CITS 5504: Data Warehousing project 1

Data Warehouse Design and Analysis for fatal Road Crashes:

Group members:

1. **Isaiah Rama Veera - 24078803**
2. **Flavian Jerotich -24001784**

1. Introduction

Road safety is a major global concern, with governments worldwide continually working towards reducing road fatalities. Despite significant efforts and policy implementations, road crashes remain a critical public health issue in Australia, affecting numerous lives annually. In everyday life, car crashes occur frequently during routine journeys to workplaces, schools, or shopping centers. These incidents vary widely in severity, from minor injuries or property damage to tragic fatalities. Although strict rules and preventive measures have been implemented, crashes continue to occur regularly. The objective of this project is to conduct a comprehensive analysis of road crash and fatality data from Australia, provided by Australian Roads Deaths Database [4]. By exploring and understanding patterns in this data, the project aims to identify major contributing factors behind these incidents. Ultimately, the findings can inform targeted strategies and interventions, aiding the government, policymakers and stakeholders in significantly reducing—and potentially preventing—avoidable crashes, thereby decreasing road fatalities nationwide.

2. Project Scope and Objectives

Project Scope and Overview

This project focuses on the development of a data warehouse designed to support the analysis of fatal road crashes in Australia. The primary objective is to consolidate multiple datasets,

including fatal crash records, individual fatality details, and LGA-level dwelling statistics—into a unified platform that supports advanced analytical operations.

The key components of the project include:

- **Data Integration:** Combining crash data, fatality records, and dwelling counts to form a comprehensive and structured dataset.
- **ETL (Extract Transform Load) Process:** Implementing extraction, transformation, and loading workflows in Python to clean, merge, and prepare the data for loading into a PostgreSQL data warehouse (road_safety_dw).
- **Dimensional Modeling:** Designing a dimensional schema using Kimball's methodology [2], comprising eight-dimensional tables and two fact tables. This model supports detailed and flexible analysis across multiple perspectives.
- **OLAP (Online Analytical Processing) Operations:** Performing analytical operations such as slicing, dicing, roll-up, and drill-down to answer various business and safety-related queries.
- **Association Rule Mining:** Using the Apriori algorithm to identify patterns and rules in the data—particularly those involving road user types—to better understand risk factors.
- **Visualization:** Creating interactive dashboards and analytical charts using Tableau to present trends and insights effectively.
- **Recommendations:** Drawing actionable conclusions from the analysis to assist government policymakers and stakeholders in improving road safety outcomes.

Project Objectives

- **To develop** a centralized data warehouse that integrates fatal crash and fatality data from multiple sources while ensuring data quality, consistency, and referential integrity.
- **To enable** multi-dimensional analysis by supporting efficient OLAP operations—such as slicing, dicing, drilldowns, and roll-ups—across key dimensions including time, location, crash conditions, and victim demographics.

- **To uncover** hidden risk patterns by applying association rule mining techniques (e.g., the Apriori algorithm) and evaluating relationships based on support, confidence, and lift, particularly where road user types are involved.
- **To generate** clear, actionable insights and evidence-based recommendations that can inform road safety policy and targeted interventions.
- **To deliver** a comprehensive end-to-end solution, encompassing data integration, dimensional modeling, ETL pipelines, analytical querying, and visual reporting using tools such as PostgreSQL, Python, and Tableau.

3. Data Sources and preprocessing

Data Overview

This project integrates multiple datasets to construct a unified data warehouse aimed at analyzing fatal road crashes in Australia. The primary data sources were obtained from the Bureau of Infrastructure, Transport and Regional Economics (BITRE) and the Australian Bureau of Statistics (ABS).

Two key Excel files formed the foundation of this project: bitre_fatal_crashes_dec2024.xlsx and bitre_fatalities_dec2024.xlsx [4].

i. The ***BITRE_Fatal_Crash sheet*** in the first file contains 51,284 rows and 20 columns, with each row representing a crash. Key fields include Crash ID, State, Month, Year, Day of Week, Time, Crash Type, Speed Limit, Number Fatalities, and vehicle involvement indicators such as Bus Involvement, Heavy Rigid Truck Involvement, and Articulated Truck Involvement. Geographic data is also included, covering Remoteness Area, SA4 Name, and LGA Name, as well as flags for Christmas Period and Easter Period. These features support both crash condition and spatial analysis.

ii. The second sheet, ***BITRE_Fatality*** from the bitre_fatalities_dec2024.xlsx file, provides detailed fatality-level data, including demographic fields such as Gender, Age, Age-Group, and Road User Type, in addition to crash-level attributes. This sheet contains 56,874 records across 23 columns, each row representing an individual fatality. Time of day classifications and holiday indicators are also included. Notably, both Excel files also

contain "Index" and "Appendix" sheets, which were excluded from our analysis as they provided no relevant data for modeling or analytics.

In addition to the above, both files contain date-specific summary sheets: ***BITRE_Fatal_Crash_Count_By_Date*** and ***BITRE_Fatality_Count_By_Date***. These sheets include 13,149 rows each and provide aggregated statistics such as Number of Fatal Crashes and Number of Fatalities by date. Even after combining the **BITRE_Fatal_Crash** and **BITRE_Fatality** sheets by Crash ID, we have 51,284 rows in each main sheet after removing duplicates based on crash Id, but the date-specific sheets still contain only 13,149 unique dates. If we attempted to join the data using a combination of Year, Month, and Day of Week, it would likely create duplicates and introduce null values. Moreover, the counts of fatal crashes and fatalities by date are already summarized in those sheets. For a more meaningful analysis, aggregated data (yearly, monthly) is preferred over a day-by-day analysis as it is not possible to determine the date exactly from the columns we have been provided in fatal crash and fatality.

Therefore, to avoid duplication and null value issues—and because aggregated analysis is more relevant—we have chosen to ignore the fatal crash count by date and fatality count by date for our detailed analysis.

iii. We also utilized a CSV file titled ***LGA (count of dwellings).csv*** as the third file, sourced from ABS data [5]. This dataset provides dwelling counts for 556 Australian Local Government Areas (LGAs). Although the file originally included metadata and footnotes, it was cleaned and truncated to retain only two relevant columns: LGA Name and Dwelling Count. These dwelling counts allow us to normalize crash data by population exposure—an important factor when analyzing crash risk per capital or per household.

4. Data Loading and Cleaning Process

Extraction:

The ETL (Extract, Transform, Load) process for this project involved several carefully designed steps to ensure data quality and integration across multiple sources. The datasets used included fatal crash data, fatality records, and LGA-level dwelling counts. The goal of this process was to unify all data into a clean and analyzable format suitable for loading into the data warehouse.

Loading:

We started by loading three primary datasets using pandas. The two Excel sheets—BITRE_Fatal_Crash and BITRE_Fatality—were extracted from their respective workbooks with headers starting from the 5th row as the first 4 rows contained metadata instead to actual data, while the CSV file containing dwelling data was loaded while skipping the first 11 metadata rows. We kept only the first two columns in the dwelling dataset, named them LGA_Name and dwelling_count, and removed any irrelevant rows after the data.

```
[39]: # Loading all Data:
df_crash = pd.read_excel("bitre_fatal_crashes_dec2024.xlsx", sheet_name="BITRE_Fatal_Crash",skiprows=4)
df_fatality = pd.read_excel("bitre_fatalities_dec2024.xlsx", sheet_name="BITRE_Fatality",skiprows=4)
# LGA dwellings
df_dwellings = pd.read_csv("LGA (count of dwellings).csv", header=None, skiprows=11, sep=",")
```

Exploration and Transformation:

Initial inspections of the two dataframes had crash associated with a unique Crash ID and fatalities listed per victim in the BITRE_Fatal_Crash and BITRE_Fatality sheet. We noticed that the data comprised of object and integer data types while time was stored as datetime.time. Initially, inconsistencies were identified in the column naming conventions across different data files. For instance, the column labeled *"Bus \nInvolvement"* in the fatal crash data contained newline characters, which could cause issues during data processing. This column was renamed "Bus Involvement" for clarity and ease of analysis. Such standardization ensures consistent referencing of columns during the data merging. “-9” (String), -9 (Integer) and Unknown were used as place holders for missing data.

The categorical column "Road User" in the fatalities dataset initially contained unclear entries labeled *"Other/-9"*. To provide a clearer analytical categorization, these values were replaced with a more meaningful category, *"Unknown"*. This allowed for better interpretability during analysis, reducing ambiguity. The dwelling count column in the dwelling dataset was converted to numeric.

The next step was to check for unique categorical values. We wrote a helper function to explore all object type columns and print out the first 20 unique values each. We observed that Crash type had only *single* and *multiple* values. Road use had 7 unique types, Speed limit 100, 80, -9,

and even strings like "<40". We then checked for missing values to identify any NaN values, count 0f '-9' and 'unknown' and get the percentage of the missing data to help us decide the methods to use in data cleaning. We got the following information:

- Most problematic columns with missing data (NaN): Time, SA4 Name 2021, National LGA Name 2021 (in df_crash and df_fatality)
- Most problematic columns with -9: Bus Involvement, Heavy Rigid Truck Involvement, Articulated Truck Involvement, Speed Limit, Gender, Age, Age Group (in df_fatality)
- Most problematic columns with "Unknown" values:National Remoteness Areas, SA4 Name 2021, National LGA Name 2021, Road User, Day of week, Time of Day (in df_crash and df_fatality)

We calculated the percantage of missing values (Nan,"-9", Unknown) for "Heavy Rigid Truck Involvement" had **35.5%** missing data, "National Remoteness Area","National LGA Name 2021" and "SA4 Name 2021" were missing in approximately **80%** of rows.

We checked for any **duplicates** in the Crash ID columns in both the fatal crash and fatality datasets. We then dropped the duplicates as keeping the duplicates would artificially inflate fatality counts or crash counts. This ensured a 1:1 match between each fatal crash and fatality record. The shape output shows that df_crash remained at (51284, 20) after removing duplicates, implying either no duplicates or few. The same check for df_fatality showed (56874, 23) reduced to (51284, 23), meaning around 5,590 duplicate rows were removed there.

Merging the files and further cleaning:

We performed an outer join on the fatal crash and fatality datasets using Crash ID. This was to ensure that if a fatal crash exists without a corresponding fatality row, or vice versa, that record is still preserved. Suffixes ("_crash", "_fatal") were applied to distinguish the columns that exist in both data frames. "Crash ID" is converted to a string type to avoid confusion with numeric IDs as it helps ensure consistent joins and lookups. Age is converted to become numeric; any non-convertible values become NaN (missing). Age being numeric allows proper statistical analysis and filtering.

We systematically rename columns that have _crash or _fatal suffixes, choosing which version of each column to keep. For instance, Year_crash becomes Year. We then remove any duplicated columns that might remain after renaming. This step cleans up the merged table, so we only have *one* column for each attribute, preventing duplicate columns. It also keeps the final column names consistent (e.g., “Speed Limit” instead of “Speed Limit_crash” and “Speed Limit_fatal” repeated).

The next step was to merge the Dwelling count data by matching National LGA Name 2021 from the merged crash fatality table with LGA_Name in the dwelling's csv. We use a *left join* so that all crash/fatality records are retained, even if some LGAs do not have a matching dwelling count. We drop the extra column LGA_Name (which is duplicated from df_dwellings) after the merger.

We ran **helper functions** to check for missing values and unique categories in the merged file. This enables us to confirm whether the final data is as expected and if additional cleaning is needed. From the percentage of missing values (Nan,”-9”, Unknown) for "Heavy Rigid Truck Involvement" had **35.5%** missing data, “National Remoteness Area”, "National LGA Name 2021" and "SA4 Name 2021" were missing in approximately **80%** of rows and dwelling count approximately **80%**. With this percentage of missing data, we could not drop the columns of missing data as it would lead to the loss of the remaining 20% data which we already have from these columns. After which we applied a comprehensive data cleaning and imputation process to prepare the data for analysis detailed below:

- First, we standardized missing values by converting all instances of -9 and "Unknown" to actual missing values (NaN). We then cleaned the **Speed Limit** column by replacing <40 with 39, converting the entire column to numeric, and filling missing values with the column’s median. Similarly, for the **Age** column, we replaced -9 with NaN, ensured the values were numeric, and filled any gaps using the median age.
- To handle categorical variables, we applied **label encoding** to temporarily convert them into numerical values. We then used the **Iterative Imputer** (MICE technique) to intelligently estimate and fill remaining missing values across all columns. Once

imputation was complete, we **decoded** the categorical columns back to their original string labels.

- For a few specific columns like "SA4 Name 2021", "Time", and "National LGA Name 2021", any remaining missing values were replaced with the label "Unknown". In the **dwelling_count** column, missing values were replaced with 0, and any negative values were clipped to zero to maintain valid counts. This strategy ensures that the data is properly cleaned, missing values are inputted using appropriate methods, and all key columns are in the correct format for further analysis. Something to note is that "*When Dwelling count is 0 it means the value is missing and "nan" also means the value is missing basically "Unknown" in categorical columns.*"

Imputation steps screenshot:

```
def clean_df(df, max_iter=10):

    # Create a copy of the DataFrame
    data = df.copy()

    # Step 1: Replace all placeholders with NaN in every column.
    data.replace({'-9': np.nan, -9: np.nan, 'Unknown': np.nan}, inplace=True)

    # Step 2: For object-type columns (except 'Speed Limit'), replace '-9' with "Unknown"
    for col in data.select_dtypes(include=['object']).columns:
        if col != 'Speed Limit':
            data[col] = data[col].replace({'-9': 'Unknown', -9: 'Unknown'})

    # Step 3: Process 'Speed Limit' column separately
    if 'Speed Limit' in data.columns:
        # Replace "<40" with "39"
        data['Speed Limit'] = data['Speed Limit'].astype(str).str.replace('<40', '39', regex=False)
        # Replace '-9' (string) and -9 (numeric) with NaN
        data['Speed Limit'] = data['Speed Limit'].replace({'-9': np.nan, -9: np.nan})
        # Convert to numeric (non-convertible values become NaN)
        data['Speed Limit'] = pd.to_numeric(data['Speed Limit'], errors='coerce')
        # Fill missing values with the median
        median_speed = data['Speed Limit'].median()
        data['Speed Limit'].fillna(median_speed, inplace=True)
        # Convert to integer
        data['Speed Limit'] = data['Speed Limit'].astype(int)

    # Step 4: Process 'Age' column
    if 'Age' in data.columns:
        # Convert to numeric if object type
        if data['Age'].dtype == 'object':
            data['Age'] = pd.to_numeric(data['Age'], errors='coerce')
        # Replace -9 with NaN
        data['Age'].replace(-9, np.nan, inplace=True)
        # Fill missing values with the median
        median_age = data['Age'].median()
        data['Age'].fillna(median_age, inplace=True)

    # Step 5: Identify numeric and categorical columns
    numeric_cols = data.select_dtypes(include=[np.number]).columns.tolist()
    cat_cols = data.select_dtypes(include=['object']).columns.tolist()
    encoders = {}

    # Step 6: Label-encode categorical columns.
    for c in cat_cols:
        data[c] = data[c].astype(str).fillna('NaNCat')
        le = LabelEncoder()
        le.fit(data[c])
        data[c] = le.transform(data[c])
        encoders[c] = le

    # Combine all columns for imputation
    all_cols = numeric_cols + cat_cols
    X = data[all_cols].values

    # Step 7: Iterative imputation (MICE-like)
    imputer = IterativeImputer(max_iter=max_iter, random_state=42)
    X_imputed = imputer.fit_transform(X)
```

```
    # Reassign numeric columns
    data[numeric_cols] = X_num

    # Step 8: Decode categorical columns back to original labels.
    for idx, c in enumerate(cat_cols):
        col_arr = np.rint(X_cat[:, idx]).astype(int)
        le = encoders[c]
        col_arr = np.clip(col_arr, 0, len(le.classes_)-1)
        data[c] = le.inverse_transform(col_arr)
        data[c] = data[c].replace('NaNCat', np.nan)

    # Step 9: For specified columns, fill missing values with "Unknown"
    for col in ["SA4 Name 2021", "National LGA Name 2021", "Time"]:
        if col in data.columns:
            data[col].fillna("Unknown", inplace=True)

    # Step 10: For 'dwelling_count', convert to numeric, fill missing with 0, and clip negatives to 0.
    if "dwelling_count" in data.columns:
        data["dwelling_count"] = pd.to_numeric(data["dwelling_count"], errors='coerce')
        data["dwelling_count"].fillna(0, inplace=True)
        data["dwelling_count"] = data["dwelling_count"].clip(lower=0)
        data["dwelling_count"] = data["dwelling_count"].round().astype(int)

    # Step 11: Convert specific numeric columns to integer (if they exist)
    for col in ["Month", "Year", "Number Fatalities", "Speed Limit", "Age"]:
        if col in data.columns:
            data[col] = data[col].round().astype(int)

    return data

# Apply the cleaning function to your DataFrame
df_merged_final = clean_df(df_merged_final)
```

Post cleaning confirmation of the merged dataset.

