
NOVEMBER 4-5, 2026
ANALYTICS
SUMMIT
in partnership with Google
20+
12
INDUSTRIES REPRESENTED
ANALYTICS
SUMMIT

Who we are



Julian Holzner
Senior BI Analyst



Nassim Gharbi
Senior BI Analyst

SEMINARS

BigQuery and SQL: Hands-on Intro (Part I)

HANDS-ON EDITION

ANALYTICS SUMMIT

Housekeeping



Wifi

Please log into this Wifi `OMCGroup-Guest`

You will be asked to create an account



Slides

You will receive all the slides, including solutions after the seminar.

To take notes and copy code: `analytics-summ.it/slides`



Access to course data

You have been given access to our training data set `as_sql_basics`: please make sure you're logged in with the right Google account.

To test access, please open: `analytics-summ.it/data`

AI writes SQL in seconds.
So why are you here today?

You

How many tickets did we sell in October?

AI

Here’s your query: you sold 296 tickets.

```sql
SELECT
  COUNT(*) AS tickets_sold
FROM `trkkn-ext-bq-trainings.as_sql_basics.orders`
WHERE order_date BETWEEN '2026-10-01' AND '2026-10-31';
-- ✅ runs fine, returns 296
```

THE CATCH

The query runs. The number is wrong. Would you notice?

| ***TRKKN*** *Proprietary & Confidential*

AI types the SQL. You own the answer.

AI is great at	What stays your job	Because AI ...
Writing syntax	01 **Asking the right question**	guesses when the question is vague
Remembering functions	02 **Knowing the data: tables, rows, definitions**	doesn't know your data
Spotting patterns	03 **Knowing the business: context & judgment**	doesn't know your business, even with your data
Explaining errors	04 **Checking the number**	makes mistakes that look plausible
Drafting a first version	05 **Being accountable**	does not sit in the meeting

| ***TRKKN*** *Proprietary & Confidential*

So today you learn to ...

01

Grasp the logic of a query.

Know what a query does without reading every line.

02

Break questions into steps.

One step = one query = one check.

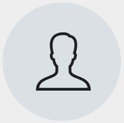
03

Check every result.

Prove the number before you share it.

A robust way to answer data questions with AI.

Quick round: who’s in the room?



Who are you?

Your name and role.



SQL so far?

Never, a bit, or daily?



Your AI tool?

Which one do you use at work?

| ***TRKKN*** *Proprietary & Confidential*

Your learning journey

overall path

Part 1: Build the foundation (01-06)

Part 2: From question to answer (07-08)



FINAL CASE **Online marketing case**

chapter by chapter

01 Understand your data
BigQuery basics, tables, what one row means

02 Get the numbers
`SELECT`, `WHERE`, `DISTINCT`, `COUNT`, `SUM`

03 Turn rows into insights
`GROUP BY`, `ORDER BY`, calculated metrics

04 Apply business rules
`CASE WHEN`

05 Connect data reliably
`LEFT JOIN`, `NULL`, duplicate keys

06 Break down complex questions
CTEs, thinking in steps

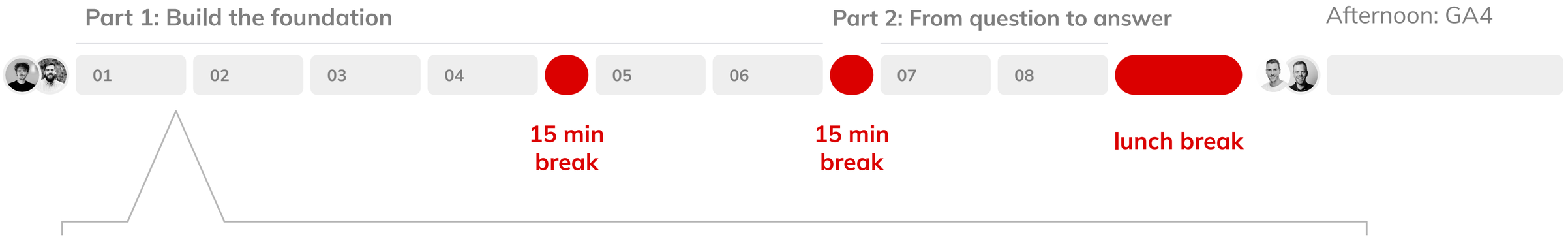
07 Use AI without losing trust
Better prompts, checking AI results

08 Practical case
ROAS, budget recommendation

| ***TRKKN*** *Proprietary & Confidential*

What to expect

overall structure



chapter structure

1 Concept & syntax

Short input: the SQL building block and how it works.

2 On our data

Seeing what we just learned in action on the Ticket Store tables.

3 Hands-On

Write, debug, self-check with check_answer.

How we practice today

Every chapter ends with a hands-on block. There are two kinds of tasks:

WRITE Answer a marketing question with your own query.

DEBUG An AI wrote a query that runs, but gives the wrong answer. Find and fix the bug.

Then self-check: copy and paste the line below, add your result, run it.

Is my answer to task 2.1 right?

```
SELECT `trkkn-ext-bq-trainings.as_sql_basics.check_answer`('2.1', 1234)
```

✅ **Task 2.1: correct**

❌ **Task 2.1: not quite yet, check your query and try again**

BONUS Extra task for the fast ones. Everyone else skips it.

