Project 2:

Graph Database Design and Cypher Query for Road Fatality Data

- **Unit Code: CITS5504**
- **Aparna Nair (23914381)**

Table of Contents

1. Introduction

1.1 Purpose

This project aims to design and implement a graph database model using real-world road crash data, enabling the exploration of complex, multidimensional relationships between crash-related entities such as location, vehicle types, road users, time, and crash characteristics. By leveraging the expressiveness of graph-based structures and the power of Cypher queries, the project demonstrates how graph databases can uncover insights that would be difficult or inefficient to derive using traditional relational models. The primary objective is to answer a set of provided and self-designed business questions that require advanced querying, path exploration, and pattern detection.

1.2 Dataset

The dataset used in this project is the **ARDD Fatality Dataset – December 2024**, which contains 10,490 records and 25 attributes. It includes detailed information on fatal road crashes in Australia between 2014 and 2024. Each row represents a unique combination of a crash and a person killed in that crash. The dataset has been cleaned to remove missing or ambiguous values and transformed into a graph-friendly structure.

Key data attributes include:

- Crash details: crash type, number of fatalities, time, date, and location
- Person details: gender, age, age group, road user type
- Vehicle involvement: bus, heavy rigid truck, articulated truck
- Contextual information: road type, remoteness area, speed limit, and holiday periods

1.3 Software and Tools Used:

The following tools and technologies were used throughout the project:

- **Neo4j Desktop**: To create and query the graph database using the Cypher query language.
- **Arrows App**: For visual graph modeling to design the property graph structure prior to implementation.
- **Python (Pandas)**: To perform the ETL (Extract, Transform, Load) process, including data cleaning, transformation, node/relationship extraction, and CSV generation.
- **Microsoft Excel**: Occasionally used for initial inspection and transformation of the dataset.

- **APOC Library** (Neo4j plugin): For advanced path querying and traversal beyond native Cypher capabilities.

2. Graph Database Design

2.1 Entity and Relationship Identification

The graph schema was designed around the central concept of a **crash event**, represented by a Crash node, and is linked to several dimension-like entities that describe its context. The main nodes and relationships identified from the dataset are as follows:

Nodes:

- **Crash**: Represents a unique crash event. Each crash contains the number of fatalities, crash type, month, and year.
- **Person**: Represents an individual killed in a crash, including demographic details like age, gender, age group, and road user type.
- **Vehicle**: Represents the combination of heavy vehicle types involved in the crash, such as buses, articulated trucks, and heavy rigid trucks.
- **Date**: Captures calendar and event-specific attributes, including the day of the week, whether it was a weekend, and if the crash occurred during Christmas or Easter periods.
- **Time**: Captures the time and time-of-day category (day/night) of the crash.
- **Location**: Represents the geographic context of the crash, including the LGA, SA4 region, state, remoteness classification, and road type.
- **SpeedZone**: Represents the posted speed limit at the site of the crash.

Relationships:

- (:Crash)-[:INVOLVED]->(:Person)
- (:Crash)-[:VEHICLE_USED]->(:Vehicle)
- (:Crash)-[:HAPPENED_ON]->(:Date)
- (:Crash)-[:OCCURRED_AT]->(:Time)
- (:Crash)-[:LOCATED_IN]->(:Location)
- (:Crash)-[:SPEED_ZONE]->(:SpeedZone)

This star-schema-like design ensures clarity and supports flexible, efficient querying for multidimensional analysis.

2.2 Data Modelling Technique

The graph was modeled using the **property graph model**, where:

- **Nodes** represent entities, each with a unique label (e.g., Crash, Person).
- **Relationships** connect nodes with meaningful types (e.g., INVOLVED), and are directed.
- Both nodes and relationships can hold **properties** (e.g., Crash.num_fatalities, Person.age).

This model was chosen because it allows for intuitive representation of the real-world domain and supports efficient traversal and pattern matching.

Dimensional Separation & Justification:

The model adopts a **dimensional approach**, similar to a star schema:

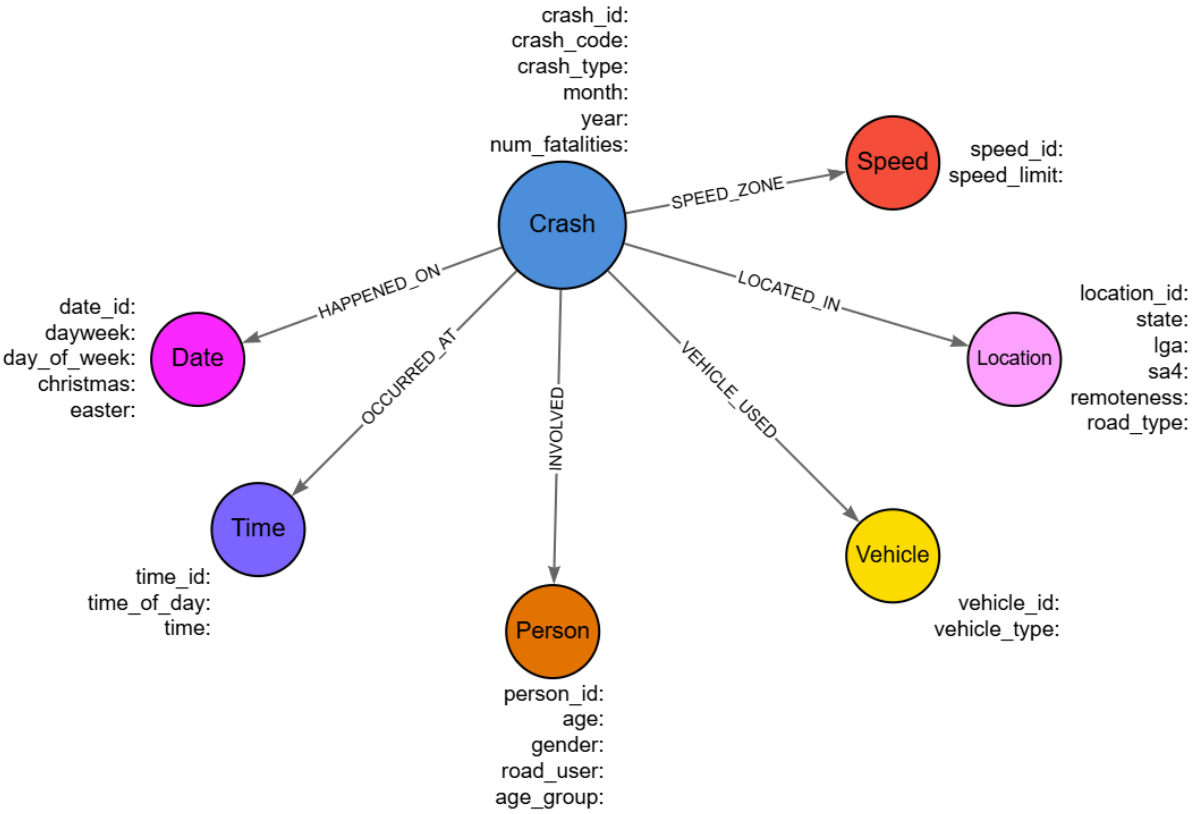
- Person, Time, Location, etc., are modeled as **dimension nodes** connected to the Crash fact node.
- This separation allows for flexibility in queries across dimensions, e.g., demographic trends, spatial analysis, or time-based filtering.

Surrogate Keys & Normalization:

- **Surrogate keys** (e.g., p1, c1, l1) were used to uniquely identify nodes and avoid reliance on natural keys that could be duplicated.
- **Normalization** was applied to eliminate redundancy — for instance, recurring values in Date, Time, and SpeedZone were deduplicated and stored as separate nodes.
- **Denormalization** was used for Vehicle by collapsing related vehicle type flags into a single descriptive property (e.g., "bus and articulated truck involved").

This balance between normalization and simplification improved query performance and reduced data volume.

2.3 Arrows App Design Diagram



The diagram shows:

- A central Crash node connected to six primary dimensions
- Clearly labeled relationships (e.g., INVOLVED, LOCATED_IN, etc.)
- All nodes modeled with relevant properties to support complex analysis

2.4 Design Justification (Pros & Cons)

Design Choice / Node	Justification	Pros	Cons
Crash	Central fact node that connects to all other entities	Enables star-schema design and simplifies querying from a crash-centric perspective	May need joins to extract full context of a crash

Person	Captures demographic details of individuals involved in crashes	Enables demographic analysis (e.g., gender, age group, road user type)	Requires deduplication if the same person appears in multiple crashes
Vehicle	Created to summarize combinations of heavy vehicle involvement	Simplifies filtering by complex vehicle types (e.g., bus + truck involved)	Less granular than modeling each vehicle type as a separate entity
Location	Models LGA, SA4, state, road type, and remoteness	Supports rich spatial analysis and regional comparisons	Doesn't allow LGA-to-LGA traversal unless explicitly linked
Date	Captures day of the week, holiday flags, and weekday/weekend	Useful for temporal filtering (e.g., holiday crashes, weekends)	Doesn't include month or year, which are modeled in the Crash node (explained below)
Time	Captures time and whether it occurred during day or night	Allows analysis of crashes by time of day	Requires additional logic to classify time_of_day
SpeedZone	Models the posted speed limit of the crash location	Normalizes repeated speed values; enables filtering/grouping by speed	Adds a join step when querying crashes by speed
Crash.month & Crash.year (not in Date node)	Month and year were stored in the Crash node instead of the Date node because they describe the crash occurrence directly and are not deduplicated.	Avoids needlessly inflating the Date node with near-unique combinations	Slight separation of temporal details between nodes; small overhead in remembering which part is where

3. ETL Process

3.1 Overview

The ETL (Extract, Transform, Load) process was central to converting the tabular road fatality dataset into a graph-ready format. This transformation was implemented in Python using the pandas library and involved the following steps:

Extract

The dataset was sourced from the **ARDD: Fatalities—December 2024** CSV file, which contains 10,490 records and 25 fields. Each row represents a fatality record associated with a specific crash.

Transform

The transformation process involved:

- **Cleaning**: Rows with missing or ambiguous values were dropped.
- **Deduplication**: Columns like Dayweek, Time, Speed Limit, and Location were deduplicated to create unique entries for corresponding dimension tables (nodes).
- **Surrogate Key Generation**: Each node type was assigned a unique ID (e.g., p1 for persons, l1 for locations) to ensure proper linking.
- **Derived Fields**:
    - A composite vehicle_type field was created from boolean flags (e.g., "Bus Involvement", "Articulated Truck Involvement").
    - A person_key was computed using Gender, Age, and Road User to track potential person deduplication across crashes.
    - Daytime category (time_of_day) was derived from the crash time (before/after 6 PM).

