# **Data Cleaning in Python Essential Training**

<https://github.com/LinkedInLearning/data_cleaning_python_2883183>

### **Why is clean data important?**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Miki] Real-world data is messy, it contains bad values, spelling mistakes and missing data. It is estimated that data scientists spend between 80 to 90% of their time cleaning data. Lucky for you, Pandas and Python are great tools for cleaning data. In this course, we'll cover many methods of cleaning data. We'll also cover how to avoid some common mistakes, detect bad data and monitor the quality of your system. Let's get started.

### **What you should know**

Selecting transcript lines in this section will navigate to timestamp in the video

- If you be familiar with the Python programming language and have a walking experience with pandas. I'm going to use Visual Studio Code, during the course with the Python extension, but feel free to use another ID if you feel more comfortable with it. We also going to use several Python packages in our code. See the requirements.txt file in the GitHub repository. You can install these requirements with the Pip tool using python -m pip install -r requirements.txt

### **Types of errors**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] You will have errors in your data. We'll discuss the cause for these errors in another video, but for now, let's have a look at some of the common errors. One of the most common errors is **missing values.** For example, if you have our shopping cart data, when you look closely, you can see that on the third line, the amount is missing. Another type of errors are **bad values.** For example, in this shopping cart data, you can see that $217 for three pounds of carrot is a bit extreme. And the last type I'm going to mention is **duplicated data.** If you look at this data, you will see that client number three and number five are duplicated. These three kinds of errors are the ones I find most in datasets and we'll cover how to detect and how to fix them.

### **Missing values**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Let's load some shopping cart data. Well, here we have some shopping cart data in a CSV file. We have the date, the name, the amount and the price. So let us load it. We import the pandas then we do a read\_csv. I'm going to hit shift and enter to run the cell. And you can see that pandas is loading the data and we have some missing values. Some of them are denoted as NaN, not a number. And here we have NaT, which is not a time. Pandas also treat Python's None as a missing value as well. If you look at the datatypes, by doing df.dtypes, you will see that the amount column is a float. But if you look at the CSV, you will see that these are always whole numbers. What pandas is doing is because we don't have special missing values for integers, it converts the data type to a float. In later versions of pandas, we have an experimental IntegerArray, where the values are integers and we can also have missing values. So if you run df amount and astype Int32, now we are going to get the numbers as whole numbers, and we have a special missing value for integers. To find out if values are missing, you can use the pandas isnull. And isnull treating NaN, the numerical one, Python's None and the NaN object in the pandas array we just saw, and also not the time. So if we run df isnull, we will get a true or false for every value in the data frame. If you want to check if a row has at least one missing value, you can say X is equal one, and then you will get the mask for every row. Note that the empty string is not considered null. You will need to use Boolean indexing to find empty strings.

### **Bad values**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Your data will have bad values. When I say bad, I mean data that was generated by error. It can be out of scale values, for example, a thousand degrees for a body temperature, or spelling mistakes, or any other kind of errors. Let's have a look at some metrics data. So here we have our metrics, let's load the data. So we'll import pandas, shift and enter to run the cell. And then we load the data frame and show a random 10 rows from the data. So we have some rows with CPU and memory and some kind of values. Let's use groupby to have a look at statistics per metric. So we are going to groupby name and then do describe. And we see that we have CPU in lowercase and CPU in upper case, and we also have the memory. And we see that, for example, for the CPU, we have a minimum of -34, which probably is a bad data. The value\_counts method is a great way to find problems in categorical data, so I'm doing the df name value\_counts, and I see that I have 49 of CPU, and one CPU with uppercase. Some people find it easier to see bad data with graphs, so I'm going to hide the side frame and let's see what I'm doing here. I'm doing a pivot table where the index is the time and the columns are the name, and then I'm going to plot it with subplots equals true. And we can see the values for the CPU, the memory, and the CPU that is misspelled in capital letters. If you know the range of valid values for your data, you can find out the values outside the range by using a query. So I'm doing the name is CPU and the value is either smaller than zero or bigger than 100. And I have two rows where the values are bad. Sometimes, you'll use a more sophisticated method to find bad data. **The standard score, or the z-score, is the distance from the mean in units of standard deviation. It's a good way to find outliers if the data is normally distributed.** You should know your data distribution and other characteristics. So here we are going to get the memory, calculate the z-score, and find out the bad memory. And now if you want to look at the rows, we'll use the index of the bad memory on the data frame. And we have one row that is bad. There are more sophisticated methods of finding outliers in the scikit-learn package.

### **Duplicates**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Man] Duplicate data will make your calculations wrong. Let's have a look at the shopping cart data. We have the date, the name, the amount and the price. So let's load the data. Start with importing pandas as pd, shift and enter to run the cell. I'm going to hide the sidebar. Now, I'm going to read the data, parsing the date column. And now I'm going to use the Data Frames duplicate method to find out if there are duplicates. And we see for every row it will tell us if the row is duplicate or not. It marks only the fifth row, which is a duplicate of the fourth one. This caught only one of the duplicate roles, right? We also have one with eggs at the same date. If you look closely, you see that in rows one and two, the name and the amount and the date are the same, but the price is different. But this will duplicate, will consider values from all the columns, but you can tell it to look for subset of columns. In databases, we call this the key. So we'll do duplicated again with only the date and the name is duplicated. And this time we see that row number two is also marked as duplicate.

## **Question 1 of 4**

What are valid values for the “cpu” metric?

* up to 75
* not negative
* between 0 and 100  
  Correct
* less than 100

## **Question 2 of 4**

What were the columns used for duplicate?

* date and name  
  Correct
* price and amount
* date and amount

## **Question 3 of 4**

What are the common errors Miki mentioned?

* bad data  
  Incorrect
* duplicated
* all of these answers  
  Correct
* missing values

## **Question 4 of 4**

What does Pandas consider a “missing value”?

