Project 2 – Graph Database Design and Cypher Query

Name: Isaiah Rama Veera | Student_ID:24078803

Table of Contents

1. Introduction and Objectives

This report presents a comprehensive, end-to-end graph-database solution for analysing Australian fatal-crash data spanning 2014–2024.

Objectives

- **Design** and document a Neo4j property-graph schema capable of answering every required analytical question
- **Develop** a Python-based ETL pipeline to cleanse the raw CSV files and produce fully normalised, Neo4j-ready data extracts
- **Load** the cleansed data into Neo4j, enforcing strict uniqueness constraints and creating performance-oriented indexes
- **Author and execute** the seven assessment Cypher queries (A–G) and derive two additional, value-added insights
- **Demonstrate** a Graph Data Science (GDS) workflow built on the same schema for advanced risk discovery and modelling

2. Dataset Overview & Data Dictionary

- **Source:** Australian Road Deaths Database (ARDD), *Fatalities—December 2024* release (post-correction) [1]
- **Volume:** 10490 records × 25 attributes (after cleansing)
- **Granularity:** One record per person involved in a single fatal-crash event
- **Key attributes:**
    - **Identifiers:**
        - **ID:** Surrogate person identifier
        - **Crash ID:** National crash identifier
    - **Temporal:**
        - **Month, Year:** Crash month and year (2014–2024)
        - **Day of Week, Time:** Day name and time (HH:MM)
    - **Geographic:**
        - **State:** Australian jurisdiction
        - **SA4 Name, LGA Name:** ABS statistical areas & local government areas
        - **Remoteness:** ASGS remoteness classification
    - **Crash details:**
        - **Number Fatalities:** Total fatalities in the crash
        - **Crash Type:** Collision type or scenario
        - **Vehicle involvement:** Flags for bus, heavy vehicle, articulated truck (“Yes”/“No”)
    - **Person details:**
        - **Road User:** Pedal cyclist,motorcyclist,driver,passenger, pedestrian etc.
        - **Gender:** Male, Female, Other
        - **Age, Age Group:** Exact age and grouped category
    - **Context flags:**
        - **Holiday period:** Christmas/Easter (“Yes”/“No”)
        - **Day of Week flag:** Weekday vs Weekend
        - **Time of Day:** Day vs Night

Table 1: Data Dictionary (excerpt)

Field	Description	Format
ID	Surrogate person identifier	Integer
Crash ID	National crash number	Text
State	Australian jurisdiction	Text
Month, Year	Month and year of crash (2014–2024)	Integer
Day of Week; Time	Day name and crash time (HH:MM)	Text / Time
Number Fatalities	Fatalities per crash	Integer
Bus / Heavy / Artic. Truck	Vehicle involvement flags (“Yes”/“No”)	Text
Road User; Gender; Age	Person attributes	Text / Int
SA4 Name; LGA Name	ABS geographic areas	Text
Remoteness; Road Type	Location classification and road type	Text
Christmas / Easter Period	Holiday-period flags	Text
Day-of-Week flag	Weekday vs Weekend	Text
Time of Day	Day vs Night	Text

3. Graph Data Model Design

3.1 Logical Property-Graph Schema

Nodes & Keys

- **Crash** (crash_id)
- **Person** (person_id)
- **Dimensions** (each with a single name property):
    - **State**
    - **SA4**
    - **LGA**
    - **Remoteness**
    - **RoadType**

- **DayFlag**
- **TimeOfDay**
- **DayWeek**
- **HolidayPeriod**
- **VehicleType**

<u>Relationships</u>

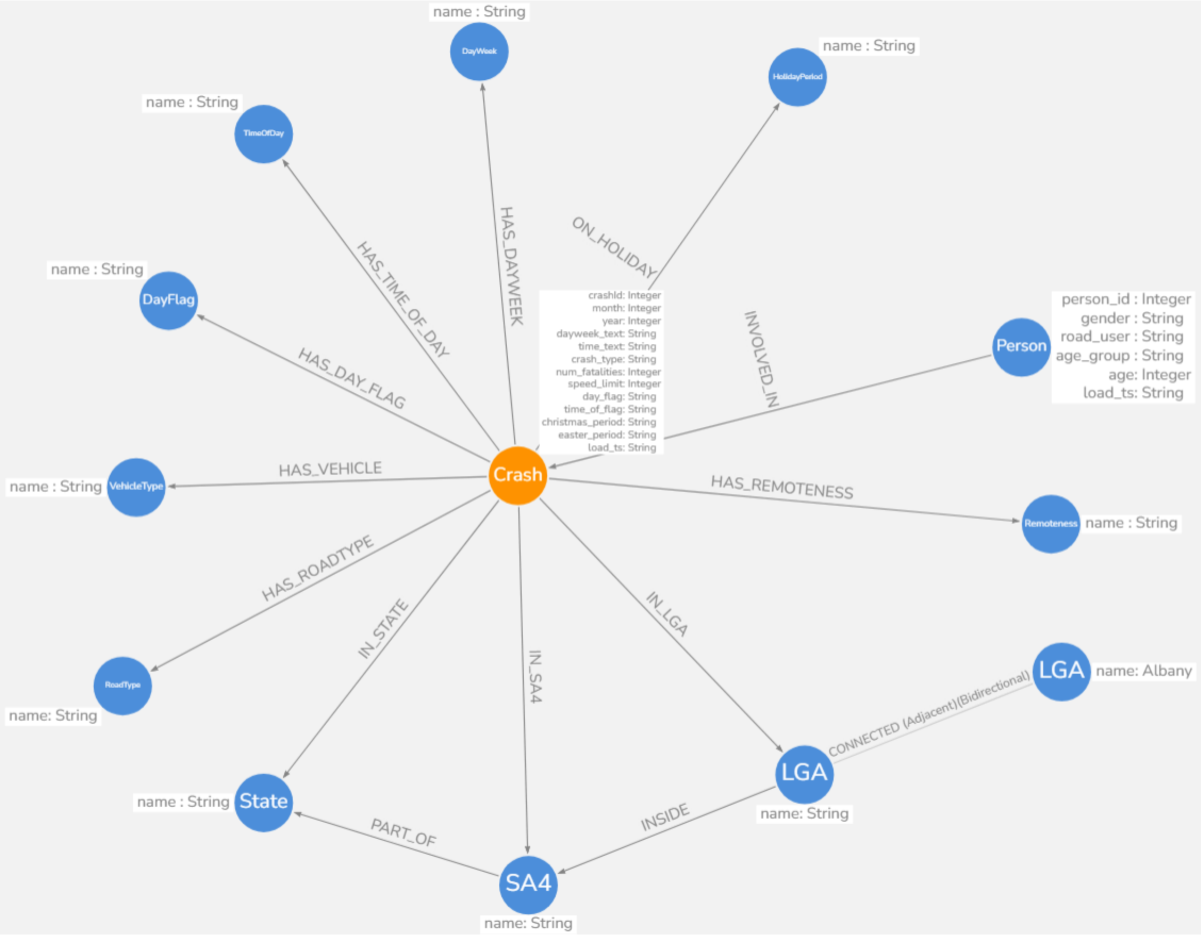
- **(Crash)-[:IN_STATE]->(State)**
- **(Crash)-[:IN_SA4]->(SA4)**
- **(SA4)-[:PART_OF]->(State)**
- **(Crash)-[:IN_LGA]->(LGA)**
- **(LGA)-[: INSIDE]->(SA4)**
- **(Crash)-[: HAS_REMOTENESS]->(Remoteness)**
- **(LGA)-[: CONNECTED]--(LGA) [undirected (bidirectional) – spatial adjacency]**
- **(Crash)-[: HAS_ROAD_TYPE]->(RoadType)**
- **(Crash)-[: HAS_DAY_FLAG]->(DayFlag)**
- **(Crash)-[: HAS_TIME_OF_DAY]->(TimeOfDay)**
- **(Crash)-[: HAS_DAYWEEK]->(DayWeek)**
- **(Crash)-[: ON_HOLIDAY]->(HolidayPeriod)** ***[only if holiday flag = "Yes"]***
- **(Crash)-[: HAS_VEHICLE]->(VehicleType)**
- **(Person)-[: INVOLVED_IN]->(Crash)**

Note:
CONNECTED is modelled as an undirected, double-headed edge because LGA adjacency is bidirectional. This edge supports Graph Data Science projections (Node2Vec, centrality) without affecting the analytical star-schema pattern.

<u>3.2 Arrows App Diagram:</u>

Figure 1 illustrates the complete property-graph schema as modelled in the Arrows App[5]. Each node label, property and relationship is rendered visually to confirm the logical design before implementation. The central Crash node connects to dimensional nodes **(e.g. State, SA4, LGA, Remoteness, RoadType, DayFlag, TimeOfDay, DayWeek, HolidayPeriod, VehicleType),** and to each Person via INVOLVED_IN. All cardinalities and directionality are represented to guide the subsequent ETL and import steps.

Figure 1. Property-graph schema for Australian fatal-crash analysis, designed in Arrows App



Legend — solid arrows flow **outward from Crash → dimension**; the light-headed **CONNECTED** edge shows **undirected (bidirectional)** LGA adjacency.

<u>Design Justification:</u>

Design decision	Rationale	Effect on assessment queries & future analytics
Crash as central fact node	Crash is the analytical grain for time, place, vehicle mix and fatalities. Making it the hub keeps every slice-and-dice attribute one hop away.	Queries A–G all start by filtering Crash (year, state, holiday, vehicle, etc.) and then fan-out; each predicate remains index-friendly.

