Open in app

Published in Towards Data Science

Sharone Li
Mar 3 · 4 min read ★ · Listen

Clean a ‘Numeric’ ID Column in Pandas Dataframe

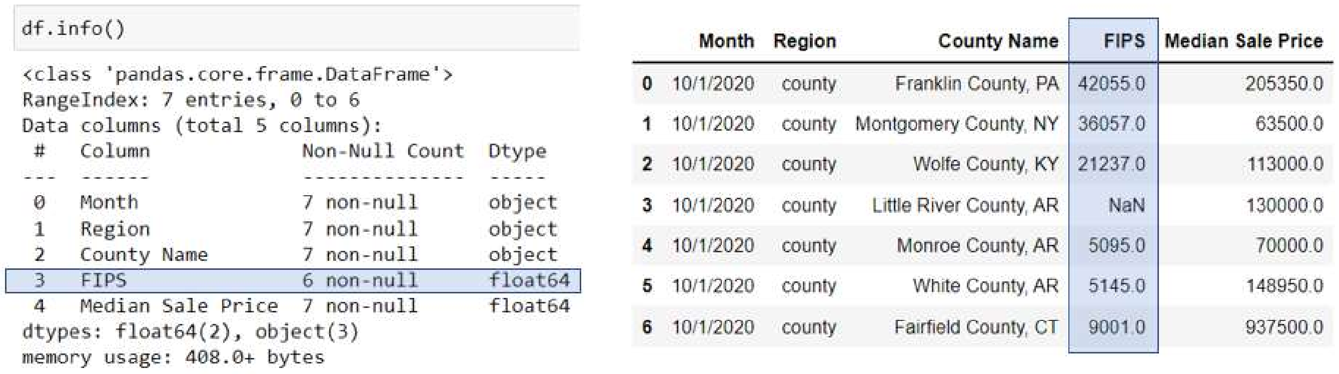
A Handy Piece of Code for Pandas Beginners

Image by Pixabay

As a data scientist, you must have encountered this problem at least once in your data science journey: you import your data into a Pandas dataframe and the ID column is

This is such a common problem and almost always needs to be treated before you carry on your data munging and analysis tasks. This is especially important if you plan to use the ID field as a key to join with other tables later on. So how to fix it in Python?

Let's look at a simple example. We have a sample dataframe that shows the median home sales price for each county in October 2020. In this dataframe, we have an ID field — FIPS which is a unique 5-digit geographic identifier for each county in the U.S.



```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Month              7 non-null      object
 1   Region             7 non-null      object
 2   County Name        7 non-null      object
 3   FIPS               6 non-null      float64
 4   Median Sale Price  7 non-null      float64
dtypes: float64(2), object(3)
memory usage: 408.0+ bytes
```

	Month	Region	County Name	FIPS	Median Sale Price
0	10/1/2020	county	Franklin County, PA	42055.0	205350.0
1	10/1/2020	county	Montgomery County, NY	36057.0	63500.0
2	10/1/2020	county	Wolfe County, KY	21237.0	113000.0
3	10/1/2020	county	Little River County, AR	NaN	130000.0
4	10/1/2020	county	Monroe County, AR	5095.0	70000.0
5	10/1/2020	county	White County, AR	5145.0	148950.0
6	10/1/2020	county	Fairfield County, CT	9001.0	937500.0

Image by Author

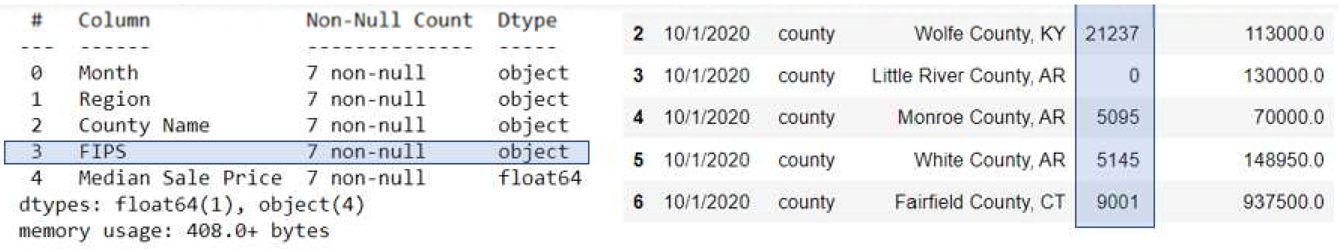
The FIPS field is supposed to be a five-digit code and should be imported as a string; instead, it is shown as a float type which doesn't make sense. We will need to change it to a string type and also add back the leading zeros. We can do it in two ways:

Method 1:

Using this method, we will first change the FIPS field from `float` type to `integer` type, and then change it to `string` . We can use Pandas' `DataFrame.astype()` method to change the data types.

```
df['FIPS'] = df['FIPS'].fillna(0).astype(int).astype(str)
```

Open in app



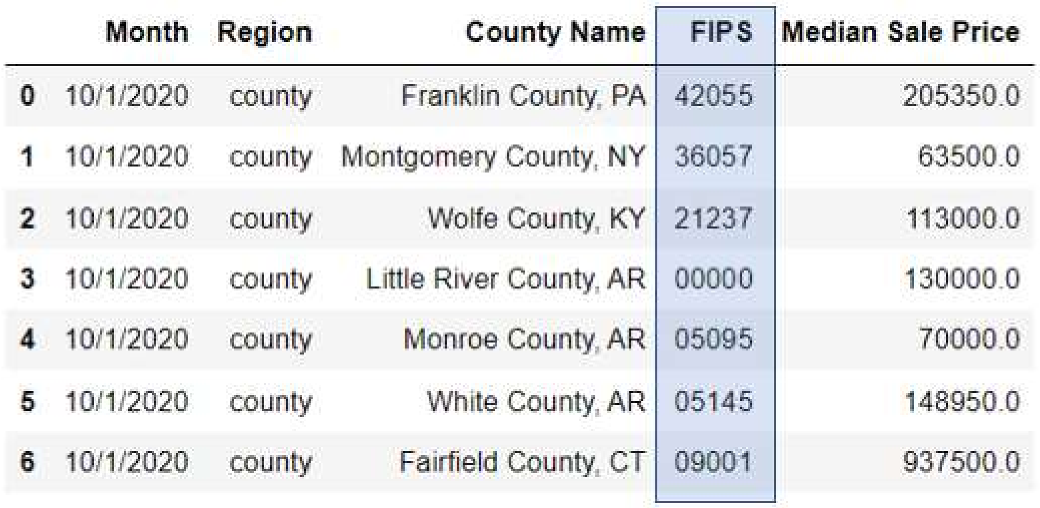
```
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Month              7 non-null      object 
 1   Region             7 non-null      object 
 2   County Name        7 non-null      object 
 3   FIPS               7 non-null      object 
 4   Median Sale Price  7 non-null      float64
dtypes: float64(1), object(4)
memory usage: 408.0+ bytes
```

2	10/1/2020	county	Wolfe County, KY	21237	113000.0
3	10/1/2020	county	Little River County, AR	0	130000.0
4	10/1/2020	county	Monroe County, AR	5095	70000.0
5	10/1/2020	county	White County, AR	5145	148950.0
6	10/1/2020	county	Fairfield County, CT	9001	937500.0

Image by Author

After the FIPS field is changed to the `string` type, we can add back the leading zeros to make it a five-digit code. We can use Pandas’ `zfill()` method to do it.

```
df['FIPS'] = df['FIPS'].str.zfill(5)
```