* NaN, NaT, and None  
  Correct
* None
* NaN and NaT
* NaN and the empty string

### **Human errors**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Humans are by far the most common cause of errors. If only we can remove them from the equation. The most common place where we humans make errors is when entering data. Let's have a look at an input form for a payment. So we have a donation for Scrooge McDuck. The name looks like it's missing a space, but actually there is a zero with Unicode space there. This is mostly caused by copy and paste error. The country has a typo. The credit card number is missing a digit. The expiration is in year and month and not month and year. And the amount is two and then a capital O, and not 20. Most of these mistakes can be avoided by creating a good UI, which does validation before submitting. However, in some cases, such as names, and emails, it's hard to do a good validation.

### **Machine errors**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Narrator] Machine errors can come for various causes. Some values are calculated, say BMI. And the code that calculates them might have bugs. Make sure you save raw data if you ever need to recalculate. Other errors can be caused by faulty sensors or machines. A very common error is clock accuracy. Your computer clock drifts away from the accurate time and we need to sync with an NTP server from time to time. Other sensors can go wrong, from dirt on camera and says a faulty network transfer error, and even cosmic rays. Some machine errors are serialization error. You might transfer data in a big NDN format, but the other side we'll read them as little NDN format.

### **Design errors**

Selecting transcript lines in this section will navigate to timestamp in the video

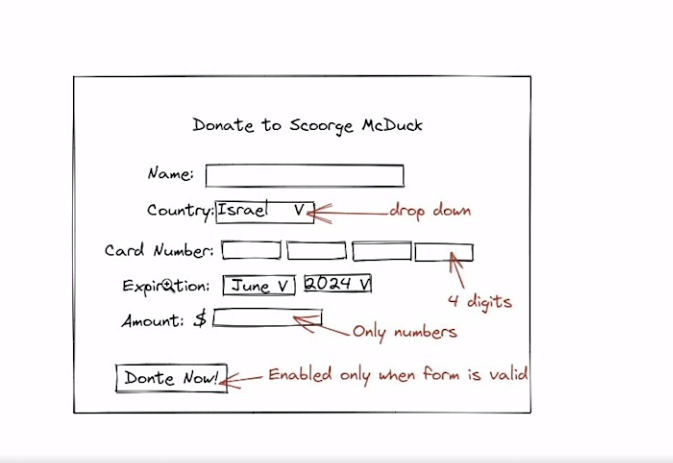
- [Instructor] There are two aspects to design. One is the UI for collecting data. For example, in our payment form, instead of writing the country, you can have the user pick a country from a dropdown list to avoid errors. The second is system design. For example, when we talk about the payments, you can say that the payment has both the currency and the amount, and this will help the UI design with the data validation. You can go further and say that the amount must be bigger than zero. **There is a field called ontology engineering, which formally defines the relationships between all the pieces of the data in your organization.** It might seem theoretical, but once you find you have several different definitions in your company for what the customer is, you would understand the need to be precise in your definitions.

### **Challenge: UI design**

Selecting transcript lines in this section will navigate to timestamp in the video

(upbeat music) - [Instructor] Improve our current payment form and try to make it error-proof as possible. Don't go fancy. Paper to sketch the UI and write the notations.

Do you need to understand how to keep data clean and well-organized for your company? In this course, instructor Miki Tebeka explains why clean data is so important, what can cause errors, and how to detect, prevent, and fix errors to keep your data clean. Miki explains the types of errors that can occur in data, as well as missing values or bad values in the data. He goes over how human errors, machine-introduced errors, and design errors can find their way into your data, then shows you how to detect these errors. Miki dives into error prevention, with techniques like digital signatures, data pipelines and automation, and transactions. He concludes with ways you can fix errors, including renaming fields, fixing types, joining and splitting data, and more.



## **Question 1 of 2**

In our example, what was the problem with the year?

* It was OK.
* The year was bad.
* It was in YY/MM format.  
  Correct
* The month was bad.

## **Question 2 of 2**

How can you prevent wrong country?

* by using a dropdown  
  Correct
* by spell checking
* by converting to uppercase

### **Schemas**

Selecting transcript lines in this section will navigate to timestamp in the video

- Schema defines the relationships and the constraints on your data. Some formats such as JSON do not have a form of scheme, and some like protocol buffers do have a scheme. Even if your data does not have a former schema, it is still out there as your assumptions about what is valid data. If you don't use a formal schema, most likely these assumptions will be scattered throughout the code and in developer heads. And that is not good. Say you're writing code to track ships in the ocean. For every observation of a ship, we have the name and the location in latitude and longitude. Here's an example of data in CSV format. We have name, latitude, longitude. We have the Black Pearl, Cobra, the Flying Dutchman, and the Empress at an unknown location. Most schemas have constraints about the type of data. There are various way to check that the data confirms to a schema. If you have control over the data pipeline, you can use tools such as JSON schema, database schema, or languages like QUE to validate your data. In **Python, you can use libraries such as pydantic or marshmallow to check the validity of the data.** In my experience, eventually you're going to write your own code to validate data. Let's load the data and have a look at the types. So in ships.py, we import pandas and load the ships.csv. Shift and enter to run the setup. And now we can use the D types attribute to check what pandas figure out the types are. Here's what an SQL schema for this type might look like. We have the name as text, the latitude and the longitude as floats. You need to think about whether you allow missing values or nulls in your data. In SQL, for example, you can add a constraint saying not null.

### **Validation**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] **Once you have a schema, you can validate your data. I'm going to use the Pandera library, which is a statistical data validation for pandas.** Let's have a look. Here, we have our shipping data with the name, the latitude, and longitude. So, we'll start by import pandas as pd and loading the data. Shift-enter to run the cell, and now I'm going to input Pandera and define a schema. For the name, it's a string, for latitude and longitude, these are floating point columns, and finally, I'm going to call validate on the data frame. So when I'm going to run this one, **it is going to fail, and the reason it's failing is because Pandera, by default, does not allow null values in columns. So we can fix this by adding nullable=True for the latitude and also nullable=True for the longitude, and if you're going to run this again, this time it's going to pass.** Should you allow null values in some of your columns? There's no good general answer. You should know your data and decide.

