**COURSE 4 MODULE 4**

When you clean data, you make changes to the original dataset. It’s important to verify the changes you make are accurate and to let your teammates know about the changes. In this part of the course, you’ll learn to verify that data is clean and report your data cleaning results. With verified clean data, you’re ready to begin analyzing!

### **Learning Objectives**

* Describe the process involved in verifying the results of cleaning data
* Describe what is involved in manually cleaning data
* Discuss the elements and importance of data-cleaning reports
* Describe the benefits of documenting data cleaning process

**MANUALLY CLEANING DATA**

[**VERIFY AND REPORT RESULTS**](https://www.coursera.org/learn/process-data/lecture/eKsSc/verify-and-report-results)

You've been learning a lot about the importance of clean data and explored some tools and strategies to help you throughout the cleaning process.

In these videos, we'll be covering the next step in the process: **verifying and reporting on the integrity of your clean data**.

**Verification** is a process to confirm that a data cleaning effort was well-executed and the resulting data is accurate and reliable.

It involves :

1. Rechecking your clean dataset, doing some manual clean ups if needed
2. Taking a moment to sit back and really think about the original purpose of the project.

That way, you can be confident that the data you collected is credible and appropriate for your purposes.

**Making sure your data is properly verified is so important because it allows you to double-check that the work you did to clean up your data was thorough and accurate.**

For example, you might have referenced an incorrect cell phone number or accidentally keyed in a typo. Verification lets you catch mistakes before you begin analysis. Without it, any insights you gain from analysis can't be trusted for decision-making. You might even risk misrepresenting populations or damaging the outcome of a product that you're actually trying to improve.

I remember working on a project where I thought the data I had was sparkling clean because I'd use all the right tools and processes, but when I went through the steps to verify the data's integrity, I discovered a semicolon that I had forgotten to remove. Sounds like a really tiny error, I know, but if I hadn't caught the semicolon during verification and removed it, it would have led to some big changes in my results. That, of course, could have led to different business decisions.

There's an example of **why verification is so crucial**. But that's not all.

The other big part of the **verification process** is reporting on your efforts. Open communication is a lifeline for any data analytics project. Reports are a super effective way to show your team that you're being 100 percent transparent about your data cleaning.

**Reporting is also a great opportunity to show stakeholders that you're accountable, build trust with your team, and make sure you're all on the same page of important project details**.

Coming up, you'll learn different strategies for reporting, like creating data-cleaning reports, documenting your cleaning process, and using something called the **changelog**.

A **changelog** is a file containing a chronologically ordered list of modifications made to a project.

It's usually organized by version and includes the date followed by a list of added, improved, and removed features.

**Changelogs are very useful for keeping track of how a dataset evolved over the course of a project.** They're also another great way to communicate and report on data to others. Along the way, you'll also see some examples of how verification and reporting can help you avoid repeating mistakes and save you and your team time.

Let's go!

[**CONFIRM DATA-CLEANING MEETS BUSINESS EXPECTATIONS**](https://www.coursera.org/learn/process-data/lecture/wmjwi/confirm-data-cleaning-meets-business-expectations)

Verification is a critical part of any analysis project. Without it you have no way of knowing that your insights can be relied on for data-driven decision-making. Think of verification as a stamp of approval.

To refresh your memory, **verification is a process to confirm that a data-cleaning effort was well-executed and the resulting data is accurate and reliable**. It also involves manually cleaning data to compare your expectations with what's actually present.

The first step in the verification process is going back to your original unclean data set and comparing it to what you have now.

Review the dirty data and try to identify any common problems.

For example, maybe you had a lot of nulls. In that case, you check your clean data to ensure no nulls are present. To do that, you could search through the data manually or use tools like **conditional formatting or filters**.

Or maybe there was a common misspelling like someone keying in the name of a product incorrectly over and over again. In that case, you'd run a **FIND in your clean data to make sure no instances of the misspelled word occur**.

Another key **part of verification** involves **taking a big-picture view of your project**. This is an opportunity to confirm you're actually focusing on the business problem that you need to solve and the overall project goals and to make sure that your data is actually capable of solving that problem and achieving those goals.

It's important to take the time to reset and focus on the big picture because projects can sometimes evolve or transform over time without us even realizing it. Maybe an e-commerce company decides to survey 1000 customers to get information that would be used to improve a product. But as responses begin coming in, the analysts notice a lot of comments about how unhappy customers are with the e-commerce website platform altogether. So the analysts start to focus on that. While the customer buying experience is of course important for any e-commerce business, it wasn't the original objective of the project. The analysts in this case need to take a moment to pause, refocus, and get back to solving the original problem.

**Taking a big picture view of your project involves doing three things.**

First, consider the business problem you're trying to solve with the data.If you've lost sight of the problem, you have no way of knowing what data belongs in your analysis. Taking a problem-first approach to analytics is essential at all stages of any project. You need to be certain that your data will actually make it possible to solve your business problem.

Second, you need to consider the goal of the project. It's not enough just to know that your company wants to analyze customer feedback about a product. What you really need to know is that the goal of getting this feedback is to make improvements to that product. On top of that, you also need to know whether the data you've collected and cleaned will actually help your company achieve that goal.

And third, you need to consider whether your data is capable of solving the problem and meeting the project objectives. That means thinking about where the data came from and testing your data collection and cleaning processes.

Sometimes data analysts can be too familiar with their own data, which makes it easier to miss something or make assumptions.

**Asking a teammate to review your data from a fresh perspective and getting feedback from others is very valuable in this stage.**

This is also the time to notice if anything sticks out to you as suspicious or potentially problematic in your data.

Again, step back, take a big picture view, and **ask yourself, do the numbers make sense?**

Let's go back to our e-commerce company example. Imagine an analyst is reviewing the cleaned up data from the customer satisfaction survey. The survey was originally sent to 1,000 customers, but what if the analyst discovers that there are more than a thousand responses in the data? This could mean that one customer figured out a way to take the survey more than once. Or it could also mean that something went wrong in the data cleaning process, and a field was duplicated. Either way, this is a signal that it's time to go back to the data-cleaning process and correct the problem.

