CITS 5504 - Data Warehousing

Project 1
Data Warehouse Design

Boya Zhang, 24324257

Lyu Lu, 24326939

11 April 2025

Contents

1. Introduction

1.1 Project Background

Traffic fatalities are a major public concern and it is one of the key indicators of road safety. Understanding the patterns and risk factors of fatal crashes can help improve public transportation networks.

In Australia, road deaths continue to be a significant issue. Based on figures released by the Department of Infrastructure, Transport, Regional Development and Communications, in 2024, fatalities reached around 1,300, which equals 4.8 deaths per 100,000 people [1].

According to the National Road Safety Action Plan 2023–2025, the Australian Government aims to reduce road fatalities by 50% and serious injuries by 30% by 2030. This strategy focuses on road safety, vehicle safety, heavy vehicle safety, and road users. It is supported by data-driven policies and requires coordination across all states [2].

This project builds a data warehouse that integrates historical fatal crash data from multiple sources. The objective is to provide a structured platform for querying and analysis, support data visualisation for key insights with tableau dashboards, and apply association rule mining to discover patterns related to high-risk factors. The findings will be used to help the government understand the importance of road safety.

The scope of analysis includes:

- **Geolocation**: identify high-risk regions at the state and LGA levels
- **Demographic characteristics**: age, gender, and road user type (e.g., pedestrian, cyclist, passenger).
- **Vehicle involvement**: crashes involving buses, heavy trucks, articulated trucks, and other vehicle types.
- **Policy-related factors**: speed zones, time of day, weekday versus weekend comparisons, and other policy-relevant aspects.

1.2 Data Sources

All data used in this project were obtained from datasets provided in the official project specification [3]. Additional population state dataset is downloaded from ABS website [4].

This includes:

- **Australian Road Deaths Database (ARDD)**

The *bitre fatal crashes* dataset (December 2024) contains crash-level information such as date, time, crash type, location (including Local Government Area), and vehicle involvement.

The *bitre fatalities* records, in addition to fatal crash information, person-level information for each fatality, including age, gender, and road user type.

- **Australian Bureau of Statistics (ABS)**

Population and *dwelling* data were used to support regional demographic analysis and calculate fatality rates per population and per dwelling. This data provides context for understanding the relative risk across areas with different population densities.

The state and territory population dataset (December 2024) was used to analyse the annual trends of fatalities across each state and territory, providing insights into performance differences among these states and territories.

- **GeoJSON spatial data**

Geospatial data files were provided for location-based visualisation. These files were used for mapping fatal crash distributions across Australian states and LGAs in Tableau.

1.3 Methodology

The project workflow includes schema design, data preparation through ETL, database implementation, result visualisation, and data mining.

A star schema is used to support multidimensional queries across different dimensions, such as time, location, demographics, and vehicle involvement. The fact table records one row per fatality and connects to multiple dimension tables.

The ETL process consists of three stages:

- **Extract**: Data was extracted from the ARDD, population, and dwelling datasets. GeoJSON files were not used during extraction but were later used for spatial visualisation.
- **Transform**: clean, reshape, categorise, and join data
- **Load**: Cleaned and transformed data was exported to CSV files in preparation for loading into the database.

The CSV files were then bulk inserted into PostgreSQL.

To use the data warehouse's capability for multidimensional analysis, this project implemented five types of business queries as examples. A StarNet model was used to illustrate each query and the results of these five business queries are shown in SQL and visualised Tableau.

Then a data mining technique, association rule mining was applied to identify frequent co-occurring factors in fatal crashes. This helped uncover combinations of attributes that are more likely to be associated with high-risk situations.

Tools used in the project:
- Python (Jupyter Notebook): data processing, cleaning, association rule mining
- PostgreSQL + pgAdmin4: database implementation
- Draw.io: schema diagrams
- Tableau: visualisations

Disclaimer: Some parts of the project involved assistance from generative AI (ChatGPT) [5]. We used it mainly to help with Python function explanations. In writing, it was also used to adjust wording. All decisions on data modelling, query logic, and analysis were made by us based on our understanding of the project requirements.

2. Data Warehouse Schema Design

2.1 Kimball's 4-Step Dimensional Design

This project follows Kimball's four-step dimensional modelling methodology to build a structured and efficient data warehouse.

Step 1: Identify the process being modelled

The business process being modelled is traffic fatalities in Australia. The warehouse is designed to support queries related to crash patterns, demographic profiles, road characteristics, vehicle involvement, and policy factors such as speed zones.

Step 2: Determine the grain at which facts will be stored

The fact table is defined at the individual fatality level. Each record in the fact table represents a single fatality.

Step 3: Choose the dimensions

A total of thirteen dimension tables were created to describe different attributes of a fatality:

- Location
- Date
- DayofWeek
- Dayweek
- Holiday
- Time
- Crash Type
- Involvement
- Road User
- Road Type
- Speed Limit
- Gender
- Age Group

Step 4: Identify the numeric measures for the facts

The fact table includes the following numeric measures:

- Fatality: always 1, since each row represents one death
- Fatality Rate LGA per 100,000 = (fatalities / LGA population) x 100,000
- Fatality Rate State per 100,000 = (fatalities / State population) x 100,000
- Fatality Rate per Dwelling = (fatalities / dwellings) x 100,000

2.2 Star Schema

Compared to the snowflake and galaxy schemas, the star schema has a simpler structure and performs efficiently in OLAP querying and aggregation. The star schema design (Figure 2.1) includes a fact table in the centre, connected to a set of thirteen dimension tables.

The fact table includes foreign key references to each dimension table, along with numeric measurements related to population, dwellings, and fatality rates (calculated per 100,000 population and dwelling). All dimension attributes are already attached to the fact table.

The diagram below shows how the fact table connects directly to each dimension table.

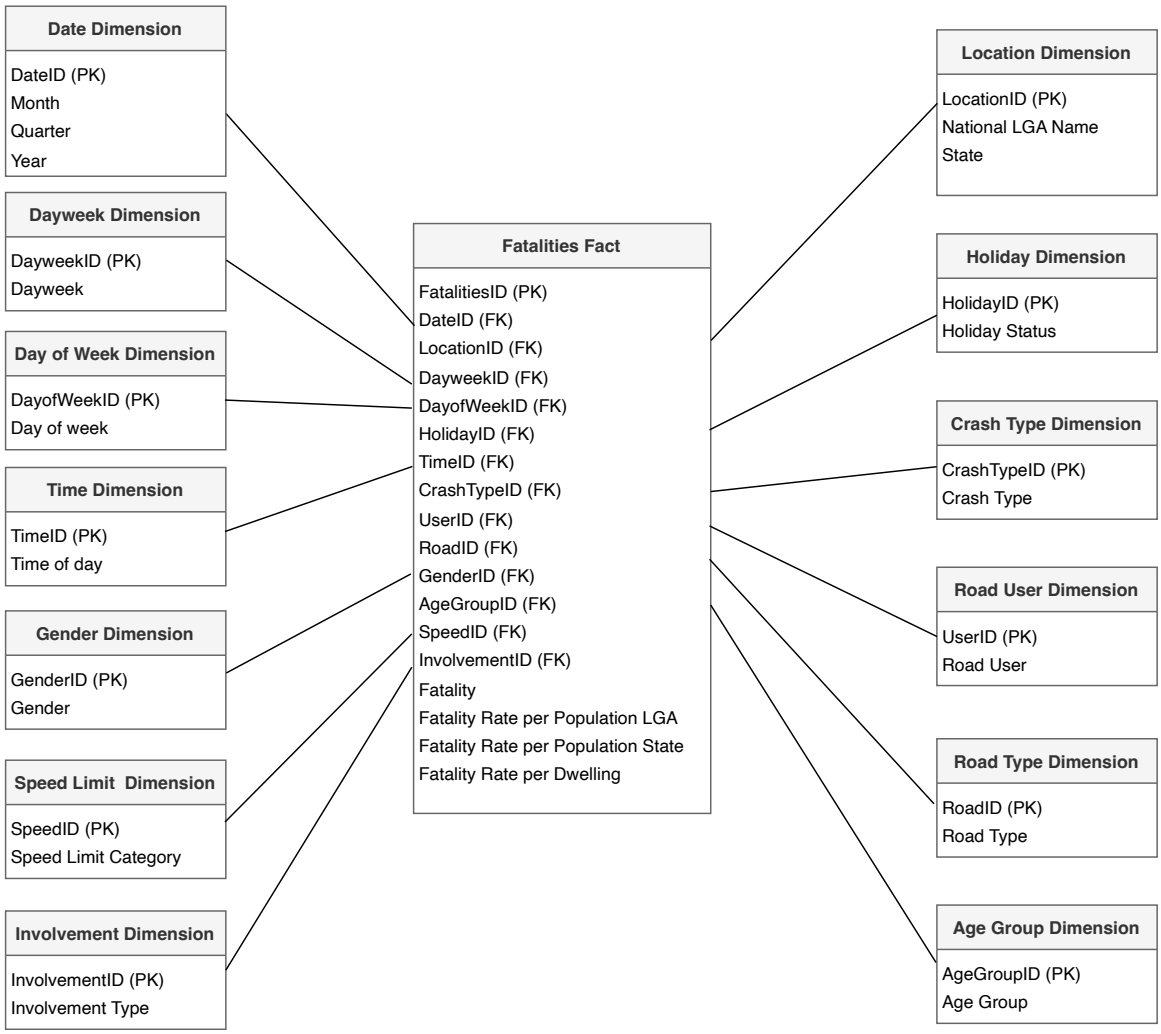


Figure 2.1. Star schema design for the fatal crash data warehouse

2.3 Concept Hierarchies for Each Dimension

We used concept hierarchies to support basic drill-down and roll-up analysis. For dimensions like Location and Date, there is a clear multi-level structure, so both schema hierarchies and concept hierarchies are included. Most other dimensions contain only one attribute, so no schema hierarchy is needed.

The following diagrams show the hierarchies for each dimension:

Location

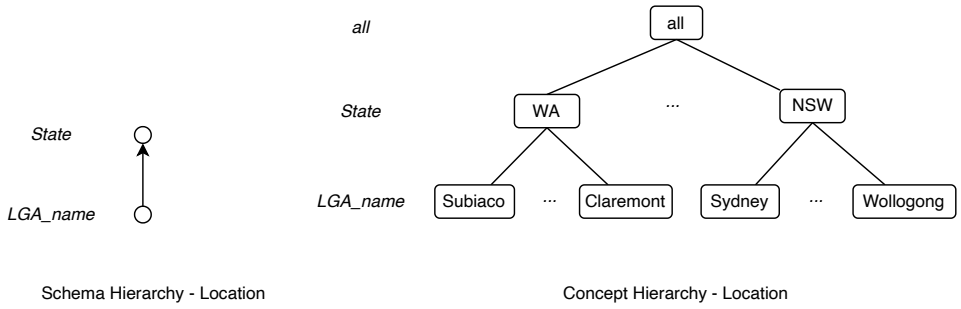


Figure 2.2. Hierarchies for location dimension

1Date

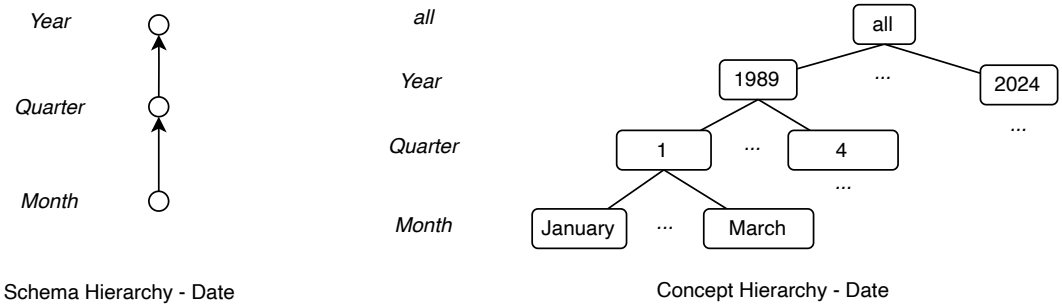
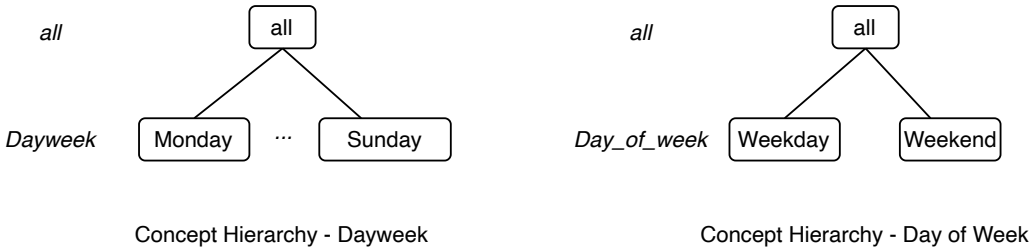
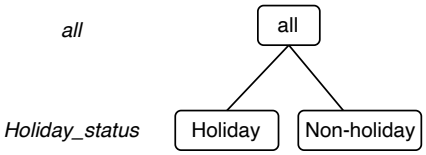


