# Troubleshooting and Solving Data Join Pitfalls v1.5 | Google Cloud Skills Boost

Qwiklabs : 21-26 minutes

---

## Overview

BigQuery is Google's fully managed, NoOps, low cost analytics database. With BigQuery you can query terabytes and terabytes of data without having any infrastructure to manage or needing a database administrator. BigQuery uses SQL and can take advantage of the pay-as-you-go model. BigQuery allows you to focus on analyzing data to find meaningful insights.

Joining data tables can provide meaningful insight into your dataset. However, when you join your data, there are common pitfalls that could corrupt your results. This lab focuses on avoiding those pitfalls. Types of joins:

- *Cross join*: combines each row of the first dataset with each row of the second dataset, where every combination is represented in the output.
- *Inner join*: requires that key values exist in both tables for the records to appear in the results table. Records appear in the merge only if there are matches in both tables for the key values.
- *Left join*: Each row in the left table appears in the results, regardless of whether there are matches in the right table.
- *Right join*: the reverse of a left join. Each row in the right table appears in the results, regardless of whether there are matches in the left table.

For more information about joins, see Join Page.

The dataset you'll use is an ecommerce dataset that has millions of Google Analytics records for the Google Merchandise Store loaded into BigQuery. You have a copy of that dataset for this lab and will explore the available fields and row for insights.

For syntax information to help you follow and update the queries, see Standard SQL Query Syntax.

### What you'll do

In this lab, you perform these tasks:

- Use BigQuery to explore a dataset
- Troubleshoot duplicate rows in a dataset
- Create joins between data tables
- Understand each join type

## Setup and requirements

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

1. Sign in to Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, `1:15:00`), and make sure you can finish within that time.
   There is no pause feature. You can restart if needed, but you have to start at the beginning.
3. When ready, click **Start lab**.
4. Note your lab credentials (**Username** and **Password**). You will use them to sign in to the Google Cloud Console.
5. Click **Open Google Console**.
6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.
   If you use other credentials, you'll receive errors or **incur charges**.
7. Accept the terms and skip the recovery resource page.

### Open BigQuery Console

1. In the Google Cloud Console, select **Navigation menu** > **BigQuery**.

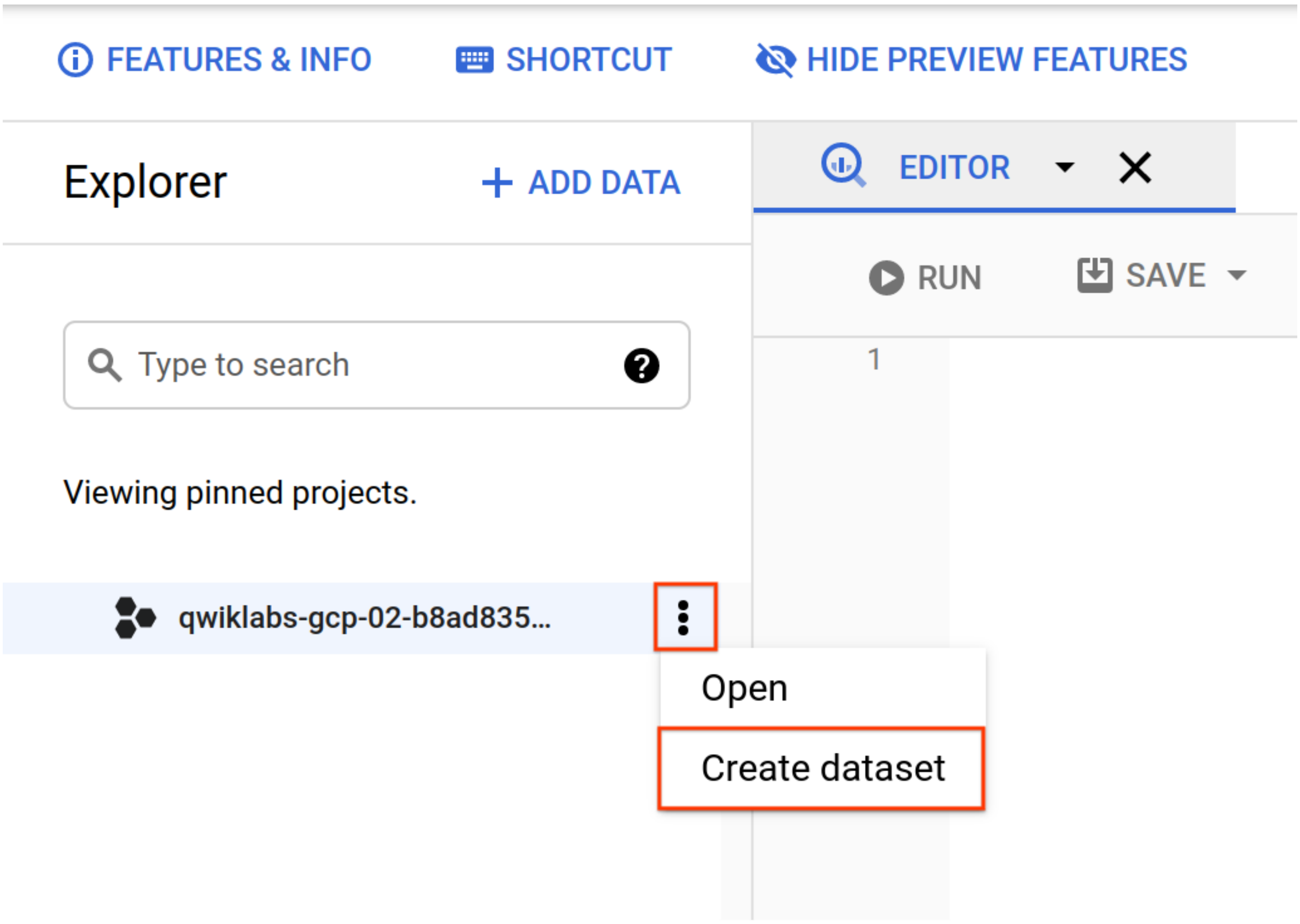
The **Welcome to BigQuery in the Cloud Console** message box opens. This message box provides a link to the quickstart guide and lists UI updates.

