SIM Global Education

Topic 7
Cleaning and restructuring data

Learning Outcomes

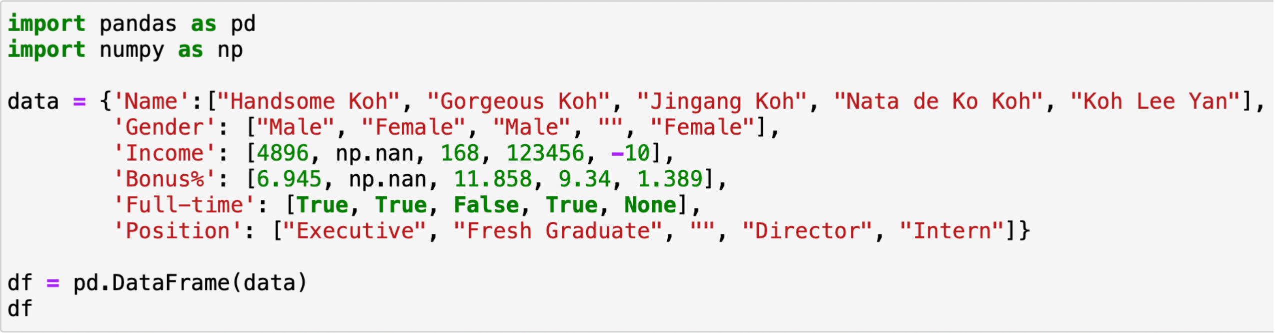
After completing this topic and the recommended reading, you should be able to:

- Explain the problems that can occur in particular data processing scenarios if data has not been properly cleaned.
- Apply data cleaning techniques to cope with missing and corrupted data.
- Use exception handling and data verification techniques to write more robust data processing code.

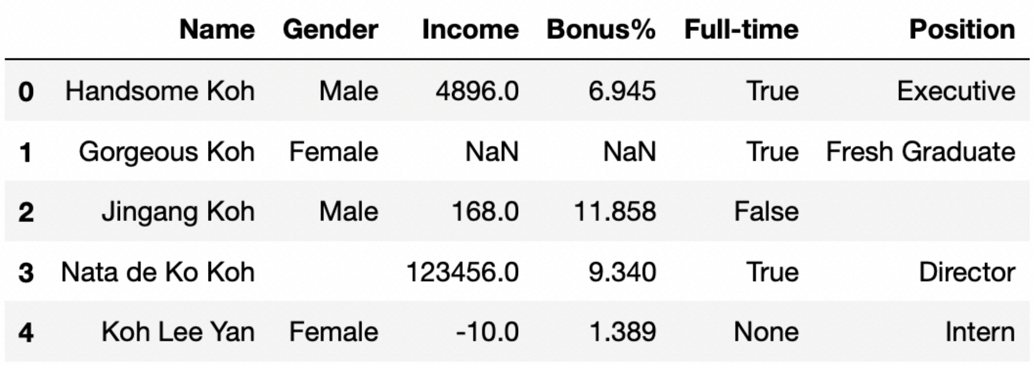
1. Missing Data

Data frame

- Creating a sample data frame, using dictionary.

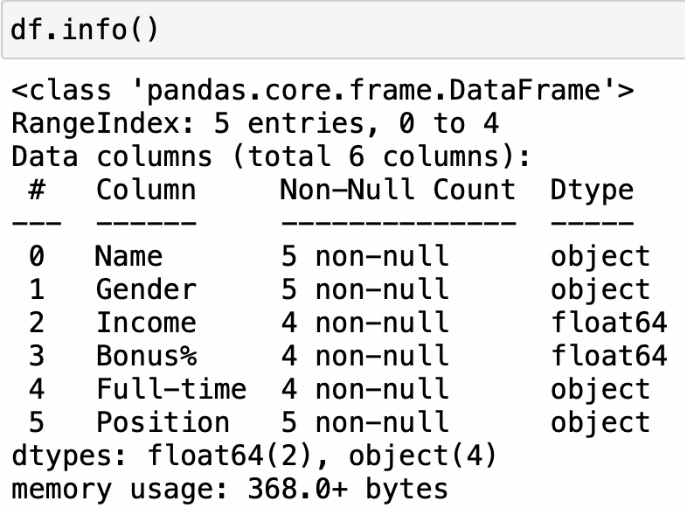


```
import pandas as pd
import numpy as np

data = {'Name':["Handsome Koh", "Gorgeous Koh", "Jingang Koh", "Nata de Ko Koh", "Koh Lee Yan"],
        'Gender': ["Male", "Female", "Male", "", "Female"],
        'Income': [4896, np.nan, 168, 123456, -10],
        'Bonus%': [6.945, np.nan, 11.858, 9.34, 1.389],
        'Full-time': [True, True, False, True, None],
        'Position': ["Executive", "Fresh Graduate", "", "Director", "Intern"]}

df = pd.DataFrame(data)
df
```



