Pandas Joining Method

21 October 2023 21:56

Pandas dataframe.groupby() Method

Pandas_groupby is used for grouping the data according to the categories and applying a function to the categories. It also helps to aggregate data efficiently. The Pandas groupby() is a very powerful function with a lot of variations. It makes the task of splitting the Dataframe over some criteria really easy and efficient.

Pandas dataframe.groupby()

Pandas **dataframe.groupby()** function is used to split the data into groups based on some criteria. Pandas objects can be split on any of their axes. The abstract definition of grouping is to provide a mapping of labels to group names.

Syntax: *DataFrame.groupby(by=None, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=False, **kwargs)*

Parameters :

- ***by :*** *mapping, function, str, or iterable*
- ***axis :*** *int, default 0*
- ***level :*** *If the axis is a MultiIndex (hierarchical), group by a particular level or levels*
- ***as_index :*** *For aggregated output, return object with group labels as the index. Only relevant for DataFrame input. as_index=False is effectively "SQL-style" grouped output*
- ***sort :*** *Sort group keys. Get better performance by turning this off. Note this does not influence the order of observations within each group. groupby preserves the order of rows within each group.*
- ***group_keys :*** *When calling apply, add group keys to index to identify pieces*
- ***squeeze :*** *Reduce the dimensionality of the return type if possible, otherwise return a consistent type*

Returns : *GroupBy object*

Dataset Used: For a link to the CSV file Used in Code, click here

Example 1: Use **groupby()** function to group the data based on the "Team".

- Python3

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

# Creating the dataframe
df = pd.read_csv("nba.csv")

# Print the dataframe
print(df.head())
```

Output:

```
             Name            Team  Number Position   Age Height  Weight            College
Salary

0  Avery Bradley  Boston Celtics     0.0       PG  25.0    6-2   180.0              Texas
7730337.0
1  Jae Crowder    Boston Celtics    99.0       SF  25.0    6-6   235.0          Marquette
6796117.0
2  John Holland   Boston Celtics    30.0       SG  27.0    6-5   205.0  Boston University
NaN
3  R.J. Hunter    Boston Celtics    28.0       SG  22.0    6-5   185.0      Georgia State
1148640.0
4  Jonas Jerebko  Boston Celtics     8.0       PF  29.0   6-10   231.0                NaN
5000000.0
```

Now apply the **groupby()** function.

- Python3

```
# applying groupby() function to
# group the data on team value.
gk = df.groupby('Team')

# Let's print the first entries
# in all the groups formed.
gk.first()
```

Output :

```
                                  Name  Number Position   Age Height   Weight
```

```
College      Salary
Team

Atlanta Hawks                  Kent Bazemore    24.0       SF  26.0    6-5    201.0           Old
Dominion   2000000.0

Boston Celtics                 Avery Bradley     0.0       PG  25.0    6-2    180.0
Texas   7730337.0

Brooklyn Nets               Bojan Bogdanovic    44.0       SG  27.0    6-8    216.0
Oklahoma State   3425510.0

Charlotte Hornets              Nicolas Batum     5.0       SG  27.0    6-8    200.0  Virginia
Commonwealth  13125306.0

Chicago Bulls               Cameron Bairstow    41.0       PF  25.0    6-9    250.0
New Mexico    845059.0

Cleveland Cavaliers       Matthew Dellavedova     8.0       PG  25.0    6-4    198.0
Saint Mary's   1147276.0

Dallas Mavericks             Justin Anderson     1.0       SG  22.0    6-6    228.0
Virginia   1449000.0

Denver Nuggets                Darrell Arthur     0.0       PF  28.0    6-9    235.0
Kansas   2814000.0

Detroit Pistons                 Joel Anthony    50.0        C  33.0    6-9    245.0
UNLV   2500000.0

Golden State Warriors         Leandro Barbosa    19.0       SG  33.0    6-3    194.0         North
Carolina   2500000.0

Houston Rockets                 Trevor Ariza     1.0       SF  30.0    6-8    215.0
UCLA   8193030.0

Indiana Pacers                   Lavoy Allen     5.0       PF  27.0    6-9    255.0
Temple   4050000.0

Los Angeles Clippers            Cole Aldrich    45.0        C  27.0   6-11    250.0
Kansas   1100602.0

Los Angeles Lakers              Brandon Bass     2.0       PF  31.0    6-8    250.0
LSU   3000000.0

Memphis Grizzlies               Jordan Adams     3.0       SG  21.0    6-5    209.0
UCLA   1404600.0

Miami Heat                       Chris Bosh     1.0       PF  32.0   6-11    235.0
Georgia Tech  22192730.0

Milwaukee Bucks         Giannis Antetokounmpo    34.0       SF  21.0   6-11    222.0
Arizona   1953960.0
```

Let's print the value contained in any one of the groups. For that use the name of the team. We use the function **get_group()** to find the entries contained in any of the groups.

- Python3

