
Troubleshooting and Solving Data Join Pitfalls

Overview

This lab focused on identifying and solving common data join pitfalls in BigQuery using an ecommerce dataset. The key lessons learned include:

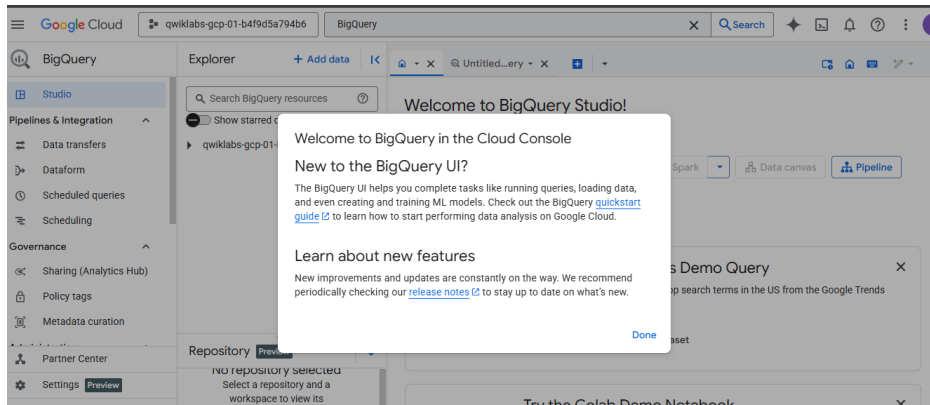
1. **Before** joining tables, it's crucial to understand whether the relationship between keys is one-to-one, one-to-many, or many-to-many.
2. **Join Types:**
 - a. **INNER JOIN:** Returns only matching records from both tables
 - b. **LEFT JOIN:** Returns all records from the left table and matching records from the right
 - c. **RIGHT JOIN:** Returns all records from the right table and matching records from the left
 - d. **FULL JOIN:** Returns all records when there's a match in either table
 - e. **CROSS JOIN:** Returns the Cartesian product of both tables

Task 1. Create a new dataset to store your tables

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

1. Click the three dots next to your Project ID and select **Create dataset**.
The **Create dataset** dialog opens.
2. Set the *dataset ID* to ecommerce.
3. 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.



Create dataset

Project ID *
qwiklabs-gcp-01-b4f9d5a794b6 [Change](#)

Dataset ID *

Letters, numbers, and underscores allowed

Location type ?

☐ Region
Specify a region to colocate your datasets with other Google Cloud services.

☒ Multi-region
Allow BigQuery to select a region within a group to achieve higher quota limits.

i Some locations have been restricted due to a policy set by your organization. [Learn more about restricting locations.](#)

Multi-region *

External Dataset
The selected region supports the following external dataset types: Cloud Spanner

☐ Link to an external dataset ?

[Create dataset](#) [Cancel](#)

