
Analysis process

4 phases of analysis:

- Organize data
- Format and adjust data: by sorting and filtering
- Get input from others
- Transform data by observing relationships between data points and making calculations

Organize data: Sort and filter

In Sheets: [SORT function](#) (=SORT(range, column you want to sort by, TRUE (ascending) or FALSE (descending) , [FILTER function](#). Sort by range and sheets.

In excel: [SORT](#), [SORTBY](#), and [FILTER](#) functions and similar to sheets.

In SQL: Filter by using

- WHERE
- AND [condition]
- **ORDER BY** [column name] (in default, it sorts data in ascending order). If want to sort in descending, add 'DESC'
- LIMIT: return the number satisfy the required limitations

Commented [NL1]: Sorting helps you answer business questions that involve phrases such as “how much,” “how many,” “best,” or “worst”

Tips: to create table to a destination, before running the query,

MORE-> QUERY SETTINGS -> Create destination.

Once done, reset to setting to Save query results in a temporary table

Formatting data

Double check if the data is in the right format (string, %, currency, date, etc.)

In spreadsheets

CONVERT function: `CONVERT(value, start_unit, end_unit)`

- `value` - the numeric value in `start_unit` to convert to `end_unit`
- `start_unit` - The starting unit, the unit currently assigned to `value` .
- `end_unit` - The unit of measure into which to convert `value`.

Commented [NL2]: Eg: =CONVERT(B2, "C", "F").
Converting Celsius into Fahrenheit

Tips: When adding data to tables using a formula, go back and paste the data in as values afterwards

- String to date
- String to number
- Combining column: CONCAT or CONCATENATE (>=2 strings)
- Number -> %

VALUE: convert text into number

To **subtract** a string:

LEN: to find the total number of characters in a box

FIND: pull a substring (find at which character the desired substring starts)

LEFT, RIGHT

Data validation functions

It allows you to control what can and can't be entered in your worksheet.

In spreadsheet:

For options: Data menu -> **Data validation** -> + **Add rule** -> **Apply to range**

Add chatboxes: **In add rule, pick "chat box"**

Protect data and formulas: **Reject input**

Create custom tools: Format menu-> **Conditional formatting**

Transforming data in SQL

The CAST function: CAST(expression AS typename)

- Converting a string to a number: CAST(abc AS INT)
- Converting a date to a string: CAST(mydate AS STRING)
- Converting a date to a datetime (YYYY-MM-DD hh: mm: ss format): CAST (MyDate AS DATETIME)

The SAFE_CAST function: returns a value of *Null* instead of an error when a query fails. The syntax is the same for CAST.

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

GROUP BY to group together summary rows

CONCAT(a,b,c): combine strings (abc)

CONCAT_WS: adds two or more strings together with a separator (a,b,c)

CONCAT with +: Adds two or more strings together using the + operator ('Google' + '.com')

JOIN: Combine rows from >=2 tables based on a related column

[List of functions in SQL](#)

[SQL Keywords](#)

Get support in DA

Get stuck -> engage with other people (offline and online)

How to find answer online: choose the right term to search online, basic knowledge of tool

[Stack Overflow](#): where the data analyst ask code-related questions and peers are available to suggest answers.

Find the question: Eg: [SQL] Question

Write a question: specific, NO opinion-based

Problem: just sit back and revise your journey (how you approach a task)

Choose the right tool

Huge data (more than 1mil rows) -> SQL

R: statistical, data visualization, and others

Data aggregation

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

In spreadsheets

VLOOKUP:

- Prepare: numeric format, trim, duplicate
- Syntax: =VLOOKUP(data to look up, range, number of column, FALSE)
- Limitations:
 - VLOOKUP only returns the first match it finds (from the *right*), even if there are lots of possible matches. (VD: Tên client – Cột A -> data khác: cột khác)
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.

