# Motivation

We will have two backfill related pipelines that are designed to run daily. [Here](https://docs.google.com/presentation/d/1rlT2L9XJtNWDsxYKHbyz8x6csaCSUU1PFC1uih7zUbM/edit?usp=sharing) is a link to the slides to briefly show the design of the two pipelines. In short, we have the backfill allerting pipeline to generate alerts to inform internal users whether the backfill update of a certain indicator is expected or not. The backfill correction pipeline is used to provide real-time backfill correction to the values of either counts or ratios. Both of the two pipelines require the input data in a certain format which is not available currently.

# Why not Fetch the data from the EpiData API?

The most ideal situation is that we are able to efficiently fetch all the as\_of data (report with all the correct versioning information). However, we are not able to do that due to multiple reasons:

* The `issue` in the EpiData API is not theoretically the issue\_date that we want for backfill correction. The current issue is defined more from the engineering side. For some indicators e.g. CHNG, there would be a one day gap between the issue date (when the data is theoretically available to Delphi) and the actual one (when the data is available to the API).
* The values reported in our API are usually smoothed or adjusted. However, the backfill related pipeline only needs the aggregated data from the raw.
* The backfill related pipeline needs counts data for the numerators and denominators of epidemiological values. For example, the numerators are the Covid claims counts and the denominators are the total claims counts if taking CHNG outpatient claims data into consideration. We only have ratios reported in the API, but not the counts (can be inferred if `sample\_size` is available).
* Fetching all the backfill data with a large number of versions can be expensive through API.

# Proposed solution

Since we already have daily production of the indicators that are backfill related, e.g. CHNG, DV, Quidel etc., we can modify the pipeline of these indicators to store the backfill data during the production. Take some indicators as examples:

* Quidel: Quidel can be the easiest one for this problem. The Quidel raw data is provided at line-level. And the pipeline is designed to aggregate the line-level raw data into zip-code level count data. In each run, the current version of update is stored as the intermediate file in the cache folder. The easiest modification can be:
  + Add the issue\_date column to the intermediate file
  + Store it and don’t overwrite it as what we currently do
* CHNG: the raw data for CHNG pipeline are the aggregated data in the CHNG SFTP server which are already at the count level. For each one, we only need to read the most recent input, have necessary preprocessing of it, and add a column to indicate the issue\_date and store it.

Since the backfill data can grow extremely fast in terms of its storage requirement, we might not want to actually `combine` all the data into one file. But we need to keep all the backfill data stored in a fixed format. *All the data should at lead have:*

* ***time\_value: the reference date***
* ***lag: the number of days between the issue date and the reference date, can be optional if issue\_date is available***
* ***Issue\_date: the issue/report date, can be optional if lag is available***
* ***geo\_value: a string or a number to indicate the location***
* ***a column to indicate the counts reported for the numerator***
* ***a column to indicate the counts reported for the denominator***