Table of Contents

Part 1- Data Management: Spatial Queries

Create a script file for this exercise called practical4.txt and also a schema called practical4

Exercise 1- Simple Spatial SQL

So far, we have seen queries that only focus on the attributes of the data we have been using, but do not operate on the geometry. However, PostgreSQL (with the PostGIS extension) also incorporates a wide range of geometry functions that you usually find in a desktop GIS such as ArcGIS. Note that in PostgreSQL/PostGIS, spatial functions begin with the letters ST_. A number of examples are included here, and a full list can be found at:

http://postgis.refractions.net/docs/reference.html

(see in particular the section on Spatial Relationships)

Note: In the following exercises your Column Names may be different – e.g. geom instead of geom, id instead of id. Change the SQL as appropriate!

1. Find the total length of roads in the London_highway subset dataset

   ```
   SELECT SUM(ST_LENGTH(geom))
   FROM public.london_highway;
   ```

2. Now run the following query:

   ```
   SELECT SUM(ST_LENGTH(geom)) as length_in_degrees
   FROM public.london_highway
   ```

   Note that we are using an *alias* for the result, which allows us to give the column a more sensible name (in this case length_in_degrees) that that provided by PostgreSQL/PostGIS

3. We can now have a closer look at the geometry which has been used to calculate the length by using the following query, which gives us the Well Known Text that makes up the geometry[1]:

```
SELECT ST_AsEWKT(geom)
FROM public.london_highway
WHERE "id" = (select min(id) from public.london_highway);
```

4. In the above queries, we used the data 'as is' in the database. However, as we have seen earlier, this dataset is not projected (from the Well Known Text you can see that it is stored as the WGS 84 Latitude/Longitude decimal degrees, SRID 4326, which is the default system used by the Global Positioning System). Given that the units for the SRID are decimal degrees, the total length returned is also in decimal degrees, which is not a unit of measurement in common usage. The data must therefore be transformed into a projected (flattened) system before the length calculation is made. This is done using an ST_TRANSFORM function as shown:

```
SELECT SUM(ST_LENGTH
(ST_TRANSFORM(geom,27700))) as Length_in_m
FROM public.london_highway
```

5. We can use an ST_AREA query to find the total area of all the Local Authorities. However, in this case the data is already projected in the correct SRID, so no ST_TRANSFORM is required.

```
SELECT SUM(ST_Area(geom)) as Area_in_sq_m
FROM public.london_counties;
```

6. Find the perimeter of the london_counties:

```
SELECT name, ST_PERIMETER(geom)
FROM public.london_counties;
```

Exercise 2 – Combining Spatial and Non-Spatial Queries

One of the strengths of a relational database is the ability to combine data from multiple sources and, using this data, perform various types of analysis. This exercise will illustrate how it is possible to combine the spatial and non-spatial SQL we have looked at above into one query.

1. Find the total length of london_highway 'primary' roads (this query may take some time to run)

```
SELECT SUM(ST_LENGTH(ST_TRANSFORM(geom, 27700)))
FROM public.london_highway
WHERE type= 'primary';
```

[1] You will also see ST_AsText – this doesn't return the SRID

2. Now find the total length of residential roads in the uk_highway dataset. Remember to use the appropriate ST_TRANSFORM to project the dataset to a flat coordinate system.

3. Find the total area of the Counties having area > 2000000 sq m.

```
SELECT SUM(st_area(geom))
FROM public.london_counties
WHERE ST_AREA(geom) > 2000000;
```

4. Repeat the above query for the Counties having area < 2000000 sq m.

5. Find the total length of roads having a name – NB: adapt this query to use ST_TRANSFORM

```
SELECT SUM(ST_LENGTH(geom))
FROM public.london_highway
where name IS NOT NULL;
```

6. Check the SRID of the London_counties geometries. Is this British National Grid?

```
SELECT name, ST_SRID(geom)
FROM london_counties;
```

7. Which counties contain a street called 'High Street'? (NB: The 'distinct' is needed as some counties are multi-part geometries)

```
SELECT distinct A.name
FROM public.london_counties A, public.london_highway B
WHERE B.name = 'High Street'
and ST_INTERSECTS(A.geom, ST_TRANSFORM(B.geom,27700)) = 't'
```

Note that we are using two spatial functions here. Firstly, we use an ST_TRANSFORM function – this is required because the london_counties data and the uk_highway datasets are not projected into the same system. We are transforming the uk_highway dataset to the EPSG of 27700, which is that used by the london_counties data.

