Troubleshooting and Solving Data Join Pitfalls

cloudskillsboost.google/focuses/3638

GSP412

Google Cloud Self-Paced Labs

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

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

To complete this lab, you need:

Access to a standard internet browser (Chrome browser recommended).

Note: Use an Incognito or private browser window to run this lab. This prevents any conflicts between your personal account and the Student account, which may cause extra charges incurred to your personal account.
Time to complete the lab---remember, once you start, you cannot pause a lab.

Note: If you already have your own personal Google Cloud account or project, do not use it for this lab to avoid extra charges to your account.

How to start your lab and sign in to the Google Cloud Console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is the **Lab Details** panel with the following:
   - The **Open Google Console** button
   - Time remaining
   - The temporary credentials that you must use for this lab
   - Other information, if needed, to step through this lab
2. Click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.

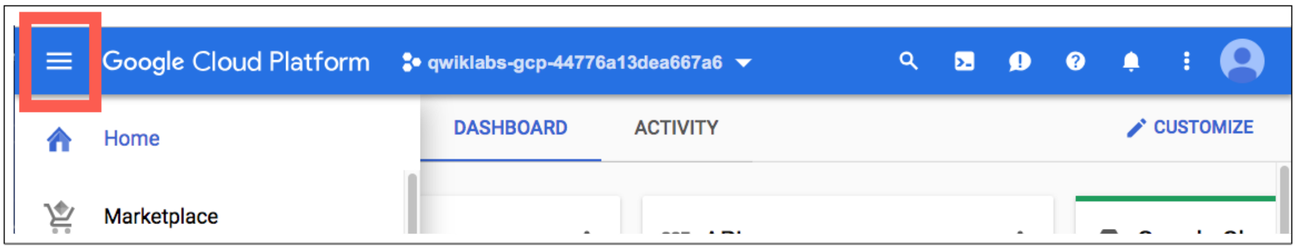
   Tip: Arrange the tabs in separate windows, side-by-side.

   Note: If you see the **Choose an account** dialog, click **Use Another Account**.
3. If necessary, copy the **Username** from the **Lab Details** panel and paste it into the **Sign in** dialog. Click **Next**.
4. Copy the **Password** from the **Lab Details** panel and paste it into the **Welcome** dialog. Click **Next**.

   Important: You must use the credentials from the left panel. Do not use your Google Cloud Skills Boost credentials. **Note:** Using your own Google Cloud account for this lab may incur extra charges.
5. Click through the subsequent pages:
   - Accept the terms and conditions.
   - Do not add recovery options or two-factor authentication (because this is a temporary account).
   - Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



Open the 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 the release notes.

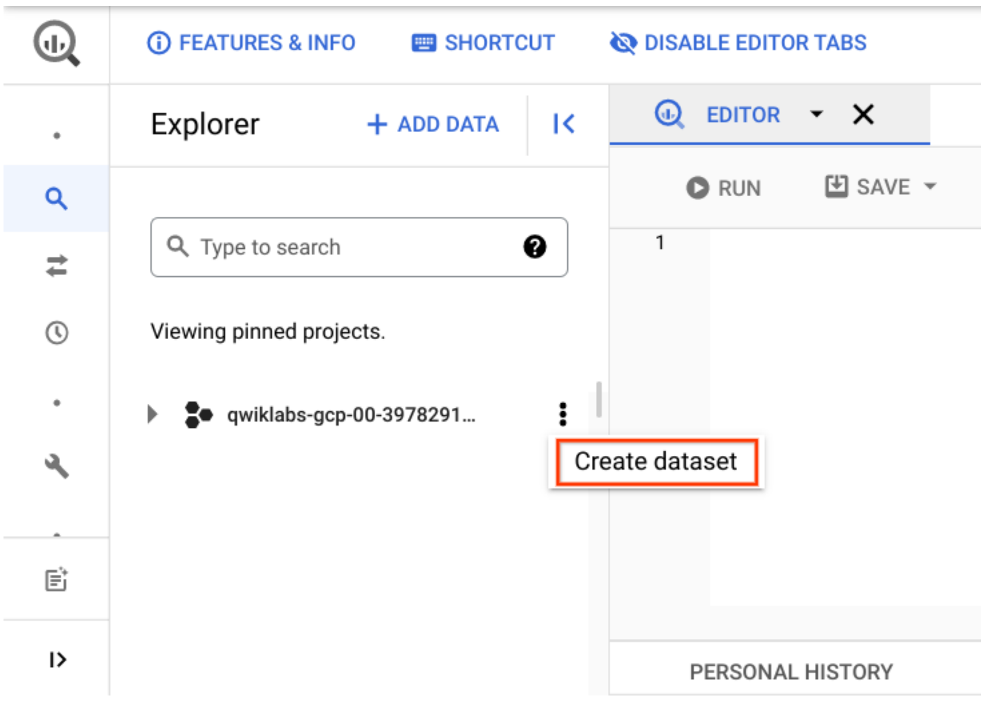
2. Click **Done**.