```
# Finding the values contained in the "Boston Celtics" group
gk.get_group('Boston Celtics')
```

Output :

```
               Name            Team  Number Position   Age Height  Weight            College
Salary
0     Avery Bradley  Boston Celtics     0.0       PG  25.0    6-2   180.0              Texas
7730337.0

1       Jae Crowder  Boston Celtics    99.0       SF  25.0    6-6   235.0          Marquette
6796117.0

2      John Holland  Boston Celtics    30.0       SG  27.0    6-5   205.0  Boston University
NaN

3       R.J. Hunter  Boston Celtics    28.0       SG  22.0    6-5   185.0      Georgia State
1148640.0

4     Jonas Jerebko  Boston Celtics     8.0       PF  29.0   6-10   231.0                NaN
5000000.0

5      Amir Johnson  Boston Celtics    90.0       PF  29.0    6-9   240.0                NaN
12000000.0

6     Jordan Mickey  Boston Celtics    55.0       PF  21.0    6-8   235.0                LSU
1170960.0

7      Kelly Olynyk  Boston Celtics    41.0        C  25.0    7-0   238.0            Gonzaga
2165160.0
```

```
8      Terry Rozier  Boston Celtics    12.0       PG  22.0    6-2   190.0         Louisville
1824360.0
9      Marcus Smart  Boston Celtics    36.0       PG  22.0    6-4   220.0     Oklahoma State
3431040.0
10  Jared Sullinger  Boston Celtics     7.0        C  24.0    6-9   260.0         Ohio State
2569260.0
11    Isaiah Thomas  Boston Celtics     4.0       PG  27.0    5-9   185.0         Washington
6912869.0
12      Evan Turner  Boston Celtics    11.0       SG  27.0    6-7   220.0         Ohio State
3425510.0
13      James Young  Boston Celtics    13.0       SG  20.0    6-6   215.0           Kentucky
1749840.0
14     Tyler Zeller  Boston Celtics    44.0        C  26.0    7-0   253.0     North Carolina
2616975.0
```

Example 2: Use **groupby()** function to form groups based on more than one category (i.e. Use more than one column to perform the splitting).

- Python3

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

# Creating the dataframe
df = pd.read_csv("nba.csv")

# First grouping based on "Team"
# Within each team we are grouping based on "Position"
gkk = df.groupby(['Team', 'Position'])

# Print the first value in each group
gkk.first()
```

Output :

```
                                      Name  Number   Age Height  Weight              College
Salary
Team               Position
Atlanta Hawks      C              Al Horford    15.0  30.0   6-10   245.0              Florida
12000000.0
                   PF          Kris Humphries    43.0  31.0    6-9   235.0            Minnesota
1000000.0
                   PG         Dennis Schroder    17.0  22.0    6-1   172.0          Wake Forest
1763400.0
                   SF           Kent Bazemore    24.0  26.0    6-5   201.0         Old Dominion
2000000.0
                   SG        Tim Hardaway Jr.    10.0  24.0    6-6   205.0             Michigan
1304520.0
...                                       ...     ...   ...    ...     ...                  ...
...
Washington Wizards C            Marcin Gortat    13.0  32.0   6-11   240.0 North Carolina State
11217391.0
                   PF             Drew Gooden    90.0  34.0   6-10   250.0               Kansas
3300000.0
                   PG          Ramon Sessions     7.0  30.0    6-3   190.0               Nevada
2170465.0
                   SF            Jared Dudley     1.0  30.0    6-7   225.0       Boston College
4375000.0
                   SG           Alan Anderson     6.0  33.0    6-6   220.0       Michigan State
4000000.0
```

Pandas Merging, Joining, and Concatenating

Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labelled 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. We can join, merge, and concat dataframe using different methods. In Dataframe `df.merge()`,`df.join()`, and `df.concat()` methods help in joining, merging and concating different dataframe.

Concatenating DataFrame

In order to concat dataframe, we use `concat()` function which helps in concatenating a dataframe. We can concat a dataframe in many different ways, they are:

- Concatenating DataFrame using `.concat()`
- Concatenating DataFrame by setting logic on axes
- Concatenating DataFrame using `.append()`
- Concatenating DataFrame by ignoring indexes
- Concatenating DataFrame with group keys
- Concatenating with mixed ndims

Concatenating DataFrame using `.concat()` :

In order to concat a dataframe, we use `.concat()` function this function concat a dataframe and returns a new dataframe.

```
# importing pandas module
import pandas as pd

# Define a dictionary containing employee data
data1 = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Define a dictionary containing employee data
data2 = {'Name':['Abhi', 'Ayushi', 'Dhiraj', 'Hitesh'],
        'Age':[17, 14, 12, 52],
        'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'],
        'Qualification':['Btech', 'B.A', 'Bcom', 'B.hons']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data1,index=[0, 1, 2, 3])

# Convert the dictionary into DataFrame
df1 = pd.DataFrame(data2, index=[4, 5, 6, 7])

print(df, "\n\n", df1)
```

