# Lab 3: Data Preparation

## CPE232 Data Models

**Owner : 66070501043 - Phoorin Chinphuad**

---

# [1] Reviews on Pandas

1.1) Discover

- methods to explore and understand your DataFrame

In [159…

```
import pandas as pd

df = pd.read_csv('./sources/lab/nss15.csv')
```

In [160…

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

```
(334839, 12)
```

In [161…

```
# 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
```

In [162…

```
# 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  
```

In [163…

```
# 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  
```

In [164…

```
# 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  
```

In [165…

```
# 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

In [166…

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

Out[166…

```
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
```

In [167…

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

Out[167…

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

In [168…

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

Out[168…

| | 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 × 2 columns

In [169…

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

Out[169…

| | caseNumber | statWeight | age | diagnosis | bodyPart | disposition | location | product |
|---|---|---|---|---|---|---|---|---|
| **0** | 150733174 | 15.7762 | 5 | 57 | 33 | 1 | 9 | 1267 |
| **1** | 150734723 | 83.2157 | 36 | 57 | 34 | 1 | 1 | 1439 |
| **2** | 150817487 | 74.8813 | 20 | 71 | 94 | 1 | 0 | 3274 |
| **3** | 150717776 | 15.7762 | 61 | 71 | 35 | 1 | 0 | 611 |
| **4** | 150721694 | 74.8813 | 88 | 62 | 75 | 1 | 0 | 1893 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **334834** | 150739278 | 15.0591 | 7 | 59 | 76 | 1 | 1 | 1864 |
| **334835** | 150733393 | 5.6748 | 3 | 68 | 85 | 1 | 0 | 1931 |
| **334836** | 150819286 | 15.7762 | 38 | 71 | 79 | 1 | 0 | 3250 |
| **334837** | 150823002 | 97.9239 | 38 | 59 | 82 | 1 | 1 | 464 |
| **334838** | 150723074 | 49.2646 | 5 | 57 | 34 | 1 | 9 | 3273 |

334839 rows × 8 columns

In [170…

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

Out[170…

