analytic_workflow_review

June 20, 2023

1 Analytic Workflow Review

So, I've done my best to go through and test as much of your script as I can. This is a pretty complicated workflow, plus it's missing a lot of the code for repeated parts of the logic, so I've only been able to properly dive into a small part of the whole picture. As a basic summary/TLDR of what follows, I think we should have a discussion about what we're trying to achieve here and re-format a lot of the process: it's quite hard to properly figure out exactly what we have here, and what I can figure out does seem to have a few issues which may lead to some problematic outputs.

I'm conscious that we're just trying to show end users what we have in our data, and have them poke holes in it, so you aren't too concerned about provenence/QA at this point. I'm still really struggling with the ethics/risks of a lot of this if I'm honest. In this particular case, we're doing a couple of big things that I think could lead to bad results:

1. We're defining cohorts i.e. "speech"/"vision"/"asd" cohorts
2. We're deriving our own observations from the data i.e. the concept of an encounter, when that first happened

The ONLY way to properly scrutinise and diagnose issues with these processes is to carefully scrutinise the logic generating the data that is feeding CLEVER, and the end users won't be doing any of that. As such we need to be certain that we understand properly the implications of any decisions we take when cutting up the data, and adding our own observations, and I'm not sure we do. For example:

- In defining a cohort, we're deciding what data NOT to show the end users. It's far less likely they can say with confidence that data is missing, or that our estimates for the number of people of type XXX or YYY are incorrect. Far more likely, they'll just trust that our logic stands, or the data is structured such that we're just representing information that is recorded at source, rather than manipulating it ourselves i.e. they'll think that "vision" or "speech" is coded for explicitly in the data, rather us generating those concepts ourselves
- It's very hard to detect small errors when aggregating data that has been merged/manipulated as much as this has - small errors are non-obvious in large datasets, but when we start joining data and estimating subsets/intersections of different groups, those errors are compounded and can be many orders of magnitude larger. The logic of showing this to pratitioners is that it's novel, so I don't follow the logic that they'll be wowed and super interested in all this exciting new information but, at the same time, be able to point out obvious issues with it because it's their bread and butter
- I don't think anybody in Connected Bradford is certain what any of the observations in the primary care data actually mean in a real sense - how they end up being entered on the system in the way they are - which observations are system generated - who enters what - how they

came to be organised the way they are. I'm worried that some problematic assumptions could lead to wild miscalculations in numbers of patients, times of encounters, demographics individuals belong to etc.

I really don't think these issues can be fixed by showing end users visualisations of our data. Those visualisations are backed up by a lot of moving and changing behind the scenes, and all that needs to be plainly laid out in logic to be able to identify errors in judgement there, and we need people capable of scrutinising that logic. I really struggled following a lot of the workflow (as you'll see below), so I'm not sure we have the former, and can't speak for the latter.

I'll leave my concerns there for now - particularly as writing really ins't my strong point, and the vast amount of unknowns involved in this project make it really hard for me to articulate all the worries I have about using this data! The following is the review I've done of your analytic workflow:

I've taken the word document you provided and re-formatted it so it can be run in a notebook. I try my hardest to encourage everybody to document their workflow this way, as a notebook is as expressive as a word document, but you can run code in it - it makes everything 10 times easier to follow and review/replicate if you have to hand the work over at any point.

I've put your comments/text in red italics and then added my own comments in regular text format. They're rough notes - I had hoped to write them up in full prose, but ran out of time. Hopefully they still make sense, but we can discuss later if not:

[8]:
```
import pandas as pd
```

1.1 *Context 1: Primary Care*

Goal*: Extract all GP visits that happened for CYP under 19 years old and find subsets in this cohort who have discussed any of the risk factors for Autism during those encounters*

Step 0*: Identify codes for the risk factor. Here we only select codes that have a valid_end_date within five years old. Older concepts are discarded for the purpose of this analysis.*

