

# Notes Data Wrangling

Has notes from Udacity's Data wrangling module

## Chapter 1: Introduction to Data Wrangling

### Assess

Low Quality Data is commonly referred as dirty data. which has issues with it's content.

Common quality issues are

missing data

invalid data

inaccurate data

inconsistent data

Data quality is a perception or an assessment of data's fitness to serve its purpose in a given context.

### Tidiness

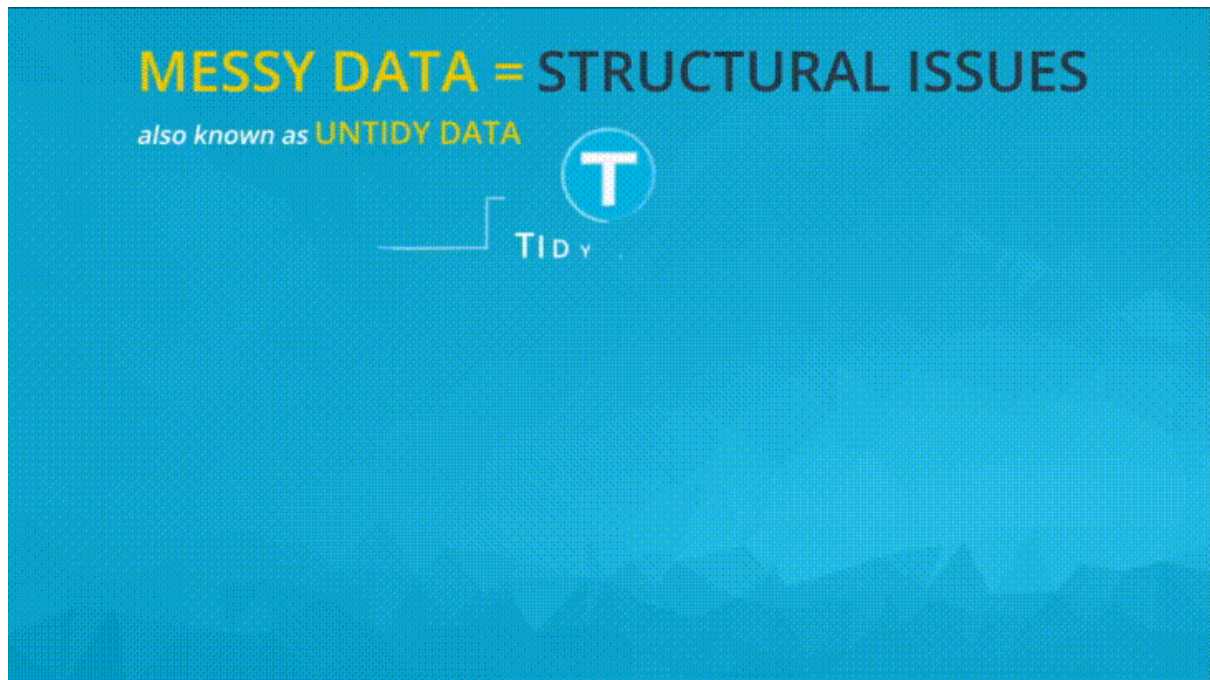
Untidy data is commonly referred to as "messy" data, data which has issues with it's structure

A dataset is messy or tidy depending on how rows, columns, and tables are matched up with observations, variables, and types. In tidy data:

Each variable forms a column.

Each observation forms a row.

Each type of observational unit forms a table.



this is gif explaining the concepts

## Visual Assessment

when data is scanned by human eyes to look for quality and tidiness.

## Programmatic Assessment

when we use computer program to check for quality and tidiness in the data  
this is done for large datasets generally.

Make sure you make notes of what problems you find in dataset regarding  
quality and tidiness.

