Working with Missing Data in Pandas

In Pandas, missing values are represented by **None or NaN**, which can occur due to uncollected data or incomplete entries. Let's explore how to detect, handle, and fill in missing values in a DataFrame to ensure accurate analysis.

Table of Content

Checking for Missing Values in Pandas DataFrame

To identify and handle the missing values, Pandas provides two useful functions: **isnull()** and **notnull()**. These functions help detect whether a value is **NaN** or not, making it easier to clean and preprocess data in a DataFrame or Series.

1. Checking for Missing Values Using isnull()

isnull() returns a DataFrame of Boolean values, where **True** represents missing data (**NaN**). This is useful when you want to locate and address missing data within a dataset.

Example 1: Detecting Missing Values in a DataFrame

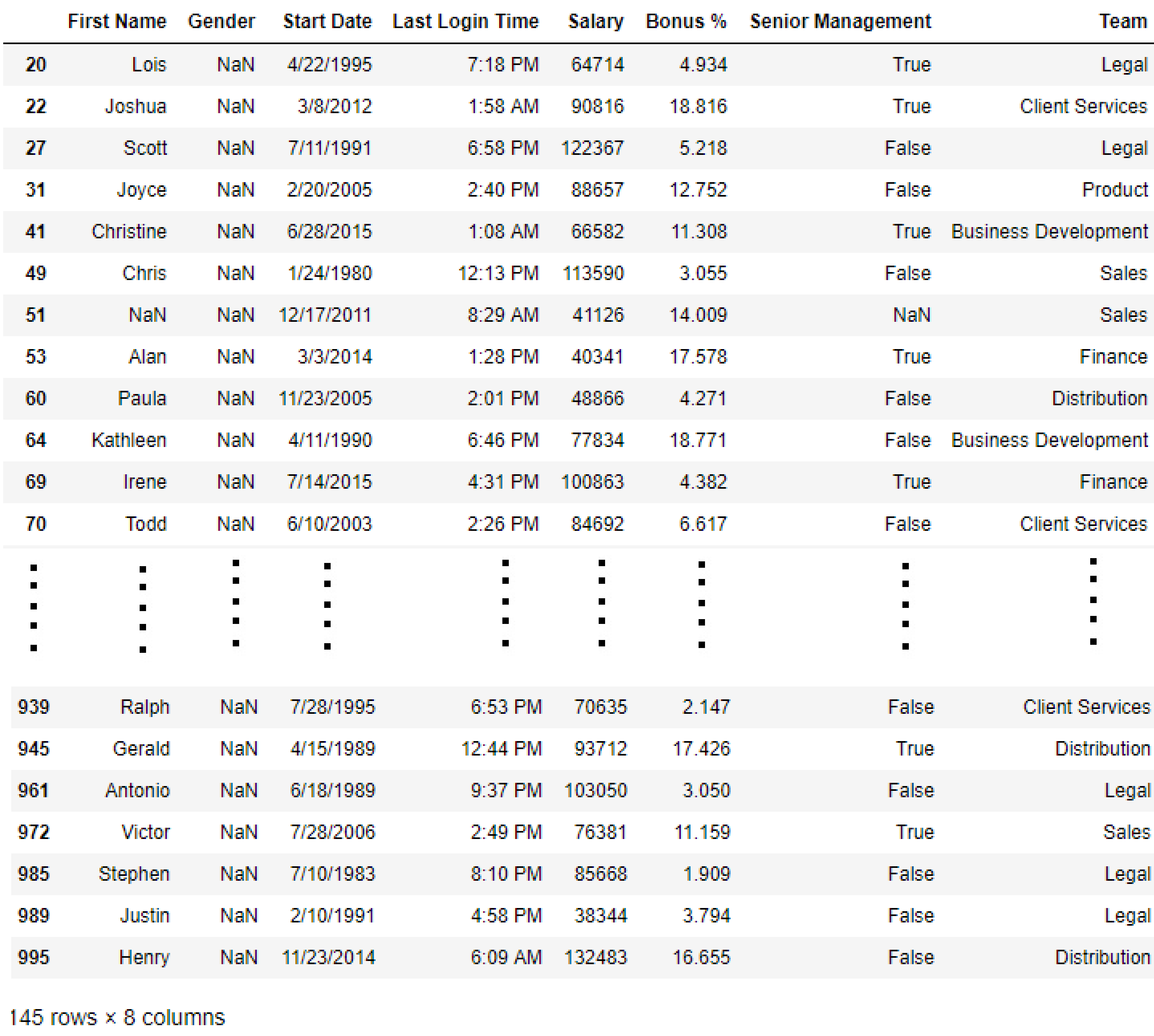
```
# Importing pandas and numpy

import pandas as pd

import numpy as np

# Sample DataFrame with missing values

data = {'First Score': [100, 90, np.nan, 95],

        'Second Score': [30, 45, 56, np.nan],

        'Third Score': [np.nan, 40, 80, 98]}

df = pd.DataFrame(data)

# Checking for missing values using isnull()

missing_values = df.isnull()

print(missing_values)
```

