Denormalization of a database can improve the speed of queries, though not noticeably to the human eye. It does this by reducing the number of joins required to retrieve the requested data of a query. To get the full location of someone with our given, normalized database containing the separate tables for city, county, and state, the query would have to first join these tables, thus requiring additional CPU processes. By denormalizing the tables into one location table you speed up the query, but you introduce data redundancy.

To get the location of someone with the normalized tables, you have the following query.

SELECT State.state\_ID, State.state\_name, county.county\_id, county.county\_name, city.city\_id, city.city\_name

FROM (State LEFT JOIN county ON State.state\_ID = county.state\_id\_) LEFT JOIN city ON county.county\_id = city.county\_id

To denormalize tbl\_city, tbl\_county, and tbl\_state into tbl\_location, the following actions are performed. I created a new table, tbl\_location, with seven fields. The first field is loc\_id and it is the primary key for the table. I imported the fields from the state, county, and city tables, with each of those id fields being foreign keys.

To get the location of someone with the denormalized table, you have the following query.

SELECT tbl\_location.state\_ID, tbl\_location.state\_name, tbl\_location.county\_id, tbl\_location.county\_name, tbl\_location.city\_id, tbl\_location.city\_name

FROM tbl\_location