Commented [NL3]: Return the exact match

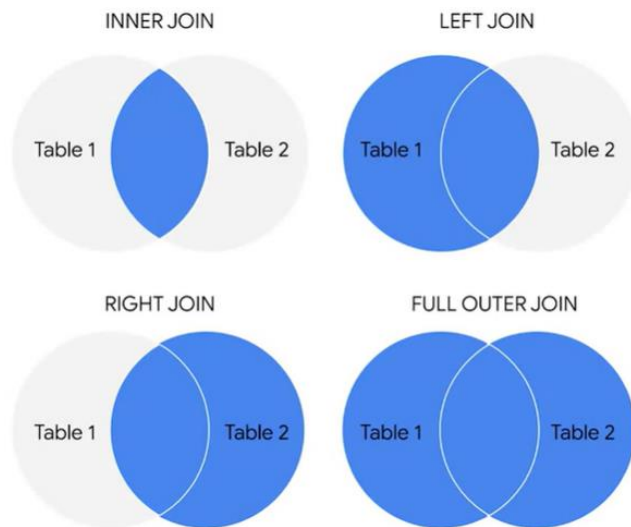
- Need to lock the range
- Lock the sheet to avoid someone else changing the lookup data (or use MATCH)
- ➔ Copying and pasting a column to the left of the data they want to look at.
- #N/A indicates that a matching value can't be returned. The error doesn't mean that anything is actually wrong with the data. But to replace the #N/A error with something more descriptive, like "Does not exist." => USE =IFNA(value, value_if_na)
Eg: =IFNA(#N/A, "Does not exist")

Troubleshooting questions

- 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?

In SQL

JOIN: combine rows from ≥ 2 tables based on a related column (SQL version of VLOOKUP)



Inner join:
default of JOIN

```
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
```

INNER JOIN

INNER JOIN returns records if the data lives in both tables.

```
SELECT
  customers.customer_name,
  orders.product_id,
  orders.ship_date
FROM
  customers
INNER JOIN
  orders
ON customers.customer_id = orders.customer_id
```

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.

LEFT JOIN

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.

Eg: Return all customer_name in customers table, give null values if customer_name doesn't match the orders table

COUNT and COUNT DISTINCT

Use any time you want to answer questions about “*how many*”

```
1 SELECT
2   COUNT(DISTINCT warehouse.state) as num_states
3 FROM
4   warehouse_orders.Orders orders
5 JOIN
6   warehouse_orders.Warehouse warehouse ON orders.warehouse_id = warehouse.warehouse_id
```

Note:

warehouse is the table name

Simplify SQL queries using Aliases

Aliases are used in SQL queries to create temporary names for a column or table.

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

```
SELECT column_name AS alias_name
FROM table_name;
```

Can refer in JOIN and directly type the referred table name in SELECT above

Eg:

```
SELECT
  edu.country_name,
  summary.country_code,
  edu.value
FROM
  bigquery-public-data.world_bank_intl_education.international_education AS edu
INNER JOIN
  bigquery-public-data.world_bank_intl_education.country_summary AS summary
ON edu.country_code = summary.country_code
```

References:

- **SQL Aliases:** This tutorial on aliasing is a really useful resource to have when you start practicing writing queries and aliasing tables on your own. It also demonstrates how aliasing works with real tables.
- **SQL Alias:** This detailed introduction to aliasing includes multiple examples. This is another great resource to reference if you need more examples.
- **Using Column Aliasing:** This is a guide that focuses on column aliasing specifically. Generally, you will be aliasing entire tables, but if you find yourself needing to alias just a column, this is a great resource to have bookmarked.

Subqueries

Eg:

```
SELECT
  id,
  name,
  number_of_rides AS number_of_rides_starting_station
FROM
  (
    SELECT
      start_station_id,
      COUNT(*) number_of_rides
    FROM
      bigquery-public-data.london_bicycles.cycle_hire
    GROUP BY
      start_station_id
  )
  AS station_num_trips
INNER JOIN
  bigquery-public-data.london_bicycles.cycle_stations ON id = start_station_id
ORDER BY
  number_of_rides DESC
```

Aggregate data in subqueries

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.

