Tweets at MPs: an explanation of SQL queries we have run in BigQuery

In order to make it easier for hackday participants to query the data, we have run some initial SQL queries ourselves to clean it up, merge it, and do other things. This document provides some background on those queries and processes.

# Rename fields with AS

The fields in our tables don’t have any meaningful names - and there are too many. The query below names them.

SELECT string\_field\_0 AS tweet\_id,

string\_field\_1 AS conversation\_id,

string\_field\_2 AS created\_at,

string\_field\_3 AS `date`,

string\_field\_4 AS `time`,

string\_field\_5 AS timezone,

string\_field\_6 AS user\_id,

string\_field\_7 AS username,

string\_field\_8 AS name,

string\_field\_9 AS place,

string\_field\_10 AS tweet,

string\_field\_11 AS `language`,

string\_field\_12 AS mentions,

string\_field\_13 AS urls,

string\_field\_14 AS photos,

string\_field\_15 AS replies\_count,

string\_field\_16 AS retweets\_count,

string\_field\_17 AS likes\_count,

string\_field\_18 AS hashtags,

string\_field\_20 AS link,

string\_field\_23 AS video,

string\_field\_24 AS thumbnail,

string\_field\_31 AS reply\_to,

FROM `datacamp-287416.tweetsatmps\_month1.tweetsatmps\_mar23\_mar30`

# Remove header rows with WHERE

The data was created by merging 30 different .tsv files which each had their own header row. That means that we have 29 rows which repeat the header row, where the column ‘date’ contains the word ‘date’, and so on.

We add another line at the end to filter out rows where the ‘id’ field says ‘id’

SELECT string\_field\_0 AS tweet\_id,

string\_field\_1 AS conversation\_id,

string\_field\_2 AS created\_at,

string\_field\_3 AS `date`,

string\_field\_4 AS `time`,

string\_field\_5 AS timezone,

string\_field\_6 AS user\_id,

string\_field\_7 AS username,

string\_field\_8 AS name,

string\_field\_9 AS place,

string\_field\_10 AS tweet,

string\_field\_11 AS `language`,

string\_field\_12 AS mentions,

string\_field\_13 AS urls,

string\_field\_14 AS photos,

string\_field\_15 AS replies\_count,

string\_field\_16 AS retweets\_count,

string\_field\_17 AS likes\_count,

string\_field\_18 AS hashtags,

string\_field\_20 AS link,

string\_field\_23 AS video,

string\_field\_24 AS thumbnail,

string\_field\_31 AS reply\_to,

FROM `datacamp-287416.tweetsatmps\_month1.tweetsatmps\_mar23\_mar30`

WHERE string\_field\_0 != "id"

# Change field types with CAST (AS)

All of our fields are classed as strings. We could change that for numerical fields like ‘likes\_count’ by adding CAST like so:

CAST(string\_field\_15 AS int) AS replies\_count,

We could try to format datestamps as datetime, but it throws an error because of the inclusion of UTC in the datetime string:

CAST(string\_field\_2 AS datetime) AS created\_at,

And trying it with ‘date’ adds extra datetime elements that aren’t in the date (i.e. midnight timestamp).

The full code is now:

SELECT string\_field\_0 AS tweet\_id,

string\_field\_1 AS conversation\_id,

string\_field\_2 AS created\_at,

string\_field\_3 AS `date`,

string\_field\_4 AS `time`,

string\_field\_5 AS timezone,

string\_field\_6 AS user\_id,

string\_field\_7 AS username,

string\_field\_8 AS name,

string\_field\_9 AS place,

string\_field\_10 AS tweet,

string\_field\_11 AS `language`,

string\_field\_12 AS mentions,

string\_field\_13 AS urls,

string\_field\_14 AS photos,

CAST(string\_field\_15 AS int) AS replies\_count,

CAST(string\_field\_16 AS int) AS retweets\_count,

CAST(string\_field\_17 AS int) AS likes\_count,

string\_field\_18 AS hashtags,

string\_field\_20 AS link,

string\_field\_23 AS video,

string\_field\_24 AS thumbnail,

string\_field\_31 AS reply\_to,

FROM `datacamp-287416.tweetsatmps\_month1.tweetsatmps\_mar14\_mar23`

WHERE string\_field\_0 != "id"

# Change field types with CAST (AS) where the field names are mostly correct but field types are wrong

In the examples above the field names have been generated arbitrarily and the field types are all strings. However, in some cases the data is exported with the correct field names (apart from tweet\_id) and different field types - including ID numbers being treated as integers and dates as datetime fields.

