Data Management in Python –

Merging / Appending & Aggregating Data

Contents

Data Snapshot

sal_data consist information about Employee's Basic Salary, their ID & full Name

Employee_ID	First_Name	Last_Name	Basic_Salary
E-1001	Mahesh	Joshi	16860
E-1002	Ra		
E-1004	Pr		
E-1005	Sn		
E-1007	R		
E-1008	N		
E-1009	Har		

Columns	Description	Type	Measurement	Possible values
Employee_ID	Employee ID	character	-	-
First_Name	First Name	character	-	-
Last_Name	Last Name	character	-	-
Basic_Salary	Basic Salary	numeric	Rs.	positive values

bonus_data has information of only Bonus given to Employees.

"Employee ID" is the common column in both datasets

Employee_ID	Bonus
E-1001	16070
E-1003	
E-1004	
E-1006	
E-1008	
E-1010	

Columns	Description	Type	Measurement	Possible values
Employee_ID	Employee ID	character	-	-
Bonus	Bonus amount	Numeric	Rs.	Positive values

Merging

pandas provide various facilities for easily combining together Series, DataFrame objects with various kinds of set logic for the indexes and relational algebra functionality in the case of join / merge-type operations.

We can use the simple function `merge()` as our entry-point to merging data in pandas.

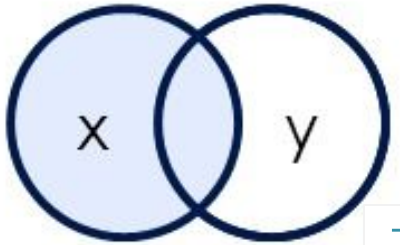
```
# Import sal_data and bonus_data
import pandas as pd

sal_data = pd.read_csv('sal_data.csv')
bonus_data = pd.read_csv('bonus_data.csv')
```

Types of Joins

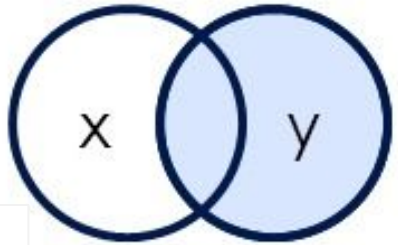
Consider sal_data =x and bonus_data=y

There are 4 types of Joins

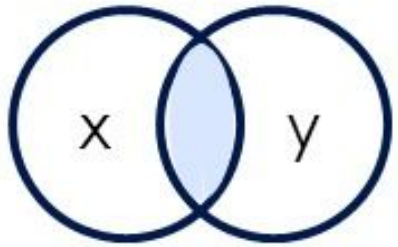
Left Join



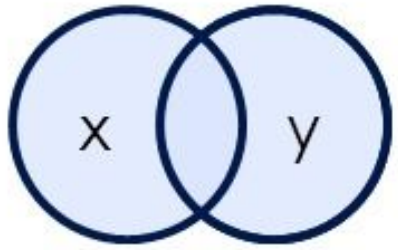
Right Join



Inner Join



Outer Join



Left Join

Left Join returns all rows from the left table, and any rows with matching keys from the right table.

```
# Display all the information(including bonus) of Employees from sal_data
leftjoin=pd.merge(sal_data,bonus_data,how='left')
leftjoin
```

how= is used to specify the type of join, in this case left.

```
 # Output
```

	Employee_ID	First_Name	Last_Name	Basic_Salary	Bonus
0	E-1001	Mahesh	Joshi	16860	16070.0
1	E-1002	Rajesh	Kolte	14960	NaN
2	E-1004	Priya	Jain	12670	13490.0
3	E-1005	Sneha	Joshi	15660	NaN
4	E-1007	Ram	Kanade	15850	NaN
5	E-1008	Nishi	Honrao	15950	15880.0
6	E-1009	Hameed	Singh	15120	NaN

Right Join

Right Join returns all rows from the right table, and any rows with matching keys from the left table.

```
# Display all the information of employees who are receiving bonus

rightjoin=pd.merge(sal_data,bonus_data,how='right')
rightjoin

# Output
```

