Learning to query and read CrUX data

I’ve decided to take another look at BigQuery in the context of the Chrome User Experience Report or CrUX.

The idea is that Google, through the Chrome team and tools they make available to developers, has collected real user metrics (RUM) for billions of sites around the world and has grouped them according to country.

Warning

In the BigQuery free tier, users can only query 1TB worth of data per month. Beyond that, the standard rate of $5/TB applies. So when Big Query tells you how many gigabytes of data are processed in each request, pay attention.

This is why I picked a smaller country to experiment with rather than use the US or the full dataset… it can't get really expensive if you're not careful.

The first example uses BigQuery to search for all unique origins in Chile as of December 2019.

```
SELECT
  COUNT(DISTINCT origin)
FROM
  `chrome-ux-report.country_cl.201912`
```

let’s unpack the query referecing Rick Viscomi’s Using the Chrome UX Report on BigQuery

`SELECT COUNT(DISTINCT origin)` means querying for the number of unique origins in the table. Roughly speaking, two URLs are part of the same origin if they have the same scheme, host, and port.

`FROM chrome-ux-report.country_cl.201912` specifies the address of the source table, which has three parts:

- The Cloud project name chrome-ux-report within which all CrUX data is organized
- The dataset `country_cl`, representing data from Chile (county code CL)
- The table 201912, representing December 2019 in `YYYYMM` format

With slight modifications we can get a list of the URLs for the unique origins. We remove `COUNT` and leave the select statement as `DISTINCT origin`

```
SELECT
  DISTINCT origin
FROM
  `chrome-ux-report.country_cl.201912`
```

You can see a detailed preview of the data in BigQuery. Note that BigQuery is moving to the Google Cloud Console and that the URL may change as a result.

Of particular importance are the performance metrics available as part of the report:

- first_paint (FP)
- first_contentful_paint (FCP)
- dom_content_loaded (DCL)
- onload (OL)
- experimental.first_input_delay (FID)

The data for each metric is organized as an array of objects that we can capture as `[metric].histogram.bin`.

The following example will pick up the sum of all First Contentful Paint density values and assign them to the variable `fast_fcp`.

It will pull the data from the 2019/12 report and will flatten all the values in `first_contentful_paint.histogram.bin`

The matching conditions are: the origin has to be `https://www.vidasecurity.cl/` and the `fcp.start` value has to be greater than 10000, meaning that the site has to take 10 seconds or more to load.

```
SELECT
  SUM(fcp.density) AS fcp_density
FROM
  `chrome-ux-report.country_cl.201912`,
  UNNEST(first_contentful_paint.histogram.bin) AS fcp
WHERE
  origin = 'https://www.vidasecurity.cl/' AND
  fcp.start > 10000
```

The following example, unlike the other ones selects multiple elements from a specific table to get more fine grained information from the data.

This example asks the question: ***in the country_cl.201912 report, how many origins took more than 20 seconds (20000 milliseconds) to load on a phone . List the results by origin***

```
SELECT
  origin,
  fcp
FROM
  `chrome-ux-report.country_cl.201912`,
  UNNEST(first_contentful_paint.histogram.bin) AS fcp
WHERE
  form_factor.name = 'phone' AND
  fcp.start > 20000
ORDER BY
  origin
```

And this is the equivalent query for desktop connections taking more than 2000 seconds to start.

```
SELECT
  _TABLE_SUFFIX AS yyyymm,
  AVG(fcp.density) AS fast_fcp
FROM
  `chrome-ux-report.country_cl.*`,
  UNNEST(first_contentful_paint.histogram.bin) AS fcp
```

```
WHERE
  form_factor.name = 'desktop' AND
  fcp.start > 20000
GROUP BY
  yyyymm
ORDER BY
  yyyymm
```

With these two queries we can start making comparisons and predictions about the data before we jump in to more in depth data.

Some of the questions I had about the data:

- How much do numbers change over the years?
- Is there a significant difference between desktop and mmobile values?

Using these questions as a starting point we can dig deeper in the general data or query specific sites for more information. If we want, we can save the data as JSON and use a visualization library like D3 to generate graphical representations of the data or save it as CSV to manipulate on Excel or Google Sheets.

Once you've gotten all the answers you can get out of the CrUX dataset you can move to the HTTP Archive dataset. This dataset is a far more comprehensive both in the breadth of the data ir collects and the frequency that it collects the data.

For more infoormation on how to use the HTTP Archive BigQuery dataset see: Getting Started Accessing the HTTP Archive with BigQuery by Paul Calvano.