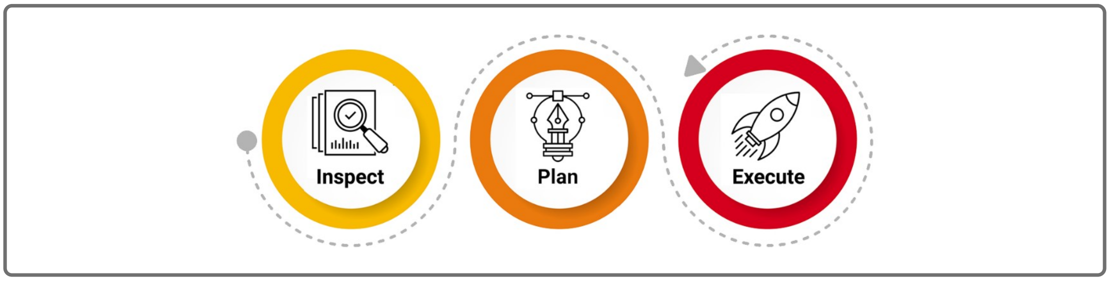
8.3.2 Iterative Process for Cleaning Data

Britta is confident in your data-cleaning strategy. You're going to follow an iterative process based on three key steps: plan, inspect, execute.



The iterative process for cleaning data can be broken down as follows:

- First, we need to **inspect** our data and identify a problem.
- Once we've identified the problem, we need to make a **plan** and decide whether it is worth the time and effort to fix it.
- Finally, we **execute** the repair.

Early iterations focus on making the data easier to investigate: deleting obviously bad data, removing superfluous columns (e.g., columns with only one value or missing an overwhelming amount of data), removing duplicate rows, consolidating columns, and reshaping the data if necessary.

As the data becomes easier to investigate, iterations focus on fixing the most obvious problems first. As obvious problems are resolved, more subtle problems become noticeable.

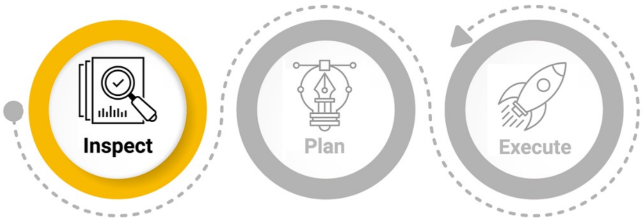
As the iterations shift toward solving more subtle problems, we might discover an earlier step needs to change as well as all the iterations that follow that step. It's frustrating when work has to be undone, but at least you now have a better understanding of your data.

NOTE

> In general, earlier iterations try to handle big chunks of data at one time, such as removing columns and rows, while later iterations focus on smaller chunks of data, such as parsing values.

It's rare to reach a point where no more problems exist in the data. More likely, a point is reached where the work to fix any remaining problems isn't worth the amount of data that would be recovered. After the remaining issues are documented, the transform step is considered finished.

Now that we know how to use our iterative process, let's review each step in detail.



Before we can do anything, we have to look at our data. The first thing we want to know is whether or not the data was imported correctly. The simplest way to confirm this is to print out the first few data points and examine the first few rows for irregularities, e.g., data in the wrong columns, all missing values, column headers that don't make sense, or garbled characters.

If the data doesn't look correct, we know it wasn't imported correctly. Sometimes the beginning of the data looks fine, but if the import went wrong somewhere in the middle of the process, the rest of the data can be affected.

Therefore, it's good practice to check the last few rows and a random sample of rows. We can also start to answer some simple questions about the data:

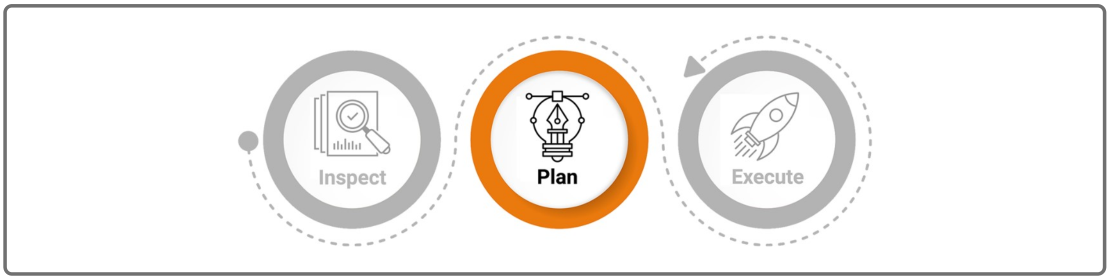
- Does it have a consistent structure (like a CSV table) or is it unstructured (like a collection of email messages)?
- How is each data point identified—is there an explicit, unique ID for each data point, or will one need to be built?

