Project Report: Analysing Australian Road Fatal Crash with a Data Warehouse

1. Introduction

1.1 Project Aim and Significance

This project focuses on designing, building, and using a data warehouse to analyze historical data on fatal road crashes in Australia. Road safety has always been an ongoing challenge to all parts of society. Australia's rate of 4.54 was ranked 18th out of the 31 nations. The nations with the 3 lowest rates were Norway (2.14), Sweden (2.17) and Iceland (2.39)[1], indicating room for improvement. Furthermore, recent national statistics highlight a concerning trend, with a 12-month total of 1205 road deaths nationally leading up to July 2024, representing a 8.4% increase from the prior year [2]. There is a critical need for using data-driven approaches to understand complex contributing factors to fatal crashes.

Therefore, we uses DW methodologies to structure relevant datasets as analytical foundation, then utilize data visualization and association rule mining techniques to uncover significant patterns and correlations within the data. The ultimate goal is to derive actionable insights or formulate evidence-based recommendations that could, hopefully, contribute towards national efforts to reduce road fatalities and improve road safety.

1.2 Data Sources Used

The primary data sources for this project are from the Australian Road Deaths Database (ARDD) [3]:

- `bitre_fatal_crashes_dec2024.xlsx`
- `bitre_fatalities_dec2024.xlsx`
- `Population estimates by LGA, Significant Urban Area, Remoteness Area, Commonwealth Electoral Division and State Electoral Division, 2001 to 2023.xlsx`

Additionally, geospatial boundary data was utilized:

- `GeoJSON` files for Local Government Areas (LGA), and States (STE).

1.3 Methodology

The project employed several key methodologies:

- **Dimensional Modeling:** Kimball's four-step process [4] guided the design of the data warehouse schema.
- **ETL (Extract, Transform, Load):** Processes were developed using Python with Pandas, Numpy library and Excel to clean, transform, and load data from source files into the PostgreSQL database.
- **Data Warehousing:** A relational database warehouse was implemented using PostgreSQL.
- **Data Visualization: Tableau** was used to create interactive dashboards visualizing key insights derived from business queries against the warehouse.
- **Association Rule Mining (ARM):** Python libraries (`mlxtend`) were used to perform ARM to discover interesting relationships within the fatality data.

1.4 Report Structure

This report details the project in the following sections: Section 2 outlines the data warehouse design process and the final schema. Section 3 presents the business questions the warehouse is designed to answer. Section 4 describes the ETL process in detail. Section 5 discusses the implementation and presents key visualizations. Section 6 details the association rule mining process, results, and recommendations. Section 7 concludes the report with a summary, limitations, and future work. Section 8 lists references, and Section 9 contains appendices.

2. Data Warehouse Design and Schema (Kimball's Steps)

2.1 Business Process Understanding -Process

The primary business process being modeled is the **recording and analysis of fatal road crashes** occurring within Australia. This involves capturing details about the crash event itself (when, where, how) and the resulting fatalities (who). A secondary, related process essential for contextual analysis (especially for calculating rates) is **tracking population distribution** across relevant geographical areas (State, Remoteness Area).

2.2 Declare the Grain

The grain, or the level of detail represented by a single row, was defined for each fact table:

- **`fact_crash`:** The grain is one row per unique fatal crash event identified by `Crash_ID`.
- **`fact_fatalities`:** The grain is one row per unique fatality identified by `fatality_id`, linked to a specific crash via `Crash_ID`.
- **`fact_population_wide`:** The grain is one row per unique combination of State and Remoteness Area, containing population estimates as separate columns for the years

2001 to 2023. *More details related to why&how we designed this fact table are illustrated in Section4.4 transformation details - Mapping*

2.3 Conceptual Schema and StarNet Diagram

A **Fact Constellation Schema** was chosen for the overall data warehouse structure, comprising multiple fact tables (`fact_crash` , `fact_fatalities` , `fact_population_wide`) that share some common dimension tables (e.g., `Dim_State` , `Dim_Remoteness`). The core analysis of crash events, centered around `fact_crash` , adheres to **Star Schema** principles. This hybrid approach was selected because:

- It naturally accommodates the different grains of data (crashes, individual fatalities, population counts).

- The Star Schema component for `fact_crash` offers simplicity and generally better query performance for common analyses compared to more normalized structures [4].

- Shared dimensions allow for integrated analysis across the different fact tables (e.g., linking crash locations to population data).

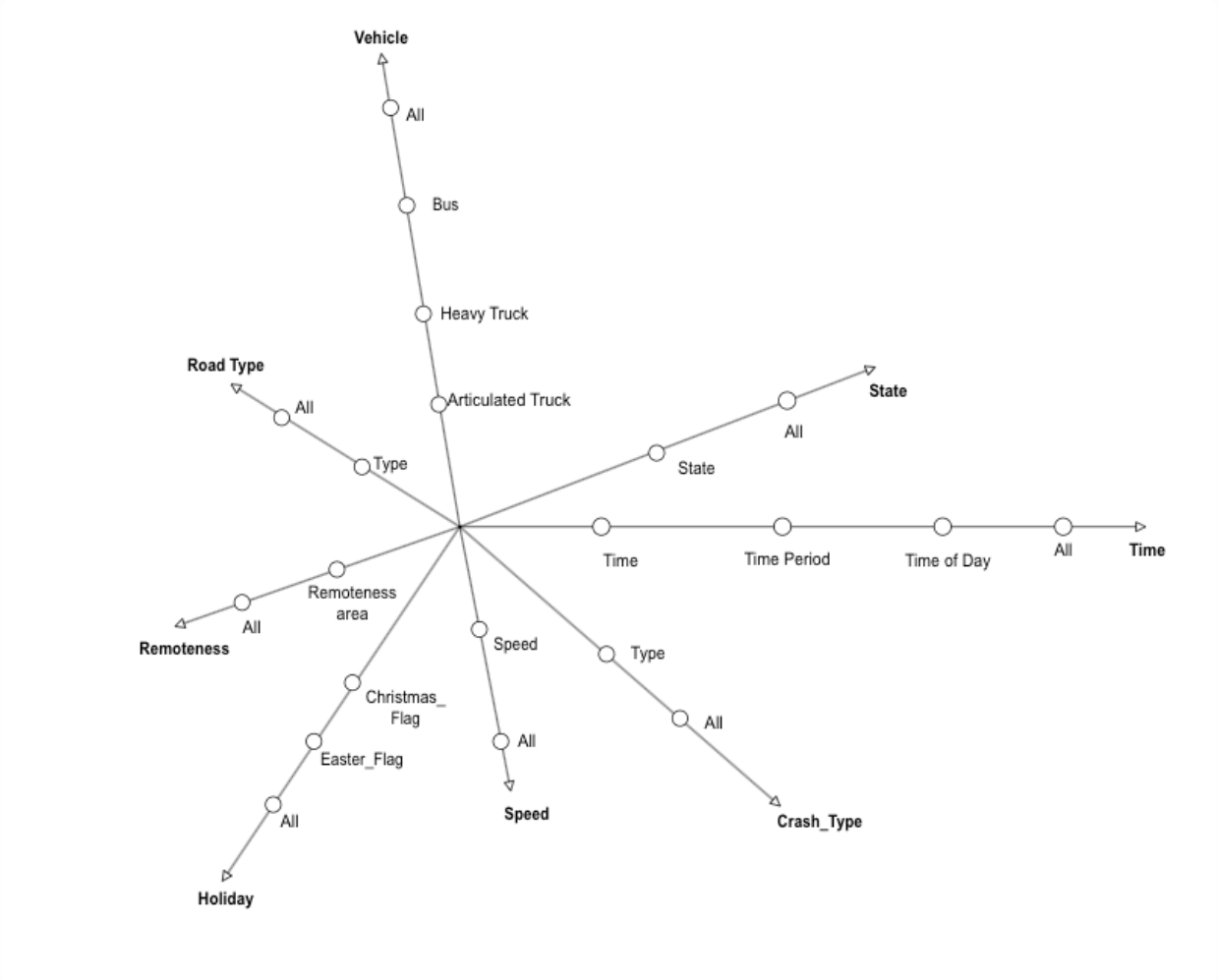


Figure 1: StarNet Diagram illustrating the Data Warehouse Schema.

Explanation: Figure 1 shows the logical structure of the warehouse. The fact tables (`fact_crash`, `fact_fatalities`, `fact_population_wide`) are central, containing measures and foreign keys. They connect to the surrounding dimension tables (`Dim_Time`, `dim_speed_limit`, etc.) via these keys. The diagram illustrates the attributes within each dimension and indicates concept hierarchies (e.g., within `Dim_Time`), visually representing the schema's capability to support analytical queries through slicing, dicing, and drill-down/roll-up operations.

2.4 Dimension Table Design (Kimball Step 3: Dimensions)

2.4.1 `Dim_Time`

```
CREATE TABLE Dim_Time (
    Time_ID SERIAL PRIMARY KEY,
    Time INTEGER, -- Represents the hour of the day (0-23)
    Time_of_Day VARCHAR(20),  -- e.g., Day, Night
    Time_Period VARCHAR(20)  -- e.g., Morning, Afternoon, Late Night
);
```

- **Purpose:** Describes the time-of-day aspects of a crash.
- **Attributes:** `Time_ID` (PK), `Time` (Hour), `Time_of_Day`, `Time_Period`.
- **Hierarchy:** `Time` (Hour) -> `Time_Period` -> `Time_of_Day`.
- **Rationale:** Simplified to focus on intra-day patterns. Date components (`Year`, `Month`, `Dayweek`) are kept in `fact_crash` for dimension simplicity.

2.4.2 `dim_speed_limit`

```
CREATE TABLE dim_speed_limit (
    Speed_ID INT PRIMARY KEY,
    Speed_Value INT NOT NULL,     -- keep -9 for unknown (data type consistency) , 3
    Description VARCHAR(20) NOT NULL -- e.g., '50 km/h', 'Unknown', '<40 km/h'
);
```

- **Purpose:** Describes the posted speed limit at the crash location.
- **Attributes:** `Speed_ID` (PK), `Speed_Value`, `Description`.
- **Hierarchy:** None formally defined; conceptual grouping possible.
- **Rationale:** Captures speed limit, handling specific codes ('Unknown', '<40').

2.4.3 `Dim_State`

```
CREATE TABLE Dim_State (
    State_ID SERIAL PRIMARY KEY,
```

```
    State VARCHAR(3) NOT NULL  -- e.g., WA, NSW, VIC,...
);
```

- **Purpose:** Identifies the Australian state or territory.
- **Attributes:** `State_ID` (PK), `State` (Abbreviation).
- **Hierarchy:** State/Territory is the primary level.

2.4.4 `Dim_Remoteness`

```
CREATE TABLE Dim_Remoteness (
    Remoteness_ID SERIAL PRIMARY KEY,
    Remoteness_Areas TEXT NOT NULL  -- e.g., Major Cities, ..., Unknown
);
```

- **Purpose:** Classifies location based on remoteness.
- **Attributes:** `Remoteness_ID` (PK), `Remoteness_Areas`.
- **Hierarchy:** Follows ABS remoteness levels (treated as categories).

2.4.5 `Dim_Road_Type`

```
CREATE TABLE Dim_Road_Type (
    Road_ID SERIAL PRIMARY KEY,
    Road_Type VARCHAR(50)   -- e.g., Highway, Arterial, ..., Undetermined
);
```

- **Purpose:** Classifies the type of road.
- **Attributes:** `Road_ID` (PK), `Road_Type`.
- **Hierarchy:** None formally defined; conceptual grouping possible.

2.4.6 `Dim_Crash_Type`

```
CREATE TABLE Dim_Crash_Type (
    Crash_Type_ID SERIAL PRIMARY KEY,
    Crash_Type VARCHAR(10)  -- 'single'/'multiple' vehicle crash
);
```

- **Purpose:** High-level classification based on vehicle involvement.
- **Attributes:** `Crash_Type_ID` (PK), `Crash_Type`.
- **Hierarchy:** Basic binary: Single / Multiple.
- **Rationale:** Simplified from detailed ARDD crash types for high-level analysis.

2.4.7 `dim_vehicle`

```
CREATE TABLE dim_vehicle (
  vehicle_id     SERIAL PRIMARY KEY,
```

```
  bus_involved   SMALLINT,   -- 1:yes, 0:no, -9:unknown
  heavy_truck    SMALLINT,
  articulated_truck SMALLINT,
  Description TEXT
);
```

- **Purpose:** Describes involvement of specific large vehicle types.
- **Attributes:** `vehicle_id` (PK), flags for Bus, Heavy Truck, Articulated Truck involvement, `Description`.
- **Hierarchy:** None formally defined. Captures involvement profile.