Secondly, the ST_INTERSECTS function finds any Counties intersecting the roads. The letters *cnty* after the London_counties table name in the first line are called an 'alias' and give us a short way to refer to a table. We need to do this as both tables have columns called *geom* so we need to be clear as to which one we mean.

8. A similar query can be used to find the points of interest in Bromley

```
SELECT * FROM public.london_poi a
WHERE ST_within(st_transform(a.geom,27700),
(SELECT c.geom  FROM public.london_counties c where c.name like
'%Bromley%'));
```

Exercise 3 – Write Your Own Combined Queries

1. Are there any the london_counties having area > 50000000 sq m?

2. List the length of road passing through each county, along with the name of the county (don't forget to use ST_TRANSFORM on your Highways data) and use a JOIN with a spatial clause. Use an ST_CONTAINS to make sure you only get highways that are entirely inside a county. Running this query may take a long time (over a minute)!

3. Find the total area of highway where oneway is true and the highway is buffered by 10m using ST_BUFFER. Don't forget to use ST_TRANSFORM as the highway is projected using WGS84 – i.e. degrees/minutes/seconds – so you need to put it into British National Grid first.

4. Using WITH and an ST_BUFFER, find the total length of highway that has a point of interest within 10m (this query may take a while to execute)

    - Use ST_TRANSFORM to re-project the highway data to British National Grid – save this as the first component of the WITH statement
    - Buffer the highways by 10m and save the SQL as the second part of the WITH statement
    - Use an ST_CONTAINS to find out where the buffered highway contains points of interest, and save this as the third element of the WITH statement
    - Finally use an ST_AREA to work out the area of the resulting buffered highway

5. Which MPs have to travel furthest? Parliament is in the county (constituency) where name = 'Cities of London and Westminster Boro Const'. Using WITH and ST_DISTANCE and ST_CENTROID find the top 5 constituencies that are furthest from parliament as the crow flies (i.e. straight line distance)
    - First find the centroid of all constituencies and make this the first WITH statement (st_centroid)
    - Find the centroid of the specific constituency - *Cities of London and Westminster Boro Const* and make that the second query in the WITH statement
    - Then find the distance between these centroids and that for Cities of London and Westminster Boro Const
    - Order these in descending order
    - Then just show the top 5 using LIMIT 5

6. Write a query to find the categories and counts of the points of interest within 200m of the shortest road in the London counties
    - Find the length of all the roads and set this as a WITH statement – remember to use ST_TRANSFORM
    - Order these road lengths and select the top one using LIMIT 1, set as a second WITH statement
    - Create a 200m buffer around this road and set as a third WITH statement – again (ST_TRANSFORM might be needed if you didn't keep a transformed geometry in an earlier query)

- Use the buffer to find any points that are contained within it and list these, add to the WITH statement (don't forget to transform the POIs)
- Finally use a GROUP BY statement to work out the categories and counts

Part 2- Working with 3D Data - DDL and DML

In this first part of the practical, you will create some test 3D data to be used to explore the 3D functionality offered by PostGIS 2.0. You will also use the 'Feature Manipulation Engine' software (FME, from SafeSoft) to visualise the data.

Exercise 1 - Creating a simple 3D Test Dataset

The aim of the first part of today's practical session is to explore the 3D functionality offered by PostGIS. One very good way to do this is to create a test dataset with known properties – i.e. a dataset that allows you to test queries including distance, area and volume measurement, along with running adjacency tests and containment tests.

