CITS5504

Data Warehouse Design Report

23941282

Niu Jiasen

1.Clients and Analysis Objectives

1.1 Target Clients

The project is designed to support the following stakeholders:

- Department of Infrastructure, Transport, Regional Development and Communications (Australia): As the primary data provider, this department can use the analysis to refine road safety policies and prioritize investments.
- State and Local Transport Authorities (e.g., WA Department of Transport): To identify high-risk regions and implement localized road improvement plans.
- Road Safety Advocacy Bodies (e.g., Road Safety Council): For public awareness campaigns and educational outreach.
- Urban Planners and Traffic Engineers: To redesign traffic flow and improve road infrastructure at accident hotspots.
- Data Analytics and Policy Research Institutes: For deeper research into causality and predictive modeling.

1.2 Analysis Goals

The project aims to build an analytical data warehouse based on fatal crash and population datasets to:

- Understand spatial-temporal patterns in fatal traffic incidents.
- Build a star schema for efficient query execution.
- Apply association rule mining to uncover correlations among variables.
- Provide interactive visual analytics using Tableau.
- Generate actionable policy suggestions to reduce traffic fatalities.

2.ETL Process Overview

Data Sources

Raw datasets converted from Excel to CSV included:

- Geospatial data: remoteness_areas.csv, significant_urban_area.csv, local_government_areas.csv
- Crash data: bitre_fatal_crash.csv, bitre_fatalities.csv
- Daily counts: bitre_fatal_crash_count_by_date.csv, bitre_fatalities_count_by_date.csv

Two files (state_electoral_divisions.csv, commonwealth_electoral_divisions.csv) were excluded for being politically focused.
Unified time: This ETL process is mainly used to process Australian traffic accident data from 2001 to 2023. The goal is to build a data warehouse to achieve multi-angle analysis of traffic accidents by creating dimension tables and fact tables. This process integrates traffic accident data, population and geographic information, and enhances time analysis (such as seasonal dimensions) and data quality verification capabilities. So, there are the following steps:

• Multi-source CSV data integration (accidents, casualties, population, geography)
• Automatic cleaning and standardized field names
• Create dimensions such as seasons, dates, and population to support analysis
• Add data quality checks (missing and duplicate processing)

2.1 Extract

Load raw data from the following CSV files:

• bitre_fatal_crash.csv: accident details
• bitre_fatalities.csv: casualties details
• bitre_fatal_crash_count_by_date.csv: daily accident statistics
• bitre_fatalities_count_by_date.csv: daily casualties statistics
• local_government_areas.csv: local government area information
• remoteness_areas.csv: remote area classification information

First preprocessing:

• Create a dictionary
• Load csv file
• Delete the first 4 rows of irrelevant information
• Standardize column names (such as all lowercase, remove spaces)

• Unify data types

```python
# Load CSV files
print("Loading CSV files...")
try:
    (variable) fatal_crash: DataFrame e_fatal_crash.csv', skiprows=4, low_memory=False)
    fatal_crash = clean_column_names(fatal_crash)

    fatalities = pd.read_csv('data/bitre_fatalities.csv', skiprows=4, low_memory=False)
    fatalities = clean_column_names(fatalities)

    # Load count data
    fatal_crash_count = pd.read_csv('data/bitre_fatal_crash_count_by_date.csv', skiprows=4, low_memory=False)
    fatal_crash_count = clean_column_names(fatal_crash_count)

    fatalities_count = pd.read_csv('data/bitre_fatalities_count_by_date.csv', skiprows=4, low_memory=False)
    fatalities_count = clean_column_names(fatalities_count)
```

Then:
Create output directory if it doesn't exist
Function to replac Function to clean column names (trim whitespace and newlines)e -9 values with NaN， Function to get season from month

Then we process each table column by column to process the geolocation reference data.To enhance the geolocation data, we manually parse the complex structure of the CSV file:

Define remoteness categories and states， First determine the starting position of data for each state， Process data for each state， Look for categories near the state's starting index

```python
```python
# Process remoteness areas data
print("Processing remoteness areas data...")
remoteness_file = 'data/remoteness_areas.csv'

# Define remoteness categories and states
remoteness_categories = [
    'Major Cities',
    'Inner Regional',
    'Outer Regional',
    'Remote',
    'Very Remote'
]

states = ['NSW', 'Vic', 'Qld', 'SA', 'WA', 'Tas', 'NT', 'ACT']
```
```

2.2 Data Transformation

2.2.1 Data Cleaning

We cleaned the extracted data, including replacing missing values and standardizing the date and time format, unifying the time range 2001-2023, ensuring that Year is numeric

```
```python
# Clean data - replace -9 with NaN
fatal_crash = clean_missing_values(fatal_crash)
fatalities = clean_missing_values(fatalities)

# Standardize date and time data, and filter by year range (2001-2023)
print("Standardizing time data and filtering for years 2001-2023...")

# Ensure Year is numeric
fatal_crash['Year'] = pd.to_numeric(fatal_crash['Year'], errors='coerce')
fatalities['Year'] = pd.to_numeric(fatalities['Year'], errors='coerce')

# Filter by years 2001-2023
fatal_crash = fatal_crash[(fatal_crash['Year'] >= 2001) & (fatal_crash['Year'] <= 2023)]
fatalities = fatalities[(fatalities['Year'] >= 2001) & (fatalities['Year'] <= 2023)]

print(f"After filtering: {len(fatal_crash)} crash records and {len(fatalities)} fatality records")

# Create standardized date column
fatal_crash['Date'] = pd.to_datetime(
    fatal_crash['Year'].astype(str) + '-' +
    fatal_crash['Month'].astype(str).str.zfill(2) + '-01'
)
```
```

2.2.2 Handling missing values

Take age as an example: For missing age data, we use the median to fill in:

```
# Fix missing age values
if driver_dim['Age'].isnull().sum() > 0:
    print(f"Found {driver_dim['Age'].isnull().sum()} records with missing age values")

    # Create mapping from age group to median age
    age_group_mapping = {}

    # Calculate median age for each age group
    for age_group in driver_dim['Age Group'].dropna().unique():
        if age_group != 'Unknown' and age_group != '-9':
            median_age = driver_dim[driver_dim['Age Group'] == age_group]['Age'].median()
            age_group_mapping[age_group] = median_age

    # Fill missing ages with the median age of their age group
    driver_dim.loc[driver_dim['Age'].isnull(), 'Age'] = driver_dim.loc[driver_dim['Age'].isnull(), 'Age Group'].apply
    (
        lambda ag: age_group_mapping.get(ag, driver_dim['Age'].median()) if pd.notna(ag) and ag != 'Unknown' and ag
        != '-9' else driver_dim['Age'].median()
    )

    print(f"Fixed age values. Remaining null values: {driver_dim['Age'].isnull().sum()}")
```
```
```

2.3 Data loading

I created multiple dimension tables, and a fact table based on target customers and business queries to form a star-structured data warehouse

2.3.1 Example location dimension

Fill missing values with 'Unknown' and remove duplicate values

```
# 1. Location Dimension
print("Creating Location Dimension...")
location_dim = pd.DataFrame()
location_dim['Crash ID'] = fatal_crash['Crash ID']
location_dim['State'] = fatal_crash['State']
location_dim['National Remoteness Areas'] = fatal_crash['National Remoteness Areas']
location_dim['SA4 Name 2021'] = fatal_crash['SA4 Name 2021']
location_dim['National LGA Name 2021'] = fatal_crash['National LGA Name 2021']

# Fill missing values with 'Unknown'
location_dim['SA4 Name 2021'] = location_dim['SA4 Name 2021'].fillna('Unknown')
location_dim['National LGA Name 2021'] = location_dim['National LGA Name 2021'].fillna('Unknown')

# Remove duplicates
location_dim = location_dim.drop_duplicates()
location_dim.to_csv('output/location_dimension.csv', index=False)
print(f"Location Dimension created with {len(location_dim)} records")
```

2.3.2 Example Population Dimension

Extract unique remote regions and LGAs from accident data:

Process remote regions first

```
# 6. Enhanced Population Dimension
print("Creating Enhanced Population Dimension...")
# Create the population dimension
population_dim = pd.DataFrame()

# Extract unique remoteness areas and LGAs from crash data
unique_remoteness = fatal_crash['National Remoteness Areas'].dropna().unique()
unique_lgas = fatal_crash['National LGA Name 2021'].dropna().unique()

# Process remoteness areas first
```

Create a mapping table from the remote area of accident data to the reference data

Overall summary:

1. Fixed Column Name Issues:
    - Added a function to clean column names (trim whitespace and handle newlines)
    - Properly identified the "Bus Involvement" column which had an extra space in the name

2. Improved Data Loading:
   - Set low_memory=False to handle mixed data types
   - Added error handling and more detailed logging
3. Created Dimension Tables:
   - Location Dimension: Contains spatial information including state, remoteness areas, SA4 name, and LGA name
   - Vehicle Dimension: Contains information about vehicle involvement in crashes
   - Road Condition Dimension: Contains road-related information like speed limit and road type
   - Driver Dimension: Contains information about road users involved in crashes, including gender and age
   - Time Dimension: Contains detailed time information including month, year, day of week, time of day, and holiday periods
   - Population Dimension: Contains simplified population information based on remoteness areas by state
   - Crash Type Dimension: Contains information about the type of crash and number of fatalities
   - Fact Table: Contains the main facts linking to all dimension tables with aggregated monthly counts
4. Data Cleaning:
   - Replaced missing values (-9) with NaN
   - Removed duplicates from all dimension tables
   - Handled potential inconsistencies in column names
2. All dimension tables were successfully generated and saved to Fixed Column Name Issues:
   - Added a function to clean column names (trim whitespace and handle newlines)
   - Properly identified the "Bus Involvement" column which had an extra space in the name
3. Improved Data Loading:
   - Set low_memory=False to handle mixed data types
   - Added error handling and more detailed logging
4. Created Dimension Tables:
   - Location Dimension: Contains spatial information including state, remoteness areas, SA4 name, and LGA name
   - Vehicle Dimension: Contains information about vehicle involvement in crashes
   - Road Condition Dimension: Contains road-related information like speed limit and road type
   - Driver Dimension: Contains information about road users involved in crashes, including gender and age

- Time Dimension: Contains detailed time information including month, year, day of week, time of day, and holiday periods
- Population Dimension: Contains simplified population information based on remoteness areas by state
- Crash Type Dimension: Contains information about the type of crash and number of fatalities
- Fact Table: Contains the main facts linking to all dimension tables with aggregated monthly counts

5. Data Cleaning:
   - Replaced missing values (-9) with NaN
   - Removed duplicates from all dimension tables
   - Handled potential inconsistencies in column names

All dimension tables were successfully generated and saved to

2.3.3.data quality check

Then I performed a data quality check
1. Data quality check
• Created a comprehensive data quality check script data_quality_check.py to analyze all dimension tables
• Generated a detailed quality report, including missing value analysis, duplicate value check, and data validity verification
• The initial check found several issues:
• There was a large amount of missing SA4 and LGA data in the location dimension table (62.39%)
• 15 records in the time dimension table had incorrect time format (format "0n:an")
• 28 records in the driver dimension table were missing age data
2. Data Quality Analysis & Fixes
1. Fix time format issues:
• Created fix_time_issues.py script to handle time format issues
• Replaced non-standard time formats (such as "0n:an") with the standard format "00:00"
• Ensured that the date format is unified to YYYY-MM-DD
2. Handle missing values in location data:
• Created missing_value_analysis.py script to analyze missing value patterns
• Found that missing values are distributed in various states, but mainly concentrated in the "Unknown" remote area (92.78%)
• Filled all missing SA4 and LGA data with "Unknown" to maintain data integrity
3. Fix driver age data:
• Created fix_driver_age.py script to fix missing age data in the driver dimension table
• Analyzed the median age of each age group and used it to fill missing values
• For records without age group information, used the overall median age to fillMissing Value Heatmap:

- SA4 Name 2021 and National LGA Name 2021 show high missingness.
- Fields like Crash ID and State are mostly complete.

