Ratchanon Tarawan 65070503464

Lab 3: Data Cleaning and Preparation

Objectives:

- To be more familiar with Pandas libraries
- To gain more hands-on experience in data cleaning and preparation

[1] More Reviews on Pandas

1.0) Discover

- methods to explore and understand your DataFrame

```
import pandas as pd

df = pd.read_csv('nss15.csv')

# see the shape of the dataframe
print(df.shape)
```

```
(334839, 12)
```

```
# seeing the summary of the dataframe
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 334839 entries, 0 to 334838
Data columns (total 12 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   caseNumber     334839 non-null  int64  
 1   treatmentDate  334839 non-null  object 
 2   statWeight     334839 non-null  float64
 3   stratum        334839 non-null  object 
 4   age            334839 non-null  int64  
 5   sex            334837 non-null  object 
 6   race           205014 non-null  object 
 7   diagnosis      334839 non-null  int64  
 8   bodyPart       334839 non-null  int64  
 9   disposition    334839 non-null  int64  
 10  location       334839 non-null  int64  
 11  product        334839 non-null  int64  
dtypes: float64(1), int64(7), object(4)
memory usage: 30.7+ MB
None
```

```python
# seeing the stats of the column in dataframe
print(df.describe())
```

```
         caseNumber     statWeight            age      diagnosis  \
count  3.348390e+05  334839.000000  334839.000000  334839.000000
mean   1.510271e+08      39.343028      31.385451      60.154591
std    1.720330e+06      34.142933      26.105098       6.170699
min    1.501032e+08       4.965500       0.000000      41.000000
25%    1.504405e+08      15.059100      10.000000      57.000000
50%    1.507358e+08      15.776200      23.000000      59.000000
75%    1.510231e+08      74.881300      51.000000      64.000000
max    1.603418e+08      97.923900     107.000000      74.000000

            bodyPart    disposition       location        product
count  334839.000000  334839.000000  334839.000000  334839.000000
mean       64.374192       1.307930       2.485451    2098.900854
std        24.002331       0.977627       3.217617    1332.222670
min         0.000000       1.000000       0.000000     106.000000
25%        35.000000       1.000000       0.000000    1211.000000
50%        75.000000       1.000000       1.000000    1807.000000
75%        82.000000       1.000000       5.000000    3265.000000
max        94.000000       9.000000       9.000000    5555.000000
```

```python
# seeing the first 5 rows of the dataframe
print(df.head())
```

```
   caseNumber treatmentDate  statWeight stratum  age     sex   race  \
0   150733174     7/11/2015     15.7762       V    5    Male    NaN
1   150734723      7/6/2015     83.2157       S   36    Male  White
2   150817487      8/2/2015     74.8813       L   20  Female    NaN
3   150717776     6/26/2015     15.7762       V   61    Male    NaN
4   150721694      7/4/2015     74.8813       L   88  Female  Other

   diagnosis  bodyPart  disposition  location  product
0         57        33            1         9     1267
1         57        34            1         1     1439
2         71        94            1         0     3274
3         71        35            1         0      611
4         62        75            1         0     1893
```

```python
# seeing the last 5 rows of the dataframe
print(df.tail())
```

```
        caseNumber treatmentDate  statWeight stratum  age     sex
race  \
334834   150739278     5/31/2015     15.0591       V    7    Male
NaN
334835   150733393     7/11/2015      5.6748       C    3  Female
Black
334836   150819286     7/24/2015     15.7762       V   38    Male
NaN
```

```
334837   150823002      8/8/2015     97.9239        M   38  Female
White
334838   150723074     6/20/2015     49.2646        M    5  Female
White

        diagnosis  bodyPart  disposition  location  product
334834         59        76            1         1     1864
334835         68        85            1         0     1931
334836         71        79            1         0     3250
334837         59        82            1         1      464
334838         57        34            1         9     3273

# seeing the list of columns in the dataframe
print(df.columns)

Index(['caseNumber', 'treatmentDate', 'statWeight', 'stratum', 'age',
'sex',
       'race', 'diagnosis', 'bodyPart', 'disposition', 'location',
'product'],
      dtype='object')
```

1.2) Selecting variables

- select specific columns from the DataFrame to create a new DataFrame with only those columns

```
df['age']

0          5
1         36
2         20
3         61
4         88
          ..
334834     7
334835     3
334836    38
334837    38
334838     5
Name: age, Length: 334839, dtype: int64

df['age'].head()

0     5
1    36
2    20
3    61
4    88
Name: age, dtype: int64

df[['caseNumber', 'age']]
```

```
        caseNumber  age
0        150733174    5
1        150734723   36
2        150817487   20
3        150717776   61
4        150721694   88
...            ...  ...
334834   150739278    7
334835   150733393    3
334836   150819286   38
334837   150823002   38
334838   150723074    5

[334839 rows x 2 columns]
```

```
# select columns based on the data type
df.select_dtypes(include=['number'])
```

```
        caseNumber  statWeight  age  diagnosis  bodyPart  disposition
\
0        150733174     15.7762    5         57        33            1

1        150734723     83.2157   36         57        34            1

2        150817487     74.8813   20         71        94            1

3        150717776     15.7762   61         71        35            1

4        150721694     74.8813   88         62        75            1

...            ...         ...  ...        ...       ...          ...

334834   150739278     15.0591    7         59        76            1

334835   150733393      5.6748    3         68        85            1

334836   150819286     15.7762   38         71        79            1

334837   150823002     97.9239   38         59        82            1

334838   150723074     49.2646    5         57        34            1


        location  product
0              9     1267
1              1     1439
2              0     3274
3              0      611
4              0     1893
...          ...      ...
334834         1     1864
```

```
334835          0     1931
334836          0     3250
334837          1      464
334838          9     3273

[334839 rows x 8 columns]
```

```
# select row by .loc
df.loc[0]
```

```
caseNumber       150733174
treatmentDate    7/11/2015
statWeight         15.7762
stratum                  V
age                      5
sex                   Male
race                   NaN
diagnosis               57
bodyPart                33
disposition              1
location                 9
product               1267
Name: 0, dtype: object
```

```
# select column by .loc
df.loc[:6,'treatmentDate':'diagnosis']
```

```
  treatmentDate  statWeight stratum  age     sex    race  diagnosis
0     7/11/2015     15.7762       V    5    Male     NaN         57
1      7/6/2015     83.2157       S   36    Male  White         57
2      8/2/2015     74.8813       L   20  Female     NaN         71
3     6/26/2015     15.7762       V   61    Male     NaN         71
4      7/4/2015     74.8813       L   88  Female  Other         62
5      7/2/2015      5.6748       C    1  Female  White         71
6      6/8/2015     15.7762       V   25    Male  Black         51
```

```
df.loc[df['age']>80, ['treatmentDate', 'age']]
```

```
       treatmentDate  age
4           7/4/2015   88
8          7/16/2015   98
39          5/3/2015   88
46         4/15/2015   91
63         1/12/2015   97
...              ...  ...
334701     4/27/2015   86
334784      7/7/2015   82
334785     7/11/2015   86
334815    10/28/2015   85
334819     1/13/2015   85
```

```
[20422 rows x 2 columns]
```

```
# select row by .iloc
df.iloc[0:5]
```

```
   caseNumber treatmentDate  statWeight stratum  age     sex    race  \
0   150733174     7/11/2015     15.7762       V    5    Male    NaN
1   150734723      7/6/2015     83.2157       S   36    Male  White
2   150817487      8/2/2015     74.8813       L   20  Female    NaN
3   150717776     6/26/2015     15.7762       V   61    Male    NaN
4   150721694      7/4/2015     74.8813       L   88  Female  Other

   diagnosis  bodyPart  disposition  location  product
0         57        33            1         9     1267
1         57        34            1         1     1439
2         71        94            1         0     3274
3         71        35            1         0      611
4         62        75            1         0     1893
```

```
# select column by .iloc
df.iloc[:,[0,1,2,3,4]]
```

```
        caseNumber treatmentDate  statWeight stratum  age
0        150733174     7/11/2015     15.7762       V    5
1        150734723      7/6/2015     83.2157       S   36
2        150817487      8/2/2015     74.8813       L   20
3        150717776     6/26/2015     15.7762       V   61
4        150721694      7/4/2015     74.8813       L   88
...            ...           ...         ...     ...  ...
334834   150739278     5/31/2015     15.0591       V    7
334835   150733393     7/11/2015      5.6748       C    3
334836   150819286     7/24/2015     15.7762       V   38
334837   150823002      8/8/2015     97.9239       M   38
334838   150723074     6/20/2015     49.2646       M    5

[334839 rows x 5 columns]
```