Example risk factor*: Speech and language*

[9]:
```
%%bigquery
CREATE TABLE `CB_1741_Relins.speech_codes` AS
SELECT * FROM `CB_CDM_VOCAB.concept`
WHERE (valid_end_date > CAST('2017-12-30' AS DATE)) AND
    (concept_name LIKE '%speech%' OR concept_name LIKE '%Speech%')
ORDER BY valid_end_date ASC
```

```
Executing query with job ID: ce0aee2a-e41a-46ee-9c56-8fbe57737848
Query executing: 0.36s


ERROR:
 409 Already Exists: Table yhcr-prd-phm-bia-core:CB_1741_Relins.speech_codes

Location: europe-west2
```

```
Job ID: ce0aee2a-e41a-46ee-9c56-8fbe57737848
```

[10]:
```
speech_codes_sql = """
SELECT * FROM `CB_1741_Relins.speech_codes`
"""
project="yhcr-prd-phm-bia-core"
speech_codes = pd.read_gbq(speech_codes_sql, project_id=project)
```

A lot of complexity is tied up in this first step. It isn't clear to me exactly what is being defined by "speech" (or indeed "vision") and I certainly don't have the expertise to criticise a codelist even if it were clear. I'm also not confident that doing a keyword search over the concept table is a good way of deriving a codelist - it leaves me with a lot of concerns that are hard to elaborate on here concicely but a flavour is:

- I don't know how thorough/reliable the descriptions in the concept dataset is - there could be many speech/vision related concepts that aren't properly described in the `concept_name`
- We're collecting together many different coding paradigms (SNOMED, ICD10 etc) that may not have been calibrated well, so we might be capturing a lot of poorly harmonised information
- The concept tables draw together diagnoses/observations/administrative information/procedures - all of which will have very different data generating processes, none of which are well understood or validated - I really worry that drawing together such disparate groups of information compounds the risk of making a lot of invalid assumptions

I think it would be good to explore other approaches. As far as I'm aware, the primary care data is all coded in SNOMED concepts, so we could use the opencodelists.org site to find curated condition lists and work from those - that would at least provide some air cover that we know what we're looking for in the code tables.

It might be easier to discuss this in person. For the moment, we'll assume the overall approach of keyword concept searching is valid and look at the data from there:

- There are an extra 29 codes when running the script over the new `concept` table in `CB_CDM_VOCAB`
- all of the "domains" are included, was this intentional? They are restricted to "Observation" and "Procedure" in the speech dataset

[5]:
```
pd.set_option('display.max_colwidth', None)
speech_codes.domain_id.value_counts()
```

[5]:
```
domain_id
Observation        558
Procedure          439
Condition          408
Device              43
Meas Value          40
Measurement         33
Note                27
Provider            17
Metadata             7
```

```
Revenue Code            7
Spec Anatomic Site      5
Type Concept            4
Visit                   3
Place of Service        2
Drug                    2
Name: count, dtype: int64
```

- some of the entries have an "invalid reason" which presumably means they shouldn't be included?

[6]:
```
speech_codes.invalid_reason.value_counts()
```

[6]:
```
invalid_reason
D    79
U     9
Name: count, dtype: int64
```

- some entries look like they don't relate to the subject's speech/language directly - might be good to see values for the numbers of observations taken using these codes - there may be large numbers of individuals included in the cohort under dubious codes - if we're using these codes as inclusion criteria for our cohort, we're suggesting we've figured out a "speech and language" baseline which I don't think we have

[56]:
```
concept_cols = ["concept_id", "domain_id", "concept_name"]
odd_codes = [4118542, 4140321, 3284050, 4173920, 37396617, 3423432]
speech_codes[concept_cols][speech_codes.concept_id.isin(odd_codes)]
```

[56]:
```
      concept_id    domain_id  \
559      4118542    Condition
748      4173920    Condition
848      3284050  Observation
925      3423432    Condition
1024    37396617    Condition
1381     4140321  Observation

                                    concept_name
559          Does not use velaric airstream for speech
748   Difficulty hearing speech in large group setting
848                                  No speech problem
925                          Uses speech to text reporter
1024                            Does not comprehend speech
1381              Referral to lip/speech-reading teacher
```

Similar step for eye and vision codes:

[46]:
```
%%bigquery
CREATE OR REPLACE TABLE `CB_1741_Relins.vision_codes` AS
```

```
SELECT * FROM `CB_CDM_VOCAB.concept`
WHERE (valid_end_date > CAST('2017-12-30' AS DATE)) AND
   ␣
  ↪(LOWER(concept_name) LIKE '%vision%' OR LOWER(concept_name) LIKE '%eye%') AND
    domain_id IN("Observation", "Procedure") AND
    LOWER(concept_name) NOT LIKE '%adult%' AND
    LOWER(concept_name) NOT LIKE '%revision%' AND
    LOWER(concept_name) NOT LIKE '%provision%' AND
    LOWER(concept_name) NOT LIKE '%supervision%' AND
    LOWER(concept_name) NOT LIKE '%division%'
ORDER BY valid_end_date ASC
```

```
/opt/conda/envs/ASDEnv/lib/python3.11/site-
packages/google/cloud/bigquery/job/query.py:1799: UserWarning: A progress bar
was requested, but there was an error loading the tqdm library. Please install
tqdm to use the progress bar functionality.
  query_result = wait_for_query(self, progress_bar_type,
max_results=max_results)
```

[46]:
```
Empty DataFrame
Columns: []
Index: []
```

[47]:
```
vision_codes_sql = """
SELECT * FROM `CB_1741_Relins.vision_codes`
"""
project="yhcr-prd-phm-bia-core"
vision_codes = pd.read_gbq(vision_codes_sql, project_id=project)
```

As I'm sure can be intuited, I have the same overall concerns as above. Again, assuming the approach is valid:

- there are another 315 codes using the new concept table
- more codes with an "invalid_reason"

[50]:
```
vision_codes.invalid_reason.value_counts()
```

[50]:
```
invalid_reason
D    232
U     68
Name: count, dtype: int64
```

- some of the codes might not make sense again, 1911 seem to relate to eyelids/eyebrows which isn't so much vision:

[65]:
```
problem_strings = ["eyelid", "eyebrow"]
problem_entries = (
    vision_codes.
    concept_name.
    apply(lambda x: any(string in x.lower() for string in problem_strings))
)
vision_codes[concept_cols][problem_entries]
```

[65]:

```
      concept_id  domain_id  \
18       4147871  Procedure
22       4198062  Procedure
34       4003904  Procedure
112     40601717  Procedure
125      3533462  Procedure
…            …          …
7439    42247407  Procedure
7442    42164072  Procedure
7443    42296859  Procedure
7444    42126315  Procedure
7445    42359136  Procedure

                                                                 concept_name
18
Recession of levator muscle of eyelid
22
Repair of eyelid retraction
34
Eyelid implantation
112
Repair of eyelid, full-thickness involving lid margin
125
Other operation on eyelid NOS
…
…
7439                                            Suture of Eyelid Laceration-
Through And Through,More than 12 months~ Under 72 months Nighttime second
surgery
7442  Excision of Eyelid Tumor-Benign,(Senior General Hospital, General
Hospital) Neonate Nighttime second surgery second surgery(General hospital or
higher)
7443
Suture of Eyelid Laceration-Simple,Nighttime second surgery
7444               Suture of Eyelid Laceration-Simple,(Senior General Hospital,
General Hospital) Under 12 months second surgery (General hospital or higher)
7445
Excision of Eyelid Tumor-Malignant

[1911 rows x 3 columns]
```

- lots of other codes relate to procedures on eyes that may not imply any issues with vision

```
[75]: odd_vision_entries = [3247127, 4218754, 3328617, 4144827, 4003369, 1005509,␣
       ↪35822480]
      vision_codes[concept_cols][vision_codes.concept_id.isin(odd_vision_entries)]
```

```
[75]:       concept_id    domain_id  \
      26       1005509  Observation
      848      3328617    Procedure
      920      4218754    Procedure
      2476     4003369    Procedure
      4886     4144827    Procedure
      5857    35822480  Observation
      6493     3247127    Procedure

      concept_name
      26                                                  PhenX measure - eye
      drop use
      848   Excision of secondary membrane of anterior segment of eye by pars plana
      approach
      920                                         Injection of posterior segment
      of eye
      2476                                                                  Eye
      incision
      4886                                                            Injection
      of eye
      5857                                                           polyfax eye
      ointment
      6493                                                              Excision
      of eye
```

- probably plenty more non-vision related codes in here
- more thought needs to go into what cohort's we're defining here.

