MODULE 03

INTRODUCTION TO BASIC TOOLS FOR DATA ANALYTICS

Step-by-Step: Make spreadsheets your friend:

Example 1: Get started

Enter basic data:

1. Begin with a new spreadsheet.
2. Select cell **A2**.
3. Enter your **first name**.
4. Select cell **B2**.
5. Enter your **last name**.

Adjust the size of rows and columns:

To make the text fit in the rows and columns, adjust their sizes. Use either of the following methods:

1. If your name is longer than the width of the column, **select and drag** the right edge of the corresponding column until it fits.
2. To **wrap text**, select the cells, columns, or rows with text that you want to reformat.
3. Select the **Format** menu.
4. Under **Wrapping**, select **Wrap**.

Example 2: Add labels

Add labels, or attributes, to help you keep track of the data:

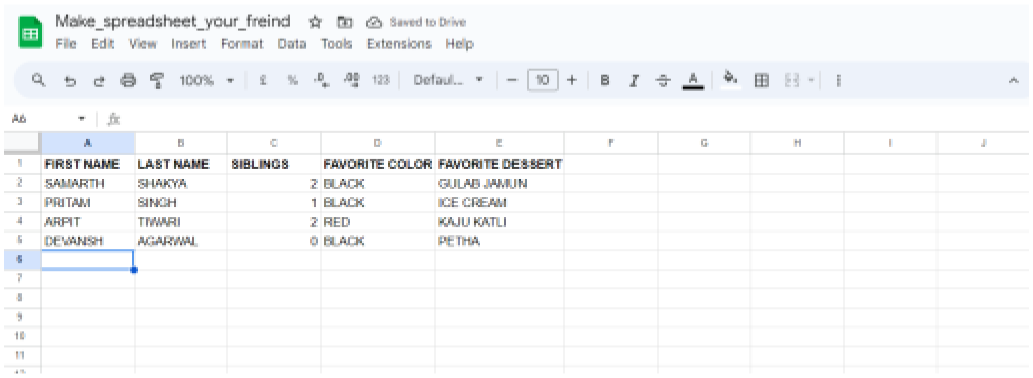
1. Select cell **A1**.
2. Enter **First Name**.
3. Select cell **B1**.
4. Enter **Last Name**.

5. Select cells **A1** and **B1**. To do this, select a single cell and drag your cursor over to the other cell to include it in the selection.
6. From the toolbar, select the **bold icon**.

Example 3: Add more attributes and data

Add more attributes and data to your spreadsheet:

1. Select cell **C1** and enter **Siblings**.
2. Select cell **D1** and enter **Favorite Color**.
3. Select cell **E1** and enter **Favorite Dessert**.
4. Select **all three cells** and make them bold by selecting the **bold icon** from the toolbar.
5. Adjust the columns to fit the new text.
6. Enter the corresponding data in cells **C2**, **D2**, and **E2** (your number of siblings, favorite color, and favorite dessert).
7. Add data about two more people in **rows 3 and 4**. These can be people you know or people you've just made up.


Make_spreadsheet_your_freind
Saved to Drive
File Edit View Insert Format Data Tools Extensions Help
100%
Defaul...
10
A6

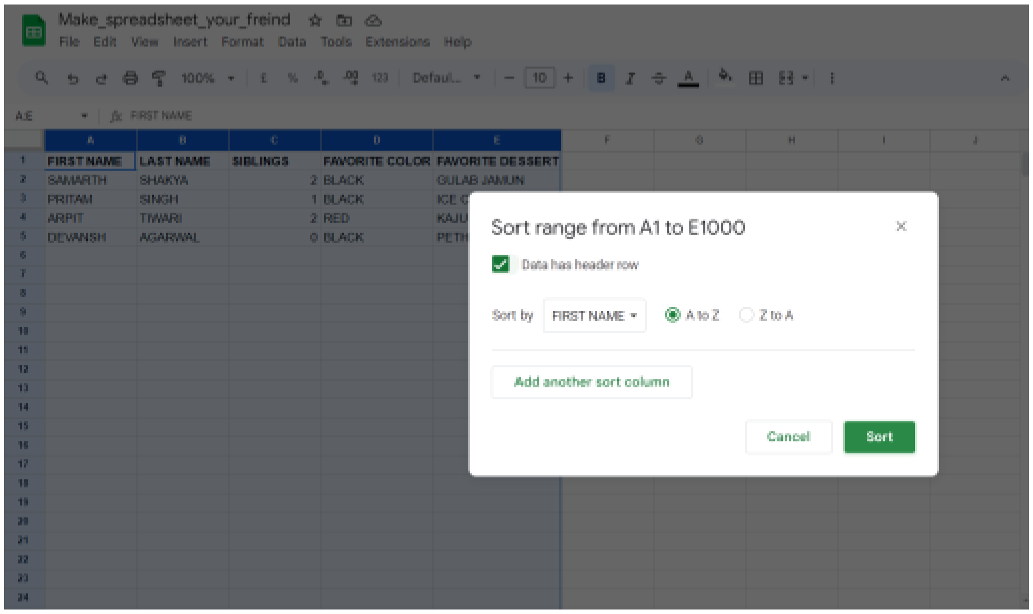
	A	B	C	D	E
1	FIRST NAME	LAST NAME	SIBLINGS	FAVORITE COLOR	FAVORITE DESSERT
2	SAMARTH	SHAKYA	2	BLACK	GULAB JAMUN
3	PRITAM	SINGH	1	BLACK	ICE CREAM
4	ARPIT	TIWARI	2	RED	KAJU KATLI
5	DEVANSH	AGARWAL	0	BLACK	PETHA