2. Click **Done**.

## Task 1. Create a new dataset to store your tables

In your BigQuery project, create a new dataset titled `ecommerce`.

1. In the left pane in the **Explorer** section, click on the **View actions** icon next to your *Project ID* and select **Create dataset**.



2. Set the *Dataset ID* to `ecommerce`. Leave the other options at their default values, and click **Create dataset**.

In the left pane, you see an `ecommerce` dataset listed under your project.

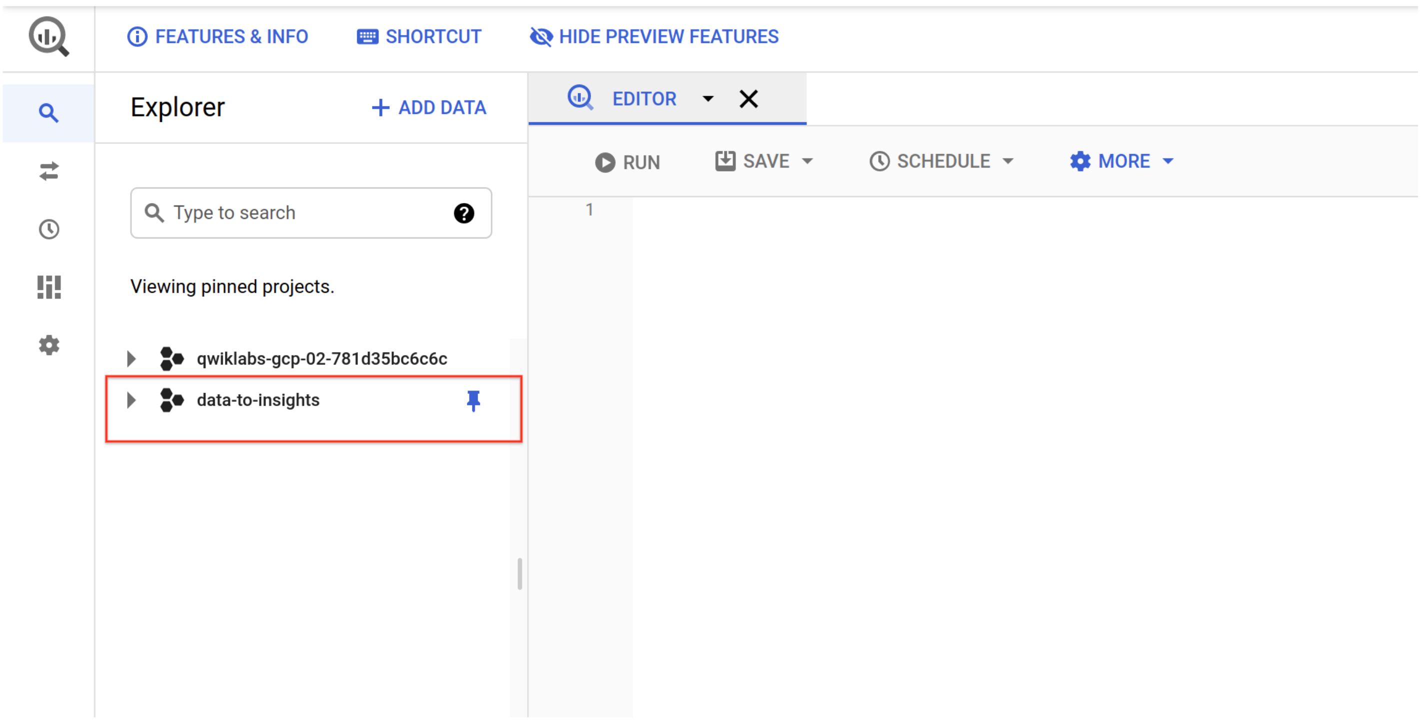
## Task 2. Pin the Lab Project in BigQuery

Scenario: Your team provides you with a new dataset on the inventory stock levels for each of your products for sale on your ecommerce website. You want to become familiar with the products on the website and the fields you could use to potentially join on to other datasets.

The project with the new dataset is **data-to-insights**.

BigQuery public datasets are not displayed by default in the BigQuery web UI. Since **data-to-insights** is a public dataset project, you have to pin it to your Resource Tree:

1. In a new browser window, open the public datasets project.



Viewing pinned projects.

qwiklabs-gcp-02-781d35bc6c6c

data-to-insights

2. Close this browser window.

3. Return to and refresh the first BigQuery browser window to refresh the BigQuery web UI.

The `data-to-insights` project is listed in the Resource section.

