GENAI Case Study Prompts - Lexi René Glatz

1. Typically I want to make code efficient, so using chatgpt I am interested in the best approach. In the drug section of my Quarto script I initially coded most of the data extraction section similar to that of the comorbidity section however I encountered a point in which I could not pipe in what I wanted, so I had to hard code a section which was not desirable. To make the section more efficient I used the following prompt:

   Convert this R code to SQLite:

   ```
   candy_exp_baseline <- run_sql_query("select distinct
   c.subject_id,
   c.ctstart_date,
   de.*,
   con.concept_name as candy_name
   from
   cohort as c
   left join (
   select
   person_id,
   candy_concept_id,
   candy_exp_start_date as candy_start_date,
   candy_exp_end_date as candy_end_date
   from
   candy_exp
   where
   candy_concept_id != 0
   ) as de on c.subject_id = de.person_id
   AND de.candy_start_date <= c.ctstart_date
   AND (de.candy_end_date >= c.ctstart_date OR de.candy_end_date is null)
   left join
   concept as con on de.candy_concept_id = con.concept_id
   where
   c.ctdefinition_id = 4
   ")


   candy_era_baseline <- run_sql_query("select distinct
   c.subject_id,
   c.ctstart_date,
   de.*,
   con.concept_name as candy_name
   ```

```
from
cohort as c
left join (
select
person_id,
candy_concept_id,
candy_era_start_date as candy_start_date,
candy_era_end_date as candy_end_date
from
candy_era
where
candy_concept_id != 0
) as de on c.subject_id = de.person_id
AND de.candy_start_date <= c.ctstart_date
AND (de.candy_end_date >= c.ctstart_date OR de.candy_end_date is null)
left join
concept as con on de.candy_concept_id = con.concept_id
where
c.ctdefinition_id = 4
")


lower_dn <- bind_rows(candy_exp_baseline |> distinct(candy_name) ,
candy_era_baseline |> distinct(candy_name)) |>
  count(candy_name) |>
  filter(!grepl(' ', candy_name)) |>
  pull(candy_name) |>
  tolower()

candys_at_baseline <- bind_rows(candy_exp_baseline |> mutate(db_exp = T) ,
candy_era_baseline |> mutate(db_era = T)) |>
  group_by(subject_id, candy_concept_id) |>
  mutate(
    data_source = case_when(
      any(db_exp == T) & any(db_era == T) ~ 'both',
      any(db_exp == T) & any(is.na(db_era)) ~ 'exp',
      any(is.na(db_exp)) & any(db_era == T) ~ 'era',
      T ~ 'check'
    )
  ) |>
  ungroup() |>
  mutate(
    candy_name_L = tolower(candy_name),
    overall_candy_name = case_when(
```

```
    grepl(lower_dn[1], candy_name_L) ~ lower_dn[1],
    grepl(lower_dn[2], candy_name_L) ~ lower_dn[2],
    grepl(lower_dn[3], candy_name_L) ~ lower_dn[3],
    grepl(lower_dn[4], candy_name_L) ~ lower_dn[4],
    grepl(lower_dn[5], candy_name_L) ~ lower_dn[5],
    grepl(lower_dn[6], candy_name_L) ~ lower_dn[6],
    grepl(lower_dn[7], candy_name_L) ~ lower_dn[7],
    grepl(lower_dn[8], candy_name_L) ~ lower_dn[8],
    grepl(lower_dn[9], candy_name_L) ~ lower_dn[9],
    grepl(lower_dn[10], candy_name_L) ~ lower_dn[10],
    grepl(lower_dn[11], candy_name_L) ~ lower_dn[11],
    grepl(lower_dn[12], candy_name_L) ~ lower_dn[12],
    grepl(lower_dn[13], candy_name_L) ~ lower_dn[13],
    T ~ candy_name_L
  )
) |> select(-candy_name_L) |>
distinct(subject_id, candy_concept_id,
         candy_start_date, .keep_all = T) |>
arrange(subject_id, candy_start_date)
```

2. As I have not yet had a complete project using the OMOP CDM framework, many of my prompts were to learn more about it in an adequate amount of time. Prompt that questioned the difference between the era and condition tables. Prompts that questioned if there were mappings outside of the concept_id's that I needed to be aware of. What it meant if there was data about a condition in one table that was not available in the other, etc.