CEGE0052 – Assignment 2 - Creating a Database

Contents

Introduction

__Submission Date:__

See Moodle

__Submission Method:__

To submit the assignment you need to create FOUR files in total

1. Upload 3 files via the following website:

http://cege0052.cege.ucl.ac.uk/cege0052/assignment2.html

- A file named *createtable_ucxxxxx.txt* which contains the scripts to create the tables
- A file named *createconstraints_ucxxxxx.txt* which contains the scripts to create the constraints
- A file named *insertdata_ucxxxxx.txt* which contains the scripts to insert the data

2. Additionally, you need to upload a PDF containing a QGIS (map) screenshot of your data, to Moodle/Turnitin. Make sure you include a cover sheet in the PDF.

This assignment is worth 5% of the marks for the module.

- *If you have questions about this assignment, please post them on Moodle*
  - *That way everyone is given the same information*
  - *That way I remember what I've said to you and don't mark you down for doing something that I wasn't expecting*
  - *Any questions should be generic – as this is an assessment which will gauge how much you've learned during the module I won't be able to solve very specific assignment-related problems for you.*

- *This is a digital submission – it is up to you to ensure that the file you upload to Moodle is not corrupt in any way (you might be able to do this by downloading the uploaded file to check it)*

Plagiarism

You should work on assignments on your own, using your own user account (username =userX, password = userXpassword).

The database has a log which records everything you do – every time you log on, every query you run, every error you make. If necessary, we can use this log to find out who is doing what and when – in particular when you are logged on and what mistakes you make. I can also tell if you just copy/paste someone else's scripts rather than do the work yourself.

*** Read these instructions a few times before starting to make sure you are clear on what you need to do ***

Background

A local transport company wish to create a database to better manage their bus stations, as there have been complaints about cleanliness and poor air quality. They have already designed the database, and have asked you to build it and populate it with data.

The design has been documented as an ER diagram, which shows

- The tables to be created
  - The columns that should be in the tables and what type of data should be stored
- The constraints – including:
  - Primary keys
  - Foreign keys
  - Unique constraints
  - Check constraints

They have also provided you with a spreadsheet with some sample data which you should use to test your design.

Additional Information:

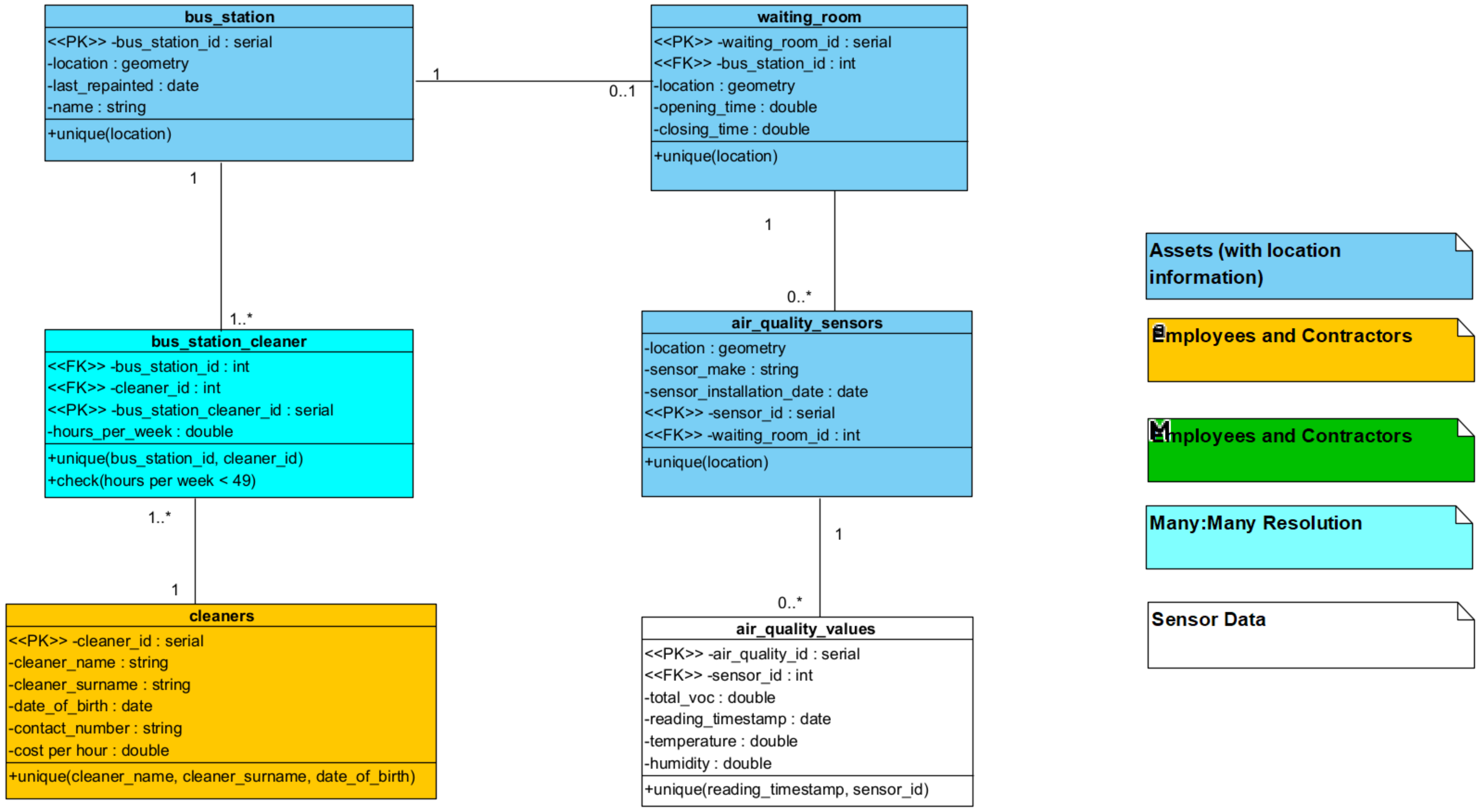
- All location information (geometry) is 2D
- All the attributes are REQUIRED - i.e. they should all have NOT NULL (except for the geometry columns)
- It is up to you to estimate the length of any character varying/string columns. You can do this by looking at the data in the spreadsheet.

