Week I

The analysis process

Analysis ✧.*
The process used to make sense of the data collected.

The goal of analysis is to identify trends and relationships within data so you can accurately answer the question you're asking

The 4 phases of analysis ✧.*
1. Organize data
2. Format and adjust data
3. Get input from others
4. Transform data by observing relationships between data poi8nts and making calculations

Always a need to organize

Sorting ✧.*
When you arrange data into a meaningful order to make it easier to understand, analyze, and visualize

Filtering ✧.*
Showing only the data that meets a specific criteria while hiding the rest

Sorting datasets

Sort sheet ✧.*
All of the data in a spreadsheet is sorted by the ranking of a specific sorted column - data across rows is kept together

Sort range ✧.*
Nothing else on the spreadsheet is rearranged besides the specified cells in a column

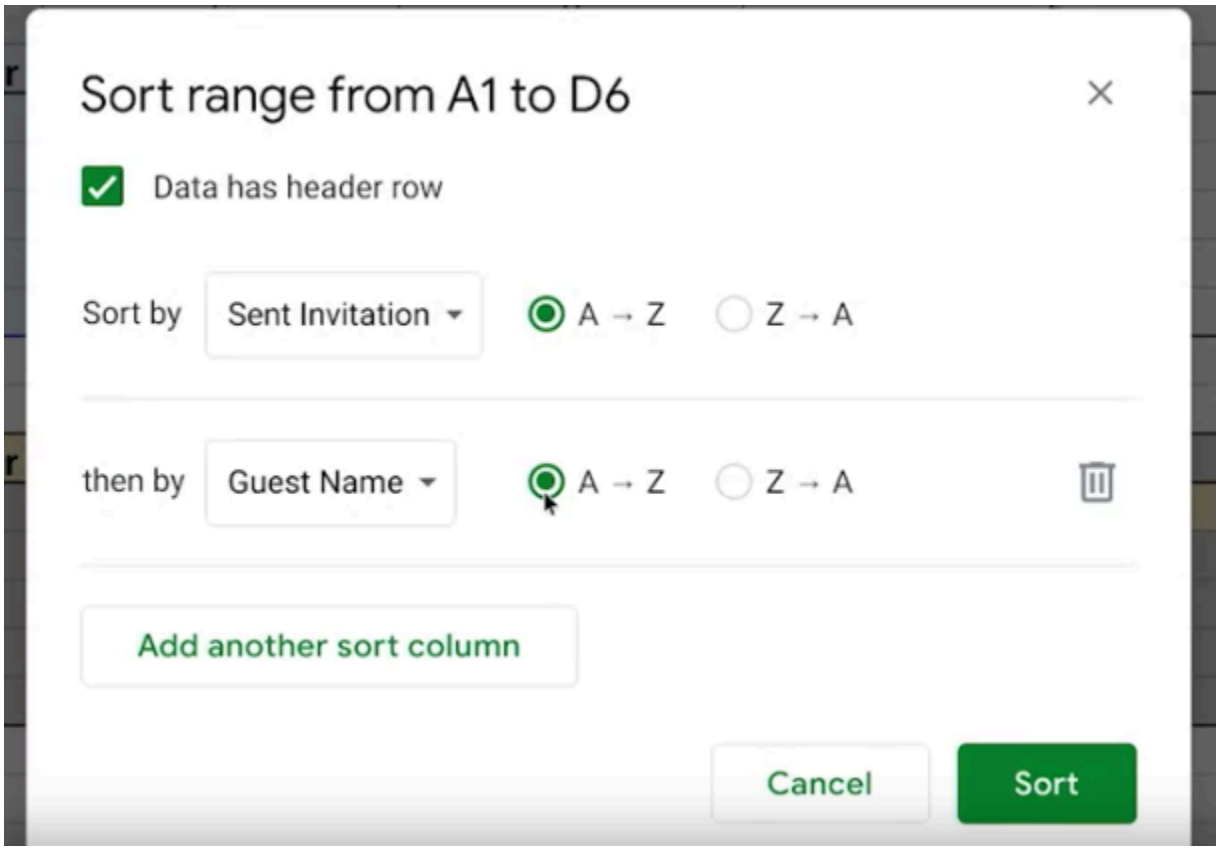
Sort function

Sort syntax ✧.*
=SORT(which cell:last cell, range which is the column number, true/false)

Ascending - true
Descending - false

Customized sort order ✧.*
When you sort data in a spreadsheet using multiple conditions

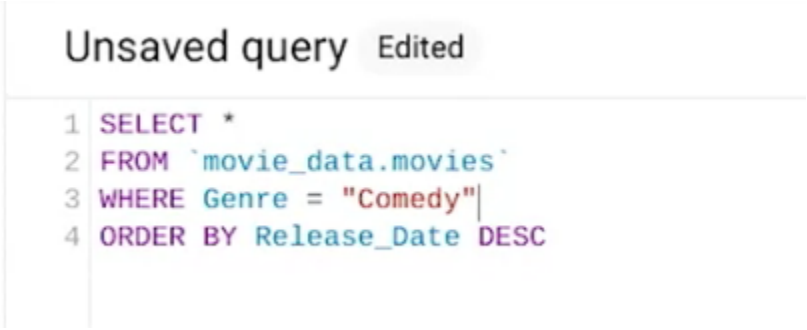
1. Highlight all the cells from the previous table. Click Data, then Sort range, then Advanced range sorting options.



	A	B	C	D	E
1	Guest Name	Table Number	Dietary Restriction	Sent Invitation	Row Index
2	Jack	2	Vegetarian	No	5
3	Nancy	3	None	No	4
4	Aida	3	None	Yes	3
5	Jianyu	2	Vegetarian	Yes	2
6	Omar	1	None	Yes	1
7					

ORDER BY ✧.*
It is a clause to sort results returned in a query. By default, it sorts data in ascending order. Use DESC for descending.

Asterisk (*) ✧.*
All columns are selected



Week II

From one type to another

Incorrectly formatted data can: * ✧.*
• Lead to mistakes
• Take time to fix
• Affect stakeholder's decision-making

Converting data in spreadsheets

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**: 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:** 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:** 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:** 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:** 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:** 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:** 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:** 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** for Excel or **Google Docs Editor Help** 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.

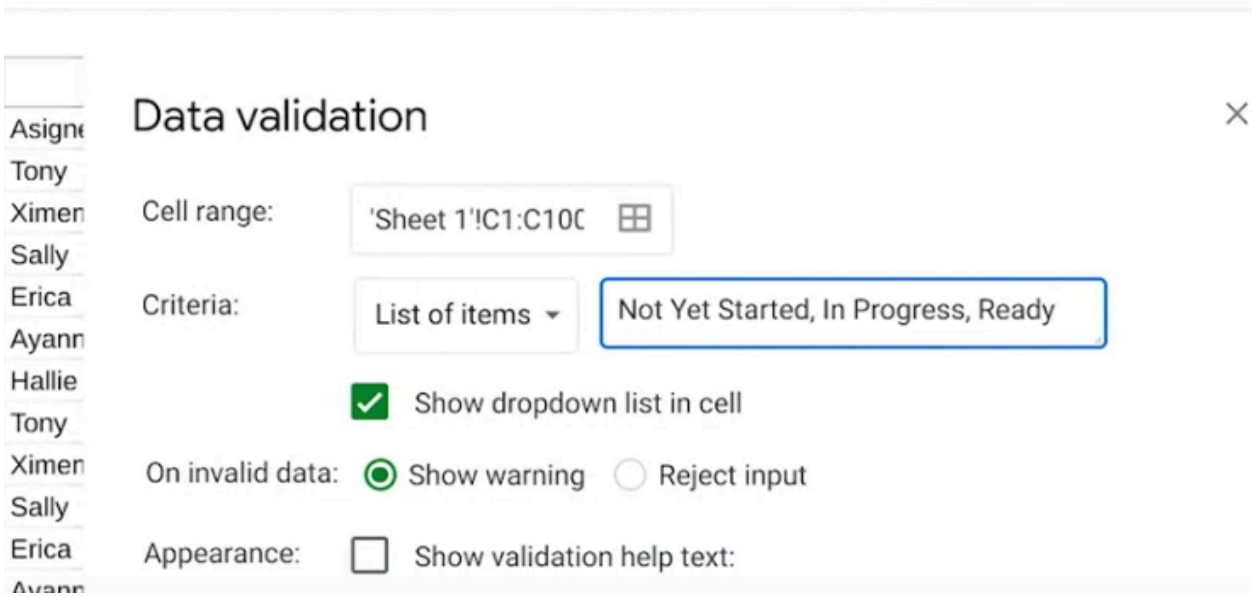
Data Validation

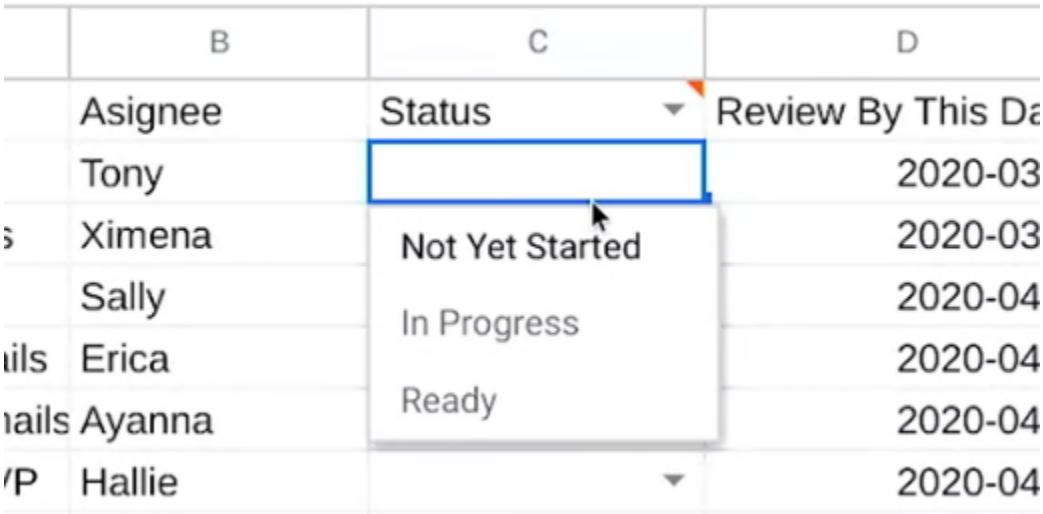
Data validation ✧.*

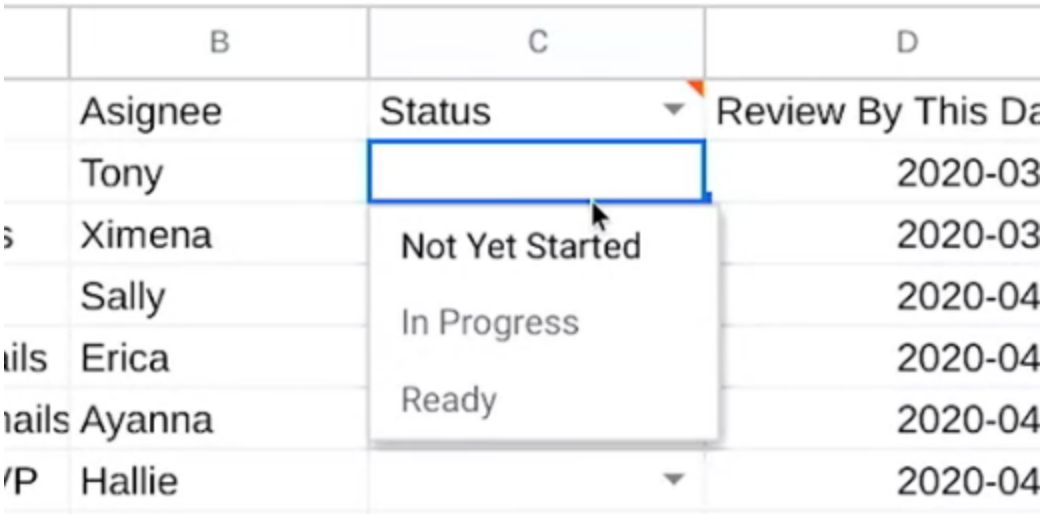
Allows you to control what can and can't be entered in your worksheet

