

Troubleshooting and Solving Data Join Pitfalls

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.

Task 1. Create a new dataset to store your tables

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

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

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

2. Click **Done**.
3. BigQuery public datasets are not displayed by default. To open the public datasets project, copy **data-to-insights**.
4. Click **Add Data** > **Star a project by name** then paste the data-to-insights name.

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

Solution:

The screenshot shows the Google Cloud BigQuery console interface. On the left, the 'Explorer' pane displays a project named 'qwiklabs-gcp-02-a62711edbf1c' with a folder 'data-to-insights' containing several datasets, including 'all_sessions_raw'. The main pane shows the 'all_sessions_raw' table selected, with tabs for 'SCHEMA', 'DETAILS', 'PREVIEW', 'LINEAGE', and 'PREVIEW'. The 'SCHEMA' tab is active, displaying a table of fields with their names, types, and modes.

Field name	Type	Mode	Collation	Default Value	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				

At the bottom of the schema pane, there are buttons for 'EDIT SCHEMA' and 'VIEW ROW ACCESS POLICIES'. Below these are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY', and a 'REFRESH' button.

Task 3. 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**:

Solution:

The screenshot shows the Google Cloud BigQuery console. The Explorer on the left lists datasets under 'data-to-insights', with 'all_sessions_raw' selected. The query editor on the right contains the following SQL:

```
1 SELECT DISTINCT
2   productSKU,
3   v2ProductName
4 FROM `data-to-insights.ecommerce.all_sessions_raw`
```

The 'Query results' section shows a table with 4 rows and 2 columns: 'productSKU' and 'v2ProductName'.

Row	productSKU	v2ProductName
1	9180824	Foam Can and Bottle Cooler
2	9180833	Rubber Grip Ballpoint Pen 4 Pa...
3	9180842	Maze Pen
4	9181019	Google Triblend Hoodie Grey

The status bar at the bottom indicates the query will process 1,012.99 MB when run.

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

Solution:

The screenshot shows the Google Cloud BigQuery console with a new query in the editor:

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

The 'Query results' section shows a table with 4 rows and 1 column: 'productSKU'.

Row	productSKU
1	9180842
2	9182569
3	9182575
4	9182588

The status bar at the bottom indicates the query is completed.

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.

Solution:

The screenshot shows the Google Cloud BigQuery console. The left sidebar displays the project hierarchy for 'qwklabs-gcp-02-a62711edb1c', with 'ecommerce' and 'all_sessions_raw' selected. The main editor contains a SQL query that uses `STRING_AGG()` to aggregate SKUs by product name. The query results are displayed in a table with columns for product name, SKU count, and the aggregated SKUs.

```
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
```

Row	v2ProductName	SKU_count	SKU
1	Waze Women's Typography Sh...	12	9184705,9184708,GGOEXXXX...
2	Google Sunglasses	10	GGOEGAA0037,GGOEGHGRD...
3	Google Men's Watershed Full Zi...	10	GGOEGAA0568,GGOEGADJ056...

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:

Solution:

The screenshot shows the Google Cloud BigQuery console interface. On the left, the 'Explorer' pane displays a project named 'qwklabs-gcp-02-a62711edbf1c' with a folder 'data-to-insights' containing several datasets, including 'all_sessions_raw'. The main editor shows a SQL query:

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

Below the query editor, the 'Query results' section is visible, showing a table with columns 'productSKU', 'product_count', and 'product_name'. The results are as follows:

Row	productSKU	product_count	product_name
1	GGOEGAX0098	3	7" Dog Frisbee, Google 7-inch Dog Flying Disc, 7" Dog Frisbee
2	GGOEGBMC056599	3	Waterproof Gear Bag, Waterproof

The bottom of the console shows the Windows taskbar with the date 30/12/2022 and time 17:42.