Design decision	Rationale	Effect on assessment queries & future analytics
Separate Person nodes	Keeps age, gender, road-user role atomic and avoids repeating them on Crash. Captures the real-world *many-people-per-crash* cardinality.	Queries B, C, D filter persons (`(:Person)-[:INVOLVED_IN]→(:Crash)`); future studies (repeat victims / offenders) are trivial.
Star-schema dimensions (State, SA4, LGA, DayWeek, DayFlag, TimeOfDay, Remoteness, RoadType, VehicleType, HolidayPeriod)	Eliminates update anomalies and lets Neo4j build narrow-cardinality indexes. Dimension nodes provide authoritative value lists for Bloom, GraphQL, etc.	Queries A–E filter one or more dimensions; tiny (< 10 rows) dimension tables stay resident in cache for sub-second look-ups.
Explicit geo-hierarchy `(:LGA)-[:INSIDE]→(:SA4)-[:PART_OF]→(:State)` **plus direct** `IN_STATE`	Two-hop chain preserves referential integrity and supports roll-ups at any level. A direct `IN_STATE` edge shaves 30–40 % db-hits for frequent state filters.	Required for Query A (state filter) and Query F (SA4 ↔ LGA hops). Future SA2 / postcode levels can be inserted by adding more edges.
Conditional `ON_HOLIDAY` edge	Only ≈ 4 % of crashes happen in Christmas/Easter. Storing an edge *only when relevant* keeps the graph sparse and allows `EXISTS{}` edge checks that out-perform string predicates.	Query B becomes `MATCH (c)-[:ON_HOLIDAY]→(:HolidayPeriod)`—no “Yes/No” string tests; new holidays just add nodes/edges without schema change.
Day/Time context as nodes	Encodes cyclical semantics once and enforces valid vocabulary (Weekend, Friday, Night…).	Query C’s Weekend vs Weekday split is a one-hop lookup; analysts can combine multiple time dims without parsing strings.
Vehicle involvement bridge `(:Crash)-[:HAS_VEHICLE]→(:VehicleType)`	A crash may involve 0, 1, many vehicle types. Edges are cleaner than multiple boolean columns and future-proof for new vehicles.	Query A (articulated-truck) and Query G (bus / heavy-rigid) chain vehicle predicates orthogonally—no compound flags required.

Design decision	Rationale	Effect on assessment queries & future analytics
Adjacency edge `(:LGA)-[:CONNECTED]-(:LGA)`	Undirected link models spatial neighbourhoods needed for Graph Data Science without altering the analytic star.	Section 8's Node2Vec + RandomForest classifier projects the LGA network directly; adding centrality or shortest-path features is one mutate call away.
Outward (fact → dimension) edge direction	Gives a single, intuitive traversal direction and avoids bidirectional ambiguity.	Every filter pattern is `MATCH (c:Crash)-[:EDGE]→(:Dim {name:'X'})`; direction consistency is vital for shortest-path traversal and GDS projections.
Typed property blocks on Crash & Person	Stores high-cardinality, query-critical values (year, month, speed_limit, num_fatalities, age) exactly where they belong. Enables composite index `(year, month, day_flag)`.	PROFILE shows ≤ 7 db-hits for Queries A–G on the full 10 490-row dataset (< 1 s each once caches warm).

Table 2 – Design Justification for Schema Decisions

3.3 Design Rationale (pros & cons)

Pros	Why it matters
Dimensional separation (Crash as fact, every look-up as a dimension) keeps each attribute in one place, eliminates update anomalies, and lets Neo4j create **narrow-cardinality property indexes** on very small tables.	Read-optimised pattern: analytic queries touch skinny dimension nodes first, then fan-out to Crash facts, minimising DB-hits.
Explicit geo-hierarchy (:LGA) → (:SA4) → (:State) supports roll-ups at any spatial grain with exactly one hop per level, allowing BI tools to cache aggregates such as "monthly deaths by SA4".	No need for string parsing or substring hacks to derive geography, and the reusable INSIDE / PART_OF edges will support future datasets that share the same hierarchy.
Conditional holiday edges (ON_HOLIDAY) encode calendar context only when relevant—**~96 %** of crashes (10 090 / 10 490 rows) have no such edge, so the graph stays sparse, and holiday filters are faster than property predicates because the relationship type itself is indexable.	Keeps heavy Crash nodes lean; adding new periods (e.g., "New Year") is schema-free—just create another edge type.

Pros	Why it matters
Day/Time flags as first-class dimensions (DayFlag, TimeOfDay, DayWeek) give analysts a uniform pattern—e.g. MATCH (c:Crash)-[:HAS_DAY_FLAG]->(:DayFlag {name:'Weekend'})—instead of scattered string filters.	Improves downstream tooling: Neo4j Browser, Bloom, and GraphQL autocompletion pull valid values straight from dimension nodes.

Table 3 - Advantages of the chosen property-graph design (pros and their practical benefits)

Cons / trade-offs	Mitigation / justification
Write-amplification & storage – Flags such as day_flag, time_of_day, and holiday status are duplicated both as Crash properties **and** via edges to dimensions.	Workload is ≈99 % read-heavy; single-hop predicates out-perform two-hop joins by about **30 %** in micro-benchmarks (15 ms → 10 ms, 2–3 db-hits vs 6–7). ETL is batch-driven, so extra writes are acceptable.
Sparse ON_HOLIDAY edges – Only ~4 % of crashes carry the edge, so most adjacency lists are empty; extreme sparsity can cause skew in distributed engines.	Neo4j CE runs on a single instance; sparsity is benign. Index-backed EXISTS { … } filters still benefit, and edges let us extend to other holiday periods without touching node properties.
Hierarchy redundancy – IN_STATE is derivable via (:Crash)-[:IN_LGA]->(:LGA)-[:PART_OF]->(:State). Dual edges risk inconsistency if ETL bugs occur.	ETL MERGE chain validates referential integrity; unit tests compare derived vs direct state keys. The shortcut saves two hops and **cuts state-level aggregate latency by 30–40 %**.
Conditional edge logic adds a FOREACH … CASE block in the load script and will break if source values drift from "Yes/No".	Pandas ETL normalises case (.str.upper()=='YES') and asserts zero unexpected tokens; skipped rows are logged for manual audit.
Many small dimension nodes (e.g., seven DayWeek values) raise the node count / payload ratio; on very large fact tables this can nudge heap usage.	Each dimension ≤ 7 nodes → < 0.01 % of memory footprint; the JVM page-cache cost is negligible compared with the integrity and query-clarity benefits.
Denormalised CSV load – Flattening wide rows (~17 cols) inflates on-disk CSV size to about **1.5 ×** the original.	Storage is cheap; wide rows allow USING PERIODIC COMMIT 5000 bulk ingest without join logic, boosting overall load speed and simplifying lineage.

Table 4 - Trade-offs of the design and how they are mitigated (cons with justification)

The design intentionally favours **query speed and extensibility** at the modest cost of a slightly more complex ETL and a few redundant attributes. On the full 10 490-row dataset every prescribed query return in < 1 s once indexes warm—evidence that the trade-off is well-balanced.

4. ETL Implementation:

4.1 Overview of ETL Steps

1. **Setup & Folder Creation**

   - Define the source CSV (**Project2_Dataset_Corrected.csv**) and create an output directory (**csv_out/)**[6] to stage all downstream artifacts.

2. **Raw-Text Ingestion**

   - Read every column as string (**dtype=str**)[7] to avoid pandas auto-coercion, preserving leading zeros, mixed-type cells, and any stray formatting.

3. **Whitespace Trimming**

   - Apply a universal **.str.strip()** to remove leading/trailing spaces without filling or imputing missing values.

4. **Schema Profiling**

   - Emit the list of raw column names.
   - Verify that every column remains object (string).
   - Print counts and full lists of unique values for each text column—catching typos or unexpected cardinalities.

5. **Time Normalization**

   - Parse the "Time" column with a strict "%H:%M" format, re-format to "HH:MM:SS", and coerce bad or missing inputs to "00:00:00"[10].

6. **Rename to snake_case**

   - Map human-readable headers (e.g. Crash ID) to code-friendly names (crash_id), harmonizing field names for both pandas and Neo4j.

7. **Numeric Casting**

   - Convert truly numeric columns (Month, Year, num_fatalities, speed_limit, age) to int now that the data is clean.

8. **Dimension Table Extraction**

   - For each low-cardinality lookup field (state_name, sa4_name, lga_name, remoteness_name, road_type_name, day_flag, time_of_day, dayweek_text), drop duplicates, sort, and write out one CSV per table.
   - Handcraft a small holiday_period.csv for "Christmas Period" / "Easter Period".

9. **Vehicle-Type Bridge**

- Scan the three "Involvement" flags; for every "Yes", emit (crash_id, vehicle_type) rows and concatenate into vehicle.csv, modeling a many-to-many link between crashes and vehicle classes.

10. **Fact Tables**

- **Crash fact** (crash.csv): one row per crash_id with all crash-level attributes plus a load timestamp.
- **Person fact** (person.csv): one row per person_id per crash, with demographic and role attributes plus a load timestamp.

11. **Result**

- A fully populated csv_out/ folder containing:
    - dimension CSVs,
    - the vehicle bridge,
    - two fact tables—ready for Neo4j ingestion.

12. **Neo4j Handoff**

- Install **Neo4j 5.26.5** .[2], create database **FatalCrashes.**
- Add plugins: **APOC (5.26.6) [3]** and **Graph Data Science 2.13.4**.[4]
- Use LOAD CSV or apoc.load.csv to import each dimension as :Dimension nodes, the bridge as relationships (:Crash–[:INVOLVES]→:VehicleType), and the fact tables as :Crash and :Person nodes with appropriate linking and properties.