## Cleaning

Cleaning means acting on the assessments we made to improve the quality  
and tidiness of the data.

```
animals.tail()
```

	<b>Animal</b>	<b>Body Weight (kg)</b>	<b>Brain Weight (g)</b>
<b>24</b>	Chimpanzee	52.160	440.0
<b>25</b>	Mouse	230.000	0.4
<b>26</b>	Apple	0.100	NaN
<b>27</b>	Brachiosaurus	87000.000	NaN
<b>28</b>	Mole	0.122	3.0

Examples of improving quality include:

- Correcting when inaccurate, like correcting the mouse's body weight to 0.023 kg instead of 230 kg
- Removing when irrelevant, like removing the row with "Apple" since an apple is a fruit and not an animal
- Replacing when missing, like filling in the missing value for brain weight for Brachiosaurus
- Combining, like concatenating the missing rows in the *more\_animals* DataFrame displayed below

## Improving Tidiness

Improving tidiness means transforming the dataset so that each variable is a column, each observation is a row, and each type of observational unit is a table.

## Programmatic data cleaning process:

Define → Code → Test.

**Defining** means defining a data cleaning plan in writing, where we turn our assessments into defined cleaning tasks. This plan will also serve as an instruction list so others (or us in the future) can look at our work and reproduce it.

**Coding** means translating these definitions to code and executing that code.

**Testing** means testing our dataset, often using code, to make sure our cleaning operations worked.

## Define

Select one of the assesment, and define how to fix this issue. somewhat like shown below.

## Clean

### Issue 1

**Define**

**Code**

In [ ]:

**Test**

In [ ]:

### Issue 2

**Define**

**Code**

In [ ]:

**Test**

In [ ]:

After the cleaning of the data we reassess then iterate over every operation if necessary and may end the process if we are satisfied with the tidiness of the data.

**Points:-**

EDA: an analysis approach that focuses on identifying general patterns in the data, and identifying outliers and features of the data that might not have been anticipated.

## ETL

You also may have heard of the extract-transform-load process also known as **ETL**. ETL differs from data wrangling in three main ways:

1. The users are different
2. The data is different
3. The use cases are different

## Chapter 2 : Gathering Data

First Step into the data wrangling process.

Rotten Tomatoes Top 10 movies of all time that the topic we'll be working on.

### **Flat file structure:**

Flat files contain tabular data in plain text format with one data record per line and each record or line having one or more fields. These fields are separated by delimiters, like commas, tabs, or colons.

