UNIVERSITY OF WESTERN AUSTRALIA

CITS3401 PROJECT 1

Data Warehouse Design: Fatal Crashes and Fatalities

Kennaldy Lukman
Celine Eloise Wang

11 April 2025

TABLE OF CONTENTS

I. Introduction **3**
II. Design, Implementation, and Usage of the Data Warehouse **4**
1. Design of the Data Warehouse 4
1.1 Fact Table 4
1.2 Dimension Tables 4
1.3 Concept Hierarchies 5
2. Implementation & Usage of the Data Warehouse 5
III. Schema and Starnet **6**
1. Star Schema 6
2. Starnet 7
IV. Data Cleaning, Preprocessing, and ETL Process **8**
1. Extract 8
2. Transform 9
2.3 Addressing unique values 10
2.4 Creating dimensions and fact table 13
3. Load 13
V. Visualization of Query Results **15**
1. Query Footprints 15
2. Visualizations and Insights 18
VI. Association Rules Mining **21**
1. Apriori Algorithm 21
2. Explanation of the Top K Rules with "Road User" as Consequent 23
Rule 1 23
Rule 2 23
3. Insights from the Mining Results 23
VII. Recommendations to the Government 24
3. Target Driver Behavior in Solo Crash Scenarios 25
Reference **26**

I. Introduction

Road safety remains a critical concern for nations worldwide. Every country is working hard to reduce traffic fatalities and crashes, striving to create a safer transportation network. While some other countries have made significant improvements in road safety, Australia still faces challenges in dealing with road crashes. As of the most recent data, Australia's road fatality rate has increased from past years, which stands at 4.8 deaths per 100,000 people[1].

The need to improve road safety has never been more urgent. In light of these statistics, data-driven solutions are needed to better understand the underlying causes of road fatalities and to implement preventive measures. This report outlines the creation of a data warehouse designed to analyze historical data on fatal crashes and a dashboard for visualizing key insights.

Through the use of advanced data mining techniques, combined with insights extracted from the data warehouse and visualizations, this project will support decision-making processes aimed at identifying trends, patterns, and risk factors that contribute to road accidents. Additionally, actionable recommendations for improving road safety will be proposed to help the government formulate more effective strategies.

II. Design, Implementation, and Usage of the Data Warehouse

1. Design of the Data Warehouse

1.1 Fact Table

The fact table, named “fatalities_fact” contains all the recorded crashes, including the following attributes:

- Crash_ID
- Date_ID
- Time_ID
- Location_ID
- Vehicle_ID
- Speed_ID
- Gender_ID
- Age_ID
- Road_User_ID
- Crashtype_ID
- number_fatalities

There is 1 primary key “Crash_ID”, 1 measure “number_fatalities”, and 9 foreign keys referencing 9 different dimension tables. All the attributes in the fact table mostly refer to another dimension table since it contains the description of the crash. This excludes the attribute “number_fatalities” which is a measure of the number of fatalities that the crash might have, as it describes a specific number for the crash that will be insightful for the data analysis.

1.2 Dimension Tables

As the data has a lot of attributes, it is split into 9 different dimensions, grouped by similar categories. The dimensions are made to be simple, which is why it is split into as many dimensions as possible, but still versatile, so they can be used to answer complex business queries. The dimensions that are made are vital factors that may contribute to the crash, which allows the analysis to be easier for the business queries.

- dim_date (Date_ID, Month, Year, Day, Day_Type, Christmas_Period, Easter_Period): Dimension for the date, month, year, and seasonal period in which the crash happened.
- dim_time (Time_ID, Time, Time_of_Day): Dimension for the time in which the crash happened.
- dim_location (Location_ID, State): Dimension for the location in which the crash happened.

- dim_vehicle (Vehicle_ID, Bus_Involvement, Heavy_Rigid_Truck_Involvement, Articulated_Truck_Involvement): Dimension for the type of vehicle that crashed.
- dim_speedlimit (Speed_ID, Speed_Limit): Dimension for the speed limit that was set in the area where the crash happened.
- dim_gender (Gender_ID, Gender): Dimension for the gender of the crash victim.
- dim_age (Age_ID, Age, Age_Group): Dimension for the age of the crash victim.
- dim_road_user (Road_User_ID, Road_User): Dimension for the crash victim's road user description.
- dim_crashtype (Crashtype_ID, Crash_Type): Dimension for the type of crash that happened.

1.3 Concept Hierarchies

One of the concept hierarchies included in the data warehouse is the date hierarchy, which is in dim_date.

Day < Month < Year

This date hierarchy allows drill-down or roll-up data based on time.

2. Implementation & Usage of the Data Warehouse

In order to extract insights from the data and produce data-driven solutions, some business questions are created to better understand the data warehouse. These questions will be able to analyze the data warehouse from a lot of factors and utilize the usage of the data warehouse maximally.

1. Which road user is most likely to be involved in multiple fatal crashes?
2. Is the pattern between road crashes and speed limit the same for each state?
3. How many crashes and fatalities have been caused by each female & male driver & passenger over the years (especially in 2021)?
4. Are there timing patterns of road fatalities (time of the day & day type)?
5. Which specific group of people (age and gender) has the biggest number of road fatalities?

III. Schema and Starnet

1. Star Schema

The data warehouse's model can be made with a star schema, with fatalities_fact as the fact table in the center, linked with 9 different dimension tables for easier and more efficient querying, especially with OLAP (Online Analytical Processing).

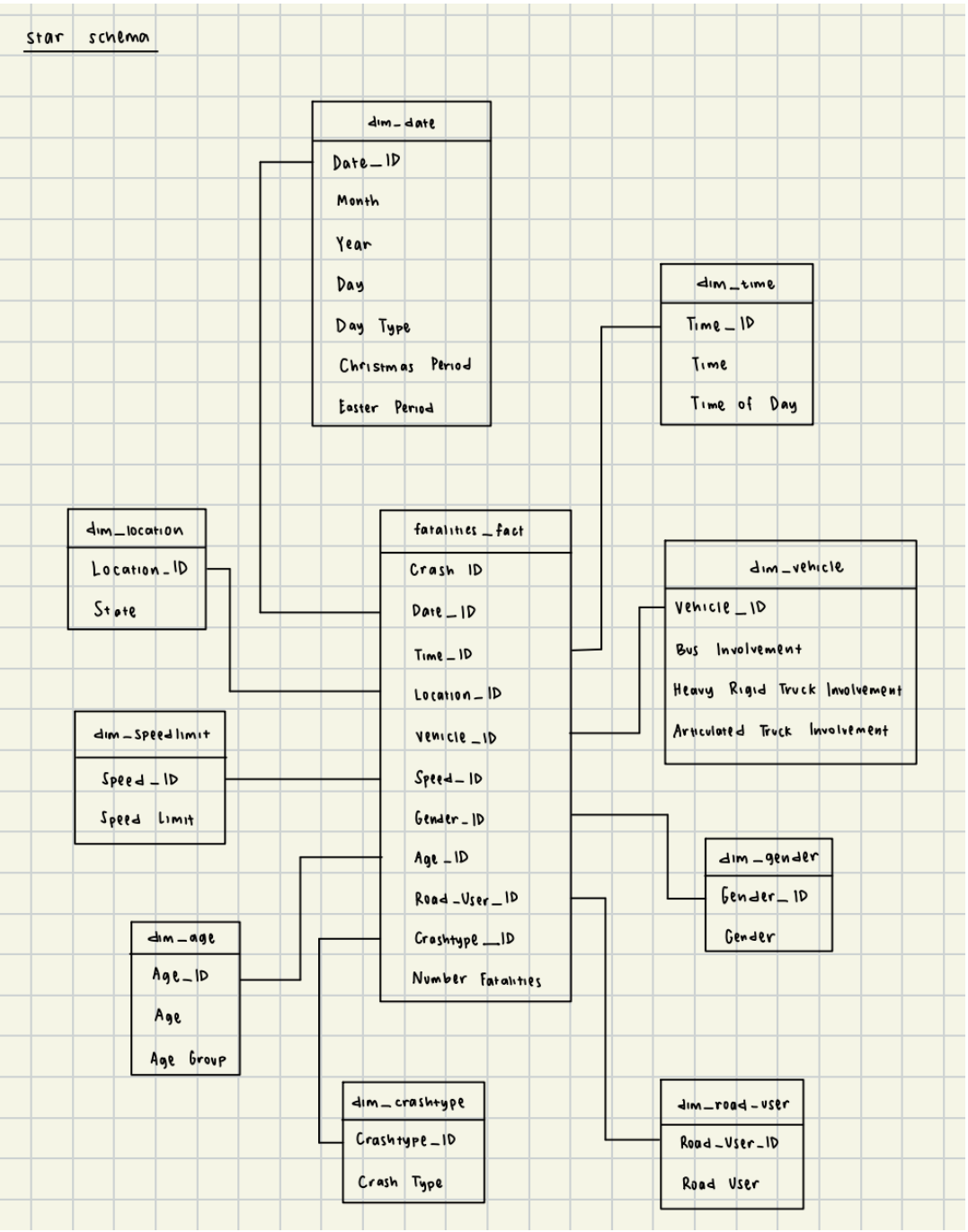

Fig 3.1 Star schema of the data warehouse

2. Starnet

To answer business queries more accurately, the star schema can be broken down into a starnet below with concept hierarchies that have been made. By breaking it down into 9 dimensions and 1 fact table, it allows the data warehouse to be able to produce queries that can involve a lot of factors, such as location, date, time, and so on.

