# Census Database

Adam Stammer, Isaac Plevak

adam.stammer@go.winona.edu, isaac.plevak@go.winona.edu

April 29th, 2019

**Roles:**

We broke the project into two main categories: Data Acquisition and Database Design. Both of us worked extensively on both categories but Adam primarily worked on Data Acquisition while Isaac worked on Database Design. This decision was made not only to organize the work and make it easier to prioritize and plan, but also because Adam has more experience with python scripting and csv formatting.

**Summary:**

For this project we wanted real life data not only for the practicality of such a decision but so that our own intuition could help us in evaluating the final result. We knew this would involve extra work in formatting the data for database use so we wanted something we wouldn't have to format manually or from scratch. US Census Bureau data seemed a good option. It provides a large quantity of relevant data in XLS format. It's also fit our criteria of intuition assistance. The goal was to format the data to make patterns easy to identify and make answering practical questions easier. The data remains just as easily readable as its original format but also provides the flexibility to ask more complicated questions. We accomplished that very thing as our use cases successfully demonstrate. Use case #3 in particular represents an application of patterns, complex questions, and intuitive verification.

Throughout this project we learned a lot and as such there are things we would do differently if it was done again. Along with the various changes we made as the project was completed the main change we would also make is the implementation of regions. Our original design was based on separate region data being available since it was a derivative of the other data it seemed redundant, thus we chose views as a more practical implementation. While this does work, adding a region attribute to the state table would be just as effective and potentially easier to work with. The addition of a region table to store the region names would further reduce redundancy and align with our existing design. Thus our final project is not ideal nor perfect, but we are pleased with the results nonetheless.

**Data:**

The US Census Bureau releases various census measurements after a given period of time. The data is released in PDF and XLS formats for use by the general public. This meant we would have to extract the data and reformat it in a way easily imported into our database. Reformatting the data required extensive knowledge of the database design and the database design relied on the formatting abilities we had, so the two categories mentioned earlier strongly relied on each other. This further emphasized the teamwork and collaboration required to complete this project. A python script, written by us, was used to extract the relevant data from the available excel documents, reformatted with proper headers and columns, and exported into csv files. A batch scripts, also written by us, were then used to run the aforementioned python script on all of the relevant excel documents and concatenate the resulting csv files into one final csv file that contained all of the state measurement data in an easily imported format. Along with some manually formatted county data and views based on geographical regions, this creates a database that makes analysis of census data much easier as the use cases demonstrate.

**Design:**

Our design for the Census database was focused on reducing redundancy as much as we possibly could, we were using a relational database afterall. To accomplish this we had year as its own table so that any time we needed the year 1970 for instance we wouldn't need to store the value 1970 for every time that we needed that date. We used a foreign key to reference 1970 from the year table. We did the same thing for state name and county name. The name for the state was stored in seperate table from the population measurement for that state. This way the name, just like the year, isn't stored more than once. To get the population and the number of representatives a state has you need to give the name and the year that you want the data from. These two make the primary key to the state_measure table. The county_measure table has the same design philosophy using county name and the year. The county_measure table does need the name of the state that the county is in as part of the primary key however, because some counties have the same name throughout the country. Originally we had region data planned as a separate table set up very similar to the state and county tables, however we found that region data was harder to come by. Thus we decided to get rid of the actual tables for region information. To get region information simply add the population from the states of the region desired. We utilized views to manage this data.

