SQL and NoSQL Script Snippets

I have chosen to create the document of screenshots to demonstrate scripted examples instead of uploading it to a public repository because some scripts:

- are part of my final MSc project
- have been marked but as part of a module assessment, but I want to avoid plagiarism from students who are yet to complete the assessment in my cohort.
- may form part of a repeated syllabus to avoid plagiarism.

If required, I can send the complete code or create a private online repository for viewing.

Contents

SQL and JavaScript: Developing API to support a Mobile and Web Geospatial Application

Developed a **Restful API** using **node.js** using the **HTTP methods GET** and **POST** to input data and retrieve data from a **PostGIS** database (Figure 1). To support a location-based web application that displayed the data using the JavaScript library, **Leaflet**.

C:\Users\titil\AppData\Local\Temp\BvSshSftp\Bv003986\Bv004a8a\geoJSON.js - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

geoJSON.js

```
4
5     var express = require('express');
6     var pg = require('pg');
7    var geoJSON = require('express').Router();
8     var fs = require('fs');
9   // Database login infomormation
10    var configtext = ""+fs.readFileSync("/home/tayeni/certs/postGISConnection.js");
11
12    // now convert the configruation file into the correct format -i.e. a name/value pair array
13    var configarray = configtext.split(",");
14    var config = {};
15    for (var i = 0; i < configarray.length; i++) {
16        var split = configarray[i].split(':');
17        config[split[0].trim()] = split[1].trim();
18    }
19    var pool = new pg.Pool(config);
20    console.log(config);
21
22
23
24  id;
25  2]
26
27  eturns the users id and places it into a global variable
28        geoJSON.get('/getUserId', function (req,res) {
29    pool.connect(function(err,client,done) {
30            if(err){
31                console.log("not able to get connection "+ err);
32                res.status(400).send(err);
33            }
34        var query = "select user_id from ucfscde.users where user_name = current_user;";
35
36            client.query(query ,function(err,result) {
37                done();
38                if(err){
39                    console.log(err);
40                    res.status(400).send(err);
41
42                }
43                res.status(200).send(result.rows);
44                console.log(result.rows);
45                 usersIdentification=JSON.stringify(result.rows);
46                 var teg = /\d+/g;
47
48            id= usersIdentification.match(teg);
49                    console.log('global var'+id);
50                    id = parseInt(id, 10);
51                    console.log(id);
52
53            });
54        });
55    });
56
57
58  2]
59  eturns a GeoJSON of all the questions
60  uploaded from the quiz setting app to the database
61
62     geoJSON.get('/geoJSONUserId/user_id', function (req,res) {
63    pool.connect(function(err,client,done) {
64            if(err){
65                console.log("not able to get connection "+ err);
```

geoJSON.js

```
325
326                 geoJSON.get('/fiveClosestQuestions/latitude/longitude', function (req,res) {
327     pool.connect(function(err,client,done) {
328             if(err){
329                 console.log("not able to get connection "+ err);
330                 res.status(400).send(err);
331             }
332
333
334             // REMOVE HARD CODING
335     var longitude= req.query.longitude;
336     var  latitude= req.query.latitude ;
337     console.log(latitude);
338     console.log(longitude);
339
340
341         var querystring ="SELECT 'FeatureCollection' As type, array_to_json(array_agg(f)) As features  FROM ";
342  querystring+="(SELECT 'Feature' As type , ST_AsGeoJSON(lg.location)::json As geometry,  ";
343  querystring+="row_to_json((SELECT l FROM (SELECT id, question_title, question_text, answer_1, answer_2, answer_3, 
344  querystring+=" )) As properties";
345  querystring+=" FROM   (select c.* from cege0043.quizquestions c ";
346  querystring+="inner join (select id, st_distance(a.location, st_geomfromtext('POINT(" +longitude +" " + latitude+"
347  querystring+="from cege0043.quizquestions a ";
348  querystring+="order by distance asc ";
349  querystring+="limit 5) b ";
350  querystring+="on c.id = b.id ) as lg) As f";
351
352
353         console.log(querystring);
354
355
356          client.query(querystring,function(err,result) {
357              done();
358              if(err){
359                  console.log(err);
360                  res.status(400).send(err);
361              }
362              res.status(200).send(result.rows);
363
364          });
365      });
366  });
367                 // [2]
368 Returns the top 5 most difficult questions in the database
369                 geoJSON.get('/fiveDifficultQuestions', function (req,res) {
370     pool.connect(function(err,client,done) {
371             if(err){
372                 console.log("not able to get connection "+ err);
373                 res.status(400).send(err);
374             }
375
376
377         var querystring ="select array_to_json (array_agg(d)) from ";
378  querystring+="(select c.* from cege0043.quizquestions c ";
379  querystring+="inner join  ";
380  querystring+="(select count(*) as incorrectanswers, question_id from cege0043.quizanswers where ";
381  querystring+="answer_selected <> correct_answer ";
382  querystring+="group by question_id ";
383  querystring+="order by incorrectanswers desc ";
384  querystring+="limit 5) b ";
385  querystring+="on b.question_id = c.id) d;";
386
```