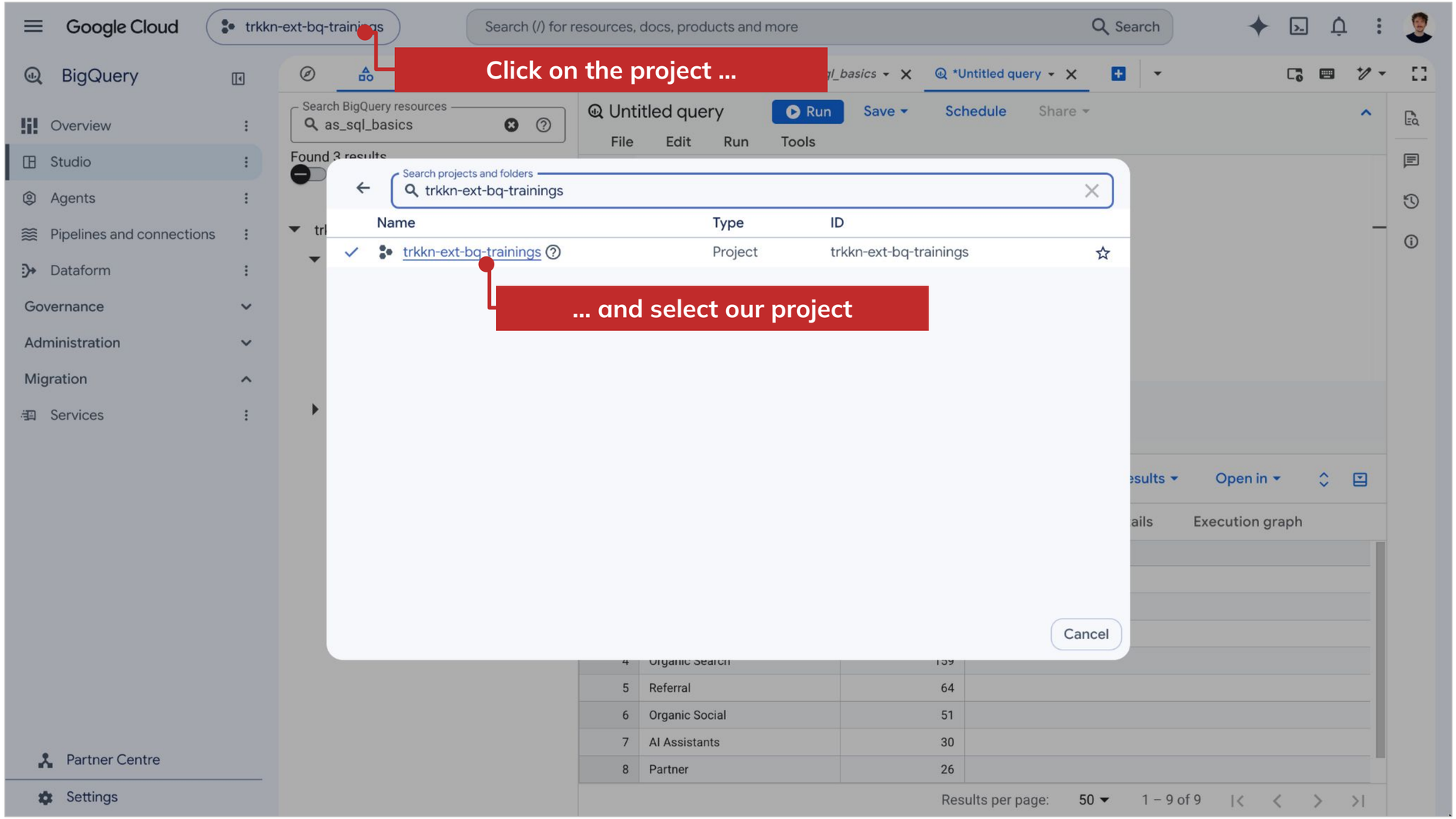
01

Understand your data

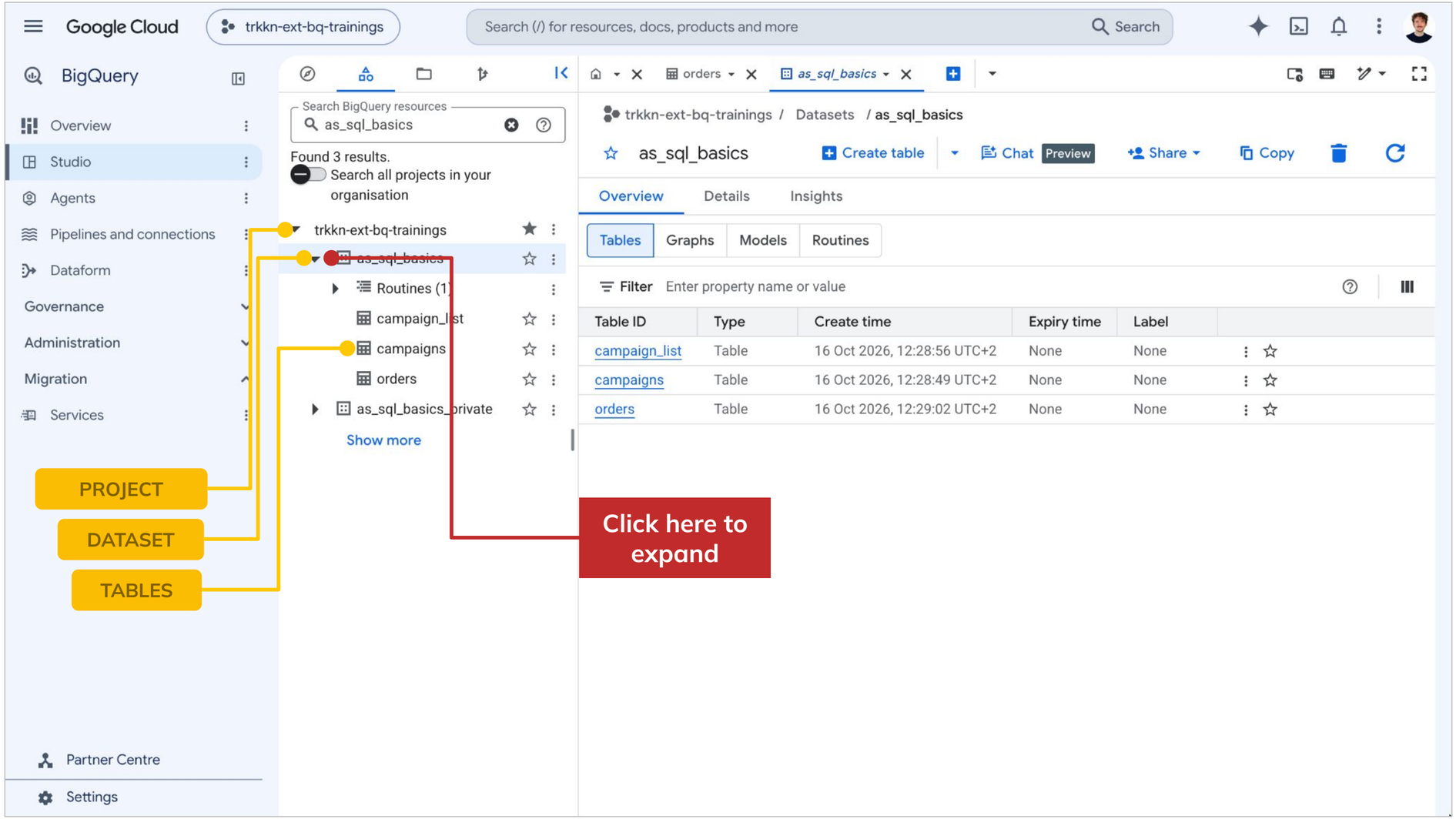
The BigQuery interface & our data

ANALYTICS SUMMIT

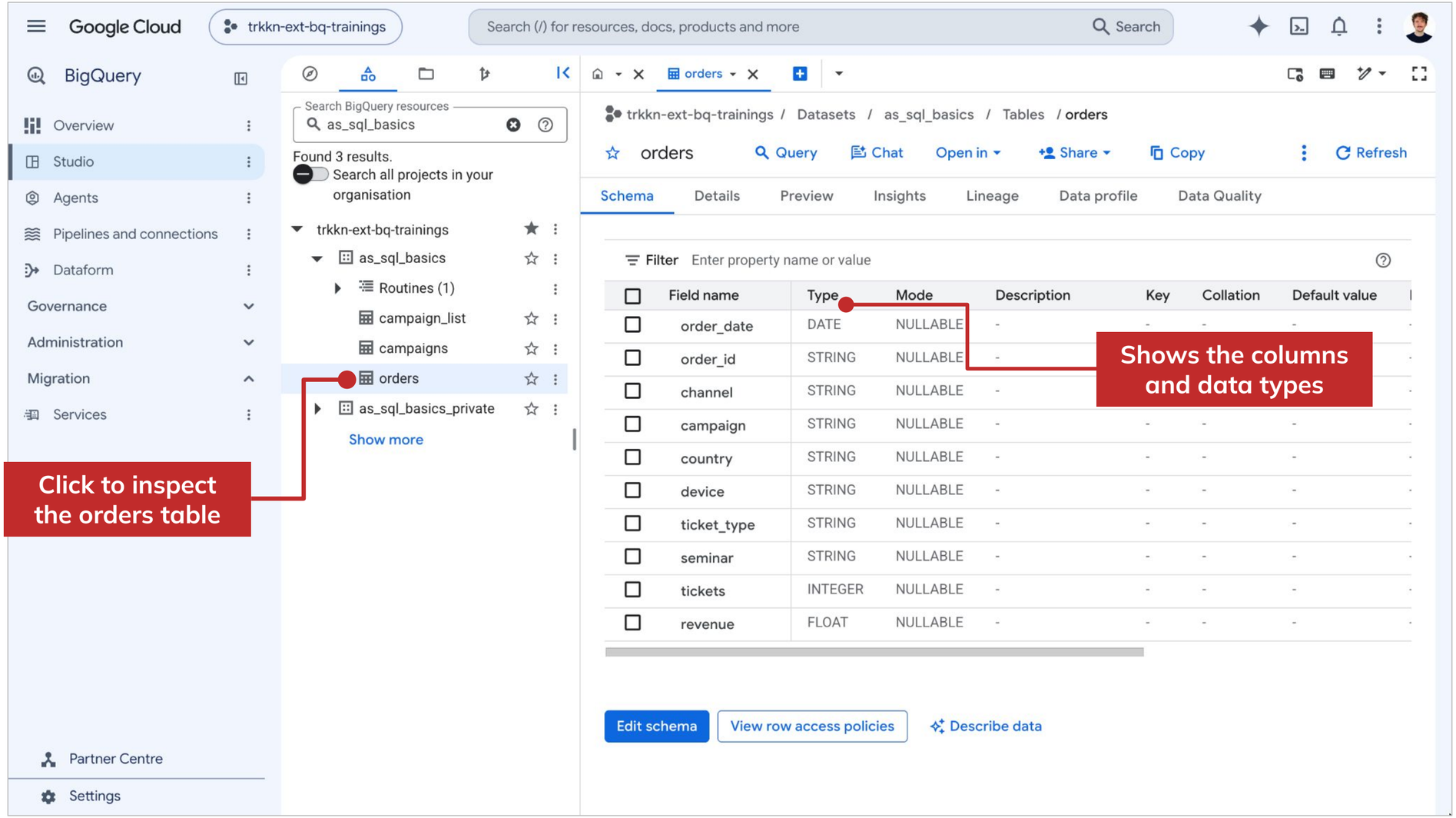
BigQuery Walkthrough: Select the right project



