Pandas Merge

The merge operation in Pandas merges two DataFrames based on their indexes or a specified column.

The `merge()` in Pandas works similar to JOINs in SQL.

Let's see an example.

```
import pandas as pd

# create dataframes from the dictionaries
data1 = {
    'EmployeeID' : ['E001', 'E002', 'E003', 'E004', 'E005'],
    'Name' : ['John Doe', 'Jane Smith', 'Peter Brown', 'Tom Johnson',
'Rita Patel'],
    'DeptID': ['D001', 'D003', 'D001', 'D002', 'D003'],
}
employees = pd.DataFrame(data1)

data2 = {
    'DeptID': ['D001', 'D002', 'D003'],
    'DeptName': ['Sales', 'HR', 'Admin']
}
departments = pd.DataFrame(data2)

# merge dataframes employees and departments
merged_df = pd.merge(employees, departments)

# display DataFrames
print("Employees:")
print(employees)
print()
print("Departments:")
print(departments)
print()
print("Merged DataFrame:")
print(merged_df)
```

```
Employees:
  EmployeeID         Name DeptID
0       E001     John Doe   D001
1       E002   Jane Smith   D003
2       E003  Peter Brown   D001
3       E004  Tom Johnson   D002
4       E005   Rita Patel   D003

Departments:
```

```
  DeptID DeptName
0   D001    Sales
1   D002       HR
2   D003    Admin

Merged DataFrame:
  EmployeeID         Name DeptID DeptName
0       E001     John Doe   D001    Sales
1       E003  Peter Brown   D001    Sales
2       E002   Jane Smith   D003    Admin
3       E005   Rita Patel   D003    Admin
4       E004  Tom Johnson   D002       HR
```

In this example, we merged the DataFrames `employees` and `departments` using the `merge()` method.

Notice that the two DataFrames are merged based on the `DeptID` column as it's common to both the DataFrames.

merge() Syntax in Pandas

The syntax of the `merge()` method in Pandas is:

```python pd.merge(left, right, on=None, how='inner', left_on=None, right_on=None, sort=False)

Here,

- **`left`**: specifies the left DataFrame to be merged.
- **`right`**: specifies the right DataFrame to be merged.
- **`on` (optional)**: specifies column(s) to join on.
- **`how` (optional)**: specifies the type of join to perform.
- **`left_on` (optional)**: specifies column(s) from the left DataFrame to use as key(s) for merging.
- **`right_on` (optional)**: specifies column(s) from the right DataFrame to use as key(s) for merging.
- **`sort` (optional)**: if `True`, sort the result DataFrame by the join keys.

## Example: Merge DataFrames Based on Keys

When there are no common columns between two DataFrames, we can merge them by specifying the columns (as keys) in the `left_on` and `right_on` arguments. For example,

```
import pandas as pd

# create dataframes from the dictionaries
data1 = {
    'EmployeeID': ['E001', 'E002', 'E003', 'E004', 'E005'],
```
```

```
    'Name': ['John Doe', 'Jane Smith', 'Peter Brown', 'Tom Johnson',
'Rita Patel'],
    'DeptID1': ['D001', 'D003', 'D001', 'D002', 'D006'],
}
employees = pd.DataFrame(data1)

data2 = {
    'DeptID2': ['D001', 'D002', 'D003', 'D004'],
    'DeptName': ['Sales', 'HR', 'Admin', 'Marketing']
}
departments = pd.DataFrame(data2)

# merge the dataframes
df_merge = pd.merge(employees, departments, left_on='DeptID1',
right_on = 'DeptID2', sort = True)

print(df_merge)
```

```
  EmployeeID         Name DeptID1 DeptID2 DeptName
0       E001     John Doe    D001    D001    Sales
1       E003  Peter Brown    D001    D001    Sales
2       E004  Tom Johnson    D002    D002       HR
3       E002   Jane Smith    D003    D003    Admin
```

In the above example, we performed a merge operation on two DataFrames, `employees` and `departments`, using the `merge()` method with various arguments.

Here, we used `DeptID1` and `DeptID2` as the key for merging the DataFrames. Then, we sorted the resulting DataFrame using `sort=True`.