** While you are building the database, another team is developing software that will use this database, so it is important that you build the database EXACTLY as the given specification **

Hints:

- As a beginner it is expected that you will make some mistakes when first creating SQL. Common problems that you should check for include:
  - Missing or extra brackets
  - Missing or extra commas
  - Missing or extra quotation marks
  - Spelling – e.g. you spelled a table differently in different places
- To make your life easier you should write a tiny bit of SQL at a time and test it.

- All the SQL must be generated manually -i.e. written by hand – and for each part of the assignment you should create a separate script file using – e.g. – notepad++
- It does take time to get used to the error messages generated by PGAdmin. As an MSc student you are expected to be an independent problem solver so do make good use of online information – e.g. Google – if you are stuck.
- If you are totally stuck then you can also post on Moodle


bus_station
<<PK>> -bus_station_id : serial
-location : geometry
-last_repainted : date
-name : string
+unique(location)
1
0..1
waiting_room
<<PK>> -waiting_room_id : serial
<<FK>> -bus_station_id : int
-location : geometry
-opening_time : double
-closing_time : double
+unique(location)
1
1..*
bus_station_cleaner
<<FK>> -bus_station_id : int
<<FK>> -cleaner_id : int
<<PK>> -bus_station_cleaner_id : serial
-hours_per_week : double
+unique(bus_station_id, cleaner_id)
+check(hours per week < 49)
1..*
1
cleaners
<<PK>> -cleaner_id : serial
-cleaner_name : string
-cleaner_surname : string
-date_of_birth : date
-contact_number : string
-cost per hour : double
+unique(cleaner_name, cleaner_surname, date_of_birth)
1
0..*
air_quality_sensors
-location : geometry
-sensor_make : string
-sensor_installation_date : date
<<PK>> -sensor_id : serial
<<FK>> -waiting_room_id : int
+unique(location)
1
0..*
air_quality_values
<<PK>> -air_quality_id : serial
<<FK>> -sensor_id : int
-total_voc : double
-reading_timestamp : date
-temperature : double
-humidity : double
+unique(reading_timestamp, sensor_id)
Assets (with location information)
Employees and Contractors
Employees and Contractors
Many:Many Resolution
Sensor Data

Part 1 – DDL – Creating the Database

For this first part of the assignment you are required to create the tables for the database. There are 6 tables in total, three of which have geometry columns (location).

1. Follow the instructions on Moodle to log in to the database using PG Admin 4 ('How do I' tab)

2. Create a file in notepad++ or sublime a similar text editor called *createtable_ucxxxxx* where *ucxxxxx* is your UCL login

3. Start by creating the *ucxxxxx* schema for this database

   drop schema if exists ucxxxxx cascade;
   create schema ucxxxxx;

4. Create all the tables – remember to include the schema name. All table names and column names should be lower case[1] and you should use _ (underscore) to separate words - i.e. don't use spaces.

