Learning_Pandas_Part_5_Reshaping

June 20, 2021

0.0.1 Prepared by Abhishek Kumar

0.0.2 https://www.linkedin.com/in/abhishekkumar-0311/

[1]:
```python
# To get multiple outputs in the same cell

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

[2]:
```python
# Import the required libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

1 Wide to Long DataFrame

One record to many records based on a ID column

`py 1. df.melt(id_vars=[ ], value_vars=[ ], var_name=[ ], value_name=[ ]) 2. pd.wide_to_long(df, i=[ ], j=[ ], stubnames=[ ], sep="_")  # stubnames provides the flexibility to add the multiple sets of series of variables` apply reset_index() to flatten out the indices and make the it more usable.

1.1 df.melt()

[3]:
```python
df = pd.DataFrame({'id': [1,2],
                   'name': ['a','b'],
                   'prem1' : [100,280],
                   'prem2' : [200,180],
                   'prem3' : [300,80],})
df
```

[3]:
```
   id name  prem1  prem2  prem3
0   1    a    100    200    300
1   2    b    280    180     80
```

[4]:
```
df_melted = df.melt(id_vars=['id','name']).sort_values('id')
df_melted
```

[4]:
```
   id name variable  value
0   1    a    prem1    100
2   1    a    prem2    200
4   1    a    prem3    300
1   2    b    prem1    280
3   2    b    prem2    180
5   2    b    prem3     80
```

[5]:
```
df2 = pd.DataFrame({'id': [1,2],
                   'name': ['a','b'],
                   'prem1' : [100,280],
                   'prem2' : [np.NaN,180],
                   'prem3' : [300,np.NaN],})
df2
```

[5]:
```
   id name  prem1  prem2  prem3
0   1    a    100    NaN  300.0
1   2    b    280  180.0    NaN
```

[6]:
```
df2_melted = df2.melt(id_vars=['id','name'], var_name = 'month', value_name =␣
 ↪'premiums').sort_values('id')
df2_melted
```

[6]:
```
   id name  month  premiums
0   1    a  prem1     100.0
2   1    a  prem2       NaN
4   1    a  prem3     300.0
1   2    b  prem1     280.0
3   2    b  prem2     180.0
5   2    b  prem3       NaN
```

[7]:
```
# df2_melted = df2_melted.loc[]
```

[8]:
```
df3 = df2.copy()

df3_melted = df3.melt(id_vars=['id'], value_vars=['prem1','prem2','prem3'],␣
 ↪var_name = 'month', value_name = 'premiums').sort_values('id')
df3_melted
```

[8]:
```
   id  month  premiums
0   1  prem1     100.0
2   1  prem2       NaN
4   1  prem3     300.0
1   2  prem1     280.0
```

```
3   2  prem2     180.0
5   2  prem3       NaN
```

1.1.1 Example 2

[9]:
```
# Setup : DataFrame creation

salary = [['1','Abhishek Kumar','AIML', 'Machine Learning Engineer','M', 'Y',␣
↪'04051990', 1121000],
          ['2','Arjun Kumar','DM', 'Tech Lead','M', 'Y', '09031992', 109000],
          ['3','Vivek Raj','DM', 'Devops Engineer','M', 'N', np.NaN , 827000],
          ['4','Mika Singh','DM', 'Data Analyst','F', 'Y', '15101991',  np.NaN],
          ['5','Anusha Yenduri','AIML', 'Data Scientist','F', 'Y', '01011989', ␣
↪921000],
          ['6','Ritesh Srivastava','AIML', 'Data Engineer','M', 'Y', np.NaN,␣
↪785000]]

columns_name=['Emp_Id','Emp_Name','Department','Role','Gender', 'WFH Status',␣
↪'DOB', 'Salary']

emp_df = pd.DataFrame(salary,columns=columns_name)
emp_df
```

[9]:
```
  Emp_Id           Emp_Name Department                       Role Gender  \
0      1     Abhishek Kumar       AIML  Machine Learning Engineer      M
1      2        Arjun Kumar         DM                  Tech Lead      M
2      3          Vivek Raj         DM            Devops Engineer      M
3      4         Mika Singh         DM               Data Analyst      F
4      5     Anusha Yenduri       AIML             Data Scientist      F
5      6  Ritesh Srivastava       AIML              Data Engineer      M

  WFH Status       DOB     Salary
0          Y  04051990  1121000.0
1          Y  09031992   109000.0
2          N       NaN   827000.0
3          Y  15101991        NaN
4          Y  01011989   921000.0
5          Y       NaN   785000.0
```