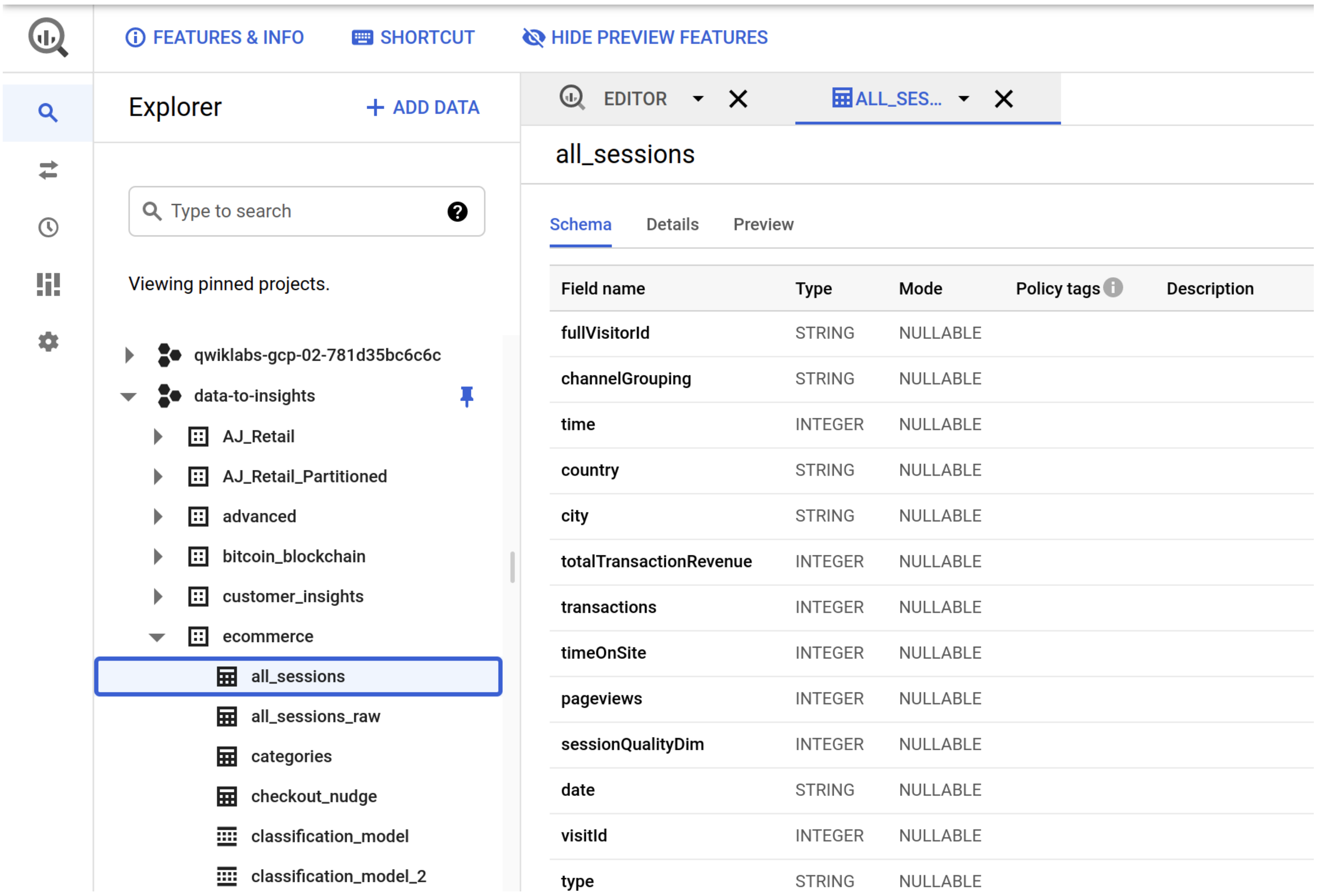
**Note:** If the public `data-to-insights` project doesn't appear in the Explorer panel, then click on **+ ADD DATA > Star a project by name > Enter project name (data-to-insights)** and click on **Star**.

## Task 3. Examine the fields

You want to become familiar with the products on the website and the fields you could use to create queries to analyze the dataset.

1. In the left pane in the **Explorer** section, navigate to `data-to-insights > ecommerce > all_sessions_raw`.

2. On the right, click the **Schema** tab to see the Fields and information about each field.



Viewing pinned projects.

- qwiklabs-gcp-02-781d35bc6c6c
- data-to-insights
  - AJ_Retail
  - AJ_Retail_Partitioned
  - advanced
  - bitcoin_blockchain
  - customer_insights
  - ecommerce
    - all_sessions
    - all_sessions_raw
    - categories
    - checkout_nudge
    - classification_model
    - classification_model_2

all_sessions

Schema | Details | Preview

| Field name | Type | Mode | Policy tags | Description |
|---|---|---|---|---|
| fullVisitorId | STRING | NULLABLE | | |
| channelGrouping | STRING | NULLABLE | | |
| time | INTEGER | NULLABLE | | |
| country | STRING | NULLABLE | | |
| city | STRING | NULLABLE | | |
| totalTransactionRevenue | INTEGER | NULLABLE | | |
| transactions | INTEGER | NULLABLE | | |
| timeOnSite | INTEGER | NULLABLE | | |
| pageviews | INTEGER | NULLABLE | | |
| sessionQualityDim | INTEGER | NULLABLE | | |
| date | STRING | NULLABLE | | |
| visitId | INTEGER | NULLABLE | | |
| type | STRING | NULLABLE | | |

## Task 4. Identify a key field in your ecommerce dataset

Examine the products and fields further. You want to become familiar with the products on the website and the fields you could use to potentially join on to other datasets.