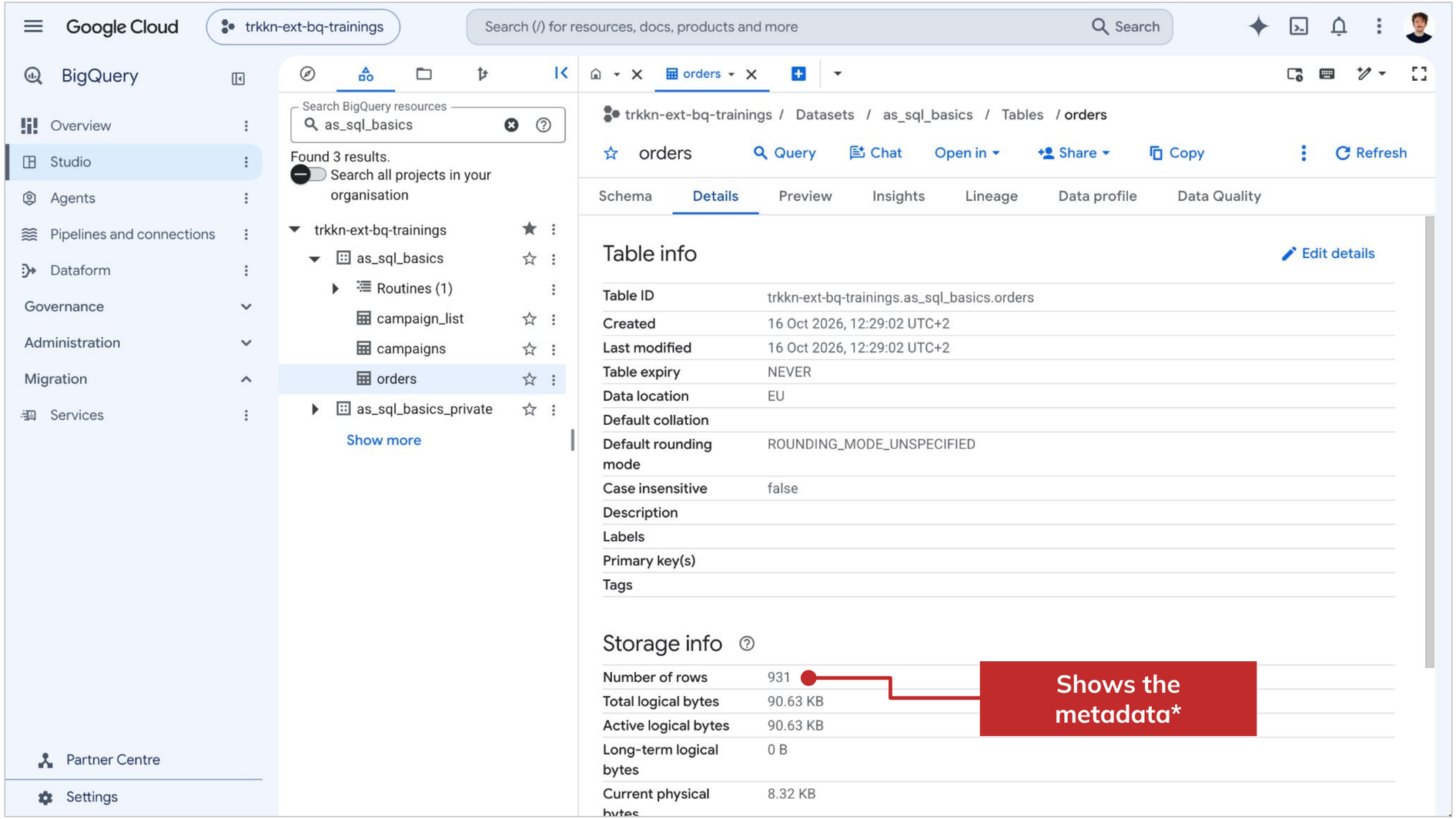
BigQuery Walkthrough: BQ Hierarchy



BigQuery Walkthrough: Inspecting Datasets & Tables

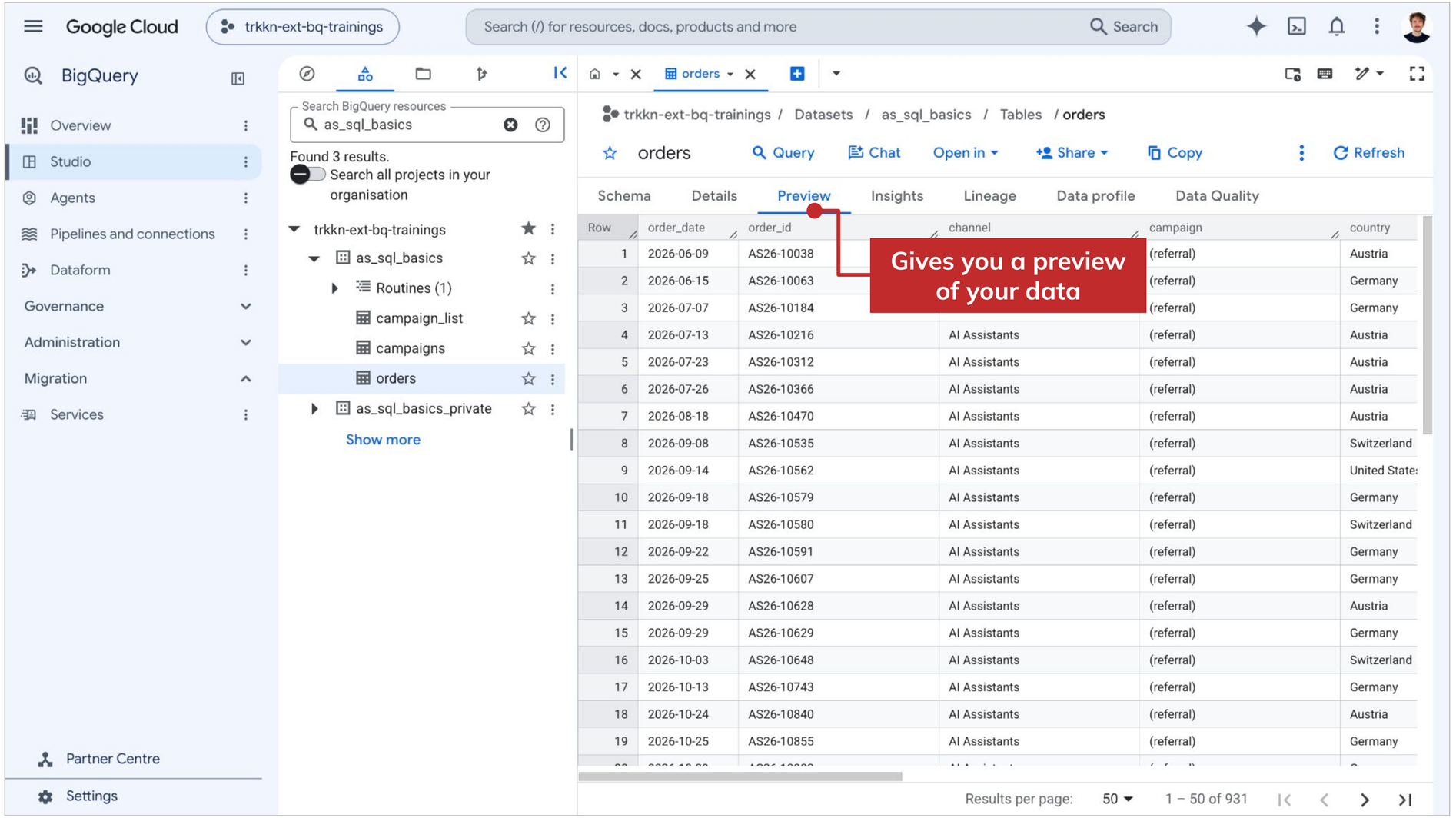


BigQuery Walkthrough: Inspecting Metadata



* Metadata: data about the table, e.g. its size, row count and creation date.

BigQuery Walkthrough: Preview data without querying



One logic for every query: SELECT … FROM …

SELECT says what you want, FROM says where it lives. Don’t worry about the details: in a moment you’ll simply run a query, and in chapter 2 we build this up step by step.

Show me the rows of the orders table

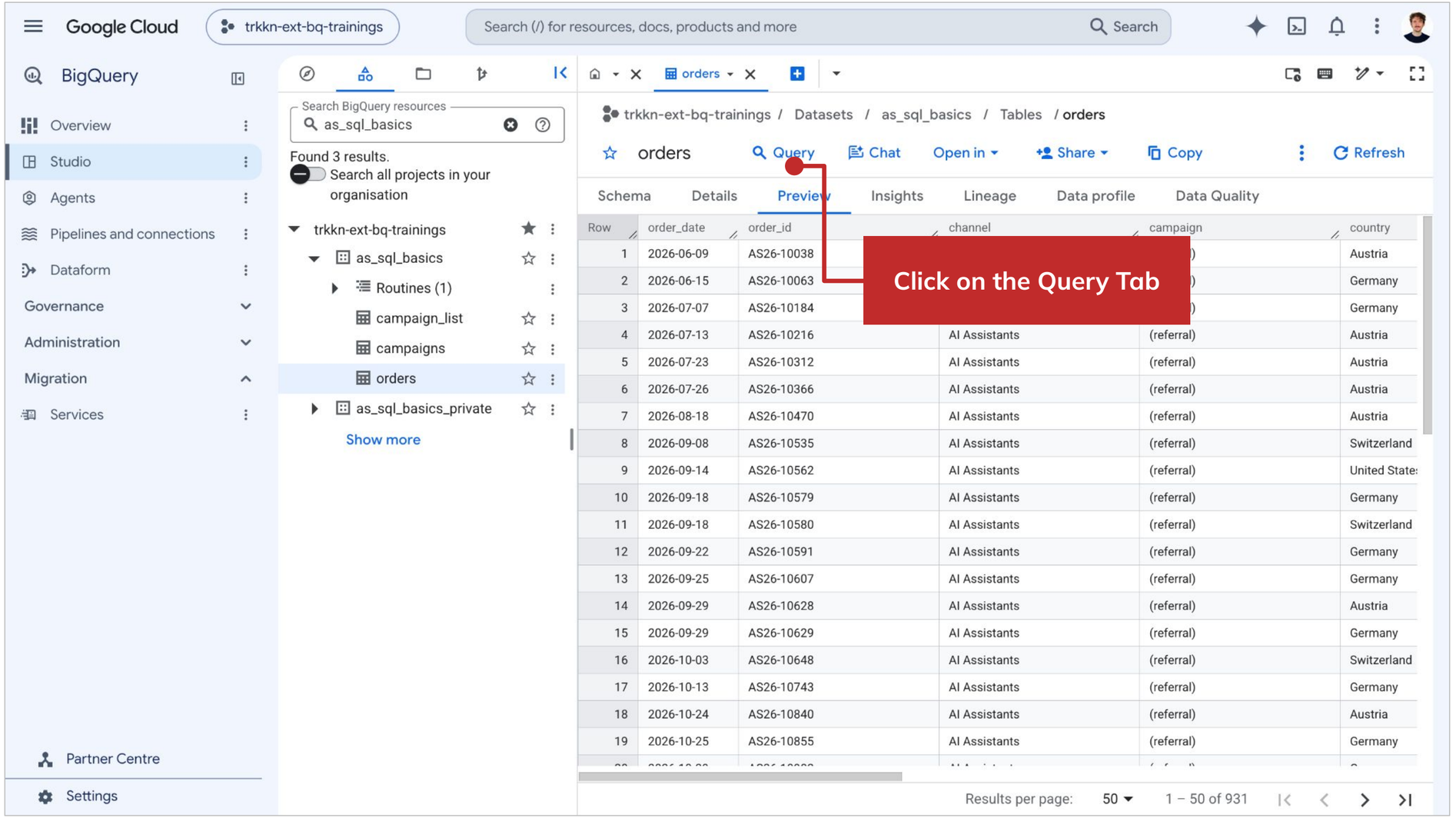
```
SELECT
  *
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`;
```

How many rows does the orders table have? COUNT(*) counts them.

