



Infrastructure Transitions Research Consortium

Working paper series

Derived and processed data report

David Alderson and Stuart Barr,
School of Civil Engineering and Geosciences, Newcastle University

June 2013

Table of Contents

Table of Contents	1
List of Figures.....	3
Software employed during processing:	4
Proprietary.....	4
Open Source	5
Common input data formats:	6
Data processing and uploading	7
ltrc_census.....	7
ltrc_economics	16
ltrc_population	19
ltrc_spatial_energy	21
ltrc_spatial_hazards.....	36
ltrc_spatial_water.....	38
ltrc_spatial_wastewater	41
ltrc_spatial_solidwaste	46
ltrc_ws1_solid_waste_distances	51
ltrc_spatial_transport	54
Ordnance Survey Points of Interest.....	64
Ordnance Survey Vectormap District	66
Ordnance Survey MasterMap.....	67
Appendix:.....	68
plpgsql function code for ws2_build_od_table.....	68
plpgsql function code for ws2_nearest_feature_to_centroid	73
plpgsql function code for ws2_nearest_feature_to_centroid_in_zone	76
plpgsql function code for ws2_all_features_in_zone	79
plpgsql function code for ws2_nearest_feature_to_centroid_in_out_zone_table.....	81
SQL Code and Django links to create Population CDAM value tables:	84
SQL Code to create Economics CDAM value tables:.....	85
SQL Code for foreign key constraints on European Environment Agency Waste Water Data	89
plpgsql function code for extracting country-specific routes from openflights data.....	91
plpgsql function code for calculate_GOR_centroid_to_boundary_distance for solid waste CDAM	94

List of Figures

Figure 1 - itrc_census database data processing steps. Contains CENSUS Journey To Work Data for 2001	15
Figure 2 - processing steps undertaken to create tables to store economics CDAM constants.....	18
Figure 3 - processing steps undertaken to create tables to store demographics CDAM constants	20
Figure 4 - GeoHack coordinate information available via Wikipedia, for Aberthaw Power Station.....	22
Figure 5 - GeoHack coordinate information available via Wikipedia, for Little Barford Power Station	24
Figure 6 - distribution lines highlighted in cyan, as polylines in dxf	25
Figure 7 - substations highlighted in cyan, as polygons within dxf.....	26
Figure 8 - names of substations from Annotation feature type highlighted in cyan	26
Figure 9 - processing steps to create and load data to the energy section of the database	35
Figure 10 - processing and upload procedure for data within the itrc hazards database	37
Figure 11 - processing and upload procedure for data within the itrc water database	40
Figure 12 - processing and upload procedures for waste water data	45
Figure 13 - National Grid Letter Mapping (http://www.ordnancesurvey.co.uk/oswebsite/gi/nationalgrid/nationalgrid.pdf)	48
Figure 14 - processing and upload procedures for solid waste data	50
Figure 15 - processing for derived data to calculate distances for WS1 Solid Waste CDAM.....	53
Figure 16 - processing and upload procedures for transport data	63
Figure 17 - processing and upload procedures for Ordnance Survey Points of Interest data (North West coverage only).....	65
Figure 18 - processing and upload procedure for Ordnance Survey Vectormap District data	66
Figure 19 - processing and upload procedure overview for North West coverage of Ordnance Survey MasterMap data.....	67
Figure 20 - plpgsql function code to build an origin-destination table, creating an output table containing both the origin zone and origin feature, as well as the destination zone and the destination feature	72
Figure 21 - plpgsql function code to calculate nearest feature to a zone centroid, whether inside or outside the feature is inside or outside the given zone geometry	75
Figure 22 - plpgsql function code to calculate the nearest feature to the centroid of each zone, that lies within that zone	78
Figure 23 - plpgsql function to create a table of all features within each zone	80
Figure 24 - plpgsql function to create a table of nearest feature to a zone centroid, beginning by selecting the nearest feature within a zone, then if no features exist within that zone, finding the nearest.	83
Figure 25 - value table definitions for population CDAM	84
Figure 26 - value table definitions for economics CDAM	88
Figure 27 - plpgsql function code to extract country-specific flight routes from openflights data	93
Figure 28 - plpgsql function code to calculate average straight line distance between output area centroid and centroid of government office region with which the output area resides.	94

Software employed during processing:

Proprietary

1. Adobe Illustrator
2. Safe Software FME Desktop 2011 SP2
 - Commonly-used feature type readers/writers
(http://fmepedia.safe.com/articles/Documentation_del/FME-Readers-and-Writers)
 - CSV - http://docs.safe.com/fme/reader_writerPDF/csv.pdf
 - Microsoft Excel - http://docs.safe.com/fme/reader_writerPDF/xls_ado.pdf
 - Microsoft Access - http://docs.safe.com/fme/reader_writerPDF/mdb_ado.pdf
 - ESRI Shapefile - http://docs.safe.com/fme/reader_writerPDF/shape.pdf
 - PostgreSQL - http://docs.safe.com/fme/reader_writerPDF/postgres.pdf
 - PostGIS - http://docs.safe.com/fme/reader_writerPDF/postgis.pdf
 - GML (OS MasterMap) - http://docs.safe.com/fme/reader_writerPDF/gml2.pdf
3. ESRI ArcGIS (ArcMap, ArcToolbox), Version 10, Service Pack 4
 - Analysis Tools
 - Extract
 - Clip - <http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/Clip/000800000004000000/>
 - Data Management Tools
 - Features
 - Feature To Point - <http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/001700000003m0000000/>
 - Feature Vertices To Points - http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/Feature_Vertices_To_Points/001700000003p0000000/
 - Multipart To Singlepart - http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/Multipart_To_Singlepart/001700000003r0000000/
 - Split Line At Point - http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/Split_Line_At_Point/001700000003w0000000/
 - Split Line At Vertices - http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/Split_Line_At_Vertices/001700000003z0000000/
 - General
 - Merge - <http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/Merge/001700000005000000/>
 - Generalization
 - Dissolve - <http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/Dissolve/001700000005n0000000/>

- Projections and Transformations
 - Feature
 - Project - <http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/Project/0017000007m000000/>
 - Define Projection - http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/Define_Projection/001700000077000000/

Open Source

- PostgreSQL 9.03, PostGIS 1.5, pgAdmin 1.12.x
 - PostGIS Shp and DBF Loader (Revision: 5983)
- PL/PGSQL – procedural language
 - Custom functions used:
 - **ws2_all_features_in_zone** – calculates using ST_Intersects all features that lie within each zone boundary supplied
 - **ws2_nearest_feature_to_centroid_in_zone** – calculates the nearest feature to the centroid of each zone, for only features that lie within that zone
 - **ws2_nearest_feature_to_centroid** – calculates the nearest feature to the centroid of each zone, regardless of whether that centroid lies within that zone or not.
 - **ws2_nearest_feature_to_centroid_in_out_zone_table** – calculates the nearest feature to the centroid of each zone, beginning by matching the nearest feature to each zone centroid that lies within that zone, and then finding the nearest feature if no features lie within the zone.
 - **ws2_build_od_table** – builds a table of origin and destination zones and features to use as origin or destination features when trying to consider start and end of a shortest path calculation.
 - **subset_openflights_routes_by_country** – subsets openflights routes data based on input set of airports, and returns only those flights beginning and ending at airports specified in the input set
 - **calculate_GOR_centroid_to_boundary_distance** – creates a table containing the average straight line distance between an output area centroid and all vertices that comprise the boundary of the government office region within which the output area resides.

Common input data formats:

Below are listed some of the common data formats used to store data that has been loaded in to the database. Alongside each format is a link to either a specification for that format, or simply further information about that particular format.

- ESRI Shapefile - <http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf>
- Microsoft Access Database - <http://office.microsoft.com/en-gb/access-help/access-2010-specifications-HA010341462.aspx>
- Microsoft Excel Spreadsheet - <http://office.microsoft.com/en-gb/excel-help/excel-specifications-and-limits-HP005199291.aspx>
- Comma separated files (.csv) – http://en.wikipedia.org/wiki/Comma-separated_values
- Geographic Markup Language (GML) - <http://www.opengeospatial.org/standards/gml>
- Keyhole Markup Language (KML) - <https://developers.google.com/kml/>

Data processing and uploading

Itrc_census

Table prefix/table name	Process Description
JTW_Bicycle_2001, JTW_Bus_2001, JTW_CarPool_Northern_Ireland_Only_2001, JTW_Car_Driver_2001, JTW_Car_Passenger_2001, JTW_Motorcycle_2001, JTW_OnFoot_2001, JTW_Other_2001, JTW_Taxi_2001, JTW_Total_2001, JTW_Train_2001, JTW_Underground_England_Wales_Only_2001 , JTW_WorkStudyAtHome_2001	<p>1) Create a new Microsoft Excel Spreadsheet document</p> <p>2) Import each Journey To Work csv file to the new document from 1) creating a separate worksheet per mode e.g. bicycle, bus, train.</p> <p>3) Create a blank workspace in Safe Software, Feature Manipulation Engine (FME).</p> <p>4) Load each worksheet as a separate import feature type, via a Microsoft Excel feature reader.</p> <p>5) Create a postgresql feature type writer, to write each worksheet out to the itrc_census database as a separate table.</p> <p>6) Create a feature type per transport mode, asking FME to interpret and copy the column names and associated data types to the output feature type writers.</p> <p>7) Link the input and output feature type readers and writers. The output feature attributes should all turn “green” if they are able to accept the input data types.</p> <p>The final result is a separate Postgres table per transport mode.</p>
WICID_JTW_2001_Interaction_Data_Districts	<p>1) Load the district geometry data for England and Wales only, supplied by Nick Malleson alongside demographic projections, using the PostGIS shapefile and DBF Loader – results in table called districts</p> <p>2) Load the boundary geometry for the Scottish Council Areas for 2001, using the PostGIS shapefile and DBF Loader – results in a table called scotland_ca_2001</p> <p>3) Load the boundary geometry for the Welsh Unitary Authorities for 2001, using the PostGIS shapefile and DBF Loader – results in a table called wales_ua_2001</p> <p>4) Load the table England_districts_scotland_ca_2001_wales_ua_2001_centroids_ft p using PostGIS shapefile and DBF Loader. This table contains geometry of centroids of districts calculated using ESRI ArcGIS Feature To Point tool.</p> <p>4) UNION the three tables, districts, Scotland_ca_2001, Wales_ua_2001 to create a table containing all the boundary geometry data called england_districts_scotland_ca_2001_wales_ua_2001</p>

	<p>This table should contain 408 rows</p> <p>5) RUN SQL:</p> <pre>CREATE TABLE "WICID_JTW_2001_Interaction_Data_Districts" AS SELECT code as "ZoneCode", "name" AS "ZoneName", geom AS geom FROM england_districts_scotland_ca_2001_wales_ua_2001</pre> <p>This will result in a table containing the correct Journey To Work Origin-Destination districts for England, Wales, Scotland.</p> <p>6) RUN SQL:</p> <pre>ALTER TABLE "WICID_JTW_2001_Interaction_Data_Districts" ADD COLUMN centroid_geom geometry;</pre> <pre>ALTER TABLE "WICID_JTW_2001_Interaction_Data_Districts" ADD COLUMN ftp_geom geometry;</pre> <p>7) RUN SQL:</p> <pre>--set the geometry of the interaction districts to be the same as the combination of england/scotland/wales districts UPDATE "WICID_JTW_2001_Interaction_Data_Districts" SET geom = england_districts_scotland_ca_2001_wales_ua_2001.geom FROM england_districts_scotland_ca_2001_wales_ua_2001 WHERE england_districts_scotland_ca_2001_wales_ua_2001.code = "WICID_JTW_2001_Interaction_Data_Districts"."ZoneCode";</pre> <pre>--set the ftp_geom of the interaction districts to be the same as the centroids calculated from ArcGIS Feature To Point on england_districts_scotland_ca_2001_wales_ua_2001 UPDATE "WICID_JTW_2001_Interaction_Data_Districts" SET ftp_geom = england_districts_scotland_ca_2001_wales_ua_2001_centroids_ftp.geom FROM england_districts_scotland_ca_2001_wales_ua_2001_centroids_ftp WHERE england_districts_scotland_ca_2001_wales_ua_2001_centroids_ftp.code = "WICID_JTW_2001_Interaction_Data_Districts"."ZoneCode";</pre> <pre>--set the centroid_geom of the interaction districts to be calculated via ST_Centroid on the geometry of the districts i.e. NOT USING THE FEATURE TO POINT CONVERSION OF ARCGIS UPDATE "WICID_JTW_2001_Interaction_Data_Districts" SET centroid_geom = ST_Centroid(england_districts_scotland_ca_2001_wales_ua_2001.geom) FROM england_districts_scotland_ca_2001_wales_ua_2001 WHERE england_districts_scotland_ca_2001_wales_ua_2001.geom IS NOT NULL AND england_districts_scotland_ca_2001_wales_ua_2001.code = "WICID_JTW_2001_Interaction_Data_Districts"."ZoneCode";</pre> <p>The result is a table containing the correct geometry for the Journey To Work Origin – Destination Zones, and their centroids.</p>
os_meridian_2_stations_per_district_st_centroid	<p>1) Ordnance Survey Meridian 2 Stations (station_point.shp) shapefile uploaded to itrc_census database, via PostGIS Shapefile and DBF Loader.</p> <p>2) Execute plpgsql function ws2_all_features_in_zone :</p>

	<p>SELECT * FROM ws2_all_features_in_zone('WICID_JTW_2001_Interaction_Data_Districts', 'geom', 'centroid_geom', 'ZoneCode', 'os_meridian_2_old_station_point', 'geom', 'name', 27700, 'os_meridian_2_stations_per_district_st_centroid');</p> <p>Result is a table of OS Meridian 2 Railway Stations within each district, using the ST_Intersects PostGIS function.</p> <p>Attributes include:</p> <p>“id” – serial type (unique), generated by function ws2_all_features_in_zone “name” – railway station name “ZoneCode” – unique identifier for each district e.g. 00AA Distance – straight line distance between railway station and district centroid, calculated using PostGIS function ST_Distance Feature_geom – POINT geometry of railway station Zone_boundary_geom – POLYGON geometry of district boundary Zone_centroid_geom – POINT geometry of district geometry centroid, calculated in this case using ST_Centroid PostGIS function</p>
os_meridian_2_stations_per_district_ftp	<p>1) Ordnance Survey Meridian 2 Stations (station_point.shp) shapefile uploaded to itrc_census database, via PostGIS Shapefile and DBF Loader.</p> <p>2) Execute plpgsql function ws2_all_features_in_zone :</p> <p>SELECT * FROM ws2_all_features_in_zone('WICID_JTW_2001_Interaction_Data_Districts', 'geom', 'ftp_geom', 'ZoneCode', 'os_meridian_2_old_station_point', 'geom', 'name', 27700, 'os_meridian_2_stations_per_district_ftp');</p> <p>Result is a table of OS Meridian 2 Railway Stations within each district, using the ST_Intersects PostGIS function.</p> <p>Attributes include:</p> <p>“id” – serial type (unique), generated by function ws2_all_features_in_zone “name” – OS Meridian 2 Railway Station name “ZoneCode” – unique identifier for each district e.g. 00AA Distance – straight line distance between railway station and district centroid, calculated using PostGIS function ST_Distance Feature_geom – POINT geometry of railway station Zone_boundary_geom – POLYGON geometry of district boundary Zone_centroid_geom – POINT geometry of district geometry centroid, calculated in this case using ESRI ArcGIS Tool Feature To Point</p>
Nearest_station_district_centroid_all_stations_ftp	<p>1) RUN SQL:</p> <p>SELECT * FROM</p>

	<p>ws2_nearest_feature_to_centroid('WICID_JTW_2001_Interaction_Data_Districts', 'ftp_geom', 'ZoneCode', 'os_meridian_2_old_station_point', 'geom', 'name', 27700, 'nearest_station_district_centroid_all_stations_ftp');</p> <p>The result is a table containing the nearest Ordnance Survey Meridian 2 Railway Station to each district centroid, for all stations, whether inside the chosen district or not.</p> <p>Attributes include:</p> <p>“id” – serial type (unique), generated by function ws2_nearest_feature_to_centroid</p> <p>“name” – OS Meridian 2 Railway Station name</p> <p>“ZoneCode” – unique identifier for each district e.g. 00AA</p> <p>Distance – straight line distance between railway station and district centroid (previously calculated using ESRI ArcGIS Feature To Point Tool)</p> <p>Feature_geom – OS Meridian 2 Railway Station geometry</p> <p>Centroid_geom – Centroid geometry of each district</p> <p>Nearest_feature_to_centroid_line_geom – Straight line geometry representing link between district centroid and nearest railway station.</p>
Nearest_station_district_centroid_all_stations_st_cent	<p>1) RUN SQL:</p> <p>SELECT * FROM ws2_nearest_feature_to_centroid('WICID_JTW_2001_Interaction_Data_Districts', 'centroid_geom', 'ZoneCode', 'os_meridian_2_old_station_point', 'geom', 'name', 27700, 'nearest_station_district_centroid_all_stations_st_cent');</p> <p>The result is a table containing the nearest Ordnance Survey Meridian 2 Railway Station to each district centroid, for all stations, whether inside the chosen district or not.</p> <p>Attributes include:</p> <p>“id” – serial type (unique), generated by function ws2_nearest_feature_to_centroid</p> <p>“name” – OS Meridian 2 Railway Station name</p> <p>“ZoneCode” – unique identifier for each district e.g. 00AA</p> <p>Distance – straight line distance between railway station and district centroid (previously calculated using ST_Centroid PostGIS function)</p> <p>Feature_geom – OS Meridian 2 Railway Station geometry</p> <p>Centroid_geom – Centroid geometry of each district</p> <p>Nearest_feature_to_centroid_line_geom – Straight line geometry representing link between district centroid and nearest railway station.</p>
Nearest_station_district_centroid_in_districts_ftp	<p>1) RUN SQL:</p> <p>SELECT * FROM ws2_nearest_feature_to_centroid_in_zone('os_meridian_2_stations_per_district_ftp', 'feature_geom', 'zone_centroid_geom', 'name', 'ZoneCode', 27700,</p>

	<p>'nearest_station_district_centroid_in_districts_ftp');</p> <p>The result is a table containing the nearest Ordnance Survey Meridian 2 Railway Station to each district centroid, for all stations, that lie inside that district.</p> <p>Note: centroids within the 'os_meridian_2_stations_per_district_ftp' table were calculated using the ESRI ArcGIS Feature To Point Tool.</p> <p>Attributes include:</p> <p>"id" – serial type (unique), generated by function ws2_nearest_feature_to_centroid_in_zone "name" – OS Meridian 2 Railway Station name "ZoneCode" – unique identifier for each district e.g. 00AA Distance – straight line distance between railway station and district centroid (previously calculated using ST_Centroid PostGIS function) Feature_geom – OS Meridian 2 Railway Station geometry Centroid_geom – Centroid geometry of each district Nearest_feature_to_centroid_line_geom – Straight line geometry representing link between district centroid and nearest railway station.</p>
Nearest_station_district_centroid_in_districts_st_cent	<p>1) RUN SQL:</p> <pre>SELECT * FROM ws2_nearest_feature_to_centroid_in_zone('os_meridian_2_stations_per_district_st_centroid', 'feature_geom', 'zone_centroid_geom', 'name', 'ZoneCode', 27700, 'nearest_station_district_centroid_in_districts_st_cent');</pre> <p>The result is a table containing the nearest Ordnance Survey Meridian 2 Railway Station to each district centroid, for all stations, that lie inside that district.</p> <p>Note: centroids within the 'os_meridian_2_stations_per_district_st_cent' table were calculated using the PostGIS ST_Centroid function.</p> <p>Attributes include:</p> <p>"id" – serial type (unique), generated by function ws2_nearest_feature_to_centroid_in_zone "name" – OS Meridian 2 Railway Station name "ZoneCode" – unique identifier for each district e.g. 00AA Distance – straight line distance between railway station and district centroid (previously calculated using ST_Centroid PostGIS function) Feature_geom – OS Meridian 2 Railway Station geometry Centroid_geom – Centroid geometry of each district Nearest_feature_to_centroid_line_geom – Straight line geometry representing link between district centroid and nearest railway</p>

	station.
nearest_station_to_district_centroid_in_out_districts_st_cent	<p>1) RUN SQL:</p> <pre>SELECT * FROM ws2_nearest_feature_to_centroid_in_out_zone_table('os_meridian_2_stations_per_district_st_centroid', 'feature_geom', 'zone_centroid_geom', 'name', 'ZoneCode', 'WICID_JTW_2001_Interaction_Data_Districts', 'os_meridian_2_old_station_point', 27700, 'nearest_station_to_district_centroid_in_out_districts_st_cent');</pre> <p>The result is a table containing the nearest Ordnance Survey Meridian 2 Railway Station to each district centroid, for all stations, by matching the nearest station to each zone centroid that lies within that zones boundary. In the event that a zone does not contain a station, then the nearest station outside the zone is selected.</p> <p>Attributes include:</p> <p>“id” – serial type (unique), generated by function ws2_nearest_feature_to_centroid_in_out_zone_table “name” – OS Meridian 2 Railway Station name “ZoneCode” – unique identifier for each district e.g. 00AA Distance – straight line distance between railway station and district centroid (previously calculated using ST_Centroid PostGIS function) Feature_geom – OS Meridian 2 Railway Station geometry Centroid_geom – Centroid geometry of each district Nearest_feature_to_centroid_line_geom – geometry of straight line between feature_geom and centroid_geom</p>
nearest_station_to_district_centroid_in_out_districts_ftp	<p>1) RUN SQL:</p> <pre>SELECT * FROM ws2_nearest_feature_to_centroid_in_out_zone_table('os_meridian_2_stations_per_district_ftp', 'feature_geom', 'zone_centroid_geom', 'name', 'ZoneCode', 'WICID_JTW_2001_Interaction_Data_Districts', 'os_meridian_2_old_station_point', 27700, 'nearest_station_to_district_centroid_in_out_districts_ftp');</pre> <p>The result is a table containing the nearest Ordnance Survey Meridian 2 Railway Station to each district centroid, for all stations, by matching the nearest station to each zone centroid that lies within that zones boundary. In the event that a zone does not contain a station, then the nearest station outside the zone is selected.</p> <p>Attributes include:</p> <p>“id” – serial type (unique), generated by function ws2_nearest_feature_to_centroid_in_out_zone_table “name” – OS Meridian 2 Railway Station name “ZoneCode” – unique identifier for each district e.g. 00AA</p>

	<p>Distance – straight line distance between railway station and district centroid (previously calculated using ST_Centroid PostGIS function)</p> <p>Feature_geom – OS Meridian 2 Railway Station geometry</p> <p>Centroid_geom – Centroid geometry of each district</p> <p>Nearest_feature_to_centroid_line_geom – geometry of straight line between feature_geom and centroid_geom</p>
JTW_Interaction_Districts_2001_OS_OS_Meridian_2_Stations	<p>1) RUN SQL:</p> <pre>SELECT * FROM ws2_build_od_table('WICID_JTW_2001_Interaction_Data_Districts', 'nearest_station_district_centroid_in_districts_st_cent', 'JTW_Train_2001', 27700, 'JTW_Interaction_Districts_2001_OD_OS_Meridian_2_Stations')</pre> <p>The result is a table containing the origin and destination districts, with the respective stations to use as the corresponding origin and destination stations.</p> <p>Attributes include:</p> <p>“id” – serial type (unique), generated by function ws2_build_od_table</p> <p>“Origin_Feature_Name” – name of OS Meridian 2 Rail Station nearest to centroid of Origin district centroid</p> <p>“Origin_Zone_Name” – name of origin district</p> <p>“Origin_Zone_Code” – unique code of origin district</p> <p>“Origin_Feature_To_Zone_Centroid_Distance” – distance between “Origin_Feature_geom” and “Origin_Zone_geom” centroid</p> <p>“Destination_Feature_Name” – name of OS Meridian 2 Rail Station nearest to centroid of Destination district centroid</p> <p>“Destination_Zone_Name” – name of destination district</p> <p>“Destination_Zone_Code” – unique code of destination district</p> <p>“Destination_Feature_To_Zone_Centroid_Distance” – distance between “Destination_Feature_geom” and “Destination_Zone_geom” centroid</p> <p>“Origin_Feature_geom” – geometry of OS Meridian 2 Rail Station in origin district</p> <p>“Origin_Zone_geom” – centroid geometry of origin district</p> <p>“Origin_Feature_To_Zone_Centroid_geom” – line geometry linking OS Meridian 2 Railway Station and origin district centroid</p> <p>“Destination_Feature_geom” – geometry of OS Meridian 2 Rail Station in destination district</p> <p>“Destination_Zone_geom” – centroid of destination district</p> <p>“Destination_Feature_To_Zone_Centroid_geom” – line geometry linking OS Meridian 2 Railway Station and destination district centroid</p> <p>“JTW_Count” – number of passengers travelling between “Origin_Zone_Code”/“Origin_Zone_Name” starting at station “Origin_Feature_name” and “Destination_Zone_Code”/“Destination_Zone_name” finishing at station “Destination_Feature_name”</p>

	<p>e.g.</p> <p>"Cannon Street (London)"; "Cannon Street (London)"; "City of London"; "City of London"; "00AA"; "00AA"; 414.318684017; 414.318684017; "Cannon Street (London)"; "Cannon Street (London)"; "City of London"; "City of London"; "00AA"; "00AA"; 414.318684017; "0101000020346C00000000000000F440204100000000E8120641"; "0101000020346C0000086BFA026E03F2041CD942C801D1F0641"; "0102000020346C000002000000086BFA026E03F2041CD942C801D1F064100000000F440204100000000E8120641"; "0102000020346C000002000000086BFA026E03F2041CD942C801D1F064100000000F440204100000000E8120641"; "0101000020346C00000000000000F440204100000000E8120641"; "0101000020346C0000086BFA026E03F2041CD942C801D1F0641"; "0102000020346C000002000000086BFA026E03F2041CD942C801D1F064100000000F440204100000000E8120641"; 53</p>
<p>ONS 2001 boundaries:</p> <p>LSOA_2001_EW_BFC LSOA_2001_EW_BFE LSOA_2001_EW_BGC LSOA_2001_EW_BGE MSOA_2001_EW_BFC MSOA_2001_EW_BFE MSOA_2001_EW_BGC MSOA_2001_EW_BGE OA_2001_EW_BFC OA_2001_EW_BGE OA_2001_EW_BGC</p> <p>ONS 2011 boundaries:</p> <p>LSOA_2011_EW_BFC LSOA_2011_EW_BFE LSOA_2011_EW_BGC LSOA_2011_EW_PWC MSOA_2011_EW_BFC MSOA_2011_EW_BFE MSOA_2011_EW_BGC MSOA_2011_EW_PWC OA_2011_EW_BFC OA_2011_EW_BFE OA_2011_EW_BGC OA_2011_EW_PWC</p>	<p>Each of the 2001, and 2011 census output area boundaries were supplied as ESRI shapefiles.</p> <p>Each file was uploaded to the census database using the PostGIS SHP and DBF Loader plugin.</p> <p>Prefix meaning:</p> <p>LSOA = lower layer output area MSOA = middle layer output area OA = output area</p> <p>Suffix meaning:</p> <p>_BFC = full resolution, clipped to coastline _BFE = full resolution, clipped to extent of realm _BGC = generalised (20m), clipped to coastline _BGE = generalised (20m), clipped to extent of realm _PWC = population weighted centroid</p> <p>No processing was performed on this data prior to it being uploaded to the database.</p>
ONS 2001 boundary centroids:	Each centroid table for the ONS 2001 and 2011 boundaries was