### Examine the records

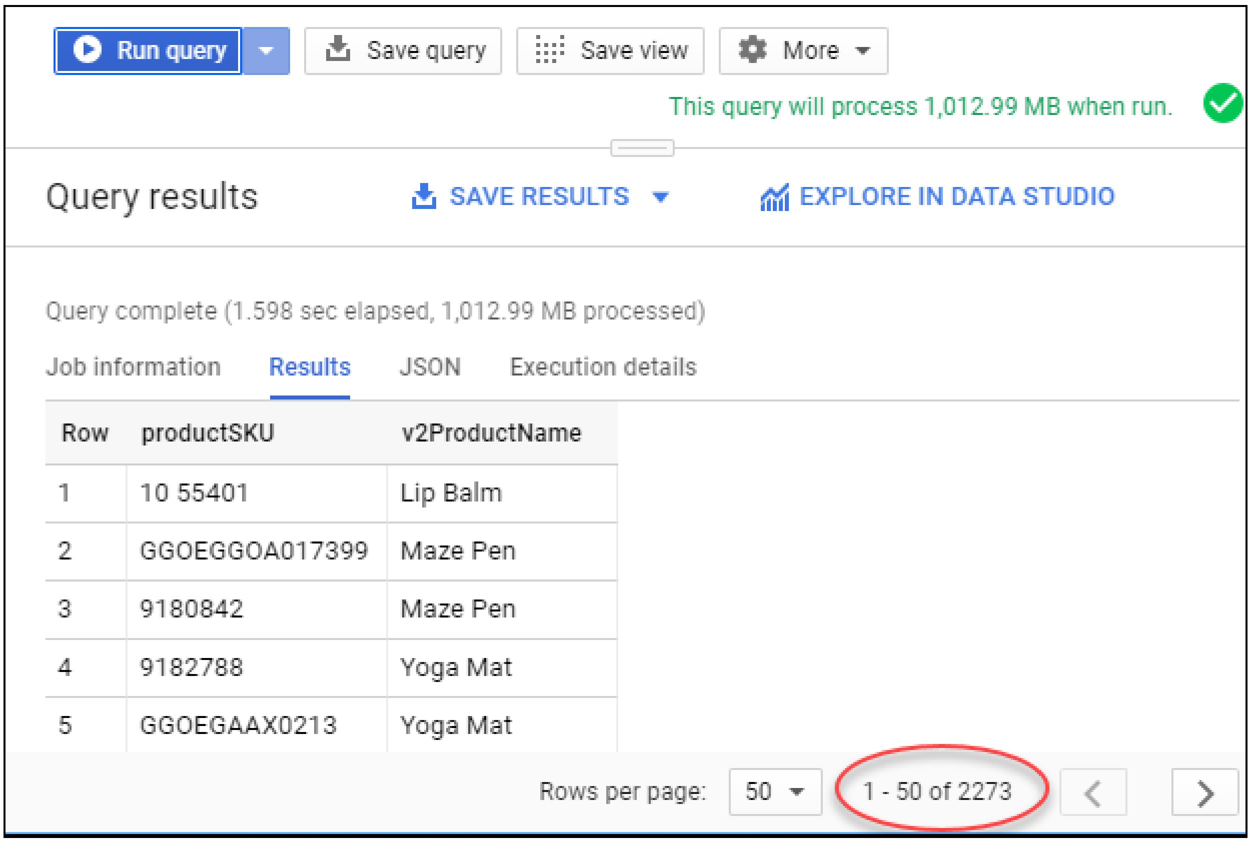
In this section, you find how many product names and product SKUs are on your website and whether either one of those fields is unique.

1. Find how many product names and product SKUs are on the website. **Copy and paste** the below query in the query **EDITOR**:

#standardSQL # how many products are on the website? SELECT DISTINCT productSKU, v2ProductName FROM `data-to-insights.ecommerce.all_sessions_raw`

2. Click **RUN**.

3. Look at the pagination results in the web UI for the total number of records returned, which in this case is 2,273 products and SKUs.



This query will process 1,012.99 MB when run.

Query results

Query complete (1.598 sec elapsed, 1,012.99 MB processed)

Job information | Results | JSON | Execution details

| Row | productSKU | v2ProductName |
|---|---|---|
| 1 | 10 55401 | Lip Balm |
| 2 | GGOEGGOA017399 | Maze Pen |
| 3 | 9180842 | Maze Pen |
| 4 | 9182788 | Yoga Mat |
| 5 | GGOEGAAX0213 | Yoga Mat |

Rows per page: 50 | 1 - 50 of 2273

But...do the results mean that there are 2,273 unique product SKUs?

4. **Copy and paste** the below query to list the number of distinct SKUs are listed using `DISTINCT`:

#standardSQL # find the count of unique SKUs SELECT DISTINCT productSKU FROM `data-to-insights.ecommerce.all_sessions_raw`

5. Click **RUN**.

The number of distinct SKUs are 1,909.

Hmmm...you have 1,909 distinct SKUs which is less than the 2,273 number for total number of products on the website. The first results probably contain products with duplicate SKUs.

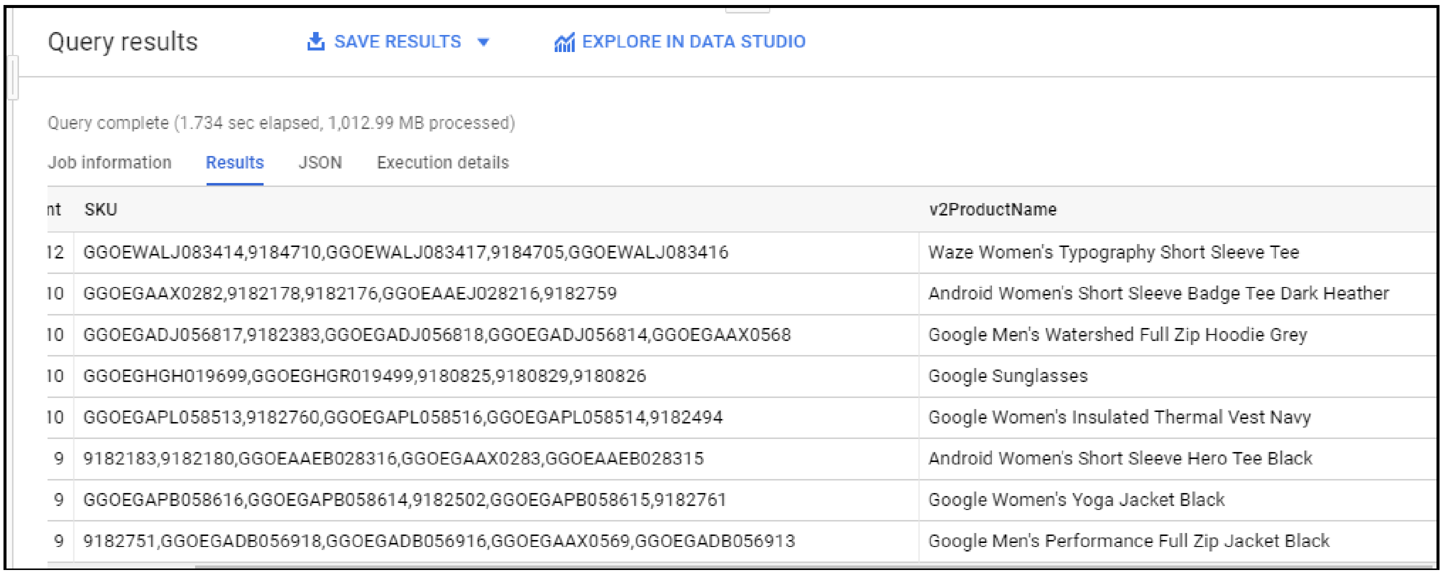
6. Take an even closer look at the records. Determine which products have more than one SKU and which SKUs have more than one product.

