17. Pandas – DataFrame - Merging or Joining

Contents

17. Pandas – DataFrame - Merging or Joining

1. Introduction

- ✓ By using pandas we can perform join operations as well.
- ✓ This is same as join operations in database.
- ✓ Here we are performing join in between the DataFrames
- ✓ Joining is the process of bringing two DataFrames into one DataFrame based on common attributes in columns

2. merge(p1, p2, p3, p4) function

- ✓ merge(p1, p2, p3, p4) is a predefined function in pandas.
  - pd.merge(df1, df2, on = "column", how = "type of join")
- ✓ We should access this function by using pandas library
- ✓ By using this function we can perform join operations over DataFrames

'how' Argument

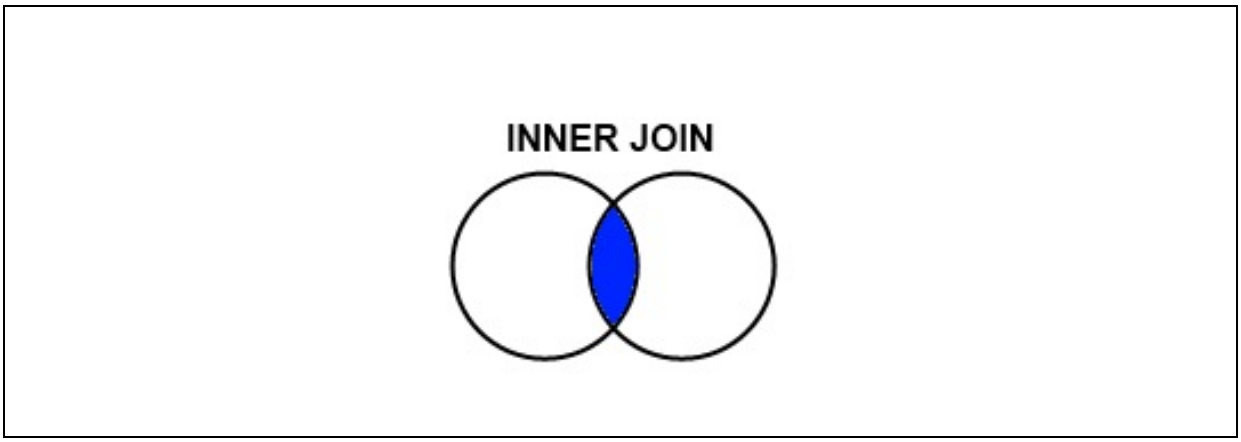
- ✓ The how argument helps to specify the type of join.

3. Types of joins

- ✓ Inner Merge / Inner join
- ✓ Left Merge / Left outer join
- ✓ Right Merge / Right outer join
- ✓ Outer Merge / Full outer join

3.1. Inner join

- ✓ We can perform inner join on DataFrames.
- ✓ This inner join is a kind of intersection means, it keeps the common data.



merge(df1, df2, on = "column", how = "type")

- ✓ merge(df1, df2, on = "column", how = "type") is predefined function in pandas.
- ✓ To perform inner join, we need to pass how value as "inner" as per the syntax

Syntax

```
pd.merge(df1, df2, on = "column", how = "inner")
```

Program Name: Creating two DataFrames
demo1.py

```
import pandas as pd

d1 = {
        "Id": [1, 2, 3, 4, 5, 6],
        "Name": ["Pradhan", "Venu", "Madhurima", "Nireekshan",
        "Shafi", "Veeru"],
        "Subject": ["English", "Java", "Html", "Python", "C", "dot
        net"]
}

d2 = {
        "Id": [11, 12, 13, 14, 15, 16],
        "Name": ["Srinu", "Sumanth", "Neelima", "Daniel", "Arjun",
        "Veeru"],
        "Subject": ["Java", "Html", "Cpp", "Python", "C", "dot net"]
}

df1 = pd.DataFrame(d1)
df2 = pd.DataFrame(d2)

print(df1)
print()
print(df2)
```

Output

```
   Id        Name  Subject
0   1     Pradhan  English
1   2        Venu     Java
2   3   Madhurima     Html
3   4  Nireekshan   Python
4   5       Shafi        C
5   6       Veeru  dot net

   Id     Name  Subject
0  11    Srinu     Java
1  12  Sumanth     Html
2  13  Neelima      Cpp
3  14   Daniel   Python
4  15    Arjun        C
5  16    Veeru  dot net
```

