A Comprehensive Data Analysis and Creation of Data Warehouse on the Crashes and Fatalities in Australia

Created By: Johan Carlo Ilagan, Student No. 23832843

Submitted on: 14 April 2025

Introduction

One of the most common modes of transportation for the average person is driving a car. With hundreds of new drivers obtaining their licenses each day, Australia's roads are becoming increasingly populated. As of late 2023, there was an estimated 19.4 million licensed drivers in Australia alone – a number that has grown since and will continue to grow [1]. Unfortunately, this rise in road usage is accompanied by a troubling trend: an increase in road crashes and fatalities. A 2023 report revealed that Australia experienced its highest number of road fatalities in five years [2].

In response, the country has adopted the Vision Zero initiative – a bold commitment to achieve a total of zero deaths and serious injuries in car crashes [3]. This strategy emphasizes that no loss of life is acceptable, regardless of the circumstances.

This paper aims to support that goal by analyzing several datasets to develop a comprehensive recommendation strategy for both the government and the public. The analysis involves building a data warehouse with appropriate fact tables, dimensions, and concept hierarchies, as well as cleaning and processing data through the *Extract, Transform, and Load (ETL)* process. Additionally, association rules mining algorithms will be applied, and business queries will be addressed using a starnet model and data visualization tools.

Tools and Methodology

This section outlines all the necessary datasets, software, libraries, and other tools used throughout the project to reach the final conclusion. It also elaborates the process of building the data warehouse and the analytical methods implemented.

To gather data on crashes and fatalities that occurred in Australia, two primary datasets were used – *'bitre_fatalities_dec2024.csv'* and *'bitre_fatal_crashes_dec2024.csv'*. The fatal crashes dataset includes attributes describing the location, date/time, and outcome of each crash. The fatalities dataset contains similar crash-related attributes, but also provides detailed information about the individuals involved, including drivers, passengers, and other road users.

For data visualization, Tableau Desktop was utilized to design the appropriate and clear charts that support the business queries. PostgreSQL (SQL) was used to create and manage the data warehouse, while Jupyter Notebook (Python) was used for data cleaning and transformation. Python libraries include *pandas*, *numpy*, and *mlxtend.*

The methodology follows Kimball's four-step approach to data modelling [4]:

1. Selecting the business process to be modelled
2. Determining the grain of the fact table
3. Choosing the dimensions
4. Identifying the numeric measures for the fact table

This structured approach guided the development of the data model, including the formulation of business queries, creation of the dimension tables and the fact table.

Once the model was designed, the next step involved data cleaning and preparation to ensure the datasets could be correctly ingested into the schema. This was performed in Jupyter Notebook using Python and pandas. After transformation, the dataset for each dimension and fact table was exported as CSV files, then put into PostgreSQL to populate the warehouse.

Finally, the association rule mining was applied to uncover patterns and correlations that could inform strategic recommendations for reducing crashes and fatalities on Australian roads.

Fact Table

Each row in the fact table represents a crash recorded in Australia. The fact table includes a primary key to uniquely identify each crash event, along with ten foreign keys that link to ten different dimension tables. For the numeric measures, the fact table contains a crash count, where each row has a constant value of 1 to later aggregate the total number of crashes during analysis. Another numeric measure is the fatalities count, based on the number of fatalities attributed depending on the crash record.

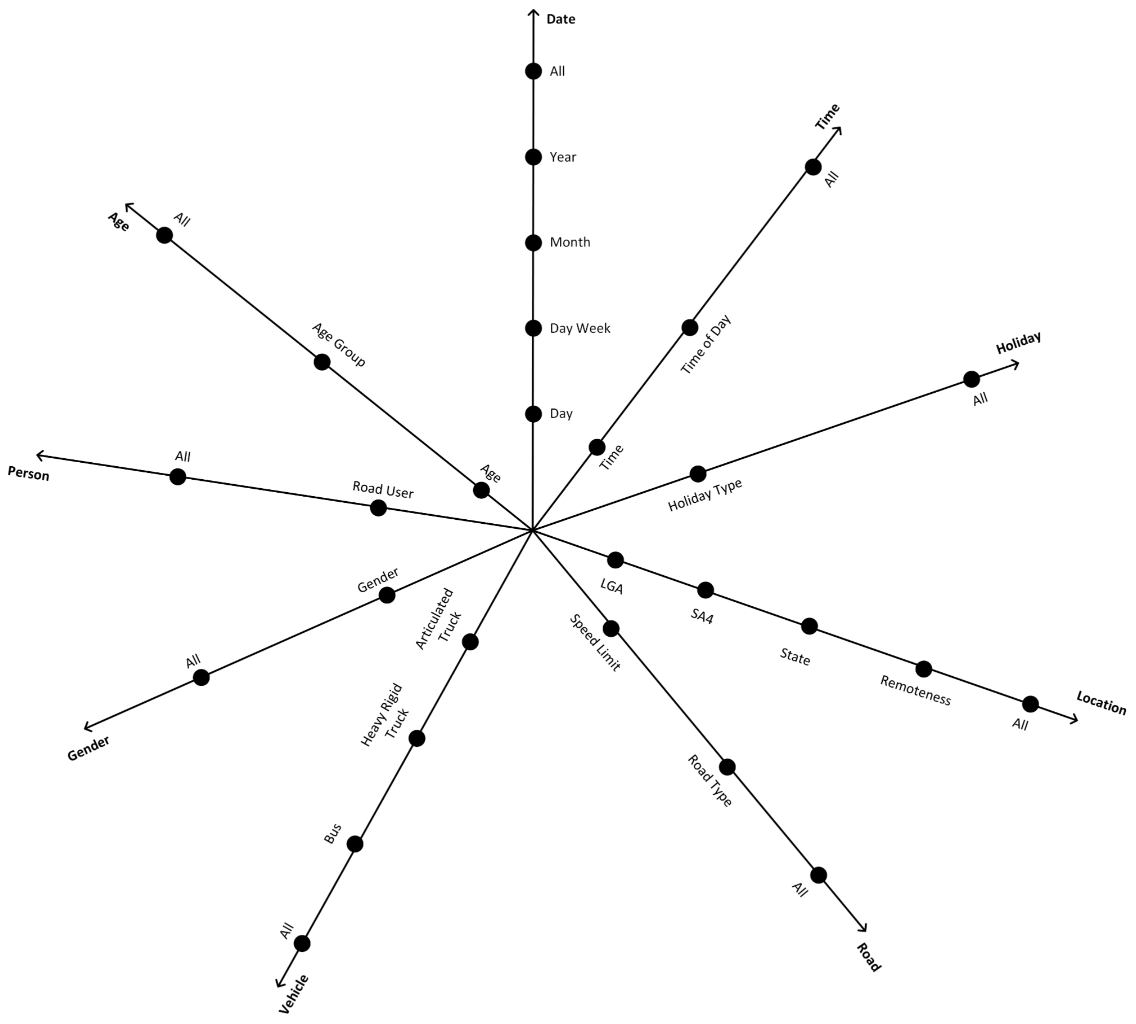
Dimension Tables

There are a total of ten dimensions for this analysis, each having independent concepts and attributes. The dimensions are as follows:

1. **Date** *dim_date* – determines details of when the crash occurred (particularly the date); includes day of the week, month, and year
2. **Time** *dim_time* – determines details regarding the time of the crash; includes time and time of day
3. **Holiday** *dim_holiday* – determines details regarding any potential holiday seasons; includes holiday type (Christmas or Easter)
4. **Outcome** *dim_outcome* – determines any details regarding the outcome of the crash; includes number of fatalities and crash-type
5. **Vehicle** *dim_vehicle* – determines details regarding vehicles that might be involved in the crash; includes boolean attributes

6. **Location** *dim_location* – determines details of where the crash occurred; includes LGA, SA4 names, state, and remoteness area
7. **Road** *dim_road* – determines the type of road where the crash occurred; includes road type and speed limit
8. **Gender** *dim_gender* – determines gender of victim
9. **Person** *dim_person* – determines what type of road user the victim is
10. **Age** *dim_age* – determines age and age group of victim or driver