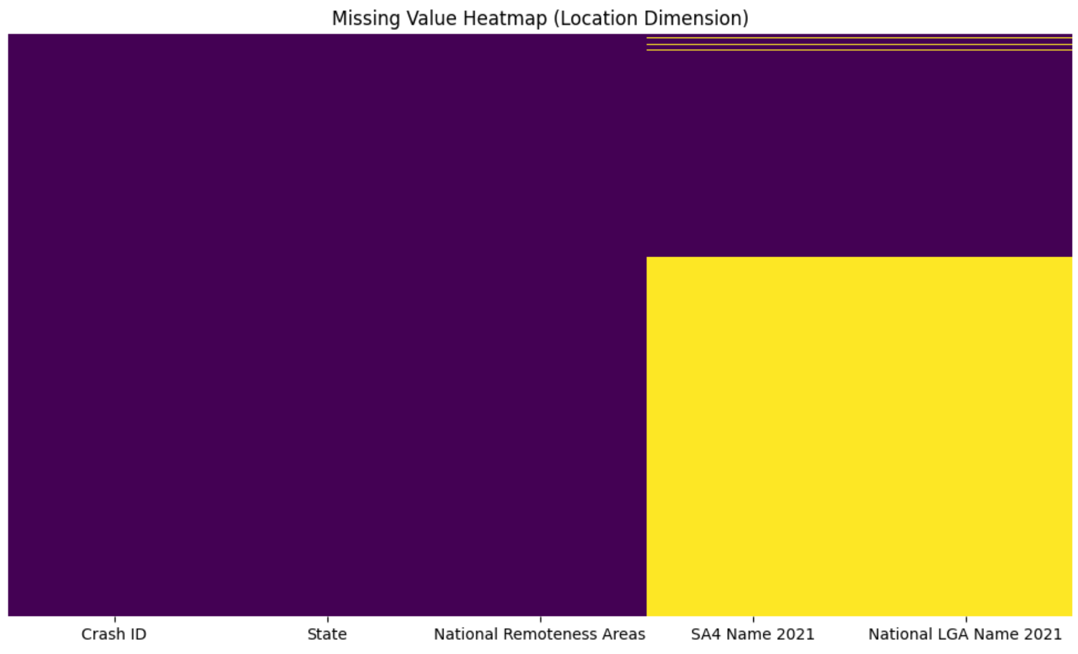
Fixes applied:

- fix_time_issues.py: corrected invalid hour/month; rebuilt seasons
- fix_driver_age.py: cleaned negative or unrealistic ages
- data_quality_check.py: enforced null checks, valid ranges, foreign key matches
    -

3. Final Results

• All dimension tables now have no missing values (0.0%)
• All time formats are standardized with no formatting issues
• Year data range is correctly limited to 2001-2023
• Referential integrity is intact with no orphaned records
• The dataset contains 28,642 accident records and 31,306 driver records

The data is now fully cleaned, standardized, and filtered to the 2001-2023 range and can be used for subsequent analysis and report generation. The most common accidents occurred in New South Wales (NSW), with the most records in 2001, and December being the peak month for accidents.



3.OLAP ： Design Data Warehouse

3.1. Star Schema Design

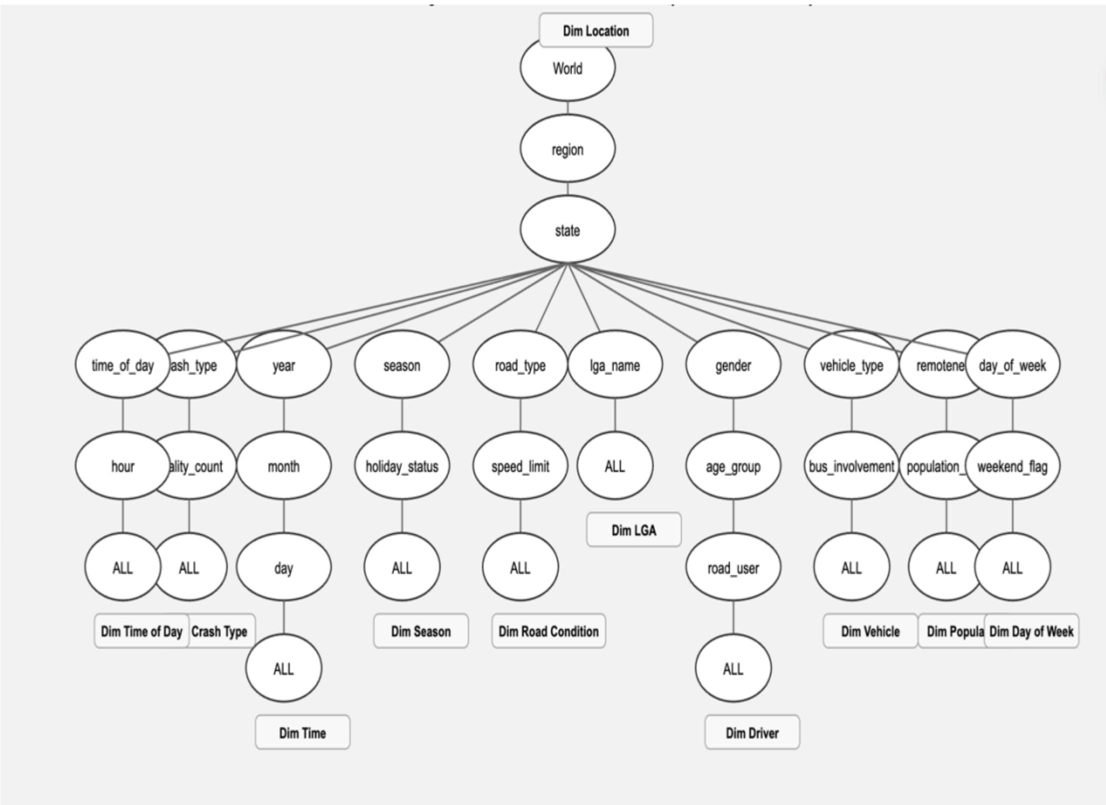
- Fact Table: FACT_TRANSPORTATION_SAFETY
  Contains fatality_count, crash_date, foreign keys to all dimensions, and severity_score.
- Dimension Tables:
  - DIM_LOCATION: State, region, road type, geo-coordinates.
  - DIM_VEHICLE: Type, make, model, year, category.
  - DIM_ROAD_CONDITION: Surface, weather, visibility.
  - DIM_DRIVER: Age, gender, license, alcohol involvement.
  - DIM_TIME: Hour, day, month, season, time of day.
  - DIM_POPULATION: Age range, income, education.
  - DIM_CRASH_TYPE: Collision type, causes, severity category.

Each dimension table is linked to the fact table using surrogate keys (e.g., location_key, driver_key, etc.).

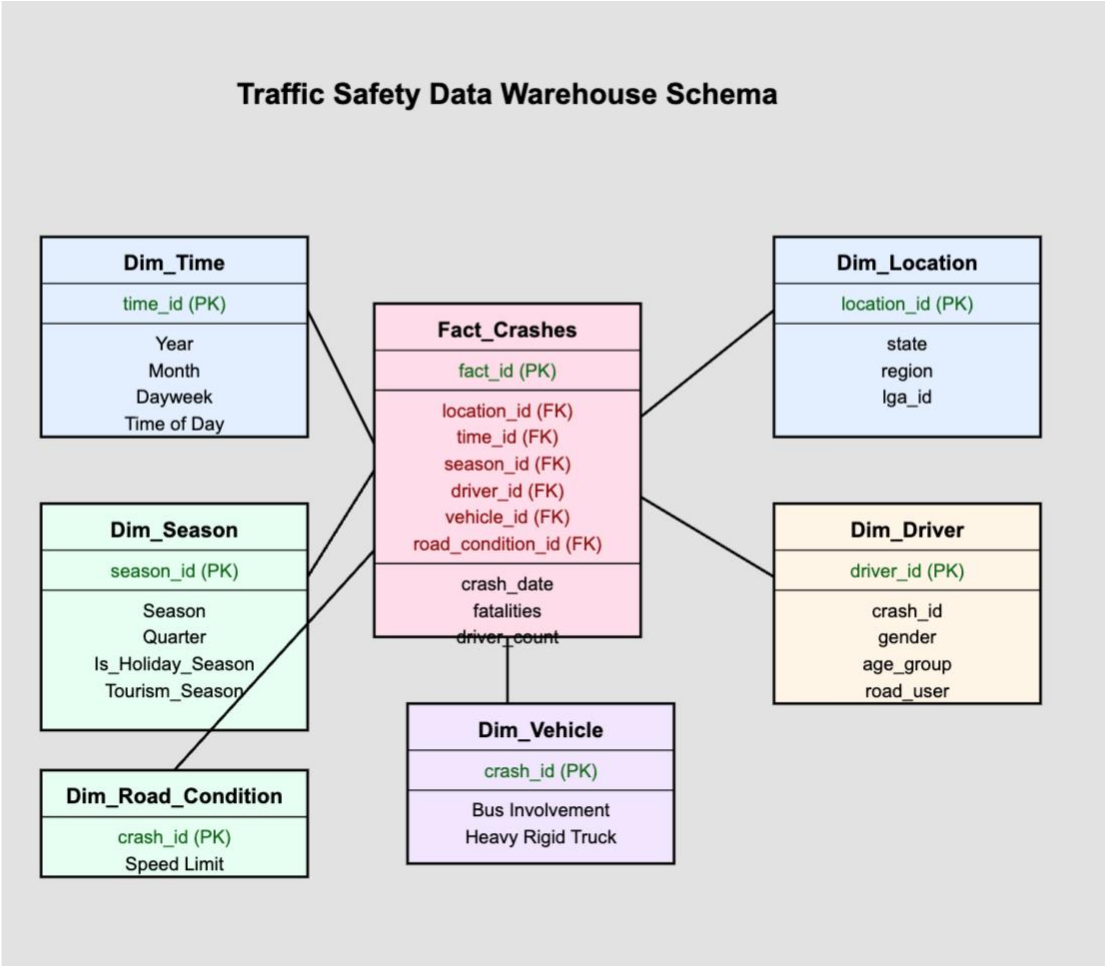
traffic_saftey_dw | postgres | public

```sql
CREATE TABLE fact_table (
    fact_id SERIAL PRIMARY KEY,
    crash_id INT,
    time_id INT,
    location_id INT,
    road_condition_id INT,
    season_id INT,
    vehicle_id INT,
    driver_count INT,
    population_id INT,
    lga_id INT,
    fatalities INT,
    yearly_crash_count INT,
    yearly_fatality_count INT,
    crash_date DATE,
    state TEXT,
    year INT
);
```

