# 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

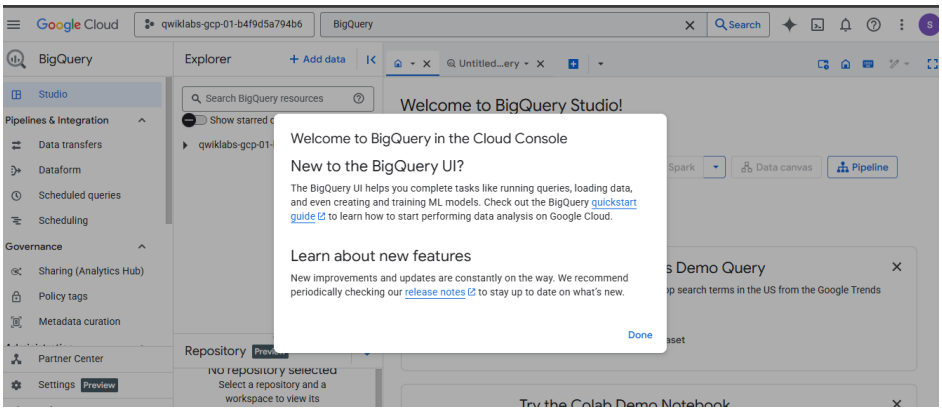
**Task 1. Create a new dataset to store your tables**

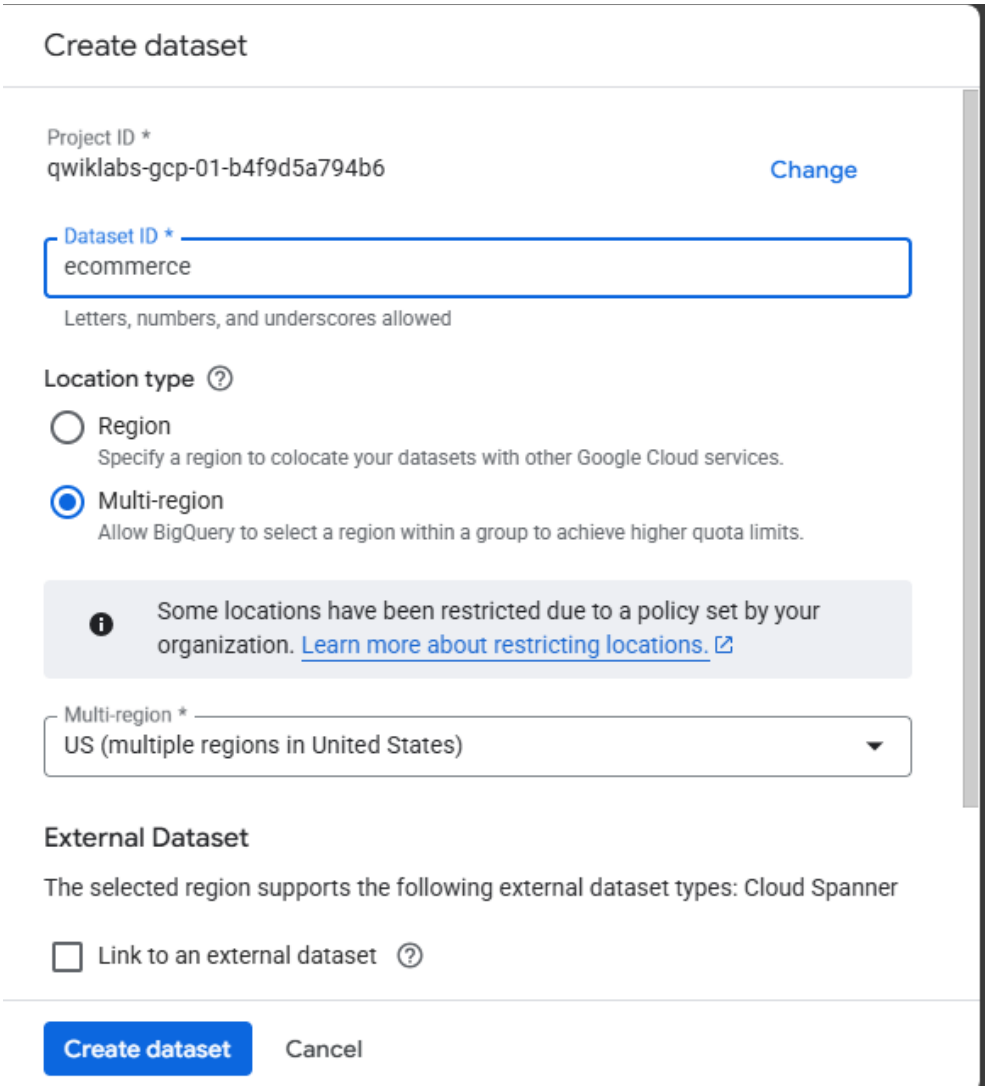
In your BigQuery project, create a new dataset titled ecommerce.

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.




## Task 2. Pin the lab project in BigQuery

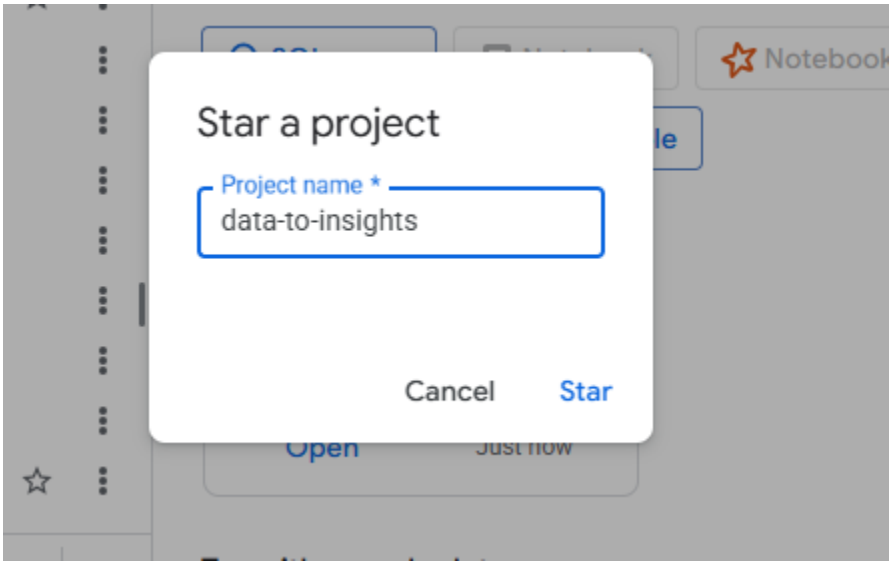
Scenario: Your team provides you with a new dataset on the inventory stock levels for each of your products for sale on your ecommerce website. You want to become