We can see that all the missing values have been dealt with.

```
# Define all your loaded DataFrames in a dictionary
dataframes = {
    "df_merged_final": df_merged_final
}

check_missing_values(dataframes)
print(df_merged_final.dtypes)
check_unique_categories(dataframes)
```

	Missing Values	-9 Count	"Unknown" Count
Crash ID	0	0	0
State	0	0	0
Month	0	0	0
Year	0	0	0
Dayweek	0	0	0
Time	0	0	0
Crash Type	0	0	0
Number Fatalities	0	0	0
Bus Involvement	0	0	0
Heavy Rigid Truck Involvement	0	0	0
Articulated Truck Involvement	0	0	0
Speed Limit	0	0	0
National Remoteness Areas	0	0	0
SA4 Name 2021	0	0	0
National LGA Name 2021	0	0	0
National Road Type	0	0	0
Christmas Period	0	0	0
Easter Period	0	0	0
Day Of Week	0	0	0
Time of Day	0	0	0
Road User	0	0	0
Gender	0	0	0
Age	0	0	0
Age Group	0	0	0
Time of day	0	0	0
dwelling_count	0	0	0

	Total Issues	Percentage
Crash ID	0	0.0
State	0	0.0
Month	0	0.0
Year	0	0.0
Dayweek	0	0.0
Time	0	0.0
Crash Type	0	0.0
Number Fatalities	0	0.0
Bus Involvement	0	0.0
Heavy Rigid Truck Involvement	0	0.0
Articulated Truck Involvement	0	0.0
Speed Limit	0	0.0
National Remoteness Areas	0	0.0
SA4 Name 2021	0	0.0
National LGA Name 2021	0	0.0
National Road Type	0	0.0
Christmas Period	0	0.0
Easter Period	0	0.0
Day Of Week	0	0.0
Time of Day	0	0.0
Road User	0	0.0
Gender	0	0.0
Age	0	0.0
Age Group	0	0.0
Time of day	0	0.0
dwelling_count	0	0.0

5. Concept Hierarchies

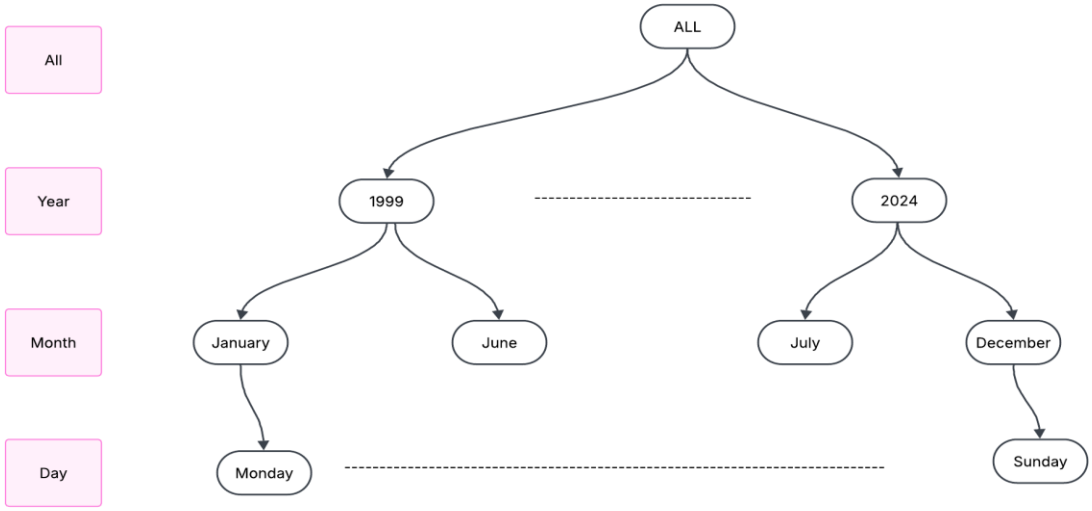
In designing a good data warehouse, a critical component is the proper construction of dimension tables. These dimensions not only provide the descriptive context for the fact tables but also enable multi-level analysis through clearly defined hierarchies. The following sections detail each dimension used in our road safety data warehouse, discussing its hierarchical structure, purpose, and how it supports business queries and decision-making.

1. **Date Dimension**

Hierarchy: **All → Year→ Month → Day**

This allows us to analyse trends in road fatalities and crashes over time. It's a fundamental dimension in any data warehouse because time-based slicing, filtering, and aggregation are

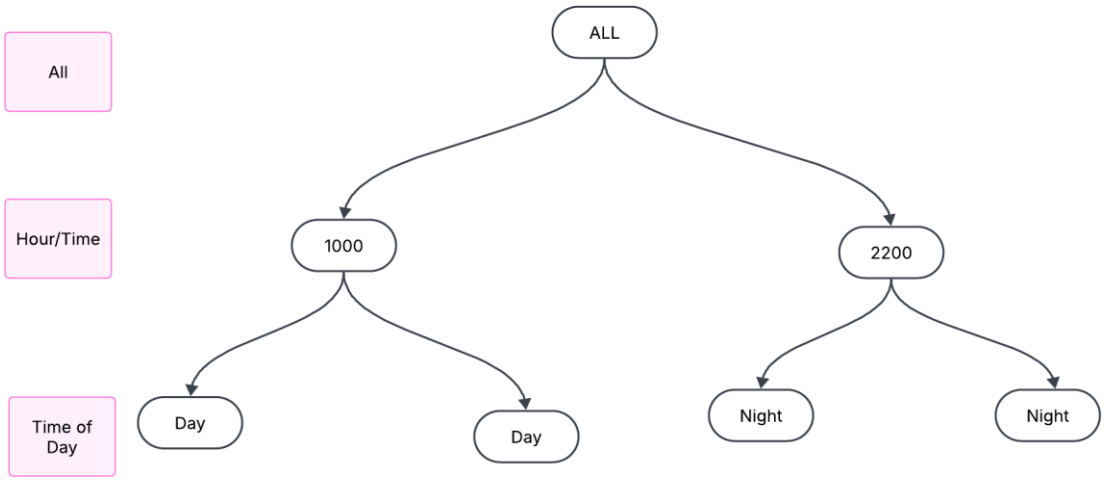
essential in understanding patterns. It supports drilldowns and aggregations over years, months, and days.



2. Time Dimension

This dimension captures the specific time of crashes and links it to intervals of the day (e.g., day, night) to support fine-grained temporal analysis.

Hierarchy: All → Time/Hour → Time of Day (Day, Night)

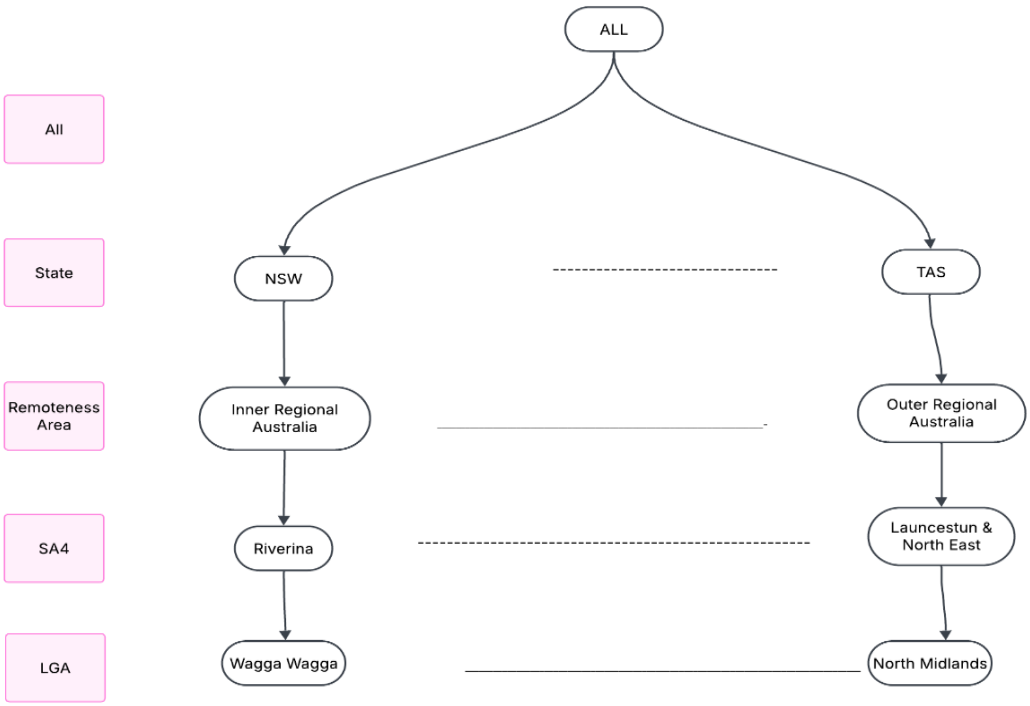


This helps analyse crash patterns by time, such as peak hours or late-night incidents, which is critical for time-based interventions.

3. Location Dimension

Hierarchy: All → State →RemotenessArea(added layer) → SA4 → LGA

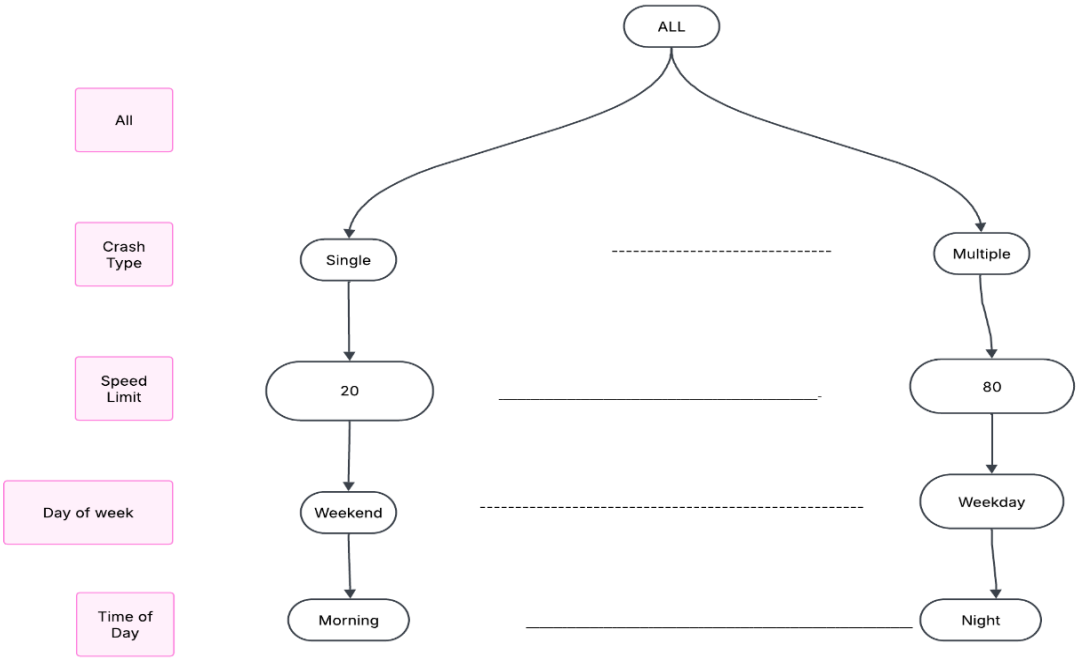
- **Purpose:** Captures hierarchical geographic data and dwelling counts, facilitating location-specific risk analysis and the calculation of metrics like fatalities per dwelling. Placing **dwelling_count** in the LocationDimension is appropriate because it adds important context about the LGA (Local Government Area) where the crash occurs, enabling location-specific analysis (e.g., fatalities per dwelling). This transforms dwelling_count into a descriptive attribute of the LGA, letting analysts normalize fatalities or crashes by population/household context and compare different regions more fairly.



4. Crash Dimension

Hierarchy: **All → Crash Type → Speed Limit → Weekday/Weekend → Time of Day**

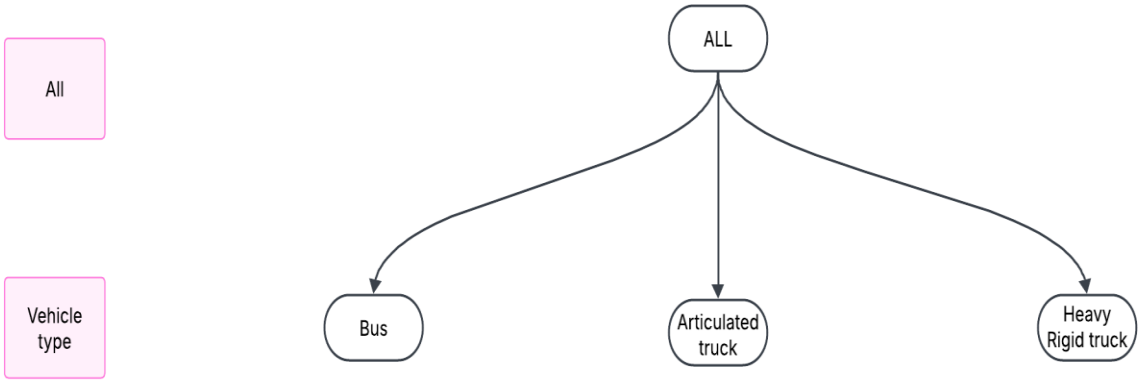
Purpose: Differentiates crash conditions by capturing attributes such as crash type and speed limit, which are key for understanding crash severity.



5. Vehicle dimension

The Vehicle dimension is composed of binary flags indicating the involvement of specific vehicle types, such as buses, heavy rigid trucks, or articulated trucks. There is no inherent hierarchy since these are categorical variables used primarily for filtering and segmentation. However, their inclusion is vital for analysing the impact of heavy vehicles on crash outcomes. This dimension helps in distinguishing whether crashes involving larger vehicles tend to have different fatality patterns compared to those that do not involve such vehicles.

Hierarchy: No inherent hierarchy.



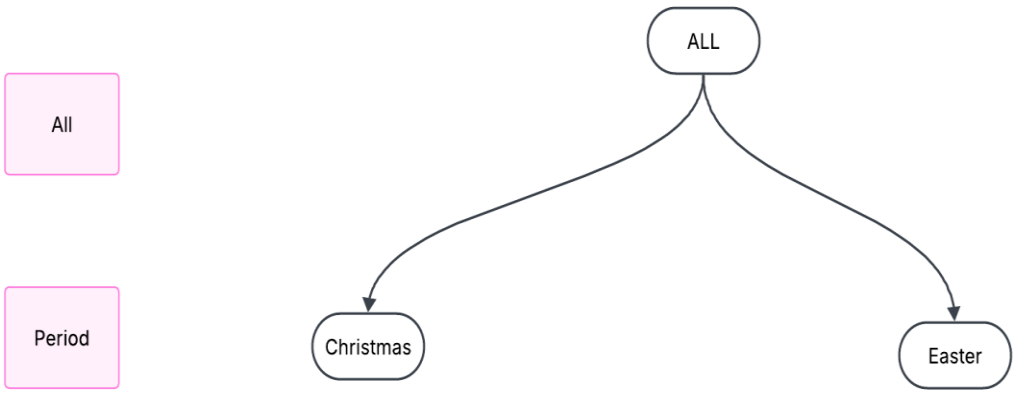
6. Period Dimensions

Hierarchy: Period Type (e.g., **Christmas**, **Easter**). **It is a single level categorization.**

The Period dimension classifies crash records based on the time in which they occur. It distinguishes between special periods, such as Christmas or Easter, and normal periods. Although it does not have multiple levels, it is crucial for analysing seasonal effects and assessing the impact of holiday-specific road safety campaigns. By isolating these periods, analysts can determine if certain times of the year present higher risks and require different enforcement or public awareness strategies.

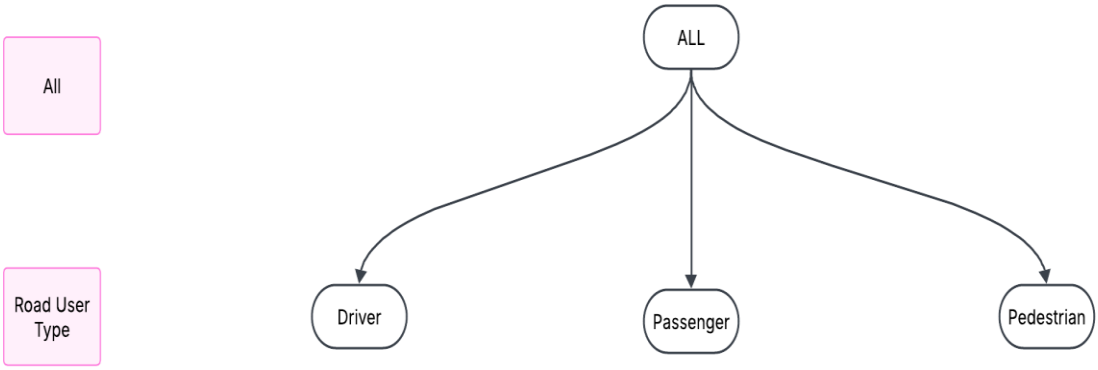
This dimension lets us:

- Isolate crashes that occurred during festive seasons
- Design campaign timings and travel advisories accordingly



7. Road User Dimension

Hierarchy: All → Road User Type (Driver, Passenger, Pedestrian, etc.)



There is no hierarchy as they organised into categories which includes: driver, passenger, pedestrian, motorcycle rider. It categorizes the type of road user involved in each fatal crash. Although it typically consists of a single-level categorization, it plays a crucial role in segmenting the data to understand which groups are most at risk. If further sub-categories are required (e.g., splitting "Motorcycle Rider" into "Rider" and "Pillion Passenger"), a hierarchy could be introduced. For now, this dimension supports simple but effective segmentation for comparative analysis.

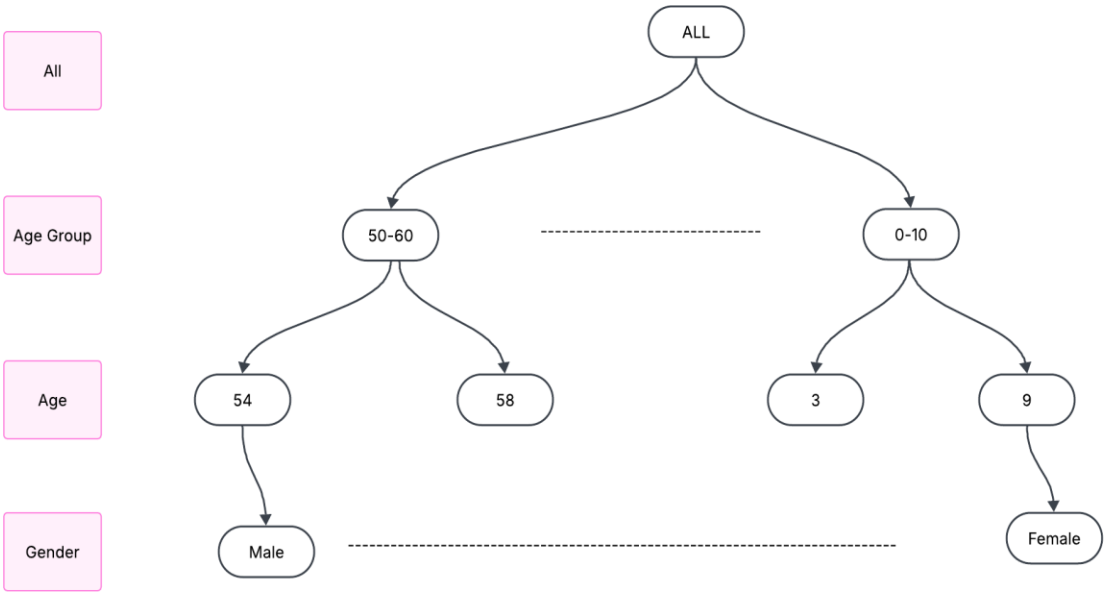
8. **Victim Dimension**

Hierarchy: All → Age Group → Age

Additional attribute: Gender

The Victim dimension focuses on the demographic characteristics of the individuals involved in crashes. It captures the exact age of the victim and then aggregates these into broader age groups **(e.g., 0–16, 17–25, 26–39, etc.).** This dual-level approach allows for both detailed and summarized demographic analysis. Gender is also included as an attribute, which supports gender-specific studies. Understanding these demographics is essential for crafting targeted interventions and understanding how risk varies across different population segments. Use cases:

- Identifying age groups with the highest death rates
- Examining gender-specific patterns
- Supporting targeted campaigns (e.g., for young male drivers)



Fact Tables

The warehouse includes two core fact tables: **FactFatalCrashes** and **FactFatalities**, and several supporting dimension tables. These tables are linked via surrogate keys to ensure referential integrity and to enable slicing, dicing, roll-up, and drill-down operations across various dimensions.