1.3) Filtering the data

```
# filter rows based on the condition
df[df['age'] > 50]
```

```
        caseNumber treatmentDate  statWeight stratum  age     sex
race  \
3        150717776     6/26/2015     15.7762       V   61    Male
NaN
4        150721694      7/4/2015     74.8813       L   88  Female
Other
7        150704114     6/14/2015     83.2157       S   53    Male
```

```
White
8        150736558     7/16/2015     83.2157        S   98    Male
Black
16       150901411     8/27/2015     83.2157        S   65  Female
White
...            ...           ...         ...      ...  ...     ...    .
..
334811   150702215     6/27/2015     15.7762        V   51  Female
NaN
334815   151100368    10/28/2015     83.2157        S   85  Female
NaN
334819   150528367     1/13/2015     49.2646        M   85  Female
NaN
334826   150648619     6/17/2015     15.7762        V   52  Female
White
334829   150633526      4/4/2015     49.2646        M   51  Female
NaN

        diagnosis  bodyPart  disposition  location  product
3              71        35            1         0      611
4              62        75            1         0     1893
7              57        30            1         0     5040
8              59        76            1         1     1807
16             59        83            1         1     1817
...           ...       ...          ...       ...      ...
334811         53        83            1         1     1426
334815         57        80            4         1     1807
334819         57        79            5         1      676
334826         64        30            1         1     1842
334829         56        92            1         1     1616

[85235 rows x 12 columns]
```

```
# filter coloum based on column name
df.filter(like='age')
```

```
        age
0         5
1        36
2        20
3        61
4        88
...     ...
334834    7
334835    3
334836   38
334837   38
334838    5

[334839 rows x 1 columns]
```

1.4) Sorting

- Sort the DataFrame by its index based on column

```
# sort the dataframe based on column name and ascending order
df.sort_values(by='statWeight', ascending=False)
```

```
        caseNumber treatmentDate  statWeight stratum  age     sex
race  \
67072    150533084     5/15/2015     97.9239       M   89    Male
NaN
313846   150521217     4/18/2015     97.9239       M   36  Female
NaN
230135   150857760     8/25/2015     97.9239       M   14    Male
White
141323   151039262    10/11/2015     97.9239       M   39  Female
White
230141   150662453      6/5/2015     97.9239       M   11  Female
White
...            ...           ...         ...     ...  ...     ...    .
..
122009   151146792    11/15/2015      4.9655       C    2  Female
White
211090   151253201    12/15/2015      4.9655       C    2    Male
White
317625   160106638    12/25/2015      4.9655       C    1    Male
White
33679    151256307    12/20/2015      4.9655       C    9  Female
Black
229596   160148171     12/4/2015      4.9655       C   16  Female
Other

        diagnosis  bodyPart  disposition  location  product
67072          53        83            1         0     1842
313846         64        79            1         0     5040
230135         64        82            1         8     1807
141323         71        35            1         1     1615
230141         59        88            1         4     3297
...           ...       ...          ...       ...      ...
122009         59        76            1         0     1893
211090         60        88            1         1      661
317625         55        32            1         1      679
33679          57        83            1         0     1842
229596         55        35            1         0     1267

[334839 rows x 12 columns]
```

```
# sort the index of the dataframe
df.sort_index()
```

```
        caseNumber treatmentDate  statWeight stratum  age     sex
race  \
0        150733174     7/11/2015     15.7762       V    5    Male
NaN
1        150734723      7/6/2015     83.2157       S   36    Male
White
2        150817487      8/2/2015     74.8813       L   20  Female
NaN
3        150717776     6/26/2015     15.7762       V   61    Male
NaN
4        150721694      7/4/2015     74.8813       L   88  Female
Other
...            ...           ...         ...     ...  ...     ...    .
..
334834   150739278     5/31/2015     15.0591       V    7    Male
NaN
334835   150733393     7/11/2015      5.6748       C    3  Female
Black
334836   150819286     7/24/2015     15.7762       V   38    Male
NaN
334837   150823002      8/8/2015     97.9239       M   38  Female
White
334838   150723074     6/20/2015     49.2646       M    5  Female
White

        diagnosis  bodyPart  disposition  location  product
0              57        33            1         9     1267
1              57        34            1         1     1439
2              71        94            1         0     3274
3              71        35            1         0      611
4              62        75            1         0     1893
...           ...       ...          ...       ...      ...
334834         59        76            1         1     1864
334835         68        85            1         0     1931
334836         71        79            1         0     3250
334837         59        82            1         1      464
334838         57        34            1         9     3273

[334839 rows x 12 columns]
```

