3-data-cleaning-and-preparation

February 1, 2024

1 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

2 [1] More Reviews on Pandas

1.0) Discover * methods to explore and understand your DataFrame

[1]:
```python
import pandas as pd

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

```
C:\Users\woosh\AppData\Local\Temp\ipykernel_18212\1974226225.py:1:
DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of
pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better
interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

[2]:
```python
# see the shape of the dataframe
print(df.shape)
```

```
(334839, 12)
```

[3]:
```python
# 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
```

[4]:
```
# 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
```

[5]:
```
# 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
```

[6]:

```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
```

[7]:

```python
# 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

[8]:

```python
df['age']
```

[8]:

```
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
```

```
[9]: df['age'].head()
```

```
[9]: 0     5
     1    36
     2    20
     3    61
     4    88
     Name: age, dtype: int64
```

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

```
[10]:         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]
```

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

```
[11]:         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]
```

[12]:

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

[12]:

```
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
```

[13]:

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

[13]:

```
  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
```

[14]:

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

[14]:

```
       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]
```

[15]:
```
# select row by .iloc
df.iloc[0:5]
```

[15]:
```
   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
```

[16]:
```
# select column by .iloc
df.iloc[:,[0,1,2,3,4]]
```

[16]:
```
        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

[17]:

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

[17]:

```
        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]
```

[18]:

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

[18]:

```
        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

[19]:
```
# sort the dataframe based on column name and ascending order
df.sort_values(by='statWeight', ascending=False)
```

[19]:
```
        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]
```

[20]:
```
# sort the index of the dataframe
df.sort_index()
```

[20]:
```
        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

[21]:
```
# Dropping the column
df.drop(columns=['disposition'])
```

[21]:
```
        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]
```

[22]:

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

[22]:

```
        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]
```

[23]:

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

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

[24]:

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

[24]:

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

```
1        150734723      7/6/2015     83.2157       S   36    Male  White
2        150817487      8/2/2015     74.8813       L   20  Female      0
3        150717776     6/26/2015     15.7762       V   61    Male      0
4        150721694      7/4/2015     74.8813       L   88  Female  Other
…            …           …         …     …  …     …    …
334834   150739278     5/31/2015     15.0591       V    7    Male      0
334835   150733393     7/11/2015      5.6748       C    3  Female  Black
334836   150819286     7/24/2015     15.7762       V   38    Male      0
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]
```

[25]:
```
# Remove the rows with missing values
df.dropna()
```

[25]:
```
        caseNumber treatmentDate  statWeight stratum  age     sex   race  \
1        150734723      7/6/2015     83.2157       S   36    Male  White
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
…            …           …         …     …  …     …    …
334830   150628863      6/8/2015     15.7762       V   30  Female  White
334831   150607637     5/22/2015      5.6748       C    1  Female  Black
334835   150733393     7/11/2015      5.6748       C    3  Female  Black
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
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 12 columns]
```

---

3 [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.

[26]:
```
# write your code here
pd.Series(data = range(120,140), dtype="int")
```

[26]:
```
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
dtype: int32
```

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

[27]:
```
# write your code here
pd.Series(data = range(120,160,2), dtype="int")
```

```
[27]: 0      120
      1      122
      2      124
      3      126
      4      128
      5      130
      6      132
      7      134
      8      136
      9      138
      10     140
      11     142
      12     144
      13     146
      14     148
      15     150
      16     152
      17     154
      18     156
      19     158
      dtype: int32
```

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, …)

```
[28]: def fibo(n):
          a, b = 0, 1
          for i in range(0, n):
              a, b = b, a + b
          return a
```

```
[29]: # write your code here
      pd.Series(data = [fibo(n) for n in range(50)])
```

```
[29]: 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
dtype: int64
```

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

```
[32]: # write your code here
      import numpy as np
      pd.Series(data = np.random.randn(20,))
```