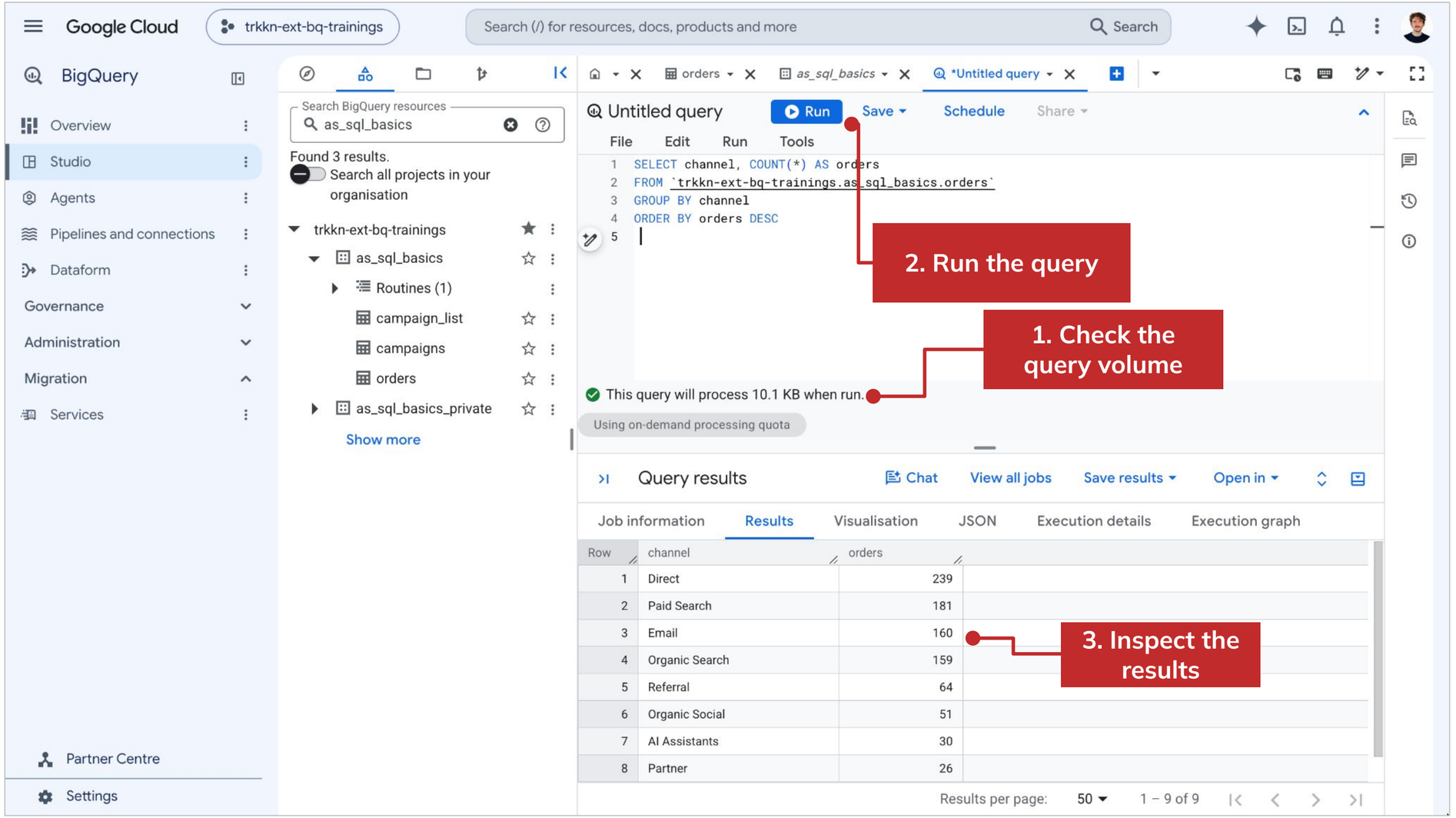
```
SELECT
  COUNT(*)
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`;
```

| ***TRKKN*** *Proprietary & Confidential*

BigQuery Walkthrough: Let's run our first query



BigQuery Walkthrough: Let's run our first query



Meet our data: the Analytics Summit Ticket Store

campaigns

date
campaign
cost
impressions
clicks

Jun to Oct 2026

1 row = 1 campaign on 1 day

orders

order_date
channel
campaign
country
device
ticket_type
seminar
tickets
revenue

1 row = ?

campaign_list

campaign
channel
owner
target_roas

Lookup list: which channel and owner each campaign belongs to. Uploaded by us as a CSV.

1 row = 1 campaign (in theory)

Dataset `trkkn-ext-bq-trainings.as_sql_basics`

The holy grain: What is one row?

Grain = the level of detail of a table. Same data, different grain → different answers.
Before you count, sum or join: ask "what is one row?"

campaigns (one row = one campaign on one day)

date	campaign	cost
2026-06-01	Search – Brand	120 €
2026-06-02	Search – Brand	140 €
2026-06-03	Search – Brand	130 €

→

the same data per campaign (one row = one campaign)

campaign	cost
Search – Brand	390 €

`COUNT(*) on the left returns 3: three campaign days, but only one campaign. (example values)`

Cheat sheet: Understand your data

Look before you query: no SQL needed to understand a table.

SCHEMA DETAILS PREVIEW

TASK 1.1

1. Look at the table first

- Schema: which columns exist, with which data types?
- Details: how many rows does the table have?
- Preview: what do real rows look like?
- All three are free: no query, no cost.

THE HOLY GRAIN

TASK 1.2

2. Ask: what is one row?

- Finish the sentence: 1 row = 1 ___.
- In **`campaigns`**, 1 row = 1 campaign on 1 day.
- Everything you count or sum depends on this answer.
- Not sure? Compare two similar rows: what differs?

BIGQUERY HIERARCHY

Every table has an address

```
`trkkn-ext-bq-trainings    -- 1. Project
  .as_sql_basics           -- 2. Dataset
  .orders`                 -- 3. Table
```

1. **`PROJECT`** Billing & permissions container
2. **`DATASET`** Folder grouping related tables
3. **`TABLE`** Grid of rows and typed columns

CHECK_ANSWER SAVE QUERIES

TASK 1.3

3. Prove your answer, keep your query

- Copy, add your number, run:

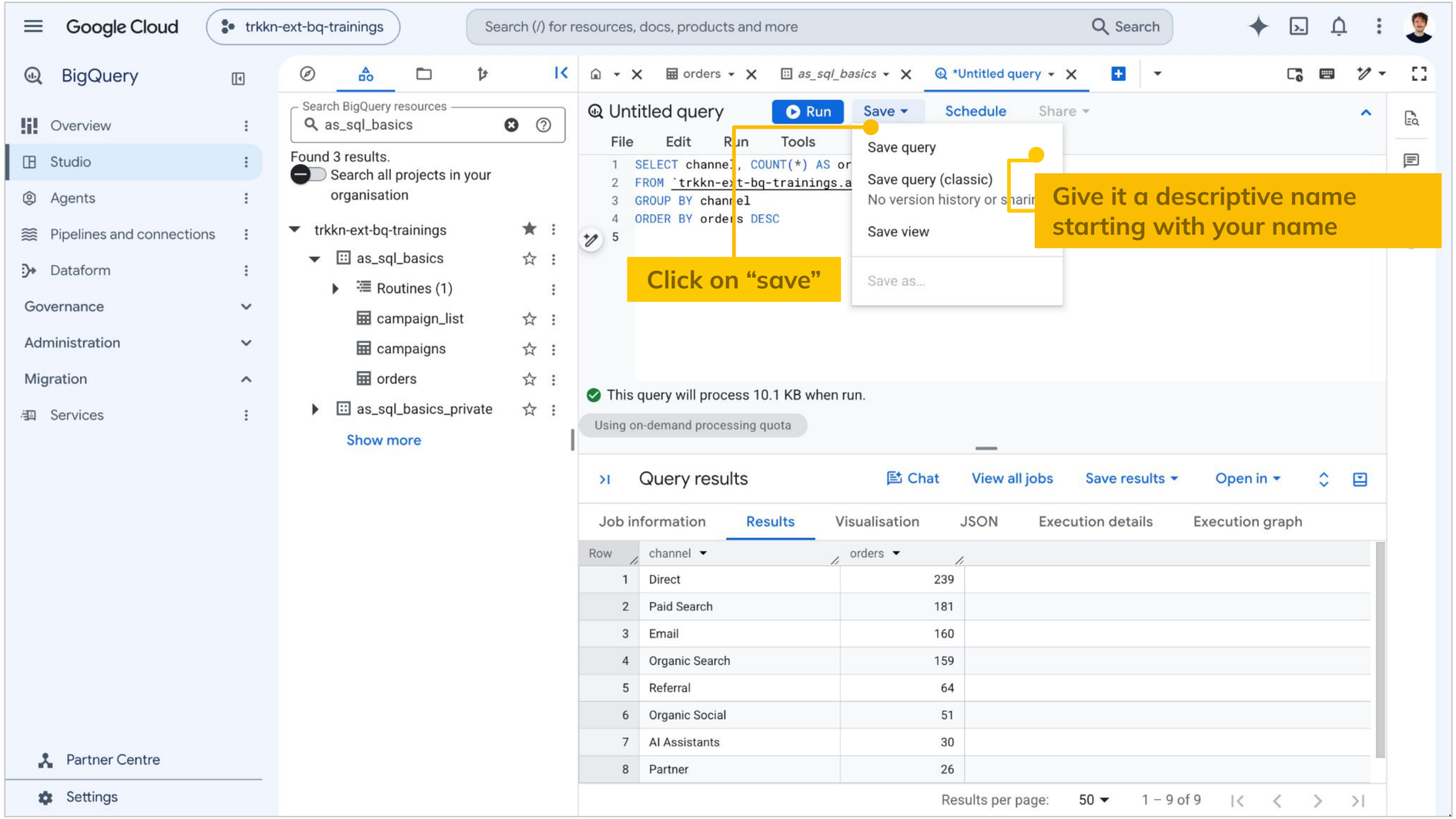
```
SELECT `trkkn-ext-bq-trainings.as_sql_basics.check_answer`('1.1', 123)
```

- ✅ means move on. ❌ means look again, nothing is lost.
- Click Save and start the title with your name, e.g. **`julian_1_1`**.



Admin & Hands-On

Admin: Saving your queries



Hands-On: Explore the orders table

6 MIN

1.1 Open the `orders` table and look at the **Schema**, **Details** and **Preview** tabs (preview is free, no query needed). How many rows does the table have?

1.2 What does one row represent? What is the difference between `tickets` and an order?

1.3 Run your first self-check: copy this line and put in your number:

```
SELECT `trkkn-ext-bq-trainings.as_sql_basics.check_answer`('1.1', 123)
```

Save your query and start its title with your name, e.g. “julian_query_check”.



02

Get the numbers

Select, filter, count

ANALYTICS SUMMIT

What's SQL?

STRUCTURED QUERY LANGUAGE SQL (pronounced "sequel") is the standard language for asking questions to a database and getting a clean table back.

THE QUESTION

What do I want to know?

"How many tickets did we sell in July?"

You start with a concrete business question and translate it into a clear instruction for the database:
- Which metric (tickets sold)
- Which timeframe (July 2026)

→

THE DATA SOURCE

Where do I find this info?

`orders (931 rows, matched!)`

`campaigns (873 daily rows)`

`campaign_list (11 rows)`

Filters and aggregates* across all tables in seconds, without changing the raw data.

→

THE RESULT

What comes out?

tickets_sold
410

Every step is written down in the query, so anyone can read, re-run, and verify the number.

* Aggregate: combine many rows into one number, e.g. a sum or a count.

The anatomy of a table

TABLE `orders`

COLUMN

order_date	channel	campaign	country	device	ticket_type	seminar	tickets	revenue
2026-06-11	Email	Newsletter – Early Bird	Netherlands	mobile	Combi	Google Cloud FinOps in Action	1	749.0
2026-06-09	Paid Search	Search – Brand	Germany	desktop	Conference		2	1198.0
2026-06-04	Direct	(none)	Germany	desktop	Seminar	How to Activate GA Raw Data in BigQuery	1	349.0
2026-07-02	Paid Social	LinkedIn – Early Bird	Germany	desktop	Conference		3	1797.0

VALUE

ROW

A table is a grid: every column holds one kind of information, every row is one record (here, one order).