Relation Model:

county(<u>county_name: String(40)</u>)

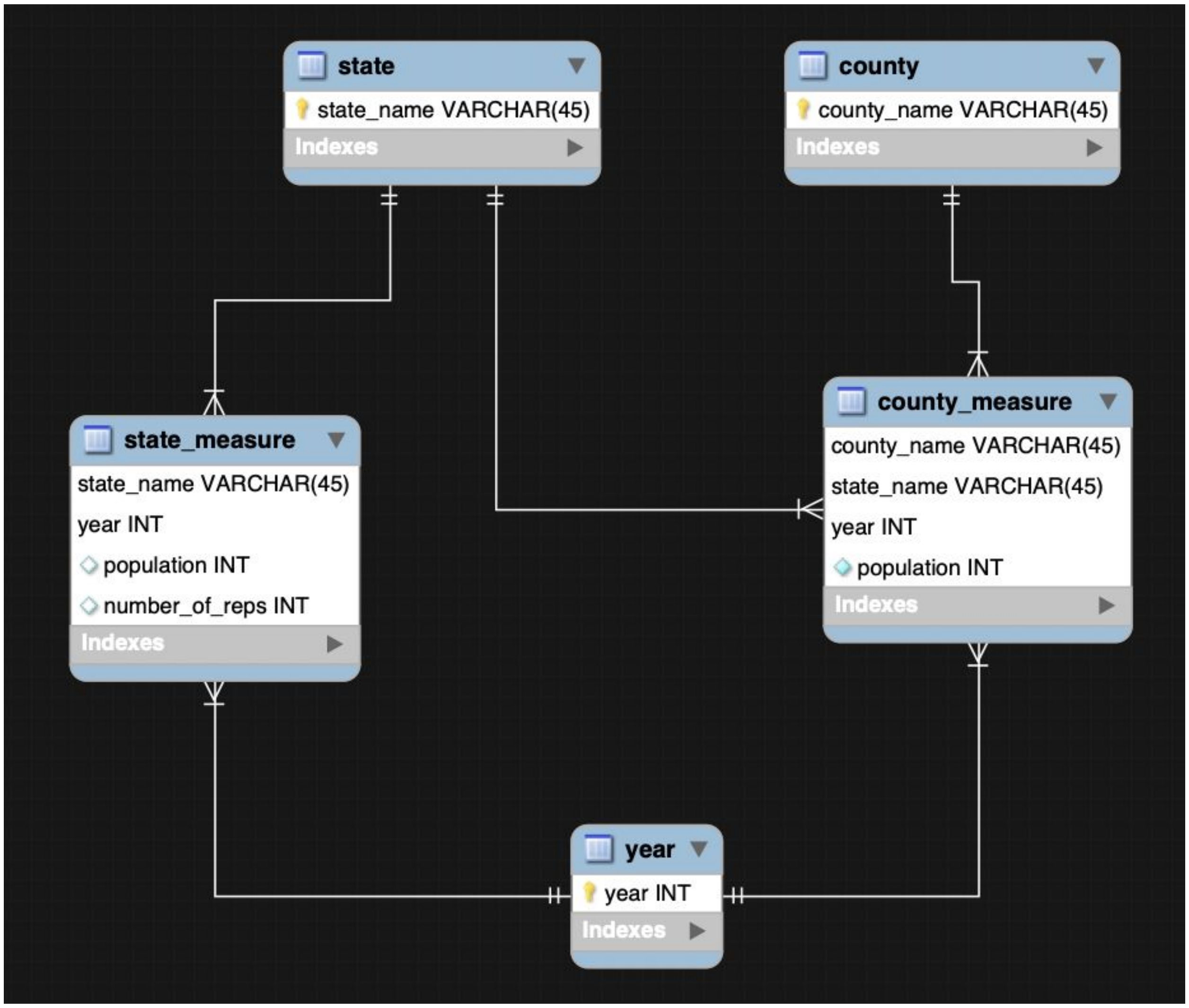
county_measure(*county_name*: String(40), *state_name*: String(40), *year*: Int(11), population: Int(11))

state(state_name: String(40))

state_measure(*state_name*: String(40), *year*: Int(11), population: Int(11), number_of_reps: Int(11))

year(year: Int(11))