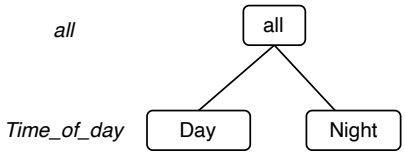
Figure 2.3. Hierarchies for date dimension

Single-attribute dimensions

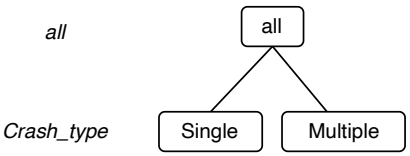




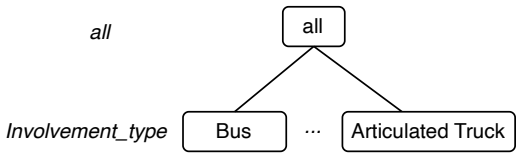
Concept Hierarchy - Holiday



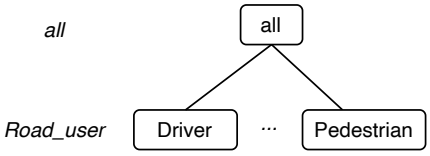
Concept Hierarchy - Time



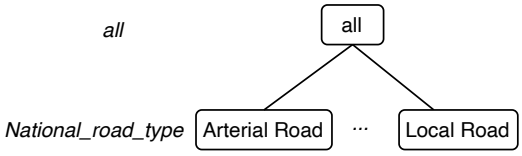
Concept Hierarchy - Crash Type



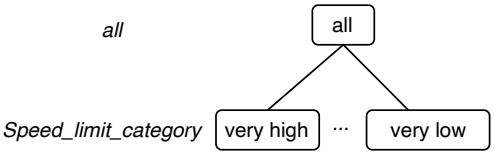
Concept Hierarchy - Involvement



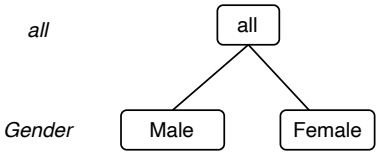
Concept Hierarchy - Road User



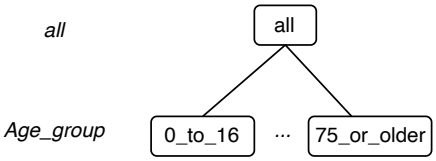
Concept Hierarchy - Road Type



Concept Hierarchy - Speed Limit



Concept Hierarchy - Gender



Concept Hierarchy - Age Group

Figure 2.4. Concept hierarchies for single-attribute dimensions

3. Data Analysis & ETL Process

The ETL process follows Kimball's 4-Step Design methodology and data warehouse schema.

- **Extract**: Data sources from ARDD datasets, population datasets, and dwelling files
- **Transform**: clean, reshape, categorise, and join data
- **Load**: populate dimension and fact tables for warehouse design

The ETL process was implemented in Python using Jupyter Notebook.

- Refer to the *CITS5504_Project1_Data_processing.ipynb* notebook for full code and output
- The raw data files located in the *DataResource* folder
- After processing, the cleaned csv files saved in the *DataWarhouse* folder and used for building and populating databse

3.1 Setup

Importing the libraries

We imported the libraries and loaded all datasets into pandas DataFrames.

```
# Import libraries for data manipulation and analysis
import pandas as pd
import numpy as np
```

Loading the datasets

- *fatalities*: individual-level road fatality records, sheet used: BITRE_Fatality

```
# Load fatality data
fatalities = pd.read_excel("./DataResource/bitre_fatalities_dec2024.xlsx",
                           sheet_name="BITRE_Fatality",
                           skiprows=4,
                           header=0
                          )
```

- *crashes*: crash-level data, sheet used: BITRE_Fatal_Crash

```
# Load crash data
crashes = pd.read_excel("./DataResource/bitre_fatal_crashes_dec2024.xlsx",
                           sheet_name="BITRE_Fatal_Crash",
                           skiprows=4,
                           header=0
                          )
```

- *population LGA*: LGA-level population data, sheet used: Table1

```
# Load population LGA data
population_LGA = pd.read_excel(("./DataResource/"
                           "Population estimates by LGA, Significant Urban Area, "
                           "Remoteness Area and electoral division, "
                           "2001 to 2024.xlsx"),
                           sheet_name="Table 1",
                           skiprows=5,
                           header=None
                          )
```

- *population state*: state-level population data, sheet used: Table1

```
# Load population state data
population_state = pd.read_excel("./DataResource/population_state.xlsx",
                           sheet_name="state",
                           header=0
                          )
```

- *dwelling*: LGA-level dwelling data, csv file

```
# Load dwelling data
dwelling = pd.read_csv("./DataResource/LGA (count of dwellings).csv",
                       skiprows=10,
                       header=0,
                       index_col=False
                      )
```

3.2 Data Exploration

We first examined the structure and content of all datasets.

```
# Display shapes of datasets
print("Fatalities shape:", fatalities.shape)
print("Crashes shape:", crashes.shape)
print("Population LGA shape:", population_LGA.shape)
print("Population State shape:", population_state.shape)
print("Dwelling shape:", dwelling.shape)
```

```
Fatalities shape: (56874, 23)
Crashes shape: (51284, 23)
Population LGA shape: (552, 26)
Population State shape: (44, 9)
Dwelling shape: (561, 2)
```

```
# Top few rows
fatalities.head()
```

	Crash ID	State	Month	Year	Dayweek	Time	Crash Type	Bus Involvement	Heavy Rigid Truck Involvement	Articulated Truck Involvement	...	Age	National Remoteness Areas
0	20241115	NSW	12	2024	Friday	04:00:00	Single	No	No	No	...	74	Inner Regional Australia
1	20241125	NSW	12	2024	Friday	06:15:00	Single	No	No	No	...	19	Inner Regional Australia
2	20246013	Tas	12	2024	Friday	09:43:00	Multiple	No	No	No	...	33	Inner Regional Australia
3	20241002	NSW	12	2024	Friday	10:35:00	Multiple	No	No	No	...	32	Outer Regional Australia
4	20242261	Vic	12	2024	Friday	11:30:00	Multiple	-9	-9	-9	...	62	Unknown

5 rows × 23 columns

```
crashes.head()
```

	Crash ID	State	Month	Year	Dayweek	Time	Crash Type	Number Fatalities	Bus \nInvolvement	Heavy Rigid Truck Involvement	...	SA4 Name 2021
0	20241115	NSW	12	2024	Friday	04:00:00	Single	1	No	No	...	Riverina
1	20241125	NSW	12	2024	Friday	06:15:00	Single	1	No	No	...	Sydney - Baulkham Hills and Hawkesbury
2	20246013	Tas	12	2024	Friday	09:43:00	Multiple	1	No	No	...	Launceston and North East
3	20241002	NSW	12	2024	Friday	10:35:00	Multiple	1	No	No	...	New England and North West
4	20242261	Vic	12	2024	Friday	11:30:00	Multiple	1	-9	-9	...	NaN

Fatalities as main dataset

The fatalities and crash datasets contain similar information. The fatalities dataset was selected as the main source for analysis because it includes more detailed individual-level attributes, such as age, gender, and road user type. These details are more suitable for our fact table. The crash dataset was used for reference only.

Join keys: we used LGA-level join (LGA name) and state-level join (state) to join datasets.

Missing LGA data

In fatalities dataset, there are total 56,874 individual records from 1989 to 2024. However, in LGA (Local Government Area) column, there are only 12,699 non-null values. Missing LGA data means the analysis and map visualisation only show areas with available LGA information.

```
# Data types, non-null counts
fatalities.info()
crashes.info()
population_state.info()
population_LGA.info()
dwelling.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56874 entries, 0 to 56873
Data columns (total 23 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Crash ID                        56874 non-null  int64
 1   State                           56874 non-null  object
 2   Month                           56874 non-null  int64
 3   Year                            56874 non-null  int64
 4   Dayweek                         56874 non-null  object
 5   Time                            56831 non-null  object
 6   Crash Type                      56874 non-null  object
 7   Bus Involvement                 56874 non-null  object
 8   Heavy Rigid Truck Involvement   56874 non-null  object
 9   Articulated Truck Involvement   56874 non-null  object
 10  Speed Limit                     56874 non-null  object
 11  Road User                       56874 non-null  object
 12  Gender                          56874 non-null  object
 13  Age                             56874 non-null  int64
 14  National Remoteness Areas       56874 non-null  object
 15  SA4 Name 2021                   12699 non-null  object
 16  National LGA Name 2021          12701 non-null  object
 17  National Road Type              56874 non-null  object
 18  Christmas Period                56874 non-null  object
 19  Easter Period                   56874 non-null  object
 20  Age Group                       56874 non-null  object
 21  Day of week                     56874 non-null  object
 22  Time of day                     56874 non-null  object
```

Population LGA

It's in wide format and will be reshaped. The data ranges from 2001 to 2024. In our analysis, population data is used to calculate the fatality rate. The fatality rate is computed by dividing the number of fatalities by the population, multiplied by 100,000.

```
population_LGA.head()
```

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21
0	NaN	NaN	2001	2002	2003	2004	2005	2006	2007	2008	...	2015	2016	2017	2018	2019	2020
1	LGA code	LGA name (a)	no.	no.	no.	no.	no.	no.	no.	no.	...	no.	no.	no.	no.	no.	no.
2	10050	Albury	45265	45816	46180	46505	47004	47566	48140	48518	...	51486	52171	53056	53922	54657	55466
3	10180	Armidale	27906	27774	27610	27410	27350	27377	27468	27788	...	29160	29310	29519	29631	29701	29600
4	10250	Ballina	37856	38417	38870	39120	39305	39537	39824	40020	...	42336	42993	43652	44385	44997	45663

Population state

Since there are LGA information missing in the fatalities data. When it merged with the population LGA, only 22% records will have the corresponding population information. Therefore, we added an additional population data in state level, which contains population data for each state from 1981 to 2024.

```
population_state.head()
```

	Unnamed: 0	NSW	Vic	Qld	SA	WA	Tas	NT	ACT
0	1981	5249455	3957333	2367477	1321235	1311284	427925	125186	228782
1	1982	5315846	4002731	2442912	1334090	1348096	430308	131517	233778
2	1983	5363744	4045185	2493373	1349553	1375244	433909	137942	240055
3	1984	5416536	4086549	2535976	1362611	1397817	438866	143934	246259
4	1985	5478254	4129796	2583368	1373324	1427370	443548	150596	253446

Dwelling

```
dwelling.head()
```

	LGA (EN)	Unnamed: 1
0	Albury	25430
1	Armidale Regional	12955
2	Ballina	20889
3	Balranald	1091
4	Bathurst Regional	18458

3.3 Data Cleaning and Preparation

Data processing

1. population data

- reshape from wide to long
- convert values to numeric types

```
# Clean population_LGA data
population_LGA.iloc[0, :2] = population_LGA.iloc[1, :2]
population_LGA.columns = population_LGA.iloc[0]
population_LGA = population_LGA.drop(index=[0, 1]).reset_index(drop=True)

# Reshape population_LGA data from wide to long format
population_LGA = population_LGA.melt(
    id_vars=["LGA code", "LGA name (a)"],
    var_name="Year",
    value_name="Population_LGA"
)

# drop rows where LGA name or Year is missing
population_LGA = population_LGA.dropna(subset=["LGA name (a)", "Year"])
```

```
# Transformation
population_state = population_state.rename(columns={population_state.columns[0]:"Year"})
population_state_long = population_state.melt(
  id_vars="Year",
  var_name= "State",
  value_name="Population_State"
)
population_state_long["Year"] = pd.to_numeric(population_state_long["Year"], errors="coerce")
population_state_long["Population_State"] = pd.to_numeric(population_state_long["Population_State"], errors="coerce")
```

Data merging

- merge dwellings with population
- merge population and dwelling into fatalities based on LGA name and year

```
# Merge dwellings with population_LGA LGA
dwelling.columns = ["LGA name (a)", "Dwelling"]
population_LGA = population_LGA.merge(dwelling, on="LGA name (a)", how="left")

# Convert data types to numeric types
population_LGA["Population_LGA"] = pd.to_numeric(population_LGA["Population_LGA"],errors="coerce")
population_LGA["Dwelling"] = pd.to_numeric(population_LGA["Dwelling"], errors="coerce")
population_LGA = population_LGA.drop(columns=["LGA code"])

population_LGA.info()
```

```
# Merge population LGA and dwelling info into fatalities
fatalities = fatalities.merge(
    population_LGA, left_on=["National LGA Name 2021", "Year"],
    right_on=["LGA name (a)", "Year"],
    how="left"
)

# Merge population LGA and dwelling info into fatalities
fatalities = fatalities.merge(
    population_state, left_on=["State", "Year"],
    right_on=["State", "Year"],
    how="left"
)
```

Missing values

Although LGA is important for geographic analysis, it is just an attribute of the fatality records. Each row represents an individual fatality, and keeping all rows to preserve historical fatality records is more important than dropping them for missing LGA data.