Different fields, different data types

TABLE `orders`

DATE	STRING	STRING	STRING	STRING	STRING	STRING	INTEGER	FLOAT
order_date	**channel**	**campaign**	**country**	**device**	**ticket_type**	**seminar**	**tickets**	**revenue**
2026-06-11	Email	Newsletter – Early Bird	Netherlands	mobile	Combi	Google Cloud FinOps in Action	1	749.0
2026-06-09	Paid Search	Search – Brand	Germany	desktop	Conference		2	1198.0
2026-06-04	Direct	(none)	Germany	desktop	Seminar	How to Activate GA Raw Data in BigQuery	1	349.0
2026-07-02	Paid Social	LinkedIn – Early Bird	Germany	desktop	Conference		3	1797.0

INTEGER **FLOAT**

Whole numbers (tickets = 2) and decimals (revenue = 749.0). Required for SUM & AVG.

STRING

Any text value, always written in single quotes in SQL (e.g. channel = 'Email').

DATE

Calendar dates formatted as YYYY-MM-DD (2026-06-11). Filter date ranges with `BETWEEN`.

BOOLEAN

True or false values, written without quotes. Our tables do not use this type.

NULL

An empty value, not a type. seminar is NULL on Conference orders. Test with `IS NULL`, never `= NULL`.

Selecting columns with SELECT & FROM

Databases don’t know our data needs and require explicit instructions about what data to fetch. SQL `SELECT` queries let us specify precisely what we need.

A basic `SELECT` query uses two essential keywords:
`SELECT` (what data are we fetching?) and `FROM` (which table are we fetching it from?).

```sql
SELECT
  order_date,
  channel,
  ticket_type,
  revenue
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`;
```

PROJECT DATASET TABLE

```sql
SELECT
  *
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`
LIMIT 50;
```

`LIMIT 50` at the very end of a query returns only the first 50 rows. Handy for a quick look, but BigQuery still scans the whole table.

Filtering rows with a WHERE clause

You can add a `WHERE` clause to your `SELECT` statement to filter the results and show only the rows that match the conditions you specify.

I want to see orders from email campaigns only

```
SELECT
  *
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`
WHERE
  channel = 'Email';
```

Sophisticated Filtering with IN, OR, AND

Sometimes you need to filter data using multiple conditions or check against several values at once. SQL provides powerful tools for this.

I want to see orders from DE or AT

```sql
SELECT *
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`
WHERE
  country IN ('Germany', 'Austria');
```

the **IN** operator lets you match against a list of values

I want to see big orders: Combi tickets or more than one ticket

```sql
SELECT *
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`
WHERE
  ticket_type = 'Combi'
  OR tickets > 1;
```

OR allows rows to meet any of several conditions

I want to see seminar orders from outside Germany

```sql
SELECT *
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`
WHERE
  country != 'Germany'
  AND ticket_type = 'Seminar';
```

AND requires all conditions to be true

| *TRKKN Proprietary & Confidential*

Filtering date ranges with BETWEEN

Use `BETWEEN ... AND ...` in your `WHERE` clause to filter a date column to a specific range (both start and end dates are included).

I'd like to see all orders from July 2026

Result (first 3 rows)

order_date	channel	revenue
2026-07-01	Direct	1198.0
2026-07-01	Direct	1198.0
2026-07-01	Direct	599.0

```sql
SELECT
  order_date,
  channel,
  revenue
FROM `trkkn-ext-bq-trainings.as_sql_basics.orders`
WHERE order_date BETWEEN '2026-07-01' AND '2026-07-31';
```

| ***TRKKN*** *Proprietary & Confidential*

Getting unique values with DISTINCT

I want to see all countries we sold tickets to.

```
SELECT
  country
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`;
```

When you select a column without any special keywords, SQL returns every single row, including duplicates, for example: *Germany, Germany, Germany, United Kingdom, Germany, Austria, United Kingdom.*

```
SELECT
  DISTINCT country
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`;
```

The `DISTINCT` keyword eliminates these duplicates, returning only unique values. So the same query with `DISTINCT` would give you just: *Germany, United Kingdom, Austria*.

Sorting data with ORDER BY

I want to see the orders in chronological order.

```sql
SELECT *
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`
ORDER BY order_date ASC;
```

I want to see the biggest orders first

```sql
SELECT *
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`
ORDER BY revenue DESC;
```

The `ORDER BY` clause sorts your query results based on one or more columns, letting you arrange data in ascending (`ASC`, default) or descending (`DESC`) order to make it easier to analyze and interpret. It works alphabetically as well as numerically.

| ***TRKKN*** *Proprietary & Confidential*

Counting rows with COUNT

The `COUNT(*)` function returns the total number of rows in your query results, making it useful for quickly determining how many records meet your filter conditions or exist in a table. `AS` gives the result column a readable name.

I'd like to know the total number of orders

```
SELECT
  COUNT(*) AS orders
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`;
```

I'd like to know how many countries we sold tickets to

```
SELECT
  COUNT(DISTINCT country)
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`;
```

Further aggregation functions

SQL provides aggregate functions to calculate across many rows: `SUM` adds up, `AVG` calculates the average, `MIN` / `MAX` find the smallest / largest value.

Total ad spend

```
SELECT
  SUM(cost)
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.campaigns`;
```

Average daily cost

```
SELECT
  AVG(cost)
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.campaigns`;
```

Highest daily cost

```
SELECT
  MAX(cost)
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.campaigns`;
```

Lowest daily cost

```
SELECT
  MIN(cost)
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.campaigns`;
```

| ***TRKKN*** *Proprietary & Confidential*

Cheat sheet: Get the numbers

Listen to the question: it tells you which keyword you need.

COUNT(*) | SUM | MIN / MAX | AVG

TASKS 2.1 + 2.3

How much? How many? How high?

- “In total” means adding up a column: **`SUM(column)`**
- “On average” means the mean per row: **`AVG(column)`**
- “Highest” or “lowest” means one single value: **`MAX(column)`**, **`MIN(column)`**
- “How many orders” means counting rows: **`COUNT(*)`**. Ask first what one row is.

DISTINCT

TASK 2.2

How many different?

- Each value counted once: **`COUNT(DISTINCT column)`**
- Just want the list? **`SELECT DISTINCT column`**
- A plain **`COUNT(column)`** counts every filled row, duplicates included.

QUERY BLUEPRINT

Every query, same order

```
SELECT
  order_date, channel, revenue
FROM `trkkn-ext-bq-trainings.as_sql_basics.orders`
WHERE channel = 'Email'
ORDER BY revenue DESC
```

Name your columns instead of **`SELECT *`**: easier to read, and BigQuery scans less data.

WHERE | IN | AND / OR | BETWEEN

TASK 2.D

Only some rows?

- One of several values: **`IN ('a', 'b')`**
- A date range, both dates included: **`BETWEEN 'from' AND 'to'`**
- Text goes in single quotes: **`channel = 'Email'`**
- Mixing **`AND`** and **`OR`**? Read your **`WHERE`** out loud: does it say what the question asks?



Hands-On

Hands-On: SELECT, WHERE & aggregations

9 MIN

Use the `campaigns` table (and `orders` for 2.D):

2.1 WRITE — How much did all campaigns cost in total? Enter EUR with 2 decimals.

2.2 WRITE — How many different campaigns are in the table?

2.3 WRITE — What was the highest cost of a single campaign on a single day? BONUS

2.D DEBUG — An AI wrote a query on `orders` to count mobile orders from Austria or Switzerland. It returns 146. Find and fix the bug (next slide).

Copy & check every answer:

```
SELECT `trkkn-ext-bq-trainings.as_sql_basics.check_answer`('2.1', <result>)
```

Debug 2.D: find the bug

3 MIN

You

How many orders were placed on mobile by people from Austria or Switzerland?

AI

Here’s your query: 146 mobile orders from Austria or Switzerland.

```
SELECT
  COUNT(*) AS orders
FROM `trkkn-ext-bq-trainings.as_sql_basics.orders`
WHERE country = 'Austria'
  OR country = 'Switzerland'
  AND device = 'mobile'
```

SANITY CHECK

Austria and Switzerland only have 217 orders together, and most people buy on desktop. Suspicious?

HINT

how do AND and OR combine?

03

Turn rows into insights

Group & calculate

ANALYTICS SUMMIT

Counting rows per group with COUNT(*) + GROUP BY

The `GROUP BY` clause organizes rows into groups based on shared column values, allowing you to count how many rows belong to each group using `COUNT`.

I'd like to know the number of orders per channel

Result (first 3 of 9 rows)

channel	orders
Direct	239
Paid Search	181
Email	160

```
SELECT
  channel,
  COUNT(*) AS orders
FROM

`trkkn-ext-bq-trainings.as_sql_basics.orders`
GROUP BY
  channel;
```

| *TRKKN Proprietary & Confidential*

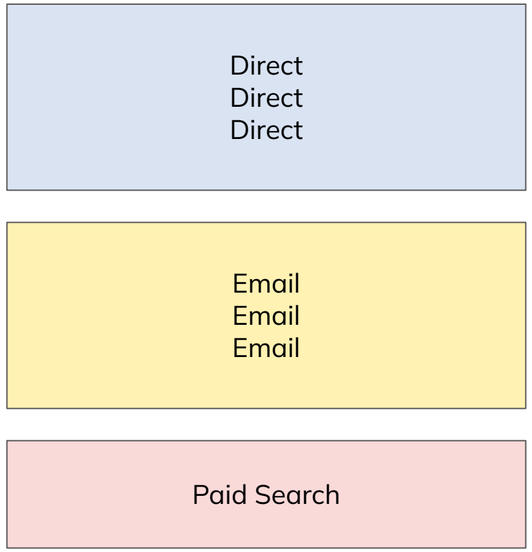
What GROUP BY does to your rows

GROUP BY channel puts all rows with the same channel into one group. COUNT(*) then counts the rows in each group.