- Add dropdown lists with predetermined options

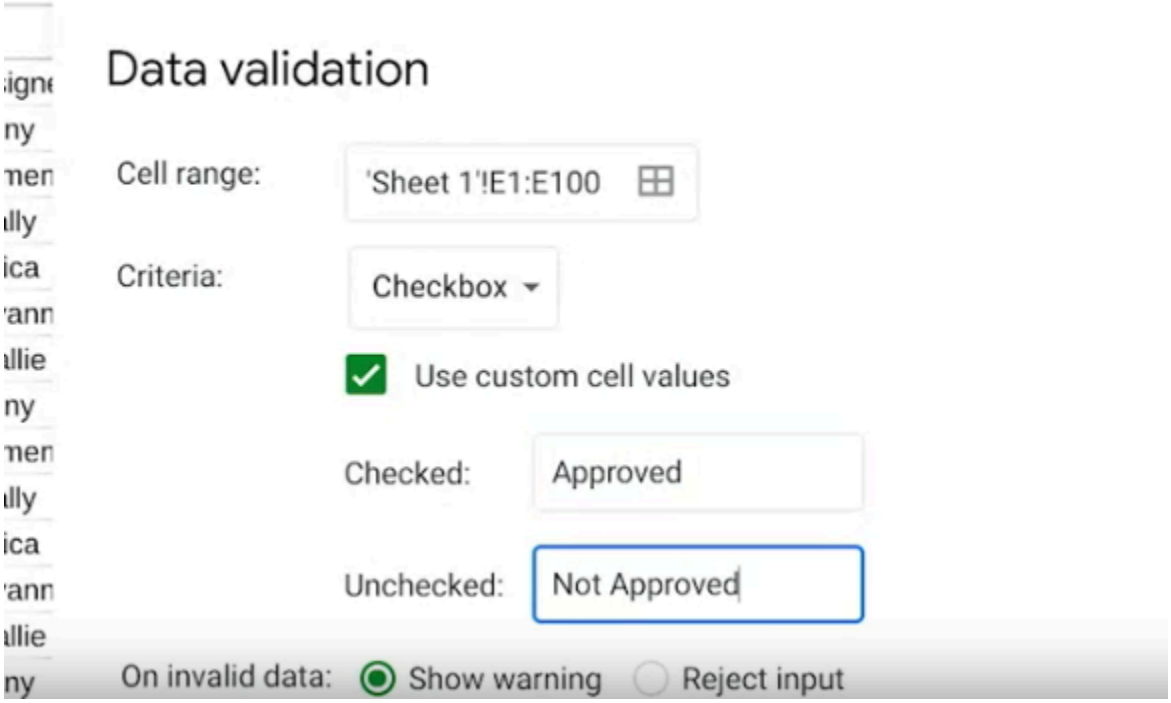
If you have a spreadsheet with a lot of collaborators, this can make it easier for them to interact with your table.

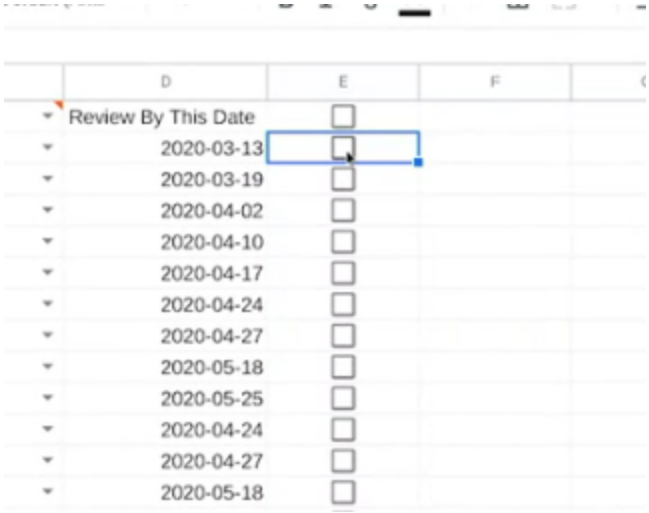






- Create custom checkboxes





	D	E	F
	Review By This Date	☐	
	2020-03-13	☐	
	2020-03-19	☐	
	2020-04-02	☐	
	2020-04-10	☐	
	2020-04-17	☐	
	2020-04-24	☐	
	2020-04-27	☐	
	2020-05-18	☐	
	2020-05-25	☐	
	2020-04-24	☐	
	2020-04-27	☐	
	2020-05-18	☐	

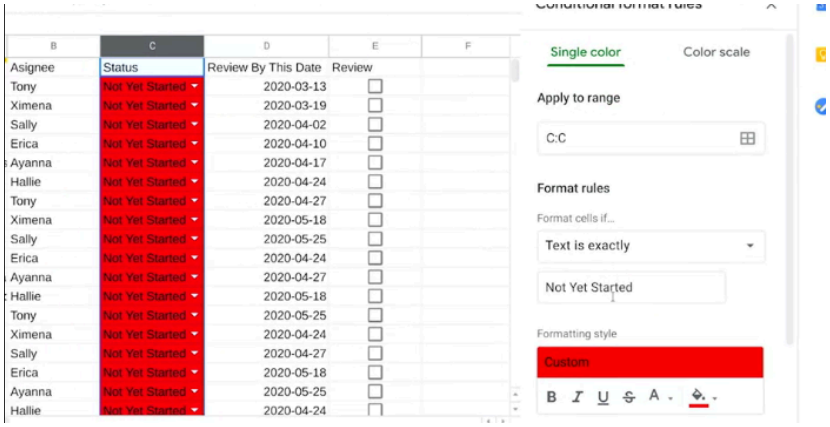
- Protect structured data and formulas - the data validation menu has an option to reject invalid inputs, which helps make sure our custom tools will continue to run correctly, even if someone puts the wrong data in by mistake.

Conditional Formatting

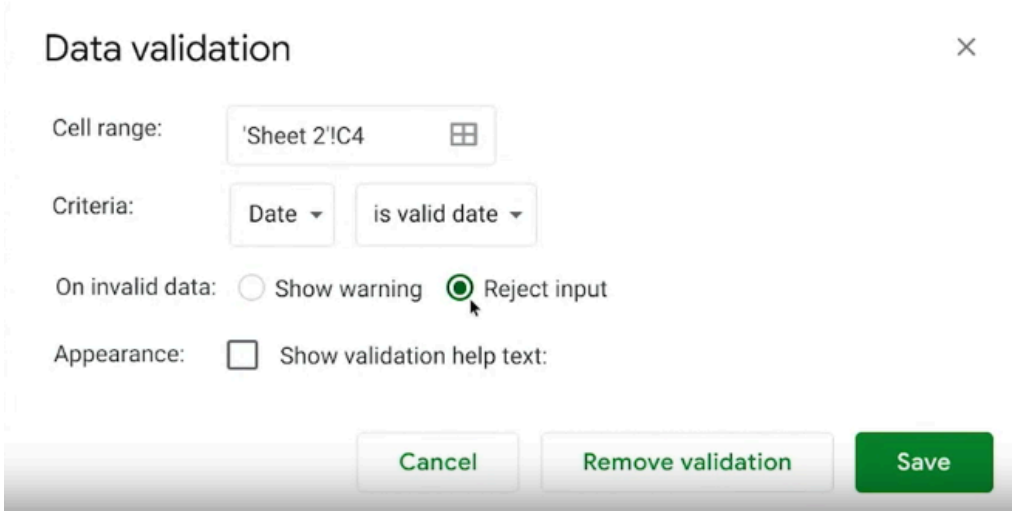
Conditional Formatting ✧.*
A spreadsheet tool that changes how cells appear when values meet specific conditions. This lets you add visual cues to your spreadsheets that make it easier to understand your table at a glance, and it makes the information in the worksheet clearer to your stakeholders.

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

- Dropdown Cell Color



- **Combination of Data Validation and Conditional Formatting**



Merging and multiple sources

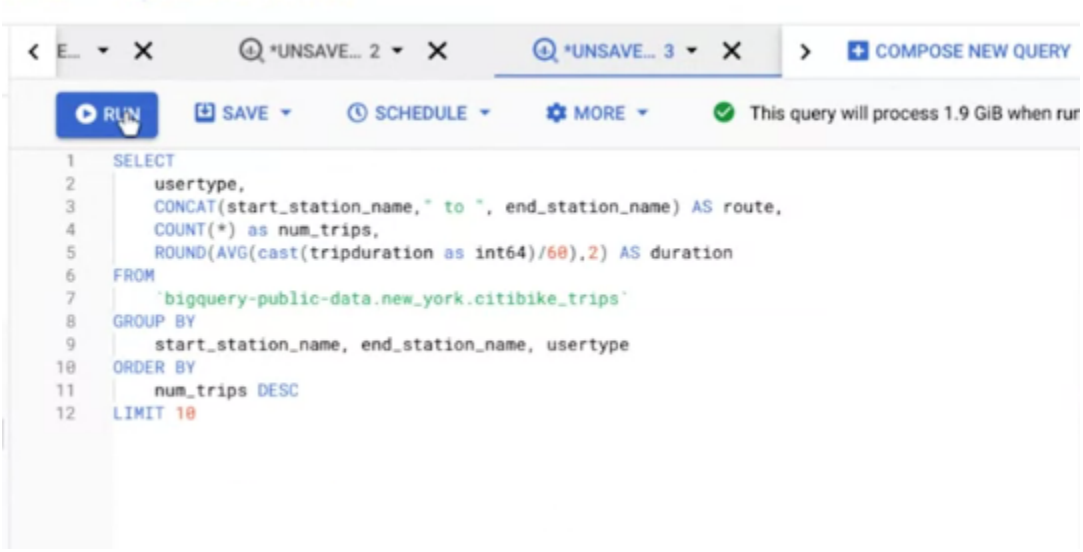
CONCATENATE ✧.*
a function that joins together two or more text strings.

