**Day 1:**

I can't lie but while I still really look forward to finding a job as I graduate university I can't help but not be bothered by recruiter’s phone interview calls doing this project, I'm currently doing in data analytics. Long story short I aspired to become a Machine Learning engineer one day but as semesters and personal projects went by, trying to constantly build and learn just eventually became difficult, and might've contributed to my eventual burnout. It's been a quite a journey for me transitioning from the field albeit not entirely considered an expert at it, I'm still grateful that I can use all that I've learned from it to transition into data analytics which I didn't know I would come to eventually enjoy primarily because I like writing in python, but more than that SQL (I don't know querying data using typical DML statements to display these queries in a creative and artful dashboard seemed something I could do all day every day).   
  
So I'm working with this healthcare dataset about chronic disease in states in the US with the goal of not only generating meaningful insights but to create artful visual representations (I chose PowerBI since the visuals were nicer looking) and just to feel like a kid again in the joy of learning new things and creating.  
  
So here's my day 1 in trying to learn data analytics using the Python, MSSQL, PowerBI, and Excel tech stack. I'll be documenting this project of mine from start to finish on LinkedIn so if you have any constructive feedback/criticism that can be used to improve this project, feel free to comment down below (just go easy haha marami pa akong bigas na kakainin kumbaga).

**2nd update:**

TLDR: I’ve learned that the main `chronic disease indicators` dataset from <https://www.kaggle.com/datasets/irakozekelly/u-s-chronic-disease-indicators-2023-release?resource=download> just wasn’t enough, and pulling more population data from other sources made sense, but funnily enough also needed more cleaning and some sort data modelling as it was still in a spreadsheet that was difficult to query using tools like SQL.

Going through much cleaning and preprocessing the dataset had rows with attribute/column values like these

```

yearstart: 2012

yearend: 2016

locationdesc: connecticut

question: cancer of the lung and bronchus mortality

datavalueunit: cases per 100000

datavaluetype: average annual crude rate

datavalue: 9.6

stratificationcategory: race/ethnicity

stratification: asian or pacific islander

```

where this can be interpreted as the `average annual crude rate of cancer of the lung and bronchus mortality from 2012 to 2016 in the state of connecticut for an asian or pacific islander is 9.6 cases per 100000`. Another example:

```

yearstart: 2015

yearend: 2015

locationdesc: florida

question: Hospitalization for chronic obstructive pulmonary disease as any diagnosis

datavalueunit: cases per 10000

datavaluetype: crude rate

datavalue: 9.6

stratificationcategory: race/ethnicity

stratification: asian or pacific islander

```

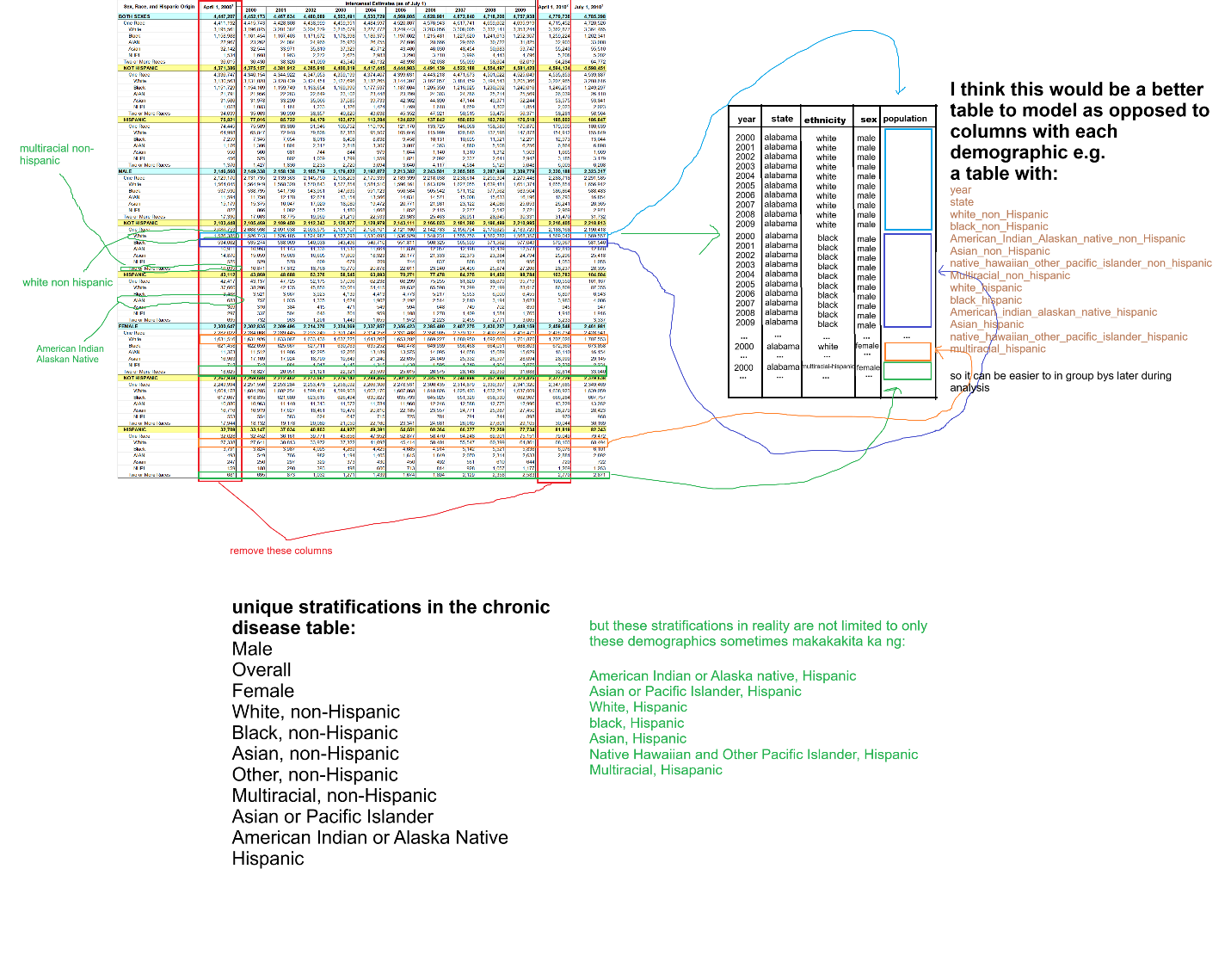
interpretation: `crude rate of for hospitalization for chronic obstructive pulmonary disease as any diagnosis from 2015 to 2015 in the state of florida for the male population was 420 cases per 10000`

I thought to myself that this if different rows of the data had different data value units and different data value types how then was, I supposed to make some analyses if the values to begin with were incomparable. So, I asked Gemini how was I going to make these rows have tangible numbers to work with particularly how was I going to calculate perhaps the total cases of a chronic disease indicator (CDI) on a population scale? And the answer was to pull the total population values of particularly each state per year of the US from 2001 to 2021.

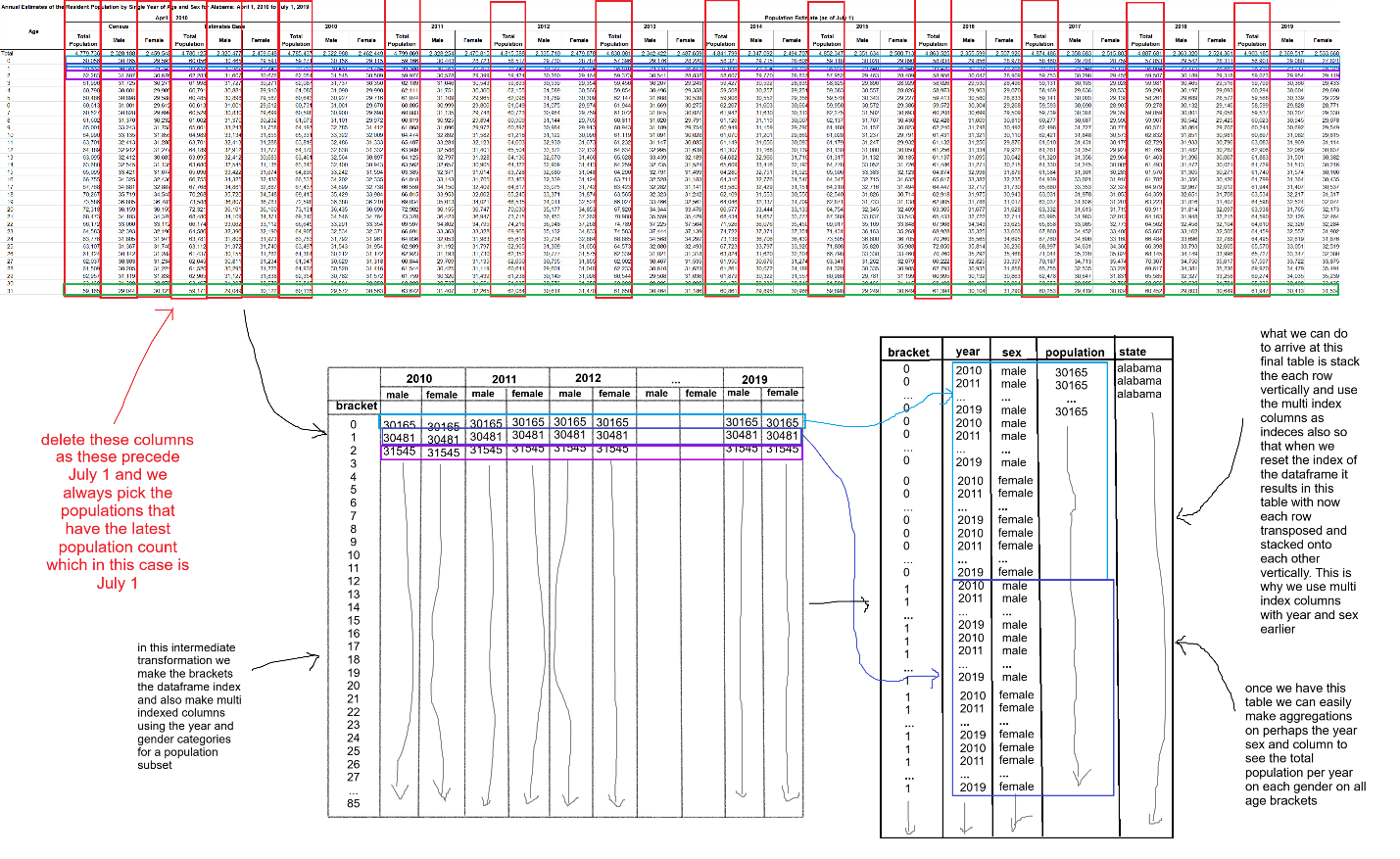
So that I did but somehow, I saw that there were still problems like stratifications I had to deal with; using the total population for all ethnicities for both sexes wasn’t enough, because what if the CDI had `current asthma prevalence among male adults aged >= 18 years` and a stratification of `asian`? Then using the total population is not enough as this CDI entails that the number was measured was with a demographic only of the male population aged 18 and above and on top of that were only Asian.

Initially I thought removing the rows with a stratification of race/ethnicity but I found that discarding roughly 500000+ rows out of 600000+ total rows seemed to waste too much data points, and that keeping the remaining data points which only had a stratification of gender seemed to be easier since I already extracted other data pertaining to the population with stratification of gender and their respective age brackets. But doing this seemed to be an injustice to the data and doing analysis with little data would be a waste. So, I did my best to collate more population data for each us state that included stratifications of all races and ethnicities.

