How to import UNNESTED data from BigQuery to Power BI

By Sangeetha Mahesh

Hello, welcome to this Inflexion Analytics technical blog. If you've reached here then it's safe to presume that you'd like to know how to import 'unnested' data from BigQuery to Power BI. So, let's get started.

What is UNNEST?

UNNEST is a function used in Google BigQueryto convert an array into a set of rows. This process is also known as "flattening". UNNEST takes an array as the input and returns a table with a row for each element in the array. Here's more information on working with arrays.

Why UNNEST data?

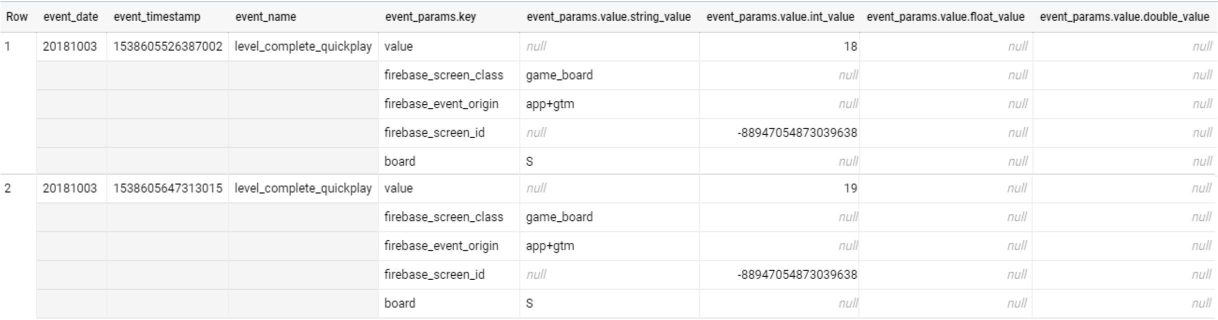
Nesting is done to improve computational performance and it is how BigQuery maintains the denormalized data. Now, we want to use this data in a tool like Power BI, but the BigQuery connector still doesn't support nested and/or repeated fields. So, we'll need to flatten the data before it could be consumed in most BI tools.

How does it work?

I’ll be making use of the public sample data for sample game of Flood-it. This data is available for everyone, so you can follow along. Let’s take a peek at the data.

```
SELECT * FROM `firebase-public-project.analytics_153293282.events_20181003`
```

If you run this, you’ll see a number of events and event parameters associated with each event.



Row	event_date	event_timestamp	event_name	event_params.key	event_params.value.string_value	event_params.value.int_value	event_params.value.float_value	event_params.value.double_value
1	20181003	1538605526387002	level_complete_quickplay	value	null	18	null	null
				firebase_screen_class	game_board	null	null	null
				firebase_event_origin	app+gtm	null	null	null
				firebase_screen_id	null	-88947054873039638	null	null
				board	S	null	null	null
2	20181003	1538605647313015	level_complete_quickplay	value	null	19	null	null
				firebase_screen_class	game_board	null	null	null
				firebase_event_origin	app+gtm	null	null	null
				firebase_screen_id	null	-88947054873039638	null	null
				board	S	null	null	null

You can notice that for each event parameter, there can be several event_params values. It is essentially an array and this is the nested data that we need to tackle.

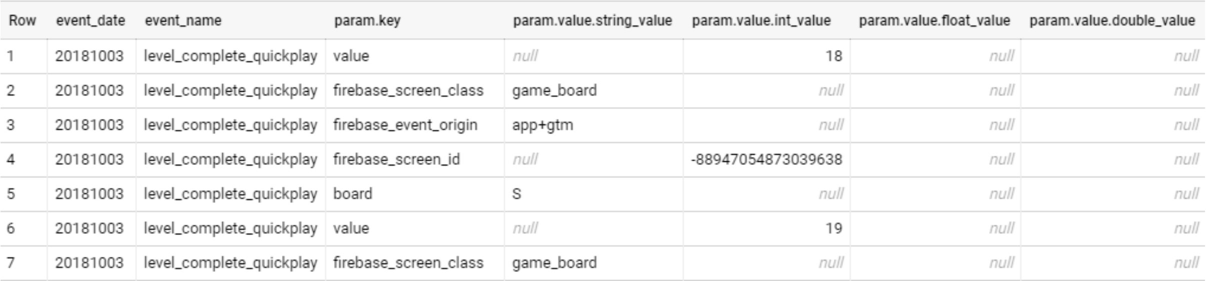
Untangling the data

The first step is to use UNNEST function and store the Analytics data in a temporary BigQuery table.