```
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
```

In [171…

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

Out[171…

| | 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 |

In [172…

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

Out[172…

| | 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 × 2 columns

In [173…

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

Out[173…

| | caseNumber | treatmentDate | statWeight | stratum | age | sex | race | diagnosis | bodyPart | dis |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 150733174 | 7/11/2015 | 15.7762 | V | 5 | Male | NaN | 57 | 33 | |
| **1** | 150734723 | 7/6/2015 | 83.2157 | S | 36 | Male | White | 57 | 34 | |
| **2** | 150817487 | 8/2/2015 | 74.8813 | L | 20 | Female | NaN | 71 | 94 | |
| **3** | 150717776 | 6/26/2015 | 15.7762 | V | 61 | Male | NaN | 71 | 35 | |
| **4** | 150721694 | 7/4/2015 | 74.8813 | L | 88 | Female | Other | 62 | 75 | |

In [174…

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

Out[174…

| | 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 × 5 columns

1.3) Filtering the data

In [175…

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

Out[175…

| | caseNumber | treatmentDate | statWeight | stratum | age | sex | race | diagnosis | bodyPar |
|---|---|---|---|---|---|---|---|---|---|
| **3** | 150717776 | 6/26/2015 | 15.7762 | V | 61 | Male | NaN | 71 | 3 |
| **4** | 150721694 | 7/4/2015 | 74.8813 | L | 88 | Female | Other | 62 | 7 |
| **7** | 150704114 | 6/14/2015 | 83.2157 | S | 53 | Male | White | 57 | 3 |
| **8** | 150736558 | 7/16/2015 | 83.2157 | S | 98 | Male | Black | 59 | 7 |
| **16** | 150901411 | 8/27/2015 | 83.2157 | S | 65 | Female | White | 59 | 8 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | . |
| **334811** | 150702215 | 6/27/2015 | 15.7762 | V | 51 | Female | NaN | 53 | 8 |
| **334815** | 151100368 | 10/28/2015 | 83.2157 | S | 85 | Female | NaN | 57 | 8 |
| **334819** | 150528367 | 1/13/2015 | 49.2646 | M | 85 | Female | NaN | 57 | 7 |
| **334826** | 150648619 | 6/17/2015 | 15.7762 | V | 52 | Female | White | 64 | 3 |
| **334829** | 150633526 | 4/4/2015 | 49.2646 | M | 51 | Female | NaN | 56 | 9 |

85235 rows × 12 columns

In [176…

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

Out[176…

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

334839 rows × 1 columns

1.4) Sorting

- Sort the DataFrame by its index based on column

In [177…

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

Out[177…

| | caseNumber | treatmentDate | statWeight | stratum | age | sex | race | diagnosis | bodyPar |
|---|---|---|---|---|---|---|---|---|---|
| **59985** | 151114128 | 11/1/2015 | 97.9239 | M | 17 | Male | Black | 64 | 3 |
| **239709** | 150809041 | 8/2/2015 | 97.9239 | M | 34 | Female | White | 57 | 8 |
| **239711** | 151018719 | 10/5/2015 | 97.9239 | M | 72 | Female | Black | 59 | 3 |
| **239659** | 151138100 | 10/25/2015 | 97.9239 | M | 24 | Male | NaN | 64 | 3 |
| **239696** | 150903084 | 8/25/2015 | 97.9239 | M | 75 | Male | NaN | 53 | 7 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | . |
| **123586** | 151201944 | 11/28/2015 | 4.9655 | C | 11 | Female | White | 56 | 3 |
| **123589** | 151226169 | 12/8/2015 | 4.9655 | C | 5 | Male | Black | 64 | 3 |
| **138289** | 151132404 | 11/8/2015 | 4.9655 | C | 6 | Female | White | 57 | 7 |
| **138295** | 151136771 | 11/14/2015 | 4.9655 | C | 5 | Female | Black | 59 | 7 |
| **138235** | 151150939 | 11/19/2015 | 4.9655 | C | 14 | Female | White | 64 | 9 |

334839 rows × 12 columns

In [178…

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

Out[178…

| | caseNumber | treatmentDate | statWeight | stratum | age | sex | race | diagnosis | bodyPar |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 150733174 | 7/11/2015 | 15.7762 | V | 5 | Male | NaN | 57 | 3 |
| **1** | 150734723 | 7/6/2015 | 83.2157 | S | 36 | Male | White | 57 | 3 |
| **2** | 150817487 | 8/2/2015 | 74.8813 | L | 20 | Female | NaN | 71 | 9 |
| **3** | 150717776 | 6/26/2015 | 15.7762 | V | 61 | Male | NaN | 71 | 3 |
| **4** | 150721694 | 7/4/2015 | 74.8813 | L | 88 | Female | Other | 62 | 7 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | . |
| **334834** | 150739278 | 5/31/2015 | 15.0591 | V | 7 | Male | NaN | 59 | 7 |
| **334835** | 150733393 | 7/11/2015 | 5.6748 | C | 3 | Female | Black | 68 | 8 |
| **334836** | 150819286 | 7/24/2015 | 15.7762 | V | 38 | Male | NaN | 71 | 7 |
| **334837** | 150823002 | 8/8/2015 | 97.9239 | M | 38 | Female | White | 59 | 8 |
| **334838** | 150723074 | 6/20/2015 | 49.2646 | M | 5 | Female | White | 57 | 3 |