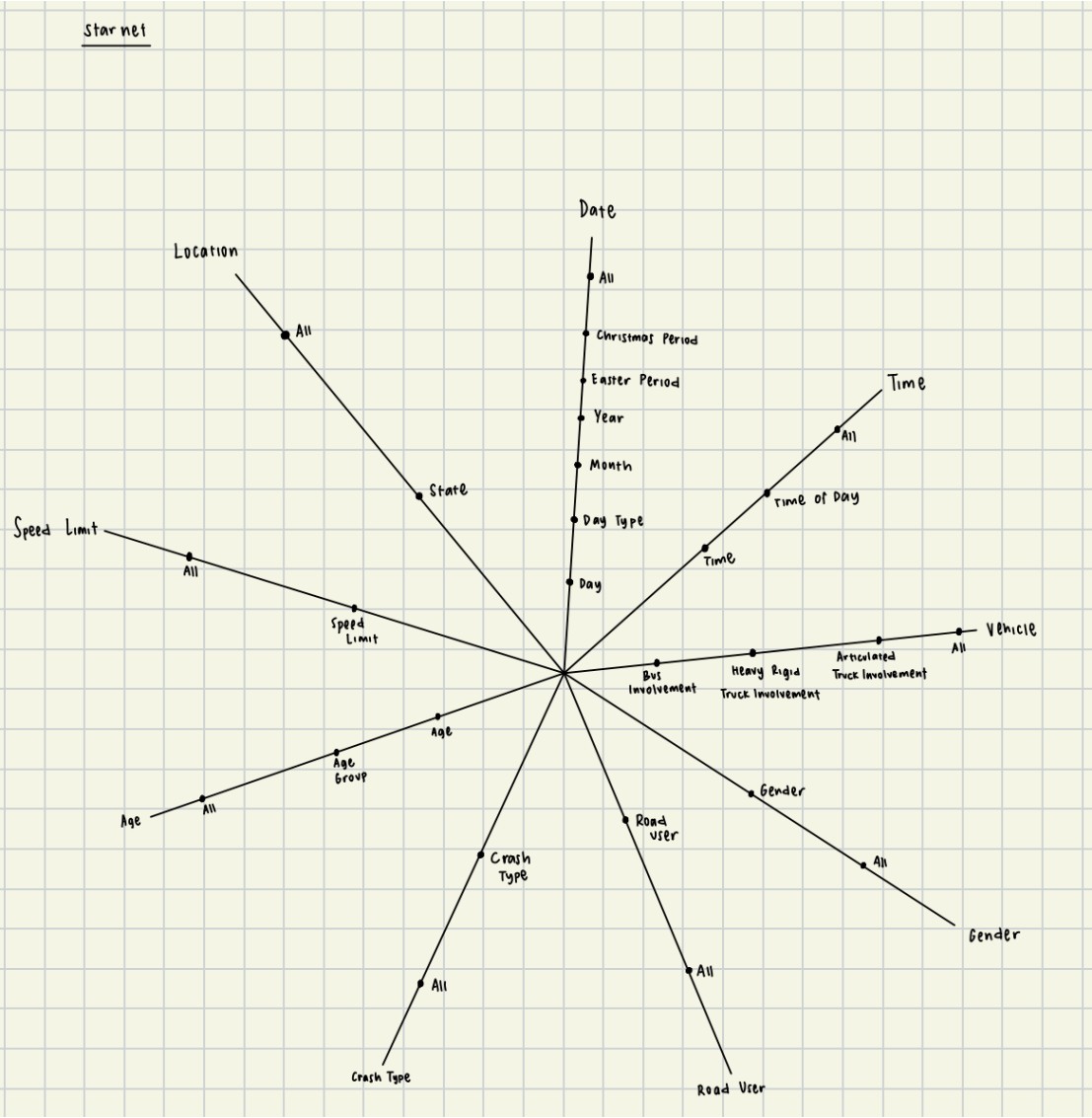

Fig 3.2 Starnet of the data warehouse

Dividing the attributes and grouping them based on similarity makes the concept hierarchy clearer and can produce more effective queries. For example, the concept hierarchy day-month-year groups those attributes into a dimension table “dim_date”, and attributes like “Bus Involvement” and “Heavy Rigid Truck Involvement” can be grouped into one dimension table “dim_vehicle” as they contain a similar description of the crash. That is why the data warehouse is broken down into 1 fact table and 9 dimension tables.

IV. Data Cleaning, Preprocessing, and ETL Process

1. Extract

From the two datasets – Fatal Crashes - December 2024[2] dataset and the Fatalities - December 2024[3] dataset, the data is downloaded and converted to CSV format, then read in Python, and merged based on similar attributes.

```python
# Reading the fatal_crashes_2024 dataset
df_fatal_crashes = pd.read_csv("basedata/bitre_fatal_crashes_dec2024_csv.csv")
df_fatal_crashes.head()
```

✓ 0.1s Python

	Crash ID	State	Month	Year	Dayweek	Time	Crash Type	Number Fatalities	Bus Involvement	Heavy Rigid Truck Involvement	Articulated Truck Involvement	Speed Limit	National Remoteness Areas	SA4 Name 2021	National LGA Name 2021
0	20241115	NSW	12	2024	Friday	04:00	Single	1	No	No	No	100	Inner Regional Australia	Riverina	Wagga Wagga
1	20241125	NSW	12	2024	Friday	06:15	Single	1	No	No	No	80	Inner Regional Australia	Sydney - Baulkham Hills and Hawkesbury	Hawkesbury
2	20246013	Tas	12	2024	Friday	09:43	Multiple	1	No	No	No	50	Inner Regional Australia	Launceston and North East	Northern Midlands
3	20241002	NSW	12	2024	Friday	10:35	Multiple	1	No	No	No	100	Outer Regional Australia	New England and North West	Armidale Regional
4	20242261	Vic	12	2024	Friday	11:30	Multiple	1	-9	-9	-9	-9	Unknown	NaN	NaN

Fig 4.1 Top: Extracting data from the Fatal Crashes - December 2024 dataset. Bottom: Extracting data from the Fatalities - December 2024 dataset.

```python
# reading the fatalities_2024 dataset
df_fatalities = pd.read_csv("basedata/bitre_fatalities_dec2024_csv.csv")
df_fatalities = df_fatalities.rename(columns={'Time of day': 'Time of Day'}) # renaming a column
df_fatalities.head()
```

✓ 0.1s Python

	Crash ID	State	Month	Year	Dayweek	Time	Crash Type	Bus Involvement	Heavy Rigid Truck Involvement	Articulated Truck Involvement	...	Age	National Remoteness Areas	SA4 Name 2021	National LGA Name 2021	Natic Road T
0	20241115	NSW	12	2024	Friday	04:00	Single	No	No	No	...	74	Inner Regional Australia	Riverina	Wagga Wagga	Arterial R
1	20241125	NSW	12	2024	Friday	06:15	Single	No	No	No	...	19	Inner Regional Australia	Sydney - Baulkham Hills and Hawkesbury	Hawkesbury	Local R
2	20246013	Tas	12	2024	Friday	09:43	Multiple	No	No	No	...	33	Inner Regional Australia	Launceston and North East	Northern Midlands	Local R
3	20241002	NSW	12	2024	Friday	10:35	Multiple	No	No	No	...	32	Outer Regional Australia	New England and North West	Armidale Regional	Nationa State High
4	20242261	Vic	12	2024	Friday	11:30	Multiple	-9	-9	-9	...	62	Unknown	NaN	NaN	Undetermi

5 rows × 23 columns

Finding common columns and columns unique to first and second dataset

```
cols_fatal_crashes = set(df_fatal_crashes.columns) # all columns in fatal_crashes dataset
cols_fatalities = set(df_fatalities.columns) # all columns in fatalities dataset
only_fatal_crashes = cols_fatal_crashes - cols_fatalities # columns only found in fatal_crashes
only_fatalities = cols_fatalities - cols_fatal_crashes # columns only found in fatalities
common_cols = cols_fatal_crashes & cols_fatalities # columns found in both

print(only_fatal_crashes)
print(only_fatalities)
print(common_cols)
```

```
{'Number Fatalities'}
{'Age', 'Road User', 'Gender', 'Age Group'}
{'Bus Involvement', 'Time of Day', 'SA4 Name 2021', 'Christmas Period', 'National Road Type', 'Crash ID', 'Year', 'National Remoteness Areas', 'Nationa
```

Fig 4.2 Top: Finding common columns on both datasets. Bottom: Merging both datasets based on the common columns and renaming some ambiguous columns.