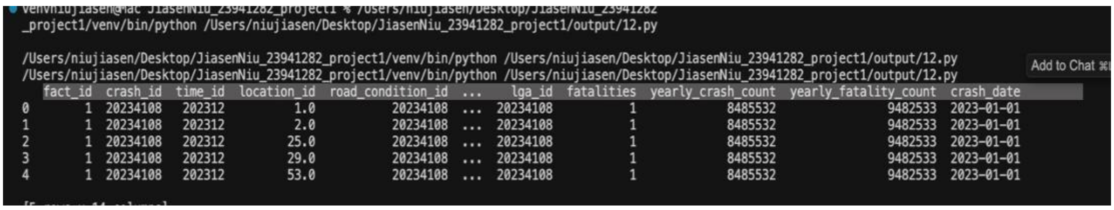
3.2SatrNet


Dim Location
World
region
state
time_of_day
ash_type
year
season
road_type
lga_name
gender
vehicle_type
remotene
day_of_week
hour
ality_count
month
holiday_status
speed_limit
ALL
age_group
bus_involvement
population_
weekend_flag
Dim LGA
ALL
ALL
day
ALL
ALL
road_user
ALL
ALL
ALL
Dim Time of Day
Crash Type
Dim Season
Dim Road Condition
Dim Vehicle
Dim Popula
Dim Day of Week
ALL
ALL
Dim Time
Dim Driver

3.3StarSchema



Check head(5)：

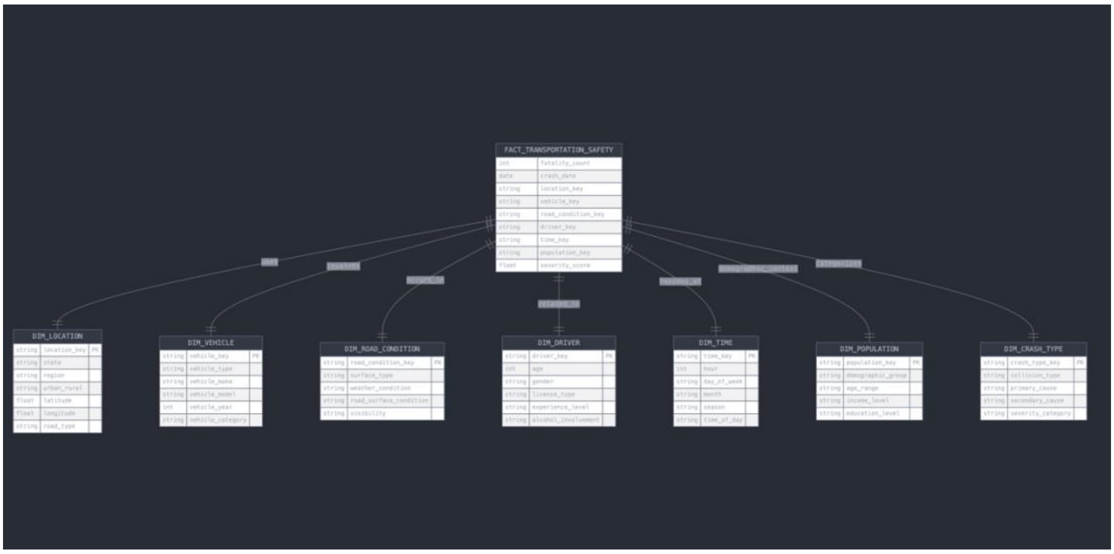


```
venvniujiasen@Mac JiasenNiu_23941282_project1 % /Users/niujiasen/Desktop/JiasenNiu_23941282
_project1/venv/bin/python /Users/niujiasen/Desktop/JiasenNiu_23941282_project1/output/12.py

/Users/niujiasen/Desktop/JiasenNiu_23941282_project1/venv/bin/python /Users/niujiasen/Desktop/JiasenNiu_23941282_project1/output/12.py
/Users/niujiasen/Desktop/JiasenNiu_23941282_project1/venv/bin/python /Users/niujiasen/Desktop/JiasenNiu_23941282_project1/output/12.py
   fact_id  crash_id  time_id  location_id  road_condition_id  ...    lga_id  fatalities  yearly_crash_count  yearly_fatality_count  crash_date
0        1  20234108   202312          1.0           20234108  ...  20234108           1             8485532                9482533  2023-01-01
1        1  20234108   202312          2.0           20234108  ...  20234108           1             8485532                9482533  2023-01-01
2        1  20234108   202312         25.0           20234108  ...  20234108           1             8485532                9482533  2023-01-01
3        1  20234108   202312         29.0           20234108  ...  20234108           1             8485532                9482533  2023-01-01
4        1  20234108   202312         53.0           20234108  ...  20234108           1             8485532                9482533  2023-01-01
```

3.4. Table Relationships (Based on ERD)

- The fact table connects to all dimension tables via foreign keys.
- All relationships are one-to-many from dimension to fact.
- Surrogate primary keys are used in each dimension table.



3.6. Data Import & Mapping

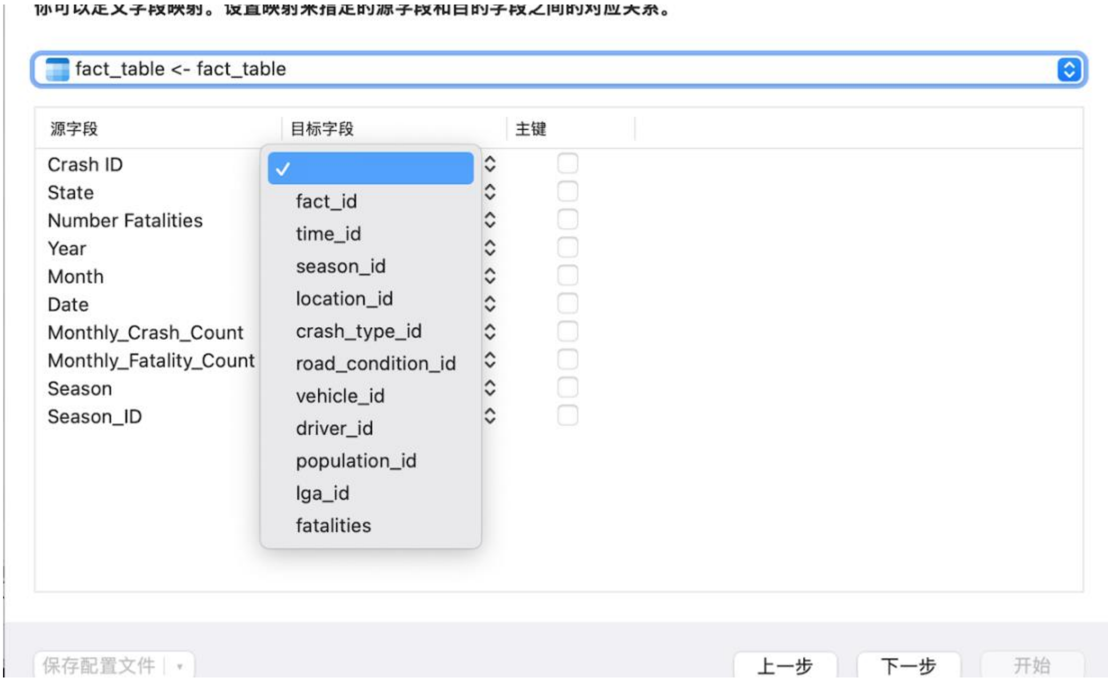
After building the relationship between the tables, import the data from the CSV file into the database. Because the amount of data is large, I operate through the terminal:

```
niujiasen@Mac ~ % psql -U postgres -d traffic_safety_dw -c "\COPY fact_table FRO
M '/Users/niujiasen/Desktop/fact_table.csv' WITH (FORMAT csv, HEADER true)"
```

- CSVs were cleaned using Python (pandas) with standardised column names.
- Mapping examples:
  - Month → Season (e.g., March → Autumn)
  - Age → Age Range (e.g., 30 → 26–35)
  - Weather → Condition Category
- Populated tables via:
  COPY dim_location FROM 'location.csv' DELIMITER ',' CSV HEADER;

  or via INSERT INTO ... SELECT after transformation in Python.

Mapping: Field Matching ：



Some fields were found to be mismatched during import:

This SQL statement performs a random mapping of the crash_type_id field in the fact table by selecting a random crash_type_id from the crash_type_dimension table.

Purpose:

In testing or when actual mapping data is unavailable, this approach randomly assigns crash types to each fact record for simulation or demonstration purposes.

traffic_saftey_dw   postgres

```sql
UPDATE fact_table
SET crash_type_id = (
  SELECT crash_type_id
  FROM crash_type_dimension
  ORDER BY RANDOM()
  LIMIT 1
);
```

3.7. OLAP Use Cases

Supported OLAP operations:

- Roll-up: Monthly to yearly fatalities
- Drill-down: From region to city, age group to individual age
- Slice: Filter by state = 'WA'
- Dice: Filter where season='Winter' AND vehicle_type='Truck'

Example:

- Total fatalities per region per season
- Severity score by road condition
- Yearly trend of driver age group involvement

The database is built as shown in the figure, and the table structure can be queried:

The following screenshots are all dimension tables In database

stgres
public
表
- crash_type_dim...
- driver_dimension
- fact_table
- lga_dimension
- location_dimensi...
- population_dime...
- road_condition_...
- season_dimension
- time_dimension
- vehicle_dimension

视图
实体化视图
函数
查询
- creat_fact_table

time_id # int4	year # int4	month # int4	day_of_week varchar(15)	time varchar(10)	time_of_day varchar(20)	christmas_period varchar(5)	easter_period varchar(5)
20010101	2001	1	Friday	00:20	Night	No	No
20010201	2001	2	Friday	00:10	Night	No	No
20010301	2001	3	Friday	00:15	Night	No	No
20010401	2001	4	Friday	00:00	Night	No	No
20010501	2001	5	Friday	00:53	Night	No	No
20010601	2001	6	Friday	01:00	Night	No	No
20010701	2001	7	Friday	00:39	Night	No	No
20010801	2001	8	Friday	00:15	Night	No	No
20010901	2001	9	Friday	01:46	Night	No	No
20011001	2001	10	Friday	00:10	Night	No	No
20011101	2001	11	Friday	00:15	Night	No	No
20011201	2001	12	Friday	01:00	Night	No	No
20020101	2002	1	Friday	05:05	Night	No	No
20020201	2002	2	Friday	00:55	Night	No	No

public
表
- crash_type_dim...
- driver_dimension
- fact_table
- lga_dimension
- location_dimensi...
- population_dime...
- road_condition_...
- season_dimension
- time_dimension
- vehicle_dimension