Types of Join Operations in `merge()`

So far, we've not defined how to merge the DataFrames, thus it defaults to an inner join.

However, we can specify the join type in the `how` argument. Here are the 5 join types we can use in the `merge()` method:

1. **Left Join**
   - Includes all rows from the left DataFrame and matched rows from the right DataFrame. Rows from the right DataFrame that do not have a match will be filled with NaN values.
2. **Right Join**
   - Includes all rows from the right DataFrame and matched rows from the left DataFrame. Rows from the left DataFrame that do not have a match will be filled with NaN values.
3. **Outer Join**
   - Includes all rows from both DataFrames. Rows with no match will be filled with NaN values.

4. **Inner Join (Default)**
   - Includes only the rows that have matching values in both DataFrames.
5. **Cross Join**
   - Produces a Cartesian product of both DataFrames. Each row from the left DataFrame is paired with each row from the right DataFrame.

Left Join

A left join combines two DataFrames based on a common key and returns a new DataFrame that contains all rows from the left DataFrame and the matched rows from the right DataFrame.

If values are not found in the right DataFrame, it fills the space with `NaN`. For example:

```
import pandas as pd

# create dataframes from the dictionaries
data1 = {
    'EmployeeID': ['E001', 'E002', 'E003', 'E004', 'E005'],
    'Name': ['John Doe', 'Jane Smith', 'Peter Brown', 'Tom Johnson',
'Rita Patel'],
    'DeptID': ['D001', 'D003', 'D001', 'D002', 'D006'],
}
employees = pd.DataFrame(data1)

data2 = {
    'DeptID': ['D001', 'D002', 'D003', 'D004'],
    'DeptName': ['Sales', 'HR', 'Admin', 'Marketing']
}
departments = pd.DataFrame(data2)

# left merge the dataframes
df_merge = pd.merge(employees, departments, on = 'DeptID', how =
'left', sort = True)

print(df_merge)
```

```
  EmployeeID         Name DeptID DeptName
0       E001     John Doe   D001    Sales
1       E003  Peter Brown   D001    Sales
2       E004  Tom Johnson   D002       HR
3       E002   Jane Smith   D003    Admin
4       E005   Rita Patel   D006      NaN
```

Right Join

A right join is the opposite of a left join. It returns a new DataFrame that contains all rows from the right DataFrame and the matched rows from the left DataFrame.

If values are not found in the left dataframe, it fills the space with `NaN`. For example,

```
import pandas as pd

# create dataframes from the dictionaries
data1 = {
    'EmployeeID': ['E001', 'E002', 'E003', 'E004', 'E005'],
    'Name': ['John Doe', 'Jane Smith', 'Peter Brown', 'Tom Johnson',
'Rita Patel'],
    'DeptID': ['D001', 'D003', 'D001', 'D002', 'D006'],
}
employees = pd.DataFrame(data1)

data2 = {
    'DeptID': ['D001', 'D002', 'D003', 'D004'],
    'DeptName': ['Sales', 'HR', 'Admin', 'Marketing']
}
departments = pd.DataFrame(data2)

# right merge the dataframes
df_merge = pd.merge(employees, departments, on = 'DeptID', how =
'right', sort = True)

print(df_merge)
```

```
  EmployeeID         Name DeptID   DeptName
0       E001     John Doe   D001      Sales
1       E003  Peter Brown   D001      Sales
2       E004  Tom Johnson   D002         HR
3       E002   Jane Smith   D003      Admin
4        NaN          NaN   D004  Marketing
```

Inner Join

An inner join combines two DataFrames based on a common key and returns a new DataFrame that contains only rows that have matching values in both of the original DataFrames.