Figure 1 Code snippet displaying the retrieval of data as GeoJSONs. The data was retrieved from a PostGIS database.

SQL: Asset Management Database

Primary and Foreign Keys | 2D and 3D Geometries | Constraints

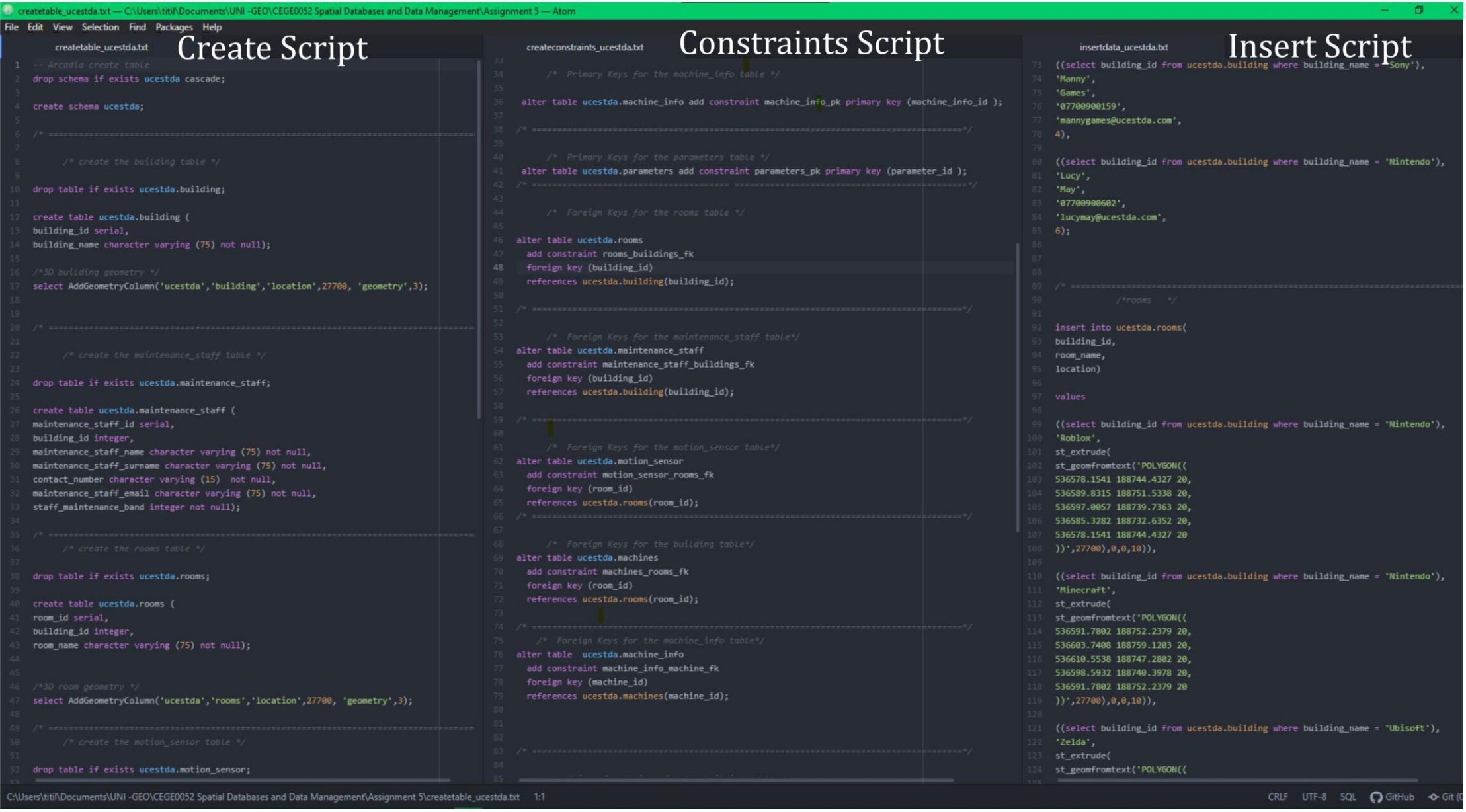

Figure 2 Demonstration of creating a schema and inserting data.

SQL Queries

Nested Queries| Common Table Expressions | Joins | Spatial Joins

C:\Users\titil\Documents\UNI -GEO\CEGE0052 Spatial Databases and Data Management\Assignment 5\SQL CODE.txt • - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

SQL CODE.txt

```sql
 ##############################
 "Which rooms and machines are and are not covered by
 motion_sensor ? The null values are machines that are not covered
 by motion_sensor."
 ucestda.parameters ;ucestda.motion_sensor;ucestda.machines;ucestda.rooms
 ############################## 1 rooms motion_sensor
with
-- find buffer value for motion sensor
buffer_radius as (select parameter_value as buffer
 from ucestda.parameters  where parameter_name = 'buffer'),
-- find the machines which intersect this motion sensors radius
 motion_sensor_covered as(
 select b.motion_sensor_id, a.machine_id,a.machine_name,c.room_name
 from ucestda.motion_sensor b
LEFT JOIN  ucestda.machines a
 on st_intersects((st_buffer(b.location,(select buffer from buffer_radius) )), a.location)
LEFT JOIN ucestda.rooms c
 on st_contains(st_envelope(c.location),(a.location))),
