In May 2017, I wrote a White Paper which started to explore using the **Ogr2Ogr** Command Line tool (within **GDAL)** to undertake attribute, geographic queries and some geoprocessing routines using flat GIS file formats such as ESRI Shapefiles. That paper can be accessed via this link:

https://www.cadlinecommunity.co.uk/hc/en-us/articles/115003496349-Utilising-Ogr2Ogr-Part-One

In this second paper, I will start to explore more advanced spatial processing options and also how to use Ogr2Ogr tools in conjunction with your **PostGIS database**.

The data used within this White paper is freely available from the Ordnance Survey and is used within all our DynamicMaps Open Source GIS training courses: http://www.dynamicmaps.co.uk/training/

- Introduction to QGIS - http://www.dynamicmaps.co.uk/introduction-to-qgis-desktop-gis/
- Introduction to GeoServer - http://www.dynamicmaps.co.uk/introduction-to-geoserver/
- Integrated Open Source GIS - http://www.dynamicmaps.co.uk/open-source-gis-integration/

This White Paper has been compiled using several online examples from the following resources:

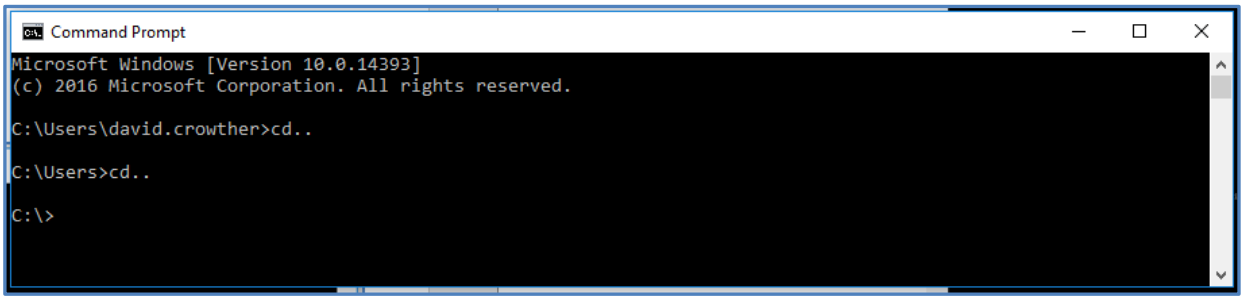
- GIS Stack Exchange
- http://www.postgresonline.com/journal/archives/31-GDAL-OGR2OGR-for-Data-Loading.html
- https://github.com/dwtkns/gdal-cheat-sheet

To access the Ogr2Ogr command tools, you need to navigate to the location where GDAL has been installed. In my case, I have used the installer provided by the OSGeo project which contains the GDAL tools:

https://live.osgeo.org/en/win_installers.html

and my installation is located at: **C:\OSGeo4W64\bin**

Using the command cd.. (change directory) navigate to the root of your local drive, for example C:\



1 – Create an Index on an ESRI Shapefile

Using the Ogrinfo options we can create an index on any field within a GIS file, such as a Shapefile. Indexes will increase the performance of your files, specifically when undertaking queries and updating columns.

Command:

ogrinfo example.shp -sql "CREATE INDEX ON example USING fieldname"

Example:

ogrinfo D:\Temp\Liverpool_Wards.shp -sql "CREATE INDEX ON Liverpool_Wards USING la_code"

```
C:\OSGeo4W64\bin>ogrinfo D:\Temp\Liverpool_Wards.shp -sql "CREATE INDEX ON Liverpool_Wards USING la_code"
INFO: Open of `D:\Temp\Liverpool_Wards.shp'
      using driver `ESRI Shapefile' successful.

C:\OSGeo4W64\bin>
```

The result will state that the index was successfully created.

2 – DISSOLVE features in an ESRI Shapefile

Using the Ogr2Ogr tools we can undertake several geoprocessing tasks on GIS files. In this example we will dissolve the boundaries between polygons with a unique value in common.

Command:

gr2ogr -f "ESRI Shapefile" InputShapefileLocation\Name.shp OutputShapefileLocation\Name.shp -dialect sqlite -sql "select ST_union(Geometry),fieldname from inputshapefile GROUP BY fieldname"

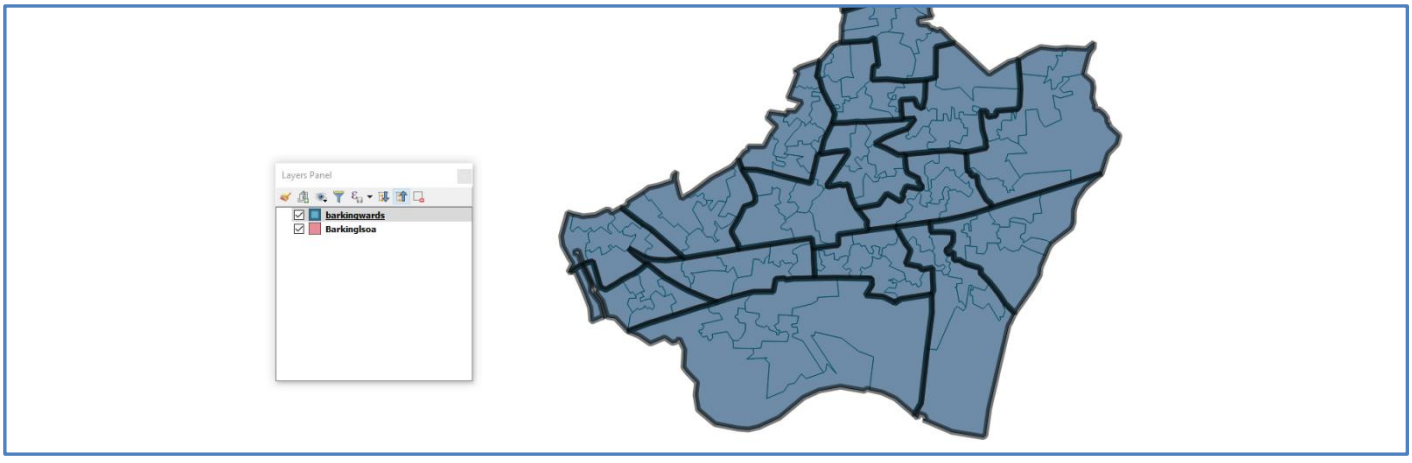
Example:

ogr2ogr -f "ESRI Shapefile" D:\Temp\barkingwards.shp D:\Temp\Barkinglsoa.shp -dialect sqlite -sql "select ST_union(Geometry),STWARD_NAM from barkinglsoa GROUP BY STWARD_NAM"

```
C:\OSGeo4W64\bin>ogr2ogr -f "ESRI Shapefile" D:\Temp\barkingwards.shp D:\Temp\Barkinglsoa.shp -dialect sqlite -sql
"select ST_union(Geometry),STWARD_NAM from barkinglsoa GROUP BY STWARD_NAM"

C:\OSGeo4W64\bin>
```

The result will be a new shapefile called barkingwards, and the LSOA polygons will have been dissolved using the Ward Name field (STWARD_NAM) to generate ward polygons.



Using the Ogr2Ogr tools we will now explore working with **PostGIS**, looking at options to; list tables, extract data, append datasets, import new tables and edit the schema of existing database tables.

To access your PostGIS database via Ogr2Ogr the most important thing is to correctly specify the database connection parameters to your PostGIS Instance. Below is an example of the ogr2ogr syntax for connecting to PostGIS.

PG:"host=*servername* port=5432 user=*'username'* password=*'password'* dbname=*'PGDatabaseName'*"

3 – Connect to PostGIS and List Tables

Ogr2Ogr provides the capability to very quickly list the spatial tables within your PostGIS database.

