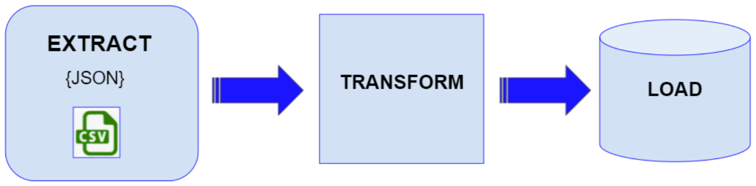
# ELT Project Report



**Completed by:** Nathan Potter and Leanne Porter

# Mobile Blackspots in Australia

Project team:

- Leanne Porter
- Nathan Potter

## Introduction

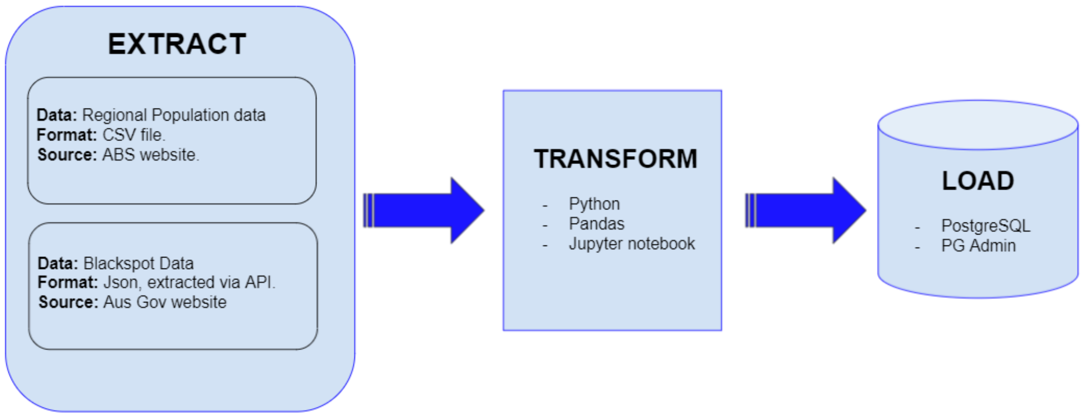
This project created a database for interrogating information about Australia's mobile blackspots and the populations of those areas.

The project was designed around an imaginary scenario where a new Federal Telecommunications Minister wants to know where all the current mobile backspots are around Australia as well as what are the key characteristics of the populations in those areas.

To deliver the database, the project team undertook three steps.

Firstly, data was extracted from different sources. Secondly, the data was transformed to ensure it was consistent and usable, and the data was segmented by creating tables in accordance with a specially designed database. Finally, the database was designed and the tables were loaded into it.

The diagram here illustrates the steps taken.



The methods used by the project team to Extract, Transform and Load (ETL) data are described in the sections below.

## 1. Extract

The data for this project was extracted from two sources: one on mobile black spots in Australia, and the other on regional population growth across all local government areas (LGAs) in the country.

Data of interest on mobile blackspots included information on:

- Their locations (site name, latitude and longitude)
- The jurisdictions in which they are found (LGA, electorate, state/ territory)

Population data of interest included:

- Population numbers for 2018 and 2019
- Population growth (in numbers and percentages)
- The type of population growth (natural, internal migration, overseas migration)
- The LGAs where the populations are located
- The size of those LGAs (km2)
- The population density of the LGAs

Mobile Blackspot API data extraction

- Before beginning the extraction note that the below dependencies are required:
    - import requests
    - import json
    - import pandas as pd
    - from sqlalchemy import create_engine
    - from config import *

- Mobile blackspot data came from the Australian Government Data website.
- The dataset is available as an API with a base url as per below:

  "https://data.gov.au/data/api/3/action/datastore_search?resource_id=c6b211ad-3aa2-4f53-8427-01b52a6433a7"

- To be able to pull all the required data from the API key, an offset and limit criteria is required to be added to the url. Offset starts at 0 and the limit is the total number of rows in the dataset.
- After inspection of the json response, it was determined that the total number of rows in the dataset was 13480. This in turn led to the below url being used as the query.

  "https://data.gov.au/data/api/3/action/datastore_search?resource_id=c6b211ad-3aa2-4f53-8427-01b52a6433a7&offset=0&limit=13480"

- Using Python, empty lists were created for all required columns. These columns include: location, identifier, state, lga, electorate, latitude and longitude.

- A for loop was then used to pull the data for each column from the json data and the response is appended to the corresponding list.
- Once the for loop is complete, the length of each list should match the total number of rows in the dataset (13480).

Regional population growth data extraction

- The population data came from the Regional population dataset on the ABS website. https://www.abs.gov.au/statistics/people/population/regional-population/2018-19#data-download
- The dataset is found under “Data downloads - data cubes” and is titled Population estimates by Local Government Area, 2018 to 2019. The file downloads as a xls file and will need to be transformed into a csv file as detailed in the transform section.