Example 2: Filtering Data Based on Missing Values

In this case, the **isnull()** function is applied to the “Gender” column to filter and display rows with missing gender information.

```
import pandas as pd

data = pd.read_csv("employees.csv")
bool_series = pd.isnull(data["Gender"])
missing_gender_data = data[bool_series]
print(missing_gender_data)
```

Output:



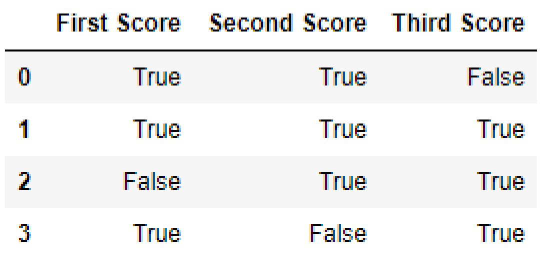
```
     First Name  Gender  Start Date  Last Login Time  Salary  Bonus %  Senior Management                  Team
20         Lois     NaN   4/22/1995          7:18 PM   64714    4.934               True                 Legal
22       Joshua     NaN    3/8/2012          1:58 AM   90816   18.816               True       Client Services
27        Scott     NaN   7/11/1991          6:58 PM  122367    5.218              False                 Legal
31        Joyce     NaN   2/20/2005          2:40 PM   88657   12.752              False               Product
41    Christine     NaN   6/28/2015          1:08 AM   66582   11.308               True  Business Development
49        Chris     NaN   1/24/1980         12:13 PM  113590    3.055              False                 Sales
51          NaN     NaN  12/17/2011          8:29 AM   41126   14.009                NaN                 Sales
53         Alan     NaN    3/3/2014          1:28 PM   40341   17.578               True               Finance
60        Paula     NaN  11/23/2005          2:01 PM   48866    4.271              False          Distribution
64     Kathleen     NaN   4/11/1990          6:46 PM   77834   18.771              False  Business Development
69        Irene     NaN   7/14/2015          4:31 PM  100863    4.382               True               Finance
70         Todd     NaN   6/10/2003          2:26 PM   84692    6.617              False       Client Services
 ⋮           ⋮       ⋮          ⋮                ⋮       ⋮        ⋮                  ⋮                     ⋮
939       Ralph     NaN   7/28/1995          6:53 PM   70635    2.147              False       Client Services
945      Gerald     NaN   4/15/1989         12:44 PM   93712   17.426               True          Distribution
961     Antonio     NaN   6/18/1989          9:37 PM  103050    3.050              False                 Legal
972      Victor     NaN   7/28/2006          2:49 PM   76381   11.159               True                 Sales
985     Stephen     NaN   7/10/1983          8:10 PM   85668    1.909              False                 Legal
989      Justin     NaN   2/10/1991          4:58 PM   38344    3.794              False                 Legal
995       Henry     NaN  11/23/2014          6:09 AM  132483   16.655              False          Distribution

145 rows × 8 columns
```

Checking for Missing Values Using notnull()

notnull() returns a DataFrame of Boolean values, where True indicates non-missing data. This This function can be useful when you want to focus on the rows that contain valid, non-missing data.

Example 3: Detecting Non-Missing Values in a DataFrame

```
# Importing pandas and numpy

import pandas as pd

import numpy as np

# Sample DataFrame with missing values

data = {'First Score': [100, 90, np.nan, 95],

        'Second Score': [30, 45, 56, np.nan],

        'Third Score': [np.nan, 40, 80, 98]}

df = pd.DataFrame(data)

# Checking for non-missing values using notnull()

non_missing_values = df.notnull()

print(non_missing_values)
```



