This part is due Wednesday, November 8 at 5pm.

In Part 1 of the project, your team must design a table schema for the data. By "table schema" I mean the `CREATE TABLE` statements necessary to create database tables that fit the data. You should follow the design principles in Section 4.1 to build a normalized structure for the database that minimizes redundant information. Include primary keys, foreign keys, column types, and any appropriate constraints. It is up to you to decide how many tables you need, their names, and their contents.

Write your `CREATE TABLE` statements in a notebook. Test them out on Azure to ensure they work correctly. You do not need to load any real data into the database yet.

In the notebook, write comments explaining the following: What are the basic entities in your schema? (In Example 4.1, entities were things like songs, record labels, and albums, that each had their own database table.) How did you choose them and what did you do to ensure there is not redundant information in your database?

You may find it useful to make a Git repository to share with your project team, though this is not required. Instead you will turn in your schema and explanation on Gradescope.

The rest of the instructions are split up by the dataset your group is using:

10.2.1 HHS data

The HHS data files have many columns; we won't be interested in all of them here. The data contains the following information:

- A unique ID for each hospital (`hospital_pk` , a string)
- The state the hospital is in (`state` , as a two-letter abbreviation, like PA)
- The hospital's name (`hospital_name`), street address (`address`), city (`city`), ZIP code (`zip`), and FIPS code (`fips_code` , a unique identifier for counties)
- The latitude and longitude of the hospital (`geocoded_hospital_address`), formatted as a string like `POINT(-91 30)` , where the first number is the longitude and the second is the latitude. When you load data, you will need to convert this to a format you can use[1]
- The week this observation is for (`collection_week`)
- The total number of hospital beds available each week, broken down into adult and pediatric (children) beds (`all_adult_hospital_beds_7_day_avg` , `all_pediatric_inpatient_beds_7_` This can change weekly depending on staffing and facilities.
- The number of hospital beds that are in use each week

(all_adult_hospital_inpatient_bed_occupied_7_day_coverage , all_pedi

- The number of ICU (intensive care unit) beds available and the number in use (`total_icu_beds_7_day_avg` and `icu_beds_used_7_day_avg`)
- The number of patients hospitalized who have confirmed COVID (`inpatient_beds_used_covid_7_day_avg`)
- The number of adult ICU patients who have confirmed COVID (`staffed_icu_adult_patients_confirmed_covid_7_day_avg`)

The data is updated weekly. In each weekly file I will provide you, each row will be one hospital, and all of the columns above will be present—so each hospital's address, location, and so on will appear every week.

There are several thousand hospitals in the United States, and this data has been updated weekly for much of the pandemic, so the data contains about 580,000 rows. In raw form, with dozens of columns, it is 257 MB.

You will also be using a hospital quality dataset from the Centers for Medicare and Medicaid Services (CMS). We are interested in the following information in this data:

- A facility ID, which matches the `hospital_pk` in the HHS data
- The facility's name, address, city, state, ZIP code, and county
- The type of hospital and its type of ownership (government, private, non-profit, etc.)
- Whether the hospital provides emergency services
- The hospital's overall quality rating. This quality rating is updated several times a year, and we want to be able to track each version of the quality rating. For instance, we might ask "What was the quality rating of this hospital in 2020?" and compare it to the rating in 2022.

In [ ]:

```
-- Hospital Table
-- This table stores static information about each hospital from HHS data.
-- The primary key, hospital_pk, uniquely identifies each hospital.
-- address, city, zip, fips_code, state, latitude, and longitutde provide de
CREATE TABLE hospital (
    hospital_pk VARCHAR(255) NOT NULL PRIMARY KEY,
    hospital_name VARCHAR(255) NOT NULL,
    address VARCHAR(255) NOT NULL,
    city VARCHAR(255) NOT NULL,
    zip VARCHAR(10) NOT NULL,
    fips_code VARCHAR(20) NOT NULL,
    state CHAR(2) NOT NULL,
    latitude DECIMAL(6,3),
    longitude DECIMAL(6,3)
);

-- Beds Table
-- Captures weekly data for each hospital.
-- The record_id serves as a unique identifier for each record.
-- The foreign key, hospital_pk, links to the hospital table.
-- Includes statistics on bed availability and usage, including COVID-19 spe
CREATE TABLE beds (
CREATE TABLE beds (
    record_id INT SERIAL PRIMARY KEY,
    hospital_pk VARCHAR(255) NOT NULL,
    collection_week DATE NOT NULL,
    all_adult_hospital_beds_7_day_avg INT,
    all_pediatric_inpatient_beds_7_day_avg INT,
    all_adult_hospital_inpatient_bed_occupied_7_day_coverage INT,
    all_pediatric_inpatient_bed_occupied_7_day_avg INT,
    total_icu_beds_7_day_avg INT,
    icu_beds_used_7_day_avg INT,
    inpatient_beds_used_covid_7_day_avg INT,
    staffed_icu_adult_patients_confirmed_covid_7_day_avg INT,
    FOREIGN KEY (hospital_pk) REFERENCES Hospital_Information(hospital_pk)
);

-- Quality Table
-- Expanded to include additional details about each hospital's facilities a
-- Facility_ID is a unique identifier for each quality record.
-- Used Facility_ID to link to the Hospital table.
-- Tracks the hospital's type, ownership, emergency service availability, ar
CREATE TABLE quality (
    quality_id INT SERIAL PRIMARY KEY,
    Facility_ID VARCHAR(255) NOT NULL REFERENCES Hospital_Information(hospit
    hospital_type VARCHAR(255),
    hospital_ownership VARCHAR(255),
    emergency_services BOOLEAN NOT NULL,
    quality_rating INT,
    rating_date DATE NOT NULL
);
```

there is not redundant information by checking that the variables in each table uniquely depend on the primary key for that table.

1. Hospital Table is designed to store static details about hospitals, such as their names and addresses, keeping this unchanging data separate from the variable HHS data to prevent duplication.

2. Beds Table is set up to record weekly fluctuating data like bed counts, linking each entry to the corresponding hospital via a foreign key to the static data. This enables to avoid the repetition of hospital details.

3. Quality tracks the less frequently updated quality ratings of hospitals by also referencing the static hospital data through a foreign key.

This schema distinguishes between permanent and changing information using primary keys for unique identification and foreign keys for relational connections. This setup facilitates easy data updates and enables comprehensive data analysis and reporting.