### **Finding missing data**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] In some cases, some data will be missing. Missing data can come from user refusing to answer a question, all data that is missing some features and other reasons. Let's have a look at some shipping data. So, here, I have the name, latitude, and longitude, we have the Black Pearl, the Cobra, the Flying Dutchman, the Empress at an unknown location in an unknown ship at some location. So let's load the data, so Shift + Enter to run the cell. And we can see that we have N-A-N, not the number values for the Empress and the unknown ship, with an empty cell. So I'm going to run isnull from pandas, and say axis=1 to see the rows that have at least one missing value. And we get only the Empress. Why is that? Let's have a look at the name of the ship in the last row, and we see that it has an empty space in it. So we are going to use the str.strip to strip the spaces and look at the name. **And, now, it's really an empty stream, so we can run isnull again. But, again, we see only the Empress. And the reason is that if you look at the isnull documentation it only recognize N-A-N, Pythons none, N-A-N from pandas and not in time, but not the empty string. So we're going to go back, and set the value of everywhere that has an empty string to NumPys.nan. So Shift + Enter, and we're going to run the cell. And, now, when we go in to run it, we see both of the ships.** The question of what is a missing value is trickier than it seems. You can think of user answering, "I don't know" for a question. All the time you will gather a set of missing values per column, make sure to update the set and use it in your code. You can also have missing rows of data. This is harder to detect, in most cases, but when you have time-based value this might be easier to detect. For example, if you track the location of a single ship, we have the dates, we can easily see that between the 8th and the 10th we have a missing value. I encourage you to read the working with missing data section, in the pandas documentation, to learn more about how to find missing data.

### **Domain knowledge**

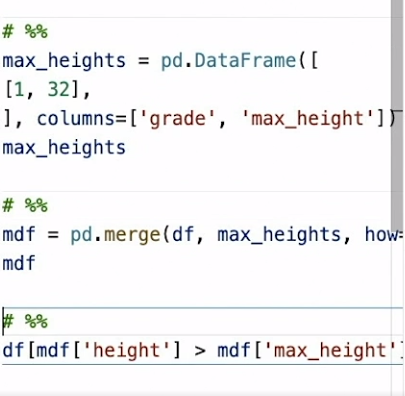
Selecting transcript lines in this section will navigate to timestamp in the video

- Domain knowledge will help you know what are valid values for column. For example, if you look at latitude and longitude, we know that latitudes should be between minus 90 and 90, and longitudes should be between minus 180 to 180. Let's use this knowledge in our scheme. So we have our shipping data with ship name, latitude, and longitude. And here are the ships. So we start by loading the data and I'm going to hide the sidebar. And now we have a data frame. And now we use the Pandera Schema. We say that the latitude is a Column, it is a Float, can be Null, and the check is that the value is between minus 90 and 90. And the same for the longitude. And finally we do validate of the data frame. And the validation pass.

### **Subgroups**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Let's say you want to measure the height of students. You have a data in the CSV for several of students. You do some research and find out that the tallest person alive was 107.1 inches. Look at the data and this seems okay, but when you look closer, you see that Beth is awfully tall for a first grader. You do some research and find out that the first grade heights are up to 32 inches. So, let's start check it in our code. First, we're going to load the data frame. So, 'shift' and 'enter' to run the cell. Now, I'm going to create another data frame for maximum heights, and it is going to have the grade and the maximum height. I'm going to have just a single row in it. Now, I'm going to join these data frames, and finally, I'm able to find the rows where the height is bigger than the maximum height, and I see just the row with Bethany.



### **Challenge: Find bad data**

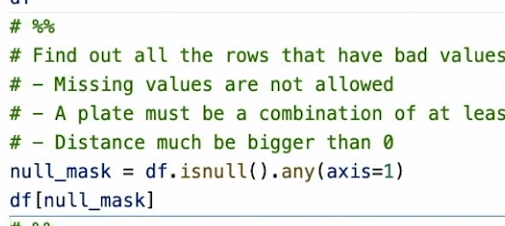
Selecting transcript lines in this section will navigate to timestamp in the video

(upbeat music) - [Instructor] Here we have some right information. We have the name of the driver, the license plate, and the distance state road. You need to validate this data by finding out all the rows that have bad values. Missing values are not allowed. The plate must be a combination of at least three upper case letters or digits, and the distance must be bigger than zero.

### **Solution: Find bad data**

Selecting transcript lines in this section will navigate to timestamp in the video

(upbeat music) - Here is my solution. We'll start by loading the data frame. Shift + enter to run the cell. And I'm going to hide the sidebar**. I'm going to create several masks to find out bad rows. The first one is for the null ones. So I'm using the isnull with axis equal 1, and look at what** I have there. Second one is that I'm going to use the plate validator. So I'm going to make sure that the plate matches the regular expressions for numbers and digits in uppercase, at least three, and do not allow any. And I have several of these. The last one is the distance mask. When I'm looking for rows where the distance is smaller than zero. Finally, I'm going to combine all of these mask using the OR sign and find out the bad rows in the data frame, which are almost all of them.



## **Question 1 of 4**

Which format has both types and schema?

* CSV  
  Incorrect
* JSON  
  Incorrect
* XML  
  Incorrect
* Protocol buffers



Replay

Review this video

Schemas

2m 6s

## **Question 2 of 4**

What did we use the domain knowldege for?

* to fix names of ship
* to valiate latitude and longitue  
  Correct
* to fix missing values
* to sail

## **Question 3 of 4**

Why did the validation fail on the first attempt?

* by default pandera don’t allow null values  
  Correct
* The name was missing.
* The latitude was bad.

