Lab 4
Irish Map Data

Learning Objectives:

- ✓ Get to know Irish map data
- ✓ Perfect spatial queries

Hints and Tricks:

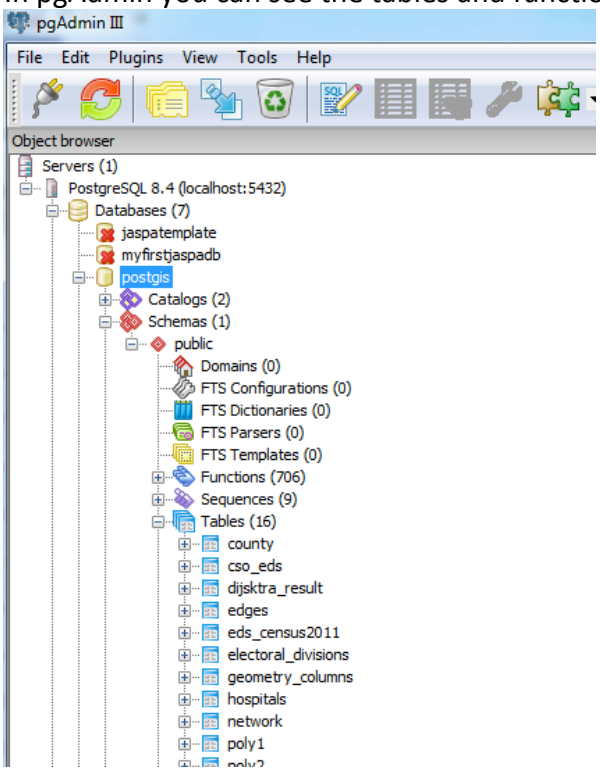
- Data is on Webcourses
- If unsure about something you see, consult the OGC's Simple Features for SQL and PostGIS manual (both in Webcourses)

Getting Started:

- Load the following SQL tables:
  - County.sql
  - Buildings_geodir.sql

Revision:

Use the the SQL-Shell , pgAdmin, or QGis as appropriate.
In pgAdmin you can see the tables and functions available as below:

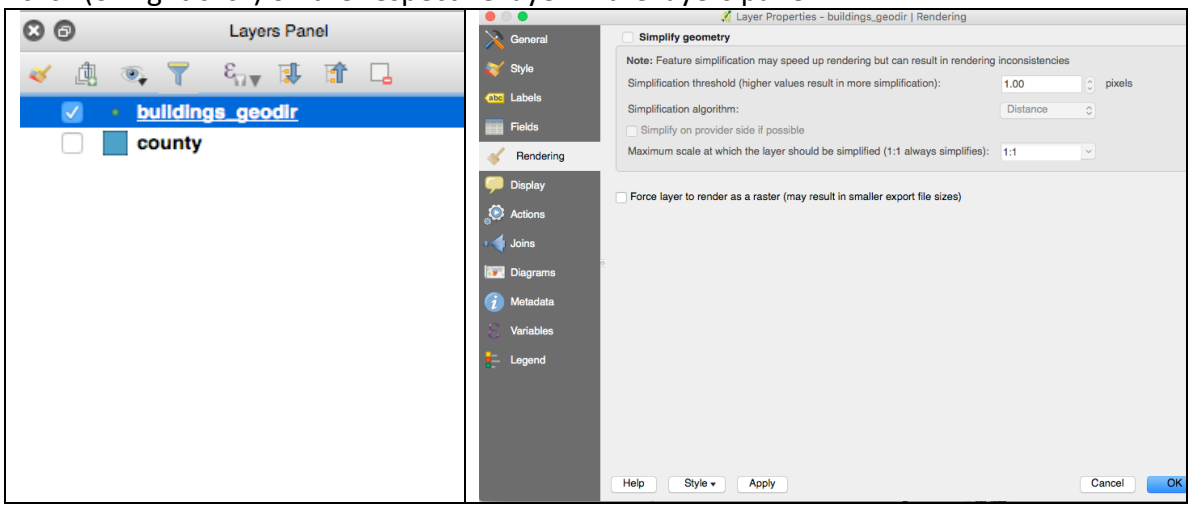


In the SQL-Shell you can check what tables/views/users exist with `\d, \dn, \dv, \du`, commands.