The BigQuery console opens.

Create a new dataset to store your tables

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

Click the three dots next to your Qwiklabs project ID and select **Create dataset**:



The **Create dataset** dialog opens.

3. 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` table listed under your project.

Click **Check my progress** to verify the objective.

Create a new dataset

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**.

1. Click **Navigation menu** > **BigQuery**.

The Welcome to BigQuery in the Cloud Console message box opens.

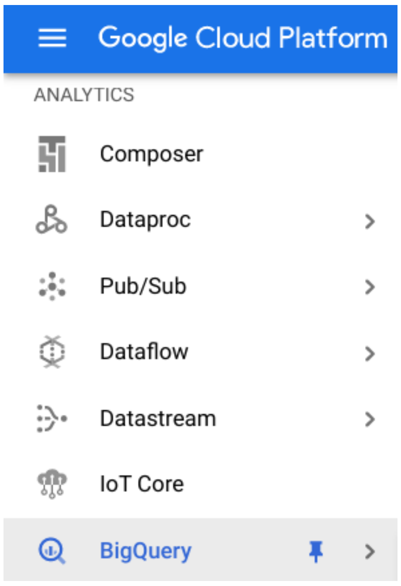
The Welcome to BigQuery in the Cloud Console message box provides a link to the quickstart guide and UI updates.

2. Click **Done**.

3. BigQuery public datasets are not displayed by default in the BigQuery web UI. To open the public datasets project, copy **data-to-insights**.

4. Click **Add Data** > **Pin a project** > **Enter Project Name**, then paste in the data-to-insights name. Click **Pin**.

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

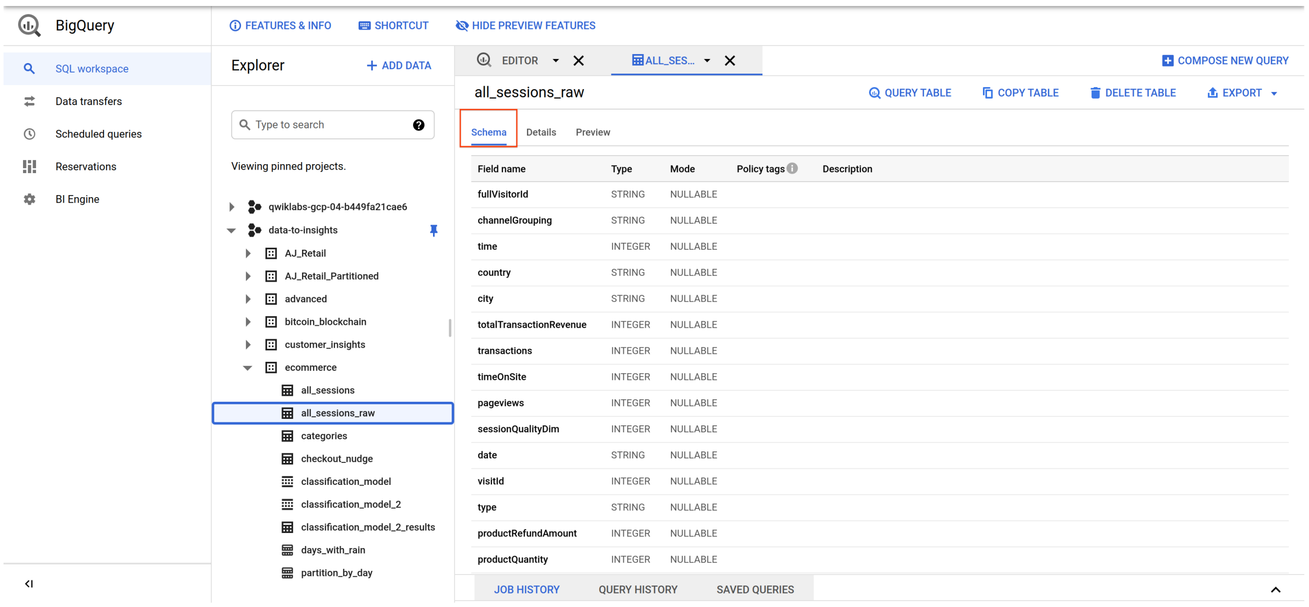


Examine the fields

Next, get familiar with the products and fields on the website you can use to create queries to analyze the dataset.