Task 2. Pin the lab project in BigQuery

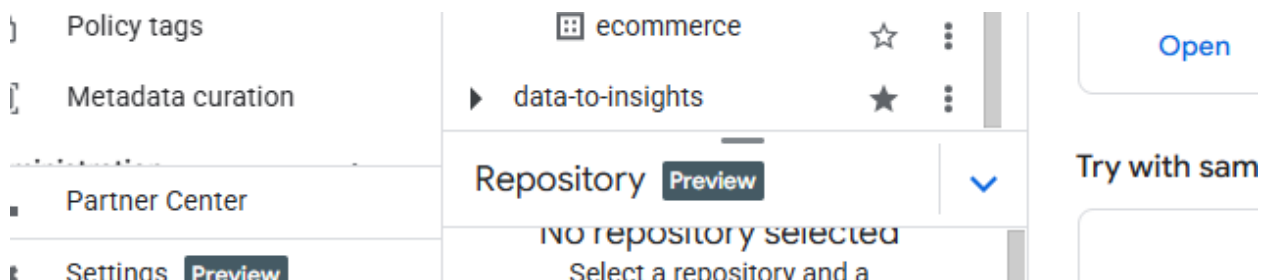
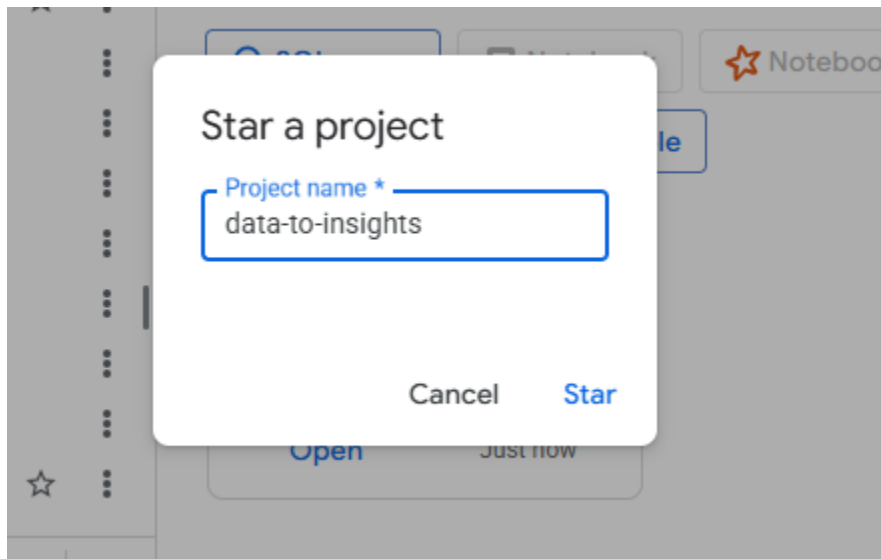
Scenario: Your team provides you with a new dataset on the inventory stock levels for each of your products for sale on your ecommerce website. You want to become

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**.

1. In the Google Cloud console, in the **Navigation menu** (☰) click **BigQuery**.
2. Click **Done**.
3. BigQuery public datasets are not displayed by default. To open the public datasets project, copy **data-to-insights** (to paste in a dialog in the next step).
4. Click **+ Add > Star a project by name** then paste the data-to-insights name.
5. Click **Star**.

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



Task 3. Examine the fields

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

1. In the left pane in the Resources section, navigate to data-to-insights > ecommerce > all_sessions_raw.
2. On the right, under the Query editor, click the **Schema** tab to see the Fields and information about each field.

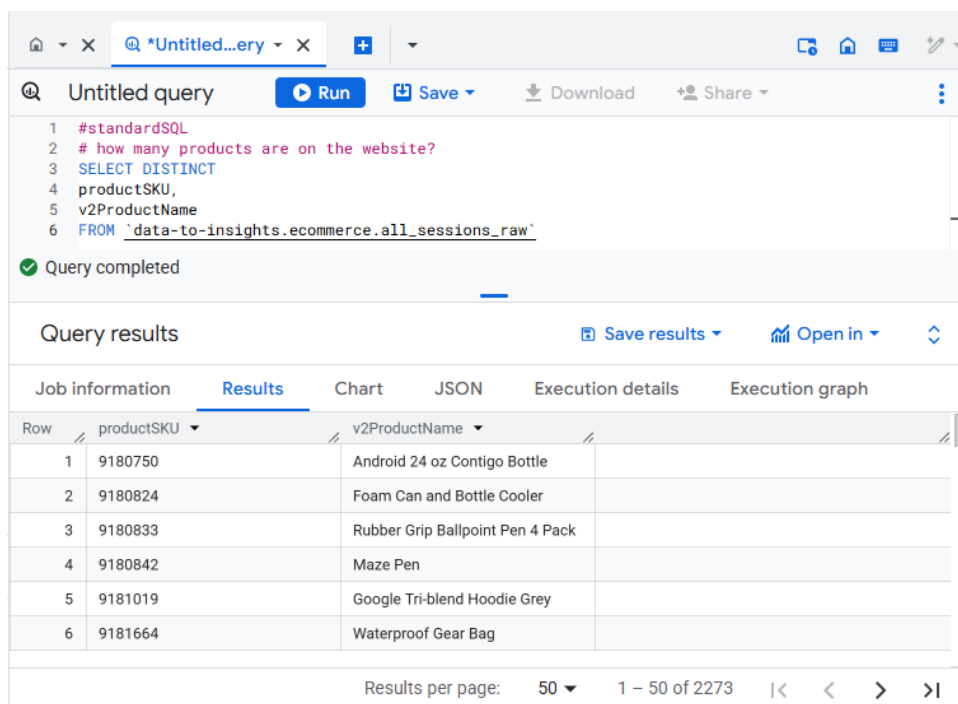
Task 4. Identify a key field in your ecommerce dataset