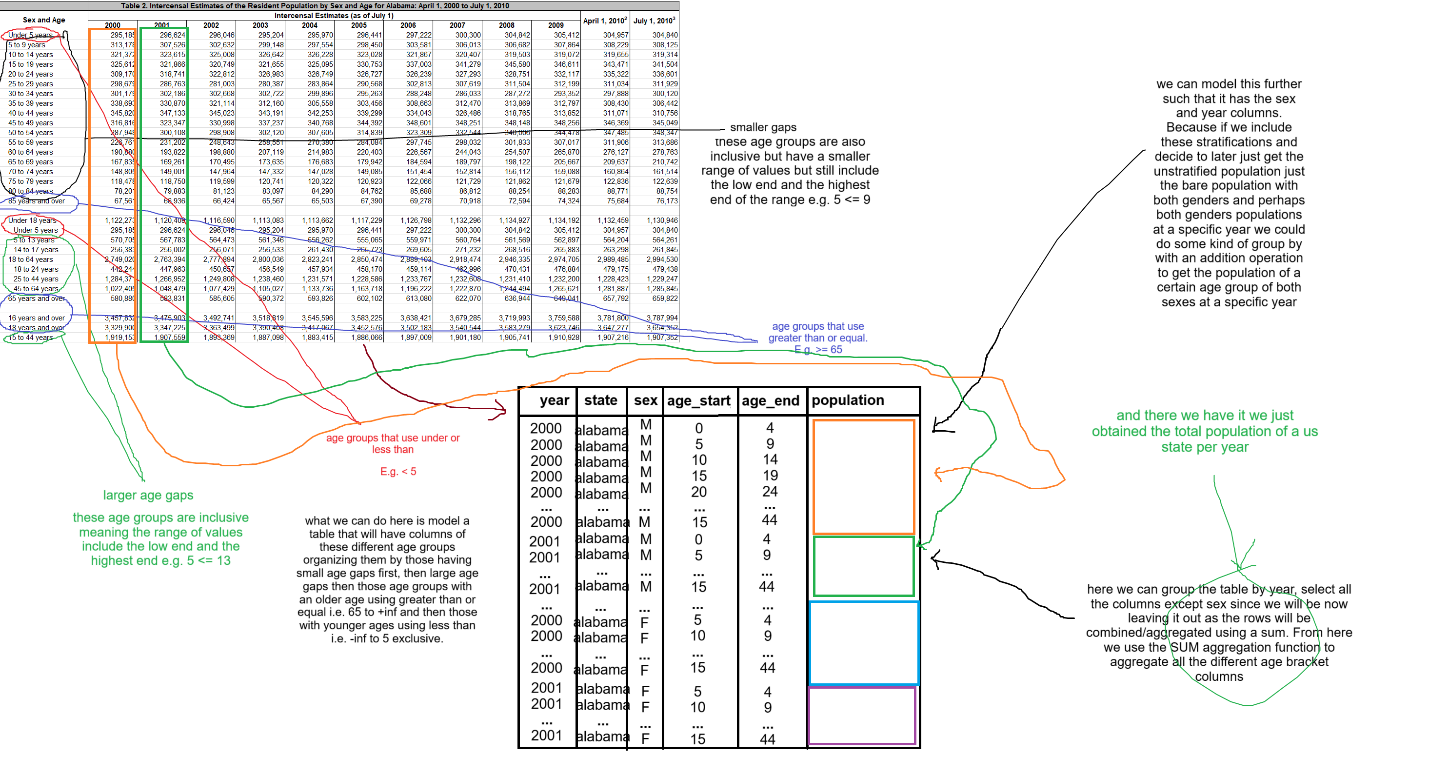
But the hard part wasn’t this surprisingly, the hard part was transforming the spreadsheets into a format that was not only readable but also easier to query and make some sort of aggregation so that when some operation like summing the population arises it can be easily done through tools like SQL. How I thought of doing the process of somewhat modeling the data from spreadsheet to a SQL table is detailed in the pictures below.



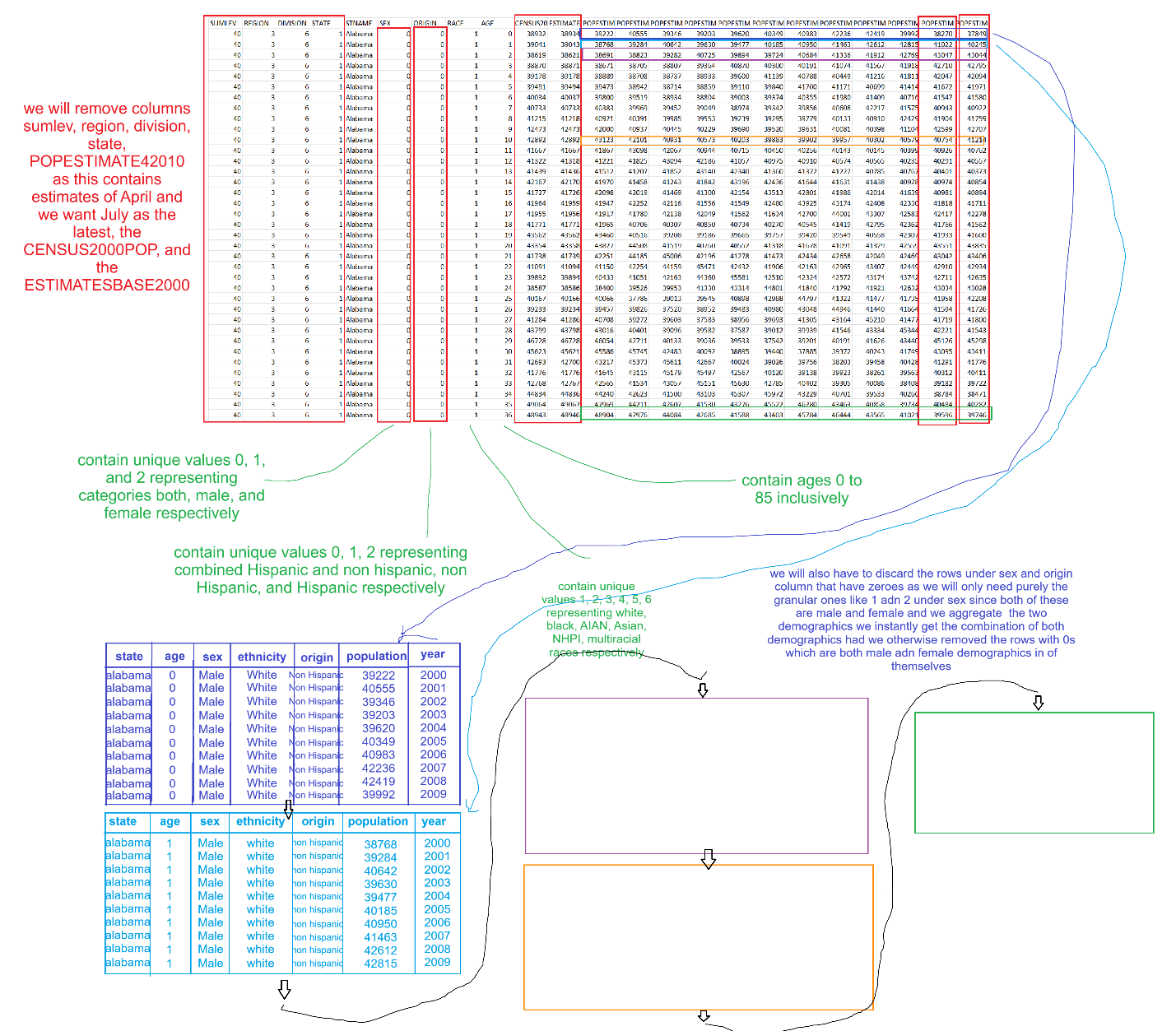
*modelling table from population data by sex race and ethnicity 2000 to 2009*



*modelling table from population data by sex and age 2010 to 2019*



*modelling table from population data by sex and age 2000 to 2009*



*modelling table from population data by sex age race and ethnicity 2000 to 2009*

**3rd Update**

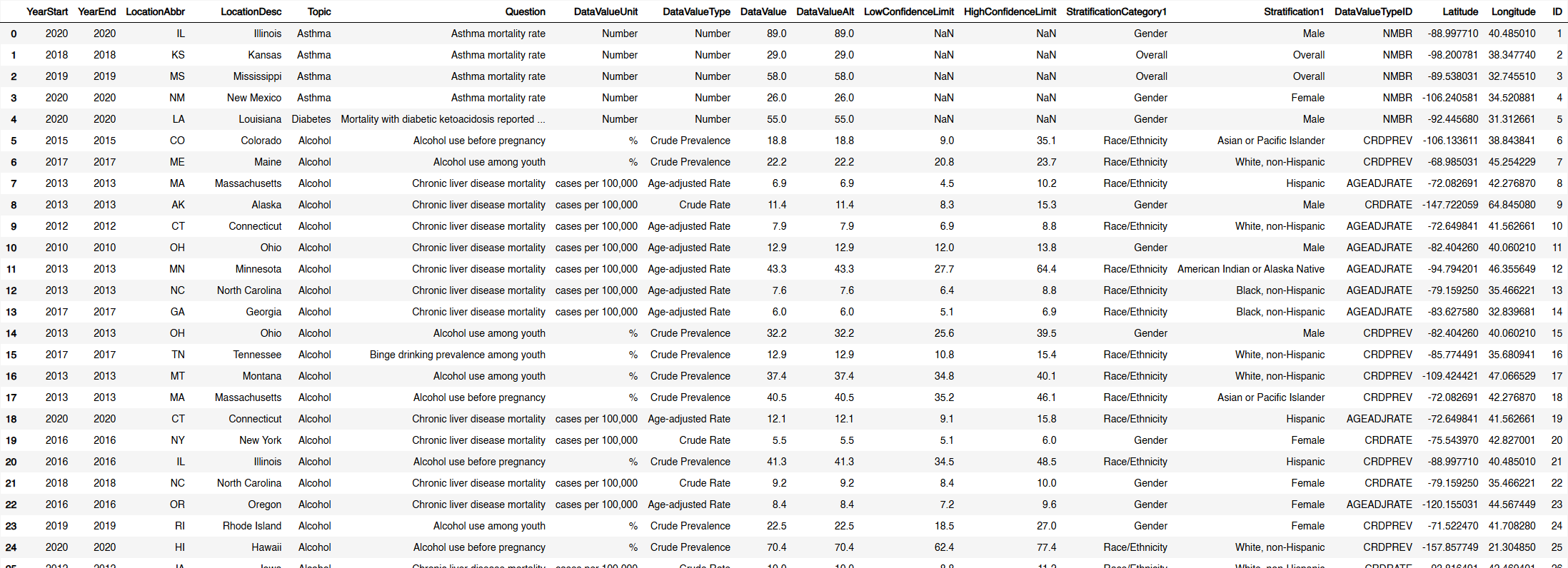
TLDR: I never thought processing data could be much more fun than training statistical models, now after doing some sort of initial modelling using pandas (images below) I'm trying to move to faster processing these same tables and the 1m datapoints from the CDI dataset using distributed computing engines like Apache-Spark (PySpark).   
  
But again if you caught my previous post basically the problem was data having difficult to interpret numbers and so I thought why not collate extra data in order to calculate a more tangible number example below...  
  
In the chronic-disease-indicators (CDI) dataset had rows with attribute/column values more or less like these  
```  
yearstart: 2012  
yearend: 2016  
locationdesc: connecticut  
question: cancer of the lung and bronchus mortality  
datavalueunit: cases per 100000  
datavaluetype: average annual crude rate  
datavalue: 9.6  
stratificationcategory: race/ethnicity  
stratification: asian or pacific islander  
```  
  
where this can be interpreted as the `average annual crude rate of cancer of the lung and bronchus mortality from 2012 to 2016 in the state of connecticut for an asian or pacific islander is 9.6 cases per 100000`. However again this isn't really useful as there isn't a tangible number we could touch on to differentiate the many datapoints of this CDI dataset.

What I thought however was that certain calculations could be made such that we can extract the total number of recorded cases for a specific state at a specific year for a specific stratification for a persons specific age bracket. And to do this I had the tables modelled on the extra data I extracted to get the total population for these features and use it to calculate the total number of cases. E.g. total population of a pacific islander, with an age bracket of 0 to 85 and above, in Connecticut, from year 2012 to 2016, is let's just say in this case 18,283,832 which I knew could be calculated using group by's, aggregations, and filtering clauses in SQL.

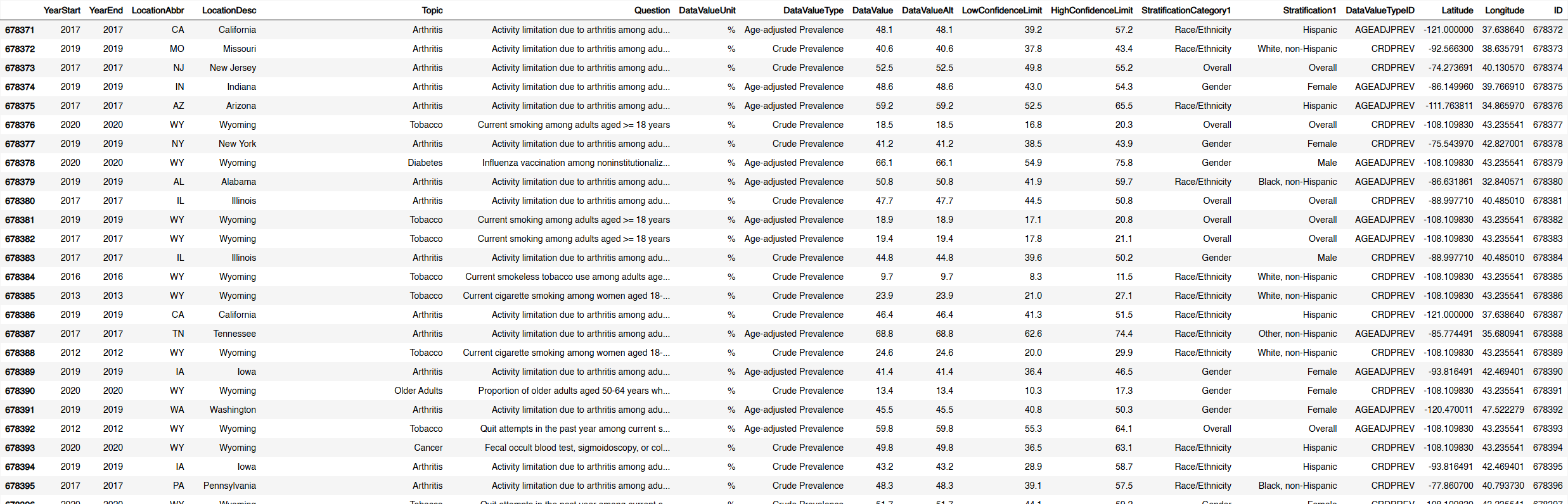
Using this number we can calculate the tangible number of cases/prevelance/occurrence of a chronic disease indicator like mortality of the cancer of the lung and bronchus using the following formula

In this case the approximate number of cases or those who died from lung and bronchus related cancers was 1755.