Command:

ogrinfo PG:"host=172.17.2.176 port=5432 user='user' password='password' dbname='dbname'"

Example:

ogrinfo PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'"

The result will be a list of the tables within the specified PostGIS database.

```
C:\OSGeo4W64\bin>ogrinfo PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'"
INFO: Open of `PG:host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps''
      using driver `PostgreSQL' successful.
1: lsoa_extra (Multi Polygon)
2: b_roads (Multi Line String)
3: mways (Multi Line String)
4: Leicester_ITN (Multi Line String)
5: 1_lsoa_select (Polygon)
6: land_parcels (Multi Line String)
7: postgisshrewapps (Polygon)
8: shrew4326 (Polygon)
9: shrewbng (Polygon)
10: papps4326 (Polygon)
11: apps4326 (Polygon)
12: planningapps27700 (Polygon)
13: hospitals (Point)
14: topographicarea (Polygon)
```

4 – Extract data from PostGIS to a GeoJSON file

Ogr2Ogr allows you to extract/export spatial database tables from PostGIS to many GIS file formats. Using the syntax below you can extract a PostGIS table to a GeoJSON file.

Command:

ogr2ogr -f "GeoJSON" outputlocation\outputfile.geojson PG:"host=server port=5432 user='username' password='password' dbname='PostgisDBName'" -sql "SELECT * from PGTablename"

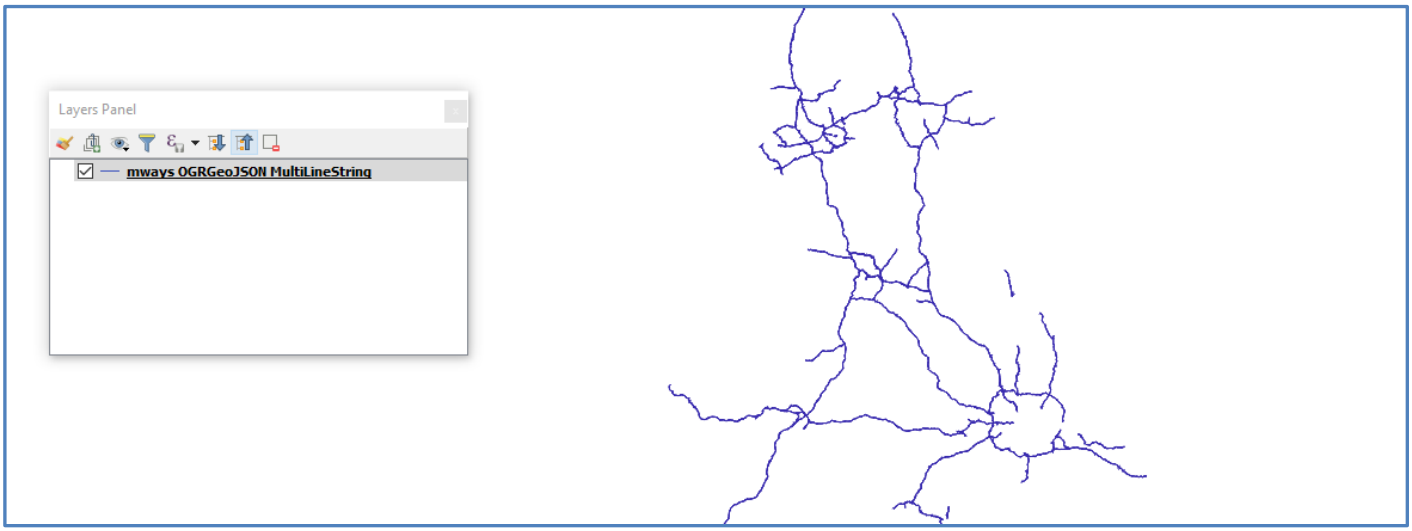
Example:

ogr2ogr -f "GeoJSON" D:\Temp\mways.geojson PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'" -sql "SELECT * from mways"

```
C:\OSGeo4W64\bin>ogr2ogr -f "GeoJSON" D:\Temp\mways.geojson PG:"host=localhost port=5432 user='postgres' password='
postgres' dbname='DynamicMaps'" -sql "SELECT * from mways"

C:\OSGeo4W64\bin>
```

The result will be a new GeoJSON file of Motorways.



5 – Extract data from PostGIS to an ESRI Shapefile

Ogr2Ogr allows you to extract/export spatial database tables from PostGIS to many GIS file formats. Using the syntax below you can extract a PostGIS table to an ESRI Shapefile.

Command:

ogr2ogr -f "ESRI Shapefile" outputlocation\outputfile.shp PG:"host=server port=5432 user='username' password='password' dbname='PostgisDBName'" -sql "SELECT * from PGTablename"

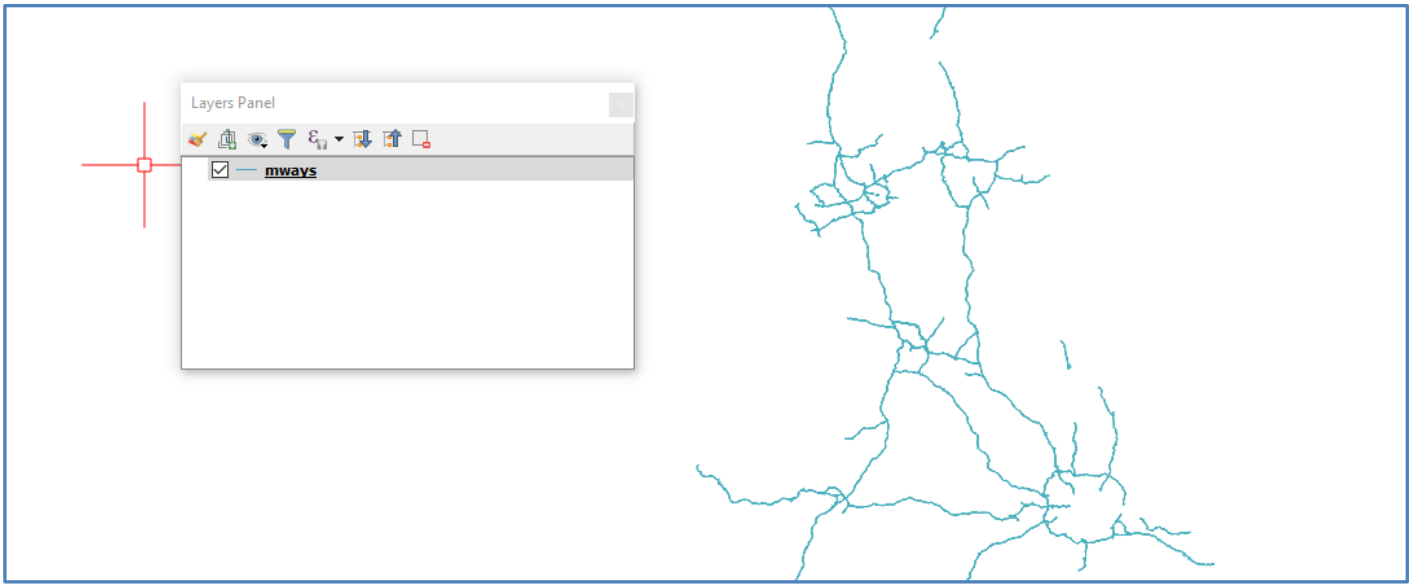
Example:

ogr2ogr -f "ESRI Shapefile" D:\Temp\mways.shp PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'" -sql "SELECT * from mways"

```
C:\OSGeo4W64\bin>ogr2ogr -f "ESRI Shapefile" D:\Temp\mways.shp PG:"host=localhost port=5432 user='postgres' passwor
d='postgres' dbname='DynamicMaps'" -sql "SELECT * from mways"

C:\OSGeo4W64\bin>
```

The result will be a new ESRI Shapefile file of Motorways.