Here’s the query you’ll need to use to get unnested data.

```
SELECT event_date, event_name, param FROM `firebase-public-
project.analytics_153293282.events_20181003`
CROSS JOIN UNNEST(event_params) as paraM
```

You might notice that I’ve chosen specific columns instead of selecting them all. This is because for selecting all the columns, you would need to provide each column name in the select statement. Just trying to keep it short as it’s an example. Here’s how the data looks:



Row	event_date	event_name	param.key	param.value.string_value	param.value.int_value	param.value.float_value	param.value.double_value
1	20181003	level_complete_quickplay	value	null	18	null	null
2	20181003	level_complete_quickplay	firebase_screen_class	game_board	null	null	null
3	20181003	level_complete_quickplay	firebase_event_origin	app+gtm	null	null	null
4	20181003	level_complete_quickplay	firebase_screen_id	null	-88947054873039638	null	null
5	20181003	level_complete_quickplay	board	S	null	null	null
6	20181003	level_complete_quickplay	value	null	19	null	null
7	20181003	level_complete_quickplay	firebase_screen_class	game_board	null	null	null

Notice how the ‘event_date’ and the ‘event_name’ are repeated for each of the ‘param.key’ values. Now the data looks more familiar.

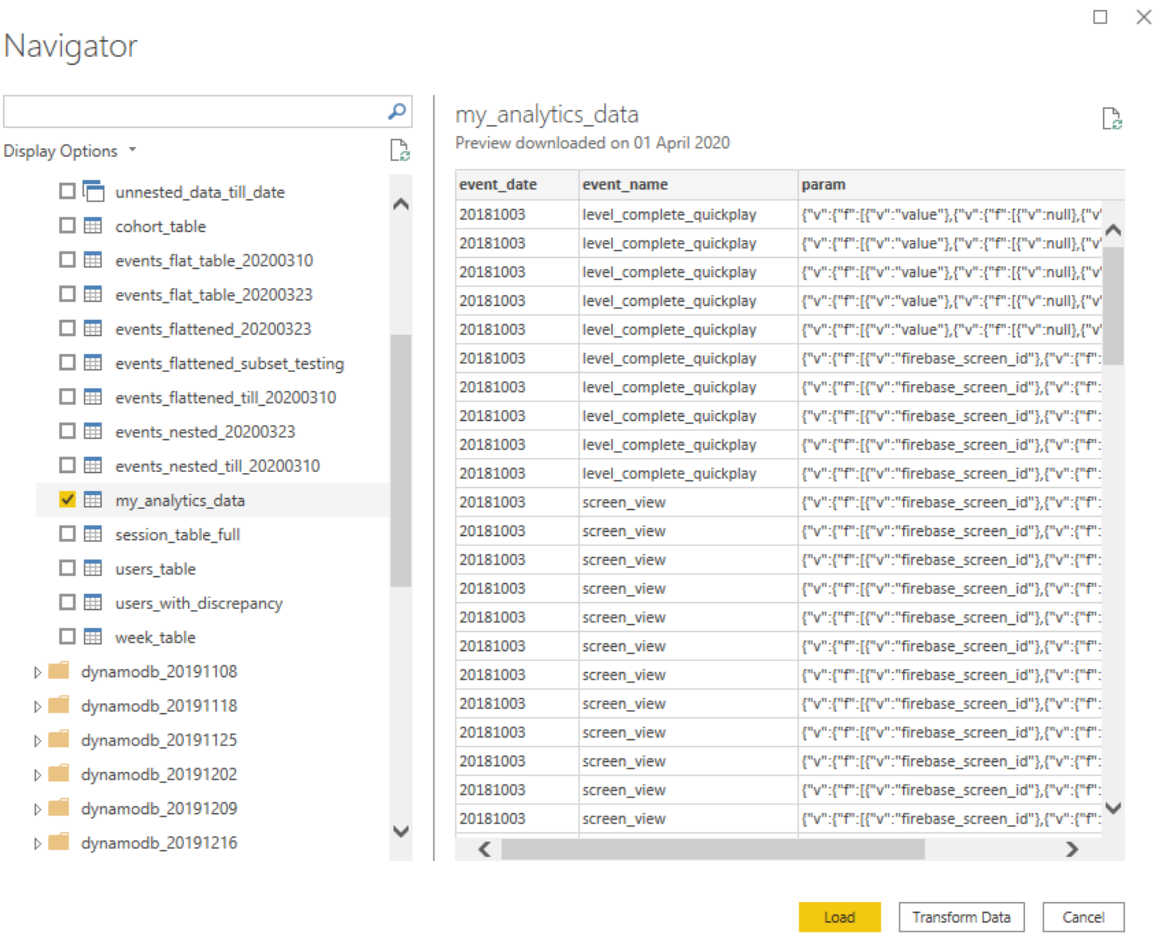
The query results can be saved on to a BigQuery table so that it can be accessed later. Click on the Save Query Resultsoption and choose a Project, Dataset and a Table name. I've saved it under the table name 'my_analytics_data'.