 -- returns the machine name , room it is is and id
  all_info as (select y.machine_name,x.room_name,
  y.machine_id
 from ucestda.machines y
 left join ucestda.rooms x
 on x.room_id = y.room_id)
-- compares the results of the two temporary table
-- returning null if the machine is not in the motion sensor area

 select m.*,i.motion_sensor_id
from all_info m
  left join motion_sensor_covered i
  on m.machine_id = i.machine_id order by machine_id ;



 # 3D ROOM AND 2D MACHINE
 select a.*,c.room_name
 from ucestda.machines a LEFT JOIN ucestda.rooms c
 on st_contains(st_envelope(c.location),(a.location));


 ##########################################################
 "SQL for most expensive costing machines that need to maintenance"

 select purchase_cost,last_maintenance_date from  ucestda.machine_info order
 by last_maintenance_date,purchase_cost ;

 select purchase_cost,last_maintenance_date from  ucestda.machine_info
 where last_maintenance_date <
 PART('day', current_date::timestamp -
last_maintenance_date::timestamp)

"Machines most in need of maintenance"
(SELECT (current_date -last_maintenance_date) as days_since_last_maintenanceed,
 last_maintenance_date,
 machine_id,
purchase_cost
  from  ucestda.machine_info where
 (current_date -last_maintenance_date) >365 order by last_maintenance_date)


WITH
 needs_maintenance as
(SELECT (current_date -last_maintenance_date) as days_since_last_maintenanceed,
```

