the basic data cleaning tasks we’ll tackle:

1. [Importing Libraries](https://monkeylearn.com/blog/data-cleaning-python/#import)
2. [Input Customer Feedback Dataset](https://monkeylearn.com/blog/data-cleaning-python/#input)
3. [Locate Missing Data](https://monkeylearn.com/blog/data-cleaning-python/#missing)
4. [Check for Duplicates](https://monkeylearn.com/blog/data-cleaning-python/#duplicates)
5. [Detect Outliers](https://monkeylearn.com/blog/data-cleaning-python/#outliers)
6. [Normalize Casing](https://monkeylearn.com/blog/data-cleaning-python/#normalize)

[Data analysis](https://monkeylearn.com/data-analysis/), and [text analysis](https://monkeylearn.com/text-analysis/), have come into the crosshairs of every business interested in ongoing success in modern markets. Quite literally, the ability to analyze and quantify [customer feedback](https://monkeylearn.com/customer-feedback/), sales patterns,

1. Importing Libraries

Let’s get Pandas and NumPy up and running on your Python script.

**INPUT:**

*import pandas as pd*

*import numpy as np*

**OUTPUT:**

In this case, your script should now have the libraries loaded. You’ll see if this is true by inputting a dataset in our next step.

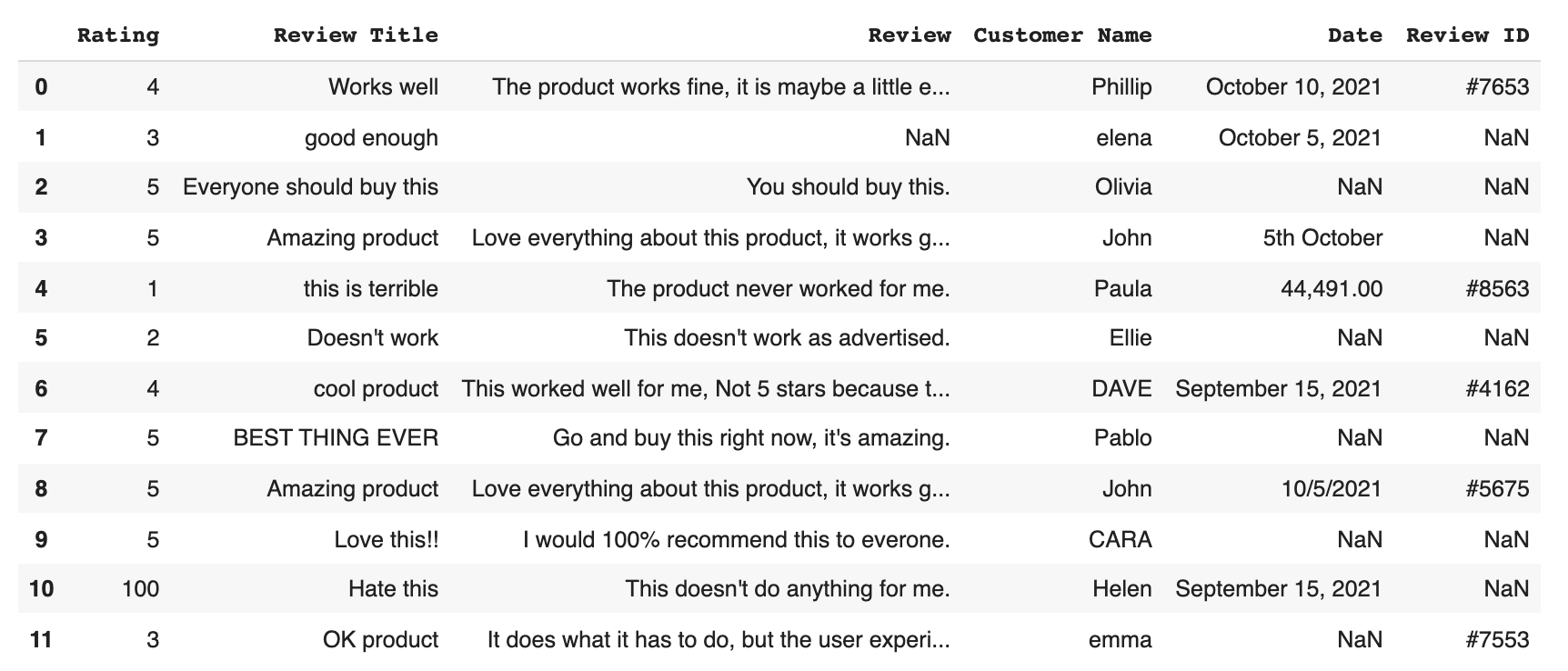
2. Input Customer Feedback Dataset

Next, we ask our libraries to read a feedback dataset. Let’s see what that looks like.