**Verifying your data ensures that the insights you gain from analysis can be trusted. It's an essential part of data-cleaning that helps companies avoid big mistakes.** This is another place where data analysts can save the day.

[**STEP-BY-STEP: VERIFICATION OF DATA CLEANING**](https://www.coursera.org/learn/process-data/supplement/Bc9rE/step-by-step-verification-of-data-cleaning)

This reading outlines the steps the instructor performs in the following video, [Verification of data cleaning](https://www.coursera.org/learn/process-data/lecture/Hx69i/the-final-step-in-data-cleaning). The video demonstrates how to verify cleaned data in both spreadsheets and SQL.

Keep this step-by-step guide open as you watch the video. It can serve as a helpful reference if you need additional context or clarification while following the video steps. This is not a graded activity, but you can complete these steps to practice the skills demonstrated in the video.

### 

### **What you’ll need**

If you’d like to follow along with the examples in this video, choose a spreadsheet tool. Google Sheets or Excel are recommended.

To access the spreadsheet the instructor uses in this video, click the link to the template to create a copy of the dataset. If you don’t have a Google account, download the data directly from the attachments below.

Link to dataset: [Jeff’s Party Planet - Data for Cleaning](https://docs.google.com/spreadsheets/d/1RaDdSEp2V6D09FE6LOFkiJGv9CMT83GIV_U9YnY2rvI/template/preview?resourcekey=0-IU2-k90CX0mrt0ebwrvwDw)

OR

[Jeff's Party Planet - Data for Cleaning](https://d3c33hcgiwev3.cloudfront.net/ve64rJeHTTylDeO1R6V-YQ_3611b683e1844cb5a165b96ea68f71e1_Jeff-s-Party-Planet---Data-for-Cleaning.xlsx?Expires=1712448000&Signature=Ehd1j55thS33zMdEUZduMKYt3GAP7uULUse9ZdLmBtXbV11EVIdbuPYkjT3Oor72q2jzAgunYjGn1ymWZFVXraHR433eqaB0Pm-cVamx9RAy9cxDpzsM0xLu1UcvEfqVIGxPDHA1hAQ3EO1~GV4JfDY7Mm2lNOKAqG4DdN3AuF4_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A)

[XLSX File](https://d3c33hcgiwev3.cloudfront.net/ve64rJeHTTylDeO1R6V-YQ_3611b683e1844cb5a165b96ea68f71e1_Jeff-s-Party-Planet---Data-for-Cleaning.xlsx?Expires=1712448000&Signature=Ehd1j55thS33zMdEUZduMKYt3GAP7uULUse9ZdLmBtXbV11EVIdbuPYkjT3Oor72q2jzAgunYjGn1ymWZFVXraHR433eqaB0Pm-cVamx9RAy9cxDpzsM0xLu1UcvEfqVIGxPDHA1hAQ3EO1~GV4JfDY7Mm2lNOKAqG4DdN3AuF4_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A)

**Note:** The SQL table used in this example is not available for this activity.

## 

## **Example 1: Verify data with spreadsheets**

Use spreadsheet tools such as Find and Replace and pivot tables to find, understand, and fix errors in your spreadsheet.

### **Use Find and Replace to replace all instances of a mistake**

1. Use the [Jeff’s Party Planet - Data for Cleaning](https://docs.google.com/spreadsheets/d/1RaDdSEp2V6D09FE6LOFkiJGv9CMT83GIV_U9YnY2rvI/template/preview?resourcekey=0-IU2-k90CX0mrt0ebwrvwDw#gid=0) dataset.
2. From the **Edit** menu, choose **Find and Replace** to open the **Find and replace** dialog box.
3. In the **Find** field, enter the misspelled word in the supplier name, **Plos**.
4. In the **Replace with** field, enter **Plus**.
5. Click **Replace all** to replace all instances of "Plos" with "Plus". Click **Done** to close the **Find and replace** dialog box.
6. Select the **Undo** button to use a different method to correct this misspelling. This can also be done with **Ctrl** (Windows) or **Command** (Mac) **Z**.

### **Use a pivot table to understand errors in a spreadsheet**

1. Select the **Suppliers** column.
2. Select **Insert > Pivot Table**. In the **Create pivot table** dialog box, choose **New Sheet** then **Create**.
3. This creates a new tab that is mostly blank.
4. Additionally, the **Pivot table editor** pane is in the window.
5. Next to **Rows**. Select **Add**, then the **Suppliers** column.
6. Next to **Values**, select **Add** then select **Suppliers**. This adds a value for the **Suppliers** column.
7. By default, Google Sheets sets the value to summarize by **COUNTA** (the total number of values in a range). This will show how many times each supplier name comes up. It’s a great way to check for misspellings and other anomalies. **Note:** Don’t use **COUNT**, because **COUNT** counts only numerical values.
8. When there is only one instance of the misspelled name, manually change it to the correct spelling.
9. To return to the original sheet, select the **Sheet1** tab.

## **Example 2: Use a CASE statement to verify data in SQL**

Use **CASE** statements to correct misspellings in SQL.

1. The SQL table used in this example is not available for download, but if you were performing a similar query you’d first make sure to load the data in BigQuery.

2. Start your SQL query with the basic structure:

**SELECT**

**FROM**

**WHERE**

3. In the **FROM** clause, specify the table you're pulling data from after **FROM**. For example, **project-id.customer\_data.customer\_name**

4. In the **SELEC**T clause, specify the columns you want to return. In this example, you want **customer\_id** and **first\_name**.

5. However, there is a misspelling in a customer’s first name.

i. To correct the misspelled name "Tnoy" to "Tony", use a **CASE** statement.

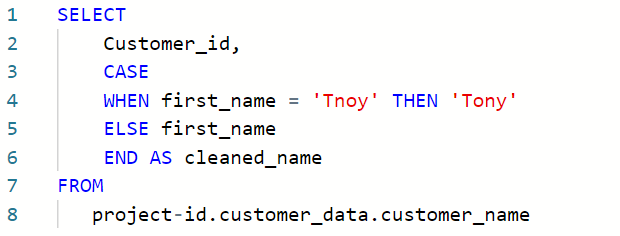
ii. Enter **CASE**. On the next line, enter **WHEN first\_name = 'Tnoy'THEN 'Tony'**. This tells SQL to replace any instances of **Tnoy** in the **first\_name** column with **Tony**.

iii. On the next line, add the **statement ELSE first\_name** to keep other names as they are.

iv. End the statement with **END AS cleaned\_name**.This creates a new column called **cleaned\_name** that will contain the data cleaned with the **CASE** statement.

