## **Advanced Pandas Functions**

#### **GroupBy**

**GroupBy** allows you to group together rows that share the same value in one or more columns and then perform an operation on each group.

import pandas as pd

# sample dataframe

df = pd.DataFrame({

'A': ['Delhi', 'Mumbai', 'Delhi', 'Mumbai'],

'B': [1, 2, 3, 4],

'C': [2.0, 5., 8., 1.]

})

# grouping by column 'A' and summing up the other columns

grouped = df.groupby('A').sum()

print(grouped)



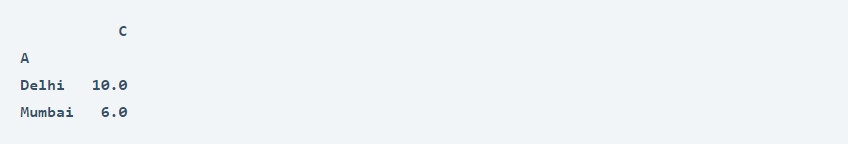
#### **Pivot Tables**

A pivot table is a data summarization tool frequently used in data processing. Pandas pivot\_table can automatically sort, count, and total the data stored in one DataFrame or Series and is particularly useful for quickly summarizing data and highlighting important aspects.

import numpy as np

pivot = pd.pivot\_table(df, values='C', index=['A'], aggfunc=np.sum)

print(pivot)



#### **Multi-Indexing**

Pandas support multi-level indexing, or hierarchical indexing, which allows you to store and manipulate data with an arbitrary number of dimensions in lower dimensional data structures like Series (1D) and DataFrame (2D).

# creating a dataframe with multi-index

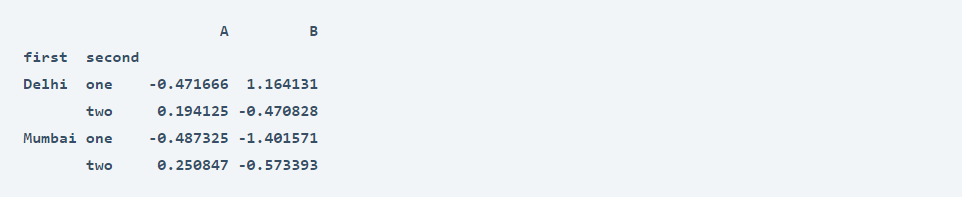
index = pd.MultiIndex.from\_tuples([('Delhi', 'one'), ('Delhi', 'two'),

('Mumbai', 'one'), ('Mumbai', 'two')],

names=['first', 'second'])

df\_multi = pd.DataFrame(np.random.randn(4, 2), index=index, columns=['A', 'B'])

print(df\_multi)



#### **Data Transformation with apply() and map()**

**apply()** allows you to apply a function along an axis of the DataFrame or on a Series. **map()** is a Series method used to substitute each value in a Series with another value.

# using apply()

df['B'] = df['B'].apply(lambda x: x \* 2)

# using map() for a Series

df['A'] = df['A'].map({'Delhi': 'New Delhi', 'Mumbai': 'Bombay'})

#### **Query Function**

The **query()** function in Pandas allows you to filter a DataFrame using a query expression passed as a string.

import pandas as pd

# sample dataframe

df = pd.DataFrame({

'A': range(1, 6),

'B': range(10, 60, 10),

'C': ['Delhi', 'Mumbai', 'Delhi', 'Mumbai', 'Delhi']

})

# using query() to filter rows

filtered\_df = df.query('(A > 2) & (C == "Delhi")')

print(filtered\_df)