Combine both datasets on the common columns

```
df = df_fatalities.merge(df_fatal_crashes, on=list(common_cols), how='left')
df = df.drop_duplicates() # ensure no duplicates when merging
df.head()
```

	Crash ID	State	Month	Year	Dayweek	Time	Crash Type	Bus Involvement	Heavy Rigid Truck Involvement	Articulated Truck Involvement	...	National Remoteness Areas	SA4 Name 2021	National LGA Name 2021	National Road Type
0	20241115	NSW	12	2024	Friday	04:00	Single	No	No	No	...	Inner Regional Australia	Riverina	Wagga Wagga	Arterial Road
1	20241125	NSW	12	2024	Friday	06:15	Single	No	No	No	...	Inner Regional Australia	Sydney - Baulkham Hills and Hawkesbury	Hawkesbury	Local Road
2	20246013	Tas	12	2024	Friday	09:43	Multiple	No	No	No	...	Inner Regional Australia	Launceston and North East	Northern Midlands	Local Road
3	20241002	NSW	12	2024	Friday	10:35	Multiple	No	No	No	...	Outer Regional Australia	New England and North West	Armidale Regional	National or State Highway
4	20242261	Vic	12	2024	Friday	11:30	Multiple	-9	-9	-9	...	Unknown	NaN	NaN	Undetermined

5 rows × 24 columns

```
df = df.rename(columns={'Dayweek': 'Day', 'Day of week': 'Day Type'})
df.columns
```

```
Index(['Crash ID', 'State', 'Month', 'Year', 'Day', 'Time', 'Crash Type',
       'Bus Involvement', 'Heavy Rigid Truck Involvement',
       'Articulated Truck Involvement', 'Speed Limit', 'Road User', 'Gender',
       'Age', 'National Remoteness Areas', 'SA4 Name 2021',
       'National LGA Name 2021', 'National Road Type', 'Christmas Period',
       'Easter Period', 'Age Group', 'Day Type', 'Time of Day',
       'Number Fatalities'],
      dtype='object')
```

Any duplicates that result from the merging should be removed. Some potentially ambiguous columns, such as *Dayweek* and *Day of week*, are renamed for improved clarity.

2. Transform

Following up after extracting the data, part of the transformation from the ETL process includes data cleaning and data preprocessing. This includes:

2.1 Addressing missing values

```
df.isna().sum()
```

```
Crash ID                          0
State                             0
Month                             0
Year                              0
Day                               0
Time                             43
Crash Type                        0
Bus Involvement                   0
Heavy Rigid Truck Involvement     0
Articulated Truck Involvement     0
Speed Limit                       0
Road User                         0
Gender                            0
Age                               0
National Remoteness Areas         0
SA4 Name 2021                 44031
National LGA Name 2021        44029
National Road Type                0
Christmas Period                  0
Easter Period                     0
Age Group                         0
Day Type                          0
Time of Day                       0
Number Fatalities              3500
dtype: int64
```

The missing values in column *Time* are considered small (only 43 out of 56708 data), therefore, the best way to handle them is to drop the missing values. The columns *SA4 Name 2021* and *National LGA Name 2021* will not be relevant to the business questions that we are answering, and as such not relevant, which means it does not need to be included in our dataset. Therefore, the best choice is to delete both columns.

Fig 4.4 Showing all the missing values in the dataset

```
# deleting null values from Time
df = df.dropna(subset=['Time'])

# deleting the columns with unknowns
df = df.drop(columns=['SA4 Name 2021','National LGA Name 2021'])
df
```

	Crash ID	State	Month	Year	Day	Time	Crash Type	Bus Involvement	Heavy Rigid Truck Involvement	Articulated Truck Involvement	...	Gender	Age	National Remoteness Areas	National Road Type
0	20241115	NSW	12	2024	Friday	04:00	Single	No	No	No	...	Male	74	Inner Regional Australia	Arterial Road
1	20241125	NSW	12	2024	Friday	06:15	Single	No	No	No	...	Female	19	Inner Regional Australia	Local Road
2	20246013	Tas	12	2024	Friday	09:43	Multiple	No	No	No	...	Female	33	Inner Regional Australia	Local Road
3	20241002	NSW	12	2024	Friday	10:35	Multiple	No	No	No	...	Female	32	Outer Regional Australia	National or State Highway
4	20242261	Vic	12	2024	Friday	11:30	Multiple	-9	-9	-9	...	Male	62	Unknown	Undetermined
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
56868	19896006	Tas	1	1989	Wednesday	20:20	Multiple	No	-9	Yes	...	Female	11	Unknown	Undetermined
56869	19896006	Tas	1	1989	Wednesday	20:20	Multiple	No	-9	Yes	...	Female	13	Unknown	Undetermined

Fig 4.5 Process of dropping null values and irrelevant columns

The column *Number Fatalities* also has some missing values, which cannot be removed because they are more significant relative to the data (3500 out of 56665, after removing null values from *Time*). but these missing values resulted from joining issues when merging the datasets. Therefore, these missing values can be easily inserted using the data available.

```
# Inserting missing values from the original dataset
df['Number Fatalities'] = df['Number Fatalities'].fillna(df['Crash ID'].map(df_fatal_crashes.set_index('Crash ID')['Number Fatalities']))
print(f"Nulls in Number Fatalities: {df['Number Fatalities'].isna().sum()}")
```

✓ 0.0s

```
Nulls in Number Fatalities: 0
```

Fig 4.6 Inserting data from the data available and checking for any null values in the column

2.3 Addressing unique values

The dataset still contains unique values, such as "-9", "Unknown", etc, which represent an unknown or missing value from the original datasets. These values have to be handled.

```
[174]
# Checking all unique values in dataset
for col in df.select_dtypes(exclude='number'):
    print(col, df[col].unique())
```

```
State ['NSW' 'Tas' 'Vic' 'Qld' 'SA' 'WA' 'ACT' 'NT']
Day ['Friday' 'Monday' 'Saturday' 'Sunday' 'Thursday' 'Tuesday' 'Wednesday']
Time ['04:00' '06:15' '09:43' ... '22:33' '06:57' '02:41']
Crash Type ['Single' 'Multiple']
Bus Involvement ['No' '-9' 'Yes']
Heavy Rigid Truck Involvement ['No' '-9' 'Yes']
Articulated Truck Involvement ['No' '-9' 'Yes']
Speed Limit ['100' '80' '50' '-9' '90' '60' '70' '110' '40' '20' '10' '5' '130' '30'
 '<40' '25' '15' '75' 110 100 60 80 -9 70 90 40 50 30 20 75]
National Remoteness Areas ['Inner Regional Australia' 'Outer Regional Australia' 'Unknown'
 'Major Cities of Australia' 'Very Remote Australia' 'Remote Australia']
National Road Type ['Arterial Road' 'Local Road' 'National or State Highway' 'Undetermined'
 'Sub-arterial Road' 'Collector Road' 'Pedestrian Thoroughfare'
 'Access road' 'Busway']
Christmas Period ['Yes' 'No']
Easter Period ['No' 'Yes']
Day Type ['Weekday' 'Weekend']
Time of Day ['Night' 'Day' 'Unknown']
Road User ['Driver' 'Passenger' 'Motorcycle rider' 'Pedestrian' 'Pedal cyclist'
 'Other/-9' 'Unknown' 'Motorcycle pillion passenger']
Gender ['Male' 'Female' '-9']
Age Group ['65_to_74' '17_to_25' '26_to_39' '40_to_64' '75_or_older' '0_to_16' '-9']
```

Fig 4.7 All the unique values for categorical columns.