	Name	Gender	Income	Bonus%	Full-time	Position
0	Handsome Koh	Male	4896.0	6.945	True	Executive
1	Gorgeous Koh	Female	NaN	NaN	True	Fresh Graduate
2	Jingang Koh	Male	168.0	11.858	False	
3	Nata de Ko Koh		123456.0	9.340	True	Director
4	Koh Lee Yan	Female	-10.0	1.389	None	Intern

- Printing information about the data frame

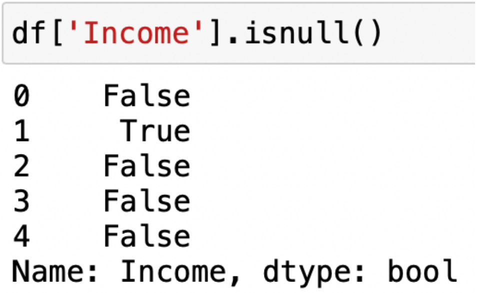


```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Name       5 non-null      object
 1   Gender     5 non-null      object
 2   Income     4 non-null      float64
 3   Bonus%     4 non-null      float64
 4   Full-time  4 non-null      object
 5   Position   5 non-null      object
dtypes: float64(2), object(4)
memory usage: 368.0+ bytes
```

Marking Missing Values

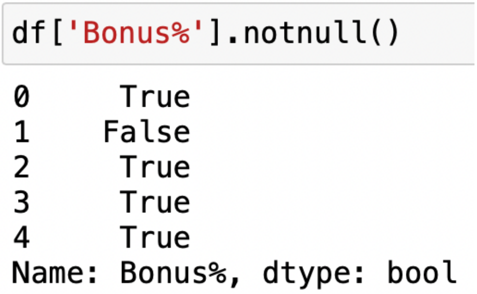
- *isnull()*
  - mark all *NaN* values in the dataset as *True*



```
df['Income'].isnull()
```

```
0    False
1     True
2    False
3    False
4    False
Name: Income, dtype: bool
```

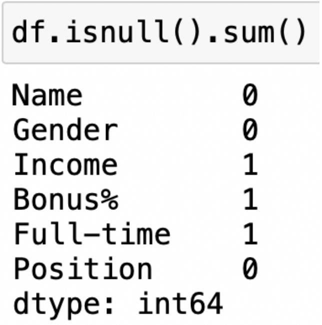
- *notnull()*
  - mark all *NaN* values in the dataset as *False*



```
df['Bonus%'].notnull()
```

```
0     True
1    False
2     True
3     True
4     True
Name: Bonus%, dtype: bool
```