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 bigquery **EDITOR**:



The screenshot shows the BigQuery Editor interface. At the top, there's a tab labeled "Untitled query". Below it, a SQL query is entered:

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

Below the query, a status bar indicates "Query completed". Underneath, the "Query results" section is visible, showing a table with 6 rows and 2 columns: "productSKU" and "v2ProductName". The results are as follows:

Row	productSKU	v2ProductName
1	9180750	Android 24 oz Contigo Bottle
2	9180824	Foam Can and Bottle Cooler
3	9180833	Rubber Grip Ballpoint Pen 4 Pack
4	9180842	Maze Pen
5	9181019	Google Tri-blend Hoodie Grey
6	9181664	Waterproof Gear Bag

At the bottom of the results section, it shows "Results per page: 50" and "1 - 50 of 2273".

3. Clear the previous query and run the below query to list the number of distinct SKUs are listed using DISTINCT:

Untitled query

```

1 #standardSQL
2 # find the count of unique SKUs
3 SELECT
4 DISTINCT
5 productSKU
6 FROM `data-to-insights.ecommerce.all_sessions_raw`

```

Query completed

Query results

Job information Results Chart JSON Execution details Execution graph

Row	productSKU
1	9180750
2	9180793
3	9180833
4	9180838
5	9180844
6	9180905

Results per page: 50 1 – 50 of 1909

Examine the relationship between SKU & Name

Now determine which products have more than one SKU and which SKUs have more than one Product Name.

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

Untitled query

```

1 SELECT
2 v2ProductName,
3 COUNT(DISTINCT productSKU) AS SKU_count,
4 STRING_AGG(DISTINCT productSKU LIMIT 5) AS SKU
5 FROM `data-to-insights.ecommerce.all_sessions_raw`
6 WHERE productSKU IS NOT NULL
7 GROUP BY v2ProductName
8 HAVING SKU_count > 1
9 ORDER BY SKU_count DESC

```

Query completed

Query results

Job information Results Chart JSON Execution details Execution graph

Row	v2ProductName	SKU_count	SKU
1	Waze Women's Typography Sho...	12	GGOEWALJ083413,9184705,91...
2	Google Sunglasses	10	GGOEGAAX0037,9180826,GGO...
3	Google Men's Watershed Full Zi...	10	GGOEGAAX0568,GGOEGADJ056814,GGOEGADJ056818,9182739,GGOEGADJ056813