[10]:
```
# Sample data set-up

emp_df_1 = emp_df.copy()

emp_df_1['Holi_Bonus'] = emp_df_1['Salary']*0.05
emp_df_1['Diwali_Bonus'] = emp_df_1['Salary']*0.075
emp_df_1['Yearly_Bonus'] = emp_df_1['Salary']*0.10
```

```
emp_df_1
```

[10]:
```
   Emp_Id            Emp_Name Department                      Role Gender  \
0       1      Abhishek Kumar       AIML  Machine Learning Engineer      M
1       2         Arjun Kumar         DM                  Tech Lead      M
2       3           Vivek Raj         DM           Devops Engineer      M
3       4          Mika Singh         DM              Data Analyst      F
4       5      Anusha Yenduri       AIML            Data Scientist      F
5       6   Ritesh Srivastava       AIML             Data Engineer      M

  WFH Status        DOB     Salary  Holi_Bonus  Diwali_Bonus  Yearly_Bonus
0          Y  04051990  1121000.0     56050.0       84075.0      112100.0
1          Y  09031992   109000.0      5450.0        8175.0       10900.0
2          N        NaN   827000.0     41350.0       62025.0       82700.0
3          Y  15101991        NaN         NaN           NaN           NaN
4          Y  01011989   921000.0     46050.0       69075.0       92100.0
5          Y        NaN   785000.0     39250.0       58875.0       78500.0
```

[11]:
```
emp_df_1_long = emp_df_1.melt(id_vars = ['Emp_Id','Emp_Name'] ,
                              value_vars = [␣
 ↪'Holi_Bonus','Diwali_Bonus','Yearly_Bonus' ],
                              var_name = 'Event',
                              value_name = 'Bonus' )
emp_df_1_long
```

[11]:
```
    Emp_Id           Emp_Name         Event     Bonus
0        1     Abhishek Kumar    Holi_Bonus   56050.0
1        2        Arjun Kumar    Holi_Bonus    5450.0
2        3          Vivek Raj    Holi_Bonus   41350.0
3        4         Mika Singh    Holi_Bonus       NaN
4        5     Anusha Yenduri    Holi_Bonus   46050.0
5        6  Ritesh Srivastava    Holi_Bonus   39250.0
6        1     Abhishek Kumar  Diwali_Bonus   84075.0
7        2        Arjun Kumar  Diwali_Bonus    8175.0
8        3          Vivek Raj  Diwali_Bonus   62025.0
9        4         Mika Singh  Diwali_Bonus       NaN
10       5     Anusha Yenduri  Diwali_Bonus   69075.0
11       6  Ritesh Srivastava  Diwali_Bonus   58875.0
12       1     Abhishek Kumar  Yearly_Bonus  112100.0
13       2        Arjun Kumar  Yearly_Bonus   10900.0
14       3          Vivek Raj  Yearly_Bonus   82700.0
15       4         Mika Singh  Yearly_Bonus       NaN
16       5     Anusha Yenduri  Yearly_Bonus   92100.0
17       6  Ritesh Srivastava  Yearly_Bonus   78500.0
```

1.2 pd.wide_to_long()

[12]:
```
df4 = pd.DataFrame({'id': [1,2],
                    'name': ['a','b'],
                    'prem1' : [100,280],
                    'prem2' : [np.NaN,180],
                    'prem3' : [300,np.NaN],
                    'disc1' : [20,40],
                    'disc2' : [np.NaN,30],
                    'disc3' : [50,np.NaN],})
df4
```

[12]:
```
   id name  prem1  prem2  prem3  disc1  disc2  disc3
0   1    a    100    NaN  300.0     20    NaN   50.0
1   2    b    280  180.0    NaN     40   30.0    NaN
```

[13]:
```
# melt is not working as expected.
# There are 2 sets of sequential columns and both the sets are transposed to␣
 ↪the same column
# NOT Working as EXPECTED

# df4_melted = df4.melt(id_vars=['id','name'],␣
 ↪value_vars=['prem1','prem2','prem3','disc1','disc2','disc3'], var_name =␣
 ↪'month', value_name = 'values').sort_values('id').reset_index(drop='index')
# df4_melted
```

Another way to transform is to use the wide_to_long() panel data convenience function. It is less flexible than melt(), but more user-friendly.