2.4.8 `Dim_Holiday`

```
CREATE TABLE Dim_Holiday (
    Holiday_ID SERIAL PRIMARY KEY,
    Christmas_Flag VARCHAR(3),  -- yes/no
    Easter_Flag VARCHAR(3)      -- yes/no
);
```

- **Purpose:** Indicates if crash occurred during Christmas or Easter periods.
- **Attributes:** `Holiday_ID` (PK), `Christmas_Flag`, `Easter_Flag`.
- **Hierarchy:** None.
- **Rationale:** Focuses analysis on these specific high-travel periods.

note: full sql script at `dimentions.sql`

2.5 Fact Table Design (Kimball Step 4: Facts)

Fact tables design justified in section 2.6

2.5.1 `fact_crash`

```
CREATE TABLE fact_crash (
    Crash_ID INT NOT NULL PRIMARY KEY, -- unique PK
    Time_ID INT NOT NULL,
    Speed_ID INT NOT NULL,
    State_ID INT NOT NULL,
    Remoteness_ID INT NOT NULL,
    Road_type_ID INT NOT NULL,
    Crash_type_ID INT NOT NULL,
    Vehicle_ID INT NOT NULL,
    Holiday_ID INT NOT NULL,
    Month INT NOT NULL, -- Kept from source
    Year INT NOT NULL, -- Kept from source
    Dayweek VARCHAR NOT NULL, -- Kept from source
    LGA_name VARCHAR(255) NOT NULL, -- Kept from source unknown for blank(missing)
    Number_Fatalities INT NOT NULL, -- Kept from source
    -- Foreign Keys
    FOREIGN KEY (Time_ID) REFERENCES dim_time(Time_ID),
    FOREIGN KEY (Speed_ID) REFERENCES dim_speed_limit(Speed_ID),
```

```
    FOREIGN KEY (State_ID) REFERENCES dim_state(State_ID),
    FOREIGN KEY (Remoteness_ID) REFERENCES dim_remoteness(Remoteness_ID),
    FOREIGN KEY (Road_type_ID) REFERENCES dim_road_type(Road_ID),
    FOREIGN KEY (Crash_type_ID) REFERENCES dim_crash_type(Crash_type_ID),
    FOREIGN KEY (Vehicle_ID) REFERENCES dim_vehicle(Vehicle_ID),
    FOREIGN KEY (Holiday_ID) REFERENCES dim_holiday(Holiday_ID)
);
```

- **Grain:** One row per fatal crash event.
- **Foreign Keys:** Link to all 8 dimension tables.
- **Other Attributes:** `Month`, `Year`, `Dayweek`, `LGA_name`, `Number_Fatalities` (kept from source)

2.5.2 `fact_fatalities`

```
CREATE TABLE fact_fatalities (
    fatality_id INT PRIMARY KEY,          -- Unique ID for each fatality record
    Crash_ID INT REFERENCES fact_crash(Crash_ID), -- Link to crash event
    Road_User VARCHAR(50),                -- -- Kept from source
    Gender VARCHAR(10),                   -- Kept from source ('Unknown' for -9)
    Age INT,                              -- Kept from source (-9 for 'Unknown')
    Age_Group VARCHAR(15)                 -- Kept from source
);
```

- **Grain:** One row per fatality.
- **Measures:** Implicit `Fatality_Count` (additive), `Age` (non-additive descriptive measure).
- **Foreign Keys:** `Crash_ID` links to `fact_crash`.
- **Other Dimensions:** `Road_User`, `Gender`, `Age_Group`.

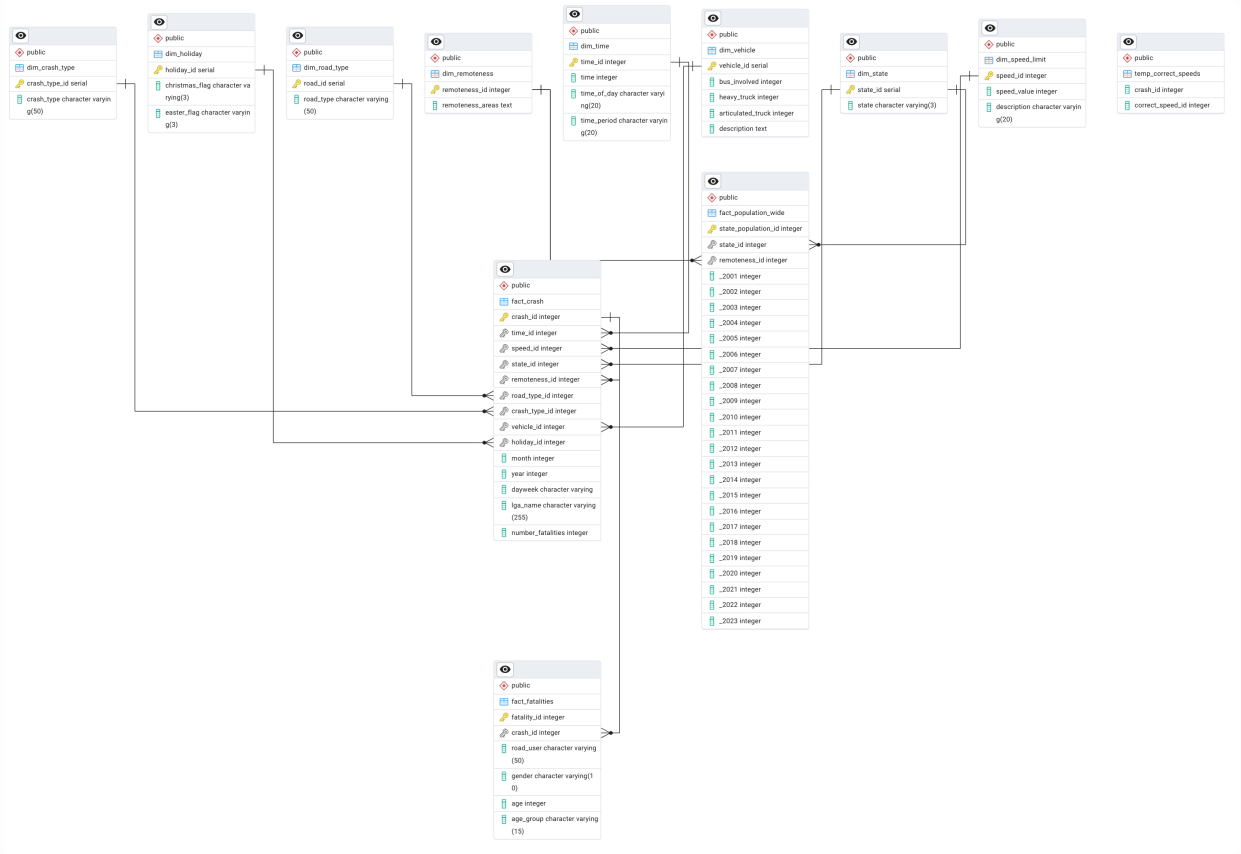


Figure 2: ERD from schema

note: full sql script at `create_fact_tables.sql`

2.5.3 `fact_population_wide`

```
CREATE TABLE fact_population_wide (
    State_Population_ID INT PRIMARY KEY,
     -- Foreign Keys linking to dimension tables
    State_ID INT NOT NULL REFERENCES dim_state(State_ID),
    Remoteness_ID INT NOT NULL REFERENCES dim_remoteness(Remoteness_ID),
    -- yearly Population columns note:column names must start with letter or _
    _2001 INT NULL, _2002 INT NULL, ..., _2023 INT NULL,

    -- CONSTRAINT to ensure the combination of State and Remoteness is unique
    -- qill enforce each row represents a unique geographic combination
    CONSTRAINT UQ_population_state_remoteness UNIQUE (State_ID, Remoteness_ID),

    -- FOREIGN KEY constraints linking to dimension tables
    CONSTRAINT FK_popwide_dim_state FOREIGN KEY (State_ID)
        REFERENCES dim_state (State_ID),

    CONSTRAINT FK_popwide_dim_remoteness FOREIGN KEY (Remoteness_ID)
        REFERENCES dim_remoteness (Remoteness_ID)
);
```

- **Grain:** One row per State-Remoteness area combination.
- **Measures:** `_2001` ... `_2023` (Population counts - semi-additive across geography, non-additive across time).

- **Foreign Keys:** `State_ID`, `Remoteness_ID` link to respective dimensions.
  note: full sql script at `creat_population.sql` with 5 queries to validate the table(also check constraint) and see some insightful results(see appendix)
  More details related to why&how we designed this fact table are illustrated in Section4.4 transformation details - Mapping

2.6 Design Rationale

Explaination

When putting this data warehouse together, we thought a lot about how to best arrange things, keeping Kimball's ideas(principles) in mind but also thinking about what we actually wanted to do with the data for this project, the specific analytical goals and data characteristics of this project. We tried different solutions and with the idea that dw design is often open-ended, so we made some specific choices aiming for a practical setup that made sense for analyzing these crash stats.

- Focus on `fact_crash` and Star Schema Core: Since the main point was understanding the crashes themselves – when, where, how fast, etc. – we built the core around the fact_crash table. This fits nicely with the star schema idea for looking at all the circumstances surrounding a crash.

- Simpler approach for `fact_fatalities` (Gender, Age_Group, Road_User): We considered about whether attributes like Age Group or Road User type needed their own separate dimension tables. In the end, we decided to keep them directly in the fact table. Our thinking was: for the kinds of charts and filters we planned for these details (like seeing age breakdowns), querying them straight from the fatalities table felt manageable enough. Creating tiny dimension tables (like one for Gender with just 'Male', 'Female', 'Unknown') seemed like it might add extra joins without much benefit for this project's goals. This keeps things a bit simpler when focusing just on who was involved.

- Keeping time details in `fact_crash` (Year, Month, Dayweek) instead of building a super detailed Dim_Date table with every single day.

- We were mainly interested in looking at yearly trends over the long run, and maybe some basic monthly or weekly patterns. Making separate small dimensions just for months 1-12 or days 1-7 didn't feel necessary since those numbers are easy to understand on their own. Building a full daily date dimension gets really big and complex quickly, and we figured we weren't likely to ask super specific questions like "fatalities on the 3rd Tuesday of March in 1995". Keeping Year, Month, and Dayweek directly in fact_crash makes it easy to filter or group for the most common time questions we had for this analysis.

We believe this design is a sensible way to set things up for analyzing this specific road fatality data, balancing standard practices with the specific goals we had in mind.

3. Analytical Capabilities: Business Questions

3.1 Target Business Questions

The data warehouse is designed to answer questions such as:

1. Temporal Patterns: What is the relationship between the time of day, day of the week, and the frequency of fatal crashes? Are there distinct peak periods or differences between weekdays and weekends? (Addresses Vis 5.2.1)
2. Demographic Profiles & Context: What are the key demographic profiles (Age Group, Gender, Road User) most frequently involved in fatal crashes, and how do these profiles intersect with geographic remoteness and speed environments? (Addresses Vis 5.2.2 - the dashboard)
3. Long-Term Trends & Interventions: How have fatal crash counts trended annually across Australian states since 1989, and do these trends show potential correlations with the approximate introduction periods of major road safety interventions like speed cameras and Graduated Driver Licensing? (Addresses Vis 5.2.3 - the state trend map)
4. Speed & Location Interaction: How does the distribution of fatal crashes across different speed limit zones (grouped by typical environment) differ between major cities versus regional and remote areas? (Addresses Vis 5.2.4)
5. Geographic Distribution & Population: How does the absolute number of fatal crashes distribute geographically across states, and how does the population-adjusted fatality risk (rate per 100k) distribute across Local Government Areas (LGAs), revealing potential discrepancies between absolute volume and relative risk? (Addresses Vis 5.2.5 - both maps combined)

3.2 Query Footprints

1. **Q1 (Temporal Patterns - Time/Day):** Requires accessing `fact_crash` (for `Crash_ID` count) and linking to `Dim_Time` (for `Time`, `Time_of_Day`, `Time_Period`) and potentially using the `Dayweek` attribute directly from `fact_crash`.