**INPUT:**

*data = pd.read\_csv('feedback.csv')*

**OUTPUT:**



As you can see the “feedback.csv” should be the dataset you want to examine. And, in this case, when we read “pd.read\_csv” as the prior function, we know we are using the Pandas library to read our dataset.

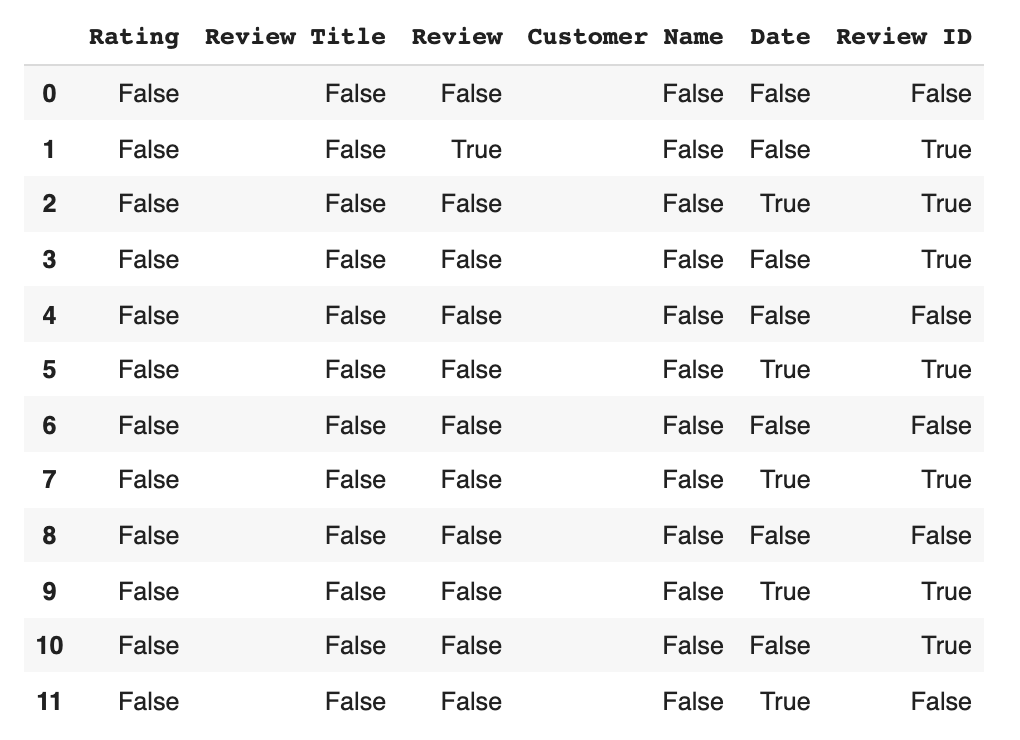
3. Locate Missing Data

Next, we are going to use a secret Python hack known as ‘isnull function’ to discover our data. Actually a common function, 'isnull' helps us find where in our dataset there are missing values. This is useful information as this is what we need to correct while data cleaning.

**INPUT:**

*data.isnull()*

**OUTPUT:**



Our output result is a list of **boolean values**.

There are several insights the list can give us. First and foremost is where the missing data is – any ‘True’ reading under a column indicates missing data in that column’s category for that data file.

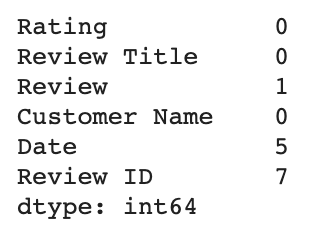
So, for example, datapoint **1** has missing data in its **Review** section and its **Review ID** section (both are marked true).