1. Every row of orders (example)

order_date	channel
2026-06-01	Direct
2026-06-01	Email
2026-06-02	Direct
2026-06-02	Email
2026-06-03	Paid Search
2026-06-03	Email
2026-06-04	Direct

2. GROUP BY channel



3. COUNT(*) per group

channel	orders
Direct	3
Email	3
Paid Search	1

```
SELECT channel, COUNT(*) AS orders FROM `trkkn-ext-bq-trainings.as_sql_basics.orders` GROUP BY channel
```

GROUP BY needs to list all non-aggregated columns

Every column in `SELECT` that is **not** inside an aggregation (`COUNT`, `SUM`, ...) must also appear in `GROUP BY`. If you forget one, BigQuery throws an error: a helpful reminder!

```
SELECT
  channel,
  device,
  COUNT(*) AS orders
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`
GROUP BY
  channel,
  device;
```

Comments in a Query

Use `--` to add a comment: everything after it is ignored. Comments explain why, for colleagues, your future self and AI.

```sql
-- orders per channel
SELECT
  channel,
  COUNT(*) AS orders  -- one row = one order
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`
GROUP BY
  channel;
```

Cheat sheet: Turn rows into insights

Before you write: per what? And count rows or sum a column?

COUNT(*) vs. SUM()

TASKS 3.2 + 3.D

Count rows or sum a column?

- First ask what one row is. In **`orders`**, 1 row = 1 order.
- Orders are rows: **`COUNT(*)`**
- Tickets, clicks and revenue are values in a column: **`SUM(column)`**
- Quick check: can one row hold more than one of what you count?

THE GROUP BY RULE

TASK 3.1

Per what?

- “Per campaign”: that column goes into **`SELECT`** and **`GROUP BY`**.
- No “per” in the question? No **`GROUP BY`**.
- An error? A column is missing in **`GROUP BY`**.

GROUP BY BLUEPRINT

Your GROUP BY blueprint

```
SELECT
  channel,
  COUNT(*) AS orders,
  SUM(tickets) AS tickets_sold
FROM `trkkn-ext-bq-trainings.as_sql_basics.orders`
GROUP BY channel
ORDER BY tickets_sold DESC
```

AS ALIAS

-- COMMENTS

TASK 3.3

Ratios, rankings, names

- Ratio: divide the totals, **`SUM(a) / SUM(b)`**
- Ranking: **`ORDER BY … DESC`**, the answer is in the first row
- **`AS`** names a result, so you can sort by it
- **`--`** leaves a note for your future self and for AI

| *TRKKN Proprietary & Confidential*



Hands-On

Hands-On: GROUP BY & dates

9 MIN

3.1 WRITE

`campaigns`: Which campaign got the most clicks in total? Enter its number of clicks.

3.2 WRITE

`orders`: What was the total revenue in July 2026 ('2026-07-01' to '2026-07-31')? Enter EUR with 2 decimals.

3.3 WRITE

`campaigns`: What was the average cost per click (CPC) over all campaigns? = total cost / total clicks. Enter 2 decimals.

BONUS

3.D DEBUG

An AI query says 296 tickets were sold in October. Find and fix the bug (next slide).

Copy & check every answer:

```
SELECT `trkkn-ext-bq-trainings.as_sql_basics.check_answer`('3.1', <result>)
```

Debug 3.D: find the bug

3 MIN

You

How many tickets did we sell in October?

AI

Here’s your query: 296 tickets sold in October.

```
SELECT
  COUNT(*) AS tickets_sold
FROM `trkkn-ext-bq-trainings.as_sql_basics.orders`
WHERE order_date BETWEEN '2026-10-01' AND '2026-10-31'
```

SANITY CHECK

Many companies buy team tickets: one order can contain up to 5 tickets. What did the AI actually count?

HINT

one row = one order

04

Apply business rules

Conditional logic with CASE WHEN

ANALYTICS SUMMIT

Bucketing with CASE WHEN

"Our table only has raw dates and numbers: how do we group rows into business categories?"

```sql
SELECT
  CASE
    WHEN condition_1 THEN 'label_1'
    WHEN condition_2 THEN 'label_2'
    ELSE 'fallback'
  END AS new_column
FROM table;
```

1. EVALUATES TOP-DOWN

Checks condition_1 first, then condition_2 in order.

2. FIRST MATCH WINS

As soon as a WHEN is true, SQL assigns that label and moves to the next row.

3. ELSE IS YOUR SAFETY NET

Without ELSE, any row that doesn't match a WHEN condition becomes NULL.

- `CASE WHEN` works top-down: the **first match wins**!
- Always add an `ELSE` as fallback: otherwise rows without a match get `NULL`.

Bucketing with CASE WHEN

The `CASE WHEN` statement allows you to create conditional logic in SQL, letting you categorize or "bucket" rows into different groups based on specified conditions, similar to if-else statements in other programming languages.

```
SELECT
  date,
  campaign,
  cost,
  CASE
    WHEN cost >= 200 THEN 'high'
    WHEN cost >= 50 THEN 'medium'
    ELSE 'low'
  END AS spend_level
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.campaigns`;
```

I'd like to label every campaign day as high, medium or low spend

Cheat sheet: Apply business rules

Every row falls from top to bottom and stops at the first match.

CASE WHEN BLUEPRINT

TASK 4.1

Turn values into labels

```
SELECT
  CASE
    WHEN cost >= 200
      THEN 'high'
    WHEN cost >= 50
      THEN 'medium'
    ELSE 'low'
  END AS spend_level,
  COUNT(*) AS campaign_days
FROM `trkkn-ext-bq-trainings.as_sql_basics.campaigns`
GROUP BY spend_level
```

1. **`WHEN ... THEN`** Tests conditions top-down
2. **`ELSE`** Fallback for unmatched rows
3. **`END AS`** Names the new bucket column

FIRST MATCH WINS

TASK 4.D

Test your order by hand

- Pick one row, e.g. a campaign day with 250 EUR.
- Walk through your **`WHEN`**s from the top, one after the other.
- Where does it stop? Is that the label you expect?
- If not, swap the order: the strictest condition goes first.

ELSE FALLBACK

Don’t forget ELSE

- Without **`ELSE`**, unmatched rows silently become **`NULL`**.
- **`END AS name`** names the column, **`GROUP BY name`** counts it.

CLAUSE ORDER

LIMIT

Clauses in order

`SELECT → FROM → WHERE → GROUP BY → ORDER BY → LIMIT`
`WHERE` filters before grouping, **`LIMIT`** cuts at the very end.



Hands-On

Hands-On: CASE WHEN

5 MIN

Use the `orders` table. The ticket shop had three sales phases:
'Early Bird' (until 31 July), 'Regular' (August - September), 'Last Minute' (October)

4.1 WRITE

Put every order into its `sales_phase` with `CASE WHEN`. How many orders were placed in the Early Bird phase? (Tip: combine with `COUNT(*)` & `GROUP BY sales_phase`)

4.D DEBUG

The AI's version of this query never returns a single 'Early Bird' order. Fix it: how many orders fall into **Regular**?

Copy & check every answer:

```
SELECT `trkkn-ext-bq-trainings.as_sql_basics.check_answer`('4.1', <result>)
```

| *TRKKN Proprietary & Confidential*

Debug 4.D: find the bug

3 MIN

You

How many orders were placed in each sales phase?

AI

Here’s your query. It returns 635 Regular orders and 296 Last Minute orders:

```
SELECT
  CASE
    WHEN order_date <= '2026-09-30' THEN 'Regular'
    WHEN order_date <= '2026-07-31' THEN 'Early Bird'
    ELSE 'Last Minute'
  END AS sales_phase,
  COUNT(*) AS orders
FROM `trkkn-ext-bq-trainings.as_sql_basics.orders`
GROUP BY sales_phase
```

SANITY CHECK

The AI’s query returns only 'Regular' and 'Last Minute', not a single 'Early Bird' order. Why?

HINT

CASE WHEN: which WHEN wins?

05

Connect data reliably

Combining tables: LEFT JOIN

ANALYTICS SUMMIT

What are JOINs?

How do we combine campaign costs from `campaigns` with channels from `campaign_list`?

```
SELECT
  table1.field1,
  table1.field2,
  table2.field2
FROM table1
LEFT JOIN table2
  ON table1.field1 = table2.field1;
```

1. TABLE PREFIX

table1.field specifies which table each column comes from.

2. LEFT JOIN

Keeps all rows of the left table (FROM) and attaches matching columns from the right table.

3. ON CLAUSE (JOIN KEY)

Names the shared column that links both tables (in our data: campaign).

A `JOIN` clause combines rows from two or more tables so you can analyze them in one place.

You match tables using a JOIN key: a column with the same values in both tables (here: c`ampaign)`.

| ***TRKKN*** *Proprietary & Confidential*

How LEFT JOIN works, row by row

Every row of the left table stays. Matching columns from the right table are attached. (example values)

campaigns: left table (FROM)

campaign	clicks
Search – Brand	120
LinkedIn – Last Call	90
Pilot – AI Search Ads	40

+

campaign_list: right table (LEFT JOIN)

campaign	channel
Search – Brand	Paid Search
LinkedIn – Last Call	Paid Social
LinkedIn – Last Call	Paid Social

```
ON campaigns.campaign = campaign_list.campaign
```

Result: 4 rows instead of 3, and 340 clicks instead of 250

campaign	clicks	channel
Search – Brand	120	Paid Search
LinkedIn – Last Call	90	Paid Social
LinkedIn – Last Call	90	Paid Social
Pilot – AI Search Ads	40	NULL

Match: the channel is attached.

Duplicate key: the row appears twice, so its clicks are counted twice.

No match: the row stays, channel is NULL.

LEFT JOIN on our data

`campaigns` knows cost and clicks per campaign, `campaign_list` knows the channel. `LEFT JOIN` keeps **every** campaign day: if a campaign is missing in the list, `channel` is `NULL`.

How many clicks per channel?

```sql
SELECT
  campaign_list.channel,
  SUM(campaigns.clicks) AS clicks
FROM `trkkn-ext-bq-trainings.as_sql_basics.campaigns` AS campaigns
LEFT JOIN `trkkn-ext-bq-trainings.as_sql_basics.campaign_list` AS campaign_list
  ON campaigns.campaign = campaign_list.campaign
GROUP BY campaign_list.channel
```

Result

channel	clicks
Paid Search	17,819
Email	14,264
Paid Social	12,993
Partner	4,627
NULL	1,188

| ***TRKKN*** *Proprietary & Confidential*

What you need to know about JOINs

```sql
-- 1. Prefix columns with table name
SELECT
  campaigns.campaign,
  campaign_list.channel,
  SUM(campaigns.cost) AS total_cost
-- 2. Left table keeps all rows
FROM `trkkn-ext-bq-trainings.as_sql_basics.campaigns` AS campaigns
LEFT JOIN `trkkn-ext-bq-trainings.as_sql_basics.campaign_list` AS campaign_list
  -- 3. Link on shared join key
  ON campaigns.campaign = campaign_list.campaign
GROUP BY
  campaigns.campaign,
  campaign_list.channel;
```

1. PREFIX WITH TABLE NAME

Prefix columns with the table name (`campaigns.cost`, `campaign_list.channel`) so it is always clear which table a column comes from.