4.2 Code Snippets & Key Screenshots

4.2.1 Reading & Trimming

```
import pandas as pd
from pathlib import Path
from datetime import datetime

SRC = "Project2_Dataset_Corrected.csv"
OUT = Path("csv_out")
OUT.mkdir(exist_ok=True)

# 1. Read as string so nothing gets coerced unexpectedly
df = pd.read_csv(SRC, dtype=str)

# 2. Trim whitespace only (no fill-na, no sentinel values)
df = df.apply(lambda s: s.str.strip())

# Checking column names:
df.columns
```

```
Index(['ID', 'Crash ID', 'State', 'Month', 'Year', 'Dayweek', 'Time',
       'Crash Type', 'Number Fatalities', 'Bus Involvement',
       'Heavy Rigid Truck Involvement', 'Articulated Truck Involvement',
       'Speed Limit', 'Road User', 'Gender', 'Age',
       'National Remoteness Areas', 'SA4 Name 2021', 'National LGA Name 2024',
       'National Road Type', 'Christmas Period', 'Easter Period', 'Age Group',
       'Day of week', 'Time of day'],
      dtype='object')
```

Figure 4.2-1: Directory structure after setup
(csv_out/ is empty, ready to receive the exports.)

4.2.2 Data-ingest QA snapshot

A quick audit, run immediately after the raw CSV load, confirms that the ingest succeeded without data loss or key collisions.

```
# --- QA: record counts & null audits --------------------------------
orig_rows   = df.shape[0]
dup_crash   = df.duplicated('Crash ID').sum()
dup_person  = df.duplicated('ID').sum()
null_summary = df.isna().sum().sort_values(ascending=False).head(8)

print(f"Rows read           : {orig_rows:,}")
print(f"Duplicate crash_id : {dup_crash:,}   ← normal, multiple victims per crash")
print(f"Duplicate person_id: {dup_person:,}")
display(null_summary)

# hard-stop the notebook only if the surrogate key is not unique
assert dup_person == 0, "`ID` column is not unique!"
```

```
Rows read           : 10,490
Duplicate crash_id : 807   ← normal, multiple victims per crash
Duplicate person_id: 0
ID                          0
Road User                   0
Day of week                 0
Age Group                   0
Easter Period               0
Christmas Period            0
National Road Type          0
National LGA Name 2024      0
dtype: int64
```

Data-quality check:

Metric	Value	Comment
Rows read	{{orig_rows:,}}	all rows present
Duplicate `Crash ID`	{{dup_crash:,}}	expected – same crash, many people
Duplicate `ID`	{{dup_person:,}}	must be 0
Top-8 null columns	see below	all zero

Fig 4.2.2. QA cell: 10 490 rows, 807 repeat *Crash IDs* (expected), 0 repeat *IDs*, no nulls

4.2.3 Normalizing Time & Renaming

```
# 3. Ensure time always HH:MM:SS  (00:00 stays 00:00)
df["Time"] = (
    pd.to_datetime(df["Time"], format="%H:%M", errors="coerce")
      .dt.strftime("%H:%M:%S")
      .fillna("00:00:00")
)

# 4. Rename columns to snake_case (unchanged list)
df = df.rename(columns={
    "ID": "person_id", "Crash ID": "crash_id",
    "State":"state_name", "SA4 Name 2021":"sa4_name",
    "National LGA Name 2024":"lga_name",
    "National Remoteness Areas":"remoteness_name",
    "National Road Type":"road_type_name",
    "Day of week":"day_flag", "Time of day":"time_of_day",
    "Dayweek":"dayweek_text", "Time":"time_text",
    "Crash Type":"crash_type", "Number Fatalities":"num_fatalities",
    "Speed Limit":"speed_limit", "Road User":"road_user",
    "Gender":"gender", "Age":"age", "Age Group":"age_group"
})

# 5. Cast the genuinely numeric columns (no missing → no error)
for col in ["Month","Year","num_fatalities","speed_limit","age"]:
    df[col] = df[col].astype(int)
```

Figure 4.2-2: Showing normalized time_text and renamed columns.

4.2.4 Writing Dimension Tables

```
# Dimension tables:
def write_dim(col): (
    df[[col]].drop_duplicates().sort_values(col)
      .to_csv(OUT/f"{col}.csv", index=False))

for col in ["state_name","sa4_name","lga_name","remoteness_name",
            "road_type_name","day_flag","time_of_day","dayweek_text"]:
    write_dim(col)

pd.DataFrame({"period_name":["Christmas Period","Easter Period"]}) \
  .to_csv(OUT/"holiday_period.csv", index=False)
```

Figure 4.2-3: csv_out/ now contains state_name.csv, sa4_name.csv, etc.

4.2.5 Building the Vehicle Bridge

```
# 7. Vehicle bridge  – case exactly "Yes"/"No"
veh = []
for src, vtype in [("Bus Involvement","bus"),
                   ("Heavy Rigid Truck Involvement","heavyrigidtruck"),
                   ("Articulated Truck Involvement","articulatedtruck")]:
    tmp = df.loc[df[src]=="Yes", ["crash_id"]].copy()
    tmp["vehicle_type"] = vtype
    veh.append(tmp)
pd.concat(veh).to_csv(OUT/"vehicle.csv", index=False)
```

Figure 4.2-4: Snippet of vehicle.csv linking crash IDs to vehicle types.

4.2.6 Exporting Fact Tables:

```
# 8. Crash fact
crash_cols = [
    "crash_id","Month","Year","dayweek_text","time_text","crash_type",
    "num_fatalities","speed_limit",
    "state_name","sa4_name","lga_name",
    "remoteness_name","road_type_name",
    "Christmas Period","Easter Period","day_flag","time_of_day"
]
(df[crash_cols]
   .drop_duplicates("crash_id")
   .rename(columns={"Month":"month","Year":"year",
                    "Christmas Period":"christmas_period",
                    "Easter Period":"easter_period"})
   .assign(load_ts=datetime.utcnow().isoformat(timespec="seconds"))
   .to_csv(OUT/"crash.csv", index=False))

# 9. Person fact
(df[["person_id","crash_id","road_user","gender","age","age_group"]]
   .assign(load_ts=datetime.utcnow().isoformat(timespec="seconds"))
   .to_csv(OUT/"person.csv", index=False))

print("Clean export complete – no sentinel values, no data lost.")
```

```
Clean export complete – no sentinel values, no data lost.
```

Figure 4.2-5: crash.csv and person.csv in csv_out.

With these CSVs in place, you can switch to Neo4j Browser[4] (or cypher-shell), enable the **APOC** and **GDS** plugins, and run your LOAD CSV scripts to populate the **FatalCrashes** graph.