- **FactFatalCrashes:**

  Grain: One row per crash with measures like the number of fatalities and a crash count**.**

  Usage: Supports aggregated, crash-level trend analysis.

- **FactFatalities:**

  Grain: One row per fatality, linking occupant-level details (victim demographics, road user type) to each crash.

  Usage: Enables detailed, occupant-level insights.

5. Data Warehouse Design

After detailed dataset cleaning, transformation and merging, we designed a galaxy schema data warehouse which was implemented in PostgresSQL. The process involved splitting the cleaned merged dataset into two fact tables and eight-dimension tables, each optimized to answer the business questions we had in mind. Although this design follows the principles of dimensional modelling with star schemas at the table level, the overall structure forms a **galaxy schema** (fact constellation), as it includes multiple fact tables — **FactFatalCrashes** and **FactFatalities** — that share common dimension tables. This design supports multidimensional analysis while maintaining clarity and performance.

Reasons why we used galaxy schema:

Our data warehouse is implemented using a galaxy schema (also known as a fact constellation) which consists of two fact tables and eight dedicated dimensions. This design choice is justified as follows:

- **Multiple Fact Tables for Different Levels of Analysis:**
    - **FactFatalCrashes**: Contains one row per crash and includes measures such as the number of fatalities and a crash count indicator. This table provides a high-level, aggregate view of crash events.
    - **FactFatalities**: Contains one row per fatality and links each fatality to detailed victim and crash information. This table supports granular, occupant-level analysis.
    - **Justification:** By having two fact tables, our warehouse can simultaneously support overall crash trend analysis and detailed analysis of individual fatalities. This flexibility is critical for answering a wide range of business queries.
- **Eight Dimensions to Support Multidimensional Analysis:**
    - **DateDimension**: Captures the temporal aspects (year, month, day of week) to analyze crash trends over time.
    - **TimeDimension**: Provides time-of-day details, allowing analysis of peak hours and day vs. night patterns.

- **LocationDimension**: Contains hierarchical geographic data (LGA, SA4, state) along with dwelling counts, enabling location-specific risk analysis and normalization (e.g., fatalities per dwelling).
- **CrashDimension**: Differentiates crash conditions (e.g., crash type, speed limit, day/weekend), helping to evaluate the severity of crashes.
- **VehicleDimension**: Records vehicle involvement (bus, heavy rigid truck, articulated truck) to analyze how specific vehicle types impact crash outcomes.
- **RoadUserDimension**: Identifies the roles of individuals involved in crashes (e.g., driver, passenger), facilitating occupant-level segmentation.
- **VictimDimension**: Includes demographic details (gender, age, age group) to support targeted road safety interventions.
- **PeriodDimension**: Flags holiday periods (Christmas, Easter) to focus on time-specific trends.
- **Justification:** These dimensions encapsulate every key aspect required for our analysis. They allow us to perform drill-down analyses and support complex queries that span time, location, crash conditions, vehicle involvement, and victim demographics.

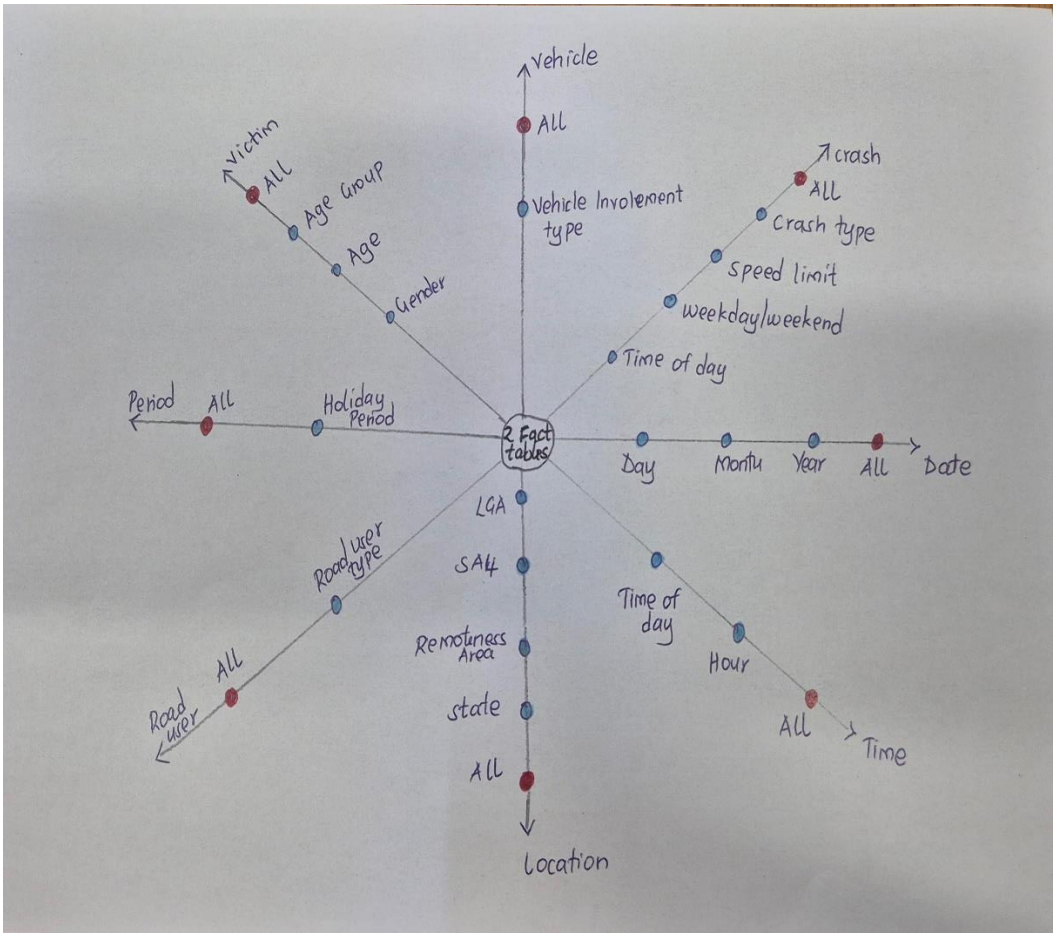
- **Alignment with Business Requirements:**
  - The design supports queries such as:
    - Analyzing holiday-period fatalities by time of day, crash type, and vehicle involvement.
    - Segmenting nighttime crashes by victim age group and road user type in specific LGAs.
    - Tracking yearly trends in single vehicle versus multi-vehicle crashes across states and periods.
    - Investigating correlations between peak crash hours, speed limits, and victim demographics.
    - Normalizing fatality counts per dwelling in remote LGAs.
  - **Justification:** By directly mapping these business needs to our dimensions and fact tables, our galaxy schema provides the analytical flexibility and granularity required to answer stakeholder queries effectively.

- **Concept Hierarchies and Scalability:**
    - Each dimension is designed with a clear concept hierarchy (e.g., DateDimension with Year → Month → Day of Week, LocationDimension with LGA → SA4 → State), which facilitates intuitive roll-up and drill-down analyses.
    - The design is scalable; new dimensions or measures can be added as additional business requirements emerge.
    - **Justification:** The inherent hierarchies support aggregation and slicing at multiple levels, ensuring that our warehouse remains adaptable to evolving analytical needs.
- **Industry Best Practices:**
    - Our design follows dimensional modeling principles as described in Kimball & Ross's The Data Warehouse Toolkit [2].
    - A galaxy schema is a proven solution when multiple fact tables share common dimensions, enabling comprehensive analysis without redundant data storage.
    - **Justification:** This approach not only meets stakeholder requirements but also leverages established best practices to ensure high performance and ease of use in analytical environments.

6. Starnet

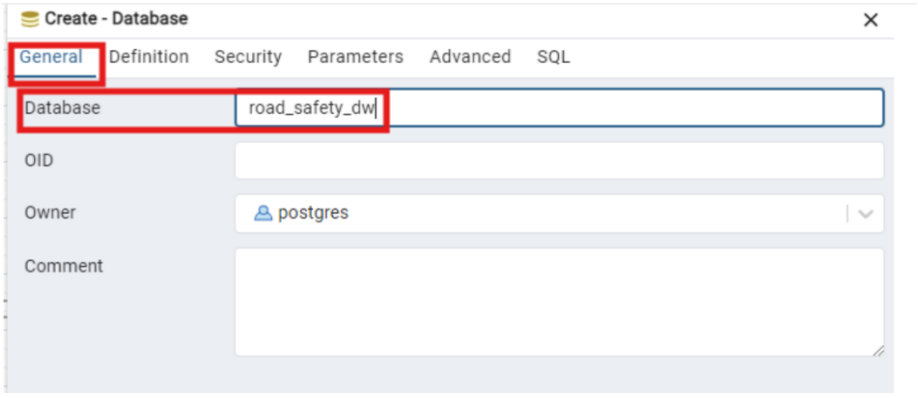
All the concept hierarchies are then combined and built a multidimensional StarNet for our data warehouse, which allows us to understand what combinations of dimensions we have in our data warehouse. By laying out the concept hierarchies in this manner, we provided a clear guide to the multidimensional analysis capabilities ensuring that our data warehouse is equipped to meet the needs of the stakeholders.

The StarNet diagram of our data warehouse is drawn below:



Creating dimension tables and fact tables in Postgres

We created our database after putting the server details in (local) pgAdmin.



We then split the merged final data frame into 8 dimensions and two fact tables by using the code provided below in the screenshots:

```python
import pandas as pd
import numpy as np
import psycopg2
from psycopg2.extras import execute_values

def build_star_schema(df):
    """
    Splits the final DataFrame into 8 dimension DataFrames + 2 fact DataFrames,
    renaming columns to remove spaces and converting Time to string.
    """

    # Rename columns
    df_renamed = df.copy()
    df_renamed = df_renamed.rename(columns={
        "Day Of Week": "day_of_week",
        "Time of Day": "time_of_day",
        "Crash Type": "crash_type",
        "Speed Limit": "speed_limit",
        "National Remoteness Areas": "remoteness_area",
        "SA4 Name 2021": "sa4_name",
        "National LGA Name 2021": "lga_name",
        "National Road Type": "road_type",
        "Bus Involvement": "bus_involvement",
        "Heavy Rigid Truck Involvement": "heavy_rigid_involvement",
        "Articulated Truck Involvement": "articulated_involvement",
        "Christmas Period": "christmas_period",
        "Easter Period": "easter_period",
        "Road User": "road_user",
        "Age Group": "age_group",
        "Dayweek": "weekday_weekend",
        "Month": "month",
        "Year": "year",
        "Gender": "gender",
        "State": "state",
        "Age": "age",
        "Crash ID": "crash_id"
    })

    # Convert Time to string
    if 'Time' in df_renamed.columns:
        df_renamed['time_str'] = df_renamed['Time'].astype(str)
    else:
        df_renamed['time_str'] = np.nan

    # DateDimension
    date_dim = (
        df_renamed[['year','month','day_of_week']]
        .drop_duplicates()
        .reset_index(drop=True)
        .copy()
    )
    date_dim['date_key'] = date_dim.index + 1
    date_dim = date_dim[['date_key','year','month','day_of_week']]

    # TimeDimension
    time_dim = (
```

```
    .drop_duplicates()
    .reset_index(drop=True)
    .copy()
)
location_dim['location_key'] = location_dim.index + 1
location_dim = location_dim[['location_key','lga_name','sa4_name','remoteness_area','state','dwelling_count']]

# CrashDimension
crash_dim = (
    df_renamed[['crash_type','speed_limit','weekday_weekend','time_of_day']]
    .drop_duplicates()
    .reset_index(drop=True)
    .copy()
)
crash_dim['crash_key'] = crash_dim.index + 1
crash_dim = crash_dim[['crash_key','crash_type','speed_limit','weekday_weekend','time_of_day']]

# VehicleDimension
vehicle_dim = (
    df_renamed[['bus_involvement','heavy_rigid_involvement','articulated_involvement']]
    .drop_duplicates()
    .reset_index(drop=True)
    .copy()
)
vehicle_dim['vehicle_key'] = vehicle_dim.index + 1
vehicle_dim = vehicle_dim[['vehicle_key','bus_involvement','heavy_rigid_involvement','articulated_involvement']]

# RoadUserDimension
road_user_dim = (
    df_renamed[['road_user']]
    .drop_duplicates()
    .reset_index(drop=True)
    .copy()
)
road_user_dim['road_user_key'] = road_user_dim.index + 1
road_user_dim = road_user_dim[['road_user_key','road_user']]

# VictimDimension
victim_dim = (
    df_renamed[['gender','age','age_group']]
    .drop_duplicates()
    .reset_index(drop=True)
    .copy()
)
victim_dim['victim_key'] = victim_dim.index + 1
victim_dim = victim_dim[['victim_key','gender','age','age_group']]

# PeriodDimension
period_dim = (
    df_renamed[['christmas_period','easter_period']]
    .drop_duplicates()
    .reset_index(drop=True)
    .copy()
)
period_dim['period_key'] = period_dim.index + 1
period_dim = period_dim[['period_key','christmas_period','easter_period']]
```

```
# Dictionaries
date_dict = date_dim.set_index(['year','month','day_of_week'])['date_key'].to_dict()
time_dict = time_dim.set_index(['time_str','time_of_day'])['time_key'].to_dict()
loc_dict = location_dim.set_index(['lga_name','sa4_name','remoteness_area','state','dwelling_count'])['location_key'].to_dict()
crash_dict = crash_dim.set_index(['crash_type','speed_limit','weekday_weekend','time_of_day'])['crash_key'].to_dict()
veh_dict = vehicle_dim.set_index(['bus_involvement','heavy_rigid_involvement','articulated_involvement'])['vehicle_key'].to_dict()
road_user_dict = road_user_dim.set_index('road_user')['road_user_key'].to_dict()
victim_dict = victim_dim.set_index(['gender','age','age_group'])['victim_key'].to_dict()
period_dict = period_dim.set_index(['christmas_period','easter_period'])['period_key'].to_dict()

# FactFatalCrashes
fact_crashes = df_renamed.drop_duplicates(subset=['crash_id']).copy()
fact_crashes['fact_crash_id'] = fact_crashes.index + 1
fact_crashes['date_key'] = fact_crashes[['year','month','day_of_week']].apply(
    lambda row: date_dict.get((row[0], row[1], row[2])), axis=1
)
fact_crashes['time_key'] = fact_crashes[['time_str','time_of_day']].apply(
    lambda row: time_dict.get((row[0], row[1])), axis=1
)
fact_crashes['location_key'] = fact_crashes[['lga_name','sa4_name','remoteness_area','state','dwelling_count']].apply(
    lambda row: loc_dict.get(tuple(row)), axis=1
)
fact_crashes['crash_key'] = fact_crashes[['crash_type','speed_limit','weekday_weekend','time_of_day']].apply(
    lambda row: crash_dict.get(tuple(row)), axis=1
)
fact_crashes['vehicle_key'] = fact_crashes[['bus_involvement','heavy_rigid_involvement','articulated_involvement']].apply(
    lambda row: veh_dict.get(tuple(row)), axis=1
)
fact_crashes['period_key'] = fact_crashes[['christmas_period','easter_period']].apply(
    lambda row: period_dict.get((row[0], row[1])), axis=1
)
fact_crashes['number_fatalities'] = fact_crashes['Number Fatalities'].astype(int)
fact_crashes['crash_count'] = 1
fact_crashes = fact_crashes[[
    'fact_crash_id','crash_id','date_key','time_key','location_key',
    'crash_key','vehicle_key','period_key','number_fatalities','crash_count'
]]

# FactFatalities
fact_fatalities = df_renamed.copy()
fact_fatalities['fact_fatality_id'] = fact_fatalities.index + 1
fact_fatalities['date_key'] = fact_fatalities[['year','month','day_of_week']].apply(
    lambda row: date_dict.get((row[0], row[1], row[2])), axis=1
)
fact_fatalities['time_key'] = fact_fatalities[['time_str','time_of_day']].apply(
    lambda row: time_dict.get((row[0], row[1])), axis=1
)
fact_fatalities['location_key'] = fact_fatalities[['lga_name','sa4_name','remoteness_area','state','dwelling_count']].apply(
    lambda row: loc_dict.get(tuple(row)), axis=1
)
fact_fatalities['crash_key'] = fact_fatalities[['crash_type','speed_limit','weekday_weekend','time_of_day']].apply(
    lambda row: crash_dict.get(tuple(row)), axis=1
)
fact_fatalities['vehicle_key'] = fact_fatalities[['bus_involvement','heavy_rigid_involvement','articulated_involvement']].apply(
    lambda row: veh_dict.get(tuple(row)), axis=1
)
fact_fatalities['road_user_key'] = fact_fatalities[['road_user']].apply(
```

```
)
fact_fatalities['victim_key'] = fact_fatalities[['gender','age','age_group']].apply(
    lambda row: victim_dict.get(tuple(row)), axis=1
)
fact_fatalities['period_key'] = fact_fatalities[['christmas_period','easter_period']].apply(
    lambda row: period_dict.get((row[0], row[1])), axis=1
)
fact_fatalities['fatality_count'] = 1
fact_fatalities = fact_fatalities[[
    'fact_fatality_id','crash_id','date_key','time_key','location_key',
    'crash_key','vehicle_key','road_user_key','victim_key','period_key',
    'fatality_count'
]]

return {
    'DateDimension': date_dim,
    'TimeDimension': time_dim,
    'LocationDimension': location_dim,
    'CrashDimension': crash_dim,
    'VehicleDimension': vehicle_dim,
    'RoadUserDimension': road_user_dim,
    'VictimDimension': victim_dim,
    'PeriodDimension': period_dim,
    'FactFatalCrashes': fact_crashes,
    'FactFatalities': fact_fatalities
}
```

