Hey there, future data analysts!

You've made a lot of progress so far.

It's not an easy journey,

but you're doing great.

Before you started this program,

something inside of you convinced you to get

your Google Data Analytics Certificate.

You had an idea,

did some research, and made the time to get started.

Then you made the decision to commit to your goal.

Now look where you are!

That is something to be proud of.

Early on, we jumped right

into the world of data analytics and

saw how data played a part in your everyday life.

You learned how to navigate spreadsheets and why

structured thinking was key to solving problems.

You also explored the best ways

to collect and store your data.

From there, you gained an understanding of

clean data and data integrity.

You've identified how to ask

the right questions and learned to clean data.

Now we'll take your skills to the next level.

Next up, you'll learn how to come up with

clear and objective answers to

any data question you encounter.

Earlier, we learned about the data analysis process.

As a quick reminder,

the phases of that process are Ask,

Prepare, Process, Analyze, Share, and Act.

We'll explore the Analyze phase more here,

focusing on how to organize and format the data

you have so that you can do all sorts of calculations.

Knowing how to analyze the data you've collected

and cleaned is essential to your work as an analyst.

Before we get started,

I'd like to introduce myself.

My name is Ayanna, and I'm

excited to be your instructor for this course.

I'm a global insights manager at Google,

and I've also taught at the Google Analytics Academy,

which is a training resource for Google analysts.

In my job, I help advertisers determine

the value of investing in Google products.

When you search for something online,

you'll often see an ad on the page.

That's an investment an advertiser has made.

I use data analysis to show advertisers

the value they could gain from investing in those ads.

That's what I love about being a data analyst:

figuring out how to create value

anytime I enter a situation.

The best way to know if you're creating

value is if you have evidence.

For me, that evidence is data.

Now that you know a little bit about my love for data,

let's talk about what you'll learn here.

You'll start by covering

best practices for organizing your data

and the different ways you can sort through

that data using spreadsheets and SQL.

We'll also spend time learning three important ways

to work with data that will boost your analytical skills.

Then we'll talk about saving time.

You'll discover tips and tricks that can help

you analyze data more efficiently.

Last but not least,

we'll work together to identify techniques to help

you be as fair and unbiased as possible.

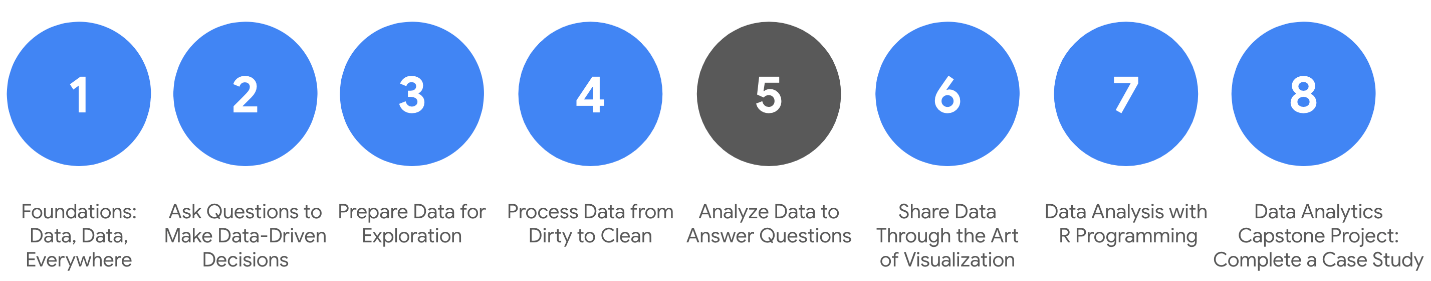
Well, that's all you need to know for now.

Coming up, we'll break down

the basics of data analysis and

bring you one step closer to a future in data.

**Course 5 overview: Set your expectations**



1. [Foundations: Data, Data, Everywhere](https://www.coursera.org/learn/foundations-data/home/welcome)
2. [Ask Questions to Make Data-Driven Decisions](https://www.coursera.org/learn/ask-questions-make-decisions/home/welcome)
3. [Prepare Data for Exploration](https://www.coursera.org/learn/data-preparation/home/welcome)
4. [Process Data from Dirty to Clean](https://www.coursera.org/learn/process-data/home/welcome)
5. **Analyze Data to Answer Questions** *(this course)*
6. [Share Data Through the Art of Visualization](https://www.coursera.org/learn/visualize-data/home/welcome)
7. [Data Analysis with R Programming](https://coursera.org/learn/data-analysis-r/home/welcome)
8. [Google Data Analytics Capstone: Complete a Case Study](https://coursera.org/learn/google-data-analytics-capstone/home/welcome)

Welcome to the fifth course in the series for the Google Data Analytics Certificate! The goal of data analysis is to make sense out of the data you collect and receive. Up until now, your focus has been on the preparations a data analyst goes through before entering the analysis phase. Specifically, in the last course, you learned about checking data for completeness and cleaning it for accuracy and reliability.

If you feel like a hiker who has climbed a great distance to get to higher ground, we are excited to tell you that you have arrived! You have reached a stage where you are ready to work directly with data. You will organize and format data. This will help you think about data in different ways. Similar to how the view from a hiker’s lookout is amazing, your view of data from this point on will be spectacular.

You will have hands-on practice organizing, sorting, filtering, formatting, converting, and combining data in spreadsheets. These are tasks you would complete in a real data analysis project. You will also learn how to sort and filter your data using SQL queries. You will be using functions and writing queries frequently as you continue your learning.

**Course content**

Course 5 – Analyze Data to Answer Questions

1. **Organize data to begin analysis.** Organizing data makes the data easier to use in an analysis. In this part of the course, you will learn the importance of organizing your data with sorting and filtering. You will explore organizing data in both spreadsheets and with SQL queries and temporary tables.
2. **Format and adjust data.** As you move closer to analyzing your data, you will want to have the data formatted and ready to go. In this part of the course, you will learn all about converting and formatting data, including how to use SQL queries to combine data. You will also discover the value of feedback and support from your colleagues and how it can lead to new insights that you can apply to your work.
3. **Aggregate data for analysis.** During an analysis, you might need to combine data to gain insights and complete business objectives. In this part of the course, you will explore the functions, procedures, and syntax to combine, or aggregate data. You will learn how to combine data within multiple cells in spreadsheets, and within multiple database tables using SQL queries.
4. **Perform data calculations.** Calculations are one of the more common tasks that data analysts perform during an analysis. In this part of the course, you will explore formulas, functions, and pivot tables in spreadsheets and SQL queries. All of these are used in data calculations. You will also learn about the benefits of using SQL to manage temporary database tables.

**What to expect**

You can expect to finish this course in about five weeks when you have completed all of the prescribed activities, which include:

* **Videos** of instructors teaching new concepts and demonstrating the use of tools
* **In-video questions** that pop up during or at the end of a video to check your learning
* **Readings** to introduce new ideas and build on the concepts from the videos
* [**Discussion forums**](https://www.coursera.org/learn/analyze-data/discussions) to discuss, explore, and reinforce new ideas for better learning
* **Discussion prompts** to promote thinking and engagement in the discussion forums
* **Hands-on activities** to introduce real-world, on-the-job situations, and the tools and tasks to complete assignments
* **Practice quizzes** to prepare you for graded quizzes
* **Hands-on activities** toreinforce learned skills for the graded quizzes
* **Graded quizzes** to measure your progress and give you valuable feedback

Hands-on activities promote additional opportunities to build your skills. Try to get as much out of them as possible. Assessments are based on the approach taken by the course to offer a wide variety of learning materials and activities that reinforce important skills. Graded and ungraded quizzes will help the content sink in. Ungraded practice quizzes are a chance for you to prepare for the graded quizzes. Both types of quizzes can be taken multiple times.

As a quick reminder, this course is designed for all types of learners, with no degree or prior experience required. Everyone learns differently, so the Google Data Analytics Certificate has been designed with that in mind. Personalized deadlines are just a guide, so feel free to work at your own pace. There is no penalty for late assignments. You'll see a **Reset deadlines** option on the Grades and Overview pages. Click it to switch to a new schedule for the course with updated deadlines.  [Contact Coursera](https://www.coursera.support/s/article/360036160591-How-to-contact-Coursera?) if you need additional assistance.

If you would like to review previous content or get a sneak peek of upcoming content, you can use the navigation links at the top of this page to go to another course in the program. When you pass all required assignments, you will be on track to earn your certificate.

**Tips**

* Try to complete all items in order. All new information builds on previous lessons.
* Treat every task as if it is real-world experience. Have a mindset that you are working at a company or in an organization as a data analyst. This will help you apply what you learn to the real world.
* Repeat demonstrated tasks on your own for extra practice and speed.
* Even though they aren’t graded, it is important to complete all practice items. They will help you build a strong foundation as a data analyst and prepare you for the graded assessments.
* Take advantage of all additional resources provided, including discussion forums and links to external articles for more information.
* When you encounter useful links in the course, remember to bookmark them so you can refer to the information later for study or review.
* Additional resources are free, but some sites place limits on how many articles you can access for free each month. Sometimes you can register on the site for full access, but you can always bookmark a resource and come back to view it later.
* Use a notebook or electronic journal to keep track of new formulas, functions, and syntax that you learn. That way, you will be able to refer back to any notes as needed.

Data analysts use spreadsheets and SQL queries a lot. If you perform all of the assigned activities in the application environments that are recommended, you will get a good idea of what you could be doing on the job as a data analyst.

**Updates to the course**

As you complete this course, you may notice updates to the content, like new practice materials and additional examples. These updates ensure the program provides up-to-date skills and guidance that will help you in your data analytics career. If you previously completed a graded activity, you *may* need to repeat the assessment in order to complete this course.  For more information, check out [the course discussion forum.](https://www.coursera.org/learn/analyze-data/discussions)

Welcome back. It's great to see you again.

So let's talk about analysis.

We've learned how to ask the right questions,

prepare data for exploration,

and then process that data

to make sure it's squeaky clean.

Now it's time for the heart of

the process: the actual analysis!

Finally, right? But what is analysis?

Basically, analysis is the process

used to make sense of the data collected.

It means taking the right steps to proceed

and think about your data in different ways.

The goal of analysis is to identify

trends and relationships within

the data so that you can

accurately answer the question you're asking.

To do this, you should stick to

the 4 phases of analysis:

organize data, format and adjust data,

get input from others,

and transform data by observing relationships

between data points and making calculations.

Let's apply the 4 phases of

analysis to a real-world scenario.

Imagine you want to buy a gift for

your friend Zara's wedding.

The problem is you're not sure what to get her.

Fortunately, you have a ton of

data from her wedding website.

But instead of reading all the data on her website

and scrolling through a photo album

of her and her partner,

you go straight to the online registry,

a wish list of gifts they'd enjoy.

The registry is like a dataset that you

can analyze to make a decision.

Now that you're checking out

organized data in the registry,

you want to make sure that the list of data, or gifts

in this case, is formatted in

a way that's easy to reference.

Formatting data streamlines things and saves you time.

Scrolling through hundreds of

gifts can be time-consuming.

Instead, you can adjust the data in a way that makes it

easy to digest by filtering and sorting your data.

You have a budget you want to stick to,

so you sort the gift prices from low to high.

You then filter prices to include

gifts that are within your budget of $60.

You're working with a newly formatted list of data.

At this point, it's good to

remember that input from other people can

also be really helpful when

analyzing information and making decisions.

You can check the list of gifts to figure out if

anyone else has already bought any of the items.

You realize a few of the items

in the list have been purchased,

and this informs your decision.

When analyzing data, gaining

input from others is important because it

gives you a viewpoint you might not

understand or have access to.

On top of gaining input from other people,

it's also important to seek

out others' perspectives early.

That way, if they predict

any obstacles or challenges, you'll know beforehand.

The people you'll look to for input don't

have to be experts to be helpful.

Sometimes all you need is for someone who's

familiar with a topic or data you're considering.

In our example, that would be

Zara's wedding guests who are purchasing

gifts from the same online registry.

They probably aren't wedding gift experts,

but their collaborative effort to mark off the item

they purchase can help you figure out what not to buy,

which will prevent Zara from getting the same gift twice.

In the end, getting input is valuable to your analysis.

This brings us to the last step of

the analysis: transforming data.

Transforming data means

identifying relationships and patterns

between the data, and making

calculations based on the data you have.

Going back to our example,

you were able to find a gift

that you knew Zara would like,

and one that fits your budget.

You were also able to choose a gift

that wasn't already purchased by someone else.

By finding the relationship between

these data points, you chose,

purchased, and sent a gift that would

answer the problem you wanted to solve.

The beauty of the analysis process is that you probably

already analyze situations in your everyday life.

Whether you're analyzing data in

your personal life or in your career,

these four tasks can help you make better decisions.

The more you do it,

the more comfortable you'll feel with the process.

I hope this gives you a better understanding

of the basics of analysis.

As we move forward,

we'll check out how to locate data for analysis,

both in a spreadsheet and using SQL.

When you're ready, you can go ahead. See you soon!

[MUSIC]

I think one of the coolest things about working with data at

Google is that we have one of the world's, most valuable datasets.

People refer to Google data as really a lens into human curiosity.

We often look at Google as really a proxy for what's happening in the world.

And so for many of our advertisers, they really, really value the data and

the insights that we're able to give them from Google because they believe it's

a proxy or a reflection of what's happening in their business or

within their industry.

And so I think the value of the data that we're able to work with at Google

really keeps me interested and excited about the work that I do.

So I came to Google about three years ago after spending a few years in consulting.

And so I was really interested in switching into a role that was really

focused on sales and marketing.

But at the same time, I still wanted to be able to leverage the analytical skill set

that I had gained prior.

This role was a great complement to the skillsets that I already had and

the interest that I had in moving into the sales and marketing function.

I think one important thing for

all students to realize is that no one learns this material overnight.

Many of your colleagues you may look at as experts, but most likely they've been

able to gain that level of expertise through their years within the field.

I think one of the biggest attributes that students should keep in mind is that

the most important thing that they need to have throughout this

learning journey is grit.

Grit to understand that it may be a struggle, it may be a challenge, but if

you put in the work, you put in the time, these concepts will eventually click,

and you'll be well on your way to becoming a data analyst.

Hi, my name is Ayanna and I'm a global insights manager here at Google.

Hi again. Let's jump back in.

Right now we're in the Analyze phase of the data analysis process.

And even though each phase is unique, data analysts make decisions about organization

throughout all of them.

Play video starting at ::15 and follow transcript0:15

That's what we're talking about here:

organization. It's super important that you keep your data organized throughout

your analysis.

How your data is classified and structured will impact your findings,

whether you're working in a spreadsheet or a database.

And once you know how your data is organized, you'll be able to capture or

collect the information you need.

Most of the data you'll use in your analysis will be organized in tables.

Tables help you organize similar kinds of data into categories and

subject areas that you can focus on as you analyze.

For example, this basic database has tables for car dealerships,

product details, and repair parts.

Each table then has several fields of data, like branch owner and

the cost of repair parts.

You can use these tables and

fields to help you decide how to move forward with your analysis.

The structure of this database can help you decide which data you need

to pull to meet your objectives.

For example, the total number of a particular brand of car sold, or

a repair part for a specific make and model of a car at a certain branch.

Play video starting at :1:24 and follow transcript1:24

Tables allow you to make decisions about data types.

They help you to figure out what variables you need and

the data type those variables should have.

So if you have a database where you need to convert a data type during your analysis,

you can do that by using the CAST command in SQL or

any other method that you learn on the job or from your own research.

Like this example where we converted a purchase price column to be

a FLOAT instead of a STRING

so that it was in a numerical form we could use for calculations.

If you're performing your analysis in a spreadsheet,

you want to make sure that the columns and rows are effectively organized.

You can even hide columns that you won't need for analysis or

that show duplicate information.

Once you have the data organized and formatted, you'll be ready to sort and

filter it to find the data you need.

We'll cover sorting and filtering soon.

But for now, just know that both filters and

sorts are affected by the type of data we're working with.

The bottom line is that it's important to have your data in the right format.

So always be prepared to adjust, no matter how far into your analysis you are.

That's all for now.

Coming up, we'll show you what filters are all about. Bye!

Required

English

​

Help Us Translate

# Sort and filter data to keep it organized

The first two phases of data analysis, **Organize data** and **Format and adjust data**, are important for data analysts because they can use these phases to manipulate their data in ways that make important patterns and trends more obvious. Most of the datasets you’ll use as a data analyst will be organized as tables. Tables are helpful because they let you manipulate and categorize your data. Having distinct categories and classifications lets you focus on, and differentiate between, the groups in your data quickly and easily.

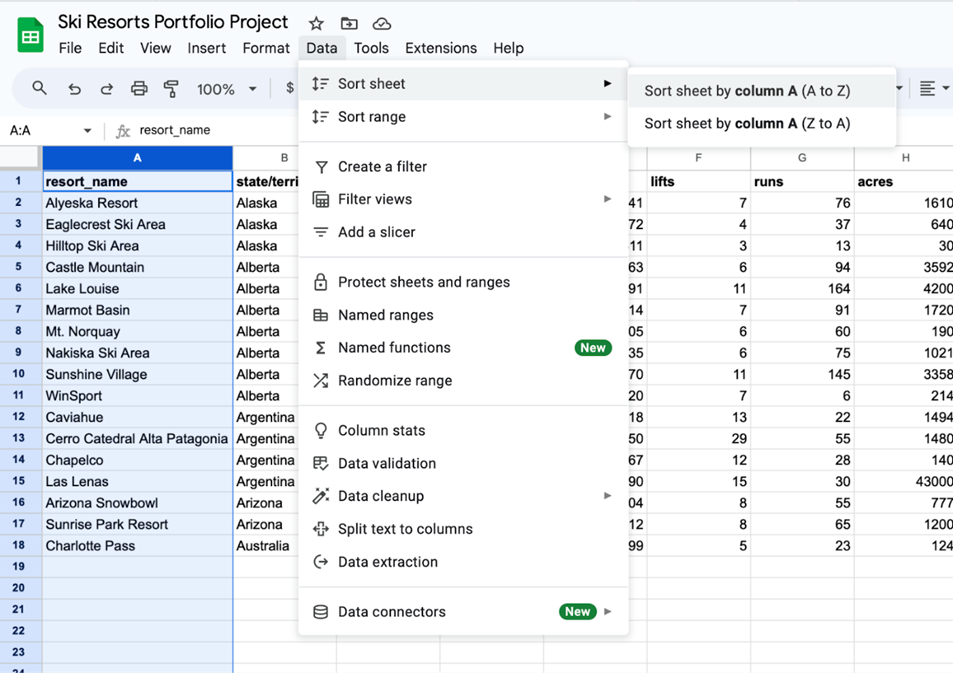
Sorting and filtering are two methods you can use to organize, format, and adjust data. For example, a filter can help you find errors or outliers so you can fix or flag them before your analysis. Outliers are data points that are very different from similarly collected data and might not be reliable values. The benefit of filtering the data is that after you fix errors or identify outliers, you can remove the filter and return the data to its original organization.

In this reading, you’ll review sorting and filtering and consider how they can be used together. You’ll also be introduced to how a particular form of sorting is done in a pivot table.

## Sort data

Sorting is the process of arranging data into a meaningful order to make it easier to understand, analyze, and visualize. It ranks your data based on a specific metric you choose. You can sort data in spreadsheets, SQL databases (when your dataset is too large for spreadsheets), and tables in documents.

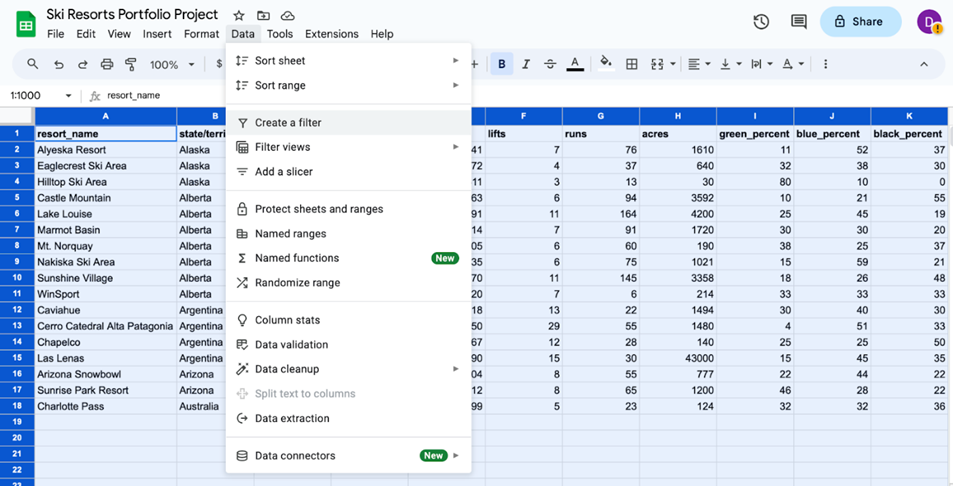
To rank items or create chronological lists, you can sort by ascending or descending order. Sorting arranges the data in a meaningful way and gives you immediate insights. Sorting also helps you to group similar data together by a classification. For example, if a ski resort design company wants to evaluate the resorts designed by a competitor, a data analyst can sort competitive resorts by locations, runs, acreage, and other factors. This way, the firm’s designers can visit the types of resorts they also design and gather information that could be used in its own future designs.

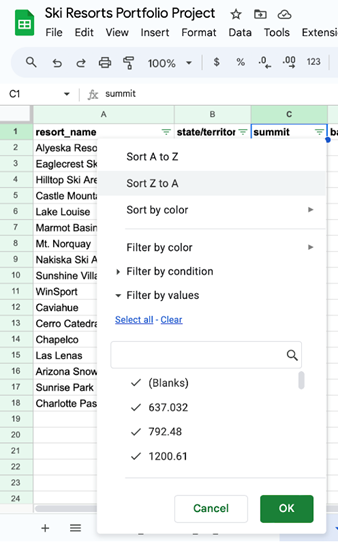
An example of sorting a spreadsheet of ski resorts, including information about resort name, state/territory/country, lifts, runs, and acres. The image taker has clicked into the Data menu option, selected Sort sheet, and is hovering over Sort sheet by column A (A to Z).

## Filter data

Sometimes, an analysis may require only a subset of the data in your dataset. You can use a filter to show only the data that meets a specified criteria while hiding the rest. Filtering is useful when you have lots of data. You can save time by zeroing in on the data that’s important for your current analysis or the data that contains errors. Most spreadsheets and SQL databases allow you to filter your data in a variety of ways. Filtering gives you the ability to find what you are looking for without too much effort.

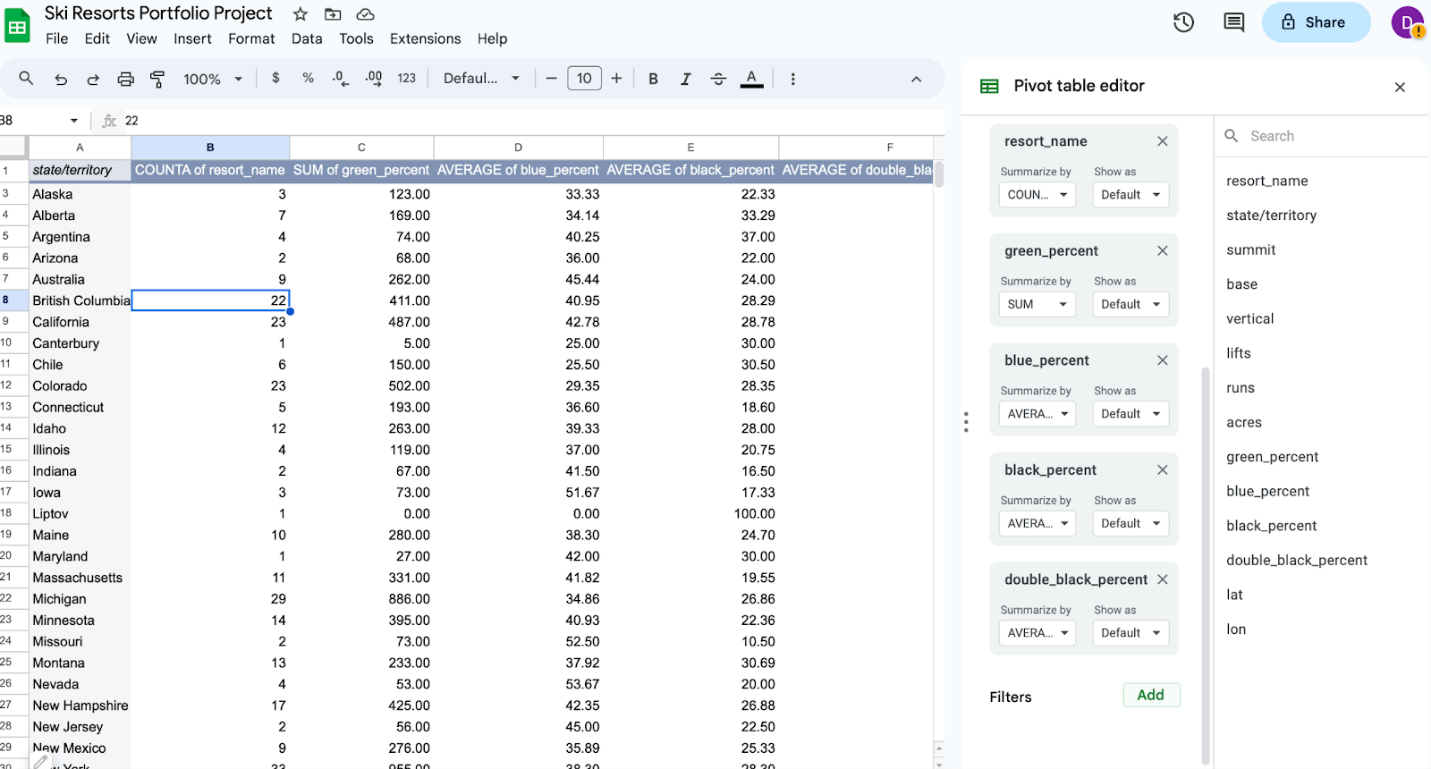
For example, if the ski resort design company wants to inspect specific criteria for the competitive ski resorts they intend to visit and evaluate, a data analyst can filter the competitive resort database to extract information about the number of runs compared to acreage to identify design trends or other insights.



An example of filtering data in a spreadsheet of ski resort information by specific evaluation criteria such as location, acreage, or number of runs

### **Sort a pivot table**

A pivot table is a data summarization tool used to sort, reorganize, group, count, total, or average data. Items in the row and column areas of a pivot table are sorted in ascending order by any custom list first. If the items aren’t in a custom list, they will be sorted in ascending order by default. But, if you sort in descending order, you are setting up a rule that controls how the field is sorted even after new data points are added. For example, in the ski resort dataset, the pivot table allows locations to be sorted alphabetically by state, territory, or country.

Image of a pivot table of the ski resort data, with the pivot table editor open and ready for parameters to be entered. The data is grouped by state, territory, or country.

## Key takeaways

Data analysts filter and sort data to organize it for better understanding, analysis, and visualization. Sorting arranges data in a meaningful order, while filtering displays only data that meets specific criteria. Combining filtering and sorting allows for organizing only relevant data for analysis. Both spreadsheets and SQL databases allow for data filtering and sorting data.

# Review: Set up your BigQuery account

**Note:** This reading is also in Courses 3 and 4 of this program. If you’re taking the courses in order, you may either review it or move on to the next new course item, [Upload the movie dataset to BigQuery](https://www.coursera.org/learn/analyze-data/supplement/sBFZn/upload-the-movie-dataset-to-bigquery). If you haven’t taken Courses 3 or 4 you should complete this reading before proceeding to the next course item.

As you’ve been learning, BigQuery is a database you can use to access, explore, and analyze data from many sources. Now, you’ll begin using BigQuery, which will help you gain SQL knowledge by typing out commands and troubleshooting errors. This reading will guide you through the process of setting up your very own BigQuery account.

**Note:** Working with BigQuery is not a requirement of this program. Additional resources for other SQL database platforms are also provided at the end of this reading if you choose to use them instead.

## BigQuery account options

BigQuery offers a variety of account tiers to cater to various user needs and has two free-of-charge entry points, a sandbox account and a free-of-charge trial account. These options allow you to explore the program before selecting the best choice to suit your needs. A sandbox account allows you to practice writing queries and to explore public datasets free of charge, but it has [quotas and limits](https://cloud.google.com/bigquery/quotas), as well as some additional [restrictions](https://cloud.google.com/bigquery/docs/sandbox#limits). If you prefer to use BigQuery with the standard limits, you can set up a free-of-charge trial account instead. The free-of-charge trial is a trial period prior to paying for a subscription. In this instance, there is no automatic charge, but you will be asked for payment information when you create the account.

This reading provides instructions for setting up either account type. An effective first step is to begin with a sandbox account and switch to a free-of-charge trial account when needed to run the SQL presented upcoming courses.

### **Sandbox account**

The sandbox account is available at no cost, and anyone with a Google account can use it. However, it does have some limitations. For instance, you are limited to a maximum of 12 projects at a time. This means that, to create a 13th project, you'll need to delete one of your existing 12 projects. Additionally, the sandbox account doesn't support all operations you’ll do in this program. For example, there are limits on the amount of data you can process and you can’t insert new records into a database or update the values of existing records. However, a sandbox account is perfect for most program activities, including all of the activities in this course. Additionally, you can convert your sandbox account into a free-of-charge trial account at any time.

**Set up your sandbox account**

To set up a sandbox account:

1. Visit the [BigQuery sandbox documentation](https://cloud.google.com/bigquery/docs/sandbox#limits) page.
2. Log in to your preferred Google account by selecting the profile icon in the BigQuery menu bar.
3. Select the **Go to BigQuery** button on the documentation page.
4. You'll be prompted to select your country and read the terms of service agreement.
5. This will bring you to the **SQL Workspace**, where you'll be conducting upcoming activities. By default, BigQuery creates a project for you.

After you set up your account, the name of the project will be in the banner in your BigQuery console.

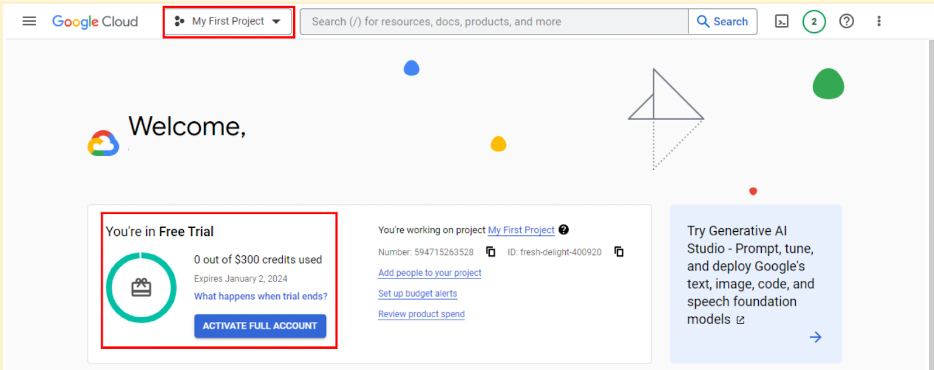
### **Free-of-charge trial**

If you wish to explore more of BigQuery's capabilities with fewer limitations, consider the Google Cloud Free Trial. It provides you with $300 in credit for Google Cloud usage during the first 90 days. If you're primarily using BigQuery for SQL queries, you're unlikely to come close to this spending limit. After you've used up the $300 credit or after 90 days, your free trial will expire, and you will only be able to use this account if you pay to do so. Google won't automatically charge your payment method when the trial ends. However, you'll need to set up a payment option with Google Cloud. This means that you’ll need to enter your financial information. Rest assured, it won't charge you unless you consciously opt to upgrade to a paid account. If you're uncomfortable providing payment information, don't worry; you can use the BigQuery sandbox account instead.

**Set up your free-of-charge trial**

1. Go to the [BigQuery](https://cloud.google.com/bigquery) page.
2. Select **Try BigQuery free**.
3. Log in using your Google email, or create an account free of charge if you don't have one. [Click here](https://cloud.google.com/bigquery?utm_source=google&utm_medium=cpc&utm_campaign=na-US-all-en-dr-bkws-all-all-trial-e-dr-1605212&utm_content=text-ad-none-any-DEV_c-CRE_665665924750-ADGP_Hybrid+%7C+BKWS+-+MIX+%7C+Txt_BigQuery-KWID_43700077225652770-kwd-274188433361&utm_term=KW_bigquery%20account-ST_bigquery+account&gclid=CjwKCAjwkNOpBhBEEiwAb3MvvYQXjIQ4TRnkITJoSXz7DFez4T-XKPG5IpfKmxUg2iHPEmiJBNQByhoCLVgQAvD_BwE&gclsrc=aw.ds) to create an account.
4. Select your country, a description of your organization or needs, and the checkbox to accept the terms of service, Then select **CONTINUE**.
5. Enter your billing information and select **START MY FREE TRIAL**.

After you set up your account, your first project, titled **My First Project** will be in the banner.



### **Transferring between BigQuery accounts**

With either a sandbox or free-of-charge trial account, you have the flexibility to upgrade to a paid account at any time. If you upgrade, all your existing projects will be retained and transferred to your new account. If you started with a free-of-charge trial, but choose not to upgrade when it ends, you can switch to a sandbox account. However, note that projects from your trial won't transfer to your sandbox. Essentially, creating a sandbox is like starting from scratch.

## Get started with other databases (if not using BigQuery)

It’s easiest to follow along with the course activities if you use BigQuery, but you may use other SQL platforms, if you prefer. If you decide to practice SQL queries on other database platforms, here are some resources to get started:

* [Getting Started with MySQL](https://dev.mysql.com/doc/mysql-getting-started/en/)
* [Getting Started with Microsoft SQL Server](https://docs.microsoft.com/en-us/sql/relational-databases/tutorial-getting-started-with-the-database-engine?view=sql-server-ver15)
* [Getting Started with PostgreSQL](https://www.postgresql.org/docs/10/tutorial-start.html)
* [Getting Started with SQLite](https://www.sqlite.org/quickstart.html)

## Key takeaways

BigQuery offers multiple account options. Keep the following in mind when you choose an account type:

* **Account tiers:** BigQuery provides various account tiers to cater to a wide range of user requirements. Whether you're starting with a sandbox account or exploring a paid account with the free-of-charge trial option, BigQuery offers flexibility to choose the option that aligns best with your needs and budget.
* **Sandbox limitations:** While a sandbox account is a great starting point, it comes with some limitations, such as a cap on the number of projects and restrictions on data manipulation operations like inserting or updating records, which you will encounter later in this program. Be aware of these limitations if you choose to work through this course using a sandbox account.
* **Easy setup and upgrades:** Getting started with any BigQuery account type is quick and easy. And if your needs evolve, you have the flexibility to modify your account status at any time. Additionally, projects can be retained even when transitioning between account types.

Choose the right BigQuery account type to match your specific needs and adapt as your requirements change!

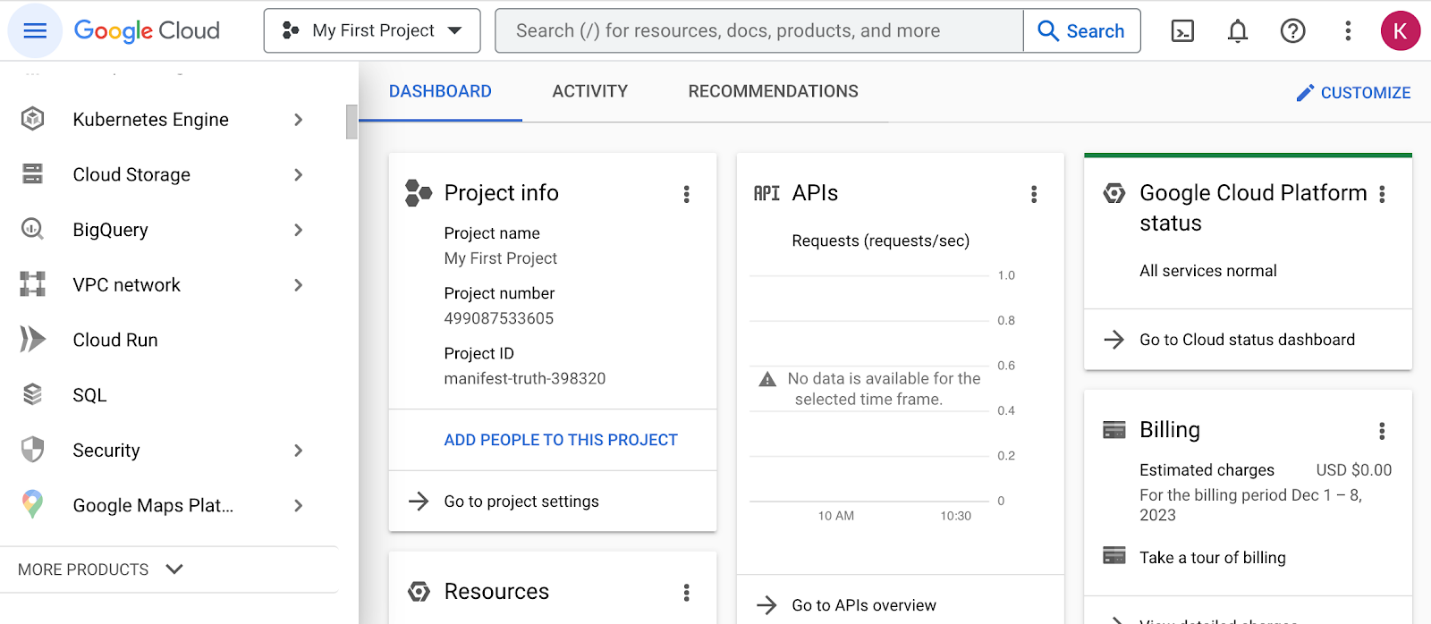
# Review: Get started with BigQuery

**Note:** This reading is also in Courses 3 and 4 of this program. If you’re taking the courses in order, you may either review it or move on to the next course item, [Upload the movie dataset to BigQuery](https://www.coursera.org/learn/analyze-data/supplement/sBFZn/upload-the-movie-dataset-to-bigquery). If you haven’t taken Courses 3 or 4 you should complete this reading before proceeding to the next course item.

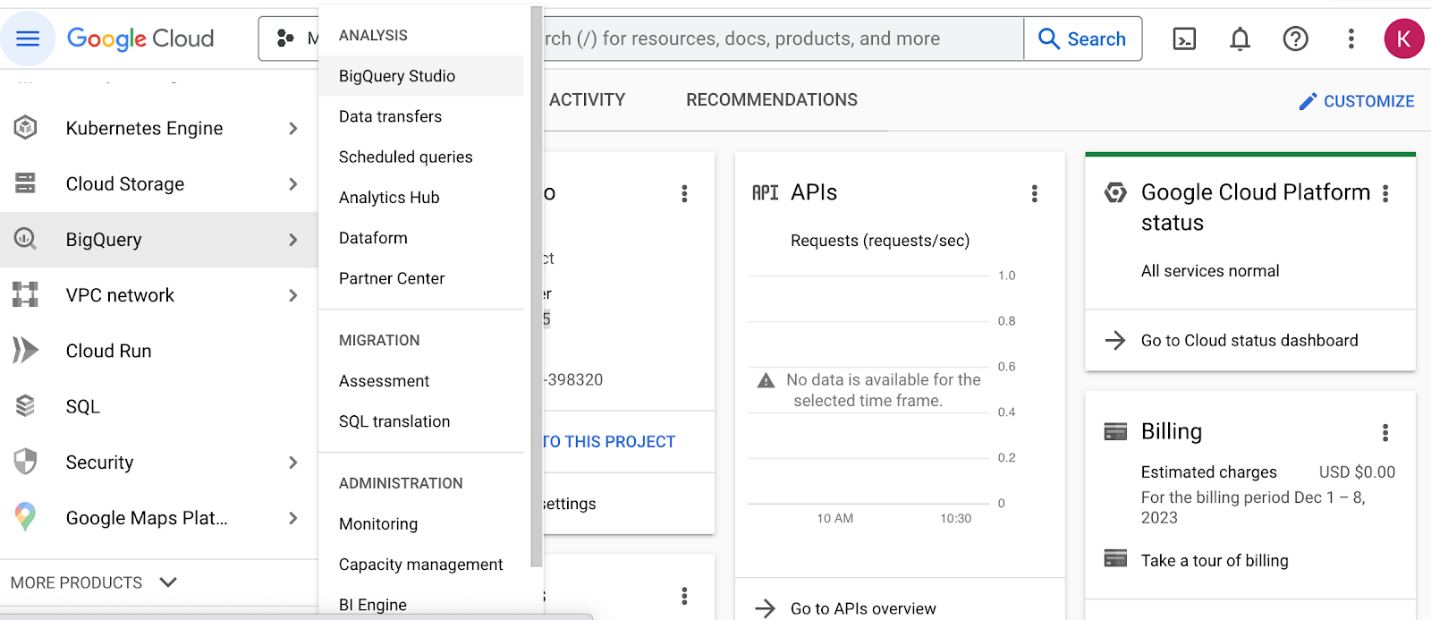
BigQuery is a data warehouse on the Google Cloud Platform used to query and filter large datasets, aggregate results, and perform complex operations. Throughout this program, you’re going to use BigQuery to practice your SQL skills and collect, prepare, and analyze data. At this point, you have set up your own account. Now, explore some of the important elements of the SQL workspace. This will prepare you for the upcoming activities in which you will use BigQuery. Note that BigQuery updates its interface frequently, so your console might be slightly different from what is described in this reading. That’s okay; use your troubleshooting skills to find what you need!

## Log in to BigQuery

When you log in to BigQuery using the landing page, you will automatically open your project space. This is a high-level overview of your project, including the project information and the current resources being used. From here, you can check your recent activity.

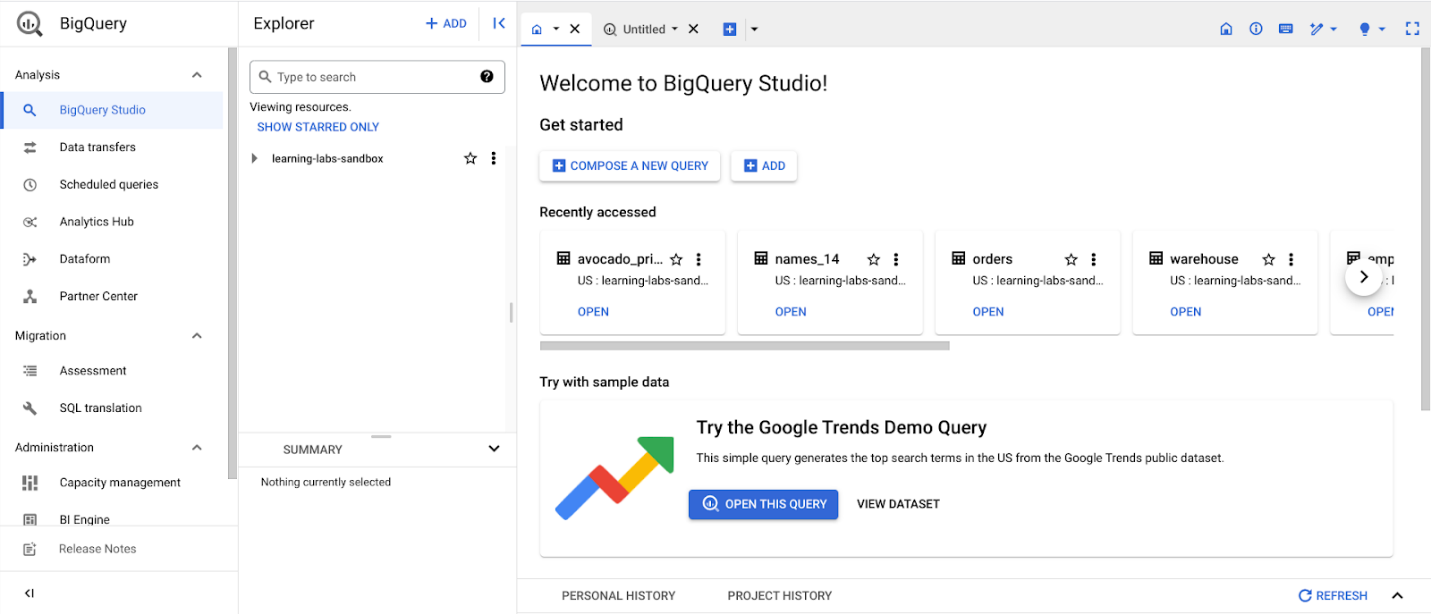


Navigate to your project’s BigQuery Studio by selecting BigQuery from the navigation menu and BigQuery Studio from the dropdown menu.



## BiqQuery Studio components

Once you have navigated to BigQuery from the project space, most of the major components of the BigQuery console will be present: the **Navigation** pane, the **Explorer** pane, and the **SQL Workspace**.

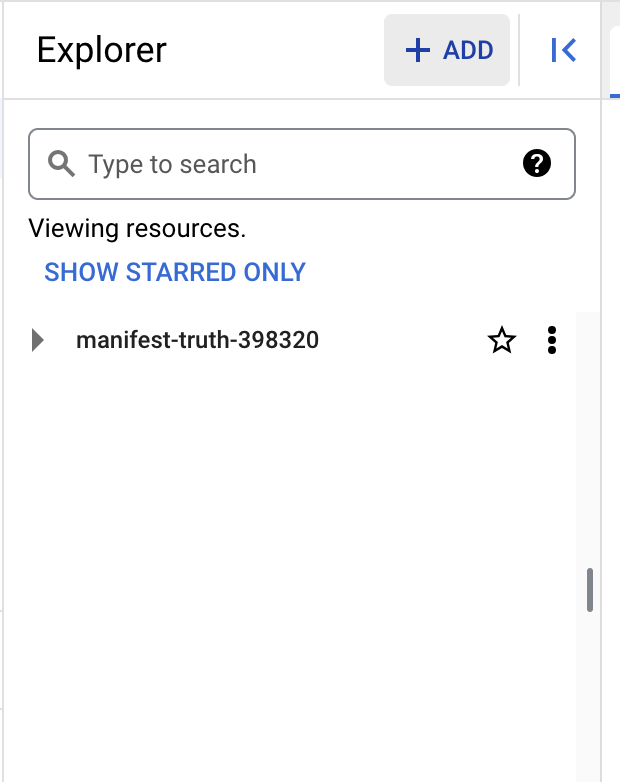


### The Navigation pane

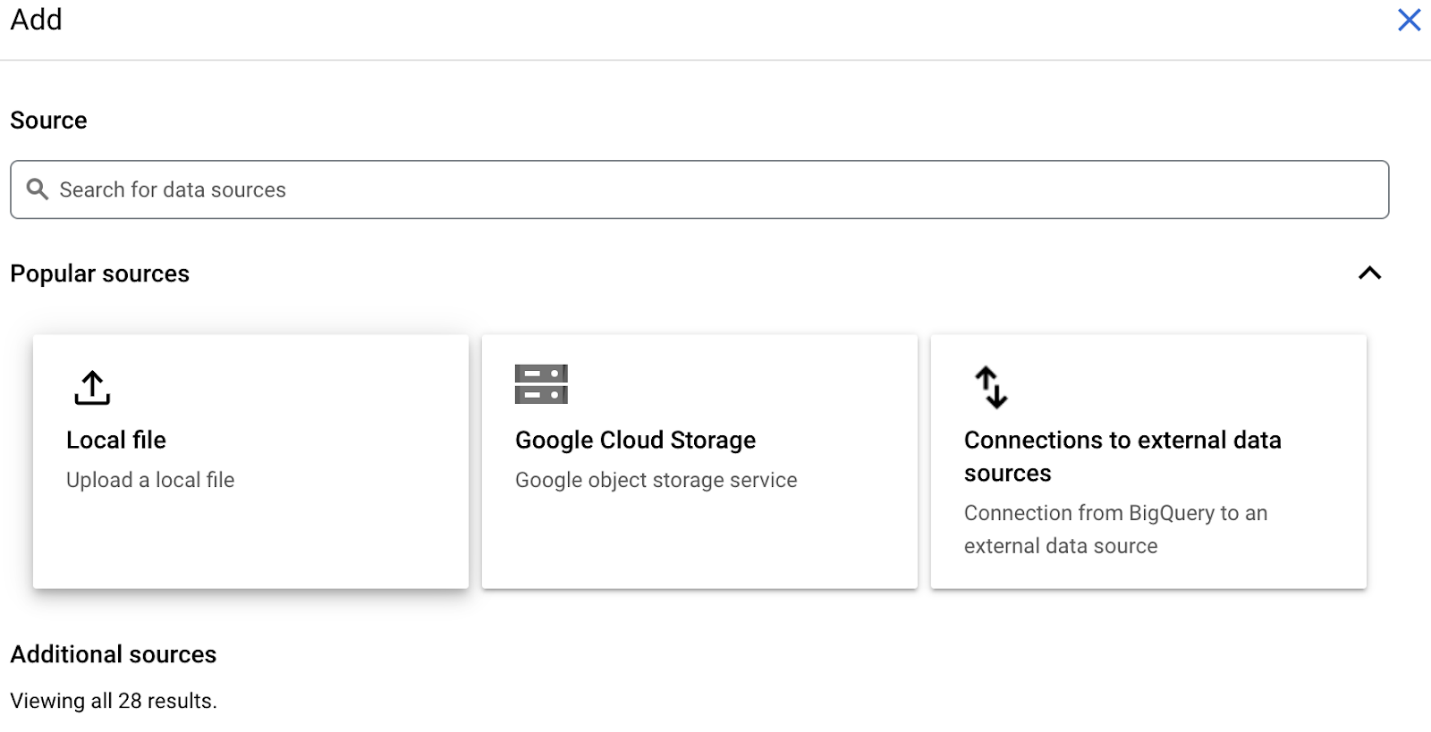
On the console page, find the **Navigation** pane. This is how you navigate from the project space to the BigQuery tool. This menu also contains a list of other Google Cloud Project (GCP) data tools. During this program, you will focus on BigQuery, but it’s useful to understand that the GCP has a collection of connected tools data professionals use every day.

### The Explorer pane

The **Explorer** pane lists your current projects and any starred projects you have added to your console. It’s also where you’ll find the **+ ADD** button, which you can use to add datasets.

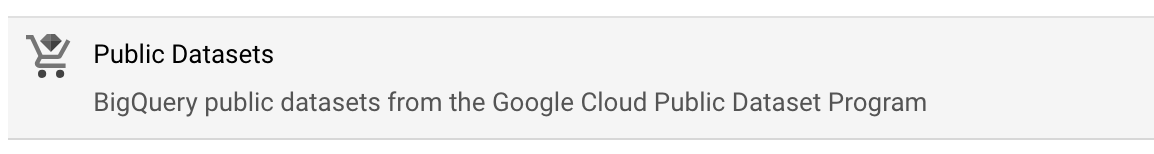


This button opens the **Add** dialog that allows you to open or import a variety of datasets.

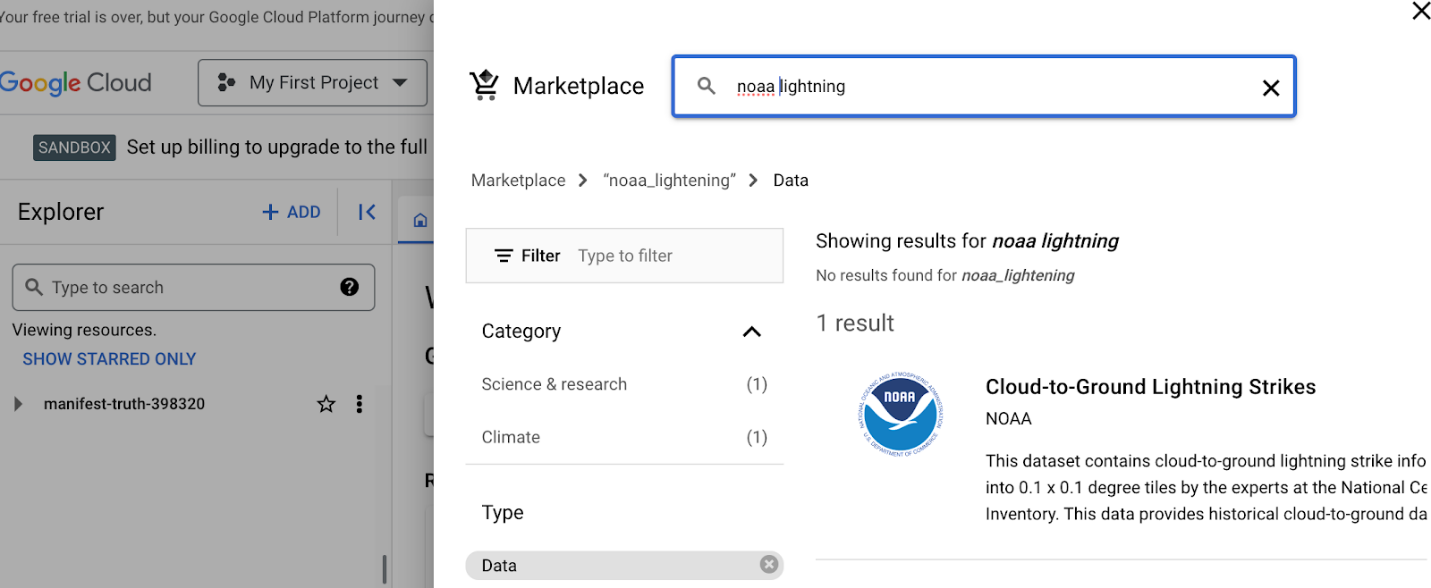


### Add Public Datasets

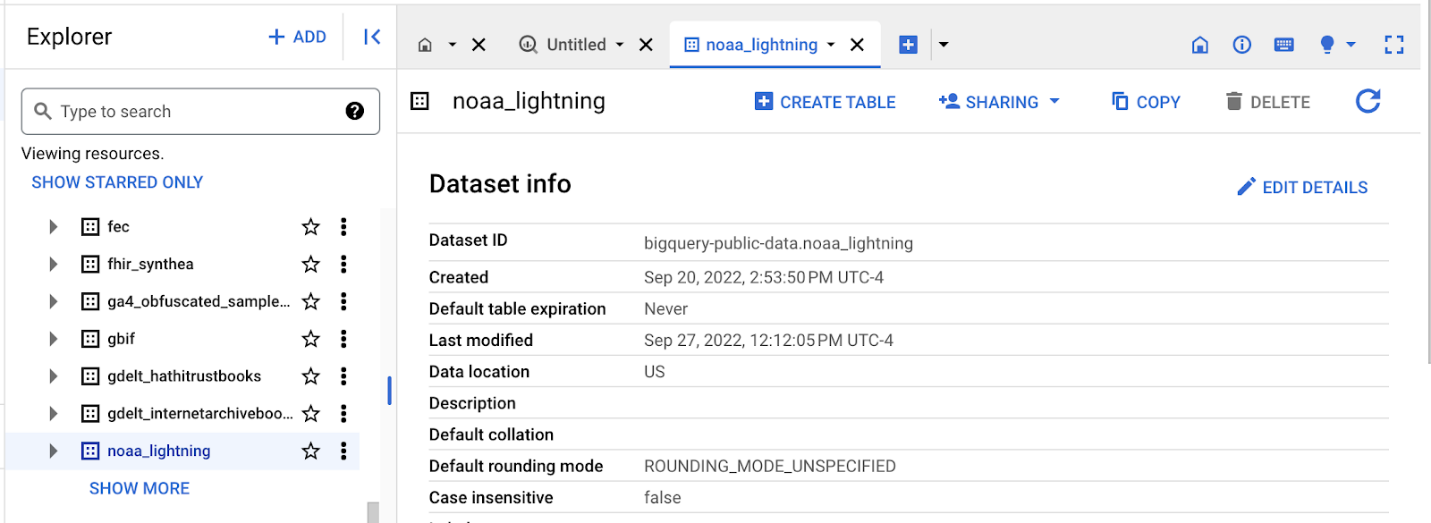
BigQuery offers a variety of public datasets from the Google Cloud Public Dataset Program. Scroll down the **Add** dialog to the **Public Datasets** option.



Select **Public Datasets**. This takes you to the **Public Datasets Marketplace**, where you can search for and select public datasets to add to your BigQuery console. For example, search for the "noaa lightning" dataset in the Marketplace search bar. When you search for this dataset, you will find NOAA’s Cloud-to-Ground Lightning Strikes data.

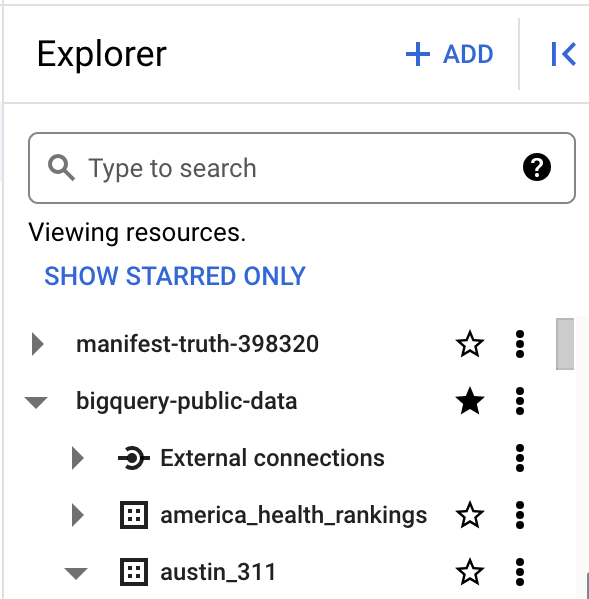


Select the dataset to read its description. Select **View dataset** to create a tab of the dataset’s information within the SQL workspace.

The Explorer Pane lists the noaa\_lightning and other public datasets.

### Star and examine Public Datasets

You added the public noaa\_lightning dataset to your BigQuery Workspace, so the **Explorer** pane displays the noaa\_lightning dataset, along with the list of other public datasets. These datasets are nested under bigquery-public-data. Star bigquery-public-data by navigating to the top of the **Explorer** pane and selecting the star next to bigquery-public-data.



