

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 points 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)

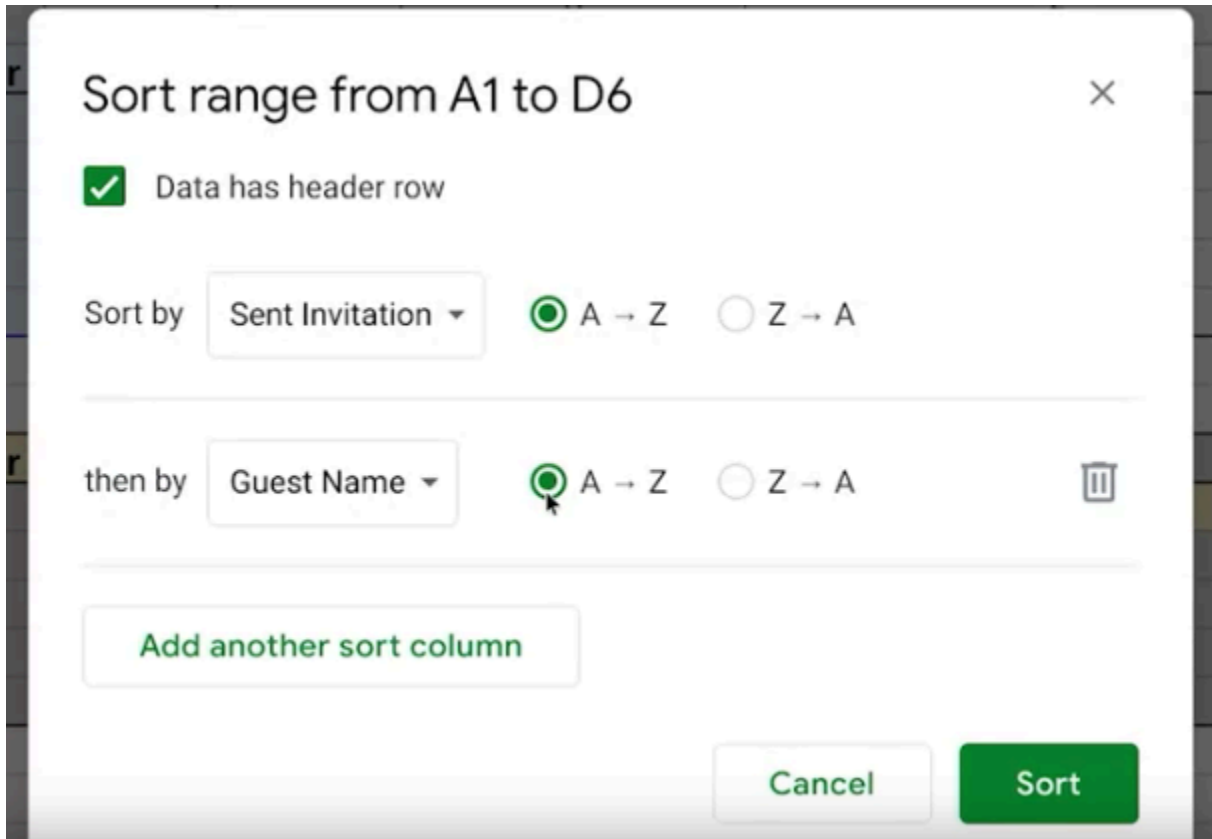
Ascending - true

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.



The screenshot shows a 'Sort range from A1 to D6' dialog box. At the top right is a close button (X). Below the title, there is a checked checkbox labeled 'Data has header row'. The first sorting condition is 'Sort by' with a dropdown menu showing 'Sent Invitation', followed by two radio buttons: 'A → Z' (which is selected) and 'Z → A'. The second sorting condition is 'then by' with a dropdown menu showing 'Guest Name', followed by two radio buttons: 'A → Z' (which is selected) and 'Z → A'. To the right of the second set of radio buttons is a trash icon. Below these conditions is a button labeled 'Add another sort column'. At the bottom right are two buttons: 'Cancel' and 'Sort'.

