IOTA ACADEMY

Become Job Ready...

Groupby

A groupby operation involves some combination of splitting the object, applying a function, and combining the results. This can be used to group large amounts of data and compute operations on these groups.

Group by: split-apply-combine

Splitting- the data into groups based on some criteria.

Applying- a function to each group independently.

Combining- the results into a data structure.

In [70]:
```
import pandas as pd
import numpy as np
```

`DataFrame.groupby()`

In [71]:
```
# Create dataframe
data = {'Gender':['m','f','f','m','f','m','m'],'Height':[172,171,169,173,170,175,178]}
df_sample = pd.DataFrame(data)
df_sample
```

Out[71]:

	Gender	Height
0	m	172
1	f	171
2	f	169
3	m	173
4	f	170
5	m	175
6	m	178

Now you can use the .groupby() method to group rows together based off of a column name. For instance let's group based off of Gender. This will create a DataFrameGroupBy object:

In [72]:
```
df_sample.groupby("Gender")
```

Out[72]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001C9A072EE20>
```

GroupBy has conveniently returned a DataFrameGroupBy object. It has split the data into separate groups. However, it won't do anything unless it is being told explicitly to do so.

In [73]:
```
# You can save this object as a new variable:
by_gender = df_sample.groupby("Gender")
```

In [74]:
```
# Applying builtin aggregation fuctions on groupby objects
by_gender.mean()

# It will display the average height of men and women.
```

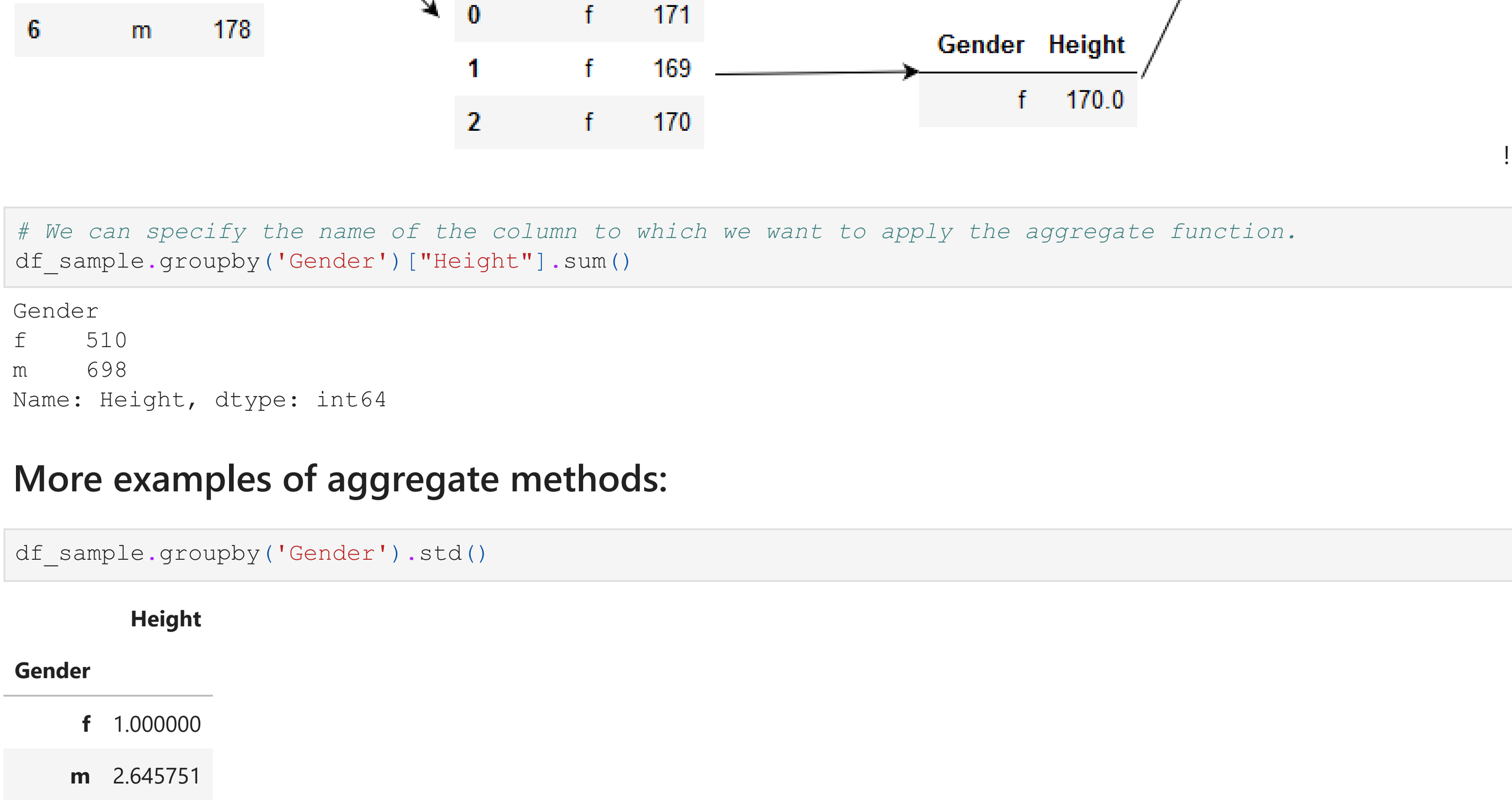
Out[74]:

Gender	Height
f	170.0
m	174.5

In [75]:
```
# Example
group = df_sample.groupby("Gender").mean() # Gender column has been set as an index.

