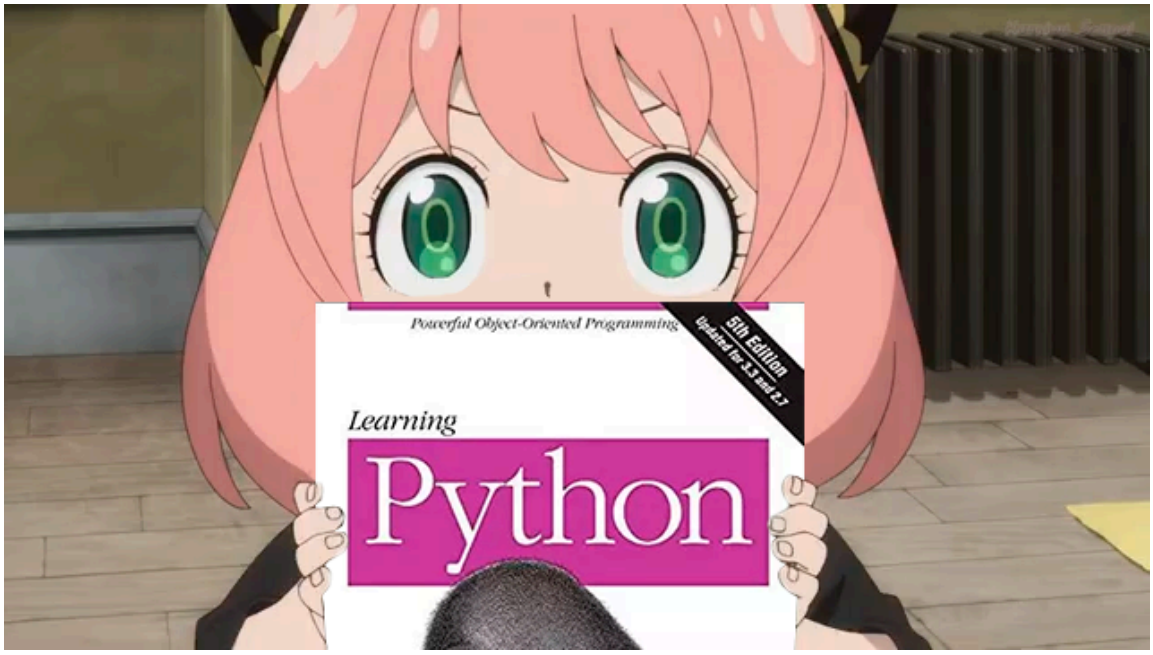
# Lab 3: Data Preparation

### CPE232 Data Models

---

For this lab, we will learn how to perform data preparation for data analysis after receiving the raw data from the data source.




# [1] Reviews on Pandas

1.1) Discover

- methods to explore and understand your DataFrame

In [72]:
```
import pandas as pd

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

Out[72]:

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

In [73]:
```
# see the shape of the dataframe
print(df.shape)
```

```
(334839, 12)
```

In [74]:
```
# seeing the summary of the dataframe
print(df.info())
```