	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

```

Unsaved query Edited
1 SELECT *
2 FROM `movie_data.movies`
3 WHERE Genre = "Comedy"
4 ORDER BY Release_Date DESC

```

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 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.

The screenshot shows the 'Data validation' dialog box in Google Sheets. The 'Cell range' is set to 'Sheet 1!C1:C10C'. The 'Criteria' is set to 'List of items' with the values 'Not Yet Started, In Progress, Ready'. The 'Show dropdown list in cell' checkbox is checked. The 'On invalid data' options are 'Show warning' (selected) and 'Reject input'. The 'Appearance' section has 'Show validation help text' unchecked.

Below the dialog box, a table is visible with columns B, C, and D. Column B lists assignees, column C lists status, and column D lists review dates. A dropdown menu is open for the 'Status' column, showing the options 'Not Yet Started', 'In Progress', and 'Ready'.

	B	C	D
	Assignee	Status	Review By This Date
	Tony		2020-03
s	Ximena	Not Yet Started	2020-03
	Sally	In Progress	2020-04
ails	Erica	Ready	2020-04
ails	Ayanna		2020-04
/P	Hallie		2020-04

	B	C	D
	Assignee	Status	Review By This Date
	Tony		2020-03-15
S	Ximena	Not Yet Started	2020-03-15
	Sally	In Progress	2020-04-15
ails	Erica	Ready	2020-04-15
ails	Ayanna		2020-04-15
VP	Hallie		2020-04-15

- Create custom checkboxes

Assignee

ny

men

ully

ica

'ann

llie

ny

men

ully

ica

'ann

llie

ny

Data validation

Cell range:

Criteria:

☒ Use custom cell values

Checked:

Unchecked:

On invalid data: ☒ Show warning ☐ Reject input

	D	E	F
▼	Review By This Date	<input type="checkbox"/>	
▼	2020-03-13	<input type="checkbox"/>	
▼	2020-03-19	<input type="checkbox"/>	
▼	2020-04-02	<input type="checkbox"/>	
▼	2020-04-10	<input type="checkbox"/>	
▼	2020-04-17	<input type="checkbox"/>	
▼	2020-04-24	<input type="checkbox"/>	
▼	2020-04-27	<input type="checkbox"/>	
▼	2020-05-18	<input type="checkbox"/>	
▼	2020-05-25	<input type="checkbox"/>	
▼	2020-04-24	<input type="checkbox"/>	
▼	2020-04-27	<input type="checkbox"/>	
▼	2020-05-18	<input type="checkbox"/>	

- 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

B	C	D	E	F
Assignee	Status	Review By This Date	Review	
Tony	Not Yet Started	2020-03-13	<input type="checkbox"/>	
Ximena	Not Yet Started	2020-03-19	<input type="checkbox"/>	
Sally	Not Yet Started	2020-04-02	<input type="checkbox"/>	
Erica	Not Yet Started	2020-04-10	<input type="checkbox"/>	
Ayanna	Not Yet Started	2020-04-17	<input type="checkbox"/>	
Hallie	Not Yet Started	2020-04-24	<input type="checkbox"/>	
Tony	Not Yet Started	2020-04-27	<input type="checkbox"/>	
Ximena	Not Yet Started	2020-05-18	<input type="checkbox"/>	
Sally	Not Yet Started	2020-05-25	<input type="checkbox"/>	
Erica	Not Yet Started	2020-04-24	<input type="checkbox"/>	
Ayanna	Not Yet Started	2020-04-27	<input type="checkbox"/>	
Hallie	Not Yet Started	2020-05-18	<input type="checkbox"/>	
Tony	Not Yet Started	2020-05-25	<input type="checkbox"/>	
Ximena	Not Yet Started	2020-04-24	<input type="checkbox"/>	
Sally	Not Yet Started	2020-04-27	<input type="checkbox"/>	
Erica	Not Yet Started	2020-05-18	<input type="checkbox"/>	
Ayanna	Not Yet Started	2020-05-25	<input type="checkbox"/>	
Hallie	Not Yet Started	2020-04-24	<input type="checkbox"/>	

Single color

Color scale

Apply to range

C:C

Format rules

Format cells if...