A value of '-9' in original dataset is used for a missing/unknown value. Non-standard values are converted as NaN. These rows are kept in the dataset to preserve historical records.

```
# Replace knwn invalid or missing codes with NaN
fatalities = fatalities.replace(
    [-9, "-9", "unknown", "Unknown", "Other/-9", "Undetermined"],
    np.nan
)
```

3.4 Dimension Tables

We created several dimension tables for later use.

Location Dimension

- Extracts unique LGA and State combinations.
- Fill missing values with "Unknown" category in LGA to make sure each row in fact table won't have null values, which is important in data visualisation part
- SA4 Name 2021 was excluded because it's not required for our queries, and LGA is more suitable for our analysis.
- LocationID as primary key.

```
# Create location dimension

location_df = fatalities[["National LGA Name 2021", "State"]].copy()
location_df["National LGA Name 2021"] = location_df["National LGA Name 2021"].fillna("Unknown")  # Create "Unknown" category
location_df = location_df.drop_duplicates()
location_df.insert(0, "LocationID", range(1, 1 + len(location_df)))
location_df.head()
```

	LocationID	National LGA Name 2021	State
0	1	Wagga Wagga	NSW
1	2	Hawkesbury	NSW
2	3	Northern Midlands	Tas
3	4	Armidale Regional	NSW
4	5	Unknown	Vic

Date Dimension

- Extracts unique Year, Month, and Dayweek combinations.
- Adds Quarter column.
- DateID as primary key.

```
# Create date dimension
date_df = fatalities[["Month", "Year"]].drop_duplicates().dropna()

# Add Quarter column
date_df["Quarter"] = ((date_df["Month"] - 1) // 3) + 1

date_df.insert(0, "DateID", range(1, 1 + len(date_df)))
date_df.head()
```

	DateID	Month	Year	Quarter
0	1	12	2024	4
113	2	11	2024	4
244	3	10	2024	4
359	4	9	2024	3
454	5	8	2024	3

Dayweek Dimension

```
# Create dayweek_df
dayweek_df = fatalities[["Dayweek"]].drop_duplicates().dropna()
dayweek_df.insert(0, "DayweekID", range(1, 1 + len(dayweek_df)))
dayweek_df
```

	DayweekID	Dayweek
0	1	Friday
14	2	Monday
31	3	Saturday
56	4	Sunday
83	5	Thursday
92	6	Tuesday
105	7	Wednesday

Date of Week Dimension

```
# Create day_of_week_df and

day_of_week_df = fatalities[["Day of week"]].drop_duplicates().dropna()
day_of_week_df.insert(0, "DayofWeekID", range(1, 1 + len(day_of_week_df)))
day_of_week_df
```

	DayofWeekID	Day of week
0	1	Weekday
8	2	Weekend

We noticed that some “Dayweek” values were misclassified. Since the “Day of week” feature is not useful for further analysis, we decided not to correct them.

Holiday Dimension

- Creates Holiday Status in fatalities based on Christmas Period and Easter Period
- Extracts unique Holiday Status values

```
# Create holiday status column in fatalities
fatalities["Holiday Status"] = np.where(
    (fatalities["Christmas Period"] == "Yes") |
    (fatalities["Easter Period"] == "Yes"), "Holiday", "Non-holiday")
```

```
# Create holiday dimension
holiday_df = fatalities[["Holiday Status"]].drop_duplicates().dropna()
holiday_df.insert(0, "HolidayID", range(1, 1 + len(holiday_df)))
holiday_df
```

	HolidayID	Holiday Status
0	1	Holiday
1	2	Non-holiday

Time Dimension

```
# Create time dimension
time_df = fatalities[["Time of day"]].drop_duplicates().dropna()
time_df.insert(0, "TimeID", range(1, 1 + len(time_df)))
time_df
```

	TimeID	Time of day
0	1	Night
1	2	Day

Crash Type Dimension

```
# Create crash type dimension
crashtype_df = fatalities[["Crash Type"]].drop_duplicates().dropna()
crashtype_df.insert(0, "CrashTypeID", range(1, 1 + len(crashtype_df)))
crashtype_df
```

	CrashTypeID	Crash Type
0	1	Single
2	2	Multiple

Involvement Dimension

- Creates Involvement Type in fatalities, classifying combinations like Bus, Heavy Rigid Truck, Articulated Truck, and their combinations

- Includes all 6 types: Bus, Heavy Rigid Truck, Articulated Truck, Bus/Heavy Rigid, Bus/Articulated Truck, and Heavy Rigid/Articulated Truck
- InvolvementID as primary key

We used ChatGPT to understand the logic and usage of the np.select( ) function, which helped us categorise both the involvement types and, later, the speed limit attribute as well [5].

```
# Create involvement columns in fatalities and fill missing values
fatalities[["Bus Involvement", "Heavy Rigid Truck Involvement", "Articulated Truck Involvement"]] = (
    fatalities[["Bus Involvement", "Heavy Rigid Truck Involvement", "Articulated Truck Involvement"]]
    .fillna("No")
)

# Create Involvement Type column based on involvement values
fatalities["Involvement Type"] = np.select(
    [
        (fatalities["Bus Involvement"] == "Yes") & (fatalities["Heavy Rigid Truck Involvement"] == "No")
            & (fatalities["Articulated Truck Involvement"] == "No"),
        (fatalities["Bus Involvement"] == "No") & (fatalities["Heavy Rigid Truck Involvement"] == "Yes")
            & (fatalities["Articulated Truck Involvement"] == "No"),
        (fatalities["Bus Involvement"] == "No") & (fatalities["Heavy Rigid Truck Involvement"] == "No")
            & (fatalities["Articulated Truck Involvement"] == "Yes"),
        (fatalities["Bus Involvement"] == "Yes") & (fatalities["Heavy Rigid Truck Involvement"] == "Yes")
            & (fatalities["Articulated Truck Involvement"] == "No"),  # Add Bus/Heavy Rigid
        (fatalities["Bus Involvement"] == "Yes") & (fatalities["Heavy Rigid Truck Involvement"] == "No")
            & (fatalities["Articulated Truck Involvement"] == "Yes"),
        (fatalities["Bus Involvement"] == "No") & (fatalities["Heavy Rigid Truck Involvement"] == "Yes")
            & (fatalities["Articulated Truck Involvement"] == "Yes"),
    ],
    ["Bus", "Heavy Rigid Truck", "Articulated Truck", "Bus/Heavy Rigid", "Bus/Articulated Truck", "Heavy Rigid/Articulated Truck"],
    default=None
)
```

```
# Create involvement dimension
involvement_df = fatalities[["Involvement Type"]].drop_duplicates().dropna()
involvement_df.insert(0, "InvolvementID", range(1, 1 + len(involvement_df)))
involvement_df
```

	InvolvementID	Involvement Type
13	1	Articulated Truck
32	2	Bus/Articulated Truck
41	3	Heavy Rigid Truck
102	4	Heavy Rigid/Articulated Truck
103	5	Bus
547	6	Bus/Heavy Rigid

User Dimension

```
# Create user dimension
user_df = fatalities[["Road User"]].drop_duplicates().dropna()
user_df.insert(0, "UserID", range(1, 1 + len(user_df)))
user_df
```

	UserID	Road User
0	1	Driver
4	2	Passenger
6	3	Motorcycle rider
12	4	Pedestrian
50	5	Pedal cyclist
488	6	Motorcycle pillion passenger

Road Dimension

```
# Create road dimension
road_df = fatalities[["National Road Type"]].drop_duplicates().dropna()
road_df.insert(0, "RoadID", range(1, 1 + len(road_df)))
road_df
```

	RoadID	National Road Type
0	1	Arterial Road
1	2	Local Road
3	3	National or State Highway
6	4	Sub-arterial Road
17	5	Collector Road
154	6	Pedestrian Thoroughfare
283	7	Access road
970	8	Busway

Speed Limit Dimension

- Extracts unique Speed Limit Category values from fatalities.
- Define speed limit categorises: Very Low (0-30), Low (31-60), Medium (61-90), High (91-110), and Very High (>110)
- SpeedID as primary key

```
# Create speed limit categories in fatalities

# Very Low : 0–30
# Low: 31–60
# Medium: 61–90
# High: 91–110
# Very high: >110

fatalities["Speed Limit Category"] = np.select(
    [(fatalities["Speed Limit"] == 5) | (fatalities["Speed Limit"] == 10) |
        (fatalities["Speed Limit"] == 15)| (fatalities["Speed Limit"] == 20) |
        (fatalities["Speed Limit"] == 25) | (fatalities["Speed Limit"] == 30),  # Very low
    (fatalities["Speed Limit"] == '<40') | (fatalities["Speed Limit"] == 40) | (fatalities["Speed Limit"] == 50)
        | (fatalities["Speed Limit"] == 60),  # Low
    (fatalities["Speed Limit"] == 70) | (fatalities["Speed Limit"] == 75) | (fatalities["Speed Limit"] == 80)
        | (fatalities["Speed Limit"] == 90),  # Medium
    (fatalities["Speed Limit"] == 100) | (fatalities["Speed Limit"] == 110),  # High
    (fatalities["Speed Limit"] == 130)  # Very High
    ], ["Very Low", "Low", "Medium", "High", "Very High"], default=None)
```

```
# Create speed limit dimension
speed_limit_df = fatalities[["Speed Limit Category"]].drop_duplicates().dropna()
speed_limit_df.insert(0, "SpeedID", range(1, 1 + len(speed_limit_df)))
speed_limit_df
```

	SpeedID	Speed Limit Category
0	1	High
1	2	Medium
2	3	Low
63	4	Very Low
554	5	Very High

Gender Dimension

```
# Create gender dimension
gender_df = fatalities[["Gender"]].drop_duplicates().dropna()
gender_df.insert(0, "GenderID", range(1, 1 + len(gender_df)))
gender_df
```

	GenderID	Gender
0	1	Male
1	2	Female

Age Group Dimension

```
# Create age group dimension
age_group_df = fatalities[["Age Group"]].drop_duplicates().dropna()
age_group_df.insert(0, "AgeGroupID", range(1, 1 + len(age_group_df)))
age_group_df
```

	AgeGroupID	Age Group
0	1	65_to_74
1	2	17_to_25
2	3	26_to_39
4	4	40_to_64
18	5	75_or_older
20	6	0_to_16

3.5 Fact Table

We built the fact table by merging the fatalities data with all dimension tables.

- Fatality: always 1, since each row represents one death
- Fatality Rate LGA per 100,000 = (fatalities / LGA population) x 100,000
- Fatality Rate State per 100,000 = (fatalities / State population) x 100,000
- Fatality Rate per Dwelling = (fatalities / dwellings) x 100,000

```
# Create fact table
fact_df = fatalities.copy()
```

Merge with dimension tables

```
# Merge with date dimension
fact_df = fact_df.merge(date_df, on = ["Month", "Year"], how="left")

# Merge with location dimension
fact_df["National LGA Name 2021"] = fact_df["National LGA Name 2021"].fillna("Unknown")
fact_df = fact_df.merge(location_df, on = ["National LGA Name 2021", "State"], how="left")

dfs = [dayweek_df, day_of_week_df, holiday_df, time_df, crashtype_df, user_df, road_df, gender_df, age_group_df,
       speed_limit_df, involvement_df]
for df in dfs:
    fact_df = fact_df.merge(df, on = df.columns[1], how="left")

# Select columns
fact_df = fact_df[["DateID", "LocationID",  "DayweekID",  "DayofWeekID", "HolidayID", "TimeID", "CrashTypeID", "UserID", "RoadID",
                   "GenderID", "AgeGroupID", "SpeedID", "InvolvementID", "Population_LGA", "Population_State", "Dwelling"]]
```

Create measures

```
# Create measures
fact_df["Fatality"] = 1
fact_df["Fatality Rate per Population LGA"] = (fact_df["Fatality"] /
                                               fact_df["Population_LGA"]).where(fact_df["Population_LGA"].notna())*100000
fact_df["Fatality Rate per Population State"] = (fact_df["Fatality"] /
                                               fact_df["Population_State"]).where(fact_df["Population_State"].notna())*100000
fact_df["Fatality Rate per Dwelling"] = (fact_df["Fatality"] /
                                              fact_df["Dwelling"]).where(fact_df["Dwelling"].notna())*100000


fact_df.insert(0, "FatalitiesID", range(1, 1+len(fact_df)))
```

3.6 Export to CSV files

```
# Export dimension and fact tables to CSV
csv_names = [
    "DimDayweek.csv", "DimDayofWeek.csv", "DimHoliday.csv", "DimTime.csv", "DimCrashType.csv",
    "DimUser.csv", "DimRoad.csv", "DimGender.csv", "DimAgeGroup.csv", "DimSpeedLimit.csv",
    "DimInvolvement.csv","DimDate.csv", "DimLocation.csv", "FactFatalities.csv"
]

dfs.extend([date_df, location_df, fact_df])
for i in range(len(dfs)):
    dfs[i].to_csv(f"./DataWarehouse/{csv_names[i]}", index=False)
```

