CityPulse Smart City Data ETL - MVP

Author: Sam Patterson

Date: 19.12.2012

Executive Summary

A minimum viable product (MVP) has been developed to run an extract, transform, load (ETL) pipeline on a variety of Smart City datasets.

The publically accessible data has been successfully ingested into a database in a format appropriate for a data analyst to conduct visualisation and analysis of the data.

Since only one iteration of development has been condcuted, several potential improvements to the data model and pipeline are suggested. Nonentless, the fully functioning end-to-end tool demonstrates a range of quality data engineering practices which means it would be straightforward to improve with furhter iteration.

Introduction

A mini-project has been undertaken over a weekend to build an extract, transform, load (ETL) pipeline for Smart City data collected from the CityPulse project.

The purpose of the document is to describe the project's activities and outcomes. After outlining the goals and constraints of the project, the following sections describe the steps taken to complete the project. Finally, some discussion about limitations and future work is given.

Goals and constraints

The primary goal of this project is to get the raw CityPulse datasets into a format that can be queried for visualisation and analysis.

To this end, the desired outcome of the project is a functioning end-to-end MVP of the pipeline which takes the raw data, applies basic transformations, and populates an empty database.

The main constraint of this project is time. It is being performed over a weekend as a recruitment challenge. Therefore, only one iteration of design and development is being conducted. This means that while the primary goal is being achieved, there will be plenty of potential follow up work to further iterate, refine and improve the solution.

In terms of technological constraints, since this project is starting from scracth and is being developed in isolation from any other systems, only open-source libraries and local computing resources are going to be used. However, technologies will be chosen that will enable the tool to be used with other databases and deployment environments.

Explore and assess the data

The first step of the project is to explore the CityPulse datasets listed on their website to understand the variety and volume of data to be processed and ingested.

Datasets

The following datasets are listed on the CityPulse website.

Data type	City	Raw format	# Datasets listed
Road Traffic Data	Aarhus	Compressed CSV files	4
Pollution Data	Aarhus, Brasov	Compressed CSV files	2
Weather Data	Aarhus, Brasov	Compressed JSON files	4
Parking Data	Aarhus	CSV file	2
Social (Webcasted) Event Data	Surrey	CSV file	1
Cultural Event Data	Aarhus	CSV file	1
Library Event Data	Aarhus	CSV file	1

In addition to these, the following meta data is also provided.

Metadata	Relates to
Road Traffic Sensors	Road Traffic and Pollution Datasets
Parking Lots	Parking Datasets

To begin with, the 'raw' format of the data was manually downloaded. However, several of the files were found to be duplicated and miscategorised in CityPulse's indexing of the files.

Dataset index issues

Upon initial review of the linked datasets, the following issues were identified:

- The name of the linked file for the 'Aarhus Road Traffic Dataset-1' (`citypulse_traffic_raw_data_surrey_feb_jun_201`) indicates that it is from Surrey.
- Linked file for the 'Aarhus Road Traffic Dataset-4' (`cultural_events_aarhus.csv`) points to either 'Aarhus Cultural Event Dataset-1' or 'Aarhus Road Traffic Dataset-3'.
- Linked file for the 'Aarhus Parking Dataset-2' (`aarhus_parking.csv`) points to the 'Aarhus Parking Event Dataset-1'.
- Linked file for the 'Brasov Pollution Dataset-1', 'Brasov Weather Dataset-1/2' files (`citypulse_pollution_annotated_da`) and `raw_weather_data_aarhus.tar.gz` and `raw_weather_data_aug_sep_2014.zip`) point to the same files from Aarhus.

An inspection into the backend file structure of the website was conducted, however the correct files did not appear to be there either. In these cases, the duplicated files are being ignored from here on and in the developed tool.

Extracting the data

To inspect the data more closely and begin building the initial 'extract' step of the ETL pipeline. Functions have been set up to automatically download and extract, where necessary, the datasets based on a JSON configuration file with the following format.

```
[
    {
        "name": "Aarhus Road Traffic Dataset-1",
        "data_type": "Road Traffic Data",
        "url": "http://iot.ee.surrey.ac.uk:8080/datasets/traffic/traffic_feb_june/citypulse_traffic_raw
        "location": "Aarhus"
    },
    {
        "name": "Surrey Social Event Dataset-1",
        "data_type": "Social Event Data",
        "url": "http://iot.ee.surrey.ac.uk:8080/datasets/surreyevents/surrey_events.csv",
        "location": "Surrey"
    },
    ...
]
```

The following steps were taken to download each dataset, clean it, and read it into the Python development environment.