6 – Extract data from PostGIS to an ESRI Shapefile and Re-project to 4326

Using the syntax below you can extract a PostGIS table to an ESRI Shapefile and reproject the extracted data to another projection.

Command:

ogr2ogr -f "ESRI Shapefile" outputlocation\outputfile.shp -t_srs "EPSG:4326" PG:"host=server port=5432 user='username' password='password' dbname='PostgisDBName'" -sql "SELECT * from PGTablename"

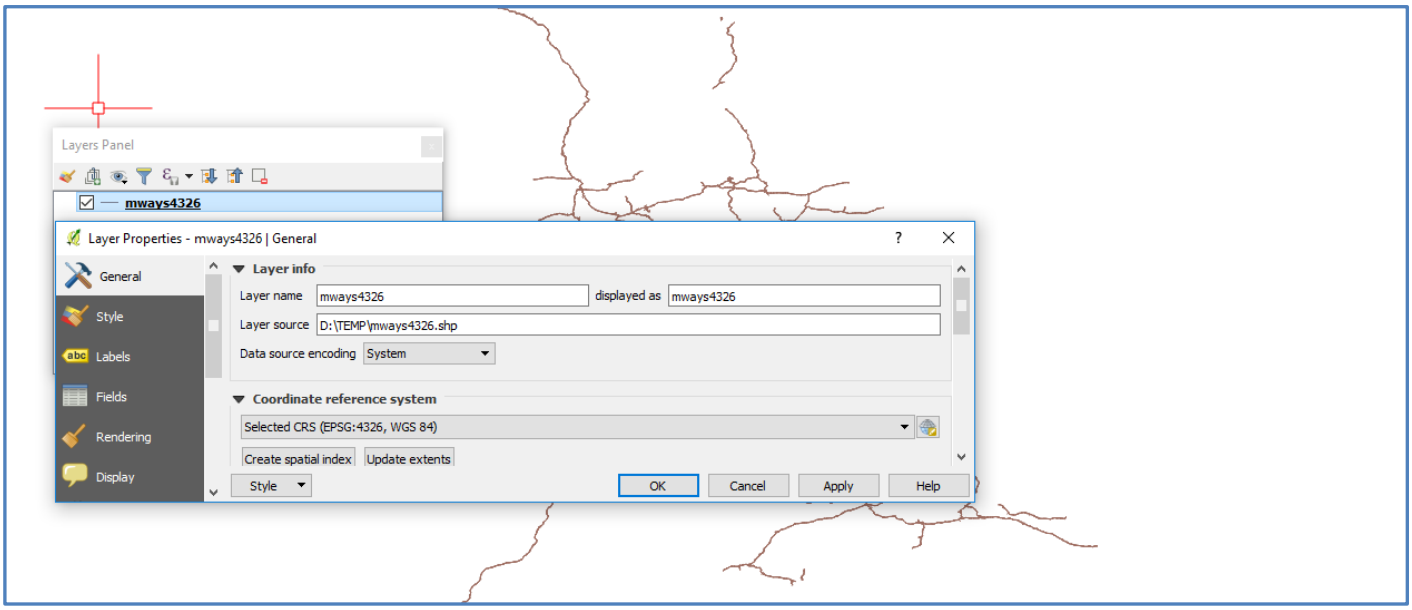
Example:

ogr2ogr -f "ESRI Shapefile" D:\Temp\mways4326.shp -t_srs "EPSG:4326" PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'" -sql "SELECT * from mways"

```
C:\OSGeo4W64\bin>ogr2ogr -f "ESRI Shapefile" D:\Temp\mways4326.shp -t_srs "EPSG:4326" PG:"host=localhost port=5432
user='postgres' password='postgres' dbname='DynamicMaps'" -sql "SELECT * from mways"

C:\OSGeo4W64\bin>
```

The result will be a new ESRI Shapefile file of Motorways now projected to 4326.



7 – Extract data from PostGIS to an ESRI Shapefile – Selecting Specific Fields

Using the syntax below you can extract a PostGIS table to an ESRI Shapefile and choose only to export specific fields. Here when we export the Mways table we will only export the 'number' field.

Command:

ogr2ogr -f "ESRI Shapefile" outputlocation\outputfile.shp -select "fieldname" PG:"host=server port=5432 user='username' password='password' dbname='PostgisDBName'" -sql "SELECT * from PGTablename"

Example:

ogr2ogr -f "ESRI Shapefile" D:\Temp\mways_Number.shp -select "number" PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'" -sql "SELECT * from mways"

```
C:\OSGeo4W64\bin>ogr2ogr -f "ESRI Shapefile" D:\Temp\mways_Number.shp -select "number" PG:"host=localhost port=5432
 user='postgres' password='postgres' dbname='DynamicMaps'" -sql "SELECT * from mways"

C:\OSGeo4W64\bin>
```

The result will be a new ESRI Shapefile file of Motorways with only the 'Number' field exported.

8 – Extract data from PostGIS to an ESRI Shapefile – Selecting Specific Records

Using the syntax below you can extract a PostGIS table to an ESRI Shapefile and using SQL you can choose to export specific records. Here when we export the Mways table we will export just the 'M90'.

Command:

ogr2ogr -f "ESRI Shapefile" outputlocation\outputfile.shp -select "fieldname" PG:"host=server port=5432 user='username' password='password' dbname='PostgisDBName'" -sql "SELECT * from PGTablename" where fieldname = 'value'

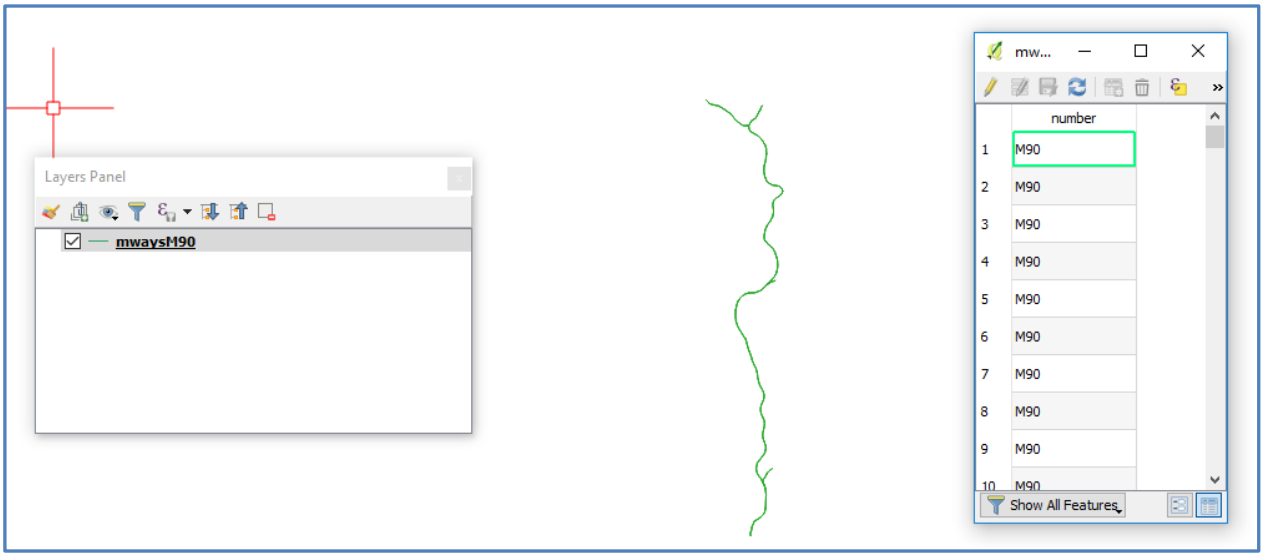
Example:

ogr2ogr -f "ESRI Shapefile" D:\Temp\mwaysM90.shp -select "number" PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'" -sql "SELECT * from mways where number = 'M90'

```
C:\OSGeo4W64\bin>ogr2ogr -f "ESRI Shapefile" D:\Temp\mwaysM90.shp -select "number" PG:"host=localhost port=5432 use
r='postgres' password='postgres' dbname='DynamicMaps'" -sql "SELECT * from mways where number = 'M90'

C:\OSGeo4W64\bin>
```

The result will be a new ESRI Shapefile file of Motorways with only the records where 'Number' value = 'M90'.



