Split-Apply-Combine Technique

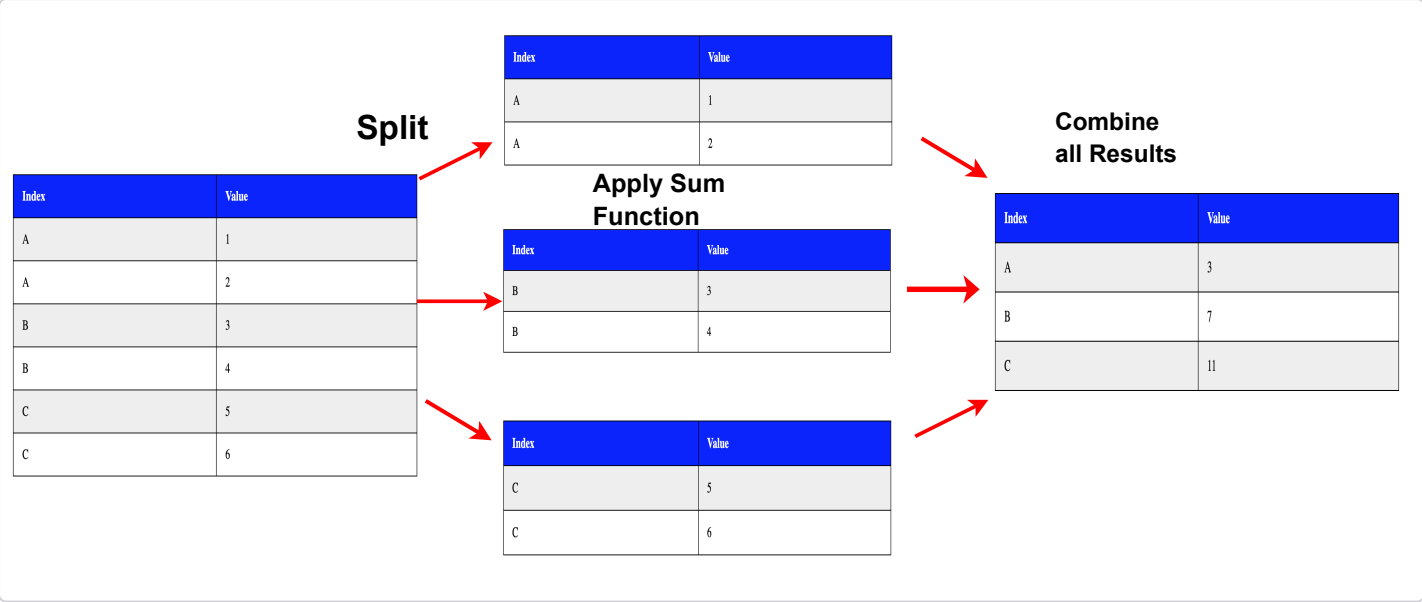
In this lesson, the split-apply-combine can be found of pandas is discussed.

We'll cover the following

- Split-Apply-Combine method
  - Sorting
  - Splitting
  - Applying function
  - Combining

Split-Apply-Combine method

In this technique, we split the data into specific groups, like in the previous lesson. Then certain operations are applied to those groups separately. Finally, all the groups are again combined to form the final required dataset. Let's review the following example.



The initial data set is first split into three groups, `A`, `B`, and `C`. Then, the `sum` operation is applied to every element of each group. Finally, the results are combined at the end, and a dataset with concise required information is formed.

Let's perform this technique on *air quality index* data and see what type of useful

Let's perform this technique on *air quality index* data and see what type of useful information can be extracted.

```
import pandas as pd

df = pd.read_csv('air.csv') # reading data from file

print(df)
```

As can be seen from the output, the file contains the `Date`, `Time`, and the number of different pollutants that are in the air in that time frame. Pollutant data for every hour of each day is mentioned. Our task is to find that time of each day when the concentration of `CO(GT)` and `NO2(GT)` are the highest as these two are the major components that play an important role in air pollution.

The following steps are performed to obtain the required result:

- First, the data needs to be sorted on the amounts of `CO(GT)` and `NO2(GT)` in *descending* order.
- Now, the data is split or grouped on `Date`.
- A function that ranks the data on the basis of the most polluted time of the day is applied.
- Finally, those results are combined where the rank of the specific time of day is highest.

Sorting

The data is sorted using the `sort_values` function explained in this lesson.

```
import pandas as pd

df = pd.read_csv('air.csv')

 # sorting data of certain columns
df = df.sort_values(['NO2(GT)','CO(GT)'], ascending=False)

print(df)
```

It can be clearly seen from the output that the data is sorted in descending order

It can be clearly seen from the output that the data is sorted in descending order. The data is sorted for the `NO2(GT)` column first and then for the `CO(GT)` column. By default, this function sorts data in ascending order, but the `ascending = False` parameter enables it to sort the data in descending order.

Splitting

In this part, the data is split on the `Date` column. The reason for splitting it on `Date` is because we want to rank each time frame of each day. So for this, we need separate groups of each individual day.

```
import pandas as pd

df = pd.read_csv('air.csv')

df = df.sort_values(['NO2(GT)','CO(GT)'], ascending=False)

df = df.groupby('Date')

print(df)
```

The `groupby` clause was used to split the data into different groups of individual days. Now, the data is ready to be ranked. There isn't any relevant output to view, as discussed in this lesson.

Applying function

As the data is already sorted, we only need to assign numbers from *1 through n (length of the day)* to the respective times of each day. This can be done by adding a new column to the data frame and populating it with these values. As the data is recorded for every hour in the day, the ranks go from **1** to **24**.

```
import pandas as pd
import numpy as np

def rank_func(df):
    df['AQI_rank'] = np.arange(0, len(df),1) + 1
    return df

df = pd.read_csv('air.csv')

df = df.sort_values(['NO2(GT)','CO(GT)'], ascending=False)

df = df.groupby('Date')
df = df.apply(rank_func)
```

```
print(df.head(10))
```

From **line 4-6**, the `rank_funk` takes a `DataFrame`, adds a new *column* `AQI_rank` to it, and assigns it numbers from *1 through n*. *n* is the length of each group in the `DataFrame`. The command on **line 5** uses `np.arange()` to assign ranks to each day in a group. As each group is received, the function has the most polluted times of the day at the top. Therefore, relevant ranks are assigned to each day. An additional 1 is added to cater for 0's.

The function then returns the modified `DataFrame`.

On **line 13**, the `apply` keyword is used to apply the `rank_func` function to each group of split data. This line takes each group and feeds it to the `rank_func` function where each row is assigned a *rank*.

It can be seen in the output that the new column is added, and each time of each day now has a rank associated with it based on the concentration of dangerous pollutants.

Combining

Now, all the data is ready and for the last step; we just need to combine the relevant data together to get those times of each day when *AQI* is supposed to be highest.

```
import pandas as pd
import numpy as np

def rank_func(df):
    df['AQI_rank'] = np.arange(0, len(df),1) + 1
    return df

df = pd.read_csv('air.csv')

df = df.sort_values(['NO2(GT)','CO(GT)'], ascending=False)

df = df.groupby('Date')

df = df.apply(rank_func)

df = df[df.AQI_rank == 1]

print(df)
```

The rows with ranks of **1** have the highest *AQI*. So, the entries from the `DataFrame` whose `AQI_rank` value is equal to **1** are selected.

It can be seen in the output that the timestamps of each day with the highest AQI have been filtered out. This is just one example where the *split-apply-technique* is efficiently used to filter out relevant data. For more information on this, refer here.

Next, a challenge awaits to test your newly acquired data manipulation skills.