334839 rows × 12 columns

1.5) Add/Remove

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

In [179…

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

Out[179…

| | caseNumber | treatmentDate | statWeight | stratum | age | sex | race | diagnosis | bodyPar |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 150733174 | 7/11/2015 | 15.7762 | V | 5 | Male | NaN | 57 | 3 |
| **1** | 150734723 | 7/6/2015 | 83.2157 | S | 36 | Male | White | 57 | 3 |
| **2** | 150817487 | 8/2/2015 | 74.8813 | L | 20 | Female | NaN | 71 | 9 |
| **3** | 150717776 | 6/26/2015 | 15.7762 | V | 61 | Male | NaN | 71 | 3 |
| **4** | 150721694 | 7/4/2015 | 74.8813 | L | 88 | Female | Other | 62 | 7 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | . |
| **334834** | 150739278 | 5/31/2015 | 15.0591 | V | 7 | Male | NaN | 59 | 7 |
| **334835** | 150733393 | 7/11/2015 | 5.6748 | C | 3 | Female | Black | 68 | 8 |
| **334836** | 150819286 | 7/24/2015 | 15.7762 | V | 38 | Male | NaN | 71 | 7 |
| **334837** | 150823002 | 8/8/2015 | 97.9239 | M | 38 | Female | White | 59 | 8 |
| **334838** | 150723074 | 6/20/2015 | 49.2646 | M | 5 | Female | White | 57 | 3 |

334839 rows × 11 columns

In [180…

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

Out[180…

| | caseNumber | treatmentDate | statWeight | stratum | age | sex | race | diagnosis | bodyPar |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 150733174 | 7/11/2015 | 15.7762 | V | 5 | Male | NaN | 57 | 3 |
| **1** | 150734723 | 7/6/2015 | 83.2157 | S | 36 | Male | White | 57 | 3 |
| **2** | 150817487 | 8/2/2015 | 74.8813 | L | 20 | Female | NaN | 71 | 9 |
| **3** | 150717776 | 6/26/2015 | 15.7762 | V | 61 | Male | NaN | 71 | 3 |
| **4** | 150721694 | 7/4/2015 | 74.8813 | L | 88 | Female | Other | 62 | 7 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | . |
| **334834** | 150739278 | 5/31/2015 | 15.0591 | V | 7 | Male | NaN | 59 | 7 |
| **334835** | 150733393 | 7/11/2015 | 5.6748 | C | 3 | Female | Black | 68 | 8 |
| **334836** | 150819286 | 7/24/2015 | 15.7762 | V | 38 | Male | NaN | 71 | 7 |
| **334837** | 150823002 | 8/8/2015 | 97.9239 | M | 38 | Female | White | 59 | 8 |
| **334838** | 150723074 | 6/20/2015 | 49.2646 | M | 5 | Female | White | 57 | 3 |

334839 rows × 13 columns

In [181…

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

1.6) Clean missing

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

In [182…

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

Out[182…

| | caseNumber | treatmentDate | statWeight | stratum | sex | race | diagnosis | bodyPart | dis |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 150733174 | 7/11/2015 | 15.7762 | V | Male | 0 | 57 | 33 | |
| **1** | 150734723 | 7/6/2015 | 83.2157 | S | Male | White | 57 | 34 | |
| **2** | 150817487 | 8/2/2015 | 74.8813 | L | Female | 0 | 71 | 94 | |
| **3** | 150717776 | 6/26/2015 | 15.7762 | V | Male | 0 | 71 | 35 | |
| **4** | 150721694 | 7/4/2015 | 74.8813 | L | Female | Other | 62 | 75 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **334834** | 150739278 | 5/31/2015 | 15.0591 | V | Male | 0 | 59 | 76 | |
| **334835** | 150733393 | 7/11/2015 | 5.6748 | C | Female | Black | 68 | 85 | |
| **334836** | 150819286 | 7/24/2015 | 15.7762 | V | Male | 0 | 71 | 79 | |
| **334837** | 150823002 | 8/8/2015 | 97.9239 | M | Female | White | 59 | 82 | |
| **334838** | 150723074 | 6/20/2015 | 49.2646 | M | Female | White | 57 | 34 | |

334839 rows × 11 columns

In [183…

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

Out[183…

| | caseNumber | treatmentDate | statWeight | stratum | sex | race | diagnosis | bodyPart | dis |
|---|---|---|---|---|---|---|---|---|---|
| **1** | 150734723 | 7/6/2015 | 83.2157 | S | Male | White | 57 | 34 | |
| **4** | 150721694 | 7/4/2015 | 74.8813 | L | Female | Other | 62 | 75 | |
| **5** | 150721815 | 7/2/2015 | 5.6748 | C | Female | White | 71 | 76 | |
| **6** | 150713483 | 6/8/2015 | 15.7762 | V | Male | Black | 51 | 33 | |
| **7** | 150704114 | 6/14/2015 | 83.2157 | S | Male | White | 57 | 30 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **334830** | 150628863 | 6/8/2015 | 15.7762 | V | Female | White | 64 | 79 | |
| **334831** | 150607637 | 5/22/2015 | 5.6748 | C | Female | Black | 59 | 94 | |
| **334835** | 150733393 | 7/11/2015 | 5.6748 | C | Female | Black | 68 | 85 | |
| **334837** | 150823002 | 8/8/2015 | 97.9239 | M | Female | White | 59 | 82 | |
| **334838** | 150723074 | 6/20/2015 | 49.2646 | M | Female | White | 57 | 34 | |

205014 rows × 11 columns

# [2] Data Cleaning and Preparation

## .isnull, .dropna, .fillna

2.1) checking

In [184…