The next step was creating tables in Postgres as shown by the code provided below:

```
import psycopg2
from psycopg2.extras import execute_values

def create_star_schema_postgres():
    ddl_statements = [
        """
        CREATE TABLE IF NOT EXISTS DateDimension (
            date_key SERIAL PRIMARY KEY,
            year INT,
            month INT,
            day_of_week VARCHAR(10)
        );
        """,
        """
        CREATE TABLE IF NOT EXISTS TimeDimension (
            time_key SERIAL PRIMARY KEY,
            time_str VARCHAR(10),
            time_of_day VARCHAR(10)
        );
        """,
        """
        CREATE TABLE IF NOT EXISTS LocationDimension (
            location_key SERIAL PRIMARY KEY,
            lga_name VARCHAR(100),
            sa4_name VARCHAR(100),
            remoteness_area VARCHAR(50),
            state VARCHAR(10),
            dwelling_count INT
        );
        """,
        """
        CREATE TABLE IF NOT EXISTS CrashDimension (
            crash_key SERIAL PRIMARY KEY,
            crash_type VARCHAR(20),
            speed_limit VARCHAR(10),
            weekday_weekend VARCHAR(10),
            time_of_day VARCHAR(10)
        );
        """,
        """
        CREATE TABLE IF NOT EXISTS VehicleDimension (
            vehicle_key SERIAL PRIMARY KEY,
            bus_involvement VARCHAR(5),
            heavy_rigid_involvement VARCHAR(5),
            articulated_involvement VARCHAR(5)
        );
        """,
        """
        CREATE TABLE IF NOT EXISTS RoadUserDimension (
            road_user_key SERIAL PRIMARY KEY,
            road_user VARCHAR(30)
        );
        """,
        """
        CREATE TABLE IF NOT EXISTS VictimDimension (
            victim_key SERIAL PRIMARY KEY,
            gender VARCHAR(10),
```

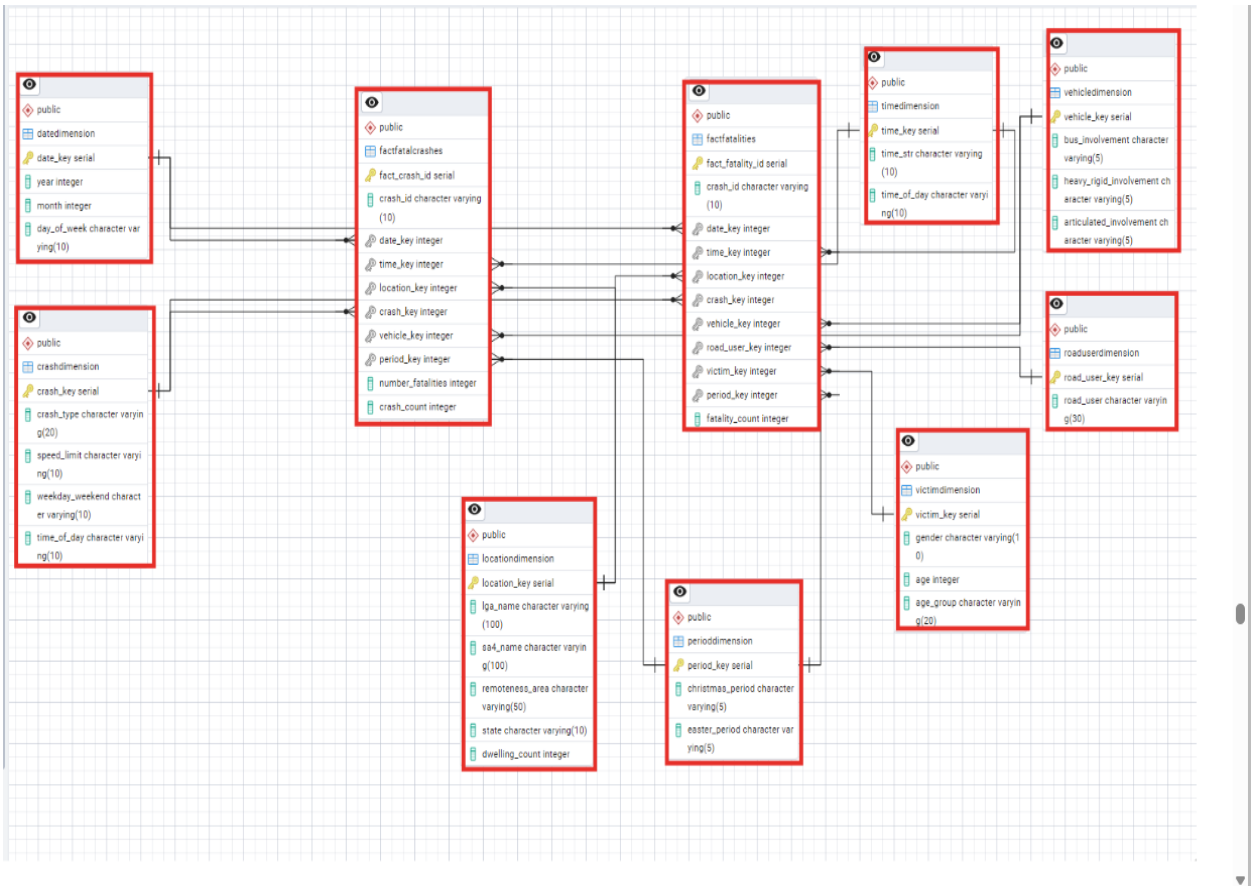
```
            gender VARCHAR(10),
            age INT,
            age_group VARCHAR(20)
        );
        """,
        """
        CREATE TABLE IF NOT EXISTS PeriodDimension (
            period_key SERIAL PRIMARY KEY,
            christmas_period VARCHAR(5),
            easter_period VARCHAR(5)
        );
        """,
        """
        CREATE TABLE IF NOT EXISTS FactFatalCrashes (
            fact_crash_id SERIAL PRIMARY KEY,
            crash_id VARCHAR(10),
            date_key INT REFERENCES DateDimension(date_key),
            time_key INT REFERENCES TimeDimension(time_key),
            location_key INT REFERENCES LocationDimension(location_key),
            crash_key INT REFERENCES CrashDimension(crash_key),
            vehicle_key INT REFERENCES VehicleDimension(vehicle_key),
            period_key INT REFERENCES PeriodDimension(period_key),
            number_fatalities INT,
            crash_count INT
        );
        """,
        """
        CREATE TABLE IF NOT EXISTS FactFatalities (
            fact_fatality_id SERIAL PRIMARY KEY,
            crash_id VARCHAR(10),
            date_key INT REFERENCES DateDimension(date_key),
            time_key INT REFERENCES TimeDimension(time_key),
            location_key INT REFERENCES LocationDimension(location_key),
            crash_key INT REFERENCES CrashDimension(crash_key),
            vehicle_key INT REFERENCES VehicleDimension(vehicle_key),
            road_user_key INT REFERENCES RoadUserDimension(road_user_key),
            victim_key INT REFERENCES VictimDimension(victim_key),
            period_key INT REFERENCES PeriodDimension(period_key),
            fatality_count INT
        );
        """
    ]
    conn = psycopg2.connect(
        dbname="road_safety_dw",
        user="postgres",
        password="Tenda@715",
        host="localhost",
        port="5432"
    )
    cur = conn.cursor()
    for ddl in ddl_statements:
        cur.execute(ddl)
    conn.commit()
    cur.close()
    conn.close()

def insert_dimension_table(conn, table_name, df, columns):
    records = df[columns].values.tolist()
```

```
        insert_stmt = f"INSERT INTO {table_name} ({','.join(columns)}) VALUES %s"
        execute_values(cur, insert_stmt, records)
    conn.commit()

def load_star_schema_to_postgres(star_dict):
    create_star_schema_postgres()
    conn = psycopg2.connect(
        dbname="road_safety_dw",
        user="postgres",
        password="Tenda@715",
        host="localhost",
        port="5432"
    )
    # Dimension Insert
    insert_dimension_table(conn, "DateDimension",
        star_dict['DateDimension'],
        ["date_key","year","month","day_of_week"]
    )
    insert_dimension_table(conn, "TimeDimension",
        star_dict['TimeDimension'],
        ["time_key","time_str","time_of_day"]
    )
    insert_dimension_table(conn, "LocationDimension",
        star_dict['LocationDimension'],
        ["location_key","lga_name","sa4_name","remoteness_area","state","dwelling_count"]
    )
    insert_dimension_table(conn, "CrashDimension",
        star_dict['CrashDimension'],
        ["crash_key","crash_type","speed_limit","weekday_weekend","time_of_day"]
    )
    insert_dimension_table(conn, "VehicleDimension",
        star_dict['VehicleDimension'],
        ["vehicle_key","bus_involvement","heavy_rigid_involvement","articulated_involvement"]
    )
    insert_dimension_table(conn, "RoadUserDimension",
        star_dict['RoadUserDimension'],
        ["road_user_key","road_user"]
    )
    insert_dimension_table(conn, "VictimDimension",
        star_dict['VictimDimension'],
        ["victim_key","gender","age","age_group"]
    )
    insert_dimension_table(conn, "PeriodDimension",
        star_dict['PeriodDimension'],
        ["period_key","christmas_period","easter_period"]
    )

    # Fact Insert
    insert_fact_table(conn, "FactFatalCrashes",
        star_dict['FactFatalCrashes'],
        ["fact_crash_id","crash_id","date_key","time_key","location_key",
         "crash_key","vehicle_key","period_key","number_fatalities","crash_count"]
    )
    insert_fact_table(conn, "FactFatalities",
        star_dict['FactFatalities'],
        ["fact_fatality_id","crash_id","date_key","time_key","location_key",
         "crash_key","vehicle_key","road_user_key","victim_key","period_key","fatality_count"]
```

Execution

```
star_dict = build_star_schema(df_merged_final)
load_star_schema_to_postgres(star_dict)
```

After all the tables were created, we generated a galaxy schema for the fact tables and dimensions in pgAdmin and got the ERD image below.



The ERD galaxy schema above illustrates the structure of our data warehouse design. It consists of two fact tables surrounded by eight dimensions. From the image we can see that the fact tables share all the dimensions except for vehicle dimension and road user dimension.

Business Questions that we aim to answer:

Structure of the questions

In this project, we chose to present the core business insights as **w**ell-defined analytical statements rather than framing them as open-ended business questions. This approach was taken intentionally for the following reasons:

1. **Clarity of Purpose**

   By clearly stating the purpose of each insight, stakeholders can immediately understand

what the analysis seeks to uncover, without needing to infer the goal from a question. This creates a stronger link between the data and its real-world application.

2. **Alignment with BI Practices**

   In professional business intelligence reporting, insights are typically framed as actionable findings or analyses, rather than as exploratory questions. This reflects how results are delivered in dashboards, executive summaries, and policy briefs — focusing on "what the data shows" rather than "what we are asking."

3. **Consistence with Dimensional Modeling**

   Each insight was derived based on the structure of the data warehouse and the available dimensions. Presenting them as structured analytical statements helps demonstrate **h**ow the warehouse design enables these insights, strengthening the traceability from schema to analysis.

4. **Depth and Readability**

   Expressing insights as statements allows for more precise articulation of the scope, filters, and analytical focus of each query — such as time of day, crash type, or geographic segmentation — without oversimplifying the analysis into a one-line question.

Business Questions and their alignment with the warehouse design:

1. **Holiday-Period Fatalities by Time of Day, Crash Type, and Vehicle Involvement**

Category: Holiday Impact Analysis; Crash Condition Analysis; Temporal Analysis

Purpose:

Stakeholders can retrieve fatal crash data during Christmas and Easter, drilled down by time of day (morning vs. night), crash type, and vehicle involvement.

Alignment:

This query leverages DateDimension, TimeDimension, CrashDimension, and VehicleDimension to offer detailed holiday-period analysis.

Questions:

This query helps answer questions such as:

a. Do fatality counts significantly differ between holidays (Christmas/Easter) and non-holiday periods?

b. Is there a difference between day and night crashes during these periods?

c. How do vehicle involvement types (bus, heavy rigid, articulated) and crash type (single vs. multiple) impact fatality counts?

2. **Age Group vs. Road User Type for Nighttime Crashes in Specific LGAs**

Category: Victim Demographics Analysis; Location-Specific Risk Analysis; Temporal Analysis

Purpose:

The warehouse supports analysis of crash data segmented by victim age group and road user type during nighttime crashes in high-fatality LGAs.

Alignment:

By using VictimDimension, RoadUserDimension, LocationDimension, and TimeDimension, this query meets the need for demographic and location-specific insights.

Questions:

This query helps answer questions such as:

- Which age groups are most affected during nighttime crashes?
- Are vulnerable road user types (e.g., pedestrians, motorcycle riders) more common in certain LGAs?

3. **Yearly Trend of Single-Vehicle vs. Multi-Vehicle Fatal Crashes, Filtered by State and Period**

Category: Temporal Analysis; Crash Condition Differentiation

Purpose:

This query compares trends in single vehicle versus multi-vehicle fatal crashes over the years, with filtering by state and holiday period.

Alignment:

It uses DateDimension for temporal trends, CrashDimension to differentiate crash types, and PeriodDimension for holiday filters, providing longitudinal insights.

Questions:

This query helps answer questions such as:

- How fatal crash trends (single vs. multiple vehicle) have evolved over the years across states.
- The impact of holiday periods on these trends

4. **Peak Time of Day and Speed Limit Correlation with Victim Demographics**

- **Category:** Temporal Analysis; Crash Condition Analysis; Victim Demographics Analysis
- **Purpose:**
  The query examines which hours and speed limit ranges correlate with higher fatality rates among specific victim demographics (e.g., older age groups).
- **Alignment:**
  TimeDimension and CrashDimension (for speed limits) combine with VictimDimension to reveal high-risk conditions by time and speed.
- **Questions:**
  This query helps answer questions such as:
  - Whether certain speed limits are linked to higher fatality counts during peak time of Day.
  - How victim demographics (age group, gender) vary with speed limits and time of day.

5. **Fatalities Per Dwelling in Remote LGAs, Split by Day vs. Night, Road User, and Holiday Period**

- **Category:** Location-Specific Risk Analysis; Holiday Impact Analysis

- **Purpose:**

  This query calculates fatalities per dwelling in remote regions, broken down by day vs. night, road user type, and holiday periods.

- **Alignment:**

  The LocationDimension (which includes dwelling_count) along with TimeDimension, RoadUserDimension, and PeriodDimension enables a normalized analysis of crash risk in remote areas.

- **Questions:**

  This query helps answer questions such as:

  - How fatality rates per dwelling differ in remote areas when segmented by time of day, road user, and holiday periods.
  - Which factors (time of day, road user type) contribute most to elevated fatality rates in remote regions.

7. Starnet footprints for each of the queries

Below is a table highlighting our questions that we aim to answer each showing which fact table that we will use together with why the fact table.

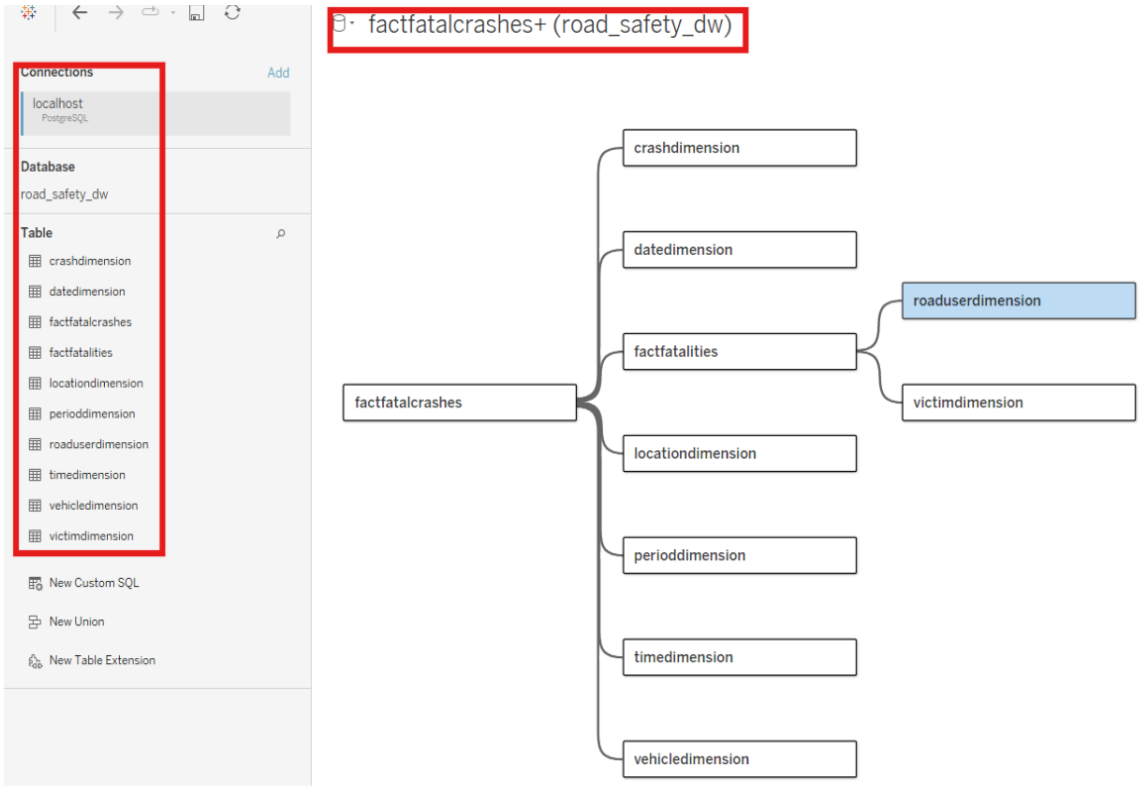
Insight No.	Insight Description	Fact Table Used	Reason
1	Holiday-Period Fatalities by Time of Day, Crash Type, and Vehicle Involvement	FactFatalCrashes	Focuses on crash-level data such as vehicle type, crash type, and timing.
2	Age Group vs. Road User Type for Nighttime Crashes in Specific LGAs	FactFatalities	Analyzes victim-level details like age group and road user roles.