Starring bigquery-public-data will enable you to search for and add public datasets by scrolling in the **Explorer** pane or by searching for them in the **Explorer** search bar.

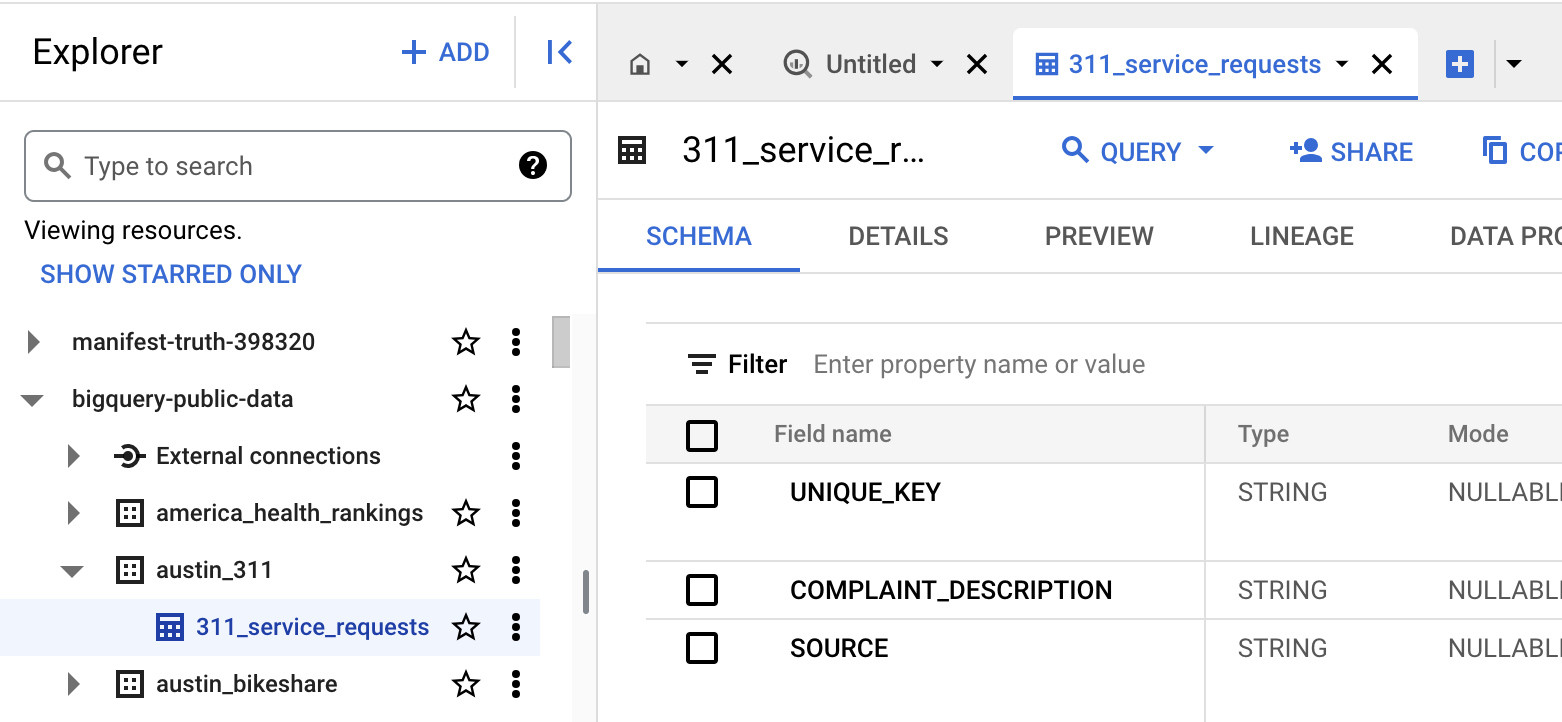
For example, you might want to select a different public dataset. If you select the second dataset, "austin\_311," it will expand to list the table stored in it, “311\_service\_requests.”

A screenshot of a search engine

Description automatically generatedThe Explorer pane with the “bigquery-public data” and “austin\_311” datasets expanded, revealing the “311\_service\_requests” table

When you select a table, its information is displayed in the SQL Workspace. Select the 311\_service\_requests table to examine several tabs that describe it, including:

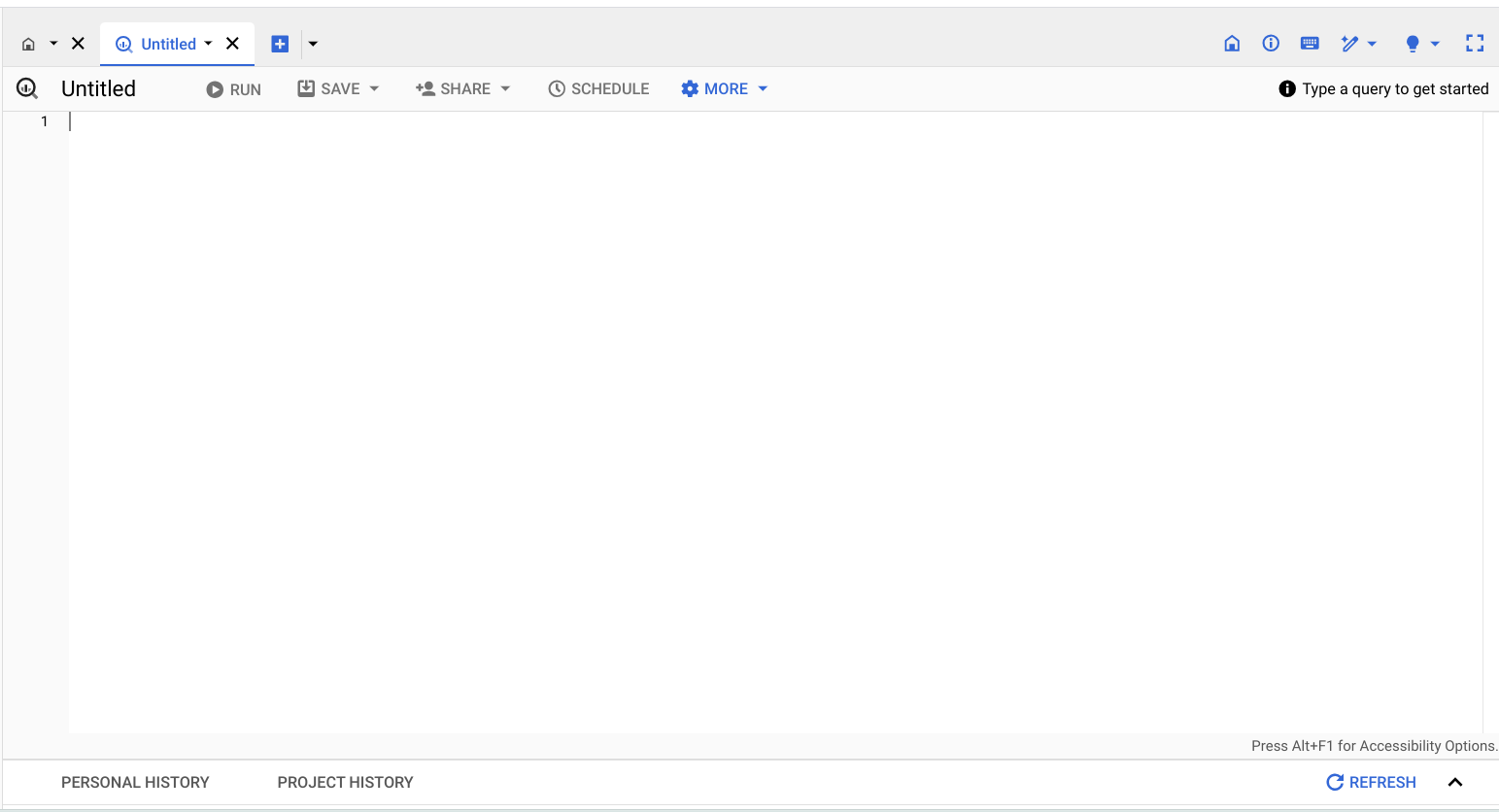
* **Schema**, which displays the column names in the dataset
* **Details**, which contains additional metadata, such as the creation date of the dataset
* **Preview**, which shows the first rows from the dataset



Additionally, you can select the **Query** button from the menu bar in the SQL Workspace to query this table.

### The SQL Workspace

The final menu pane in your console is the SQL Workspace. This is where you will actually write and execute queries in BigQuery.



The SQL Workspace also gives you access to your personal and project history, which stores a record of the queries you’ve run. This can be useful if you want to return to a query to run it again or use part of it in another query.

## Upload your data

In addition to offering access to public datasets, BigQuery also gives you the ability to upload your own data directly into your workspace. Access this feature by opening the **+ ADD** menu again or by clicking the three vertical dots next to your project’s name in the Explorer pane. This will give you the option to create your own dataset and upload your own tables. You will have the opportunity to upload your own data in an upcoming activity to practice using this feature!

## Key takeaways

BigQuery's SQL workspace allows you to search for public datasets, run SQL queries, and even upload your own data for analysis. Whether you're working with public datasets, running SQL queries, or uploading your own data, BigQuery’s SQL workspace offers a range of features to support all kinds of data analysis tasks. Throughout this program, you will be using BigQuery to practice your SQL skills, so being familiar with the major components of your BigQuery console will help you navigate it effectively in the future!

**Upload the movie dataset to BigQuery**

The next video demonstrates how to use SQL to filter data in a large dataset in BigQuery.

If you would like to follow along with the instructor, you will need to log in to your BigQuery account and upload the movie dataset provided as a .csv file.

**Prepare for the next video**

* First, download the .csv file from the attachment below:

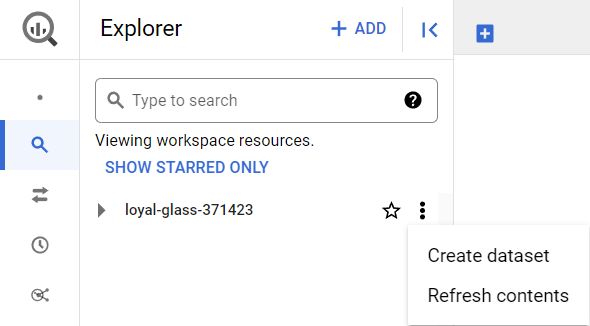
[Movie Data](https://d3c33hcgiwev3.cloudfront.net/KNFbyUKxRKiRW8lCsQSo8A_8adeac50825b4df68daaf6055c404ef1_Movie-Data.csv?Expires=1706832000&Signature=QhXfF2uFLIy-QodYHc3DMemj86zP580Ai4Dyb9HES4ShLQZ2B36oaYo8kgNbjHWVgRWL9y0KrQFLmTyEKQaMLCFLSdk~0FwCOA7K10vsfclZ5Vtt-fW5-0z-dvc4g1n7TBCESchU~AK~aj79bsbPzGjmSER9QdDSZ8zfbg3F5Js_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[CSV File](https://d3c33hcgiwev3.cloudfront.net/KNFbyUKxRKiRW8lCsQSo8A_8adeac50825b4df68daaf6055c404ef1_Movie-Data.csv?Expires=1706832000&Signature=QhXfF2uFLIy-QodYHc3DMemj86zP580Ai4Dyb9HES4ShLQZ2B36oaYo8kgNbjHWVgRWL9y0KrQFLmTyEKQaMLCFLSdk~0FwCOA7K10vsfclZ5Vtt-fW5-0z-dvc4g1n7TBCESchU~AK~aj79bsbPzGjmSER9QdDSZ8zfbg3F5Js_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

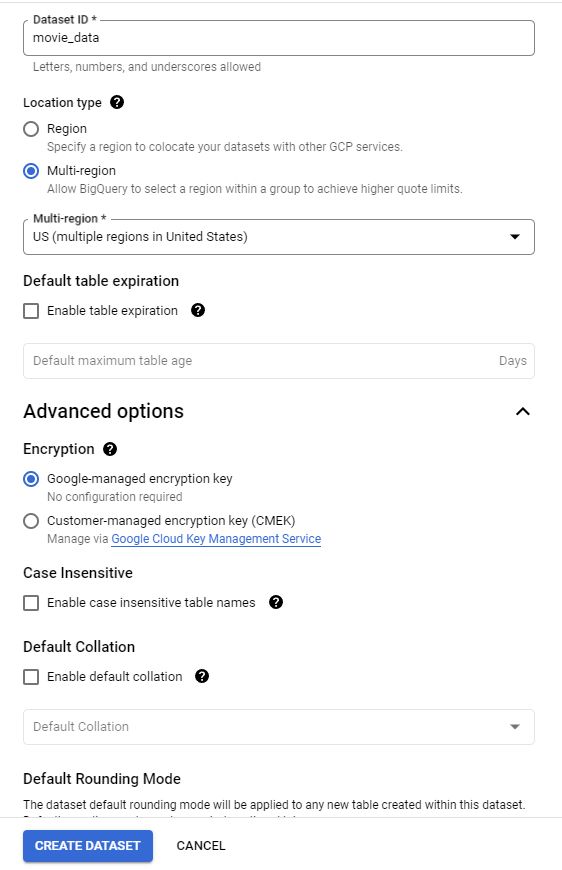
* Next, complete the following steps in your BigQuery console to upload the movie dataset.

**Step 1:** Open your BigQuery console and click on the project you want to upload the data to.

**Step 2:** In the **Explorer** on the left, click the **Actions** icon (three vertical dots) next to your project name and select **Create dataset**.

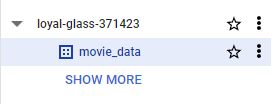


**Step 3:** In the upcoming video, the name **movie\_data** will be used for the dataset. If you plan to follow along with the video, enter **movie\_data** for the Dataset ID. In the *Location Type* section, select **Multi-region,** then from the *Multi-region* dropdown select **US (multiple regions in United States)**, and make sure the default *Encryption* method within the Advanced options is set to the **Google\_managed encryption key**.



**Step 4:** Click **CREATE DATASET** (blue button) to add the dataset to your project.

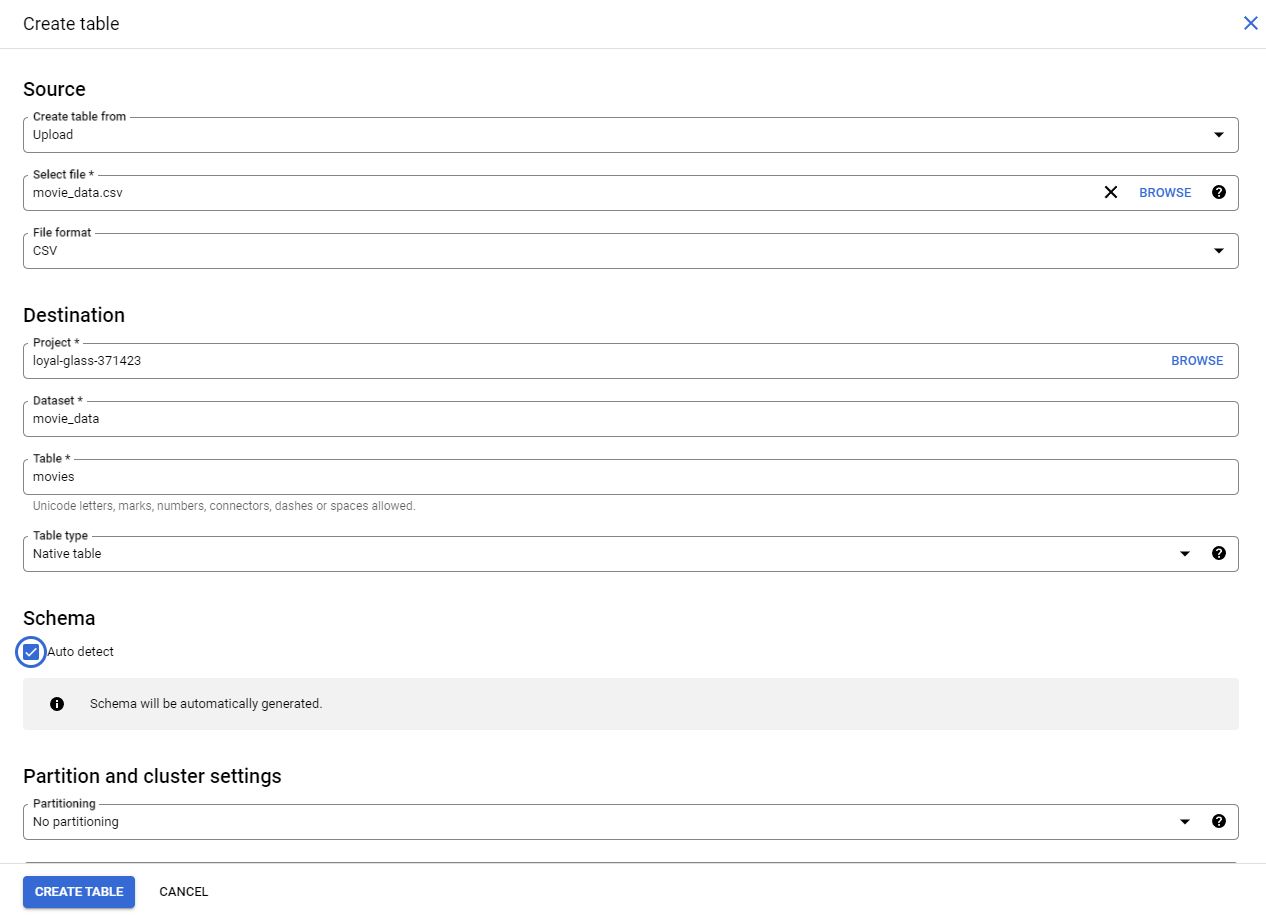
**Step 5:** In the **Explorer** on the left, click to expand your project and then click the **movie\_data** dataset you just created.



**Step 6:** A new window will appear titled **movie\_data,** and make a note of the **Dataset ID.**



**Step 7**: Towards the right hand side of the page, you will see a tab row of additional commands. Click the first blue button titled **+ CREATE TABLE** at the top right. A new window titled *Create table* will appear.



**Step 8:** Under *Source,* for the *Create table from* dropdown, select the data source.

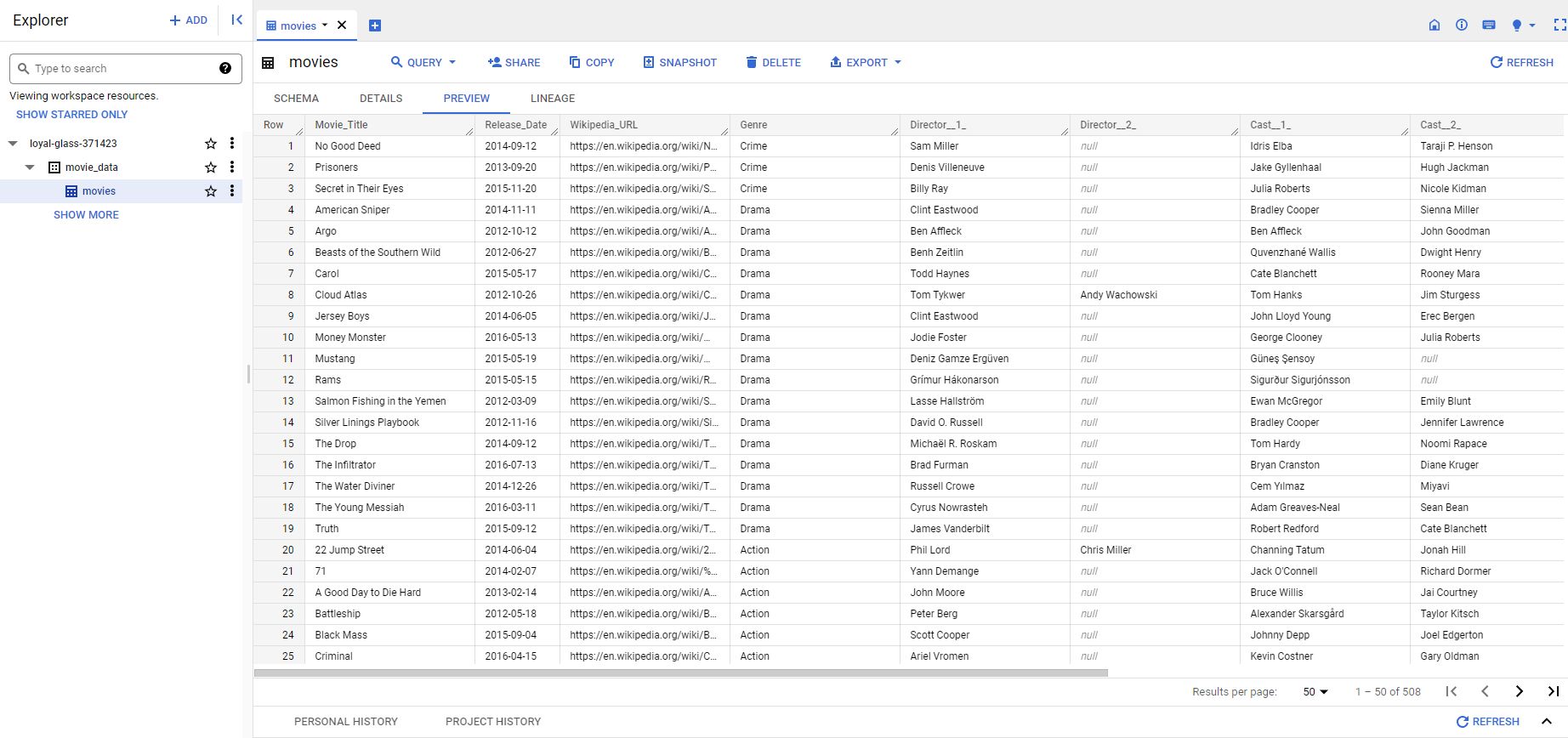
* Select **Upload**.
* Click **Browse** to select the **movie\_data.csv** file you downloaded.
* Select **CSV** from the file format dropdown, but BigQuery should automatically change the format.

**Step 9:** Under *Destination,* for the *Table* name, enter **movies** to match the table in the video.

**Step 10:** Under *Schema,* click the **Auto detect** checkbox.

**Step 11**: Once all of the settings have matched the image above, click the **CREATE TABLE** (blue button) at the bottom. You will now see the **movies** table under your **movie\_data** dataset in your project.

**Step 12:** Click **movies** and then select the **Preview** tab. Confirm that you have access to the appropriate table.



Congratulations, you are now ready to follow along with the next video.

**Step-by-Step: Filter data with SQL**

This reading outlines the steps the instructor performs in the following video, [Filter data with SQL](https://www.coursera.org/learn/analyze-data/lecture/Y5Nmb/more-on-sorting-and-filtering). In the video, the instructor demonstrates filtering data with SQL using **WHERE** clauses.

Keep this step-by-step guide open as you watch the video. It can serve as a helpful reference tool 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 instructor, you will need to log in to your BigQuery account and upload the Movies dataset. To do this, follow the instructions in the reading [**Upload the movie dataset to BigQuery**](https://www.coursera.org/learn/analyze-data/supplement/sBFZn/optional-upload-the-movie-dataset-to-bigquery).

empty alt text

**Example 1: Filter data in SQL**

Complete the following steps to use the **WHERE** clause to filter the database and narrow down the list to movies in the comedy genre.

1. In the BigQuery **Explorer pane**, select the **movie** dataset then the **movies** table.

2. Select the **Preview** tab from the **Details pane**.

3. Select **Query** then **In new tab** and enter the following code into the query editor:

1

2

SELECT \*

FROM `projectID.movie\_data.movies`

**Note:** If you’re completing this code in BigQuery, replace **projectID** in the code block to your own projectID.

4. Use the **WHERE** clause to filter the data. Enter **WHERE Genre = 'Comedy';** to filter for and select rows with 'Comedy' in the Genre column.

5. Your code should now match this code block:

1

2

3

SELECT \*

FROM `projectID.movie\_data.movies`

WHERE Genre = 'Comedy';

6. Select **RUN** to run the query. The results display a shorter list of movies, all in the comedy genre.

Hey, great to see you again.

Earlier we talked about why you should organize your data, no matter what part of

the lifecycle it's in.

Just like any collection, it's easier to manage and care for a group of things

when there's structure around them.

Play video starting at ::15 and follow transcript0:15

Now we should keep in mind that organization isn't just about making

things look orderly.

It's also about making it easier to search and

locate the data you need in a quick and easy way.

As a data analyst, you'll find yourself rearranging and

sifting through databases pretty often.

Two of the most common ways of doing this are with sorting and filtering.

We've briefly discussed sorting and filtering before, and

it's important you know exactly what each one does.

Sorting is when you arrange data into a meaningful order to make it easier to

understand, analyze, and visualize.

Sorting ranks your data based on a specific metric that you can choose.

You can sort data in spreadsheets and databases that use SQL.

We'll get to all the cool functions you can use in both a little later on.

A common way to sort items when you're shopping on a website is from lowest to

highest price, but you can also sort by alphabetical order,

like books in a library.

Or you can sort from newest to oldest,

like the order of text messages in a phone.

Or nearest to furthest away, like when you're searching for restaurants online.

Another way to organize information is with a filter.

Filtering is showing only the data that meets a specific criteria while

hiding the rest.

Typically you can use filters when you want to narrow down the amount of data you

want to sift through.

Say you're searching for green sneakers online. To save time, you filter for

green shoes only.

Using a filter slims down larger data sets to smaller subsets

that are relevant to what you need.

Sorting and filtering are two actions you probably perform a lot online.

Whether you're sorting movie showtimes from earliest to latest, or

filtering your search results to just images, you're probably already familiar

with how helpful they can be for making sense of data.

Now let's take that knowledge and apply it.

When it comes to sifting through large, disorganized piles of data,

filters are your friend.

You might remember from a previous video that you can use filters and

spreadsheet programs, like Excel and Sheets,

to only display data from rows that match the range or condition you've set.

You can also filter data in SQL using the WHERE clause. The WHERE

clause works similarly to filtering in a spreadsheet because it returns rows

based on a condition you name.

Let's learn how you can use a WHERE clause in a database.

We'll use BigQuery to access the database and run our query.

If you're joining us, open up your tool of choice for using SQL and

reference the earlier resource on how to access the dataset.

Otherwise, watch as the WHERE clause does its thing.

Here's the database.

Play video starting at :3:5 and follow transcript3:05

You might recognize it from past videos. Basically, it's a long list of movies.

Each row includes an entry for the columns named Movie\_Title,

Release\_Date, Genre, Director, Cast\_Members, Budget, and

Total\_Revenue. It also includes a link to the film's Wikipedia page.

If you scroll down the list, the list goes on for a long time.

Of course, we won't need to go through everything to find the data we want.

That's the beauty of a filter!

In this case, we'll use the WHERE clause to filter the database and

narrow down the list to movies in the comedy genre.

To start, we'll use the SELECT command followed by an asterisk.

In SQL, an asterisk selects all of the data.

On a new line, we'll type FROM and

the name of the database: movie\_data.movies.

To filter the movies by comedy, we're going to type WHERE,

then list the condition, which is Genre.

Play video starting at :4:5 and follow transcript4:05

Genre is a column in the dataset, and we only want to select rows where

the cell in the Genre column exactly matches "Comedy."

Next we'll type the equals sign and write the specific genre we're filtering for,

which is comedy.

Since the data in the Genre column is a string format,

we have to use single or double quotations when writing it.

And keep in mind that capitalization matters here,

so we have to make sure that the letter casing matches the column name exactly.

And now we can click Run to check out the results.

What we're left with is a shorter list of comedy movies. Pretty cool, right?

Here's something else you should know.

You can apply multiple filters to a database. You can even sort and

filter data at the same time for even more precise results.

As a data analyst, knowing how to sort and filter data will make you a superstar.

That's all for now. Coming up, we'll get down to the nitty-gritty of sorting

functions in spreadsheets. See you there!

### 1.

Question 1

Fill in the blank: The goal of analysis is to identify \_\_\_\_\_ within data in order to accurately answer questions and solve problems.

1 point

value and meaning

opportunities and risks

trends and relationships

all possible insights

### 2.

Question 2

A data analyst organizes a database to show only the 100 most recent real estate sales in Stamford, Connecticut. What steps do they take?

1 point

Filter out sales in Stamford, Connecticut, then sort the most recent sales at the top of the list.

Add a filter to return only sales in Stamford, Connecticut, then sort the most recent sales at the top of the list.

Add a filter to return only sales in Stamford, Connecticut, then sort the least recent sales at the top of the list.

Filter out sales in Stamford, Connecticut, then sort the least recent sales at the top of the list.

### 3.

Question 3

What term describes data points that are very different from similarly collected data and, therefore, might not be reliable values?

1 point

Outliers

Noise

Errors

Norms

### 4.

Question 4

In what way is the SQL **WHERE** clause similar to filtering in spreadsheets?

1 point

Both the **WHERE** clause and spreadsheet filters return a subset of data based on specified criteria.

Both the **WHERE** clause and spreadsheet filters can be used to create pivot tables.

Both the **WHERE** clause and spreadsheet filters enable data professionals to remove duplicate columns.

Both the **WHERE** clause and spreadsheet filters enable efficient sorting of data.

**Step-by-step: Sort datasets in spreadsheets**

This reading outlines the steps the instructor performs in the following video, [Sort datasets in spreadsheets](https://www.coursera.org/learn/analyze-data/lecture/6f6R0/sorting-datasets). In this video, the instructor demonstrates how to sort data in spreadsheets with the data menu.

Keep this step-by-step guide open as you watch the video. It can serve as a helpful reference tool 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 access the spreadsheet the instructor uses in this video, click the link to the dataset to create a copy. If you don’t have a Google account, you may download the data directly from the attachments below.

Link to movie data starter project: [Movie data starter project](https://docs.google.com/spreadsheets/d/1FLaUmMn62YlHYihV6pK1DJqWcFYCnuoqoxFWmm_o5b0/template/preview).

OR

[Movie Data Starter Project](https://d3c33hcgiwev3.cloudfront.net/NrzEyoWGRhix-2gQUzoZag_31090f90399f4347b98b63d02b95a9e1_Movie-Data-Starter-Project.xlsx?Expires=1706832000&Signature=DWguKVtPW5ICNZcGjodViXQutl7isDIBBieECQbFiL0fIrMlOKff0Xy2Zn-bta~zTtcRYYi1EVUAJPwi6M3bdGQf9cTMRHe8UVuPPbnl2oAsovTpHZUOcgc4zzg5MKmStzojVAOFbFVvNmJTf-u7UngDwJW4b7kXypgG-MFNb3w_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[XLSX File](https://d3c33hcgiwev3.cloudfront.net/NrzEyoWGRhix-2gQUzoZag_31090f90399f4347b98b63d02b95a9e1_Movie-Data-Starter-Project.xlsx?Expires=1706832000&Signature=DWguKVtPW5ICNZcGjodViXQutl7isDIBBieECQbFiL0fIrMlOKff0Xy2Zn-bta~zTtcRYYi1EVUAJPwi6M3bdGQf9cTMRHe8UVuPPbnl2oAsovTpHZUOcgc4zzg5MKmStzojVAOFbFVvNmJTf-u7UngDwJW4b7kXypgG-MFNb3w_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

empty alt text

**Example 1: Sort a sheet with the menu**

Use the menu to quickly and easily sort an entire sheet, keeping rows together.

1. Open the [Movie data starter project](https://docs.google.com/spreadsheets/d/1FLaUmMn62YlHYihV6pK1DJqWcFYCnuoqoxFWmm_o5b0/template/preview) spreadsheet.
2. Select cell **B** to highlight all of **column B**.
3. Select **Data** from the menu.
4. Select **Sort sheet**.
5. Select **Sort sheet by column B (A to Z)**.
6. Notice that the movies in your spreadsheet are now arranged in chronological order based on the release date.

**Note:** In the video, the sorted results are not entirely correct as shown. While the release dates appear to be sorted in ascending order, many of the movie titles are also in alphabetical order. As a data analyst, you may suspect that this is too much of a coincidence and that something is not quite right with the data. In this example, the movie titles in column A may have been previously sorted alphabetically and the data wasn’t restored to the original unsorted format.

**Example 2: Sort data in a specific column with the menu**

Use the menu to sort one column without affecting how the rest of the sheet is arranged. Each row in a table describes a single observation, so if you sort only one column you can introduce errors throughout your dataset. Use caution when using this option!

1. Select cell **A** to highlight all of **column A**.
2. Select **Data** from the menu.
3. Select **Sort range**.
4. Select **Sort range by column A (A to Z)**.
5. Notice that the movie data across the rows is now jumbled because sorting a single column in a sheet doesn’t keep data in a row together.

**Note:** As you’ve learned, each row in a table describes a single observation. Here, the values in Column A (movie titles) were sorted in A-to-Z order, but the rest of the sheet wasn’t sorted. For example, before sorting by column A, *The Devil Inside* was listed as having a release date of 2012-01-06. After sorting only the movie titles in Column A in A-to-Z order, however, *The Devil Inside* is listed as having a release date of 2015-06-16. This is incorrect.

Hey there, data pro!

Happy to see you back and ready to explore more of

the organizational side of data analysis.

In this video, we'll learn how to

sort data in spreadsheets.

We've done some sorting in

spreadsheets earlier in the program.

Now it's time to build on what we've covered and

introduce some more advanced sorting techniques.

Sorting is amazing.

Not only does it add

order and meaning to your spreadsheets,

it also gives you the power

to reimagine data altogether.

When you sort data based on a specific metric,

you can uncover new patterns and

relationships within datasets

you might not have otherwise noticed.

This is especially true for spreadsheets,

which you'll use a lot in your work as a data analyst.

Knowing how to sort data in spreadsheets can

make you a stronger and more confident analyst.

In many ways, sorting relies on your creativity

to reimagine the information you have in front of you.

In spreadsheets, you can sort data by ascending

or descending order using numbers or letters.

If cells are labeled with color,

you can sort them by color, too.

When sorting data in a spreadsheet,

you can choose to "Sort sheet" or "Sort range."

If "Sort sheet" is applied,

all of the data in a spreadsheet is

sorted by the conditions of a single column,

but the related information

across each row stays together.

On the other hand,

"Sort range" doesn't keep

the information across rows together.

When you sort a range,

you're selecting a specific collection of

cells or the range that you want the sorting limited to.

Nothing else on the spreadsheet gets

rearranged but the specified cells.

There's two methods for sorting spreadsheet data:

one involves using the menu;

the other involves writing out the sort function.

For now, we'll focus on sorting with the menu.

We'll get to writing out functions later on.

Now, depending on the program you use,

the process might seem slightly different,

but the instructions and concepts we

discuss will be basically the same.

Back to sorting with the data menu.

To give you an idea of how to do it,

we'll use the movie spreadsheet. Let's check it out.

In this example, we'll sort movies by release date.

We'll head to column B,

which is listed as "Release Date."

Click on the "B" button to

highlight all the cells in the column.

From there, we'll head to the Data tab in the menu.

Now you have two choices:

sort a sheet or a range of data.

You'll notice that we've selected just the release dates,

but these release dates are specifically

related to the movies in their row.

Play video starting at :2:46 and follow transcript2:46

In this case, you want

the release date and the movie title to

stay in the same row as you sort because they're related.

To do this, you'll want to "Sort sheet."

This will keep all the data together by row,

no matter how you sort it.

Depending on the order you

want the release dates to be in,

you can sort from A to Z,

which will also rank the dates

numerically. Or you can sort from Z to A,

which will sort data the opposite way.

Since we want the release dates to be in order,

we'll click "Sort sheet by

column B" from A to Z. And there you go.

You just sorted a sheet of data using the menu.

Now the movies are arranged in

chronological order based on release date.

Let's say you want to sort data in a specific column,

but don't need the cells in that column

tied to a specific row of information.

Instead, you want to isolate the column's data and sort it

on its own without affecting

how the rest of the sheet's arranged.

For fun, we'll use

the Movie Title column in this example.

First, we'll select the column we want to

sort: column A. Clicking

on column A highlights all the cells

in the column which contain the movie titles.

Then we'll go to the menu and click

Data because we're isolating

the column from the rest of the sheet.

When we sort this time,

we'll click "Sort range by column A."

For this example, we'll sort the movie titles

alphabetically from A to Z. And that's it!

You'll notice that "Sort range"

doesn't keep the rows together,

so the data are a bit jumbled.

You'll probably end up using "Sort sheet" more often,

but it's important to understand them both

so you don't accidentally get them mixed up.

You've just sorted data in a spreadsheet

using the menu, and you've learned how

to sort data by an entire sheet or by a range of cells.

That's something you'll be able to take with you

wherever you go as a data analyst.

Coming up, we'll learn about the second way to sort in a

spreadsheet: by writing out a function.

We'll also take sorting to the next level by

custom sorting your data. See you there!

**Step-by-step: Use the SORT function in spreadsheets**

This reading outlines the steps the instructor performs in the next video, [Use the SORT function im spreadsheets](https://www.coursera.org/learn/analyze-data/lecture/K6WB8/the-sort-function). In this video, the instructor demonstrates how you can use the **SORT** function to arrange data into a meaningful order to make it easier to understand, analyze, and visualize.

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: [Party plan spreadsheet](https://docs.google.com/spreadsheets/d/1L1Z6b3X9WCpwzisHcxvjC5-Qett-BOMo9yg8Cz36NME/template/preview)

OR

[Party Plan Spreadsheet](https://d3c33hcgiwev3.cloudfront.net/SifBvA71QgKf-sX-0JyvSw_7a45f15631e54e65aa3541c76da0d3e1_Party-Plan-Spreadsheet.xlsx?Expires=1706832000&Signature=JzaoTUFcISMN7D7ca3rc8ALagyS4-NwU0r4J4tCHlNBhCoZ0Sie5L0rBJj6-IvL1TCD~EQ0P7BPPHqo5udk4VvdZheDJhU6n4p0Xg19VUUskMM51VruQSHoGZoRbGC6N1qPfYY~GUu881BHZki-wFfp1hk44~JoQeJRnaIno1H8_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[XLSX File](https://d3c33hcgiwev3.cloudfront.net/SifBvA71QgKf-sX-0JyvSw_7a45f15631e54e65aa3541c76da0d3e1_Party-Plan-Spreadsheet.xlsx?Expires=1706832000&Signature=JzaoTUFcISMN7D7ca3rc8ALagyS4-NwU0r4J4tCHlNBhCoZ0Sie5L0rBJj6-IvL1TCD~EQ0P7BPPHqo5udk4VvdZheDJhU6n4p0Xg19VUUskMM51VruQSHoGZoRbGC6N1qPfYY~GUu881BHZki-wFfp1hk44~JoQeJRnaIno1H8_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

empty alt text

**Example 1: Sort guests by table**

Use a spreadsheet function to sort guests by the table to which they’re assigned.

1. Open the [Party plan spreadsheet](https://docs.google.com/spreadsheets/d/1L1Z6b3X9WCpwzisHcxvjC5-Qett-BOMo9yg8Cz36NME/template/preview).
2. Select cell G1 and enter the equal sign **=**.
3. Enter **SORT** followed by an open parenthesis **(** to begin the **SORT** function.
4. Enter the first cell from the Guest column of the spreadsheet, **A2**, followed by a colon [**:**], then enter the last cell from the Sent Invitation column, **D6**, followed by a comma. **A2:D6** is the range of cells over which this function will sort.
5. Enter **2** followed by a comma to specify the column by which to sort the data. Note that the function doesn't recognize letters, so use the column’s number. Column A corresponds to 1, B corresponds to 2, and so on.
6. Enter **TRUE** followed by a close parenthesis **)**. **TRUE** means the function will return results in ascending order, so the tables will be listed starting with table one. **Note:** To return results in descending order, enter **FALSE**.
7. Press **Enter** (Windows) or Return (Mac).

The party guests are now sorted by the table to which they’re assigned.

**Example 2: Customized sort order**

When you sort data in a spreadsheet using multiple conditions with a customized sort order, data is sorted based on the order in which the conditions are applied to the data. To sort party guests by whether or not they've been sent an invitation and to list the guests alphabetically:

1. Highlight all the data in the party plan spreadsheet: **cells A1 to D6**.
2. In the menu, select **Data > Sort range > Advanced range sorting options**. This opens the **Sort range** from A1 to D6 dialog box.
3. Check the **Data has a header row** checkbox to make sure column titles aren’t included in the sorted data.
4. From the **Sort by** drop-down list, select **Sent Invitation**.
5. Select the **A to Z** radio button to make sure the *No* responses are listed first and the *Yes* responses are listed second.
6. Select **Add another sort column** to add the additional sorting condition.
7. From the **Sort by** drop-down list, select **Guest Names** to list guests alphabetically.
8. Select the **A to Z** radio button.
9. Select **SORT**.

This returns your customized sort order, which lists the *No* invitations and those guests alphabetically, followed by the *Yes* invitations and those guests alphabetically.

Happy to have you back.

Earlier in the program, we covered some basics of sorting in spreadsheets.

We learned the differences between sorting a range and an entire sheet, and

how to sort a spreadsheet using the menu.

Now that we've laid the groundwork,

it's time to move on to more advanced ways to sort information.

We've talked about how there's two methods of sorting data in spreadsheets.

The first method uses the Data tab in the menu of your spreadsheet program.

The second way to store information in a spreadsheet is by writing a SORT function.

In spreadsheets, functions are preset commands that perform a specific process.

So in this case, the SORT function, as you might be able to guess, sorts your data.

Play video starting at ::43 and follow transcript0:43

Let's check out this spreadsheet of party plans to witness the SORT function

in action.

The first arranged set of data is our original dataset of guests and

some information about them.

So let's say you want to sort the party guests by table to get an idea of who will

be sitting where. To do that, start by typing a function in an empty cell.

Just like any function, you do this by typing the equal sign, and

then write SORT after it.

Play video starting at :1:15 and follow transcript1:15

After your first open parenthesis, reference the first cell in which data is

collected from. In this case, that's A2.

Play video starting at :1:28 and follow transcript1:28

Then you'll include a colon and

write the last cell you want included in the function, which is D6.

Play video starting at :1:41 and follow transcript1:41

A2 colon D6 is the range for this function.

Next, write a comma to separate the range from what we're sorting by,

which is column B.

Play video starting at :1:57 and follow transcript1:57

You should keep in mind that this part of the function doesn't recognize

column letters.

So in this case, we use the corresponding number instead, which is 2,

since column B is the second column in our range.

Play video starting at :2:15 and follow transcript2:15

Now add another comma.

Play video starting at :2:19 and follow transcript2:19

In this next part you'll need to decide whether you want the data in

this column to be in ascending or descending order.

A TRUE statement is in ascending order, and FALSE is descending.

Because we want the tables to be listed starting from table number one, we'll

write TRUE for ascending, and then end the function with a closed parenthesis.

Now, let's see our function play out.

Play video starting at :2:46 and follow transcript2:46

Our party guests are now sorted by which table they're seated.

Once you have an idea of the data you want to be sorted and how,

applying functions to your data is simple.

Now, you have two different tools in your tool belt for sorting data.

After you've tackled writing SORT functions,

you'll want to customize sort orders, too.

A customized sort order is when you sort data in a spreadsheet using

multiple conditions.

This means that sorting will be based on the order of the conditions you select.

Let's go back to our party spreadsheet.

Imagine you want the guests to be sorted by whether or

not they've been sent an invitation.

And based on that, we want those guest names to be listed alphabetically.

You can do that easily with the "Sort range" option under Data.

Play video starting at :3:34 and follow transcript3:34

First, highlight all the data in the set from cells A1 to D6.

Then under the Data tab in the menu, click "Sort range."

Play video starting at :3:46 and follow transcript3:46

In this case, check "Data has a header row,"

which makes sure that the title of the column isn't mixed into the sorting.

Then, we'll make sure it's being sorted by "Sent invitation."

Here, we want the "No" responses first and the "Yes" responses second,

so we'll make sure A to Z is clicked to sort the responses in that order.

Because we want to add an additional sorting condition,

we'll now click on "Add another sort column."

The guest names should be in alphabetical order.

So let's select "Guest Names" and sort from A to Z.

Play video starting at :4:27 and follow transcript4:27

Then we'll click Sort. And voilà!

You've officially applied a custom sort order like a champ.

Play video starting at :4:37 and follow transcript4:37

Okay, so you've tackled sorting in spreadsheets by sheet, by range,

through the menu, and by using a function. On top of all that, you've

added to your organizational skills by learning how to create custom sort orders.

Pretty soon you'll learn another powerful tool: how to sort data using SQL.

Even though databases can sometimes be a lot to digest, learning these skills gives

you the power to rearrange data in a way that makes sense to you.

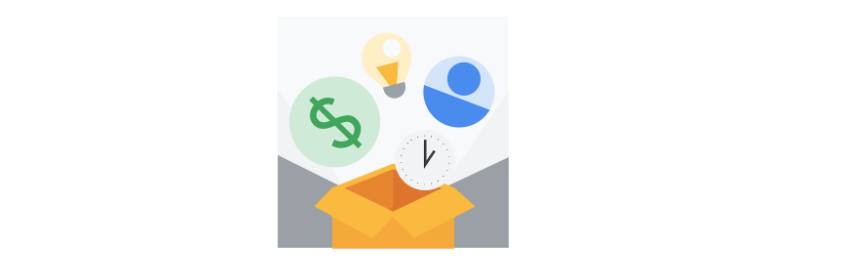
Once you've sorted data in a way that really clicks,

you'll understand why it's so valuable to you as a data analyst.

Bye for now!

**Sort and filter in Sheets and Excel**

In this reading, you’ll examine the sorting and filtering options in Google Sheets and Microsoft Excel. Both offer basic sorting and filtering from set menu options. But if you need more advanced sorting and filtering capabilities, you can use their respective **SORT** and **FILTER** functions.



**Sort and filter in Sheets**

Sorting in Google Sheets helps you quickly spot trends in numbers and text. For example, as the vice president of sales at a candy company, you want to improve chocolate sales in lower-performing regions—your company makes delicious chocolate and you know sales can improve. As a first step, you examine this by calculating gross (total) revenue of chocolate by sales region. In this case, you could sort the gross revenue column in **descending** (Z to A) order to find the top performing regions at the top, or sort the gross revenue column in **ascending** (A to Z) order to find the lowest performing regions at the top. Then, you can look at patterns in the best and worst regions to explore how to increase sales in the lower-performing regions.

If you want to learn more about the set menu options for sorting and filtering, start with these resources:

* [Sort and filter data](https://support.google.com/docs/answer/3540681) (Google Help Center): instructions to sort data in alphabetical or numerical order and create filter views
* [Sort data by selecting a range of data in a column](https://www.youtube.com/watch?v=VcRBHXBMKBU): video of steps to achieve the task
* [Sort a range of data using sort criteria for multiple columns](https://blog.sheetgo.com/google-sheets-formulas/sort-formula-google-sheets/): technical tip instructions by SheetGo company to sort data across multiple columns

In addition to the standard menu options, the **SORT** function allows you to do more advanced sorting. Use this function to create custom sorting rules. You can sort the rows of a given range of data by the values in one or more columns. And you can set the sort criteria for each column. Refer to the [SORT function](https://support.google.com/docs/answer/3093150?hl=en) page for the syntax.

Like the **SORT** function, use the [FILTER function](https://support.google.com/docs/answer/3093197?hl=en) to filter by any matching criteria you like. This creates a custom filter.

As you’ve learned, you can filter data and then sort the filtered results. Using the **FILTER** and **SORT** functions together in a range of cells can programmatically and automatically achieve these results for you.

**Sort and filter in Excel**

You can also sort in ascending (A to Z) and descending (Z to A) order in Microsoft Excel. Excel offers **Smallest to Largest** and **Largest to Smallest** sorting options when you’re working with numbers.

Similar to the **SORT** function in Google Sheets, Excel includes custom sort capabilities that are available from the menu. After you select the data range, click the **Sort & Filter** button to select the criteria for sorting. You can even sort by the data in rows instead of by the data in columns if you select **Sort left to right** under **Options**. (**Sort top to bottom** is the default setting to sort the data in columns.)

If you want to learn more about sorting and filtering in Excel, start with these resources:

* [Sort data in a range or table](https://support.microsoft.com/en-us/office/sort-data-in-a-range-or-table-62d0b95d-2a90-4610-a6ae-2e545c4a4654) (Microsoft Support): instructions to perform sorting in a variety of use cases
* [Excel training: sort and filter data](https://support.microsoft.com/en-us/office/video-sort-data-in-a-range-or-table-ffb9fcb0-b9cb-48bf-a15c-8bec9fd3a472#ID0EAABAAA=Transcript) (Microsoft Support): sorting and filtering videos with transcripts
* [Excel: sorting data](https://www.youtube.com/watch?v=Ep5q1cUhQas): video of how to use the **Sort & Filter** and **Data** menu options for sorting

Excel also has [SORT](https://support.microsoft.com/en-us/office/sort-function-22f63bd0-ccc8-492f-953d-c20e8e44b86c), [SORTBY](https://support.microsoft.com/en-us/office/sortby-function-cd2d7a62-1b93-435c-b561-d6a35134f28f), and [FILTER](https://support.microsoft.com/en-us/office/filter-function-f4f7cb66-82eb-4767-8f7c-4877ad80c759) functions. Explore how you can use these functions to automatically sort and filter your data in spreadsheets without having to select any menu options at all.

**Sort and filter manually with menus and buttons**

Both Sheets and Excel feature menu options designed to let you sort and filter without using functions. For example, after selecting the data you’d like to sort in Google Sheets, you can choose **Data > Sort sheet** or **Data > Sort range** to sort that data. To filter the data, select all the columns and rows and choose **Data > Create a filter**. In Excel, you can use the **Sort & filter** button to bring up a user-friendly interface that guides you through sorting or filtering.

Finally, when using menus or buttons, here are a couple of best practices:

* Back up or make a copy of your data before making major changes.
* When filtering data, keep in mind that other users may also be accessing the spreadsheet. For example: Filters in Google Sheets can affect all viewers, so you should use filter views for personal filtering.

**Key takeaways**

As you’ve learned, you can sort and filter data in Google Sheets and Excel with functions or by using menus and buttons. Sorting data is the process of arranging data into a meaningful order to make it easier to understand, analyze, and visualize. This can help you identify trends in the data. Filtering is the process of showing only the data that meets a specified criteria while hiding the rest. Once you’ve filtered data, you can sort it to find trends within those criteria. Menus and buttons offer the ability to do basic sort and filter functions, but you’ll need to use a function for custom sorting and filtering.

I'm Emma, and I'm a product analyst on Google Health.

The product that I am helping

analyze data for is a tool for clinicians,

and what this is going to allow clinicians to do is find

their patients' health data just as easily

as you can find data on Google Search.

What I specifically focus on

is standardizing health care data,

analyzing it to find anomalies or data quality issues,

and talking to product managers

about which features we should launch and why.

I've worked on problems from

solving locomotive failures in

predicting them before they happen,

to forecasting sales for

handbags before the handbag is even launched,

to now working with health care data to try to

allow clinicians to easily have

their patients' data at their fingertips.

I was really interested in working in data analytics,

but I was trying to figure out

what type of data I wanted to work with

or what field within

data analytics I wanted to

focus on because there are so many.

I was ultimately drawn

towards working with health care data.

I just really fell in love with all of

the problems that we have today in

health care and how just this vast amount

of data that's available in

the health care industry could be better

utilized to help patients,

to help clinicians, to improve population health.

Working with meaningful datasets is what really makes me

excited to come to work every day and

excited to solve these problems.

What I've found in my career is

that following the datasets that interested me,

the types of problems that interested me,

always yielded better results because

I was just that much more driven to go

to work every day to do my best to solve

these interesting problems because

it's just what interested me.

The amazing thing is there's data everywhere,

there are data problems at every company in every field,

and you really get to just

follow what you're passionate about.

### 1.

Question 1

Which menu function sorts all spreadsheet data by the ranking of a specific sorted column?

1 point

Sort range

Sort by rank

Sort sheet

Sort data

### 2.

Question 2

When using the **SORT** function in spreadsheets, which statement will return the data in ascending order?

1 point

**FALSE**

**REVERSE**

**ASCEND**

**TRUE**

### 3.

Question 3

Fill in the blank: A customized sort order involves sorting data in a spreadsheet using \_\_\_\_\_ conditions.

1 point

multiple

random

interactive

statistical

### 4.

Question 4

Why would a data professional select **Data has a header row** when sorting data in a spreadsheet?

1 point

To ensure the first row of data is treated as header labels instead of data points

To remove duplicate rows from the sorted data

To insert a new row above the first row of the spreadsheet prior to sorting

To hide the first row of the spreadsheet in order to better organize the sort

**Step-by-Step: Sort data with SQL**

This reading outlines the steps the instructor performs in the following video, [Sort data with SQL](https://www.coursera.org/learn/analyze-data/lecture/P6Yu3/sorting-queries-in-sql). In the video, the instructor sorts data with SQL using the **ORDER BY** command. Then, the instructor uses **WHERE** and **ORDER BY** together to filter and sort data.

Keep this step-by-step guide open as you watch the video. It can serve as a helpful reference tool 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 instructor, you will need to log in to your BigQuery account and upload the Movies dataset. To do this, follow the instructions in the reading [**Upload the movie dataset to BigQuery**](https://www.coursera.org/learn/analyze-data/supplement/sBFZn/optional-upload-the-movie-dataset-to-bigquery).

empty alt text

**Example 1: Sort data by one column**

The **ORDER BY** command sorts data by column in a database. By default, the data is sorted in ascending order.

1. In the BigQuery **Explorer pane**, select the movie **dataset**, then the **movies** table.

2. Select the **Preview** tab from the **Details pane**.

3. Select **Query** then **In new tab** and enter the following code into the query editor:

1

2

SELECT \*

FROM 'projectID.movie\_data.movies'

**Note:** If you’re completing this code in BigQuery, replace **projectID** in the code block to your own projectID.

4. Use the **ORDER BY** command to sort the data. Enter **ORDER BY Release\_Date;** to sort by the Release\_Date column.

5. Your code should now match this code block:

1

2

3

SELECT \*

FROM 'projectID.movie\_data.movies'

ORDER BY Release\_date;

6. Select **RUN**. The results return all the movies in the database sorted from oldest to newest.

**Example 2: Sort data in descending order**

To use the **ORDER BY** command to sort data by descending order, specify **DESC** at the end of the **ORDER BY** command.

1. Enter **DESC** after **ORDER BY Release\_Date** in the query editor.

2. Your code should now match this code block:

1

2

3

SELECT \*

FROM 'projectID.movie\_data.movies'

ORDER BY Release\_Date DESC;

3. Select **RUN**. The results include the same list of movies, this time sorted from newest to oldest.

**Example 3: Filter and sort data in descending order**

Use **WHERE** and **ORDER BY** together to filter, then sort, data.

1. Select the beginning of the **ORDER BY** line and press Enter (Windows) or Return (Mac) to add a line between the **FROM** and **ORDER BY** clauses. The **ORDER BY** command is written on the last line to ensure all data is sorted.

2. Select the line you added. Enter **WHERE Genre = "Comedy"** to filter for rows in which the Genre column exactly matches **"Comedy"**.

3. Your code should now match this code block:

1

2

3

4

SELECT \*

FROM 'projectID.movie\_data.movies'

WHERE Genre = "Comedy"

ORDER BY Release\_Date DESC;

4. Select **RUN** to run the query. The results include only comedy movies sorted from newest to oldest.

**Example 4: Filter on two conditions, then sort data in descending order**

Use **WHERE**, **AND**, and **ORDER BY** to filter data on two conditions and then sort it.

1. Select the beginning of the **ORDER BY** line and press Enter (Windows) or Return (Mac) to add a line between the **WHERE** and **ORDER BY** clauses.

2. Select the line you added. Enter **AND Revenue > 300000000** to add a condition to your query to return only rows where the Revenue column is greater than 300,000,000.

3. Your code should now match this code block:

1

2

3

4

5

SELECT \*

FROM 'projectID.movie\_data.movies'

WHERE Genre = "Comedy"

AND Revenue > 300000000

ORDER BY Release\_Date DESC;

4. Select **RUN**. The results are sorted from newest to oldest and include only comedy movies with revenues greater than $300,000,000.

Hello there! If you're hoping to learn

about sorting—in SQL this time—

you've definitely come to the right place.

So far, we've sorted spreadsheets

through the menu and with a written function.

Which brings us to the next part of our learning:

more sort functions, but this time in SQL.

Data analysts love playing

with the way data is presented.

Sorting is a useful way to rearrange data because it can

help you understand the data

you have in a different light.

As you've probably already noticed,

a lot of things you can do in

spreadsheets can also be done in SQL.

Sorting is one of those things.

We've talked about using SQL with large datasets before.

When a spreadsheet has too much data,

you can get error messages,

or it can cause your program to crash.

That's definitely something we want to avoid.

SQL shortens processes that would otherwise take

a very long time or be

impossible to complete in a spreadsheet.

Personally, I use SQL to

pull and combine different data tables.

It's much quicker than

a spreadsheet, and that usually comes in handy.

Here's something pretty helpful you can do with SQL.

You can use the ORDER BY clause

to sort results returned in a query.

Let's go back to our movie spreadsheet to

get a better idea of how this works.

Feel free to follow along in

a SQL tool of your choice as we go.

As a quick refresher,

we have a database of movies listed with

data like release date, director, and more.

We can sort this table in lots of

different ways using the ORDER BY function.

For this example, let's sort by release date.

First, we have the SELECT function and an asterisk.

Play video starting at :1:51 and follow transcript1:51

Keep in mind that the asterisk

means all columns are selected.

Then we have FROM and the name of

the database and table we're in right now.

Now let's check out the next line.

It's empty, but that's where we'll

write our ORDER BY function.

The ORDER BY command is

usually the last clause in your query.

Back to the actual sorting!

We'll type ORDER BY with the space.

With this clause, you can choose to

order data by fields in a certain column.

Because we want to sort by release date,

we'll type Release\_Date.

By default, the ORDER BY

clause sorts data in ascending order.

If you run the query as it is right now,

the movies will be sorted from

oldest to the most recent release dates.

Let's run the query and see what we've got.

You can also sort the release dates in

the reverse order from

the most recent dates to the oldest.