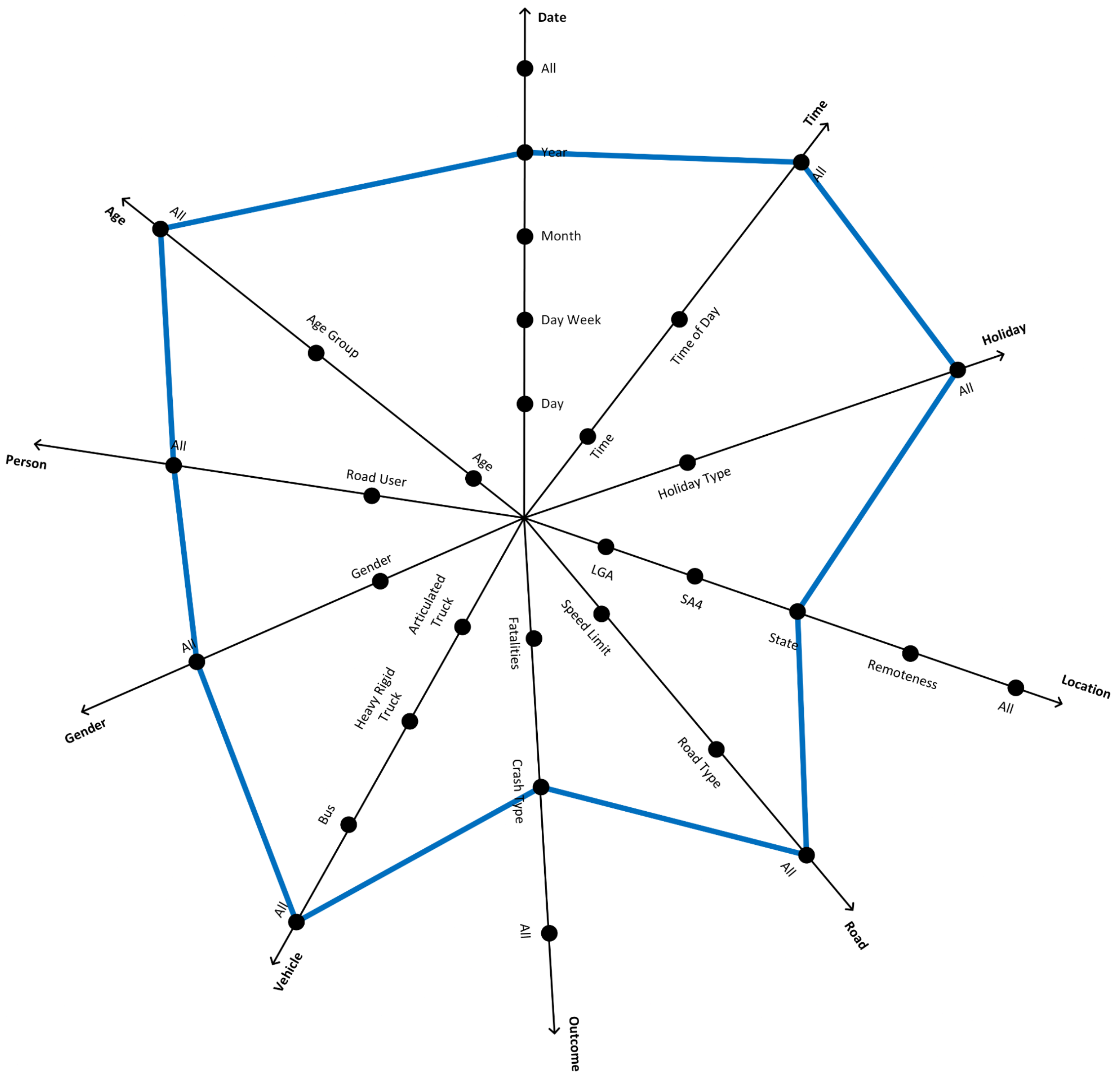
Starnet Model and Business Queries



With the fact table and dimensions now established, a starnet schema can be constructed to visualize how the dimensions interact with each business query. Each query utilizes specific attributes from relevant dimensions, while all other dimensions are set to "All" in the conceptual hierarchy. This model allows for intuitive exploration and aggregation of data across multiple perspectives.

Below are the sample business queries and the corresponding paths through the starnet model:

Query 1: Which state in Australia recorded the most multiple crashes in the last 10 years?

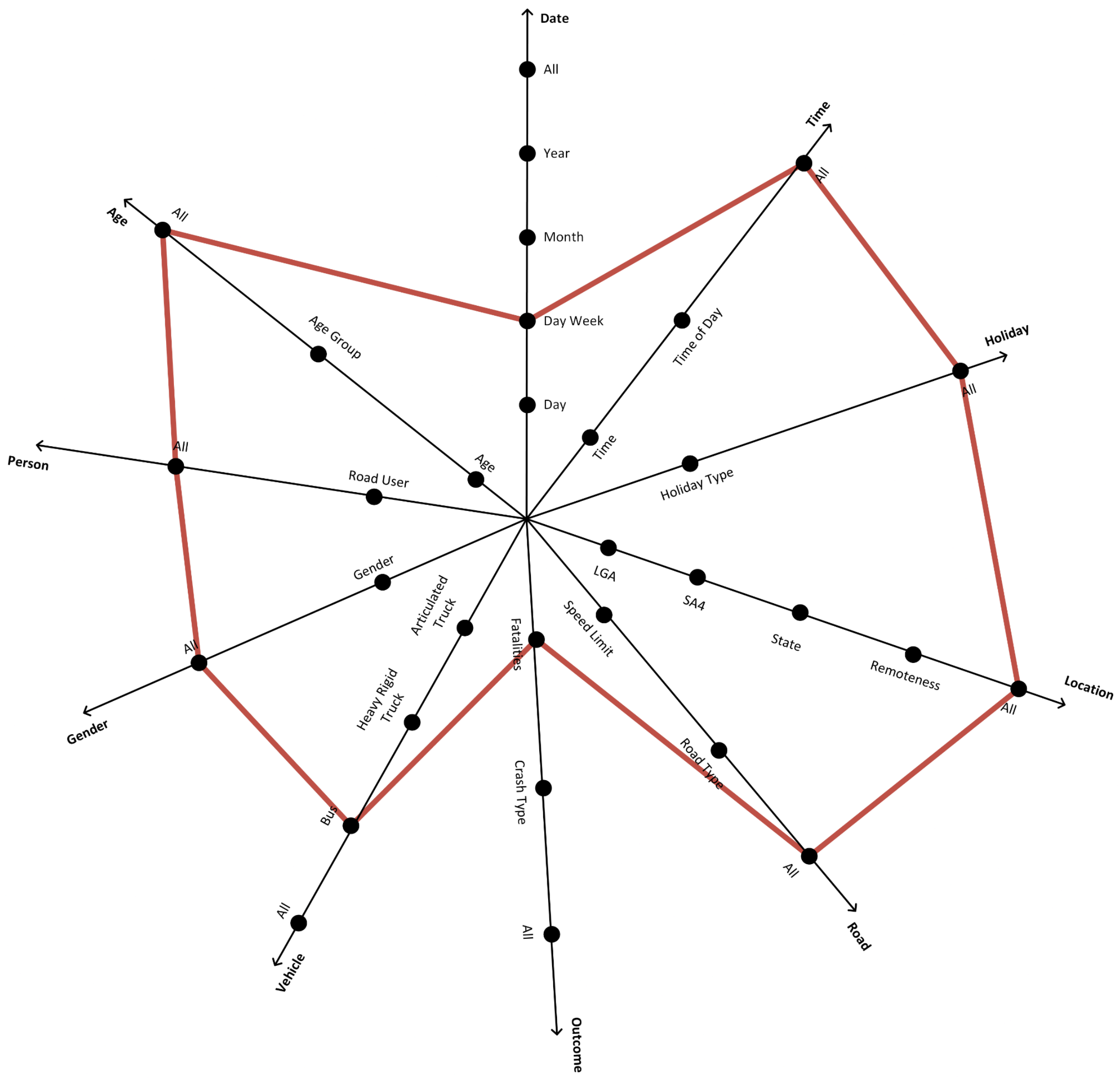


Relevant Dimensions:

- Location Dimension: State
- Crash Dimension: Crash Type (filtered to "Multiple")
- Date Dimension: Year (filtered to the last 10 years)

Other Dimensions: Set to All

Query 2: How many fatalities occurred on a Monday crash that involved a bus?

