Pandas Aggregate Function

Aggregate function in Pandas performs summary computations on data, often on grouped data. But it can also be used on Series objects.

This can be really useful for tasks such as calculating mean, sum, count, and other statistics for different groups within our data.

Syntax:

Here's the basic syntax of the aggregate function,

df.aggregate(func, axis=0, *args, **kwargs)

Here,

1. `func` - an aggregate function like `sum`, `mean`, etc.
2. `axis` - specifies whether to apply the aggregation operation along rows or columns.
3. `*args` and `**kwargs` - additional arguments that can be passed to the aggregation functions.

Apply Single Aggregate Function

Here's how we can apply a single aggregate function in Pandas.

```
import pandas as pd

data = {
    'Category': ['A', 'A', 'B', 'B', 'A', 'B'],
    'Value': [10, 15, 20, 25, 30, 35]
}

df = pd.DataFrame(data)

# calculate total sum of the Value column
total_sum = df['Value'].aggregate('sum')
print("Total Sum:", total_sum)

# calculate the mean of the Value column
average_value = df['Value'].aggregate('mean')
print("Average Value:", average_value)

# calculate the maximum value in the Value column
max_value = df['Value'].aggregate('max')
print("Maximum Value:", max_value)
```

```
Total Sum: 135
Average Value: 22.5
Maximum Value: 35
```

Here,

- `df['Value'].aggregate('sum')` - calculates the total sum of the `Value` column in the `data` DataFrame
- `df['Value'].aggregate('mean')` - calculates the mean (average) of the `Value` column in the `data` DataFrame
- `df['Value'].aggregate('max')` - computes the maximum value in the `Value` column.

Apply Multiple Aggregate Functions in Pandas

We can also apply multiple aggregation functions to one or more columns using the `aggregate()` function in Pandas. For example,

```
import pandas as pd

data = {
    'Category': ['A', 'A', 'B', 'B', 'A', 'B'],
    'Value': [10, 15, 20, 25, 30, 35]
}

df = pd.DataFrame(data)


# applying multiple aggregation functions to a single column
result = df.groupby('Category')['Value'].agg(['sum', 'mean', 'max',
'min'])
print(result)
```

```
          sum       mean  max  min
Category
A          55  18.333333   30   10
B          80  26.666667   35   20
```

In the above example, we're using the `aggregate()` function to apply multiple aggregation functions (`sum`, `mean`, `max`, and `min`) to the `Value` column after grouping by the `Category` column.

The resulting DataFrame shows the calculated values for each category.

Apply Different Aggregation Functions

In Pandas, we can apply different aggregation functions to different columns using a dictionary with the `aggregate()` function. For example,

```
import pandas as pd

data = {
    'Category': ['A', 'A', 'B', 'B', 'A', 'B'],
    'Value1': [10, 15, 20, 25, 30, 35],
    'Value2': [5, 8, 12, 15, 18, 21]
}

df = pd.DataFrame(data)

agg_funcs = {

 # applying 'sum' to Value1 column
    'Value1': 'sum',

# applying 'mean' and 'max' to Value2 column
    'Value2': ['mean', 'max']
}

result = df.groupby('Category').aggregate(agg_funcs)
print(result)
```

```
          Value1     Value2
             sum       mean max
Category
A             55  10.333333  18
B             80  16.000000  21
```

Here, we're using the `aggregate()` function to apply different aggregation functions to different columns after grouping by the `Category` column.

The resulting DataFrame shows the calculated values for each category and each specified aggregation function.