Text string ✧.*
group of characters within a cell most often composed of letters.

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.

Openness ✧.*
A.k.a open data, is free access, usage, and sharing of data.

How to use CONCAT ✧.*



- 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.
- 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.
- 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.
- Divided by 60, which is the number of seconds in a minute. Dividing by 60 returns the output "duration" in minutes instead of seconds. 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.
- We'll use FROM and the location we're pulling it from.

Reminder: Make sure to use a backtick (`) instead of an apostrophe (') in the FROM statement.

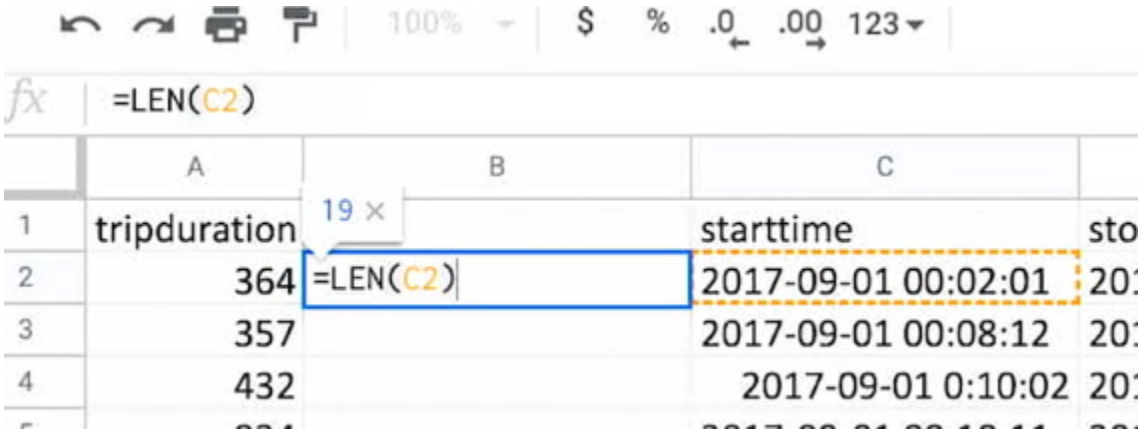
```
6   FROM
7 ✓ `bigquery-public-data.new_york.citibike_trips`
8
9 ✗ 'bigquery-public-data.new_york.citibike_trips'
```

-
- 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 in a column and use DESC to put it in descending order.
- Finally, we only want the top 10, so let's add LIMIT 10.

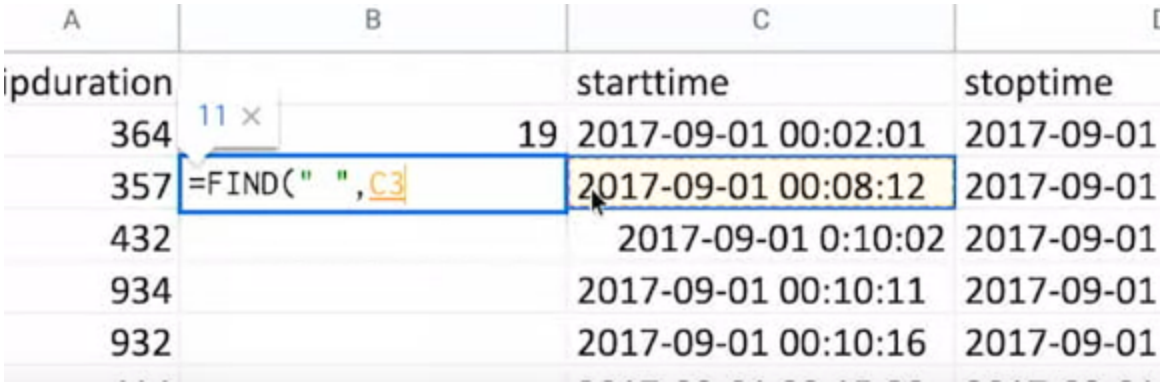
Strings in spreadsheets

LEN ✧.*
Tell us the length of the string.

To start, we'll input the first part of the formula. And then we'll just select one of the cells with the datetime string in it.

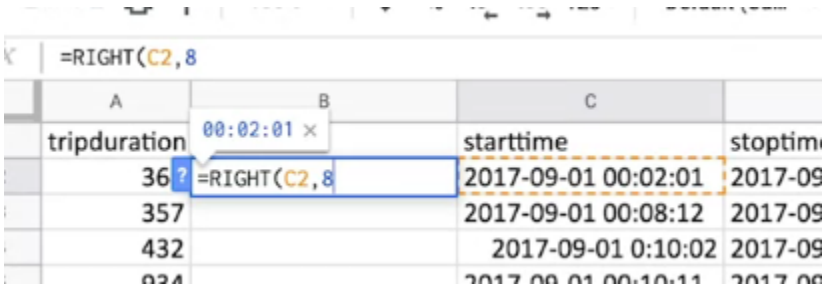


FIND ✧.*
So if you're using FIND to pull a substring, make sure that you've input the substring correctly. We notice that all of the datetime strings have a space separating the date and the timestamp. So we can actually use FIND to figure out where the date ends. Okay, seems like the space is the 11th character in this string. So the timestamp substring will start at character 12.



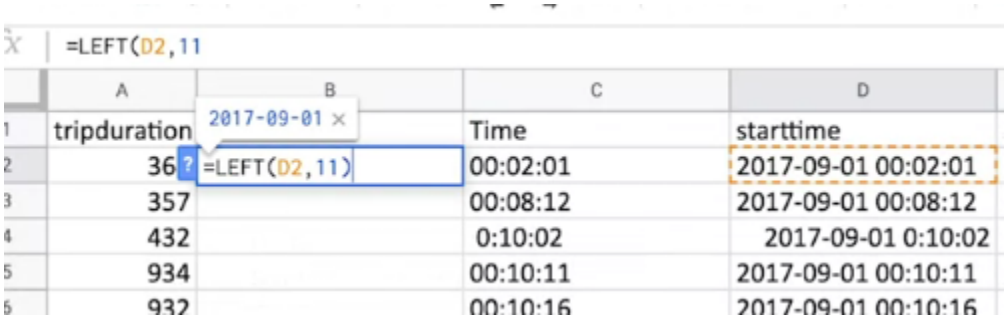
RIGHT ✧.*

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.



LEFT ✧.*

We can use the LEFT and RIGHT functions to select which parts of the string we want to isolate in a new column. 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.



SQL Syntax

Convert ✧.*

Change the unit of measurement of a value in data

Concat ✧.*
Add strings together to create new text strings that can be used as unique keys

Join ✧.*
Combine rows from two or more tables based on a related column

Limit ✧.*
Return a certain number of records

Round ✧.*
Limit records to a certain number of decimal places

LEN ✧.*
Return the length of a string of text by counting the number of characters it contains

RIGHT ✧.*
Return a set number of characters from the right side of a text string

FIND ✧.*
Locate specific characters in a string

LEFT ✧.*
Return a set number of characters from the left side of a text string

What to do when you get stuck

IF Formula ✧.*
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.
FORMULA: *=IF(end>start, end-start, 24+end-start)*

MOD Formula ✧.*
This flips the negative values into positive ones, solving your calculation problem.
FORMULA: *=MOD(end-start,1)*

Running into challenges? Not to worry!

Best practices for searing online ✧.*

- **Thinking skills** - From analytical, to mathematical, to structured thinking. 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.

Mental model ✧.*

Your thought process and the way you approach a problem

- **Data analytics terms -** Use the right terms. 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.
- **Basic knowledge of tools -** 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.

When to use which tool

- **Pivot Table -** 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
- **SQL -** 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.
- **R -** 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.

Week III

Aggregate data for analysis

Aggregation ✧.*

Collecting or gathering many separate pieces into a whole

Data Aggregation ✧.*
The process of gathering data from multiple sources in order to combine it into a single summarized collection. It helps data analysts to:

- Identify trends
- Make comparisons
- Gain insights

Summarized collection ✧.*
or summary, describes identifying the data you need and gathering it all together in one place.

• Puzzle pieces = data
• Organization = aggregation
• Pile of pieces = summary
• Putting the pieces together = gaining insights

Data can also be aggregated over a given time period to provide statistics such as:
• Averages
• Minimums
• Maximums
• Sums

- Functions help make data aggregation possible

Subquery ✧.*
A.k.a inner or nested query, is a query within another query

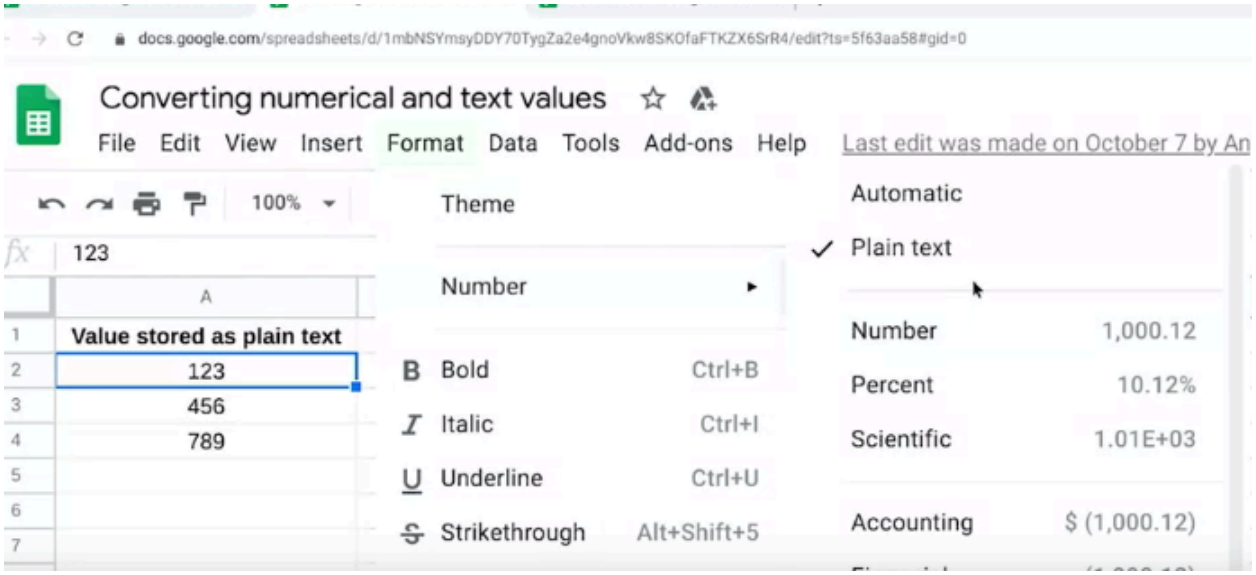
Preparing for VLOOKUP

Data Aggregation ✧.*
The process of gathering data from multiple sources in order to combine it into a single summarized collection

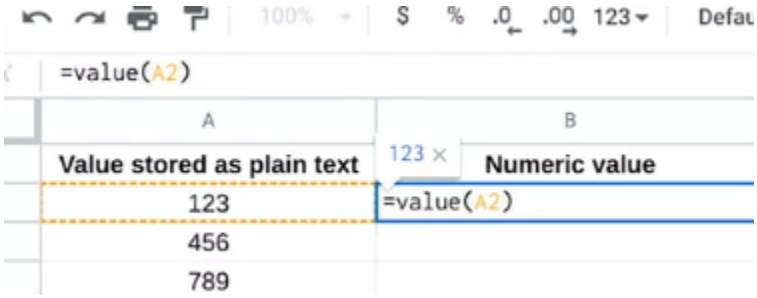
Vertical Lookup (VLOOKUP) ✧.*
A function that searches for a certain value in a column to return a corresponding piece of information

- **Converting text to numerical value**

To do this, you could use the Format menu to select a type of number, but you could also use the VALUE function.