Text is exactly

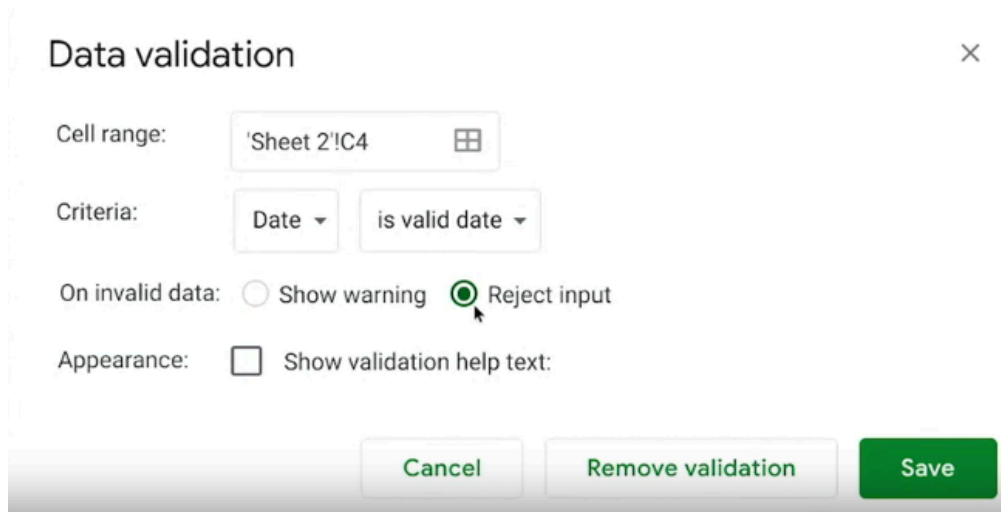
Not Yet Started

Formatting style

Custom

B I U S A

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

The image shows the 'Data validation' dialog box in Microsoft Excel. The title bar says 'Data validation' with a close button (X) on the right. The 'Cell range' field contains 'Sheet2!C4' with a grid icon to its right. The 'Criteria' section has two dropdown menus: the first is set to 'Date' and the second is set to 'is valid date'. The 'On invalid data' section has two radio buttons: 'Show warning' (unselected) and 'Reject input' (selected). The 'Appearance' section has a checkbox for 'Show validation help text' which is unchecked. At the bottom, there are three buttons: 'Cancel', 'Remove validation', and 'Save' (which is green).

Merging and multiple sources

CONCATENATE ✧.*

a function that joins together two or more text strings.

Text string ✧.*

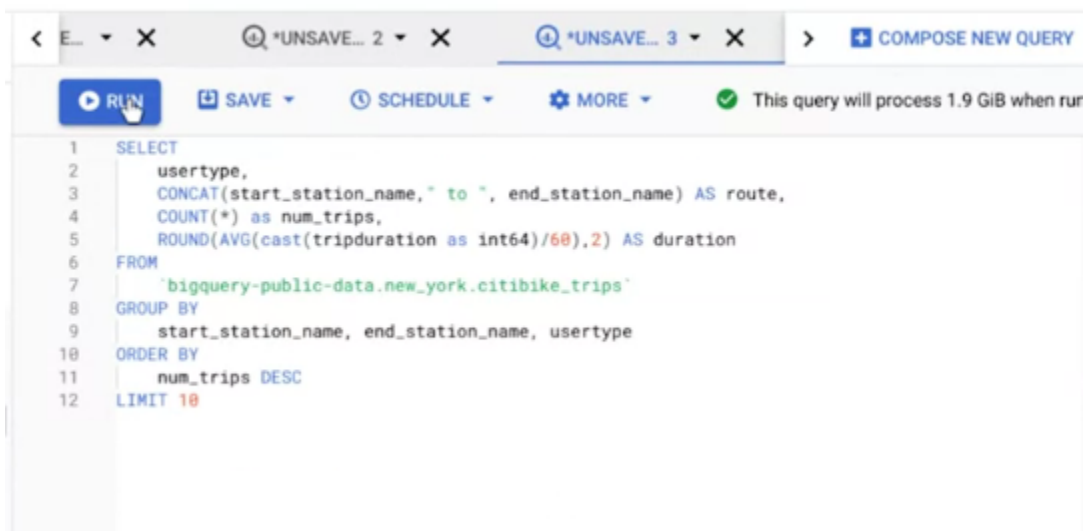
group of characters within a cell most often composed of letters.