1.5) Add/Remove

- This section shows how to manipulate the DataFrame's structure

```
# Dropping the column
df.drop(columns=['disposition'])
```

```
        caseNumber treatmentDate  statWeight stratum  age     sex
race  \
0        150733174     7/11/2015     15.7762       V    5    Male
```

```
NaN
1        150734723      7/6/2015     83.2157        S   36    Male
White
2        150817487      8/2/2015     74.8813        L   20  Female
NaN
3        150717776     6/26/2015     15.7762        V   61    Male
NaN
4        150721694      7/4/2015     74.8813        L   88  Female
Other
...            ...           ...         ...      ...  ...     ...    .
..
334834   150739278     5/31/2015     15.0591        V    7    Male
NaN
334835   150733393     7/11/2015      5.6748        C    3  Female
Black
334836   150819286     7/24/2015     15.7762        V   38    Male
NaN
334837   150823002      8/8/2015     97.9239        M   38  Female
White
334838   150723074     6/20/2015     49.2646        M    5  Female
White

        diagnosis  bodyPart  location  product
0              57        33         9     1267
1              57        34         1     1439
2              71        94         0     3274
3              71        35         0      611
4              62        75         0     1893
...           ...       ...       ...      ...
334834         59        76         1     1864
334835         68        85         0     1931
334836         71        79         0     3250
334837         59        82         1      464
334838         57        34         9     3273

[334839 rows x 11 columns]
```

```
# Adding column and create into a new column
df.assign(new_column=df['diagnosis'] + df['bodyPart'])
```

```
        caseNumber treatmentDate  statWeight stratum  age     sex
race  \
0        150733174     7/11/2015     15.7762        V    5    Male
NaN
1        150734723      7/6/2015     83.2157        S   36    Male
White
2        150817487      8/2/2015     74.8813        L   20  Female
NaN
3        150717776     6/26/2015     15.7762        V   61    Male
NaN
```

```
4        150721694      7/4/2015     74.8813        L   88  Female
Other
...            ...           ...         ...      ...  ...     ...    .
..
334834   150739278     5/31/2015     15.0591        V    7    Male
NaN
334835   150733393     7/11/2015      5.6748        C    3  Female
Black
334836   150819286     7/24/2015     15.7762        V   38    Male
NaN
334837   150823002      8/8/2015     97.9239        M   38  Female
White
334838   150723074     6/20/2015     49.2646        M    5  Female
White

        diagnosis  bodyPart  disposition  location  product
new_column
0              57        33            1         9     1267
90
1              57        34            1         1     1439
91
2              71        94            1         0     3274
165
3              71        35            1         0      611
106
4              62        75            1         0     1893
137
...           ...       ...          ...       ...      ...          ..
.
334834         59        76            1         1     1864
135
334835         68        85            1         0     1931
153
334836         71        79            1         0     3250
150
334837         59        82            1         1      464
141
334838         57        34            1         9     3273
91

[334839 rows x 13 columns]
```

```
# Removing the column and assigning it to a new variable
df.pop('age')
```

```
0          5
1         36
2         20
3         61
4         88
```

```
          ..
334834     7
334835     3
334836    38
334837    38
334838     5
Name: age, Length: 334839, dtype: int64
```

1.6) Clean missing

- to remove rows with missing values or replace missing values with a specified value

```
# replaceing the missing values with a specified value
df.fillna(value=0)
```

```
        caseNumber treatmentDate  statWeight stratum     sex   race  \
0        150733174     7/11/2015     15.7762       V    Male      0
1        150734723      7/6/2015     83.2157       S    Male  White
2        150817487      8/2/2015     74.8813       L  Female      0
3        150717776     6/26/2015     15.7762       V    Male      0
4        150721694      7/4/2015     74.8813       L  Female  Other
...            ...           ...         ...     ...     ...    ...
334834   150739278     5/31/2015     15.0591       V    Male      0
334835   150733393     7/11/2015      5.6748       C  Female  Black
334836   150819286     7/24/2015     15.7762       V    Male      0
334837   150823002      8/8/2015     97.9239       M  Female  White
334838   150723074     6/20/2015     49.2646       M  Female  White

        diagnosis  bodyPart  disposition  location  product
0              57        33            1         9     1267
1              57        34            1         1     1439
2              71        94            1         0     3274
3              71        35            1         0      611
4              62        75            1         0     1893
...           ...       ...          ...       ...      ...
334834         59        76            1         1     1864
334835         68        85            1         0     1931
334836         71        79            1         0     3250
334837         59        82            1         1      464
334838         57        34            1         9     3273

[334839 rows x 11 columns]
```

```
# Remove the rows with missing values
df.dropna()
```

```
        caseNumber treatmentDate  statWeight stratum     sex   race  \
1        150734723      7/6/2015     83.2157       S    Male  White
4        150721694      7/4/2015     74.8813       L  Female  Other
5        150721815      7/2/2015      5.6748       C  Female  White
6        150713483      6/8/2015     15.7762       V    Male  Black
```

```
7        150704114     6/14/2015     83.2157        S    Male  White
...            ...           ...         ...      ...     ...    ...
334830   150628863      6/8/2015     15.7762        V  Female  White
334831   150607637     5/22/2015      5.6748        C  Female  Black
334835   150733393     7/11/2015      5.6748        C  Female  Black
334837   150823002      8/8/2015     97.9239        M  Female  White
334838   150723074     6/20/2015     49.2646        M  Female  White

        diagnosis  bodyPart  disposition  location  product
1              57        34            1         1     1439
4              62        75            1         0     1893
5              71        76            1         1     1715
6              51        33            4         9     1138
7              57        30            1         0     5040
...           ...       ...          ...       ...      ...
334830         64        79            1         1     1522
334831         59        94            1         0     1616
334835         68        85            1         0     1931
334837         59        82            1         1      464
334838         57        34            1         9     3273

[205014 rows x 11 columns]
```

---

[2] Pandas Practice

Now that the knowledge about Pandas is still fresh, let's practice!

2.1) **[Question]** Use pandas to generate a *series* of 20 consecutive numbers, starting from 120.

```
df = pd.DataFrame()
start_number = 120
consecutive_numbers = pd.Series(range(start_number, start_number +
20), name='number')

df['number'] = consecutive_numbers

print(df)
```

```
    number
0      120
1      121
2      122
3      123
4      124
5      125
6      126
```

```
7      127
8      128
9      129
10     130
11     131
12     132
13     133
14     134
15     135
16     136
17     137
18     138
19     139
```

2.2) **[Question]** Use pandas to generate a *series* of 20 even numbers, starting from 120.

```
start_number = 120
even_numbers = pd.Series(range(start_number, start_number + 40, 2),
name='number')
df['even_number'] = even_numbers
print(df)
```

```
    number  even_number
0      120          120
1      121          122
2      122          124
3      123          126
4      124          128
5      125          130
6      126          132
7      127          134
8      128          136
9      129          138
10     130          140
11     131          142
12     132          144
13     133          146
14     134          148
15     135          150
16     136          152
17     137          154
18     138          156
19     139          158
```

2.3) **[Question]** Use pandas to generate a *series* of 50 numbers in the Fibonacci sequence.