[32]:

```
0     1.152249
1    -0.591620
2    -1.791902
3     0.155772
4    -0.614648
5    -0.742754
6     0.654407
7    -0.110480
8    -1.785995
9    -0.180858
10    0.442804
11    0.146879
12   -1.320452
13    0.563261
14    0.862737
15    0.933455
16    0.767730
17    1.461301
18    0.214919
19    0.383272
dtype: float64
```

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

[33]:

```
# write your code here
pd.Series(data=np.random.randn(20,), index=[chr(i) for i in range(65,85)])
```

[33]:

```
A   -1.281986
B   -0.578905
C    1.022104
D   -1.551667
E   -0.370524
F    1.916036
G    0.543641
H   -0.304513
I    0.789417
J    0.498850
K   -1.075449
L    0.554306
M    0.377492
N   -2.329619
O    0.166598
P    2.139807
Q   -2.496479
R   -0.528367
S    1.819391
T    0.628939
```

```
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).

[34]:
```
# df = pd.read_csv('nss15.csv') # uncomment this line if the dataframe has been␣
 ↪deleted.
df.head()
```

[34]:
```
   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*

[35]:
```
# write your code here
df.head(12)
```

[35]:
```
    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*

[36]:
```
# write your code here
df.tail(7)
```

[36]:
```
        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).

[37]:
```
# write your code here
df.tail()
```

[37]:
```
        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

[38]:
```
# write your code here
df['statWeight']
```

[38]:
```
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
Name: statWeight, Length: 334839, dtype: float64
```

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

[40]:
```
# write your code here
df['statWeight'].iloc[:20]
df['statWeight'].head(20)
```

[40]:
```
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
Name: statWeight, dtype: float64
```

2.11) [**Question**] Select the last 50 rows of the column ‘statWeight’ and find/compute the following values: - Minimum - Maximum - Average - Standard Deviation

[42]:

```
# write your code here
last50 = df['statWeight'].iloc[-50:]
minn = last50.min()
maxx = last50.max()
mean = last50.mean()
std = last50.std()
last50,minn,maxx,mean,std
```

[42]:

```
(334789     5.6748
 334790    83.2157
 334791    74.8813
 334792    74.8813
 334793    97.9239
 334794    15.0591
 334795    15.7762
 334796    74.8813
 334797    15.0591
 334798    49.2646
 334799    15.0591
 334800    15.7762
 334801    49.2646
 334802    74.8813
 334803    74.8813
 334804    74.8813
 334805    15.0591
 334806    97.9239
 334807    15.0591
 334808    15.7762
 334809    15.0591
 334810    97.9239
 334811    15.7762
 334812    85.7374
 334813    97.9239
 334814    85.7374
 334815    83.2157
 334816    15.7762
 334817    15.7762
 334818    97.9239
 334819    49.2646
 334820    15.0591
 334821    15.0591
 334822     5.6748
 334823     5.6748
 334824     5.6748
 334825    80.8381
 334826    15.7762
 334827    15.7762
 334828    74.8813
 334829    49.2646
 334830    15.7762
 334831     5.6748
 334832    83.2157
 334833    83.2157
 334834    15.0591
 334835     5.6748
 334836    15.7762
 334837    97.9239
 334838    49.2646
 Name: statWeight, dtype: float64,
 5.6748,
 97.9239,
 45.411078,
 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