Firstly, the column *Speed Limit* has to be handled as it contains a mix of integers and strings, including a categorical value, "<40", denoting speed limits that are below 40 km/h. The missing values will be handled by dropping them since the amount of missing values in this column is insignificant (1462 missing values out of 56665). All speed limits below 40 km/h will be converted to the value "<40" for data analysis and simplicity purposes.

```
neg_9_count = df[(df['Speed Limit'] == '-9') | (df['Speed Limit'] == -9)]['Crash ID'].count()
all_data_len = df['Crash ID'].count()
print(f"Invalid values in Speed Limit: {neg_9_count}")
print(f"Percentage: {round(neg_9_count / all_data_len * 100, 2)}%")
```

```
Invalid values in Speed Limit: 1462
Percentage: 2.58%
```

Fig 4.8 Top: Finding the number and proportion of missing values in the dataset. Bottom: Removing all data with missing values, recategorizing speed limits, and converting all data to string for data type consistency

```
df = df[(df['Speed Limit'] != -9) & (df['Speed Limit'] != "-9")] # removing all invalid values (-9)
df['Speed Limit'] = pd.to_numeric(df['Speed Limit'], errors='coerce') # if the value cannot be converted, set to null.
# functiono to replace values to "<40"
df['Speed Limit'] = df['Speed Limit'].apply(lambda x: '<=40' if (x > 0 and x <= 40) else int(x) if pd.notna(x) else x)
df['Speed Limit'] = df['Speed Limit'].fillna('<=40') # the values that cannot be converted are "<40" strings.
df['Speed Limit'] = df['Speed Limit'].astype(str) # ensure datatype consistency
df['Speed Limit'].unique()
```

```
array(['100', '80', '50', '90', '60', '70', '110', '<=40', '130', '75'],
      dtype=object)
```

The other columns are generally more straightforward. Each column will be checked for the number of missing values, and depending on the results, they will be treated appropriately. For example, columns such as *Road User* and *Gender* have an insignificant amount of missing values (103 and 29, respectively), which can be easily removed. However, columns such as *National Remoteness Areas* and *Heavy Rigid Truck Involvement* have a very large number of missing values (43952 and 1962,5 respectively)

```
road_user_invalids = df[(df['Road User'] == 'Other/-9') | (df['Road User'] == 'Unknown')]['Crash ID'].count() # Road User
bus_invalids = df[df['Bus Involvement'] == '-9']['Crash ID'].count() # Bus Involvements
heavy_rigid_invalids = df[df['Heavy Rigid Truck Involvement'] == '-9']['Crash ID'].count() # Heavy Rigid Truck
articulated_invalids = df[df['Articulated Truck Involvement'] == '-9']['Crash ID'].count() # Articulated Truck
gender_invalids = df[df['Gender'] == '-9']['Crash ID'].count() # Gender
age_groups_invalids = df[df['Age Group'] == '-9']['Crash ID'].count() # Age Group
time_of_day_invalids = df[df['Time of Day'] == 'Unknown']['Crash ID'].count() # Time of Day

print(f"Invalid values in Road User: {road_user_invalids}")
print(f"Invalid values in Bus Involvement: {bus_invalids}")
print(f"Invalid values in Heavy Rigid Truck Involvement: {heavy_rigid_invalids}")
print(f"Invalid values in Articulated Truck Involvement: {articulated_invalids}")
print(f"Invalid values in Gender: {gender_invalids}")
print(f"Invalid values in Age Groups: {age_groups_invalids}")
print(f"Invalid values in Time of Day: {time_of_day_invalids}")
```

✓ 0.0s

```
Invalid values in Road User: 103
Invalid values in Bus Involvement: 52
Invalid values in Heavy Rigid Truck Involvement: 19625
Invalid values in Articulated Truck Involvement: 46
Invalid values in Gender: 29
Invalid values in Age Groups: 86
Invalid values in Time of Day: 1
```

```
remoteness_invalids = df[df['National Remoteness Areas'] == 'Unknown']['Crash ID'].count() # National Remoteness Areas
road_type_invalids = df[df['National Road Type'] == 'Undetermined']['Crash ID'].count() # National Road Type

print(f"Invalid values in National Remoteness Areas: {remoteness_invalids}")
print(f"Invalid values in National Road Type: {road_type_invalids}")
```

```
Invalid values in National Remoteness Areas: 43952
Invalid values in National Road Type: 44556
```

Fig 4.9 Finding the number of missing values in the dataset for the other untreated columns

```
# Showing a countplot of heavy rigid truck involvement
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(data = df, x = "Heavy Rigid Truck Involvement", palette='Set2')
plt.title("Heavy Rigid Truck Involvement");
```

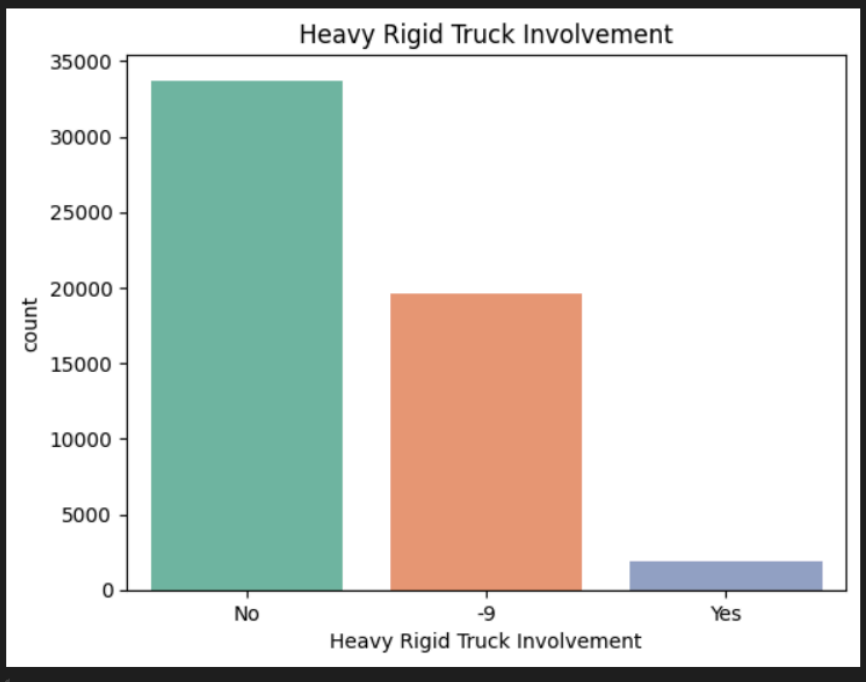

Fig 4.10 Top: Analysis of the column "Heavy Rigid Truck Involvement". Bottom: Applying the relevant changes to the dataset.

```
# Imputation of heavy rigid truck
df['Heavy Rigid Truck Involvement'] = df['Heavy Rigid Truck Involvement'].replace('-9', 'No')

# Deleting irrelevant columns
df = df.drop(columns=['National Remoteness Areas','National Road Type'])

# Handling road user
df = df[(df['Road User'] != 'Other/-9') & (df['Road User'] != 'Unknown')]

# Other columns
df = df[~df[['Bus Involvement', 'Articulated Truck Involvement', 'Gender', 'Age Group', 'Time of Day']].isin(["-9","Unknown"]).any(axis=1)]
```

The columns *National Remoteness Area* and *National Road Type* contain too many unique values to be deleted. These columns may also not be relevant to our data warehouse. Therefore, the columns will be removed. However, the same treatment is not applicable for the column *Heavy Rigid Truck Involvement*, as this will be relevant. From our analysis above, the mode of the column is "No". It can also be assumed that in the cases of these unknowns, trucks are most likely not involved. Therefore, the null values will be imputed with the value "No".

2.4 Creating dimensions and fact table

After making sure the data cleaning is done, the data is broken down into dimension and fact tables, which will be then converted to CSV files to be loaded into our data warehouse later on. The design for these tables have been discussed in Section 3.

```
# Date dimension
dim_date = df[['Month','Year','Day','Day Type','Christmas Period','Easter Period']].drop_duplicates().reset_index(drop=True)
dim_date['Date_ID'] = dim_date.index + 1 # creating ID
dim_date = dim_date[[dim_date.columns[-1]] + dim_date.columns[:-1].tolist()] # moving ID to first col
print(dim_date)
dim_date.to_csv('warehouse/dim_date.csv', index=False)

# Fact table for date
fact_df = df.merge(dim_date, how='left', on=['Month','Year','Day','Day Type','Christmas Period','Easter Period'])
fact_df = fact_df.drop(columns=['Month','Year','Day','Day Type','Christmas Period','Easter Period'])
fact_df
```

Fig 4.11 Creating the dimension tables, and creating the fact table.

