Drop rows where all the observations are missing.

```
my_DF.dropna(how='all')
'Output':
         Capital  LGAs  Population    State
0           Yola  22.0   3178950.0  Adamawa
2            NaN  17.0   2321339.0     Yobe
3  Port-Harcourt  23.0         NaN      NaN
4        Jalingo  16.0   2294800.0   Taraba
```

Drop rows based on an observation threshold. By adjusting the **thresh** attribute, we can drop rows where the number of observations in the row is less than the **thresh** value.

```
# drop rows where number of NaN is less than 3
my_DF.dropna(thresh=3)
'Output':
   Capital  LGAs  Population    State
0     Yola  22.0   3178950.0  Adamawa
2      NaN  17.0   2321339.0     Yobe
4  Jalingo  16.0   2294800.0   Taraba
```

Imputing Values into Missing Data

Imputing values as substitutes for missing data is a standard practice in preparing data for machine learning. Pandas has a **fillna()** function for this purpose. A simple approach is to fill **NaNs** with zeros.

```
my_DF.fillna(0) # we can also run my_DF.replace(np.nan, 0)
'Output':
         Capital  LGAs  Population    State
0           Yola  22.0   3178950.0  Adamawa
1              0   0.0         0.0        0
2              0  17.0   2321339.0     Yobe
3  Port-Harcourt  23.0         0.0        0
4        Jalingo  16.0   2294800.0   Taraba
```

Another tactic is to fill missing values with the mean of the column value.

```
my_DF.fillna(my_DF.mean())
'Output':
         Capital  LGAs  Population    State
0           Yola  22.0   3178950.0  Adamawa
1            NaN  19.5   2598363.0      NaN
2            NaN  17.0   2321339.0     Yobe
3  Port-Harcourt  23.0   2598363.0      NaN
4        Jalingo  16.0   2294800.0   Taraba
```

Data Aggregation (Grouping)

We will touch briefly on a common practice in data science, and that is grouping a set of data attributes, either for retrieving some group statistics or applying a particular set of functions to the group. Grouping is commonly used for data exploration and plotting graphs to understand more about the dataset. Missing data are automatically excluded in a grouping operation.

Let's see examples of how this works.

```
# create a data frame
my_DF = pd.DataFrame({'Sex': ['M', 'F', 'M', 'F','M', 'F','M', 'F'],
 'Age': np.random.randint(15,60,8),
 'Salary': np.random.rand(8)*10000})
my_DF
'Output':
   Age       Salary Sex
0   54  6092.596170   M
1   57  3148.886141   F
2   37  5960.916038   M
3   23  6713.133849   F
4   34  5208.240349   M
5   25  2469.118934   F
6   50  1277.511182   M
7   54  3529.201109   F
```