[44]:

```python
# write your code here
last50 = df[['statWeight','age']].iloc[-50:]
minn = last50.min()
maxn = last50.max()
mean = last50.mean()
std = last50.std()
print(last50)
minn,maxx,mean,std
```

```
        statWeight  age
334789      5.6748   11
334790     83.2157    9
334791     74.8813   61
334792     74.8813   38
334793     97.9239   13
334794     15.0591    2
334795     15.7762   15
334796     74.8813   46
334797     15.0591   48
334798     49.2646   33
334799     15.0591   31
334800     15.7762   52
334801     49.2646    2
334802     74.8813   29
334803     74.8813   23
```

```
334804     74.8813   17
334805     15.0591   55
334806     97.9239   18
334807     15.0591    2
334808     15.7762    2
334809     15.0591    4
334810     97.9239   33
334811     15.7762   51
334812     85.7374    3
334813     97.9239   21
334814     85.7374   12
334815     83.2157   85
334816     15.7762    4
334817     15.7762    5
334818     97.9239    5
334819     49.2646   85
334820     15.0591   12
334821     15.0591   27
334822      5.6748   15
334823      5.6748    1
334824      5.6748    1
334825     80.8381    5
334826     15.7762   52
334827     15.7762    8
334828     74.8813    2
334829     49.2646   51
334830     15.7762   30
334831      5.6748    1
334832     83.2157   14
334833     83.2157    2
334834     15.0591    7
334835      5.6748    3
334836     15.7762   38
334837     97.9239   38
334838     49.2646    5
```

```
[44]: (statWeight    5.6748
       age           1.0000
       dtype: float64,
       97.9239,
       statWeight    45.411078
       age           22.540000
       dtype: float64,
       statWeight    34.838055
       age           22.104769
       dtype: float64)
```

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

[45]:

```
# write your code here
df.select_dtypes(include=['int'])
```

[45]:

```
        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*

[46]:

```
# write your code here
df.select_dtypes(include=['object'])
```

[46]:

```
       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'*

[47]:

```
# write your code here
df['race'].unique()
```

[47]:

```
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

```
[48]: # write your code here
      oldmale = df[(df['sex'] == 'Male') & (df['age'].between(35, 60)) & (~df['race'].
       ↪isna())]
      oldmale
```

```
[48]:         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
      32       150908859     8/27/2015     37.6645       L   38  Male  Black
      36       151029422     10/6/2015     97.9239       M   37  Male  White
      44       150407764     3/27/2015     74.8813       L   36  Male  White
      …              …            …          …       …   …     …     …
      334744   151013354     9/30/2015     83.2157       S   49  Male  White
      334751   151042699    10/16/2015     16.5650       V   51  Male  Other
      334769   150648575     6/16/2015     15.7762       V   47  Male  White
      334800   150648581     6/16/2015     15.7762       V   52  Male  White
      334805   150511998     4/20/2015     15.0591       V   55  Male  Black

              diagnosis  bodyPart  disposition  location  product
      1              57        34            1         1     1439
      7              57        30            1         0     5040
      32             53        36            1         4     5040
      36             64        35            1         0     1267
      44             62        75            1         0     1842
      …            …       …           …          …        …
      334744         57        82            1         5     1807
      334751         59        92            1         0      832
      334769         62        75            4         1     1615
      334800         64        35            1         1     4074
      334805         71        31            6         1     4014

      [20683 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

```
[49]: # write your code here
      oldmale[['caseNumber','treatmentDate','race','diagnosis','bodyPart','product']]
```

```
[49]:         caseNumber treatmentDate   race  diagnosis  bodyPart  product
      1        150734723      7/6/2015  White         57        34     1439
      7        150704114     6/14/2015  White         57        30     5040
      32       150908859     8/27/2015  Black         53        36     5040
      36       151029422     10/6/2015  White         64        35     1267
      44       150407764     3/27/2015  White         62        75     1842
      …              …            …    …        …         …       …
      334744   151013354     9/30/2015  White         57        82     1807
```

```
334751   151042699    10/16/2015  Other          59        92       832
334769   150648575     6/16/2015  White          62        75      1615
334800   150648581     6/16/2015  White          64        35      4074
334805   150511998     4/20/2015  Black          71        31      4014

[20683 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

[50]:

```
# write your code here
younggirl = df[(df['sex'] == 'Female') & (df['age'].between(5, 40)) &␣
 ↪(~df['race'].isna())]
younggirl
```

[50]:

```
        caseNumber treatmentDate  statWeight stratum  age     sex  race  \
11       150821622     7/20/2015     83.2157       S   20  Female  White
13       150666343     6/27/2015     15.7762       V   26  Female  White
26       151005691     9/29/2015     74.8813       L   27  Female  Black
43       150413327     3/30/2015     37.6645       L   32  Female  White
53       150362991     3/11/2015     74.8813       L   13  Female  Black
…            …            …          …     …  …      …    …
334817   150639257      6/8/2015     15.7762       V    5  Female  White
334818   150630000     5/20/2015     97.9239       M    5  Female  Black
334830   150628863      6/8/2015     15.7762       V   30  Female  White
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
11             57        36            1         9     1267
13             62        75            1         1     1807
26             64        93            1         0     1884
43             71        79            1         0     3216
53             55        30            1         9     1205
…           …       …          …       …      …
334817         57        76            2         0     3286
334818         59        94            1         1     1871
334830         64        79            1         1     1522
334837         59        82            1         1      464
334838         57        34            1         9     3273

[45160 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

[51]:

```
# write your code here
younggirl[['caseNumber','treatmentDate','race','diagnosis','bodyPart','product']]
```

[51]:

```
        caseNumber treatmentDate   race  diagnosis  bodyPart  product
11       150821622     7/20/2015  White         57        36     1267
13       150666343     6/27/2015  White         62        75     1807
26       151005691     9/29/2015  Black         64        93     1884
43       150413327     3/30/2015  White         71        79     3216
53       150362991     3/11/2015  Black         55        30     1205
…              …           …    …        …       …       …
334817   150639257      6/8/2015  White         57        76     3286
334818   150630000     5/20/2015  Black         59        94     1871
334830   150628863      6/8/2015  White         64        79     1522
334837   150823002      8/8/2015  White         59        82      464
334838   150723074     6/20/2015  White         57        34     3273

[45160 rows x 6 columns]
```

4 [3] Data Cleaning and Preparation

4.0.1 .isnull, .dropna, .fillna

3.1) checking

[52]:

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

[52]:

```
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
```

[53]:

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

[53]:

```
38.772365226272925
```

[54]:

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

[54]:

```
334839
```

3.2) Drop column

[55]:

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

[56]:

```
df.head()
```

[56]:

```
   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

[57]:

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

3.4) Drop row that have missing value

[58]:

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

[59]:

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

[59]:

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

4.0.2 Datetime

3.5) Working with the datetime format

```
[60]: df["treatmentDate"] = pd.to_datetime(df["treatmentDate"], format="%m/%d/%Y")␣
        ↪#doesn't actually work along the format.
      df
```

```
[60]:        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
      …             …           …          …      …   …      …        …
      334834  150739278    2015-05-31     15.0591       V    7    Male         59
      334835  150733393    2015-07-11      5.6748       C    3  Female         68
      334836  150819286    2015-07-24     15.7762       V   38    Male         71
      334837  150823002    2015-08-08     97.9239       M   38  Female         59
      334838  150723074    2015-06-20     49.2646       M    5  Female         57

              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
      …          …            …         …        …
      334834        76            1         1     1864
      334835        85            1         0     1931
      334836        79            1         0     3250
      334837        82            1         1      464
      334838        34            1         9     3273

      [334837 rows x 11 columns]
```

```
[61]: df.info()
```

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

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