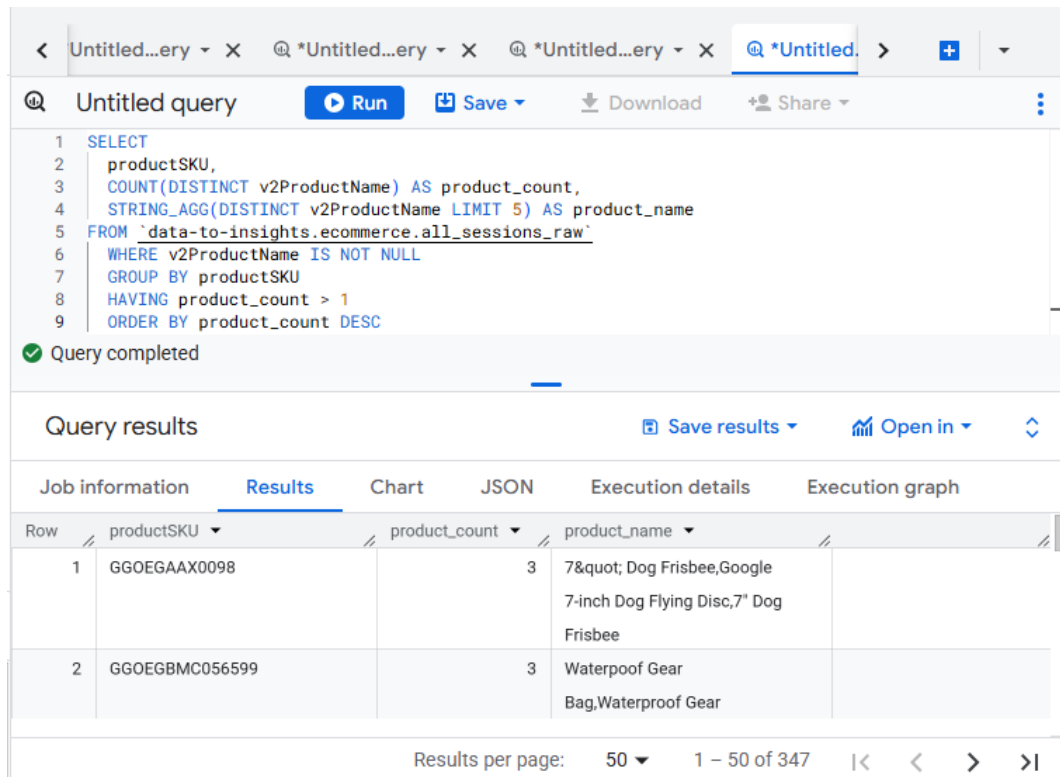
Results per page: 50 1 – 50 of 493

2. Click **Run**.

The [ecommerce website catalog](#) shows that each product name may have multiple options (size, color) -- which are sold as separate SKUs.

So you 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:



Query completed

Query results

Save results Open in

Job information Results Chart JSON Execution details Execution graph

Row	productSKU	product_count	product_name
1	GGOEGAAX0098	3	7" Dog Frisbee, Google 7-inch Dog Flying Disc, 7" Dog Frisbee
2	GGOEGBMC056599	3	Waterproof Gear Bag, Waterproof Gear

Results per page: 50 1 - 50 of 347

Task 5. 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.

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

Query completed

Query results

Job information Results Chart JSON Execution details Execution graph

Row	v2ProductName	productSKU
1	7" Dog Frisbee	GGOEGPJC019099
2	7" Dog Frisbee	GGOEGPJC019099
3	Google 7-inch Dog Flying Disc B...	GGOEGPJC019099

Joining website data against your product inventory list

Now 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:

Query completed

Query results

Job information Results Chart JSON Execution details Execution graph

Row	SKU	name	stockLevel
1	GGOEGPJC019099	7" Dog Frisbee	154

Join pitfall: Unintentional many-to-one SKU relationship

You now have two datasets: one for inventory stock level and the other for our website analytics. 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. Clear the previous query and run the below query:

```
1 SELECT DISTINCT
2   website.v2ProductName,
3   website.productSKU,
4   inventory.stockLevel
5 FROM `data-to-insights.ecommerce.all_sessions_raw` AS website
6 JOIN `data-to-insights.ecommerce.products` AS inventory
7   ON website.productSKU = inventory.SKU
8 WHERE productSKU = 'GGOEGPJC019099'
```

Query completed

Query results

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	v2ProductName	productSKU	stockLevel		
1	Google 7-inch Dog Flying Disc B...	GGOEGPJC019099	154		
2	7" Dog Frisbee	GGOEGPJC019099	154		
3	7" Dog Frisbee	GGOEGPJC019099	154		

