Data wrangling in Python - Data wrangling with Pandas - 3

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

Review quiz

- Before we begin exploring more about Pandas today, let's review some key terms and information through a **matching quiz**
- Link: ***https://quizlet.com/698423398/match***
- If you would like to do the quiz on your cell phone, you can scan the QR code
- **Raise your hand** when you have completed the quiz

DATASOCIETY: © 2023

Module completion checklist

Objective	Complete
Load data into Python using Pandas	
Review and inspect loaded data using Pandas	

DATASOCIETY: © 2023

Dataset

- In order to implement what you learn in this course, we will be using the `healthcare-dataset-stroke-data.csv` dataset
- We will be working with columns from the dataset such as:
  - stroke
  - gender
  - age
  - hypertension
  - heart_disease
  - ever_married

DATASOCIETY: © 2023

Reading data from a file

- Before beginning, you will need to **import your data** into your environment
- You will also need to **set your data directory** to the location where your data is stored
- Your data will likely be stored in a database or as a file
  - A common data format for storing and sharing data is the `csv` (comma separated value)
  - Pandas has a `read_csv` function to import such a file
  - Other common file types include Excel, JSON, HTML, Stata, SAS, and even files from a SQL connection
  - The full list of readable and writable file formats is available ***here***

Read data from csv file

- We are now going to use the function `read_csv` to read in our healthcare dataset

```
df = pd.read_csv(str(data_dir)+'/'+ 'healthcare-dataset-stroke-data.csv')
```

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

```
      id  gender   age  ...   bmi   smoking_status stroke
0   9046    Male  67.0  ...  36.6  formerly smoked      1
1  51676  Female  61.0  ...   NaN     never smoked      1
2  31112    Male  80.0  ...  32.5     never smoked      1
3  60182  Female  49.0  ...  34.4           smokes      1
4   1665  Female  79.0  ...  24.0     never smoked      1

[5 rows x 12 columns]
```


DATASOCIETY: © 2023

Module completion checklist

Objective	Complete
Load data into Python using Pandas	✔
Review and inspect loaded data using Pandas	

DATASOCIETY: © 2023

Inspect data

- Let's start by inspecting the dataset

```
print(type(df))  #<- a Pandas DataFrame!
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
print(len(df))   #<- returns the number of rows
```

```
5110
```

```
# You can also save the shape of the dataframe into 2 variables
# (since the returned is a tuple with 2 values).

nrows, ncols = df.shape
print(nrows)      #<- returns the number of rows, or observations
```

```
5110
```

```
print(ncols)      #<- returns the number of columns, or variables
```

```
12
```

Previewing data - using head method

- We can use the `.head()` command to display the first few rows

```
print(df.head())  #<- pulls the first 5 rows (the default is 5)
```

```
      id  gender   age  ...   bmi   smoking_status stroke
0   9046    Male  67.0  ...  36.6  formerly smoked      1
1  51676  Female  61.0  ...   NaN     never smoked      1
2  31112    Male  80.0  ...  32.5     never smoked      1
3  60182  Female  49.0  ...  34.4           smokes      1
4   1665  Female  79.0  ...  24.0     never smoked      1

[5 rows x 12 columns]
```

- This is a great way to understand the data that we have without having to call the entire dataset

Previewing data - using head method (cont'd)

- We can also specify the number of rows we want to see:

```
print(df.head(3)) #<- pulls the first 3 rows
```

```
      id  gender   age  ...   bmi   smoking_status stroke
0   9046    Male  67.0  ...  36.6  formerly smoked      1
1  51676  Female  61.0  ...   NaN     never smoked      1
2  31112    Male  80.0  ...  32.5     never smoked      1

[3 rows x 12 columns]
```

Previewing data - using sample method

- We can view some random rows in the DataFrame by using the `.sample()` method

```
print(df.sample(n = 3))       #<- 3 random rows
```

```
          id  gender    age  ...   bmi  smoking_status stroke
2928   57609    Male   1.64  ...  20.8             NaN      0
447    48368  Female  65.00  ...  36.8   never smoked      0
149    58978  Female  70.00  ...  26.1   never smoked      1

[3 rows x 12 columns]
```

Previewing data - using sample method (cont'd)

- We can also sample a percentage of the data rather than a number of rows

```
print(df.sample(frac = .02)) #<- a random 2% of the rows
```

```
          id  gender   age  ...   bmi   smoking_status stroke
3305  21989  Female  25.0  ...  48.3              NaN       0
4300  60455    Male  48.0  ...  28.5    never smoked       0
4712  18020    Male  57.0  ...  29.2    never smoked       0
1239  27145  Female  26.0  ...  48.4           smokes       0
249   30669    Male   3.0  ...  18.0              NaN       0
...     ...     ...   ...  ...   ...              ...     ...
961   16402  Female   5.0  ...  19.1              NaN       0
3547  69020  Female  74.0  ...  25.8    never smoked       0
4137  12594  Female  28.0  ...  28.6           smokes       0
2932  48455  Female  37.0  ...  24.1              NaN       0
2056  17492  Female   3.0  ...  24.8              NaN       0

[102 rows x 12 columns]
```