Load

Each node and relationship was saved as a separate CSV file. These CSVs were then imported into Neo4j using Cypher's LOAD CSV WITH HEADERS syntax. The separation of nodes and relationships aligned with Neo4j's property graph structure and allowed efficient indexing and traversal.

3.2 Python Scripting

The entire transformation was scripted in Python using pandas to automate:

- Column selection and deduplication

- Surrogate ID assignment
- Creation of relationship tables

Following are some of the snippets from the ETL script:

Generate vehicle_type column:

This transformation was done to combine multiple binary vehicle involvement columns into a single descriptive vehicle_type attribute, making the data more interpretable and enabling simpler, more expressive queries in Neo4j.

```
def determine_vehicle_type(row):
    bus = row['Bus Involvement'] == 'Yes'
    heavy = row['Heavy Rigid Truck Involvement'] == 'Yes'
    articulated = row['Articulated Truck Involvement'] == 'Yes'
    if bus and articulated:
        return "bus and articulated truck involved"
    elif bus and heavy:
        return "bus and heavy rigid truck involved"
    elif articulated and heavy:
        return "articulated and heavy rigid truck involved"
    elif bus:
        return "only bus involved"
    elif heavy:
        return "only heavy rigid involved"
    elif articulated:
        return "only articulated truck involved"
    else:
        return "no heavy vehicles involved"

df['vehicle_type'] = df.apply(determine_vehicle_type, axis=1)
vehicle_df =
df[['vehicle_type']].drop_duplicates().reset_index(drop=True)
vehicle_df['vehicle_id'] = ['v' + str(i+1) for i in
range(len(vehicle_df))]
vehicle_df.to_csv("nodes_vehicle.csv", index=False)
```

Examples of creating unique id for each nodes and removing duplicates:

- **Create "Person" node:**

```
person_df = df[['Road User', 'Gender', 'Age', 'Age
Group']].drop_duplicates().reset_index(drop=True)
person_df['person_id'] = ['p' + str(i+1) for i in
range(len(person_df))]
person_df.to_csv("nodes_person.csv", index=False)
```

Examples of creating relationship CSVs:

- **Create "INVOLVED" relationship:**

```
rel_person = df.merge(person_df, on=['Road User', 'Gender', 'Age',
'Age Group'])
rel_person_df = rel_person[['ID',
'person_id']].rename(columns={'ID': 'crash_id'})
rel_person_df.to_csv("rel_involved.csv", index=False)
```

3.3 Output Summary

A total of **7 node CSV files** and **6 relationship CSV files** were generated:

Node CSV Files:

CSV File	Label	Primary Key	Attributes / Columns
nodes_crash.csv	Crash	crash_id	crash_id, crash_code, crash_type, month, year, num_fatalities
nodes_person.csv	Person	person_id	person_id, road_user, gender, age, age_group
nodes_vehicle.csv	Vehicle	vehicle_id	vehicle_id, vehicle_type
nodes_date.csv	Date	date_id	date_id, dayweek, weekday_weekend, christmas, easter
nodes_time.csv	Time	time_id	time_id, time, time_of_day
nodes_location.csv	Location	location_id	location_id, state, lga, sa4, remoteness, road_type
nodes_speedzone.csv	SpeedZone	speed_id	speed_id, speed_limit

Relationship CSV Files:

CSV File	Relationship Type	From → To	Columns
rel_involved.csv	INVOLVED	Crash→Person	crash_id, person_id
rel_vehicle_used.csv	VEHICLE_USED	Crash→Vehicle	crash_id, vehicle_id
rel_happened_on.csv	HAPPENED_ON	Crash → Date	crash_id, date_id
rel_occurred_at.csv	OCCURRED_AT	Crash → Time	crash_id, time_id

rel_located_in.csv	LOCATED_IN	Crash→Location	crash_id, location_id
rel_speed_zone.csv	SPEED_ZONE	Crash→ SpeedZone	crash_id, speed_id

- All surrogate keys (e.g., p1, v1, etc.) are **generated automatically** during ETL to uniquely identify each node.
- Each file was tested for structural consistency and successfully loaded into Neo4j using LOAD CSV statements, ensuring that all nodes and relationships were correctly instantiated in the graph.

4. Graph Database Implementation

Once the CSV files were generated through the ETL process, the graph database was implemented in Neo4j using **Cypher scripts**. The process involved creating all nodes and relationships using LOAD CSV WITH HEADERS and verifying the structure using built-in Neo4j metadata queries.

4.1 Node Creation in Neo4j

Each node type was loaded from its respective CSV file using LOAD CSV and CREATE.

Below is an example:

Loading CSV for Crash Nodes:

```
LOAD CSV WITH HEADERS FROM 'file:///nodes_crash.csv' AS row
CREATE (:Crash {
  crash_id: toInteger(row.crash_id),
  crash_code: row.crash_code,
  crash_type: row.crash_type,
  month: toInteger(row.month),
  year: toInteger(row.year),
  num_fatalities: toInteger(row.num_fatalities)
});
```

Repeat similar statements for Persons, Locations, Date, Time, Vehicle and SpeedZone.

4.2 Relationship Creation

Relationships were created using MATCH to find source and target nodes and then CREATE to link them.