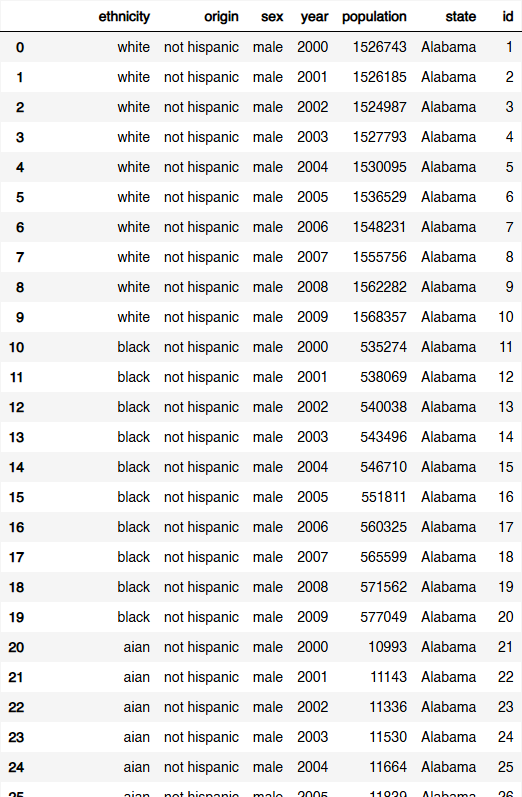
This is in part I thought of modelling these extra data so that querying using SQL could easily be done when the tables were finally uploaded to some data warehouse. And having recently learned about OLAPs and OLTPs as per feedback from fellow connections here 😊, as OLAPs and OLAP cubes store data and features in a manner that could be used for quick slicing and aggregations for data analysis, I thought this was an apt situation for this problem.



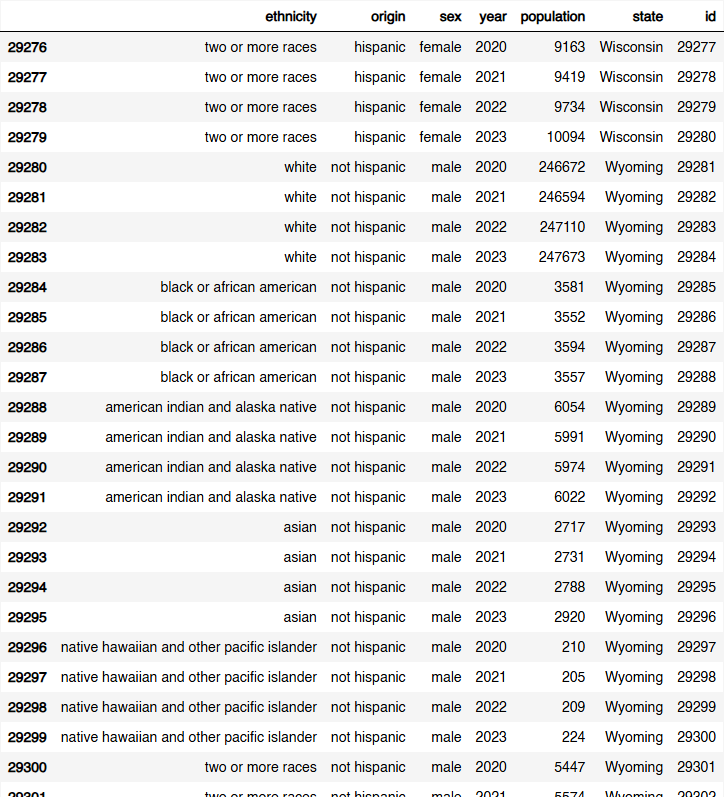
*CDI dataset head*



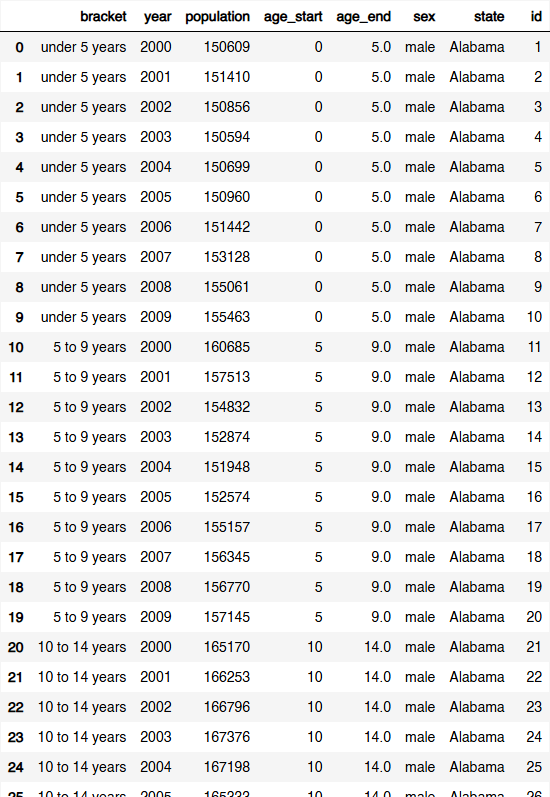
*CDI dataset tail*



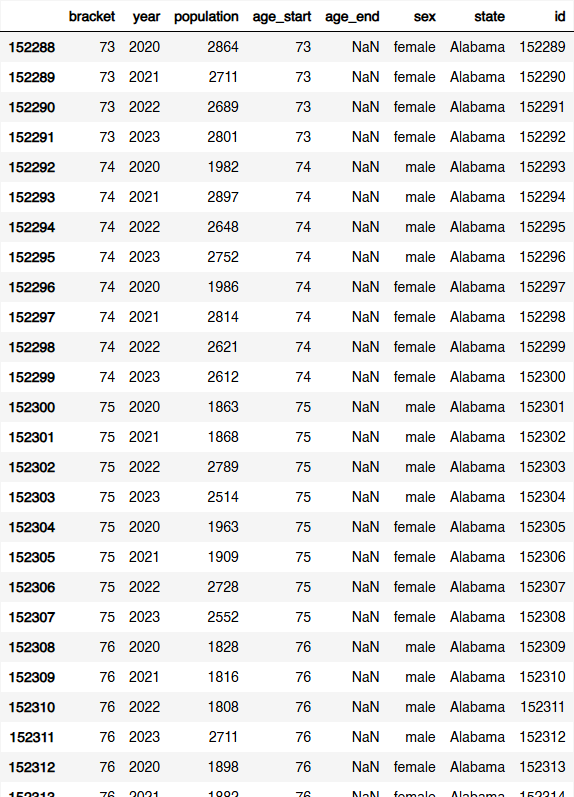
*State populations by sex, race, and origin 2000 to 2023 head*



*State populations by sex, race, and origin 2000 to 2023 tail*



*State populations by sex and race 2000 to 2023 head*



*State populations by sex and race 2000 to 2023 tail*

**4th Update:**

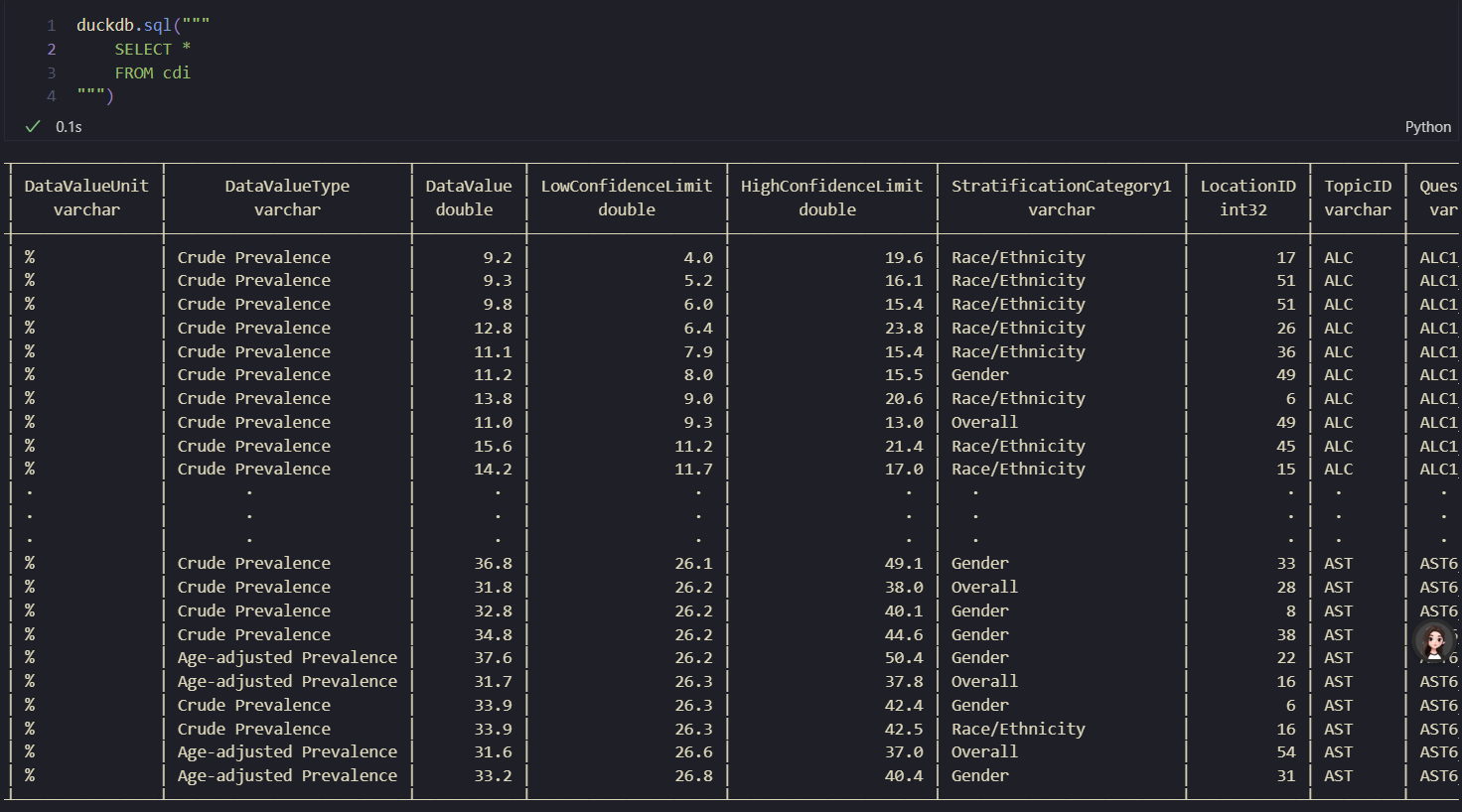
I really learned a lot especially when it came to using apache spark for transformations as opposed to pandas previously, such as using the right configurations for the spark cluster (how many workers and their memory size) I was going to submit the transformation script to, using external jar packages to process excel files which was personally the hardest part of this, and as much as possible not stupidly collecting all the spark dataframes in a list and then concatenating them all at once resulting in an `out of memory` error 😅.

My reasoning for all this unnecessary hard work? That I was going to use spark anyway, one way or another in future corporate work, so I might as well suffer now and learn the hard (but fun) way 😊.