2. **Q2 (Demographic Profiles & Context):** Primarily requires `fact_fatalities` (for `Age_Group`, `Gender`, `Road_User`, fatality count). To analyze intersections with context, it needs joins back to `fact_crash` and then onwards to:
    - `Dim_Remoteness` (for `Remoteness_Areas`).
    - `dim_speed_limit` (for `Speed_Value` / Speed Group).
    - *(Implicitly uses `fact_crash.Year` for time trends shown in the dashboard).*

3. **Q3 (Long-Term Trends & Interventions):** Requires `fact_crash` (for `Crash_ID` count and `Year`) and linking to `Dim_State` (for `State`). *(If calculating state-level rates, it also requires aggregating `fact_population_wide` linked via `Dim_State`).*

4. **Q4 (Speed & Location Interaction):** Requires `fact_crash` (for `Crash_ID` count) and linking to `dim_speed_limit` (for Speed Group) and `Dim_Remoteness` (for `Remoteness_Areas`).

5. **Q5 (Geographic Distribution & Population):** involves two parts and requires multiple sources:
    - **State-Level Absolute Counts Map:** Requires `fact_crash` (for `Crash_ID` count and `State_ID`), `Dim_State` (for mapping state names), and the **`STE_2021_AUST_GDA94.geojson`** data source (for `Geometry` and the state name identifier used for linking).
    - **LGA-Level Rate per 100k Map:** Requires `fact_crash` (for `Number_Fatalities`, `Year`, and `LGA_name` / `LGA_code`), the **`lga_pop`** data source (for `Population` figures linked via `LGA_name` / `LGA_code`), and the **`LGA_2021_AUST_GDA94.geojson`** data source (for `Geometry` and the `LGA_name` / `LGA_code` used for linking). Also requires the `[Select Year]` parameter.

---

4. Data Integration: ETL Process

4.1 Data Sources and Tools Used

- **Sources:** ARDD XLSX files [3], Population XLSX, GeoJSON files.
- **Tools:** Python with libraries including Pandas (for data manipulation), NumPy (for numerical operations), PostgreSQL as the database, for database management.

4.2 ETL Workflow Overview

The ETL process followed a standard workflow:

1. **Extract:** Reading data from source Excel into Pandas DataFrames.
2. **Explore & Profile:** Initial inspection and profiling of the raw data to understand its structure, data types, and potential quality issues, particularly focusing on identifying various forms of missing or invalid data.
3. **Transform:** Cleaning data, handling missing/special values, standardizing formats, deriving new attributes, generating surrogate keys for dimensions, and structuring data according to the target warehouse schema.

4. **Load:** Populating the dimension tables first, followed by the fact tables in the PostgreSQL database, respecting foreign key dependencies.

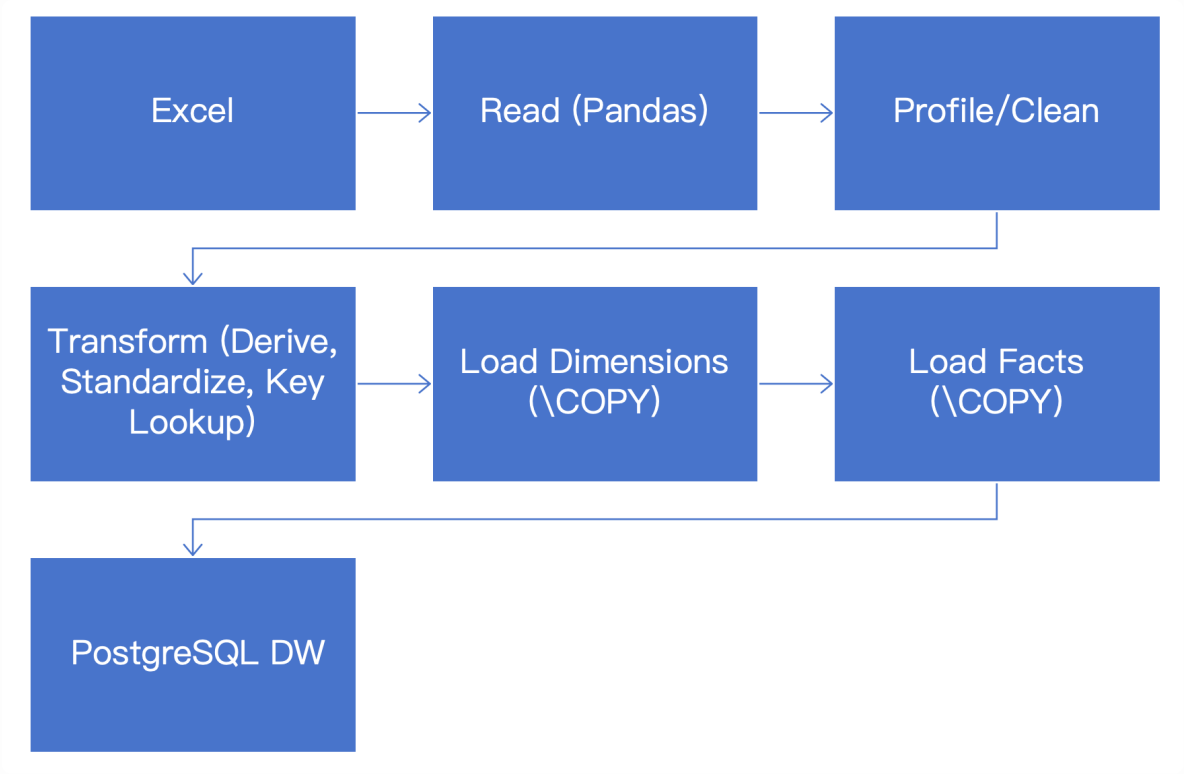


Figure 2: ETL Workflow chart

4.3 Extraction and Initial Exploration

Data was extracted from source files using Pandas `pd.read_excel` for XLSX. Deatils in `ETL.ipynb`

Following extraction, we did an initial exploration to understand data quality. This involved using tools like Excel for a visual overview and Pandas functions (.info(), .describe(), .value_counts()) for programmatic assessment.
A key early step was to identify the prevalence and different representations of missing or invalid data across various columns. Common indicators observed included actual NaN values, as well as specific string values like **blank strings ('), 'Unknown', and 'Undetermined'.** Additionally, the ARDD data dictionary indicated that the numeric code -9 often represented 'Unknown'.

To systematically quantify these varied forms of missing vlues, the following Python function using Pandas was built:

```
crash_df = pd.read_excel(
    'bitre_fatal_crashes_dec2024.xlsx',
    sheet_name='BITRE_Fatal_Crash',
    skiprows=4
)
```

[2] ✓ 3.1s

```
def count_missing(series):
    # check NaN (missing) values first
    mask = series.isna()
    # For numerical data, count -9 as missing
    if pd.api.types.is_numeric_dtype(series):
        mask |= (series == -9)
    # For object data, convert values to string, then count empty strings,
    # 'Unknown', or 'Undetermined' as missing
    elif series.dtype == object:
        mask |= series.fillna('').astype(str).str.strip().isin(['', 'Unknown', 'Undetermined'])
    return mask.sum()

missing_counts = crash_df.apply(count_missing)
# sort missing values count and print
print(missing_counts.sort_values(ascending=False))
```

[5] ✓ 0.1s

```
SA4 Name 2021                41117
National LGA Name 2021       41116
National Road Type           41115
National Remoteness Areas    40811
Time of Day                     40
Time                            39
Day of week                     12
Speed Limit                      0
Easter Period                    0
Christmas Period                 0
Crash ID                         0
State                            0
```

Figure 3: python count_missing in crash excel script and results using jupyter notebook

Same method applied to fatalities excel as well.

```
fatality_df = pd.read_excel(
    'bitre_fatalities_dec2024.xlsx',
    sheet_name='BITRE_Fatality',
    skiprows=4
)
```

[8] ✓ 3.8s

```
missing_counts_fatality = fatality_df.apply(count_missing)

print(missing_counts_fatality.sort_values(ascending=False))
```

[9] ✓ 0.1s

```
National Road Type               46136
SA4 Name 2021                    45851
National LGA Name 2021           45849
National Remoteness Areas        45520
Age                                116
Time of day                         44
Time                                43
Day of week                         13
Road User                           11
Articulated Truck Involvement        0
Speed Limit                          0
State                                0
Gender                               0
Heavy Rigid Truck Involvement        0
Bus Involvement                      0
Crash Type                           0
Dayweek                              0
Year                                 0
Christmas Period                     0
Easter Period                        0
Age Group                            0
```

Figure 4: python count_missing in fatalities excel script and results using jupyter notebook

Key Findings

- Critical Location Missing: A very high number of records lack essential geographic context(e.g. SA4 Name 2021,41,117 missing, including blank and unknown), National LGA Name 2021 (41,116 missing), National Road Type (41,115 missing), and National Remoteness Areas (40,811 missing) show widespread missingness according to our definition (including blanks, 'Unknown', 'Undetermined', NaN, -9). This affects a vast majority of the dataset.
- Minor Time Missingness: A small number of records are missing time-related information: Time of Day (40 missing), Time (39 missing), and Day of week (12 missing).
- Minor Age Missingness: A small number of exact age records are missing (116). But no missing data in age_group.
- Core Fields Complete: Crucially, core identifiers and outcome fields like Crash ID, State, Year, Month, Number Fatalities, and basic Crash Type appear complete (0 missing values based on the defined criteria). Speed Limit and vehicle involvement flags also

showed no missing values according to the function's logic (likely encoded differently or truly complete).

Assumption and Action Plan:
The missing data profile dictates the following approach:

1. Assumption 1 (Location Data Incompleteness): Given the overwhelming lack of usable LGA, SA4, Remoteness and road type information in the majority of records, detailed sub-state geographic analysis using these specific fields from the source file seems not feasible for the bulk of the data. The analysis will rely on the State field and the LGA_name field present in the dataset, unknown/missing data keeps as unknown.
2. Action on Time Data: The Time of Day field has only 40 missing entries. An examination revealed these records also lack Time and Day of week information. Lacking any time context makes these records less useful for temporal analysis and potentially indicates broader data quality issues for these specific entries. Therefore, these 40 records missing Time of Day will be excluded from the dataset during transformation. This constitutes a negligible portion (<0.1%) of the total data and simplifies the creation and population of the Dim_Time dimension, ensuring all entries in the dimension have valid time attributes.
3. Assumption 2 (Handling Other Missing Values): For attributes where missingness is handled by specific codes (like potentially -9 in some original numeric fields not shown in this count summary, or 'Unknown' categories in dimensions), these will be mapped to appropriate 'Unknown' categories in the target warehouse structure (as detailed in Section 4.4) rather than causing row exclusion.

4.4 Transformation Details

This involved several key steps: Deatils can be found at `ETL.ipynb` - `dimension tables` & `fact tables` section

- **Data Cleaning:**
    - Consistency Checks: Basic checks were performed, such as ensuring Crash_IDs in crash excel are unique, and all Crash_ID values present in the fact_fatalities data existed in the fact_crash data before attempting joins or loading, preventing orphaned records.
    - Handling `-9` (Unknown/missing): such as In Gender (fatalities): Mapped to the string 'Unknown'. In dim_vehicle flags (bus_involved, etc.): Retained as the value -9, representing a distinct 'Unknown Involvement' state alongside 1 (Yes) and 0 (No).
    - Handling Blank strings, explicit 'Unknown', or 'Undetermined' text values in source columns (e.g., Road_Type, Remoteness_Areas) were generally **retained. They were treated as valid categories and loaded into the respective dimension tables, preserving the uncertainty present in the source data.**

- **Standardization and Derivations:**
    - Type Conversion: This was a primary focus. All columns were checked and converted to their intended database types using Pandas. Numeric fields were set to appropriate integer types (INTEGER, SMALLINT for flags). Text fields were standardized to string types. **Some dimension tables examples:**

    - `Dim_Time`: Extracted hour from source time, derived `Time_of_Day` ('Day'/'Night'; **Note that: Time_of_Day in original dataset has logical problem, as shown in figure 5 or crash_id 20117019 for example, data was categorized into wrong day/night even though the `ardd_dictionary` shows "Note: 'Day' refers to 6am through to 5:59 pm". so I restructure Time_of_Day based on the extracted hour**) and `Time_Period` based on the hour (divided by six equally such as "Midnight","Early Morning", methods refere to figure 6).