Example 4: Organize your data

One way to organize your data is by sorting it.

1. Select all columns that contain data. There are a few ways to select multiple cells:

   1. To select nonadjacent cells and/or cell ranges, hold the **Command** (Mac) or **Ctrl** (PC) **key** and select the cells.
   2. To select a range of cells, hold the **Shift** key and either drag your cursor over which cells you want to include or use the arrow keys to select a range.
   3. Select a single cell and drag your cursor over the cells you want to include in your selection.
2. Select the **Data** menu.
3. Select **Sort range**, then select **Advanced range sorting options**.
4. In the **Advanced range sorting options** window, select the checkbox for **Data has header row**. Make sure that **A to Z** is selected.
5. Select the **Sort by** drop-down menu, then select **Siblings**.
6. Select **Sort**. This will organize the spreadsheet by the number of siblings, from lowest to highest.

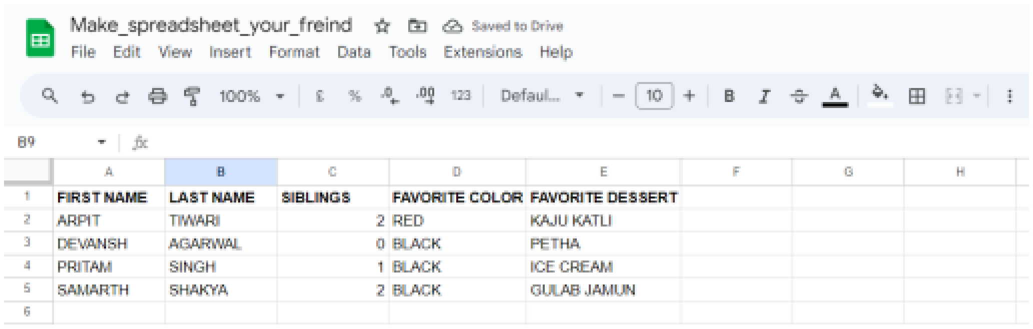

Make_spreadsheet_your_freind
File Edit View Insert Format Data Tools Extensions Help
100%
Defaul...
10
A:E
FIRST NAME
Add another sort column
Cancel
Sort

FIRST NAME	LAST NAME	SIBLINGS	FAVORITE COLOR	FAVORITE DESSERT
SAMARTH	SHAKYA	2	BLACK	GULAB JAMUN
PRITAM	SINGH	1	BLACK	ICE C
ARPIT	TIWARI	2	RED	KAJU
DEVANSH	AGARWAL	0	BLACK	PETH

Sort range from A1 to E1000

[✓] Data has header row

Sort by FIRST NAME [✓ A to Z] Z to A

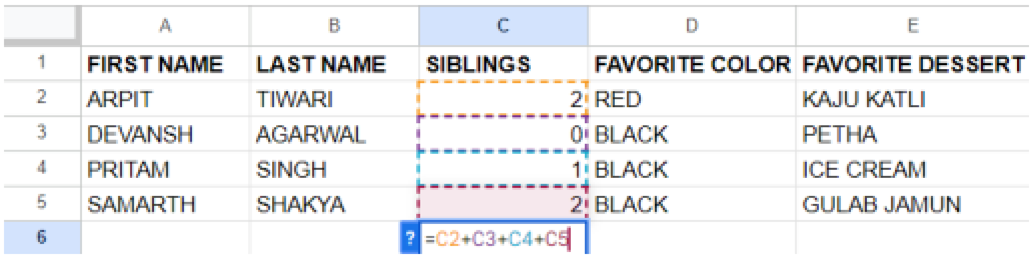

Make_spreadsheet_your_freind
Saved to Drive
File Edit View Insert Format Data Tools Extensions Help
100%
Defaul...
10
B9

FIRST NAME	LAST NAME	SIBLINGS	FAVORITE COLOR	FAVORITE DESSERT
ARPIT	TIWARI	2	RED	KAJU KATLI
DEVANSH	AGARWAL	0	BLACK	PETHA
PRITAM	SINGH	1	BLACK	ICE CREAM
SAMARTH	SHAKYA	2	BLACK	GULAB JAMUN