Tips:

- write the subqueries 1st, then others thing 2nd. Understand what you are doing! (by typing description)

- Comparison operators such as >, <, or = help you compare data in subqueries. You can also use multiple row operators including IN, ANY, or ALL.

Eg: `SELECT employeeID FROM employee. employee_name`

`WHERE`

`referenceID = (SELECT referenceID FROM employee.firstname`

`WHERE EmpID = 276)`

HAVING basically allows you to add a filter to your query instead of the underlying table when you're working with aggregate functions.

⇒ only returns records that meet your specific conditions.

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

CASE

`WHEN condition1 THEN result1`

`WHEN condition2 THEN result2`

`WHEN conditionN THEN resultN`

`ELSE result`

END;

HAVING ..

Data calculations

Some basic calculations

Spreadsheets

Conditional formatting: Use Color Scale to blend the color from min -> max value

COUNTIF(range, "value condition"): count number of cell satisfy the "value condition" (=1,>1..)

COUNTIFS(criteria_range1, criterion1, [criteria_range2, criterion2, ...])

SUMIF(condition_range, condition, [sum_range])

SUMIFS(sum_range, criteria_range1, criterion1, [criteria_range2, criterion2, ...])

SUMPRODUCT(array1, array2,..)

PIVOT tables:

A pivot table has four basic parts: rows, columns, values, and filters.

Commented [NL4]: range where we want to add the numbers is placed in the formula if that range is different from the range being evaluated.

The **rows** of a pivot table organize and group data you select horizontally

The **columns** organize and display values from your data vertically.

Values are used to calculate and count data. This is where you input the variables you want to measure.

A **calculated field** is a new field within a pivot table that carries out certain calculations based on the values of other fields

1. Reference should point to the source data, not to the Pivot table itself.
2. Use field labels in the source data instead of range references. For example, I've used 'price per unit' instead of the range D4:D9 in my formulas.
3. [Field labels](#) in the formula must be enclosed within single quotes.

[Pivot tables in Google Sheets](#)

[Customize a pivot table](#): sorting, filtering pivot tables in Google Sheets.

[Create and edit pivot tables](#): formatting your pivot table

In excel: Integrate the [Power Pivot add-on](#)
SQL

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

SUM

AVG

modulo (%) operator: the remainder of a division calculation (Eg: $10\%3 = 1$)

Note: Use '<>' or '!=' to represent A is 'not' B

EXTRACT: extract a part from a given date

GROUP BY: group rows that have the same values from a table to a summary rows

ORDER BY

Row	year	number_of_rides
1	2013	5037185
2	2014	8081216
3	2015	9937969
4	2016	10262649

Data validation in SQL

Control what can and cant be put -> check and recheck

Check if a calculation = data input: WHERE **Total_Bags != Total_Bags_Calc**

Types of data validation

Data type: Check that the data matches the data type defined for a field

Data range

Data constraints: Check that the data meets certain conditions or criteria for a field.

Data consistency: Check that the data makes sense in the context of other related data.

Eg: Data values for product shipping dates can't be earlier than product production dates.

Data structure: Check that the data follows or conforms to a set structure

Note: Even the data is consistence and follows the structure, it still can be wrong

Code validation: Check that the application code systematically performs any of the previously mentioned validations during user data input. (more than one data type allowed, data range checking not done, or ending of text strings not well defined, etc.)

Temporary tables – pre-processing data

Temporary tables, or temp tables, store subsets of data from standard data tables for a certain period of time.