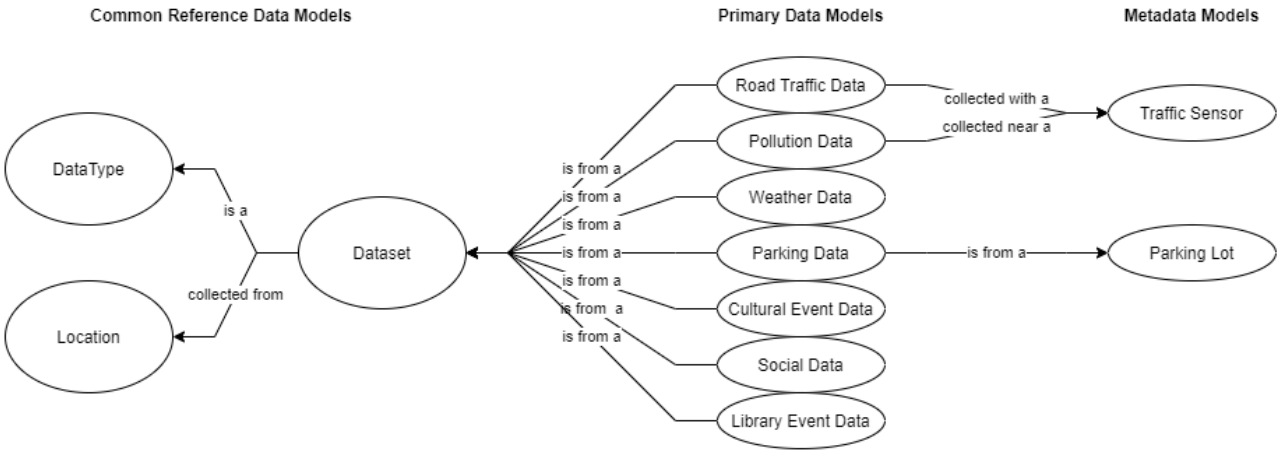
- Downloading raw files using the requests library and saving to a local data cache
- Unpacking from either `.tar.gz` or `.zip` archives where necessary
- Ignored non-data files (e.g. `__MACOSX` hidden directories)
- Applied column names to CSVs where they were missing (based on other similar files or the developer's best guess)
- Column names were converted into the same (`snake_case`) styling for consistency across the datasets
- Read into pandas DataFrame objects
- Setting correct data types (e.g. `datetime` for timestamps)
- Removing duplicated rows (e.g. columns with identical values from the road traffic datasets)
- Removing duplicated columns (e.g. urls in columns 2 and 4 of 'Surrey Social Event' dataset are identical)

Define the data model

The initial design of the CityPulse ETL package comprises of a conceptual data model (which aligns with the tables of the relational database) along with the functional requirements of the command line tool that runs the ETL pipeline.

First pass conceptual data model

A first iteartion of the conceptual data model was developed based on the key types of data.



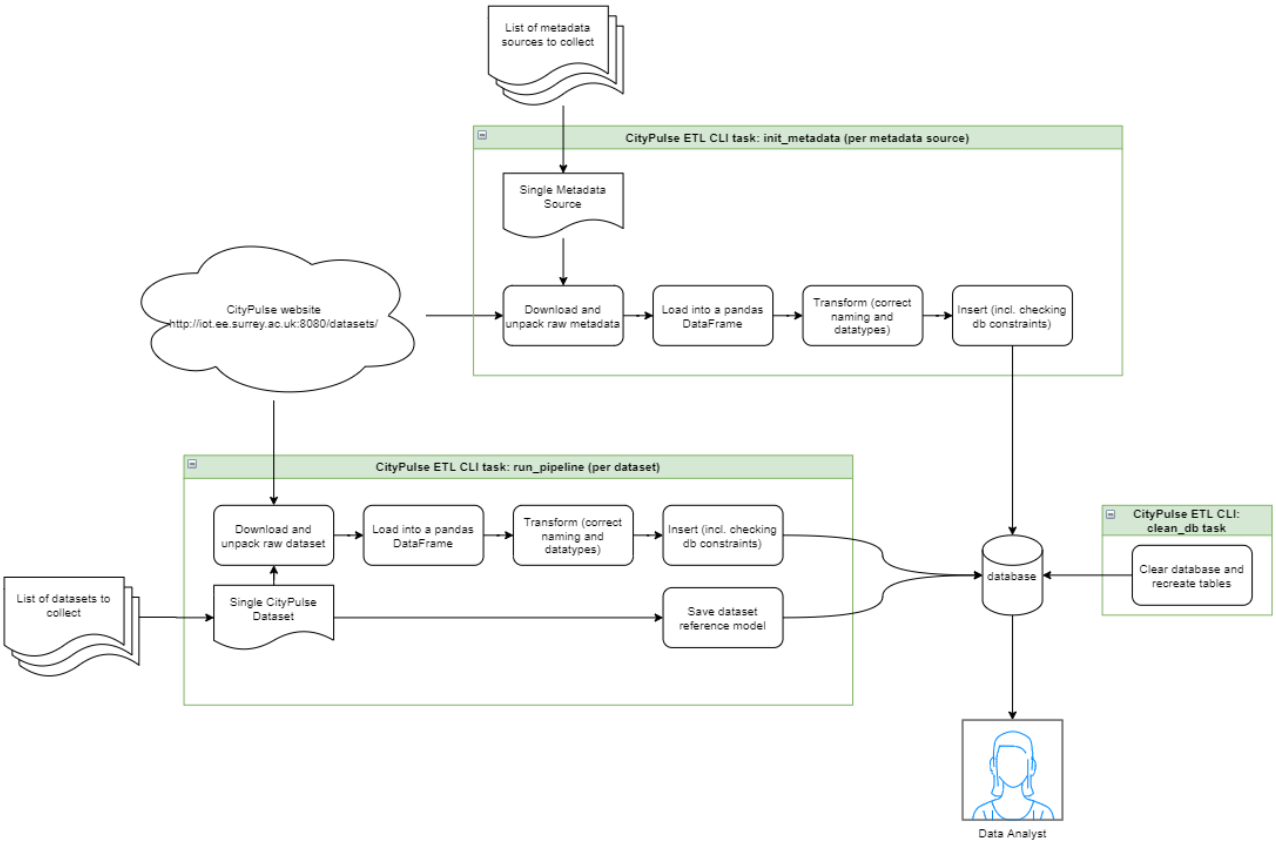
The primary data models are the seven different types of datasets hosted CityPulse. Alongside these are the two metatata models provided that relate to the Road Traffic, Pollution, and Parking Lot data.