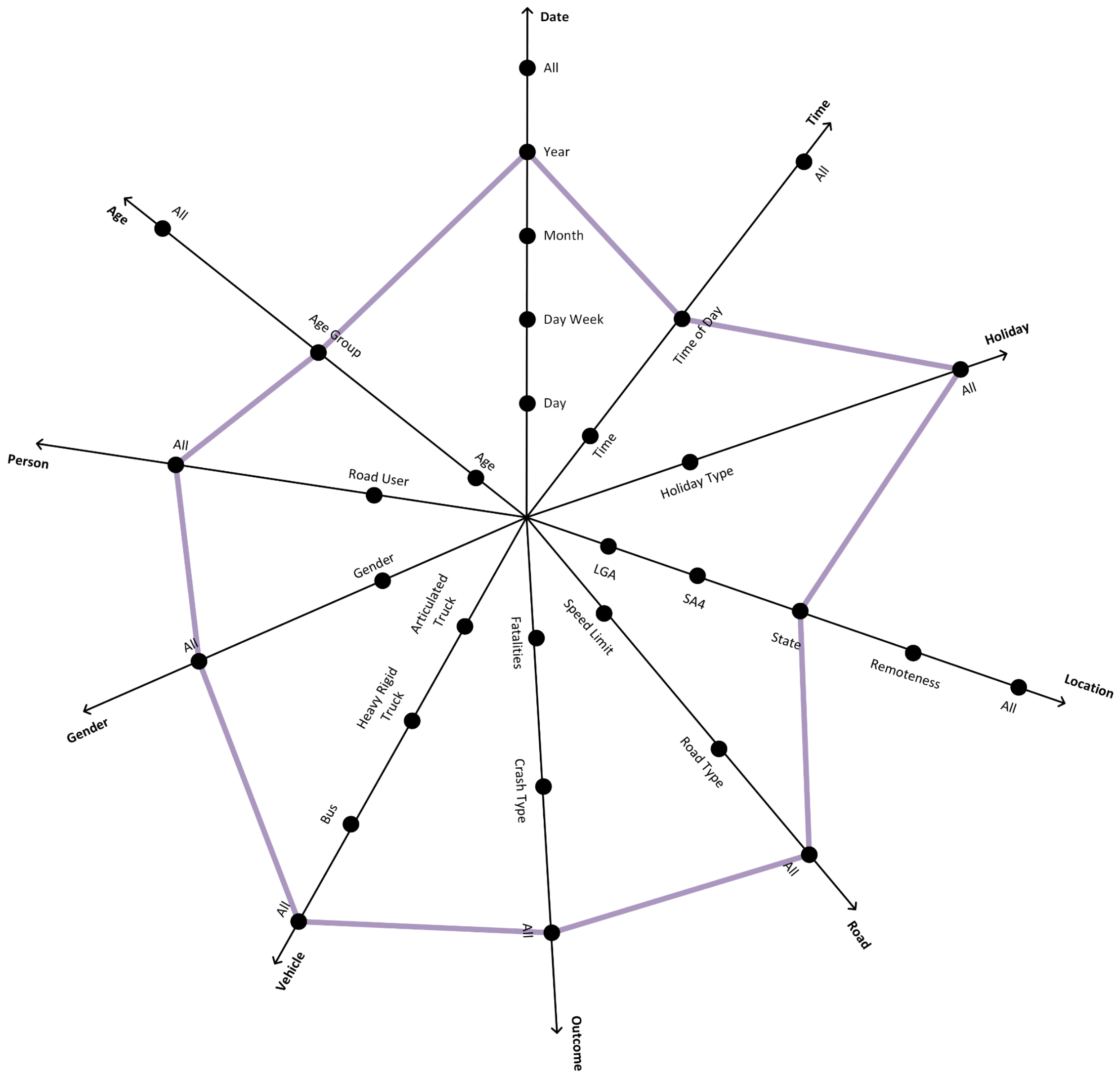


Relevant Dimensions:

- Date Dimension: Day of Week (filtered to "Monday")
- Vehicle Dimension: Vehicle Type (filtered to "Bus")
- Fact Table Measure: Number of Fatalities

Other Dimensions: Set to All

Query 3: What age group is most involved in crashes in states NSW, Queensland, and Victoria at night time in 2021?

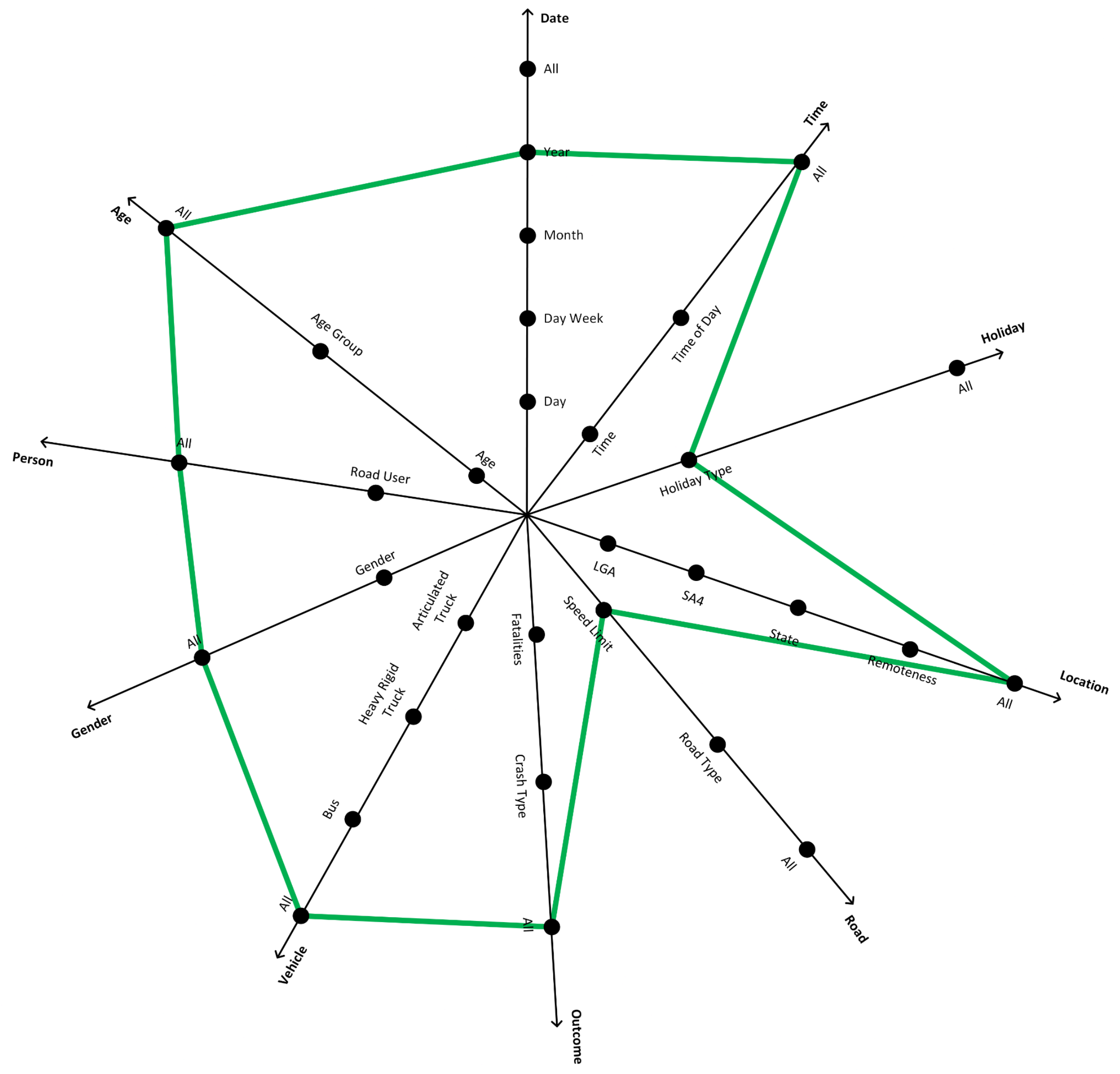


Relevant Dimensions:

- Demographics Dimension: Age Group
- Location Dimension: State (filtered to "NSW", "Qld", "Vic")
- Time Dimension: Time of Day (filtered to "Night")
- Date Dimension: Year (filtered to "2021")

Other Dimensions: Set to All

Query 4: How does the speed limit affect the number of crashes in Australia the past 3 Christmas seasons?