Crash ID	State	Month	Year	Dayweek	Time	Time of Day
20192137	Vic	10	2019	Monday	11:04	Night
20192102	Vic	9	2019	Friday	10:50	Night
20192070	Vic	9	2019	Thursday	10:50	Night
20192182	Vic	8	2019	Monday	11:15	Night
20192034	Vic	7	2019	Thursday	11:20	Night
20192184	Vic	6	2019	Friday	12:00	Night
20192109	Vic	4	2019	Friday	11:15	Night
20192232	Vic	4	2019	Monday	11:10	Night
20192167	Vic	3	2019	Monday	11:32	Night
20192185	Vic	2	2019	Sunday	10:50	Night
20192044	Vic	2	2019	Sunday	12:00	Night
20192033	Vic	2	2019	Tuesday	11:52	Night
20192042	Vic	1	2019	Thursday	11:25	Night

Figure 5: inconsistence of time and time_of_day in original data

```python
# update only keep time data now, keep month, year and weekday in fact table
# there are 39 blank for time, 40 unknown for time_of_day. and they have no location data -> remove directly from raw dataset,

crash_df['Time'] = crash_df['Time'].astype(str).str.strip()
crash_df['Time'] = pd.to_datetime(crash_df['Time'], format='%H:%M:%S', errors='coerce').dt.hour
crash_df['Time'] = crash_df['Time'].astype(int)

# 2. Select 'Time', drop duplicates, sort
dim_time_df = crash_df[['Time']].drop_duplicates().sort_values(by='Time', ascending=True).reset_index(drop=True)

# Add 'Time_ID' column (starting from 1)
dim_time_df.insert(0, 'Time_ID', range(0, len(dim_time_df)))

def get_time_of_day(hour):
    if 6 <= hour <= 18:
        return 'Day'
    elif 0 <= hour <= 5 or 19 <= hour <= 23:
        return 'Night'
    else:
        return 'Unknown'

dim_time_df['Time of Day'] = dim_time_df['Time'].apply(get_time_of_day)
# 5. Define time period based on hour
def get_time_period(hour):
    if 0 <= hour <= 3:
        return 'Midnight'
    elif 4 <= hour <= 7:
        return 'Early Morning'
    elif 8 <= hour <= 11:
        return 'Morning'
    elif 12 <= hour <= 15:
        return 'Afternoon'
    elif 16 <= hour <= 19:
        return 'Evening'
    elif 20 <= hour <= 23:
        return 'Late Night'
    else:
        return 'Unknown'

dim_time_df['Time Period'] = dim_time_df['Time'].apply(get_time_period)

# Export to CSV without the DataFrame index
dim_time_df.to_csv('dim_time.csv', index=False)

print(f" {len(dim_time_df)} unique time combinations.")
print(dim_time_df.head())
```

```
 24 unique time combinations.
   Time_ID  Time Time of Day    Time Period
0        0     0       Night       Midnight
1        1     1       Night       Midnight
2        2     2       Night       Midnight
3        3     3       Night       Midnight
```

Figure 6: python script regarding time restructure and export dim_time

- `dim_vehicle`: Converted Yes to 1, No to 0, Unknown to -9. Populated the `Description` field based on the flag values.

- `dim_speed_Limit`: Speed Limit column was handled: values reading '<40' were mapped to the integer 35 (affecting only a small number of rows), and representations of Unknown speed limit keeps as -9, allowing the Speed_Value column to be consistently numeric

- Each dim table handling methods can be found at `ETL.ipynb` - `dimension tables` section and subsections(total 8). All csv files generated from scripts and used for importing submitted. Such as `dim_crash_type.csv`, total 8 csv files for dimension tables, `fact_crash_1.csv` for fact_crash, `fact_fatalities.csv` for fact_fatalities,and `fact_population.csv` for fact_population_wide.

- **Note that:** 1. `speed.csv` is the correct version for `dim_time`, `dim_speed_limit.csv` had mapping problem, which I illustrated in **Section4.6 troubleshooting**; 2. `(NotUsed)dim_location.csv` is one of the trial we made for building the dw, but did not use for the final crash dw

- **Data Reduction:**
  As determined during initial exploration (Section 4.3), the 40 records identified as missing Time_of_Day (same crash_id as missing Time and Day_of_week) were excluded from the dataset prior to loading. This represents less than 0.1% of the total records and ensures the integrity of time-based analysis and the Dim_Time dimension.
  No other significant row exclusions were performed

- **Mapping:**

  - Dimension tables were populated first by selecting distinct attribute combinations from the source data. Surrogate keys (`SERIAL` or generated integers like for `dim_speed_limit`) were assigned.

  - For each dimension, a dictionary was defined mapping the unique natural key value(s) from the source data to the corresponding pre-defined or database-generated surrogate ID. *note: Generation AI was used to write mapping method since there are too many fields and to avoid mistakes*

  - During fact table loading, to replace the natural keys and descriptive values in the fact_crash data with the appropriate foreign key (surrogate ID) values, Python dictionaries were used as mapping lookup tables within the ETL script.

  - For `fact_population_wide` table, I used `read_df` to read the raw data first, then I decided to **directly extracte from dataset using Excel(also for the data cleaning process)** -> sheet_name='Table 3'-Estimated resident population, Remoteness Areas, Australia ->keep State, Remoteness Areas and each year's data with col name per year, which can be found at `state_remote_population.xlsx` -> Then mapping to each ID -> generated `fact_population.csv` for importing into PostgreSQL.

  - *Note that: We also did few trials on how to better use the raw population dataset and then generate our fact table. our footprints can be found at `(NotUsed)only_remote_population.xlsx` , `(NotUsed)population_lga.xlsx` , and `etl_population.ipynb`*

Examples:

`state_mapping` : Maps state abbreviations (e.g., 'WA', 'NSW') to their State_ID;

`vehicle_map` : Maps the tuple combination of the numeric flags (Bus_Involved, Heavy_Truck, Articulated_Truck) (e.g., (0, 1, 0)) to its Vehicle_ID(shown in Figure 7). Similar mapping methods were used for other dimensions.(python script can be found in file `ETL.ipynb` fact table)

```python
# Vehicle involvement mapping
vehicle_map = {
    (0, 0, 0): 1,
    (-9, -9, -9): 2,
    (0, 0, 1): 3,
    (1, 0, 1): 4,
    (0, 1, 0): 5,
    (-9, 1, -9): 6,
    (0, 1, 1): 7,
    (1, 0, 0): 8,
    (1, 1, 0): 9,
    (-9, -9, 1): 10,
    (0, -9, 0): 11,
    (0, -9, 1): 12,
    (1, -9, 0): 13,
    (1, -9, 1): 14
}


# Extract hour from Time column as Time_ID after crash_id
crash_df['Time'] = crash_df['Time'].astype(str).str.strip()
crash_df['Time'] = pd.to_datetime(crash_df['Time'], format='%H:%M:%S', errors='coerce').dt.hour
crash_df['Time'] = crash_df['Time'].fillna(0).astype(int)
fact_crash_data.insert(1, 'Time_ID', crash_df['Time'])


# Replace 'Yes', 'No', and '-9' with respective values
fact_crash_data['Bus_Involved'].replace({'Yes': 1, 'No': 0, '-9': -9}, inplace=True)
fact_crash_data['Heavy_Truck'].replace({'Yes': 1, 'No': 0, '-9': -9}, inplace=True)
fact_crash_data['Articulated_Truck'].replace({'Yes': 1, 'No': 0, '-9': -9}, inplace=True)

# Create Vehicle_ID based on Bus_Involved, Heavy_Truck, and Articulated_Truck columns
fact_crash_data['Vehicle_ID'] = list(zip(fact_crash_data['Bus_Involved'], fact_crash_data['Heavy_Truck'], fact_crash_data['Articulated_Truck']))
fact_crash_data['Vehicle_ID'] = fact_crash_data['Vehicle_ID'].map(vehicle_map)
```

Figure 7: python script regarding fact tables mapping methods

- **Data Linking:**
  - `fact_fatalities` linked to `fact_crash` using `Crash_ID`.
  - `fact_population_wide` data was cleaned, ensuring `State` and `Remoteness_Areas` and linked IDs are matched the values used in `Dim_State` and `Dim_Remoteness` for joining.

4.5 Loading Method

- Data was loaded into PostgreSQL in dependency order: Dimension tables were loaded first, followed by fact tables (`fact_crash` , `fact_fatalities` , `fact_population_wide`).
- loading method: Generated intermediate CSV files(all generated files provided) and used psql `\COPY` command for loading.

```
-- import data into dw tables using the psql command line
-- \cd to my file path first

-- Dimension Tables
\copy Dim_State FROM 'dim_state.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');

\copy Dim_Remoteness FROM 'dim_remoteness.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');

\copy Dim_Time FROM 'dim_time.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');

\copy dim_vehicle FROM 'dim_vehicle.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');

\copy Dim_Road_Type FROM 'dim_road_type.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');

\copy Dim_Crash_Type FROM 'dim_crash_type.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');

\copy Dim_Holiday FROM 'dim_holiday.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');

\copy dim_speed_limit FROM 'dim_speed_limit.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');


-- Fact Tables

\copy fact_crash FROM 'fact_crash.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');

\copy fact_fatalities FROM 'fact_fatalities.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');

\copy fact_population_wide FROM 'fact_population_wide.csv' WITH (FORMAT CSV, HEADER true, DELIMITER ',');
```

Figure 8: psql commands regarding data loading

4.6 Validation and Troubleshooting

Ensuring the accuracy of the loaded data warehouse, particularly the fact tables which integrate information from multiple sources and transformations, is a critical final step in the ETL process. While the transformation logic aimed for correctness, verifying the results in the database against the original source data is essential.

Validation check methods after the initial data load into the PostgreSQL tables:

- Random Sampling: Selecting a small, random sample of Crash_IDs from the loaded fact_crash table.
- Cross-Referencing: For these sampled Crash_IDs, querying the fact_crash, fact_fatalities, and relevant dimension tables in PostgreSQL to retrieve the loaded values (e.g., State_ID, Speed_ID, Time_ID, corresponding Road_User, Age_Group, etc.).
- Source Comparison: Manually looking up the same Crash_IDs in the original source Excel files (Crash and Fatality data) and comparing the corresponding raw values (e.g., State abbreviation, original Speed Limit text, Time, Fatality Age, Road User text).
- Verification: Checking if the loaded dimension keys and attributes correctly matched the information derived from the source data for those specific crash events. For example, confirming that the State_ID loaded in fact_crash corresponded to the correct State abbreviation found in the source file for that Crash_ID.

Troubleshooting Example & Correction: During the check, an inconsistency was identified in the loaded Speed_ID values within the fact_crash table, indicating an error in the mapping

or loading logic for that specific foreign key during the initial ETL run.
Corrective Action: Based on this finding, the ETL mapping script responsible for the Speed_ID mapping and loading was reviewed and corrected to ensure accuracy.(refer to `ETL.ipynb` troubleshooting section, `speed.csv` has the correct `Speed_ID` column linked to fact_crash and details of db update process at `troubleshoot_speed.sql`).

note that: for our project, this step was taken too late, so we made mistake in the database. The fixing process also include: ALTER CONSTRAINT(drop fk) -> create a temp table -> import correct csv ->check -> add fk back.Therefore shows the importance of Validation and Troubleshooting.

Automated ETL processes, while efficient, are susceptible to logical errors or unexpected data variations. Direct comparison of loaded data against source records for a sample set provides essential confirmation of data integrity and allows for the correction of errors (like the identified Speed_ID issue) before proceeding with analysis, thereby significantly enhancing the reliability of the final data warehouse.

5. Data Visualization

5.1 Visualization Overview and Our Approach

Interactive charts and maps were created using Tableau to explore some patterns hidden in the data warehouse, and collected into dashboards. We aim to seek more accessible overview of fatal crash trends, geographical distributions, and contributing factors based on the business questions defined in Section 3.1.

Explain chosen visualizations or metrics

When thinking about how to show the data, especially trends over time and geographic patterns, we had a few choices to make mainly focused on when, where and who.

- Looking at **Time-when**: We considered plotted things day-by-day or month-by-month. That might show short spikes, maybe around holidays (though we have a specific flag for Christmas/Easter). But we were also interested in whether big policy changes, like when speed cameras or new rules for young drivers came in, had any effect. Those kinds of changes usually show up more gradually, over years. So, for looking at long-term changes and potential policy correlations, focusing on annual (Year) trends felt like the clearest way to see the bigger picture without getting lost in monthly ups and downs.
- Looking at **Location-where**: Just mapping total crash counts usually highlights the big cities simply because that's where most people live. Australia's geography is vast, with many areas having very few residents. To get a fairer comparison of risk across different LGAs, calculating the 'Fatality Rate per 100,000 Population' seemed essential. We thought about comparing population growth to crash growth, but fatality numbers in

small LGAs can jump dramatically year-to-year (e.g., 1 vs 2 deaths is 100% growth), making growth rates potentially misleading locally. The per capita rate felt more stable and easier to understand for comparing risk between small towns and large cities, although we still need to be cautious interpreting it in very low-population areas.