## **Question 4 of 4**

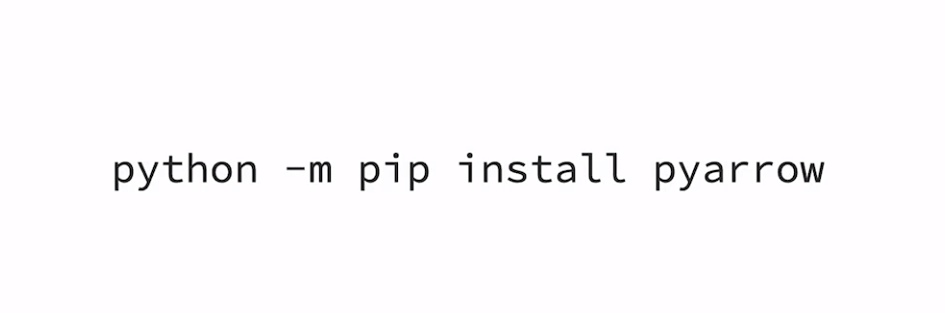
When is it easy to detect missing rows?

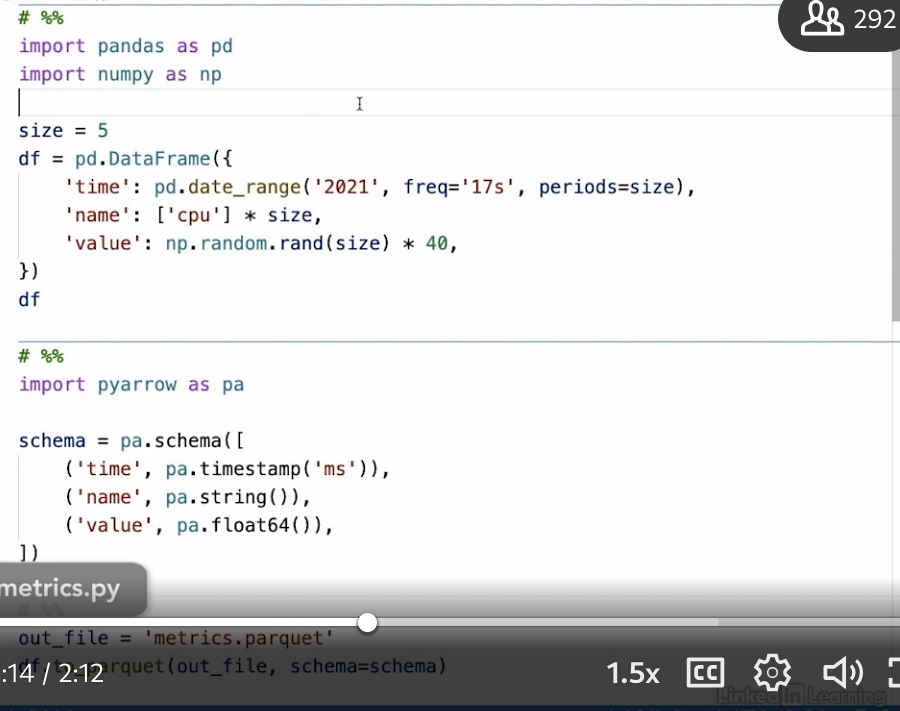
* in floats  
  Incorrect
* in time series data
* in numerical data  
  Incorrect
* in textual data  
  Incorrect

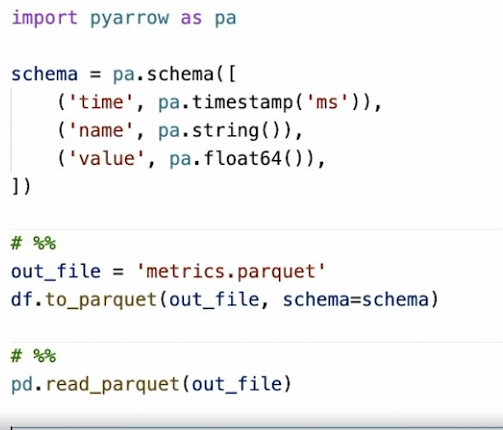
### **Serialization formats**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] There are many ways and formats to store your data. Some serialization formats, **such as CSV, are only text.** Pandas has a very versatile read\_csv function, and you can give pandas a lot of things about which types each column has. **Some serialization formats, such as JSON, have types. However, JSON has only very basic types. It has a string, a Boolean, an array, null, and a single type of number**. **Some serialization formats, such as SQL, have both types and a schema, which is usually the best case to use.** We are going to use the **Apache Parquet format, which is a format for storing columnar data on disc and is very popular nowadays. In order to use Parquet from Python,** you can use the Apache Arrow library. You can install PyArrow with pip, python -m pip install pyarrow. Let's have a look. First, we're going to generate a data frame with some random values for a CPU metric, so shift + enter to run, and I'm going to hide the sidebar. And we have a data frame, the metric is the CPU, and we have several values and several times. And now we can define the Apache Arrow schema. So we define the schema saying the time is a timestamp in milliseconds, the name is a string, and the value is float64, and it can run it. And now we can define the out file and save the data to the file using this schema. And once we do that, we can read back the data and you can see we get the same types for the values. The time column is a timestamp, the name, and the value is a float number. If you're going to change, for example, we'll convert the time to a string and then try to save it, you will see that Parquet will complain and say that the data does not match the schema, which can save us a lot of trouble in the future.