In **QGIS** you can check the attributes of a spatial table or query as follows. Say we want to see the attributes of our `Geodirectory` sample call `buildings_geodir`. We open

Lab 4
Irish Map Data

a connection to Postgres and get the table into QGis. For changing attributes we double click (or right click) on the respective layer in the layers panel.



1. Construct a table called Ireland from the county table.

--create table Ireland for geometries of Ireland

`ireland(gid, code, Ireland)`

where gid is the primary key, and Ireland is a geometry column, constructing a multipolygon for the srid 29900.

You can do this either the 'old way' of using `AddGemoetryColumn`, or the new way of just adding a geometry column at construction.

```
CREATE TABLE ireland (gid serial PRIMARY KEY, "code" int4);
```

--add geometry column

```
SELECT AddGeometryColumn('ireland','the_geom','29900', 'MULTIPOLYGON',2);
```

-- Most spatial operations and predicates require the prefix ST_.

-- In Postgres8.4/PostGIS1.5 the AddGeometryColumn command does not use the prefix ST_.

--generate merged polygons from original layers based on common values from the "code" field

--note: use the ST_multi function because the MemUnion could downcast MULTIPOLYGON to POLYGON

```
INSERT INTO ireland (the_geom)
SELECT ST_multi(ST_MemUnion(c.the_geom)) AS the_geom FROM county
c;
SELECT count(*)  FROM ireland;
SELECT ST_geometryType(the_geom)  FROM ireland ;
SELECT ST_NumGeometries(the_geom)  FROM ireland ;
```

View the result in **QGis**.

Does the use of `'ST_multi'` make any difference? Check this in the SQL-Shell. The statements use two different equality predicates.

```
SELECT
ST_equals(ST_MemUnion(the_geom),ST_multi(ST_MemUnion(the_geom)))
FROM county;

SELECT
ST_OrderingEquals(ST_MemUnion(the_geom),ST_multi(ST_MemUnion(the_
geom)))
FROM county;
```

2. In SQL-Shell**. Find all the counties contained in your `ireland` table, using `ST_contains`.**

```
select c.name  from county c, ireland i where
ST_contains(i.the_geom,c.the_geom);
```

Note this query uses the spatial predicate `ST_contains`, the result is just a list of county names. How could we get the same result without a spatial predicate?

For some reason the above query does not give the correct number of counties, several counties are missing.

Lab 4
Irish Map Data

Created Table	Original Table
Carlow	Carlow
	Cavan
Clare	Clare
Cork	Cork
Cork County Borough	Cork County Boro
	Donegal
Dublin Belgard	Dublin Belgard
Dublin County Borough	Dublin County Bo
Dublin Fingal	Dublin Fingal
Dun Laoghaire - Rathdown	Dun Laoghaire - R
Galway	Galway
Galway County Borough	Galway County Bo
Kerry	Kerry
Kildare	Kildare
Kilkenny	Kilkenny
Laois	Laois
	Leitrim
Limerick	Limerick
Limerick County Borough	Limerick County B
Longford	Longford
	Louth
Mayo	Mayo
Meath	Meath
	Monaghan
	Northern Ireland
Offaly	Offaly
Roscommon	Roscommon
Sligo	Sligo
Tipperary North Riding	Tipperary North R
Tipperary South Riding	Tipperary South R
Waterford	Waterford
Waterford County Borough	Waterford County
Westmeath	Westmeath
Wexford	Wexford
	Wicklow

One way to get the full list of 35 counties we need to expand the geometry in the ireland table by say 0.5 metres.

```
SELECT c.name  from county c, ireland i
WHERE ST_contains(ST_Expand(i.the_geom,0.5),c.the_geom)
ORDER by name;
```

```
SELECT name from county ORDER by name;
```

Both give 35 counties.

Can you find a reason that some counties are not contained in the union? Hint use `ST_Difference`.