The dimension tables are created by selecting features from the main dataset, removing duplicates, creating a surrogate key, and then exporting it into a CSV file. In the above example, for the *Date* dimension table, we take the features *Month, Year, Day, Day Type, Christmas Period,* and *Easter Period* from the dataset, removing the duplicates, and create the surrogate key *Date_ID*. This process is repeated for all other dimension tables. As the dimension tables are being created, it will be continuously joined with the main dataset to create the fact table.

```
# Finalising the fact table
fact_df = fact_df.drop(columns='Crash ID') # drop original crash ID
fact_df = fact_df[fact_df.columns[1:].tolist() + [fact_df.columns[0]]] # fact moved to last column
fact_df = fact_df.rename(columns={'Number Fatalities':'Fatalities'})
fact_df['Crash_ID'] = fact_df.index + 1
fact_df = fact_df[[fact_df.columns[-1]] + fact_df.columns[:-1].tolist()] # crashID moved to first column
fact_df.to_csv('warehouse/fatalities_fact.csv', index=False)
fact_df
```

Fig 4.12 Finalizing the fact table after creation of the last dimension table

After some additional cleanups to the fact table, such as replacing the original *Crash_ID* with a surrogate key and renaming the *Fatalities* column, the fact table is then exported as a CSV file.

3. Load

In order to load the data into a data warehouse, the dimension tables and fact table need to be created.

```
-- Create a dimension table for date
CREATE TABLE dim_date (
        Date_ID INT PRIMARY KEY,
        Month VARCHAR(20),
        Year INT,
        Day VARCHAR(20),
```

```
    Day_Type VARCHAR(20),
    Christmas_Period VARCHAR(5),
    Easter_Period VARCHAR(5)
```

This process is repeated for all 9 dimension tables, with the relevant attributes.

```
-- Create fact table, link the primary keys from dimension tables as foreign keys in the fact table
CREATE TABLE fatalities_fact (
    Crash_ID INT PRIMARY KEY,
    Date_ID INT NOT NULL,
    Time_ID INT NOT NULL,
    Location_ID INT NOT NULL,
    Vehicle_ID INT NOT NULL,
    Speed_ID INT NOT NULL,
    Gender_ID INT NOT NULL,
    Age_ID INT NOT NULL,
    Road_User_ID INT NOT NULL,
    Crashtype_ID INT NOT NULL,
    FOREIGN KEY (Date_ID) REFERENCES dim_date(Date_ID),
    FOREIGN KEY (Time_ID) REFERENCES dim_time(Time_ID),
    FOREIGN KEY (Location_ID) REFERENCES dim_location(Location_ID),
    FOREIGN KEY (Vehicle_ID) REFERENCES dim_vehicle(Vehicle_ID),
    FOREIGN KEY (Speed_ID) REFERENCES dim_speedlimit(Speed_ID),
    FOREIGN KEY (Gender_ID) REFERENCES dim_gender(Gender_ID),
    FOREIGN KEY (Age_ID) REFERENCES dim_age(Age_ID),
    FOREIGN KEY (Road_User_ID) REFERENCES dim_road_user(Road_User_ID),
    FOREIGN KEY (Crashtype_ID) REFERENCES dim_crashtype(Crashtype_ID),
    Number_Fatalities INT NULL
);
```

The above is the query used to create the fact table. Afterwards, the data warehouse is populated with the CSV files.

```
-- Insert data to dim_date
COPY dim_date FROM '<path>/dim_date.csv' WITH (FORMAT csv, HEADER true)
```

V. Visualization of Query Results

1. Query Footprints

Business questions:

a. Which road user is most likely to be involved in multiple fatal crashes?

```
SELECT road_user_id, SUM(number_fatalities) AS “Number_of_Deaths”
FROM fatalities_fact
WHERE crashtype_id = (SELECT crashtype_id FROM dim_crashtype WHERE crash_type = 'Multiple')
GROUP BY road_user_id
ORDER BY road_user_id ASC;
```

b. Is the pattern between road crashes and speed limit the same for each state?

```
SELECT location_id, speed_id, COUNT(crash_id) AS "Number_of_Crashes"
FROM fatalities_fact
GROUP BY CUBE(speed_id, location_id)
ORDER BY location_id, speed_id ASC;
```

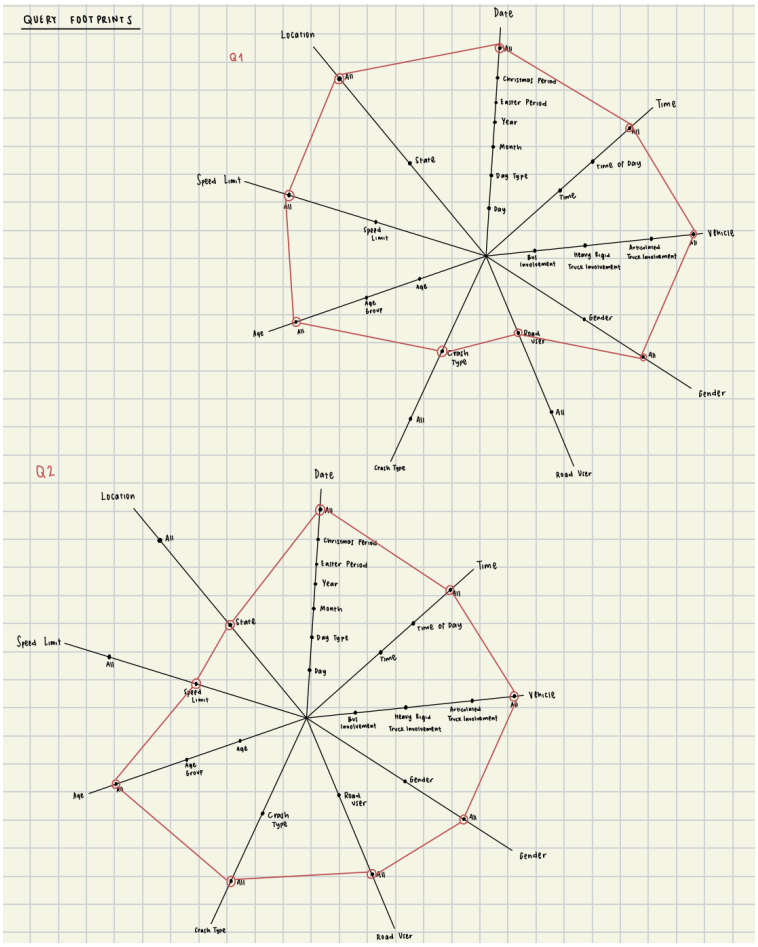

Fig 5.1 Starnet for business queries (a) and (b)

c. How many crashes and fatalities have been caused by each female & male driver & passenger over the years (especially in 2021)?

```
SELECT COUNT(crash_id) AS “Crashes”, SUM(number_fatalities) AS “Deaths”
FROM fatalities_fact
WHERE date_id IN (SELECT date_id FROM dim_date WHERE year = '2021')
AND road_user_id IN (SELECT road_user_id FROM dim_road_user WHERE road_user = 'Driver')
AND gender_id IN (SELECT gender_id FROM dim_gender WHERE gender = 'Female');
```

d. Are there timing patterns of road fatalities (time of the day & day type)?

```
SELECT AVG(number_fatalities) AS "Deaths"
FROM fatalities_fact
WHERE date_id IN (SELECT date_id FROM dim_date WHERE day_type = 'Weekday')
AND time_id IN (SELECT time_id FROM dim_time WHERE time_of_day = 'Day');
```