(Hint: The Fibonacci sequence is the series of numbers where each number is the sum of the two preceding numbers. For example, 0, 1, 1, 2, 3, 5, ...)

```python
def generate_fibonacci(n):
    fibonacci_sequence = [0, 1]
    for i in range(2, n):
        next_number = fibonacci_sequence[-1] + fibonacci_sequence[-2]
        fibonacci_sequence.append(next_number)
    return fibonacci_sequence

fibonacci_sequence = generate_fibonacci(50)
fibonacci_series = pd.Series(fibonacci_sequence, name='Fibonacci')

print(fibonacci_series)
```

```
0              0
1              1
2              1
3              2
4              3
5              5
6              8
7             13
8             21
9             34
10            55
11            89
12           144
13           233
14           377
15           610
16           987
17          1597
18          2584
19          4181
20          6765
21         10946
22         17711
23         28657
24         46368
25         75025
26        121393
27        196418
28        317811
29        514229
30        832040
31       1346269
32       2178309
33       3524578
34       5702887
35       9227465
36      14930352
```

```
37      24157817
38      39088169
39      63245986
40     102334155
41     165580141
42     267914296
43     433494437
44     701408733
45    1134903170
46    1836311903
47    2971215073
48    4807526976
49    7778742049
Name: Fibonacci, dtype: int64
```

2.4) **[Question]** Use pandas to generate a *series* of 20 random numbers.

```
import random
def generate_random(n):
  sequence_all = []
  for i in range(0,n):
    i = random.random()
    sequence_all.append(i)
  return sequence_all

random_list = pd.Series(generate_random(20), name ='Random number')
print(random_list)
```

```
0     0.166512
1     0.577022
2     0.428999
3     0.738932
4     0.263911
5     0.236592
6     0.651873
7     0.079695
8     0.603324
9     0.366480
10    0.056460
11    0.618120
12    0.209721
13    0.107587
14    0.948998
15    0.534687
16    0.460256
17    0.234193
18    0.402593
19    0.383902
Name: Random number, dtype: float64
```

2.5) **[Question]** Use pandas to generate a *series* of 20 random numbers, indexed in alphabetical order.

```
alphabetical_index = [chr(i) for i in range(ord('A'), ord('A')+20)]
random_list = pd.Series(generate_random(20), index =
alphabetical_index, name ='Random number With Alphabetical Order')
print(random_list)
```

```
A    0.006644
B    0.315075
C    0.677273
D    0.990840
E    0.318784
F    0.316621
G    0.584781
H    0.746924
I    0.107519
J    0.499295
K    0.443439
L    0.392940
M    0.214590
N    0.474877
O    0.763583
P    0.532495
Q    0.510503
R    0.557308
S    0.113809
T    0.968904
Name: Random number With Alphabetical Order, dtype: float64
```

Next, we're going to use a dataframe which has already been created earlier at the beginning of this notebook. Let's view the first 5 rows (by default).

```
df = pd.read_csv('nss15.csv') # uncomment this line if the dataframe
has been deleted.
df.head()
```

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	\
0	150733174	7/11/2015	15.7762	V	5	Male	NaN	
1	150734723	7/6/2015	83.2157	S	36	Male	White	
2	150817487	8/2/2015	74.8813	L	20	Female	NaN	
3	150717776	6/26/2015	15.7762	V	61	Male	NaN	
4	150721694	7/4/2015	74.8813	L	88	Female	Other	

	diagnosis	bodyPart	disposition	location	product
0	57	33	1	9	1267
1	57	34	1	1	1439
2	71	94	1	0	3274
3	71	35	1	0	611
4	62	75	1	0	1893

2.6) **[Question]** Display *the first 12 rows*

```
df.head(12)
```

```
    caseNumber treatmentDate  statWeight stratum  age     sex
race  \
0    150733174     7/11/2015     15.7762       V    5    Male    NaN
1    150734723      7/6/2015     83.2157       S   36    Male  White
2    150817487      8/2/2015     74.8813       L   20  Female    NaN
3    150717776     6/26/2015     15.7762       V   61    Male    NaN
4    150721694      7/4/2015     74.8813       L   88  Female  Other
5    150721815      7/2/2015      5.6748       C    1  Female  White
6    150713483      6/8/2015     15.7762       V   25    Male  Black
7    150704114     6/14/2015     83.2157       S   53    Male  White
8    150736558     7/16/2015     83.2157       S   98    Male  Black
9    150734928     7/13/2015     74.8813       L   48  Female  Black
10   150734952      7/4/2015     15.7762       V   20    Male  Black
11   150821622     7/20/2015     83.2157       S   20  Female  White


    diagnosis  bodyPart  disposition  location  product
0          57        33            1         9     1267
1          57        34            1         1     1439
2          71        94            1         0     3274
3          71        35            1         0      611
4          62        75            1         0     1893
5          71        76            1         1     1715
6          51        33            4         9     1138
7          57        30            1         0     5040
8          59        76            1         1     1807
9          53        79            1         5     4057
10         59        82            1         1     1894
11         57        36            1         9     1267
```

2.7) **[Question]** Display *the last 7 rows*

```
df.tail(7)
```

```
        caseNumber treatmentDate  statWeight stratum  age     sex
race  \
```

```
334832   150747209     7/24/2015     83.2157        S   14  Female
NaN
334833   150747217     7/24/2015     83.2157        S    2    Male
NaN
334834   150739278     5/31/2015     15.0591        V    7    Male
NaN
334835   150733393     7/11/2015      5.6748        C    3  Female
Black
334836   150819286     7/24/2015     15.7762        V   38    Male
NaN
334837   150823002      8/8/2015     97.9239        M   38  Female
White
334838   150723074     6/20/2015     49.2646        M    5  Female
White

        diagnosis  bodyPart  disposition  location  product
334832         62        75            1         5     1807
334833         62        75            1         1     1301
334834         59        76            1         1     1864
334835         68        85            1         0     1931
334836         71        79            1         0     3250
334837         59        82            1         1      464
334838         57        34            1         9     3273
```

2.8) **[Question]** Display the last 5 rows (by default).

```
df.tail()

        caseNumber treatmentDate  statWeight stratum  age     sex
race  \
334834   150739278     5/31/2015     15.0591        V    7    Male
NaN
334835   150733393     7/11/2015      5.6748        C    3  Female
Black
334836   150819286     7/24/2015     15.7762        V   38    Male
NaN
334837   150823002      8/8/2015     97.9239        M   38  Female
White
334838   150723074     6/20/2015     49.2646        M    5  Female
White

        diagnosis  bodyPart  disposition  location  product
334834         59        76            1         1     1864
334835         68        85            1         0     1931
334836         71        79            1         0     3250
334837         59        82            1         1      464
334838         57        34            1         9     3273
```

2.9) **[Question]** Select the column 'statWeight' and display

```
df.filter(like='statWeight')
```

```
        statWeight
0          15.7762
1          83.2157
2          74.8813
3          15.7762
4          74.8813
...            ...
334834     15.0591
334835      5.6748
334836     15.7762
334837     97.9239
334838     49.2646

[334839 rows x 1 columns]
```