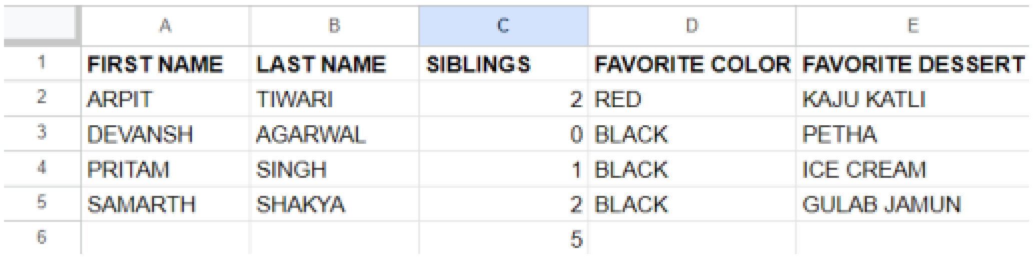
Example 5: Use a formula

Spreadsheets enable data professionals to analyze data. In this example, the instructor uses a formula to calculate a sum.

1. Select the next empty cell in the **Siblings** column (**C5**).
2. Enter the formula *=C2+C3+C4*.
3. Press **Enter** on your keyboard to complete the formula.
4. The formula calculates the total number of siblings.



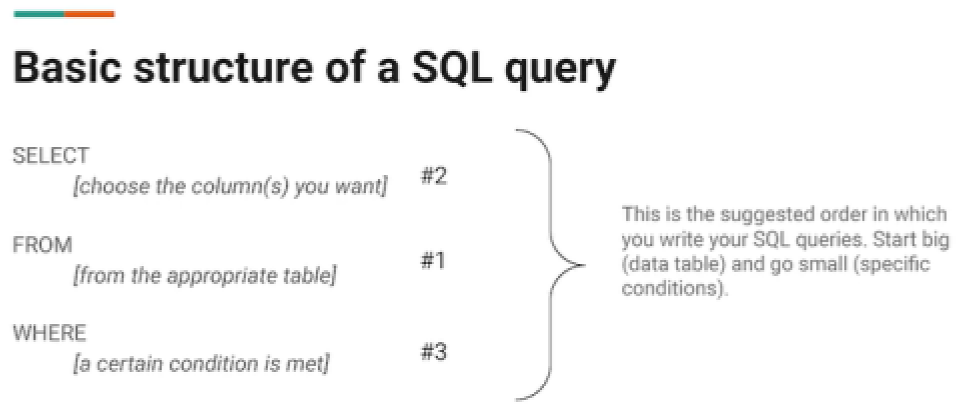
	A	B	C	D	E
1	FIRST NAME	LAST NAME	SIBLINGS	FAVORITE COLOR	FAVORITE DESSERT
2	ARPIT	TIWARI	2	RED	KAJU KATLI
3	DEVANSH	AGARWAL	0	BLACK	PETHA
4	PRITAM	SINGH	1	BLACK	ICE CREAM
5	SAMARTH	SHAKYA	2	BLACK	GULAB JAMUN
6			=C2+C3+C4+C5		



	A	B	C	D	E
1	FIRST NAME	LAST NAME	SIBLINGS	FAVORITE COLOR	FAVORITE DESSERT
2	ARPIT	TIWARI	2	RED	KAJU KATLI
3	DEVANSH	AGARWAL	0	BLACK	PETHA
4	PRITAM	SINGH	1	BLACK	ICE CREAM
5	SAMARTH	SHAKYA	2	BLACK	GULAB JAMUN
6			5		

SQL: Structured query language

QUERY: A request for data or information from a database.



Basic structure of a SQL query

SELECT
[choose the column(s) you want] #2

FROM
[from the appropriate table] #1

WHERE
[a certain condition is met] #3

This is the suggested order in which you write your SQL queries. Start big (data table) and go small (specific conditions).

What is a query?

A **query** is a request for data or information from a database. When you query databases, you use SQL to communicate your question or request. You and the database can always exchange information as long as you speak the same language.

Every programming language, including SQL, follows a unique set of guidelines known as **syntax**. Syntax is the predetermined structure of a language that includes all required words, symbols, and punctuation, as well as their proper placement. As soon as you enter your search criteria using the correct syntax, the query starts working to pull the data you've requested from the target database.

The syntax of every SQL query is the same:

- Use *SELECT* to choose the columns you want to return.
- Use *FROM* to choose the tables where the columns you want are located.
- Use *WHERE* to filter for certain information.

`SELECT`	Specifies the columns from which to retrieve data
`FROM`	Specifies the table from which to retrieve data
`WHERE`	Specifies criteria that the data must meet

Example of a query

Here is how a simple query would appear in BigQuery, a data warehouse on the Google Cloud Platform.