6. Delete the **WHERE** clause because you don’t want to filter the query.

7. The final statement should be:



8. This SQL query will correct the misspelled name and leave other names unchanged in a new column called **cleaned\_name**. Note that this query corrects only the display of the name; it does not update the table’s data.

[**VERIFICATION OF DATA CLEANING**](https://www.coursera.org/learn/process-data/lecture/Hx69i/verification-of-data-cleaning)

This topic is just the video of Sally doing the same as described on the precious topic.

[**DATA-CLEANING VERIFICATION CHECKLIST**](https://www.coursera.org/learn/process-data/supplement/c9vis/data-cleaning-verification-checklist)

This reading will give you a checklist of common problems you can refer to when doing your data cleaning verification, no matter what tool you are using. When it comes to data cleaning verification, there is no one-size-fits-all approach or a single checklist that can be universally applied to all projects. Each project has its own organization and data requirements that lead to a unique list of things to run through for verification.



Keep in mind, as you receive more data or a better understanding of the project goal(s), you might want to revisit some or all of these steps.

## **Correct the most common problems**

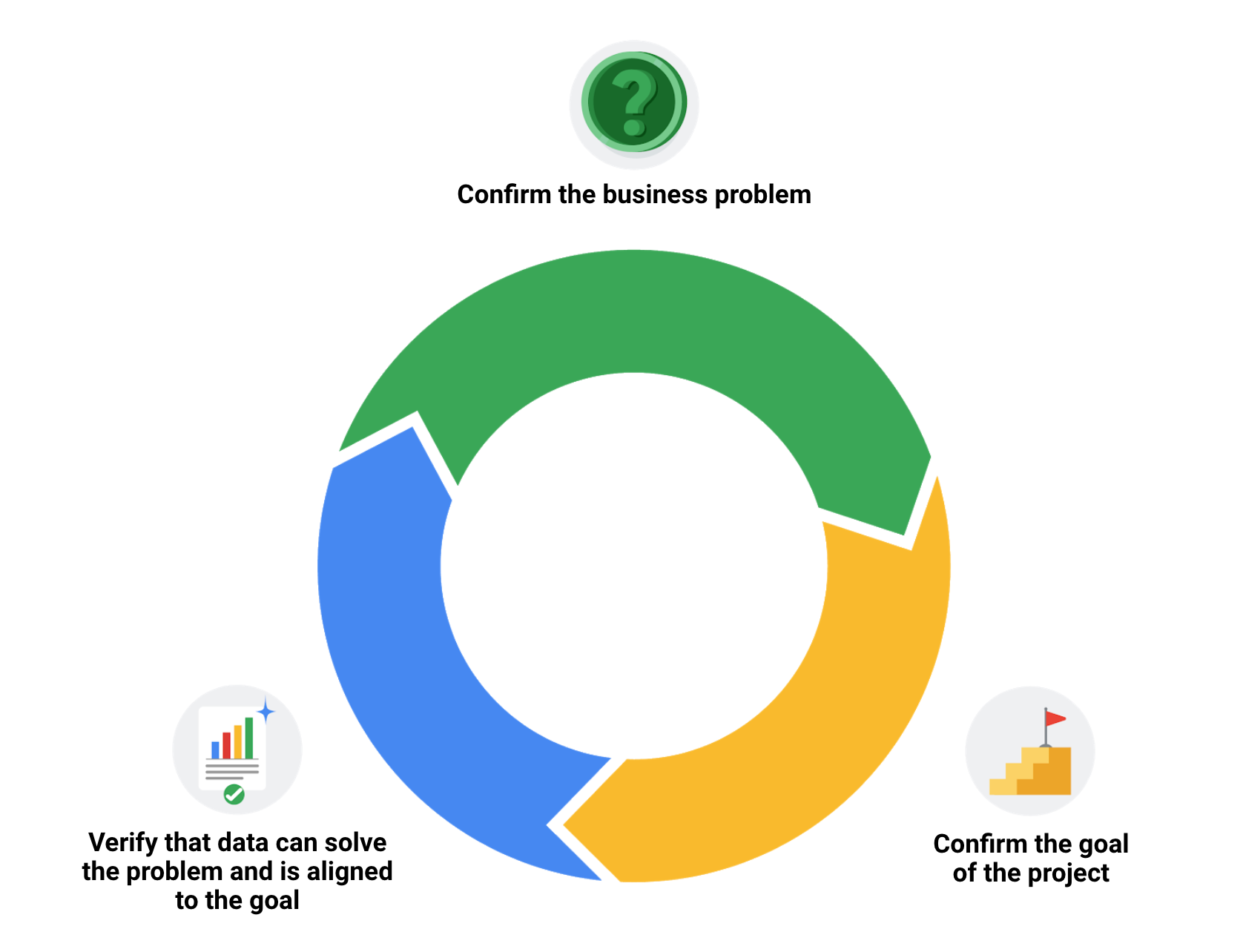
Make sure you identified the most common problems and corrected them, including:

* **Sources of errors**: Did you use the right tools and functions to find the source of the errors in your dataset?
* **Null data**: Did you search for NULLs using conditional formatting and filters?
* **Misspelled words**: Did you locate all misspellings?
* **Mistyped numbers**: Did you double-check that your numeric data has been entered correctly?
* **Extra spaces and characters**: Did you remove any extra spaces or characters using the **TRIM** function?
* **Duplicates**: Did you remove duplicates in spreadsheets using the **Remove Duplicates** function or **DISTINCT** in SQL?
* **Mismatched data types**: Did you check that numeric, date, and string data are typecast correctly?
* **Messy (inconsistent) strings**: Did you make sure that all of your strings are consistent and meaningful?
* **Messy (inconsistent) date formats**: Did you format the dates consistently throughout your dataset?
* **Misleading variable labels (columns)**: Did you name your columns meaningfully?
* **Truncated data:** Did you check for truncated or missing data that needs correction?
* **Business Logic**: Did you check that the data makes sense given your knowledge of the business?