4. Database Schema and SQL Implementation

Tool used: Postgres, pdAdmin4

4.1 Creating the Data Warehouse and Tables

We started by creating a new database to store the data. Each dimension table includes a primary key and one or more descriptive attributes.

```sql
CREATE TABLE DimAgeGroup (
 "AgeGroupID" INT PRIMARY KEY,
 "Age Group" VARCHAR(50) NULL
);

CREATE TABLE DimCrashType (
 "CrashTypeID" INT PRIMARY KEY,
 "Crash Type" VARCHAR(50) NULL
);

CREATE TABLE DimDate (
 "DateID" INT PRIMARY KEY,
 "Month" INT NOT NULL,
  "Year" INT NOT NULL,
 "Quarter" INT NOT NULL
);

CREATE TABLE DimDayweek (
 "DayweekID" INT PRIMARY KEY,
 "Dayweek" VARCHAR(50) NULL
);

CREATE TABLE DimDayofWeek (
 "DayofWeekID" INT PRIMARY KEY,
 "Day of week" VARCHAR(50) NULL
);

CREATE TABLE DimGender (
 "GenderID" INT PRIMARY KEY,
 "Gender" VARCHAR(50) NULL
);

CREATE TABLE DimHoliday (
 "HolidayID" INT PRIMARY KEY,
 "Holiday Status" VARCHAR(50) NULL
);
```

```sql
CREATE TABLE DimInvolvement (
 "InvolvementID" INT PRIMARY KEY,
 "Involvement Type" VARCHAR(50) NULL
);

CREATE TABLE DimLocation (
 "LocationID" INT PRIMARY KEY,
 "National LGA Name 2021" VARCHAR(50) NULL,
 "State" VARCHAR(50) NULL
);

CREATE TABLE DimRoad (
 "RoadID" INT PRIMARY KEY,
 "National Road Type" VARCHAR(50) NULL
);

CREATE TABLE DimSpeedLimit (
 "SpeedID" INT PRIMARY KEY,
 "Speed Limit Category" VARCHAR(50) NULL
);

CREATE TABLE DimTime (
 "TimeID" INT PRIMARY KEY,
 "Time of day" VARCHAR(50) NULL
);

CREATE TABLE DimUser (
 "UserID" INT PRIMARY KEY,
 "Road User" VARCHAR(50) NULL
);
```

Figure 4.1.1. SQL command for creating Dimension tables

```sql
CREATE TABLE FactFatalities (
 "FatalitiesID" INT PRIMARY KEY,
 "DateID" INT  NULL,
 "LocationID" INT  NULL,
 "DayweekID" INT  NULL,
 "DayofWeekID" INT  NULL,
 "HolidayID" INT  NULL,
 "TimeID" INT  NULL,
 "CrashTypeID" INT NULL,
 "UserID" INT  NULL,
 "RoadID" INT  NULL,
 "GenderID" INT  NULL,
 "AgeGroupID" INT  NULL,
 "SpeedID" INT  NULL,
 "InvolvementID" INT NULL,
 "Fatality" INT  NULL,
 "Fatality Rate per Population LGA" FLOAT NULL,
 "Fatality Rate per Population State" FLOAT NULL,
 "Fatality Rate per Dwelling" FLOAT NULL,
 FOREIGN KEY ("DateID") REFERENCES DimDate("DateID"),
 FOREIGN KEY ("DayweekID") REFERENCES DimDayweek("DayweekID"),
 FOREIGN KEY ("DayofWeekID") REFERENCES DimDayofWeek("DayofWeekID"),
 FOREIGN KEY ("HolidayID") REFERENCES DimHoliday("HolidayID"),
 FOREIGN KEY ("TimeID") REFERENCES DimTime("TimeID"),
 FOREIGN KEY ("CrashTypeID") REFERENCES DimCrashType("CrashTypeID"),
 FOREIGN KEY ("UserID") REFERENCES DimUser("UserID"),
 FOREIGN KEY ("RoadID") REFERENCES DimRoad("RoadID"),
 FOREIGN KEY ("GenderID") REFERENCES DimGender("GenderID"),
 FOREIGN KEY ("AgeGroupID") REFERENCES DimAgeGroup("AgeGroupID"),
 FOREIGN KEY ("LocationID") REFERENCES DimLocation("LocationID"),
 FOREIGN KEY ("SpeedID") REFERENCES DimSpeedLimit("SpeedID"),
 FOREIGN KEY ("InvolvementID") REFERENCES DimInvolvement("InvolvementID")
);
```

Figure 4.1.2. SQL command for creating the Fact table

4.2 Populate data

We imported data from cleaned CSV files using bulk insert methods. The dimension tables were loaded first, followed by the fact table, due to its references to the dimension tables.

```sql
COPY DimAgeGroup
FROM '/tmp/DataWarehouse/DimAgeGroup.csv'
WITH CSV HEADER;

COPY DimCrashType
FROM '/tmp/DataWarehouse/DimCrashType.csv'
WITH CSV HEADER;

COPY DimDate
FROM '/tmp/DataWarehouse/DimDate.csv'
WITH CSV HEADER;

COPY DimDayweek
FROM '/tmp/DataWarehouse/DimDayweek.csv'
WITH CSV HEADER;

COPY DimDayofWeek
FROM '/tmp/DataWarehouse/DimDayofWeek.csv'
WITH CSV HEADER;

COPY DimGender
FROM '/tmp/DataWarehouse/DimGender.csv'
WITH CSV HEADER;

COPY DimHoliday
FROM '/tmp/DataWarehouse/DimHoliday.csv'
WITH CSV HEADER;
```

```sql
COPY DimInvolvement
FROM '/tmp/DataWarehouse/DimInvolvement.csv'
WITH CSV HEADER;

COPY DimLocation
FROM '/tmp/DataWarehouse/DimLocation.csv'
WITH CSV HEADER;

COPY DimRoad
FROM '/tmp/DataWarehouse/DimRoad.csv'
WITH CSV HEADER;

COPY DimSpeedLimit
FROM '/tmp/DataWarehouse/DimSpeedLimit.csv'
WITH CSV HEADER;

COPY DimTime
FROM '/tmp/DataWarehouse/DimTime.csv'
WITH CSV HEADER;

COPY DimUser
FROM '/tmp/DataWarehouse/DimUser.csv'
WITH CSV HEADER;

COPY FactFatalities
FROM '/tmp/DataWarehouse/FactFatalities.csv'
WITH CSV HEADER;
```

Figure 4.2.1. SQL command for inserting data

After all tables were in place, an ER diagram was generated in pgAdmin 4 to visualise the complete star schema. The diagram shows the central fact table and its direct connections to each of the dimension tables.

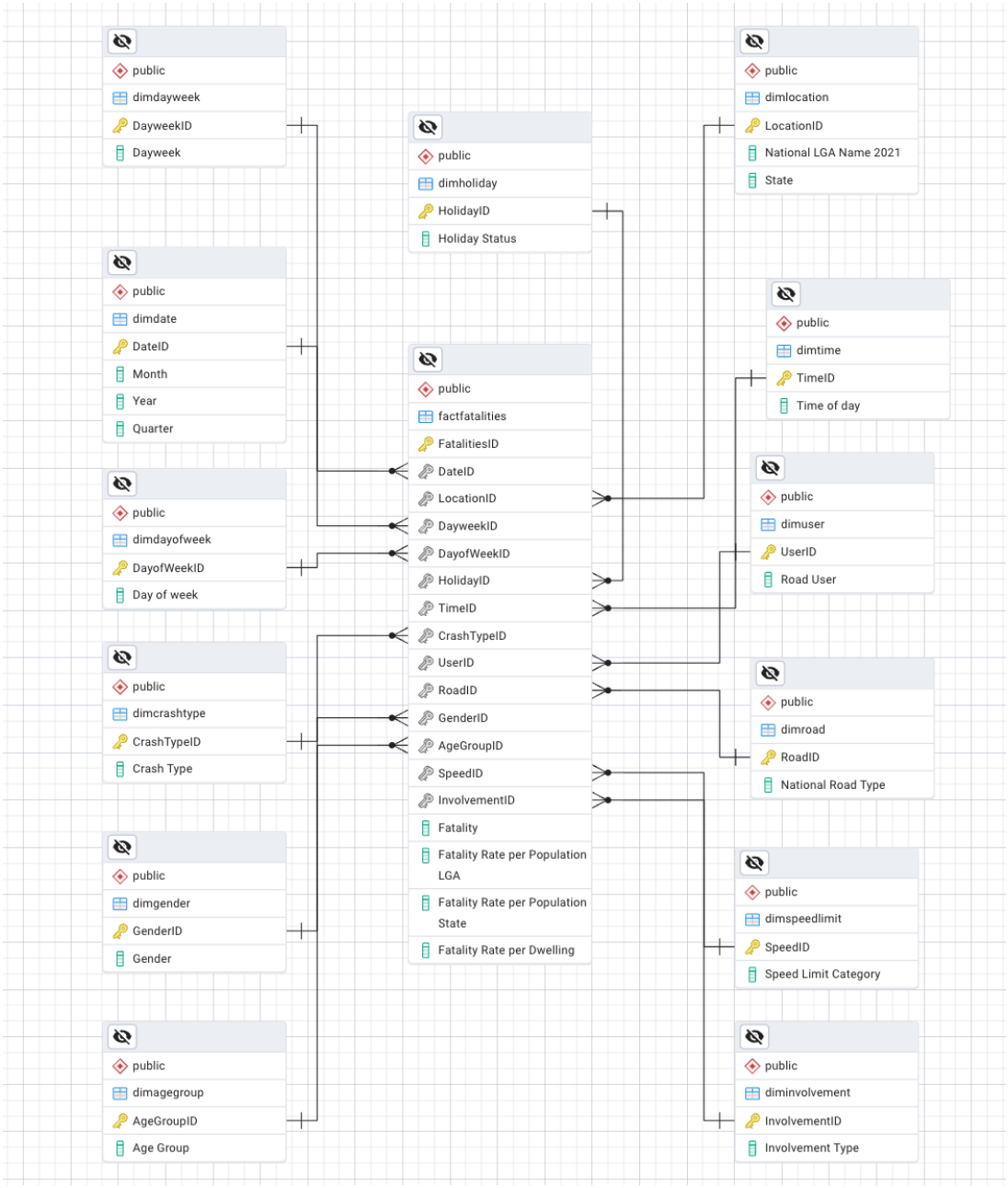


Figure 4.2.2. ER diagram of the star schema generated in pgAdmin4

5. Business Queries and StarNet

StarNet model

The following StarNet model shows the query capabilities of data warehouse. It was used to support multi-dimensional analysis of road fatalities. Data can be explored across several dimensions, such as locations, speed limit, and type of involvement.

- Using roll-up and drill-down operations, it can move between summary and detailed views, such as from state to local area.
- With slice and dice, the data can be filtered by values like gender, road user type, and age group. It can be used on analysing specific groups.

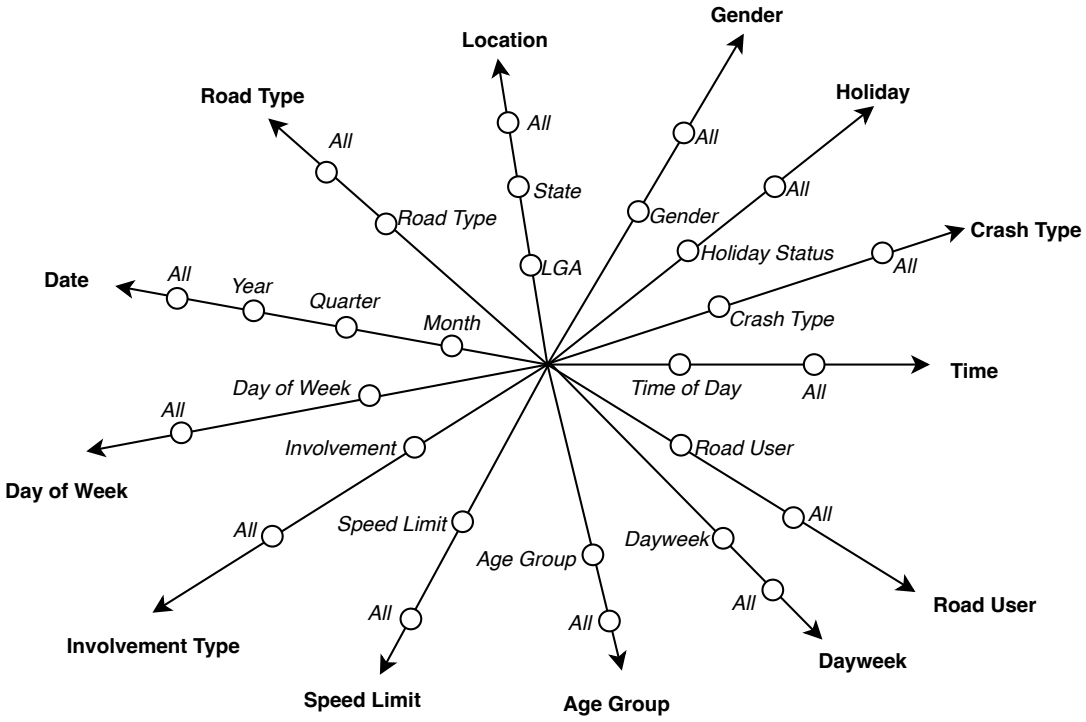


Figure 5.1. StarNet for business query

In order to identify patterns related to high-risk factors over the past five years (2020-2024), this report applied OLAP operations, such as slice, drill down and drill up, to address the following business queries.