3. In SQL shell. Find, names, area and perimeter of Dublin.

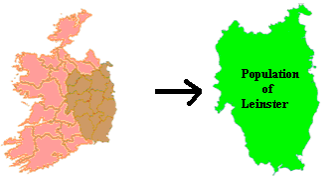
Note in this database Dublin is made up of
Fingal
Dublin City
South Dublin
Dun Laoghaire – Rathdown

Note the perimeter is held in the km2 column. Try this query in the SQL-Shell.

```
select name from county where name like '%Dublin%';
-- get all names
select name from county AS c where
 (c.name like '%Dublin%') or
 (name like '%Dun Laoghaire%');

-- get perimeter in kilometres
select ST_perimeter(the_geom)/1000 from county where name =
'Laois';
select km2  from county where name = 'Laois';

-- get area in square kilometres
select ST_area(the_geom) /1000000 from county where name =
'Laois';
select area_km2  from county where name = 'Laois';

-- get perimeter in kilometres of all of 'Dublin'
-- Note we cannot sum the perimeters of each Dublin area
select ST_perimeter(ST_MemUnion(c.the_geom))/1000 as “Kms” from
county AS c where  (c.name like '%Dublin%') or
 (c.name like '%Dun Laoghaire%');

-- get area in square kilometres of all of 'Dublin'
select sum(ST_area(the_geom))/1000000 as “sq Kms” from county AS
c where  (c.name like '%Dublin%') or
 (name like '%Dun Laoghaire%');
```

Lab 4
Irish Map Data

4. Merging counties.
It is required to produce a single map of Leinster with no counties borders.



The map should be labelled with the total area of Leinster in square kilometres. The area should be the sum of the area of each county (including the four county council areas of Dublin). Issue the following in **QGis.**

```
SELECT ST_Union(the_geom),sum(st_area(the_geom)/ 1000000) as area
FROM county
WHERE name='Wexford' OR  name='Wicklow' OR  name='Westmeath' OR
name='Meath' OR  name='Louth' OR  name='Carlow' OR
name='Kildare' OR
name='Kilkenny' OR  name='Laois' OR name='Longford' OR
name='Offaly' OR name='Fingal' OR name = 'South Dublin' OR name =
'Dun Laoghaire - Rathdown' OR name = 'Dublin City';
```

Check the area in QGis visually.What does it return?

Now calculate the area from the union

```
SELECT ST_Union(the_geom), ST_area(ST_Union(the_geom))/1000000 as
area
FROM county
WHERE name='Wexford' OR  name='Wicklow' OR  name='Westmeath' OR
name='Meath' OR  name='Louth' OR  name='Carlow' OR
name='Kildare' OR
name='Kilkenny' OR  name='Laois' OR name='Longford' OR
name='Offaly' OR name='Fingal' OR name = 'Dublin Belgard' OR name
= 'Dun Laoghaire - Rathdown' OR name = 'Dublin City';
```

Gives 19256.29 km2

Now calculate the area from the stored areas in the county table.

```
SELECT ST_Union(the_geom), sum(area_km2) as area
FROM county
WHERE name='Wexford' OR  name='Wicklow' OR  name='Westmeath' OR
name='Meath' OR  name='Louth' OR  name='Carlow' OR
name='Kildare' OR
name='Kilkenny' OR  name='Laois' OR name='Longford' OR
name='Offaly' OR name='Fingal' OR name = 'South Dublin' OR name =
'Dun Laoghaire - Rathdown' OR name = 'Dublin City';
```

This time we get 18960.35 km2 (a bit smaller than above)
Why might this be?

5. Load in Bus Stop data for Dublin.
The data in the file STOPS.SQL represents Dublin bus stops.
The original data is in the file WGS84 (EPSG 4326).
We will transform the geometry from WGS84 (EPSG 4326) to ING (29900)
The instructions are in the file STOPS.SQL.

6. Explore the tables used on this course.
Look at the attributes and geometry of following tables and geometry by viewing them in **QGis**. The tables are stored in SQL files on your Webcourses folder.

- dublin_historical,
- bedrock,
- roads,
- dublin_highway1,
- buildings_geodir,
- dublin_eds,
- stops,
- DCCLitterBinSurvey2008.