9 – Append Records to an existing PostGIS Table – Point Data

Using the syntax below you can upload and append the records from one GIS file into an existing table in your PostGIS database. Here we will append School Point objects to a PostGIS table.

Command:

ogr2ogr -append -f "PostgreSQL" PG:"dbname=db" shapefile.shp -nln mytable

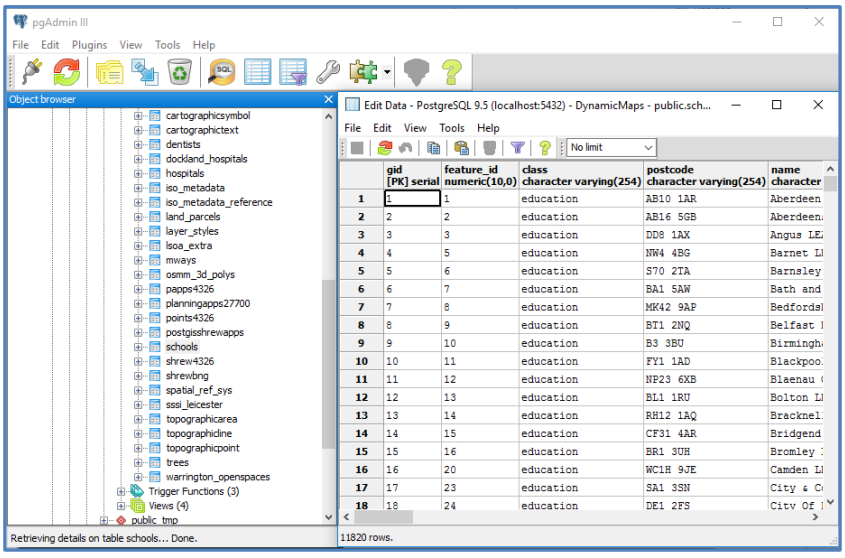
Example:

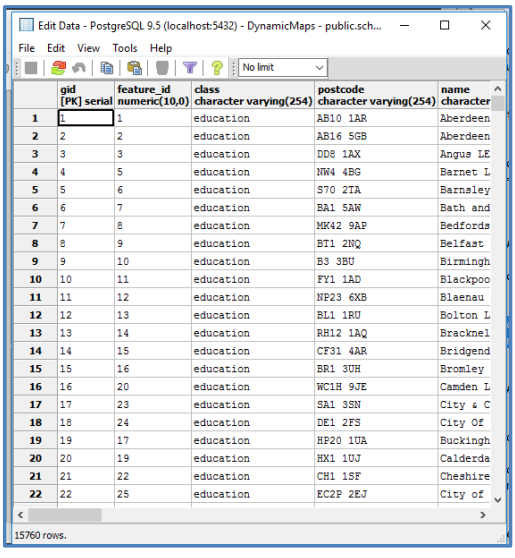
ogr2ogr -append -f "PostgreSQL" PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'" D:\Temp\schools.shp -nln schools

```
C:\OSGeo4W64\bin>ogr2ogr -append -f "PostgreSQL" PG:"host=localhost port=5432 user='postgres' password='postgres' d
bname='DynamicMaps'" D:\Temp\schools.shp -nln schools

C:\OSGeo4W64\bin>
```

The result will append the records from the Schools Shapefile into the existing records in the Schools PostGIS table. Using PGAdmin we can see how the 3940 records from the Schools Shapefile have now been appended to the existing 11820 records, to create a new record count of **15760**.





10 – Append Records to an existing PostGIS Table – Line Data

Using the syntax below you can upload and append the records from one GIS file into an existing table in your PostGIS database. Here we will append motorway line objects to a PostGIS table.

Command:

ogr2ogr -append -f "PostgreSQL" PG:"dbname=db" shapefile.shp -nln mytable

Example:

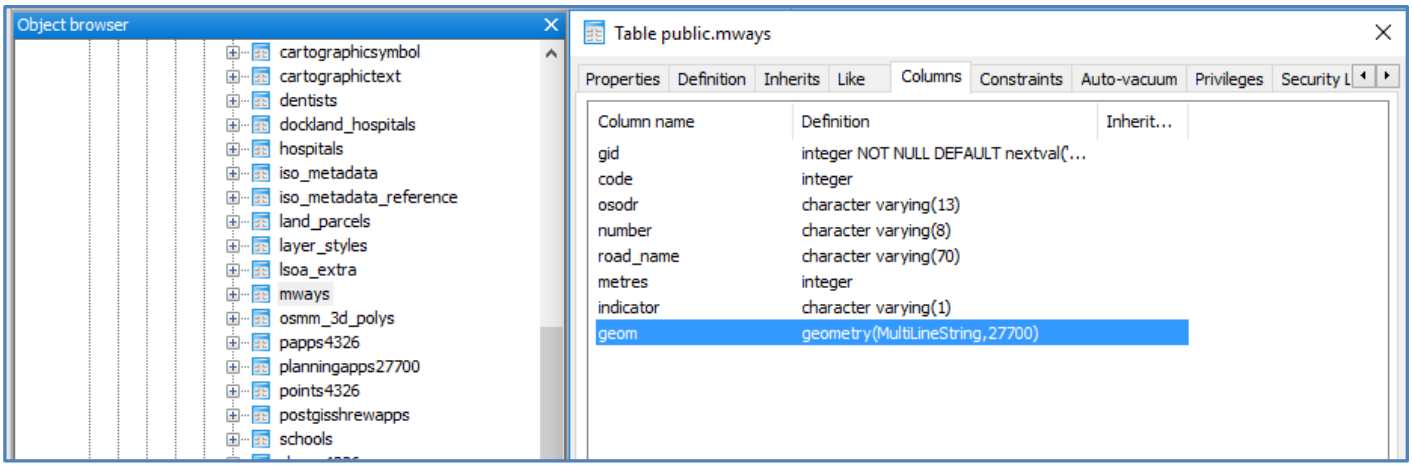
ogr2ogr -append -f "PostgreSQL" PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'" D:\Temp\mways.shp -nlt mways

```
C:\OSGeo4W64\bin>ogr2ogr -append -f "PostgreSQL" PG:"host=localhost port=5432 user='postgres' password='postgres' d
bname='DynamicMaps'" D:\Temp\mways.shp -nln mways
Warning 1: Geometry to be inserted is of type Line String, whereas the layer geometry type is Multi Line String.
Insertion is likely to fail
ERROR 1: ERROR:  Geometry type (LineString) does not match column type (MultiLineString)

ERROR 1: INSERT command for new feature failed.
ERROR:  Geometry type (LineString) does not match column type (MultiLineString)

Command: INSERT INTO "mways" ("geom" , "code", "osodr", "number", "metres") VALUES ('0102000020346C0000020000000000
000030E31241000000009E0A264100000000C0EB1241000000000D0072641'::GEOMETRY, 3000, 'O13U7WC504CAV', 'M90', 655) RETURNI
NG "gid"
ERROR 1: Unable to write feature 0 from layer mways.
ERROR 1: Terminating translation prematurely after failed
translation of layer mways (use -skipfailures to skip errors)
```

If however, there is a mismatch between the input files geometry format and the PostGIS database geometry format an error will be returned. In this instance the Shapefile contains **LineString** objects and the PostGIS database table has **MultiLineString** geometry.