2. Clear the previous query and run the below query:

```
1 WITH inventory_per_sku AS (
2   SELECT DISTINCT
3     website.v2ProductName,
4     website.productSKU,
5     inventory.stockLevel
6 FROM `data-to-insights.ecommerce.all_sessions_raw` AS website
7 JOIN `data-to-insights.ecommerce.products` AS inventory
8   ON website.productSKU = inventory.SKU
9 WHERE productSKU = 'GGOEGPJC019099'
10 )
11
```

Query completed

Query results

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	productSKU	total_inventory			
1	GGOEGPJC019099	462			

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.

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

1. Gather all the possible names into an array:

```
1 SELECT
2   productSKU,
3   ARRAY_AGG(DISTINCT v2ProductName) AS push_all_names_into_array
4 FROM `data-to-insights.ecommerce.all_sessions_raw`
5 WHERE productSKU = 'GG0EGAA0098'
6 GROUP BY productSKU
```

✓ Query completed

Query results [Save results](#) [Open in](#)

Job information **Results** Chart JSON Execution details Execution graph

Row	productSKU	push_all_names_into_array
1	GG0EGAA0098	7" Dog Frisbee Google 7-inch Dog Flying Disc 7" Dog Frisbee

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

Untitled query [Run](#) [Save](#) [Download](#) [Share](#)

```
1 SELECT
2   productSKU,
3   ARRAY_AGG(DISTINCT v2ProductName LIMIT 1) AS push_all_names_into_array
4 FROM `data-to-insights.ecommerce.all_sessions_raw`
5 WHERE productSKU = 'GG0EGAA0098'
6 GROUP BY productSKU
```

✓ Query completed

Query results [Save results](#) [Open in](#)

Job information **Results** Chart JSON Execution details Execution graph

Row	productSKU	push_all_names_into_array
1	GG0EGAA0098	7" Dog Frisbee

Join pitfall: losing data records after a join

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

1. Clear the previous query and run the below query:

Query completed

Query results

Job information Results Chart JSON Execution details Execution graph

Row	productSKU
1	9180793
2	9180833
3	9180838
4	9180844
5	9180905
6	9181019

Results per page: 50 1 - 50 of 1090

2. Clear the previous query and run the below query:

Query completed

Query results

Job information Results Chart JSON Execution details Execution graph

Row	website_SKU	inventory_SKU
1	9180833	9180833
2	9180838	9180838
3	9180844	9180844
4	9180905	9180905
5	9181019	9181019

Results per page: 50 1 - 50 of 1090

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.

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.

Untitled query Run Save Download Share

```

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

```

Query completed

Query results Save results Open in

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	website_SKU	inventory_SKU			
1	9180833	9180833			
2	9180838	9180838			
3	9180840	9180840			
4	9180842	null			
5	9180844	9180844			

Results per page: 50 1 - 50 of 1909

Click **Run**

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

Possible solution:

Untitled query Run Save Download Share

```

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

```

Query completed

Query results Save results Open in

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	website_SKU	inventory_SKU			
1	9182668	null			
2	9182838	null			
3	9183211	null			
4	10 93132	null			

Results per page: 50 1 - 50 of 819

- Clear the previous query and run the below query to confirm using one of the specific SKUs from the website dataset:

Untitled query Run Save Download Share

```

1 #standardSQL
2 # you can even pick one and confirm
3 SELECT * FROM `data-to-insights.ecommerce.products`
4 WHERE SKU = 'GG0EGATJ060517'
5 # query returns zero results

```

Query completed

Query results Save results Open in

Job information **Results** Chart JSON Execution details Execution graph

There is no data to display.