## 2. Transform

Blackspot data

- Created a dataframe in pandas with extracted blackspot data from API.
- Ensured blackspot dataframe columns matched required names (column names changed: lga to local_government_area, latitude to lat, longitude to lon).
- Identified non-matching LGAs between blackspot data and regional population growth data using a V-Look-Up in Excel. This resulted in 10 unmatched LGA’s (Unincorporated ACT, Unincorp. Other Territories, Western Plains Regional, Gundagai, Mallala, Orroroo/Carrieton, Break O'Day, Glamorgan/Spring Bay, Waratah/Wynyard, Kalgoorlie/Boulder).
- Removed local government areas from blacksport dataframe where there were no matches with the regional population growth data.
- Created two new dataframes which are then used to import data into database tables inPostgreSQL.
    - A blackspot_transformed dataframe was created with the columns location, lat, lon and local_government_area. The index name was set to “bs_id”.
    - A jurisdiction_transformed dataframe was created with the columns local_government_area, electorate and state. The index name was set to “index”. Duplicate values were dropped in the local_government_area column.
- The data frames were loaded into the below tables (see Load section for further details):
    - blackspot_transformed was loaded into the Blackout_spots table.
    - jurisdiction_transformed was loaded into the Jurisdiction table.

Regional population growth data

- In Excel, tables for each state and territory (from different spreadsheet tabs) were extracted and adjoined in one table that contained all the regional population growth data.
- Column names were aligned (final column names: LGA code, Local Government Area, 2018, 2019, Growth no. Growth %, Natural increase no., Net internal migration no., Net overseas migration no., Area (km2), Population density 2019 (persons/km2)
- The table was saved as a csv file, which was further transformed using pandas.
- In pandas, from the Local Government Area column, the brackets and the space which followed LGA names were removed using "str.replace(r"\(.*\)","")" and str.strip()
- In pandas, column names were renamed to match database table names. (renamed columns: lga_code, local_government_area, pop_2018, pop_2019, growth_no, births_no, int_migration_no, os_migration_no, lga_area, pop_density).
- The following dataframes were created in pandas from the extracted regional population growth data:
    - An lga_transformed dataframe which contained the column local_government_area. The duplicates in the column were dropped.
    - A location_transformed dataframe which contained the columns lga_code and local_government_area. The index was set to lga_code.
    - A growth_transformed dataframe which contained the columns lga_code, growth_no and growth_per. The index was named growth_index.
    - A num_transformed dataframe which contained the columns lga_code, pop_2018 and pop_2019. The index was named pop_index.
    - A density_transformed dataframe which contained the columns lga_code, lga_area and pop_density. The index was named density_index.
    - A growth_type_transformed dataframe which contained the columns lga_code, births_no, int_migration_no and os_migration_no. The index was named growth_type_index.