Program Name: Inner Join
demo2.py

```
import pandas as pd

d1 = {
        "Id":[1, 2, 3, 4, 5, 6],
        "Name": ["Pradhan", "Venu", "Madhurima", "Nireekshan",
        "Shafi", "Veeru"],
        "Subject":["English", "Java", "Html", "Python", "C", "dot
        net"]
}

d2 = {
        "Id":[11, 12, 13, 14, 15, 16],
        "Name": ["Srinu", "Sumanth", "Neelima", "Daniel", "Arjun",
        "Veeru"],
        "Subject":["Java", "Html", "Cpp", "Python", "C", "dot net"]
}

df1 = pd.DataFrame(d1)
df2 = pd.DataFrame(d2)

inn_join = pd.merge(df1, df2, on = "Subject", how = "inner")

print(df1)
print()
print(df2)
print()
print(inn_join)
```

Output

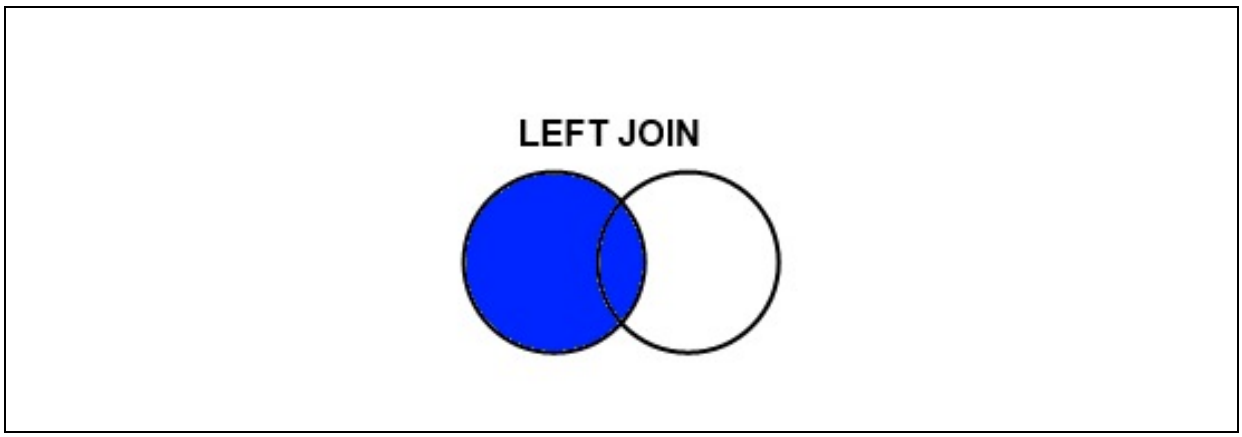
```
   Id        Name  Subject
0   1     Pradhan  English
1   2        Venu     Java
2   3   Madhurima     Html
3   4  Nireekshan   Python
4   5       Shafi        C
5   6       Veeru  dot net

   Id     Name  Subject
0  11    Srinu     Java
1  12  Sumanth     Html
2  13  Neelima      Cpp
3  14   Daniel   Python
4  15    Arjun        C
5  16    Veeru  dot net

   Id_x      Name_x  Subject  Id_y   Name_y
0     2        Venu     Java    11    Srinu
1     3   Madhurima     Html    12  Sumanth
2     4  Nireekshan   Python    14   Daniel
3     5       Shafi        C    15    Arjun
4     6       Veeru  dot net    16    Veeru
```

3.2. Left join

- ✓ Keep every row in the left dataframe.
- ✓ Where there are missing values of the "on" variable in the right dataframe filled with NaN values in the result.



Program Name: Left Join
demo3.py

```
import pandas as pd

d1 = {
        "Id":[1, 2, 3, 4, 5, 6],
        "Name": ["Pradhan", "Venu", "Madhurima", "Nireekshan",
        "Shafi", "Veeru"],
        "Subject":["English", "Java", "Html", "Python", "C", "dot
        net"]
}

d2 = {
        "Id":[11, 12, 13, 14, 15, 16],
        "Name": ["Srinu", "Sumanth", "Neelima", "Daniel", "Arjun",
        "Veeru"],
        "Subject":["Java", "Html", "Cpp", "Python", "C", "dot net"]
}

df1 = pd.DataFrame(d1)
df2 = pd.DataFrame(d2)

left_join = pd.merge(df1, df2, on = "Subject", how = "left")

print(df1)
print()
print(df2)
print()
print(left_join)
```