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

Possible solution:

Untitled query Run Save Download Share

```

1 #standardSQL
2 # reverse the join
3 # find records in website but not in inventory
4 SELECT DISTINCT
5 website.productSKU AS website_SKU,
6 inventory.SKU AS inventory_SKU
7 FROM `data-to-insights.ecommerce.all_sessions_raw` AS website
8 RIGHT JOIN `data-to-insights.ecommerce.products` AS inventory
9 ON website.productSKU = inventory.SKU
10 WHERE website.productSKU IS NULL

```

Query completed

Query results Save results Open in

Job information **Results** Chart JSON Execution details Execution graph

Row	website_SKU	inventory_SKU
1	null	GGADFBBSBK542347
2	null	GG0BJGOWUSG69402

2. Click **Run**.
3. Clear the previous query and run the below query:

Untitled query Run Save Download Share

```

1 #standardSQL
2 # what are these products?
3 # add more fields in the SELECT STATEMENT
4 SELECT DISTINCT
5 website.productSKU AS website_SKU,
6 inventory.*
7 FROM `data-to-insights.ecommerce.all_sessions_raw` AS website
8 RIGHT JOIN `data-to-insights.ecommerce.products` AS inventory
9 ON website.productSKU = inventory.SKU
10 WHERE website.productSKU IS NULL

```

Query completed

Query results Save results Open in

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	website_SKU	SKU	name	orderedQuantity	
1	null	GGOBJGOWUSG69402	USB wired soundbar - in store o...		
2	null	GGADFBBSKS42347	PC gaming speakers		

Results per page: 50 1 - 2 of 2

1. Write a query using a different join type.

Possible solution:

Untitled query Run Save Download Share

```

1 #standardSQL
2 SELECT DISTINCT
3 website.productSKU AS website_SKU,
4 inventory.SKU AS inventory_SKU
5 FROM `data-to-insights.ecommerce.all_sessions_raw` AS website
6 FULL JOIN `data-to-insights.ecommerce.products` AS inventory
7 ON website.productSKU = inventory.SKU
8 WHERE website.productSKU IS NULL OR inventory.SKU IS NULL

```

Query completed

Query results Save results Open in

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	website_SKU	inventory_SKU			
1	GGOEGAAX0332	null			
2	GGOEGAE031014	null			
3	GGOEGAWC062150	null			
4	GGOEGATB060215	null			
5	GGOEGAAJ073017	null			

Results per page: 50 1 - 50 of 821

2. 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

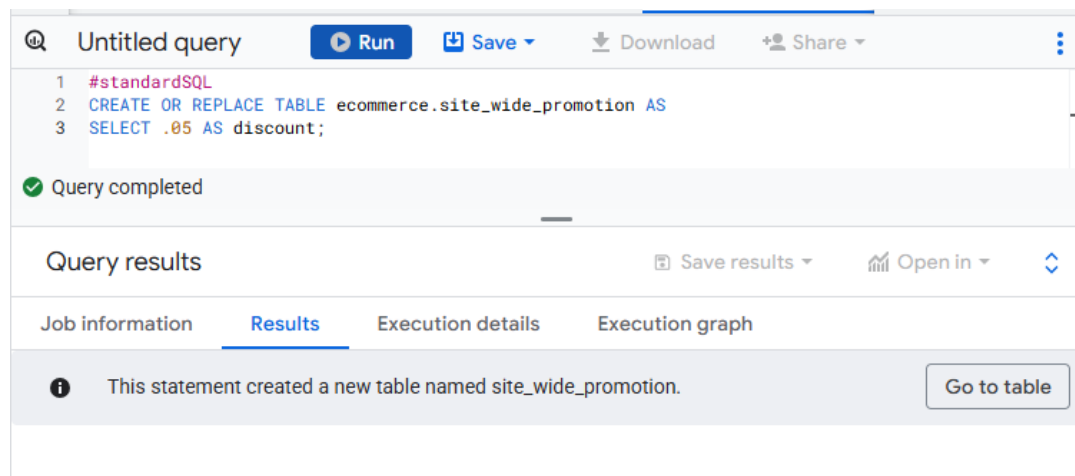
Join pitfall: unintentional cross join

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.