3	Yearly Trend of Single-Vehicle vs. Multi-Vehicle Fatal Crashes	FactFatalCrashes	Compares overall crash events by type and year, not individual victims.
4	Peak Time of Day and Speed Limit Correlation with Victim Demographics	FactFatalities	Involves demographic analysis, requiring individual fatality information.
5	Fatalities Per Dwelling in Remote LGAs	FactFatalities	Calculates per-capita fatalities, which requires victim-level granularity.

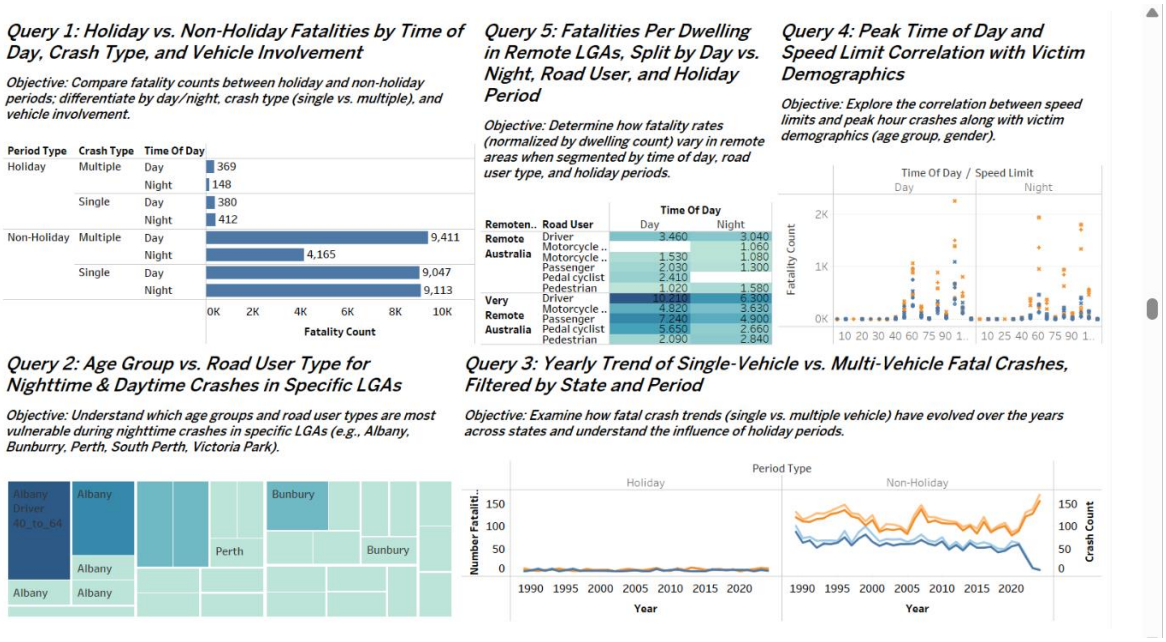
Tableau Visualization Process

After preparing and loading the cleaned data into our PostgreSQL data warehouse, we connected **Tableau** directly to the database to build interactive dashboards and charts. The screenshot (shown below) is a **Tableau Data Source page**, which illustrates how each table (facts and dimensions) is laid out and how Tableau recognizes their relationships:

1. **Establishing the Connection**
    - We used Tableau's **PostgreSQL connector** to log into our road_safety_dw database.
    - Tableau automatically detected the dimension tables (e.g., crashdimension, locationdimension, etc.) and fact tables (factfatalcrashes, factfatalities).
2. **Defining Relationships**
    - On Tableau's Data Source page, we dragged each table into the workspace.
    - We confirmed the **join fields** (primary keys in the dimension tables, foreign keys in the fact tables) to reflect our dimensional schema.
    - This gave Tableau a clear, star-like structure, letting it seamlessly query each dimension alongside the fact tables.



3. **Creating Worksheets and Dashboards**
    - Once the data source was set, we moved to **Sheet** view.
    - Tableau allowed us to easily select dimensions (like **State**, **Time of Day**, **Crash Type**) and measures (like **Number of Fatalities**, **Crash Counts**) to build a variety of visualizations.
    - By combining multiple sheets into a **Dashboard**, as shown below.



Postgres SQL queries and their StarNet diagrams along with tableau visualizations:

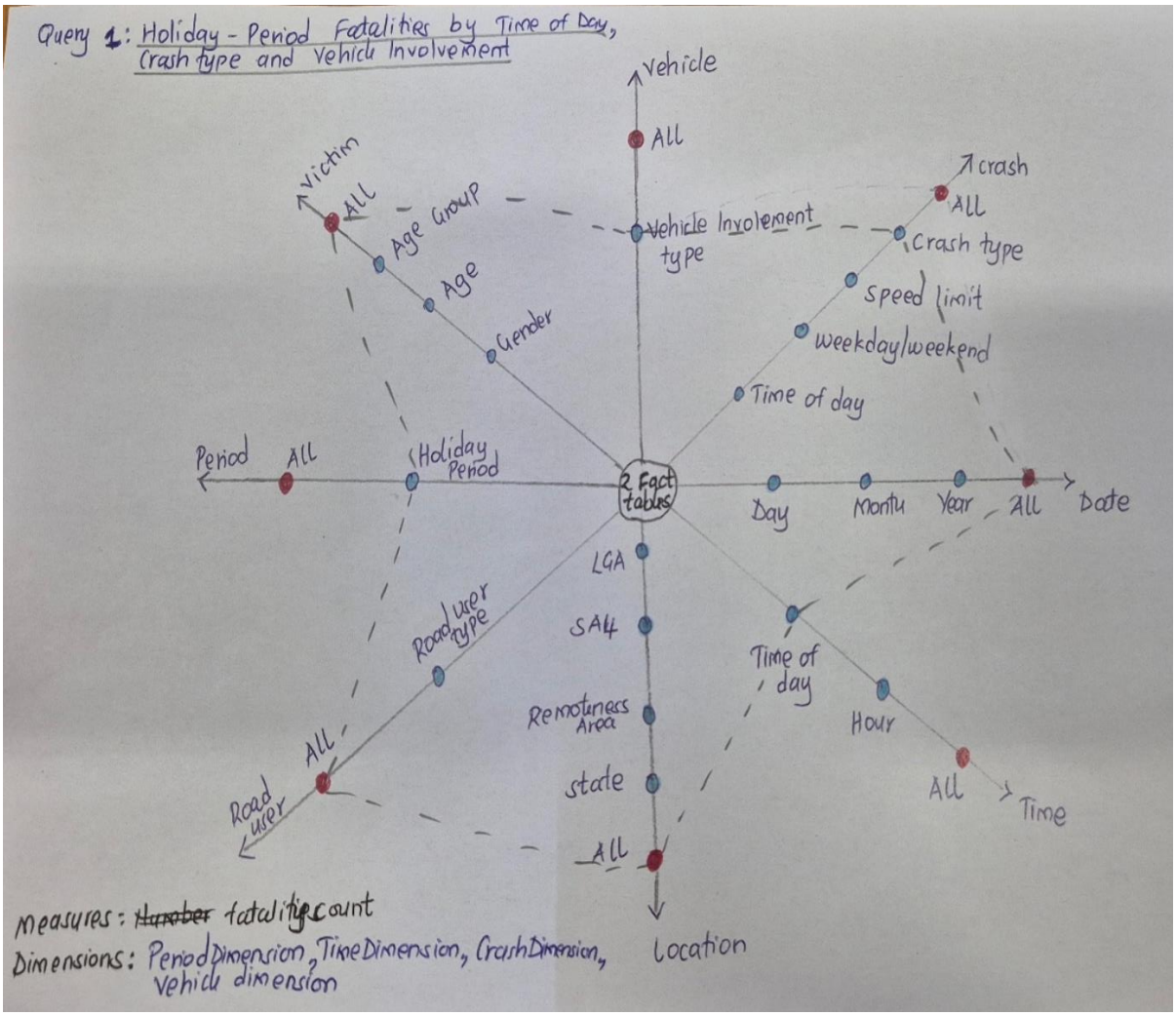
Query 1: Holiday-Period Fatalities by Time of Day, Crash Type, and Vehicle InvolvementHoliday-Period Fatalities by Time of Day, Crash Type, and Vehicle Involvement

```
SELECT
    p.christmas_period,
     p.easter_period,
     t.time_of_day,
     c.crash_type,
     v.bus_involvement,
     v.heavy_rigid_involvement,
     v.articulated_involvement,
     SUM(ff.fatality_count) AS total_fatalities
 FROM FactFatalities ff
 JOIN PeriodDimension p ON ff.period_key = p.period_key
 JOIN TimeDimension t ON ff.time_key = t.time_key
 JOIN CrashDimension c ON ff.crash_key = c.crash_key
```

```
JOIN VehicleDimension v ON ff.vehicle_key = v.vehicle_key
WHERE (p.christmas_period = 'Yes' OR p.easter_period = 'Yes')
GROUP BY CUBE (
    p.christmas_period,
    p.easter_period,
    t.time_of_day,
    c.crash_type,
    v.bus_involvement,
    v.heavy_rigid_involvement,
    v.articulated_involvement
)
ORDER BY total_fatalities DESC;
```

In our project, we leveraged OLAP (Online Analytical Processing) operations such as slicing, dicing, drill-down, and roll-up to analyze fatality data across multiple dimensions. To do this, we wrote SQL queries—one with a standard GROUP BY and another with GROUP BY CUBE—that enabled multidimensional insights. We began by applying **slicing** to isolate holiday-related data by selecting only records where either the Christmas or Easter period was marked as 'Yes'. We then **diced** the data by considering combinations of other dimensions, including time of day, crash type, and vehicle involvement (bus, heavy rigid truck, and articulated truck), allowing us to observe fatalities from multiple intersecting perspectives. The use of GROUP BY CUBE enabled both **drill-down** and **roll-up** operations. Drill-down allowed us to examine granular details, such as the number of fatalities involving buses at night during Easter. Conversely, roll-up summarized the data to higher levels—providing subtotals by individual dimensions and an overall total of fatalities across all combinations.

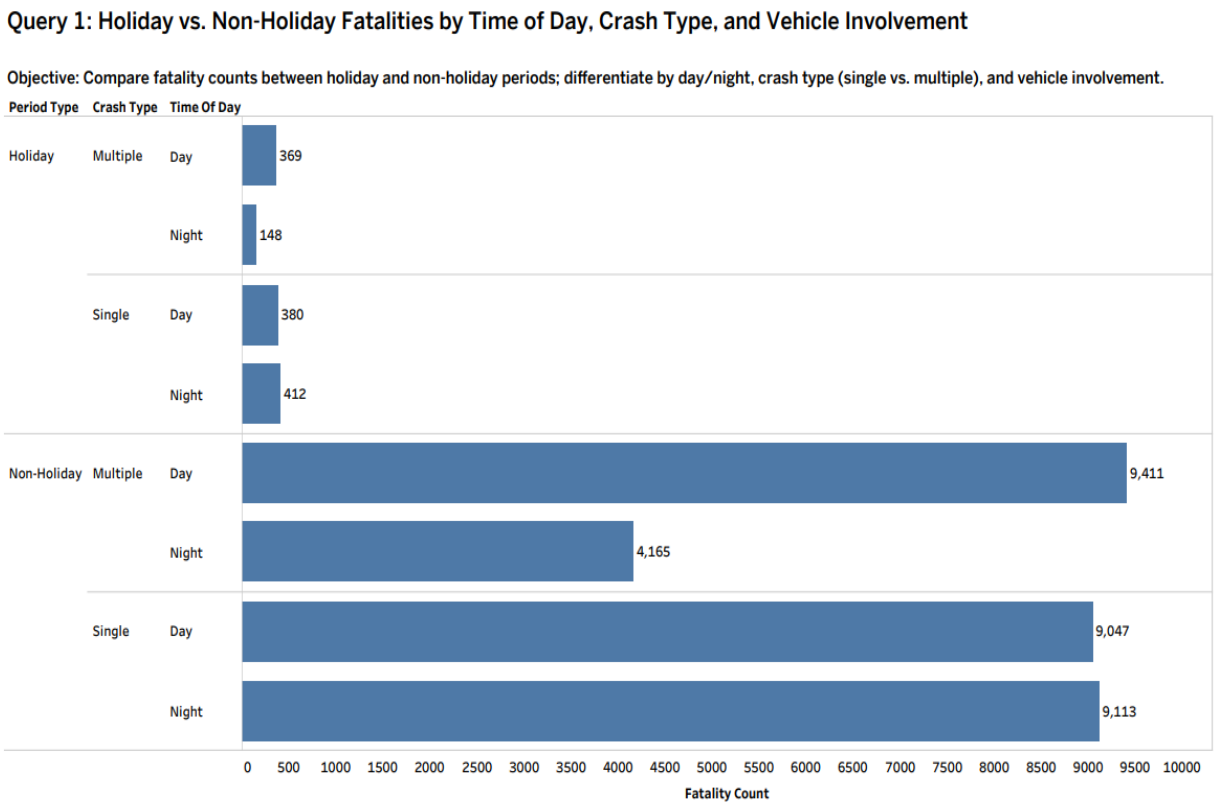
StarNet for query 1:



Output of the query 1:

```
Query 1 - CUBE result:
  christmas_period easter_period time_of_day crash_type bus_involvement  \
0             None          None        None       None             None
1             None          None        None       None               No
2             None          None        None       None             None
3             None          None        None       None               No
4              Yes          None        None       None             None

  heavy_rigid_involvement articulated_involvement  total_fatalities
0                    None                    None              1871
1                    None                    None              1845
2                    None                      No              1760
3                    None                      No              1745
4                    None                    None              1570
```

Tableau visualization for query 1:



Insights from Tableau Visualizations

Our visual analysis in Tableau revealed several important patterns in fatal crashes, especially when comparing holiday and non-holiday periods, types of crashes, time of day, and vehicle involvement.

1. Overall Fatality Volume: The data clearly shows that most fatal crashes occur on **non-holiday days**. During the day, fatalities reach over **9,000**, while at night, the numbers range from **4,000 to 9,100**. This is expected, as non-holiday periods span a larger portion of the year, leading to a higher overall count.

2. Holiday Crash Patterns: When we focused on holiday periods like Christmas and Easter, distinct patterns emerged:

- Multi-vehicle crashes during the day resulted in 369 fatalities, while nighttime incidents caused 148 fatalities.
- On the other hand, single-vehicle crashes were slightly more deadly at night (412 fatalities) compared to the day (380 fatalities). This suggests that daytime traffic congestion might contribute to multi-vehicle crashes, whereas nighttime fatigue, speeding, or impaired driving may drive the rise in single-vehicle accidents.

3. Vehicle Involvement – Articulated Vehicles: A separate breakdown of crashes involving **articulated vehicles** showed higher risks:

- For multi-vehicle crashes on non-holiday days, there were 1,359 fatalities during the day and 741 at night.
- For single-vehicle crashes, fatalities were 362 during the day and 336 at night. These numbers are significantly higher than for buses or heavy rigid trucks, highlighting articulated vehicles as particularly dangerous in multi-vehicle scenarios.

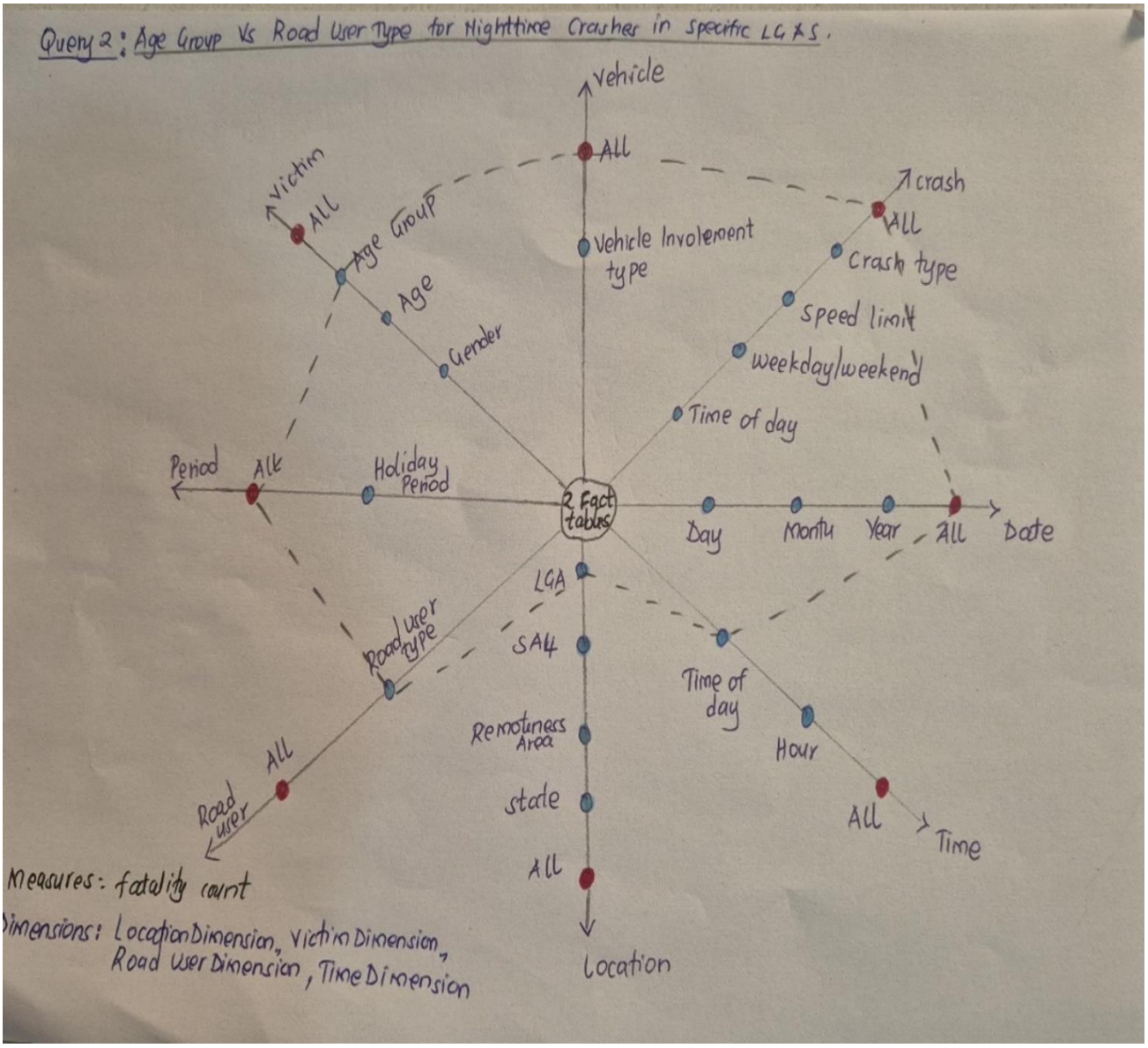
Query 2: Age Group vs. Road User Type for Nighttime Crashes in Specific LGAs

```
SELECT
    l.lga_name,
    v.age_group,
    ru.road_user,
    SUM(ff.fatality_count) AS total_fatalities
FROM FactFatalities ff
JOIN LocationDimension l ON ff.location_key = l.location_key
```

```
JOIN VictimDimension v ON ff.victim_key = v.victim_key
JOIN RoadUserDimension ru ON ff.road_user_key = ru.road_user_key
JOIN TimeDimension t ON ff.time_key = t.time_key
WHERE t.time_of_day = 'Night'
  AND l.lga_name IN ('Albany', 'Bunburry', 'South Perth','Perth','Victoria Park')
GROUP BY CUBE (
    l.lga_name,
    v.age_group,
    ru.road_user
)
ORDER BY total_fatalities DESC;
```

This query is designed to analyze fatality patterns during nighttime hours across three selected Local Government Areas (LGAs): **Albanny, Bunburry, South Perth, Victoria Park**. It focuses on Four main dimensions: location, victim,time and location dimensions.). By using the GROUP BY CUBE clause, the query allows for comprehensive OLAP operations—**slicing**, **dicing**, **roll-up**, and **drill-down**—to be performed on the data.