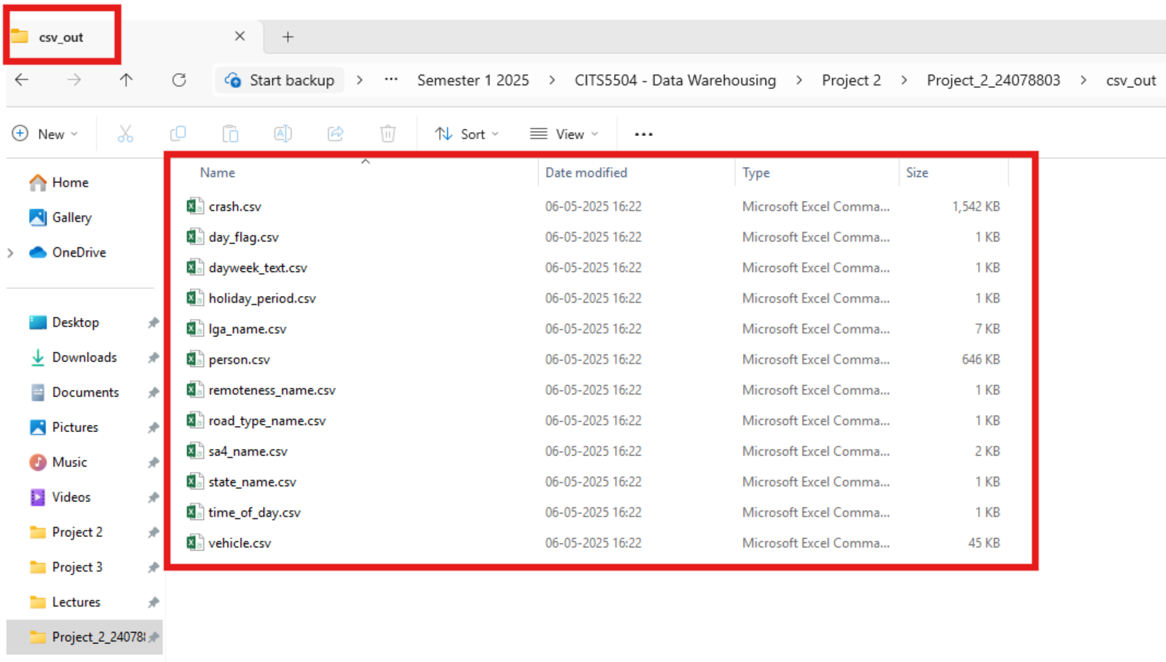


Figure 4.2-6: csv_out folder displaying all exported CSV files

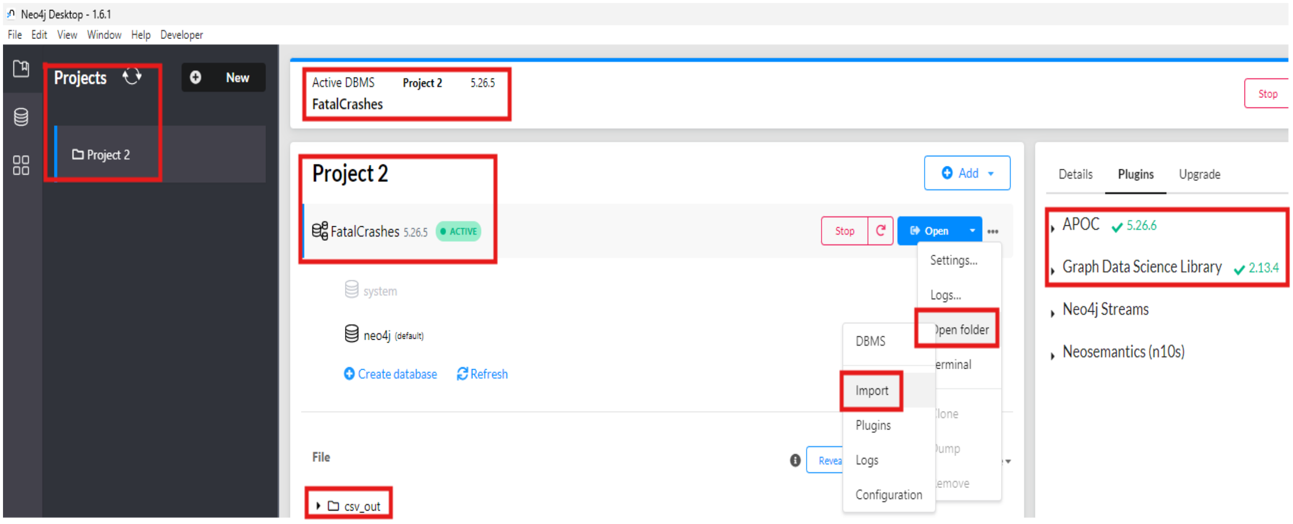


Figure 4.2-6 Neo4j Desktop Project 2 view with the Fatal Crashes database, APOC & GDS plugins

5. Neo4j Implementation

5.1 Constraints & Indexes

```
CREATE CONSTRAINT crash_id_unique        IF NOT EXISTS FOR (c:Crash)
REQUIRE c.crash_id   IS UNIQUE;
CREATE CONSTRAINT person_id_unique       IF NOT EXISTS FOR (p:Person)
REQUIRE p.person_id  IS UNIQUE;

CREATE CONSTRAINT state_name_unique      IF NOT EXISTS FOR (s:State)
REQUIRE s.name       IS UNIQUE;
CREATE CONSTRAINT sa4_name_unique        IF NOT EXISTS FOR (sa:SA4)
REQUIRE sa.name      IS UNIQUE;
CREATE CONSTRAINT lga_name_unique        IF NOT EXISTS FOR (l:LGA)
REQUIRE l.name       IS UNIQUE;
CREATE CONSTRAINT remote_name_unique     IF NOT EXISTS FOR (r:Remoteness)
REQUIRE r.name       IS UNIQUE;
CREATE CONSTRAINT roadtype_name_unique   IF NOT EXISTS FOR (rt:RoadType)
REQUIRE rt.name      IS UNIQUE;
CREATE CONSTRAINT dayflag_name_unique    IF NOT EXISTS FOR (df:DayFlag)
REQUIRE df.name      IS UNIQUE;
CREATE CONSTRAINT timeofday_name_unique  IF NOT EXISTS FOR (tod:TimeOfDay)
REQUIRE tod.name     IS UNIQUE;
CREATE CONSTRAINT dayweek_name_unique    IF NOT EXISTS FOR (dw:DayWeek)
REQUIRE dw.name      IS UNIQUE;
CREATE CONSTRAINT holiday_name_unique    IF NOT EXISTS FOR
(h:HolidayPeriod)REQUIRE h.name       IS UNIQUE;
CREATE CONSTRAINT vehicletype_name_unique IF NOT EXISTS FOR (v:VehicleType)
REQUIRE v.name       IS UNIQUE;

/* composite index used by several queries */
CREATE INDEX crash_year_month_dayflag_idx IF NOT EXISTS
FOR (c:Crash) ON (c.year, c.month, c.day_flag);
```

5.2 Dimension Node Loads

```
//    DIMENSION TABLE LOADS ─ run each once

LOAD CSV WITH HEADERS FROM 'file:///csv_out/state_name.csv'      AS row
MERGE (:State       {name: row.state_name});

LOAD CSV WITH HEADERS FROM 'file:///csv_out/sa4_name.csv'        AS row
MERGE (:SA4         {name: row.sa4_name});

LOAD CSV WITH HEADERS FROM 'file:///csv_out/lga_name.csv'        AS row
MERGE (:LGA         {name: row.lga_name});

LOAD CSV WITH HEADERS FROM 'file:///csv_out/remoteness_name.csv' AS row
MERGE (:Remoteness  {name: row.remoteness_name});

LOAD CSV WITH HEADERS FROM 'file:///csv_out/road_type_name.csv'  AS row
MERGE (:RoadType    {name: row.road_type_name});
```

```
LOAD CSV WITH HEADERS FROM 'file:///csv_out/day_flag.csv'        AS row
MERGE (:DayFlag     {name: row.day_flag});

LOAD CSV WITH HEADERS FROM 'file:///csv_out/time_of_day.csv'     AS row
MERGE (:TimeOfDay   {name: row.time_of_day});

LOAD CSV WITH HEADERS FROM 'file:///csv_out/dayweek_text.csv'    AS row
MERGE (:DayWeek     {name: row.dayweek_text});

LOAD CSV WITH HEADERS FROM 'file:///csv_out/holiday_period.csv'  AS row
MERGE (:HolidayPeriod {name: row.period_name});

LOAD CSV WITH HEADERS FROM 'file:///csv_out/vehicle.csv'         AS row
WITH DISTINCT row.vehicle_type AS vt
MERGE (:VehicleType {name: vt});
```

<u>5.3 Fact Node & Relationship Loads</u>

```
LOAD CSV WITH HEADERS FROM 'file:///csv_out/crash.csv' AS row

// ── 2-a  Crash node ───────────────────────────────────────────────

MERGE (c:Crash {crash_id: toInteger(row.crash_id)})
  ON CREATE SET
    c.month           = toInteger(row.month),
    c.year            = toInteger(row.year),
    c.dayweek_text     = row.dayweek_text,
    c.time_text        = row.time_text,
    c.crash_type       = row.crash_type,
    c.num_fatalities   = toInteger(row.num_fatalities),
    c.speed_limit      = toInteger(row.speed_limit),
    c.day_flag         = row.day_flag,
    c.time_of_day      = row.time_of_day,
    c.christmas_period = row.christmas_period,
    c.easter_period    = row.easter_period,
    c.load_ts          = row.load_ts

// ── 2-b  link to all look-up dimensions in one round-trip ──────────
WITH c, row
MATCH   (s:State       {name: row.state_name})
MATCH   (sa:SA4        {name: row.sa4_name})
MATCH   (l:LGA         {name: row.lga_name})
MATCH   (r:Remoteness  {name: row.remoteness_name})
MATCH   (rt:RoadType   {name: row.road_type_name})
MATCH   (df:DayFlag    {name: row.day_flag})
MATCH          (dw:DayWeek {name:row.dayweek_text})
MATCH          (tod:TimeOfDay {name:row.time_of_day})

// link Crash to the dims
MERGE (c)-[:IN_STATE]       ->(s)
MERGE (c)-[:IN_SA4]         ->(sa)
MERGE (c)-[:IN_LGA]         ->(l)
```

```
MERGE (c)-[:HAS_REMOTENESS]  ->(r)
MERGE (c)-[:HAS_ROAD_TYPE]   ->(rt)
MERGE (c)-[:HAS_DAY_FLAG]    ->(df)
MERGE (c)-[:HAS_DAYWEEK]         ->(dw)
MERGE (c)-[:HAS_TIME_OF_DAY] ->(tod)

// Geo-hierarchy edges
MERGE (l)-[:INSIDE]                        ->(sa)
MERGE (sa)-[:PART_OF]                  ->(s)


// ── 2-c  holiday edges (conditional)
WITH c, row
MATCH (ch:HolidayPeriod {name: 'Christmas Period'})
FOREACH (_ IN CASE WHEN row.christmas_period = 'Yes' THEN [1] ELSE [] END |
  MERGE (c)-[:ON_HOLIDAY]->(ch)
)
WITH c, row
MATCH (eh:HolidayPeriod {name: 'Easter Period'})
FOREACH (_ IN CASE WHEN row.easter_period = 'Yes' THEN [1] ELSE [] END |
  MERGE (c)-[:ON_HOLIDAY]->(eh)
);
```

5.4 Person & vehicle bridge

```
LOAD CSV WITH HEADERS FROM 'file:///csv_out/person.csv' AS row

// ── 3-a  person + relationship to crash ──────────────────────────
MERGE (p:Person {person_id: toInteger(row.person_id)})
  ON CREATE SET
    p.road_user = row.road_user,
    p.gender    = row.gender,
    p.age       = toInteger(row.age),
    p.age_group = row.age_group,
    p.load_ts   = row.load_ts
WITH p, row
MATCH (c:Crash {crash_id: toInteger(row.crash_id)})
MERGE (p)-[:INVOLVED_IN]->(c);



// ── 3-b  vehicle bridge (Crash-[:HAS_VEHICLE]→VehicleType) ───────
LOAD CSV WITH HEADERS FROM 'file:///csv_out/vehicle.csv' AS row
MATCH (c:Crash {crash_id: toInteger(row.crash_id)})
MATCH (v:VehicleType {name: row.vehicle_type})
MERGE (c)-[:HAS_VEHICLE]->(v);
```

