Pandas groupby

In Pandas, the `groupby` operation lets us group data based on specific columns. This means we can divide a DataFrame into smaller groups based on the values in these columns.

Once grouped, we can then apply functions to each group separately. These functions help summarize or aggregate the data in each group.

Group by a Single Column in Pandas

In Pandas, we use the `groupby()` function to group data by a single column and then calculate the aggregates. For example,

```
import pandas as pd

# create a dictionary containing the data
data = {'Category': ['Electronics', 'Clothing', 'Electronics',
'Clothing'],
        'Sales': [1000, 500, 800, 300]}

# create a DataFrame using the data dictionary
df = pd.DataFrame(data)

# group the DataFrame by the Category column and
# calculate the sum of Sales for each category
grouped = df.groupby('Category')['Sales'].sum()

# print the grouped data
print(grouped)
```

```
Category
Clothing        800
Electronics    1800
Name: Sales, dtype: int64
```

In the above example, `df.groupby('Category')['Sales'].sum()` is used to group by a single column and calculate sum.

This line does the following:

- `df.groupby('Category')` - groups the `df` DataFrame by the unique values in the `Category` column.
- `['Sales']` - specifies that we are interested in the `Sales` column within each group.
- `.sum()` - calculates the sum of the `Sales` values for each group.

Group by a Multiple Column in Pandas

We can also group multiple columns and calculate multiple aggregates in Pandas.

Let's look at an example.

```
import pandas as pd

# create a DataFrame with student data
data = {
    'Gender': ['Male', 'Female', 'Male', 'Female', 'Male'],
    'Grade': ['A', 'B', 'A', 'A', 'B'],
    'Score': [90, 85, 92, 88, 78]
}

df = pd.DataFrame(data)

# define the aggregate functions to be applied to the Score column
agg_functions = {
    # calculate both mean and maximum of the Score column
    'Score': ['mean', 'max']
}

# group the DataFrame by Gender and Grade, then apply the aggregate
functions
grouped = df.groupby(['Gender', 'Grade']).aggregate(agg_functions)

# print the resulting grouped DataFrame
print(grouped)
```

```
              Score
               mean max
Gender Grade
Female A       88.0  88
       B       85.0  85
Male   A       91.0  92
       B       78.0  78
```

Here, In the output, we can see that the data has been grouped by `Gender` and `Grade`, and the `mean` and `maximum` scores for each group are displayed in the resulting DataFrame `grouped`.

Group With Categorical Data

We group with categorical data where we want to analyze data based on specific categories.

Pandas provides powerful tools to work with categorical data efficiently using the `groupby()` function.

Let's look at an example.

```
import pandas as pd

# sample data
data = {'Category': ['A', 'B', 'A', 'B', 'A', 'B'],
        'Sales': [100, 150, 200, 50, 300, 120]}

df = pd.DataFrame(data)

# convert Category column to categorical type
df['Category'] = pd.Categorical(df['Category'])

# group by Category  and calculate the total sales
grouped = df.groupby('Category')['Sales'].sum()

print(grouped)
```

```
Category
A    600
B    320
Name: Sales, dtype: int64

C:\Users\rmnjs\AppData\Local\Temp\ipykernel_22792\592795457.py:13:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  grouped = df.groupby('Category')['Sales'].sum()
```

Here, first the `Category` column is converted to a categorical data type using `pd.Categorical()`.

The data is then grouped by the `Category` column using the `groupby()` function. And the total sales for each category are calculated using the `sum()` aggregation function.