Task 4. 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'

Solution:

The screenshot shows the Google Cloud BigQuery console interface. On the left, the 'Explorer' pane displays a project named 'qwklabs-gcp-02-a62711edbf1c' with a folder 'data-to-insights' containing several datasets, including 'all_sessions_raw' which is selected. The main editor shows a SQL query:

```
1 SELECT DISTINCT
2   v2ProductName,
3   productSKU
4 FROM `data-to-insights.ecommerce.all_sessions_raw`
5 WHERE productSKU = 'GG0EGPJC019099'
```

Below the query editor, the 'Query results' section is active, displaying a table with 3 rows and 2 columns: 'v2ProductName' and 'productSKU'. The results are:

Row	v2ProductName	productSKU
1	7" Dog Frisbee	GG0EGPJC019099
2	7" Dog Frisbee	GG0EGPJC019099
3	Google 7-inch Dog Flying Disc	GG0EGPJC019099

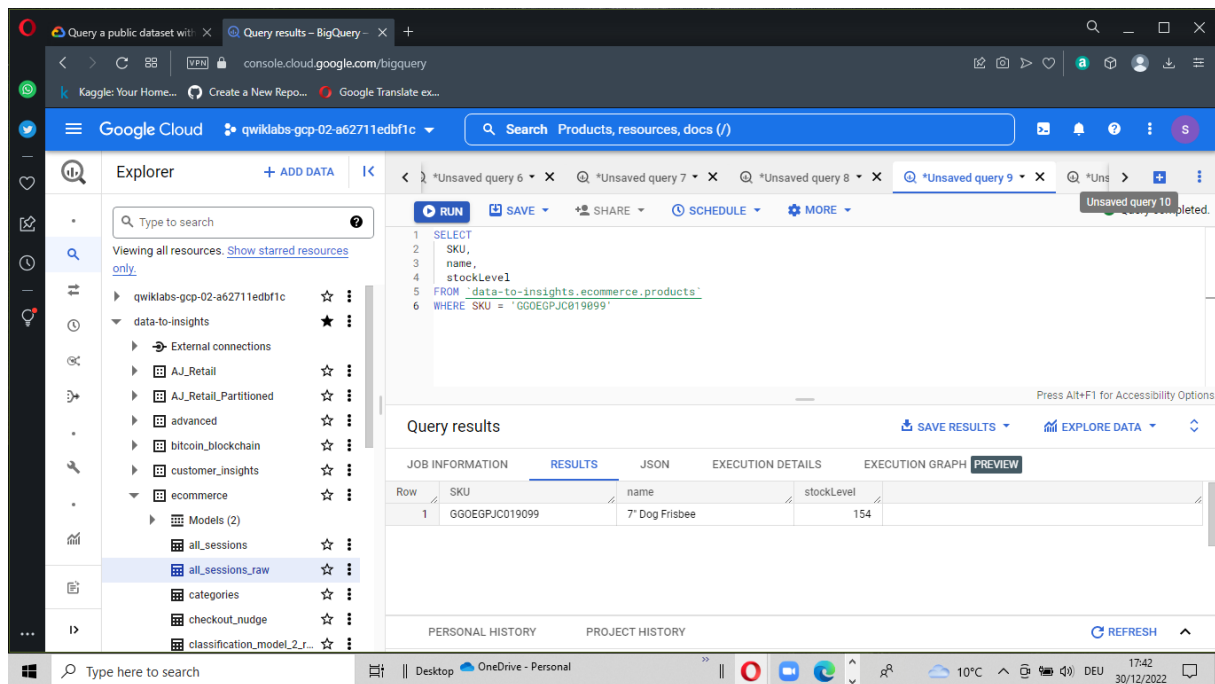
The bottom of the screen shows the Windows taskbar with the date 30/12/2022 and time 17:42.

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:

Solution:



The screenshot shows the Google Cloud BigQuery console interface. On the left, the 'Explorer' pane displays a list of datasets under the project 'qwiklabs-gcp-02-a62711edbf1c'. The 'data-to-insights' dataset is expanded, showing various tables including 'all_sessions_raw'. The main editor displays a SQL query:

```
1 SELECT
2   SKU,
3   name,
4   stockLevel
5 FROM `data-to-insights.ecommerce.products`
6 WHERE SKU = 'GG0EGPJC019099'
```

Below the query editor, the 'Query results' section shows a table with the following data:

Row	SKU	name	stockLevel
1	GG0EGPJC019099	7" Dog Frisbee	154

The bottom of the console shows the Windows taskbar with the date and time as 17:42 on 30/12/2022.