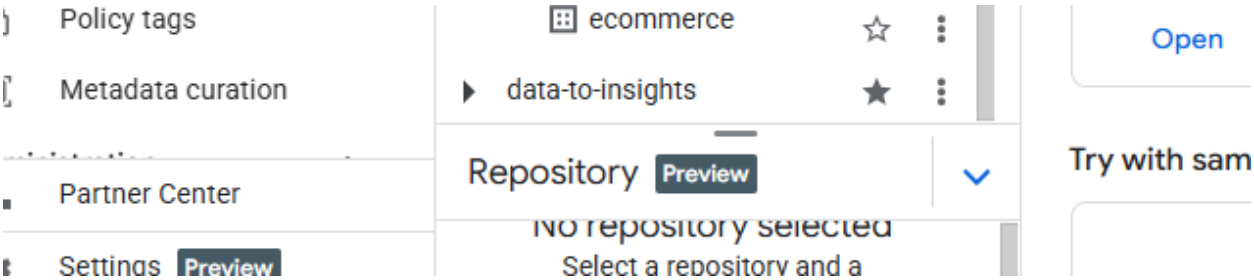
familiar with the products on the website and the fields you could use to potentially join on to other datasets.

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** sectio




**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.

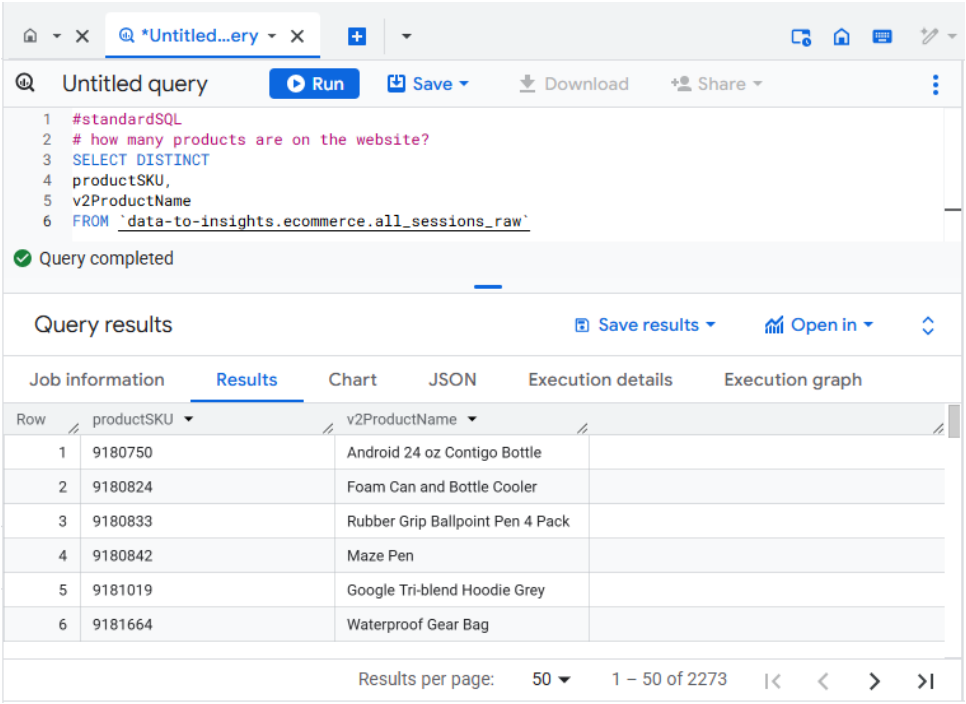
**Task 4. Identify a key field in your ecommerce dataset**

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.

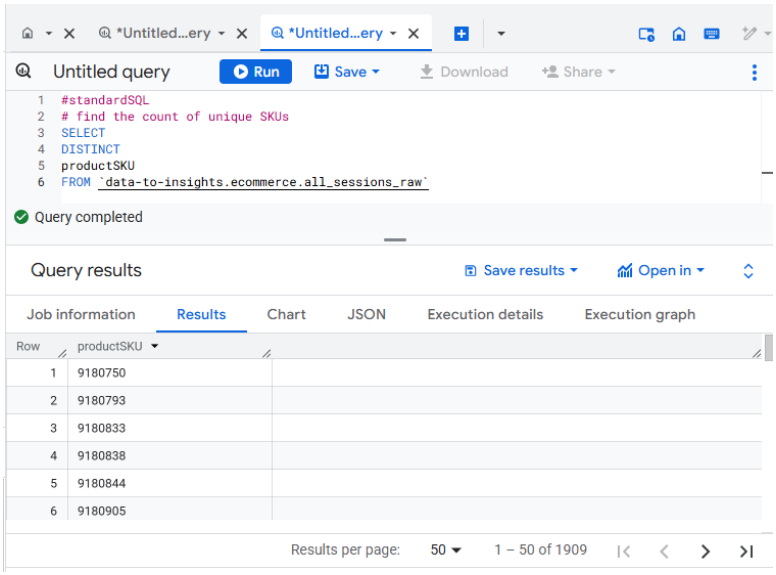
Examine the records

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**:



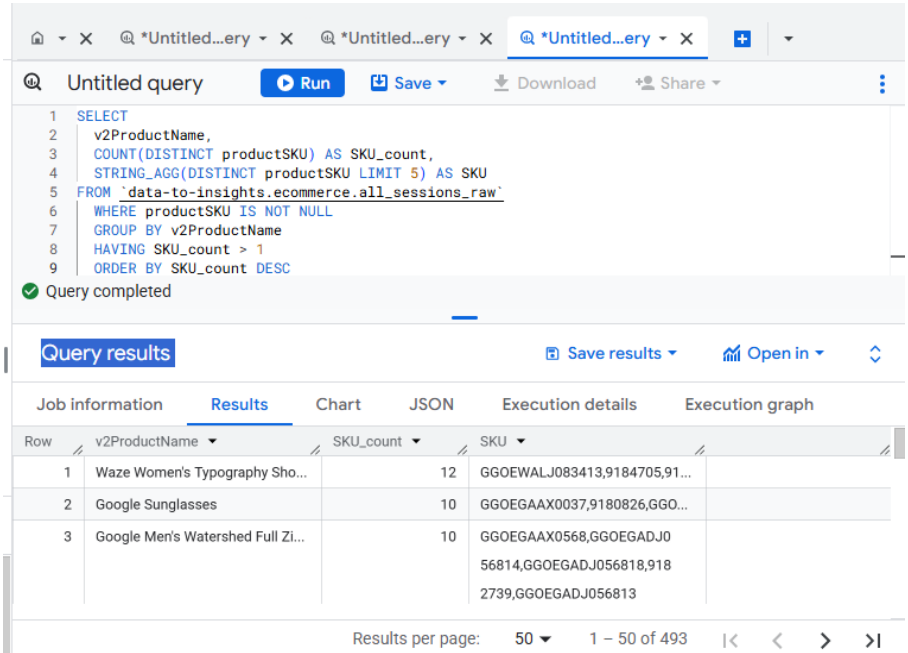

3. Clear the previous query and run the below query to list the number of distinct SKUs are listed using DISTINCT:




**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.

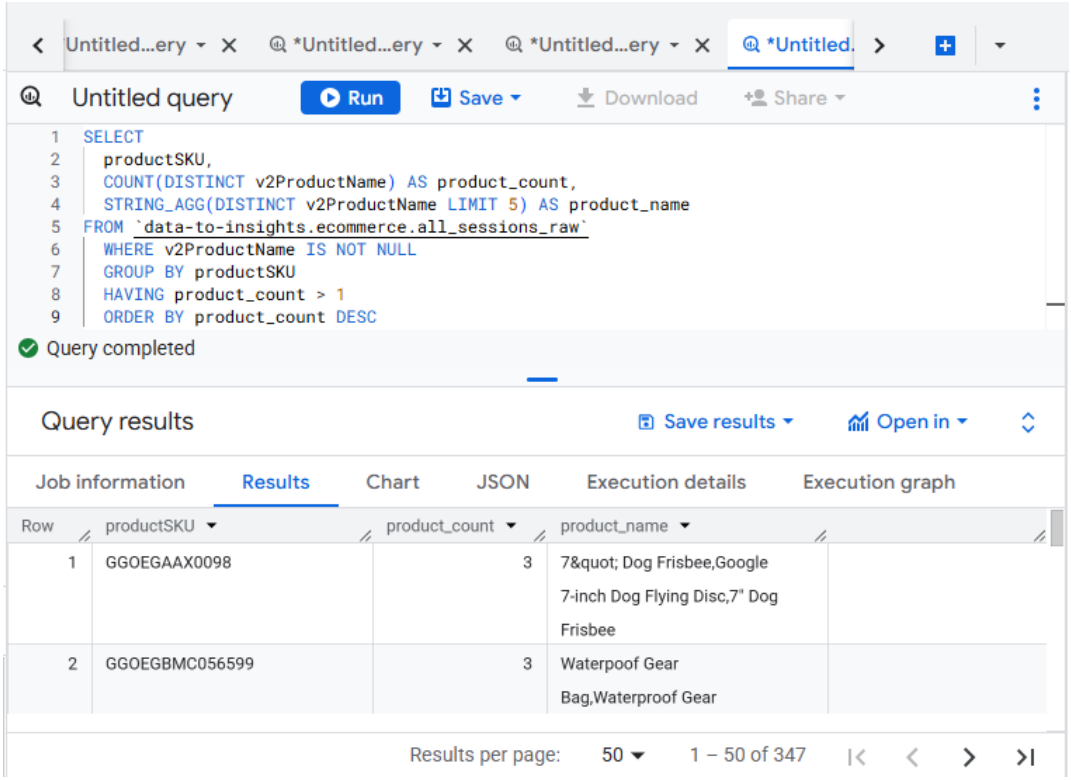



2. Click **Run**.

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?

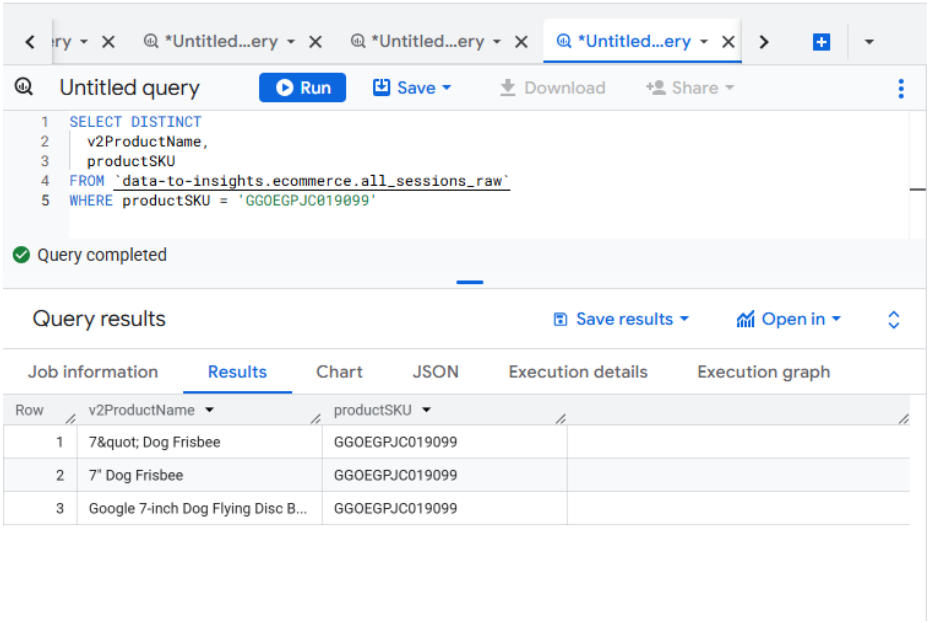
- Clear the previous query and run the below query to find out:




**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.

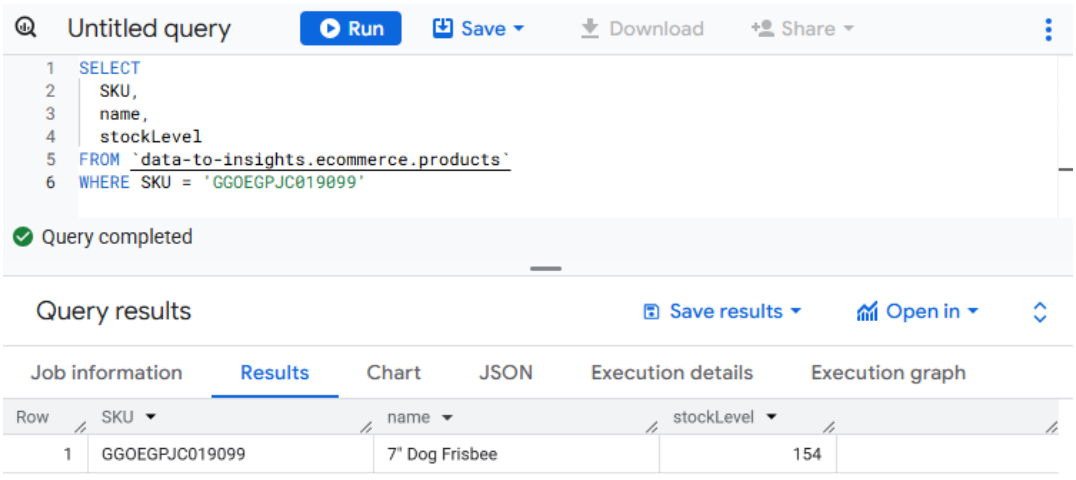
1. **Write a query** to identify all the product names for the SKU 'GGOEGPJC019099'.