	Employee_ID	First_Name	Last_Name	Basic_Salary	Bonus
0	E-1001	Mahesh	Joshi	16860.0	16070
1	E-1004	Priya	Jain	12670.0	13490
2	E-1008	Nishi	Honrao	15950.0	15880
3	E-1003	NaN	NaN	NaN	15200
4	E-1006	NaN	NaN	NaN	14200
5	E-1010	NaN	NaN	NaN	15120

To keep all rows from the data set **y** and only those from **x** that match, specify **how='right'**

Inner Join

Inner Join returns only the rows in which the x have matching keys in the y.

```
# Display all the information about employees which are common in
both
the tables
innerjoin=pd.merge(sal_data,bonus_data)
innerjoin

# Output
```

By default **merge** returns inner join.

	Employee_ID	First_Name	Last_Name	Basic_Salary	Bonus
0	E-1001	Mahesh	Joshi	16860	16070
1	E-1004	Priya	Jain	12670	13490
2	E-1008	Nishi	Honrao	15950	15880

Outer Join

Outer Join returns all rows from x and y, join records from x which have matching keys in the y.

```
# Combine sal_data and bonus_data
outerjoin=pd.merge(sal_data,bonus_data,how='outer')
outerjoin
```

```
# Output
```

	Employee_ID	First_Name	Last_Name	Basic_Salary	Bonus
0	E-1001	Mahesh	Joshi	16860.0	16070.0
1	E-1002	Rajesh	Kolte	14960.0	NaN
2	E-1004	Priya	Jain	12670.0	13490.0
3	E-1005	Sneha	Joshi	15660.0	NaN
4	E-1007	Ram	Kanade	15850.0	NaN
5	E-1008	Nishi	Honrao	15950.0	15880.0
6	E-1009	Hameed	Singh	15120.0	NaN
7	E-1003	NaN	NaN	NaN	15200.0
8	E-1006	NaN	NaN	NaN	14200.0
9	E-1010	NaN	NaN	NaN	15120.0

Appending Data - Data Snapshot

basic_salary - 1

Variables

Observations

First_Name	Last_Name	Grade	Location	ba	ms
Alan	Brown	GR1	DELHI	17990	16070
Agatha	Williams	GR2	MUMBAI	13290	6630
Rajesh	Kolte	GR1	MUMBAI	17990	16070
Ameet	Mishra	GR2	DELHI	13290	6630
Neha	Rao	GR1	MUMBAI	17990	16070

basic_salary - 2

Variables

Observations

First_Name	Last_Name	Grade	Location	ba	ms
Sagar	Chavan	GR2	MUMBAI	13290	6630
Aaron	Jones	GR1	MUMBAI	17990	16070
John	Patil	GR2	MUMBAI	13290	6630
Sneha	Joshi	GR1	DELHI	20660	
Gaurav	Singh	GR2	DELHI	13760	13220
Adela	Thomas	GR2	DELHI	13660	6840
Anup	Save	GR2	MUMBAI	11960	7880

Data Description

basic salary – 1 data has 5 rows and 6 columns
basic salary – 2 data has 7 rows and 6 columns

Columns	Description	Type	Measurement	Possible values
First_Name	First Name	character	-	-
Last_Name	Last Name	character	-	-
Grade	Grade	character	GR1, GR2	2
Location	Location	character	DELHI, MUMBAI	2
ba	Basic Allowance	numeric	Rs.	positive values
ms	Management Supplements	numeric	Rs.	positive values

Appending

- Append means adding cases/observations to a dataset.
- `concat()` appends data on one axis while computing the conditions on another.
- `frames=[Salary_1,Salary_2]` is the sequence that is passed as an object to `concat()` function.