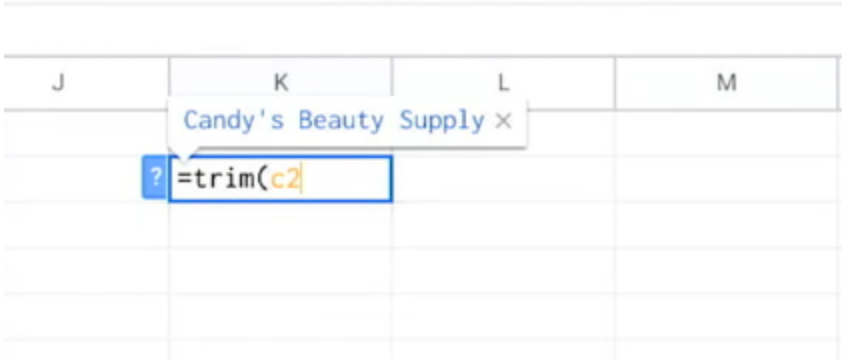


VALUE ✧.*
a function that converts a text string that represents a number to a numerical value.



- **Having extra spaces**

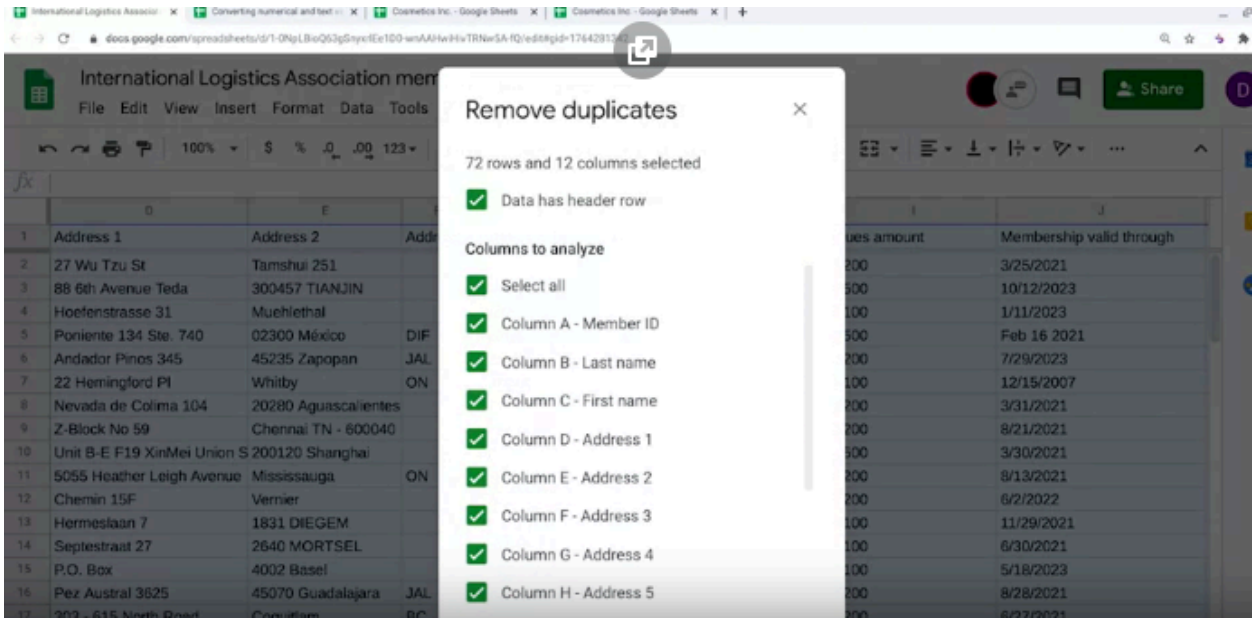
TRIM ✧.*
automatically deletes any extra spaces added to the cell.



- **Duplicates**

Remove Duplicates ✧.*

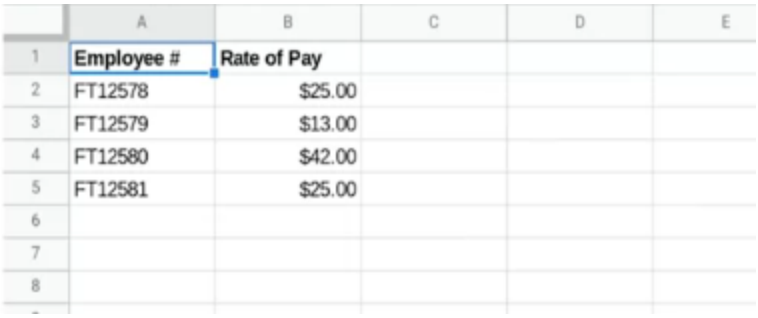
A tool that automatically searches for and eliminates duplicate entries from a spreadsheet



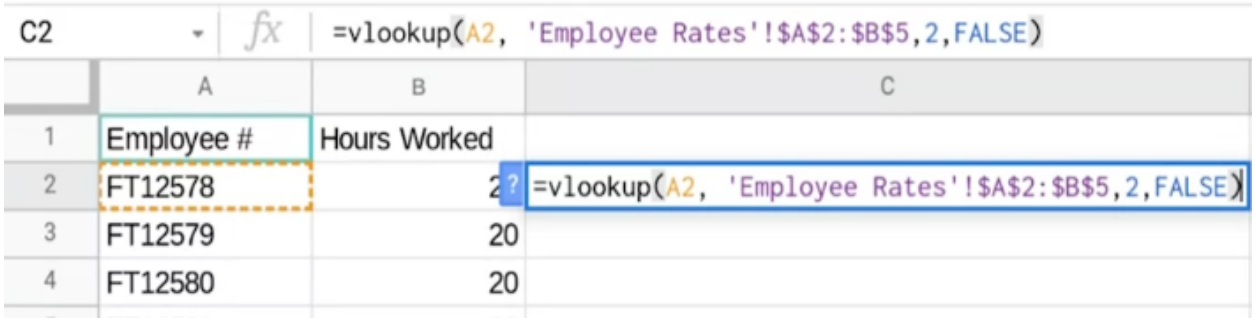
VLOOKUP in action

=VLOOKUP(103, A2:B26, 2, FALSE)

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.



	A	B	C	D	E
1	Employee #	Rate of Pay			
2	FT12578	$25.00			
3	FT12579	$13.00			
4	FT12580	$42.00			
5	FT12581	$25.00			
6					
7					
8					



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.

Identifying common VLOOKUP errors

Troubleshooting ✧.*

Has to do with asking the right questions. 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.

- How should I prioritize these issues?
- In a single sentence, what's the issue I'm facing?
- What resources can help me solve the problem?

- How can I stop this problem from happening in the future?

VLOOKUP only returns the first match it finds

Absolute reference ✧.*
A reference that is locked so that rows and columns won't change when copied

MATCH ✧.*
a function used to locate the position of a specific lookup value and can help you with version control.

TRUE tells VLOOKUP to look for approximate matches

FALSE tells VLOOKUP to look for exact matches

VLOOKUP core concepts

Functions can be used to quickly find information and perform calculations using specific values. In this reading, you will learn about the importance of one such function, VLOOKUP, or Vertical Lookup, which searches for a certain value in a spreadsheet column and returns a corresponding piece of information from the row in which the searched value is found.

When do you need to use VLOOKUP?

Two common reasons to use VLOOKUP are:

- Populating data in a spreadsheet
- Merging data from one spreadsheet with data in another

VLOOKUP syntax

A VLOOKUP function is available in both Microsoft Excel and Google Sheets. You will be introduced to the general syntax in Google Sheets. (You can refer to the resources at the end of this reading for more information about VLOOKUP in Microsoft Excel.)

```
VLOOKUP(10003, A2:B26, 2, FALSE)
```

Here is the syntax.

```
VLOOKUP(search_key, range, index, [is_sorted])
```

search_key

- The value to search for.
- For example, 42, "Cats", or I24.

range

- The range to consider for the search.
- The first column in the range is searched to locate data matching the value specified by search_key.

index

- The column index of the value to be returned, where the first column in range is numbered 1.
- If index is not between 1 and the number of columns in range, #VALUE! is returned.

is_sorted

- Indicates whether the column to be searched (the first column of the specified range) is sorted. TRUE by default.
- It's recommended to set is_sorted to FALSE. If set to FALSE, an exact match is returned. If there are multiple matching values, the content of the cell corresponding to the first value found is returned, and #N/A is returned if no such value is found.
- If is_sorted is TRUE or omitted, the nearest match (less than or equal to the search key) is returned. If all values in the search column are greater than the search key, #N/A is returned.

What if you get #N/A?

As you have just read, #N/A indicates that a matching value can't be returned as a result of the VLOOKUP. The error doesn't mean that anything is actually wrong with the data, but people might have questions if they see the error in a report. You can use the **IFNA** function to replace the #N/A error with something more descriptive, like "Does not exist."

```
IFNA(#N/A, "Does not exist")
```

Here is the syntax.

```
IFNA(value, value_if_na)
```

value

- This is a required value.
- The function checks if the cell value matches the value; such as #N/A.

value_if_na

- This is a required value.

- The function returns this value if the cell value matches the value in the first argument; it returns this value when the cell value is #N/A.