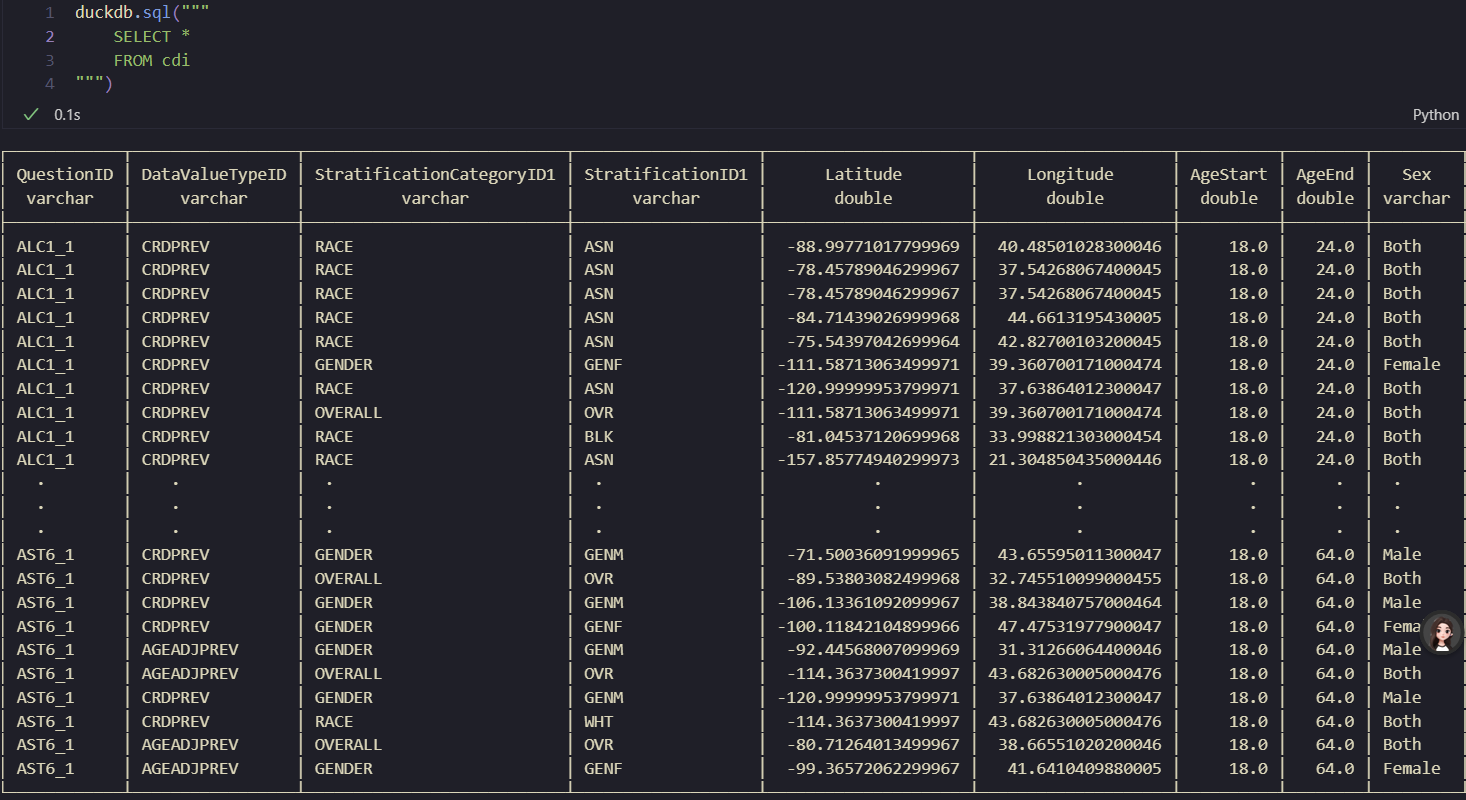
But here it is now: basically right after saving the partitioned dataframes as .parquet files (which I researched was actually a format faster in reading and writing than .csv). I now loaded them to an in-process open source OLAP data warehouse called DuckDB which I recently also learned also was basically an equivalent of the OLTP DB sqlite. I figured also I couldn't pay for DWHs like snowflake or databricks so I thought of using free alternatives like this instead.

Although this probably isn't modelled properly yet I'll be posting again on a more updated schema on these tables, so let me know what columns I should probably place on a separate dimension table I'd be happy for the feedback.

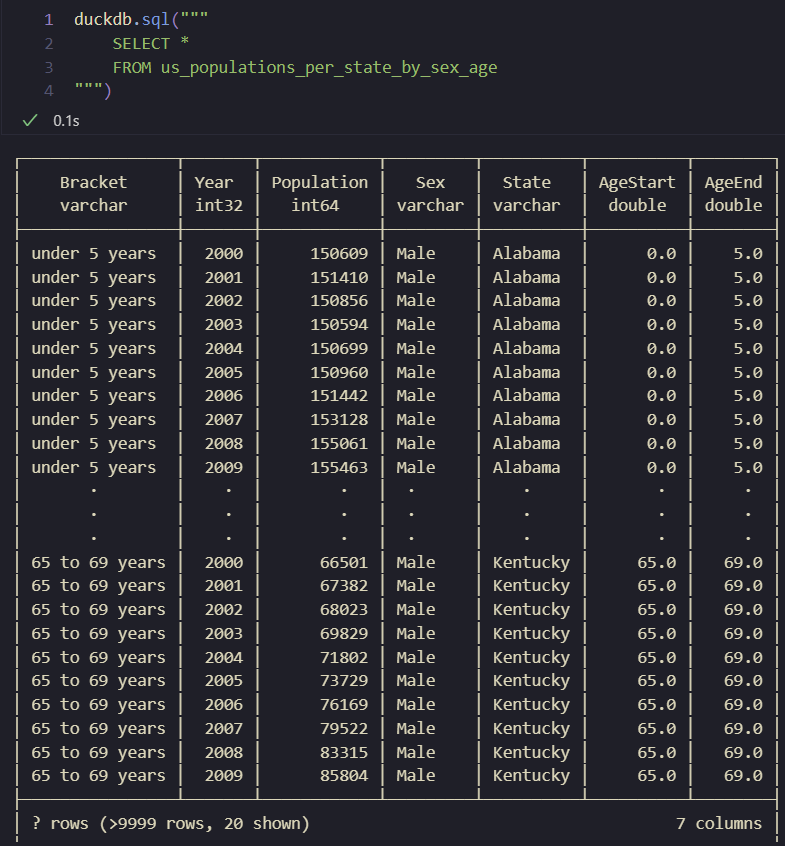
So now I ought to do some initial analytics on these tables using SQL and post here again hopefully with a dashboard using PowerBI.



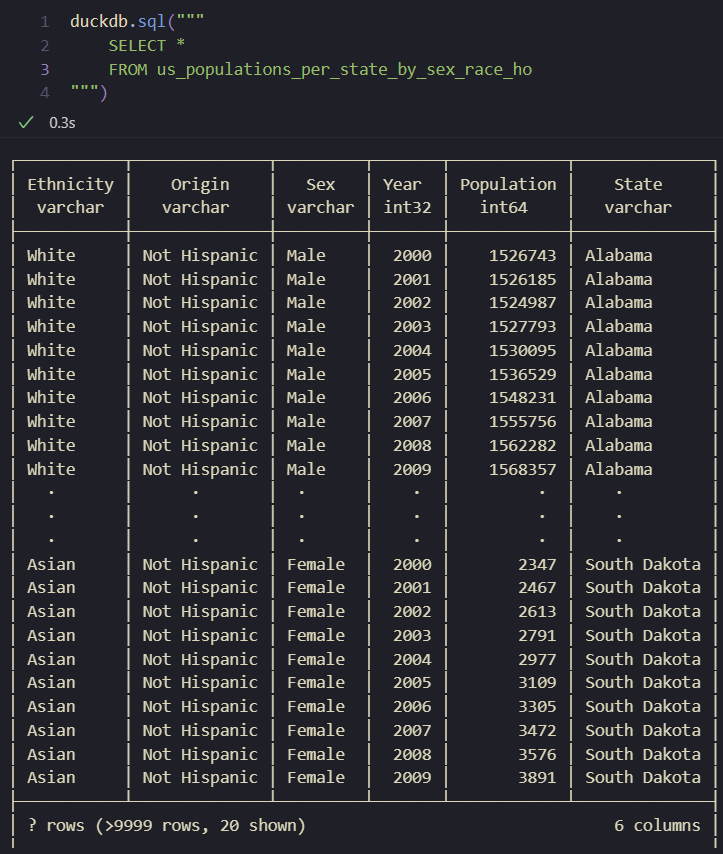
*CDI table loaded into DuckDB*



*CDI Table loaded into DuckDB 2nd half of columns*



*Population table by sex and age 2000 to 2023*



*Population table by sex, race, and Hispanic origin 2000 to 2023*

**5th Update:**

TLDR: finally managed to load the fact and dimension tables into an open source DWH using DuckDB, together with a schema detailed below of the relationship of the two chronic disease indicator (CDI) and population tables to their respective dimension tables

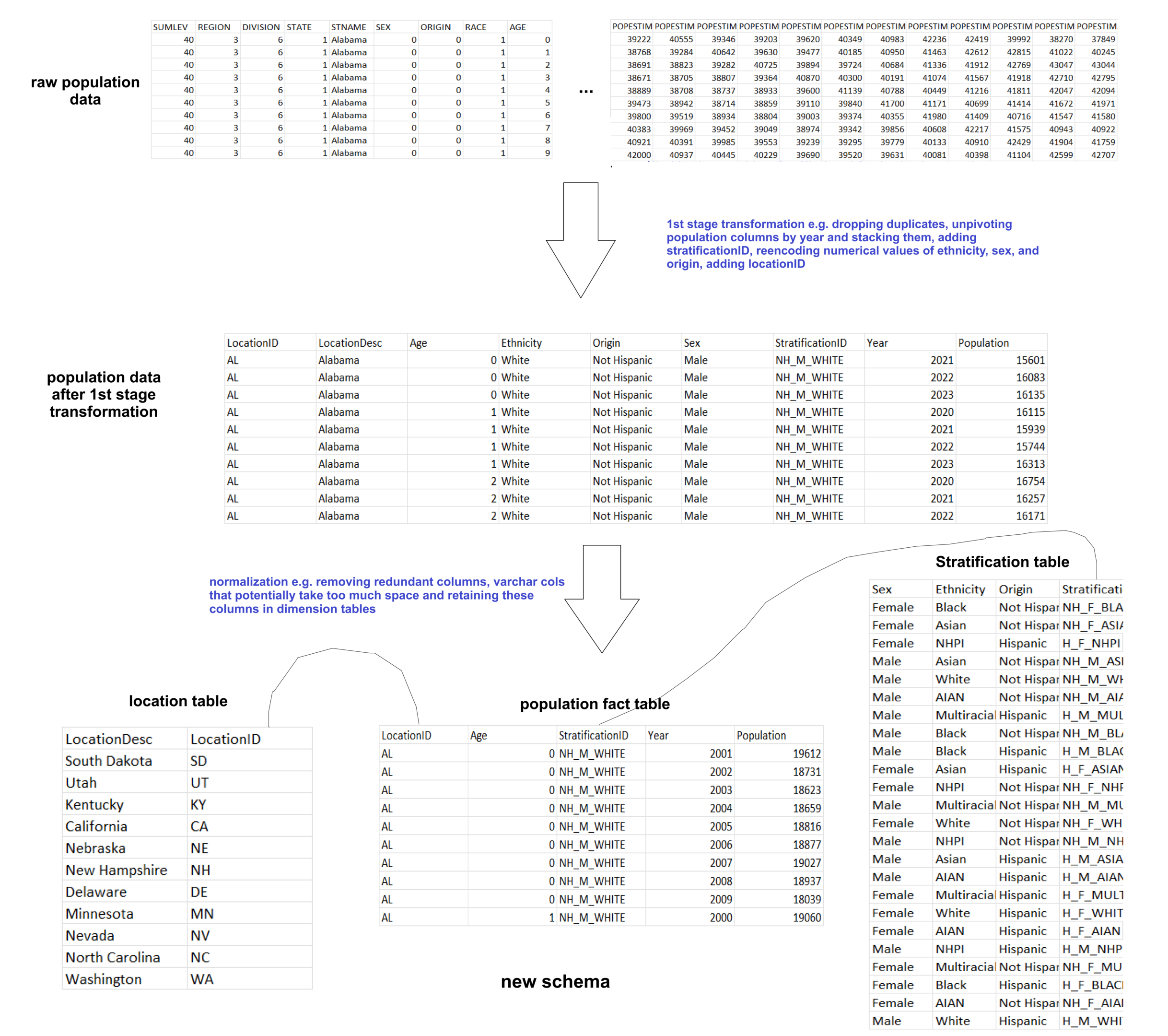
After running through error after error in loading the extracted raw data to an s3 bucket, reading this raw data from s3 (using Apache Spark), and finally loading the transformed and normalized tables to the s3 bucket, I finally was able to read the final tables (which I saved as parquet files) to an open source OLAP DB like DuckDB and finally be able to start some sort of data analysis.