5.1 From 2020 to 2024, how have fatality rates (per 100,000 population) changed over time across Australian states, and which states had the highest average annual fatality rates per 100,000 population?

1.StarNet

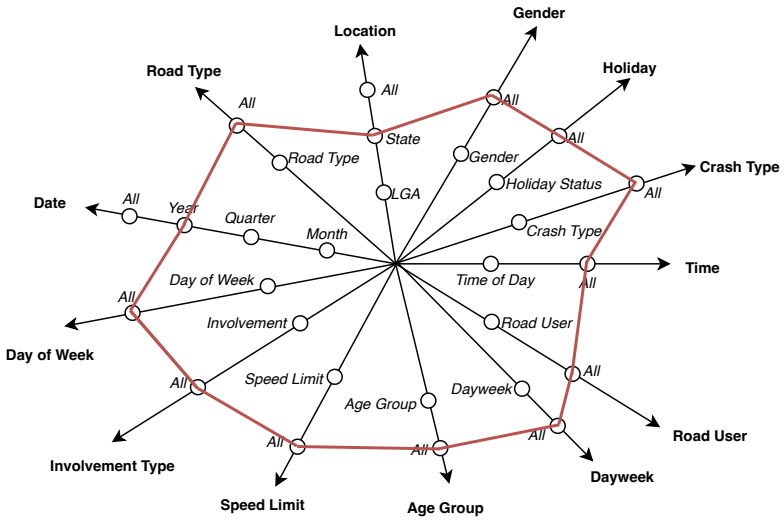


Figure 5.1.1 StarNet for business query 1

2. Visualisation

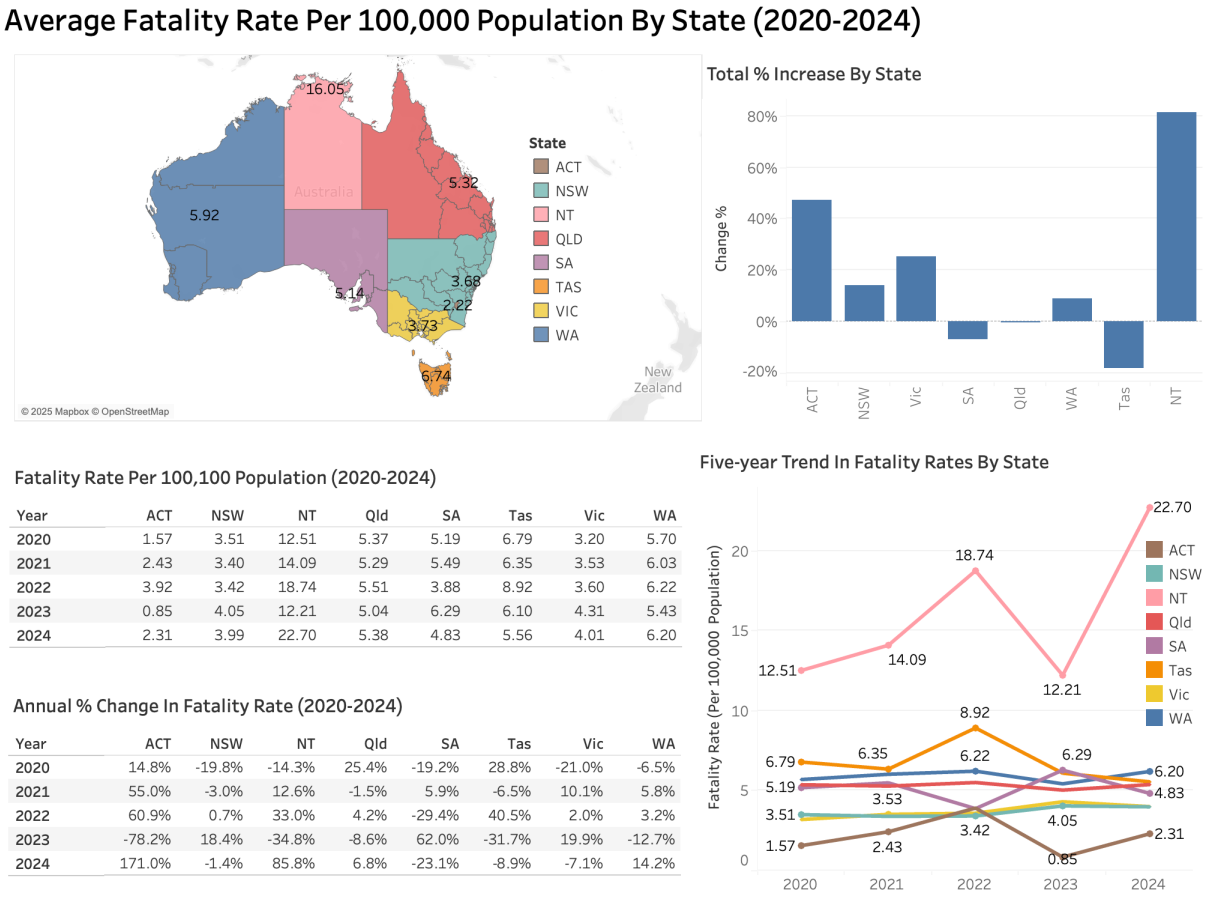


Figure 5.1.2 Dashboard for business query 1

Over the past five years, Northern Territory exhibited the highest average fatality rate per 100,000 population at 1,605%.

3. Cube Design

```
SELECT
  dl."State",
  ROUND(SUM(ft."Fatality Rate per Population State")::numeric / 5.0, 4) AS avg_fatality_rate_per_year
FROM FactFatalities ft
JOIN DimLocation dl
  ON ft."LocationID" = dl."LocationID"
JOIN DimDate dd
  ON ft."DateID" = dd."DateID"
WHERE dd."Year" BETWEEN 2020 AND 2024
GROUP BY CUBE(dl."State")
HAVING dl."State" is not null
ORDER BY avg_fatality_rate_per_year DESC
LIMIT 1;
```

	State character varying (50)	avg_fatality_rate_per_year numeric
1	NT	16.0512

Figure 5.1.3 SQL Result for business query 1

5.2 From 2020 to 2024, which states and LGAs had the highest number of male fatalities during daytime hours?

1. StarNet

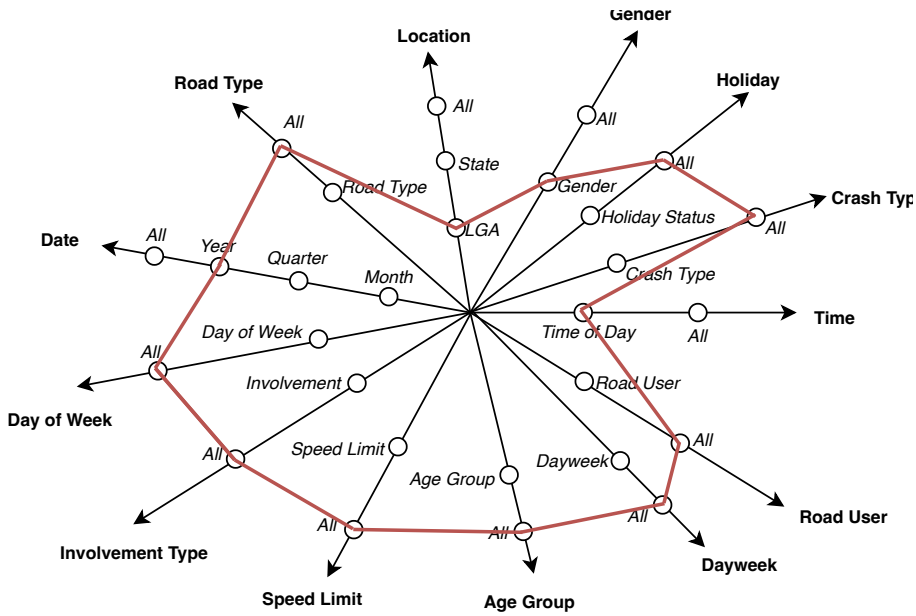


Figure 5.2.1 StarNet for business query 2

2. Visualisation

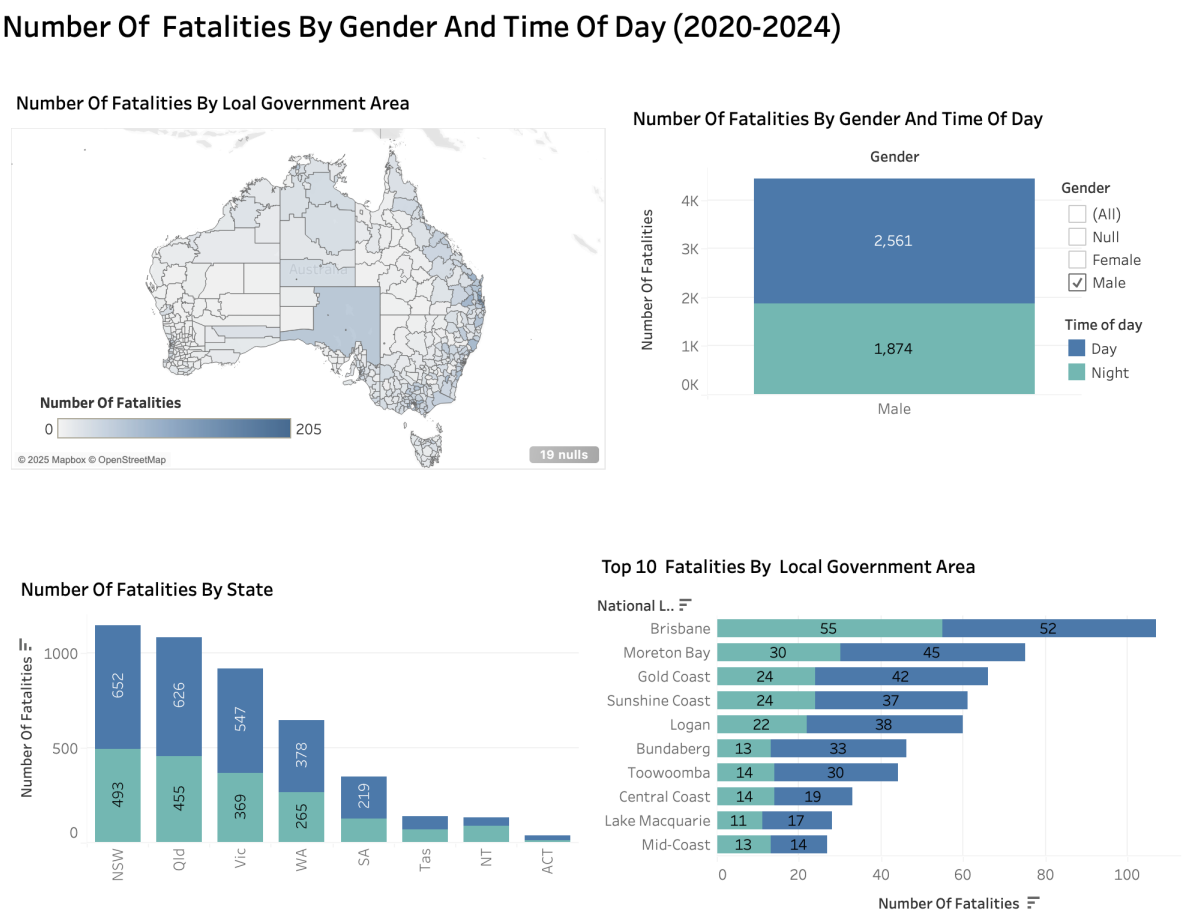


Figure 5.2.2 Dashboard for business query 2

During the daytime hours, New South Wales recorded the highest male fatalities with a total number of 652. However, among local government areas, Brisbane reported the highest count at 52.