Helpful VLOOKUP reminders

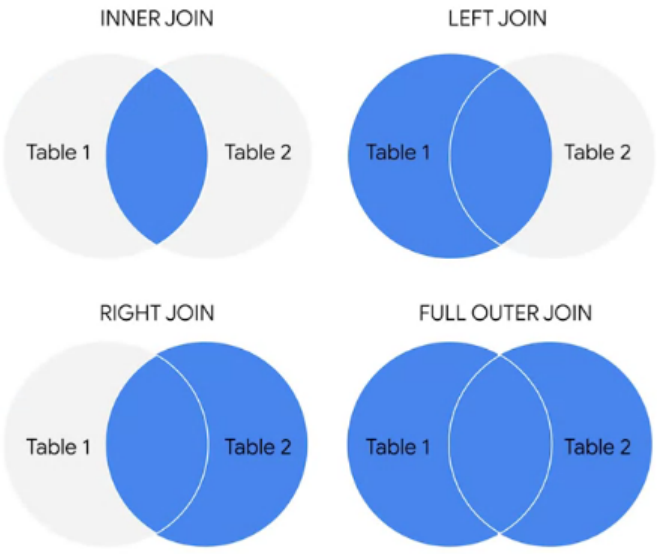
- TRUE means an approximate match, FALSE means an exact match on the search key. If the data used for the search key is sorted, TRUE can be used.
- You want the column that matches the search key in a VLOOKUP formula to be on the left side of the data. VLOOKUP only looks at data to the right after a match is found. In other words, the index for VLOOKUP indicates columns to the right only. This may require you to move columns around before you use VLOOKUP.
- After you have populated data with the VLOOKUP formula, you may copy and paste the data as values only to remove the formulas so you can manipulate the data again.

Understanding JOINS

JOIN ✧.*
A SQL clause that is used to combine rows from two or more tables based on a related column

Common JOINs

- **Innter**
- **Left**
- **Right**
- **Outer**

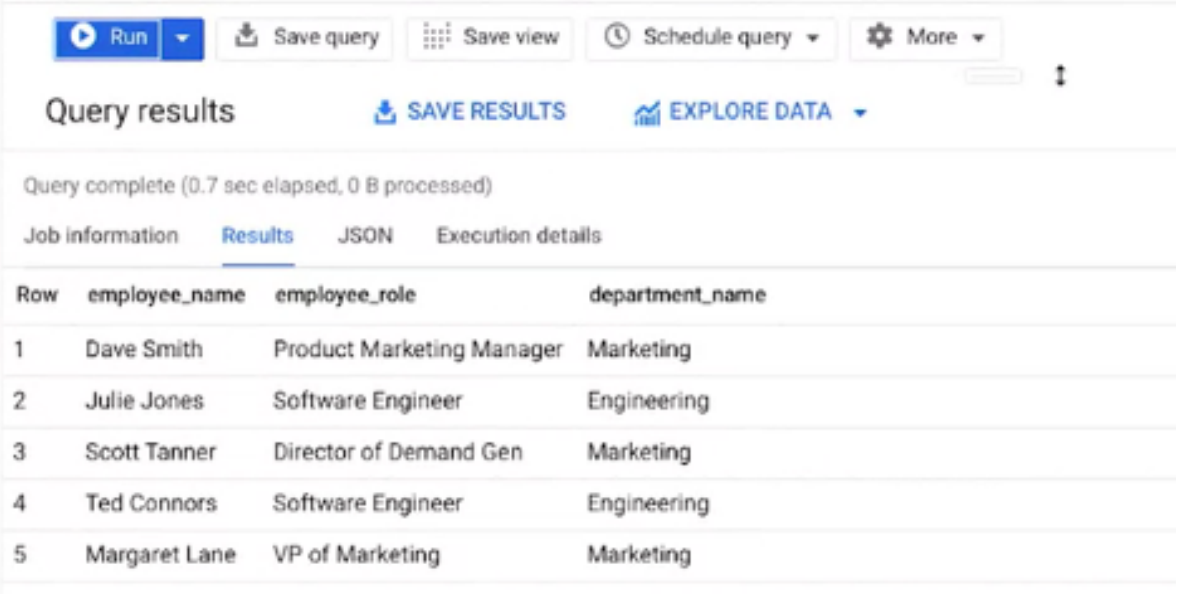


Primary keys reference columns in which each value is unique

Foreign keys are primary keys in other tables

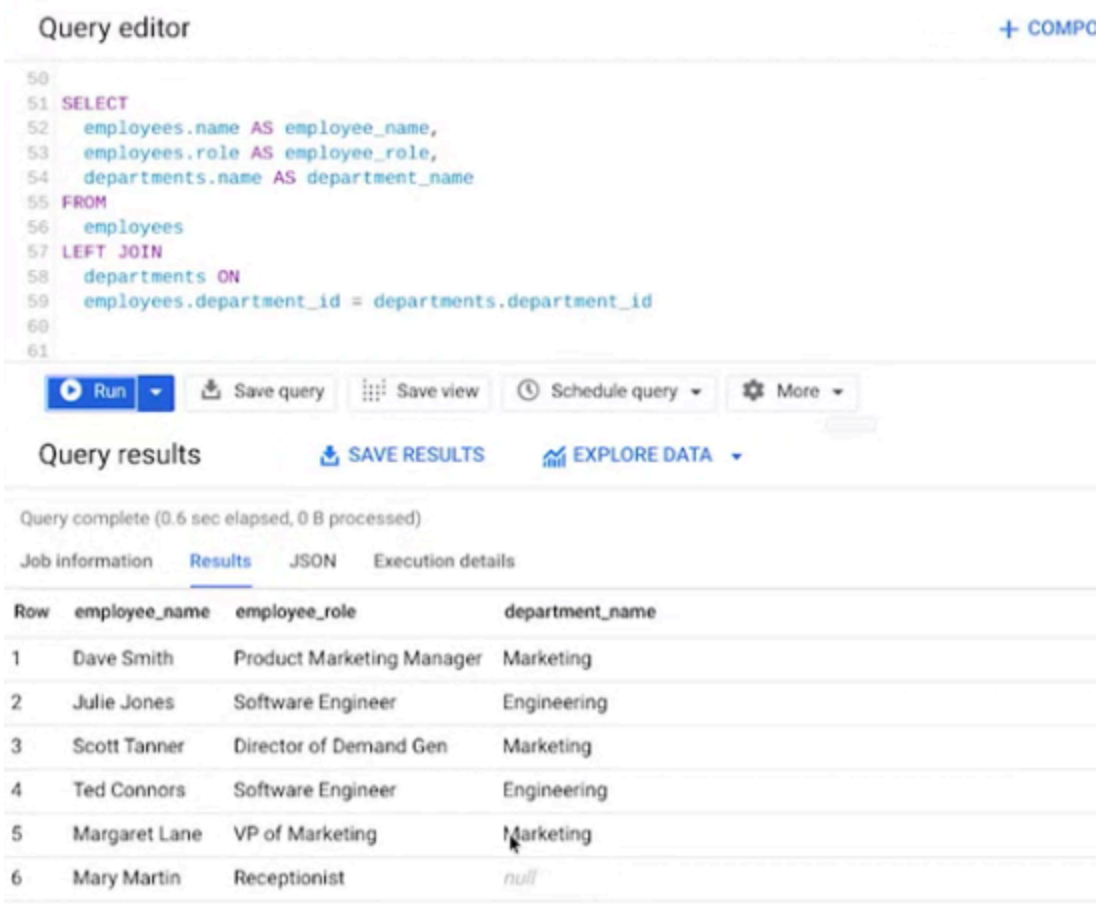
INNER JOIN ✧.*
A function that returns records with matching values in both tables. Default

```
SELECT
  employees.name AS employee_name,
  employees.role AS employee_role,
  departments.name AS department_name
FROM
  employees
INNER JOIN
  departments ON
  employees.department_id = departments.department_id
```



Row	employee_name	employee_role	department_name
1	Dave Smith	Product Marketing Manager	Marketing
2	Julie Jones	Software Engineer	Engineering
3	Scott Tanner	Director of Demand Gen	Marketing
4	Ted Connors	Software Engineer	Engineering
5	Margaret Lane	VP of Marketing	Marketing

LEFT JOIN ✧.*
A function that will return all the records from the left table and only the matching records from the right table



The table mentioned first is left, and the table mentioned second is right

RIGHT JOIN ✧.*

A function that will return all records from the right table and only the matching records from the left

Query results SAVE RESULTS EXPLORE DATA

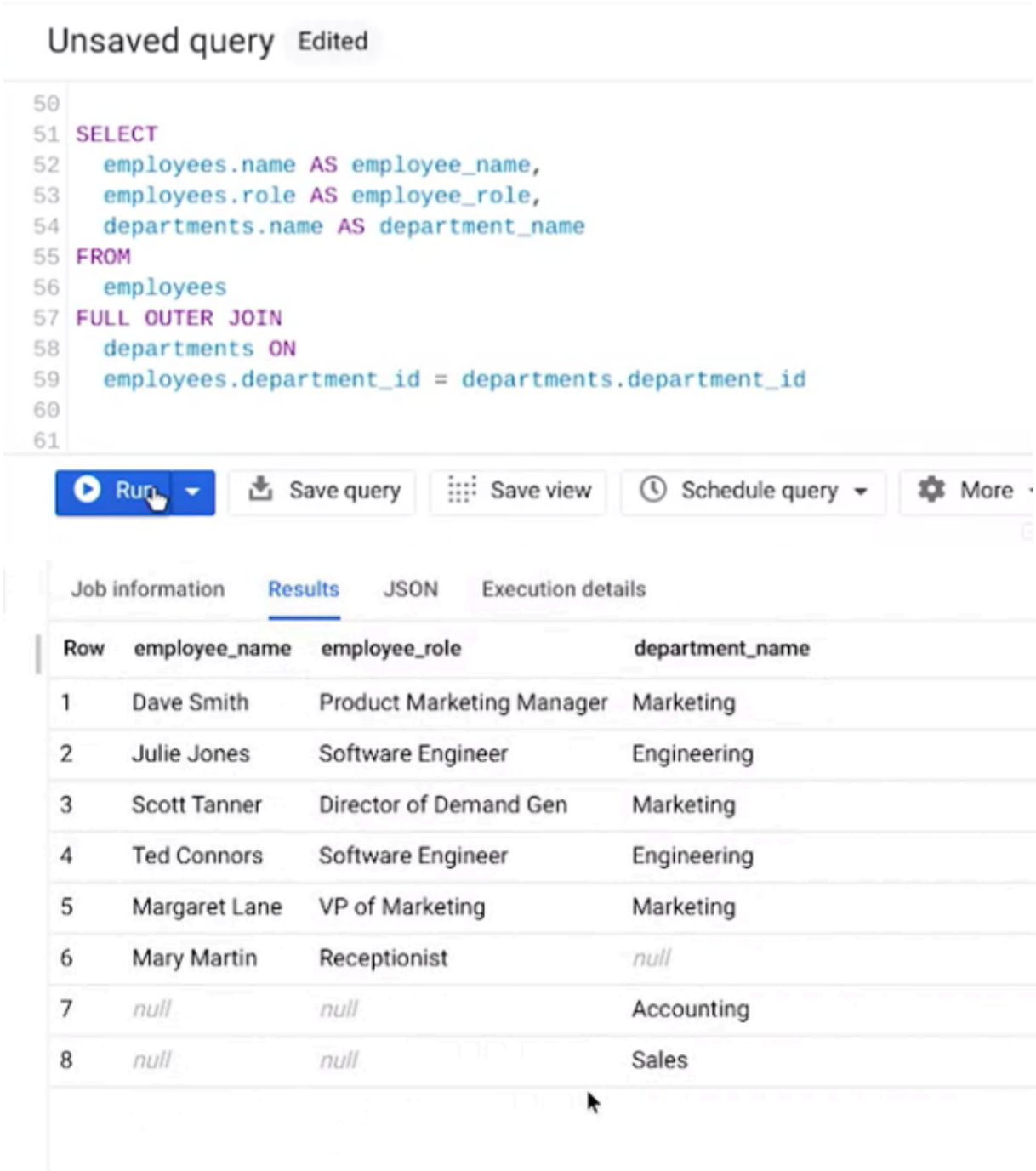
Query complete (0.6 sec elapsed, 0 B processed)