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 screenshot shows a SQL query editor with a toolbar at the top containing buttons for RUN, SAVE, SCHEDULE, and MORE. A status message indicates the query will process 1.9 GiB. The query itself is as follows:

```
1 SELECT
2   usertype,
3   CONCAT(start_station_name, " to ", end_station_name) AS route,
4   COUNT(*) as num_trips,
5   ROUND(AVG(cast(tripduration as int64)/60),2) AS duration
6 FROM
7   `bigquery-public-data.new_york.citibike_trips`
8 GROUP BY
9   start_station_name, end_station_name, usertype
10 ORDER BY
11   num_trips DESC
12 LIMIT 10
```

- 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'
```

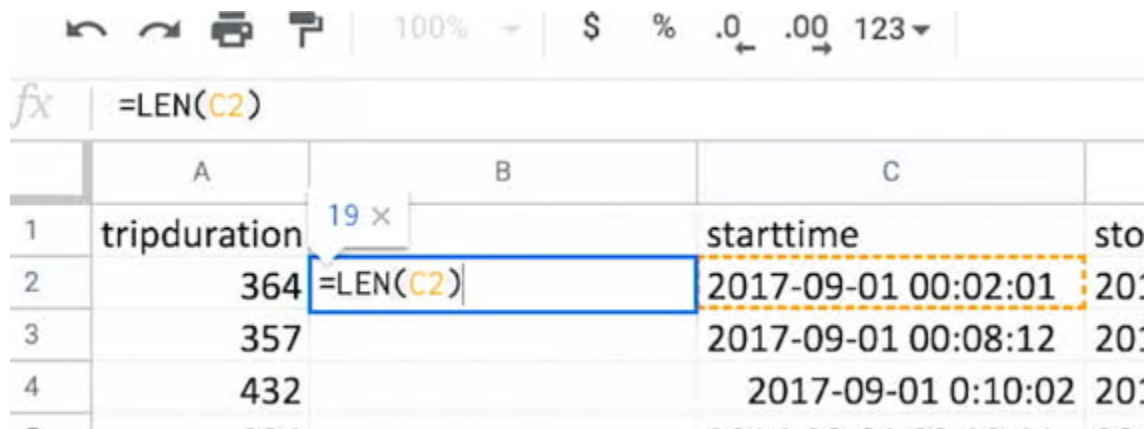
-
- we have to use GROUP BY to group together summary rows. Let's group by the start station, the end station, and the user type for this query.
- Finally, we'll use ORDER BY to tell it how we want to organize this data. For this, we want to figure out the most common trips so we can input the number 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.