7. **Copy and paste** the below query to determine if some product names have more than one SKU. Notice ou use the STRING_AGG() function to aggregate all the product SKUs that are associated with one product name.

#standardSQL # how can we find the products with more than 1 sku? SELECT DISTINCT COUNT(DISTINCT productSKU) AS SKU_count, STRING_AGG(DISTINCT productSKU LIMIT 5) AS SKU, v2ProductName FROM `data-to-insights.ecommerce.all_sessions_raw` WHERE productSKU IS NOT NULL GROUP BY v2ProductName HAVING SKU_count > 1 ORDER BY SKU_count DESC # product name is not unique (expected for variants)

8. Click **RUN**.

Results:



Query results SAVE RESULTS EXPLORE IN DATA STUDIO

Query complete (1.734 sec elapsed, 1,012.99 MB processed)

Job information Results JSON Execution details

| nt | SKU | v2ProductName |
|---|---|---|
| 12 | GGOEWALJ083414,9184710,GGOEWALJ083417,9184705,GGOEWALJ083416 | Waze Women's Typography Short Sleeve Tee |
| 10 | GGOEGAAX0282,9182178,9182176,GGOEAAEJ028216,9182759 | Android Women's Short Sleeve Badge Tee Dark Heather |
| 10 | GGOEGADJ056817,9182383,GGOEGADJ056818,GGOEGADJ056814,GGOEGAAX0568 | Google Men's Watershed Full Zip Hoodie Grey |
| 10 | GGOEGHGH019699,GGOEGHGR019499,9180825,9180829,9180826 | Google Sunglasses |
| 10 | GGOEGAPL058513,9182760,GGOEGAPL058516,GGOEGAPL058514,9182494 | Google Women's Insulated Thermal Vest Navy |
| 9 | 9182183,9182180,GGOEAAEB028316,GGOEGAAX0283,GGOEAAEB028315 | Android Women's Short Sleeve Hero Tee Black |
| 9 | GGOEGAPB058616,GGOEGAPB058614,9182502,GGOEGAPB058615,9182761 | Google Women's Yoga Jacket Black |
| 9 | 9182751,GGOEGADB056918,GGOEGADB056916,GGOEGAAX0569,GGOEGADB056913 | Google Men's Performance Full Zip Jacket Black |

Do some product names have more than one SKU? Look at the query results to confirm.

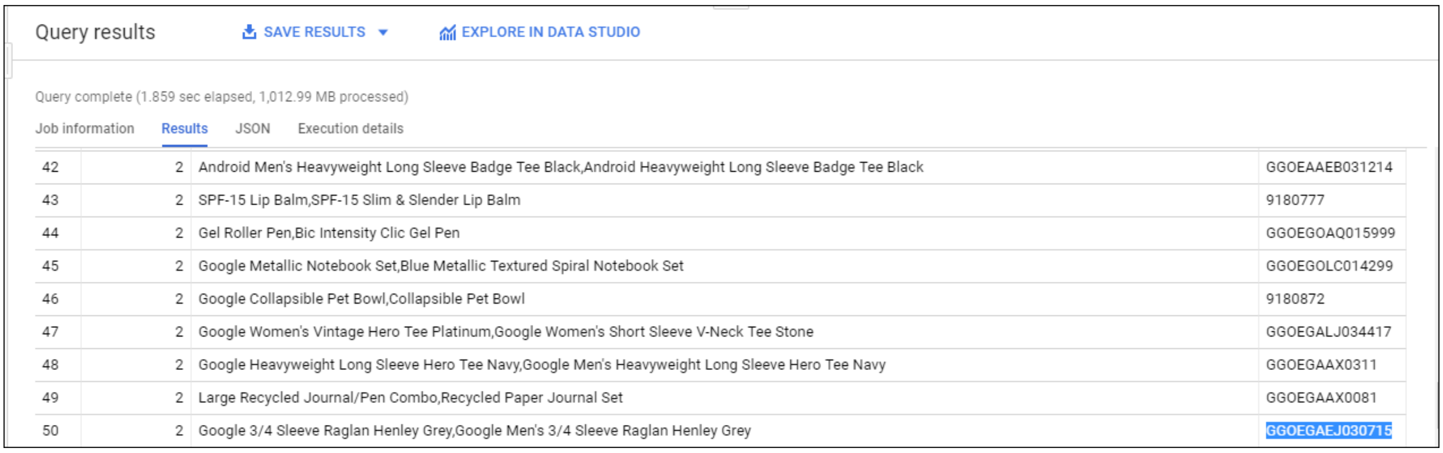
Answer: Yes

It may also be true that one product name be associated with more than one SKU. This can happen due to variation. For example, one product name (e.g. T-Shirt) can have multiple product variants like color, size, etc. You would expect one product to have many SKUs.