```
[63]: df['Month'] = df['treatmentDate'].dt.month
```

```
[64]: df.head()
```

```
[64]:    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.

```
[65]: # write your code here
      df['treatmentDate'] = df['treatmentDate'].dt.strftime('%d/%m/%Y')
      df
```

```
[65]:         caseNumber treatmentDate  statWeight stratum  age     sex  diagnosis  \
      0        150733174    11/07/2015     15.7762       V    5    Male         57
      1        150734723    06/07/2015     83.2157       S   36    Male         57
      2        150817487    02/08/2015     74.8813       L   20  Female         71
      3        150717776    26/06/2015     15.7762       V   61    Male         71
      4        150721694    04/07/2015     74.8813       L   88  Female         62
      …              …             …          …      …   …      …          …
      334834   150739278    31/05/2015     15.0591       V    7    Male         59
      334835   150733393    11/07/2015      5.6748       C    3  Female         68
      334836   150819286    24/07/2015     15.7762       V   38    Male         71
      334837   150823002    08/08/2015     97.9239       M   38  Female         59
      334838   150723074    20/06/2015     49.2646       M    5  Female         57

              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
…          …          …       …       …    …     …
334834        76            1         1     1864  2015        5
334835        85            1         0     1931  2015        7
334836        79            1         0     3250  2015        7
334837        82            1         1      464  2015        8
334838        34            1         9     3273  2015        6

[334837 rows x 13 columns]
```

4.0.3 Combine Dataframe by .merge and .concat

3.6 Merge

```
[66]: superstore_order = pd.read_csv('Superstore\superstore_order.csv')
      superstore_people = pd.read_csv('Superstore\superstore_people.csv')
      superstore_return = pd.read_csv('Superstore\superstore_return.csv')
```

```
<>:1: SyntaxWarning: invalid escape sequence '\s'
<>:2: SyntaxWarning: invalid escape sequence '\s'
<>:3: SyntaxWarning: invalid escape sequence '\s'
<>:1: SyntaxWarning: invalid escape sequence '\s'
<>:2: SyntaxWarning: invalid escape sequence '\s'
<>:3: SyntaxWarning: invalid escape sequence '\s'
C:\Users\woosh\AppData\Local\Temp\ipykernel_18212\1019190268.py:1:
SyntaxWarning: invalid escape sequence '\s'
  superstore_order = pd.read_csv('Superstore\superstore_order.csv')
C:\Users\woosh\AppData\Local\Temp\ipykernel_18212\1019190268.py:2:
SyntaxWarning: invalid escape sequence '\s'
  superstore_people = pd.read_csv('Superstore\superstore_people.csv')
C:\Users\woosh\AppData\Local\Temp\ipykernel_18212\1019190268.py:3:
SyntaxWarning: invalid escape sequence '\s'
  superstore_return = pd.read_csv('Superstore\superstore_return.csv')
```

```
[67]: superstore_order
```

```
[67]:       Row ID         Order ID  Order Date   Ship Date       Ship Mode  \
      0          1  CA-2016-152156  08/11/2016  11/11/2016    Second Class
      1          2  CA-2016-152156  08/11/2016  11/11/2016    Second Class
      2          3  CA-2016-138688  12/06/2016  16/06/2016    Second Class
      3          4  US-2015-108966  11/10/2015  18/10/2015  Standard Class
      4          5  US-2015-108966  11/10/2015  18/10/2015  Standard Class
      …        …             …         …         …             …
      8875    8876  US-2016-141264  13/08/2016  19/08/2016  Standard Class
      8876    8877  US-2016-141264  13/08/2016  19/08/2016  Standard Class
```

```
8877    8878  CA-2017-126928  17/09/2017  23/09/2017  Standard Class
8878    8879  CA-2017-126928  17/09/2017  23/09/2017  Standard Class
8879    8880  US-2015-107944  23/03/2015  25/03/2015     First Class

     Customer ID    Customer Name    Segment        Country             City  \
0       CG-12520      Claire Gute   Consumer  United States        Henderson
1       CG-12520      Claire Gute   Consumer  United States        Henderson
2       DV-13045  Darrin Van Huff  Corporate  United States      Los Angeles
3       SO-20335    Sean ODonnell   Consumer  United States  Fort Lauderdale
4       SO-20335    Sean ODonnell   Consumer  United States  Fort Lauderdale
…          …              …        …            …              …
8875    CT-11995     Carol Triggs   Consumer  United States           Irving
8876    CT-11995     Carol Triggs   Consumer  United States           Irving
8877    GZ-14470    Gary Zandusky   Consumer  United States       Morristown
8878    GZ-14470    Gary Zandusky   Consumer  United States       Morristown
8879    AM-10360   Alice McCarthy  Corporate  United States      Los Angeles

      … Postal Code   Region       Product ID         Category Sub-Category  \
0     …       42420    South  FUR-BO-10001798        Furniture    Bookcases
1     …       42420    South  FUR-CH-10000454        Furniture       Chairs
2     …       90036     West  OFF-LA-10000240  Office Supplies       Labels
3     …       33311    South  FUR-TA-10000577        Furniture       Tables
4     …       33311    South  OFF-ST-10000760  Office Supplies      Storage
…   …         …      …              …              …          …
8875  …       75061  Central  OFF-SU-10003505  Office Supplies     Supplies
8876  …       75061  Central  OFF-AP-10002534  Office Supplies   Appliances
8877  …        7960     East  TEC-MA-10004626       Technology     Machines
8878  …        7960     East  OFF-ST-10000615  Office Supplies      Storage
8879  …       90008     West  OFF-PA-10000659  Office Supplies        Paper

                                           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
4                         Eldon Fold N Roll Cart System   22.3680         2
…                                                  …       …       …
8875                     Premier Electric Letter Opener  185.3760         2
8876  3.6 Cubic Foot Counter Height Office Refrigerator   58.9240         1
8877  Lexmark 20R1285 X6650 Wireless All-in-One Printer  480.0000         4
8878  SimpliFile Personal File Black Granite 15w x 6…   34.0500         3
8879  TOPS Carbonless Receipt Book Four 2-3/4 x 7-1/…  192.7200        11

      Discount    Profit
0         0.00   41.9136
1         0.00  219.5820
2         0.00    6.8714
```

```
3         0.45 -383.0310
4         0.20    2.5164
…        …        …
8875      0.20  -34.7580
8876      0.80 -153.2024
8877      0.00  225.6000
8878      0.00    9.5340
8879      0.00   92.5056

[8880 rows x 21 columns]
```