You should insert your data into a table similar to the following (remember to change the schema name):

```
create table practical4.threedbuildings
(id serial not null,
location geometry,
geometryname  character varying(150));

UPDATE practical4.threedbuildings SET location=ST_SetSRID(location,27700);
```

Make sure to add a primary key to the table, as otherwise you will not be able to see the table data in FME.

```
alter table practical4.threedbuildings add CONSTRAINT threedbuildings_pk PRIMARY KEY
(id);
```

1. Insert a 3D point into the table with coordinates 10 12 13
2. Insert a 3D line with coordinates 11 11 11 ending in 23 21 22
3. Insert a 3D polygon with coordinates 11 11 11 11 22 11 22 22 11 22 11 22 11 11 11 (remember that a polygon needs an extra set of brackets)
4. Using ST_Extrude, insert a polyhedral surface with base coordinates 14 14 14 14 22 14 22 22 14 22 14 14 14 14 14 and height of 6m

Exercise 2- Visualising your Test Data in FME

Once you have created the dataset, it is a good idea to visualise it to ensure that it has in fact been created correctly. As there are no real 3D GIS that connect directly to PostGIS, you can make use of FME. FME is originally designed as a translation tool (between different GIS formats) and a transformation tool (for example to take one buildings dataset with a set of columns and merge it with another one having different columns). However, they also provide a 3D visualisation tool called the FME Data Inspector.

NB: Instructions here are for FME 2015, installed on the UCL machines. Later versions of FME may differ.

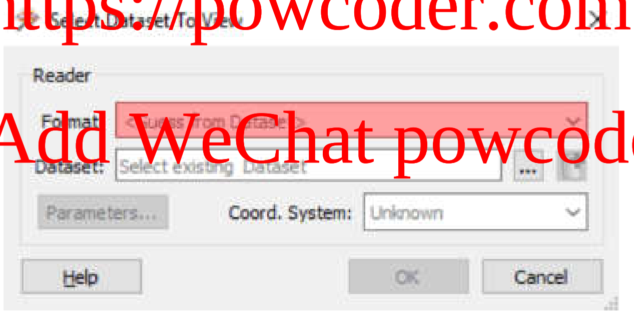
1. Make sure you are on Desktop Anywhere or the UCL VPN service if you are working on your own machine.

2. Start the FME Data Inspector tool. If you are asked for licensing information, use the following details for the license server:

   SAFELM.CEGE.UCL.AC.UK:27651

3. Connect to the PostGIS database where you have your data, by clicking on FILE > OPEN DATASET.

4. Change the data format by clicking on the down arrow and select MORE FORMATS


Select Dataset To View

Reader

Format: <Guess from Dataset>

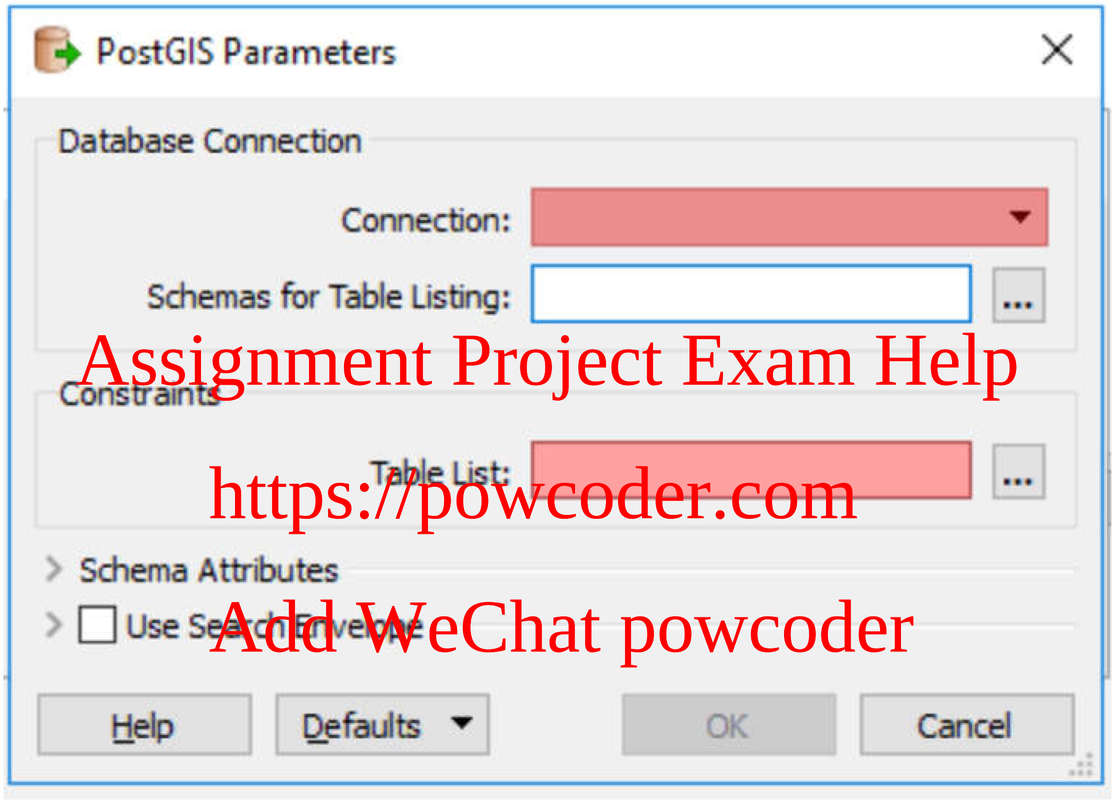
Dataset: Select existing Dataset

Coord. System: Unknown

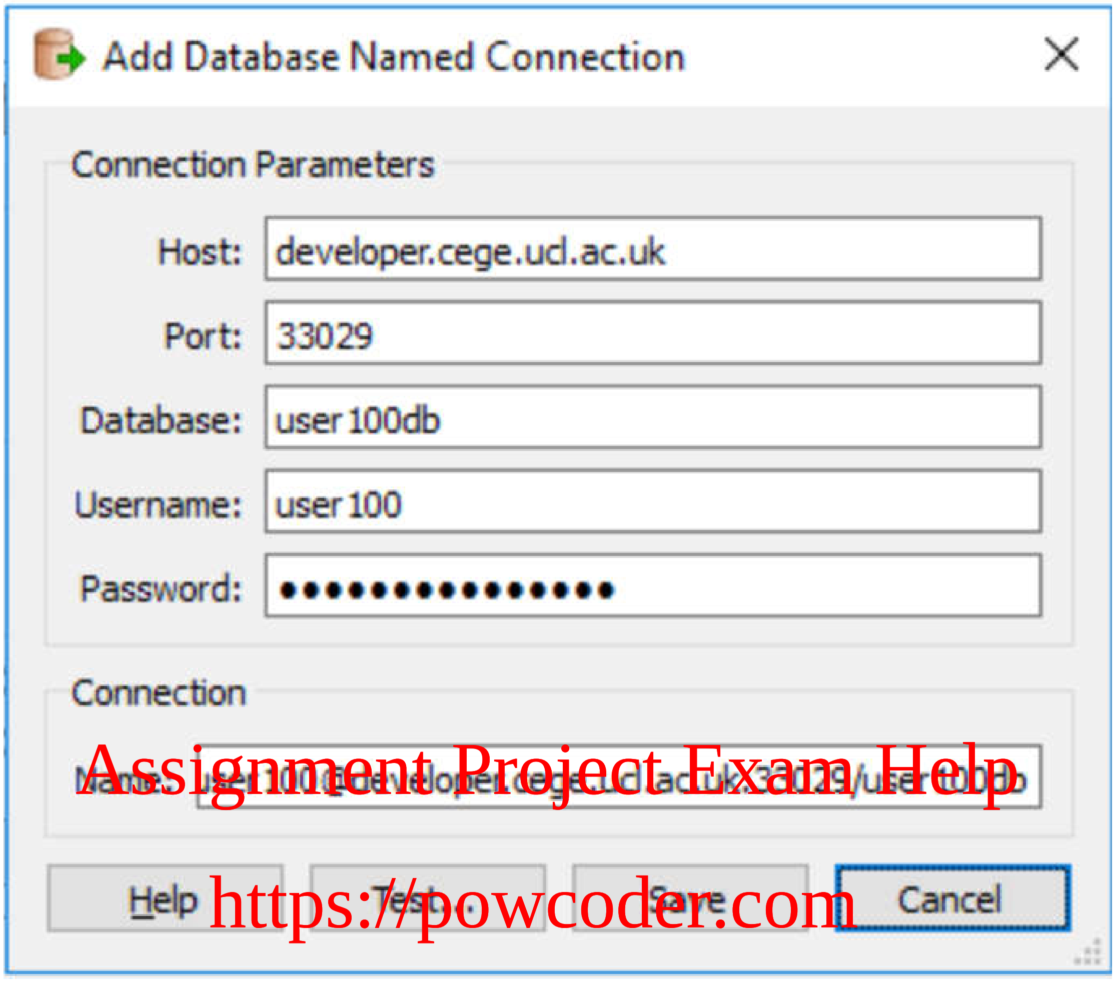
5. Select PostGIS from the list and click OK (NB not PostGIS Raster!). Now you need to fill in the connection details for the database.