To do this, just specify

the descending order in the ORDER BY

command written as DESC,

D-E-S-C. Let's run this query.

Play video starting at :3:6 and follow transcript3:06

As you'll notice, the most recently released films

are now at the top of the database.

In spreadsheets, you can combine sorts and

filters to display information differently.

You can do something similar in SQL too.

You might remember that while sorting

puts data in a specific order,

filters narrow down data,

so you only see data that fits the filter.

For example, let's say we want to filter movies by

genre so that we're only working with comedies.

But we still want release dates to be

sorted in descending order,

from most recent to oldest films.

We can do this with the WHERE clause.

Let's try that now.

First, we'll check that the ORDER BY

clause is always the last line.

That makes sure that all the results of

the query you're running are sorted by that clause.

Then, we'll add a new line for the WHERE clause

after FROM and before ORDER BY.

Play video starting at :4:9 and follow transcript4:09

Here's what we've got so far.

From there, we want to type

the column we're filtering for.

In this case, we want to

filter the database for comedies.

After the WHERE clause,

we'll type the column list's name as Genre.

Now, we'll add an equal sign after Genre because we

only want to include genres that

match what we're filtering for.

In this case, we're filtering for comedy,

so we'll type Comedy between two apostrophes.

Now, if you check out the entire query as a whole,

you'll notice that we're selecting all columns,

and we know it's all columns

because that's what an asterisk means.

The FROM clause specifies

the name of the movie database we're using,

and the WHERE clause filters the data to include

entries whose genre is specified as comedy.

Then in the last line,

we have the ORDER BY clause,

which will sort the data we've chosen to filter

by release dates in descending order.

This means when we run the query,

we'll only have comedy movies listed

from newest releases to oldest releases.

Let's run it and figure out if that's the case.

Play video starting at :5:25 and follow transcript5:25

Cool. Check out all those comedy movies

and the way those dates are sorted.

Play video starting at :5:33 and follow transcript5:33

Now, let's take this query a step further.

We'll filter for two conditions at

once using the AND filter.

Working off the query we've been using,

we'll add a second condition in the WHERE clause.

We'll keep the sorting the same.

Let's say you wanted to filter by comedy movies and

movies that earned over 300 million in the box office.

In this case, after the AND function,

you'd add the revenue condition by typing Revenue.

From there, you'll specify that you only want to return

films with revenues over $300 million.

To do that, type the greater than sign

and then the complete number of

300 million without commas.

Now let's run the query.

Play video starting at :6:23 and follow transcript6:23

Here, the data only shows comedy movies with revenues of

over $300 million, and it's

sorted in descending order by release date.

It looks really good.

You just filtered and sorted

a database like it's your job.

And with practice, one day it can be.

Just like that, you've finished

another step in your data analyst journey.

By now, you really dug and learned

about the analysis process with

a special emphasis on how organization

can change how you go through your data.

You also learned about both spreadsheets and SQL,

and how to sort and filter data

in both types of programs.

To help you get more comfortable

using spreadsheet and SQL features,

you'll be getting some materials you

can use as a resource.

Coming up, we'll check out how

an organizational mindset can take

your analytical skills even further.

We'll also cover converting, formatting,

and adjusting data to combine

information in a way that makes sense.

Learning those skills early on

can make your work as a data analyst much more

efficient and effective in the long run. See you soon.

**Scenario**

You’re a public health researcher with a state government agency. For your current project, you need to identify counties in the United States that have the most and least births in the 2016-2018 time frame. To do this, you’ll complete the following steps:

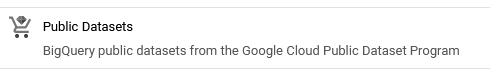
* Load the dataset
* Query the data to explore its structure
* Use **ORDER BY** to sort relevant data
* Use sorted data to answer questions

**Load the CDC Births Data dataset**

1. Open the BigQuery console.

2. Select **+ADD** from the **Explorer** pane.

3. In the **Add** window, navigate to and then select **Public Datasets**.



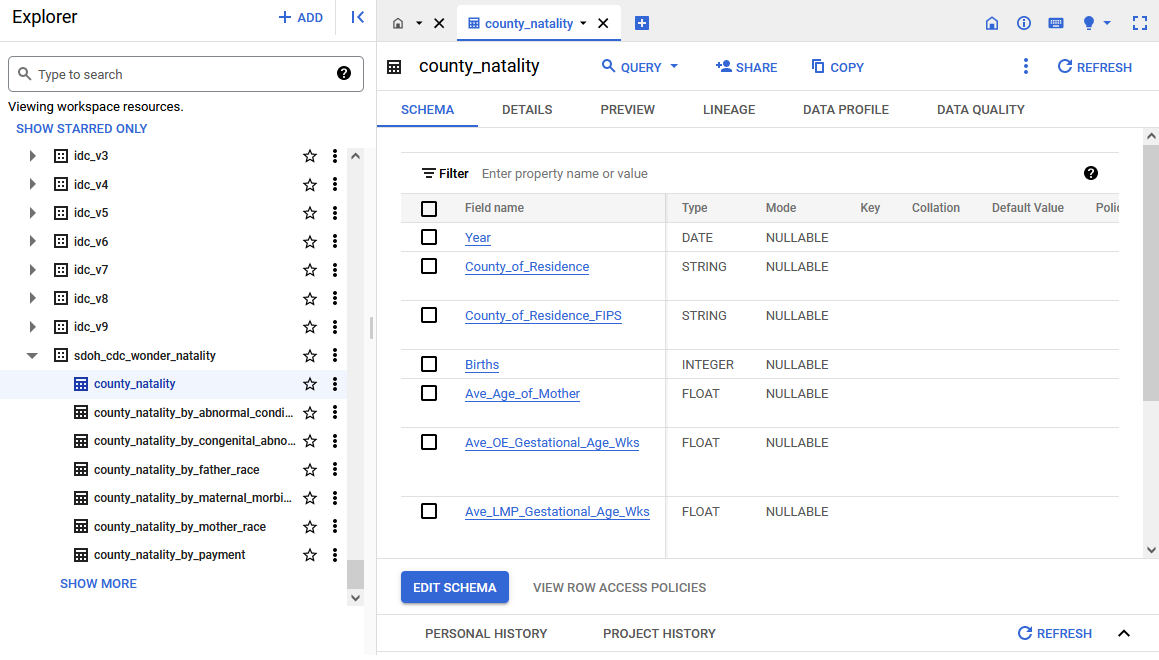
4. In the Marketplace search bar, enter **sdoh\_cdc\_wonder\_natality** and press enter.

5. Select the result **Births Data Summary** from the CDC.

6. Select **VIEW DATASET**. This will bring you back to the console and open a **Dataset info** tab about the CDC dataset in the **Details** pane.

7. Select **sdoh\_cdc\_wonder\_natality** in the **Explorer** pane to examine the tables available within the dataset.

8. Select the table **county\_natality** and explore this table’s schema, details, and preview.



**Query the data to explore its structure**

Now, it’s time to start working with the CDC births data. First, run a query to examine the dataset without sorting it.

1. Select **Query**, then **In new tab**.

2. Enter the following query into the Query Editor to display the first 1,000 rows of the **county\_natality** table.

1

2

3

4

5

6

SELECT

  \*

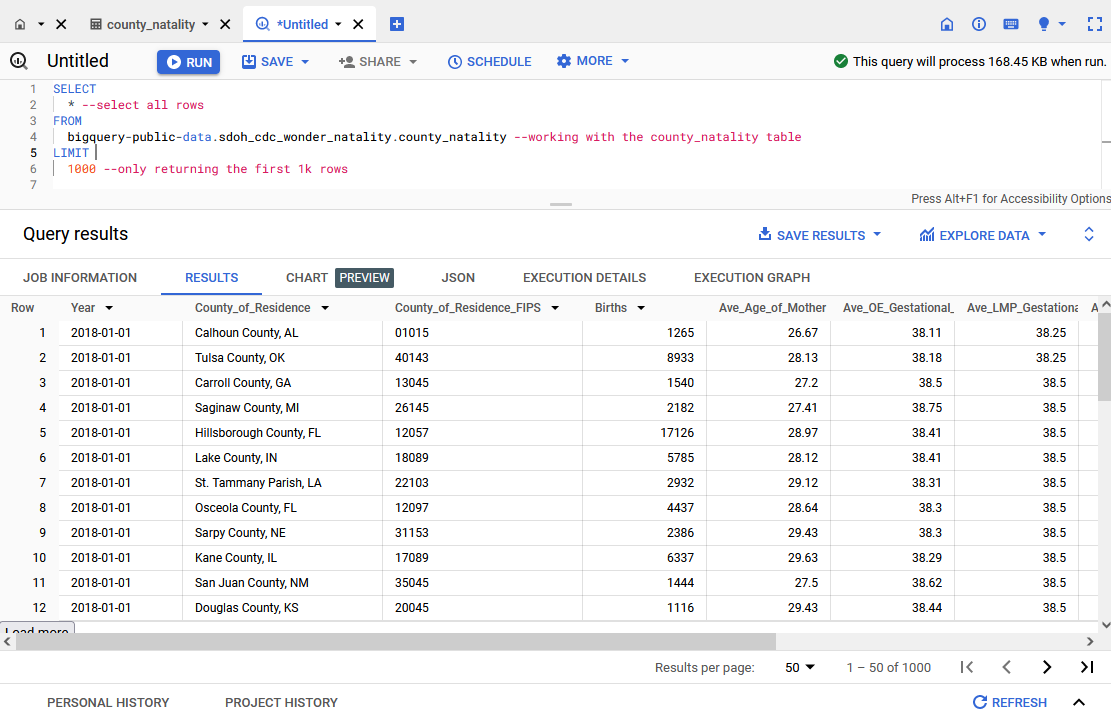
FROM

  bigquery-public-data.sdoh\_cdc\_wonder\_natality.county\_natality

LIMIT

  1000

3. Select **RUN**.

A screenshot of the Query results pane. The data returned include Year, County\_of\_Residence, County\_of\_Residence\_FIPS, Births, Avg\_Age\_of\_Mother, Ave\_OE\_Gestationa, and Ave\_LMP\_Gestational.

Examine the dataset you just loaded. Take a moment to familiarize yourself with the columns and fields available.

**Use ORDER BY to sort relevant data**

Now, sort the data with SQL’s **ORDER BY** function. Enter the following query into the Query Editor. The text preceded by two hyphens (--) are comments that explain the code. Run the Query.

1

2

3

4

5

6

7

8

SELECT

  \*

FROM

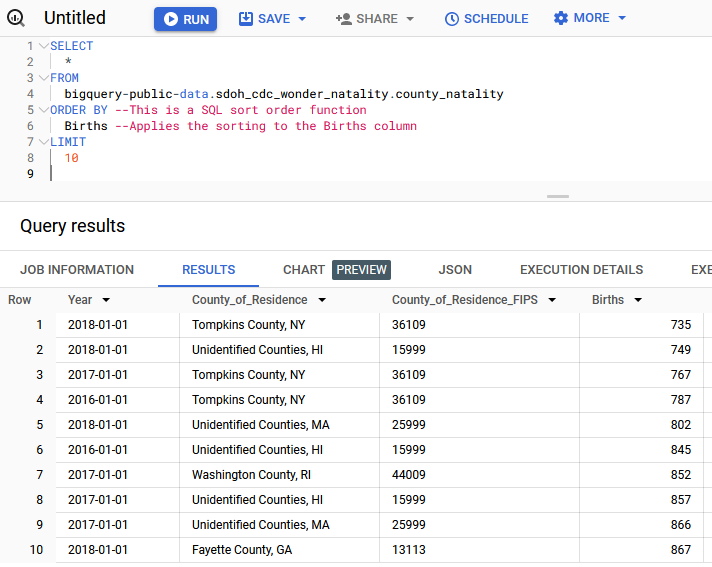
  bigquery-public-data.sdoh\_cdc\_wonder\_natality.county\_natality

ORDER BY --This is a SQL sort order function

  Births --Applies the sorting to the Births column

LIMIT

  10



Examine the Births column. Notice that it’s sorted from smallest to largest. When the **ORDER BY** function is applied to sort a given column, SQL will default to sorting in ascending order, which orders items from smallest to largest.

If you want the largest number to appear first, then you’d want to specify the sort order to be descending by adding a command to the **ORDER BY** clause. You can make your code easier to read by using a command to specify either sort order. Here are the corresponding commands:

* **ASC** = Ascending
* **DESC** = Descending

Next, you’ll use the same query, but this time you’ll explicitly state the order of your **ORDER BY** function using **ASC**. Enter and run the following SQL query:

1

2

3

4

5

6

7

8

SELECT

  \*

FROM

  bigquery-public-data.sdoh\_cdc\_wonder\_natality.county\_natality

ORDER BY

  Births ASC --Place the ASC or DESC specifier directly after the column name separated by a space (no other punctuation)

LIMIT

  10

Notice that the results did not change. Tompkins County, NY, had just 735 births in 2018—the lowest birth count of any county in the US between 2016-2018.

Now, change the order from ascending (**ASC**) to descending (**DESC**) to find the most births. Enter and run this query:

1

2

3

4

5

6

7

8

SELECT

  \*

FROM

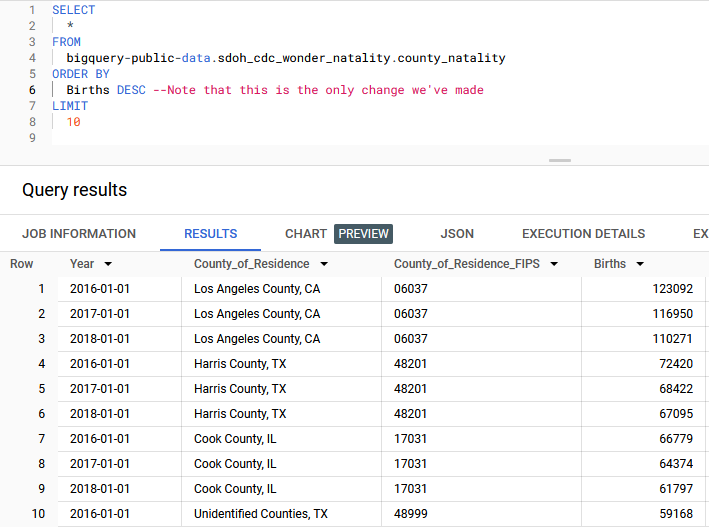
  bigquery-public-data.sdoh\_cdc\_wonder\_natality.county\_natality

ORDER BY

  Births DESC --Note that this is the only change you've made

LIMIT

  10



The query returns the 10 rows with the largest values in the Births column. Los Angeles County takes up the top three spots.

**Use sorted data to answer questions**

Now that you've become familiar with the basics of sorting functions, use them to answer questions about your data. This exercise will require you to apply both your previous learnings (especially filtering with the WHERE clause) and your new understanding of sorting.

In your work as a public health researcher, you’re exploring whether the birth rate trends in several counties in upstate New York have been increasing or decreasing—and whether they follow the same pattern.

To answer this, you’ll need the following information:

* Results from Erie, Niagara, and Chautauqua counties in New York state
* Results ordered by county of residence and year to find the trend

The following query will filter the results by county and sort the results year and county. This will allow you to determine if the number of births is increasing or decreasing in each county.

Enter the following query into the Query Editor, then select **RUN**.

1

2

3

4

5

6

7

8

9

10

11

SELECT

  \*

FROM

bigquery-public-data.sdoh\_cdc\_wonder\_natality.county\_natality

WHERE

  County\_of\_Residence = 'Erie County, NY'

  OR County\_of\_Residence = 'Niagara County, NY'

  OR County\_of\_Residence = 'Chautauqua County, NY'

ORDER BY

  County\_of\_Residence,

  Year

You’ve now successfully used both **ORDER BY** (sort) and **WHERE** (filter) clauses in the same query. Based on the results of this query, are births in these three counties following the same trend?

**Scenario**



You’re a data analyst at a news station in New York City. You’ve been tasked with answering questions about the weather for meteorologists. You’ll work with a public dataset that contains global summaries of the day (GSOD) from the National Oceanic and Atmospheric Administration (NOAA). The GSOD dataset includes information about daily weather elements, such as mean temperature and wind speed, from more than 9,000 weather stations across the globe. This dataset is updated daily, so in addition to being large, this dataset is constantly changing. So, you’ll save a subset of the data about the New York region in a new table to make your analysis easier.

**Analyze weather data**

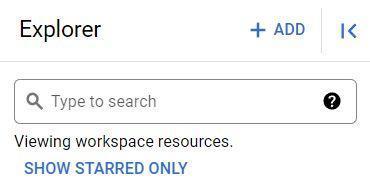
empty alt text

**Add the NOAA weather data to BigQuery**

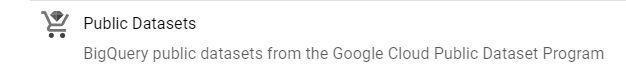
Log in to your BigQuery account (or [BigQuery sandbox](https://cloud.google.com/bigquery/docs/sandbox)) and initiate a new project or locate an existing project that you already created.

First, add the NOAA weather data from BigQuery’s public datasets.

1. Select the **+ ADD** button in the **Explorer** pane.



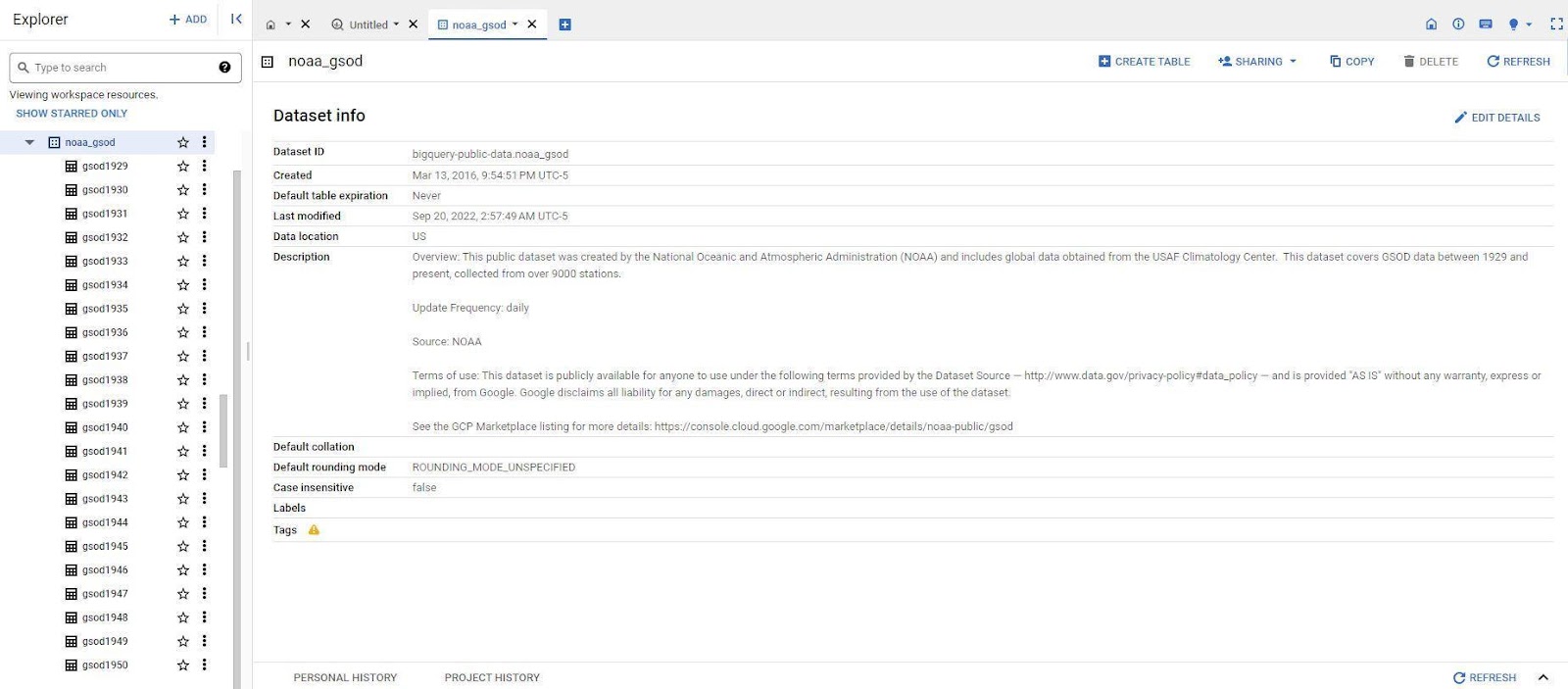
2. This opens a new **Add menu** where you can search public datasets available through Google Cloud. Scroll down the list and select **Public Datasets**.



3. This opens the **Marketplace** menu. In the search bar, enter the acronym **gsod** and press enter.

4. Select the search result titled **GSOD**.

5. Select **VIEW DATASET** to return to the main workspace with the NOAA dataset tables in the **Explorer** pane. The **Details** pane now contains details about this dataset.



**Query the data**

The meteorologists you’re working with have asked you to obtain the temperature, wind speed, and precipitation for stations La Guardia and JFK, for every day in 2020. They’ve also requested the data be presented to them in descending order by date and ascending order by Station ID. To return this information:

1. Select the **QUERY** button in the row of tab functions

2. Select the In **split tab** option in the drop-down menu.

3. Enter the following query into the Query Editor:

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

SELECT

  stn,

  date,

  IF(

     temp=9999.9,

     NULL,

     temp) AS temperature,

  IF(

     wdsp="999.9",

     NULL,

     CAST(wdsp AS Float64)) AS wind\_speed,

  IF(

     prcp=99.99,

     0,

     prcp) AS precipitation

FROM

  `bigquery-public-data.noaa\_gsod.gsod2020`

WHERE

  stn="725030" -- La Guardia

  OR stn="744860" -- JFK

ORDER BY

  date DESC,

  stn ASC

**Note:** This query uses the **IF** function to replace values 9999, 999.9, and 99.99 with **NULL**. The dataset description explains that these are the default values when the measurement is missing.

4. Select **RUN**.

Now you’ve run the query and gotten results. But these results aren’t saved anywhere, so each time you want to examine them, you would have to run this query again.

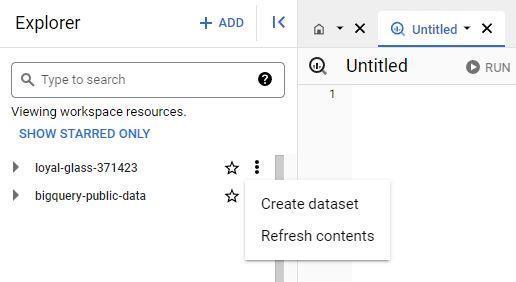
**Save a new table**

In addition to the data the meteorologists requested, they also asked you some questions while preparing for the nightly news: They want to know the average temperature in June 2020 and the average wind speed in December 2020.

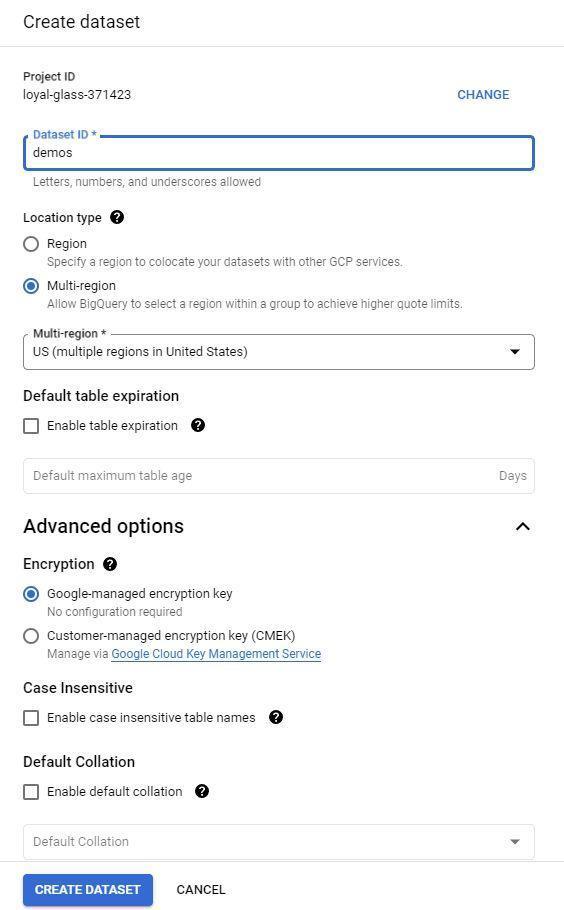
Instead of rewriting similar, but slightly different, queries over and over again, there is an easier approach: Save the results from the original query as a table for future queries.

To make this subset of data easier to query from, you’ll save this query as a table. First, though, you’ll create a new dataset to store the table.

1. From the **Explorer** pane, **select the three vertical dots** next to your project and select **Create dataset**. Note that unless you have specified your own project name, a unique name is assigned to your project by BigQuery, typically in the format of two words and a number, separated by hyphens (e.g., loyal-glass-371423 in the image below). You are not allowed to create a new dataset in the **bigquery-public-data** project.

A screenshot of the results of selecting the three vertical dots next to the project. The menu options are Create dataset and Refresh contents.

2. Enter **demos** into the Data ID box and set the **Location type** to **Multi-region,** then select **US (multiple regions in United States)**. Leave the rest of the **Advanced Options** as the default. Once you have done this, select **CREATE DATASET**.



3. Open the new dataset.

4. Click on the **+ button** in the **Details** pane.

5. If needed, re-enter the query you ran in the previous section:

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

SELECT

  stn,

  date,

    IF(

       temp=9999.9,

       NULL,

       temp) AS temperature,

    IF(

       wdsp="999.9",

       NULL,

       CAST(wdsp AS Float64)) AS wind\_speed,

    IF(

       prcp=99.99,

       0,

       prcp) AS precipitation

FROM

  `bigquery-public-data.noaa\_gsod.gsod2020`

WHERE

  stn="725030" -- La Guardia

  OR stn="744860" -- JFK

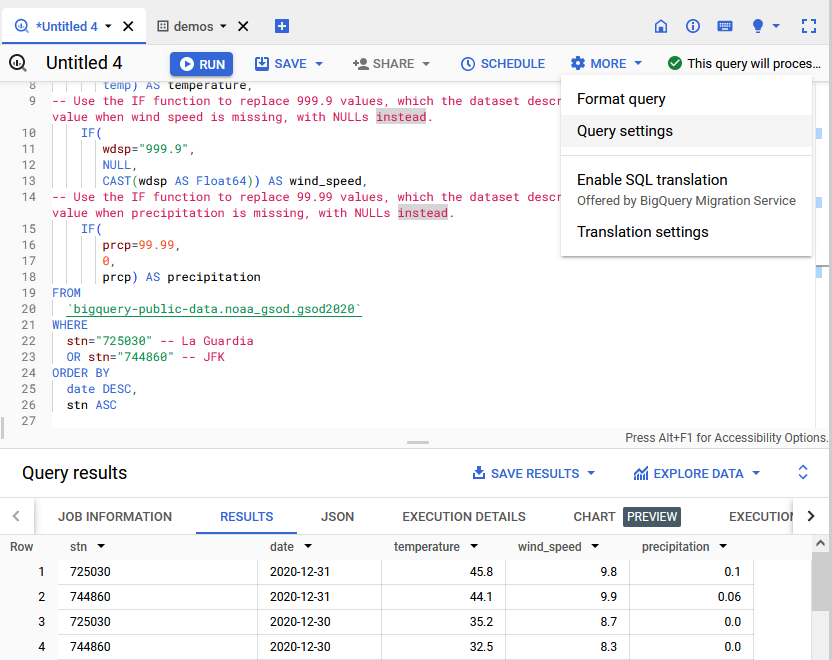
ORDER BY

  date DESC,

  stn ASC

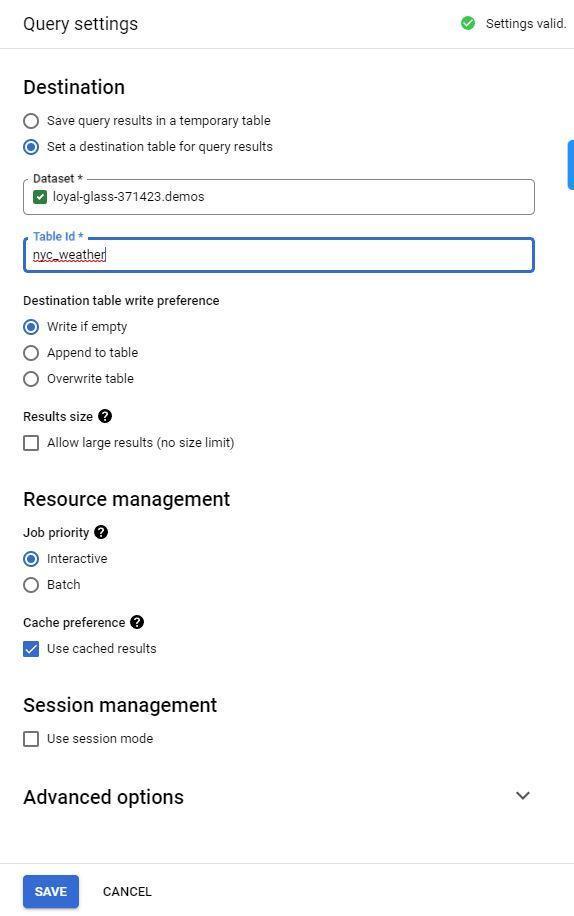
**Note:** Use the **IF** function to replace values 9999, 999.9, and 99.99 with **NULL**. The dataset description explains that these are the default values when the measurement is missing.

6. **Before** you run the query, select the **MORE** menu from the Query Editor and open the **Query Settings** menu.

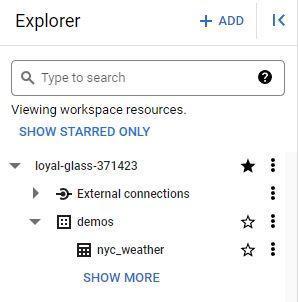


7. From the **Query Settings** menu, select the button next to **Set a destination table for query results**.

8. Set the dataset option to **demos** and name the table **nyc\_weather**.



9. Run the query to save the results as a new table in the **demos** dataset.



Before:



After:

A black text on a white background

Description automatically generated

10. Return to the **Query settings menu** by selecting the **MORE** dropdown menu.

11. Reset the settings to **Save query results in a temporary table**. This will prevent you from accidentally adding every query as a table to your new dataset.

**Query the new table**

Now that you have the subset of this data saved in a new table, you can use the following query to find the average temperature in June 2020. First, replace **your\_project\_name** with your project name in BigQuery.

1

2

3

4

5

6

SELECT

    AVG(temperature)

FROM

    `your\_project\_name.demos.nyc\_weather`

WHERE

    date BETWEEN '2020-06-01' AND '2020-06-30'

**Note:** Format the beginning syntax to your project name before running this query. View the full Table ID by selecting on the **Details** tab of your new **nyc\_weather** data table.

You can also use this syntax to find the average wind speed or any other information from this subset of data. Try writing a few more queries to answer the meteorologists’ questions.

The ability to save your results into a new table is helpful when you know you're only interested in a subset of a larger complex dataset that you plan on querying multiple times, such as the weather data for just La Guardia and JFK. This also helps minimize errors during your analysis.

### 1.

Question 1

A data analyst at a gift retailer sorts a list of handmade candles by price from least expensive to most expensive. Which statement do they use?

1 point

**ORDER BY candle\_price**

**WHERE candle\_price ASC**

**ORDER BY candle\_price DESC**

**WHERE candle\_price**

### 2.

Question 2

What will this query return?

1

2

3

4

5

SELECT \*

FROM widgets

WHERE

ORDER BY

    manufacture\_date DESC

1 point

All of the columns in the manufacture\_date table

All of the rows in the widgets table, with the least-recently manufactured widgets listed first

All of the rows in the widgets table, with the most-recently manufactured widgets listed first

All of the rows in the widgets table

### 3.

Question 3

You are working with a database table that contains employee data. Which **ORDER BY** clause will sort employees by the earliest hire dates at the top of the list?

1 point

**ORDER BY hire\_date DESC**

**ORDER BY hire\_date REV**

**ORDER BY hire\_date ASC**

**ORDER BY hire\_date MIN**

### 4.

Question 4

What SQL operator enables a data professional to filter for two conditions at once when using a **WHERE** statement?

1 point

Ampersand **&**

Plus sign **+**

**IN**

**AND**

**ORDER BY:** A SQL clause that sorts results returned in a query

### 1.

Question 1

A data professional at a finance company sorts spreadsheet data. They sort all data by ranking in the Financial Performance column, keeping together all data across rows. What spreadsheet tool are they using?

1 point

Sort Rows

Sort Column

Sort Sheet

Sort Together

### 2.

Question 2

Fill in the blank: To filter for all students in the Sophomore table who live in Fairfield County, a data professional uses the \_\_\_\_\_ clause in SQL.

1 point

**EXCEPT**

**FILTER**

**WHERE**

**LIMIT**

### 3.

Question 3

A data analyst determines whether there are any patterns in a dataset. What phase of analysis is the analyst in?

1 point

Transform data

Organize data

Format and adjust data

Get input from others

### 4.

Question 4

Which of the following statements accurately describe sorting and filtering? Select all that apply.

1 point

Filtering enables data professionals to view the data that is most important.

Sorting involves arranging data into a meaningful order.

Filtering can be performed in spreadsheets, but not SQL databases.

Sorting can be performed in both spreadsheets and SQL databases.

### 5.

Question 5

Fill in the blank: During an analysis project, \_\_\_\_\_ might involve converting dates to a consistent format in order to prepare the dataset for analysis.

1 point

organizing data

transforming data

getting input from others

formatting and adjusting data

### 6.

Question 6

Which query will return a list of all pickup trucks that have fewer than 10,000 miles, in order from the oldest to the most recent year produced?

1 point

1

2

3

4

SELECT \*

FROM 'car\_dealership'

WHERE Mileage = 'Pickup', Revenue < 10000

ORDER BY year\_produced ASC

1

2

3

4

5

SELECT \*

FROM 'car\_dealership'

WHERE Type = 'Pickup'

AND Mileage < 10000

ORDER BY year\_produced ASC

1

2

3

4

5

SELECT \*

FROM 'car\_dealership'

WHERE Type = 'Pickup'

WHERE Mileage > 10000

ORDER BY year\_produced DESC

1

2

3

4

5

SELECT \*

FROM 'car\_dealership'

WHERE Type = 'Pickup'

AND Mileage > 10000

ORDER BY year\_produced DESC

### 7.

Question 7

A data professional at a manufacturing company is tasked with identifying which machines are most likely to need repairs. In the analyze phase of the data analysis process, what activities might this involve? Select all that apply.

1 point

Format the data to filter for machines that need the most maintenance

Organize a dataset by machine type and performance levels

Get input from colleagues on the data team

Prepare a report for the stakeholders

### 8.

Question 8

Which function sorts a spreadsheet range between cells C1 and D70 in ascending order by the first column, Column C?

1 point

**=SORT(C1:D70, 1, FALSE)**

**=SORT(C1:D70, A, FALSE)**

**=SORT(C1:D70, 1, TRUE)**

**=SORT(C1:D70, A, TRUE)**

Hey, it's great to have you back! You've learned so much already, and

now you're ready to start analyzing data.

Coming up, we'll cover some final things you'll need to do for

your analysis to make sure your data is formatted and adjusted correctly.

Play video starting at ::16 and follow transcript0:16

We'll start converting and formatting your data; using data validation in

spreadsheets; and conditional formatting.

You'll also learn how to combine multiple pieces of data.

And finally, we'll talk about how to get support during your analysis and

find resources whenever you're stuck.

Play video starting at ::33 and follow transcript0:33

These skills will help make sure that your data analysis process is as smooth as

possible. And even when it isn't,

you'll know how to tackle any problems that might come up.

Play video starting at ::44 and follow transcript0:44

A big piece of being an analyst is troubleshooting and problem-solving.

You're as good of an analyst as your ability to ask the right questions,

which is why we'll spend some time learning about problem-

solving strategies you can use during analysis.

So whenever you're ready to start learning about data formatting and

solving problems, head to the next video, and we'll get started.

**Step-by-Step: From one type to another**

This reading provides you with the steps the instructor performs in the following video, [From one type to another](https://www.coursera.org/learn/analyze-data/lecture/FOAwr/from-one-type-to-another). Watch as the instructor demonstrates how to format numbers and convert units of measurement in your spreadsheets.

Keep this step-by-step guide open as you watch the video. It can serve as a helpful reference tool 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 access the spreadsheets the instructor uses in this video, click the links to the dataset to create a copy. If you don’t have a Google account, you may download the data directly from the attachments below.

Link to movie data starter project: [Movie data starter project](https://docs.google.com/spreadsheets/d/1FLaUmMn62YlHYihV6pK1DJqWcFYCnuoqoxFWmm_o5b0/template/preview).

Link to weather table - data for convert: [Weather Table - Data for CONVERT](https://docs.google.com/spreadsheets/d/15VeWQLQ5lUKvywYJL-0cGqmehvE8OH8W9cOlJ2P0J_I/template/preview).

OR

[Movie Data Starter Project](https://d3c33hcgiwev3.cloudfront.net/7rkSg1b6TCeiM37bJ_iPQA_cc9c5dd03dbe4e09bc83a3c244e4d6e1_Movie-Data-Starter-Project.xlsx?Expires=1706832000&Signature=hNrM8-Y5cZFU8Jx05x-nw5tSR3tjBuJ8dYdPfPXb9rw5YQPfgsVZZbNE6ltlquQx~ypQzvLZqLB~1S7C6ZmA81d93p2DNaGUUOImbB2DXRYUOeFU3oP2gFP0SB945mOctq92b44BWXpsmCYgM5DDJdCfc~78Q6uXOQxrXC~NTEg_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[XLSX File](https://d3c33hcgiwev3.cloudfront.net/7rkSg1b6TCeiM37bJ_iPQA_cc9c5dd03dbe4e09bc83a3c244e4d6e1_Movie-Data-Starter-Project.xlsx?Expires=1706832000&Signature=hNrM8-Y5cZFU8Jx05x-nw5tSR3tjBuJ8dYdPfPXb9rw5YQPfgsVZZbNE6ltlquQx~ypQzvLZqLB~1S7C6ZmA81d93p2DNaGUUOImbB2DXRYUOeFU3oP2gFP0SB945mOctq92b44BWXpsmCYgM5DDJdCfc~78Q6uXOQxrXC~NTEg_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[Weather Table - Data for CONVERT](https://d3c33hcgiwev3.cloudfront.net/dtqtPkiTSV27NcIQALH5gw_e797db6bf7374b11b5e41841e34d71e1_Weather-Table---Data-for-CONVERT.xlsx?Expires=1706832000&Signature=LjmsMWW3~lu95NQxR~OXHI3ZsZqi9CofaUf0b7vCIM2P5tIcgr1ghCZd5e5shgfP-Jo9ud7~~bgKqSXmZO7CP8wJpoN2W2Z-DJVjfXLm4Nqh8hnO84PWnQBUuoeb7Cny1iL-PE9Jf-twaL92257cEPgC-qvqCTIHSb7gaRxEXPA_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[XLSX File](https://d3c33hcgiwev3.cloudfront.net/dtqtPkiTSV27NcIQALH5gw_e797db6bf7374b11b5e41841e34d71e1_Weather-Table---Data-for-CONVERT.xlsx?Expires=1706832000&Signature=LjmsMWW3~lu95NQxR~OXHI3ZsZqi9CofaUf0b7vCIM2P5tIcgr1ghCZd5e5shgfP-Jo9ud7~~bgKqSXmZO7CP8wJpoN2W2Z-DJVjfXLm4Nqh8hnO84PWnQBUuoeb7Cny1iL-PE9Jf-twaL92257cEPgC-qvqCTIHSb7gaRxEXPA_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

empty alt text

**Example 1: Check and change data type**

Check your data for inconsistent units of measurement to prevent problems during data analysis.

1. Open the [**Movie Data Starter Project**](https://docs.google.com/spreadsheets/d/1FLaUmMn62YlHYihV6pK1DJqWcFYCnuoqoxFWmm_o5b0/template/preview) spreadsheet using the link in the video.
2. Select **Column M** [Budget ($)] and **Column N** [Box Office Revenue ($)].
3. From the menu, select **$**, the currency shortcut key.
4. Notice that the currency in **Columns M** and **N** are now formatted correctly.

**Example 2: Convert temperatures from Fahrenheit to Celsius**

Use the **CONVERT** function to change units of measurement.

1. Open the [**Weather Table - Data for CONVERT**](https://docs.google.com/spreadsheets/d/15VeWQLQ5lUKvywYJL-0cGqmehvE8OH8W9cOlJ2P0J_I/template/preview) spreadsheet using the link in the video.
2. Select cell **F2** and begin typing the Convert function formula as **=CONVERT**.
3. Indicate the cell you want to convert. After **=CONVERT**, enter **(B2,**.
4. Indicate the conversion you’d like to make: from Fahrenheit to Celsius. Enter **"F", "C")**.
5. The formula in its entirety should look like this: **=CONVERT (B2, "F", “C”)**.
6. Cell **F2** now contains the temperature from cell **B2** in Celsius.
7. Calculate temperature in Celsius for the rest of the column. Hover over cell **F2** and select the fill handle, a small circle on a corner of the cell. Drag the fill handle to cell **F193** to convert the other cells in the column to Celsius.

**Note**: Would you like more practice? Try converting the wind speed in Column **D** from miles per hour (mph) to meters per second (m/s) using **CONVERT**.  In cell **H2**, enter: **=CONVERT(D2, "mph", "m/s")**.

You can check if your conversion is correct by entering 8.5248 in a metric conversion tool, [metric-conversions.org/speed/miles-per-hour-to-meters-per-second.htm](https://www.metric-conversions.org/speed/miles-per-hour-to-meters-per-second.htm).

**Example 3: Lock data in a table**

Using functions to convert data can lead to problems, which data professionals must be prepared to fix. For example, if a reference value changes, the calculated value also changes. Locking data in a table by changing it from a function to a value ensures a cell stays consistent even if the data around it changes.

1. Select cell **F2**. In the formula bar, notice that the contents of this cell are the function you entered in the previous example.
2. Right-click cell **F** and select **Copy** from the drop-down menu.
3. Right-click cell **G** and select **Paste special** from the drop-down menu. Then, select **Paste values only**. This option pastes only the values from the original selection, removing any formatting, functions, or other information.
4. Select cell **G2**.
5. In the formula bar, notice that the contents of this cell is a value. This means that the value won’t change when other cells change.

Hey there!

So far, we've learned about typecasting data with SQL as a way of converting

data from one type to another in databases.

Now I want to check out another way to format data types within spreadsheets.

In this video, we'll talk more about why making

sure your data is formatted properly is so important

and how to format numbers and convert units of measurement in your spreadsheets.

Let's get started.

Play video starting at ::29 and follow transcript0:29

Sometimes, you need to convert data when you're working with spreadsheets.

That might mean changing numbers into dates, strings, percentages, or

even currency.

Play video starting at ::39 and follow transcript0:39

It's important to double check that all of your data is in the right format for

your analysis.

Sometimes even after cleaning and processing data,

it still might not be in the right format you need.

Play video starting at ::51 and follow transcript0:51

Let's think back to the table with movie data from before.

There were a lot of different data types that included numbers, such as dates,

budgets, and text strings, like actors' names.

Play video starting at :1:3 and follow transcript1:03

These are distinct values, but

the spreadsheet doesn't always automatically know that.

Here's an example.

Let's say you wanted to sort the movies in this spreadsheet by most recent.

If the spreadsheet cast them as strings instead of dates,

it might sort them alphabetically.

Until you change the data type, you won't be able to sort them the way you want.

It's also possible that your datasets contain inconsistent units of

measurement that you'll need to convert.

Like say, a table that includes both US dollars and English pounds.

That's why it's important to check those data types again,

so you don't run into any problems during the actual analysis.

Think about the incorrectly cast dates in our movie table.

If your boss needed a list of the 20 most recent movies, but

your spreadsheet was organized alphabetically instead of by

the most recent, you wouldn't be giving her the list of movies she needed.

Incorrectly formatted data can lead to time-consuming mistakes in your analysis,

and might end up affecting your stakeholders' decision-making.

Play video starting at :2:10 and follow transcript2:10

But taking the time early on to convert and format your data

can help you avoid that.

Play video starting at :2:16 and follow transcript2:16

And now that you know why you'll need to convert data types while working in

spreadsheets, let's find out how.

Play video starting at :2:23 and follow transcript2:23

First, let me show you a really useful menu for specifying data types in

spreadsheets.

Here's the movie data table we use before, but

now the money columns aren't typed as currency.

On the toolbar at the top of the sheet, you'll find a menu that can help you

convert these numbers into specific data types.

It gives you a lot of choices just from the drop-down menu, such as number,

currency, date, percentage....

And if you click to open the full menu, there's even more options,

including one for a custom number format.

We know that we want these columns to be in currency format, so let's do that.

All I have to do is select this column and then hit the currency shortcut.

Play video starting at :3:8 and follow transcript3:08

And now it's all typed correctly. But it doesn't stop there.

You can go even further and convert the unit of measurement you're using.

For this example, let's check out a different table.

Imagine that you're working with a weather channel to gather data about daily

temperatures.

You have a table with some data about daily observations on the temperature,

wind speed, and precipitation in this area.

Right now, the temperatures are in Fahrenheit, but for

your analysis you need them to be in Celsius.

No problem.

All you need to do is use the CONVERT function to change the unit of

measurement.

We'll use this empty column here.

Here's the first temperature in the table.

We'll input the CONVERT function in our new column to change it to Celsius.

Then we need to put what cell we want converted.

And finally, we're going to convert.

Play video starting at :4:2 and follow transcript4:02

And presto!

Now this cell has the right unit of measurement for your analysis.

You can simply apply it to the rest of this column.

Now this temperature data is all in Celsius, and

your unit of measurement is consistent across the table.

And here's another tip.

When adding data to tables using a formula, go back and paste the data in

as values afterwards. That way they're locked in.

Otherwise the cell stays as a formula and

could get confusing when you start working with the data.

So let's do that now.

We'll copy the values and then right click in a new column.

There's an option for "Paste special." And there's an option to "Paste values only."

And now we have the static values in this column.

Making sure your data is in the right format before you start analysis is so

important.

Do this, and

your analysis will return the kinds of answers you're really searching for.

And now you know some ways to typecast numbers and

convert units of measurement in spreadsheets.

You can feel confident your data is formatted the right way.

Play video starting at :5:13 and follow transcript5:13

Next up, we'll talk more about adjusting your data for analysis and

data validation.

See you soon.

# Convert data in spreadsheets

In this reading, you will learn about converting data from one format to another. One of the ways to help ensure that you have an accurate analysis of your data is by putting all of it in the correct format. This is true even if you have already cleaned and processed your data. As a part of getting your data ready for analysis, you will need to convert and format your data early on in the process.



As a data analyst, there are lots of scenarios when you might need to convert data in a spreadsheet:

### **String to date**

* [**How to convert text to date in Excel**](https://www.ablebits.com/office-addins-blog/2015/03/26/excel-convert-text-date/#:~:text=Excel%20DATEVALUE%20function%20%2D%20change%20text,Excel%20recognizes%20as%20a%20date.&text=So%2C%20the%20formula%20to%20convert,stored%20as%20a%20text%20string.): Transforming a series of numbers into dates is a common scenario you will encounter. This resource will help you learn how to use Excel functions to convert text and numbers to dates, and how to turn text strings into dates without a formula.
* [**Google Sheets: Change date format:**](https://www.ablebits.com/office-addins-blog/2019/08/13/google-sheets-change-date-format/) If you are working with Google Sheets, this resource will demonstrate how to convert your text strings to dates and how to apply the different date formats available in Google Sheets.

### **String to numbers**

* [**How to convert text to number in Excel:**](https://www.ablebits.com/office-addins-blog/2018/07/18/excel-convert-text-to-number/) Even though you will have values in your spreadsheet that resemble numbers, they may not actually be numbers. This conversion is important because it will allow your numbers to add up and be used in formulas without errors in Excel.
* [**How to convert text to numbers in Google Sheets:**](https://productivityspot.com/convert-text-to-numbers-google-sheets/) This resource is useful if you are working in Google Sheets; it will demonstrate how to convert text strings to numbers in Google Sheets. It also includes multiple formulas you can apply to your own sheets, so you can find the method that works best for you.

### **Combining columns**

* [**Convert text from two or more cells:**](https://support.microsoft.com/en-us/office/combine-text-from-two-or-more-cells-into-one-cell-81ba0946-ce78-42ed-b3c3-21340eb164a6) Sometimes you may need to merge text from two or more cells. This Microsoft Support page guides you through two distinct ways you can accomplish this task without losing or altering your data. It also includes a step-by-step video tutorial to help guide you through the process.
* [**How to split or combine cells in Google Sheets:**](https://www.techrepublic.com/article/how-to-split-or-combine-text-cells-with-google-sheets/) This guide will demonstrate how to to split or combine cells using Google Sheets specifically. If you are using Google Sheets, this is a useful resource to reference if you need to combine cells. It includes an example using real data.

### **Number to percentage**

* [**Format numbers as percentages:**](https://support.microsoft.com/en-us/office/format-numbers-as-percentages-de49167b-d603-4450-bcaa-31fba6c7b6b4) Formatting numbers as percentages is a useful skill to have on any project. This Microsoft Support page will provide several techniques and tips for how to display your numbers as percentages.
* [**TO\_PERCENT:**](https://support.google.com/docs/answer/3094284?hl=en) This Google Sheets support page demonstrates how to use the **TO\_PERCENT** formula to convert numbers to percentages. It also includes links to other formulas that can help you convert strings.

**Pro tip:** Keep in mind that you may have lots of columns of data that require different formats. Consistency is key, and best practice is to make sure an entire column has the same format.

## Additional resources

If you find yourself needing to convert other types of data, you can find resources on [**Microsoft Support**](https://support.microsoft.com/) for Excel or [**Google Docs Editor Help**](https://support.google.com/docs/?hl=en#topic=1382883) for Google Sheets.

Converting data is quick and easy, and the same functions can be used again and again. You can also keep these links bookmarked for future use, so you will always have them ready in case any of these issues arise. Now that you know how to convert data, you are on your way to becoming a successful data analyst.

Welcome back! While we're learning about formatting data,

I want to talk to you about

another spreadsheet feature: data validation.

In this video, I'll teach you a little bit

about data validation and show you how to use it.

For now, when I say data validation,

I'm talking about the function,

which is different from the data validation process.

We'll get into that later on.

But first, let's talk about what

data validation does in spreadsheets.

Basically, it allows you to control what

can and can't be entered in your worksheet.

Usually, data validation is used to add

drop-down lists to cells with

predetermined options for users to choose from.

If you have a spreadsheet with a lot of collaborators,

this can make it easier for them to

interact with your table.

You can think of it like

a multiple choice question on a quiz.

Since you control what's

being entered into the worksheet,

it cuts down on how much data

cleaning you have to do later on.

Let's figure out how we might do that.

For this example, we'll work on a project with

a lot of milestones and deadlines to keep track of.

Let's say our team has a spreadsheet

that tracks everyone's progress.

But instead of making everyone write in

where they are in their task individually,

we can provide a drop-down menu

with multiple options, like

"Not Yet Started," "In Progress," and "Ready."

So we'll select the column that we want

to add the drop-down menus to,

in this case, the "Status" column.

Then we'll go to the Data pull-down menu

here at the top and click "Data validation."

This brings up a pop-up menu

with options for data validation.

In this case, we know that we want to add

a list of items for other users to choose from.

So we'll select the "list of items" option

from the possible criteria and

type in the selections we want to create.

Then hit Save, and now all of those cells have drop-down

menus that we can use to

easily mark progress for each task.

But there's other things that you can do with

data validation and spreadsheets, too,

like creating custom check boxes.

To do this, let's select

the cells under the "Review" column to make

a checkbox that will let us

know if tasks have been approved or not.

We'll go back to the data validation menu.

But instead of choosing "List from a

range," we'll choose "Checkbox."

There's an option to use custom cell values.

Let's choose that and put in

"Approved" and "Not approved."

Now these tasks can be checked

off by whoever's reviewing them,

like a project manager, for example.

Another way we can use data validation is to

protect structured data and formulas.

The more people that are working

together in a spreadsheet,

the more likely someone can accidentally break a formula.

But good news: the data validation menu

has an option to reject invalid inputs,

which helps make sure

our custom tools will continue to run correctly,

even if someone puts the wrong data in by mistake.

All right, now you know three uses

for data validation in your spreadsheets:

adding drop-down lists, creating custom checkboxes,

and protecting structured data and formulas.

Data validation can help your team track progress,

protect your tables from

breaking when working in big teams,

and help you customize tables to your needs.

Coming up, we'll learn more

about conditional formatting and some ways you can

use conditional formatting and

data validation together. See you soon!

Hi again.

So earlier we talked about conditional formatting as a spreadsheet tool that

changes how cells appear when values meet specific conditions.

This lets you add visual cues to your spreadsheets that

make it easier to understand your table at a glance, and

it makes the information in the worksheet clearer to your stakeholders.

In this video, we'll take that even further by combining

conditional formatting and data validation to create custom tools for our spreadsheets.

So far, we've used conditional formatting to highlight empty cells that still needed

data so that we could quickly pinpoint what information our table was missing and

add it in.

Now, let's build on that by using it to make our scheduling table easier to read

at a glance.

Play video starting at ::48 and follow transcript0:48

Here's a table we worked with when we covered data validation.

It's tracking the status of different tasks on our project for

our team to check on.

But now there's even more tasks than the last time we looked at it.

This table has useful information, but it takes a second to understand.

Right now we don't have a visual on how many tasks are in progress or

how many upcoming deadlines there are.

But if we color-coded those elements of the table,

we could quickly see key pieces of data really easily.

Let's start with the Status column, column C.

In the last example, we created these drop- down menus with the data validation tool.

Now we can use conditional formatting to add some color.

Let's go to the conditional formatting option under the Format menu.

Play video starting at :1:37 and follow transcript1:37

This brings up a sidebar where we can select our range rule in formatting style.

We need to decide which rows to apply our formatting to when the condition we

set is met.

We can click this button in the range options to select all of the rows we're

applying the formatting to instead of typing it in.

Now that we have those cells selected,

we can choose the rule that we want to apply to these cells.

We already have drop-down menus with specific text.

So we can choose "Format Cells if... Text is exactly" from the rules.

For our first rule, let's write "Not Yet Started" as the text condition.

Then we'll choose a color to apply to those cells that have "Not Yet

Started" in them.

Let's use red.

Now all cells that have "Not Yet

Started" selected from the drop- down menu will be red.

Play video starting at :2:26 and follow transcript2:26

Let's hit the "Add another rule" button to add conditional formatting to other status

options.

Let's add the condition "In Progress" next.

We can make that one yellow. And then we'll add one last rule for "Ready."

Let's choose green.

And there. Now we have an easy-to- understand visual cue that tells us how

many tasks are in progress, and how many are completed.

We can also combine data validation and

conditional formatting to track upcoming deadlines.

We have a column of dates called "Review By This Date."

First, let's use the data validation functionality to make sure users

only enter valid dates.

We'll go back to the Data dropdown at the top, pull up Data validation, and

select Date as our criteria.

Play video starting at :3:15 and follow transcript3:15

Then we can go to the Format menu at the top. Go down to conditional formatting and

open the sidebar again.

We'll click the "Select range" icon and select the "Review By This Date" column.

Now under Format rules, we can select "Date is after,"

which will give us another option.

Let's choose "today."

Play video starting at :3:38 and follow transcript3:38

And finally, let's choose the color for these cells.

So if the date listed in these rows is after today, it'll be filled in orange.

You can also choose a specific locked date if needed.

But for now, let's go with today.

Now all of the upcoming review dates have an easy-to-see color code,

so anyone using this table can quickly reference these deadlines.

You'll find that some spreadsheet programs, like Excel, have

built-in color codes that you can use, too. And there you go.

Now you know how to use data validation and conditional formatting to create

custom tools and visual cues that make your information easy to understand.

There's a lot of different ways to use these tools,

so feel free to experiment with them in your own spreadsheets. Coming up,

we'll keep learning about new tools for spreadsheets and SQL. Bye for now.

# Transform data with SQL

Data analysts usually need to convert data from one format to another to complete an analysis. But what if you are using SQL rather than a spreadsheet? Just like spreadsheets, SQL uses standard rules to convert one type of data to another. If you are wondering why data transformation is an important skill to have as a data analyst, think of it like being a driver who is able to change a flat tire. Being able to convert data to the right format speeds you along in your analysis. You don’t have to wait for someone else to convert the data for you.



In this reading, you will go over the conversions that can be done using the **CAST** function. There are also more specialized functions like **COERCION** to work with big numbers, and **UNIX\_DATE** to work with dates. **UNIX\_DATE** returns the number of days that have passed since January 1, 1970 and is used to compare and work with dates across multiple time zones. You will likely use **CAST** most often.

## Common conversions

The following table summarizes some of the more common conversions made with the **CAST** function. Refer to [Conversion Rules in Standard SQL](https://cloud.google.com/bigquery/docs/reference/standard-sql/conversion_rules) for a full list of functions and associated rules.

| **Starting with** | **CAST function can convert to:** |
| --- | --- |
| Numeric (number) | - Integer - Numeric (number) - Big number - Floating integer - String |
| String | - Boolean - Integer - Numeric (number) - Big number - Floating integer - String - Bytes - Date - Date time - Time - Timestamp |
| Date | - String - Date - Date time - Timestamp |

## The CAST function (syntax and examples)

**CAST** is an American National Standards Institute (ANSI) function used in lots of programming languages, including BigQuery. This section provides the BigQuery syntax and examples of converting the data types in the first column of the previous table. The syntax for the **CAST** function is as follows:

1

CAST(expression AS typename)

Where **expression** is the data to be converted and **typename** is the data type to be returned.

### **Converting a number to a string**

The following **CAST** statement returns a string from a numeric identified by the variable **MyCount** in the table called **MyTable**.