<p>LSOA_2001_EW_BFC_ST_Centroid LSOA_2001_EW_BFE_ST_Centroid LSOA_2001_EW_BGC_ST_Centroid LSOA_2001_EW_BGE_ST_Centroid MSOA_2001_EW_BFC_ST_Centroid MSOA_2001_EW_BFE_ST_Centroid MSOA_2001_EW_BGC_ST_Centroid MSOA_2001_EW_BGE_ST_Centroid OA_2001_EW_BFC_ST_Centroid OA_2001_EW_BGE_ST_Centroid OA_2001_EW_BGC_ST_Centroid</p> <p>ONS 2011 boundary centroids:</p> <p>LSOA_2011_EW_BFC_ST_Centroid LSOA_2011_EW_BFE_ST_Centroid LSOA_2011_EW_BGC_ST_Centroid MSOA_2011_EW_BFC_ST_Centroid MSOA_2011_EW_BFE_ST_Centroid MSOA_2011_EW_BGC_ST_Centroid OA_2011_EW_BFC_ST_Centroid OA_2011_EW_BFE_ST_Centroid OA_2011_EW_BGC_ST_Centroid</p>	<p>calculated using the ST_Centroid function of PostGIS e.g.</p> <pre>DROP TABLE IF EXISTS "LSOA_2001_EW_BFC_ST_Centroids"; CREATE TABLE "LSOA_2001_EW_BFC_ST_Centroids" AS SELECT gid, lsoa01cd, lsoa01nm, ST_Centroid(geom) FROM "LSOA_2001_EW_BFC"; ALTER TABLE "LSOA_2001_EW_BFC_ST_Centroids" ADD CONSTRAINT enforce_dims_geom CHECK (st_ndims(geom) = 2); ALTER TABLE "LSOA_2001_EW_BFC_ST_Centroids" ADD CONSTRAINT enforce_geotype_geom CHECK (geometrytype(geom) = 'POINT'::text OR geom IS NULL); ALTER TABLE "LSOA_2001_EW_BFC_ST_Centroids" ADD CONSTRAINT enforce_srid_geom CHECK (st_srid(geom) = 27700);</pre> <p>This SQL was executed for each boundary set.</p>
<p>OA01_OA11_EW_LU MSOA01_MSOA11_EW_LUv2 LSOA01_LSOA11_EW_LUv2 OA11_LSOA11_MSOA11_LAD11_EW_LUv2 OA11_RGN11_EN_LU OA11_CTY11_EN_LU OA11_CTRY11_EW_LU</p>	<p>The ONS lookup tables were supplied as individual csv files for each output level (output area, lower layer and middle layer)</p> <p>FME Workbench was used to upload this data. A CSV feature type reader was used to read each file, whilst a Postgres feature type writer was used to write each input file to a separate table.</p>

Figure 1 - itrc_census database data processing steps. Contains CENSUS Journey To Work Data for 2001

Itrc_economics

Table prefix/table name	Process Description
uk_investing_sectors	<p>1) RUN the following SQL to create a table “uk_investing_sectors”.</p> <pre>CREATE TABLE uk_investing_sectors (uk_investing_sector_id double precision, uk_investing_sector_name character varying(50)) WITH (oids = FALSE);</pre> <p>ALTER TABLE uk_investing_sectors OWNER TO postgres;</p> <p>2) RUN the following SQL to populate the “uk_investing_sectors” table (values missing here, contact David.Alderson@ncl.ac.uk or Chris Thoung (ct@camecon.com) for further details)</p> <pre>INSERT INTO uk_investing_sectors (uk_investing_sector_id, uk_investing_sector_name) VALUES (1, "1 Agriculture etc");</pre> <p>...</p> <pre>INSERT INTO uk_investing_sectors (uk_investing_sector_id, uk_investing_sector_name) VALUES (28, "28 Unallocated");</pre> <p>NOTE: This table stores the uk investing sectors related to the economics CDAM model produced by Cambridge Econometrics, related to the “UK Investment” variable.</p>
sectors	<p>1) RUN the following SQL to create a table “sectors” (values missing here, contact David.Alderson@ncl.ac.uk or Chris Thoung (ct@camecon.com) for further details)</p> <pre>CREATE TABLE sectors (sector_id serial PRIMARY KEY, sector_name character varying (30)) WITH (oids = FALSE);</pre> <p>ALTER TABLE sectors OWNER TO postgres;</p> <p>2) RUN the following SQL to populate “sectors” table.</p> <pre>INSERT INTO sectors (sector_name) VALUES ("Agriculture etc");</pre> <p>...</p> <pre>INSERT INTO sectors(sector_name) VALUES ("Unallocated");</pre> <p>NOTE: This table stores the sectors related to the economics CDAM model produced by Cambridge Econometrics, relates to the following variables:</p> <ul style="list-style-type: none"> • UK Industry Exports • UK Industry Imports • UK Industry Expenditure • UK Industry Output Prices • Employment • GVA
regional_investing_sector	<p>1) RUN the following SQL to create a table “regional_investing_sector” (values missing here, contact David.Alderson@ncl.ac.uk or Chris Thoung (ct@camecon.com) for</p>

	<p>further details)</p> <p>CREATE TABLE regional_investing_sector(regional_investing_sector double precision, regional_investing_sector_name character varying(50)) WITH (oids = FALSE);</p> <p>ALTER TABLE regional_investing_sector OWNER TO postgres;</p> <p>2) RUN the following SQL to populate “regional_investing_sector”.</p> <p>INSERT INTO regional_investing_sector(1, "1 Agriculture etc") ... INSERT INTO regional_investing_sector(16, "16 Unallocated")</p>
Fuel_users	<p>1) RUN the following SQL to create a table “fuel_users” (values missing here, contact David.Alderson@ncl.ac.uk or Chris Thoun (ct@camecon.com) for further details)</p> <p>CREATE TABLE fuel_users (fuel_user_id serial PRIMARY KEY, fuel_user_name character varying (50)) WITH (oids=false);</p> <p>ALTER TABLE fuel_users OWNER TO postgres;</p> <p>2) RUN the following SQL to populate “fuel_users”</p> <p>INSERT INTO fuel_users (fuel_user_name) VALUES (“1 Power generation”) ... INSERT INTO fuel_users(fuel_user_name) VALUES ("25 Miscellaneous")</p> <p>NOTE: This table stores the fuel user categories related to the economics CDAM model produced by Cambridge Econometrics, related to the following variables:</p> <ul style="list-style-type: none"> • UK CO₂ Emissions • UK GHG Emissions • Energy
consumption_categories	<p>1) RUN the following SQL to create a table “consumption_categories” (values missing here, contact David.Alderson@ncl.ac.uk or Chris Thoun (ct@camecon.com) for further details)</p> <p>CREATE TABLE consumption_categories(consumption_category_id double precision, consumption_category_name character varying(50)) WITH (oids=false);</p> <p>ALTER TABLE consumption_categories OWNER TO postgres;</p> <p>2) RUN the following SQL to populate “consumption_categories”</p> <p>INSERT INTO consumption_categories(1, "1 Food") ...</p>

	<p>INSERT INTO consumption_categories(52, "52 Unallocated")</p> <p>NOTE: This tables stores the consumption categories related to the economics CDAM model produced by Cambridge Econometrics, relates to the following variables:</p> <ul style="list-style-type: none"> • UK Household Expenditure
government_office_regions	<p>1) Download English Government Office Regions ESRI Shapefile from Share-Geo or UKBORDERS</p> <p>2) Download the Scottish Country Outline ESRI Shapefile from UKBORDERS</p> <p>3) Download the Wales Country Outline ESRI Shapefile from UKBORDERS</p> <p>4) Open ESRI ArcGIS ArcMap</p> <p>5) Load the English Government Office Region, Scottish and Wales Country Outlines into ArcMap</p> <p>6) Open ArcToolbox, and select:</p> <p>Data Management Tools -> General -> Merge</p> <p>7) Select each loaded dataset, as an input to the Merge Tool. Specify an output path, and name of english_gor_merge_wales_scotland_border.shp.</p> <p>8) Execute the ESRI ArcToolbox Merge Tool.</p> <p>The result of the operation is an ESRI shapefile containing the English Government Office Regions, merged with the Scottish and Welsh Country Outlines.</p> <p>9) Using the PostGIS Shp and DBF Loader plugin, upload the government_office_regions shapefile to the itrc_economics database.</p> <p>NOTE: The results produced by the Cambridge Econometrics Economic CDAM model are aggregated at the Government Office Region Level.</p>

Figure 2 - processing steps undertaken to create tables to store economics CDAM constants

Itirc_population

Table prefix/table name	Process Description
districts	<p>1) Download English Government Office Regions ESRI Shapefile from Share-Geo or UKBORDERS</p> <p>2) Using the PostGIS Shp and DBF Loader, load the districts table in to the itirc_population database</p> <p>NOTE: The results of running the demographics CDAM is disaggregated at the district level.</p> <p>The resultant table should contain 354 records (England and Wales only)</p>
districts_gor_union	<p>1) RUN the following SQL to create a table containing all the districts (354 records for England and Wales only) and also the government office regions (11)</p> <pre>CREATE TABLE districts_gor_union AS SELECT code, "name", geom FROM districts UNION ALL SELECT "name" as code, "name" as "name", geom FROM government_office_regions</pre> <p>ALTER TABLE districts_gor_union OWNER TO postgres;</p> <pre>CREATE INDEX districts_gor_union_geom ON districts_gor_union USING gist(geom)</pre>
government_office_regions	<p>1) Download English Government Office Regions ESRI Shapefile from Share-Geo or UKBORDERS</p> <p>2) Download the Scottish Country Outline ESRI Shapefile from UKBORDERS</p> <p>3) Download the Wales Country Outline ESRI Shapefile from UKBORDERS</p> <p>4) Open ESRI ArcGIS ArcMap</p> <p>5) Load the English Government Office Region, Scottish and Wales Country Outlines into ArcMap</p> <p>6) Open ArcToolbox, and select:</p> <p>Data Management Tools -> General -> Merge</p> <p>7) Select each loaded dataset, as an input to the Merge Tool. Specify an output path, and name of english_gor_merge_wales_scotland_border.shp.</p> <p>8) Execute the ESRI ArcToolbox Merge Tool.</p>

	<p>The result of the operation is an ESRI shapefile containing the English Government Office Regions, merged with the Scottish and Welsh Country Outlines.</p> <p>9) Using the PostGIS Shp and DBF Loader plugin, upload the government_office_regions shapefile to the itrc_population database.</p> <p>NOTE: Some of the results produced by the Demographics CDAM model are aggregated at the Government Office Region Level.</p>
--	---

Figure 3 - processing steps undertaken to create tables to store demographics CDAM constants

Itrc_spatial_energy

Table prefix/table name	Process Description
AEP_CWIntakesAtNuclearAndOtherLargeUKPowerStations1000MWe	<p>The cold water intake at nuclear and other large (1000Mwe+) UK power stations data was supplied in Appendix A, as part of the Cooling Water Options for the New Generation of Nuclear Power Stations in the UK report.</p> <p>cdn.environment-agency.gov.uk/scho0610bsot-e-e.pdf</p> <p>The data was copied to a csv file, where the column names were mapped to something compatible with tables in a relational database.</p> <p>Name = Name Fuel or Type = FuelOrType Current Status = CurrentStatus Installed Capacity Mwe = InstalledCapacity Maximum CW flow m^3s^{-1} = MaxFlow Cooling Water Source = CWSource Intake Position = IntakePos Moving Screens = Screen Fish Protection or Return = FPR</p> <p>Additional latitude and longitude columns were added to the csv file, to store the coordinate information for each power station site.</p> <p>The corresponding location of each power station was found by searching the internet in the following manner:</p> <ol style="list-style-type: none"> The company website, using the “Name” parameter. If a location, or postcode was defined then this was used to refine the location further. If no useful location information could be found from the company website, a simple search using the “Name” was performed. If a response was returned from Wikipedia, this was initially followed e.g. <ul style="list-style-type: none"> http://en.wikipedia.org/wiki/Aberthaw_power_station - result from “Name” search for “Aberthaw” http://toolserver.org/~geohack/geohack.php?pagename=Aberthaw_power_stations&params=51.387312_N_-3.404866_E_type:landmark – GeoHack location link supplied with link above, giving WGS84 and UTM coordinates e.g.

GeoHack - Aberthaw power stations



Figure 4 - GeoHack coordinate information available via Wikipedia, for Aberthaw Power Station

- The decimal values highlighted in **RED**, in the above image denote those used as reference to the location of a power station, and extracted to the csv document

- c) Alternatively, if the first two options did not yield a location, Google Maps was utilised to locate the power station in question.

NOTE: The accuracy of the location of these power stations is therefore questionable, as a result of the available data sources giving reference to the location of a power station.

5) This same operation was performed for all power stations listed in the original Appendix A document.

AEP_ClosuresPowerStationNuclearAGR

AEP_ClosuresPowerStationNuclearMagnox

AEP_ClosuresPowerStationsCoal

AEP_ClosuresPowerStationsOil

Data converted to separate .csv files from original pdf, found at:

www.aepuk.com/.../Power%20Station%20Closures%202025.pdf

Geographic coordinates of each station captured or copied using method described for DECC_OperationalPowerStationsMay2011

CSV data loaded as separate files into ESRI ArcGIS ArcMap, with the columns of Easting and Northing within each CSV file used as the x and y coordinates within ArcMap.

Each subsequent file was then exported to a separate shapefile e.g.

AEP_ClosuresPowerStationNuclearAGR.shp
AEP_ClosuresPowerStationNuclearMagnox.shp
AEP_ClosuresPowerStationsCoal.shp
AEP_ClosuresPowerStationsOil.shp

These files were then uploaded to the ITRC energy database using the PostGIS Shp and DBF Loader plugin for PostgreSQL / PostGIS.

AEP_PlannedNewBuildPowerStationsCoal

AEP_PlannedNewBuildPowerStationsGas

Data converted to separate .csv files from original pdf, for coal fuel and gas types only, found at:

www.berr.gov.uk/files/file49436.xls

Geographic coordinates of each coal station captured or copied

	<p>using the method described for DECC_OperationalPowerStationsMay2011</p> <p>Excel spreadsheet data loaded in to ESRI ArcGIS ArcMap, with the columns of Easting and Northing used for coordinates.</p> <p>Two shapefiles were then created as a result of exporting the data from ArcMap e.g.</p> <p>PlannedNewBuildPowerStationsCoal.shp PlannedNewBuildPowerStationsGas.shp</p> <p>These files were then uploaded to the ITRC energy database using the PostGIS Shp and DBF Loader plugin for PostgreSQL / PostGIS.</p>
DECC_OperationalPowerStationsMay2011	<p>1) DUKES5_11.csv document created, containing reference to the data of Power Stations in the United Kingdom (operational as of end of May 2011).</p> <p>Attributes include:</p> <ul style="list-style-type: none"> • Company Name • Station Name • Fuel • Installed Capacity (MW) • Year of commission or year generation began • Location (one of Scotland, Wales, Northern Ireland, England) <p>Original raw data found in table 5.11 at http://www.decc.gov.uk/assets/decc/11/stats/publications/dukes/2307-dukes-2011-chapter-5-electricity.pdf</p> <p>2) The column names were mapped from the original raw data (left) to the following names (right):</p> <p>Company Name = co_name Station Name = station_name Fuel = fuel_type Installed Capacity = capacity Year of commission or year generation began = com_year Location was ignored</p> <p>3) Columns were added to the csv document for latitude and longitude.</p> <p>4) The corresponding location of each power station was found by searching the internet in the following manner:</p> <p>d) The company website, using the “Company Name” parameter. If a location, or postcode was defined then this was used to define the location further.</p>

- e) If no useful location information could be found from the company website, a simple search using the “Station Name” was performed. If a response was returned from Wikipedia, this was initially followed e.g.
- http://en.wikipedia.org/wiki/Little_Barford_Power_Station - result from “Station Name” search for “Little Barford”
 - http://toolserver.org/~geohack/geohack.php?pagename=Little_Barford_Power_Station¶ms=52_12_16_N_0_16_8_W_region:GB_type:landmark – GeoHack location link supplied with link above, giving WGS84 and UTM coordinates e.g.

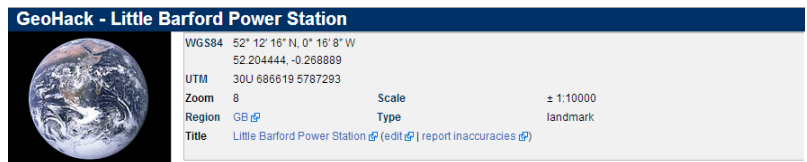


Figure 5 - GeoHack coordinate information available via Wikipedia, for Little Barford Power Station

- The decimal values highlighted in **RED**, in the above image denote those used as reference to the location of a power station, and extracted to the DUKES5_11.csv document.

- f) Alternatively, if the first two options did not yield a location, Google Maps was utilised to locate the power station in question.

NOTE: The accuracy of the location of these power stations is therefore questionable, as a result of the available data sources giving reference to the location of a power station.

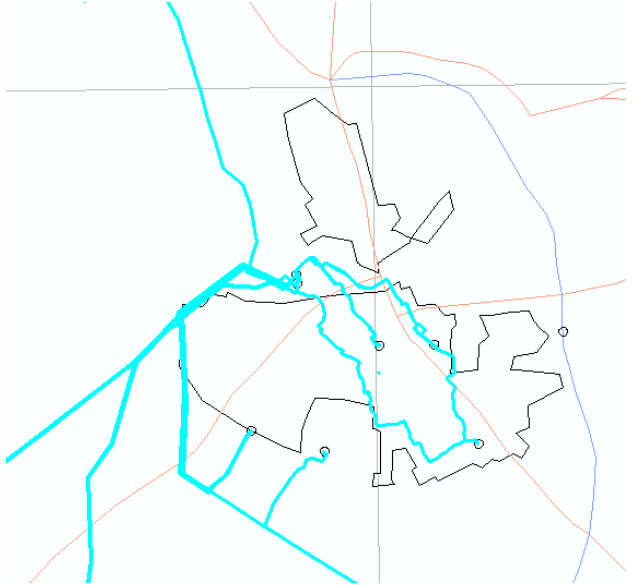
5) This same operation was performed for all power stations listed in the original DUKES document.

6) The subsequent .csv file was loaded in to ESRI ArcMap, and converted to an ESRI Shapefile, along side all afore-mentioned attributes.

7) The shapefile was converted to British National Grid coordinates, using the ArcToolbox tool:

Data Management->Projections and Transformations->Feature->Project

8) The shapefile was subsequently uploaded to the itrc_spatial_energy database using the PostGIS Shp and DBF Loader.

DECC_ListOfPotentialElectricityGeneratingCapacity_UK	
<p>ENW_LTDS_2010_132kV_geographic_expanded_rev2_7</p> <p>ENW_LTDS_2010_33kV_Geographic_Lakes_3</p> <p>ENW_LTDS_2010_33kV_Geographic_Lancs_3</p> <p>ENW_LTDS_2010_33kV_Geographic_South_3</p>	<p>NOTE: Initial Electricity North West data supplied as a series of 3 separate pdf documents, containing no geographic coordinate information. These steps detail how each separate pdf was converted to a georeferenced ESRI Shapefile:</p> <p>PDF -> DXF</p> <ol style="list-style-type: none"> 1) Open Adobe Illustrator. 2) Next select to filter only "Files of type: Adobe PDF (*.AI, *.AIT, *.PDF)" 3) Select the chosen PDF file e.g. LTDS_2010_33kV_Geographic_Lakes.pdf 4) Once the file is open, click File->Export, and select the "AutoCAD Interchange File (*.DXF)" format. 5) The result should be a .dxf file of the pdf input <p>DXF -> Non-georeferenced ESRI Shapefile (lines, polygons, annotation)</p> <ol style="list-style-type: none"> 1) Open ESRI ArcGIS ArcMAP 2) Click the Add Data button, and for a single dxf document, only load the "Polygon", "Polyline" and "Annotation" feature types. Double click the main .dxf file to receive a list of feature types, including Polygon and Polyline 3) From the Polyline data loaded, using "Select by Attributes", select only those with "Color = 3". This should select only those features that are actually ENW distribution lines from that file e.g. <div data-bbox="783 1384 1417 1966" data-label="Figure">  </div> <p>Figure 6 - distribution lines highlighted in cyan, as polylines in dxf</p> <ol style="list-style-type: none"> 4) Export this data to a separate ESRI Shapefile, labelled accordingly.

5) From the **Polygon** data loaded, again using “Select by Attributes”, select only those with “LineWt” = 5 AND “Color = 7”. This should select all the polygons representing substations within the network e.g.

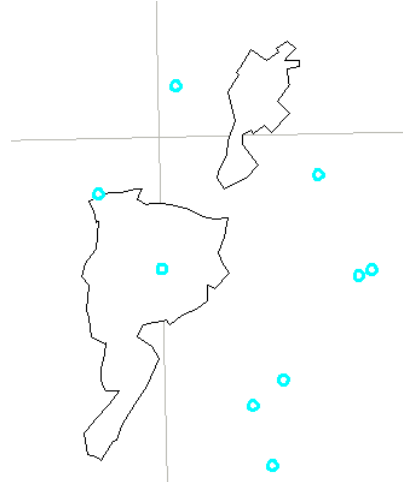


Figure 7 - substations highlighted in cyan, as polygons within dxf

NOTE: Be careful that this selection has only selected the features of interest, and that all other features remain unselected.

6) Export this data to a separate ESRI Shapefile, labelled accordingly

7) From the **Annotation** data loaded, again using “Select by Attributes”, select only those features with “Color = 7” e.g.

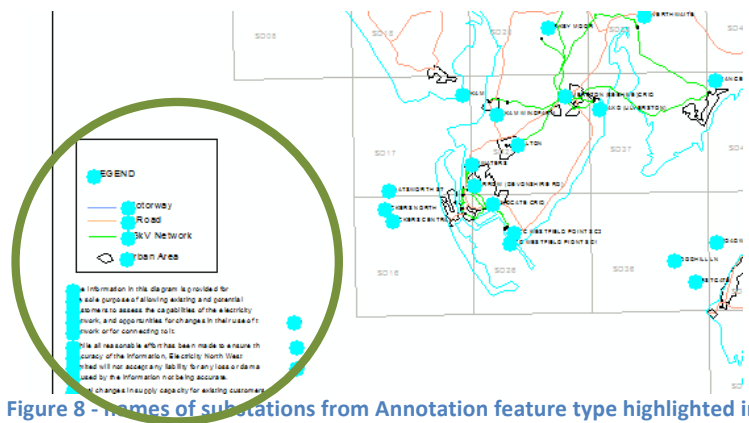


Figure 8 - names of substations from Annotation feature type highlighted in cyan

NOTE: Be careful to subsequently inspect those selected features and remove any selected features that are not features of interest, or any duplicates. For example those selected within the legend (see **GREEN** circle in figure.6)

Converting Substation Polygons to Points:

- 1) Open ESRI ArcGIS ArcMap
- 2) Load the Substations shapefile generated as part of the

	<p>previous steps into ArcMap</p> <p>3) Open ArcToolbox and select:</p> <p>Data Management Tools -> Features -> Feature To Point</p> <p>4) Select the ESRI shapefile loaded in step 2, as the input to the tool, and specify an appropriate output path and name</p> <p>5) Execute the tool for each Substation shapefile generated from the raw .dxf. The result should be a separate ESRI point shapefile for each substation shapefile generated earlier.</p> <p>The result of these various steps is a set of ESRI shapefiles that can then be used in combination, alongside other data sources to begin to georeference the network.</p> <p>Add X/Y coordinates to each substation centroid file</p> <p>1) Open ESRI ArcGIS ArcMap</p> <p>2) For each substation centroid file generated in the previous steps, a set of local, non-geographic coordinates need to be added to the attribute table of each in order to then be able to adjust these points to their true or near true geographic coordinates.</p> <p>3) Open the attribute table and select "Add Field", called "X" with type float</p> <p>4) Select "Add Field" called "Y", with type float</p> <p>5) Right click the newly-created "X" field and select "Calculate Geometry", selecting the "X Coordinate of geometry"</p> <p>6) Right click the newly-created "Y" field and select "Calculate Geometry", selecting the "Y Coordinate of geometry"</p> <p>Georeferencing the data</p> <p>1) Open ESRI ArcGIS ArcMap</p> <p>2) Load a single set of non-georeferenced substations, distribution lines and annotation points, generated as part of the previous steps</p> <p>3) Load the Ordnance Survey 1:50k Gazetteer Shapefile.</p> <p>The coordinates of the substations are retrieved by matching the name of the substation against a record in the OS 1:50k Gazetteer with the same name. For example:</p> <p>Greenfield (ID, from_x, from_y, to_x, to_y):</p> <p>Where from_x and from_y are local coordinates relative to the pdf document within which the data resides, whereas to_x and to_y are real world (British National Grid) coordinates.</p> <p>15 567.119728 751.67876 399500 404500</p> <p>4) The process of matching each substation name to a</p>
--	--

	<p>corresponding name within the Gazetteer should be continued for as many matches as possible. Each link file created is not only used to transform the substations, but also the distribution lines as well. It is sensible to save each of these links for each pdf within a separate link file (http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Working_with_link_files_and_control_point_files – Creating displacement links as a text file)</p> <p>NOTE: this method is effectively using the OS 1:50k Gazetteer to help georeference the ENW data. The coordinates taken from the gazetteer refer to the geographic centre of a place, and therefore by using these coordinates we are assuming that the substation of the same name is also co-located at the same centre point – this is obviously not always true, but this method represented the best and quickest way to extract a spatial network from the input pdf.</p> <p>5) Having created the respective link files, linking local pdf coordinates to real-world geographic coordinates, the raw data can be adjusted. For each pdf file, 3 files will need to be adjusted; the substation points, the distribution lines, and finally the annotation points. For each file a separate edit session will need to be started. Using the Spatial Adjustment Toolbar in ArcMap:</p> <ul style="list-style-type: none"> - Set the Adjust Data to be “all features” - Set the adjustment method to be affine transform <p>It is then possible to open the appropriate Links file created earlier and perform the adjustment, by selecting Adjust.</p> <p>The resultant output, georeferenced shapefiles, were then uploaded to the itrc energy database using the plugin PostGIS Shp and DBF Loader.</p>
<p>EHCS_2year_weight_2007 EHCS_actual_costs_0203_2003 – EHCS_actual_costs_06_and_07_2007 EHCS_adapt_2003 – EHCS_adapt_2007 EHCS_adaptation_2003 – EHCS_adaptation_2005 EHCS_amenity_2003 – EHCS_amenity_2007 EHCS_around_2003 – EHCS_around_2007 EHCS_around_physical_2006 EHCS_attitudes_2003 – EHCS_attitudes_2007 EHCS_benefits_2003 – EHCS_benefits_2007 EHCS_chimney_2003 – EHCS_chimney_2007 EHCS_commac_2003 – EHCS_commac_2007 EHCS_common_2003 – EHCS_common_2007 EHCS_contact_2003 – EHCS_contact_2007 EHCS_damp_2003 – EHCS_damp_2007 EHCS_dampmould_0506_2006 EHCS_damppc_2003 – EHCS_damppc_2007</p>	<p>The English House Condition Survey data was supplied as a series of tab delimited text files.</p> <p>FME Workbench was used to convert this data to Postgres tables within the database. A CSV Feature Type reader was used to read each dataset, and then a Postgres Feature Type writer was used to write the data to the database.</p> <p>In general, a separate FME workspace was created for each different year’s worth of data supplied.</p> <p>The type and year of data was used to build each table name, and therefore helps to distinguish each source directly.</p>

