं |
- ये बड़े data को तुरंत ***summarize*** करके meaningful reports तैयार करने में आपके समय की बहुत बचत करते हैं |
- Pivot table हमें एक बड़े, विस्तृत डेटा सेट से महत्वपूर्ण record निकालने की अनुमति देती है।
- Pivot tables स्वतः sort, count, total इत्यादि कर लेती हैं |
- एक सामान्य बात कहें तो pivot करने का अर्थ है किसी index या column से unique value को प्रयोग करना और DataFrame बनाना |
- Pandas के द्वारा pivot table बनाने के लिए हम ***pivot( )*** या ***pivot_table()*** method का प्रयोग करते हैं |

pivot( ) method का प्रयोग करके Pivoting करना

- pivot() method, column values के आधार पर data को reshape करके नया DataFrame बनाता है |
- यह method 3 arguments लेता है - ***index***, ***columns*** और ***values*** | इनमे से 2 को लेना अनिवार्य है |
- इन arguments की value के रूप में आपको original table के column नाम देने होते हैं |
- तब pivot ( ) एक नयी table बनाता है जिनके row और column के index वही होते हैं जिनको आपने argument के रूप में pass किया है |
- नयी table की cell values उसी column से आयेंगी जिसको आपने parameter के रूप में दिया था | इसका syntax निम्न है –

```
pandas.pivot(index, columns, values)
```

- जहाँ index के द्वारा नए DataFrame का index बनता है जो table से लिया गया column name है |
- जहाँ columns के द्वारा नए DataFrame के columns बनते है जो table से लिए गए column name है |
- जहाँ values के द्वारा नए DataFrame की columns बनती है जो table से लिए गए column name की values हैं |

pivot( ) method का प्रयोग करके Pivoting करना

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

- उदाहरण→ DataFrame बनाना

```
>>> 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'])
```

- pivot table बनाना →

```
>>> 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
```

हम इस pivot table में देख सकते हैं कि एक नयी table बनी है और Score column की values अलग अलग column में आयीं हैं | जबकि इसकी Name और Subject column, original table से match हो रही हैं| जहाँ values match नहीं हो रही हैं उस जगह NaN (None) स्वतः ही चला गया है |

pivot( ) method का प्रयोग .fillna( ) के साथ

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

- उदाहरण→ DataFrame बनाना

```
>>> 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'])
```

- pivot table .fillna( ) के साथ बनाना बनाना →

```
>>> 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
```

हम इस pivot table में देख सकते हैं कि एक नयी table बनी है और Score column की values अलग अलग column में आयीं हैं | जबकि इसकी Name और Subject column, original table से match हो रही हैं| जहाँ values match नहीं हो रही हैं उस जगह NaN (None) भी नहीं आया बस रिक्त स्थान आगया है |

Multiple columns के द्वारा pivoting करना

इसमें बस values parameter को हटा देते हैं |

syntax → pandas.pivot(index, columns)

- उदाहरण→ DataFrame बनाना

```
>>> 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'])
```

- pivot table .fillna( ) के साथ बनाना बनाना →

```
>>> 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
```

Multiple columns के द्वारा pivoting करना. . .

- पिछले उदहारण में हमने देखा कि कई index बन गए Subjects के और उनकी values भी एक बार Score के लिए और एक बार Grade के लिए प्रत्येक नाम के लिए दिखाई गयी है |
- इन्हें हम फ़िल्टर भी कर सकते हैं →

```
>>> 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

- हमेशा ध्यान रखिये यदि index और columns के multiple values के साथ combination होंगे तो value error आयेगी |

```
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
```

stack( ) और unstack( ) methods का प्रयोग

- stack( ) और unstack( ) methods दोनों DataFrame का layout पलट देते हैं अर्थात columns के सारे levels को row में और row के सारे levels को column में पलट देते हैं | DataFrame की ***stacking*** का मतलब है innermost column index को innermost row index की ओर ले जाना | और इसके उलटे क्रम को ***unstacking*** कहते हैं |

```
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
```

Stack Method का प्रयोग करने पर subjects जो horizontal में थे वो सभी vertical आगये | और यहाँ यह column breakdown में last level लेता है और इसे last row breakdown में बदल देता है

Stack Method का प्रयोग

```
>>> 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
```

stack( ) और unstack( ) methods का प्रयोग

```
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
```

इस तरह से भी stack का प्रयोग किया जा सकता है जिसके stacking के बाद पुनः stacking की गयी है तो इसमें यह बचे हुए column level को भी move कर देता है |

stack( ) और unstack( ) methods का प्रयोग