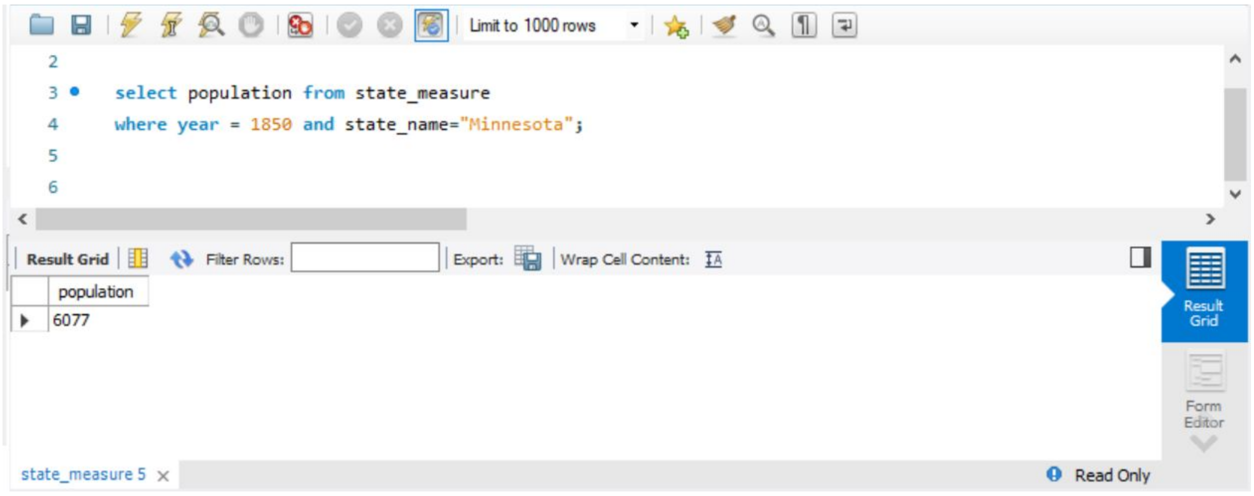
Entity Relationship Diagram:



Original designs included state and city data due to the data we had already found, but we changed the city based design to that of counties when further research revealed that county data was much more common. We also added year as a foreign key when we realized the memory cost of our original design.

**Use Cases:**

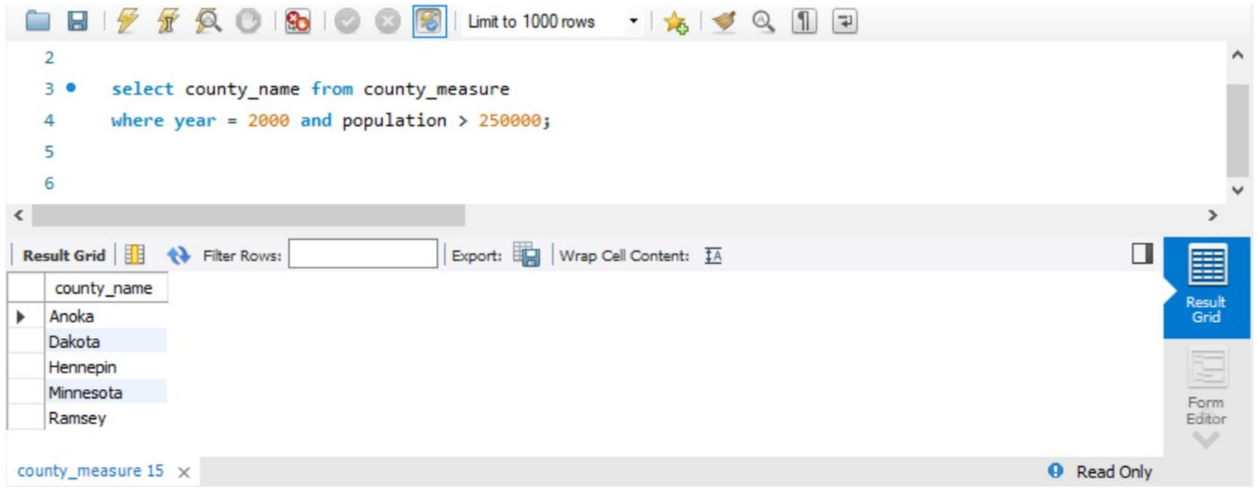
1. What was the Minnesota state population in 1850?



```sql
select population from state_measure
where year = 1850 and state_name="Minnesota";
```

```
population
6077
```