Run on IDE

Now we apply `.concat` function in order to concat two dataframe

```
# using a .concat() method
frames = [df, df1]

res1 = pd.concat(frames)
res1
```

Output :

As shown in the output image, we have created two dataframe after concatenating we get one dataframe

	Name	Age	Address	Qualification
0	Jai	27	Nagpur	Msc
1	Princi	24	Kanpur	MA
2	Gaurav	22	Allahabad	MCA
3	Anuj	32	Kannuaj	Phd

	Name	Age	Address	Qualification
4	Abhi	17	Nagpur	Btech
5	Ayushi	14	Kanpur	B.A
6	Dhiraj	12	Allahabad	Bcom
7	Hitesh	52	Kannuaj	B.hons

	Name	Age	Address	Qualification
0	Jai	27	Nagpur	Msc
1	Princi	24	Kanpur	MA
2	Gaurav	22	Allahabad	MCA
3	Anuj	32	Kannuaj	Phd
4	Abhi	17	Nagpur	Btech
5	Ayushi	14	Kanpur	B.A
6	Dhiraj	12	Allahabad	Bcom
7	Hitesh	52	Kannuaj	B.hons

Concatenating DataFrame by setting logic on axes :

In order to concat dataframe, we have to set different logic on axes. We can set axes in the following three ways:

- Taking the union of them all, `join='outer'`. This is the default option as it results in zero information loss.
- Taking the intersection, `join='inner'`.
- Use a specific index, as passed to the `join_axes` argument

```
# importing pandas module
import pandas as pd

# Define a dictionary containing employee data
data1 = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd'],
        'Mobile No': [97, 91, 58, 76]}
```

```
# Define a dictionary containing employee data
data2 = {'Name':['Gaurav', 'Anuj', 'Dhiraj', 'Hitesh'],
        'Age':[22, 32, 12, 52],
        'Address':['Allahabad', 'Kannuaj', 'Allahabad', 'Kannuaj'],
        'Qualification':['MCA', 'Phd', 'Bcom', 'B.hons'],
        'Salary':[1000, 2000, 3000, 4000]}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data1,index=[0, 1, 2, 3])

# Convert the dictionary into DataFrame
df1 = pd.DataFrame(data2, index=[2, 3, 6, 7])

print(df, "\n\n", df1)
```

Run on IDE

Now we set axes `join = inner` for intersection of dataframe

```
# applying concat with axes
# join = 'inner'
res2 = pd.concat([df, df1], axis=1, join='inner')

res2
```

Output :

As shown in the output image, we get the intersection of dataframe

	Name	Age	Address	Qualification	Mobile No
0	Jai	27	Nagpur	Msc	97
1	Princi	24	Kanpur	MA	91
2	Gaurav	22	Allahabad	MCA	58
3	Anuj	32	Kannuaj	Phd	76

	Name	Age	Address	Qualification	Salary
2	Gaurav	22	Allahabad	MCA	1000
3	Anuj	32	Kannuaj	Phd	2000
6	Dhiraj	12	Allahabad	Bcom	3000
7	Hitesh	52	Kannuaj	B.hons	4000

	Name	Age	Address	Qualification	Mobile No	Name	Age	Address	Qualification	Salary
2	Gaurav	22	Allahabad	MCA	58	Gaurav	22	Allahabad	MCA	1000
3	Anuj	32	Kannuaj	Phd	76	Anuj	32	Kannuaj	Phd	2000

Now we set axes `join = outer` for union of dataframe.

```
# using a .concat for
# union of dataframe
res2 = pd.concat([df, df1], axis=1, sort=False)

res2
```

Output :

As shown in the output image, we get the union of dataframe

	Name	Age	Address	Qualification	Mobile No
0	Jai	27	Nagpur	Msc	97
1	Princi	24	Kanpur	MA	91
2	Gaurav	22	Allahabad	MCA	58
3	Anuj	32	Kannuaj	Phd	76

	Name	Age	Address	Qualification	Salary
2	Gaurav	22	Allahabad	MCA	1000
3	Anuj	32	Kannuaj	Phd	2000
6	Dhiraj	12	Allahabad	Bcom	3000
7	Hitesh	52	Kannuaj	B.hons	4000