StarNet for query 2:



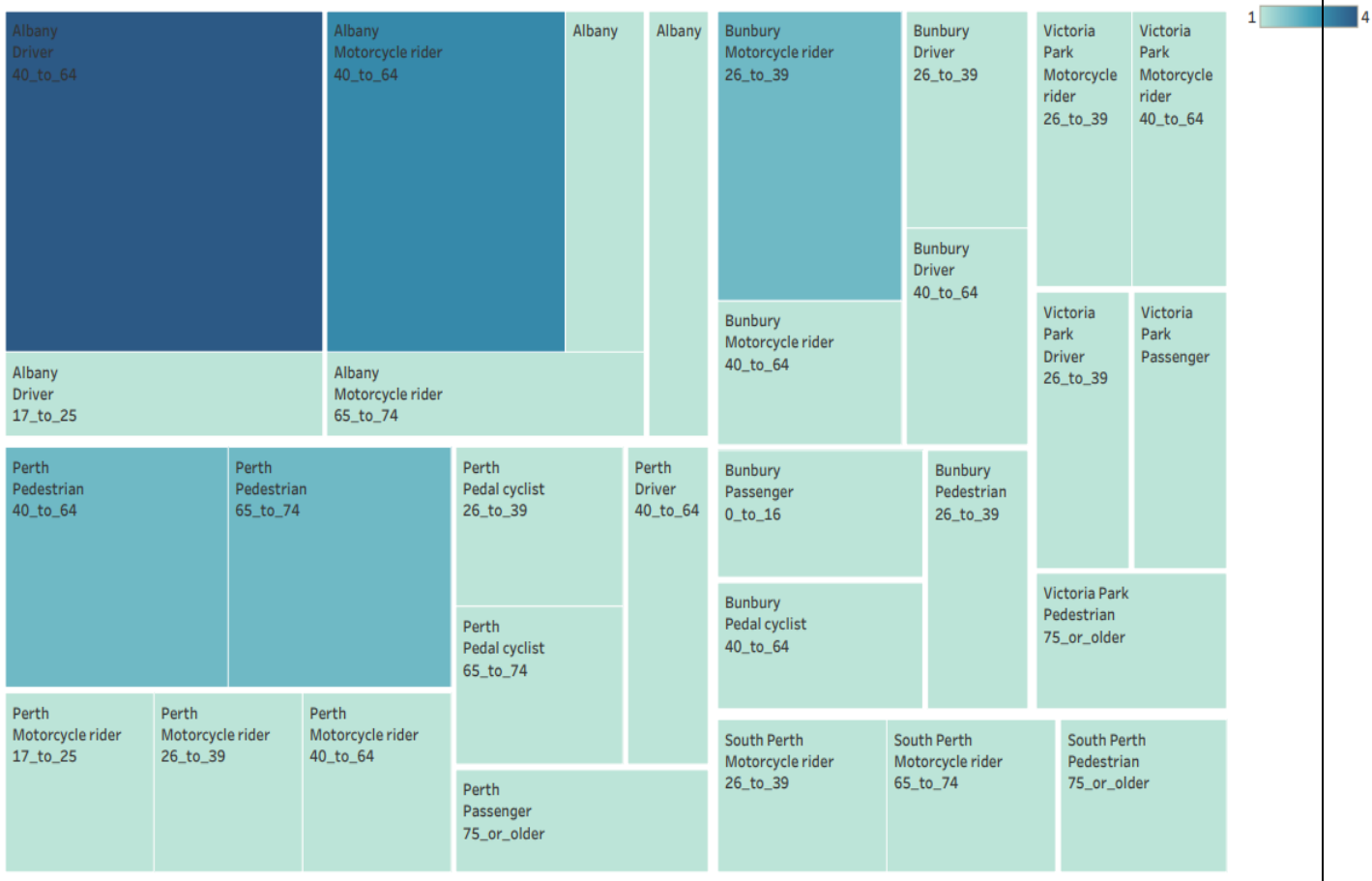
Output for query 2:

Query 2 - CUBE result:

	lga_name	age_group	road_user	total_fatalities
0	None	None	None	14
1	Perth	None	None	6
2	None	None	Motorcycle rider	6
3	None	40_to_64	None	5
4	None	26_to_39	None	5

Tableau visualizations for query 2:

Objective: Understand which age groups and road user types are most vulnerable during nighttime crashes in specific LGAs (e.g., Albany, Bunburry, Perth, South Perth, Victoria Park).



Lga Name, Road User and Age Group. Color shows sum of Fatality Count. Size shows sum of Fatality Count. The marks are labeled by Lga Name, Road User and Age Group. The data is filtered on Time Of Day (Timedimension) and Period Type. The Time Of Day (Timedimension) filter keeps Day and Night. The Period Type filter keeps Holiday and Non-Holiday. The view is filtered on Lga Name, which keeps Albany, Bunbury, Perth, South Perth and Victoria Park.

Key Insights from the visualization

1. **Younger vs. Middle-Aged Groups**
    a) **17_to_25 and 26_to_39** appear frequently as both drivers and motorcycle riders, particularly at night.
    b) **40_to_64** also shows up in multiple LGAs, indicating that middle-aged drivers remain at risk during both day and night.
2. **Motorcyclists and Drivers Lead in Fatalities**
    a) In several LGAs, the largest counts involve motorcycle riders or drivers rather than pedestrians or pedal cyclists.
    b) This suggests motorized vehicle users, especially younger adults, face greater risks after dark.
3. **Pedestrians and Cyclists Have Lower Recorded Incidents**
    a. While these categories are present (often as 1 or 2 occurrences), they appear less frequently than motorcycles and cars.
    b. This may indicate either safer infrastructure for pedestrians/cyclists or fewer of them on roads at night.
4. **Variations Across LGAs**
    a. Certain LGAs (e.g., Perth) show more uniform distributions across age groups, while others (e.g., Albany, Bunbury) highlight specific groups (such as motorcycle riders 17_to_25).
    b. Tailored local measures could address each area's unique crash patterns

Query 3: Yearly Trend of Single-Vehicle vs. Multi-Vehicle Fatal Crashes, Filtered by State and Period

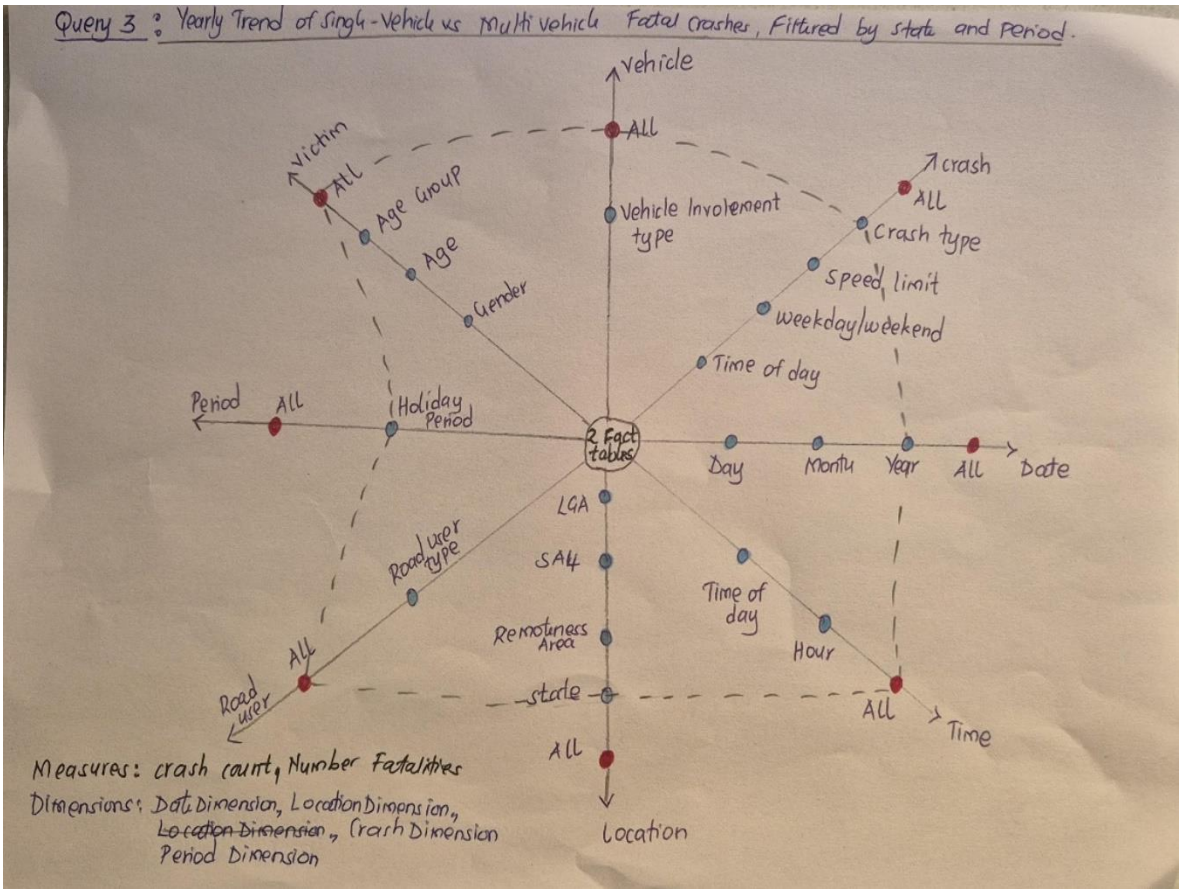
```
SELECT
    d.year,
```

```
    l.state,
    c.crash_type,
    p.christmas_period,
    p.easter_period,
    SUM(fc.number_fatalities) AS total_fatalities,
    SUM(fc.crash_count) AS total_crashes
FROM FactFatalCrashes fc
JOIN DateDimension d ON fc.date_key = d.date_key
JOIN LocationDimension l ON fc.location_key = l.location_key
JOIN CrashDimension c ON fc.crash_key = c.crash_key
JOIN PeriodDimension p ON fc.period_key = p.period_key
GROUP BY CUBE (
    d.year,
    l.state,
    c.crash_type,
    p.christmas_period,
    p.easter_period
)
ORDER BY d.year, total_fatalities DESC;
```

Through this approach, **slicing** occurs when the output is filtered by a specific value within one dimension—for example, it only results from the Christmas period. **Dicing** involves selecting a more detailed sub-section of the data, such as multi-vehicle crashes in New South Wales during Easter. **Drill-down** is achieved by moving from higher-level summaries (e.g., total fatalities by year) into more detailed views (e.g., by state and crash type). **Roll-up** works in the opposite direction by summarizing detailed data into higher levels, such as combining results across all crash types or across all states.

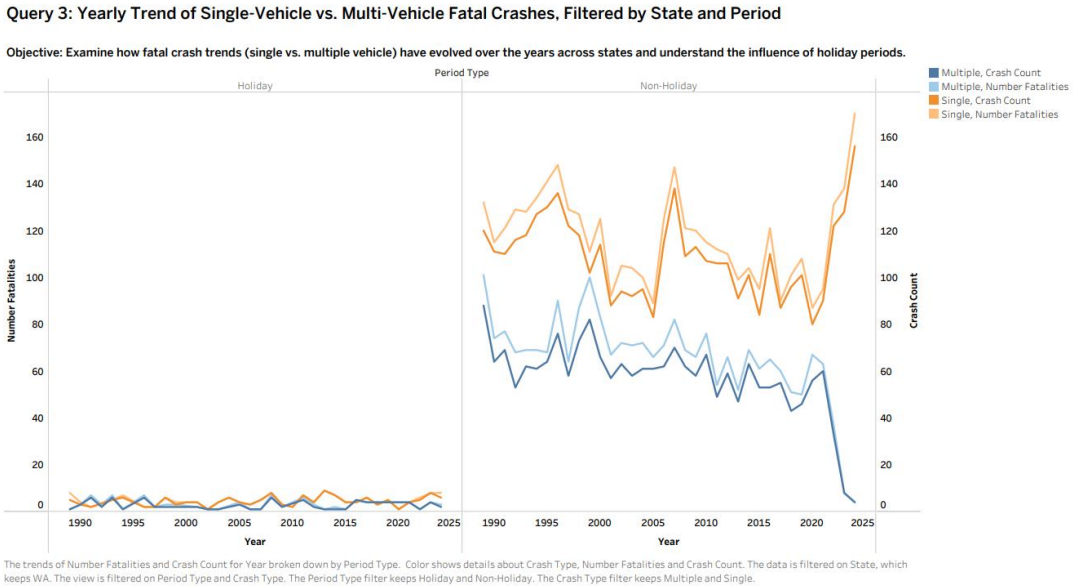
Including both **fatality counts,** and **crash counts** allows for a comparative trend analysis. It helps identify where crashes occur most frequently and where they tend to result in more fatalities. This dual-metric analysis supports informed decision-making around crash severity and geographic risk concentration.

StarNet for query 3



Output for query 3

```
Query 3 - CUBE result:
     year state crash_type christmas_period easter_period  total_fatalities  \
0  1989.0  None       None             None          None              2800
1  1989.0  None       None             None            No              2800
2  1989.0  None       None               No            No              2723
3  1989.0  None       None               No          None              2723
4  1989.0  None     Single             None            No              1428

   total_crashes
0           2407
1           2407
2           2343
3           2343
4           1324
```



Key Insights

1. **Long-Term Downward Trend (1990–Mid-2000s)**
    a. **Observation:** Both holiday and non-holiday lines generally decline from the early 1990s to the mid-2000s, reflecting improved road safety measures (e.g., safer vehicles, better enforcement).

b. **Interpretation:** Policy interventions and vehicle technology advancements likely contributed to fewer fatal crashes during this time.

2. **Persistent Gap: Multi-Vehicle vs. Single-Vehicle**

    a. **Observation:** Across nearly all years, multi-vehicle crashes show higher fatalities and/or higher crash counts than single-vehicle crashes, whether in holiday or non-holiday contexts.

    b. **Interpretation:** Collisions involving multiple vehicles have inherently higher fatality risks due to the involvement of more road users.

3. **Holiday vs. Non-Holiday Patterns**

    a. **Observation:** Holiday periods show fewer total crash days but can still have sharp spikes (particularly in multi-vehicle lines), indicating intense travel periods.

    b. **Observation:** Non-holiday periods dominate in absolute numbers, simply because most of the year is non-holiday. Nevertheless, these lines also show meaningful fluctuations over time.

4. **Recent Fluctuations (Post-2010)**

    a. **Observation:** After a notable dip in the mid- to late-2000s, there are some upticks in the lines around the 2015–2025 range, especially for multi-vehicle crashes.

    b. **Interpretation:** Changes in road usage patterns, population growth, or relaxed enforcement could be contributing to these modest rebounds.