1

SELECT CAST(MyCount AS STRING) FROM MyTable

In the above SQL statement, the following occurs:

* **SELECT** indicates that you will be selecting data from a table
* **CAST** indicates that you will be converting the data you select to a different data type
* **AS** comes before and identifies the data type which you are casting to
* **STRING** indicates that you are converting the data to a string
* **FROM** indicates which table you are selecting the data from

### **Converting a string to a number**

The following **CAST** statement returns an integer from a string identified by the variable **MyVarcharCol** in the table called **MyTable**. (An integer is any whole number.)

1

SELECT CAST(MyVarcharCol AS INT) FROM MyTable

In the above SQL statement, the following occurs:

* **SELECT** indicates that you will be selecting data from a table
* **CAST** indicates that you will be converting the data you select to a different data type
* **AS** comes before and identifies the data type which you are casting to
* **INT** indicates that you are converting the data to an integer
* **FROM** indicates which table you are selecting the data from

### **Converting a date to a string**

The following **CAST** statement returns a string from a date identified by the variable **MyDate** in the table called **MyTable**.

1

SELECT CAST(MyDate AS STRING) FROM MyTable

In the above SQL statement, the following occurs:

* **SELECT** indicates that you will be selecting data from a table
* **CAST** indicates that you will be converting the data you select to a different data type
* **AS** comes before and identifies the data type which you are casting to
* **STRING** indicates that you are converting the data to a string
* **FROM** indicates which table you are selecting the data from

### **Converting a date to a datetime**

Datetime values have the format of YYYY-MM-DD hh: mm: ss format, so date and time are retained together. The following **CAST** statement returns a datetime value from a date.

1

SELECT CAST (MyDate AS DATETIME) FROM MyTable

In the above SQL statement, the following occurs:

* **SELECT** indicates that you will be selecting data from a table
* **CAST** indicates that you will be converting the data you select to a different data type
* **AS** comes before and identifies the data type which you are casting to
* **DATETIME** indicates that you are converting the data to a datetime value
* **FROM** indicates which table you are selecting the data from

## The SAFE\_CAST function

Using the **CAST** function in a query that fails returns an error in BigQuery. To avoid errors in the event of a failed query, use the **SAFE\_CAST** function instead. The **SAFE\_CAST** function returns a value of Null instead of an error when a query fails.

The syntax for **SAFE\_CAST** is the same as for **CAST**. Simply substitute the function directly in your queries. The following **SAFE\_CAST** statement returns a string from a date.

1

SELECT SAFE\_CAST(MyDate AS STRING) FROM MyTable

## More information

Browse these resources for more information about data conversion using other SQL dialects (instead of BigQuery):

* [CAST and CONVERT](https://docs.microsoft.com/en-us/sql/t-sql/functions/cast-and-convert-transact-sql?view=sql-server-ver15): SQL Server reference documentation
* [MySQL CAST Functions and Operators](https://dev.mysql.com/doc/refman/8.0/en/cast-functions.html): MySQL reference documentation
* [How to: SQL Type Casting](https://www.blendo.co/blog/how-to-sql-type-casting/): Blog about type casting that has links to other SQL short guides

### 1.

Question 1

A spreadsheet cell contains the coldest temperature ever recorded in Austria: -37 degrees Celsius. Which function would convert that to Fahrenheit?

1 point

**=CONVERT(-37, F, C)**

**=CONVERT(-37, "C", "F")**

**=CONVERT(-37, "F", "C")**

**=CONVERT(-37, C, F)**

### 2.

Question 2

Fill in the blank: A data professional uses \_\_\_\_\_ in order to ensure spreadsheet values are static, rather than carrying over a preexisting formula or function.

1 point

formatting

paste values only

data validation

conditional formatting

### 3.

Question 3

Which data-validation menu option highlights data entry errors to ensure spreadsheet formulas continue to run correctly?

1 point

Forbid entry

Deny text

Reject invalid inputs

Remove validation

### 4.

Question 4

A data analyst selects Format Cells and the option Text Is Exactly Baseball. This changes the color of all the cells that contain the word “Baseball.” What spreadsheet tool is the analyst using?

1 point

Conditional formatting

Filtering

**CONVERT**

Data validation

# Import and combine data in spreadsheets and databases

In earlier lessons, you discovered how to use the **IMPORTRANGE** and **CONCATENATE** functions in spreadsheets. In this reading, you will have the opportunity to extend your knowledge about these concepts to SQL queries.

## Import data

As a data analyst, there are many occasions where you will need to import data from one file or location to another. Both spreadsheets and SQL include functionality that enables you to import data.

### **Import data in spreadsheets**

As you learned earlier, in spreadsheets you use the **IMPORTRANGE** function to import a range of cells from another spreadsheet into your current spreadsheet. The syntax is: **=IMPORTRANGE(spreadsheet\_url, range\_string)**.

In this formula, **spreadsheet\_url** is the URL of the spreadsheet from which you want to import data. The specific cells you want to import, such as A2:B6, are specified by **range\_string**. If the spreadsheet has multiple tabs, you also need to specify the name of the tab as part of the range.

An example of this is a company that needs to track who made retirement contributions so that it can make sure the company match is correctly distributed. The analysts would use **IMPORTRANGE** to pull all retirement contribution information into a spreadsheet that contains all of the employees year-end salaries and bonuses. This enables them to determine which employees made contributions and are eligible for matching funds.

### **Import data in SQL**

In contrast to spreadsheets, SQL does not include a function for importing data. Instead, a method you can use to import data from one table to another is to use the **INSERT INTO** command together with a **SELECT** statement. The syntax is:

1

2

3

4

INSERT INTO [destination\_table\_name]

SELECT [column names, separated by commas, or \* for all columns]

FROM [source\_table\_name]

WHERE [condition]

In this syntax, the SQL query inserts rows from a source table into a destination table based on the **WHERE** clause.

For example, imagine you work for a retail company that stores its sales and customer information in a SQL database. The marketing director asks you to provide them with a table containing the names and addresses of customers who have not made a purchase this year and who live in specific postal codes. One way you could gather this information is to use the **INSERT INTO** along with the **SELECT** and **WHERE** commands, as follows:

1

2

3

4

INSERT INTO customer\_promotion

SELECT \*

FROM customers

WHERE total\_sales = 0 AND postal\_code = '12345'

## Combine data

Another tool in your data analyst toolkit is your ability to join together two or more text strings that are stored in separate columns or fields. For example, you might want to combine a customer’s first and last name to create mailing labels for a marketing promotion. In both spreadsheets and SQL, joining together text strings is referred to as concatenation.

A machine with a conveyor belt

Description automatically generated

### **Combine data in spreadsheets**

In spreadsheets, you use the **CONCATENATE** function to join together two or more text strings, such as combining street addresses and primary contacts in a business’ vendor database.

The basic syntax is **=CONCATENATE(item 1, item 2)**. You can add multiple items by separating them with commas. Where appropriate, such as when you’re combining a customer’s first and last name, you should add a space between the items you’re combining by typing quotation marks space quotation marks [“ ”] between the items. Separate this information by a comma as well. This would change the formula to: **=CONCATENATE(item 1, " ", item 2)**.

### **Combine data in SQL**

In SQL, use the **CONCAT** function to join strings together to create new text strings. You might combine data simply to improve the readability of reports (such as combining a customer’s first and last name when generating a customer list). Or, you might combine data to generate a unique identifier for the rows in a table. Here is the basic syntax:

1

2

SELECT CONCAT(field1, " ", field2)

FROM [table\_name]

Notice that this syntax includes " " so that there is a space between the combined fields. With this syntax, SQL combines field1 and field2 with a space between them.

By default, SQL includes the field names as headers when you run a query. However, if you use the **CONCAT** function, SQL doesn’t know what to use as a header. For this reason, you should include an alias for the combined fields to help with readability. You give the combined fields an alias by using **AS**:

1

2

SELECT CONCAT(field1, " ", field2) AS alias

FROM [table\_name]

For example, if you plan to use **CONCAT** to combine the first and last names of your company’s customers into a single expression, you could use this query:

1

2

SELECT CONCAT(first\_name, " ", last\_name) AS Customer\_Name

FROM [table\_name]

## Key takeaways

Data can be imported and combined in both spreadsheets and SQL databases. To import data into a spreadsheet, use the **IMPORTRANGE** function. To import data into a SQL table, use the **INSERT INTO**, **SELECT**, and **WHERE** commands. Use **CONCATENATE** to combine two or more data strings in spreadsheets. In SQL, use the **CONCAT** function to combine fields.

**Step-by-Step: Merge text strings to gain insights**

This reading outlines the steps the instructor performs in the following video, [Merge text strings to gain insights](https://www.coursera.org/learn/analyze-data/lecture/9V6L5). In the video, the instructor uses SQL’s **CONCAT** function to combine strings from multiple columns to create a new column. Additionally, the instructor uses other SQL commands such as **AVG**, **GROUP BY**, and **ORDER BY** to gain insights about the new column.

Keep this step-by-step guide open as you watch the video. It can serve as a helpful reference tool 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 would like to follow along with the instructor, you will need to log in to your BigQuery account to use the open (public) dataset called **new\_york\_citibike**. The table you will use is called **citibike\_trips**. You previously used a BigQuery public dataset in the activity, [Hands-On Activity: Analyze weather data in BigQuery](https://www.coursera.org/learn/analyze-data/quiz/yRIIz/hands-on-activity-analyze-weather-data-in-bigquery). Review that activity if you need a refresher on loading public datasets!

empty alt text

**Example: Use CONCAT on the bike sharing dataset**

The **CONCAT** function can combine data from separate columns to provide new insights.

* In the BigQuery editor, enter **SELECT** and press Enter (Windows) or Return (Mac).
* Enter **usertype,** on line 2.
* On line 3, enter **CONCAT(start\_station\_name," to ", end\_station\_name)** to combine the names of the beginning and ending stations for each trip in a new column. This will create one column of routes.
* Enter **AS route,** at the end of line 3 to name the column route.
* On line 4, enter **COUNT (\*) as num\_trips,** to count the number of trips. The asterisk tells SQL to count the number of rows you’re selecting. Each row represents a trip, so you can count all of the rows you’ve selected to count the number of trips.
* Next, calculate the average trip duration for each route. On line 5, enter:  **ROUND(AVG(cast(tripduration as int64)/60),2) AS duration**

This line of code accomplishes several tasks:

* It uses the **CAST** function to cast **tripduration** as an integer and divides that number by 60 to convert the number from seconds to minutes.
* It uses the **AVG** function to find the average duration of each route.
* It uses the **ROUND** function to round the output to 2 decimal places.
* It uses the **AS** command to give this output the alias **duration**.

**Note 1:** BigQuery stores numbers in a 64-bit memory system, which is why there's a 64 after integer in this case.

**Note 2:** While explaining this code, the instructor says "divide by the number of rows." Instead, they meant "divide by 60."

* Enter **FROM** on line 6 and press return.
* Enter **`bigquery-public-data.new\_york.citibike\_trips`** on line 7 (enclosed in back-ticks).
* Enter **GROUP BY** on line 8.
* Enter **start\_station\_name, end\_station\_name, usertype** on line 9.
* Enter **ORDER BY** on line 10 to tell SQL how to organize this data.
* Enter **num\_trips DESC** on line 11 to sort it in descending order.
* Enter **LIMIT 10** on line 12.
* Your completed query should match the following code:

1

2

3

4

5

6

7

8

9

10

11

12

SELECT

  usertype,

  CONCAT (start\_station\_name," to ", end\_station\_name) AS route,

  COUNT (\*) as num\_trips,

  ROUND(AVG(cast(tripduration as int64)/60),2) AS duration

FROM

  `bigquery-public-data.new\_york.citibike\_trips`

GROUP BY

  start\_station\_name, end\_station\_name, usertype

ORDER BY

  num\_trips DESC

LIMIT 10

* Select **RUN** to view the results.

Now you can easily read these route names and trace them back to real places. You can also explore the types of customers taking each route. This type of information can help decision-makers at the bike-sharing company understand their user base in different parts of the city and where to keep more bikes for people to rent.

Great to see you back.

In this video, we'll build on

what we've learned about CONCATENATE

and IMPORTRANGE by exploring a new SQL query: CONCAT.

You might remember that CONCATENATE is a function

that joins together two or more text strings.

As a quick reminder,

a text string is a group of characters within

a cell most often composed of letters.

You've seen how that works within a single spreadsheet.

But there's a similar function in

SQL that allows you to join

multiple text strings from multiple sources, CONCAT.

Let's use CONCAT to combine strings

from multiple tables to create new strings.

For this example, we'll use open data from Citi Bike,

which is a public bicycle sharing system in New York.

As you've learned earlier,

open data initiatives have created

a ton of data for analysts to use.

Openness or open data is free access,

usage, and sharing of data.

It's a great resource if you want to practice or

experiment with the data analysis tools

you've been learning here.

You have open access to

the New York city bike-sharing data,

which has information about

the use of shared bikes across the city.

Now we can use CONCAT to pull and

concatenate data from different columns stored here.

The first thing we need to do is

figure out which columns we need.

That way we can tell SQL where the strings we want are.

For example, the bike-sharing company

has two different kinds of customers;

one-time paying customers and subscribers.

Let's say we want to find out what routes are

most popular with different user types.

To do that, we need to create strings of

recognizable route names that we can count and sort.

We know that the information we need

is in the stations and trips table.

We'll start building our query from there.

First, we'll input SELECT user type to

let SQL know that we want the user type as a column.

Then we'll use CONCAT

to combine the names of the beginning

and ending stations for each trip in a new column.

This will create one column

based on the routes people take.

We also need to input a title for this new column.

We'll type in, AS route,

to name the route column using those beginning and

ending station names we combined with CONCAT.

This will make these route names

easy for us to read and understand.

After that, we want SQL to count the number of trips.

So we'll input COUNT to do that.

We can use an asterisk to tell it to count up

the number of rows in the data we're selecting.

In this case, each row represents a trip,

which is why we can just count all

of the rows we've selected.

We'll name this output as num\_trips.

Play video starting at :2:46 and follow transcript2:46

Now let's also get the average

trip duration for each route.

In this case, we don't need the exact average,

so we can use the ROUND function to round up.

We'll put that first and then in the parentheses

use average to get the average trip duration.

We'll also want this data to be in

integer form for this calculation,

so we'll input cast as int 64.

Big query stores numbers in a 64-bit memory system,

which is why there's a 64 after integer in this case.

Next, we'll divide it by the number

of rows and tell it how

far we want it to round, two decimal places.

We'll name this output as duration.

We'll need to tell SQL where this information is stored.

We'll use FROM and the location we're pulling it from.

Play video starting at :3:42 and follow transcript3:42

Since we're using COUNT and

AVERAGE functions in our select clause,

we have to use GROUP BY to group together summary rows.

Let's group by the start station,

the end station, and the user type for this query.

Finally, we'll use ORDER BY to

tell it how we want to organize this data.

For this, we want to figure out

the most common trips so we can input the number of

trips column and use DESC to put it in descending order.

Finally, we only want the top 10,

so let's add LIMIT 10.

Now thanks to CONCAT,

we can easily read

these route names and trace them back to real places.

We can see which kinds of

customers are taking which routes,

which can help the bike-sharing company

understand their user base

in different parts of the city and where

to keep more bikes for people to rent.

Being able to combine multiple pieces of data

can give you new ways to organize and analyze data.

There's a lot of different tools to help you do that.

Now you've seen CONCAT in action,

and later you will come across

another similar query, JOIN.

But up next, we'll talk more about

working with strings. See you soon.

**Step-by-Step: Strings in spreadsheets**

This reading outlines the steps the instructor performs in the next video, [Strings in spreadsheets](https://www.coursera.org/learn/analyze-data/lecture/syKyK/strings-in-spreadsheets). In this video, the instructor demonstrates the **LEN**, **LEFT**, **RIGHT**, and **FIND** functions and discusses how you can use them to better understand your data.

Keep this step-by-step guide open as you watch the video. It can serve as a helpful reference tool 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 access the spreadsheet the instructor uses in this video, click the link to the dataset to create a copy. If you don’t have a Google account, you may download the data directly from the attachments below. Note that this is a larger database so it may take a moment or two to load.

Link to the Citi Bike dataset: [Citi Bike Trip Data](https://docs.google.com/spreadsheets/d/1ZetYs2n7csR1pI92hOuMfV_RXmxbOh1UOOxx1KdDv0E/template/preview?usp=sharing&resourcekey=0-qpI7g9md35n648x5jgNwaQ).

OR

[Citi Bike Trip Data](https://d3c33hcgiwev3.cloudfront.net/-eSTMJ-bSD2WQNV_8mQn2A_f258cf38d2ad4f90baf1ad10e7bf3be1_Citi-Bike-Trip-Data.xlsx?Expires=1706832000&Signature=E73wsQQOnGNOcfPm1KKbElXVG8moBmCIIiZen7DtEM-Hii8z5x2uuaJ~WTBVrMyfuYDkednftWR24ob3NDfGrh1a8k9aSGAWKTInJkd3iFyqFwgnBV8xAS1SDSYkamBzU8Dn~9kaF7ESs6h1qWuYOf3sFvHKr13wxeYqj1RojWg_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[XLSX File](https://d3c33hcgiwev3.cloudfront.net/-eSTMJ-bSD2WQNV_8mQn2A_f258cf38d2ad4f90baf1ad10e7bf3be1_Citi-Bike-Trip-Data.xlsx?Expires=1706832000&Signature=E73wsQQOnGNOcfPm1KKbElXVG8moBmCIIiZen7DtEM-Hii8z5x2uuaJ~WTBVrMyfuYDkednftWR24ob3NDfGrh1a8k9aSGAWKTInJkd3iFyqFwgnBV8xAS1SDSYkamBzU8Dn~9kaF7ESs6h1qWuYOf3sFvHKr13wxeYqj1RojWg_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

**Note:** If the directions in the video do not work for the version of Excel you have, visit the free online training center [Microsoft Excel for Windows Training](https://support.microsoft.com/en-us/office/excel-video-training-9bc05390-e94c-46af-a5b3-d7c22f6990bb), and search for these functions to learn how to use them in Excel.

empty alt text

**Example 1: The LEN function**

The **LEN** function calculates a string’s length. Use this formula to check the length of the datetime strings in column C.

1. Open the [Citi Bike Trip Data](https://docs.google.com/spreadsheets/d/1ZetYs2n7csR1pI92hOuMfV_RXmxbOh1UOOxx1KdDv0E/template/preview?usp=sharing&resourcekey=0-qpI7g9md35n648x5jgNwaQ) spreadsheet.
2. In cell **B2**, enter the equals sign [**=**] to begin the function.
3. Enter **LEN**, followed by an open parenthesis [**(**].
4. Select cell **C2**. Then add a close parenthesis [**)**].
5. Press **Enter**. The result, 19, indicates the string in cell **C2** is 19 characters.

**Example 2: The FIND function**

The **FIND** function locates specific characters and substrings in a string. All of the start-time (column C) and stop-time (column D) strings in the spreadsheet have a space between the date and the time. Use the **FIND** function to determine where in the string this space is located.

1. Select cell **B3** and enter the equals sign[**=**].
2. Enter **FIND** followed by an open parenthesis [**(**].
3. Enter quotation mark, then space, then close quotation mark [**" "**] to specify that you want to find a space.
4. Select cell **C3** and add a close parenthesis [**)**].
5. Press **Enter** to run the formula. This function returns 11, indicating that the space is the 11th character in the string.

**Note: FIND** is case sensitive, so always make sure you input the substring correctly.

**Example 3: The RIGHT function**

Use the **RIGHT** function to select a specific number of characters on the right side of a cell. Here, you want to return the substring that represents time, which is contained within the eight characters to the right of the space.

1. Reopen the spreadsheet so you are working with an unaltered version of the document [Citi Bike Trip Data](https://docs.google.com/spreadsheets/d/1ZetYs2n7csR1pI92hOuMfV_RXmxbOh1UOOxx1KdDv0E/template/preview?usp=sharing&resourcekey=0-qpI7g9md35n648x5jgNwaQ).
2. In cell **B2**, enter the equals sign [**=**].
3. Enter **RIGHT** followed by an open parenthesis [**(**].
4. Select cell **C2** then enter a comma [**,**].
5. Enter **8** to specify that you want the function to return the eight rightmost characters in the string.
6. Add a close parenthesis [**)**] to complete the formula.
7. Press **Enter** to run the formula. The time stamp from the start-time data string has now been isolated in cell **B2**.
8. Double-click the fill handle in cell **B2** to fill the rest of the column.
9. Enter **Time** in cell **B1** to add a column header.

**Example 4: The LEFT function**

Use the **LEFT** function to select a specific number of characters on the left side of a cell. Here, you want to return the date substring, which is the 10 characters to the left of the space. These characters represent the start date.

1. Right-click cell B.
2. Select **Insert 1 column left** to create a new column **B** for the date substring.
3. Enter **Date** in cell **B1** to add a column header.
4. Enter the equals sign [**=**] in cell **B2**.
5. Enter **LEFT** followed by an open parenthesis [**(**].
6. Select cell **D2** then enter a comma [**,**]. **Note:** You added a column, so the start time column is now column **D**.
7. **Add a comma** [**,**] after **D2** in the formula.
8. Enter **10**, to indicate that you want to return the first 10 characters in the date string.
9. Add a close parenthesis [**)**] to complete the formula. **Note:** The instructor enters 11, which will return the date substring and the space.
10. Press **Enter**. The date from the start-time data string has now been isolated in the new cell **B2**.
11. Double-click the fill handle in cell **B2** to fill the rest of the column.

Hey, welcome back.

So far we've worked with strings in both SQL and spreadsheets before, and

we've learned that they usually have similar functions.

In this video, we'll take another look at LEN, LEFT, RIGHT and FIND.

Play video starting at ::16 and follow transcript0:16

You've come across these functions used in SQL, but

now you'll find out how they work in spreadsheets.

Going back to our bike sharing dataset,

let's check out one of their spreadsheets.

This is one of the Trip Data spreadsheets.

In the starttime and stoptime columns,

there are strings that contain information about date and time of each ride.

Play video starting at ::38 and follow transcript0:38

This is all useful data, but chances are we'll only

need part of the strings to make a formula or answer a question.

For example, these strings contain multiple data points, like date and time.

But if we're trying to find the average time between start times,

we won't need the date.

We can actually use LEN, LEFT and RIGHT, and

FIND to split the timestamps into separate columns if we want.

Let's build a simple formula to separate the dates in these strings.

We know that LEN tells us the length of a string.

Let's check how long these datetime strings are now.

To start, we'll input the first part of the formula.

Play video starting at :1:19 and follow transcript1:19

And then we'll just select one of the cells with the datetime string in it.

These strings are 19 characters long.

We can use the FIND function to locate specific characters in a string.

Keep in mind, this is case-sensitive.

So if you're using FIND to pull a substring,

make sure that you've input the substring correctly.

We notice that all of the datetime strings have a space separating the date

and the timestamp.

So we can actually use FIND to figure out where the date ends.

Okay, seems like the space is the 11th character in this string.

So the timestamp substring will start at character 12.

We can use the LEFT and RIGHT functions to select which parts of the string we want

to isolate in a new column.

We'll use RIGHT on one of these cells to indicate that we want to grab

the right side.

Play video starting at :2:18 and follow transcript2:18

And like we've come across before, LEFT actually works exactly the same way.

Now we can apply that to the rest of column C to pull those timestamps.

Play video starting at :2:29 and follow transcript2:29

As a data analyst, being able to work with strings is a key skill,

especially when you find yourself working with data from outside sources.

Hopefully you're a little bit more comfortable applying LEN, RIGHT, LEFT and

FIND functions in both SQL and spreadsheets.

Later on, we'll use these functions to perform even more complicated formulas,

so feel free to try them out on some data yourself,

maybe even some open data like we've been using today.

See you later.

# Manipulate strings with SQL

An important part of a data analyst’s job is knowing how to convert and manipulate data for analysis. One way data analysts manipulate strings is to concatenate them, which means to join together two or more text strings. Once strings are concatenated, they form a new, longer text string for analysis. In this reading, you'll learn about different SQL functions that can be used to concatenate strings.

## CONCAT in action

Here are some examples of how you might use **CONCAT** as you work with data.

### CONCAT

You’re working with the marketing team on an email campaign, and you need to generate full names from your database’s first and last name columns. SQL's **CONCAT** function allows you to join together two or more string values, simplifying this task, as follows:

1

2

3

4

SELECT

    CONCAT(first\_name, ' ', last\_name) AS full\_name

FROM

    customers;

In this example, **CONCAT** merges the **first\_name** and **last\_name** fields to create a new field called **full\_name**. The space (**' '**) separator ensures the full name appears properly.

### CONCAT\_WS

Now, you're tasked with creating a report that includes a website's URL components: the protocol (http), domain name (**your\_company**), and domain (**com**). You'd use **CONCAT\_WS**, which stands for **CONCAT With Separator**, to achieve this. It's similar to **CONCAT**, but it includes a separator, such as a space or period, between the strings.

1

SELECT CONCAT\_WS('.', 'www', 'your\_company', 'com') as website FROM web\_data;

Here, **CONCAT\_WS** adds a period (**'.'**) between each part of the website URL, ensuring the URL is in the correct, navigable format.

### CONCAT **with** ||

In BigQuery, you can use the **||** operator to concatenate strings. For instance, if you're working with a dataset of book information and want to create a full title by combining the book's name and its edition, you could use **||**, like so:

1

SELECT book\_name || ' - ' || edition AS full\_book\_title FROM library;

This script combines the book name and edition, separated by a hyphen for clarity, providing a complete, informative title for your records.

**Note:** In some other SQL environments, you cannot use the **||** operator to concatenate strings. You must use **+** instead. For example, to concatenate the strings **'Google'** and **'.com'** in Microsoft SQL server, you would use:

1

SELECT 'Google' + '.com'

Always ensure you're using the correct syntax for the specific SQL environment you're working in!

## Concatenate strings with SQL

Review the table below as a summary of the **CONCAT** function and its variations in SQL.

| **Function/ operator** | **Use** | **Example** | **Result** |
| --- | --- | --- | --- |
| **CONCAT** | Concatenate strings to create new text strings | **CONCAT('Google', '.com')** | **Google.com** |
| **CONCAT\_WS** | Concatenate two or more strings together with a separator between each string | **CONCAT\_WS(' . ', 'www', 'google', 'com')** | **www.google.com** |
| **||** | Concatenate two or more strings together with the **||** operator | **'Google' || '.com'** | **Google.com** |

## Key takeaways

In SQL, **CONCAT** is a function that joins strings together to create new text strings. This is useful for creating new variables or features for data analysis, as well as more readable and informative output. In this way, **CONCAT** can simplify your data analysis and make you more efficient.

### 1.

Question 1

Which SQL function combines groups of text strings from multiple cells in order to create a new string?

1 point

**CONCAT**

**COMBINE**

**CONSOLIDATE**

**CONNECT**

### 2.

Question 2

What SQL clause can be added to this query to ensure only the first 50 results are returned?

1

2

3

SELECT \*

FROM Leaf\_Database

WHERE tree\_type = maple

1 point

**LIMIT 50**

**FIRST 50**

**RETURN 50**

**ONLY 50**

### 3.

Question 3

A data professional runs a query that will return a dataset containing numbers out to five decimal places. Which SQL function will limit the records to two decimal places?

1 point

**NUM**

**LIMIT**

**ROUND**

**LEN**

### 4.

Question 4

Which function will return the number of characters in spreadsheet cell F8 in order to confirm it contains exactly 15 characters?

1 point

**=LEN(F8, 15)**

**=LEN(F8)**

**=LEN(15)**

**=LEN(15, F8)**

Hi there. Data analysts

spend a lot of time problem-solving,

and that means there's going to be

times when you get stuck,

but the trick is knowing what to do when that happens.

In this video, we'll talk about

the importance of knowing how to get help,

whether that means asking someone else for

help or searching the internet for answers.

Asking other people about

a problem you're having can help

you find new solutions that move a project forward.

It's always a good idea to

reach out to your peers and mentors,

especially if they're working with you on that project.

Your team members have valuable

knowledge and insight that can

help you find the solution you need to get unstuck.

Sometimes we spend a lot of

time spinning our wheels saying,

"I can do this myself," but we can be way more

productive if we engage with other people,

find new resources to lean on and try

to get as many voices as we can involved.

For example, let's say you're working with

the bike trip time data from the previous videos.

Maybe you're trying to find the average time

between bike rides in a given month.

Calculating the difference between

bike rides before midnight is easy,

but you can run into a problem if

the elapsed time crosses into the next day.

If someone went on a bike ride at 11:00 PM,

but the next ride wasn't until 06:00 AM,

your formula would return a negative number

because the end time is less than the start time.

You know that you can add one minus

the start time if two bike rides

start and end on different days,

but that formula won't work on

times that happened in the same day,

and it's pretty inefficient to scroll through

every bike ride to pinpoint these special cases.

You need to find a way to build a conditional formula,

but you aren't sure how.

You decide to check in with other analysts

working on your team to see if they have any ideas.

You could send them a quick email,

or stop by their desk,

to find out if they have a minute

to talk it over with you.

Turns out they had a similar problem

on a previous project,

and they're able to show you a conditional formula that

you could use to speed up your calculations.

Great! They suggest using an IF formula like this.

This basically says that,

"if the end time is larger than the start time,

replace the standard end time minus start

time formula with one minus start time plus end time."

Now it's also possible that your team members

don't have an answer; that's okay too.

There's definitely someone else with

the same problem asking the same questions online.

Knowing how to find solutions online is

an incredibly valuable problem-solving tool

for data analysis.

There's also all kinds of forums where

spreadsheet users can ask questions,

and you never know what you can turn

up with just a basic search.

For example, let's say you look

at "calculate number of hours between

times" spreadsheets and find

a helpful walk-through for

a more complicated formula using MOD.

This flips the negative values into

positive ones, solving your calculation problem.

Whether you're asking someone you know

or searching the internet for answers,

reaching out for help can give you

some really interesting solutions and

new ways to solve problems for future analysis.

Coming up, we'll learn even more about searching

for solutions online. See you soon.

# Advanced spreadsheet tips and tricks

Like a lot of the things you’re learning in this program, spreadsheets will get easier the more you practice. This reading provides you with a list of resources that may help advance your knowledge and experience with spreadsheet functions and functionality. The goal is to provide you with access to a variety of advanced tips and tricks that will help make you more efficient and effective when working with spreadsheets to perform data analysis. Review the description of each resource below, click the links to learn more, and save or bookmark any links that are useful to you. You can immediately start practicing anything that you learn to increase the chances of your understanding and to build your familiarity with spreadsheets. This reading provides a range of resources, so feel free to explore the ones that are applicable to you and skip the ones that aren’t.

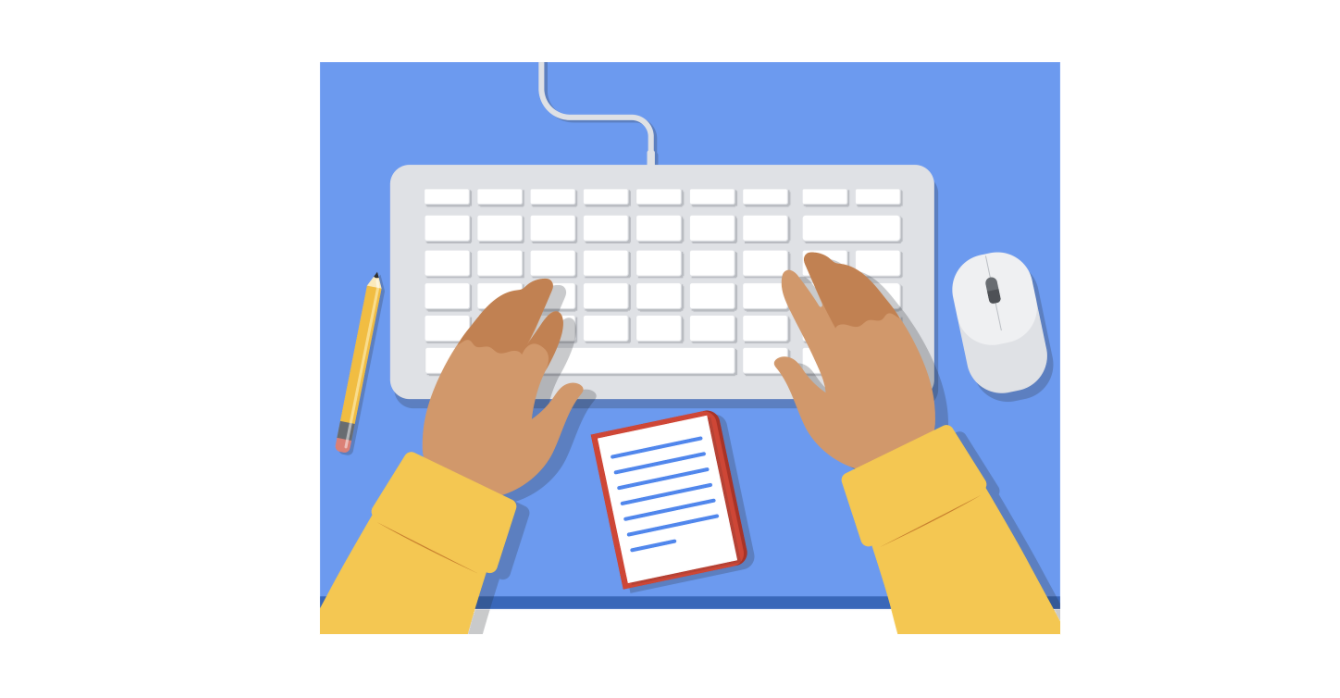
### Google Sheets

* [**Keyboard shortcuts for Google Sheets:**](https://support.google.com/docs/answer/181110)This is a great resource for quickly learning a range of keyboard shortcuts that can make regular tasks quicker and easier, like navigating your spreadsheet or accessing formulas and functions. This list contains shortcuts for the desktop and mobile versions of Google Sheets so that you can apply them to your work no matter what device you are using.
* [**List of Google Sheets Functions**](https://support.google.com/docs/table/25273?hl=en)**:** This is a comprehensive list of the Google Sheets functions and syntax. Each function is listed with a link to learn more.
* [**23 Must-Know Google Sheet Formulas**](https://blog.golayer.io/google-sheets/google-sheets-formulas)**:** This blog article from Layer summarizes and describes 20 of the most useful Google Sheets formulas.
* [**18 Google Sheets Formula Tips and Techniques:**](https://www.benlcollins.com/spreadsheets/google-sheets-formulas-techniques/) These are tips for using Google Sheets shortcuts when working with formulas.

### Excel

* [**Keyboard shortcuts in Excel:**](https://support.microsoft.com/en-us/office/keyboard-shortcuts-in-excel-1798d9d5-842a-42b8-9c99-9b7213f0040f?ui=en-US&rs=en-US&ad=US)Earlier in this list, you were provided with a resource for keyboard shortcuts in Google Sheets. Similarly, this resource provides a list of keyboard shortcuts in Excel that will make performing regular spreadsheet tasks more efficient. This includes keyboard shortcuts for both desktop and mobile versions of Excel, so you can apply them no matter what platform you are working on.
* [**222 Excel shortcuts:**](https://exceljet.net/keyboard-shortcuts)A compilation of shortcuts includes links to more detailed explanations about how to use them. This is a great way to quickly reference keyboard shortcuts. The list has been organized by functionality, so you can go directly to the sections that are most useful to you.
* [**List of spreadsheet functions:**](https://exceljet.net/excel-functions)This is a comprehensive list of Excel spreadsheet functions with links to more detailed explanations. This is a useful resource to save so that you can reference it often; that way, you’ll have access to functions and examples that you can apply to your work.
* [**List of spreadsheet formulas:**](https://exceljet.net/formulas)Similar to the previous resource, this comprehensive list of Excel spreadsheet formulas with links to more detailed explanations and can be saved and referenced any time you need to check out a formula for your analysis.
* [**Essential Excel Skills for Analyzing Data:**](https://learntocodewith.me/posts/excel-skills/)This blog post includes more advanced functionalities of some spreadsheet tools that you have previously learned about, like pivot tables and conditional formatting. These skills have been identified as particularly useful for data analysis. Each section includes a how-to video that will take you through the process of using these functions step-by-step, so that you can apply them to your own analysis.
* [**Advanced Spreadsheet Skills:**](https://www.slideshare.net/markjhonoxillo/advanced-spreadsheet-skills)Mark Jhon C. Oxillo’s presentation starts with a basic overview of spreadsheet but also includes advanced functions and exercises to help you apply formulas to actual data in Excel. This is a great way to review some basic concepts and practice the skills you have been learning so far.

There are lots of resources online about advanced spreadsheet tips and tricks. You'll probably discover new resources and tools on your own, but this list is a great starting point as you become more familiar with spreadsheets.



Hi. My name is Layla, and I'm

an analytical lead at Google.

An analytical lead is someone who helps

advertisers understand the value

of their advertising dollars.

We also help them understand if

they were to spend another dollar on ads,

where should they spend that dollar,

and what could they expect to get from it.

The skill set involved in this kind of role

has all to do with being able to

look at a dataset and make sense of it,

and then tell a story to people who

maybe don't have that same experience level with data.

What's going on in the data?

What's driving growth for your client or your company?

What could they do more of or less of to

drive more of what they want to happen?

The analyze stage is like preparing a fabulous meal.

You have done all the cleaning and

the preparing and the cooking, and you're finally

able to take a bite and to see if what you're originally

hoping to happen or what you were

expecting, to see if that is really the case.

Is it delicious? Is it exactly like you expected?

Or is the consistency a little

off and you need to add a little bit more salt?

The analysis stage begins

once you've prepped and cleaned your data.

You don't want to have those blank fields

that will throw you off

or duplicate entries that

will enlarge your dataset beyond what's actually true.

The analyze stage is where you

become the expert about your dataset.

Here, you're going to

understand all of the different fields.

You're going to understand their averages,

potentially the median of the data.

You're going to understand how

different rows in your data differ from each other.

And it's where you're going to gain the confidence to be

able to explain your findings to

an audience that maybe does not have

the same level of expertise with data that you have.

When I analyze data,

I often like to use SQL and spreadsheets.

You can use these tools to, for example, sort

your data and understand

which entries are larger than others.

Or to understand how many times something

happens by selecting the distinct entries.

Here, you can also filter out

data that you are specifically interested in analyzing,

or in a spreadsheet,

use conditional formatting to show which entries

show a more positive outcome and

which ones are maybe more negative.

Using SQL and spreadsheets to help you

through the analyze phase are absolutely crucial.

With these tools, you can format your dataset in

a way that is digestible and

then begin to tell a story with the data.

My favorite part of

working in a spreadsheet is when you finally

have that data that you want cleaned up

and exported from your SQL query.

Then you get to turn it into a pivot table and chart out

exactly the cut of data that you were

interested in looking at in the first place

and explore the trends that are happening there.

When you get to do that,

you basically unlock a whole world

of information and you get to pick

what story that you actually want to tell with

your data without just saying,

"This is the largest number;

that's the smallest number."

Here, you have to show what is happening over time

potentially or what you should

expect to see in the future.

Welcome back. Earlier, we

talked a little bit about finding resources

online to help you figure out

solutions to problems during analysis.

The internet has so much knowledge and advice to offer,

but you need to know how to find it.

In this video, we'll talk more

about finding answers online.

You might think that great data analysts don't regularly

rely on outside resources, but that's a myth.

The best data analysts

know that finding answers to their problems

online can be empowering and

give them new knowledge for the future.

Being able to find

new ideas and combine them with what you

already know can help you

come up with some amazing things.

Don't be afraid to turn to

the internet to find your answers.

It's a great resource that lots of

analysts use. Me included.

But let's talk more about how you can make sure you're

using web resources in the best way possible.

There's a combination of best practices that you can

use to guide your search for answers online.

By practicing the thinking skills

we've learned in this program, using

the right data analytics terms,

and your basic knowledge of analysis tools,

you have everything you need to find

answers and apply them to your own work.

And it starts with how you approach a problem mentally.

You've learned about different kinds of thinking skills

and how to practice them in your data analysis work.

From analytical, to mathematical, to structured thinking.

This helps build your mental model, or

your thought process, and the way you approach a problem.

Data analysts use these thinking skills to approach

a problem logically and break it into smaller parts.

Building this into your own problem-solving process

can help you pinpoint

specific questions, which you can

use to find resources more easily.

For example, maybe you keep

running into an error in your analysis.

You narrow it down to two possibilities:

your formula or the data itself.

You double check your formula,

and you see that it's correct.

So now you know that you need to make sure

that data has been entered correctly.

You consider the problem

logically and track it down to the source,

using your mental model.

Next, it's important to use

the right terms when searching for solutions.

Knowing how to frame data analytics questions

with the same language other analysts are using

will help you get more search results,

and it'll help you understand

what other analysts are saying.

For example, maybe you need to use

the left four characters of a string for a column in SQL.

How would you search for this?

Searching for "four characters in a column" is a little

vague and might not bring up specific resources.

But "left string query SQL"

uses some keywords that

other data analysts are also

using to talk about these things.

On top of being able to use

the right terms to search online,

you also need to be familiar with basic tools.

That way, when an online resource is walking

you through a new function

and a tool that you've used before,

you'll know how those tools work.

For example, if you find a spreadsheet formula online,

you need to understand how formulas work

to apply it to your own spreadsheet,

or maybe the dataset you're working with is too large

for a single spreadsheet,

and you'll need to switch to SQL.

Having a variety of tools in

your toolkit is important as a data analyst,

but just as important is knowing when to use them.

If you find yourself stuck on a problem,

it can be a good idea to take a step back

and reconsider how you're approaching a task.

We've covered a lot of tools that you can

use as a data analyst throughout this program.

Later, you'll learn one more,

R. We'll talk about R more later on,

but here's a sneak peak to get you excited for it.

R is another programming language,

but it's not a database language like SQL.

It's a programming language

frequently used for statistical analysis,

visualization, and other data analysis.

R is a little different from

other tools we've been working with,

but it's a great complement for

the tools you're already using,

and it will give you more potential solutions

when you run into problems.

Using the thinking skills we've learned throughout

this program, the right terms,

and your understanding of different analysis tools,

we'll get you ready for the next steps of this process:

actually searching for answers online.

There's a lot of resources like

program support websites and forums

where other data analysts are

asking and answering questions.

In an earlier video,

we ran into a problem trying to calculate the time

elapse between bike rides and the bike sharing data.

Maybe our first search,

"Calculate time in spreadsheets,"

didn't turn up the answers we needed.

By thinking about our specific question and

how other data analysts might be asking it,

we could change that search into

"Conditional formula for calculating

elapsed time in spreadsheets."

Now, we have more specific solutions to our problem.

Finally, being able to modify

example code to fit your own needs is so useful.

Understanding the syntax of

formulas and functions for different tools will

allow you to take what you learned

online and make it work for you,

and maybe even build on it to

create a whole new solution.

For example, the MOD formula we built to account for

trips that started and ended on

different days in our bike sharing data.

The MOD formula we found online

wasn't created for the data we were working with.

But because we are familiar with spreadsheet tools,

we were able to apply it to our data

and use it as a solution to our problem.

Great data analysts know how to find and use resources

online to help them build

new solutions to problems they face.

By using the thinking skills you've

already learned in this program and using

your knowledge of data analytics tools

and terms, you can, too.

Once you've found some answers to your problems,

you can build them into your analysis work

to get past any challenge you might face.

Hello there. In these videos,

you've been introduced to spreadsheets,

SQL, and so many other tools.

We've also talked about choosing

the right tool before you start a project.

But sometimes you find yourself stuck on

a problem during your data analysis.

That might mean it's time to

reconsider which tool you're using for the job.

For example, if you're working with a simple spreadsheet,

maybe five to ten rows and a few columns,

then pivot tables are a great way to visualize that data.

But if that spreadsheet is more than a million rows,

it'll start to crash,

making a pivot table hard to complete.

When you find yourself working with

a huge spreadsheet that keeps crashing,

you might switch to SQL to pull the data you need from

different locations in a database

instead of from a single spreadsheet.

You might remember that SQL can

handle trillions of rows of

data and is now

a standard language for working with database programs.

SQL is great for querying,

updating, and optimizing data.

But trying to analyze your data with

only SQL can get complicated.

As you continue to progress as a data analyst,

you might find yourself spending a lot of time building

long, nested queries and then debugging them.

It might be time to consider another tool,

R. R is a new tool that you'll work with later on,

but for now, I'll tell you a little bit

about it so that you can start getting excited.

R is another programming language,

but it's not a database language like SQL.

It's a programming language frequently

used for statistical analysis,

visualization, and other data analysis.

R is a little different from

other tools we've been working with,

but it's a great complement for

the tools you're already using.

With R, you'll be able to analyze and visualize data

in all kinds of new ways.

We'll talk about R more later on,

but I hope this sneak peek

gives you an exciting first look.

Having a variety of tools in

your tool kit is important as a data analyst,

but just as important is knowing when to use them.

If you find yourself stuck on a problem,

it can be a good idea to take a step

back and reconsider how you're approaching a task.

Do you have too much data for a single spreadsheet?

Switch to SQL.

Are you spending more time

debugging queries than actually analyzing data?

Maybe you should consider R. You

also know how to find answers online now.

So if you ever run into

a problem and need to try a different tool,

a quick search can be really helpful.

There might be resources online, or someone else

may have had the same problem and posted about it.

This is great if you start feeling stuck on a problem,

and you might even find a new way to use

a tool you're already familiar with.

That brings us to the end of this module.

Great job. We've covered a lot of information.

We learned about converting and formatting data,

how to combine multiple pieces of data,

and how to search for help when you

need support during your analysis.

Coming up next, you'll take on the weekly challenge.

As always, feel free to go back over

anything we've learned from these past videos.

Then I'll see you for the next video. Good luck.

**ROUND:** A SQL function that returns a number rounded to a certain number of decimal places

### 1.

Question 1

Fill in the blank: A data professional at a junior college uses the spreadsheet \_\_\_\_\_ function to locate specific characters from a listing of course codes.

1 point

**IDENTIFY**

**FROM**

**WHERE**

**FIND**

### 2.

Question 2

A data analyst works with a spreadsheet containing product information that often has very long text strings. To check for consistency, they use a function to count the number of characters in cell P12. What is the correct syntax of the function?

1 point

**=LEN(P,12)**

**=LEN(P:P12)**

**=LEN(P:12)**

**=LEN(P12)**

### 3.

Question 3

You prepare a project tracker spreadsheet. Next to each project is the name of the team member responsible. What spreadsheet tool will create a drop-down list with team member names to save you time when assigning the projects?

1 point

Conditional formatting

Data validation

Find

Pop-up menus

### 4.

Question 4

An analyst works with a dataset of financial data that is formatted in U.S. dollars. However, it should be formatted as Japanese yen. What spreadsheet tool should they use to correct the format?

1 point

Format as yen

Format as amount

Format as currency

Format as money

### 5.

Question 5

You use a SQL database to examine food truck sales destinations. The database table contains one column of lunch destinations and another column of dinner destinations. What function will combine the lunch and dinner destinations into a new column?

1 point

**JOIN**

**CONCAT**

**COMBINE**

**GROUP**

### 6.

Question 6

A data analyst at a symphony orchestra uses a spreadsheet to track how many concerts require more than 65 musicians. What spreadsheet tool can they use to change how cells appear when values equal 65 or more?

1 point

Data validation

**CONVERT**

Conditional formatting

Add color

### 7.

Question 7

Fill in the blank: A junior data analyst working with interest rate data uses the SQL function \_\_\_\_\_ to adjust 6.23456789012345678901234567890% to just one decimal place.

1 point

**ROUND**

**DECIMAL**

**SHORTEN**

**TRUNCATE**

### 8.

Question 8

Spreadsheet cell E4 contains the text string Insights. To return the substring sights, what is the correct syntax?

1 point

**=RIGHT(6, E4)**

**=RIGHT(E4, 6)**

**=LEFT(6, E4)**

**=LEFT(E4, 6)**

Module 3

[MUSIC]

Welcome back! In

the next few videos, we'll explore something called data aggregation.

Aggregation means collecting or gathering many separate pieces into a whole. For

example, the Milky Way galaxy is an aggregation of stars, dust, and gases.

So data aggregation is the process of gathering data from multiple

sources in order to combine it into a single summarized collection.

In data analytics, a summarized collection, or summary, describes

identifying the data you need and gathering it all together in one place.

For example, let's say you have a cabinet full of different puzzles.

One day, a shelf breaks, and all the boxes

topple over, scattering the puzzle pieces everywhere.

To get each puzzle organized again,

you need to identify the pieces that correspond to each particular puzzle,

gather them together and put them back into their correct boxes.

Only then can you work with these pieces and create a complete picture.

So in data,

the puzzle pieces represent the data that lives in different, separate datasets.

Getting them organized is the aggregation process.

Then the piles of pieces that complete a single puzzle become your summary.

And finally,

putting those pieces back together is like analyzing them to gain important insights.

Data aggregation helps data analyst identify trends, make comparisons and

gain insights that wouldn't be possible if each of the data elements

were analyzed on its own.

For instance, data on high school graduations for individual students can be

aggregated into a single graduation rate for an entire class.

Data can also be aggregated over a given time period to provide statistics,

such as averages, minimums, maximums, and sums.

For example, that same yearly graduation rate data can be aggregated

once again into a summary that shows us graduation rates for districts,

states, and countries.

Here's another example. Let's say you had data on real estate sales in

a particular neighborhood for each of the past 10 years.

If you aggregated all of that data, you'd be able to discover the average price

of a home in that area and how values have increased or decreased over time.

Functions are a big help in making data aggregation possible.

You'll learn how to use some of the most common ones to create your summaries soon.

In addition, we'll talk about aggregating data using something called a subquery.

You've seen SQL in action, and you understand that a query is a request for

information from a database.

So a subquery, also called an inner or

nested query, is a query within another query.

After the next several videos, you'll know how to aggregate data and

understand the tools you'll be using along the way.

Let's get started!

Hi, again. In this video,

we'll prep our data for VLOOKUP,

a data aggregation tool.

As you learned before,

data aggregation is the process of gathering data from

multiple sources in order to

combine it into a single summarized collection.

Data aggregation can give you all kinds of

information about the data you are looking at.

For example, in marketing,

you can aggregate data from an ad campaign to see how it

performed over time and for particular customers.

Travel companies use data aggregation to figure out how

much their competitors charge for a certain flight,

hotel room, or rental car type.

Then, they can make sure they price

their own products as competitively as possible.

One thing these businesses

all have in common is that they can

use VLOOKUP to help them achieve these goals.

As a reminder, VLOOKUP stands for vertical lookup.

Basically, it's a function that

searches for a certain value in

a column to return a corresponding piece of information.

Earlier, we used VLOOKUP to take

the value in one cell and search for a match in another.

We were able to match

a product code made up of numbers and letters that

lived in one spreadsheet to

the actual name of the product that lived in another.

But before any of that can happen,

we need to make sure our data is properly prepared.

As you've heard many times,

clean data is much more

likely to give you accurate results.

Let's start with the first common data-cleaning task:

different data types.

For example, a dataset might have dates formatted as

numbers, or numbers represented as

text strings instead of numeric values.

When data is not in a consistent format or

a format that the spreadsheet application recognizes,

VLOOKUP won't know what to do with

that data, and it will return an error.

Earlier, you learned how to convert

numbers to dates using the Format tool.

Now, let's focus on converting text to numeric values.

To do this, you could use

the Format menu to select a type of number,

but you could also use the VALUE function.

VALUE is a function that converts a text string that

represents a number to a numerical value.

Here's an example.

In this spreadsheet, the numbers in

column A are currently text strings.

We can confirm this by running a simple SUM function.

The syntax is equals SUM, open parenthesis,

and then the items you want to add together.

Here, it's A2 to A4.

The colon says we're including

everything between these two references.

Now you can add a closed parenthesis and

press Enter, or you can click

and drag on the cells you want

inside the parentheses to save a little bit of time.

The result is zero.

That's because the function doesn't work on text strings.

But if we apply the VALUE function,

it automatically converts that text to a numeric value.

To do that, we'll type equals VALUE,

then an open parenthesis.

Inside, we reference

the cell whose value we want to convert,

in this case A2.

Now if we close the parentheses and press Enter,

you'll notice that the 1,

2, 3 is numeric.

If we drag it down the column, the 4, 5,

6 and 7, 8,

9 also become numeric.

Now we can test it by running another SUM function.

We'll type equals SUM and an open parenthesis,

then B2, colon, and B4.

B2, B3, and B4 are included in the sum.

Close the parentheses and press Enter.

Now it shows that the total is 1,368.

The next common error comes from having

extra spaces in your spreadsheet.

As you've learned, when data is

copied from one source to another,

sometimes a few leading or trailing spaces tag along.

These can cause problems when using VLOOKUP.

We want to make sure to use

TRIM during the data- cleaning process.

TRIM automatically deletes any extra spaces

added to the cell.

Another typical mistake in VLOOKUP,

which you can easily catch during

data cleaning, are duplicates.

If there are duplicate rows in the search,

it will return only the first match it finds.

As you learned before,

Remove duplicates is a tool that automatically

searches for and eliminates

duplicate entries from a spreadsheet.

Using Remove duplicates, as you

saw in a video a little while ago,

is a great way to get rid of duplicates and help make

sure you find the right record during the lookup.

It's always good to remember that clean data

is the foundation that everything else is built on.

VLOOKUP can be a very useful data-cleaning tool.

In the next video,

we'll keep exploring more ways you

can use VLOOKUP. See you there.

Hi and welcome back.

In a previous video,

we talked about VLOOKUP for data cleaning.

We also discussed the importance of preparing

our spreadsheet before putting VLOOKUP to use.

Now we're going to experience it in action.

As a quick reminder,

VLOOKUP is a spreadsheet function

that vertically searches for

a certain value in a column to

return a corresponding piece of information.

Let's start with VLOOKUP syntax.

This example, 103 is a value to search for.

A2:B26 is the range that will be searched.

As you may remember,

VLOOKUP will not recognize column names such as A,

B, or C. We use a number to indicate the column.

Lastly, FALSE tells VLOOKUP to find an exact match.

If this said true,

the function will return only a close match,

which might not be what we want.

Now let's put VLOOKUP to use.

One of the most common things data analysts do with

VLOOKUP is populating data

in one spreadsheet from another.

Here's an example. Let's say we're working

with data that exists in two different spreadsheets,

but we need information from both

in order to answer our business question.

VLOOKUP can connect two sheets

together on a matching column

to populate one single sheet. Check it out.

In this spreadsheet, we have

employee ID numbers and their rates of pay.

In this spreadsheet we have

the same employee ID numbers and

how many hours each person worked.

We can use VLOOKUP to search for the rate of pay from

the employee rates spreadsheet

and add it to the employee hours spreadsheet.

The formula is equals VLOOKUP open parentheses,

then A2, which is

the first employee ID number

and the employee hours spreadsheet.

Next, we add a comma,

the name of the spreadsheet we want to

search in, employee rates.

Be sure to put single quotation marks around

the spreadsheet name and add

an exclamation point after it.

This is the way to reference the other spreadsheet.

Next, we add the range,

which is A2 through B5.

As you saw in a previous video,

we can also choose to add dollar signs to

lock the range with absolute cell references.

This prevents them from changing when

copying the formula to other cells.

Add another comma, then a two.

The two indicates that we want to

search for a match in the second column,

column B for rate of pay.

Finally, one more comma and we

add false to look up an exact match.

Drag the formula down the column and now we can

use a simple multiplication formula to calculate

each person's paycheck by multiplying hours

worked by our newly created pay rate column.

Great work. In an upcoming reading,

you'll learn even more about VLOOKUP and

access some helpful VLOOKUP reminders and resources.

VLOOKUP is one of the more complicated functions

, so keep practicing.

When people start out in data analytics, they often think

that those of us who've been in the field for a while know everything.

But trust me, we're all still figuring things out.

And a lot of the time that means troubleshooting.

Troubleshooting has to do with asking the right questions, and

that's what we'll focus on in this video.

We'll learn how you can use troubleshooting to solve all kinds of

problems.

To do this, we'll need to talk about some of the limitations of VLOOKUP and

then practice fixing some of the most common problems that data analysts face.

Some of the troubleshooting questions I like to ask myself:

How should I prioritize these issues?

Trying to solve lots of problems all at once can feel overwhelming.

I find it helps when you take things one at a time.

Next I ask, In a single sentence, what's the issue I'm facing?

This helps to clarify what's really going on, so

I don't get bogged down with extra details.

After all, if you don't have a clear objective before looking at the data,

you can find just about anything.

It's always best to start with your own clear understanding of the situation.

Then let the data tell you if you're on the right track or not.

The next question I ask myself is, What resources can help me solve the problem?

The internet is one of the best resources out there.

If you have a question,

chances are thousands of others run into exactly the same thing.

So a quick search can be really helpful.

And it's good to remember that people are resources, too.

Don't be afraid to ask questions.

Not only is it a great way to learn,

it can also help you build strong relationships with your colleagues.

And a final important question I think about:

How can I stop this problem from happening in the future?

If a new procedure or guideline can stop the same issue from popping up again,

that's a great time-saver.

All right. Let's start by noting that VLOOKUP only returns the first match it

finds, even if there are lots of possible matches.

Something else to keep in mind is that VLOOKUP can only return a value from

the data to the right.

It can't look left.

Play video starting at :2:8 and follow transcript2:08

Good news. There's a simple solution.

Data analysts usually get around the problem by copying and

pasting a column to the left of the data they want to look at.

This way, the lookup value is in the leftmost column and

the data they want is to the right of it.

Play video starting at :2:24 and follow transcript2:24

Here's another problem I see a lot.

Let's say the first few rows of a VLOOKUP have returned the correct result.

But when you drive the function down the column, problems start popping up.

Play video starting at :2:36 and follow transcript2:36

This is probably because the table array part of the function hasn't been locked or

made absolute.

Play video starting at :2:43 and follow transcript2:43

An absolute reference is a reference that is locked so that rows and

columns won't change when copied.

You can fix this issue by wrapping the table array in dollar signs.

As you learned a while back,

the dollar sign controls how the reference will be updated.