```
df.columns
```

Out[184…

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

In [185…

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

Out[185…

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

In [186…

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

Out[186…

```
np.float64(38.772365226272925)
```

In [187…

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

Out[187…

```
334839
```

2.2) Drop column

In [188…

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

In [189…

```python
df.head()
```

Out[189…

| | caseNumber | treatmentDate | statWeight | stratum | sex | diagnosis | bodyPart | disposition | loca |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 150733174 | 7/11/2015 | 15.7762 | V | Male | 57 | 33 | 1 | |
| **1** | 150734723 | 7/6/2015 | 83.2157 | S | Male | 57 | 34 | 1 | |
| **2** | 150817487 | 8/2/2015 | 74.8813 | L | Female | 71 | 94 | 1 | |
| **3** | 150717776 | 6/26/2015 | 15.7762 | V | Male | 71 | 35 | 1 | |
| **4** | 150721694 | 7/4/2015 | 74.8813 | L | Female | 62 | 75 | 1 | |

2.3) Data imputation

In [190…

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

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File c:\Users\Feen_Phoorin\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas
\core\indexes\base.py:3805, in Index.get_loc(self, key)
   3804 try:
-> 3805     return self._engine.get_loc(casted_key)
   3806 except KeyError as err:

File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas\\_libs\\hashtable_class_helper.pxi:7081, in pandas._libs.hashtable.PyObjectHashT
able.get_item()

File pandas\\_libs\\hashtable_class_helper.pxi:7089, in pandas._libs.hashtable.PyObjectHashT
able.get_item()

KeyError: 'age'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[190], line 2
      1 # fillna
----> 2 df['age'] = df['age'].fillna(df['age'].median())

File c:\Users\Feen_Phoorin\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas
\core\frame.py:4102, in DataFrame.__getitem__(self, key)
   4100 if self.columns.nlevels > 1:
   4101     return self._getitem_multilevel(key)
-> 4102 indexer = self.columns.get_loc(key)
   4103 if is_integer(indexer):
   4104     indexer = [indexer]

File c:\Users\Feen_Phoorin\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas
\core\indexes\base.py:3812, in Index.get_loc(self, key)
   3807     if isinstance(casted_key, slice) or (
   3808         isinstance(casted_key, abc.Iterable)
   3809         and any(isinstance(x, slice) for x in casted_key)
   3810     ):
   3811         raise InvalidIndexError(key)
-> 3812     raise KeyError(key) from err
   3813 except TypeError:
   3814     # If we have a listlike key, _check_indexing_error will raise
   3815     #  InvalidIndexError. Otherwise we fall through and re-raise
   3816     #  the TypeError.
   3817     self._check_indexing_error(key)

KeyError: 'age'
```

[Q1] From the above cell, Why it showing an error?

Ans: **Because the line `ages = df.pop('age')` removes the 'age' column from df, so when I try to fill missing values in df['age'], the column no longer exists.**

[Q2] Fix the error from Q1 problem.

In [191…

```
# [Q2]

# hint: see the cell that run `df.pop()`
df["age"] = ages
```

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

df.head()
```

Out[191…

|   | caseNumber | treatmentDate | statWeight | stratum | sex | diagnosis | bodyPart | disposition | loca |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 150733174 | 7/11/2015 | 15.7762 | V | Male | 57 | 33 | 1 | |
| **1** | 150734723 | 7/6/2015 | 83.2157 | S | Male | 57 | 34 | 1 | |
| **2** | 150817487 | 8/2/2015 | 74.8813 | L | Female | 71 | 94 | 1 | |
| **3** | 150717776 | 6/26/2015 | 15.7762 | V | Male | 71 | 35 | 1 | |
| **4** | 150721694 | 7/4/2015 | 74.8813 | L | Female | 62 | 75 | 1 | |

2.4) Drop row that have missing value

In [192…

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

In [193…

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

Out[193…

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

## Datetime

2.5) Working with the datetime format

In [194…

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

```
C:\Users\Feen_Phoorin\AppData\Local\Temp\ipykernel_19824\3208943844.py:1: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
  df["treatmentDate"] = pd.to_datetime(df["treatmentDate"], format="%m/%d/%Y")
```

In [195…

```
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   sex            334837 non-null  object
 5   diagnosis      334837 non-null  int64
 6   bodyPart       334837 non-null  int64
 7   disposition    334837 non-null  int64
 8   location       334837 non-null  int64
 9   product        334837 non-null  int64
 10  age            334837 non-null  int64
dtypes: datetime64[ns](1), float64(1), int64(7), object(2)
memory usage: 30.7+ MB
```

In [196…

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

```
C:\Users\Feen_Phoorin\AppData\Local\Temp\ipykernel_19824\1686165144.py:1: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
  df['Year'] = df['treatmentDate'].dt.year
```

In [197…

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

```
C:\Users\Feen_Phoorin\AppData\Local\Temp\ipykernel_19824\404848564.py:1: SettingWithCopyWarn
ing:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
  df['Month'] = df['treatmentDate'].dt.month
```

In [198…

```
df.head()
```

Out[198…

| | caseNumber | treatmentDate | statWeight | stratum | sex | diagnosis | bodyPart | disposition | loca |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 150733174 | 2015-07-11 | 15.7762 | V | Male | 57 | 33 | 1 | |
| **1** | 150734723 | 2015-07-06 | 83.2157 | S | Male | 57 | 34 | 1 | |
| **2** | 150817487 | 2015-08-02 | 74.8813 | L | Female | 71 | 94 | 1 | |
| **3** | 150717776 | 2015-06-26 | 15.7762 | V | Male | 71 | 35 | 1 | |
| **4** | 150721694 | 2015-07-04 | 74.8813 | L | Female | 62 | 75 | 1 | |

[Q3] Can you change the format to DD/MM/YYYY? Show your work.

In [199…