Query 4: Peak Time of Day and Speed Limit Correlation with Victim Demographics

```
SELECT
```

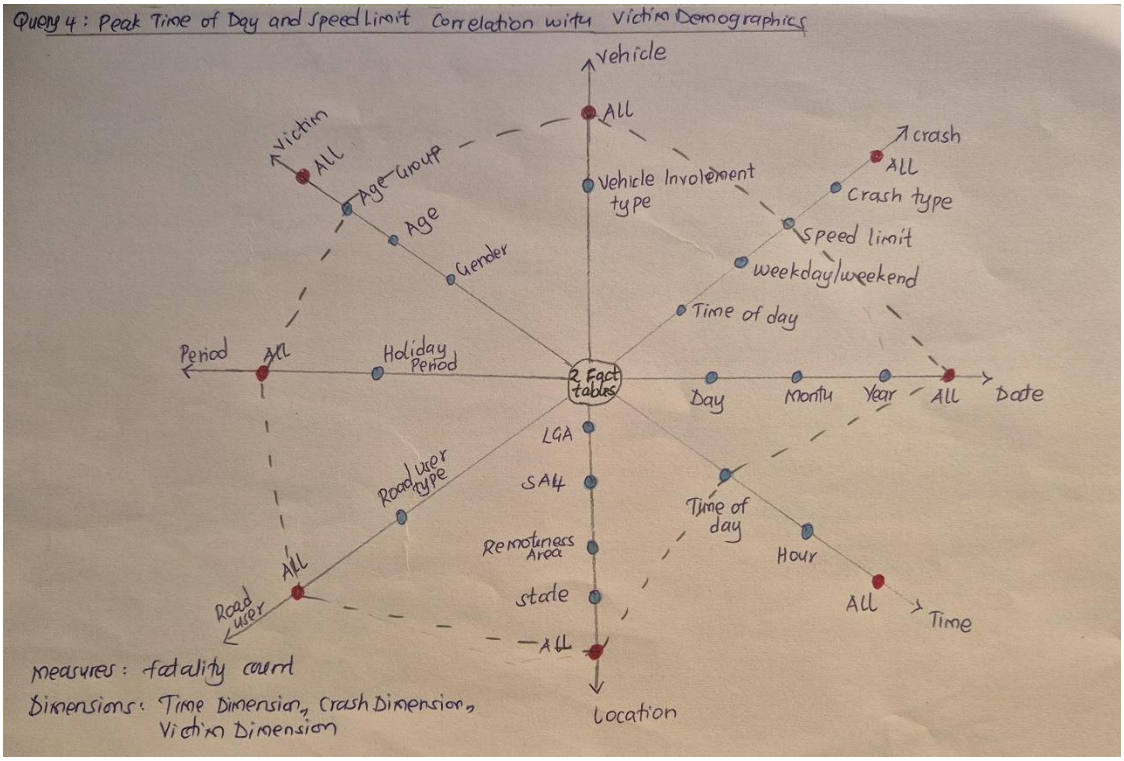
```
    t.time_of_day,
    c.speed_limit,
    v.age_group,
    v.gender,
    SUM(ff.fatality_count) AS total_fatalities
FROM FactFatalities ff
JOIN TimeDimension t ON ff.time_key = t.time_key
JOIN CrashDimension c ON ff.crash_key = c.crash_key
JOIN VictimDimension v ON ff.victim_key = v.victim_key
GROUP BY CUBE (
    t.time_of_day,
    c.speed_limit,
    v.age_group,
    v.gender
)
ORDER BY total_fatalities DESC;
```

In this query, we explored fatality patterns based on **time of day**, **speed limit**, and victim demographics—specifically **age group** and **gender**. The query uses GROUP BY CUBE to generate aggregates across all combinations of these four dimensions. This approach supports flexible and layered analysis from broad trends to detailed breakdowns.

Slicing is performed when data is filtered by a single attribute—for instance, looking only at fatalities that occurred during the night. **Dicing** involves more refined subsetting, such as analyzing fatalities among females aged 40–64 in zones with a 100 km/h speed limit at night. **Drill-down** allows for deeper exploration from general categories (e.g., total fatalities by time of day) into more specific layers (e.g., time of day by age group and gender). Conversely, **roll-up** enables summarization from detailed views to higher-level overviews, such as total fatalities across all genders for each age group.

This multidimensional analysis provides insights into how crash severity varies by speed environment, time context, and vulnerable population groups. It also reveals intersections between risky driving conditions and demographic sensitivity, which are crucial for targeting road safety campaigns and interventions.

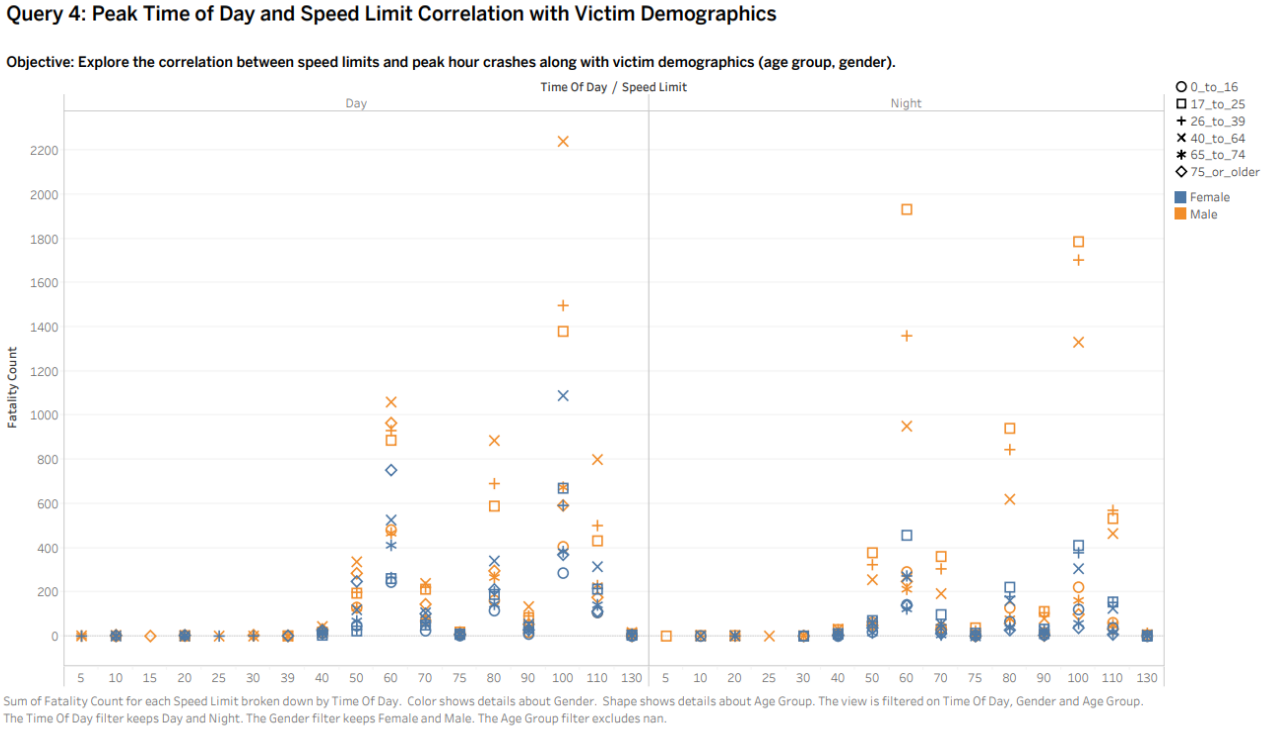
Starnet for query 4



Output for query 4

```
Query 4 - CUBE result:
  time_of_day speed_limit age_group gender  total_fatalities
0        None        None      None   None             51284
1        None        None      None   Male             37147
2         Day        None      None   None             29238
3       Night        None      None   None             22006
4         Day        None      None   Male             19729
```

Tableau visualization for query 4



Key Insights

1. **Higher Speed Limits, Higher Fatalities**
    a) **Observation:** At or above 80–100 km/h, there is a clear increase in fatality counts, particularly during nighttime hours.
    b) **Interpretation:** High-speed roads pose greater risks at night, likely due to reduced visibility and slower reaction times.
2. **Day vs. Night Contrast**
    a. **Observation:** Daytime crashes show a more gradual climb in fatality counts as speed limits increase.
    b. **Observation:** Nighttime crashes often peak sharply around 100–110 km/h, indicating that darkness intensifies the dangers of already high-speed travel.
3. **Demographic Variation**

a. **Observation:** Certain age groups and genders (e.g., younger males or middle-aged drivers) appear clustered in the higher speed limit categories (≥100 km/h), especially at night.

b. **Interpretation:** Some populations may be more prone to speed-related risk or drive more frequently on high-speed roadways.

4. **Lower Speed Limits Show Fewer Fatalities**

a. **Observation:** In the 40–60 km/h range, fatality counts remain relatively low for both day and night.

b. **Interpretation:** Urban or residential zones with lower limits seem less prone to fatal crashes, underscoring the value of safer speeds.

Query 5: Fatalities Per Dwelling in Remote LGAs, Split by Day vs. Night, Road User, and Holiday Period

```
SELECT
  l.remoteness_area,
  t.time_of_day,
  ru.road_user,
  p.christmas_period,
  p.easter_period,
  ROUND(
    (SUM(ff.fatality_count)::numeric / NULLIF(SUM(l.dwelling_count), 0)) * 10000, 2
  ) AS fatalities_per_10k_dwellings
FROM FactFatalities ff
JOIN LocationDimension l ON ff.location_key = l.location_key
JOIN TimeDimension t ON ff.time_key = t.time_key
JOIN RoadUserDimension ru ON ff.road_user_key = ru.road_user_key
JOIN PeriodDimension p ON ff.period_key = p.period_key
WHERE l.remoteness_area IN ('Remote Australia', 'Very Remote Australia')
GROUP BY CUBE (
  l.remoteness_area,
```

```
    t.time_of_day,
    ru.road_user,
    p.christmas_period,
    p.easter_period
)
ORDER BY fatalities_per_10k_dwellings DESC;
```

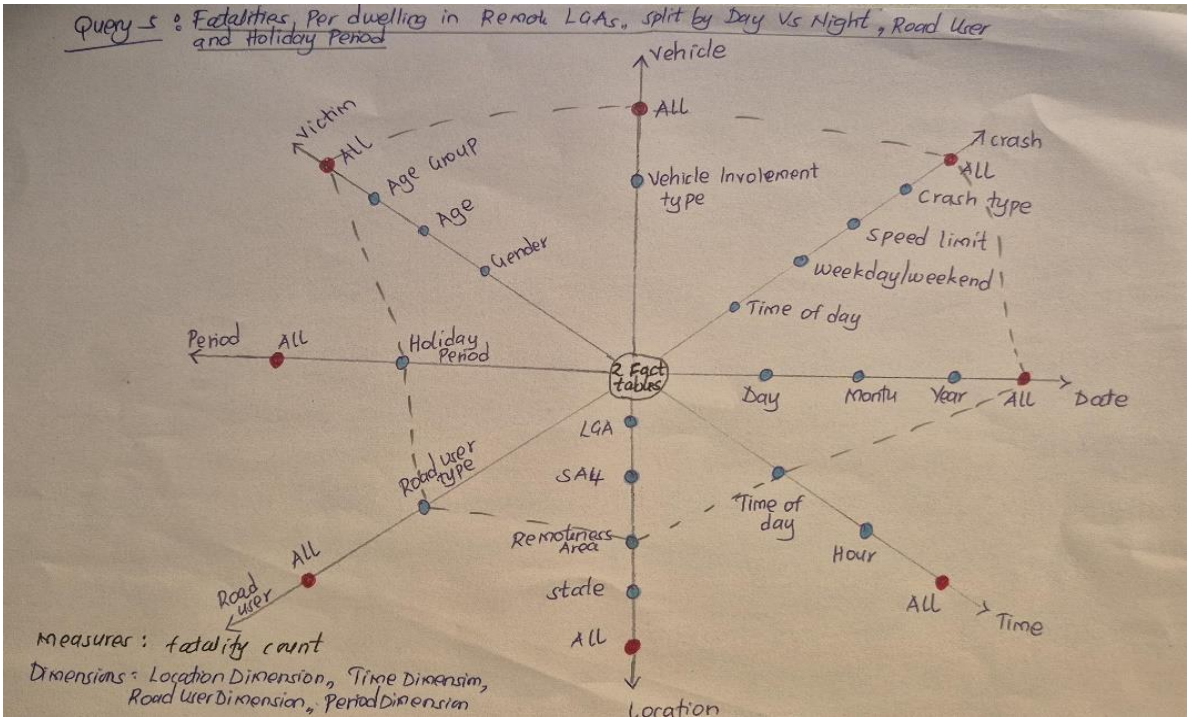
For this query, we focused on calculating the **fatalities per 10,000 dwellings** in Remote and Very Remote Australia. Rather than just counting the number of fatalities, we normalized the data using the total number of dwellings to make the comparison fair across regions with different population sizes.

We grouped the data based on five key dimensions: **remoteness area**, **time of day**, **road user type**, and whether the crash occurred during the **Christmas** or **Easter period**. To explore the data across all possible combinations of these dimensions, we used the GROUP BY CUBE function. This allowed us to perform **slicing**, such as viewing fatalities only for pedestrians or only for night-time crashes. We could also **dice** the data to look at smaller subsets, like fatalities involving passengers during Easter night in Very Remote areas.

We were also able to **drill down** into more detailed combinations—starting from the overall fatality rate in Remote Australia and narrowing it down to specific road users or holiday periods. Conversely, we could **roll up** to higher-level summaries, such as total fatalities per remoteness area without splitting by other attributes.

This analysis gave us deeper insight into how location, time, and demographic factors interact to influence fatality risks in remote communities. It helped us identify conditions under which people are more vulnerable and supported more focused recommendations for intervention.

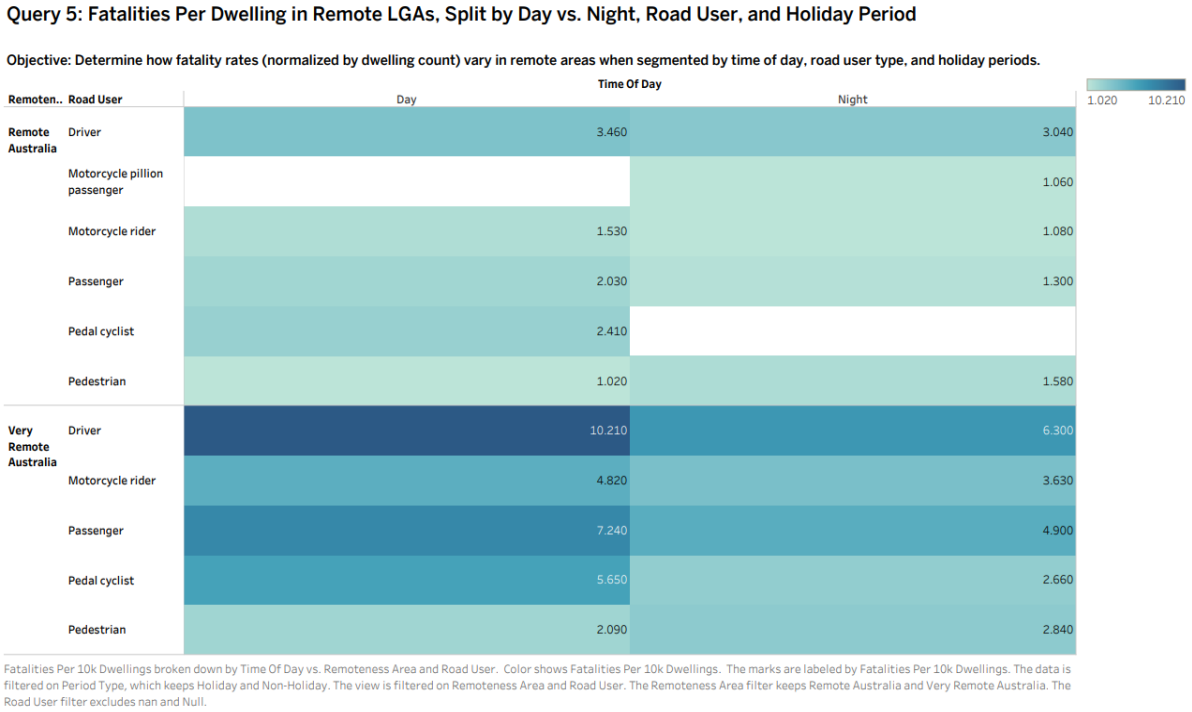
StarNet for query 5



Output for query 5

```
Query 5 - CUBE result:
          remoteness_area time_of_day      road_user christmas_period  \
0  Very Remote Australia         None  Pedal cyclist                No
1  Very Remote Australia          Day  Pedal cyclist                No
2  Very Remote Australia          Day  Pedal cyclist                No
3  Very Remote Australia         None  Pedal cyclist                No
4  Very Remote Australia        Night      Passenger               Yes

  easter_period  fatalities_per_10k_dwellings
0          None                         15.82
1            No                         15.82
2          None                         15.82
3            No                         15.82
4            No                          7.69
```

Tableau visualization for query 5



Key Insights

1. **Very Remote vs. Remote: Significantly Higher Daytime Rates**
    a. **Observation:** Very Remote Australia generally shows higher daytime fatality rates compared to Remote Australia, especially for drivers.
    b. **Interpretation:** Sparse emergency services, greater distances traveled, and higher average speeds in very remote areas may increase daytime risk.
2. **Nighttime Risk Remains Prominent**
    a. **Observation:** Although daytime rates spike in Very Remote regions, nighttime fatalities can also be substantial for certain road users (e.g., motorcycle riders, drivers), in both Remote and Very Remote areas.
    b. **Interpretation:** Limited lighting, drowsiness, and slower emergency response times at night compound risks, even if absolute numbers are lower than daytime in some cases.

3. **Road User Disparities**

    a. **Observation:** Drivers and Motorcycle Riders consistently show higher fatalities per dwelling than other categories (passengers, pedal cyclists, pedestrians).

    b. **Interpretation:** Motorized vehicles moving at higher speeds in remote conditions face more severe outcomes in a crash. Non-motorized users appear less frequently, yet they remain vulnerable on unlit or poorly maintained roads.

4. **Mixed Holiday vs. Non-Holiday Patterns**

    a. **Observation:** Filtering by holiday periods sometimes reveals slightly elevated rates (depending on LGA and road user) but does not universally eclipse the ongoing year-round challenges.

    b. **Interpretation:** While holidays might create travel spikes in remote areas, the overarching risk factors—long distances, limited infrastructure—are present regardless of season.

8. Association Rule Mining Explanation

I) Algorithm & Rationale

- We **chose** the **Apriori** algorithm to discover patterns from our fatal-crash dataset as introduced by Agrawal et al. [1]. Apriori is straightforward to understand and implement, requiring two main parameters:
- **Min Support**: Controls how frequently an itemset must appear in our data.
- **Min Confidence**: Ensures that any rule we accept has a strong conditional probability of the consequent given the antecedent.
- We **referenced** the work of Agrawal et al. [1] for Apriori's theoretical foundation and leveraged the *mlxtend* library [3] for a practical Python implementation.

II) Steps We Followed

- **Selecting & Binning Attributes**

- We identified relevant categorical features—like State, Crash Type, Road User, and a binned version of Speed Limit (e.g., Low/Med/High).
- Binning Speed Limit helps reduce the numeric range to manageable discrete categories, making our rules more interpretable.

- **Preparing Transactions (One-Hot Encoding)**
  - We converted each row of our data into "transactions" by creating strings like "State=NSW", "Crash Type=Single", etc.
  - We applied TransactionEncoder to transform these transactions into a **one-hot** matrix. This step is essential for Apriori to identify frequent itemsets.

- **Running Apriori & Generating Rules**
  - We ran Apriori with a chosen min_support (e.g., 1%) to find itemsets that appear frequently in our data.
  - We used association_rules(...) to derive rules from these frequent itemsets, specifying a minimum confidence (e.g., 50%). This phase gave us rules along with their support, confidence, and lift.