- Total number of missing values per column



```
df.isnull().sum()
```

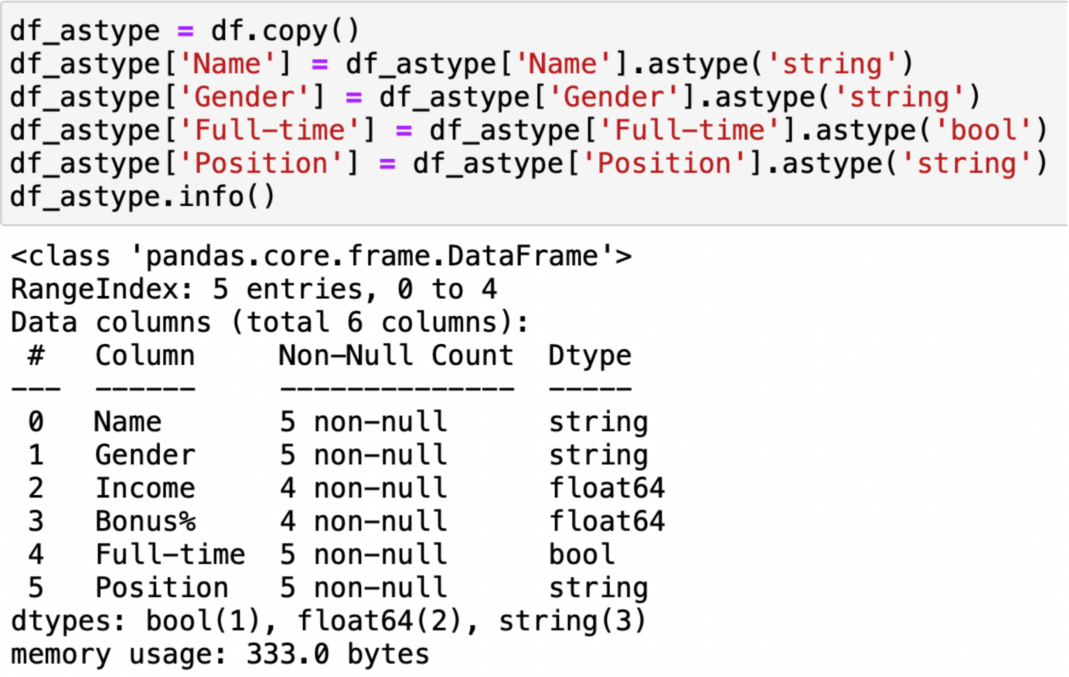
```
Name         0
Gender       0
Income       1
Bonus%       1
Full-time    1
Position     0
dtype: int64
```

- Visible errors:
  - Blank cells
  - NA (Not Available)
  - NaN (Not a Number)
  - None (Null value)
- Obscure errors:
  - Non-corrupt but invalid values

- E.g. negative income

Handling Invalid Data Types

- *Pandas dataframe.astype()*



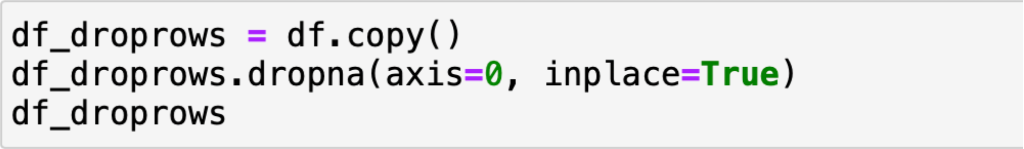
```
df_astype = df.copy()
df_astype['Name'] = df_astype['Name'].astype('string')
df_astype['Gender'] = df_astype['Gender'].astype('string')
df_astype['Full-time'] = df_astype['Full-time'].astype('bool')
df_astype['Position'] = df_astype['Position'].astype('string')
df_astype.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Name       5 non-null      string 
 1   Gender     5 non-null      string 
 2   Income     4 non-null      float64
 3   Bonus%     4 non-null      float64
 4   Full-time  5 non-null      bool   
 5   Position   5 non-null      string 
dtypes: bool(1), float64(2), string(3)
memory usage: 333.0 bytes
```

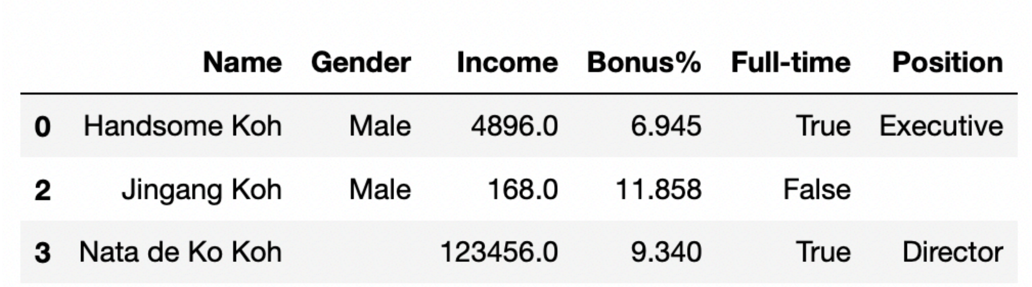
2. Removing Missing Values

Pandas dropna()

