## 0.1 Question 1: Unboxing the Data

### 0.1.1 Question 1a

As mentioned above, we are working with just one month of data. In the full database (which we don't have access to), tables like the `data` table have billions of rows. What do you notice about the design of the database schema above that helps support the large amount of data and minimize redundancy? Keep your response to at most three sentences.

**Hint:** There is no need to examine any data here. What is a technique learned in lecture 16? Define that technique.

*The database above employs the use of normalization, which is the process of splitting up relations into multiple relations to minimize redundancy. This will minimize the amount anomalies that need to be deleted or updated as "missing" or new data will not be necessary to check across all relations because relations are more decomposed. We are able to know this because there are some elements of the schema that we know as more consistent than others, so values of data can be represented just once instead of multiple times repeatedly.*

### 0.1.2 Question 1d

Do you see any issues with the schema given? In particular, please address the two questions below: - Can you uniquely determine the building given the sensor data? Why? (**Hint:** given a row in the `data` table, can you determine a **uniquely** associated row in `real_estate_metadata` table? Your answer should draw insights from 1b.) - Could `buildings_site_mapping.building` be a valid foreign key pointing to `real_estate_metadata.building_name`? (**Hint:** think about the definition / constraints of a foreign key.)

Please keep your response to **at most three sentences.**

*You cannot determine the building from sensor data because, as demonstrated in 1b, there are many rows that map from the real_estate_metadata table to a singular row in the data table, making it impossible to uniquely determine the building. Foreign keys must not create dangling tuples, so buildings_site_mapping.building cannot point to real_estate_metadata.building_name as there are building names not found in real_estate_metadata that are f.*

## 0.2 Question 3: Entity Resolution

### 0.2.1 Question 3a

There is a lot of mess in this dataset related to entity names. As a start, have a look at all of the distinct values in the `units` field of the `metadata` table. What do you notice about these values? Are there any duplicates? **Limit your response to one sentence.**

*Many of the values of the unit column of the metadata are duplicates with different cases (e.g. kWh vs. KWH) and some of them are abbreviations of each other (e.g. gal vs Gallons).*

### 0.2.2 Question 3d

Moving on, have a look at the `real_estate_metadata` table—starting with the distinct values in the `location` field! What do you notice about these values? Keep your response to at most two sentences.

*Some of the values from the location field have random spacings in between letters that are not originally supposed to be there. For example, "SAN DSAIENG O" should be equivalent to "SAN DIEGO" and "FRANCISC O" should reference "SAN FRANCISCO."*