**Advantages of flat files** include:

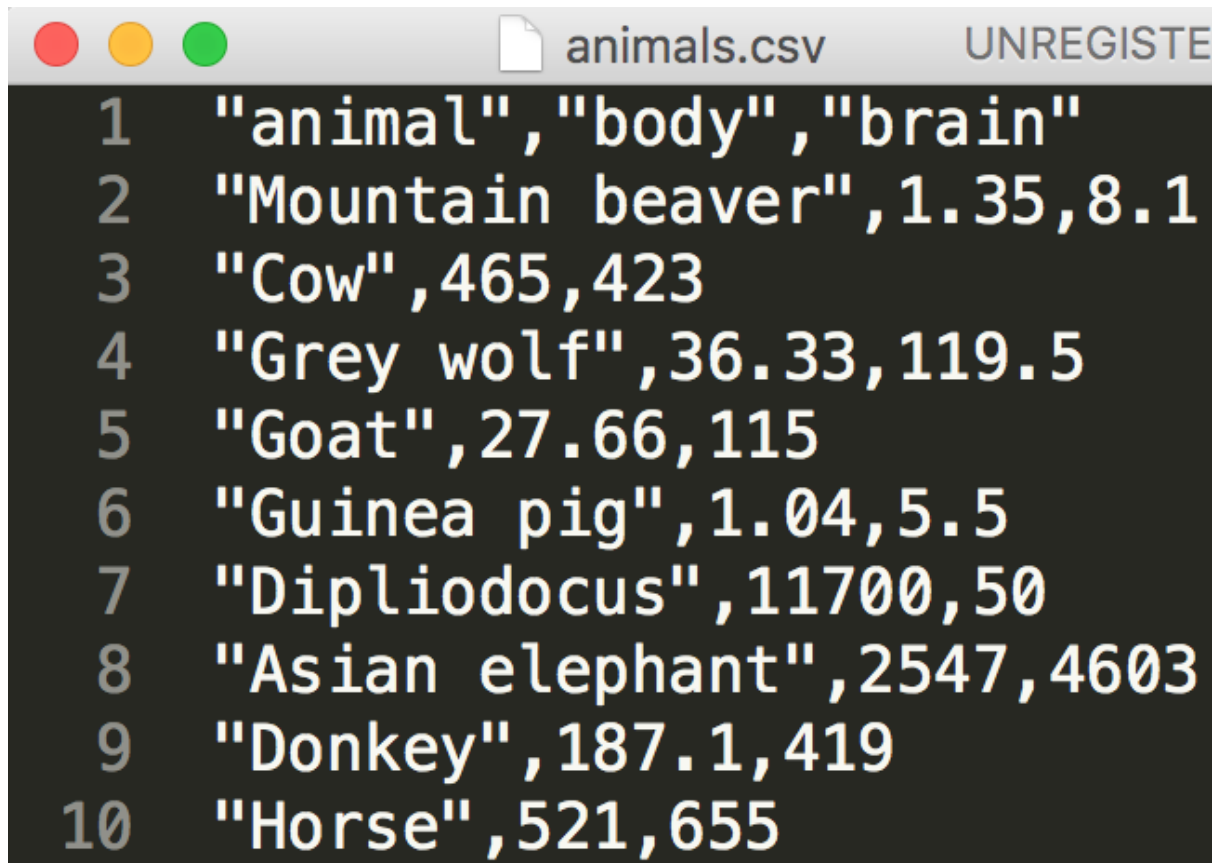
- They're text files and therefore human readable.
- Lightweight.
- Simple to understand.
- Software that can read/write text files is ubiquitous, like text editors.
- Great for small datasets.

**Disadvantages of flat files**, in comparison to relational databases, for example, include:

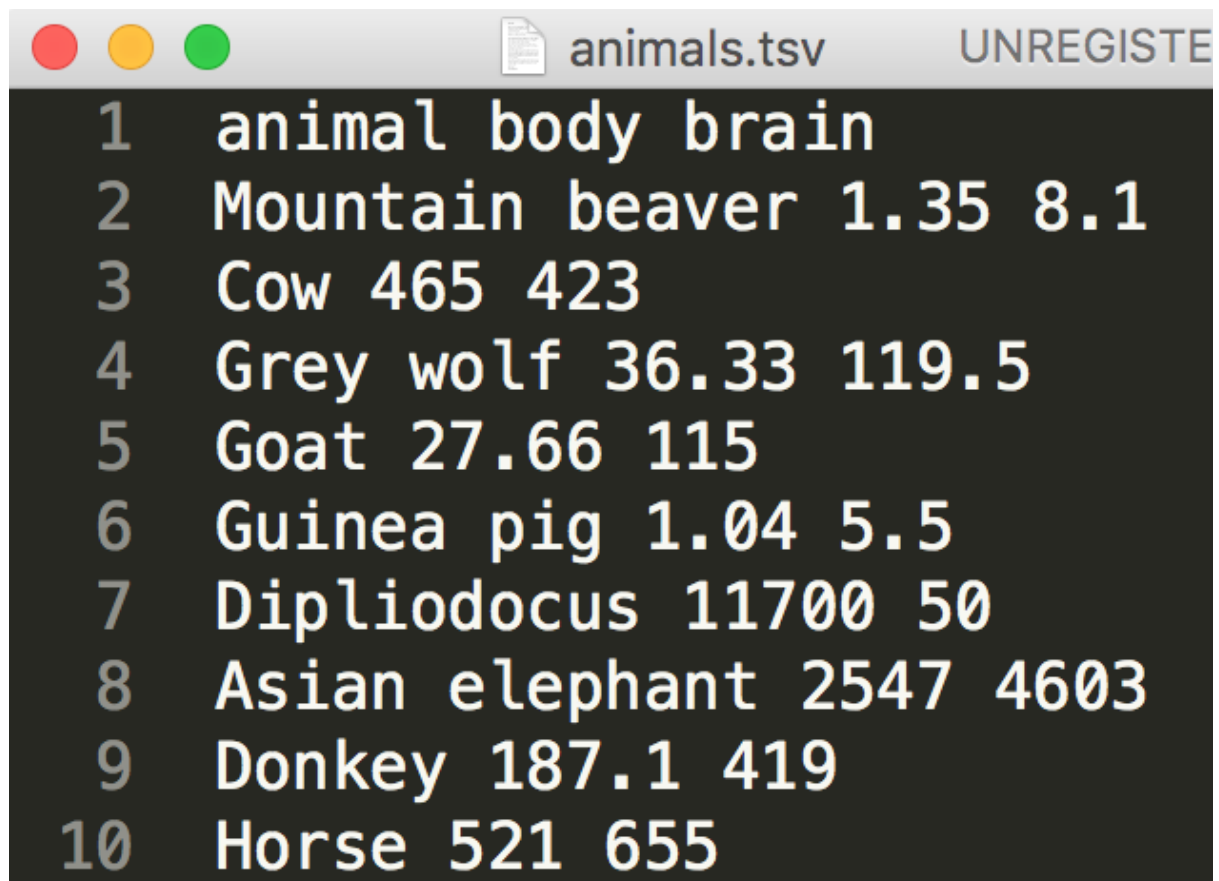
- Lack of standards.
- Data redundancy.