Job information | Results | JSON | Execution details

Row	employee_name	employee_role	department_name
1	Dave Smith	Product Marketing Manager	Marketing
2	Julie Jones	Software Engineer	Engineering
3	Scott Tanner	Director of Demand Gen	Marketing
4	Ted Connors	Software Engineer	Engineering
5	Margaret Lane	VP of Marketing	Marketing
6	*null*	*null*	Accounting
7	*null*	*null*	Sales

OUTER JOIN ✧.*

A function that combines RIGHT and LEFT JOIN to return all matching records in both tables



Secret identities: The importance of aliases

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 retype 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:

```
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:

```
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:

```
FROM work_day.employees AS employees
```

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.

Using JOINs effectively

A **JOIN** combines tables by using a primary or foreign key to align the information coming from both tables in the combination process. JOINs 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 for this course, or go back to Databases in data analytics.

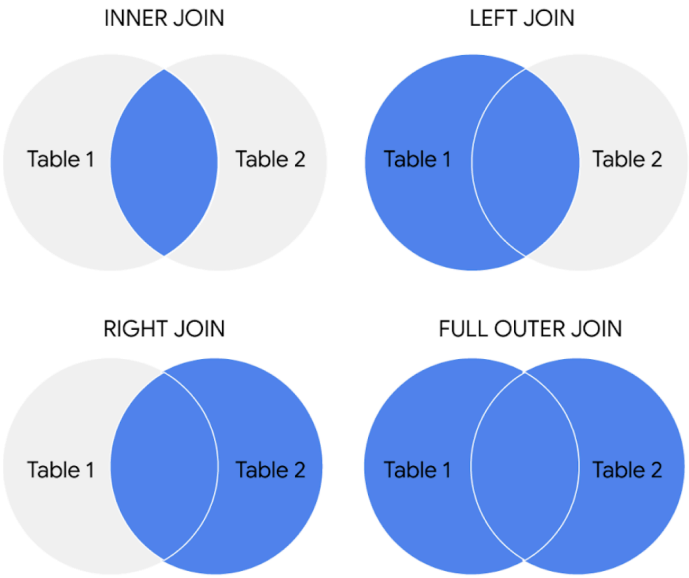
The general JOIN syntax

```
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 JOINs 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.

```
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.

```
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:

```
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.

```
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.

COUNT and COUNT DISTINCT

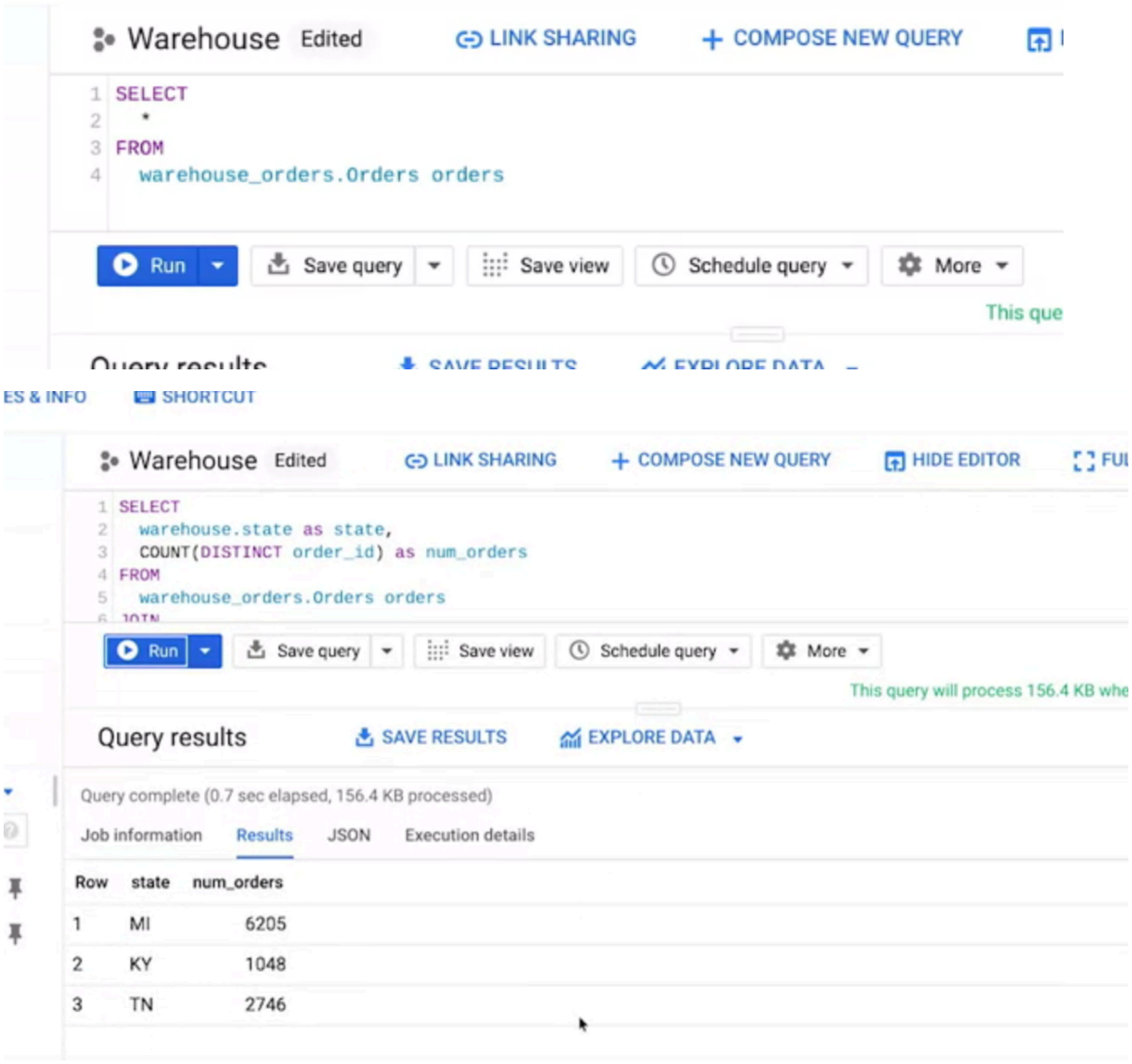
COUNT in spreadsheets ✧.*
Can be used to count the total number of numerical values within a specific range in spreadsheets

COUNT in SQL ✧.*
A query that returns the number of rows in a specified range

COUNT DISTINCT ✧.*
A query that only returns the distinct values in a specified range

You'll use COUNT and COUNT DISTINCT any time you want to answer questions about "how many"

Aliasing ✧.*
When you temporarily name a table or column in your query to make it easier to read and write



Queries within queries

Subquery ✧.*
A SQL query that is nested inside a larger query

Outer query ✧.*
A.k.a outer select, this is the statement containing the subquery. 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.

Using subqueries to aggregate data

HAVING ✧.*
Allows you to add a filter to your query instead of the underlying table that can only be used with aggregate functions

CASE ✧.*
Returns records with your conditions by allowing you to include if/then statements in your query

SQL functions and subqueries: A functional friendship

How do SQL functions, function?

SQL functions are what help make data aggregation possible. (As a reminder, data aggregation is the process of gathering data from multiple sources in order to combine it into a single, summarized collection.) So, how do SQL functions work? Going back to W3Schools, let's review some of these functions to get a better understanding of how to run these queries:

- **SQL HAVING:** This is an overview of the HAVING clause, including what it is and a tutorial on how and when it works.
- **SQL CASE:** Explore the usage of the CASE statement and examples of how it works.
- **SQL IF:** This is a tutorial of the IF function and offers examples that you can practice with.
- **SQL COUNT:** The COUNT function is just as important as all the rest, and this tutorial offers multiple examples to review.

Subqueries - the cherry on top

Think of a query as a cake. A cake can have multiple layers contained within it and even layers within those layers. Each of these layers are our subqueries, and when you put all of the layers together, you get a cake (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 is as follows:

```
SELECT account_table.*
  FROM (
         SELECT *
           FROM transaction.sf_model_feature_2014_01
          WHERE day_of_week = 'Friday'
        ) account_table
 WHERE account_table.availability = 'YES'
```

You will find that, within the first SELECT clause is another SELECT clause. The second SELECT clause marks the start of the subquery in this statement. There are many different ways in which you can make use of subqueries, and resources referenced will provide additional guidance as you learn. But first, let's recap the subquery rules.
There are a few rules that subqueries must follow:

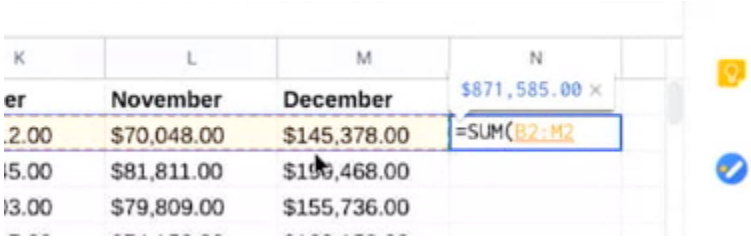
- Subqueries must be enclosed within parentheses
- A subquery can have only one column specified in the SELECT clause. But if you want a subquery to compare multiple columns, those columns must be selected in the main query.
- Subqueries that return more than one row can only be used with multiple value 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.

Week IV

Common calculation formulas

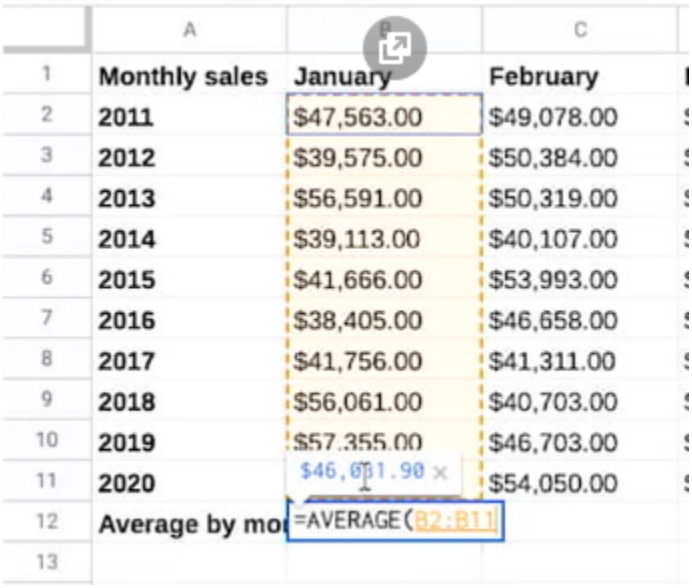
SUM function ✧.*

=SUM(cell reference first:last)



AVERAGE function ✧.*

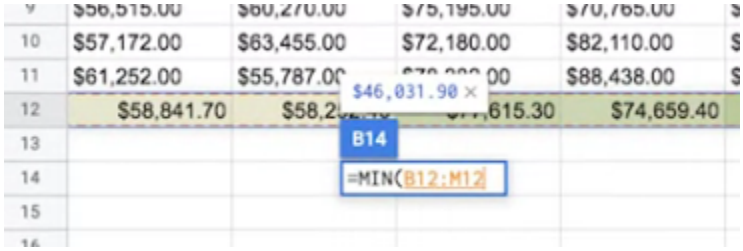
AVERAGE function ✧.*



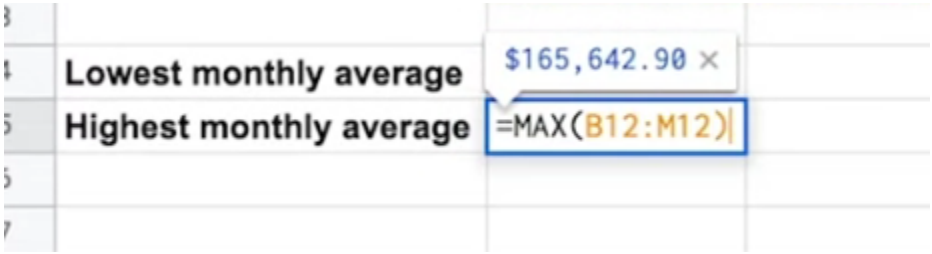
Conditional formatting ✧.*

A spreadsheet tool that changes how cells appear when values meet specific conditions

MIN function ✧.*



MAX function ✧.*



Functions and conditions

COUNTIF function ✧.*

returns the number of cells that match a specified value.
=COUNTIF(start cell:end cell,”number condition”)

Summary table ✧.*

a table used to summarize statistical information about data

SUMIF function ✧.*

a function that adds numeric data based on one condition.
=SUMIF(range, criteria.condition,[sum_range])

Functions with multiple conditions

In this reading, you will learn more about conditional functions and how to construct functions with multiple conditions. Recall that conditional functions and formulas perform calculations according to specific conditions. Previously, you learned how to use functions like **SUMIF** and **COUNTIF** that have one condition. You can use the **SUMIFS** and **COUNTIFS** functions if you have two or more conditions. You will learn their basic syntax in Google Sheets, and check out an example.
Refer to the resources at the end of this reading for information about similar functions in Microsoft Excel.

SUMIF to SUMIFS

The basic syntax of a SUMIF function 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, you might have a table with a list of expenses, their cost, and the date they occurred.

	A	B	C
1	Expense	Price	Date
2	Fuel	$48.00	12/14/2020
3	Food	$12.34	12/14/2020
4	Taxi	$21.57	12/14/2020
5	Coffee	$2.50	12/15/2020
6	Fuel	$36.00	12/15/2020
7	Taxi	$15.88	12/15/2020
8	Coffee	$4.15	12/15/2020
9	Food	$6.75	12/15/2020

You could use SUMIF to calculate the total price of fuel in this table, like this:

```
A11   fx   =SUMIF(A1:A9, "Fuel",B1:B9)
```

But, you could also build in multiple conditions by using the SUMIFS function. SUMIF and SUMIFS are very similar, but SUMIFS can include multiple conditions.
The basic syntax is: **=SUMIFS(sum_range, criteria_range1, criterion1, [criteria_range2, criterion2, ...])**

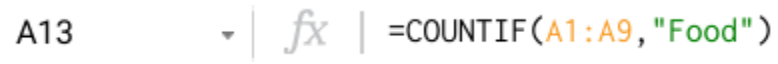
The square brackets let you know that this is optional. The ellipsis at the end of the statement lets you know that you can have as many repetition of these parameters as needed. For example, if you wanted to calculate the sum of the fuel costs for one date in this table, you could create a SUMIFS statement with multiple conditions, like this:

```
A12   fx   =SUMIFS(B1:B9,A1:A9,"Fuel",C1:C9,"12/15/2020")
```

This formula gives you the total cost of every fuel expense from the date listed in the conditions. In this example, C1:C9 is our second criterion_range and the date 12/15/2020 is the second condition. 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 basic syntax for COUNTIF is: **=COUNTIF(range, criterion)**
Just like SUMIF, you set the range and then the condition that needs to be met. For example, if you wanted to count the number of times Food came up in the Expenses column, you could use a COUNTIF function like this:



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 times Coffee appeared in the Expenses column on 12/15/2020, you could use COUNTIFS to apply those conditions, like this:

This formula follows the basic syntax to create conditions for “Coffee” and the specific date. Now we can find every instance where both of these conditions 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. But, there are other functions with multiple conditions that you can use in your data analysis. There are a lot of resources available online to help you get started with these other functions:

- **How to use the Excel IFS function**: This resource includes an explanation and example of the IFS function in Excel. This is a great reference if you are 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**: 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:** This resource explains how to use the INDEX and MATCH functions with multiple criteria. It also includes an example which helps demonstrate how these functions work with multiple criteria and actual data.
- **Using IF with AND, OR, and NOT functions in Excel:** 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.

Composite functions

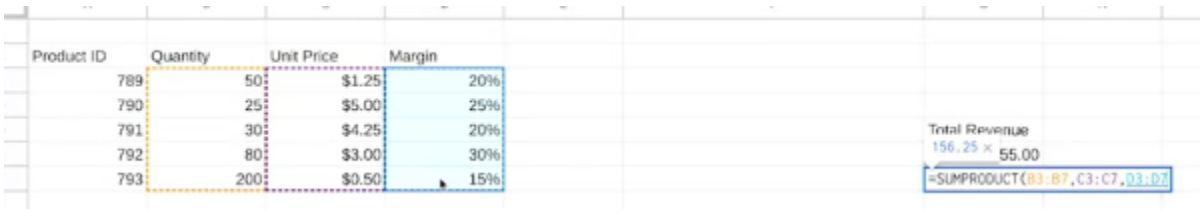
SUMPRODUCT ✧.*

a function that multiplies arrays and returns the sum of those products.