## **Review the goal of your project**

Once you have finished these data cleaning tasks, it is a good idea to review the goal of your project and confirm that your data is still aligned with that goal. This is a continuous process that you will do throughout your project-- but here are three steps you can keep in mind while thinking about this:

* Confirm the business problem
* Confirm the goal of the project
* Verify that data can solve the problem and is aligned to the goal



**DOCUMENT THE CLEANING PROCESS**

[**CAPTURE CLEANING CHANGES**](https://www.coursera.org/learn/process-data/lecture/jjOrO/capture-cleaning-changes)

Now that you've learned how to make your data squeaky clean, it's time to address all the dirt you've left behind.

**When you clean your data, all the incorrect or outdated information is gone, leaving you with the highest-quality content**.

**But all those changes you made to the data are valuable too.**

We'll discuss why **keeping track of changes is important to every data project** and how to document all your cleaning changes to make sure **everyone stays informed**.

This involves documentation which is the process of tracking changes, additions, deletions and errors involved in your data cleaning effort.

You can think of it like a crime TV show. Crime evidence is found at the scene and passed on to the forensics team. They analyze every inch of the scene and document every step, so they can tell a story with the evidence. A lot of times, the forensic scientist is called to court to testify about that evidence, and they have a detailed report to refer to. The same thing applies to data cleaning.

Data errors are the crime, data cleaning is gathering evidence, and documentation is detailing exactly what happened for peer review or court.

**Having a record of how a data set evolved does three very important things.**

**First**, it lets us recover data-cleaning errors. Instead of scratching our heads, trying to remember what we might have done three months ago, we have a cheat sheet to rely on if we come across the same errors again later. It's also a good idea to create a clean table rather than overriding your existing table. This way, you still have the original data in case you need to redo the cleaning.

**Second**, documentation gives you a way to inform other users of changes you've made. If you ever go on vacation or get promoted, the analyst who takes over for you will have a reference sheet to check in with.

**Third**, documentation helps you to determine the quality of the data to be used in analysis. The first two benefits assume the errors aren't fixable. But if they are, a record gives the data engineer more information to refer to. It's also a great warning for ourselves that the data set is full of errors and should be avoided in the future. If the errors were time-consuming to fix, it might be better to check out alternative data sets that we can use instead.

Data analysts usually use a changelog to access this information. As a reminder, **a changelog is a file containing a chronologically ordered list of modifications made to a project**. You can use and view a changelog in spreadsheets and SQL to achieve similar results.

Let's start with the **spreadsheet**.

We can use Sheet's version history, which provides a real-time tracker of all the changes and who made them from individual cells to the entire worksheet. To find this feature, click the File tab, and then select Version history. In the right panel, choose an earlier version. We can find who edited the file and the changes they made in the column next to their name.

To return to the current version, go to the top left and click "Back." If you want to check out changes in a specific cell, we can right-click and select Show Edit History. Also, if you want others to be able to browse a sheet's version history, you'll need to assign permission.

Now let's switch gears and talk about **SQL**.

The way you create and view a changelog with SQL depends on the software program you're using. Some companies even have their own separate software that keeps track of changelogs and important SQL queries. This gets pretty advanced. Essentially, all you have to do is specify exactly what you did and why when you commit a query to the repository as a new and improved query. This allows the company to revert back to a previous version if something you've done crashes the system, which has happened to me before. Another option is to just add comments as you go while you're cleaning data in SQL. This will help you construct your changelog after the fact. For now, we'll check out query history, which tracks all the queries you've run.

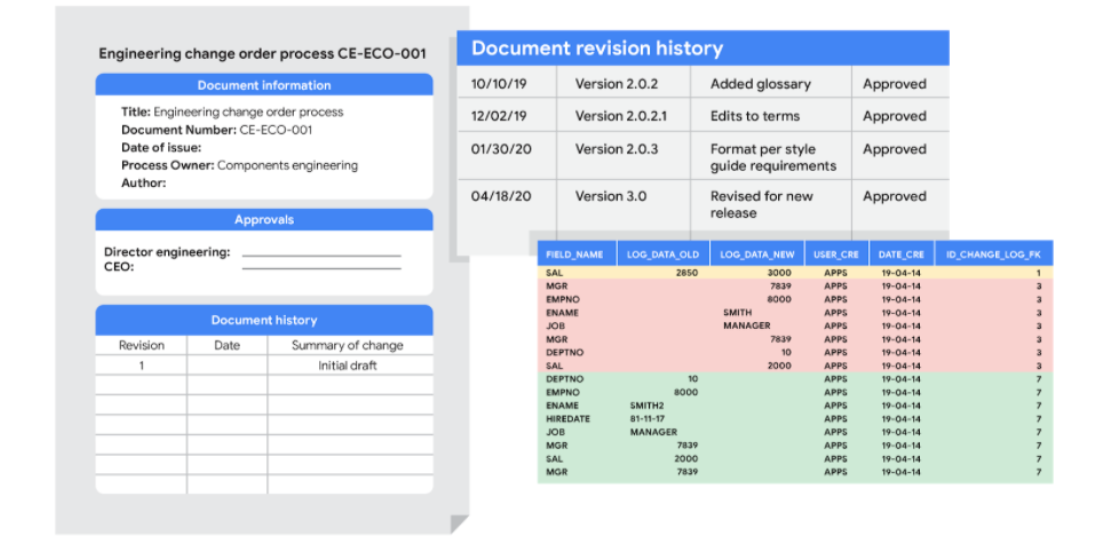
You can click on any of them to revert back to a previous version of your query or to bring up an older version to find what you've changed. Here's what we've got. I'm in the Query history tab. Listed on the bottom right are all the queries that run by date and time. You can click on this icon to the right of each individual query to bring it up to the Query editor.

Changelogs like these are a great way to keep yourself on track. It also lets your team get real-time updates when they want them. But there's another way to keep the communication flowing, and that's reporting. Stick around, and you'll learn some easy ways to share your documentation and maybe impress your stakeholders in the process.