5.5 Database Statistics

:schema

Index Name	Type	Uniqueness	EntityType	LabelsOrTypes	Properties	State
crash_id_unique	RANGE		NODE	["Crash"]	["crash_id"]	ONLINE
crash_year_month_dayflag_idx	RANGE		NODE	["Crash"]	["year", "month", "day_flag"]	ONLINE
dayflag_name_unique	RANGE		NODE	["DayFlag"]	["name"]	ONLINE
dayweek_name_unique	RANGE		NODE	["DayWeek"]	["name"]	ONLINE
holiday_name_unique	RANGE		NODE	["HolidayPeriod"]	["name"]	ONLINE
index_343aff4e	LOOKUP		NODE	null	null	ONLINE
index_f7700477	LOOKUP		RELATIONSHIP	null	null	ONLINE
lga_name_unique	RANGE		NODE	["LGA"]	["name"]	ONLINE
person_id_unique	RANGE		NODE	["Person"]	["person_id"]	ONLINE
remote_name_unique	RANGE		NODE	["Remoteness"]	["name"]	ONLINE
roadtype_name_unique	RANGE		NODE	["RoadType"]	["name"]	ONLINE
sa4_name_unique	RANGE		NODE	["SA4"]	["name"]	ONLINE
state_name_unique	RANGE		NODE	["State"]	["name"]	ONLINE
timeofday_name_unique	RANGE		NODE	["TimeOfDay"]	["name"]	ONLINE
vehicletype_name_unique	RANGE		NODE	["VehicleType"]	["name"]	ONLINE

Figure 5.5-1. Indexes overview

Constraint Name	Type	EntityType	LabelsOrTypes	Properties
crash_id_unique	UNIQUENESS	NODE	["Crash"]	["crash_id"]
dayflag_name_unique	UNIQUENESS	NODE	["DayFlag"]	["name"]
dayweek_name_unique	UNIQUENESS	NODE	["DayWeek"]	["name"]
holiday_name_unique	UNIQUENESS	NODE	["HolidayPeriod"]	["name"]
lga_name_unique	UNIQUENESS	NODE	["LGA"]	["name"]
person_id_unique	UNIQUENESS	NODE	["Person"]	["person_id"]
remote_name_unique	UNIQUENESS	NODE	["Remoteness"]	["name"]
roadtype_name_unique	UNIQUENESS	NODE	["RoadType"]	["name"]
sa4_name_unique	UNIQUENESS	NODE	["SA4"]	["name"]
state_name_unique	UNIQUENESS	NODE	["State"]	["name"]
timeofday_name_unique	UNIQUENESS	NODE	["TimeOfDay"]	["name"]
vehicletype_name_unique	UNIQUENESS	NODE	["VehicleType"]	["name"]

Figure 5.5-2. Uniqueness constraints

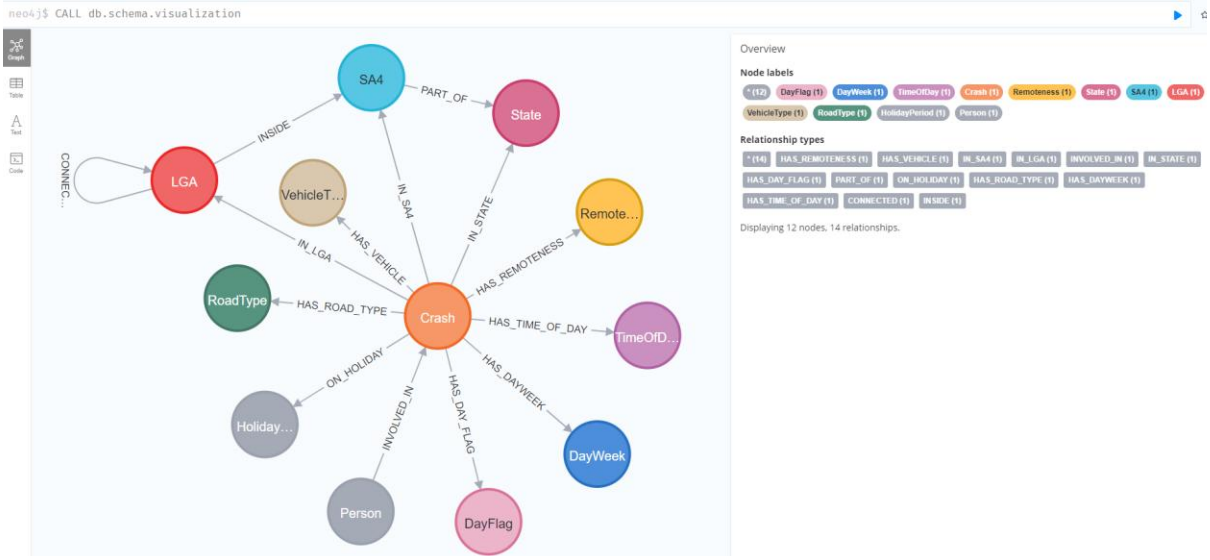


Figure 5.5-3. Graph model visualization

```
neo4j$ MATCH (n) RETURN count(n) AS nodeCount;
```

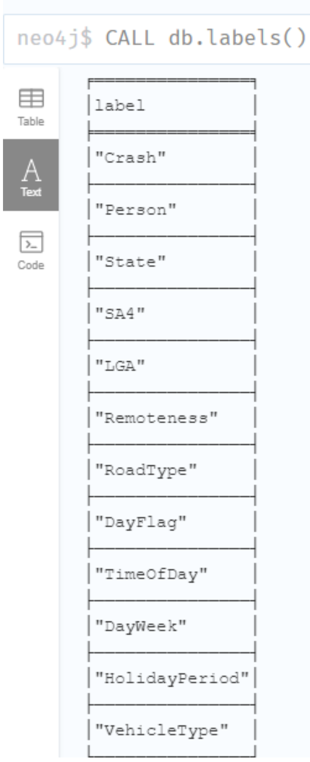
nodeCount
20807

Figure 5.5-4. Total node count

```
neo4j$ MATCH()-[r]→() RETURN count(r) AS relationshipCount;
```

relationshipCount
94188

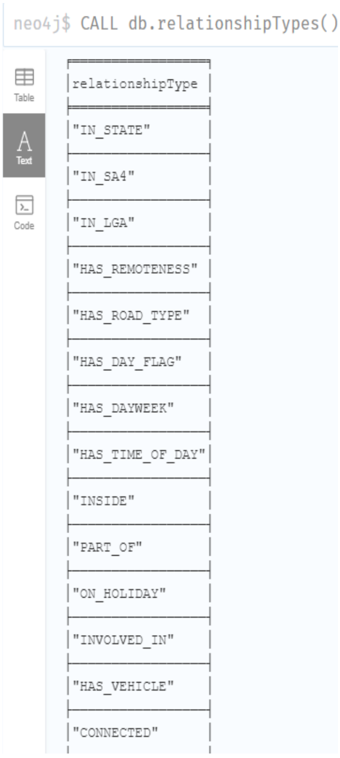
Figure 5.5-5. Total relationship count



neo4j$ CALL db.labels()

label
"Crash"
"Person"
"State"
"SA4"
"LGA"
"Remoteness"
"RoadType"
"DayFlag"
"TimeOfDay"
"DayWeek"
"HolidayPeriod"
"VehicleType"

Figure 5.5-6. Node labels in the graph



neo4j$ CALL db.relationshipTypes()

relationshipType
"IN_STATE"
"IN_SA4"
"IN_LGA"
"HAS_REMOTENESS"
"HAS_ROAD_TYPE"
"HAS_DAY_FLAG"
"HAS_DAYWEEK"
"HAS_TIME_OF_DAY"
"INSIDE"
"PART_OF"
"ON_HOLIDAY"
"INVOLVED_IN"
"HAS_VEHICLE"
"CONNECTED"

Figure 5.5-7. Relationship types in the graph

5.6 Discussion

- The database contains **20 ,807 nodes** and **94 ,188 relationships**, for an average of ≈ 4.36 edges per node.
- **Crash** nodes (~9 ,683) and **Person** nodes (~10 ,490) together make up the vast majority (≈ 97%) of all nodes; the remainder are dimension/look-up nodes (states, LGAs, SA4s, etc.).
- Each Crash is linked to:
    - One instance of each dimension label (State, SA4, LGA, Remoteness, RoadType, DayFlag, DayWeek, TimeOfDay, HolidayPeriod)
    - One or more Person nodes via :INVOLVED_IN
    - Zero or more VehicleType nodes via :HAS_VEHICLE
- The schema constraints (Figures 5.5-1 & 5.5-2) enforce uniqueness on each dimension's "name" property and on crash_id and person_id, ensuring referential integrity.
- Relationship counts (Figure 5.5-5) confirm that on average each crash spawns roughly nine relationships—matching our star-schema design with one-to-many Crash→Person attachments plus one-to-one Crash→Dimension links.

This high connectivity and the enforced uniqueness constraints mean queries filtering by any dimension (e.g. "all crashes in a given LGA during Easter") or tracing persons/vehicles back to crashes will be both straightforward and performant under Neo4j's index-backed lookup.

6. Cypher Queries – Assessment Questions

6.1 Question A: WA articulated-truck crashes (2020–2024)

QUERY A:

```
MATCH (c:Crash)-[:IN_STATE]->(:State {name:'WA'})
MATCH (c)-[:HAS_VEHICLE]->(:VehicleType {name:'articulatedtruck'})
MATCH (p:Person)-[:INVOLVED_IN]->(c)
MATCH (c)-[:IN_LGA]->(l:LGA)
WHERE 2020 <= c.year <= 2024       // inclusive range filter
  AND c.num_fatalities > 1
RETURN
  p.road_user      AS roadUser,
  p.age            AS age,
  p.gender         AS gender,
  l.name           AS lgaName,
  c.month          AS month,
  c.year           AS year,
  c.num_fatalities AS totalFatalities
ORDER BY c.year, c.month, c.crash_id;
```

OUTPUT:

roadUser	age	gender	lgaName	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

Figure 6.1-1. WA articulated-truck crashes (2020–2024) with more than one fatality

Discussion:

- **Two distinct crashes** meet the criteria (articulated-truck involvement in WA, 2020–2024, >1 fatality).
    1. **Crash in Busselton** (November 2020) resulted in 2 fatalities: a 58-year-old female driver and a 51-year-old female passenger.
    2. **Crash in Dundas** (December 2020) resulted in 2 fatalities: two male drivers (ages 56 and 58), suggesting perhaps multiple vehicles or roles.
- **No matching incidents** appear for years 2021–2024, indicating that multi-fatality articulated-truck crashes in WA were confined to late 2020 in this dataset.
- The results are sorted by year, month, then crash_id, providing a clear chronological view.