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:

Solution:

The screenshot shows the Google Cloud BigQuery console interface. The Explorer on the left lists datasets under the project 'qwklabs-gcp-02-a62711edbf1c'. The 'data-to-insights' dataset is expanded, showing tables like 'all_sessions_raw'. The main editor displays a SQL 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 = 'GG0EGPJC019099'
```

The 'Query results' section shows a table with 3 rows and 4 columns: 'v2ProductName', 'productSKU', 'stockLevel', and an unnamed column. The results are:

Row	v2ProductName	productSKU	stockLevel
1	7" Dog Frisbee	GG0EGPJC019099	154
2	7" Dog Frisbee	GG0EGPJC019099	154
3	Google 7-inch Dog Flying Disc	GG0EGPJC019099	154

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

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

Solution:

The screenshot shows the Google Cloud BigQuery console interface with a new query entered:

```
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 = 'GG0EGPJC019099'
10 )
11 SELECT
```

The 'Query results' section shows a table with 1 row and 2 columns: 'productSKU' and 'total_inventory'. The result is:

Row	productSKU	total_inventory
1	GG0EGPJC019099	462

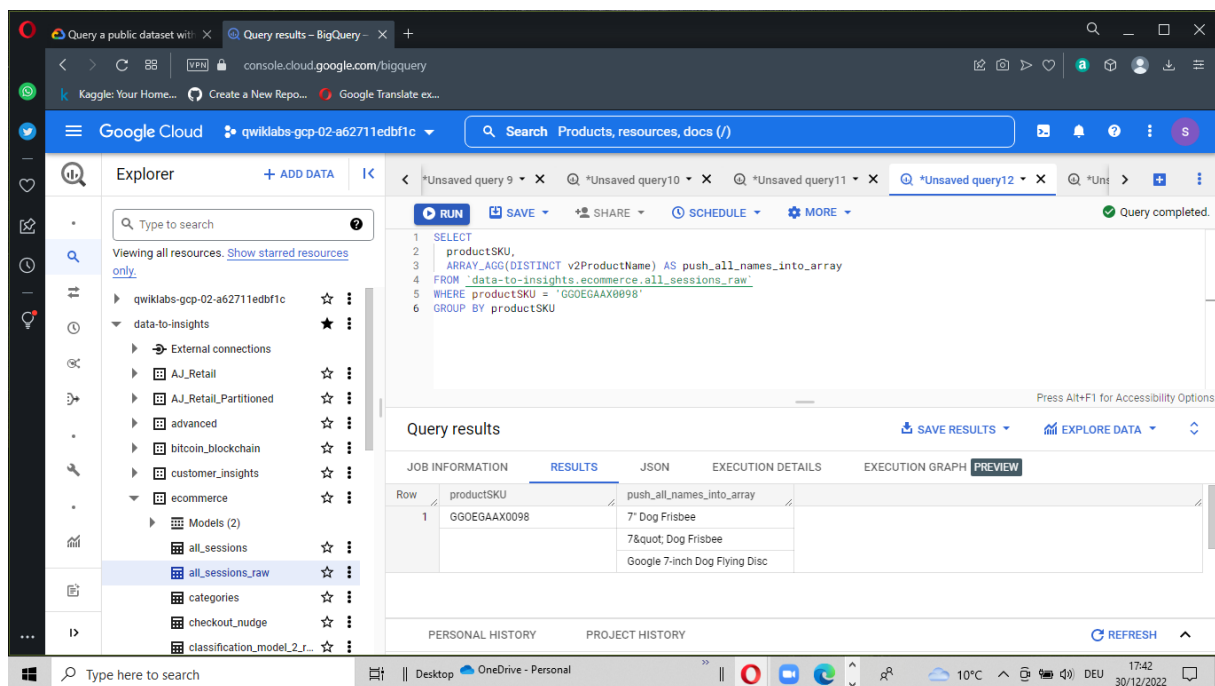
Task 5. Join pitfall solution: use distinct SKUs before joining

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:

Solution:



The screenshot shows the Google Cloud BigQuery console. The query editor contains the following SQL:

```
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 = 'GGOEGAAX0098'
6 GROUP BY productSKU
```

The query results are displayed in a table with the following data:

Row	productSKU	push_all_names_into_array
1	GGOEGAAX0098	7" Dog Frisbee 7" Dog Frisbee Google 7-inch Dog Flying Disc

Now instead of having a row for every Product Name, you only have a row for each unique SKU.

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:

It seems 819 SKUs were lost after joining the datasets Investigate by adding more specificity in your fields (one SKU column from each dataset):

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

Solution:

The screenshot shows the Google Cloud BigQuery console interface. On the left, the 'Explorer' pane displays a project named 'qwiklabs-gcp-02-a62711edbf1c' with a folder 'data-to-insights' containing several datasets, including 'all_sessions_raw'. The main pane shows a SQL query with five lines: a SELECT DISTINCT statement, a FROM clause for 'website.productSKU', a JOIN clause for 'data-to-insights.ecommerce.all_sessions_raw' AS website, a JOIN clause for 'data-to-insights.ecommerce.products' AS inventory, and an ON clause for 'website.productSKU = inventory.SKU'. Below the query, the 'Query results' section is active, displaying a table with four rows of productSKUs: 9180793, 9180833, 9180838, and 9180844. The bottom of the interface shows a Windows taskbar with the date 30/12/2022 and time 17:42.

It appears the SKUs are present in both of those datasets after the join for these 1,090 records. How can you 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.

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.

Solution:

The screenshot shows the Google Cloud BigQuery console interface. The Explorer panel on the left displays the project hierarchy, with 'all_sessions_raw' selected under the 'data-to-insights' dataset. The SQL editor in the center contains the following query:

```
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
```

The Query results panel at the bottom shows a table with two columns: 'website_SKU' and 'inventory_SKU'. The results are as follows:

Row	website_SKU	inventory_SKU
1	9180793	9180793
2	9180824	9180824
3	9181019	9181019
4	9182502	9182502

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

Solution:

The screenshot shows the Google Cloud BigQuery console interface. The Explorer panel on the left displays the project hierarchy, with 'all_sessions_raw' selected under the 'data-to-insights' dataset. The SQL editor in the center contains the following query:

```
1 # find product SKUs in website table but not in product inventory table
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 LEFT JOIN `data-to-insights.ecommerce.products` AS inventory
7 ON website.productSKU = inventory.SKU
8 WHERE inventory.SKU IS NULL
```

The Query results panel at the bottom shows a table with two columns: 'website_SKU' and 'inventory_SKU'. The results are as follows:

Row	website_SKU	inventory_SKU
32	GG0EG0AC016199	null
33	GG0EG0AR021999	null
34	GG0EG0GA016299	null
35	GG0EG0AX0447	null

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

Solution:

The screenshot shows the Google Cloud BigQuery console. The Explorer on the left lists datasets under 'data-to-insights', including 'ecommerce'. The query editor on the right contains the following SQL:

```
1 SELECT * FROM "data-to-insights.ecommerce.products"
2 WHERE SKU = 'GG0EGATJ060517'
3
```

The 'Query results' section shows a message: 'There is no data to display.' The interface includes tabs for JOB INFORMATION, RESULTS, JSON, EXECUTION DETAILS, and EXECUTION GRAPH. The bottom status bar shows the time as 17:43 on 30/12/2022.

Write a query using a different join type to investigate.

Solution:

The screenshot shows the Google Cloud BigQuery console with a new query in the editor:

```
1 # reverse the join
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 RIGHT JOIN "data-to-insights.ecommerce.products" AS inventory
7 ON website.productSKU = inventory.SKU
8 WHERE website.productSKU IS NULL
```

The 'Query results' section displays a table with two rows of data:

Row	website_SKU	inventory_SKU
1	null	GGADFB8SK542347
2	null	GG0BJGOWUSG69402

The interface includes tabs for JOB INFORMATION, RESULTS, JSON, EXECUTION DETAILS, and EXECUTION GRAPH. The bottom status bar shows the time as 17:43 on 30/12/2022.

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:

Solution:

The screenshot shows the Google Cloud BigQuery console. The query editor contains the following SQL code:

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

The query results are displayed in a table with the following columns: website_SKU, SKU, name, orderedQuantity, stockLevel, and restockin. The results show two rows of data:

Row	website_SKU	SKU	name	orderedQuantity	stockLevel	restockin
1	null	GG0BJGOWUG69402	USB wired soundbar - in store o...	10	15	
2	null	GGADFB8BKS42347	PC gaming speakers	0	100	

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

1. Write a query using a different join type.

Solution:

The screenshot shows the Google Cloud BigQuery console. The query editor contains the following SQL code:

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

The query results are displayed in a table with the following columns: website_SKU and inventory_SKU. The results show four rows of data:

Row	website_SKU	inventory_SKU
1	GGOEGAA0332	null
2	GGOEGAE031014	null
3	GGOEGAWC062150	null
4	GGOEGATB060215	null

Results per page: 50 1 - 50 of 821

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

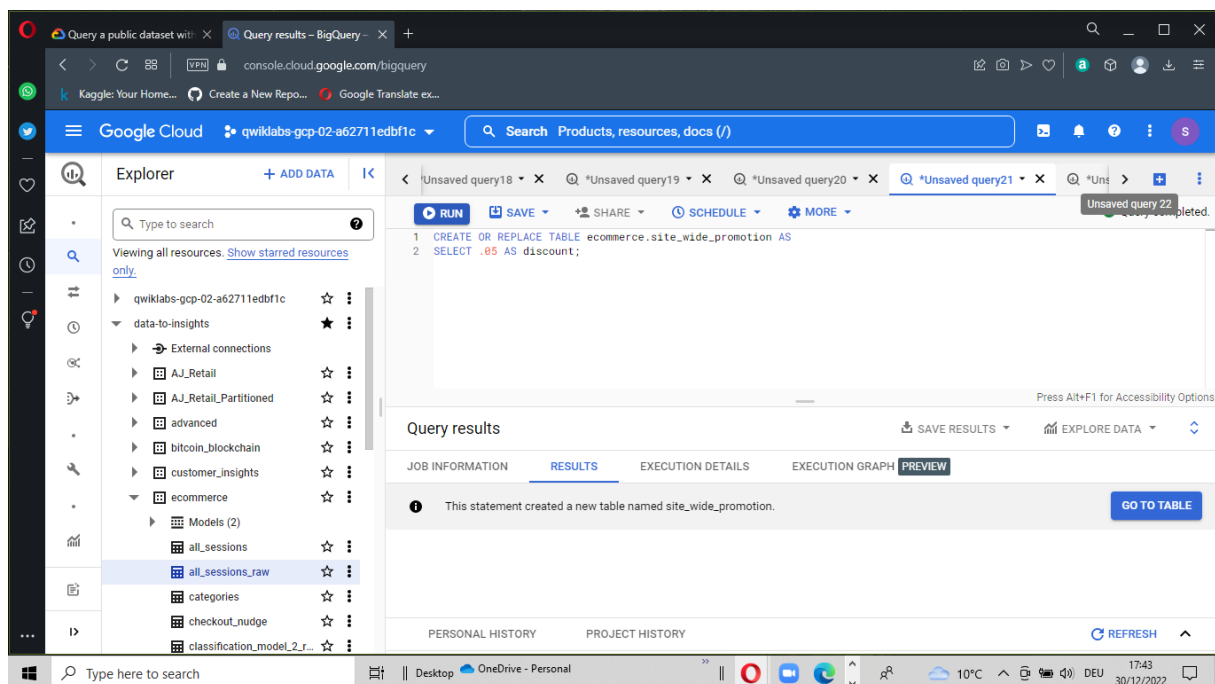
Not knowing the relationship between data table keys (1:1, 1:N, N:N) can return unexpected results and also significantly reduce query performance.

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:

Solution:



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:

Solution:

The screenshot shows the Google Cloud BigQuery console interface. On the left is the Explorer pane with a tree view of resources. The main area displays a SQL query and its results.

SQL Query:

```
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 results table:

Row	productSKU	v2ProductCategory	discount
27	GGGEGAAAX0363	Home/Clearance Sale/	0.05
28	GGGEGAAAX0213	Home/Clearance Sale/	0.05
29	GGGEGAAAX0568	Home/Clearance Sale/	0.05

The interface also shows tabs for 'JOB INFORMATION', 'RESULTS', 'JSON', 'EXECUTION DETAILS', and 'EXECUTION GRAPH'. The 'RESULTS' tab is active, displaying the table above. At the bottom, there are sections for 'PERSONAL HISTORY' and 'PROJECT HISTORY'.

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.

See the impact of unintentionally adding more than one record in the discount table.

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

Solution:

The screenshot shows the Google Cloud BigQuery console interface. The Explorer panel on the left lists the project resources, including the 'ecommerce' dataset. The main panel displays a query editor with the following SQL code:

```
1 INSERT INTO ecommerce.site_wide_promotion (discount)
2 VALUES (.84),(.83);
```

The query results section shows a message: "Billing has not been enabled for this project. Enable billing at <https://console.cloud.google.com/billing>. DML queries are not allowed in the free tier. Set up a billing account to remove this restriction."