[68]:

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

[68]:

```
    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: Superstore is checking for honest customer that bought their products and have returned the exact products. hahahahahah. The table show customers that returned their product.

More merging…

[69]:

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

[69]:

```
     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

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

```
[70]:    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]
```

4.0.4 Groupby

[71]:

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

[71]:

```
                               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: This table is a summary of the sales in all segments. In each segment is divided into difference ship mode. As can be seen, the majority sales quantity is from consumer segment. The profit significantly come from standard class ship mode from every segment. Same day ship mode is not popular like the others.

[72]:

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

[73]:

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

[73]:

```
                              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: The table show the mean of profit ration to see which product can make a profit. As can be seen roughly, office supplies is the most profitable category. Labels is the best one that can make a profit comparing to it's sales.

4.0.5 Pivot and Melt

Pivot

[74]:
```
superstore_order.pivot_table(index="State", columns="Ship Mode", values="Order␣
  ↪ID", aggfunc="count").fillna(0).head(10)
```

[74]:
```
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
```

[75]:
```
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

[76]:

```
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]
```

---

5 [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 the average values of the 'Profit' column to group the 'Category'.

[77]:

```
#enter your code here
print(superstore_order.groupby('Category')['Profit'].mean().
 ↪sort_values(ascending=True))
superstore_order.groupby('Category').agg(Average_profit = ("Profit","mean"))
```

```
Category
Furniture          8.967320
Office Supplies   19.743848
Technology        81.347862
Name: Profit, dtype: float64
```

[77]:

```
                 Average_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

[78]:

```
#enter your code here
superstore_order["total_price"] =␣
 ↪superstore_order["Sales"]*superstore_order["Quantity"]
superstore_order.groupby(['Product ID','Category']).agg(meam_total_price =␣
 ↪("total_price", "mean"))
```

[78]:

```
                                 meam_total_price
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

[1846 rows x 1 columns]
```