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

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



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		
productRefundAmount	INTEGER	NULLABLE		
productQuantity	INTEGER	NULLABLE		

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.

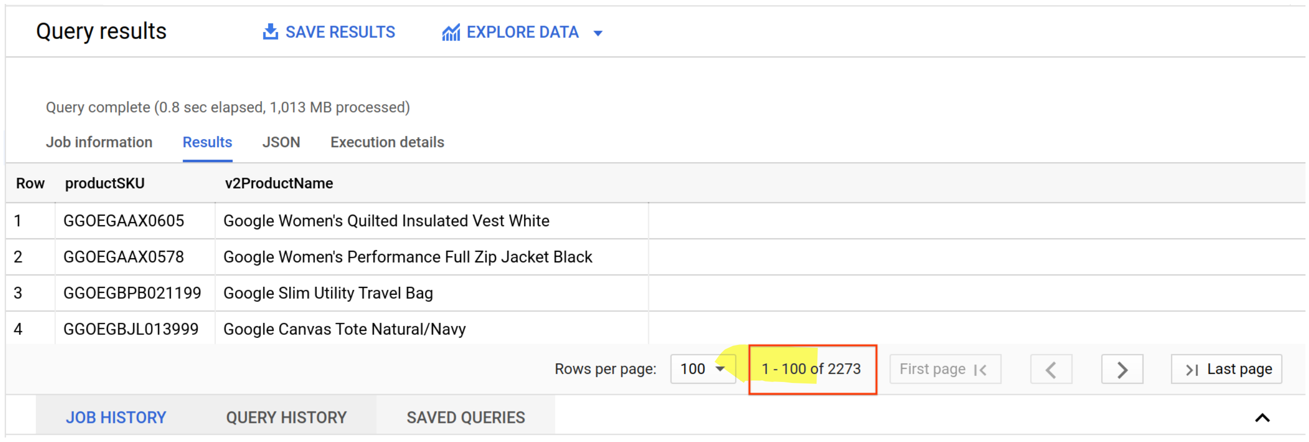
Examine the Records

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.

Find how many product names and product SKUs are on the website. **Copy and Paste** the below query in bigquery **EDITOR**.

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

Look at the pagination results in the console for the total number of records returned.



Query complete (0.8 sec elapsed, 1,013 MB processed)

Row	productSKU	v2ProductName
1	GGOEGAAX0605	Google Women's Quilted Insulated Vest White
2	GGOEGAAX0578	Google Women's Performance Full Zip Jacket Black
3	GGOEGBPB021199	Google Slim Utility Travel Bag
4	GGOEGBJL013999	Google Canvas Tote Natural/Navy

1 - 100 of 2273

But...do the results mean that there are that many unique product SKUs? One of the first queries you will run as a data analyst is looking at the uniqueness of your data values.

Clear the previous query and run 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`