We can further expand the missing data of each feature by coding:

**INPUT:**

*data.isnull().sum()*

**OUTPUT:**



From here, we use code to actually clean the data. This boils down to two basic options. **1) Drop the data** or, **2) Input missing data**. If you opt to:

1. Drop the data

You’ll have to make another decision – whether to drop only the missing values and keep the data in the set, or to eliminate the feature (the entire column) wholesale because there are so many missing datapoints that it isn’t fit for analysis.

If you want to drop the missing values you’ll have to go in and mark them void according to Pandas or NumBy standards (see section below). But if you want to drop the entire column, here’s the code:

**INPUT:**

*remove = ['Review ID','Date']*

*data.drop(remove, inplace =True, axis =1)*

**OUTPUT:**



Now, let’s examine our other option.

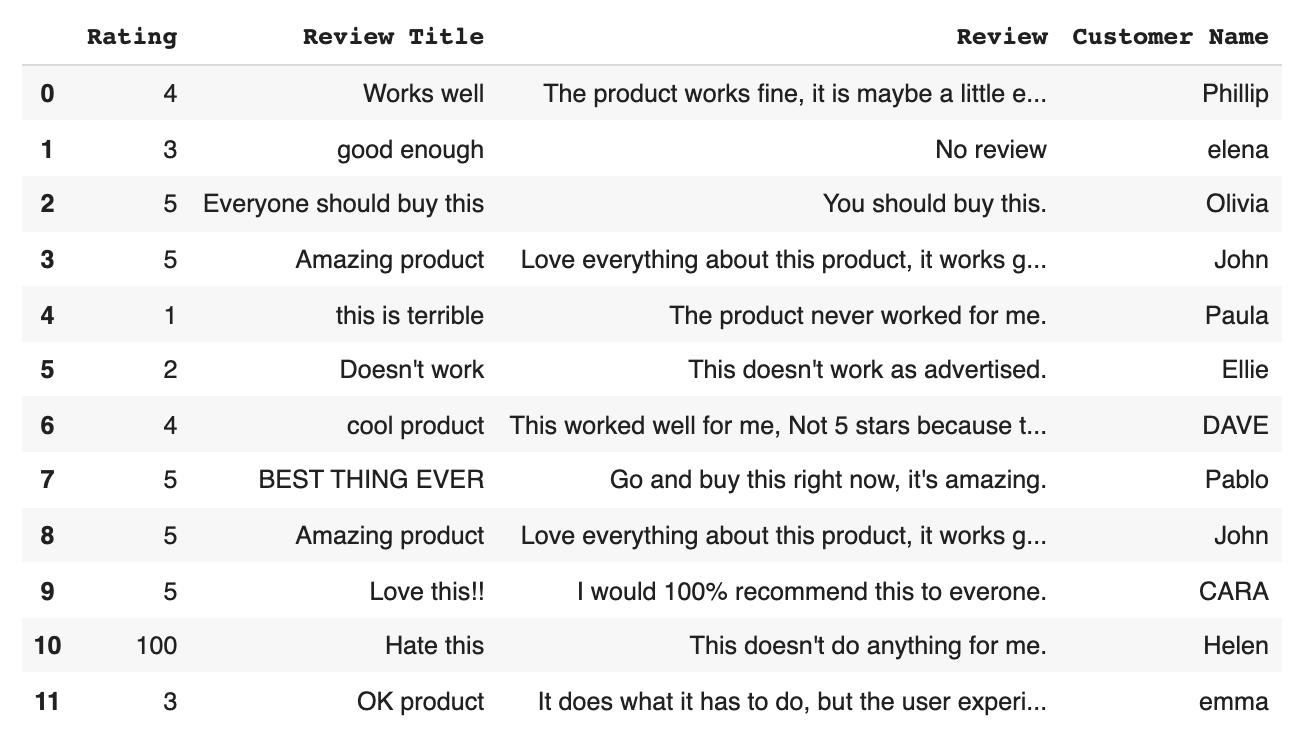
2. Input missing data

Technically, the method described above of filling in individual values with Pandas or NumBy standards is also a form of inputting missing data – we call it adding ‘No Review’. When it comes to inputting missing data you can either add ‘No Review’ using the code below, or manually fill in the correct data.