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:

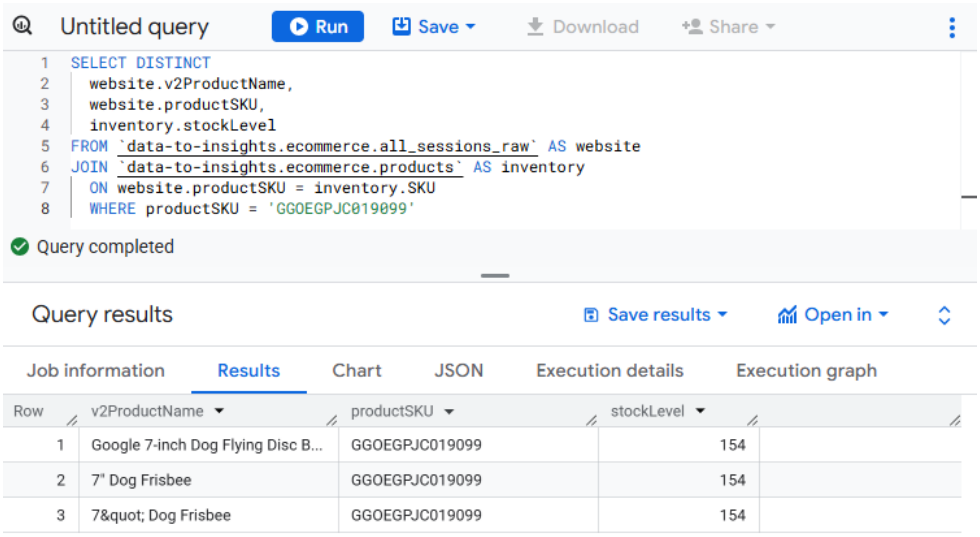



Join pitfall: Unintentional many-to-one SKU relationship

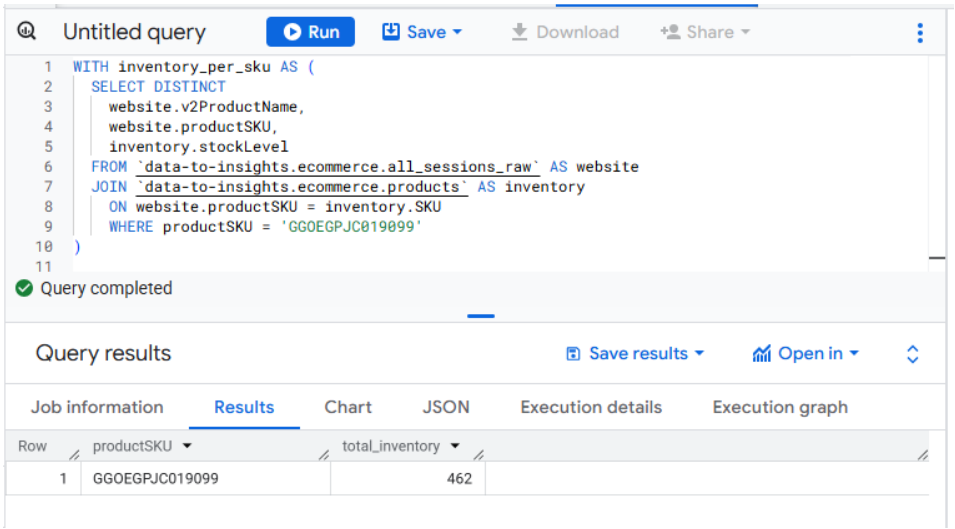
You now have two datasets: one for inventory stock level and the other for our website analytics. JOIN the inventory dataset against your website product names

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:




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




**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:

```
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
```

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 | GGOEGAAX0098 | 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

```
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
```

Query completed

Query results — Save results — Open in

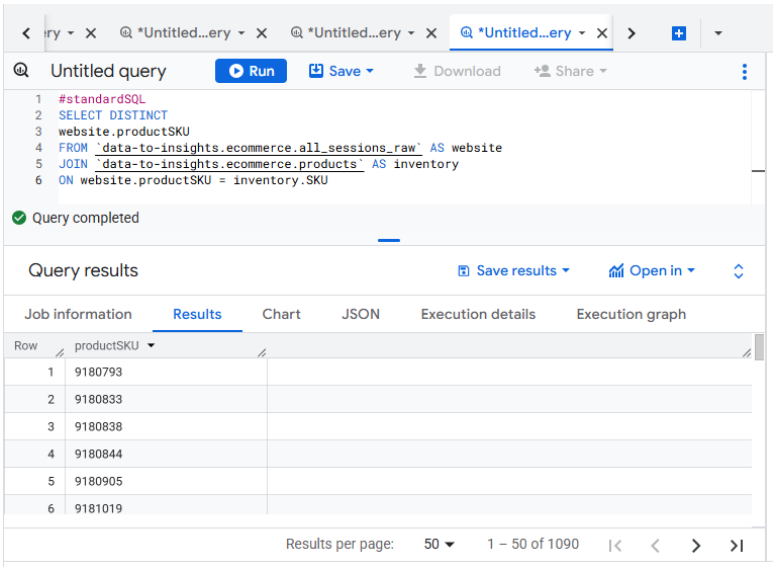
Job information | Results | Chart | JSON | Execution details | Execution graph

| Row | productSKU | push_all_names_into_array |
|---|---|---|
| 1 | GGOEGAAX0098 | 7" Dog Frisbee |

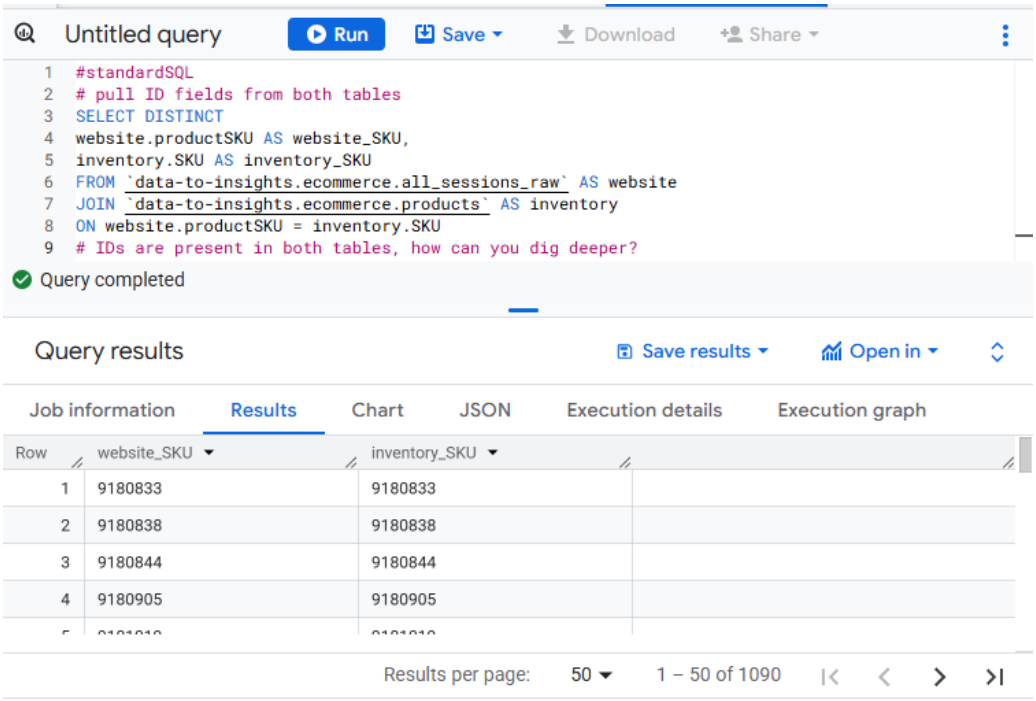
Join pitfall: losing data records after a join