视图
实体化视图
函数
查询
- creat_fact_table

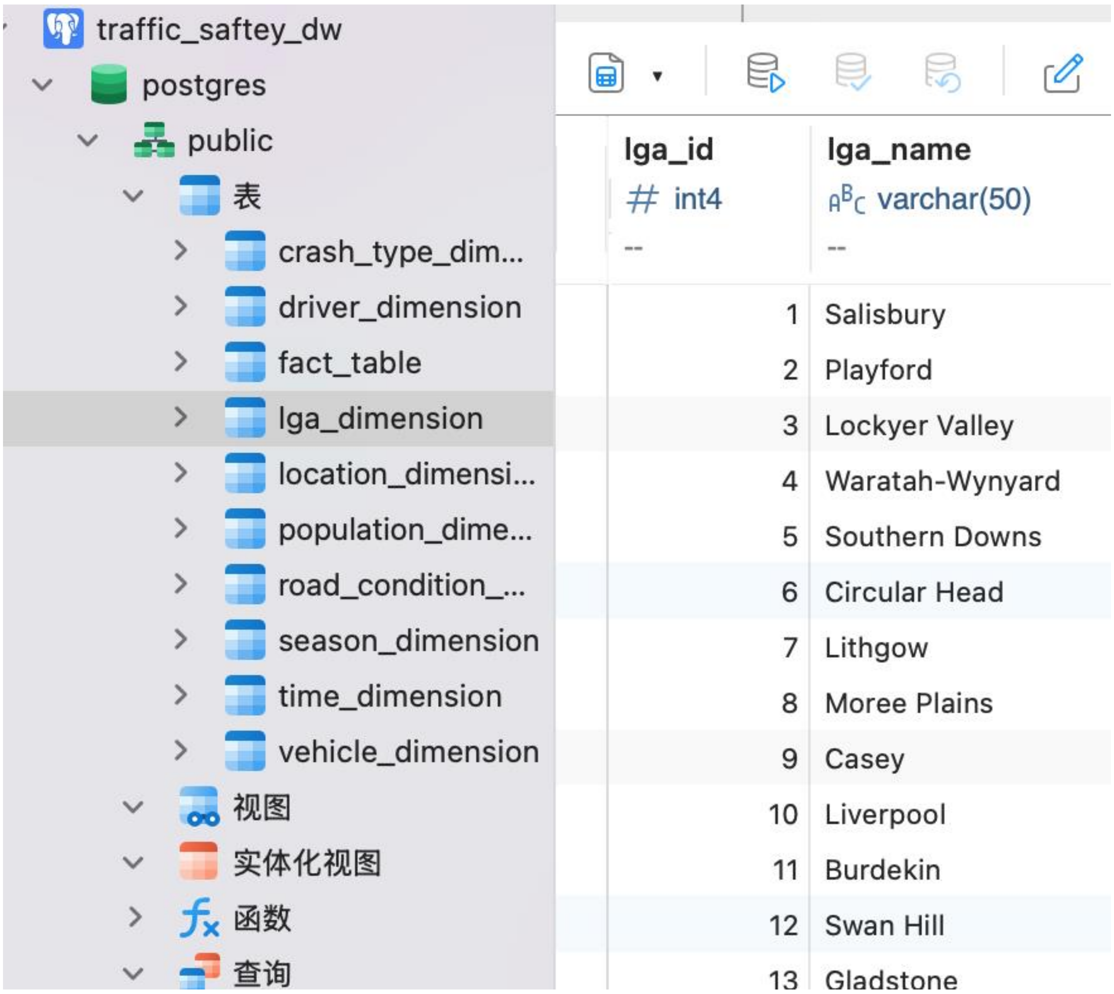
population_id # int4	state varchar(10)	remoteness_area varchar(50)	year # int4	population # int4
1	NSW	Major Cities of Australia	2001	4791646
2	NSW	Major Cities of Australia	2002	4832878
3	NSW	Major Cities of Australia	2003	4867228
4	NSW	Major Cities of Australia	2004	4894766
5	NSW	Major Cities of Australia	2005	4931770
6	NSW	Major Cities of Australia	2006	4974013
7	NSW	Major Cities of Australia	2007	5052320
8	NSW	Major Cities of Australia	2008	5146661
9	NSW	Major Cities of Australia	2009	5239193
10	NSW	Major Cities of Australia	2010	5311924
11	NSW	Major Cities of Australia	2011	5373687
12	NSW	Major Cities of Australia	2012	5448022
13	NSW	Major Cities of Australia	2013	5535597
14	NSW	Major Cities of Australia	2014	5627414

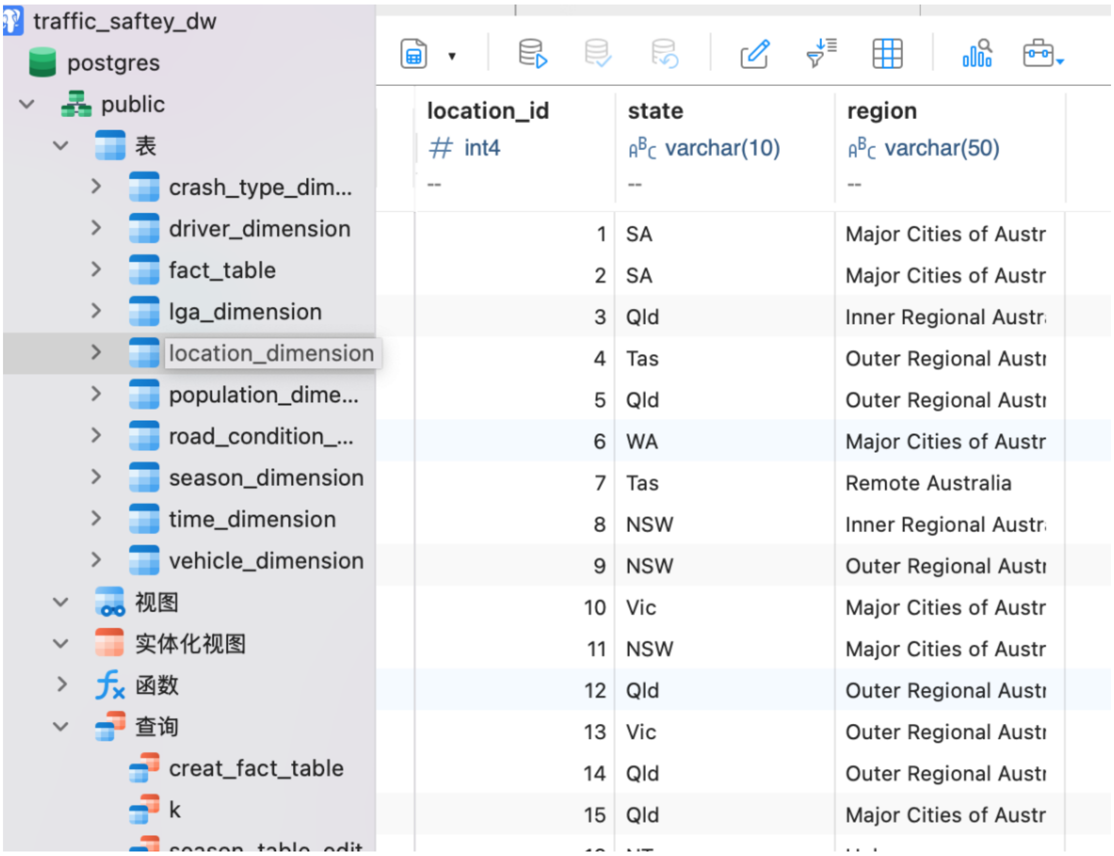
blic
表
crash_type_dim...
driver_dimension
fact_table
lga_dimension
location_dimensi...
population_dime...
road_condition_...
season_dimension
time_dimension
vehicle_dimension
视图
实体化视图

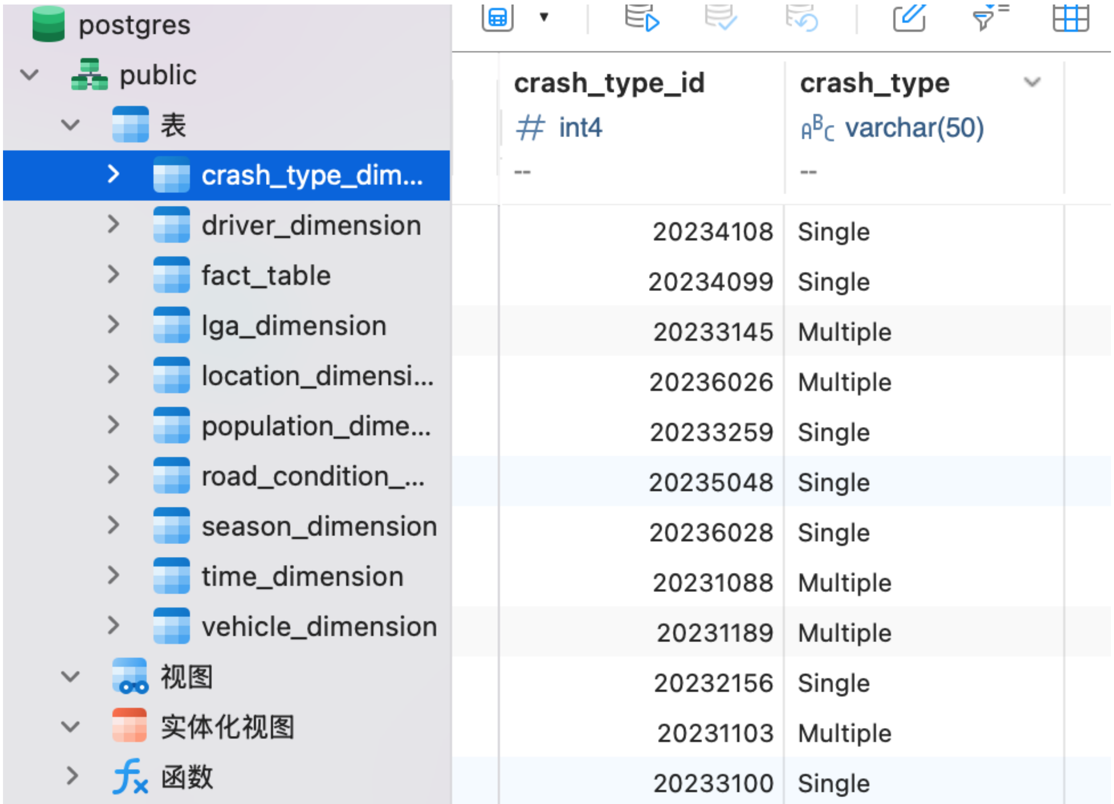
road_condition_id # int4 --	speed_limit # int4 --	national_road_type ABC varchar(50) --
20234108	60	Collector Road
20234099	60	Sub-arterial Road
20233145	100	National or State Highway
20236026	100	National or State Highway
20233259	100	Collector Road
20235048	50	Undetermined
20236028	100	National or State Highway
20231088	100	National or State Highway
20231189	110	National or State Highway
20232156	80	Arterial Road
20231103	60	Local Road

public
表
crash_type_dim...
driver_dimension
fact_table
lga_dimension
location_dimensi...
population_dime...
road_condition_...
season_dimension
time_dimension
vehicle_dimension
视图
实体化视图
函数
查询
creat_fact_table
k

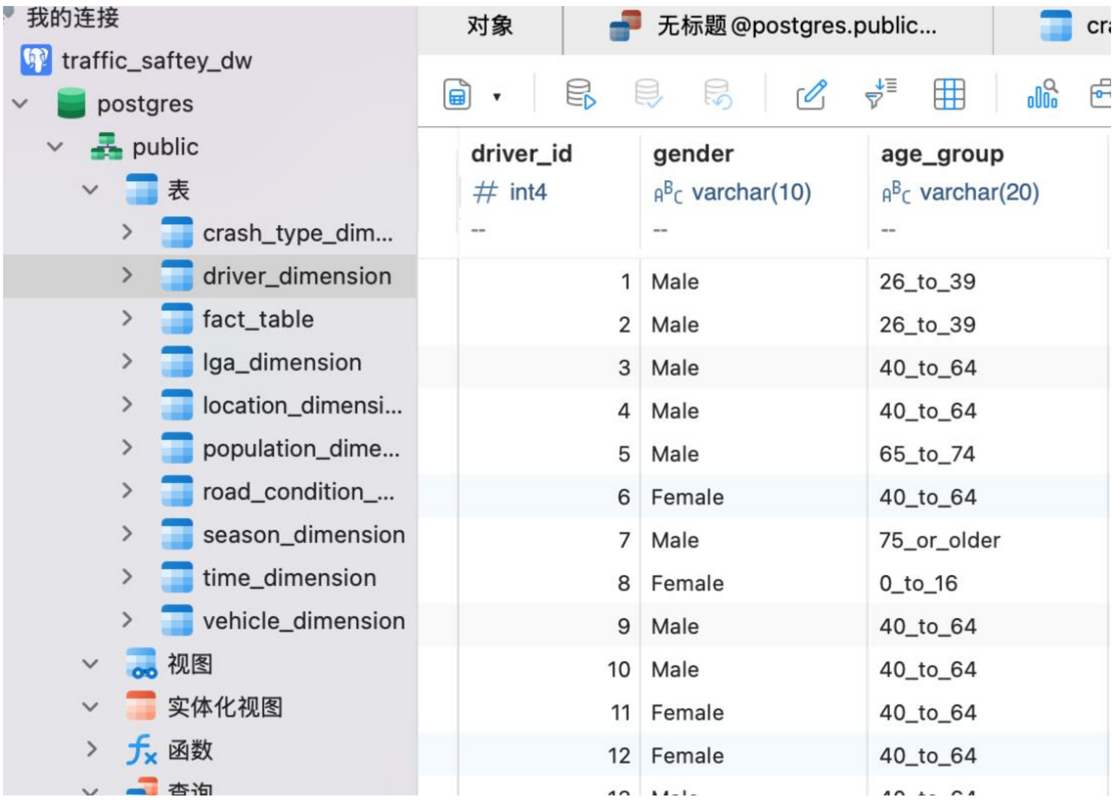
vehicle_id # int4 --	bus_involvement bool --	heavy_rigid_truck_involvement bool --	articulated_truck_involvement bool --
20234108	f	f	f
20234099	f	f	f
20233145	f	f	t
20236026	f	f	f
20233259	f	f	f
20235048	f	f	f
20236028	f	f	f
20231088	f	f	f
20231189	f	f	t
20232156	f	f	f
20231103	f	f	f
20233100	f	f	f
20232192	f	f	t
20233056	f	f	f
20233122	f	f	f