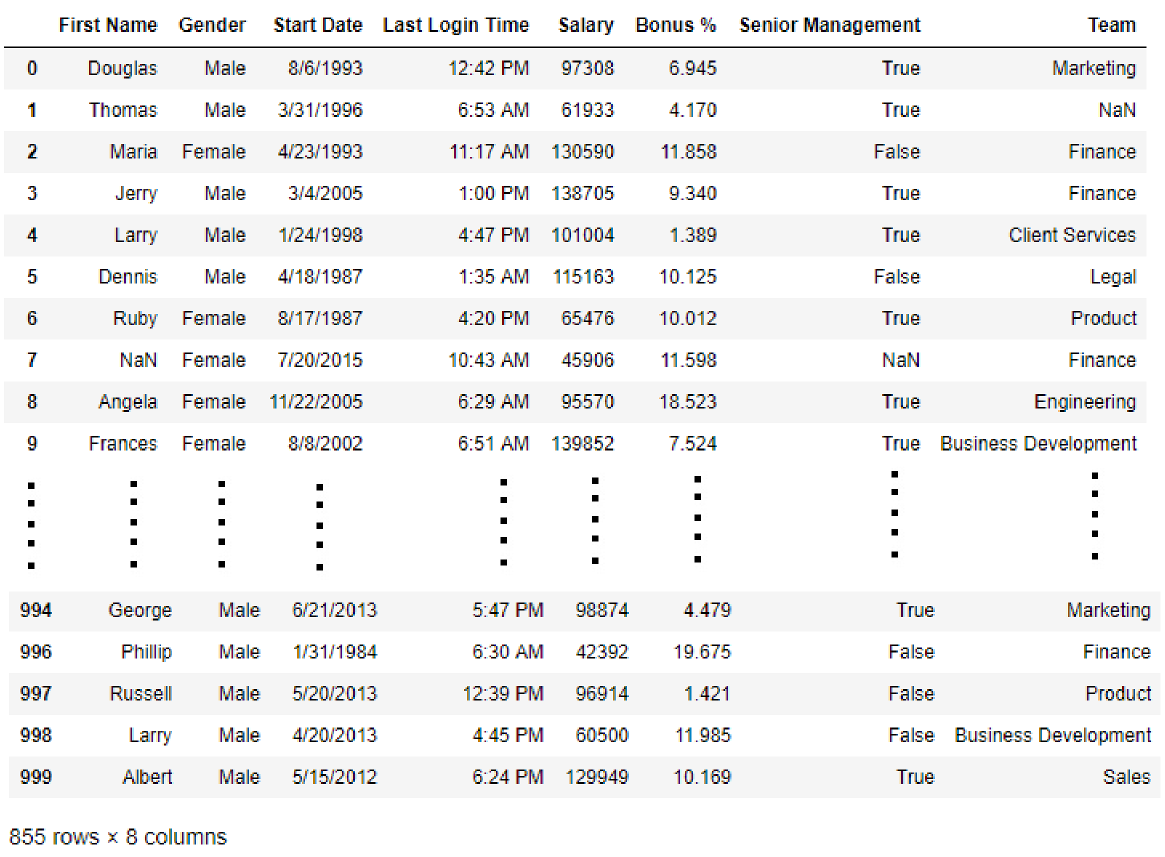
	First Score	Second Score	Third Score
0	True	True	False
1	True	True	True
2	False	True	True
3	True	False	True

Example 4: Filtering Data with Non-Missing Values

This code snippet uses the **notnull()** function to filter out rows where the "Gender" column does not have missing values.

```
# Importing pandas

import pandas as pd

# Reading data from a CSV file

data = pd.read_csv("employees.csv")

# Identifying non-missing values in the 'Gender' column

non_missing_gender = pd.notnull(data["Gender"])
```

```
# Filtering rows where 'Gender' is not missing

non_missing_gender_data = data[non_missing_gender]

display(non_missing_gender_data)
```



	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services
5	Dennis	Male	4/18/1987	1:35 AM	115163	10.125	False	Legal
6	Ruby	Female	8/17/1987	4:20 PM	65476	10.012	True	Product
7	NaN	Female	7/20/2015	10:43 AM	45906	11.598	NaN	Finance
8	Angela	Female	11/22/2005	6:29 AM	95570	18.523	True	Engineering
9	Frances	Female	8/8/2002	6:51 AM	139852	7.524	True	Business Development
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
994	George	Male	6/21/2013	5:47 PM	98874	4.479	True	Marketing
996	Phillip	Male	1/31/1984	6:30 AM	42392	19.675	False	Finance
997	Russell	Male	5/20/2013	12:39 PM	96914	1.421	False	Product
998	Larry	Male	4/20/2013	4:45 PM	60500	11.985	False	Business Development
999	Albert	Male	5/15/2012	6:24 PM	129949	10.169	True	Sales

855 rows × 8 columns

Filling Missing Values in Pandas Using fillna(), replace(), and interpolate()

When working with missing data in Pandas, the **fillna()**, **replace()**, and **interpolate()** functions are commonly used to fill NaN values. These functions allow you to replace missing values with a specific value or use interpolation techniques.

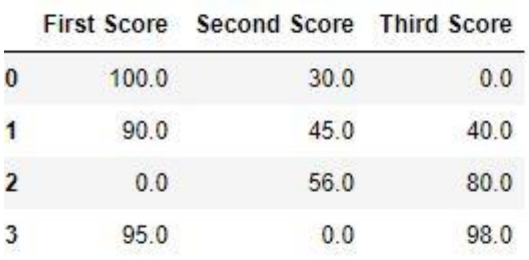
1. Filling Missing Values with a Specific Value Using fillna()

The **fillna()** function is used to replace missing values (**NaN**) with a specified value. For example, you can fill missing values with 0.