To ensure those are consistent we need to CAST those as strings where needed to fit with other datasets. Here’s the query:

SELECT CAST (id AS string) AS tweet\_id,

CAST (conversation\_id AS string) AS conversation\_id,

CAST (created\_at AS string) AS created\_at,

CAST (date AS string) AS `date`,

CAST (time AS string) AS `time`,

CAST (timezone AS string) AS timezone,

CAST (user\_id AS string) AS user\_id,

username AS username,

name AS name,

place AS place,

tweet AS tweet,

language AS `language`,

mentions AS mentions,

urls AS urls,

photos AS photos,

replies\_count,

retweets\_count,

likes\_count,

hashtags,

link,

CAST (video AS string) AS video,

thumbnail,

reply\_to,

FROM `datacamp-287416.tweetsatmps\_month1.tweetsatmps\_apr13\_20`

# Joining the different scrapes with UNION ALL

To join multiple tables we used this query:

SELECT \*

FROM `datacamp-287416.tweetsatmps\_month1.tweetsatmps\_mar14\_mar23\_clean`

UNION ALL

SELECT \*

FROM `datacamp-287416.tweetsatmps\_month1.tweetsatmps\_mar23\_mar30\_clean`

Using UNION ALL means duplicates are retained, resulting in a dataset of 1,125,402 rows. To remove duplicates we change it to UNION DISTINCT.

SELECT \*

FROM `datacamp-287416.tweetsatmps\_month1.tweetsatmps\_mar14\_mar23\_clean`

UNION DISTINCT

SELECT \*

FROM `datacamp-287416.tweetsatmps\_month1.tweetsatmps\_mar23\_mar30\_clean`

That gives us 970,978 rows.

Merging 3 tables we use this query:

SELECT \*

FROM `datacamp-287416.tweetsatmps\_month1.tweetsatmps\_mar14\_mar23\_clean`

UNION DISTINCT

SELECT \*

FROM `datacamp-287416.tweetsatmps\_month1.tweetsatmps\_mar23\_mar30\_clean`

UNION DISTINCT

SELECT \*

FROM `datacamp-287416.tweetsatmps\_month1.tweetsatmps\_mar30\_apr06\_clean`

That results in a [table with 1,440,143 rows](https://console.cloud.google.com/bigquery?project=datacamp-287416&authuser=1&d=tweetsatmps_month1&p=datacamp-287416&t=tweetsatmps_mar14_apr06_DEDUPED&page=table&ws=!1m14!1m3!8m2!1s126394013532!2s71cfdebf6e6d410a92076b79fb45659f!1m4!4m3!1sdatacamp-287416!2stweetsatmps_month1!3stweetsatmps_mar14_mar23!1m4!4m3!1sdatacamp-287416!2stweetsatmps_month1!3stweetsatmps_mar14_apr06_DEDUPED).

Merging 4 tables we use this query:

SELECT \*

FROM `datacamp-287416.tweetsatmps\_month1.tweetsatmps\_mar14\_mar23\_clean`

UNION DISTINCT

SELECT \*

FROM `datacamp-287416.tweetsatmps\_month1.tweetsatmps\_mar23\_mar30\_clean`

UNION DISTINCT

SELECT \*

FROM `datacamp-287416.tweetsatmps\_month1.tweetsatmps\_mar30\_apr06\_clean`

UNION DISTINCT

SELECT \*

FROM `datacamp-287416.tweetsatmps\_month1.tweetsatmps\_apr06\_13\_clean`

That results in a [table with 2,080,441 rows](https://console.cloud.google.com/bigquery?project=datacamp-287416&authuser=1&ws=!1m23!1m4!4m3!1sdatacamp-287416!2stweetsatmps_month1!3stweetsatmps_apr06_13!1m3!8m2!1s126394013532!2s524ea721cf6345458d0fd29dd5043a2e!1m4!4m3!1sdatacamp-287416!2stweetsatmps_month1!3stweetsatmps_apr06_13_clean!1m3!8m2!1s126394013532!2sb560c9a8eb18475ba71113ce7b7401a5!1m4!4m3!1sdatacamp-287416!2stweetsatmps_month1!3stweetsatmps_mar14_apr13&d=tweetsatmps_month1&p=datacamp-287416&t=tweetsatmps_mar14_apr13&page=table).