- Remove all rows that contain missing values
- *axis = 0* (default)

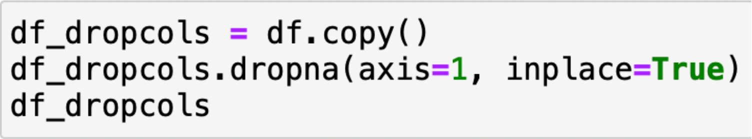


```
df_droprows = df.copy()
df_droprows.dropna(axis=0, inplace=True)
df_droprows
```

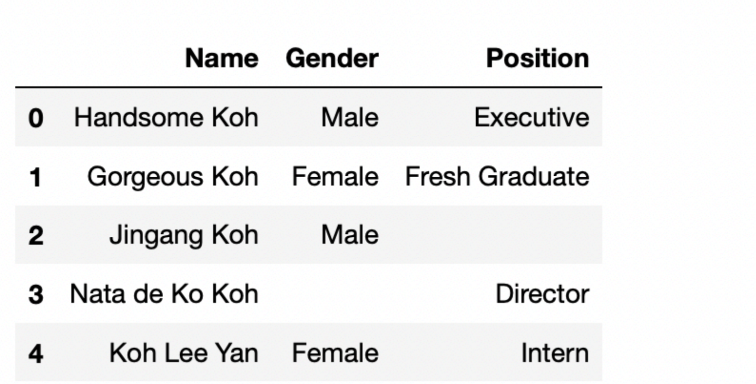


	Name	Gender	Income	Bonus%	Full-time	Position
0	Handsome Koh	Male	4896.0	6.945	True	Executive
2	Jingang Koh	Male	168.0	11.858	False	
3	Nata de Ko Koh		123456.0	9.340	True	Director

- Remove all columns that contain missing values



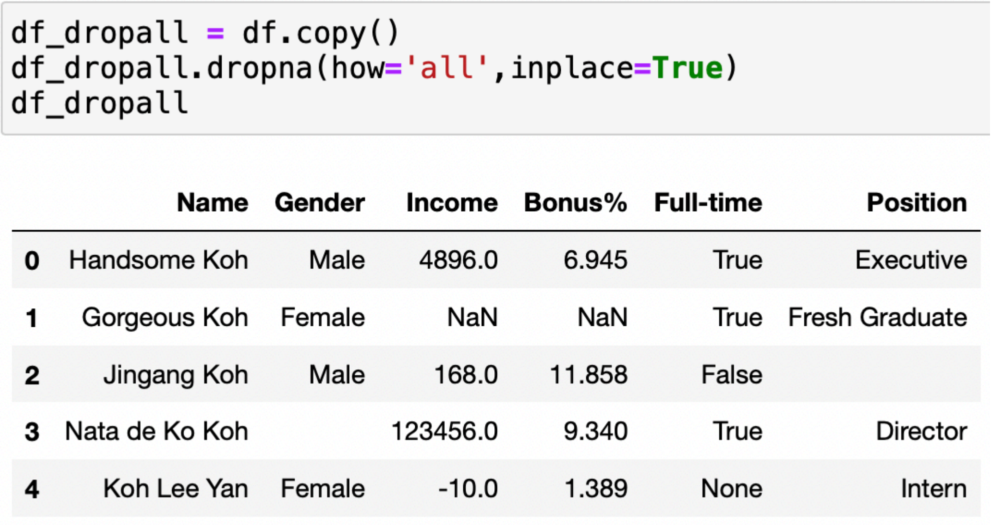
```
df_dropcols = df.copy()
df_dropcols.dropna(axis=1, inplace=True)
df_dropcols
```



	Name	Gender	Position
0	Handsome Koh	Male	Executive
1	Gorgeous Koh	Female	Fresh Graduate
2	Jingang Koh	Male	
3	Nata de Ko Koh		Director
4	Koh Lee Yan	Female	Intern