***Step 1**: select a cohort of patients who have had this risk factor*

```
[20]: %%bigquery
      CREATE OR REPLACE TABLE `CB_1741_Relins.speech_person_codes` AS
      SELECT *  FROM `CB_1741_Relins.speech_codes` asd
      LEFT JOIN `CB_FDM_PrimaryCare_V7.tbl_srcode` p
      ON asd.concept_code = p.snomedcode
```

```
/opt/conda/envs/ASDEnv/lib/python3.11/site-
packages/google/cloud/bigquery/job/query.py:1799: UserWarning: A progress bar
was requested, but there was an error loading the tqdm library. Please install
tqdm to use the progress bar functionality.
  query_result = wait_for_query(self, progress_bar_type,
max_results=max_results)
```

[20]:
```
Empty DataFrame
Columns: []
Index: []
```

For vision:

[21]:
```
%%bigquery
CREATE OR REPLACE TABLE `CB_1741_Relins.vision_person_codes` AS
SELECT *  FROM `CB_1741_Relins.vision_codes` asd
LEFT JOIN `CB_FDM_PrimaryCare_V7.tbl_srcode` p
ON asd.concept_code = p.snomedcode
```

```
/opt/conda/envs/ASDEnv/lib/python3.11/site-
packages/google/cloud/bigquery/job/query.py:1799: UserWarning: A progress bar
was requested, but there was an error loading the tqdm library. Please install
tqdm to use the progress bar functionality.
  query_result = wait_for_query(self, progress_bar_type,
max_results=max_results)
```

[21]:
```
Empty DataFrame
Columns: []
Index: []
```

- logic of these two queries seems to work

For DCD:

[ ]:
```
%%bigquery
CREATE OR REPLACE TABLE `CY_1401_Elshehaly.dcd_person_codes` AS
SELECT *  FROM `CY_1401_Elshehaly.dcd_assessment_concepts_expanded` asd
LEFT JOIN `CY_FDM_PrimaryCare_V6.tbl_SRCode` p
ON asd.concept_code = p.src_snomedcode
```

- I don't have the script to create the `dcd_assessment_concepts_expanded` table so I'll leave this out for now

Step 2: *add person details so we can find out their age at the time of risk*

[22]:
```
%%bigquery
CREATE OR REPLACE TABLE `CB_1741_Relins.speech_person_codes_v2` AS
SELECT pc.*, p.birthplace, p.datebirth, p.ethnicity, p.gender,
    p.languagespoken, p.rowidentifier AS patientRowID, p.soa,
    p.speaksenglish, p.ward, p.spinematched
FROM `CB_1741_Relins.speech_person_codes` pc
LEFT JOIN `CB_FDM_PrimaryCare_V7.tbl_srpatient` p
ON pc.person_id = p.person_id
```

```
/opt/conda/envs/ASDEnv/lib/python3.11/site-
packages/google/cloud/bigquery/job/query.py:1799: UserWarning: A progress bar
was requested, but there was an error loading the tqdm library. Please install
```

```
tqdm to use the progress bar functionality.
  query_result = wait_for_query(self, progress_bar_type,
max_results=max_results)
```

[22]:
```
Empty DataFrame
Columns: []
Index: []
```

- The `srpatient` table contains multiple entries per person, which results in many duplicate records for each of the individuals in the cohort. The initial `speech_person_codes` table has 1,618,489 rows - this increases to 5,487,100 rows after the join to the `srpatient` table. Assuming the logic of this is to add individual details of the person to each record of a clinical code, rather than duplicating each entry multiple times
- This might be better done with the demographics table, but the provenence of that would require some investigation too

At this point we have over 271,000 unique person_id's who have encountered one or more of the speech codes.