Use \d to find the columns in the tables.

```
select rmp_prop, map_symbol, entity_id, co_id, smr_val0,
nat_grid_e, class_desc, nat_grid_n,objectid, townlands,
scope_n1,smrs, the_geom from  dublin_historical;
```

You can also get the geometry type by using SQL.

```
SELECT distinct(ST_geometryType(the_geom))
FROM  dublin_historical;

SELECT distinct(ST_geometryType(the_geom))
FROM  dublin_eds;
```

Compare the names in the litter bins, bus stops and road tables (OSM).

```
SELECT distinct(h.name, s.name)
FROM dublin_highway1 h, bus_stops s
WHERE s.name ilike h.name;

SELECT name,street
FROM dub_bins2008,dublin_highway1 WHERE name ilike street;
```

Lab 4
Irish Map Data

7. What do these SQL statements do?

Write out in English what these queries are doing. Check the OGC's Simple Features for SQL and the PostGIS Manual.

```
select name, ST_asText(Centroid(the_geom)) from county;

select  find_srid('', 'county', 'the_geom');

select name, ST_asText(envelope(the_geom)) from county;

select name, ST_asText(ConvexHull (the_geom)) from county;

select name, ST_extent(the_geom)) from county group by name;

select name from county where ST_GeomFromText('POINT(309612.0
233192.0)', 29900) && the_geom;

SELECT sum(male1_1+female1_1), ST_multi(ST_MemUnion(ed.the_geom))
FROM dublin_eds ed
WHERE saps_label like '%Royal%';
```

-- Write an insert statement to insert DIT College Kevin Street at (315470 233300) into the `buildings_geodir` table. You will have to insert a `WKT  POINT` and convert it to geometry. Assign the name field 'DUBLIN INSTITUTE OF TECHNOLOGY KEVIN STREET' Then run the following query:

```
SELECT  b1.name
from buildings_geodir b1, buildings_geodir b2
WHERE b2.name = 'DUBLIN INSTITUTE OF TECHNOLOGY KEVIN STREET' and
ST_DWithin(b2.the_geom,b1.the_geom, 500)
ORDER BY st_distance(b2.the_geom,b1.the_geom) desc limit 1;


SELECT b.the_geom
FROM Roads r, buildings_geodir b
WHERE ST_contains(buffer(r.the_geom,500),b.the_geom) and
r.name ='Kevin Street Lower';


select townlands from dublin_historical WHERE distance(the_geom,
ST_GeomFromText('POINT(309612.0 233192.0)', 29900)) < 100;
```