SqlQuizAns.sql

```sql
" Write a query that joins the window information to the criticality information to see the
criticality of each window."


select a.* ,b.*,c.*
from ucfscde.rooms a inner join ucfscde.windows b inner join ucfscde.window_condition c
on  a.room_id = b.room_id
on b.window_id = c.window_id;
select a.* ,b.*,c.*
from ucfscde.rooms a inner join ucfscde.windows b
on a.room_id = b.room_id
inner join ucfscde.window_condition c
on b.window_id = c.window_id;



"Write a query to calculate the cost of a total refurbishment of the classrooms in each building, using
    information from the parameters table."

With
classroom_cost as (select * from ucfscde.parameters
where parameter_type ='cost'and
parameter_name = 'rooms' and parameter_subname='total refurbishment'),

classrooms as (select * from ucfscde.rooms where room_use = 'classroom'),

 classroom_cost_calculation as (select a.*, b.parameter_value, b.parameter_units,
 st_area(a.location)* b.parameter_value as refurb_cost from classrooms a, classroom_cost b)

 select c.*, d.*
from ucfscde.buildings c INNER JOIN classroom_cost_calculation d
on c.building_id = d.building_id;



"Write a query to join the sensors to the university. Use st_intersects, and a LEFT JOIN with the sensors table
    as table a first and then the university as table b. Select all columns from both tables. Do not use WITH"

select a.*, b.*
from ucfscde.temperature_sensors a LEFT JOIN
ucfscde.university b
on st_intersects(a.location,b.location);


"Write a query to link the latest condition report for each room with the details of that room. You will need
    to use ORDER BY and DISTINCT ON to get the latest condition report for each room. Use a WITH statement to
    get the latest condition report for each room, with the temporary table called latest_condition_report.
    Then use a LEFT join with the rooms table first with alias a and the latest_condition_report table second
    with alias b."


WITH
latest_condition_report as (select distinct on (room_id) room_id, ceiling_condition, wall_condition,
    windows_condition, doors_condition,
            furniture_equipment_condition, air_conditioning_condition, sockets_condition,
                lighting_and_switches_condition,
            report_date, user_id from ucfscde.room_condition order by room_id ,report_date desc)
select a.*,b.*
from ucfscde.rooms a left join
latest_condition_report b
on a.room_id = b.room_id;
```

Line 98, Column 35 Spaces: 2 SQL

Figure 3 The left script queries the schema in Figure 2, and the right script is part of a Spatial Databases module quiz.

FME, NoSQL and JavaScript: Creating Spatial Operators (Polygon Area) in MongoDB

This page demonstrates how to use **JavaScript** to create a **spatial operator** in **MongoDB**. First, Ordnance Survey Building footprint **shapefiles** downloaded from **Digimap** are read into **FME**, an area attribute was added for later data validation then uploaded to a **MongoDB Collection** (Figure 5). Finally, the function is tested in a **JavaScript IDE** (Figure5), then the **aggregation operator** is used to test the function in MongoDB(Figure 6), and the solution can be compared to the stored attribute.