```
# change the format to DD/MM/YYYY
df["treatmentDate"] = df["treatmentDate"].dt.strftime("%d/%m/%Y")
df
```

```
C:\Users\Feen_Phoorin\AppData\Local\Temp\ipykernel_19824\2894307848.py:2: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
  df["treatmentDate"] = df["treatmentDate"].dt.strftime("%d/%m/%Y")
```

Out[199…

| | caseNumber | treatmentDate | statWeight | stratum | sex | diagnosis | bodyPart | dispositio |
|---|---|---|---|---|---|---|---|---|
| **0** | 150733174 | 11/07/2015 | 15.7762 | V | Male | 57 | 33 | |
| **1** | 150734723 | 06/07/2015 | 83.2157 | S | Male | 57 | 34 | |
| **2** | 150817487 | 02/08/2015 | 74.8813 | L | Female | 71 | 94 | |
| **3** | 150717776 | 26/06/2015 | 15.7762 | V | Male | 71 | 35 | |
| **4** | 150721694 | 04/07/2015 | 74.8813 | L | Female | 62 | 75 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **334834** | 150739278 | 31/05/2015 | 15.0591 | V | Male | 59 | 76 | |
| **334835** | 150733393 | 11/07/2015 | 5.6748 | C | Female | 68 | 85 | |
| **334836** | 150819286 | 24/07/2015 | 15.7762 | V | Male | 71 | 79 | |
| **334837** | 150823002 | 08/08/2015 | 97.9239 | M | Female | 59 | 82 | |
| **334838** | 150723074 | 20/06/2015 | 49.2646 | M | Female | 57 | 34 | |

334837 rows × 13 columns

## Combine Dataframe by .merge and .concat

2.6 Merge

In [280…

```python
import pandas as pd

superstore_order = pd.read_csv('./sources/lab/Superstore/superstore_order.csv')
superstore_people = pd.read_csv('./sources/lab/Superstore/superstore_people.csv')
superstore_return = pd.read_csv('./sources/lab/Superstore/superstore_return.csv')
```

In [281…

```python
superstore_order.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8880 entries, 0 to 8879
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Row ID         8880 non-null   int64  
 1   Order ID       8880 non-null   object 
 2   Order Date     8880 non-null   object 
 3   Ship Date      8880 non-null   object 
 4   Ship Mode      8880 non-null   object 
 5   Customer ID    8880 non-null   object 
 6   Customer Name  8880 non-null   object 
 7   Segment        8880 non-null   object 
 8   Country        8880 non-null   object 
 9   City           8880 non-null   object 
 10  State          8880 non-null   object 
 11  Postal Code    8880 non-null   int64  
 12  Region         8880 non-null   object 
 13  Product ID     8880 non-null   object 
 14  Category       8880 non-null   object 
 15  Sub-Category   8880 non-null   object 
 16  Product Name   8880 non-null   object 
 17  Sales          8880 non-null   float64
 18  Quantity       8880 non-null   int64  
 19  Discount       8880 non-null   float64
 20  Profit         8880 non-null   float64
dtypes: float64(3), int64(3), object(15)
memory usage: 1.4+ MB
```

In [282…

```
superstore_people.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Person  4 non-null      object
 1   Region  4 non-null      object
dtypes: object(2)
memory usage: 196.0+ bytes
```

In [283…