EHCS_decent_2003
EHCS_deregionalised_costs_0203_2003 –
EHCS_deregionalised_costs_0607_2007
EHCS_dimensions_0203_2003 –
EHCS_dimensions_0607_2007
EHCS_dims_0203_2003 –
EHCS_dims_0607_2007
EHCS_disability_2003 – EHCS_disability_2007
EHCS_doors_2003 – EHCS_doors_2007
EHCS_dormers_2003 - EHCS_dormers_2007
EHCS_elevate_2003 – EHCS_elevate_2007
EHCS_employment_2003 –
EHCS_employment_2007
EHCS_energy_performance_0506_2006 –
EHCS_energy_performance_0607_2007
EHCS_energydims_0203_2003 –
EHCS_energydims_0607_2007
EHCS_equivalised_income_0203_2003 –
EHCS_equivalised_income_0607_2007
EHCS_firstimp_2003 – EHCS_firstimp_2007
EHCS_firstimp_physical_2003 –
EHCS_firstimp_physical_2007
EHCS_fitness_2003 – EHCS_firstness_2007
EHCS_flatdets_2003 – EHCS_flatdets_2007
EHCS_fuel_poverty_dataset_2003 –
EHCS_fuel_poverty_dataset_2007
EHCS_hhsrs_2006 – EHCS_hhsrs_2007
EHCS_hmodata_2003 – EHCS_hmodata_2007
EHCS_hq_2003 – EHCS_hq_2007
EHCS_income_2003 – EHCS_income_2007
EHCS_intenure_2003 – EHCS_intenure_2007
EHCS_interior_2003 – EHCS_interior_2007
EHCS_introoms_2003 – EHCS_introoms_2007
EHCS_modernisation_2003
EHCS_mvs_2003 – EHCS_mvs_2007
EHCS_numflats_2003 – EHCS_numflats_2007
EHCS_people_2003 – EHCS_people_2007
EHCS_plotlvl_2003 – EHCS_plotlvl_2007
EHCS_plotwall_2003 – EHCS_plotwall_2007
EHCS_regionalised_comp_rcm_0203_2003
EHCS_repair_costs_0203_2003 -
EHCS_repair_costs_0405_2005
EHCS_repairs_2003 – EHCS_repairs_2007
EHCS_rootcov_2003 – EHCS_rootcov_2007
EHCS_rooffeat_2003 – EHCS_rooffeat_2007
EHCS_roofstru_2003 – EHCS_roofstru_2007
EHCS_rooms_2003 – EHCS_rooms_2007
EHCS_services_2003 – EHCS_services_2007
EHCS_shared_2003 – EHCS_shared_2007
EHCS_standardised_costs_0203_2003 –
EHCS_standardised_costs_0607_2007
EHCS_structure_2003 – EHCS_structure_2007

<p>EHCS_unfit_2003 EHCS_vacant_2003 – EHCS_vacant_2007 EHCS_wallfin_2003 – EHCS_wallfin_2007 EHCS_wallstru_2003 – EHCS_wallstru_2007 EHCS_windows_2003 – EHCS_windows_2007 EHCS_workdone_2003 – EHCS_workdone_2007</p>	
<p>EPS_AvAnDomGasBillByHomeNonHomeSupplier_CashTerms EPS_AvAnDomGasBillByHomeNonHomeSupplier_RealTerms EPS_AvAnnDomElecBillsByHomeNonHomeSupplier_E7_CashTerms EPS_AvAnnDomElecBillsByHomeNonHomeSupplier_E7_RealTerms EPS_AvAnnDomElecBillsByHomeNonHomeSupplier_ST_CashTerms EPS_AvAnnDomElecBillsByHomeNonHomeSupplier_ST_RealTerms EPS_AvAnnDomElecBillsSelectedTownCityUKAvUnitCosts_1998 - EPS_AvAnnDomElecBillsSelectedTownCityUKAvUnitCosts_2010 EPS_AvAnnDomElecBillsForUKCountries_E7_CashTerms EPS_AvAnnDomElecBillsForUKCountries_E7_RealTerms EPS_AvAnnDomElecBillsForUKCountries_ST_CashTerms EPS_AvAnnDomElecBillsForUKCountries_ST_RealTerms EPS_AvAnnDomGasBillSelectedTownCityUKAvUnitCosts_1998 - EPS_AvAnnDomGasBillSelectedTownCityUKAvUnitCosts_2010 EPS_AvAnnDomGasBillUKCountries_CashTerms EPS_AvAnnDomGasBillUKCountries_RealTerms EPS_AvExpWeekFuelPerConsumeHouseHoldUK EPS_AvVariableUnitCostFixedCostElecSelectedTownCity_E7_2010 EPS_AvVariableUnitCostFixedCostElecSelectedTownCity_ST_2010 EPS_AvVariableUnitCostFixedCostGasSelectedTownCityGB_2010 EPS_TotalHouseholdExpEnergyUK_2006Price EPS_TotalHouseholdExpEnergyUK_CurrentPrice</p>	<p>Energy Price Statistics data was supplied as a series of Microsoft Excel spreadsheets.</p> <p>The data was divided into the following categories once downloaded:</p> <ul style="list-style-type: none"> • Average Annual Domestic Electricity Bills By Home / Non-home supplier • Average Annual Domestic Electricity Bills for selected towns and cities uk average unit costs • Average Annual Domestic Electricity Bills for UK Countries • Average Annual Domestic Gas Bills By Home and Non Home Supplier • Average Annual Domestic Gas Bills For Selected towns selected and cities uk average unit costs • Average Annual Domestic Gas Bills for UK Countries • Average Expenditure each week on fuel per consuming household in the uk • Average Variable Unit Costs and Fixed Costs for Electricity in 2010 for selected towns and cities • Average Variable Unit Costs and Fixed Costs for Gas in 2010 For Selected Towns and Cities Great Britain • Total Household Expenditure on Energy in the UK <p>A separate FME Workbench workspace was created for each type of data. A Microsoft Excel feature type reader was used to read the data in to FME, and a Postgres feature type writer was used to write the data to the database.</p>
<p>Os_electricitysubstations_intersect_nat_floodzone2_v3_8 Os_electricitysubstations_intersect_nat_floodzone3_v3_8 Os_energyproductionsites_intersect_nat_floodzone2_v3_8</p>	<p>Each table was created by creating an intersection of the EA flood zone shapefiles with the original OS shapefiles.</p> <p>Within ESRI ArcGIS ArcMap, each OS shapefile was loaded alongside each EA floodzone shapefile. Using the operation “Select by Location”, selecting features from the OS feature in</p>

<p>Os_energyproductionsites_intersect_nat_floodzone3_v3_8</p> <p>Os_refineries_intersect_nat_floodzone2_v3_8</p> <p>Os_refineries_intersect_nat_floodzone3_v3_8</p> <p>Os_telco_masts_int_nat_floodzone2_v3_8</p> <p>Os_telco_masts_int_nat_floodzone3_v3_8</p>	<p>question as the “Target Layer” and selecting the respective EA floodzone shapefile as the “Source Layer”, the OS features were intersected against the EA floodzones.</p> <p>Each resultant set of intersecting features was then exported to a separate shapefile, based on the input OS feature type and floodzone selected for intersection.</p> <p>Each shapefile was uploaded using the PostGIS SHP and DBF Loader plugin</p>
<p>OS_ElectricitySubStations</p> <p>OS_EnergyProductionSites</p> <p>OS_Refineries</p> <p>OS_TelcoMasts</p>	<p>The Ordnance Survey, energy-related data, supplied at a national-scale was delivered as a series of ESRI Shapefiles.</p> <p>Each shapefile was uploaded using the PostGIS SHP and DBF Loader plugin</p>
<p>OS_EnergyProductionSites_Stations</p> <p>OS_EnergyProductionSites_Turbines</p> <p>OS_EnergyProductionSites_Wind</p> <p>OS_Refineries_GasAssets</p> <p>OS_Refineries_OilAssets</p> <p>OS_Refineries_OilBoreHoles</p> <p>OS_Refineries_OilGatheringStations</p> <p>OS_Refineries_OilRefineries</p> <p>OS_Refineries_OilWells</p>	<p>The Ordnance Survey Energy Production Site and Refineries data was filtered in to respective tables, based upon the name attribute of the original input Ordnance Survey Energy Production Sites data:</p> <p>Within ESRI ArcGIS ArcMap, the filtering was applied by using the “Select by Attributes” tool. Once the resultant subset of features was selected within ArcMap, they were subsequently exported to a separate ESRI Shapefile.</p> <ul style="list-style-type: none"> • "NAME" LIKE '%Station%' • "NAME" LIKE '%Turbine%' • "NAME" LIKE '%Wind%' • "NAME" LIKE '%Gas%' • "NAME" LIKE '%Oil%' <p>The shapefiles were uploaded to the database using the PostGIS SHP and DBF Loader plugin</p> <p>To filter the Oil Assets further from the original refineries features, a second series of attribute value filters were applied to the oil filter:</p> <ul style="list-style-type: none"> • “NAME” LIKE ‘%Oil Bore Hole%’ • “NAME” LIKE ‘%Oil Gathering%’ • “NAME” LIKE ‘%Oil Gathering%’ • “NAME” LIKE ‘%Oil Refinery%’ • “NAME” LIKE ‘%Oil Well%’
<p>NEDL_132kV</p> <p>NEDL_33kV</p> <p>NEDL_66kV</p> <p>NEDL_Primary_Substations</p>	<p>The NEDL data was supplied as a series of ESRI Shapefiles. Each shapefile was uploaded to the database using the PostGIS SHP and DBF Loader plugin.</p>

<p>WS1_CGEN_DB_BusData WS1_CGEN_DB_CompressorData WS1_CGEN_DB_Connections WS1_CGEN_DB_DaySeason WS1_CGEN_DB_DomLink WS1_CGEN_DB_DomesticGas WS1_CGEN_DB_ElecLoad WS1_CGEN_DB_Fuel WS1_CGEN_DB_GasImports WS1_CGEN_DB_GasLoad WS1_CGEN_DB_GasSupply WS1_CGEN_DB_GenData WS1_CGEN_DB_GenPar WS1_CGEN_DB_ImportLink WS1_CGEN_DB_Importexpand1 WS1_CGEN_DB_Importexpand2 WS1_CGEN_DB_LNG WS1_CGEN_DB_LNGLink WS1_CGEN_DB_LNGexpand WS1_CGEN_DB_LineData WS1_CGEN_DB_Location WS1_CGEN_DB_NewAssets WS1_CGEN_DB_NodeData WS1_CGEN_DB_OptimData WS1_CGEN_DB_PipeData WS1_CGEN_DB_StorageData WS1_CGEN_DB_StorageLocation WS1_CGEN_DB_TPeriods WS1_CGEN_DB_Terminalexpend WS1_CGEN_DB_WindData</p>	<p>The Workstream 1 CGEN database was supplied initially as a Microsoft Access database. The database contained separate tables that were mapped to individual output tables within Postgres</p> <p>FME Workbench was used to upload the data to the database. A Microsoft Access feature type reader was used to read the data, and a Postgres feature type writer used to write the data to the database.</p> <p>This data was initially supplied by Modassar Chaudry, from Cardiff University (chaudrym@cardiff.ac.uk). This data will likely change as the CDAM for energy alters.</p>
<p>WS1_CGEN_DB_StorageData_Name_NationalGrid_Gas_Assets</p>	<p>This table was generated by using the “Name” attribute of the table WS1_CGEN_DB_StorageData and matching, using SQL, this against the “name” attribute of the table “data_national_grid_assets”. This enabled the geometry of the “data_national_grid_assets” table to be extracted and used as the geometry for features within the WS1 CGEN model.</p> <p>This dataset was created in an attempt to find coordinate information for assets and facilities listed within the WS1 CGEN model.</p>
<p>WS1_CGEN_DB_GenData_GenName_AEP_CW_Nuclear_LargePS WS1_CGEN_DB_GenData_GenName_AEP_ClosureCoal WS1_CGEN_DB_GenData_GenName_AEP_ClosureNuclearAGR WS1_CGEN_DB_GenData_GenName_AEP_ClosureNuclearMagnox WS1_CGEN_DB_GenData_GenName_AEP_ClosureOil WS1_CGEN_DB_GenData_GenName_AEP_New</p>	<p>Each table was generated by matching the “GenName” attribute of the WS1_CGEN_DB_GenData table against the respective “name” attribute that is found in the following tables:</p> <ul style="list-style-type: none"> AEP_CWIntakesAtNuclearAndOtherLargeUKPowerStations1000MWe AEP_ClosuresPowerStationNuclearAGR AEP_ClosuresPowerStationNuclearMagnox AEP_ClosuresPowerStationsCoal AEP_ClosuresPowerStationsOil AEP_PlannedNewBuildPowerStationsCoal

<p>Coal</p> <p>WS1_CGEN_DB_GenData_GenName_AEP_New Gas</p> <p>WS1_CGEN_DB_GenData_GenName_AEP_DEC C_PotElec_name</p> <p>WS1_CGEN_DB_GenData_GenName_AEP_Station_Na</p> <p>WS1_CGEN_DB_GenData_GenName_AEP_Geom_Matches</p> <p>WS1_CGEN_DB_GenData_GenName_AEP_Geom_No_Matches</p>	<ul style="list-style-type: none"> AEP_PlannedNewBuildPowerStationsGas <p>These datasets were created in an attempt to find geographic coordinate information for assets and facilities found within the WS1 CGEN model.</p>
<p>nationalgrid_cable</p> <p>nationalgrid_gas_site</p> <p>nationalgrid_gas_pipeline_feeder</p> <p>nationalgrid_line</p> <p>nationalgrid_substation_site</p> <p>nationalgrid_tower</p>	<p>Each of the original National Grid shapefiles were loaded in to the database using the PostGIS SHP and DBF Loader plugin.</p>
<p>Nationalgrid_cable_join_GOR</p> <p>Nationalgrid_gas_pipeline_feeder_join_GOR</p> <p>Nationalgrid_gas_site_join_GOR</p> <p>Nationalgrid_line_join_GOR</p> <p>Nationalgrid_substation_site_join_GOR</p> <p>Nationalgrid_tower_join_GOR</p>	<p>Each table was created by performing a spatial join between the government office region data, and the original National Grid shapefiles (see above).</p> <p>Within ESRI ArcGIS ArcMap, each National Grid shapefile was loaded. By right-clicking the National Grid data and selecting Join -> Join data from another layer based on spatial location, it was possible to join the government office region data.</p> <p>Make sure that the government office region data is the input layer (1), and then ensure that the option “Joining polygon to points” is specified.</p> <p>Finally, Select “it falls inside” to denote that the point data i.e. National Grid data will have the government office region within which it lies added to itself.</p> <p>Each resultant shapefile was loaded in to the database using the PostGIS SHP and DBF Loader plugin.</p>
<p>Nationalgrid_cable_intersect_nat_floodzone2_v3_8_Join_GOR</p> <p>Nationalgrid_cable_intersect_nat_floodzone3_v3_8_Join_GOR</p> <p>Nationalgrid_gas_pipeline_feeder_int_nat_fldz2_v3_8_GOR</p> <p>Nationalgrid_gas_pipeline_feeder_int_nat_fldz3_v3_8_GOR</p> <p>Nationalgrid_line_int_floodzone2_v3_8_Join_GOR</p> <p>Nationalgrid_line_int_floodzone3_v3_8_Join_GOR</p> <p>Nationalgrid_substation_site_int_nat_floodzone2_v3_8_Join_GOR</p> <p>Nationalgrid_substation_site_int_nat_floodzone3_v3_8_Join_GOR</p>	<p>Each table was created by creating an intersection of the EA flood zone shapefiles with the shapefiles generated by joining the government office region data (see above) and the original National Grid shapefiles</p> <p>Within ESRI ArcGIS ArcMap, each OS shapefile was loaded alongside each EA floodzone shapefile. Using the operation “Select by Location”, selecting features from the National Grid feature in question as the “Target Layer” and selecting the respective EA floodzone shapefile as the “Source Layer”, the National Grid features were intersected against the EA floodzones.</p> <p>Each resultant set of intersecting features was then exported to a separate shapefile, based on the input National Grid feature type and floodzone selected for intersection.</p>

e3_v3_8_Join_GOR Nationalgrid_tower_int_nat_floodzone2_v3_8_Join_GOR Nationalgrid_tower_int_nat_floodzone3_v3_8_Join_GOR	Each shapefile was uploaded using the PostGIS SHP and DBF Loader plugin
nationalgrid_jan2012_cable nationalgrid_jan2012_gas_pipe_feeder nationalgrid_jan2012_line nationalgrid_jan2012_substation_site	Each of the January 2012 National Grid shapefile updates were loaded in to the database using the PostGIS SHP and DBF Loader plugin
Nationalgrid_jan2012_gas_agis_singlepart Nationalgrid_tower_singlepart	<p>The January 2012 National Grid shapefiles for gas sites and electricity transmission towers was received as a 'multipart' set of geometries.</p> <p>Within ESRI ArcGIS ArcMap the tool Data Management Tools -> Features -> Multipart to Singlepart was utilised to convert the gas site shapefile and tower shapefile to singlepart shapefiles.</p> <p>The two resultant output shapefiles were loaded in to the database using the PostGIS SHP and DBF Loader plugin</p>
NationalGrid_Derived_Gas_Compressor NationalGrid_Derived_Gas_LNG_Operators NationalGrid_Derived_Gas_Storage NationalGrid_Derived_Gas_Terminals Data_national_grid_assets	<p>Each of these tables were created by extracting information from the names of the input National Grid features, found within the original ESRI Shapefiles, and also that found within the table WS1_CGEN_DB_GenData.</p> <p>Many node or point features within the original data supplied by the National Grid for the gas network were missing. In particular this was found to be the case for smaller gas assets, often found along the length of a gas pipeline, or at either end.</p> <p>The end points of the original gas pipeline shapefile were extracted and the names of these features were compared to those found within the WS1_CGEN energy CDAM to determine the likely "type" of feature.</p>
Subnationaldomestic_elec_2008 Subnationaldomestic_elec_2009 Subnationaldomestic_elec_lloa_2008 Subnationaldomestic_elec_lloa_2009 Subnationaldomestic_gas_2008 Subnationaldomestic_gas_2009 Subnationaldomestic_gas_lloa_2008 Subnationaldomestic_gas_lloa_2009 Socioeconomic_englandwales_lloa_2008 Socioeconomic_englandwales_mloa_2008 Socioeconomic_scotland_igz_2008 Subnationalelectricitygasconsumption_2005 Subnationalelectricitygasconsumption_2006 Subnationalelectricitygasconsumption_2007 Subnationalnondomestic_2005	<p>The original subnational gas and electricity consumption data was supplied as a series of Microsoft Excel spreadsheets, with a separate file being supplied for each year of data, and then for each government office region within England and Wales.</p> <p>Each file, for a particular set of data, for each year, was combined in to a single csv file via the use of Python scripting, and the xlrd and csv Python modules.</p> <p>Each output csv file was then written to the database using FME Workbench. A CSV Feature type reader was used to read the data, with a Postgres feature type writer used to write the data to the database. A PostGIS feature type writer was used where specific geometry for government office regions was available.</p>

Subnationalnondomestic_2006 Subnationalnondomestic_2007	
Socioeconomic_scotland_igz_geom_2008 Socioeconomic_englandwales_llsoa_geom_2008 Socioeconomic_englandwales_mlsa_geom_2008	

Figure 9 - processing steps to create and load data to the energy section of the database

Itrc_spatial_hazards

Table prefix/table name	Process Description
EA_NAFRA_all_Nov2012	<p>EA_NAFRA data was supplied as a series of 85 shapefiles e.g. nf730024.shp, nf740024.shp.</p> <ol style="list-style-type: none"> 1) Open ESRI ArcGIS ArcMap 2) Load all the shapefiles starting with with nf1 3) Open ArcToolbox, selecting the Merge Tool: <p>Data Management Tools -> General -> Merge</p> <ol style="list-style-type: none"> 4) Select all the nf1 shapefiles as input, and specify an output file of nf1x.shp. Execute the tool. 5) Repeat this process for each set of shapefiles i.e. nf1, nf2, nf3, nf4, nf5, nf6, nf7. 6) Once each has completed processing, upload each file to the itrc_hazards database using the PostGIS Shp and DBF Loader. The result is the following tables: <p>EA_NAFRA_nf1x EA_NAFRA_nf2x EA_NAFRA_nf3x EA_NAFRA_nf4x EA_NAFRA_nf5x EA_NAFRA_nf6x EA_NAFRA_nf7x</p> <ol style="list-style-type: none"> 7) Run the following SQL to union all the data together: <pre> DROP TABLE IF EXISTS "EA_NAFRA_all_Nov2012"; CREATE TABLE "EA_NAFRA_all_Nov2012" AS SELECT * FROM "NAFRA_nf1x" UNION ALL SELECT * FROM "NAFRA_nf2x" UNION ALL SELECT * FROM "NAFRA_nf3x" UNION ALL SELECT * FROM "NAFRA_nf4x" UNION ALL SELECT * FROM "NAFRA_nf5x" UNION ALL SELECT * FROM "NAFRA_nf6x" UNION ALL SELECT * FROM "NAFRA_nf7x"; ALTER TABLE "EA_NAFRA_all_Nov2012" ADD CONSTRAINT enforce_dims_geom CHECK (st_ndims(geom) = 2); ALTER TABLE "EA_NAFRA_all_Nov2012" ADD CONSTRAINT enforce_geotype_geom CHECK (geometrytype(geom) = 'MULTIPOLYGON'::text OR geom IS NULL); ALTER TABLE "EA_NAFRA_all_Nov2012" ADD CONSTRAINT enforce_srid_geom CHECK (st_srid(geom) = 27700); </pre>
Indicative_fluvial_flood_10k_2000 Indicative_tidal_flood_10k_2000	All of these datasets, supplied by the Environment Agency, as shapefiles, were uploaded to the itrc_hazards database via the

Nat_floodzone2_v3_8 Nat_floodzone3_v3_8	PostGIS SHP and DBF Loader. No processing was undertaken prior to this uploading.
NSRI_NPD_Flood	This data was supplied as an ESRI Shapefile and loaded in to the itrc_hazards database using the PostGIS SHP and DBF Loader. Further information is available at: http://www.landis.org.uk/npd

Figure 10 - processing and upload procedure for data within the itrc hazards database

Itrc_spatial_water

Table prefix/table name	Process Description
EA_WM_areawm_250k EA_WM_regionwm_250k	<p>This data was supplied by the Environment Agency, and uploaded to the water database using the plugin PostGIS SHP and DBF Loader.</p> <p>No processing was performed on this data prior to uploading.</p>
OS_Reservoirs	<p>This data was supplied by the Ordnance Survey, and uploaded to the water database using the plugin PostGIS SHP and DBF Loader.</p> <p>No processing was performed on this data prior to uploading.</p>
OS_Reservoirs_Covered OS_Reservoirs_Disused	<p>These tables were produced by filtering the “name” attribute of the OS_Reservoirs data (see above):</p> <ul style="list-style-type: none"> • “name” ILIKE ‘%Covered%’ • “name” ILIKE ‘%Disused%’ <p>This filtering was performed within PostgreSQL, using standard SQL.</p>
OS_WaterPumpingStations	<p>This data was supplied by the Ordnance Survey, and uploaded to the water database using the plugin PostGIS SHP and DBF Loader.</p> <p>No processing was performed on this data prior to uploading.</p>
OS_WaterPumpingStations_DisusedAssets OS_WaterPumpingStations_DrainingDrainageAssets OS_WaterPumpingStations_HydraulicAssets OS_WaterPumpingStations_Pumping OS_WaterPumpingStations_UndergroundAssets OS_WaterPumpingStations_WaterTowers OS_WaterPumpingStations_WindAssets	<p>These tables were produced by filtering the “name” attribute of the OS_WaterPumpingStations data (see above):</p> <ul style="list-style-type: none"> • "NAME" LIKE '%Disused%' • "NAME" LIKE '%Drainage%' OR "NAME" LIKE '%Draining%' • "NAME" LIKE '%Pumping%' • "NAME" LIKE '%Underground%' • "NAME" LIKE '%WaterTowers%' • "NAME" LIKE '%Wind%' <ol style="list-style-type: none"> 1) Open ESRI ArcGIS ArcMap 2) Load the original raw OS_WaterPumpingStations shapefile 3) Select by Attributes, and run each query as defined above. 4) Save each query as an ESRI expression file (*.exp) 5) Export the selected records to a new shapefile e.g. <ul style="list-style-type: none"> • OS_WaterPumpingStations_DisusedAssets.shp • OS_WaterPumpingStations_DrainingDrainageAssets.shp • OS_WaterPumpingStations_HydraulicAssets.shp • OS_WaterPumpingStations_Pumping.shp • OS_WaterPumpingStations_UndergroundAssets.shp • OS_WaterPumpingStations_WaterTowers.shp • OS_WaterPumpingStations_WindAssets.shp

	<p>6) Each of the afore-mentioned shapefiles was subsequently uploaded to the water database using the PostGIS SHP and DBF Loader.</p>
<p>os_waterpumpingstations_intersect_nat_floodzone2_v3_8 os_waterpumpingstations_intersect_nat_floodzone3_v3_8</p>	<p>These tables were produced by intersecting the original raw OS_WaterPumpingStations shapefile supplied by Ordnance Survey, with each of the flood zone shapefiles supplied by the Environment Agency.</p> <ol style="list-style-type: none"> 1) Open ESRI ArcGIS ArcMap 2) Load the OS_WaterPumpingStations shapefile data 3) Load the chosen flood zone shapefile 4) Using “Select by Location”, set the following settings: <ul style="list-style-type: none"> • “select features from” as Selection Method • OS_WaterPumpingStations as Target Layer • Nat_floodzone2_v3_8 or Nat_floodzone3_v3_8 as Source Layer • Target feature(s) intersect the Source layer feature as Spatial Selection Method 5) Export the selected features to a new ESRI Shapefile, for each flood zone. 6) Upload the resultant shapefiles to the water database using the PostGIS SHP and DBF Loader.
<p>OS_WaterPumpingStations_Intersect_nat_floodzone2_v3_8_Join_GOR OS_WaterPumpingStations_Intersect_nat_floodzone2_v3_8_Join_GOR</p>	<p>These tables were produced by initially performing a spatial join between the government office regions for England and Wales and the original raw OS_WaterPumpingStation data.</p> <p>NOTE: Only the government office regions for England and Wales were used as the OS_WaterPumpingStation data only covers these areas</p> <ol style="list-style-type: none"> 1) Open ESRI ArcGIS ArcMap 2) Load the OS_WaterPumpingStations shapefile data 3) Load the government office region data 3) Right click the pumping station data and select Join 4) Select “Join data from another layer based on spatial location” 5) Select the government office region data as the input layer (1) 6) Ensure that the option “Joining polygon to points” is specified. 7) Select “it falls inside” to denote that the point data i.e. pumping station data will have the government office region within which it lies added to itself. <p>The result is that each pumping station knows which government office region it lies inside.</p> <p>The data was then uploaded to the water database using the PostGIS SHP and DBF Loader plugin</p>
<p>os_strategi_combined_coast_polyline os_strategi_combined_foreshor_region</p>	<p>Each of these tables was created by combining the original separate feature types into a single feature type for the whole of</p>

<div>os_strategi_combined_lakes_region</div> <div>os_strategi_combined_rivers_polyline</div>	<div>the UK. The original Ordnance Survey-supplied Strategi data is separated into data related to the North, and data related to the South of the UK. Each of the original supplied shapefiles was uploaded using the PostGIS SHP and DBF Loader plugin. FME WorkBench was then utilised to merge data from the “North” and “South” versions of the data, into a single “combined” outcome, again stored in PostGIS.</div> <div>The original shapefiles, separated in to North and South of the UK, are stored within the itrc_spatial database, as PostGIS tables.</div>
--	--

Figure 11 - processing and upload procedure for data within the itrc water database