- *inplace = True*
    - causes all changes to happen in the same data frame instead of returning a new one
- *how='any'* (default)
    - at least one value must be null

- *how='all'*
    - all values must be null



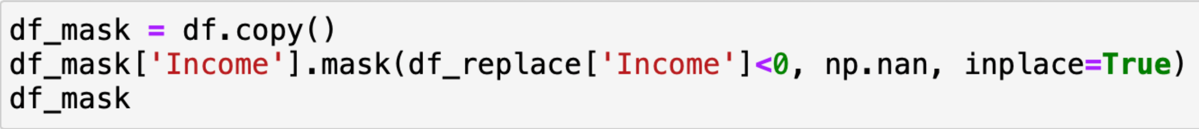
```
df_dropall = df.copy()
df_dropall.dropna(how='all',inplace=True)
df_dropall
```

	Name	Gender	Income	Bonus%	Full-time	Position
0	Handsome Koh	Male	4896.0	6.945	True	Executive
1	Gorgeous Koh	Female	NaN	NaN	True	Fresh Graduate
2	Jingang Koh	Male	168.0	11.858	False	
3	Nata de Ko Koh		123456.0	9.340	True	Director
4	Koh Lee Yan	Female	-10.0	1.389	None	Intern

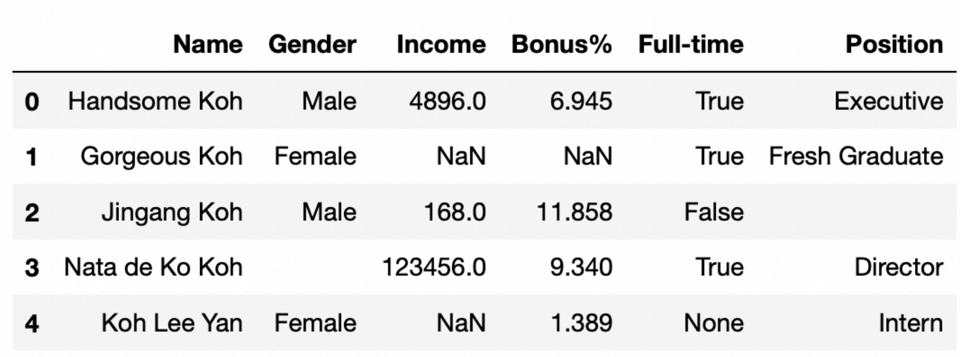
3. Imputing Missing Values

Pandas dataframe.mask()

- It replaces the values of the rows where the condition evaluates to *True*.



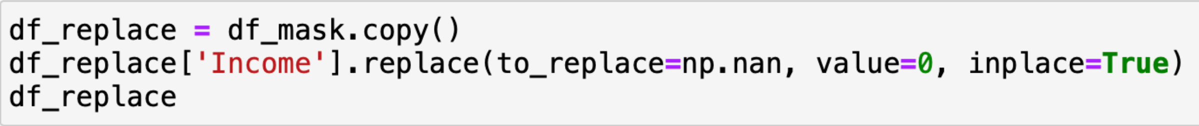
```
df_mask = df.copy()
df_mask['Income'].mask(df_replace['Income']<0, np.nan, inplace=True)
df_mask
```



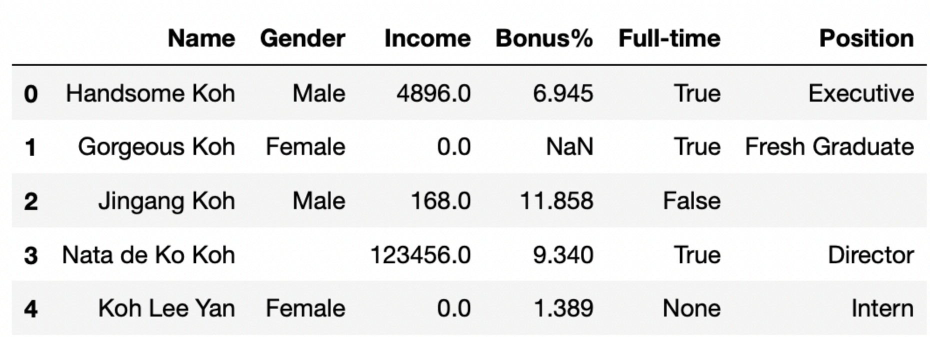
	Name	Gender	Income	Bonus%	Full-time	Position
0	Handsome Koh	Male	4896.0	6.945	True	Executive
1	Gorgeous Koh	Female	NaN	NaN	True	Fresh Graduate
2	Jingang Koh	Male	168.0	11.858	False	
3	Nata de Ko Koh		123456.0	9.340	True	Director
4	Koh Lee Yan	Female	NaN	1.389	None	Intern