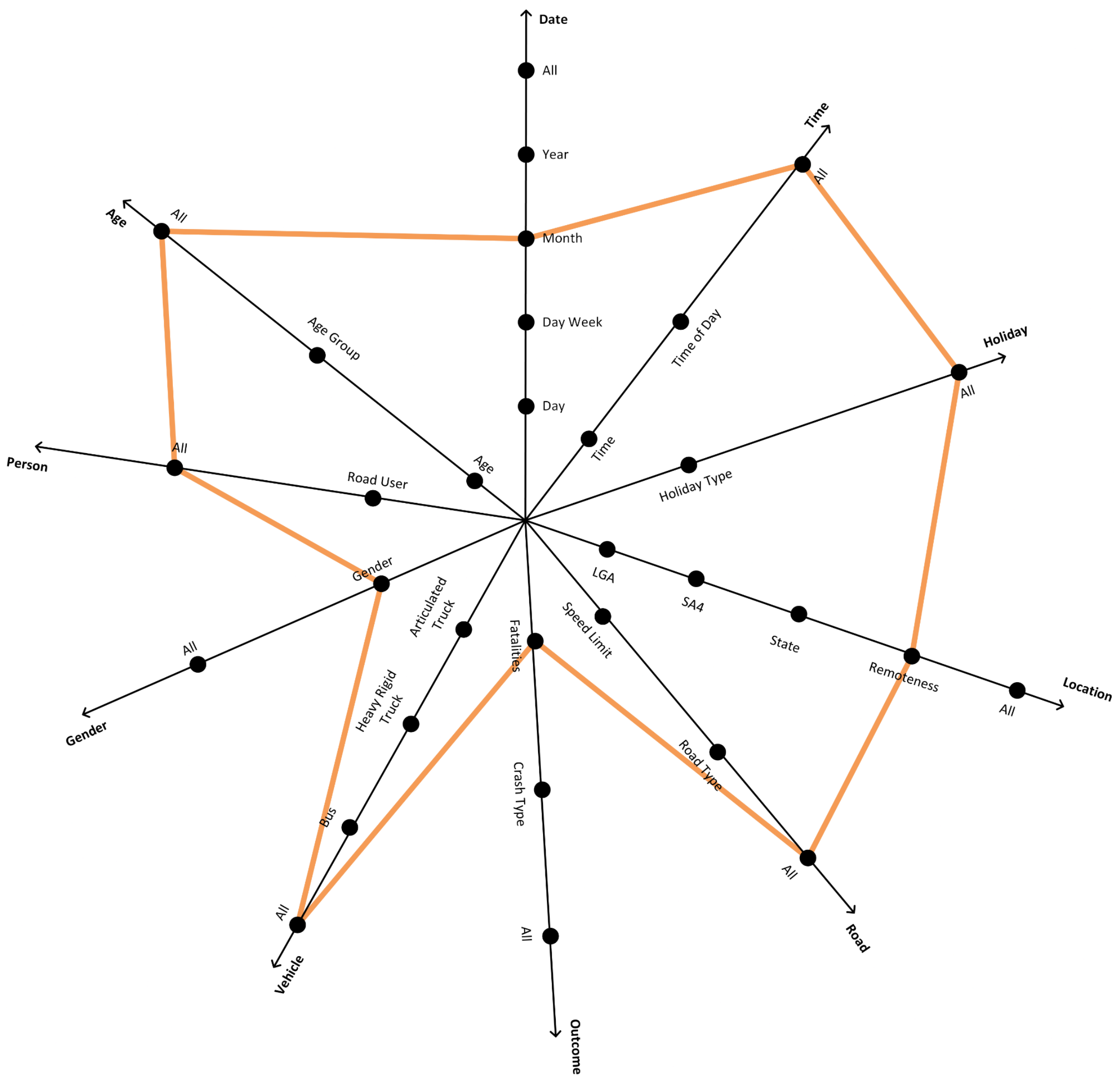


Relevant Dimensions:

- Speed Dimension: Speed Limit
- Holiday Dimension: Holiday Type (filtered to "Christmas")
- Date Dimension: Year (filtered to last 3 years)

Other Dimensions: Set to All

Query 5: Which gender and month combination had the most fatalities in major cities of Australia?



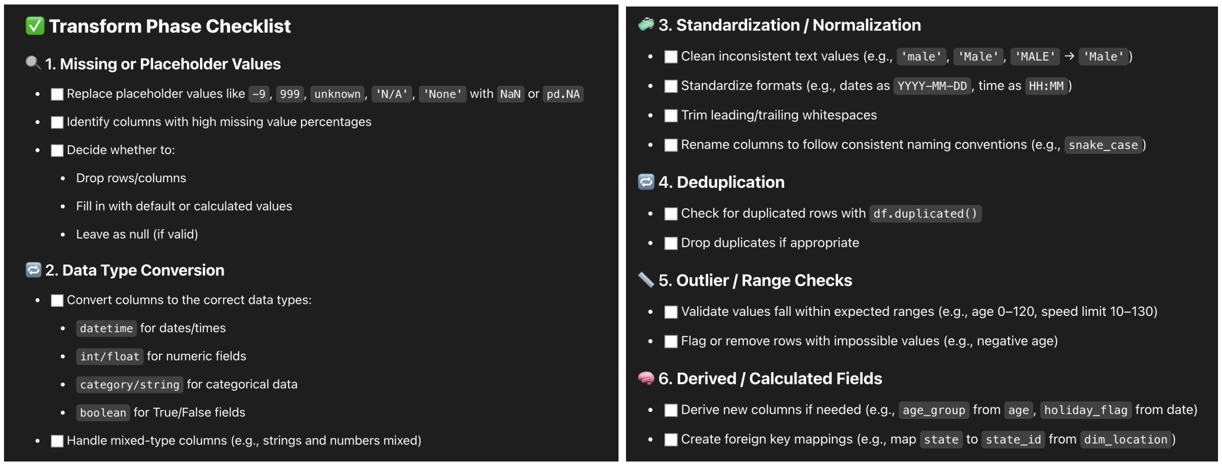
Relevant Dimensions:

- Demographics Dimension: Gender
- Date Dimension: Month
- Remoteness Dimension: Remoteness Area (filtered to "Major Cities")
- Fact Table Measure: Number of Fatalities
- Date Dimension: Year (filtered to last 10 years)

Other Dimensions: Set to All

Data Cleaning

To prepare the datasets for the ETL (Extract, Transform, Load) process, several cleaning steps were applied to ensure the data is reliable and compatible with the schema. The two primary datasets, *fatal_crashes* and *fatalities* were first read into Jupyter Notebook using pandas. These datasets were then merged on the shared Crash ID to allow for simultaneous processing and cleaning.