Itrc_spatial_wastewater

Table prefix/table name	Process Description
OS_Reservoirs	<p>This data was supplied by the Ordnance Survey, and uploaded to the water database using the plugin PostGIS SHP and DBF Loader.</p> <p>No processing was performed on this data prior to uploading.</p>
OS_WaterPumpingStations_DisusedAssets OS_WaterPumpingStations_DrainingDrainageAssets OS_WaterPumpingStations_HydraulicAssets OS_WaterPumpingStations_Pumping OS_WaterPumpingStations_UndergroundAssets OS_WaterPumpingStations_WaterTowers OS_WaterPumpingStations_WindAssets	<p>These tables were produced by filtering the “name” attribute of the OS_WaterPumpingStations data (see above):</p> <ul style="list-style-type: none"> • "NAME" LIKE '%Disused%' • "NAME" LIKE '%Drainage%' OR "NAME" LIKE '%Draining%' • "NAME" LIKE '%Pumping%' • "NAME" LIKE '%Underground%' • "NAME" LIKE '%WaterTowers%' • "NAME" LIKE '%Wind%' <ol style="list-style-type: none"> 1) Open ESRI ArcGIS ArcMap 2) Load the original raw OS_WaterPumpingStations shapefile 3) Select by Attributes, and run each query as defined above. 4) Save each query as an ESRI expression file (*.exp) 5) Export the selected records to a new shapefile e.g. <ul style="list-style-type: none"> • OS_WaterPumpingStations_DisusedAssets.shp • OS_WaterPumpingStations_DrainingDrainageAssets.shp • OS_WaterPumpingStations_HydraulicAssets.shp • OS_WaterPumpingStations_Pumping.shp • OS_WaterPumpingStations_UndergroundAssets.shp • OS_WaterPumpingStations_WaterTowers.shp • OS_WaterPumpingStations_WindAssets.shp 6) Each of the afore-mentioned shapefiles was subsequently uploaded to the water database using the PostGIS SHP and DBF Loader.
OS_WaterPumpingStations_Intersect_nat_floodzone2_v3_8_Join_GOR OS_WaterPumpingStations_Intersect_nat_floodzone2_v3_8_Join_GOR	<p>These tables were produced by initially performing a spatial join between the government office regions for England and Wales and the original raw OS_WaterPumpingStation data.</p> <p>NOTE: Only the government office regions for England and Wales were used as the OS_WaterPumpingStation data only covers these areas</p> <ol style="list-style-type: none"> 1) Open ESRI ArcGIS ArcMap 2) Load the OS_WaterPumpingStations shapefile data 3) Load the government office region data 3) Right click the pumping station data and select Join 4) Select “Join data from another layer based on spatial location” 5) Select the government office region data as the input layer

	<p>(1)</p> <p>6) Ensure that the option “Joining polygon to points” is specified.</p> <p>7) Select “it falls inside” to denote that the point data i.e. pumping station data will have the government office region within which it lies added to itself.</p> <p>The result is that each pumping station knows which government office region it lies inside.</p> <p>The data was then uploaded to the water database using the PostGIS SHP and DBF Loader plugin</p>
<p>Derived_poi_waste_storage_processing_disposal_wage_features</p> <p>Derived_poi_waste_storage_processing_disposal_sludge_features</p> <p>Derived_poi_waste_storage_processing_disposal_slurry_features</p>	<p>These derived data tables were generated by filtering the “Points_of_interest” data table using the pointx_classification_code attribute, and varying the value to extract the different feature types:</p> <pre>CREATE TABLE derived_poi_waste_storage_processing_disposal AS SELECT * FROM points_of_interest WHERE pointx_classification_code = '06340441';</pre> <pre>ALTER TABLE derived_poi_waste_storage_processing_disposal ADD CONSTRAINT "enforce_srid_geom" CHECK (st_srid(geom) = 27700); ALTER TABLE derived_poi_waste_storage_processing_disposal ADD CONSTRAINT "enforce_geotype_geom" CHECK (geometrytype(geom) = 'POINT'::text OR geom IS NULL); ALTER TABLE derived_poi_waste_storage_processing_disposal ADD CONSTRAINT "enforce_dims_gemo" CHECK (st_ndims(geom) = 2);</pre> <p>Waste Storage Processing Disposal (All) = 06340441</p> <p>In order to produce the filtered waste storage processing and disposal tables for sewage, sludge and slurry respectively, a SQL filter was applied e.g.</p> <pre>CREATE TABLE derived_poi_waste_storage_processing_disposal_sewage_features AS SELECT * FROM derived_poi_waste_storage_processing_disposal WHERE name ILIKE '%sew%';</pre> <pre>CREATE TABLE derived_poi_waste_storage_processing_disposal_slurry_features AS SELECT * FROM derived_poi_waste_storage_processing_disposal WHERE name ILIKE '%slur%';</pre> <pre>CREATE TABLE derived_poi_waste_storage_processing_disposal_sludge_features AS SELECT * FROM derived_poi_waste_storage_processing_disposal WHERE name ILIKE '%slud%';</pre>
<p>UWWTW_Waterbase_UWWTD_v2_codelist_Bigcities</p> <p>UWWTW_Waterbase_UWWTD_v2_codelist_LOV</p> <p>UWWTW_Waterbase_UWWTD_v2_codelist_NUTS</p>	<p>The city names mapping table (Bigcities), code mapping table (LOV) and NUT mapping tables (NUTS) were supplied as comma separated text files.</p> <p>FME Workbench was used to upload this data to the</p>

	<p>database. A csv feature type reader was used to read the data, and then a Postgres feature type writer used to write the data to the database.</p>
<p>T_UWWTPS_UK_ONLY_OSGB36_Join_GOR</p>	<p>The Urban Waste Water Treatment Plants data was supplied as a csv file, containing two columns used to generate coordinates of each plant; uwwlatitude and uwwlongitude.</p> <p>The data was loaded in to ESRI ArcGIS ArcMap, where it was subsequently exported as an ESRI shapefile, in WGS84. This data was then transformed to British National Grid using the ArcToolbox -> Projections and Transformations -> Features -> Project tool.</p> <p>The ArcToolbox -> Extract -> Clip tool was then used to clip the data to include UK only sites. The clipping polygon used was the UK outline, as used to denote all the government office regions.</p> <p>A spatial join between the resultant output features and the government office regions was performed, to create a shapefile containing all the waste water treatment plants, with each containing attributes denoting which government office region it lies within.</p> <p>This data was uploaded to the database using the PostGIS SHP and DBF Loader</p>
<p>UWWTW_T_MSLevel UWWTW_T_ReceivingAreas UWWTW_T_ReportPeriod UWWTW_T_Reporter UWWTW_T_Uwwtp_Agglo</p> <p>Feb2011 Update: UWWTW_Feb2011_T_MSLevel UWWTW_Feb2011_T_Reporter UWWTW_Feb2011_T_Uwwtp_Agglo UWWTW_Feb2011_T_ReceivingAreas UWWTW_Feb2011_T_ReportPeriod</p> <p>Aug2012 Update: UWWTW_Aug2012_T_MSLevel UWWTW_Aug2012_T_Reporter UWWTW_Aug2012_T_Uwwtp_Agglo UWWTW_Aug2012_T_ReceivingAreas UWWTW_Aug2012_T_ReportPeriod</p>	<p>This data was received in Excel spreadsheet format. FME Workbench was used to upload the data to the database.</p> <p>A Microsoft Excel reader feature type was used to read the data, whilst a Postgres feature type writer was used to write the data to the database.</p> <p>The SQL code executed to add the foreign key constraints between these tables, can be found in the appendix of this document, under title: SQL Code for foreign key constraints on European Environment Agency Waste Water Data</p>

<p>UWWTW_T_Agglomerations_UK_ONLY_OSGB UWWTW_T_DischargePoints_UK_ONLY_OSGB</p> <p>Feb 2011 Update</p> <p>UWWTW_Feb2011_T_Agglomerations_UK_ONLY_O SGB36 UWWTW_Feb2011_T_DischargePoints_UK_ONLY_O SGB36</p> <p>Aug 2012 Update</p> <p>UWWTW_T_Agglomerations_Aug2012_UK_ONLY_O SGB36 UWWTW_T_DischargePoints_Aug2012_UK_ONLY_O SGB36</p>	<p>This data was supplied as a series of csv files, containing two columns of coordinate information with each file:</p> <p>Discharge Point Coordinate Columns:</p> <p>Dcplattitud Dcplongitu</p> <p>Agglomeration Coordinate Columns:</p> <p>Agglattitud Agglongitu</p> <p>The separate csv files for agglomerations and discharge points were loaded in to ESRI ArcGIS ArcMap. A separate shapefile was created for the agglomerations and discharge points. This was subsequently converted to British National Grid coordinates from ETRS 1989 using ArcToolbox -> Projections and Transformations -> Project.</p> <p>Each resultant shapefile was then clipped to include features within the UK, using the ArcToolbox -> Extract -> Clip Tool.</p>
<p>UWWTW_EU_SA_TW_UK_ONLY_OSGB UWWTW_EU_SA_catchm_UK_ONLY_OSGB UWWTW_EU_SA_coastA_UK_ONLY_OSGB UWWTW_EU_SA_coastL_UK_ONLY_OSGB UWWTW_EU_SA_lake_UK_ONLY_OSGB UWWTW_EU_SA_river_UK_ONLY_OSGB</p> <p>Feb 2011 Update:</p> <p>UWWTW_Feb2011_EU_LSA_coastL_UK_ONLY_OSG B36 UWWTW_Feb2011_EU_SA_TW_UK_ONLY_OSGB36 UWWTW_Feb2011_EU_SA_catchm_UK_ONLY_OSG B36 UWWTW_Feb2011_EU_SA_coastA_UK_ONLY_OSGB 36 UWWTW_Feb2011_EU_SA_coastL_UK_ONLY_OSGB 36 UWWTW_Feb2011_EU_SA_lake_UK_ONLY_OSGB36 UWWTW_Feb2011_EU_SA_river_UK_ONLY_OSGB3 6</p> <p>Aug 2012 Update: No data supplied</p>	<p>The original data was supplied as a series of shapefiles in ETRS 1989. The data was loaded in to ESRI ArcGIS ArcMap, and transformed to WGS84 coordinates using ArcToolbox -> Projections and Transformations -> Project.</p> <p>The data was also transformed to British National Grid coordinates using the same method.</p> <p>Each resultant shapefile was then clipped to include only features within the UK. This was performed using the ArcToolbox -> Extract -> Clip tool.</p> <p>The result was a shapefile for UK only features in British National Grid coordinates.</p> <p>These shapefiles were loaded in to the database using the PostGIS SHP and DBF Loader.</p>

<p>Feb 2011 Update:</p> <p>UWWTW_T_DischargePoints_UK_ONLY_OSGB36_Feb2011_Int_nat_fldz_2_v</p> <p>UWWTW_T_DischargePoints_UK_ONLY_OSGB36_Feb2011_Int_nat_fldz_3_v</p> <p>Aug 2012 Update:</p> <p>UWWTW_T_DischargePoints_UK_ONLY_OSGB36_Aug2012_Int_nat_fldz_2_v</p> <p>UWWTW_T_DischargePoints_UK_ONLY_OSGB36_Aug2012_Int_nat_fldz_3_v</p> <p>UWWTW_T_UWWTPS_UK_ONLY_OSGB36_Intersect_nat_floodzone_2_v3_8</p> <p>UWWTW_T_UWWTPS_UK_ONLY_OSGB36_Intersect_nat_floodzone_3_v3_8</p> <p>Feb 2011 Update:</p> <p>T_UWWTPS_UK_ONLY_OSGB36_Feb2011_Int_nat_fldz_2_v3_8</p> <p>T_UWWTPS_UK_ONLY_OSGB36_Feb2011_Int_nat_fldz_3_v3_8</p> <p>Aug 2012 Update:</p> <p>T_UWWTPS_UK_ONLY_OSGB36_Int_nat_floodzone_2_v3_8_Join_GOR</p> <p>T_UWWTPS_UK_ONLY_OSGB36_Int_nat_floodzone_3_v3_8_Join_GOR</p>	<p>Each separate discharge point/agglomeration shapefile (for Feb 2011, Aug 2012) was loaded in to ESRI ArcGIS ArcMap</p> <p>The Environment Agency Flood Zone shapefiles were also loaded in to ArcMap. Using “Select by Location” the discharge point files were intersected against each flood zone shapefile:</p> <p>Selection method: ‘select features from’</p> <p>Target Layer: <input_discharge_point_shapefile> / <input_agglomeration_shapefile></p> <p>Source Layer: <input_EA_flood_zone_shapefile></p> <p>Spatial selection method: Target layer(s) features intersect the Source layer feature</p> <p>The result of executing this intersection was a set of features that lie within the different EA flood zone files.</p> <p>Each output shapefile was then loaded in to the database using the PostGIS SHP and DBF Loader.</p>
---	--

Figure 12 - processing and upload procedures for waste water data

Itrc_spatial_solidwaste

Table prefix/table name	Process Description
Scotland_Active_Landfill_Capacity_2007 Scotland_Non_Active_Landfill_Capacity_2007	<p>This data was uploaded using FME Workbench. A workspace reading in the excel spreadsheets of active and non active landfill capacity data was created. A PostgreSQL writer, pointing to the solid waste database was used to write this data out to the database.</p> <p>No processing was applied to this data prior to it's upload to the solid waste database.</p>
Scotland_Business_waste_arisings_LA_2009 Scotland_Business_waste_arisings_sector_2009 Scotland_Construction_Demolition_Waste_Managed_2009 Scotland_LA_Composted_Material_2009_2010 Scotland_LA_Controlled_Waste_Landfill_2009 Scotland_LA_Municipal_Compost_Method_2009_2010 Scotland_LA_Municipal_Disposed_2009_2010 Scotland_LA_Municipal_Waste_Disposed_Type_Method_2009_2010 Scotland_LA_Municipal_Waste_Material_2009_2010 Scotland_LA_Municipal_Waste_Recycled_By_Source_2009_2010 Scotland_LA_Municipal_Waste_Recycled_Collection_Type_2009_2010 Scotland_LA_Municipal_Recycling_Compost_2009_2010 Scotland_LA_Special_Waste_Landfill_2009 Scotland_LA_Waste_EWC_Stat Scotland_LA_collected_municipal_waste_2009_2010 Scotland_LA_collected_municipal_waste_breakdown_2009_2010 Scotland_LA_collected_non_municipal_waste_breakdown_2009_2010 Scotland_Total_Waste_Arisings_2009 Scotland_Type_Construction_Demolition_Waste_Managed_2009	<p>This data was sourced from the Scotland Waste Data Digest (http://www.sepa.org.uk/waste/waste_data/waste_data_digest.aspx).</p> <p>The original data is delivered as an Excel spreadsheet, with each sheet corresponding to a different table of data. The attribute names of each table were mapped to a sensible value, reducing the length of the attribute name and making them therefore compatible with storage in a relational database.</p> <p>The data was uploaded using FME Workbench. A workspace was created that read the resultant mapped excel spreadsheet, containing only the tables of interest from the Data Digest. This was then output through a PostgreSQL writer.</p> <p>No processing on the data values themselves was performed prior to uploading the data.</p>
Derived_poi_recycling_centres Derived_poi_refuse_disposal_facilities Derived_poi_waste_storage_processing_disposal	<p>This data was originally supplied by the Ordnance Survey for the North West region only, as a .csv file. The data was loaded in to the "points of interest" database using FME Workbench. A CSV reader was used to read the input data, whilst a PostGIS writer was used to write the data to the database.</p> <p>The original csv file stored the coordinates of each point of interest in the attributes "ITN_Easting", and "ITN_Northing". To convert these attributes to a point geometry within FME, a 2DPointReplacer Transformer was used. This assigns the ITN_Easting attribute as the X component, and ITN_Northing</p>

	<p>attribute as the Y component. Further information can be found here: http://docs.safe.com/fme/2010/html/FME_Transformers/content/transformers/2dpointreplacer.htm.</p> <p>Finally once the data had been written to the database, the following SQL was executed to create tables of different points of interest (recycling centres (06340462), refuse_disposal_facilities (06340440), waste_storage_processing_disposal (06340441)):</p> <pre> DROP TABLE IF EXISTS derived_poi_recycling_centres; CREATE TABLE derived_poi_recycling_centres AS SELECT * FROM points_of_interest WHERE pointx_classification_code = '06340462'; ALTER TABLE derived_poi_recycling_centres ADD CONSTRAINT "enforce_srid_geom" CHECK (st_srid(geom) = 27700); ALTER TABLE derived_poi_recycling_centres ADD CONSTRAINT "enforce_geotype_geom" CHECK (geometrytype(geom) = 'POINT':text OR geom IS NULL); ALTER TABLE derived_poi_recycling_centres ADD CONSTRAINT "enforce_dims_gemo" CHECK (st_ndims(geom) = 2); DROP TABLE IF EXISTS derived_poi_refuse_disposal_facilities; CREATE TABLE derived_poi_refuse_disposal_facilities AS SELECT * FROM points_of_interest WHERE pointx_classification_code = '06340440'; ALTER TABLE derived_poi_refuse_disposal_facilities ADD CONSTRAINT "enforce_srid_geom" CHECK (st_srid(geom) = 27700); ALTER TABLE derived_poi_refuse_disposal_facilities ADD CONSTRAINT "enforce_geotype_geom" CHECK (geometrytype(geom) = 'POINT':text OR geom IS NULL); ALTER TABLE derived_poi_refuse_disposal_facilities ADD CONSTRAINT "enforce_dims_gemo" CHECK (st_ndims(geom) = 2); DROP TABLE IF EXISTS derived_poi_waste_storage_processing_disposal; CREATE TABLE derived_poi_waste_storage_processing_disposal AS SELECT * FROM points_of_interest WHERE pointx_classification_code = '06340441'; ALTER TABLE derived_poi_waste_storage_processing_disposal ADD CONSTRAINT "enforce_srid_geom" CHECK (st_srid(geom) = 27700); ALTER TABLE derived_poi_waste_storage_processing_disposal ADD CONSTRAINT "enforce_geotype_geom" CHECK (geometrytype(geom) = 'POINT':text OR geom IS NULL); ALTER TABLE derived_poi_waste_storage_processing_disposal ADD CONSTRAINT "enforce_dims_gemo" CHECK (st_ndims(geom) = 2); </pre>
Scotland_active_landfill_sites_2007 Scotland_active_landfill_sites_2008 Scotland_non_active_landfill_sites_2007 Scotland_non_active_landfill_sites_2008	<p>This data was sourced from the 2007 and 2008 Scottish Environment Protection Agency (SEPA) reports: <i>Landfill Capacity Report for Scotland</i></p> <p>Each report contained a table of remaining landfill capacities by site, at the end of each reporting year. Within the data, a National Grid reference was given to locate each site, on the 100km grid. To retrieve the true National Grid coordinate i.e.</p>

Easting and Northing, the following procedure was applied to each coordinate within Excel:

e.g. NO 6882 9731

- 1) Extract the two letters at the beginning of the reference e.g. NO, to a separate column
- 2) Extract the first component e.g. 6882, called e
- 3) Extract the second component e.g. 9731, called n
- 4) The following grid was used to determine what the initial Easting and Northing coordinate digit should be, based on the letters extracted in 1).

e.g. NO

Easting for NO = 300

Northing for NO = 700

The additional '00' can be ignored for now.

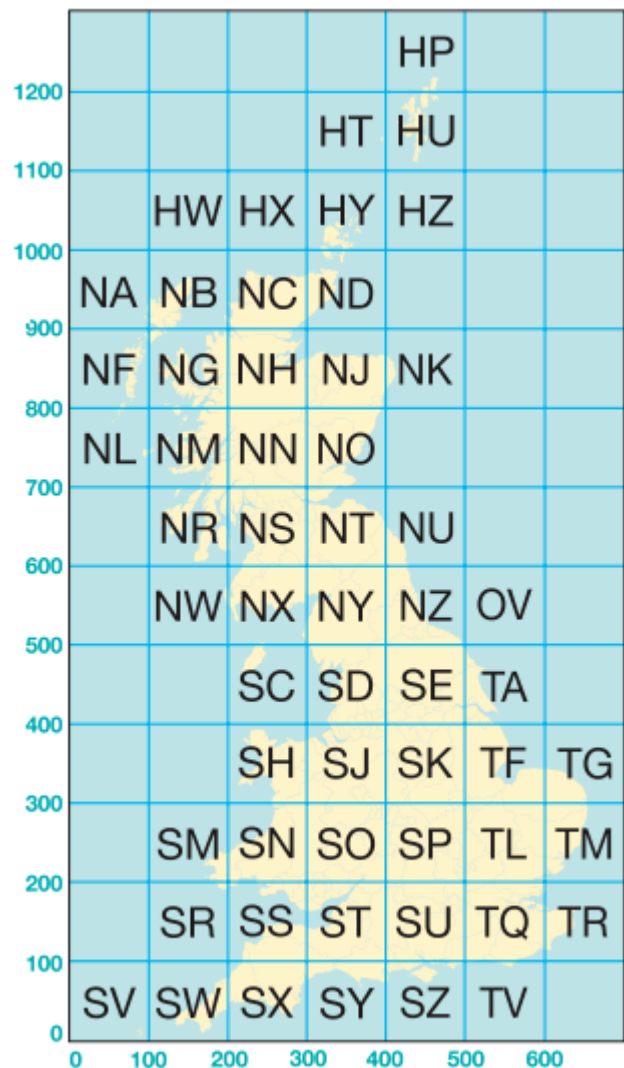


Figure 13 - National Grid Letter Mapping

(<http://www.ordnancesurvey.co.uk/oswebsite/gi/nationalgrid/nationalgrid.pdf>)

	<p>This was repeated for all sites within all reports that contained National Grid references as the only geographic coordinate information.</p> <p>The attribute names of each sheet were mapped to a sensible value, reducing the length of the attribute name and making them therefore compatible with storage in a relational database.</p> <p>Each spreadsheet <i>sheet</i> was subsequently created as an ESRI Feature Class within a Geodatabase. A separate Geodatabase was created for each year's worth of data. This operation was performed within ESRI ArcMap.</p> <p>FME Workbench was used to read the resultant feature classes using a Geodatabase reader, and then subsequently written to the database using a PostGIS writer.</p>
Scotland_landfills_authorised_2009 Scotland_landfills_authorised_2010 Scotland_landfills_closed_2009 Scotland_landfills_closed_2010 Scotland_landfills_restoration_2009 Scotland_landfills_restoration_2010	<p>This data was sourced from the 2009 and 2010 Scottish Environment Protection Agency (SEPA) reports: <i>Landfill Capacity Report for Scotland</i>.</p> <p>The original spreadsheet for each year contained a column of Easting and Northing's. This negated the need to perform the steps detailed above to determine the full Easting or Northing component of the coordinate.</p> <p>The attribute names of each sheet were mapped to a sensible value, reducing the length of the attribute name and making them therefore compatible with storage in a relational database.</p> <p>Each spreadsheet <i>sheet</i> was created as an ESRI Feature Class within a Geodatabase, with a separate Geodatabase created for each year's worth of data. This operation was performed within ESRI ArcMap.</p> <p>Finally FME Workbench was used to read the resultant feature classes using a Geodatabase reader, and then subsequently written to the database using a PostGIS writer.</p>
Scotland_wastesites_nothandlingwaste_2007 Scotland_wastesites_wml_ppcsitehandlingwaste_2007 Scotland_ppcsites_2007	<p>This data was sourced from spreadsheets released alongside the 2007 Scottish Environment Protection Agency (SEPA) reports: <i>National Waste Capacity Report for Scotland</i></p> <p>Initially the data was separated into three distinct sets:</p> <ul style="list-style-type: none"> • Waste Management Licence/Pollution Prevention and Control Sites • Sites not handling waste • Pollution Prevention Control Sites only

	<p>Within the 2007 data, only a 100km grid reference was given to locate each site, therefore these were converted to full National Grid coordinates (see method detailed above for full procedure).</p> <p>Each resultant set was then converted to a Feature Class to be stored in a Geodatabase, via the use of ESRI ArcMap.</p>
<p>Scotland_wastesites_siteshandlingwaste_2008</p> <p>Scotland_wastesites_sitesnohandlingwaste_2008</p> <p>Scotland_wastesites_allsites_2009</p> <p>Scotland_wastesites_2010</p>	<p>This data was sourced from spreadsheets released alongside the Scottish Environment Protection Agency (SEPA) reports:</p> <ul style="list-style-type: none"> • National Waste Capacity Report 2008 • National Waste Capacity Report 2009 • Waste Sites and Capacity Report 2010 <p>Initially the data for 2008 was split into those sites handling, and those not handling sites. Each of the spreadsheet <i>sheets</i> already contained full National Grid coordinates.</p> <p>A separate ESRI Geodatabase was created for each year's worth of data. A separate feature class for each subset of data for each year was then created via ESRI ArcMap, using the National Grid coordinates already present in the original data.</p> <p>FME Workbench was then used to read each of the subsequent ESRI Geodatabase's and a PostGIS writer to write the data out to the solid waste database.</p>

Figure 14 - processing and upload procedures for solid waste data

Itrc_ws1_solid_waste_distances

Table prefix/table name	Process Description
East_Midlands_Centroid East_Of_England_Centroid GLA_Centroid North_East_Centroid North_West_Centroid South_East_Centroid South_West_Centroid Yorkshire_Humber_Centroid West_Midlands_Centroid Wales_Centroid Scotland_Centroid	<p>Each government office region centroid was calculated using the ArcToolbox -> Data Management Tools -> Features -> Feature To Point tool, available from ESRI ArcGIS ArcMap</p> <p>The result was an ESRI shapefile per government office region. Each shapefile was loaded in to the database using the PostGIS SHP and DBF Loader plugin.</p>
East_Midlands_Vertices East_Of_England_Vertices GLA_Vertices North_East_Vertices North_West_Vertices South_East_Vertices South_West_Vertices Yorkshire_Humber_Vertices West_Midlands_Vertices Wales_Vertices Scotland_Vertices	<p>Each government office region vertices table was calculated using the ESRI ArcGIS ArcToolbox tool ArcToolbox -> Data Management Tools -> Feature Vertices To Point.</p> <p>The result was an ESRI shapefile per government office region. Each shapefile was loaded in to the database using the PostGIS SHP and DBF Loader plugin.</p>
England Output Areas (County) England_OA_2001_Centroid_EastEngland_County_GOR_BD_AVG England_OA_2001_Centroid_EastMidlands_County_GOR_BD_AVG England_OA_2001_Centroid_GLA_County_GOR_BD_AVG England_OA_2001_Centroid_NorthEast_County_GOR_BD_AVG England_OA_2001_Centroid_NorthWest_County_GOR_BD_AVG England_OA_2001_Centroid_SouthEast_County_GOR_BD_AVG England_OA_2001_Centroid_SouthWest_County_GOR_BD_AVG England_OA_2001_Centroid_WestMidlands_County_GOR_BD_AVG England_OA_2001_Centroid_YorkshireHumber_County_GOR_BD_AVG England Output Areas (Unitary Authority) England_UA_2001_Centroid_EastMidlands_GOR_BD_AVG England_UA_2001_Centroid_EastOfEngland_GOR_BD_AVG England_UA_2001_Centroid_NorthEast_GOR_BD_A	<p>Each table was calculated by executing the plpgsql function calculate_GOR_centroid_to_boundary_distance for each government office region e.g.</p> <p>The input to the function are:</p> <ol style="list-style-type: none"> 1) A table containing the centroids of output areas within the government office region 2) The vertices of the government office region boundary 3) The name of the output table to create <p>SELECT * FROM calculate_GOR_centroid_to_boundary_distance('England_OA_2001_County_Centroid_East_Midlands_Near_GOR_Dist','England_GOR_East_Midlands_Vertices','England_OA_2001_Centroid_EastMidlands_County_GOR_BoundDist_AVG');</p> <p>The details of the function calculate_GOR_centroid_to_boundary_distance can be found in appendix section '<i>plpgsql function code for calculate_GOR_centroid_to_boundary_distance</i>'.</p>