Below is an example:

Crash → Person (INVOLVED)

```
LOAD CSV WITH HEADERS FROM 'file:///rel_involved.csv' AS row
MATCH (c:Crash {crash_id: toInteger(row.crash_id)})
MATCH (p:Person {person_id: row.person_id})
CREATE (c)-[:INVOLVED]->(p);
```

Repeat for other relationships: LOCATED_IN, VEHICLE_USED, OCCURRED_AT, HAPPENED_ON and SPEED_ZONE.

4.3 Verifying Graph Structure

After node and relationship creation, Neo4j's built-in functions were used to validate the graph:

Labels and Node Counts

```
CALL db.labels();
```

Output:

```
╒═══════════╕
│label      │
╞═══════════╡
│"Crash"    │
├───────────┤
│"Date"     │
├───────────┤
│"Time"     │
├───────────┤
│"Vehicle"  │
├───────────┤
│"Person"   │
├───────────┤
│"Location" │
├───────────┤
│"SpeedZone"│
└───────────┘
```

```
MATCH (n) RETURN labels(n)[0] AS Label, count(*) AS Count ORDER BY
Label;
```

Output:

Label	Count
"Crash"	10490
"Date"	25
"Location"	2278
"Person"	821
"SpeedZone"	16
"Time"	1333
"Vehicle"	7

Relationships and Types

```
CALL db.relationshipTypes();
```

Output:

relationshipType
"HAPPENED_ON"
"OCCURRED_AT"
"VEHICLE_USED"
"INVOLVED"
"LOCATED_IN"
"SPEED_ZONE"

```
MATCH ()-[r]->() RETURN type(r) AS Relationship, count(*) AS Count
ORDER BY Relationship;
```

Output:

```
╒══════════════╤═════╕
│Relationship  │Count│
╞══════════════╪═════╡
│"HAPPENED_ON" │10490│
├──────────────┼─────┤
│"INVOLVED"    │10490│
├──────────────┼─────┤
│"LOCATED_IN"  │10490│
├──────────────┼─────┤
│"OCCURRED_AT" │10490│
├──────────────┼─────┤
│"SPEED_ZONE"  │10490│
├──────────────┼─────┤
│"VEHICLE_USED"│10490│
└──────────────┴─────┘
```

5. Cypher Query Results

5.1 Queries given in the project

Query (a): WA Crashes with Articulated Trucks and Multiple Fatalities (2020–2024)

```
MATCH (c:Crash)-[:VEHICLE_USED]->(v:Vehicle)
WHERE v.vehicle_type CONTAINS "articulated"
  AND c.year >= 2020 AND c.year <= 2024
  AND c.num_fatalities > 1

MATCH (c)-[:INVOLVED]->(p:Person)
MATCH (c)-[:LOCATED_IN]->(l:Location)
WHERE l.state = 'WA'

RETURN
  p.road_user AS RoadUser,
  p.age AS Age,
  p.gender AS Gender,
  l.lga AS LGA_Name,
  c.month AS Month,
  c.year AS Year,
  c.num_fatalities AS TotalFatalities
ORDER BY c.year, c.month
```

Output:

RoadUser	Age	Gender	LGA_Name	Month	Year	TotalFatalities
"Driver"	58	"Female"	"Busselton"	11	2020	2
"Passenger"	51	"Female"	"Busselton"	11	2020	2
"Driver"	56	"Male"	"Dundas"	12	2020	2
"Driver"	58	"Male"	"Dundas"	12	2020	2

Key Observations:

- All multi-fatality crashes involving articulated trucks in WA (2020–2024) occurred in just two LGAs: *Busselton* and *Dundas*.
- Both crashes involved older individuals (aged 51–58), suggesting a potential vulnerability in older age groups.
- The involvement of both drivers and passengers indicates the severity and impact across multiple roles in the vehicle.

Query (b): Motorcycle riders during Christmas/Easter in inner regional areas

```
MATCH (c:Crash)-[:INVOLVED]->(p:Person)
WHERE toLower(p.road_user) CONTAINS "motorcycle"

MATCH (c)-[:LOCATED_IN]->(l:Location)
WHERE l.remoteness = 'Inner Regional Australia'

MATCH (c)-[:HAPPENED_ON]->(d:Date)
WHERE d.christmas = 'Yes' OR d.easter = 'Yes'

RETURN
  p.gender AS Gender,
  MAX(p.age) AS MaxAge,
  MIN(p.age) AS MinAge
ORDER BY Gender;
```

Output:

Gender	MaxAge	MinAge
"Male"	73	14

Key Observations:

- Only male motorcycle riders were involved in fatal crashes during the Christmas or Easter periods in inner regional Australia.
- The age range was wide — from 14 to 73 years, highlighting risk across both young and senior male riders.
- The absence of female entries suggests a gender disparity in motorcycle fatalities during holiday periods in these regions.

Query (c): Young drivers on weekends vs weekdays in 2024

```
MATCH (c:Crash)-[:INVOLVED]->(p:Person)
WHERE p.age_group = '17_to_25'
  AND p.road_user = 'Driver'
  AND c.year = 2024

MATCH (c)-[:HAPPENED_ON]->(d:Date)
MATCH (c)-[:LOCATED_IN]->(l:Location)

WITH
  l.state AS State,
  d.weekday_weekend AS DayType,
  p.age AS Age

RETURN
  State,
  COUNT(CASE WHEN DayType = 'Weekend' THEN 1 END) AS WeekendCount,
  COUNT(CASE WHEN DayType = 'Weekday' THEN 1 END) AS WeekdayCount,
  ROUND(AVG(Age), 1) AS AverageAge
ORDER BY State;
```

Output:

State	WeekendCount	WeekdayCount	AverageAge
"NSW"	13	19	20.9
"QLD"	8	14	20.0
"SA"	1	4	20.6
"TAS"	0	2	22.0
"VIC"	7	13	21.4

Key Observations:

- NSW recorded the highest number of young driver fatalities in both weekends and weekdays.
- Fatal crashes were more common on weekdays across all states.
- The average age was consistently close to 21 years, with TAS having the highest at 22.0, despite very low counts.
- States like SA and TAS had very low weekend involvement, potentially indicating safer regional driving or smaller populations.