- Looking at **Who is Involved**: Beyond when and where, understanding who is involved in fatal crashes is critical. Therefore, dedicated visualizations were created to dissect the data by key demographic factors available in fact_fatalities: Age_Group, Gender, and Road_User type. Analyzing these factors individually and, more importantly, in combination with each other and with context like location (Remoteness) and speed environments, helps identify specific high-risk groups and the circumstances under which they are most vulnerable. This demographic breakdown is essential for developing targeted safety interventions.

So, the visualizations that follow use these approaches – annual trends, per capita rates for detailed geography, along with breakdowns by time of day, demographics, speed, and location – to piece together a clearer understanding of the fatal crash data.

5.2 Visualizations Answering Business Questions

5.2.1 Question 1: What is the relationship between crashes, day of the week, and time of day?

To investigate temporal patterns in fatal crashes, a line chart was generated to visualize crash counts by hour of day (0–23) across days of the week.

Data Sources and Fields Used:

- `Time` — the hour of the crash (0 to 23), extracted from the Time field.
- `Day of Week` — the day the crash occurred (e.g., Monday, Tuesday…).
- `Fact_crash` — used to count the number of unique crashes.

The visualization shows crash frequency over 24 hours, color-coded by day of the week, allowing temporal comparisons between weekdays and weekends.

Tableau Implementation Steps:

1. Drag `Time` to Columns, and use it as a discrete dimension to show hour-by-hour changes.
2. Drag `CNT(fact_crash)` to Rows to calculate the number of crashes per hour.
3. Drag `Day of Week` to the Color mark card to generate separate lines for each day.

Visualization:

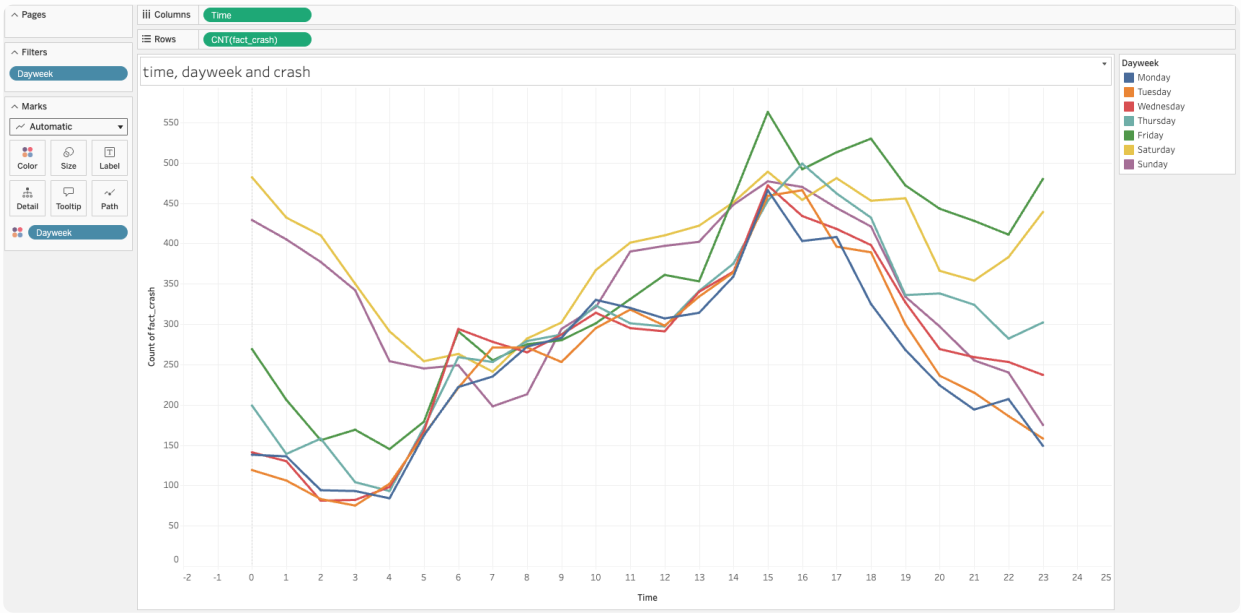


Figure 9: Temporal patterns in fatal crashes.

Interpretation of Insights:

1. **Weekday Patterns Are Consistent**
   Monday to Friday curves are highly overlapping, indicating a consistent pattern of crashes during workdays. This likely reflects routine commuting behavior and peak-hour traffic volumes.
2. **Afternoon Peak at 3 PM**
   Across all days, a noticeable spike occurs at 15:00 (3 PM), representing the daily peak for fatal crashes. This may coincide with school release hours, work shift changes, or early evening congestion.
3. **Higher Early-Morning Risk on Weekends (0:00–3:00)**
   Saturday and Sunday show distinctly elevated crash counts during the early morning hours (midnight to 3 AM) compared to weekdays. This may relate to late-night driving, drowsiness, or impaired driving behavior common during social activities.
4. **Friday and Saturday Nighttime Spikes (18:00–24:00)**
   Fatal crashes are notably higher on Friday and Saturday evenings compared to other days, highlighting elevated weekend evening risk. This likely corresponds to increased travel, social events, or alcohol-related driving.
5. **Saturday Night is Particularly Risky**
   Combining high early-morning and late-night crash counts, Saturday overall presents a heightened risk for fatal accidents compared to other days of the week.

Business Implications & Recommendations:

- Enhance enforcement and visibility during 3 PM to 6 PM, especially on weekdays, to reduce peak-hour crashes.

- Increase random breath testing and policing on weekend nights, especially Friday and Saturday, to address potential drink-driving risks.
- Deploy early-morning patrols (0:00–3:00) on weekends to discourage unsafe late-night driving behavior.

5.2.2 Question 2 Key Demographic Profiles (Age, Gender, Road User)

- Primary Question: What are the key demographic profiles (Age, Gender, Road User) most frequently involved in fatal crashes, and how do these profiles intersect with factors like geographic remoteness, speed environment, and long-term time trends?
- SubQuestions: How does the age and gender distribution differ significantly between Drivers and Motorcycle Riders involved in fatal crashes? How have the fatal crash trends differed for various Age Groups over the long term (since ~1989)? Have interventions like GDL correlated with steeper declines for younger groups compared to older groups? For Drivers within different Age Groups, what is the distribution of the Speed Limit Zones where their fatal crashes occurred? (e.g., Do younger drivers have more high-speed zone fatalities compared to older drivers?)

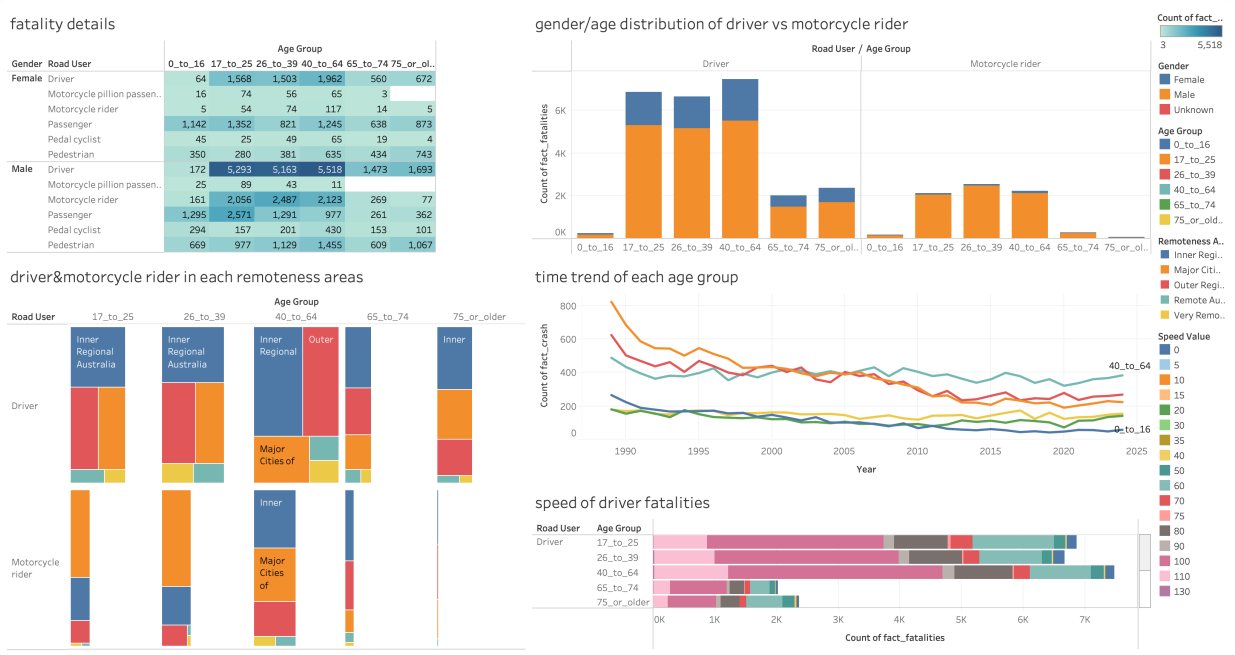


Figure 10: Key Demographic Profiles related to road fatalities.

Data Sources and Fields Used:

- `fact_fatalities` : `age_group` , `gender` , `road user` ,count of fact_fatalities
- `dim_remoteness` : remoteness_areas
- `dim_speed` : speed_value

Key Insight:

- Male Drivers & Motorcyclists Dominate: The data consistently highlights male fatalities significantly outnumbering female fatalities across most driver and motorcyclist age groups, particularly in the young adult (17-25) and middle-aged (26-64) brackets. (Based on "fatality details" heatmap and "gender/age distribution..." chart)
- Distinct Age Profiles for Drivers vs. Motorcyclists: Fatal drivers show high numbers across a broad range of adult age groups (e.g., 17-25, 26-39, 40-64). In contrast, fatal motorcyclists appear more concentrated in the young adult and middle-aged groups (e.g., peaking maybe in 26-39 or 40-64), with fewer fatalities among the very young or very old compared to drivers. (Based on "gender/age distribution..." chart).
- Geographic Variation in Driver/Motorcyclist Risk: The "remoteness areas" chart suggests potential differences in where drivers vs. motorcyclists face fatal risks. For instance, younger driver fatalities might be more spread across different remoteness areas, while older motorcyclist fatalities could be more concentrated in inner/outer regional areas. (Based on "driver&motorcycle rider in each remoteness area" chart).
- Speed Environment Varies by Driver Age: The "speed of driver fatalities" chart likely indicates a relationship between driver age and the speed zone of the fatal crash. For example, it might show that younger drivers (e.g., 17-25) have a higher proportion of their fatal crashes occurring in high-speed zones (>=100) compared to middle-aged or older drivers. (Based on "speed of driver fatalities" chart).

5.2.3 Question 3: Road Safety Interventions VS Long-term Trends in fatal crash

- Primary Question: To what extent do major road safety interventions implemented since 1989 correlate with long-term trends in fatal crash counts or rates across Australian states?
- Sub-Question (Trends & Comparison): How have fatal crash counts/rates trended annually across different Australian states since 1989, and how do these trends compare? (Holiday Periods & WA Policy): How do fatal crash counts during defined Christmas/Easter holiday periods compare to non-holiday periods, and does WA exhibit a distinct pattern potentially linked to its Double Demerits policy?

Government-implemented traffic regulations, including widespread deployment of speed cameras (common from the 1990s/2000s onwards[7]), Graduated Driver Licensing systems, and the introduction of stricter penalties, have been extensively discussed in the literature[5][6] as key factors influencing road safety outcomes.
Victoria: a new program of speed camera enforcement was introduced in December 1989. the Traffic Camera Office was established in June 1990 that the increasing number of cameras were used extensively.; NSW: Mobile speed cameras were first used in 1991; WA: The governmentstarted using speed cameras in 1988.[7]

Given our dataset spanning from 1989 to 2024, allows for visual exploration of potential correlations between the introduction periods of these broad policy shifts and changes in

fatal crash trends across states.
Additionally, we investigate specific holiday period patterns, focusing on WA's Double Demerits policy (active during holidays since ~1998) as a case study, using our Dim_Holiday flags for Christmas/Easter.
While different factors could be involved, visualizing trends alongside policy timelines can potentially reveal suggestive correlations.