Example: Fill Missing Values with Zero

```
import pandas as pd

import numpy as np

dict = {'First Score': [100, 90, np.nan, 95],

    'Second Score': [30, 45, 56, np.nan],

    'Third Score': [np.nan, 40, 80, 98]}
```

```
df = pd.DataFrame(dict)

# Filling missing values with 0

df.fillna(0)
```



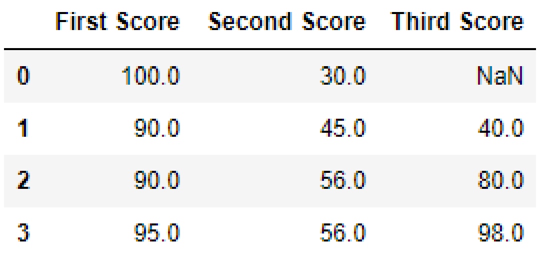
	First Score	Second Score	Third Score
0	100.0	30.0	0.0
1	90.0	45.0	40.0
2	0.0	56.0	80.0
3	95.0	0.0	98.0

2. Filling Missing Values with the Prev/Next Value Using fillna

You can use the **pad** method to fill missing values with the previous value, or **bfill** to fill with the next value. We will be using the above dataset for the demonstration.

Example: Fill with Previous Value (Forward Fill)

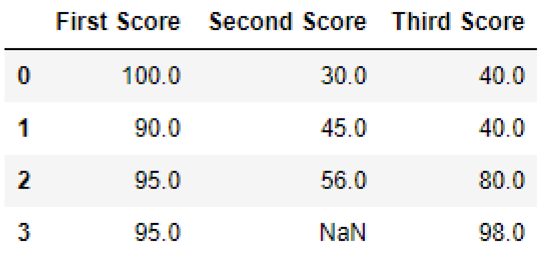
```
df.fillna(method='pad')   # Forward fill
```



	First Score	Second Score	Third Score
0	100.0	30.0	NaN
1	90.0	45.0	40.0
2	90.0	56.0	80.0
3	95.0	56.0	98.0

Example: Fill with Next Value (Backward Fill)

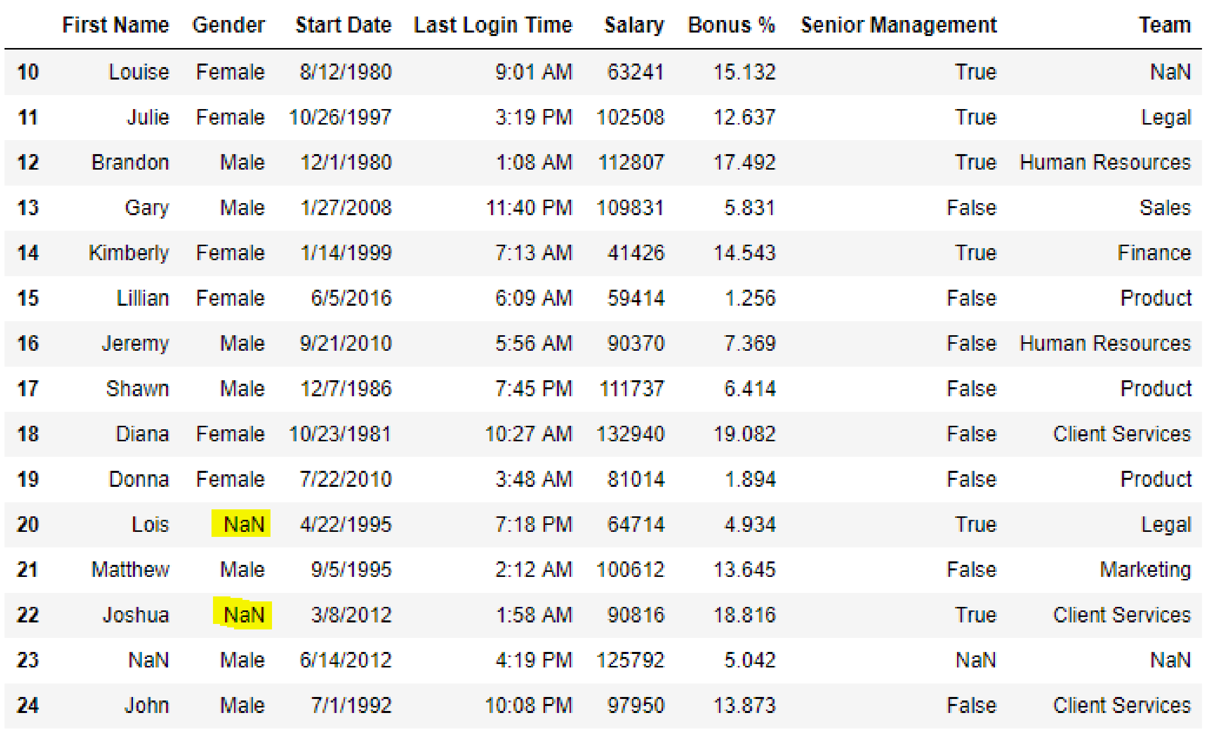
```
df.fillna(method='bfill')   # Backward fill
```



	First Score	Second Score	Third Score
0	100.0	30.0	40.0
1	90.0	45.0	40.0
2	95.0	56.0	80.0
3	95.0	NaN	98.0

Example: Fill NaN Values with ‘No Gender’ using fillna()

