

## Working with JSON, Arrays, and Structs in BigQuery

### Task 1. Create a new dataset to store the tables

1. In your BigQuery, click the three dots next to your Project ID and select **Create dataset**:
2. Name the new dataset `fruit_store`. Leave the other options at their default values (Data Location, Default Expiration).
3. Click **Create dataset**.

### Task 2. Practice working with arrays in SQL

An array is simply a list of items in brackets [ ]

BigQuery visually displays arrays as *flattened*. It simply lists the value in the array vertically (note that all of those values still belong to a single row).

### Solution:

The screenshot shows the Google Cloud BigQuery console interface. On the left, the Explorer pane shows a project named 'qwiklabs-gcp-03-9c82c7270a3e' with a dataset 'fruit\_store' containing a table 'fruit\_details'. The main editor shows a SQL query: `SELECT ['raspberry', 'blackberry', 'strawberry', 'cherry'] AS fruit_array`. The query results are displayed in a table with one row and one column named 'fruit\_array'. The values are listed vertically: 'raspberry', 'blackberry', 'strawberry', and 'cherry'. The bottom of the console shows the Windows taskbar with the date 30/12/2022.

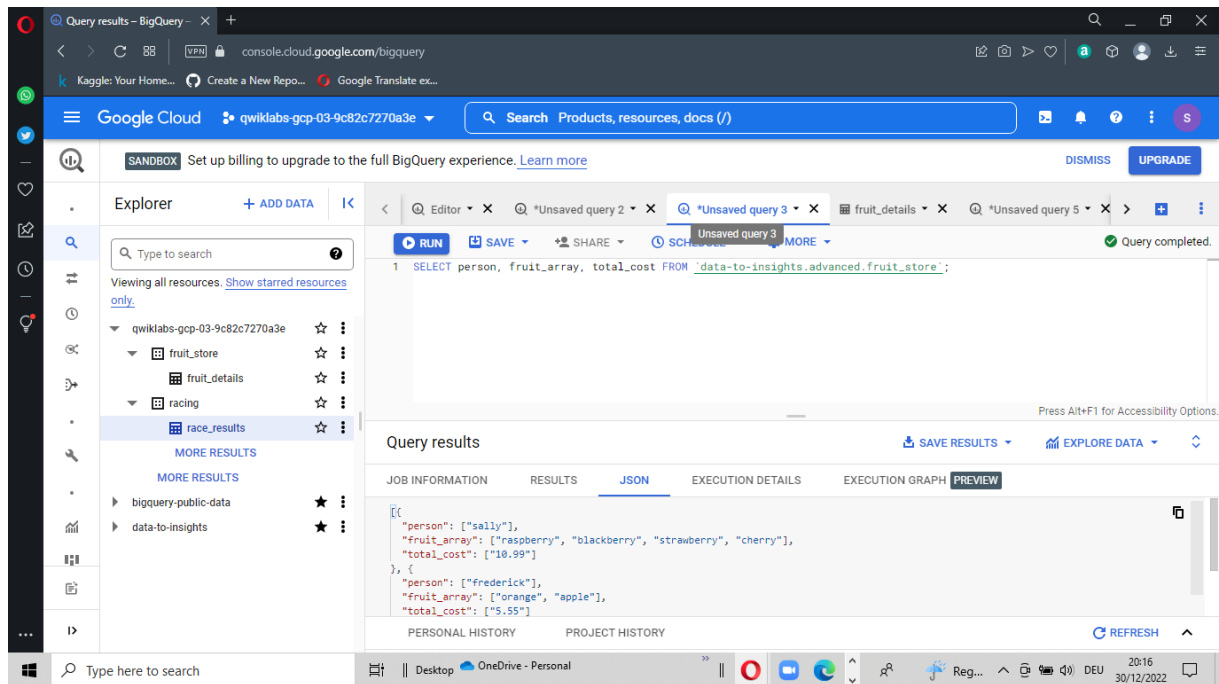
| Row | fruit_array                                     |
|-----|---|
| 1   | raspberry<br>blackberry<br>strawberry<br>cherry |

Arrays can only share one data type (all strings, all numbers).

Here's the final table to query against:

After viewing the results, click the **JSON** tab to view the nested structure of the results.

## Solution:



## Loading semi-structured JSON into BigQuery

What if you had a JSON file that you needed to ingest into BigQuery?