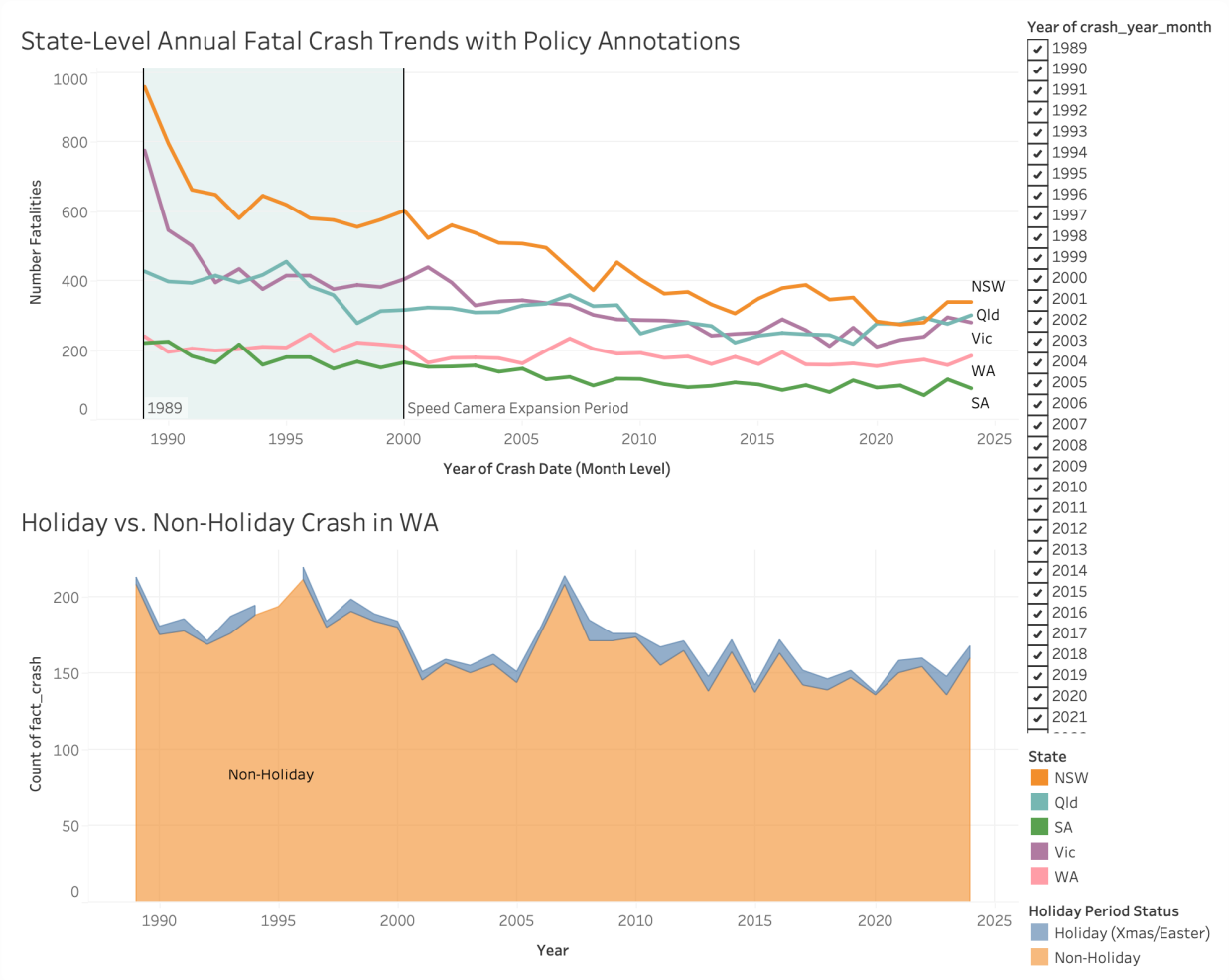


Figure 11: Key Demographic Profiles related to road fatalities.

- **Data Sources and Fields Used:**
  `fact_crash`: Crash_ID (for counting), Year (for time axis), State_ID (FK), Holiday_ID (FK).
  `Dim_State`: State (for grouping/coloring lines).
  `Dim_Holiday`: Christmas_Flag, Easter_Flag (used to derive holiday status).

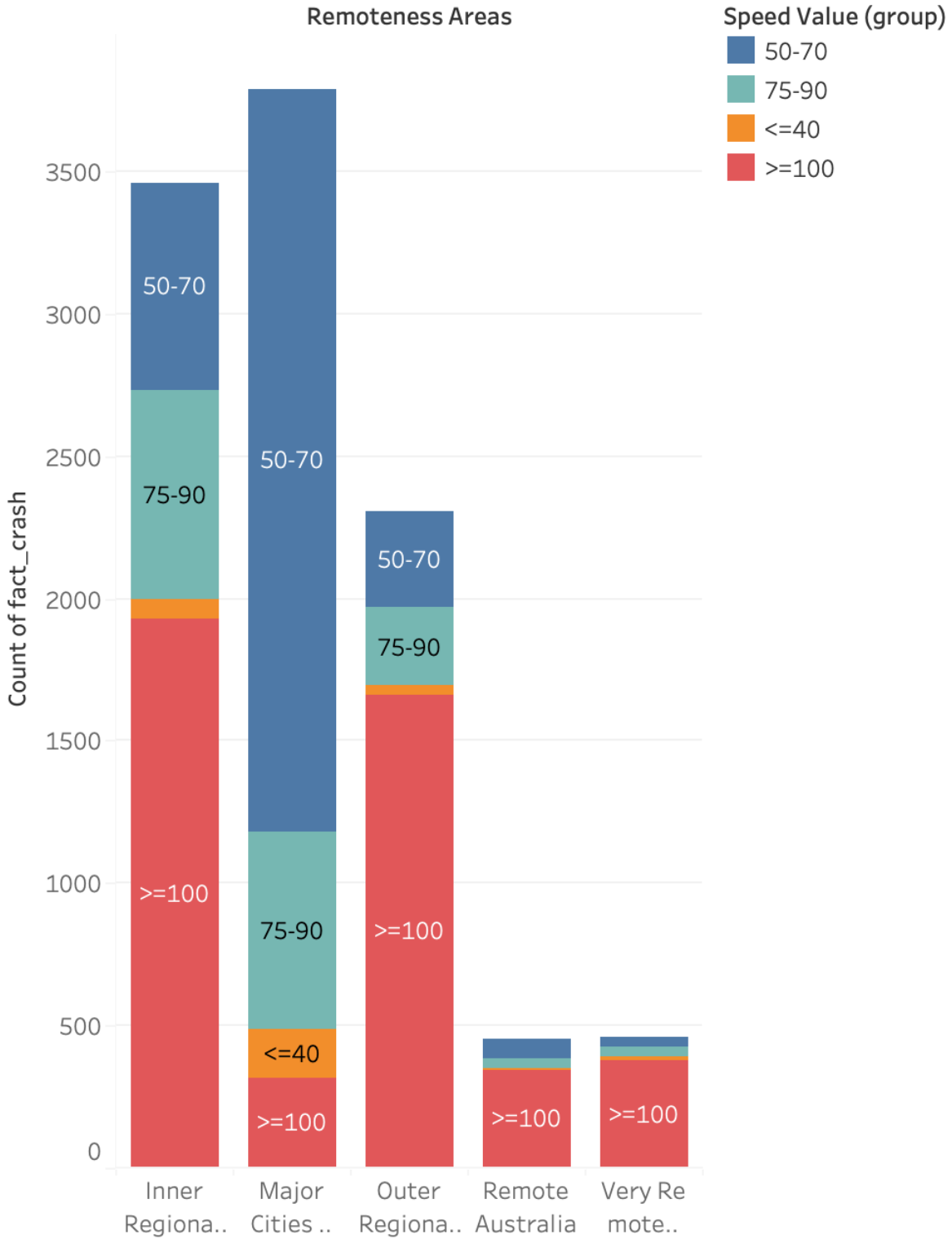
- **Key Insight:**
  The long-term view across most states generally shows a declining trend in fatal crash counts starting from the 1989 and continuing through the 2000s, roughly coinciding with the periods of wider speed camera adoption and GDL implementation. While various factors contribute, this visual correlation suggests these large-scale

interventions likely played a role in the overall improvement compared to earlier periods. However, trends differ significantly between states.

Holiday vs non-holiday in WA is not suitable for understanding the influence of WA's Double Demerits. Because Dim_Holiday only covers Xmas/Easter and cannot fully evaluate the broader Double Demerits policy. Clearly, isolating the impact of any single intervention is inherently difficult due to confounding factors.

5.2.4 How does the distribution of fatal crashes across different speed limit zones differ between major cities/regional/remote areas?

Fatal Crashes by Speed Limit Zone and Remoteness Area



Count of fact_crash for each Remoteness Areas. Color shows details about Speed Value (group). The marks are labeled by Speed Value (group). The view is filtered on Remoteness Areas, which keeps Inner Regional Australia, Major Cities of Australia, Outer Regional Australia, Remote Australia and Very Remote Australia.

Figure 12 : Fatal Crashes by Speed Limit Zone and Remoteness Area

Understanding the interplay between speed environment and geographic remoteness is important for identifying distinct risk profiles. Comparing crash frequencies in different speed zones across varying levels of remoteness helps determine if speed management strategies need tailoring for urban versus regional/remote contexts. A grouped or stacked bar chart provides a clear visual comparison of these distributions.

- Data Sources and Fields Used:
  `fact_crash`: Crash_ID (for counting), Speed_ID (FK), Remoteness_ID (FK).
  `dim_speed_limit`: Speed_Value (using tableau creat GROUP.
  there are clearly high crash rate when speed limit is 100 and 60 based on initial exploration. The groups designed to broadly represent common operational speed ranges: 'Low Speed (<=40 km/h)' (often residential/school zones), 'Urban/Suburban (50-70 km/h)' (covering the frequent 50 and 60 km/h limits where initial analysis showed a peak), 'Arterial/Regional (75-90 km/h)', 'Highway/Rural (>=100 km/h)'
  `Dim_Remoteness`: Remoteness_Areas

Key Insights:

- Fatal crashes in Major Cities predominantly occur in 'Urban/Suburban (50-70 km/h)' zones. Conversely, regional and remote areas see the majority of their fatal crashes in 'Highway/Rural (>=100 km/h)' zones.
- This highlights a clear shift in risk profile: lower/medium-speed urban risks versus high-speed rural/remote risks.
- 'Arterial/Regional (75-90 km/h)' crashes are significant in transitional zones, while 'Low Speed (<=40 km/h)' crashes are relatively infrequent everywhere.
- This chart directly answers the business question by demonstrating the need for geographically tailored road safety strategies based on dominant speed environments. The clear contrast confirms that interventions in Major Cities must prioritize risks mainly in 50-70 km/h zones (e.g., intersection safety, urban speed management), while Regional/Remote efforts require a strong focus on mitigating dangers associated with high-speed (>=100 km/h) travel (e.g., highway design, fatigue, remote enforcement). Transitional zones clearly require a blended approach.

5.2.5 How does the geographic distribution of fatal crashes risk vary across Australia at both State&LGA levels? is it related to population?