It is possible to use multiple UNNEST functions in a single query. Also, it's good practice to replace the 'CROSS JOIN' by a comma. Here's an example

```
SELECT event_date, event_name, param, User_PROP FROM `firebasepublic-
project.analytics_153293282.events_20181003`,
UNNEST(event_params) as paraM,
UNNEST(USER_PROPERTIES) AS USER_PROP
```

Importing the data to Power BI

Now that we have it all unnested and ready, let's try importing it to Power BI. Open Power BI and click on the 'Get Data' option. Search for Google BigQueryand select it. Once it opens up, navigate to your temporary table(my_analytics_data) and click on Load.



Navigator

Display Options

- □ unnested_data_till_date
- □ cohort_table
- □ events_flat_table_20200310
- □ events_flat_table_20200323
- □ events_flattened_20200323
- □ events_flattened_subset_testing
- □ events_flattened_till_20200310
- □ events_nested_20200323
- □ events_nested_till_20200310
- ☑ my_analytics_data
- □ session_table_full
- □ users_table
- □ users_with_discrepancy
- □ week_table
- dynamodb_20191108
- dynamodb_20191118
- dynamodb_20191125
- dynamodb_20191202
- dynamodb_20191209
- dynamodb_20191216

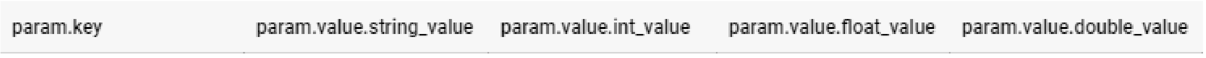
my_analytics_data

Preview downloaded on 01 April 2020

event_date	event_name	param
20181003	level_complete_quickplay	{"v":{"f":[{"v":"value"},{"v":{"f":[{"v":null},{"v'
20181003	level_complete_quickplay	{"v":{"f":[{"v":"value"},{"v":{"f":[{"v":null},{"v'
20181003	level_complete_quickplay	{"v":{"f":[{"v":"value"},{"v":{"f":[{"v":null},{"v'
20181003	level_complete_quickplay	{"v":{"f":[{"v":"value"},{"v":{"f":[{"v":null},{"v'
20181003	level_complete_quickplay	{"v":{"f":[{"v":"value"},{"v":{"f":[{"v":null},{"v'
20181003	level_complete_quickplay	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":
20181003	level_complete_quickplay	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":
20181003	level_complete_quickplay	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":
20181003	level_complete_quickplay	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":
20181003	level_complete_quickplay	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":
20181003	screen_view	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":
20181003	screen_view	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":
20181003	screen_view	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":
20181003	screen_view	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":
20181003	screen_view	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":
20181003	screen_view	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":
20181003	screen_view	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":
20181003	screen_view	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":
20181003	screen_view	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":
20181003	screen_view	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":
20181003	screen_view	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":
20181003	screen_view	{"v":{"f":[{"v":"firebase_screen_id"},{"v":{"f":

It looks like something's not right …in the preview.

Here's why; we'll need an additional step to get the data readable by Power BI. UNNEST function automatically allocates column names which are essentially made up of the array name followed by the nested parameter's name with a period separating them. It looks something like this