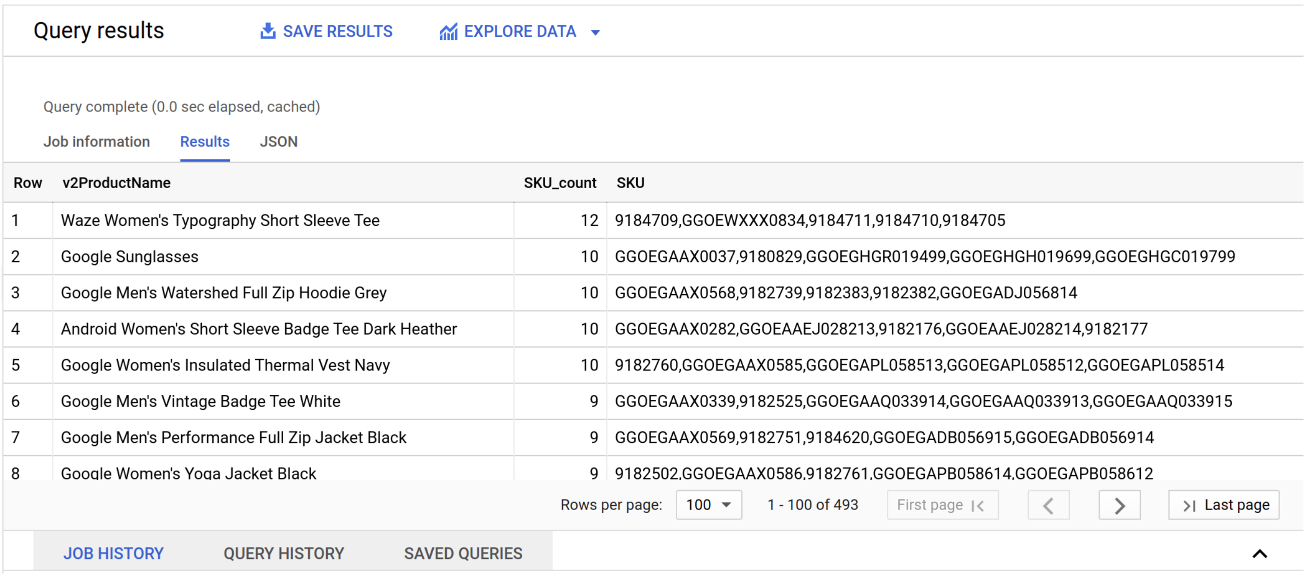
Examine relationship between SKU & Name

Let's determine which products have more than one SKU and which SKUs have more than one Product Name.

Clear the previous query and run the below query to determine if some product names have more than one SKU. Notice we use the STRING_AGG() function to aggregate all the product SKUs that are associated with one product name into comma separated values.

SELECT v2ProductName, COUNT(DISTINCT productSKU) AS SKU_count, STRING_AGG(DISTINCT productSKU LIMIT 5) AS SKU 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
Click **Run**.

Results:



Query complete (0.0 sec elapsed, cached)

Row	v2ProductName	SKU_count	SKU
1	Waze Women's Typography Short Sleeve Tee	12	9184709,GGOEWXXX0834,9184711,9184710,9184705
2	Google Sunglasses	10	GGOEGAAX0037,9180829,GGOEGHGR019499,GGOEGHGH019699,GGOEGHGC019799
3	Google Men's Watershed Full Zip Hoodie Grey	10	GGOEGAAX0568,9182739,9182383,9182382,GGOEGADJ056814
4	Android Women's Short Sleeve Badge Tee Dark Heather	10	GGOEGAAX0282,GGOEAAEJ028213,9182176,GGOEAAEJ028214,9182177
5	Google Women's Insulated Thermal Vest Navy	10	9182760,GGOEGAAX0585,GGOEGAPL058513,GGOEGAPL058512,GGOEGAPL058514
6	Google Men's Vintage Badge Tee White	9	GGOEGAAX0339,9182525,GGOEGAAQ033914,GGOEGAAQ033913,GGOEGAAQ033915
7	Google Men's Performance Full Zip Jacket Black	9	GGOEGAAX0569,9182751,9184620,GGOEGADB056915,GGOEGADB056914
8	Google Women's Yoga Jacket Black	9	9182502,GGOEGAAX0586,9182761,GGOEGAPB058614,GGOEGAPB058612

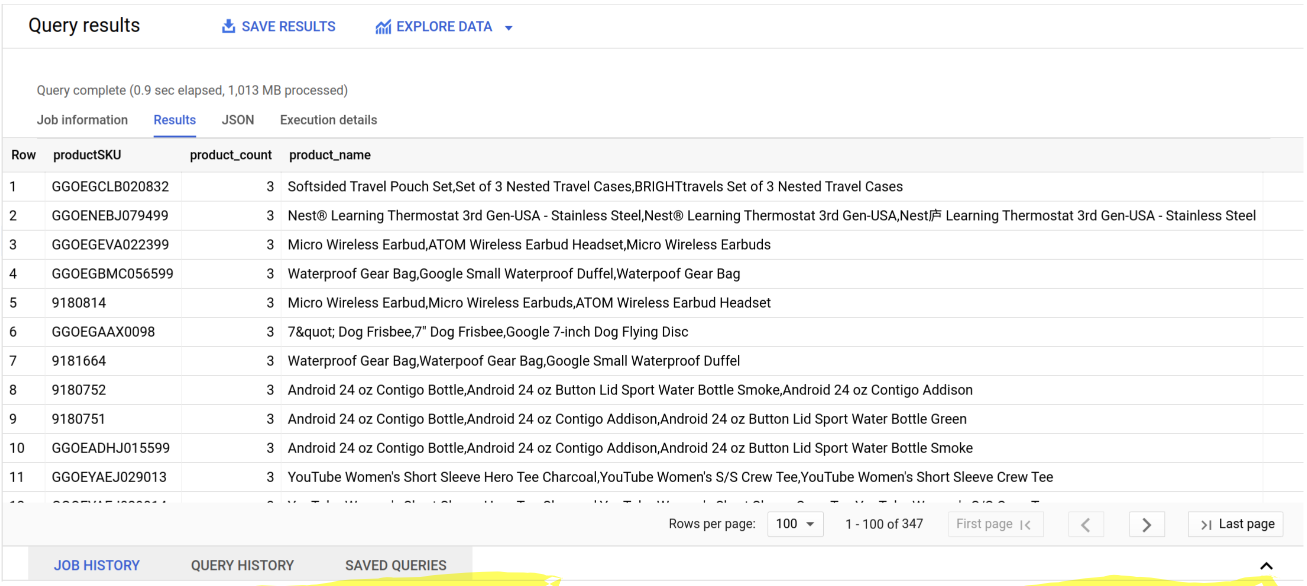
Rows per page: 100 1 - 100 of 493 First page Last page