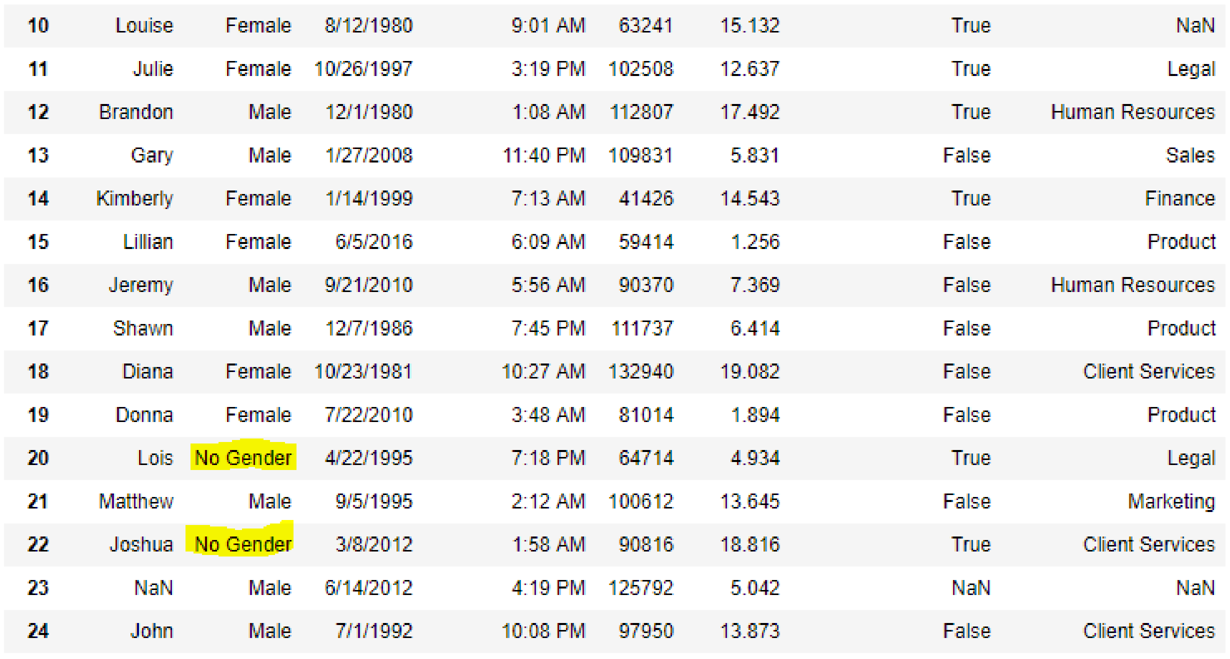
```
import pandas as pd

import numpy as np

data = pd.read_csv("employees.csv")

data[10:25]          # Print records from 10th row to 24th row
```



	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
10	Louise	Female	8/12/1980	9:01 AM	63241	15.132	True	NaN
11	Julie	Female	10/26/1997	3:19 PM	102508	12.637	True	Legal
12	Brandon	Male	12/1/1980	1:08 AM	112807	17.492	True	Human Resources
13	Gary	Male	1/27/2008	11:40 PM	109831	5.831	False	Sales
14	Kimberly	Female	1/14/1999	7:13 AM	41426	14.543	True	Finance
15	Lillian	Female	6/5/2016	6:09 AM	59414	1.256	False	Product
16	Jeremy	Male	9/21/2010	5:56 AM	90370	7.369	False	Human Resources
17	Shawn	Male	12/7/1986	7:45 PM	111737	6.414	False	Product
18	Diana	Female	10/23/1981	10:27 AM	132940	19.082	False	Client Services
19	Donna	Female	7/22/2010	3:48 AM	81014	1.894	False	Product
20	Lois	NaN	4/22/1995	7:18 PM	64714	4.934	True	Legal
21	Matthew	Male	9/5/1995	2:12 AM	100612	13.645	False	Marketing
22	Joshua	NaN	3/8/2012	1:58 AM	90816	18.816	True	Client Services
23	NaN	Male	6/14/2012	4:19 PM	125792	5.042	NaN	NaN
24	John	Male	7/1/1992	10:08 PM	97950	13.873	False	Client Services

Now we are going to fill all the null values in Gender column with “No Gender”

```
 # filling a null values using fillna()
data["Gender"].fillna('No Gender', inplace = True)
data[10:25]
```