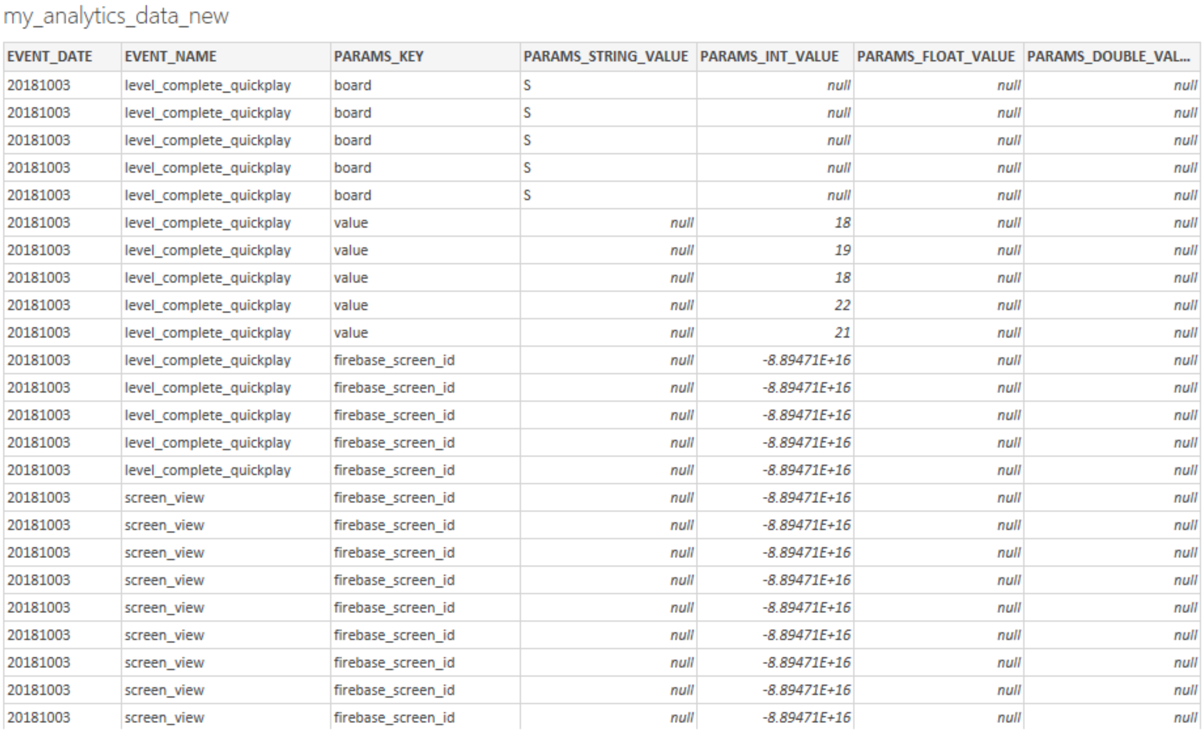


param.key	param.value.string_value	param.value.int_value	param.value.float_value	param.value.double_value

This format isn't handled well by Power BI and doesn't render properly. The solution is pretty simple. We need to provide aliases for each of unnested column names while selecting them from the stored table. I'll save the below query results in another table 'my_analytics_data_new'

```
SELECT event_date, event_name, param.key as params_key,
param.value.string_value as params_string_value, param.value.int_value as
params_int_value, param.value.float_value as params_float_value,
param.value.double_value as params_double_value
FROM `MY_ANALYTICS_DATA`
```

Let's just check if this worked for us in Power BI by connecting to the my_analytics_data_newtable in BigQuery



my_analytics_data_new

EVENT_DATE	EVENT_NAME	PARAMS_KEY	PARAMS_STRING_VALUE	PARAMS_INT_VALUE	PARAMS_FLOAT_VALUE	PARAMS_DOUBLE_VAL...
20181003	level_complete_quickplay	board	S	null	null	null
20181003	level_complete_quickplay	board	S	null	null	null
20181003	level_complete_quickplay	board	S	null	null	null
20181003	level_complete_quickplay	board	S	null	null	null
20181003	level_complete_quickplay	board	S	null	null	null
20181003	level_complete_quickplay	value	null	18	null	null
20181003	level_complete_quickplay	value	null	19	null	null
20181003	level_complete_quickplay	value	null	18	null	null
20181003	level_complete_quickplay	value	null	22	null	null
20181003	level_complete_quickplay	value	null	21	null	null
20181003	level_complete_quickplay	firebase_screen_id	null	-8.89471E+16	null	null
20181003	level_complete_quickplay	firebase_screen_id	null	-8.89471E+16	null	null
20181003	level_complete_quickplay	firebase_screen_id	null	-8.89471E+16	null	null
20181003	level_complete_quickplay	firebase_screen_id	null	-8.89471E+16	null	null
20181003	level_complete_quickplay	firebase_screen_id	null	-8.89471E+16	null	null
20181003	screen_view	firebase_screen_id	null	-8.89471E+16	null	null
20181003	screen_view	firebase_screen_id	null	-8.89471E+16	null	null
20181003	screen_view	firebase_screen_id	null	-8.89471E+16	null	null
20181003	screen_view	firebase_screen_id	null	-8.89471E+16	null	null
20181003	screen_view	firebase_screen_id	null	-8.89471E+16	null	null
20181003	screen_view	firebase_screen_id	null	-8.89471E+16	null	null
20181003	screen_view	firebase_screen_id	null	-8.89471E+16	null	null
20181003	screen_view	firebase_screen_id	null	-8.89471E+16	null	null
20181003	screen_view	firebase_screen_id	null	-8.89471E+16	null	null

Worked like a charm! We hope this was useful and feel free to ask any questions us at contact@inflexionanalytics.com