# DATA LITERACY

The term “data” typically means any information that is stored in a **structured format** (think a table with rows and columns), however when requesting data it’s important to clarify what you want.

It’s also important to know how data are collected and specifically what is collected. There are lots of ways data could be collected. Here are a few:

- A person fills out a paper form and another person types that into a database or spreadsheet
- A person fills out an electronic online form and the information automatically flows into a database or spreadsheet
- A person types it straight into a database or spreadsheet
- One government agency sends tabular data to another (i.e. local school districts sending test score results to state education agency)

Structured data should follow some basic principles.

- First, **each row should be one observation**. This might be one incident, one person, one place. Whatever it is, it needs to be consistent. You don’t want a dataset that has one row for an incident and then suddenly a row for a person (or anything other than an incident). You might, however, have a dataset with one row for each person involved in an incident – perhaps a car crash. And in this scenario, there might be multiple records that apply to one incident. However, still each row represents one person.
- Second, **each column (or field) will contain characteristics about that one thing**. And each column should contain a single characteristic. For example, let’s say you have death certificate data. Each row is one death certificate (or you can think of this as one person who died). The columns contain information about that death that was included in the certificate – first name, last name, date of birth, date of death, cause of death, residence city, place of death, etc.
- Third, the **data should be a rectangle**, without any empty rows or columns stuck in between.
- Fourth, **there should be a header row** with simple, but clear labels. Try to avoid naming columns with spaces. An alternative is to use underscore (“first_name”) or camel case (“FirstName”).

Sometimes data that goes together is stored in separate tables – these are called “**relational databases.”** For example, restaurant inspections data oftentimes have one table with basic information about the restaurant, with one record for each restaurant (i.e. address, date opened, name of owner, contact information). Then they probably have another data table with one record for each inspection at each restaurant. So if a restaurant has been inspected 5 times since the database was created, there would be five records for that restaurant – with the date of the inspection, name of inspector and possibly some other aggregate information (like number of violations). Then there is probably a third table with one record for each violation from each inspection. If that restaurant had 2 violations from each of those 5 inspections, then there would be 10 violation records. All three tables will have an ID number field that represents the restaurant, so that you can "join" the tables together easily.

Typically, data is stored in a way so that each column has one piece of information.

For example, if you wanted to build a database of information about your classmates you might structure a database like this….Each row would represent one student. These might be the columns: first name, middle name, last name, hometown, major, minor, age, date of birth, year in school, phone number, email address.

Notice that there are separate columns for last name, middle name and first name. This is the best way to keep information so that you can easily sort on any one of those things. Same with major and minor. Having them separate enables you to quickly find (either through sorting or filtering) which students are French minors, regardless of what their major is.

These columns then are set up in the database or spreadsheet software according to their "type." Defining the correct type is crucial for being able to sort the data properly and being able to do mathematical calculations.

Here are the most **common field types**:

**Text** – a sequence of letters, numbers, punctuation and other printable characters including white space. This field type allows you to input virtually anything. So you could store numbers in here or dates, but it wouldn't allow you to do mathematical calculations on the numbers, nor would it sort the dates properly.

**Date** – a representation of month/day/year. You can change how this is displayed so it could look like "03/22/1990" or it could be "March 22, 1990" or it could be European format of "22/3/1990", etc. Usually these fields allow you to have a date/time combination as well, such as "03/22/1990 1:00 PM". But you can't put anything other than a date/time in any date format field, and you can't have more than one date in there. For example, you can't do "March 22, 1990 to March 23, 1990" and you can't do something like "approximately march 22, 1990" or "late in the evening of March 22"

**Number** – any numeric format, either with decimal places or without (although in some software you have to specify if you want to store decimal places). Like dates, this field type does not allow for variation. You can't have more than one number or any letters or other characters.

**ID numbers** --- if you have a field that stores an ID number – such as an employee ID or a school district number – you would set that as text, especially if those IDs have leading zeros. You only want to use number format if you think you might do mathematical calculations on it. ZIP codes are another example (ZIPs on the East Coast have leading zeroes!).

Some software will also have other field types….for hyperlinks, long text (that exceeds the 255 byte limit), images, etc.

And most database software will often have multiple types of number fields or text fields or date fields. For example, in Microsoft Access we can have “short integer” (think 1 to 10), “long integer” (bigger numbers, without decimal places), “double” (numbers with 2 or more decimal places). You might have a date type or a datetime type.

So, for our earlier example about building a database of your classmates almost all of the fields would be set as text. The exceptions would be date of birth (date format), and age (numeric format).

**Aggregate/summary data:** Reporters have long dealt with aggregate data. This would be something like a list of each type of crime and the number of incidents of each in the past year; the median sale price of homes in each city around the country; or anything similar that shows either the total number of something or an average or median.

Anything that has been aggregated like this SHOULD also have detailed data underlying it. For example, the police department should have a database that has one record for each crime.

The detailed data is always going to be more useful and flexible, if you can get it.

But sometimes you can’t get that detailed information, often due to public records laws. And there may be legitimate situations where a government agency only collects aggregate information.