[**EMBRACE CHANGELOGS**](https://www.coursera.org/learn/process-data/supplement/FvuSF/embrace-changelogs)

What do engineers, writers, and data analysts have in common? Change.

Engineers use **engineering change orders** (ECOs) to keep track of new product design details and proposed changes to existing products. Writers use **document revision histories** to keep track of changes to document flow and edits. And data analysts use **changelogs** to keep track of data transformation and cleaning. Here are some examples of these:



## **Automated version control takes you most of the way**

Most software applications have a kind of history tracking built in. For example, in Google sheets, you can check the version history of an entire sheet or an individual cell and go back to an earlier version. In Microsoft Excel, you can use a feature called **Track Changes**. And in BigQuery, you can view the history to check what has changed.

Here’s how it works:

| Google Sheets | 1. Right-click the cell and select **Show edit history**.  2. Click the left-arrow < or right arrow > to move backward and forward in the history as needed. |
| --- | --- |
| Microsoft Excel | 1. If Track Changes has been enabled for the spreadsheet: click **Review**.  2. Under **Track Changes**, click the **Accept/Reject Changes** option to accept or reject any change made. |
| BigQuery | Bring up a previous version (without reverting to it) and figure out what changed by comparing it to the current version. |

## **Changelogs take you down the last mile**

A **changelog** can build on your automated version history by giving you an even more detailed record of your work. This is where data analysts record all the changes they make to the data. Here is another way of looking at it. Version histories record *what* was done in a data change for a project, but don't tell us *why*. Changelogs are super useful for helping us understand the reasons changes have been made. Changelogs have no set format and you can even make your entries in a blank document. But if you are using a shared changelog, it is best to agree with other data analysts on the format of all your log entries.

Typically, a changelog records:

* Data, file, formula, query, or any other component that changed
* Description of what changed
* Date of the change
* Person who made the change
* Person who approved the change
* Version number
* Reason for the change

Let’s say you made a change to a formula in a spreadsheet because you observed it in another report and you wanted your data to match and be consistent. If you found out later that the report was actually using the wrong formula, an automated version history would help you *undo* the change. But if you also recorded the reason for the change in a changelog, you could go back to the creators of the report and let them know about the incorrect formula. If the change happened a while ago, you might not remember who to follow up with. Fortunately, your changelog would have that information ready for you! By following up, you would ensure data integrity outside your project. You would also be showing personal integrity as someone who can be trusted with data. That is the power of a changelog!

Finally, a changelog is important for when lots of changes to a spreadsheet or query have been made. Imagine an analyst made four changes and the change they want to revert to is change #2. Instead of clicking the undo feature three times to undo change #2 (and losing changes #3 and #4), the analyst can undo just change #2 and keep all the other changes. Now, our example was for just 4 changes, but try to think about how important that changelog would be if there were hundreds of changes to keep track of.

## **Bonus tip**

****

If an analyst is making changes to an existing SQL query that is shared across the company, the company most likely uses what is called a **version control system**. An example might be a query that pulls daily revenue to build a dashboard for senior management.

Here's how a version control system affects a change to a query:

1. A company has official versions of important queries in their **version control system**.
2. An analyst makes sure the most up-to-date version of the query is the one they will change. This is called **syncing**
3. The analyst makes a change to the query.
4. The analyst might ask someone to review this change. This is called a **code review** and can be informally or formally done. An informal review could be as simple as asking a senior analyst to take a look at the change.
5. After a reviewer approves the change, the analyst submits the updated version of the query to a repository in the company's version control system. This is called a **code commit**. A best practice is to document exactly what the change was and why it was made in a comments area. Going back to our example of a query that pulls daily revenue, a comment might be: *Updated revenue to include revenue coming from the new product, Calypso*.
6. After the change is **submitted**, everyone else in the company will be able to access and use this new query when they **sync** to the most up-to-date queries stored in the version control system.
7. If the query has a problem or business needs change, the analyst can ***undo*** the change to the query using the version control system. The analyst can look at a chronological list of all changes made to the query and who made each change. Then, after finding their own change, the analyst can **revert** to the previous version.
8. The query is back to what it was before the analyst made the change. And everyone at the company sees this reverted, original query, too.

## **Key takeaways**

Engineers, writers, and data analysts use different methods to keep track of changes they make to their work. Automated version control, changelogs, and version control systems are all common tools used to track changes. Changelogs are particularly useful, as they can be used to record the reasons for changes made to data. This can help to ensure data integrity and consistency. Version control systems are most commonly used when making changes to shared queries. They enable analysts to track any changes made and revert to previous versions if necessary.

[**SELF-REFLECTION: CREATING A CHANGELOG**](https://www.coursera.org/learn/process-data/quiz/ejgYq/self-reflection-creating-a-changelog)



## **Activity Overview**

****

Now that you have learned about the importance of keeping track of changes in your data analysis, you can pause for a moment and track what you are learning. In this self-reflection, you will consider your thoughts about changelogs and respond to brief questions.

This self-reflection will help you develop insights into your own learning and prepare you to incorporate changelogs into your data cleanings procedures. As you answer questions—and come up with questions of your own—you will consider concepts, practices, and principles to help refine your understanding and reinforce your learning. You’ve done the hard work, so make sure to get the most out of it: This reflection will help your knowledge stick!

Review the following explanation on the importance of changelogs. Then complete the step-by-step instructions.

### 

### **Step-By-Step Instructions**

### **Step 1: Consider the importance of changelogs**

In previous activities, you’ve reviewed the different types of questions to ask before exploring data, the importance of pre-cleaning data, the basic functions of SQL, how to clean data with spreadsheets, and more. As a junior data analyst, most of your projects will consist of these activities. As you have experienced, each of these tasks follows a complicated process. Therefore, consistent and accurate record-keeping is essential to keeping you on track.

A changelog is a document used to record the notable changes made to a project over its lifetime across all of its tasks. It is typically curated so that the changes it records are listed chronologically across all versions of the project.

The major benefit to using changelogs is that contributors and users connected with the project get a specific list of what important alterations have been made, when they were made, and sometimes, what version they were released for. It is an invaluable tool for communicating how the project has evolved over time to coworkers, management, and stakeholders.