- There are now 274k unique person_ids in the speech table with the new concepts and the V7 primary care dataset

Step 3*: now let's filter by age at time of recording – only those records entered for young people (when they were younger than 19 years old) are kept*

[24]:
```
%%bigquery
CREATE OR REPLACE TABLE `CB_1741_Relins.speech_person_codes_v3` AS
SELECT *
FROM `CB_1741_Relins.speech_person_codes_v2` pc
WHERE DATE_DIFF(pc.dateevent, CAST(pc.datebirth AS DATE FORMAT 'yyyymm'),␣
↪YEAR) < 19
```

```
/opt/conda/envs/ASDEnv/lib/python3.11/site-
packages/google/cloud/bigquery/job/query.py:1799: UserWarning: A progress bar
was requested, but there was an error loading the tqdm library. Please install
tqdm to use the progress bar functionality.
  query_result = wait_for_query(self, progress_bar_type,
max_results=max_results)
```

[24]:
```
Empty DataFrame
Columns: []
Index: []
```

- logic of the SQL seems to work

At this point we have over 237,000 unique person_id's which means a majority of the records were actually for CYP $<$ 19 years old.

- new datasets result in ~239K unique person_ids < 19 y/o

Step 4*: Create a EHR for each person in this cohort*

- this seems to be a bit strange - we just restricted the cohort to people who were under the age of 19 when a speech-related code was added to their health records. Now we're collecting the entire health record for all these individuals, so many of the entries in this table will be codes added after they were 19. I'm a bit confused by these inclusion criteria.

```
[ ]: %%bigquery
CREATE OR REPLACE TABLE `CB_1741_Relins.speech_person_ehr` AS
SELECT recs.*
FROM (
    SELECT DISTINCT person_id
    FROM `CB_1741_Relins.speech_person_codes_v3`
) AS persons
LEFT JOIN `CB_FDM_PrimaryCare_V7.tbl_srcode` recs
ON persons.person_id = recs.person_id
```

```
/opt/conda/envs/ASDEnv/lib/python3.11/site-
packages/google/cloud/bigquery/job/query.py:1799: UserWarning: A progress bar
was requested, but there was an error loading the tqdm library. Please install
tqdm to use the progress bar functionality.
  query_result = wait_for_query(self, progress_bar_type,
max_results=max_results)
```

```
[ ]: Empty DataFrame
Columns: []
Index: []
```

- logic of the SQL seems to work

***Step 4b**: person details were lost in the query above, so we restore them here*

```
[11]: %%bigquery
CREATE OR REPLACE TABLE `CB_1741_Relins.speech_person_ehr_v2` AS
SELECT ehr.*, person.concept_name, person.soa, person.speaksenglish,
    person.ethnicity, person.gender, person.birthplace,
    person.datebirth, person.languagespoken, person.ward
FROM `CB_1741_Relins.speech_person_ehr` ehr
LEFT JOIN `CB_1741_Relins.speech_person_codes_v3` person
ON ehr.person_id = person.person_id
```

```
/opt/conda/envs/ASDEnv/lib/python3.11/site-
packages/google/cloud/bigquery/job/query.py:1799: UserWarning: A progress bar
was requested, but there was an error loading the tqdm library. Please install
tqdm to use the progress bar functionality.
  query_result = wait_for_query(self, progress_bar_type,
max_results=max_results)
```

```
[11]: Empty DataFrame
Columns: []
Index: []
```

- this duplicates every row in the original `speech_person_ehr` table over every row in the `speech_person_codes` table. The result is going from a ~107M row table to a 3 Billion row table!