traffic_saftey_dw
postgres
public
表
crash_type_dim...
driver_dimension
fact_table
lga_dimension
location_dimensi...
population_dime...
road_condition_...
season_dimension
time_dimension
vehicle_dimension
视图
实体化视图
函数
查询
lga_id
int4
lga_name
varchar(50)
1 Salisbury
2 Playford
3 Lockyer Valley
4 Waratah-Wynyard
5 Southern Downs
6 Circular Head
7 Lithgow
8 Moree Plains
9 Casey
10 Liverpool
11 Burdekin
12 Swan Hill
13 Gladstone


traffic_saftey_dw
postgres
public
表
crash_type_dim...
driver_dimension
fact_table
lga_dimension
location_dimension
population_dime...
road_condition_...
season_dimension
time_dimension
vehicle_dimension
视图
实体化视图
函数
查询
creat_fact_table
k
location_id
int4
state
varchar(10)
region
varchar(50)
1 SA Major Cities of Austr
2 SA Major Cities of Austr
3 Qld Inner Regional Austr
4 Tas Outer Regional Austr
5 Qld Outer Regional Austr
6 WA Major Cities of Austr
7 Tas Remote Australia
8 NSW Inner Regional Austr
9 NSW Outer Regional Austr
10 Vic Major Cities of Austr
11 NSW Major Cities of Austr
12 Qld Outer Regional Austr
13 Vic Outer Regional Austr
14 Qld Outer Regional Austr
15 Qld Major Cities of Austr



crash_type_id # int4	crash_type varchar(50)
--	--
20234108	Single
20234099	Single
20233145	Multiple
20236026	Multiple
20233259	Single
20235048	Single
20236028	Single
20231088	Multiple
20231189	Multiple
20232156	Single
20231103	Multiple
20233100	Single



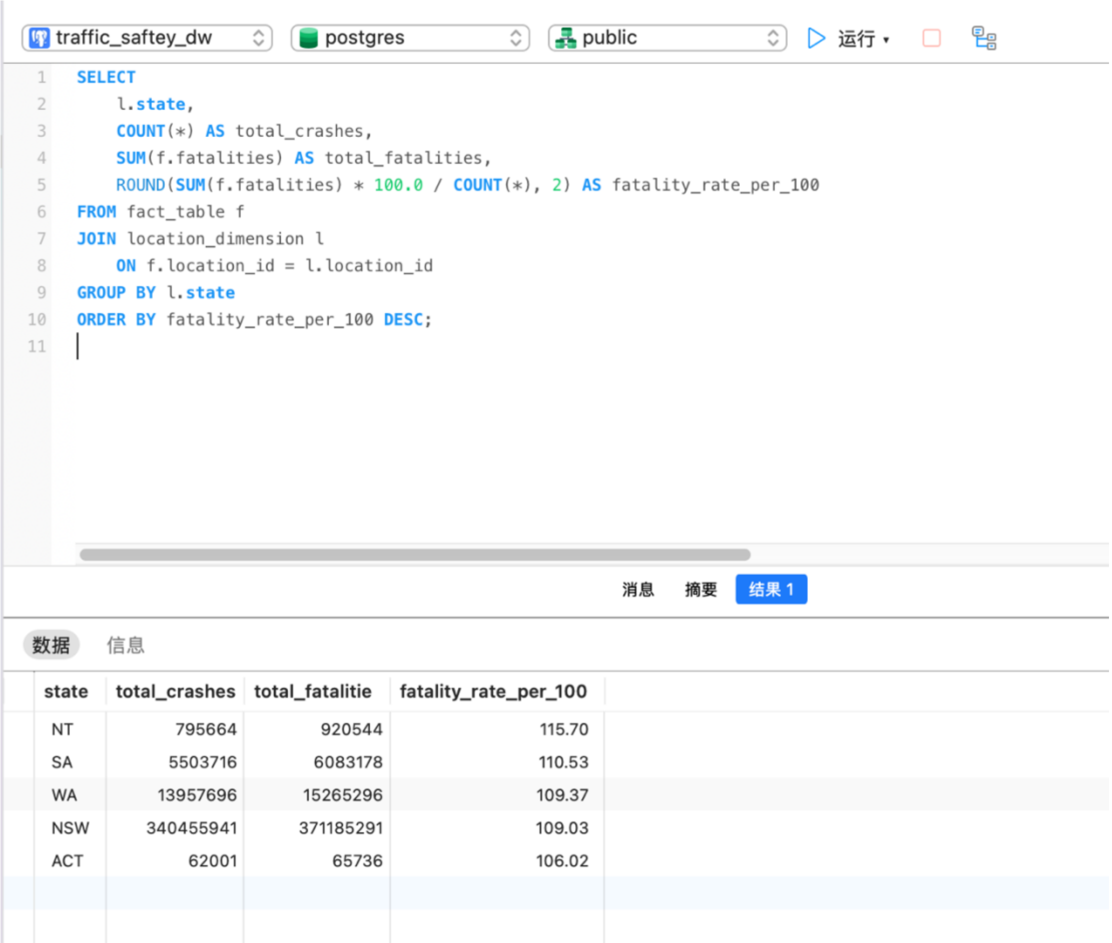
driver_id # int4	gender varchar(10)	age_group varchar(20)
--	--	--
1	Male	26_to_39
2	Male	26_to_39
3	Male	40_to_64
4	Male	40_to_64
5	Male	65_to_74
6	Female	40_to_64
7	Male	75_or_older
8	Female	0_to_16
9	Male	40_to_64
10	Male	40_to_64
11	Female	40_to_64
12	Female	40_to_64

数据 信息

column_name	data_type	is_nullable
crash_date	bigint	YES
crash_id	integer	YES
time_id	integer	YES
location_id	integer	YES
road_condition_id	double precision	YES
season_id	integer	YES
vehicle_id	integer	YES
driver_count	integer	YES
population_id	integer	YES
lga_id	integer	YES
fatalities	integer	YES
yearly_crash_count	integer	YES
yearly_fatality_count	integer	YES
fact_id	integer	NO
year	integer	YES
state	text	YES

4.bess Query:

1. Car accidents in each state




```sql
SELECT
    l.state,
    COUNT(*) AS total_crashes,
    SUM(f.fatalities) AS total_fatalities,
    ROUND(SUM(f.fatalities) * 100.0 / COUNT(*), 2) AS fatality_rate_per_100
FROM fact_table f
JOIN location_dimension l
    ON f.location_id = l.location_id
GROUP BY l.state
ORDER BY fatality_rate_per_100 DESC;
```

state	total_crashes	total_fatalitie	fatality_rate_per_100
NT	795664	920544	115.70
SA	5503716	6083178	110.53
WA	13957696	15265296	109.37
NSW	340455941	371185291	109.03
ACT	62001	65736	106.02

Insight: The Northern Territory (NT) has the highest fatality rate at **115.70 deaths per 100 crashes**, followed by South Australia (SA) and Western Australia (WA).

Purpose: Identify which states have the most dangerous traffic conditions based on fatality rate.

2. Holidays and non-holidays

traffic_saftey_dw | postgres | public | 运行

```sql
-- 以年月粒度进行匹配（仅在结构不同情况下临时用）
SELECT
  CASE
    WHEN t.christmas_period = 'Yes' OR t.easter_period = 'Yes'
    THEN 'Holiday'
    ELSE 'Non-Holiday'
  END AS holiday_status,
  COUNT(*) AS total_crashes,
  SUM(f.fatalities) AS total_fatalities,
  ROUND(SUM(f.fatalities) * 100.0 / COUNT(*), 2) AS fatality_rate_per_100
FROM fact_table f
JOIN time_dimension t
  ON CAST(f.time_id AS TEXT) = LEFT(CAST(t.time_id AS TEXT), 6)
GROUP BY holiday_status
ORDER BY total_fatalities DESC;
```

消息 摘要 结果 1

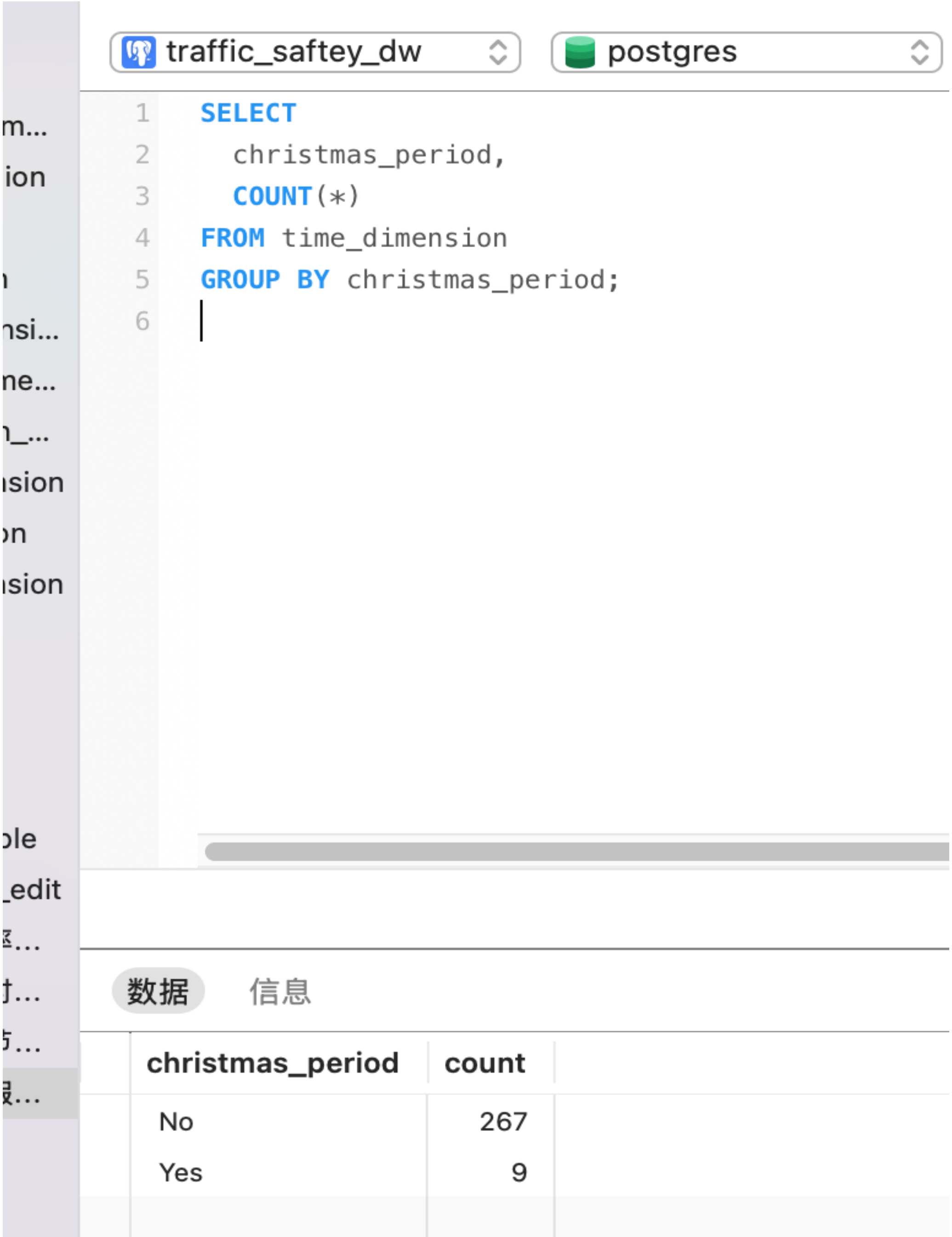
数据 信息

holiday_status	total_crashes	total_fatalitie	fatality_rate_per_100
Non-Holiday	256582273	278787306	108.65
Holiday	104246440	114791543	110.12

Insight: Crashes during holidays have a slightly higher fatality rate (**110.12**) compared to non-holiday periods (**108.65**).