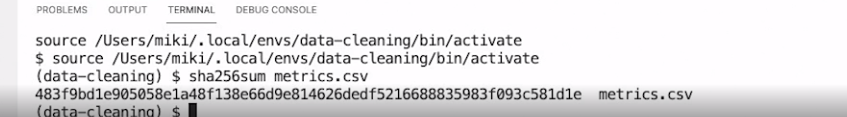


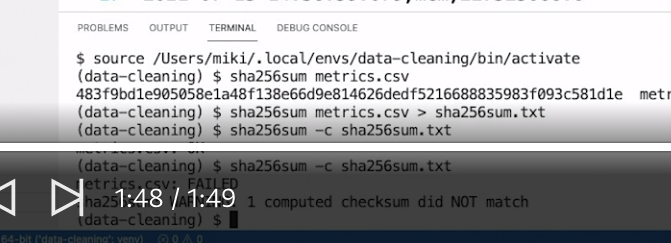


### **Digital signatures**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Files can get corrupted. Corruption can happen due to various reasons, such as a network error, a disk failure, and many others. A common practice is to calculate the digital signature of a file, also known as a hash, and store it next to the file. **This way, you can validate the integrity of your data before you start processing it. There are several digital signature algorithms, such as MD5, SHA1, SHA256, CRC32, and more. Pick one that is secure, and depending on your performance goals, not too slow. Most of these you'll find in *hashlib* in the Standard Library, and several of these are also in the *zlib,* in the Standard Library as well. And there are also command line utilities to calculate these signatures.** Let's have a look. Say, we have this file with some metrics, and we would like to see how to validate it. So, we are going to open a terminal next to the file. And now, I can run SHA56, sum of the metrics, and I will get the long number. **And this number is the digital signature of the file. If the file will change, the digital signature is going to change.** **So, what we can do is we can store it in a file next to our data file. And now, we can tell the SHA256 utility to check, so we do slash C for checking, and it will tell us that everything is okay.** Let's change something, so I'm just going to delete one character and save the file. And now when I run the utility, it will tell me that there is a mismatch in the file.

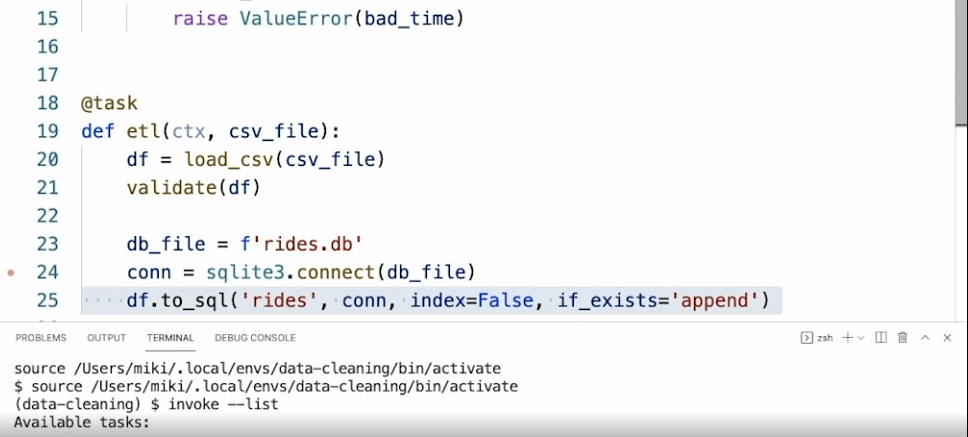




### **Data pipelines and automation**

Selecting transcript lines in this section will navigate to timestamp in the video

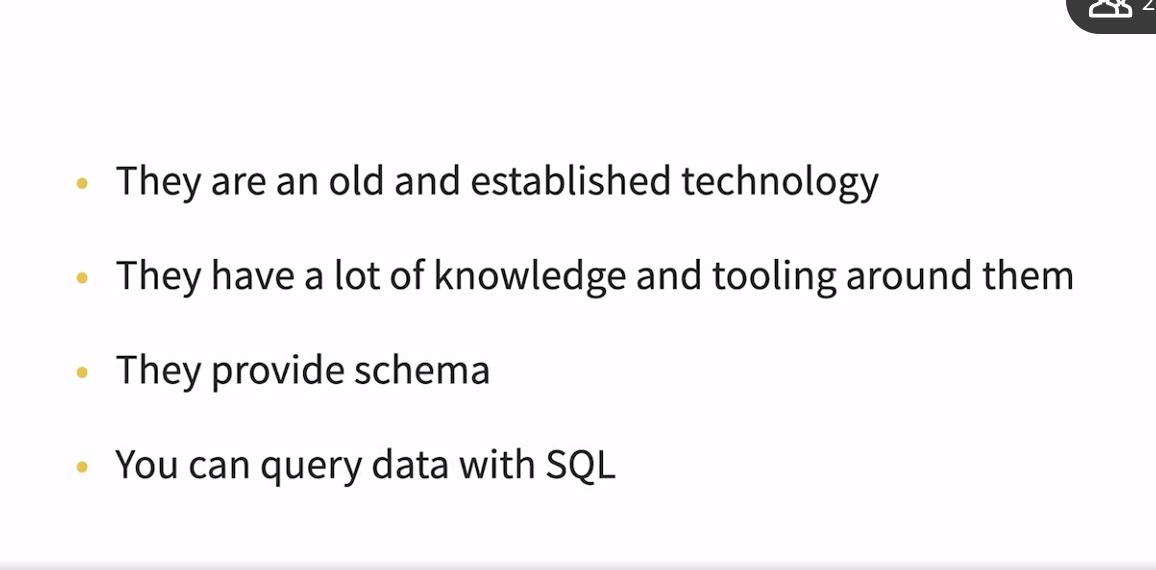
- [Instructor] Data pipeline is a series of steps, each consuming input and producing output. There are many systems for creating data pipelines, such as **Apache Airflow**. I think your own is not that hard, but I recommend investing your time in an existing one. The main advantage of a data pipeline is that each step is small, self-contained and easier to check. Some data pipeline systems also allow you to resume the pipeline from the middle, which is going to be a big time-saver. **When designing data pipelines, it's important to a data validation and cleaning into the data pipeline.** Once **you have these in place, you can quickly detect errors and stop the pipeline. For simplicity, I'm going to use the *pyinvoke library,* which lets me write tasks and then execute them.** So here's our data, it's a CSV file and we are going to load it inside an escalate database. So let's see the code. The first step is loading the CSV. We're doing it using pandas. And then we do a validation and here we just detecting if we have start time that is bigger or equal to the end time. And finally we have the etl short for extract, transform and load, which is decorated with a task decorator and we load the data, validate, and then insert it into the database. So let's have a look. I'm going to open the terminal and I'm going to write invoke --list. And this will show me the list of available tasks and surprisingly, we have only the etl task. So now I can invoke etl and I'm going to pass it the CSV file, which is rides.csv, and it is done. You can see here in the sidebar that we have now rides of DB. Let's have a look inside of it using the sqlite3 utility. So sqlite3 rides.db. And if I do .schema, I'm going to see that we have the schema for our data and I can do also select star from rides. It's not such a big data, so I'm able to see that my data is there.