✅ Transform Phase Checklist

1. Missing or Placeholder Values

- ☐ Replace placeholder values like `-9`, `999`, `unknown`, `'N/A'`, `'None'` with `NaN` or `pd.NA`
- ☐ Identify columns with high missing value percentages
- ☐ Decide whether to:
  - Drop rows/columns
  - Fill in with default or calculated values
  - Leave as null (if valid)

2. Data Type Conversion

- ☐ Convert columns to the correct data types:
  - `datetime` for dates/times
  - `int/float` for numeric fields
  - `category/string` for categorical data
  - `boolean` for True/False fields
- ☐ Handle mixed-type columns (e.g., strings and numbers mixed)

3. Standardization / Normalization

- ☐ Clean inconsistent text values (e.g., `'male'`, `'Male'`, `'MALE'` → `'Male'`)
- ☐ Standardize formats (e.g., dates as `YYYY-MM-DD`, time as `HH:MM`)
- ☐ Trim leading/trailing whitespaces
- ☐ Rename columns to follow consistent naming conventions (e.g., `snake_case`)

4. Deduplication

- ☐ Check for duplicated rows with `df.duplicated()`
- ☐ Drop duplicates if appropriate

5. Outlier / Range Checks

- ☐ Validate values fall within expected ranges (e.g., age 0–120, speed limit 10–130)
- ☐ Flag or remove rows with impossible values (e.g., negative age)

6. Derived / Calculated Fields

- ☐ Derive new columns if needed (e.g., `age_group` from `age`, `holiday_flag` from date)
- ☐ Create foreign key mappings (e.g., map `state` to `state_id` from `dim_location`)

In the data cleaning process, Generative AI was consulted to identify common data quality issues and best practices. Some are used in the data cleaning, such as missing values, standardization/normalization, deduplication, range checks, and derived fields.

```
# For values that have '-9', it is a missing/unknown value
crashes_fatalities = crashes_fatalities.replace([-9, '-9'], pd.NA)
# For values, specifically in remoteness, that have 'Unknown'
crashes_fatalities = crashes_fatalities.replace('Unknown', pd.NA)
# For values, specifically in road type, that have 'Undetermined'
crashes_fatalities = crashes_fatalities.replace('Undetermined', pd.NA)
# For values that are blank
crashes_fatalities = crashes_fatalities.replace('', pd.NA)
```

Values such as blanks, ‘unknown’, and ‘undetermined’ were standardized to NaN to ensure they would later be interpreted as NULL in SQL. Specifically in these datasets, the value ‘-9’ was used to represent missing or unknown data, and it was converted accordingly.

```
crashes_fatalities['Bus Involvement'] = crashes_fatalities['Bus
Involvement'].replace({'Yes': True, 'No': False})
```

```
crashes_fatalities['Heavy Rigid Truck Involvement'] = crashes_fatalities['Heavy
Rigid Truck Involvement'].replace({'Yes': True, 'No': False})

crashes_fatalities['Articulated Truck Involvement'] =
crashes_fatalities['Articulated Truck Involvement'].replace({'Yes': True, 'No':
False})

crashes_fatalities['Christmas Period'] = crashes_fatalities['Christmas
Period'].replace({'Yes': True, 'No': False})

crashes_fatalities['Easter Period'] = crashes_fatalities['Easter
Period'].replace({'Yes': True, 'No': False})
```

Categorical fields with values “Yes” and “No” were converted to Python Boolean types (True/False) to ensure compatibility with SQL’s boolean data type. Without this conversion, these fields would be interpreted as strings, which could cause data type inconsistencies in the database.

```
crashes_fatalities = crashes_fatalities.rename(columns={'Crash ID': 'crashkey',
                   'State': 'state_name',
                   'Month':'month_name',
                   ‘Year':'year',
                   'Dayweek':'day_of_week',
                   'Time':'time_value',
                   'Crash Type':'crash_type',
                   'Number Fatalities':'number_fatalities',
                   'Bus Involvement':'bus_involved',
                   'Heavy Rigid Truck Involvement':'heavy_rigid_truck_involved',
                   'Articulated Truck Involvement':'articulated_truck_involved',
                   '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',
                   'Day of week':'weekday',
                   'Time of Day':'time_of_day',
                   'Road User':'road_user_value',
                   'Gender':'gender_value',
                   'Age':'age_value',
                   'Age Group':'age_grp'
                   })
```

Column names were renamed to match those of the corresponding dimension tables in the SQL schema. This facilitated a smoother data load and improved schema clarity.

```
duplicates = crashes_fatalities[crashes_fatalities.duplicated()]
crashes_fatalities = crashes_fatalities.drop_duplicates()
```

Duplicate rows, particularly those with identical values across all columns, including Crash ID, were removed. These duplicates could otherwise lead to inflated counts or inaccurate analysis results.

```
invalid_speeds = crashes_fatalities[(crashes_fatalities['speed_limit'] < 5) |
(crashes_fatalities['speed_limit'] > 130)]
```

Numeric columns were validated to ensure all values fell within logical and legal boundaries. For instance, the speed_limit column was checked to confirm all values were within the valid range of 5 to 130 km/h, which corresponds to the lowest and highest permissible limits in Australia. Outliers outside this range should be flagged as invalid and removed. However, in this dataset, there were none.

```
crashes_fatalities['holiday_type'] = 'None'  # default value
crashes_fatalities.loc[crashes_fatalities['Christmas Period'] == True,
'holiday_type'] = 'Christmas'
crashes_fatalities.loc[(crashes_fatalities['Easter Period'] == True) &
(crashes_fatalities['Christmas Period'] != True), 'holiday_type'] = 'Easter'
```

New columns were created as needed to match the schema. For example, a holiday_type column was added by checking existing flags for holidays such as Easter and Christmas, allowing for clarity in the data warehouse.

```
print('States included in the data are:',
crashes_fatalities['state_name'].unique().tolist())
print('Months included in the data are:',
crashes_fatalities['month_name'].unique().tolist())
```

All categorical columns were reviewed for inconsistent entries or typos. For instance, hypothetical entries such as "January" and "Janiary" would be normalized to "January". This step ensured clean and reliable joins across dimension tables.

```
# Filter rows where NaN count is at least 6
rows_with_10_na = crashes_fatalities[na_counts >= 6]
crashes_fatalities = crashes_fatalities[crashes_fatalities.isna().sum(axis=1) < 6]
```

Rows containing more than six missing (NaN, NA, or NULL) values were dropped, as they were deemed too incomplete to be trustworthy. This filtering helped maintain data integrity.

```
# Create date_df
date_df = crashes_fatalities[["year","month_name","day_of_week"]].drop_duplicates()
d_index = range(101,101+len(date_df))
date_df["date_id"] = ["D" + str(i) for i in d_index]
date_df = date_df[["date_id","day_of_week","month_name","year"]]
```

Once cleaned, the data was split into separate DataFrames corresponding to each dimension table and the fact table. These DataFrames were then exported as CSV files and loaded into PostgreSQL to populate the final schema.

ETL Process

With the data cleaned and transformed into CSV format, the next step in the ETL (Extract, Transform, Load) pipeline is the Load phase, where the data is inserted into the previously designed data warehouse. This phase was carried out using PostgreSQL, where both the dimension tables and the fact table were created and populated.

```
CREATE TABLE dim_age (
age_id INT PRIMARY KEY,
age_value INT NULL,
age_grp VARCHAR(20) NULL
);

CREATE TABLE FactCrash (
CrashKey INT PRIMARY KEY,
DateKey INT NOT NULL,
TimeKey INT NOT NULL,
HolidayKey INT NOT NULL,
LocationKey INT NOT NULL,
RoadKey INT NOT NULL,
OutcomeKey INT NOT NULL,
VehicleKey INT NOT NULL,
GenderKey INT NULL,
PersonKey INT NOT NULL,
AgeKey INT NULL,
crash_count INTEGER,
fatality_count INTEGER,
FOREIGN KEY (DateKey) REFERENCES dim_date(date_id),
FOREIGN KEY (TimeKey) REFERENCES dim_time(time_id),
FOREIGN KEY (HolidayKey) REFERENCES dim_holiday(holiday_id),
FOREIGN KEY (LocationKey) REFERENCES dim_location(location_id),
FOREIGN KEY (RoadKey) REFERENCES dim_road(road_id),
FOREIGN KEY (OutcomeKey) REFERENCES dim_outcome(outcome_id),
FOREIGN KEY (VehicleKey) REFERENCES dim_vehicle(vehicle_id),
FOREIGN KEY (GenderKey) REFERENCES dim_gender(gender_id),
FOREIGN KEY (PersonKey) REFERENCES dim_person(person_id),
FOREIGN KEY (AgeKey) REFERENCES dim_age(age_id)
);
```