<p>VG England_UA_2001_Centroid_NorthWest_GOR_BD_AVG England_UA_2001_Centroid_SouthEast_GOR_BD_AVG England_UA_2001_Centroid_SouthWest_GOR_BD_AVG England_UA_2001_Centroid_WestMidlands_GOR_BD_AVG England_UA_2001_Centroid_YorkshireHumber_GOR_BD_AVG</p>	
<p>England Output Areas (County)</p> <p>England_OA_2001_Centroid_EastEngland_County_GOR_BD_AVG_Join England_OA_2001_Centroid_EastMidlands_County_GOR_BD_AVG_Join England_OA_2001_Centroid_GLA_County_GOR_BD_AVG_Join England_OA_2001_Centroid_NorthEast_County_GOR_BD_AVG_Join England_OA_2001_Centroid_NorthWest_County_GOR_BD_AVG_Join England_OA_2001_Centroid_SouthEast_County_GOR_BD_AVG_Join England_OA_2001_Centroid_SouthWest_County_GOR_BD_AVG_Join England_OA_2001_Centroid_WestMidlands_County_GOR_BD_AVG_Join England_OA_2001_Centroid_YorkshireHumber_County_GOR_BD_AVG_Join</p> <p>England Output Areas (Unitary Authority)</p> <p>England_UA_2001_Centroid_EastMidlands_GOR_BD_AVG_Join England_UA_2001_Centroid_EastOfEngland_GOR_BD_AVG_Join England_UA_2001_Centroid_NorthEast_GOR_BD_AVG_Join England_UA_2001_Centroid_NorthWest_GOR_BD_AVG_Join England_UA_2001_Centroid_SouthEast_GOR_BD_AVG_Join England_UA_2001_Centroid_SouthWest_GOR_BD_AVG_Join England_UA_2001_Centroid_WestMidlands_GOR_BD_AVG_Join England_UA_2001_Centroid_YorkshireHumber_GOR_BD_AVG_Join</p>	<p>Each join table was derived by joining the appropriate output area centroid geometry back to the average distance tables calculated for each government office region. This was repeated for each set of output areas per government office region e.g.</p> <pre>CREATE TABLE "England_OA_2001_Centroid_EastMidlands_County_GOR_BoundDist_AVG_Join" AS SELECT "England_OA_2001_County_Centroid_East_Midlands_Near_GOR_Dist".*, "England_OA_2001_Centroid_EastMidlands_County_GOR_BoundDist_AVG"."AVG_Vertex_Distance" FROM "England_OA_2001_Centroid_EastMidlands_County_GOR_BoundDist_AVG", "England_OA_2001_County_Centroid_East_Midlands_Near_GOR_Dist" WHERE "England_OA_2001_Centroid_EastMidlands_County_GOR_BoundDist_AVG".gid = "England_OA_2001_County_Centroid_East_Midlands_Near_GOR_Dist".gid ;</pre> <pre>ALTER TABLE "England_OA_2001_Centroid_EastMidlands_County_GOR_BoundDist_AVG_Join" ADD CONSTRAINT enforce_dims_geom CHECK (st_ndims(geom) = 2); ALTER TABLE "England_OA_2001_Centroid_EastMidlands_County_GOR_BoundDist_AVG_Join" ADD CONSTRAINT enforce_geotype_geom CHECK (geometrytype(geom) = 'POINT'::text OR geom IS NULL); ALTER TABLE "England_OA_2001_Centroid_EastMidlands_County_GOR_BoundDist_AVG_Join" ADD CONSTRAINT enforce_srid_geom CHECK (st_srid(geom) = 27700);</pre>

Figure 15 - processing for derived data to calculate distances for WS1 Solid Waste CDAM

Itrc_spatial_transport

Table prefix/table name	Process Description
CAA_Table_01_Size_of_UK_Airports_xxxx CAA_Table_02_1_Main_Outputs_of_UK_Airports_xxxx CAA_Table_02_2_Summary_Of_Activity_at_UK_Airports_xxxx CAA_Table_02_3_Use_of_UK_Airports_xxxx CAA_Table_02_4_Use_of_UK_Airports_by_Purpose_xxxx CAA_Table_03_1_Aircraft_Movements_xxxx CAA_Table_03_2_Aircraft_Movements CAA_Table_04_1_Air_Transport_Movements_xxxx CAA_Table_04_1_Air_Transport_Movements_xxxx CAA_Table_05_Air_Transport_Movements_xxxx CAA_Table_06_Air_Transport_Movements_vs_Previous_Year_xxxx CAA_Table_07_1_Air_Transport_Movements_Diverted_xxxx CAA_Table_07_2_Number_of_Diversions_to_UK_Airports_xxxx CAA_Table_08_Air_Passengers_by_Type_and_Nat_of_Operator_xxxx CAA_Table_09_Terminal_and_Transit_Pax_xxxx CAA_Table_10_1_EU_and_Other_Intl_Terminal_Pax_Traffic_xxxx CAA_Table_10_2_Domestic_Terminal_Pax_Traffic_xxxx CAA_Table_10_3_Terminal_Pax_xxxx CAA_Table_11_Intl_Pax_Traffic_to_from_UK_by_Country_xxxx CAA_Table_12_1_Intl_Air_Pax_Route_Analysis CAA_Table_13_1_Freight_by_Type_and_Nat_of_Operator_xxxx CAA_Table_13_2_Freight_xxxx CAA_Table_14_Intl_and_Domestic_Freight_xxxx CAA_Table_15_Freight_by_Aircraft_Configuration_xxxx CAA_Table_16_1_Mail_by_Type_and_Nat_of_Operator_xxxx CAA_Table_16_2_Mail_xxxx CAA_Table_17_Intl_and_Domestic_Mail_xxxx CAA_Table_18_Mail_by_Acft_Configuration_xxxx CAA_Table_19_Pax_and_Air_Transport_Movements_by_fixed_and_rotar	<p>This data was supplied a series of 19 separate Excel spreadsheets, with each output table corresponding to a sheet within each spreadsheet.</p> <p>FME Workbench was used to write this data to the database. A Microsoft Excel feature type reader was used to read each sheet from each file. A Postgres feature type writer was then used to write the data to the database.</p> <p>No processing of the data was performed prior to it being uploaded to the database.</p>
OS_Airports OS_Seaports	<p>This data was loaded using the PostGIS SHP and DBF Loader plugin.</p> <p>No processing was performed on this data prior to it being uploaded.</p>
Os_strategi_combined_a_road_polyline	Each of these tables was created by combining the original

<p>Os_strategi_combined_b_road_polyline Os_strategi_combined_ferry_polyline Os_strategi_combined_minor_rd_polyline Os_strategi_combined_motorway_polyline Os_strategi_combined_primry_rd_polyline Os_strategi_combined_railway_polyline</p>	<p>separate feature types into a single feature type for the whole of the UK. The original Ordnance Survey-supplied Strategi data is separated into data related to the North, and data related to the South of the UK. Each of the original supplied shapefiles was uploaded using the PostGIS SHP and DBF Loader plugin. FME WorkBench was then utilised to merge data from the “North” and “South” versions of the data, into a single “combined” outcome, again stored in PostGIS.</p> <p>The original shapefiles, separated in to North and South of the UK, are stored within the itrc_spatial database, as PostGIS tables.</p>
<p>os_strategi_comb_a_road_polyline_int_nat_floodzone_2_v3_8 os_strategi_comb_a_road_polyline_int_nat_floodzone_3_v3_8 os_strategi_comb_b_road_polyline_int_nat_floodzone_2_v3_8 os_strategi_comb_b_road_polyline_int_nat_floodzone_3_v3_8 os_strategi_comb_minor_rd_polyline_int_nat_floodzone_2_v3_8 os_strategi_comb_minor_rd_polyline_int_nat_floodzone_3_v3_8 os_strategi_comb_motorway_polyline_int_nat_floodzone_2_v3_8 os_strategi_comb_motorway_polyline_int_nat_floodzone_3_v3_8 os_strategi_comb_primry_rd_polyline_int_nat_floodzone_2_v3_8 os_strategi_comb_primry_rd_polyline_int_nat_floodzone_3_v3_8 os_strategi_combined_railway_polyline_int_nat_floodzone_2_v3_8 os_strategi_combined_railway_polyline_int_nat_floodzone_3_v3_8</p>	<p>These tables were produced by intersecting the combined OS Strategi data, with each of the flood zone shapefiles supplied by the Environment Agency.</p> <ol style="list-style-type: none"> 1) Open ESRI ArcGIS ArcMap 2) Load the OS Strategi shapefile data 3) Load the chosen flood zone shapefile 4) Using “Select by Location”, set the following settings: <ul style="list-style-type: none"> • “select features from” as Selection Method • <insert OS Strategi Layer of choice> as Target Layer • Nat_floodzone2_v3_8 or Nat_floodzone3_v3_8 as Source Layer • Target feature(s) intersect the Source layer feature as Spatial Selection Method 5) Export the selected features to a new ESRI Shapefile, for each flood zone. 6) Upload the resultant shapefiles to the transport database using the PostGIS SHP and DBF Loader.
<p>os_strategi_comb_a_road_polyline_int_nat_fldzn_2_v3_8_Join_GOR os_strategi_comb_a_road_polyline_int_nat_fldzn_3_v3_8_Join_GOR os_strategi_comb_b_road_polyline_int_nat_fldzn_2_v3_8_Join_GOR os_strategi_comb_b_road_polyline_int_nat_fldzn_3_v3_8_Join_GOR os_strategi_comb_minor_rd_plyln_int_nat_fldzn_2_v3_8_Join_GOR os_strategi_comb_minor_rd_plyln_int_nat_fldzn_3_v3_8_Join_GOR os_strategi_comb_motorway_plyln_int_nat_fldzn_2_v3_8_Join_GOR</p>	<p>These tables were produced by initially performing a spatial join between the government office regions for England and Wales and the shapefiles produced by intersecting the OS Strategi data with the EA flood zones.</p> <ol style="list-style-type: none"> 1) Open ESRI ArcGIS ArcMap 2) Load the OS_Strategi shapefile data 3) Load the government office region data 3) Right click the OS Strategi data and select Join 4) Select “Join data from another layer based on spatial location” 5) Select the government office region data as the input layer (1) 6) Ensure that the option “Joining polygon to points” is

os_strategi_comb_motorway_plyln_int_nat_fldzn_3_v3_8_Join_GOR os_strategi_comb_primry_rd_plyln_int_nat_fldzn_2_v3_8_Join_GOR os_strategi_comb_primry_rd_plyln_int_nat_fldzn_3_v3_8_Join_GOR os_strategi_comb_railway_plyln_int_nat_fldzn_2_v3_8_Join_GOR os_strategi_comb_railway_plyln_int_nat_fldzn_3_v3_8_Join_GOR	<p>specified.</p> <p>7) Select “it falls inside” to denote that the point data i.e. OS Strategi data will have the government office region within which it lies added to itself.</p> <p>The result is that each pumping station knows which government office region it lies inside.</p> <p>The data was then uploaded to the transport database using the PostGIS SHP and DBF Loader plugin</p>
Os_meridian_2_a_road_polyline Os_meridian_2_b_road_polyline Os_meridian_2_junction_font_point Os_meridian_2_minor_rd_polyline Os_meridian_2_motorway_polyline Os_meridian_2_rail_ln_polyline Os_meridian_2_rndabout_point Os_meridian_2_roadnode_point Os_meridian_2_station_point	<p>Each of these tables was originally supplied as a series of ESRI shapefiles, covering all of the UK.</p> <p>FME WorkBench was used to create a workspace able to read the shapefiles, and then write them out as PostGIS tables.</p> <p>No processing was performed on these raw datasets, prior to their uploading.</p>
os_meridian_a_road_intersect_nat_floodzone2_v3_8 os_meridian_a_road_intersect_nat_floodzone3_v3_8 os_meridian_b_road_intersect_nat_floodzone2_v3_8 os_meridian_b_road_intersect_nat_floodzone3_v3_8 os_meridian_minor_road_intersect_nat_floodzone2_v3_8 os_meridian_minor_road_intersect_nat_floodzone3_v3_8 os_meridian_motorway_intersect_nat_floodzone2_v3_8 os_meridian_motorway_intersect_nat_floodzone3_v3_8 os_meridian_rail_ln_intersect_nat_floodzone2_v3_8 os_meridian_rail_ln_intersect_nat_floodzone3_v3_8 os_meridian_rndabout_pnt_intersect_nat_floodzone2_v3_8 os_meridian_rndabout_pnt_intersect_nat_floodzone3_v3_8 os_meridian_station_pnt_intersect_nat_floodzone2_v3_8 os_meridian_station_pnt_intersect_nat_floodzone3_v3_8	<p>These tables were produced by intersecting the combined OS Meridian 2 data, with each of the flood zone shapefiles supplied by the Environment Agency.</p> <ol style="list-style-type: none"> 1) Open ESRI ArcGIS ArcMap 2) Load the OS Meridian 2 shapefile data 3) Load the chosen flood zone shapefile 4) Using “Select by Location”, set the following settings: <ul style="list-style-type: none"> • “select features from” as Selection Method • <insert OS Meridian 2 Layer of choice> as Target Layer • Nat_floodzone2_v3_8 or Nat_floodzone3_v3_8 as Source Layer • Target feature(s) intersect the Source layer feature as Spatial Selection Method 5) Export the selected features to a new ESRI Shapefile, for each flood zone. 6) Upload the resultant shapefiles to the transport database using the PostGIS SHP and DBF Loader.
OS_Meridian_A_Road_Int_nat_floodzone2_v3_8_Join_GOR OS_Meridian_A_Road_Intersect_nat_floodzone3_v3_8_Join_GOR	<p>These tables were produced by initially performing a spatial join between the government office regions for England and Wales and the shapefiles produced by intersecting the OS Meridian data with the EA flood zones.</p>

<p>OS_Meridian_B_Road_Intersect_nat_floodzone2_v3_8_Join_GOR</p> <p>OS_Meridian_B_Road_Intersect_nat_floodzone3_v3_8_Join_GOR</p> <p>OS_Meridian_Minor_Rd_Intersect_nat_floodzone2_v3_8_Join_GOR</p> <p>OS_Meridian_Minor_Rd_Intersect_nat_floodzone3_v3_8_Join_GOR</p> <p>OS_Meridian_Motorway_Intersect_nat_floodzone2_v3_8_Join_GOR</p> <p>OS_Meridian_Motorway_Intersect_nat_floodzone3_v3_8_Join_GOR</p> <p>OS_Meridian_Rail_Ln_Intersect_nat_floodzone2_v3_8_Join_GOR</p> <p>OS_Meridian_Rail_Ln_Intersect_nat_floodzone3_v3_8_Join_GOR</p> <p>OS_Meridian_Station_Pnt_Intersect_nat_floodzone2_v3_8_Join_GOR</p> <p>OS_Meridian_Station_Pnt_Intersect_nat_floodzone3_v3_8_Join_GOR</p>	<ol style="list-style-type: none"> 1) Open ESRI ArcGIS ArcMap 2) Load the OS_Meridian shapefile data 3) Load the government office region data 3) Right click the OS Meridian data and select Join 4) Select “Join data from another layer based on spatial location” 5) Select the government office region data as the input layer (1) 6) Ensure that the option “Joining polygon to points” is specified. 7) Select “it falls inside” to denote that the point data i.e. OS Strategi data will have the government office region within which it lies added to itself. <p>The result is that each pumping station knows which government office region it lies inside.</p> <p>The data was then uploaded to the transport database using the PostGIS SHP and DBF Loader plugin</p>
<p>Naptan_coachreferences</p> <p>Naptan_ferryreferences</p> <p>Naptan_flexible</p> <p>Naptan_hailride_end</p> <p>Naptan_hailride_start</p> <p>Naptan_railreferences</p> <p>Naptan_stopareas</p> <p>Naptan_stops</p>	<p>The original National Public Transport Access Nodes (NAPTAN) data was supplied as a set of csv files. Each file contained Easting and Northing columns giving geographic coordinates to each feature in each file. Each file was loaded in to ESRI ArcGIS ArcMap, and subsequently converted in to an ESRI Shapefile. The data was then uploaded via the use of the PostGIS DBF and SHP Loader plugin.</p>
<p>Nptg_localities</p> <p>Nptg_plusbusmapping</p>	<p>The original National Public Transport Gazetteer (NPTG) data was supplied as a series of csv files. These files were converted to ESRI Shapefiles via the use of ESRI ArcGIS ArcMap, as each contained Easting and Northing coordinate information. The resultant shapefiles were subsequently uploaded to the database via the PostGIS DBF and SHP Loader plugin.</p>
<p>DFT_Annualaveragedailyflow</p> <p>DFT_Traffic</p>	<p>Annual Average Daily Flow and Traffic Data sourced from the Department for Transport (Dft) was delivered in separate csv files for each government office region (12).</p> <p>An FME Workbench workspace was created, with a CSV reader added to read either all by-region daily flow csv files, or all by-region traffic data csv files. A Postgres writer was added to then write out this data to Postgres tables; a table for daily flow data, and a table for traffic data.</p>
<p>DFT_CP_locations</p>	<p>The count point locations for the annual average daily flow and traffic data are repeated within each set. To extract a table of distinct count point locations from the data, the following SQL was executed, creating a table of unique count point locations.</p>

	<pre>CREATE TABLE dft_cp_locations AS SELECT DISTINCT("CP") as cp, "SRefE" as srefe, "SRefN" as srefn, ST_SetSRID(ST_MakePoint("SRefE", "SRefN"),27700) as geom FROM dft_annualaveragedailyflow ORDER BY "CP" ASC; ALTER TABLE dft_cp_locations ADD CONSTRAINT dft_cp_unique UNIQUE (cp); ALTER TABLE dft_cp_locations ADD CONSTRAINT enforce_dims_geom CHECK (st_ndims(geom) = 2); ALTER TABLE dft_cp_locations ADD CONSTRAINT enforce_geotype_geom CHECK (geometrytype(geom) = 'POINT'::text OR geom IS NULL); ALTER TABLE dft_cp_locations ADD CONSTRAINT enforce_srid_geom CHECK (st_srid(geom) = 27700); CREATE INDEX dft_cp_locations_geom_gist ON dft_cp_locations USING gist (geom);</pre>
<p>DFT_Annualaveragedailyflow_with_geom DFT_Traffic_with_geom</p>	<p>The tables “DFT_Annualaveragedailyflow” and “DFT_Traffic” do not contain the associated count point geometry for each. The geometry was added, and a new table created for the daily flow and traffic data by executing the following SQL:</p> <p>Annual Average Daily Flow:</p> <pre>DROP TABLE IF EXISTS dft_annualaveragedailyflow_with_geom; CREATE TABLE dft_annualaveragedailyflow_withgeom AS SELECT data.*, cp.cp as cp, cp.srefe as srefe, cp.srefn as srefn, cp.geom as geom FROM dft_annualaveragedailyflow as data, dft_cp_locations as cp WHERE data."CP" = cp.cp; ALTER TABLE dft_annualaveragedailyflow_with_geom ADD CONSTRAINT enforce_dims_geom CHECK (st_ndims(geom) = 2); ALTER TABLE dft_annualaveragedailyflow_with_geom ADD CONSTRAINT enforce_geotype_geom CHECK (geometrytype(geom) = 'POINT'::text OR geom IS NULL); ALTER TABLE dft_annualaveragedailyflow_with_geom ADD CONSTRAINT enforce_srid_geom CHECK (st_srid(geom) = 27700); DROP INDEX IF EXISTS dft_annualaveragedailyflow_withgeom_gist; CREATE INDEX dft_annualaveragedailyflow_withgeom_gist ON dft_annualaveragedailyflow_withgeom USING gist (geom);</pre> <p>Traffic:</p> <pre>DROP TABLE IF EXISTS dft_traffic_withgeom; CREATE TABLE dft_traffic_withgeom AS SELECT data.*, cp.cp as cp, cp.srefe as srefe, cp.srefn as srefn, cp.geom as geom FROM dft_traffic as data, dft_cp_locations as cp WHERE data."CP" = cp.cp; ALTER TABLE dft_traffic_withgeom ADD CONSTRAINT enforce_dims_geom CHECK (st_ndims(geom) = 2); ALTER TABLE dft_traffic_withgeom ADD CONSTRAINT enforce_geotype_geom CHECK (geometrytype(geom) = 'POINT'::text OR geom IS NULL);</pre>

	<p>ALTER TABLE dft_traffic_withgeom ADD CONSTRAINT enforce_srid_geom CHECK (st_srid(geom) = 27700);</p> <p>DROP INDEX IF EXISTS dft_traffic_withgeom;</p> <p>CREATE INDEX dft_traffic_withgeom_gist ON dft_traffic_withgeom USING gist (geom);</p>
Openflights_airlines	<p>The openflights airline data is available as .dat text file. This was converted to a csv file by adding the following column names to the first line of the airline file:</p> <ul style="list-style-type: none"> • airlineid • name • alias • iata_code • icao_code • callsign • country • active
Openflights_airports_uk_only	<p>The openflights airports data is available as a .dat text file. This file was converted to a standard .csv file, by adding the following column names to the first line of each original file:</p> <p>Airport attribute names:</p> <p>airportid name city country iata_faa_c icao_code latitude longitude altitude timezone dst</p> <p>NOTE: This data is still in WGS84 (latitude/longitude). See steps for dataset Openflights_airports_uk_only_osgb36 for converting data to British National Grid</p> <ol style="list-style-type: none"> 1) Open ESRI ArcGIS ArcMap 2) Load the new airport csv file, with headers added 3) Create a shapefile of these airports, using the latitude as the 'Y' component of the coordinate, and longitude as the 'X' component of the coordinate. 4) Export this to a new shapefile e.g.

	<p>openflights_airports_worldwide_wgs84.shp</p> <p>This data was loaded in to the database using FME Workbench. A shapefile reader reads the data in to FME, and a PostGIS writer, writes the data to the database.</p> <p>5) The data was then filtered to extract the UK only airports by running the following SQL:</p> <pre>DROP TABLE IF EXISTS openflights_airports_uk_only; CREATE TABLE openflights_airports_uk_only AS SELECT * FROM openflights_airports_worldwide_wgs84 WHERE country = 'United Kingdom'; ALTER TABLE openflights_airports_uk_only ADD CONSTRAINT openflights_airports_uk_only_prkey PRIMARY KEY (airportid); ALTER TABLE openflights_airports_uk_only ADD CONSTRAINT enforce_dims_geom CHECK (st_ndims(geom) = 2); ALTER TABLE openflights_airports_uk_only ADD CONSTRAINT enforce_geotype_geom CHECK (geometrytype(geom) = 'POINT'::text OR geom IS NULL) DROP INDEX IF EXISTS openflights_airports_uk_only_geom_gist; CREATE INDEX openflights_airports_uk_only_geom_gist ON openflights_airports_uk_only USING gist(geom);</pre>
openflights_airports_uk_only_osgb36	<p>The following SQL converts the table openflights_airports_uk_only to use British National Grid coordinates;</p> <pre>CREATE TABLE openflights_airports_uk_only_OSGB36 AS SELECT * FROM openflights_airports_uk_only; ALTER TABLE openflights_airports_uk_only_OSGB36 ADD COLUMN geom_36 geometry; UPDATE openflights_airports_uk_only_OSGB36 SET geom_36 = ST_Transform(ST_SetSRID(geom, 4326), 27700); ALTER TABLE openflights_airports_uk_only_OSGB36 DROP COLUMN geom; ALTER TABLE openflights_airports_uk_only_OSGB36 RENAME COLUMN geom_36 TO geom; ALTER TABLE openflights_airports_uk_only_OSGB36 ADD CONSTRAINT enforce_geotype_geom CHECK (geometrytype(geom) = 'POINT'::text OR geom IS NULL); ALTER TABLE openflights_airports_uk_only_OSGB36 ADD CONSTRAINT enforce_dims_geom CHECK (st_ndims(geom) = 2); ALTER TABLE openflights_airports_uk_only_OSGB36 ADD CONSTRAINT enforce_srid_geom CHECK (st_srid(geom) = 27700); ALTER TABLE openflights_airports_uk_only_OSGB36 ADD PRIMARY KEY (airportid); DROP INDEX IF EXISTS openflights_airports_uk_only_OSGB36_geom_gist; CREATE INDEX openflights_airports_uk_only_OSGB36_geom_gist ON openflights_airports_uk_only_OSGB36 USING gist(geom);</pre>

<p>Openflights_routes_uk_only</p>	<p>The openflights airports data is available as a .dat text file. This file was converted to a standard .csv file, by adding the following column names to the first line of each original file:</p> <p>Route attribute names:</p> <p>airline airline_id source_airport source_airport_id destination_airport destination_airport_id codeshare stops equipment</p> <p>NOTE: This data is still in WGS84 (latitude/longitude). See steps for dataset Openflights_routes_uk_only_osgb36 for converting data to British National Grid</p> <p>This data was loaded in to the database using FME Workbench. A csv reader reads the data in to FME, and a Postgres writer, writes the data to the database.</p> <p>A function was developed in plpgsql to extract only the routes that begin and end in the UK. This function definition can be found in the appendix section “SQL function and function execution code for extracting UK only routes from openflights data”. The execution code is listed here:</p> <pre>SELECT * FROM subset_openflights_routes_by_country('openflights_routes','openflights_ai rports_uk_only', 'openflights_routes_uk_only') f(airline char(4), airlineid integer, source_airport char(4), source_airport_id integer, destination_airport char(4), destination_airport_id integer, codeshare char(2), stops float, equipment char(44), source_airport_geom geometry, destination_airport_geom geometry, source_destination_route_geom geometry);</pre>
-----------------------------------	---

<p>openflights_routes_uk_only_osgb36</p>	<p>The following SQL converts the table openflights_airports_uk_only to use British National Grid coordinates;</p> <pre> DROP TABLE IF EXISTS openflights_routes_uk_only_OSGB36; CREATE TABLE openflights_routes_uk_only_OSGB36 AS SELECT * FROM openflights_routes_uk_only; ALTER TABLE openflights_routes_uk_only_OSGB36 ADD COLUMN source_destination_route_geom_36 geometry; ALTER TABLE openflights_routes_uk_only_OSGB36 ADD COLUMN source_airport_geom_36 geometry; ALTER TABLE openflights_routes_uk_only_OSGB36 ADD COLUMN destination_airport_geom_36 geometry; UPDATE openflights_routes_uk_only_OSGB36 SET source_destination_route_geom_36 = ST_Transform(ST_SetSRID(source_destination_route_geom, 4326), 27700); UPDATE openflights_routes_uk_only_OSGB36 SET source_airport_geom_36 = ST_Transform(ST_SetSRID(source_airport_geom, 4326), 27700); UPDATE openflights_routes_uk_only_OSGB36 SET destination_airport_geom_36 = ST_Transform(ST_SetSRID(destination_airport_geom, 4326), 27700); ALTER TABLE openflights_routes_uk_only_OSGB36 DROP COLUMN source_destination_route_geom; ALTER TABLE openflights_routes_uk_only_OSGB36 RENAME COLUMN source_destination_route_geom_36 TO source_destination_route_geom; ALTER TABLE openflights_routes_uk_only_OSGB36 DROP COLUMN source_airport_geom; ALTER TABLE openflights_routes_uk_only_OSGB36 RENAME COLUMN source_airport_geom_36 TO source_airport_geom; ALTER TABLE openflights_routes_uk_only_OSGB36 DROP COLUMN destination_airport_geom; ALTER TABLE openflights_routes_uk_only_OSGB36 RENAME COLUMN destination_airport_geom_36 TO destination_airport_geom; ALTER TABLE openflights_routes_uk_only_OSGB36 ADD CONSTRAINT enforce_geotype_source_airport_geom CHECK (geometrytype(source_airport_geom) = 'POINT'::text OR source_airport_geom IS NULL); ALTER TABLE openflights_routes_uk_only_OSGB36 ADD CONSTRAINT enforce_dims_source_airport_geom CHECK (st_ndims(source_airport_geom) = 2); ALTER TABLE openflights_routes_uk_only_OSGB36 ADD CONSTRAINT enforce_srid_source_airport_geom CHECK (st_srid(source_airport_geom) = 27700); ALTER TABLE openflights_routes_uk_only_OSGB36 ADD CONSTRAINT enforce_geotype_destination_airport_geom CHECK (geometrytype(destination_airport_geom) = 'POINT'::text OR destination_airport_geom IS NULL); ALTER TABLE openflights_routes_uk_only_OSGB36 ADD CONSTRAINT enforce_dims_destination_airport_geom CHECK (st_ndims(destination_airport_geom) = 2); ALTER TABLE openflights_routes_uk_only_OSGB36 ADD CONSTRAINT enforce_srid_destination_airport_geom CHECK (st_srid(destination_airport_geom) = 27700); </pre>
--	--