9. **Copy and paste** the below query to find out:

#standardSQL SELECT DISTINCT COUNT(DISTINCT v2ProductName) AS product_count, STRING_AGG(DISTINCT v2ProductName LIMIT 5) AS product_name, productSKU FROM `data-to-insights.ecommerce.all_sessions_raw` WHERE v2ProductName IS NOT NULL GROUP BY productSKU HAVING product_count > 1 ORDER BY product_count DESC # SKU is not unique (indicates data quality issues)

10. Click **RUN**.



Query results SAVE RESULTS EXPLORE IN DATA STUDIO

Query complete (1.859 sec elapsed, 1,012.99 MB processed)

Job information Results JSON Execution details

| | | | |
|---|---|---|---|
| 42 | 2 | Android Men's Heavyweight Long Sleeve Badge Tee Black,Android Heavyweight Long Sleeve Badge Tee Black | GGOEAAEB031214 |
| 43 | 2 | SPF-15 Lip Balm,SPF-15 Slim & Slender Lip Balm | 9180777 |
| 44 | 2 | Gel Roller Pen,Bic Intensity Clic Gel Pen | GGOEGOAQ015999 |
| 45 | 2 | Google Metallic Notebook Set,Blue Metallic Textured Spiral Notebook Set | GGOEGOLC014299 |
| 46 | 2 | Google Collapsible Pet Bowl,Collapsible Pet Bowl | 9180872 |
| 47 | 2 | Google Women's Vintage Hero Tee Platinum,Google Women's Short Sleeve V-Neck Tee Stone | GGOEGALJ034417 |
| 48 | 2 | Google Heavyweight Long Sleeve Hero Tee Navy,Google Men's Heavyweight Long Sleeve Hero Tee Navy | GGOEGAAX0311 |
| 49 | 2 | Large Recycled Journal/Pen Combo,Recycled Paper Journal Set | GGOEGAAX0081 |
| 50 | 2 | Google 3/4 Sleeve Raglan Henley Grey,Google Men's 3/4 Sleeve Raglan Henley Grey | GGOEGAEJ030715 |

**Question:** When you look at the query results, are there single SKU values with more than one product name associated? What do you notice about those product names?

**Answer:** Yes, it looks like there are quite a few SKUs that have more than one product name. Several of the product names appear to be closely related with a few misspellings (e.g. Waterproof Gear Bag vs Waterproof Gear Bag).

You see why this could be an issue in the next section.

## Task 5. Pitfall: non-unique key

A SKU is designed to uniquely identify one product and will be the basis of your join condition when you join against other tables. Having a non-unique key can cause serious data issues.

1. **Write a query** to identify all the product names for the SKU `'GGOEGPJC019099'`.

Possible solution:

#standardSQL # multiple records for this SKU SELECT DISTINCT v2ProductName, productSKU FROM \`data-to-insights.ecommerce.all_sessions_raw\` WHERE productSKU = 'GGOEGPJC019099'

2. Click **RUN**.

| v2ProductName | productSKU |
|---|---|
| 7" Dog Frisbee | GGOEGPJC019099 |
| 7" Dog Frisbee | GGOEGPJC019099 |
| Google 7-inch Dog Flying Disc Blue | GGOEGPJC019099 |

From the query results, it looks like there are three different names for the same product. In this example, there is a special character in one name and a slightly different name for another:

### Joining website data against your product inventory list

See the impact of joining on a dataset with multiple products for a single SKU. First, explore the product inventory dataset (the `products` table) to see if this SKU is unique there.

1. **Copy and paste** the below query:

#standardSQL # join in another table # products (has inventory) SELECT * FROM \`data-to-insights.ecommerce.products\` WHERE SKU = 'GGOEGPJC019099'

2. Click **RUN**.

### Join pitfall: Unintentional many-to-one SKU relationship

Next, join the inventory dataset against your website product names and SKUs so you can have the inventory stock level associated with each product for sale on the website.

1. **Copy and paste** the below query:

#standardSQL SELECT DISTINCT website.v2ProductName, website.productSKU, inventory.stockLevel FROM \`data-to-insights.ecommerce.all_sessions_raw\` AS website JOIN \`data-to-insights.ecommerce.products\` AS inventory ON website.productSKU = inventory.SKU WHERE productSKU = 'GGOEGPJC019099'

2. Click **RUN**.

What happens when you join the website table and the product inventory table on SKU? Do you now have inventory stock levels for the product?

Answer: Yes but the stockLevel is showing three times (one for each record)!

Next, run a query that shows the total stock level for each item in inventory.

3. **Copy and paste** the below query:

#standardSQL SELECT productSKU, SUM(stockLevel) AS total_inventory FROM ( SELECT DISTINCT website.v2ProductName, website.productSKU, inventory.stockLevel FROM `data-to-insights.ecommerce.all_sessions_raw` AS website JOIN `data-to-insights.ecommerce.products` AS inventory ON website.productSKU = inventory.SKU WHERE productSKU = 'GGOEGPJC019099' ) GROUP BY productSKU

4. Click **RUN**.

## Task 6. Join pitfall solution: use distinct SKUs before joining

What are the options to solve your triple counting dilemma? First, you need to only select distinct SKUs from the website before joining on other datasets.

- Write a query to return the count of distinct productSKU from `data-to-insights.ecommerce.all_sessions_raw`.