Commented [NL5]: It is like a sticky note

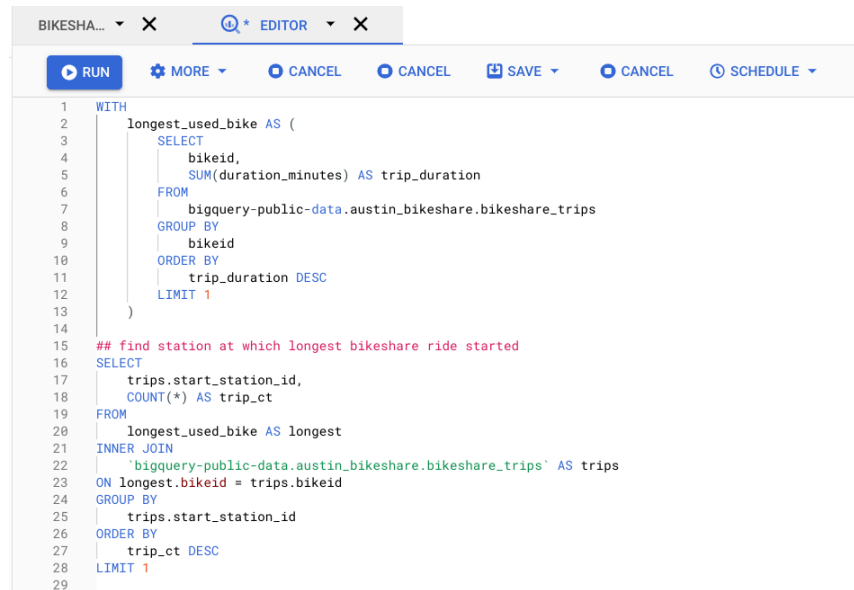
Create a tempt table that just exist in a code and will be deleted when end your SQL database session.

```
WITH longest_used_bike AS (  
  SELECT  
    bikeid,  
    SUM(duration_minutes) AS trip_duration  
  FROM `bigquery-public-data.austin_bikeshare.bikeshare_trips`  
  GROUP BY bikeid  
  ORDER BY trip_duration DESC  
)  
/* add your SELECT statement here */  
SELECT *  
FROM longest_used_bike;
```

Since the WITH just create a tempt table, we need to add SELECT to call that table out

If want to comeback to this tempt table -> query history

Since the temp table just have 2 column, which is the longest_used_bike and trip duration, if we want to analyse other information in the data base, we need to use JOIN



The screenshot shows a BigQuery SQL editor window titled 'BIKESHA...'. The editor contains a SQL query with a Common Table Expression (CTE) and an inner join. The query is as follows:

```
1 WITH  
2   longest_used_bike AS (  
3     SELECT  
4       bikeid,  
5       SUM(duration_minutes) AS trip_duration  
6     FROM  
7       bigquery-public-data.austin_bikeshare.bikeshare_trips  
8     GROUP BY  
9       bikeid  
10    ORDER BY  
11      trip_duration DESC  
12    LIMIT 1  
13  )  
14  
15  ## find station at which longest bikeshare ride started  
16  SELECT  
17    trips.start_station_id,  
18    COUNT(*) AS trip_ct  
19  FROM  
20    longest_used_bike AS longest  
21  INNER JOIN  
22    `bigquery-public-data.austin_bikeshare.bikeshare_trips` AS trips  
23  ON longest.bikeid = trips.bikeid  
24  GROUP BY  
25    trips.start_station_id  
26  ORDER BY  
27    trip_ct DESC  
28  LIMIT 1  
29
```

Note: Instead of using the WITH clause, you can use the SELECT INTO or the CREATE TABLE clauses.

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

The **SELECT INTO** clause copies data from one table into a new table, but doesn't add the new table to the database. It's useful if you want to make a copy of a table with a specific condition.

Note: It not supported in BigQuery, but other version of SQL (SQL Server, MySQL, etc.)

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

The **CREATE TABLE** clause is a good option when several people need to access the same temp table. This statement adds the table into the database.

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,
  ....
)
```

More information: [Choosing Between Table Variables and Temporary Tables](#)

[BigQuery Documentation for Temporary Tables](#)

[SQL Server Temporary Tables](#)

Connected sheet with Query
Connected Sheets integrates both BigQuery and Google Sheets, allowing the user to analyze billions of rows of data in Sheets without any need for specialized knowledge, such as SQL.



- [Get started with BigQuery data in Google Sheets](#)