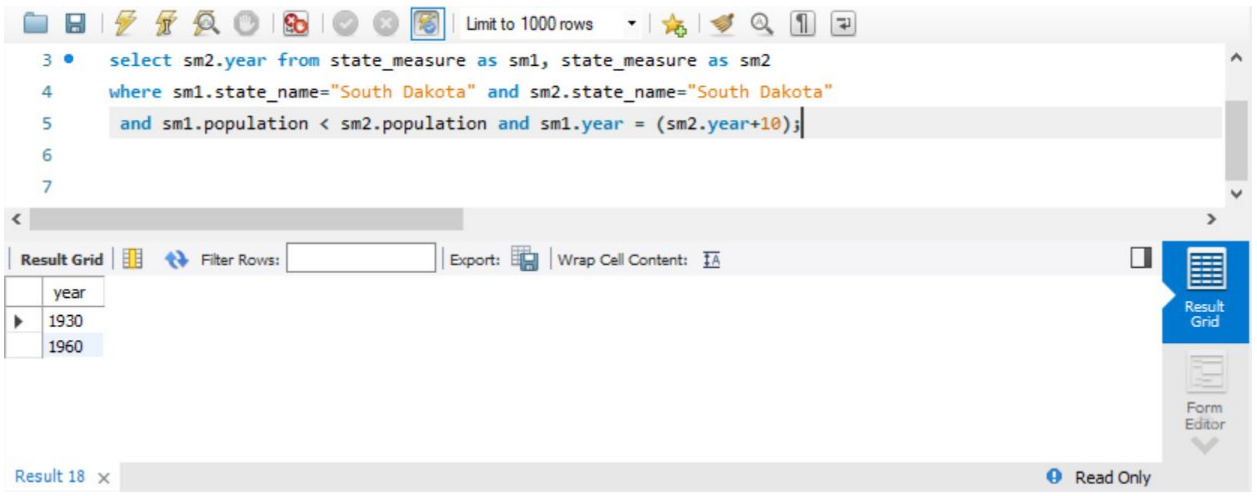
2. Name all counties in Minnesota with a population greater than 250,000 in the year 2000.



```sql
select county_name from county_measure
where year = 2000 and population > 250000;
```

```
county_name
Anoka
Dakota
Hennepin
Minnesota
Ramsey
```

3. During which decades did the population of South Dakota decline? In this example we see that SD population declined in the 30s and 60s, likely attributed to The Great Depression & WWII, and the growth of the Sunbelt & The Vietnam War respectively.