```
superstore_return.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 296 entries, 0 to 295
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Returned  296 non-null    object
 1   Order ID  296 non-null    object
dtypes: object(2)
memory usage: 4.8+ KB
```

In [284…

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

Out[284…

|  | 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 × 2 columns

[Q4] What does the argument `how="inner"` do?

Ans: **Keeps only the rows with matching values in both DataFrames based on the specified `on = column`**

[Q5] In your opinion, what information that the result above conveys?

Ans: **The result above conveys customer return behavior by listing customers who have returned at least one product.**

More merging...

In [285…

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

Out[285…

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 19 | CA-2014-143336 | 27/08/2014 | 01/09/2014 | Second Class | ZD-21925 | Zuschuss Donatelli | Consumer | United States | Fr |
| **1** | 20 | CA-2014-143336 | 27/08/2014 | 01/09/2014 | Second Class | ZD-21925 | Zuschuss Donatelli | Consumer | United States | Fr |
| **2** | 21 | CA-2014-143336 | 27/08/2014 | 01/09/2014 | Second Class | ZD-21925 | Zuschuss Donatelli | Consumer | United States | Fr |
| **3** | 56 | CA-2016-111682 | 17/06/2016 | 18/06/2016 | First Class | TB-21055 | Ted Butterfield | Consumer | United States | |
| **4** | 57 | CA-2016-111682 | 17/06/2016 | 18/06/2016 | First Class | TB-21055 | Ted Butterfield | Consumer | United States | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **702** | 8870 | CA-2017-101805 | 01/12/2017 | 06/12/2017 | Standard Class | SH-19975 | Sally Hughsby | Corporate | United States | |
| **703** | 8871 | CA-2017-101805 | 01/12/2017 | 06/12/2017 | Standard Class | SH-19975 | Sally Hughsby | Corporate | United States | |
| **704** | 8872 | CA-2017-101805 | 01/12/2017 | 06/12/2017 | Standard Class | SH-19975 | Sally Hughsby | Corporate | United States | |
| **705** | 8873 | US-2014-105137 | 10/10/2014 | 10/10/2014 | Same Day | RB-19435 | Richard Bierner | Consumer | United States | Co |
| **706** | 8874 | US-2014-105137 | 10/10/2014 | 10/10/2014 | Same Day | RB-19435 | Richard Bierner | Consumer | United States | Co |

707 rows × 22 columns

2.7) Concatenate

In [286…

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

Out[286…

|   | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country |   |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | CA-2016-152156 | 08/11/2016 | 11/11/2016 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Hend |
| **1** | 2 | CA-2016-152156 | 08/11/2016 | 11/11/2016 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Hend |
| **2** | 3 | CA-2016-138688 | 12/06/2016 | 16/06/2016 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | An |
| **3** | 4 | US-2015-108966 | 11/10/2015 | 18/10/2015 | Standard Class | SO-20335 | Sean ODonnell | Consumer | United States | Laude |

4 rows × 23 columns

[Q6] What is the difference between `inner` and `outer` on parameter `join` in `pd.concat`?

Ans:

- **Inner Join: Included only the rows with matching indices in both DataFrames.**
- **Outer Join: Includes all rows in both DataFrames, filling missing values with NaN.**

## Groupby

In [287…

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

Out[287…

| | | 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 |

[Q7] Describe an information that the result above conveys?

Ans: **The result above conveys a summation of the sales activity, profit, quantity sold, and discounting patterns based on the customer segments and shipping modes.**

In [288…

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

superstore_order[['Profit', 'Sales', 'Profit Ratio']]
```

Out[288…

| | Profit | Sales | Profit Ratio |
|---|---|---|---|
| **0** | 41.9136 | 261.9600 | 0.1600 |
| **1** | 219.5820 | 731.9400 | 0.3000 |
| **2** | 6.8714 | 14.6200 | 0.4700 |
| **3** | -383.0310 | 957.5775 | -0.4000 |
| **4** | 2.5164 | 22.3680 | 0.1125 |
| **...** | ... | ... | ... |
| **8875** | -34.7580 | 185.3760 | -0.1875 |
| **8876** | -153.2024 | 58.9240 | -2.6000 |
| **8877** | 225.6000 | 480.0000 | 0.4700 |
| **8878** | 9.5340 | 34.0500 | 0.2800 |
| **8879** | 92.5056 | 192.7200 | 0.4800 |

8880 rows × 3 columns

In [289…

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

Out[289…

| Category | Sub-Category | mean_profit_ratio |
|---|---|---|
| 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 |

[Q8] Describe an information that the result above conveys?

Ans: **The results above show the average profit ratio for each subcategory in each category, with positive values indicating profit and negative values indicating loss.**

## Pivot and Melt

Pivot

In [290…

```
superstore_order.pivot_table(index="State", columns="Ship Mode", values="Order ID", aggfunc
```

Out[290…

| 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 |

In [291…

```
pivot_table_result = superstore_order.pivot_table(index="State", columns="Ship Mode", value
pivot_table_result
```

Out[291…

| 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 |

| Ship Mode | First Class | Same Day | Second Class | Standard Class |
|---|---|---|---|---|
| State | | | | |
| 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

In [292…

```
melted_result = pd.melt(pivot_table_result.reset_index(), id_vars=["State"], var_name="Ship
melted_result
```

Out[292…

| | 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 × 3 columns

[Q9] What is the advantage of using `melt`?

Ans: **Using `melt()` helps simplify data manipulation by turning wide-format data into a more flexible, readable, and analyzable long format. It's especially useful for tasks like grouping, summarizing, and visualizing data efficiently.**

[Q10] From the superstore_order, display the ascending order considering values in the 'Profit' column to group the 'Category'.

