Experiment No.: 6

Title: Implementation of dataframe merging using python.

Objectives:

1. To learn how to merge dataframe.

Theory:

Pandas has full-featured, high performance in-memory join operations idiomatically very similar to relational databases like SQL

Pandas provides a single function, **merge**, as the entry point for all standard database join operations between DataFrame objects

pd.merge(left, right, how='inner', on=None, left_on=None, right_on=None,left_index=False, right_index=False, sort=True)

Here, we have used the following parameters –

- **left** − A DataFrame object.
- **right** − Another DataFrame object.
- **on** − Columns (names) to join on. Must be found in both the left and right DataFrame objects.
- **left_on** − Columns from the left DataFrame to use as keys. Can either be column names or arrays with length equal to the length of the DataFrame.
- **right_on** − Columns from the right DataFrame to use as keys. Can either be column names or arrays with length equal to the length of the DataFrame.
- **left_index** − If **True,** use the index (row labels) from the left DataFrame as its join key(s). In case of a DataFrame with a MultiIndex (hierarchical), the number of levels must match the number of join keys from the right DataFrame.
- **right_index** − Same usage as **left_index** for the right DataFrame.
- **how** − One of 'left', 'right', 'outer', 'inner'. Defaults to inner. Each method has been described below.
- **sort** − Sort the result DataFrame by the join keys in lexicographical order. Defaults to True, setting to False will improve the performance substantially in many cases.

create two different DataFrames and perform the merging operations on it.

```
# import the pandas library
import pandas as pd
left = pd.DataFrame({
   'id':[1,2,3,4,5],
   'Name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
   'subject_id':['sub1','sub2','sub4','sub6','sub5']})
right = pd.DataFrame(
   {'id':[1,2,3,4,5],
   'Name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
   'subject_id':['sub2','sub4','sub3','sub6','sub5']})
print left
print right
```

Merge Two DataFrames on a Key

```
import pandas as pd
left = pd.DataFrame({
  'id':[1,2,3,4,5],
  'Name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
  'subject_id':['sub1','sub2','sub4','sub6','sub5']})
right = pd.DataFrame({
           'id':[1,2,3,4,5],
  'Name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
  'subject_id':['sub2','sub4','sub3','sub6','sub5']})
print pd.merge(left,right,on='id')
```

Merge Two DataFrames on Multiple Keys

```
import pandas as pd
left = pd.DataFrame({
  'id':[1,2,3,4,5],
  'Name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
  'subject_id':['sub1','sub2','sub4','sub6','sub5']})
right = pd.DataFrame({
           'id':[1,2,3,4,5],
  'Name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
  'subject_id':['sub2','sub4','sub3','sub6','sub5']})
print pd.merge(left,right,on=['id','subject_id'])
```

Merge Using 'how' Argument

The **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 the 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	SQL Equivalent	Description
left	LEFT OUTER JOIN	Use keys from left object

right	RIGHT OUTER JOIN	Use keys from right object
outer	FULL OUTER JOIN	Use union of keys
inner	INNER JOIN	Use intersection of keys

Left Join

```
import pandas as pd
left = pd.DataFrame({
   'id':[1,2,3,4,5],
   'Name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
   'subject_id':['sub1','sub2','sub4','sub6','sub5']})
right = pd.DataFrame({
   'id':[1,2,3,4,5],
   'Name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
   'subject_id':['sub2','sub4','sub3','sub6','sub5']})
print pd.merge(left, right, on='subject_id', how='left')
```

Right Join

```
import pandas as pd
left = pd.DataFrame({
   'id':[1,2,3,4,5],
   'Name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
   'subject_id':['sub1','sub2','sub4','sub6','sub5']})
right = pd.DataFrame({
   'id':[1,2,3,4,5],
   'Name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
   'subject_id':['sub2','sub4','sub3','sub6','sub5']})
print pd.merge(left, right, on='subject_id', how='right')
```

Outer Join

```
import pandas as pd
left = pd.DataFrame({
   'id':[1,2,3,4,5],
   'Name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
   'subject_id':['sub1','sub2','sub4','sub6','sub5']})
right = pd.DataFrame({
   'id':[1,2,3,4,5],
   'Name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
   'subject_id':['sub2','sub4','sub3','sub6','sub5']})
```

```
print pd.merge(left, right, how='outer', on='subject_id')
```

Inner Join

Joining will be performed on index. Join operation honors the object on which it is called. So, **a.join(b)** is not equal to **b.join(a)**.

```
import pandas as pd
left = pd.DataFrame({
   'id':[1,2,3,4,5],
   'Name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
   'subject_id':['sub1','sub2','sub4','sub6','sub5']})
right = pd.DataFrame({
   'id':[1,2,3,4,5],
   'Name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
   'subject_id':['sub2','sub4','sub3','sub6','sub5']})
print pd.merge(left, right, on='subject_id', how='inner')
```