	Name	Age	Address	Qualification	Mobile No	Name	Age	Address	Qualification	Salary
0	Jai	27.0	Nagpur	Msc	97.0	NaN	NaN	NaN	NaN	NaN
1	Princi	24.0	Kanpur	MA	91.0	NaN	NaN	NaN	NaN	NaN
2	Gaurav	22.0	Allahabad	MCA	58.0	Gaurav	22.0	Allahabad	MCA	1000.0
3	Anuj	32.0	Kannuaj	Phd	76.0	Anuj	32.0	Kannuaj	Phd	2000.0
6	NaN	NaN	NaN	NaN	NaN	Dhiraj	12.0	Allahabad	Bcom	3000.0
7	NaN	NaN	NaN	NaN	NaN	Hitesh	52.0	Kannuaj	B.hons	4000.0

Now we used a specific index, as passed to the `join_axes` argument

```
# using join_axes
res3 = pd.concat([df, df1], axis=1, join_axes=[df.index])

res3
```

Output :

	Name	Age	Address	Qualification	Mobile No
0	Jai	27	Nagpur	Msc	97
1	Princi	24	Kanpur	MA	91
2	Gaurav	22	Allahabad	MCA	58
3	Anuj	32	Kannuaj	Phd	76

	Name	Age	Address	Qualification	Salary
2	Gaurav	22	Allahabad	MCA	1000
3	Anuj	32	Kannuaj	Phd	2000
6	Dhiraj	12	Allahabad	Bcom	3000
7	Hitesh	52	Kannuaj	B.hons	4000

	Name	Age	Address	Qualification	Mobile No	Name	Age	Address	Qualification	Salary
0	Jai	27	Nagpur	Msc	97	NaN	NaN	NaN	NaN	NaN
1	Princi	24	Kanpur	MA	91	NaN	NaN	NaN	NaN	NaN
2	Gaurav	22	Allahabad	MCA	58	Gaurav	22.0	Allahabad	MCA	1000.0
3	Anuj	32	Kannuaj	Phd	76	Anuj	32.0	Kannuaj	Phd	2000.0

Concatenating DataFrame using `.append()`

In order to concat a dataframe, we use `.append()` function this function concatenate along axis=0, namely the index. This function exist before `.concat`.

```
# importing pandas module
import pandas as pd

# Define a dictionary containing employee data
data1 = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Define a dictionary containing employee data
data2 = {'Name':['Abhi', 'Ayushi', 'Dhiraj', 'Hitesh'],
        'Age':[17, 14, 12, 52],
        'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'],
        'Qualification':['Btech', 'B.A', 'Bcom', 'B.hons']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data1,index=[0, 1, 2, 3])

# Convert the dictionary into DataFrame
df1 = pd.DataFrame(data2, index=[4, 5, 6, 7])

print(df, "\n\n", df1)
```

Run on IDE

Now we apply `.append()` function inorder to concat to dataframe

```
# using append function

res = df.append(df1)
res
```

Output :

	Name	Age	Address	Qualification
0	Jai	27	Nagpur	Msc
1	Princi	24	Kanpur	MA
2	Gaurav	22	Allahabad	MCA
3	Anuj	32	Kannuaj	Phd

	Name	Age	Address	Qualification
4	Abhi	17	Nagpur	Btech
5	Ayushi	14	Kanpur	B.A
6	Dhiraj	12	Allahabad	Bcom
7	Hitesh	52	Kannuaj	B.hons

	Name	Age	Address	Qualification
0	Jai	27	Nagpur	Msc
1	Princi	24	Kanpur	MA
2	Gaurav	22	Allahabad	MCA
3	Anuj	32	Kannuaj	Phd
4	Abhi	17	Nagpur	Btech
5	Ayushi	14	Kanpur	B.A
6	Dhiraj	12	Allahabad	Bcom
7	Hitesh	52	Kannuaj	B.hons

Concatenating DataFrame by ignoring indexes :

In order to concat a dataframe by ignoring indexes, we ignore index which don't have a meaningful meaning, you may wish to append them and ignore the fact that they

may have overlapping indexes. In order to do that we use `ignore_index` as an argument.

```
# importing pandas module
import pandas as pd

# Define a dictionary containing employee data
data1 = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd'],
        'Mobile No': [97, 91, 58, 76]}
```

```
# Define a dictionary containing employee data
data2 = {'Name':['Gaurav', 'Anuj', 'Dhiraj', 'Hitesh'],
        'Age':[22, 32, 12, 52],
        'Address':['Allahabad', 'Kannuaj', 'Allahabad', 'Kannuaj'],
        'Qualification':['MCA', 'Phd', 'Bcom', 'B.hons'],
        'Salary':[1000, 2000, 3000, 4000]}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data1,index=[0, 1, 2, 3])

# Convert the dictionary into DataFrame
df1 = pd.DataFrame(data2, index=[2, 3, 6, 7])


print(df, "\n\n", df1)
```
Run on IDE

Now we are going to apply `ignore_index` as an argument.

```
# using ignore_index
res = pd.concat([df, df1], ignore_index=True)

res
```

Output :