Output

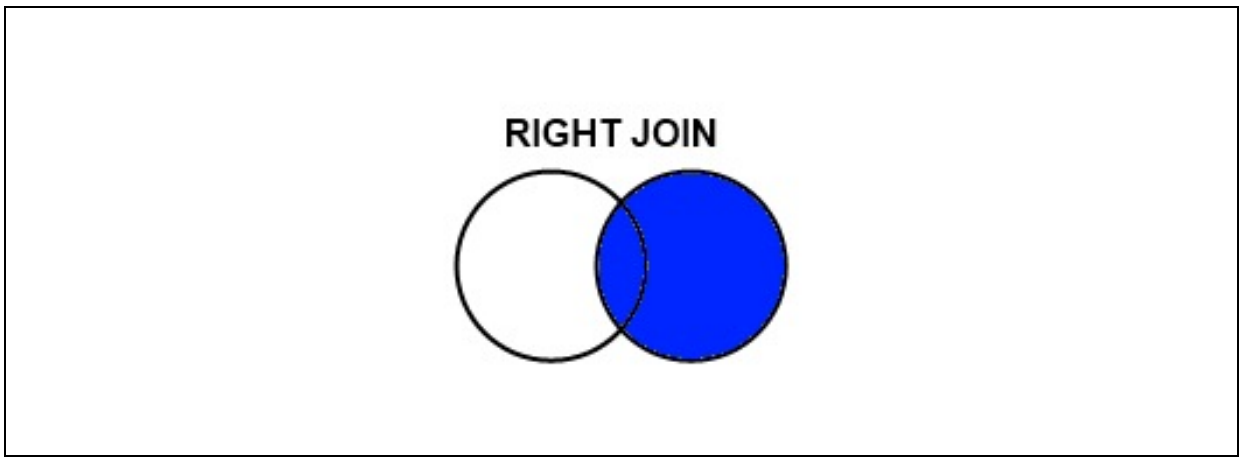
```
   Id        Name  Subject
0   1     Pradhan  English
1   2        Venu     Java
2   3   Madhurima     Html
3   4  Nireekshan   Python
4   5       Shafi        C
5   6       Veeru  dot net

   Id     Name  Subject
0  11    Srinu     Java
1  12  Sumanth     Html
2  13  Neelima      Cpp
3  14   Daniel   Python
4  15    Arjun        C
5  16    Veeru  dot net

   Id_x      Name_x  Subject  Id_y   Name_y
0     1     Pradhan  English   NaN      NaN
1     2        Venu     Java  11.0    Srinu
2     3   Madhurima     Html  12.0  Sumanth
3     4  Nireekshan   Python  14.0   Daniel
4     5       Shafi        C  15.0    Arjun
5     6       Veeru  dot net  16.0    Veeru
```

3.3. Right join

- ✓ Keep every row in the right dataframe.
- ✓ Where there are missing values of the "on" variable in the left column filled with NaN values in the result.



Program Name: Right Join
demo4.py

```
import pandas as pd

d1 = {
        "Id":[1, 2, 3, 4, 5, 6],
        "Name": ["Pradhan", "Venu", "Madhurima", "Nireekshan",
        "Shafi", "Veeru"],
        "Subject":["English", "Java", "Html", "Python", "C", "dot
        net"]
}

d2 = {
        "Id":[11, 12, 13, 14, 15, 16],
        "Name": ["Srinu", "Sumanth", "Neelima", "Daniel", "Arjun",
        "Veeru"],
        "Subject":["Java", "Html", "Cpp", "Python", "C", "dot net"]
}

df1 = pd.DataFrame(d1)
df2 = pd.DataFrame(d2)

right_join = pd.merge(df1, df2, on = "Subject", how = "right")

print(df1)
print()
print(df2)
print()
print(right_join)
```