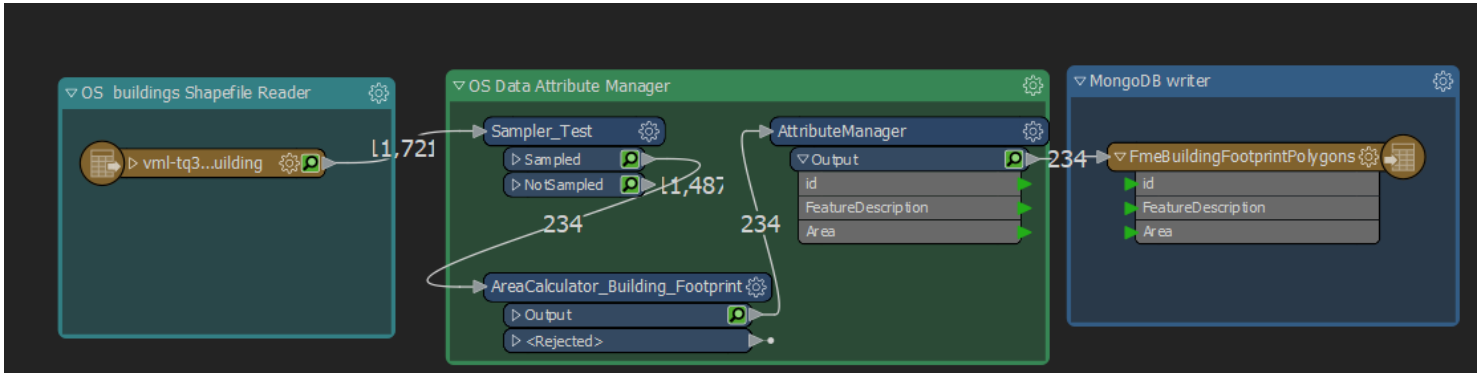

Figure 4 Simple tester FME workbench reading in and Ordnance Survey polygons and writing it into a MongoDB collection.

```
function polygonArea(geojson){
var coordArray = geojson.geometry.coordinates[0][0];
  var  numPoints= coordArray.length;
  var X= []; var Y =[];

  for (let i = 0; i < coordArray.length; i++) {
    console.log(i, coordArray[i],
      X.push(coordArray[i][0]),
      Y.push(coordArray[i][1])

    );
  }


var area = 0;  var j; j = numPoints-1;

for ( var k =0; k<numPoints; k++)
{ area +=  (X[j]+X[k]) * (Y[j]-Y[k]);
  j = k;
}
  return area/2;
}
```

Figure 5 Javascript code to calculate the area of a regular polygon using coordinates.

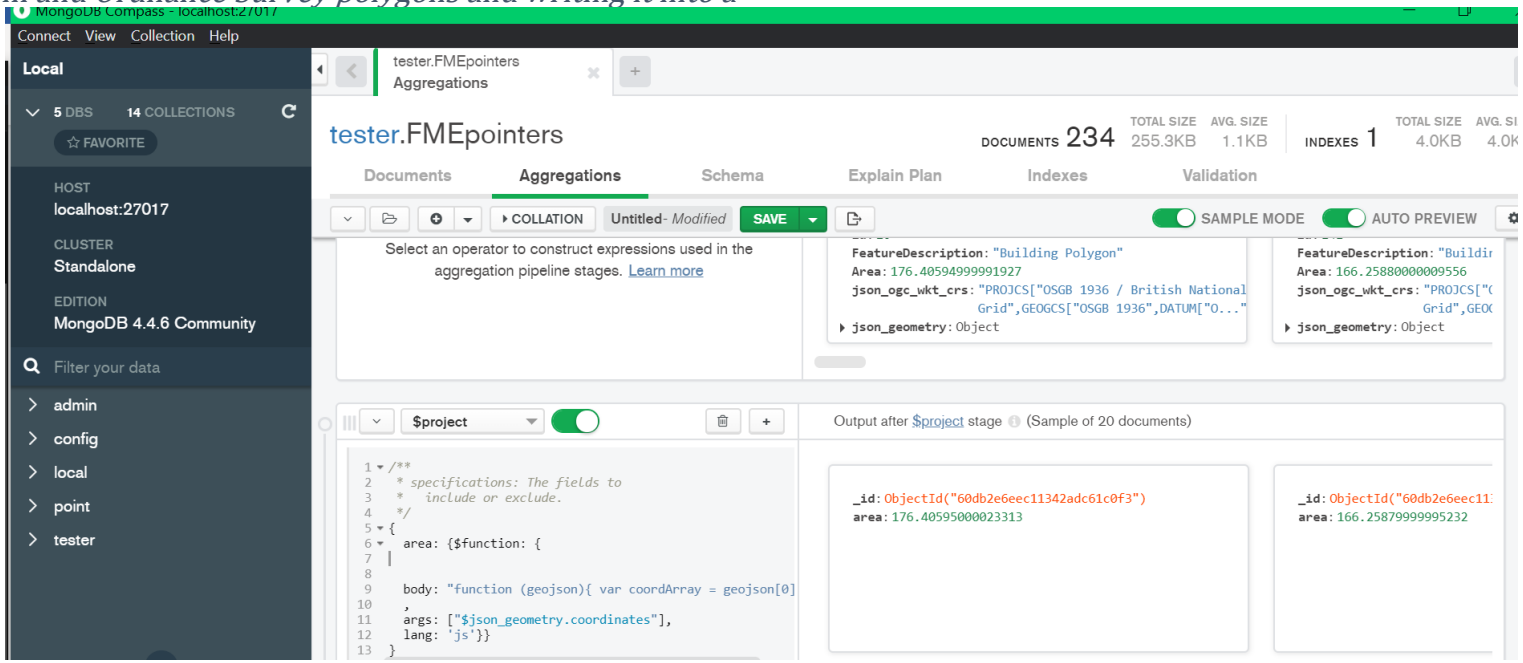

Figure 6 Results of querying the area of the building footprints in MongoDB using the Javascript function in a query. The attribute "Area" is used to validate the solution.