2.10) **[Question]** Select the first 20 rows of the column 'statWeight' and display

```
df.filter(like='statWeight').head(20)
```

```
    statWeight
0      15.7762
1      83.2157
2      74.8813
3      15.7762
4      74.8813
5       5.6748
6      15.7762
7      83.2157
8      83.2157
9      74.8813
10     15.7762
11     83.2157
12     15.7762
13     15.7762
14     37.6645
15     83.2157
16     83.2157
17      5.6748
18     15.7762
19     97.9239
```

2.11) **[Question]** Select the last 50 rows of the column 'statWeight' and find/compute the following values:

- Minimum
- Maximum
- Average

- Standard Deviation

```
data = df['statWeight'].tail(50)

avg_value = data.sum()/50
min_value = data.min()
max_value = data.max()
std_value = data.std()


print(f"Avg: {avg_value}")
print(f"Min: {min_value}")
print(f"Max: {max_value}")
print(f"STD: {std_value}")
```

```
Avg: 45.411078
Min: 5.6748
Max: 97.9239
STD: 34.83805532712222
```

2.12) **[Question]** Select the first 25 rows of *two columns* 'statWeight' and 'age', then find/compute the following values for both columns:

- Minimum
- Maximum
- Average
- Standard Deviation

```
data = df[['statWeight', 'age']].head(25)

avg_value_stats = data['statWeight'].sum()/25
min_value_stats = data['statWeight'].min()
max_value_stats = data['statWeight'].max()
std_value_stats = data['statWeight'].std()

avg_value_age = data['age'].sum()/50
min_value_age = data['age'].min()
max_value_age = data['age'].max()
std_value_age = data['age'].std()

print(f"Avg Weight: {avg_value_stats}")
print(f"Min Weight: {min_value_stats}")
print(f"Max Weight: {max_value_stats}")
print(f"STD Weight: {std_value_stats}")

print(f"Avg Age: {avg_value_age}")
print(f"Min Age: {min_value_age}")
print(f"Max Age: {max_value_age}")
print(f"STD Age: {std_value_age}")
```

```
Avg Weight: 47.033063999999996
Min Weight: 5.6748
Max Weight: 97.9239
STD Weight: 34.547734626417984
Avg Age: 16.92
Min Age: 1
Max Age: 98
STD Age: 26.67501952514124
```

2.13) **[Question]** Select only columns that are of the *type integer*

```
df.select_dtypes(include='int')
```

```
        caseNumber  age  diagnosis  bodyPart  disposition  location
product
0        150733174    5         57        33            1         9
1267
1        150734723   36         57        34            1         1
1439
2        150817487   20         71        94            1         0
3274
3        150717776   61         71        35            1         0
611
4        150721694   88         62        75            1         0
1893
...            ...  ...        ...       ...          ...       ...
...
334834   150739278    7         59        76            1         1
1864
334835   150733393    3         68        85            1         0
1931
334836   150819286   38         71        79            1         0
3250
334837   150823002   38         59        82            1         1
464
334838   150723074    5         57        34            1         9
3273

[334839 rows x 7 columns]
```

2.14) **[Question]** Select only columns that are of the type *string* or *character*

```
df.select_dtypes(include='object')
```

```
       treatmentDate stratum     sex   race
0          7/11/2015       V    Male    NaN
1           7/6/2015       S    Male  White
2           8/2/2015       L  Female    NaN
3          6/26/2015       V    Male    NaN
```

```
4           7/4/2015       L  Female  Other
...              ...     ...     ...    ...
334834     5/31/2015       V    Male    NaN
334835     7/11/2015       C  Female  Black
334836     7/24/2015       V    Male    NaN
334837      8/8/2015       M  Female  White
334838     6/20/2015       M  Female  White

[334839 rows x 4 columns]
```

2.15) **[Question]** Display only unique values in *the column 'race'*

```
df['race'].unique()

array([nan, 'White', 'Other', 'Black', 'Asian', 'American Indian'],
      dtype=object)
```

2.16) **[Question]** Display rows with the following conditions:

- Patients are male
- The age ranges from 35 to 60 years old
- Could be of any race

```
# df2 = df[['sex','age', 'race']]

filtered_df = df[(df['sex'] == 'Male') & (df['age'] >= 35) &
(df['age'] <= 60)]
print(filtered_df)

        caseNumber treatmentDate  statWeight stratum  age   sex   race
\
1        150734723      7/6/2015     83.2157       S   36  Male  White

7        150704114     6/14/2015     83.2157       S   53  Male  White

15       150655986      6/6/2015     83.2157       S   36  Male    NaN

27       150913230      9/4/2015     15.7762       V   39  Male    NaN

32       150908859     8/27/2015     37.6645       L   38  Male  Black

...            ...           ...         ...     ...  ...   ...    ...

334769   150648575     6/16/2015     15.7762       V   47  Male  White

334779   150612283      6/2/2015     15.7762       V   46  Male    NaN

334800   150648581     6/16/2015     15.7762       V   52  Male  White

334805   150511998     4/20/2015     15.0591       V   55  Male  Black
```

```
334836   150819286     7/24/2015     15.7762       V   38  Male    NaN


        diagnosis  bodyPart  disposition  location  product
1              57        34            1         1     1439
7              57        30            1         0     5040
15             59        82            1         0      894
27             71        94            1         0     3274
32             53        36            1         4     5040
...           ...       ...          ...       ...      ...
334769         62        75            4         1     1615
334779         68        85            4         9     5041
334800         64        35            1         1     4074
334805         71        31            6         1     4014
334836         71        79            1         0     3250

[36406 rows x 12 columns]
```

2.17) **[Question]** Based on your output in 2.16), select only the columns below to display.

- caseNumber
- treatmentDate
- race
- diagnosis
- bodyPart
- product

```
df2= filtered_df[['caseNumber', 'treatmentDate','race', 'diagnosis',
'bodyPart', 'product']]
print(df2)

        caseNumber treatmentDate   race  diagnosis  bodyPart  product
1        150734723      7/6/2015  White         57        34     1439
7        150704114     6/14/2015  White         57        30     5040
15       150655986      6/6/2015    NaN         59        82      894
27       150913230      9/4/2015    NaN         71        94     3274
32       150908859     8/27/2015  Black         53        36     5040
...            ...           ...    ...        ...       ...      ...
334769   150648575     6/16/2015  White         62        75     1615
334779   150612283      6/2/2015    NaN         68        85     5041
334800   150648581     6/16/2015  White         64        35     4074
334805   150511998     4/20/2015  Black         71        31     4014
334836   150819286     7/24/2015    NaN         71        79     3250

[36406 rows x 6 columns]
```

2.18) **[Question]** Let's change the condition a bit.