Query (d): WA Friday crashes categorized as weekend with both male and female victims

```
MATCH (c:Crash)-[:LOCATED_IN]->(l:Location)
WHERE l.state = 'WA' AND c.num_fatalities > 1

MATCH (c)-[:HAPPENED_ON]->(d:Date)
WHERE toLower(d.dayweek) = 'friday' AND
toLower(d.`weekday_weekend`) = 'weekend'

MATCH (c)-[:INVOLVED]->(p:Person)

WITH c.crash_code AS crash_event,
     l.sa4 AS SA4_Name,
     l.remoteness AS Remoteness,
```

```
     l.road_type AS Road_Type,
     COLLECT(DISTINCT toLower(trim(p.gender))) AS genders

WHERE 'male' IN genders AND 'female' IN genders

RETURN DISTINCT
  crash_event,
  SA4_Name,
  Remoteness,
  Road_Type
ORDER BY crash_event;
```

Output:

crash_event	SA4_Name	Remoteness	Road_Type
"20155048"	"Western Australia - Outback (North)"	"Very Remote Australia"	"National or State Highway"
"20195098"	"Perth - South East"	"Major Cities of Australia"	"Local Road"

Key Observations:

- Fatal crashes involving both male and female victims on Fridays (categorized as weekends) occurred in both urban (*Perth – South East*) and remote (*Outback North*) areas.
- The incidents spanned different road types — a local road in a major city and a highway in a very remote area, showing that such crashes are not restricted to a single environment.
- This suggests a need for targeted interventions in both metro and remote regions.

Query (e): Top 5 SA4 regions for peak-hour crashes

```
MATCH (c:Crash)-[:OCCURRED_AT]->(t:Time),
      (c)-[:LOCATED_IN]->(l:Location)
WHERE c.num_fatalities > 0
  AND t.time IS NOT NULL
WITH l.sa4 AS sa4_region,
     CASE
       WHEN time(t.time) >= time("07:00") AND time(t.time) <
time("09:00") THEN "Morning Peak"
       WHEN time(t.time) >= time("16:00") AND time(t.time) <
time("18:00") THEN "Afternoon Peak"
       ELSE NULL
     END AS peak_period,
     c.crash_code AS crash_event
WHERE peak_period IS NOT NULL
```

```
WITH sa4_region, peak_period, COUNT(DISTINCT crash_event) AS
crash_count
WITH sa4_region,
     SUM(CASE WHEN peak_period = "Morning Peak" THEN crash_count
ELSE 0 END) AS morning_peak,
     SUM(CASE WHEN peak_period = "Afternoon Peak" THEN crash_count
ELSE 0 END) AS afternoon_peak,
     SUM(crash_count) AS total_crashes
ORDER BY total_crashes DESC
LIMIT 5
RETURN sa4_region AS sa4,
       morning_peak AS `Morning Peak`,
       afternoon_peak AS `Afternoon Peak`
```

Output:

sa4	Morning Peak	Afternoon Peak
"Melbourne - South East"	22	34
"South Australia - South East"	22	31
"Wide Bay"	21	32
"New England and North West"	18	32
"Capital Region"	22	27

Key Observations:

- Melbourne – South East recorded the highest number of peak-hour crashes, especially during afternoon peak (34).
- All top 5 SA4 regions show a consistent pattern of more crashes in the afternoon than in the morning.
- Capital Region is the only region in the top 5 where morning and afternoon crashes are nearly balanced.
- The results suggest afternoon commutes are generally riskier, possibly due to fatigue, traffic congestion, or distraction.

Query (f): LGA-to-LGA paths of length 3

```
MATCH path = (l1:Location)-[*3]-(l2:Location)
```

```
WHERE l1.location_id <> l2.location_id

WITH l1.lga AS startLGA, l2.lga AS endLGA, path
ORDER BY startLGA, endLGA
LIMIT 3

RETURN startLGA, endLGA, path;
```

Output:

```
(no changes, no records)
```

Key Observations:

- The query returned no results, indicating that there are no 3-hop paths between LGAs in the current graph model.
- This is expected because the LGA is modeled as a property of the Location node, which is directly connected only to crashes and not to other LGAs.
- Without shared intermediate nodes or explicit LGA-to-LGA relationships, path traversal across LGAs is not supported.