```
=sumproduct(array1, [array2]....)
```

Profit margin ✧.*

a percentage that indicates how many cents of profit have been generated for each dollar of sale.



Start working with pivot tables

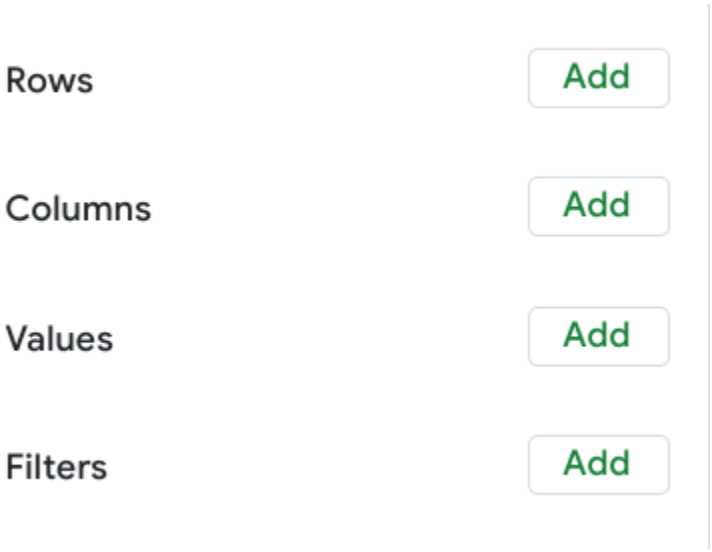
Calculated field ✧.*

a new field within a pivot table that carries out certain calculations based on the values of other fields.

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.



The **rows** of a pivot table organize and group data you select horizontally. For example, in the Working with pivot tables video, the Release Date values were used to create rows that grouped the data by year.

Release Date - Year
2012
2013
2014
2015
2016

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.

SUM of Box Office Revenue ($)	AVERAGE of Box Office Revenue ($)	COUNT of Box Office Revenue ($)
$18,078,040,000.00	$170,547,547.17	106
$13,672,800,000.00	$160,856,470.59	85
$20,013,420,000.00	$168,180,000.00	119
$13,521,310,000.00	$109,042,822.58	124
$11,921,900,000.00	$161,106,756.76	74
$77,207,470,000.00	**$151,983,208.66**	**508**

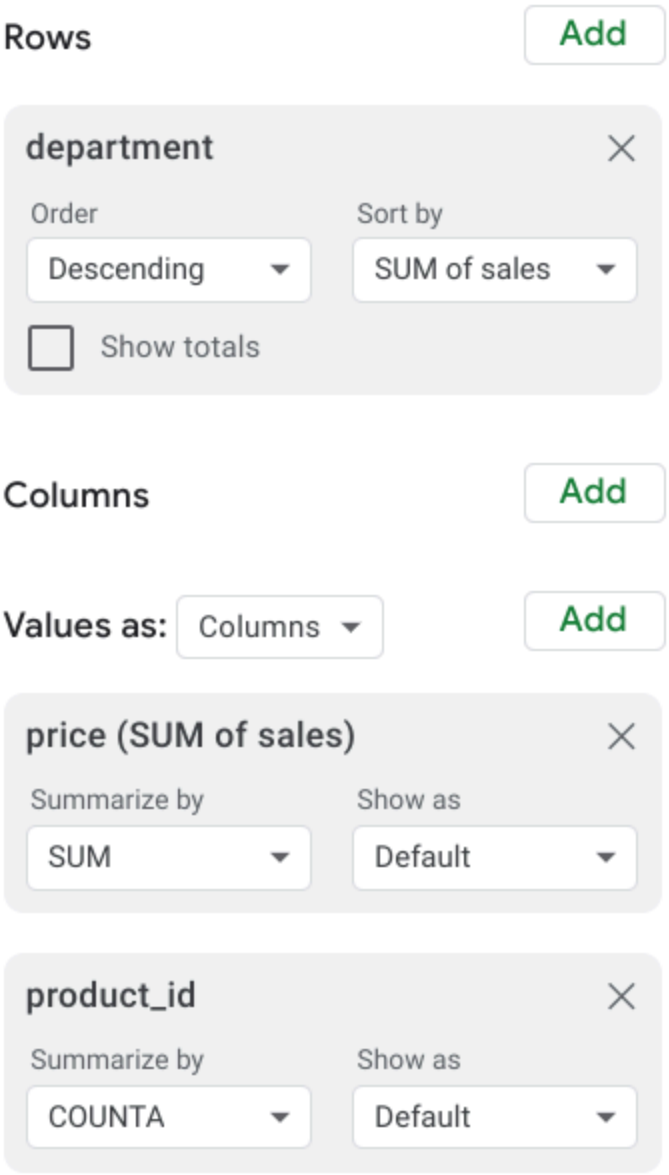
Finally, the **filters** section of a pivot table enables you to apply filters based on specific criteria — just like filters in regular spreadsheets! For example, a filter was added to the movie data pivot table so that it only included movies that generated less than $10 million in revenue.

Release Date - Year	SUM < $10M	AVERAGE < $10 M	COUNT < $10 M
2012	$18,078,040,000.00	$170,547,547.17	106
2013	$13,672,800,000.00	$160,856,470.59	85
2014	$20,013,420,000.00	$168,180,000.00	119
2015	$13,521,310,000.00	$109,042,822.58	124
2016	$11,921,900,000.00	$161,106,756.76	74
Grand Total	**$77,207,470,000.00**	**$151,983,208.66**	**508**

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.



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.

department	SUM of sales	COUNTA of product_id
Toys	$3,045.95	49
Beauty	$2,958.37	55
Movies	$2,880.55	57
Tools	$2,869.96	48
Games	$2,785.80	49
Industrial	$2,728.90	51
Jewelry	$2,669.25	52
Health	$2,613.56	48
Automotive	$2,589.56	47
Garden	$2,586.92	45
Sports	$2,460.12	46
Grocery	$2,459.22	44
Outdoors	$2,399.48	47
Electronics	$2,353.65	41
Books	$2,313.01	42
Baby	$2,272.77	46
Home	$2,222.99	38
Computers	$2,206.20	44
Kids	$2,117.98	41
Shoes	$2,108.33	39
Clothing	$1,858.47	38
Music	$1,809.36	33

Now they know that the Toys department generated the most revenue!

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.

Using pivot tables in analysis

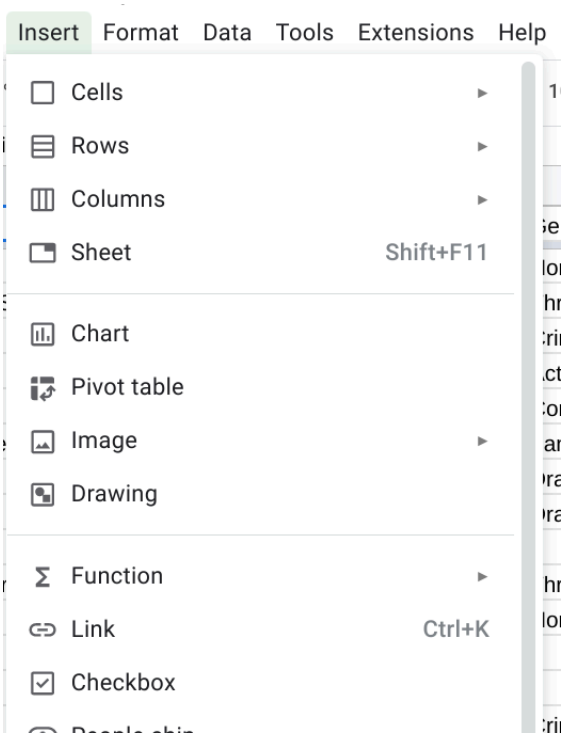
In this reading, you will learn how to create and use pivot tables for data analysis. You will also get some resources about pivot tables that you can save for your own reference when you start creating pivot tables yourself. **Pivot tables** are a spreadsheet tool that let you view data in multiple 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.

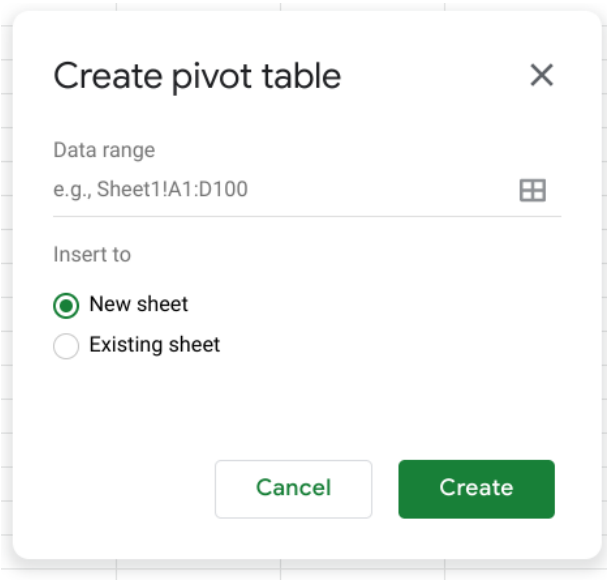
Create your pivot table

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**.

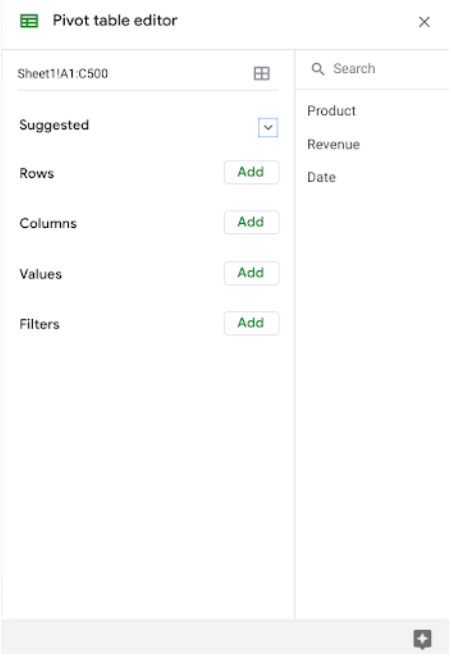


This pop-up menu will appear:



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: 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:** 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: 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:** 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: 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:** 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: 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: 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: 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: 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: 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: 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

Filter data in a pivot table: 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.

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.

Google Sheets

Customize a pivot table: 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.

Filter multiple values in pivot table: 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: 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: 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.

Queries and calculations

Operator ✧.*
symbol that names the type of operation or calculation to be performed in a formula.

The syntax of a query is its structure

```
SELECT
      columnA,
      columnB,
      columnA + columnB AS columnX
FROM
     table_name
```

Column A	Column B	Column X
A1	B1	Sum of A1 + B1
A2	B2	Sum of A2 + B2
A3	B3	Sum of A3 + B3

```
SELECT
      columnA,
      columnB,
      columnC,
      (columnA + columnB) * columnC AS columnX
FROM
      table_name
```

Column A	Column B	Column C	Column X
A1	B1	C1	(A1 + B1)*C1
A2	B2	C2	(A2 + B2)*C2
A3	B3	C3	(A3 + B3)*C3

Modulo ✧.*
This is an operator that returns the remainder when one number is divided by another.

Embedding simple calculations in SQL

```
SELECT
      columnA,
      columnB,
      columnA/columnB AS columnX
FROM
      table_name
```

Underscores ✧.*
Lines used to underline words and connect text characters

Calculations with other statements

Group By ✧.*
A command that groups rows that have the same values from a table into summary rows

Extract command ✧.*
Let us pull one part of a given date to use

Query editor

```
SELECT
  EXTRACT(YEAR FROM STARTTIME) AS year,
  COUNT(*) AS number_of_rides
FROM
  bigquery-public-data.new_york.citibike_trips
GROUP BY
  year
ORDER BY
  year
```

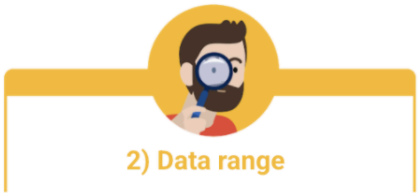
Check and recheck

Data validation process ✧.*
This process involves checking and rechecking the quality of your data so that it is complete, accurate, secure, and consistent.

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.

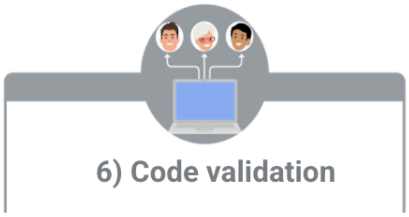


- **Purpose**: Check that the data falls within an acceptable range of values defined for the field.
- **Example**: Data values for school grades should be values between 1 and 12.
- **Limitations**: The data value 11.5 would be in the data range and would also pass as a numeric data type. But, it would 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.

Temporary tables

Temporary tables ✧.*
A database table that is created and exists temporarily on a database server.

WITH Clause ✧.*
Type of temporary table that you can query from multiples times

Multiple table variations

```
SELECT
      *
INTO
      AfricaSales
FROM
      GlobalSales
WHERE
      Region = "Africa"
```

```
CREATE TABLE AfricaSales AS
(
SELECT *
FROM GlobalSales
WHERE Region = "Africa"
)
```

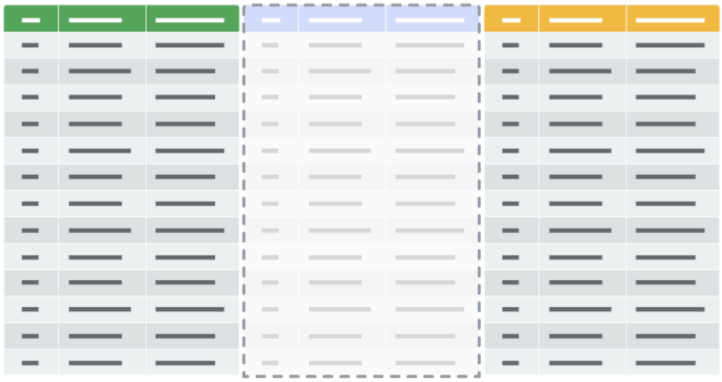
Working with temporary tables

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:

```
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:

```
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.

```
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:

```
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.