### **Transactions**

Selecting transcript lines in this section will navigate to timestamp in the video

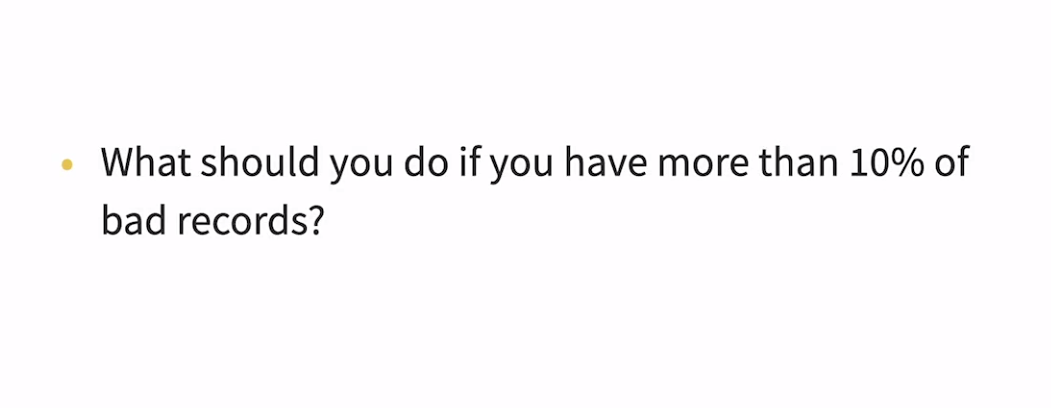
- [Instructor] There are many reasons why I like relational databases. They're an old and established technology, they have a lot of knowledge and tooling around them, they provide a schema, and you can query your data with SQL, but one of the most important features when talking about data pipelines is transactions. A transaction means either all of your changes go into the database, or none of them. Say you insert data and suddenly there's an invald record and you get an exception. Without a transaction, you need to manually figure out which records made it into the database, and which one you need to try again. Also, if you have a downstream system that processes data for a given day, how would it know that the upstream process is done inserting data? With transaction, it's either there or not. Let's have a look at how we can use transactions with SQLite 3 that comes bundled in with Python, but this works with almost every other relational database. So we have our ships database, we have four ships and the Empress is missing the latitude and the longitude. So here's our process, we start by loading the data and I'm going to hide the sidebar. All right, so we now have the data. So I'm going to define a schema for this table, which says that I'm not allowing null values in the latitude and longitude, then I'm going to connect to the database and create a table, and now I'm going to do with connection as cur, and this is starting a transaction when I do a BEGIN, and then I do data frame to SQL, to the ships table with connection, and if the table already exists, we append the data and we don't want to store the index inside. Let's run this, and unsurprisingly, there is an error because the Empress has some null values. Let's see about the data. So as we can see, the database was created, and I'm going to use the SQLite command line utility to have a look. So sqlite3, ships.database, and when I do the .schema, I see that I created a table, it is there, but when I'm doing select star from ships, I see that there's no data inside the table. Nothing came in.

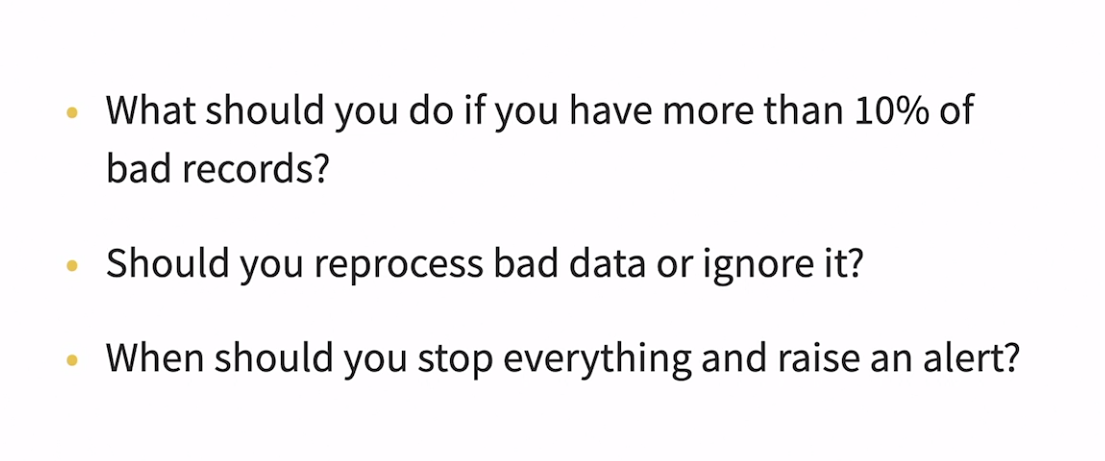


### **Data organization and tidy data**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] One of the worst thing you can do to your data is let it grow organically. I worked with companies who have no idea what some columns mean and how they get populated. You need to think about your data structures, relations, and more, and actively maintain your data scheme. One good way to organize your data is in a tidy data format. In tidy data each row has a single observation. Let's see an example. Here is the metrics data and this is known as a wide format. We have more than a single observation in every row because we have both the value for the CPU and for the memory. What we'd like to see is what is known as a narrow format or tidy format where each row contains a single observation. We have the time, we have the metric name, and then we have the value.