`I have provided 2 answers for [Q10] , since this question is confusing.`

In [302...

```
superstore_order.pivot_table(index="Category", values="Profit", aggfunc="sum").sort_values(
```

Out[302...

| | Profit |
|---|---|
| **Category** | |
| **Furniture** | 16858.5619 |
| **Office Supplies** | 105827.0238 |
| **Technology** | 133410.4932 |

In [303...

```
superstore_order.sort_values(by='Profit', ascending=True).groupby('Category').head(5)
# Display the 5 rows with the lowest profit in each group.
```

Out[303…

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | |
|---|---|---|---|---|---|---|---|---|---|---|
| **7772** | 7773 | CA-2016-108196 | 25/11/2016 | 02/12/2016 | Standard Class | CS-12505 | Cindy Stewart | Consumer | United States | |
| **683** | 684 | US-2017-168116 | 04/11/2017 | 04/11/2017 | Same Day | GT-14635 | Grant Thornton | Corporate | United States | |
| **3011** | 3012 | CA-2017-134845 | 17/04/2017 | 23/04/2017 | Standard Class | SR-20425 | Sharelle Roach | Home Office | United States | |
| **4991** | 4992 | US-2017-122714 | 07/12/2017 | 13/12/2017 | Standard Class | HG-14965 | Henry Goldwyn | Corporate | United States | |
| **3151** | 3152 | CA-2015-147830 | 15/12/2015 | 18/12/2015 | First Class | NF-18385 | Natalie Fritzler | Consumer | United States | |
| **5310** | 5311 | CA-2017-131254 | 19/11/2017 | 21/11/2017 | First Class | NC-18415 | Nathan Cano | Consumer | United States | |
| **1199** | 1200 | CA-2016-130946 | 08/04/2016 | 12/04/2016 | Standard Class | ZC-21910 | Zuschuss Carroll | Consumer | United States | |
| **2697** | 2698 | CA-2014-145317 | 18/03/2014 | 23/03/2014 | Standard Class | SM-20320 | Sean Miller | Home Office | United States | J |
| **27** | 28 | US-2015-150630 | 17/09/2015 | 21/09/2015 | Standard Class | TB-21520 | Tracy Blumstein | Consumer | United States | P |
| **3324** | 3325 | CA-2014-165309 | 11/11/2014 | 15/11/2014 | Standard Class | KD-16270 | Karen Daniels | Consumer | United States | |
| **2928** | 2929 | US-2017-120390 | 19/10/2017 | 26/10/2017 | Standard Class | TH-21550 | Tracy Hopkins | Home Office | United States | |
| **5320** | 5321 | US-2017-162558 | 02/10/2017 | 05/10/2017 | First Class | Dp-13240 | Dean percer | Home Office | United States | |
| **463** | 464 | CA-2016-109869 | 22/04/2016 | 29/04/2016 | Standard Class | TN-21040 | Tanja Norvell | Home Office | United States | |
| **1369** | 1370 | US-2015-103471 | 24/12/2015 | 28/12/2015 | Standard Class | JR-15670 | Jim Radford | Consumer | United States | |

|  | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country |
|---|---|---|---|---|---|---|---|---|---|
| **6639** | 6640 | CA-2014-108609 | 31/10/2014 | 02/11/2014 | Second Class | AJ-10780 | Anthony Jacobs | Corporate | United States |

15 rows × 23 columns

[Q11] 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

In [306…

```
#enter your code here
superstore_order["Total Price Before Discount"] = superstore_order["Sales"]*superstore_orde

superstore_order.groupby(["Product ID", "Category"]).agg(mean_total_price_bf_discounting =
```

Out[306…

|  |  | mean_total_price_bf_discounting |
|---|---|---|
| **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 × 1 columns

[Q12] Complete the function to apply `ratio` column that calculates from `First Class` and `Standard Class` columns on `pivot_table_result`

In [295…

```
# [Q12] Complete the function to apply `ratio` column that calculates from `First Class` an

# function to transform the ratio
def get_class_ratio(row):

    # get the first class column
    first_class = row['First Class']

    # get the standard class column
    standard_class = row['Standard Class']

    # calculate the ratio
    ratio = first_class / standard_class
    return ratio

pivot_table_result["ratio"] = pivot_table_result.apply(get_class_ratio, axis=1)

pivot_table_result.head()
```

Out[295…

| Ship Mode | First Class | Same Day | Second Class | Standard Class | ratio |
|---|---|---|---|---|---|
| **State** | | | | | |
| **Alabama** | 9.0 | 1.0 | 18.0 | 30.0 | 0.300000 |
| **Arizona** | 42.0 | 15.0 | 22.0 | 123.0 | 0.341463 |
| **Arkansas** | 10.0 | 2.0 | 8.0 | 35.0 | 0.285714 |
| **California** | 302.0 | 106.0 | 346.0 | 1000.0 | 0.302000 |
| **Colorado** | 43.0 | 5.0 | 32.0 | 95.0 | 0.452632 |

[Q13] After complete Q12, What does the `apply` function do?