- Patients are female
- The age ranges from 5 to 40 years old

- Could be of any race

```
filtered_df = df[(df['sex'] == 'Female') & (df['age'] >= 5) &
(df['age'] <= 40)]
print(filtered_df)
```

```
        caseNumber treatmentDate  statWeight stratum  age     sex
race  \
2        150817487      8/2/2015     74.8813       L   20  Female
NaN
11       150821622     7/20/2015     83.2157       S   20  Female
White
13       150666343     6/27/2015     15.7762       V   26  Female
White
24       151029050      9/5/2015     49.2646       M   27  Female
NaN
26       151005691     9/29/2015     74.8813       L   27  Female
Black
...            ...           ...         ...     ...  ...     ...    .
..
334827   150640832      6/8/2015     15.7762       V    8  Female
NaN
334830   150628863      6/8/2015     15.7762       V   30  Female
White
334832   150747209     7/24/2015     83.2157       S   14  Female
NaN
334837   150823002      8/8/2015     97.9239       M   38  Female
White
334838   150723074     6/20/2015     49.2646       M    5  Female
White

        diagnosis  bodyPart  disposition  location  product
2              71        94            1         0     3274
11             57        36            1         9     1267
13             62        75            1         1     1807
24             58        76            1         1      611
26             64        93            1         0     1884
...           ...       ...          ...       ...      ...
334827         64        32            1         0     3216
334830         64        79            1         1     1522
334832         62        75            1         5     1807
334837         59        82            1         1      464
334838         57        34            1         9     3273

[71275 rows x 12 columns]
```

2.19) **[Question]** Likewise, based on your output in 2.18), select only the columns below to display.

- caseNumber
- treatmentDate

- race
- diagnosis
- bodyPart
- product

```
df2= filtered_df[['caseNumber', 'treatmentDate','race', 'diagnosis',
'bodyPart', 'product']]
print(df2)
```

```
        caseNumber treatmentDate   race  diagnosis  bodyPart  product
2        150817487      8/2/2015    NaN         71        94     3274
11       150821622     7/20/2015  White         57        36     1267
13       150666343     6/27/2015  White         62        75     1807
24       151029050      9/5/2015    NaN         58        76      611
26       151005691     9/29/2015  Black         64        93     1884
...            ...           ...    ...        ...       ...      ...
334827   150640832      6/8/2015    NaN         64        32     3216
334830   150628863      6/8/2015  White         64        79     1522
334832   150747209     7/24/2015    NaN         62        75     1807
334837   150823002      8/8/2015  White         59        82      464
334838   150723074     6/20/2015  White         57        34     3273

[71275 rows x 6 columns]
```

[3] Data Cleaning and Preparation

.isnull, .dropna, .fillna

3.1) checking

```
# isnull checking
df.isnull().sum()
```

```
caseNumber             0
treatmentDate          0
statWeight             0
stratum                0
age                    0
sex                    2
race              129825
diagnosis              0
bodyPart               0
disposition            0
location               0
product                0
dtype: int64
```

```
# percentage of missing values for the race
df.race.isnull().sum()/df.shape[0]*100
```

```
38.772365226272925
```

```
df.shape[0]
```

```
334839
```

3.2) Drop column

```
# remove column by using
df = df.drop(columns=['race'])

df.head()
```

```
   caseNumber treatmentDate  statWeight stratum  age     sex
diagnosis  \
0   150733174     7/11/2015     15.7762       V    5    Male
57
1   150734723      7/6/2015     83.2157       S   36    Male
57
2   150817487      8/2/2015     74.8813       L   20  Female
71
3   150717776     6/26/2015     15.7762       V   61    Male
71
4   150721694      7/4/2015     74.8813       L   88  Female
62

   bodyPart  disposition  location  product
0        33            1         9     1267
1        34            1         1     1439
2        94            1         0     3274
3        35            1         0      611
4        75            1         0     1893
```

3.3) Data imputation

```
# fillna
df['age'] = df['age'].fillna(df['age'].median())
```

3.4) Drop row that have missing value

```
# remove column by using .dropna()
df = df.dropna()

df.isnull().sum()
```

```
caseNumber       0
treatmentDate    0
```

```
statWeight        0
stratum           0
age               0
sex               0
diagnosis         0
bodyPart          0
disposition       0
location          0
product           0
dtype: int64
```

Datetime

3.5) Working with the datetime format

```
df["treatmentDate"] = pd.to_datetime(df["treatmentDate"],
format="%m/%d/%Y")

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 334837 entries, 0 to 334838
Data columns (total 11 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   caseNumber      334837 non-null  int64
 1   treatmentDate   334837 non-null  datetime64[ns]
 2   statWeight      334837 non-null  float64
 3   stratum         334837 non-null  object
 4   age             334837 non-null  int64
 5   sex             334837 non-null  object
 6   diagnosis       334837 non-null  int64
 7   bodyPart        334837 non-null  int64
 8   disposition     334837 non-null  int64
 9   location        334837 non-null  int64
 10  product         334837 non-null  int64
dtypes: datetime64[ns](1), float64(1), int64(7), object(2)
memory usage: 30.7+ MB
```

```
df['Year'] = df['treatmentDate'].dt.year

df['Month'] = df['treatmentDate'].dt.month

df.head()
```

```
   caseNumber treatmentDate  statWeight stratum  age     sex
diagnosis  \
0   150733174    2015-07-11     15.7762       V    5    Male
57
1   150734723    2015-07-06     83.2157       S   36    Male
```

```
57
2   150817487    2015-08-02     74.8813       L   20  Female
71
3   150717776    2015-06-26     15.7762       V   61    Male
71
4   150721694    2015-07-04     74.8813       L   88  Female
62

   bodyPart  disposition  location  product  Year  Month
0        33            1         9     1267  2015      7
1        34            1         1     1439  2015      7
2        94            1         0     3274  2015      8
3        35            1         0      611  2015      6
4        75            1         0     1893  2015      7
```

[Question] Can you change the format to DD/MM/YYYY? Show your work.

```
df['treatmentDate'].dt.strftime('%d/%m/%Y')

0         11/07/2015
1         06/07/2015
2         02/08/2015
3         26/06/2015
4         04/07/2015
             ...
334834    31/05/2015
334835    11/07/2015
334836    24/07/2015
334837    08/08/2015
334838    20/06/2015
Name: treatmentDate, Length: 334837, dtype: object
```

Combine Dataframe by .merge and .concat

3.6 Merge

```
superstore_order = pd.read_csv('superstore_order.csv')
superstore_people = pd.read_csv('superstore_people.csv')
superstore_return = pd.read_csv('superstore_return.csv')

superstore_order.merge(superstore_return[superstore_return["Returned"]
=="Yes"],
 on="Order ID" ,
 how="inner")\
 [["Customer ID", "Returned"]]\
 .drop_duplicates()

    Customer ID Returned
0      ZD-21925      Yes
```

```
3      TB-21055      Yes
10     JS-15685      Yes
13     LC-16885      Yes
20     BS-11755      Yes
..          ...      ...
688    ED-13885      Yes
689    TS-21205      Yes
696    MF-17665      Yes
702    SH-19975      Yes
705    RB-19435      Yes

[222 rows x 2 columns]
```

[Question] In your opinion, what information that the result above conveys?