To overcome this issue we can use the following options for either importing into a Line or Polygon table:

-nlt MULTIPOLGON

-nlt MULTILINESTRING

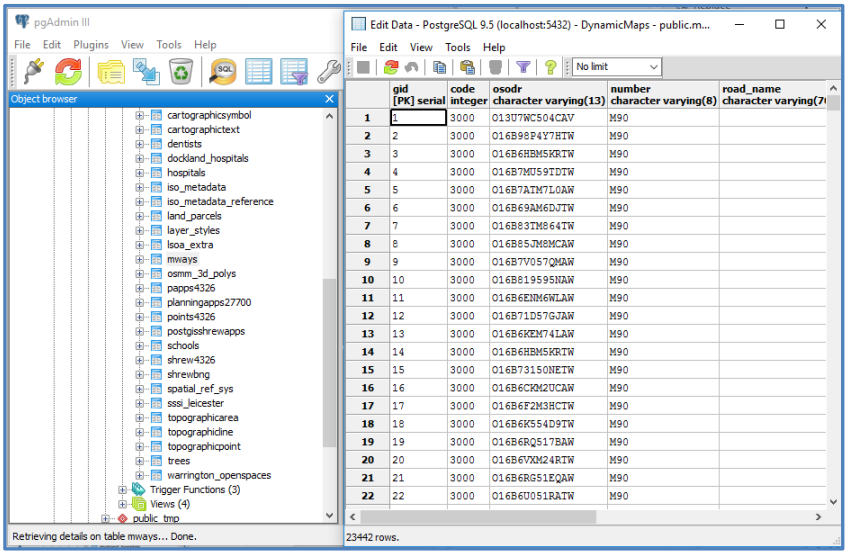
Example:

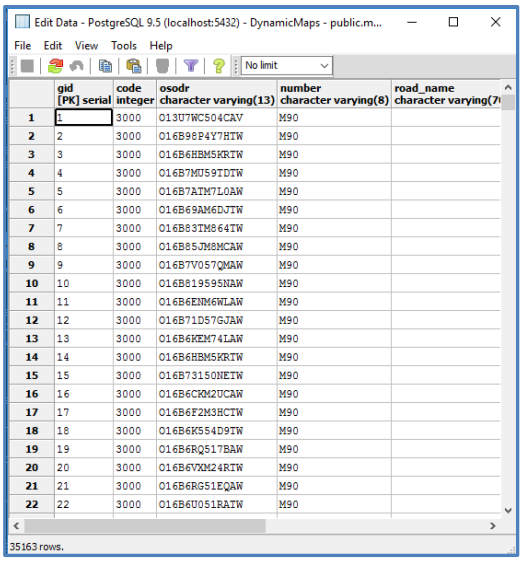
ogr2ogr -append -f "PostgreSQL" PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'" D:\Temp\mways.shp -nlt MULTILINESTRING mways

```
C:\OSGeo4W64\bin>ogr2ogr -append -f "PostgreSQL" PG:"host=localhost port=5432 user='postgres' password='postgres'
dbname='DynamicMaps'" D:\Temp\mways.shp -nlt MULTILINESTRING mways

C:\OSGeo4W64\bin>
```

The result will append the records from the Mways Shapefile into the existing records in the Mways PostGIS table. Using PGAdmin we can see how the 11721 records from the Mways Shapefile have now been appended to the existing 23442 records, to create a new record count of **35163.**





11 – INSERT records to a new PostGIS Table

Using the syntax below you can upload records from one GIS file and create a new PostGIS database table, as well as re-projecting the source data during import. Here we will upload a Trees Shapefile into PostGIS.

Command:

ogr2ogr -f PostgreSQL PG: ne_10m_admin_0_map_subunits.shp

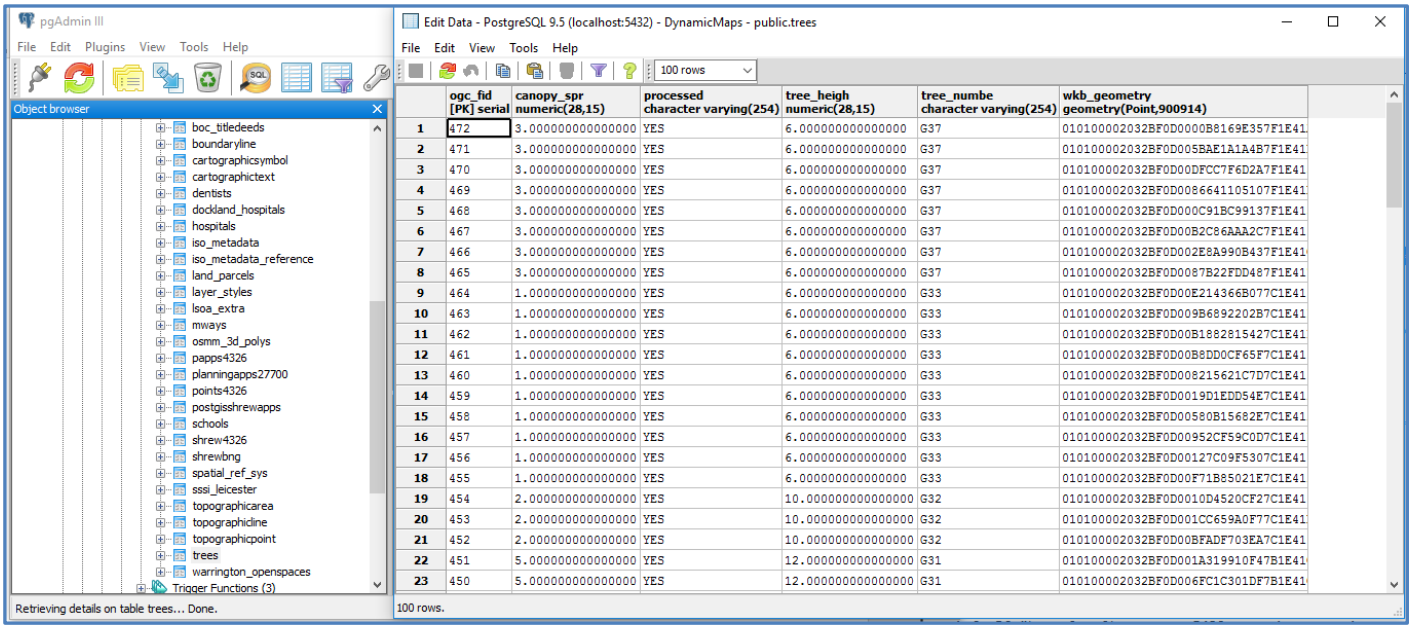
Example:

ogr2ogr -f "PostgreSQL" PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'" D:\Temp\Trees.shp

```
C:\OSGeo4W64\bin>ogr2ogr -f "PostgreSQL" PG:"host=localhost port=5432 user='postgres' password='postgres' dbname=
'DynamicMaps'" D:\Temp\Trees.shp

C:\OSGeo4W64\bin>
```

The result will create a new table called Trees within the PostGIS database.



