


# CS50's Introduction to Databases with SQL

OpenCourseWare

Donate  (<https://cs50.harvard.edu/donate>)

Carter Zenke (<https://carterzenke.me>)



[carter@cs50.harvard.edu](mailto:carter@cs50.harvard.edu)



 (<https://github.com/carterzenke>)  (<https://www.linkedin.com/in/carterzenke/>)

David J. Malan (<https://cs.harvard.edu/malan/>)

[malan@harvard.edu](mailto:malan@harvard.edu)

 (<https://www.facebook.com/dmalan>)  (<https://github.com/dmalan>)  (<https://www.instagram.com/davidjmalan/>)

 (<https://www.linkedin.com/in/malan/>)  (<https://www.reddit.com/user/davidjmalan/>)

 (<https://www.threads.net/@davidjmalan>)  (<https://twitter.com/davidjmalan>)

## Census Taker



*"The Himalayan mountain ranges in the style of a Nepali painting", generated by [DALL·E 2](https://openai.com/dall-e-2) (<https://openai.com/dall-e-2>)*

## Problem to Solve

You are a census taker working for the Nepali government. As you crest one final hill, your breath catches at the sight of a Himalayan sunrise, casting a glow on the village you've journeyed so far to reach. Your guide, a local, halts abruptly. Underneath the steady rustle of your census papers, you feel an itch of curiosity. After all, it's not every day your job takes you to a village like this one.

In `census.db`, process your data into views the Nepali government can use for record-keeping.

## Demo

```
$ sqlite3 census.db
sqlite> SELECT "district", "locality"
```

Recorded with [asciinema](#)

## Distribution Code

For this problem, you'll need to download `census.db`, along with a few `.sql` files in which you'll write your queries.

### ▼ Download the distribution code

Log into [cs50.dev](https://cs50.dev) (<https://cs50.dev/>), click on your terminal window, and execute `cd` by itself. You should find that your terminal window's prompt resembles the below:

```
$
```

Next execute

```
wget https://cdn.cs50.net/sql/2023/x/psets/4/census.zip
```

in order to download a ZIP called `census.zip` into your codespace.

Then execute

```
unzip census.zip
```

to create a folder called `census`. You no longer need the ZIP file, so you can execute

```
rm census.zip
```

and respond with “y” followed by Enter at the prompt to remove the ZIP file you downloaded.

Now type

```
cd census
```

followed by Enter to move yourself into (i.e., open) that directory. Your prompt should now resemble the below.

```
census/ $
```

If all was successful, you should execute

```
ls
```

and see a database named `census.db` alongside several `.sql` files. If not, retrace your steps and see if you can determine where you went wrong!

## Schema

In `census.db` you'll find a single table, `census`. In the `census` table, you'll find the following columns:

- `id`, which uniquely identifies each census record
- `district`, which is the name of the census record's district
- `locality`, which is the name of the census record's locality within the district
- `families`, which is the number of families associated with the census record
- `households`, which is the total number of households associated with the census

record (multiple families may live in the same household)

- `population`, which is the population associated with the census record
- `male`, which is the number of people associated with the census record who have identified as male
- `female`, which is the number of people associated with the census record who have identified as female

## Specification

---

In each of the corresponding `.sql` files, write a SQL statement to create each of the following views of the data in `census.db`. Note that, while views can be created from other views, each of your views should stand alone (i.e., not rely on a prior view).

### Rural

In `rural.sql`, write a SQL statement to create a view named `rural`. This view should contain all census records relating to a rural municipality (identified by including “rural” in their name). Ensure the view contains all of the columns from the `census` table.

### Total

In `total.sql`, write a SQL statement to create a view named `total`. This view should contain the sums for each numeric column in `census`, across all districts and localities. Ensure the view contains each of the following columns:

- `families`, which is the sum of families from every locality in Nepal.
- `households`, which is the sum of households from every locality in Nepal.
- `population`, which is the sum of the population from every locality in Nepal.
- `male`, which is the sum of people identifying as male from every locality in Nepal.
- `female`, which is the sum of people identifying as female from every locality in Nepal.

### By District

In `by_district.sql`, write a SQL statement to create a view named `by_district`. This view should contain the sums for each numeric column in `census`, grouped by `district`. Ensure the view contains each of the following columns:

- `district`, which is the name of the district.
- `families`, which is the total number of families in the district.
- `households`, which is the total number of households in the district.
- `population`, which is the total population of the district.

- `male`, which is the total number of people identifying as male in the district.
- `female`, which is the total number of people identifying as female in the district.

## Most Populated

In `most_populated.sql`, write a SQL statement to create a view named `most_populated`. This view should contain, in order from greatest to least, the most populated *districts* in Nepal. Ensure the view contains each of the following columns:

- `district`, which is the name of the district.
- `families`, which is the total number of families in the district.
- `households`, which is the total number of households in the district.
- `population`, which is the total population of the district.
- `male`, which is the total number of people identifying as male in the district.
- `female`, which is the total number of people identifying as female in the district.

## Usage

To test your views as you write them in your `.sql` files, you can run a query on the database by running

```
.read FILENAME
```

where `FILENAME` is the name of the file containing your SQL query. For example,

```
.read rural.sql
```

Keep in mind you can also use

```
DROP VIEW name;
```

where `name` is the name of your view, to remove a view before creating it anew.

## How to Test

While `check50` is available for this problem, you're encouraged to also test your views on your own.

## Correctness

```
check50 cs50/problems/2023/sql/census
```

## How to Submit

---

In your terminal, execute the below to submit your work.

```
submit50 cs50/problems/2023/sql/census
```

## Acknowledgements

---

Data retrieved from Open Data Nepal, [opendatanepal.com \(https://opendatanepal.com/\)](https://opendatanepal.com/).