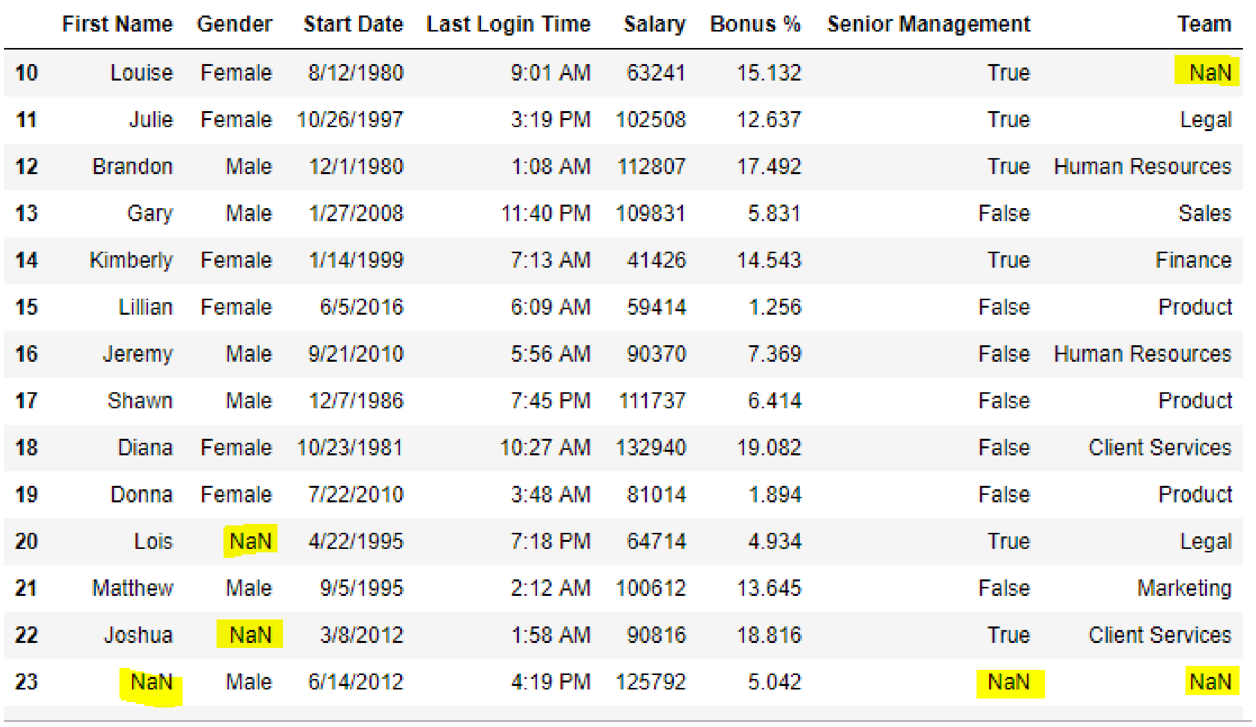
10	Louise	Female	8/12/1980	9:01 AM	63241	15.132	True	NaN
11	Julie	Female	10/26/1997	3:19 PM	102508	12.637	True	Legal
12	Brandon	Male	12/1/1980	1:08 AM	112807	17.492	True	Human Resources
13	Gary	Male	1/27/2008	11:40 PM	109831	5.831	False	Sales
14	Kimberly	Female	1/14/1999	7:13 AM	41426	14.543	True	Finance
15	Lillian	Female	6/5/2016	6:09 AM	59414	1.256	False	Product
16	Jeremy	Male	9/21/2010	5:56 AM	90370	7.369	False	Human Resources
17	Shawn	Male	12/7/1986	7:45 PM	111737	6.414	False	Product
18	Diana	Female	10/23/1981	10:27 AM	132940	19.082	False	Client Services
19	Donna	Female	7/22/2010	3:48 AM	81014	1.894	False	Product
20	Lois	No Gender	4/22/1995	7:18 PM	64714	4.934	True	Legal
21	Matthew	Male	9/5/1995	2:12 AM	100612	13.645	False	Marketing
22	Joshua	No Gender	3/8/2012	1:58 AM	90816	18.816	True	Client Services
23	NaN	Male	6/14/2012	4:19 PM	125792	5.042	NaN	NaN
24	John	Male	7/1/1992	10:08 PM	97950	13.873	False	Client Services

3. Replacing Missing Values Using replace()

Use **replace()** to replace **NaN** values with a specific value like **-99**.