- Sharing data can be cumbersome.
- Not great for large datasets (see *"When does small become large?"* in the Cornell link in *More Information*).

Examples of flat files

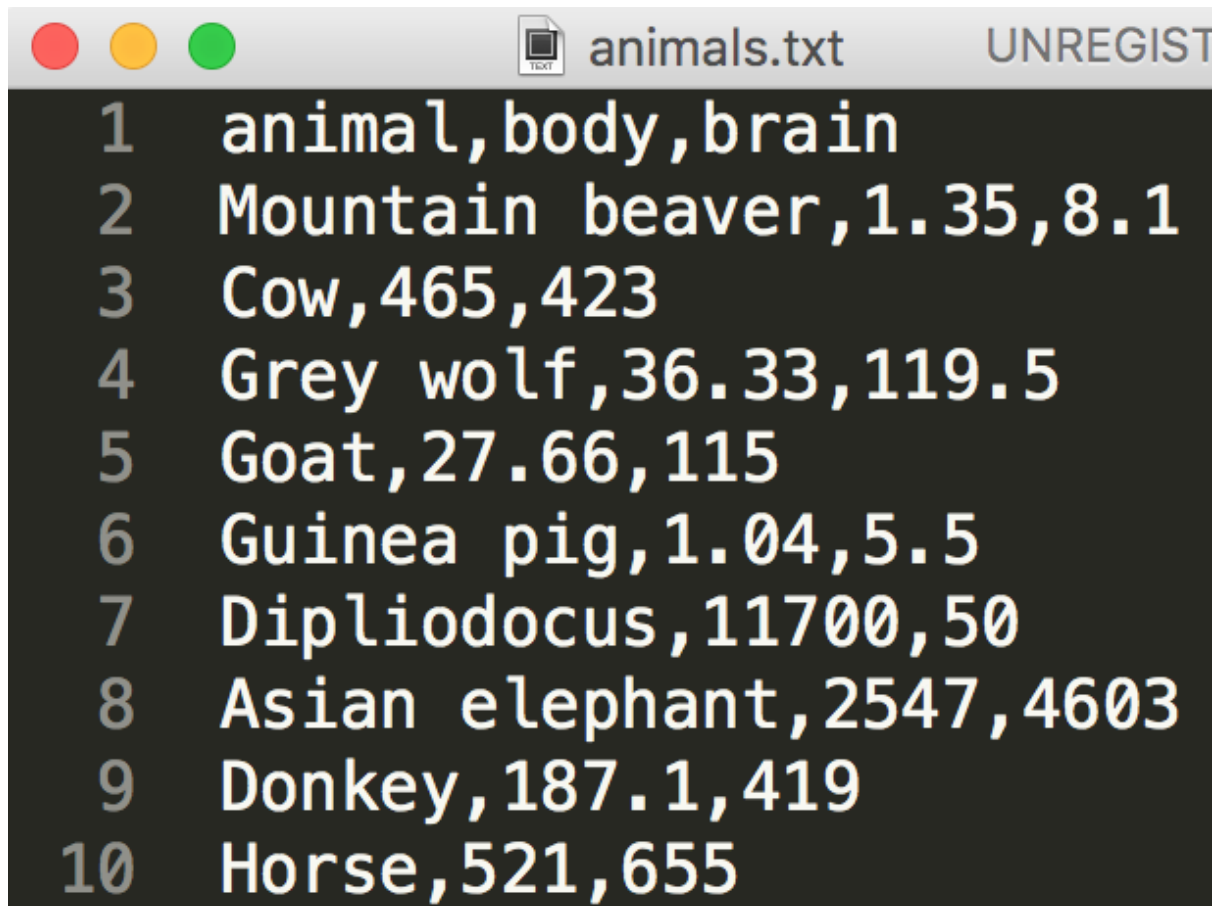


```
animals.csv  UNREGISTE
1  "animal","body","brain"
2  "Mountain beaver",1.35,8.1
3  "Cow",465,423
4  "Grey wolf",36.33,119.5
5  "Goat",27.66,115
6  "Guinea pig",1.04,5.5
7  "Dipliodocus",11700,50
8  "Asian elephant",2547,4603
9  "Donkey",187.1,419
10 "Horse",521,655
```