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

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




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

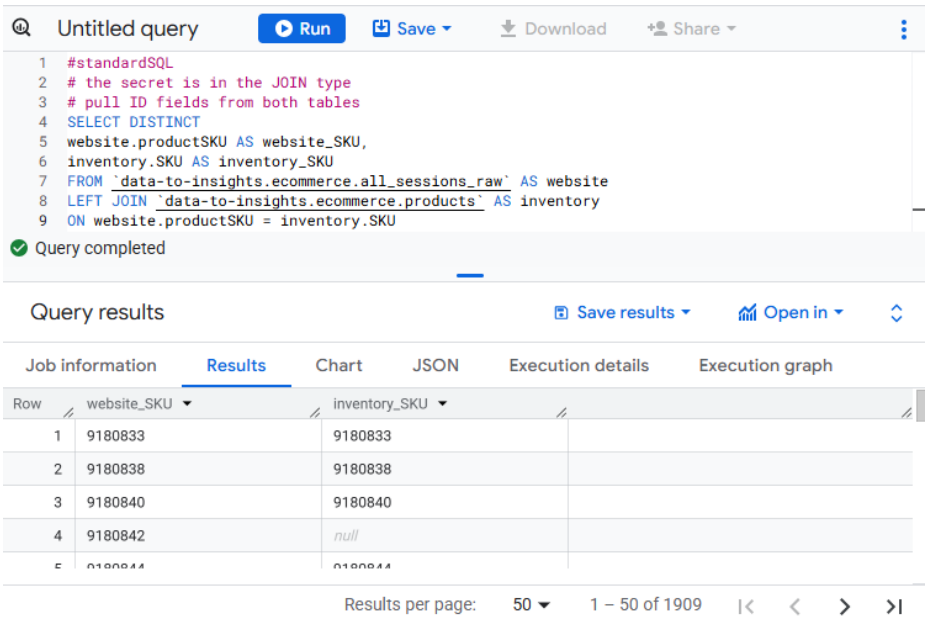



**Join pitfall solution: selecting the correct join type and filtering for NULL**

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

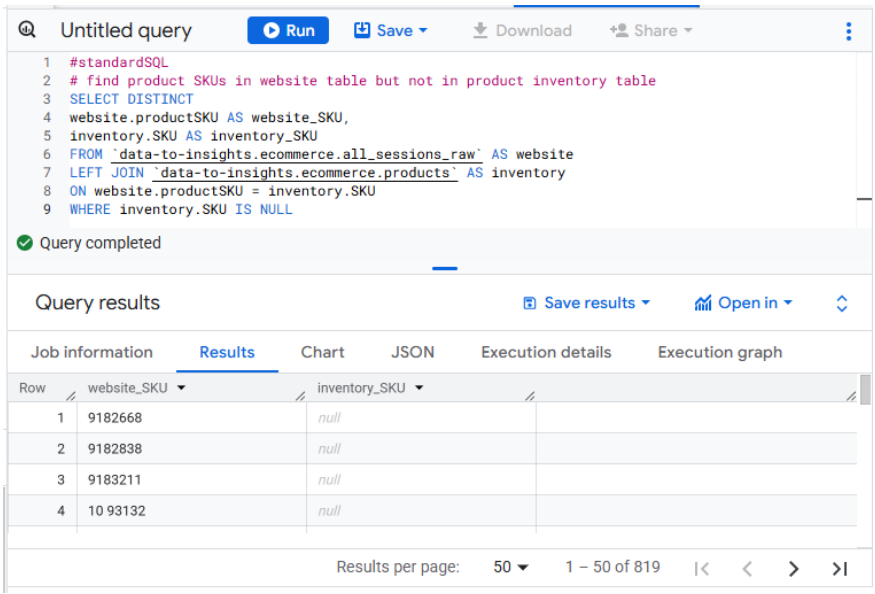
SKU record. Join type options: INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, CROSS JOIN.



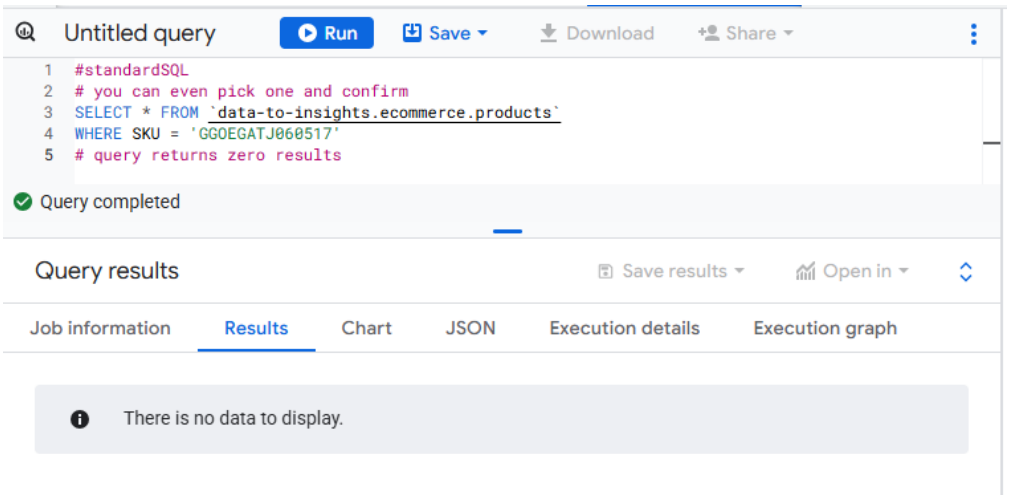

Click **Run**

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

   Possible solution:

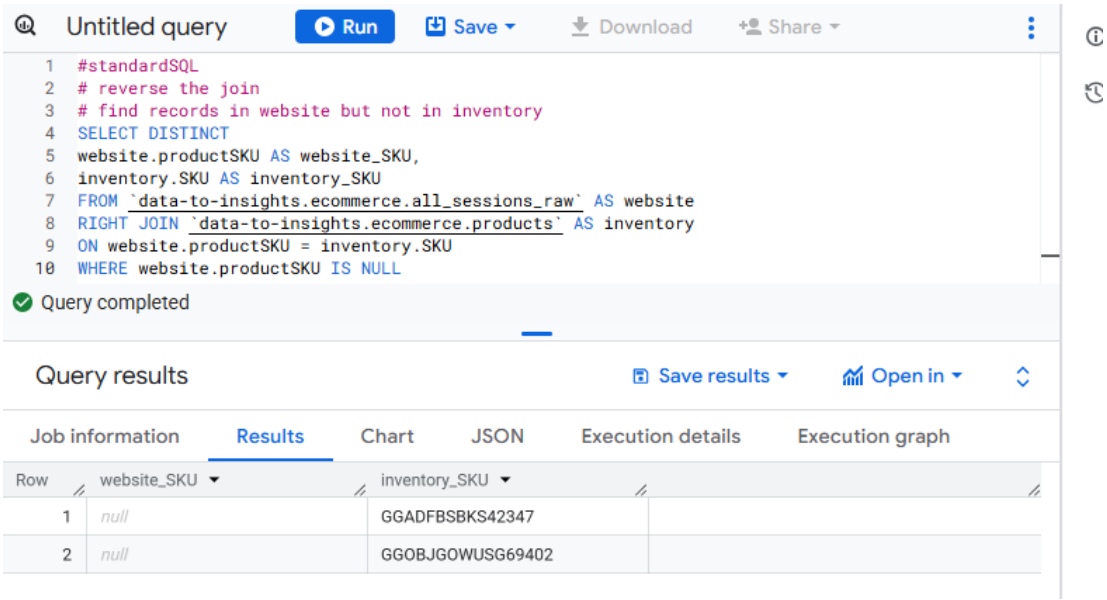



- Clear the previous query and run the below query to confirm using one of the specific SKUs from the website dataset:



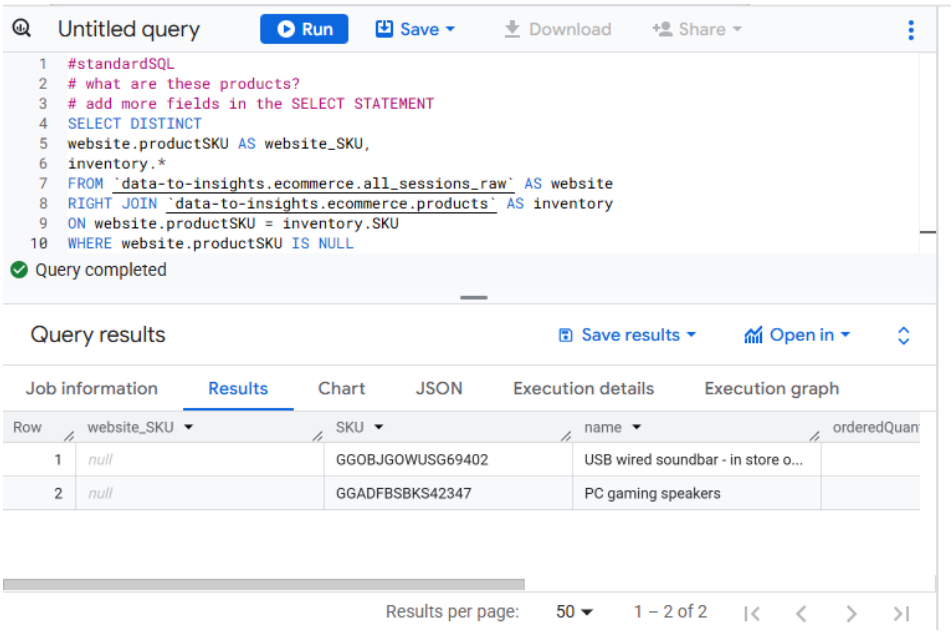

1. Write a query using a different join type to investigate.

   Possible solution:



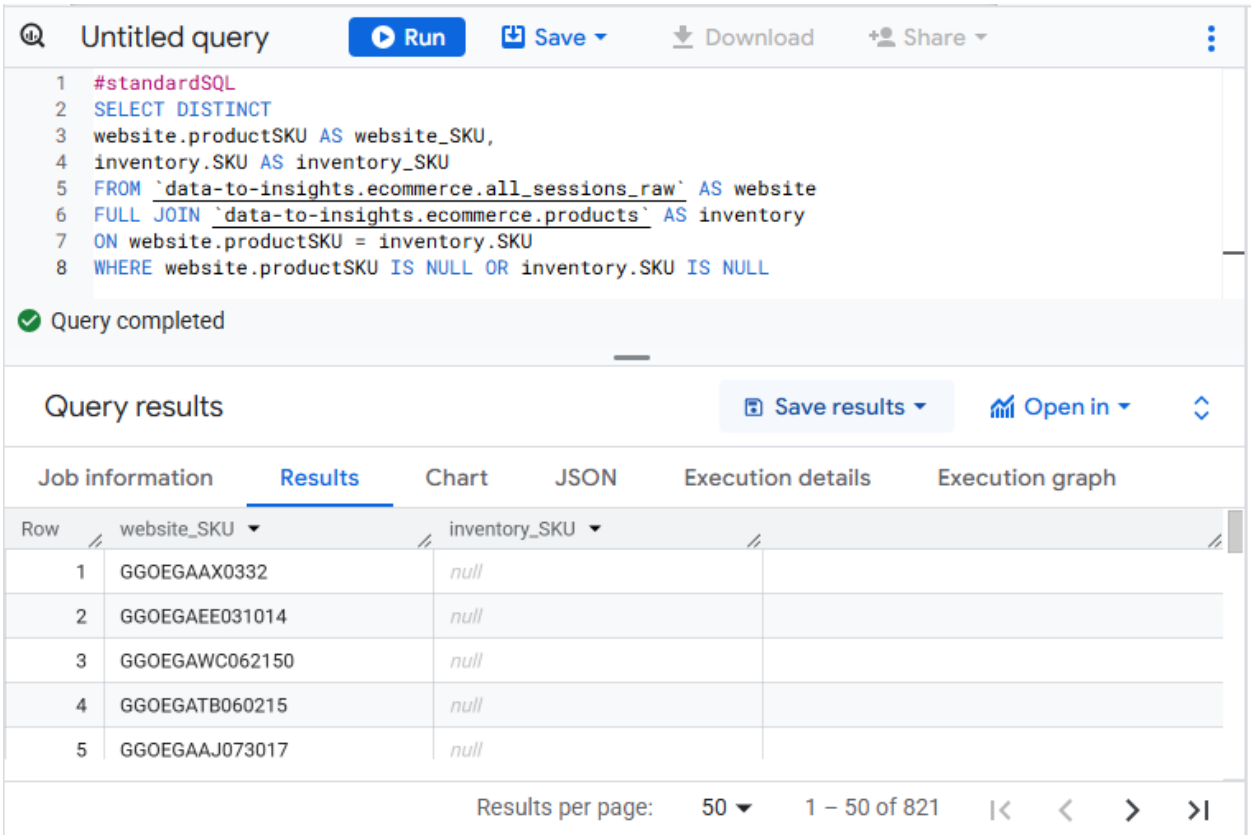

2. Click **Run**.

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




1. Write a query using a different join type.

   Possible solution:




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

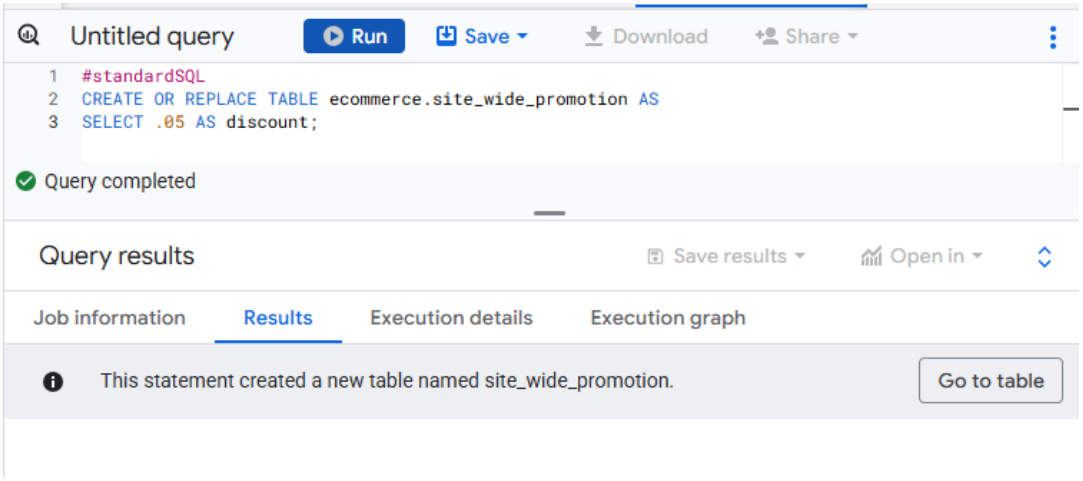
Join pitfall: unintentional cross join

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.

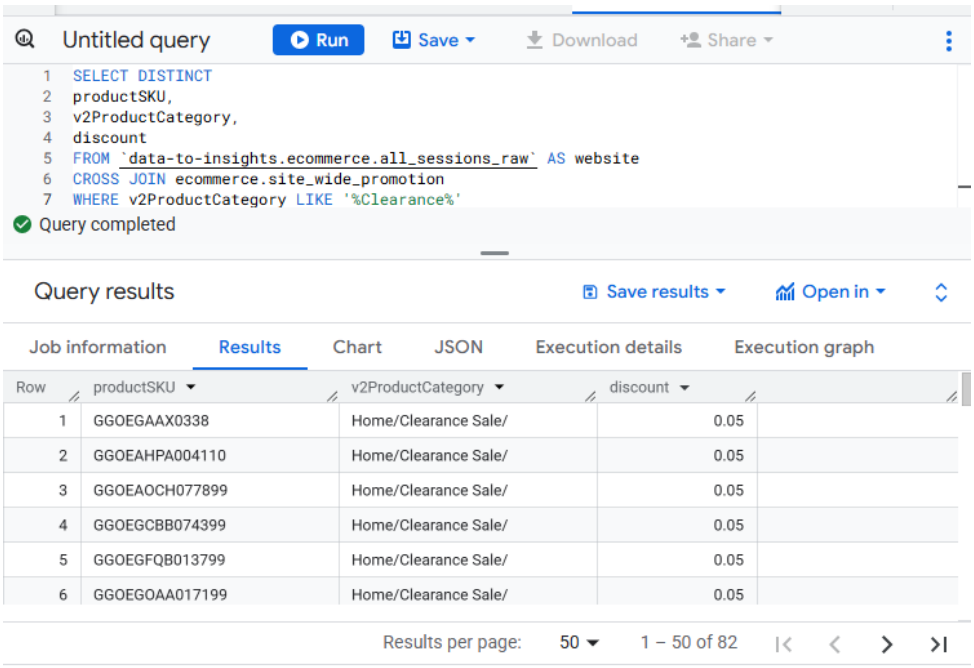
Create a new table with a site-wide discount percent that you want applied across products in the Clearance category.

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

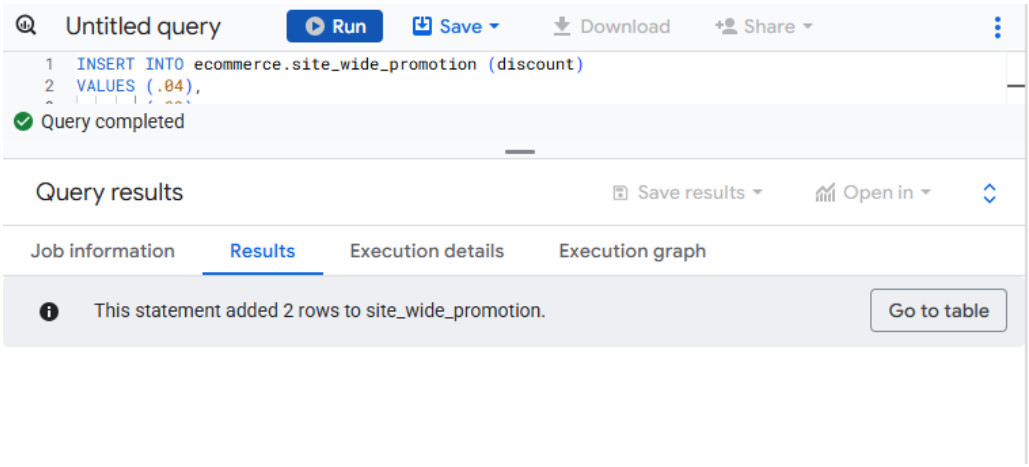



In the left pane, site_wide_promotion is now listed in the Resource section under your project and dataset.