6. Click on the PARAMETERS button. The form should look similar to those shown below (note that you may have to change the host, port, database, username and password).

NB: The appearance of the Connection Parameters for PostGIS will vary depending on the version:



7. Click on CONNECTION > ADD NAMED DATABASE CONNECTION and fill in your usual database connection details in the form



Add Database Named Connection

Connection Parameters

Host:	developer.cege.ucl.ac.uk
Port:	33029
Database:	user100db
Username:	user100
Password:	•••••••••••••••

Connection

Name: user100@developer.cege.ucl.ac.uk:33029/user100db

8. Click SAVE to save the parameters.

9. In the SCHEMA box type practical4, and in the table option click on the button and select the threedbuildings table

10. Click OK to set the parameters and OK again to start the visualisation process. Your data will initially appear in 2D but you can view it in 3D by clicking the 'Lock the Current View in 3D' option button and then the 'Use the Orbit Mode' button to explore the data.

11. Check that all your data is as expected. You may wish to explore the different 2D and 3D visualisation options that the FME Data Explorer tool offers.

Exercise 3- The Flying Freehold

1. Create a new table called practical4.freehold with ID, name and geometry columns (call the geometry column location) and an SRID of 27700 (British National Grid). Remember to add the geometry column as a separate column

2. Take the flying freehold example that we sketched in class (from the 'draw some 3D objects' exercise) and create the two base buildings by using ST_Extrude to a height of 10m (call them building1 and building2)

   Coordinate lists:
   polygon1: 529802 182284 0,529812 182292 0,529810
   182294 0,529815 182297 0, 529812 182300 0, 529798 182290 0,529802 182284 0

   polygon2: 529802 182284 0,529807 182279 0, 529820 182289 0, 529816 182294 0, 529812 182292 0, 529802 182284 0

3. Now create a third row in the table and insert the freehold geometry separately – call it 'freehold'. You can use ST_Extrude again, but remember the coordinates will not have z value of 0

   freehold: 529803 182289 3,529808 182283 3,529812 182286 3,529807 182292 3,529803 182289 3

4. Once you have created the three buildings, visualise the data in FME.

5. Optional extra (for geospatial students): you can also add additional layers to the FME visualisation – do the buildings appear as expected when you overlay the MasterMap data for the UCL area (which you can download from EDINA's Digitmap Service)?