```
<class 'pandas.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  str
 2   statWeight     334839 non-null  float64
 3   stratum        334839 non-null  str
 4   age            334839 non-null  int64
 5   sex            334837 non-null  str
 6   race           205014 non-null  str
 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), str(4)
memory usage: 30.7 MB
None
```

In [75]:
```
# seeing the stats of the column in dataframe
df.describe()
```

Out[75]:

| | caseNumber | statWeight | age | diagnosis | bodyPart | dispc |
|---|---|---|---|---|---|---|
| **count** | 3.348390e+05 | 334839.000000 | 334839.000000 | 334839.000000 | 334839.000000 | 334839.0 |
| **mean** | 1.510271e+08 | 39.343028 | 31.385451 | 60.154591 | 64.374192 | 1.3 |
| **std** | 1.720330e+06 | 34.142933 | 26.105098 | 6.170699 | 24.002331 | 0.9 |
| **min** | 1.501032e+08 | 4.965500 | 0.000000 | 41.000000 | 0.000000 | 1.0 |
| **25%** | 1.504405e+08 | 15.059100 | 10.000000 | 57.000000 | 35.000000 | 1.0 |
| **50%** | 1.507358e+08 | 15.776200 | 23.000000 | 59.000000 | 75.000000 | 1.0 |
| **75%** | 1.510231e+08 | 74.881300 | 51.000000 | 64.000000 | 82.000000 | 1.0 |
| **max** | 1.603418e+08 | 97.923900 | 107.000000 | 74.000000 | 94.000000 | 9.0 |

1.2) Selecting variables

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

In [76]:
```
# select columns based on the data type
df.select_dtypes(include=['number'])
```

Out[76]:

| | 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 [77]:
```
# select column by .loc
df.loc[:6,'treatmentDate':'diagnosis']
```

Out[77]:

| | 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 [78]:
```
# select row by .iloc
df.iloc[0:5]
```

Out[78]:

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

1.3) Filtering the data

In [79]:
```
# filter rows based on the condition
df[df['age'] > 50]
```

Out[79]:

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

85235 rows × 12 columns

In [80]:
```
# filter coloum based on column name
df.filter(like='age')
```

Out[80]:

| | 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 [81]:
```
# sort the dataframe based on column name and ascending order
df.sort_values(by='statWeight', ascending=False)
```

Out[81]:

| | caseNumber | treatmentDate | statWeight | stratum | age | sex | race | diagnosis |
|---|---|---|---|---|---|---|---|---|
| **275174** | 150343700 | 3/9/2015 | 97.9239 | M | 48 | Male | NaN | 57 |
| **36** | 151029422 | 10/6/2015 | 97.9239 | M | 37 | Male | White | 64 |
| **334806** | 150612491 | 5/29/2015 | 97.9239 | M | 18 | Female | White | 59 |
| **334810** | 150725804 | 7/8/2015 | 97.9239 | M | 33 | Female | Black | 71 |
| **275161** | 150450816 | 4/13/2015 | 97.9239 | M | 24 | Male | White | 71 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **44011** | 160222258 | 12/29/2015 | 4.9655 | C | 2 | Female | Other | 71 |
| **325320** | 151213065 | 11/29/2015 | 4.9655 | C | 16 | Female | White | 62 |
| **43891** | 160113865 | 12/28/2015 | 4.9655 | C | 4 | Male | White | 59 |
| **43628** | 151130111 | 11/9/2015 | 4.9655 | C | 13 | Male | Black | 53 |
| **43523** | 151139237 | 11/16/2015 | 4.9655 | C | 2 | Female | Black | 57 |

334839 rows × 12 columns

In [82]:
```
# sort the index of the dataframe
df.sort_index()
```

Out[82]:

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

334839 rows × 12 columns

1.5) Add/Remove

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

In [83]:
```
# Dropping the column
df.drop(columns=['disposition'])
```

Out[83]:

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

334839 rows × 11 columns

In [84]:
```
# Adding column and create into a new column
df.assign(new_column=df['diagnosis'] + df['bodyPart'])
```

Out[84]:

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

334839 rows × 13 columns

In [85]:
```
# 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 [86]:
```
# replaceing the missing values with a specified value
df.fillna(value=0)
```

Out[86]:

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

334839 rows × 11 columns

In [87]:
```
# Remove the rows with missing values
df.dropna()
```

Out[87]:

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

205014 rows × 11 columns

---

# [2] Data Cleaning and Preparation

## .isnull, .dropna, .fillna

2.1) checking

In [88]:
```
df.columns
```

Out[88]:
```
Index(['caseNumber', 'treatmentDate', 'statWeight', 'stratum', 'sex', 'race',
       'diagnosis', 'bodyPart', 'disposition', 'location', 'product'],
      dtype='str')
```

In [89]:
```
# isnull checking
df.isnull().sum()
```

Out[89]:
```
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 [90]:
```
# percentage of missing values for the race
df.race.isnull().sum()/df.shape[0]*100
```

Out[90]:
```
np.float64(38.772365226272925)
```

In [91]:
```
df.shape[0]
```

Out[91]:
```
334839
```

2.2) Drop column

In [92]:
```
# remove column by using
df = df.drop(columns=['race'])
```

In [93]:
```
df.head()
```

Out[93]:

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

2.3) Data imputation

In [94]:
```
# fillna
df['age'] = df['age'].fillna(df['age'].median())
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File c:\Users\win25\Desktop\Desktop\work\CPE232 Data models\.venv\Lib\site-packages
\pandas\core\indexes\base.py:3641, in Index.get_loc(self, key)
   3640 try:
-> 3641     return self._engine.get_loc(casted_key)
   3642 except KeyError as err:

File pandas/_libs/index.pyx:168, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/index.pyx:197, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/hashtable_class_helper.pxi:7668, in pandas._libs.hashtable.PyObjec
tHashTable.get_item()

File pandas/_libs/hashtable_class_helper.pxi:7676, in pandas._libs.hashtable.PyObjec
tHashTable.get_item()

KeyError: 'age'

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

KeyError                                  Traceback (most recent call last)
Cell In[94], line 2
      1 # fillna
----> 2 df['age'] = df[      ].fillna(df['age'].median())

File c:\Users\win25\Desktop\Desktop\work\CPE232 Data models\.venv\Lib\site-packages
\pandas\core\frame.py:4378, in DataFrame.__getitem__(self, key)
   4376 if self.columns.nlevels > 1:
   4377     return self._getitem_multilevel(key)
-> 4378 indexer = self.columns.get_loc(key)
   4379 if is_integer(indexer):
   4380     indexer = [indexer]

File c:\Users\win25\Desktop\Desktop\work\CPE232 Data models\.venv\Lib\site-packages
\pandas\core\indexes\base.py:3648, in Index.get_loc(self, key)
   3643     if isinstance(casted_key, slice) or (
   3644         isinstance(casted_key, abc.Iterable)
   3645         and any(isinstance(x, slice) for x in casted_key)
   3646     ):
   3647         raise InvalidIndexError(key) from err
-> 3648     raise KeyError(key) from err
   3649 except TypeError:
   3650     # If we have a listlike key, _check_indexing_error will raise
   3651     #  InvalidIndexError. Otherwise we fall through and re-raise
   3652     #  the TypeError.
   3653     self._check_indexing_error(key)

KeyError: 'age'
```

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

Ans: Because Dataframe df don't have Column name "age" so it will occurs error when trying to access using df["age"]

[Q2] Fix the error from Q1 problem.

In [ ]:
```
# [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[ ]:

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

2.4) Drop row that have missing value

In [ ]:
```
# remove column by using .dropna()
df = df.dropna()
```

In [ ]:
```
df.isnull().sum()
```

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

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

```
<class 'pandas.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[us]
 2   statWeight     334837 non-null  float64
 3   stratum        334837 non-null  str
 4   sex            334837 non-null  str
 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[us](1), float64(1), int64(7), str(2)
memory usage: 30.7 MB
```

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

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

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

Out[ ]:

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

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

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

Out[ ]:

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

334837 rows × 14 columns

## Combine Dataframe by .merge and .concat

2.6 Merge

In [ ]:
```
import pandas as pd

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')
```

In [ ]:
```
superstore_order.head()
```

Out[ ]:

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Countr |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | CA-2016-152156 | 08/11/2016 | 11/11/2016 | Second Class | CG-12520 | Claire Gute | Consumer | Unite Stat |
| **1** | 2 | CA-2016-152156 | 08/11/2016 | 11/11/2016 | Second Class | CG-12520 | Claire Gute | Consumer | Unite Stat |
| **2** | 3 | CA-2016-138688 | 12/06/2016 | 16/06/2016 | Second Class | DV-13045 | Darrin Van Huff | Corporate | Unite Stat |
| **3** | 4 | US-2015-108966 | 11/10/2015 | 18/10/2015 | Standard Class | SO-20335 | Sean ODonnell | Consumer | Unite Stat |
| **4** | 5 | US-2015-108966 | 11/10/2015 | 18/10/2015 | Standard Class | SO-20335 | Sean ODonnell | Consumer | Unite Stat |

5 rows × 21 columns

In [ ]:
```
superstore_people.head()
```

Out[ ]:

| | Person | Region |
|---|---|---|
| **0** | Anna Andreadi | West |
| **1** | Chuck Magee | East |
| **2** | Kelly Williams | Central |
| **3** | Cassandra Brandow | South |

In [ ]:
```
superstore_return.head()
```

Out[ ]:

| | Returned | Order ID |
|---|---|---|
| **0** | Yes | CA-2017-153822 |
| **1** | Yes | CA-2017-129707 |
| **2** | Yes | CA-2014-152345 |
| **3** | Yes | CA-2015-156440 |
| **4** | Yes | US-2017-155999 |