	Name	Age	Address	Qualification	Mobile No
0	Jai	27	Nagpur	Msc	97
1	Princi	24	Kanpur	MA	91
2	Gaurav	22	Allahabad	MCA	58
3	Anuj	32	Kannuaj	Phd	76

	Name	Age	Address	Qualification	Salary
2	Gaurav	22	Allahabad	MCA	1000
3	Anuj	32	Kannuaj	Phd	2000
6	Dhiraj	12	Allahabad	Bcom	3000
7	Hitesh	52	Kannuaj	B.hons	4000

	Address	Age	Mobile No	Name	Qualification	Salary
0	Nagpur	27	97.0	Jai	Msc	NaN
1	Kanpur	24	91.0	Princi	MA	NaN
2	Allahabad	22	58.0	Gaurav	MCA	NaN
3	Kannuaj	32	76.0	Anuj	Phd	NaN
4	Allahabad	22	NaN	Gaurav	MCA	1000.0
5	Kannuaj	32	NaN	Anuj	Phd	2000.0
6	Allahabad	12	NaN	Dhiraj	Bcom	3000.0
7	Kannuaj	52	NaN	Hitesh	B.hons	4000.0

Concatenating DataFrame with group keys :

In order to concat dataframe with group keys, we override the column names with the use of the keys argument. Keys argument is to override the column names when creating a new DataFrame based on existing Series.


```
# importing pandas module
import pandas as pd

# Define a dictionary containing employee data
data1 = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Define a dictionary containing employee data
data2 = {'Name':['Abhi', 'Ayushi', 'Dhiraj', 'Hitesh'],
        'Age':[17, 14, 12, 52],
        'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'],
        'Qualification':['Btech', 'B.A', 'Bcom', 'B.hons']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data1,index=[0, 1, 2, 3])

# Convert the dictionary into DataFrame
df1 = pd.DataFrame(data2, index=[4, 5, 6, 7])

print(df, "\n\n", df1)
```
Run on IDE

Now we use keys as an argument.

```
# using keys
frames = [df, df1 ]

res = pd.concat(frames, keys=['x', 'y'])
res
```

Output :

	Name	Age	Address	Qualification
0	Jai	27	Nagpur	Msc
1	Princi	24	Kanpur	MA
2	Gaurav	22	Allahabad	MCA
3	Anuj	32	Kannuaj	Phd

	Name	Age	Address	Qualification
4	Abhi	17	Nagpur	Btech
5	Ayushi	14	Kanpur	B.A
6	Dhiraj	12	Allahabad	Bcom
7	Hitesh	52	Kannuaj	B.hons

		Name	Age	Address	Qualification
x	0	Jai	27	Nagpur	Msc
	1	Princi	24	Kanpur	MA
	2	Gaurav	22	Allahabad	MCA
	3	Anuj	32	Kannuaj	Phd
y	4	Abhi	17	Nagpur	Btech
	5	Ayushi	14	Kanpur	B.A
	6	Dhiraj	12	Allahabad	Bcom
	7	Hitesh	52	Kannuaj	B.hons

Concatenating with mixed ndims :

User can concatenate a mix of Series and DataFrame. The Series will be transformed to DataFrame with the column name as the name of the Series.

```
# importing pandas module
import pandas as pd

# Define a dictionary containing employee data
data1 = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data1,index=[0, 1, 2, 3])

# creating a series
s1 = pd.Series([1000, 2000, 3000, 4000], name='Salary')

print(df, "\n\n", s1)
```

Run on IDE

Now we are going to mix Series and dataframe together

```
# combining series and dataframe
res = pd.concat([df, s1], axis=1)

res
```

Output :

	Name	Age	Address	Qualification
0	Jai	27	Nagpur	Msc
1	Princi	24	Kanpur	MA
2	Gaurav	22	Allahabad	MCA
3	Anuj	32	Kannuaj	Phd

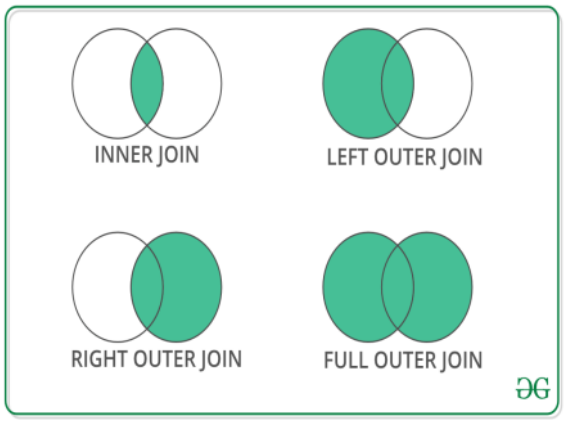
```
0    1000
1    2000
2    3000
3    4000
Name: Salary, dtype: int64
```

	Name	Age	Address	Qualification	Salary
0	Jai	27	Nagpur	Msc	1000
1	Princi	24	Kanpur	MA	2000
2	Gaurav	22	Allahabad	MCA	3000
3	Anuj	32	Kannuaj	Phd	4000

Merging DataFrame

Pandas have options for high-performance in-memory merging and joining. When we need to combine very large DataFrames, joins serve as a powerful way to perform these operations swiftly. Joins can only be done on two DataFrames at a time, denoted as left and right tables. The key is the common column that the two DataFrames will be joined on. It's a good practice to use keys which have unique values throughout the column to avoid unintended duplication of row values. Pandas provide a single function, merge(), as the entry point for all standard database join operations between DataFrame objects.