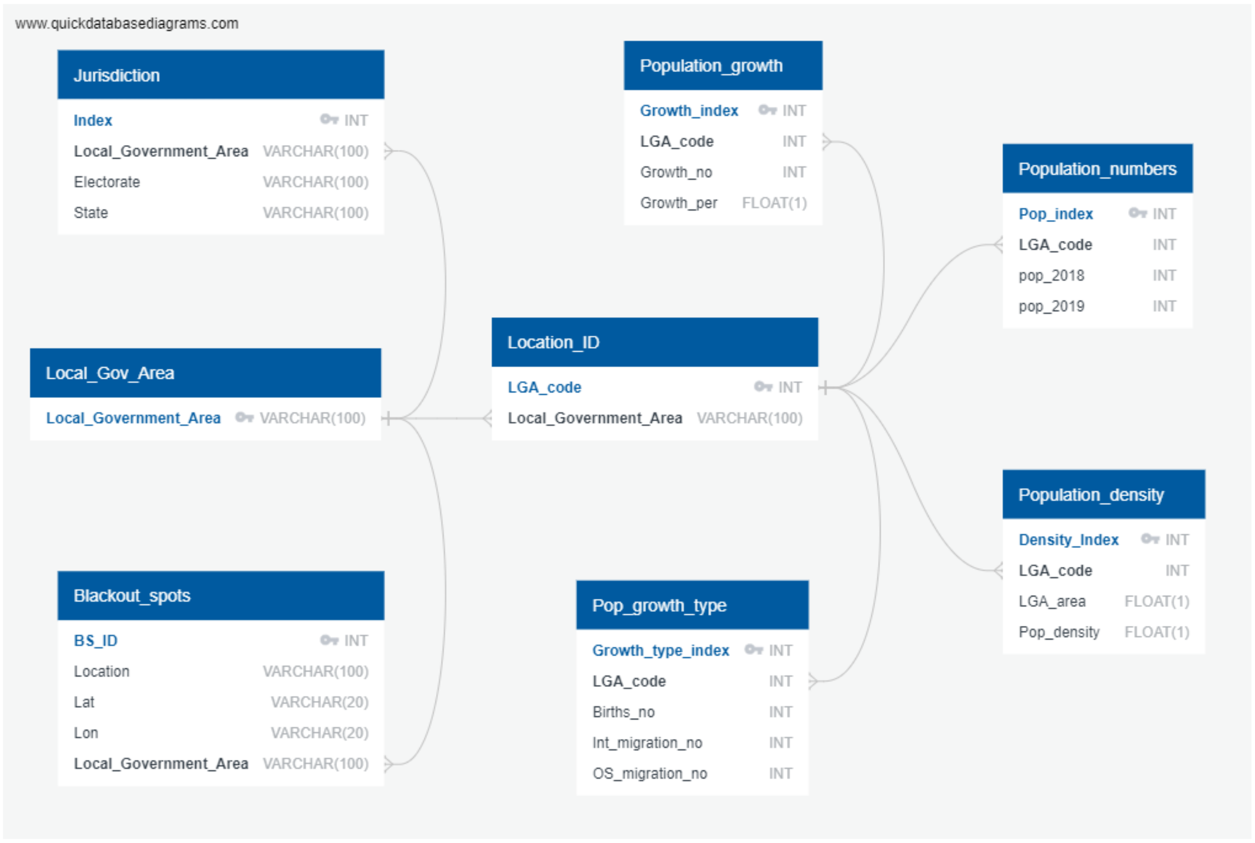
- The data frames were loaded into the tables as per below (see Load section for further details):
    - lga_transformed was loaded into the Local_Gov_Area table.
    - location_transformed was loaded into the Location_ID table.
    - growth_transformed was loaded into the Population_growth table.
    - num_transformed was loaded into the Population_numbers table.
    - density_transformed was loaded into the Population_density table.
    - growth_type_transformed was loaded into the Pop_growth_type table.

## 3. Load

To load the data, a database was designed that sets out the structure of the tables needed. The database was set up in PGAdmin, and the table structure for the database was created using PostgreSQL. The extracted data was then loaded into the database's tables using Pandas.

Database design

- A data schema was designed using QUICK DBD. See below.



- To avoid a unique constraint error when uploading into PostgreSQL, a 'local_government_area' table and a 'Location_ID' table were created. This allowed every table to be assigned a primary key that could be matched with a primary key of another table by assigning a foreign key.
- In the above diagram, primary keys appear with a key next to the name. Foreign keys reference a primary key in another table. Columns with a foreign key link appear in black.

PGAdmin set up of the database and tables structure

- In PGAdmin a database ("LGA_db") was created.
- The database schema was exported from QUICK DBD as a PostgreSQL file.
- The SQL code was cleaned in VS Code. The quotation marks in the exported file were removed using a find and replace method.
- The code was then used as described in the next section to create tables in PGAdmin, including the table relationships for our database.

Creating tables and table-relations

- When creating the tables, 'drop table' code was added for each table first. Then the code to create each table in the database was added.
- The code to create the tables specified column names and primary key. It also included the column type and set all columns to "NOT NULL". Refer to DBD diagram for column types and restrictions.
- Data for each table was loaded into PGAdmin from Pandas (as described below).
- After tables were loaded into PGAdmin from Pandas, the PostgreSQL code to assign foreign keys was added.
- This allows for a workable database where different tables can be joined to interrogate specific aspects of the date.

Loaded data into database tables with Pandas

- A config file is required that contains a username and password for the postgreSQL connection.
- The connection string was defined in the below format: `"{}:{}@localhost:5432/LGA_db".format(username, password)`
- The engine was created which uses the connection string to link to postgresql.
- Each dataframe created was loaded into PostgreSQL by specifying the name and then using the .to_sql method. This method requires the PostgreSQL table name to be specified, the engine to be configured, if the data exists it needs to be appended and depending on the requirement the index is set to either true or false). E.g. `location_transformed.to_sql(name='location_id', con=engine, if_exists='append', index=True)`
- Note: The only dataframe that does not need to have the index imported is the lga_transformed database. This is only one column and it is unique and therefore an index is not required.
- Once each dataframe was loaded into PostgreSQL, PGAdmin was used to confirm tables were loaded and created correctly.

This completed the mobile blackspots database.

## Conclusion

For this project, data was extracted, transformed and loaded using the methods described above to create a database.

The database allows users to interrogate different information about mobile blackspots in Australia and the characteristics of the populations in those areas. For this, each table in the database can be joined as needed by using the primary and foreign keys assigned.