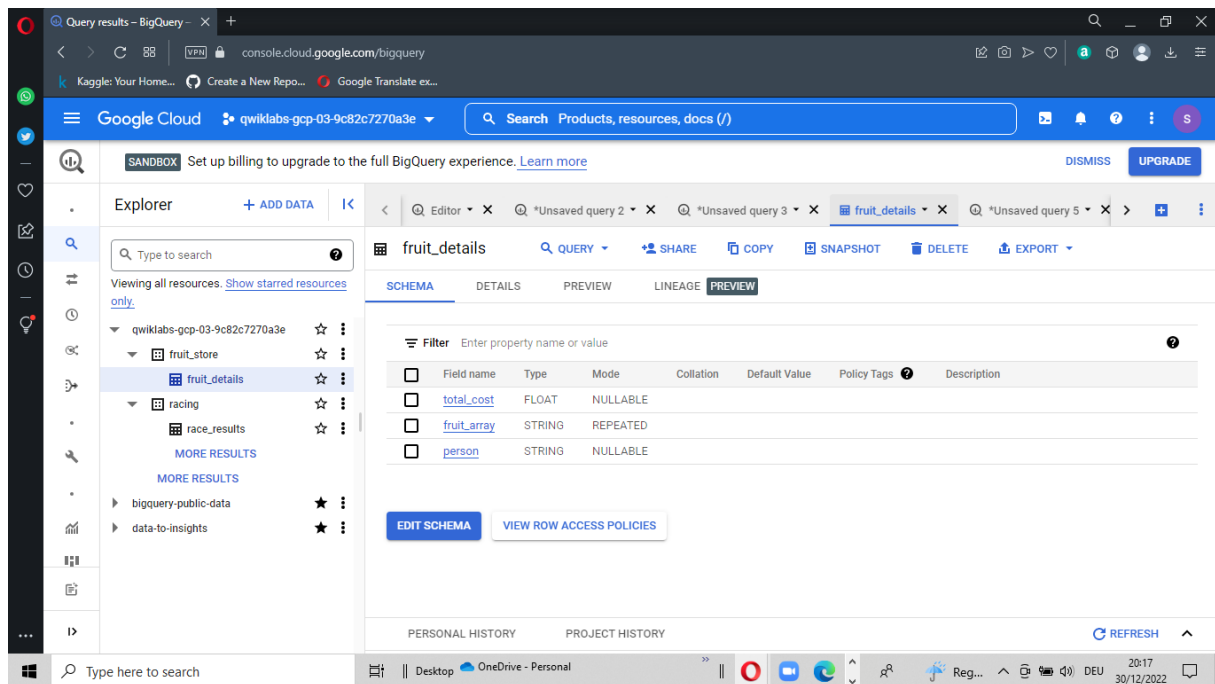
Create a new table `fruit_details` in the dataset.

1. Click on `fruit_store` dataset, then click on the vertical 3-dots, and select **Open**.

Now you will see the **Create Table** option.

2. Name the table `fruit_details`.
3. Add the following details for the table:
  - **Source:** Choose **Google Cloud Storage** in the **Create table from** dropdown.
  - **Select file from Cloud Storage bucket:** `data-insights-course/labs/optimizing-for-performance/shopping_cart.json`
  - **File format:** JSONL (Newline delimited JSON)
4. Call the new table `fruit_details`.
5. Check the checkbox of **Schema (Auto detect)**.
6. Click **Create table**.

## Solution:

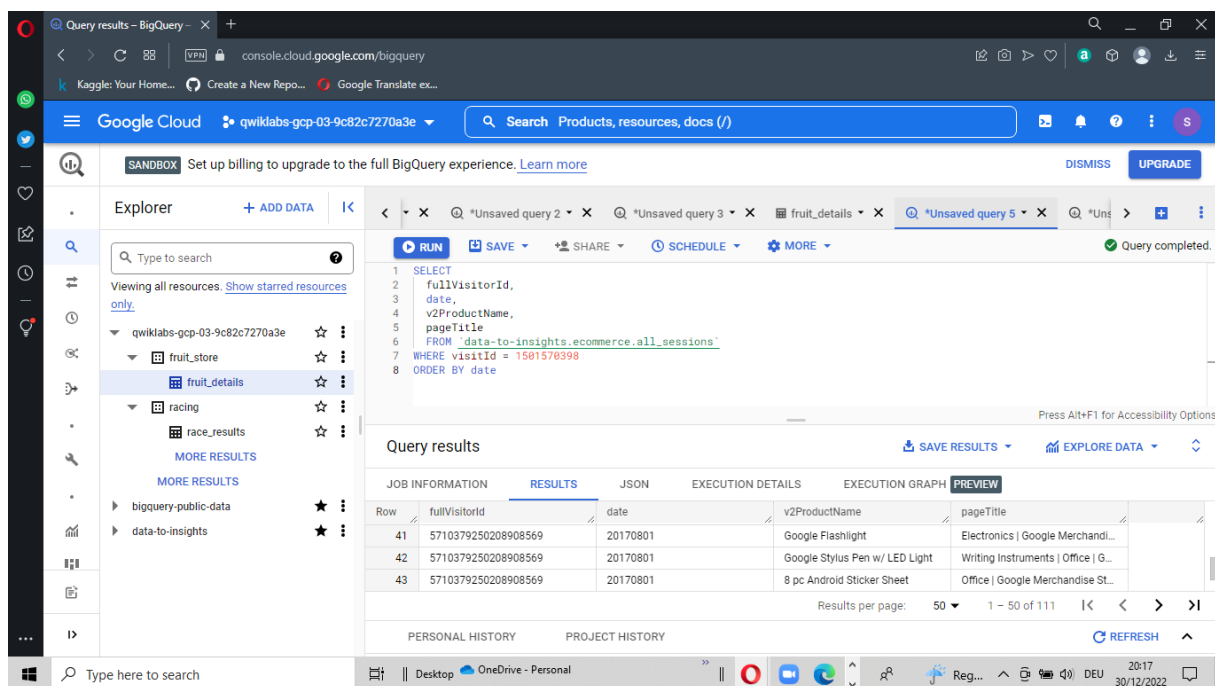


### Task 3. Creating your own arrays with ARRAY\_AGG()

Don't have arrays in your tables already? You can create them!

1. Run the query to explore this public dataset:

## Solution:



Now, use the ARRAY\_AGG() function to aggregate our string values into an array.