Example: Replace NaN with -99

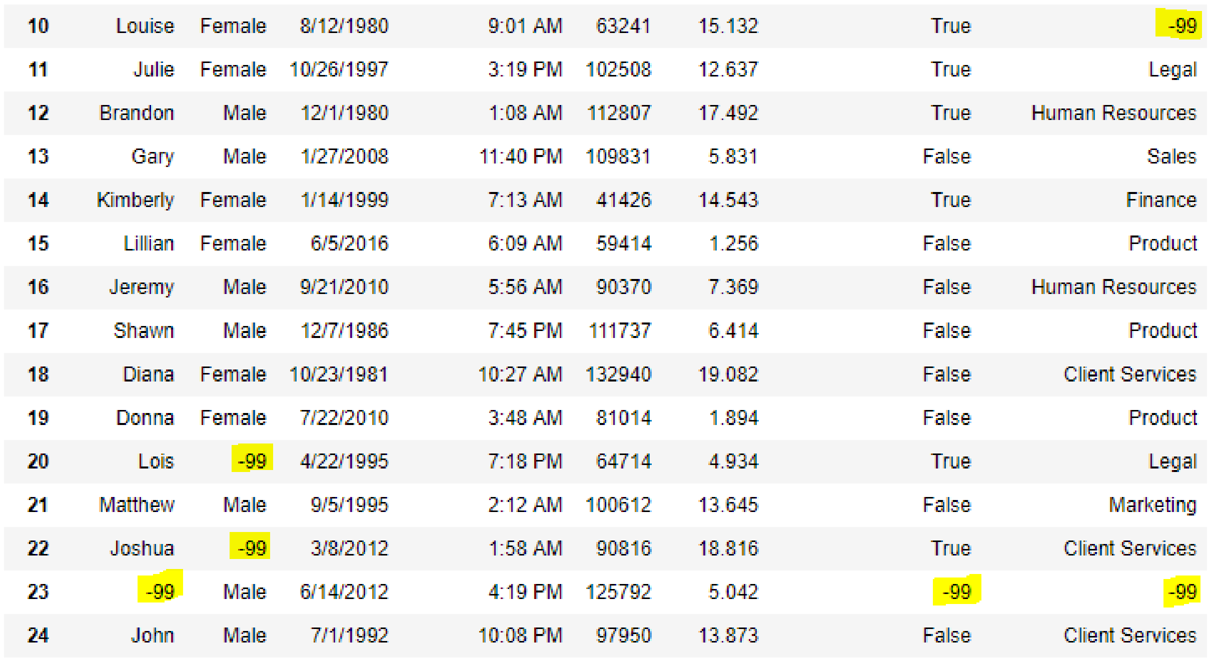
```
import pandas as pd
import numpy as np

data = pd.read_csv("employees.csv")
data[10:25]
```



	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
10	Louise	Female	8/12/1980	9:01 AM	63241	15.132	True	NaN
11	Julie	Female	10/26/1997	3:19 PM	102508	12.637	True	Legal
12	Brandon	Male	12/1/1980	1:08 AM	112807	17.492	True	Human Resources
13	Gary	Male	1/27/2008	11:40 PM	109831	5.831	False	Sales
14	Kimberly	Female	1/14/1999	7:13 AM	41426	14.543	True	Finance
15	Lillian	Female	6/5/2016	6:09 AM	59414	1.256	False	Product
16	Jeremy	Male	9/21/2010	5:56 AM	90370	7.369	False	Human Resources
17	Shawn	Male	12/7/1986	7:45 PM	111737	6.414	False	Product
18	Diana	Female	10/23/1981	10:27 AM	132940	19.082	False	Client Services
19	Donna	Female	7/22/2010	3:48 AM	81014	1.894	False	Product
20	Lois	NaN	4/22/1995	7:18 PM	64714	4.934	True	Legal
21	Matthew	Male	9/5/1995	2:12 AM	100612	13.645	False	Marketing
22	Joshua	NaN	3/8/2012	1:58 AM	90816	18.816	True	Client Services
23	NaN	Male	6/14/2012	4:19 PM	125792	5.042	NaN	NaN

Now, we are going to replace the all Nan value in the data frame with -99 value.

```
data.replace(to_replace=np.nan, value=-99)
```



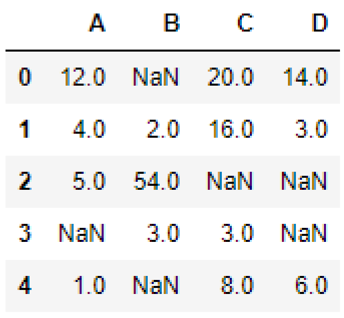
10	Louise	Female	8/12/1980	9:01 AM	63241	15.132	True	-99
11	Julie	Female	10/26/1997	3:19 PM	102508	12.637	True	Legal
12	Brandon	Male	12/1/1980	1:08 AM	112807	17.492	True	Human Resources
13	Gary	Male	1/27/2008	11:40 PM	109831	5.831	False	Sales
14	Kimberly	Female	1/14/1999	7:13 AM	41426	14.543	True	Finance
15	Lillian	Female	6/5/2016	6:09 AM	59414	1.256	False	Product
16	Jeremy	Male	9/21/2010	5:56 AM	90370	7.369	False	Human Resources
17	Shawn	Male	12/7/1986	7:45 PM	111737	6.414	False	Product
18	Diana	Female	10/23/1981	10:27 AM	132940	19.082	False	Client Services
19	Donna	Female	7/22/2010	3:48 AM	81014	1.894	False	Product
20	Lois	-99	4/22/1995	7:18 PM	64714	4.934	True	Legal
21	Matthew	Male	9/5/1995	2:12 AM	100612	13.645	False	Marketing
22	Joshua	-99	3/8/2012	1:58 AM	90816	18.816	True	Client Services
23	-99	Male	6/14/2012	4:19 PM	125792	5.042	-99	-99
24	John	Male	7/1/1992	10:08 PM	97950	13.873	False	Client Services

Filling Missing Values Using interpolate()

The **interpolate()** function fills missing values using interpolation techniques, such as the linear method. This method fills missing values by treating the data as equally spaced.