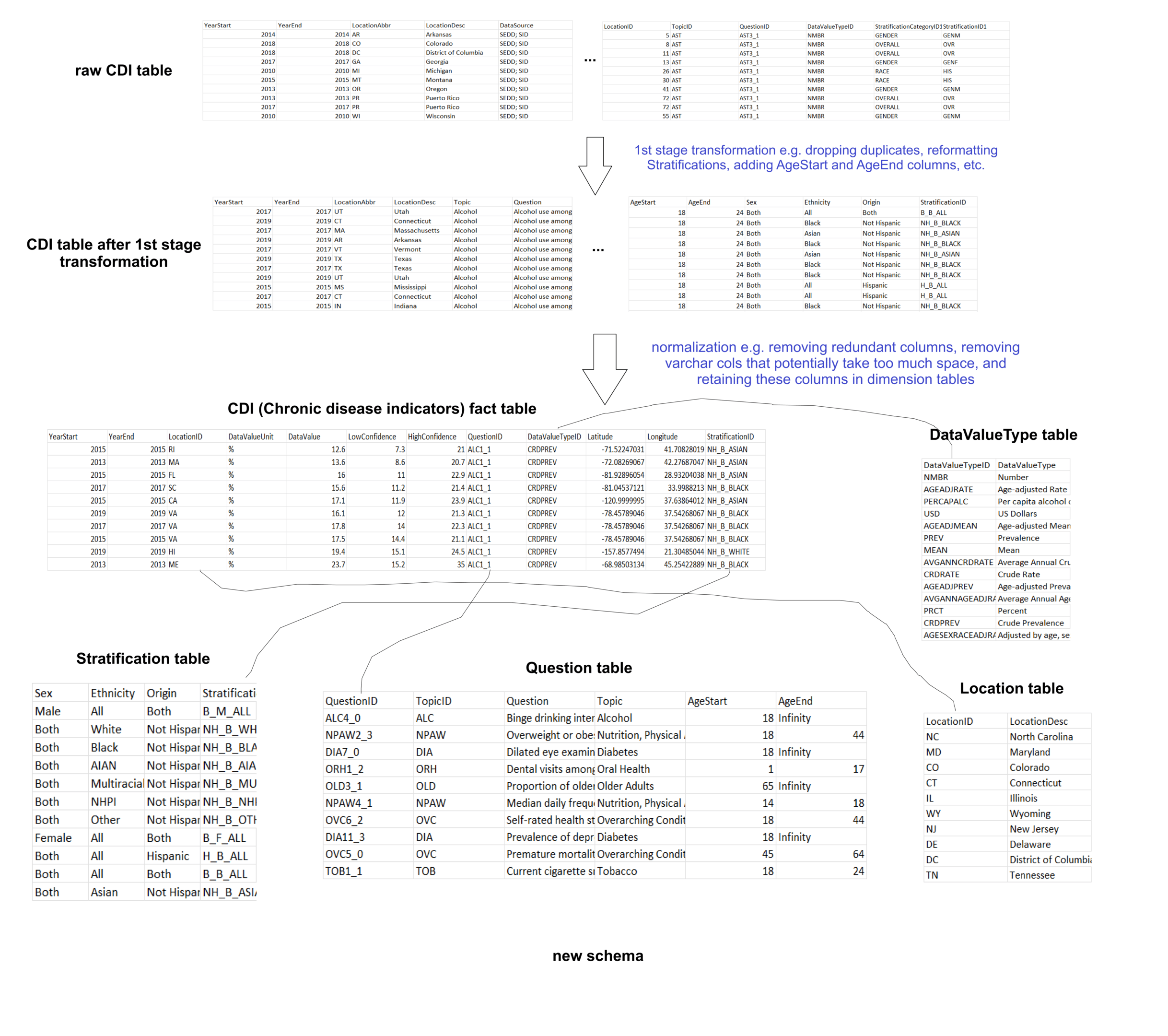
I think this was yet by far the part where I learned the most especially when it came to using s3 as a data lake where I could dump the raw data and also the transformed data, since I ran through countless errors involving IAM permissions and especially policy errors involving the s3 bucket instance itself, which I never knew was completely separate from each other (this <https://www.youtube.com/watch?v=gWAwqY76JQs> video in particular helped a lot).

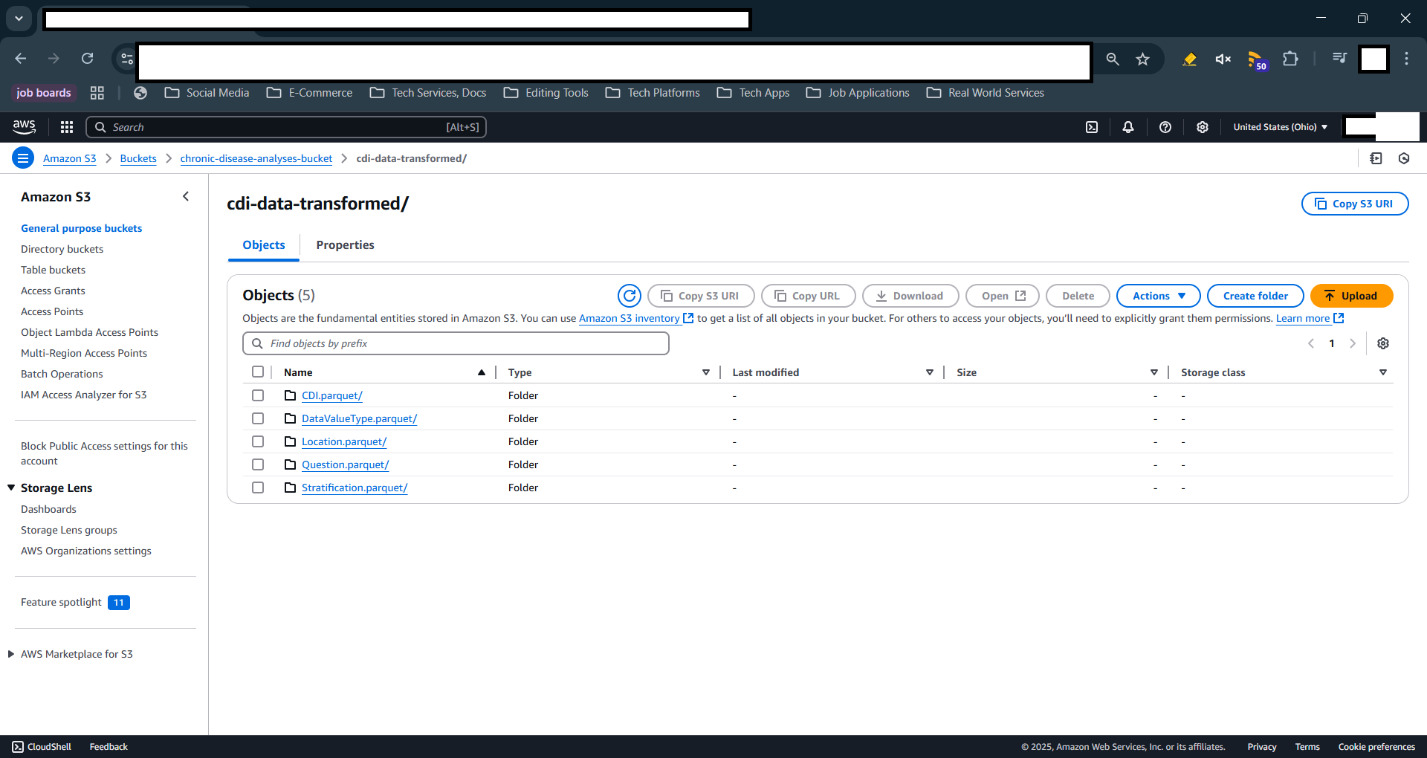
I also learned immensely from using external jar packages in Spark when submitting jobs, (talagang di basta basta ang mag-read and write from and to S3 with spark) as using the right version of the package had to be compatible with the other provided packages, and if not would result in an obscure error which I fortunately had managed to scour a solution all over the internet for in order to resolve. Case in point a `hadoop-aws:3.2.0` jar may not work with an `aws-sdk-sdk-bundle:1.12.367`, and a hadoop version you may currently have installed in your system may not work with `hadoop-aws:3.3.0`



*CDI and Population fact tables raw to 1st stage to normalized with schema*

*population data raw to 1st stage to normalized with relationships to dimension tables*

*CDI data raw to 1st stage to normalized with relationships to dimension tables*

**

*Result of a script that successfully uploads the transformed and normalized CDI and Population fact tables and their respective dimension tables to AWS S3*

**6th Update:**

I thought of using DAX in PowerBI initially for some joins with the CDI and Population fact tables again if you caught my previous posts basically the problem was rows of the CDI had difficult to interpret numbers which had rows with attribute/column values more or less like these  
```  
yearstart: 2012  
yearend: 2016  
locationdesc: connecticut  
question: cancer of the lung and bronchus mortality  
datavalueunit: cases per 100000  
datavaluetype: average annual crude rate  
datavalue: 9.6  
stratificationcategory: race/ethnicity  
stratification: asian or pacific islander  
```  
  
where this can be interpreted as the `average annual crude rate of cancer of the lung and bronchus mortality from 2012 to 2016 in the state of connecticut for an asian or pacific islander is 9.6 cases per 100000`. However again this isn't really useful as there isn't a tangible number we could touch on to differentiate the many datapoints of this CDI dataset. So I thought why not collate extra data in order to calculate a more tangible number, which in this specific row was more focused on the population demographic of Asian or pacific islanders, from the year 2012 to year 2016, in the state of Connecticut

