<u>The dataset you'll use is an ecommerce dataset that has millions of Google Analytics records from the Google Merchandise Store. You will explore the available fields and row for insights. This lab focuses on how to create new reporting tables using SQL JOINS and UNIONs.</u>

Task 1. The BigQuery console 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.

Task 2. Create a new dataset to store your tables

First, create a new dataset titled **ecommerce** in BigQuery.

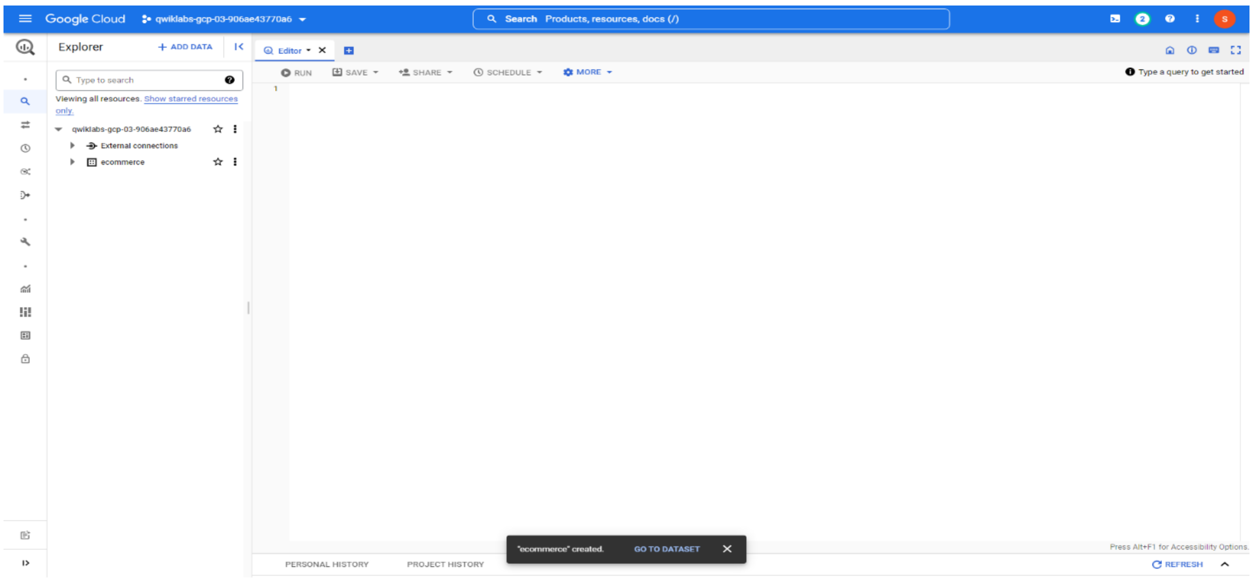
1. In the left pane, click on the name of your BigQuery project (qwiklabs-gcp-xxxx).

2. Click on the three dots next to your project name, then select **CREATE DATASET**.

The **Create dataset** dialog opens.

3. Set the *Dataset ID* to ecommerce, leave all other options at their default values.Click **Create dataset**.

Solution:

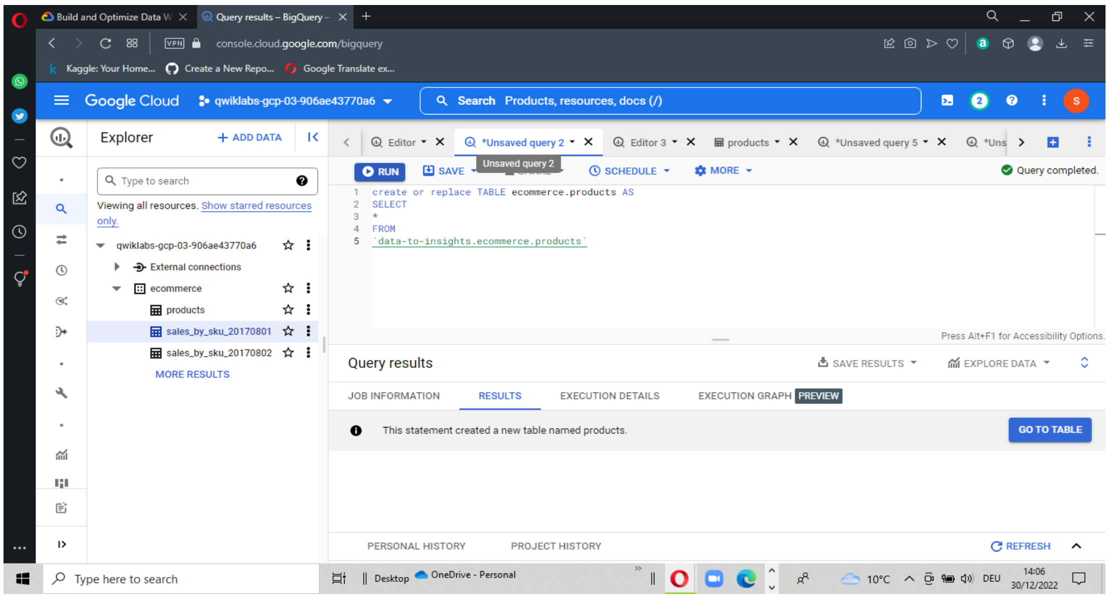

Google Cloud
qwiklabs-gcp-03-906ae43770a6
Search Products, resources, docs (/)
Explorer
+ ADD DATA
Editor
RUN
SAVE
SHARE
SCHEDULE
MORE
Type a query to get started
Type to search
Viewing all resources. Show starred resources only.
qwiklabs-gcp-03-906ae43770a6
External connections
ecommerce
"ecommerce" created.
GO TO DATASET
PERSONAL HISTORY
PROJECT HISTORY
REFRESH
Press Alt+F1 for Accessibility Options.