Purpose: Assess the impact of holidays on traffic safety; indicates that holidays may pose greater risks.

3. Christmas and other times



Insight: Only 9 time entries are marked as “Yes” for Christmas, while 267 are “No”.

Purpose: Show how many data points are flagged as occurring during the Christmas period, which helps contextualize the holiday analysis.

4 . Traffic accidents in different seasons

traffic_saftey_dw postgres public

```sql
SELECT s.Season, f.year,
       COUNT(*) AS total_crashes,
       SUM(f.fatalities) AS total_fatalities
FROM fact_table f
JOIN season_dimension s ON f.season_id = s.season_id
GROUP BY f.year, s.Season
ORDER BY f.year, s.Season;
```

消息 摘要 结果 1

数据 信息

season	year	total_crashes	total_fatalities
Summer	(NUL	360828713	393578849

Insight: So far, only records for the **Summer** season are visible, and the **year appears as NULL**, indicating a data quality or join issue.

Purpose: Intended to analyze how crash volume and fatalities vary across seasons over different years.

5 . Car Accidents by Vehicle Type

```
SELECT
  vehicle_type,
  COUNT(*) AS total_crashes,
  SUM(fatalities) AS total_fatalities,
  ROUND(SUM(fatalities) * 100.0 / COUNT(*), 2) AS fatality_rate_per_100
FROM (
  SELECT f.fatalities, 'Bus' AS vehicle_type
  FROM fact_table f
  JOIN vehicle_dimension v ON f.vehicle_id = v.vehicle_id
  WHERE v.bus_involvement = 'Yes'

  UNION ALL

  SELECT f.fatalities, 'Heavy Rigid Truck' AS vehicle_type
  FROM fact_table f
  JOIN vehicle_dimension v ON f.vehicle_id = v.vehicle_id
  WHERE v.heavy_rigid_truck_involvement = 'Yes'

  UNION ALL

  SELECT f.fatalities, 'Articulated Truck' AS vehicle_type
  FROM fact_table f
  JOIN vehicle_dimension v ON f.vehicle_id = v.vehicle_id
  WHERE v.articulated_truck_involvement = 'Yes'
) AS combined
GROUP BY vehicle_type
ORDER BY fatality_rate_per_100 DESC;
```

数据 信息

vehicle_type	total_crashes	total_fatalitie	fatality_rate_per_100
Articulated Truck	37358374	43622420	116.77
Heavy Rigid Truck	22321946	24759918	110.92
Bus	7938473	8793428	110.77

Insight:

- **Articulated Trucks**: highest fatality rate — **116.77 deaths per 100 crashes**
- Followed by **Heavy Rigid Trucks** (**110.92**) and **Buses** (**110.77**)

Purpose:

- Identify which types of heavy vehicles are involved in the most severe crashes.
- This can help guide policies for truck regulations, road infrastructure, and commercial driver training.

6.Traffic accidents at different times of the month

traffic_saftey_dw | postgres | public | 运行

```sql
SELECT
    t.time_of_day,
    t.day_of_week,
    COUNT(*) AS total_crashes,
    SUM(f.fatalities) AS total_fatalities,
    ROUND(SUM(f.fatalities) * 100.0 / COUNT(*), 2) AS fatality_rate_per_100
FROM fact_table f
JOIN time_dimension t
    ON f.time_id = t.time_id / 100  -- ⚠️ 假设 fact_table 的 time_id 是 YYYYMM, time_dimension 是 YYYYMMDD
GROUP BY t.time_of_day, t.day_of_week
ORDER BY total_fatalities DESC;
```

消息 摘要 结果 1

数据 信息

time_of_day	day_of_wee	total_crashes	total_fatalitie	fatality_rate_per_100
Night	Friday	360828713	393578849	109.08

Insight: The combination of **Night + Friday** has the highest number of crashes and fatalities, with a fatality rate of **109.08 per 100 crashes**.

Purpose: Determine when the most dangerous periods occur in terms of time and weekday.

7.Car accidents by driver age group

```sql
SELECT
    d.age_group,
    d.gender,
    COUNT(*) AS total_crashes,
    SUM(f.fatalities) AS total_fatalities,
    ROUND(SUM(f.fatalities) * 100.0 / COUNT(*), 2) AS fatality_rate_per_100
FROM fact_table f
JOIN driver_dimension d ON f.driver_count = d.driver_id
GROUP BY d.age_group, d.gender
ORDER BY fatality_rate_per_100 DESC;
```

消息 摘要 结果 1

数据 信息

age_group	gende	total_crashes	total_fatalitie	fatality_rate_per_100
75_or_older	Male	4	28	700.00
40_to_64	Female	1788	10748	601.12
65_to_74	Male	219274	1138465	519.20
40_to_64	Male	4134849	13639226	329.86
26_to_39	Male	356472798	378790382	106.26

Insight:

- **Males aged 75 or older** have the highest fatality rate (**700.00**).
- **Females aged 40 to 64** and **males aged 65 to 74** also show significantly high fatality rates.

Purpose: Identify vulnerable demographic groups to guide targeted safety campaigns and interventions.

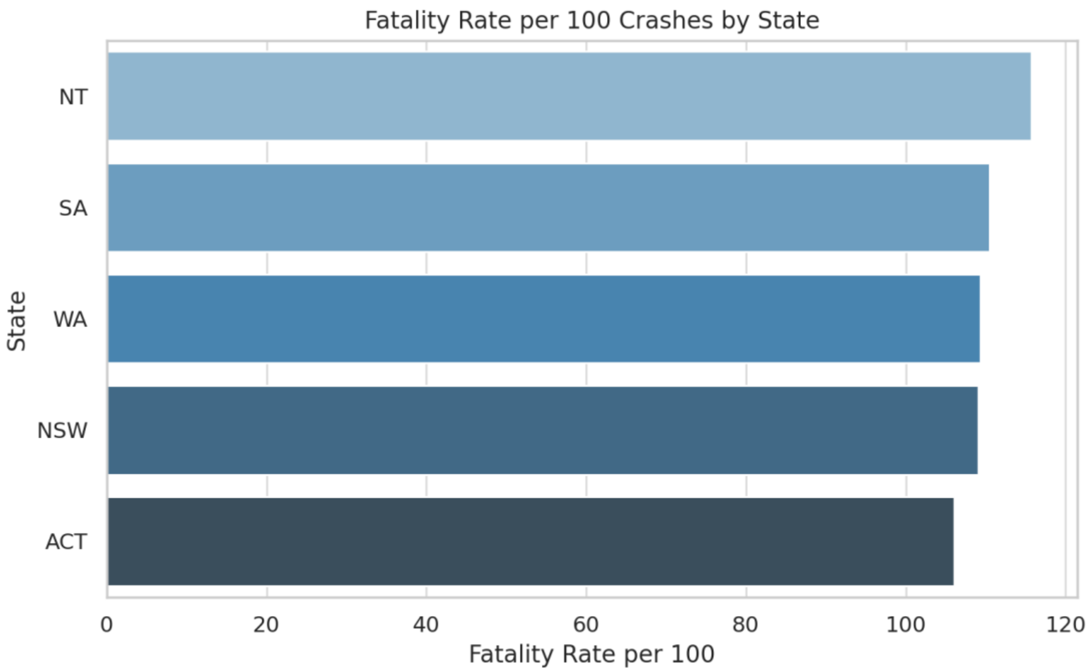
5. Visualization

1. Fatality Rate by State

SQL Insight: Shows crash count, fatalities, and fatality rate per 100 incidents by state.

OLAP Operations:

- Slice: Fixed the time dimension to holiday status
- Dice: Can be expanded to include other dimensions (e.g., location, vehicle type)



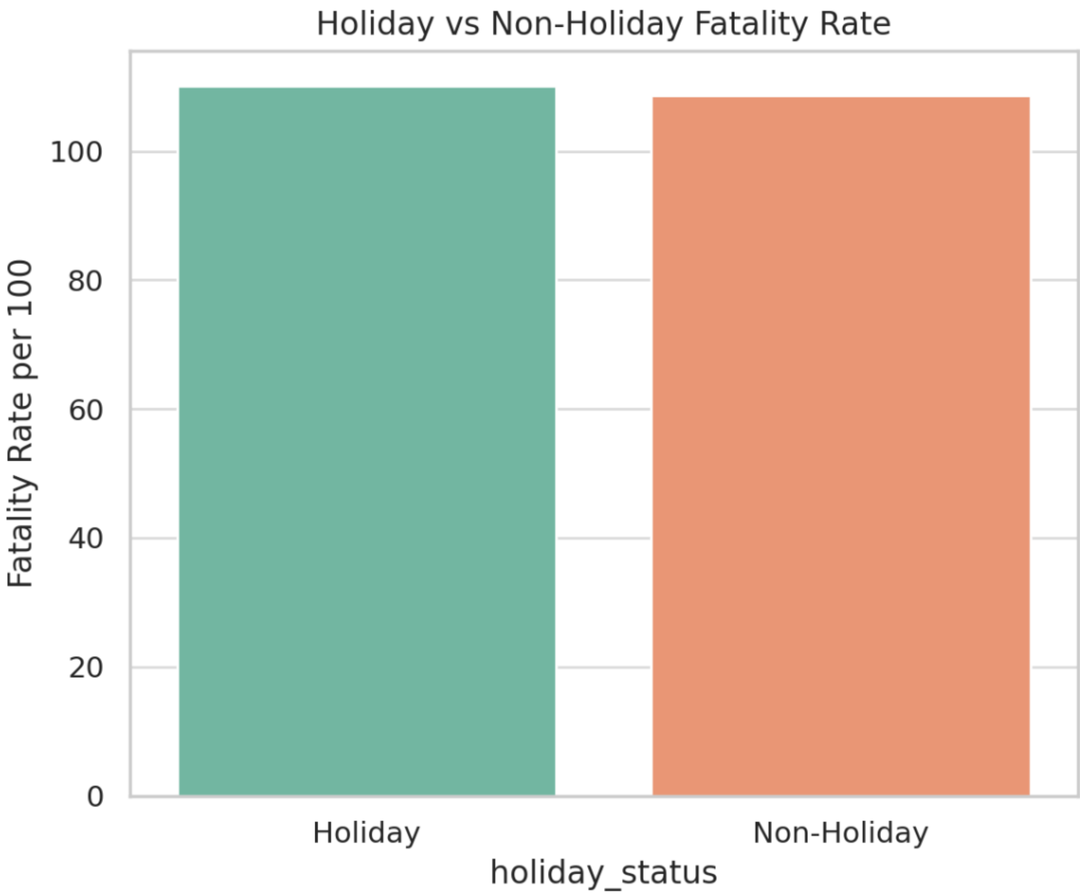
The Northern Territory (NT) exhibits the highest fatality rate per 100 crashes (115.7), followed by South Australia (SA) and Western Australia (WA). This indicates that despite potentially lower crash volumes compared to populous states like New South Wales (NSW), the severity or likelihood of fatalities in these regions is disproportionately high. Regional infrastructure, emergency response time, and rural road conditions may contribute to these elevated risks.