Pandas dataframe.replace()

- It is used to replace values in the data frame



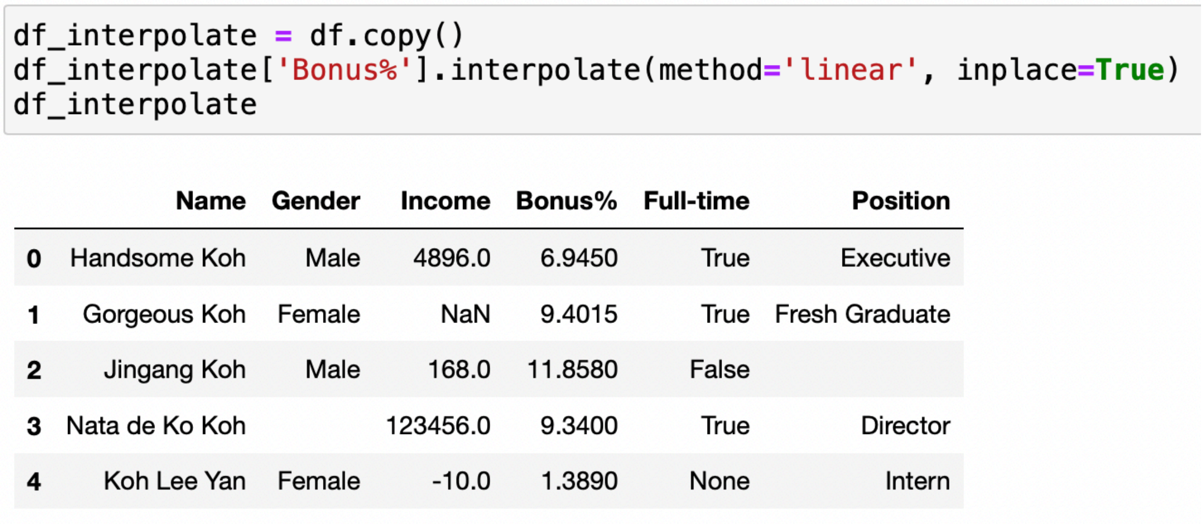
```
df_replace = df_mask.copy()
df_replace['Income'].replace(to_replace=np.nan, value=0, inplace=True)
df_replace
```



	Name	Gender	Income	Bonus%	Full-time	Position
0	Handsome Koh	Male	4896.0	6.945	True	Executive
1	Gorgeous Koh	Female	0.0	NaN	True	Fresh Graduate
2	Jingang Koh	Male	168.0	11.858	False	
3	Nata de Ko Koh		123456.0	9.340	True	Director
4	Koh Lee Yan	Female	0.0	1.389	None	Intern

Pandas dataframe.interpolate()

- It is used to fill NA or NaN values in the dataframe or series
- Using various interpolation techniques



```python
df_interpolate = df.copy()
df_interpolate['Bonus%'].interpolate(method='linear', inplace=True)
df_interpolate
```

	Name	Gender	Income	Bonus%	Full-time	Position
0	Handsome Koh	Male	4896.0	6.9450	True	Executive
1	Gorgeous Koh	Female	NaN	9.4015	True	Fresh Graduate
2	Jingang Koh	Male	168.0	11.8580	False	
3	Nata de Ko Koh		123456.0	9.3400	True	Director
4	Koh Lee Yan	Female	-10.0	1.3890	None	Intern

4. Exercises

7.17 Cleaning Data

- Refers to “7.17 cleaningData.html”

5. Practice Quiz

- Work on *Practice Quiz 07* posted on Canvas.

Useful Resources

-
    - http://