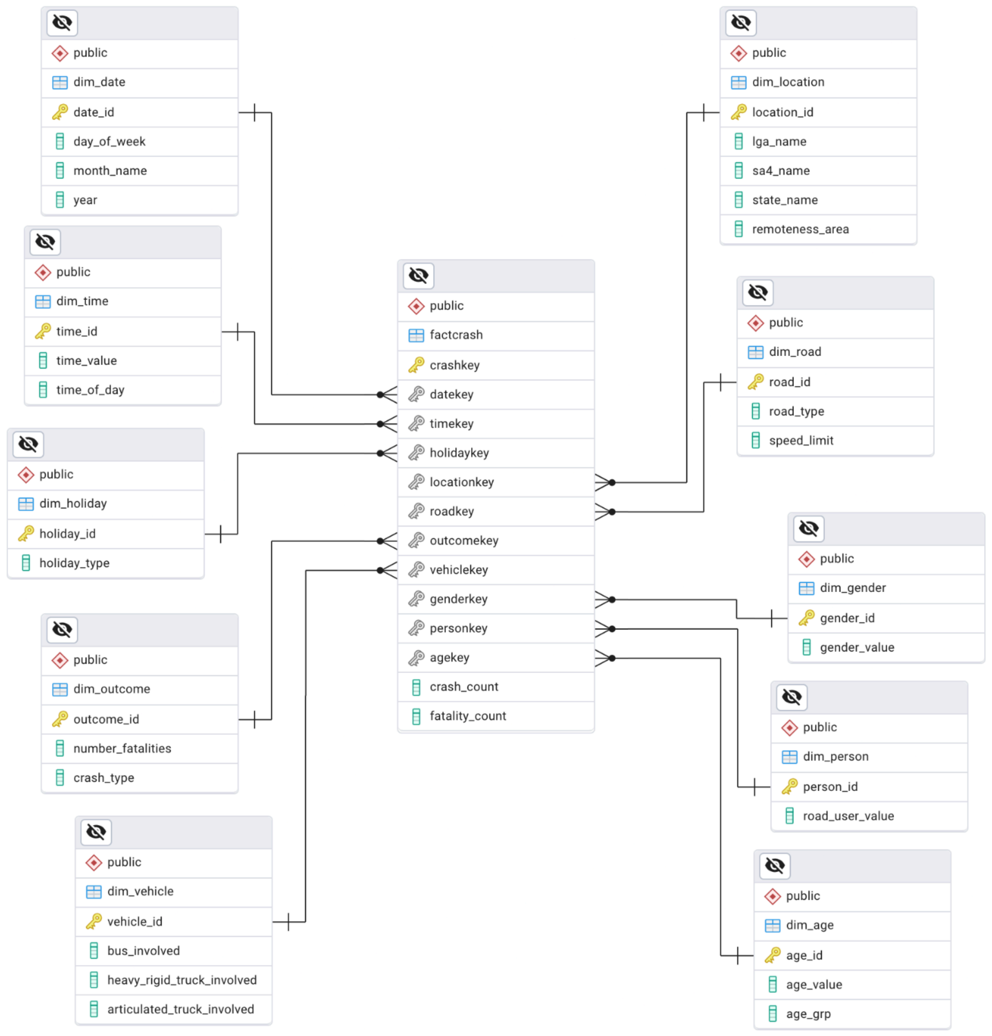
The first step was to define the schema by creating SQL tables for each dimension and the fact table. Each dimension table was created with a primary key (usually an integer ID), relevant attributes (e.g., age group, gender, vehicle type values), and proper data types to match the cleaned data. On the other hand, the fact table included a primary key, foreign keys referencing each of the ten dimension tables, and numeric measures – in this case, the number of crashes and the number of facilities associated with each crash.

```
COPY dim_vehicle
FROM '/tmp/Dim_vehicle.csv'
WITH (FORMAT csv, HEADER true);
COPY factcrash
FROM '/tmp/Fact_crashes.csv'
WITH (FORMAT csv, HEADER true);
```

After the tables were created, the cleaned CSV files generated from Jupyter Notebook were loaded into PostgreSQL. This was done using the COPY command. Similar commands were executed for the remaining dimension tables and the fact table.

Schema ERD

Following the ETL process, the dimension tables and the fact table within the data warehouse were fully populated with data. This is when we can create the ERD for the visualization of our schema. PostgreSQL can generate the ERD automatically by reading the foreign key relationships established when the table was created.

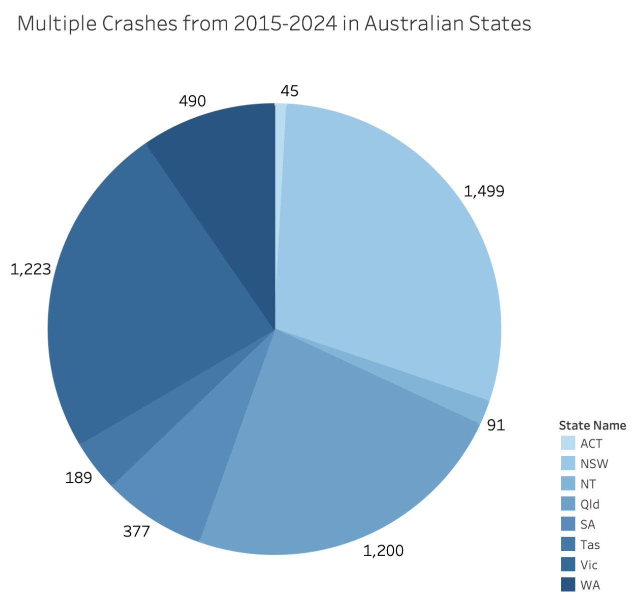


This ERD represents ***a star schema***, where the fact table connects directly to multiple dimension tables, which is better for efficiency as there are a lower number of tables. Having a low number of tables results in having fewer joins, which makes its performance faster. Star schemas are also easier to use since the numeric measures involve counting and aggregating the crash and fatalities data. Since it is more straightforward than a snowflake schema, it is also beneficial to use this when answering business queries.

Data Visualization

This section presents the visualizations used to answer the business queries formulated earlier. The following queries can be answered by:

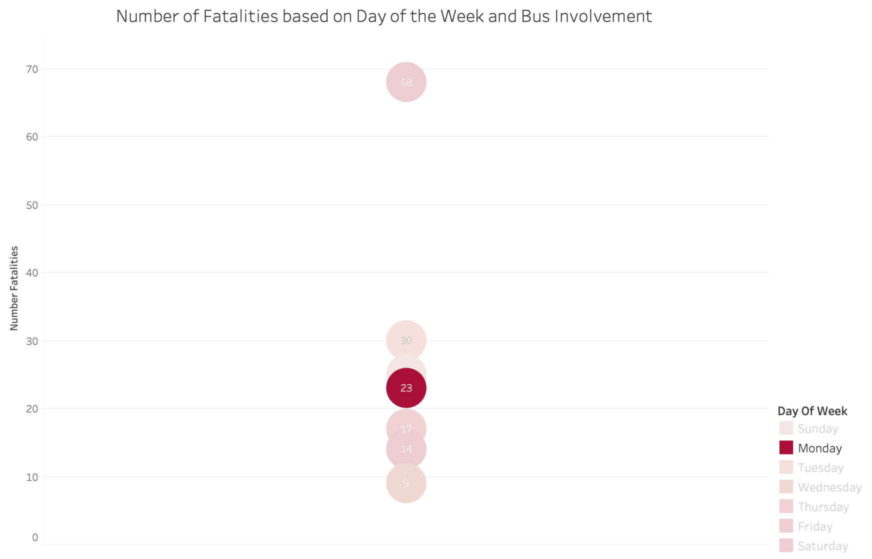
Query 1: Which state in Australia recorded the most multiple crashes in the last 10 years?



Based on this chart, it can be seen that NSW recorded the most multiple crashes in the last 10 years. Aside from that, it can also be observed how wide the gap is between the third highest value (Vic) and the fourth highest value (WA). This may tell that the rate of crashes in NSW, Qld, and Vic have been higher than desired.

A pie chart was used in this instance since they are ideal for comparing the same value across multiple categories – in this case, states. It becomes easy to see which state has the highest, and the lowest values, along with any other potential patterns that can be seen.