## Solution:

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

```
1 SELECT
2   fullVisitorId,
3   date,
4   ARRAY_AGG(v2ProductName) AS products_viewed,
5   ARRAY_AGG(pageTitle) AS pages_viewed
6 FROM `data-to-insights.ecommerce.all_sessions`
7 WHERE visitId = 1501570398
8 GROUP BY fullVisitorId, date
9 ORDER BY date
```

The query results table shows two rows of data:

| Row | fullVisitorId       | date     | products_viewed   | pages_viewed   |
|-----|---------------------|----------|---|--|
| 1   | 5710379250208908569 | 20170731 | Google Snapback Hat Black<br>Google Women's Lightweight ... | Google RFID Journal<br>Google Snapback Hat Black                       |
| 2   | 5710379250208908569 | 20170801 | Android Hard Cover Journal<br>Suitcase Organizer Cubes      | Office   Google Merchandise St...<br>Office   Google Merchandise St... |

Next, use the `ARRAY_LENGTH()` function to count the number of pages and products that were viewed:

## Solution:

The screenshot shows the Google Cloud BigQuery console with a new query. The query editor displays the following SQL query:

```
1 SELECT
2   fullVisitorId,
3   date,
4   ARRAY_AGG(v2ProductName) AS products_viewed,
5   ARRAY_LENGTH(ARRAY_AGG(v2ProductName)) AS num_products_viewed,
6   ARRAY_AGG(pageTitle) AS pages_viewed,
7   ARRAY_LENGTH(ARRAY_AGG(pageTitle)) AS num_pages_viewed
8 FROM `data-to-insights.ecommerce.all_sessions`
9 WHERE visitId = 1501570398
10 GROUP BY fullVisitorId, date
```

The query results table shows two rows of data:

| Row | fullVisitorId       | date     | products_viewed   | num_products_viewed | pages_viewed   |
|-----|---------------------|----------|---|---------------------|--|
| 1   | 5710379250208908569 | 20170731 | Google Snapback Hat Black<br>Google Women's Lightweight ... | 2                   | Google RFID Journal<br>Google Snapback Hat Black                       |
| 2   | 5710379250208908569 | 20170801 | Android Hard Cover Journal<br>Suitcase Organizer Cubes      | 109                 | Office   Google Merchandise St...<br>Office   Google Merchandise St... |

Next, deduplicate the pages and products so you can see how many unique products were viewed by adding `DISTINCT` to `ARRAY_AGG()`:

## Solution:

The screenshot shows the Google Cloud BigQuery console interface. The Explorer on the left lists the project 'qwklabs-gcp-03-9c82c7270a3e' and its datasets, including 'fruit\_store', 'fruit\_details', 'racing', 'race\_results', 'bigquery-public-data', and 'data-to-insights'. The main panel displays a SQL query for 'Unsaved query 8' that selects visitor information and aggregates product and page views. The query results are shown in a table with columns for visitor ID, date, products viewed, distinct products, pages viewed, and distinct pages viewed. The results table has two rows of data.

```
1 SELECT
2   fullVisitorId,
3   date,
4   ARRAY_AGG(DISTINCT v2ProductName) AS products_viewed,
5   ARRAY_LENGTH(ARRAY_AGG(DISTINCT v2ProductName)) AS distinct_products_viewed,
6   ARRAY_AGG(DISTINCT pageTitle) AS pages_viewed,
7   ARRAY_LENGTH(ARRAY_AGG(DISTINCT pageTitle)) AS distinct_pages_viewed
8 FROM `data-to-insights.ecommerce.all_sessions`
9 WHERE visitId = 1501570398
10 GROUP BY fullVisitorId, date
```

| Row | fullVisitorId       | date     | products_viewed                | distinct_products_viewed | pages_viewed                      | distinct_pages_viewed             |
|-----|---------------------|----------|--------------------------------|--------------------------|-----------------------------------|-----------------------------------|
| 1   | 5710379250208908569 | 20170731 | Google Snapback Hat Black      | 2                        | Google RFID Journal               | Google Snapback Hat Black         |
| 2   | 5710379250208908569 | 20170801 | Google Women's Lightweight ... | 61                       | Office   Google Merchandise St... | Electronics   Google Merchandi... |

## Task 4. Querying datasets that already have arrays

The BigQuery Public Dataset for Google Analytics `bigquery-public-data.google_analytics_sample` has many more fields and rows than our course dataset `data-to-insights.ecommerce.all_sessions`. More importantly, it already stores field values like products, pages, and transactions natively as ARRAYS.

## Solution:

The screenshot shows the Google Cloud BigQuery console interface. The Explorer on the left lists the project 'qwklabs-gcp-03-9c82c7270a3e' and its datasets, including 'fruit\_store', 'fruit\_details', 'racing', 'race\_results', 'bigquery-public-data', and 'data-to-insights'. The main panel displays a SQL query for 'Unsaved query 9' that selects visitor information and aggregates pageviews. The query results are shown in a table with columns for visitor ID, visit number, visit ID, visit start time, date, totals visits, totals hits, and totals pageviews. The results table has one row of data.

```
1 SELECT
2   *
3 FROM `bigquery-public-data.google_analytics_sample.ga_sessions_20170801`
4 WHERE visitId = 1501570398
```

| Row | visitorId | visitNumber | visitId    | visitStartTime | date     | totals.visits | totals.hits | totals.pageviews |
|-----|-----------|-------------|------------|----------------|----------|---------------|-------------|------------------|
| 1   | null      | 1           | 1501570398 | 1501570840     | 20170801 | 1             | 11          | 11               |

Before you can query REPEATED fields (arrays) normally, you must first break the arrays back into rows.

For example, the array for hits.page.pageTitle is stored currently as a single row like: ['homepage','product page','checkout'] and it needs to be:

```
['homepage',  
  
'product page',  
  
'checkout']
```

How do you do that with SQL?