Output

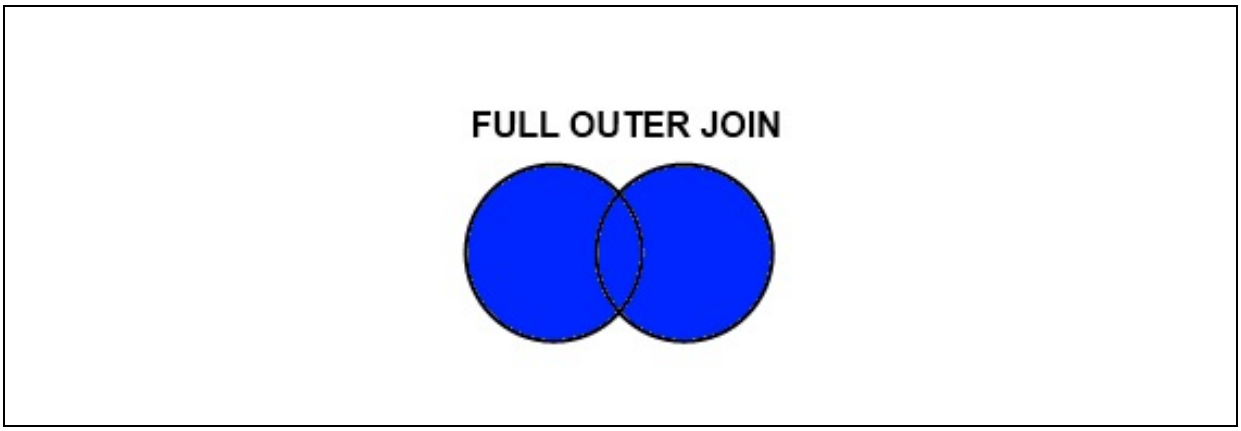
```
   Id        Name  Subject
0   1     Pradhan  English
1   2        Venu     Java
2   3   Madhurima     Html
3   4  Nireekshan   Python
4   5       Shafi        C
5   6       Veeru  dot net

   Id      Name  Subject
0  11    Srinu      Java
1  12  Sumanth      Html
2  13  Neelima       Cpp
3  14   Daniel    Python
4  15    Arjun         C
5  16    Veeru  dot net

   Id_x      Name_x  Subject  Id_y   Name_y
0   2.0        Venu     Java    11    Srinu
1   3.0   Madhurima     Html    12  Sumanth
2   NaN         NaN      Cpp    13  Neelima
3   4.0  Nireekshan   Python    14   Daniel
4   5.0       Shafi        C    15    Arjun
5   6.0       Veeru  dot net    16    Veeru
```

3.4. Outer Merge / Full outer join

- ✓ A full outer join returns all the rows from the left and right DataFrames.
- ✓ Where there is no common data there it will be fills with NaN values.



Program Name: Outer Join
demo5.py

```
import pandas as pd

d1 = {
        "Id":[1, 2, 3, 4, 5, 6],
        "Name": ["Pradhan", "Venu", "Madhurima", "Nireekshan",
        "Shafi", "Veeru"],
        "Subject":["English", "Java", "Html", "Python", "C", "dot
        net"]
}

d2 = {
        "Id":[11, 12, 13, 14, 15, 16],
        "Name": ["Srinu", "Sumanth", "Neelima", "Daniel", "Arjun",
        "Veeru"],
        "Subject":["Java", "Html", "Cpp", "Python", "C", "dot net"]
}

df1 = pd.DataFrame(d1)
df2 = pd.DataFrame(d2)

outer_join = pd.merge(df1, df2, on = "Subject", how = "outer")

print(df1)
print()
print(df2)
print()
print(outer_join)
```

Output

```
   Id        Name  Subject
0   1     Pradhan  English
1   2        Venu     Java
2   3   Madhurima     Html
3   4  Nireekshan   Python
4   5      Shafi        C
5   6      Veeru  dot net

   Id     Name  Subject
0  11    Srinu     Java
1  12  Sumanth     Html
2  13  Neelima      Cpp
3  14   Daniel   Python
4  15    Arjun        C
5  16    Veeru  dot net

   Id_x      Name_x  Subject  Id_y   Name_y
0   1.0     Pradhan  English   NaN      NaN
1   2.0        Venu     Java  11.0    Srinu
2   3.0   Madhurima     Html  12.0  Sumanth
3   4.0  Nireekshan   Python  14.0   Daniel
4   5.0       Shafi        C  15.0    Arjun
5   6.0       Veeru  dot net  16.0    Veeru
6   NaN         NaN      Cpp  13.0  Neelima
```

4. Other type of joins

- ✓ One to one
- ✓ Many to one
- ✓ Many to many

Program Name: Creating DataFrames
demo6.py

```
import pandas as pd

d1 = {
   "Employee": ["Nireekshan", "Veeru", "Lavanya", "Pradhan"],
   "Group": ["Development", "Testing", "Testing", "HR"]
}

d2 = {
   "Employee": ["Lavanya", "Nireekshan", "Veeru", "Pradhan"],
   "Hire_date": [2010, 2012, 2014, 2016]
}

df1 = pd.DataFrame(d1)
df2 = pd.DataFrame(d2)

print(df1)
print()
print(df2)
```