Reviewing the data

Let's get to know our data better using the following `pandas` techniques:

- `.columns`
- `.dtypes`
- `.info()`
- `.describe()`

```
print(df.columns)
```

```
Index(['id', 'gender', 'age', 'hypertension',
'heart_disease', 'ever_married',
       'work_type', 'Residence_type',
'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')
```

```
print(df.dtypes)
```

```
id                   int64
gender              object
age                float64
hypertension         int64
heart_disease        int64
ever_married        object
work_type           object
Residence_type      object
avg_glucose_level  float64
bmi                float64
smoking_status      object
stroke               int64
dtype: object
```

Reviewing the data - info

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 5110 non-null   int64  
 1   gender             5110 non-null   object 
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64  
 4   heart_disease      5110 non-null   int64  
 5   ever_married       5110 non-null   object 
 6   work_type          5110 non-null   object 
 7   Residence_type     5110 non-null   object 
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     3566 non-null   object 
 11  stroke             5110 non-null   int64  
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

Reviewing the data - describe

```
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]
```

Reviewing the data - index

- Let’s check the index of the DataFrame

```
print(df.index)
```

```
RangeIndex(start=0, stop=5110, step=1)
```

- Remember: we can set the index as one of our columns, preferably a unique identifier
- What would make most sense with this dataset?

```
df = df.set_index(['id'])
print(df.index)
```

```
Int64Index([ 9046, 51676, 31112, 60182,  1665, 56669, 53882, 10434, 27419,
             60491,
             ...
             68398, 36901, 45010, 22127, 14180, 18234, 44873, 19723, 37544,
             44679],
            dtype='int64', name='id', length=5110)
```

- Now we can look up rows by the actual IDs

Looking up by ID

- Let's practice looking up values using the index
- We can use `.loc` to look up by specific ID

```
# Look up a specific row by index.
# Rake a random index from the dataframe
np.set_printoptions(suppress=True)
index_random = np.random.permutation(df.index)[:1]
print(index_random)
```

```
[13380]
```

- Now we can use this specific ID to look up the row

```
print(df.loc[index_random])
```

```
      gender   age  hypertension  ...   bmi smoking_status stroke
id                                 ...
13380   Male  14.0             0  ...  23.2            NaN      0

[1 rows x 11 columns]
```

Looking up by ID (cont'd)

- We can also use it to view more than one row

```
print(df.loc[np.random.permutation(df.index)[:10]])
```

```
       gender   age  hypertension  ...   bmi    smoking_status stroke
id                                 ...
41940    Male  57.0             0  ...  31.0  formerly smoked       0
61299  Female  79.0             1  ...  39.0               NaN       0
47701    Male   8.0             0  ...  20.6               NaN       0
37150  Female  34.0             0  ...  48.5  formerly smoked       0
63280  Female  65.0             0  ...  27.8  formerly smoked       0
18412    Male  41.0             0  ...  27.9               NaN       0
40571    Male  29.0             0  ...  28.3     never smoked       0
57210  Female  28.0             0  ...  30.3     never smoked       0
16906    Male  43.0             0  ...  30.0     never smoked       0
69120  Female  31.0             0  ...  39.6     never smoked       0

[10 rows x 11 columns]
```

- **When would something like this be useful in your data?**

DATASOCIETY: © 2023

Looking up with iloc

- We can use `.iloc` to look up a specific observation by row number of the index

```
# Look up a specific row by index.
print(df.iloc[1])
```

```
gender                         Female
age                              61.0
hypertension                        0
heart_disease                       0
ever_married                      Yes
work_type               Self-employed
Residence_type                  Rural
avg_glucose_level              202.21
bmi                               NaN
smoking_status          never smoked
stroke                              1
Name: 51676, dtype: object
```

Resetting the index

- And finally, we can reset our index back to the original index

```
df = df.reset_index()
```

- Now you’re ready to load a dataset into your notebook and summarize data using Pandas

Knowledge check

Link: ***https://forms.gle/fGoDZYFVFJrnyKxq6***

Module completion checklist

Objective	Complete
Load data into Python using Pandas	✔
Review and inspect loaded data using Pandas	✔

Congratulations on completing this module!

You are now ready to try Tasks 9-12 in the Exercise for this topic