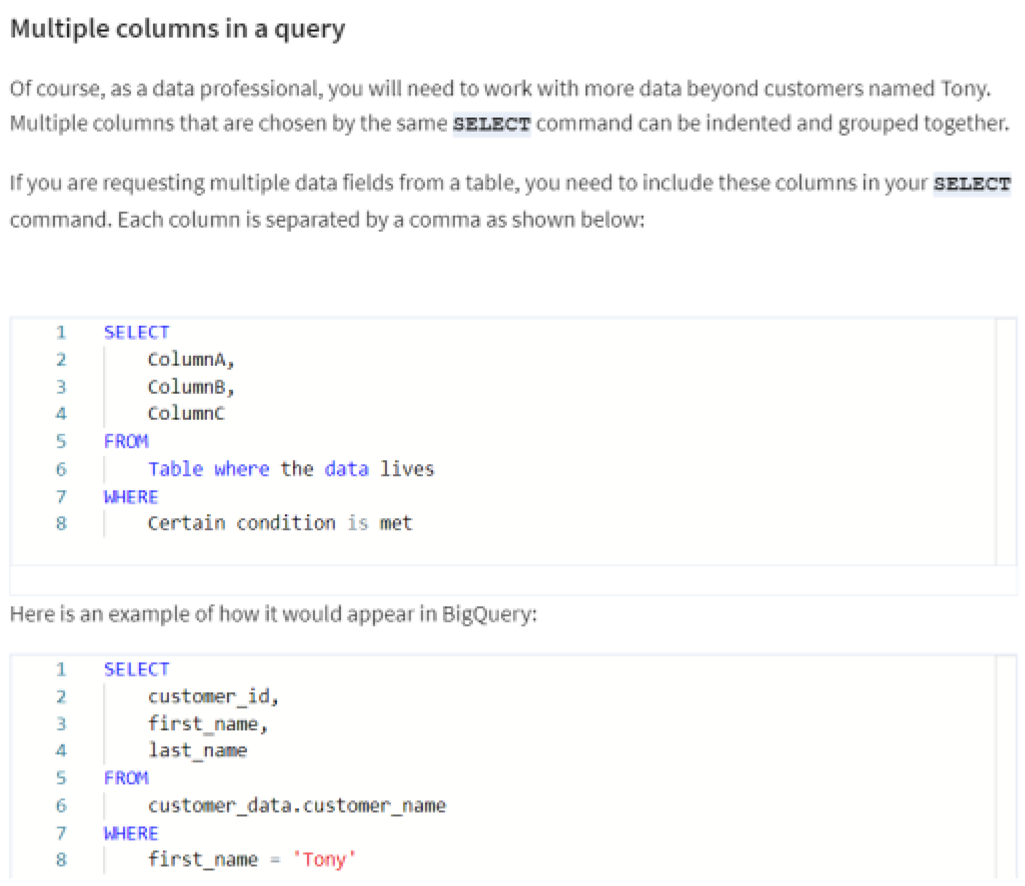
```sql
SELECT first_name
FROM customer_data.customer_name
WHERE first_name = 'Tony'
```

The above query uses three commands to locate customers with the `first_name`, `'Tony'`:

1. `SELECT` the column named `first_name`
2. `FROM` a table named `customer_name` (in a dataset named `customer_data`)
   (The dataset name is always followed by a dot, and then the table name.)
3. But only return the data `WHERE` the `first_name` is `'Tony'`

The results from the query might be similar to the following:

first_name
Tony
Tony
Tony



Multiple columns in a query

Of course, as a data professional, you will need to work with more data beyond customers named Tony. Multiple columns that are chosen by the same **`SELECT`** command can be indented and grouped together.

If you are requesting multiple data fields from a table, you need to include these columns in your **`SELECT`** command. Each column is separated by a comma as shown below:

```sql
SELECT
    ColumnA,
    ColumnB,
    ColumnC
FROM
    Table where the data lives
WHERE
    Certain condition is met
```