- **Filtering for "Road User"**
  - Because our primary interest is how certain conditions predict occupant type, we filtered rules to keep only those where the consequent (right-hand side) is "Road User=<some role>".
  - We ranked these final rules by lift (descending) and secondarily by confidence.

- **Interpreting Top k Rules**
  - For each of the top rules, we translated the results into plain English—explaining support ("how common"), confidence ("probability of occupant type given the conditions"), and lift ("factor by which occupant type is more likely under the conditions").

III) Key Metrics

- **Support**: Fraction of total crashes/fatalities that have the entire rule's antecedent plus "Road User=XYZ."
- **Confidence**: Probability that "Road User=XYZ" occurs given the antecedent items (e.g., if State=NSW and Speed=Med, how often does that occupant end up being a Pedestrian?).
- **Lift**: The ratio of the rule's confidence to the base probability of the occupant type, telling us how many times more likely the occupant type is under those specific conditions.

IV) Results & Insights

- Top Rules often showed that single-vehicle crashes in certain states (e.g., Vic or NSW), combined with moderate speed zones, significantly increased the likelihood of a pedestrian occupant (lift > 3).
- Some rules highlighted that multiple-vehicle crashes in high speed-limit areas corresponded strongly to a "road_user=Driver" occupant.

Code for Association Rule Mining

```python
def association_rule_mining(df, min_support=0.01, min_confidence=0.5):
    """
    Performs association rule mining on a subset of df_merged_final.
    1) Select & discretize columns.
    2) Convert to transactions or one-hot encoding.
    3) Apply Apriori for frequent itemsets.
    4) Generate association rules, focusing on 'Road User' in the consequent.
    5) Return filtered rules sorted by lift & confidence.

    Parameters
    ----------
    df : pd.DataFrame
        Cleaned, final DataFrame.
    min_support : float
        Minimum support threshold for Apriori.
    min_confidence : float
        Minimum confidence threshold for association_rules.

    Returns
    -------
    rules_filtered : pd.DataFrame
        The top rules that have "Road User=..." as part of their consequent,
        sorted by lift (descending) and then confidence (descending).
    """

    # 1) Select relevant categorical columns. Binning 'Speed Limit'
    #    to reduce cardinality.
    df_copy = df.copy()

    def bin_speed_limit(x):
        if x < 40:
            return "Speed=Low"
        elif x < 80:
            return "Speed=Med"
        else:
            return "Speed=High"

    if 'Speed Limit' in df_copy.columns:
        df_copy['speed_limit_bin'] = df_copy['Speed Limit'].apply(bin_speed_limit).astype(str)
    else:
        df_copy['speed_limit_bin'] = "Speed=Missing"

    # Convert 'Road User' to string to ensure consistent handling.
    df_copy['road_user'] = df_copy['Road User'].astype(str)

    # Subset columns to transform into 'transactions'.
    columns_for_mining = [
        'State',            # e.g., 8 categories
        'Crash Type',       # Single/Multiple
        'road_user',        # e.g., Driver, Passenger, etc.
        'speed_limit_bin'   # Binned speed
    ]

    # 2) Convert each row to a list of "col=value" items for one-hot encoding.
    transaction_list = []
    for _, row in df_copy.iterrows():
        row_items = []
        for col in columns_for_mining:
            value_str = str(row[col]).strip()
            if value_str.lower() not in ['nan', 'none', '']:
                item_str = f"{col}={value_str}"
```

```
                row_items.append(item_str)
        transaction_list.append(row_items)

    # 3) One-hot encode the transactions for Apriori.
    te = TransactionEncoder()
    te_ary = te.fit(transaction_list).transform(transaction_list)
    df_onehot = pd.DataFrame(te_ary, columns=te.columns_)

    # 4) Run Apriori
    frequent_itemsets = apriori(df_onehot, min_support=min_support, use_colnames=True)

    # 5) Generate rules
    rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=min_confidence)

    # 6) Filter rules => 'road_user=' in the consequent
    rules_filtered = rules[rules['consequents'].apply(
        lambda c: any('road_user=' in item for item in c)
    )].copy()

    # 7) Sort by lift (desc), then confidence (desc)
    rules_filtered.sort_values(by=["lift", "confidence"], ascending=[False, False], inplace=True)

    return rules_filtered
```

Usage

```
if __name__ == "__main__":
    # 1) Suppose 'df_merged_final' is your final cleaned dataset.
    print("Running association rule mining on df_merged_final...")

    rules_road_user = association_rule_mining(
        df_merged_final,
        min_support=0.01,
        min_confidence=0.5
    )

    # 2) Display top 10 rules
    print("\nTop 10 association rules with 'Road User' in the consequent:\n")
    print(rules_road_user.head(10))
```

Results:

Top 10 association rules with 'Road User' in the consequent:

	antecedents	consequents	antecedent support	consequent support	support
167	(Crash Type=Single, State=Vic, speed_limit_bin...	(road_user=Pedestrian)	0.049138	0.168259	0.026324
147	(State=NSW, Crash Type=Single, speed_limit_bin...	(road_user=Pedestrian)	0.077373	0.168259	0.040792
120	(State=NSW, speed_limit_bin=Speed=High, Crash ...	(road_user=Driver)	0.079030	0.457726	0.046584
164	(Crash Type=Single, speed_limit_bin=Speed=High...	(road_user=Driver)	0.069222	0.457726	0.040422
95	(State=NSW, speed_limit_bin=Speed=High)	(road_user=Driver)	0.171418	0.457726	0.099797
128	(State=Qld, speed_limit_bin=Speed=High, Crash ...	(road_user=Driver)	0.054617	0.457726	0.031628
143	(State=NSW, Crash Type=Single, speed_limit_bin...	(road_user=Driver)	0.092387	0.457726	0.053213
161	(State=SA, Crash Type=Single, speed_limit_bin=...	(road_user=Driver)	0.031277	0.457726	0.017842
105	(State=SA, speed_limit_bin=Speed=High)	(road_user=Driver)	0.052590	0.457726	0.029873
40	(speed_limit_bin=Speed=High, Crash Type=Multiple)	(road_user=Driver)	0.261446	0.457726	0.148253

	confidence	lift	representativity	leverage	conviction
167	0.535714	3.183865	1.0	0.018056	1.791442
147	0.527218	3.133368	1.0	0.027774	1.759247
120	0.589440	1.287758	1.0	0.010409	1.320816
164	0.583944	1.275750	1.0	0.008737	1.303368
95	0.582186	1.271911	1.0	0.021335	1.297886
128	0.579079	1.265122	1.0	0.006628	1.288304
143	0.575981	1.258355	1.0	0.010925	1.278893
161	0.570449	1.246268	1.0	0.003526	1.262421
105	0.568039	1.241002	1.0	0.005801	1.255377
40	0.567050	1.238842	1.0	0.028582	1.252509

	zhangs_metric	jaccard	certainty	kulczynski
167	0.721363	0.137769	0.441790	0.346082
147	0.737952	0.199143	0.431575	0.384828
120	0.242632	0.095035	0.242892	0.345606
164	0.232223	0.083083	0.232757	0.336127
95	0.258009	0.188529	0.229516	0.400107
128	0.221670	0.065793	0.223786	0.324088
143	0.226211	0.107091	0.218074	0.346119
161	0.203984	0.037868	0.207871	0.304714
105	0.204980	0.062178	0.203426	0.316651
40	0.261043	0.259674	0.201603	0.445470

V) Interpreting the Top 3 Rules

```python
def interpret_rule(row):
    """
    Return a plain-English summary for a single rule row.
    """
    ant_items = ", ".join(list(row['antecedents']))
    con_items = ", ".join(list(row['consequents']))
    return (f"RULE: If {ant_items}, THEN {con_items}\n"
            f"    - Support: {row['support']:.4f}\n"
            f"    - Confidence: {row['confidence']:.4f}\n"
            f"    - Lift: {row['lift']:.4f}\n")

def explain_top_k_rules(rules_df, k=3):
    """
    Print out top k rules with plain-English commentary.
    """
    top_k_rules = rules_df.head(k)
    for i, (_, row) in enumerate(top_k_rules.iterrows(), 1):
        print(f"Top Rule #{i}\n{interpret_rule(row)}")


explain_top_k_rules(rules_road_user, k=3)
```

Output

```
Top Rule #1
RULE: If Crash Type=Single, State=Vic, speed_limit_bin=Speed=Med, THEN road_user=Pedestrian
   - Support: 0.0263
   - Confidence: 0.5357
   - Lift: 3.1839

Top Rule #2
RULE: If State=NSW, Crash Type=Single, speed_limit_bin=Speed=Med, THEN road_user=Pedestrian
   - Support: 0.0408
   - Confidence: 0.5272
   - Lift: 3.1334

Top Rule #3
RULE: If State=NSW, speed_limit_bin=Speed=High, Crash Type=Multiple, THEN road_user=Driver
   - Support: 0.0466
   - Confidence: 0.5894
   - Lift: 1.2878
```

1. **Rule #1**
    a. **Antecedent**: Single-vehicle crash, Victoria, medium speed limit (40–80 km/h)
    b. **Consequent**: Road User = Pedestrian
    c. **Interpretation**: If a crash occurs under these conditions, there is a **53.57%** chance the road user is a pedestrian (confidence). This probability is **3.18 times** higher than expected by random chance (lift).
2. **Rule #2**

a. **Antecedent**: Single-vehicle crash, New South Wales, medium speed limit (40–80 km/h)
b. **Consequent**: Road User = Pedestrian
c. **Interpretation**: Under these conditions, **52.72%** of crashes involve a pedestrian, which is about **3.13 times** more likely than by chance alone (lift).

3. **Rule #3**

a. **Antecedent**: Multiple-vehicle crash, New South Wales, high speed limit (>80 km/h)
b. **Consequent**: Road User = Driver
c. **Interpretation**: When multiple vehicles crash in NSW at high speeds, **58.94%** of those crashes involve drivers (lift = 1.29). Though the lift is smaller here, it still indicates an above-chance likelihood that the primary road user is the driver.

Rule #	Antecedents	Consequent	Support	Confidence	Lift
1	Crash Type=Single, State=Vic, Speed=Med	Road User=Pedestrian	0.0263	0.5357	3.1839
2	Crash Type=Single, State=NSW, Speed=Med	Road User=Pedestrian	0.0408	0.5272	3.1334
3	State=NSW, Speed=High, Crash Type=Multiple	Road User=Driver	0.0466	0.5894	1.2878

Plain English meaning

- Support: Out of all crashes/fatalities, what fraction had these antecedent conditions and the Road User=XYZ?
- Confidence: If the antecedent is present (e.g., State=Vic, CrashType=Single, Speed=Med), how likely (fraction) is it that the occupant is Road User=Pedestrian?
- Lift: How many times more likely is Road User=Pedestrian given the antecedent, compared to the unconditional chance of a pedestrian occupant?

VI) Insights & Recommendations

After identifying the top rules with "Road User" in the consequent:

1. If Pedestrians Are Common in Low-Speed or Single-Vehicle Crashes

   Recommendation*:* Improve pedestrian safety infrastructure (e.g., enhanced crosswalks, signage) even in lower-speed areas, as single-vehicle crashes might heavily involve pedestrians.

2. If Motorcyclists Are Over-Represented in High-Speed Regions

- **Recommendation:** Target speed enforcement, especially in highways or outer roads where motorcycles appear frequently. Educate riders on safe speed compliance.

3. If Drivers Dominate Crashes in Certain States or Crash Types

- **Recommendation***:* Focus driver awareness campaigns (billboards, driver rest areas) in those states or SA4 regions. Possibly lower speed limits if we see "Speed=High" is a big factor.

These derived rules and suggestions allow policymakers to prioritize safety interventions based on actual occupant-level patterns uncovered by the data.

VII) Conclusion

- **Algorithm:** Apriori with support, confidence, and lift metrics.
- **Focus:** "Road User" in the consequent.
- **Findings:** High-lift rules highlight specific state + crash type + speed-limit conditions that strongly predict occupant roles (Driver, Passenger, Pedestrian, etc.).

- **Policy Impact:** Through analyzing these top rules, the government can tailor safety measures—enforcement, educational campaigns, or infrastructural improvements—to the occupant roles and crash contexts that pose the greatest risk.

9. Road Safety Strategy to reduce fatal crashes and fatalities derived from queries, tableau visualizations and association rule mining

1.Year-Round Multi-Pronged Enforcement

1. **Non-Holiday Speed Control**
    a. **Rationale**: High fatality counts are observed on regular (non-holiday) days across day/night times.
    b. **Action**:
        i. Increase **mobile speed camera** deployments in known hotspots.
        ii. Expand **DUI checkpoints** on busy non-holiday nights.
        iii. Strengthen **rapid incident response** units in high-traffic corridors.
2. **Targeted Holiday Interventions**
    a. **Daytime, Multi-Vehicle Focus**: Given higher multi-vehicle crashes during the day on holidays, deploy **traffic flow management**, real-time alerts, and add more **police patrols** to prevent dangerous overtaking.
    b. **Nighttime, Single-Vehicle Emphasis**: Address **fatigue and impaired driving** through expanded night checks, public service announcements (PSAs), and well-lit rest areas.

2. State- and LGA-Specific Measures

1. **Tailored Local Collaborations**
    a. **Rationale**: Different LGAs (e.g., Perth, Albany, Bunbury) show unique crash patterns and road-user distributions.

b. **Action**:

i. Form local task forces with council members to **analyze peak crash hours** and locations.

ii. Implement **traffic-calming** (e.g., speed humps, chicanes) in local areas with high motorcycle or pedestrian risk.

2. **Focused Driver Education**

a. **Rationale**: Younger (17–25) and middle-aged (40–64) drivers/motorcyclists are consistently overrepresented at night.

b. **Action**:

i. Use **refresher courses** and advanced driving/riding programs emphasizing **night-driving risks**, fatigue, and defensive driving.

3. <u>Infrastructure & Technology Upgrades</u>

1. **High-Speed Road Improvements**

a. **Rationale**: Fatalities spike at speed limits ≥80 km/h, especially at night.

b. **Action**:

i. Install **rumble strips**, **barriers**, and **enhanced lighting** on high-speed routes.

ii. Consider **dynamic speed limits** that automatically lower at night or in poor visibility.

2. **Pedestrian & Motorcyclist Protections**

a. **Rationale**: Single-vehicle and lower-speed crashes often involve pedestrians, while high-speed regions pose greater risks for motorcycles.

b. **Action**:

i. Create **safe pedestrian zones** (e.g., raised crosswalks, better-lit sidewalks) near urban centers.

ii. For motorcyclists, **designated lanes** or improved lane markings where possible, plus advanced rider-assistance technology (e.g., proximity sensors) in high-use corridors.

3. **Articulated Vehicle Safety Enhancements**

    a. **Rationale**: Crashes involving articulated vehicles show a higher fatality risk, often in multi-vehicle contexts.

    b. **Action**:

        i. Tighten regulations on **driver training** and rest periods.

        ii. Explore **dedicated heavy-vehicle lanes** on major highways with high freight traffic.

4. <u>Continuous Data-Driven Monitoring & Analysis</u>

1. **Real-Time Crash & Traffic Data**

    a. **Rationale**: Post-2010 upticks in multi-vehicle crashes suggest ongoing changes in traffic patterns and demographics.

    b. **Action**:

        i. Deploy **live traffic monitoring systems** for immediate detection of congestion or collisions.

        ii. Use **predictive analytics** to identify emerging hotspots and intervene promptly.

2. **Feedback Loop for Policy Refinement**

    a. **Rationale**: Sustained declines from the 1990s to mid-2000s prove that consistent policy efforts can work but may need updating.

    b. **Action**:

        i. Conduct **annual audits** of fatality data (disaggregated by speed limit, time of day, occupant type).

ii. Adjust programs (e.g., shift patrol hours, expand telematics-driven insurance discounts) based on new findings.

5. <u>Nighttime & Fatigue-Focused Campaigns</u>

1. **High-Risk Age Groups**
    a. **Rationale**: Younger and middle-aged drivers and riders face elevated risk after dark, with potential spikes in fatigue or alcohol use.
    b. **Action**:
        i. Develop **multimedia awareness** campaigns on the dangers of **speeding, drowsy driving**, and **mobile phone distraction** at night.
        ii. Offer **free rest stops**, coffee stations, or discounted lodging near major highways for holiday travelers.
2. **Advanced Vehicle Technology Promotion**
    a. **Rationale**: Fatigue detection systems, lane-departure warnings, and autopilot features can reduce night crashes.
    b. **Action**:
        i. Provide **incentives** (tax credits, insurance discounts) for vehicles equipped with **driver-assistance tech**.
        ii. Encourage fleet owners (including heavy vehicles) to adopt **collision-avoidance systems**.

6. <u>Summary of Expected Outcomes</u>

- **Lower Fatality Counts** in both holiday and non-holiday periods through coordinated **enforcement, engineering, and education**.
- **Reduced Multi-Vehicle Crashes** via advanced **infrastructure** (e.g., improved signage, rumble strips) and **driver education** on safe following distances.

- **Heightened Safety** for **pedestrians and motorcyclists**, especially at medium-speed limits, by investing in **protective infrastructure** and targeted awareness campaigns.
- **Sustained Improvement** over time through **annual data reviews**, enabling adaptive responses to any resurgence in crash figures.

This Road Safety Strategy directly incorporates the insights and recommendations gained from multiple queries, Tableau visualizations, and association rule mining. By strengthening enforcement, upgrading high-risk infrastructure, and tailoring interventions to specific times, demographics, and regions, we can reduce the frequency and severity of fatal crashes—ultimately saving lives across all types of road users.

10. References

[1] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Proc. 1993 ACM SIGMOD Int. Conf. Management of Data*, Washington, D.C., USA, June 1993, pp. 207–216.

[2] R. Kimball and M. Ross, *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. 3rd ed. Hoboken, NJ, USA: John Wiley & Sons, 2013.

[3] S. Raschka, "MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack," *Journal of Open-Source Software*, vol. 3, no. 24, p. 638, 2018.

[4] Bureau of Infrastructure, Transport and Regional Economics (BITRE), "Australian Road Deaths Database," Canberra, Australia, 2024. https://www.bitre.gov.au/statistics/safety/fatal_road_crash_database.

[5] Australian Bureau of Statistics (ABS), "Local Government Area (LGA) Dwelling Counts," Canberra, Australia, 2024. https://tablebuilder.abs.gov.au/.