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

Out[ ]:

| | 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: ' how = "inner" ' is how to merge 2 table between superstore_order and superstore_return[superstore_return["Returned"]=="Yes"]
"inner" mean these 2 table merge using inner join (CustomerID) after joining we get superstore_return Dataframe with "Returned" Column
from superstore_return[superstore_return["Returned"]=="Yes" Dataframe

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

Ans: It shows CustomerID of Customer who returns the order to supermarket

2.7) Concatenate

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

Out[ ]:

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

4 rows × 23 columns

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

Out[ ]:

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Co |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | CA-2016-152156 | 08/11/2016 | 11/11/2016 | Second Class | CG-12520 | Claire Gute | Consumer | U |
| **1** | 2 | CA-2016-152156 | 08/11/2016 | 11/11/2016 | Second Class | CG-12520 | Claire Gute | Consumer | U |
| **2** | 3 | CA-2016-138688 | 12/06/2016 | 16/06/2016 | Second Class | DV-13045 | Darrin Van Huff | Corporate | U |
| **3** | 4 | US-2015-108966 | 11/10/2015 | 18/10/2015 | Standard Class | SO-20335 | Sean ODonnell | Consumer | U |
| **4** | 5 | US-2015-108966 | 11/10/2015 | 18/10/2015 | Standard Class | SO-20335 | Sean ODonnell | Consumer | U |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **8875** | 8876 | US-2016-141264 | 13/08/2016 | 19/08/2016 | Standard Class | CT-11995 | Carol Triggs | Consumer | U |
| **8876** | 8877 | US-2016-141264 | 13/08/2016 | 19/08/2016 | Standard Class | CT-11995 | Carol Triggs | Consumer | U |
| **8877** | 8878 | CA-2017-126928 | 17/09/2017 | 23/09/2017 | Standard Class | GZ-14470 | Gary Zandusky | Consumer | U |
| **8878** | 8879 | CA-2017-126928 | 17/09/2017 | 23/09/2017 | Standard Class | GZ-14470 | Gary Zandusky | Consumer | U |

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Co |
|---|---|---|---|---|---|---|---|---|---|
| **8879** | 8880 | US-2015-107944 | 23/03/2015 | 25/03/2015 | First Class | AM-10360 | Alice McCarthy | Corporate | U |

8880 rows × 23 columns

[Q6] What is the difference between `inner` and `outer` on parameter `join` in `pd.concat`?
Ans 'inner' is inner join between 2 Dataframe of pd.concat, inner join works like intersection its join two Dataframe without having NaN value
while 'outer' is outer join between 2 Dataframe of pd.concat, outer join works like union after join sometimes new Dataframe will have empty entity

[Q7] Append new record at the end of superstore_order. Set the 'Row ID' to your student ID and mock other required fields based on existing entries.

In [ ]:
```
# write your code here
from datetime import datetime
mydata = {
    'Row ID' : 67070501021,
    "Order ID" : "TH-2015-107944",
    "Order Date" : "12/12/2012",
    "Ship Date"  : "22/12/2012",
    "Ship Mode" : "Standard Class",
    "Customer ID" : "TT-10210",
    "Customer Name" : "Thanaboon Tikaew",
    "Segment" : "Consumer",
    "Country" : "Thailand",
    "City" : "MaeMoh",
    "Postal Code" : 52220,
    "Region" : "North",
    "Product ID" : "OFF-MM-10000353",
    "Category" : "Furniture",
    "Sub-Category" : "Bookcases",
    "Product Name" : "Alien Cat",
    "Sales" : 23.22,
    "Quantity" : 1,
    "Discount" : 0.00,
    "Profit" : 3.11
}
my_df = pd.DataFrame([mydata])
superstore_order = pd.concat([superstore_order, my_df], ignore_index=True)
superstore_order.tail()
```

Out[ ]:

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Seg |
|---|---|---|---|---|---|---|---|---|
| **8876** | 8.877000e+03 | US-2016-141264 | 13/08/2016 | 19/08/2016 | Standard Class | CT-11995 | Carol Triggs | Cons |
| **8877** | 8.878000e+03 | CA-2017-126928 | 17/09/2017 | 23/09/2017 | Standard Class | GZ-14470 | Gary Zandusky | Cons |
| **8878** | 8.879000e+03 | CA-2017-126928 | 17/09/2017 | 23/09/2017 | Standard Class | GZ-14470 | Gary Zandusky | Cons |
| **8879** | 8.880000e+03 | US-2015-107944 | 23/03/2015 | 25/03/2015 | First Class | AM-10360 | Alice McCarthy | Corp |
| **8880** | 6.707050e+10 | TH-2015-107944 | 12/12/2012 | 22/12/2012 | Standard Class | TT-10210 | Thanaboon Tikaew | Cons |