They make sure that the corresponding part of the reference doesn't change.

Something else that can throw off your VLOOKUP results are version

control issues.

In other words, a function worked perfectly at first, but

then something in the spreadsheet it was referencing changed.

For example, maybe a user inserted a column.

So now the columns in your function no longer direct VLOOKUP to the right place.

When something like this happens, it'll return an incorrect value.

There are a few actions data analysts can take to ensure this doesn't happen.

First, lock the spreadsheet.

This stops other people from making changes.

Play video starting at :3:35 and follow transcript3:35

To do this in Sheets, select Data, then Protected sheets and ranges.

In other spreadsheet applications, there are other tools that do the same thing.

Next, choose what you want to protect.

In this case, we want to protect the entire sheet.

Then you can set permissions to either show a warning or restrict who can edit.

Choose only you, then Done.

Play video starting at :3:58 and follow transcript3:58

But keep in mind,

there will be times when other people need to work in the spreadsheet,

so locking them out might make you pretty unpopular with your coworkers.

When that's the case, you can use MATCH, which is a function used to locate

the position of a specific lookup value and can help you with version control.

We won't get into that right now, but

just know that it's an option in case you ever need it.

The final problem we'll talk about has to do with exact and approximate matching.

When using VLOOKUP, you're likely to get different results,

depending on whether you enter the word TRUE or FALSE within your function.

TRUE tells VLOOKUP to look for approximate matches, and

FALSE tells VLOOKUP to look for exact matches.

So if a function looks like this, it's telling VLOOKUP to find the closest match

to the text or number we're looking for.

It's important to know that VLOOKUP starts at the top of a specified range and

searches downward vertically in each cell to find the right value.

It stops searching when it finds any value that's greater than or

equal to the lookup value.

That's why data analysts typically use FALSE, like this.

That way VLOOKUP only returns the exact match to what you've entered

in the lookup value.

VLOOKUP is one of the most popular lookup and reference functions in spreadsheets.

It's also one of the trickiest.

Coming up, you'll learn about more of these common challenges.

Everything you learn will help you run into fewer problems when you start using

VLOOKUP as a future data analyst.

# VLOOKUP core concepts

Spreadsheet functions can be used to quickly find information and perform calculations using specific values. **VLOOKUP**, or Vertical Lookup, is one such function that vertically searches for a certain value in a column to return a corresponding piece of information. In this reading, you’ll examine the intricacies of this extremely useful function so you understand how it works when you use it to analyze data.

## VLOOKUP functionality

**VLOOKUP** searches for a search term, called a **search\_key**, in one column of a spreadsheet. When the **search\_key** is found, the function returns the data from another column of the row from which it was located. **VLOOKUP** returns only the value that corresponds to the first item it matches. So, if there are multiple matching values, the spreadsheet will return only data about the first one.

## VLOOKUP use cases

Here are two common reasons why you might use **VLOOKUP**:

* **Populating data in a spreadsheet.** Perhaps a store manager is tracking incoming shipments before a busy holiday. They could use **VLOOKUP** to look up product ID codes in a product spreadsheet and retrieve the corresponding product information from another spreadsheet. This would help the manager know how many stock clerks they need to schedule to work when the shipments arrive.
* **Merging data from one spreadsheet with data in another.** If a teacher keeps one spreadsheet for student grades and another for attendance, they could use **VLOOKUP** to combine the spreadsheets. That way, they could search for a particular student in the attendance sheet, and **VLOOKUP** would pull the corresponding attendance record into the grades spreadsheet.

## VLOOKUP syntax

**VLOOKUP** is available in both Microsoft Excel and Google Sheets. Here, you’ll explore its syntax in Google Sheets. Refer to the resources at the end of this reading for more information about **VLOOKUP** in Microsoft Excel.

**VLOOKUP**’s syntax is:

1

VLOOKUP(search\_key, range, index, is\_sorted)

The following sections explain each of the four parts of the syntax.

### search\_key

This is the value the function will search for. It can be a number, text string, or cell reference.

### range

This is the range of cells over which the function will search and return information. The first column in the **range** is searched. When the search key is found, the index from that row is returned.

For example, if you search for the **search\_key** in column B and return the data from column D, the range would need to include columns B through D, such as the range B2:D10. If you specified a range of A2:D10, the function would search for the search term in column A.

The **search\_key** must be to the left of the information you want the function to return. This may require you to move columns around before you use **VLOOKUP**. For example, if you plan to search for the **search\_key** column D, but the information you want the function to return is in column A, you must rearrange your columns before using **VLOOKUP**.

### index

This is the position of the column that contains the data to be returned. The first column in the range is column number 1, and each column is numbered sequentially to the right.

For example, if the range is B2:D10 and you want to return a value from column D, the index number would be 3. If the index is not between 1 and the number of columns in range, the error message **#VALUE!** will be returned.

### is\_sorted

This indicates whether to return an approximate or exact match. For example, if you’re searching for Google, then google would not count as a match.

* To return an exact match, set **is\_sorted** to **FALSE**. This is recommended.
* To return an approximate match, set **is\_sorted** to **TRUE**. The nearest match (less than or equal to the **search\_key**) is returned. To use this option to obtain accurate results, you must sort your data in ascending order. But, you could still find a value.
* If neither **TRUE** nor **FALSE** are selected, the function will default to **TRUE**.

## The #N/A error

**#N/A** indicates that a matching value can't be returned because no matches were found.

## Key takeaways

Use **VLOOKUP** to search for a value in a column and return a corresponding piece of information. It’s a very useful tool for data professionals, as it enables them to combine data from multiple sources and find information quickly. Keep in mind that the column that matches the **search\_key** in a **VLOOKUP** formula should be on the left side of the data. The range must include both the column being searched and the column that contains the information being returned. **TRUE** means an approximate match, and **FALSE** means an exact match on the **search\_key**.

## VLOOKUP resources for Microsoft Excel

**VLOOKUP** may slightly differ in Microsoft Excel, but the overall concepts can still be generally applied. Refer to the following resources if you’re working with Excel:

* [How to use VLOOKUP in Excel](https://support.microsoft.com/en-us/office/vlookup-function-0bbc8083-26fe-4963-8ab8-93a18ad188a1): This tutorial includes a video to help you get a general understanding of how the **VLOOKUP** function works in Excel, as well as practical examples to look through.
* [VLOOKUP in Excel tutorial](https://www.youtube.com/watch?v=d3BYVQ6xIE4): Follow along in this video lesson and learn how to write a **VLOOKUP** formula in Excel and master time-saving useful tips and tricks.
* [23 things you should know about VLOOKUP in Excel](https://exceljet.net/things-you-should-know-about-vlookup): Explore this list of **VLOOKUP** facts, common challenges and their solutions.
* [How to use Excel's VLOOKUP function](https://edu.gcfglobal.org/en/excel-tips/how-to-use-excels-vlookup-function/1/): This article shares a specific example about applying **VLOOKUP** in your searches.
* [VLOOKUP in Excel vs Google Sheets](https://infoinspired.com/sheets-vs-excel-formula/vlookup-formula-in-excel-and-google-sheets/): This guide offers a **VLOOKUP** comparison of Excel and Google Sheets.

Question 1

Fill in the blank: The purpose of \_\_\_\_\_ is to gather data from multiple sources in order to combine it into a single, summarized collection.

1 point

data collection

data aggregation

data blending

data transfer

### 2.

Question 2

Which function can be used to change a text string in spreadsheet cell B11 to a numerical value?

1 point

**=CONVERT(B11)**

**=VALUE(B11)**

**=NUM(B11)**

**=MATCH(B11)**

### 3.

Question 3

What is the purpose of the absolute references within the function **=AVERAGE($B$2:$B$11)**?

1 point

Ensure the rows and columns will not change if the function is copied

Represent missing values in a formula or function

Remove unnecessary instructions from a formula or function

Make formulas and functions unconditional

### 4.

Question 4

In this spreadsheet, which **VLOOKUP** function will search for the height of the building in Shenzhen?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | B | C | D |
| 1 | **Location** | **Building** | **Height** | **Year completed** |
| 2 | Dubai | Burj Khalifa | 2,717 feet | 2010 |
| 3 | Shanghai | Shanghai Tower | 2,073 feet | 2015 |
| 4 | Mecca | Makkah Royal Clock Tower | 1,972 feet | 2012 |
| 5 | Shenzhen | Ping An Finance Center | 1,965 feet | 2017 |
| 6 | St. Petersburg | Lakhta Center | 1,516 feet | 2019 |
| 7 | Chicago | Willis Tower | 1,451 feet | 1974 |

1 point

**=VLOOKUP(“Shenzhen”, A2:D7, 3, FALSE)**

**=VLOOKUP(Shenzhen, A2,D7, 3, TRUE)**

**=VLOOKUP(Shenzhen, A2:D7, 2, TRUE)**

**=VLOOKUP(“Shenzhen” A2-D7, 2, FALSE)**

# Upload the employee dataset to BigQuery

This reading outlines the steps you need to perform before watching the video and following along in the step-by-step guide, [Explore how JOINs work](https://www.coursera.org/learn/analyze-data/lecture/uLZJH/how-joins-work).

**What you will need**

Download the two .csv files from the attachments below:

[Employees Table - Understanding JOINS](https://d3c33hcgiwev3.cloudfront.net/TMwinKTQQ2aYTZGdcL0Fog_84586bd2265a4888af22e8060747c8e1_Employees-Table---Understanding-JOINS.csv?Expires=1706832000&Signature=e5Ob09XXevzeSheiUMmRTui3DlWLg789tq1H9kdC3Ev1I1aez~hcUbZQEZA4sDlgaGCken3GgU8sbbnWeTvy4yfZLk2Nk-JeuqKROX3pQ2FhJrMeQTEqYINJJkaqS6irJQgf6kV9GRSz~B~AZ5oxtD9XYiibmPcTU~-TXJZjbc4_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[CSV File](https://d3c33hcgiwev3.cloudfront.net/TMwinKTQQ2aYTZGdcL0Fog_84586bd2265a4888af22e8060747c8e1_Employees-Table---Understanding-JOINS.csv?Expires=1706832000&Signature=e5Ob09XXevzeSheiUMmRTui3DlWLg789tq1H9kdC3Ev1I1aez~hcUbZQEZA4sDlgaGCken3GgU8sbbnWeTvy4yfZLk2Nk-JeuqKROX3pQ2FhJrMeQTEqYINJJkaqS6irJQgf6kV9GRSz~B~AZ5oxtD9XYiibmPcTU~-TXJZjbc4_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[Departments Table - Understanding JOINS](https://d3c33hcgiwev3.cloudfront.net/9vVSOuERRjusAUScVM-rOw_d98f88a17db84f198dee412210c13ae1_Departments-Table---Understanding-JOINS.csv?Expires=1706832000&Signature=NnACn8qXDW9rIePAMXL9jSnvXTqK6ckmPbW1hB1CYRNa6QAifM~JbkzuNHZawBPht80GbvHHUcv1T8~uNVFQ5SET2yOQOwY7sAhQjmVrOK2HsMjQ4E7hDhbjpx8jdytvIIvhITXPyxDTMDLxqC3UvAyWCT29PXQdvO~YyeSXOMM_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[CSV File](https://d3c33hcgiwev3.cloudfront.net/9vVSOuERRjusAUScVM-rOw_d98f88a17db84f198dee412210c13ae1_Departments-Table---Understanding-JOINS.csv?Expires=1706832000&Signature=NnACn8qXDW9rIePAMXL9jSnvXTqK6ckmPbW1hB1CYRNa6QAifM~JbkzuNHZawBPht80GbvHHUcv1T8~uNVFQ5SET2yOQOwY7sAhQjmVrOK2HsMjQ4E7hDhbjpx8jdytvIIvhITXPyxDTMDLxqC3UvAyWCT29PXQdvO~YyeSXOMM_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

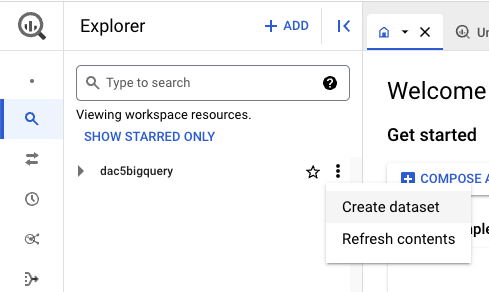
Then, log in to your BigQuery account and follow along with this reading to upload the employee data provided as two new tables.

## Prepare for the next video

### **Create a new dataset**

1. Open your BigQuery console and select the project to which you’ll upload data. For the purpose of this reading, the example project is named **dac5bigquery**. Your project will have a different name.

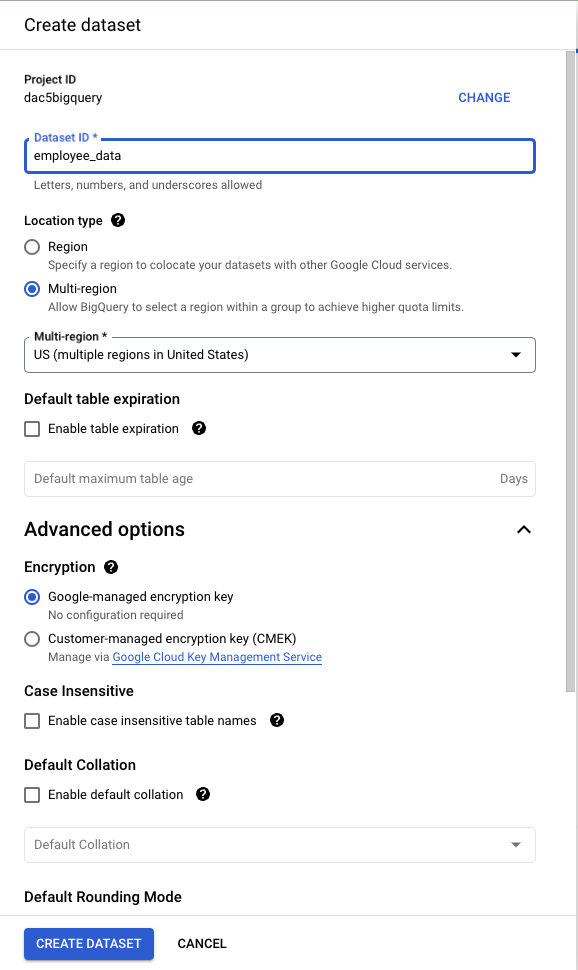
2. From the **Explorer** pane, select the **Actions** icon (the three vertical dots) next to your project name. Then select **Create dataset**.



3. In the **Create dataset** window, enter **employee\_data** for the **Dataset ID**.

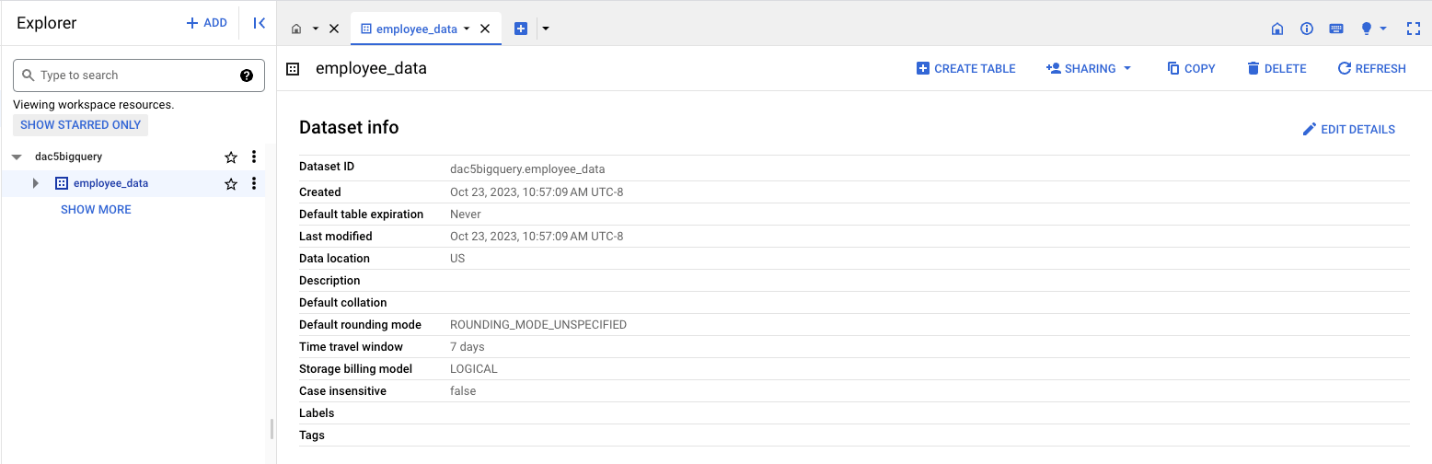
4. Make sure the **Location type** is set to **Multi-region** with **US(multiple regions in the United States)** selected.

5. Leave the **Advanced options** set to their default settings.



6. Select **CREATE DATASET** to add the dataset to your project. It will now appear under your project in the **Explorer** pane. If you don’t see the new dataset listed, select the arrow next to your project in the **Explorer** pane to expand its contents.

7. In the **Explorer** pane, select the **employee\_data** dataset you just created. The **Dataset info** window opens in the **Details** pane.



### **Create the employees table**

Create a table for the employees within your employee\_data dataset. To do this:

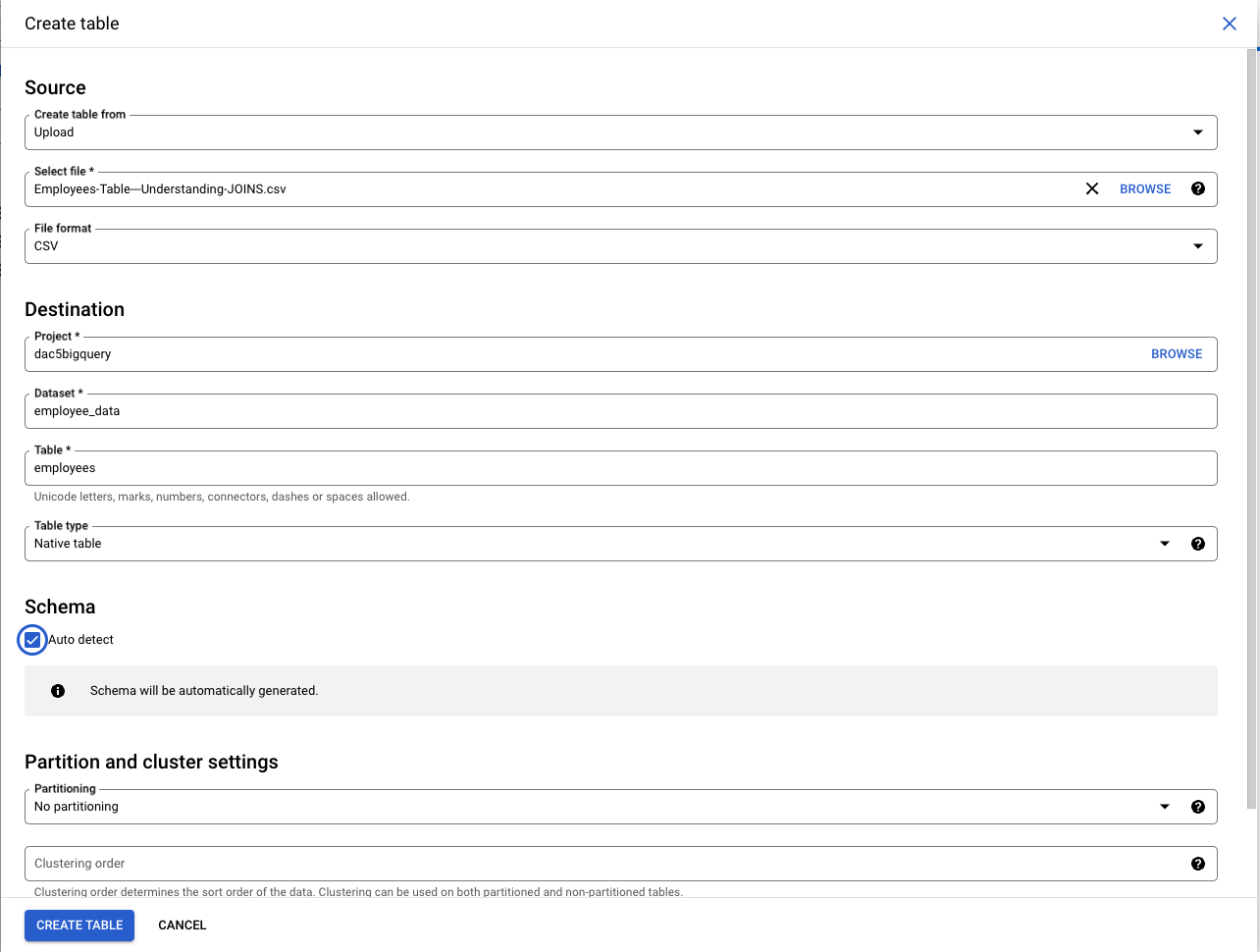
1. Select **+ CREATE TABLE** from the options in the **Details** pane. Under **Source**:

* From the dropdown in the field **Create table from**, select **Upload**.
* From the **Select file** field, select **BROWSE**. Then, find and select the [Employees Table - Understanding JOINS](https://d3c33hcgiwev3.cloudfront.net/TMwinKTQQ2aYTZGdcL0Fog_84586bd2265a4888af22e8060747c8e1_Employees-Table---Understanding-JOINS.csv?Expires=1700006400&Signature=XeqsCQhRu8D7cS7Ew8kaS3ySfbc2aYOuk~bTZZ8eBdF07KQtiMu4sAgRB4V1xboPLvCry4H61bXFNjqBT1PmbKFo8dgzPbmOUsS0msC6Fca6c-gj95HVM1PJq5pRctkQiC8OaBKml3T9KRs0lctwoqeJI-HKUo5e7hs2dF-q~EA_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A).csv File you downloaded previously.
* The file format should be automatically detected after selecting the Employee Table .csv file. If it’s not, select **CSV** from the **File format** drop-down.

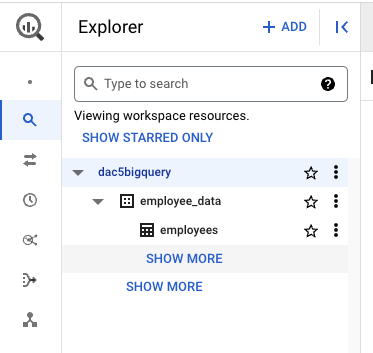
2. Under Destination:

* Enter **employees** in the **Table** field.
* **Project**, **Dataset**, and **Table type** will be automatically filled. You do not need to update these fields.

3. Under **Schema**, select the **Auto detect** checkbox.



4. Select **CREATE TABLE**. The employees table will be nested under **employee\_data** in the **Explorer** pane.



### **Create a departments table**

Next, create a table for the departments within the employee\_data dataset. To do this:

1. Select the **employee\_data** dataset from the **Explorer** panel.

2. Select **+ CREATE TABLE** from the options located in the main editor window.

3. Under **Source**:

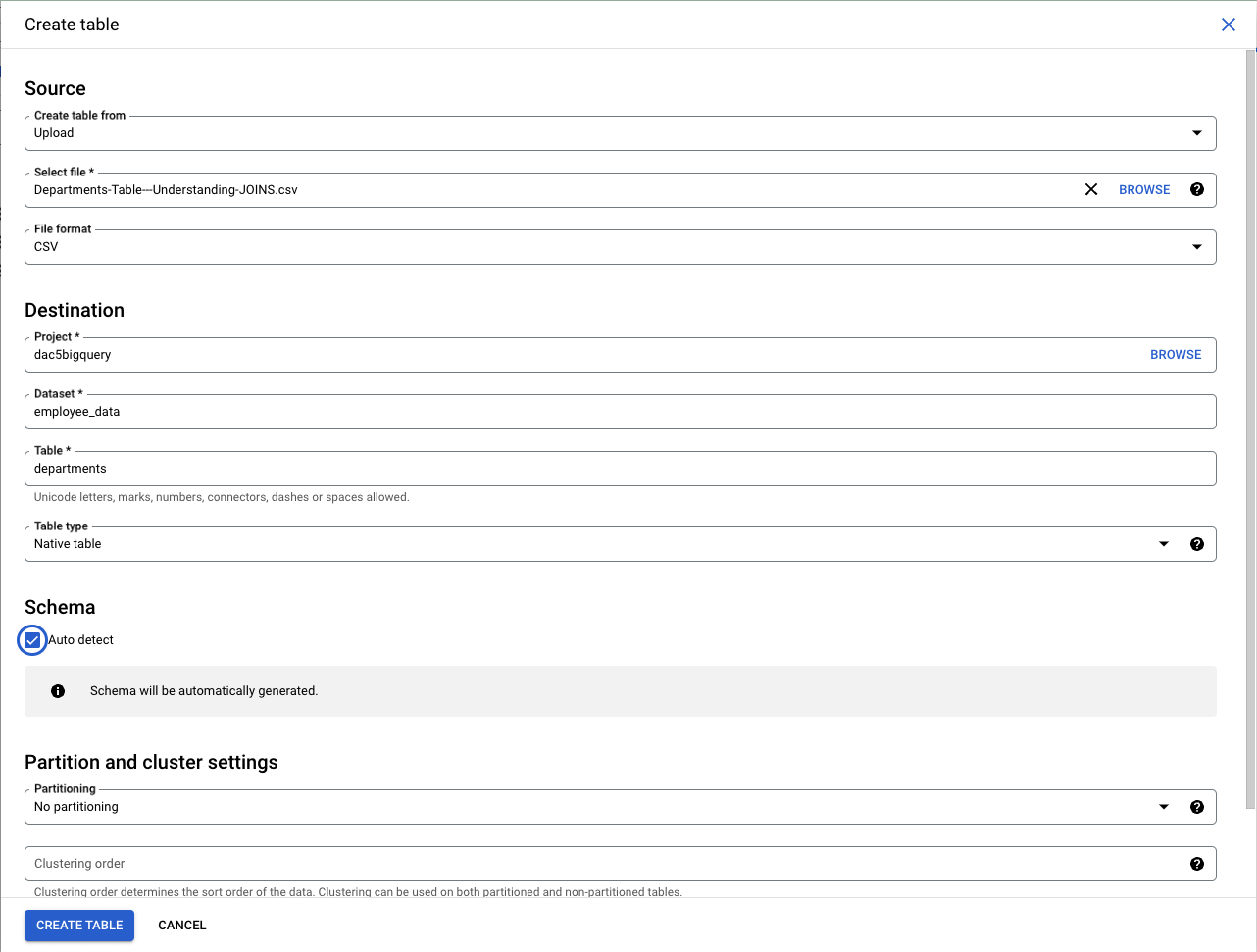
* From the dropdown in the field **Create table from**, select **Upload** from the list of options.
* From the **Select file** field, select **BROWSE**. Find and select the [Departments Table - Understanding JOINS](https://d3c33hcgiwev3.cloudfront.net/9vVSOuERRjusAUScVM-rOw_d98f88a17db84f198dee412210c13ae1_Departments-Table---Understanding-JOINS.csv?Expires=1700006400&Signature=BTJ2YLesayzmJfmfNXNbbpPk1iw5Gtn-S5VuVwvjSBrHHVcx43MzcGPWP6skOxWPQZATx2JVKw2x0aR9HKbxe80VLIK9enqXKETlIOcPYpxjDw52KFdA-Ugi4gpqegDFSl8E1bqumIk-tPxDwRe9SKnBDAeTxXGgroVc~ssKB88_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A).csv File you downloaded previously.

4. The file format may be automatically detected after selecting the Departments Table .csv file. If it’s not, select **CSV** from the **File format** drop-down.

5. Under the **Destination** section:

* Enter **departments** in the **Table** field.
* **Project**, **Dataset**, and **Table type** will be automatically filled out so you do not need to update these fields.

6. Under **Schema**, select the **Auto detect** checkbox.



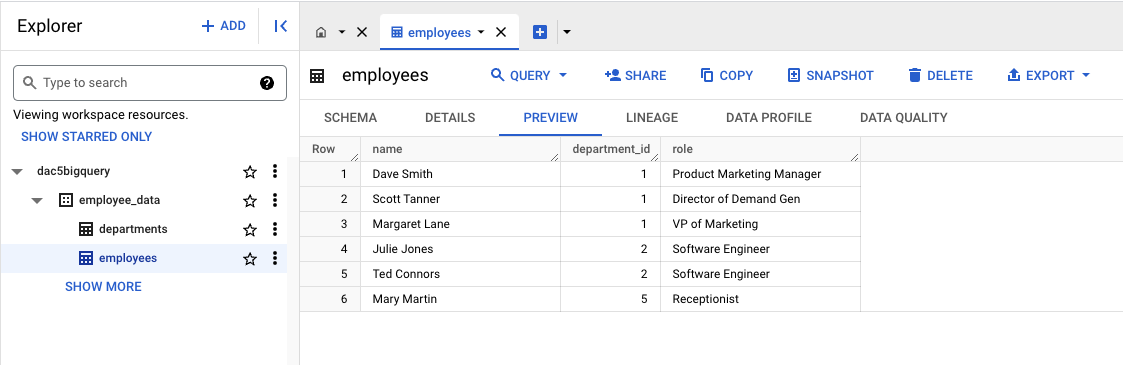
7. Select **CREATE TABLE** in the **Create Table** window. The **departments** table will appear under the **employee\_data** dataset listed within your project.

### **Verify the data**

Make sure that you’ve uploaded the tables correctly by previewing the tables you just created.

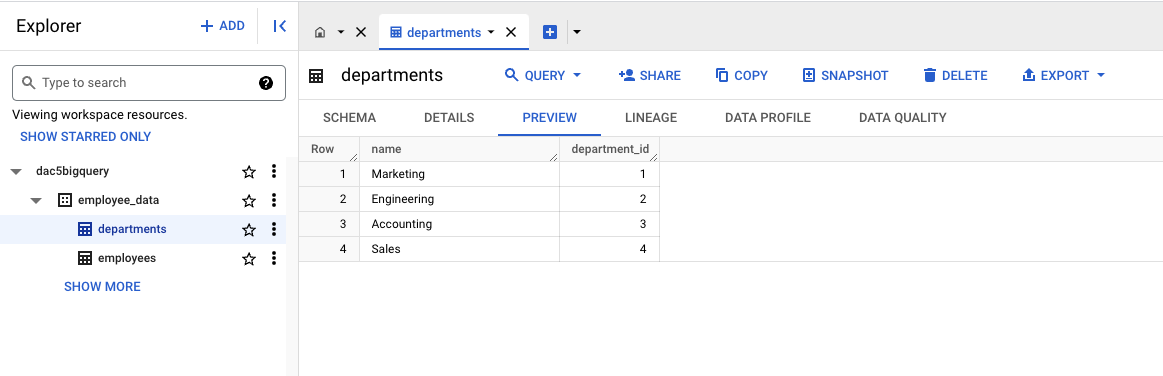
1. From the **Explorer** pane, select the **employees** table.

2. Select the **Preview** tab to verify that you have the following data:



3. Select the **departments** table.

4. Select the **Preview** tab to verify that you have the following data:

There are six rows in the table. In Row 1: name: Dave Smith; department\_id: 1; role: Product Marketing Manager. In Row 2: name: Scott Tanner; department\_id: 1; role: Director of Demand Gen.

When your data previews match these instructions, you are ready to follow along with the step-by-step guide and video on [Explore how JOINs work](https://www.coursera.org/learn/analyze-data/lecture/uLZJH/how-joins-work).

**Step-by-Step: Explore how JOINs work**

This reading provides you with the steps the instructor performs in the following video, [Explore how JOINs work](https://www.coursera.org/learn/analyze-data/lecture/uLZJH/how-joins-work). The video teaches you how to use **JOIN** in SQL to aggregate data in databases.

Keep this step-by-step guide open as you watch the video. It can serve as a helpful reference tool 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**

In order to follow along with the instructor, you will need the employee dataset uploaded into your project space. If you haven’t already uploaded this data, follow the instructions in the [Upload the employee dataset to BigQuery](https://www.coursera.org/learn/analyze-data/supplement/13KQO/optional-upload-the-employee-dataset-to-bigquery) reading.

**Common JOINs**

This video explores exactly how **JOIN**s work. A **JOIN** is a SQL clause that is used to combine rows from two or more tables based on a related column. The instructor discusses the different types of **JOIN**s in more detail in the video; here’s a quick reference you can review as you follow along:

* **INNER JOIN:** a function that returns records with matching values in both tables
* **LEFT JOIN:** a function that returns all the records from the left table (first mentioned) and only the matching records from the right table (second mentioned)
* **RIGHT JOIN:** a function that returns all records from the right table (second mentioned) and only the matching records from the left table (first mentioned).
* **OUTER JOIN:** a function that combines the **RIGHT JOIN** and **LEFT JOIN** to return all matching records in both tables.

**Example 1: INNER JOIN**

In the video, the instructor uses BigQuery to join the employees and departments tables. The following steps take you through typing the query into the query window. If you prefer, you can copy and paste the following query into the query window instead.

1. In BigQuery, select the **COMPOSE A NEW QUERY** button. BigQuery opens a query window where you can enter your query. The instructor has already executed some queries so their starting line number may be different from yours. If you’ve just opened BigQuery, your line number will be 1. It’s okay if your line numbers aren’t the same as what’s in the video.
2. In line 1 of the query window, enter **SELECT** and then press **Enter**.
3. Press Tab and then in line 2 enter **employees.name AS employee\_name**, then press **Enter**.
4. In line 3, enter **employees.role AS employee\_role**, then press **Enter**.
5. In line 4, enter **departments.name AS department\_name**, then press **Enter**.
6. In line 5, press **Backspace** to stop indenting, enter **FROM**, then press **Enter**.
7. In line 6, press **Tab**, then enter **employee\_data.employees**. Press **Enter**.
8. In line 7, press **Backspace** to stop indenting, enter **INNER JOIN**, then press **Enter**.
9. In line 8, press **Tab**, enter **employee\_data.departments ON**, then press **Enter**.
10. In line 9, enter **employees.deparment\_id = departments.department\_id**.
11. Run the query by selecting the **Run** button.

1

2

3

4

5

6

7

8

9

SELECT

    employees.name AS employee\_name,

    employees.role AS employee\_role,

    departments.name AS department\_name

FROM

    employee\_data.employees

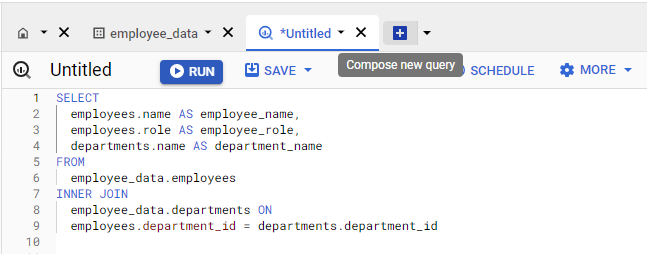
INNER JOIN

    employee\_data.departments ON

    employees.department\_id = departments.department\_id

**Example 2: LEFT JOIN**

You start a new query in BigQuery either by deleting your previous query in the current query window or by opening a new query window. To open a new query window, select the **+** button.



Now, add a new query that uses **LEFT JOIN**. If you prefer, you can copy and paste this query into the query window in BigQuery.

1. In line 1, enter **SELECT** and then press **Enter**.
2. In line 2, press **Tab** to indent the row, enter **employees.name AS employee\_name**, then press **Enter**.
3. In line 3, enter **employees.role AS employee\_role**, then press **Enter**.
4. In line 4, enter **departments.name AS department\_name**, then press **Enter**.
5. In line 5, press **Backspace** to stop indenting, enter **FROM**, then press **Enter**.
6. In line 6, press **Tab**, then enter **employee\_data.employees**, then press **Enter**.
7. In line 7, press **Backspace** to stop indenting, enter **LEFT JOIN**, then press **Enter**.
8. In line 8, press **Tab**, enter **employee\_data.departments ON**, then press **Enter**.
9. In line 9, enter **employees.deparment\_id = departments.department\_id**.
10. Run the query by selecting the **Run** button.

1

2

3

4

5

6

7

8

9

SELECT

    employees.name AS employee\_name,

    employees.role AS employee\_role,

    departments.name AS department\_name

FROM

    employee\_data.employees

LEFT JOIN

    employee\_data.departments ON

    employees.department\_id = departments.department\_id

**Example 3: RIGHT JOIN**

Use the following steps to write a **RIGHT JOIN** query. If you prefer, you can copy and paste the query below into the query window.

1. Delete the previous query or open a new query window.
2. In line 1, enter **SELECT** and then press **Enter**.
3. In line 2, press **Tab** to indent the row, enter **employees.name AS employee\_name**, then press **Enter**.
4. In line 3, enter **employees.role AS employee\_role**, then press **Enter**.
5. In line 4, enter **departments.name AS department\_name**, then press **Enter**.
6. In line 5, press **Backspace** to stop indenting, enter **FROM**, then press **Enter**.
7. In line 6, press **Tab**, then enter **employee\_data.employees**, then press **Enter**.
8. In line 7, press **Backspace** to stop indenting, enter **RIGHT JOIN**, then press **Enter**.
9. In line 8, press **Tab**, enter **employee\_data.departments ON**, then press **Enter**.
10. In line 9, enter **employees.deparment\_id = departments.department\_id**.
11. Run the query by selecting the **Run** button.

1

2

3

4

5

6

7

8

9

SELECT

    employees.name AS employee\_name,

    employees.role AS employee\_role,

    departments.name AS department\_name

FROM

    employee\_data.employees

RIGHT JOIN

    employee\_data.departments ON

    employees.department\_id = departments.department\_id

**Example 4: OUTER JOIN**

Use the following steps to write an **OUTER JOIN** query. If you prefer, you can copy and paste the query below into the query window.

1. Delete the previous query or open a new query window.
2. In line 1, enter **SELECT** and then press **Enter**.
3. In line 2, press **Tab** to indent the row, enter **employees.name AS employee\_name**, then press **Enter**.
4. In line 3, enter **employees.role AS employee\_role**, then press **Enter**.
5. In line 4, enter **departments.name AS department\_name**, then press **Enter**.
6. In line 5, press **Backspace** to stop indenting, enter **FROM**, then press **Enter**.
7. In line 6, press **Tab**, then enter **employee\_data.employees**, then press **Enter**.
8. In line 7, press **Backspace** to stop indenting, enter **FULL OUTER JOIN**, then press **Enter**.
9. In line 8, press **Tab**, enter **employee\_data.departments ON**, then press **Enter**.
10. In line 9, enter **employees.deparment\_id = departments.department\_id**.
11. Run the query by selecting the **Run** button.

1

2

3

4

5

6

7

8

9

SELECT

    employees.name AS employee\_name,

    employees.role AS employee\_role,

    departments.name AS department\_name

FROM

    employee\_data.employees

FULL OUTER JOIN

    employee\_data.departments ON

    employees.department\_id = departments.department\_id

Hey, welcome back.

So far we've checked out a few different tools you

can use to aggregate data within spreadsheets.

In this video, we'll cover how to use

JOIN in SQL to aggregate data in databases.

First, I'll tell you

a little bit about what a JOIN actually is,

and then we'll explore some of

the most common JOINs in action. Let's get started.

JOIN is a SQL clause that's used to combine

rows from two or more tables based on a related column.

Basically, you can think of a JOIN as

a SQL version of VLOOKUP which we just covered.

There are four common JOINs data analysts use,

inner, left, right, and outer.

Here's a handy visualization of

what each JOIN actually does.

We'll use these to help us understand these functions.

JOINs help you combine

matching or related columns from different tables.

When we learned about relational databases,

we refer to these values as primary and foreign keys.

Primary keys reference columns in

which each value is unique to that table.

But that table can have

multiple foreign keys which are

primary keys in other tables.

For example, in a table about employees,

the employee ID is

a primary key and the office ID is a foreign key.

JOIN use these keys to

identify relationships and corresponding values.

An inner JOIN is a function that returns

records with matching values in both tables.

If we think about our tables as

a circles of this Venn diagram,

then an inner JOIN would return the records

that exist where the tables are overlapping.

For the records to appear in the results table,

they'll have to be key values in both tables.

The records will only merge if there are

matches in both tables.

When we input JOIN into SQL,

it usually defaults to inner JOIN.

A lot of analysts will use JOIN

as shorthand instead of typing the whole query.

A LEFT JOIN is a function

that will return all the records

from the left table and

only the matching records from the right table.

Here's how you can figure out

which table is left or right.

In English and SQL we read from left to right.

The table mentioned first is

left and the table mentioned second is right.

You can also think of left as a table name to the left of

the JOIN statement and right as

a table name to the right of the JOIN statement.

In this diagram, you'll notice

that the entire left table is colored in,

and that's the overlap with

the right table which shows us that

the left table and the records it

shares with the right table are being selected.

Each row in the left table appears in

the results even if there are

no matches in the right table.

RIGHT JOIN does the opposite.

It will return all records from

the right table and only the

matching records from the left.

You can get the same results if you flip the order

of the tables and use a left JOIN.

For example, SELECT from table A,

LEFT JOIN table B is the same as SELECT from table B,

RIGHT JOIN table A.

Finally, there's OUTER JOIN.

OUTER join combines RIGHT and LEFT JOIN to

return all matching records in both tables.

This means it will return all records in both tables.

If there are records in one table without a match,

it'll create a record with no values for the other table.

Using JOINs can make working with

multiple data sources a lot easier and it can make

relationships between tables more

clear. Here's an example.

Let's say we're working with

employee data across multiple departments.

We have an employees table and

a departments table which both

have some columns like department ID.

We can use different JOIN clauses to help us

put different data from our tables and aggregate it.

Maybe we want to get a list of

employees with their department name,

excluding any employee without a department ID.

Because the department ID record is used in both tables,

we can use an INNER JOIN to return

a list with only those employees.

As a quick reminder,

analysts will sometimes just input JOIN for

an INNER JOIN but for this example, we'll write it out.

To build this query,

we'll start with SELECT and AS

to tell SQL how we want the columns titled.

Play video starting at :4:43 and follow transcript4:43

Then we'll use FROM to

tell it where we're getting this data,

in this case the employees table.

Then we'll input INNER JOIN and

the other table we're using, which is departments.

Play video starting at :4:58 and follow transcript4:58

We can specify which column and each table

will contain the matching JOIN key by writing

ON employees.department\_id

equals departments.departments\_id.

Now, let's run it, and there.

Now we've got a list of employee names and

department IDs for the employees that have those IDs.

But we could use LEFT or RIGHT join to return

a list of all employee names

and their departments when available.

Let's try both really quickly.

This will start similar to the last query,

we'll put in SELECT AS and FROM again.

But this time we'll say LEFT JOIN

and use ON like we did with the last query.

When we execute the query,

we get back this new list with

the employee names and departments.

But you'll notice there's null values.

These are places where the right table which is

departments in this case

didn't have corresponding values.

Let's try RIGHT JOIN just to test it out.

This query will be almost the same.

Only difference is that we'll use

the RIGHT JOIN clause to return

all the rows from the right table,

whether they have matching values in

the table to the left of the JOIN statement or not.

In this case, the right table is departments.

Play video starting at :6:28 and follow transcript6:28

Now, let's try out one last JOIN: OUTER.

OUTER JOIN will fetch all of

the employee names and departments.

Again, this query will

start a lot like the other ones we've done,

we'll use SELECT AS and FROM to

choose what data we want and how.

We'll grab this from the employees table,

and put FULL OUTER JOIN with

the departments table to get

all of the records from both.

We'll also use ON again here.

Now we can run this,

Play video starting at :7:1 and follow transcript7:01

and we'll get all of the employee names

and departments from these tables.

There will be nulls in the

department.name column, the employee.name

column and role column because we've joined columns

that don't have matching values, and there.

Now you know how JOINs work.

JOINs are super useful when you need to work

with data from multiple related tables.

They give you a lot of flexibility with

how you combine and view that data.

If you ever have trouble remembering what INNER, RIGHT,

LEFT, or OUTER JOIN do,

just think back to our Venn diagram.

We'll keep learning about aggregating data in

SQL next time. See you soon.

# Secret identities: The importance of aliases

In this reading, you will learn about using aliasing to simplify your SQL queries. **Aliases** are used in SQL queries to create temporary names for a column or table. Aliases make referencing tables and columns in your SQL queries much simpler when you have table or column names that are too long or complex to make use of in queries. Imagine a table name like **special\_projects\_customer\_negotiation\_mileages**. That would be difficult to re-enter every time you use that table. With an alias, you can create a meaningful nickname that you can use for your analysis. In this case **special\_projects\_customer\_negotiation\_mileages** can be aliased to simply **mileage**. Instead of having to write out the long table name, you can use a meaningful nickname that you decide.

## Basic syntax for aliasing

**Aliasing** is the process of using aliases. In SQL queries, aliases are implemented by making use of the **AS** command. The basic syntax for the **AS** command can be seen in the following query for aliasing a table:

1

2

SELECT column\_name(s)

FROM table\_name AS alias\_name;

Notice that **AS** is preceded by the table name and followed by the new nickname. It is a similar approach to aliasing a column:

1

2

SELECT column\_name AS alias\_name

FROM table\_name;

In both cases, you now have a new name that you can use to refer to the column or table that was aliased.

### **Alternate syntax for aliases**

If using **AS** results in an error when running a query because the SQL database you are working with doesn't support it, you can leave it out. In the previous examples, the alternate syntax for aliasing a table or column would be:

* **FROM table\_name alias\_name**
* **SELECT column\_name alias\_name**

The key takeaway is that queries can run with or without using **AS** for aliasing, but using **AS** has the benefit of making queries more readable. It helps to make aliases stand out more clearly.

## Aliasing in action

Let’s check out an example of a SQL query that uses aliasing. Let’s say that you are working with two tables: one of them has employee data and the other one has department data. The **FROM** statement to alias those tables could be:

1

FROM work\_day.employees AS employees

These aliases still let you know exactly what is in these tables, but now you don’t have to manually input those long table names. Aliases can be really helpful for long, complicated queries. It is easier to read and write your queries when you have aliases that tell you what is included within your tables.

## For more information

If you are interested in learning more about aliasing, here are some resources to help you get started:

* [**SQL Aliases**](https://www.w3schools.com/sql/sql_alias.asp)**:** This tutorial on aliasing is a really useful resource to have when you start practicing writing queries and aliasing tables on your own. It also demonstrates how aliasing works with real tables.
* [**SQL Alias**](https://www.sqltutorial.org/sql-alias/)**:** This detailed introduction to aliasing includes multiple examples. This is another great resource to reference if you need more examples.
* [**Using Column Aliasing**](https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.5&docsetId=sqlproc&docsetTarget=p0aymxwsvbt5wcn1lncugwjtf758.htm&locale=en)**:** This is a guide that focuses on column aliasing specifically. Generally, you will be aliasing entire tables, but if you find yourself needing to alias just a column, this is a great resource to have bookmarked.

# Use JOINs effectively

In this reading, you will review how **JOIN**s are used and will be introduced to some resources that you can use to learn more about them. A **JOIN** combines tables by using a primary or foreign key to align the information coming from both tables in the combination process. **JOIN**s use these keys to identify relationships and corresponding values across tables.

If you need a refresher on primary and foreign keys, refer to the [glossary](https://www.coursera.org/learn/analyze-data/supplement/0p8b6/glossary-terms-and-definitions) for this course, or go back to [Databases in data analytics](https://www.coursera.org/learn/data-preparation/supplement/uXqEX/databases-in-data-analytics).

## The general JOIN syntax

1

2

3

4

5

6

7

8

9

SELECT

   -- table columns from tables are inserted here

   table\_name1.column\_name

   table\_name2.column\_name

FROM

   table\_name1

JOIN

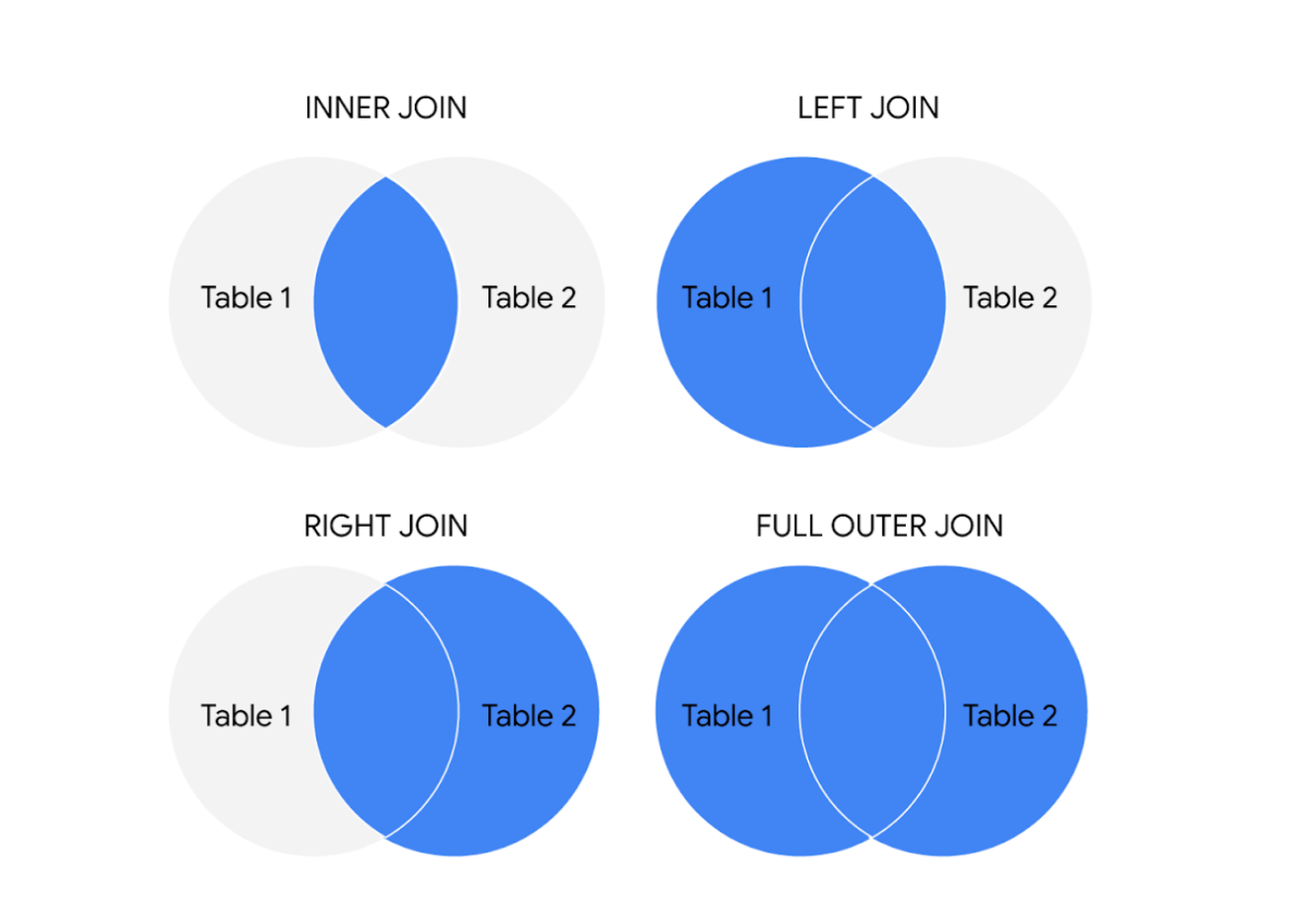
   table\_name2

ON table\_name1.column\_name = table\_name2.column\_name

As you can see from the syntax, the **JOIN** statement is part of the FROM clause of the query. **JOIN** in SQL indicates that you are going to combine data from two tables. **ON** in SQL identifies how the tables are to be matched for the correct information to be combined from both.

## Type of JOINs

There are four general ways in which to conduct **JOIN**s in SQL queries: **INNER**, **LEFT**, **RIGHT**, and **FULL OUTER**.

The circles represent left and right tables, and where they are joined is highlighted in blue

Here is what these different **JOIN** queries do.

### INNER JOIN

INNER is optional in this SQL query because it is the default as well as the most commonly used **JOIN** operation. You may see this as **JOIN** only. **INNER JOIN** returns records if the data lives in both tables. For example, if you use **INNER JOIN** for the **customers** and **orders** tables and match the data using the **customer\_id** key, you would combine the data for each **customer\_id** that exists in both tables. If a **customer\_id** exists in the **customers** table but not the **orders** table, data for that **customer\_id** isn’t joined or returned by the query.

1

2

3

4

5

6

7

8

9

SELECT

   customers.customer\_name,

   orders.product\_id,

   orders.ship\_date

FROM

   customers

INNER JOIN

   orders

ON customers.customer\_id = orders.customer\_id

The results from the query might look like the following, where customer\_name is from the customers table and product\_id and ship\_date are from the orders table:

| **customer\_name** | **product\_id** | **ship\_date** |
| --- | --- | --- |
| Martin's Ice Cream | 043998 | 2021-02-23 |
| Beachside Treats | 872012 | 2021-02-25 |
| Mona's Natural Flavors | 724956 | 2021-02-28 |
| ... etc. | ... etc. | ... etc. |

The data from both tables was joined together by matching the **customer\_id** common to both tables. Notice that **customer\_id** doesn’t show up in the query results. It is simply used to establish the relationship between the data in the two tables so the data can be joined and returned.

### LEFT JOIN

You may see this as **LEFT OUTER JOIN**, but most users prefer **LEFT JOIN**. Both are correct syntax. **LEFT JOIN** returns all the records from the left table and only the matching records from the right table. Use **LEFT JOIN** whenever you need the data from the entire first table and values from the second table, if they exist. For example, in the query below, **LEFT JOIN** will return **customer\_name** with the corresponding **sales\_rep**, if it is available. If there is a customer who did not interact with a sales representative, that customer would still show up in the query results but with a **NULL** value for **sales\_rep**.

1

2

3

4

5

6

7

8

SELECT

   customers.customer\_name,

   sales.sales\_rep

FROM

   customers

LEFT JOIN

   sales

ON customers.customer\_id = sales.customer\_id

The results from the query might look like the following where **customer\_name** is from the **customers** table and **sales\_rep** is from the **sales** table. Again, the data from both tables was joined together by matching the **customer\_id** common to both tables even though **customer\_id** wasn't returned in the query results.

| **customer\_name** | **sales\_rep** |
| --- | --- |
| Martin's Ice Cream | Luis Reyes |
| Beachside Treats | NULL |
| Mona's Natural Flavors | Geri Hall |
| ...etc. | ...etc. |

### RIGHT JOIN

You may see this as **RIGHT OUTER JOIN** or **RIGHT JOIN**. **RIGHT JOIN** returns all records from the right table and the corresponding records from the left table. Practically speaking, **RIGHT JOIN** is rarely used. Most people simply switch the tables and stick with **LEFT JOIN**. But using the previous example for **LEFT JOIN**, the query using **RIGHT JOIN** would look like the following:

1

2

3

4

5

6

7

8

SELECT

   sales.sales\_rep,

   customers.customer\_name

FROM

  sales

RIGHT JOIN

  customers

ON sales.customer\_id = customers.customer\_id

The query results are the same as the previous **LEFT JOIN** example.

| **customer\_name** | **sales\_rep** |
| --- | --- |
| Martin's Ice Cream | Luis Reyes |
| Beachside Treats | NULL |
| Mona's Natural Flavors | Geri Hall |
| ...etc. | ...etc. |

### FULL OUTER JOIN

You may sometimes see this as **FULL JOIN**. **FULL OUTER JOIN** returns all records from the specified tables. You can combine tables this way, but remember that this can potentially be a large data pull as a result. **FULL OUTER JOIN** returns all records from both tables even if data isn’t populated in one of the tables. For example, in the query below, you will get all customers and their products’ shipping dates. Because you are using a **FULL OUTER JOIN**, you may get customers returned without corresponding shipping dates or shipping dates without corresponding customers. A **NULL** value is returned if corresponding data doesn’t exist in either table.

1

2

3

4

5

6

7

8

SELECT

   customers.customer\_name,

   orders.ship\_date

FROM

   customers

FULL OUTER JOIN

   orders

ON customers.customer\_id = orders.customer\_id

The results from the query might look like the following.

| **customer\_name** | **ship\_date** |
| --- | --- |
| Martin's Ice Cream | 2021-02-23 |
| Beachside Treats | 2021-02-25 |
| NULL | 2021-02-25 |
| The Daily Scoop | NULL |
| Mountain Ice Cream | NULL |
| Mona's Natural Flavors | 2021-02-28 |
| ...etc. | ...etc. |

## For more information

**JOIN**s are going to be useful for working with relational databases and SQL—and you will have plenty of opportunities to practice them on your own. Here are a few other resources that can give you more information about **JOIN**s and how to use them:

* [**SQL JOINs**](https://www.w3schools.com/sql/sql_join.asp)**:** This is a good basic explanation of **JOIN**s with examples. If you need a quick reminder of what the different **JOIN**s do, this is a great resource to bookmark and come back to later.
* [**Database JOINs - Introduction to JOIN Types and Concepts**](https://www.essentialsql.com/introduction-database-joins/)**:** This is a really thorough introduction to **JOIN**s. Not only does this article explain what **JOIN**s are and how to use them, but it also explains the various scenarios in more detail of when and why you would use the different **JOIN**s. This is a great resource if you are interested in learning more about the logic behind **JOIN**ing.
* [**SQL JOIN Types Explained in Visuals**](https://dataschool.com/how-to-teach-people-sql/sql-join-types-explained-visually/)**:** This resource has a visual representation of the different **JOIN**s. This is a really useful way to think about **JOIN**s if you are a visual learner, and it can be a really useful way to remember the different **JOIN**s.
* [**SQL JOINs: Bringing Data Together One Join at a Time**](https://towardsdatascience.com/sql-join-8212e3eb9fde)**:** Not only does this resource have a detailed explanation of **JOIN**s with examples, but it also provides example data that you can use to follow along with their step-by-step guide. This is a useful way to practice **JOIN**s with some real data.
* [**SQL JOIN:**](https://www.dofactory.com/sql/join) This is another resource that provides a clear explanation of **JOINs** and uses examples to demonstrate how they work. The examples also combine **JOIN**s with aliasing. This is a great opportunity to see how **JOIN**s can be combined with other SQL concepts that you have been learning about in this course.