1. Clear the previous query and run the below query:



In the left pane, site_wide_promotion is now listed in the Resource section under your project and dataset.

2. Clear the previous query and run the below query to find out how many products are in clearance:

Untitled query Run Save Download Share

```

1 SELECT DISTINCT
2   productSKU,
3   v2ProductCategory,
4   discount
5 FROM `data-to-insights.ecommerce.all_sessions_raw` AS website
6 CROSS JOIN ecommerce.site_wide_promotion
7 WHERE v2ProductCategory LIKE '%Clearance%'

```

Query completed

Query results Save results Open in

Job information	Results	Chart	JSON	Execution details	Execution graph
Row	productSKU	v2ProductCategory	discount		
1	GGOEGAA0338	Home/Clearance Sale/	0.05		
2	GGOEAHPA004110	Home/Clearance Sale/	0.05		
3	GGOEAOCH077899	Home/Clearance Sale/	0.05		
4	GGOEGCBB074399	Home/Clearance Sale/	0.05		
5	GGOEGFQB013799	Home/Clearance Sale/	0.05		
6	GGOEGOAA017199	Home/Clearance Sale/	0.05		

Results per page: 50 1 – 50 of 82

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

Untitled query Run Save Download Share

```

1 INSERT INTO ecommerce.site_wide_promotion (discount)
2 VALUES (.04),

```

Query completed

Query results Save results Open in

Job information	Results	Execution details	Execution graph
<p>This statement added 2 rows to site_wide_promotion.</p> <p>Go to table</p>			

- Clear the previous query and run the below query:

Untitled query Run Save Download Share

```
1 SELECT discount FROM ecommerce.site_wide_promotion
```

✓ Query completed

Query results Save results Open in

Job information **Results** Chart JSON Execution details Execution graph

Row	discount
1	0.05
2	0.04
3	0.03

5. Clear the previous query and run the below query:

Untitled query Run Save Download Share

```
1 SELECT DISTINCT
2 productSKU,
3 v2ProductCategory,
4 discount
5 FROM `data-to-insights.ecommerce.all_sessions_raw` AS website
6 CROSS JOIN ecommerce.site_wide_promotion
7 WHERE v2ProductCategory LIKE '%Clearance%'
```

✓ Query completed

Query results Save results Open in

Job information **Results** Chart JSON Execution details Execution graph

Row	productSKU	v2ProductCategory	discount
1	GGOEGBPB021199	Home/Clearance Sale/	0.04
2	GGOEGCBQ016499	Home/Clearance Sale/	0.03
3	GGOEGFYQ016599	Home/Clearance Sale/	0.05
4	GGOEGFYQ016599	Home/Clearance Sale/	0.04
5	GGOEGOCC077299	Home/Clearance Sale/	0.05

Results per page: 50 1 – 50 of 246

Now investigate the underlying cause by examining one product SKU.

6. Clear the previous query and run the below query:

Untitled query

Run

Save

Download

Share

```
1 #standardSQL
2 SELECT DISTINCT
3 productSKU,
4 v2ProductCategory,
5 discount
6 FROM `data-to-insights.ecommerce.all_sessions_raw` AS website
7 CROSS JOIN ecommerce.site_wide_promotion
8 WHERE v2ProductCategory LIKE '%Clearance%'
9 AND productSKU = 'GGOEGOLC013299'
```

Query completed

Query results

Save results

Open in

Job information

Results

Chart

JSON

Execution details

Execution graph

Row	productSKU	v2ProductCategory	discount
1	GGOEGOLC013299	Home/Clearance Sale/	0.05
2	GGOEGOLC013299	Home/Clearance Sale/	0.03
3	GGOEGOLC013299	Home/Clearance Sale/	0.04

Results per page: 50 1 - 3 of 3

Conclusion

This lab provided valuable hands-on experience with real-world data join challenges. By understanding these pitfalls and solutions, you can ensure more accurate data analysis and reporting in BigQuery