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
```

Let's find the mean age and salary for observations in our dataset grouped by **Sex**.

```
my_DF.groupby('Sex').mean()
'Output':
       Age       Salary
Sex
F    39.75  3965.085008
M    43.75  4634.815935
```

We can group by more than one variable. In this case for each Sex group, also group the age and find the mean of the other numeric variables.

```
my_DF.groupby([my_DF['Sex'], my_DF['Age']]).mean()
'Output':
              Salary
Sex Age
F   23   6713.133849
    25   2469.118934
    54   3529.201109
    57   3148.886141
M   34   5208.240349
    37   5960.916038
    50   1277.511182
    54   6092.596170
```

Also, we can use a variable as a group key to run a group function on another variable or sets of variables.

```
my_DF['Age'].groupby(my_DF['Salary']).mean()
'Output':
Salary
1277.511182    50
2469.118934    25
3148.886141    57
3529.201109    54
5208.240349    34
5960.916038    37
```

```
6092.596170    54
6713.133849    23
Name: Age, dtype: int64
```

Statistical Summaries

Descriptive statistics is an essential component of the data science pipeline. By investigating the properties of the dataset, we can gain a better understanding of the data and the relationship between the variables. This information is useful in making decisions about the type of data transformations to carry out or the types of learning algorithms to spot check. Let's see some examples of simple statistical functions in Pandas.

First, we'll create a Pandas dataframe.

```
my_DF = pd.DataFrame(np.random.randint(10,80,[7,4]),\
            columns=['First','Second','Third', 'Fourth'])
'Output':
   First  Second  Third  Fourth
0     47      32     66      52
1     37      66     16      22
2     24      16     63      36
3     70      47     62      12
4     74      61     44      18
5     65      73     21      37
6     44      47     23      13
```

Use the **describe** function to obtain summary statistics of a dataset. Eight statistical measures are displayed. They are count, mean, standard deviation, minimum value, 25th percentile, 50th percentile or median, 75th percentile, and the maximum value.

```
my_DF.describe()
'Output':
           First     Second      Third     Fourth
count   7.000000   7.000000   7.000000   7.000000
mean   51.571429  48.857143  42.142857  27.142857
std    18.590832  19.978560  21.980511  14.904458
min    24.000000  16.000000  16.000000  12.000000
```