For example,

```
import pandas as pd

# create dataframes from the dictionaries
data1 = {
    'EmployeeID': ['E001', 'E002', 'E003', 'E004', 'E005'],
    'Name': ['John Doe', 'Jane Smith', 'Peter Brown', 'Tom Johnson',
'Rita Patel'],
    'DeptID': ['D001', 'D003', 'D001', 'D002', 'D006'],
}
employees = pd.DataFrame(data1)

data2 = {
    'DeptID': ['D001', 'D002', 'D003', 'D004'],
```

```
    'DeptName': ['Sales', 'HR', 'Admin', 'Marketing']
}
departments = pd.DataFrame(data2)

# inner merge the dataframes
df_merge = pd.merge(employees, departments, on = 'DeptID', how =
'inner', sort = True)

print(df_merge)
```

```
  EmployeeID         Name DeptID DeptName
0       E001     John Doe   D001    Sales
1       E003  Peter Brown   D001    Sales
2       E004  Tom Johnson   D002       HR
3       E002   Jane Smith   D003    Admin
```

Outer Join

An outer join combines two DataFrames based on a common key. Unlike an inner join, an outer join returns a new DataFrame that contains all rows from both original DataFrames.

If values are not found in the DataFrames, it fills the space with `NaN`.

For example,

```
import pandas as pd

# create dataframes from the dictionaries
data1 = {
    'EmployeeID': ['E001', 'E002', 'E003', 'E004', 'E005'],
    'Name': ['John Doe', 'Jane Smith', 'Peter Brown', 'Tom Johnson',
'Rita Patel'],
    'DeptID': ['D001', 'D003', 'D001', 'D002', 'D006'],
}
employees = pd.DataFrame(data1)

data2 = {
    'DeptID': ['D001', 'D002', 'D003', 'D004'],
    'DeptName': ['Sales', 'HR', 'Admin', 'Marketing']
}
departments = pd.DataFrame(data2)

# outer merge the dataframes
df_merge = pd.merge(employees, departments, on = 'DeptID', how =
'outer', sort = True)

print(df_merge)
```

```
  EmployeeID         Name DeptID   DeptName
0       E001     John Doe   D001      Sales
```

```
1       E003  Peter Brown   D001      Sales
2       E004  Tom Johnson   D002         HR
3       E002   Jane Smith   D003      Admin
4        NaN          NaN   D004  Marketing
5       E005   Rita Patel   D006        NaN
```

Cross Join

A cross join in Pandas creates the cartesian product of both DataFrames while preserving the order of the left DataFrame.

For example,

```
import pandas as pd

# create dataframes from the dictionaries
data1 = {
    'EmployeeID': ['E001', 'E002', 'E003', 'E004', 'E005'],
    'Name': ['John Doe', 'Jane Smith', 'Peter Brown', 'Tom Johnson',
'Rita Patel'],
    'DeptID': ['D001', 'D003', 'D001', 'D002', 'D006'],
}
employees = pd.DataFrame(data1)

data2 = {
    'DeptID': ['D001', 'D002', 'D003', 'D004'],
    'DeptName': ['Sales', 'HR', 'Admin', 'Marketing']
}
departments = pd.DataFrame(data2)

# merge the dataframes
df_merge = pd.merge(employees, departments, how = 'cross')

print(df_merge)
```

```
   EmployeeID         Name DeptID_x DeptID_y   DeptName
0        E001     John Doe     D001     D001      Sales
1        E001     John Doe     D001     D002         HR
2        E001     John Doe     D001     D003      Admin
3        E001     John Doe     D001     D004  Marketing
4        E002   Jane Smith     D003     D001      Sales
5        E002   Jane Smith     D003     D002         HR
6        E002   Jane Smith     D003     D003      Admin
7        E002   Jane Smith     D003     D004  Marketing
8        E003  Peter Brown     D001     D001      Sales
9        E003  Peter Brown     D001     D002         HR
10       E003  Peter Brown     D001     D003      Admin
11       E003  Peter Brown     D001     D004  Marketing
12       E004  Tom Johnson     D002     D001      Sales
```

```
13       E004  Tom Johnson     D002     D002         HR
14       E004  Tom Johnson     D002     D003      Admin
15       E004  Tom Johnson     D002     D004  Marketing
16       E005   Rita Patel     D006     D001      Sales
17       E005   Rita Patel     D006     D002         HR
18       E005   Rita Patel     D006     D003      Admin
19       E005   Rita Patel     D006     D004  Marketing
```

Join vs Merge vs Concat

There are three different methods to combine DataFrames in Pandas:

`join()`: joins two DataFrames based on their indexes, performs left join by default `merge()`: joins two DataFrames based on any specified columns, performs inner join by default `concat()`: stacks two DataFrames along the vertical or horizontal axis

```
```