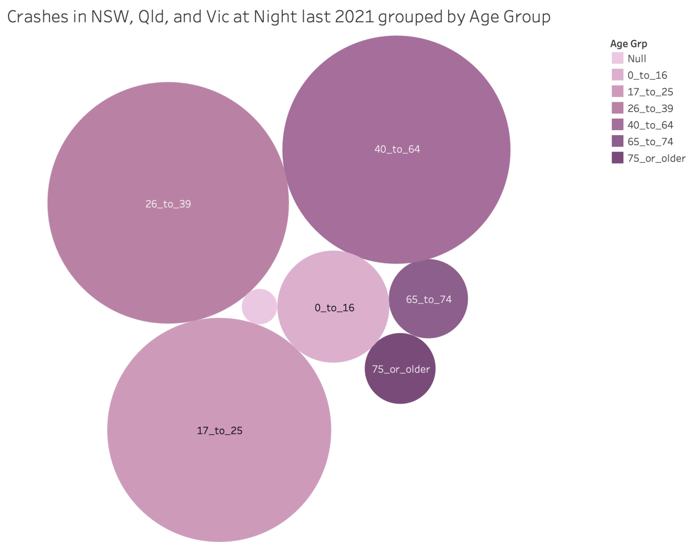
Query 2: How many fatalities occurred on a Monday crash that involved a bus?



Based on this chart, the number of fatalities that occur on a Monday crash involving a bus is 23. From this chart, it can be seen that Mondays are about average of the six days, while Saturday seems to have the highest number, by more than half of the next highest.

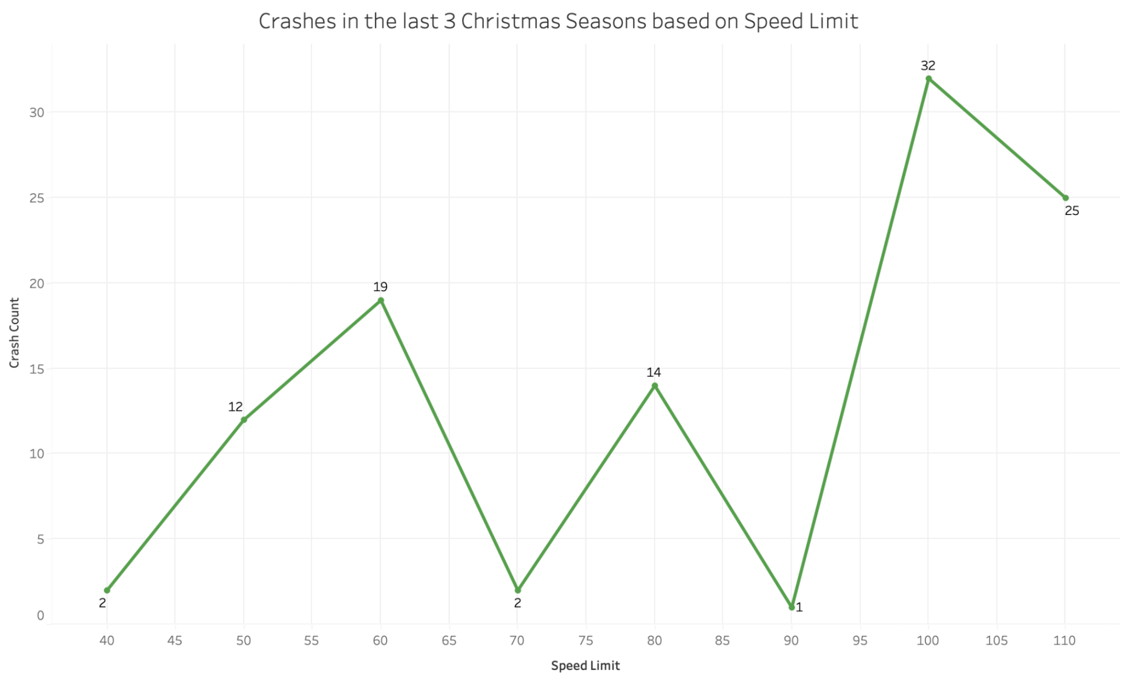
A circle chart was used here, since it is an easy-to-understand way of presenting the values for each day. The value that we are looking for is also highlighted, making it even easier to interpret.

Query 3: What age group is most involved in crashes in states NSW, Queensland, and Victoria at night time in 2021?



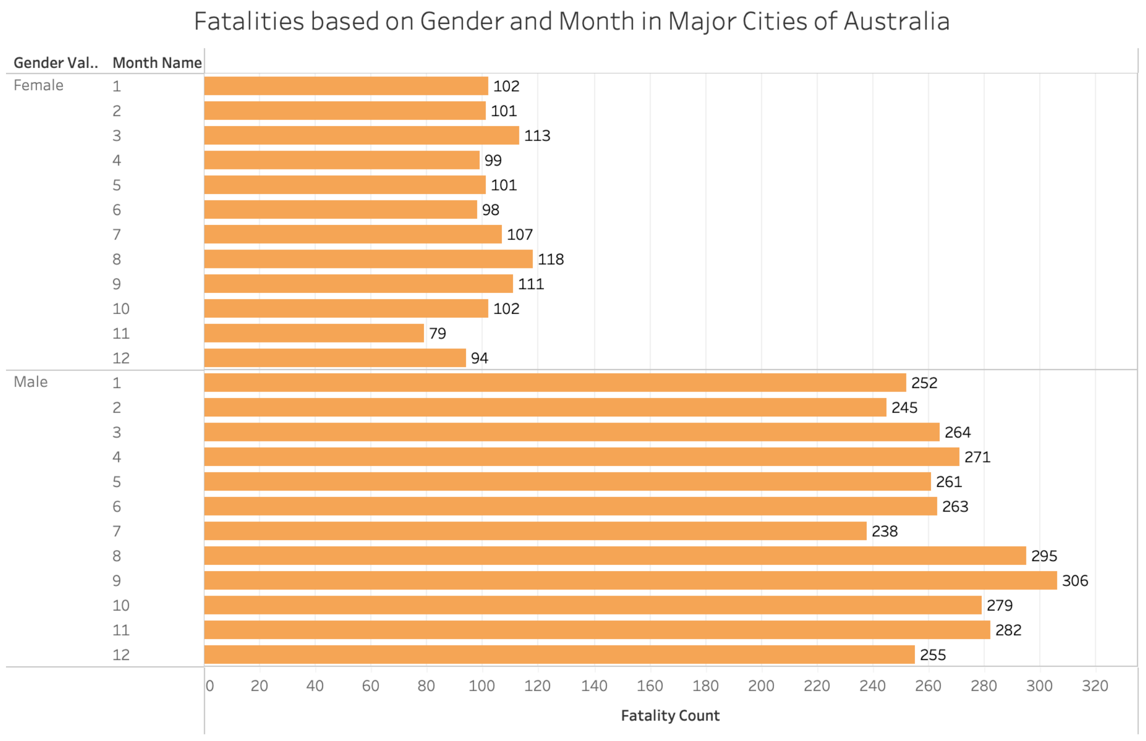
Based on this data, the age group 40-64 is most involved in crashes in the three states with the highest numbers of crashes. This could be caused by numerous factors, particularly the fact that it has the biggest age range among the age groups. Another circle chart was used, for the same reason as the previous, to present it in an easy way.

Query 4: How does the speed limit affect the number of crashes in Australia the past 3 Christmas seasons?



Based on the line chart, the speed limit with the highest number of crashes in the past three Christmas seasons is 100. The main reason a line chart was used is to explore potential trends, perhaps considering the possibility that the speed limit is proportional to the crash count. However, what it does not take into account here, is the amount of roads that have these speed limits. There is still a possibility that the majority of the roads in Australia have a speed limit of 100, and only a few have a speed limit of 90. This will result in a higher tendency for 100 speed limit roads to have higher crash counts.

Query 5: Which gender and month combination had the most fatalities in major cities of Australia?



Based on the grouped bar chart, the combination of males and the month of September records the highest fatality count in major cities of Australia. The grouped bar chart helps in comparing groups (in this case, month and gender). It can be seen from this chart that males have higher fatality counts than females in general. Aside from that, the months seem to have an inconsistent pattern, other than the fact that August and September have the two highest fatality counts among the months in both genders.