Appending Data Sets

```
# Import the data sets and append them using pd.concat()
Salary_1= pd.read_csv('basic_salary - 1.csv')
Salary_2= pd.read_csv('basic_salary - 2.csv')
frames=[Salary_1,Salary_2]
appendsalary=pd.concat(frames)
appendsalary
```

```
# Output
```

	First_Name	Last_Name	Grade	Location	ba	ms
0	Alan	Brown	GR1	DELHI	17990	16070.0
1	Agatha	Williams	GR2	MUMBAI	12390	6630.0
2	Rajesh	Kolte	GR1	MUMBAI	19250	14960.0
3	Ameet	Mishra	GR2	DELHI	14780	9300.0
4	Neha	Rao	GR1	MUMBAI	19235	15200.0
0	Sagar	Chavan	GR2	MUMBAI	13390	6700.0
1	Aaron	Jones	GR1	MUMBAI	23280	13490.0
2	John	Patil	GR2	MUMBAI	13500	10760.0
3	Sneha	Joshi	GR1	DELHI	20660	NaN
4	Gaurav	Singh	GR2	DELHI	13760	13220.0
5	Adela	Thomas	GR2	DELHI	13660	6840.0
6	Anup	Save	GR2	MUMBAI	11960	7880.0

You can see that the original index of the dataframes has been maintained. Use argument **ignore_index=TRUE** for 0 to n-1 index.

Aggregating Data - Data Snapshot

basic_salary data consist salary of each employee with it's Location & Grade.

Variables

Observations

First_Name	Last_Name	Grade	Location	ba	ms
Alan	Brown	GR1	DELHI	17990	16070

Columns	Description	Type	Measurement	Possible values
First_Name	First Name	character	-	-
Last_Name	Last Name	character	-	-
Grade	Grade	character	GR1, GR2	2
Location	Location	character	DELHI, MUMBAI	2
ba	Basic Allowance	numeric	Rs.	positive values
ms	Management Supplements	numeric	Rs.	positive values

Introduction to Aggregation

Aggregating data means splitting data into subsets, computing summary statistics on each subset and displaying the results in a conveniently summarised form.

Suppose a database has millions of rows.

`groupby()` function in pandas carries out the process of taking numerous records and collapsing them into a single summary record.

```
# Import basic_salary data

salary_data=pd.read_csv('basic_salary.csv')
```

Aggregating Single Variable by Single Factor

```
# Calculate sum of variable ‘ms’ by variable ‘Location’
# In this example we are giving one variable and one factor
```

```
A=salary_data.groupby('Location')['ms'].sum()
A
```

```
# Output
```

```
Location
DELHI     45430.0
MUMBAI    75620.0
Name: ms, dtype: float64
```

- To aggregate, we need to create the **groupby** object first. In this case we are grouping **ms** by **Location**.
- **('Location')** tells function to group according to the Location variables. This creates an instance where groupings for all variables are done

`groupby()` by default ignores the missing data values.

ms

Aggregating Multiple Variables by Single Factor

```
# Calculate sum of variables 'ba' and 'ms' by variables 'Location'
# In this example we are giving multiple variables and one factor
```

```
B=salary_data.groupby('Location')['ba','ms'].sum()
B
```

```
# Output
```

Location	ba	ms
DELHI	80850	45430.0
MUMBAI	113005	75620.0

To get the sum for both ba and ms, we're passing these labels to the index via **[ ]**.

Aggregating Multiple Variables by Multiple Factors

```
# Calculate sum of variable 'ms' and 'ba' by variables ‘Location’ and
‘Grade’
# In this example we are giving two variables and two factors

C=salary_data.groupby(['Location', 'Grade'])['ba','ms'].sum()
C

# Output
```

Location	Grade	ba	ms
DELHI	GR1	38650	16070.0
	GR2	42200	29360.0
MUMBAI	GR1	61765	43650.0
	GR2	51240	31970.0

Multiple factors are added as a dictionary, hence need to be contained within a **[ ]**.

Quick Recap

In this session, we learnt different ways of joining two data sets using `merge()` and `concat()` and aggregating data. . Here is the quick recap:

Merging Data	4 types of joins: • left_join • right join • inner join • outer join
Appending Data	`concat()` combines the vector, matrix or data frame by rows
groupby()	• Creates a group as per the object passed, which can include multiple factors and multiple variables • Allows you to run any function ranging from `sum()` to `mean()`, `median()` etc. • Ignores NaN by default