Ans: Returning product stats of each customer.

More merging...

```
superstore_order.merge(superstore_return,
 on="Order ID" ,
 how="inner")
```

```
     Row ID        Order ID  Order Date   Ship Date       Ship Mode  \
0        19  CA-2014-143336  27/08/2014  01/09/2014    Second Class
1        20  CA-2014-143336  27/08/2014  01/09/2014    Second Class
2        21  CA-2014-143336  27/08/2014  01/09/2014    Second Class
3        56  CA-2016-111682  17/06/2016  18/06/2016     First Class
4        57  CA-2016-111682  17/06/2016  18/06/2016     First Class
..      ...             ...         ...         ...             ...
702    8870  CA-2017-101805  01/12/2017  06/12/2017  Standard Class
703    8871  CA-2017-101805  01/12/2017  06/12/2017  Standard Class
704    8872  CA-2017-101805  01/12/2017  06/12/2017  Standard Class
705    8873  US-2014-105137  10/10/2014  10/10/2014        Same Day
706    8874  US-2014-105137  10/10/2014  10/10/2014        Same Day

    Customer ID       Customer Name    Segment        Country
City  \
0      ZD-21925  Zuschuss Donatelli   Consumer  United States  San
Francisco
1      ZD-21925  Zuschuss Donatelli   Consumer  United States  San
Francisco
2      ZD-21925  Zuschuss Donatelli   Consumer  United States  San
Francisco
3      TB-21055     Ted Butterfield   Consumer  United States
Troy
4      TB-21055     Ted Butterfield   Consumer  United States
Troy
..          ...                 ...         ...             ...
...
```

```
702    SH-19975       Sally Hughsby  Corporate  United States
Seattle
703    SH-19975       Sally Hughsby  Corporate  United States
Seattle
704    SH-19975       Sally Hughsby  Corporate  United States
Seattle
705    RB-19435     Richard Bierner   Consumer  United States
Columbus
706    RB-19435     Richard Bierner   Consumer  United States
Columbus

     ... Region       Product ID         Category Sub-Category  \
0    ...   West  OFF-AR-10003056  Office Supplies          Art
1    ...   West  TEC-PH-10001949       Technology       Phones
2    ...   West  OFF-BI-10002215  Office Supplies      Binders
3    ...   East  OFF-ST-10000604  Office Supplies      Storage
4    ...   East  OFF-PA-10001569  Office Supplies        Paper
..   ...    ...              ...              ...          ...
702  ...   West  OFF-BI-10002003  Office Supplies      Binders
703  ...   West  FUR-FU-10000023        Furniture  Furnishings
704  ...   West  OFF-ST-10002756  Office Supplies      Storage
705  ...   East  TEC-MA-10002694       Technology     Machines
706  ...   East  OFF-BI-10002429  Office Supplies      Binders

                                            Product Name    Sales
Quantity  \
0                                          Newell 341    8.560
2
1                               Cisco SPA 501G IP Phone  213.480
3
2             Wilson Jones Hanging View Binder White 1   22.720
4
3                      Home/Office Personal File Carts  208.560
6
4                                           Xerox 232   32.400
5
..                                                ...      ...       .
..
702       Ibico Presentation Index for Binding Systems   15.920
5
703                          Eldon Wave Desk Accessories   70.680
12
704  Tennsco Stur-D-Stor Boltless Shelving 5 Shelve...  541.240
4
705  Hewlett-Packard Deskjet F4180 All-in-One Color...  101.994
2
706               Premier Elliptical Ring Binder Black   18.264
2

     Discount   Profit  Returned
```

```
0         0.0   2.4824       Yes
1         0.2  16.0110       Yes
2         0.2   7.3840       Yes
3         0.0  52.1400       Yes
4         0.0  15.5520       Yes
..        ...      ...       ...
702       0.2   5.3730       Yes
703       0.0  31.0992       Yes
704       0.0   5.4124       Yes
705       0.7 -71.3958       Yes
706       0.7 -13.3936       Yes

[707 rows x 22 columns]
```

3.7) Concatenate

```
pd.concat([superstore_order, superstore_people], axis=1, join='inner')
```

```
   Row ID        Order ID  Order Date   Ship Date       Ship Mode
Customer ID  \
0       1  CA-2016-152156  08/11/2016  11/11/2016    Second Class
CG-12520
1       2  CA-2016-152156  08/11/2016  11/11/2016    Second Class
CG-12520
2       3  CA-2016-138688  12/06/2016  16/06/2016    Second Class
DV-13045
3       4  US-2015-108966  11/10/2015  18/10/2015  Standard Class
SO-20335

     Customer Name    Segment        Country             City  ...  \
0      Claire Gute   Consumer  United States        Henderson  ...
1      Claire Gute   Consumer  United States        Henderson  ...
2  Darrin Van Huff  Corporate  United States      Los Angeles  ...
3    Sean ODonnell   Consumer  United States  Fort Lauderdale  ...

        Product ID         Category Sub-Category  \
0  FUR-BO-10001798        Furniture    Bookcases
1  FUR-CH-10000454        Furniture       Chairs
2  OFF-LA-10000240  Office Supplies       Labels
3  FUR-TA-10000577        Furniture       Tables

                                        Product Name     Sales
Quantity  \
0                  Bush Somerset Collection Bookcase  261.9600
2
1  Hon Deluxe Fabric Upholstered Stacking Chairs ...  731.9400
3
2  Self-Adhesive Address Labels for Typewriters b...   14.6200
2
3      Bretford CR4500 Series Slim Rectangular Table  957.5775
```

```
5
```

```
  Discount    Profit             Person   Region
0     0.00   41.9136      Anna Andreadi     West
1     0.00  219.5820        Chuck Magee     East
2     0.00    6.8714     Kelly Williams  Central
3     0.45 -383.0310  Cassandra Brandow    South

[4 rows x 23 columns]
```

Groupby

```
superstore_order.groupby(['Segment','Ship Mode'])
[['Sales','Quantity','Discount','Profit']].sum()
```

```
                                  Sales  Quantity  Discount
Profit
Segment     Ship Mode

Consumer    First Class     138594.9328      2455    110.29
18953.7264
            Same Day         53660.6340      1001     43.85
8555.7193
            Second Class    203605.6822      3489    127.29
24701.9148
            Standard Class  627061.3262     10430    443.05
68864.9892
Corporate   First Class      97720.1209      1670     73.07
12660.2526
            Same Day         41716.5550       366     14.50
1120.9222
            Second Class    130759.9288      2027     71.47
15582.1762
            Standard Class  359359.2109      6203    262.82
49832.6780
Home Office First Class      76743.8674       924     39.82
11829.8821
            Same Day         20968.5170       343     12.50
3909.3442
            Second Class     77175.1080      1148     37.80
12785.8953
            Standard Class  218325.9795      3595    142.14
27298.5786
```

[Question] Briefly describe an information that the result above conveys?