***Step 5a**: Find the first encounter of the risk factor*

- isn't clear what is meant by this - first encounter of ANY of the codes identified in each of the "person_codes" tables? First encounter FOR EACH of the codes encoundered in the "person_codes" tables?

```
[5]: %%bigquery
CREATE TABLE `CB_1741_Relins.speech_person_first_encounter` AS
SELECT DISTINCT person_id, snomedcode, ctv3text, idvisit,
    MIN(dateeventrecorded) AS first_encountered
FROM `CB_1741_Relins.speech_person_ehr_v2` ehr
WHERE snomedcode IN (
    SELECT DISTINCT snomedcode
    FROM `CB_1741_Relins.speech_person_codes_v3`
)
GROUP BY person_id, snomedcode, idvisit, ctv3text
```

```
/opt/conda/envs/ASDEnv/lib/python3.11/site-
packages/google/cloud/bigquery/job/query.py:1799: UserWarning: A progress bar
was requested, but there was an error loading the tqdm library. Please install
tqdm to use the progress bar functionality.
  query_result = wait_for_query(self, progress_bar_type,
max_results=max_results)
```

```
[5]: Empty DataFrame
Columns: []
Index: []
```

- The `DISTINCT` statement isn't necessary as the query is already grouped by all the selected variables (if you remove the `DISTINCT` you get exactly the same result) - also not advisable to use a `DISTINCT` statement over multiple variables with grouping too - it's prone to errors
- I'm not sure of the logic here - prior to this point we've been defining the "risk factor" as ANY of the numerous clinical codes in the "codes" tables. This query finds the earliest event record for ALL the unique combinations of person_id, snomedcode, ctv3text and idvisit. So for every individual, we'll potentially have multiple entries with the earliest date for every code recorded for each unique visit in their records. This doesn't make sense to me. Particularly grouping by visits - presumably you'll just end up with the same dates as you had in the original table as most visits won't span multiple dates
- My intuition is that we're looking for the earliest record for each individual of one of the (in this example) speech "codes", which is a much simpler query
- Probably need to discuss this as I can't really guess at what's supposed to be happening here - and given the complexity of the query and the size of the table, it's even harder to figure out what's going on with the results and if they're to be expected or not

***Step 5b**: attach this information back to the EHR table (EDITED workflow here)*

[7]:
```
%%bigquery
CREATE OR REPLACE TABLE `CB_1741_Relins.speech_person_ehr_v3` AS
SELECT ehr.*, f.first_encountered
FROM `CB_1741_Relins.speech_person_ehr_v2` ehr
LEFT JOIN `CB_1741_Relins.speech_person_first_encounter` f
ON ehr.person_id = f.person_id
```

```
/opt/conda/envs/ASDEnv/lib/python3.11/site-
packages/google/cloud/bigquery/job/query.py:1799: UserWarning: A progress bar
was requested, but there was an error loading the tqdm library. Please install
tqdm to use the progress bar functionality.
  query_result = wait_for_query(self, progress_bar_type,
max_results=max_results)
```

[7]:
```
Empty DataFrame
Columns: []
Index: []
```

- again, there are several person_ids in the `speech_person_first_encouter` table with multiple associated records, so this query is creating a duplicate of the entire record for each person_id the `speech_person_ehr` for every duplicate entry in the `speech_person_first_encounter` table. The result is a 3 billion row table becomming a 6.5 billion row table, which I'm guessing certainly isn't intended.
- It's hard to suggest a fix for this without knowing what supposed to be happening with the "first encounter" variable, but its likely we just want to join by both person_id and the other features were grouping over in the "first encounter" query

***EDIT**: Attach the first diagnosed variable to the person_codes table because I ended up using it in the dashboard instead of the EHR one. NB: We still need to attach this to EHR so we can trace referrals after first encounter, etc.*

[9]:
```
%%bigquery
CREATE OR REPLACE TABLE `CB_1741_Relins.speech_person_codes_v4` AS
SELECT ehr.*, f.first_encountered
FROM `CB_1741_Relins.speech_person_codes_v3` ehr
LEFT JOIN `CB_1741_Relins.speech_person_first_encounter` f
ON ehr.person_id = f.person_id
```

```
/opt/conda/envs/ASDEnv/lib/python3.11/site-
packages/google/cloud/bigquery/job/query.py:1799: UserWarning: A progress bar
was requested, but there was an error loading the tqdm library. Please install
tqdm to use the progress bar functionality.
  query_result = wait_for_query(self, progress_bar_type,
max_results=max_results)
```

[9]:
```
Empty DataFrame
Columns: []
Index: []
```

- again this will create the same issue with duplicated records as above - 4M rows becomes 8M rows