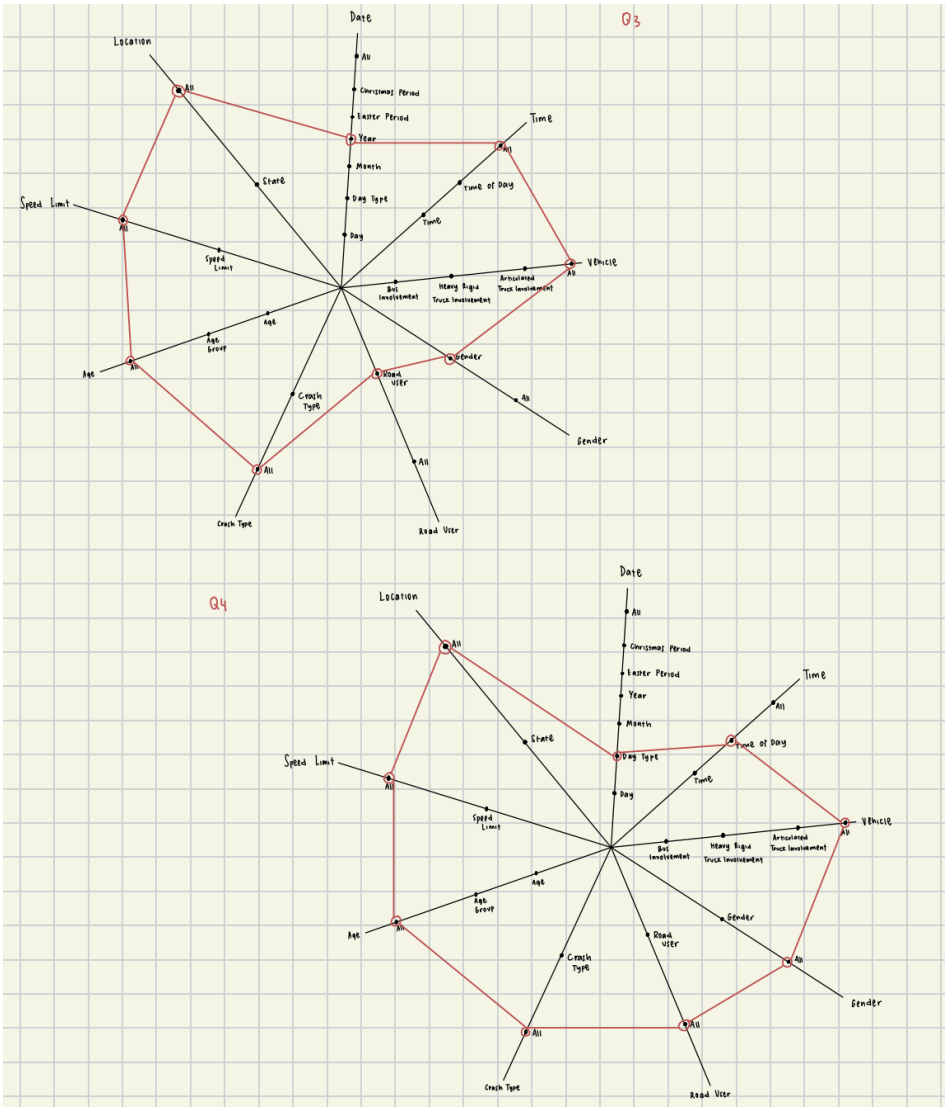

Fig 5.2 Starnet for business queries (d) and (e)

e. Which specific group of people (age and gender) has the biggest number of road fatalities?

```
SELECT age_id, gender_id, number_fatalities
FROM fatalities_fact
GROUP BY (age_id, gender_id, number_fatalities)
ORDER BY number_fatalities DESC
LIMIT 1;
```

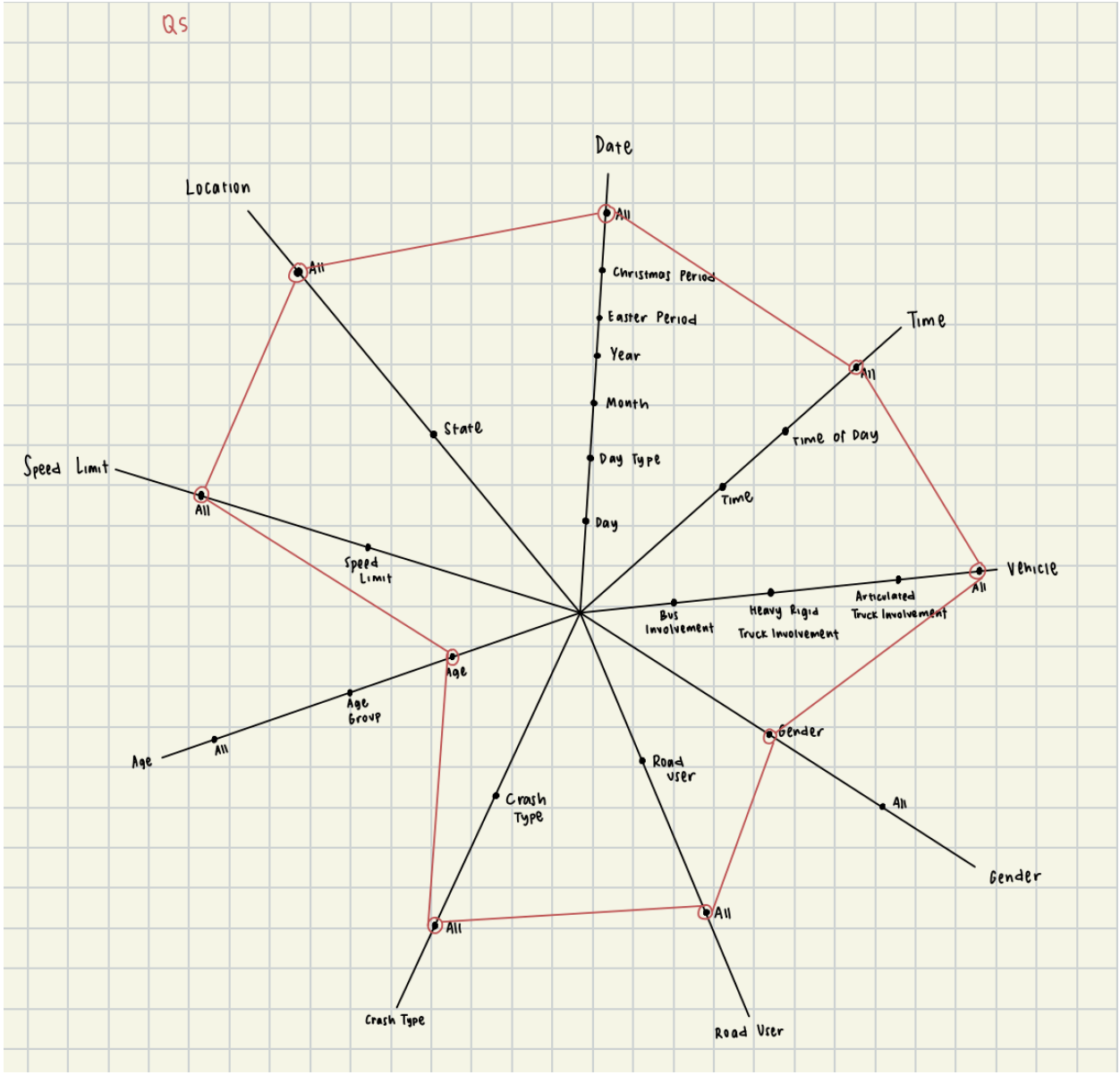

Fig 5.3 Starnet for business queries (e)

2. Visualizations and Insights

a. Which road user is most likely to be involved in multiple crashes?

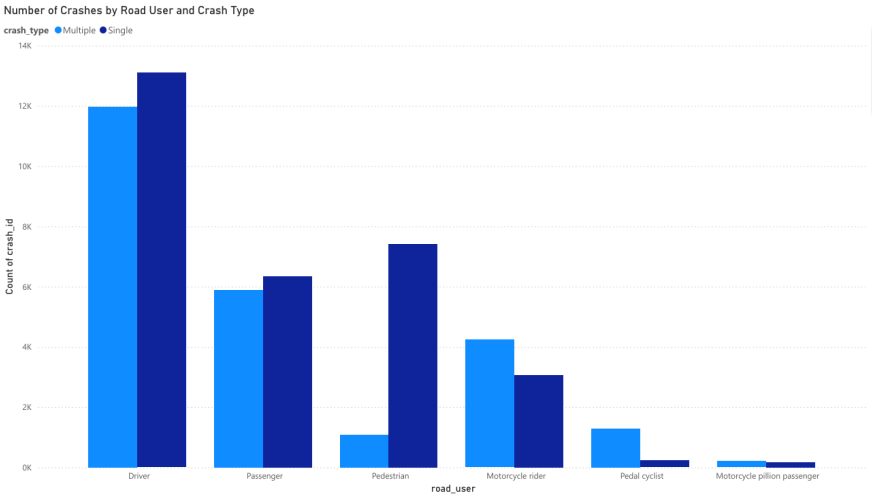

Fig 5.4 Bar plot of number of crashes grouped by Crash Type and Road User

From this graph, it can be concluded that although drivers and passengers are most involved in crashes, pedestrians are also impacted heavily in road crashes (8.488). Drivers (25.055), passengers (12.226), and pedestrians (8488) are the road users that have the most number of crashes. Pedestrians (7411) also have a larger number of single-type crashes compared to passengers (6336).

b. Is the pattern between road crashes and speed limit the same for each state?

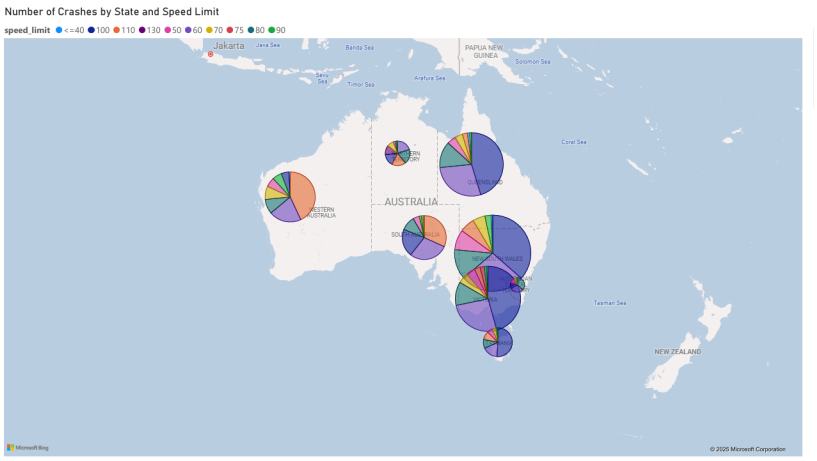

Fig 5.5 Map of Australia, plotted by number of crashes depending on the speed limit

This chart shows that the crashes that happened based on every speed limit vary for each state. For states in the east coast of Australia (NSW, VIC, QLD), crashes tend to be caused by 100 km/hr and 60 km/hr speed limits. Whereas for the rest of the states (WA, NT, SA), the number of crashes varies for every speed limit.

c. How many crashes and fatalities have been caused by each female & male driver & passenger over the years?