### **Challenge: ETL**

Selecting transcript lines in this section will navigate to timestamp in the video

(upbeat music) - [Instructor] Here we have a CSV with some information about traffic to our website. We have the IP, the time, the path, the status code that was returned by the web server, and how many bytes were returned. I would like you to write an ETL into an sqlite database. You should drop and report invalid roles. And the constraints are, the IP should be a valid IP, and you can use the IP address module. The time must not be in the future. The path cannot be empty. The status code must be a valid HTTP status code, and you can see the HTTP status class for that. And the size cannot be negative or empty. You pull the percentage of bad rows and failed ETL if there are more than 5% of bad rows.

## **Question 1 of 6**

What do we use digitial signatures for?

* to calculate values
* to validate the integrity of the data  
  Correct
* to catch forgery
* to fix transport errors

## **Question 2 of 6**

What are good properties of a data pipeline step?

* have fixed output  
  Incorrect
* fast and small  
  Incorrect
* small and self contained
* easy to reason about  
  Incorrect



Replay

Review this video

Data pipelines and automation

2m 26s

## **Question 3 of 6**

Why is parquest better than CSV?

* It’s has types.
* It has types and a schema.  
  Correct
* It’s smaller.
* It’s faster.

## **Question 4 of 6**

What should you do with data metrics?

* Print them.
* Read them from time to time.
* Send them to data monitoring system.  
  Correct
* Log them.

## **Question 5 of 6**

What starts the transaction?

* the BEGIN statement  
  Correct
* the cursor
* the connection
* the context manager

## **Question 6 of 6**

What is tidy data?

* good values
* short names
* one observation per row  
  Correct
* many rows

### **Renaming fields**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] In some cases, the fields in the raw data does not match the name that we want to use, so we can rename them. Say, for example, we have weather data, which looks like this. We have the DATE, TMIN, and TMAX, and we'd like to change it to date with lowercase, min temp and max temp. So first we're going to load the CSV as is into a data frame, hitting shift, and enter to run the cell. Next, we're going to run the data frame, rename method, giving it the columns we want to change and passing it as a dictionary between the old name and the new name. And we also say in place equal true meaning change the count data frame and don't return a new one and let's run this one. And now we have the right names. In some cases, the transformation can be trickier. Say for example, we have donations data, which looks like this. One, first name, two, last name and three, donation. And we'd like to normalize that, to remove the spaces, make everything lowercase. So here again, we can load the data frame of the donation. And now we define a function, that fix the column name. It gets the column name and then use the regular expression model to substitute digits and spaces to nothing, convert it to lowercase and then convert space to an underscore. Finally, we call rename again, and this time we give it the function and have the dictionary and with it in place again. And this time we have first underscore name, last underscore name and donation underscore amount.

### **Fixing types**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] When dealing with CSV, pandas is going to guess the types for us, and most of the time it does a good job, but in some cases we'll need to help it a little. Let's have a look at data with some points and colors. So we have our points. We have the X and the Y coordinates, we have the color, and we have if the point is visible or not. Let's slow the points and see what data types pandas' got for us, hitting 'shift' and 'enter' to run the cell. And we see that the X and Y are good, but the color and the visible are object, which means a string in pandas. We would like to fix that. Let's start with the color. I'm going to hide the sidebar. Okay. So we define the function as integer, which gets the value. It's going to be a string, and we're going to call "int" on the value and say, the base is zero, meaning Python will guess the base, according to the prefix, and because there is a zero X at the prefix, it's going to guess base 60. And if you're on this cell, now we see that the color is an integer. Next we go to the visible which is a boolean. This time we're going to use a dictionary between "Yes" and the value "True," "No," and the value is "False." And this time you're going to use "map" instead of "apply" to get the dtypes. Let's run the cell. And now the btypes are integer, integer, integer, and a boolean, and finally, we can look at the resulting data frame, and this looks fine. The numbers because they're hexadecimal, but the pandas shows them it's decimal, looks a bit different than the ones in the CSV.

### **Joining and splitting data**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Narrator] Sometimes, the data will be combined together in a single column, and you will need to split it up and maybe join it later. Say for example, that you are a freelancer working for several companies, and would like to calculate how much time did you work? You have a CSV file with the day of the month, the time that you started and ended in a single column, and then the customer name. Also note the name of the CSV file contains the date. So let's stop. We'll start by loading the CSV into a data frame. Shift + Enter to run the cell, and I'm going to hide the sidebar. So we have the day, we have the time, which is a string, and the client. We are going to add a column for the date, from the file name itself. So now, we have the date as a column as well. Next, we're going to take the time, and split it. So we're going to do df['time']STR, accessing the string methods on the column, saying we want to split by the dash, and expand, meaning create a data frame with two columns, give it to the name of the two columns that we can use. So in other times of the start and the end, and everyone has a start time and an end time. And now, we're going to use pd.concat to concatenate these two data frames together to create one single data frame. We say access to Q1, meaning we want to do it horizontally. And now we have a data frame and we have our two new columns start and end at the end. Next, we want to convert these to daytime. So again, we're going to use the STR method of the date, and we're going to do it cat with the start, and add a separator of T between them. This will create a string and format that pandas to daytime can understand. And now in look at the columns, starting date are daytime objects that have a start and an end. And now we can do the calculation. We're going to do the end minus the start, and do a sum and see that we worked 23 hours up to now.

### **Deleting bad data**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Presenter] You will have bad rows in your data. And one of the easiest decision is just to delete them. Say, for example, you have data on some rides. So we have the rides, we have the name of the driver, the license plate and the distance. And you would like to remove the bad rows, which either has an empty name or a distance, which is zero or less. So how are we going to do that? First, we're going to load the CSV into a data frame, shift and enter to run the cell. And I'm going to hide the sidebar. And now we're going to create a mask, we're going to use the data frame eval method saying either the name is null or the distance is smaller or equal to zero. And we are going to get a mask of false and trues, whether the row is bad or not. What we're going to do now is we're going to invert this mask, using the tilde sign, meaning I want all the rows that are not marked as bad, and once we run that, we will get the data frame. Note that the index here has changed, we have one, two, and then it jumped to five. You can use the reset index to reset the index if you want.