**Answer:** Use the UNNEST() function on your array field:

**Solution:**

The screenshot shows the Google Cloud BigQuery console interface. On the left is the Explorer pane showing a project named 'qwiklabs-gcp-03-9c82c7270a3e' with datasets like 'fruit\_store', 'fruit\_details', 'racing', and 'race\_results'. The main area displays a SQL query in the editor:

```
1 SELECT DISTINCT  
2   visitId,  
3   h.page.pageTitle  
4 FROM `bigquery-public-data.google_analytics_sample.ga_sessions_20170801`,  
5 UNNEST(hits) AS h  
6 WHERE visitId = 1501570398  
7 LIMIT 10
```

Below the query editor, the 'Query results' section is visible, showing a table with columns 'visitId' and 'pageTitle'. The first four rows of results are:

| Row | visitId    | pageTitle                         |
|-----|------------|-----------------------------------|
| 1   | 1501570398 | Fun   Accessories   Google Mer... |
| 2   | 1501570398 | Home                              |
| 3   | 1501570398 | Shop by Brand   Google Mercha...  |
| 4   | 1501570398 | Office   Google Merchandise St... |

## Task 5. Introduction to STRUCTs

You may have wondered why the field alias hit.page.pageTitle looks like three fields in one separated by periods. Just as ARRAY values give you the flexibility to *go deep* into the granularity of your fields, another data type allows you to *go wide* in your schema by grouping related fields together. That SQL data type is the [STRUCT](#) data type.

The easiest way to think about a STRUCT is to consider it conceptually like a separate table that is already pre-joined into your main table.

A STRUCT can have:

- One or many fields in it
- The same or different data types for each field
- It's own alias

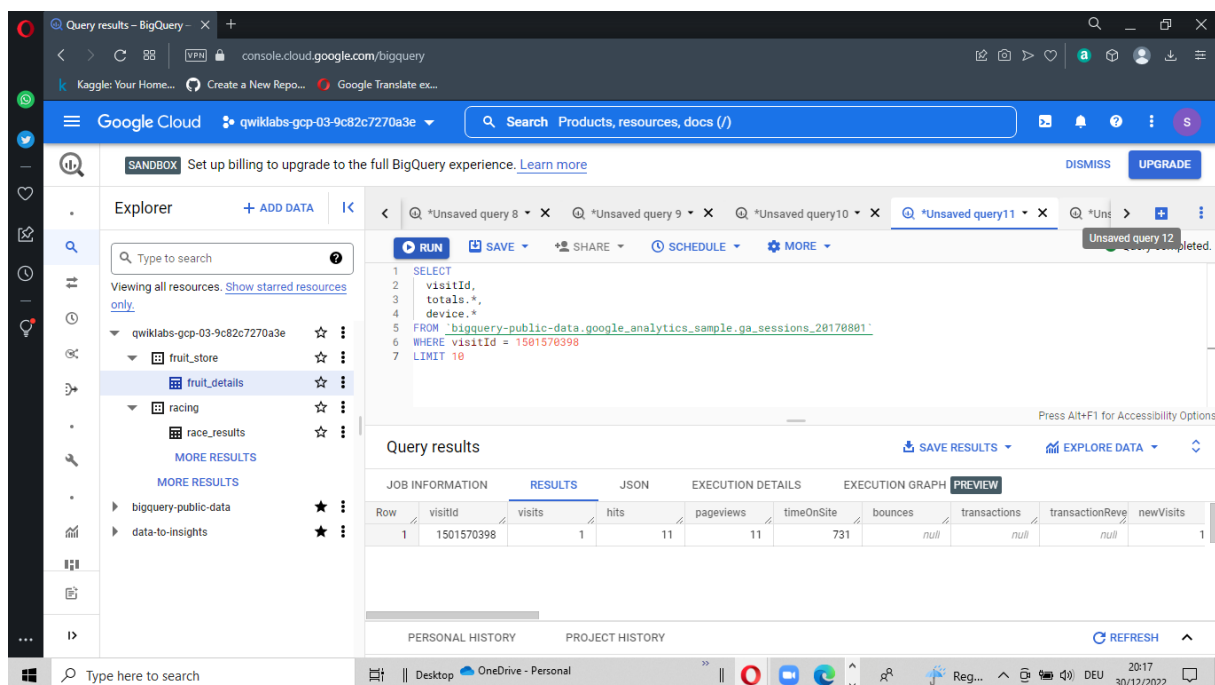
## Explore a dataset with STRUCTs

1. To open the **bigquery-public-data** dataset, click **+Add Data > Pin a project > Enter Project Name**, then write the bigquery-public-data name.
2. Click **Pin**.

The bigquery-public-data project is listed in the Explorer section.

3. Open **bigquery-public-data**.
4. Find and open **google\_analytics\_sample** dataset.
5. Click the **ga\_sessions** table.
6. Start scrolling through the schema and answer the following question by using the find feature of your browser (i.e. CTRL + F).

**Solution:**



The screenshot shows the Google Cloud BigQuery console interface. On the left, the 'Explorer' panel lists the 'bigquery-public-data' project, which is expanded to show the 'google\_analytics\_sample' dataset. The 'ga\_sessions' table is selected. The main panel displays a SQL query: 

```
SELECT visitId, totals.*, device.* FROM `bigquery-public-data.google_analytics_sample.ga_sessions_20170801` WHERE visitId = 1501570398 LIMIT 10
```

. Below the query, the 'Query results' section shows a table with columns: visitId, visits, hits, pageviews, timeOnSite, bounces, transactions, transactionRevenue, and newVisits. The first row of data is visible, showing visitId 1501570398 and 1 visit. The bottom of the screen shows the Windows taskbar with the date 30/12/2022.