2. LEFT JOIN BY DEFAULT

`LEFT JOIN` keeps all rows of the left table. If there is no match, right-table columns are NULL.

3. SHARED KEY IN ON

The join key is the column both tables share (here: campaign), specified in the ON clause.

4. WATCH FOR DUPLICATE KEYS

Count rows before and after joining: if a key appears twice in the right table, sums are duplicated!

Cheat sheet: Connect data reliably

A join is only as good as the checks around it.

LEFT JOIN BLUEPRINT

TASK 5.1

Attach columns from a second table

```
SELECT
  campaign_list.channel,
  SUM(campaigns.cost) AS cost
FROM `trkkn-ext-bq-trainings.as_sql_basics.campaigns` AS campaigns
LEFT JOIN `trkkn-ext-bq-trainings.as_sql_basics.campaign_list` AS campaign_list
  ON campaigns.campaign = campaign_list.campaign
GROUP BY campaign_list.channel
ORDER BY cost DESC
```

TABLE PREFIXES

ON CLAUSE

Write it cleanly

- The table in **`FROM`** stays complete.
- Prefix every column: **`campaigns.cost`**
- **`ON`** names the column both tables share.

SANITY CHECK

DUPLICATE KEYS

TASK 5.D

Before: is the key unique?

- Count the rows per key in the right table.
- A key that appears twice repeats every matching row.
- Your sums come out too high, without any error.

UNMATCHED ROWS

TASK 5.2

After: did anything change?

- Compare the total before and after the join.
- **`NULL`** on the right? That key has no match there.
- Find those rows with **`WHERE column IS NULL`**.



Hands-On

Hands-On: LEFT JOIN

8 MIN

Join the tables `campaigns` and `campaign_list` on `campaign`:

5.1 WRITE

How much did the channel Paid Search cost in total? Enter EUR with 2 decimals.

5.2 WRITE

How much cost has **no** channel (channel IS NULL)? Enter EUR with 2 decimals. Which campaign is it, and why could that happen?

5.D DEBUG

An AI built a cost-per-channel report. All channels add up to more than the total ad spend. Find the duplicated campaign and enter its total cost in EUR with 2 decimals.

Copy & check every answer:

```
SELECT `trkkn-ext-bq-trainings.as_sql_basics.check_answer`('5.1', <result>)
```

Debug 5.D: find the problem

4 MIN

You

How much ad spend did we have per marketing channel?

AI

Here's your cost-per-channel report: the channels add up to 75,277.24 EUR:

```
SELECT
  campaign_list.channel,
  SUM(campaigns.cost) AS cost
FROM `trkkn-ext-bq-trainings.as_sql_basics.campaigns` AS campaigns
LEFT JOIN `trkkn-ext-bq-trainings.as_sql_basics.campaign_list` AS campaign_list
  ON campaigns.campaign = campaign_list.campaign
GROUP BY campaign_list.channel
ORDER BY cost DESC
```

SANITY CHECK

Total ad spend is only 66,833.76 EUR (task 2.1). Where does the extra spend come from?

HINT

count the rows per campaign in `campaign_list`

06

Break down complex questions

Thinking in steps: CTEs

ANALYTICS SUMMIT

Why use CTEs?

On how many days did we get more than 10 orders?

```sql
WITH step_one AS (
  -- Step 1: intermediate result
  SELECT ...
  FROM ...
  GROUP BY ...
),
step_two AS (
  -- Step 2: build on step_one
  SELECT ...
  FROM step_one
)
-- Final query
SELECT *
FROM step_two;
```

1. NAMED BUILDING BLOCKS

`WITH` step_one AS (...) creates a temporary intermediate table you can query in the next step.

2. EASY TO SANITY-CHECK

Each CTE is a small, standalone query that you can highlight, run, and verify on its own.

3. BEST FOR AI WORKFLOWS

Thinking in steps is the most reliable way to work with AI: one step = one instruction = one check.

One query step is not enough when you need to filter or aggregate on an already aggregated result!

A **CTE** (Common Table Expression) breaks a complex question into named steps you can check one by one.

A CTE on our data

How many orders do we get on an average day?

```sql
WITH orders_per_day AS (
  -- step 1: one row per day
  SELECT
    order_date,
    COUNT(*) AS orders
  FROM `trkkn-ext-bq-trainings.as_sql_basics.orders`
  GROUP BY order_date
)
-- step 2: one row for all days
SELECT
  AVG(orders) AS avg_orders_per_day
FROM orders_per_day
```

1. STEP 1: DEFINE THE CTE

`WITH` orders_per_day AS (...) defines an intermediate result, here one row per day (2026-06-01: 3, 2026-06-02: 9, ...). Run the inner query on its own first!

2. STEP 2: QUERY THE CTE

The final query reads FROM orders_per_day just like from a normal table to compute the daily average: 6.38 orders.

3. CHAINING MORE STEPS

Need more steps? Separate multiple CTEs with a comma: `WITH` step_a AS (...), step_b AS (...).

Readable SQL: code structure

BEST PRACTICE: CODE STYLE

```
SELECT
order_date,channel,ticket_type, revenue
FROM
`trkkn-ext-bq-trainings.as_sql_basics.orders`
```

```
SELECT
  order_date,
  channel,
  ticket_type,
  revenue
FROM
  `trkkn-ext-bq-trainings.as_sql_basics.orders`
```

- Each column should appear on its own line in **SELECT** statements for better readability.
- Indent column names and conditions by 2 spaces.
- Align keywords like **SELECT**, **FROM**, **WHERE**, and **JOIN** to the left margin without indentation.
- Keep the **FROM** clause on the same indentation level as **SELECT**

Cheat sheet: Break down complex questions

Stuck on one big query? Write the steps in plain words first.

WHEN TO USE A CTE | GRAIN PER STEP

TASKS 6.1 + 6.2

When do I need a CTE?

- When you want to filter or count on something you already aggregated, e.g. “how many countries brought more than …?”
- When you combine two tables that must first reach the same level of detail.

CTE BLUEPRINT (WITH)

Two steps, two checks

```
WITH orders_per_day AS (
  -- Step 1: one row per day
  SELECT
    order_date,
    COUNT(*) AS orders
  FROM `trkkn-ext-bq-trainings.as_sql_basics.orders`
  GROUP BY order_date
)
-- Step 2: average across days
SELECT
  AVG(orders) AS avg_orders
FROM orders_per_day
```

ONE STEP, ONE CHECK

Check every step

- Highlight the inner **`SELECT`** and run it alone.
- Is one row what you expected? Only then go on.

READABLE SQL

Keep it readable

- One column per line, 2 spaces indent.
- One comment per step: what is one row here?



Hands-On

Hands-On: CTEs

9 MIN

Build your query in steps:

6.1 WRITE

`campaigns`: How many campaigns have a cost per click above 2 EUR? Step 1: cost and clicks per campaign (CTE). Step 2: count the campaigns where cost / clicks > 2.

6.2 WRITE

`orders`: How many countries brought more than 20,000 EUR revenue? Step 1: revenue per country. Step 2: count.

BONUS

Copy & check every answer:

```
SELECT `trkkn-ext-bq-trainings.as_sql_basics.check_answer`('6.1', <result>)
```

| *TRKKN Proprietary & Confidential*

07

Use AI without losing trust

Prompt and verify

ANALYTICS SUMMIT

What a bad prompt looks like

You

write me an sql query for revenue per channel

AI

Sure! Here is a query for revenue per channel:

```
SELECT
  channel,
  SUM(amount) AS revenue
FROM sales
GROUP BY channel;
```

QUESTION

What will the AI have to guess?

Prompting: the 5 building blocks

Objective: turn “revenue per channel” into a prompt the AI cannot misread.

You

I’m working in BigQuery (Standard SQL).

The data is in `` `trkkn-ext-bq-trainings.as_sql_basics.orders` ``.
One row is one ticket order.
The columns I need: `channel` (STRING) and `revenue` (FLOAT, in EUR).

By “revenue per order” I mean total revenue divided by the number of orders.

My question: which channel brings the most revenue per order?
One row per channel, highest first, plus the number of orders.

Please use CTEs with a short comment per step, and no SELECT *.

AI

...

Why it matters

1. DIALECT*

Right syntax (BigQuery Standard SQL)

2. TABLE & ROWS

No invented tables or columns

3. DEFINITIONS

No guessing what “revenue” means

4. QUESTION & OUTPUT

The exact grain & sorting you need

5. STYLE

A clean, readable query you can verify

* Dialect: the SQL variant of a database. The syntax differs slightly between e.g. BigQuery and MySQL.

Mind the data

Share the schema* and the question, never the data. The AI can’t un-see what you paste.

Share freely with AI

- Table and column names (orders, campaigns)
- Data types from the Schema tab (STRING, DATE, FLOAT)
- Your business question and metric definitions
- What one row represents (table grain)

Never paste into AI

- Customer or personal data (names, emails, IDs)
- Raw query results and CSV exports
- Passwords, API keys, or service credentials
- Confidential business figures and revenue totals

Use your company’s approved AI tools. You’re accountable for every number.

* Schema: the list of a table’s column names and data types.

Remember this?

You

How many tickets did we sell in October?

AI

Here's your query: you sold 296 tickets.

```
SELECT
  COUNT(*) AS tickets_sold
FROM `trkkn-ext-bq-trainings.as_sql_basics.orders`
WHERE order_date BETWEEN '2026-10-01' AND '2026-10-31';
-- runs fine, returns 296
```

SANITY CHECK

Which check would have caught `COUNT(*)` vs. `SUM(tickets)` without knowing the answer?

What to check before you trust an AI query

4 checks that catch typical AI mistakes, even when the SQL runs without an error:

01 SCHEMA & FILTERS

Did the AI invent columns or misread logic?