### **Filling missing values**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] In some cases, you will have missing values inside the rows. Say for example, we have a shopping cart data. So we have the date, the name, the amount, and the price, and in some rows, we're missing the price. In some roles, we're missing the date. In some rows, we are missing the name. **Pandas has a great fillna method that helps fill missing** values in a data frame. Let's have a look. So, we start by loading the data frame, shift and enter to run the cell, and I'm going to hide the sidebar. So, the first thing you're going to fix is the amount. We're just going to fill it with a fixed number of one, meaning the amount is one if you don't know about them. You can see now that in row number two, we have amount of one. Next, we're going to fill the missing names. We're going to take the **most common name, so we're going to do the mode** of the name and take the first one, and then again, fillna and inplace for the same data frame, and we're going to run it now. You're going to see that the last name is potato instead of NaN. Next, we're going to fill the dates, and this time, we're going to use the **ffill** method, which means forward fill. It will fill from the row below and this is great for time series data. And now we can see that we filled row number four from not a time to the time that the date was before it. The last one is a bit trickier. We want to fill out the prices. So first, we're going to do a groupby operation on the name. We're going to take the price and you're going through the transform and the NumPy mean, which means we're going to get now a data frame with a name and the mean price for that name. So these are the prices and now we are going to fill the price from these prices in place again, and now all the prices are filled as well.

### **Reshaping data**

Selecting transcript lines in this section will navigate to timestamp in the video

- In some cases, you'll need to reshape your data. Say for example, we have some metrics data. This metrics data is in wide format, it has both the CPU and the memory in the same row and we would like to convert it to a tidy format when we have just one observation per row. To do that, **pandas has a great melt function for transforming wide format to a long format or a narrow format.** So, we start by loading the data, shift-enter, and I'm going to hide the sidebar, and now I'm going to use the melt. I'm saying, take these data frame, create values from the CPU and the memory. The identifier is the time and the variable name, please call it metric, and once I do that, it's like magic. Now we have a time column, we have the metric name, and we have the value, one per metric. Pandas also have a pivot table that can help you reshape data in various other formats.

### **Challenge: Workshop earnings**

Selecting transcript lines in this section will navigate to timestamp in the video

(upbeat music) - [Instructor] Say, you were keeping track of your workshops in Excel spreadsheet. You have a nice row for the year, and then the month, and then the start and end date, the name of the workshop, and the earnings. But once you start working it with Python and export it to CSV, the data gets messy. You have a row with only the year, one with only the month, and the rows with the data are missing both the year and the month. What you'd like to do is fix the data frame. You should have the following columns, start which is upon this timestamp and which is, again, at timestamp, the name which is str, the topic should be either Python or go and it should be a str, and the earnings should be float 64 number.

### **Solution: Workshop earnings**

Selecting transcript lines in this section will navigate to timestamp in the video

(upbeat music) - Here is my solution. We'll start by loading the CSV to a data frame. Shift and enter to run the cell. And now I'm going to hide the sidebar. You can see, we have the rows for the year and the month here, and here, that are full with not a number. First we're going to use the fill in A, with forward filling to fill the month and the year. And now we have the month, and we have the year. Next we can drop this rows, the ones that have NAN in the earning, because they are just for helping us out on the Excel, but they don't have a meaning when we do calculation. So I'm going to do where, not null of earnings, and I'm going to copy the data frame because we're going to make changes to the output. Next, I'm going to fix the date. So I'm extracting the year, the month, and the day from the row, I'm creating a timestamp string and then using PD two date time telling it the format of the string to convert it. And we're going to create two new columns called start and end with these timestamps. So when I run this, now you can see in the right side, I have start and end columns. Next, I'm going to extract the topic. So, I wrote the function. If go is in the name I return go. If python is in the name, I would return python. And I'm going to use the dot STR lower to convert the names to lowercase. So this role, which has Python with an opera case P will also work when we do this condition. And I'm going to apply the topic. And when we run this, now we have a topic column that was editing in the end. Now I'm going to fix the earnings. I'm going to use the STR.replace, and I'll give it a regular expression, either the dollar sign, or the coma, and remove them. And then I'm saying Pandas, converting to a numerical value with a float 64. And here again I have at the end, an earnings column with just floating point numbers. And finally, I'm going to take only the columns that are interesting to me. So I'm taking the start, the end, the name, the topic, and the earning. And I'm going to rename the name column through lowercase name. And finally we have the data frame as we wanted. With the start, and end, a name, topic, and earnings.

## **Question 1 of 6**

What’s the name of the function used to reshape the data?

* pivot\_table  
  Incorrect
* slim
* melt  
  Correct
* merge

## **Question 2 of 6**

How do you negate a mask?

* with not  
  Incorrect
* with ~  
  Correct
* You can’t.
* with !

## **Question 3 of 6**

What does “ffill” stand for?

* forward fill  
  Correct
* fixed fill
* function fill
* float fill

## **Question 4 of 6**

How does “int” know the base?

* It’s 10.  
  Incorrect
* by the string suffix  
  Incorrect
* by the string prefix
* It’s 16.  
  Incorrect



Replay

Review this video

Fixing types

1m 36s

## **Question 5 of 6**

What does “expand=True” in str.split do?

* It joins the time.
* It returns a DataFrame.  
  Correct
* It fixes the time.
* It returns a series.  
  Incorrect

## **Question 6 of 6**

What are the values in the dict passed to “columns”?

* Keys are old name, values are new name.
* a transformation function.
* Keys are new name, values are old name.  
  Incorrect
* Key are columns, values are rows.

### **Next steps**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] We've covered a lot of ground in this course. Next you should try to implement what you learned in your own project. Grab some data and start cleaning it up. If you don't have data, go and search for it. There's a lot of free data out there. For example, Kaggle is a great place to find real-world data. And feel free to reach out. I'd love to hear your war stories about messy data and how you managed to tackle it.