There are four basic ways to handle the join (inner, left, right, and outer), depending on which rows must retain their data.



Code #1 : Merging a dataframe with one unique key combination

```
# importing pandas module
import pandas as pd

# Define a dictionary containing employee data
data1 = {'key': ['K0', 'K1', 'K2', 'K3'],
        'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
       'Age':[27, 24, 22, 32],}

# Define a dictionary containing employee data
data2 = {'key': ['K0', 'K1', 'K2', 'K3'],
        'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'],
       'Qualification':['Btech', 'B.A', 'Bcom', 'B.hons']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data1)

# Convert the dictionary into DataFrame
df1 = pd.DataFrame(data2)


print(df, "\n\n", df1)
```

Run on IDE

Now we are using `.merge()` with one unique key combination

```
# using .merge() function
res = pd.merge(df, df1, on='key')

res
```

Output :

```
   key    Name  Age
0  K0     Jai   27
1  K1   Princi   24
2  K2   Gaurav   22
3  K3     Anuj   32


   key    Address Qualification
0  K0     Nagpur         Btech
1  K1     Kanpur           B.A
2  K2  Allahabad          Bcom
3  K3    Kannuaj        B.hons
```

	key	Name	Age	Address	Qualification
0	K0	Jai	27	Nagpur	Btech
1	K1	Princi	24	Kanpur	B.A
2	K2	Gaurav	22	Allahabad	Bcom
3	K3	Anuj	32	Kannuaj	B.hons

Code #2: Merging dataframe using multiple join keys.

```
# importing pandas module
import pandas as pd

# Define a dictionary containing employee data
data1 = {'key': ['K0', 'K1', 'K2', 'K3'],
        'key1': ['K0', 'K1', 'K0', 'K1'],
        'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
       'Age':[27, 24, 22, 32],}

# Define a dictionary containing employee data
data2 = {'key': ['K0', 'K1', 'K2', 'K3'],
        'key1': ['K0', 'K0', 'K0', 'K0'],
        'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'],
       'Qualification':['Btech', 'B.A', 'Bcom', 'B.hons']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data1)

# Convert the dictionary into DataFrame
df1 = pd.DataFrame(data2)


print(df, "\n\n", df1)
```

Run on IDE

Now we merge dataframe using multiple keys

```
# merging dataframe using multiple keys
res1 = pd.merge(df, df1, on=['key', 'key1'])

res1
```

Output :

```
  key key1    Name  Age
0  K0   K0     Jai   27
1  K1   K1  Princi   24
2  K2   K0  Gaurav   22
3  K3   K1    Anuj   32
```

```
  key key1    Address Qualification
0  K0   K0     Nagpur         Btech
1  K1   K0     Kanpur          B.A
2  K2   K0  Allahabad          Bcom
3  K3   K0    Kannuaj        B.hons
```

	key	key1	Name	Age	Address	Qualification
0	K0	K0	Jai	27	Nagpur	Btech
1	K2	K0	Gaurav	22	Allahabad	Bcom

Merging dataframe using `how` in an argument:

We use how argument to merge specifies how to determine which keys are to be included in the resulting table. If a key combination does not appear in either the left or right tables, the values in the joined table will be NA. Here is a summary of the how options and their SQL equivalent names:

MERGE METHOD	JOIN NAME	DESCRIPTION
left	LEFT OUTER JOIN	Use keys from left frame only
right	RIGHT OUTER JOIN	Use keys from right frame only
outer	FULL OUTER JOIN	Use union of keys from both frames
inner	INNER JOIN	Use intersection of keys from both frames

```
# importing pandas module
import pandas as pd

# Define a dictionary containing employee data
data1 = {'key': ['K0', 'K1', 'K2', 'K3'],
        'key1': ['K0', 'K1', 'K0', 'K1'],
        'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
       'Age':[27, 24, 22, 32],}

# Define a dictionary containing employee data
data2 = {'key': ['K0', 'K1', 'K2', 'K3'],
        'key1': ['K0', 'K0', 'K0', 'K0'],
        'Address':['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj'],
       'Qualification':['Btech', 'B.A', 'Bcom', 'B.hons']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data1)

# Convert the dictionary into DataFrame
df1 = pd.DataFrame(data2)


print(df, "\n\n", df1)
```