### 

### **Step 2: Follow best practices for changelogs**

A changelog for a personal project may take any form desired. However, in a professional setting and while collaborating with others, readability is important. These guiding principles help to make a changelog accessible to others:

* Changelogs are for humans, not machines, so write legibly.
* Every version should have its own entry.
* Each change should have its own line.
* Group the same types of changes. For example, *Fixed* should be grouped separately from *Added*.
* Versions should be ordered chronologically starting with the latest.
* The release date of each version should be noted.

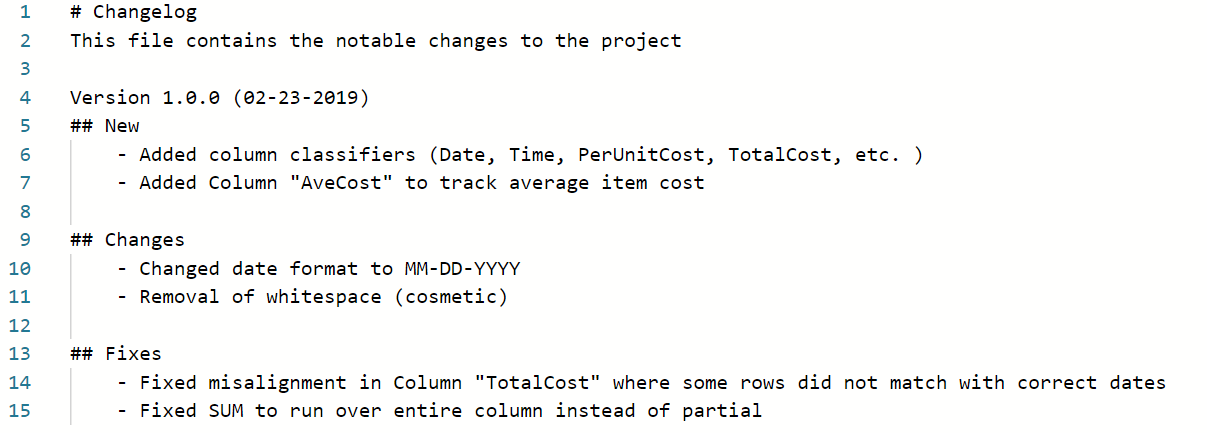
All the changes for each category should be grouped together. Types of changes usually fall into one of the following categories:

* Added: new features introduced
* Changed: changes in existing functionality
* Deprecated: features about to be removed
* Removed: features that have been removed
* Fixed: bug fixes
* Security: lowering vulnerabilities

### **Step 3: Examine a sample changelog**

Examine the figure below for an example of a changelog. Note that the following example is written in [Markdown](https://docs.github.com/en/free-pro-team@latest/github/writing-on-github/basic-writing-and-formatting-syntax), as it is common to keep changelogs as a readme file in a code repository.





### **Step 4: Consider what to record in a changelog**

Now that you're familiar with the example, consider what changes you need to record in a changelog. To start, you record the various changes, additions, and fixes that were discussed above. Arrange them using bullets or numbering with one change per line. Group similar changes together with a label describing the change immediately above them.

Use different version numbers for each milestone reached in your project. Within each version, place the logged changes that were made since the previous version (milestone). Dates are not generally necessary for each change, but they are recommended for each version.

In an upcoming course, you will have the opportunity to complete a capstone project. This will be a great chance to demonstrate your ability to organize a project like a professional data analyst by keeping your own changelog.

You can do this using a simple text file or spreadsheet and include your changelog with the project write-up. It will help you stay organized and collaborate with others. Keep this in mind when you reach the capstone project in an upcoming course, and don’t be afraid to revisit this lesson if you have questions.

[**WHY DOCUMENTATION IS IMPORTANT**](https://www.coursera.org/learn/process-data/lecture/TvyoD/why-documentation-is-important)

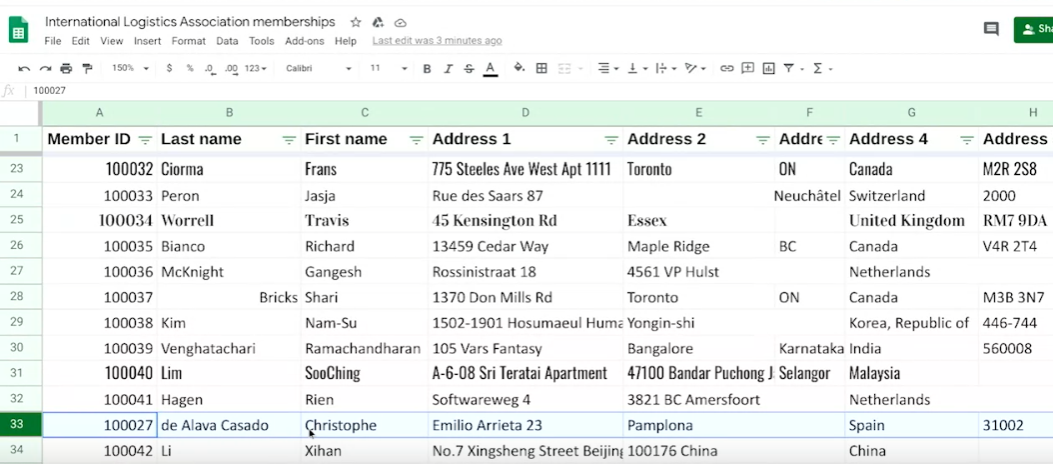
The crime is dirty data. We've gathered the evidence. It's been cleaned, verified, and cleaned again. Now it's time to present our evidence. We'll retrace the steps and present our case to our peers. As we discussed earlier, data cleaning, verifying, and reporting is a lot like crime drama. Now it's our day in court. Just like a forensic scientist testifies on the stand about the evidence, data analysts are counted on to present their findings after a data cleaning effort.

Earlier, we learned how to document and track every step of the data cleaning process, which means we have solid information to pull from. As a quick refresher, documentation is the process of tracking changes, additions, deletions, and errors involved in a data cleaning effort, changelogs are good examples of this. Since it's staged chronologically, it provides a real-time account of every modification. **Documenting will be a huge time saver for you as a future data analyst**. **It's basically a cheatsheet you can refer to if you're working with a similar dataset or need to address similar errors**.