- Check that every table and column exists in your schema.
- Check `WHERE` date ranges, `CASE WHEN` order (strictest first!), and brackets around `OR`.

02 GRAIN & METRIC

What does one row represent?

- Never confuse counting rows with summing values: `COUNT(*)` counts orders (296), whereas `SUM(tickets)` sums tickets sold (385).

03 JOIN MULTIPLICATION

Did the row count grow after joining?

- Count rows before and after every `JOIN`.
- Aggregate both tables to the exact same grain in separate CTEs before joining them.

04 BENCHMARK REALITY CHECK

Does the number match a known total?

- Compare against a number you already know.
- Seeing 296 for October tickets immediately rings a bell if you know October had 296 orders!

ANSWER TO "REMEMBER THIS?": Check 02 (Grain: `COUNT(*)` vs. `SUM(tickets)`) and Check 04 (296 = October orders) catch the bug immediately.

| ***TRKKN*** *Proprietary & Confidential*

Make the AI check itself

When a check fails, or you can't read the query, ask:

01 UNDERSTAND

“Explain this query line by line.”
When the AI uses unfamiliar syntax or functions.

02 UNCOVER ASSUMPTIONS

“What assumptions did you make?”
Reveals guessed filters, join keys, or column meanings.

03 VERIFY THE RESULT

“Write a query that checks this result.”
Lets the AI write a row-count or total-sum sanity check.

04 MAKE IT READABLE

“Rewrite it with CTEs and a comment per step.”
Breaks a nested monster query into checkable steps.

| ***TRKKN*** *Proprietary & Confidential*

Cheat sheet: Use AI without losing trust

Prompt, run, check. Only then trust the number.

THE 5 PROMPT BLOCKS

TASK 7.1

1. Give the AI all the context

```
1. Dialect: BigQuery Standard SQL
2. Table: `trkkn-ext-bq-trainings.as_sql_basics.orders`
   Grain: 1 row = 1 order
   Cols: channel, tickets, revenue
3. Rule: AOV* = SUM(revenue)/COUNT(*)
4. Output: 1 row per channel, sorted
5. Style: Use readable CTEs
```

1. Dialect & Table Exact **`BigQuery`** path & types
2. Grain & Rules What 1 row is + business logic
3. Output & Style Expected columns + **`CTEs`**

SANITY CHECKS

BONUS 7.D

2. Check before you trust

- Do all tables and columns exist? AI loves hallucinated column names.
- Read the filters: **`CASE WHEN`** order, brackets around **`OR`**.
- What is one row? **`COUNT(*)`** or **`SUM(column)`**?
- Same row count after the **`JOIN`**? Does the total match a known number?

TYPICAL AI TRAPS

SELF-CHECK PROMPTS

3. Check fails? Ask

- “Explain this query line by line.”
- “What assumptions did you make?”
- “Write a query that checks this result.”

MIND THE DATA

Mind the data

Share schema and question. Never paste customer data or results.

* AOV: average order value.



Hands-On

Hands-On: Prompt, check, fix

10 MIN

7.1 WRITE

Prompt & check (alone or in a small team): Use your AI tool and the 5 building blocks: Which campaign has the best ROAS (revenue from `orders` / cost from `campaigns`)? Enter its ROAS with 2 decimals. Then run the validation checks.

7.D DEBUG

An AI wrote a “Top 3 ticket types by revenue in October” query. It runs, but has **three** bugs. Find all three, then enter the revenue of the #1 ticket type in EUR with 2 decimals.

BONUS

Copy & check every answer:

```
SELECT `trkkn-ext-bq-trainings.as_sql_basics.check_answer`('7.1', <result>)
```

Debug 7.D: three bugs

BONUS

You

What were the top 3 ticket types by revenue in October 2026?

AI

Here’s your query for the top 3 ticket types in October 2026:

```
SELECT
  orders.ticket_type,
  SUM(orders.tickets) AS revenue
FROM `trkkn-ext-bq-trainings.as_sql_basics.orders` AS orders
LEFT JOIN `trkkn-ext-bq-trainings.as_sql_basics.campaigns` AS campaigns
  ON orders.campaign = campaigns.campaign
WHERE orders.order_date BETWEEN '2026-10-01' AND '2026-10-31'
GROUP BY orders.ticket_type
ORDER BY revenue ASC
LIMIT 3
```

SANITY CHECK

The query runs without an error, but the ranking makes no sense. Can you spot all 3 bugs?

HINT

join, revenue column, sort order

08

Practical case

Online marketing budget allocation

ANALYTICS SUMMIT

Hands-On

ML

From: Marketing Lead

To: You

Subject: **Which channel deserves more budget next year?**

Hi,

next year's budget is on my desk, and I have to defend it in front of the CFO. I need a recommendation I can stand behind: which channel deserves more money?

A few things to keep in mind:

Budget. The total stays the same, but you're free to move money between channels. Direct, Organic & Co. have no ad spend, so they're not part of this decision.

Break-even. A channel has to earn back at least 3 EUR for every 1 EUR we spend (ROAS 3, with ROAS = revenue ÷ ad spend). Below that, it doesn't pay for itself.

Email. The newsletter goes to our existing list. Spending more won't make that list any bigger.

Paid Social. Our agency swears it's how new people find us.

Thanks, I'm counting on you!

Marketing Lead

| ***TRKKN*** *Proprietary & Confidential*

Hands-On

ML

From: Marketing Lead

To: You

Subject: **Which channel deserves more budget next year?**

Hi,

next year's budget is on my desk, and I have to defend it in front of the CFO. I need a recommendation I can stand behind: which channel deserves more money?

A few things to keep in mind:

Budget. The total stays the same, but you're free to move money between channels. Direct, Organic & Co. have no ad spend, so they're not part of this decision.

Break-even. A channel has to earn back at least 3 EUR for every 1 EUR we spend (ROAS 3, with ROAS = revenue ÷ ad spend). Below that, it doesn't pay for itself.

Email. The newsletter goes to our existing list. Spending more won't make that list any bigger.

Paid Social. Our agency swears it's how new people find us.

Thanks, I'm counting on you!

Marketing Lead

RECOMMENDATION
Which channel deserves more budget?

CHANNELS WITH AD SPEND
Exclude Direct & Organic (no spend)

KEY NUMBERS
ROAS per channel vs. break-even 3.0

LIMITATIONS & CONTEXT
Email list ceiling vs. Social reach

| ***TRKKN*** *Proprietary & Confidential*

Hands-On: Practical case

17 MIN

C.1 WRITE

Paid channels only: one row per channel with ad spend, showing spend, revenue, orders and ROAS. Enter Paid Search's ROAS with 2 decimals.

C.2 WRITE

Same table. Enter Paid Social's ROAS with 2 decimals.

C.3 WRITE

Formulate your recommendation in 2 to 3 sentences: which channel should get more budget, and why?

Copy & check every answer:

```
SELECT `trkkn-ext-bq-trainings.as_sql_basics.check_answer`('C.1', <ROAS Paid Search>)
```

```
SELECT `trkkn-ext-bq-trainings.as_sql_basics.check_answer`('C.2', <ROAS Paid Social>)
```

Alone or in small teams, 17 min, AI allowed, same budget, break-even ROAS 3, remember to deduplicate `campaign_list` (5.D)

At minute 12 we show the finished table, so everyone can work on the recommendation.

| *TRKKN Proprietary & Confidential*

How to find and communicate your answer

1. ANALYSIS STRATEGY: HOW TO FIND YOUR ANSWER

AD SPEND

Channels with ad spend
Exclude Direct & Organic (no spend in `campaigns`).

KEY NUMBERS

Check ROAS vs. the break-even of 3.0
Combine spend and revenue per channel (deduplicate `campaign_list`!).

TIME & LEVEL

Look beneath the channel average
Is one campaign or month driving the result? Check `GROUP BY campaign`.

LIMITATIONS

Factor in what is not in the numbers
Email list cannot grow via ad spend; last-click* in orders.channel.

2. ANSWER STRATEGY: HOW TO EXPLAIN & BACK IT UP

RECOMMENDATION

Start with a clear recommendation
Sentence 1: Which channel should get more budget (and where from)?

KEY NUMBERS

Back it up with two key metrics
Sentence 2: Cite the spend, revenue, and ROAS vs. the 3.0 break-even.

LIMITATIONS

Address the main caveat proactively
Sentence 3: Name the limitation (e.g. Email ceiling or campaign mix).

NEXT CHECK

Suggest one concrete follow-up step
Optional: What would you test or check before shifting the full budget?

* Last-click: the whole order counts for the last channel someone used before buying.

Hands-On: Share your recommendation

7 MIN

Marketing Lead

ML

Which channel deserves more budget next year?
I need a clear recommendation I can defend in front of the CFO.

You (2 to 3 sentences)

Your recommendation in 2 to 3 sentences:

- **Recommendation:** Which channel should get more budget (and where from)?
- **Key numbers:** Which spend, revenue, and ROAS figures back it up?
- **Limitation & next check:** Which caveat matters most before shifting the budget?

| ***TRKKN*** *Proprietary & Confidential*

Key takeaways: 7 questions for every data task

01 UNDERSTAND YOUR DATA

What is one row?

Look at Schema, Details and Preview before you write anything. Every count and sum depends on the answer.

02 GET THE NUMBERS

Which keyword does my question ask for?

“In total”, “how many different”, “highest”: the wording tells you. Read your `WHERE` out loud.

03 TURN ROWS INTO INSIGHTS

Per what? Count or sum?

“Per” means `GROUP BY`. Orders are rows, tickets are values: 296 orders are not 385 tickets.

04 APPLY BUSINESS RULES

Where does each row stop?

`CASE WHEN` takes the first match. Test one row by hand, strictest condition first.

05 CONNECT DATA RELIABLY

Is the key unique? Same total after the join?

Before: count the rows per key. After: compare the totals and look for `NULL`. Duplicates multiply your sums without any error.

06 BREAK DOWN QUESTIONS

Can I split this into steps?

One CTE per step, one comment per step. Run each step on its own and check what one row is.

07 USE AI WITHOUT LOSING TRUST

Did I give context and check the result?

Prompt with the 5 blocks. Check columns, filters, grain and totals. Share the schema, never the data.

| ***TRKKN*** *Proprietary & Confidential*

Thanks!

TRKKN

Trakken GmbH

www.trkkn.com











ANALYTICS SUMMIT