**INPUT:**

*data['Review'] = data['Review'].fillna('No review')*

**OUTPUT:**



As you can see, now the data point **1** have now been marked as ‘No Review’ – success!

4. Check for Duplicates

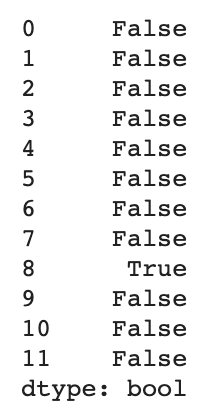
Duplicates, like missing data, cause problems and clog up analytics software. Let’s locate and eliminate them.

To locate duplicates we start out with:

**INPUT:**

*data.duplicated()*

**OUTPUT:**



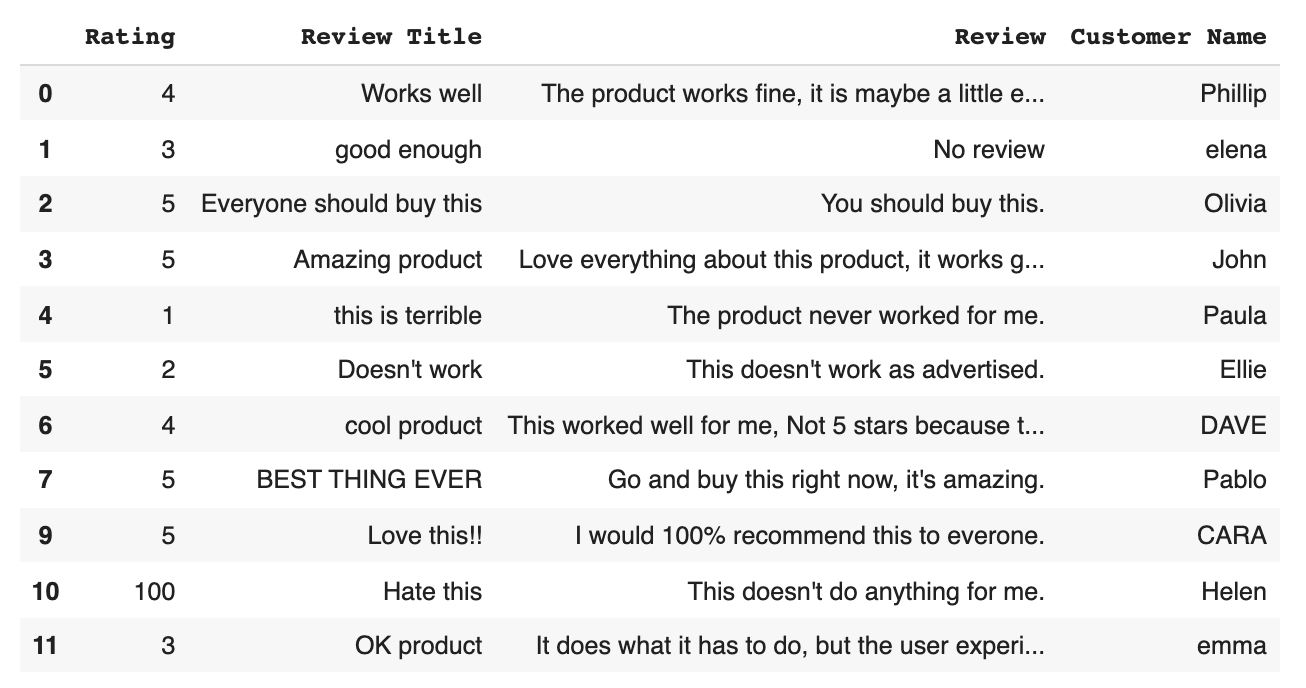
Aka a list of boolean values where a ‘True’ reading indicated duplicate values.

Let’s go and get ahead and get rid of that duplicate (datapoint **8**).

**INPUT:**

*data.drop\_duplicates()*

**OUTPUT:**



And there we have it, our dataset with our duplicate removed. Onwards.