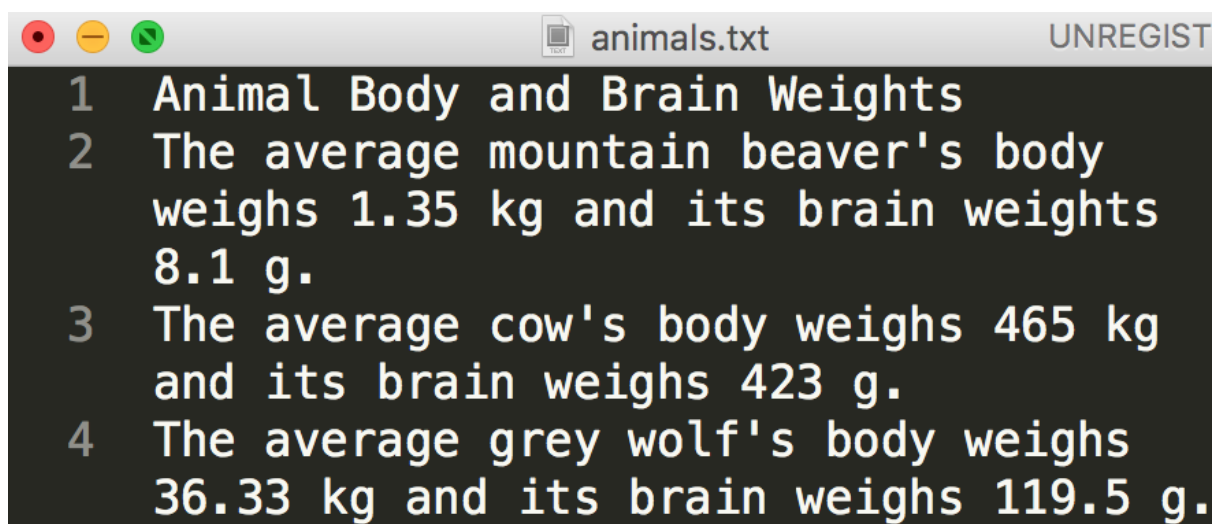
	animal	body	brain
1	Mountain beaver	1.35	8.1
2	Cow	465	423
3	Grey wolf	36.33	119.5
4	Goat	27.66	115
5	Guinea pig	1.04	5.5
6	Dipliodocus	11700	50
7	Asian elephant	2547	4603
8	Donkey	187.1	419
9	Horse	521	655
10			





```
1 animal,body,brain
2 Mountain beaver,1.35,8.1
3 Cow,465,423
4 Grey wolf,36.33,119.5
5 Goat,27.66,115
6 Guinea pig,1.04,5.5
7 Dipliodocus,11700,50
8 Asian elephant,2547,4603
9 Donkey,187.1,419
10 Horse,521,655
```