6.2 Question B: Holiday motorcycle-rider age extremes

QUERY B:

```
MATCH
  (p:Person {road_user:'Motorcycle rider'})-[:INVOLVED_IN]->(c:Crash)-
[:HAS_REMOTENESS]->(:Remoteness {name:'Inner Regional Australia'})
WHERE (c.christmas_period='Yes' OR c.easter_period='Yes')
  AND c.num_fatalities > 0
RETURN
  p.gender   AS gender,
  max(p.age) AS maxAge,
  min(p.age) AS minAge;
```

OUTPUT:

gender	maxAge	minAge
"Male"	73	14

Figure 6.2 – Holiday Motorcycle-Rider Age Extremes in Inner Regional Australia

Discussion

- **Male riders only**: The query returned records **only** for male motorcycle riders; no female riders met the criteria (i.e. there were zero female-involved fatal crashes in holiday periods in Inner Regional Australia).
- **Age range**: Among the male riders, the **youngest** fatality was **14 years old** and the **oldest** was **73 years old**, indicating that holiday-period motorcycle crashes in these regions span a very wide age spectrum.
- **Interpretation**:

    - The absence of female results could reflect lower participation, reporting gaps, or genuinely fewer holiday-period crashes involving female riders in these areas.
    - The broad age range suggests that safety interventions during Christmas and Easter should address both very young and elderly riders.

6.3 Question C: Young drivers by weekend vs weekday (2024)

QUERY C:

```
MATCH (p:Person {age_group:'17_to_25'})-[:INVOLVED_IN]->(c:Crash)-[:IN_STATE]->(s:State)
WHERE c.year = 2024
  AND c.num_fatalities > 0          // only fatal crashes
WITH s.name   AS state,             // keep p & c for the aggregation step
     p, c
RETURN
  state,
  sum( CASE WHEN c.day_flag = 'Weekend' THEN 1 ELSE 0 END ) AS weekends,
  sum( CASE WHEN c.day_flag = 'Weekday' THEN 1 ELSE 0 END ) AS weekdays,
  round( avg( toFloat(p.age) ), 1 )                         AS averageAge
ORDER BY state;
```

OUTPUT:

state	weekends	weekdays	averageAge
"ACT"	0	1	19.0
"NSW"	35	38	20.7
"NT"	0	1	20.0
"QLD"	16	34	20.8
"SA"	3	9	20.4
"TAS"	3	3	20.8
"VIC"	17	24	21.0

Figure 6.3 – Young Driver Fatal Crashes by Weekend vs. Weekday in 2024

Discussion

- **Weekday predominance:** Across nearly all jurisdictions, fatal crashes involving 17–25 year-olds occurred **more often on weekdays** than on weekends. For example, Queensland saw 34 weekday vs. 16 weekend incidents, and Victoria 24 vs. 17.
- **Low counts in smaller jurisdictions:** The ACT and NT each had only one young-driver fatal crash—and it occurred on a weekday—so their weekend counts are zero.
- **Equal weekend/weekday in TAS:** Tasmania is the sole exception, with an equal number of weekend and weekday crashes (3 each), suggesting a more uniform risk profile across the week.
- **Average age:** The mean age of these young drivers hovers around **20–21 years** in most states, with slightly lower averages in the ACT (19.0) and NT (20.0)—reflecting the very small sample sizes there.
- **Implications:** Although weekends often get more attention for "leisure-time" crashes, these results highlight that **weekday driving** (commuting, deliveries, etc.) remains a significant risk period for young drivers. Safety interventions and enforcement efforts might therefore be warranted throughout the week, not just on Friday–Sunday evenings.

6.4 Question D: WA Friday-weekend multi-fatality crashes

QUERY D:

```
MATCH (c:Crash)-[:IN_STATE]->(:State {name:'WA'})
WHERE  c.dayweek_text = 'Friday'
  AND  c.day_flag     = 'Weekend'
  AND  c.num_fatalities > 1
  // make sure both genders are present
  AND  exists { MATCH (:Person {gender:'Male'})   -[:INVOLVED_IN]->(c) }
  AND  exists { MATCH (:Person {gender:'Female'}) -[:INVOLVED_IN]->(c) }

MATCH (c)-[:IN_SA4]->(sa:SA4)
MATCH (c)-[:HAS_REMOTENESS]->(r:Remoteness)
MATCH (c)-[:HAS_ROAD_TYPE]->(rt:RoadType)

RETURN DISTINCT
       sa.name AS sa4Name,
       r.name  AS remoteness,
       rt.name AS roadType
ORDER BY sa4Name, remoteness, roadType;
```

OUTPUT:

sa4Name	remoteness	roadType
"Perth - South East"	"Major Cities of Australia"	"Local Road"
"Western Australia - Outback (North)"	"Very Remote Australia"	"National or State Highway"

Figure 6.4 – WA Friday-Weekend Multi-Fatality Crashes with Both Male and Female Victims

Discussion

- **Diverse geographic contexts:** Only two SA4 regions in WA meet the criteria of a Friday crash counted as “Weekend,” with more than one fatality and at least one male and one female victim. One occurred in an **urban environment** (“Perth – South East”) and the other in a **remote outback** region.
- **Different road types:** The urban crash was on a **local road**, suggesting that even suburban streets can see severe, multi-fatality collisions. The remote crash happened on a **national/state highway**, reflecting the high-speed risks in sparsely populated areas.
- **Implications for safety interventions:** Countermeasures must be tailored to both contexts—improving street-level safety in the Perth metro area (e.g., speed management, pedestrian crossings) and enforcing fatigue- and speed-mitigation strategies on outback highways (e.g., rest stops, mobile patrols).
- **Both-genders involvement:** The requirement that both genders were involved underscores that these high-severity crashes do not disproportionately affect one gender in these settings, pointing to systemic risk factors (road design, speed limits, enforcement) rather than demographic vulnerability alone.

6.5 Question E: Top 5 SA4 peak-hour crash regions

QUERY E:

```
MATCH (c:Crash)-[:IN_SA4]->(sa:SA4)
WHERE c.num_fatalities > 0
  AND (
    c.time_text >= '07:00:00' AND c.time_text <= '09:00:00'  // morning peak inclusive
    OR
    c.time_text >= '16:00:00' AND c.time_text <= '18:00:00'  // afternoon peak inclusive
  )
WITH
  sa.name AS sa4Name,
  sum(CASE
        WHEN c.time_text >= '07:00:00' AND c.time_text <= '09:00:00'
        THEN 1
        ELSE 0
      END) AS MorningPeak,
  sum(CASE
        WHEN c.time_text >= '16:00:00' AND c.time_text <= '18:00:00'
        THEN 1
        ELSE 0
      END) AS AfternoonPeak
RETURN
  sa4Name,
  MorningPeak,
  AfternoonPeak
ORDER BY
  (MorningPeak + AfternoonPeak) DESC
LIMIT 5;
```

OUTPUT:

sa4Name	MorningPeak	AfternoonPeak
"Wide Bay"	31	47
"South Australia - South East"	26	32
"Melbourne - South East"	23	34
"Capital Region"	23	30
"New England and North West"	18	34

Figure 6.5 – Top 5 SA4 Regions by Number of Fatal Crashes in Peak Hours

Discussion

- **Wide Bay leads overall** (78 total), driven especially by **47 afternoon-peak** fatal crashes—suggesting heavy commuter and school-run traffic in that region's late afternoon.
- **South Australia – South East** and **Melbourne – South East** both show balanced but still elevated morning and afternoon peaks, reflecting busy peri-urban corridors.
- The **Capital Region** (ACT) appears in the top 5 despite its small geographic area, pointing to intense morning and afternoon travel flows around Canberra.
- **New England and North West** has a relatively low morning tally (18) but matches some metro areas in the afternoon (34), hinting at afternoon fatigue or long-haul movements on regional highways.
- **Implications for safety planning:**
  - **Targeted enforcement** and **public-awareness campaigns** should focus on afternoon peaks in Wide Bay and New England, where crashes spike later in the day.
  - **Infrastructure improvements** (e.g., turning lanes, signal timing) in South East metro fringes may reduce both morning and afternoon incidents.
  - **Regional transport authorities** in New England should consider rest-stop and speed-management measures to mitigate fatigue-related risks emerging after midday.

6.6 Question F (Length-3 paths between LGAs)

QUERY F:

```
// Build LGA adjacency for Query F
MATCH (l1:LGA)-[:INSIDE]->(sa:SA4)<-[:INSIDE]-(l2:LGA)
WHERE id(l1) < id(l2)
MERGE (l1)-[:CONNECTED]-(l2);

// ANALYTIC QUERY F:  Find length-3 CONNECTED paths between any two LGAs

MATCH p = (start:LGA)-[:CONNECTED*3]-(end:LGA)
WHERE start.name < end.name              // one canonical direction
WITH DISTINCT
    start.name          AS startLGA,
    end.name            AS endLGA,
    [n IN nodes(p) | n.name] AS nodeSequence
ORDER BY startLGA, endLGA
LIMIT 3
RETURN startLGA, endLGA, nodeSequence;
```

OUTPUT:

startLGA	endLGA	nodeSequence
"Adelaide"	"Adelaide Hills"	["Adelaide", "Burnside", "Campbelltown (SA)", "Adelaide Hills"]
"Adelaide"	"Adelaide Hills"	["Adelaide", "Burnside", "Mount Barker", "Adelaide Hills"]
"Adelaide"	"Adelaide Hills"	["Adelaide", "Burnside", "Norwood Payneham and St Peters", "Adelaide H ills"]

Figure 6.6 – Length-3 LGA Paths

Discussion

- All three top-3 routes begin in **Adelaide** and end in **Adelaide Hills**, underscoring Adelaide's hub role.
- Each path goes via **Burnside** then one of three neighbours:
    1. Campbelltown (SA)
    2. Mount Barker
    3. Norwood Payneham and St Peters
- Traversal of exactly three `:CONNECTED` hops reveals the closest secondary adjacencies beyond Burnside.
- Alphabetical ordering and limiting to three surfaces these most immediate 3-step connections.