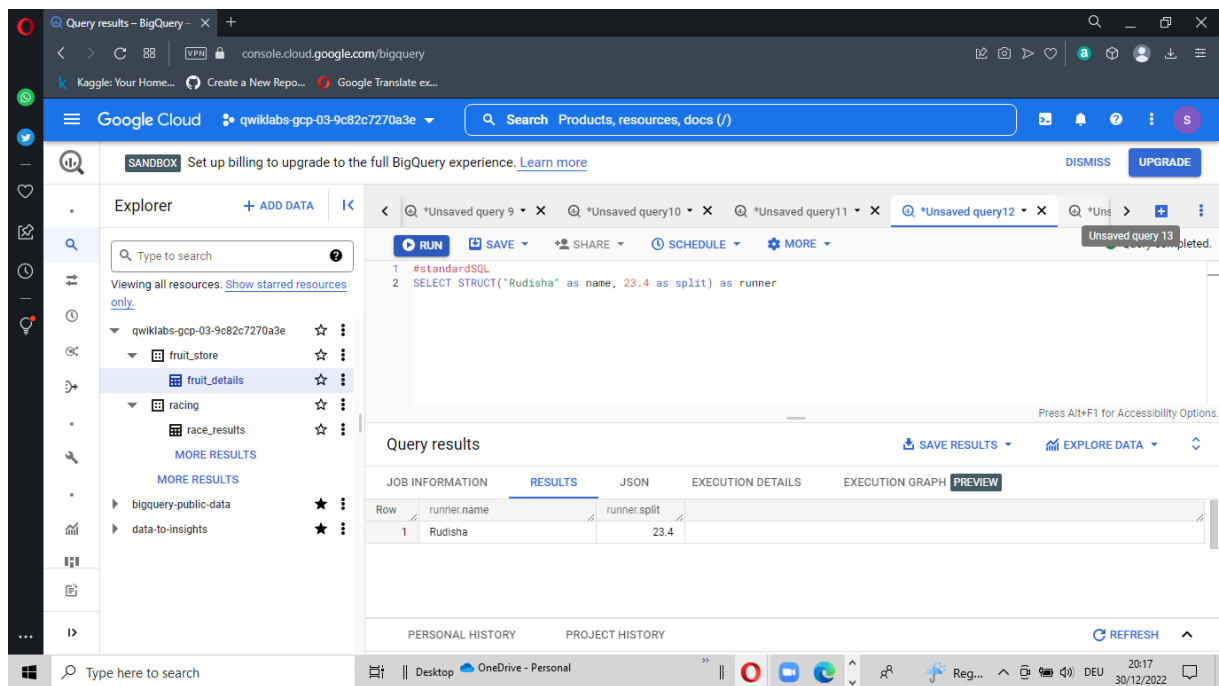
As you can imagine, there is an incredible amount of website session data stored for a modern ecommerce website.

The main advantage of having 32 STRUCTs in a single table is it allows you to run queries like this one without having to do any JOINS:

### Task 6. Practice with STRUCTs and arrays:

The next dataset will be lap times of runners around the track. Each lap will be called a "split".

1. With this query, try out the STRUCT syntax and note the different field types within the struct container:



The screenshot shows the Google Cloud BigQuery console interface. On the left is the Explorer pane showing the project hierarchy: qwiklabs-gcp-03-9c82c7270a3e > fruit\_store > fruit\_details. The main editor shows a SQL query:

```
1 #standardSQL
2 SELECT STRUCT("Rudisha" as name, 23.4 as split) as runner
```

The query has been executed, and the results are displayed in a table:

| Row | runner.name | runner.split |
|-----|-------------|--------------|
| 1   | Rudisha     | 23.4         |

Below the results table, there are tabs for JOB INFORMATION, RESULTS (selected), JSON, EXECUTION DETAILS, and EXECUTION GRAPH. At the bottom, there are sections for PERSONAL HISTORY and PROJECT HISTORY.

With an array of course!



## Solution:

The screenshot shows the Google Cloud BigQuery console interface. The left sidebar displays the Explorer view with a tree structure of datasets and tables. The main panel shows a query editor with a SQL query and its results.

**Query Editor:**

```
1 #standardSQL
2 SELECT STRUCT("Rudisha" as name, [23.4, 26.3, 26.4, 26.1] as splits) AS runner
```

**Query Results:**

| Row | runner.name | runner.splits |
|-----|-------------|---------------|
| 1   | Rudisha     | 23.4          |
|     |             | 26.3          |
|     |             | 26.4          |
|     |             | 26.1          |

## Practice ingesting JSON data

1. Create a new dataset titled racing.
  2. Create a new table titled **race\_results**.
  3. Click on racing dataset and click Create table.
- **Source:** select **Google Cloud Storage** under **Create table from** dropdown.
  - **Select file from Cloud Storage bucket:** data-insights-course/labs/optimizing-for-performance/race\_results.json
  - **File format:** JSONL (Newline delimited JSON)
  - In **Schema**, click on **Edit as text** slider and add the following:

## Solution:

The screenshot shows the Google Cloud BigQuery console interface. The left sidebar contains the Explorer panel with a search bar and a tree view of resources. The main panel displays the schema for the 'race\_results' table. The schema table has the following columns: Field name, Type, Mode, Collation, Default Value, Policy Tags, and Description.

| Field name   | Type   | Mode     | Collation | Default Value | Policy Tags | Description |
|--------------|--------|----------|-----------|---------------|-------------|-------------|
| race         | STRING | NULLABLE |           |               |             |             |
| participants | RECORD | REPEATED |           |               |             |             |

Buttons for 'EDIT SCHEMA' and 'VIEW ROW ACCESS POLICIES' are visible below the schema table. The bottom status bar shows the system clock as 20:17 on 30/12/2022.