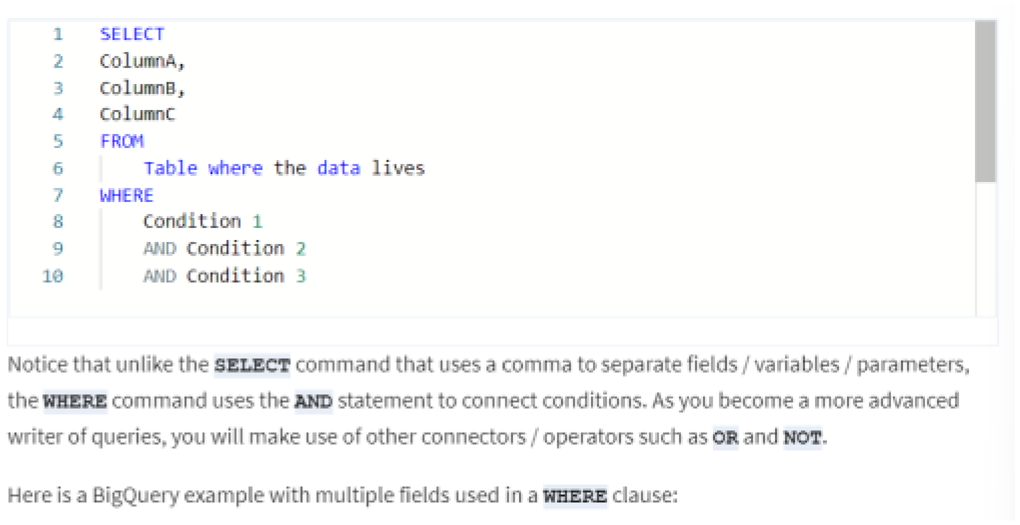
Here is an example of how it would appear in BigQuery:

```sql
SELECT
    customer_id,
    first_name,
    last_name
FROM
    customer_data.customer_name
WHERE
    first_name = 'Tony'
```

The above query uses three commands to locate customers with the *first_name*, *'Tony'*.

1. *SELECT* the columns named *customer_id*, *first_name*, and *last_name*
2. *FROM* a table named *customer_name* (in a dataset named *customer_data*)(The dataset name is always followed by a dot, and then the table name.)
3. But only return the data *WHERE* the *first_name* is *'Tony'*

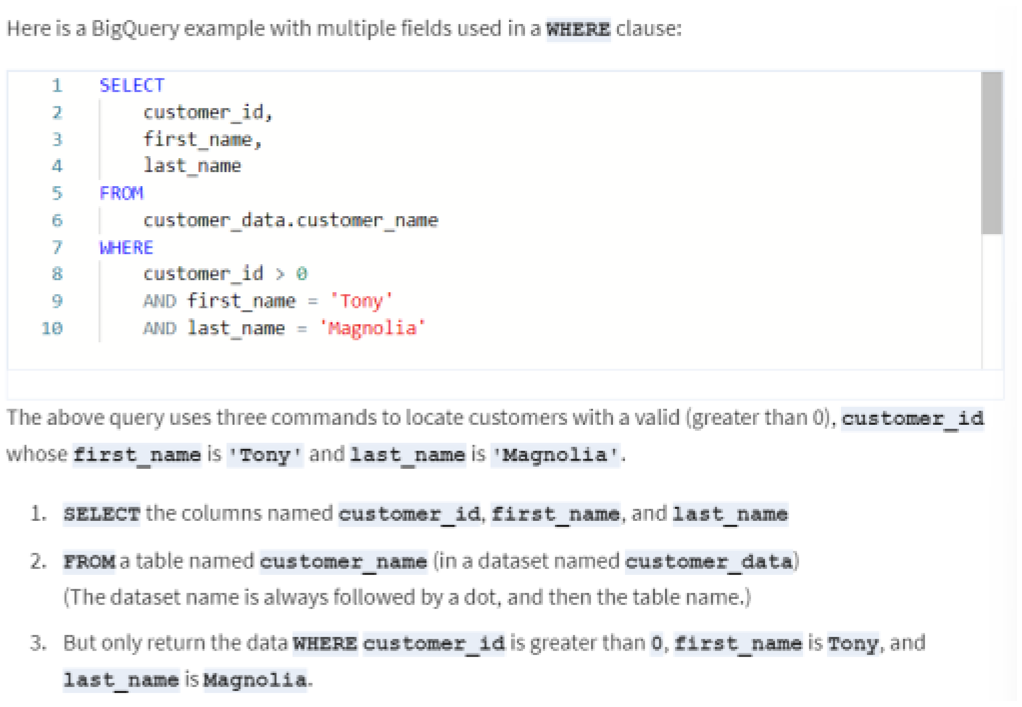
The only difference between this query and the previous one is that more data columns are selected. The previous query selected *first_name* only while this query selects *customer_id* and *last_name* in addition to *first_name*. In general, it is a more efficient use of resources to select only the columns that you need. For example, it makes sense to select more columns if you will actually use the additional fields in your *WHERE* clause. If you have multiple conditions in your *WHERE* clause, they may be written like this:



```sql
SELECT
ColumnA,
ColumnB,
ColumnC
FROM
    Table where the data lives
WHERE
    Condition 1
    AND Condition 2
    AND Condition 3
```

Notice that unlike the **`SELECT`** command that uses a comma to separate fields / variables / parameters, the **`WHERE`** command uses the **`AND`** statement to connect conditions. As you become a more advanced writer of queries, you will make use of other connectors / operators such as **`OR`** and **`NOT`**.