Query (g): Weekday pedestrian crashes with heavy vehicles in speed zones < 40 or ≥ 100

```
MATCH (c:Crash)-[:INVOLVED]->(p:Person),
      (c)-[:HAPPENED_ON]->(d:Date),
      (c)-[:VEHICLE_USED]->(v:Vehicle),
      (c)-[:SPEED_ZONE]->(s:SpeedZone),
      (c)-[:OCCURRED_AT]->(t:Time)
WHERE toLower(p.road_user) = 'pedestrian'
  AND toLower(d.weekday_weekend) = 'weekday'
  AND (
    toLower(v.vehicle_type) CONTAINS 'bus' OR
    toLower(v.vehicle_type) CONTAINS 'heavy rigid'
  )
  AND (
    toInteger(s.speed_limit) < 40 OR
    toInteger(s.speed_limit) >= 100
```

```
  )
WITH
  t.time_of_day AS time_of_day,
  p.age_group AS age_group,
  CASE
    WHEN toLower(v.vehicle_type) CONTAINS 'bus' THEN 'Bus'
    ELSE 'Heavy Rigid Truck'
  END AS vehicle_type,
  toInteger(s.speed_limit) AS speed_limit,
  COUNT(DISTINCT c.crash_code) AS crash_count
RETURN
  time_of_day,
  age_group,
  vehicle_type,
  speed_limit,
  crash_count
ORDER BY time_of_day ASC, age_group ASC;
```

Output:

time_of_day	age_group	vehicle_type	speed_limit	crash_count
"Day"	"0_to_16"	"Heavy Rigid Truck"	110	1
"Day"	"17_to_25"	"Heavy Rigid Truck"	20	1
"Day"	"26_to_39"	"Heavy Rigid Truck"	100	2
"Day"	"40_to_64"	"Heavy Rigid Truck"	100	1
"Day"	"40_to_64"	"Heavy Rigid Truck"	110	2
"Day"	"40_to_64"	"Bus"	10	1
"Day"	"75_or_older"	"Heavy Rigid Truck"	100	1
"Day"	"75_or_older"	"Bus"	10	1
"Night"	"17_to_25"	"Heavy Rigid Truck"	110	1
"Night"	"26_to_39"	"Heavy Rigid Truck"	100	1
"Night"	"40_to_64"	"Heavy Rigid Truck"	100	1

Key Observations:

- Most crashes occurred during the day, particularly among age groups 40_to_64 and 75_or_older, often involving heavy rigid trucks.
- Low-speed bus-related crashes (10 km/h) were exclusively daytime events, likely in urban or pedestrian-heavy zones.
- Nighttime crashes were fewer overall but still involved young to middle-aged groups and occurred in high-speed zones (≥100 km/h).
- The 40_to_64 group had the broadest involvement, across both day and night and both vehicle types, marking them as a consistently at-risk demographic.

5.2 Self- designed Queries

Query 1: In which remoteness areas do young male drivers (17–25) most frequently die in crashes involving articulated trucks during night hours, and how does it compare to female victims?

```
MATCH (c:Crash)-[:INVOLVED]->(p:Person),
      (c)-[:VEHICLE_USED]->(v:Vehicle),
      (c)-[:OCCURRED_AT]->(t:Time),
      (c)-[:LOCATED_IN]->(l:Location)
WHERE p.age_group = '17_to_25'
  AND p.road_user = 'Driver'
  AND t.time_of_day = 'Night'
  AND toLower(v.vehicle_type) CONTAINS 'articulated'

RETURN
  l.remoteness AS Remoteness,
  COUNT(DISTINCT CASE WHEN p.gender = 'Male' THEN c.crash_code
END) AS MaleCrashCount,
  COUNT(DISTINCT CASE WHEN p.gender = 'Female' THEN c.crash_code
END) AS FemaleCrashCount
ORDER BY (MaleCrashCount + FemaleCrashCount) DESC;
```

Output:

Remoteness	MaleCrashCount	FemaleCrashCount
"Inner Regional Australia"	9	3
"Outer Regional Australia"	10	1
"Major Cities of Australia"	4	0
"Remote Australia"	1	0
"Very Remote Australia"	0	1

Importance of Query and Key Observations:

- Outer and Inner Regional Australia recorded the highest number of crashes involving young male drivers and articulated trucks at night, with 10 and 9 crashes respectively — indicating these regions are high-risk zones.
- Inner Regional Australia is the only area with notable female involvement (3 crashes), while Very Remote Australia had 1 female crash and no male crashes.
- Urban areas like Major Cities of Australia showed fewer incidents, suggesting regional and remote driving conditions play a significant role in these fatal crashes.
- The query highlights the need for region-specific road safety policies for young drivers and heavy vehicle interactions, especially in regional and remote areas where night travel is more common.

Query 2: Across all states, how many fatal crashes involving any heavy vehicles (bus, heavy rigid truck or articulated truck) during Easter or Christmas periods occurred on national/state highways, and what were the average ages of the victims?

```
MATCH (c:Crash)-[:VEHICLE_USED]->(v:Vehicle),
      (c)-[:LOCATED_IN]->(l:Location),
      (c)-[:HAPPENED_ON]->(d:Date),
      (c)-[:INVOLVED]->(p:Person)
WHERE
  NOT toLower(v.vehicle_type) CONTAINS "no heavy vehicles
involved"
  AND (d.christmas = 'Yes' OR d.easter = 'Yes')
  AND toLower(l.road_type) CONTAINS "highway"

RETURN
  l.state AS State,
  COUNT(DISTINCT c.crash_code) AS CrashCount,
```

```
  ROUND(AVG(p.age), 1) AS AvgVictimAge
ORDER BY CrashCount DESC;
```

Output:

State	CrashCount	AvgVictimAge
"SA"	6	44.7
"NSW"	6	49.0
"QLD"	5	28.9
"VIC"	3	50.0
"WA"	2	69.0
"TAS"	1	61.0

Importance of Query and Key Observations:

- South Australia (SA) and New South Wales (NSW) recorded the highest number of such crashes (6 each), showing a concentration of holiday-related bus fatalities in these states.
- Queensland (QLD) had a noticeably lower average victim age (28.9), suggesting younger individuals were more affected in this state compared to others.
- States like Western Australia (WA) and Tasmania (TAS) had fewer incidents but the highest average victim ages (69.0 and 61.0), indicating a vulnerability among older populations.
- These findings highlight the importance of state-specific bus safety strategies during holidays, particularly for younger passengers in QLD and elderly passengers in WA and TAS.