Next, view the data values in the promotion table.

4. Clear the previous query and run the below query

Solution:

The screenshot shows the Google Cloud BigQuery console interface. The Explorer panel on the left lists the project resources, including the 'ecommerce' dataset. The main panel displays a query editor with the following SQL code:

```
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%'
```

The query results section shows a table with 4 rows and 3 columns: productSKU, v2ProductCategory, and discount.

Row	productSKU	v2ProductCategory	discount
1	GGOEGFYQ016599	Home/Clearance Sale/	0.05
2	GGOEGOAB016099	Home/Clearance Sale/	0.05
3	GGOEGOCC077299	Home/Clearance Sale/	0.05
4	GGOEGOCR078499	Home/Clearance Sale/	0.05

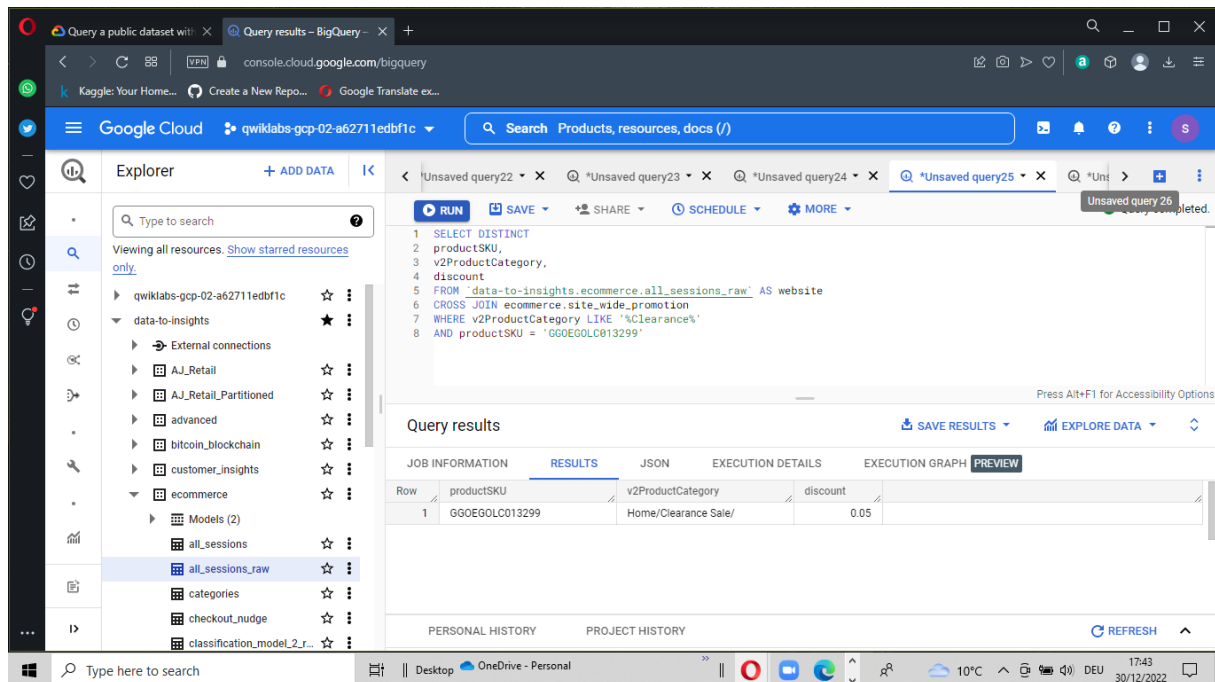
How many products are returned?

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

Now investigate the underlying cause by examining one product SKU.

5. Clear the previous query and run the below query

Solution:



The screenshot shows the Google Cloud BigQuery console interface. On the left, the 'Explorer' pane displays a project named 'qwiklabs-gcp-02-a62711edb1c' with a folder 'data-to-insights' containing several datasets. The 'all_sessions_raw' dataset is selected. The main pane shows a SQL query:

```
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%'
8 AND productSKU = 'GG0EGOLC013299'
```

Below the query, the 'Query results' section shows a table with the following data:

Row	productSKU	v2ProductCategory	discount
1	GG0EGOLC013299	Home/Clearance Sale/	0.05

The bottom of the screen shows the Windows taskbar with the date and time as 17:43 on 30/12/2022.