Ans: Basically, it is stats whether product stock or amount of sale product in each segment / group. Also, showing the profit performance too.

```
superstore_order["Profit Ratio"] =
superstore_order["Profit"]/superstore_order["Sales"]

superstore_order.groupby(["Category", "Sub-
Category"]).agg(mean_profit_ratio = ("Profit Ratio", "mean"))
```

```
                              mean_profit_ratio
Category        Sub-Category
Furniture       Bookcases              -0.127756
                Chairs                  0.045028
                Furnishings             0.140782
                Tables                 -0.147916
Office Supplies Appliances             -0.145513
                Art                     0.251678
                Binders                -0.191641
                Envelopes               0.421913
                Fasteners               0.301157
                Labels                  0.429984
                Paper                   0.425586
                Storage                 0.092382
                Supplies                0.104970
Technology      Accessories             0.219012
                Copiers                 0.317826
                Machines               -0.059535
                Phones                  0.118926
```

[Question] Briefly describe an information that the result above conveys?

Ans: Profit ratio of each product categories that will be shown profit performance of each product.

Pivot and Melt

Pivot

```
superstore_order.pivot_table(index="State", columns="Ship Mode",
values="Order ID", aggfunc="count").fillna(0).head(10)
```

```
Ship Mode             First Class  Same Day  Second Class  Standard
Class
State

Alabama                       9.0       1.0          18.0
30.0
Arizona                      42.0      15.0          22.0
123.0
Arkansas                     10.0       2.0           8.0
35.0
California                  302.0     106.0         346.0
1000.0
```

```
Colorado                       43.0        5.0          32.0
95.0
Connecticut                    19.0        8.0          11.0
39.0
Delaware                       16.0        2.0          13.0
55.0
District of Columbia            0.0        0.0           3.0
7.0
Florida                        47.0       25.0          57.0
210.0
Georgia                        19.0       15.0          31.0
108.0
```

```
pivot_table_result = superstore_order.pivot_table(index="State",
columns="Ship Mode", values="Order ID", aggfunc="count").fillna(0)
print(pivot_table_result)
```

```
Ship Mode              First Class  Same Day  Second Class  Standard
Class
State

Alabama                        9.0        1.0          18.0
30.0
Arizona                        42.0       15.0          22.0
123.0
Arkansas                       10.0        2.0           8.0
35.0
California                    302.0      106.0         346.0
1000.0
Colorado                       43.0        5.0          32.0
95.0
Connecticut                    19.0        8.0          11.0
39.0
Delaware                       16.0        2.0          13.0
55.0
District of Columbia            0.0        0.0           3.0
7.0
Florida                        47.0       25.0          57.0
210.0
Georgia                        19.0       15.0          31.0
108.0
Idaho                           3.0        0.0           2.0
13.0
Illinois                       58.0       24.0          96.0
249.0
Indiana                        13.0        3.0          30.0
79.0
Iowa                            1.0        1.0           4.0
17.0
Kansas                          6.0        1.0           2.0
```

15.0
Kentucky 12.0 5.0 49.0
62.0
Louisiana 7.0 2.0 14.0
15.0
Maine 0.0 0.0 0.0
5.0
Maryland 18.0 7.0 12.0
63.0
Massachusetts 14.0 4.0 35.0
71.0
Michigan 20.0 16.0 43.0
151.0
Minnesota 9.0 4.0 13.0
59.0
Mississippi 3.0 4.0 7.0
36.0
Missouri 7.0 2.0 20.0
24.0
Montana 1.0 1.0 0.0
13.0
Nebraska 6.0 3.0 6.0
20.0
Nevada 4.0 1.0 12.0
17.0
New Hampshire 2.0 0.0 10.0
13.0
New Jersey 5.0 1.0 20.0
87.0
New Mexico 1.0 0.0 9.0
22.0
New York 155.0 57.0 183.0
606.0
North Carolina 36.0 14.0 40.0
139.0
North Dakota 0.0 0.0 5.0
2.0
Ohio 66.0 47.0 84.0
199.0
Oklahoma 5.0 6.0 7.0
44.0
Oregon 20.0 0.0 15.0
81.0
Pennsylvania 103.0 9.0 78.0
341.0
Rhode Island 16.0 0.0 21.0
16.0
South Carolina 3.0 5.0 18.0
16.0

```
South Dakota                    2.0        0.0           0.0
9.0
Tennessee                      21.0        2.0          24.0
118.0
Texas                         125.0       37.0         161.0
537.0
Utah                            4.0        2.0          19.0
28.0
Vermont                         0.0        0.0           1.0
2.0
Virginia                       39.0        4.0          33.0
115.0
Washington                     56.0       34.0          97.0
265.0
West Virginia                   0.0        0.0           0.0
3.0
Wisconsin                      12.0        3.0          10.0
66.0
Wyoming                         0.0        0.0           0.0
1.0
```

Melt

```
melted_result = pd.melt(pivot_table_result.reset_index(),
id_vars=["State"], var_name="Ship Mode", value_name="Order Count")
print(melted_result)
```

```
             State       Ship Mode  Order Count
0          Alabama     First Class          9.0
1          Arizona     First Class         42.0
2         Arkansas     First Class         10.0
3       California     First Class        302.0
4         Colorado     First Class         43.0
..             ...             ...          ...
191       Virginia  Standard Class        115.0
192     Washington  Standard Class        265.0
193  West Virginia  Standard Class          3.0
194      Wisconsin  Standard Class         66.0
195        Wyoming  Standard Class          1.0

[196 rows x 3 columns]
```

---

[4] Some more questions!

Let's practice more using the superstore dataset :D

4.1) **[Question]** From the superstore_order, display the ascending order considering values in the 'Profit' column to group the 'Category'.

```
 superstore_order.groupby('Category')[['Profit']].mean()
```

```
                    Profit
Category
Furniture         8.967320
Office Supplies  19.743848
Technology       81.347862
```

4.2) **[Question]** Create a new column that calculates the total price (sale*quantity) before discount then group by 'product id' and 'category', then show the mean of the total price

```
superstore_order['TotalPriceBeforeDiscount'] =
superstore_order['Sales'] * superstore_order['Quantity']

superstore_order.groupby(['Product ID', 'Category'])
['TotalPriceBeforeDiscount'].mean()
```

```
Product ID       Category
FUR-BO-10000112  Furniture     7426.566000
FUR-BO-10000330  Furniture     1258.192000
FUR-BO-10000362  Furniture     1726.898000
FUR-BO-10000468  Furniture      426.532400
FUR-BO-10000711  Furniture     3194.100000
                                   ...
TEC-PH-10004912  Technology     747.320000
TEC-PH-10004922  Technology     673.249500
TEC-PH-10004924  Technology      57.149333
TEC-PH-10004959  Technology     412.009000
TEC-PH-10004977  Technology    2441.475429
Name: TotalPriceBeforeDiscount, Length: 1846, dtype: float64
```