***Step 6**: attach post codes (EDITED)*

[12]:
```
%%bigquery
CREATE OR REPLACE TABLE `CB_1741_Relins.speech_person_ehr_v4` AS
SELECT *
FROM `CB_1741_Relins.speech_person_ehr_v3` fd
LEFT JOIN `CB_1401_Elshehaly.lsoa_to_postcode` po
ON fd.soa = po.LSOA_code
```

```
/opt/conda/envs/ASDEnv/lib/python3.11/site-
packages/google/cloud/bigquery/job/query.py:1799: UserWarning: A progress bar
was requested, but there was an error loading the tqdm library. Please install
tqdm to use the progress bar functionality.
  query_result = wait_for_query(self, progress_bar_type,
max_results=max_results)
```

[12]:
```
Empty DataFrame
Columns: []
Index: []
```

- not sure how you can convert an LSOA to a postcode - LSOAs as I understand them potentially have many postcodes/postcode areas
- Likely as a result of the above comment, in several cases there are multiple entries per `LSOA_code` in the `lsoa_to_postcode` table - again, for all of these examples the rows with corresponding LSOAs will be duplicated the number of times over that `LSOA_code` appears in the `lsoa_to_postcode` table - 6.5 Billion rows becomes 10 billion rows

***EDIT**:*

[13]:
```
%%bigquery
CREATE OR REPLACE TABLE `CB_1741_Relins.speech_person_codes_v5` AS
SELECT *
FROM `CB_1741_Relins.speech_person_codes_v4` fd
LEFT JOIN `CB_1401_Elshehaly.lsoa_to_postcode` po
ON fd.soa = po.LSOA_code
```

```
/opt/conda/envs/ASDEnv/lib/python3.11/site-
packages/google/cloud/bigquery/job/query.py:1799: UserWarning: A progress bar
was requested, but there was an error loading the tqdm library. Please install
tqdm to use the progress bar functionality.
  query_result = wait_for_query(self, progress_bar_type,
max_results=max_results)
```

[13]:
```
Empty DataFrame
Columns: []
Index: []
```

- as per above 8M rows becomes 12M rows

***Step 7**: attach months elapsed between event and first encounter and also attach the person's age at visit (both are derived columns) (EDITED)*

[16]:
```
%%bigquery
CREATE OR REPLACE TABLE `CB_1741_Relins.speech_person_ehr_v5` AS
SELECT *, DATE_DIFF(dateevent,
                    CAST(datebirth AS DATE FORMAT 'yyyymm'),
                    YEAR) AS age_at_encounter,
          DATE_DIFF(dateevent,
                    first_encountered,
                    MONTH) AS months_elapsed
FROM `CB_1741_Relins.speech_person_ehr_v4` ehr
```

```
/opt/conda/envs/ASDEnv/lib/python3.11/site-
packages/google/cloud/bigquery/job/query.py:1799: UserWarning: A progress bar
was requested, but there was an error loading the tqdm library. Please install
tqdm to use the progress bar functionality.
  query_result = wait_for_query(self, progress_bar_type,
max_results=max_results)
```

[16]:
```
Empty DataFrame
Columns: []
Index: []
```

- `age_at_encounter` seems fine logic wise
- the `months_elapsed` variable exemplifes the issues above with the `first_encountered` logic - several values are negative - as said above, needs some more thought about what information we're tyring to capture here

***EDIT**:*

[2]:
```
%%bigquery
CREATE OR REPLACE TABLE `CB_1741_Relins.speech_person_codes_v6` AS
SELECT *, DATE_DIFF(dateevent,
                    CAST(datebirth AS DATE FORMAT 'yyyymm'),
                    YEAR) AS age_at_encounter,
          DATE_DIFF(dateevent,
                    first_encountered,
                    MONTH) AS months_elapsed
FROM `CB_1741_Relins.speech_person_codes_v5` ehr
```

```
/opt/conda/envs/ASDEnv/lib/python3.11/site-
packages/google/cloud/bigquery/job/query.py:1799: UserWarning: A progress bar
was requested, but there was an error loading the tqdm library. Please install
tqdm to use the progress bar functionality.
  query_result = wait_for_query(self, progress_bar_type,
max_results=max_results)
```