5 rows × 22 columns

## Groupby

In [ ]:
```
superstore_order["Profit Ratio"] = superstore_order["Profit"]/superstore_order["Sal

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

Out[ ]:

| Category | Sub-Category | mean_profit_ratio |
|---|---|---|
| **Furniture** | **Bookcases** | -0.126473 |
| | **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: Mean profit ratio of each product grouping by Category and Sub-Category

## Pivot and Melt

Pivot

In [ ]:
```
superstore_order.pivot_table(index="State", columns="Ship Mode", values="Order ID",
```

Out[ ]:

| Ship Mode<br>State | First Class | Same Day | Second Class | Standard Class |
|---|---|---|---|---|
| **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 [ ]:
```
pivot_table_result = superstore_order.pivot_table(index="State", columns="Ship Mode
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

In [ ]:
```
melted_result = pd.melt(pivot_table_result.reset_index(), id_vars=["State"], var_na
```

```
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]
```

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

Ans : Its tidy data from wide format to long format which easier to read

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

In [ ]:
```
#enter your code
category_profit = superstore_order.groupby('Category')['Profit'].sum()
category_profit.sort_values(ascending=True)
```

Out[ ]:
```
Category
Furniture           16861.6719
Office Supplies    105827.0238
Technology         133410.4932
Name: Profit, dtype: float64
```

[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 [ ]:
```
#enter your code here
superstore_order["Total Price"] = superstore_order["Sales"] * superstore_order["Qua
Grouped_mean_price = superstore_order.groupby(["Product ID","Category"])["Total Pri
Grouped_mean_price
```

Out[ ]:
```
Product ID       Category
FUR-BO-10000112  Furniture     7426.566000
FUR-BO-10000330  Furniture     1258.192000
FUR-BO-10000362  Furniture     1726.898000
FUR-BO-10000468  Furniture      426.532400
FUR-BO-10000711  Furniture     3194.100000
                                   ...
TEC-PH-10004912  Technology     747.320000
TEC-PH-10004922  Technology     673.249500
TEC-PH-10004924  Technology      57.149333
TEC-PH-10004959  Technology     412.009000
TEC-PH-10004977  Technology    2441.475429
Name: Total Price, Length: 1847, dtype: float64
```

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

In [ ]:
```python
# 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[ ]:

| Ship Mode / State | First Class | Same Day | Second Class | Standard Class | ratio |
|---|---|---|---|---|---|
| **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 Q13, What does the `apply` function do?

Ans: Apply function calls "get_class_ratio" on each row of pivot table result also "axis = 1" mean we calls by row

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

In [ ]:
```
pivot_table_result["short_ratio"] = pivot_table_result.apply(lambda row: row["First

pivot_table_result.head()
```

Out[ ]:

| Ship Mode<br>State | First Class | Same Day | Second Class | Standard Class | ratio | short_ratio |
|---|---|---|---|---|---|---|
| **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 `apply` and `lambda` function? give 2 examples use case.

Ans : "apply" is Pandas method that operate function in each row of Dataframe while "lambda" is Python function that uses on Math/Logic calculation

Ex1: In superstore_order["Customer Name"].apply(len) we called "len" function to find length of Customer name in each row

In [ ]:
```
superstore_order["Customer Name"].apply(len)
```

Out[ ]:
```
0       11
1       11
2       15
3       13
4       13
        ..
8876    12
8877    13
8878    13
8879    14
8880    16
Name: Customer Name, Length: 8881, dtype: int64
```

Ex2: In superstore_order.apply(lambda row: row["Total Price"]/100, axis=1)
we use "apply" to access all row of superstore_order
and in each row we use lambda to calculate its Totalprice/100 without using define function

In [ ]:
```
superstore_order.apply(lambda row: row["Total Price"]/100, axis=1)
```

Out[ ]:
```
0        5.239200
1       21.958200
2        0.292400
3       47.878875
4        0.447360
           ...
8876     0.589240
8877    19.200000
8878     1.021500
8879    21.199200
8880     0.232200
Length: 8881, dtype: float64
```