Task 3. Explore the product sentiment dataset

Your data science team has run all of your product reviews through the API and provided you with the average sentiment score and magnitude for each of your products.

1. First, create a copy the table that the data science team made so you can read it:

Solution:



```
create or replace TABLE ecommerce.products AS
SELECT
*
FROM
`data-to-insights.ecommerce.products`
```

Query completed.

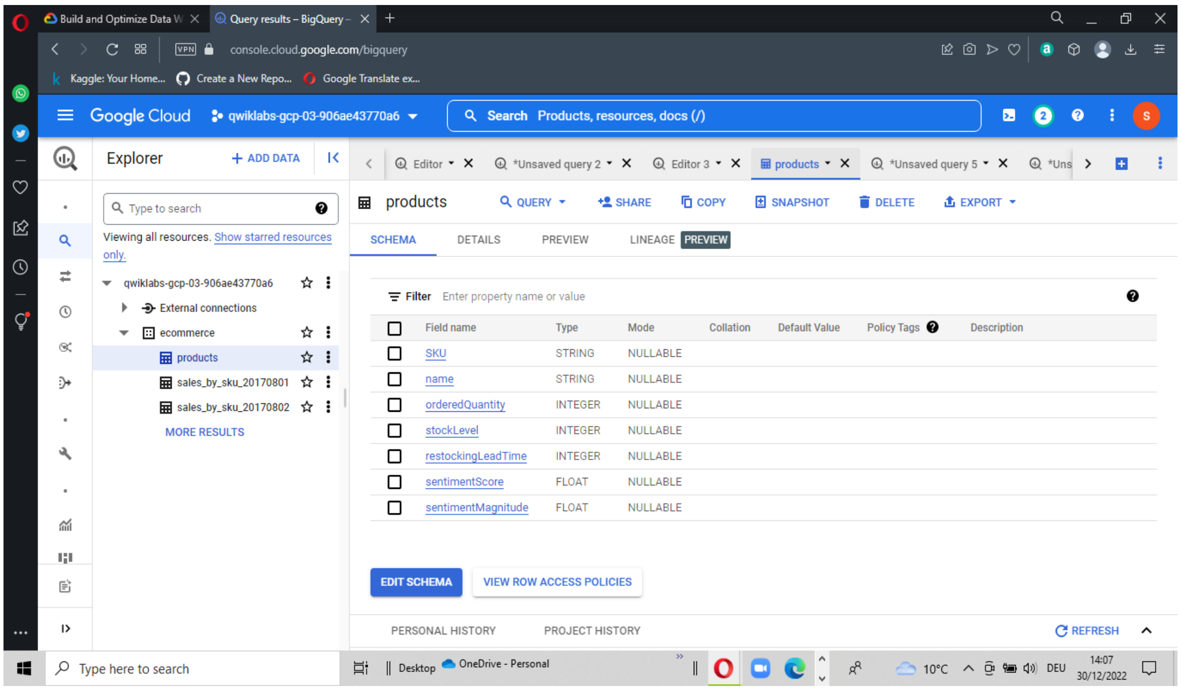
Query results

This statement created a new table named products.

Task 4. Examine the data

1. Navigate to the **ecommerce** > **products** dataset and click the **Schema** tab to see the data. What data type are the sentimentScore and sentimentMagnitude fields?

Solution:

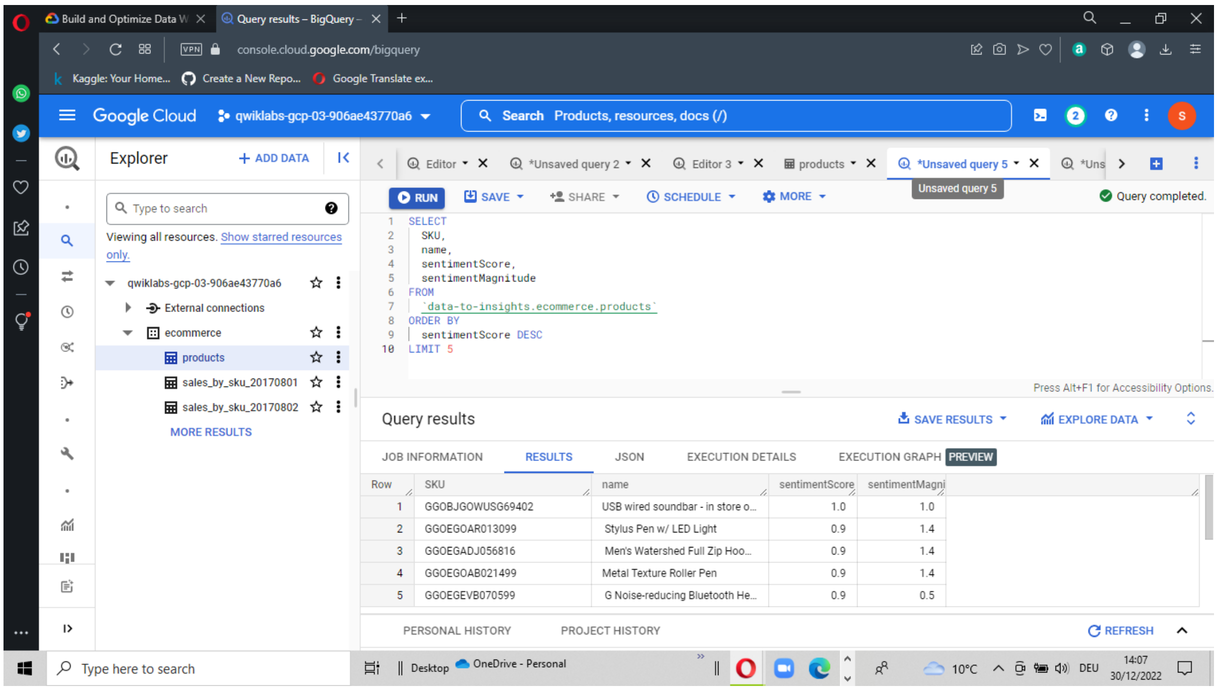


Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
SKU	STRING	NULLABLE				
name	STRING	NULLABLE				
orderedQuantity	INTEGER	NULLABLE				
stockLevel	INTEGER	NULLABLE				
restockingLeadTime	INTEGER	NULLABLE				
sentimentScore	FLOAT	NULLABLE				
sentimentMagnitude	FLOAT	NULLABLE				

Create a query that shows the top 5 products with the most positive sentiment

1. In the **Query Editor**, write your SQL query.

Solution:



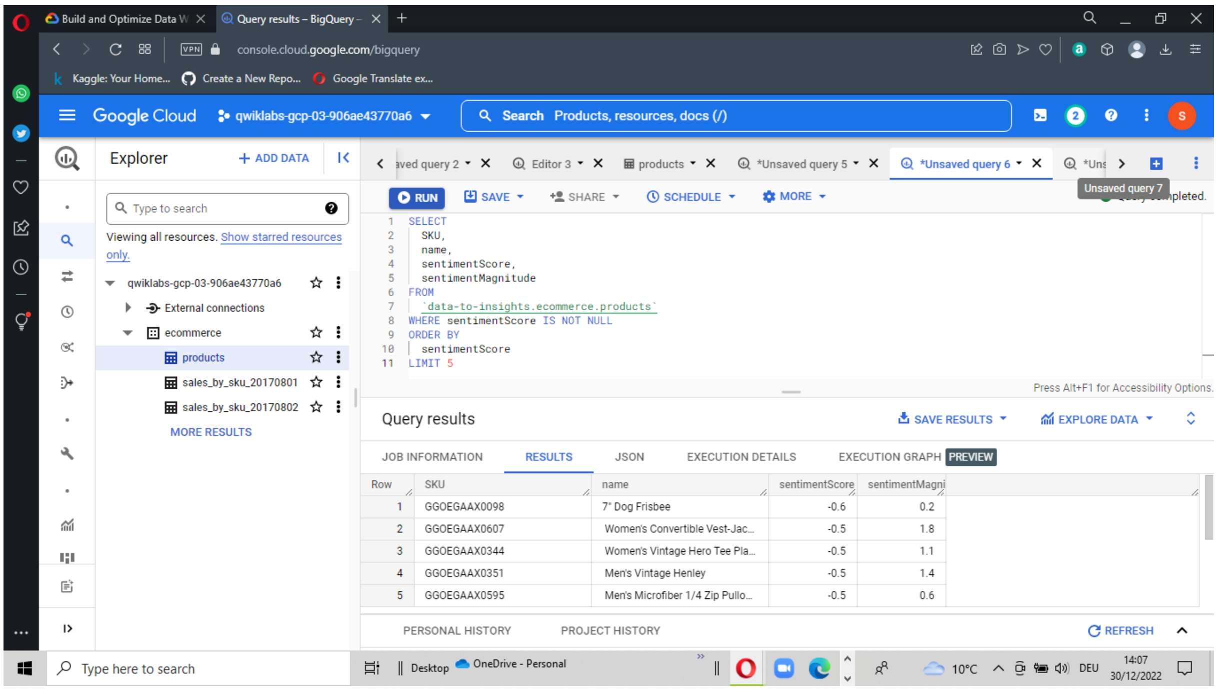
```
SELECT
  SKU,
  name,
  sentimentScore,
  sentimentMagnitude
FROM
  `data-to-insights.ecommerce.products`
ORDER BY
  sentimentScore DESC
LIMIT 5
```