## Practice querying nested and repeated fields

1. Let's see all of our racers for the 800 Meter race:

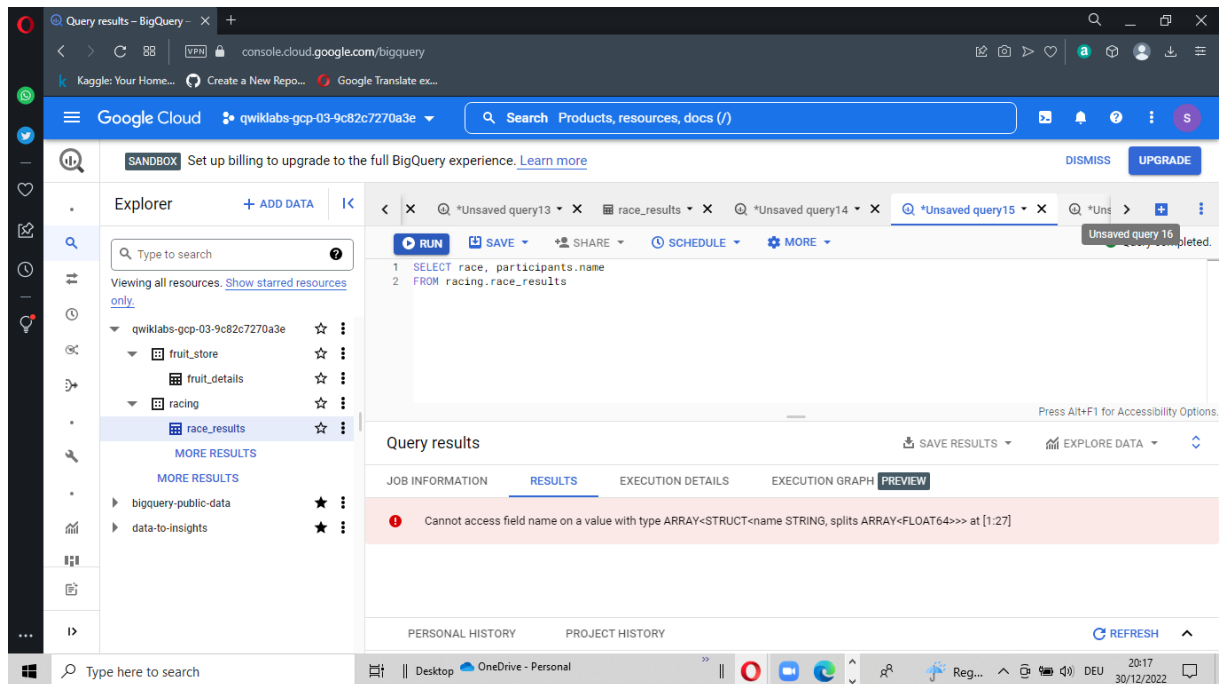
The screenshot shows the Google Cloud BigQuery console with a query executed. The query is: `SELECT * FROM racing.race_results`. The results are displayed in a table with the following columns: Row, race, participants.name, and participants.splits. The first row shows the 800M race with participant 'Rudisha' and splits of 23.4, 26.3, 26.4, and 26.1.

| Row | race | participants.name | participants.splits |
|-----|------|-------------------|---------------------|
| 1   | 800M | Rudisha           | 23.4                |
|     |      |                   | 26.3                |
|     |      |                   | 26.4                |
|     |      |                   | 26.1                |

The bottom status bar shows the system clock as 20:17 on 30/12/2022.

What if you wanted to list the name of each runner and the type of race?

**Solution:**

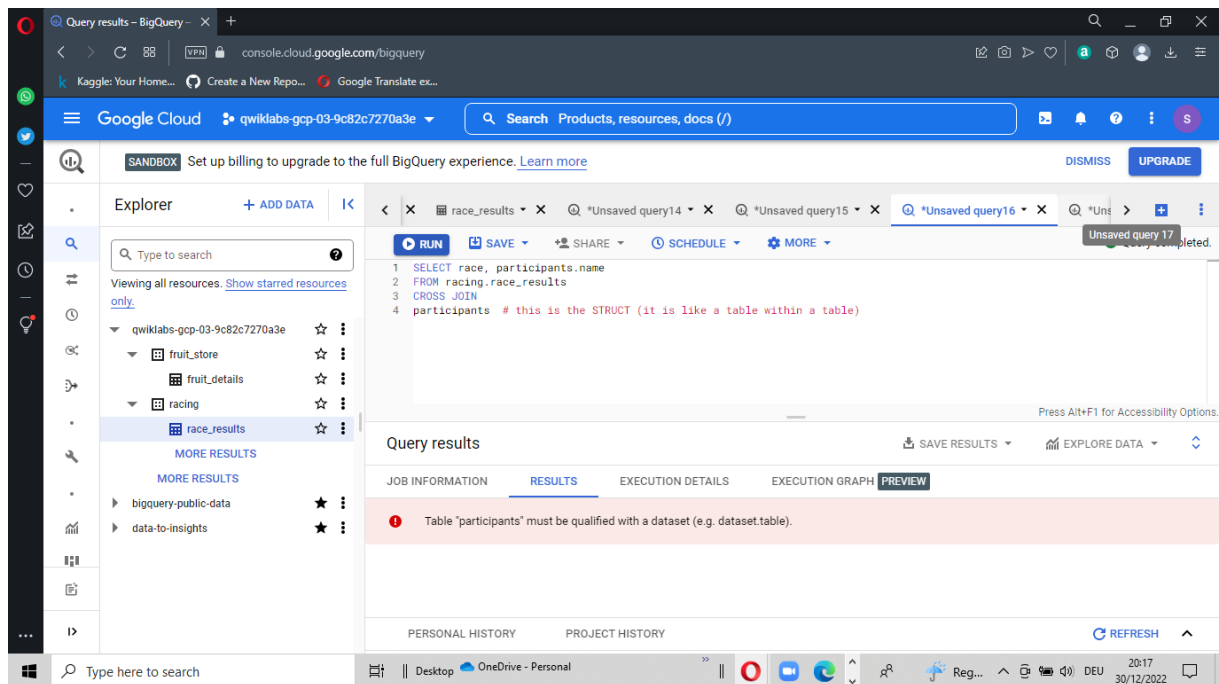


In traditional relational SQL, if you had a races table and a participants table what would you do to get information from both tables? You would JOIN them together. Here the participant STRUCT (which is conceptually very similar to a table) is already part of your races table but is not yet correlated correctly with your non-STRUCT field "race".