	<pre> ALTER TABLE openflights_routes_uk_only_OSGB36 ADD CONSTRAINT enforce_geotype_source_destination_route_geom CHECK (geometrytype(source_destination_route_geom) = 'LINESTRING'::text OR source_destination_route_geom IS NULL); ALTER TABLE openflights_routes_uk_only_OSGB36 ADD CONSTRAINT enforce_dims_source_destination_route_geom CHECK (st_ndims(source_destination_route_geom) = 2); ALTER TABLE openflights_routes_uk_only_OSGB36 ADD CONSTRAINT enforce_srid_source_destination_route_geom CHECK (st_srid(source_destination_route_geom) = 27700); DROP INDEX IF EXISTS openflights_routes_uk_only_OSGB36_geom_gist; CREATE INDEX openflights_routes_uk_only_OSGB36_geom_gist ON openflights_routes_uk_only_OSGB36 USING gist(source_airport_geom); DROP INDEX IF EXISTS openflights_routes_uk_only_OSGB36_geom_gist; CREATE INDEX openflights_routes_uk_only_OSGB36_geom_gist ON openflights_routes_uk_only_OSGB36 USING gist(destination_airport_geom); DROP INDEX IF EXISTS openflights_routes_uk_only_OSGB36_geom_gist; CREATE INDEX openflights_routes_uk_only_OSGB36_geom_gist ON openflights_routes_uk_only_OSGB36 USING gist(source_destination_route_geom); </pre>
--	--

Figure 16 - processing and upload procedures for transport data

Ordnance Survey Points of Interest

Table prefix/table name	Process Description
POI_Categories POI_Classifications POI_Groups POI_Categories_v31 POI_Classes_v31 POI_Groups_v31	These tables were created using FME Workbench. A Microsoft Access feature type reader was used to read the input tables from a Microsoft Access database, then a Postgres feature writer was used to write the data to the database.
Points_of_interest	<p>This data was originally supplied by the Ordnance Survey for the North West region only, as a .csv file. The data was loaded in to the “points of interest” database using FME Workbench. A CSV reader was used to read the input data, whilst a PostGIS writer was used to write the data to the database.</p> <p>The original csv file stored the coordinates of each point of interest in the attributes “ITN_Easting”, and “ITN_Northing”. To convert these attributes to a point geometry within FME, a 2DPointReplacer Transformer was used. This assigns the ITN_Easting attribute as the X component, and ITN_Northing attribute as the Y component. Further information can be found here: http://docs.safe.com/fme/2010/html/FME_Transformers/content/transformers/2dpointreplacer.htm.</p>
Derived_poi_electrical_features Derived_poi_energy_production Derived_poi_fuel_distributors_suppliers Derived_poi_gas_features Derived_poi_london_underground_entrances Derived_poi_oil_gas_extraction_refine_manufacture Derived_poi_pipelines Derived_poi_railway_stations_junctions_halts Derived_poi_recycling_centres Derived_poi_refuse_disposal_facilities Derived_poi_signalling_facilities Derived_poi_underground_network_stations Derived_poi_waste_storage_processing_disposal Derived_poi_waste_storage_processing_disposal_sewage_features Derived_poi_waste_storage_processing_disposal_sludge_features Derived_poi_waste_storage_processing_disposal_slurry_features	<p>These derived data tables were generated by filtering the “Points_of_interest” data table using the pointx_classification_code attribute, and varying the value to extract the different feature types:</p> <p>e.g.</p> <pre> DROP TABLE IF EXISTS derived_poi_electrical_features; CREATE TABLE derived_poi_electrical_features AS SELECT * FROM points_of_interest WHERE pointx_classification_code = '06340433'; ALTER TABLE derived_poi_electrical_features ADD CONSTRAINT "enforce_srid_geom" CHECK (st_srid(geom) = 27700); ALTER TABLE derived_poi_electrical_features ADD CONSTRAINT "enforce_geotype_geom" CHECK (geometrytype(geom) = 'POINT':text OR geom IS NULL); ALTER TABLE derived_poi_electrical_features ADD CONSTRAINT "enforce_dims_gemo" CHECK (st_ndims(geom) = 2); Electrical_features = 06340433 Energy_Production = 07410534 Fuel_Distributors_Suppliers = 09480766 Gas_Features = 06340437 London_Underground_Entrances = 10570794 Oil_Gas_Extraction_Refine_Manufacture = 07380501 Pipelines = 07410538 Railway, Stations, Junctions, Halts = 10570738 Recycling Centres = 06340462 </pre>

	<p>Refuse Disposal Facilities = 06340440 Signalling Facilities = 10540740 Underground network stations = 10570761 Waste Storage Processing Disposal (All) = 06340441</p> <p>In order to produce the filtered waste storage processing and disposal tables for sewage, sludge and slurry respectively, a SQL filter was applied e.g.</p> <pre>CREATE TABLE derived_poi_waste_storage_processing_disposal_sewage_features AS SELECT * FROM derived_poi_waste_storage_processing_disposal WHERE name ILIKE '%sew%';</pre> <pre>CREATE TABLE derived_poi_waste_storage_processing_disposal_slurry_features AS SELECT * FROM derived_poi_waste_storage_processing_disposal WHERE name ILIKE '%slur%';</pre> <pre>CREATE TABLE derived_poi_waste_storage_processing_disposal_sludge_features AS SELECT * FROM derived_poi_waste_storage_processing_disposal WHERE name ILIKE '%slud%';</pre>
--	---

Figure 17 - processing and upload procedures for Ordnance Survey Points of Interest data (North West coverage only)

Ordnance Survey Vectormap District

Table prefix/table name	Process Description
AdministrativeBoundary Airport Building ElectricityTransmissionLine Foreshore Glasshouse HeritageSite Land MotorwayJunction NamedPlace Ornament PublicAmenity RailwayStation RailwayTrack RailwayTunnel Road RoadTunnel Spotheight SurfaceWater_Area SurfaceWater_Line TidalBoundary TidalWater Woodland	<p>The Ordnance Survey Vectormap District data was supplied as a series of ESRI Shapefiles, split up by feature type, and then subsequently by which 100km National Grid tile. For example for the NS tile, a separate shapefile per feature type exists, provided that a feature of that type lies within that tile.</p> <p>NS_AdministrativeBoundary NS_Airport NS_Building NS_ElectricityTransmissionLine NS_Foreshore NS_Glasshouse NS_HeritageSite NS_Land NS_MotorwayJunction NS_NamedPlace NS_Ornament NS_PublicAmenity NS_RailwayStation NS_RailwayTrack NS_RailwayTunnel NS_Spotheight NS_SurfaceWater_Area NS_SurfaceWater_Line NS_TidalBoundary NS_TidalWater NS_Woodland</p> <p>The data was organised on file system into folders representing the different feature types, for all tiles. By organising the data like this, it was possible to use FME Workbench to read all the shapefiles, for all tiles from a particular feature type folder, as a single feature type. Subsequently a PostGIS writer feature type was used to write each feature type to the database as a separate table.</p> <p>A single table per feature type exists in the database.</p> <p>No processing on the data was performed prior to uploading the data to the database.</p>

Figure 18 - processing and upload procedure for Ordnance Survey Vectormap District data

Ordnance Survey MasterMap

Table prefix/table name	Process Description
AddressPoint BoundaryLine FerryLink FerryNode FerryTerminal InformationPoint Road RoadLink RoadLinkInformation RoadNode RoadNodeInformation RoadRouteInformation TopographicArea TopographicLine TopographicPoint	<p>The Ordnance Survey MasterMap data was supplied in Geographic MarkUp Language (GML) format, on a series of 12 DVDs.</p> <p>Disc by disc FME to PostGIS</p> <p>Each disc of data was loaded in to the database separately using FME Workbench. A GML Feature Type reader was used to read each disc of data, and a subsequent PostGIS Feature Type writer was then used to write the data to the database.</p> <p>Each feature type was prefixed with the name of the disc from which it came e.g.</p> <p>Disc1_AddressPoint – stores all AddressPoint data from Disc 1</p> <p>A separate FME Workbench workspace was created for each disc of data to be uploaded.</p> <p>Once uploaded, the different feature types of data were combined to produce a single table of data for that particular feature type e.g.</p> <pre> CREATE TABLE "AddressPoint" AS SELECT * FROM "Disc1_AddressPoint" UNION ALL SELECT * FROM "Disc2_AddressPoint" UNION ALL SELECT * FROM "Disc3_sd_AddressPoint" UNION ALL SELECT * FROM "Disc4_se_AddressPoint" UNION ALL SELECT * FROM "Disc5_se_AddressPoint" UNION ALL SELECT * FROM "Disc6_se_AddressPoint" UNION ALL SELECT * FROM "Disc6_sj_AddressPoint" UNION ALL SELECT * FROM "Disc7_sj_AddressPoint" UNION ALL SELECT * FROM "Disc8_sj_AddressPoint" UNION ALL SELECT * FROM "Disc9_sj_AddressPoint" UNION ALL SELECT * FROM "Disc10_sj_AddressPoint" UNION ALL SELECT * FROM "Disc10_sk_AddressPoint" UNION ALL SELECT * FROM "Disc11_sk_AddressPoint" UNION ALL SELECT * FROM "Disc12_sk_AddressPoint" ALTER TABLE "AddressPoint" ADD CONSTRAINT enforce_dims_geom CHECK (st_ndims(geom) = 2); ALTER TABLE "AddressPoint" ADD CONSTRAINT enforce_srid_geom CHECK (st_srid(geom) = 27700); CREATE INDEX "AddressPoint_geom_ind" ON "AddressPoint" USING gist (geom); </pre>

Figure 19 - processing and upload procedure overview for North West coverage of Ordnance Survey MasterMap data

Appendix:

plpgsql function code for ws2_build_od_table

```
CREATE OR REPLACE FUNCTION ws2_build_od_table(character varying, character varying, character varying, integer, character varying)
RETURNS integer AS
$BODY$
DECLARE

    --table containing zones / boundaries (must contain ZoneCode and ZoneName columns)
    zone_table_name ALIAS for $1;
    origin_zone_table_record RECORD;
    destination_zone_table_record RECORD;

    --table containing zone centroids and nearest feature geometry (must contain the following columns)
    --id serial/integer
    --"name" text/char var
    --"ZoneCode" unique code per zone
    --distance - distance between zone centroid and feature of interest
    --feature_geom - feature geometry
    --centroid_geom - geometry of centroid of zone
    --nearest_feature_to_centroid_line - line geometry linking feature_geom and centroid_geom
    nearest_feature_to_zone_centroid_table_to_query ALIAS for $2;
    origin_nearest_feature_record RECORD;
    destination_nearest_feature_record RECORD;

    --table containing origin and destination passenger counts
    --must have first column as origin, with origins listed
    --column names are destinations
    travel_to_work_table ALIAS for $3;

    --srid
    srid ALIAS for $4;

    --output table
    output_table_name ALIAS for $5;

    od_table_row_count integer := 0;

    origin_zone_code character varying := '';
    origin_zone_name character varying := '';

    origin_nearest_feature_name character varying := '';
    origin_nearest_feature_zonecode character varying := '';
    origin_distance_to_nearest_feature_from_zone_centroid double precision := 0.0;
    origin_feature_geom text := '';
    origin_zone_centroid_geom text := '';
    origin_nearest_feature_to_centroid_line_geom text := '';

    destination_zone_code character varying := '';
    destination_zone_name character varying := '';

    destination_nearest_feature_name character varying := '';
    destination_nearest_feature_zonecode character varying := '';
    destination_distance_to_nearest_feature_from_zone_centroid double precision := 0.0;
    destination_feature_geom text := '';
    destination_zone_centroid_geom text := '';
    destination_nearest_feature_to_centroid_line_geom text := '';

    passenger_count integer := 0;

BEGIN

    --drop output table
```

```

EXECUTE 'DROP TABLE IF EXISTS ' || quote_ident(output_table_name);

--create result table
EXECUTE 'CREATE TABLE ' || quote_ident(output_table_name) || ' (id serial NOT NULL, "Origin_Feature_Name" character varying,
"Origin_Zone_Name" character varying, "Origin_Zone_Code" character varying, "Origin_Feature_To_Zone_Centroid_Distance" double
precision, "Destination_Feature_Name" character varying, "Destination_Zone_Name" character varying, "Destination_Zone_Code" character
varying, "Destination_Feature_To_Zone_Centroid_Distance" double precision, "Origin_Feature_geom" geometry, "Origin_Zone_geom"
geometry, "Origin_Feature_To_Zone_Centroid_geom" geometry, "Destination_Feature_geom" geometry, "Destination_Zone_geom"
geometry, "Destination_Feature_To_Zone_Centroid_geom" geometry, "JTW_Count" integer);

--loop origin zones
FOR origin_zone_table_record IN EXECUTE 'SELECT "ZoneCode", "ZoneName" FROM ' || quote_ident(zone_table_name) || ' ORDER BY
"ZoneCode", "ZoneName"' LOOP

    --set origin zone code and names
    origin_zone_code := origin_zone_table_record."ZoneCode";
    origin_zone_name := origin_zone_table_record."ZoneName";

    --check for null origin zones
    IF ((origin_zone_code IS NULL OR origin_zone_code = '') OR (origin_zone_name IS NULL OR origin_zone_name = '')) THEN
        RAISE NOTICE 'EMPTY ORIGIN ZONE';
        CONTINUE;
    END IF;

    --check for apostrophe in zonecode
    IF strpos(origin_zone_code, '''') > -1 THEN
        origin_zone_code := replace(origin_zone_code, '''', ''''');
    END IF;

    --check for apostrophe in zonenumber
    IF strpos(origin_zone_name, '''') > -1 THEN
        origin_zone_name := replace(origin_zone_name, '''', ''''');
    END IF;

    --loop origin features
    FOR origin_nearest_feature_record IN EXECUTE 'SELECT "name" as origin_nearest_feature_name, "ZoneCode" as
"ZoneCode", distance as origin_distance_to_nearest_feature_from_zone_centroid, ST_AsText(feature_geom) as origin_feature_geom,
ST_AsText(centroid_geom) as origin_zone_centroid_geom, ST_AsText(nearest_feature_to_centroid_line_geom) as
origin_nearest_feature_to_centroid_line_geom FROM ' || quote_ident(nearest_feature_to_zone_centroid_table_to_query) || ' WHERE
"ZoneCode" = ' || quote_literal(origin_zone_code) || ' ORDER BY "ZoneCode"' LOOP

        --set origin feature attributes
        origin_nearest_feature_name := origin_nearest_feature_record.origin_nearest_feature_name;
        origin_nearest_feature_zonecode := origin_nearest_feature_record."ZoneCode";
        origin_distance_to_nearest_feature_from_zone_centroid :=
origin_nearest_feature_record.origin_distance_to_nearest_feature_from_zone_centroid;
        origin_feature_geom := origin_nearest_feature_record.origin_feature_geom;
        origin_zone_centroid_geom := origin_nearest_feature_record.origin_zone_centroid_geom;
        origin_nearest_feature_to_centroid_line_geom :=
origin_nearest_feature_record.origin_nearest_feature_to_centroid_line_geom;

        --check if origin feature is empty or null
        IF ((origin_nearest_feature_name IS NULL) OR (origin_nearest_feature_name = '')) THEN

            RAISE NOTICE 'EMPTY ORIGIN FEATURE';
            CONTINUE;
        END IF;

        --check if origin distance is null or empty
        IF ((origin_distance_to_nearest_feature_from_zone_centroid IS NULL) or
(origin_distance_to_nearest_feature_from_zone_centroid = '')) THEN
            origin_distance_to_nearest_feature_from_zone_centroid := 0;
        END IF;
    
```

```

--check if origin feature name has ""
IF strpos(origin_nearest_feature_name, "") > -1 THEN
    origin_nearest_feature_name := replace(origin_nearest_feature_name, "", "");
END IF;

--loop destination zones
FOR destination_zone_table_record IN EXECUTE 'SELECT "ZoneCode", "ZoneName" FROM
' || quote_ident(zone_table_name) || ' ORDER BY "ZoneCode", "ZoneName"' LOOP

    --set destination zone code and name
    destination_zone_code := destination_zone_table_record."ZoneCode";
    destination_zone_name := destination_zone_table_record."ZoneName";

    --check for null destination code and names
    IF ((destination_zone_code IS NULL OR destination_zone_code = '') OR (destination_zone_name IS
NULL OR destination_zone_name = '')) THEN
        RAISE NOTICE 'EMPTY DESTINATION ZONE';
        CONTINUE;
    END IF;

    --check for apostrophe in zonecode
    IF strpos(destination_zone_code, "'") > -1 THEN
        destination_zone_code := replace(destination_zone_code, "'", '');
    END IF;

    --check for apostrophe in zonename
    IF strpos(destination_zone_name, "'") > -1 THEN
        destination_zone_name := replace(destination_zone_name, "'", '');
    END IF;

    --loop destination features
    FOR destination_nearest_feature_record IN EXECUTE 'SELECT "name" as
destination_nearest_feature_name, "ZoneCode" as "ZoneCode", distance as destination_distance_to_nearest_feature_from_zone_centroid,
ST_AsText(feature_geom) as destination_feature_geom, ST_AsText(centroid_geom) as destination_zone_centroid_geom,
ST_AsText(nearest_feature_to_centroid_line_geom) as destination_nearest_feature_to_centroid_line_geom FROM
' || quote_ident(nearest_feature_to_zone_centroid_table_to_query) || ' WHERE "ZoneCode" = ' || quote_literal(destination_zone_code) || '
ORDER BY "ZoneCode"' LOOP

        --set destination feature attributes
        destination_nearest_feature_name :=
destination_nearest_feature_record.destination_nearest_feature_name;
        destination_nearest_feature_zonecode := destination_nearest_feature_record."ZoneCode";
        destination_distance_to_nearest_feature_from_zone_centroid :=
destination_nearest_feature_record.destination_distance_to_nearest_feature_from_zone_centroid;
        destination_feature_geom :=
destination_nearest_feature_record.destination_feature_geom;
        destination_zone_centroid_geom :=
destination_nearest_feature_record.destination_zone_centroid_geom;
        destination_nearest_feature_to_centroid_line_geom :=
destination_nearest_feature_record.destination_nearest_feature_to_centroid_line_geom;

        --check if destination feature is empty or null
        IF ((destination_nearest_feature_name IS NULL) OR (destination_nearest_feature_name = ''))
THEN
            RAISE NOTICE 'EMPTY DESTINATION FEATURE';
            CONTINUE;
        END IF;

        --check if destination distance is null or empty
        IF ((destination_distance_to_nearest_feature_from_zone_centroid IS NULL) or
(destination_distance_to_nearest_feature_from_zone_centroid = '')) THEN
            destination_distance_to_nearest_feature_from_zone_centroid := 0;

```

```

END IF;

--check if destination feature name has ""
IF strpos(destination_nearest_feature_name, "") > -1 THEN
    destination_nearest_feature_name := replace(destination_nearest_feature_name,
"", "");

END IF;

--set passenger count
EXECUTE 'SELECT ' || quote_ident(destination_zone_code) || ' FROM
' || quote_ident(travel_to_work_table) || ' WHERE "Origin" = ' || quote_literal(origin_zone_code) INTO passenger_count;

--insert into output table
EXECUTE 'INSERT INTO ' || quote_ident(output_table_name) || ' ("Origin_Feature_Name",
"Origin_Zone_Name", "Origin_Zone_Code", "Origin_Feature_To_Zone_Centroid_Distance", "Destination_Feature_Name",
"Destination_Zone_Name", "Destination_Zone_Code", "Destination_Feature_To_Zone_Centroid_Distance", "Origin_Feature_geom",
"Origin_Zone_geom", "Origin_Feature_To_Zone_Centroid_geom", "Destination_Feature_geom", "Destination_Zone_geom",
"Destination_Feature_To_Zone_Centroid_geom", "JTW_Count") VALUES (' || quote_literal(origin_nearest_feature_name) || ',
' || quote_literal(origin_zone_name) || ', ' || quote_literal(origin_zone_code) || ', ' || origin_distance_to_nearest_feature_from_zone_centroid || ',
' || quote_literal(destination_nearest_feature_name) || ', ' || quote_literal(destination_zone_name) || ', ' || quote_literal(destination_zone_code) || ',
' || destination_distance_to_nearest_feature_from_zone_centroid || ', ST_GeomFromText(' || quote_literal(origin_feature_geom) || ',
' || srid || '), ST_GeomFromText(' || quote_literal(origin_zone_centroid_geom) || ', ' || srid || '),
ST_GeomFromText(' || quote_literal(origin_nearest_feature_to_centroid_line_geom) || ', ' || srid || '),
ST_GeomFromText(' || quote_literal(destination_feature_geom) || ', ' || srid || '),
ST_GeomFromText(' || quote_literal(destination_zone_centroid_geom) || ', ' || srid || '),
ST_GeomFromText(' || quote_literal(destination_nearest_feature_to_centroid_line_geom) || ', ' || srid || '), ' || passenger_count || ');

END LOOP;

END LOOP;

END LOOP;

EXECUTE 'SELECT COUNT(*) FROM ' || quote_ident(output_table_name) INTO od_table_row_count;

--geometry column constraints (origin_feature_geom, origin_zone_geom, origin_feature_to_zone_centroid_geom,
destination_feature_geom, destination_zone_geom, destination_feature_to_zone_centroid_geom

--dims
--origin
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_dims_origin_feature_geom CHECK
(st_ndims(origin_feature_geom) = 2)';
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_dims_origin_zone_geom CHECK
(st_ndims(origin_zone_geom) = 2)';
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT
enforce_dims_origin_feature_to_zone_centroid_geom CHECK (st_ndims(origin_feature_to_zone_centroid_geom) = 2)';

--destination
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_dims_destination_feature_geom CHECK
(st_ndims(destination_feature_geom) = 2)';
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_dims_destination_zone_geom CHECK
(st_ndims(destination_zone_geom) = 2)';
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT
enforce_dims_destination_feature_to_zone_centroid_geom CHECK (st_ndims(destination_feature_to_zone_centroid_geom) = 2)';

--geotype
--origin
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_geotype_origin_feature_geom CHECK
(geometrytype(origin_feature_geom) = "POINT"::text OR origin_feature_geom IS NULL)';
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_geotype_origin_zone_geom CHECK

```

```
(geometrytype(origin_zone_geom) = "POINT"::text OR origin_zone_geom IS NULL);
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT
enforce_geotype_origin_feature_to_zone_centroid_geom CHECK (geometrytype(origin_feature_to_zone_centroid_geom) =
"LINESTRING"::text OR origin_feature_to_zone_centroid_geom IS NULL)';

--destination
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_geotype_destination_feature_geom
CHECK (geometrytype(destination_feature_geom) = "POINT"::text OR destination_feature_geom IS NULL)';
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_geotype_destination_zone_geom CHECK
(geometrytype(destination_zone_geom) = "POINT"::text OR destination_zone_geom IS NULL)';
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT
enforce_geotype_destination_feature_to_zone_centroid_geom CHECK (geometrytype(destination_feature_to_zone_centroid_geom) =
"LINESTRING"::text OR destination_feature_to_zone_centroid_geom IS NULL)';

--srid
--origin
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_srid_origin_feature_geom CHECK
(st_srid(origin_feature_geom) = ' || srid || ');
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_srid_origin_zone_geom CHECK
(st_srid(origin_zone_geom) = ' || srid || ');
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT
enforce_srid_origin_feature_to_zone_centroid_geom CHECK (st_srid(origin_feature_to_zone_centroid_geom) = ' || srid || ');

--destination
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_srid_destination_feature_geom CHECK
(st_srid(destination_feature_geom) = ' || srid || ');
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_srid_destination_zone_geom CHECK
(st_srid(destination_zone_geom) = ' || srid || ');
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT
enforce_srid_destination_feature_to_zone_centroid_geom CHECK (st_srid(destination_feature_to_zone_centroid_geom) = ' || srid || ');

RETURN od_table_row_count;

END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
ALTER FUNCTION ws2_build_od_table(character varying, character varying, character varying, integer, character varying) OWNER TO postgres;
```