The absolute number of fatal crashes within each state can be helpful in identifying where the largest volumes of incidents occur. A map visualizing these counts provides immediate geographic context.
State-Level Fatal Crash Count Map - addressing geographic distribution of the absolute

number of fatal road crashes

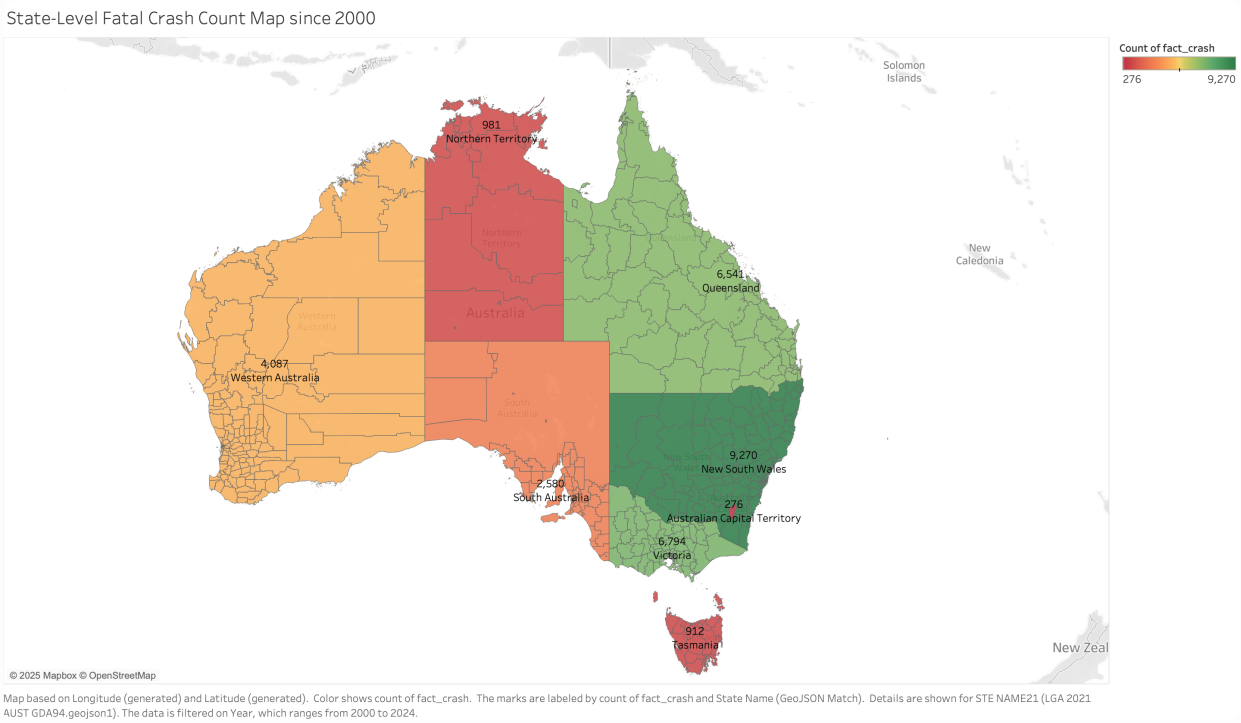


Figure 13: Fatal Crashes Count Map State-Level Since 2000

Data Sources and Fields Used:

- `fact_crash`: Crash_ID (for counting), Year (for filtering), State_ID (FK).
- `Dim_State`: State (Abbreviation or field used for mapping).
- `STE_2021_AUST_GDA94.geojson` (State GeoJSON Source): Geometry, STE_NAME21 (containing state names like 'Western Australia').
  Mapping mechanism (Calculated Field [State Name (GeoJSON Match)]) to link Dim_State.State abbreviation to the STE_NAME21 used in the GeoJSON.

Key Insights:

- This map clearly indicates that the highest absolute burden of fatal crashes falls upon the most populous eastern states: NSW (9,270 crashes), Victoria (6,794 crashes), and Queensland (6,541 crashes). WA also records a substantial number (4,087 crashes). In contrast, states and territories with significantly smaller populations, having far lower total crash counts over this period.
- This distribution strongly correlates with **population density and overall traffic volume**, highlighting the scale of the road safety challenge in the larger states. While essential for understanding the total volume of incidents per jurisdiction, this map does not reflect the relative risk per capita, which requires normalization by population (as explored in next map - the LGA rate map).

LGA Fatality Rate per 100,000 Population Map - addressing geographical risk distribution relative to population

While maps showing raw fatality counts identify areas with the most incidents, they are heavily biased towards densely populated LGAs. Calculating the fatality rate per 100,000 population normalizes for population differences, providing a more equitable measure of road safety risk across diverse LGAs. Identifying LGAs with high fatality rates, even if absolute numbers are low, highlights areas where the risk relative to the resident population is significant, potentially indicating infrastructure issues, high through-traffic risks, or other localized factors needing attention. This visualization is crucial for targeted resource allocation and safety interventions.

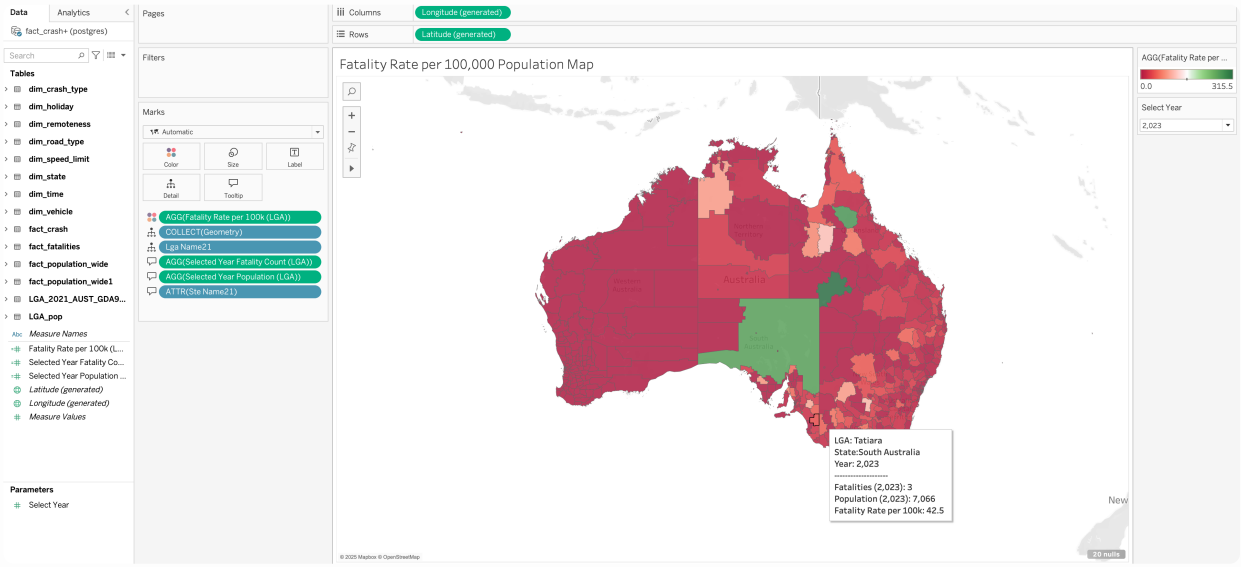


Figure 14: Road Fatality Rate per 100,000 Population by LGA for [2023].

Data Sources and Fields Used:

1. `fact_crash`:
   - `Crash_ID`
   - `Number_Fatalities`: Aggregated to get fatality counts per LGA/Year.
   - `Year`: Used for filtering and linking to the population year.
   - `LGA_name`: Used as the primary key to link crash data to population and GeoJSON data.

2. `LGA_pop` **(Excel Source - Wide Format):**
   I extract and format this table from `Population estimates by LGA, Significant Urban Area, Remoteness Area, Commonwealth Electoral Division and State Electoral Division, 2001 to 2023.xlsx` sheet `Table 1`
   - `LGA code`: The linking key.
   - Columns `2001`, `2002`, ..., `2023`: Contain the Estimated Resident Population (ERP) for each LGA for each year. These are accessed dynamically via a calculated field.

3. **GeoJSON Source:** `Geometry`, `LGA Code21`, `Ste Name21`

Tableau Implementation Steps: [appendix 2: relationships, calculated fields, Tooltip]

Key Insights:

This LGA-level fatality rate map for [2023] reveals considerable geographic variation in road safety risk relative to population. While most LGAs exhibit low-to-moderate rates, a number of sparsely populated regional and remote LGAs display markedly elevated rates (>50 in yellowish color or >100 per 100k in dark green).

But we have to interpret these high rates with caution; they often reflect the significant statistical impact of even a single fatality within a very small resident population base, rather than necessarily indicating consistently extreme underlying risk for residents themselves. The specific LGAs appearing as high-rate 'hotspots' demonstrate considerable volatility year-on-year, suggesting these high rates in low-population areas may be driven by infrequent but high-consequence events, potentially involving through-traffic on major routes within those LGA boundaries.

Therefore, while highlighting areas where fatalities have a disproportionate impact relative to local population, this map underscores the need for analyzing multi-year trends and considering factors beyond resident population when assessing risk in geographically vast and sparsely populated regions like much of Australia

Limitations:

- Rates calculated for very low population LGAs can be volatile year-on-year due to small changes in fatality numbers having a large percentage impact.
- The rate reflects fatalities occurring *within* the LGA boundary relative to the *resident* population, not necessarily the residency of the person involved in the crash. High rates in some LGAs might be influenced by crashes involving non-residents on major highways passing through.

6. Data Mining: Association Rules for Road Safety

scripts can be found at `Association_Rules_Mining.ipynb`

6.1 Data Preparation for ARM

- Data for association rule mining was prepared by merging relevant attributes from the **BITRE_Fatal_Crash** and **BITRE_Fatality** datasets on the `Crash ID` field.
- The following attributes were selected based on their interpretability and relevance to road user involvement:
    - `Gender`

    - `Age Group`
    - `Crash Type_fatal` (e.g., Single or Multiple vehicle crash)
    - `Bus Involvement`
    - `Articulated Truck Involvement_fatal`
    - `Speed Limit_fatal` (binned into discrete intervals: `0-50`, `51-80`, `81-100`, `101-130`, `131+`)
    - `Christmas Period_fatal`
    - `Easter Period_fatal`
    - `Day of week_fatal`
    - `Time of day`
    - `Road User` (used as the target on the RHS of association rules)
- **Numerical attributes**, such as `Speed Limit_fatal`, were handled by:
    - Replacing non-numeric values (e.g., `'<40'`) with valid equivalents (e.g., `'40'`)
    - Converting to float using `pd.to_numeric(errors='coerce')`
    - Binning into categorical ranges using `pd.cut()`
- **Missing values** were represented as `-9` in the dataset and were converted to `NaN` for proper handling. Rows containing missing values in selected attributes were removed for the purpose of ARM.
- Categorical values were transformed into transactional format by converting each row into a list of `attribute=value` pairs (e.g., `'Crash Type_fatal=Single'`, `'Gender=Male'`). Each list became a transaction suitable for association rule mining.
- The `mlxtend.preprocessing.TransactionEncoder` was used to convert the transaction list into a one-hot encoded Boolean DataFrame required by the Apriori algorithm.
- Attributes with excessive missing values (e.g., `National Road Type`, `National Remoteness Areas`) were **excluded** from analysis due to high frequency of `'Unknown'` values, which would otherwise bias the results.

6.2 Algorithm Selection and Setup

This project employs the **Apriori algorithm** for association rule mining, implemented using the `mlxtend` Python library. The algorithm is chosen for its efficiency in finding frequent itemsets in Boolean transactional data and its support for rule-based interpretation.

Each row in the dataset (representing a fatality record) was transformed into a transaction, where every field was converted into the format `attribute=value`. For example, a fatality record with a male driver involved in a single vehicle crash would be represented by items such as: `"Gender=Male"`, `"Road User=Driver"`, `"Crash Type_fatal=Single"`, etc.

The following configuration was applied for rule mining:

- **Frequent itemsets** were identified using `apriori()` with:
  - `min_support = 0.05` to retain itemsets that occurred in at least 5% of the transactions.
- **Association rules** were extracted using `association_rules()` with:
  - `metric = 'confidence'`
  - `min_threshold = 0.5` to filter out weak rules.
- Rules were further filtered to include only those where `Road User` appeared in the **consequent (right-hand side)**.
- Top rules were ranked based on **Lift** and **Confidence**, to ensure both statistical significance and practical strength of association.

To ensure quality of rules:

- The minimum confidence threshold was tuned from `0.5` to `0.7` to evaluate rule stability.
- Only rules with **Lift > 1.0** were retained for interpretation, ensuring that they provide information beyond random co-occurrence.

6.3 Top-k Rules with 'Road User' on RHS

Antecedents	Consequents	Support	Confidence	Lift
Articulated Truck Involvement=Yes, Bus Involvement=No	Crash Type=Multiple, Road User=Driver	5.01%	50.3%	2.32
Articulated Truck Involvement=Yes, Easter Period=No	Crash Type=Multiple, Road User=Driver	5.03%	50.1%	2.31
Articulated Truck Involvement=Yes	Crash Type=Multiple, Road User=Driver	5.05%	50.1%	2.31
Articulated Truck Involvement=Yes, Gender=Male	Road User=Driver, Bus Involvement=No, Christmas Period=No, Easter=No	5.00%	66.3%	1.53
Articulated Truck Involvement=Yes, Gender=Male, Easter Period=No	Bus Involvement=No, Christmas Period=No, Road User=Driver	5.00%	66.6%	1.53

6.5 Interpretation in Plain English

- If a **fatal crash involves an articulated truck**, there's a high probability that:
    - The crash is of type **multiple vehicles**.
    - The **road user who died was a driver**, not a pedestrian or passenger.
    - The crash **did not involve a bus**, and it was **not during Easter or Christmas**.

- For crashes involving **articulated trucks and male drivers**, about **66% of the time** the victim was a **driver**, and it was **not a holiday period**.

- **Lift > 2.3** suggests that articulated truck involvement is strongly associated with multiple-vehicle crashes leading to driver fatalities.