Query results

Row	SKU	name	sentimentScore	sentimentMagni
1	GGOBJGOWUSG69402	USB wired soundbar - in store o...	1.0	1.0
2	GGOEGOAR013099	Stylus Pen w/ LED Light	0.9	1.4
3	GGOEGADJ056816	Men's Watershed Full Zip Hoo...	0.9	1.4
4	GGOEGOAB021499	Metal Texture Roller Pen	0.9	1.4
5	GGOEGEVB070599	G Noise-reducing Bluetooth He...	0.9	0.5

2. Revise your query to show the top 5 products with the most negative sentiment and filter out NULL values.

Solution:



```
SELECT
  SKU,
  name,
  sentimentScore,
  sentimentMagnitude
FROM
  `data-to-insights.ecommerce.products`
WHERE sentimentScore IS NOT NULL
ORDER BY
  sentimentScore
LIMIT 5
```

Query results

Row	SKU	name	sentimentScore	sentimentMagni
1	GGOEGAAX0098	7" Dog Frisbee	-0.6	0.2
2	GGOEGAAX0607	Women's Convertible Vest-Jac...	-0.5	1.8
3	GGOEGAAX0344	Women's Vintage Hero Tee Pla...	-0.5	1.1
4	GGOEGAAX0351	Men's Vintage Henley	-0.5	1.4
5	GGOEGAAX0595	Men's Microfiber 1/4 Zip Pullo...	-0.5	0.6

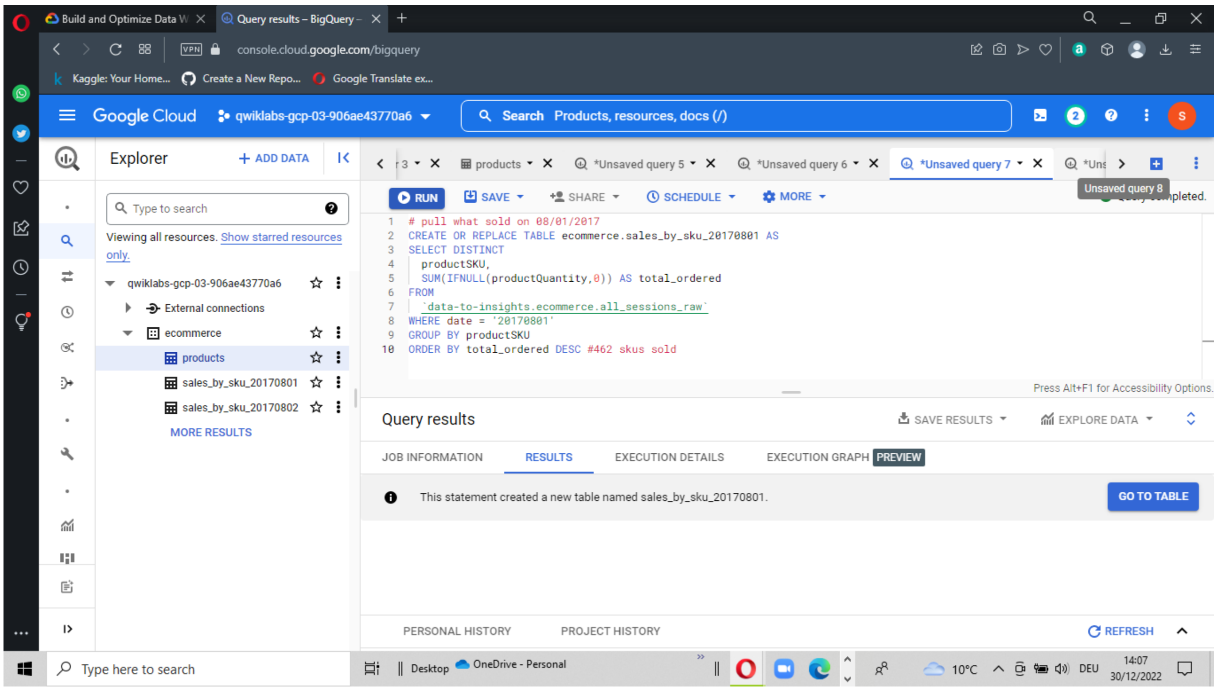
Task 5. Join datasets to find insights

Scenario It's the first of the month and your inventory team has informed you that the orderedQuantity field in the product inventory dataset is out of date. They need your help to query the total sales by product for 08/01/2017 and reference that against the current stock levels in inventory to see which products need to be resupplied first.Calculate daily sales volume by productSKU

1. Create a new table in your **ecommerce** dataset with the below requirements:

- Title it sales_by_sku_20170801
- Source the data from data-to-insights.ecommerce.all_sessions_raw
- Include only distinct results
- Return productSKU
- Return the total quantity ordered (productQuantity). Hint: Use a SUM() with a IFNULL condition
- Filter for only sales on 20170801
- ORDER BY the SKUs with the most orders first

solution:



```
# pull what sold on 08/01/2017
CREATE OR REPLACE TABLE ecommerce.sales_by_sku_20170801 AS
SELECT DISTINCT
  productSKU,
  SUM(IFNULL(productQuantity,0)) AS total_ordered
FROM
  `data-to-insights.ecommerce.all_sessions_raw`
WHERE date = '20170801'
GROUP BY productSKU
ORDER BY total_ordered DESC #462 skus sold
```

This statement created a new table named sales_by_sku_20170801.