Query 3: Which combinations of age group and road user type are most at risk in fatal crashes involving articulated trucks, specifically in very remote areas, and how do they differ between day and night?

```
MATCH (c:Crash)-[:INVOLVED]->(p:Person),
      (c)-[:VEHICLE_USED]->(v:Vehicle),
      (c)-[:LOCATED_IN]->(l:Location),
      (c)-[:OCCURRED_AT]->(t:Time)
WHERE toLower(v.vehicle_type) CONTAINS 'articulated'
  AND l.remoteness = 'Very Remote Australia'

RETURN
  t.time_of_day AS TimeOfDay,
  p.age_group AS AgeGroup,
  p.road_user AS RoadUserType,
  COUNT(DISTINCT c.crash_code) AS CrashCount
ORDER BY CrashCount DESC;
```

Output:

TimeOfDay	AgeGroup	RoadUserType	CrashCount
"Day"	"40_to_64"	"Driver"	15
"Day"	"26_to_39"	"Driver"	5
"Night"	"26_to_39"	"Driver"	5
"Night"	"40_to_64"	"Driver"	5
"Night"	"65_to_74"	"Driver"	4
"Day"	"17_to_25"	"Driver"	3
"Day"	"40_to_64"	"Passenger"	3
"Day"	"65_to_74"	"Driver"	3
"Day"	"26_to_39"	"Passenger"	2
"Day"	"17_to_25"	"Passenger"	2
"Day"	"40_to_64"	"Motorcycle rider"	1

Importance of Query and Key Observations:

- Daytime crashes involving drivers aged 40–64 are the most frequent, with 15 fatalities, indicating this group is at highest risk in very remote areas involving articulated trucks.
- Drivers aged 26–39 and 40–64 also appear prominently in night-time crashes, with 5 crashes each, showing that the risk continues beyond daytime hours for middle-aged drivers.
- There is also notable presence of passengers and pedestrians across multiple age groups, especially 17–25 and 26–39, suggesting that articulated truck crashes in remote areas affect a variety of road users — not just drivers.

- The data reveals that night-time risks are more evenly distributed across age groups and user types, while daytime crashes are more concentrated among older drivers, calling for time- and role-specific road safety interventions in remote zones.

Query 4: Which LGAs recorded the highest number of multi-fatality crashes (fatalities > 1) during weekends, and what were the predominant vehicle types involved?

```
MATCH (c:Crash)-[:HAPPENED_ON]->(d:Date),
      (c)-[:LOCATED_IN]->(l:Location),
      (c)-[:VEHICLE_USED]->(v:Vehicle)
WHERE d.weekday_weekend = 'Weekend'
  AND toInteger(c.num_fatalities) > 1

RETURN
  l.lga AS LGA,
  v.vehicle_type AS VehicleType,
  COUNT(DISTINCT c.crash_code) AS MultiFatalCrashes
ORDER BY MultiFatalCrashes DESC
LIMIT 10;
```

Output:

LGA	VehicleType	MultiFatalCrashes
"Bundaberg"	"no heavy vehicles involved"	6
"Moreton Bay"	"no heavy vehicles involved"	6
"Central Coast (NSW)"	"no heavy vehicles involved"	4
"Logan"	"no heavy vehicles involved"	4
"Barkly"	"no heavy vehicles involved"	4
"Wattle Range"	"no heavy vehicles involved"	4
"Gold Coast"	"no heavy vehicles involved"	3
"Brisbane"	"no heavy vehicles involved"	3
"Wyndham"	"no heavy vehicles involved"	3
"Muswellbrook"	"no heavy vehicles involved"	3

Importance of Query and Key Observations:

- Bundaberg and Moreton Bay topped the list, each with 6 multi-fatality crashes, all involving no heavy vehicles — indicating that severe weekend crashes in these areas often involve light vehicles.
- Other LGAs with high counts include Central Coast (NSW), Logan, Barkly, and Wattle Range, each with 4 such crashes, also not involving heavy vehicles.
- The complete dominance of “no heavy vehicles involved” across all top LGAs suggests that passenger vehicles are the primary contributors to weekend multi-fatality events in these areas.
- This highlights the need for targeted road safety interventions in high-risk LGAs focused on private vehicle driver behavior during weekends — such as anti-speeding campaigns, fatigue awareness, and community enforcement.

6. Graph Data Science Discussion

6.1 Background and Motivation

Traditional crash analysis often uses tabular methods to assess individual factors like speed limits, vehicle types, or victim demographics. However, road crashes are multi-faceted and interconnected a single crash involves a time, location, multiple people, and vehicles. These elements often recur across incidents in complex ways.