Example: Linear Interpolation

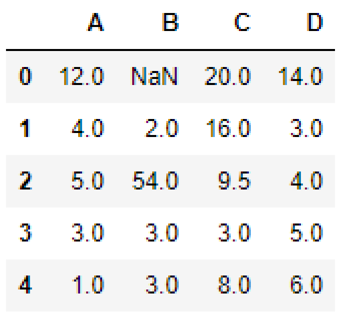
```
# importing pandas as pd
import pandas as pd

# Creating the dataframe
df = pd.DataFrame({"A": [12, 4, 5, None, 1],
                   "B": [None, 2, 54, 3, None],
                   "C": [20, 16, None, 3, 8],
                   "D": [14, 3, None, None, 6]})

# Print the dataframe
print(df)
```



	A	B	C	D
0	12.0	NaN	20.0	14.0
1	4.0	2.0	16.0	3.0
2	5.0	54.0	NaN	NaN
3	NaN	3.0	3.0	NaN
4	1.0	NaN	8.0	6.0

Let's interpolate the missing values using Linear method. Note that Linear method ignore the index and treat the values as equally spaced.

```
# to interpolate the missing values
df.interpolate(method ='linear', limit_direction ='forward')
```



	A	B	C	D
0	12.0	NaN	20.0	14.0
1	4.0	2.0	16.0	3.0
2	5.0	54.0	9.5	4.0
3	3.0	3.0	3.0	5.0
4	1.0	3.0	8.0	6.0

Dropping Missing Values in Pandas Using dropna()

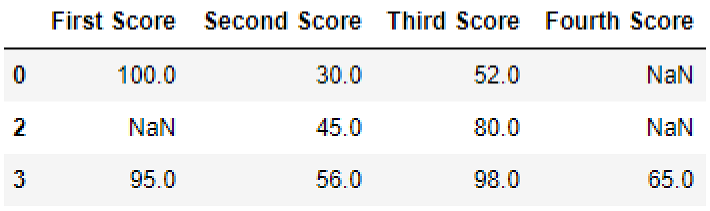
The **dropna()**function in Pandas removes rows or columns with **NaN** values. It can be used to drop data based on different conditions.

1. Dropping Rows with At Least One Null Value

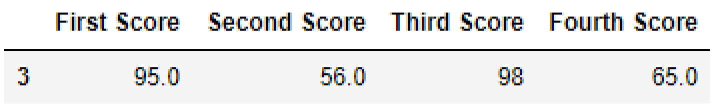
Use **dropna()** to remove rows that contain at least one missing value.

Example: Drop Rows with At Least One NaN

```
import pandas as pd
import numpy as np

dict = {'First Score': [100, 90, np.nan, 95],
        'Second Score': [30, np.nan, 45, 56],
        'Third Score': [52, 40, 80, 98],
        'Fourth Score': [np.nan, np.nan, np.nan, 65]}

df = pd.DataFrame(dict)

# Drop rows with at least one missing value
df.dropna()
```



	First Score	Second Score	Third Score	Fourth Score
0	100.0	30.0	52.0	NaN
2	NaN	45.0	80.0	NaN
3	95.0	56.0	98.0	65.0



	First Score	Second Score	Third Score	Fourth Score
3	95.0	56.0	98	65.0

Dropping Rows with All Null Values

You can drop rows where all values are missing using **dropna(how='all')**.

Example: Drop Rows with All NaN Values

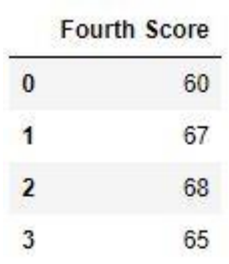
```
dict = {'First Score': [100, np.nan, np.nan, 95],
        'Second Score': [30, np.nan, 45, 56],
        'Third Score': [52, np.nan, 80, 98],
        'Fourth Score': [np.nan, np.nan, np.nan, 65]}


df = pd.DataFrame(dict)

# Drop rows where all values are missing
df.dropna(how='all')
```

Dropping Columns with At Least One Null Value

To remove columns that contain at least one missing value, use **dropna(axis=1).**

Example: Drop Columns with At Least One NaN

```
dict = {'First Score': [100, np.nan, np.nan, 95],
        'Second Score': [30, np.nan, 45, 56],
        'Third Score': [52, np.nan, 80, 98],
        'Fourth Score': [60, 67, 68, 65]}

df = pd.DataFrame(dict)

# Drop columns with at least one missing value
df.dropna(axis=1)
```



	Fourth Score
0	60
1	67
2	68
3	65

Dropping Rows with Missing Values in CSV Files

When working with data from CSV files, you can drop rows with missing values using **dropna()**.

Example: Drop Rows with NaN in a CSV File

```
import pandas as pd

data = pd.read_csv("employees.csv")

# Drop rows with any missing value
new_data = data.dropna(axis=0, how='any')

# Compare lengths of original and new dataframes
print("Old data frame length:", len(data))
print("New data frame length:", len(new_data))
print("Rows with at least one missing value:", (len(data) - len(new_data)))
```

output:

```
Old data frame length: 1000
New data frame length: 764
Rows with at least one missing value: 236
```

Since the difference is 236, there were 236 rows which had at least 1 Null value in any column.