Figure 20 - plpgsql function code to build an origin-destination table, creating an output table containing both the origin zone and origin feature, as well as the destination zone and the destination feature

plpgsql function code for ws2_nearest_feature_to_centroid

```
CREATE OR REPLACE FUNCTION ws2_nearest_feature_to_centroid(character varying, character varying, character varying, character varying,
character varying, character varying, integer, character varying)
RETURNS character varying AS
$BODY$
DECLARE
    --name of table containing centroids of districts/wards/government office regions
    centroid_table_name ALIAS for $1;
    --name of column in centroid_table_name containing centroids of regions
    centroid_geometry_column_name ALIAS for $2;
    --distinct column name of centroid table to uniquely identify each (returned in output table as "ZoneCode")
    centroid_distinct_column_name ALIAS for $3;

    --name of table containing features, of which one will be selected as "nearest"
    feature_table_name ALIAS for $4;

    --name of column in feature_table_name containing POINT/LINESTRING/POLYGON features of interest
    feature_geometry_column_name ALIAS for $5;
    feature_geometry_type character varying;
    feature_dims integer := 0;

    --distinct column name of feature table to uniquely identify each (returned in output table as "name")
    feature_distinct_column_name ALIAS for $6;

    --srid code of output table geometry
    srid ALIAS for $7;

    --name for generated output table
    output_table_name ALIAS for $8;

    centroid_feature_line text;
    centroid_table_record RECORD;
    result_record RECORD;

    feature_geom_srid integer := 0;
    centroid_geom_srid integer := 0;

BEGIN

    --DROP OUTPUT TABLE
    EXECUTE 'DROP TABLE IF EXISTS ' || quote_ident(output_table_name);

    --CREATE OUTPUT TABLE
    EXECUTE 'CREATE TABLE ' || quote_ident(output_table_name) || ' (id serial NOT NULL,
    ' || quote_ident(feature_distinct_column_name) || ' text, ' || quote_ident(centroid_distinct_column_name) || ' text, distance double precision,
    feature_geom geometry, centroid_geom geometry, nearest_feature_to_centroid_line_geom geometry)';

    --define geometry type for features
    EXECUTE 'SELECT ST_NDims(' || quote_ident(feature_geometry_column_name) || ') FROM ' || quote_ident(feature_table_name) INTO
    feature_dims;

    --LOOP DISTRICT CENTROID TABLE
    FOR centroid_table_record IN EXECUTE 'SELECT DISTINCT(' || quote_ident(centroid_distinct_column_name) || ') as "ZoneCode" FROM
    ' || quote_ident(centroid_table_name) || ' ORDER BY ' || quote_ident(centroid_distinct_column_name) || ' ASC' LOOP

        EXECUTE 'SELECT feature.' || quote_ident(feature_distinct_column_name) || ' as "name",
        centroid.' || quote_ident(centroid_distinct_column_name) || ' as "ZoneCode",
        st_distance(feature.' || quote_ident(feature_geometry_column_name) || ', centroid.' || quote_ident(centroid_geometry_column_name) || ') as
        distance, ST_AsText(feature.' || quote_ident(feature_geometry_column_name) || ') as feature_geom,
        ST_AsText(centroid.' || quote_ident(centroid_geometry_column_name) || ') as centroid_geom FROM ' || quote_ident(feature_table_name) || '
        as feature, ' || quote_ident(centroid_table_name) || ' as centroid WHERE centroid.' || quote_ident(centroid_distinct_column_name) || ' =
        ' || quote_literal(centroid_table_record."ZoneCode") || ' GROUP BY feature.' || quote_ident(feature_distinct_column_name) || ',
        centroid.' || quote_ident(centroid_distinct_column_name) || ', feature.' || quote_ident(feature_geometry_column_name) || ',
```

```

centroid.'||quote_ident(centroid_geometry_column_name)||' ORDER BY feature.'||quote_ident(feature_geometry_column_name)||' <->
centroid.'||quote_ident(centroid_geometry_column_name)||' LIMIT 1' INTO result_record;

IF result_record.centroid_geom != 'NULL' AND result_record.centroid_geom IS NOT NULL THEN

    EXECUTE 'SELECT ST_AsText(ST_MakeLine(ST_GeomFromText('||quote_literal(result_record.centroid_geom)||',
'||srid||'),ST_GeomFromText('||quote_literal(result_record.feature_geom)||','||srid||')))' INTO centroid_feature_line;

    EXECUTE 'INSERT INTO '||quote_ident(output_table_name)||'
('||quote_ident(feature_distinct_column_name)||','||quote_ident(centroid_distinct_column_name)||', distance, feature_geom,
centroid_geom, nearest_feature_to_centroid_line_geom) VALUES ('||quote_literal(result_record."name")||',
'||quote_literal(result_record."ZoneCode")||', '||result_record.distance||',
ST_GeomFromText('||quote_literal(result_record.feature_geom)||','||srid||'),
ST_GeomFromText('||quote_literal(result_record.centroid_geom)||','||srid||'),
ST_GeomFromText('||quote_literal(centroid_feature_line)||','||srid||'))';

ELSE

    EXECUTE 'INSERT INTO '||quote_ident(output_table_name)||'
('||quote_ident(centroid_distinct_column_name)||') VALUES ('||quote_literal(result_record."ZoneCode")||')';

END IF;

END LOOP;

--ADD CONSTRAINTS ON GEOMETRY COLUMNS
--DIMS
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT
'||quote_ident('enforce_dims_'||centroid_geometry_column_name)||' CHECK (st_ndims(centroid_geom) = 2);
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT nearest_feature_to_centroid_line_geom CHECK
(st_ndims(nearest_feature_to_centroid_line_geom) = 2);
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT
'||quote_ident('enforce_dims_'||feature_geometry_column_name)||' CHECK (st_ndims(feature_geom) = '||feature_dims||');

--GEOTYPE
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT
'||quote_ident('enforce_geotype_'||centroid_geometry_column_name)||' CHECK (geometrytype(centroid_geom) = "POINT"::text OR
centroid_geom IS NULL);
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT
enforce_geotype_nearest_feature_to_centroid_line_geom CHECK (geometrytype(nearest_feature_to_centroid_line_geom) =
"LINESTRING"::text OR nearest_feature_to_centroid_line_geom IS NULL);

--SRID
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT
'||quote_ident('enforce_srid_'||centroid_geometry_column_name)||' CHECK (st_srid(centroid_geom) = '||srid||');
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT
enforce_srid_nearest_feature_to_centroid_line_geom CHECK (st_srid(nearest_feature_to_centroid_line_geom) = '||srid||');
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT
'||quote_ident('enforce_srid_'||feature_geometry_column_name)||' CHECK (st_srid(feature_geom) = '||srid||');

--feature_geom spatial index
EXECUTE 'DROP INDEX IF EXISTS '||quote_ident(output_table_name)||'_feature';
EXECUTE 'CREATE INDEX '||quote_ident(output_table_name)||'_feature' ON '||quote_ident(output_table_name)||' USING
gist(feature_geom);

--centroid_geom spatial index
EXECUTE 'DROP INDEX IF EXISTS '||quote_ident(output_table_name)||'_centroid';
EXECUTE 'CREATE INDEX '||quote_ident(output_table_name)||'_centroid' ON '||quote_ident(output_table_name)||' USING
gist(centroid_geom);

--nearest_feature_to_centroid_line spatial index
EXECUTE 'DROP INDEX IF EXISTS '||quote_ident(output_table_name)||'_near_feature_to_centroid';

```

```
EXECUTE 'CREATE INDEX '||quote_ident(output_table_name)||'_near_feature_to_centroid')||' ON
'||quote_ident(output_table_name)||' USING gist(nearest_feature_to_centroid_line_geom)';

RETURN quote_ident(output_table_name);

END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
ALTER FUNCTION ws2_nearest_feature_to_centroid(character varying, character varying, character varying, character varying, character
varying, character varying, integer, character varying) OWNER TO postgres;
```

Figure 21 - plpgsql function code to calculate nearest feature to a zone centroid, whether inside or outside the feature is inside or outside the given zone geometry

plpgSQL function code for ws2_nearest_feature_to_centroid_in_zone

```
CREATE OR REPLACE FUNCTION ws2_nearest_feature_to_centroid_in_zone(character varying, character varying, character varying, character
varying, character varying, integer, character varying)
RETURNS character varying AS
$BODY$
DECLARE
    --table containing features of interest and matched zone boundaries and centroids
    features_per_zone_table_name ALIAS for $1;

    --geometry column of above table containing the feature geometry of interest
    feature_geometry_column_name ALIAS for $2;

    --geometry column of above table containing the zone centroid geometry
    zone_centroid_geometry_column_name ALIAS for $3;

    --column uniquely identifying each feature
    feature_unique_column_name ALIAS for $4;

    --column uniquely identifying each zone and zone centroid
    zone_unique_column_name ALIAS for $5;

    --srid of input feature and zone geometries, and output table (must match up)
    srid ALIAS for $6;

    --name of table to create to store results
    output_table_name ALIAS for $7;

    --store feature geometry settings
    feature_dims integer := 0;
    feature_geometry_type text := "";

BEGIN
    --set feature geometry settings
    EXECUTE 'SELECT ST_NDims('||quote_ident(feature_geometry_column_name)||') FROM
'||quote_ident(features_per_zone_table_name) INTO feature_dims;
    --EXECUTE 'SELECT GeometryType('||quote_ident(feature_geometry_column_name)||') FROM
'||quote_ident(features_per_zone_table_name) INTO feature_geometry_type;

    --DROP TABLE
    EXECUTE 'DROP TABLE IF EXISTS '||quote_ident(output_table_name);

    --CREATE TABLE
    EXECUTE 'CREATE TABLE '||quote_ident(output_table_name)||' AS SELECT * FROM (WITH A_QUERY AS (SELECT
'||quote_ident(zone_unique_column_name)||', min(distance) as min_distance, count(*) as feature_count FROM
'||quote_ident(features_per_zone_table_name)||' GROUP BY '||quote_ident(zone_unique_column_name)||' ORDER BY
'||quote_ident(zone_unique_column_name)||') SELECT
'||quote_ident(features_per_zone_table_name)||'. '||quote_ident(feature_unique_column_name)||' as "name",
'||quote_ident(features_per_zone_table_name)||'. '||quote_ident(zone_unique_column_name)||' as "ZoneCode",
'||quote_ident(features_per_zone_table_name)||'.distance,
'||quote_ident(features_per_zone_table_name)||'. '||quote_ident(feature_geometry_column_name)||' as feature_geom,
'||quote_ident(features_per_zone_table_name)||'. '||quote_ident(zone_centroid_geometry_column_name)||' as zone_centroid_geom,
A_QUERY.feature_count FROM '||quote_ident(features_per_zone_table_name)||', A_QUERY WHERE
'||quote_ident(features_per_zone_table_name)||'.distance = A_QUERY.min_distance ORDER BY
'||quote_ident(features_per_zone_table_name)||'. '||quote_ident(zone_unique_column_name)||') as QUERY';

    --add a column to store the line geometry between the closest feature and zone centroid
    EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD COLUMN nearest_feature_to_centroid_line_geom geometry';

    --create a line between the centroid and the nearest station
    EXECUTE 'UPDATE '||quote_ident(output_table_name)||' SET nearest_feature_to_centroid_line_geom =
ST_MakeLine('||quote_ident(feature_geometry_column_name)||', '||quote_ident(zone_centroid_geometry_column_name)||')';
```

```
--feature geometry constraints
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_dims_feature_geom CHECK
(st_ndims(feature_geom) = ' || feature_dims || ');
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_srid_feature_geom CHECK
(st_srid(feature_geom) = ' || srid || ');
--EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_geotype_feature_geom CHECK
(geometrytype(feature_geom) = ' || quote_literal(feature_geometry_type) || '::text OR feature_geom IS NULL)';

--zone geometry constraints
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_dims_zone_centroid_geom CHECK
(st_ndims(zone_centroid_geom) = 2)';
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_srid_zone_centroid_geom CHECK
(st_srid(zone_centroid_geom) = ' || srid || ');
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_geotype_zone_centroid_geom CHECK
(geometrytype(zone_centroid_geom) = "POINT"::text OR zone_centroid_geom IS NULL)';

--centroid->feature line geometry constraints
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT
enforce_dims_nearest_feature_to_centroid_line_geom CHECK (st_ndims(nearest_feature_to_centroid_line_geom) = 2)';
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT
enforce_srid_nearest_feature_to_centroid_line_geom CHECK (st_srid(nearest_feature_to_centroid_line_geom) = ' || srid || ');
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT
enforce_geotype_nearest_feature_to_centroid_line_geom CHECK (geometrytype(nearest_feature_to_centroid_line_geom) =
"LINESTRING"::text OR nearest_feature_to_centroid_line_geom IS NULL)';

--sequence
EXECUTE 'DROP SEQUENCE IF EXISTS ' || quote_ident(output_table_name) || '_id_seq';
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD COLUMN "id" INTEGER';
EXECUTE 'CREATE SEQUENCE ' || quote_ident(output_table_name) || '_id_seq';
EXECUTE 'UPDATE ' || quote_ident(output_table_name) || ' SET id = nextval(' || quote_ident(output_table_name) || '_id_seq') || ')';
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ALTER COLUMN "id" SET DEFAULT
nextval(' || quote_ident(output_table_name) || '_id_seq') || ')';
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ALTER COLUMN "id" SET NOT NULL';
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD UNIQUE ("id")';

--feature geometry index
EXECUTE 'DROP INDEX IF EXISTS ' || quote_ident(output_table_name) || '_feature';
EXECUTE 'CREATE INDEX ' || quote_ident(output_table_name) || '_feature' || ' ON ' || quote_ident(output_table_name) || ' USING
gist(feature_geom)';

--zone centroid geometry index
EXECUTE 'DROP INDEX IF EXISTS ' || quote_ident(output_table_name) || '_zone_centroid';
EXECUTE 'CREATE INDEX ' || quote_ident(output_table_name) || '_zone_centroid' || ' ON ' || quote_ident(output_table_name) || '
USING gist(zone_centroid_geom)';

--nearest feature to centroid line geometry index
EXECUTE 'DROP INDEX IF EXISTS ' || quote_ident(output_table_name) || '_near_feature_to_centroid_geom';
EXECUTE 'CREATE INDEX ' || quote_ident(output_table_name) || '_near_feature_to_centroid_geom' || ' ON
' || quote_ident(output_table_name) || ' USING gist(nearest_feature_to_centroid_line_geom)';

--zonecode index
EXECUTE 'DROP INDEX IF EXISTS ' || quote_ident(output_table_name) || '_zonecode';
EXECUTE 'CREATE UNIQUE INDEX ' || quote_ident(output_table_name) || '_zonecode' || ' ON
' || quote_ident(output_table_name) || ' ("ZoneCode")';

--name index
EXECUTE 'DROP INDEX IF EXISTS ' || quote_ident(output_table_name) || '_name';
EXECUTE 'CREATE UNIQUE INDEX ' || quote_ident(output_table_name) || '_name' || ' ON
' || quote_ident(output_table_name) || ' ("name")';

RETURN quote_ident(output_table_name);

END;
```

```
$BODY$  
LANGUAGE plpgsql VOLATILE  
COST 100;  
ALTER FUNCTION ws2_nearest_feature_to_centroid_in_zone(character varying, character varying, character varying, character varying,  
character varying, integer, character varying) OWNER TO postgres;
```

Figure 22 - plpgSQL function code to calculate the nearest feature to the centroid of each zone, that lies within that zone

plpgSQL function code for ws2_all_features_in_zone

```
CREATE OR REPLACE FUNCTION ws2_all_features_in_zone(character varying, character varying, character varying, character varying, character
varying, character varying, character varying, integer, character varying)
RETURNS character varying AS
$BODY$
DECLARE

    --table containing zones to bound features by
    zone_table_name ALIAS for $1;

    --geometry column of zone table containing actual zone boundary
    zone_boundary_geometry_column_name ALIAS for $2;

    --geometry column of zone table containing centroid of actual zone boundary
    zone_centroid_geometry_column_name ALIAS for $3;

    --column name uniquely identifying each zone
    zone_unique_column_name ALIAS for $4;

    --table of features
    feature_table_name ALIAS for $5;

    --geometry column name of feature table
    feature_geometry_column_name ALIAS for $6;

    --column name uniquely identifying each feature
    feature_unique_column_name ALIAS for $7;

    --srid for output table
    srid ALIAS for $8;

    --output table name to create
    output_table_name ALIAS for $9;

    --feature settings
    feature_geometry_type character varying;
    --feature_geometry_srid integer := 0;

    --zone settings
    --zone_boundary_geometry_srid integer := 0;
    --zone_centroid_geometry_srid integer := 0;
    zone_boundary_geometry_type character varying;
    zone_boundary_dims integer := 0;

BEGIN

    --DROP OUTPUT TABLE
    EXECUTE 'DROP TABLE IF EXISTS ' || quote_ident(output_table_name);

    EXECUTE 'CREATE TABLE ' || quote_ident(output_table_name) || ' AS SELECT
feature.' || quote_ident(feature_unique_column_name) || ' as "name", zone.' || quote_ident(zone_unique_column_name) || ' as "ZoneCode",
ST_Distance(feature.' || quote_ident(feature_geometry_column_name) || ', zone.' || quote_ident(zone_centroid_geometry_column_name) || ')
as distance, feature.' || quote_ident(feature_geometry_column_name) || ' as feature_geom,
zone.' || quote_ident(zone_boundary_geometry_column_name) || ' as zone_boundary_geom,
zone.' || quote_ident(zone_centroid_geometry_column_name) || ' as zone_centroid_geom FROM ' || quote_ident(feature_table_name) || ' as
feature, ' || quote_ident(zone_table_name) || ' as zone WHERE ST_Intersects(feature.' || quote_ident(feature_geometry_column_name) || ',
zone.' || quote_ident(zone_boundary_geometry_column_name) || ');

    --constraints on output table (feature geometry)
    EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_dims_feature_geom CHECK
(st_ndims(feature_geom) = 2);
    EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_srid_feature_geom CHECK
(st_srid(feature_geom) = ' || srid || ');
    --EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT enforce_geotype_feature_geom CHECK
```

```
(geometrytype(feature_geom) = '||quote_literal(feature_geometry_type)||::text OR feature_geom IS NULL)';

--constraints on output table (zone boundary geometry)
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT enforce_dims_zone_boundary_geom CHECK
(st_ndims(zone_boundary_geom) = 2)';
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT enforce_srid_zone_boundary_geom CHECK
(st_srid(zone_boundary_geom) = '||srid||')';
--EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT enforce_geotype_zone_boundary_geom CHECK
(geometrytype(zone_boundary_geom) = '||zone_boundary_geometry_type||::text OR zone_boundary_geom IS NULL)';

--constraints on output table (zone centroid geometry)
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT enforce_dims_zone_centroid_geom CHECK
(st_ndims(zone_centroid_geom) = 2)';
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT enforce_srid_zone_centroid_geom CHECK
(st_srid(zone_centroid_geom) = '||srid||')';
--EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT enforce_geotype_zone_centroid_geom CHECK
(geometrytype(zone_centroid_geom) = "POINT"::text OR zone_centroid_geom IS NULL);

--create spatial index
EXECUTE 'CREATE INDEX '||quote_ident(output_table_name)||'_feature'||' ON '||quote_ident(output_table_name)||' USING
gist(feature_geom)';
EXECUTE 'CREATE INDEX '||quote_ident(output_table_name)||'_centroid'||' ON '||quote_ident(output_table_name)||' USING
gist(zone_centroid_geom)';
EXECUTE 'CREATE INDEX '||quote_ident(output_table_name)||'_boundary'||' ON '||quote_ident(output_table_name)||' USING
gist(zone_boundary_geom)';

--add a sequence
EXECUTE 'DROP SEQUENCE IF EXISTS '||quote_ident(output_table_name)||'_id_seq';
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD COLUMN "id" INTEGER';
EXECUTE 'CREATE SEQUENCE '||quote_ident(output_table_name)||'_id_seq';
EXECUTE 'UPDATE '||quote_ident(output_table_name)||' SET id = nextval('||quote_ident(output_table_name)||'_id_seq')||''';
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ALTER COLUMN "id" SET DEFAULT
nextval('||quote_ident(output_table_name)||'_id_seq')||''';
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ALTER COLUMN "id" SET NOT NULL';
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD UNIQUE ("id");

--drop zonecode index
EXECUTE 'DROP INDEX IF EXISTS '||quote_ident(output_table_name)||'_zonecode';

--create zonecode index
EXECUTE 'CREATE INDEX '||quote_ident(output_table_name)||'_zonecode'||' ON
'||quote_ident(output_table_name)||' ("ZoneCode");

--drop name index
EXECUTE 'DROP INDEX IF EXISTS '||quote_ident(output_table_name)||'_name';

--create name index
EXECUTE 'CREATE INDEX '||quote_ident(output_table_name)||'_name'||' ON '||quote_ident(output_table_name)||' ("name");

RETURN quote_ident(output_table_name);

END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
ALTER FUNCTION ws2_all_features_in_zone(character varying, character varying, character varying, character varying, character varying,
character varying, character varying, integer, character varying) OWNER TO postgres;
```

Figure 23 - plpgsql function to create a table of all features within each zone

plpgsql function code for ws2_nearest_feature_to_centroid_in_out_zone_table

```
CREATE OR REPLACE FUNCTION ws2_nearest_feature_to_centroid_in_out_zone_table(character varying, character varying, character varying,
character varying, character varying, character varying, character varying, integer, character varying)
RETURNS character varying AS
$BODY$
DECLARE

    --table containing features of interest and matched zone boundaries and centroids
    features_per_zone_table_name ALIAS for $1;

    --geometry column of above table containing the feature geometry of interest
    feature_geometry_column_name ALIAS for $2;

    --geometry column of above table containing the zone centroid geometry
    zone_centroid_geometry_column_name ALIAS for $3;

    --column uniquely identifying each feature
    feature_unique_column_name ALIAS for $4;

    --column uniquely identifying each zone and zone centroid
    zone_unique_column_name ALIAS for $5;

    --all zone table name
    all_zone_table_name ALIAS for $6;

    --feature table
    feature_table_name ALIAS for $7;

    --srid of input feature and zone geometries, and output table (must match up)
    srid ALIAS for $8;

    --name of table to create to store results
    output_table_name ALIAS for $9;

    --store feature geometry settings
    feature_dims integer := 0;
    feature_geometry_type text := "";

    missing_zonecode_record RECORD;
    result_record RECORD;
    output_table_row_count integer := 0;
    feature_table_row_count integer := 0;
BEGIN

    --set feature geometry settings
    EXECUTE 'SELECT ST_NDims('||quote_ident(feature_geometry_column_name)||') FROM
'||quote_ident(features_per_zone_table_name) INTO feature_dims;

    --DROP TABLE
    EXECUTE 'DROP TABLE IF EXISTS '||quote_ident(output_table_name);

    --calculate a table containing all the zones with the nearest feature in each zone
    --CREATE TABLE
    EXECUTE 'CREATE TABLE '||quote_ident(output_table_name)||' AS SELECT * FROM (WITH A_QUERY AS (SELECT
'||quote_ident(zone_unique_column_name)||', min(distance) as min_distance, count(*) as feature_count FROM
'||quote_ident(features_per_zone_table_name)||' GROUP BY '||quote_ident(zone_unique_column_name)||' ORDER BY
'||quote_ident(zone_unique_column_name)||') SELECT
'||quote_ident(features_per_zone_table_name)||'.'||quote_ident(feature_unique_column_name)||' as "name",
'||quote_ident(features_per_zone_table_name)||'.'||quote_ident(zone_unique_column_name)||' as "ZoneCode",
'||quote_ident(features_per_zone_table_name)||'.distance,
'||quote_ident(features_per_zone_table_name)||'.'||quote_ident(feature_geometry_column_name)||' as feature_geom,
'||quote_ident(features_per_zone_table_name)||'.'||quote_ident(zone_centroid_geometry_column_name)||' as zone_centroid_geom,
A_QUERY.feature_count FROM '||quote_ident(features_per_zone_table_name)||', A_QUERY WHERE
```

```

'||quote_ident(features_per_zone_table_name)||'.distance = A_QUERY.min_distance ORDER BY
'||quote_ident(features_per_zone_table_name)||'.'||quote_ident(zone_unique_column_name)||') as QUERY';

EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD COLUMN nearest_feature_to_centroid_line_geom geometry';

--need to extract from table above features_per_zone_table_name all those zones not stored in output_table_name
FOR missing_zonecode_record IN EXECUTE 'SELECT DISTINCT(all_zone_table_name."ZoneCode") FROM
'||quote_ident(all_zone_table_name)||' as all_zone_table_name WHERE NOT EXISTS (SELECT * FROM '||quote_ident(output_table_name)||'
WHERE '||quote_ident(output_table_name)||'."ZoneCode" = all_zone_table_name.'||quote_ident(zone_unique_column_name)||')' LOOP

EXECUTE 'SELECT feature.'||quote_ident(feature_unique_column_name)||' as "name",
centroid.'||quote_ident(zone_unique_column_name)||' as "ZoneCode", st_distance(feature.geom, centroid.centroid_geom) as distance,
ST_AsText(feature.geom) as feature_geom, ST_AsText(centroid.centroid_geom) as centroid_geom FROM
'||quote_ident(feature_table_name)||' as feature, '||quote_ident(all_zone_table_name)||' as centroid WHERE
centroid.'||quote_ident(zone_unique_column_name)||' = '||quote_literal(missing_zonecode_record."ZoneCode")||' GROUP BY
feature.'||quote_ident(feature_unique_column_name)||', centroid.'||quote_ident(zone_unique_column_name)||', feature.geom,
centroid.centroid_geom ORDER BY feature.geom <-> centroid.centroid_geom LIMIT 1' INTO result_record;

IF result_record.centroid_geom IS NOT NULL AND result_record.feature_geom IS NOT NULL THEN

EXECUTE 'INSERT INTO '||quote_ident(output_table_name)||' ("name", "ZoneCode", distance, feature_geom,
zone_centroid_geom, feature_count, nearest_feature_to_centroid_line_geom) VALUES
('||quote_literal(result_record."name")||', '||quote_literal(result_record."ZoneCode")||', '||quote_literal(result_record.distance)||',
ST_GeomFromText('||quote_literal(result_record.centroid_geom)||', '||srid||'),
ST_GeomFromText('||quote_literal(result_record.feature_geom)||', '||srid||'), 1,
ST_MakeLine(ST_GeomFromText('||quote_literal(result_record.centroid_geom)||', '||srid||'),
ST_GeomFromText('||quote_literal(result_record.feature_geom)||', '||srid||'))));

END IF;

END LOOP;

--add a sequence
EXECUTE 'DROP SEQUENCE IF EXISTS '||quote_ident(output_table_name)||'_id_seq';
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD COLUMN "id" INTEGER';
EXECUTE 'CREATE SEQUENCE '||quote_ident(output_table_name)||'_id_seq';
EXECUTE 'UPDATE '||quote_ident(output_table_name)||' SET id = nextval('||quote_ident(output_table_name)||'_id_seq')||'';
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ALTER COLUMN "id" SET DEFAULT
nextval('||quote_ident(output_table_name)||'_id_seq')||'';
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ALTER COLUMN "id" SET NOT NULL';
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD UNIQUE ("id");

--ADD CONSTRAINTS ON GEOMETRY COLUMNS
--DIMS
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT enforce_dims_centroid_geom CHECK
(st_ndims(zone_centroid_geom) = 2)';
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT nearest_feature_to_centroid_line_geom CHECK
(st_ndims(nearest_feature_to_centroid_line_geom) = 2)';
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT enforce_dims_feature_geom CHECK
(st_ndims(feature_geom) = '||feature_dims||')';

--GEOTYPE
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT enforce_geotype_centroid_geom CHECK
(geometrytype(zone_centroid_geom) = "POINT"::text OR zone_centroid_geom IS NULL)';
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT
enforce_geotype_nearest_feature_to_centroid_line_geom CHECK (geometrytype(nearest_feature_to_centroid_line_geom) =
"LINESTRING"::text OR nearest_feature_to_centroid_line_geom IS NULL)';

--SRID
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT enforce_srid_centroid_geom CHECK
(st_srid(zone_centroid_geom) = '||srid||')';
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT
enforce_srid_nearest_feature_to_centroid_line_geom CHECK (st_srid(nearest_feature_to_centroid_line_geom) = '||srid||')';

```

```
EXECUTE 'ALTER TABLE '||quote_ident(output_table_name)||' ADD CONSTRAINT enforce_srid_feature_geom CHECK
(st_srid(feature_geom) = '||srid||')';

RETURN quote_ident(output_table_name);

END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
ALTER FUNCTION ws2_nearest_feature_to_centroid_in_out_zone_table(character varying, character varying, character varying, character
varying, character varying, character varying, character varying, integer, character varying) OWNER TO postgres;
```

Figure 24 - plpgsql function to create a table of nearest feature to a zone centroid, beginning by selecting the nearest feature within a zone, then if no features exist within that zone, finding the nearest.