This was the key to calculating the population with these specific demographics as calculating the total population will lead to later on being to calculate the total amount of cases/occurrence/prevalence of a chronic disease indicator. In Motherduck/DuckDB where the CDI and Population fact tables were housed having about 600k+ and 2 million+ rows respectively I wrote a query that would specifically calculate the total population given the specific demographics of each row in the CDI dataset

```

-- adds a new column to the cdi table to be populated

-- later on with values from update query

ALTER TABLE CDI

ADD COLUMN IF NOT EXISTS Population BIGINT

-- no condition needed to update rows as we are updating all

-- rows from blank to the population values

UPDATE CDI

SET Population = CalculatedPopulation.Population

FROM (

    -- Creates a CTE that will join the necessary

    -- values from the dimension tables to the fact

    -- table

    WITH MergedCDI AS (

        SELECT

            c.LogID,

            c.DataValueUnit,

            c.DataValue,

            c.YearStart,

            c.YearEnd,

            cl.LocationID,

            cl.LocationDesc,

            q.QuestionID,

            q.AgeStart,

            q.AgeEnd,

            dvt.DataValueTypeID,

            dvt.DataValueType,

            s.StratificationID,

            s.Sex,

            s.Ethnicity,

            s.Origin

        FROM CDI c

        LEFT JOIN CDILocation cl

        ON c.LocationID = cl.LocationID

        LEFT JOIN Question q

        ON c.QuestionID = q.QuestionID

        LEFT JOIN DataValueType dvt

        ON c.DataValueTypeID = dvt.DataValueTypeID

        LEFT JOIN Stratification s

        ON c.StratificationID = s.StratificationID

    ),

    -- joins necessary values to Population table

    -- via primary keys of its dimension tables

    MergedPopulation AS (

        SELECT

            ps.StateID,

            ps.State,

            p.Age,

            p.Year,

            s.Sex,

            s.Ethnicity,

            s.Origin,

            p.Population

        FROM Population p

        LEFT JOIN PopulationState ps

        ON p.StateID = ps.StateID

        LEFT JOIN Stratification s

        ON p.StratificationID = s.StratificationID

    ),

    -- performs an inner join on both CDI and Population

    -- tables based

    CDIWithPop AS (

        SELECT

            mcdi.LogID AS LogID,

            -- mcdi.DataValueUnit AS DataValueUnit,

            -- mcdi.DataValue AS DataValue,

            -- mcdi.YearStart AS YearStart,

            -- mcdi.YearEnd AS YearEnd,

            -- mcdi.LocationID AS LocationID,

            -- mcdi.LocationDesc AS LocationDesc,

            -- mcdi.QuestionID as QuestionID,

            -- mcdi.AgeStart AS AgeStart,

            -- mcdi.AgeEnd AS AgeEnd,

            -- mcdi.DataValueTypeID AS DataValueTypeID,

            -- mcdi.DataValueType AS DataValueType,

            -- mcdi.StratificationID AS StratificationID,

            -- mcdi.Sex AS Sex,

            -- mcdi.Ethnicity AS Ethnicity,

            -- mcdi.Origin AS Origin,

            SUM(mp.Population) AS Population

            -- mp.State PState,

            -- mp.Age AS PAge,

            -- mp.Year AS PYear,

            -- mp.Sex AS PSex,

            -- mp.Ethnicity AS PEthnicity,

            -- mp.Origin AS POrigin

        FROM MergedPopulation mp

        INNER JOIN MergedCDI mcdi

        ON (mp.Year BETWEEN mcdi.YearStart AND mcdi.YearEnd) AND

        (mp.StateID = mcdi.LocationID) AND

        ((mp.Age BETWEEN mcdi.AgeStart AND (CASE WHEN mcdi.AgeEnd = 'infinity' THEN 85 ELSE mcdi.AgeEnd END)) OR (mcdi.AgeStart IS NULL AND mcdi.AgeEnd IS NULL)) AND

        (mp.Sex = mcdi.Sex OR mcdi.Sex = 'Both') AND

        (mp.Ethnicity = mcdi.Ethnicity OR mcdi.Ethnicity = 'All') AND

        (mp.Origin = mcdi.Origin OR mcdi.Origin = 'Both')

        GROUP BY LogID

        ORDER BY LogID ASC

    )

    -- return all columns from resulting CTE in

    -- subquery

    SELECT \*

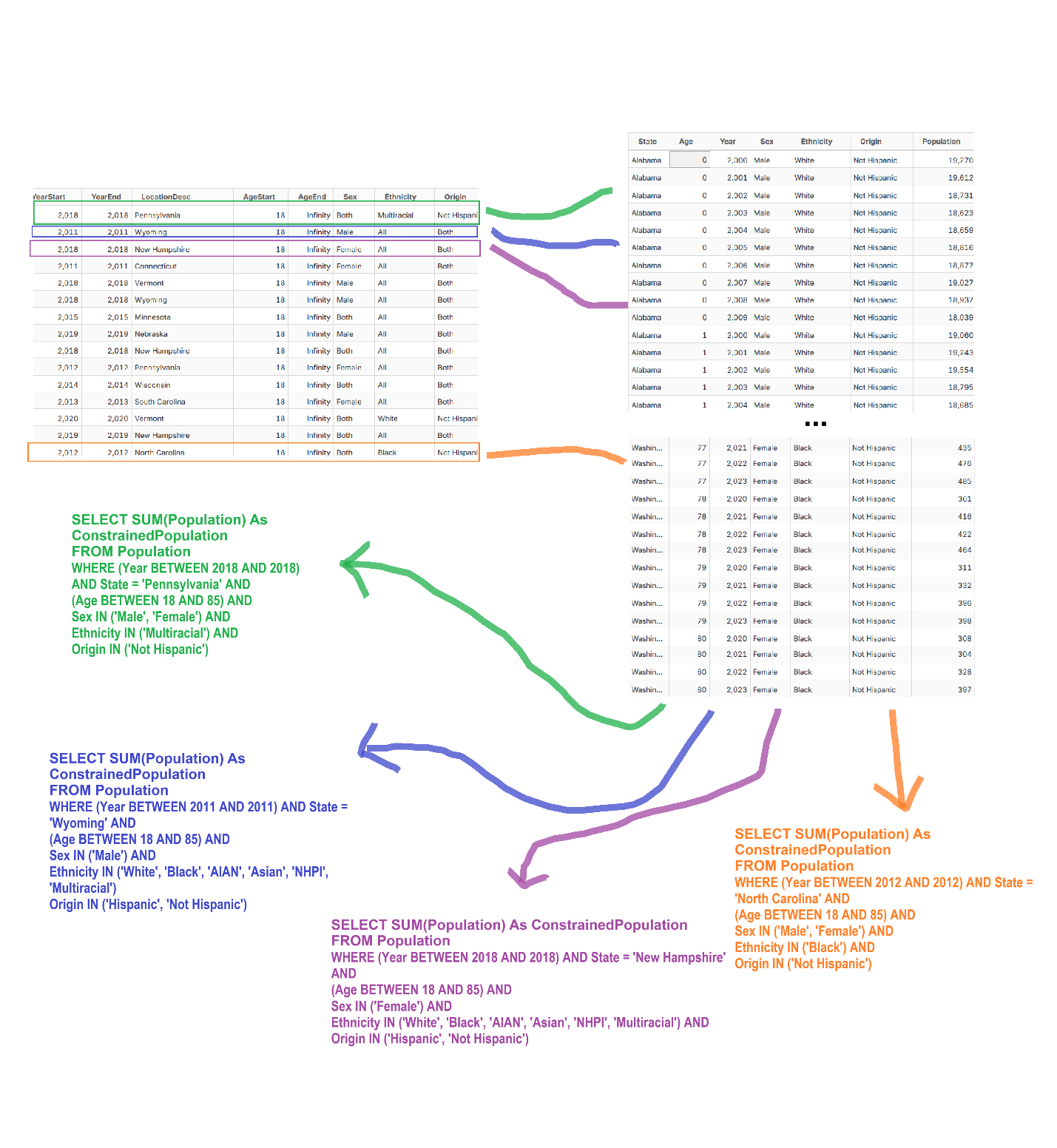
    FROM CDIWithPop

) AS CalculatedPopulation

WHERE CDI.LogID = CalculatedPopulation.LogID

```

The picture below visualizes how this query calculates the TotalPopulation column



*Visual representation of updating the CDI table to calculate the TotalPopulation column*

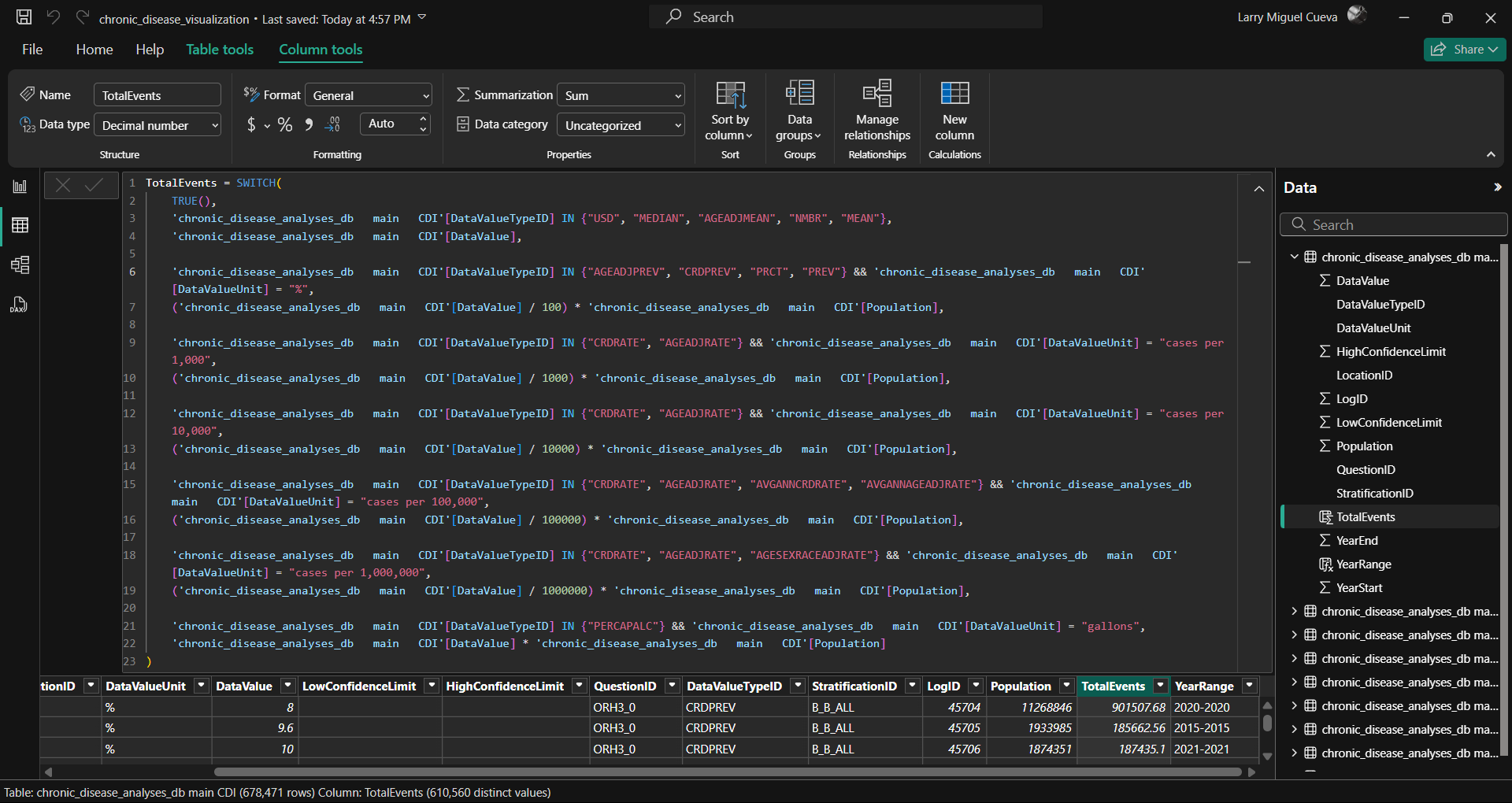
What this basically does is that per row more likely than not there will be a specific demographic say having a yearstart of 2012, a yearend of 2016, a location/state of Pennsylvania, an agestart of 18, an ageend of 85, a sex of male, an ethinicity of white, and an origin of Hispanic, these rows are joined to the necessary rows in the Population fact table. Imagine each row in the 600k+ CDI fact table and being joined to on average 50 rows in the Population but done 600k times. This would result in more or less 650,000,000 rows when these two fact tables are joined.

This is in part why I decided to use Motherduck/DuckDB as the main analytics database or data warehouse was so that I could do complex and computationally intensive calculations like these in a much faster time as opposed to using databases specifically tailored for transaction type queries. I figured that these large joins and large aggregations could be done by OLAP DBs like DuckDB/Motherduck.

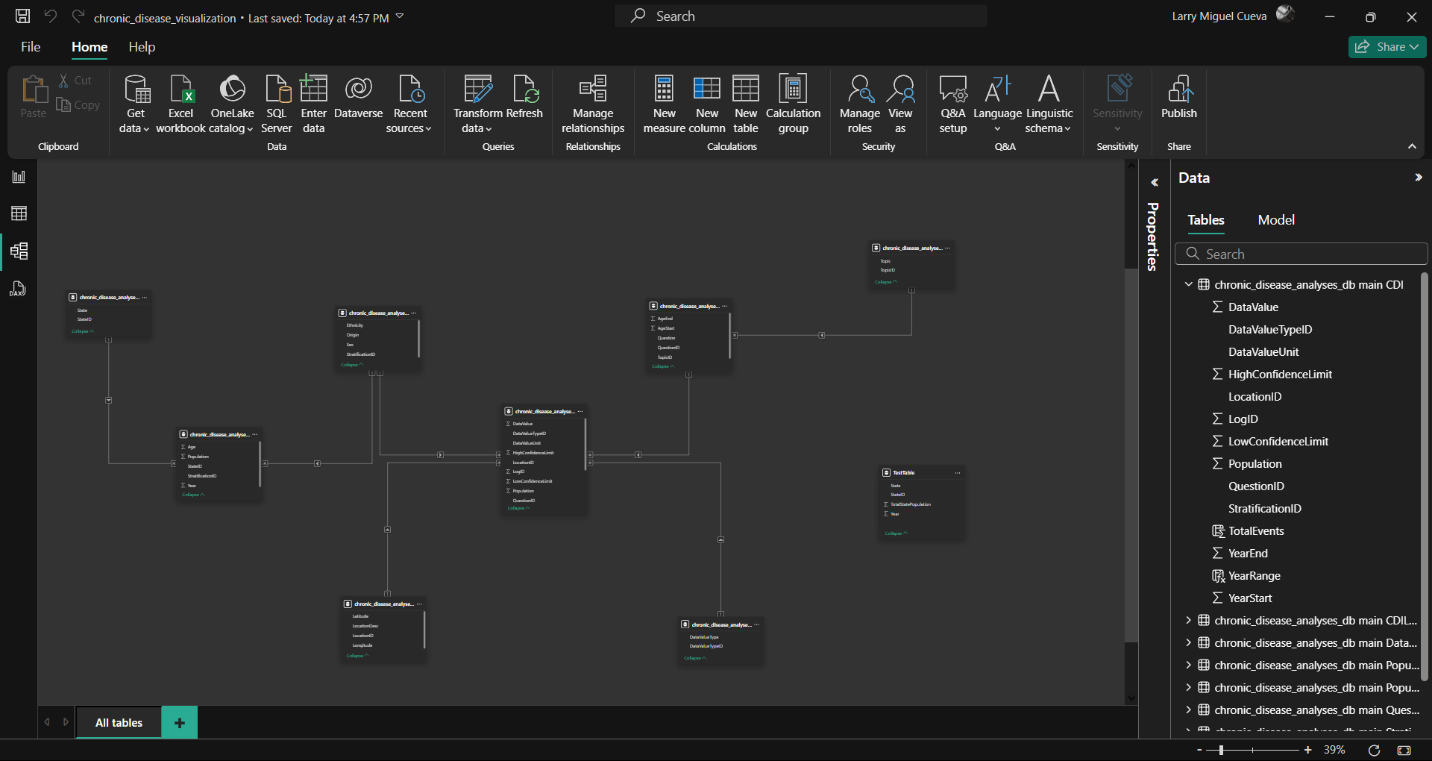
However, I see that even with SQL, processing 650,000,000 rows of data resulting from a join and then aggregating all these rows (even if inside already data warehouse) can be still quite slow 😂😮‍💨.