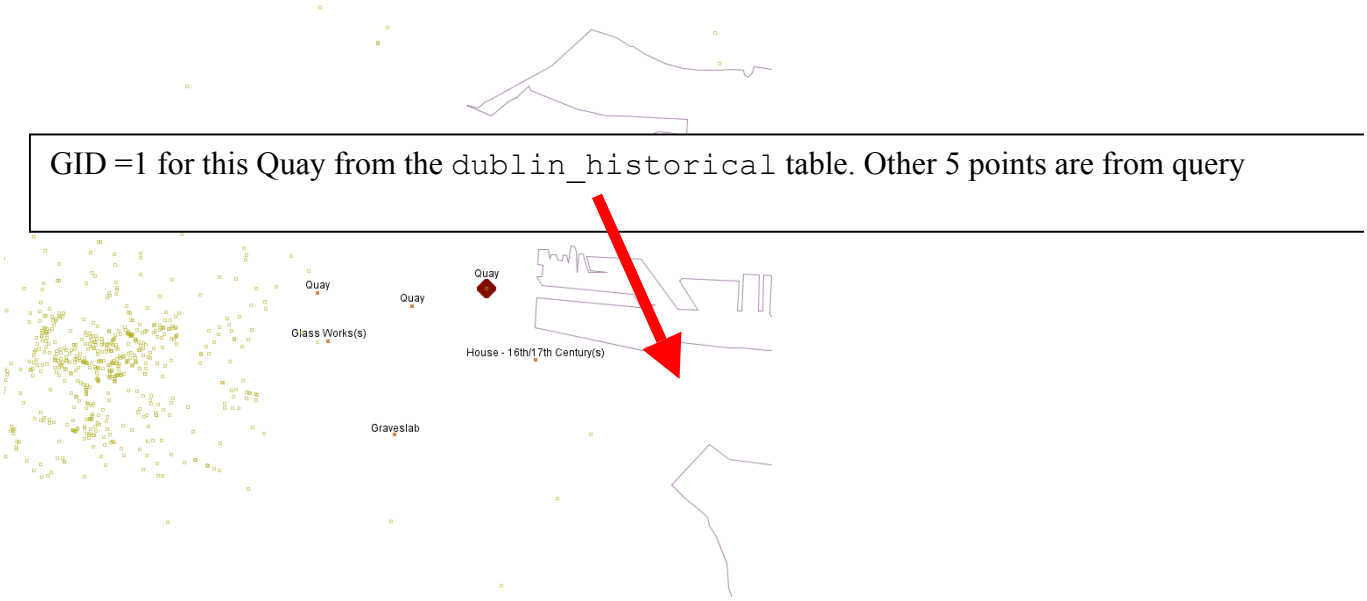
Lab 4

Irish Map Data

```
select c.name, h.townlands from county AS c, dublin_historical AS
h WHERE ST_distance(h.the_geom, ST_GeomFromText('POINT(317431.0
231704.0)', 29900)) < 1000 and c. Name = 'Dublin County Borough';
```

```
SELECT g1.gid as object, g1.class_desc,g2.gid as neighbour,
g2.class_desc,cast(ST_Distance(g1.the_geom,g2.the_geom) as Int)
FROM dublin_historical As g1, dublin_historical As g2
WHERE g1.gid = 1 and g1.gid <> g2.gid
ORDER BY ST_Distance(g1.the_geom,g2.the_geom)
LIMIT 5;
```

Based on the query above, how would you create the following map in QGis?



8. DISTINCT and DISTINCT ON

DISTINCT eliminates duplicate rows from the result. For Example

```
select distinct(h.name, s.name) from dublin_highway1 h, stops s
where
s.name like h.name;
```

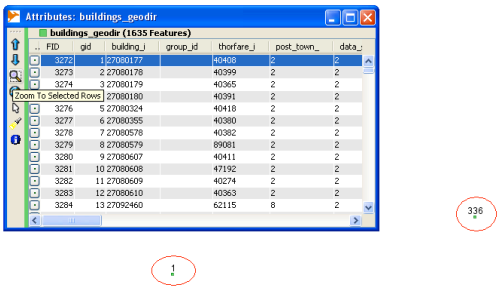
DISTINCT ON eliminates rows that match on all the specified expressions.
DISTINCT ON (expression [, ...]) keeps only the first row of each set of rows where the given expressions evaluate to equal. The DISTINCT ON expressions are interpreted using the same rules as for ORDER BY. Note that the "first row" of each set is unpredictable unless ORDER BY is used to ensure that the desired row appears first. For example:

Lab 4

Irish Map Data

```
SELECT DISTINCT ON(g1.gid)  g1.gid As gref_gid, g2.gid As gnn_gid
, cast( ST_Distance(g1.the_geom,g2.the_geom) as INT)
    FROM buildings_geodir As g1, buildings_geodir As g2
    WHERE g1.gid <> g2.gid AND ST_DWithin(g1.the_geom,
g2.the_geom, 100)
    ORDER BY gref_gid, cast(ST_Distance(g1.the_geom,g2.the_geom)
as INT) ;
```

Examine output in SQL Shell. Note distance between point with `gid=1` and point with `gid=336`. Now, bring entire `buildings_geodir` into **QGis**.
Using the distance tool in **QGis** measure the distance between gid1- and gid336. How does it compare with the output in the SQL-Shell? Try the same experiment for some more points.