Here is a BigQuery example with multiple fields used in a **`WHERE`** clause:

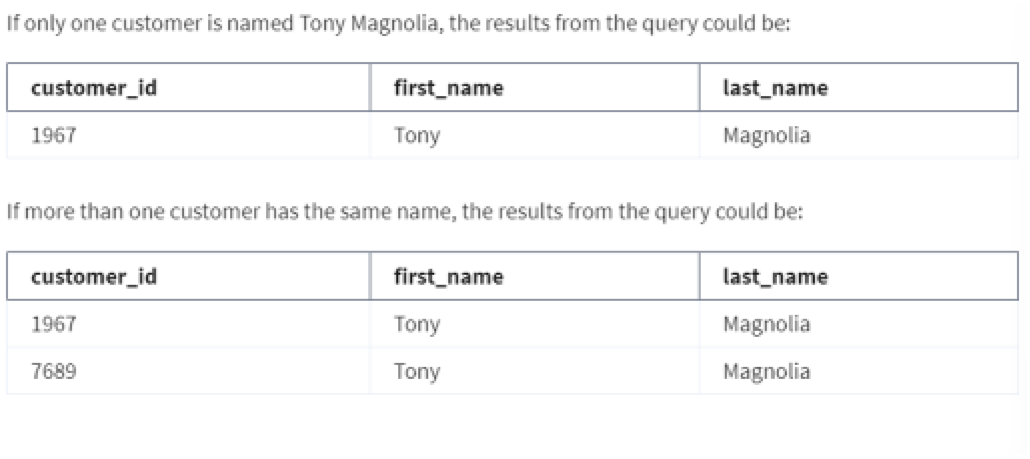


Here is a BigQuery example with multiple fields used in a **`WHERE`** clause:

```sql
SELECT
    customer_id,
    first_name,
    last_name
FROM
    customer_data.customer_name
WHERE
    customer_id > 0
    AND first_name = 'Tony'
    AND last_name = 'Magnolia'
```

The above query uses three commands to locate customers with a valid (greater than 0), **`customer_id`** whose **`first_name`** is **`'Tony'`** and **`last_name`** is **`'Magnolia'`**.

1. **`SELECT`** the columns named **`customer_id`**, **`first_name`**, and **`last_name`**
2. **`FROM`** a table named **`customer_name`** (in a dataset named **`customer_data`**)
   (The dataset name is always followed by a dot, and then the table name.)
3. But only return the data **`WHERE`** **`customer_id`** is greater than **`0`**, **`first_name`** is **`Tony`**, and **`last_name`** is **`Magnolia`**.


If only one customer is named Tony Magnolia, the results from the query could be:

customer_id	first_name	last_name
1967	Tony	Magnolia

If more than one customer has the same name, the results from the query could be:

customer_id	first_name	last_name
1967	Tony	Magnolia
7689	Tony	Magnolia

Capitalization, indentation, and semicolons

You can write your SQL queries in all lowercase and don't have to worry about extra spaces between words. However, using capitalization and indentation can help you read the information more easily. Keep your queries neat, and they will be easier to review or troubleshoot if you need to check them later on.

WHERE conditions

In the query shown above, the *SELECT* clause identifies the column you want to pull data from by name, *field1*, and the *FROM* clause identifies the *table* where the column is located by name, table. Finally, the *WHERE* clause narrows your query so that the database returns only the data with an exact value match or the data that matches a certain condition that you want to satisfy.
For example, if you are looking for a specific customer with the last name Chavez, the *WHERE* clause would be:
WHERE field1 = 'Chavez'
However, if you are looking for all customers with a last name that begins with the letters "Ch," the *WHERE* clause would be:
WHERE field1 LIKE 'Ch%'
You can conclude that the *LIKE* clause is very powerful because it allows you to tell the database to look for a certain pattern! The percent sign *%* is used as a wildcard to match one or more characters. In the example above,

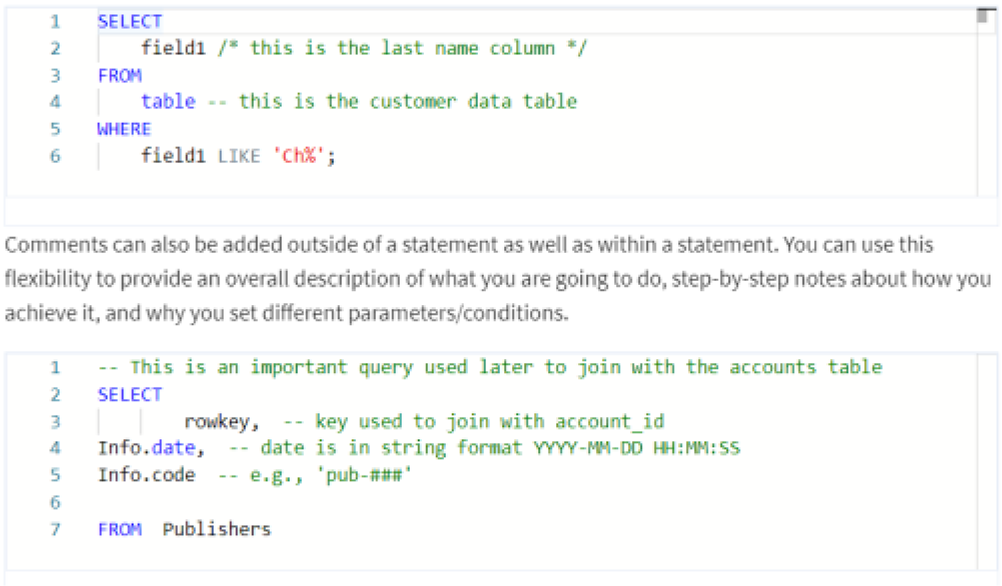
both **Chavez** and **Chen** would be returned. Note that in some databases an asterisk * is used as the wildcard instead of a percent sign *%*.