**7th update:**

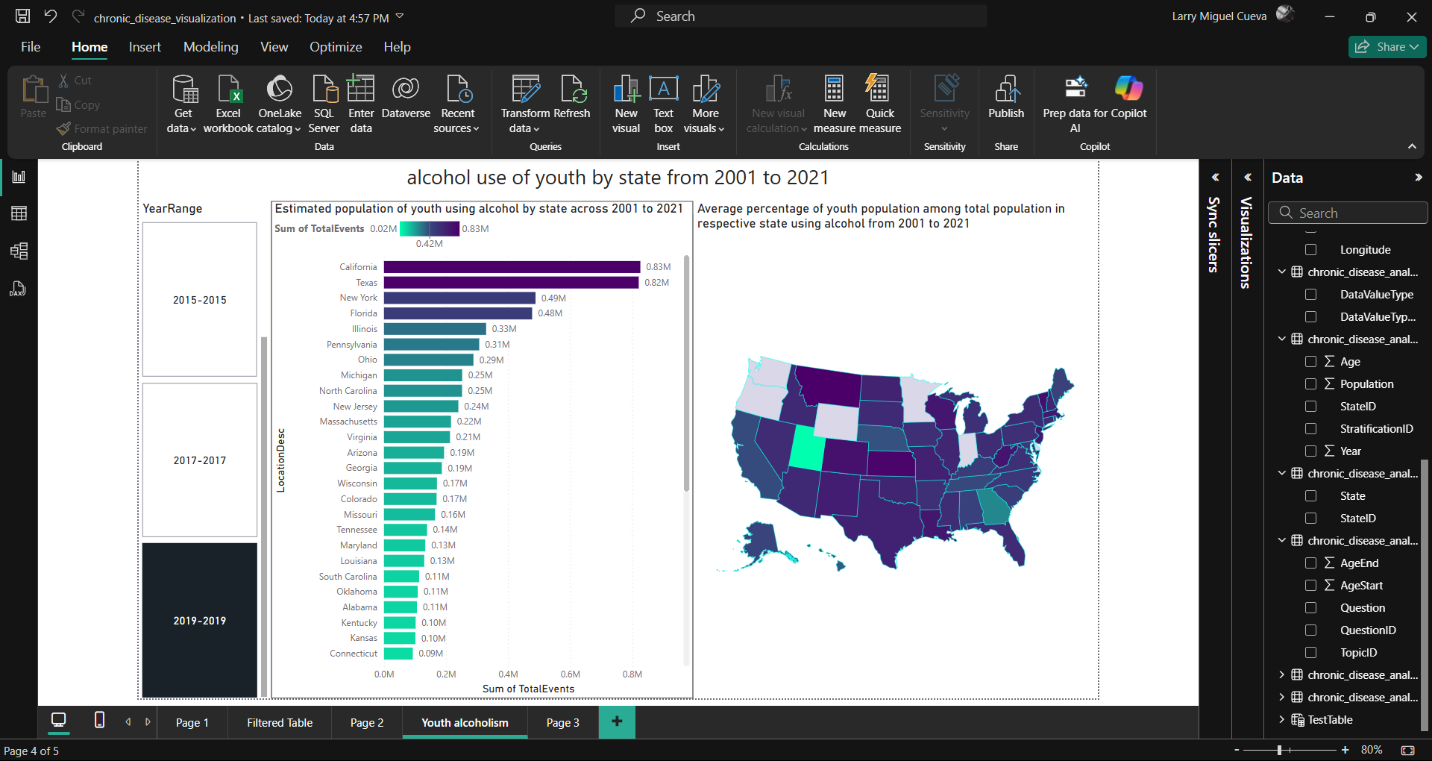
It's probably been a while since my last update but after countless modelling and re-modelling of the chronic disease indicators (CDI) and population fact table, I've finally managed to make the schema come into fruition through PowerBI's model view tab. Which allowed me to do finally some initial visualizations and ask some questions about the data.  
  
The visualization here used a few slicers, some simple bar charts, and a map of the states involved in the CDI dataset. It honestly probably looks like crap for now, but I'm happy that I was finally able to learn a lot from using PowerBI by using DAX queries to create new columns that could be used for these visualizations, by using slicers to filter out data like where clauses in SQL, and most importantly answer my first question related to this dataset: What is the prevalence of alcohol use among youth (male and female) in listed year ranges?  
  
interestingly there were multiple ways to answer the question:  
- the states with the most estimated youth populations recorded in 2013 using alcohol were in Texas, New York, Florida, Illinois, and Ohio  
- the states with the most estimated youth populations recorded in 2015 using alcohol were in California, Florida, New York, Illinois, and Pennsylvania  
- the states with the most estimated youth populations recorded in 2017 using alcohol were in California, Texas, New York, Florida, and Pennsylvania  
- the states with the most estimated youth populations recorded in 2019 using alcohol were in California, Texas, New York, Florida, and Illinois  
  
It's probably a basic question with a basic answer but I'm quite happy about the leaps of progress I made in learning to finally shift in the field of data analytics. Hopefully I can bring more in the next updates of this project.



*DAX code that calculates a calculated column TotalEvents representing the number of cases/prevalence/occurence of a chronic disease indicator*



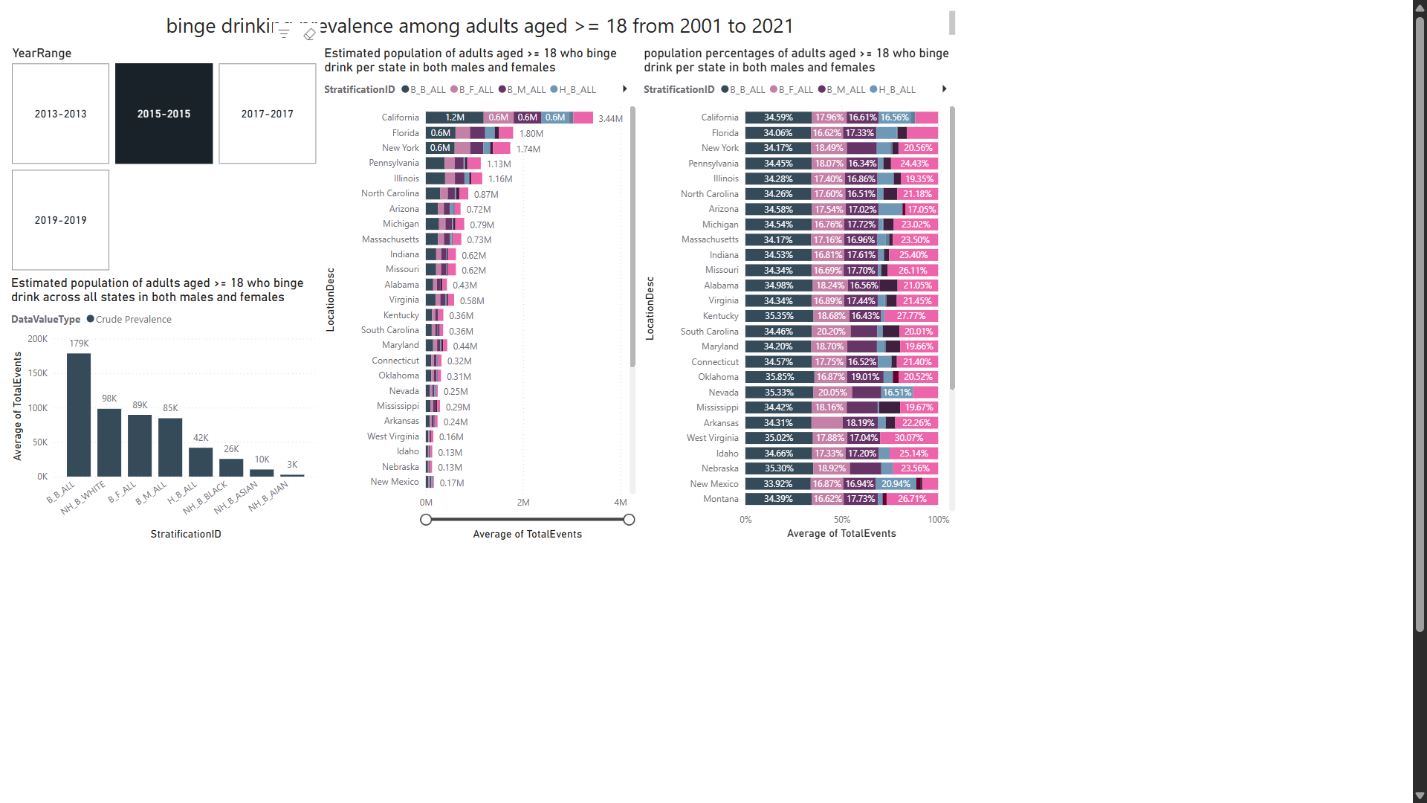
*PowerBI data model made from the CDI and Population fact tables and their respective dimension tables*



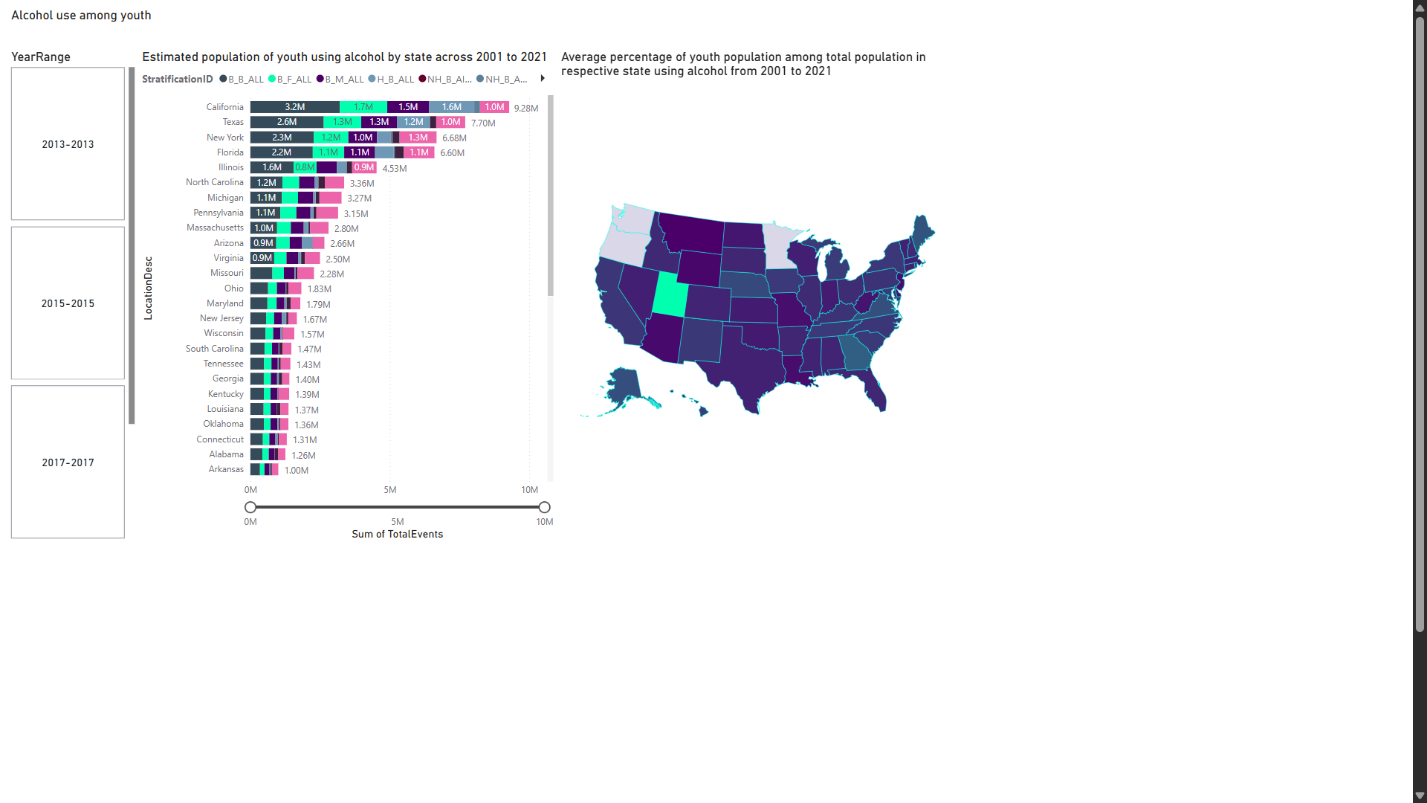
*Initial visualizations made to see the total events of chronic disease indicators across state and year*