Possible solution:

#standardSQL SELECT COUNT(DISTINCT website.productSKU) AS distinct_sku_count FROM `data-to-insights.ecommerce.all_sessions_raw` AS website

Answer: 1,909 distinct SKUs from the website dataset

### Join pitfall: Losing data records after a join

Now you're ready to join against your product inventory dataset again.

1. **Copy and paste** the below query:

#standardSQL SELECT DISTINCT website.productSKU FROM `data-to-insights.ecommerce.all_sessions_raw` AS website JOIN `data-to-insights.ecommerce.products` AS inventory ON website.productSKU = inventory.SKU

2. Click **RUN**.

How many records were returned? All 1,909 distinct SKUs?

Answer: No, just 1,090 records

You lost 819 SKUs after joining the datasets, investigate by adding more specificity in your fields.

3. **Copy and paste** the below query:

#standardSQL # pull ID fields from both tables SELECT DISTINCT website.productSKU AS website_SKU, inventory.SKU AS inventory_SKU FROM `data-to-insights.ecommerce.all_sessions_raw` AS website JOIN `data-to-insights.ecommerce.products` AS inventory ON website.productSKU = inventory.SKU # IDs are present in both tables, how can we dig deeper?

4. Click **RUN**.

It appears the SKUs are present in both of those datasets after the join.

**Join pitfall solution: Selecting the correct join type and filtering for NULL**

The default JOIN type is an INNER JOIN which returns records only if there is a match on both the left and the right tables that are joined.

1. **Rewrite the previous query to use a different join type** to include all records from the website table, regardless of whether there is a match on a product inventory SKU record. Join type options: INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, CROSS JOIN

Possible solution:

#standardSQL # the secret is in the JOIN type # pull ID fields from both tables SELECT DISTINCT website.productSKU AS website_SKU, inventory.SKU AS inventory_SKU FROM `data-to-insights.ecommerce.all_sessions_raw` AS website LEFT JOIN `data-to-insights.ecommerce.products` AS inventory ON website.productSKU = inventory.SKU

2. Click **RUN**.

You have successfully used a LEFT JOIN to return all of the original 1,909 website SKUs in your results.

How many SKUs are missing from your product inventory set?

3. **Write a query** to filter on NULL values from the inventory table.

Possible solution:

#standardSQL # find product SKUs in website table but not in product inventory table SELECT DISTINCT website.productSKU AS website_SKU, inventory.SKU AS inventory_SKU FROM `data-to-insights.ecommerce.all_sessions_raw` AS website LEFT JOIN `data-to-insights.ecommerce.products` AS inventory ON website.productSKU = inventory.SKU WHERE inventory.SKU IS NULL

4. Click **RUN**.

5. **Copy and paste** the below query to confirm using one of the specific SKUs from the website dataset:

#standardSQL # you can even pick one and confirm SELECT * FROM `data-to-insights.ecommerce.products` WHERE SKU = 'GGOEGATJ060517' # query returns zero results

6. Click **RUN**.

Why might the product inventory dataset be missing SKUs?

Answer: Unfortunately, there is no easy answer. It is most likely a business-related question:

- Some SKUs could be digital products that you don't store in inventory
- Old products you sold in past website orders are no longer offered in current inventory
- Legitimate missing data from inventory and should be tracked

Are there any products are in the product inventory dataset but missing from the website?

7. Write a query using a different join type to investigate.

Possible solution:

```
#standardSQL # reverse the join # find records in website but not in inventory SELECT DISTINCT
website.productSKU AS website_SKU, inventory.SKU AS inventory_SKU FROM `data-to-
insights.ecommerce.all_sessions_raw` AS website RIGHT JOIN `data-to-insights.ecommerce.products` AS
inventory ON website.productSKU = inventory.SKU WHERE website.productSKU IS NULL
```

8. Click **RUN**.

Answer: Yes. There are two product SKUs missing from the website dataset

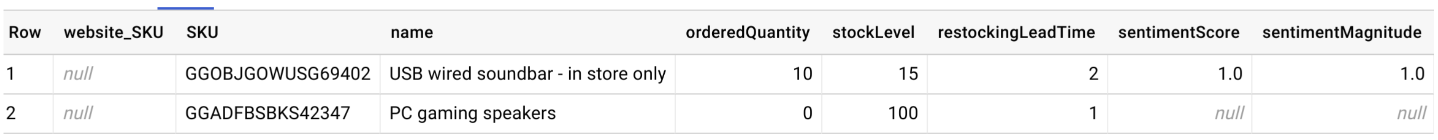
Next, add more fields from the product inventory dataset for more details.

9. **Copy and paste** the below query:

```
#standardSQL # what are these products? # add more fields in the SELECT STATEMENT SELECT
DISTINCT website.productSKU AS website_SKU, inventory.* FROM `data-to-
insights.ecommerce.all_sessions_raw` AS website RIGHT JOIN `data-to-insights.ecommerce.products` AS
inventory ON website.productSKU = inventory.SKU WHERE website.productSKU IS NULL
```

10. Click **RUN**.

Why would the below products be missing from the ecommerce website dataset?



| Row | website_SKU | SKU | name | orderedQuantity | stockLevel | restockingLeadTime | sentimentScore | sentimentMagnitude |
|---|---|---|---|---|---|---|---|---|
| 1 | null | GGOBJGOWUSG69402 | USB wired soundbar - in store only | 10 | 15 | 2 | 1.0 | 1.0 |
| 2 | null | GGADFBSBKS42347 | PC gaming speakers | 0 | 100 | 1 | null | null |

Possible answers:

- One new product (no orders, no sentimentScore) and one product that is "in store only"
- Another is a new product with 0 orders

Why would the new product not show up on your website dataset?