group.loc["f"]
```

Out[75]:
```
Height    170.0
Name: f, dtype: float64
```



In [76]:
```
# We can specify the name of the column to which we want to apply the aggregate function.
df_sample.groupby('Gender')["Height"].sum()
```

Out[76]:
```
Gender
f    510
m    698
Name: Height, dtype: int64
```

More examples of aggregate methods:

In [77]:
```
df_sample.groupby('Gender').std()
```

Out[77]:

Gender	Height
f	1.000000
m	2.645751

In [78]:
```
# It will display the minimum height of men and women.
df_sample.groupby('Gender').min()
```

Out[78]:

Gender	Height
f	169
m	172

In [79]:
```
# It will display the maximum height of men and women.
df_sample.groupby('Gender').max()
```

Out[79]:

Gender	Height
f	171
m	178

In [80]:
```
# It will count the number of men and women in the DataFrame.
df_sample.groupby('Gender').count()
```

Out[80]:

Gender	Height
f	3
m	4

In [81]:
```
# Iterating through groups
by_gender = df_sample.groupby('Gender')

for name,group in by_gender:
    print(name)
    print(group)
```

```
f
  Gender  Height
1      f     171
2      f     169
4      f     170
m
  Gender  Height
0      m     172
3      m     173
5      m     175
6      m     178
```

In [82]:
```
pd.DataFrame(df_sample.groupby(['Gender']).mean()['Height']).reset_index()
```

Out[82]:

	Gender	Height
0	f	170.0
1	m	174.5

We can use the following aggregation:

count() – Number of non-null observations

sum() – Sum of values

mean() – Mean of values

median() – Arithmetic median of values

min() – Minimum

max() – Maximum

mode() – Mode

std() – Standard deviation

var() – Variance

In [83]:
```
# Importing a dataset
df = pd.read_csv("sample_data.csv")
```

In [84]:
```
# Example
df.groupby("Region").sum()
```

Out[84]:

Region	Units	Sales
East	8110.0	167763
North	4359.0	138700
South	2798.0	59315
West	2624.0	61476

We did not tell GroupBy which column we wanted it to apply the aggregation function on, so it applied it to all the relevant columns(numeric) and returned the output.

In [85]:
```
# To use Pandas groupby and aggregate only a single column,we need to index that column.