6.6 Insights from the Rules

1. **Articulated trucks** play a significant role in fatal crashes, especially **multi-vehicle** crashes involving **drivers**.
2. Fatalities **rarely occur during holidays** when buses are not involved — possibly because of heightened safety campaigns.
3. **Male drivers** are overrepresented in fatal crashes involving heavy vehicles, consistent with national statistics showing higher male driving exposure.

6.7. Recommendations for Government Policy

1. **Enhance enforcement and safety inspections for articulated trucks**
   Due to their strong link to multi-vehicle fatal crashes, articulated trucks should face stricter maintenance checks, speed monitoring, and fatigue compliance.

2. **Targeted driver safety training for high-risk demographics (e.g. males driving heavy vehicles)**
   Safety campaigns and certification programs can be made mandatory for male drivers handling heavy trucks, particularly during non-holiday periods when risk is high.

3. **Expand bus lanes and separate articulated truck routes in high-volume corridors**
   Since articulated truck crashes rarely involve buses, infrastructure adjustments can minimize interactions and improve safety for vulnerable road users.

7. Conclusion

7.1 Summary of Key Findings

In this project, we successfully designed and implemented a data warehouse using Australian road fatality and population data, enabling a multi-faceted analysis of crash patterns. Building this warehouse allowed us to move from simple counts and explore trends, demographics, and geographic risks in more detail.

Our key findings from the visualizations and analyses include:

- Temporal Patterns: Fatal crashes exhibit clear time-based patterns, with consistent weekday peaks around the afternoon commute (especially 3 PM) and significantly elevated risks during weekend late nights/early mornings (midnight-3 AM) and evenings (Friday/Saturday 6 PM onwards).
- Demographic Risk Profiles: Males, particularly as drivers and motorcyclists in the 17-64 age range, represent the majority of fatalities. However, the specific age distribution varies by road user type, and older pedestrians (75+) also emerge as a vulnerable group. Long-term trends show declines across most age groups since the 1990s, potentially correlating with major policy interventions like GDL, although the rate of change varies.
- Geographic & Environmental Context: A stark difference exists between urban and non-urban areas. Major cities see most fatal crashes in lower/medium speed zones (50-70 km/h), while regional and remote areas are dominated by high-speed (>=100 km/h) incidents. Geographically, while absolute crash numbers are highest in populous eastern states, calculating the fatality rate per 100k population reveals disproportionately high relative risk in numerous sparsely populated LGAs across the country, though this metric shows significant year-to-year volatility in those areas.
- Specific Crash Factors (from ARM): Association rule mining strongly highlighted incidents involving articulated trucks as being frequently associated with multiple-vehicle crashes where the driver was the victim, especially for male drivers outside of major holiday periods.

Collectively, these findings provide valuable insights into when, where, who, and under what circumstances fatal crashes are most likely to occur in Australia, supporting the aim of using data to understand and potentially mitigate road trauma.

7.2 Project Limitations

- Data Scope: The ARDD dataset, while detailed on fatalities, lacks crucial contextual information like real-time weather, specific road geometry beyond basic type, traffic volume, and details on non-fatal or injury crashes. This inherently limits our ability to perform deep causal analysis or build comprehensive risk models.
- Data Quality: There are significant missing values in certain source fields (particularly geographic related LGA), which restricted some potential avenues of analysis. We also identified and corrected logical inconsistencies in the raw Time_of_Day data.

- Design Simplifications: To manage complexity and focus on key project goals, we made design choices like using degenerate dimensions in fact_fatalities and simplifying Dim_Time, Dim_Crash_Type, and Dim_Holiday. This streamlined the model but reduced the potential for very granular analysis in those specific dimensions (e.g., detailed crash mechanism analysis or analysis of all public holidays).
- LGA Rate Volatility: As discussed, the fatality rate per 100k for LGAs with very small populations is highly sensitive to small numbers of incidents, making year-on-year comparisons potentially volatile and requiring cautious interpretation.
- ARM Limitations: Association rules identify interesting co-occurrences (correlations), not causation. The required transactional data format also simplifies potentially complex real-world interactions.
- ETL Process: Data cleaning and mapping (like handling '-9' codes or '<40' speed limits) involved some necessary interpretations and assumptions based on the data dictionary and observed data.

7.3 Future Work

Future work might enhance this analysis:

- **Incorporate Additional Data:** The most significant enhancement would be integrating additional datasets, if available, such as detailed weather records (e.g., from BOM), traffic volume counts, more specific road characteristic data (curvature, gradient, intersection type from mapping services or asset databases), and non-fatal crash data for comparative severity analysis.
- **Refine Dimensions:** If more detailed source data becomes available, developing a full Dim_Date, a more granular Dim_Crash_Type, or a richer Dim_Location (potentially incorporating SA2 or mesh block data if feasible) could enable deeper insights.
- **Advanced Analytical Techniques:** Move towards predictive modeling (e.g., using machine learning to predict crash severity risk based on conditions) or clustering techniques to identify geographically distinct crash hotspots based on multiple factors.

8. References

[1] Bureau of Infrastructure and Transport Research Economics, International Road Safety Comparisons 2023, Australian Government, Dec, 2023. [Online]. Available: https://www.bitre.gov.au/publications/ongoing/international_road_safety_comparisons. [Accessed: 02-Apr-2025].
[2] Bureau of Infrastructure and Transport Research Economics, "Road Deaths Australia: July 2024," Australian Government, Jul. 2024. [Online]. Available: https://www.bitre.gov.au/sites/default/files/documents/rda_jul2024.pdf. [Accessed: 02-Apr-2025].

[3] Bureau of Infrastructure and Transport Research Economics, "Fatal Road Crash Database," Australian Government. [Online]. Available: https://www.bitre.gov.au/statistics/safety/fatal_road_crash_database. [Accessed: 20-Mar-2025].
[4] R. Kimball and M. Ross, *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*, 3rd ed. Indianapolis, IN: Wiley, 2013.
[5] Z. Ebrahim and H. Nikraz, “Camera system deployment for speeding control in Australia,” International Journal of Transportation Science and Technology, vol. 3, pp. 353–364, 2014, doi: 10.1260/2046-0430.3.4.353.
[6] A. Soltani, S. Afshari, and M. A. Amiri, "Time-series projecting road traffic fatalities in Australia: Insights for targeted safety interventions," Injury, vol. 56, no. 3, article 112166, 2025, doi: 10.1016/j.injury.2025.112166.
[7] "Road speed limit enforcement in Australia," Wikipedia, The Free Encyclopedia. Wikimedia Foundation, Inc. [Online]. Available: https://en.wikipedia.org/wiki/Road_speed_limit_enforcement_in_Australia. [Accessed: 03-Apr-2025].

9. Appendices

[Appendix 1] Sample SQL queries results checking `fact_population_wide` and get linkde `fact_crash` data

- Get Population Data for All Remoteness Areas within Vic for Recent Years

```
    fpw.State_ID = 1      -- Filter by State ID for NSW
    AND fpw.Remoteness_ID = 3; -- Filter by Remoteness ID for Major Cities

-- Query 2: Get Population Data for All Remoteness Areas within a Specific State for Recent Years
SELECT
    dr.Remoteness_Areas, -- Get the name from the dimension table
    fpw._2021,
    fpw._2022,
    fpw._2023
FROM
    fact_population_wide fpw
JOIN
    dim_remoteness dr ON fpw.Remoteness_ID = dr.Remoteness_ID -- Link to get names
WHERE
    fpw.State_ID = 3      -- Filter by Vic
ORDER BY
    dr.Remoteness_Areas;

-- Get Population Data for a Specific Remoteness Area across All States for a Specific Year
SELECT
    ds.State,           -- Get the state name from the dimension table
```

Data Output | Messages | Notifications

	remoteness_areas text	_2021 integer	_2022 integer	_2023 integer
1	Inner Regional Australia	1212737	1228322	1242464
2	Major Cities of Australia	5075096	5141424	5311624
3	Outer Regional Austral...	256701	257600	258103
4	Remote Australia	3288	3285	3250

- Compare Population between Two Years for a Specific State/Remoteness Combination

```sql
ORDER BY
    fpw._2023 DESC;

-- Query 4: Compare Population between Two Years for a Specific State/Remoteness Combination
SELECT
    ds.State,
    dr.Remoteness_Areas,
    fpw._2010 AS Population_2010,
    fpw._2020 AS Population_2020,
    (fpw._2020 - fpw._2010) AS Population_Change_2010_2020 -- Calculate the difference
FROM
    fact_population_wide fpw
JOIN
    dim_state ds ON fpw.State_ID = ds.State_ID
JOIN
    dim_remoteness dr ON fpw.Remoteness_ID = dr.Remoteness_ID
WHERE
    fpw.State_ID = 4      -- Filter for Queensland
    AND fpw.Remoteness_ID = 2; -- Filter for Outer

-- Query 5: Calculate Total State Population for a Specific Year
```

Data Output | Messages | Notifications

state character varying (3)	remoteness_areas text	population_2010 integer	population_2020 integer	population_change_2010_2020 integer
Qld	Outer Regional Australia	639106	694107	55001

- Get Total Crash Count, Total Fatalities Count and Total Population for NSW in 2023 - Check CONSTRAINTs

```sql
-- Use CTE to calculate each metric separately
WITH CrashCountNSW2023 AS (
    -- Calculate the total number of crashes in NSW for 2023
    SELECT
        COUNT(fc.Crash_ID) AS total_crashes,
        SUM(fc.Number_Fatalities) AS total_fatalities
    FROM
        fact_crash fc
    JOIN
        dim_state ds ON fc.State_ID = ds.State_ID
    WHERE
        ds.State = 'NSW' -- Filter by state name
        AND fc.Year = 2023  -- Filter by year
),
PopulationNSW2023 AS (
    -- Calculate the total population for NSW in 2023
    -- Requires summing the population across all Remoteness areas for the state
    SELECT
        SUM(fpw._2023) AS total_population -- Directly reference the 2023 population column and sum it
    FROM
        fact_population_wide fpw
    JOIN
        dim_state ds ON fpw.State_ID = ds.State_ID
    WHERE
        ds.State = 'NSW' -- Filter by state name
)
-- Combine the results from the two CTEs for the final output
SELECT
    'NSW' AS State,               -- state name
    2023 AS Year,
    cc.total_crashes,           -- Get the total crash count and total fatalities from the first CTE
    cc.total_fatalities,
    pt.total_population         -- Get the total population from the second CTE
FROM
    CrashCountNSW2023 cc,       -- Reference the first CTE
    PopulationNSW2023 pt;       -- Reference the second CTE
```

Data Output | Messages | Notifications

	state text	year integer	total_crashes bigint	total_fatalities bigint	total_population bigint
1	NSW	2023	303	340	8342285

- full script at `creat_population.sql`;
- note that: GenAI was used to help structure some of these complex SQL scripts like Common Table Expressions - CTEs because writting mainly to ensure accuracy. This also shows that while the chosen schema (e.g., keeping Year/Month in fact_crash, using a wide population table) served specific design goals, it sometimes necessitates more elaborate SQL compared to, for instance, a schema with a fully comprehensive Date dimension or a pivoted population table

[Appendix 2] LGA Fatality Rate per 100,000 Population Map - Tableau Implementation Steps

1. **Established relationships** :
    - `LGA_pop` related to `GeoJSON` on `LGA Code`
    - `fact_crash` related to `LGA_pop` on `LGA name`.
    - `fact_crash` related to `GeoJSON` source on `LGA name`
2. **Calculated Field:**
    - `[Selected Year Fatality Count (LGA)]`:

```
// Calculates total fatalities for the selected year per LGA
SUM(IF [Year] = [Select Year] THEN [Number_Fatalities] ELSE 0 END)
```

- `[Selected Year Population (LGA)]`: (Using the unpivoted approach)

```
// Retrieves population for the selected year from the wide LGA_pop table
MIN(
    CASE [Select Year]
        WHEN 2001 THEN [_2001]
        WHEN 2002 THEN [_2002]
        // ... includes all years from 2003 to 2022 ...
        WHEN 2023 THEN [_2023]
        ELSE NULL
    END
)
```

- `[Fatality Rate per 100k (LGA)]`:

```
// Calculates the rate, handling potential division by zero
IF [Selected Year Population (LGA)] > 0 THEN
    ([Selected Year Fatality Count (LGA)] * 100000) / [Selected Year Popula
ELSE
    NULL
END
```

note: GenAI was used to write calculated field formulas since there are too many year fields

3. **Tooltip**
   - Edited the Tooltip to display `LGA_name`, the selected `Year` (from the parameter), the calculated `Selected Year Fatality Count (LGA)`, `Selected Year Population (LGA)`, and the formatted `Fatality Rate per 100k (LGA)`.