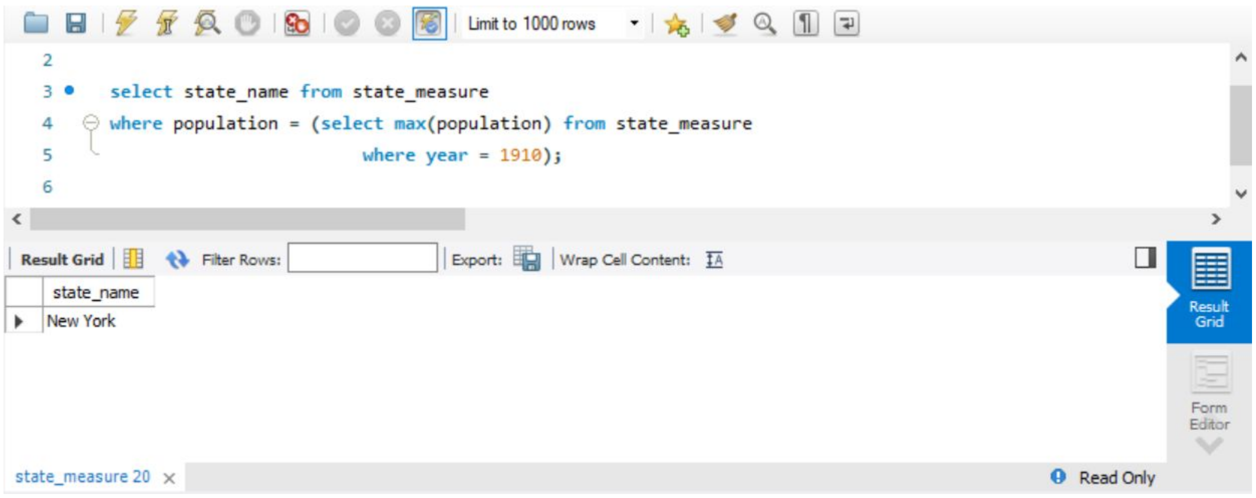
```
select sm2.year from state_measure as sm1, state_measure as sm2
where sm1.state_name="South Dakota" and sm2.state_name="South Dakota"
 and sm1.population < sm2.population and sm1.year = (sm2.year+10);
```

| year |
|---|
| 1930 |
| 1960 |

Result 18

Read Only

4. In 1910, which state had the greatest population?



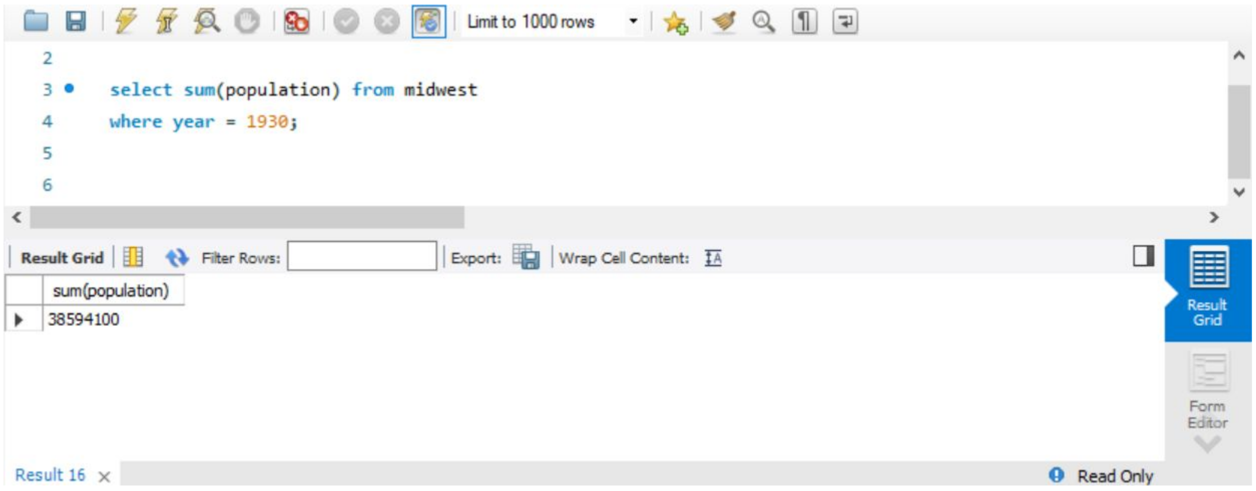
```
select state_name from state_measure
where population = (select max(population) from state_measure
                                  where year = 1910);
```

| state_name |
|---|
| New York |

state_measure 20

Read Only

5. Total population of the midwest region in 1930



```sql
select sum(population) from midwest
where year = 1930;
```

```
sum(population)
38594100
```