```
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
```

यह unstacking है

```
>>> 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 भी stack के जैसे ही होती है बस उसमे एक argument '0' pass कर दिया जाता है |

pivot_table( ) method का प्रयोग करके Pivoting करना

- यह pivot( ) method का generalization है |
- जब आपके पास एक ही index या column के लिए duplicate values हों तो pivot_table( ) method का प्रयोग किया जाता है |
- एक pivot table में counts, sums तथा table data से सम्बंधित अन्य functions भी उपलब्ध होते हैं |
- pivot_table( ) method एक प्रकार से excel sheet जैसी ही DataFrame बनाती है |
- यह भी row को column में और column को row में बदलने के काम आता है |
- यह किसी भी data field की grouping को allow करता है |
- इसका syntax है →

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

- .pivot_table() method में सारे arguments ज़रूरी नहीं हैं क्योंकि इसके अन्दर कुछ default values होती हैं |

pivot_table( ) method का प्रयोग करके Pivoting करना…

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

- .pivot_table() method में सारे arguments ज़रूरी नहीं हैं क्योंकि इसके अन्दर कुछ default values होती हैं |
- इसके syntax में -
  - **DataFrame** → एक pandas DataFrame है |
  - **values**→यह optional है और aggregate किये जाने वाला column है |
  - **index**→column, grouper, array, या list का नाम है |
  - **columns**→यह column, grouper, array, या list है |
  - **aggfunc** → aggregation function है |
  - **fill_value**→इसके द्वारा हम default values सेट कर सकते हैं यदि values न दी गयीं हो |
  - **margins**→ यह एक boolean होता है जिसका default false होता है, यदि हम इसे true कर देते हैं तो resulting dataframe में row और column का sum भी जुड़ जाता है |
  - **dropna**→यदि यह true है तो यह missing data वाली row को drop कर देता है |
  - **margins_name='All'** →जब margins true हो तब total वाले row और column के नाम रखता है |

pivot_table( ) method का प्रयोग करके Pivoting करना ...

हम निम्न data को consider करके pivot table बनाते हैं →

```
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
```

हम यह data CSV फाइल से भी le सकते हैं

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

या

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

pivot_table( ) method का प्रयोग करके Pivoting करना …

pivot table को हम निम्न तरीके से भी बना सकते हैं →

```
>>> 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
```

यहाँ aggfunc के values पर ध्यान दीजिये |

```
>>> 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
```

pivot_table( ) method का प्रयोग करके Pivoting करना …

आपके लिए एक excercise →

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.

pivot_table( ) method का प्रयोग करके Pivoting करना …

Solution→

सबसे पहले आप दी गयी table का DataFrame बनायेंगे pandas का प्रयोग करके |

उसके बाद आपको निम्न function अप्लाई करने हैं -

(*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')`

DataFrames की Sorting

- DataFrame के data को भी row और column के values के आधार पर sort किया जा सकता है |
- By default sorting, row labels पर होती है वो भी बढ़ते हुए क्रम में |
- Pandas DataFrames के पास दो उपयोगी sort functions होते हैं →
  - sort_values( ): यह function दिए गए column के data को ascending या descending आर्डर में sort करता है |
  - sort_index( ): यह function rows (axis=0) या columns (axis=1) को sort करता है
- इनका syntax निम्नवत है →
- *DataFrame.sort_values(by = None, axis=0, ascending = True, inplace = False)*
- *DataFrame.sort_index(by = None, axis=0, ascending = True, inplace = False)*
- यहाँ –
  - **by:** sort किया जाने वाला column
  - **axis:** यहाँ 0 pass करने का मतलब सोर्टिंग row wise और 1 का मतलब column wise
  - **ascending:** default में ascending true रहता है
  - **inplace:** default false होती है यदि आप नया dataframe नहीं चाहते हैं तो true pass करना होगा |

DataFrames की 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
```

यहाँ by default ascending आर्डर में sort हुआ है |

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

या

```
>>> 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+
```

Descending आर्डर में sort करने के लिए निम्न उदाहरण है |

```
>>> 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
```

Ascending parameter की वैल्यू False pass कर दी है |

```
>>> 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+
```

यहाँ यदि हम दो column इस प्रकार से देदेते हैं तो multiple columns पर sorting apply हो जाती है |

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
```

यहाँ यह ascending आर्डर में sorting है |

```
>>> 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+
```

यहाँ यह descending आर्डर में sorting है |

याद रखने योग्य बातें:

1. pivot( ) method एक नयी table बनाता है जिनके row और column unique होते हैं |
2. pivot( ) method का उपयोग aggregation के बिना pivot के लिए किया जाता है।
3. *stacking* का मतलब है innermost column index को innermost row index की ओर ले जाना |

- कृपया हमारे ब्लॉग को फॉलो करिए और youtube channel को subscribe करिए | ताकि आपको और सारे chapters मिल सकें |

www.pythontrends.wordpress.com