3. Cube Design

State:

```
SELECT
  dl."State",
  dg."Gender",
  dt."Time of day",
  SUM(ft."Fatality") AS male_daytime_fatalities
FROM FactFatalities ft
JOIN DimLocation dl
  ON ft."LocationID" = dl."LocationID"
JOIN DimGender dg
  ON ft."GenderID" = dg."GenderID"
JOIN DimTime dt
  ON ft."TimeID" = dt."TimeID"
JOIN DimDate dd
  ON ft."DateID" = dd."DateID"
WHERE dg."Gender" = 'Male'
  AND dt."Time of day" = 'Day'
  AND dd."Year" BETWEEN 2020 AND 2024
GROUP BY CUBE(dl."State",dg."Gender",dt."Time of day")
HAVING dl."State" IS NOT NULL
ORDER BY male_daytime_fatalities DESC
LIMIT 1;
```

	State character varying (50)	Gender character varying (50)	Time of day character varying (50)	male_daytime_fatalities bigint
1	NSW	Male	Day	652

Figure 5.2.3 SQL for business query 2

LGA:

```
SELECT
  dl."National LGA Name 2021" AS lga,
  dg."Gender",
  dt."Time of day",
  SUM(ft."Fatality") AS male_daytime_fatalities
FROM FactFatalities ft
JOIN DimLocation dl
  ON ft."LocationID" = dl."LocationID"
JOIN DimGender dg
  ON ft."GenderID" = dg."GenderID"
JOIN DimTime dt
  ON ft."TimeID" = dt."TimeID"
JOIN DimDate dd
  ON ft."DateID" = dd."DateID"
WHERE dg."Gender" = 'Male'
  AND dt."Time of day" = 'Day'
  AND dd."Year" BETWEEN 2020 AND 2024
  AND dl."National LGA Name 2021" IS NOT NULL
  AND dl."National LGA Name 2021" != 'Unknown'
GROUP BY CUBE(dl."National LGA Name 2021", dg."Gender", dt."Time of day")
HAVING dl."National LGA Name 2021" IS NOT NULL
ORDER BY male_daytime_fatalities DESC
LIMIT 1;
```

	lga character varying (50)	Gender character varying (50)	Time of day character varying (50)	male_daytime_fatalities bigint
1	Brisbane	Male	Day	52

Figure 5.2.3 SQL Result for business query 2

5.3 From 2020 to 2024, what was the highest fatality rate per 100,000 dwelling in the unincorporated areas of South Australia (unincorporated SA) , broken down by national road type, day of the week and time of day?

1.StarNet

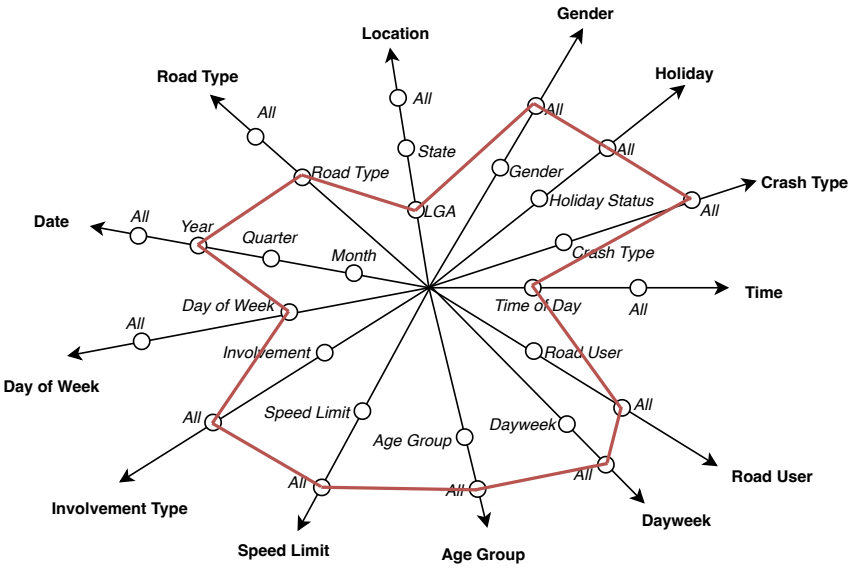


Figure 5.3.1 StarNet for business query 3

2.Visualisation

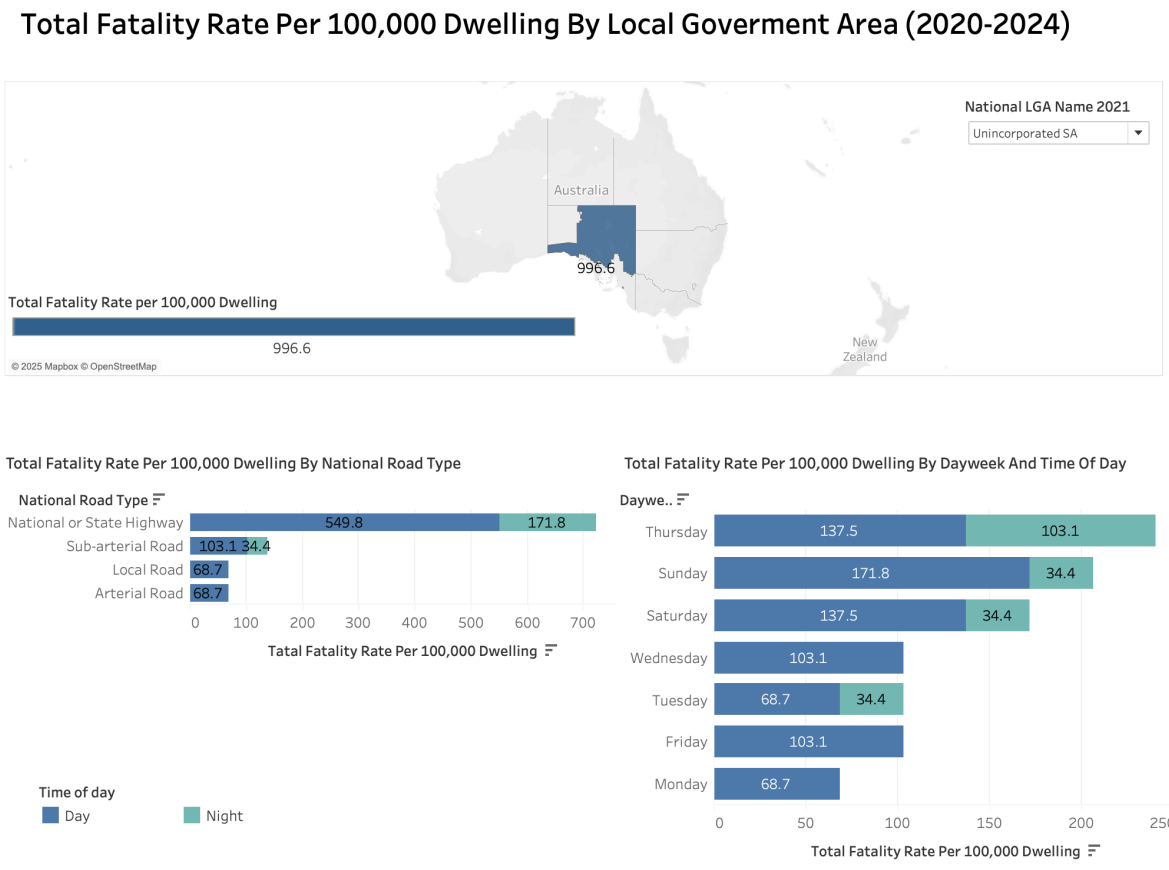


Figure 5.3.2 Dashboard for business query 3

Unincorporated South Australia recorded a fatality rate of 99,700% per 100,000 dwellings over the past five years. In terms of national road type, most crashes occurred on national or state highways, with the highest rate observed daytime (54,980% per 100,000 dwellings). Additionally, the highest daytime crash rate was reported on Sunday, reaching 17,180% per 100,000 dwellings.

3. Cube Design

National Road type

```
-- National road type
SELECT
  dl."National LGA Name 2021",
  dr."National Road Type",
  dt."Time of day",
  ROUND(SUM(ft."Fatality Rate per Dwelling")::numeric, 4) AS avg_fatality_rate_per_dwelling
FROM FactFatalities ft
JOIN DimLocation dl
  ON ft."LocationID" = dl."LocationID"
JOIN DimRoad dr
  ON ft."RoadID" = dr."RoadID"
JOIN DimTime dt
  ON ft."TimeID" = dt."TimeID"
JOIN DimDate d
  ON ft."DateID" = d."DateID"
WHERE d."Year" BETWEEN 2020 AND 2024
GROUP BY CUBE(dl."National LGA Name 2021",dr."National Road Type", dt."Time of day")
HAVING dl."National LGA Name 2021" = 'Unincorporated SA'
    AND dt."Time of day" = 'Day'
    AND dr."National Road Type" IS NOT NULL
ORDER BY avg_fatality_rate_per_dwelling DESC
LIMIT 1;
```

	National LGA Name 2021 character varying (50)	National Road Type character varying (50)	Time of day character varying (50)	avg_fatality_rate_per_dwelling numeric
1	Unincorporated SA	National or State Highway	Day	549.8282

Day of the week and Time of day

```
SELECT
  dl."National LGA Name 2021",
  ddw."Dayweek",
  dt."Time of day",
  ROUND(SUM(ft."Fatality Rate per Dwelling")::numeric, 4) AS avg_fatality_rate_per_dwelling
FROM FactFatalities ft
JOIN DimLocation dl
  ON ft."LocationID" = dl."LocationID"
JOIN DimRoad dr
  ON ft."RoadID" = dr."RoadID"
JOIN DimDayWeek ddw
  ON ft."DayweekID" = ddw."DayweekID"
JOIN DimTime dt
  ON ft."TimeID" = dt."TimeID"
JOIN DimDate d
  ON ft."DateID" = d."DateID"
WHERE d."Year" BETWEEN 2020 AND 2024
GROUP BY CUBE(dl."National LGA Name 2021",ddw."Dayweek", dt."Time of day")
HAVING dl."National LGA Name 2021" = 'Unincorporated SA'
    AND dt."Time of day" = 'Day'
    AND ddw."Dayweek" IS NOT NULL
ORDER BY avg_fatality_rate_per_dwelling DESC
LIMIT 1;
```

	National LGA Name 2021 character varying (50)	Dayweek character varying (50)	Time of day character varying (50)	avg_fatality_rate_per_dwelling numeric
1	Unincorporated SA	Sunday	Day	171.8213

Figure 5.3.3 SQL Result for business query 3

5.4 From 2020 to 2024, which combination of road user type and age group had the highest number of fatalities on national or state highways.

