MileStone 2 - Great Britain Road Safety Data Visualization

Yingchuan Sun, Jiahong Xu, Carter Cheng

November 11 2025

1 Introduction

We aim to turn the UK government's Road Safety Data (also known as STATS19) into an intuitive, open website for public analysis and exploration.

The Road Safety Data (STATS19) dataset, published by the UK Department for Transport (DfT), contains police-reported road traffic collisions across Great Britain. It consists of three linked tables—collisions, vehicles, and casualties. In this project, we focus on the five most recent years (2020–2024), which include approximately 0.5 million collision records. Each record provides time and date, location (latitude/longitude and grid references), accident severity (Fatal/Serious/Slight), numbers of vehicles and casualties, road type and speed limit, weather and light conditions, road surface, local authority and police force, and contributory factors.

Our project will develop a open-source and interactive web platform built on DuckDB with a streamlit UI. The platform will provide interactive maps, trend charts, and severity-aware comparisons that make the data easy to explore and understand. The website will help planners, researchers, and the public use this dataset as a reliable reference for improving road safety outcomes.

2 Feature list

- **Data Cleaning (15%)**

  Deliver a reproducible pipeline that ingests the STATS19 tables and outputs a unified analytic schema. Tasks include dropping records with missing or invalid latitude or longitude, decoding categorical codes to readable labels using a codebook, standardizing date, hour, day of week, and month, and harmonizing accident severity across years with a toggle for original versus adjusted series.

  - **Sample queries:** coordinate validation; code to label mapping; derive `month = date_trunc('month', accident_date)`.
  - **Success:** one command builds the dataset and all cleaned tables.

- **Overview Dashboard (20%)**

  Show the annual and monthly safety picture: collisions, casualties, and severity composition, plus year over year and month over month changes. Include a switch to compare original and adjusted severity series.

  - **Sample queries:** monthly counts by severity; cumulative year totals.
  - **Success:** choosing a year updates charts and values match backend aggregates. Load everything within 1.5 seconds.

- **Interactive Heatmap (25%)**

  Visualize collision locations as a heatmap with filters for severity, weekday, hour, road type, speed limit, weather, light, and surface conditions.

  - **Sample queries:** select records for a chosen year with non null latitude and longitude and active filters.
  - **Success:** smooth map interaction that load within 2 seconds, and all plotted points have valid coordinates.

- **Mode and Demographics Explorer (20%)**

  Present casualties by road user type such as pedestrian, pedal cycle, motorcycle, and car, with severity breakdowns and demographic splits by age group and sex.

  - **Sample queries:** counts from the casualty table grouped by `casualty_class`, `casualty_severity`, age bins, and sex.
  - **Success:** clicking a user type filters linked charts and unknown age is shown explicitly.

- **Road and Environment $\times$ Severity (20%)**

  Analyze how conditions relate to outcomes: speed limit, road type, junction control, road surface, light, and weather. Provide single dimension and two way views.

  - **Sample queries:** counts by `speed_limit` and severity; two way pivot such as `road_type` by `light_conditions`.

- **Documentation and Deployment** Provide clear setup, data update, and deployment instructions. Include a one command local run and a hosted option.

  - **Success:** a new machine reproduces the pipeline and launches the site easily.

3 Database Design

Technology Stack

- **Frontend:** Streamlit for UI; Plotly for charts; PyDeck for maps.
- **Backend/Data:** Python (pandas, DuckDB).
- **Deploy:** `requirements.txt` + run script.

Initial Database Design (Schema)

- `collision` (PK: `accident_index`): date, time, latitude, longitude, severity, vehicle/casualty counts, road_type, speed_limit, weather, light, surface, authority.
- `vehicle` (PK: `accident_index, vehicle_reference`; FK→`collision`): vehicle_type, manoeuvre, driver_age, driver_sex, towing, etc.
- `casualty` (PK: `accident_index, casualty_reference`; FK→`collision`): casualty_class, casualty_severity, age, sex, position, etc.
- `contributory_factor` (PK: `accident_index, factor_code, rank`; FK→`collision`): one row per factor mention.
- `code_map`: lookup of STATS19 codes to human-readable labels.

Expected Size

- **Columns per table:** `collision` 44; `vehicle` 32; `casualty` 23.
- **Five-year total:** ∼0.5M collisions, ∼0.92M vehicles, ∼0.64M casualties.

Assumptions

1. We analyze GB police-reported injury collisions only (STATS19). Property-damage-only events are out of scope.
2. All categorical fields are decoded with code lists and normalized via a `code_map` for cross-year consistency.
3. Because the definition of severity changed over time, the UI offers an Original vs Adjusted severity toggle in trends.
4. Map views include only valid latitude/longitude.

Task List

Setup

- Create the repository and set up the Python environment.
- Provide a script to run setup and start the app.

Data and ETL

- Ingest CSVs
- Build code_map
- Decode coded columns
- Persist to DuckDB

Database

- Create DuckDB tables: `collision`, `vehicle`, `casualty`, `contributory_factor`, `code_map`.
- Build pre-aggregated tables: `kpi_monthly`, `by_hour`, `by_dow`.

Checks

- Add tests for code mapping, time parsing, foreign-key joins, and coordinate validity.
- Verify key queries finish in an acceptable time.

Backend Queries

- Implement Python functions for overview KPIs, heatmap data, mode/demographics, and road/environment $\times$ severity.

Frontend (Streamlit)

- Build sidebar filters: year, severity, time, conditions.
- Implement pages: Overview, Heatmap, Mode & Demographics, Road & Environment.

Docs and Deploy

- Write a concise README with run steps and data caveats.
- Deploy to cloud and confirm it loads.