NaN Equivalence Comparison Misused

Context

NaN equivalence comparison behaves differently from None equivalence comparison.

Problem

While **None == None evaluates to `True`, `np.nan` == `np.nan` evaluates to `False`** in NumPy. As Pandas treats None like `np.nan` for simplicity and performance reasons, a comparison of `DataFrame` elements with `np.nan` always returns `False`. Therefore, given **a python conditional statement (e.g., ==, !=, is)**, if **one of the two extremes of the condition is np.nan, this condition can lead to unintentional bugs in the code.**

Solution

Developers **should avoid to compare objects with np.nan and use df.isna()**.

Existing Stage	Effect
Data Cleaning	Error-prone

Example

```Python
### Pandas & NumPy
import pandas as pd
- import numpy as np

df = pd.DataFrame([1, None, 3])
- df_eq_nan = df == np.nan
- df_is_nan = df is np.nan
+ df_eq_nan = df.isna()
+ df_is_nan = df.isna()
```