However, most usable data contains too many data points to review every single one, so we'll need to use strategies that tell us about the whole dataset.

First, count how many data points or rows exist. If the data is structured, count the number of columns and missing values in each column. If possible, count the number of unique values in each column and how frequently each unique value appears. To determine if this is possible, we'll need to investigate the data types for each column.

When investigating the data type for a column, we want to know what the data type is and what the data type should be. For example, if we see "True" and "False" as entries for a column, we expect that the data type will be a Boolean. If the data type is a string, we need to investigate further.

If a column's data type is numeric, we can summarize its data with some basic statistics, such as measures of central tendency (e.g., mean and/or median) and measures of spread (e.g., standard deviation, interquartile range, minimum/maximum). We can also investigate columns with statistical plots, like scatter plots and histograms.



After we've investigated our data and started to identify problem areas, we can make decisions about how to fix the problems. This requires articulating the problems clearly—even if that is simply expressing the problems to ourselves—and devising a plan to modify the data and fix the problem. In this step, we'll answer several questions, including:

- If a column doesn't have the right data type, is it a problem with the whole column? Or are just a handful of rows causing the issues?
- Do rows have outliers due to spurious data? Or are they valid data points?
- When values are missing, will they need to be removed, replaced, or interpolated?

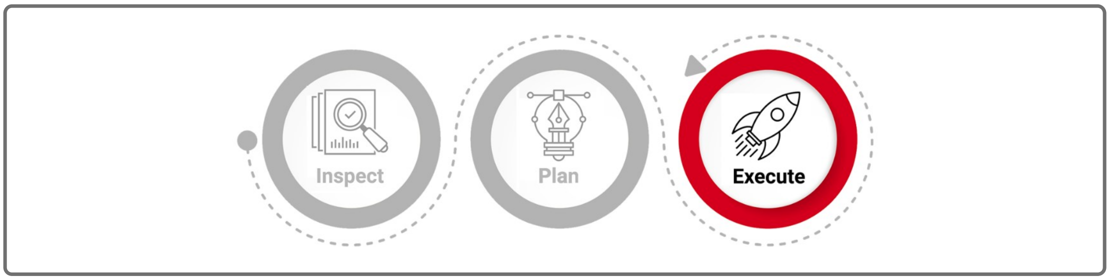
The answers to these questions will tell us how we need to modify our data. Keep in mind, there are two main ways: we can modify values and we can modify structure.

Modifying data values includes removing rows or columns, replacing values, or generating new columns from old ones. We might remove rows with missing or corrupted data, columns with only one value, or columns mostly missing data. There are many ways we might replace data. Instead of dropping missing values, we might replace them with zeros or empty strings. We might have a column that contains nonstandard values, such as percentages that are stored as whole numbers from 0 to 100 and also as fractions from 0 to 1, and we would replace them with one standard form.

Converting a column to a new data type is also a form of replacing values. We can also bin data (like rounding to the nearest hundred), replacing numeric data (e.g., income) with categorical data (e.g., income brackets). We might generate new columns by splitting an existing column into several new columns—by splitting an address column to street, city, state, and zip code columns, for example—or by calculating a new column from multiple existing columns, like calculating total price by multiplying item prices by quantities.

Modifying data structure includes pivoting the values of one column into multiple columns, aggregating rows, and merging multiple data sets. It can also include aggregating large amounts of data into summary data or summary statistics.

With clearly stated steps to fix the problem, we can make an informed decision about whether implementing the plan is worth the effort. Sometimes there are multiple viable resolutions to choose from. To decide, we weigh trade-offs and ultimately choose the best option.



Once we have a detailed list of steps to modify our dataset, it's time to implement it. We'll start writing code to fix the problem we're focusing on.

As we write, we might discover that the problem is more difficult than initially expected. This is a normal part of the process. As you implement your changes, try to take into account any unintended consequences you could introduce.

After implementing your changes, the next step is to return and **inspect** the data in a new iteration. This step is important, especially when modifying data structure, which can introduce missing data points, or inadvertently create more bad data.

Cleanup Is Messy Work

While transforming your data, you might bounce between steps in the iteration—for example, making a plan, then realizing you need to inspect more; executing a plan, then realizing a step was missed and you need to quickly rework the plan. We offer these steps as a descriptive, not prescriptive, approach. Cleaning up messy data is a messy process. The best practice is to document every step of your thought process and actions in detail.

Now let's go get our hands dirty with some messy data.