5. Detect Outliers

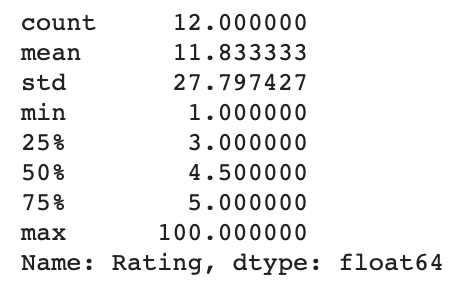
Outliers are numerical values that lie significantly outside of the statistical norm. Cutting that down from unnecessary science garble – they are data points that are so out of range they are likely misreads.

They, like duplicates, need to be removed. Let’s sniff out an outlier by first, pulling up our dataset.

**INPUT:**

*data['Rating'].describe()*

**OUTPUT:**

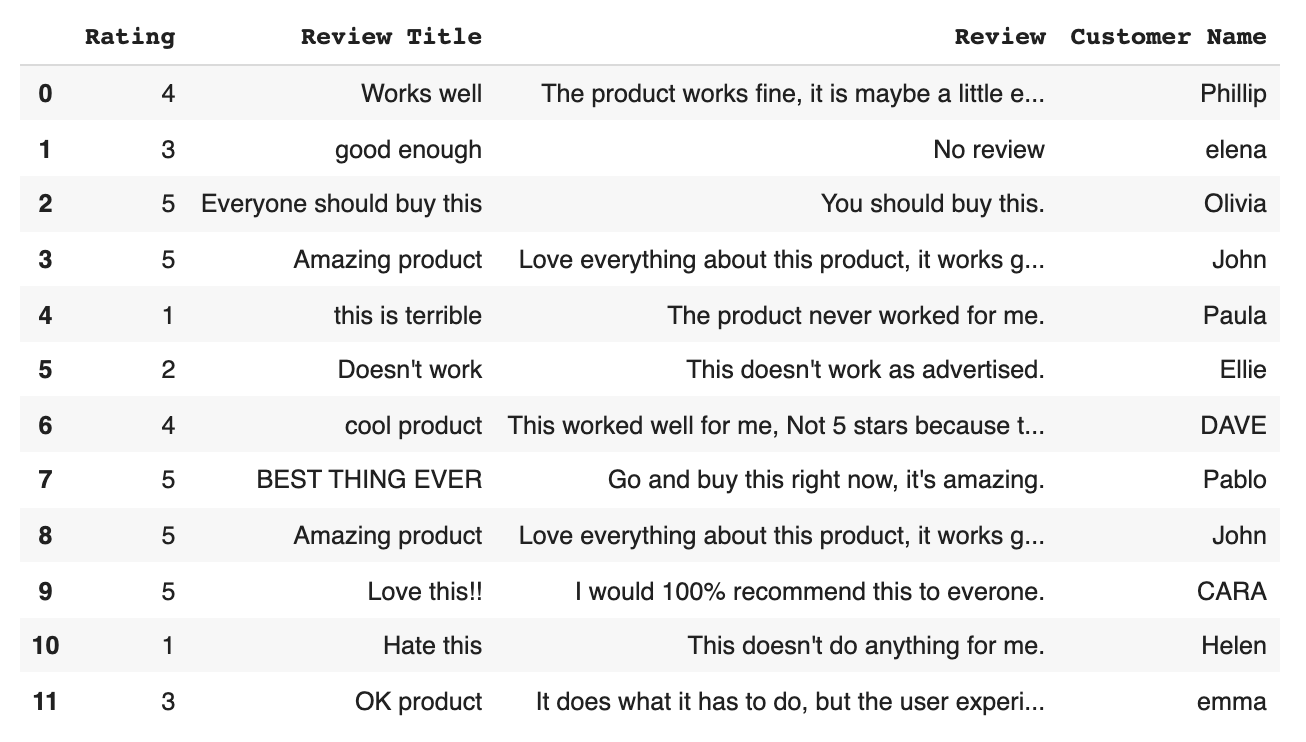


Take a look at that ‘max’ value - none of the other values are even close to 100, with the mean (the average) being 11. Now, your solution to outliers will depend on your knowledge of your dataset. In this case, the data scientists who input the knowledge know that they meant to put a value of 1 not 100. So, we can safely remove the outlier to fix our data.