	ogc_fid [PK] serial	canopy_spr numeric(28,15)	processed character varying(254)	tree_heigh numeric(28,15)	tree_numbe character varying(254)	wkb_geometry geometry(Point,900914)
1	472	3.000000000000000	YES	6.000000000000000	G37	010100002032BF0D0000B8169E357F1E41
2	471	3.000000000000000	YES	6.000000000000000	G37	010100002032BF0D005BAE1A1A4B7F1E41
3	470	3.000000000000000	YES	6.000000000000000	G37	010100002032BF0D00DFCC7F6D2A7F1E41
4	469	3.000000000000000	YES	6.000000000000000	G37	010100002032BF0D0086641105107F1E41
5	468	3.000000000000000	YES	6.000000000000000	G37	010100002032BF0D000C91BC99137F1E41
6	467	3.000000000000000	YES	6.000000000000000	G37	010100002032BF0D00B2C86AAA2C7F1E41
7	466	3.000000000000000	YES	6.000000000000000	G37	010100002032BF0D002E8A990B437F1E41
8	465	3.000000000000000	YES	6.000000000000000	G37	010100002032BF0D0087B22FDD487F1E41
9	464	1.000000000000000	YES	6.000000000000000	G33	010100002032BF0D00E214366B077C1E41
10	463	1.000000000000000	YES	6.000000000000000	G33	010100002032BF0D009B6892202B7C1E41
11	462	1.000000000000000	YES	6.000000000000000	G33	010100002032BF0D00B1882815427C1E41
12	461	1.000000000000000	YES	6.000000000000000	G33	010100002032BF0D00B8DD0CF65F7C1E41
13	460	1.000000000000000	YES	6.000000000000000	G33	010100002032BF0D008215621C7D7C1E41
14	459	1.000000000000000	YES	6.000000000000000	G33	010100002032BF0D0019D1EDD54E7C1E41
15	458	1.000000000000000	YES	6.000000000000000	G33	010100002032BF0D00580B15682E7C1E41
16	457	1.000000000000000	YES	6.000000000000000	G33	010100002032BF0D00952CF59C0D7C1E41
17	456	1.000000000000000	YES	6.000000000000000	G33	010100002032BF0D00127C09F5307C1E41
18	455	1.000000000000000	YES	6.000000000000000	G33	010100002032BF0D00F71B85021E7C1E41
19	454	2.000000000000000	YES	10.000000000000000	G32	010100002032BF0D0010D4520CF27C1E41
20	453	2.000000000000000	YES	10.000000000000000	G32	010100002032BF0D001CC659A0F77C1E41
21	452	2.000000000000000	YES	10.000000000000000	G32	010100002032BF0D00BFADF703EA7C1E41
22	451	5.000000000000000	YES	12.000000000000000	G31	010100002032BF0D001A319910F47B1E41
23	450	5.000000000000000	YES	12.000000000000000	G31	010100002032BF0D006FC1C301DF7B1E41

12 – INSERT records to a new PostGIS Table and Re-project to 4326

Using the syntax below you can upload records from one GIS file and create a new PostGIS database table. Here we will upload a Trees Shapefile into PostGIS and re-project that on import to 4326.

Command:

ogr2ogr -f PostgreSQL PG:"host=server port=5432 user='username' password='password' dbname='PostgisDBName'" shapefilename.shp -t_srs "EPSG:4326

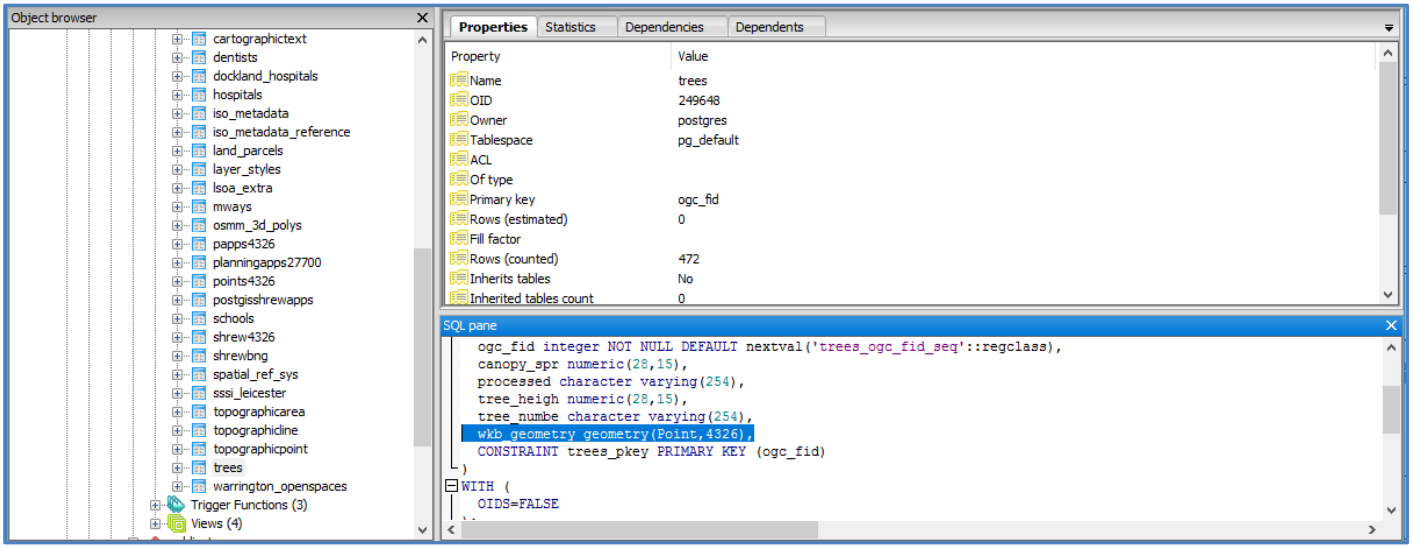
Example:

ogr2ogr -f "PostgreSQL" PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'" D:\Temp\Trees.shp -t_srs "EPSG:4326

```
C:\OSGeo4W64\bin>ogr2ogr -f "PostgreSQL" PG:"host=localhost port=5432 user='postgres' password='postgres' dbname=
'DynamicMaps'" D:\Temp\Trees.shp -t_srs "EPSG:4326

C:\OSGeo4W64\bin>
```

The result will create a new table called Trees within the PostGIS database, which has the projection of 4326.



13 – Update a PostGIS Database Table with a new Field

Before we can update values into a field in PostGIS, you may need to create that new field first. We can use ogrinfo to edit the Schema of a PostGIS database table. Here we will use the Trees table that we imported earlier and create a new field called 'Surveyor'.

Command:

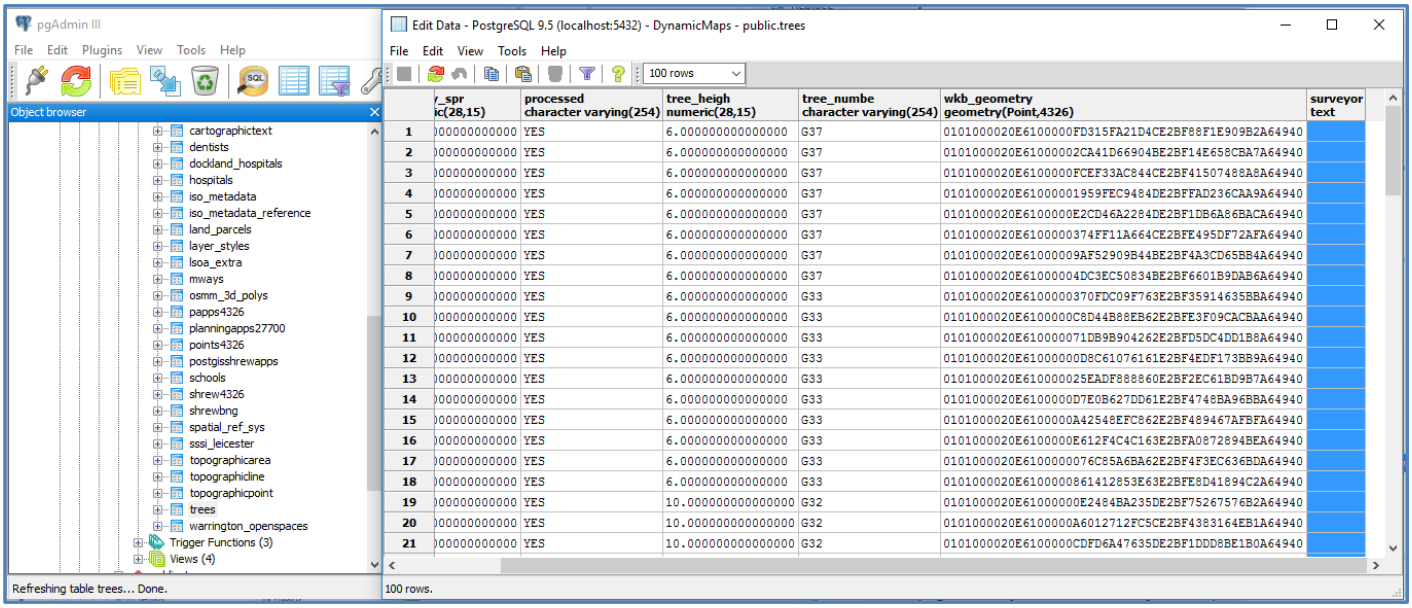
ogrinfo PG:"host=server port=5432 user='username' password='password' dbname='PostgisDBName'" -sql "ALTER TABLE test ADD COLUMN newfield TEXT"

Example:

ogrinfo PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'" -sql "ALTER TABLE trees ADD COLUMN Surveyor TEXT"

```
C:\OSGeo4W64\bin>ogrinfo PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'" -
sql "ALTER TABLE trees ADD COLUMN Surveyor TEXT"
INFO: Open of `PG:host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps''
      using driver `PostgreSQL' successful.
```

The result will alter the structure of the Trees table by adding a new Text/String field called Surveyor.



14 – Update a PostGIS Database Table with values in a Field

Now that we have a new field called ‘Surveyor’ we will use ogrinfo to update that field with new values for all records.

Command:

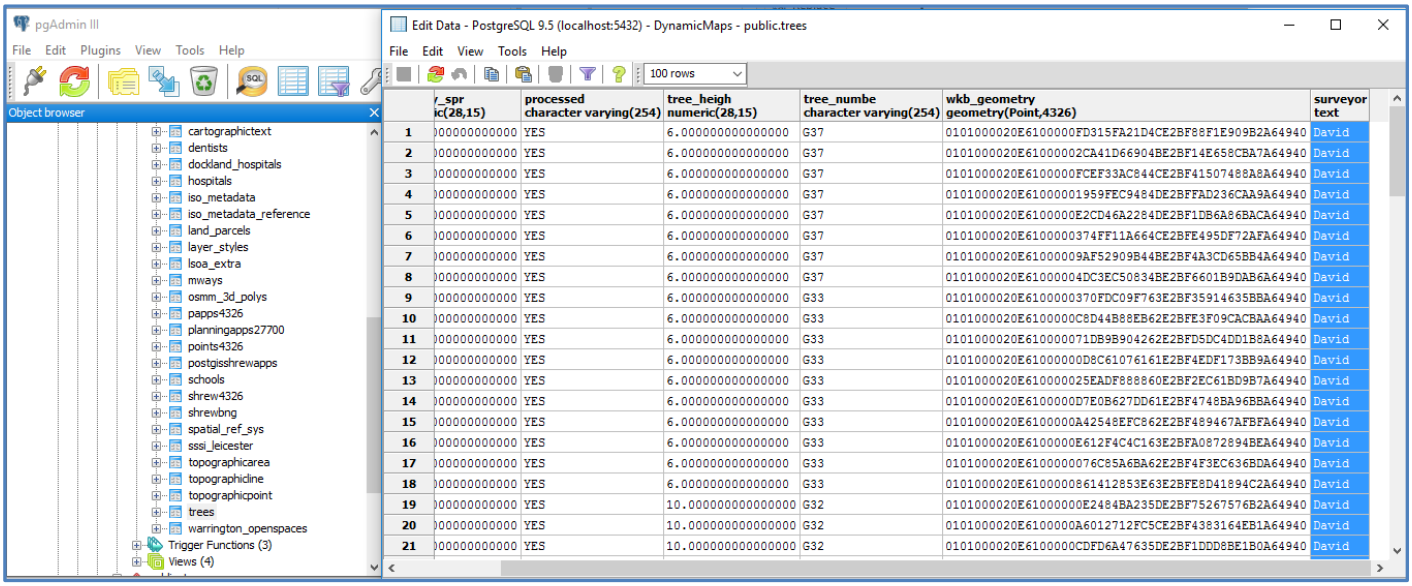
ogrinfo PG:"host=server port=5432 user='username' password='password' dbname='pgdatabase'" -dialect SQLite -sql "UPDATE table SET field = 'Value'"

Example:

ogrinfo PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'" -dialect SQLite -sql "UPDATE trees SET Surveyor = 'David'"

```
C:\OSGeo4W64\bin>ogrinfo PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'"
-dialect SQLite -sql "UPDATE trees SET Surveyor = 'David'"
INFO: Open of `PG:host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps''
      using driver `PostgreSQL' successful.
```

The result will update the ‘Surveyor’ field with the value of ‘David’ for all records.



	_spr ic(28,15)	processed character varying(254)	tree_heigh numeric(28,15)	tree_numbe character varying(254)	wkb_geometry geometry(Point,4326)	surveyor text
1	000000000000	YES	6.000000000000000	G37	0101000020E6100000FD315FA21D4CE2BF88F1E909B2A64940	David
2	000000000000	YES	6.000000000000000	G37	0101000020E61000002CA41D66904BE2BF14E658CBA7A64940	David
3	000000000000	YES	6.000000000000000	G37	0101000020E6100000FCEF33AC844CE2BF41507488A8A64940	David
4	000000000000	YES	6.000000000000000	G37	0101000020E61000001959FEC9484DE2BFFAD236CAA9A64940	David
5	000000000000	YES	6.000000000000000	G37	0101000020E6100000E2CD46A2284DE2BF1DB6A86BACA64940	David
6	000000000000	YES	6.000000000000000	G37	0101000020E6100000374FF11A664CE2BFE495DF72AFA64940	David
7	000000000000	YES	6.000000000000000	G37	0101000020E61000009AF52909B44BE2BF4A3CD65BB4A64940	David
8	000000000000	YES	6.000000000000000	G37	0101000020E61000004DC3EC50834BE2BF6601B9DAB6A64940	David
9	000000000000	YES	6.000000000000000	G33	0101000020E6100000370FDC09F763E2BF35914635BBA64940	David
10	000000000000	YES	6.000000000000000	G33	0101000020E6100000C8D44B88EB62E2BFE3F09CACBAA64940	David
11	000000000000	YES	6.000000000000000	G33	0101000020E61000071DB9B904262E2BFD5DC4DD1B8A64940	David
12	000000000000	YES	6.000000000000000	G33	0101000020E61000000D8C61076161E2BF4EDF173BB9A64940	David
13	000000000000	YES	6.000000000000000	G33	0101000020E610000025EADF888860E2BF2EC61BD9B7A64940	David
14	000000000000	YES	6.000000000000000	G33	0101000020E6100000D7E0B627DD61E2BF4748BA96BBA64940	David
15	000000000000	YES	6.000000000000000	G33	0101000020E6100000A42548EFC862E2BF489467AFBFA64940	David
16	000000000000	YES	6.000000000000000	G33	0101000020E6100000E612F4C4C163E2BFA0872894BEA64940	David
17	000000000000	YES	6.000000000000000	G33	0101000020E6100000076C85A6BA62E2BF4F3EC636BDA64940	David
18	000000000000	YES	6.000000000000000	G33	0101000020E61000008614128 53E63E2BFE8D41894C2A64940	David
19	000000000000	YES	10.000000000000000	G32	0101000020E61000000E2484BA235DE2BF75267576B2A64940	David
20	000000000000	YES	10.000000000000000	G32	0101000020E6100000A6012712FC5CE2BF4383164EB1A64940	David
21	000000000000	YES	10.000000000000000	G32	0101000020E6100000CDFD6A47635DE2BF1DDD8BE1B0A64940	David