[14]:
```
df4_melted1 = pd.wide_to_long(df4, i=['id','name'], j='month',␣
 ↪stubnames=['prem','disc'])
df4_melted1
```

[14]:
```
                prem  disc
id name month
1  a    1      100.0  20.0
        2        NaN   NaN
        3      300.0  50.0
2  b    1      280.0  40.0
        2      180.0  30.0
        3        NaN   NaN
```

[15]:
```
df4_melted1.reset_index(inplace=True)
df4_melted1
```

[15]:
```
   id name  month   prem  disc
0   1    a      1  100.0  20.0
```

```
1   1     a       2    NaN   NaN
2   1     a       3  300.0  50.0
3   2     b       1  280.0  40.0
4   2     b       2  180.0  30.0
5   2     b       3    NaN   NaN
```

[16]:
```
# Trying to see the usage of suffix= parameter. Not completed yet.
# df4_melted2 = pd.wide_to_long(df4, i=['id','name'], j='month',␣
 ↪stubnames=['prem','disc'])#, suffix='1')
# df4_melted2
```

1.3 df.stack()

[17]:
```
df5 = pd.DataFrame({'id': [1,2],
                   'name': ['a','b'],
                   'prem1' : [100,280],
                   'prem2' : [np.NaN,180],
                   'prem3' : [300,np.NaN]})
df5
```

[17]:
```
   id name  prem1  prem2  prem3
0   1    a    100    NaN  300.0
1   2    b    280  180.0    NaN
```

[18]:
```
df5.set_index(['id','name']).stack().reset_index()
```

[18]:
```
   id name level_2      0
0   1    a   prem1  100.0
1   1    a   prem3  300.0
2   2    b   prem1  280.0
3   2    b   prem2  180.0
```

```
> 1. Important thing to note - there is single series of variable (perm1 - perm3), which is tra
> 2. The index is set before the process of stacking.
> 3. If there is multile sets of series of variables, then this would not work as expected.
> 4. By default, dropna = True, and hence it drops the NaN values
```

[19]:
```
df5.set_index(['id','name']).stack(dropna=False).reset_index()
```

[19]:
```
   id name level_2      0
0   1    a   prem1  100.0
1   1    a   prem2    NaN
2   1    a   prem3  300.0
3   2    b   prem1  280.0
4   2    b   prem2  180.0
5   2    b   prem3    NaN
```

```
[20]: df6 = pd.DataFrame({'id': [1,2],
                         'name': ['a','b'],
                         'prem1' : [100,280],
                         'prem2' : [np.NaN,180],
                         'prem3' : [300,np.NaN],
                         'disc1' : [20,40],
                         'disc2' : [np.NaN,30],
                         'disc3' : [50,np.NaN]})
      df6
```

```
[20]:    id name  prem1  prem2  prem3  disc1  disc2  disc3
      0   1    a    100    NaN  300.0     20    NaN   50.0
      1   2    b    280  180.0    NaN     40   30.0    NaN
```

```
[21]: df6_stacked = df6.set_index(['id','name']).stack().reset_index()
      df6_stacked
```

```
[21]:    id name level_2      0
      0   1    a   prem1  100.0
      1   1    a   prem3  300.0
      2   1    a   disc1   20.0
      3   1    a   disc3   50.0
      4   2    b   prem1  280.0
      5   2    b   prem2  180.0
      6   2    b   disc1   40.0
      7   2    b   disc2   30.0
```

```
[22]: # stack is not working as expected.
      # There are 2 sets of sequential columns and both the sets are transposed to␣
       ↪the same column
      # NOT Working as EXPECTED
```

```
[ ]:
```

2 Long to Wide DataFrame

Multiple records per ID to a single(one) record of each ID.

`python 1. pd.pivot() 2. pd.pivot_table() 3. Use df.set_index([id_vars columns and var_name columns]) and chain it with .unstack(level=2 (here))`

2.0.1 pd.pivot() - Does not work for multiple indexes, So in this case, does not work