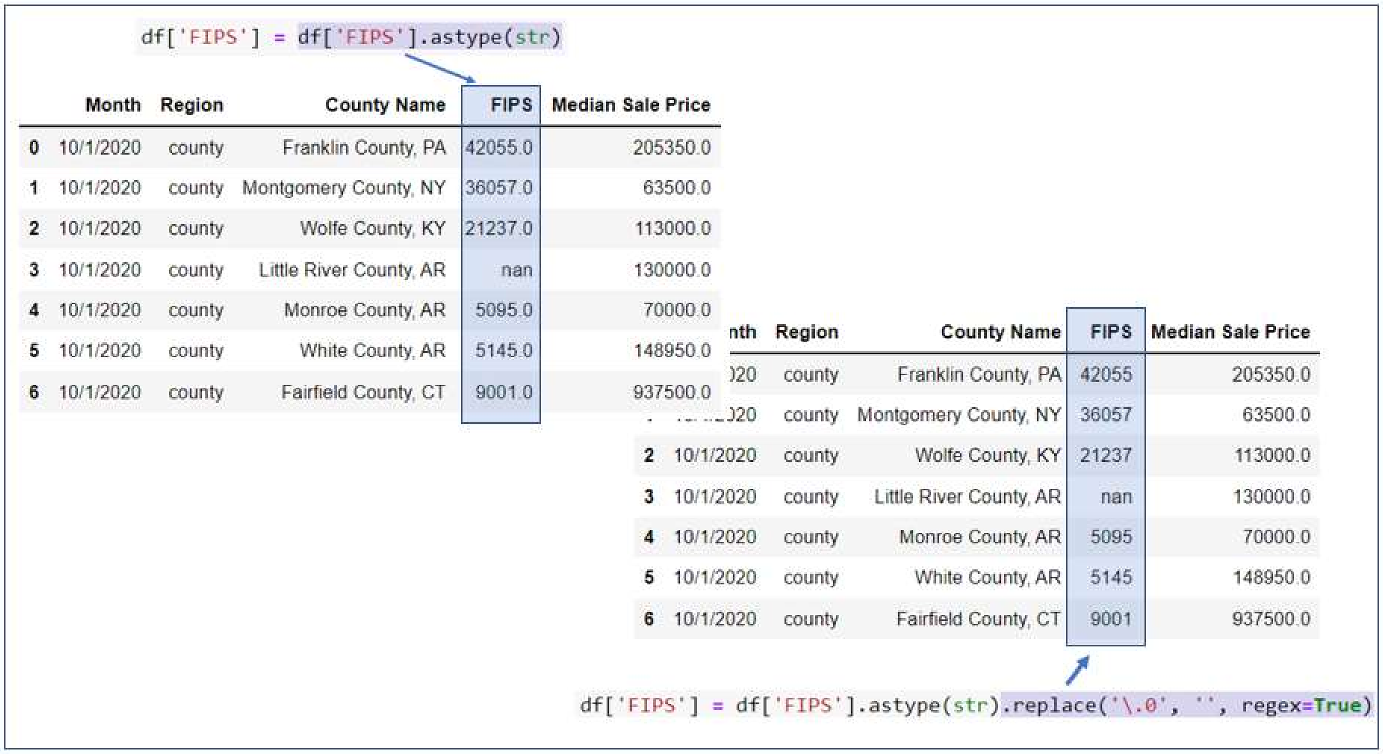
	Month	Region	County Name	FIPS	Median Sale Price
0	10/1/2020	county	Franklin County, PA	42055	205350.0
1	10/1/2020	county	Montgomery County, NY	36057	63500.0
2	10/1/2020	county	Wolfe County, KY	21237	113000.0
3	10/1/2020	county	Little River County, AR	00000	130000.0
4	10/1/2020	county	Monroe County, AR	05095	70000.0
5	10/1/2020	county	White County, AR	05145	148950.0
6	10/1/2020	county	Fairfield County, CT	09001	937500.0

Image by Author

Method 2:

Another way to clean the FIPS field is to first change its data type to `string`, and then use `regex` (regular expressions) in Python to search and replace certain patterns in the string in order to remove the decimal places. The following code uses `regex` to remove the ‘.0’ part from the string in the FIPS column.

Open in app



```
df['FIPS'] = df['FIPS'].astype(str)
```

	Month	Region	County Name	FIPS	Median Sale Price
0	10/1/2020	county	Franklin County, PA	42055.0	205350.0
1	10/1/2020	county	Montgomery County, NY	36057.0	63500.0
2	10/1/2020	county	Wolfe County, KY	21237.0	113000.0
3	10/1/2020	county	Little River County, AR	nan	130000.0
4	10/1/2020	county	Monroe County, AR	5095.0	70000.0
5	10/1/2020	county	White County, AR	5145.0	148950.0
6	10/1/2020	county	Fairfield County, CT	9001.0	937500.0