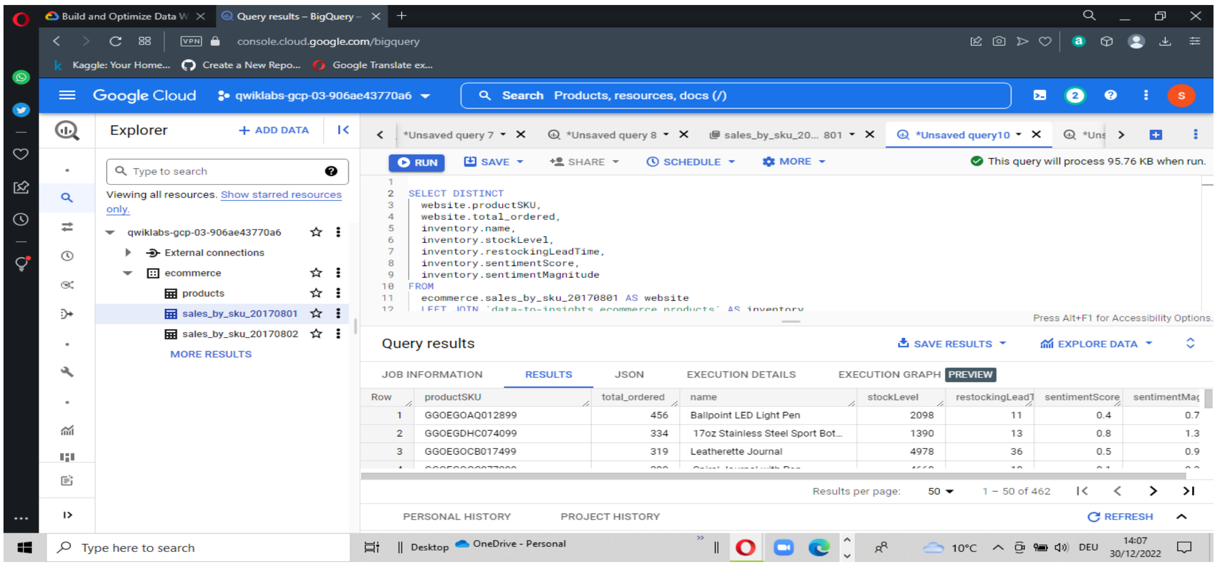
Next, enrich your sales data with product inventory information by joining the two datasets.

Join sales data and inventory data

1. Using a JOIN, enrich the website ecommerce data with the following fields from the product inventory dataset:

- name
- stockLevel
- restockingLeadTime
- sentimentScore
- sentimentMagnitude

Solution:



```
SELECT DISTINCT
  website.productSKU,
  website.total_ordered,
  inventory.name,
  inventory.stockLevel,
  inventory.restockingLeadTime,
  inventory.sentimentScore,
  inventory.sentimentMagnitude
FROM
  ecommerce.sales_by_sku_20170801 AS website
  LEFT JOIN `data-to-insights.ecommerce.products` AS inventory
```

This query will process 95.76 KB when run.