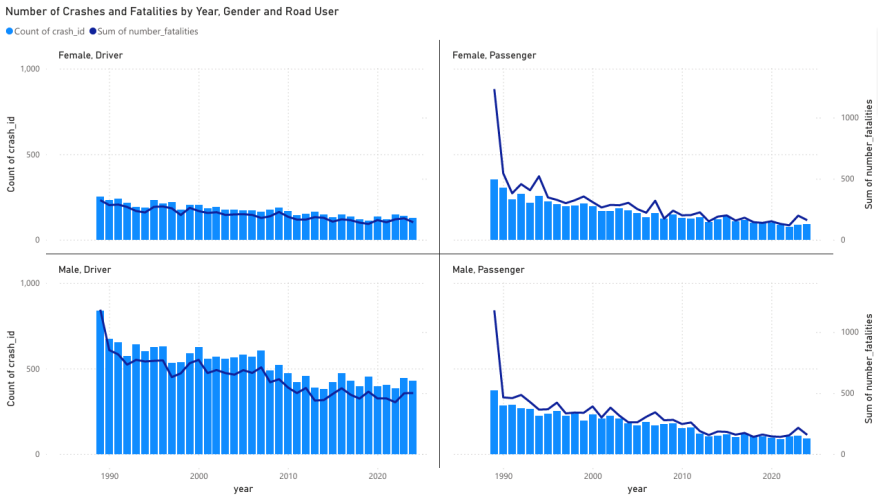

Fig 5.6 Crashes and fatalities by year, gender, and road user

The graph shows that male drivers are more likely to be involved in crashes. Male drivers may make up most of the drivers, but based on the bar chart, it can be concluded that crashes caused by male drivers have a higher chance of being fatal compared to female drivers. On the other hand, the trend across genders is generally similar for passengers who are victims of the crash.

d. Are there timing patterns of road fatalities (time of the day & day type)?

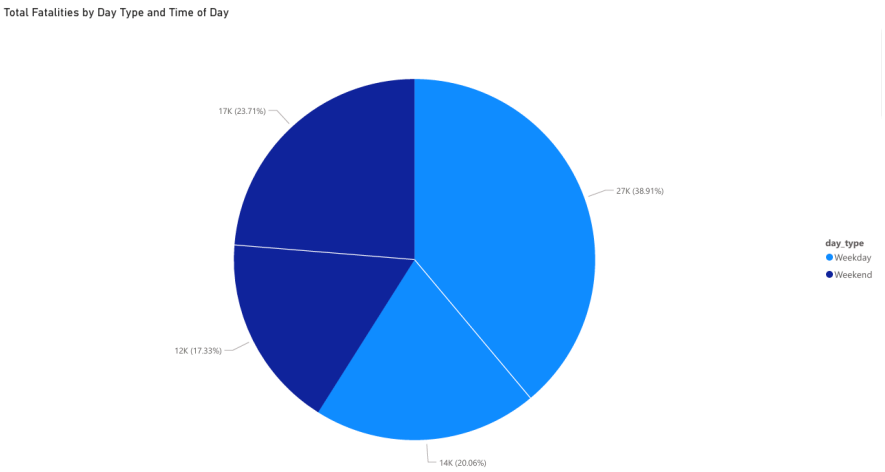

Fig 5.7 Pie chart of total fatalities by day type and time of day

The pie chart shows that crashes tend to happen in the daytime for weekdays (38.91%), but nighttime for weekends (23.71%). Though weekdays are longer than weekends, the pie chart still shows a big proportion of crashes happening on weekends (41.04%).

e. Which specific group of people (age and gender) has the biggest number of road fatalities?

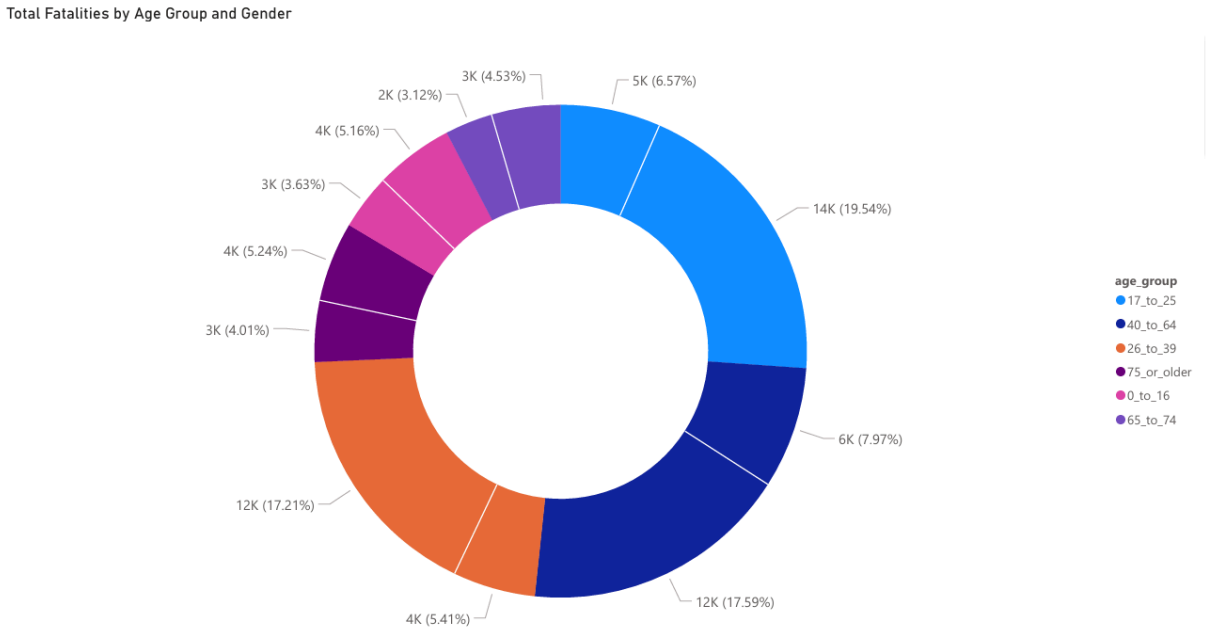

Fig 5.8 Chart showing total fatalities by age group and gender

From the chart given, it can be concluded that the 2 age groups that have the biggest number of road crashes are people aged 17-25 (26.11%) and 40-64 (25.56%). Among all age groups, half of them (age group 17-25, 40-64, 26-39) consist of mostly male drivers, whereas the rest (age group 75-older, 0-16, 65-74) have a more or less equal proportion of female and male drivers.

VI. Association Rules Mining

1. Apriori Algorithm

In this project, the **Apriori algorithm**[4] is used, which is a classic algorithm in data mining used for mining frequent itemsets and learning association rules. The Apriori algorithm works by first identifying any frequent individual items in the dataset. In this case, some examples would be *Speed Limit=100, Road User=Driver*. However, Apriori can only process transactional data, as it operates on the principle of item presence and absence. Therefore, the data has to be transformed with one-hot encoding for it to be compatible with Apriori. Afterwards, the algorithm then extends to larger itemsets as long as their frequency is above a certain threshold - a support threshold. The support threshold for this project is set at 0.2 (20%). The algorithm will then generate association rules from these frequent itemsets, and these have to satisfy a minimum confidence threshold. The minimum confidence threshold for the project is set at 0.1 (10%). The task also requires filtering for rules where "Road User" is the consequent and sorting by lift and confidence. The lift score assesses the degree to which the occurrence of an item also affects the occurrence of another.

```
# transforming data
df_str = df.astype(str).apply(lambda x: x.name + '=' + x) # TransactionEncoder can only handle string types
df_list = df_str.values.tolist() # Convert values in dataframe to list

te = TransactionEncoder()
array_te = te.fit(df_list).transform(df_list) # Convert values through one-hot encoding

transformed_df = pd.DataFrame(array_te, columns=te.columns_)
transformed_df.head()
```

✓ 0.5s

	Age Group=0_to_16	Age Group=17_to_25	Age Group=26_to_39	Age Group=40_to_64	Age Group=65_to_74	Age Group=75_or_older	Age=0	Age=1	Age=10	Age=100	...
0	False	False	False	False	True	False	False	False	False	False	...
1	False	True	False	False	False	False	False	False	False	False	...
2	False	False	True	False	False	False	False	False	False	False	...
3	False	False	True	False	False	False	False	False	False	False	...
4	False	False	False	True	False	False	False	False	False	False	...