1. StarNet

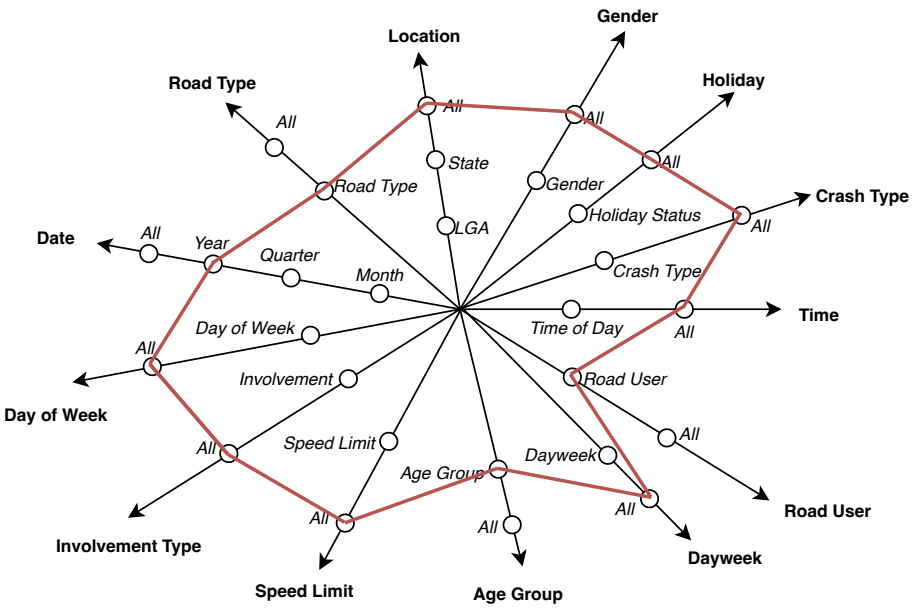


Figure 5.4.1 StarNet for business query 4

2. Visualisation

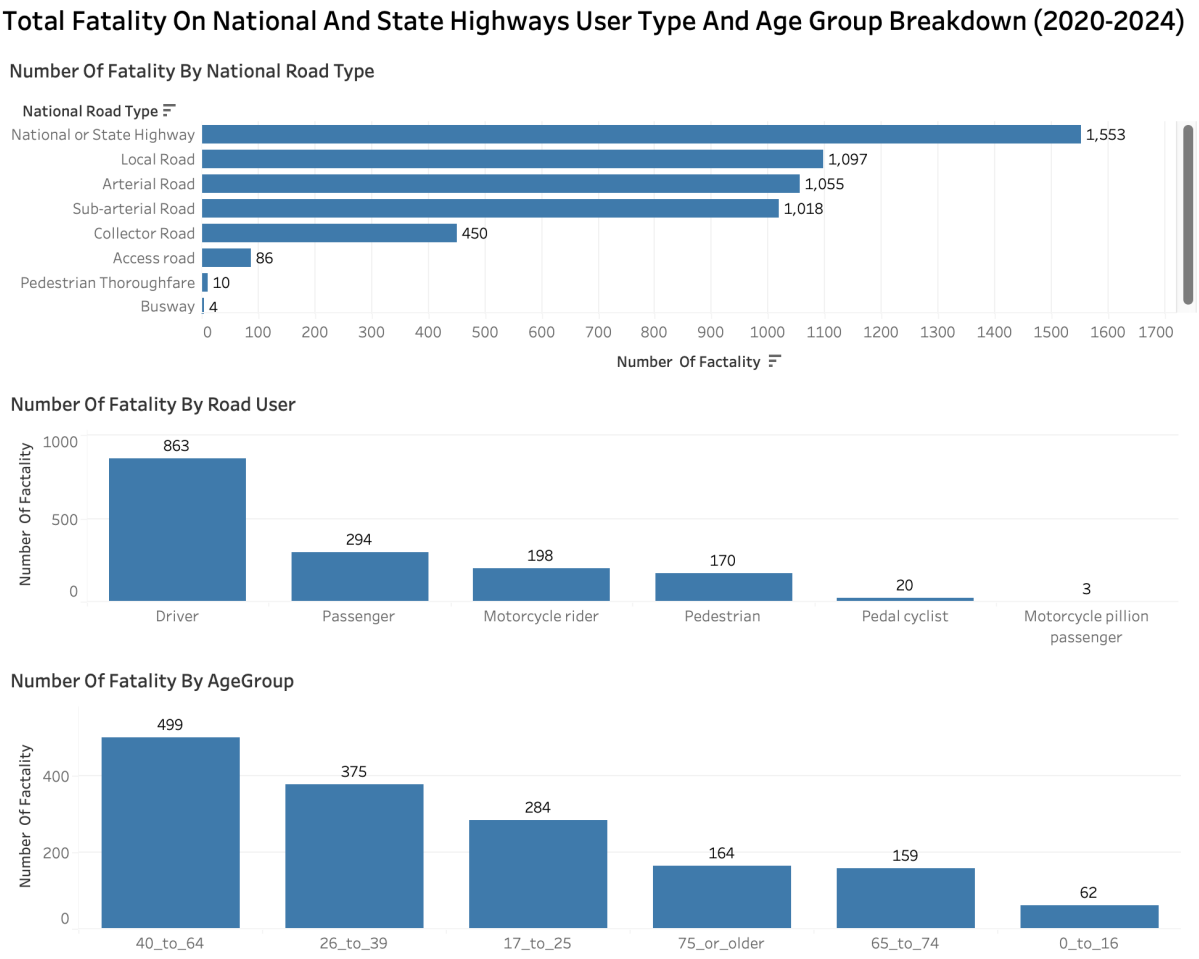


Figure 5.4.2 Dashboard for business query 4

Over the past five years, a total of 1553 crashes occurred on national or state highway. Specifically, drivers accounted for the highest number with 863, and the 40-64 age group showed a similar pattern, with 499 fatalities.

3. Cube Design

```
SELECT
  dr."National Road Type",
  du."Road User",
  dag."Age Group",
  SUM(ft."Fatality") AS total_fatalities
FROM FactFatalities ft
JOIN DimUser du
  ON ft."UserID" = du."UserID"
JOIN DimAgeGroup dag
  ON ft."AgeGroupID" = dag."AgeGroupID"
JOIN DimRoad dr
  ON ft."RoadID" = dr."RoadID"
JOIN DimDate d
  ON ft."DateID" = d."DateID"
WHERE dr."National Road Type" = 'National or State Highway'
  AND d."Year" BETWEEN 2020 AND 2024
 AND du."Road User" IS NOT NULL
GROUP BY CUBE(dr."National Road Type",du."Road User", dag."Age Group")
HAVING dr."National Road Type" IS NOT NULL
ORDER BY total_fatalities DESC
LIMIT 3;
```

	National Road Type character varying (50)	Road User character varying (50)	Age Group character varying (50)	total_fatalities bigint
1	National or State Highway	[null]	[null]	1543
2	National or State Highway	Driver	[null]	863
3	National or State Highway	[null]	40_to_64	499

Figure 5.4.3 SQL Result for business query 4

5.5 From 2020 to 2024, which road user type had the highest number of fatalities in crashes involving articulated trucks on high-speed roads?

1. StarNet

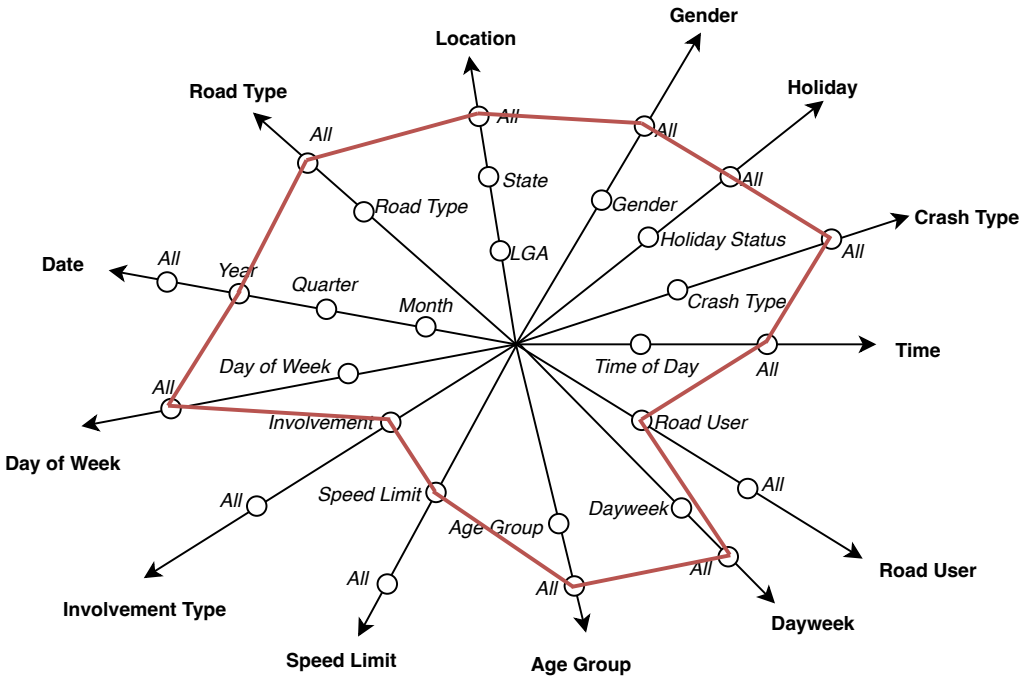


Figure 5.5.1 StarNet for business query 5

2. Visualisation

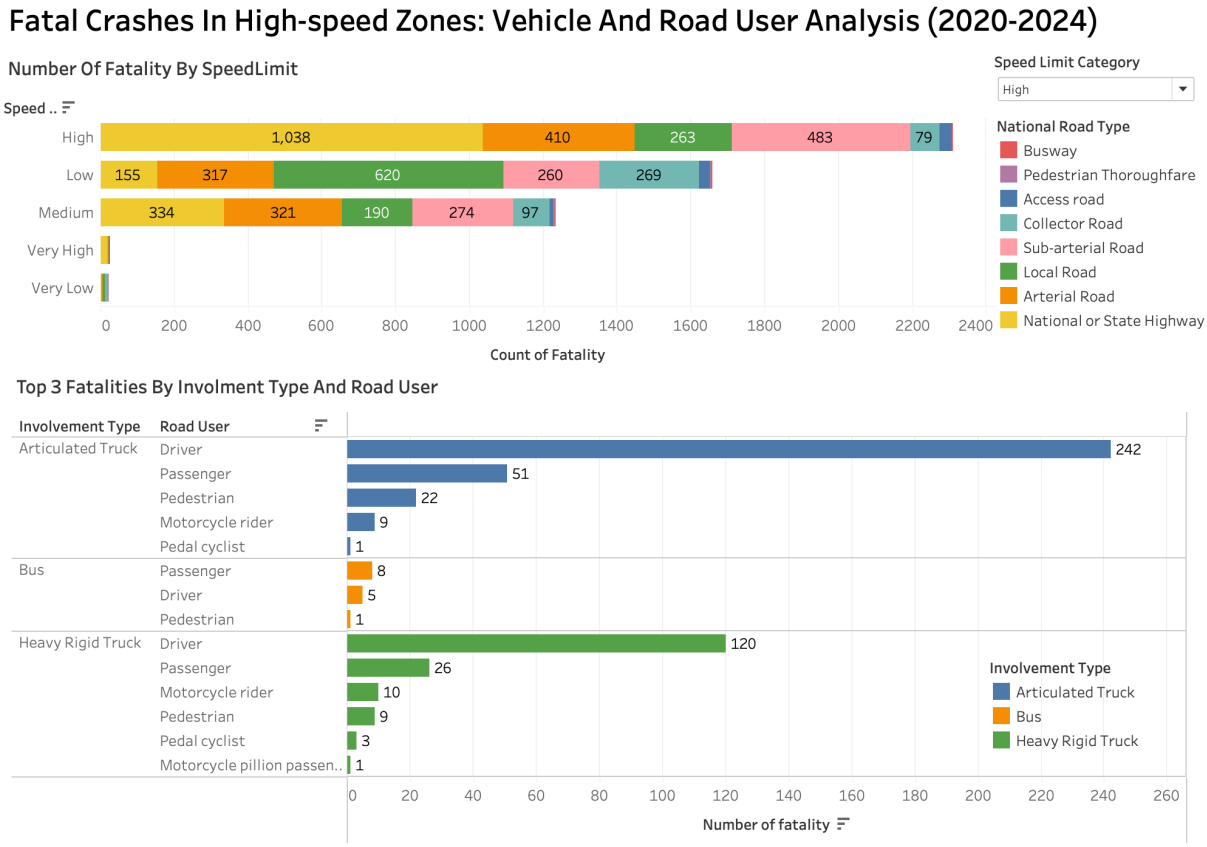


Figure 5.5.2 Dashboard for business query 5

The high-speed category exhibited the highest number of fatalities. A total of 242 drivers were killed in crashes involving articulated trucks, which was the largest proportion.

3. Cube Design

```
SELECT
  du."Road User",
  ds."Speed Limit Category",
  di."Involvement Type",
  SUM(ft."Fatality") AS total_fatalities
FROM FactFatalities ft
JOIN DimUser du
  ON ft."UserID" = du."UserID"
JOIN dimspeedlimit ds
  ON ft."SpeedID" = ds."SpeedID"
JOIN diminvolvement di
  ON ft."InvolvementID" = di."InvolvementID"
JOIN DimDate d
  ON ft."DateID" = d."DateID"
WHERE d."Year" BETWEEN 2020 AND 2024
  AND du."Road User" IS NOT NULL
GROUP BY CUBE(du."Road User",ds."Speed Limit Category",di."Involvement Type")
HAVING du."Road User" IS NOT NULL
  AND ds."Speed Limit Category" IS NOT NULL
  AND di."Involvement Type" IS NOT NULL
ORDER BY total_fatalities DESC
LIMIT 1;
```

	Road User character varying (50)	Speed Limit Category character varying (50)	Involvement Type character varying (50)	total_fatalities bigint
1	Driver	High	Articulated Truck	242

Figure 5.5.3 SQL Result for business query 5

6. Association Rule Mining

Association rule mining was used to identify patterns and co-occurring conditions related to traffic fatalities. The analysis aimed to find meaningful combinations of factors—such as road user type, location, time of day, and vehicle involvement—that tend to appear together in fatal crashes.

Apriori algorithm is a widely used method for association rule mining, and it is relatively simple and easy to interpret compared with FP-Growth and ECLAT algorithms. Moreover, according to ChatGPT, the Apriori algorithm is efficient in identifying frequent patterns and association rules in small or medium-sized datasets, although it does not perform well in terms of memory usage and runtime efficiency [5]. Since the fatality dataset size was relatively small and for the sake of implementation convenience, this report employed the Apriori algorithm and focused on the top 5 rules.

6.1 Setup

All association rule mining processes are implemented in Jupyter Notebook (refer to the *CITS5504_Project1_Data_processing.ipynb* notebook for full details).