SELECT all columns

By using *SELECT ** we can select all coulumns.

Comments

Some tables aren't designed with descriptive enough naming conventions. In the example, *field1* was the column for a customer's last name, but you wouldn't know it by the name. A better name would have been something such as *last_name*. In these cases, you can place comments alongside your SQL to help you remember what the name represents. Comments are text placed between certain characters, /* and */, or after two dashes --) as shown below.



```sql
SELECT
    field1 /* this is the last name column */
FROM
    table -- this is the customer data table
WHERE
    field1 LIKE 'Ch%';
```

Comments can also be added outside of a statement as well as within a statement. You can use this flexibility to provide an overall description of what you are going to do, step-by-step notes about how you achieve it, and why you set different parameters/conditions.

```sql
-- This is an important query used later to join with the accounts table
SELECT
        rowkey,  -- key used to join with account_id
Info.date,  -- date is in string format YYYY-MM-DD HH:MM:SS
Info.code  -- e.g., 'pub-###'

FROM  Publishers
```

Example of a query with comments

Here is an example of how comments could be written in BigQuery:

```
-- Pull basic information from the customer table
SELECT
    customer_id, --main ID used to join with customer_addresss
    first_name, --customer's first name from loyalty program
    last_name --customer's last name
FROM
    customer_data.customer_name
```

Aliases

You can also make it easier on yourself by assigning a new name or **alias** to the column or table names to make them easier to work with (and avoid the need for comments). This is done with a SQL AS clause. In the example below, aliases are used for both a table name and a column. Within the database, the table is called `actual_table_name` and the column in that table is called `actual_column_name`. They are aliased as `my_table_alias` and `my_column_alias`, respectively. These aliases are good for the duration of the query only. An alias doesn't change the actual name of a column or table in the database.

Example of a query with aliases

```
SELECT
    my_table_alias.actual_column_name AS my_column_alias
FROM
    actual_table_name AS my_table_alias
```

Putting SQL to work as a data analyst

Imagine you are a data analyst for a small business and your manager asks you for some employee data. You decide to write a query with SQL to get what you need from the database.

You want to pull all the columns: **empID, firstName, lastName, jobCode,** and **salary.** Because you know the database isn't that big, instead of entering each column name in the `SELECT` clause, you use `SELECT *`. This will select all the columns from the Employee table in the `FROM` clause.

```
SELECT
    *
FROM
    Employee
```

Now, you can get more specific about the data you want from the **Employee** table. If you want all the data about employees working in the `'SFI'` job code, you can use a `WHERE` clause to filter out the data based on this additional requirement.

Here, you use:

```sql
SELECT
    *
FROM
    Employee
WHERE
    jobCode = 'SFI'
```

A portion of the resulting data returned from the SQL query might look like this:

empID	firstName	lastName	jobCode	salary
0002	Homer	Simpson	SFI	15000
0003	Marge	Simpson	SFI	30000
0034	Bart	Simpson	SFI	25000
0067	Lisa	Simpson	SFI	38000
0088	Ned	Flanders	SFI	42000
0076	Barney	Gumble	SFI	32000

Suppose you notice a large salary range for the `'SFI'` job code. You might like to flag all employees in all departments with lower salaries for your manager. Because interns are also included in the table and they have salaries less than $30,000, you want to make sure your results give you only the full time employees with salaries that are $30,000 or less. In other words, you want to exclude interns with the `'INT'` job code who also earn less than $30,000. The `AND` clause enables you to test for both conditions.

You create a SQL query similar to below, where <> means "does not equal":

```sql
SELECT
    *
FROM
    Employee
WHERE
    jobCode <> 'INT'
      AND salary <= 30000;
```

The resulting data from the SQL query might look like the following (interns with the job code **INT** aren't returned):

empID	firstName	lastName	jobCode	salary
0002	Homer	Simpson	SFI	15000
0003	Marge	Simpson	SFI	30000
0034	Bart	Simpson	SFI	25000
0108	Edna	Krabappel	TUL	18000
0099	Moe	Szyslak	ANA	28000

Plan a data visualization

Earlier, you learned that **data visualization** is the graphical representation of information. As a data analyst, you will want to create visualizations that make your data easy to understand and interesting to look at. Because of the importance of data visualization, most data analytics tools (such as spreadsheets and databases) have a built-in visualization component while others (such as Tableau) specialize in visualization as their primary value-add.