Run on IDE

Now we set `how = 'left'` in order to use keys from left frame only.

```
# using keys from left frame
res = pd.merge(df, df1, how='left', on=['key', 'key1'])

res
```

Output :

Left

```
   key key1    Name  Age
0  K0   K0     Jai   27
1  K1   K1  Princi   24
2  K2   K0  Gaurav   22
3  K3   K1    Anuj   32
```

Right

```
   key key1    Address Qualification
0  K0   K0     Nagpur         Btech
1  K1   K0     Kanpur           B.A
2  K2   K0  Allahabad          Bcom
3  K3   K0    Kannuaj        B.hons
```

	key	key1	Name	Age	Address	Qualification
0	K0	K0	Jai	27	Nagpur	Btech
1	K1	K1	Princi	24	NaN	NaN
2	K2	K0	Gaurav	22	Allahabad	Bcom
3	K3	K1	Anuj	32	NaN	NaN

Now we set `how = 'right'` in order to use keys from right frame only.

```
# using keys from right frame
res1 = pd.merge(df, df1, how='right', on=['key', 'key1'])

res1
```

Output :

Left

```
   key key1    Name  Age
0  K0   K0     Jai   27
1  K1   K1  Princi   24
2  K2   K0  Gaurav   22
3  K3   K1    Anuj   32
```

Right

```
   key key1    Address Qualification
0  K0   K0     Nagpur         Btech
1  K1   K0     Kanpur           B.A
2  K2   K0  Allahabad          Bcom
3  K3   K0    Kannuaj        B.hons
```

	key	key1	Name	Age	Address	Qualification
0	K0	K0	Jai	27.0	Nagpur	Btech
1	K2	K0	Gaurav	22.0	Allahabad	Bcom
2	K1	K0	NaN	NaN	Kanpur	B.A
3	K3	K0	NaN	NaN	Kannuaj	B.hons

Now we set `how = 'outer'` in order to get union of keys from dataframes.

```
# getting union  of keys
res2 = pd.merge(df, df1, how='outer', on=['key', 'key1'])

res2
```

Output :

Left

```
   key key1    Name  Age
0  K0   K0     Jai   27
1  K1   K1  Princi   24
2  K2   K0  Gaurav   22
3  K3   K1    Anuj   32
```

Right

```
   key key1    Address Qualification
0  K0   K0     Nagpur         Btech
1  K1   K0     Kanpur           B.A
2  K2   K0  Allahabad          Bcom
3  K3   K0    Kannuaj        B.hons
```

	key	key1	Name	Age	Address	Qualification
0	K0	K0	Jai	27.0	Nagpur	Btech
1	K1	K1	Princi	24.0	NaN	NaN
2	K2	K0	Gaurav	22.0	Allahabad	Bcom
3	K3	K1	Anuj	32.0	NaN	NaN
4	K1	K0	NaN	NaN	Kanpur	B.A
5	K3	K0	NaN	NaN	Kannuaj	B.hons

Now we set `how = 'inner'` in order to get intersection of keys from dataframes.

```
# getting intersection of keys
res3 = pd.merge(df, df1, how='inner', on=['key', 'key1'])

res3
```

Output :

Left

	key	key1	Name	Age
0	K0	K0	Jai	27
1	K1	K1	Princi	24
2	K2	K0	Gaurav	22
3	K3	K1	Anuj	32

Right

	key	key1	Address	Qualification
0	K0	K0	Nagpur	Btech
1	K1	K0	Kanpur	B.A
2	K2	K0	Allahabad	Bcom
3	K3	K0	Kannuaj	B.hons

	key	key1	Name	Age	Address	Qualification
0	K0	K0	Jai	27	Nagpur	Btech
1	K2	K0	Gaurav	22	Allahabad	Bcom

Joining DataFrame

In order to join dataframe, we use `.join()` function this function is used for combining the columns of two potentially differently-indexed DataFrames into a single result DataFrame.

```
# importing pandas module
import pandas as pd

# Define a dictionary containing employee data
data1 = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
      'Age':[27, 24, 22, 32]}

# Define a dictionary containing employee data
data2 = {'Address':['Allahabad', 'Kannuaj', 'Allahabad', 'Kannuaj'],
      'Qualification':['MCA', 'Phd', 'Bcom', 'B.hons']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data1,index=['K0', 'K1', 'K2', 'K3'])

# Convert the dictionary into DataFrame
df1 = pd.DataFrame(data2, index=['K0', 'K2', 'K3', 'K4'])


print(df, "\n\n", df1)
```

Run on IDE

Now we are use `.join()` method in order to join dataframes

```
# joining dataframe
res = df.join(df1)

res
```

Output :

Left

	Name	Age
K0	Jai	27
K1	Princi	24
K2	Gaurav	22
K3	Anuj	32

Right

	Address	Qualification
K0	Allahabad	MCA
K2	Kannuaj	Phd
K3	Allahabad	Bcom
K4	Kannuaj	B.hons