Query results

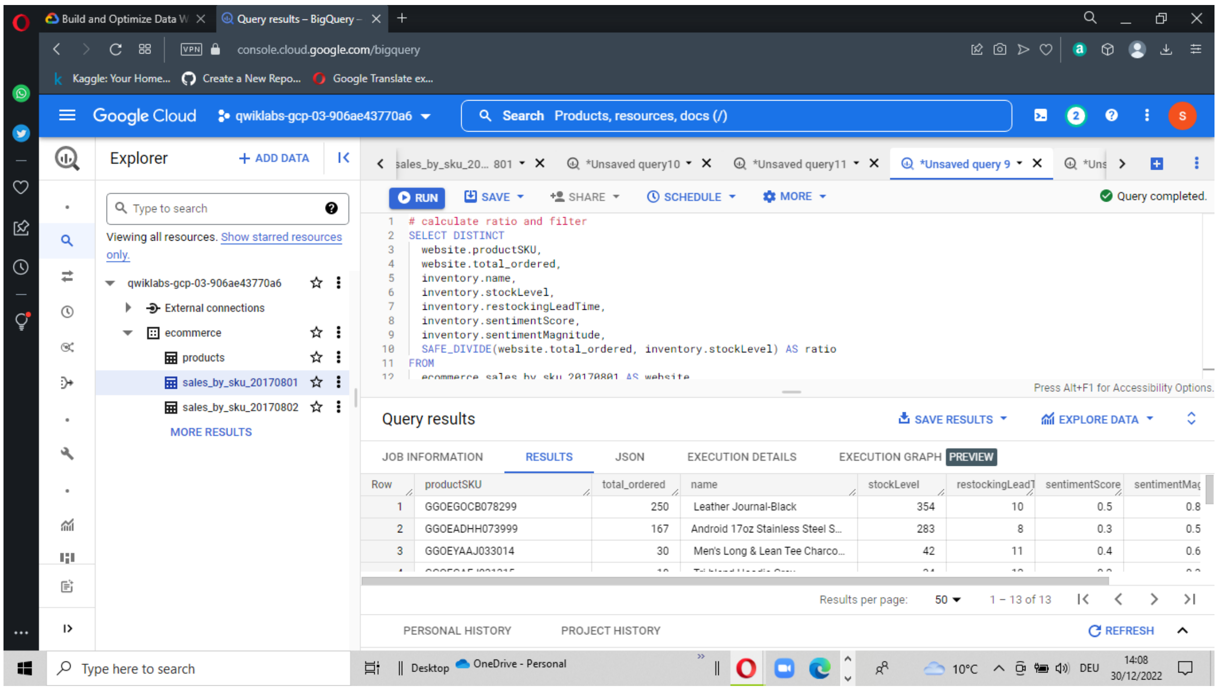
Row	productSKU	total_ordered	name	stockLevel	restockingLeadT	sentimentScore	sentimentMag
1	GGOEGOAQ012899	456	Ballpoint LED Light Pen	2098	11	0.4	0.7
2	GGOEGDHC074099	334	17oz Stainless Steel Sport Bot...	1390	13	0.8	1.3
3	GGOEGOCB017499	319	Leatherette Journal	4978	36	0.5	0.9

Results per page: 50 1 – 50 of 462

2. Modify the query you wrote to now include:

- A calculated field of (total_ordered / stockLevel) and alias it "ratio". **Hint:** Use SAFE_DIVIDE(field1,field2) to avoid divide by 0 errors when the stock level is 0.
- Filter the results to only include products that have gone through 50% or more of their inventory already at the beginning of the month.

Solution:



```
# calculate ratio and filter
SELECT DISTINCT
  website.productSKU,
  website.total_ordered,
  inventory.name,
  inventory.stockLevel,
  inventory.restockingLeadTime,
  inventory.sentimentScore,
  inventory.sentimentMagnitude,
  SAFE_DIVIDE(website.total_ordered, inventory.stockLevel) AS ratio
FROM
  ecommerce.sales_by_sku_20170801 AS website
```

Query completed.

Query results

Row	productSKU	total_ordered	name	stockLevel	restockingLeadT	sentimentScore	sentimentMag
1	GGOEGOCB078299	250	Leather Journal-Black	354	10	0.5	0.8
2	GGOEADHH073999	167	Android 17oz Stainless Steel S...	283	8	0.3	0.5
3	GGOEYAAJ033014	30	Men's Long & Lean Tee Charco...	42	11	0.4	0.6

Results per page: 50 1 – 13 of 13

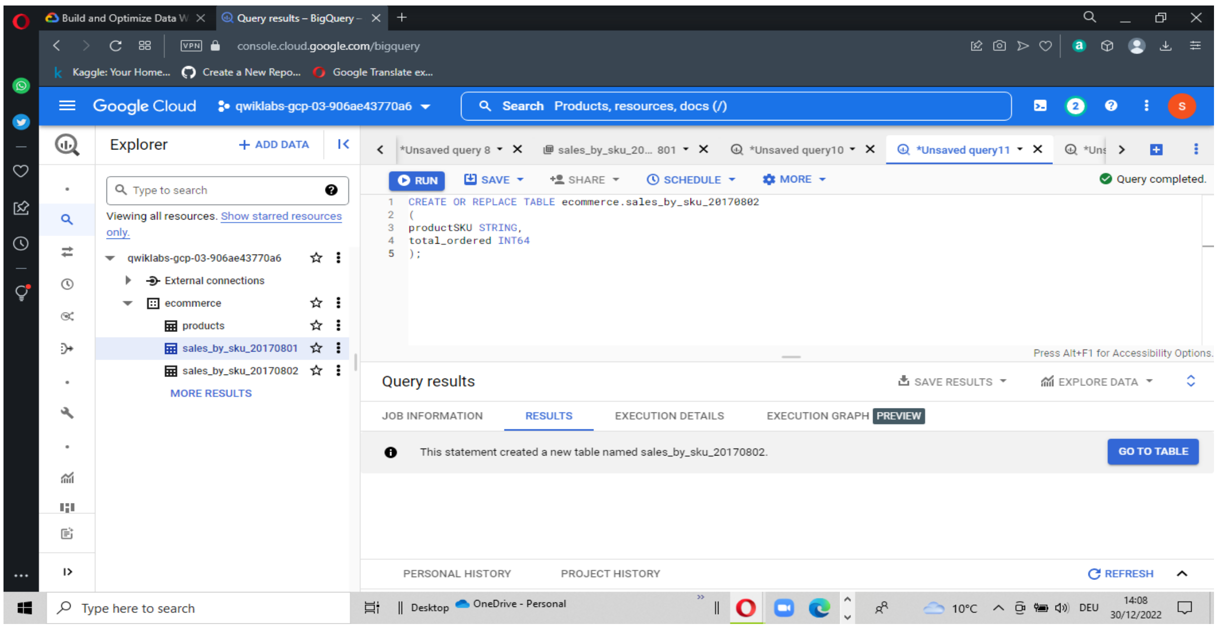
Task 6. Append additional records

Your international team has already made in-store sales on 08/02/2017 which you want to record in your daily sales tables. Create a new empty table to store sales by productSKU for 08/02/2017