You can see on the ecommerce website catalog that each product name may have multiple options (size, color) -- which are sold as separate SKUs.

So we have seen that 1 Product can have 12 SKUs. What about 1 SKU? Should it be allowed to belong to more than 1 product?

Clear the previous query and run the below query to find out.

SELECT productSKU, COUNT(DISTINCT v2ProductName) AS product_count, STRING_AGG(DISTINCT v2ProductName LIMIT 5) AS product_name 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



Query complete (0.9 sec elapsed, 1,013 MB processed)

Row	productSKU	product_count	product_name
1	GGOEGCLB020832	3	Softsided Travel Pouch Set,Set of 3 Nested Travel Cases,BRIGHTtravels Set of 3 Nested Travel Cases
2	GGOENEBJ079499	3	Nest® Learning Thermostat 3rd Gen-USA - Stainless Steel,Nest® Learning Thermostat 3rd Gen-USA,Nest廌 Learning Thermostat 3rd Gen-USA - Stainless Steel
3	GGOEGEVA022399	3	Micro Wireless Earbud,ATOM Wireless Earbud Headset,Micro Wireless Earbuds
4	GGOEGBMC056599	3	Waterproof Gear Bag,Google Small Waterproof Duffel,Waterpoof Gear Bag
5	9180814	3	Micro Wireless Earbud,Micro Wireless Earbuds,ATOM Wireless Earbud Headset
6	GGOEGAAX0098	3	7&quot; Dog Frisbee,7" Dog Frisbee,Google 7-inch Dog Flying Disc
7	9181664	3	Waterproof Gear Bag,Waterpoof Gear Bag,Google Small Waterproof Duffel
8	9180752	3	Android 24 oz Contigo Bottle,Android 24 oz Button Lid Sport Water Bottle Smoke,Android 24 oz Contigo Addison
9	9180751	3	Android 24 oz Contigo Bottle,Android 24 oz Contigo Addison,Android 24 oz Button Lid Sport Water Bottle Green
10	GGOEADHJ015599	3	Android 24 oz Contigo Bottle,Android 24 oz Contigo Addison,Android 24 oz Button Lid Sport Water Bottle Smoke
11	GGOEYAEJ029013	3	YouTube Women's Short Sleeve Hero Tee Charcoal,YouTube Women's S/S Crew Tee,YouTube Women's Short Sleeve Crew Tee

Rows per page: 100 1 - 100 of 347 First page Last page

Note: Try replacing STRING_AGG() with ARRAY_AGG() instead. Pretty cool, right? BigQuery natively supports nested array values. You can learn more here.
You will see why this many-to-many data relationship will be an issue in the next section.

Click **Check my progress** to verify the objective.

Identify a key field in your ecommerce dataset

Pitfall: non-unique key

In inventory tracking, a SKU is designed to uniquely identify one and only one product. For us, it will be the basis of your JOIN condition when you lookup information from other tables. Having a non-unique key can cause serious data issues as you will see.

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

Possible Solution:

SELECT DISTINCT v2ProductName, productSKU FROM `data-to-insights.ecommerce.all_sessions_raw` WHERE productSKU = 'GGOEGPJC019099'
Click **Run**.

v2ProductName	productSKU
7&quot; 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

Let's 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.

Clear the previous query and run the below query.

SELECT SKU, name, stockLevel FROM `data-to-insights.ecommerce.products` WHERE SKU = 'GGOEGPJC019099'

Join pitfall: Unintentional many-to-one SKU relationship

We now have two datasets: one for inventory stock level and the other for our website analytics. Let's 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.

