**Data Cleaning**

* Columns that are not required or redundant can be dropped.
* Renaming of some columns after we have dropped certain columns.
* Nan Analysis and doing something about NaNs
  + Check if NaNs are present and how many
  + Drop some columns or row that have a large number of NaNs
  + Fill with some values so that it doesn’t affect the analysis.
* Type conversions (upcasting, down casting)

**Titles.info()**

* gives the overview of the quality of the data that we have.
* Tells how many non nulls values are present in series or columns

Try and see if you can reduce the memory usage.

Title.Unique()

* Gives you the unique values.

Df.rename()

To rename a column

Df.col.astype(type, errors = ‘ignore’)

If you want to ignore the errors

Find out missing data

Df.isna()

Checks for NA values

**Missing Data:**

* + Either drop it completely
  + Fill with some values.

**Df.dropna(subset = [‘’])**

* + - Drops na values
    - Subset lets you select the cols

**Df.fillna({col: val, col:val})**

Fills na values

**Downcasting to save memory:**

**df.memory\_usage()**

* let’s you check the memory usage per column

| **Operation** | **Code Snippet** |
| --- | --- |
| Create | df = pd.read\_\*(file, sep='\t') |
| Info | df.info() |
| Dtypes | df.dtypes |
|  | df.col.dtype |
| Drop | df.drop([cols], axis='columns') |
| Rename | df.rename(dict, axis='columns' |
| Unique | df.col.unique() |
| NaN | df.isna() |
|  | df.col.isna() |
|  | df.dropna() |
|  | df.fillna() |
| Type-cast | df.col = df.col.astype(type) |
| Memory Usage | df.memory\_usage() |
|  | df.col.memory\_usage() |
| In-place | df.operation(args, inplace=True) |

**Types and Memory Usage**

Type Memory (bytes) Range of values

bool 1 True, False

int8 1 -128 to 127

int16 2 -32,768 to 32,767

int32 4 -2,147,483,648 to 2,147,483,647

int64 8 -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

float16 2 6.5e+/-04

float32 4 3.4e+/-38

float64 8 1.7e+/-308