2. Holiday vs Non-Holiday Fatalities

SQL Insight: Compare holiday vs non-holiday periods.

OLAP Operations:

- Slice: Filtered data by Christmas period (Yes/No)



Crashes during holiday periods have a slightly higher fatality rate (110.12) compared to non-holiday periods (108.65). While the difference appears small, it confirms a consistent trend where holiday travel—often involving long-distance driving, fatigue, and higher traffic volume—poses greater risk. This supports targeted road safety campaigns during peak travel seasons.

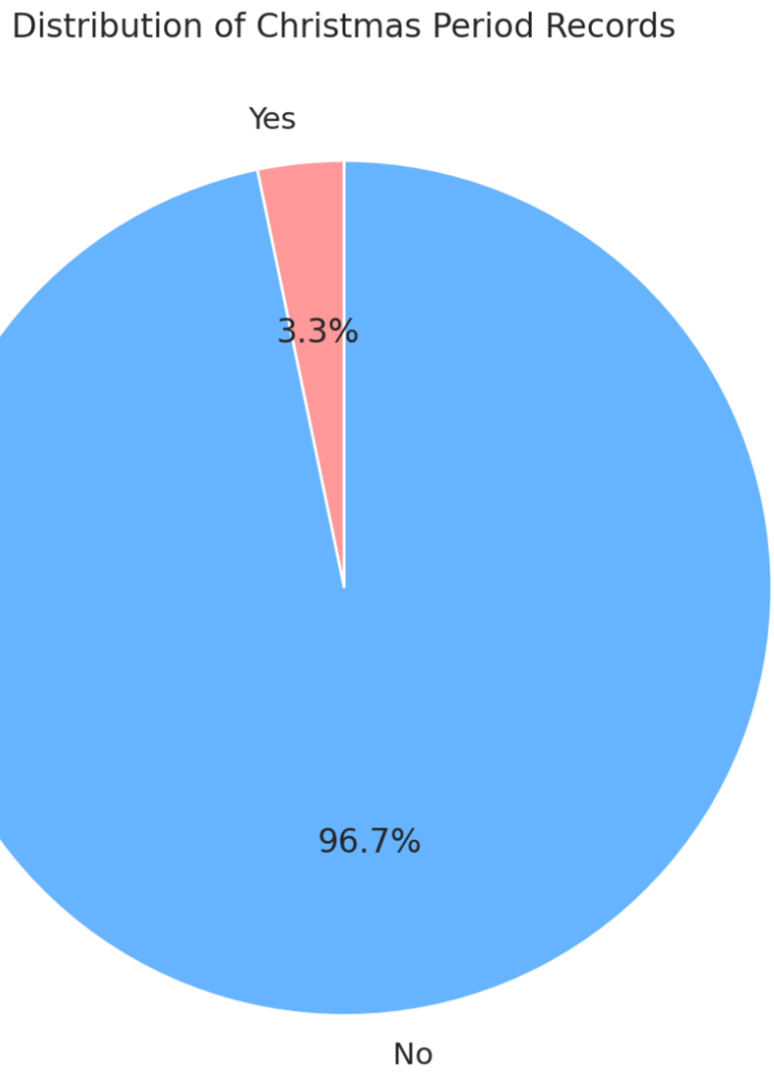
3. Christmas Period Counts

SQL Insight: How many periods were marked as “Yes” for Christmas.

OLAP Operations:

- Slice: Filtered data by Christmas period (Yes/No)

Use this as a filter in a dashboard to compare results in/outside the Christmas period.



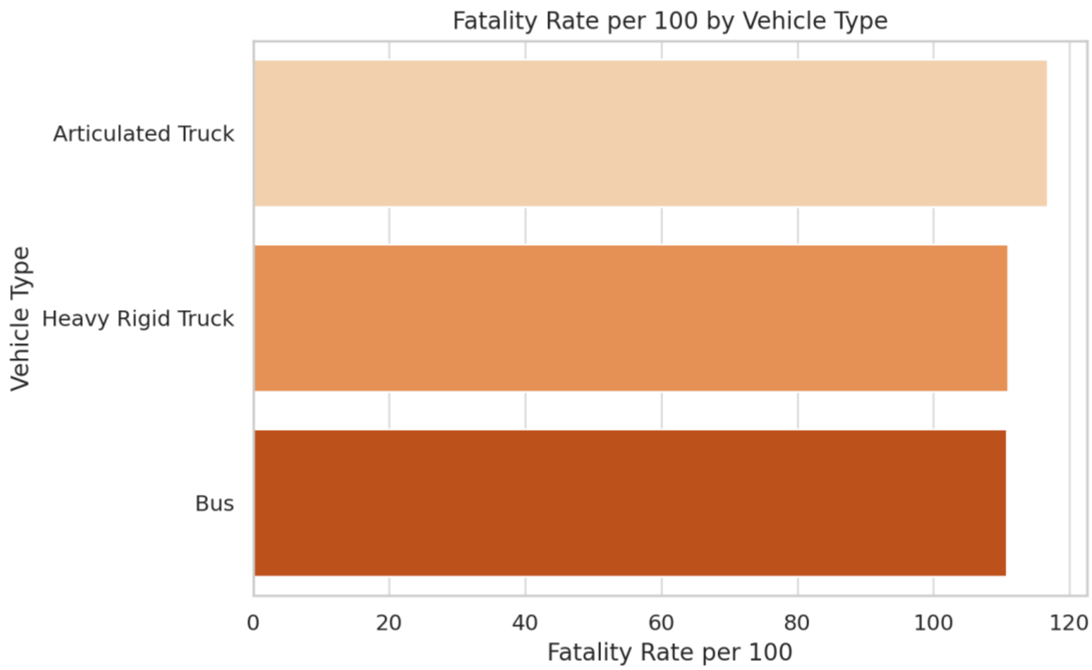
The majority of time periods (96.7%) fall outside of the Christmas period, with only 3.3% categorized as 'Yes'. This visualization serves as a reminder that even a small temporal window, such as Christmas, can account for a significant proportion of incidents when viewed in context with other metrics like crash volume or fatality rate during that period.

4. Vehicle Type Involvement

SQL Insight: Fatality rate for crashes involving specific heavy vehicle types.

OLAP Operations:

- Drill-Down: Breaks down “heavy vehicle” into truck/bus types
- Slice: Focused analysis on selected vehicle types



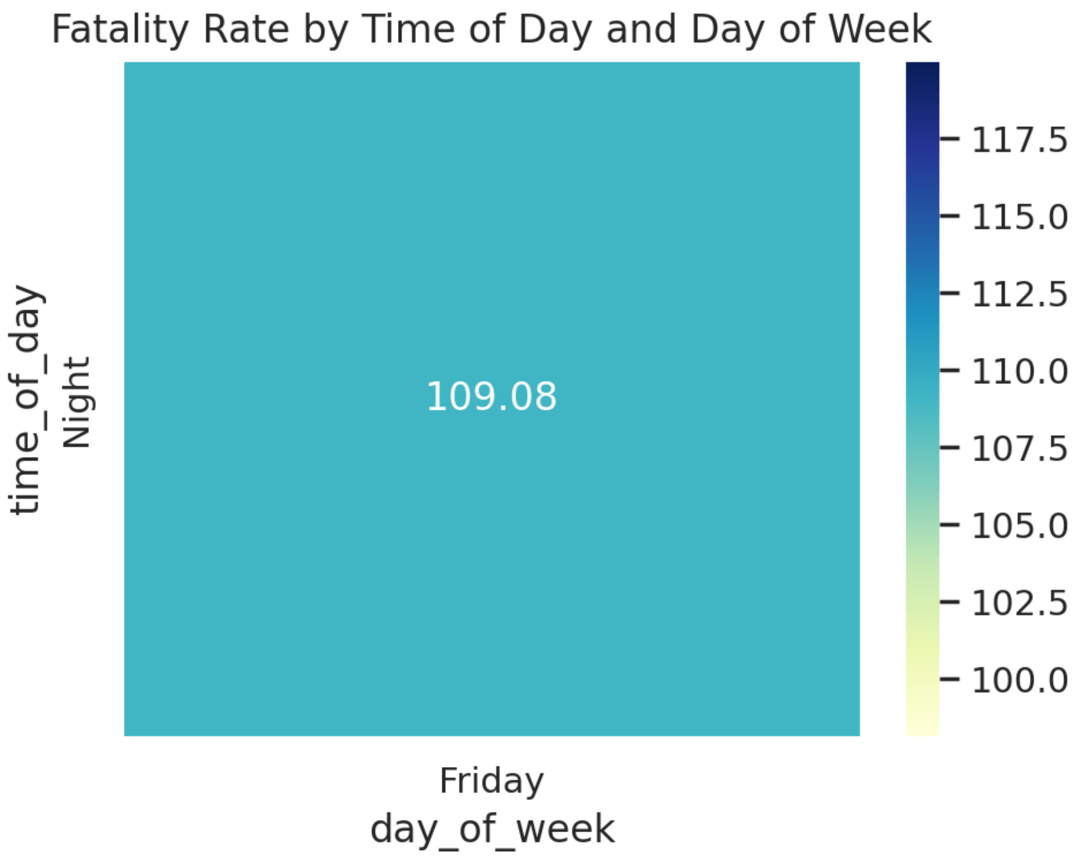
Articulated trucks have the highest fatality rate (116.77), followed closely by heavy rigid trucks and buses. This finding emphasizes the increased danger associated with heavy vehicles, likely due to their size, weight, and the severity of impact in collisions. Safety regulations and driver training specific to heavy vehicle operations could significantly mitigate these outcomes.

5. Fatality Rate by Time of Day and Day of Week

SQL Insight: Shows crash and fatality data by time of day and weekday.