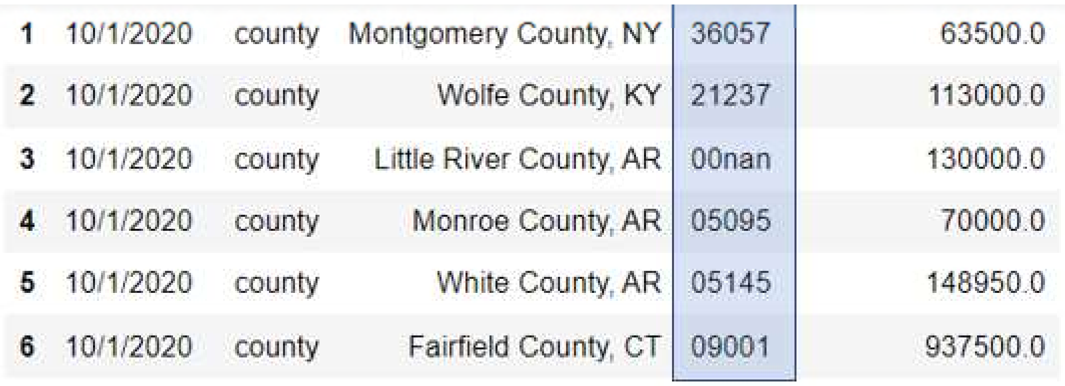
	nth	Region	County Name	FIPS	Median Sale Price
	)20	county	Franklin County, PA	42055	205350.0
	)20	county	Montgomery County, NY	36057	63500.0
2	10/1/2020	county	Wolfe County, KY	21237	113000.0
3	10/1/2020	county	Little River County, AR	nan	130000.0
4	10/1/2020	county	Monroe County, AR	5095	70000.0
5	10/1/2020	county	White County, AR	5145	148950.0
6	10/1/2020	county	Fairfield County, CT	9001	937500.0

```
df['FIPS'] = df['FIPS'].astype(str).replace('\.0', '', regex=True)
```

Image by Author

We then use `zfill()` to add back the leading zeros to the FIPS field and make it a five-digit code.

```
df['FIPS'] = df['FIPS'].str.zfill(5)
```



1	10/1/2020	county	Montgomery County, NY	36057	63500.0
2	10/1/2020	county	Wolfe County, KY	21237	113000.0
3	10/1/2020	county	Little River County, AR	00nan	130000.0
4	10/1/2020	county	Monroe County, AR	05095	70000.0
5	10/1/2020	county	White County, AR	05145	148950.0
6	10/1/2020	county	Fairfield County, CT	09001	937500.0

Image by Author

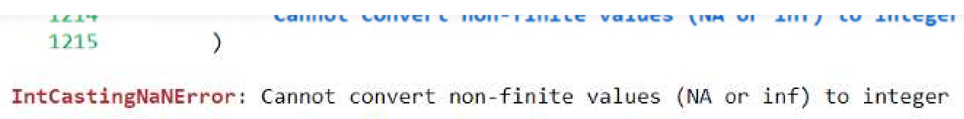
You can also choose to replace all the '00nan' with '00000' or other values using the following code, but it is optional.

```
df['FIPS'] = df['FIPS'].replace('00nan', '00000')
```

One thing I want to point out is that in method 1, please make sure that you use `fillna(0)` to replace all the missing values with 0 in the FIPS column before using `astype(int)` to change the data type. This is because `astype(int)` won't work if your ID column (i.e., FIPS) has missing values.

For example, if you use the following code to try to change FIPS from `float` to `integer` directly, you will get a traceback error shown below:

```
df['FIPS'] = df['FIPS'].astype(int)
```



```
1214            Cannot convert non-finite values (NA or inf) to integer
1215        )

IntCastingNaNError: Cannot convert non-finite values (NA or inf) to integer
```

Image by Author

In summary, you can easily clean a ‘numeric’ ID column with leading zeros by using one of the following two methods. It is a very common and seemingly easy problem to fix but could be tricky to figure out for python beginners. Therefore, I hope you find this short tutorial and the code helpful, especially for those who just started their data science and python journey. Thanks for reading!

Method 1:

```
df['FIPS'] = df['FIPS'].fillna(0).astype(int).astype(str)
df['FIPS'] = df['FIPS'].str.zfill(5)
```

Method 2:

```
df['FIPS'] = df['FIPS'].astype(str).replace('\.0', '', regex=True)
df['FIPS'] = df['FIPS'].str.zfill(5)
```

Data Source:

Redfin Data Center: Redfin Monthly Housing Market Data — County Level. This is an open dataset provided by *Redfin*, a national real estate brokerage, that you can download for free and for your own purposes with citation.

Open in app

membership ($5 per month) through this referral link. By signing up through this link, I will receive a portion of your membership fee at no additional cost to you. Thank you!

Join Medium with my referral link - Sharone Li

As a Medium member, a portion of your membership fee goes to writers you read, and you get full access to every story...

medium.com