0.1 Question 1: Unboxing the Data

0.1.1 Question 1a

Note that the `data` table, in the full database, is billions of rows. What do you notice about the design of the database schema that helps support the large amount of data?

Hint: There is no need to examine any data here. What is a technique learned in lecture 15? Define that technique.

The data is taking normalization. Normalization is creating a table and establishing relationships between the small bits of data that are stored in smaller tables. Normalization gets r

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.)

We can not uniquely determine the building given in the sensor data. There is a many to many relationship because there are duplicate keys pointing to the real_estate_metadata.building_name so there can not be a valid foreign key.

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?

Type your answer here, replacing this text.

0.2.2 Question 3b

Sometimes, entity resolution is as simple as a text transformation. For example, how many unique `units` values are there, and how many would there be if we ignored case (upper vs. lower case)? Your output should be a table with one row and two columns; the first column should contain the number of unique `units` values, and the second column should contain the number of unique `units` values if we ignored case. The two columns can have arbitrary names.

```
In [29]: %sql select * from metadata LIMIT 10
```

```
   postgresql://jovyan@127.0.0.1:5432/template1
 * postgresql://jovyan@127.0.0.1:5432/ucb_buildings
10 rows affected.
```

```
Out[29]: [('fb5267b9-8cf4-5710-979e-5ab7d617e6e5', 'Main Electric Meter (kWh)', '2000 Carleton Street',
          ('4e7e5872-a3f7-5300-8433-b4dfd0254cb1', 'Main Electric Meter Demand (kW)', '2000 Carleton St
          ('a2fbb537-3e9c-5c47-94c9-2b5cc3735f48', 'Main Electric Meter Min (kW)', '2000 Carleton Stree
          ('4eb9d406-9aeb-56ab-aa9d-aaa368679e1f', 'Main Electric Meter Instantaneous (kW)', '2000 Carl
          ('3c44bdad-d1a9-5155-b786-fb2ce6e1e76e', 'Main Electric Meter Max (kW)', '2000 Carleton Stree
          ('1ec76d24-e7a8-5b02-ad54-1e0c00c3f408', 'Voltage phase A-B (Volts)', '2200 Bancroft Way', 'V
          ('70c50787-e685-5ad3-84db-3094bda63bb6', 'Voltage phase B-C (Volts)', '2200 Bancroft Way', 'V
          ('ab8cc53d-e9ad-51b9-9383-0dc812d5a7da', 'Apparent Power (kVA)', '2200 Bancroft Way', 'kVA'),
          ('bca43b3a-d3c9-53e8-ab76-c473722e065c', 'Voltage phase B-N (Volts)', '2200 Bancroft Way', 'V
          ('27ada1cb-1e7d-52b0-b768-164090b9d821', 'Present Demand (subinterval) (kW)', '2200 Bancroft
```

```
In [30]: %%sql result_3b <<
         SELECT count(distinct(units)) as unique, count(distinct(lower(units))) as lower  from metadata
```

```
   postgresql://jovyan@127.0.0.1:5432/template1
 * postgresql://jovyan@127.0.0.1:5432/ucb_buildings
1 rows affected.
Returning data to local variable result_3b
```

0.2.3 Question 3c

Arguably we shouldn't care about these alternative unit labels, *as long as each sensor class uses a single value of* ***units*** *for all its sensor ids*. After all, maybe the capitalization means something to somebody!

Write a SQL query that returns single row with one column of value `true` if the condition in italics above holds, or a single row with one column of value `false` otherwise. Please do not hard code this query - we reserve the right to penalize your score if you do so.

```
In [34]: %%sql result_3c <<
         with unit as(
             select id, count((units))
             from metadata
             group by id),

         newCount as(
             select *,
             case when count = 1 then true
             when count = 0 then false
             end as t
             from unit)

         select (count(newCount)  filter (where t = true) = 9509) from newCount
```

```
   postgresql://jovyan@127.0.0.1:5432/template1
 * postgresql://jovyan@127.0.0.1:5432/ucb_buildings
1 rows affected.
Returning data to local variable result_3c
```

```
In [35]: result_3c
```

```
Out[35]: [(True,)]
```

0.2.4 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?

Type your answer here, replacing this text.