1. For the schema, specify the following fields:

- table name is ecommerce.sales_by_sku_20170802
- productSKU STRING
- total_ordered as an INT64 field

Solution:

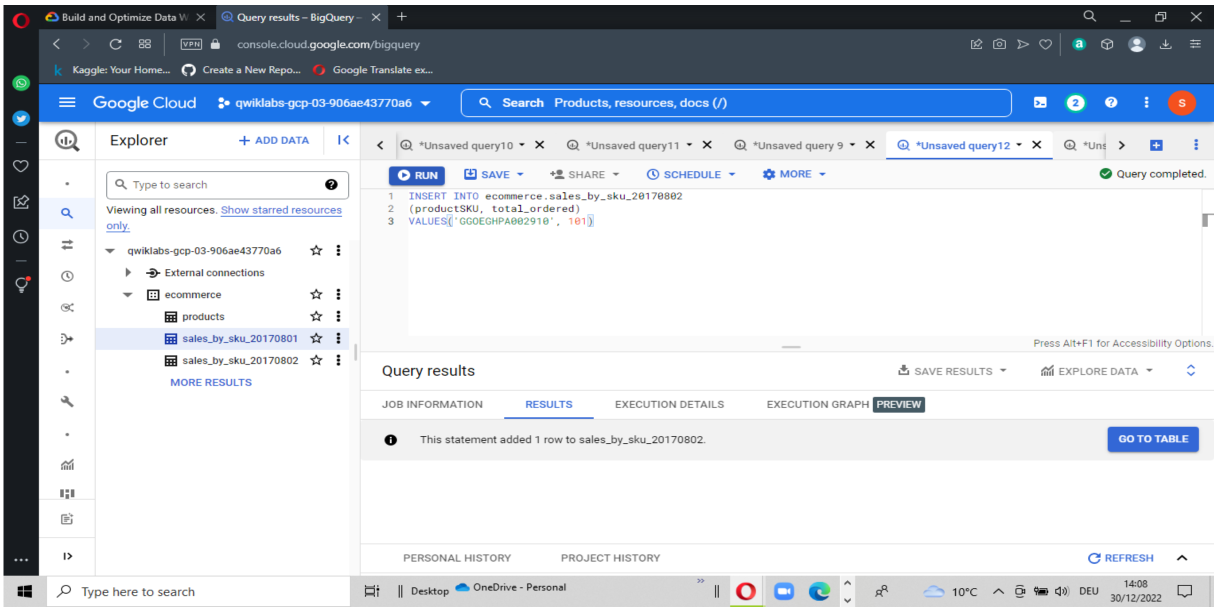


```
CREATE OR REPLACE TABLE ecommerce.sales_by_sku_20170802
(
productSKU STRING,
total_ordered INT64
);
```

This statement created a new table named sales_by_sku_20170802.

2. Insert the sales record provided to you by your sales team:

Solution:



```
INSERT INTO ecommerce.sales_by_sku_20170802
(productSKU, total_ordered)
VALUES('GGOEGHPA002910', 101)
```

This statement added 1 row to sales_by_sku_20170802.

Append together historical data:

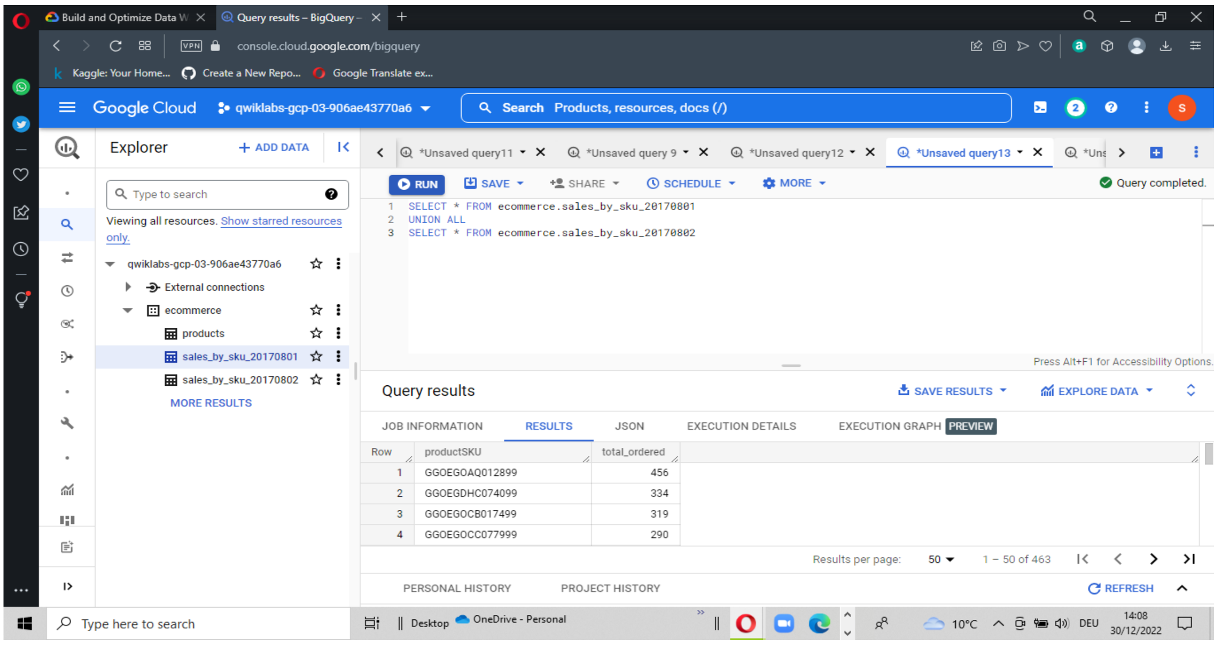
There are multiple ways to append together data that has the same schema. Two common ways are using UNIONs and table wildcards.