Clear the previous query and run the below query.

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'
Next, let's expand our previous query to simply SUM the inventory available by product.

Clear the previous query and run the below query.

WITH inventory_per_sku AS (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') SELECT productSKU, SUM(stockLevel) AS total_inventory FROM inventory_per_sku GROUP BY productSKU
Oh no! It is 154 x 3 = 462 or triple counting the inventory! This is called an unintentional cross join (a topic we'll revisit later).

Click **Check my progress** to verify the objective.

Pitfall: non-unique key

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.

We know that there can be more than one product name (like 7" Dog Frisbee) that can share a single SKU.

Let's gather all the possible names into an array:

SELECT productSKU, ARRAY_AGG(DISTINCT v2ProductName) AS push_all_names_into_array FROM `data-to-insights.ecommerce.all_sessions_raw` WHERE productSKU = 'GGOEGAAX0098' GROUP BY productSKU
Now instead of having a row for every Product Name, we only have a row for each unique SKU.

If you wanted to deduplicate the product names, you could even LIMIT the array like so:

SELECT productSKU, ARRAY_AGG(DISTINCT v2ProductName LIMIT 1) AS push_all_names_into_array FROM `data-to-insights.ecommerce.all_sessions_raw` WHERE productSKU = 'GGOEGAAX0098' GROUP BY productSKU

Join pitfall: Losing data records after a join

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

Clear the previous query and run 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
```
It seems we lost 819 SKUs after joining the datasets Let's investigate by adding more specificity in your fields (one SKU column from each dataset):

Clear the previous query and run 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?
```
It appears the SKUs are present in both of those datasets after the join for these 1,090 records. How can we find the missing records?

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 SKU match on both the left and the right tables that are joined.

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
```
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?

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
```
Click **Run**.

Question: How many products are missing?

Answer: 819 products are missing (SKU IS NULL) from your product inventory dataset.

Clear the previous query and run 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
```
Now, what about the reverse situation? Are there any products in the product inventory dataset but missing from the website?

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
```
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.

Clear the previous query and run 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
```
Why would the below products be missing from the ecommerce website dataset?

website_SKU	SKU	name	orderedQuantity	stockLevel	restockingLeadTime	sentimentScore	sentim

null	GGOBJGOWUSG69402	USB wired soundbar - in store only	10	15	2	1.0	1.0
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.

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?

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
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 applied across products in the Clearance category.

Clear the previous query and run the below query.

#standardSQL CREATE OR REPLACE TABLE ecommerce.site_wide_promotion AS SELECT .05 AS discount;
In the left pane, `site_wide_promotion` is now listed in the Resource section under your project and dataset.

Clear the previous query and run 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%'
Note: For a CROSS JOIN you will notice there is no join condition (e.g. ON or USING). 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.

Clear the previous query and run the below query to insert two more records into the promotion table.

INSERT INTO ecommerce.site_wide_promotion (discount) VALUES (.04), (.03);
Next let's view the data values in the promotion table.

Clear the previous query and run the below query.

SELECT discount FROM ecommerce.site_wide_promotion
How many records were returned?

Answer: 3

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

Clear the previous query and run the below query.

```
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%'
```

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.

Clear the previous query and run 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'
```

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

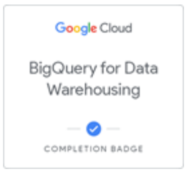
The solution is to know your data relationships before you join and don't assume keys are unique.

Click **Check my progress** to verify the objective.

Join pitfall solution

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!



Finish your Quest

This self-paced lab is part of the Quest, BigQuery for Data Warehousing Quest. A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above, to recognize your achievement. You can make your badge (or badges) public and link to them in your online resume or social media account. Enroll in this Quest and get immediate completion credit if you've taken this lab. See other available Quests.

Take your next lab

Continue your Quest with another lab, for example Working with JSON, Arrays, and Structs in BigQuery, or check out these labs:

- Introduction to SQL for BigQuery and Cloud SQL
- Predict Taxi Fare with a BigQuery ML Forecasting Model

Next steps/learn more

Already have a Google Analytics account and want to query your own datasets in BigQuery? Follow this export guide.

Learn more about best practices that provide guidance on Optimizing Query Computation.

If you want to practice with more SQL syntax for JOINS, check out the BigQuery JOIN documentation.

Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. Our classes include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. Certifications help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated: April 4, 2022

Lab Last Tested: March 28, 2022