The screenshot shows a Google Sheet interface. At the top, there's a toolbar with icons for undo, redo, print, and copy, along with formatting options like 100%, \$, %, .0, .00, and 123. Below the toolbar, the formula bar shows '=LEN(C2)'. The spreadsheet has columns A, B, and C. Column A is labeled 'tripduration', column B is labeled 'starttime', and column C is labeled 'stopduration'. Row 1 contains the headers. Row 2 contains the values 364, 2017-09-01 00:02:01, and 2017-09-01 00:02:01. Row 3 contains 357, 2017-09-01 00:08:12, and 2017-09-01 00:08:12. Row 4 contains 432, 2017-09-01 0:10:02, and 2017-09-01 0:10:02. A blue box highlights the cell B2, which contains the formula '=LEN(C2)'. A tooltip above the box shows '19 x'.

	A	B	C
1	tripduration		starttime
2	364	=LEN(C2)	2017-09-01 00:02:01
3	357		2017-09-01 00:08:12
4	432		2017-09-01 0:10:02

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.

A	B	C	D
tripduration		starttime	stoptime
364	11 x	19 2017-09-01 00:02:01	2017-09-01
357	=FIND(" ",C3)	2017-09-01 00:08:12	2017-09-01
432		2017-09-01 0:10:02	2017-09-01
934		2017-09-01 00:10:11	2017-09-01
932		2017-09-01 00:10:16	2017-09-01

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.

=RIGHT(C2,8)			
A	B	C	D
tripduration		starttime	stoptime
364	00:02:01 x =RIGHT(C2,8)	2017-09-01 00:02:01	2017-09-01
357		2017-09-01 00:08:12	2017-09-01
432		2017-09-01 0:10:02	2017-09-01
934		2017-09-01 00:10:11	2017-09-01

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.

=LEFT(D2,11)			
A	B	C	D
tripduration		Time	starttime
364	2017-09-01 x =LEFT(D2,11)	00:02:01	2017-09-01 00:02:01
357		00:08:12	2017-09-01 00:08:12
432		0:10:02	2017-09-01 0:10:02
934		00:10:11	2017-09-01 00:10:11
932		00:10:16	2017-09-01 00:10:16

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 ✧.*

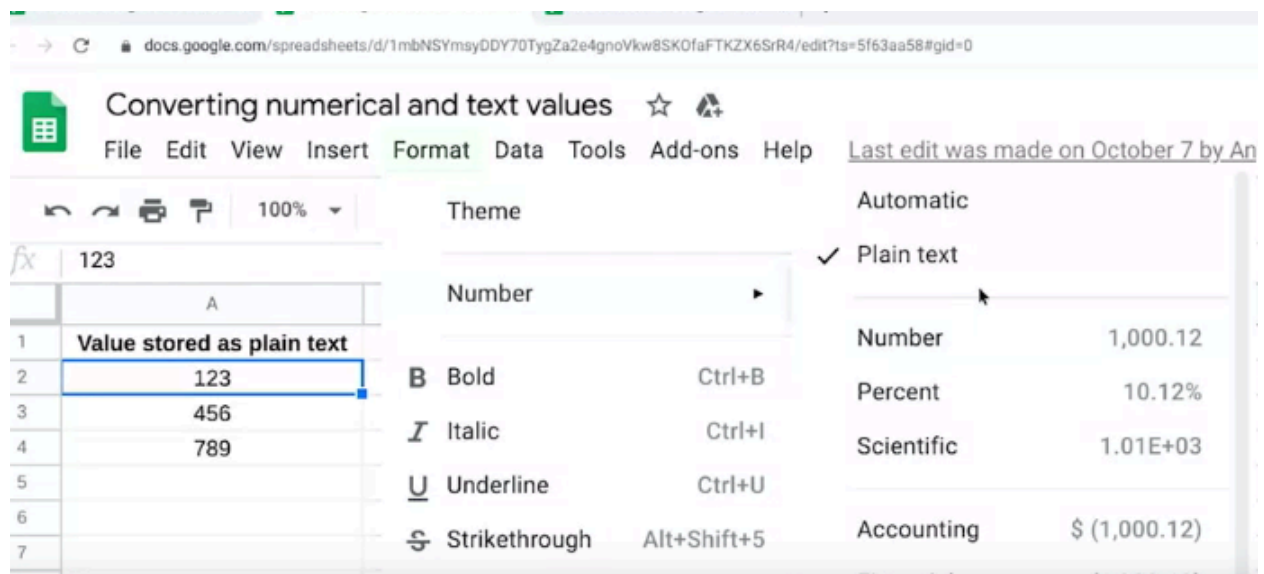
The process of gathering data from multiple sources in order to combine it into a single summarized collection

