Member Roster Aggregation and Standardization

Albert Yuan

Results: Summary Statistics

How many distinct members are eligible in April 2022?
86418/109213 members

How many members were included more than once?
22795/109213 members

What is the breakdown of members by payer?
32347 members under Madv; 54071 members under Mdcd

How many members live in a zip code with a food_access_score lower than 2?
6676 members

What is the average social isolation score for the members?
Average score: 3.068

Which members live in the zip code with the highest algorex_sdoh_composite_score?
38 members live in zip code 95950 where the score is 8.77

Approach: Table Creation

1. Create std_member_info
2. Populate std_member_info
   a. Filter distinct members
   b. Filter eligibility window
   c. Date standardization
   d. State standardization

```
CREATE TABLE std_member_info (cols);

INSERT INTO std_member_info select
…;
DISTINCT Person_Id

WHERE eligibility_start_date < DATE('2022-04-01') AND
eligibility_end_date >= DATE('2022-05-01')

DATE(SUBSTR(Dob, 7,4 ) || '-' || SUBSTR(Dob,1,2) || '-' ||
SUBSTR(Dob, 4,2))

CASE State
      WHEN 'AL' THEN 'Alabama'
      WHEN 'AK' THEN 'Alaska'...
```

Approach: Summary Statistic Queries

```
--DISTINCT MEMBERS
SELECT COUNT(DISTINCT member_id)

--DUPLICATE MEMBERS
SELECT COUNT()
FROM(SELECT COUNT(member id) as cnt
     FROM std member info
     GROUP BY member_id)
WHERE cnt > 1

--MEMBERS BY PAYER
SELECT payer, COUNT(DISTINCT member_id)
...GROUP BY payer

--AVG social_isolation_score
SELECT avg(social_isolation_score)
...LEFT JOIN model_scores_by_zip...
```

```
--FOOD_ACCESS_SCORE < 2
SELECT COUNT(DISTINCT member_id)
...LEFT JOIN model_scores_by_zip...
WHERE m.food_access_score < 2

--HIGHEST algorex_sdoh_composite_score
WITH highest_score as (
      SELECT algorex_sdoh_composite_score, zcta
      FROM model_scores_by_zip
      ORDER BY algorex_sdoh_composite_score DESC
      LIMIT 1)
SELECT count(*), zip_code, algorex_sdoh_composite_score
...RIGHT JOIN highest_score…
GROUP BY zip_code, algorex_sdoh_composite_score
```

Handling Future Rosters

1. Identify any columns that need to be standardized
2. Reuse/write new SELECT queries
3. Insert new roster
4. Run summary statistics queries

```
def insert_std_member_info_no_dupe_memberid (query, con):
    insert_std_member_info_query = f"""
    INSERT INTO std_member_info (member_id,
member_first_name, member_last_name, date_of_birth,
main_address, city, state, zip_code, payer)
    {query}
    AND Person_Id NOT IN (
        SELECT member_id FROM std_member_info)
    """

    con.cursor().execute(insert_std_member_info_query)
    con.commit()
```