**Upload the warehouse dataset to BigQuery**

Coming up, you’re going to learn more about how to use **COUNT** and **COUNT DISTINCT** in SQL to count and return the number of certain values in a dataset.

To prepare for these activities, you will need to log in to your BigQuery account and upload the warehouse data provided below as two .csv files.

**Upload the data**

To begin, download the two .csv files from the attachments below:

[Warehouse Orders - Warehouse](https://d3c33hcgiwev3.cloudfront.net/PcJNDf5ySCqk7V2xQoiu2Q_90874562d5cd4e7c9ba848fd44d512e1_Warehouse-Orders---Warehouse.csv?Expires=1706832000&Signature=S6~aIOXocFgYebXLQhy8sWVBjeYMVD~r1owKGfby~rZLZyTu9FlMz4JfoYSEEXI8hVP416Iy3IR0n6QVh2zbfnbiMufYu~zX0Me4TbXCKG8M4Kn8OVNjcA5xqPF4rnLBLTBb4TunfwDbA9nURUBir1QHq-giTv~x8E3O1dgTjRY_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[CSV File](https://d3c33hcgiwev3.cloudfront.net/PcJNDf5ySCqk7V2xQoiu2Q_90874562d5cd4e7c9ba848fd44d512e1_Warehouse-Orders---Warehouse.csv?Expires=1706832000&Signature=S6~aIOXocFgYebXLQhy8sWVBjeYMVD~r1owKGfby~rZLZyTu9FlMz4JfoYSEEXI8hVP416Iy3IR0n6QVh2zbfnbiMufYu~zX0Me4TbXCKG8M4Kn8OVNjcA5xqPF4rnLBLTBb4TunfwDbA9nURUBir1QHq-giTv~x8E3O1dgTjRY_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

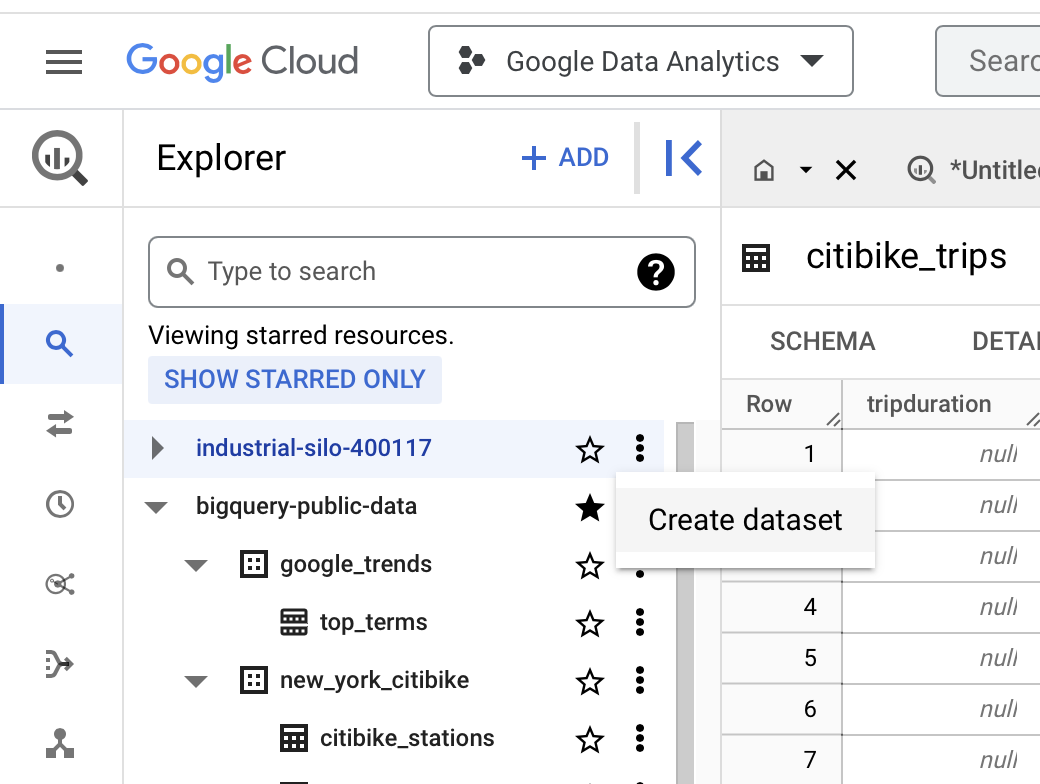
[Warehouse Orders - Orders](https://d3c33hcgiwev3.cloudfront.net/helGYTW6Tv-vyOH0l-yIPA_1a019ecb2cce49f48fd69d85f7dadce1_Warehouse-Orders---Orders.csv?Expires=1706832000&Signature=OxpqoyUyK80PtChhZK6FzFHxHtkGJiUQyNBjW2ZKOHBIlFo6ASuhBNlR3AJJUFwEFxIoXdkTdKvoXqCCN0YmYH92tWRrCSCjdUgE9yh6Aslt~qH1Kc7vQ6fvoxqjsdnBq7-pJuRshz6uRRrZ1JMPzvjDdK4xMQ6Np9RnuyXfPwE_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

[CSV File](https://d3c33hcgiwev3.cloudfront.net/helGYTW6Tv-vyOH0l-yIPA_1a019ecb2cce49f48fd69d85f7dadce1_Warehouse-Orders---Orders.csv?Expires=1706832000&Signature=OxpqoyUyK80PtChhZK6FzFHxHtkGJiUQyNBjW2ZKOHBIlFo6ASuhBNlR3AJJUFwEFxIoXdkTdKvoXqCCN0YmYH92tWRrCSCjdUgE9yh6Aslt~qH1Kc7vQ6fvoxqjsdnBq7-pJuRshz6uRRrZ1JMPzvjDdK4xMQ6Np9RnuyXfPwE_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

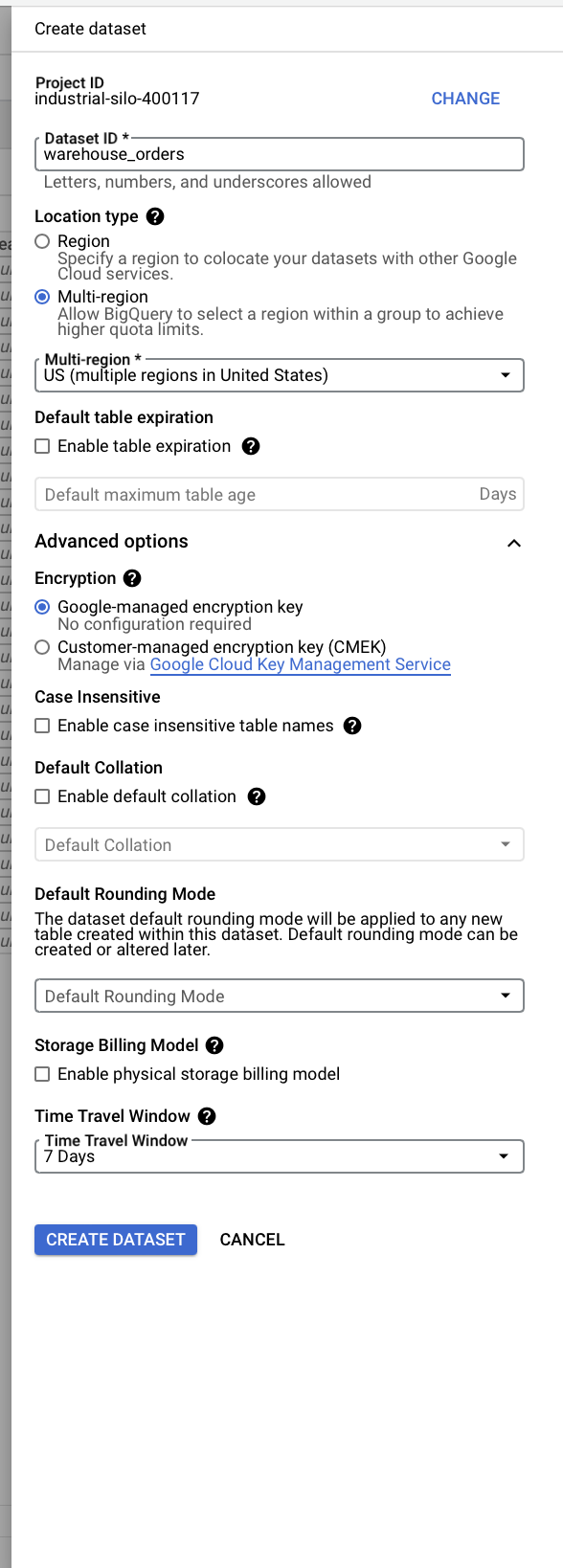
Next, complete the following steps in your BigQuery console to upload the Warehouse Orders dataset with the two Warehouse and Orders tables.

1. Open your [BigQuery console](https://console.cloud.google.com/bigquery) and select the project you want to upload the data to.

2. In the Explorer pane, select the Actions icon (three vertical dots) and select **Create dataset**.

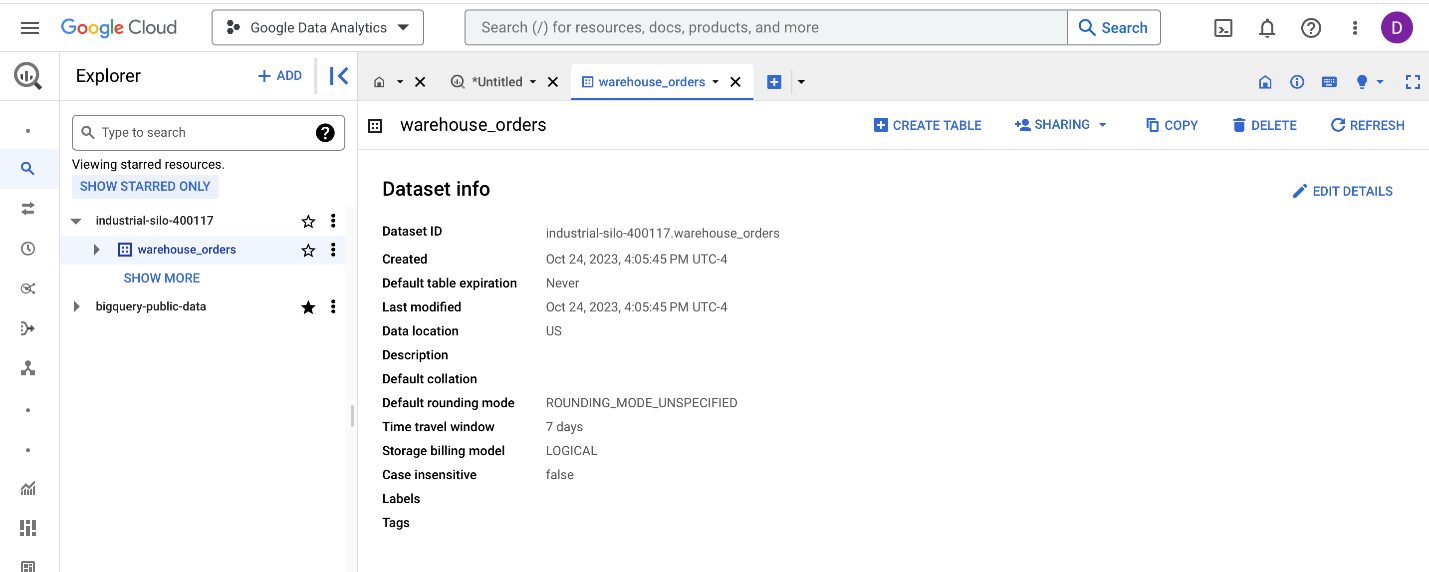


3. The name "warehouse\_orders" will be used for the dataset. Enter **warehouse\_orders** for the Dataset ID in the Create dataset pane. Make sure the **Location type** is set to **Multi-region** and that the region selected is **US (multiple regions in the United States)**. Leave the **Advanced options** at their default settings.

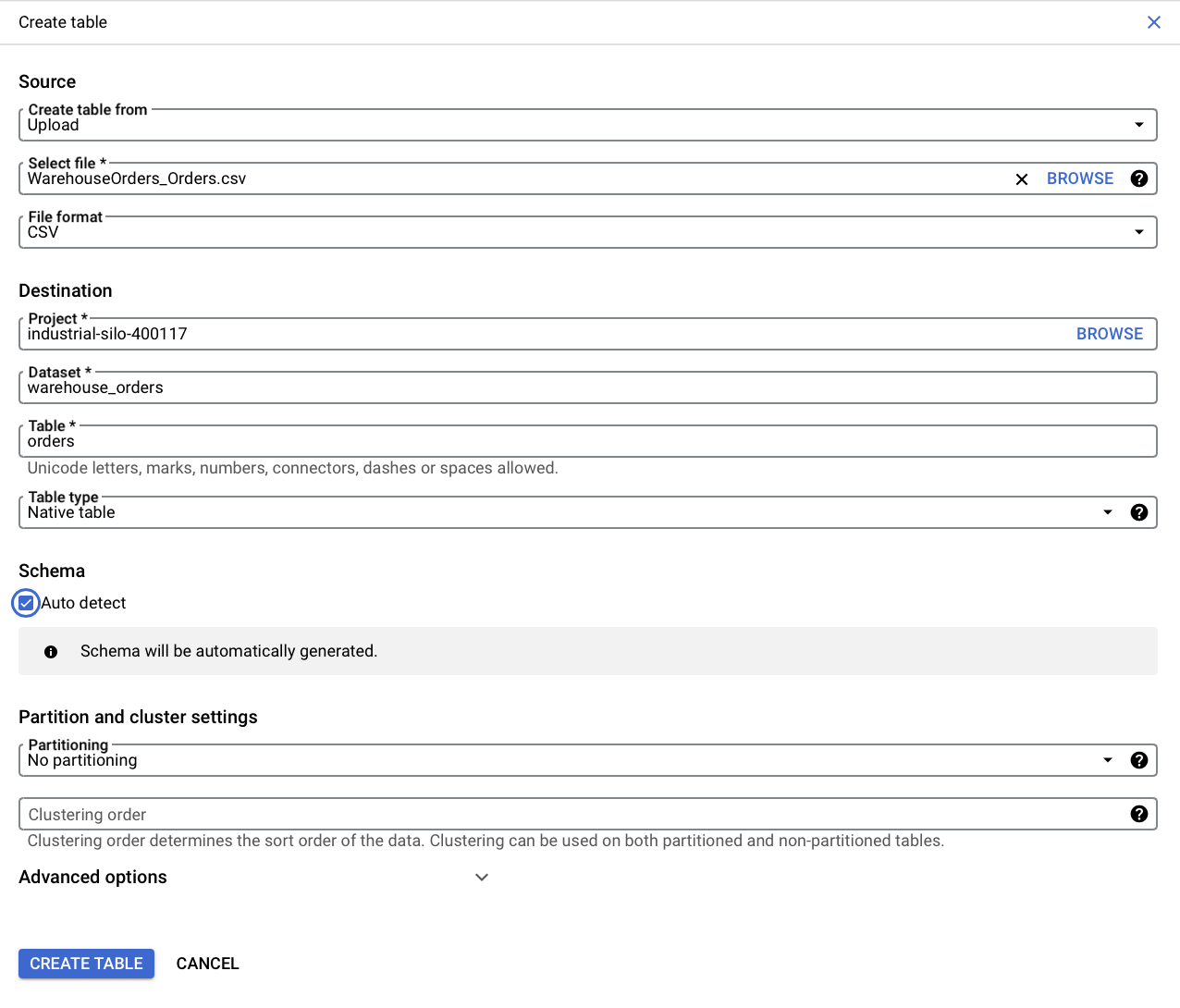


4. Select the **CREATE DATASET** button to add the dataset to your project.

5. In the Explorer, select the **warehouse\_orders** dataset you just created. You will then view the dataset info window in the main editor window.



6. Navigate to the **+ CREATE TABLE** button to open the **Create table** pane.



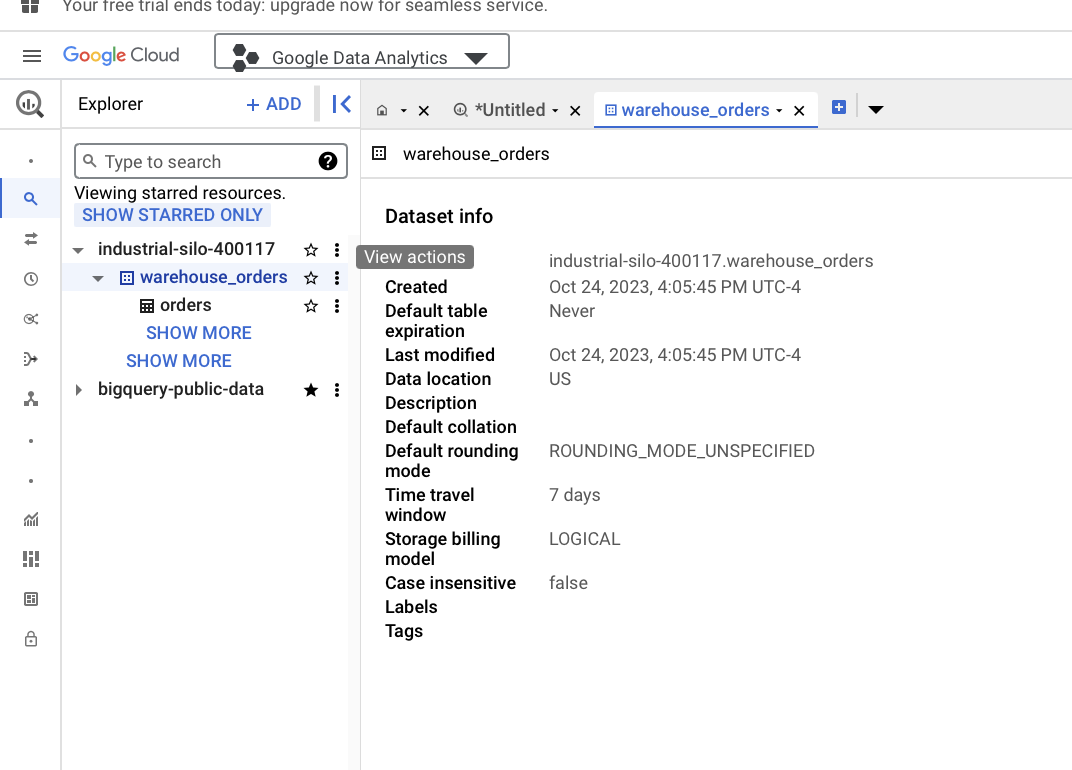
7. From the **Create table from dropdown list**, choose where the data will be coming from.

* Select **Upload**.
* Select the **Browse** button to select the **Warehouse Orders - Orders.csv** file you downloaded.
* Choose **CSV** from the file format dropdown list (file type may automatically be detected).

8. In the **Table** text box, enter **orders** if you plan to follow along with the video.

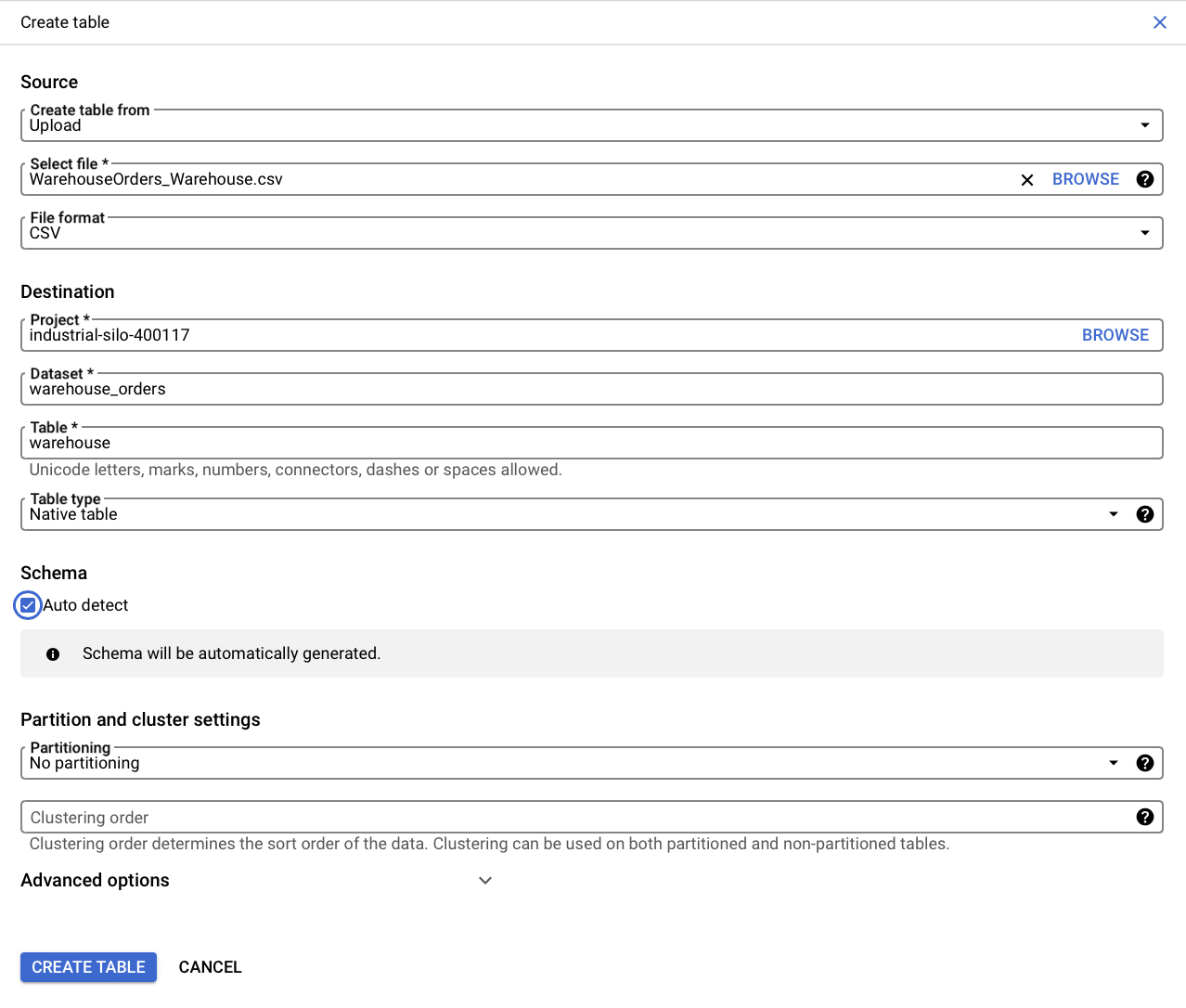
9. Below **Schema**, select the **Auto detect** checkbox.

10. Select **CREATE TABLE**. You will now find the **orders** table below your **warehouse\_orders** dataset in your project.



11. Select the **warehouse\_orders** dataset again.

12. Navigate to the **+ CREATE TABLE** button to open the **Create table pane** again.



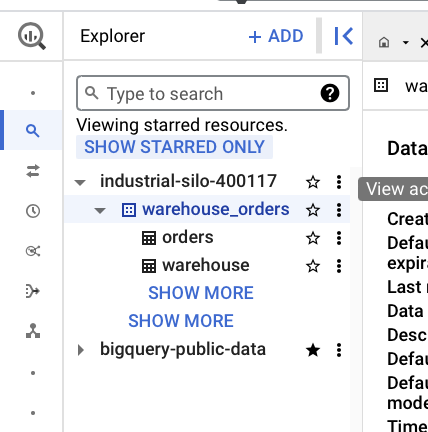
13. For the Create table from selection, choose where the data will be coming from.

* Select **Upload**.
* Select **Browse** to select the **Warehouse Orders - Warehouse.csv** file you downloaded.
* Choose **CSV** from the **File format** dropdown list.

14. For Table name, enter **warehouse** if you plan to follow along with the video.

15. For Schema, select the **Auto detect** checkbox.

16. Select the **CREATE TABLE** button. You will now find the **warehouse** table under your **warehouse\_orders** dataset in your project.



17. In the Explorer pane, select the **orders** table and then select the **Preview** tab to verify that you retrieve the first 50 rows of data. You may have to scroll down the page to view all 50 rows (there are 9999 pages total).



18. Select the **warehouse** table and select the **Preview** tab to verify that you can view 10 rows of data. If both your data previews match, you are ready to move along to the [COUNT and COUNT DISTINCT](https://www.coursera.org/teach/analyze-data/authoringBranch~4f7eJX9NEe6L0QpSVaIh4Q/content/item/project/3K5St) activity.

### 1.

Question 1

Fill in the blank: To combine rows from two or more tables based on a \_\_\_\_\_ column, data professionals use the SQL **JOIN** clause.

1 point

unique

related

foreign

dissimilar

### 2.

Question 2

A data analyst wants to retrieve only records from a database that have matching values in two different tables. Which **JOIN** function should they use?

1 point

**LEFT JOIN**

**RIGHT JOIN**

**INNER JOIN**

**OUTER JOIN**

### 3.

Question 3

You write a SQL query that will count values in a specified range. Which function should you include in your query to only count each value once, even if it appears multiple times?

1 point

**COUNT RANGE**

**COUNT DISTINCT**

**COUNT VALUES**

**COUNT**

### 4.

Question 4

Fill in the blank: Aliasing involves \_\_\_\_\_ naming a table or column to make a query easier to read and write.

1 point

perpetually

privately

permanently

temporarily

**Step-by-Step: Queries within queries**

This reading outlines the steps the instructor performs in the next video, [Queries within queries](https://www.coursera.org/learn/analyze-data/lecture/yVQoh/queries-within-queries). In the video, the instructor introduces subqueries, another type of SQL query, and demonstrates how to pull data from a large dataset then incorporate it as a subquery.

Keep this step-by-step guide open as you watch the video. It can serve as a helpful reference tool 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 will need**

In order to follow along with the instructor, you will need to log in to your BigQuery account and access the public dataset **new\_york\_citibike**. If you haven’t already located this data, follow the instructions in the [Prepare to use the bike sharing dataset in BigQuery](https://www.coursera.org/learn/analyze-data/supplement/vYBY0/optional-prepare-to-use-the-bike-sharing-dataset-in-bigquery) reading to access the bike sharing dataset.

**Note:** If you would like to follow along with the video, make sure to use the **citibike\_stations** table instead of the **citibike\_trips** table, especially when writing your queries.

**Example 1: Create a subquery with a SELECT statement**

In this query, you will compare the number of bikes available at a particular station to the average number of bikes available.

1. Create a new query. Return to your Editor tab or select **Compose new query** to open a query window.
2. In line 1 of the query window: Enter **SELECT** to begin your query. Then press **Enter** (Windows) or **Return** (Mac).
3. Press **Tab** and then in line 2, enter **AVG(num\_bikes\_available)**. This creates a column in the query results of the average number of bikes available.
4. Press **Enter/Return** to add another line.
5. In line 3: Enter **FROM bigquery-public-data.new\_york.citibike\_stations** to specify which dataset you want the query to pull data from.
6. Go back to line 1. Add a few lines before the **SELECT** statement to create the outer query. The original **SELECT** statement will become the subquery and will begin on line 4. Press **Enter/Return** to add a line before the **SELECT** statement.
7. In the new line 1, enter **SELECT**. Then press **Enter/Return**.
8. In line 2: Press **Tab**. Then enter **station\_id** followed by a comma. Press **Enter/Return**.
9. In line 3: Enter **num\_bikes\_available**. This makes up the query that will pull the number of bikes available with the station IDs.
10. In line 4: Press **Tab**. Add **parentheses** [**()**] around the original SELECT and FROM statements to create the subquery.
11. In line 6: Enter **AS avg\_num\_bike\_available** after the parentheses. This names the column in the Query results containing the average number of bikes available.
12. Press **Enter/Return**.
13. In line 7: Enter **FROM**. Press **Enter/Return**.
14. In line 8: Press **Tab**. Enter **bigquery-public-data.new\_york.citibike\_stations**.
15. Select **RUN**.

If you prefer, copy and paste the following query into the query window instead.

1

2

3

4

5

6

7

8

SELECT

    station\_id,

    num\_bikes\_available

(SELECT

        AVG(num\_bikes\_available)

FROM bigquery-public-data.new\_york.citibike\_stations) AS avg\_num\_bikes\_available

FROM

    bigquery-public-data.new\_york.citibike\_stations

A table containing the number of bikes available and the average number of bikes available at different stations will be generated in the **Query results** window.

**Example 2: Create a subquery using a FROM statement**

Next, calculate the number of rides that have started at each station over time.

1. Create a new query.
2. In line 1, enter **SELECT**, then press **Enter/Return**.
3. In line 2, press **Tab**. Enter **station\_id** followed by a comma, then press **Enter/Return**.
4. In line 3, enter **name** followed by a comma, then press **Enter/Return**.
5. In line 4, enter **number\_of\_rides AS number\_of\_rides\_starting\_at\_station** to tell the query how the table should be labeled. Press **Enter/Return**.
6. In line 5, press **Backspace** to align the cursor with **SELECT**. Enter **FROM** to tell the query where to pull the data from. Then press **Enter/Return**.
7. In line 6, press **Tab**. Enter **(SELECT**, then press **Enter/Return**. The close parenthesis [**)**] will automatically be added to the query.
8. In line 7, press **Tab**. Enter **start\_station\_id** followed by a comma, then press **Enter/Return**.
9. In line 8, enter **COUNT(\*) number\_of\_rides**. Press **Enter/Return**.
10. In line 9, press **Backspace** to align the cursor with **SELECT**. Enter **FROM**, then press **Enter/Return**.
11. In line 10, press **Tab**. Enter **bigquery-public-data.new\_york.citibike\_trips**, then press **Enter/Return**.
12. In line 11, enter **GROUP BY** to tell the query to group the trip data by the station ID. Press **Enter/Return**.
13. In line 12, press **Tab**. Enter **start\_station\_id**. After the close parenthesis, press **Enter/Return**.
14. In line 13, enter **AS station\_num\_trips**, then press **Enter/Return**.
15. In line 14, enter **INNER JOIN**, then press **Enter/Return**.
16. In line 15, enter **bigquery-public-data.new\_york.citibike\_stations ON station\_id=start\_station\_id** to join the subquery with the station ID data. Press **Enter/Return**.
17. In line 16, enter **ORDER BY**, then press **Enter/Return**.
18. In line 17, press **Tab**. Enter **number\_of\_rides DESC** to tell the query to put the data in descending order.
19. Select **RUN**.

If you prefer, copy and paste the following query into the query window instead.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

SELECT

    station\_id,

    name,

    number\_of\_rides AS number\_of\_rides\_starting\_at\_station

FROM

    (SELECT

        start\_station\_id,

        COUNT(\*) number\_of\_rides

    FROM

        bigquery-public-data.new\_york.citibike\_trips

        GROUP BY

            start\_station\_id)

            AS station\_num\_trips

            INNER JOIN

            bigquery-public-data.new\_york.citibike\_stations ON station\_id=start\_station\_id

            ORDER BY

                number\_of\_rides DESC

The number of rides started at each station will be listed in the **Query results**.

**Example 3: Create a subquery using a WHERE statement**

Finally, write a query that will allow you to generate a list of stations used by subscribers.

1. Create a new query.
2. In line 1, enter **SELECT**, then press **Enter/Return**.
3. In line 2, press **Tab**. Enter **station\_id** followed by a comma, then press **Enter/Return**.
4. In line 3, enter **name**, then press **Enter/Return**.
5. In line 4, press **Backspace** to remove the indentation. Enter **FROM**, then press **Enter/Return**.
6. In line 5, enter **bigquery-public-data.new\_york.citibike\_stations**. Press **Enter/Return**.
7. In line 6, enter **WHERE**, then press **Enter/Return**.
8. In line 7, press **Tab**. Enter **station\_id IN**, then press **Enter/Return**. This will allow you to specify multiple values.
9. In line 8, begin the subquery with an **open parenthesis** [**(**]. Then press **Enter/Return**.
10. In line 9, press **Tab**. Enter **SELECT**, then press **Enter/Return**.
11. In line 10, press **Tab**. Enter **start\_station\_id**. Press **Enter/Return**.
12. In line 11, press **Backspace** to align the cursor with **SELECT**. Enter **FROM**, then press **Enter/Return**.
13. In line 12, press **Backspace** so the cursor is in line with the open parenthesis.
14. Enter **bigquery-public-data.new\_york.citibike\_trips**. Then press **Enter/Return**.
15. In line 13, press **Tab**. Enter **WHERE**, then press **Enter/Return**.
16. In line 14, press **Tab**. Enter **usertype =** to indicate that you only want subscriber user data in the results. Press **Enter/Return**.
17. In line 15, press **Backspace** to align the cursor with **WHERE**. Enter **'Subscriber'**.
18. Select **RUN** to run the query.

If you prefer, copy and paste the following query into the query window instead.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

SELECT

    station\_id,

    name

FROM

bigquery-public-data.new\_york.citibike\_stations

WHERE

    station\_id IN

    (

        SELECT

            Start\_station\_id

        FROM

    Bigquery-public-data.new\_york.citibike\_trips

    WHERE

        usertype =

    'Subscriber'

)

A list containing the station IDs and the names for stations that fit the designated criteria—stations used by subscribers—will be listed in the **Query results**.

Hey, it's great to have you back.

In this video I'll introduce you to

another SQL query: subqueries.

A subquery is a SQL query that is

nested inside of a larger query.

Have you ever seen one of those nesting doll toys?

They're also known as matryoshka Russian nesting dolls.

Subqueries are a lot like nesting dolls.

No, really. Your larger query can have

a subquery in it and then

that subquery could have a subquery,

and then that subquery can have another subquery.

But when you stack them all together,

they make one query.

With subqueries you can combine different pieces of logic

together. Because the logic of

your outer query relies on the inner query,

you can get more done with a single query.

This means all of the logic is in one place,

which makes it more efficient and easier to read.

The statement containing the subquery can also be

called the outer query or the outer select.

This makes the subquery the inner query or inner select.

The inner query executes first

so that the results can be passed

on to the outer query to use.

Subqueries can get a little

confusing because there's so many layers.

But if you keep in mind that

the innermost query executes first,

it'll be easier to order

your subqueries when you want them to execute.

Subqueries can also be nested inside all sorts of other queries.

Usually you'll find subqueries

nested in FROM or WHERE clauses.

Let's try out some common subqueries.

We'll start with a subquery in a SELECT statement

using the bike-sharing data from an earlier example.

For the first statement,

let's say we want to compare the number of bikes available

at a station to the average number of bikes available.

We're going to use this query to pull

the average number of bikes available.

Then we're going to incorporate it as a subquery.

Play video starting at :2:2 and follow transcript2:02

Now let's build our outer SELECT query.

We want to select the station ID

and the number of bikes available.

Then we'll put the SELECT query

that's pulling the average number of

bikes inside that outer query by using parentheses.

We'll also build FROM into the subquery before closing

it with another parenthesis

and completing the outer query.

Play video starting at :2:31 and follow transcript2:31

The end of the outer join query

has AS to show what we want to call

this column and a final FROM

statement to indicate which table we're referring to.

Play video starting at :2:45 and follow transcript2:45

Now let's run it. And there! We've got a table with

both the number of bikes available and

the average number of bikes

available at different stations.

It's really common to see subqueries

nested in FROM and WHERE statements.

So let's try those next.

We could use a FROM statement to calculate the number of

rides that have started at each station over time.

We'll start with our outer query

and input SELECT station\_id,

name, and number\_ of\_ rides.

We'll use AS to tell it how we want the table labeled,

and FROM to tell it where we're pulling data from.

But before we finish that query, we'll add a subquery.

We'll put our parenthesis here and

then SELECT the start\_station\_id.

Then we can tell it to COUNT the number\_of \_rides

FROM the trip data and group it by the start\_station\_id.

Play video starting at :3:50 and follow transcript3:50

After that, we'll close the subquery with

a parenthesis so that we can

continue building the outer query.

We'll use AS again and then use

INNER JOIN and ON to join it with the station ID data.

Play video starting at :4:12 and follow transcript4:12

Finally we'll tell it to put it in descending order.

Let's see what happens when we run that.

We now have the number of rides started at each station.

Play video starting at :4:27 and follow transcript4:27

One last example.

Let's use a WHERE statement.

The bike-sharing company has two kinds of users:

subscribers and one-time customers.

Let's say we wanted a list of stations subscribers used.

As always, we start with the outer query.

SELECT the station\_ id and

name FROM the public dataset we're using.

This time we'll use a WHERE statement.

We'll also use IN so that we can specify

multiple values and this WHERE statement.

Then we'll put our subquery in the parenthesis.

We'll add SELECT, FROM, and WHERE again.

But this time we'll tell it that we only

want data on specific customers.

Play video starting at :5:21 and follow transcript5:21

It's good to note that you can use

comparison operators in subqueries,

even multiple row operators like IN, ANY, or ALL.

In this case we'll use equals to indicate

that we only want the subscriber user data.

Now let's run the query

and we've got the station id and names

for stations that fit our criteria.

That's subqueries in action.

Subqueries can be challenging.

There's a lot of layers to think through and you

might find yourself running

into errors when you practice.

That's totally okay.

Having to go through that challenge means you're growing.

If everything was easy,

we wouldn't find new ways to grow.

For me it's all about

how much work and

how much time I need to put in to do it.

Give yourself time to practice this new concept.

Coming up you'll get a chance to use subqueries to

aggregate data or you

can move on to the weekly challenge.

You'll take everything you've learned, using VLOOKUP,

different JOINS, and subqueries

and apply it to this upcoming assessment.

We've been doing a lot of complex work.

If you want to take a moment to review

these videos before moving on, feel free.

Once you've finished a challenge,

I'll see you again for

our next big learning adventure. See you soon.

# Step-by-Step: Use subqueries to aggregate data

This reading provides you with the steps the instructor performs in the following video, [Use subqueries to aggregate data](https://www.coursera.org/learn/analyze-data/lecture/JjPZ5/using-subqueries-to-aggregate-data). The video teaches you how to aggregate or combine data using subqueries in SQL.

Keep this step-by-step guide open as you watch the video. It can serve as a helpful reference tool 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**

In order to follow along with the instructor, you will need the Warehouse dataset uploaded into your project space. If you haven’t already uploaded this data, follow the instructions in the [Upload the warehouse dataset to BigQuery](https://www.coursera.org/learn/analyze-data/supplement/HuXCc/optional-upload-the-warehouse-dataset-to-bigquery) reading.

## Example: Alias the tables

As a refresher, aliasing is when you temporarily name a table or column in your query to make it easier to read and write. To alias the Warehouse and Orders tables and join the tables, follow these steps:

1. In row 3, enter **FROM warehouse\_orders.Warehouse Warehouse**
2. In row 4, enter **LEFT JOIN warehouse\_orders.Orders Orders**
3. In row 5, enter **ON Orders.warehouse\_id = Warehouse.warehouse\_id**

## Example: Organize your new table

Use the **GROUP BY** clause in SQL to group rows that have the same values in specified columns into aggregated data, such as sum, count, average, maximum, or minimum, based on the values in another column. This operation is particularly useful in databases where there is a need to analyze data based on certain criteria.

1. In row 6, enter **GROUP BY**
2. In row 7, enter **Warehouse.warehouse\_id,**
3. In row 8, enter **warehouse\_name**

## Example: Build subquery logic

Use the **CONCAT** and **COUNT** clauses in SQL for tasks including counting distinct combinations, counting rows based on concatenated values, concatenating and counting subquery results, and counting and concatenating in the same query.

1. In row 1, enter **SELECT**
2. In row 2, enter **Warehouse.warehouse\_id,**
3. In row 3, enter **CONCAT (Warehouse.state, ':', Warehouse.warehouse\_alias) AS warehouse\_name,**
4. In row 4, **COUNT(Orders.order\_id) AS number\_of\_orders,**
5. In row 5, enter **(SELECT**
6. In row 6, enter **COUNT(\*)**
7. In row 7, enter **FROM warehouse\_orders.Orders Orders)**
8. In row 8, enter **AS total\_orders,**

## Example: Create categories using CASE

Use the **CASE** statement in SQL to create categories or group data based on specific conditions. This is valuable when dealing with numerical or textual data that needs to be segmented into different groups or categories for analysis, reporting, or visualization purposes.

1. In row 9, enter **CASE**
2. In row 10, enter **WHEN COUNT(Orders.order\_id)/(SELECT COUNT(\*) FROM warehouse\_orders.Orders Orders) <= 0.20**
3. In row 11, enter **THEN "Fulfilled 0-20% of Orders"**
4. In row 12, enter **WHEN COUNT(Orders.order\_id)/(SELECT COUNT(\*) FROM warehouse\_orders.Orders Orders) > 0.20**
5. In row 13, enter **AND COUNT(Orders.order\_id)/(SELECT COUNT(\*) FROM warehouse\_orders.Orders Orders) <= 0.60**
6. In row 14, enter **THEN "Fulfilled 21-60% of Orders"**
7. In row 15, enter **ELSE "Fulfilled more than 60% of Orders"**
8. In row 16, enter **END AS fulfillment\_summary**

**Note:** At this point, your lines of code from “Alias the tables” and “Organize your new table” should be lines 17-22.

## Example: Filter using HAVING

Use the **HAVING** clause in SQL in combination with the **GROUP BY** clause to filter the results of aggregate functions in a query. While the **WHERE** clause filters individual rows before they are grouped, the **HAVING** clause filters groups of rows after they have been grouped. To filter out the warehouses that are currently being built,

1. In row 23, enter **HAVING**
2. In row 24, enter **COUNT(Orders.order\_id) > 0**

## Example: Run the new query

It’s important to run the new queries that you write to test that they are working correctly.

1. Select the **Run** button
2. Now, you can identify what percent of our company’s total orders are being fulfilled by each warehouse

Hey, there. We've got some experience nesting subqueries

into our SQL statements to

perform more complicated queries.

Now it's time to talk about how to

aggregate data with subqueries.

Coming up, we'll learn about

some new subquery statements and

use them to aggregate data.

The query we're going to build

in this video is pretty advanced.

It's going to be a little complicated,

but I know you've got what it takes.

Let's get started.

We've used functions like

WHERE to filter our data before,

but the WHERE function can't be

used with aggregate functions.

For example, you can use WHERE

on a statement and follow it with GROUP BY.

But when you want to use GROUP BY first and then

use WHERE on that output,

you'll need a different function.

This is where HAVING comes in.

HAVING basically allows you to add a filter to your query

instead of the underlying table

when you're working with aggregate functions.

That way it only returns

records that meet your specific conditions.

Similarly, CASE returns records with your conditions by

allowing you to include if/then statements in your query.

Let's try to aggregate our data with

subqueries and test out these new functions.

Let's say we're working with a company that

makes socks that we talked about earlier.

We've been asked to calculate what percentage of

the orders are fulfilled by each warehouse.

Basically, we're interested in knowing which

warehouses are delivering the most orders.

We've seen these tables before,

but as a quick refresher,

here's the Orders table.

Play video starting at :1:38 and follow transcript1:38

You can see the columns here: order\_id,

customer\_id, warehouse\_id, order\_date, and ship\_date.

If we pull up the warehouse table,

we can check out its columns.

We have the warehouse\_id, warehouse\_alias,

the maximum\_capacity, the total number of employees,

and the state where the warehouse is located.

Before we start building

the rest of our query, we'll want to alias our table names.

As a reminder, aliasing is when you temporarily name

a table or column in

your query to make it easier to read and write.

This example query is a little bit more

complicated than the ones we've seen before.

Aliasing will help save us some time.

We'll start by aliasing

the Warehouse table in our FROM statement.

The FROM statement in this query is near the end,

but we'll build this first so that we

can use the alias everywhere else.

We'll simplify it to just Warehouse

for the rest of this query.

We know that we're going to JOIN these tables together.

Let's add that while we're working

on this part of the query anyway.

We're using a LEFT JOIN here because we want

all the information from our Warehouse data,

even if it doesn't show up in the Orders table.

Then we'll alias the Orders table

as part of this statement.

Play video starting at :3: and follow transcript3:00

Now both of our tables have temporary names we can use.

We've already finished a JOIN statement.

But before we can build the beginning of this query,

let's go ahead and add

our GROUP BY statement after this JOIN.

We'll group these by the warehouse\_id and name.

Now we'll go back to the beginning of the query.

We'll select the warehouse \_id.

Then we'll use CONCAT to combine

the strings with the warehouse's state and

alias AS the warehouse name.

Then we'll use COUNT to get

the number of orders per warehouse.

Play video starting at :3:49 and follow transcript3:49

Next, we'll build in a subquery to pull

the total number of orders placed across all warehouses.

We'll input SELECT again and

then write the subquery in parentheses.

We'll put an asterisk after COUNT to indicate that

we want to include everything from the Orders table.

Finally, we'll close out the subquery and use

AS to name this column total\_orders.

Play video starting at :4:19 and follow transcript4:19

Now that our subquery logic is complete,

we can use a CASE statement to create categories for

our warehouses based on

how many orders they will fulfill.

We'll represent these as percentages.

You should notice COUNT in the statement a few times.

We'll start by saying WHEN the number of orders FROM

our Orders table is less than or equal to 0.2,

THEN the table will say

"Fulfilled 0-20% of Orders."

Then we'll use WHEN

again to indicate that

when the number of orders is greater than

0.2 and less than or equal to 0.6,

Play video starting at :5:9 and follow transcript5:09

it'll say, "fulfilled 21-60% of Orders."

After that, we can use ELSE to have

everything that doesn't meet

the criteria of our CASE statement

say, "Fulfilled more than 60% of Orders."

Then we'll use END AS to

name this column fulfillment\_summary.

That brings us back to the portion of

the query we've already written.

But we're going to add

a HAVING statement at the very end of this query.

Our Warehouse table has

warehouses that are currently being built, and

we want to filter those out since

they aren't fulfilling orders yet.

We can use HAVING to only include

warehouses that have at least one order.

Play video starting at :5:56 and follow transcript5:56

Now, before we execute this query,

let's take a moment to look at the whole thing.

We have an outer SELECT, a COUNT subquery,

a CASE statement, a JOIN and HAVING,

all wrapped into one query.

We've built a really complex query.

So let's run it to see the new table.

There. Now we can easily identify what percent of

our company's total orders are

being fulfilled by each warehouse.

These warehouses met our criteria.

And we can see here in the fulfillment\_summary column

the percentage categories we

outlined in our CASE statement.

Obviously, since we included a HAVING statement

to specify only warehouses with at least one order,

there aren't any warehouses currently

under construction in this table.

That really complicated query we wrote

created this specific table of data

we can use to easily compare

how these warehouses are performing.

There you go. That's a quick taste of what it's

like to work with subqueries and data aggregation.

Clauses like HAVING and CASE paired with

subqueries will help you build

more and more complex queries,

which lets you do more and more complex things in SQL.

# SQL functions and subqueries: A functional friendship

As you’ve been learning, **SQL functions** are tools built into SQL to facilitate performing calculations. For example, you could use the **AVG()** function to calculate the average salary of employees in a table so management knows what to budget for next year. Another example might be using the **COUNT()** function to count the number of orders in a table to track daily order inventory.

A **subquery**, also called an inner or nested query, is a SQL query that is nested inside a larger query. Going back to the previous example, you could add a subquery to your average calculation to identify the names of employees who earn more or less than the average salary to include that information in performance reviews. Subqueries allow more complex questions to be answered in a single query, making data retrieval more efficient. In this reading, you will learn about SQL functions and how they might be used with subqueries.

## How do SQL functions function?

SQL functions help make data aggregation possible. As a refresher, data aggregation is the process of gathering data from multiple sources in order to combine it into a single, summarized collection. Take a moment to review some of these functions to better understand how to run these queries:

* **HAVING:** The **HAVING** clause filters the results of a SQL query based on conditions applied after the grouping. Check out [W3School’s HAVING overview](http://www-db.deis.unibo.it/courses/TW/DOCS/w3schools/sql/sql_having.asp.html) for a tutorial on this clause
* **CASE: CASE** provides conditional logic in SQL queries, similar to an 'if-else' structure in programming languages. The [W3School’s CASE overview](https://www.w3schools.com/sql/sql_case.asp) explores the use of the **CASE** statement and how it works.
* **IF:** IFperforms a simple conditional test and returns a value depending on the outcome. Review [W3School’s IF overview](https://www.w3schools.com/sql/func_mysql_if.asp) for a tutorial of the **IF** function and examples that you can practice with.
* **COUNT: COUNT** performs a simple conditional test and returns a value depending on the outcome. Though it seems simple, the **COUNT** function is just as important as all the rest. The [W3School’s COUNT overview](http://www-db.deis.unibo.it/courses/TW/DOCS/w3schools/sql/sql_func_count.asp.html) provides a tutorial and examples.

## Subqueries

Subqueries can make projects easier and more efficient by allowing complex operations to be performed in a single query, reducing the need for multiple trips to the database. Subqueries also make your code more readable and maintainable. Take the employee salary example mentioned before.:The original query was used to find the average employee salary. By adding a subquery, you can learn this plus identify employees who earn more than the average—all in a single query.

Usually, you will find subqueries nested in the **SELECT**, **FROM**, and/or **WHERE** clauses. There is no general syntax for subqueries, but the syntax for a basic subquery follows a similar pattern:

1

2

3

4

5

6

7

SELECT account\_table.\*

    FROM (

        SELECT \*

        FROM transaction.sf\_model\_feature\_2014\_01

        WHERE day\_of\_week = 'Friday'

        ) account\_table

    WHERE account\_table.availability = 'YES'

Basically, there’s another **SELECT** clause inside the first **SELECT** clause. The second **SELECT** clause marks the start of the subquery in this statement. There are many different ways in which you can use subqueries, but there are a few rules to follow:

* Subqueries must be enclosed within parentheses.
* A subquery can have one or more columns specified in the **SELECT** clause.
* Subqueries that return more than one row can only be used with multiple value operators, such as the **IN** operator which allows you to specify multiple values in a **WHERE** clause.
* A subquery can’t be nested in a **SET** command. The **SET** command is used with **UPDATE** to specify which columns (and values) are to be updated in a table.

### **Additional resources**

The following resources offer more guidance into subqueries and their usage:

* [**SQL subqueries**](https://www.w3resource.com/sql/subqueries/understanding-sql-subqueries.php):This detailed introduction includes the definition of a subquery, its purpose in SQL, when and how to use it, and what the results will be.
* [**Writing subqueries in SQL**](https://mode.com/sql-tutorial/sql-sub-queries/): Explore the basics of subqueries in this interactive tutorial, including examples and practice problems that you can work through.

As you continue to learn more about using SQL, functions, and subqueries, you will realize how much time you can truly save when memorizing these tips and tricks.

Hi, I'm Justin.

I work here at Google in the Google Cloud space.

I lead a small team of data analysts

who answer business problems for our executive team.

The first thing I would tell you about my journey

to analytics was, it was not direct.

I came to Google three years ago and

I have been doing data analytics

and I've really been enjoying that role.

It's tied together this through line of excitement about

data and answering questions that have an impact.

Your career path is not always straightforward.

Maybe data analytics won't be my final destination.

But what I would say is just

keep changing little by little,

figure out what's exciting about your role right now,

in my case, it was,

I loved avoiding politics

and coalition-building and really just

bringing better facts and

better insights to really motivate decisions.

So figure out what you like about your current role,

your current job, and then,

figure out what different role

could you take that would build on that,

but maybe get you more of what you like.

Be curious. The number one skill

is really just asking why and then

going and trying to answer that question. It will lead you

down along whether it's

Wikipedia to understand this model

there is someone usually saying,

"Why did they use that model?"

You go and look up that model

and sort of follow that thread.

Or there's so many great resources

for different languages,

if you want to understand SQL.

There's so many great tools,

but I guess the number one thing is

just sort of follow your curiosity.

When I'm reviewing resumes,

the first thing obviously I'm looking

for is those core skills,

the ability to analyze data,

demonstrate it, experience with

some of the tooling we use.

But I'm also looking for real passion

in answering questions.

Example is where someone's really dug in and tried to

understand the why and they just kept asking,

"Why is this happening, why is

this happening?" and really dig in.

### 1.

Question 1

Which of the following queries contain subqueries? Select all that apply.

1 point

1

2

3

4

5

SELECT employee\_id

FROM employees

WHERE department\_id IN (SELECT department\_id

FROM departments

WHERE location\_id = 1000)

1

2

3

4

5

6

7

SELECT product\_name,

CASE

WHEN price < 10 THEN 'Low price'

WHEN price >= 10 AND price < 20 THEN 'Medium price'

ELSE 'High price'

END AS price\_category

FROM products

1

2

3

4

SELECT price

FROM sales

WHERE price = (SELECT MAX (salary)

FROM sales)

1

2

3

SELECT call

FROM recordings

ORDER BY call.employee\_id, call.start\_time

### 2.

Question 2

When working with subqueries, which query will execute first?

1 point

Leftmost

Innermost

Rightmost

Outermost

### 3.

Question 3

Which **HAVING** clause indicates to only retrieve products that have been sold more than 100 times?

1 point

**HAVING COUNT(order\_items.product\_id) > 100**

**HAVING COUNT(order\_items.product\_id) < 100**

**HAVING (order\_items.product\_id) > 100**

**HAVING (order\_items.product\_id > 100)**

### 4.

Question 4

Fill in the blank: A data professional uses the SQL \_\_\_\_\_ statement to return records that meet conditions by including an if/then statement in a query.

1 point

**CASE**

**HAVING**

**WHEN**

**CONCAT**

Coursera Honor Code  [Learn more](https://learner.coursera.help/hc/articles/209818863)

# Glossary terms from module 3

## ****Terms and definitions for Course 5, Module 3****

**Absolute reference:** A reference within a function that is locked so that rows and columns won’t change if the function is copied

**Aggregation:** The process of collecting or gathering many separate pieces into a whole

**Aliasing:** Temporarily naming a table or column in a query to make it easier to read and write

**COUNT DISTINCT:** A SQL function that only returns the distinct values in a specified range

**Data aggregation:** The process of gathering data from multiple sources and combining it into a single, summarized collection

**INNER JOIN :** A SQL function that returns records with matching values in both tables

**JOIN:** A SQL function that is used to combine rows from two or more tables based on a related column

**LEFT JOIN:** A SQL function that will return all the records from the left table and only the matching records from the right table

**LIMIT:** A SQL clause that specifies the maximum number of records returned in a query

**MATCH:** A spreadsheet function used to locate the position of a specific lookup value

**OUTER JOIN:** A SQL function that combines RIGHT and LEFT JOIN to return all matching records in both tables

**RIGHT JOIN:** A SQL function that will return all records from the right table and only the matching records from the left.

**Subquery:** A SQL query that is nested inside a larger query

**VALUE:** A spreadsheet function that converts a text string that represents a number to a numeric value

### 1.

Question 1

While using **VLOOKUP**, you encounter an error because some of your spreadsheet values have leading and trailing spaces. What function should you use to eliminate these spaces?

1 point

**NOSPACE**

**TRIM**

**CUT**

**VALUE**

### 2.

Question 2

Fill in the blank: The spreadsheet function \_\_\_\_\_ can be used to add up the number of times a range of cells contains the value “paid.”

1 point

**COUNT DISTINCT**

**RANGE**

**RETURN**

**COUNT**

### 3.

Question 3

A junior data analyst writes the following formula: **=AVERAGE($C$1:$C$100)**. What are the purposes of the dollar signs (**$**)? Select all that apply.

1 point

Perform the calculation more efficiently.

Ensure rows and columns do not change.

Average the values in cells C1 to C100 regardless of whether the formula is copied.

Create an absolute reference.

### 4.

Question 4

What will this query return?

1

2

3

SELECT \*

FROM Inventory\_table

LEFT JOIN Scrap\_table

1 point

All rows from the inventory table joined together with the scrap table

All records in the inventory table and any matching rows from the scrap table

All records in the scrap table and any matching rows from the inventory table

All records in both the inventory table and the scrap table

### 5.

Question 5

In this spreadsheet, which function will search for the water type of Lake Urmia?

|  |  |  |  |
| --- | --- | --- | --- |
| (n/a) | A | B | C |
| 1 | **Lake** | **Surface area (sq. miles)** | **Water type** |
| 2 | Caspian Sea | 143,000 | Saline |
| 3 | Superior | 31,700 | Freshwater |
| 4 | Victoria | 26,590 | Freshwater |
| 5 | Huron | 23,000 | Freshwater |
| 7 | Tanganyika | 12,600 | Freshwater |
| 8 | Balkhash | 6,300 | Saline |
| 9 | Athabasca | 3,030 | Freshwater |
| 10 | Urmia | 2,320 | Freshwater |

1 point

**=VLOOKUP(“Urmia”, A2:C10, 3, false)**

**=VLOOKUP(Urmia, A2:C10, 2, false)**

**=VLOOKUP(Urmia, A2:B10, false)**

**=VLOOKUP(“Urmia”, B2:C10, 2, false)**

### 6.

Question 6

Fill in the blank: A SQL clause containing **HAVING** adds a \_\_\_\_\_ to a query instead of the underlying table.

1 point

filter

subquery

limit

join

### 7.

Question 7

A junior data analyst in a marketing department works with a spreadsheet containing email click-through data. To calculate the average click-through rate for a campaign, the analyst uses a function to convert the number of clicks to numeric values. What function do they use?

1 point

**EXCHANGE**

**NUM**

**VALUE**

**PROCESS**

### 8.

Question 8

Which query will select all columns from the highways table and alias them to hwys?

1 point

1

2

SELECT \*

FROM highways ALIAS hwys

1

2

SELECT \*

FROM highways NEW hwys

1

2

SELECT \*

FROM highways TO hwys

1

2

SELECT \*

FROM highways AS hwys

Module 4

Hi.

Good to have you back.

Coming up, we're going to be reviewing some familiar concepts and

then using those concepts to explore new ones.

As a data analyst, you'll use key tools and processes over and

over, but you'll also learn new things as you grow in your job.

It could be anything from building a new

kind of analysis to a time-saving shortcut.

When I first got to Google, I relied on just a couple of programs and

tools to access data and do my analysis.