Output

```
     Employee        Group
0  Nireekshan  Development
1       Veeru      Testing
2     Lavanya      Testing
3     Pradhan           HR

     Employee  Hire_date
0     Lavanya       2010
1  Nireekshan       2012
2       Veeru       2014
3     Pradhan       2016
```

4.1. One to one

- ✓ This is very simple join and similar to the column-wise concatenation

Program Name: Creating DataFrames
demo7.py

```
import pandas as pd

d1 = {
   "Employee": ["Nireekshan", "Veeru", "Lavanya", "Pradhan"],
   "Group": ["Development", "Testing", "Testing", "HR"]
}

d2 = {
   "Employee": ["Lavanya", "Nireekshan", "Veeru", "Pradhan"],
   "Hire_date": [2010, 2012, 2014, 2016]
}

df1 = pd.DataFrame(d1)
df2 = pd.DataFrame(d2)

one_one = pd.merge(df1, df2)

print(df1)
print()
print(df2)
print()
print(one_one)
```

Output

```
     Employee         Group
0  Nireekshan  Development
1       Veeru      Testing
2     Lavanya      Testing
3     Pradhan           HR

     Employee  Hire_date
0     Lavanya       2010
1  Nireekshan       2012
2       Veeru       2014
3     Pradhan       2016

     Employee         Group  Hire_date
0  Nireekshan  Development       2012
1       Veeru      Testing       2014
2     Lavanya      Testing       2010
3     Pradhan           HR       2016
```

4.2. Many to one

- ✓ Many-to-one joins are joins in which one of the two key columns contains duplicate entries

Program Name: Many to one
demo8.py

```
import pandas as pd

d1 = {
   "Employee": ["Nireekshan", "Veeru", "Lavanya", "Pradhan"],
   "Group": ["Development", "Testing", "Testing", "HR"]
}

d2 = {
   "Employee": ["Lavanya", "Nireekshan", "Veeru", "Pradhan"],
   "Hire_date": [2010, 2012, 2014, 2016]
}

d3 = {
   "Group": ["Testing", "Development", "HR"],
   "supervisor": ["Shafi", "Daniel", "Neelima"]
}

df1 = pd.DataFrame(d1)
df2 = pd.DataFrame(d2)
df3 = pd.DataFrame(d3)

one_one = pd.merge(df1, df2)

many_one = pd.merge(one_one, df3)

print(df1)
print()
print(df2)
print()
print(many_one)
```

Output

```
     Employee        Group
0  Nireekshan  Development
1       Veeru      Testing
2     Lavanya      Testing
3     Pradhan           HR

     Employee  Hire_date
0     Lavanya       2010
1  Nireekshan       2012
2       Veeru       2014
3     Pradhan       2016

     Employee        Group  Hire_date supervisor
0  Nireekshan  Development       2012     Daniel
1       Veeru      Testing       2014      Shafi
2     Lavanya      Testing       2010      Shafi
3     Pradhan           HR       2016    Neelima
```

4.3. Many to many

- ✓ If the key column in both the left and right DataFrame contains duplicates, then the result is a many-to-many merge

Program Name: Many to many
demo9.py

```
import pandas as pd

d1 = {
    "Employee": ["Nireekshan", "Veeru", "Lavanya", "Pradhan"],
    "Group": ["Development", "Testing", "Testing", "HR"]
}

d2 = {
    "Group": ["Testing", "Testing", "Development",
    "Development", "HR", "HR"],
    "Skills": ["Manual", "Automation", "Coding", "Logical",
    "Spreadsheets", "Organization"]
}


df1 = pd.DataFrame(d1)
df2 = pd.DataFrame(d2)

many_many = pd.merge(df1, df2)

print(df1)
print()
print(df2)
print()
print(many_many)
```

Output

```
     Employee        Group
0  Nireekshan  Development
1       Veeru      Testing
2     Lavanya      Testing
3     Pradhan           HR

         Group        Skills
0      Testing        Manual
1      Testing    Automation
2  Development        Coding
3  Development       Logical
4           HR  Spreadsheets
5           HR  Organization

     Employee        Group        Skills
0  Nireekshan  Development        Coding
1  Nireekshan  Development       Logical
2       Veeru      Testing        Manual
3       Veeru      Testing    Automation
4     Lavanya      Testing        Manual
5     Lavanya      Testing    Automation
6     Pradhan           HR  Spreadsheets
7     Pradhan           HR  Organization
```