Now try the query without distance in the order clause:

```
SELECT DISTINCT ON(g1.gid)  g1.gid As gref_gid, g2.gid As gnn_gid
, cast( ST_Distance(g1.the_geom,g2.the_geom) as INT)
        FROM Buildings_geodir As g1, Buildings_geodir As g2
        WHERE g1.gid <> g2.gid AND ST_DWithin(g1.the_geom,
g2.the_geom, 100)
    ORDER BY gref_gid ;
```

Using ORDER BY and GROUP BY

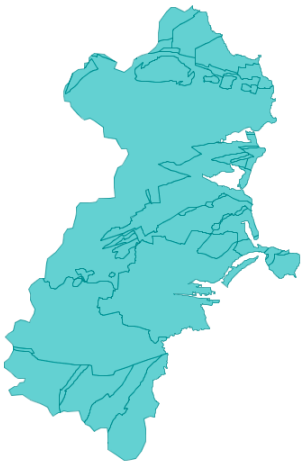
```
SELECT name, ST_area(the_geom)/10000 AS hectares FROM county
ORDER BY hectares DESC LIMIT 1;

SELECT c.name, sum(length(r.the_geom))/1000 as roads_km FROM
dublin_highway1 AS r, county AS c WHERE r.the_geom && r.the_geom
 AND ST_contains(c.the_geom,r.the_geom)
AND c.name = 'Dublin City'
GROUP BY c.name
ORDER BY roads_km;
```

```
SELECT townlands FROM dublin_historical h, dublin_highway1 r,
county c WHERE
 ST_distance(h.the_geom,r.the_geom) < 1000
AND c.name = 'Dublin City';
```

-- to see the result from this query you will need to create a table to store the result

```
select ST_intersection(b.the_geom, c.the_geom) as the_geom,
b.agebracket as  agebracket
from bedrock b, county c
where b.the_geom && c.the_geom and ST_intersects(b.the_geom,
c.the_geom) and c.name like '%Dublin%';
```

You should get a map like this. Notice that the bedrock information has been cut by some Dublin regions.



```
create table dublin_bedrock1 as
select ST_intersection(b.the_geom, c.the_geom) as the_geom,
b.agebracket as  agebracket
from bedrock b, county c
where b.the_geom && c.the_geom and ST_intersects(b.the_geom,
c.the_geom) and c.name like '%Dublin%';
```

The above query uses a shorthand table creation method. It should be executed in SQL-Shell. This table can be viewed in QGis as follows:

```
select the_geom from dublin_bedrock1;
```

Lab 4

Irish Map Data

9. What do these queries do?

Explain the overall purpose of the following queries. Describe any operations based on the OGC Simple Features for SQL Standard that are used in the queries.

```
select c.name,b.unitname, ST_intersection(c.the_geom,b.the_geom)
from bedrock as b, county as c where unitname ilike '%Deep
marine%';
```

What does the result tell you about the geological entities?

```
SELECT sum(male1_1+female1_1), ST_multi(ST_MemUnion(ed.the_geom))
FROM dublin_eds ed
WHERE saps_label like '%Royal%';

SELECT  b1.num_residents
from buildings_geodir b1, buildings_geodir b2
WHERE b2.name = 'DUBLIN INSTITUTE OF TECHNOLOGY' and
ST_DWithin(b2.the_geom,b1.the_geom, 500)
ORDER BY num_residents desc limit 1;

SELECT b.the_geom
FROM Roads r, buildings_geodir b
WHERE ST_contains(buffer(r.the_geom,500),b.the_geom) and
r.name ='Kevin Street Lower';
```

Explain the overall purpose of the following queries. Describe any operations from the OGC Simple Features for SQL Standard that are used in the queries. Please refer to the schemas using \d were necessary.