Can you think of what two words SQL command you would use to correlate the 800M race with each of the racers in the first table?

**Answer:** CROSS JOIN

## Solution:



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

```
1 SELECT race, participants.name
2 FROM racing.race_results
3 CROSS JOIN
4 participants # this is the STRUCT (it is like a table within a table)
```

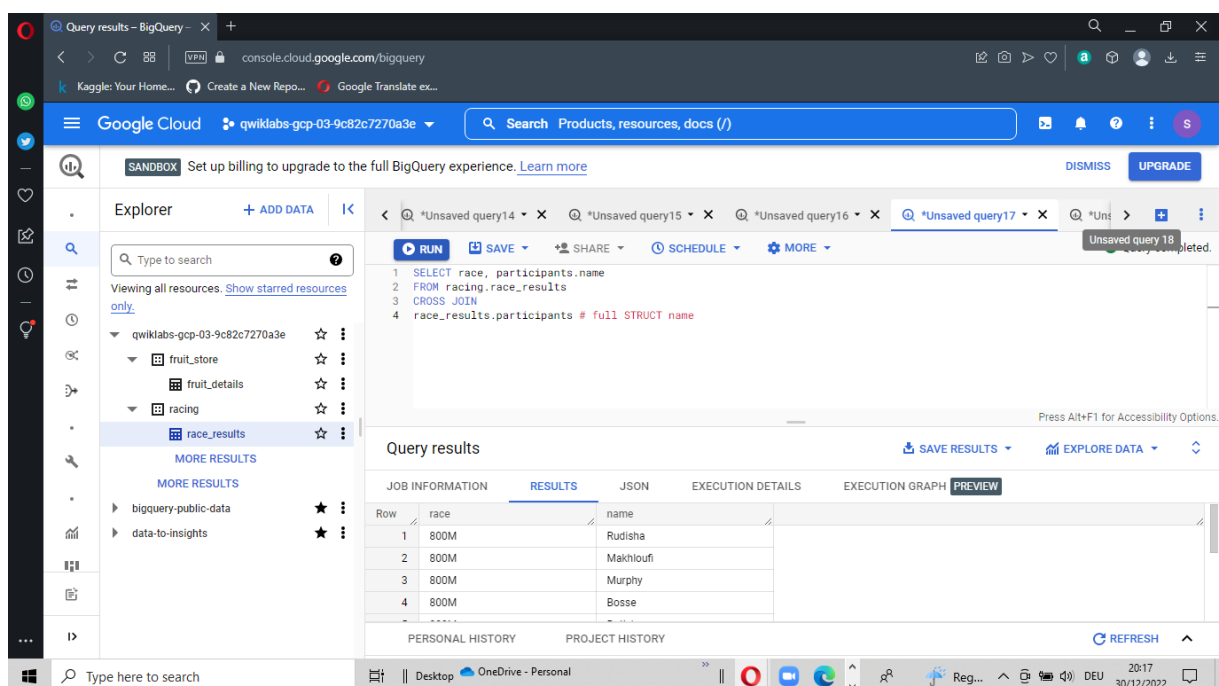
The query results section shows an error message: "Table 'participants' must be qualified with a dataset (e.g. dataset.table)." The Explorer on the left shows the project structure, including the 'racing' dataset and its 'race\_results' table.

Table name "participants" missing dataset while no default dataset is set in the request.

Even though the participants STRUCT is like a table, it is still technically a field in the racing.race\_results table.

Add the dataset name to the query:

## Solution:



The screenshot shows the Google Cloud BigQuery console with the corrected query:

```
1 SELECT race, participants.name
2 FROM racing.race_results
3 CROSS JOIN
4 race_results.participants # full STRUCT name
```

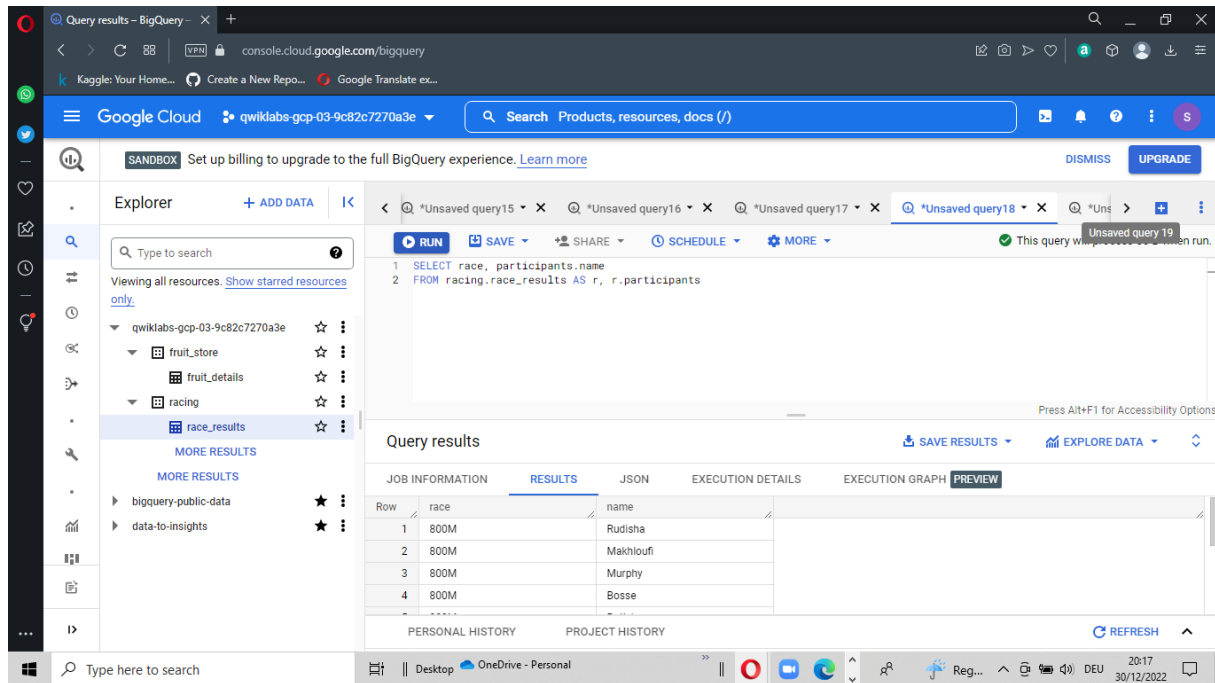
The query results are displayed in a table format:

| Row | race | name      |
|-----|------|-----------|
| 1   | 800M | Rudisha   |
| 2   | 800M | Makhloufi |
| 3   | 800M | Murphy    |
| 4   | 800M | Bosse     |

simplify the last query by:

- Adding an alias for the original table
- Replacing the words "CROSS JOIN" with a comma (a comma implicitly cross joins)

This will give you the same query result:



The screenshot shows the Google Cloud BigQuery console interface. The left sidebar contains the Explorer panel with a search bar and a list of resources. The main panel displays a query editor with the following SQL code:

```
1 SELECT race, participants.name
2 FROM racing.race_results AS r, r.participants
```

Below the query editor, the 'Query results' section is visible, showing a table with the following data:

| Row | race | name      |
|-----|------|-----------|
| 1   | 800M | Rudisha   |
| 2   | 800M | Makhloufi |
| 3   | 800M | Murphy    |
| 4   | 800M | Bosse     |

If you have more than one race type (800M, 100M, 200M), wouldn't a CROSS JOIN just associate every racer name with every possible race like a cartesian product?

**Answer:** No. This is a *correlated* cross join which only unpacks the elements associated with a single row. For a greater discussion, see [working with ARRAYS and STRUCTs](#).

Recap of STRUCTs:

- A SQL [STRUCT](#) is simply a container of other data fields which can be of different data types. The word struct means data structure. Recall the example from earlier: `STRUCT(`"Rudisha" as name, [23.4, 26.3, 26.4, 26.1] as splits)` AS runner`
- STRUCTs are given an alias (like runner above) and can conceptually be thought of as a table inside of your main table.
- STRUCTs (and ARRAYS) must be unpacked before you can operate over their elements. Wrap an `UNNEST()` around the name of the struct itself or the struct field that is an array in order to unpack and flatten it.

## Task 7. Lab question: STRUCT()

Answer the below questions using the racing.race\_results table you created previously.

**Task:** Write a query to COUNT how many racers were there in total.

- To start, use the below partially written query:

**Solution:**

The screenshot shows the Google Cloud BigQuery console interface. On the left, the Explorer pane displays the project hierarchy, with the 'race\_results' table under the 'racing' dataset selected. The main editor area contains a SQL query:

```
1 SELECT COUNT(p.name) AS racer_count
2 FROM racing.race_results AS r, UNNEST(r.participants) AS p
```

Below the query editor, the 'Query results' section is visible, showing a table with one row and one column:

| Row | racer_count |
|-----|-------------|
| 1   | 8           |

The bottom of the console shows the 'PERSONAL HISTORY' and 'PROJECT HISTORY' tabs, and a 'REFRESH' button.

## Task 8. Lab question: Unpacking arrays with UNNEST( )

Write a query that will list the total race time for racers whose names begin with R. Order the results with the fastest total time first. Use the UNNEST() operator and start with the partially written query below.

## Solution:

The screenshot shows the Google Cloud BigQuery console. The Explorer on the left lists datasets: `fruit_store`, `fruit_details`, `racing`, `race_results`, `bigquery-public-data`, and `data-to-insights`. The `race_results` dataset is selected. The query editor shows the following SQL:

```
1 SELECT
2   p.name,
3   SUM(split_times) as total_race_time
4 FROM racing.race_results AS r
5 , UNNEST(r.participants) AS p
6 , UNNEST(p.splits) AS split_times
7 WHERE p.name LIKE 'R%'
8 GROUP BY p.name
9 ORDER BY total_race_time ASC;
```

The query results table shows two rows:

| Row | name    | total_race_time |
|-----|---------|-----------------|
| 1   | Rudisha | 102.199999...   |
| 2   | Rotich  | 103.6           |

## Task 9. Filtering within array values

You happened to see that the fastest lap time recorded for the 800 M race was 23.2 seconds, but you did not see which runner ran that particular lap. Create a query that returns that result.

## Solution:

The screenshot shows the Google Cloud BigQuery console. The Explorer on the left lists datasets: `fruit_store`, `fruit_details`, `racing`, `race_results`, `bigquery-public-data`, and `data-to-insights`. The `race_results` dataset is selected. The query editor shows the following SQL:

```
1 SELECT
2   p.name,
3   split_time
4 FROM racing.race_results AS r
5 , UNNEST(r.participants) AS p
6 , UNNEST(p.splits) AS split_time
7 WHERE split_time = 23.2;
```

The query results table shows one row:

| Row | name     | split_time |
|-----|----------|------------|
| 1   | Kipketer | 23.2       |

