```
'Output':
Out[246]:
   First  Second  Third  First  Second  Third
0    2.0     3.0    9.0      3       6      3
1    8.0     7.0    7.0      2       2      1
2    8.0     6.0    4.0      9       3      8
3    NaN     NaN    NaN      2       9      2
```

Handling Missing Data

Dealing with missing data is an integral part of the data cleaning/data analysis process. Moreover, some machine learning algorithms will not work in the presence of missing data. Let's see some simple Pandas methods for identifying and removing missing data, as well as imputing values into missing data.

Identifying Missing Data

In this section, we'll use the **isnull()** method to check if missing cells exist in a DataFrame.

```
# let's create a data frame with missing data
my_DF = pd.DataFrame({'age': [15,17,np.nan,29,25], \
            'state_of_origin':['Lagos', 'Cross River', 'Kano',
            'Abia', np.nan]})
my_DF
'Output':
    age state_of_origin
0  15.0           Lagos
1  17.0     Cross River
2   NaN            Kano
3  29.0            Abia
4  25.0             NaN
```

Let's check for missing data in this data frame. The **isnull()** method will return **True** where there is a missing data, whereas the **notnull()** function returns **False**.

```
my_DF.isnull()
'Output':
     age  state_of_origin
0  False            False
1  False            False
2   True            False
3  False            False
4  False             True
```

However, if we want a single answer (i.e., either **True** or **False**) to report if there is a missing data in the data frame, we will first convert the DataFrame to a NumPy array and use the function **any()**.

The **any** function returns **True** when at least one of the elements in the dataset is **True**. In this case, **isnull()** returns a DataFrame of booleans where **True** designates a cell with a missing value.

Let's see how that works.

```
my_DF.isnull().values.any()
'Output':  True
```

Removing Missing Data

Pandas has a function **dropna()** which is used to filter or remove missing data from a DataFrame. **dropna()** returns a new DataFrame without missing data. Let's see examples of how this works.

```
# let's see our dataframe with missing data
my_DF = pd.DataFrame({'age': [15,17,np.nan,29,25], \
            'state_of_origin':['Lagos', 'Cross River', 'Kano',
            'Abia', np.nan]})
my_DF
'Output':
    age state_of_origin
0  15.0           Lagos
1  17.0     Cross River
2   NaN            Kano
3  29.0            Abia
4  25.0             NaN
```