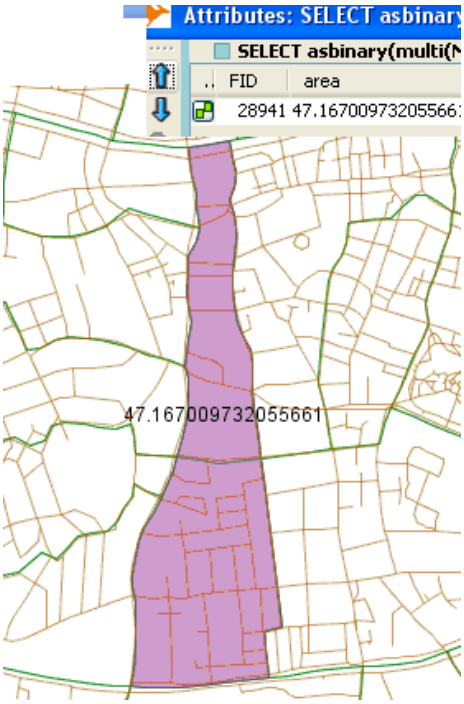
```
SELECT b1.the_geom
FROM buildings_geodir b1, buildings_geodir b2
WHERE
b2.name = 'DIT Kevin Street' and
ST_DWithin(b2.the_geom,b1.the_geom, 200)
ORDER BY ST_distance(b2.the_geom,b1.the_geom) ASC LIMIT 1;


SELECT  st_distance(
 transform(GeometryFromText('POINT(-6 53)', 4326), 29900),
 transform(GeometryFromText('POINT(-8 52)', 4326), 29900))/1000;

select saps_label from dublin_eds WHERE saps_label like '%Wood
Quay A%
' OR  saps_label like '%Wood Quay B%';
```

Lab 4

Irish Map Data

```
SELECT ST_multi(ST_MemUnion(the_geom)),
 sum(area(the_geom)/ 10000) as area
FROM dublin_eds
WHERE saps_label like '%Wood Quay A%
' OR  saps_label like '%Wood Quay B%';
```

-- As can be seen below these two EDs are adjacent.



#

10 Left Right Above Below

(A |&> B) Returns TRUE if A's bounding box overlaps or is above B's.
(A &<| B) Returns TRUE if A's bounding box overlaps or is below B's.
(A &< B) Returns TRUE if A's bounding box overlaps or is to the left of B's.
(A &> B) Returns TRUE if A' bounding box overlaps or is to the right of B's.

```
select a.saps_label, b.the_geom
from dublin_eds as a, dublin_eds as b
where a.the_geom &<  b.the_geom and
(b.the_geom |&>  a.the_geom) and
ST_touches(a.the_geom, b.the_geom) and
a.saps_label ilike '%Phoenix Park%';
```

Lab 4

Irish Map Data

11. Nearest neighbours

The following query finds all nearest neighbours within 300 metres of building with gid=1, with nearest being listed first, using ORDER BY. The inclusion of the ST_Within stops searching after 300 metres, otherwise the entire table/space would be searched. gid=1 is excluded from the search. OGC data type POINT.
The OGC functions are: ST_distance and ST_within.

```
SELECT g1.gid As gref_gid, g2.gid As gnn_gid,
ST_Distance(g1.the_geom,g2.the_geom)
FROM buildings_geodir As g1, buildings_geodir As g2
WHERE g1.gid = 1 AND  g1.gid <> g2.gid  AND
ST_DWithin(g1.the_geom, g2.the_geom, 300)
ORDER BY ST_Distance(g1.the_geom,g2.the_geom);

SELECT g1.gid As gref_gid, g2.gid As gnn_gid,
ST_Distance(g1.the_geom,g2.the_geom) as dist
FROM buildings_geodir As g1, buildings_geodir As g2
WHERE g1.gid = 1 AND  g2.gid != 1 AND
ST_DWithin(g1.the_geom, g2.the_geom, 300)
ORDER BY dist;
```

13. De-aggregate Functions

PostGIS also supports the **de**-aggregation of data from one record into many. See
http://postgis.net/docs/ST_Dump.html
http://postgis.net/docs/geometry_dump.html

```
SELECT ST_AsText((ST_Dump(the_geom)).geom)
  FROM county
  WHERE name='Meath';

SELECT ST_AsText((ST_Dump(the_geom)).geom)
  FROM roads;
  WHERE name='Meath';

select the_geom,name from roads where name = 'Bray By-pass';

select ST_AsText(the_geom) from roads where name = 'Bray By-
pass';
select ST_AsText((ST_Dump(the_geom)).geom) from roads where name
= 'Bray By-pass';
```