There are also three reference data models which align with how CityPulse indexes the data.

While there is potential for further normalisation of the data model, this model achieves the project's primary goal within the time constraints. Furthermore, it would be recommened to further refine and clarify the user cases for the data before investing more effort to design a more sophisticated model, following an agile approach.

Functional design

A Command Line Interface (CLI) will be responsible for setting up the database and running the pipeline.



The CLI can run three tasks:

- `run-pipeline` - This runs the ETL pipeline on a collection of datasets hosted by CityPulse. It takes a json file with a list of dictionaries that defines the datasets to be processed by the pipeline.
- `init-metadata` - This initialises the metadata tables. It should be run each time the database is recreated and potentially more often if new metadata becomes available.
- `clean-db` - This clears all the data from the database and creates the empty tables. For development it is useful when making large changes to the pipeline. For deployment to production, it only needs to be run once.

Run ETL to model the data

The above design has been developed as an MVP and is available in CityPulse ETL GitLab repository.

The collection of technologies and libraries used have been selected based on familiarity to the developer, their maturity, and their popularity across the industry.

- Python 3 - often considered the default language to use in the data engineering / data science fields
- SQLAlchemy - provides an abstraction away from the database which will allow the tool to work on different databases if required
- pandas - gives powerful functionality for transforming large quantities of data
- requests - a widely-used and robust library for HTTP resources
- SQLite - a simple and self-contained SQL database

The tool has been built as a installable Python package that provides the CLI tool to the console when installed for ease-of-deployment. As well as being useful for running the pipeline, installing the package also gives access to the SQLAlchemy ORM classes which could be used by a consumer to interact with the data instead of SQL.

Discussion

This section discusses the developed tool from the perspective of using it or adapting it to meet some additional requirements.

Streaming data in real time

- Ping the server for new datasets

- Run the command as a scheduled task
- Or alter to accomodate an IoT pub/sub type of communication

100x more IoT

- Use a different database
- Run from cloud or on edge
- Consider a more robust ETL framework / product

Converting geographic coordinates into human-readable addresses ...

Harmonising the cultural and library event datasets ...

Other potential uses of this data ...

Recommended improvements / future work

In addition to the potential improvements discussed above, the following work could also be conducted to improve the tool.

- Find out where the wrongly linked datasets are
- Use blob storage for raw data? (e.g. Azure Blog / AWS S3)
- Use Object Oriented patterns better in `models` (i.e. inheritence / shared methods)
- Document system more formually, using something like arc42
- Validate column name guesses (e.g. Cultural Event Data)

Summary

The developed MVP has successfully met the primary goal of the project within the constraints.

A CLI tool allows the user to initialise an SQL database and run the ETL pipeline on the data hosted on the CityPulse website to populate the target database.

Discussion into future potential uses higlights the flexibility of the tool as well as potentials for future improvements.