But I soon realized that I wasn't working as efficiently as I wanted to.

Once I got comfortable pulling data and analyzing it using SQL,

it allowed me to be a lot more efficient than before.

And the better I got at SQL and pulling the data from data tables,

the faster I completed my analysis. I was hooked.

Over the next few videos, I will show you some ways to be as efficient as

possible while completing calculations during your analysis.

We'll start by revisiting spreadsheets

where we'll look at formulas for basic calculations.

Then we'll move into conditional formulas that use the IF function to check whether

a condition is met through a calculation.

After that, we'll explore the multifunctional SUMPRODUCT function.

Try saying that five times quickly! SUMPRODUCT adds and

multiplies all in one step, so it's very useful.

Next we'll take another look at pivot tables.

If you've skipped around, and it's your first time learning about them,

you'll get to know all about them.

Pivot tables have tons of uses, including organizing your calculations.

We'll then pivot to SQL, pun intended.

We'll show how queries and calculations go hand in hand in SQL.

We'll also look at temporary tables in SQL, which are helpful for

temporarily storing your data during analysis.

We'll be covering lots of new concepts in these videos, so

feel free to hit the pause button at any time to think through the problem or

steps to try it on your own.

And you can always review the videos as much as you need to.

Play video starting at :2:9 and follow transcript2:09

So to recap, we'll have a little bit of a review, and

then cover some all new concepts, all about calculations.

Are you ready? Good. Me, too.

Hey there. You probably do

a lot of calculations in your daily life.

Maybe it's figuring out how much to tip

someone or balancing your budget.

You might do some of these calculations

in your head or with

paper and pencil or the calculator on your phone.

You might even have shortcuts to use

to make the calculations easier.

You'll perform a lot of

calculations as a data analyst too.

But they'll involve more numbers

in a wider range of calculations.

That's where you'll put your data analyst tools to work.

We'll show you how you could use formulas in

a spreadsheet to complete some of

the most basic calculations.

Formulas are one of the many shortcuts

that data analysts use.

But rest assured, even though they're shortcuts,

they'll still calculate with complete accuracy.

We've covered a lot of these calculations

earlier in the program.

But if you skip that part and want a refresher,

we'll review them here.

These calculations will also be more

advanced than the ones we've covered so far.

But they'll also be closer to

what you might use on the job.

We'll be using Google Sheets in this video,

but you can also use Excel.

The steps might look a little different in Excel,

but the outcomes will be the same.

Let's try out some calculations with

sales data from a discount store chain.

We'll look at data for one of the stores in the chain.

Our objective: use the existing sales data

to find any trends.

This is a great way to see a lot of the ways

formulas can be useful in your analysis.

We'll start by finding annual sales

over the years 2011-2020.

The data is already organized in

columns by month and in rows by year.

But we don't have the total sales for each year yet.

We can use a sum function to help us figure that out.

We'll add the sales from 2011 first.

We'll add a heading for the annual sales column,

then we can type our sum function and a formula.

All formulas begin with an equal sign.

We'll type that first,

followed by sum and then an open parenthesis.

After the open parenthesis,

we need to tell the formula which cells are being added.

In this case, we need data from

the whole row which begins in cell B2.

B2 is a cell reference we'll use.

Instead of typing each cell one by one,

we can put them in the formula quickly by

selecting cell B2 and dragging

the fill handle across the row to

the last cell with sales data, M2.

Play video starting at :2:28 and follow transcript2:28

Now we'll complete the formula by

closing the parentheses and pressing Enter.

Play video starting at :2:37 and follow transcript2:37

Just like that, we've calculated

the total sales for 2011.

Here's another shortcut we

worked with in an earlier video.

The fill handle is

the tiny box in the corner of each sale.

You can use it for lots of things

like selecting multiple cells for

a formula or continuing a pattern across several cells,

the fill handle definitely qualifies as a shortcut.

We can use the formula we created to calculate

the total sales for the other years in the dataset.

All we have to do is drag

the fill handle down the other cells in

the annual sales column and we'll have

total sales data for

the rest of the years in the dataset.

Let's say, we also need to find the growth

in annual sales from year to year.

This would be a good time to think through

the problem before we try to solve it.

Do we have the data we need to solve this? Not yet.

Thinking backwards like this helps us

plan out the steps to move forward.

The first step we'll need to do is

calculate the total sales per year.

Then we'll measure the rate of change between years.

We'll start by labeling a new column.

Play video starting at :3:55 and follow transcript3:55

In this case, we won't need to

use a function or parentheses,

since we're only using data from two cells.

We can just use the name of those cells,

we'll type an equals sign and then click in "Cell N3",

which automatically populates that sale in the formula.

Next, we'll add a minus sign

to the formula because we're

subtracting to find the difference

between two consecutive years.

Clicking in "Cell N2" gives us the total from 2011,

which we can then subtract from the total from 2012.

Then we hit Enter and get

our sales growth from 2011-2012.

We're definitely getting some useful data here.

Let's keep going. We can also use

our sales growth to find

the growth rate between the two years.

We'll show this as a percentage.

We'll head our column with the percent sign and growth.

To do this, we'll divide the total in cell O3

by the annual sales from 2011 in cell N2.

A slash is a symbol that

a formula recognizes as division,

so we'll place that between

the two cell references and presto,

there's the growth rate.

Growth rates are usually shown as percentages,

which can be easier than

a decimal to read and understand.

Let's change this number to a percentage.

Time for another shortcut.

All we have to do is click the percent style

button and our growth rate will become a percentage.

We can select the cells for both the total growth

and the growth rate to populate

the rest of the two columns.

Play video starting at :5:37 and follow transcript5:37

We have some negative numbers,

but that just means that there was

negative growth from one year to the next.

We've got just a few more things to

calculate for our stakeholders.

Next step is finding the average sales.

We want to compare sales

between months to learn if there's a trend.

We'll add this in a row instead of a column.

This will line up our averages under each month.

Play video starting at :6:9 and follow transcript6:09

To find our averages,

will calculate the total and then divide

that total by the number of values added to get it.

We can do this by using the average function.

Play video starting at :6:24 and follow transcript6:24

Between our parentheses will

select the cells that contain

the sales data for January, B2 through B11.

Play video starting at :6:38 and follow transcript6:38

We'll duplicate that formula across

the row through December to look for trends.

Play video starting at :6:50 and follow transcript6:50

Right away, we know that summer months and

December have the highest average sales.

Play video starting at :6:59 and follow transcript6:59

Since our stakeholders will want to

understand our findings quickly and easily,

we'll add a little visualization to

the data with conditional formatting.

You'll learn more about

data visualizations like conditional formatting soon.

But here's a sneak peek.

Conditional Formatting is a spreadsheet tool that changes

how cells appear when values meet specific conditions.

Let's apply conditional formatting to

the cells with the average sales by month.

Play video starting at :7:41 and follow transcript7:41

We'll use a color scale to show the range of averages.

Well, the lowest monthly average remaining as

white and we'll apply

shades of green to the rest of the values.

Play video starting at :8:1 and follow transcript8:01

The brighter the green,

the higher the average.

Now, when we share our analysis with our stakeholders,

they will be able to tell right away

which months have the highest average sales.

Just a couple more steps to complete our analysis.

Now we need to find the minimum and

maximum for average monthly sales.

With the dataset this small,

it might be easy to find

the minimum and maximum values without a formula,

but it's still good practice to use one.

Not to mention, using a formula

helps prevent human error,

will again rely on formulas with

Functions to do these calculations,

we'll start with the lowest monthly average.

Play video starting at :8:51 and follow transcript8:51

Our function here is MIN,

followed by the cells with

the average month B12 through M12.

Play video starting at :9:3 and follow transcript9:03

After we press Enter,

the lowest monthly average is calculated.

We can repeat the same steps to

find the highest monthly average,

Play video starting at :9:17 and follow transcript9:17

in this formula will use the same data,

but we'll replace MIN with MAX for maximum.

Play video starting at :9:28 and follow transcript9:28

For this store location,

sales are strongest in December and weakest in January.

We could share these findings with

stakeholders if they've met our objectives.

If they haven't, we might need to

continue with our analysis.

Either way, I hope you've learned how

spreadsheet formulas can be

valuable tools when doing calculations.

Coming up, we'll check out more formulas. See you soon.

Welcome back!

One of the first calculations most kids learn how to do is counting.

Soon after, they learn adding, and that doesn't go away.

No matter what age we are, we're always counting or adding something,

whether it's change at the grocery store or measurements in a recipe.

Data analysts do a lot of counting and adding too.

And with the amount of data you'll come across as a data analyst,

you'll be grateful to have functions that can do the counting and adding for you.

So let's learn how these functions COUNTIF and

SUMIF can help you do calculations for your analysis more easily and accurately.

We'll start with the COUNTIF function.

You might remember COUNTIF from some of the earlier videos about data cleaning.

COUNTIF returns the number of cells that match a specified value.

Earlier, we showed how COUNTIF can be used to find and count errors in a data set.

Play video starting at ::55 and follow transcript0:55

Here we'll only be counting.

Just a reminder though, while we won't be actively searching for

errors in this video, you'll still want to watch out for

any data that doesn't look right when doing your own analysis.

As a data analyst, you'll look for and fix errors every step of the way.

Play video starting at :1:13 and follow transcript1:13

For this example,

we'll look at a sample of data from an online kitchen supplies retailer.

Play video starting at :1:19 and follow transcript1:19

Our stakeholders have asked us to answer a few questions about the data to understand

more about customer transactions, including the revenue they're bringing in.

We've added the questions we need to answer to the spreadsheet.

Play video starting at :1:34 and follow transcript1:34

We'll set up a simple summary table,

which is a table used to summarize statistical information about data.

We'll use the questions to create the attributes for our table columns:

count, revenue total, and average revenue per transaction.

Play video starting at :1:52 and follow transcript1:52

Each of our questions ask about transactions with one item or transactions

with more than one item, so those will be the observations for our rows.

Play video starting at :2:6 and follow transcript2:06

We'll make Quantity the heading for our observations.

Play video starting at :2:14 and follow transcript2:14

We'll also add borders to make the summary table nice and clear.

Play video starting at :2:22 and follow transcript2:22

The first question asks, How many transactions include exactly one item?

To answer this, we'll add a formula using the COUNTIF function in cell G11.

Play video starting at :2:33 and follow transcript2:33

We'll begin with an equal sign, COUNTIF, and an open parenthesis.

Play video starting at :2:40 and follow transcript2:40

Column B has data about quantity.

So we'll select cells B3 through B50, followed by a comma.

Play video starting at :2:53 and follow transcript2:53

Next, we need to tell the formula the value that we're looking for

in the cells we've selected.

We want to tell the data to count the number of transactions if they equal 1.

In this case, between quotation marks, we'll type an equal sign and

the number 1 because that's the exact value we need to count.

When we add a closed parenthesis and press enter, we get the total count for

transactions with only one item, which is 25.

We can follow the same steps to count values greater than one.

Play video starting at :3:40 and follow transcript3:40

But this time, because we only want values greater than 1,

we'll type a greater than sign in our formula inside of an equals sign.

Play video starting at :3:48 and follow transcript3:48

Getting this information helps us compare the data about quantity.

Play video starting at :3:54 and follow transcript3:54

Okay, now we need to find out how much total revenue each transaction type

brought in.

Since the data isn't organized by quantity,

we'll use the SUMIF function to help us add the revenue for

transactions with one item and with one more item separately.

SUMIF is a function that adds numeric data based on one condition.

Building a formula with SUMIF is a bit different than one with COUNTIF.

They both start the same way with an equal sign and the function, but

a SUMIF formula contains the range of cells to be evaluated by your criteria,

and the criteria.

In other words,

SUMIF has a list of cells to check based on the criteria you set in the formula.

Then the range where we want to add the numbers is placed in the formula if that

range is different from the range being evaluated.

There's commas between each of these parts.

Adding a space after each comma is optional.

So let's try this.

In cell H11, we'll type our formula.

Play video starting at :5:1 and follow transcript5:01

The range to be evaluated is in column B, so we'll select those cells.

Play video starting at :5:14 and follow transcript5:14

The condition we want the data to meet is for the values in

the column to be equal to one.

So we'll type a comma and then inside quotes an equal sign and the number one.

Play video starting at :5:24 and follow transcript5:24

Then we'll select the range to be added based on whether the data from our first

range is equal to one.

This range is in column C, which lists the revenue for each transaction.

Play video starting at :5:46 and follow transcript5:46

So every amount of revenue earned from a transaction with only one item will be

added together.

And there's our total.

Since this is revenue, we'll change the format of the number to currency, so

it shows up as dollars and cents.

Play video starting at :6: and follow transcript6:00

So the transactions with exactly one item earned $1,555.00 in revenue.

Let's see how much the transactions with more than one item earned.

Play video starting at :6:38 and follow transcript6:38

Okay, let's check out the results.

Just like with our COUNTIF examples, the second SUMIF formula will be the same

as the first, except for the condition, which will make it greater than one.

Play video starting at :6:51 and follow transcript6:51

When we run the formula,

we discover that the revenue total is much higher, $4,735.00.

This makes sense,

since the revenue is coming from transactions with more than one item.

Good news.

To complete our objective, we'll do two more quick calculations.

First, we'll find the average revenue per transaction by dividing each total by

its count.

This will show our stakeholders how much of a difference there is

in revenue per transaction between one item and multiple item transactions.

This information could be useful for lots of reasons.

For example, figuring out whether to add a discount on purchases with more than one

item to encourage customers to buy more.

We'll put these calculations in the last column of our summary table.

You might remember that we use a slash in a formula as the operator for

division calculations.

Play video starting at :7:44 and follow transcript7:44

The average revenue for transactions with one item is $62.20.

Play video starting at :7:55 and follow transcript7:55

And the average revenue for transactions with more than one item is $205.87.

And that's it for our analysis.

Our summary table now gives the stakeholders and

team members a snapshot of the analysis that's easy to understand.

Our COUNTIF and SUMIF functions played a big role here.

Using these functions to complete calculations,

especially in large datasets, can help speed up your analysis.

They can also make counting and adding a little more interesting.

Nothing wrong with that.

And coming up,

we'll explore more functions to make your calculations run smoothly.

Bye for now.

**Functions with multiple conditions**

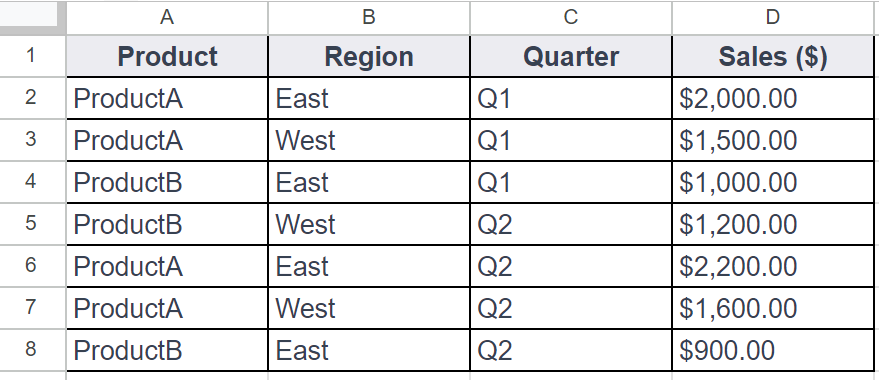
As you’ve been learning, conditional functions and formulas perform calculations according to specific conditions. In addition, functions including **SUMIF** and **COUNTIF** only work in cases where there is one condition. However, if you have more than one condition, you would need to use the **SUMIFS** or the **COUNTIFS** function instead. These functions enable you to perform calculations if you have two or more conditions. In this reading, you will learn more about conditional functions and how to construct functions with multiple conditions by exploring their basic syntax and checking out an example. You will also be able to access resources for similar functions in Excel.

**SUMIF to SUMIFS**

Previously, you learned that the **SUMIF** function adds values in a particular range based on a single condition. The basic syntax is **=SUMIF(range, criterion, sum\_range)**.

The first range is where the function will search for the condition that you have set. The criterion is the condition you are applying and the sum\_range is the range of cells that will be included in the calculation. For example, in an accounting spreadsheet, you could use **SUMIF** to calculate the total expenses for a specific category, like Travel expenses, within a given month.

Or, you could find the total sales for automotive fuel treatment products– in this table, the ProductA is high octane fuel and ProductB is standard octane. Table 1 includes columns for Product, Region, Quarter, and Sales.



You could use **SUMIF** to calculate the total sales for Product A using a formula like this:

**=SUMIF(A2:A8, "ProductA", D2:D8)**

But, you could also build in multiple conditions by using the **SUMIFS** function. **SUMIF** and **SUMIFS** are very similar: They add up values in a range. But SUMIFS can include multiple conditions. This gives you more control over your summing criteria, which, in turn, allows you to perform more complex data analysis easily.

The basic syntax is: **=SUMIFS(sum\_range, criteria\_range1, criterion1, [criteria\_range2, criterion2, ...])**

The square brackets let you know that this is optional. The ellipsis at the end of the statement enables as many repetitions of these parameters as needed. For example, if you wanted to calculate the sum of sales for ProductA in the East district in the first quarter, you could create a **SUMIFS** statement with multiple conditions, like this:

**=SUMIFS(D2:D8, A2:A8, "ProductA", B2:B8, "East", C2:C8, "Q1")**

In this example, B2:B8 is the second criterion\_range and East is the second condition. The third criterion\_range is C2:C8 and the third condition is Q1. As long as you follow the basic syntax, you can add up to 127 conditions to a **SUMIFS** statement!

**COUNTIF to COUNTIFS**

Just like the **SUMIFS** function, **COUNTIFS** allows you to create a **COUNTIF** function with multiple conditions. The definition for **COUNTIF** is a function that counts the number of cells in a range that meet a single condition. For example, using **COUNTIF** to track the number of days an temporary employee was absent in an attendance record.

The basic syntax is: **=COUNTIF(range, criterion)**

Just like **SUMIF**, you set the range and then the condition that needs to be met. For example, in Table 1, if you wanted to count the number of transactions for ProductA, you could use a **COUNTIF** function like this:

**=COUNTIF(A2:A8, "ProductA")**

**COUNTIFS** has the same basic syntax as **SUMIFS**: **=COUNTIFS(criteria\_range1, criterion1, [criteria\_range2, criterion2, ...])**

The criteria\_range and criterion are in the same order, and you can add more conditions to the end of the function. So, if you wanted to find the number of sales transactions for ProductA in the East region in the first quarter, you could use **COUNTIFS** to apply those conditions, like this:

**=COUNTIFS(A2:A8, "ProductA", B2:B8, "East", C2:C8, "Q2")**

This enables you to find every instance where both of conditions (East and Q1) are true.

**For more information**

**SUMIFS** and **COUNTIFS** are just two examples of functions with multiple conditions. They help demonstrate how multiple conditions can be built into the basic syntax of a function. There are other functions with multiple conditions that you can use in your data analysis and many resources available online to help you get started:

* [**How to use the Excel IFS function**](https://exceljet.net/excel-functions/excel-ifs-function): This includes an explanation and example of the **IFS** function in Excel. It’s a great reference if you’re interested in learning more about **IFS**. The example is a useful way to understand this function and how it can be used.
* [**VLOOKUP in Excel with multiple criteria**](https://exceljet.net/formula/vlookup-with-multiple-criteria): Similar to the previous resource, this resource goes into more detail about how to use **VLOOKUP** with multiple criteria. Being able to apply **VLOOKUP** with multiple criteria will be a useful skill, so check out this resource for more guidance on how you can start using it on your own spreadsheet data.
* [**INDEX and MATCH in Excel with multiple criteria**](NULL): This resource explains how to use the **INDEX** and **MATCH** functions with multiple criteria. It also includes an example, which demonstrates how these functions work with multiple criteria and actual data.
* [**Using IF with AND, OR, and NOT functions in Excel**](https://support.microsoft.com/en-us/office/using-if-with-and-or-and-not-functions-d895f58c-b36c-419e-b1f2-5c193a236d97): This resource combines IF with AND, OR, and NOT functions to create more complex functions. By combining these functions, you can perform your tasks more efficiently and cover more criteria at once.

Hi, again. Data analysts love

discovering new ways to work on their analysis,

especially when those new ways simplify their work.

I know I'm a big fan of learning

new tricks to complete tricky tasks.

Instead of trying to find a new way to do

something every time I do an analysis,

I try to learn from other people by

asking questions and getting help when I need it.

The people I work with like to use

the phrase, stealing with pride.

All this means is that you should

feel no shame for using a process

in your analysis that you

learned from someone or somewhere else.

Fellow team members, message board posts,

online searches, I've used all of

these resources for ideas. With pride!

Of course, I always cite my sources when I do.

That's a super important step to remember.

The SUMPRODUCT function is

also one of those tricks that analysts

come across either on their own or from another source.

You can also think of it as a shortcut

for doing more complex calculations.

We'll show you how SUMPRODUCT works and

when you might use it to make your work life simpler.

SUMPRODUCT is a function that multiplies

arrays and returns the sum of those products.

Here's what the SUMPRODUCT formula looks like;

equal sign, the SUMPRODUCT followed by

an open parenthesis and arrays

being multiplied and then added together.

Each array is separated by a comma.

An array is kind of like a range in a spreadsheet.

But keep in mind, an array is

a collection of values in cells,

not the cells themselves.

When added to a formula,

the SUMPRODUCT function multiplies each of

the values in two or more arrays together.

For example, each value in the array of cells B3 through

B7 can be multiplied by

its corresponding value in the array of

cells C3 through C7.

B3 times C3, B4 times C4, and so on.

It will then return the sum

of all of those multiplications.

Let's check out an example using

data from a kitchen supplies company.

You might remember this example from

our COUNTIF and SUMIF video.

We've been given some data about a product order,

including the quantity of each product that was

sold in the order and the unit price,

which tells how much one of each product cost.

Our job is to use the data in

these two columns to find out

the total revenue for this order.

That's where SUMPRODUCT comes in.

To find the total revenue,

we need to do both addition and

multiplication calculations.

First off, we need to find the revenue

that each item brought in separately.

If we did this without SUMPRODUCT,

we'd have to multiply each quantity by its unit price:

50 times $1.25, 25 times $5, and so on.

Then we'd have to add all of

those revenue amounts together to get the total revenue.

Fortunately, the SUMPRODUCT function

does all of that for us.

Let's add the label Total Revenue in

cell G5 and then click G6 to input our formula.

We'll then start our formula with

an equal sign and

the function followed by an open parenthesis.

It's good to remind ourselves that the arrays we add to

our formula should always be inside the parentheses.

Next, we'll select cells B3 through

B7 for the first array followed by a comma.

The comma acts as a separator

between the two arrays and the formula.

Then, we'll select cell C3

through C7 for the second array,

followed by a closed parenthesis to complete our formula.

We don't need to include

the brackets in our actual formula.

We included them in the syntax example to

clearly define each array for you.

Then we press Enter to get our total revenue.

Since we're dealing with revenue,

we'll format the number as currency.

We've learned the total revenue is $655.

But that's not the actual profit

from the sales of these kitchen supplies

because we haven't included

the profit margin in our calculations.

The profit margin is

a percentage that indicates how many cents

of profit have been generated for each dollar of sale.

In our dataset, product #

789 has a profit margin of 20 percent,

meaning each product sold earns

a total profit of $0.20 for every dollar.

And just like the calculation for revenue,

we can save time finding

profit margin by using the SUMPRODUCT function.

There's only one difference between the formula for

profit margin and revenue in this spreadsheet.

But it's an important difference.

To start, in cell G7

we type the same first part of the formula.

Then we include the two arrays in the same way as well.

Play video starting at :4:54 and follow transcript4:54

But instead of ending our formula,

we add another comma followed by another array.

This time, we'll select the cells with

a profit margin, D3 through D7.

We'll finish our formula, and our calculation is complete.

Play video starting at :5:15 and follow transcript5:15

The SUMPRODUCT function saved us from having to multiply

each individual revenue amount by

each profit margin percentage,

then add each profit margin amount together.

Using SUMPRODUCT for calculations is

a time-saver and helps you avoid making mistakes.

Definitely a trick worth remembering,

and there's more worth remembering about

calculations coming up next.

### 1.

Question 1

In this spreadsheet, which function will sum only the values from column B that cost more than $40?

|  |  |  |
| --- | --- | --- |
|  | A | B |
| 1 | **Expense** | **Amount** |
| 2 | screwdrivers | $68.00 |
| 3 | hammers | $30.00 |
| 4 | pliers | $55.00 |
| 5 | wrenches | $41.00 |
| 6 | nails | $22.00 |
| 7 | duct tape | $18.00 |
| 8 | saw | $79.00 |
| 9 | files | $45.00 |
| 10 | paint | $90.00 |
| 11 | bolts | $19.00 |
| 12 | brushes | $33.00 |

1 point

=SUMIF(B2:B12,">40")

=SUMIF(B2:B12,>40)

B: =SUMIF(B2:B12,"<40")

=SUMIF(B2:B12,<40)

### 2.

Question 2

Fill in the blank: The spreadsheet function \_\_\_\_\_ returns the number of cells within a range that match a specified value.

1 point

COUNTIF

COUNT DISTINCT

ARRAY

VALUE

### 3.

Question 3

What is an example of an array in a spreadsheet?

1 point

The values in cells B2 through B31

All cells with values greater than 100

Cells D7, E14, and F20

All cells with number values

### 4.

Question 4

Which function will calculate the sum of the products of the corresponding items in the arrays M1:M4 and P1:P4?

1 point

=SUMPRODUCT(M1:M4, P1:P4)

=MULTIPLY(M1:M4, P1:P4)

=PRODUCT(M1:M4, P1:P4)

=ARRAY(M1:M4, P1:P4)

Hey, there.

By now, we've learned a lot about functions and formulas.

They are very helpful tools for your toolbox and great for

finding shortcuts to complete calculations.

But there's another tool out there that does some of the same things in

a spreadsheet: the pivot table.

As a quick reminder,

pivot tables let you view data in multiple ways to find insights and trends.

We've talked before about how pivot tables help with cleaning and

organizing your data including sorting and grouping data.

But pivot tables can also help with calculations.

For example, they're great for quickly calculating sums and averages.

Let's revisit our movie data set to show you how pivot tables and

calculations worked hand in hand.

Earlier, we summarized and organized this data in pivot tables.

We'll do that here too.

But in this case the organization is a bonus to using pivot tables for

calculations.

You can do these steps in Excel as well, though some of the steps might

look different.

If you're using Excel, you can check out the reading after this video for

more detail about pivot tables and Excel spreadsheets.

In this example, your manager ask you to find some trends to help them think

through new movie ideas using revenue calculations.

This spreadsheet has data about movies from several years ago.

So it probably wouldn't be as useful right now.

Still, the steps we take to analyze the data absolutely apply then and now.

So let's get into it.

First, we need to find out how much revenue was generated each year.

A pivot table is a good way to organize this.

So we'll build a pivot table to show this.

In our pivot table, we can also find the average revenue per movie.

We can then check our findings for some possible trends.

We'll start by finding the revenue generated each year.

This gives us the release date for each movie in column B and

the box office revenue in column N.

Instead of reorganizing the table by year and building a formula

to calculate the revenue per year we'll create a pivot table.

Well, at the pivot table in a new sheet keeping the data range

of cell A1 to cell N509.

Play video starting at :2:27 and follow transcript2:27

Adding a new sheet is especially helpful when working in a large dataset like this.

It helps keep our calculations together in one place and

separate from the rest of the data.

We'll rename this sheet revenue to call out where our calculations are,

both for ourselves and for anyone else on our team who might need our analysis.

Now we can build our pivot table, starting with the rows.

We'll sort the rows by release date to find out the revenue for each year.

You might notice this creates a row for every date on which one or

more movies in this dataset was released.

Since we only need the years, we'll right click in one of the cells in

the release date column to create a pivot date group and group by year.

Now we have rows, one for each year in which these movies were released.

Next, we want to work with the values.

Well, at the box office revenue data here.

This populates the columns next to the release dates

with the total box office revenue and each year.

These calculations are automatic because the pivot table is already set

to summarize the data using the sum function.

So no need to change this setting.

There are other functions and the summarized by menu though,

such as min for the minimum amount of revenue each year and

count for the number of movies that generated revenue in each year.

Okay, let's check out what we've got here.

This data shows that 2014 have the highest revenue, while 2016 have the lowest.

This might be useful information, but

finding the average revenue per movie would most likely be more useful

since there was a different number of movies released each year.

So we'll add another column for

the average revenue earned by each year's movie.

We can do this in the same pivot table.

We'll add another value and

change the function that we use to summarize from sum to average.

Play video starting at :4:40 and follow transcript4:40

The average function gives us the average revenue per year for

the movies in the dataset.

We can see that the average revenue in 2015 was much lower than the other years.

Since this data stands out so much, let's keep exploring to find out why.

Taking your analysis to another level like this is a sign of a great analyst.

When you're in your job, you want to answer the questions that your manager and

stakeholders ask.

But you also want to answer the ones that come up while you're doing your analysis.

So let's try to figure this out.

First, we'll know how many movies from each year were included in the dataset,

we'll add a new value and use the count function this time.

Play video starting at :5:28 and follow transcript5:28

This shows us that there are more movies in the data set from 2015

than from any other year.

But 2015 still has the second lowest total box office revenue.

This could mean a few things.

It's likely that a lot of the movies from 2015 just didn't earn much revenue

compared to the other years, which would bring down the overall average revenue.

Even if the total revenue remained on par with the other years.

We'll explore just this one possibility here.

But you can always go further when you analyze data in your own job.

It will depend on your objectives and the questions you need to answer.

For now let's copy and paste our pivot table so we can test our hypothesis.

Play video starting at :6:23 and follow transcript6:23

We'll rename the columns and

our copy table to differentiate them from our original table.

We'll name them based on the data we'll be looking at,

which I'll explain in the next video.

Play video starting at :6:48 and follow transcript6:48

Now our copied pivot table is ready for us to test our hypothesis.

Next, we're going to use filters to find out how many movies earned less

than $10 million revenue in 2015.

Then we'll also create a calculated field to determine what

percentage of the total movies from that year they represent.

I'll be here when you're ready to learn more about pivot tables.

Welcome back. In the last video,

we created a pivot table of movie data and

revenue calculations to help our manager

think through new movie ideas.

We used our pivot table to make

some initial observations about annual revenue.

We also discovered that the average revenue for 2015

was lower than other years even

though more movies were released that year.

We hypothesized that this was because more movies

that earn less than $10 million

in revenue were released in 2015.

To test this theory,

we created a copy of our original pivot table.

Now we are going to apply filters in

calculated fields to explore the data more.

Let's get started.

You all remember that the filter option

lets us view only the values we need.

We'll select a cell in

our copied pivot table and add

a filter to the box office revenue column.

The filter will then be applied to the entire table.

When we open the status menu,

we can choose to filter the data to show specific values.

Play video starting at :1:11 and follow transcript1:11

But in our case, we want to filter by condition so we can

figure out how many movies

in each year earn less than $10 million.

The condition we'll use in

our filter is less than and our value will

be $10 million which is why

we renamed these columns earlier.

We'll type our number in a dollar and cents format

so the condition matches the data in our pivot table.

This might not be necessary,

but it prevents potential errors from happening.

Now we know that 20 movies released in

2015 made less than $10 million.

This seems like a high number

compared to the other years.

But keep in mind,

there were more movies from our data

set released in 2015.

Before we move on,

let's use a calculated field to verify our average

because it was copied from

another pivot table before we filtered it.

That way we can check that it's correct.

We'll create a customized column called

a calculated field using our values menu.

A calculated field is

a new field within a pivot table that carries

out certain calculations based

on the values of other fields.

You can do this in Excel too using

field settings and the create formula menu.

For the formula in our calculated field,

we'll use the sum function and divide the sum of

the box office revenue data from

our original table by the count of the same data.

Because we applied our filter

to this pivot table earlier,

this formula will only return

the average revenue of movies under $10 million.

That worked. We were able to check

the accuracy of some of our data before analyzing it.

Always a good thing.

But it's still difficult to tell how much of an impact

these lower earning movies had on the average revenue.

Let's run a quick formula to find the percentage of

movies for each year that earned less than $10 million.

This will make it easier to compare from year to year.

Instead of a calculated field,

we'll add this as a formula in a new column,

that way we can pull data from both of our pivot tables.

We'll put a header for our table in

cell G10 and name it percent of total movies.

Then we'll add our formula to

the next cell in the column.

Divide the number of movies in

the copy table by the number

of movies in the original table.

Then we'll use the fill handle in the cell with

a formula and drag it to apply

the formula to the rest of the years.

Finally, we'll format these numbers as percentages.

Now our analysis shows that

16 percent of the movies released in

2015 earned less than $10 million of revenue.

The other years are all close to 10 percent.

This is one possible explanation for why

the average revenue is comparatively low in 2015.

In real life, we'd most likely need to take

our analysis even further depending on our goals.

But for now, we're all set.

You've learned how you can use

pivot tables to perform data calculations.

It will take practice,

but pivot tables are worth it

because they do more than calculate.

They organize and filter data too.

Together we've covered functions,

formulas, and pivot tables.

All great tools to use in analysis.

With practice and experience,

it will feel like you've used them forever.

Just take your time getting to know how they work.

Keep exploring these videos and the readings. Great work.

**Elements of a pivot table**

Previously, you learned that a pivot table is a tool used to sort, reorganize, group, count, total, or average data in spreadsheets. In this reading, you will learn more about the parts of a pivot table and how data analysts use them to summarize data and answer questions about their data.

**Pivot tables** make it possible to view data in multiple ways in order to identify insights and trends. They can help you quickly make sense of larger data sets by comparing metrics, performing calculations, and generating reports. They’re also useful for answering specific questions about your data.

A pivot table has four basic parts: rows, columns, values, and filters.

A screenshot of a computer

Description automatically generated

The **rows** of a pivot table organize and group data you select horizontally. For example, in the [Start working with pivot tables](https://www.coursera.org/learn/analyze-data/lecture/HCOme/start-working-with-pivot-tables) video, the Release Date values were used to create rows that grouped the data by year.

A close-up of a date

Description automatically generated

The **columns** organize and display values from your data vertically. Similar to rows, columns can be pulled directly from the data set or created using **values**. **Values** are used to calculate and count data. This is where you input the variables you want to measure. This is also how you create calculated fields in your pivot table. As a refresher, a **calculated field** is a new field within a pivot table that carries out certain calculations based on the values of other fields

In the previous movie data example, the **Values** editor created columns for the pivot table, including the **SUM** of Box Office Revenue, the **AVERAGE** of Box Office Revenue, and the **COUNT** of Box Office Revenue columns.

A screenshot of a graph

Description automatically generated

Finally, the **filters** section of a pivot table enables you to apply filters based on specific criteria—just like filters in regular spreadsheets! For example, a filter was added to the movie data pivot table so that it only included movies that generated less than $10 million in revenue.

A screenshot of a graph

Description automatically generated

Being able to use all four parts of the pivot table editor will allow you to compare different metrics from your data and execute calculations, which will help you gain valuable insights.

**Using pivot tables for analysis**

Pivot tables can be a useful tool for answering specific questions about a dataset so you can quickly share answers with stakeholders. For example, a data analyst working at a department store was asked to determine the total sales for each department and the number of products they each sold. They were also interested in knowing exactly which department generated the most revenue.

Instead of making changes to the original spreadsheet data, they used a pivot table to answer these questions and easily compare the sales revenue and number of products sold by each department.

A screenshot of a cell phone

Description automatically generated

They used the department as the rows for this pivot table to group and organize the rest of the sales data. Then, they input two Values as columns: the SUM of sales and a count of the products sold. They also sorted the data by the SUM of sales column in order to determine which department generated the most revenue.

A screenshot of a screen

Description automatically generated

Now they know that the Toys department generated the most revenue!

**Key takeaways**

Pivot tables are an effective tool for data analysts working with spreadsheets because they highlight key insights from the spreadsheet data without having to make changes to the spreadsheet. Coming up, you will create your own pivot table to analyze data and identify trends that will be highly valuable to stakeholders.

# Use pivot tables in analysis

In this reading, you will learn how to create and use pivot tables for data analysis. You will also get some resources about pivot tables that you can save for your own reference when you start creating pivot tables yourself. **Pivot tables** are a spreadsheet tool that let you view data in multiple ways to find insights and trends.

Pivot tables allow you to make sense of large data sets by giving you tools to easily compare metrics, quickly perform calculations, and generate readable reports. You can create a pivot table to help you answer specific questions about your data. For example, if you were analyzing sales data, you could use pivot tables to answer questions like, “Which month had the most sales?” and “What products generated the most revenue this year?” When you need answers to questions about your data, pivot tables can help you cut through the clutter and focus on only the data you need.

## Create your pivot table

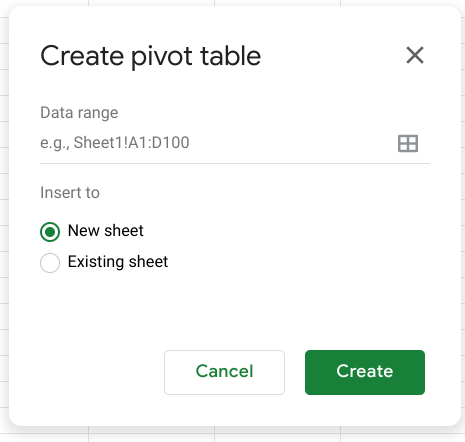
Before you can analyze data with pivot tables, you will need to create a pivot table with your data. The following includes the steps for creating a pivot table in Google Sheets, but most spreadsheet programs will have similar tools.

First, you will open the **Insert** menu from the toolbar; there will be an option for **Pivot table**.

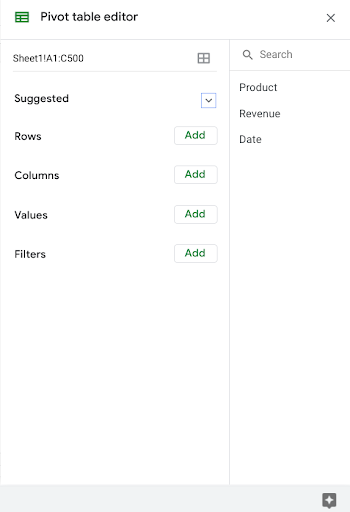
A screenshot of a computer

Description automatically generated

This pop-up menu will appear:

There is an option to select New sheet or Existing sheet and a Create button

Generally, you will want to create a new sheet for your pivot table to keep your raw data and your analysis separate. You can also store all of your calculations in one place for easy reference. Once you have created your pivot table, there will be a pivot table editor that you can access to the right of your data.



This is where you will be able to customize your pivot table, including what variables you want to include for your analysis.

## Using your pivot table for analysis

You can perform a wide range of analysis tasks with your pivot tables to quickly draw meaningful insights from your data, including performing calculations, sorting, and filtering your data. Below is a list of online resources that will help you learn about performing basic calculations in pivot tables as well as resources for learning about sorting and filtering data in your pivot tables.

### **Perform calculations**

| **Microsoft Excel** | **Google Sheets** |
| --- | --- |
| [**Calculate values in a pivot table**](https://support.microsoft.com/en-us/office/calculate-values-in-a-pivottable-11f41417-da80-435c-a5c6-b0185e59da77)**:** Microsoft Support’s introduction to calculations in Excel pivot tables. This is a useful starting point if you are learning how to perform calculations with pivot tables specifically in Excel. | [**Create and use pivot tables**](https://support.google.com/docs/answer/1272900?co=GENIE.Platform%3DDesktop&hl=en)**:** This guide is focused on using pivot tables in Google Sheets and it provides instructions for creating calculated fields. This is a quick how-to guide you can save and reference as a quick reminder on how to add calculated fields. |
| [**Pivot table calculated field example**](https://exceljet.net/pivot-table/pivot-table-calculated-field-example)**:** This resource includes a detailed example of a pivot table being used for calculations. This step-by-step process demonstrates how calculated fields work, and provides you with some idea of how they can be used for analysis. | [**All about calculated field in pivot tables**](https://infoinspired.com/google-docs/spreadsheet/all-about-calculated-field-in-pivot-table-in-google-sheets/)**:** This is a comprehensive guide to calculated fields for Google Sheets. If you are working with Sheets and are interested in learning more about pivot tables, this is a great resource. |
| [**Pivot table calculated fields: step-by-step tutorial**](https://powerspreadsheets.com/pivottable-calculated-fields/)**:** This tutorial for creating your own calculated fields in pivot tables is a really useful resource to save and bookmark for when you start to apply calculated fields to your own spreadsheets. | [**Pivot tables in Google Sheets**](https://www.benlcollins.com/spreadsheets/pivot-tables-google-sheets/)**:** This beginner’s guide covers the basics of pivot tables and calculated fields in Google Sheets and uses examples and how-to videos to help demonstrate these concepts. |

### **Sort your data**

| **Microsoft Excel** | **Google Sheets** |
| --- | --- |
| [**Sort data in a pivot table or PivotChart**](https://support.microsoft.com/en-us/office/sort-data-in-a-pivottable-or-pivotchart-e41f7107-b92d-44ef-861f-24430830450a)**:** This is a Microsoft Support how-to guide to sorting data in pivot tables. This is a useful reference if you are working with Excel and are interested in checking out how filtering will appear in Excel specifically. | [**Customize a pivot table**](https://support.google.com/docs/answer/7572895?co=GENIE.Platform%3DDesktop&hl=en)**:** This guide from Google Support focuses on sorting pivot tables in Google Sheets. This is a useful, quick reference if you are working on sorting data in Sheets and need a step-by-step guide. |
| [**Pivot tables- Sorting data**](https://www.tutorialspoint.com/excel_pivot_tables/excel_pivot_tables_sorting_data.htm)**:** This tutorial for sorting data in pivot tables includes an example with real data that demonstrates how sorting in Excel pivot tables works. This example is a great way to experience the entire process from start to finish. | [**How to sort pivot table columns**](https://infoinspired.com/google-docs/spreadsheet/pivot-table-columns-in-custom-order-in-google-sheets/)**:** This detailed guide uses real data to demonstrate how the sorting process for Google Sheet pivot tables will work. This is a great resource if you need a slightly more detailed guide with screenshots of the actual Sheets environment. |
| [**How to sort a pivot table by value**](https://exceljet.net/lessons/how-to-sort-a-pivot-table-by-value)**:** This source uses an example to explain sorting by value in pivot tables. It includes a video, which is a useful guide if you need a demonstration of the process. | [**Pivot table ascending and descending order**](https://medium.com/actiondesk/pivot-table-ascending-descending-order-in-google-sheets-and-excel-1-minute-ultimate-beginners-8f9f4c560492)**:** This 1-minute beginner’s guide is a great way to brush up on sorting in pivot tables if you are interested in a quick refresher. |

### **Filter your data**

| **Microsoft Excel** | **Google Sheets** |
| --- | --- |
| [**Filter data in a pivot table**](https://support.microsoft.com/en-us/office/filter-data-in-a-pivottable-cc1ed287-3a97-4e95-b377-ddfafe79fa8f)**:** This resource from the Microsoft Support page provides an explanation of filtering data in pivot tables in Excel. If you are working in Excel spreadsheets, this is a great resource to have bookmarked for quick reference. | [**Customize a pivot table**](https://support.google.com/docs/answer/7572895?co=GENIE.Platform%3DDesktop&hl=en)**:** This is the Google Support page on filtering pivot table data. This is a useful resource if you are working with pivot tables in Google Sheets and need a quick resource to review the process. |
| [**How to filter Excel pivot table data**](https://www.dummies.com/software/microsoft-office/excel/how-to-filter-excel-pivot-table-data/)**:** This how-to guide for filtering data in pivot tables demonstrates the filtering process in an Excel spreadsheet with data and includes tips and reminders for when you start using these tools on your own. | [**Filter multiple values in pivot table**](https://infoinspired.com/google-docs/spreadsheet/filter-multiple-values-in-pivot-table-sheets/)**:** This guide includes details about how to filter for multiple values in Google Sheet pivot tables. This resource expands some of the functionality that you have already learned and sets you up to create more complex filters in Google Sheets. |

### **Format your data**

| **Microsoft Excel** | **Google Sheets** |
| --- | --- |
| [**Design the layout and format of a PivotTable**](https://support.microsoft.com/en-us/office/design-the-layout-and-format-of-a-pivottable-a9600265-95bf-4900-868e-641133c05a80): This Microsoft Support article describes how to change the format of the PivotTable by applying a predefined style, banded rows, and conditional formatting. | [**Create and edit pivot tables**](https://support.google.com/a/users/answer/9308944#group_data_in_a_pivot_table): This Help Center article provides information about how to edit a pivot table to change its style, and group data. |

Pivot tables are a powerful tool that you can use to quickly perform calculations and gain meaningful insights into your data directly from the spreadsheet file you are working in! By using pivot table tools to calculate, sort, and filter your data, you can immediately make high-level observations about your data that you can share with stakeholders in reports.

But, like most tools we have covered in this course, the best way to learn is to practice. This was just a small taste of what you can do with pivot tables, but the more you work with pivot tables, the more you will discover.

### 1.

Question 1

Fill in the blank: The \_\_\_\_\_ of a pivot table organize and group the selected data horizontally.

1 point

filters

values

rows

columns

### 2.

Question 2

What is the purpose of this pivot table?

|  |  |
| --- | --- |
| Route | Mileage |
| 10 | 799 |
| 15 | 778 |
| 12 | 234 |
| 21 | 412 |
| 9 | 900 |
| 2 | 505 |
| 22 | 112 |
| 13 | 607 |
| 7 | 199 |
| 4 | 299 |
| 11 | 808 |
| 20 | 190 |
| **Total** | **5843** |

1 point

Sum the mileage totals for each individual route

Filter routes by mileage type

Sort the routes by greatest to least mileage

Find the average mileage total for each route

### 3.

Question 3

How could this pivot table be adjusted to show the same data, but only for routes categorized as West Coast?

|  |  |
| --- | --- |
| Route | Mileage |
| 10 | 799 |
| 15 | 778 |
| 12 | 234 |
| 21 | 412 |
| 9 | 900 |
| 2 | 505 |
| 22 | 112 |
| 13 | 607 |
| 7 | 199 |
| 4 | 299 |
| 11 | 808 |
| 20 | 190 |
| **Total** | **5843** |

1 point

Summarize the values by route location

Add a new column labeled West Coast Locations

Add a filter to show only routes located in the West Coast

Sort the current row by route location

### 4.

Question 4

Which spreadsheet tool finds an average value using values generated within a pivot table?

1 point

Data validation

Filter

Calculated field

Conditional formatting

By now, you probably know that there's

more than one way to do the daily task of a data analyst.

Calculations are no exception.

As we've shown in earlier videos,

you can complete the same calculations in

lots of different ways in spreadsheets.

You can also complete them using SQL.

In this video, we'll give you an overview of how

SQL calculations compare to spreadsheet calculations.

Let's look at the arithmetic operators that

are used in both spreadsheets and SQL.

An operator is a symbol that names the type of

operation or calculation to be performed in a formula.

As you learned earlier,

the four basic arithmetic operators in

spreadsheet formulas are the plus sign for addition,

the minus or hyphen for subtraction,

the asterisk for multiplication,

and the forward slash for division.

These same operators calculate data

in the same way when writing queries in SQL.

The operators are embedded in

the queries when pulling data from a database.

Just like spreadsheet formulas,

there's a few different ways to

perform calculations using queries.

Let's look at the syntax for one possible query.

The syntax of a query is its structure.

It should include all the specific details

of the data you want to

pull into a new table

where those details should be placed.

If you want to add values from two columns of a table,

you start with the SELECT command,

followed by the name of your first column,

then the name of your second column.

Then you'd add the names of

both columns with a plus sign between them.

After that, you'd type AS followed

by the name you'd like to give

the column with the added totals.

Finally, you then complete your query by typing FROM

and then the name of

the table that you're pulling the data from.

Running this query would get you a table

showing the two columns whose values are

being added together plus

a new column showing the sums of those values.

The operator in this query is

a plus sign since values are being added.

If you needed to subtract, multiply, or divide,

you'd follow the same steps

using the appropriate operators.

If you need to use

more than one arithmetic operator in a calculation,

you'd use parentheses to control

the order of the calculations.

If we included column C in our query,

we could place parentheses around column A plus column B.

We then add an asterisk if we're multiplying followed by

column C. This query would return in a new column,

the sum of the values in column A and B

multiplied by the values in column C. Now,

let's say you only wanted

the remainder from a division calculation.

Well, you need a different operator for

this, the modulo operator.

The modulo operator is represented by the percent symbol.

This is an operator that returns

the remainder when one number is divided by another.

In a spreadsheet, you could complete

the same calculation using the MOD function.

This brings us to another similarity between

calculations in spreadsheets and SQL.

A lot of times, you can use functions instead

of operators to complete calculations.

For example, the SUM function can

complete addition problems in spreadsheets and SQL.

The AVERAGE function in a spreadsheet is the same

as the AVG function in SQL.

They both return the average value of a set of numbers.

In SQL, these functions are

considered aggregate functions because they perform

a calculation on one or more values

and return a single value.

You'll learn more about how they're used

with the GROUP BY command in a query soon.

Those are the basics of SQL calculations.

Knowing how to write a query for

a calculation is a good first step.

Stay with us, and you'll learn more about

calculations in SQL. Bye for now.

**Upload the avocado dataset to BigQuery**

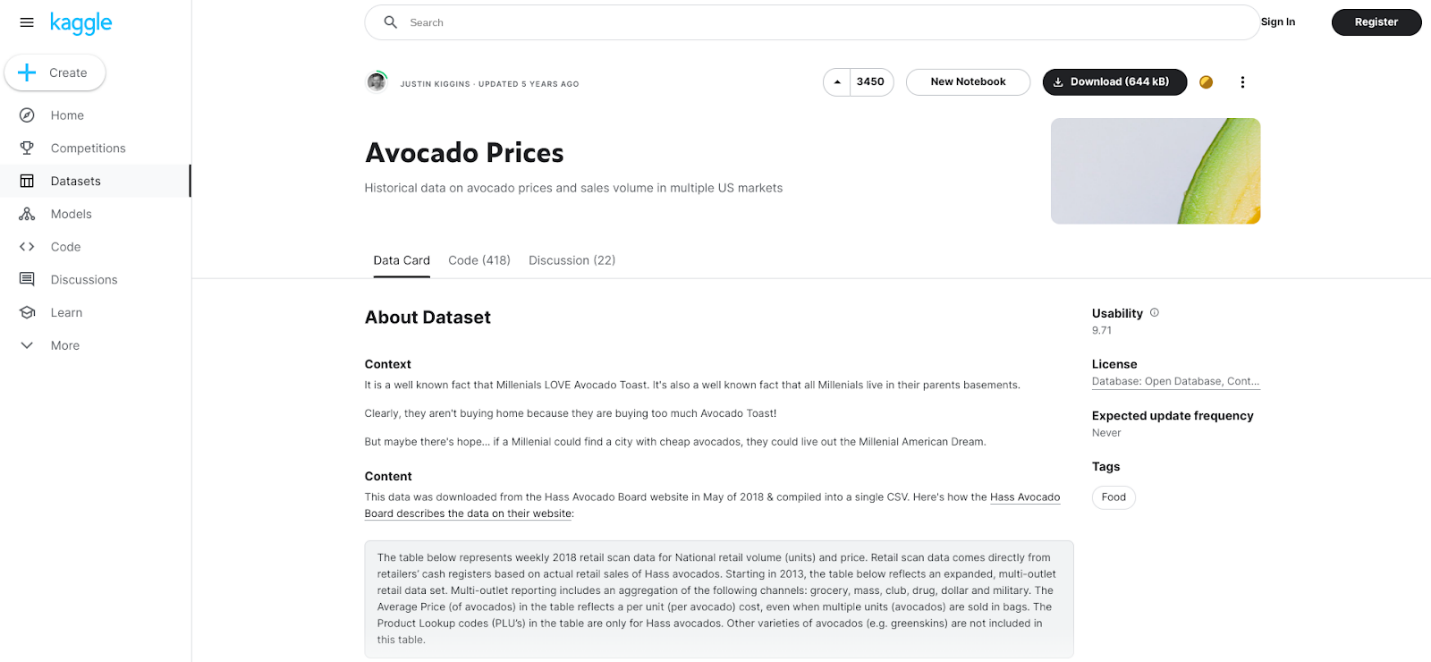
Using public datasets is a great way to practice working with SQL. Later in the course, you are going to use historical data on avocado prices to perform calculations in BigQuery. This is a step-by-step guide to help you load this data into your own BigQuery console so that you can follow along with the upcoming video.

**Upload the avocado dataset to BigQuery**

In an upcoming video, the instructor demonstrates how to embed simple calculations in SQL. For this example, they use a publicly available Avocado Prices dataset from Kaggle. Follow the directions below to upload the avocado dataset to BigQuery. Once you’ve uploaded this data, you’ll be able to practice with this data on your own!

**Step 1:** Download the publicly available dataset [Avocado Prices from Kaggle.](https://www.kaggle.com/neuromusic/avocado-prices) This data has been made available by [Justin Kiggins](https://www.kaggle.com/neuromusic) under an [Open Data Commons](https://opendatacommons.org/licenses/odbl/1-0/) license. Kaggle is a great resource for all types of data analytics resources, and there are also other public datasets on the platform that you can download and use.

**Note:** You will need to create a free account before downloading the zipped data files.



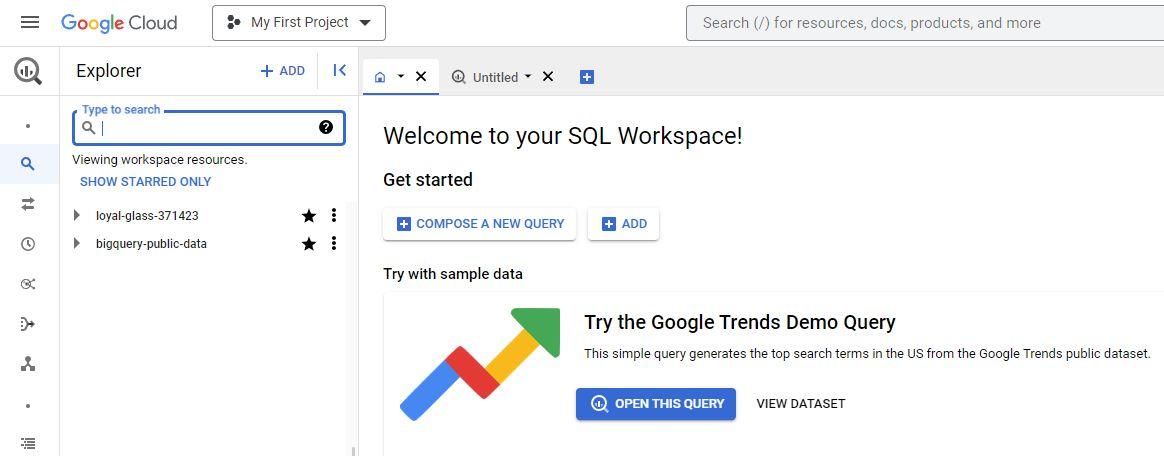
You will find some more information about the avocado dataset, including the context, content, and original source on this page. For now, you can simply download the file.

**Step 2:** Create the dataset.

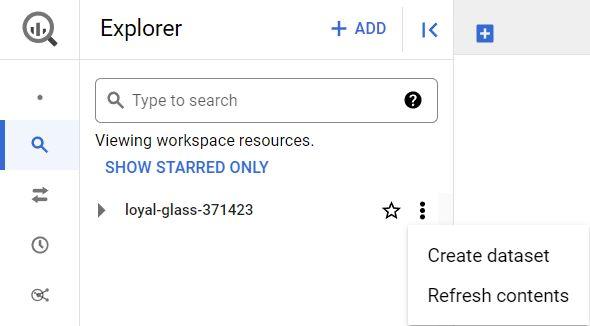
1. After you have downloaded the dataset from Kaggle, extract the **zipped folder**. Remember where you save the avocado .csv file for upload into your BigQuery console.

2. Open your **BigQuery Console** and create a new dataset.

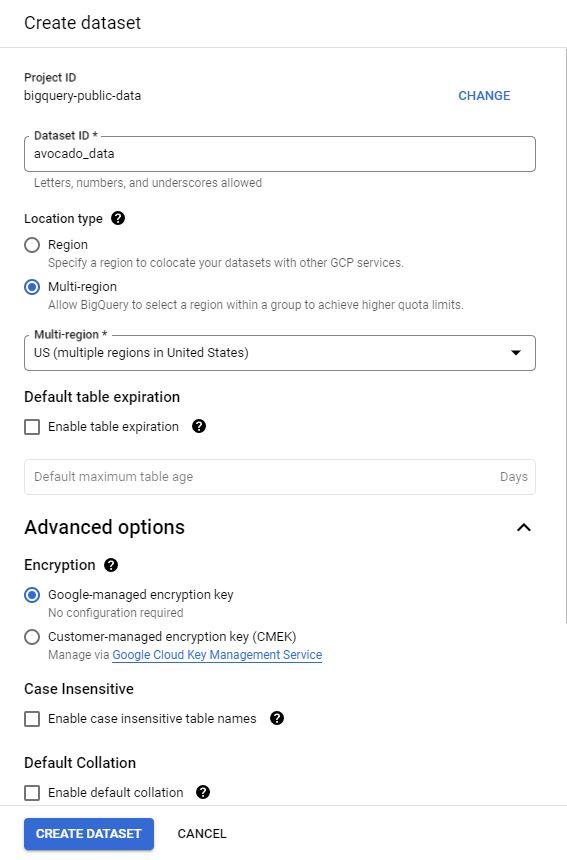
3. In the **Explorer** pane on the left side of your console, **select the project** where you want to add a dataset. Note that your project will have a unique name and won't be the same as the one in the example pictured below (loyal-glass-371423). If you already have it starred, don't choose bigquery-public-data as your project because that's a public project that you can't change.