Example of not flat file



```
1 Animal Body and Brain Weights
2 The average mountain beaver's body
  weighs 1.35 kg and its brain weights
  8.1 g.
3 The average cow's body weighs 465 kg
  and its brain weighs 423 g.
4 The average grey wolf's body weighs
  36.33 kg and its brain weighs 119.5 g.
```

## Flat Files in python

in python we use pandas to handle file and datasets , pandas has a generalized approach towards flat files , one can simply use **read\_csv()**

function to read all kinds of flat files just by setting the **sep** argument = the separator used.

## Web Scrapping

It's the process of getting of extracting data from HTML pages using parsers and scripts

The two main ways to work with HTML files are:

- Saving the HTML file to your computer (using the **Requests** library for example) library and reading that file into a **BeautifulSoup** constructor
- Reading the HTML response content directly into a **BeautifulSoup** constructor (again using the Requests library for example)

## Text Files in Python

glob : The glob module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell, although results are returned in arbitrary order. No tilde expansion is done, but \*, ?, and character ranges expressed with [] will be correctly matched.

for creating a pandas dataframe the best method is to create a list and then append each feature using dictionary like `df_list.append({'title':title, 'var_name':var_value ...})`

then this list can be converted to dataframe

So scrapping is fun and labourous. but the point here is if the data is not available or the some features are missing then scrapping should only used when necessary , otherwise the use of APIs should be preferred. because they are easier and time saving.

APIs - Application programming interface:

here they are used to get data from different sites and organisations for educational purposes.

we are fetching data from wikipedia using wptools.

JSON DATA TYPE	PYTHON DATA STRUCTURE
JSON array	Python list
JSON object	Python dictionary

APIs that return JSON response in python the entire response can be treated like a dictionary in python and the items inside in the dictionary can be treated as lists.

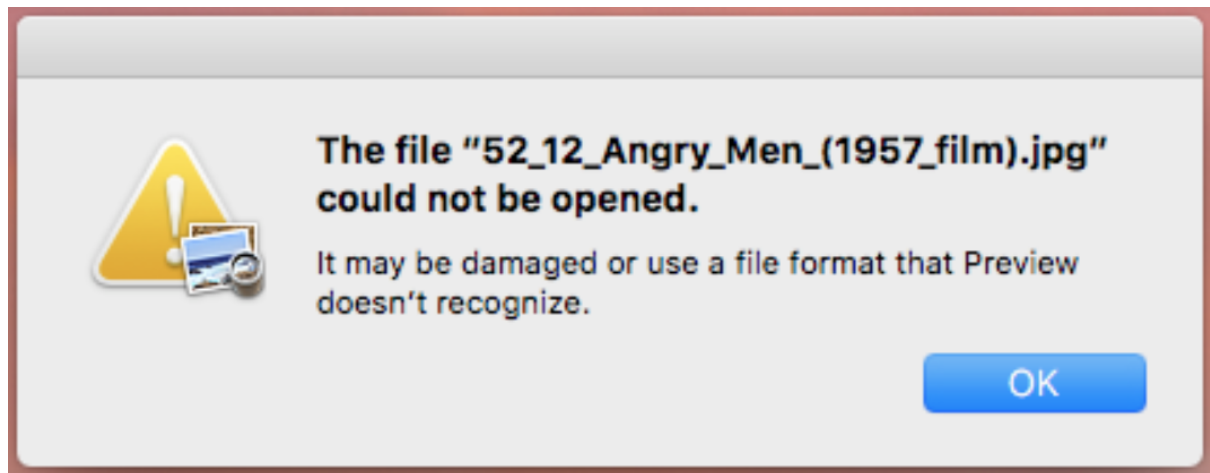
pandas also has JSON functions (the `read_json` function and the `to_json` DataFrame method), but the hierarchical advantage of JSON is wasted in pandas' tabular DataFrame so the uses are limited.