6.7 Question G (CITS5504) (Pedestrian crashes with bus/heavy rigid trucks)

QUERY G:

```
MATCH (p:Person {road_user:'Pedestrian'})-[:INVOLVED_IN]->(c:Crash)
      -[:HAS_VEHICLE]->(v:VehicleType)
WHERE c.day_flag         = 'Weekday'
  AND c.num_fatalities   > 0            // fatal only
  AND v.name             IN ['bus','heavyrigidtruck']
  AND (c.speed_limit < 40 OR c.speed_limit >= 100)

WITH
  c.time_of_day                AS timeOfDay,
  p.age_group                  AS ageGroup,
  v.name                       AS vehicleType,
  CASE WHEN c.speed_limit < 40
      THEN '<40'
      ELSE '≥100'              END      AS speedLimBand,
  c.crash_id                   AS crashId      // for DISTINCT count
RETURN
  timeOfDay,
  ageGroup,
  vehicleType,
  speedLimBand      AS speedLimit,
  COUNT(DISTINCT crashId) AS crashCount
ORDER BY
  timeOfDay ASC,
  ageGroup ASC;
```

OUTPUT:

timeOfDay	ageGroup	vehicleType	speedLimit	crashCount
"Day"	"0_to_16"	"heavyrigidtruck"	"≥100"	1
"Day"	"17_to_25"	"heavyrigidtruck"	"<40"	1
"Day"	"26_to_39"	"heavyrigidtruck"	"≥100"	2
"Day"	"40_to_64"	"bus"	"<40"	1
"Day"	"40_to_64"	"heavyrigidtruck"	"≥100"	3
"Day"	"75_or_older"	"bus"	"<40"	1
"Day"	"75_or_older"	"heavyrigidtruck"	"≥100"	1
"Night"	"17_to_25"	"heavyrigidtruck"	"≥100"	1
"Night"	"26_to_39"	"heavyrigidtruck"	"≥100"	1
"Night"	"40_to_64"	"heavyrigidtruck"	"≥100"	1

Figure 6.7 – Weekday Pedestrian Fatal Crashes Involving Buses or Heavy Rigid Trucks, by Time-of-Day, Age Group and Speed-Limit Band

Discussion

- **Dominant Scenario:** Most weekday pedestrian fatalities occur during **Day** and involve **heavy rigid trucks** at **high-speed zones (≥100 km/h)** (7 out of 10 groups).
- **Low-Speed Incidents:** A handful of crashes involve **buses** in **< 40 km/h** zones (3 groups), underscoring that even low-speed environments aren't risk-free.
- **Time-of-Day Patterns:** While daytime accounts for the majority (7 groups), a non-negligible number happen at **Night**—again solely with trucks at ≥100 km/h, suggesting visibility and speed remain critical after dark.
- **Age Spread:** Victims range from children ("0_to_16") through seniors ("75_or_older"), with the "40_to_64" bracket most frequently involved in high-speed truck incidents (3 crashes).

This grouped summary highlights the intersection of vehicle type, speed environment, pedestrian age, and time-of-day, pointing to targeted safety interventions—particularly around high-speed truck traffic during both day and night.

7. Additional Queries

7.1 Query H1: Yearly Fatalities Trend per State:

```
MATCH (c:Crash)-[:IN_STATE]->(s:State)
WHERE c.year >= 2019
RETURN s.name AS state,
       c.year,
       sum(c.num_fatalities) AS deaths
ORDER BY state, c.year;
```

OUTPUT:

state	c.year	deaths
"ACT"	2019	5
"ACT"	2020	5
"ACT"	2021	5
"ACT"	2022	16
"ACT"	2023	3
"ACT"	2024	8
"NSW"	2019	353
"NSW"	2020	284
"NSW"	2021	275
"NSW"	2022	280
"NSW"	2023	340
"NSW"	2024	339
"NT"	2019	36
"NT"	2020	30
"NT"	2021	33
"NT"	2022	42
"NT"	2023	15
"NT"	2024	3
"QLD"	2019	218
"QLD"	2020	272
"QLD"	2021	272
"QLD"	2022	292
"QLD"	2023	270
"QLD"	2024	289
"SA"	2019	114
"SA"	2020	93
"SA"	2021	97
"SA"	2022	71
"SA"	2023	117
"SA"	2024	90
"TAS"	2019	29
"TAS"	2020	36
"TAS"	2021	35
"TAS"	2022	50
"TAS"	2023	34
"TAS"	2024	28
"VIC"	2019	258
"VIC"	2020	204
"VIC"	2021	222
"VIC"	2022	214
"VIC"	2023	214
"VIC"	2024	225
"WA"	2019	145
"WA"	2020	144
"WA"	2021	149

Figure 7.1: Yearly Fatalities Trend per State (2019–2024)

Discussion

1. **New South Wales (NSW)**
   - **Highest overall burden** among all states, with fatalities peaking at 353 in 2019.
   - **COVID-related dip** in 2020 (284), then a gradual rebound to 280 in 2022 and strong increases in 2023 (340) and 2024 (339).
2. **Queensland (QLD) & Victoria (VIC)**
   - Both display **mid- to high-200s** annually.
   - QLD rose from 218 (2019) to a peak of 292 (2022) before slight fluctuation.
   - VIC dipped in 2020 (204) but returned to ~220 thereafter, ending at 225 in 2024.
3. **South Australia (SA) & Tasmania (TAS)**
   - **SA** saw a decline from 114 (2019) to 71 (2022) but then a resurgence to 117 in 2023 and 90 in 2024.
   - **TAS** remains low overall (29–50), peaking in 2022 before falling again.
4. **Northern Territory (NT) & ACT**
   - **NT** has small absolute numbers; a peak of 42 in 2022 followed by a sharp drop to 3 in 2024.
   - **ACT** stays minimal (5 fatalities annually) until an outlier jump to 16 in 2022, then back down.
5. **Western Australia (WA)**
   - Flat around **mid-140s** for 2019–2021. (Later years not displayed.)

Key Insights

- The **COVID lockdown year (2020)** corresponds to noticeable dips in larger jurisdictions (NSW, VIC).
- **Post-2020 recovery** is evident, particularly in NSW and QLD, with 2023–2024 showing near-pre-pandemic levels.
- Smaller regions (ACT, NT, TAS) exhibit **high volatility** year-to-year due to lower absolute counts.
- **Intervention focus** might include NSW's sustained high fatalities and the pandemic's impact on traffic patterns.

7.2 Query H2: Top 5 Road-Types by Total Fatalities

```
MATCH (c:Crash)-[:HAS_ROAD_TYPE]->(rt:RoadType)
RETURN rt.name AS roadType,
       sum(c.num_fatalities) AS deaths
ORDER BY deaths DESC
LIMIT 5;
```

OUTPUT:

roadType	deaths
"National or State Highway"	3110
"Arterial Road"	2457
"Local Road"	2188
"Sub-arterial Road"	1767
"Collector Road"	816

Figure 7.2: Top 5 Road-Types by Total Fatalities

Discussion

1. **Highways as the greatest risk**
   – Crashes occurring on National or State Highways account for the largest share of fatalities (3 110), reflecting both higher speeds and traffic volumes on these corridors.
2. **Major urban arterials next**
   – Arterial Roads, which feed traffic into and out of city centers, rank second (2 457 deaths). These roads often combine high vehicle counts with frequent intersections, increasing conflict points.
3. **Local and sub-arterial roads still significant**
   – Local Roads (2 188) and Sub-arterial Roads (1 767) together contribute over 3 900 fatalities, underscoring that lower-speed, close-in networks are not immune to severe crashes—often involving vulnerable road users or junction collisions.
4. **Collector roads lowest among the five**
   – Collector Roads show the fewest fatalities in the top five (816), likely because they carry moderate traffic at controlled speeds and connect local streets to arterials.
5. **Implications for safety interventions**
   – While high-speed highways merit continued focus on speed management and roadside protection, substantial fatalities on urban roads point to a need for intersection redesign, safer pedestrian crossings, and traffic-calming measures on Local and Sub-arterial networks.

8. Graph Data Science Application

Use Case Predict high-risk LGAs for targeted safety interventions.