df.groupby("Region")["Sales"].sum()
```

Out[85]:
```
Region
East     167763
North    138700
South     59315
West      61476
Name: Sales, dtype: int64
```

`get_group()`

In order to select a group, we can select group using GroupBy.get_group()

- **Syntax:** GroupBy.get_group(name, obj=None)

In [86]:
```
# Example
df.groupby("Region").get_group("West")
# It will return a DataFrame of the specified group.
```

Out[86]:

	Date	Region	Type	Units	Sales
4	19-03-2020	West	Women's Clothing	3.0	33
15	26-11-2020	West	Men's Clothing	27.0	864
21	23-06-2020	West	Women's Clothing	18.0	288
24	18-06-2020	West	Men's Clothing	5.0	70
30	13-07-2020	West	Children's Clothing	30.0	450
...	...	...	...	...	...
969	20-07-2020	West	Women's Clothing	25.0	442
970	28-11-2020	West	Women's Clothing	12.0	770
972	21-09-2020	West	Men's Clothing	35.0	437
985	08-02-2020	West	Men's Clothing	32.0	928
991	17-11-2020	West	Men's Clothing	27.0	486

136 rows × 5 columns

`agg()`

agg function in Pandas gives us the flexibility to perform several statistical computations all at once!

- **Syntax:** DataFrame.agg(func=None, axis=0, *args, **kwargs)

In [87]:
```
df2 = df.groupby(["Region"])["Sales"].agg(['max','min','sum'])
df2
```

Out[87]:

Region	max	min	sum
East	1122	36	167763
North	1155	42	138700
South	1122	48	59315
West	1023	33	61476

In [88]:
```
# We can change the name of a column.
df2.rename(columns={"max":"Max_Sales","min":"Min_Sales","sum":"Total Sales"})
```

Out[88]:

Region	Max_Sales	Min_Sales	Total Sales
East	1122	36	167763
North	1155	42	138700
South	1122	48	59315
West	1023	33	61476

In [89]:
```
# Pandas allows us to use different aggregations per column
df.groupby(["Region"]).agg({"Sales":["min","max"],"Units":["sum","mean"]})


# In the .agg() method, we pass in a dictionary. The keys are the column labels and the values represent
# the aggregations we want to apply.
```

Out[89]:

	Sales		Units	
	min	max	sum	mean
Region				
East	36	1122	8110.0	19.732360
North	42	1155	4359.0	19.202643
South	48	1122	2798.0	20.423358
West	33	1023	2624.0	19.294118

Groupby with Multiple Columns

In [90]:
```
# Create a new grouped DataFrame by giving the list of column names into the. groupby() function
df.groupby(["Region","Type"]).sum()
```

Out[90]:

Region	Type	Units	Sales
East	Children's Clothing	2318.0	45849
	Men's Clothing	2420.0	51685
	Women's Clothing	3372.0	70229
North	Children's Clothing	1763.0	37306
	Men's Clothing	0.0	39975
	Women's Clothing	2596.0	61419
South	Children's Clothing	1017.0	18570
	Men's Clothing	725.0	18542
	Women's Clothing	1056.0	22203
West	Children's Clothing	789.0	20182
	Men's Clothing	829.0	19077
	Women's Clothing	1006.0	22217

In [91]:
```
# Changed the value of the as_index parameter to False.
# This way the grouped index would not be output as an index.

pd.DataFrame(df.groupby(["Region","Type"],as_index=False)["Sales"].sum())
```

Out[91]:

	Region	Type	Sales
0	East	Children's Clothing	45849
1	East	Men's Clothing	51685
2	East	Women's Clothing	70229
3	North	Children's Clothing	37306
4	North	Men's Clothing	39975
5	North	Women's Clothing	61419
6	South	Children's Clothing	18570
7	South	Men's Clothing	18542
8	South	Women's Clothing	22203
9	West	Children's Clothing	20182
10	West	Men's Clothing	19077
11	West	Women's Clothing	22217

Great Job!