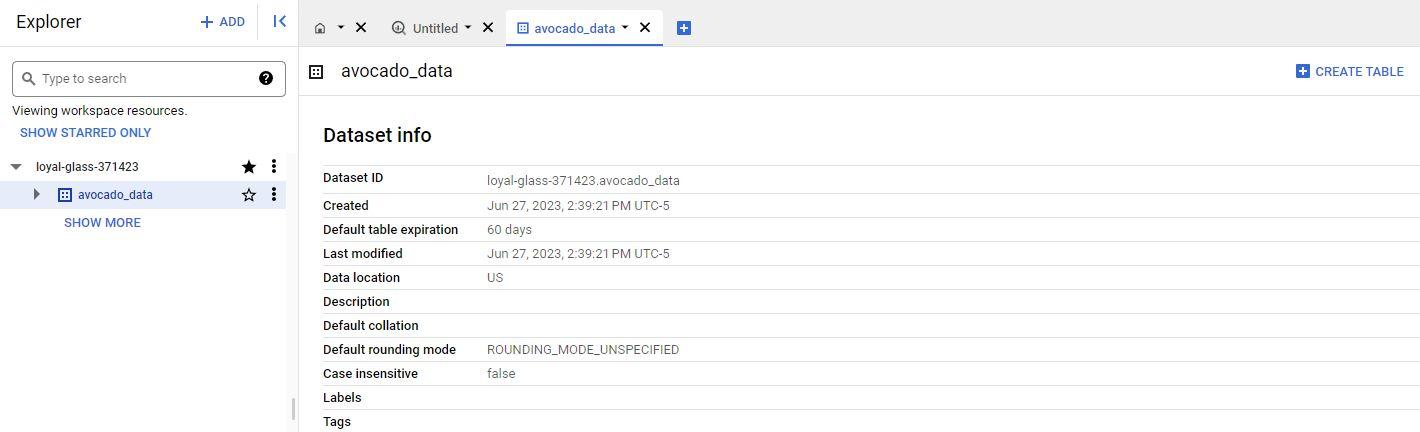
4. Select the **Actions** icon (three vertical dots) next to your project and select **Create dataset**.



5. Name the dataset **avocado\_data**. In the **Location type** section, select **Multi-region,** then select **US (multiple regions in United States**, and make sure the default **Encryption** method within the **Advanced options** is set to the **Google\_managed encryption key**. Then, select **Create dataset** (blue button) to create your new dataset. This will add data in the **Explorer** on the left of your console.

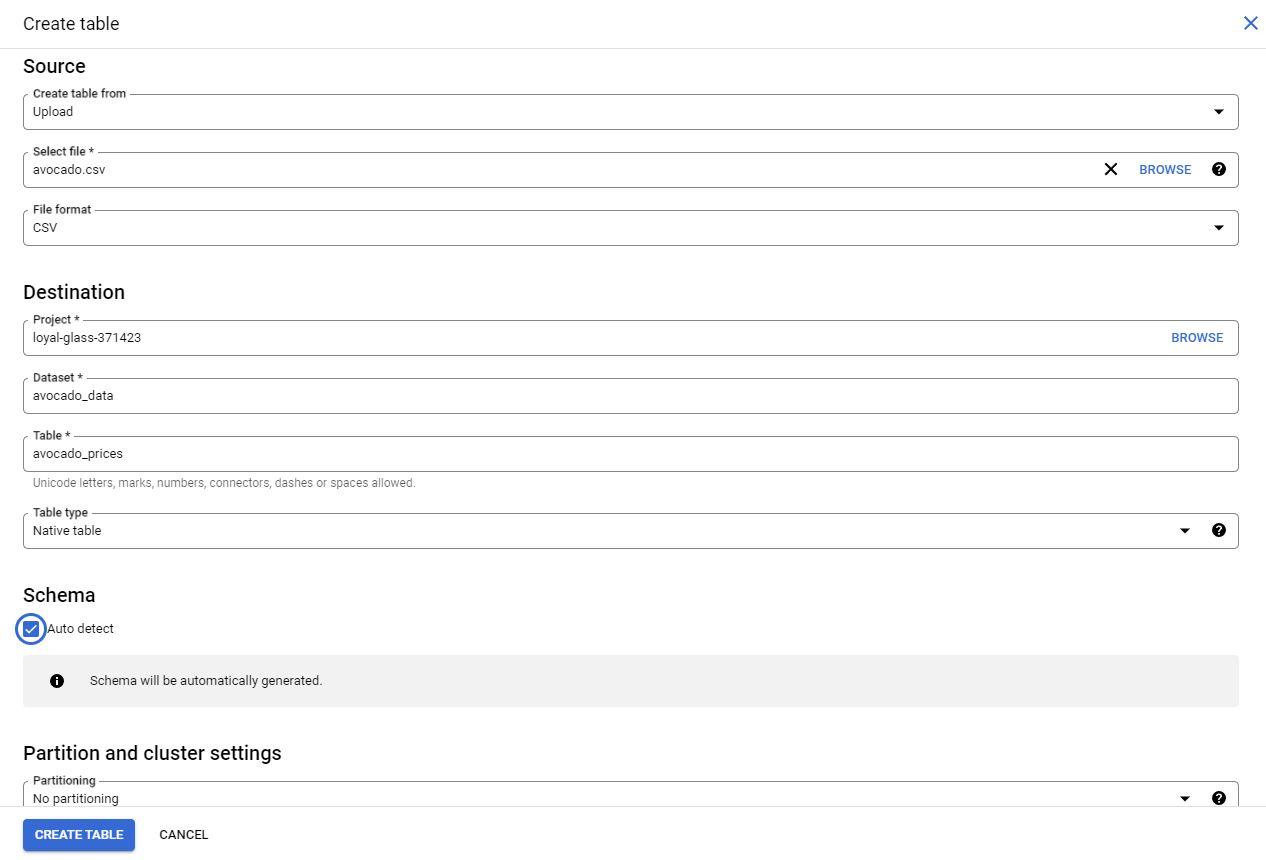


6. Navigate to the dataset in your console by expanding your project and selecting the correct dataset listed. In this case, it will be **avocado\_data**.



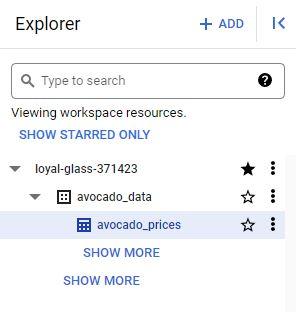
7. On the far right of the screen, select the blue **+ CREATE TABLE** button to open the Create table window. To fill out this window:

* Under **Source**, for the Create table from selection, select **Upload**.
* Click **Browse** to select the unzipped .csv file titled avocado you just downloaded to your computer from Kaggle. The **File format** should automatically change from **Avro** to **CSV** when you select the file.
* For **Table** name, enter **avocado\_prices.**
* For **Schema,** click the **Auto detect** checkbox.
* Finally, to create the table, select **Create table** (blue button).

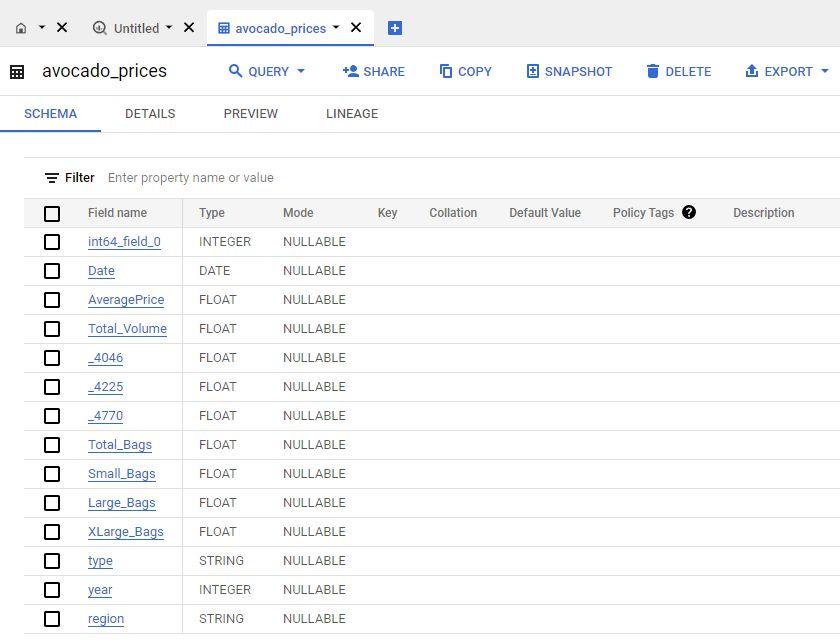


**Step 3:** Review the table information.

1. In the **Explorer**, the **avocado\_prices** table will appear under the dataset you created.



2. Select the table you created to develop a greater understanding of the table schema, details, and the data preview in the main editor window.



Well done! You are now ready to follow along with the video and learn more about performing calculations with queries!

**Step-by-Step: Embed simple calculations with SQL**

This reading outlines the steps the instructor performs in the next video, [Embed simple calculations with SQL](https://www.coursera.org/learn/analyze-data/lecture/RlnmJ/embed-simple-calculations-in-sql). In this video, the goal is to find out the total number of bags of avocados sold on each date at each location using the dataset you loaded to BigQuery.

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**

In order to follow along with the instructor, you will need the employee dataset uploaded into your project space. If you haven’t already uploaded this data, follow the instructions in the [Upload the avocado dataset to BigQuery](https://www.coursera.org/learn/analyze-data/supplement/7Khvl/optional-upload-the-avocado-dataset-to-bigquery) reading.

**Example 1: Verify the total number of bags**

Use the following steps to perform some simple calculations with SQL and verify the total number of bags.

1. Open the BigQuery editor.

2. On line 1, enter **SELECT** and press **Enter**. You'll use the **SELECT** command to pull certain columns from the table. Because you are selecting several columns, press **Enter** after **SELECT** and after the comma after each column name.

3. Enter the following column names into your editor:

1

2

3

4

5

6

7

SELECT

    Date,

    Region,

    Small\_Bags,

    Large\_Bags,

    XLarge\_Bags

    Total\_Bags

4. Note the use of underscores in this example. Spaces can confuse certain servers and applications. Using underscores helps avoid potential issues while keeping the names readable.

5. Now add the calculation to the query using the names of the three columns with plus signs between them, as shown below. Add **\_Calc** to your new column to compare the columns to each other after you calculate your results.

1

2

3

4

5

6

7

8

SELECT

    Date,

    Region,

    Small\_Bags,

    Large\_Bags,

    XLarge\_Bags

    Total\_Bags

    Small\_Bags + Large\_Bags + XLarge\_Bags AS Total\_Bags\_Calc

6. Finally, finish the query with **FROM** and the name of the dataset and subset: avocado\_data.avocado\_prices.

1

2

3

4

5

6

7

8

9

SELECT

    Date,

    Region,

    Small\_Bags,

    Large\_Bags,

    XLarge\_Bags

    Total\_Bags

    Small\_Bags + Large\_Bags + XLarge\_Bags AS Total\_Bags\_Calc

FROM avocado\_data.avocado\_prices

7. Select **RUN** to run the query.

**Example 2: Calculating the percentage of small bags**

Now that you have verified the total number of bags, you can use those values in another query. You need to find what percent of the total number of bags were small bags. This information might help stakeholders make decisions about how to package avocados or which size bag to run a sale on. So run a new query to calculate the percentage of small bags:

1. Create a new query.

2. Use the **SELECT** command to select the Date, Region, Total\_Bags, and Small\_Bags columns.

1

2

3

4

5

SELECT

    Date,

    Region,

    Total\_Bags,

    Small\_Bags

3. To find the percentage of small bags, divide the number of small bags by the number of total bags using a slash as the operator. Put this part of the calculation in parentheses to let the server know that this calculation should be performed first. Then add \*100 to give a percentage versus a decimal.

1

2

3

4

5

6

SELECT

    Date,

    Region,

    Total\_Bags,

    Small\_Bags

    (Small\_Bags / Total\_Bags)\*100

4. Use the **AS** command to name this new column Small Bags Percent.

1

2

3

4

5

6

SELECT

    Date,

    Region,

    Total\_Bags,

    Small\_Bags

    (Small\_Bags / Total\_Bags)\*100 AS Small\_Bags\_Percent

5. Add **FROM** and the dataset.

1

2

3

4

5

6

7

8

SELECT

    Date,

    Region,

    Total\_Bags,

    Small\_Bags

    (Small\_Bags / Total\_Bags)\*100 AS Small\_Bags\_Percent

FROM

    avocado\_data.avocado\_prices

6. Select **RUN** to run the query.

7. Resolve the “divide by 0” error message using the **WHERE** command. **WHERE** lets you add a condition to your calculation.

1

2

3

4

5

6

7

8

9

10

SELECT

    Date,

    Region,

    Total\_Bags,

    Small\_Bags

    (Small\_Bags / Total\_Bags)\*100 AS Small\_Bags\_Percent

FROM

    avocado\_data.avocado\_prices

WHERE

    Total\_Bags <>0

8. Select **RUN** to run the query again. Notice your new column shows the percent of small bags in the total bags count

With SQL, you can complete just about any calculation you want during your analysis. Embedding the calculations in your queries will help you keep your analysis organized while getting your results. The methods shown here are just the beginning!

Hi again. Earlier we showed you

how to complete calculations in SQL.

While there are a few different ways,

embedding them in queries is a very useful one.

When you include a calculation in

a query with other commands,

you can do more work faster.

Here's a basic query syntax that we talked about.

We start with SELECT and then the names of

the columns we want to use in our calculations.

Then we add in calculation details including

an operator like a forward slash for division.

Next, we type AS followed by

the new column name to label

the column with the calculated values.

Finally, we end our query with

the FROM command and the name of

the table that we're pulling data from.

Now, let's take it to the next level with

some embedded calculations that

use a syntax like this one.

Even better, we'll do this with some data about avocados.

Sorry to those of you who aren't avocado lovers like me.

Let's get started. Feel free to continue

watching as we show you the steps using BigQuery.

If you're joining us, open up

your tool of choice for using SQL.

Be sure to look through the instructions in the reading

right before this video to help you get started.

Data is already loaded, so we can jump right in.

Our goal is to find out the total number of bags of

avocados sold on each date

at each location using this data.

There's already a column that shows us the total,

but we want to make sure we

understand how that total is calculated.

We want to make sure that the total column is just small,

large, and extra-large bags added together.

We'll add the values in those three columns together in

our query and then compare them to

the total bags column in the dataset.

We'll start with the SELECT command,

which we'll use to pull certain columns from the table.

We are selecting several columns,

so we'll press Enter after SELECT and

after the comma after each column name.

Next, we'll type those column names: Date, Region,

Play video starting at :2:11 and follow transcript2:11

Small\_bags, Large\_bags, XLarge\_Bags, and Total\_Bags.

Underscores are the lines used to

underline words and connect text characters.

Using spaces can confuse

certain servers and applications.

Using underscores instead helps avoid

potential issues while keeping the names readable.

Now we'll add the calculation to

the query using the names of

the three columns with plus signs between them:

small bags plus large bags plus extra large bags.

Play video starting at :2:57 and follow transcript2:57

Since we want this calculation in a new column,

we'll use the AS command to

name the column Total\_Bags\_Calc.

Play video starting at :3:8 and follow transcript3:08

We've added the word "Calc" so we can compare

the columns to each other after we calculate our results.

Now, we'll finish our query with FROM

and the name of the dataset and subset

we're pulling from, avocado\_data.avocado\_prices.

Play video starting at :3:33 and follow transcript3:33

Let's run the query.

Play video starting at :3:37 and follow transcript3:37

In the "Total Bags Calc" column,

the data shows the sum of

each date for the number of small,

large, and extra large bags of

avocados that were sold at each location.

If we quickly compare

the two columns showing the total number of bags,

we learn that the values are the same.

Play video starting at :4: and follow transcript4:00

This lets us know that the data we

want to use is the right data.

Now that we have verified the total number of bags,

we can use those values in another query.

We need to find what percent of

the total number of bags were small bags.

Finding this out might help

stakeholders make decisions on how to

package avocados or which size bag to run a sale on.

Our job is to get that information to the stakeholders.

So we'll set up a new query.

We'll select the Date, Region,

Total Bags, and Small Bags columns for this query.

Play video starting at :4:49 and follow transcript4:49

Next, we'll set up a new column

starting with our calculation.

To find the percentage of small bags,

we need to first divide the number of small bags by

the number of total bags using a slash as the operator.

We'll put this part of the calculation

in parentheses to let

the server know that

this calculation should be performed first.

Then we'll multiply this total by

100 using an asterisk as our operator.

Multiplying by 100 gives us

a value that's a percentage instead of a decimal.

Percentages usually make it easier for people to

understand quickly when you share results with them.

We'll use the AS command to name

this new column, "Small Bags Percent."

Play video starting at :5:38 and follow transcript5:38

Then we'll add FROM and

the name of the set we're pulling from,

Play video starting at :5:49 and follow transcript5:49

and we'll run our query.

Play video starting at :5:57 and follow transcript5:57

We got an error in our results.

It says that we can't divide by zero.

Since we're finding percentages,

dividing by zero won't work.

This means that somewhere in

the dataset there is total bags equal to zero.

We'll have to fix this in our query.

We can fix this using the WHERE command.

WHERE lets us add a condition to our calculation.

After we type WHERE,

we'll type Total\_Bags followed by a less than sign,

and then a greater than sign.

These symbols tell the server that the values we are

calculating should not be equal to the value we specify.

In this case, that value is zero.

So we'll add a zero to our query.

Now, when we run the query,

you'll notice our new column shows the percent

of small bags in the total bags count.

Play video starting at :6:50 and follow transcript6:50

We'd get the same result if we

use an exclamation mark followed

by an equal sign in place of the

less than and greater than signs.

Play video starting at :7:7 and follow transcript7:07

Note that this is one way for doing it.

But there are functions such as

SAFE\_DIVIDE that also allow you to avoid this error.

Those are just a couple of examples to get you started.

But with SQL, you can complete just about

any calculation you want during your analysis.

Embedding the calculations in your queries will help you

keep your analysis organized while getting your results.

The calculation methods we showed

you here are just the beginning.

So look for more coming up. See you soon.

Hey, good to see you.

As a data analyst, you'll find that your calculations come in all shapes and sizes.

Earlier we showed you how to do some of the more basic calculations in SQL.

While, basic calculations are great.

Sometimes you'll need to group data before completing calculations.

The GROUP BY and ORDER BY commands help you do this.

These commands are usually paired with aggregate functions like SUM or COUNT.

We'll show you how you can use these commands and functions to calculate and

summarize data from groups of rows in a table.

Let's explore the GROUP BY command first.

GROUP BY is a command that groups rows that have the same values

from a table into summary rows.

The GROUP BY command is used with SELECT statements. In a basic SELECT FROM or

SELECT-FROM-WHERE query, GROUP BY comes at the end of the query.

All right, let's try using GROUP BY.

We'll work with a database, with data from a bike sharing system.

We want to find out how many rides people took on these bikes per year.

Play video starting at :1:14 and follow transcript1:14

This data has several columns, but for this task,

we only need the start time column.

Play video starting at :1:20 and follow transcript1:20

Since this data set isn't organized by date, and the start time column isn't

organized by year, we'll need to include steps in our code to organize it.

We also want a total number of rides each year.

So we'll need to include a calculation in our query for this.

And depending on the questions we've been tasked with answering,

this might be the first step of many in our analysis.

Play video starting at :1:44 and follow transcript1:44

We'll start our query with the SELECT command.

We'll then add EXTRACT to our query.

Play video starting at :1:54 and follow transcript1:54

The EXTRACT command lets us pull one part of a given date to use.

We'll extract the year from the start time column.

To do this, we'll add an open parenthesis, followed by YEAR,

which lets the server know the part of the date we need.

Then we'll add the FROM command and STARTTIME so

we can get the year from all of the start times in that column.

We'll close the parentheses and then use AS and

the word year to name the column we're creating.

Play video starting at :2:24 and follow transcript2:24

On the next line of the query,

we'll use the aggregate function COUNT followed by an asterisk in parenthesis.

This will count the bike rides in the start time column.

Using the asterisk makes sure that all the start times are counted in the data.

Then we'll name our column number of rides with underscore between each word instead

of spaces.

Play video starting at :2:45 and follow transcript2:45

We will add FROM in the database we are pulling on the next line.

In this case, that's

bigquery-public-data.new\_york.citybike\_trips.

Play video starting at :3:1 and follow transcript3:01

And here is our GROUP BY command.

We'll use this to group the data by year.

So we'll type GROUP BY followed by year.

Play video starting at :3:10 and follow transcript3:10

We can further organize our results by using the ORDER BY command.

Adding this after GROUP BY orders the results.

Play video starting at :3:18 and follow transcript3:18

We'll add year to order the data by year.

It's good to note that by default, ORDER BY sorts data in ascending order.

Play video starting at :3:27 and follow transcript3:27

Now we can run our query to get the results.

Play video starting at :3:31 and follow transcript3:31

The years are ordered started with 2013 and ending with 2016.

If we want to change this to descending order,

we can add the keyword DESC to the end of the query and run it again.

Play video starting at :3:49 and follow transcript3:49

But whichever order you use, the GROUP BY and ORDER BY commands are great for

helping us complete and organise a calculation for our analysis.

This is one way to include calculations when aggregating data.

And it's just one of many ways SQL helps keep your analysis running smoothly

and moving forward.

There's still more to come on calculations and SQL.

Coming up, we'll learn more about data validation.

See you soon.

### 1.

Question 1

In a SQL query, what is the purpose of the modulo (%) operator?

1 point

Return the remainder of a division calculation

Convert a decimal to a percent

Apply an exponent to a value

Find the square root of a number

### 2.

Question 2

A data professional writes a query that uses more than one arithmetic operator. What do they add to the query to control the order of the calculations?

1 point

Colon [**:**]

Backslash [**/**]

Dollar sign (**$**)

Parenthesis [**()**]

### 3.

Question 3

Spreadsheet cell D5 contains the decimal .74. Which formula will convert it to a percentage?

1 point

**=D5,100**

**=D5(100)**

**=D5%100**

**=D5\*100**

### 4.

Question 4

What will **GROUP BY** do in this query?

1

2

3

SELECT apartment, AVG(price) AS apt\_prices

FROM rent\_data

GROUP BY apartment;

1 point

Group only the rows in the apt\_prices table

Group the rows in the table by apartment

Group together the apartment and rent\_data tables

Group together the rent\_data by apartment prices

Hi again. Earlier we covered data validation,

a spreadsheet function that

adds drop down lists to cells.

Using data validation lets you control

what can and can't be entered into your worksheet.

One of its uses is protecting

structured data and formulas in your spreadsheets.

But as useful as it is,

the data validation function is

just one part of a larger data validation process.

This process involves checking and

rechecking the quality of your data,

so that it is complete,

accurate, secure, and consistent.

While the data validation process

is a form of data cleaning,

you should use it throughout your analysis.

If this all sounds familiar to you, that's good.

Ensuring you have good data is super important.

In my opinion,

it's fun because you can pair

your knowledge of the business

with your technical skills.

This will help you understand your data,

check that it's clean, and make sure

you're aligning with your business objectives.

In other words, it's what you do to

make sure your data makes sense.

Keep in mind you'll build

your business knowledge with time and experience.

Here's a pro-tip, asking as many questions as

possible whenever you need to will make this much easier.

Let's say we're analyzing

some data for a furniture retailer.

We want to check that the values

in the purchase price column are

always equal to the number of

items sold times the product price.

We'll add a formula and a new column to recalculate

the purchase prices using a multiplication formula.

Play video starting at :1:46 and follow transcript1:46

Now comparing the totals,

there's at least one value that doesn't

match the value in the purchase price column.

We need to find an answer to

help us move forward with our analysis.

By doing some research and asking questions,

we find that there's a discount of

30% when customers buy five or more certain items.

If we hadn't run this check,

we could have missed this completely.

You've learned that as an analyst,

calculations are a big part of your job.

It's important that whenever you do calculations,

you always check to make

sure you've done them in the right way.

Sometimes you'll run data validation checks

that are common sense checks.

For example, let's say you're working

on an analysis to figure out

the effectiveness of in

store promotions for a business

that's only open on weekdays.

You check to make sure that there's

no sales data for Saturday and Sundays.

If your data does show sales on weekends,

it might not be a problem with the data itself.

It might not even be a problem at all.

There might be a good reason.

Like maybe your business hosts

special events on Saturdays and Sundays,

then you would have sales for those weekends.

You still might want to leave out the weekend sales in

your analysis if your objective

is only to look at the weekdays.

But doing the state of validation might save you from

miscalculations and other errors in your analysis.

You should always do data validation

no matter what analysis tool you're using.

In an earlier video, we use SQL

to analyze some data about avocados.

One of the queries was a check to make sure the data

showing the total number of bags was the sum of small,

large, and extra large bags.

By running this query,

we were able to determine that

the total number column was accurate.

That led us to continue our analysis.

Play video starting at :3:41 and follow transcript3:41

But when we tried to find

what percent of the total number of bags was small,

we ran into a small problem.

We received an error message about dividing by zero.

We fix that error by adjusting our query.

If we had linked that query to

a presentation that went to our stakeholders,

they'd show us the divide by

zero error instead of the figures we wanted.

By building in these types of checks,

as part of your data validation process,

you can avoid errors in your analysis and complete

your business objectives to make everyone happy.

Trust me, it's a great feeling when you do.

Another great feeling is knowing that you've made it

through another video and learned something new.

We have more where that came from coming soon. See you.

**Types of data validation**

This reading describes the purpose, examples, and limitations of six types of data validation. The first five are validation types associated with the data (type, range, constraint, consistency, and structure) and the sixth type focuses on the validation of application code used to accept data from user input.

As a junior data analyst, you might not perform all of these validations. But you could ask if and how the data was validated before you begin working with a dataset. Data validation helps to ensure the integrity of data. It also gives you confidence that the data you are using is clean. The following list outlines six types of data validation and the purpose of each, and includes examples and limitations.



* **Purpose**: Check that the data matches the data type defined for a field.
* **Example**: Data values for school grades 1-12 must be a numeric data type.
* **Limitations**: The data value 13 would pass the data type validation but would be an unacceptable value. For this case, data range validation is also needed.



* **Purpose**: Check that the data falls within an acceptable range of values defined for the field.
* **Example**: Data values for school grades should be values between 1 and 12.
* **Limitations**: The data value 11.5 would be in the data range and would also pass as a numeric data type. But, it would be unacceptable because there aren't half grades. For this case, data constraint validation is also needed.



* **Purpose**: Check that the data meets certain conditions or criteria for a field. This includes the type of data entered as well as other attributes of the field, such as number of characters.
* **Example**: Content constraint: Data values for school grades 1-12 must be whole numbers.
* **Limitations**: The data value 13 is a whole number and would pass the content constraint validation. But, it would be unacceptable since 13 isn’t a recognized school grade. For this case, data range validation is also needed.



* **Purpose**: Check that the data makes sense in the context of other related data.
* **Example**: Data values for product shipping dates can’t be earlier than product production dates.
* **Limitations**: Data might be consistent but still incorrect or inaccurate. A shipping date could be later than a production date and still be wrong.



* **Purpose**: Check that the data follows or conforms to a set structure.
* **Example**: Web pages must follow a prescribed structure to be displayed properly.
* **Limitations**: A data structure might be correct with the data still incorrect or inaccurate. Content on a web page could be displayed properly and still contain the wrong information.



* **Purpose:** Check that the application code systematically performs any of the previously mentioned validations during user data input.
* **Example:** Common problems discovered during code validation include: more than one data type allowed, data range checking not done, or ending of text strings not well defined.
* **Limitations:** Code validation might not validate all possible variations with data input.

Hello again.

Now, if you're like me, you always have sticky notes available nearby to write

a reminder or figure out a quick math problem.

Sticky notes are useful and important, but they're also disposable since you usually

only need them for a short time before you recycle them.

Data analysts have their own version of sticky notes when they're working in SQL.

They're called temporary tables and we're here to find out what they're all about.

A temporary table is a database table that is created and

exists temporarily on a database server.

Temp tables as we call them store subsets of data from standard data tables for

a certain period of time.

Then they're automatically deleted when you end your SQL database session.

Since temp tables aren't stored permanently, they're useful when you only

need a table for a short time to complete analysis tasks, like calculations.

For example, you might have a lot of tables you're performing calculations on

at the same time.

If you have a query that needs to join seven or eight of them,

you could join the two or three tables having the fewest number of rows and

store their output in a temp table.

You could then join this temp table to one of the other bigger tables.

Another example is when you have lots of different databases you're running queries

on. You can run these initial queries in each separate database, and

then use a temp table to collect the results of all of these queries.

The final report query would then run on the temporary table.

You might not be able to make use of this reporting structure without

temporary tables.

They're also useful if you've got a large number of records in a table and

you need to work with a small subset of those records repeatedly to complete

some calculations or other analysis.

So instead of filtering the data over and over to return the subset,

you can filter the data once and store it in a temporary table.

Then you can run your queries using a temporary table you've created.

Imagine that you've been asked to analyze data about the bike sharing system we

looked at earlier. You only need to analyze the data for

bike trips that were over 60 minutes or longer, but

you have several questions to answer about the specific data.

Play video starting at :2:11 and follow transcript2:11

Using a temporary table will let you run several queries about this data without

having to keep filtering it.

There's different ways to create temporary tables in SQL,

depending on the relational database management system you're using.

We'll explore some of these options soon.

For this scenario we'll use BigQuery. We'll apply a WITH clause to our query.

The WITH clause is a type of temporary table that you can query from

multiple times.

The WITH clause approximates a temporary table.

Basically, this means it creates something that does the same thing as a temporary

table. Even if it doesn't add a table to the database you're working in for

others to see, you can still see your results and

anyone who needs to review your work can see the code that led to your results.

Play video starting at :2:59 and follow transcript2:59

Let's get this query started.

We'll start this query with the WITH command.

Play video starting at :3:5 and follow transcript3:05

We'll then name our temp table trips, underscore, over,

underscore, 1, underscore, hr.

Then we'll type the AS command and an open parenthesis.

On a new line, we'll use the SELECT- FROM-WHERE structure for our subquery.

We'll type SELECT followed by an asterisk.

You might remember the asterisk means you're selecting all the columns in

the table.

Play video starting at :3:33 and follow transcript3:33

Now we'll type the FROM command and

name the database that we're pulling from bigquery, dash, public, dash,

data, dot, new, underscore, york, dot, citibike, underscore, trips.

Play video starting at :3:55 and follow transcript3:55

Next, we'll add a WHERE clause with the condition that the length of the bike

trips we need in our temp table are greater than or equal to 60 minutes.

In the query it goes like this:

trip duration, space, greater than sign, equal sign, space, 60.

Finally, we'll add a close parenthesis on a new line to end our subquery.

And that sets up our temporary table.

Now we can run queries that'll only return results for

trips that lasted 60 minutes or longer.

Let's try one.

Since we're working in our version of a temp table,

we don't need to open a new query.

Instead, we'll label our queries before we add our code to describe what we're doing.

For this query, we'll type two hashtags.

Play video starting at :4:42 and follow transcript4:42

This tells the server that this is a description and not part of the code.

Next, we'll add the query description.

Play video starting at :4:49 and follow transcript4:49

Count how many trips are 60 plus minutes long.

Play video starting at :4:59 and follow transcript4:59

And then we'll add our query. SELECT,

then on a new line COUNT with an asterisk in parentheses.

As followed by cnt to name the column with our COUNT.

Play video starting at :5:12 and follow transcript5:12

Next we'll add FROM and the name we're using for

our version of a temporary table: trips over one hour.

Play video starting at :5:21 and follow transcript5:21

When we run our query, the results show the total number of bike trips

from the dataset that lasted 60 minutes or longer,

Play video starting at :5:35 and follow transcript5:35

We can keep running queries on this temp table over and

over as long as we're looking to analyze bike trips that were 60 minutes and over.

And if you need to end your session and start a new runtime later,

most servers store the code used in temp tables.

You'll just need to recreate the table by running the code.

Play video starting at :5:55 and follow transcript5:55

When you use temporary tables, you make your own work more efficient. Naming and

using temp tables can help you deal with a lot of data in a more streamlined way,

so you don't get lost repeating query after query with the same code that you

could just include in a temp table.

And here's another bonus to using temp tables:

they can help your fellow team members too.

With temp tables your code is usually less complicated and easier to read and

understand which your team will appreciate!

Play video starting at :6:24 and follow transcript6:24

Once you start to explore temporary tables on your own,

you might not be able to stop.

Don't say I didn't warn you.

Coming up, we'll explore even more things you can do with temp tables.

See you soon.

Hello there. Earlier we

introduced you to temporary tables.

They're a great resource to

use during your analysis because they

help you keep your SQL code organized and efficient.

You learned how to use a WITH clause

to create a type of temporary table.

Now, we'll get into some other ways you can create

temp tables along with the pros and cons they present.

That's one of the great things about data analytics.

There's almost always more than one way

to get your analysis done.

The SELECT INTO statement is

a good example of how to get a temp table done.

This statement copies data from one table into

a new table but it doesn't

add the new table to the database.

It's useful if you want to make

a copy of a table with a specific condition,

like a query with a WHERE clause.

So far, we've been using

BigQuery to show you how SQL works.

But BigQuery doesn't currently

recognize the SELECT INTO command.

Instead, here's an example of how

a SELECT INTO statement might look in another RDBMS.

In the statement, a new table named Africa Sales is

created using the data from

the global sales database about the African region.

Using SELECT INTO is

a good practice when you want to keep

the database uncluttered and you

don't need other people using the table.

Now, if lots of people will be using the same table,

then the CREATE TABLE statement

might be the better option.

This statement does add the table into the database.

If everyone needs access to the Africa Sales table,

your query would start with CREATE TABLE,

followed by the same SELECT-FROM-WHERE

query as in the SELECT INTO statement.

In most relational database management systems or RDBMSs,

you can add metadata to describe

the data that's contained in the table you've created.

This can help make the table easier to

understand for anyone using it.

The CREATE TABLE statement is also

useful for tables that are more complex.

For example, if the code's difficult to replicate,

then making a temp table in this way

means it'll be safe for you to access later.

The way you create a temporary table

using the WITH clause or a

SELECT INTO or CREATE TABLE statement

is usually up to you and your needs.

The more you work in SQL,

the more you might have preferences as well,

especially since there's more than one way

to create temporary tables.

You may also find that you're working in

an RDBMS that uses a different syntax.

For example, you might need to use

a CREATE TEMP TABLE statement instead of CREATE TABLE.

Here's some good news.

The syntax that you need for

each unique RDBMS is usually

pretty easy to find with a quick online search.

But no matter how or where you create temporary tables,

there isn't much downside to them.

It's good to note though that sometimes building

a temp table can interrupt your workflow.

Again, that will depend on

your objectives and your preferences.

You can repeat your code over and over instead of making

a temp table but that usually leaves your queries

less readable and more vulnerable to typos.

As you continue exploring the world of data analytics,

you'll find that temporary tables are just

one of the many resources you'll be able to use.

The more you use them,

the easier it'll be to navigate that world.

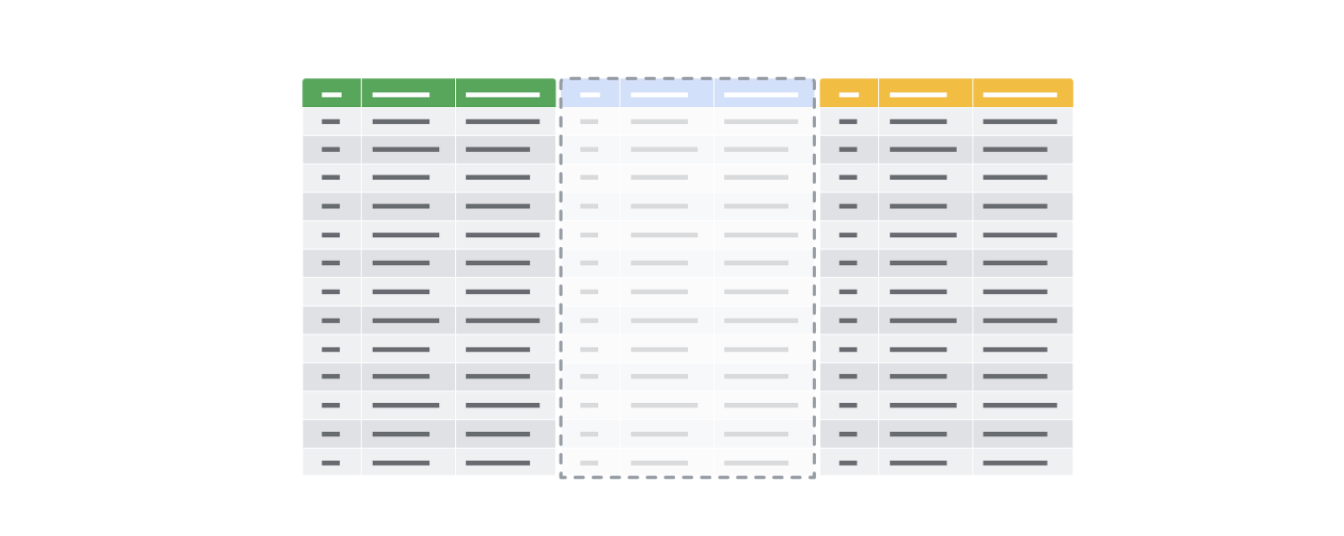
**Work with temporary tables**

**Temporary tables** are exactly what they sound like—temporary tables in a SQL database that aren’t stored permanently. In this reading, you will learn the methods to create temporary tables using SQL commands. You will also learn a few best practices to follow when working with temporary tables.

**A quick refresher on what you have already learned about temporary tables**

* They are automatically deleted from the database when you end your SQL session.
* They can be used as a holding area for storing values if you are making a series of calculations. This is sometimes referred to as **pre-processing** of the data.
* They can collect the results of multiple, separate queries. This is sometimes referred to as data **staging**. Staging is useful if you need to perform a query on the collected data or merge the collected data.
* They can store a filtered subset of the database. You don’t need to select and filter the data each time you work with it. In addition, using fewer SQL commands helps to keep your data clean.

It is important to point out that each database has its own unique set of commands to create and manage temporary tables. We have been working with BigQuery, so we will focus on the commands that work well in that environment. The rest of this reading will go over the ways to create temporary tables, primarily in BigQuery.



**Temporary table creation in BigQuery**

Temporary tables can be created using different clauses. In BigQuery, the **WITH** clause can be used to create a temporary table. The general syntax for this method is as follows:

1

2

3

4

5

6

7

8

WITH

new\_table\_data AS (

SELECT \*

FROM

Existing\_table

WHERE

Tripduration >=60

)

Breaking down this query a bit, notice the following:

* The statement begins with the **WITH** clause followed by the name of the new temporary table you want to create
* The **AS** clause appears after the name of the new table. This clause instructs the database to put all of the data identified in the next part of the statement into the new table.
* The opening parenthesis after the **AS** clause creates the subquery that filters the data from an existing table. The subquery is a regular **SELECT** statement along with a **WHERE** clause to specify the data to be filtered.
* The closing parenthesis ends the subquery created by the **AS** clause.

When the database executes this query, it will first complete the subquery and assign the values that result from that subquery to **new\_table\_data**, which is the temporary table. You can then run multiple queries on this filtered data without having to filter the data every time.

**Temporary table creation in other databases (not supported in BigQuery)**

The following method isn’t supported in BigQuery, but most other versions of SQL databases support it, including SQL Server and mySQL. Using **SELECT** and **INTO**, you can create a temporary table based on conditions defined by a **WHERE** clause to locate the information you need for the temporary table. The general syntax for this method is as follows:

1

2

3

4

5

6

7

8

SELECT

\*

INTO

AfricaSales

FROM

GlobalSales

WHERE

Region = "Africa"

This **SELECT** statement uses the standard clauses like **FROM** and **WHERE**, but the **INTO** clause tells the database to store the data that is being requested in a new temporary table named, in this case, **AfricaSales**.

**User-managed temporary table creation**

So far, we have explored ways of creating temporary tables that the database is responsible for managing. But, you can also create temporary tables that you can manage as a user. As an analyst, you might decide to create a temporary table for your analysis that you can manage yourself. You would use the **CREATE TABLE** statement to create this kind of temporary table. After you have finished working with the table, you would then delete or drop it from the database at the end of your session.

**Note:** BigQuery uses **CREATE TEMP TABLE** instead of **CREATE TABLE**, but the general syntax is the same.

1

2

3

4

5

6

CREATE TABLE table\_name (

    column1 datatype,

    column2 datatype,

    column3 datatype,

   ....

)

After you have completed working with your temporary table, you can remove the table from the database using the **DROP TABLE** clause. The general syntax is as follows:

1

DROP TABLE table\_name

**Best practices when working with temporary tables**

* **Global vs. local temporary tables:** Global temporary tables are made available to all database users and are deleted when all connections that use them have closed. Local temporary tables are made available only to the user whose query or connection established the temporary table. You will most likely be working with local temporary tables. If you have created a local temporary table and are the only person using it, you can drop the temporary table after you are done using it.
* **Dropping temporary tables after use:** Dropping a temporary table is a little different from deleting a temporary table. Dropping a temporary table not only removes the information contained in the rows of the table, but removes the table variable definitions (columns) themselves. Deleting a temporary table removes the rows of the table but leaves the table definition and columns ready to be used again. Although local temporary tables are dropped after you end your SQL session, it may not happen immediately. If a lot of processing is happening in the database, dropping your temporary tables after using them is a good practice to keep the database running smoothly.

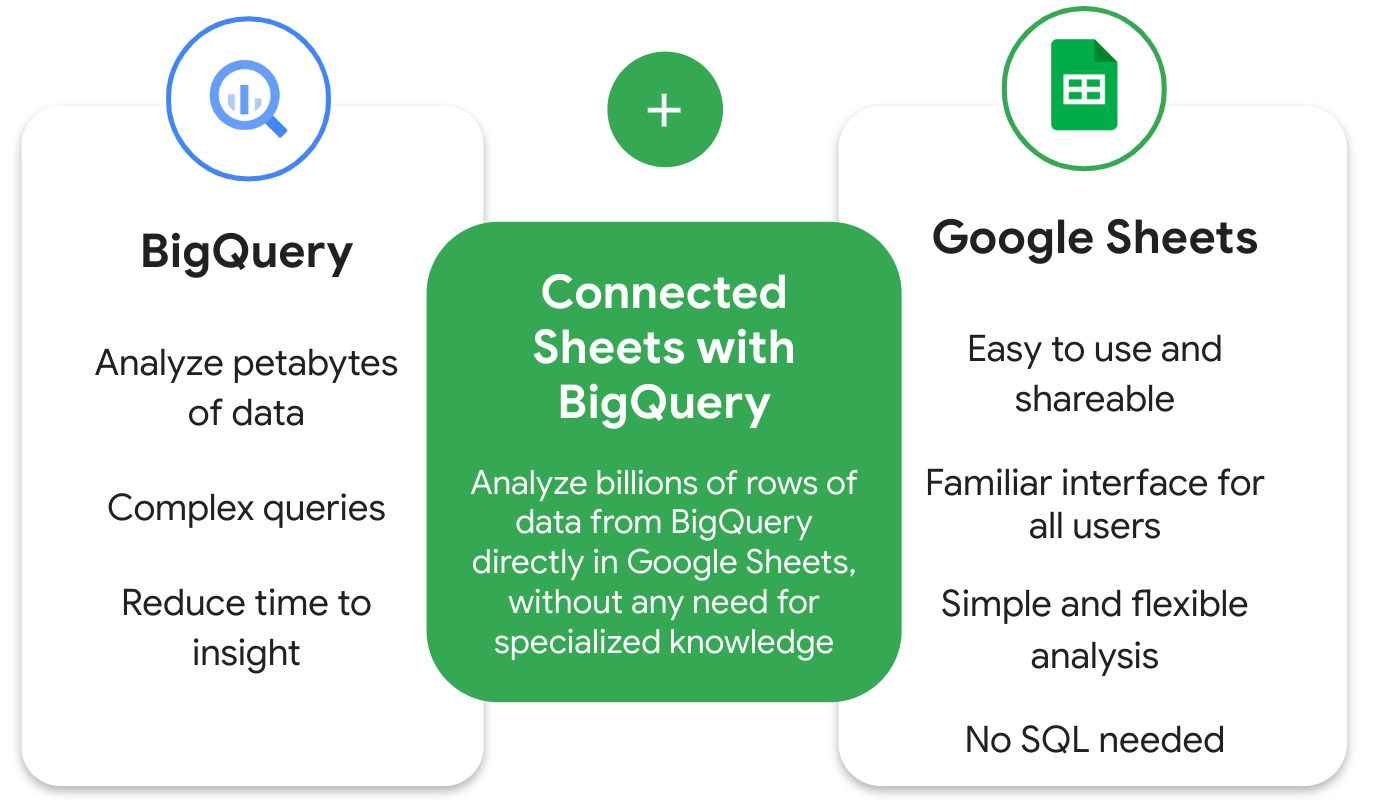
**For more information**

* [BigQuery Documentation for Temporary Tables](https://cloud.google.com/bigquery/docs/reference/standard-sql/data-definition-language#temporary_tables)**:** Documentation has the syntax to create temporary tables in BigQuery
* [How to use temporary tables via WITH in Google BigQuery](https://www.pascallandau.com/bigquery-snippets/use-temporary-tables-with-named-subquery/?utm_source=blog&utm_medium=rss&utm_campaign=development-feed)**:** Article describes how to use **WITH**
* [Introduction to Temporary Tables in SQL Server](https://codingsight.com/introduction-to-temporary-tables-in-sql-server/)**:** Article describes how to use **SELECT INTO** and **CREATE TABLE**
* [SQL Server Temporary Tables](https://www.sqlservertutorial.net/sql-server-basics/sql-server-temporary-tables/)**:** Article describes temporary table creation and removal
* [Choosing Between Table Variables and Temporary Tables](https://www.red-gate.com/hub/product-learning/sql-prompt/choosing-table-variables-temporary-tables)**:** Article describes the differences between passing variables in SQL statements vs. using temporary tables

**Use Connected Sheets with BigQuery**

In this reading, you will learn about Connected Sheets, a tool that allows data professionals to use basic spreadsheet functions to analyze large datasets housed in BigQuery. With Connected Sheets users don’t need to know SQL. Instead, anyone, not just data professionals, can generate insights with basic spreadsheet operations such as formulas, charts, and pivot tables.

**What is Connected Sheets?**



Recall that BigQuery allows users to analyze petabytes (a million gigabytes) of data using complex queries. A benefit of BigQuery is that it reduces the time needed to develop insights from large datasets.

Google Sheets, on the other hand, is a spreadsheet tool that is easy to use and shareable with a familiar interface. It also allows simple and flexible analysis with tools like pivot tables, charts, and formulas.

Connected Sheets integrates both BigQuery and Google Sheets, allowing the user to analyze billions of rows of data in Sheets without any need for specialized knowledge, such as SQL.

Additionally, Connected Sheets is built to handle big data. Users won’t experience the same limitations or performance issues they’ve had in the past (such as data loss) when working with large data sets in spreadsheets.

**Why would a data analytics professional use Connected Sheets?**

As a data analytics professional, Connected Sheets can help with several tasks, such as:

* Collaborating with partners, analysts, or other stakeholders in a familiar spreadsheet interface;
* Ensuring a single source of truth for data analysis without additional .csv exports;
* Defining variables so that all users are working with the same data;
* Sharing insights with your team in a secure environment; and
* Streamlining your reporting and dashboard workflows.

Many teams and industries benefit from Connected Sheets such as finance, marketing, and operations teams.

A few example use cases of Connected Sheets include:

* **Business planning:** A user can build and prepare datasets, and then find insights from the data. For example, a data analyst can analyze sales data to determine which products sell better in different locations.
* **Customer service:** A user can find out which stores have the most complaints per 10,000 customers.
* **Sales:** A user can create internal finance and sales reports. After completing, they can share revenue reports with sales reps.
* **Logistics, fulfillment, and delivery:** A user can run real-time inventory management and intelligent analytics tools.

**Connected Sheets benefits**

**Collaborate with teammates and stakeholders**

Since Connects Sheets lives in Google Workspace, you can easily collaborate with other teammates and stakeholders in your company. If you’d like to limit access, you also control permissions for who can view, edit, or share the data.

**Do more with familiar tools**

With Connected Sheets, you can access billions of rows of BigQuery data directly in Sheets. This direct access makes it easier for all employees to track, forecast, and analyze their data to get to better decisions faster.

**Easily visualize data**

You can unlock insights from your BigQuery datasets using features you’re already familiar with in Sheets, such as pivot tables, charts, and formulas. These features help visualize large datasets more easily than using a more advanced language such as SQL. However, if you know SQL, you may prefer to use it in certain situations.

**Up to date data**

With Connected Sheets, data professionals can ensure they are making decisions based on a single source of truth by setting up automatic refreshes of BigQuery data in Sheets.

**Less data integrity and security risk**

While users can access big data with Connected Sheets, they won’t be able to accidentally manipulate or jeopardize the integrity of the data. There’s less security risk because data isn’t stored on individual workstations, it’s stored in the cloud.

**Connected Sheets shortcomings**

**Limited free pricing tier**

A shortcoming of Connected Sheets is that for the free pricing tier, users only receive 1 terabyte (TB) of processed query data each month. To process more data, you will need to move to a paid tier.

**Data must be housed in BigQuery**

Another shortcoming is that you will need access to your data set in BigQuery. Without access to BigQuery, you won’t be able to analyze data in Connected Sheets.

**Query will fail with large results**

A third shortcoming is that the Connected Sheets query will fail if the results are too large. Your query will fail if your pivot table has a significant amount of results, which could be anywhere from 30,000 to 50,000. To reduce your results, you can use filters or limit the number of rows per breakout.

**Key takeaways**

Connected Sheets provides a tremendous opportunity to analyze large data sets without specialized skills like SQL. Use familiar spreadsheet skills such as pivot tables, charts, and formulas to analyze the data. For junior data analysts in particular, Connected Sheets can help them perform key tasks within BigQuery and increase their marketable skills.

**Resources for more information**

* [Get started with BigQuery data in Google Sheets](https://support.google.com/docs/answer/9702507)
* [Insights at scale with Google Sheets](https://www.youtube.com/watch?v=jMKxhOJogEE)
* [Connected Sheets product announcement](https://workspace.google.com/blog/product-announcements/connected-sheets-is-generally-available)

### 1.

Question 1

When working with a temporary table in a SQL database, at what point will the table be automatically deleted?

1 point

After ending the session in the SQL database

After completing all calculations in the table

After running a report from the table

After running the query in the SQL database

### 2.

Question 2

What data will appear in the temporary table created through this query?

1

2

3

4

5

WITH plant\_variety AS (

    SELECT \*

    FROM bigquery-public-data.plants.African\_species

    WHERE daily\_growth\_rate\_percentage = 0.05

)

1 point

A random subset of African plant species

Plant varieties that grow exactly 0.05 percent per day

Plant varieties that are equal to 0.05 inches tall

All plant species that exist in the public dataset

### 3.

Question 3

Fill in the blank: A data analyst uses \_\_\_\_\_ to copy data from one table into a temporary table without adding the new table to the database.

1 point

**SELECT INTO**

**TEMP**

**WITH**

**COPY TO**

### 4.

Question 4

Why might a data professional add a **CREATE TABLE** statement to a temporary table?

1 point

Create a second table within the temporary table

Include metadata about the data in the table

Automate calculations in the table

Give multiple people access to the table

# Glossary terms from module 4

## ****Terms and definitions for Course 5, Module 4****

**Array:** A collection of values in spreadsheet cells

**Calculated field:** A new field within a pivot table that carries out certain calculations based on the values of other fields

**Data security:** Protecting data from unauthorized access or corruption by adopting safety measures

**Data validation process:** The process of checking and rechecking the quality of data so that it is complete, accurate, secure and consistent

**GROUP BY:** A SQL clause that groups rows that have the same values from a table into summary rows

**Modulo:** An operator (%) that returns the remainder when one number is divided by another

**Profit margin:** A percentage that indicates how many cents of profit has been generated for each dollar of sale

**Summary table:** A table used to summarize statistical information about data

**SUMPRODUCT:** A function that multiplies arrays and returns the sum of those products

**Temporary table:** A database table that is created and exists temporarily on a database server

**Underscores**: Lines used to underline words and connect text characters

### 1.

Question 1

A data analyst at a beverage producer manually recalculates the new column carbonated\_bev. They want to identify any rows with values that do not match those in the original column, all\_bev. Which SQL clauses would enable them to do so? Select all that apply.

1 point

**WHERE all\_bev >< carbonated\_bev**

**WHERE all\_bev != carbonated\_bev**

**WHERE all\_bev !! carbonated\_bev**

**WHERE all\_bev <> carbonated\_bev**

### 2.

Question 2

Fill in the blank: To group table rows with the same values into\_\_\_\_\_, a data analyst uses the SQL command **GROUP BY**.

1 point

new columns

a temporary table

an aggregate table

summary rows

### 3.

Question 3

What will this spreadsheet function return?

**=SUMIF(H1:H50, ”>=50”, J1:J50)**

1 point

The sum of any values in cells H1 to H50and cells J1 to J50 that are greater than or equal to 50.

The sum of all values in cells J1 to J50 that correspond to values in cells H1 to H50 that are greater than or equal to 50.

The count of the number of cells in the array H1:H50 that have a value greater than or equal to 50.

The sum of all values in cells H1 to H50 for which the value in cells J1 to J50 is greater than or equal to 50.

### 4.

Question 4

Which of the following statements accurately describe pivot tables? Select all that apply.

1 point

The calculated field in a pivot table is a new field that carries out calculations based on the values of other fields.

The columns of a pivot table organize and group data horizontally.

A pivot table can be used to count, total, or average data.

The filters section of a pivot table is used to apply filters based on specific criteria.

### 5.

Question 5

A data analyst at a party planning business reviews attendee counts. They calculate the number of spreadsheet rows that contain values less than 500. Which function do they use?

1 point

**=COUNTIF(”<500”,L10:L300)**

**=COUNTIF(“>500”,L10:L300)**

**=COUNTIF(L10:L300,”<500”)**

**=COUNTIF(L10:L300,“>500”)**

### 6.

Question 6

Fill in the blank: The \_\_\_\_\_ statement is useful for making a copy of a table with a specific condition without adding the new table to the database.

1 point

**CREATE TABLE**

**SELECT INTO**

**DROP TABLE**

**WITH TEMP**

### 7.

Question 7

Which SQL statement will create a temporary table?

1 point

1

2

3

4

WITH my\_table AS (

    SELECT \*

    FROM other\_table WHERE x = 0

);

1

2

SELECT \*

FROM table;

1

2

3

4

WITH my\_table FROM (

    SELECT \*

    FROM other\_table

);

1

2

3

4

CREATE TABLE my\_table AS (

    SELECT \*

    FROM other\_table

);

### 8.

Question 8

Which column is set as a value in this pivot table?

|  |  |  |
| --- | --- | --- |
| MIN of Region | Products | (blank cell) |
| Region | Cat. | Avg. |
| Asia | (blank cell( | 1000 |
| Asia | 3500 | 1200 |
| North America | 1700 | 1600 |
| Europe | 2400 | (blank cell) |

1 point

Region

Products

Cat.

MIN

Glossary

Data Analytics

Terms and definitions from Course 5

A

Absolute reference: A reference within a function that is locked so that rows and columns

won’t change if the function is copied

Aggregation: The process of collecting or gathering many separate pieces into a whole

Aliasing: Temporarily naming a table or column in a query to make it easier to read and write

Array: A collection of values in spreadsheet cells

B

C

Calculated field: A new field within a pivot table that carries out certain calculations based on

the values of other fields

COUNT DISTINCT: A SQL function that only returns the distinct values in a specified range

D

Data aggregation: The process of gathering data from multiple sources and combining it into a

single, summarized collection

Data security: Protecting data from unauthorized access or corruption by adopting safety

measures

Data validation process: The process of checking and rechecking the quality of data so that it

is complete, accurate, secure and consistent

E

F

G

GROUP BY: A SQL clause that groups rows that have the same values from a table into

summary rows

H

I

INNER JOIN : A SQL function that returns records with matching values in both tables

J

JOIN: A SQL function that is used to combine rows from two or more tables based on a related

column

K

L

LEFT JOIN: A SQL function that will return all the records from the left table and only the

matching records from the right table

LIMIT: A SQL clause that specifies the maximum number of records returned in a query

M

MATCH: A spreadsheet function used to locate the position of a specific lookup value

Modulo: An operator (%) that returns the remainder when one number is divided by another

N

O

ORDER BY: A SQL clause that sorts results returned in a query

OUTER JOIN: A SQL function that combines RIGHT and LEFT JOIN to return all matching

records in both tables

P

Profit margin: A percentage that indicates how many cents of profit has been generated for

each dollar of sale

Q

R

RIGHT JOIN: A SQL function that will return all records from the right table and only the

matching records from the left.

ROUND: A SQL function that returns a number rounded to a certain number of decimal places

S

Subquery: A SQL query that is nested inside a larger query

Summary table: A table used to summarize statistical information about data

SUMPRODUCT: A function that multiplies arrays and returns the sum of those products

T

Temporary table: A database table that is created and exists temporarily on a database server

U

Underscores: Lines used to underline words and connect text characters

V

VALUE: A spreadsheet function that converts a text string that represents a number to a

numeric value

W

X

Y

Z

You completed another course. Congrats!

We still got some more to learn,

but you can always go back and

review what you've covered already.

Coming up, you'll meet my colleague Kevin.

Kevin can't wait to show you everything

the next part of the program has to offer.

And now that you're familiar with lots of

different ways to analyze data,

you'll be ready to share your analysis

results through visualizations and reports.

Up next, we'll focus on helping you develop

your data visualization skills with tools, like

Tableau, that'll help you get

your visualizations looking great.

You'll also learn how sharing your data

is really about telling the story

of your data and how it answers

the questions that prompted your analysis.

Finally, you'll learn what it takes to make

an effective presentation and how to

manage and respond to the questions.

Thanks so much for joining me in

exploring the world of data analytics.

You've made it this far,

so be proud and keep it

up as you move to the next course.