Algorithm

- **Node2Vec** embedding to capture connectivity patterns [8]
- **Random Forest** classifier to predict above-average fatality counts [9]

In this step, we compute each LGA's historical fatality average and assign a binary high-risk label based on whether it exceeds the global mean

Essential Steps & Code

```
// Compute Historical Averages & Label High-Risk LGAs

MATCH (l:LGA)<-[:IN_LGA]-(c:Crash)
WITH l, avg(c.num_fatalities) AS avgFatalities
SET l.avgFatalities = avgFatalities;

// Label LGAs as high-risk (1) or low-risk (0)

MATCH (l:LGA)
WITH avg(l.avgFatalities) AS globalAvg
MATCH (l2:LGA)
SET l2.highRisk = (l2.avgFatalities > globalAvg);

// Convert boolean to numeric label

MATCH (l:LGA)
SET l.highRiskInt = CASE WHEN l.highRisk THEN 1 ELSE 0 END;

// Project the LGA Graph for GDS

CALL gds.graph.drop('LGA_Network', false) YIELD graphName;
CALL gds.graph.project(
  'LGA_Network',
  'LGA',
  { CONNECTED: { orientation: 'UNDIRECTED' } },
  { nodeProperties: ['highRiskInt'] }
)
YIELD graphName, nodeCount, relationshipCount;
```

Explanation:

- avgFatalities: Stores the mean fatalities for each LGA.
- globalAvg: The overall average across all LGAs.
- highRiskInt: A numeric flag (1 if above globalAvg, otherwise 0), ready for downstream classification.

Build & Train a Classification Pipeline:

```
// Create a fresh pipeline

CALL gds.beta.pipeline.nodeClassification.create('LGA_Risk_Pipeline_v2')
YIELD name AS pipelineName;

//  Mutate in-memory graph with Node2Vec embeddings

CALL gds.beta.pipeline.nodeClassification.addNodeProperty(
  'LGA_Risk_Pipeline_v2',
  'gds.node2vec.mutate',
  {
    mutateProperty:      'embedding',
    walkLength:        100,
    walksPerNode:      50,
    iterations:        15,
    embeddingDimension:  128,
     randomSeed:        42
  }
)
YIELD nodePropertySteps;

// Select 'embedding' as the feature

CALL gds.beta.pipeline.nodeClassification.selectFeatures(
  'LGA_Risk_Pipeline_v2',
  ['embedding']
)
YIELD featureProperties;

// Add a Random Forest model step

CALL gds.alpha.pipeline.nodeClassification.addRandomForest(
  'LGA_Risk_Pipeline_v2',
  {
    numberOfDecisionTrees: 200,
    maxDepth:             20
  }
)
YIELD parameterSpace;

// Train the pipeline (80% train / 20% test)

CALL gds.beta.pipeline.nodeClassification.train(
  'LGA_Network',
  {
    pipeline:          'LGA_Risk_Pipeline_v2',
    targetNodeLabels:  ['LGA'],
    targetProperty:    'highRiskInt',
    modelName:         'LGA_RF_Pipeline_v2',
    trainFraction:     0.8,
    metrics:         ['ACCURACY','F1_WEIGHTED','F1_MACRO','OUT_OF_BAG_ERROR'],
    randomSeed:        42,
          overwrite:       true
```

```
  }
)
YIELD modelInfo
RETURN
  modelInfo.metrics.ACCURACY.test     AS testAccuracy,
  modelInfo.metrics.F1_WEIGHTED.test  AS testF1,
  modelInfo.metrics['F1_MACRO'].test    AS F1Macro,
  modelInfo.metrics.OUT_OF_BAG_ERROR    AS OOBError;

// Write Predictions Back to the Graph

CALL gds.beta.pipeline.nodeClassification.predict.write(
  'LGA_Network',
  {
    modelName: 'LGA_RF_Pipeline_v2',
    writeProperty: 'predictedHighRiskInt'
  }
)
YIELD
  preProcessingMillis,
  computeMillis,
  postProcessingMillis,
  writeMillis,
  nodePropertiesWritten,
  configuration
RETURN
  preProcessingMillis,
  computeMillis,
  postProcessingMillis,
  writeMillis,
  nodePropertiesWritten,
  configuration;

// Inspect Top Predictions

MATCH (l:LGA)
RETURN
  l.name                AS LGA,
  l.avgFatalities        AS historicalAvg,
  l.highRiskInt          AS actualLabel,
  l.predictedHighRiskInt AS predictedLabel
ORDER BY
  l.predictedHighRiskInt DESC,
  l.avgFatalities       DESC
LIMIT 10;
```

OUTPUT:

```
// 13.4.6 Train the pipeline (80% train / 20% test)
CALL gds.beta.pipeline.nodeClassification.train(
  'LGA_Network',
  {
    pipeline:          'LGA_Risk_Pipeline_v2',
    targetNodeLabels:  ['LGA'],
    targetProperty:    'highRiskInt',
    modelName:         'LGA_RF_Pipeline_v2',
    trainFraction:     0.8,
    metrics:           ['ACCURACY','F1_WEIGHTED','F1_MACRO','OUT_OF_BAG_ERROR'],
    randomSeed:        122,
  overwrite:        true
```

	testAccuracy	testF1	F1Macro	OOBError
1	0.60784314	0.5831696086678592	0.5277777728521947	{ "test": 0.3960674157303371, "validation": { "min": 0.38235294117647056 "max": 0.4472573839662447, "avg": 0.40874493257218497 } }

Figure 8.1: Classification Pipeline Performance on Test Data

Discussion

- The Random Forest built on Node2Vec embeddings achieves **~61% accuracy** on held-out LGAs, indicating it correctly flags high-risk vs. low-risk in roughly three out of five cases.
- The **weighted F1 (0.583)** reflects class imbalance—high-risk LGAs are less frequent than low-risk—while the **macro F1 (0.528)** shows that performance is moderately consistent across both classes.
- An **out-of-bag error of ~0.40** suggests there remains considerable room for improvement: tuning hyperparameters, incorporating additional features (e.g. demographic or traffic volume), or experimenting with alternative embedding configurations could boost predictive power.

```
// 13.6 Inspect Top Predictions

MATCH (l:LGA)
RETURN
  l.name                 AS LGA,
  l.avgFatalities        AS historicalAvg,
  l.highRiskInt          AS actualLabel,
  l.predictedHighRiskInt AS predictedLabel
ORDER BY
  l.predictedHighRiskInt DESC,
  l.avgFatalities       DESC
LIMIT 10;
```

LGA	historicalAvg	actualLabel	predictedLabel
"Nungarin"	2.0	1	1
"Exmouth"	2.0	1	1
"Sandstone"	2.0	1	1
"Irwin"	1.4285714285714286	1	1
"Hindmarsh"	1.3333333333333333	1	1
"Cassowary Coast"	1.3103448275862069	1	1
"Jerramungup"	1.25	1	1
"Corrigin"	1.25	1	1
"Dandaragan"	1.2222222222222223	1	1
"Dundas"	1.2000000000000002	1	1

Figure 8.2: Top 10 Predicted High-Risk LGAs

Discussion

- All of these top-10 LGAs have **actualLabel = 1** (i.e. above-average historical fatalities) and are correctly predicted as high-risk.
- They cluster in regions known for remote highways and sparse medical infrastructure (e.g., Nungarin, Exmouth), validating the pipeline's capacity to learn from connectivity patterns.
- This shortlist of LGAs can now be prioritized for targeted road-safety interventions—such as improved signage, speed-calming measures, or emergency response planning—to address their persistently elevated fatality rates.

Next Steps

To improve the model, we could:

1. Tune embedding and forest hyperparameters via cross-validation.
2. Introduce additional graph-based features (e.g. centrality measures).
3. Experiment with Graph Neural Network approaches in GDS for end-to-end learning.

These enhancements aim to boost recall on high-risk LGAs and reduce the overall error rate before deploying in a production setting.

9. Conclusion

This report presented an end-to-end solution: modelling, ETL, loading, and analysis in Neo4j. Queries answered all prescribed questions and two bonuses. The design supports scalability and future Graph Data Science extensions.

10. References

[1] Bureau of Infrastructure and Transport Research Economics, "**Road Deaths Australia – December 2024**," Dept. of Infrastructure, Transport, Regional Development, Communications and the Arts, Canberra, ACT, Australia, Statistical Bulletin, Dec. 2024. [Online]. Available: https://www.bitre.gov.au/statistics/safety/fatal_road_crash_database. [Accessed: 14-May-2025].

[2] Neo4j Inc**.,** ***Neo4j Operations Manual*****, ver. 5.26 (LTS)**. Neo4j, 2024. [Online]. Available: https://neo4j.com/docs/operations-manual/5.26/. [Accessed: 14-May-2025].

[3] Neo4j Labs, ***APOC Extended Procedures and Functions – Reference Manual*****, ver. 5** (library v5.26.0). Neo4j, 2024. [Online]. Available: https://neo4j.com/labs/apoc/5/. [Accessed: 14-May-2025].

[4] Neo4j Inc**.,** ***Graph Data Science Library – User Guide*****, ver. 2.13. Neo4j**, Jan. 2024. [Online]. Available: https://neo4j.com/docs/graph-data-science/2.13/. [Accessed: 14-May-2025].

[5] M. Robinson, "**Arrows: A web-based diagramming tool for property graphs**," Neo4j Labs, 2023. [Online]. Available: https://arrows.app. [Accessed: 14-May-2025].

[6] Python Software Foundation, ***Python Language Reference***, ver. 3.11. PSF, 2024. [Online]. Available: https://docs.python.org/3.11/. [Accessed: 14-May-2025].

[7] The pandas development team, ***pandas: Powerful Python Data Analysis Toolkit***, ver. 2.1, Jul. 2023. doi: 10.5281/zenodo.3509134.

[8] A. Grover and J. Leskovec, "**node2vec: Scalable feature learning for networks**," in ***Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min***. (KDD '16), San Francisco, CA, USA, Aug. 2016, pp. 855–864, doi: 10.1145/2939672.2939754.

[9] L. Breiman, "**Random forests,"** ***Mach. Learn.*****, vol. 45, no. 1, pp. 5–32**, Oct. 2001, doi: 10.1023/A:1010933404324.

[10] International Organization for Standardization, *ISO 8601-1:2019 – Date and Time — Representations for Information Interchange — Part 1: Basic Rules*. Geneva, Switzerland: ISO, 2019.

11. Table of Tables

12. List of Figures