Considering the correlations between risk factors and practicality of government implement, we selected 11 factors for further analysis.

```
# Based on the main risk factors
features = ['Dayweek', 'Road User', 'Gender', 'National Remoteness Areas','National LGA Name 2021',
            'National Road Type',  'Age Group', 'Day of week', 'Time of day', 'Involvement Type',
            'Speed Limit Category']
```

The key measures for association rule mining are support, confidence and lift. Support measures how frequently an itemset appears in the dataset, filtering out rare combinations that may not be meaningful. Confidence measures the conditional probability that a transaction having X also contains Y, and a high confidence means that when X happens, Y is likely to happen too. Lift compares the confidence of a rule to the expected confidence if X and Y were independent.

```
#Find the frequent itemsets
frequent_itemsets = apriori(df_arm,min_support=0.2,use_colnames =True)

#Check the length of rules
frequent_itemsets['length']=frequent_itemsets['itemsets'].apply(lambda x: len(x))

#Assume the length is 2 and the min support is >= 0.3
frequent_itemsets[ (frequent_itemsets['length']==2) &
                   (frequent_itemsets['support']>=0.3)]
```

```
#Assume the min confidence is 0.5
rules_con = association_rules(frequent_itemsets, metric="confidence",min_threshold=0.5)
```

```
#Assume the min lift is 1
rules_lift = association_rules(frequent_itemsets, metric="lift",min_threshold=1)

#Based on min confidence (=0.5),
#output antecedents, consequents, support, confidence and lift.
result_arm = rules_con[['antecedents','consequents','support','confidence','lift']]
```

```
#Find the rules whose confidence >= 0.5
new_result_arm = result_arm[result_arm['confidence']>=0.5]

ranked_result = new_result_arm.sort_values(by=['lift', 'confidence'], ascending=False)

ranked_result
```

With the help of ChatGPT, we understood how to set appropriate thresholds based on dataset size, rule significance and tolerance for noise [5] .Finally, we set the minimum support threshold to 0.2, filtering out factors that do not occur frequently; we set the confidence as 0.5 to filter out weak association rules, and we set the minimum lift to 1 to identify associations between two factors that are stronger than the expectation of individual factors.

```
# Data label
for x,i in enumerate(features):
  df_select.iloc[:,x] = i + ":" + df_select.iloc[:,x].astype(str)
```

```
df_select.head()
```

	Dayweek	Road User	Gender	National Remoteness Areas	National LGA Name 2021
0	Dayweek:Friday	Road User:Driver	Gender:Male	National Remoteness Areas:Inner Regional Austr...	National LGA Name 2021:Wagga Wagga
1	Dayweek:Friday	Road User:Driver	Gender:Female	National Remoteness Areas:Inner Regional Austr...	National LGA Name 2021:Hawkesbury
2	Dayweek:Friday	Road User:Driver	Gender:Female	National Remoteness Areas:Inner Regional Austr...	National LGA Name 2021:Northern Midlands
3	Dayweek:Friday	Road User:Driver	Gender:Female	National Remoteness Areas:Outer Regional Austr...	National LGA Name 2021:Armidale Regional
4	Dayweek:Friday	Road User:Passenger	Gender:Male	National Remoteness Areas:Unknow	National LGA Name 2021:Unknow

To better present the association rules, we categorised the numerical attributes and added labels to each value of factors to distinguish different attributes clearly.

6.2 Missing and “Unknown” Values

During the analysis, we identified that a significant portion of the dataset contained “unknown” values in key attributes, such as Involvement Type, National Road Type, LGA Name and National Remoteness Areas. These missing data posed a challenge for performing meaningful analysis, especially in tasks like association rule mining and map visualisations, where complete data is crucial for accurate results.

```
# Missing value
df_select.isna().sum().sort_values(ascending=False)
```

```
Involvement Type            48492
National Road Type          46136
National LGA Name 2021      45849
National Remoteness Areas   45520
Speed Limit Category         1485
Road User                     132
Age Group                     117
Time of day                    44
Gender                         34
Day of week                    13
Dayweek                         0
dtype: int64
```

After transforming the missing data to “Unknown” values, the dataset consisted of 56,874 records and 11 attributes.

```
# Replace the nan with Unknow
df_select = df_select.fillna("Unknow")
df_select.isna().sum()
```

```
Dayweek                     0
Road User                   0
Gender                      0
National Remoteness Areas   0
National LGA Name 2021      0
National Road Type          0
Age Group                   0
Day of week                 0
Time of day                 0
Involvement Type            0
Speed Limit Category        0
dtype: int64
```

```
#Find the frequent itemsets
frequent_itemsets = apriori(df_arm,min_support=0.2,use_colnames =True)

#Check the length of rules
frequent_itemsets['length']=frequent_itemsets['itemsets'].apply(lambda x: len(x))

#Assume the length is 2 and the min support is >= 0.3
frequent_itemsets[ (frequent_itemsets['length']==2) &
                   (frequent_itemsets['support']>=0.3)]
```

```
#Assume the min confidence is 0.5
rules_con = association_rules(frequent_itemsets, metric="confidence",min_threshold=0.5)
```

```
#Assume the min lift is 1
rules_lift = association_rules(frequent_itemsets, metric="lift",min_threshold=1)

#Based on min confidence (=0.5),
#output antecedents, consequents, support, confidence and lift.
result_arm = rules_con[['antecedents','consequents','support','confidence','lift']]
```

```
#Find the rules whose confidence >= 0.5
new_result_arm = result_arm[result_arm['confidence']>=0.5]

ranked_result = new_result_arm.sort_values(by=['lift', 'confidence'], ascending=False)

ranked_result
```

The results of top 5 rules are presented as follows:

ID	Antecedents	Consequents	Support	Confidence	Lift
1927	(Speed Limit Category:High, National LGA Name 2021:Unknow)	(Road User:Driver, National Remoteness Areas:Unknow, National Road Type:Unknow)	0.204030	0.551468	1.544807
1926	(Road User:Driver, National Remoteness Areas:Unknow, National Road Type:Unknow)	(Speed Limit Category:High, National LGA Name 2021:Unknow)	0.204030	0.571541	1.544807
1174	(Speed Limit Category:High, National LGA Name 2021:Unknow)	(Road User:Driver, National Remoteness Areas:Unknow)	0.204311	0.552229	1.544807
1171	(Road User:Driver, National Remoteness Areas:Unknow)	(Speed Limit Category:High, National LGA Name 2021:Unknow)	0.204311	0.571541	1.544807
1932	(Road User:Driver, National Remoteness Areas:Unknow)	(Speed Limit Category:High, National LGA Name 2021:Unknow, National Road Type:Unknow)	0.204030	0.570754	1.544368

The result suggests that the “Unknown” values cannot explain the relationships between different fatal risk factors, including National Road Type, National LGA Name 2021, and National Remoteness Areas. Therefore, these factors will be excluded from further analysis.

6.3 Association Rule Mining with valid factors

After dropping the attributes with large portion of missing and null values of other attributes, this dataset consisted of 55,140 records and 7 attributes.

```
# Drop features
# Considering the missing data and the uncontrollable factors
df_acc = df_acc.drop(columns=["Involvement Type","National Road Type","National LGA Name 2021","National Remoteness Areas"])
```

```
# Date Cleaning
df_acc = df_acc.dropna()
df_acc.shape
```

```
(55140, 7)
```

```
#Find the frequent itemsets
frequent_itemsets = apriori(df_arm,min_support=0.2,use_colnames =True)

#Check the length of rules
frequent_itemsets['length']=frequent_itemsets['itemsets'].apply(lambda x: len(x))

#Assume the length is 2 and the min support is >= 0.3
frequent_itemsets[ (frequent_itemsets['length']==2) &
                   (frequent_itemsets['support']>=0.3)]
#Assume the min confidence is 0.5
rules_con = association_rules(frequent_itemsets, metric="confidence",min_threshold=0.5)
```

```
#Assume the min lift is 1
rules_lift = association_rules(frequent_itemsets, metric="lift",min_threshold=1)

#Based on min confidence (=0.5),
#output antecedents, consequents, support, confidence and lift.
result_arm = rules_con[['antecedents','consequents','support','confidence','lift']]
```

```
#Find the rules whose confidence >= 0.5
new_result_arm = result_arm[result_arm['confidence']>=0.5]

ranked_result = new_result_arm.sort_values(by=['lift', 'confidence'], ascending=False)

ranked_result
```

The top 5 rules ranked by lift and confidence values are as follows:

ID	Antecedents	Consequents	Support	Confidence	Lift
14	(Night)	(Weekend)	0.236162	0.550055	1.342690
15	(Weekend)	(Night)	0.236162	0.576475	1.342690
26	(Male, Speed: High)	(Road User: Driver)	0.201106	0.604503	1.329145
25	(Road User: Driver, Male)	(Speed: High)	0.201106	0.587684	1.252410
16	(Road User: Driver)	(Speed: High)	0.266522	0.586012	1.248848

The third rule indicates that, 20.11% of crashes involved high-speed areas and male individuals, and these two factors are significantly associated with drivers. The confidence was 60.45%, suggesting more than half of road users in such crashes were drivers. Additionally, the lift value is greater than 1, which means the high-speed, male and drivers have a stronger strength of the association than the expectation of individual factors in these incidents.

6.4 Interpretation & Recommendations

For top 5 association rules identified above, the interpretations are as follows:

Rule 1: If a crash happened during the night, it was likely to occur on weekends.

Rule 2: If a crash happened on weekends, it was likely to occur during the night.

Rule 3: If the person was a male and the crash happened in high-speed areas, the person was likely to be a driver.

Rule 4: If the road user was a male driver, the crash was likely to occur in high-speed areas.

Rule 5: If the road user was a driver, the crash was likely to occur in high-speed areas.

In conclusion, a high proportion of crashes happened on weekends during the night. Meanwhile, male individuals and drivers exhibited a higher fatal risk, especially in crashes occurred in a high-speed areas.

To improve road safety and reduce the crashes, the government released two reports for 2023, including National Road Safety Strategy 2021-30 (Strategy) [6] and National Road Safety Action Plan 2023-25 (Action Plan) [2] , setting the agenda to improve road safety over the next 10 years. However, these reports did not provide detailed action plan. Based on the top 5 association rules, we suggest that:

1. Boosting Infrastructure Investment

The main risk factors contributing to nighttime crashes may be poor visibility and conspicuity. To address this, the government should enhance road lighting systems and install more warning signs along roadsides, especially on main roads used during weekend nights.

2. Strengthening Supervision Measures

Supervision measures could serve as an effective way to mitigate risks, such as installing more digital surveillance equipment, smart radar systems and increase traffic patrols in high-speed areas and during the weekday night-time period.

3. Enhancing Education

In terms of drivers, government should strengthen traffic education and increase the frequency of traffic knowledge testing, especially for male drivers, to improve their risk perception, recognition and response abilities in high-risk traffic environments [7].

7. Conclusions

This project developed a data warehouse to support the analysis of traffic fatality data in Australia. ARDD and ABS datasets were integrated and structured into a star schema with one fact table and thirteen dimension tables. The schema allows flexible querying across date, location, demographics, and other relevant dimensions.

Key Findings

Business queries and visualisations were used to explore high-risk patterns by region, gender, age group, road type, time and road user type. Over the past five years, we found that Northern Territory exhibited the highest average fatality rate, and Brisbane also reported the highest rate. In the local government area - Unincorporated SA, the most crashes happened on national or state highways, with daytime crashes peaking on Sunday. On national or state highway, most crashes involved driver and 40-64 age group. Moreover, high-speed areas are strongly associated with crash occurrences, and the highest proportion involving articulated trucks and drivers.

Association rule mining helped identify combinations of factors that frequently co-occurred in fatal crashes. These findings support the potential for data-driven recommendations targeting weekday, night, driver, male and high-speed areas. For the sake of road safety, the government should take action through infrastructure investment, supervisory measures, and road safety education.

Overall, each record in this dataset represents a real life loss. Road safety is not just a technical or policy problem, but a matter of human lives. For the sake of a safer environment, it requires the joint efforts of both government and individuals, especially for male drivers. Only through that, we can build a stable, safe and harmonious society where every journey ends up safely [7].

Limitations

There are some limitations. The data only includes fatal crashes and does not cover minor or serious injuries. Time coverage is limited to one year, and some attributes are not fully detailed. Future work could include adding more years of data, integrating other factors such as weather or road condition, and extending the schema to support predictive analysis.

References

[1] Department of Infrastructure, Transport, Regional Development and Communications. "Road Deaths Australia – Monthly Bulletin, December 2024". Accessed Mar. 10, 2025. [Online]. Available: https://www.bitre.gov.au/publications/ongoing/road-deaths-australia-monthly-bulletin

[2] Australian Government. "National Road Safety Action Plan 2023–2025".Accessed Mar. 11, 2025. [Online]. Available: https://www.roadsafety.gov.au/action-plan/national-road-safety-action-plan-2023-25

[3] UWA CSSE. "Data Warehouse Project 1 - S1 2025". Accessed Mar. 5, 2025.[Online]. Available: https://csse-uwa.gitbook.io/data-warehouse-project-1-s1-2025

[4] Australian Bureau of Statistics. Accessed Mar. 15, 2025.[Online]. Available: https://www.abs.gov.au/statistics/people/population/national-state-and-territory-population/sep-2024

[5] OpenAI. "ChatGPT (GPT-4)". Accessed Apr. 10, 2025.[Online]. Available: https://chat.openai.com

[6] National Road Safety Strategy 2021-30 (Strategy) . Accessed Mar. 20, 2025.[Online]. Available: https://www.roadsafety.gov.au/nrss

[7] N. Zhang, M. Fard, J. Xu, et al., "Road safety: The influence of vibration frequency on driver drowsiness, reaction time, and driving performance," *Appl. Ergon.*, vol. 114, Art. no. 104148, 2024, doi: 10.1016/j.apergo.2023.104148.