**INPUT:**

*data.loc[10,'Rating'] = 1*

**OUTPUT:**



Now our dataset has ratings ranging from 1 to 5, which will save major skew from if there was a rogue 100 in there.

6. Normalize Casing

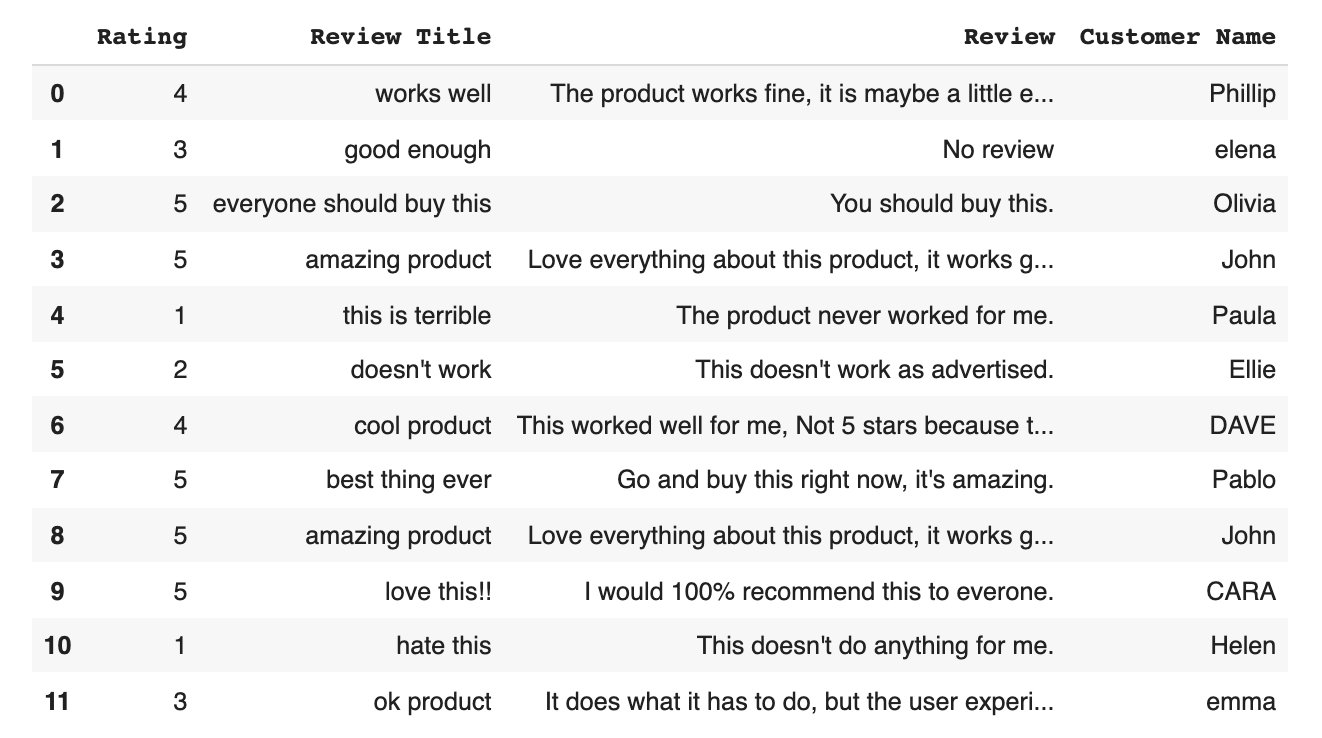
Last but not least we are going to dot our i’s and cross our t’s. Meaning we are going to standardize (lowercase) all review titles so as not to confuse our algorithms, and we are going to capitalize Customer Names, so that our algorithms know they are variables (you’ll see this in action below).

Here’s how to make every review title lowercase:

**INPUT:**

*data['Review Title'] = data['Review Title'].str.lower()*

**OUTPUT:**



Looks great! On to making sure our high-powered programs don’t get tripped up and miscategorize a customer name because it isn’t capitalized. Here’s how to ensure Customer Name capitalization:

**INPUT:**

*data['Customer Name'] = data['Customer Name'].str.title()*

**OUTPUT:**