Steps to plan a data visualization

Step 1: Explore the data for patterns

First, you ask your manager or the data owner for access to the current sales records and website analytics reports. This includes information about how customers behave on the company's existing website, basic information about who visited, who bought from the company, and how much they bought.
While reviewing the data you notice a pattern among those who visit the company's website most frequently: geography and larger amounts spent on purchases. With further analysis, this information might explain why sales are so strong right now in the northeast—and help your company find ways to make them even stronger through the new website.

Step 2: Plan your visuals

Next it is time to refine the data and present the results of your analysis. Right now, you have a lot of data spread across several different tables, which isn't an ideal way to share your results with management and the marketing team. You will want to create a data visualization that explains your findings quickly and effectively to your target audience. Since you know your audience is sales oriented, you already know that the data visualization you use should:

- Show sales numbers over time
- Connect sales to location
- Show the relationship between sales and website use

- Show which customers fuel growth

Step 3: Create your visuals

Now that you have decided what kind of information and insights you want to display, it is time to start creating the actual visualizations. Keep in mind that creating the right visualization for a presentation or to share with stakeholders is a process. It involves trying different visualization formats and making adjustments until you get what you are looking for. In this case, a mix of different visuals will best communicate your findings and turn your analysis into the most compelling story for stakeholders. So, you can use the built-in chart capabilities in your spreadsheets to organize the data and create your visuals.

Line charts can track sales over time

Maps can connect sales to locations

Donut charts can show customer segments

Bar charts can compare total visitors and visitors that make a purchase

Build your data visualization toolkit

There are many different tools you can use for data visualization.

- You can use the visualizations tools in your spreadsheet to create simple visualizations such as line and bar charts.
- You can use more advanced tools such as Tableau that allow you to integrate data into dashboard-style visualizations.
- If you're working with the programming language R you can use the visualization tools in RStudio.

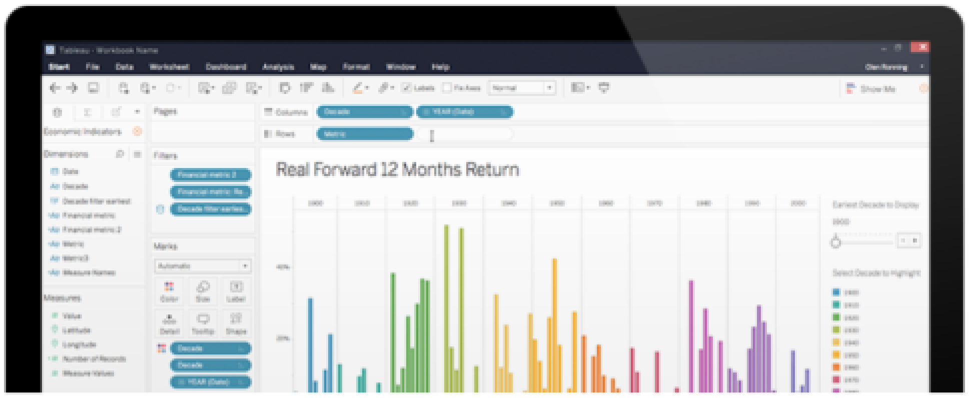
Your choice of visualization will be driven by a variety of drivers including the size of your data, the process you used for analyzing your data (spreadsheet, or databases/queries, or programming languages). For now, just consider the basics.

Spreadsheets (Microsoft Excel or Google Sheets)

In our example, the built-in charts and graphs in spreadsheets made the process of creating visuals quick and easy. Spreadsheets are great for creating simple visualizations like bar graphs and pie charts, and even provide some advanced visualizations like maps, and waterfall and funnel diagrams (shown in the following figures). But sometimes you need a more powerful tool to truly bring your data to life. Tableau and RStudio are two examples of widely used platforms that can help you plan, create, and present effective and compelling data visualizations.

Visualization software (Tableau)

Tableau is a popular data visualization tool that lets you pull data from nearly any system and turn it into compelling visuals or actionable insights. The platform offers built-in visual best practices, which makes analyzing and sharing data fast, easy, and (most importantly) useful. Tableau works well with a wide variety of data and includes an interactive dashboard that lets you and your stakeholders click to explore the data interactively.



Programming language (R with RStudio)

A lot of data analysts work with a programming language called R. Most people who work with R end up also using RStudio, an integrated developer environment (IDE), for their data visualization needs. As with Tableau, you can create dashboard-style data visualizations using RStudio.