SQL Code and Django links to create Population CDAM value tables:

Table prefix/table name	Process Description
population_population	Each of these tables are used to store the results of executing the demographics CDAM. The tables themselves are defined as Django models within Python. Once the tables are defined as Python-based Django models, they are created as “real” tables within the database by executing the Django management function “syncdb” Django models: https://www.djangoproject.com/ https://docs.djangoproject.com/en/dev/ref/models/fields/ https://docs.djangoproject.com/en/dev/topics/db/models/ Django manage.py syncdb: https://docs.djangoproject.com/en/dev/ref/django-admin/ However, the tables can be created via execution of the following SQL:
population_populationfile	
population_populationmetaquery	
<p>Population_population:</p> <pre>CREATE TABLE population_population (id serial NOT NULL, file_uuid_id integer NOT NULL, "year" integer NOT NULL, gender integer NOT NULL, category integer NOT NULL, "location" text NOT NULL, "value" double precision NOT NULL, added_on timestamp with time zone NOT NULL, CONSTRAINT population_population_pkey PRIMARY KEY (id), CONSTRAINT population_population_file_uuid_id_fkey FOREIGN KEY (file_uuid_id) REFERENCES population_populationfile (id) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION DEFERRABLE INITIALLY DEFERRED, CONSTRAINT population_population_year_check CHECK (year >= 0)) WITH (OIDS=FALSE); ALTER TABLE population_population OWNER TO postgres;</pre> <p>Population values added to this table are checked for:</p> <p>Year – is positive integer (2000-2100)</p> <p>Gender – is one of Male (0), Female (1), Persons (2)</p> <p>Category – is one of 0-4 (0), 5-9 (1), 10-14 (2), 15-19 (3), 20-24 (4), 25-29 (5), 30-34 (6), 35-39 (7), 40-44 (8), 45-49 (9), 50-54 (10), 55-59 (11), 60-64 (12), 65-69 (13), 70-74 (14), 75-79 (15), 80-84 (16), 85-89 (17), 90+ (18), births (19), deaths (20), migration (21), urban density (22)</p> <p>Location – is one of Scotland, Wales, Northern Ireland, North East, North West, South East, South West, East Midlands, West Midlands, East of England, Yorkshire and The Humber, Greater London Authority/GLA/London or a district code value e.g. 00AA</p>	

Figure 25 - value table definitions for population CDAM

SQL Code to create Economics CDAM value tables:

Table prefix/table name	Process Description
Economics_economicsemployment Economics_economicsgva Economics_economicsenergy Economics_economicsukco2emissions Economics_economicsukghgemissions Economics_economicsukhouseholdexpenditure Economics_economicsindustryexpenditure Economics_economicsindustryexports Economics_economicsindustryimports Economics_economicsindustryoutputprices Economics_economicsukinvestment	<p>Each of these tables are used to store the results of executing the demographics CDAM. The tables themselves are defined as Django models within Python. Once the tables are defined as Python-based Django models, they are created as “real” tables within the database by executing the Django management function “syncdb”</p> <p>Django models: https://www.djangoproject.com/ https://docs.djangoproject.com/en/dev/ref/models/fields/ https://docs.djangoproject.com/en/dev/topics/db/models/</p> <p>Django manage.py syncdb: https://docs.djangoproject.com/en/dev/ref/django-admin/</p> <p>However, the tables can be created via execution of the following SQL:</p>
<p>Employment: CREATE TABLE economics_economicsemployment (id serial NOT NULL, file_uuid character varying(40) NOT NULL, "year" integer NOT NULL, sector integer NOT NULL, region integer NOT NULL, "value" double precision NOT NULL, added_on timestamp with time zone NOT NULL, CONSTRAINT economics_economicsemployment_pkey PRIMARY KEY (id), CONSTRAINT economics_economicsemployment_year_check CHECK (year >= 0)) WITH (OIDS=FALSE); ALTER TABLE economics_economicsemployment OWNER TO postgres;</p> <p>Year – is positive integer Sector – is a value in sectors table Region – is a value from government_office_regions table</p>	
<p>Energy: CREATE TABLE economics_economicsenergy (id serial NOT NULL, file_uuid character varying(40) NOT NULL, "year" integer NOT NULL, fuel_user integer NOT NULL, "value" double precision NOT NULL, added_on timestamp with time zone NOT NULL, CONSTRAINT economics_economicsenergy_pkey PRIMARY KEY (id), CONSTRAINT economics_economicsenergy_year_check CHECK (year >= 0)) WITH (OIDS=FALSE); ALTER TABLE economics_economicsenergy OWNER TO postgres;</p> <p>Year – is positive integer</p>	

Fuel_user – is value in fuel_users table

GVA:

```
CREATE TABLE economics_economicsgva
(
  id serial NOT NULL,
  file_uuid character varying(40) NOT NULL,
  "year" integer NOT NULL,
  sector integer NOT NULL,
  region integer NOT NULL,
  "value" double precision NOT NULL,
  added_on timestamp with time zone NOT NULL,
  CONSTRAINT economics_economicsgva_pkey PRIMARY KEY (id),
  CONSTRAINT economics_economicsgva_year_check CHECK (year >= 0)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE economics_economicsgva OWNER TO postgres;
```

Year – is positive integer

Sector – is a value in sectors table

Region – is a region in government_office_regions table

UK C02 Emissions:

```
CREATE TABLE economics_economicsukco2emissions
(
  id serial NOT NULL,
  file_uuid character varying(40) NOT NULL,
  "year" integer NOT NULL,
  fuel_user integer NOT NULL,
  "value" double precision NOT NULL,
  added_on timestamp with time zone NOT NULL,
  CONSTRAINT economics_economicsukco2emissions_pkey PRIMARY KEY (id),
  CONSTRAINT economics_economicsukco2emissions_year_check CHECK (year >= 0)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE economics_economicsukco2emissions OWNER TO postgres;
```

Year – is positive integer

Fuel_user – is value in fuel_users table

UK GHG Emissions:

```
CREATE TABLE economics_economicsukghgemissions
(
  id serial NOT NULL,
  file_uuid character varying(40) NOT NULL,
  "year" integer NOT NULL,
  fuel_user integer NOT NULL,
  "value" double precision NOT NULL,
  added_on timestamp with time zone NOT NULL,
  CONSTRAINT economics_economicsukghgemissions_pkey PRIMARY KEY (id),
  CONSTRAINT economics_economicsukghgemissions_year_check CHECK (year >= 0)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE economics_economicsukghgemissions OWNER TO postgres;
```

UK Household Expenditure:

```
CREATE TABLE economics_economicsukhouseholdexpenditure
(
  id serial NOT NULL,
```

```
file_uuid character varying(40) NOT NULL,
"year" integer NOT NULL,
consumption_category integer NOT NULL,
region integer NOT NULL,
"value" double precision NOT NULL,
added_on timestamp with time zone NOT NULL,
CONSTRAINT economics_economicsukhouseholdexpenditure_pkey PRIMARY KEY (id),
CONSTRAINT economics_economicsukhouseholdexpenditure_year_check CHECK (year >= 0)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE economics_economicsukhouseholdexpenditure OWNER TO postgres;
```

Year – is positive integer

Consumption Category – is one of values stored in consumption_categories table

Region – is a value from government_office_regions table

UK Industry Expenditure:

```
CREATE TABLE economics_economicsukindustryexpenditure
(
  id serial NOT NULL,
  file_uuid character varying(40) NOT NULL,
  "year" integer NOT NULL,
  sector integer NOT NULL,
  "value" double precision NOT NULL,
  added_on timestamp with time zone NOT NULL,
  CONSTRAINT economics_economicsukindustryexpenditure_pkey PRIMARY KEY (id),
  CONSTRAINT economics_economicsukindustryexpenditure_year_check CHECK (year >= 0)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE economics_economicsukindustryexpenditure OWNER TO postgres;
```

Year – is positive integer

Sector – is one of values stored in sectors table

UK Industry Exports:

```
CREATE TABLE economics_economicsukindustryexports
(
  id serial NOT NULL,
  file_uuid character varying(40) NOT NULL,
  "year" integer NOT NULL,
  sector integer NOT NULL,
  "value" double precision NOT NULL,
  added_on timestamp with time zone NOT NULL,
  CONSTRAINT economics_economicsukindustryexports_pkey PRIMARY KEY (id),
  CONSTRAINT economics_economicsukindustryexports_year_check CHECK (year >= 0)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE economics_economicsukindustryexports OWNER TO postgres;
```

Year – is positive integer

Sector – is one of values stored in sectors table

UK Industry Imports:

```
CREATE TABLE economics_economicsukindustryimports
(
  id serial NOT NULL,
  file_uuid character varying(40) NOT NULL,
  "year" integer NOT NULL,
  sector integer NOT NULL,
  "value" double precision NOT NULL,
```

```
added_on timestamp with time zone NOT NULL,
CONSTRAINT economics_economicsukindustryimports_pkey PRIMARY KEY (id),
CONSTRAINT economics_economicsukindustryimports_year_check CHECK (year >= 0)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE economics_economicsukindustryimports OWNER TO postgres;
```

Year – is positive integer

Sector – is one of values stored in sectors table

UK Industry Output Prices:

```
CREATE TABLE economics_economicsukindustryoutputprices
(
  id serial NOT NULL,
  file_uuid character varying(40) NOT NULL,
  "year" integer NOT NULL,
  sector integer NOT NULL,
  "value" double precision NOT NULL,
  added_on timestamp with time zone NOT NULL,
  CONSTRAINT economics_economicsukindustryoutputprices_pkey PRIMARY KEY (id),
  CONSTRAINT economics_economicsukindustryoutputprices_year_check CHECK (year >= 0)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE economics_economicsukindustryoutputprices OWNER TO postgres;
```

Year – is positive integer

Sector – is one of values stored in sectors table

UK Investment:

```
CREATE TABLE economics_economicsukinvestment
(
  id serial NOT NULL,
  file_uuid character varying(40) NOT NULL,
  "year" integer NOT NULL,
  uk_investing_sector integer NOT NULL,
  region integer NOT NULL,
  "value" double precision NOT NULL,
  added_on timestamp with time zone NOT NULL,
  CONSTRAINT economics_economicsukinvestment_pkey PRIMARY KEY (id),
  CONSTRAINT economics_economicsukinvestment_year_check CHECK (year >= 0)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE economics_economicsukinvestment OWNER TO postgres;
```

Year – is positive integer

UK_investing_sector – value in uk_investing_sectors_table

Region – is a value from government_office_regions table

Figure 26 - value table definitions for economics CDAM

SQL Code for foreign key constraints on European Environment Agency Waste Water Data

```
--ORIGINAL
ALTER TABLE "UWWTW_T_ReportPeriod" ADD CONSTRAINT "UWWTW_T_ReportPeriod_prkey" PRIMARY KEY (rptmstatekey);
ALTER TABLE "UWWTW_T_Reporter" ADD CONSTRAINT "UWWTW_T_ReportPeriod_frkey_reporter" FOREIGN KEY (rptmstatekey)
REFERENCES "UWWTW_T_ReportPeriod" (rptmstatekey) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION ALTER TABLE
"UWWTW_T_ReportPeriod" ADD CONSTRAINT;
ALTER TABLE "UWWTW_T_Uwwtp_Agglo" ADD CONSTRAINT "UWWTW_T_Uwwtp_Agglo_prkey" PRIMARY KEY (aucid);
ALTER TABLE "UWWTW_T_Uwwtp_Agglo" ADD CONSTRAINT "UWWTW_T_Uwwtp_Agglo_frkey_reporter" FOREIGN KEY (rptmstatekey)
REFERENCES "UWWTW_T_Reporter" (rptmstatekey) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION;
ALTER TABLE "UWWTW_T_ReceivingAreas" ADD CONSTRAINT "UWWTW_T_ReceivingAreas_prkey" PRIMARY KEY (rcaid);
ALTER TABLE "UWWTW_T_MSLevel" ADD CONSTRAINT "UWWTW_T_MSLevel_prkey" PRIMARY KEY (rptmstatekey);
ALTER TABLE "UWWTW_T_MSLevel" ADD CONSTRAINT "UWWTW_T_MSLevel_frkey_reporter" FOREIGN KEY (rptmstatekey) REFERENCES
"UWWTW_T_Reporter" (rptmstatekey) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION;

--FEB 2011 Update

ALTER TABLE "UWWTW_Feb2011_T_ReportPeriod" ADD CONSTRAINT "UWWTW_Feb2011_ReportPeriod_prkey" PRIMARY KEY
(rptmstatekey);
ALTER TABLE "UWWTW_Feb2011_T_Reporter" ADD CONSTRAINT "UWWTW_Feb2011_Reporter_prkey" PRIMARY KEY (rptmstatekey);
ALTER TABLE "UWWTW_Feb2011_ReportPeriod" ADD CONSTRAINT "UWWTW_Feb2011_ReportPeriod_frkey_reporter" FOREIGN KEY
(rptmstatekey) REFERENCES "UWWTW_Feb2011_T_Reporter" (rptmstatekey) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION;
ALTER TABLE "UWWTW_Feb2011_T_Uwwtp_Agglo" ADD CONSTRAINT "UWWTW_Feb2011_Uwwtp_Agglo_prkey" PRIMARY KEY (aucid);
ALTER TABLE "UWWTW_Feb2011_T_Uwwtp_Agglo" ADD CONSTRAINT "UWWTW_Feb2011_Uwwtp_Agglo_frkey_reporter" FOREIGN KEY
(rptmstatekey) REFERENCES "UWWTW_Feb2011_T_Reporter" (rptmstatekey) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION;
ALTER TABLE "UWWTW_Feb2011_T_ReceivingAreas" ADD CONSTRAINT "UWWTW_Feb2011_ReceivingAreas_prkey" PRIMARY KEY (rcaid);
ALTER TABLE "UWWTW_Feb2011_T_ReceivingAreas" ADD CONSTRAINT "UWWTW_Feb2011_Receiving_Areas_frkey_reporter" FOREIGN KEY
(rptmstatekey) REFERENCES "UWWTW_Feb2011_T_Reporter" (rptmstatekey) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION;
ALTER TABLE "UWWTW_Feb2011_T_MSLevel" ADD CONSTRAINT "UWWTW_Feb2011_MSLevel_prkey" PRIMARY KEY (rptmstatekey);
ALTER TABLE "UWWTW_Feb2011_T_MSLevel" ADD CONSTRAINT "UWWTW_Feb2011_MSLevel_frkey_reporter" FOREIGN KEY (rptmstatekey)
REFERENCES "UWWTW_Feb2011_T_Reporter" (rptmstatekey) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION;

--AUG 2012 Update

ALTER TABLE "UWWTW_Aug2012_T_ReportPeriod" ADD CONSTRAINT "UWWTW_Aug2012_T_ReportPeriod_prkey" PRIMARY KEY
("rptMStateKey");
ALTER TABLE "UWWTW_Aug2012_T_Reporter" ADD CONSTRAINT "UWWTW_Aug2012_T_Reporter_prkey" PRIMARY KEY ("rptMStateKey");
ALTER TABLE "UWWTW_Aug2012_T_ReportPeriod" ADD CONSTRAINT "UWWTW_Aug2012_T_ReportPeriod_frkey_reporter" FOREIGN KEY
("rptMStateKey") REFERENCES "UWWTW_Aug2012_T_Reporter" ("rptMStateKey") MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO
ACTION;
ALTER TABLE "UWWTW_Aug2012_T_Uwwtp_Agglo" ADD CONSTRAINT "UWWTW_Aug2012_T_Uwwtp_Agglo_prkey" PRIMARY KEY ("aucID");
ALTER TABLE "UWWTW_Aug2012_T_Uwwtp_Agglo" ADD CONSTRAINT "UWWTW_Aug2012_T_Uwwtp_Agglo_frkey_reporter" FOREIGN KEY
("rptMStateKey") REFERENCES "UWWTW_Aug2012_T_Reporter" ("rptMStateKey") MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO
ACTION;
ALTER TABLE "UWWTW_Aug2012_T_ReceivingAreas" ADD CONSTRAINT "UWWTW_Aug2012_T_ReceivingAreas_prkey" PRIMARY KEY
("rcalD");
ALTER TABLE "UWWTW_Aug2012_T_ReceivingAreas" ADD CONSTRAINT "UWWTW_Aug2012_T_Receiving_Areas_frkey_reporter" FOREIGN
KEY ("rptMStateKey") REFERENCES "UWWTW_Aug2012_T_Reporter" ("rptMStateKey") MATCH SIMPLE ON UPDATE NO ACTION ON DELETE
NO ACTION;
ALTER TABLE "UWWTW_Aug2012_T_MSLevel" ADD CONSTRAINT "UWWTW_Aug2012_T_MSLevel_prkey" PRIMARY KEY ("rptMStateKey");
ALTER TABLE "UWWTW_Aug2012_T_MSLevel" ADD CONSTRAINT "UWWTW_Aug2012_T_MSLevel_frkey_reporter" FOREIGN KEY
("rptMStateKey") REFERENCES "UWWTW_Aug2012_T_Reporter" ("rptMStateKey") MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO
ACTION;

--Comments (these can be applied to each version i.e. Feb2011, Aug2012)

COMMENT ON TABLE "UWWTW_T_Reporter" IS 'Table provides information on reporting state - (http://www.eea.europa.eu/data-and-maps/data/waterbase-uwwdt-urban-waste-water-treatment-directive/waterbase-uwwdt);
COMMENT ON TABLE "UWWTW_T_ReportPeriod" IS 'Table specifies version of reported data and reference year for reporting -
(http://www.eea.europa.eu/data-and-maps/data/waterbase-uwwdt-urban-waste-water-treatment-directive/waterbase-uwwdt);
COMMENT ON TABLE "UWWTW_T_ReceivingAreas" IS 'Table summarises information on designated sensitive areas, date of designation, the
purpose of the designation, type of sensitive areas. Moreover, the table specifies which particular article of the UWWD is applied in regards
to designation of sensitive areas - (http://www.eea.europa.eu/data-and-maps/data/waterbase-uwwdt-urban-waste-water-treatment;
```

directive/waterbase-uwwtdd');

COMMENT ON TABLE "UWWTW_T_MSLevel" IS 'Table summarises aggregated (on MS level) information on the sludge handling, its discharge and/or disposal and re-use of treated water - (<http://www.eea.europa.eu/data-and-maps/data/waterbase-uwwtdd-urban-waste-water-treatment-directive/waterbase-uwwtdd>);

COMMENT ON TABLE "UWWTW_T_DischargePoints_UK_ONLY_OSGB" IS 'Table contains information on individual points of discharge from treatment plants or collecting systems, localisation of discharge, link to specific treatment plant, type of receiving area into which the effluent/wastewater is discharged, related waterbody (or river basin), information on the discharge on land - (<http://www.eea.europa.eu/data-and-maps/data/waterbase-uwwtdd-urban-waste-water-treatment-directive/waterbase-uwwtdd>);

COMMENT ON TABLE "UWWTW_T_Agglomerations_UK_ONLY_OSGB" IS 'Table contains information on agglomerations with generated load \geq 2000 P.E., including names, coordinates, generated load and information whether the load generated is collected through collecting system or addressed via Individual Appropriate Systems (IAS) or not collected not addressed via IAS - (<http://www.eea.europa.eu/data-and-maps/data/waterbase-uwwtdd-urban-waste-water-treatment-directive/waterbase-uwwtdd>);

COMMENT ON TABLE "UWWTW_T_UWWTWS_UK_ONLY_OSGB" IS 'Table includes data on individual waste water treatment plants and collecting systems without UWWTW, their localisation, capacity and actual load treated, type of treatment, aggregated data on the performance of plants - (<http://www.eea.europa.eu/data-and-maps/data/waterbase-uwwtdd-urban-waste-water-treatment-directive/waterbase-uwwtdd>);

plpgsql function code for extracting country-specific routes from openflights data

```
--$1 - table name of all routes
--$2 - table name of subsetted airports e.g. UK only airports
CREATE OR REPLACE FUNCTION subset_openflights_routes_by_country(vvarchar, varchar, varchar)
RETURNS SETOF RECORD AS
$BODY$
DECLARE

    --all routes table name
    all_route_table_name ALIAS for $1;

    --subset table of airports
    subset_airport_table_name ALIAS for $2;

    --output table name
    output_table_name ALIAS for $3;

    --record when using all_route_table
    all_route_table_record RECORD;

    --record when using subset airport table
    subset_airport_table_record RECORD;

    --to store geometry of source airport
    source_geom text;

    --to store geometry of destination airport
    destination_geom text;

    --to store straight line between source and destination airport
    route_geom text;

BEGIN

    --loop around the airport table, getting each airport ID
    --select all records from the all_route table with corresponding ID (source_airport_id) and insert into temporary table
    --once complete, loop around the airport table again and delete from the second table all records where the destination id is not UK
    airport ID
    --the result is a table with only those routes starting and ending at UK airports i.e. UK internal domestic flights (or based upon set of
    airports specified as input, as could be from a different country)
    --drop the temporary table
    EXECUTE 'DROP TABLE IF EXISTS ' || quote_ident(output_table_name);
    --create a new temporary table with same structure as all route table
    EXECUTE 'CREATE TABLE ' || quote_ident(output_table_name) || ' AS SELECT * FROM ' || quote_ident(all_route_table_name);
    --remove any records
    EXECUTE 'DELETE FROM ' || quote_ident(output_table_name);
    --add a geometry column (source airport)
    EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD COLUMN source_airport_geom geometry';
    --add a geometry column (destination airport)
    EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD COLUMN destination_airport_geom geometry';
    --add a geometry column (source->destination route geometry)
    EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD COLUMN source_destination_route_geom geometry';
    --add geometry checks source geom
    EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT "enforce_srid_source_geom" CHECK
(st_srid(source_airport_geom) = 4326)';
    EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT "enforce_geotype_source_geom" CHECK
(geometrytype(source_airport_geom) = "POINT"::text OR source_airport_geom IS NULL)';
    --add geometry checks destination geom
    EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT "enforce_srid_destination_geom" CHECK
(st_srid(destination_airport_geom) = 4326)';
    EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT "enforce_geotype_destination_geom" CHECK
(geometrytype(destination_airport_geom) = "POINT"::text OR destination_airport_geom IS NULL)';
    --add geometry checks source_destination_route geom
```

```

EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT "enforce_srid_source_destination_route_geom"
CHECK (st_srid(source_destination_route_geom) = 4326)';
EXECUTE 'ALTER TABLE ' || quote_ident(output_table_name) || ' ADD CONSTRAINT
"enforce_geotype_source_destination_route_geom" CHECK (geometrytype(source_destination_route_geom) = "MULTILINESTRING"::text OR
geometrytype(source_destination_route_geom) = "LINESTRING" OR source_destination_route_geom IS NULL)';

--loop all the airports
FOR subset_airport_table_record IN EXECUTE 'SELECT DISTINCT(airportid) FROM ' || quote_ident(subset_airport_table_name) || '
ORDER BY airportid ASC' LOOP

    --loop all the routes with UK as starting airport
    IF subset_airport_table_record.airportid IS NOT NULL THEN

        FOR all_route_table_record IN EXECUTE 'SELECT * FROM ' || quote_ident(all_route_table_name) || ' WHERE
source_airport_id = ' || subset_airport_table_record.airportid LOOP

            --get the source airport geom
            EXECUTE 'SELECT ST_AsText(geom) FROM ' || quote_ident(subset_airport_table_name) || ' WHERE
airportid = ' || quote_nullable(all_route_table_record.source_airport_id) INTO source_geom;

            RAISE NOTICE 'source_geom: %', source_geom;

            --get the destination airport geom
            EXECUTE 'SELECT ST_AsText(geom) FROM ' || quote_ident(subset_airport_table_name) || ' WHERE
airportid = ' || quote_nullable(all_route_table_record.destination_airport_id) INTO destination_geom;

            RAISE NOTICE 'destination_geom: %', destination_geom;

            IF source_geom IS NOT NULL AND destination_geom IS NULL THEN

                EXECUTE 'INSERT INTO ' || quote_ident(output_table_name) || ' (airline, airlineid,
source_airport, source_airport_id, destination_airport, destination_airport_id, codeshare, stops, equipment, source_airport_geom) VALUES
(' || quote_nullable(all_route_table_record.airline) || ',' || quote_nullable(all_route_table_record.airlineid) || ',' || quote_nullable(all_route_table
_record.source_airport) || ',' || quote_nullable(all_route_table_record.source_airport_id) || ',' || quote_nullable(all_route_table_record.destinati
on_airport) || ',' || quote_nullable(all_route_table_record.destination_airport_id) || ',' || quote_nullable(all_route_table_record.codeshare) || ',' ||
quote_nullable(all_route_table_record.stops) || ',' || quote_nullable(all_route_table_record.equipment) || ',
ST_GeomFromText(' || quote_literal(source_geom) || ', 4326))';

                END IF;

                IF destination_geom IS NOT NULL AND source_geom IS NULL THEN

                    EXECUTE 'INSERT INTO ' || quote_ident(output_table_name) || ' (airline, airlineid,
source_airport, source_airport_id, destination_airport, destination_airport_id, codeshare, stops, equipment, destination_airport_geom)
VALUES
(' || quote_nullable(all_route_table_record.airline) || ',' || quote_nullable(all_route_table_record.airlineid) || ',' || quote_nullable(all_route_table
_record.source_airport) || ',' || quote_nullable(all_route_table_record.source_airport_id) || ',' || quote_nullable(all_route_table_record.destinati
on_airport) || ',' || quote_nullable(all_route_table_record.destination_airport_id) || ',' || quote_nullable(all_route_table_record.codeshare) || ',' ||
quote_nullable(all_route_table_record.stops) || ',' || quote_nullable(all_route_table_record.equipment) || ',
ST_GeomFromText(' || quote_literal(destination_geom) || ', 4326))';

                    END IF;

                    IF source_geom IS NOT NULL AND destination_geom IS NOT NULL THEN

                        --create a line for the route between the two airports
                        EXECUTE 'SELECT
ST_AsText(ST_MakeLine(ST_GeomFromText(' || quote_nullable(source_geom) || ', 4326),
ST_GeomFromText(' || quote_nullable(destination_geom) || ', 4326))) INTO route_geom;

                        RAISE NOTICE 'route_geom: %', route_geom;

                        EXECUTE 'INSERT INTO ' || quote_ident(output_table_name) || ' (airline, airlineid,

```

```

source_airport, source_airport_id, destination_airport, destination_airport_id, codeshare, stops, equipment, source_airport_geom,
destination_airport_geom, source_destination_route_geom) VALUES
('||quote_nullable(all_route_table_record.airline)||','||quote_nullable(all_route_table_record.airlineid)||','||quote_nullable(all_route_table
_record.source_airport)||','||quote_nullable(all_route_table_record.source_airport_id)||','||quote_nullable(all_route_table_record.destinati
on_airport)||','||quote_nullable(all_route_table_record.destination_airport_id)||','||quote_nullable(all_route_table_record.codeshare)||','||
|quote_nullable(all_route_table_record.stops)||','||quote_nullable(all_route_table_record.equipment)||',
ST_GeomFromText('||quote_literal(source_geom)||', 4326), ST_GeomFromText('||quote_literal(destination_geom)||', 4326),
ST_GeomFromText('||quote_literal(route_geom)||', 4326));

                END IF;

            END LOOP;

        END IF;
    END LOOP;

    --remove records where destination is not in uk airport list i.e. flights starting but leaving the UK
    EXECUTE 'DELETE FROM '||quote_ident(output_table_name)||' WHERE destination_airport_id NOT IN (SELECT DISTINCT (airportid)
as destination_airport_id FROM '||quote_ident(subset_airport_table_name)||' ORDER BY airportid)';

    RETURN QUERY EXECUTE 'SELECT * FROM '||quote_ident(output_table_name);

END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
ALTER FUNCTION subset_openflights_routes_by_country(varchar, varchar, varchar) OWNER TO postgres;

```

Figure 27 - plpgsql function code to extract country-specific flight routes from openflights data

plpgsql function code for calculate_GOR_centroid_to_boundary_distance for solid waste CDAM

```
CREATE OR REPLACE FUNCTION calculate_GOR_centroid_to_boundary_distance(text, text, text) RETURNS void AS $BODY$
DECLARE

    centroid_table_name ALIAS for $1;
    vertex_table_name ALIAS for $2;
    output_table_name ALIAS for $3;

    centroid_record RECORD;

BEGIN

    --delete the output table if it already exists (param 3)
    EXECUTE 'DROP TABLE IF EXISTS ' || quote_ident(output_table_name);

    --create the output table
    EXECUTE 'CREATE TABLE ' || quote_ident(output_table_name) || ' AS SELECT ' || quote_ident(centroid_table_name) || '.gid' || ',
' || quote_ident(centroid_table_name) || '.orig_fid' || ', ' || quote_ident(centroid_table_name) || '.in_fid' || ',
' || quote_ident(centroid_table_name) || '.near_fid' || ', ' || quote_ident(centroid_table_name) || '.near_dist',
AVG(ST_Distance(' || quote_ident(centroid_table_name) || '.geom, ' || quote_ident(vertex_table_name) || '.geom)) AS "AVG_Vertex_Distance"
FROM ' || quote_ident(centroid_table_name) || ', ' || quote_ident(vertex_table_name) || ' GROUP BY
' || quote_ident(centroid_table_name) || '.gid, ' || quote_ident(centroid_table_name) || '.orig_fid,
' || quote_ident(centroid_table_name) || '.in_fid, ' || quote_ident(centroid_table_name) || '.near_fid,
' || quote_ident(centroid_table_name) || '.near_dist';

END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
ALTER FUNCTION calculate_GOR_centroid_to_boundary_distance(text, text, text) OWNER TO postgres;
```

Figure 28 - plpgsql function code to calculate average straight line distance between output area centroid and centroid of government office region with which the output area resides.