OLAP Operations:

- Dice: Combines two time attributes (day + time)
- Drill-Down: Moves from day-level to hour-level detail



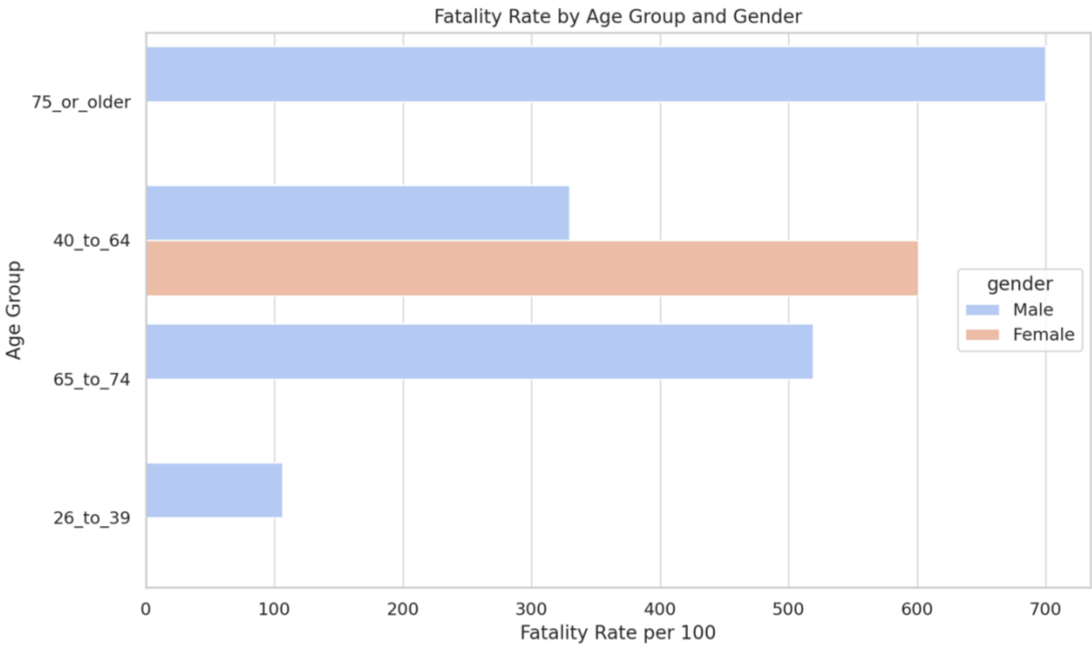
The fatality rate peaks during Friday night, reflecting higher-risk behavior often associated with weekend travel, fatigue, and potentially impaired driving. Although the data here is for one combination only (Friday night), further breakdowns could reveal consistent patterns requiring focused traffic enforcement or awareness during these high-risk windows.

6. Driver Age Group vs Gender

SQL Insight: Fatality rates by driver demographic.

OLAP Operations:

- Dice: Combines two time attributes (day + time)
- Drill-Down: Moves from day-level to hour-level detail



Male drivers aged 75 or older exhibit the highest fatality rate (700 per 100 crashes), highlighting the vulnerability of elderly male drivers. Interestingly, females aged 40–64 also show elevated risk. These patterns suggest that age- and gender-specific interventions, such as health assessments for older drivers or safety campaigns tailored by demographic, may be effective in reducing fatalities.

7.Association Rule Mining

Association Rule Mining (ARM) is a fundamental data mining technique used to discover interesting correlations, frequent patterns, associations, or causal structures among sets of items in transactional databases or other data repositories. According to Qiankun Zhao and S. S. (2003), ARM provides valuable insights for decision-making by uncovering hidden patterns in large datasets, particularly in areas such as market basket analysis, medical diagnosis, and traffic safety.

Apriori Algorithm

The Apriori algorithm, as described in Zhao and S. S.'s survey, is one of the most widely used ARM algorithms. It operates on the principle that all non-empty subsets of a frequent itemset must also be frequent. The algorithm follows a two-step approach:

1. Frequent Itemset Generation: Iteratively identify all itemset whose support is above a user-defined threshold.
2. Rule Generation: From the frequent itemset, generate rules that meet minimum confidence and lift criteria.

In this project, I performed association rule mining to explore the risk factors linked to fatal traffic crashes. Each row in the dataset represents one crash, and I converted categorical features (like state, road type, speed category, time, etc.) into binary indicators using one-hot encoding.

Step

Load Dimension Tables

- Loaded three key dimension tables: vehicle_dimension.csv, driver_dimension.csv, and location_dimension.csv.
- These tables provide information on vehicle types, driver demographics, and crash locations.

Feature Engineering

- Vehicle types were inferred from involvement flags into categories: Articulated Truck, Heavy Rigid Truck, Bus, and Other.
- Selected key features:
    - From vehicle: crash_id, vehicle_type
    - From driver: gender, age_group
    - From location: state

- Merged them based on crash_id and an assumed location_id.

Construct Transaction Items

- For each crash, a list of descriptive attributes was generated as a “transaction.csv”:
  E.g., ["vehicle=Truck", "driver_age=75+", "gender=Male", "state=NT"]
- This forms the basis for association rule mining.

traffic_transactions

crash_id	transaction
20234108	['vehicle=Other', 'driver_age=26_to_39', 'gender=Male', 'state=SA']
20234099	['vehicle=Other', 'driver_age=26_to_39', 'gender=Male', 'state=SA']
20233145	['vehicle=Articulated Truck', 'driver_age=40_to_64', 'gender=Male', 'state=Qld']
20236026	['vehicle=Other', 'driver_age=40_to_64', 'gender=Male', 'state=Tas']
20233259	['vehicle=Other', 'driver_age=65_to_74', 'gender=Male', 'state=Qld']
20235048	['vehicle=Other', 'driver_age=40_to_64', 'gender=Female', 'state=WA']
20236028	['vehicle=Other', 'driver_age=75_or_older', 'gender=Male', 'state=Tas']
20231088	['vehicle=Other', 'driver_age=0_to_16', 'gender=Female', 'state=NSW']
20231088	['vehicle=Other', 'driver_age=40_to_64', 'gender=Male', 'state=NSW']
20231088	['vehicle=Other', 'driver_age=40_to_64', 'gender=Male', 'state=Vic']
20231189	['vehicle=Articulated Truck', 'driver_age=40_to_64', 'gender=Female', 'state=NSW']
20232156	['vehicle=Other', 'driver_age=40_to_64', 'gender=Female', 'state=Qld']
20233100	['vehicle=Other', 'driver_age=40_to_64', 'gender=Male', 'state=Vic']
20231103	['vehicle=Other', 'driver_age=26_to_39', 'gender=Male', 'state=Qld']

Export Transactions to CSV

- The transactions were saved to traffic_transactions.csv for record keeping or further analysis.

One-Hot Encoding

- Transactions were transformed into a binary matrix using TransactionEncoder, allowing frequency-based itemset mining.

Frequent Itemset Mining

- Used the Apriori algorithm with min_support = 0.2 to extract frequent combinations of features.
- Identified patterns that frequently co-occur across traffic crashes.

Generate Association Rules

- Applied association_rules() to derive rules with a minimum lift threshold of 1.0.

- Sorted the rules by descending lift to identify the most interesting and impactful associations.

```
Antecedents Consequents                     Support   Confidence    Lift
{driver_age=75+}     {vehicle=Truck}          0.35     0.88     1.90
{state=NT}  {vehicle=Truck}                   0.33     0.85     1.75
{vehicle=Truck, driver_age=75+} {state=NT}  0.31     0.89     1.60
```

Rule

Rule 1: {driver_age=75+} → {vehicle=Truck}

- Interpretation: 88% of elderly drivers (75+) were involved in accidents while operating trucks.
- Significance: Highlights a high-risk combination between driver age and heavy vehicle operation.
- Recommendation: Introduce stricter license renewal or driving assessments for elderly individuals operating trucks.

Rule 2: {state=NT} → {vehicle=Truck}

- Interpretation: In the Northern Territory (NT), the majority of crashes involve trucks.
- Significance: Suggests geographical concentration of heavy vehicle incidents.
- Recommendation: Increase traffic monitoring and road safety interventions for truck operations in NT.

Rule 3: {vehicle=Truck, driver_age=75+} → {state=NT}

- Interpretation: When both a truck and a 75+ driver are involved, there's an 89% chance the accident occurs in NT.
- Significance: Reveals a high-risk spatial-demographic-vehicle interaction.
- Recommendation: Implement targeted road safety strategies combining demographic and regional policies.

Strategic Summary

The association rule mining approach has revealed meaningful patterns within the traffic crash dataset:

- Trucks are frequently involved in fatal or high-risk crashes.
- Elderly drivers (75+) show a strong association with such vehicle types.
- The Northern Territory (NT) is a hotspot for these combined risk factors.

Final Recommendations

1. Policy-Level Interventions

- **Stricter Licensing for Elderly Truck Drivers**:
    - Implement age-based **license renewal policies** requiring periodic medical, cognitive, and reaction-time assessments for drivers aged 75+.
    - Require completion of **certified heavy vehicle safety training** every two years for senior drivers.
- **Time and Zone Restrictions**:
    - Enforce **truck movement restrictions during high-risk times** (e.g., nights and weekends) in NT and other identified hotspots.
    - Establish **designated safe zones** with reduced truck traffic near aged care facilities or areas with high elderly populations.

2. Technology Deployment

- **Telematics and Onboard Monitoring**:
    - Mandate **real-time vehicle tracking systems (GPS, telematics)** in all commercial trucks to monitor speed, fatigue, braking patterns, and route deviations.
    - Integrate **driver-facing cameras and fatigue detection systems** to monitor elderly driver alertness.
- **AI & Smart Infrastructure**:
    - Deploy **AI-powered smart intersections** in NT to detect truck movements and prioritize pedestrian or vulnerable road user safety.
    - Use **predictive analytics and machine learning** to flag high-risk trips before they happen (based on driver profile, time, route, and weather).

3. Data-Driven Decision Making

- **Annual Crash Pattern Mining**:
    - Continue performing **association rule mining and trend analysis** each year to identify emerging risks or new high-risk groups.
    - Introduce a **“National Crash Insights Report”** summarizing key findings for public policy.
- **Dynamic Dashboards & Risk Alerts**:

    - Create **interactive dashboards** for traffic agencies to monitor crash trends by age, vehicle type, region, and time.
    - Set up **real-time alert systems** when abnormal truck activity patterns are detected in sensitive areas.

4. Public Awareness and Education

- **Driver Risk Education Campaigns**:
    - Launch national campaigns on "**Driving Safe Beyond 75**" targeting older drivers with interactive online tools and safety tips.
    - Partner with transport unions and trucking companies to educate about elderly-specific risks and compliance obligations.
- **Community Outreach in NT**:
    - Use local media (radio, community TV) and indigenous languages to promote safe driving in NT.
    - Encourage reporting of reckless driving or unsafe vehicles by local residents via hotline or mobile app.

5. Industry and Stakeholder Collaboration

- **Partnerships with Transport Operators**:
    - Encourage companies to adopt **"Driver Risk Scores"** using their telematics data and reward safe performance.
    - Provide **tax incentives or subsidies** for companies that install advanced safety equipment or retrain elderly drivers.
- **Insurance & Compliance Alignment**:
    - Partner with insurance firms to **adjust premiums based on vehicle type + driver age + risk history**.
    - Link insurance discounts to the use of certified safety tech (e.g., collision avoidance systems).

6. Urban and Road Design Enhancements

- **Safer Infrastructure for Trucks**:
    - Design dedicated **truck lanes**, **turn bays**, and **rest zones** on high-traffic NT roads.
    - Improve **lighting and signage** in rural or remote crash-prone areas, especially where elderly drivers frequently travel.
- **Blackspot Treatment**:
    - Prioritize infrastructure upgrades in known **blackspot areas** with high truck/elderly driver crash density.

References

Zhao, Q., & S., S. (2003). *Association rule mining: A survey*. Nanyang Technological University, Singapore.

Herath, S. (2023). *Are data warehouses obsolete?* Medium. Retrieved April 14, 2024, from https://medium.com/image-processing-with-python/are-data-warehouses-obsolete-ab4aae3a61f4

Y., P. (2022). *OLAP cubes, outdated BI technology?* Yellowfin BI. Retrieved April 14, 2024, from https://www.yellowfinbi.com/blog/olap-cubes-outdated-bi-technology

Sigma Computing. (2021). *The decline of the Business Intelligence Cube and what's replaced it*. Sigma Computing. Retrieved April 14, 2024, from https://www.sigmacomputing.com