	Name	Age	Address	Qualification
K0	Jai	27	Allahabad	MCA
K1	Princi	24	NaN	NaN
K2	Gaurav	22	Kannuaj	Phd
K3	Anuj	32	Allahabad	Bcom

Now we use `how = 'outer'` in order to get union

```
# getting union
res1 = df.join(df1, how='outer')

res1
```

Output :

	Name	Age	Address	Qualification	Mobile No
0	Jai	27	Nagpur	Msc	97
1	Princi	24	Kanpur	MA	91
2	Gaurav	22	Allahabad	MCA	58
3	Anuj	32	Kannuaj	Phd	76

	Name	Age	Address	Qualification	Salary
2	Gaurav	22	Allahabad	MCA	1000
3	Anuj	32	Kannuaj	Phd	2000
6	Dhiraj	12	Allahabad	Bcom	3000
7	Hitesh	52	Kannuaj	B.hons	4000

	Name	Age	Address	Qualification	Mobile No	Name	Age	Address	Qualification	Salary
2	Gaurav	22	Allahabad	MCA	58	Gaurav	22	Allahabad	MCA	1000
3	Anuj	32	Kannuaj	Phd	76	Anuj	32	Kannuaj	Phd	2000

Joining dataframe using `on` in an argument :

In order to join dataframes we use `on` in an argument. `join()` takes an optional on argument which may be a column or multiple column names, which specifies that the passed DataFrame is to be aligned on that column in the DataFrame.

```
# importing pandas module
import pandas as pd

# Define a dictionary containing employee data
data1 = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Key':['K0', 'K1', 'K2', 'K3']}

# Define a dictionary containing employee data
data2 = {'Address':['Allahabad', 'Kannuaj', 'Allahabad', 'Kannuaj'],
        'Qualification':['MCA', 'Phd', 'Bcom', 'B.hons']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data1)

# Convert the dictionary into DataFrame
df1 = pd.DataFrame(data2, index=['K0', 'K2', 'K3', 'K4'])


print(df, "\n\n", df1)
```

Run on IDE

Now we are using `.join` with “on” argument

```
# using on argument in join
res2 = df.join(df1, on='Key')

res2
```

Output :

Left

	Name	Age	Key
0	Jai	27	K0
1	Princi	24	K1
2	Gaurav	22	K2
3	Anuj	32	K3

Right

	Address	Qualification
K0	Allahabad	MCA
K2	Kannuaj	Phd
K3	Allahabad	Bcom
K4	Kannuaj	B.hons

	Name	Age	Address	Qualification
K0	Jai	27.0	Allahabad	MCA
K1	Princi	24.0	NaN	NaN
K2	Gaurav	22.0	Kannuaj	Phd
K3	Anuj	32.0	Allahabad	Bcom
K4	NaN	NaN	Kannuaj	B.hons

Joining singly-indexed DataFrame with multi-indexed DataFrame :

In order to join singly indexed dataframe with multi-indexed dataframe, the level will match on the name of the index of the singly-indexed frame against a level name of the multi-indexed frame.

```
# importing pandas module
import pandas as pd

# Define a dictionary containing employee data
data1 = {'Name':['Jai', 'Princi', 'Gaurav'],
        'Age':[27, 24, 22]}

# Define a dictionary containing employee data
data2 = {'Address':['Allahabad', 'Kannuaj', 'Allahabad', 'Kanpur'],
        'Qualification':['MCA', 'Phd', 'Bcom', 'B.hons']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data1, index=pd.Index(['K0', 'K1', 'K2'], name='key'))
```

```
index = pd.MultiIndex.from_tuples([('K0', 'Y0'), ('K1', 'Y1'),
                                   ('K2', 'Y2'), ('K2', 'Y3')],
                                   names=['key', 'Y'])

# Convert the dictionary into DataFrame
df1 = pd.DataFrame(data2, index= index)


print(df, "\n\n", df1)
```
Run on IDE

Now we join singly indexed dataframe with multi-indexed dataframe

```
# joining singly indexed with
# multi indexed
result = df.join(df1, how='inner')

result
```

Output :

```
        Name  Age
key
K0       Jai   27
K1    Princi   24
K2    Gaurav   22


           Address Qualification
key Y
K0  Y0  Allahabad           MCA
K1  Y1    Kannuaj           Phd
K2  Y2  Allahabad          Bcom
    Y3     Kanpur        B.hons
```

key	Y	Name	Age	Address	Qualification
K0	**Y0**	Jai	27	Allahabad	MCA
K1	**Y1**	Princi	24	Kannuaj	Phd
K2	**Y2**	Gaurav	22	Allahabad	Bcom
	Y3	Gaurav	22	Kanpur	B.hons