- The website dataset is past order transactions by customers brand new products which have never been sold won't show up in web analytics until they're viewed or purchased

**Note:** You typically will not see RIGHT JOINs in production queries. You would simply just do a LEFT JOIN and switch the ordering of the tables.

What if you wanted one query that listed all products missing from either the website or inventory?

11. Write a query using a different join type.

Possible solution:

#standardSQL SELECT DISTINCT website.productSKU AS website_SKU, inventory.SKU AS inventory_SKU FROM `data-to-insights.ecommerce.all_sessions_raw` AS website FULL JOIN `data-to-insights.ecommerce.products` AS inventory ON website.productSKU = inventory.SKU WHERE website.productSKU IS NULL OR inventory.SKU IS NULL

12. Click **RUN**.

You have your 819 + 2 = 821 product SKUs

LEFT JOIN + RIGHT JOIN = FULL JOIN which returns all records from both tables regardless of matching join keys. You then filter out where you have mismatches on either side

## Join pitfall: Unintentional Cross Join

Not knowing the relationship between data table keys (1:1, 1:N, N:N) can return unexpected results and also significantly reduce query performance.

The last join type is the CROSS JOIN.

Create a new table with a site-wide discount percent that you want to apply across products in the Clearance category.

Replacing the table named ***qwiklabs-****.

1. **Copy and paste** the below query:

#standardSQL CREATE OR REPLACE TABLE ecommerce.site_wide_promotion AS SELECT .05 AS discount;

2. Click **RUN**.

In the left pane, site_wide_promotion is now listed in the Resource section under `qwiklabs-gcp-xxx` > `ecommerce`.

3. **Copy and paste** the below query to find out how many products are in clearance:

SELECT DISTINCT productSKU, v2ProductCategory, discount FROM `data-to-insights.ecommerce.all_sessions_raw` AS website CROSS JOIN ecommerce.site_wide_promotion WHERE v2ProductCategory LIKE '%Clearance%'

4. Click **RUN**.

**Note**: In the syntax there is no join condition (e.g. ON or USING) for a CROSS JOIN. The field is simply multiplied against the first dataset or .05 discount across all items.

Let's see the impact of unintentionally adding more than one record in the discount table.

5. **Copy and paste** the below query to insert two more records into the promotion table:

#standardSQL INSERT INTO ecommerce.site_wide_promotion (discount) VALUES (.04), (.03);

6. Click **RUN**.

Next let's view the data values in the promotion table.

7. **Copy and paste** the below query:

#standardSQL SELECT discount FROM ecommerce.site_wide_promotion

8. Click **RUN**.

What happens when you apply the discount again across all 82 clearance products?

9. **Copy and paste** the below query:

#standardSQL # now what happens: SELECT DISTINCT productSKU, v2ProductCategory, discount FROM `data-to-insights.ecommerce.all_sessions_raw` AS website CROSS JOIN ecommerce.site_wide_promotion WHERE v2ProductCategory LIKE '%Clearance%'

10. Click **RUN**.

How many products are returned?

Answer: Instead of 82, you now have 246 returned which is more records than your original table started with.

Let's investigate the underlying cause by examining one product SKU.

11. **Copy and paste** the below query:

#standardSQL SELECT DISTINCT productSKU, v2ProductCategory, discount FROM `data-to-insights.ecommerce.all_sessions_raw` AS website CROSS JOIN ecommerce.site_wide_promotion WHERE v2ProductCategory LIKE '%Clearance%' AND productSKU = 'GGOEGOLC013299'

12. Click **RUN**.

What was the impact of the CROSS JOIN?

Answer:

Since there are 3 discount codes to cross join on, you are multiplying the original dataset by 3.

**Note:** This behavior isn't limited to cross joins, with a normal join you can unintentionally cross join when the data relationships are many-to-many this can easily result in returning millions or even billions of records unintentionally.

The solution is to know your data relationships before you join and don't assume keys are unique.

## Task 7. Deduplicating rows

At the start of the lab you wrote a query that showed multiple product names for a single SKU. Deduplicating records like this is a common skill for data analysts. Examine one way you can select only one product per SKU.

First, start with the query to show all product names per SKU.

1. **Copy and paste** the below query:

#standardSQL # recall the earlier query that showed multiple product_names for each SKU SELECT DISTINCT COUNT(DISTINCT v2ProductName) AS product_count, STRING_AGG(DISTINCT v2ProductName LIMIT 5) AS product_name, productSKU FROM `data-to-insights.ecommerce.all_sessions_raw` WHERE v2ProductName IS NOT NULL GROUP BY productSKU HAVING product_count > 1 ORDER BY product_count DESC

2. Click **RUN**.

Since most of the product names are extremely similar (and you want to map a single SKU to a single product), write a query to only choose one of the product_names. You will be using this StackOverflow post by Felipe Hoffa as inspiration.

3. **Copy and paste** the below query:

#standardSQL # take the one name associated with a SKU WITH product_query AS ( SELECT DISTINCT v2ProductName, productSKU FROM `data-to-insights.ecommerce.all_sessions_raw` WHERE v2ProductName IS NOT NULL ) SELECT k.* FROM ( # aggregate the products into an array and # only take 1 result SELECT ARRAY_AGG(x LIMIT 1)[OFFSET(0)] k FROM product_query x GROUP BY productSKU # this is the field you want deduplicated );

4. Click **RUN**.

You have successfully deduplicated the product names for each SKU. Experiment with the above query and your own datasets to deduplicate your fields before joining against other datasets.

## Task 8. Test your understanding

## Congratulations!

You've concluded this lab and worked through some serious SQL join pitfalls by identifying duplicate records and knowing when to use each type of JOIN. Nice work!

## End your lab

When you have completed your lab, click **End Lab**. Google Cloud Skills Boost removes the resources you've used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied

- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.