5. Add the geometry columns separately using *addgeometrycolum*. All geometry should have SRID of 27700 (British National Grid)

Checking your progress

By the end of this first part of the assignment you should have 6 tables in your ucxxxxx schema. Make sure you can see all of them in PGAdmin

1. cleaners
2. bus_station_cleaner
3. bus_station
4. air_quality_sensors
5. air_quality_values
6. waiting_room

Hints:

- *Make sure you spell things correctly – as they are on the ER Diagram.*
- *If you get something wrong:*
  - *you can use the DROP command to make changes (drop the table, drop the column)*[2]
  - or you can run the script again and completely drop the schema[3]
- You can find examples of create table scripts on Moodle (relating to the University Asset Management case study we're using in class)
- For the *createtable_ucxxxxx.txt* script, make sure that the first lines of this script are
  drop schema if exists ucxxxx cascade;
  create schema ucxxxxx;
- Check that the table names and column names are 100% identical to those in the diagram

[1] PostgreSQL doesn't really work well with table and column names that contain capital letters.
[2] A reminder – the DROP command completely deletes something – so don't use this if you have data in the tables otherwise data is lost
[3] in a real database you would never want to do this as if you have data in the database and it isn't backed up then the data will be lost when you drop the schema

Part 2 – Constraints – Creating Rules for the Database

For the second part of the assignment you are required to constraints for the database. These are the rules that will ensure that data created in the database is of good quality (no missing information, no incorrect information, no duplicate information).

1. Follow the instructions on Moodle to log in to the database using PG Admin 4 ('How do I' tab)

2. Create a file in notepad++ or sublime a similar text editor called *createconstraints_ucxxxxx.txt* where *ucxxxxx* is your UCL login

3. Create the primary keys

4. Create the foreign keys

5. Create the check constraints

6. Create the unique constraints

Checking your progress

By the end of this second part of the assignment you should have SQL to create the following constraints (NB as we're using NOT NULLs more constraints will be created by the system)

1. 6 primary keys
2. 5 foreign keys
3. 6 unique constraints
4. 1 check constraint

You can check that they have all be created by drilling down into the tables in PGAdmin

Hints:

- *If you get something wrong: you can use the ALTER TABLE DROP command to make changes - i.e. to drop the constraints* [4]
- or you can run the script again and completely drop the schema (NB: in a real database you would never want to do this as if you have data I the database and it isn't backed up then the data will be lost when you drop the schema)
- Make sure you create all the primary keys before any of the foreign keys – otherwise you will get an error as a foreign key must reference an existing primary key
- You can find examples of constraint creation scripts on Moodle (relating to the University Asset Management case study we're using in class)

[4] A reminder – the DROP command completely deletes something. Dropping a constraint doesn't delete any data though.

Part 3 – DML – Inserting Data

You have been given a spreadsheet which contains the data that the transport company have provided to allow you to check your work.   For the final part of the assignment you should create the required INSERT scripts for each table.

1. Follow the instructions on Moodle to log in to the database using PG Admin 4 ('How do I' tab)

2. Create a file in notepad++ or sublime a similar text editor called *insertdata_ucxxxxx.txt* where *ucxxxxx* is your UCL login

3. Because of the PRIMARY/FOREIGN KEY relationships, the statements should be created in this order:
    - bus_station
    - waiting_room
    - air_quality_sensors
    - air_quality_values
    - cleaners
    - bus_station_cleaner

Checking your progress

By the end of this second part of the assignment you should have the following data

- bus_station - 1 row
- waiting_room – 1 row
- air_quality_sensors – 3 rows
- air_quality_values – 5 rows
- cleaners – 3 rows
- bus_station_cleaner – 3 rows

You can check that they have all be created by right clicking on the tables in PGAdmin and selecting VIEW/EDIT DATA > ALL ROWS

You should also check that your data is correct by visualising it in QGIS – see Appendix 3.

Hints:

- You can find sample data creation scripts (relating to the in class worked example on Asset Management) on Moodle
- Be careful when writing the SQL to create geometry – the number of brackets for points and polygons in ST_GEOMFROMTEXT is different and you should also be careful about missing quotation marks
- Although it might be tempting to copy/paste the INSERT statements be careful when you are doing so – you need to make sure that you don't insert the wrong data or cause a problem with a CONSTRAINT - e.g. if you try to create two air quality sensors in the same location you will violate the UNIQUE constraint on location for that table.
- See Appendix 3 for some examples of what to do when the 'cross reference to'

Part 4 – Visualising the Data in QGIS

As with Assignment 1 and any data, it is important to validate that you have created the data correctly. We will see how to use select statements to do this later on in the module, but as we're dealing with map/spatial data an easy way to validate the data is to create a map! You can use this to check that the data makes sense –

- Do the three features draw correctly?
- Is the waiting room inside the bus station?
- Are the sensors inside the waiting room?