Association Rules Mining

Association rules are if-then relationships between variables in datasets, designed to help identify frequent items and patterns within a dataset. It has two parts – the antecedent (if) and the consequent (then) [6]. It can be denoted by the following equation:

$$A \rightarrow B$$

Its strength can be measured by values such as the lift and confidence. The lift is the amount of times an association rule is expected to be true, while the confidence indicates the amount of times the rule is found to be true.

In this analysis, the association rule mining algorithm that will be used is the Apriori algorithm. This algorithm runs iterations over the data to identify k-itemsets, meaning k items that frequently occur together (starting with k = 1). It then uses the k-itemsets to identify the k+1 itemsets [7]. It follows the principle, *'If an itemset is infrequent, then all the supersets of that itemset are also infrequent'*. Once there is a list of association rules, measures like lift, confidence and support can be relied on to further filter the association rules, until the rule with the strongest relationship is achieved.

Apriori algorithm is implemented with rules that have "Road User" on the right side, meaning that the consequent is supposed to be a value from the "Road User" column. This is then sorted based on the lift and confidence measures, and the top 3 rules are used to provide 3 recommendations to the government.

Recommendations Regarding Prevention of Crashes

The top 3 rules from the data association rule mining algorithm are as follows:

1. {100, Weekday} → {Driver}
2. {Male, 100} → {Driver}
3. {Single, 100} → {Driver}

Recommendation 1 – Monitor areas with 100 speed limit more

Seeing that the top 3 rules involve '100', which pertains to the speed limit, it points to the claim that a lot of the crashes can look at the relationship between the road user and the speed limit which is 100. The recommendation calls for stricter compliance with the speed limit, stopping overspeeders, and ensuring that the speed limit in those areas is correct, i.e. ensuring that the safest speed limit in those roads is indeed 100.

Recommendation 2 – Investigating more on weekdays, male-involved, and single-type crashes

Now that it can be seen that with the top 3 rules, the government could take initiative and investigate what can be similar with these three attributes. This could involve rush-hours on weekdays, possibly higher tendency for males to drive recklessly, and upping the standards with the driving skills of drivers.

Recommendation 3 – Focusing more on driving skills and procedures

Since the top 3 rules' consequent only involve the driver, this could point to drivers' skills as a whole in Australia. This could be caused by different factors, possibly the quality of the driving lessons curriculum, too much leniency to learners completing their hours, or even just the need to upgrade and add more driving protocols.

[1] Z. Kean. "What is it like not driving as an adult?" ABC News.https://www.abc.net.au/news/2023-01-18/adults-who-do-not-drive/101767786 (accessed April 14, 2025).

[2] J. Beazley. "2023 the deadliest year on Australia's roads in more than half a decade, data shows." The Guardian. https://www.theguardian.com/australia-news/2023/dec/18/2023-the-deadliest-year-on-australias-roads-in-more-than-half-a-decade-data-shows?CMP=soc_568 (accessed April 14, 2025).

[3] National Road Safety Strategy. “Fact sheet: Vision Zero and the Safe System.” National Road Safety Strategy. https://www.roadsafety.gov.au/nrss/fact-sheets/vision-zero-safe-system (accessed April 14, 2025).

[4] Kimball Group. “Four-Step Dimensional Design Process.” Kimball Group. https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dimensional-modeling-techniques/four-4-step-design-process/ (accessed April 14, 2025).

[5] ChatGPT (2025), OpenAI. Accessed April 14, 2025. [Online]. Available: https://chat.openai.com/

[6] C. Hashemi-Pour. “Association rules.” TechTarget. https://www.techtarget.com/searchbusinessanalytics/definition/association-rules-in-data-mining (accessed April 14, 2025).

[7] J. Noble. “What is the Apriori algorithm?” IBM. https://www.ibm.com/think/topics/apriori-algorithm (accessed April 14, 2025)

SQL Code

```
CREATE TABLE dim_date (
date_id INT PRIMARY KEY,
day_of_week VARCHAR(20) NULL,
month_name VARCHAR(20) NULL,
year INT NULL
);

CREATE TABLE dim_time (
time_id INT PRIMARY KEY,
time_value TIME,
time_of_day VARCHAR(20) NULL
);

CREATE TABLE dim_holiday (
holiday_id INT PRIMARY KEY,
holiday_type VARCHAR(20) NULL
);

CREATE TABLE dim_location (
location_id INT PRIMARY KEY,
lga_name VARCHAR(100) NULL,
sa4_name VARCHAR(100) NULL,
state_name VARCHAR(50) NULL,
remoteness_area VARCHAR(50) NULL
);
```

```
CREATE TABLE dim_road (
road_id INT PRIMARY KEY,
road_type VARCHAR(50) NULL,
speed_limit INT NULL
);

CREATE TABLE dim_outcome (
outcome_id INT PRIMARY KEY,
number_fatalities INT NULL,
crash_type VARCHAR(50) NULL
);

CREATE TABLE dim_vehicle (
vehicle_id INT PRIMARY KEY,
bus_involved BOOLEAN NULL,
heavy_rigid_truck_involved BOOLEAN NULL,
articulated_truck_involved BOOLEAN NULL
);

CREATE TABLE dim_gender (
gender_id INT PRIMARY KEY,
gender_value VARCHAR(20) NULL
);

CREATE TABLE dim_person (
person_id INT PRIMARY KEY,
road_user_value VARCHAR(50) NULL
);

CREATE TABLE dim_age (
age_id INT PRIMARY KEY,
age_value INT NULL,
age_grp VARCHAR(20) NULL
);

CREATE TABLE FactCrash (
CrashKey INT PRIMARY KEY,
DateKey INT NOT NULL,
TimeKey INT NOT NULL,
HolidayKey INT NOT NULL,
LocationKey INT NOT NULL,
RoadKey INT NOT NULL,
OutcomeKey INT NOT NULL,
VehicleKey INT NOT NULL,
GenderKey INT NULL,
PersonKey INT NOT NULL,
AgeKey INT NULL,
crash_count INTEGER,
fatality_count INTEGER,
FOREIGN KEY (DateKey) REFERENCES dim_date(date_id),
FOREIGN KEY (TimeKey) REFERENCES dim_time(time_id),
FOREIGN KEY (HolidayKey) REFERENCES dim_holiday(holiday_id),
FOREIGN KEY (LocationKey) REFERENCES dim_location(location_id),
FOREIGN KEY (RoadKey) REFERENCES dim_road(road_id),
FOREIGN KEY (OutcomeKey) REFERENCES dim_outcome(outcome_id),
FOREIGN KEY (VehicleKey) REFERENCES dim_vehicle(vehicle_id),
FOREIGN KEY (GenderKey) REFERENCES dim_gender(gender_id),
FOREIGN KEY (PersonKey) REFERENCES dim_person(person_id),
FOREIGN KEY (AgeKey) REFERENCES dim_age(age_id)
);
```

```
COPY dim_date
FROM '/tmp/Dim_date.csv'
WITH (FORMAT csv, HEADER true);

COPY dim_time
FROM '/tmp/Dim_time.csv'
WITH (FORMAT csv, HEADER true);

COPY dim_holiday
FROM '/tmp/Dim_holiday.csv'
WITH (FORMAT csv, HEADER true);

COPY dim_gender
FROM '/tmp/Dim_gender.csv'
WITH (FORMAT csv, HEADER true);

COPY dim_person
FROM '/tmp/Dim_person.csv'
WITH (FORMAT csv, HEADER true);

COPY dim_age
FROM '/tmp/Dim_age.csv'
WITH (FORMAT csv, HEADER true);
```

```
COPY dim_location
FROM '/tmp/Dim_location.csv'
WITH (FORMAT csv, HEADER true);

COPY dim_road
FROM '/tmp/Dim_road.csv'
WITH (FORMAT csv, HEADER true);

COPY dim_outcome
FROM '/tmp/Dim_outcome.csv'
WITH (FORMAT csv, HEADER true);

COPY dim_vehicle
FROM '/tmp/Dim_vehicle.csv'
WITH (FORMAT csv, HEADER true);

COPY factcrash
FROM '/tmp/Fact_crashes.csv'
WITH (FORMAT csv, HEADER true);
```