While your team can view changelogs directly, **stakeholders can't** and have to rely on your report to know what you did.

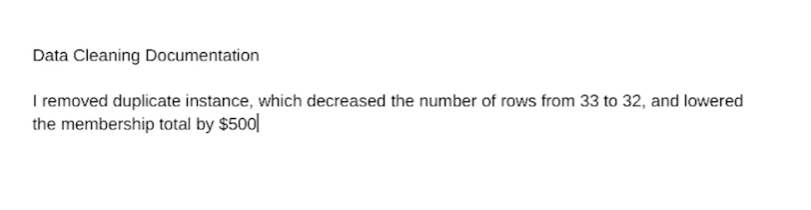
Let's check out how we might document our data cleaning process using the example we worked with earlier. In that example, we found that this association had two instances of the same membership for $500 in its database.

We decided to fix this manually by deleting the duplicate info.



There're plenty of ways we could go about documenting what we did.

One common way is to just **create a doc listing out the steps we took and the impact they had**. For example, first on your list would be that you remove the duplicate instance, which decreased the number of rows from 33 to 32,and lowered the membership total by $500.



**If we were working with SQL**, we could include a comment in the statement describing the reason for a change without affecting the execution of the statement. That's something a bit more advanced, which we'll talk about later. Regardless of how we capture and share our changelogs, we're setting ourselves up for success by being 100 percent transparent about our data cleaning. This keeps everyone on the same page and shows project stakeholders that we are accountable for effective processes. In other words, this helps build our credibility as witnesses who can be trusted to present all the evidence accurately during testimony. For dirty data, it's an open and shut case.

[**FEEDBACK AND CLEANING**](https://www.coursera.org/learn/process-data/lecture/k3c6N/feedback-and-cleaning)

Welcome back. By now it's safe to say that verifying, documenting and reporting are valuable steps in the data-cleaning process. You have proof to give stakeholders that your data is accurate and reliable. And the effort to attain it was well-executed and documented. The next step is getting feedback about the evidence and using it for good, which we'll cover in this video.

Clean data is important to the task at hand. But the data-cleaning process itself can reveal insights that are helpful to a business. The feedback we get when we report on our cleaning can transform data collection processes, and ultimately business development. For example, one of the biggest challenges of working with data is dealing with errors.

**Some of the most common errors** involve human mistakes like mistyping or misspelling, flawed processes like poor design of a survey form, and system issues where older systems integrate data incorrectly.

Whatever the reason, data-cleaning can shine a light on the nature and severity of error-generating processes.

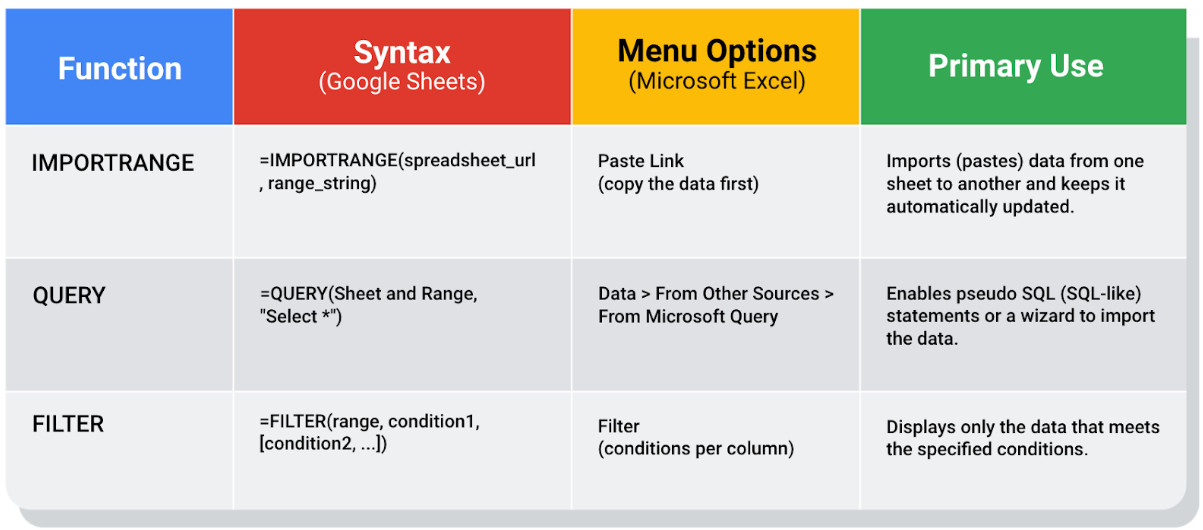
With consistent documentation and reporting, we can uncover error patterns in data collection and entry procedures and use the feedback we get to make sure common errors aren't repeated. Maybe we need to reprogram the way the data is collected or change specific questions on the survey form.

In more extreme cases, the feedback we get can even send us back to the drawing board to rethink expectations and possibly update quality control procedures. For example, sometimes it's useful to schedule a meeting with a data engineer or data owner to make sure the data is brought in properly and doesn't require constant cleaning.

Once errors have been identified and addressed, stakeholders have data they can trust for decision-making. And by reducing errors and inefficiencies in data collection, the company just might discover big increases to its bottom line. Congratulations! You now have the foundation you need to successfully verify a report on your cleaning results. Stay tuned to keep building on your new skills.