Vertical Lookup (VLOOKUP) ✧.*

A function that searches for a certain value in a column to return a corresponding piece of information

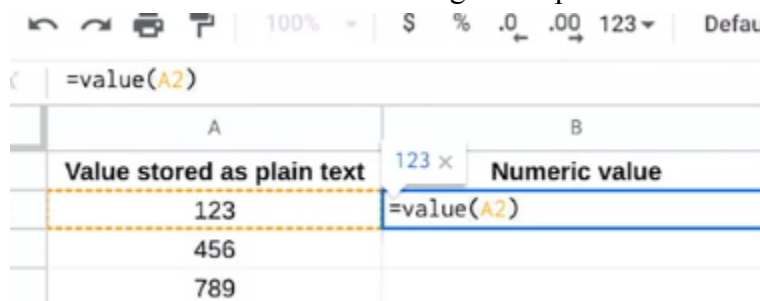
- **Converting text to numerical value**

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



VALUE ✧.*

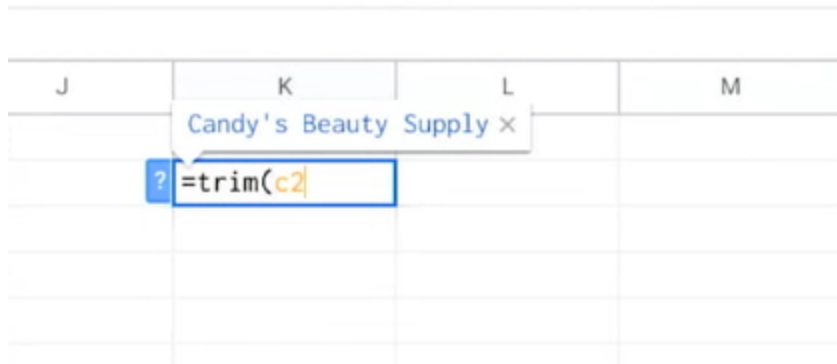
a function that converts a text string that represents a number to a numerical value.



- Having extra spaces

TRIM ✧.*

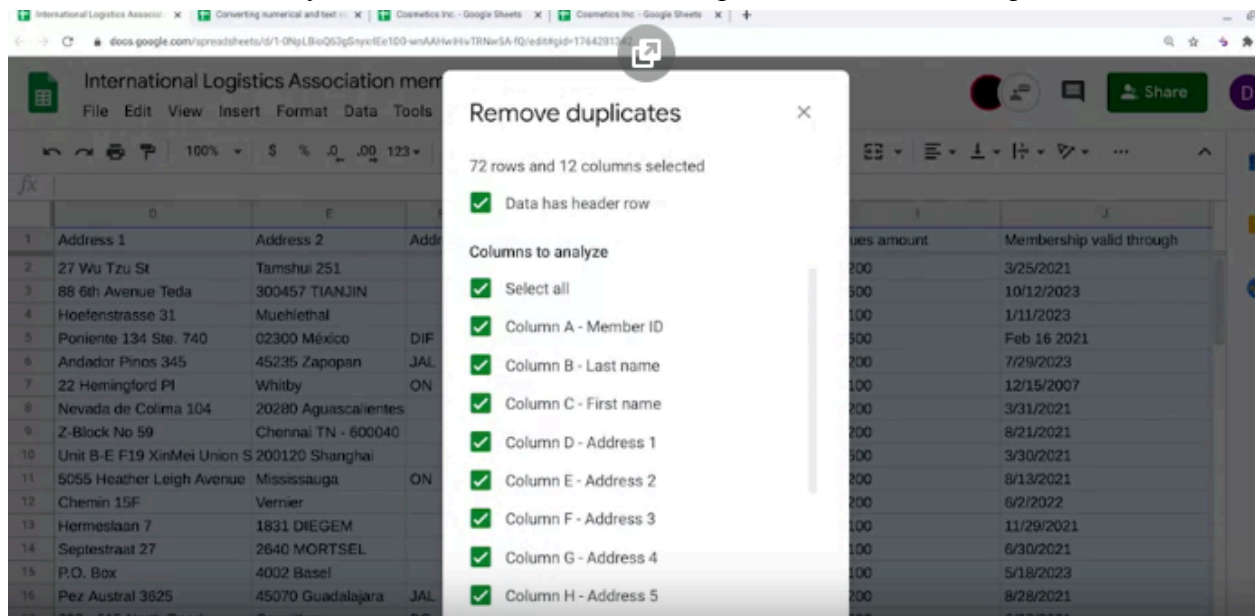
automatically deletes any extra spaces added to the cell.



- 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					

C2	Σ	fx	=vlookup(A2, 'Employee Rates'!\$A\$2:\$B\$5,2,FALSE)		
	A	B	C		
1	Employee #	Hours Worked			
2	FT12578	20	=vlookup(A2, 'Employee Rates'!\$A\$2:\$B\$5,2,FALSE)		
3	FT12579	20			
4	FT12580	20			

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.

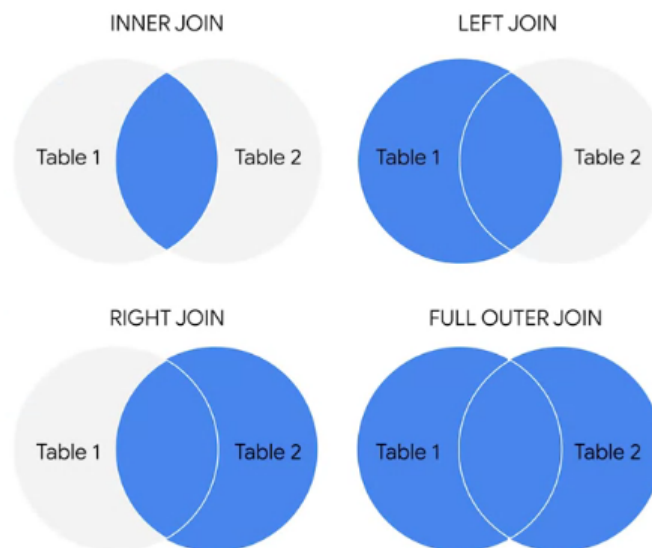
Understanding JOINS

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