Here’s an example of data where the aggregate is the only portion legally available under the public records law: In Minnesota you can get data showing the number and percentage of students in each school that were “proficient” on standardized tests, but you cannot get data showing specific scores for each student (even if they leave off all identifying information).

Census data is another good example….we can find out how many or what percentage of families in each place are headed by a single mother, but we can’t get data showing one record for each household and their various attributes that Census measures (at least not until the data is made public 72 years after it’s collected)

In these situations, government agencies often supply a lot of “cross tabulation” tables. For example, the Census Bureau has a table called “household type by tenure” that will show how many households headed by a single parent rent their homes, versus own their homes, and then how many households with a married couple rent versus own, etc.

If the detailed data you want is not publically available, it’s likely that the agency is required to provide you aggregate or summary data. Asking for a series of cross tabulations might get you close to what you need.

**Some data terminology that is useful to know:**

**Field**: The columns. Another way to think about this is that each field represents a separate piece of information for each item stored in the database. It’s best to only have one piece of information stored in a given field.

**Record**: The horizontal rows. Each line should represent a distinct thing. For example, one player or one team or one year.

**Byte**: This is how data is measured. A field that has the word "Joplin" and nothing else in it takes up 6 bytes. A field that has "Joplin, MO" takes up 10 bytes (including the space and the comma). Most database programs won't allow you to have columns that hold more than 255 bytes.

**ASCII text:** This is like a Notepad file. It is plain text – no formatting. Every computer program should be able to dump their database out as ASCII text, with some form of delimiter to indicate where the field breaks are. When you're asking for data from someone, asking for "plain text" or "ASCII" (pronounced… ass-key) is the easiest method for them to get the data to you. First….every program should be able to output a "copy" of their data in ASCII. Second, because it doesn't have any formatting, an ASCII file is very small and much easier to transport (via e-mail, CD-ROM, etc). ASCII stands for American standard code for information interchange – although you'll never need to know that!

ASCII data files will either be **delimited** or **fixed width**.

**Delimiters**: The method used to delineate field breaks when data is in text format.
**Comma-Delimited:** the breaks are indicated by a comma. Also might be called comma-separated value or CSV.
Can also have data delimited by some other mark such as carrots (^) or semi-colons (;) or pipes (|).
**Tab-delimited**: Breaks are indicated by a tab.

**Fixed-width:** Each field is a set number of spaces, no matter how many the data takes up. With this format, it's essential to get a record layout indicating the size of each field.

**Database or Dbase File**: The data is stored in rows and columns and these files can be opened in Excel or Access. It is essentially the "table" that you find in Access.

**JSON:** JavaScript Open Notation is an open-standard file format that presents the data in a human readable format that is intended for transmitting data. It's often used in web applications.

**XML:** Extensible Markup Language is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

**Shapefile:** A popular format for storing geographic/GIS data; this is how most government agencies store any geographic data they maintain. A shapefile actually consists of a group of files, all with the same name but different extensions. You will need a GIS software, such as ArcGIS or QGIS, to open a shapefile. The online mapping tool called Carto will open a shapefile, as well.

**KML**: Another format for geographic/GIS data. Keyhole Markup Language is an XML notation for expressing geographic annotation and visualization within Internet-based, two-dimensional maps and three-dimensional Earth browsers

**Append** – When you add rows with new data to an existing dataset (and all the columns match). For example, let's say that last year you got a data file with the school test scores from the whole state. This year, you got the new batch. All the columns are the same (including with field names spelled the same and in the same order). Then there are various ways to add that new file to the bottom of that old file.

**Join** – This is when you match rows in one table to rows in another table. There are a variety of reasons to do this. One is that your dataset is a relational dataset and you need to join the tables together, another is that you want to marry data from one table to data in another table (known as an enterprise join).

**Concatenate** – Marry together values from two or more fields/columns into one new column. For example, let's say your data comes with a field for the house number, another for the street name and another for the street type (Ave, Ln, St., etc) and you need to have them all in one column. Every type of software has a way to have it merge the information from the 3 columns so they end up in one column together.

**Portable Media:** This refers to the physical storage mechanism that will be used to transfer the data from the agency computer to yours. The most common you will encounter are:

USB flash drive (a.k.a jump drive, flash drive)
CD-ROMs or DVD
9-track magnetic tape (looks like a giant movie reel) – very rare
Magnetic cartridges (looks like an 8-track tape) – very rare

However, these days you are far more likely to transfer data via the web – either downloading from a website or FTP server, or having it sent as an attachment in your email.

**FTP Server:** This stands for File Transfer Protocol. It's a program language that permits a user to transfer a file from one computer to another over the Internet. Many government agencies are now making data available through FTP – usually putting it in a location that is readily accessible all the time so you don't have to submit a formal request.

**Record Layout:** A listing of all the fields included in the database, including their size, type (character, numeric or date) and a short description.

**Codebook or Data dictionary**: Definitions of any codes used in the database. Codes are used to make data more consistent and to save space. For example, whether a person is male or female might be coded 1 for males and 2 for females. You need the codebook to determine which is which.  Sometimes the record layout and codes are all in one file and these terms – record layout, codebook, data dictionary – are used interchangeably.

By MaryJo Webster
@MaryJoWebster
Mjwebster71@gmail.com
Last updated: Aug 2018