DATA SOCIETY:

Data wrangling in Python - Data wrangling with Pandas - 4

One should look for what is and not what he thinks should be. (Albert Einstein)

Module completion checklist

Objective	Complete
Summarize data using Pandas	
Filter and sort data using Pandas	

DataFrame description metrics

- Let’s get started by using `.describe()` again to display the summary metrics for the our dataset

```
print(df.describe())
```

```
                 id          age  ...          bmi       stroke
count   5110.000000  5110.000000  ...  4909.000000  5110.000000
mean   36517.829354    43.226614  ...    28.893237     0.048728
std    21161.721625    22.612647  ...     7.854067     0.215320
min       67.000000     0.080000  ...    10.300000     0.000000
25%    17741.250000    25.000000  ...    23.500000     0.000000
50%    36932.000000    45.000000  ...    28.100000     0.000000
75%    54682.000000    61.000000  ...    33.100000     0.000000
max    72940.000000    82.000000  ...    97.600000     1.000000

[8 rows x 7 columns]
```

Methods to summarize and group data in Pandas

- What if we want more detailed summary metrics? Use `groupby()`!
- `groupby()` describes a process involving the following steps:
  - **splitting the data** into groups based on some criteria
  - **applying a function** to each group independently
- We'll be starting with the most straightforward part of `groupby()`, the **split** step

Choosing columns for summarization

- We can choose any column available in the dataset to perform the `groupby()` operation
- To demonstrate, let’s use a column that has a lower number of unique values
- To do this, we first identify the number of unique values in each of the columns of our DataFrame using the function `nunique()`
- We will store the result in the form of a **dictionary** using `to_dict()`

```
col_dict = df.nunique().to_dict()
print(col_dict)
```

```
{'id': 5110, 'gender': 3, 'age': 104, 'hypertension': 2, 'heart_disease': 2, 'ever_married':
2, 'work_type': 5, 'Residence_type': 2, 'avg_glucose_level': 3979, 'bmi': 418,
'smoking_status': 3, 'stroke': 2}
```

Choosing columns for summarization (cont'd)

- We'll now identify and pick a column which has the least number of unique values in it
- Note: If there are multiple columns with the same number of unique levels, the `min` function retrieves the key which occurs first in the order of dictionary values

```
grouping_col = min(col_dict, key=col_dict.get)
grouping_col
```

```
'hypertension'
```

Splitting using groupby()

- A string passed to `groupby()` may refer to either a column or an index level
- We can either group by **column** or by **index**
- Let's group our dataset by the grouping column we just identified

```
grouped = df.groupby(grouping_col)
print(grouped.first())
```

```
                  id  gender   age  ...   bmi   smoking_status stroke
hypertension                        ...
0               9046    Male  67.0  ...  36.6  formerly smoked      1
1               1665  Female  79.0  ...  24.0     never smoked      1

[2 rows x 11 columns]
```

Summarizing using groupby()

- All the summary functions can also be applied to a group
- As a refresher, here are the summary functions:

Function	Description
count	Number of non-null observations
sum	Number of non-null observations
max	Maximum of values
min	Minimum of values
mean	Mean of values
median	Arithmetic median of values
var	Variance of each object
std	Standard deviation of each object

Groupby() and summary functions

- We can now move to the second step of summarizing data, **applying a function** to the group
- Let's inspect the distribution of the resulting DataFrame

```
# Let's count the number of IDs and create a
DataFrame.
df_ID = grouped.count()[['id']]
print(df_ID)
```

```
                  id
hypertension
0               4612
1                498
```

```
# This syntax would do the same, but create a
Series.
print(grouped.count()['id'])
```

Module completion checklist

Objective	Complete
Summarize data using Pandas	✔
Filter and sort data using Pandas	

Sorting data with Pandas

- We can specify whether we want to **sort** our data using `sort_values()` method and specifying some parameters
- Pandas will order rows by the value of the column, either low to high (default) or high to low (`ascending = False`)

```
print(df_ID.sort_values(by = ['id'], ascending = [False]))
```

```
                 id
hypertension
0              4612
1               498
```

Adding a new column

- **Create a new column** by creating a series and adding it to a current DataFrame
- We can even stipulate a specific condition in the column to be added

```
over100_ID = df_ID[['id']] > 100

# Add the new column.
df_ID['over100_ID'] = over100_ID
print(df_ID.head())
```

```
                 id  over100_ID
hypertension
0              4612        True
1               498        True
```

Knowledge check

Link: **https://forms.gle/unNRfRh7ZnVt1T4BA**

Module completion checklist

Objective	Complete
Summarize data using Pandas	✔
Filter and sort data using Pandas	✔

Congratulations on completing this module!

You are now ready to try Tasks 13-16 in the Exercise for this topic