[2]:
```
Empty DataFrame
Columns: []
Index: []
```

- same as above

***Step 8**: Calculating summaries for the main CLEVER panel:*

calculate the proportion of people in this cohort who also exist in the milestone cohort. In this case, the milestone is Autism diagnosis

[ ]:
```
%%bigquery
SELECT DISTINCT person_id
FROM `CY_1401_Elshehaly.speech_person_first_encounter` sp
WHERE EXISTS (
    SELECT DISTINCT person_id
    FROM `CY_1401_Elshehaly.asd_person_first_diagnosis` asd
    WHERE sp.person_id = asd.person_id
)
```

Results in 4625

- I don't have any detials of how the `asd_person_first_diagnosis` is generated so it's hard to say exactly what's going on here
- The logic works here in terms of "count the number of people in this table that appear in that table", but as per above comments on the `asd_person_first_diagnosis`, I don't know if this is a sensible cohort to be summarising

calculate the proportion of people in this cohort who have had a referral following the first encounter event (this will include all referrals but let's see if we can pursue further filtering to see if any of the referrals were relevant to autism)

[ ]:
```
%%bigquery
SELECT DISTINCT person_id
FROM `CY_1401_Elshehaly.speech_person_ehr` ehr
WHERE (src_ctv3text LIKE '%referral%' OR src_ctv3text LIKE '%Referral%')
AND DATE_DIFF(src_dateevent, first_encountered, MONTH) < 6
```

- issues with `first_encountered` carry forward
- will take some work to validate all the codes that are pulled out of the health records when searching for "referral" - don't have time for that at the moment but will be something to consider later

The rest of the workflow I can't replicate without the code for the `social_person_codes` or without spending a lot of time reverse engineering the workflow, which would probably not be a good use of time. Best to discuss a way forward before we do any more work validating.

***NOT INCLUDED HERE**: is a repetition of the above workflow to create a table social_person_codes which holds CYP records for those with social and behavioural concerns.*

***STEP 10**: Merge both tables:*

First let's add a label for each table to know where each record originates

[7]:
```
%%bigquery
CREATE OR REPLACE TABLE `CY_1401_Elshehaly.social_person_codes` AS
SELECT pc.*, ifnull(tbl_src, "social") AS table_source
FROM `CY_1401_Elshehaly.social_person_codes` pc
```

```
UsageError: Cell magic `%%bigquery` not found.
```

do the same thing for each table, adding the corresponding label to the table_source field.

[ ]:
```
INSERT INTO `CY_1401_Elshehaly.speech_and_social_person_codes` (
    person_id, concept_id, concept_code, concept_name, src_rowidentifier,
   ␣
  ↪src_dateevent, src_ctv3code, src_ctv3text, src_snomedcode, src_idorganisation,
    src_idevent,␣
  ↪src_idreferralin, src_idappointment, src_idvisit, care_site_id,
    provider_id, procedure_source_concept_id,  observation_source_concept_id,
    measurement_source_concept_id, src_soa,  src_speaksenglish,  src_ethnicity,
    src_gender, src_birthplace, src_datebirth, src_languagespoken, src_ward,
    first_encountered, LSOA_code, postcode, age_at_encounter, months_elapsed,
    table_source
)
SELECT person_id, concept_id, concept_code, concept_name, src_rowidentifier,
   ␣
  ↪src_dateevent, src_ctv3code, src_ctv3text, src_snomedcode, src_idorganisation,
    src_idevent,␣
  ↪src_idreferralin, src_idappointment, src_idvisit, care_site_id,
    provider_id, procedure_source_concept_id,  observation_source_concept_id,
    measurement_source_concept_id, src_soa,  src_speaksenglish,  src_ethnicity,
    src_gender, src_birthplace, src_datebirth, src_languagespoken, src_ward,
    first_encountered, LSOA_code, postcode, age_at_encounter, months_elapsed,
    table_source
FROM `CY_1401_Elshehaly.speech_person_codes`
```