We'll use GIS (geographical information systems) software – QGIS – to create this map.

1. Open QGIS and connect to the database by following the instructions in the *Connecting QGIS 3.2.3 to PostGIS* guide on the 'how do I' tab in Moodle

2. Open the three tables you created above, and put them on the map. Make sure they are stacked so that you can see all the map data (i.e. the building at the bottom, then the waiting room, then the air quality sensors on top)

3. If you wish you can change the symbology – colours and styles – by right clicking on the layer[5] and selecting PROPERTIES and then SYMBOLOGY

4. Take a screenshot of your map (you are not required to produce a professional quality map for this assignment – the screenshot is just to demonstrate that you have created all the data correctly)
    a. The screenshot should clearly show the layers list and the map data
    b. The screenshot should also include an Open Street Map basemap – make sure that your data is visible but also that it is possible to identify the location with respect to major towns and cities (i.e. don't zoom in too far)

5. Paste the screenshot into a Word document and create a PDF for upload to Moodle – you will need a cover sheet of some sort.

[5] A *layer* is a GIS word for a table that has some data that can be mapped.

Appendix 1 – Checking Your Work in PGAdmin

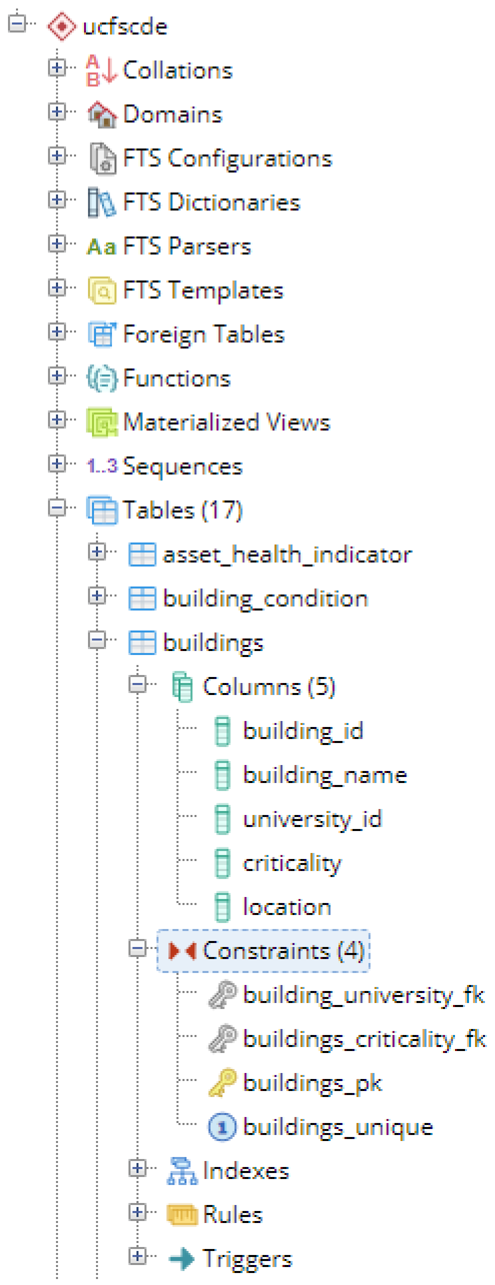
1. To see your work, double click on the database and then navigate down through the options as follows:

   database > schemas

2. If you don't see your ucxxxxx schema, right click on the SCHEMAS and select REFRESH.

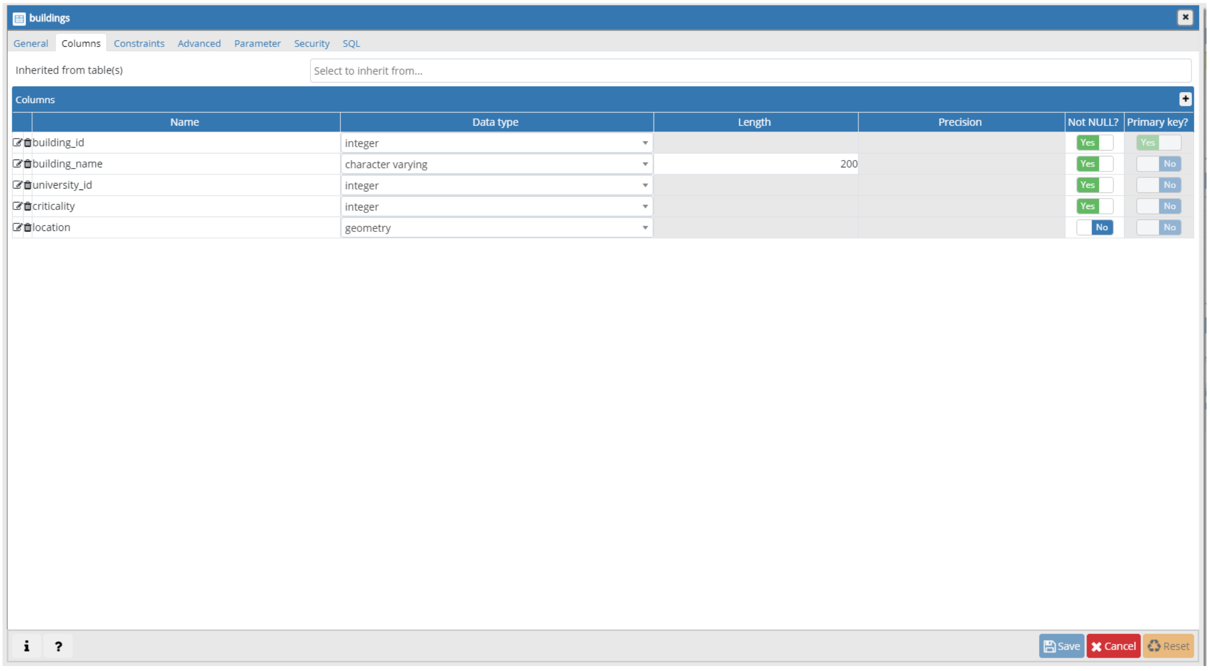
3. Double click on your ucxxxxx schema and drill down further – you will see a list of any tables that are currently stored in the schema.

4. Double-click on each of the tables to explore their structure. PostgreSQL gives you a description of each of the columns (fields) in the table, and you can also see any constraints that have been added (such as primary keys and foreign keys). (Your screen may show schemas and tables that are different to the one shown below)



```
ucfscde
    Collations
    Domains
    FTS Configurations
    FTS Dictionaries
    FTS Parsers
    FTS Templates
    Foreign Tables
    Functions
    Materialized Views
    Sequences
    Tables (17)
        asset_health_indicator
        building_condition
        buildings
            Columns (5)
                building_id
                building_name
                university_id
                criticality
                location
            Constraints (4)
                building_university_fk
                buildings_criticality_fk
                buildings_pk
                buildings_unique
            Indexes
            Rules
            Triggers
```

5. You can also right-click on a table and select PROPERTIES to see further details



Inherited from table(s) | Select to inherit from...

Columns

Name	Data type	Length	Precision	Not NULL?	Primary key?
building_id	integer			Yes	Yes
building_name	character varying	200		Yes	No
university_id	integer			Yes	No
criticality	integer			Yes	No
location	geometry			No	No

Appendix 2 – Cross Referenced Data

Note that coordinate values here are only examples!

As the ID values assigned by a database are completely arbitrary you can't simply type them in when inserting data as you don't know them in advance, so you need to work them out. This is particularly the case when you have millions of records in your database.

Working out the required ID involves putting a 'nested' select statement into the SQL to work out the ID value we need, based on the information we have.

We haven't done much with *select* statements yet so to get you started, here are some examples for your INSERT scripts:

Air Quality Sensors
(find the waiting_room_id that the air quality sensor is inside)

```
insert into ucfscde.air_quality_sensors
(location, sensor_make, sensor_installation_date, waiting_room_id)
values
(st_geomfromtext('POINT(500051 500051)',27700),'UCL Sensors','20190202', (select
waiting_room_id from ucfscde.waiting_room b
where st_intersects(b.location, (st_geomfromtext('POINT(500051 500051)',27700))))) ;
```

Bus Station Cleaners
(find the cleaner_id with the given name, surname and date of birth)

```
insert into ucfscde.bus_station_cleaner
(bus_station_id, cleaner_id, hours_per_week)
values
((select bus_station_id from ucfscde.bus_station where name = 'Main Bus Station'), (select cleaner_id
from ucfscde.cleaners where cleaner_name = 'Sarah' and cleaner_surname = 'Smith' and
date_of_birth = '1992-05-23'), 25);
```

Air Quality Values
(find the sensor_id based on extracting coordinates of the sensor location)

```
insert into ucfscde.air_quality_values
(sensor_id, total_voc, reading_timestamp,temperature, humidity)
values
((select sensor_id from ucfscde.air_quality_sensors b where st_astext( b.location) = 'POINT(500051
500051)'), 300, '2020-04-18',33.5,75),
```