## Downloading Image Files

Downloading images may seem tricky from a reading and writing perspective, in comparison to text files which you can read line by line, for example. But in reality, image files aren't special—they're just binary files. To interact with them, you don't need special software (like Photoshop or something) that "understands" images. You can use regular file opening, reading, and writing techniques, like this:

```
import requests
r = requests.get(url)
with open(folder_name + '/' + filename, 'wb') as f:
    f.write(r.content)
```

But this technique can be error-prone. It will work most of the time, but sometimes the file you write to will be damaged. This happened to me when preparing this lesson:



This type of error is why the *requests* library maintainers recommend using the *PIL* library (short for Pillow) and `BytesIO` from the *io* library for non-text requests, like images. They recommend that you access the response body as bytes, for non-text requests. For example, to create an image from binary data returned by a request:

```
import requests
from PIL import Image
from io import BytesIO
r = requests.get(url)
i = Image.open(BytesIO(r.content))
```

Though you may still encounter a similar file error, this code above will at least warn us with an error message, at which point we can manually download the problematic images.

## Storing Data

Storing is usually done after cleaning, but it's not always done, which excludes it from being a core part of the data wrangling process. Sometimes you just analyze and visualize and leave it at that, without saving your new data.

sometimes the data is stored into file or sometimes it is stored into databases which are then used for fetching data.

SQL - Structured Query Language, advantages :- easy to replicate, and can work on large data.

It can help in answering more deeper and complex questions.

## Why databases ?

- ▼ provides storage.
- ▼ provides concurrency
- ▼ data integrity
- ▼ much faster than flat files

## How relational Databases store data ?

Tables - rows and columns

Database tables are organized by columns and each column has a unique name. and the datatype for every value in that column is same.

## Types of SQL statements

Create - creates a table in database.

Drop - delete a table or column.

Select - read data and display it.

## Relational Databases in Python

Data Wrangling and Relational Databases

In the context of data wrangling, we recommend that databases and SQL only come into play for gathering data or storing data. That is:

- **Connecting to a database and importing data** into a pandas DataFrame (or the analogous data structure in your preferred programming language), then assessing and cleaning that data, or
- **Connecting to a database and storing data** you just gathered (which could potentially be from a database), assessed, and cleaned

These tasks are especially necessary when you have large amounts of data, which is where SQL and other databases excel over flat files.

The two scenarios above can be further broken down into three main tasks:

- Connecting to a database in Python
- Storing data **from** a pandas DataFrame **in** a database to which you're connected, and
- Importing data **from** a database to which you're connected **to** a pandas DataFrame

## Other File Formats

The types of files you mastered in this lesson are the ones you'll interact with for the vast majority of your wrangling projects in the future. Again, these were:

- Flat files (e.g. CSV and TSV)
- HTML files
- JSON files
- TXT files
- Relational database files

Additional, less common file formats include:

- Excel files
- Pickle files
- HDF5 files
- SAS files
- STATA files

pandas has functions to read (and write, to most of them) these files. Also, you now have the foundational understanding of **gathering** and file formats in general, so learning these additional formats won't be too hard if you need them.