```
51 SELECT
52   employees.name AS employee_name,
53   employees.role AS employee_role,
54   departments.name AS department_name
55 FROM
56   employees
57 INNER JOIN
58   departments ON
59   employees.department_id = departments.department_id
60
61
```

Run Save query Save view Schedule query More

Query results SAVE RESULTS EXPLORE DATA

Query complete (0.7 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

LEFT JOIN ✧.*

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

Query editor

+ COMP

50
51 **SELECT**
52 employees.name AS employee_name,
53 employees.role AS employee_role,
54 departments.name AS department_name
55 **FROM**
56 employees
57 **LEFT JOIN**
58 departments **ON**
59 employees.department_id = departments.department_id
60
61

Run

Save query

Save view

Schedule query

More

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	Mary Martin	Receptionist	null

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	<i>null</i>	<i>null</i>	Accounting		
7	<i>null</i>	<i>null</i>	Sales		

```

SELECT
  *
FROM
  tableA
LEFT JOIN
  tableB
ON
  keyA = keyB

SELECT
  *
FROM
  tableB
RIGHT JOIN
  tableA
ON
  keyA = keyB

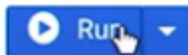
```

OUTER JOIN ⇧.*

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

Unsaved query Edited

```
50
51 SELECT
52     employees.name AS employee_name,
53     employees.role AS employee_role,
54     departments.name AS department_name
55 FROM
56     employees
57 FULL OUTER JOIN
58     departments ON
59     employees.department_id = departments.department_id
60
61
```



Run



Save query



Save view



Schedule query



More

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	Mary Martin	Receptionist	null
7	null	null	Accounting
8	null	null	Sales

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](#).

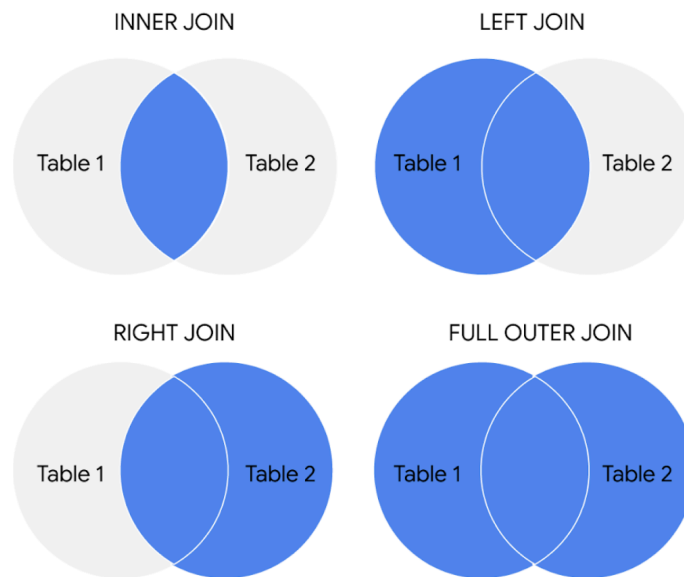
The general JOIN syntax

```
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

The top screenshot shows a SQL query editor with the following query:

```
1 SELECT
2 *
3 FROM
4 warehouse_orders.Orders orders
```

The bottom screenshot shows a SQL query editor with the following query:

```
1 SELECT
2 warehouse.state as state,
3 COUNT(DISTINCT order_id) as num_orders
4 FROM
5 warehouse_orders.Orders orders
6 JOIN
```

Both screenshots show the 'Query results' section with a table of data:

Row	state	num_orders
1	MI	6205
2	KY	1048
3	TN	2746

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)

K	L	M	N
er	November	December	\$871,585.00 x
2.00	\$70,048.00	\$145,378.00	=SUM(B2:M2)
15.00	\$81,811.00	\$199,468.00	
13.00	\$79,809.00	\$155,736.00	

AVERAGE function ✧.*

	A	B	C
1	Monthly sales	January	February
2	2011	\$47,563.00	\$49,078.00
3	2012	\$39,575.00	\$50,384.00
4	2013	\$56,591.00	\$50,319.00
5	2014	\$39,113.00	\$40,107.00
6	2015	\$41,666.00	\$53,993.00
7	2016	\$38,405.00	\$46,658.00
8	2017	\$41,756.00	\$41,311.00
9	2018	\$56,061.00	\$40,703.00
10	2019	\$57,355.00	\$46,703.00
11	2020	\$46,031.90	\$54,050.00
12	Average by month	=AVERAGE(B2:B11)	
13			

Conditional formatting ✧.*

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

MIN function ✧.*

9	\$56,515.00	\$60,270.00	\$75,195.00	\$70,765.00
10	\$57,172.00	\$63,455.00	\$72,180.00	\$82,110.00
11	\$61,252.00	\$55,787.00	\$76,888.00	\$88,438.00
12	\$58,841.70	\$58,215.30	\$74,659.40	
13				
14				
15				
16				

MAX function ✧.*

3		
4	Lowest monthly average	\$165,642.90
5	Highest monthly average	=MAX(B12:M12)
6		
7		