Graph Data Science (GDS) enables a shift from isolated analysis to one that captures the relational structure of crash data modeling entities as nodes and their connections as relationships [1]. This allows exploration not just of what happened, but how crash characteristics are structurally linked across dimensions like time, people, and vehicles.

The Neo4j Graph Data Science (GDS) library provides advanced algorithms to extract insight from such structures, including [1]:

- **Pathfinding** (e.g., BFS, A*) to explore connections
- **Centrality** (e.g., Betweenness, PageRank) to identify key nodes
- **Community detection** to find groups of similar entities
- **Similarity and classification** for clustering and predictive modelling

6.2 Why Community Detection?

Community detection is ideal for analyzing time-based patterns in road fatalities. By treating each Time node as an entity and connecting them based on shared crash characteristics (e.g., involving the same vehicle types in 2023–2024), we can uncover structural similarities between time slots.

This lets us answer questions like:

- Which times of day exhibit similar crash behavior?
- Are late-night and early-morning crashes structurally linked?
- Do certain periods form high-risk communities even if they are not adjacent in the clock?

Rather than simply grouping hours by day/night, community detection can reveal hidden clusters [2] of crash times that behave similarly enabling data-driven time-based interventions.

6.3 Why the Louvain Algorithm?

The Louvain algorithm, introduced by Blondel et al. [4], is a widely used method for community detection. It optimizes modularity a measure proposed by Newman [3] to find communities that are densely connected internally but loosely connected to others.

Louvain is especially suited to our dataset because:

- It scales efficiently to **large, sparse graphs**
- It is **unsupervised** and requires no labels
- It produces a **hierarchical clustering**, enabling multi-level analysis

Given that Time nodes are sparsely but meaningfully connected via shared crash patterns, Louvain is a natural choice to detect structurally similar time periods.

6.4 Suitability for This Dataset

In our graph model:

- Each Time node represents a specific hh:mm crash time (e.g., "07:45", "21:15")
- Nodes are connected by SIMILAR_TIME relationships if crashes at those times involved the same vehicle types during 2023–2024
- This creates a structural similarity graph, not based on clock proximity, but on crash behavior

Running community detection over this graph enables discovery of behavioral clusters of crash times. For example, crashes at 07:30 and 16:45 may be grouped if they involve similar crash patterns despite being in different parts of the day.

Such insights are valuable for:

- Targeting peak-risk windows more precisely
- Designing time-specific safety campaigns
- Informing policing and emergency service scheduling

6.5 Practical Example and Query

We first created SIMILAR_TIME relationships between time nodes for crashes involving the same vehicle type during 2023 and 2024. Then we projected this graph into the GDS engine and ran Louvain community detection.

```
// Create SIMILAR_TIME relationships
MATCH (t1:Time)<-[:OCCURRED_AT]-(c1:Crash)-[:VEHICLE_USED]-
>(v:Vehicle),
      (t2:Time)<-[:OCCURRED_AT]-(c2:Crash)-[:VEHICLE_USED]->(v)
WHERE t1 <> t2 AND c1.year IN [2023, 2024] AND c2.year IN [2023,
2024]
MERGE (t1)-[:SIMILAR_TIME]-(t2);

// Project the graph
CALL gds.graph.project(
  'timeGraph',
  'Time',
  {
    SIMILAR_TIME: {
      type: 'SIMILAR_TIME',
      orientation: 'UNDIRECTED'
    }
```

```
  }
);

// Run Louvain community detection
CALL gds.louvain.stream('timeGraph')
YIELD nodeId, communityId
RETURN gds.util.asNode(nodeId).time AS Time, communityId
ORDER BY communityId, Time;
```

6.6 Real-World Application

Let's say the Louvain algorithm groups together night-time (e.g., 02:00, 03:30) and early morning times (e.g., 06:00, 07:00). This suggests these hours share structural risk characteristics, such as:

- Low visibility
- Fatigued drivers
- Commercial vehicle involvement

With this knowledge, policymakers can:

- Focus educational campaigns around specific hours, not just "night"
- Allocate resources for increased patrols or lighting improvements in those periods
- Identify emerging peak-risk slots that might be overlooked in traditional "peak vs. off-peak" categorization

The Louvain algorithm thus empowers a structurally informed, time-targeted approach to road safety.

7. Conclusion

This project demonstrated the power of graph-based modeling for uncovering complex spatial-temporal patterns in fatal road crashes. By transforming tabular crash data into a property graph and leveraging Neo4j's Graph Data Science tools, we were able to answer nuanced queries that span across time, location, vehicle involvement, and demographic attributes. The use of community detection, particularly on time-based similarities, revealed hidden risk clusters that traditional methods may overlook. Overall, the graph approach provided a flexible and insightful framework for analyzing interconnected crash factors, supporting more targeted and informed road safety strategies.

References

[1] Neo4j. (2024). *Graph data science documentation*. Neo4j. https://neo4j.com/docs/graph-data-science/current/

[2] Neo4j. (2024). *Louvain modularity optimization algorithm*. Neo4j. https://neo4j.com/docs/graph-data-science/current/algorithms/louvain/

[3] Newman, M. E. J. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences, 103*(23), 8577–8582. https://doi.org/10.1073/pnas.0601602103

[4] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment, 2008*(10), P10008. https://doi.org/10.1088/1742-5468/2008/10/P10008

[5] Lu, H., Halappanavar, M., & Kalyanaraman, A. (2014). Parallel heuristics for scalable community detection. *arXiv*. https://arxiv.org/abs/1410.1237