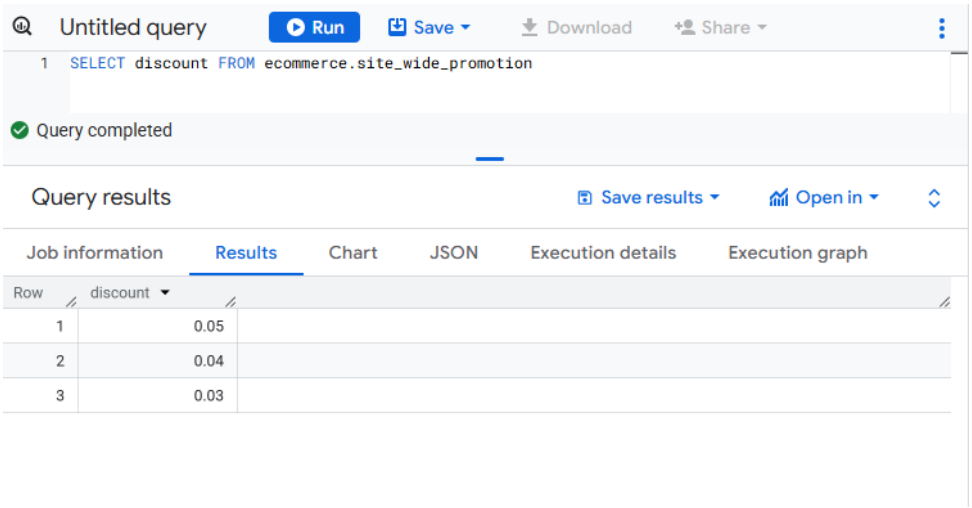
2. Clear the previous query and run the below query to find out how many products are in clearance:



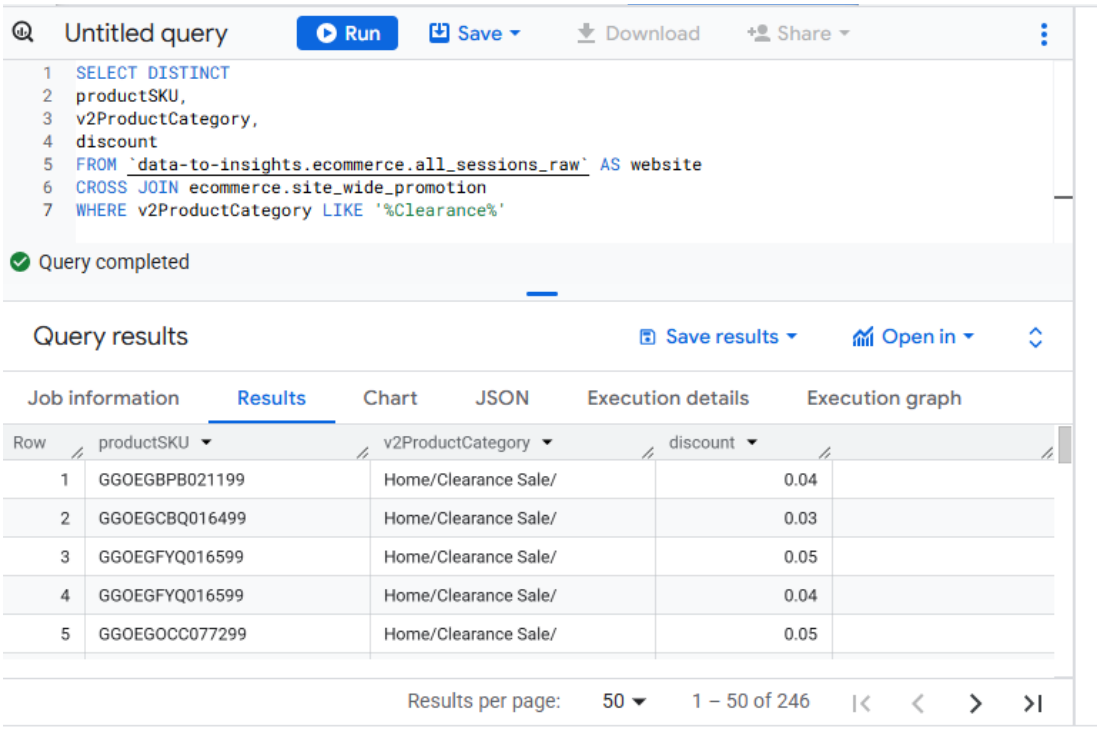

3. Clear the previous query and run the below query to insert two more records into the promotion table:




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

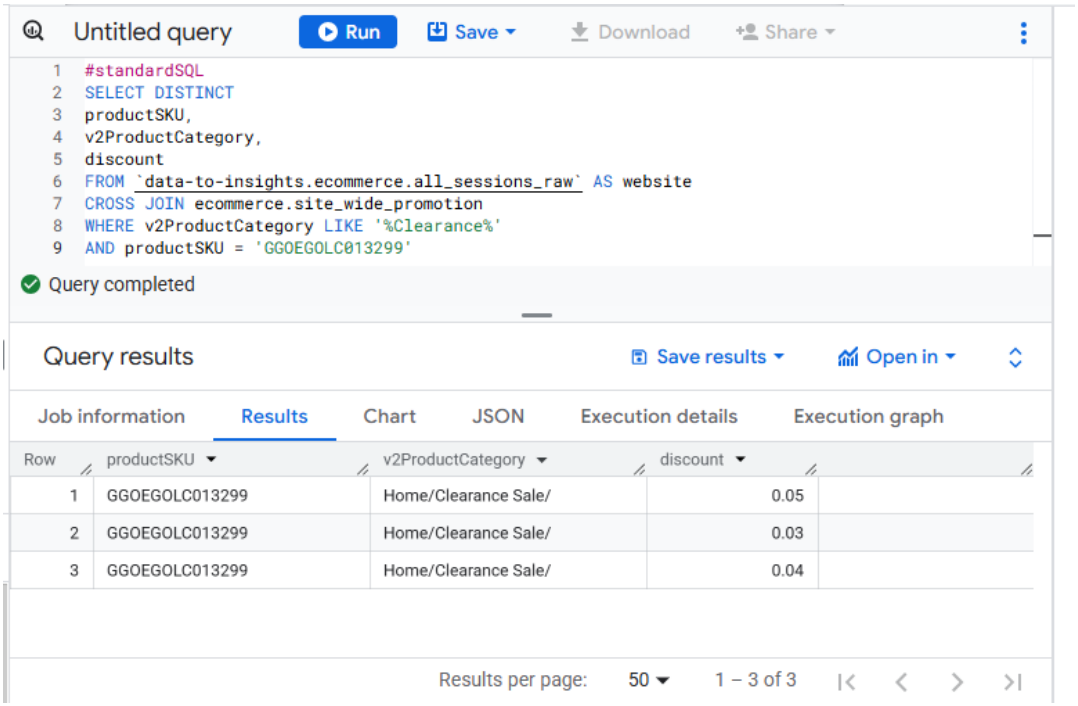



5. Clear the previous query and run the below query:




Now investigate the underlying cause by examining one product SKU.

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




## 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