[**ADVANCED FUNCTIONS FOR SPEEDY DATA CLEANING**](https://www.coursera.org/learn/process-data/supplement/PLnRS/advanced-functions-for-speedy-data-cleaning)

In this reading, you will learn about some advanced functions that can help you speed up the data cleaning process in spreadsheets. Below is a table summarizing three functions and what they do:



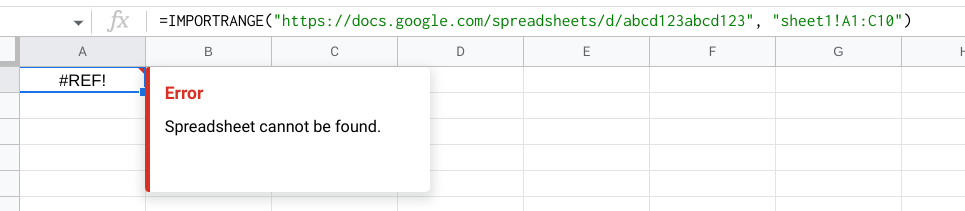
## 

## **Keeping data clean and in sync with a source**

The [**IMPORTRANGE**](https://support.google.com/docs/answer/3093340?hl=en) function in Google Sheets and the [**Paste Link**](https://professor-excel.com/how-to-paste-cell-links/) feature (a Paste Special option in Microsoft Excel) both allow you to insert data from one sheet to another. Using these on a large amount of data is more efficient than manual copying and pasting. They also reduce the chance of errors being introduced by copying and pasting the wrong data. They are also helpful for data cleaning because you can “cherry pick” the data you want to analyze and leave behind the data that isn’t relevant to your project. Basically, it is like canceling noise from your data so you can focus on what is most important to solve your problem. This functionality is also useful for day-to-day data monitoring; with it, you can build a tracking spreadsheet to share the relevant data with others. The data is synced with the data source so when the data is updated in the source file, the tracked data is also refreshed.

In Google Sheets, you can use the **IMPORTRANGE** function. It enables you to specify a range of cells in the other spreadsheet to duplicate in the spreadsheet you are working in. You must allow access to the spreadsheet containing the data the first time you import the data.

**The URL shown below is for syntax purposes only. Don't enter it in your own spreadsheet. Replace it with a URL to a spreadsheet you have created so you can control access to it by clicking the Allow access button.**

****

Refer to the [Google support page for IMPORTRANGE](https://support.google.com/docs/answer/3093340?hl=en#) for the sample usage and syntax.

### 

### **Example of using IMPORTRANGE**

An analyst monitoring a fundraiser needs to track and ensure that matching funds are distributed. They use **IMPORTRANGE** to pull all the matching transactions into a spreadsheet containing all of the individual donations. This enables them to determine which donations eligible for matching funds still need to be processed. Because the total number of matching transactions increases daily, they simply need to change the range used by the function to import the most up-to-date data.

On Tuesday, they use the following to import the donor names and matched amounts:

**=IMPORTRANGE("https://docs.google.com/spreadsheets/d/abcd123abcd123", "sheet1!A1:C10", "Matched Funds!A1:B4001")**

On Wednesday, another 500 transactions were processed. They increase the range used by 500 to easily include the latest transactions when importing the data to the individual donor spreadsheet:

**=IMPORTRANGE("https://docs.google.com/spreadsheets/d/abcd123abcd123", "Matched Funds!A1:B4501")**

**Note: The above examples are for illustrative purposes only. Don't copy and paste them into your spreadsheet. To try it out yourself, you will need to substitute your own URL (and sheet name if you have multiple tabs) along with the range of cells in the spreadsheet that you have populated with data.**

## **Pulling data from other data sources**

The [**QUERY**](https://support.google.com/docs/answer/3093343?hl=en) function is also useful when you want to pull data from another spreadsheet. The **QUERY** function's SQL-like ability can extract specific data within a spreadsheet. For a large amount of data, using the **QUERY** function is faster than filtering data manually. This is especially true when repeated filtering is required. For example, you could generate a list of all customers who bought your company’s products in a particular month using manual filtering. But if you also want to figure out customer growth month over month, you have to copy the filtered data to a new spreadsheet, filter the data for sales during the following month, and then copy those results for the analysis. With the **QUERY** function, you can get all the data for both months without a need to change your original dataset or copy results.

The **QUERY** function syntax is similar to **IMPORTRANGE**. You enter the sheet by name and the range of data that you want to query from, and then use the SQL **SELECT** command to select the specific columns. You can also add specific criteria after the **SELECT** statement by including a **WHERE** statement. But remember, all of the SQL code you use has to be placed between the quotes!

Google Sheets run the Google Visualization API Query Language across the data. Excel spreadsheets use a query wizard to guide you through the steps to connect to a data source and select the tables. In either case, you are able to be sure that the data imported is verified and clean based on the criteria in the query.

### **Examples of using QUERY**

Check out the [Google support page for the QUERY function](https://support.google.com/docs/answer/3093343?hl=en) with sample usage, syntax, and examples you can download in a Google sheet.

Link to make a copy of the sheet: [QUERY examples](https://docs.google.com/spreadsheets/d/1815H5TCe91LLT6tD6FmxMHmeJAAkr4o5Q6rNpV6xiFk/copy)

### **The solution**

Analysts can use SQL to pull a specific dataset into a spreadsheet. They can then use the **QUERY** function to create multiple tabs (views) of that dataset. For example, one tab could contain all the sales data for a particular month and another tab could contain all the sales data from a specific region. This solution illustrates how SQL and spreadsheets are used well together.

## **Filtering data to get what you want**

The [**FILTER**](https://support.google.com/docs/answer/3093197?hl=en) function is fully internal to a spreadsheet and doesn’t require the use of a query language. The **FILTER** function lets you view only the rows (or columns) in the source data that meet your specified conditions. It makes it possible to pre-filter data before you analyze it.

The **FILTER** function might run faster than the **QUERY** function. But keep in mind, the **QUERY** function can be combined with other functions for more complex calculations. For example, the **QUERY** function can be used with other functions like **SUM** and **COUNT** to summarize data, but the **FILTER** function can't.

### **Example of using FILTER**

Check out the [Google support page for the FILTER function](https://support.google.com/docs/answer/3093197?hl=en) with sample usage, syntax, and examples you can download in a Google sheet.

Link to make a copy of the sheet: [FILTER examples](https://docs.google.com/spreadsheets/d/1caULJLQvQuzBnCN7rO9utg0xSKrYms7wM0Ph7A2JXY4/copy)

## Terms and definitions for Course 4, Module 4

CASE: A SQL statement that returns records that meet conditions by including an if/then statement in a query

Changelog: A file containing a chronologically ordered list of modifications made to a project

COUNTA: A spreadsheet function that counts the total number of values within a specified range

Find and replace: A tool that finds a specified search term and replaces it with something else

Verification: A process to confirm that a data-cleaning effort was well executed and the resulting data is accurate and reliable

**\**

**MODULE 4 CHALLENGE**