Based on column wise

- ✓ We can also do merge DataFrames based on columns wise as well

Program Name: Creating DataFrames
demo10.py

```
import pandas as pd

d1 = {
        "Id":[1, 2, 3, 4, 5, 6],
        "Name": ["Pradhan", "Venu", "Madhurima", "Nireekshan",
        "Shafi", "Veeru"],
        "Subject":["English", "Java", "Html", "Python", "C", "dot
        net"]
}

d2 = {
        "Id":[11, 12, 13, 14, 15, 16],
        "Name": ["Srinu", "Sumanth", "Neelima", "Daniel", "Arjun",
        "Veeru"],
        "Subject":["Java", "Html", "Cpp", "Python", "C", "dot net"]
}

df1 = pd.DataFrame(d1)
df2 = pd.DataFrame(d2)

print(df1)
print()
print(df2)
```

Output

```
   Id        Name  Subject
0   1     Pradhan  English
1   2        Venu     Java
2   3   Madhurima     Html
3   4  Nireekshan   Python
4   5       Shafi        C
5   6       Veeru  dot net

   Id     Name  Subject
0  11    Srinu     Java
1  12  Sumanth     Html
2  13  Neelima      Cpp
3  14   Daniel   Python
4  15    Arjun        C
5  16    Veeru  dot net
```

Program Name: Merging two DataFrames based on column
demo11.py

```
import pandas as pd

d1 = {
        "Id":[1, 2, 3, 4, 5, 6],
        "Name": ["Pradhan", "Venu", "Madhurima", "Nireekshan",
        "Shafi", "Veeru"],
        "Subject":["English", "Java", "Html", "Python", "C", "dot
        net"]
}

d2 = {
        "Id":[11, 12, 13, 14, 15, 16],
        "Name": ["Srinu", "Sumanth", "Neelima", "Daniel", "Arjun",
        "Veeru"],
        "Subject":["Java", "Html", "Cpp", "Python", "C", "dot net"]
}

df1 = pd.DataFrame(d1)
df2 = pd.DataFrame(d2)

result = pd.merge(df1, df2, on = 'Subject')

print(df1)
print()
print(df2)
print()
print(result)
```

Output

```
   Id        Name  Subject
0   1     Pradhan  English
1   2        Venu     Java
2   3   Madhurima     Html
3   4  Nireekshan   Python
4   5       Shafi        C
5   6       Veeru  dot net

   Id     Name  Subject
0  11    Srinu     Java
1  12  Sumanth     Html
2  13  Neelima      Cpp
3  14   Daniel   Python
4  15    Arjun        C
5  16    Veeru  dot net

   Id_x      Name_x  Subject  Id_y   Name_y
0     2        Venu     Java    11    Srinu
1     3   Madhurima     Html    12  Sumanth
2     4  Nireekshan   Python    14   Daniel
3     5       Shafi        C    15    Arjun
4     6       Veeru  dot net    16    Veeru
```

Program Name: Merging two DataFrames with multiple columns
demo12.py

```
import pandas as pd

d1 = {
        "Id":[1, 2, 3, 4, 5, 6],
        "Name": ["Pradhan", "Venu", "Madhurima", "Nireekshan",
        "Shafi", "Veeru"],
        "Subject":["English", "Java", "Html", "Python", "C", "dot
        net"]
}

d2 = {
        "Id":[11, 12, 13, 14, 15, 16],
        "Name": ["Srinu", "Sumanth", "Neelima", "Daniel", "Arjun",
        "Veeru"],
        "Subject":["Java", "Html", "Cpp", "Python", "C", "dot net"]
}

df1 = pd.DataFrame(d1)
df2 = pd.DataFrame(d2)

result = pd.merge(df1, df2, on=["Name", "Subject"])

print(df1)
print()
print(df2)
print()
print(result)
```

Output

```
   Id        Name  Subject
0   1     Pradhan  English
1   2        Venu     Java
2   3   Madhurima     Html
3   4  Nireekshan   Python
4   5       Shafi        C
5   6       Veeru  dot net

   Id     Name  Subject
0  11    Srinu     Java
1  12  Sumanth     Html
2  13  Neelima      Cpp
3  14   Daniel   Python
4  15    Arjun        C
5  16    Veeru  dot net

   Id_x   Name  Subject  Id_y
0     6  Veeru  dot net    16
```