Ans: `apply` **function in Pandas is used to apply function along the rows or columns of a DataFrame**

[Q14] Create a new column(`short_ratio`) that works the same as Q12 but with `lambda` function

In [ ]:

```
# [Q14] Create a new column(`short_ratio`) that works the same as Q12 but with `lambda` fun

pivot_table_result["short_ratio"] = pivot_table_result.apply(lambda row: row['First Class']

pivot_table_result.head()
```

Out[ ]:

| Ship Mode | First Class | Same Day | Second Class | Standard Class | ratio | short_ratio |
|---|---|---|---|---|---|---|
| **State** | | | | | | |
| **Alabama** | 9.0 | 1.0 | 18.0 | 30.0 | 0.300000 | 0.300000 |
| **Arizona** | 42.0 | 15.0 | 22.0 | 123.0 | 0.341463 | 0.341463 |
| **Arkansas** | 10.0 | 2.0 | 8.0 | 35.0 | 0.285714 | 0.285714 |
| **California** | 302.0 | 106.0 | 346.0 | 1000.0 | 0.302000 | 0.302000 |
| **Colorado** | 43.0 | 5.0 | 32.0 | 95.0 | 0.452632 | 0.452632 |

[Q15] What is the difference between using `function` in apply and `lambda` function? give 2 examples use case.

Ans: **A** `function` **is generally used when there is a need to reuse logic, handle complex calculations, or perform multiple operations, but it requires a prior definition. In contrast, a** `lambda` **function is perfect for simple and concise logic, especially when the operation is a one-time task that doesn't need to be reused or require a prior definition.**

## Example use case :

---

In [317…

```
df1 = pd.read_csv('./sources/lab/Superstore/superstore_order.csv')
```

In [318…

```
df1['Ship Mode'].unique()
```

Out[318…

```
array(['Second Class', 'Standard Class', 'First Class', 'Same Day'],
      dtype=object)
```

In [319…

```
# using apply regular function to define the shipping rate by shipping mode (ignore a dista
def calculate_shipping_rate(row):
    if row['Ship Mode'] == 'Same Day':
        shipping_rate = 100.00
    elif row['Ship Mode'] == 'First Class':
        shipping_rate = 80.00
    elif row['Ship Mode'] == 'Second Class':
        shipping_rate = 50.00
    else:
        shipping_rate = 30.00

    return shipping_rate

df1['Ship Rate'] = df1.apply(calculate_shipping_rate, axis=1)

df1[['Ship Mode', 'Ship Rate']]
```

Out[319…

| | Ship Mode | Ship Rate |
|---|---|---|
| **0** | Second Class | 50.0 |
| **1** | Second Class | 50.0 |
| **2** | Second Class | 50.0 |
| **3** | Standard Class | 30.0 |
| **4** | Standard Class | 30.0 |
| **...** | ... | ... |
| **8875** | Standard Class | 30.0 |
| **8876** | Standard Class | 30.0 |
| **8877** | Standard Class | 30.0 |
| **8878** | Standard Class | 30.0 |
| **8879** | First Class | 80.0 |

8880 rows × 2 columns

---

In [320…

```
# using apply lambda to calculate total price
superstore_order['Total Price'] = superstore_order.apply(lambda row : row['Total Price Befo

superstore_order[['Total Price Before Discount', 'Discount', 'Total Price']]
```

Out[320…

| | Total Price Before Discount | Discount | Total Price |
|---|---|---|---|
| **0** | 523.9200 | 0.00 | 523.920000 |
| **1** | 2195.8200 | 0.00 | 2195.820000 |
| **2** | 29.2400 | 0.00 | 29.240000 |
| **3** | 4787.8875 | 0.45 | 2633.338125 |
| **4** | 44.7360 | 0.20 | 35.788800 |
| **...** | ... | ... | ... |
| **8875** | 370.7520 | 0.20 | 296.601600 |
| **8876** | 58.9240 | 0.80 | 11.784800 |
| **8877** | 1920.0000 | 0.00 | 1920.000000 |
| **8878** | 102.1500 | 0.00 | 102.150000 |
| **8879** | 2119.9200 | 0.00 | 2119.920000 |

8880 rows × 3 columns