5 rows × 1638 columns

Fig 6.1 Results of transforming the data with one-hot encoding

```
# Find the frequent itemsets
frequent_itemsets = apriori(transformed_df,min_support=0.2,use_colnames =True)
# Check the length of rules
frequent_itemsets['length']=frequent_itemsets['itemsets'].apply(lambda x: len(x))

frequent_itemsets
```

✓ 21.3s

	support	itemsets	length
0	0.255265	(Age Group=17_to_25)	1
1	0.232440	(Age Group=26_to_39)	1
2	0.259269	(Age Group=40_to_64)	1
3	0.899033	(Articulated Truck Involvement=No)	1
4	0.981889	(Bus Involvement=No)	1
...	...	...	...
2164	0.258942	(Bus Involvement=No, Articulated Truck Involve...	8
2165	0.228253	(Day Type=Weekend, Bus Involvement=No, Articul...	8
2166	0.227780	(Bus Involvement=No, Articulated Truck Involve...	8
2167	0.263274	(Bus Involvement=No, Articulated Truck Involve...	8
2168	0.242633	(Bus Involvement=No, Articulated Truck Involve...	8

Fig 6.2 Top: Results of frequent itemset mining with support threshold = 0.2. Bottom: results of association rule mining using minimum confidence threshold = 0.1

```
rules = association_rules(frequent_itemsets, metric='confidence', min_threshold=0.1)
road_user_rules = rules[rules['consequents'].apply(
    lambda x: any('Road User=' in item for item in x)
)]
road_user_rules.sort_values(by=['lift','confidence'], ascending=False).head(10)
```

✓ 0.4s

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
1513	(Bus Involvement=No, Speed Limit=100)	(Road User=Driver)	0.346421	0.456051	0.200022	0.577396	1.266077
1517	(Speed Limit=100)	(Bus Involvement=No, Road User=Driver)	0.351080	0.451119	0.200022	0.569732	1.262933
2438	(Easter Period=No, Speed Limit=100)	(Road User=Driver)	0.348860	0.456051	0.200386	0.574403	1.259513
277	(Speed Limit=100)	(Road User=Driver)	0.351080	0.456051	0.201478	0.573880	1.258368
2441	(Speed Limit=100)	(Road User=Driver, Easter Period=No)	0.351080	0.453740	0.200386	0.570769	1.257923
55873	(Bus Involvement=No, Articulated Truck Involve...	(Crash Type=Single, Road User=Driver)	0.711134	0.238282	0.202297	0.284471	1.193840
35052	(Bus Involvement=No, Articulated Truck Involve...	(Crash Type=Single, Road User=Driver)	0.715557	0.238282	0.203498	0.284392	1.193506
55895	(Bus Involvement=No, Articulated Truck Involve...	(Crash Type=Single, Road User=Driver, Easter P...	0.715557	0.236936	0.202297	0.282713	1.193205
40880	(Number Fatalities=1.0, Articulated Truck Invo...	(Crash Type=Single, Road User=Driver)	0.723967	0.238282	0.202534	0.279756	1.174050
55914	(Number Fatalities=1.0, Articulated Truck Invo...	(Bus Involvement=No, Crash Type=Single, Road U...	0.723967	0.238009	0.202297	0.279429	1.174024

2. Explanation of the Top K Rules with "Road User" as Consequent

Insights will be taken based on the top 10 rules ranked by lift and confidence. The results of this are shown in the screenshot above. The figure above shows that most of the rules describe variations of the same pattern. For example, rules 1-5 describe variations involving *Speed Limit=100* zones, while rules 6 onwards describe variations involving other vehicles. In this case, there are two interesting insights:

Rule 1

- **Antecedent**: *(Speed Limit=100, Bus Involvement=No)*
- **Consequent**: *(Road User=Driver)*
- **Confidence**: 57.73%
- **Lift**: 1.266

Based on our results, we can conclude that when the speed limit is 100, and no bus is involved in the crash, there is a 57.73% chance that the road user involved is a driver. The lift of 1.266 indicates a positive correlation, meaning that under these conditions, it is more likely than average that the road user type is a driver.

Rule 2

- **Antecedent**: *(Articulated Truck Involvement=No, Bus Involvement=No, Easter Period=No, Heavy Rigid Truck Involvement=No, Number Fatalities=1)*
- **Consequent**: *(Road User=Driver, Crash Type=Single)*
- **Confidence**: 28.44%
- **Lift**: 1.193

If no other vehicles are involved, with only one fatality, and it's not the Easter period, 28.44% of the fatal crashes involve a single-vehicle crash where the road user is a driver. The lift of 1.193 indicates that under these conditions, a driver is more likely to be involved in a single-vehicle crash in these scenarios.

3. Insights from the Mining Results

- High-speed areas (100 km/h**)** are a consistent factor in rules involving drivers.
- Drivers are the most likely to be involved in fatal accidents
- The Easter period is negatively associated with some of the top rules, suggesting possibly different road user patterns during holidays.
- Single-vehicle crashes also appear frequently in other top rules (e.g., articulated trucks and buses being uninvolved), implying potential driver error or loss of control.

VII. Recommendations to the Government

1. Visualization Insights

Based on the charts we have visualized, we recommend that the government:

1. **Implement Safety Measures for Pedestrians**
    - Building pedestrian bridges, underpasses and increasing more marked crosswalks and pedestrian signals at high-risk zones like intersections will help reduce pedestrian deaths.
2. **Safety Measures for Specific States**
    - There has to be improved infrastructure for high-speed zones, especially in the East Coast, where most of the accidents are high-speed accidents.
    - Across all speed limits and all Australian states, safety standards have to be maintained.
3. **Target Audience for Road Safety Campaigns**
    - Male drivers are generally most prone to accidents. Safety campaigns should be aimed to target male drivers.
    - People aged 17-25 and 40-64 years old are more prone to accidents. Campaigns should also be aimed to target these demographics, especially towards male drivers.
4. **Implementation of Timing-Related Solutions**
    - More traffic cameras should be implemented, additional lanes should be created, and traffic signals need to be optimized during weekdays to maximize safety during daytime.
    - For nighttime, more street lights should be installed, and stricter speed limits should be implemented, especially during the weekends.

2. Association Rules Mining Insights

Based on our association mining rules, we have three recommendations for the government:

1. **More Safety Campaigns and Enforcement in High-Speed Zones**
    - High-speed areas are frequently associated with fatal crashes involving drivers.
    - The government needs to implement speed management measures, such as speed enforcement and rumble strips.
    - The government also needs to conduct public awareness campaigns that target driver behavior in high-speed zones.
2. **More Active Safety Measures Outside Holiday Periods**
    - Our mining process suggests lower crash involvement during the Easter period, likely due to campaigns and enforcement efforts.
    - Any successful holiday safety strategies should be replicated throughout the non-holiday periods.
3. **Target Driver Behavior in Solo Crash Scenarios**
    - Single-vehicle crashes with no large vehicle involvement are very common, suggesting that driver error plays a significant role in preventing crashes.
    - Possibly introduce training or licensing refreshers about handling loss of control of vehicles, especially at high-speed zones.

Reference

[1] European Commission, "2023 figures show stalling progress in reducing road fatalities in too many countries," *Mobility and Transport*, Mar. 8, 2024. [Online]. Available: https://transport.ec.europa.eu/news-events/news/2023-figures-show-stalling-progress-reducing-road-fatalities-too-many-countries-2024-03-08_en.

[2] Bureau of Infrastructure, Transport and Regional Economics, "Fatal Crashes - December 2024," *Australian Government Department of Infrastructure, Transport, Regional Development and Communications*. [Online]. Available: https://www.bitre.gov.au/sites/default/files/documents/bitre_fatal_crashes_dec2024.xlsx.

[3] Bureau of Infrastructure, Transport and Regional Economics, "Fatalities - December 2024," *Australian Government Department of Infrastructure, Transport, Regional Development and Communications*. [Online]. Available: https://www.bitre.gov.au/sites/default/files/documents/bitre_fatalities_dec2024.xlsx.

[4] Agrawal, R., & Srikant, R. (1994). *Fast algorithms for mining association rules*. Proceedings of the 20th VLDB Conference. [Online]. Available: https://www.vldb.org/conf/1994/P487.PDF