15 – Update a PostGIS Database Table with values in a Field – Using a Where Clause

We can then expand on the SQL statement to only update those records which meet specific criteria. For example, only updating the records in the Trees table where the 'Processed' field = 'No'.

Command:

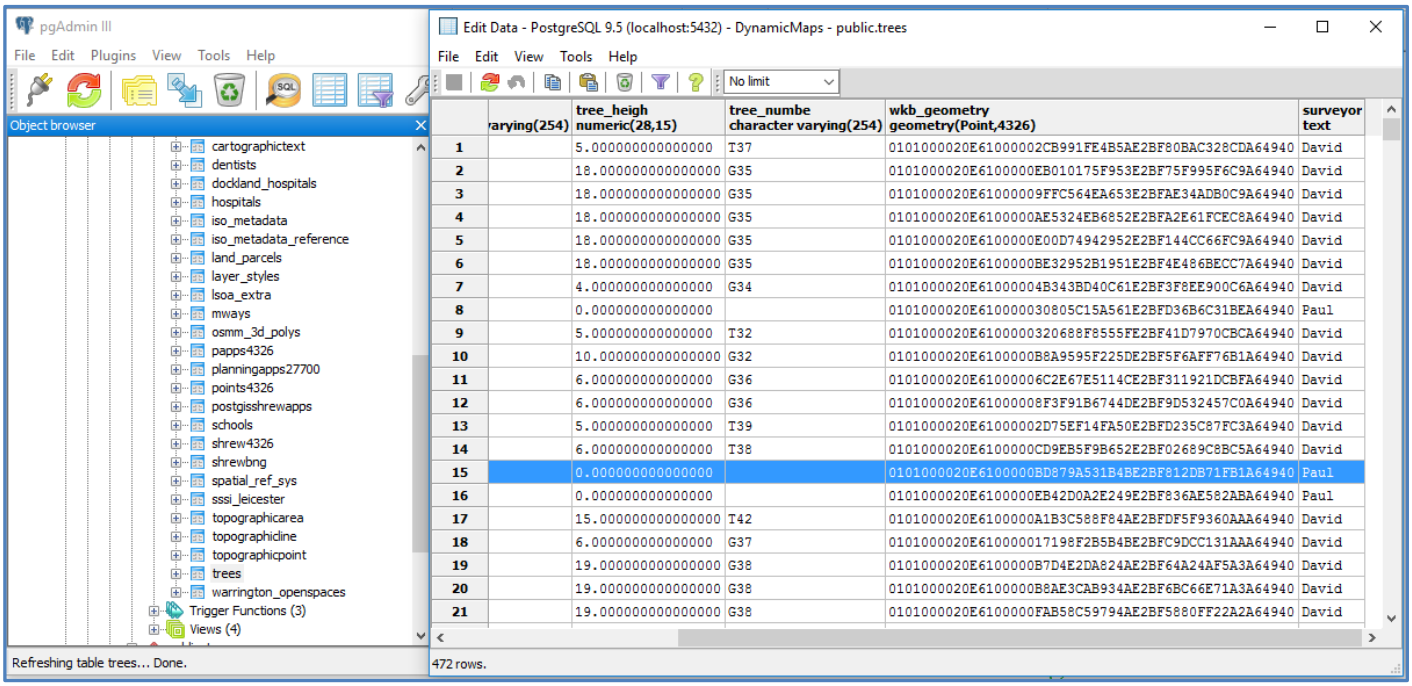
ogrinfo PG:"host=server port=5432 user='username' password='password' dbname='pgdatabase'" -dialect SQLite -sql "UPDATE table SET field = 'Value' where Processed = 'NO'"

Example:

ogrinfo PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'" -dialect SQLite -sql "UPDATE trees SET Surveyor = 'Paul' where Processed = 'NO'"

```
C:\OSGeo4W64\bin>ogrinfo PG:"host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps'"
-dialect SQLite -sql "UPDATE trees SET Surveyor = 'Paul' where Processed = 'NO'"
INFO: Open of `PG:host=localhost port=5432 user='postgres' password='postgres' dbname='DynamicMaps''
      using driver `PostgreSQL' successful.
```

The result will update the 'Surveyor' field with the value of 'Paul' for the selected records.



	varying(254)	tree_heigh numeric(28,15)	tree_numbe character varying(254)	wkb_geometry geometry(Point,4326)	surveyor text
1		5.000000000000000	T37	0101000020E61000002CB991FE4B5AE2BF80BAC328CDA64940	David
2		18.000000000000000	G35	0101000020E6100000EB010175F953E2BF75F995F6C9A64940	David
3		18.000000000000000	G35	0101000020E61000009FFC564EA653E2BFAE34ADB0C9A64940	David
4		18.000000000000000	G35	0101000020E6100000AE5324EB6852E2BFA2E61FCEC8A64940	David
5		18.000000000000000	G35	0101000020E6100000E00D74942952E2BF144CC66FC9A64940	David
6		18.000000000000000	G35	0101000020E6100000BE32952B1951E2BF4E486BECC7A64940	David
7		4.000000000000000	G34	0101000020E61000004B343BD40C61E2BF3F8EE900C6A64940	David
8		0.000000000000000		0101000020E610000030805C15A561E2BFD36B6C31BEA64940	Paul
9		5.000000000000000	T32	0101000020E6100000320688F8555FE2BF41D7970CBCA64940	David
10		10.000000000000000	G32	0101000020E6100000B8A9595F225DE2BF5F6AFF76B1A64940	David
11		6.000000000000000	G36	0101000020E61000006C2E67E5114CE2BF311921DCBFA64940	David
12		6.000000000000000	G36	0101000020E61000008F3F91B6744DE2BF9D532457C0A64940	David
13		5.000000000000000	T39	0101000020E61000002D75EF14FA50E2BFD235C87FC3A64940	David
14		6.000000000000000	T38	0101000020E6100000CD9EB5F9B652E2BF02689C8BC5A64940	David
15		0.000000000000000		0101000020E6100000BD879A531B4BE2BF812DB71FB1A64940	Paul
16		0.000000000000000		0101000020E6100000EB42D0A2E249E2BF836AE582ABA64940	Paul
17		15.000000000000000	T42	0101000020E6100000A1B3C588F84AE2BFDF5F9360AAA64940	David
18		6.000000000000000	G37	0101000020E610000017198F2B5B4BE2BFC9DCC131AAA64940	David
19		19.000000000000000	G38	0101000020E6100000B7D4E2DA824AE2BF64A24AF5A3A64940	David
20		19.000000000000000	G38	0101000020E6100000B8AE3CAB934AE2BF6BC66E71A3A64940	David
21		19.000000000000000	G38	0101000020E6100000FAB58C59794AE2BF5880FF22A2A64940	David