2.0.2 pd.pivot_table() - Although it is for aggregation, it worked to change LONG to WIDE Data

```
[23]: df4_melted1
```

```
[23]:    id name  month   prem  disc
      0   1    a      1  100.0  20.0
      1   1    a      2    NaN   NaN
      2   1    a      3  300.0  50.0
      3   2    b      1  280.0  40.0
      4   2    b      2  180.0  30.0
      5   2    b      3    NaN   NaN
```

```
[24]: df_wide = pd.pivot_table(df4_melted1, index=['id','name'], columns='month',␣
       ↪values=['prem','disc'])
      df_wide
```

```
[24]:         disc                prem
      month      1     2     3     1      2      3
      id name
      1  a    20.0   NaN  50.0  100.0    NaN  300.0
      2  b    40.0  30.0   NaN  280.0  180.0    NaN
```

```
[25]: df_wide.columns
```

```
[25]: MultiIndex([('disc', 1),
                  ('disc', 2),
                  ('disc', 3),
                  ('prem', 1),
                  ('prem', 2),
                  ('prem', 3)],
                 names=[None, 'month'])
```

```
[26]: # df_wide = df4_melted1.pivot(index=['id','name'], columns='month',␣
       ↪values=['prem'])
      # df_wide
```

```
[27]: df_wide.columns = ['_'.join(map(str, tup)) for tup in df_wide.columns]
      df_wide.reset_index()
```

```
[27]:    id name  disc_1  disc_2  disc_3  prem_1  prem_2  prem_3
      0   1    a    20.0     NaN    50.0   100.0     NaN   300.0
      1   2    b    40.0    30.0     NaN   280.0   180.0     NaN
```

2.0.3 df.unstack() -

Use df.set__index([id__vars columns and var__name columns]) and chain it with .unstack(level=2 (here))

```
[28]: wide_df = df4_melted1.set_index(['id','name','month']).unstack(level=2)
      wide_df
```

```
[28]:           prem                    disc
      month        1       2       3     1     2     3
      id name
      1  a      100.0     NaN  300.0  20.0   NaN  50.0
      2  b      280.0   180.0    NaN  40.0  30.0   NaN
```

ID: level = 0; RegionVariable: level = 1; 'EXP': level = 2; 'ModelID': level = 3;

```
[29]: wide_df.columns
```

```
[29]: MultiIndex([('prem', 1),
                  ('prem', 2),
                  ('prem', 3),
                  ('disc', 1),
                  ('disc', 2),
                  ('disc', 3)],
                 names=[None, 'month'])
```

```
[30]: # Code to flatten the list and at the same time concatenating it.

      wide_df.columns = ['_'.join(map(str, tup)) for tup in wide_df.columns] #␣
       ↪Everything is back to the first dataframe
```

```
[31]: wide_df.columns
```

```
[31]: Index(['prem_1', 'prem_2', 'prem_3', 'disc_1', 'disc_2', 'disc_3'],
      dtype='object')
```

```
[32]: wide_df
```

```
[32]:          prem_1  prem_2  prem_3  disc_1  disc_2  disc_3
      id name
      1  a      100.0     NaN   300.0    20.0     NaN    50.0
      2  b      280.0   180.0     NaN    40.0    30.0     NaN
```

```
[33]: wide_df.reset_index()
```

```
[33]:    id name  prem_1  prem_2  prem_3  disc_1  disc_2  disc_3
      0   1    a   100.0     NaN   300.0    20.0     NaN    50.0
      1   2    b   280.0   180.0     NaN    40.0    30.0     NaN
```

```
[ ]:
```

2.0.4 Example 2

[34]:
```
emp_df_1_long
```

[34]:
```
   Emp_Id            Emp_Name          Event     Bonus
0       1      Abhishek Kumar     Holi_Bonus   56050.0
1       2         Arjun Kumar     Holi_Bonus    5450.0
2       3           Vivek Raj     Holi_Bonus   41350.0
3       4          Mika Singh     Holi_Bonus       NaN
4       5      Anusha Yenduri     Holi_Bonus   46050.0
5       6   Ritesh Srivastava     Holi_Bonus   39250.0
6       1      Abhishek Kumar   Diwali_Bonus   84075.0
7       2         Arjun Kumar   Diwali_Bonus    8175.0
8       3           Vivek Raj   Diwali_Bonus   62025.0
9       4          Mika Singh   Diwali_Bonus       NaN
10      5      Anusha Yenduri   Diwali_Bonus   69075.0
11      6   Ritesh Srivastava   Diwali_Bonus   58875.0
12      1      Abhishek Kumar   Yearly_Bonus  112100.0
13      2         Arjun Kumar   Yearly_Bonus   10900.0
14      3           Vivek Raj   Yearly_Bonus   82700.0
15      4          Mika Singh   Yearly_Bonus       NaN
16      5      Anusha Yenduri   Yearly_Bonus   92100.0
17      6   Ritesh Srivastava   Yearly_Bonus   78500.0
```