**8th Update:**

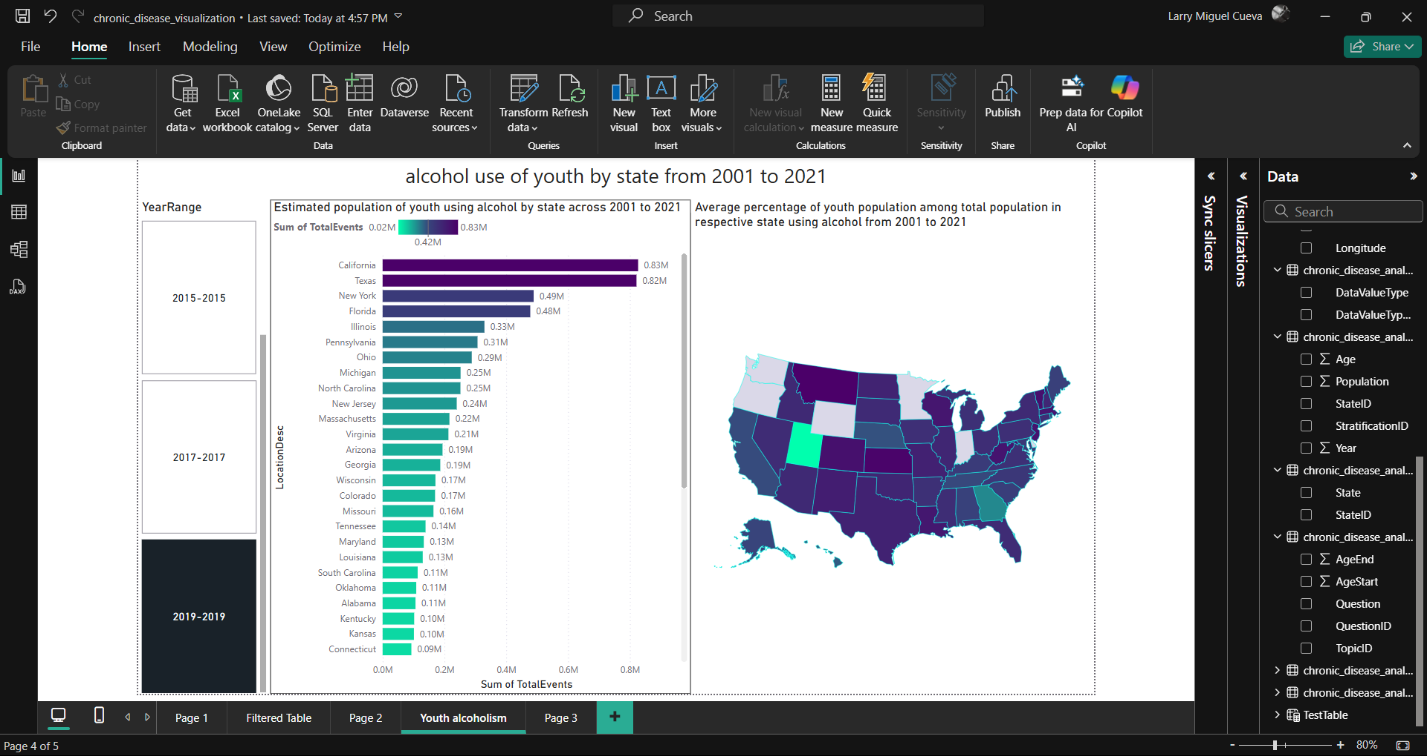
but I finally made an initial dashboard which probably looks bad now but still can improve in the future. I also deployed it online using vercel for public viewing.



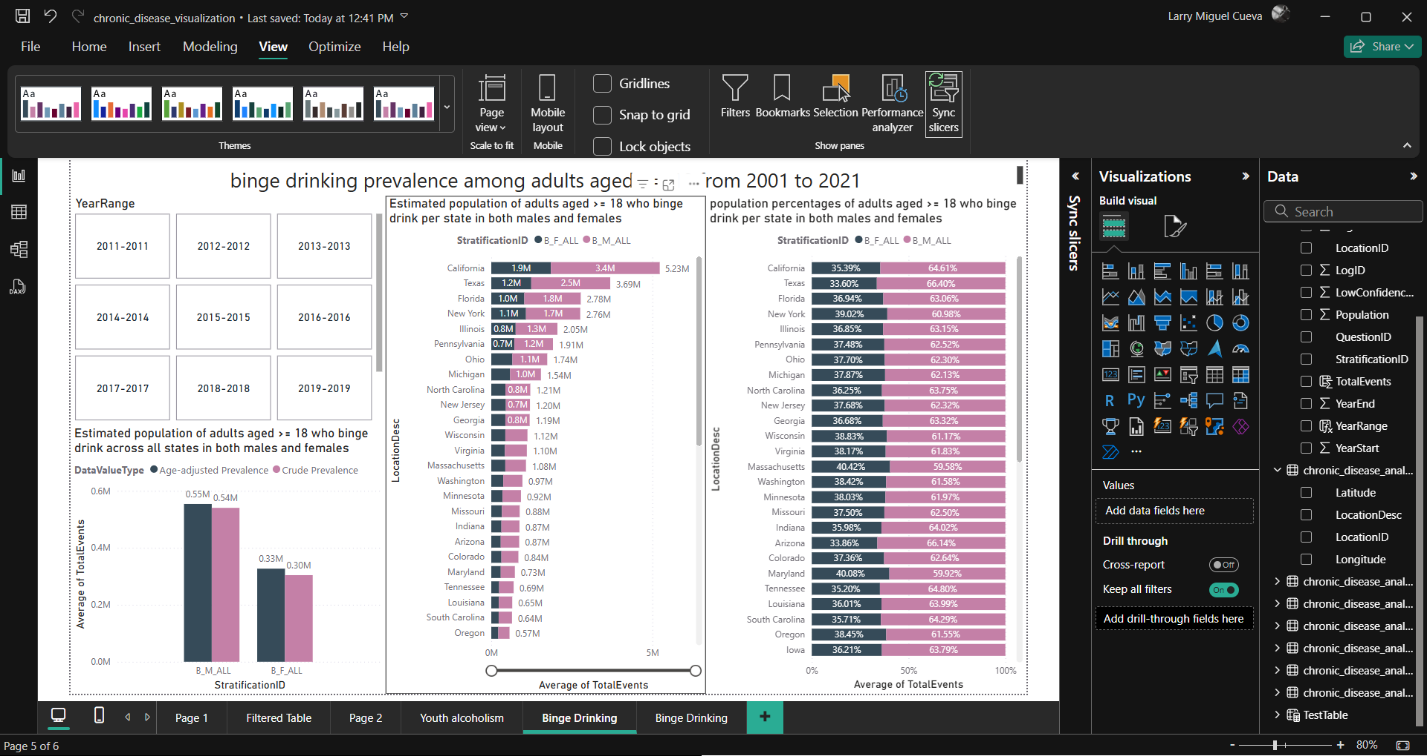
*Initial analyses of chronic disease indicators relating to alcohol*



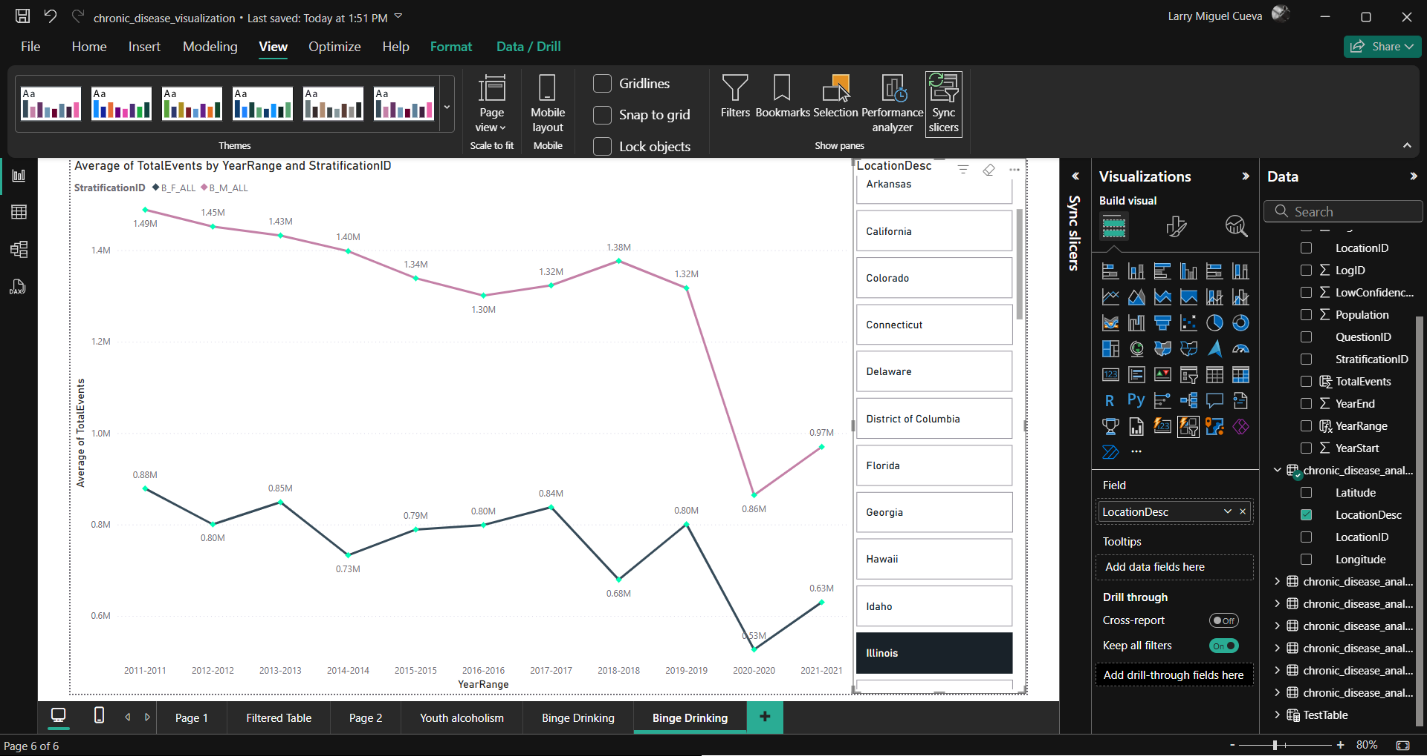
*Initial analyses of chronic disease indicators relating to alcohol*



*Initial analyses of chronic disease indicators relating to alcohol*



*Initial analyses of chronic disease indicators relating to alcohol*



*Initial analyses of chronic disease indicators relating to alcohol*

**9th update:**

Drawing this project to hopefully its main end I found in the process that there were just certain rows in the dataset that needed separate visualizations like those concerning amounts of .

it seems that `alcohol use among youth` question is something that can't make use of the `Population` table at first glance as what is measured here is alcohol use. But if we look closely the data value can't exactly say for sure whether alcohol amount is used e.g. 3.6% isn't exactly a measurement of alcohol amount but more likely a percentage of population namely the youth using alcohol

We also have other information like the datavaluetype used for the question. This is basically how PowerBI works when we put a slicer in our workspace we essentially get the unique values a unique value can take in, in this case a topic of `alcohol` has its unique questions like `alcohol use among youth` and then under it it has the unique `datavaluetype's` it uses like `crude prevalence`, and we know `crude prevalence` has exactly one corresponding `datavalueunit` which is the `%` symbol

because this question can make use of the `Population` table we can calculate the population of youth using alcohol.

it seems that `binge drinking frequency among adults aged >= 18 years who binge drink` question is something that can't make use of the `Population` table as is only specific to the population of those 18+ **\*\*who binge drink\*\***. To get a tangible number namely the drinking frequency of these people aged 18+ who binge drink, we need to get the prevalence of these people aged 18+ who indeed binge drink, which we know can be derived from other questions namely `binge drinking prevalence among adults aged >= 18 years` and then from there multiply it by the binge drinking frequency, since it is assumed that each person that binge drinks has this frequency.

calculation would be $prevalence\\_of\\_binge\\_drinkers\\_aged\\_18+ \cdot datavalue$ but since we don't have prevalence of binge drinkers aged 18+ until we further calculate it the alternative calculation would be $mean\\_binge\\_drinking\\_frequency\\_among\\_adults\\_aged\\_18+\\_who\\_binge\\_drink$, since its datavalueunit is just Number meaning this number as it is will be the representation of this indicator

Filtering DataValueTypeIDs with MEAN, AGEADJMEAN, PERCAPALC, MEDIAN, USD as these indicate Non total events related