Creating PostgreSQL databases

It is easy to create databases in postgreSQL using either the GUI, or by directly running SQL queries.

Creating a database

1. In pgAdmin 4 Connect to the database server.
2. In the server tree select the postgres database.
3. Open up the query tool. Tools > Query Tool.
4. Create a new database using the following query. Once you've written the query click the Lightning icon to run.

- CREATE DATABASE documentation

```
CREATE DATABASE training;
```

On refreshing the server tree by right clicking and selecting 'Refresh' you will see the new database. From now on we'll do everything on this database.

An alternative to this would be to use the GUI. Right click on databases and select Create.

Creating a table

- CREATE TABLE documentation

Databases are made up of tables. To create these we can either run SQL statements or use the GUI.

```
CREATE TABLE postcodes (
    postcode varchar(8),
    positional_quality_indicator integer,
    po_box_indicator char(1),
    total_number_of_delivery_points integer,
    delivery_points_cplc integer,
    domestic_delivery_points integer,
    non_domestic_delivery_points integer,
    po_box_delivery_points integer,
    matched_address_premises integer,
    unmatched_delivery_points integer,
    country_code varchar(9),
    nhs_regional_ha_code varchar(9),
    nhs_ha_code varchar(9),
    admin_county_code varchar(9),
    admin_district_code varchar(9),
    admin_ward_code varchar(9),
    postcode_type char(1),
    wkt text,
    CONSTRAINT pk_postcodes_postcode PRIMARY KEY(postcode)
);
```

Data types

One of the most important considerations in creating databases is the data types used in tables. Here are a selection of commonly used PostgreSQL data types. When creating database tables PostgreSQL will handle a number of common ways of specifying data types. For example, character(10) is the same as char(10).

Type	Description
character varying(10)/varchar(10)	variable-length characters with a specified limit
character(10)/char(10)	fixed-length set of characters
text	variable unlimited length
integer	non decimal number -2147483648 to +2147483647
numeric	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point

serial	auto-incrementing integer
date	a date value
timestamp	a date and time value
integer array/integer[]	2 dimensional and 3 dimensional arrays of a specified type

Loading data

Data can be manually loaded into PostgreSQL using SQL commands.

- INSERT INTO documentation

```
INSERT INTO postcodes
VALUES('EX1 1EE',10,'N',32,32,18,14,0,15,0,'E92000001',
'E19000002','E18000010','E10000008','E07000041','E05003502',
'S','POINT (292079 92307)');
```

Postgres also allows for loading in data using the COPY command. Typically this will be loading into individual tables from CSV data.

A tip for combining CSV files in Windows is to use the command prompt copy command. For example, in a directory full of postcode CSV files run:

- Microsoft copy syntax

```
copy *.csv postcodes.csv
```

Now, taking an Ordnance Survey sample from code point we'll use the PostgreSQL COPY command to import all postcodes from a CSV file.

Before we do this we need to clear down the postcode added previously. For this we can just wipe all data from the table.

- TRUNCATE documentation

```
TRUNCATE postcodes;
```

Then we can load in the postcodes from Code Point.

- COPY documentation

```
COPY postcodes FROM 'C:\Development\DaveBathnes\PostgreSQL-Training\data\codepoint.csv' HEADER CSV;
```