[35]:
```
emp_df_1_wide_1 = emp_df_1_long.pivot_table(index =  ['Emp_Id','Emp_Name'] ,
                                           columns = 'Event',
                                           values = 'Bonus' ).reset_index()
emp_df_1_wide_1
```

[35]:
```
Event Emp_Id           Emp_Name  Diwali_Bonus  Holi_Bonus  Yearly_Bonus
0          1     Abhishek Kumar       84075.0     56050.0      112100.0
1          2        Arjun Kumar        8175.0      5450.0       10900.0
2          3          Vivek Raj       62025.0     41350.0       82700.0
3          5     Anusha Yenduri       69075.0     46050.0       92100.0
4          6  Ritesh Srivastava       58875.0     39250.0       78500.0
```

[36]:
```
emp_df_1_wide_2 = emp_df_1_long.pivot_table(index =  ['Emp_Id','Emp_Name'] ,
                                            columns = 'Event',
                                            values = 'Bonus',
                                            margins = True ).reset_index()  #␣
 ↪default aggfunc = 'mean'
emp_df_1_wide_2
```

[36]:
```
Event Emp_Id           Emp_Name  Diwali_Bonus  Holi_Bonus  Yearly_Bonus  \
0          1     Abhishek Kumar       84075.0     56050.0      112100.0
1          2        Arjun Kumar        8175.0      5450.0       10900.0
2          3          Vivek Raj       62025.0     41350.0       82700.0
```

```
3           5     Anusha Yenduri       69075.0     46050.0       92100.0
4           6  Ritesh Srivastava       58875.0     39250.0       78500.0
5         All                          56445.0     37630.0       75260.0

Event       All
0       84075.0
1        8175.0
2       62025.0
3       69075.0
4       58875.0
5       56445.0
```

[37]:
```
emp_df_1_wide_3 = emp_df_1_long.pivot_table(index =  ['Emp_Id','Emp_Name'] ,
                                           columns = 'Event',
                                           values = 'Bonus',
                                           margins = True,
                                           aggfunc = 'sum').reset_index()
emp_df_1_wide_3
```

[37]:
```
Event Emp_Id           Emp_Name  Diwali_Bonus  Holi_Bonus  Yearly_Bonus  \
0          1     Abhishek Kumar       84075.0     56050.0      112100.0
1          2        Arjun Kumar        8175.0      5450.0       10900.0
2          3          Vivek Raj       62025.0     41350.0       82700.0
3          4         Mika Singh           0.0         0.0           0.0
4          5     Anusha Yenduri       69075.0     46050.0       92100.0
5          6  Ritesh Srivastava       58875.0     39250.0       78500.0
6        All                         282225.0    188150.0      376300.0

Event        All
0       252225.0
1        24525.0
2       186075.0
3            NaN
4       207225.0
5       176625.0
6       846675.0
```

[38]:
```
# Only row-wise aggregation

emp_df_1_wide_4 = emp_df_1_long.pivot_table(index =  ['Emp_Id','Emp_Name']) #␣
 ↪default aggfunc = 'mean'
emp_df_1_wide_4
```

[38]:
```
                              Bonus
Emp_Id Emp_Name
1      Abhishek Kumar     84075.0
2      Arjun Kumar         8175.0
```

```
3      Vivek Raj          62025.0
5      Anusha Yenduri     69075.0
6      Ritesh Srivastava  58875.0
```

[39]:
```
emp_df_1_wide_4 = emp_df_1_long.pivot_table(index =  ['Emp_Id','Emp_Name'] ,
                                            columns = 'Event',
                                            values = 'Bonus',
                                            fill_value = 1000)
emp_df_1_wide_4
```

[39]:
```
Event                     Diwali_Bonus  Holi_Bonus  Yearly_Bonus
Emp_Id Emp_Name
1      Abhishek Kumar            84075       56050        112100
2      Arjun Kumar                8175        5450         10900
3      Vivek Raj                 62025       41350         82700
5      Anusha Yenduri            69075       46050         92100
6      Ritesh Srivastava         58875       39250         78500
```

2.0.5 There are other techniques that enables Re-Shaping of dataframes.

```
i. pivot()
ii. stack() & unstack()
iii. wide_to_long()
iv. crosstab()
v. cut()
```

[ ]:
```

```