- **Union** is an SQL operator that appends together rows from different result sets.
- **Table wildcards** enable you to query multiple tables using concise SQL statements. Wildcard tables are available only in standard SQL.

1. **Write a UNION query that will result in all records from the below two tables:**

- ecommerce.sales_by_sku_20170801
- ecommerce.sales_by_sku_20170802

Solution:



```
SELECT * FROM ecommerce.sales_by_sku_20170801
UNION ALL
SELECT * FROM ecommerce.sales_by_sku_20170802
```

Query completed.

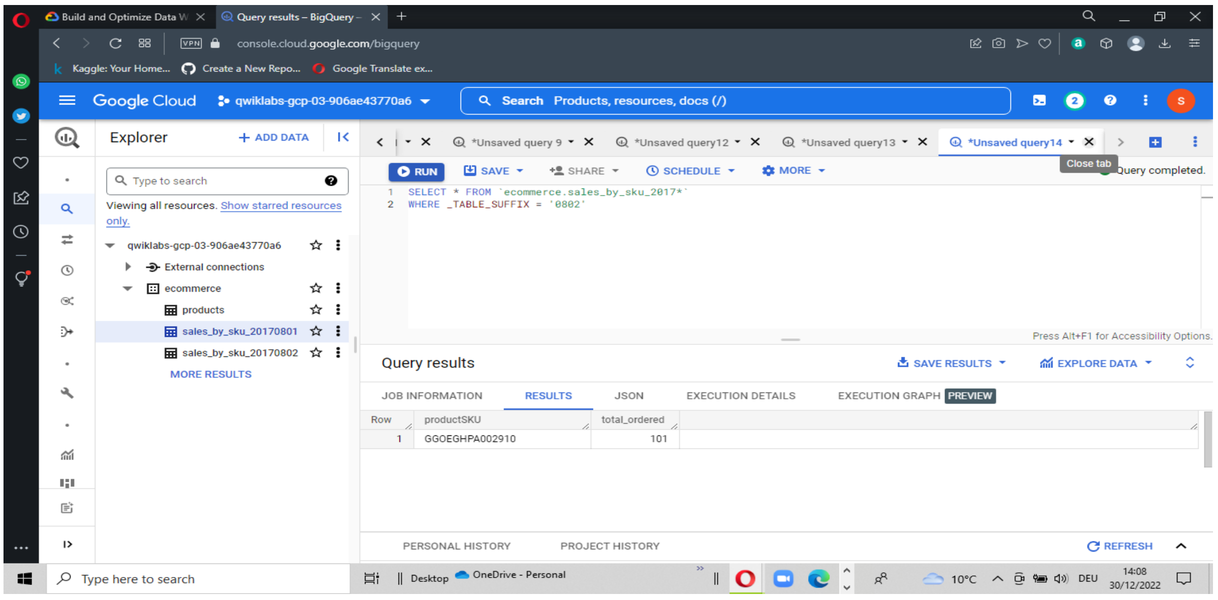
Row	productSKU	total_ordered
1	GGOEGOAQ012899	456
2	GGOEGDHC074099	334
3	GGOEGOCB017499	319
4	GGOEGOCC077999	290

Results per page: 50 1 – 50 of 463

What is a pitfall of having many daily sales tables? You will have to write many UNION statements chained together.
A better solution is to use the table wildcard filter and _TABLE_SUFFIX filter.

2. **Write a query that uses the (*) table wildcard to select all records from ecommerce.sales_by_sku_ for the year 2017.**

Solution:



```
SELECT * FROM `ecommerce.sales_by_sku_2017*`
WHERE _TABLE_SUFFIX = '0802'
```

Query completed.

Row	productSKU	total_ordered
1	GGOEGHPA002910	101